UiT The Arctic University of Norway

Faculty of Science and Technology
Department of Computer Science

Guorrat

The Research and Development of a Real-Time System for Generation of Ball Positions in Football

Fredrik Stenvoll Nylund

INF-3990 Master thesis in Computer Science, 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

Abstract

In contemporary football, advanced analysis techniques are increasingly important, enhancing clubs' capabilities to make strategic in-game adjustments. Central to this advancement is the availability of real-time analytics, which relies heavily on accurate player and ball tracking technologies. While real-time player tracking has become more accessible, automatic and reliable real-time ball tracking remains a significant challenge. This challenge is particularly acute for lower-tier football clubs that often lack the necessary resources and infrastructure.

This thesis introduces Guorrat, a novel real-time system designed to generate accurate ball positions throughout live football matches. Developed to serve as a cost-effective alternative to existing real-time ball tracking technologies, Guorrat uniquely combines manual event tagging with real-time player positional data, which is generated by a system developed in parallel with Guorrat. This integration enables the system to produce precise ball location data during live matches, addressing the gap faced by resource-constrained clubs.

The system's effectiveness was evaluated in a series of experiments that demonstrated Guorrat's capability to provide frequent and precise ball position data in real-time. The results indicate that Guorrat is not only feasible but also a practical alternative for real-time ball tracking in football.

Acknowledgements

I would like to thank my supervisor, Dag Johansen, and co-supervisor, Martin Rypdal, for their invaluable guidance and support throughout this thesis. I would also like to send my gratitude to Tor-Arne Schmidt Nordmo, for his valuable help and feedback throughout this thesis.

Additionally, I want to thank my fellow students, William Alexander Stimpson-Karlsson and Børge Bårdsen, for all the discussion, support, and generally good times over the past 9 months.

I also send my gratitude to my family and friends, which has been a constant support throughout the duration this thesis.

Lastly, I would like to acknowledge the use of ChatGPT in various parts of this thesis. This tool was employed to improve the grammar, language flow, and overall readability of the thesis.

Contents

List of Figures

List of Tables

List of Abbreviations

ABR Adaptive Bitrate Streaming

API application programming interface

CSG Cyber Security Group

FPS Frames per Second

GPS Global Positioning System

HLS HTTP Live Streaming

HTML HyperText Markup Language

HTTP Hypertext Transfer Protocol

IMU Inertial Measurement Unit

IQR Interquartile Range

JSON JavaScript Object Notation

KB Kilobytes

LPS Local Positioning System

NRCR Non-Rectangle Click Registration

POC Proof of Concept

RMSE Root Mean Squared Error

RTMP Real-Time Messaging Protocol

TIL Tromsø Idrettslag

TUIL Tromsdalen Ungdom- og Idrettslag

UiT The Arctic University of Norway

URI Uniform Resource Identifier

VIK Viking Fotballklubb

xG Expected Goals

1

Introduction

Football, heralded as the most popular sport globally, captivates billions with its dynamic play and intense competition. The sport not only embodies a rich cultural significance but also commands vast economic stakes, with a substantial amount of money invested in club management and player performance. This financial aspect heightens the incentives for clubs to excel, compelling them to leverage every possible advantage to stay competitive. In this high-stakes environment, football analysis has become an indispensable tool, providing strategic insights that can significantly influence match outcomes.

Currently, top-tier football clubs invest heavily in comprehensive analytical capabilities, employing advanced technologies to gather and interpret game data. This disparity in resources leads to a competitive imbalance as smaller clubs, constrained by budget limitations, struggle to access similar analytical advantages. Despite the critical need for data-driven insights to enhance team performance and tactical planning, the existing solutions are often prohibitively expensive and complex, putting them out of reach for many clubs.

As technology advances, particularly in the fields of surveillance video systems, positional systems, and wearable sensors, the possibilities for analyzing team and individual performance continue to expand. As these technologies become more affordable and widely available, they pave the way for more cost-effective analysis systems. Such developments are crucial for narrowing the gap in analysis capabilities between top-tier and lower-budget clubs. This is the gap that this thesis aims to bridge.

1.1 Problem Definition

In modern football, the ability to analyze gameplay in real-time significantly enhances in-game decision-making and post-match strategy development. Current sensor-based technologies allow for extensive real-time positional tracking of players primarily using Global Positioning System (GPS) vests. However, similar advancements in sensor-based ball tracking technology is not accessible for most football clubs, for several reasons.

Technological advancements in the field of machine learning and object detection allows for sensor-less real time tracking of players and ball. However, such systems depends on expensive infrastructure investment to reliably detect and track the positions of both the players and ball, which acts as a barrier for most football clubs.

Therefore, most football clubs, especially those at lower-level with limited resources, lack access to affordable systems that provide both real-time and precise ball tracking. Consequently, these clubs miss out on crucial tactical and performance insight that could significantly influence their competitive performance.

This thesis carries out research and development of a Proof of Concept (POC) system that leverages generated player positional data to generate ball positions in real-time by utilizing manual event tagging during a live football match. This system should be able to function as an alternative to automatic real-time ball detection methods used today. To make this possible, an event tagging system must be created where users can tag ball related events, and the system utilizes this tagged data to generate the ball's position. The received player positional data is generated by another system, Sadji[51], that is developed in parallel with Guorrat.

The tagging system should use the generated player positional data received by Sadji[51] to innovatively overlay interactive rectangles over player outlines in the live video stream of a football match. These rectangles should contain the metadata about each player, such as their position on the field and their team affiliation, and can be clicked to register events, such as passes and tackles. Guorrat should then use the players field positions associated with these events to generate real-time ball positions.

Furthermore, the operational requirements for Guorrat should be minimal, enhancing its appeal for clubs of all sizes, particularly those with constrained resources. Primarily, to use Guorrat, it only requires a camera that can cover the entire field, which should be obtainable for most clubs. Additionally, it requires a computer to access the event tagging interface to generate ball positions. This

simplicity in setup eliminates the need for extensive hardware investments and complex integration processes. By leveraging existing infrastructure in this way, the system ensures that even clubs with limited technological capabilities or budgets can benefit from advanced analytics, making sophisticated and advanced analysis accessible without significant overhead costs.

Based on these points regarding the problem and how we attempts to solve it, the thesis statement is as follows:

> *This thesis will carry out research and development of an alternative to state-of-the-art real-time ball detection systems. This POC (Proof of Concept) will utilize player positional data in combination with manual event tagging to generate ball positions, with focus on real-time and precision properties.*

The name 'Guorrat' is derived from the Northern Sami language, meaning 'to track.' Given that this system is engineered to track the ball during football matches, 'Guorrat' was selected as an apt name.

1.2 Methodology

The latest report of the ACM Task Force[10] introduces a comprehensive framework for understanding computer science through three paradigms: Theory, Abstraction and Design. These paradigms represent distinct approaches to the discipline, highlighting it's breadth, depth, and interconnected nature.

Theory is the first paradigm. It is rooted in mathematics, and involves a four-step process critical for the development of coherent and valid theories within computer science. These steps are as follows:

- **Definition:** The initial step involves defining of the object of study, ensuring that there is no ambiguity about what is being analyzed.
- **Theorem:** After defining the objects, the next step involves hypothesizing potential relationships among them. These theorems propose how these objects might interact or relate to each other under certain conditions.
- **Proof:** The theorems are then subjected to rigorous proof to verify their validity. This involves logical reasoning and mathematical rigor to establish that the proposed relationships hold under the specified

conditions.

- **Results:** The final step involves analyzing the implications of the proofs and theorems. This often leads to further questions and the refinement of original definitions or theorems, thus iterating over the cycle to deepen and refine understanding.

Abstraction is the second paradigm. It is rooted in the experimental scientific method, and is essential for investigating phenomena within computer science. This paradigm also involves a four-step process, which is:

- **Hypothesis:** This stage involves formulating assumptions based on initial observations or existing theories. The hypothesis acts as a foundation for developing a predictive model.

- **Model:** Using the hypotheses, a model is constructed that simplifies the phenomena into a manageable framework, allowing for predictions and simulations.

- **Experiment:** : Experiments are then designed to test the predictions made by the model. This involves collecting data through various means, including simulations or real-world implementations, to observe how well the model's predictions align with actual outcomes.

- **Analyze:** The collected data is analyzed to evaluate the accuracy of the model. Based on this analysis, the model may be validated, refined, or even rejected, leading to new hypotheses and further cycles of abstraction.

Design is the third and last paradigm. This paradigm is rooted in engineering, and focuses on creating systems to solve specific problems. This is also a four-step process, which is as follows:

- **Requirements:** This initial phase involves defining the precise needs and constraints of the system to be developed. It sets out what the system is supposed to achieve and under what conditions it must operate.

- **Specification:** Based on the requirements, a detailed specification of the system is developed. This includes the architecture, components, interfaces, and data flows of the system, providing a blueprint for implementation.

- **Design and Implementation:** With specifications in place, the ac-

tual system design and implementation occur. This phase involves both high-level architectural design and detailed programming, ensuring that every aspect of the specification is translated into a functional system.

- **Testing:** Once implemented, the system is rigorously tested against the requirements.

This thesis research is rooted in the last paradigm, Design. First, the benefits and downsides with the existing real-time ball detection methods will be explored. Based on this, a requirement specification for Guorrat will be devised, and a system will be designed and implemented bases on the specified requirements. Next, the POC will be evaluated experimentally, with primary focus on the non-functional properties of real-time and precision.

1.3 Scope and Limitations

The operation of the system relies on the premise that the metadata used is produced by an external source. The system that generates this metadata is Sadji[51], which has been developed in parallel with Guorrat. For Guorrat to function correctly, it necessitates a mutual agreement on the metadata format between both parties.

Since this is a POC system, and the development focus has been on the unique functionalities of my Guorrat, some common non-functional requirements typically associated with computer systems, such as security, have not received as much attention.

Guorrat is currently designed and implemented specifically for football. However, with some adjustments, it could be adapted for use in other similar team sports like basketball or hockey.

1.4 Research Context

This thesis has been produced under the guidance of the Cyber Security Group (CSG) at the The Arctic University of Norway (UiT). CSG is a research group focused on advancing knowledge and developing innovative technologies regarding distributed systems within the field of computer science. CSG is dedicated to conducting impactful research and innovation that spans multiple disciplines and faculties, residing at the crossroads of computer science and

various other scientific domains.

Its overarching aim is to forge new insights, research methodologies, and groundbreaking technologies in the evolving domain where computer science is central. By tackling complex, real-world problems, CSG combines academic principles with practical innovations and applications. CSG conducts research in experimental computer science with a focus on the design, architecture, and implementation of distributed systems that are scalable, efficient, fault-tolerant, privacy-preserving, compliant, and secure.

A key collaborator of the Cyber Security Group is the Corpore Sano Centre. This center is dedicated to pioneering research and innovation within the life sciences, bridging the disciplines of computer science, sports science, and medicine. Its interdisciplinary efforts focus on enhancing elite sports performance and injury prevention, advancing preventive healthcare, conducting large-scale population screenings, and undertaking epidemiological health studies.

CSG has been in the Artificial Intelligence domain for a while, with systems such as StormCast[17]. StormCast is a distributed artificial intelligence application designed for severe storm forecasting, employing a hierarchical architecture that enhances cooperation among distributed modules to collect, process, and synthesize weather data efficiently across different geographical areas. This system enables scalable, fault-tolerant operations by utilizing a combination of expert and transmission modules to predict and communicate severe weather conditions effectively.

In the sports science domain, CSG is known for research and development of several systems around soccer, e.g. Bagadus and Muithu. Bagadus[16] is an integrated sports analysis system combining camera arrays, the ZXY Sport Tracking system[53], and human annotations to enhance game performance analytics. Bagadus uniquely integrates real-time player tracking with ZXY sensors, providing precise player positions, orientations, and biometric data at high frequencies. This data facilitates the automatic extraction and playback of specific game events, such as sprints, directly from statistical data.

Muithu[22] is a sports notational analysis designed to integrate real-time coach notations with video sequences during sports practices and games, using economical, off-the-shelf components that require minimal human post-processing. This system allows coaches to annotate metadata to an event on their smartphone during match or practice, and Muithu automatically finds the associated video sequence and makes it available to watch at half-time or after the match or practice session.

CSG does also have have experience in the physical performance part of professional football, with several studies on the aspect of physical performance in professional women's football[3][38][58]. These studies, along with Muithu and Bagadus, are some examples that demonstrates CSG's experience in the domain of professional football, and how it can be integrated with state-of-the-art technology.

1.5 Outline

The content and structure of this thesis is as follows:

Chapter 2 explores relevant background information regarding both football analysis and automatic ball tracking, some existing analysis systems used by clubs today, and explains technologies used in the implementation of Guorrat.

Chapter 3 describes the functional and non-functional requirements of Guorrat.

Chapter 4 covers the design and implementation of Guorrat, and discusses the decisions made in regard to its design and implementation.

Chapter 5 details how Guorrat has been evaluated, exploring both the various experiments and their results.

Chapter 6 is used to discuss different aspects of Guorrat, and how it fulfills its functional and non-functional requirements.

Chapter 7 concludes and summarizes this thesis, in addition to discussing some possible future work.

2 Background

This chapter will explore relevant background information regarding football analysis, and dive deeper into real-time automatic ball tracking and why it is not suitable for widespread use. Additionally, it will evaluate some existing analysis systems, with primary focus on their real-time ball tracking and event-tagging properties.

2.1 Football Analysis

Detailed football analysis has become increasingly common in the modern game for several compelling reasons. Primarily, the potential for competitive advantage is perhaps the most compelling reason for most teams. In football, where games are frequently decided by one or two goals, even minor edges can be the difference between winning, drawing, or losing. With real-time advanced analysis, teams gain the critical opportunity to make strategic adjustments on the fly, potentially transforming a likely defeat into victory.

It is not difficult to see how every small advantage gained through detailed analysis can provide the critical boost needed to tip the scales in favor of a team.

Another important factor for the rise of football analysis, comes from the large amount of money present in football today. Elite football clubs in today's world

are regularly spending tens to hundred of million euros on new players, the most extreme being Paris Saint German's acquisition of Neymar from Barcelona, for the amount of 222 million euros[54]. An incentive for teams to invest and focus on analysis, is to get a more solid foothold when evaluating possible prospects. For instance, while the majority of people might evaluate a striker primarily based on goals and assists, a team's analytics staff will likely delve into more nuanced statistics, such as shot conversion rate or off-ball positioning. These deeper metrics might help determine whether a player's value justifies their potential transfer fee, or if their performance is largely a result of the team's overall dynamics, and could allow a team to find potential cheaper players that they believe is an equal or better fit to the team's style of play.

One notable instance of analytics influencing player transfers occurred in 2017 when Liverpool FC acquired Mohamed Salah from AS Roma[12]. A key factor in their interest was the detailed analysis by Liverpool's head of analytics, Ian Graham, whose models highlighted Salah's exceptional off-ball movement and superior Expected Goals (xG) metrics. This analytical insight was pivotal, as Salah's subsequent performances at Liverpool FC significantly exceeded expectations, becoming the Premier league's top scorer 3 times[39].

Broadly, football analysis can be divided into two main parts: tagging and analysis. The tagging part involves annotating metadata to specific football video sequences and events, while the analysis part uses this annotated metadata to create analytical metrics and tools. The focus of this thesis is primarily on the real-time tagging aspect of football analysis, and we will explore the most common methods of tagging in the following sections.

2.1.1 Event Tagging

A way of generating analysis data is by event tagging, or annotating of different events during a match or practice. This can be done in a wide range of complexity, from a coach on the sideline writing physical notes, to analysts using a fully-fledged tagging software program, to outsourcing the analysis to external companies. There are numerous companies around the world that specializes in the tagging of football matches[45, p. 7]. Many of these companies employ hundreds or even thousands of workers, each responsible for recording and annotating a wide range of events during the games. These events include standard actions like passes, shots, and tackles, as well as more detailed and specialized ones, such as noting the height of the ball when a player receives a pass. Once a match concludes, the footage is sent to one of these companies, and within 12-24 hours, a comprehensive set of analysis data is returned.

2.1.2 Tracking Data

Another method of generating metadata for analyzing, is by positional tracking. Both Global Positioning System (GPS) and Local Positioning System (LPS) can be used to measure data about players and ball. Measurements such as a player's position, total distance covered, and high-intensity runs, can be used to perform both strategic and performance analysis. STATSports' GPS tracking vests[50] are used by many of the highest ranking clubs in the world, including the local team Tromsø IL (TIL). Bagadus, mentioned in section 1.4, uses a LPS, the ZXY tracking technology[53], to track the position, movement and certain biometrics of the players on the field.

2.1.3 Video Analysis

One method of analysis involves reviewing recorded video footage of matches or practice sessions. By studying this footage, coaches, players, or analysts can identify strengths and weaknesses in various scenarios. The video can reveal useful patterns applicable to the team as a whole, specific groups of players, or individual players. Players will often find it hard to assess their own performance and decision-making due to their position on the field and the fast pace of the game. Video analysis provides a comprehensive view of each situation, enabling players to critically evaluate their actions and, if necessary, make improvements.

The various analytical methods are frequently integrated and considered collectively to achieve a comprehensive and detailed analytics overview of a match or practice session. For example, when analysts tag events during a match or practice, they use timestamps for each event to swiftly access the corresponding video footage. This footage can then be examined in greater detail, providing a complete picture of the events.

2.2 Automatic Ball Detection

To enable automatic real-time advanced football analytics, the ability to accurately locate and track the ball is crucial. When both player and ball positional data are available, the depth of the analysis is increased and a variety of football-specific analysis can be conducted. The effectiveness of these analyses largely depends on the quality and precision of the tracking of both the players and the ball. If only player positions are tracked, and not the ball position, the number of possible metrics and statistics diminishes significantly, resulting in a lower overall value of the analysis.

Today, many elite football clubs utilize GPS tracking to monitor player movements, often using GPS vests worn under their jerseys. This technology provides teams with precise positional data for each player. For instance ,TIL, playing in Norway's top division, Eliteserien, employs this method. However, although Eliteserien uses a standardized ball for all matches, this ball lacks any integrated tracking technology, making it impossible to gather real-time positional data for the ball for analysis purposes.

