INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

Deteção de eventos relevantes de automobilismo com base em Computer Vision

Catarina Maria Rodrigues Mota

Licenciada

Projecto Final para obtenção do Grau de Mestre
em Engenharia Informática e Multimédia

Orientadores : Prof. Doutor Pedro Mendes Jorge
Xavier Frazão

Júri:

Presidente: Prof. Doutor Rui Manuel Feliciano de Jesus

Vogais: Prof. Doutor André Ribeiro Lourenço
Prof. Doutor Pedro Mendes Jorge

setembro, 2023

INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA

Departamento de Engenharia de Electrónica e Telecomunicações e de Computadores

Deteção de eventos relevantes de automobilismo com base em Computer Vision

Catarina Maria Rodrigues Mota

Licenciada

Projecto Final para obtenção do Grau de Mestre
em Engenharia Informática e Multimédia

Orientadores : Prof. Doutor Pedro Mendes Jorge
Xavier Frazão

Júri:

Presidente: Prof. Doutor Rui Manuel Feliciano de Jesus

Vogais: Prof. Doutor André Ribeiro Lourenço
Prof. Doutor Pedro Mendes Jorge

setembro, 2023

Aos meus pais, avó, irmão, restante família, namorado,
amigos chegados e Ruca.

Acknowledgments

I extend my heartfelt gratitude to all those who have contributed to the completion of this project. Without your dedication, support, and expertise, the success of this project would not have been possible.

First and foremost, I would like to express my profound appreciation to my supervisors for their invaluable guidance, insightful feedback, and unwavering encouragement throughout the course of this project. Your expertise has been instrumental in shaping our ideas and ensuring the quality of my work.

Mu sincere thanks go to my colleagues for their support and encouragement during these six years I've been at Instituto Superior de Engenharia de Lisboa. I realize I'm a complete different person thanks to you, thanks for boosting my confidence.

Furthermore, I express my gratitude to the staff at Instituto Superior de Engenharia de Lisboa for providing the necessary resources, facilities, and support that facilitated the progress of my formation.

Lastly, I dedicate my acknowledgment to my family, mom, grandmother, dad, brother, cousins, uncles and aunts, cat Ruca, all the loved ones who sadly passed away before this precious moment, my close friends and my boyfriend. Your unwavering support, understanding, and patience have been the foundation upon which we were able to embark on this project.

With the most heartfelt appreciation, to conclude, the completion of this project would not have been possible without the collective effort of everyone mentioned above. Each contribution, no matter how small, has played a vital role in bringing this project to fruition. Thank you for your dedication, support, and commitment.

Abstract

This master's thesis focuses on the development of an advanced program for the automatic extraction of relevant events from motor sports broadcasts. Formula 1 races, with their global audience and dynamic nature, serve as an ideal ground for this research. The project's objective is to enhance the viewing experience for Formula 1 enthusiasts while providing valuable information for analysis purposes. The research begins with an exploration of related work and then delves into the theoretical foundations of the project, including frame comparison methods and pattern extraction techniques. The project's motivation centers around the use of more classical methods for event extraction. The thesis concludes by summarizing the main contributions and discussing their implications, offering a comprehensive understanding of the potential impact of the project on event extraction in broadcasts, particularly in Formula 1 races.

Keywords: Computer Vison, Pattern Recognition, Optical Character Recognition (OCR) ...

Resumo

Esta tese de mestrado concentra-se no desenvolvimento de um programa avançado para a extração automática de eventos relavantes de transmissões de desportos motorizados. As corridas de Fórmula 1, com a sua audiência global e natureza dinâmica, servem como terreno ideal para esta investigação. O objetivo do projeto é melhorar a experiência de visualização para os entusiastas da Fórmula 1, ao mesmo tempo que fornece informações valiosas para fins de análise. A pesquisa começa com uma exploração do trabalho relacionado e depois aprofunda as bases teóricas do projeto, incluindo métodos de comparação de *frames* e técnicas de extração de padrões. A motivação do projeto centra-se na abordagem por métodos mais clássicos para a extração de eventos. A tese conclui resumindo as principais contribuições e discutindo as suas implicações, oferecendo uma compreensão abrangente do potencial impacto do projeto na extração de eventos em transmissões, particularmente em corridas de Fórmula 1.

Palavras-chave: Processamento de Vídeo, Reconhecimento Ótico de Caracteres (OCR), Extração de Padrões ...

Contents

List of Figures

List of Tables

List of Listings

1

Introduction

In the realm of motorsports, Formula 1 racing stands as one of the most captivating and globally watched events. Each year, millions of fans eagerly tune in to witness thrilling races that unfold on the world's most prestigious tracks. The broadcast of Formula 1 races encapsulates a diverse set of events, ranging from high-stakes overtakes and strategic pit stops to dramatic crashes and intense on-track battles.

The content shared in video format has become an integral part of modern communication, spanning from entertainment and news broadcasting to educational and business presentations. The ability to sift through vast amounts of video footage and pinpoint key events not only enhances the viewer's experience but also offers practical applications in sectors such as journalism and sports analysis.

The challenge at hand lies in the development of a sophisticated program capable of autonomously identifying and extracting the most enthralling and pivotal events from live Formula 1 race broadcasts. The goal is to elevate the viewing experience for avid Formula 1 enthusiasts but also offer invaluable data insights to analysis. The final ambition is to extend this project to various types of motorsports broadcasts, for now, this thesis focuses primarily on a specific domain: Formula 1 racing.

The exploration of event extraction in sports broadcasting exposes a multifaceted body of research and technologies deployed across various disciplines. Numerous studies have delved into the intricacies of event identification and extraction, shedding light on the challenges and opportunities inherent in this field. In-depth examinations of

methodologies and technologies employed before, provide a foundation for understanding the state of event extraction in the context of Formula 1 racing broadcasts.

The contributions of this thesis focus on the simplicity of the methodologies used by adopting a more straightforward methodology, ensuring the expected results without the need of more sophisticated approaches. The intention behind this approach is to achieve the expected results effectively and efficiently. Given that the project revolves around broadcasting races, one of the objectives is to minimize the execution time to the greatest extent possible, achieving meaningful outcomes.

To address the goals of this research, this thesis begins with an exploration of related work in video event classification, establishing the foundation upon which the program's development is based in Chapter 2.

In Chapter 3, the background of frame comparison methods, template matching techniques, and text recognition is discussed, providing the necessary theoretical support for the program's implementation. The mentioned frame comparison methods include the comprehension of the Euclidean Distance, Histogram Comparison, Edge Change Ratio and Peak Signal-to-Noise-Ratio in this context.

In Chapter 4, it's detailed the program's design, including shot detection and events extraction methodologies. It starts by determining the most suitable frame comparison method for the proposed approach. Additionally, it explores three distinct methodologies for event extraction: template matching, color detection, and text recognition. Within this context, it is also discussed how these events are extracted.

Following this, in Chapter 5, empirical results obtained from testing across various races are presented.

In Chapter 7, potential directions for refinement and expansion are outlined for future work.

Ultimately, in Chapter 6, the thesis synthesizes the key contributions of the research and discusses their implications. Through this structured approach, the thesis aims to offer a comprehensive understanding of the program's development and its potential impact on video event extraction in broadcasting.

This project constituted the master's thesis in Engenharia Informática e Multimédia at ISEL, developed in collaboration with the company Six Floor Solutions.

2

Related Work

This chapter presents a review of the state-of-the-art techniques for video events classification based on template matching, machine learning, and deep neural network-based approaches. Furthermore, some of the recent works bases on Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, will be presented to provide further insight into this field.

2.1 Video Event Classifier

A video event classifier is a system designed to analyze and classify events occurring in videos. It's a subset of video understanding tasks within the field of computer vision. The primary goal of a video event classifier is to automatically recognize and categorize different types of events, actions, or activities within video sequences.

This type of projects involves the categorization of videos based on their content, particularly focusing on identifying and labeling specific actions or activities occurring within the video data. This technology finds application in various fields such as surveillance, content recommendation, video summarization and event recognition. However, it comes with challenges stemming from the temporal nature of videos, variations in lighting conditions, camera angles, and the complexity of human actions. These challenges require the use of sophisticated approaches to obtain accurate classification results.

2.1.1 Frame-based Approaches

In the field of video event classification, different approaches are employed to tackle the complexities associated with temporal sequences and diverse visual conditions. Frame-based approaches focus on the analysis of individual DataFrames within the video. 3D Convolutional Neural Networks (CNNs) extend traditional 2D CNNs into the temporal dimension, enabling them to capture motion patterns across DataFrames.

First, it's important understand how Convolutional Neural Networks (CNNs) work. These networks are well-suited for working with images or groups of images as their input data. They achieve this by subjecting the input to a series of convolutional layers that extract crucial features. To perform classification, the downsampled features vector traverses the final layer of the neural network, generating a probability distribution indicating its affiliation with a particular class.

Deep Learning's integration into video event classification empowers the extraction of robust and unique feature representations. An interesting approach was the creation of a dataset with diverse videos encompassing crowd events [1] such as Marriage, Cricket, Jallikkattu, and Shopping mall scenarios. This system leverages two Deep CNN architectures, namely the baseline model and VGG16, both designed to identify predefined events and provide temporal context. Through automated testing of input video frames, the CNN model estimates central events within the video. By extracting event features from these input frames, the CNN accurately labels the event type.

Considering sports events, there are some advancements that consider 3D CNNs over 2D CNNs, indicating marginal enhancements in video classification. 3D CNNs excel in capturing the temporal information due to their capacity to convolve multiple images simultaneously, mirroring the sequential frames of a video. This approach was applied to football video data [12], where several CNN models were tested, including VGG16, VGG19, InceptionV3, MobileNet, and ResNet50. Among these models, the latter two demonstrated superior performance. Here, the goal was to explore a model with similar architecture of the ResNet50 CNN, alongside two 3D ResNet architectures tailored for video classification: R3D_18 and MC3_18.

Another interesting utilization of the CNNs was to correct illumination [18], training a model to study the illumination variation at different moments in a day at the scene. The outcome of the CNN model would provide an estimate of the light intensity within a provided image, which can then be employed to adjust the image's light intensity. When integrated with the YOLO darknet-53 framework [19], the CNN can be fused to create an end-to-end detection network model. This model not only transforms input images into color constancy images but also facilitates semantic recognition.

There are multiple interesting methods, using CNNs, to improve the detection of events automatically, but the focus of this thesis is making all the process much simpler.

