NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

DUARTE PEREIRA FERNANDES

BSc in Electrical and Computer Engineering

AUTOMATIC EVENT DETECTION FOR MOTORSPORTS

INTEGRATING COMPUTER VISION TECHNIQUES WITH RACING EVENTS

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
September, 2024

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

AUTOMATIC EVENT DETECTION FOR MOTORSPORTS

INTEGRATING COMPUTER VISION TECHNIQUES WITH RACING EVENTS

DUARTE PEREIRA FERNANDES

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

Examination Committee

Chair: João Francisco Martinho Lêdo Guerreiro
Assistant Professor, NOVA University Lisbon

Rapporteur: José Manuel Matos Ribeiro da Fonseca
Associate Professor with Aggregation, NOVA University Lisbon

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

João Francisco Martinho Lêdo Guerreiro
Assistant Professor, NOVA University Lisbon

José Manuel Matos Ribeiro da Fonseca
Associate Professor with Aggregation, NOVA University Lisbon

MASTER IN ELECTRICAL AND COMPUTER ENGINEERING

NOVA University Lisbon
September, 2024

Automatic Event Detection For Motorsports
Integrating Computer Vision Techniques with Racing Events

This document was created with the (pdf/Xe/Lua)LaTeX processor and the NOVAthesis template (v7.2.5) [1].

To my younger self, who had the courage to embark on this challenging journey.

Acknowledgements

I would like to thank everyone who has supported and motivated me throughout the making of this paper. Without them, this thesis would have not been possible.

Firstly, I would like to thank my adviser, Daniel Silvestre, who has guided me through every step of the way here. His knowledge, guidance and patience were instrumental in shaping this thesis and consequently has made it quite pleasurable.

I also want to show my appreciation to Six Floor Solutions for their continuous support, especially by Xavier Frazão, who always took some time from his day to mentor me and discuss ideas for certain approaches. Their availability, support and amount of resources provided were invaluable to the making of this thesis.

To my friends, Vasco Raposeiro, Manuel Mantas, Filipe Oliveira, Miguel Paiva, João Amorim and António Pegado I offer my heartfelt thanks. Your friendship has helped me to make many fruitful decisions and stay motivated.

Last but not least, my family has supported me, as always, and guided me throughout my whole life and have made me into the person I am today. For that I am eternally grateful.

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.

Abstract

The current surge in popularity of specific motorsports, particularly Formula 1, has led to an increasing number of individuals seeing races via streaming platforms. For anyone unable to attend the event physically, these services offer comprehensive coverage of the races. Streaming providers should provide a shortened version of the race that encompasses all significant moments for viewers with limited time. This method enables spectators to remain apprised of the critical advancements in the race.

The primary focus of this thesis will be the development of an application that effectively delivers a condensed version of a race. This document examines previously created and validated solutions for various sports and applies these concepts to an application designed to automatically recognize events in motorsports. The developed solution will employ techniques including optical character recognition and image similarity algorithms.

Resumo

Devido ao recente aumento na popularidade de alguns desportos motorizados, como a Fórmula 1, cada vez mais pessoas estão a assistir corridas através de serviços de streaming. Para aqueles que não podem assistir ao evento pessoalmente, estes fornecem uma cobertura total das corridas. Para ajudar os espectadores que não têm tempo para assistir à corrida na sua totalidade, os forncedores de streaming devem produzir uma versão condensada da corrida que inclua todos os eventos importantes. Estes espectadores seriam capazes de permanecer informados sobre os desenvolvimentos importantes da corrida graças a esta estratégia.

Este será o principal tema a desenvolver, uma aplicação capaz de fornecer uma versão simplificada de uma corrida eficientemente. Este documento aborda soluções previamente desenvolvidas e testadas para outros desportos e aplica esses fundamentos e princípios a uma aplicação capaz de detetar eventos automaticamente para desportos motorizados. A solução desenvolvido irá utilizar técnicas de reconhecimento ótico de caracteres e de semelhança de imagens.

Contents

List of Figures **xiii**

Acronyms **xvii**

1 Introduction **3**
1.1 Motivation 4

2 State-of-the-art **5**
2.1 Different levels of features 5
2.2 Audio features 6
2.3 Video features 7
2.3.1 Line detection 7
2.3.2 Scene transition detection 9
2.3.3 Car detection 10
2.4 Textual features 11
2.4.1 Text extraction from external sources 12
2.4.2 Text extraction from video stream 12
2.4.3 Textual features for motorsports 13
2.5 Event detection 14

3 Motorsports Event Detection **15**
3.1 Image Similarity 16
3.1.1 MSE, PSNR and Histogram Comparison 17
3.1.2 Structural Similarity Index Measure 18
3.1.3 Template Matching 18
3.2 Overtake Detection and Starting Grid 19
3.3 Element Detection on Leaderboard 21
3.4 Replay Identification 22
3.5 Detecting Scene Changes 24
3.6 Configuration File 25

3.7 Lap Extraction 26
3.8 Final results and Future Improvements 27

4 Conclusions **31**

Bibliography **33**

List of Figures

2.1 Hierarchical representation of features 6
2.2 Hough transform example, [16] 9
2.3 Architecture for YOLOv1 [20] 11
2.4 Proposed architecture to retrieve text from external sources, adapted from [23] 12
2.5 Pseudo-code for texture-based text recognition [25] 13

3.1 Template Matching 16
3.2 Template Matching 19
3.3 Original leaderboard 20
3.4 Processed segment 20
3.5 Leaderboard processing 20
3.6 Processing retired drivers 20
3.7 Dynamic element 22
3.8 Static element 22
3.9 Examples of elements present on the leaderboard 22
3.10 Replay identification and detection sections 23
3.11 Dynamic element 23
3.12 Static element 23
3.13 Additional element 23
3.14 Find text algorithm, firstly applies the mask and then finds relevant area . . 23
3.15 Scene change algorithm 25
3.16 Sample of configuration file 26
3.17 Subtraction of previous lap and current lap 26

Listings

Acronyms

CNN Convolutional Neural Networks *(pp. 9–11)*

FFT Fast Fourier Transform *(p. 6)*
FSM Finite State Machine *(p. 14)*

HMM Hiden Markov Model *(pp. 6, 10)*

JSON JavaScript Object Notation *(pp. 19, 25)*

MFCC Mel-Frequency Cepstral Coefficient *(p. 6)*
MSE Mean Square Error *(p. 17)*

OCR Optical Character Recognition *(pp. 13, 19, 20, 23, 26)*

PSNR Peak signal-to-noise ratio *(pp. 17, 21, 22, 24, 25)*

SSIM Structural similarity index measure *(pp. 18, 24)*

TTS Text-To-Speech *(p. 6)*

.

1

Introduction

Ever since the creation of the first automobile in the 19th century, humankind has been wheel to wheel racing relying on the very essence of human innovation and the spirit of challenge. The first race cars were very rudimentary, with no more than an inefficient power train, 4 wheels and a resemblance of brakes. Nowadays, motorsports requires vehicles to be extremely advanced and precise in order to be competitive, each part has had countless hours of engineering work behind it so that it can perform optimally even during the harshest of conditions. This was the result of budget increments and tighter design requirements over the years. With the teams having their financial support increased, the standard and caliber of racing have improved. Fan interest in motorsports has increased as a result of this advancement.

The increase in popularity of motorsports came with associated social and economic impacts. These events boost tourism activity, albeit temporarily, in the hosted city, which can generate a substantial amount of revenue [2] [3]. Besides increasing cash flow, there are several benefits revolving around motorsports, such as an improvement in infrastructure, the creation of new jobs, and the encouragement of new hobbies and recreational activities for the residents [2]. While the advantages of hosting motorsport events are significant, they are not without their challenges. These events often require a partial redesign of the host city, leading to substantial initial investments. Additionally, the dependency of racing events on favorable weather conditions poses a risk; adverse weather can lead to cancellations, potentially leaving the host city with substantial financial burdens. Given that the race is not canceled an influx of tourists can be expected to increase noise pollution and littering [2] [3], which may cause discontent among residents, particularly those not engaged in the tourism industry. Studies also show that older residents, or residents who are not educated also tend to have a negative outcome about these types of events [3].

1.1 Motivation

Motorsports is widely recognized as a global phenomenon, with its popularity and economic significance continuously growing, [4]. According to the last chapter Formula 1's industry has demonstrating the immense scale and reach of the motorsport economy. With the increasing accessibility of live streaming and digital platforms, fan engagement continues to grow, and this trend is expected to continue across various racing leagues worldwide. The use of an automatic event detection system in motorsports can revolutionize the way races are broadcasted and experienced, [5]. By streamlining the process of highlight generation, such systems can deliver instant replays, enhance live commentary, and provide in-depth analysis of race-defining moments such as overtakes, crashes, and pit stops. This technology not only elevates the viewing experience but also offers valuable tools for drivers and teams to analyze performance and strategies in real time [6]. Moreover, the development of such systems aligns with the increasing demand for personalized and engaging content, enabling broadcasters to cater to diverse audience preferences. The insights derived from these algorithms can also inspire advancements in other sports and fields where real-time analysis of complex events is essential. By leveraging cutting-edge technology, motorsports can continue to evolve as both a thrilling sport and a leading example of innovation-driven entertainment.

