GUSTAVO ANDRÉ QUITÉRIO MURTA

Bsc in Electrical and Computer Engineering

AUTOMATIC EVENT DETECTION FOR VIDEOGAMES

THIS THESIS WILL FOCUS ON THE DEVELOPMENT OF A SOLUTION FOR AUTOMATIC EVENTS DETECTION FOR VIDEOGAMES STREAMS

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
September, 2024

DEPARTMENT OF ELECTRICAL
AND COMPUTER ENGINEERING

AUTOMATIC EVENT DETECTION FOR VIDEOGAMES

THIS THESIS WILL FOCUS ON THE DEVELOPMENT OF A SOLUTION FOR AUTOMATIC EVENTS DETECTION FOR VIDEOGAMES STREAMS

GUSTAVO ANDRÉ QUITÉRIO MURTA

Bsc in Electrical and Computer Engineering

Adviser: Daniel de Matos Silvestre
Assistant Professor, NOVA University Lisbon

Co-adviser: Xavier Marques Frazão
Software Engineer, Six Floor Solutions

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
September, 2024

Automatic event detection for Videogames

This thesis will focus on the development of a solution for automatic events detection for videogames streams

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v7.1.11) [1].

To all who helped in this journey, from the smallest contributions to the biggest, your support, guidance, and encouragement made this possible.

Acknowledgements

Firstly, I would like to thank all of my professors who helped me develop my skills during my studies at FCT NOVA, especially my advisor Daniel Silvestre who has guided me through this final journey of my studies. His valuable support has been crucial in shaping my dissertation to the beauty that it is now.

Next, I would also like to express my profound gratitude to the entire Six Floor Solutions team for their generous support throughout this journey which is the development of the system of this thesis. Their availability of essential resources and extensive knowledge, especially from software engineer Xavier Frazão, greatly enriched my thesis development, such as my technical skills.

Finally, I want to express my thanks to all of my colleagues and friends who throughout my learning path have listened to me and my rambles, and still have not given up on me, and even inspired me to do better. I have to express particular gratitude to my friends João Viegas, Jorge Santos, Rodrigo Carvalho, António Malato, Tomás Vasques, Diogo Carloto, Diogo Sousa, Miguel Pinto and Mariana Mariano who together with me have walked through this perilous path and have helped me overcome all the obstacles stronger, as I hope that I have helped them.

I also need to express a huge thanks to my family for their love, encouragement, and nagging in every single step of my journey. Your efforts have not been for nothing and one small proof of that is arriving at this bifurcation and be able to take a step forward without fear or regrets.

This work was partially supported by the Portuguese Fundação para a Ciência e a Tecnologia (FCT) through project FirePuma (https://doi.org/10.54499/PCIF/MPG/0156/2019), through LARSyS FCT funding (DOI: 10.54499/LA/P/0083/2020, 10.54499/UIDP/50009/2020, and 10.54499/UIDB/50009/2020) and through COPELABS, University Lusófona project 10.54499/UIDB/ 04111/2020.

"The fox knows many things, but the hedgehog knows one big thing."

— **Archilochus**, Fragments of Archilochus' poetry (Ancient Greek poet)

Abstract

The vast amount of gameplay data in modern video games presents a challenge in identifying relevant events efficiently. This project proposes a solution for real-time automatic event detection in videogames broadcasts.

After examining the current state of the art in image processing, machine learning, and event detection across various sports, methods based on computer vision techniques are introduced and described to detect, classify, and label key in-game events during League Of Legends matches, a videogame chosen because of its popularity. These events include objectives such as champion picks, turret destructions, and champion kills. We compare the performance of our developed system with manual event annotation and verify the accuracy and effectiveness using a custom **League Of Legends Event Detection** prototype.

Our research aims to provide a robust and efficient solution for detecting critical events in League of Legends broadcasts, with the potential to inspire similar systems in other eSports and gaming genres.

Keywords: Computer Vision, Video Games, Event detection, Highlights detection, League Of Legends

Resumo

Devido à grande quantidade de informação presente nos videojogos de hoje em dia torna-se difícil de identificar os momentos mais importantes de forma eficiente. Este projeto pretende responder a esse desafio com um sistema de deteção de eventos automática para videojogos.

Depois de examinar o estado de arte atual sobre o processamento de imagem, inteligência artificial e sobre a deteção de eventos em vários desportos, introduzimos e descrevemos os vários métodos desenvolvidos, baseados em técnicas de processamento de imagem, para detetar e classificar eventos durante um jogo de League of Legends, um videojogo escolhido devido à sua grande popularidade e influência no mundo dos videojogos. Estes eventos incluem a fase de escolha dos campeões para o jogo, a destruição de torres inimigas, tal como o abater dos campeões adversários. Comparou-se a performance do sistema desenvolvido com as anotações realizadas manualmente acerca do jogo, e daí demonstrou-se a exatidão do sistema automático para o jogo de LOL.

A nossa investigação visa fornecer uma solução robusta e eficiente para a detecção de eventos chave nas transmissões de League of Legends, com potencial para inspirar sistemas semelhantes em outros eSports e gêneros de jogos.

Palavras-chave: Computer Vision, Videojogos, Deteção de momentos chave, Deteção de melhores momentos

Contents

List of Figures

List of Tables

Acronyms

API	Application Programming Interface
AV	Audio-visual
BGR	Blue-Green-Blue
CDN	Content Delivery Network
CNN	Convolutional Neural Network
CRNN	Convolutional Recurrent Neural Network
DNN	Deep Neural Network
FN	False Negative
FoD	Fusion of Decisions
FoF	Fusion of features
FP	False Positive
HLS	High-level semantics
HMM	Hidden Markov Model
ISP	Internet Service Providers
LLF	Low-level features
LOL	League of Legends
MFCCs	Mel-Frequency Cepstral Coefficients
MLR	Mid-level representations
MOBA	Multiplayer Online Battle Arena
MSE	Mean Square Error

NLP	Natural Language Processing
OCR	Optical Character Recognition
OT	Overlaid text
PSNR	Peak Signal-to-Noise Ration
RGB	Red-Green-Blue
ROI	Regions of interest
RTMP	Real-Time Messaging Protocol
SVM	Support vector machine
TN	True Negative
TP	True Positive
VOD	Video On Demand
XOR	Exclusive-OR
ZCR	Zero-Crossing Rate

1 Introduction

The video game industry has become a cornerstone of modern entertainment, captivating players worldwide with its blend of creativity, technology, and interactivity. Video games offer an immersive experience, allowing players to engage with dynamic environments, compelling narratives, and intricate challenges. The industry spans diverse platforms, including PCs, consoles, mobile devices, and virtual reality, making it one of the most versatile and rapidly growing entertainment sectors [2].

Event detection in video games is a crucial process that utilizes machine learning and computer vision technologies to identify and interpret in-game events in real-time. This process is challenging due to the unique mechanics and visual styles of different game genres, as well as the variability introduced by factors like frame rates, graphical fidelity, and player behaviors. Additionally, the availability of labeled data for training machine learning models is a critical issue. Despite these challenges, event detection presents an opportunity for innovation, enhancing the gaming experience and providing developers with powerful tools for game improvement.

In the following sections and chapter, we will explore the various techniques and technologies used for automatic event detection in video games. Our goal is to develop a robust system that can analyze real-time gameplay data, segment key events, and provide actionable insights to developers, broadcasters, and players alike.

1.1 Motivation

The gaming industry is swiftly progressing in the entertainment sector, presenting ample possibilities for substantial innovation and the smooth integration of cutting-edge technologies, as evidenced in [3]. Concurrently, the Video games streaming industry is also on the rise, experiencing annual growth in both revenue and popularity, as indicated in [4] [5] [6].

Video games are rich in events, ranging from key in-game milestones such as level completions, achievements, and boss fights to player interactions like trades, voice chat, or team collaborations. And the identification and description of those events can hold transformative benefits for the gaming streaming industry, such as:

- **Enhanced Viewer Experience**: Focusing on key gameplay moments ensures a consistently engaging and enjoyable experience for viewers.
- **Efficient Highlight Generation**: Automated event detection streamlines highlight creation, saving content creators time and effort, resulting in more dynamic and shareable content.
- **Increased Stream Quality**: Automatic identification of significant events improves overall stream quality by minimizing irrelevant content, keeping viewers captivated.
- **Audience Engagement**: Prompt showcasing of exciting moments boosts audience engagement, encouraging viewers to stay longer and participate in discussions around highlighted events.
- **Competitive Edge for Streamers**: Streamers utilizing advanced event detection systems gain a competitive edge by providing a more refined and captivating streaming experience compared to others in the industry.
- **Industry Advancement**: The widespread adoption of such systems contributes to the overall advancement of the video game streaming industry, fostering innovation and attracting attention from both creators and audiences. This collective progress positions the industry as a leader in leveraging technology for immersive content delivery.
- **Bypass over the streamer**: This enables viewers or third-party entities to evaluate and extract valuable event information directly from the content, cutting expenses and increasing autonomy in accessing insights.

1.2 Background Information

The concept of games has been around for millenniums, and their origins are dated to the very birthplace of civilization - Ancient Mesopotamia. In the history of this civilization we can find the first ever example of a recorded game, the Royal Game of Ur [7]. The Royal Game of Ur consisted of a two-player game where each player races with their pieces from one end to the other over the twenty-square rectangular board. This game mirrors real-life events from its development, incorporating situations that occurred during its creation. The real-life scenarios depicted in the game continue to influence contemporary games, with current titles retaining the fundamental mechanics introduced by the original game. Although these mechanics have undergone updates, they still share the core principles established by the earlier game [7] [8].

The civilization, since its birth, has been growing and evolving but so did the games. The games matured to involve more complex mechanics, and strategies and to use new technologies, such as wood, clay, rocks, and finally computer technology, thus introducing the concept of video games, as depicted in the Oxford English Dictionary as the concept of Video games as: "A game played by electronically manipulating images produced by a computer program on a monitor or other display (now usually a program running on a games console, personal computer, or mobile device); (also) a software package for such a game; cf. computer game n."[9].

1.2.1 Streaming Platforms Overview

There are many live-streaming platforms available to the public, as depicted in [10], but not all platforms are suited to live-stream video games. From those that are suited, two platforms surpass all the others: Twitch and Youtube, emerging as the leading platforms for gamers and streamers.

When we compare the two leading platforms [11] it is important to highlight the community and audience that each one presents. Twitch is more suited to gamers, while YouTube is more appropriate to a general audience, which justifies why Twitch has a more close-knit and engaged community, allowing more gaming-related interaction between streamers and viewers. This solidifies which platform connects better with the intent of the solution to be designed.

Twitch

Twitch is a live-streaming video platform with a primary focus on gaming. Users can broadcast their gameplay to an audience who tunes in via a web interface. Streams encompass a variety of content, ranging from casual playthroughs by amateur users to large-scale broadcasts of eSports competitions. Users can assume one of two roles within the Twitch community: broadcaster or viewer. A broadcaster shares their gameplay through a dedicated channel, while a viewer watches the channel. Each streamer is restricted to a single live channel, available for a predetermined period when actively broadcasting. To facilitate communication, channels are equipped with an integrated chat room, enabling interaction among users, both broadcasters and viewers [12].

Broadcast Structure

The video broadcast structure on the Twitch platform, as illustrated in Figure 1.1, consists of five essential components. In the center is the Video component showcasing the live stream. Positioned at the top of this component are Options, displaying actions accessible to viewers during the stream. Below, broadcast information is presented. On the right of the video stream, two related elements are found: the Chat, facilitating interactions between viewers and the broadcaster, and the Send Message feature, enabling users to submit messages to the Chat.

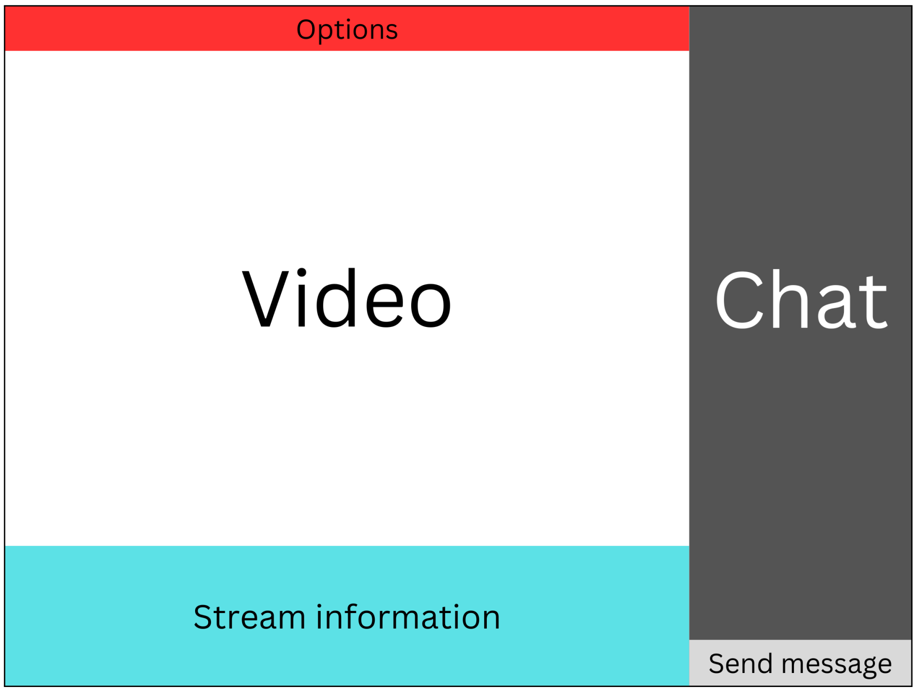


Figure 1.1: Typical structure of Twitch video streams, featuring its main elements

Within the overall structure, three visual elements play crucial roles in conveying information about the events in a gaming broadcast. The Video element visually showcases the gameplay, while the Chat facilitates viewer interaction with the broadcaster and fellow viewers. Additionally, the stream information element offers details about the content, including the name of the video game being played, the channel name, the stream title, and the relevant categories for the broadcast.

Twitch Engineering

The Twitch platform has many engineering teams trying to empower live interactive communities, fostering the creation of unique, unforgettable moments in the dynamic interactions between streamers and their audiences. To reach this goal of a consistently high-quality experience in a seemly way these teams create services and tools [13]. The following steps illustrate the process.

The Twitch Video Ingest team manages the challenge of handling hundreds of thousands of simultaneous live video streams and distributing them globally. This is achieved through a system where creators' streams are received and prepared for viewing. Twitch operates data centers, referred to as edges, distributed globally and connected to local Internet Service Providers (ISP). These edges enable streamers to quickly connect to the network, ensuring optimal video quality. Upon connection, the edge directs the stream to larger data centers known as Origins, which host numerous servers dedicated to transcoding streams into various bitrates and formats for the Twitch Content Delivery Network (CDN). This intricate process ensures viewers worldwide, on any network, can seamlessly watch their favorite streams.

Following video ingestion, the crucial next step is transcoding, where the streamers' video streams undergo conversion into formats conducive to an optimal playback experience. This ensures an excellent viewing experience regardless of the viewer's device capabilities or network conditions.

The transcoding system takes the incoming Real-Time Messaging Protocol (RTMP) stream from the streamer and transforms it into an HTTP Live Streaming-compliant stream. This transformative process generates multiple variations of the stream with diverse video resolutions and bitrates, offering viewers the flexibility to watch at the highest possible quality.

Upon arrival at an Origin and completion of the transcoding process, the responsibility of delivering the stream to viewers swiftly falls under the purview of Video Distribution. Twitch upholds a global network of distributed data centers organized in a directed graph hierarchy known as a Replication Tree.

When a viewer loads a live video page on Twitch, their request is intelligently routed to the nearest available data center. From there, the stream request is forwarded up to the Replication Tree. This hierarchical structure ensures an efficient and optimized delivery process. Ultimately, the request reaches the Origin, which promptly returns the stream data to the viewer, facilitating a seamless and quick streaming experience.