2.1.2 Sequence-based Approaches

Sequence-based approaches, on the other hand, address the temporal aspect by treating video DataFrames as a sequential data. Recurrent Neural Networks (RNNs), such as Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) networks, are capable of capturing long-term dependencies in sequences, which is crucial for recognizing events spanning multiple DataFrames. Furthermore, the adaptation of Transformer-based models from the field of natural language processing to video data provides a means to capture relationships between distant DataFrames.

2.1.3 Hybrid Approaches

Hybrid approaches aim to integrate the strengths of multiple techniques. Combining 3D CNNs with LSTMs creates a powerful model that captures both spatial and temporal features effectively, making it well-suited for events involving dynamic interactions. The incorporation of attention mechanisms into two-stream networks enhances their capability to focus on essential spatial and temporal components, thus improving recognition accuracy.

2.2 Video Summarization

Video summarization has become a focal point for researchers due to its ability to condense videos into concise and informative versions, thereby aiding users and systems in saving time and effort when searching for and comprehending desired content [20]. Current techniques employ various approaches to determine which segments of a video should be included in the summarized version. The main challenge lies in processing the diverse data types present in videos to identify relevance cues, such as redundancy or complementary information, which assist in making informed decisions. One recent strategy involves utilizing subjectivity detection, where the presence or absence of subjectivity is leveraged as a relevant clue to align video summaries more closely with the expectations of the end users.

In [20], it's introduced an approach that aimed to develop a versatile subjectivity classification model for sentences, capable of handling multiple languages and domains.

This method proves valuable in applying subjectivity criteria to video summarization tasks. Through this proposed method, a subjectivity classification model can be constructed and utilized, irrespective of the language or domain of the target sentences. Furthermore, this approach can enhance subjectivity-driven video summarization. It empowers summarization efforts in domains where subjective content holds significance, like in movie reviews, as well as in domains where the primary content is objective, such as in the realm of news reporting.

In [21], was implemented a template matching algorithm that can be flexibly adjusted to accommodate both shrinking and stretching, enabling the search for periodicity and the extraction of patterns based on human movement understanding. The summarization process relies on validating these extracted patterns using a search-through subsystem based on correlation. Human actions often exhibit repetitive patterns over time, suggesting that actions can be considered as having a periodic nature. Periodicity can be defined as the recurring segments or elements within an action, which constitute the fundamental and most essential components of that action. These recurring segments represent the core and smallest significant units of the action.Binary Large OBject Blobs (Binary Large OBjects) tracked along the way construct sequence of interest regions. Patterns acquired by applying HoG to the interest blobs that are sequenced in time dimension for periodicity and continuity analysis. Template matching is employed to locate patterns, while the histogram of oriented gradients (HoG) is applied to blobs in a $3x3$ cell fashion, which is particularly well-suited for HoG (Histogram of Oriented Gradients) [22] [23]. HoG is a feature descriptor, similar to the Canny Edge Detector and SIFT (Scale Invariant Feature Transform). It is widely used in computer vision and image processing, particularly for the task of object detection. HOG operates by quantifying the occurrences of gradient orientations within localized regions or blocks of an image. This technique plays a crucial role in identifying and describing distinctive visual characteristics in images, making it valuable for effective object detection and recognition. These histograms serve as the input features upon which the correlation and template matching are based. The periodicity of the HoG descriptors is explored in the temporal dimension, and patterns that are discovered are extracted to construct key frames for video summarization.

Video summarization typically begins with two fundamental steps: shot boundary detection and key frame extraction [6]. Shot boundary detection is the process of segmenting the video into individual shots, while from each shot, a key frame is selected to capture maximum information about that particular shot. These key frames are selected based on the analysis of Higher Order Color Moments (VSUHCM), which include higher-order statistics such as image Histogram, skewness, and kurtosis. This

selection process helps transform a lengthy and potentially monotonous video into a shorter and more engaging one. Video summarization techniques span various domains, including movies, sports, news, home videos, e-learning, and more. These techniques can be categorized into different approaches such as object-based summaries, event-based summaries, content-based summaries, feature-based summaries, among others. The comparison of various methods, including VSUHCM, DT, STIMO, VSUMM1, VSUMM2, and OV summaries, is performed based on the analysis of color distribution to evaluate their effectiveness in generating video summaries. Video summarization is a technique that produces a concise representation, offering an abstract overview of the original video sequence. Such summaries are valuable for video browsing and retrieval systems. Various methods are employed to select key frames within this process. These methods primarily rely on low-level features like color histograms, edge histograms, and frame correlation. However, it's important to note that these methods do not take into account the specific colors present in the images when making their selections. Some of the methods presented are: Delaunay Triangulation [24], K-means clustering algorithm [2], STIMO (Still and Moving) [3], Edge Change Ratio [4] and Vscan [5]. These approaches are detailed in the paper [6].

In [7], it's introduced a methodology for creating cricket sports highlights through the application of optical character recognition (OCR) techniques. Initially, it involves extracting the score bar from individual frames, followed by the utilization of character recognition methods to extract data regarding notable events like sixes, fours, and wickets. Subsequently, a brief video summary is generated, encompassing frames featuring these significant events, referred to as "Highlights". A library of possible image of the score was developed by extracting the score from different templates of the score bar of cricket videos. There are two methods present in the paper to match the score from the video and the library: Brute Force and Template Matching. The tempalte matching method is implemented by computing the normalized cross-correlation between the connected objects and a library of reference images. This method allows for the efficient extraction of the score with fewer iterations, resulting in reduced processing time.

3

Background

This chapter serves as the foundational background for the research project, outlining the essential methods and techniques employed. It covers a range of frame comparison methods, including Euclidean Distance, Histogram Comparison, Edge Change Ratio (ECR) with Canny Edge Detection, and Peak Signal-to-Noise Ratio (PSNR), as well as decision-making methods. Additionally, it introduces template matching and discusses the Tesseract OCR for extracting textual information from video frames. This chapter provides a comprehensive understanding of the technical tools and methodologies used throughout the project to enhance the Formula 1 viewing experience and generate valuable analytical insights from broadcast races.

3.1 Frame Comparison Methods

In this section some methods that were implemented to compare two frames will be discussed, which was needed to detect shots. The detection of shots in a video refers to the process of identifying and segmenting the video into individual shots or scenes. A "shot" in the context of video analysis is a continuous sequence of frames captured by a camera without any interruption.

First, the concept of "shot detection" will be clarified. It refers to the automated identification of shot transitions in digital videos, involving the recognition of transitions between frames. The goal is to achieve temporal segmentation of the video content by

comparing two consecutive frames and determining whether there is a significant difference between them. In this context, a scene change is likely to be detected by directly analyzing the images or by utilizing extracted features like histograms or contours.

3.1.1 Euclidean Distance

Euclidean distance, also known as straight-line distance or L2 distance, is a fundamental concept in geometry and mathematics that measures the length of the shortest path between two points in a multidimensional space. It is commonly used to quantify the similarity or dissimilarity between two vectors in Euclidean space.

Euclidean distance can also be applied to compare two frames in the context of image or video analysis. When comparing frames, each frame is treated as a multidimensional vector, where each pixel or feature in the frame contributes to a dimension in the vector. The Euclidean distance between these vectors then measures the similarity or dissimilarity between the frames.

In this case, each frame is represented as a vector by flattening its pixel values. For grayscale images, this results in a one-dimensional vector. For color images, each pixel has multiple color channels, so the vector would have the dimension of 3.

Given two matrixes (frames) $F1$ and $F2$, the distance is calculated as follows,

$$d(F1, F2) = \sqrt{\sum_{I} (F2(I) - F1(I))^2}$$

3.1.2 Histogram Comparison

Histogram comparison is a method used to measure the similarity or dissimilarity between two histograms, which are representations of the distribution of values in a dataset [8]. Histograms provide a way to visualize the frequency of different values or ranges of values within a dataset. Histogram comparison is often used in image processing, computer vision, and data analysis to assess the similarity between two datasets or to identify patterns or changes in data distribution.

There are several techniques for comparing histograms, each with its own advantages and limitations:

- **Correlation**: Measures the correlation between histograms, which can indicate how well they match. A higher correlation implies a stronger similarity.

Given two histograms $H1$ and $H2$, this can be achived by,

$$d(H1,H2) = \frac{\sum_I (H1(I) - \overline{H1})(H2(I) - \overline{H2})}{\sqrt{\sum_I (H1(I) - \overline{H1})^2 \sum_I (H2(I) - \overline{H2})^2}}$$

where,

$$\overline{H_k} = \frac{1}{N} \sum_J H_k(J)$$

and N is the total number of histogram bins.

- **Chi-Square Test**: The chi-square test measures the difference between observed and expected values in the histograms. It's commonly used to assess the independence of variables, but it can also be applied to histogram comparison.

  Given two histograms $H1$ and $H2$, this can be achived by,

$$d(H1,H2) = \sum_I \frac{(H1(I) - H2(I))^2}{H1(I)}$$

- **Histogram Intersection**: This method calculates the minimum value of each bin from the two histograms being compared. It can be used to measure the similarity between histograms. Computes the intersection between histograms, highlighting overlapping areas that indicate similarity.

  Given two histograms $H1$ and $H2$, this can be achived by,

$$d(H1,H2) = \sum_I \min(H1(I), H2(I))$$

- **Bhattacharyya Distance**: Bhattacharyya distance measures the overlap between two probability distributions. It's often used in image processing to compare color distributions in images.

- **Earth Mover's Distance (EMD)**: EMD, also known as Wasserstein distance, calculates the minimum "cost" required to transform one histogram into another. It's particularly useful for comparing histograms in a way that considers the spatial relationships of the data.

- **Kullback-Leibler Divergence**: KL divergence quantifies the difference between two probability distributions. It's often used to compare how much one distribution differs from another.

3.1.3 Edge Change Ratio (ECR)

The edge change ratio (ECR) is adopted to measure the dissimilarity between different video frames [9] [10]. The maximum value of ECR inside a temporal sliding window is compared with a given threshold to determine a potential shot transition. Saying that, a higher ECR value suggests a more significant change between frames.

It takes the edge information extracted from the frames using edge detection methods like Canny. By comparing the number of edge pixels that have changed between the frames with the total number of edge pixels in each frame, the ECR captures the relative magnitude of change in edge structure.