2

State-of-the-art

This section will examine the current state of computer vision systems and how they are used in racing for autonomous event detection. It will discuss a number of contemporary methods, emphasizing both their benefits and drawbacks. A preliminary investigation has shown that there are few systems expressly built for motorsports. Nevertheless, this research will consider technologies used in event detection for other sports like football, basketball, and tennis, drawing analogies and insights applicable to racing. This chapter seeks to provide a thorough understanding of how current computer vision technologies might be used and potentially improved.

To achieve this, a brief explanation of the different levels of features extracted must be given. By combining multiple low-level features, we can successfully extract higher-level ones, such as racing strategies or even tire-wearing patterns. We follow this with the different types of elements that can be extracted: Audio, Video, and Textual; Having all these characteristics extracted, a multitude of events can be extrapolated from a video stream. The extraction of only one of these types of features would prove to be detrimental [7], decreasing significantly the accuracy of event detection, therefore a robust computer vision system used for event detection must be able to detect a multitude of different features.

2.1 Different levels of features

A good computer vision system must be able to detect higher level characteristics like racing strategies and pit stop prediction, through the careful analysis of an entire race. To do so, the system needs to absorb all the small details on the live stream. These are called low-level features; they are pretty common and serve as the basis for higher-level features. To provide context, to accurately predict a pit stop, the system should be informed about the number of laps a pilot has on a set of tires as well as its compound. This paper [8] talks about a division in 3 levels, the lower ones are a lot more common than the higher ones, just like previously discussed.

Low-level features in live data can manifest in various forms: visually (such as track

limits), audibly (like changes in the pitch of commentators' voices), or textually (for instance, through analysis of scoreboard changes). To illustrate, in computer vision systems designed for motorsports, a change in the scoreboard is considered a textual lower-level feature, while a change on voice activity represents an auditory feature. This hierarchical approach is crucial in automatic event detection, where understanding both basic and sophisticated aspects of the features leads to more accurate and meaningful detection, especially in dynamic fields like motorsports.

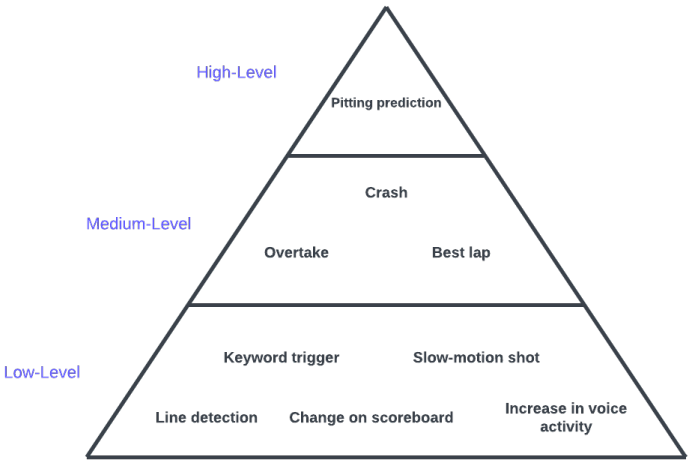


Figure 2.1: Hierarchical representation of features

2.2 Audio features

As previously stated, recognizing several different features is detrimental to accurately detecting events on a computer vision system. Audio features are no exception, albeit some approaches may prove inflexible in a motorsports setting.

The first approach involves giving a system the ability to interpret human words through the use of a speech recognition algorithm. As [9] noted, there are powerful and free Text-To-Speech (TTS) tools that are readily configured, which assist the system in understanding what is being said on stream. A computer vision system can infer low-level features from many sorts of conversations by combining a known collection of keywords and a decent TTS algorithm. The problem with this technique is that the key set is not dynamic, which makes it impossible to adapt a vocabulary suitable for every commentator on the world. Another significant issue is that some feeds lack analysts entirely.

The latter, a more mathematical method, examines voice and audio activity on a video stream. By detecting frequent sound patterns for a specific sport, the computer vision system now knows that if a sound clip is similar to the ones previously detected, a low level feature can be detected. Similar solutions have been presented, [10][11] discovered that Mel-Frequency Cepstral Coefficient (MFCC) are particularly good at detecting audio patterns using Fast Fourier Transform (FFT) (2.1) and Mel-Frequency (2.2), primarily because it can still work when there is considerable noise on the audio clip. Hiden Markov

Model (HMM) is a sturdy way to model all the different audio patterns [11].

$$X[k] = \sum_{n=0}^{N-1} x[n] \cdot e^{-\frac{i2\pi}{N} kn} \tag{2.1}$$

$$\text{Mel}(f) = 2595 \log_{10} \left(1 + \frac{f}{700}\right) \tag{2.2}$$

With n being the index of the discrete time sample, f the frequency, x[n] the input signal in the time domain, N the total number of samples and $e^{-\frac{i2\pi}{N} kn}$ is the core of the Fourier transform by projecting the input signal, x[n], onto a basis of complex sinusoids.

These forms of feature extraction can help a computer vision identify when there is a commercial break or, for example, when there is a scene change, which is usually accompanied by a distinctive tune. Furthermore, if a large enough crowd is present at a racing event, a large applause can create a low-level feature that can begin the detection of an overtake, for example. Alternatively, if a simple speech-to-text algorithm is available, the term "pit", or "pitlane", can be added to the vocabulary and used to track the number of pit stops made by a designated driver throughout a race. Even though audio feature extraction has proven to be successful, video and textual feature extraction must still be present when detecting an event to confirm its authenticity.

2.3 Video features

The identification of lines or other geometric shapes, such as squares or circles, is one of the most often utilized detections on computer systems used for automatic event detection. This is useful in sports like tennis and football, where the result of a play is decided by whether the ball has crossed the field's boundaries. The field is divided into multiple sections, each of which can accommodate a variety of different plays. This suggests that shape detection is crucial to describing the various plays. Characterizing different types of camera shots and their length, as well as identifying cars, are useful aspects for this application. They both aid a computer vision system in recognizing scene transitions.

2.3.1 Line detection

Before an image can go through the process of line detection it first needs some pre processing. First and foremost, smoothing the image could be necessary before moving on to edge detection [12] [13], if noise is present. There are numerous image smoothing algorithms available today that eliminate unnecessary noise; some are slower but still filter a large amount of noise, while others take a while to process but filter nearly all of it [14].

The previously cited paper finds that the mean filter is a quick and easy way to remove noise, however it is not very good at doing so. On the other hand, a Gaussian filter smooths an image by averaging pixel values based on a Gaussian kernel, substantially decreasing noise while slightly softening edges. A median filter, on the other hand, replaces each pixel with the median value of its neighborhood, making it excellent at reducing impulsive noise while retaining edges. Gaussian filters are quicker and better for general noise, whereas median filters are slower but more effective for impulse noise and edge retention. This in turn, makes the Gaussian filter a perfect balance of edge retention and performance.

Once superfluous noise has been eliminated from the picture, an edge detection technique can be utilized. Sobel Operator and Canny Edge Detection are the two contenders. Canny's Edge Detection process starts by using an already smoothed image. It then determines the direction and severity of color or brightness changes, which aids with edge identification. The next step involves highlighting the edges that are the most noticeable and classifying them as strong or weak according to how clear they are. In order to ensure that the most significant outlines are caught and the lesser ones are ignored, it finally concentrates on the strong edges [13]. Lastly, the Sobel operator consists of convolving, (2.5), two matrices, (2.3), or *kernels*, with the original image to find the gradient approximations of the image intensity, (2.4). The edges represent the places where the intensity of the image has the highest rate of variation from light to dark.

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad \text{and} \quad G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \tag{2.3}$$

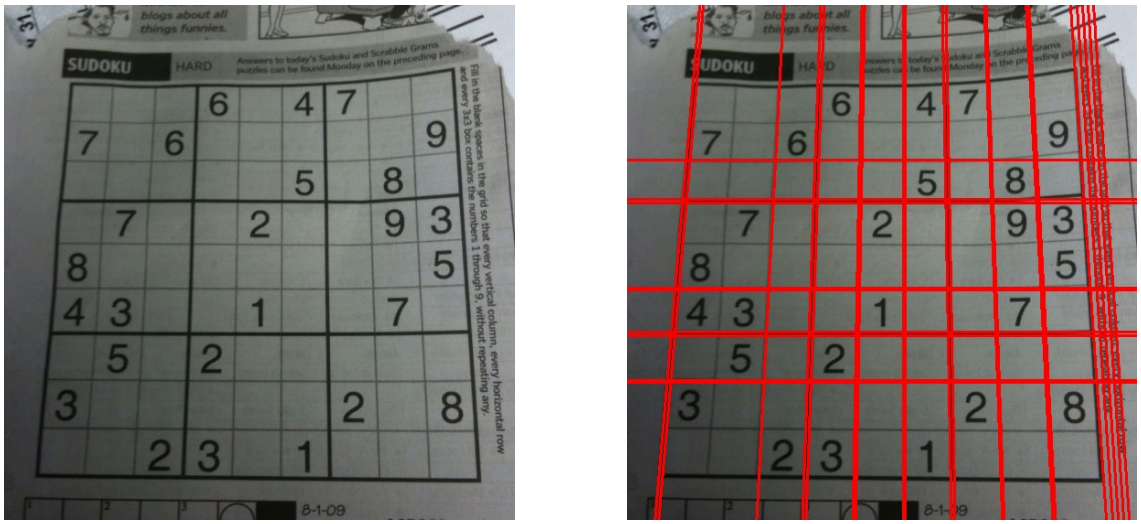
$$Sobel(x) = G_x * G_y * Image \tag{2.4}$$

Convolution is a mathematical procedure that shows how the shape of one function is changed by another by combining two functions to create a third. Mathematically speaking, the integral of the sum of the two functions after one is shifted and reversed is the convolution of two functions, f and g, represented by the symbol (f * g):

$$(f * g)(k) = \sum_{j=0}^{k} f(j) \cdot g(k - j) \tag{2.5}$$

By employing a filter or a kernel, this procedure can produce a modified version of an input signal, such as an image. Convolution is a tool that can be used for a variety of tasks in image processing, such as blurring, sharpening, and edge detection. Convolutional Neural Networks (CNN) used for deep learning purposes use convolution as the main working principle, aswell as some filters mentioned above.