Following, in section 2.2.1 and 2.2.2, the two main methods of automatic real-time ball detection will be explored and investigated.

2.2.1 Sensor Based-Ball Tracking

Sensor-based ball tracking in soccer utilizes sensors within the ball to capture real-time data on its movement and location during play. These systems typically use external tracking systems to determine the position of the ball, mainly by utilizing a LPS[15][25].

One example of such a technology being used is during the FIFA 2022 World Cup in Qatar, when advanced ball tracking technology was introduced into the tournament. Adidas developed the official ball, which included a lightweight motion sensor by KINEXON[25] inside. This sensor provided real-time data on spatial positioning and movement through an Inertial Measurement Unit (IMU) that utilizes ultra-wide band (UWB)[24] radio frequency to broadcast the information to KINEXON's LPS, which is installed around the pitch. The gathered data was used for analysis, assisting in the new semi automatic offside detection system used in the World Cup, and for real-time match statistics for a better viewing experience for the audience.

This technology allows teams to use the provided ball data, in combination with their own sensor- and/or optical based player tracking data, to perform performance analysis and make strategic decisions in real-time. In 2022, this kind of system was tested during a relegation match in Liga Portugal[26]. Here, more than 300 different analysis metrics were tracked by using KINEXON's ball sensor in combination with tracking vests on the player.

Although this technology is accessible to the world's most elite clubs and national teams, it is unlikely to be available to most of the world's clubs anytime soon. Several factors contribute to this limitation, including the budget and infrastructure required to support such a system. Another constraint is the use of standardized balls in certain leagues, where this technology has not been implemented. For instance, as discussed in section 2.1.2, TIL employs GPS vests for player tracking. However, since the Norwegian top division, Eliteserien, uses

a standardized ball, it is not feasible to integrate player positional data with ball sensor data for advanced real-time analysis. Given that Guorrat aims to be accessible to a broad range of football clubs, including those with limited budgets, an alternative to this technology was necessary.

2.2.2 Object Detection-Based Ball Tracking

Object Detection-based ball tracking in football leverages advanced Object detection algorithms to accurately locate and track the ball during gameplay[11][27][28][36]. Object detection is a crucial technique within the realms of computer vision and image processing, which typically utilizes Machine Learning (ML) and Deep Learning (DL) models to identify and classify objects within images or video frames.

Modern object detection algorithms, such as YOLO[56] (You Only Look Once), Faster R-CNN[41] (Region Convolutional Neural Networks), and SSD[29] (Single Shot MultiBox Detector), have significantly improved the precision and speed of detecting and tracking objects, including footballs, in real-time applications. These algorithms work by generating bounding boxes around detected objects and classifying them with a certain confidence level.

The various object detection algorithms differ in their speed and accuracy. YOLO is faster than Faster R-CNN but generally provides lower accuracy in detections, particularly for small objects, a ball for instance. Faster R-CNN employs a more complex two-step process, the first of which involves identifying regions of interest across the image. These regions are then classified by a Fast R-CNN detector. In contrast, YOLO and SSD utilize a faster one-step process. These trade-offs illustrate why accurately and reliably detecting and tracking the ball in real-time with object detection can be challenging.

From the outset of developing the player detection and tracking system utilizing the YOLO object detection algorithm, used by the external source, Sadji[51], to generate the metadata Guorrat utilizes, it became apparent that ball detection and tracking with object detection would pose a significant challenge. Other systems that uses YOLO to detect the ball with more success typically uses better video sources, such as TV-images[11] or multiple cameras[36]. Since this system should only rely on a single camera, effectively detecting and tracking the ball proved unfeasible.

Factors such as the ball's small size, rapid change of directions, and fast movement makes the process of detecting and tracking the ball via object detection challenging. With the importance of controlling the ball in football, the ball will regularly find itself close to and in between players. This leads to cases where

either the ball is occluded by a player, or the detection algorithm struggles to separate the ball from a player. In addition, there are several factors not related to the game itself that make this process even more challenging, like the video quality, camera placement, lightning conditions, etc.

For instance, the camera used on TIL's home field, Rommsa Arena, is placed quite low relative to the field, making object detection of players more challenging, and especially ball detection. In contrast, other systems that more successfully track the ball using object detection typically depend on additional video inputs, such as TV broadcasts or specialized cameras. However, since the aim of this system is to be accessible to football clubs of all levels, it cannot depend on TV footage, highly-optimized infrastructure, or specialized cameras for its functionality.

2.3 Existing Systems

In this section, we will explore some of the existing systems that are used by teams today, and evaluate if and how these system can enhance real-time analysis, with a focus on ball-related events. The reason these existing systems have been selected, is that they are either used by two of the top teams in Tromsø, that both have relations with UiT and CSG, or by one of the top teams in Premier League in England.

2.3.1 Hudl

Hudl[18] is a sports performance analysis tool that has become widely used by teams and coaches across various sports disciplines, including football, basketball, American football, and many others. Founded in 2006, Hudl has grown to become a leading platform in the sports technology industry, offering a range of services aimed at improving team performance and individual player development through detailed analysis and insights. Hudl is currently in use by (TIL), as well as the entire Eliteserien.

Hudl software can be integrated with Hudl's own cameras, Hudl Focus[20]. These cameras are installed to cover the field and is fully automatic, removing the need for any person manually filming.

Hudl Assist

Hudl Assist[19] is a feature offered by Hudl that significantly enhances the video analysis process for teams and coaches, by leveraging Hudl's team of professional analysts and sophisticated software to tag and break down game footage. It provides teams with a more efficient and less labor-intensive method for analyzing game footage, by outsourcing the analysis.

While Hudl Assist delivers advanced analysis with the position of each ball-related event, a notable limitation is the absence of real-time analysis capabilities, as Hudl Assist necessitates the uploading of a complete match, followed by a waiting period of 12-24 hours before the analysis is complete. Additionally, by outsourcing the tagging with such a system, it makes it harder to verify the precision and quality of the analysis data.

Hudl Manual Tagging

Hudl offers a manual event tagging system[21] that allows users to annotate match footage themselves. This feature enables users to upload videos to Hudl's platform and manually label various match events, such as passes, shots, and crosses, including the outcomes of these actions. The interface for this manual tagging is shown in Figure 2.1.

Hudl's manual tagging system does not provide the user with the ability to register precise positional data regarding events. As the user can only select in which third of the field the event occurred, the positional context for an event is not precise enough to perform any meaningful advanced analysis.

Figure 2.1: Screenshot of Hudl's manual tagging interface [21].

2.3.2 Spiideo

Spiideo[47] is a Swedish technology company that specializes in providing sports video analysis and performance evaluation tools. Spiideo is currently used by the local team Tromsdalen Ungdom- og Idrettslag (TUIL). TUIL is currently playing in the third highest professional division in Norway.

Spiideo Perform

Spiideo Perform[48] is a cloud-based sports video analysis tool that provides users with game and player analysis. For a team to use the Spiideo Perform software, a Spiideo sports camera must be installed.

Spiideo Perform provides users the ability to manually tag a wide range of events, both live and after a match is finished, with the use of their customizable tag panels, as seen in Figure 2.2. However, while Spiideo Perform allows for real-time event tagging, it does not provide the user any way of adding positional context or ball position to the tagged events.

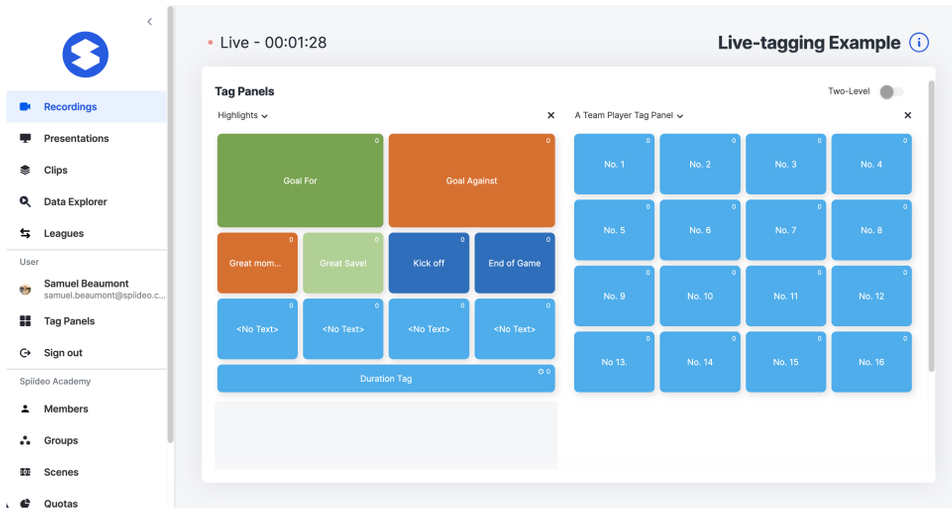

Figure 2.2: Screenshot of Spiideo Perform's tagging interface [49].

AutoData Soccer LIVE

Spiideo Autodata Soccer LIVE[46] uses player tracking technology to automatically detect and capture game event data. This data includes events such as goals, shots and corners, in addition to individual player statistics, such as high-intensity runs, distance covered, and running speed. AutoData soccer LIVE can be paired with Spiideo Perform to create a more detailed analysis, where AutoData Soccer LIVE can provide positional context to manually tagged ball-related events in Spiideo Perform.

As experienced on this project, reliably detecting the ball's position with a single camera in a real-time fashion is quite the challenging task. Spiideo claims on their website that AutoData Soccer LIVE is a fully automatic system, requiring no human intervention. However, from the information presented on Spiideo's website, it is not possible to validate the correctness of their generated data. While such a system might work well for higher-budget clubs with good facilities for automatic ball detection, it is hard to imagine it working well for most clubs. Anecdotal experience has shown that AutoData Soccer LIVE fails to detect most ball related events.

2.3.3 Nacsport

Nacsport[32] is an analysis platform, used by for instance Liverpool FC. Nacsport provides users with a tagging interface to tag a large variety of events. These tagging interfaces can be customized however a user wants, allowing users to

choose which events they want to register. Nacsport allows for tagging of both live streams and uploaded recordings of football matches.

Nacsport Enhanced Graphic Descriptors[33] is a tool provided by Nacsport that can be utilized in tandem with the tagging interface. This tool allows users to add precise positional data to ball-related events, by having a clickable 2-d field in the tagging interface. An example of a Nacsport tagging interface is visualized in figure 2.3. This results in having a registered ball position at the time of start and end of each event, but does not result in a continuous ball position throughout the match, which can enhance the analysis significantly.

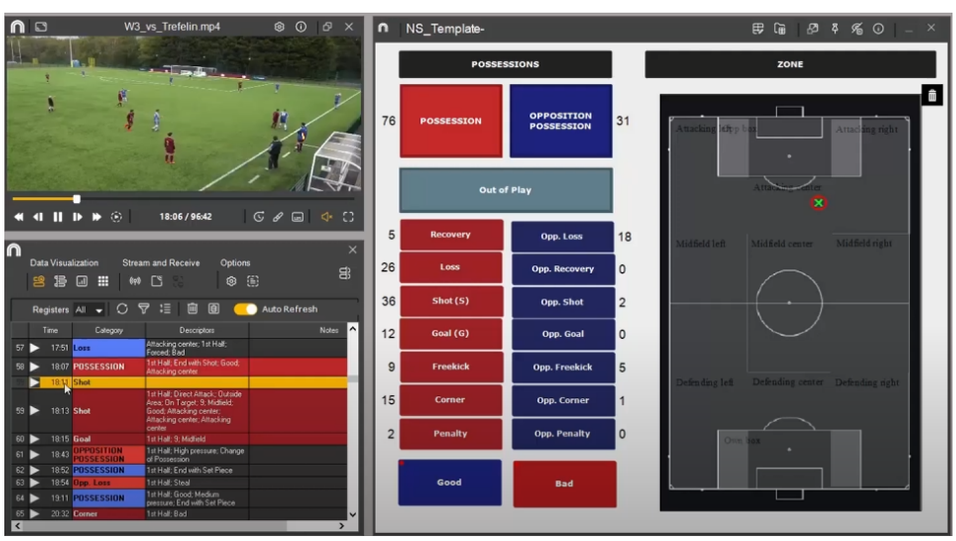

Figure 2.3: Screenshot of a Nacsport tagging interface from their official YouTube channel[34]

However, as the Enhanced Graphic Descriptor tool is only available for some of the higher tier Nacsport packages, costing a yearly minimum of 595 euros[35] as of the time of writing this thesis.This tiered access to features creates a significant barrier for users with limited financial resources, such as smaller clubs or independent analysts, who might not afford the higher-tier packages required for more detailed analysis.

2.3.4 Summary of Existing Systems

In the previous section we have explored and evaluated some existing analysis systems that provide manual or automatic event tagging in football, with a focus on their real-time capabilities and the precision of ball-related generated analysis data. Both the manual tagging systems of Hudl and Spiideo lacks

precise positional data associated with the tagged events. For Nacsport, it does enable real-time tagging with positional context, but it does not provide continuous ball positions throughout the match.

While Hudl Assist provides detailed analytics with precise positional data, it lacks in real-time performance, as the returned analysis is available 12-24 hours after the match has ended. In addition, the precision of the generated analysis data can be hard to verify. Spiideo AutoData Soccer LIVE is the opposite of Hudl Assist on the real-time performance and precision scale. While AutoData Soccer Live generates analysis data in real-time, the precision of the generated ball-related data is questionable, especially for resource-limited clubs with subpar infrastructure.

This demonstrates that there are no existing systems that can automatically provide real-time and precise ball positions and ball-related events for football clubs at all levels, especially those with limited budgets and infrastructure.

2.4 Technical Background

In this section, some of the technologies and concepts used in Guorrat is explained.

2.4.1 HTTP Live Streaming

HTTP Live Streaming (HLS) is the protocol used to live stream football matches in Guorrat, with that decision being discussed later in Section 6.2. HLS is an adaptive bitrate streaming protocol developed by Apple[2]. It is widely used for streaming of live and on-demand video content over the internet. HLS works by breaking down a video stream into a sequence of shorter HTTP-based file downloads, each containing one short chunk of the overall stream. This method allows the video player to download each segment and play it almost immediately, which facilitates streaming of content without the need for a specialized streaming server. The central parts of HLS is:

Segmentation: The original video content is divided into a series of short segments, usually with a duration of 2 to 10 seconds each. These segments can be encoded at various quality levels to support different bandwidth conditions.

Index File: An index file, or playlist, is created in the form of an M3U8 file. This file contains metadata about each segment, including URIs pointing

to each segment file and information about the sequence and duration of each segment. For adaptive streaming, multiple index files for different quality streams are created.

Delivery: Both the segments and the index file are served over standard HTTP web servers. The client player reads the index file to understand the sequence of video segments to play. As the video plays, the client may switch between different quality streams by choosing a different index file, adapting to changing network conditions to maintain smooth playback.

Adaptive Bitrate Streaming (ABR): HLS supports adaptive bitrate streaming, allowing clients to automatically select the most appropriate bitrate based on current network conditions. This ensures an optimal balance between video quality and buffering.

HLS has support for a wide range of devices and platforms, including iOS devices, Android devices and most web browsers. HLS is also relatively easy to deploy and scale without requiring specialized infrastructure, since it uses standard HTTP servers for delivery.

2.4.2 Homography Transformation

A homography transformation is a computer vision transformation tool that defines the relationship between two images of the same planar surface viewed from different perspectives. It is used in Guorrat to translate a coordinate from the video screen of a football match to the corresponding coordinate on the physical football field. It facilitates the translation of coordinates from one image plane to another while keeping the straight-line structures in the scene despite perspective distortions.

Homography transformations use a 3x3 matrix to represent the transformation between two images with different perspectives in a homogeneous coordinate space. The homography matrices used by Guorrat is calculated by the external source, Sadji[51], and is retrieved along with the player positional metadata. How the mapping of a coordinate is transformed using homography is visualized in Figure 2.4.

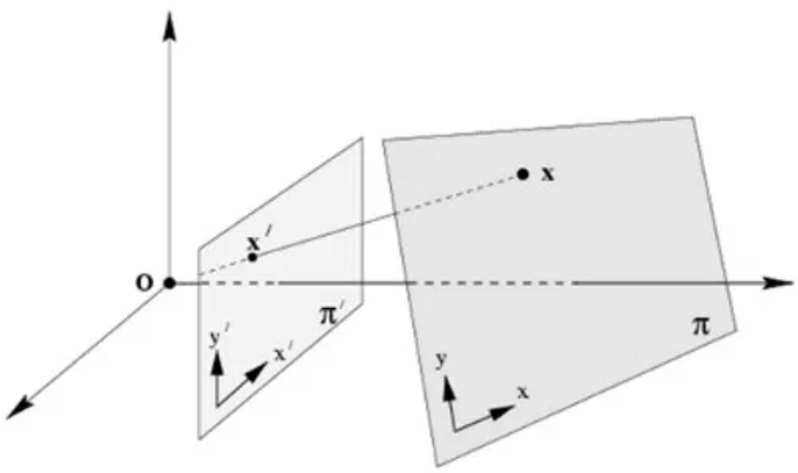

Figure 2.4: Visualization of a homography transformation between two different perspectives. Source: [30]

2.5 Summary

In this chapter, we have explored football analysis and how automatic real-time ball detection is usually performed, and explained the reasons why these methods is not available for most football clubs. Additionally, we have investigated and evaluated some of the similar existing system, where it is clear that these systems is not capable of tracking the ball reliably in a precise and real-time manner.

3

Requirement Specification

Guorrat is intended to generate real-time and precise ball positions during a football match by utilizing manual tagging in combination with player positional data generated by another system, Sadji[51]. For Guorrat to accomplish this, a live streaming server and manual tagging system must be in place. In this chapter, the functional and non-functional requirements, in the context of fulfilling this goal, is specified.

3.1 Functional Requirements

In this section we will look at the functional requirements of Guorrat, which is the specific features and capabilities that Guorrat should provide to fulfill its intended purpose.

3.1.1 Generate Real-Time Ball Positions

The pivotal function of Guorrat is to generate real-time precise ball positions during a live football match, with the use of generated player positional data from the external source, Sadji[51]. Guorrat should utilize manual tagged ball-related events, such as passes and tackles, to derive the position of the ball during these events with the highest possible precision.