In case not all viewers could watch the stream live the creators can record a Video On Demand (VOD) of their content and distribute it, being from all the stream or some of the best moments. The last one is denominated as Highlights.

2 State-of-the-Art

The realm of automated event detection and content analysis has witnessed substantial progress over the years, aiming to enhance the overall experience for viewers, sports analysts, and enthusiasts of physical sports, video games, or sports in general [14][15]. The progression has resulted in advanced software capable of identifying and categorizing multiple events within a match or gaming session. This software can provide valuable insights into the sport and create tools to better understand the performance of players or teams [16].

In this chapter, we will present an overview of the current state-of-the-art in event detection and classification for video games. We will delineate the various events that can occur within various gaming sessions and the challenges that arise from each. Various approaches have been developed over the years and we will review them, and compare them against the most recent works that pertain to machine learning. We will also include some of the advancements in computer vision techniques that can be of use to the to be developed solution.

2.1 Gaming and sports video analysis

Many systems were created to detect events in video games. Some were implemented as fully automatic solutions and some others as semi-automatic. Still, any of them aimed to extract the most valuable information possible at each moment and translate it into a concise textual description.

To extract all of that high-level information it is needed to collect low-level information and process them in a way that all together can translate to a particular high-level event. This low-level information can be denominated as Low-level features (LLF). These LLF allow to extract direct information from visual or audio elements, such as the progression of color in a certain element or the variation of noise in a certain frequency.

Identifying high-level information in gaming and sports videos follows some common steps. Just as described in [17] first, it is needed to identify low-level audiovisual and textual features present in the streaming feed. Second, a sequence of those features is grouped to identify the movement of the video game character, the round of the game played, or other audiovisual components. The results from those groups are denominated Mid-level representations (MLR), a representation of a higher level but that cannot be used to infer the events of the video. To infer those events it is needed some prior knowledge of the sport played in the feed. When the models from MLR are processed with that knowledge the result is high-level information, denominated of High-level semantics (HLS), where the semantic concepts are used to identify the events. For example, when the colors of the screen turn gray and the character has stopped moving, in a game of League of Legends, it means that the character that the gamer is manipulating has died and it will be revived, as is depicted in [14]. The process is depicted in Figure 2.1 for a better understanding of the steps and the different levels of information present in a sports video.

The following content presents a synopsis of the features present in video game feeds, as well as the strategies used to retrieve them. First, it introduces approaches made to sports videos that can be applied to video game feeds in the solution to be developed. Then it clarifies approaches already developed for video game feed and its system of event detection and highlights detection.

2.1.1 Visual Patterns Analysis in Sports videos

Extracting visual LLF involves obtaining information from pixel intensity values, edges, color histograms, and other elements. These features are identifiable for their ease of definition and extraction but possess limited semantic significance [18]. Nonetheless, relying solely on LLF is insufficient for the development of intricate projects, such as automatic event detection due to the semantic gap.

Therefore, to create a visual model to analyze a video game feed it is needed to rely upon MLR. Visual MLR are semantic concepts, which are derived from specific combinations of LLF. In sports videos those specific combinations are related to standard camera shooting style transmission practices and domain-specific objects and their locations [19], and in video games feeds are related to camera settings. Because the camera settings vary among individuals, the visual patterns are not identical but rather exhibit similarities.

Video event detection

Video event detection, highlight extraction, and summarization have been extensively researched over the years, especially regarding football. The Bagadus system [15] effectively integrated data from multiple cameras installed in a stadium and data from sensors on football players. It provided a real-time interaction subsystem for experts to annotate football events. This system allows users to track specific players, view events in panorama videos, and create video summaries. Due to the high cost of expert annotations, it was developed another similar system but instead, it uses crowdsourcing to integrate annotations from crowd workers [20]. A Bayesian network-based method proposed in

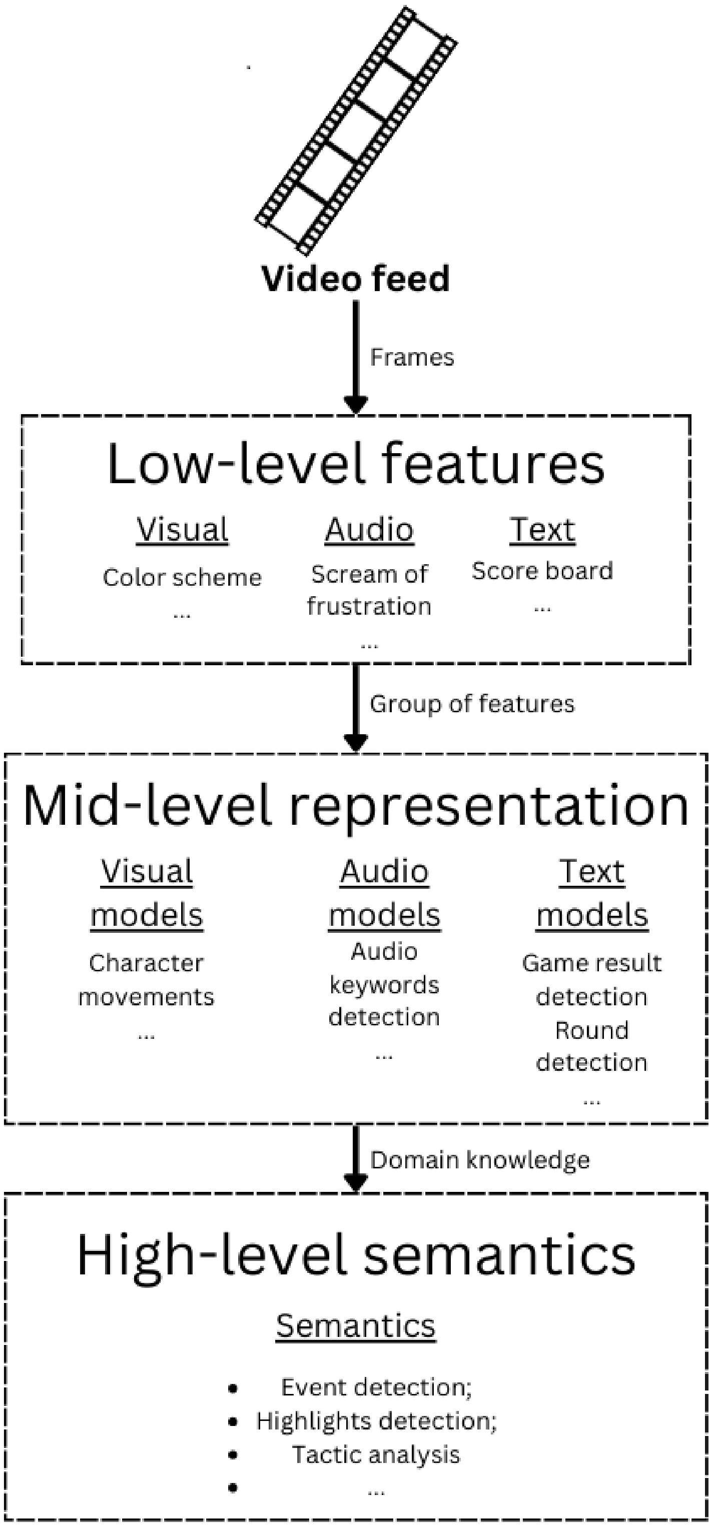


Figure 2.1: Relation between low-level features and high-level semantics (adapted from [17]

[21] captured dependencies among extracted features based on the automatically learned joint distribution of variables for football video event detection.

In addition to conventional highlight events like goals and penalty kicks, in [22] is proposed to include scenes of intense competition and emotional moments in football video summaries. They measured the interest level of a video clip based on cinematographic and motion features. In this system, the feature analysis consisted of identifying the movement of the ball and the impact that movement had on the flow of the game.

Regardless of baseball videos, in [23] the developed system aligned high-level webcast text with video content to avoid unstable performance caused by purely content-based methods. In this system, the feature analysis consisted of identifying visual features like camera movement, player movement, and ball movements, and combined with webcast text and text detected in the video resulted in tagging the pitching types.

In the area of basketball broadcast videos, it was developed a professional-oriented system for analyzing broadcast basketball videos [24]. This system introduced a robust player tracking system and adopted player trajectories to detect highlight events and conduct tactic analysis. This system consisted of using background subtraction, edge detection, and object tracking algorithms to detect the players' movements. After the features were obtained, they were filtered using particle filter algorithms to obtain a player tracking system. Finally, the motion analysis module analyzes it to infer statistical analysis of player movement, and events like shots made and to reconstruct the game in a 3D world simulation. In addition to the player tracking system, another system was proposed as a framework for basketball videos involving scoreboard detection, webcast text/video alignment, and replay detection, formulated as a discrete optimization problem compared to similar systems that analyze every video frame for semantic events [25].

2.1.2 Audio Patterns Analysis in Sports videos

The audio component of a sports video such as the voice of the commentator, music, and various kinds of environmental sounds is a very important type of media and very correlated to the video component. Various previous works on audio analysis have been developed to complement the visual component, such as [26][27][28].

The most basic concept used in audio pattern analysis is the audio keyword. This concept refers to any sound that is used in a game environment to describe the actions of players, referees, commentators, and audience [28]. As depicted in Figure 2.1, audio keywords can represent aural MLR, which combined with some domain knowledge can construct the HLS. These representations can be generated from different audio LLF and supported by algorithms, such as those based on Zero-Crossing Rate (ZCR) or Mel-Frequency Cepstral Coefficients (MFCCs). In [29] is proposed an audio keyword generation system with an adaptive HMM to correct a previous system that compromised of a hierarchical SVM classifier where a frame-based classification followed segments of frames of 20 ms. In this previous system, for diverse sports games, the task involved gathering substantial audio samples and manually labeling them to train keyword recognizers, thereby enhancing performance across a range of sports activities.

As shown in 2.2, the proposed system consists of the following steps to create HMM probabilities. The LLF, represented on the *Observation Vector* of the figure, such as the energy and the power of a signal, and pitch features including the fundamental frequency and harmonic frequencies of a signal, are extracted and tokens are added to create an observation vector. Next, the training of the HMM models is performed using dynamic programming. With those HMM models obtained the next step is to select testing data to compare with the incoming audio testing. The testing audio sequence is labeled as a series of predefined audio keywords according to the maximum posterior probability.

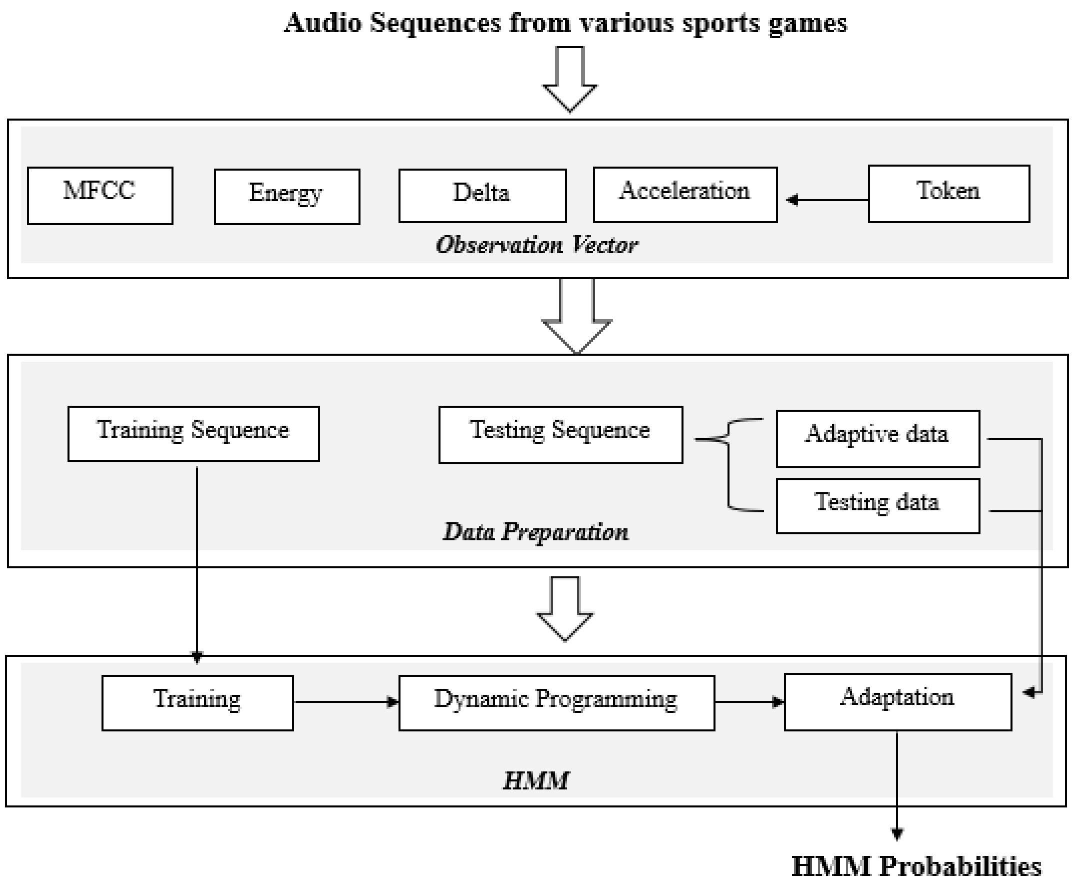


Figure 2.2: HMM-based audio keyword generation system (adapted from [29])

This system was then put to the test. The experimental audio data came from tennis, soccer, and basketball game videos with a total length of 2 hours. To analyze the influence of different HMM structures and different sample sequence lengths on the performance of the systems it was conducted some experiments, resulting in a 4-state left-right structure. For audio keywords that were of short duration, like whistling and the rebound of a ball in the courts, a length of 0.2 seconds was chosen, but for those related to speeches and interactions of the audience, it was selected a length of 1 second. The ones with a longer length are closely related to moments of high intensity and excitement, so it is used for highlight detection, thus the demonstration emphasized the generation of excited commentator speech and excited audience. In Table 2.1, were listed some partial experimental results that show the efficiency of this system when compared with the previous system (frame-level SVM keyword classification) in Basketball videos. The improvement of 5% in the recall and precision of those keywords related to highlight detection indicates a better performance of the newly developed system.

Audio keywords	Methods	Recall (%)	Precision (%)
Whistling	SVM	99.45	99.45
	HMM	100	100
Audience	SVM	83.71	79.52
	HMM	95.74	95.74
Commentator	SVM	79.09	78.27
	HMM	98.04	94.34
Excited Audience	SVM	80.14	81.17
	HMM	85.71	85.71
Excited Commentator	SVM	78.44	82.57
	HMM	86.67	100

Table 2.1: Comparison between HMM and SVM in detecting some of the audio features present in a basketball video (adapted from [29])

2.1.3 Textual features Analysis in Sports videos

The OT present in sports videos can be divided into scene and superimposed text [19]. The former refers to text that appears naturally in the recording environment, such as names on players' jerseys, slogans on banners, the result of the game, and text messages identifying that a character of a player has been slain by another [19] [14]. The latter is text artificially overlaid onto the video using dedicated hardware and/or software, such as the chat where the viewers and streamers interact as depicted in Figure 1.1 or the result of tagging webcast text in Baseball videos [23] or information about the rules of the game [30].

The article [30] proposes a framework for event detection in American Football and Soccer videos that utilizes both audio and visual features analysis combined with a system to extract information from external sources. The visual feature analysis is used to detect players' trajectories, positions, and interactions, as well as the ball. Combined with the audio features analysis that captures the commentary, crowd noise and players' verbal interactions creates the Audio-visual (AV) features analysis. The authors also proposed to incorporate external information sources to improve the event detection accuracy. These sources vary from the knowledge base of sports rules to rule out certain events as improbable or impossible, to the identification of the players that are actively on the court to rule out players that cannot be in the field and therefore cannot be tracked. To combine the AV features with the external sources of information the system proposed refers to a

Fusion of features (FoF) and to a Fusion of Decisions (FoD) processes. The FoF process refers to the process of combining the AV features with the external sources before sending them as inputs to a classifier, such as Fisher's Linear Discriminant algorithm [31]. The FoD process refers to combining individual classifiers, such as stacked SVM and bagging algorithms. A third system is also proposed, the fusion of rank lists that uses a proposed algorithm of the article [32].