The Canny edge detection algorithm is used as a means to extract the edge information that is crucial for calculating the ECR. The ECR considers the change in edge pixels between frames to determine whether a scene transition has occurred. A significant increase in the number of edge pixels (edges) in one frame compared to the other can indicate a transition.

In summary, the Canny edge detection algorithm is employed to obtain edge information, and this information is then used within the ECR calculation to measure the change or dissimilarity between frames.

3.1.3.1 Canny Edge Detection

This multi-stage algorithm involves several steps for enhancing the accuracy of edge detection. Here the implementation of this method in *OpenCV* will be presented [11].

The initial phase focuses on noise reduction through a $5x5$ Gaussian filter to create a smoother image. Subsequently, the intensity gradient of the smoothed image is computed using Sobel kernels in both horizontal and vertical directions, yielding the horizontal gradient (G_x) and vertical gradient (G_y). The gradient magnitude and direction are then derived, with the gradient direction indicating the orientation of edges.

The following stage, known as non-maximum suppression, entails scanning the image to identify potential edges by checking if each pixel is a local maximum within its gradient direction. Any pixels that do not qualify as local maxima are suppressed, resulting in a binary image with thinner, more refined edges.

The last phase of the algorithm, hysteresis thresholding, determines definitive edge candidates. This requires two threshold values, the minimum value and the maximum value. Pixels with gradient intensities surpassing the maximum value are confidently classified as edges, while those with intensities lower than the minimum value are

considered non-edges and discarded. Intermediate pixels between these thresholds are evaluated based on their connectivity to established edge pixels. If they are linked to these "sure-edge" pixels, they are retained as valid edges. This step also removes minor pixel noise by assuming edges to be continuous lines. The outcome is a set of well-defined, prominent edges in the image.

Figure 3.1: Original image.

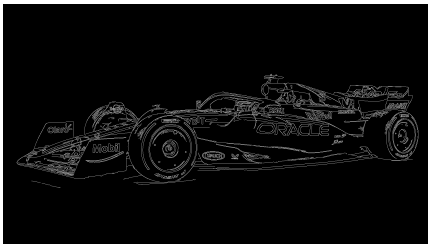

Figure 3.2: Result of the Canny filter.

3.1.4 Peak Signal-to-Noise Ratio (PSNR)

The Peak Signal-to-Noise Ratio (PSNR) is a metric in image and video quality assessment [13]. It serves as a tool to measure the quality of an image that has undergone compression or reconstruction compared to the original, unaltered image. The central idea behind PSNR is to quantify the difference between the reference image (the original) and the processed image (the compressed or reconstructed version) in terms of signal-to-noise ratio. This method can also be employed to detect a scene transition by subtracting two frames and calculating their PSNR. Lower the value returned, the most distinct will the frames be, so a higher PSNR value suggests a higher fidelity between the processed image and the reference image.

In this context, "signal" refers to the original, unchanged image that should be retained and preserved. It embodies the essential information and features of the image. On the other hand, "noise" encompasses the distortions or errors that might arise during compression or reconstruction processes.

$$PSNR = 10 \cdot \log_{10} \left(\frac{\mathrm{MAX}^2}{MSE} \right)$$

where MAX is the maximum possible pixel value (e.g., 255 for an 8-bit image).

3.1.4.1 Mean Squared Error (MSE)

The calculation of PSNR relies on the concept of Mean Squared Error (MSE). The MSE is computed by determining the average of the squared differences between corresponding pixels in the original and processed images. Mathematically, the MSE is the sum of the squared differences between pixel values, divided by the total number of pixels. This operation measures the extent of distortion or divergence from the reference image.

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(I_{\text{original}}(i) - I_{\text{compressed}}(i))^2$$

where N is the total number of pixels, $I_{\text{original}}(i)$ is the value of the ith pixel in the original image, and $I_{\text{compressed}}(i)$ is the corresponding value in the compressed image.

3.1.5 Decision

The previous measurements generate a measure of similarity or difference between consecutive images, and it is necessary to define when in fact if it is considered a shot.

One of the methods include the utilization of thresholds, fixed or adaptive, or the application of machine learning techniques. Each approach offers distinct advantages and is suited to different types of video content. Below, the following methods will be presented in more detail:

- **Fixed Threshold:** This method relies on comparing the computed scores against a predetermined threshold value. When a score surpasses this predefined threshold, a transition point is marked, indicating a cut between shots. This approach is straightforward and efficient for situations where the characteristics of the video content remain relatively consistent.
- **Adaptive Threshold:** In contrast to the fixed threshold approach, the adaptive threshold technique influences the variations in scores across the entire video. By dynamically adjusting the threshold based on these variations, it tailors the cut detection process to the specific properties of the video at hand. This adaptability makes it particularly well-suited for videos with dynamic content or diverse visual elements.
- **Machine Learning:** Incorporating machine learning techniques adds a layer of complexity and sophistication to the decision process. Machine learning models

can be trained on a dataset of videos with known shot transitions, enabling them to learn patterns and features that indicate a change in shots. These models can then be applied to new videos to accurately predict and mark shot boundaries. While this approach requires training data and computational resources, it often delivers enhanced accuracy and the ability to handle more complex scenarios, such as gradual transitions or unconventional shot changes.

Here are a few machine learning techniques commonly used in shot detection and related tasks:

- **Support Vector Machines (SVM):** SVM is a supervised learning algorithm that works well for binary classification tasks like shot detection. In this context, SVM can learn to distinguish between shot transition frames and non-transition frames based on extracted features.

- **Convolutional Neural Networks (CNN):** CNNs are deep learning models that stand out in image and video analysis tasks. They can learn hierarchical features from raw pixel data, making them suitable for shot detection by identifying patterns and transitions in visual content.

- **Recurrent Neural Networks (RNN):** RNNs are well-suited for tasks involving sequential data. For shot detection, they can be used to model the temporal dependencies between frames and learn to predict shot transitions based on the sequence of frames.

- **Long Short-Term Memory (LSTM):** LSTMs are a type of RNN designed to handle longer sequences by mitigating the vanishing gradient problem. They are particularly useful for capturing context over extended periods, making them suitable for shot boundary detection where transitions might span multiple frames.

- **Random Forest:** Random Forest is an ensemble learning method that combines multiple decision trees to make predictions. It can be applied to shot detection by training on features extracted from video frames and making predictions on whether a frame corresponds to a shot transition.

- **Gradient Boosting:** Gradient Boosting is another ensemble technique that builds a strong predictive model by combining the outputs of multiple weak learners. XGBoost and LightGBM are popular implementations of gradient boosting that can be used for shot detection tasks.

- **Neural Architecture Search (NAS):** NAS involves using machine learning algorithms to search for optimal neural network architectures for a given task. In shot detection, NAS can help design customized network structures that capture the nuances of shot transitions.

- **Siamese Networks:** Siamese networks are designed to compare similarity between inputs. They can be used for shot detection by learning to distinguish between frames belonging to the same shot and frames from different shots.

- **One-Class SVM:** Unlike traditional SVM, a one-class SVM is trained on a single class (in this case, non-transition frames) and is used to identify deviations from this class, which might indicate shot transitions.

- **Unsupervised Learning (Clustering):** Techniques like k-means clustering or hierarchical clustering can be used to group similar frames together, potentially helping to identify shot boundaries.

These techniques can be employed individually or in combination, depending on the complexity of the shot detection task and the available data.

In summary, the choice of the method involves choosing the most appropriate approach based on the specific characteristics of the video, the desired level of accuracy, and the computational resources available. Whether opting for the simplicity of fixed thresholds, the adaptability of adaptive thresholds, or the advanced capabilities of machine learning, the ultimate aim remains consistent: to accurately and efficiently detect shot transitions within digital videos.

3.2 Template Matching

Template Matching is a technique employed to locate a template image within a larger image. In *OpenCV*, this task can be accomplished using the $matchTemplate()$ function [14]. This function essentially slides the template image across the input image, similar to a 2D convolution, and assesses how well the template matches each patch of the input image. *OpenCV* offers various comparison methods to facilitate this process. The outcome is a grayscale image in which each pixel represents the degree of similarity between the neighborhood of that pixel and the template. This is an interesting technique that was employed to detect certain elements within video frames. Note that this process can also be applied with edge-detected versions of the images. This can

be tested to achieve greater confidence when the template is present within the video DataFrame with varying levels of opacity and color.

The results observed in the Figures 3.4 to 3.6 were obtained obtaining using the *matchTemplate*() and *minMaxLoc*() functions, the last, identifies the location of the maximum or minimum value. This location serves as the top-left corner of the matching region, with (w, h) denoting the width and height of the identified rectangle. The red rectangle (Figure 3.6) precisely delineates the region where the template is located within the source image.

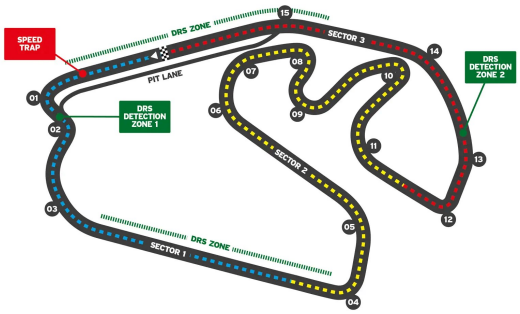

Figure 3.3: Source Image.

Figure 3.4: Template Image of the Speed Trap.

Figure 3.5: Matching Result.

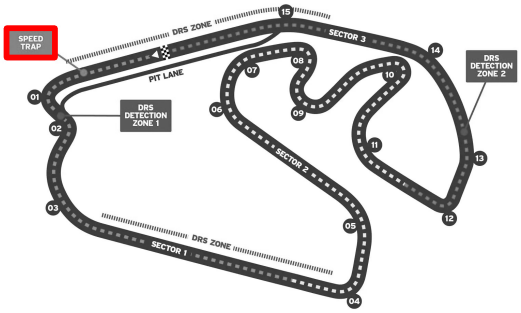

Figure 3.6: Detected Point with Red Rectangle.

In summary, the process of template matching starts by comparing a source image (larger image) with a template image (smaller image) as input and calculates the similarity between the template and different regions of the source image using a specified method, and stores the comparison results in a result (for example, Figure 3.5). After the function finishes the comparison, the best matches can be found as global minimums (TM_SQDIFF is normally applied to achieve this purpose) or maximums (TM_CCORR or TM_CCOEFF), and this can be achieved by utilizing the *minMaxLoc*() function.