When a ball-related event is manually tagged by the user, the system should record the involved players position on the field to derive the ball position during that event. Guorrat should be able to derive if the same player has possessed the ball between two events, by evaluating if the same player is involved in two consecutive event. If Guorrat detects a possession, it should be able to generate ball positions during the possession by recording the positions of the involved player during the possession. In Figure 3.1, Player 2 receives the ball in Event 1 and passes it on during Event 2, and the system should therefore derive that Player 2 has possessed the ball between events 1 and 2, and thereby generate the ball positions during that phase of play.

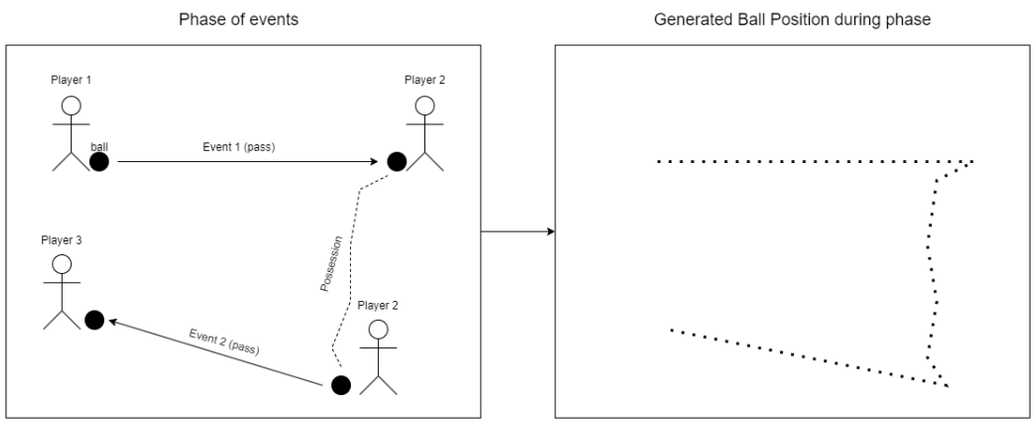

Figure 3.1: Demonstration of how Guorrat should use tagged events to determine the position of the ball during these events.

3.1.2 Retrieve Video Stream and Metadata from External Source

If Guorrat is to be able to provide a live stream of a football match along with an interface to tag events in the context of player positional data, the system should be able to connect to the external source, Sadji[51], to retrieve the HLS video stream segments along with each associated metadata segment.

3.1.3 Provide Live Stream of Match with Synchronized Player Rectangles

Guorrat should be able to provide a video live stream of a football match. The video stream should also display a rectangle outline of each player, based on the received metadata from the external source, Sadji[51]. The video stream should provide the user with a simple and responsive user interface for video

controls, including play/pause, rewind, and skip, to accommodate a good-flowing manual event tagging system.

3.1.4 Handle Event Tagging

While viewing the live match stream, a user should have the ability to click on the rectangles representing two different players to record a ball-related event as it occurs. Following this, the user can select from various event types, such as a pass or a tackle. The specific events available for selection should vary depending on whether the two selected players are teammates or opponents. The system should keep track of all registered events, capturing details such as the sender and receiver, the timing of the event, its success or failure, and other relevant information needed to generate ball positions.

3.1.5 Track Names of Players

In addition to allowing generation of ball positional data, the tagging functionality of Guorrat also registers the ball related events themselves, to complement the generated ball position data for more advanced analysis. To obtain more detailed event data, the user has the option to select the name of the players associated with each player rectangle involved in an event. Guorrat should attempt to track the selected player name for each player rectangle, to make the event tagging process more efficient.

3.1.6 Display Generated Ball and Event Tagging Data

A user should be provided an interface with the display of all generated ball positions and registered ball-related events, to immediately evaluate and validate the quality of the ball positions and event data. Regarding the tagged event data, it should be possible for the user to filter these events based on team, players, success and time of event.

3.2 Non-Functional Requirements

The section on non-functional requirements is critical for outlining the criteria that define the operational characteristics of Guorrat. This requirements regard properties such as real-time performance, the precision of the system's generated output, its availability, and other properties not directly connected to the unique features of Guorrat.

3.2.1 Real-time Performance

A key requirement for Guorrat is its capability to function in real-time. This means it should be able to retrieve and stream a live football match as it unfolds, while also enabling users to tag events as they occur to generate ball positions. As Guorrat requires manual input from a user to tag events, the speed at which a match is fully tagged ultimately depends on the user's efficiency. However, the ball positions and event data generated from these tags should be available immediately once the events are registered.

3.2.2 Precision

A crucial aspect of Guorrat is the precision of the output data it generates. For users to rely on this system, it is essential that they can trust the accuracy and quality of the data produced. If the ball positional data generated by this system is intended for use in conjunction with other analysis platforms, the correctness of this data must be maintained at the highest possible level.

3.2.3 Usability

A non-functional requirement for Guorrat is Usability. It is critical that the system is designed and implemented to be simple and accessible for all intended users. The system should enable users such as analysts and coaches, irrespective of their technical expertise, to efficiently navigate and utilize the system's features. During the development process, Guorrat will be exposed to coaches with experience in the domain, to receive feedback.

3.2.4 Availability

Given the critical need for continuous user access during a match, the design and implementation of Guorrat should be developed with a focus on the non-functional requirement of availability. Guorrat must be available for users at the time of the live match. Additionally, as Guorrat's function depends on the generated metadata from the external source, Sadji[51], Guorrat's availability will ultimately depend on the availability of this external source.

3.2.5 Reliability

Guorrat should be reliable, and should operate consistently over time. It is important that the system performs its intended functions without failure,

maintaining a high level of service and reliability even under heavy use.

3.2.6 Maintainability

Guorrat should be designed with a strong emphasis on maintainability, ensuring that both ongoing maintenance and future development are straight-forward for new developers. It should feature a logical and well-structured architectural design, complemented by clearly organized and comprehensible source code, to make it easier for potential new developers to work on the system.

3.3 Summary

In this section, we have explored the functional and non-functional requirements needed to ensure that Guorrat works as intended. Guorrat should be able to generate real-time and precise ball positions, by providing users with an event tagging system for a live football match. It is crucial that Guorrat performs in a real-time fashion, and with the highest possible precision regarding the generated ball positions. In the next chapter, it will be explained how Guorrat is designed and implemented to fulfill these requirements.

4

Design & Implementation

In this chapter the system architecture of Guorrat will be presented. Additionally, it will explore the design and implementation of Guorrat, in the context of the functional and non-functional requirements specified in the previous chapter. The chapter will explain how Guorrat's event tagging functionality is designed, and how it is used to generate precise ball positions in real-time.

4.1 System Architecture

The overarching architect of Guorrat consist of a frontend (1), backend (2) and a data storage (3). Additionally, it connects to an external source (4) to fetch the live video stream of a match along with the generated metadata. This external source, Sadji[51] is developed in parallel with Guorrat, and is responsible for generating the metadata, which contains data regarding each player, used by Guorrat. A visualization of the system architecture is displayed in Figure 4.1.

The role of the frontend (1) is to retrieve video stream and metadata and synchronize it (1.1), and provide users with a real-time event tagging system (1.2), as well as providing an interface for displaying the generated ball and event data (1.3). The backend server is responsible for retrieving, processing and storing the video stream (2.1) and metadata files (2.2) from the external

source[51], in addition to handling tagged events (2.3) to generate ball positions. The data storage (3) stores the metadata files (3.1) and the video stream files (3.2).

In the following sections, these different parts of Guorrat will be explained in further detail.

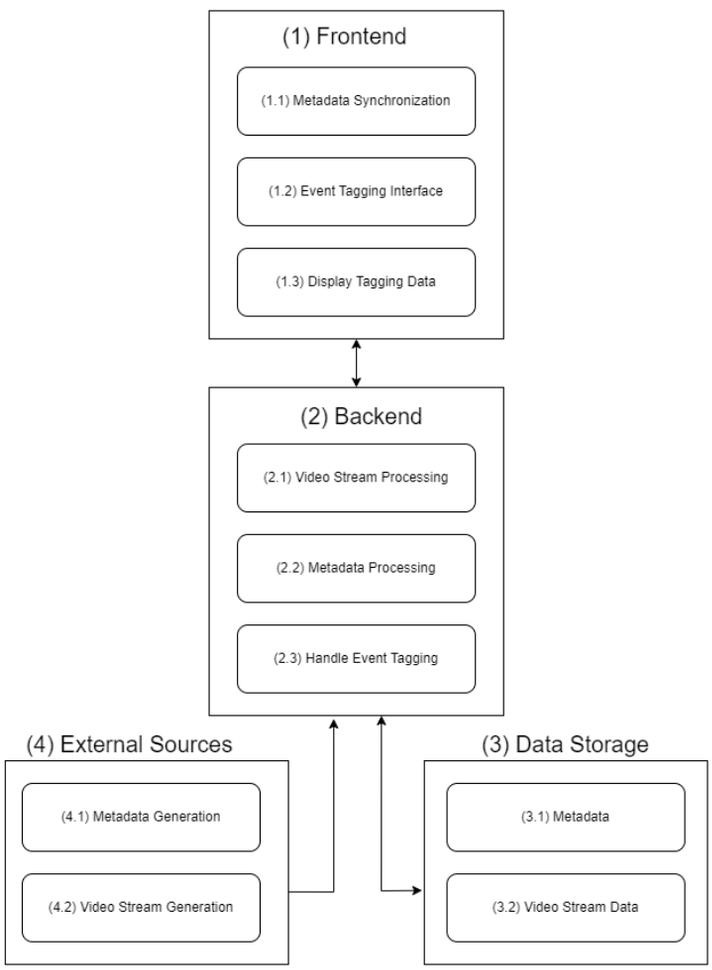

Figure 4.1: Overview of the system architecture for Guorrat.

4.2 Frontend

The frontend's main responsibility is to synchronize the received video stream and metadata, and provide the user interface of the event tagging functionality needed to generate ball positions. In addition, the frontend provides a

visualization of the generated ball-positions and event data.

For the implementation of the frontend server, React[40] was selected as the framework. The decision was influenced by my familiarly with React and the high level of support because of its large community, which ensures a smoother and more efficient development process. React comes with a wide range of available external libraries, to speed up and simplify the development process. On important React library for this system is the 'HLS.js'[57] library, facilitating the streaming of HLS video formats directly in the web browser.

TypeScript[55], a programming language built on JavaScript, is chosen as the programming language for the frontend server. The main reason for this is its strict type safety, which increases the maintainability and reliability of Guorrat.

4.3 Backend

The backend server has two main responsibilities. The first is to fetch the HLS video stream files and metadata files from the external source, Sadji[51], process it, and serve it to the frontend when requested. For each video stream segment, the external source provides a corresponding metadata file. Each video segment has an associated metadata segment which contains the detections for each frame for that video segment. All video stream and metadata file pairs are associated by a segment number present in their filenames. When the frontend server requests a specific video segment, it will send a request for the corresponding metadata file as well, and the backend server will then respond with the requested metadata.

The second responsibility of the backend server is to process and store the tagged event data received from the client. This includes handling tagged events, determining and processing possession, and use this data to generate ball positions throughout the match.

The backend server for this system is developed using the Rust programming language. The selection of Rust was made after evaluating several potential programming languages, each with its own set of strengths and suitability for the project's requirements.

Rust was chosen for its outstanding performance capabilities [42], which are essential for meeting the real-time non-functional requirement of the system. The performance capabilities of Rust also makes it possible to support a larger number of concurrent users effectively, which directly contributes to enhancing

the system's ability to scale well. Additionally, Rust's strict type[43] and memory[44] safety play a crucial role in boosting the reliability and availability of the system. The Rust ecosystem also provides a wide range of different libraries, so-called crates, that provides reusable code for different functionalities, to increase the efficiency of the development process.

Other languages considered included Python, Go and C. Python is well-known for its rapid development capabilities and extensive support libraries, alongside strong community backing. However, its performance and memory safety metrics were not deemed sufficient for the demands of this system. Go, on the other hand, offers a closer comparison to Rust in terms of performance and safety[5]. Both Go and Rust would be suitable choices based on these attributes, but Rust was ultimately preferred due to its slight edge in performance and my prior experience with the language. While C has a small edge in performance over Rust, Rust contains more high-level language features, which makes fast development and maintainability easier[7], which was important for this system.

This careful consideration of language features and system requirements led to the selection of Rust, ensuring that the backend is robust, efficient, and capable of supporting the system's intended functionalities.

As the backend server is implemented in Rust, Actix Web[1] was chosen as the preferred backend web framework. It was preferred to use a web framework to implement the HTTP server, as a web framework handles most of the HTTP server setup, to easier focus on the implementation of this system's unique features and requirements. Actix Web was chosen because of familiarity, its ease-to-use, wide support, and good documentation.

4.4 Data Storage

For this system, it was decided to store data on the backend server's file system. Using the file system for data storage offers simplicity, speed in development, and flexibility, making it ideal for quickly testing functionality with minimal setup. This approach was preferable for a POC like Guorrat due to the limited development time available and the primary focus being on evaluating its unique functionalities, rather than on how data is stored. However, should Guorrat transition to commercial use, moving to a database would likely be necessary to accommodate increased data volumes, enhance security, and improve data management capabilities, such as handling user and event data.

The metadata received from the external source, Sadji[51], is formatted as

JavaScript Object Notation (JSON) structures, on an agreed-upon format. The formats of this metadata can be seen in Figures 4.2 and 4.3. JSON[23] is a popular language-independent data-interchange format, known for its simplicity and ease of use, as it is both straightforward for humans to read and write and for computers to parse and generate. The metadata is stored directly at the backend's file system as JSON files.

4.5 Retrieving Video Stream and Metadata from External Source

During a live football match, Guorrat will continuously connect to the external source, Sadji[51], to retrieve the video stream and metadata. Periodically, matching the specified length of the HLS segment files, the backend server will connect to the external source to fetch the HLS segment file and its associated metadata file, assuming it is ready. Upon a successful retrieval, the video stream data and metadata are processed and stored on the backend server, and the HLS manifest file is promptly updated to reflect the availability and location of the newly acquired HLS segment. A part of the metadata processing is to run it through an algorithm that attempts to improve the continuity of the IDs associated with each detection in the received metadata. This algorithm is covered in more detail in Section 4.10. Once the video stream data and metadata is processed and stored in the file system, the associated segment number is added to a list of distribution-ready segment numbers, and the HLS manifest file is updated to reflect the presence of the newly added HLS segment file and the corresponding metadata file.

4.6 Backend Streaming Server

As one functional requirement of the system is to provide the frontend with a live video stream of a football match, along with the corresponding metadata, the backend server is designed to provide that service. The backend serves a HTTP server, providing the frontend server with endpoints to request the HLS manifest and segment files, as well as the associated metadata files. The HTTP server is also responsible for handling event tagging related requests from the frontend.

While HLS provides Adaptive Bitrate Streaming (ABR) to accommodate for different levels of network quality, this system only offers one stream. This was chosen since Guorrat is supposed to be a POC, and the focus has therefore

been at implementing the functional requirements that are unique for this system.

4.7 Synchronizing Video and Metadata

An essential part of the system is ensuring the video stream and metadata generated by the external source, Sadji[51], are synchronized, for the interactive player rectangles to correctly line up with the players when drawn onto the video. Due to the utilization of the HLS protocol, which relies on the segmentation of the video stream, the metadata is also divided to match these HLS segments. With an HLS segment duration of 2 seconds and a video frame rate of 30 Frames per Second (FPS), each segment contains a total of 60 frames. However, with some configuration changes, the system should be capable of handling different frame rate and segment length setups. The segmentation of the metadata files are performed by the external source.

How the metadata received from the external source, Sadji[51], is structured can be seen in Figures 4.2 and 4.3. Each frame in the metadata contains up to 22 detections, decided by how many of the players the external source is able to detect.

```
{
    "frame": "1",
    "detections": [...],
    "homography": [
        [
          0.6376710495779074,
          -0.00015582586796899614,
          1011.8963446285936
        ],
        [
          0.0006676863375638175,
          0.6379639539749161,
          104.04555853337479
        ],
        [
          7.498534648589027e-07,
          4.0181671317781595e-07,
          1.0
        ]
      ]
}
```

Figure 4.2: Example of how a frame is represented in the JSON metadata.

```
{
    "id": "1000006",
    "team": 1,
    "pixel_pos": {
        "x1": 1475,
        "y1": 332,
        "x2": 1494,
        "y2": 383,
    },
    "field_pos": {
        "x": 558,
        "y": 258,
    }
}
```

Figure 4.3: Example of how a detection is represented in the JSON metadata.

A crucial choice during the development of Guorrat was where to synchronize the video stream and metadata segments and draw the interactive player rectangles onto the video. Ultimately, the frontend was chosen, and in the following section the benefits and downsides of each solution is explored, in the context of the requirements of Guorrat.

4.7.1 Frontend vs Backend

By handling the synchronization and video processing at the backend, the frontend server is relieved from having to recalculate and draw rectangles at a rate of 30 times per second, assuming a frame rate of 30 FPS. The initial concern was that if these tasks were managed by the frontend server, which is also tasked with streaming the video, facilitating tagging interactions, and displaying data—the workload could overwhelm the frontend, reducing the usability of the system. In addition, by doing the metadata synchronization and video processing at the backend server, the process would only have to be done once for each match, instead of having each client doing it separately. Therefore, by doing it that way, it could increase the scalability of the system.

As Rust was chosen as the programming language for the backend server, it became clear that there were clear obstacles regarding video and image processing in Rust. The processing process consists of three stages: decoding the received video segment into separate frames, synchronizing and drawing the associated player rectangles on each frame, and to re-encode the video segment. The two options to use for the video processing task was either FFMPEG or OpenCV. FFmpeg[13] is an open-source multimedia framework designed to process multimedia files, including decoding, encoding and transcoding video files. OpenCV[37] is an open-source library for computer vision, and widely used for its advanced video processing capabilities. OpenCV can leverage FFmpeg for much of its under-the-hood video I/O operations, as FFmpeg operates at a lower level and provides developers with direct access to control encoders, decoders, and transcoders.