Studies have shown that Canny's Edge Detection algorithm has superior accuracy and lower execution time [13]. With edges now clearly defined, a line identification technique should to be used. The Hough transform is a good fit for this since it is highly effective at detecting lines even in noisy or incomplete data [15]. This algorithm consists on transferring feature points from the image space, (x,y), into the parameter space, (ρ, θ), where a voting method is applied. In line detection, for example, every point in the image space casts a vote for every line in the parameter space that is possible to pass through it. The accumulator is an array that holds the total of these votes. A shape is present in the image space when the accumulator has high values.



(a) Original image

(b) Hough transform on image

Figure 2.2: Hough transform example, [16]

Finding lines is important, but for the present application is not. Line detection would prove to be useful for track limits detection, since cars are not constantly monitored, a computer vision system of this caliber would not accurately infer track limits violations using visual features.

2.3.2 Scene transition detection

Multiple ad breaks are frequently incorporated into live streaming, especially for sporting events like motorsports. These pauses, which are usually planned during intermissions or less crucial times, pose a unique problem for autonomous event detection systems. During these commercial intervals, there are no significant events related to the live stream's main

content to locate or investigate. This means that in order to maintain the focus on capturing and processing important moments of the real event, these systems must be able to discern between the main action and the commercial breaks.

To detect scene changes, a comparison of color histograms could be carried out. A visual feature could be identified by comparing the color histogram of the current frame to the color histogram of a known scene transition, thanks to computer vision's real-time processing capabilities. This is limited to sports with constant camera positions, though, as the histogram will diverge if the background of the new frame differs from that of the reference frame [6]. In racing, color histogram comparison is not a good strategy because the cameras are typically mobile, either by swivel or even helicopter in some higher level leagues.

Another intriguing method for identifying scene changes is to categorize camera shots [17]. Because this method was developed for football, examples, and methodologies for football are provided. Shots come in three varieties: long, medium, and short. While medium-length films concentrate on a certain area of the field, long-length films capture the entire field. Conversely, a close-up shot called "short" just reveals above a player's waist. According to the aforementioned paper, close-up pictures are taken during a highlight, whereas long shots are linked to game events. On the other hand, broadcasters frequently employ lengthy shots during commercial breaks. This research therefore considers shot length in addition to shot type to locate game events. Research also shows that highlights can also be located by identifying slowed-down video [17]. The use of slow motion ensures better comprehension to the viewers.

A technique created to detect slow-motion replays is presented in this paper [18]. This method uses an inference methodology to calculate the probabilities of slow-motion segment boundaries and analyzes the structure of the segments using a HMM. It combines characteristics based on the magnitude of time-varying fluctuations in video field differences and zero-crossings, which detect motion changes. This method's ability to discriminate between slow-motion replays and comparable content in ads is one of its main advantages, as it increases its application to a wider range of sports and video genres.

2.3.3 Car detection

In the field of computer vision, object detection is a crucial procedure that enables the recognition and positioning of objects in frames. Algorithms for object detection can identify unique objects and tell them apart from other objects or the backdrop by examining edges, textures, shapes, or color patterns of a certain image. Recently, with the development of CNN, the performance of object detection models have improved [19] [20] [21]. Modern models are of two types: Single-Stage Two-Stage.

Two-stage models, such as Faster R-CNN, produce region proposals (possible locations for objects) first, then categorize each proposal and improve its bounding box. They process information more slowly, but they are renowned for their great accuracy, particularly with

small things. One-stage models are faster but typically less accurate because they complete object detection and localization in a single pass over the image [20]. Examples of these models are YOLO (You Only Look Once) and SSD (Single Shot Multibox Detector). For this application, where real-time processing is needed, a single-stage model would be the appropriate choice due to its faster execution time.

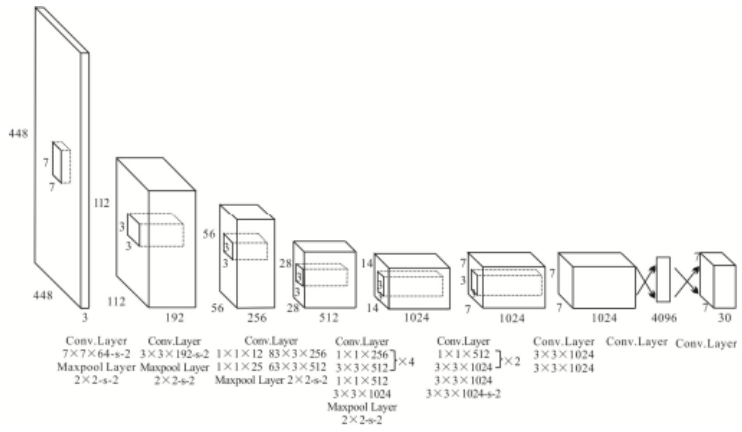


Figure 2.3: Architecture for YOLOv1 [20]

Single-stage models were shown to be the most effective method for this specific application. Previous research has demonstrated that YOLO offers the optimum trade-off between accuracy and inference time. That being said, this pre-trained model was designed to identify dozens, if not hundreds, of distinct objects. As a result, the CNN has more layers, which lengthens the inference time. The number of convolution layers decreases with the use of more specialized models, [22], which shortens the time required for inferences. Since the system will operate in real-time and only needs to detect automobiles, this might be an appropriate method for autonomous event detection for motorsports.

Multiple visual elements can be developed with the precise detection and identification of cars during a race event. For instance, it is possible to construct a visual feature that, when paired with an aural feature, can accurately identify an overtaking by identifying two cars driving side by side. Cars are lined up at the start grid, if the system can accurately identify every car, it will be able to determine the exact moment when the race has begun.

2.4 Textual features

Finding text features is an important application of computer vision systems for motorsports. Annotations, scoreboard statistics, and scene text are examples of textual data that can provide important context for understanding what's happening throughout a race. Through efficient identification and analysis of this material, systems are able to furnish real-time updates, identify significant occurrences, and even furnish contextual comprehension of the race. This is especially helpful for automated systems that seek to summarize or highlight key events during live sports broadcasts, when quick and precise

information extraction is critical to improving viewer experience.

2.4.1 Text extraction from external sources

Two primary text sources are used in the majority of applications created using computer vision systems to automatically recognize events: scene text overlaid on the video stream and text from external sources such as webcasts and closed captions. The application in this study, [23], retrieves text from both of these sources. The suggested technique for extracting closed caption text, or web cast text, involves regularly storing a .html file and looking for predetermined keywords. The system then searches the file for modifications to the transcription, and adds it to the database where it can be processed.

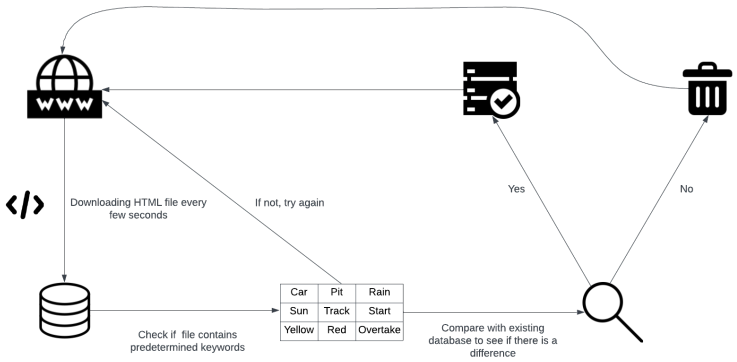


Figure 2.4: Proposed architecture to retrieve text from external sources, adapted from [23]

2.4.2 Text extraction from video stream

Different approaches are recommended for text found in the video stream; some can detect text on natural images, others can detect text on backgrounds with varying degrees of complexity, and some can detect text exclusively in the horizontal orientation [24]. Textual feature extraction is separated into two phases: detection and extraction. During the detection phase, regions of the image containing textual content are firstly identified. This process is followed by a verification of their status as coherent words by the use of a model. Therefore, this operation has similarities to the one mentioned on Chapter **2.3.3**.

Texture-based text recognition approaches treat text as a distinct type of texture, using properties such as local intensities, Figure **2.5**, and filter responses to identify text from non-text areas. These approaches are computationally demanding because they necessitate scanning across all locations and scales, and they primarily focus on horizontal text, with limits in dealing with rotation and scale variations.