The work in [33] presents a solution to OT Recognition applied to Football keywords, a solution fraught with challenges such as motion blur, occlusion, and variations in font and text size. The proposed method involves determining the location of OT, extracting its bounding rectangles, and then pre-processing them to mitigate noise and enhance contrast. Subsequently, a text detection algorithm is employed to pinpoint and extract the overlaid text. This process involves the application of the Sobel operator to the original frame to identify edges and boundaries. The complete process is described in Figure 2.3.

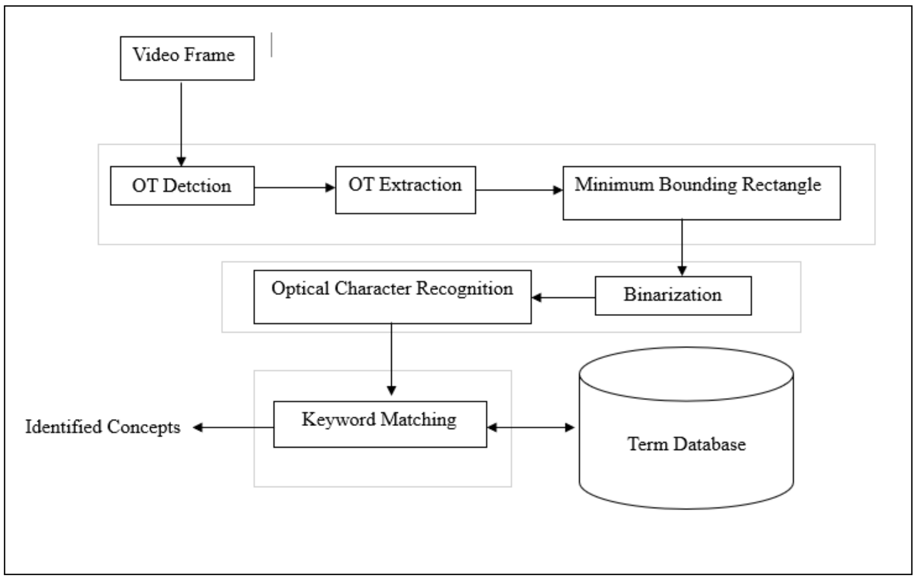


Figure 2.3: Complete process of OT extraction and recognition (adapted from [33])

The Sobel operator [34] is a renowned technique for edge detection. It effectively identifies the sharp transitions in brightness that delineate the boundaries between objects and regions within an image. It utilizes two 3x3 masks, one for horizontal and one for vertical edges. Each mask contains positive and negative values arranged in a specific pattern that emphasizes the intensity gradient along a particular direction.

Subsequently, the obtained image undergoes element dilation, binarization, and morphological opening processes [35]. Finally, the textual information is encapsulated by boxes through component analysis.

With the regions of OT already identified, it is now necessary to apply a technique to recognize the text. The technique used in the article is the OCR technology [36]. This technology is responsible for converting the image of a text into machine-encoded text, and its performance is increased accordingly with an increase in the contrast between the background and the letters, thus the use of the process of binarization [37].

The devised approach to extract and identify overlaid text in soccer videos yields a collection of textual attributes suitable for event detection. These features, acquired using OCR technology, can be cross-referenced with a predefined set of soccer-related keywords. Furthermore, the OCR system can identify numerical values within the score marker, offering a valuable verification mechanism for event detection. This facilitates a precise determination of when significant events, such as goals or points, take place during the match.

2.2 Event detection and Highlight detection in Video games

In this section, we will present two solutions for event detection and highlight detection in Video games. One uses a tool adopted by industry-leading graphics engines such as Unreal Engine 4.18 and Unity 5.6 to capture and share a gamer's best moments [38] and the other an automatic feature analysis approach [14].

Game engine integrated tools - NVIDIA Highlights

NVIDIA Highlights is a feature of *GeForce Experience* [38] that automatically captures and saves in-game highlights for compatible games, and this cutting-edge highlight recording tool has been crafted through collaboration between NVIDIA and top-tier graphics engines in the industry. This tool empowers game developers to indicate key highlights during gameplay, triggering the creation of a video sequence from the player's viewpoint. Integrated seamlessly into Unity and Unreal, NVIDIA Highlights can also be implemented in other engines using their respective SDKs.

Automatic feature analysis approach

This article [14] proposes a framework for event detection and highlight detection in League of Legends (LOL). In video games like LOL important events can be identified through all the types of LLF. To infer the game progress of each session the system developed uses a similar approach as [33]. For each video frame, is applied a preprocessing to detect Regions of interest (ROI), the Sobel edge detector to detect all the edges, and binarization to filter out the weak ones. Next, the morphological operations such as dilation and erosion filter out more weak edges and minimize the number of bounding boxes by eliminating the ones that are smaller than the minimum. Finally, it is employed the Tesseract OCR package to recognize text in each detected bounding box. When comparing the text resulting from the OCR with the set of predefined sentences from the table with the domain knowledge it results in a non-match that box will be discarded. But when it matches the text will be used to represent an event.

The same article also depicts a system to detect highlights in a video game feed. This system extracts features from both the video and viewers and uses them to construct models based on a psychological approach and a data-driven approach, respectively. From the video segment, it is possible to extract features like motion intensity that increases accordingly with the peak interest that visual content can bring, frame dynamics that indicate possible visual effects like when invoking an ability, number of player characters that might indicate a group fight between the players, event ratio that indicates that if multiple import events are occurring in such short time it means that must be a highlight in creation, number of viewers chat that indicate that if a burst of chats messages occur it means that it is occurring a highlight, and the same applies to emojis in the chat.

Now, with a greater understanding of the current state of the art in event detection systems and computer vision algorithms, the foundational knowledge required for system development has been established. It will then be presented to its development steps, obstacles, and solutions, as well as the final results of this development.

3 Videogame event detection

We will commence the system description upon grasping the fundamental concepts related to video analysis and image processing techniques. We will provide a detailed explanation of our developed work, elucidating the high-level characteristics extractable from the video game stream and demonstrating how this information can be accessed in real-time. All videos will adhere to 1920x1080 resolution, 16:9 aspect ratio, and 60 FPS. Content will be sourced from Twitch streams, following the structure outlined in Figure 1.1 and detailed in Section 1.2. The developed system was written in the C++ programming language and used the OpenCV framework and its associated resources.

The videogame stream chosen as the target of our system to be developed is LOL [39], a Multiplayer Online Battle Arena (MOBA) game developed and published by Riot Games, more specifically the international tournament "2024 Mid-Season Invitational" [40] that occurred between the dates of May 1st and 19th, stream worldwide through the official channel of Riot Games. Right now, this is the most-viewed video game and one of the ones with the biggest prize pools [41].

It is essential to explain the system's operation before going into great detail about each milestone that comes out in the developed system. The system may be visualized in a flowchart, as depicted in Figure 3.1, where each process is a milestone that will be deconstructed in the following sections. The system accepts two files as input: the Configuration file, which contains any variables required for the system's proper operation, and the Video stream file to be examined. And as output, there is the Interactions file, which contains every interaction gathered, as well as a file containing all of the events registered during the stream.

Similarly to the order of phases that takes to play one round of the game LOL, our system starts by searching for the starting point of the CHAMPIONS PICKS phase. During this phase, each team is responsible for selecting their champions and excluding some options for the other team. With the champions determined, the system searches for the start of the round play phase, in which the teams will compete for the round win. In this ROUND phase, the system will look for most of the events (kills, destroyed turrets, replays, and the round's end). Following the end of the gameplay, the algorithm passes over the stream frames to begin searching for the CHAMPIONS PICKS phase, again completing the cycle.

While part of the system searches for events on the gameplay, the other part collects the interactions of the viewers displayed on the Twitch chat, which corresponds to the process of "Chat interactions processing and collection".

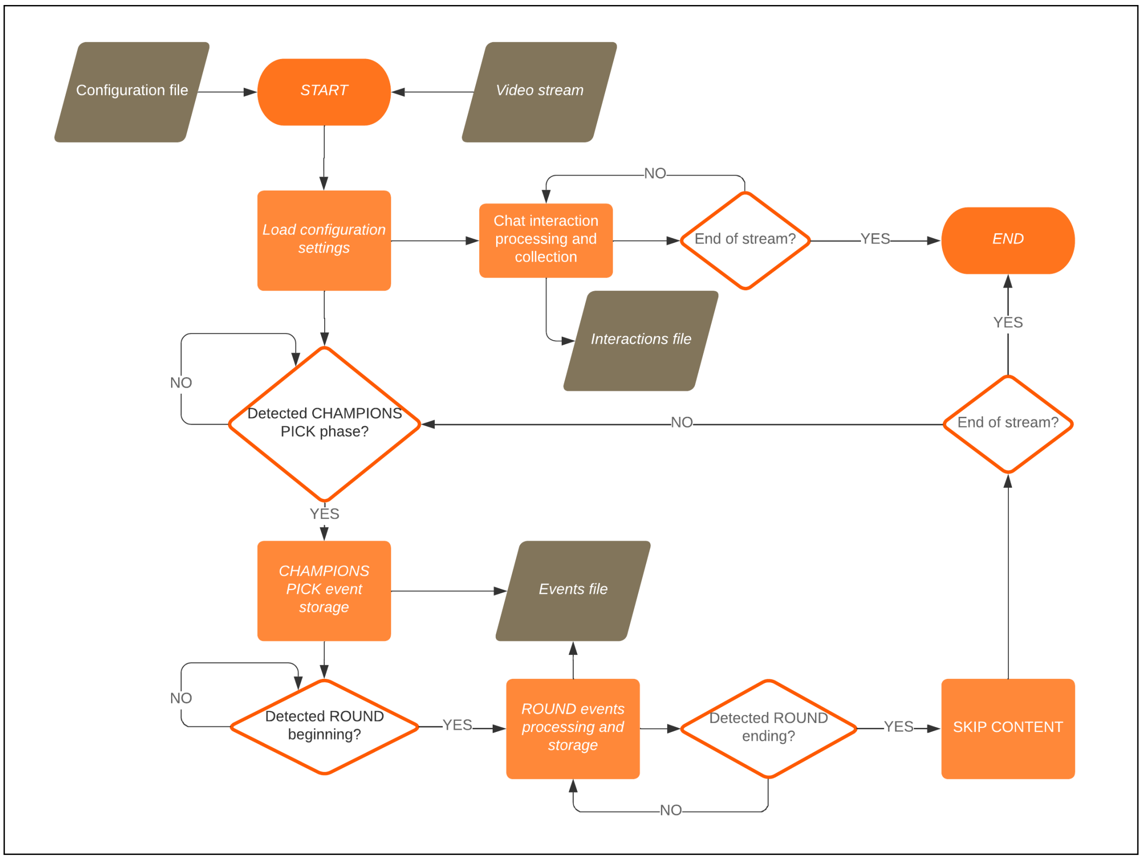

Configuration file
START
Video stream
Load configuration settings
Chat interaction processing and collection
NO
End of stream?
YES
END
Interactions file
YES
NO
Detected CHAMPIONS PICK phase?
NO
End of stream?
YES
CHAMPIONS PICK event storage
Events file
NO
Detected ROUND beginning?
YES
ROUND events processing and storage
Detected ROUND ending?
YES
SKIP CONTENT
NO

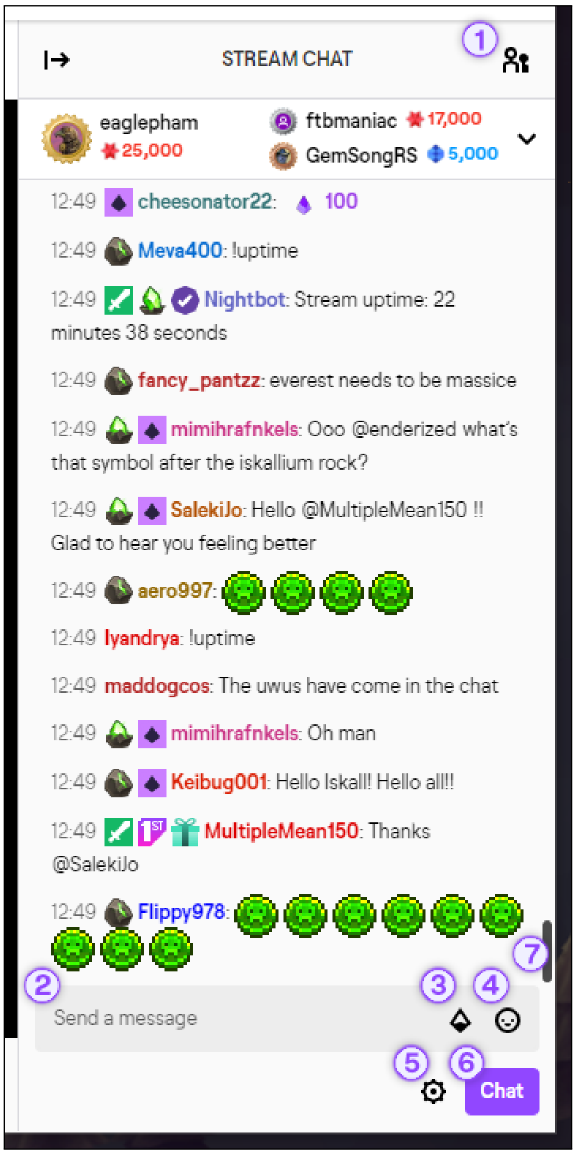
Figure 3.1: High level Flowchart.

To focus only on the necessary information of the stream to its contents it was necessary to separate the gameplay stream from the chat stream by cropping the images analyzed, separating the chat images and the gameplay images. The initial milestone focuses on extracting the viewer's interactions from Twitch stream chat, and it corresponds to the process of "Chat interactions processing and collection".

3.1 Chat Interactions

The Twitch platform's video broadcast structure comprises five essential components, detailed in 1.2. Among these, the chat function serves as a vital tool for identifying significant moments in gameplay, enabling interaction between streamers and viewers. Chat interactions often serve as descriptors of pivotal moments; for instance, multiple congratulatory messages may indicate a noteworthy gameplay segment.

The chat component can be further segmented into a header labeled 'STREAM CHAT,' an input box for viewer participation, and the ongoing interactions between viewers and the streamer, as depicted in the article [42]. Each interaction within the chat can also be segmented into three key components, as illustrated at the top of Figure 3.3: the timestamp denoting the stream time of the interaction, the username of the participant, and the message content, which besides the interaction text also feature emotes (small animations) often specific to the watched stream.