Rust offers a crate with Rust bindings for OpenCV; however, after encountering initial difficulties in getting it operational and considering the warning in the crate's official README—"The API is usable, but unstable and not very battle-tested; use at your own risk"[8]—it was decided against using it. It was decided to use 'video-rs'[9], a high-level video toolkit Rust crate, to implement the video processing at the backend server. An experiment was performed to evaluate this implementation, and the results from this experiment showed that while the implementation could support real-time processing, it did not perform especially well. In Section 5.8, this experiment will be explored in detail.

A crucial distinction between handling metadata synchronization and video processing on the backend versus the frontend lies in the scale and nature of the processes involved. When performed on the backend, this task necessitates decoding the video into individual frames, synchronizing metadata, drawing rectangles on each frame, and then re-encoding these frames back into a video segment. In contrast, conducting this process on the frontend simply involves synchronizing the metadata and drawing the rectangles directly over the video during playback, without altering the underlying video frames themselves.

In addition, by drawing the rectangles over the video when the video stream is played at the frontend, with React, click event registration can be used. As each rectangle is its own React component, it is quite straight-forward to register when a rectangle has been clicked by the user. As a rectangle is clicked, its associated information, such as its position, connected ID/player, can easily be retrieved and sent to the backend server to register a new event. Also, as a user holds the mouse pointer over a rectangle component, this can be detected by React, and makes it possible to highlight the given rectangle, making it easier to ensure for an operator that the correct rectangle is clicked. That effect can easily result in a better and easier user experience, and increases the usability of the system. If the rectangles were drawn on the video at the backend, the identification of the clicked rectangle would have to be done by calculating the position of the click, making it a more complex and error-prone process, making the system harder to maintain.

After evaluating the aforementioned factors in the context of the functional and non-functional requirements of Guorrat, the frontend was chosen to do the metadata synchronization and drawing. The non-functional requirements of usability and real-time performance were especially important for this choice.

4.7.2 Frontend Handling

As the frontend server only stores a limited metadata segments at a time, to limit client memory usage, it uses a sliding window approach to handle metadata segments. As the video player processes the video stream and parses the provided HLS manifest file, it preemptively loads available video segments, a technique commonly known as buffering. This buffering helps provide a smooth and uninterrupted video viewing experience. As video segments are fetched in advance, the frontend also requests its associated metadata segments. However, as the video playback must allow frequent pausing and skipping by the user to properly record all events, it is important that the frontend does not load in too many metadata segments in advance, as there is only a limited

number of metadata segments loaded concurrently.

Currently, the frontend will not prefetch metadata segment more than 3 segments ahead of the currently playing segment. As the video player starts playing a new segment, it will, if not yet fetched, try to fetch the 3 next metadata segments from the backend server. The system is designed like this to ensure that there are always loaded metadata segments that follows the currently playing segment, to ensure smooth synchronization between video and metadata. As Guorrat should allow frequent skips forwards and backwards in the video to make it easier for a user to tag event correctly, it is important that there are loaded metadata segments both following and preceding the currently playing segment.

On the frontend server, the HTML video player object monitors its current position within the video, which is measured in milliseconds. The value starts at 0 milliseconds, representing the beginning of the stream, and increments as the video plays. This timestamp is crucial for synchronizing each video stream frame with the associated positions of the player outline rectangles. The frontend manages this by storing the loaded metadata in a hash-map structure, with the JavaScript data structure Map[31]. In this Map, each entry consists of a key and an associated value: the key represents a frame number of the metadata, and the value holds all detection data for that specific metadata frame. Given that accessing elements in a hash-map has an average time complexity of O(1), it serves effectively as a storage mechanism for metadata, particularly since this metadata needs to be retrieved at a rate of 30 FPS to update the positions of rectangles on the screen accurately.

Each interactive player rectangle is its own React component, and is rendered on top of the playing video stream for each frame, as seen in Figure 4.6. The metadata received from the external source, Sadji[51], contains the position of each detection on the screen, in addition to the height and width of the detection rectangle. These values are utilized by Guorrat to determine where to draw the interactive player rectangles. Each detection in the metadata does also contain which team the detected player plays for, which this system uses to decide which color the interactive player rectangle should be.

As the video stream is playing, the program will continuously check which frame number it is currently on. As the video player only knows its current time in milliseconds, the current frame has to be calculated using the given frame rate of the video, currently 30 FPS. If the measured current time shows that the video player has reached a new frame, it uses the new frame number to retrieve the associated detection for that frame from the Map of loaded metadata. These detections are then added to another dictionary, where each key is a player ID, and the value matches the associated detection data. So as

the frame changes, the program loops over that dictionary and rerenders each player rectangle on top of the video player.

4.7.3 Backend Handling

When the frontend loads a new HLS segment from the backend server, it parses the filename of the HLS segment file it retrieved, and sends an additional request to the backend server asking for the associated metadata file. Upon success, the metadata is read and added to the previously loaded metadata. As the loaded metadata segments can be quite large, around 100-150 KB, the frontend only stores a limited number of metadata segments, to reduce unnecessary memory use at the frontend.

An in-memory cache is implemented at the backend server. This cache temporarily stores metadata segments in memory, in an attempt to reduce the overhead when the client requests a metadata segment. New metadata files received from the external source are added to this cache, as well as metadata files requested by the client that is not present in the cache. In addition, when a requested metadata file is read and added to the cache, the backend also reads and tries to add the 3 succeeding metadata segments to the cache. When a client request metadata segment N, it is likely that the next requested will be N + 1, and that is why the backend tries to preemptively cache metadata segment.

4.8 Event Tagging

For Guorrat to be able to generate real-time ball positions during a football match, it required a system where user can manually tag ball-related events, mainly passes and tackles. In the following sections, the design and implementation decisions regarding the event tagging functionality of Guorrat will be explained.

4.8.1 Configure Teams

When using Guorrat's event tagging functionality, users have the ability to configure the starting lineup for each team. To ensure that player names are available for event tagging, they must first be entered on this configuration page, seen in Figure 4.4. Whenever a team makes a substitution during a match, users can return to this page to update the lineup, replacing the name of the player who has been substituted with the incoming player's name.

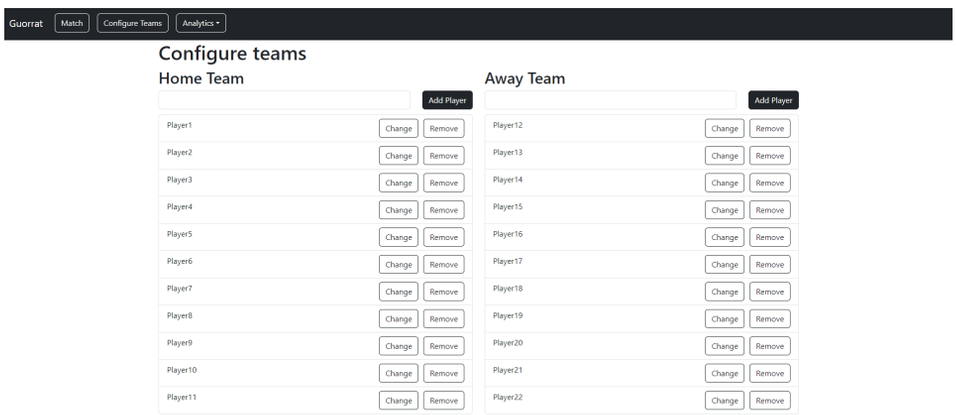

Figure 4.4: The layout of the configure teams page.

4.8.2 Event Registration

How the frontend and backend manages the registration of ball-related events will be explained in the following two subsections.

```
{
    "end_frame": 33,
    "event_id": 0,
    "event_type": 0,
    "field_coordinate_receiver":
        {
            x: 295,
            y: 541
        },
    "field_coordinate_sender":
        {
            "x": 274,
            "y": 299
        },
    "id_receiver" 2000008,
    "id_sender": 2000009,
    "name_receiver": "Player1",
    "name_sender": "Player2",
    "pixel_coordinate_receiver":
        {
            x: 301,
            y: 174
        },
    "pixel_coordinate_sender":
        {
            "x": 479,
            "y": 188
        },
    "start_frame": 7,
    "success": true,
    "team": 0
}
```

Figure 4.5: Example of an Event object.

Frontend Handling

Guorrat's event tagging system is based on a two-click principle. Each event contains a sender and a receiver. If the event is a pass, the sender is the player passing the ball, and the receiver is the player receiving the ball. In the event of a tackle, the sender is the player being tackled, while the receiver is the player that takes the ball.

Figure 4.6: Layout of stream video window with player rectangles present.

As a user accesses the video stream page in Guorrat, a "start tag" button can be clicked. When this button is clicked, the program starts drawing the player rectangles onto the video stream, as can be seen in Figure 4.6.

To register an event, the user must first select a sender. To do this, the operator must click on a player with the left mouse button at the time the sender passes or loses possession of the ball. Then, at the time the ball reaches the receiving player, the operator must click on the receiving player's rectangle with the right mouse button. The frontend temporarily stores info about the last left and right clicked rectangles, such as the connected id, field position, team and the frame number when it was clicked.

When both a sender and a receiver has been selected, an event popup will show on screen, as seen in Figure 4.7 and Figure 4.8. The event popup consists of name fields for both the sender and receiver, as well as a button to change either one. In addition, the popup will contain buttons for different event types, determined by the context regarding the event. If the two involved players in an event is on the same team, a 'pass' is the only available event type, as can be seen in Figure 4.7. If the two involved players are on different teams, the event can be classified as either a 'pass' or a 'tackle', as is displayed in Figure 4.8.

As an event type is selected, an event object is created to register the event. The event object consists of metadata regarding the event, including data regarding the sender and receiver, and the event itself. The content of the event object can be seen in Figure 4.5. This event object is then added to a list containing

all registered events. In addition, the newly created event object is sent to the backend server to process a possible possession and to generate the ball positions.

Up until now, only clicks on player rectangles have been discussed. However, the metadata received from the external source, Sadji[51], occasionally fails to detect all players. To compensate for this, Guorrat allows a user to click anywhere on the video to select either sender or receiver of an event, or in the case the ball goes out of play. This feature will be explained in more detail in Section 4.11.

Figure 4.7: Event popup when registering an event between two players of the same team.

Figure 4.8: Event popup when registering an event between players on different teams.

Backend Handling

As an event is tagged and registered on the frontend, it sends a POST request to the backend containing the newly created event object. As the backend server receives this request, it reads the content and adds the event object to a list of events. The backend uses the received event object and the previous received event objects to determine if a player possession has occurred.

Then, the server uses the received event object to determine if a new player possession has happened and to generate the ball positions during these events. This process to determine possession and generate ball positions is explained in more detail in Sections 4.8.3 and 4.9.

4.8.3 Possession Registration

As mentioned in Section 4.8.2, when an event is tagged, the system initiates a process to determine if a player possession has occurred. This operation is executed by the backend after it receives the tagged event object from the client. The process involves two steps: firstly, assessing whether a possession has indeed taken place, and secondly, if necessary, retrieving the positions of the involved player throughout the duration of the possession.

In the first step, the system checks two conditions to determine if a possession

has occurred. First, the system verifies if the received event is the first tagged event; if so, it concludes that no possession could have occurred previously. The second condition checked is whether the sender of the most recently registered event is the same as the receiver of the preceding event, and that the IDs match. The definition of sender and receiver is defined in Section 4.8.2. This continuity is essential for determining a possession sequence. The logic for determining if a possession has occurred is visualized as a flowchart in Figure 4.9.

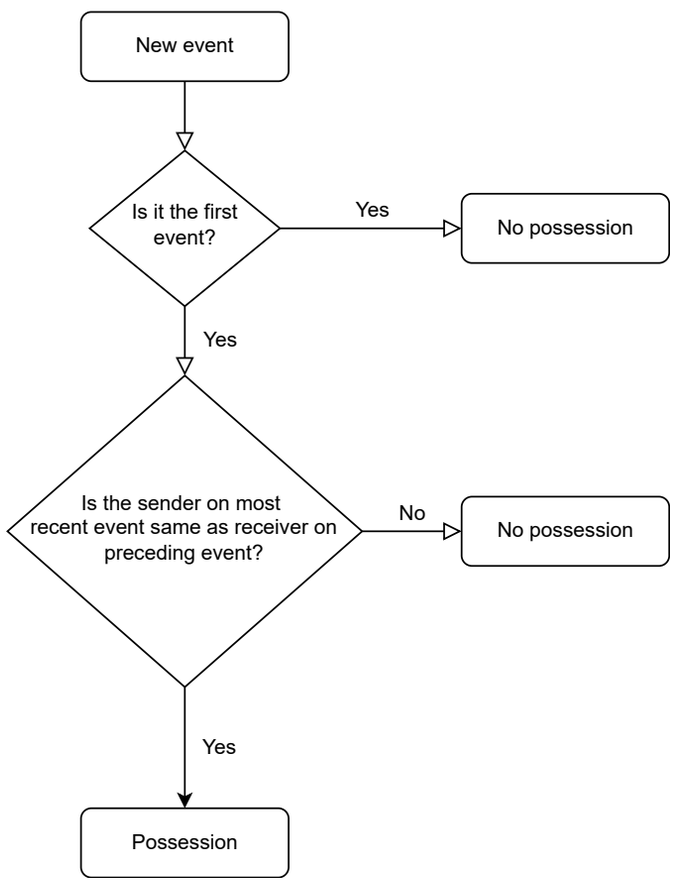

Figure 4.9: A flowchart representing the logic to determine a possession.

The second step, if needed, reads the needed metadata to retrieve the given player's position on the field during the possession. The possession's time frame starts at the end of the preceding event, and ends at the start of the most recent event. The system uses these frame numbers to determine which metadata segment files it must read to cover the entire possession. Each metadata segment is then read and loaded into memory. For each frame within the possession period, the system extracts and records the field position of the detection that matches the possessing player's ID into a list of coordinates.

If the ID of a player possessing the ball switches during the possession, this system is not able to generate the ball positions during the duration of the possession. This is because the ID is used to decide which detection's field position in the metadata that are to be recorded. This is the main reason the algorithm for improved ID tracking has been developed, which is detailed in Section 4.10.

Once all relevant frames have been processed, this list will contain the complete positional data of the player during the possession. The structure of the possession object can be seen in Figure 4.10.

```
{
    "player_id": 17,
    "player_name": "Player 1",
    "team": 0,
    "start_frame": 164,
    "end_frame": 228,
    "coordinates": [
        {
            "x": 825,
            "y": 295
        },
        {
            "x": 829,
            "y": 389
        },
        ...,
    ]
}
```

Figure 4.10: Example of a possession object.

4.8.4 Connect Player Names to Events

As mentioned earlier, the user is provided the possibility to select the name of the players that are involved in each tagged event, to improve the quality of the event data. As mentioned in Section 4.8.1, this requires that the user has added the players manually before tagging.

When the register event popup is displayed on screen during the tagging of an event, the popup will contain a field for the name of both the sender and receiver. The first time a rectangle associated with a given player ID is involved

in an event, that player's name field will not contain any names. To make it easier to spot for the user, the name fields is color coded. If a player name has been connected to that rectangle, the background of the player name field will be green, as seen in Figures 4.7 and 4.8. In the case no player has been selected, the field will be colored red, as seen in Figure 4.11.

Figure 4.11: Event popup where no player names have been associated with the receiving player.

After a player name has been connected to a player rectangle, the system will attempt to keep track of the selected player name. The frontend uses each players unique player ID to keep track of each rectangle's associated player. Section 4.10 will explain further how Guorrat's player ID algorithm works. As each rectangle contains a unique ID, a dictionary is used to store the mapping between ID and player names. Each key in the dictionary is associated with a player name. When registering an event, the system will use the IDs of the clicked player rectangles to search the dictionary for a connected player name.

When an ID switch occurs between two players, p1 and p2, their corresponding rectangles often switch places directly. This means that if rectangle r1 associated with p1 exchanges with rectangle r2 associated with p2, r1 will likely end up associated with p2, and r2 with p1. As a result, if a user records a subsequent event involving either p1 or p2, the incorrect player name may appear in the event popup. Since the system anticipates a direct swap, when the user corrects r1's associated player name back to p1, the system will automatically update r2's associated player name to p2. This is done in an attempt to minimize the required number of user inputs needed.

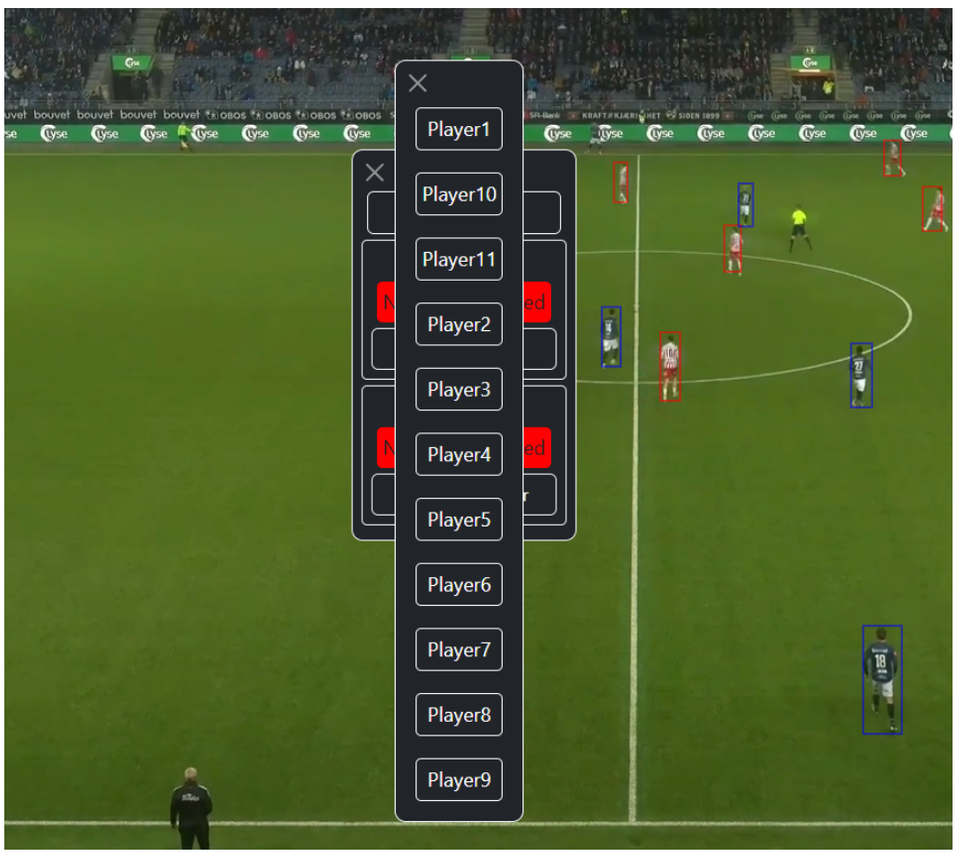

Figure 4.12: Menu for selecting player.