Component-based algorithms, on the other hand, begin by extracting probable text components using techniques such as color clustering. These elements are then filtered using either handwritten rules or machine learning classifiers. This method is more efficient since the number of components processed is reduced, and it is more adaptable to

```
detectTextRegions(Image edgeImg){
 comment: edgeImg is previously created
 comment: textRegion is a data structure with 4 fields: x0, x1, y0, y1
 Integer[] H = calculateLineHistogram(edgeImg)
 textRegions[] TC = determineYCoordinate(H)
 TC = determineXCoordinate(edgeImg, TC)
 return(TC)
}
```

Figure 2.5: Pseudo-code for texture-based text recognition [25]

changes in rotation, scale, and font variation [24] [25]. The first approach is more suited to natural images where text can be inserted on a complex image with different backgrounds, colors and shapes.

Now that the text-containing region has been effectively discovered, the computer vision system must learn how to recognize the text. Optical Character Recognition (OCR) is one of the most powerful techniques available. The computerized conversion of text or the development of a digital copy of text from sources such as handwritten documents, printed text, or natural pictures is referred to as OCR. Because the efficacy of this approach is heavily reliant on image pre-processing, namely edge enhancement and noise suppression, [26],an image should be passed through a Gaussian filter and a Canny Edge Detection Operator before utilizing OCR, as described in chapter **2.3.1**. There are currently various OCR frameworks, however only a few of them are free. Tesseract is a strong contender in this category [27].

2.4.3 Textual features for motorsports

Open captions are often included and integrated directly into the video stream, but closed captions are rarely found in contemporary video streaming services. Although closed captions are less common on these platforms, they are more flexible because you can turn them on and off. Conversely, open captions are always present in the video file and are always visible to viewers. An additional limitation on the accessibility of real-time textual information is the absence of web-cast text services, such as live-chat sections, on these platforms. Since closed-caption text is becoming a rarity nowadays and most streaming services do not offer web-cast text it may not be necessary for this application to conduct textual feature extraction from external sources.

Much like audio feature recognition, certain terms that show up on the screen, like "Best lap!" can cause additional features to be created. Through the correlation of these prompts' timestamp with other data, such as lap times, the system is able to produce insightful highlights. Furthermore, for context-aware event detection, tracking other textual elements such as timers or lap counters is essential. For instance, alterations to the scoreboard could suggest an overtake. Because scanning features like scoreboards

require a lot of resources, they could be conditionally enabled by other signals like a rise in speech activity or visual hints like cars racing side by side. The examination of motorsport occurrences is made more precise and efficient by this integrated method.

2.5 Event detection

Event Detection, a crucial component of this study, combines the audio, video, and textual elements that were identified and covered in earlier parts to detect motorsport events. This section explores how to use these separate elements together to build an efficient automatic event detection system. During the research done on this chapter, possible chains of features that lead to an event were mentioned.

During a race, switching between long and mid, or close-up, shots is an essential way to visually identify significant moments. These changes frequently signal the start of a pause in the action or a change in the racing session. Recognizing this trend is essential to producing a succinct and powerful highlight reel since it facilitates the removal of scenes that aren't as interesting, including commercial breaks or session transitions. Furthermore, slow-motion video, which is frequently used to highlight noteworthy moments, works well for event recognition. When the first lap appears on the scoreboard and the vehicles are lined up in a line, the race has officially begun.

A race event will almost entirely be recorded using medium or close-up shots, this is where most of low-levels features are going to be extracted. While many of the low-level elements that cause each event are shared, each one is set off by a distinct sequence:

Event	Voice activity	Cars side by side	Change in scoreboard	Others
Overtake	Increase	Yes	Yes	-
Crash	Increase	No	Yes	Smoke
Pit stop	No change	No	Yes	Outside of track limits
Best lap	Maybe	No	No	Screen prompt:"Best Lap!"

Table 2.1: Low-level features needed to trigger an event

This research, [23], addresses the application of a Finite State Machine (FSM) for event detection based on the low-level feature sequence in the scenario that is suggested. However, in circumstances when feature recurrence can vary, like in a crash where smoke may not always be present, the rigid sequential dependence inherent in FSM may not be desirable. An alternate method that makes use of a parallel algorithm, possibly implemented via threading, could be a possible solution for accurately inferring events based on the latest features detected. This algorithm would keep track of and contrast newly discovered features on a constant basis. Regardless of the order of features, it would notify the main program of the detected event once it was identified. When feature sequences are inconsistent or unpredictable, this approach can still be used to recognize events with flexibility.

3

Motorsports Event Detection

The creation of a motorsport event detection system that uses visual input to pinpoint crucial points in a race is covered in this chapter. Through frame-by-frame image analysis, the application is made to evaluate video footage and identify significant occurrences, such as race milestones. The methodology utilizes proven techniques from the state of the art, such as classifying events using a hierarchical framework and identifying particular moments of significance by chaining lower-level visual and textual elements. Optical Character Recognition (OCR), one of the main methods employed in this application, makes it possible to identify text displayed on the screen, including driver names, lap numbers, and other race-related data. This passage is essential for recognizing particular incidents and connecting them to particular race timestamps. The development of this application also included information extraction from the leaderboard and the use of histogram comparison to determine whether a scene has changed. Although the program takes certain important ideas from the state of the art, it purposefully refrains from using other widely used methods. For instance, sophisticated car detection models and racetrack line detection algorithms—both of which are frequently employed in more intricate systems—are not included in the program. Furthermore, the program simply analyzes and processes the visual information from the movie, it does not undertake any audio analysis or processing. In the same way as a spectator watches a race, the program is made to take input from a single video channel. It provides an effective means of tracking and documenting the race's progress by processing the video frame by frame, recognizing and logging the timestamps of significant events as they happen. By recognizing times of interest only based on the race's visual content, this approach guarantees that the application replicates the experience of a viewer. The application will have this structure:

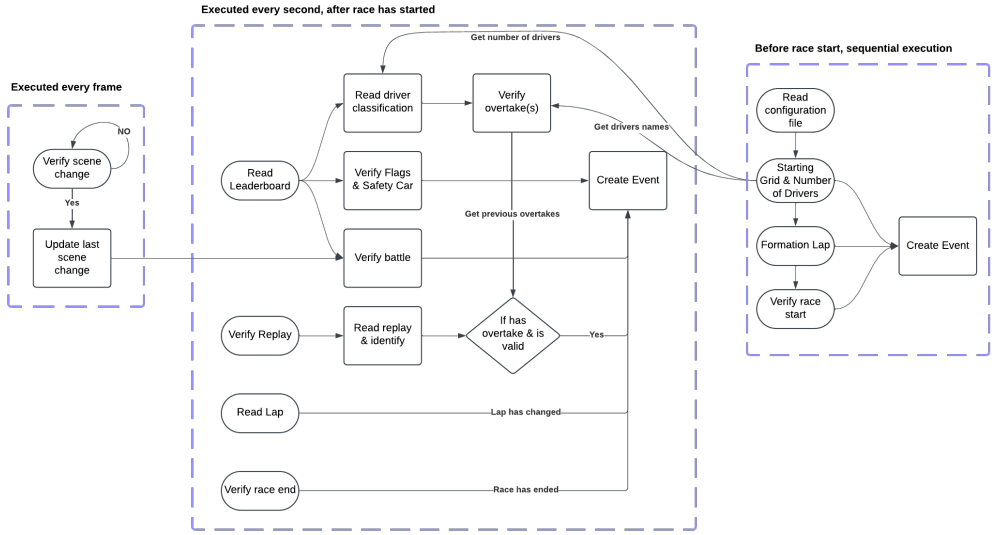


Figure 3.1: Template Matching

The application will be developed in C++ due to its high-performance capabilities and its robust support for object-oriented programming (OOP) principles. C++ provides the ability to produce highly optimized code, ensuring efficient resource utilization, which is critical for performance-intensive applications. Additionally, its OOP features, such as encapsulation, inheritance, and polymorphism, enable the design of modular, maintainable, and scalable systems, aligning well with the project's requirements for flexibility and code reusability. All of the processes shown above will be explained in the following sections, aswell as the supporting algorithms needed to make them work correctly and seamlessly.

3.1 Image Similarity

Certain visual components may recur throughout a video stream, frequently signifying their importance to the recorded event. These recurrent components may indicate significant times, such as alerts, flags, or other essential points for the progression of the event. The detection and analysis of these visual clues are crucial for comprehending the context of the live transmission.

There exist a multitude of techniques for determining the timing of various components in a video. The approaches encompass a spectrum of methodologies, ranging from simple pattern recognition or color matching to intricate algorithms that use machine learning or sophisticated image processing. Although more advanced techniques may provide more accuracy and adaptability, simpler methods can still be very efficient and resource-efficient, making them suitable for specific applications that prioritize real-time processing or limited computing capacity.

3.1.1 MSE, PSNR and Histogram Comparison

The histogram comparison method quantifies the similarity of two images by comparing their color distributions, often known as histograms. A histogram is a visual depiction that illustrates the occurrence rate of individual colors or intensity values in a picture, therefore facilitating the capture of the general color or intensity patterns rather than precise pixel values. This technique is less influenced by spatial variations and emphasizes the overall content rather than exact pixel alignment.