In case of a color image, template summation in the numerator and each sum in the denominator is done over all of the channels and separate mean values are used for each channel. That is, the function can take a color template and a color image. The result will still be a single-channel image, a format that is notably easier to analyze and interpret.

Here are some common comparison methods available in *OpenCV*:

- "TM_CCOEFF" (Cross-Correlation Coefficient): This method calculates the cross-correlation coefficient between the template and the region of interest in the input image. It is not normalized, which means it does not scale the result to a specific range.
- "TM_CCOEFF_NORMED" (Normalized Cross-Correlation Coefficient): This method calculates the normalized cross-correlation coefficient between the template and the region of interest in the input image. It scales the result to a range of -1 to 1, where 1 indicates a perfect match and -1 indicates a perfect negative match.
- "TM_CCORR": This method calculates the cross-correlation between a template and a region of interest in a source image. Is suitable for considering variations in image intensity or brightness. It is not as sensitive to changes in lighting conditions as other methods.
- "Some Other Methods": *OpenCV* provides several other methods for template matching, such as "TM_SQDIFF" and "TM_SQDIFF_NORMED", which use squared differences instead of cross-correlation.

The methods that use cross-correlation are robust to changes in image intensity and are often a good choice when the template and the object have different lighting conditions. For the example above (Figures 3.4 to 3.6), the method used was the "TM_CCOEFF_NORMED".

3.3 Tesseract OCR

Tesseract OCR (Optical Character Recognition) is an open-source software developed primarily for recognizing and extracting text from images and scanned documents [25] [15]. It is designed to convert printed or handwritten text within images into machine-readable text that can be processed and edited.

The Tesseract OCR uses advanced algorithms to analyze the shapes and patterns of characters in the input image, making it capable of recognizing a wide range of fonts, languages, and writing styles. It has been widely used in various applications, including document digitization, automated data entry, and text extraction from images for tasks like document indexing, translation, and text-to-speech conversion.

When working with a Tesseract OCR instance, various options and settings can be specified to customize the behavior and enhance the accuracy of text recognition.

Here are some of the parameters that can be specified on a Tesseract instance:

- **Language:** Tesseract supports a wide range of languages, and accepts one or multiple languages as the target.
- **Page Segmentation Mode:** Tesseract uses different page segmentation modes to identify text regions within an image. The segmentation mode can be specified based on the layout of the text, such as single column, single block, or a combination of both.
- **OCR Engine Mode:** Tesseract offers different OCR engine modes, including "OEM_TESSERACT_ONLY", "OEM_LSTM_ONLY", and "OEM_DEFAULT", which determine the underlying recognition engine used. The default mode usually works well for most scenarios.
- **Character Whitelist and Blacklist:** A whitelist of characters can be specified for consideration during Tesseract recognition, along with a blacklist of characters for exclusion. This proves useful when the expected character set is known in advance.
- **Image Preprocessing:** Various image preprocessing techniques, including resizing, binarization, noise reduction, and deskewing, can be applied in Tesseract to enhance the quality of input images.
- **Dictionary and User Patterns:** Custom dictionaries and user-defined character patterns can be provided to enhance recognition accuracy for domain-specific terms or symbols in Tesseract.
- **Output Format:** The format for recognized text output can be specified in Tesseract. Options encompass plain text, hOCR (HTML), PDF, and additional formats.
- **Confidence Threshold:** Tesseract assigns confidence scores to recognized text. A confidence threshold can be set to filter out recognition results with low confidence in Tesseract.

- **Debugging and Logging:** Tesseract provides options to enable debugging and log output, which can be helpful for diagnosing recognition issues.
- **Custom Configurations:** Custom configuration files, known as Tesseract config files, can be created to store and reuse specific settings for various OCR tasks.
- **Page Segmentation Analysis:** The page segmentation analysis mode can be specified in Tesseract to control its analysis of page layout and the order in which it processes text.
- **Text Position and Orientation:** Tesseract can detect and provide information about the position, orientation, and bounding boxes of recognized text.

The Page Segmentation Modes allows to instruct the Tesseract on how to analyze the layout and structure of text within an image, which can significantly impact recognition accuracy.

Here are some examples of Page Segmentation Modes that can be used in Tesseract OCR:

- "PSM_AUTO": Tesseract automatically determines the page segmentation mode based on the content and layout of the image. This is the default mode.
- "PSM_SINGLE_BLOCK": Treats the entire image as a single text block. Useful for images with a single column of text.
- "PSM_SINGLE_COLUMN": Detects and processes text as a single column, even if there are multiple columns in the image.
- "PSM_SINGLE_LINE": Analyzes the image as a single line of text. Useful for reading vertically aligned text.
- "PSM_SINGLE_WORD": Recognizes individual words in the image. Useful when it's needed to extract words separately.
- "PSM_SINGLE_CHAR": Processes the image character by character. Useful for character-level recognition.
- "PSM_SPARSE_TEXT": Assumes that the image contains sparse text, such as handwriting. It attempts to find text in different orientations and sizes.
- "PSM_RAW_LINE": Considers the image as a single text line without considering the layout. Useful for fixed-width text.

- "PSM_COUNT": This is not a segmentation mode but is used to count the number of available PSM modes.

The choice of settings depends on the specific requirements of the text recognition task and the characteristics of the input images.

Another interesting topic includes improving the accuracy and performance of Tesseract OCR. This involves a combination of steps. Image preprocessing is a crucial initial phase. It encompasses techniques like noise reduction, which cleans up the image by applying filters such as Gaussian or Median. Binarization is another essential step, converting the image into a binary format to enhance text contrast. Correcting any skew or rotation through deskewing ensures that the text appears horizontally, aiding recognition. In addition to preprocessing, it's essential to consider the resolution and size of the input image. Adequate resolution, typically at least 300 DPI, can greatly impact character recognition. Resizing the image to an appropriate size is essential, as overly large or small images can hinder recognition accuracy.

Now, an example will be presented on how to improve the text recognition on a given image (Figure 3.7).

Figure 3.7: Original Image.

In the case of the image shown above, it needs to be inverted because the Tesseract OCR generally works better when the text has a light background (white) and the letters are in a light dark (black). This high-contrast configuration helps Tesseract to more accurately identify and segment the text from the background, leading to improved recognition results. This is a process included on the preprocessing phase.

Improving OCR text recognition can be achieved also through several specifications besides the preprocessing steps. One effective approach involves setting a **Whitelist** that includes all the possible letters appearing in the sentence. Additionally, adjusting the **Page Segmentation Mode** to one that matches the layout of the text in the image can enhance recognition accuracy (for example, "PSM_SINGLE_BLOCK", "PSM_SINGLE_COLUMN", "PSM_RAW_LINE").

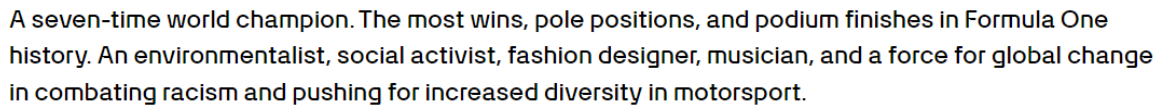

Figure 3.8: Binary Image.

The output text is "A seven-time world champion. The most wins, pole positions, and podium finishes in Formula 1 history. An environmentalist, social activist, fashion designer, musician, and a force for global change in combating racism and pushing for increased diversity in motorsport.".

In summary, these are a few steps to consider to improve the Tesseract recognition:

- **Image Quality Assessment:** Begin by evaluating the quality of the input image. Ensure it is clear, well-focused, and high-resolution. Poor image quality can significantly affect OCR accuracy.
- **Contrast Adjustment:** Increase contrast to make text stand out from the background.
- **Brightness Adjustment:** Optimize brightness for better readability.
- **Noise Reduction:** Apply filters or algorithms to remove noise, speckles, or artifacts.
- **De-skewing:** Correct any image rotation or skew to make text lines horizontal.
- **Binarization:** Convert the image to binary format, where pixels are either black or white. Binarization can be done using various techniques, including thresholding. Ensure that text remains distinct from the background. Remembering that the text should be black and the background should be white.
- **Resizing:** Resize the image to an appropriate resolution. Images that are too large can slow down processing, while very small images may lead to reduced accuracy.
- **Region of Interest (ROI) Extraction:** If the image contains multiple regions of text, extract relevant regions of interest (ROIs). This reduces processing time and improves accuracy.
- **Denoising:** Further denoise the image if needed, especially in cases where OCR performance is impacted by remaining artifacts or speckles.

- **Page Segmentation:** If the document has complex layouts, headers, or footers, specify the page segmentation mode to guide Tesseract in identifying text regions correctly.

4

Approach

In this chapter, the methods and decisions that led to the implemented solutions will be discussed. This chapter offers insights into the decision-making process, highlighting the practical steps taken to address the issues at hand.

Initially, the focus is on shot detection methodologies, which include techniques like Histogram Comparison, Euclidean Distance, Edge Change Ratio (ECR), and Peak Signal-to-Noise Ratio (PSNR). This is followed by a discussion of the results obtained through these methods. Subsequently, attention shifts to event detection, where the emphasis lies in defining events, their associated elements, delineating search areas, and employing techniques like template matching, color detection, and text recognition for event identification. Lastly, the objective is to elucidate the output of detected events, providing an understanding of how they are represented, particularly in the form of JSON objects.

4.1 Shot Detection

The first phase of the development of the program relied on the shot detection. To achieve this goal, were implemented the four methods described before (Euclidean Distance, Histogram Comparison, Edge Change Ratio (ECR) and eak Signal-to-Noise Ratio (PSNR)) (see Section 3.1).

This feature was developed thinking in broadcasting scenarios, for users who may

have missed specific events and want to review them, positioning the video at the point where the event occurred (first shot before the event), enhancing the viewer experience.

4.1.1 Euclidean Distance

The Euclidean Distance (see Section 3.1.1) here refers to the measurement of visual dissimilarity between two consecutive frames. By quantifying the pixel-level differences using this mathematical metric, it becomes possible to identify transitions or significant alterations in a video sequence, facilitating the automatic detection of shot transitions.

The implementation of this problem consists of a function that calculates the Euclidean distance between grayscale versions of two consecutive frames. The frames are converted from BGR color space to grayscale using specific conversions. The Euclidean distance is then computed using the "norm" function ([17]), representing the difference between the pixel values of the two frames. The purpose of this approach is to measure the visual dissimilarity between frames based on pixel intensity, which can help identify potential shot changes in a video sequence.