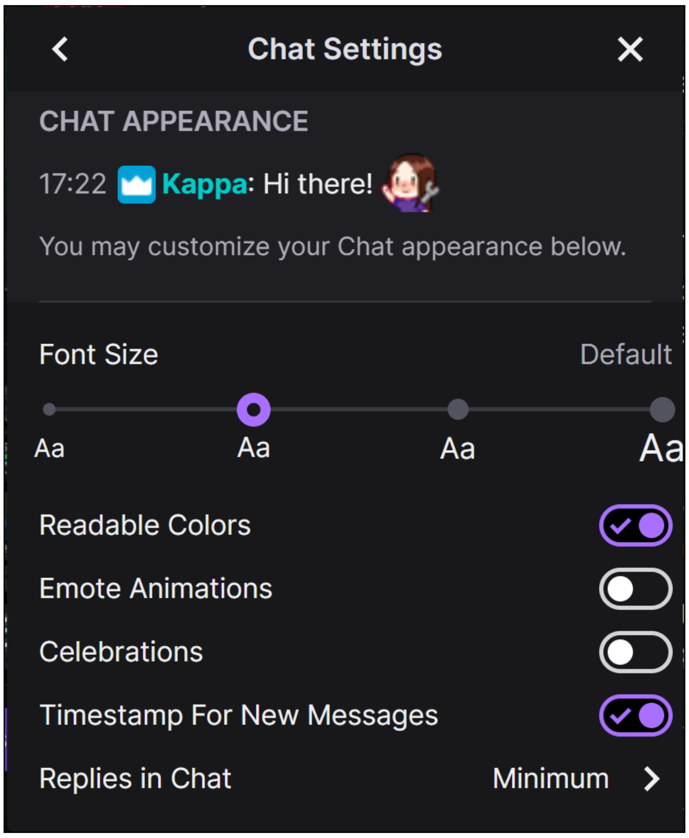
eaglepham 25,000
ftbmaniac 17,000
GemSongRS 5,000

12:49 cheesonator22: 100
12:49 Meva400: !uptime
12:49 Nightbot: Stream uptime: 22 minutes 38 seconds
12:49 fancy_pantzz: everest needs to be massice
12:49 mimihrafnkels: Ooo @enderized what's that symbol after the iskallium rock?
12:49 SalekiJo: Hello @MultipleMean150 !! Glad to hear you feeling better
12:49 aero997:
12:49 lyandrya: !uptime
12:49 maddogcos: The uwus have come in the chat
12:49 mimihrafnkels: Oh man
12:49 Keibug001: Hello Iskall! Hello all!!
12:49 MultipleMean150: Thanks @SalekiJo
12:49 Flippy978:

Figure 3.2: Twitch chat layout overview (Chat Layout Overview image taken from [42]).

The following text will detail the approach taken to extract the chat interactions, which is visually illustrated in Figure 3.4.

The ongoing interactions are not always being updated seeing that not every game moment is worthy of an interaction, so the viewers do not output messages to the chat. To only extract the interactions of the chat when there is a new interaction added it is


Chat Settings

CHAT APPEARANCE

17:22 Kappa: Hi there!

You may customize your Chat appearance below.

Font Size: Default

Readable Colors: [✓ On]

Emote Animations: Off

Celebrations: Off

Timestamp For New Messages: [✓ On]

Replies in Chat: Minimum

Figure 3.3: Twitch Chat settings defined (screenshot of the Appearance Settings)[43].

necessary to identify when there is a chat update. For that challenge was developed an algorithm to compare two consecutive frames of the stream, based on the binary operation Exclusive-OR (XOR) [44]. The operation between these two successive frames highlights the differences and eliminates the similarities. This process generates a matrix of pixels, having differences highlighted in the values of the three Red-Green-Blue (RGB) channels, as depicted in Figure 3.5.The number of pixels highlighted versus darkened can be used to determine whether or not the chat was updated.

However, because emotes may be included in the interactions, the resulting matrix can occasionally not be entirely highlighted or darkened. As seen in Figure 3.5a, these emotes' movements can be perceived as variations in the XOR operation. As a result, a threshold has to be set to differentiate the conversation update from the occasional emotive movement. This threshold was derived using a combination of estimation and trial and error. In Figure 3.5 it is possible to observe the two possible results of the comparison. In Subfigure 3.5a is depicted a possible result of when there are no new interactions in the chat, and in Subfigure 3.5b when there are new interactions, and the differences are highlighted.

After determining when the chat was updated, it is necessary to extract the new interactions. The procedure entails separating each interaction independently and decomposing it into its constituents. As illustrated in Figure 3.6, each interaction size differs

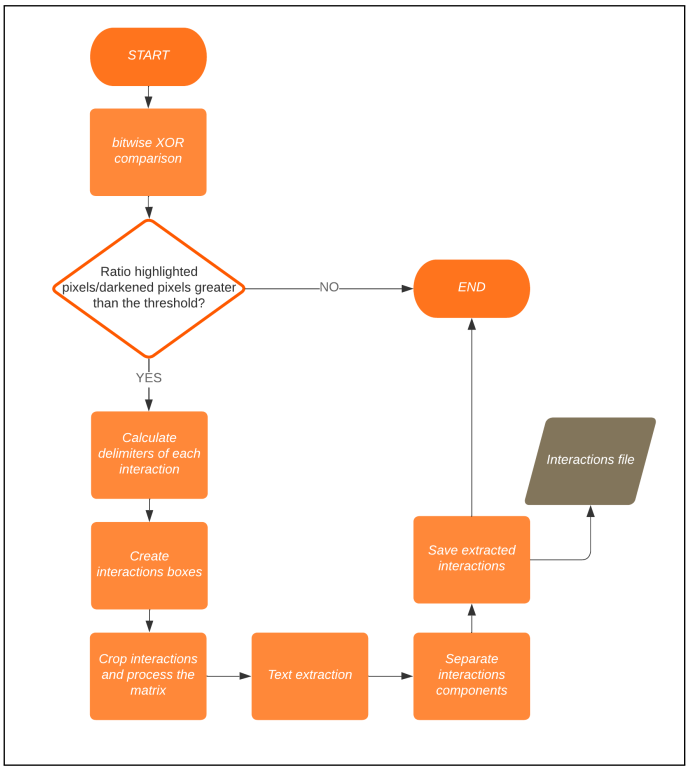

START
bitwise XOR comparison
Ratio highlighted pixels/darkened pixels greater than the threshold?
NO
END
YES
Calculate delimiters of each interaction
Create interactions boxes
Crop interactions and process the matrix
Text extraction
Separate interactions components
Save extracted interactions
Interactions file

Figure 3.4: Flowchart of the algorithm to extract the interactions from the Twitch chat.

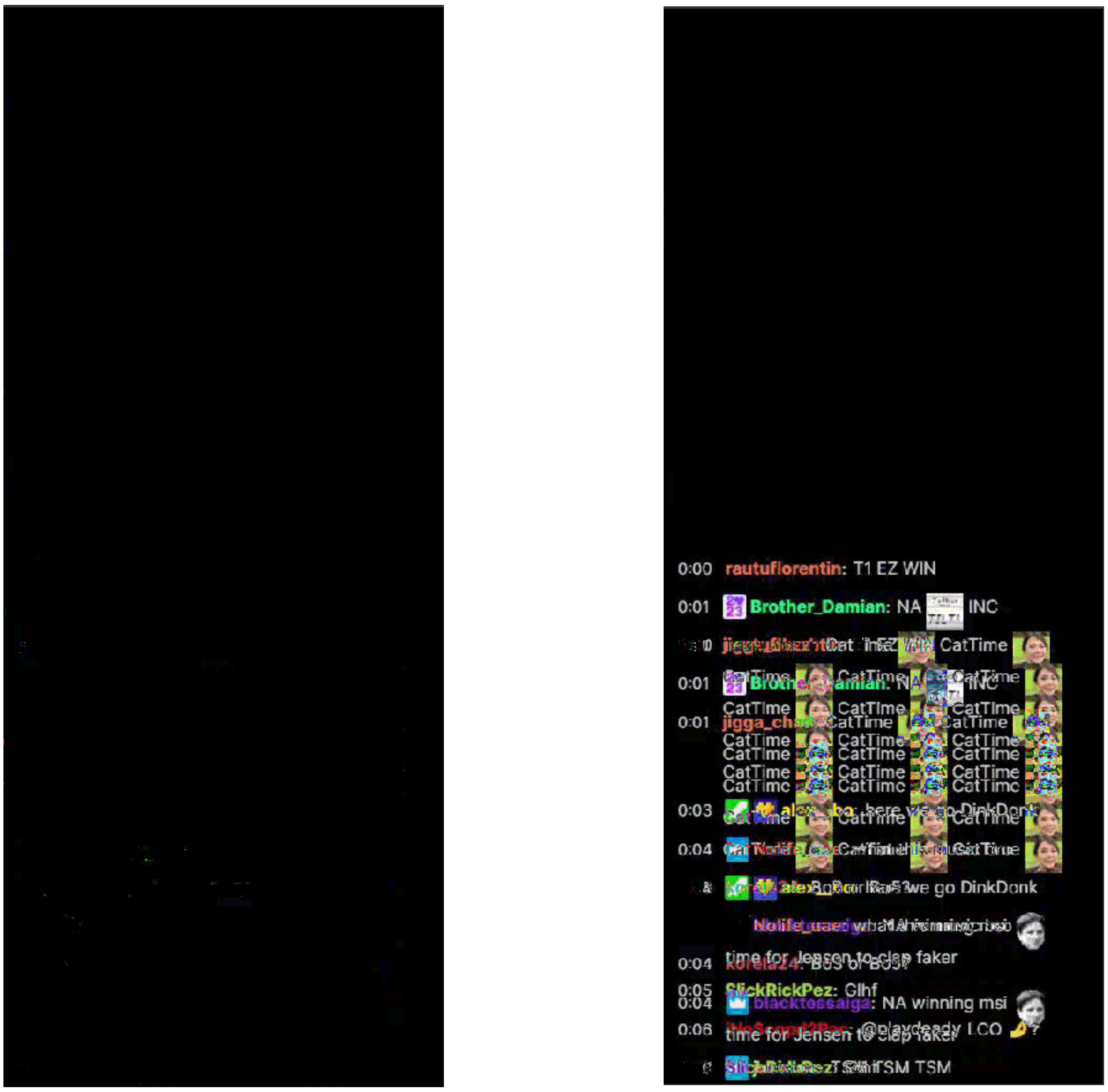
from the next, implying that separation should be done taking their sizes into mind. First, by analyzing the distance between the timestamps of two successive interactions, it is possible to determine the number of text lines in each interaction and the size of each interaction text box. An algorithm was developed to specify the limits of each interaction text box and produce bounding boxes for each interaction, as seen in 3.7a, based on that analysis. The technique works by evaluating the column of pixels at the x coordinates and counting the number of non-white pixels between the white pixels. When that number is compared to a previously estimated threshold, it is possible to determine whether it represents a gap between interactions or a gap between lines of the same interaction.

With each interaction well defined the next step consists of extracting the text with its multiple components and grouping them in a well-organized structure.

The algorithm developed to extract the text consists of preprocessing an image into a black-and-white image, with black text over a white background, followed by applying the OCR technology to extract the proper text. This preprocessing enhances the performance and efficiency of the OCR method.

The preprocessing consists of the following steps: resizing the grayscale converted interaction image using the method of bicubic interpolation followed by a binarization using Otsu's method to define the thresholds. The before and after processing images can be visualized in Subfigures 3.7b and 3.7c.

After converting the cropped image from the color space Blue-Green-Blue (BGR) to grayscale [45], the matrix was resized in a ratio of 3 in height and width for better


0:00 rautuflorentin: T1 EZ WIN

0:01 Brother_Damian: NA INC

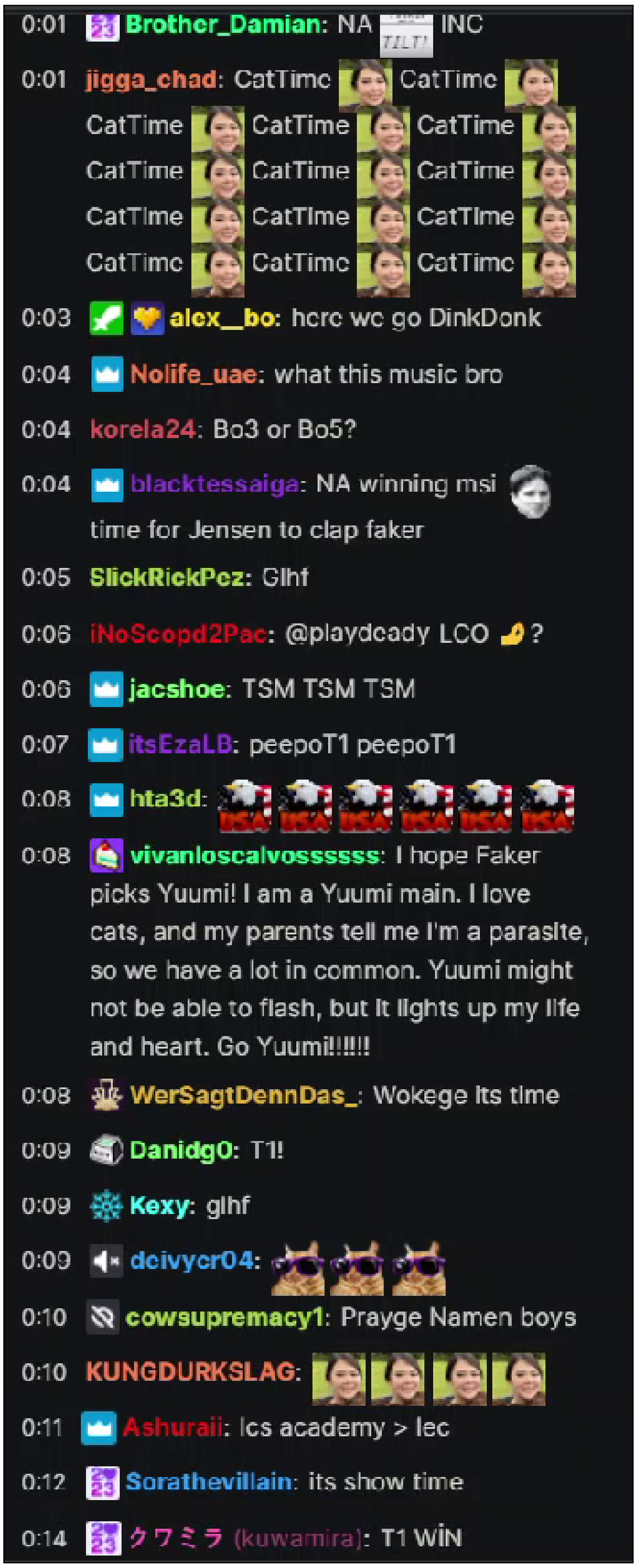
(a) New interactions - negative
(b) New interactions - positive

Figure 3.5: Results of the comparison between two consecutive frames.

performance of the engine later in the process. The enlargement of the matrix was done using the method of bicubic interpolation, due to its sharper results and the balance between processing time and output quality [46]. The interpolated value $f(x, y)$ of this technique is calculated using the Equation 3.1, where $I(x+i, y+j)$ is the value of the pixel at the position $(x+i, y+j)$ in the original image and $w(i, j)$ are the bicubic interpolation weights based on the distance from the interpolated point (x, y) to the neighboring pixel $(x+i, y+j)$:

$$f(x, y) = \sum_{i=-1}^{2} \sum_{j=-1}^{2} w(i, j) \cdot I(x+i, y+j) \tag{3.1}$$

Now that the image is represented by intensity (gray image), it is necessary to binarize



0:01 Brother_Damian: NA INC

0:01 jigga_chad: CatTime CatTime CatTime CatTime CatTime CatTime CatTime CatTime CatTime CatTime CatTime CatTime CatTime CatTime

0:03 alex__bo: here we go DinkDonk

0:04 Nolife_uae: what this music bro

0:04 korela24: Bo3 or Bo5?

0:04 blacktessaiga: NA winning msi time for Jensen to clap faker

0:05 SlickRickPez: Glhf

0:06 iNoScopd2Pac: @playdeady LCO ?

0:06 jacshoe: TSM TSM TSM

0:07 itsEzaLB: peepoT1 peepoT1

0:08 hta3d:

0:08 vivanloscalvosssssss: I hope Faker picks Yuumi! I am a Yuumi main. I love cats, and my parents tell me I'm a parasite, so we have a lot in common. Yuumi might not be able to flash, but it lights up my life and heart. Go Yuumi!!!!!

0:08 WerSagtDennDas_: Wokege its time

0:09 Danidg0: T1!