4.8.5 Display Data

A functional requirement for Guorrat is to present the user with a display of tagged event and ball data, as detailed in Section 3.1.6. This is achieved via three additional frontend pages. The first page showcases the tagged passes and tackles, the second page presents the derived possession data, and the third page displays the generated ball positions.

For both pages, users have the ability to filter the events based on various criteria, including team, player, success of the event, and the time of occurrence. The interface for displaying passes can be seen in Figure 4.14.

Figure 4.13: How the generated ball data is displayed.

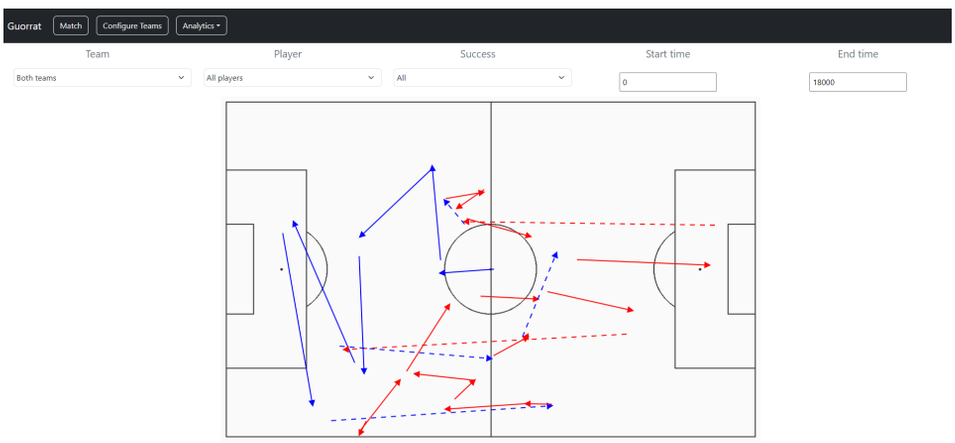

Figure 4.14: How tagged passes are displayed.

4.9 Generate Ball Positions

As the primary contribution of Guorrat is to generate real-time ball positions for a football match, this section will explain how it is done. Guorrat is able to register ball positions in two ways. The first is through tagged events. For any tagged event, Guorrat uses the recorded start and end field position and frame number to interpolate the ball position for the duration of an event. These are the field positions generated for each player by the external source, Sadji[51],

and is present in the received metadata.

The other way, is the method explained in Section 4.8.3, where the ball position is generated by utilizing the field position of the involved player during a possession. By combining these two methods, Guorrat attempts to generate a ball position in as many frames as possible, with the highest possible accuracy. How well Guorrat succeeds with its ball generation is evaluated in Section 5.2.

4.10 ID Tracking Algorithm

Each detection in the metadata is tagged with an ID for identification, assigned when it is generated at the external source, Sadji[51]. The external source system employs a multi-object tracking algorithm, ByteTrack[59], in conjunction with the object detection algorithm to track detections across consecutive video frames. As a result, each detection in the received metadata includes an ID field that assigns a unique value to each new detection. However, as players intersect or move in and out of the video frame, the multi-object tracking algorithm faces challenges in consistently tracking all players, leading to instances where IDs may switch between two players or a player may be assigned a new ID.

Guorrat relies on these received IDs to correctly track possessions to generate ball positions, and to track which player name is selected for each player rectangle. However, when these IDs switch between players or when a player receives a new ID, it becomes more difficult to accurately track ball positions and player names. To address this, Guorrat works on enhancing the consistency of ID tracking across continuous frames. When the backend server receives a new metadata segment from the external source, an algorithm is executed to refine the IDs of each detection before the metadata is sent to the frontend server. This improvement helps maintain accurate ball tracking and player selection throughout the video analysis process. The implemented ID algorithm works as follows:

The backend has a dictionary that keeps track of each teams outfield players and keeper. For each entry in the player dictionary, it is given an new unique ID that should stay consistent throughout the game, hereby referred to as P-ID. Each detection's ID as it arrives from the external source, Sadji[51], is hereby referred to as D-ID. Each entry in this dictionary contains three elements. The first is the P-ID value to identify each entry. The second element is the last recorded field position associated with that P-ID, as in the field position of the last detection that were mapped to that entry. The third element is the D-ID of

the last detection that were mapped to that entry.

When a new metadata segment has arrived from the external source[51], the backend reads the segment metadata file and loads it into memory. For each frame in the segment, all detections are looped over. If the D-ID of a detection in the new frame matches an entry with the same D-ID in the dictionary, that entry is updated with the field position of the new detection. If a detection in the new frame does not match any of the entries in the dictionary, it is noted that there is a new/unknown D-ID. As the system tracks every new/unknown D-ID, it also tracks which, if any, entries in the dictionary that has not been matched with any new detections. The first step of the algorithm is to map all detections from the new frame to their corresponding entry and update that entry. The reason all detections in the frame are looped through before any potential new detections are mapped to a dictionary entry, is to prevent a new detection being mapped and overwriting an entry connected to another detection that is still present.

After that is done, the algorithm goes through all new detections that does not match any of the previous entries, and finds the available entry that has the closest field position to the new detection. When the closest entry has been found, any other new/unknown detections on the same frame can not be mapped to that entry. After all new detection has been mapped to a entry in the dictionary, the new detection's ID is changed from the D-ID to the entry's P-ID.

The impact of this ID-tracking algorithm and how it improves the system is tested and evaluated in sections 5.3 and 5.5.

4.11 Non-Rectangle Click Registration

At times, the external source, Sadji[51], producing the metadata will fail to detect all players in a frame. If it happens that the metadata is missing the detection of a player during an event the given player is involved with, and thereby missing the given player's position on screen and on the field, there will be no rectangle on screen representing the given player, and an event can not be registered by clicking on the rectangle. For easier referencing, this feature is hereby referred to as Non-Rectangle Click Registration (NRCR).

The NRCR feature allows a user, when registering an event where an involved player is not detected, to click anywhere on the video to manually select either a player, or register that the ball has gone out of play. When the user then clicks on the given player's position on the screen, that pixel coordinate is registered

and sent to the backend, and the associated field position is calculated and returned. By adding this feature, the precision of the generated event data is increased. This improvement is investigated and evaluated in section 5.4.

For the external source, Sadji[51], to be able to translate a player's position on the video screen to their position on the field, a panorama image has to be created for that match. This consists of stitching together frames from different angles to create a panorama image that contains the whole field. For each match, a homography matrix used to calculate the relation between the panorama image and the field is calculated by the external source. An assumption made for this system, is that for each match, this homography matrix is received from the external source.

At the backend server, when the received pixel coordinate is received, it is first scaled to the original dimensions of the video stream. As visualized in Figure 4.2, frames in the metadata can contain the homography matrix used to translate the position between a point on the given frame to the corresponding point on the panorama image. The backend multiplies this frame-specific homography matrix with the given pixel coordinate vector, which returns the associated coordinate on the panorama image. Then the same operation is performed again, but with the panorama image coordinate and the panorama-to-field homography matrix, which then returns the associated coordinate on the field. The calculated field coordinate is then returned to the client, where a new event object is created and registered.

Due to the computationally intensive nature of calculating the transformation matrix between a frame and the panoramic image, the external source, Sadji[51], can not provide this matrix for every frame without compromising real-time performance, as detailed in the thesis regarding the external source[51]. To meet real-time requirements, the external source is able to supply the transformation matrix only every 30th, 25th, or 20th frame, depending on the resolution of the video stream.

To address this issue, when the backend needs to translate a user's click on a specific frame into its corresponding field position, it employs interpolation. The process begins by parsing the metadata to identify the closest frames—both preceding and succeeding—that contain a transformation matrix. It then calculates the field positions for these two frames. By interpolating between these positions, the backend estimates the field position of the clicked frame. This approach represents a strategic compromise aimed at balancing two critical non-functional requirements: real-time performance and precision. How this added feature influences the system is evaluated in section 5.4.

4.12 Summary

As real-time ball tracking comes with several challenges, especially for football clubs with limited budgets and infrastructure, as detailed in Sections 2.2.1 and 2.2.2, Guorrat is designed and implemented to be an effective solution for real-time ball detection. This chapter has explained how the architecture of Guorrat is laid out, and how the system uses manual event tagging in combination with the received player positional data to generate real-time and precise ball positions. The chapter has also covered some of the features added to the system, the improved ID-algorithm and the NRCR feature, in an attempt to improve the precision of the generated ball data.

5 Evaluation

In this chapter, Guorrat will be evaluated. The precision and quality of the generated ball positions by Guorrat will be evaluated, in addition to experiments focused the real-time performance requirement of Guorrat.

5.1 Experiment Setup

All video segments used for evaluating of Guorrat is taken from the match between Viking Fotballklubb (VIK) and Tromsø Idrettslag (TIL) at SR Bank Arena at 22.10.2023[1]. This is because early in the development process of this project, ball and player positional ground truth were manually created for several segments from this match to test and evaluate the correctness of the system during development.

5.1.1 Hardware Specifications

As Guorrat should be accessible for football clubs with limited budget, it is important that the system has been tested on relatively cheap hardware that should be obtainable for even resource-restricted football clubs.

1. https://highlights.eliteserien.no/game/3857

Lab Desktop

The lab desktop computer used during the development and experimentation process of Guorrat has the following hardware specification.

- OS: Windows 11
- CPU: 13th Gen Intel Core i7-3700
- GPU: NVIDIA GeForce RTX 3070
- Memory: 128GB DDR4 RAM

Laptop

As coaches/analysts might require more mobile computers, such as laptops, to use Guorrat at the stadium, a laptop with the following hardware specification has been used on some of the experiments.

- OS: Ubuntu 20.04
- CPU: Intel Core i7-5500 @2.40GHz
- GPU: Intel HD Graphics 5500
- Memory: 8GB DDR3 RAM

5.2 Measure Frequency and Precision in Generated Ball Positions

Guorrat offers an alternative method for tracking the ball in football matches, differing from the sensor-based and object detection-based methods detailed in sections 2.2.1 and 2.2.2. This experiment aims to evaluate and compare the ball positions generated by two different iterations of Guorrat (Iteration 1 and Iteration 2) with those provided by the external source, Sadji[51], that generates the metadata. This experiment is intended to evaluate both the frequency of the generated ball positions and the accuracy of these generated positions.

The key difference between the two iterations of Guorrat is the additional feature in Iteration 2, NRCR , which allows users to click anywhere on the

frame when registering an event, with the system's ability to translate any point on the frame to the associated position on the field. This is particularly useful when a player is not detected or the ball goes out of play. Iteration 1 lacks this functionality. The details of this feature are thoroughly explained in Section 4.11.

5.2.1 Experiment Setup

For this experiment, a 2 minute and 15 seconds long segment from the second half of VIK-TIL is used to evaluate the generated ball positions. This segment was chosen as its a relatively long segment of open play, with action occurring on most of the field. The ball position result from the external source is received directly from Sadji[51] for this experiment. For the two iterations of Guorrat, the ball position results is generated by executing the event tagging during the segment, with the system processing the events and returning the derived ball positions.

The ball position ground truth that has been generated by manually marking the position of the ball for each frame on the video, and then using homography transformation to translate to the associated field position. All generated ball positions in this experiment is compared to these ground truth ball positions.

While the segment consists of 4,034 frames from start to finish, 118 of these frames were excluded from the evaluation, since the ball is outside of the field during these frames. Therefore, only 3,916 frames are used to calculate the resulting percentage of frames with a generated ball position.

For assessing the accuracy of each solution, Root Mean Squared Error (RMSE) is employed. RMSE quantifies the average error, called Residual, between the actual data points and the generated data points, expressed in the original units of measurement. In this context, the RMSE value represents the average distance, in meters, between the actual ball positions and the predicted ball positions generated by the various solutions.

To ensure the accuracy of the RMSE calculations, it is essential to consider the impact of outliers, which can disproportionately skew the RMSE values. Therefore, RMSE is computed both with and without these outliers. Outliers are identified using the Interquartile Range (IQR) method. The IQR, which represents the range between the first quartile (Q1) and the third quartile (Q3), measures the statistical spread of the middle 50% of the data, indicating where the bulk of the values lie. In this context, the dataset consists of the residuals between the actual and generated ball positions for each frame. Any

Dataset	Total ball detections	% of total frames (3,916)
Ball positions from metadata	151	3.9
Guorrat iteration 1	2,837	72.5
Guorrat iteration 2	3,856	98.5

Table 5.1: Number of generated ball positions by each solution.

residual that is less than 2 times the IQR below Q1 or more than 2 times the IQR above Q3 will be classified as an outlier. This method helps in maintaining the integrity of the RMSE calculations by mitigating the effect of extreme values.

5.2.2 Results

Position Frequency

We will first explore the results regarding the ball position frequency of each solution. With the ball positions generated by the external source, the ball position is registered on 151 (3.9%) frames. The result from Iteration 1 of Guorrat generates a ball position on 2,837 (72.5%) frames, while Iteration 2 of Guorrat generates a ball position on 3,856 (98.5%) frames.

Position Accuracy

The results for the ball positions generated by the external source can be seen in Figure 5.1A. The distribution of the Residuals Magnitudes can be seen in Figure 5.1B. Figure 5.4 displays the calculated RMSE values for the external source generated ball positions equals 8.04 with outliers, meaning that, on average, each ball position generated by the external source is located 8.04 meters from the ground truth ball positions. With outliers removed, the average distance shrinks to 0.62 meters.

For Iteration 1 of Guorrat, the resulting ball positions can be seen in Figure 5.2A, compared with the actual ball positions. The residual distribution can be seen in Figure 5.2B. As we can see in Figure 5.4, this solutions results in an average distance between the generated and ground truth ball position of 8.03 meters including outliers, and 0.86 meters when the outliers are excluded.

Regarding Iteration 2 of Guorrat, we can see the results of the generated ball positions in figure 5.3A, and the distribution of residuals in Figure 5.3B. For this iteration, the average distance between generated and actual ball position

is only 3.85 meters, and 0.86 meters when outliers are removed, as displayed in Figure 5.4.

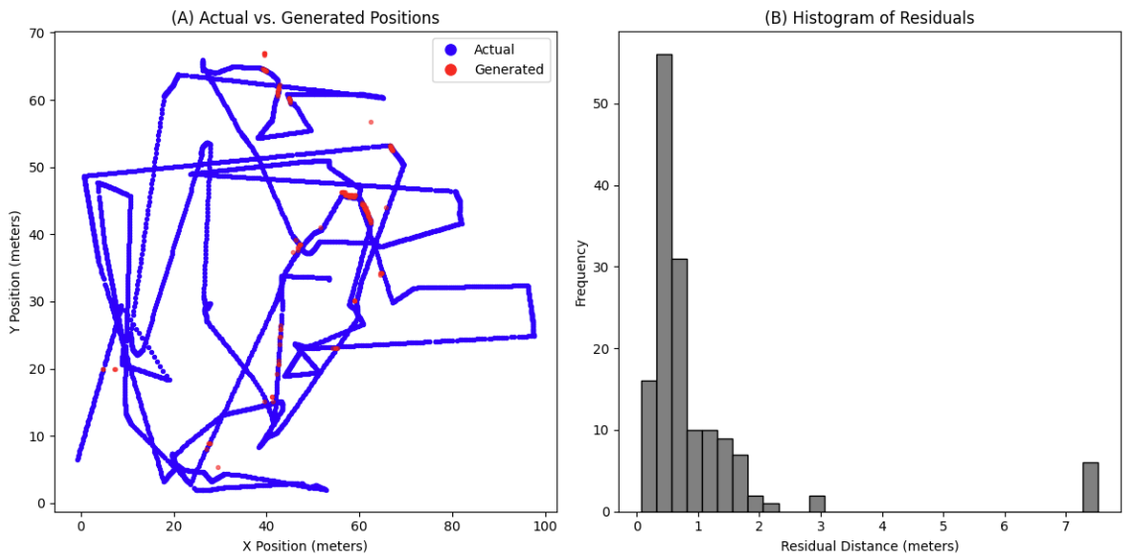

Figure 5.1: **A**: This figure shows the ball positions generated by the external source on top of the actual ball positions. **B**: A histogram showing the distribution of the residuals.

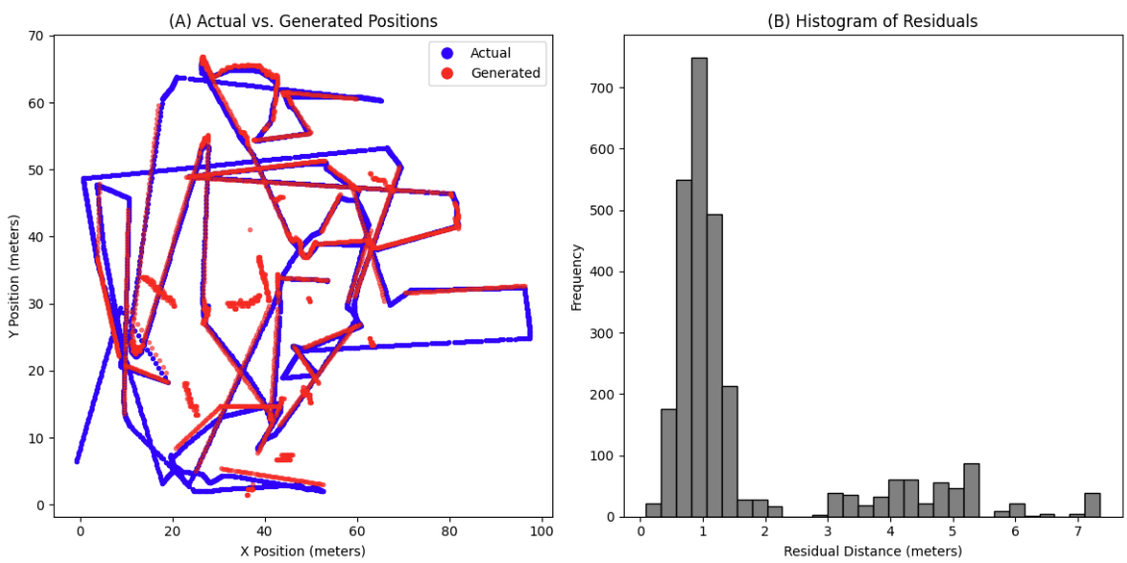

Figure 5.2: **A**: This figure shows the ball positions generated by Iteration 1 of Guorrat on top of the actual ball positions. **B**: A histogram showing the distribution of the residuals.