1. Start
2. Convert frames to grayscale
3. Calculate Euclidean Distance between the grayscale frames
4. End

This distance acts as a numerical measure of the geometric difference between the pixel values. A lower Euclidean distance indicates that the frames are more similar in terms of pixel intensities, while a higher value suggests greater dissimilarity.

4.1.2 Histogram Comparison

Histogram Comparison (see Section 3.1.2) in a video involves analyzing the distribution of color or intensity values across frames to detect changes or patterns. By calculating the differences between histograms of consecutive frames, this technique helps to identify shot transitions.

The implementation of this problem consists of a function that calculates and compares histograms of color distribution in video frames to detect shot changes. The purpose of the function is to compute differences between histograms of consecutive frames in

a video sequence, considering different color spaces (such as grayscale and HSV). The differences are measured using the Manhattan distance formula and are normalized.

The function takes as input two frames (the current frame and the previous frame), in the desired color space for analysis (grayscale or HSV). It processes the frames to convert them into the specified color space and calculates histograms for each color channel. The histograms are used to quantify the distribution of color values in the frames. The differences between the histograms of the current and last frames are computed, considering individual color channels where applicable.

In summary, the code performs histogram-based shot change detection by measuring the differences in color distribution between consecutive video frames, providing insights into potential shot transitions in the video sequence.

Below, there is a flowchart for the Histogram Comparison function.

1. Start
2. Decision: Check Color Space
   - If Color Space is Grayscale
     - Convert frames to grayscale
     - Calculate Histograms for Grayscale frames
   - Else if Color Space is HSV
     - Convert frames to HSV color space
     - Calculate Histograms for HSV frames and channels
   - Else
     - Invalid Color Space
     - End
3. Calculate Absolute Difference Histogram
4. Calculate Normalized Absolute Difference Value
5. End

These differences, whether derived from grayscale or HSV analysis, provide valuable information about potential shot changes or significant alterations in the video sequence. Through this histogram-based approach, the function contributes to video analysis and shot change detection in scenarios where subtle or abrupt shifts in content occur.

4.1.3 Edge Change Ratio (ECR)

The Edge Change Ratio (see Section 3.1.3) serves as a tool in video analysis, quantifying variations in edge features between successive frames. By calculating the extent of these changes, it facilitates the detection of scene transitions within the video sequence. This metric provides valuable insights into the dynamic evolution of visual content.

The implementation of this problem consists of a function that operates by identifying changes in edge patterns, often indicative of shot transitions. It processes two consecutive frames from a video, assessing the extent of change through edge detection and dilation. By comparing these frames' edge patterns, the Edge Change Ratio (ECR) can be calculated. A higher ECR value indicates higher dissimilarity between the frames. In other words, a larger ECR value signifies more significant changes or variations in edge patterns between the two frames being compared.

1. Start
2. Convert frames to grayscale
3. Apply Canny edge detection to the frames
4. Dilate the edged frame using a specified dilate rate
5. Invert pixel values in the dilated frames
6. Calculate ECR Value
7. End

The edges are subjected to dilation, where the expansion level is controlled by a dilate parameter. Dilation aims to enhance and amplify the edges, potentially aiding in the detection of significant changes. Subsequent to dilation, pixel value inversion is applied, turning white pixels into black and vice versa.

These processed images are used to calculate the Edge Change Ratio (ECR). This ratio is a measure of the proportion of changes in edge features between consecutive frames. It helps in quantifying the extent of transition or shift in the video content.

4.1.4 Peak Signal-to-Noise Ratio (PSNR)

Peak Signal-to-Noise Ratio (PSNR) (see Section 3.1.4) is a metric that can be used in video analysis to quantify the similarity or dissimilarity between consecutive frames

by measuring the quality of pixel intensity changes. PSNR helps in identifying content transitions within video sequences.

The implementation of this problem consists of a function, which operates within the realm of video analysis to assess the quality and dissimilarity of successive frames using the Peak Signal-to-Noise Ratio (PSNR) metric. Beginning with grayscale conversions, the code transforms both the current and the previous frame into grayscale representations, isolating luminance values for further analysis. The following steps involve the dilation of these grayscale frames and the process involves expanding the detected edges or features within the images.

The core calculation centers around the PSNR, a metric that quantifies the similarity or difference between two images. By assessing the absolute difference between the grayscale frame and the previous one, the code progresses to compute the Squared Sum of Errors (SSE). This step captures the cumulative differences between pixel intensities.

1. Start
2. Convert frames to grayscale
3. Dilate frames
4. Calculate Absolute Difference between frames
5. Square each element in the Absolute Difference Vector
6. Calculate Sum of Squared Differences in all channels
7. Calculate Mean Squared Error and then PSNR value
8. End

Handling the SSE value, the code includes a conditional statement to check if it is extremely small. If so, the PSNR value is assigned as 100, indicating high similarity between frames. However, in cases where changes exist, the Mean Squared Error (MSE) is computed. This subsequently facilitates the derivation of the PSNR value.

4.1.5 Results and Discussion

In this section, the outcomes of our methods for a single video will be presented and analyzed. The goal is to explore and discuss the results and consider the limitations.

Now, the results of the implementation of the methods will be presented on plots for 593 frames of the specific video showcasing a Formula 1 race.

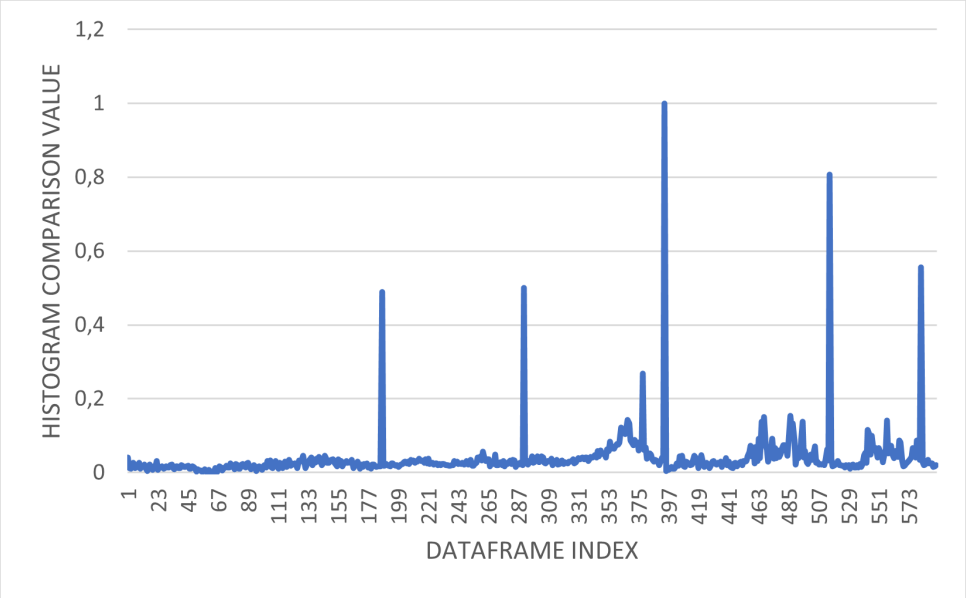

Figure 4.1: Histogram Comparison values for each frame.

The plot shown in Figure 4.1 already shows that the transitions are very distinct from the non-transition frames. The peaks of greater magnitude within the graph serve as an indicator that a shot transition might have occurred.

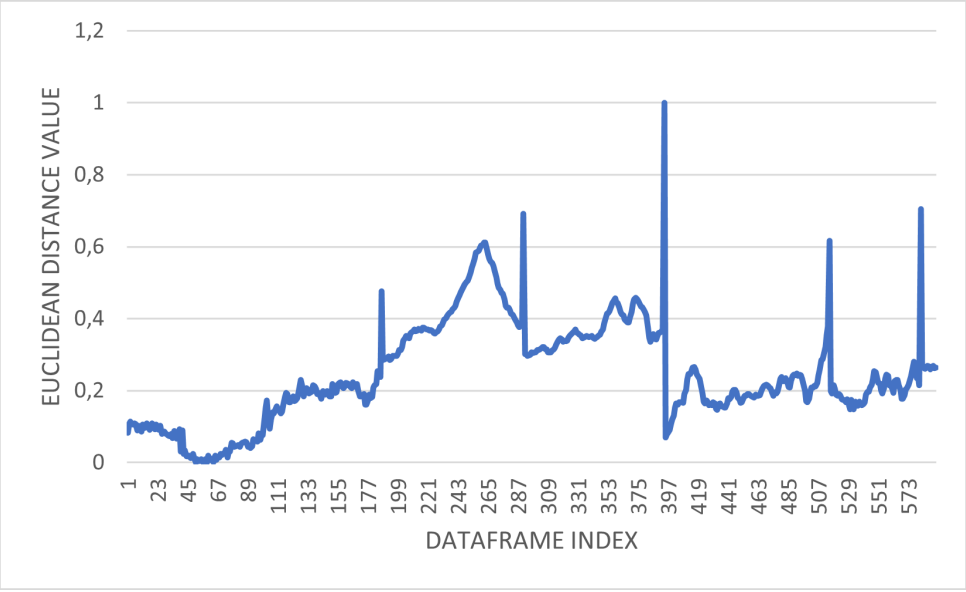

Figure 4.2: Euclidean Distance values for each frame.

The use of the Euclidean Distance method has proven to be significantly less effective (see Figure 4.2), as the peaks lack a distinct contrast from the surrounding data.

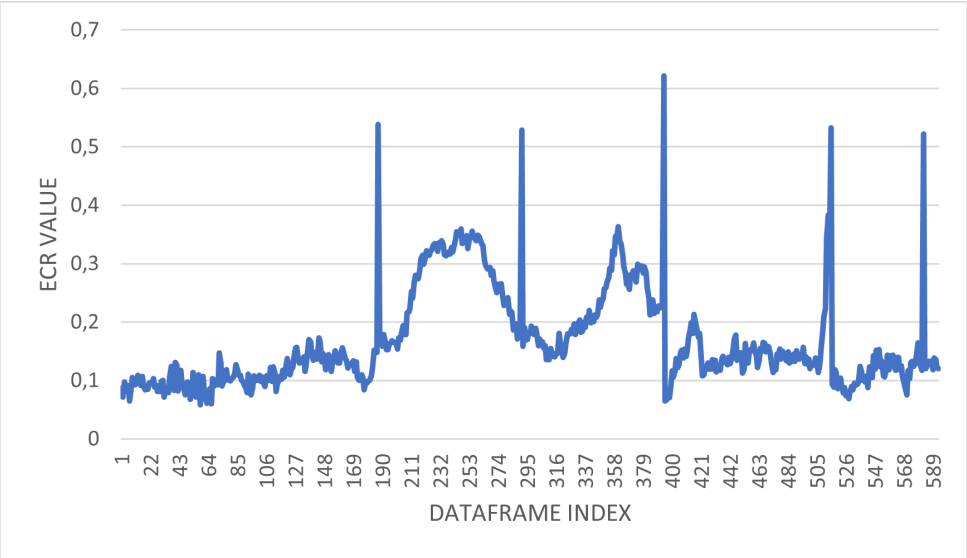

Figure 4.3: Edge Change Ratio values for each frame.