Conversely, pixel-based techniques such as PSNR and MSE evaluate visual similarity by directly comparing the brightness of individual pixels. An image similarity algorithm can be implemented as a straightforward process of subtracting two pictures, as exemplified by the instance of Mean Square Error (MSE), which computes the mean squared difference between matching pixels in two pictures, requiring that both images are of equal dimensions and have same number of channels.

$$MSE = \frac{1}{mn} \sum_{i=0}^{m-1} \sum_{j=0}^{n-1} \left[I_2(i,j) - I_1(i,j) \right]^2 \quad \text{where } \mathbf{m} \text{ is rows and } \mathbf{n} \text{ columns} \tag{3.1}$$

Thus, the simplicity of MSE might be a disadvantage. It has a high degree of sensitivity towards minor deviations, such as noise present in one of the pictures. Insignificant variations at the pixel level might cause a large MSE number, which may lead the algorithm to mistakenly determine that the images are not identical. This problem arises when MSE fails to differentiate between substantial variations and those that are perceptually inconsequential, such as random noise.

This issue can be solved with Peak signal-to-noise ratio (PSNR), obtained from MSE, which standardizes the error in relation to the highest attainable pixel value and modeling it on a logarithmic decibel (dB) scale. The logarithmic scale of PSNR, (3.2), reduces the sensitivity of the signal to small mistakes, therefore mitigating the influence of slight variations that have minimal impact on the perceived image similarity. Thus, PSNR is more resilient in comparing pictures where incremental differences are not crucial, offering a more precise assessment of similarity even in the presence of noise or other little alterations.

$$PSNR = 10 \cdot \log_{10} \left(\frac{MAX_I^2}{MSE} \right) \quad \text{where } MAX \text{ is the maximum pixel value} \tag{3.2}$$

Consistently, all three of the discussed approaches analyze the color composition of the two images being compared. MSE is a fundamental technique suitable for simple pictures with minimal noise. On the other hand, PSNR can handle residual noise but requires more processing capacity. Ultimately, histogram comparison remains unaffected by the rotation and translation of a picture or some components inside it. However, if, for instance, the physical structure of a picture is changed but the color composition stays unchanged, any of these approaches might result in false positives.

3.1.2 Structural Similarity Index Measure

The Structural similarity index measure (SSIM) is a quantitative measure specifically developed to evaluate the similarity of images by finding visual information that the human eye can perceive. Contrary to conventional measures such as MSE and PSNR, which assess variations in intensity at the pixel level, SSIM quantifies pictures by taking into account their structural information, which includes three primary components: brightness, contrast, and structure;

$$\text{SSIM} = l(x, y)^{\alpha} \cdot c(x, y)^{\beta} \cdot s(x, y)^{\gamma} \tag{3.3}$$

Typically, α, β, and Υ are all set to 1, [28], so the formula becomes a simple product of these three components:

$$\text{SSIM}(x, y) = \frac{(2\mu_x\mu_y + c_1)(2\sigma_{xy} + c_2)}{(\mu_x^2 + \mu_y^2 + c_1)(\sigma_x^2 + \sigma_y^2 + c_2)} \tag{3.4}$$

With μ_x being the mean average of the first image, μ_y the mean average of the second image, σ_x the covariance of the first image, σ_y the covariance of the second image and, lastly, c_1 and c_2 are variables utilized to stabilize the computation where certain values could be very small. Optimal values were found to be 6.5 and 58.5, respectively [29].

SSIM quantifies the degree of similarity between the structural patterns of the original picture and the reference image by comparing these three previous components within tiny local areas of the image. The implementation of this localized technique guarantees the inclusion of differences across several locations, therefore enhancing the precision of classifying image similarity.

Through assessing the degree of alignment between the structural information of a picture and the original, SSIM provides a more perceptually significant metric of similarity. The resilience of SSIM to subtle variations at the pixel level, which may be of little importance to human perception, enables it to more effectively preserve visual quality in real-world scenarios compared to conventional measures such as MSE and PSNR, but at the expense of increased computational resources.

3.1.3 Template Matching

This chapter concludes with the most adaptable approach discussed, Template Matching. The objective is to determine the exact location of a template, which is a smaller picture or pattern, inside a bigger sized image. The matching process entails the procedure of moving the template across the source picture and then comparing each individual subregion of the source image with the template. For each point, a similarity measure is computed, and the site exhibiting the greatest similarity is determined as the match.

There are two categories of template matching: Intensity-Based and Feature-Based strategies. The first approach is the most direct, including the comparison of pixel intensity levels between the template and subregions of the source picture. This method is effective

for images characterized by a consistent visual appearance and lighting circumstances. Conversely, the latter process involves comparing unique characteristics (such as edges, corners, crucial spots, or curves) between the template and the original picture. This approach can exhibit more resilience to factors like as size, rotation, or differences in illumination.

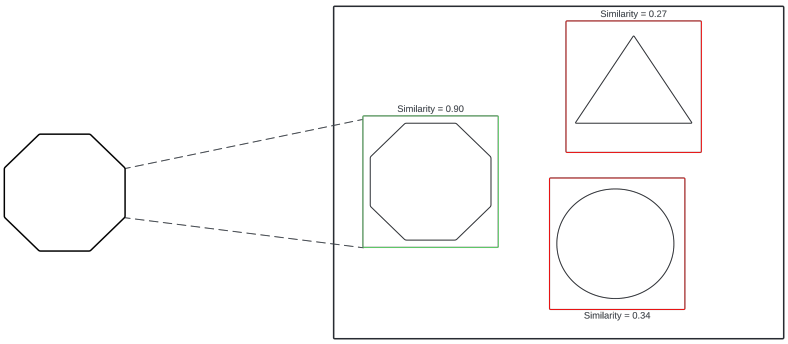


Figure 3.2: Template Matching

Predictably, Template Matching has some limitations. It is very responsive to variations in scale and rotation, as well as lighting conditions, as certain metrics rely on pixel intensity values but has a great drawback which is its exection time. By traversing a picture in search of a corresponding pair for the template, this process may be quite resource-intensive. For this application the template will always have the same size because it does not change from race to race.

3.2 Overtake Detection and Starting Grid

In order to precisely identify overtakes, it is crucial to extract and examine the leaderboard from the video live stream. To carry out this procedure, it is essential to determine the location of the leaderboard within the video frame. Considering the unique positioning of each leaderboards in each major racing series, it becomes imperative to develop a configuration file. This file, created in JavaScript Object Notation (JSON) format, specifies the position of each active racing series on the leaderboard. This file structure and function will be discussed in the following sections.

Once the position of the leaderboard is established, it is divided into segments that correspond to the number of drivers involved in the race. Following the segmentation, the picture is processed to achieve precise interpretation by the OCR algorithm, as explained in Chapter **2.4.2**. This processing includes binarization using the Otsu method and the implementation of morphological operations.

Although the TextRecognitionModel in OpenCV has a similar functionality, it failed to produce satisfactory outcomes for this particular situation. Tesseract's OCR technique requires black characters applied to a white backdrop to achieve accurate results. Consequently, depending on the frame, an inversion may be necessary. For determination of that need, the program computes the proportion of white pixels in the provided image:

$$Ratio = \frac{countNonZero(frame)}{Width * Height} \tag{3.5}$$

Furthermore, apart from the inversion, morphological techniques are used to modify the thickness of the text. The OCR method has improved performance when used to identify thin letters. An appropriate ratio is calculated, similar to the one shown above, to ascertain whether erosion, which decreases letter thickness, or dilation, increases letter thickness, is justified.

After undergoing processing, Figure **3.2**, the outcomes derived by Tesseract's OCR method are stored as a vector of strings. To identify instances of mismatched positions, usually indication overtakes, this vector is compared with the previous driver rankings, deliberately excluding entries that include certain terms, which will be explained further on this chapter. Upon detecting a discrepancy between the prior and present ranking, the algorithm infers that an overtaking has taken place.

Figure 3.3: Original leaderboard

Figure 3.4: Processed segment

Figure 3.5: Leaderboard processing

Having achieved accurate reading of the leaderboard, the program now requires knowledge of the situations in which it is unable to accomplish this task. An example of such a situation occurs when a driver chooses to withdraw from a race. Regarding retirement, drivers frequently experience a significant decrease in their ranking before formally retiring, resulting in overtakes that lack contextual significance for shooting highlight videos. In order to address this issue, the program promptly deletes all previous overtakes linked to the retiring driver on the detection of the word "OUT".

Figure 3.6: Processing retired drivers

Another potential situation that might lead to incoherent overtakes is pit stops. In order to tackle this issue, the term "PIT" is added to the name of a driver specifically during their pit stops. This system guarantees that overtakes conducted by a driver while

pitting are not mistakenly documented as authentic, therefore maintaining the integrity of the race analysis.

The initial grid utilizes a very similar procedure due to its similarly structured display arrangement. At the start of the race, the conventional leaderboard position displays the initial tyres used by each driver. The use of this feature enables the algorithm to precisely ascertain the starting quantity of drivers, a critical factor for the segmentation phase of overtaking detection. Furthermore, it provides the precise identities of the drivers, which is crucial for verifying following identifications, as will be elaborated upon in coming sections.

3.3 Element Detection on Leaderboard

Given that the identification of overtakes is dependent entirely on the leaderboard, it is imperative for the program to recognize scenarios in which the leaderboard cannot be reliably interpreted or when overtakes are prohibited. The identification of these situations is based on the observation of flags or other visual markings inside the leaderboard region, which have the potential to disrupt the precise interpretation of the driver ranks, as previously mentioned. Therefore, in the event of such situations, the division of the leaderboard is modified correspondingly. Certain elements can be either static, meaning their location remains constant, or dynamic, meaning their position might vary during a race.