0:09 Kexy: glhf

0:09 deivycr04:

0:10 cowsupremacy1: Prayge Namen boys

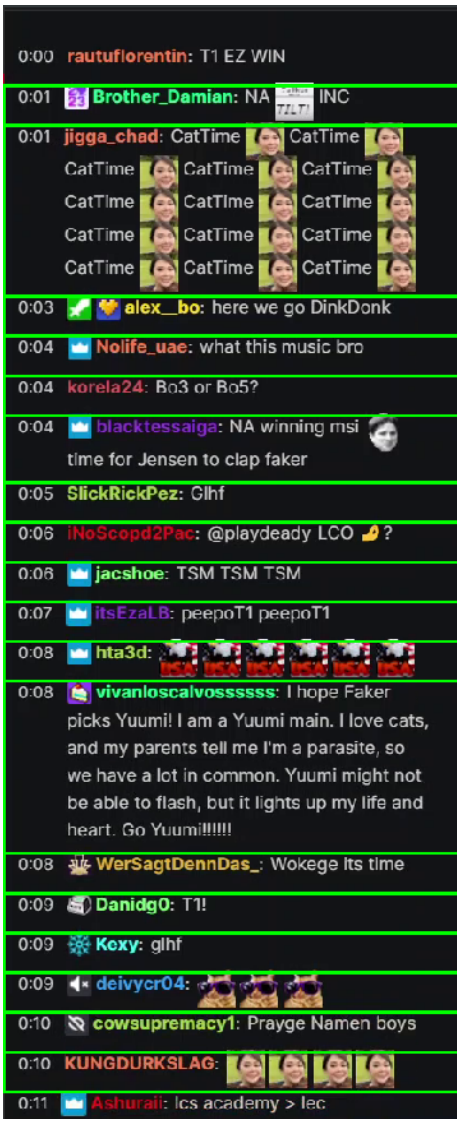
0:10 KUNGDURKSLAG:

0:11 Ashuraii: lcs academy > lec

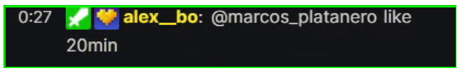
0:12 Sorathevillain: its show time

0:14 クワミラ (kuwamira): T1 WIN

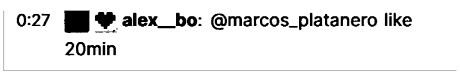
Figure 3.6: Example of multiple interactions depicted in the Twitch chat.


0:00 rautuflorentin: T1 EZ WIN
0:01 Brother_Damian: NA INC
0:01 jigga_chad: CatTime CatTime CatTime CatTime CatTime CatTime CatTime CatTime CatTime CatTime CatTime CatTime CatTime CatTime
0:03 alex__bo: here we go DinkDonk
0:04 Nolife_uae: what this music bro
0:04 korela24: Bo3 or Bo5?
0:04 blacktessaiga: NA winning msi time for Jensen to clap faker
0:05 SlickRickPez: Glhf
0:06 iNoScopd2Pac: @playdeady LCO ?
0:06 jacshoe: TSM TSM TSM
0:07 itsEzaLB: peepoT1 peepoT1
0:08 hta3d:
0:08 vivanloscalvossssss: I hope Faker picks Yuumi! I am a Yuumi main. I love cats, and my parents tell me I'm a parasite, so we have a lot in common. Yuumi might not be able to flash, but it lights up my life and heart. Go Yuumi!!!!!!
0:08 WerSagtDennDas_: Wokege Its time
0:09 Danidg0: T1!
0:09 Kexy: glhf
0:09 deivycr04:
0:10 cowsupremacy1: Prayge Namen boys
0:10 KUNGDURKSLAG:
0:11 Ashuraii: lcs academy > lec

(a) Bounding boxes of multiple interactions


0:27 alex__bo: @marcos_platanero like 20min

(b) Single interaction cut using its bounding box


0:27 alex__bo: @marcos_platanero like 20min

(c) Single interaction processed

Figure 3.7: The sequentially numbered process images resulting from the algorithm to extract and process each interaction individually: **a -** Bounding boxes of the interactions; **b -** A single interaction cut by its bounding box; **c -** The same interaction processed to be read by the OCR engine.

it. The pixels whose intensity falls short of the threshold are turned white and those that have higher intensity than the threshold are turned black. Given that Twitch chat has a black background with white text, the threshold type that must be used is the inverse binary. This type of threshold applies the threshold described previously but inverted. Instead of turning the pixels black when the intensity is higher than the threshold, it turns it white. The threshold value for this binarization can not be static. In parallel to the threshold behavior described above, it is also applied Otsu's method [47][37][48], which calculates a threshold value from the image histogram for a bimodal image. This method tries to find a threshold value t which minimizes the weighted within-class variance given by the relation of Equation 3.2:

$$\sigma_w^2(\mathrm{t}) = \mathrm{q}_1(\mathrm{t})\sigma_1^2(\mathrm{t}) + \mathrm{q}_2(\mathrm{t})\sigma_2^2(\mathrm{t}) \tag{3.2}$$

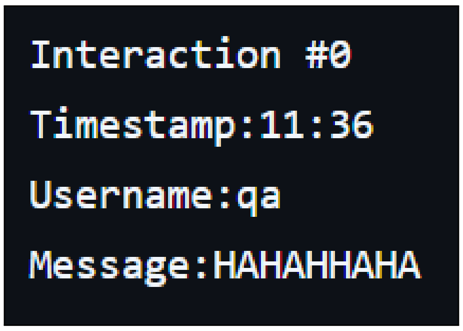
where q_1 and q_2 are the probabilities of the two classes divided by a threshold t, whose value is within the range from 0 to 255 inclusively, and μ_1 and μ_2 are the means of the two clusters of the histogram, as σ_1 and σ_2 are its variances.

$$\mathrm{q}_1(\mathrm{t}) = \sum_{i=1}^{\mathrm{t}} P(\mathrm{i}) \quad , \quad \mathrm{q}_2(\mathrm{t}) = \sum_{i=t+1}^{\mathrm{I}} P(i) \tag{3.3}$$

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} \quad , \quad \mu_2(t) = \sum_{i=t+1}^{1} \frac{iP(i)}{q_2(t)} \tag{3.4}$$

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i - \mu_1(t)]^2 \frac{P(i)}{q_1(t)} \quad \sigma_2^2(t) = \sum_{i=t+1}^{1} [i - \mu_1(t)]^2 \frac{P(i)}{q_2(t)} \tag{3.5}$$

Once the image is binarized, the text may be extracted using the OCR Tesseract engine [36], which converts the prepared images into textual information. This data is then processed to extract the various components of the interaction: timestamp, username, and comment. The separation is done based on the standardization of the interactions: First comes the timestamp composed of numbers and separated by ":"; next comes the username that always ends in another ":," which separates it from the remarks that follow it. As shown in Figure 3.8, these elements are then saved in a neatly structured text document, with a single field for each component.



```
Interaction #0
Timestamp:11:36
Username:qa
Message:HAHAHHAHA
```

Figure 3.8: Example of the structure of an interaction saved.

After collecting the chat images, the next stage is to analyze the gameplay snaps that show the current game stream. In this broadcast, the first gaming phase to be reviewed is the CHAMPIONS PICKS, which begins all of the games of LOL. The following section emphasizes the difficulty of identifying the start of this phase, the methods attempted, and the ultimate solution implemented.

3.2 Champions Picks

The champion picks phase in LOL is a critical part of the game that takes place before the match begins. This phase is often referred to as the *"draft phase"* and plays a crucial role in determining the game's strategies, team compositions, and potential outcomes. It is a process where each team selects their champions, the unique characters with distinct abilities that they will play during the match. This phase is separated into two parts: ban and picking. Such parts are done in an alternate picking and banning manner, as better explained on its official website [39].

In the context of the international tournament "2024 Mid-Season Invitational" [40], at the beginning and the ending of the "draft phase", the sponsors' logotypes are showcased in the lower part of the stream as a banner on the screen, as depicted in Figure 3.9.

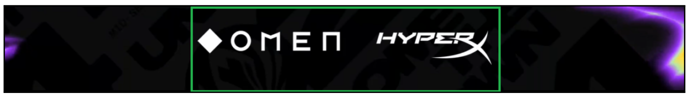


Figure 3.9: Sponsors' banner that pops up before the start of the Champions Picks phase.

The sponsors' banner indicates the beginning of this phase. Keeping this in mind, we begin by saving a cropped image of the banner, tighter around the logotypes, in its original dimensions and its coordinates in the stream, as shown in the green box of Figure 3.9. The choice to crop closer to the logos is to reduce the data processing and remove surrounding animations. Because the banner is static in position, there are numerous approaches to solving this problem. Specifically, we can detect when the banner is displayed by comparing the saved image to the cropped image of the stream at the previous saved coordinates. The adopted method is based on calculating the absolute difference norm of two arrays [44], or, more specifically, the magnitude of the vector in space carrying the differences between them. The magnitude can be determined using several mathematical definitions of distance, however, the one utilized in the method created is based on the Euclidean norm, depicted in Equation 3.6, where src_1 and src_2 are the arrays of values to be compared.

$$norm = \|\mathrm{src}_1 - \mathrm{src}_2\|_{L_2} = \sqrt{\sum_I (\mathrm{src}_1(I) - \mathrm{src}_2(I))^2} \tag{3.6}$$

Due to potential interference affecting the stream, it is expected that the two images will not be identical in every respect. Therefore, it is essential to establish a threshold for acceptable variations in magnitude. Such interference may arise from various sources, including transmission errors, data compression, or signal degradation during processing. By defining this threshold, we can ensure that the deviations remain within an acceptable range, allowing for consistent and reliable interpretation of the images. The Figure 3.10 presents a flowchart of the developed algorithm.

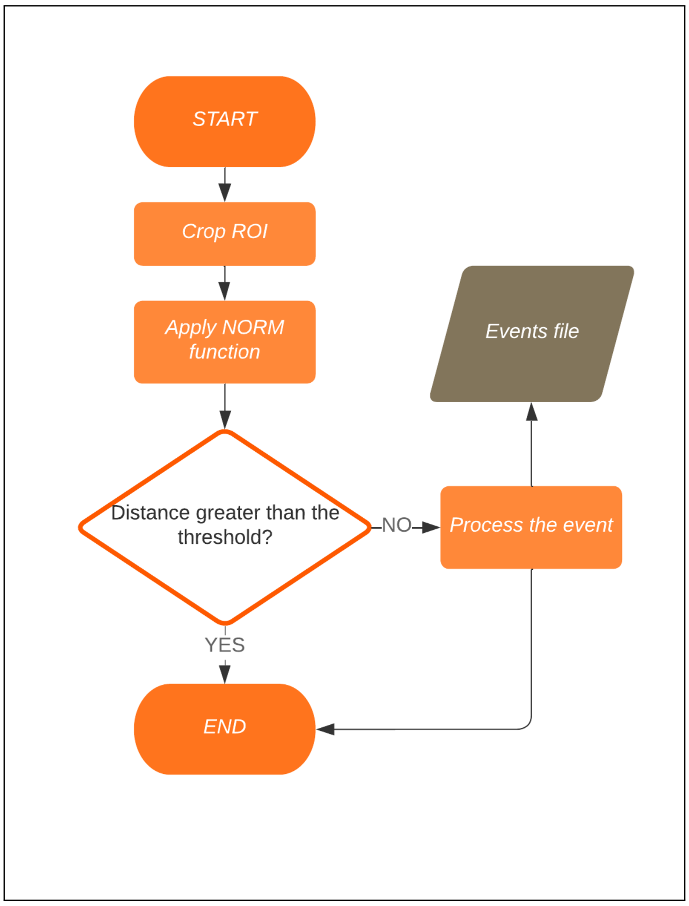


Figure 3.10: Flowchart of the algorithm to detect the start of the Champions Pick phase of the game.

Another approach used the Peak Signal-to-Noise Ration (PSNR) value to compare the similarities between the two images, and its value compared to a threshold. The engineering term PSNR refers to the ratio of a signal's peak signal intensity to noise strength. It is defined as the ratio of a signal's maximum attainable power to the power of noise that corrupts the signal and compromises its representational integrity. The PSNR metric is commonly used to evaluate the quality of photo compression and transmission techniques. The image quality improves as the PSNR value rises. This value is typically measured in decibels (dB).

To determine the differences between the current and prior frames, we must first calculate the Mean Square Error (MSE). MSE analyzes the two images to identify differences or dissimilarities. This is calculated by averaging the squared changes in pixel values between two photographs. The MSE is a scalar value ranging from zero to infinity. Lower numbers suggest a higher resemblance between the two photos.

MSE measures average pixel difference, while PSNR evaluates image quality compared to other images. Equations 3.7 and 3.8 combine to measure image differences more precisely and robustly. Let us consider two images, I_1 and I_2, each with a two-dimensional size of i and j and c channels.

$$MSE = \frac{1}{c \times i \times j} \sum (I_1 - I_2)^2 \tag{3.7}$$

$$PSNR = 10 \times \log_{10}(\frac{MAX_I^2}{MSE}) \tag{3.8}$$

The MAX_I represents the maximum acceptable value for a pixel. When two pictures are identical, the MSE returns zero, causing an erroneous division by zero operation in the PSNR formula. In this scenario, the PSNR is undefined and should be treated individually. Because the pixel values have such a broad dynamic range, they are converted to a logarithmic scale.

This is a more complex method than was needed. The *norm* method is more intuitive and sensitive to variances in perception.

After completing the *"draft phase"*, the next step is to determine the start and end of the game during which the selected champions will be used. The majority of the action will take place between these two events.

3.3 Round Begin and Round End

In the context of the video game LOL, the beginning of the round can be identified by the appearance of the scoreboard after the "draft phase." The scoreboard starts with its values reset, displaying the scores as zero and the team money at its initial value. Each side of the scoreboard shows the teams' logos, their acronyms, and the league in which they play, correlated to the region of the team. The scoreboard also indicates the number of rounds each team has won up to that point below their acronyms. The figure below (Figure 3.11) shows all of these details.

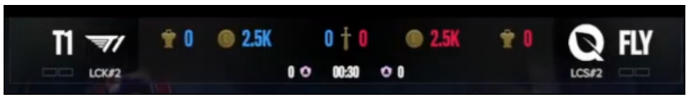


Figure 3.11: Example of a scoreboard at the beginning of a round

While the initial display of the scoreboard following the Champions Picks phase signals the start of the round, the banner describing the round's outcome signals the conclusion of the round. The banner shows the victorious team's logotype, the word *"VICTORY"* to its right, and a description of the game's state below. This description ranges from determining whether the winning team won the series, if the series is tied, or who is currently leading the series. The figure 3.12 shows an example of a banner describing the conclusion of the first round of a game, with team *T1* winning the round against the team *FLY (FlyQuest)* and leading the series by the smallest margin.

The algorithms developed to detect both events rely on comparing two cropped images of the regions of interest. The area on the scoreboard is used to determine when the round officially begins. The portion of the banner that reads "VICTORY" is used for the round

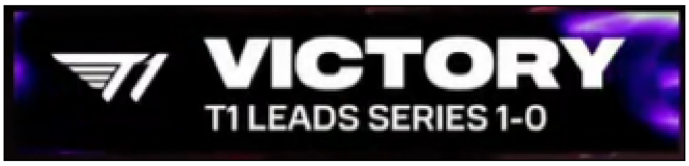


Figure 3.12: Example of a banner describing the conclusion of the first round of a game, with team *T1* winning the round against the team *FLY (FlyQuest)* and leading the series by the smallest margin. On the left side of the banner, it is displayed the logotype of the winning team. The example is relative to the end of the first round of the game between the teams *T1* and *FLY*, in which team *T1* wins the series with the result of 2 - 0 [MAY 3, 2024 - 09:00 WEST; Upper Bracket Final]

end. Due to the ever-changing remaining banner info, the algorithm is applied to a smaller region. The game state description, winning team, and logotype are always shifting. The adopted method is based on the PSNR value obtained from the equations 3.7 and 3.8. Similar to the algorithm developed to detect when the banner of the *CHAMPIONS PICKS* is displayed, by comparing the PSNR value to a predetermined threshold it is possible to infer when both of regions of interest are displayed.