The Edge Change Ratio method (Figure 4.3) has demonstrated its effectiveness, as evidenced by the enhancement in peak visibility.

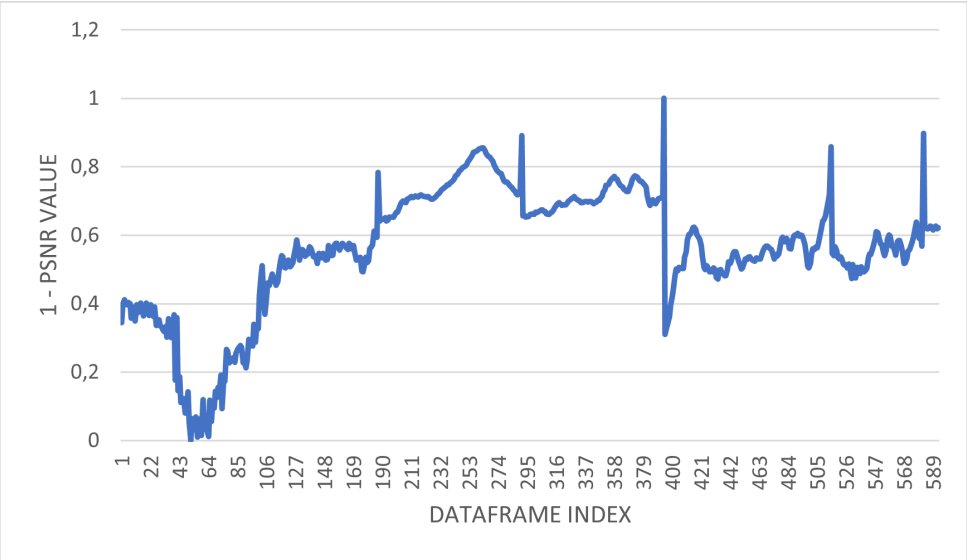

Figure 4.4: Peak Signal-to-Noise Ratio values (positive) for each frame.

After analysing the plot shown in Figure 4.4, a good contrast between the peaks and the rest of the data cannot be found. So it was decided to analyse the graph in other way, subtracting the current PSNR result from the previous one to improve the contrast.

The results are clearer since the peaks are a lot more evident (see Figure 4.5).

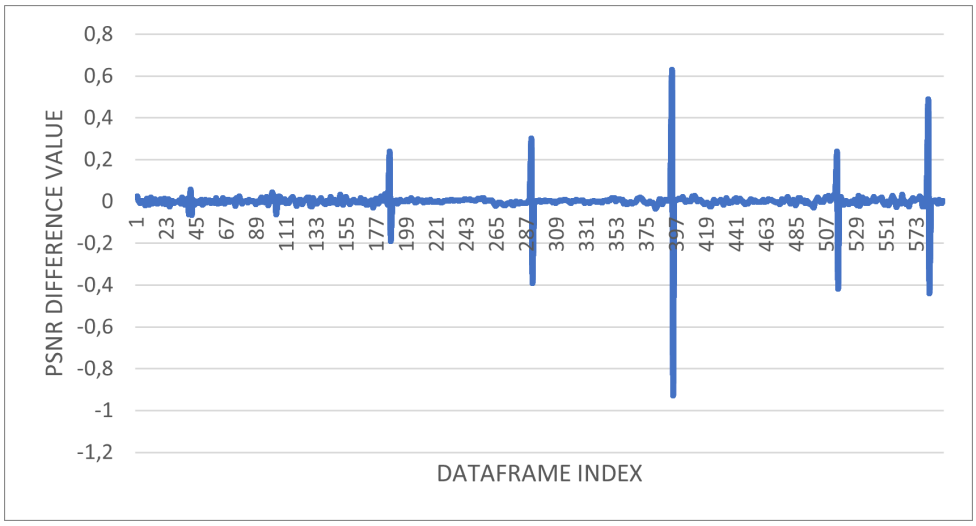

Figure 4.5: Peak Signal-to-Noise Ratio difference values (normalized) for each frame.

The Table 4.2 displays the results of each method for six frames from a given video. This was a simpler test to show a particular range of results. The final decision was determined by the application of the methods to a larger video.

The **Ground Truth** column indicates whether the frames are transition frames of false transitions frames (frames that some methods detected a transition that didn't actually occurred). The **Frame Number** index indicates the index of the frames in the video. The columns on the right under **Frame Comparison Methods**, ECR, PSNR, ED, and HC, correspond to the Edge Change Ratio, Peak Signal-to-Noise Ratio, Euclidean Distance, and Histogram Comparison methods, respectively. In here, a **TP** (True Positive) signifies that the algorithm accurately detected the transition, a **FP** (False Positive) indicates that the algorithm detect a transition that didn't actually occurred, a **TN** (True Negative) indicates that the algorithm didn't detect a transition that didn't happen, and a **FN** (False Negative) indicates that the algorithm didn't detect a transition that happened. The Table 4.1 summarizes this explanation.

	Transition	**Detected**
TP	Yes	Yes
FP	No	Yes
FN	Yes	No
TN	No	No

Table 4.1: Explanation of the TP, FP, FN, TN in this context.

Additionally, the row **Execution Time (seconds)** shows the execution time for each

frame comparison method, and the row **Accuracy** shows the percentage of correct classifications using the following formula:

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

		Frame Comparison Methods			
Ground Truth	**Frame Number**	**ECR**	**PSNR**	**ED**	**HC**
Transition Frame	186	TP	FN	FN	TP
Transition Frame	290	TP	TP	TP	TP
Non-Transition Frame	377	TN	TN	TN	FP
Transition Frame	393	TP	TP	TP	TP
Transition Frame	514	TP	FP	TP	TP
Transition Frame	581	TP	TP	TP	TP
Execution Time (seconds)		13,5	5,5	4,5	5,3
Accuracy		100%	66%	83%	83%

Table 4.2: Results of the Shot Detection algorithms.

The method with the highest accuracy was the ECR method, however, due to its lower time efficiency compared to the other methods, it was excluded. Execution time is crucial, particularly for live-streamed videos, where minimizing delays is very important. The methods with higher accuracy, apart from ECR, were the Histogram Comparison method and the Euclidean Distance method. The Euclidean Distance method exhibited the same accuracy as the Histogram Comparison method. However, it is preferable to detect a transition that didn't occur rather than failing to detect a transition that did occur, so it was decided to go with the grayscale Histogram Comparison method.

The plot shown in Figure 4.6 the results of the Histogram Comparison method for the entire video.

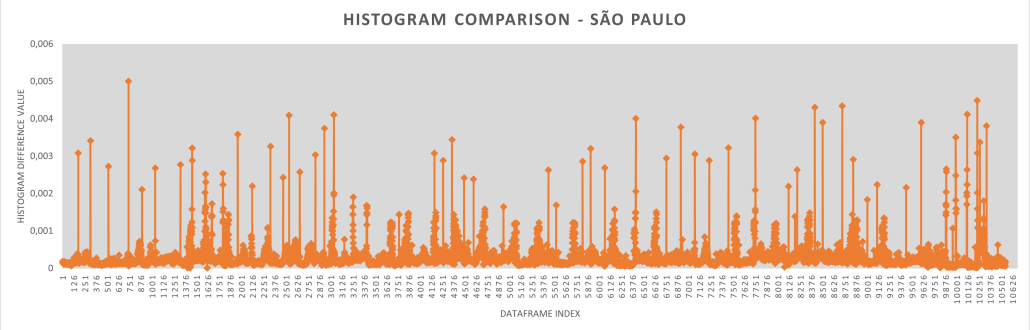

Figure 4.6: Histogram Comparison values for each frame.

4.2 Events Detection

In this section, the focus will be on defining the specific events capturing attention, including discussions about their significance and the criteria used for their selection in the recognition process. Furthermore, an examination of the recognition methods and algorithms employed will be presented regarding the technical aspects facilitating the identification of these events within the video content. This section will also elaborate on the outcomes from the recognition process. This approach aims to provide a thorough understanding of how the output of the events is organized. Essentially, serves as the focal point of the discussion, where the complexity of event recognition, its methodologies, and the impact on overarching goals and objectives will be presented.

4.2.1 Events

To initiate the process, it is essential to identify the specific events under consideration for recognition. From the vantage point of the viewer. The events that are considered as the most captivating are listed in the Table 4.3.

Event Type
Formation Lap
Starting Grid
Race Start
Lap
Red Flag, Yellow Flag and Green Flag
Safety Car
Replay
Fastest Lap
In Pit and Pit Exit
Radio
Out
Overtake
Classification

Table 4.3: Events.

The first event that is detected is the **Formation Lap**, begins 3 minutes before the race, and the racers must stay in formation and cannot overtake each other. During the formation lap, the starting tires element is detected (present in Table 4.4), creating the event **Starting Grid**, it displays the initial tires each one use and is possible to obtaining the initial position for each racer, since it displays the tires in order of the racers initial

position (1-20). This is important to detect the overtakes that happen after the race starts. This final event stores the initial positions of the racers as well as the type of tires each racer is using.

The next event that can be detected is the **Race Start**, this can be detected in several ways, one of them, is detecting the first F1 element (see Table 4.4). This element only appears if the race already has started. This is the element that appears on the video that is on every frame, except when are replays.

The race continues for a set number of laps, or until a certain amount of time has elapsed. During the race, the drivers can pit to change tires, refuel, and make repairs, this leads us to two new events: **In Pit** and **Pit Exit**. The race ends when the leader crosses the finish line for the final time, the final event. During this time, the racers can overtake each other, generating an event **Overtake** storing the information of the racers involved and the new positions. The last event is the event **Classification** storing the information of the final ranking.