Deterrents such as yellow flags and safety cars forbid overtaking, while red flags, Figure **3.3**, momentarily suspend the race. While some modifications are required for the application to operate well, identifying these components is very straightforward. This simplicity of detection arises from the static nature of these components, which remain in the same position constantly throughout a race and frequently for an entire season. Formation laps are identified in the same manner as these flags.

Several image similarity algorithms were assessed throughout the development phase, and PSNR was established as the optimal selection for this specific application. Previous research has demonstrated that a simple and resource-efficient algorithm, such as PSNR, can accurately detect the presence of warning flags and safety vehicle indications, which typically have a uniform structure and color due to their solid color backgrounds.

The configuration of the leaderboard can also be shaped by variables such as a competition between two drivers or the fastest lap completed thus far. While not explicitly forbidding overtakes, these features alter the structure of the leaderboard, which may thereby undermine the effectiveness of overtake detection for this specific application. While the preceding components were readily recognizable, they nonetheless remained static. Given the dynamic nature of their position throughout a race, a simple algorithm relying on pixel intensity will be inadequate. Identifying these components necessitates the utilization of Template Matching, as described in Chapter **3.1**. By traversing a picture,

this approach is able to identify the template regardless of its location, as long as it has the same color and structure.

Figure 3.7: Dynamic element

Figure 3.8: Static element

Figure 3.9: Examples of elements present on the leaderboard

The primary purpose of this information is to assist in the accurate operation of overatake detection. However, it may also be utilized to give the end user valuable information on more intricate aspects of the race.

3.4 Replay Identification

An examination of replays is essential for identifying the pivotal moments throughout a race. They epitomize past overtakings, collisions, and other crucial occurrences that mold the narrative of a race. Oftentimes, these incidents are either inadequately captured by camera focal points during the live broadcast or, in other instances, not shown at all. Hence, accurate detection and identification of such occurrences are essential for systems seeking to automatically detect events.

The initial stage, detection, is quite uncomplicated. Each time a replay is displayed, a little symbol regularly appears in the upper left corner of the screen to signify the replay, Figure **3.5**. This symbol remains consistent throughout all replays and races, facilitating quick identification. The efficient identification of static components within a frame may be achieved utilizing PSNR, as examined in the preceding chapter. Once a replay is detected, the software stops all concurrent processes and focuses its efforts on identifying the replay.

Once the program detects the presence of a replay, it proceeds to identify the replay by pinpointing a specific area at the bottom of the screen, Figure **3.5**, where the broadcaster displays relevant information about the replay, including the drivers involved. The area is then processed using an algorithm that considers the level of variability in its dimensions.

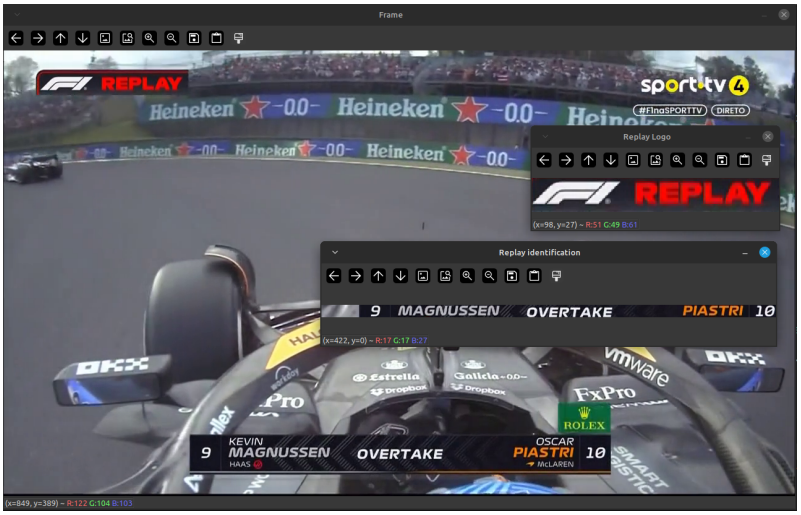


Figure 3.10: Replay identification and detection sections

Optimal performance of Tesseract's OCR is achieved when the frame completely encloses the text. However, during replay identification, the analysis of frames may provide text of different sizes. This requires the development of an algorithm with the ability to precisely identify and separate text inside a frame. This technique guarantees the correct encapsulation of the text within the frame, therefore raising the efficiency of the OCR process and enhancing the precision of the recovered information.

The method starts by applying a mask specifically created to visually emphasize only the background region where text or the pertinent portion of the frame is located. The lack of a predetermined mask enables the algorithm to be flexible and suitable for a wide range of applications. Upon completion of masking, the algorithm proceeds to detect transitions from a white pixel to a black pixel, which might potentially signify the beginning of a pertinent region for extraction. Should the algorithm identify a white pixel interval that surpasses 3% of the total width or height of the frame, it is probable that it has detected a prospective region of interest. In addition, if this region has a black pixel count above 1% of the frame's overall size, the algorithm verifies its significance and proceeds to explore other relevant regions inside the same frame.

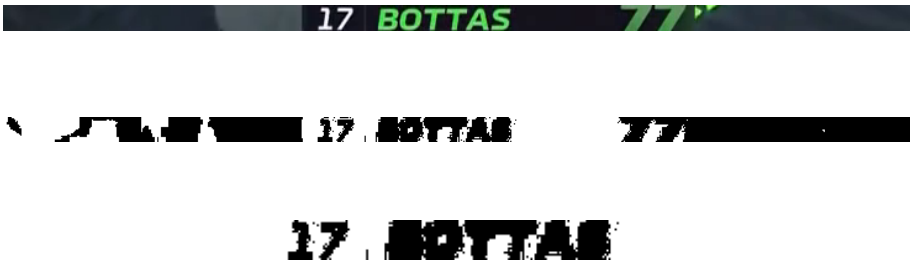


Figure 3.14: Find text algorithm, firstly applies the mask and then finds relevant area

Due to its restricted search of a single line or row of a frame, this method achieves rapid and efficient text identification. The algorithm's emphasis on individual lines or rows serves to decrease the computational burden and improve processing speed, therefore

rendering it very efficient for applications that need quick text identification.

Having demonstrated the ability to retrieve visible data, the software now needs the knowledge of how to interpret it. Drawing from the races examined up to this point, this region routinely features the name of at least one driver. Alternatively, the car in question may have recently executed an overtaking move. The validation procedure entails the comparison of the driver's name shown during the replay with the documentation of overtaking incidents that took place earlier in the race.

One other method to verify if a replay is exhibiting a prior overtaking is to assess the presence of both the car initiating the overtake and the one being overtaken. Similarly to the previous method, this requires cross-referencing with past overtakes to confirm its precise nature as an overtake. Lastly, the replay space specifically allocated for identification can incorporate the keyword 'Overtake', therefore allowing the software to promptly ascertain the nature of this replay without requiring reference to its previous overtakes.

Furthermore, replays frequently showcase not just overtakes but also other noteworthy occurrences such as accidents and pit stops. Exactly like overtakes, when one of these occurrences happens, the designated region for identification includes specific phrases like 'INCIDENT' and 'PIT', which promptly notify the software about the type of replay. The use of keyword-based identification enables the system to efficiently and precisely classify replays, therefore guaranteeing the correct recognition and processing of all crucial points of the race.

3.5 Detecting Scene Changes

Scene change detection relies on the analysis of disparities in different picture similarity metrics across successive frames. A scene transition is deemed to have taken place when two frames display substantial disparities, signifying a shift from one camera to another.

An essential first step in assessing scene changes is to compare the histograms of two frames, which serves as a rudimentary method for identifying scene transitions. If the comparison of the histogram produces a substantial value, the software abstains from computing further metrics in order to ascertain the occurrence of a scene change. The adoption of this method is justified due to the computational efficiency of histogram comparison, which is beneficial in this particular situation as each frame includes this processing. SSIM and PSNR measurements are calculated to verify the existence of a scene transition. In particular, if the SSIM value is less than 0.52 and the average PSNR of the last three frames is less than 5.0, the software determines that a scene change has taken place. In this specific scenario, SSIM is computed by taking the mean of the values of each BGR channel.

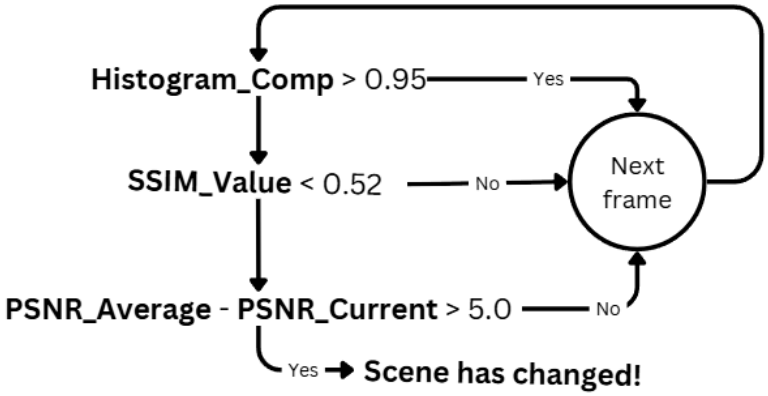


Figure 3.15: Scene change algorithm

Utilizing the average PSNR of the three previous frames is aimed at minimizing the likelihood of false positive alerts. Through the computation of the mean of these values, the influence of abnormally high readings is diminished. Moreover, that average only considers PSNR values of 30 or below, while values above 30 are often seen in two very identical frames. The program ensures that the vector containing PSNR values is always kept with exactly three entries.