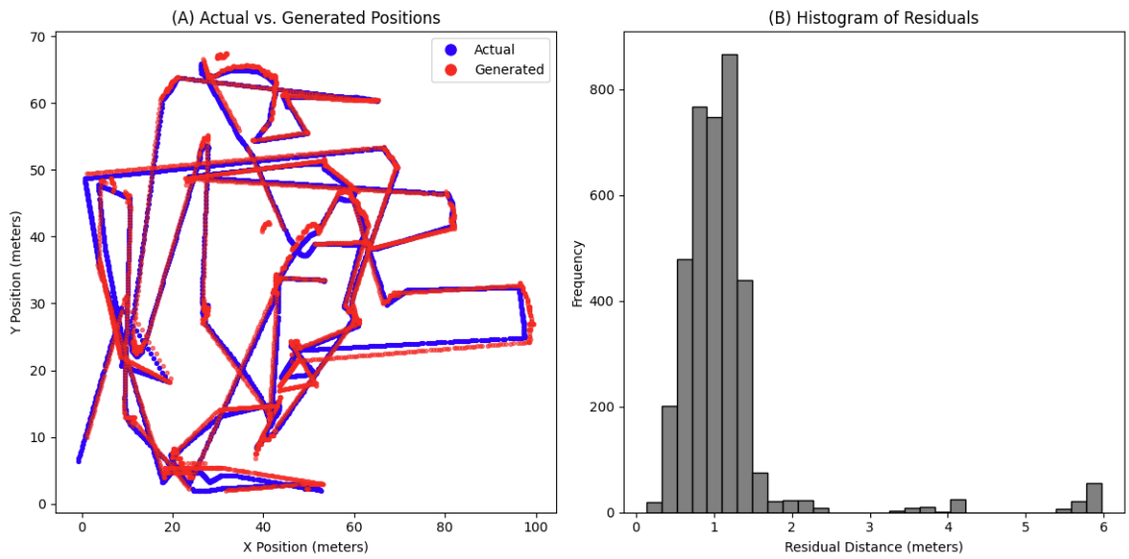

Figure 5.3: **A**: This figure shows the ball positions generated by Iteration 2 of Guorrat on top of the actual ball positions. **B**: A histogram showing the distribution of the residuals.

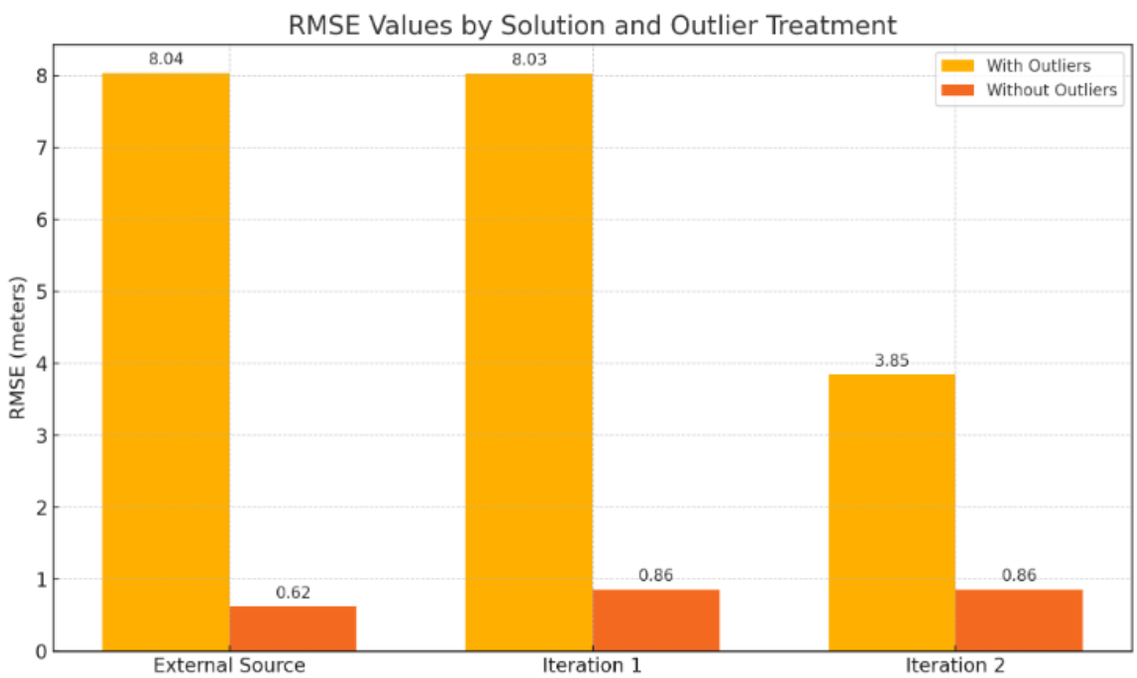

Figure 5.4: A bar chart showing the calculated RMSE values for each solution, both with and without outliers.

5.2.3 Discussion

Regarding the frequency of the different solutions, the results demonstrate a substantial improvement of ball position data compared to the data provided by the external source, Sadji[51].

The reason Iteration 2 does not achieve 100% accuracy primarily relates to how ball positions are derived from possessions, as outlined in Section 4.8.3. For the system to recognize that a possession has occurred and to subsequently calculate the ball's position, it requires the ID of a detection. This ID allows the system to track the detected player's position throughout the possession. However, if an event is registered and the involved player is not detected by the external source, the system fails to recognize the possession. Consequently, it cannot calculate the ball's position during that time frame, regardless of whether the possession actually occurred.

As for the accuracy of the different solutions, we see that, when removing outliers, all solutions performs quite equal. For the ball positions generated by the external source[51], it makes sense that it comes very close to the actual ball positions when removing outliers. When the object detection algorithm used by the external source[51] manages to detect the ball, it should be able to place the position of the ball more accurately than the two other solutions. This higher accuracy is attributable to the algorithm's capability to directly calculate the ball's actual location, as opposed to Guorrat's iterations 1 and 2, which derive ball positions indirectly through the position of the possessing player or via interpolation between two events. When players possess the ball, it is rarely located directly between their feet. This means that even if the player detections were perfect, the actual and generated ball positions will be different by generating ball positions with this method.

Given that the ball positions provided by the external source[51] are relatively sparse, it is reasonable that a few substantial outliers can significantly skew the RMSE result. These outliers are likely instances where the Object Detection algorithm used by the external source[51] fails to detect the ball correctly, resulting in large deviations from the true positions. By removing these outliers, we mitigate their disproportionate impact on the RMSE, leading to a more representative evaluation of the system's performance.

As illustrated in Figure 5.2B, there is a noticeable prevalence of positions with high residuals. This likely explains why Iteration 1 exhibits comparable scores to the other solutions once outliers are removed. A significant proportion of generated positions with high error margins are disregarded in the outlier-removed analysis, skewing the performance assessment towards more favorable outcomes.

When considering the comprehensive evaluation of ball position frequency, accuracy, and the distribution of residuals, it becomes evident that Iteration 2 of Guorrat significantly outperforms the other two solutions. It is crucial to acknowledge, however, that the ground truth data for ball positions was generated manually, rather than through automated sensory methods. This manual process could introduce potential inaccuracies, particularly in frames where precise ball positioning is challenging to determine visually.

5.3 Generated Ball Position Improvement with Improved ID Tracking

The process of tracking a possession requires a consistent ID for the involved player throughout its duration, as detailed in Section 4.8.3. The ID tracking algorithm employed by Guorrat, detailed in Section 4.10, aims to enhance these IDs, which should in turn improve the precision of ball positions during specific possessions. However, this improvement is expected to affect only a subset of total possessions during a match, specifically those where the raw metadata from the external source, Sadji[51], contains an ID switch between the possessing player and another. Such ID switches can result in erroneous ball positions, recording the wrong player's field positions up until the switch occurs. This experiment focuses on comparing derived field positions of the ball by examining a scenario where this ID switch takes place.

5.3.1 Experiment Setup

In this test, a possession by a TIL player is used to visualize the derived ball positions, and compare between the results when using the original IDs from the raw metadata from the external source[51] and the generated IDs from Guorrat. In the metadata received from the external source, Sadji[51], an ID switch occurs as the player, possessing the ball, runs past a Viking defender about halfway into the possession.

5.3.2 Results

Figure 5.5 shows the generated ball position from this possession when the raw metadata IDs are used. In figure 5.6 we see the resulting derived ball positions during the possession when the improved IDs are used.

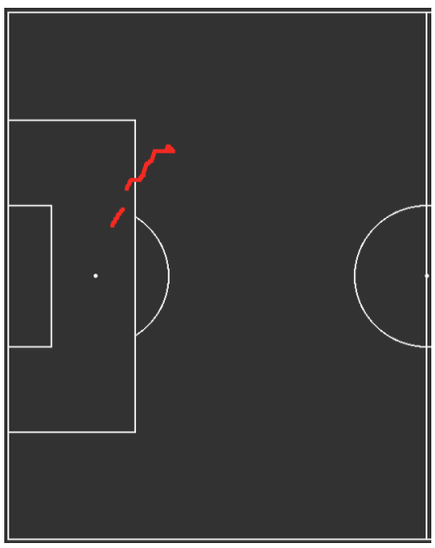

Figure 5.5: With original ID tracking.

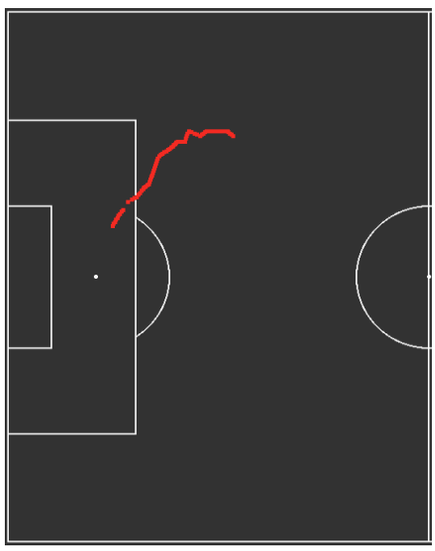

Figure 5.6: With improved ID tracking.

5.3.3 Discussion

Although the frequency of such situations might vary, the results from this example shows that the improvement in ID tracking makes a difference in the quality and precision of the generated ball position data by Guorrat. Meaning that the improvement is of importance, as it increases the non-functional requirement Precision for this system.

5.4 Generated ball position improvement with Non-Rectangle Click Registration

One issue with the metadata received from the external source, Sadji[51], is that it occasionally fails to detect all players. Various factors, including the player's distance from the camera and the contrast between the player and the background, can hinder player detection. Previously, Guorrat required that both the sender and receiver of a pass to be detected for an event to be registered, as there would be no interactive player rectangle to click on otherwise. However, the current iteration of the system now enables users to click anywhere on the video, and it calculates the corresponding field position. This enhancement enables the registration of events even when one or both of the involved players are not detected to increase the precision of the generated ball and event data, and is explained further in Section 4.11.

5.4.1 Experiment Setup

This test explores an example where such a situation occurs. The example contains a brief 4-pass sequence during the second half of the VIK-TIL match. The reason this match is used is explained in Section 5.1. During this sequence, a Viking player is not detected in the metadata during his possession, between the time he receives the ball and the time he passes it on. A snapshot during this possession, with the mentioned VIK player circled, can be viewed in Figure 5.7.

Figure 5.7: Screenshot from VIK-TIL showing the situation evaluated in Section 5.4. The mentioned Viking player is highlighted.

5.4.2 Results

In figure 5.8 and 5.9, the resulting registered passes can be seen. We can see that with the improved functionality of the current iteration of Guorrat, all four passes are registered, while with the previous iteration, the passes involving the undetected VIK player is not registered.

5.4.3 Discussion

This example demonstrates that the improved functionality of the current iteration of Guorrat is more adept at managing situations where the external source[51] fails to detect all involved players. By providing the user the ability to register all passes, and calculate the field position of said passes, even in

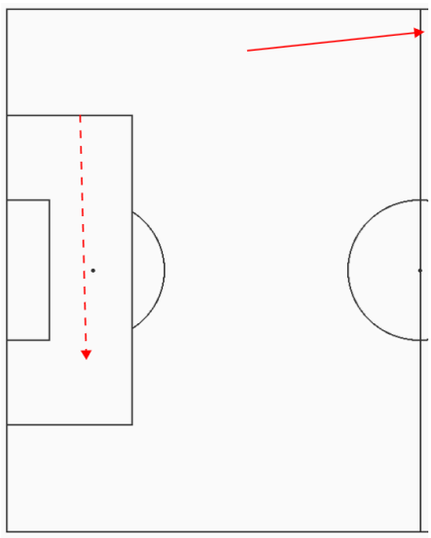

Figure 5.8: Registered passes without NRCR solution.

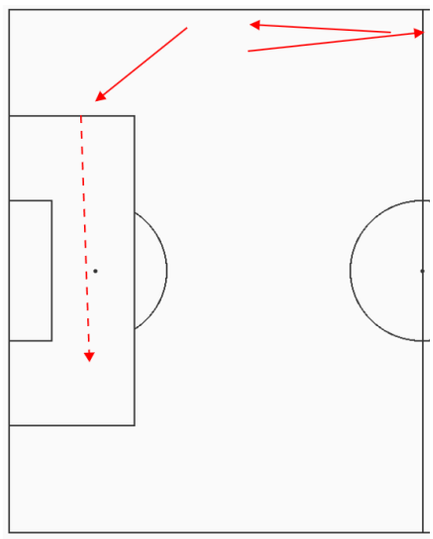

Figure 5.9: Registered passes with NRCR solution.

the case of lacking metadata, the precision of the generated ball positions and event data is increased.

5.5 Measure ID Improvement with Changing Player Names

Guorrat allows users to manually designate which player is on the receiving or sending end of an event. This capability enhances the tagged event data, which can be used in addition to the generated ball position data to enhance analysis. As detailed in Section 4.8.4, this feature leverages the unique ID associated with each player rectangle to track the selected player linked to that rectangle, as determined by the user. In an ideal scenario, users would only need to select an associated player for each player rectangle once, with the system subsequently maintaining these selections throughout the match. However, as the received metadata from the external source, Sadji[51], does not provide reliable continuous player IDs, Guorrat uses an ID-tracking algorithm, detailed in Section 4.10, to improve the continuity of each player's unique ID. This experiment is designed to evaluate the improvement of Guorrat's ID-tracking algorithm and if required user input is decreased.

5.5.1 Experiment Setup

The experiment is performed on several different segments from the VIK - TIL match, the reason being explained in Section 5.1. For each segment, Guorrat is ran twice. On the first run, the player IDs generated by the external source, Sadji[51], is used. In the second run, the IDs generated by Guorrat's ID-tracking algorithm is used. For each run, an associated player is preselected for each rectangle. The reason the preselection is done is to more accurately measure the frequency that the ID tracking system fails, by isolating the experiment down to only these happenings. The experiment itself is quite simple, as it works by counting the number of times the user has to manually change the associated player for a rectangle. Since associated players for each rectangle is preselected, manual input will only be needed when a player's ID switches.

A problem with the metadata received from the external source [51], is that its object detection algorithm struggles to effectively detect and track players in tight situations, especially far from the camera. Such situations is hereby referred to as ID-challenging situations. A prime example of such a ID-challenging situation is corner kicks, where the majority of players are gathered tightly in one of the team's penalty area and the action takes place far away from the camera. Because of this, the segments used in the experiment are divided into two different groups. The first group consists of the segments that only consists of open play, while the second group consists of segments containing ID-challenging situations. An example of an ID-challenging situation, where several players in the box is not detected, can be seen in Figure 5.10.

Figure 5.10: Example of an ID-challenging situation.

5.5.2 Results

As we can see in table 5.2 and table 5.3, the improved ID algorithm shows an improvement in all tested segments. In table 5.2 the number of times manual input is needed from an user to correct the suggested associated player for a given rectangle on the first group of segments, the ones without ID-challenging situations. The results for this group of segments show an average per minute decrease of 81.4% in required manual input.

In table 5.3 the same measurement is done, but on the second group of segments, the ones containing ID-challenging situations. For this group, the average decrease in required manual input per minute is 53.9%.

Segment	Segment length	Improved IDs	Original IDs	Improvement
Segment 1	2 min 15 sec	4	29	86.2%
Segment 2	49 sec	0	2	100%
Segment 3	1 min 27 sec	4	12	66.6%
Per minute	1 min	1.77	9.51	81.4%

Table 5.2: Table showing number of times a user have to manually change the associated player for a rectangle. Segments without ID-challenging situations.

Segment	Segment length	Improved IDs	Original IDs	Improvement
Segment 4	10 min	27	65	58.5%
Segment 5	2 min 30 sec	12	26	53.8%
Segment 6	1 min 48 sec	8	11	27.3%
Per minute	1 min	3.29	7.13	53.9%

Table 5.3: Table showing number of times a user have to manually change the associated player for a rectangle. Segments with ID-challenging situations.

5.5.3 Discussion

The results of this experiment clearly demonstrates that Guorrat's ID algorithm improves the continuity of the player IDs, and that the number of user inputs required is reduced. This improvement is attributed to the improved ID tracking algorithm used by Guorrat, detailed in Section 4.10, which effectively reduces reliance on the less reliable IDs generated by the external source, Sadji[51].