There are more events adding to the ones that were previously described. These events are not mandatory, which means they only happen if something out of the ordinary occur. When there is an accident or other incident on the track, the **Safety Car** event will be detected. This means that all the cars must slow down and follow a safety car until it is safe to resume racing. If there is a serious accident or other incident on the track, the race may be red flagged and it generates another event (**Red Flag** event). This means that the race is stopped and the cars must return to the pit lane. The race will only resume once the track is clear and safe. There are more flag elements that can be detected, like the **Green Flag** and the **Yellow Flag**. The first is used to indicate that the track is clear, whether this is at the start of a warm-up lap, practice session or qualifying session, or immediately after an incident that necessitated the use of one or more yellow flags. The yellow flag is used to warn drivers of a hazard on or close to the track, but the meaning actually differs depending on whether one or two yellows flags are waved on the track. Normally, there is the information of the sector that is affected, it appears on the frame below the yellow flag element (see Table 4.4). When there is a crash or a racer gets out by himself, the event **Out** is extracted with the information of the racer involved.

Additionally, there's more events to consider, the appearance of replay (event **Replay**), fastest lap (event **Fastest Lap**) and radio elements (event **Radio**).

The information extracted from each event differs, but they all share four common attributes: the event's *ID*, its type (see Table 4.3), the timestamp, and the duration.

Further clarification regarding this information will be provided in Section 4.3.

4.2.2 Elements

The mentioned events can be detected using the information that appears within the video frames. The areas where the displayed information can match a specific meaning are designated as elements. The video starts without any relevant elements, but as it progresses, specific elements, appear and disappear within the video frame at specific times. With this information, some of the events mentioned can be detected by detecting these elements.

After analyzing several videos, the elements can be identified and utilized to detect the precise moment when an event occurred. The elements that have been visually recorded are listed in Table 4.4.

Element Type	Template
F1	
Race	
Formation Lap	
Starting Tires	
Red Flag	
Yellow Flag	
Green Flag	
Safety Car	
Virtual Safety Car	
In Pit	
Pit Exit	

Radio	
Replay	
Rank 1	
Rank 2	
Classification	

Table 4.4: Elements as templates.

For a clearer understanding of the identification of elements within video frames, Figure 4.7 illustrates an example of a video frame featuring five relevant elements: F1, Race, Red Flag, Rank 1, and Rank 2. The red flag element appearance generates the event **Red Flag**.

Figure 4.7: Video frame that includes the five elements described.

The detection of these elements introduces the template matching issue. The correlation with event identification will be discussed in Section 4.2.4.

The flowchart in Figure 4.8 demonstrates a possible sequence of elements detection.

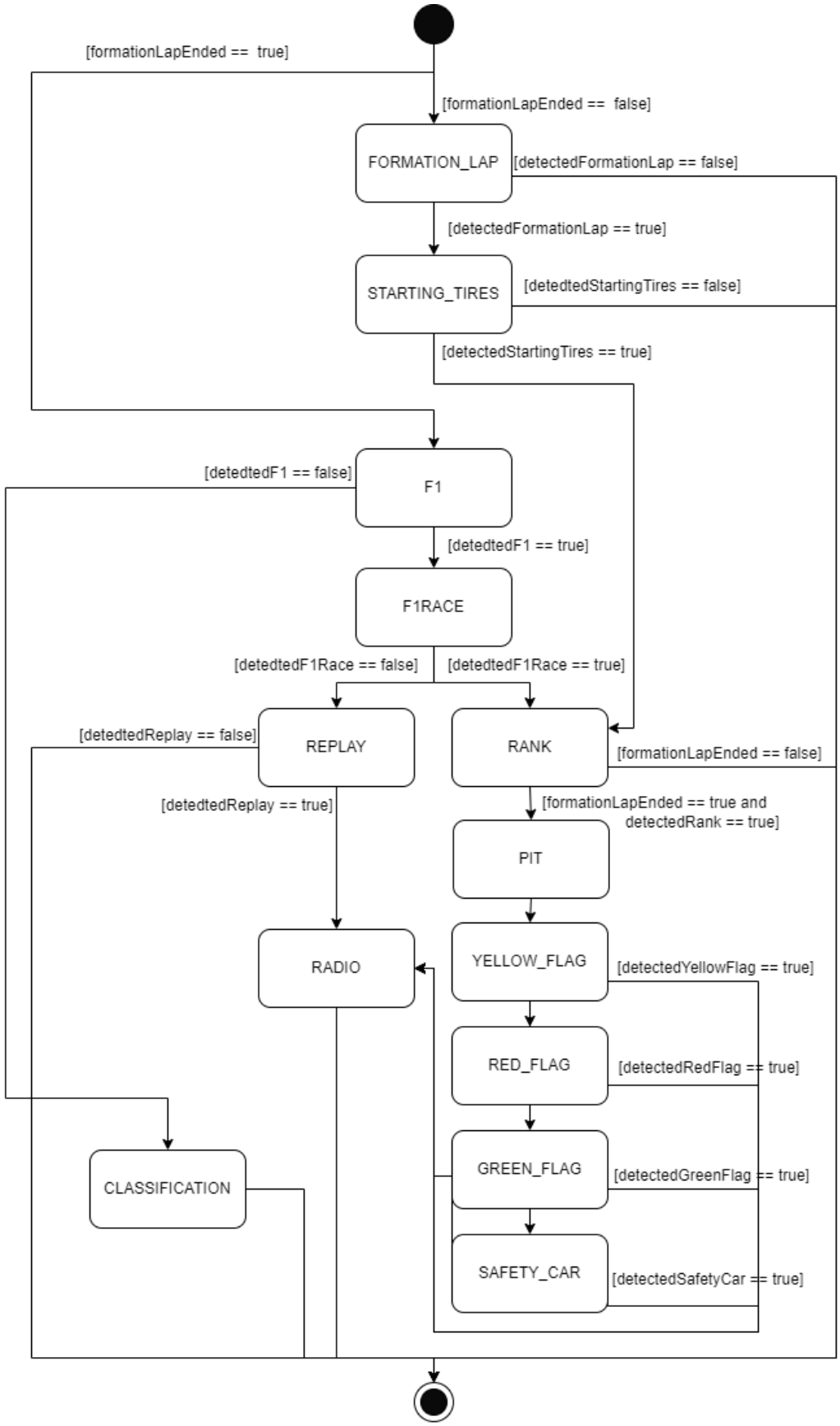

Figure 4.8: Elements detection Flowchart.

4.2.3 Search Areas

This initial step can present various challenges, notably related to where and when these elements appear on the video frame. A distinction can be made between elements with a unchanging position throughout the video and elements that vary in their location. By doing so, defined search areas can be established, which is a crucial strategy to minimize errors and expedite the execution process.

In the context of Formula 1 races, several identifiable elements can be classified as static. This refers to occurrences in which the information remains consistently located within the video frame, their position within the frame remains fixed (or shifted). These elements can easily be detected using template matching techniques, since they're the same in every video throughout the season.

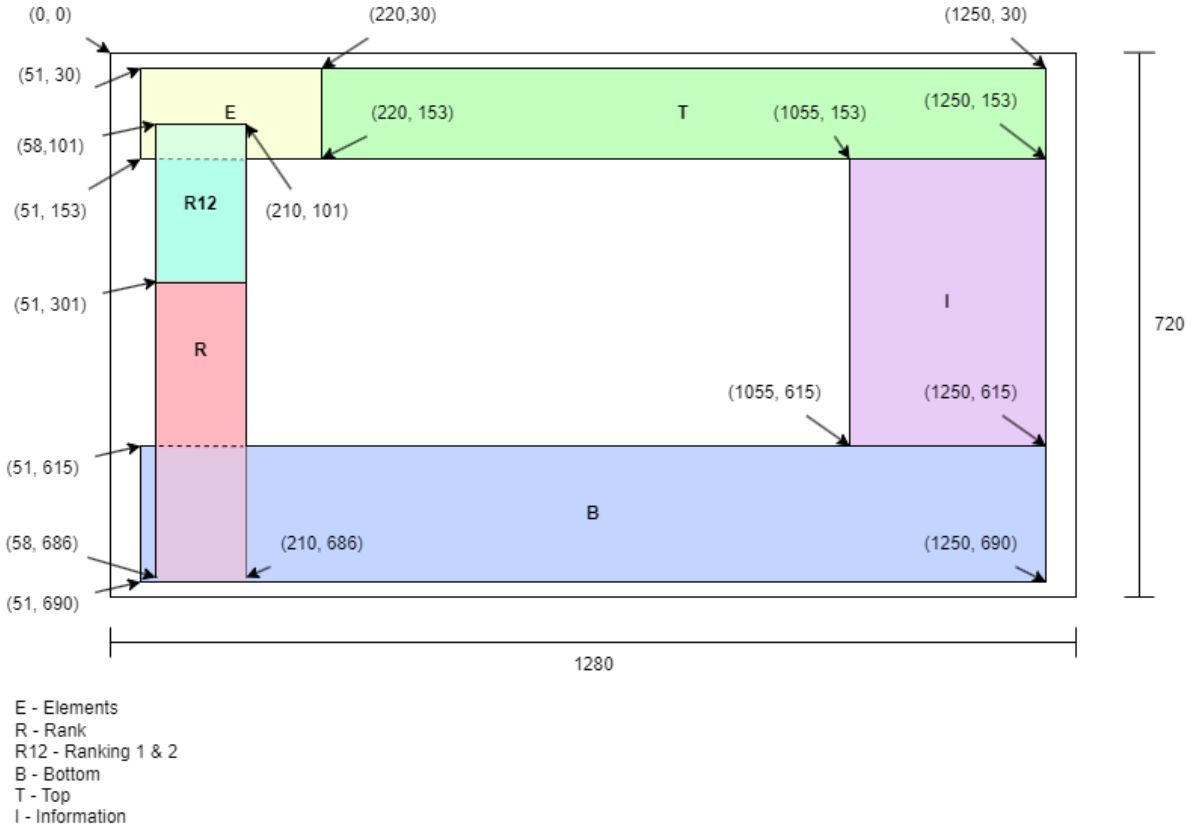

Figure 4.9: Search Areas.