In this application, scene changes are mostly used to determine timestamps of a past driver battle. Specifically, upon detecting a dispute, the software documents the preceding scene change and then waits for the next scene change once the battle has ended. This methodology enables the software to produce a precise timestamp that may be utilized to emphasize the occurrence.

3.6 Configuration File

Within motorsports, every major racing series operates according to its own unique regulations, circuit layouts and presentation styles. Notwithstanding these differences, there are several aspects that are widely recognized and used in several series, such as leaderboards, formation laps and other conventional components.

To guarantee the functioning of a system across different streams from distinct races, it is essential to configure certain parameters that account for the existence of these common features. By utilizing a configuration file, the software gains a high degree of versatility and adaptability to meet the particular demands of each racing events.

This configuration file defines the absolute values of the top-left and bottom-right coordinates of a certain element, enabling the system to calculate the width and height of the element that has to be removed from the original frame. Furthermore, the file provides the system with information on the precise components that are now included in the video stream.

JSON is the chosen format for the file due to its ease of interpretation by humans, language independence and efficiency. The JSON data format is a very efficient representation of structured data, offering smooth integration with online services and extensive

compatibility with many computer languages.

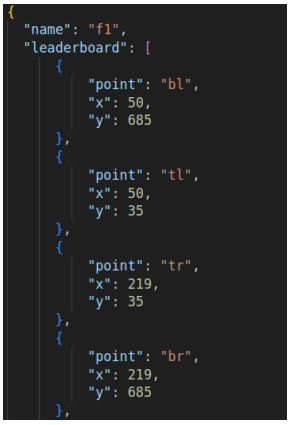


Figure 3.16: Sample of configuration file

3.7 Lap Extraction

The lap extraction procedure adheres to a methodology that is comparable to other processes employed in this thesis. Specifically, the procedure entails choosing a pertinent area from a frame, Chapter **3.4**, analyzing it, and then using a OCR algorithm to decipher the displayed text. The preceding chapters examined an alternative OCR algorithm called **TextRecognitionModel**, which was created by OpenCV. Despite its limited suitability for most applications due to poor results and prolonged execution times, this technique exhibits exceptional performance for jobs that involve tiny numerical values, such as lap extraction. Under these particular circumstances, it surpasses the Tesseract algorithm in terms of precision.

Given the enormous computing resources involved, the **TextRecognitionModel** should only be invoked when necessary, specially when a lap change is recognized. To enable this, the application maintains a binary duplicate of the preceding lap. By subtracting the current frame, which has the updated lap information, from the stored frame, the software may establish if a change has happened. If the lap remains the same, the ensuing subtraction provides a dark picture, indicating no differences between the frames. Conversely, if the lap has changed, the subtraction provides a picture with white pixels showing the changes, Figure **3.7**.

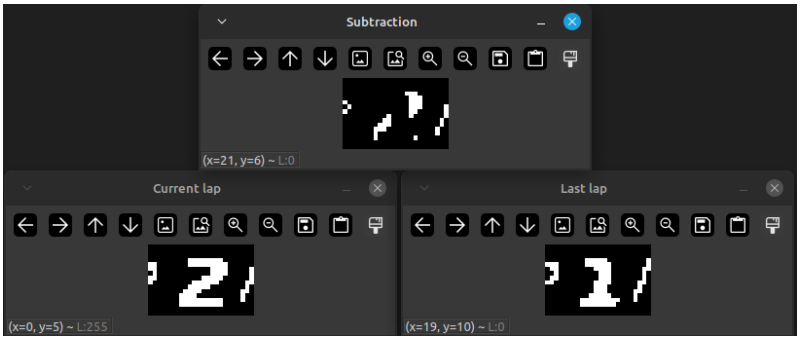


Figure 3.17: Subtraction of previous lap and current lap

As a consequence, the **TextRecognitionModel** is only active when the subtraction generates white pixels, so preserving computing resources by limiting its use to instances where changes are detected. In order to mitigate minor flaws resulting from noise in a stream, the algorithm focuses on determining the proportion of white pixels rather than determining whether the subtraction includes white pixels or not. In this specific scenario, any amount above 1% of white pixels is classified as a lap change.

Furthermore, apart from identifying a change in lap, it is imperative to authenticate the accuracy of the identified change. This validation procedure consists of two criteria: firstly, verifying that the recently identified lap is distinct from the current lap, and secondly, guaranteeing that the discrepancy between the newly detected lap and the last is not greater than two. Under red flag conditions, the leaderboard is momentarily suspended and reintroduced only after the formation lap, indicating that a minimum of two laps have elapsed since the last display of the leaderboard.

3.8 Final results and Future Improvements

This section assesses the accuracy of the software, specifically developed for the analysis and processing of motorsport video streams, by examining four pre-selected Formula 1 race streams. Formula 1 is the sole motorsport that the program accurately functions at the moment, but can be configured to work on other series, as mentioned above. This selection comprises feeds from four Grand Prix events held in Japan, China, Australia, and Canada during the 2024 season. While any Formula race has the potential to be used because of its standardized format, Formula 1 was specifically selected because of its position as one of the top series overseen by the Fédération Internationale de l'Automobile (FIA).

All streams were captured at a consistent resolution of 1280x720 at a rate of 25 frames per second, using the same equipment and format, to guarantee uniformity in data intake. This standardization enables a meticulous assessment of the software performance in races with varying numbers of starting drivers, to which the program may adjust. The following table displays the outcomes of various tests:

	Japan	China	Australia	Canada
Fastest laps	6/6	4/4	18/18	46/46
Flags & S.Car	3/3	4/4	4/4	9/9
Battles	10/10	5/5	11/11	9/9
Replays	24/24	19/19	15/15	19/19
Processing time	28m30	29m6s	23m43s	28m43s
Video length	147m17s	124m59s	110m01s	118m44s

Table 3.1: Final results

An analysis of the table above indicates that the produced application has a relatively lower execution time when compared to the video length and has perfect detection of the more common, and simpler to identify, elements. Although the majority of evaluated

measures exhibited practically impeccable outcomes, replay identification presented a significant difficulty, as shown on the table below. This weaker characteristic stemmed from the insufficient presence of distinctive visual features in replays, which are crucial for accurate detection. Some recordings are primarily identifiable by behavioral patterns, such as a vehicle crossing track borders, rather than distinct visual cues, complicating their recognition during replay analysis.

	Japan	China	Australia	Canada
Race Start	2/2	1/1	1/1	2/2
Overtake	4/8	9/12	4/4	7/11
Crash	2/3	0/3	-	2/6
Pit Stop	2/3	4/4	1/2	-
Penalty/Track Limits	0/7	0/3	0/3	0/5
Retirement	-	1/1	3/5	1/1
Other	0/2	-	0/1	-
End Celebration	1/1	1/1	1/1	1/1
False Positives	2	1	2	3

Table 3.2: Replay Identification

The application not only occasionally fails to correctly recognize certain replays but also experiences some false positives, which is shown on the last column. These erroneous detections frequently arise from broadcasters presenting visual signals that are deceptive within the context of the replay. This is the case when the screen depicts a car who has successfully executed an overtaking maneuver before the replay was shown but the broadcaster decided not to show it. Although, the biggest reason of false positives is the application's inability to recognize newly emerging replay formats that have not been incorporated into the detection algorithm. With each new race broadcast, innovative replay forms may arise, heightening the risk of misidentification unless the application is prepared to handle or ignore any identification when new formats are present.

Currently, the scene change detection mechanism is notably time-consuming because it utilizes three parameters to ascertain the occurrence of a scene change. To alleviate this issue, the application should be optimized in order to utilize a singular measure, ideally one that operates swiftly. Rather than juxtaposing a metrics value with a fixed threshold, the application needs to assess significant changes in the metric across consecutive frames, since this serves as a definitive signal of a scene transition. Furthermore, reducing the image to one-third of its original dimensions might alleviate processing requirements. These steps would boost the application's performance by decreasing execution time and improving its adaptability to new broadcast formats, hence minimizing false positives and missed detections.

Finally, one final improvement might include the creation of an algorithm capable of automatically identifying essential components, such as the leaderboard, inside a frame. This method would allow the application to be flexible with various racing series, therefore

reducing or maybe obviating the necessity for manual configuration files. This flexibility not only optimizes system maintenance but also improves user experience by minimizing downtime and manual setup mistakes. This approach may also be employed to identify and rectify discrepancies in the streaming formats of racing series that may arise between seasons, as these subtle changes might impair the application's performance.

One approachable way to accomplish this is done by computing the average frames data inside a certain race. This procedure entails aggregating pixel data from several frames to detect repeating visual components, which are subsequently subjected to statistical analysis to emphasize elements of interest. Elements that are displayed more frequently on the screen are allocated greater values than those that are less prevalent. Utilizing a mask to augment the contrast of particular parts enables the use of a contour detection algorithm to precisely ascertain the locations of these items inside each frame. This adaptive method guarantees the algorithm's operability across various series and seasons while enhancing the application's overall dependability and efficiency.