Two cropped images of the regions of interest—one from the frame to be analyzed and the other from a template image that had previously been saved—as well as their coordinates provide the basis for the algorithms designed to identify both events. The static nature of the regions of interest's locations is considered by both algorithms. However, the distinction is that the image processing used for the cropped images is different in each instance. In the Round Begin scenario, the cropped image of the scoreboard must first have a mask applied to hide the statistics; however, in the Round End scenario, the cropped image can be passed on directly to the comparison using the PSNR formulas. The applied mask serves multiple purposes, including minimizing the amount of data to be processed and masking statistics that, by default, belong in the reset values. However, occasionally the round's transmission begins after a brief interval, which may impact the stats even when no relevant information is displayed. The template images are shown in the following figures (Figure 3.13 and Figure 3.14).

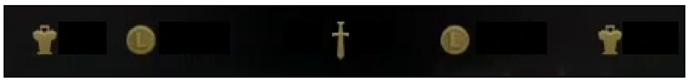


Figure 3.13: Template image for the identification of the beginning of a round (scoreboard with the stats covered)

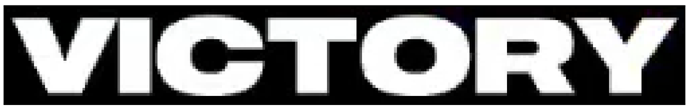


Figure 3.14: Template image for the identification of the end of a round.

A different method to identify these events was also tested. Instead of comparing two cropped images and calculating the similarities between them, this method is based on

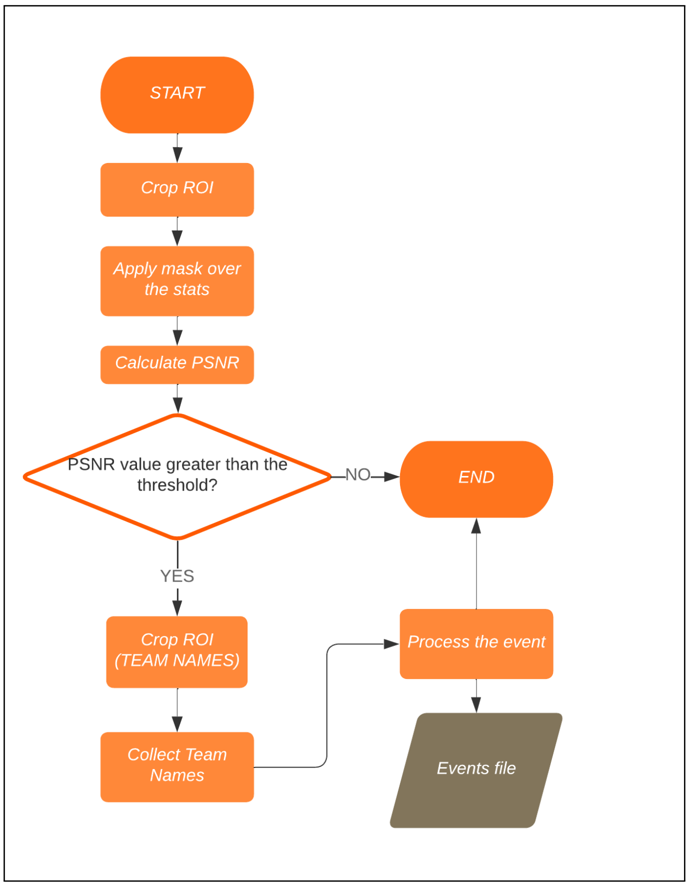


Figure 3.15: Flowchart of the algorithm to detect the start of the Round.

searching in the frame for the templates, more specifically the banner and the scoreboard. The method uses the Template Matching technique [49] to find areas of an image that match a template image. The logic behind this technique is that to identify the matching area of the image, it is necessary to compare the template image against the source by sliding over it. After every movement over the two axes, a metric is calculated and inserted into a matrix. After obtaining that matrix of values, depending on the metrics utilized, the lowest/highest value indicates the location of the better match. This method also requires defining a threshold value to determine the confidence of the match.

The PSNR-based algorithm has many advantages over the Template Matching technique. Among the advantages of the PSNR technique are: it consumes less calculation time and processes data faster because measurements don't have to be made for every slide. Another benefit is that the PSNR method is not affected negatively by any changes to the image, such as degradation or alterations.

It is possible to obtain further information about the round once the game's beginning has been determined. Which team is the red team and which is the blue team is that knowledge. The way the scoreboard information is read during the stream is impacted by this information.

A simple algorithm was developed to read its names from the scoreboard. The Blue

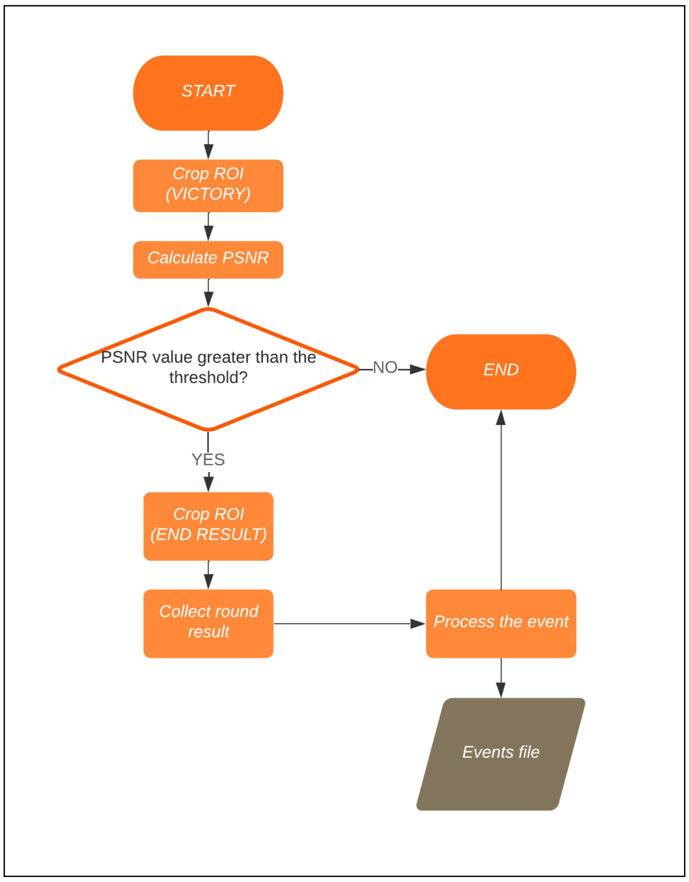


Figure 3.16: Flowchart of the algorithm to detect the end of the Round.

team is on the left side of the scoreboard, while the Red team is on the right, as seen in Figure 3.11. The algorithm consists on cropping the names from its static positions and read the texts by passing the cropped images through a neural network that has been trained to recognize characters and sequences. In order to use this Deep Neural Network (DNN)-based text recognition it is necessary to use a Application Programming Interface (API) and involves setting up models like Convolutional Recurrent Neural Network (CRNN) for recognition of the text. Every information and files necessary to the process are accessible in *OpenCV* documentation [50].

The process of setting up this API involves several important steps. First, we need to load the model weights by using an ONNX file of a pre-trained model available on the *OpenCV* documentation. The chosen model has a classification number of 94, which means it was specifically trained to recognize all the letters of the Latin alphabet, including both lowercase and uppercase letters, as well as numbers and certain punctuation marks.

After loading the model, the next step involves the decoding method. In this case, we are using the CTC-greedy method, which means that the output of the text recognition model will be a probability matrix. This matrix has a specific shape, denoted as (T, B, Dim), where T represents the sequence length, B represents the batch size, and Dim represents the length of the vocabulary plus one for the blank space.

Additionally, it is crucial to set up the vocabulary based on the specific requirements of the text. In this case, we ensure that the vocabulary includes all the alphabet letters, numbers, and only certain relevant punctuation marks.

Once all these steps are completed, we can infer the text using the API. The website [50] provides a fuller explanation of all the model files and information.

The decision to utilize this method for the text extraction of the teams' names, as opposed to the OCR engine, was based on several factors. This method demonstrated superior performance when dealing with small texts containing only a single word or a single group of letters. Additionally, it offered faster processing capabilities, making it the preferred choice for our specific requirements.

Similar to the scoreboard at the beginning of the round, the banner at the end also provides more information about the round, particularly the round's result. As shown in Figure 3.12, the banner displays a description of the round's score. For example, it shows that *T1* has taken the lead in the game with a victory.

The process of extracting the text of the description involves three distinct steps. Firstly, the image is cropped using pre-obtained coordinates. This is necessary due to the static nature of the sentence. Secondly, the cropped image undergoes the same processing as the chat, resulting in a binarized image ready for text extraction. Finally, the *Tesseract* engine and OCR technology are applied to convert the binarized image into textual information. This information is then stored and can be utilized in the creation of events, particularly for the conclusion of rounds.

The specific context in which the *Tesseract* engine was employed for text extraction involved the need to extract a sentence consisting of multiple words. When compared to the API method, the *Tesseract* engine demonstrated superior performance in this particular scenario. However, it is important to note that the engine's processing speed for the conversion task is slower. Despite this drawback, the slower processing capabilities are deemed necessary in order to achieve the desired level of performance.

Figure 3.15 and Figure 3.16 show the flowcharts that illustrate how the algorithm operates.

Once the round begins, it's important to identify important moments such as kills and turrets destroyed during the confrontation. It is not enough to just note when these events happen; it is also important to identify the initial actions that led to these outcomes. This detailed information is essential for understanding the game's progression. For viewers and analysts, these moments offer valuable insights into team performance and the application of strategies and tactics.

3.4 Kills and Turrets destroyed

When it comes to identifying the kills and turrets destroyed, the process begins with isolating specific areas of the image that contain the score information for each team. These areas are located next to the turret icons for the turrets destroyed score and a single sword icon for the kills of each team, as depicted in the Figure 3.17. The scores are updated whenever a kill or turret destruction occurs for either team. However, as the scores are not updated with every frame of the video, we have implemented a system to only check the scores every 60 frames or 1 second of video to reduce the overuse of resources unnecessarily.

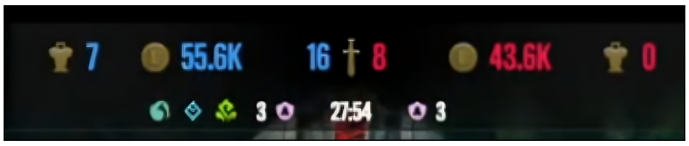


Figure 3.17: Example of a scoreboard during a round: The score relative to the blue team is displayed on the left side of the scoreboard, while the red team's score is displayed on the right. The number of turrets each team has destroyed is shown next to an icon of a turret, and the kills score is displayed on each side of the sword at the center of the scoreboard. Each score is written in the color of the respective team.

After cropping the images, the next step involves running them through a specialized neural network trained to identify and extract the text from the images. This process is akin to the one used for extracting the names of the teams in the previous section (Section 3.3), where an API is utilized to interpret the text within the image [50]. The key difference lies in the vocabulary used. Instead of considering any letter, number, and punctuation in the inference process, we restrict the choices to only numbers. This restriction allows the algorithm to perform better and reduces any possible misinterpretation of the score, such as interpreting the score through letters, but does not cover all the possible misreadings. During the process of text extraction, there is still a possibility of misreadings based on the numbers, such as mistaking a 1 for a 7 or erroneously detecting two consecutive 1's when there is only one in the image. To address these potential errors, extensive testing was conducted, and specific heuristics were developed to accurately handle such misinterpretations. All the text extracted is then stored and used to identify the events and to help the heuristics previously mentioned.

The algorithm was developed to extract scores from the scoreboard. However, there are instances when the scoreboard doesn't display scores but instead shows other game-related information such as the destruction of turret plates by each team, and the money earned from those destroyed plates. This can be observed in Figure 3.18. To distinguish when the scoreboard is displaying the required scores versus other information, it was utilized the same algorithm used to identify the start of a round. By identifying specific components of the scoreboard, such as the middle sword and the turrets, we can ensure that the extracted text relates to the kill score and the turret score.

To enhance the information available about these events, an algorithm was created to identify the the start of the play that ended in the event. This method identifies the most recent scene change and uses its timestamp as the start of the play. This task was accomplished by examining two successive frames for similarities. By calculating the PSNR between the frames, we may assess the similarities. If the value compared to a threshold indicates that the two frames are not identical, it suggests that there was a scene change.

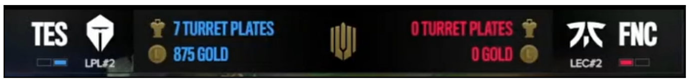


Figure 3.18: Example of a scoreboard during a round displaying the statistics of plates destroyed by each team: Destroying each turret plate grants gold to players, providing a significant early game advantage for the team that takes more plates.

Figure 3.19 illustrates the algorithm's flow of operation.

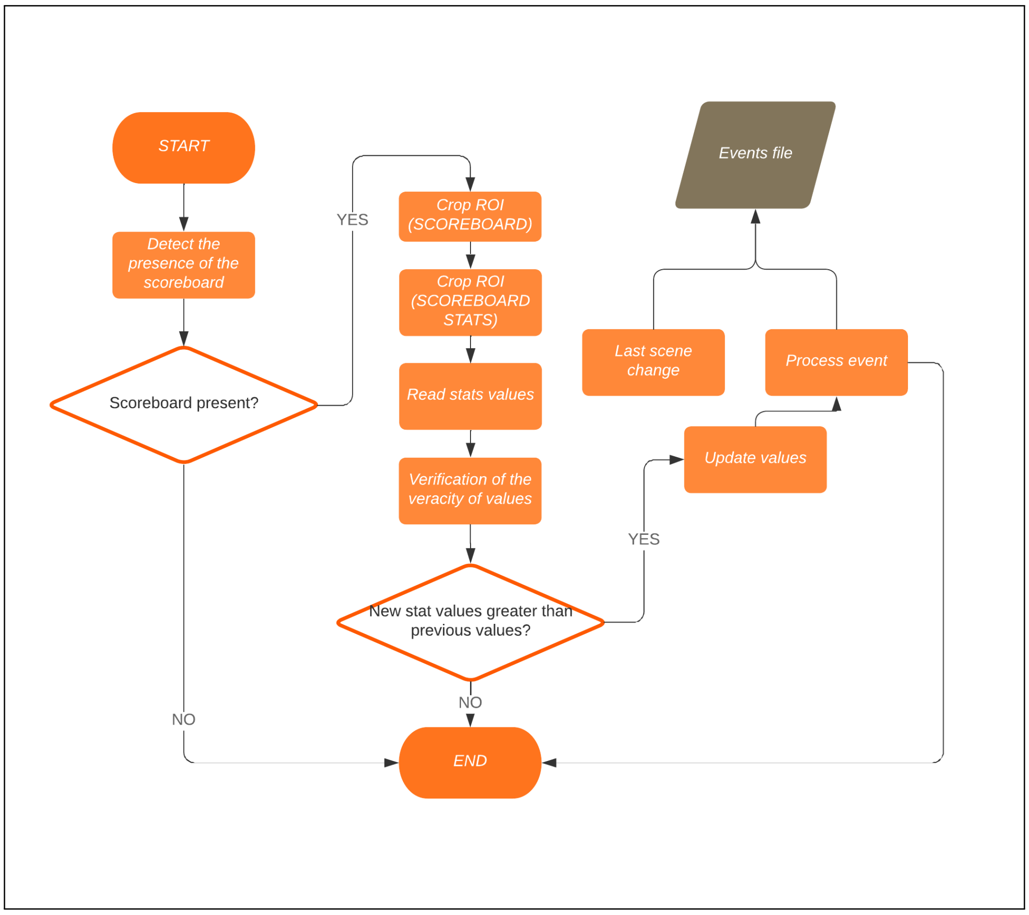

START
Detect the presence of the scoreboard
Scoreboard present?
YES
NO
Crop ROI (SCOREBOARD)
Crop ROI (SCOREBOARD STATS)
Read stats values
Verification of the veracity of values
New stat values greater than previous values?
YES
NO
END
Events file
Last scene change
Process event
Update values

Figure 3.19: Flowchart of the algorithm to collect the stats of the scoreboard (KILLS/TURRETS).