It is a substantial contrast between the level of improvement for the two different segment groups. It is not surprising that the improvement for open-play segments beats the improvement for segments with ID-challenging situations. It demonstrates some of the limitations with using an object detection approach to automatically detect and track players, as used by the external source, Sadji[51], especially in these ID-challenging situations.

5.6 In-Memory Metadata Caching

While metadata segment files are stored on the file system and accessed upon client requests, the backend server maintains a cache that temporarily stores metadata segments in memory to ensure quicker response times. Since a user should be able to frequently jump to different parts of the video stream, and given that the frontend stores a limited number of metadata segments in memory, it is crucial for the backend to respond quickly when the client requests a specific metadata segment, to ensure a as smooth as possible user experience.

The goal of this experiment is to evaluate the difference in response time when a client retrieves a metadata segment, depending on whether the requested segment is cached.

5.6.1 Experiment Setup

The experiment involved sending 20 requests to the backend and recording the total time taken for the client to receive the response. The number 20 was chosen to get a more representative result, and to reduce the effect of outliers. Since the experiment was conducted over localhost, network latency did not influence the results.

5.6.2 Results

In Figure 5.11, the experiment's results are depicted. When there is a cache hit, the average server response time is 1.87 milliseconds. However, if the requested metadata segment is not in the cache, the average response time increases to 57.86 milliseconds.

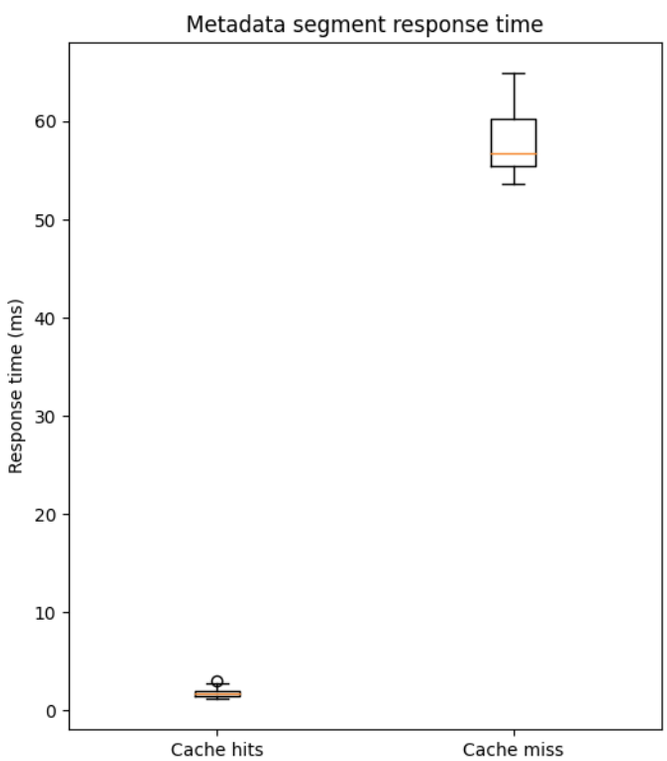

Figure 5.11: Time it takes for the client to receive a metadata segment from the backend server.

5.6.3 Discussion

The response time with a cache hit is approximately 3.2% of the response time compared to a cache miss, indicating that caching metadata segments significantly speeds up the system's response to frequent time skips. The larger the metadata cache size is, the fewer metadata segments have to be kept in a client's memory, while still being able to respond quickly to frequent skips and rewinds by a user.

Under acceptable network conditions, the performance gains from a cache hit are relatively more significant than under poor network conditions, since the time required at the backend to read the metadata constitutes a smaller fraction of the total latency. This implies that in optimal network environments, the client can maintain minimal metadata segments in memory to reduce memory usage without sacrificing performance. Conversely, in suboptimal network conditions, the client may need to store a larger number of metadata segments to achieve similar levels of user experience. This strategy helps offset the increased total latency caused by slower network speeds.

5.7 Frontend Video Processing Performance

As Real-time performance is one of the non-functional requirements for this system, it is vital that the frontend server can execute the metadata synchronization and video processing fast enough. Since coaches and analysts should have the option to use Guorrat from the sidelines or the arena, the frontend should perform well on a portable laptop, which typically uses less powerful hardware than the desktop computer used during the development of this system. This experiment is performed on both the desktop and a less powerful laptop.

As the current system operates at a frame rate of 30 FPS, the process of updating the rectangles for each frame should ideally take no more than 1/30 of a second, equivalent to 33.3 ms. Results exceeding this 33.3 ms threshold would indicate that the system fails to meet the real-time performance requirement. Furthermore, it would be preferable for the processing time to remain well below this limit to ensure smooth user interactions and maintain overall system responsiveness.

5.7.1 Experiment Setup

This experiment measures the time it takes at the client to update the position of all rectangle and for React to re-render each rectangle on the screen. Google Chrome's developer tools DevTools[14] was used to measure the frontend performance. The hardware specification of the desktop computer is detailed in Section 5.1.1, and the hardware specification for the laptop computer is detailed in Section 5.1.1. The measurements has been performed 20 times for both setups, to get a more representative result and to reduce the impact of outliers.

5.7.2 Results

Figure 5.12 shows the results of this experiment. With the desktop setup, the average processing time at the frontend per frame is 2.294ms. With the laptop setup, the average processing time is 7.266ms. As the threshold for maximum processing time is 33.3 ms, both setups perform well below this limit, at respectively 6.9% and 21.8% of the upper threshold.

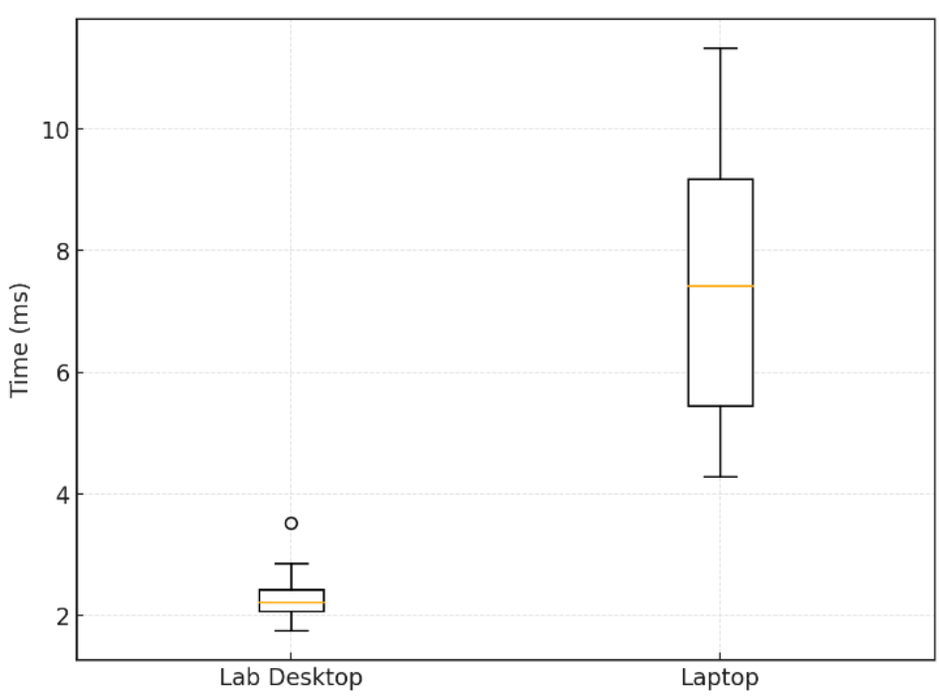

Figure 5.12: Enter Caption

5.7.3 Discussion

The results indicate that the frontend operates well within the established performance threshold, even on less powerful devices. Concerning the non-functional requirement of real-time performance, the frontend's tasks of metadata synchronization and rectangle updating do not compromise the system's ability to perform in real-time.

5.8 Backend Video Processing Performance

This experiment was done in the context of the choice between doing the metadata synchronization and rectangle drawing on the backend or frontend.

5.8.1 Experiment Setup

The experiment consisted of the following steps:

1. Decode a given '.ts' video segment file, and put all frames, as bytes, in a list.

2. Read the associated metadata file from storage, loop through each frame and draw each player detection rectangle on the frame.

3. Re-encode the updated list of frames into a '.ts' video segment file, and store it.

The encoder utilized in this experiment was h264 with the YUV420P pixel format. To assess whether segment length impacts processing time, three different video segment lengths of 2,4, and 8 seconds were tested in this experiment. The process is executed 10 times per video segment length, to get a more representative result and to limit the effect of outliers. The experiment was performed on the lab computer, with its specifications listed in Section 5.1.1.

5.8.2 Results

In Figure 5.13, the height of each bar represents the average total processing time for each video segment length. Additionally, each bar is divided into three sections, with the height of each section indicating the average time spent on decoding the video, synchronizing and drawing player rectangles on each frame,

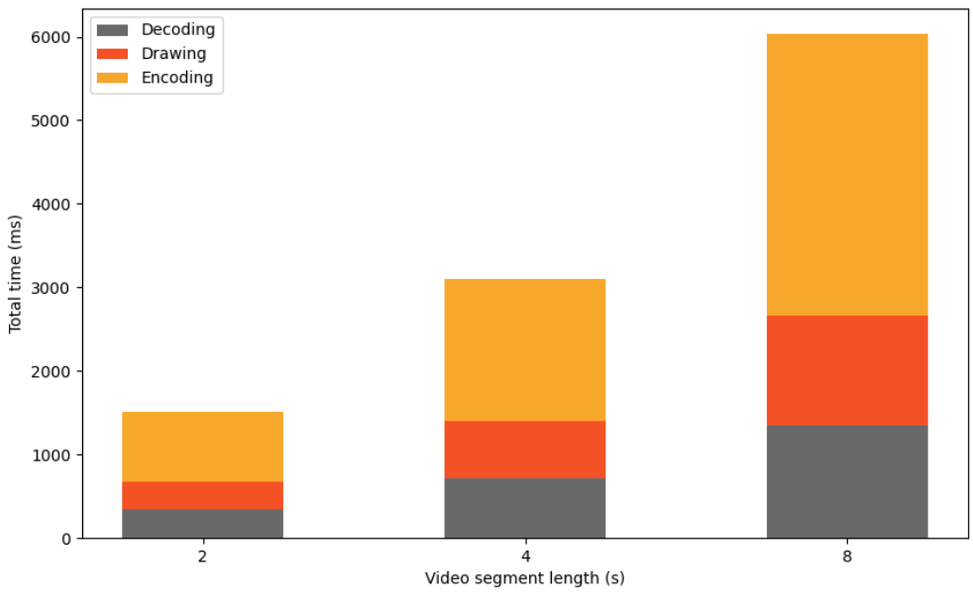

Figure 5.13: Total time to process each segment at the backend server.

and re-encoding the video, respectively. Figure 5.14 illustrates the proportion of video segment processing time relative to the total duration of each segment, expressed as a percentage. This demonstrates that it takes upwards of 75% of the total segment time to process each segment. Additionally, the results shows that, percentage-wise, the video processing is fastest for 2-second segments and slowest for 8-second segments, with a difference of approximately 5 percentage points.

5.8.3 Discussion

While the results are not exceptional, they suggest that this implementation could meet the system's requirements for real-time service if employed. However, as detailed in section 4.7.1, this solution was not chosen for the final implementation, and thus alternative, potentially more effective configurations were not explored. Since variations in video encoders and pixel formats can significantly impact encoding speeds and video quality[6], exploring different configurations might have resulted in improved performance.

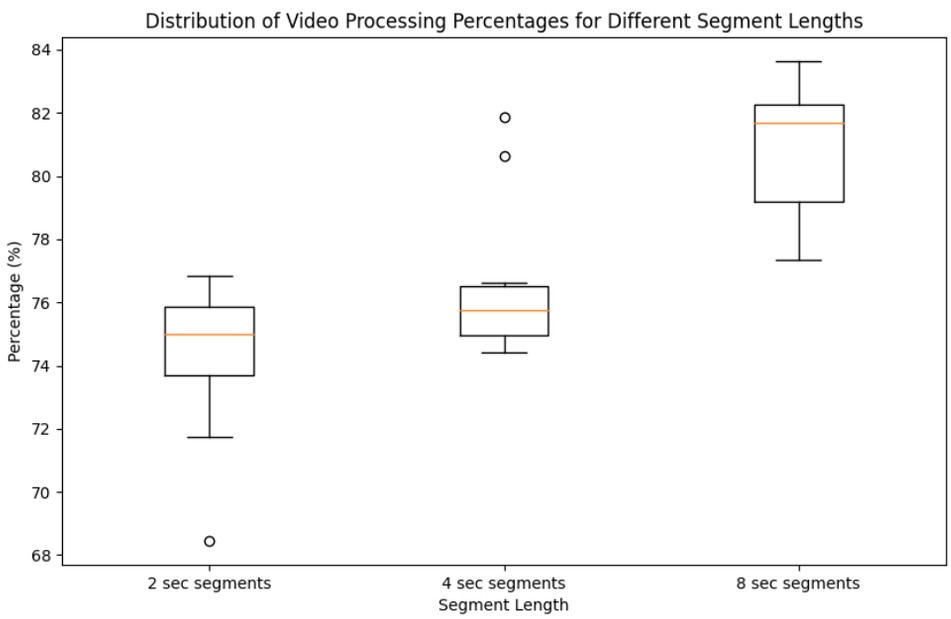

Figure 5.14: Percentage of total segment time to process the video segments at the backend server.

5.9 Summary

After these experiments and evaluation of Guorrat, it is clear that Guorrat can act as an alternative to the real-time ball tracking methods explained in Sections 2.2.1 and 2.2.2. The experiments demonstrates that Guorrat can deliver real-time, high precision and frequency ball positions throughout a live football match.

6 Discussion

6.1 Requirements

In Sections 3.1 and 3.2, the functional and non-functional requirements of Guorrat are detailed. In this section, the design and implementation choices influenced by these requirements will be explored, and how successful Guorrat is in fulfilling these requirements will be evaluated.

6.1.1 Functional Requirements

The main purpose of Guorrat is to **generate real-time ball positions**, with as high precision as possible. Guorrat leverages the positional player data received by the external system, Sadji[51], in combination with a manual event tagging system to generate the ball's position throughout a live football match. Guorrat should be able to act as an alternative way to detect the ball in real-time, in contrast to the automatic methods most commonly used today, as detailed in Sections 2.2.1 and 2.2.2. In Section 5.2, it was demonstrated that the final iteration of Guorrat detects the ball position in 98.5% of frames with high precision, during the segment of play used in the experiment. However, the number of frames where Guorrat can generate a ball position will vary between different segments of play. The frequency of ball detections will ultimately depend on the quality of the received metadata from the external source[51].

The second requirement, **Retrieve video stream and metadata from external**

source, is essential for the system's operation. During development and testing, however, this metadata received from the external source, Sadji[51], has been simulated by storing and reading it from disk. The design details of this functionality are further explained in Section 4.3.

The third requirement, **Provide live stream of match with synchronized player rectangles**, is addressed in Sections 4.6 and 4.7. This feature is crucial for the system's functionality, allowing users to view live matches with interactive, overlaid rectangles. The synchronization of metadata and rectangle rendering occurs on the frontend rather than on the backend server, with a detailed discussion of this design choice presented in Section 4.7.1.

The fourth requirement, **Handle Event Tagging**, is essential for enabling users to tag ball-related events like passes or tackles. Initially processed on the frontend, these events are then transmitted to the backend, where they are utilized to determine and register possessions and to generate ball positions. This workflow and the backend processing are elaborated in Section 4.8.2, 4.8.3, 4.9, respectively. The event tagging functionality works, and is evaluated in context of the ball positions generated, in Sections 5.2, 5.3 and 5.4.

Regarding the fifth functional requirement, to **Track Names of Players**, Guorrat strives to track the selected player name for each player on a best-effort basis. This functionality relies on player IDs to successfully link to the corresponding player name. To enhance ID continuity, Guorrat employs an improved ID-tracking algorithm, detailed in Section 4.10. While this algorithm reduces ID switches between players and the need for frequent user inputs, as evaluated in Section 5.5, it is not perfect and still requires occasional user intervention to correctly associate player names.

For the last functional requirement, **Display Registered Ball and Event Tagging Data**, Guorrat provides an interface for a user to view the generated ball positions, along with the tagged passes, tackles, and player possessions. This interface allows a user to filter events on different criteria, making it easy for the user to decide which data they want to see. This interface is displayed in figure 4.14.

6.1.2 Non-Functional Requirements

For Guorrat's Event Tagging Functionality to work properly, the system must perform at **real-time performance** speed. The part of the system where real-time performance was in question was mainly the process of updating and drawing the position of each detection rectangle for each frame. Solutions for this has been implemented at both the frontend and backend of the system,

and both have been tested and evaluated. The details regarding this design choice is present in Section 4.7.1. The backend solution required the decoding and re-encoding of all video segments to draw the detection rectangles on the video. The results of this solution, detailed in Section 5.8, showed that this solution could support the real-time performance requirement, but by a quite small margin. By doing this process at the frontend instead, the video segments does not have to be decoded and re-encoded. The frontend solution performed quite well and could support real-time performance, as detailed in Section 5.7, even on computers with older and less powerful hardware, proving that the system can support real-time performance. Additionally, Guorrat's ability to perform in real-time is dependent on that the external source, Sadji[51], can generate player positional metadata in real-time.

Concerning the non-functional requirement **Precision**, several factors influence the quality and accuracy of the event and ball data generated by Guorrat. A critical determinant is the precision of the metadata received from the external source, Sadji[51]. If the positional data linked to player detections in this metadata is inaccurate, there is limited capacity for Guorrat to enhance the precision of the generated data. However, should the received metadata lack player detections, this system incorporates Non-Rectangle Click Registration (NRCR), which allows users to tag ball related events and select involved players regardless. This functionality, detailed in Section 4.11 and evaluated in Section 5.4, enriches the generated ball positions by ensuring that all events are registered. Nevertheless, it cannot compensate fully for imperfect metadata, as possessions and thereby the associated bass positions during the possession can not be accurately determined from such incomplete events. The experiments performed on the frequency and accuracy of Guorrat's generated ball data, detailed in Section 5.2, shows that the current iteration of Guorrat has close to 100% generated ball positions with high precision.