The application's performance, especially in terms of detection accuracy, dependability, and processing speed, might be markedly enhanced by certain interventions. Firstly, augmenting the number of races within the test set would provide a more thorough evaluation of detection capacities across different groups. Moreover, improving the overtaking detecting capabilities might augment accuracy in dynamic settings. Furthermore, including an automatic element detection method would enhance the adaptability of the application without needing to completely alter the core behavior of the application. Ultimately, simplifying the scene change detection procedure would enhance system efficiency by lowering complexity.

4

Conclusions

Currently, there are several published methodologies for effective autonomous event detection systems in sports such as football, tennis, and basketball; however, there are few for motorsports. This thesis seeks to mimic the functionality and key ideas of the aforementioned systems and apply them to motorsport streams. The application created in this study can detect significant racing occurrences, like overtakes, collisions, and pit stops, at a fast pace by evaluating video and textual elements from live feeds. This method offers an optimized watching experience that improves the accessibility and engagement of racing broadcasts, catering to the increasing need for concise, highlight-focused race reports.

The proposed method illustrates the viability of real-time event detection, functioning at velocities three times swifter than live broadcasts. This facilitates the prompt dissemination of updates, ensuring that remote spectators remain engaged with the pivotal moments of a race. Advanced algorithms to find image similarities and text extraction were utilized to enhance the accuracy and dependability of the event detection process. The system's versatility is augmented by the use of configuration files, enabling it to be operated across several motorsport formats, if needed, therefore rendering it a flexible option for forthcoming racing events.

Nonetheless, obstacles persist, especially in identifying intricate replays and adjusting to novel broadcast formats that may differ throughout seasons or races. Augmenting the replay identification algorithm with an expanded test set would provide near-comprehensive coverage and analysis of each replay presented during a racing event. Moreover, automating the identification of essential components like leaderboards will diminish the necessity for human setup and enhance the system's adaptability. Future endeavors will seek to tackle these problems, emphasizing the enhancement of the system's flexibility, the minimization of false positives, and the augmentation of its capacities to accommodate new and developing race forms.

This thesis demonstrates the capacity of automated event identification to revolutionize the viewing experience for motorsport enthusiasts. This research establishes a foundation

for future advancements in the field by tackling existing technical constraints and investigating opportunities for system improvement. This study illustrates the feasibility of sophisticated computer vision techniques in a complex and dynamic setting while also paving the way for more innovation. With further enhancements, this technology has the potential to become a vital resource for broadcasters and enthusiasts, rendering racing more accessible, captivating and pleasurable.

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] E. Cheng and N. Jarvis. "Residents' Perception of the Social-Cultural Impacts of the 2008 Formula 1 Singtel Singapore Grand Prix". In: *Event Management* 14 (2010-09), pp. 91–106. DOI: 10.3727/152599510X12766070300849 (cit. on p. 3).

[3] L. L. Mao and H. Huang. "Social impact of Formula One Chinese Grand Prix: A comparison of local residents' perceptions based on the intrinsic dimension". In: *Sport Management Review* 19 (3 2016), pp. 306–318. ISSN: 1441-3523. DOI: https://doi.org/10.1016/j.smr.2015.08.007. URL: https://www.sciencedirect.com/science/article/pii/S1441352315000777 (cit. on p. 3).

[4] B. Ramasamy, H. Wu, and M. Yeung. "Hosting annual international sporting events and tourism: Formula 1, golf or tennis?" In: *Tourism Economics* 28 (8 2022-12), pp. 2082–2098. ISSN: 1354-8166. DOI: 10.1177/13548166211029053 (cit. on p. 4).

[5] H. Inani et al. "The Intersection of Industry 4.0 with Formula 1 and Other Motorsports: Future of Racing". In: 2024, pp. 171–192. DOI: 10.1007/978-981-97-3173-2_12 (cit. on p. 4).

[6] B. Li and I. Sezan. "Event detection and summarization in sports video". In: 2001-02, pp. 132–138. ISBN: 0-7695-1354-9. DOI: 10.1109/IVL.2001.990867 (cit. on pp. 4, 10).

[7] Q. Huang and S. Cox. "Hierarchical language modeling for audio events detection in a sports game". In: *2010 IEEE International Conference on Acoustics, Speech and Signal Processing*. 2010, pp. 2286–2289. DOI: 10.1109/ICASSP.2010.5495935 (cit. on p. 5).

[8] S. Júnior, A. Araújo, and D. Menotti. "An overview of automatic event detection in soccer matches". In: 2011-01, pp. 31–38. DOI: 10.1109/WACV.2011.5711480 (cit. on p. 5).

[9] K. Bastin and C. Healey. "Visual Analytics for NASCAR Motorsports". In: 2023-09. DOI: 10.21203/rs.3.rs-3348518/v1 (cit. on p. 6).

[10] 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. 6).

[11] M. Baillie and J. Jose. "Audio-Based Event Detection for Sports Video". In: vol. 2728. 2003-07, pp. 300–309. ISBN: 978-3-540-40634-1. DOI: 10.1007/3-540-45113-7_30 (cit. on pp. 6, 7).

[12] D. Ziou and S. Tabbone. "Edge Detection Techniques - An Overview". In: 8 (2000-06) (cit. on p. 7).

[13] V. Mohan. "Performance Analysis of Canny and Sobel Edge Detection Algorithms in Image Mining". In: *International Journal of Innovative Research in Computer and Communication Engineering* (2013-10), pp. 1760–1767 (cit. on pp. 7–9).

[14] P. Li et al. "Overview of Image Smoothing Algorithms". In: *Journal of Physics: Conference Series* 1883 (2021-04), p. 012024. DOI: 10.1088/1742-6596/1883/1/012024 (cit. on p. 7).

[15] J. Illingworth and J. Kittler. "A survey of the hough transform". In: *Computer Vision, Graphics, and Image Processing* 44.1 (1988), pp. 87–116. ISSN: 0734-189X. DOI: https://doi.org/10.1016/S0734-189X(88)80033-1. URL: https://www.sciencedirect.com/science/article/pii/S0734189X88800331 (cit. on p. 9).

[16] OpenCV. *Hough Line Transform*. 2024-12 (cit. on p. 9).

[17] A. Ekin and A. Tekalp. "Generic play-break event detection for summarization and hierarchical sports video analysis". In: 2003-08, pp. I–169. ISBN: 0-7803-7965-9. DOI: 10.1109/ICME.2003.1220881 (cit. on p. 10).

[18] H. Pan, P. van Beek, and M. Sezan. "Detection of slow-motion replay segments in sports video for highlights generation". In: *2001 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No.01CH37221)*. Vol. 3. 2001, 1649–1652 vol.3. DOI: 10.1109/ICASSP.2001.941253 (cit. on p. 10).

[19] X. Lu et al. "MimicDet: Bridging the Gap Between One-Stage and Two-Stage Object Detection". In: 2020-11, pp. 541–557. ISBN: 978-3-030-58567-9. DOI: 10.1007/978-3-030-58568-6_32 (cit. on p. 10).

[20] J. Deng et al. "A review of research on object detection based on deep learning". In: *Journal of Physics: Conference Series* 1684 (1 2020-11), p. 012028. ISSN: 1742-6588. DOI: 10.1088/1742-6596/1684/1/012028 (cit. on pp. 10, 11).

[21] A. Kumar, J. Zhang, and H. Lyu. "Object detection in real time based on improved single shot multi-box detector algorithm". In: *EURASIP Journal on Wireless Communications and Networking* 2020 (2020-10). DOI: 10.1186/s13638-020-01826-x (cit. on p. 10).

[22] N. Boyko, O. Basystiuk, and N. Shakhovska. "Performance evaluation and comparison of software for face recognition, based on dlib and opencv library". In: *2018 IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*. IEEE. 2018, pp. 478–482 (cit. on p. 11).

[23] C. Xu et al. "Live sports event detection based on broadcast video and web-casting text". In: 2006-10, pp. 221–230. DOI: 10.1145/1180639.1180699 (cit. on pp. 12, 14).

[24] Y. Zhu, C. Yao, and X. Bai. "Scene text detection and recognition: recent advances and future trends". In: *Frontiers of Computer Science* 10 (1 2016), pp. 19–36. ISSN: 2095-2236. DOI: 10.1007/s11704-015-4488-0. URL: https://doi.org/10.1007/s11704-015-4488-0 (cit. on pp. 12, 13).

[25] J. Gllavata, R. Ewerth, and B. Freisleben. "A robust algorithm for text detection in images". In: *3rd International Symposium on Image and Signal Processing and Analysis, 2003. ISPA 2003. Proceedings of the*. Vol. 2. 2003, 611–616 Vol.2. DOI: 10.1109/ISPA.2003.1296349 (cit. on p. 13).

[26] K. Kanagarathinam et al. "Steps Involved in Text Recognition and Recent Research in OCR; A Study". In: 8 (2019-05), pp. 3095–3100 (cit. on p. 13).

[27] R. Mittal and A. Garg. "Text extraction using OCR: A Systematic Review". In: *2020 Second International Conference on Inventive Research in Computing Applications (ICIRCA)*. 2020, pp. 357–362. DOI: 10.1109/ICIRCA48905.2020.9183326 (cit. on p. 13).

[28] Z. Wang et al. "Image Quality Assessment: From Error Visibility to Structural Similarity". In: *Image Processing, IEEE Transactions on* 13 (2004-05), pp. 600–612. DOI: 10.1109/TIP.2003.819861 (cit. on p. 18).

[29] OpenCV. *Video Input with OpenCV and similarity measurement*. 2024-12 (cit. on p. 18).