In live broadcasts of LOL, *RIOT GAMES* often highlights important moments through replays, adding excitement and value for viewers. These moments are often edited and may be shown using 3D animation. In the next section, we will introduce the algorithm we developed to detect and identify these moments during gameplay.

3.5 Replay

In the realm of professional League of Legends (LOL) matches and high-stakes tournaments, replays play a crucial role. They offer the audience the opportunity to revisit critical moments like team fights, objective steals, solo kills, and tower dives, which often sway the outcome of a match. Through replays, casters and viewers can delve into the intricate mechanics, strategic positioning, and decision-making that shaped these pivotal instances. Furthermore, in the fast-paced environment of a LOL match, there's a chance that the production team or the primary camera may overlook minor skirmishes, rotations, or cunning maneuvers while focusing on the primary action. Replays empower the production team to revisit and showcase these significant moments, ensuring that no important details are missed. Additionally, replays can seamlessly fill lulls in the game, such as when players retreat to base or during downtime between objectives, maintaining a dynamic and captivating viewing experience even during slower moments of live action.

In the context of the international tournament "2024 Mid-Season Invitational", a banner with the word "REPLAY" is displayed at the top left of the screen during the replay. This banner, set against a black background with small animations in the borders, is constantly shown while the replay is playing. As a result, it is possible to detect the start, end, and duration of the replay by simply tracking the presence of this banner on the screen. The banner is displayed in Figure 3.20.

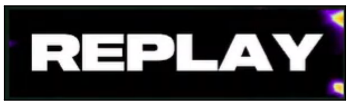


Figure 3.20: Template image for detecting the presence of the banner during replays.

The process begins by capturing and storing a cropped image of the banner along with its location in the stream frames. This stored image is then compared to the cropped image from the live frames to determine if the banner is present. This comparison method resembles the one used to identify the scoreboard. It involves calculating the PSNR value between the two images and then comparing it to a predefined threshold, obtained through trial and error. If the value exceeds the threshold, the Replay banner is being displayed. If the calculated value falls below the threshold, it indicates that the replay has ended and the live gameplay continues.

Figure 3.21 illustrates the algorithm's flow of operation.

Now that we have developed algorithms to detect and identify every event, we are ready to test them using live transmissions of the tournament on *Riot Games' Twitch channel*. This critical phase will validate the effectiveness and robustness of the algorithms, contributing to the successful achievement of our objectives in analyzing LOL game broadcasts.

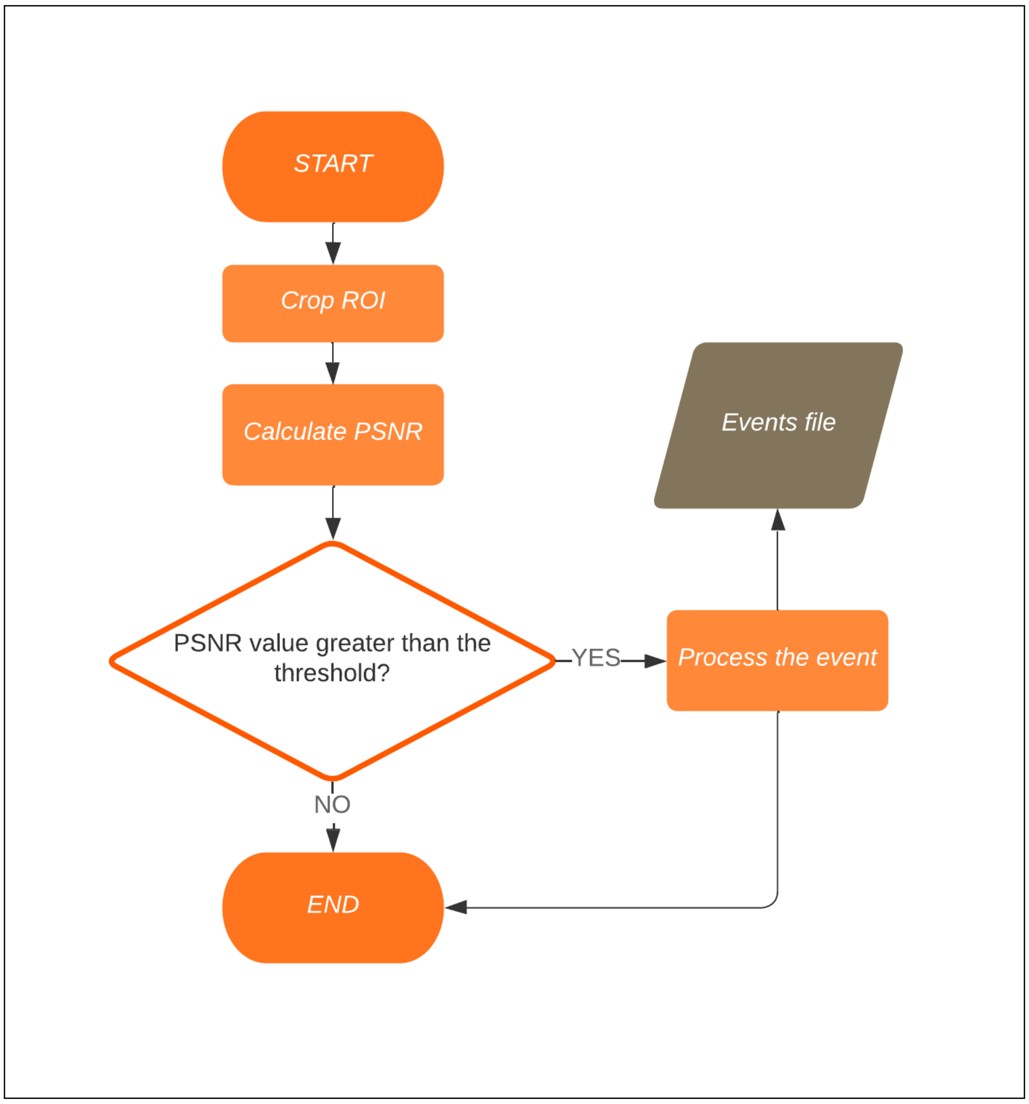
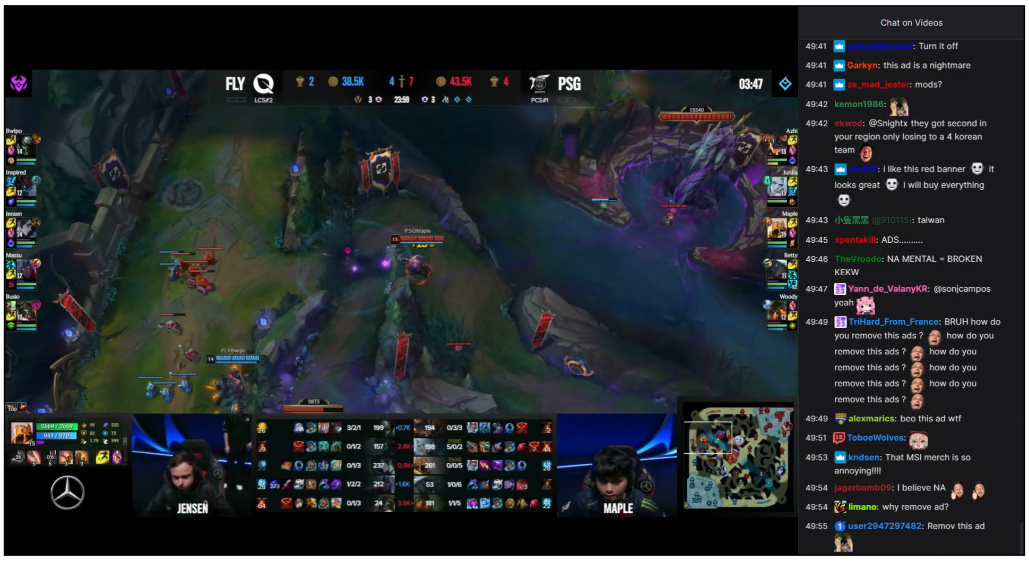


Figure 3.21: Flowchart of the algorithm to detect the presence of the REPLAY banner.

3.6 Final Tests and Conclusion

This section is dedicated to presenting the obtained results, analyzing them, and drawing conclusions about the proposed solutions for engineering problems in the context of the videogame of League of Legends (LOL).

In order to assess the quality of the developed algorithm, we decided to test it with three video streams from the "2024 Mid-Season Invitational" tournament [40]. Specifically, we tested it with the streams from days 3, 4, and 5 of the tournament, corresponding to May 3rd, 4th, and 5th, last three days of the Play-In stage of the tournament. All tests were conducted on a Linux environment on a computer with an Intel(R) Core(TM) i7-8565U CPU running at 1.80GHz and 16GB of RAM. The videos were recorded in MPEG-4 (.mp4) format at a resolution of 1920x1080 and an aspect ratio of 16:9, with a frame rate of 60 frames per second. In Figure 3.22, you can see the appearance of the recording with the display of the Twitch stream and structure, as analyzed in Figure 1.1 at Section 1.2.



49:41 qweqwdasassd: Turn it off
49:41 Garkyn: this ad is a nightmare
49:41 ze_mad_jester: mods?
49:42 kemon1986:
49:42 ekwod: @Snightx they got second in your region only losing to a 4 korean team
49:43 rAnEtE: i like this red banner it looks great i will buy everything
49:43 小隻黑黑 (jjj910115): taiwan
49:45 xpentakill: ADS..........
49:46 TheVroodo: NA MENTAL = BROKEN KEKW
49:47 Yann_de_ValanyKR: @sonjcampos yeah
49:49 TriHard_From_France: BRUH how do you remove this ads ? how do you remove this ads ? how do you remove this ads ? how do you remove this ads ? how do you remove this ads ?
49:49 alexmarics: beo this ad wtf
49:51 ToboeWolves:
49:53 kndsen: That MSI merch is so annoying!!!!
49:54 jagerbomb09: I believe NA
49:54 limano: why remove ad?
49:55 user2947297482: Remov this ad

Figure 3.22: Appearance of the recording: Twitch stream with the full display of the gameplay and chat.

The results of the tests are outlined in Table 3.1. Each column represents a stream, and the rows correspond to the events to be identified within each stream. The results will be presented in the format [(True Positive (TP) | False Positive (FP) | True Negative (TN) | False Negative (FN)) | real]. This structure helps to better understand the accuracy values (Equation 3.9) of the events detected. The Twitch Chat component presents the percentage of messages successfully extracted.

The presented value of TN is shown as 0 in all of the table, despite the expectation of a much higher value. This discrepancy arises because the system is designed to detect the presence of events rather than their absence. However, it is important to note that this value has minimal significance in the overall evaluation of the developed system.

Upon reviewing Table 3.1, we can discern promising outcomes for the automated detection of events. Notably, the algorithms for identifying the start and end of the Round exhibit flawless accuracy at 100%. However, the algorithm responsible for pinpointing the start of the Champions Picks phase shows a lower accuracy rate of 56%. In the realm of in-game events, the Kills algorithm demonstrates an impressive accuracy of 98%, while the Turrets destroyed algorithm follows closely behind at 92%. Moreover, the algorithm designed to detect the beginning of the gameplay leading to the aforementioned events of Kills and Turrets destroyed achieves an accuracy of 84%. Conversely, the algorithms tasked with identifying the start and end of the replay exhibit lower accuracies of 67% and 69%, respectively. We can confirm the viability of the system application in LOL live streams because the processing time for detecting all events and associated metadata is always close to half of the stream duration.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{3.9}$$

MSI 2024 Primer	DAY #3	DAY #4	DAY #5
Champions Pick	[(4 | 1 | 0 | 1) | **5**]	[(4 | 1 | 0 | 1) | **5**]	[(2 | 2 | 0 | 2) | **4**]
Round Begin	[(5 | 0 | 0 | 0) | **5**]	[(5 | 0 | 0 | 0) | **5**]	[(4 | 0 | 0 | 0) | **4**]
Round End	[(5 | 0 | 0 | 0) | **5**]	[(5 | 0 | 0 | 0) | **5**]	[(4 | 0 | 0 | 0) | **4**]
Kills	[(130 | 3 | 0 | 3) | **133**]	[(125 | 0 | 0 | 0) | **125**]	[(130 | 0 | 0 | 3) | **133**]
Turret Destroyed	[(50 | 2 | 0 | 2) | **52**]	[(59 | 1 | 0 | 2) | **61**]	[(49 | 2 | 0 | 4) | **53**]
Replay Begin	[(26 | 0 | 0 | 0) | **26**]	[(28 | 1 | 0 | 1) | **29**]	[(0 | 0 | 0 | 25) | **25**]
Replay End	[(26 | 0 | 0 | 0) | **26**]	[(29 | 0 | 0 | 0) | **29**]	[(0 | 0 | 0 | 25) | **25**]
Scene Change	[(153 | 32 | 0 | 0) | **185**]	[(153 | 32 | 0 | 0) | **185**]	[(161 | 23 | 0 | 0) | **184**]
Twitch Chat	80.401%	59.784%	77.234%
Processing time (with all algorithms)	15h41m05s	18h36m31s	14h10m37s
Processing time (without chat interactions extraction algorithm)	02h17m14s	02h50m21s	01h55m12s
Stream duration	04h18m36s	04h35m55s	04h17m15s

Table 3.1: Test Results Summary

When the algorithm to extract the interactions of the *Twitch* chat is added to the previous system the total processing time increases massively. This algorithm presents a success rate of 72.473%, which is promising. This algorithm will need to be improved in efficiency to reduce the processing time. Further tests revealed that although the interactions were well separated, the text extraction components presented inaccuracies and sometimes could not discern the text from the background.

4 Conclusions

The gaming industry is rapidly advancing within the entertainment sector, offering significant opportunities for substantial innovation and seamless integration of cutting-edge technologies. In parallel the streaming industry is flourishing both in revenue and popularity. This growth brings possible problems that can be resolved with innovative technology, such as an automatic event detection system. This system can bring multiple benefits to the streamer who wants to create VODs and to the viewer who wants to watch a specific moment of a video that is too long to browse manually.

To navigate through the concepts presented in the dissertation, an overview of video games, their evolution, and the optimal platform for video game streaming was introduced in the section labeled Background Information. In this section, it is described the structure of the platform *Twitch* and the technology behind the delivery of the broadcast to every viewer.

The development of an event detection system involves examining current technologies used to detect events in sports videos, as it was done in Chapter State-of-the-Art. These technologies collect low-level information, group it, and combine it to create models called MLR. These models, which include visual, audio, or text, are processed with domain knowledge to produce high-level information called HLS.

In the same chapter, it was given an overlook of tools that are being currently used to detect events and highlights in video games. The section Event detection and Highlight detection in Video games describes an integrated tool in graphic engines that allows developers to trigger flags when a key moment of the game is played. This technology is another approach to the problem of detecting events in video game feeds.

In our investigation, we decided to solve the problem of automatically detecting and categorizing the events that occur during a broadcast of a videogame. Due to the variety of the videogames that have been created and the videogames that will be created, it was decided to restrict the target to only one videogame: LOL, a MOBA game developed and published by *Riot Games*. The most recent international tournament of this videogame was the *"2024 Mid-Season Invitational"* [40] that occurred between the dates of May 1st and 19th, streamed worldwide through the official channel of *Riot Games*.

To achieve this goal, we have developed specialized algorithms leveraging traditional image processing and vision systems. These advanced algorithms are tailored to analyze *Twitch* broadcasts, extracting a wealth of information including the duration of each event, match outcomes, team names, the exact start times of each round, the Champions Pick phase, game scores, and the presence of replays.

As analyzed in Section 3.6 of the previous chapter the system developed presented promising results regarding the identification of the events but the extraction of the Twitch chat presented a massive processing time, which turns its use not adequate for live streams extraction, revealing that it needs to be improved and/or to have a new approach to the problem. The following section expands on possible improvements and future work to be done.

4.1 Future work

The system of event detection for videogames presented in this dissertation has demonstrated significant potential for identifying key events during a game. However, there are several areas where future research and development could extend the system's capabilities, especially within the context of the game of LOL. This section outlines potential avenues for further work that would improve the system and broaden its applicability.