However, the frequency and precision of the generated ball positions by Guorrat will vary depending on how a match unfolds. For instance, if a match contains a high number of tight situations, resulting in less complete metadata generated by the external source, Sadji[51], Guorrat will most likely generate fewer and less precise ball positions.

Another key factor contributing to the precision of the generated ball data is the ID-tracking algorithm employed by Guorrat, as outlined in Section 4.10. This algorithm significantly enhances the continuity of the player IDs, directly improving the precision of the ball data, which is further evaluated in Section 5.3.

While the generated ball positions might not always be precise enough to be used directly for advanced analysis, the generated event data can supplement

these analyses. As the registered events contain the players involved and their positions, and since it might be just as relevant to know which player has the ball and his position, instead of the actual ball position, several advanced analysis metrics can still be performed.

While the **usability** of Guorrat has not been directly exposed to users and tested, it has been exposed to coaches with experience in the domain during the development process for feedback, which has influenced the design of the system. The two-click tagging system, detailed in Section 4.8.2, is designed in an attempt to make the event tagging functionality. Also, by drawing the player rectangles on the video with the use of React components, as explained in Sections 4.7.1 and 4.8.2, the rectangles have direct click registration and can easily be highlighted when hovered over by the mouse pointer. This is a design decision that should make Guorrat easier to use.

By implementing the backend of Guorrat in Rust, we enhance the non-functional requirements of **Reliability** and **Availability**. As detailed in Section 4.3, Rust's enforcement of strict type and memory safety significantly increases the system's reliability, thereby facilitating higher availability. Similarly, the use of TypeScript for the frontend promotes these requirements by enforcing strict type safety, which reduces the likelihood of undefined behaviors in the source code. Additionally, the adoption of languages with strict type safety means that potential errors are more likely to be caught during compile time rather than at runtime. This approach also ensures that the source code is more self-documenting and easier to understand, further contributing to the overall **maintainability** of the system.

6.2 Selection of Streaming Protocol

For the streaming of the live match from the backend server to the frontend server it was chosen to use HLS. Real-Time Messaging Protocol (RTMP) would also be a natural choice, as it is a popular choice for streaming live content, because of its lower latency than HLS. RTMP uses a persistent connection where the video stream data is transmitted continuously from the streaming server to the client, allowing for near real-time delivery. HLS, in contrast, must, in the case of 2-second segments, wait 2 seconds from recording until it can deliver a segment, incurring a minimum of 2 seconds off real-time.

While the 2-second HLS segment length used in Guorrat incurs a minimum of 2-second delay, the system can still be considered real-time in any practical sense. This is because it would be practically impossible for a user to properly tag all ball-related events without pausing the video stream at times, meaning

that the HLS delay will most likely not have any practical effect on when the generated ball positions and event data is ready to be used for analysis.

By using HLS, the synchronization between the video stream and the generated metadata by the external source, Sadji[51], is made easier to perform as each video segment is associated with a metadata segment with a unique segment number. In addition, as the backend already employs a HTTP server to handle the event tagging requests from the client, it only requires two more server endpoints to deliver the HLS stream to the client. With RTMP streaming, a separate connection point must be implemented to deliver the video stream.

6.3 Summary

In this chapter, it has been explored and discussed how Guorrat fulfills the functional and non-functional requirements of the system. It is demonstrated how Guorrat can generate frequent and high precision ball positions with real-time performance over the course of a live football match. Additionally, the choice of streaming protocol has been discussed.

/7

Conclusion & Future Work

This chapter will summarize the thesis and provide some concluding remarks regarding the problem statement. Lastly, it will explore some possible future work.

7.1 Concluding Remarks

This thesis has proposed Guorrat, a real-time system for alternative generation of precise ball positions during a live football match. Guorrat has been designed to act as an alternative to current automatic methods for real-time ball detection, and to provide precise ball data to improve real-time advanced football analysis.

In Chapter 1, this problem statement was defined:

> *This thesis will carry out research and development of an alternative to state-of-the-art real-time ball detection systems. This POC (Proof of Concept) will utilize player positional data in combination with manual event tagging to generate ball positions, with focus on real-time and precision properties.*

Following, in Chapter 2, technologies for real-time automatic ball detection was

explored, and how these methods is mostly reserved for larger football clubs with high budgets and solid infrastructure. Additionally, it was evaluated how some existing analysis system delivers on real-time ball tracking and tagging of ball-related events.

In Chapter 3, the functional and non-functional requirements of Guorrat was specified. A live stream of a football match with a manual event tagging system to generate ball positions were the essence of the functional requirements, with real-time performance and precision of the generated ball position data as the most crucial non-functional requirements.

Chapter 4 detailed how Guorrat has been designed and implemented, and how the functional and non-functional requirements of Guorrat influenced the decision-making in the design and implementation phase. How Guorrat fetches the metadata generated by from the external source, Sadji[51], how it synchronizes the video and metadata, and how it uses the event tagging functionality to generate ball positions were explained in this chapter.

Chapter 5 focused on the evaluation of Guorrat. It was demonstrated that Guorrat generates frequent and precise ball positions, and that the system can handle the real-time requirement property.

In Chapter 6, discussed and evaluated how Guorrat fulfills the requirements of the system specified in Chapter 3. It is clear that Guorrat can generate precise ball positions in a real-time fashion. However, the frequency and precision of the generated ball and event data will ultimately depend on the quality of the generated data produced by the external source, Sadji[51].

7.2 Thesis Conclusion

Guorrat, a POC system for generating precise ball positions in real time has been developed, by utilizing real-time player positional data in combination with manual event tagging.

To conclude, with the results demonstrated and discussed in Chapters 5 and 6, we can determine that Guorrat has the ability to generate real-time and precise ball positions during a live football match. It demonstrates that Guorrat can act as an alternative to state-of-the-art real-time ball tracking systems, to supplement and enhance real-time advanced analysis in football.

However, the precision and quality of the generated ball data will ultimately depend on the player positional data received by the external source, Sadji[51],

which has been developed in parallel with Guorrat.

We do not necessarily recommend a complete replacement of state-of-the-art ball detection systems in these scenarios, but rather combining our approach with these systems to improve the overall precision.

7.3 Future Work

As of now, Guorrat is only developed as a POC, meaning that there are room for further development of the system. Following is a few further developments that could be possible.

7.3.1 Integration with other Analysis Platforms

It could be preferable to integrate Guorrat with other analysis platform, where it can act as a feature in a bigger system. An integration with Sárgut[4], a system for high level real-time analysis visualization developed in parallel with this system, could provide users with a more detailed analysis.

As concluded, it would be interesting to combine state-of-the-art (AI-based) ball detection algorithms and combine with the Guorrat approach.

There are numerous advanced analytical metrics that can be calculated when both the player and ball positional data is present, such as Packing[45, p. 4], Expected Threat[52], and numerous other. Therefore, Guorrat can be integrated with systems that lacks real-time ball positions to enhance the potential for analysis.

7.3.2 Deploy Guorrat for Commercial Use

If not integrated into another analysis platform, another future work could be to continue to develop Guorrat into its own complete analysis system and deploy it for commercial use. As it receives real-time player positional data from Sadji[51], and generates real-time ball position, it could be further developed to generate numerous advanced metrics and display this for users in a real-time fashion. This would require a larger focus on security and compliance, as Guorrat would most likely need to store both user and team data. Additionally, more focus on the scalability of the system would be needed, and how it would affect the availability and reliability of the system.

References

[1] Actix. *Actix Homepage*. URL: https://actix.rs/. [Accessed: 2024-04-22]. 2024.

[2] Apple Inc. *HTTP Live Streaming (HLS)*. URL: https://developer.apple.com/documentation/http-live-streaming. Accessed: 2024-05-04. 2024.

[3] Ivan Baptista et al. "The variability of physical match demands in elite women's football." In: *Science and Medicine in Football* 6.5 (2022), pp. 559–565.

[4] Børge Bårdsen. "Sárgut." Submitted for review, May 15, 2024. MA thesis. UiT The Arctic University of Norway, 2024.

[5] Bitfield Consulting. *Rust vs. Go*. URL: https://bitfieldconsulting.com/posts/rust-vs-go. [Accessed: 2024-04-22]. 2024.

[6] Petr Cika, Dominik Kovac, and Jan Bilek. "Objective video quality assessment methods: Video encoders comparison." In: *2015 7th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT)*. IEEE. 2015, pp. 335–338.

[7] Manuel Costanzo et al. "Performance vs programming effort between rust and c on multicore architectures: Case study in n-body." In: *2021 XLVII Latin American Computing Conference (CLEI)*. IEEE. 2021, pp. 1–10.

[8] crates.io. *OpenCV Crate*. URL: https://crates.io/crates/opencv. [Accessed: 2024-04-18]. 2024.

[9] crates.io. *video-rs Crate*. URL: https://crates.io/crates/video-rs. [Accessed: 2024-04-18]. 2024.

[10] Peter Denning et al. "Computing as a discipline: preliminary report of the ACM task force on the core of computer science." In: *Proceedings of the nineteenth SIGCSE technical symposium on Computer science education*. 1988, pp. 41–41.

[11] Sayed Mohammad Majidi Dorcheh et al. "SmartCrop: AI-based cropping of soccer videos." In: *2023 IEEE International Symposium on Multimedia (ISM)*. IEEE. 2023, pp. 20–27.

[12] Mohammed Dougramaji. *Data Analytics in Football: LFC*. URL: https://rockborne.com/graduates/blog/data-analytics-in-football-lfc. [Accessed: 2024-04-29]. 2023.

[13] FFmpeg. *About FFmpeg*. URL: https://ffmpeg.org/about.html. [Accessed: 2024-04-18]. 2024.

[14] Google Chrome Developer Tools. *Chrome Developer Tools Documentation*. URL: https://developer.chrome.com/docs/devtools. [Accessed: 2024-05-11]. 2024.

[15] Thomas von der Grün et al. "A real-time tracking system for football match and training analysis." In: *Microelectronic Systems: Circuits, Systems and Applications* (2011), pp. 199–212.

[16] Pål Halvorsen et al. "Bagadus: an integrated system for arena sports analytics: a soccer case study." In: *Proceedings of the 4th ACM Multimedia Systems Conference*. 2013, pp. 48–59.

[17] Gunnar Hartvigsen and Dag Johansen. "Co-operation in a distributed artificial intelligence environment—The stormcast application." In: *Engineering Applications of Artificial Intelligence* 3.3 (1990), pp. 229–237.

[18] Hudl. *About Hudl*. URL: https://www.hudl.com/en_gb/about. [Accessed: 2024-03-19]. 2024.

[19] Hudl. *Hudl Assist for Soccer*. URL: https://www.hudl.com/products/assist/soccer. [Accessed: 2024-03-19]. 2024.

[20] Hudl. *Hudl Focus*. URL: https://www.hudl.com/en_gb/products/focus. [Accessed: 2024-05-03]. 2024.

[21] Hudl. *Tag Video After the Match - Soccer (Hudl V3)*. URL: https://support.hudl.com/s/article/tag-video-after-the-match-soccer-hudl-v3. [Accessed: 2024-03-19]. 2024.

[22] Dag Johansen et al. "Muithu: Smaller footprint, potentially larger imprint." In: *Seventh International Conference on Digital Information Management (ICDIM 2012)*. IEEE. 2012, pp. 205–214.

[23] JSON.org. *JSON*. URL: https://www.json.org/json-en.html. [Accessed: 2024-04-17]. 2024.

[24] Kinexon. *UWB Technology*. URL: https://kinexon.com/products/uwb-technology. [Accessed: 2024-04-15]. 2024.

[25] Kinexon Sports. *Ball Tracking Technology*. URL: https://kinexon-sports.com/technology/ball-tracking. [Accessed: 2024-04-15]. 2024.

[26] Kinexon Sports. *Liga Portugal Showcase*. URL: https://kinexon-sports.com/blog/liga-portugal-showcase. [Accessed: 2024-04-16]. 2022.

[27] Jacek Komorowski, Grzegorz Kurzejamski, and Grzegorz Sarwas. "FootAndBall: Integrated Player and Ball Detector." In: *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 5: VISAPP,* INSTICC. SciTePress, 2020, pp. 47–56. ISBN: 978-989-758-402-2. DOI: 10.5220/0008916000470056.

[28] Lei Li et al. *Design and Implementation of A Soccer Ball Detection System with Multiple Cameras*. 2023. arXiv: 2302.00123 [cs.CV].

[29] Wei Liu et al. "Ssd: Single shot multibox detector." In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. Springer. 2016, pp. 21–37.

[30] Matt Maulion. *Homography Transform Image Processing*. URL: https://mattmaulion.medium.com/homography-transform-image-processing-eddbcb8e4ff7. [Accessed: 2024-05-15]. 2023.

[31] Mozilla Developer Network. *Map - JavaScript | MDN*. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Map. Accessed: 2024-05-15. 2024.

[32] *NACSport*. URL: https://www.nacsport.com/index.php?lc=en-gb. [Accessed: 2024-05-02]. 2024.

[33] NACsport. *NACsport Enhanced Graphic Descriptors*. URL: https://www.nacsport.com/blog/en-gb/Tips/nacsport-enhanced-graphic-descriptors. [Accessed: 2024-05-02]. 2024.

[34] Nacsport. *Graphic Descriptors*. URL: https://www.youtube.com/watch?v=VrF7POt_mRU&list=PLwKcZgIlh_bW5L7nfAvn29n746l5xMRgC&index=9. Accessed: 2024-05-02. 2022.

[35] *NACsport Versions*. URL: https://www.analysispro.com/nacsport-versions. [Accessed: 2024-05-02]. 2024.

[36] Takuya Nakabayashi et al. "Event-based High-speed Ball Detection in Sports Video." In: *Proceedings of the 6th International Workshop on Multimedia Content Analysis in Sports*. MMSports '23. <conf-loc>, <city>Ottawa ON</city>, <country>Canada</country>, </conf-loc>: Association for Computing Machinery, 2023, pp. 55–62. ISBN: 9798400702693. DOI: 10.1145/3606038.3616164. URL: https://doi.org/10.1145/3606038.3616164.

[37] OpenCV. *OpenCV Homepage*. URL: https://opencv.org/. [Accessed: 2024-04-18]. 2024.

[38] Sigurd Pedersen et al. "Improved maximal strength is not associated with improvements in sprint time or jump height in high-level female football players: A cluster-randomized controlled trial." In: *BMC Sports Science, Medicine and Rehabilitation* 11 (2019), pp. 1–8.

[39] Premier League. *Top Players - Goals*. URL: https://www.premierleague.com/stats/top/players/goals. [Accessed: 2024-04-29]. 2024.

[40] React. *React Homepage*. URL: https://react.dev/. [Accessed: 2024-04-22]. 2024.

[41] Shaoqing Ren et al. "Faster R-CNN: Towards real-time object detection with region proposal networks." In: *IEEE transactions on pattern analysis and machine intelligence* 39.6 (2016), pp. 1137–1149.

[42] Rust. *Rust Homepage*. URL: https://www.rust-lang.org/. [Accessed: 2024-04-22]. 2024.

[43] Rustacean Principles. *How Rust Empowers: Reliable Type Safety*. URL: https://rustacean-principles.netlify.app/how_rust_empowers/reliable/type_safety.html. [Accessed: 2024-04-22]. 2024.

[44] Oey Kevin Andrian Santoso et al. "Rust's Memory Safety Model: An Evaluation of Its Effectiveness in Preventing Common Vulnerabilities." In: *Procedia Computer Science* 227 (2023), pp. 119–127.

[45] Rory Smith. *Expected Goals: The Story of how Data Conquered Football and Changed the Game Forever*. Mudlark, 2022. ISBN: 978-0-00-848403-3.

[46] Spiideo. *Introducing AutoData to Spiideo Perform: The World's Only Video Analysis Platform with Automated Live Tagging and Player Tracking*. URL: https://www.spiideo.com/news/introducing-autodata-to-spiideo-perform-the-worlds-only-video-analysis-platform-with-automated-live-tagging-and-player-tracking. [Accessed: 2024-20-04]. Nov. 2023.

[47] Spiideo. *Spiideo Homepage*. URL: https://www.spiideo.com/. [Accessed: 2024-03-10]. 2024.

[48] Spiideo. *Spiideo Perform*. URL: https://www.spiideo.com/spiideo-perform/. [Accessed: 2024-04-10]. 2024.

[49] Spiideo. *Tagging in Spiideo Perform*. URL: https://support.spiideo.com/en/articles/4708322-tagging-in-spiideo-perform. [Accessed: 2024-04-20]. 2024.

[50] STATSports. *STATSports Homepage*. URL: https://statsports.com/. [Accessed: 2024-04-27]. 2024.

[51] William Alexander Stimpson-Karlsson. "Sadji." Submitted for review, May 15, 2024. MA thesis. UiT The Arctic University of Norway, 2024.

[52] David Sumpter. *Explaining Expected Threat*. URL: https://soccermatics.medium.com/explaining-expected-threat-cbc775d97935. [Accessed: 15.05.2024]. 2021.

[53] Tracab. *Tracab Homepage*. URL: https://tracab.com/. [Accessed: 2024-04-13]. 2024.

[54] Transfermarkt. *Transfer Records*. URL: https://www.transfermarkt.com/statistik/transferrekorde. [Accessed: 2024-04-29]. 2024.

[55] TypeScript. *TypeScript Homepage*. URL: https://www.typescriptlang.org/. [Accessed: 2024-04-22]. 2024.

[56] Ultralytics. *Ultralytics GitHub Repository*. URL: https://github.com/ultralytics/ultralytics. [Accessed: 2024-05-15]. 2024.

[57] video-dev. *hls.js GitHub Repository*. URL: https://github.com/video-dev/hls.js. [Accessed: 2024-04-22]. 2024.

[58] Andreas Kjæreng Winther et al. "Position specific physical performance and running intensity fluctuations in elite women's football." In: *Scandinavian journal of medicine & science in sports* 32 (2022), pp. 105–114.

[59] Yifu Zhang. *ByteTrack: Multi-Object Tracking by Associating Every Detection Box*. URL: https://github.com/ifzhang/ByteTrack. Accessed: 2024-05-15. 2021.