- **Addressing Existing Limitations**

  While the current system detects many in-game events, there are several key objectives in LOL that are not yet fully integrated into the event detection algorithms. For instance, the detection of objectives such as Baron Nashor, and Dragon kills events could be more finely tuned. Future work could focus on developing algorithms that capture these objective-related events with greater accuracy, offering insights into when and how these objectives are taken.

  Additionally, the system cannot currently identify the specific intervenients involved in these critical events. The inclusion of detailed identification of players who contribute to kills and turret destruction would provide more granular data on player performance and decision-making, enhancing the system's value for real-time analysis.

- **Expanding the Scope of Detected Events**

  One significant extension for future work lies in expanding the detection of in-game events to include commentator analysis. By identifying and mapping the periods when commentators focus on specific aspects of the game, such as team fights, strategic objectives, or individual player actions, the system could offer a richer narrative of the game flow. This would be especially useful for viewers and analysts, who could leverage such data to understand how game moments are being framed and emphasized by commentators.

This requires the development of algorithms capable of parsing commentary audio or text to extract the key moments being discussed, aligning them with in-game events. This integration could bridge the gap between technical event detection and the more human elements of game commentary.

- **Further Developing Twitch Chat Extraction**

  The current algorithm for extracting Twitch chat during a game could be further refined and extended. While the system can pull chat interactions during live streams, future developments should aim to analyze the sentiment and context of chat messages concerning in-game events. For example, detecting how viewers react to major plays or objectives being taken could provide a real-time gauge of audience engagement.

  Additional research could also investigate the possibility of filtering out irrelevant or spam messages, which often flood Twitch chats during live games. Developing advanced filtering techniques, possibly using machine learning or Natural Language Processing (NLP), could isolate meaningful viewer reactions and provide more accurate real-time insights.

- **Alternative Approaches and Interdisciplinary Research**

  To enhance the current system, future research could explore the use of deep learning and other AI-driven techniques to improve event detection accuracy and efficiency. For example, incorporating Convolutional Neural Network (CNN) for video frame analysis might lead to more precise detection of visual cues associated with in-game objectives or player actions.

  Moreover, interdisciplinary research with the field of behavioral sciences could offer novel insights into the interaction between players, commentators, and audience members. This would provide an opportunity to study not just the technical aspects of the game but also the psychological impact of game events on both players and viewers.

In conclusion, the current system of event detection for videogames offers a solid foundation, but there are many exciting opportunities for future work. By addressing the limitations in objective detection, expanding to cover commentator analysis, further developing Twitch chat extraction, and exploring new technologies and interdisciplinary approaches, the system could become an even more powerful tool for real-time analysis and engagement with live game content. These extensions represent valuable opportunities for researchers and developers to continue improving the capabilities of event detection in competitive gaming environments.

Bibliography

[1] J. M. Lourenço. *The NOVAthesis LaTeX Template User's Manual*. NOVA University Lisbon. 2021. URL: https://github.com/joaomlourenco/novathesis/raw/main/template.pdf (cit. on p. i).

[2] P. Zackariasson and T. L. Wilson. *The Video Game Industry: Formation, Present State, and Future*. First. Routledge, 2012. ISBN: 978-0-415-89652-8 (cit. on p. 1).

[3] J. Smith and J. Doe. "A data driven survey of video games". In: *2020 12th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. Bucharest, Romania: IEEE, 2020-06, pp. 1–6. ISBN: 978-1-72816-843-2. DOI: 10.1109/ECAI50035.2020.9223203 (cit. on p. 2).

[4] M. Iqbal. *Twitch Revenue and Usage Statistics (2023)*. Accessed: January 4, 2024. 2023. URL: https://www.businessofapps.com/data/twitch-statistics/ (cit. on p. 2).

[5] D. M. Ewalt. *How Big Is Twitch's Audience? Huge.* URL: https://www.forbes.com/sites/davidewalt/2014/01/16/twitch-streaming-video-audience-growth/. (Accessed: January 4, 2024) (cit. on p. 2).

[6] J. P. Alexei Scerbakov and F. Kappe. "When a Pandemic Enters the Game: The Initial and Prolonged Impact of the COVID-19 Pandemic on Live-Stream Broadcasters on Twitch". In: *Hawaii International Conference on System Sciences* (2022). DOI: 10.24251/HICSS.2022.391 (cit. on p. 2).

[7] A. Bossom and B. Dunning. *Video games: an introduction to the industry*. Fifth. Fairchild Books, An imprint of Bloomsbury Publishing PLC, 2016. ISBN: 978-1-4725-6711-6 (cit. on p. 3).

[8] *Rules of the Royal Game of Ur*. Accessed: January 4, 2024. URL: https://royalur.net/rules (cit. on p. 3).

[9] Oxford English Dictionary. *Videogame*. Accessed: January 4, 2024. 2023. URL: https://doi.org/10.1093/OED/5546735918 (cit. on p. 3).

[10] E. Charts. *All video game streaming sites & platforms | Esports Charts*. URL: https://escharts.com/platforms. (Accessed: January 10, 2024) (cit. on p. 3).

[11] L. SIUTY. *Twitch vs YouTube: Which Platform is Right for Your Game Stream?* URL: https://steelseries.com/blog/twitch-vs-youtube-which-platform-is-right-for-your-game-stream-1018. (Accessed: January 10, 2024) (cit. on p. 3).

[12] G. T. Jie Deng Felix Cuadrado and S. Uhlig. "Behind the game: Exploring the twitch streaming platform". In: *2015 International Workshop on Network and Systems Support for Games (NetGames)*. Zagreb, Croatia: IEEE, 2015-12, pp. 1–6. ISBN: 978-1-5090-0068-5. DOI: 10.1109/NetGames.2015.7382994 (cit. on p. 4).

[13] Twitch. *Twitch State of Engineering 2023*. URL: https://blog.twitch.tv/en/2023/09/28/twitch-state-of-engineering-2023/. (Accessed: January 10, 2024) (cit. on p. 5).

[14] W.-T. Chu and Y.-C. Chou. "On broadcasted game video analysis: event detection, highlight detection, and highlight forecast". In: *Springer Science Business Media* (2017-04). DOI: 10.1007/s11042-016-3577-x. URL: http://link.springer.com/10.1007/s11042-016-3577-x (cit. on pp. 6, 7, 11, 13).

[15] H. K. Stensland et al. "Bagadus: An integrated real-time system for soccer analytics". In: *ACM Trans. Multimedia Comput. Commun. Appl.* 10.1s (2014-01). ISSN: 1551-6857. DOI: 10.1145/2541011. URL: https://doi.org/10.1145/2541011 (cit. on pp. 6, 7).

[16] G. T. Thomas B. Moeslund and A. Hilton. *Computer Vision in Sports*. Springer International Publishing, 2014. ISBN: 978-3-319-09395-6 978-3-319-09396-3. URL: https://link.springer.com/10.1007/978-3-319-09396-3 (cit. on p. 6).

[17] J. E. González and F. T. Ruiz. "Automatic event detection for tennis broadcasting". In: (2011-07). URL: http://hdl.handle.net/2099.1/12564 (cit. on pp. 6, 8).

[18] A. Sethi. "(PDF) INTERACTION BETWEEN MODULES IN LEARNING SYSTEMS FOR VISION APPLICATIONS". In: (2014) (cit. on p. 7).

[19] A. A. Halin. "SOCCER VIDEO EVENT DETECTION VIA COLLABORATIVE TEXTUAL, AURAL AND VISUAL ANALYSIS". In: (2011). URL: https://www.academia.edu/15757026/SOCCER_VIDEO_EVENT_DETECTION_VIA_COLLABORATIVE_TEXTUAL_AURAL_AND_VISUAL_ANALYSIS (cit. on pp. 7, 11).

[20] I. G. Fabio Sulser and H. Schuldt. "Crowd-based Semantic Event Detection and Video Annotation for Sports Videos". In: *MM '14: 2014 ACM Multimedia Conference* (2014-11). DOI: 10.1145/2660114.2660119. URL: https://dl.acm.org/doi/10.1145/2660114.2660119 (cit. on p. 7).

[21] M. K. Mostafa Tavassolipour and S. Kasaei. "Event Detection and Summarization in Soccer Videos Using Bayesian Network and Copula". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2014-02). DOI: 10.1109/TCSVT.2013.2243640. URL: http://ieeexplore.ieee.org/document/6422365/ (cit. on p. 9).

[22] N. Nguyen and A. Yoshitaka. “Soccer video summarization based on cinematography and motion analysis”. In: *2014 IEEE 16th International Workshop on Multimedia Signal Processing (MMSP)* (2014-09). DOI: 10.1109/MMSP.2014.6958804. URL: http://ieeexplore.ieee.org/document/6958804/ (cit. on p. 9).

[23] C.-Y. Chiu et al. “Tagging Webcast Text in Baseball Videos by Video Segmentation and Text Alignment”. In: *IEEE Transactions on Circuits and Systems for Video Technology* 22.7 (2012), pp. 999–1013. DOI: 10.1109/TCSVT.2012.2189478 (cit. on pp. 9, 11).

[24] J.-L. W. Min-Chun Hu Ming-Hsiu Chang and L. Chi. “Robust Camera Calibration and Player Tracking in Broadcast Basketball Video”. In: *IEEE Trans. Multimedia* (2011). DOI: 10.1109/TMM.2010.2100373. URL: http://ieeexplore.ieee.org/document/5671490/ (cit. on p. 9).

[25] C.-M. Chen and L.-H. Chen. “Novel framework for sports video analysis: A basketball case study”. In: *2014 IEEE International Conference on Image Processing (ICIP)* (2014). DOI: 10.1109/ICIP.2014.7025193. URL: http://ieeexplore.ieee.org/document/7025193/ (cit. on p. 9).

[26] U. S. Surya Nepal and G. Reynolds. “Automatic Detection of ‘Goal’ Segments in Basketball Videos”. In: *MULTIMEDIA ’01: Proceedings of the ninth ACM international conference on Multimedia* (2001-10). DOI: https://doi.org/10.1145/500141.500181. URL: https://dl.acm.org/doi/abs/10.1145/500141.500181 (cit. on p. 9).

[27] A. G. Yong Rui and A. Acero. “Automatically extracting highlights for TV Baseball programs”. In: *MM00: ACM Multimedia 2000* (2000-10). DOI: 10.1145/354384.354443. URL: https://dl.acm.org/doi/10.1145/354384.354443 (cit. on p. 9).

[28] M. Xu et al. “Creating audio keywords for event detection in soccer video”. In: *2003 International Conference on Multimedia and Expo. ICME ’03. Proceedings (Cat. No.03TH8698)*. Vol. 2. 2003, pp. II–281. DOI: 10.1109/ICME.2003.1221608 (cit. on p. 9).

[29] L.-T. C. Min Xu Ling-Yu Duan and C. sheng Xu. “Audio keyword generation for sports video analysis”. In: *MM04: 2004 12th Annual ACM International Conference on Multimedia* (2004-10). DOI: 10.1145/1027527.1027702. URL: https://dl.acm.org/doi/10.1145/1027527.1027702 (cit. on pp. 9–11).

[30] H. Xu and T.-S. Chua. “Fusion of AV Features and External Information Sources for Event Detection in Team Sports Video”. In: *ACM Transactions on Multimedia Computing, Communications, and Applications* 2 (2006-02), pp. 44–67. DOI: https://doi.org/10.1145/1126004.1126007. URL: https://dl.acm.org/doi/abs/10.1145/1126004.1126007 (cit. on p. 11).

[31] P. M. P. Petros Xanthopoulos and T. B. Trafalis. “Linear Discriminant Analysis”. In: *Robust Data Mining*. New York, NY, USA: Springer New York, 2013. ISBN: 978-1-4419-9877-4 978-1-4419-9878-1. DOI: http://doi.acm.org/10.1145/964001.964023 (cit. on p. 12).

[32] K. C.-C. C. Yi Wu Edward Y. Chang and J. R. Smith. “Optimal multimodal fusion for multimedia data analysis”. In: New York, NY, USA: ACM, 2004, pp. 572–579. ISBN: 978-1-58113-893-1. DOI: 10.1145/1027527.1027665 (cit. on p. 12).

[33] M. R. Alfian Abdul Halin and D. Ramachandram. “Overlaid Text Recognition for Matching Soccer-Concept Keywords”. In: Penang, Malaysia: IEEE, 2008, pp. 235–241. ISBN: 978-1-58113-893-1. DOI: 10.1109/CGIV.2008.34 (cit. on pp. 12, 13).

[34] Q. Memon. *Sobel operator in digital image processing*. URL: https://www.educative.io/answers/sobel-operator-in-digital-image-processing. (Accessed: January 4, 2024) (cit. on p. 12).

[35] OpenCV. *OpenCV: Morphological Transformations*. URL: https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html. (Accessed: January 4, 2024) (cit. on p. 12).

[36] A. S. “An overview of Tesseract OCR Engine”. In: (2016) (cit. on pp. 12, 23).

[37] N. Otsu. “A Threshold Selection Method from Gray-Level Histograms”. In: *IEEE Transactions on Systems, Man, and Cybernetics* 9.1 (1979), pp. 62 –66. DOI: 10.1109/TSMC.1979.4310076 (cit. on pp. 13, 22).

[38] NVIDIA. *GeForce Experience*. URL: https://www.nvidia.com/en-us/geforce/geforce-experience/. (Accessed: January 10, 2024) (cit. on p. 13).

[39] R. Games. *League of Legends - how to play*. URL: https://www.leagueoflegends.com/en-us/how-to-play/. (Accessed: July 28, 2024) (cit. on pp. 15, 24).

[40] R. Games. *MSI - 2024 Mid-Season Invitational*. URL: https://lolesports.com/en-US/news/msi-2024-primer. (Accessed: July 28, 2024) (cit. on pp. 15, 24, 34, 37).

[41] E. CHARTS. *Popular esports games in 2024 by viewership*. URL: https://escharts.com/top-games?order=peak. (Accessed: July 29, 2024) (cit. on p. 15).

[42] TWITCH. *Chat Basics*. URL: https://help.twitch.tv/s/article/chat-basics?language=en_US. (Accessed: March 30, 2024) (cit. on p. 17).

[43] Twitch. *Twitch*. URL: https://www.twitch.tv/. (Accessed: April 17, 2024) (cit. on p. 18).

[44] OpenCV. *OpenCV: Operations on arrays*. URL: https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html. (Accessed: March 30, 2024) (cit. on pp. 18, 24).

[45] OpenCV. *OpenCV: Color Space Conversions*. URL: https://docs.opencv.org/4.x/d8/d01/group__imgproc__color__conversions.html#gaf86c09fe702ed037c03c2bc603ceab14. (Accessed: December 2, 2024) (cit. on p. 19).

[46] OpenCV. *OpenCV: Geometric Image Transformations*. URL: https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html#ga47a974309e9102f5f08231edc7e7529d. (Accessed: December 2, 2024) (cit. on p. 20).

[47] BIGVISION. *Otsu's Thresholding Technique | LearnOpenCV*. URL: https://learnopencv.com/otsu-thresholding-with-opencv/. (Accessed: August 28, 2024) (cit. on p. 22).

[48] OpenCV. *Image Thresholding — OpenCV 3.0.0-dev documentation*. URL: https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html. (Accessed: August 28, 2024) (cit. on p. 22).

[49] OpenCV. *Template Matching*. URL: https://docs.opencv.org/4.x/de/da9/tutorial_template_matching.html. (Accessed: August 28, 2024) (cit. on p. 28).

[50] OpenCV. *OpenCV: High Level API: TextDetectionModel and TextRecognitionModel*. URL: https://docs.opencv.org/4.x/d4/d43/tutorial_dnn_text_spotting.html. (Accessed: August 28, 2024) (cit. on pp. 29–31).

2024 Automatic event detection for Videogames: This thesis will focus on the development of a solution for automatic events detection for videogames streams Gustavo Murta

NOVA
NOVA SCHOOL OF
SCIENCE & TECHNOLOGY