These are the elements that can be identified in each area:

- **E**: F1, Race, Flags, Starting Tires, Replay, Red, Yellow and Green Flags, Starting Tires, Safety Car, Virtual Safety Car, Replay;

- **R12**: Rank 1 and 2;
- **T**: Formation Lap, Classification;
- **R**: In Pit, Pit Exit;
- **B**: Replay - Racer Focus;
- **I**: Radio;

4.2.4 Template Matching

The extraction of some events within the context of Formula 1 video analysis relies on the detection of static elements within the video frames. These elements, which consistently manifest in predetermined positions and maintain a common design, underscore the applicability of the template matching concept.

Template matching's significance in Formula 1 video frame analysis lies in its simplicity, real-time performance, flexibility, and ability to handle noisy data. Its straightforward implementation and minimal computational demands make it well-suited for live broadcasts, where immediate and precise element detection is vital for delivering a compelling viewing experience. Its robustness in handling variations in lighting, scaling, and orientation ensures reliable detection even when dealing with the unpredictable nature of Formula 1 racing, since the races can happen in different types of weather and during the day and night. Various alternatives to template matching have been explored. Feature-based methods, such as key-point matching, aim to identify distinctive features in images for object recognition. Deep learning-based approaches, notably Convolutional Neural Networks (CNNs), have demonstrated remarkable capabilities in complex pattern recognition tasks, including object detection. Furthermore, Optical Character Recognition (OCR) techniques specialize in extracting text from images, making them suitable for reading textual elements within the frames. While these alternatives have their strengths, they may face challenges in handling non-textual or less feature-rich elements, demanding more extensive computational resources, and necessitating large labeled datasets. The choice between these alternatives and template matching depends on the specific requirements, complexities, and available resources of the Formula 1 video analysis task at hand.

When discussing the events, the **Race Start** event can be extracted by detecting the F1 Race element using template matching techniques. Template matching (see Section 3.2)

works by comparing a small template image to a larger target image in order to locate instances of the template within the target image. This leads to the issue of creating a database with the templates of the elements that need to be identified. This process starts by cropping the elements out of the video frames. The figure below (Figure 4.11), shows an example of the result of this process.

Figure 4.10: Video frame used to crop the F1 element.

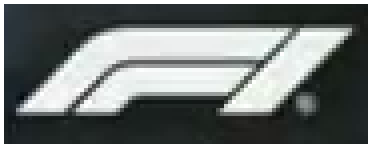

Figure 4.11: Template of F1.

The video frames can be used as the "larger target image", but since the search areas were created, for each element, the respective search area can be croped out of entire video frame and be used as the "larger target image". This reduces errors and improves the time efficiency of the execution process.

The **Race Start** event, particularly, doesn't have any information on the video frame that indicates that it's happening or that already has happened, but can be observed that it happens X frames before the first time the Race element appears. Since the only information that is needed to extract this event is the timestamp when it occured, it's easy to calculate the real timestamp of the event using the frame rate of the video in question:

$$EventTime(s) : \frac{FrameIndex - X}{FrameRate(frame/s)}$$

The detection of the **Flags** event is once again a case of template matching for each flag's template within the area where these flags can appear (search area "E"). This is valid for some of the events mentioned before, as the **Safety Car**, **In Pit**, **Pit Exit**, **Radio**, **Replay** and **Formation Lap** events.

The detection of the **Replay** event has more information associated to it, because while it is ongoing, information about the racers in focus can appear, in five different locations. This is information can be useful to determine if the replay is about an overtake

that happened before if the racer in question was involved in one of the 3 previous overtakes.

The first step starts by applying Canny edge detection to the search area ("E") and template images, followed by template matching between their edge-detected versions. Lastly, the minimum and maximum values, along with their corresponding positions at specific coordinates, the maximum location coordinates with those observed in multiple videos can be cross-checked and be more confident about the readability of its detection. The flowchart presented below illustrates a possible sequence of procedural steps:

1. **Start**
2. Apply Canny Edge Detection on the Template Image
3. Find Min and Max Values and Locations
4. Check Max Value
5. Detected Element?
6. **End**

To determine if the area given by the maximum location is indeed similar to the template image in question, a fixed threshold value was decided. Setting a predetermined threshold value serves as a dividing line between what is considered valid and what is not.

4.2.5 Color Detection

The concept of color detection takes center stage to identify and track the tires associated with each racer. This approach plays provides critical insights into the race dynamics. Unlike template matching, where individual templates are created for each element, color-based detection leverages the unique color of each element, that in this case is the color of the tires displayed. This method offers unparalleled speed, simplicity, and accuracy, making it ideal for real-time applications and scenarios where tires have distinct and consistent colors.

The Formula 1 tire selection offers a range of options, each tailored to specific racing conditions. Five types of tires can be itentified:

- Hard (H): The hard compound tires are the most durable and have the longest lifespan among the tire choices. They are often used on tracks with abrasive surfaces or high-speed tracks with fewer turns. While they provide excellent durability, they may lack the grip of softer compounds. These tires are displayed in white.

- Medium (M): Medium compound tires offer a balance between grip and durability. They are used on a wide range of tracks and are often the choice for longer stints in a race. Medium tires are favored when teams are looking for a compromise between performance and longevity. These tires are displayed in yellow.

- Soft (S): Soft tires provide higher levels of grip but wear out more quickly than the hard and medium compounds. They are often chosen for qualifying sessions or when teams need to maximize their performance during a short stint in a race. These tires are displayed in red.

- Wet (W): Wet weather tires are specially designed for use in rainy conditions. They have deep treads to disperse water and provide better traction on wet surfaces. These tires are displayed in blue.

- Intermediate (I): Intermediate tires are designed for damp or drying conditions. They have less tread than full wet tires and are used when the track is no longer completely wet but not yet dry enough for slick tires. These tires are displayed in green.

The information about the tires is displayed at two different times: on the starting grid and after the race has started, in the ranking. The two scenarios are presented in Figures 4.12 and 4.13, with the area where the tires are listed highlighted in green.

The color displayed in the videos matches the color on each tire.

Since the tire information is consistently displayed in the same manner in every video, it is possible to determine the tire colors for each racer and get its type, rather than relying on Tesseract to extract the tire names. Collecting the colors is a faster and more reliable process than using Tesseract.

The Figures to illustrate the tire information displayed (elements) on the starting grid for the three types of tires implemented.

To obtain the tire information for each racer, first, the region designated for this purpose is identified, and then extract the number of pixels in specific color ranges within that image. The color ranges of interest are, red, yellow, and white (only the dry tires

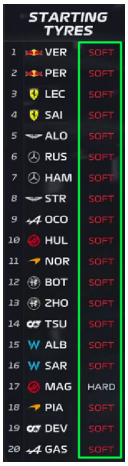

Figure 4.12: Starting Grid with the tired highlighted.

Figure 4.13: Ranking with the tired highlighted.

Figure 4.14: Different types of tires.

Figure 4.15: Hard tires.

Figure 4.16: Medium tires.

Figure 4.17: Soft tires.

are considered yet), which are defined using lower and upper bounds in the BGR color space. Then, a mask can be created for each color range, and count the number of pixels in each mask. Basically if the number of pixels (different of zero) of a certain mask is higher than the rest of the masked images, the tire will be the one who correspond to that color.

The range of the colors used for the three tire types are the following:

Figure 4.18: Hard tires.

Figure 4.19: Upper bound.

Figure 4.20: Lower bound.

Figure 4.21: Medium tires.

Figure 4.22: Upper bound.

Figure 4.23: Lower bound.

Figure 4.24: Soft tires.

Figure 4.25: Upper bound.

Figure 4.26: Lower bound.

4.2.6 Text Recognition

Certain events within the broadcast may pose challenges when employing the previously described procedures. In some instances, the methods of template matching and color detection may prove to be prohibitively slow or even ineffective when applied to the events mentioned in this section. The events that were detected using the Tesseract OCR are the **Overtake** and **Lap** event.

One of the most significant events involves detecting the laps. In this context, an event is created to indicate when a lap transitions to the next. The process begins by detecting the lap's position within the video frame. There are two instances where the position differs, as the lap can appear when the Race element is either on or off. The Figures 4.27 and 4.28 illustrate the difference in position between the two cases:

Both images were cropped with their left corners aligned to the left edge of the video frame. Here, the lap information is identified as being shifted horizontally to the right. This is essential for automatically cropping only the lap information within the video

Figure 4.27: Lap without the Race element present.

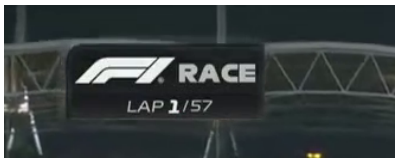

Figure 4.28: Lap with the Race element present.

frames, eliminating any blank spaces.

After detecting the positioning of the lap information, the initial step involves preprocessing the designated area for the OCR (Figure 4.29). As it was described in the Chapter 3, the Tesseract OCR receives an image and extracts the information to be machine-readable text. First, there's a few steps that may be considered to improve the extraction of the text. The test videos were all with a resolution of 720p, which is relatively small, especially when considering text areas that occupy a maximum of 475 pixels. So, after converting the area to a grayscale image, it is resized to three times the original size, enhancing image quality as a preprocessing step. Next, Otsu's binary thresholding is applied to create a binary image. A $3x3$ rectangular structuring element is defined, and morphological erosion is performed on the binary image to reduce the size of white regions. The final step involves bitwise inverting the binary image to produce the final binary result (Figure 4.30).

In the first time the lap information appears, what goes to the OCR, after the preprocessing phase, is the following (Figure 4.30):

Figure 4.29: First lap area.

Figure 4.30: First lap area after the preprocessing phase.

This enables the extraction of the total number of laps, and with this information already obtained, only the digits before the slash remain for the OCR input in subsequent steps. This would produce a very small image containing only the digits representing the current lap (Figure 4.31). Since Tesseract OCR was primarily trained on sentences, the recognition of isolated characters is less accurate compared to words or phrases. Therefore, a decision was made to include additional information ("-CAT") to the digits area (concatenate to the right), resulting in significantly improved results (Figure 4.32). The decision to add information instead of cropping a larger area was due to the

consistent nature of the "-CAT" information. Since the elements have a certain opacity, there were instances where OCR, for example, failed to detect the slash.

Figure 4.31: Second Lap area.

Figure 4.32: Second Lap area after the pre-processing phase.

The next event that required the OCR utilization is the detection of overtakes (**Overtake** event). This event is identified by detecting differences in the ranking displayed. The procedure for checking if the ranking has changed is the same as detecting shot transitions. The image used to calculate the difference of histograms is instead a cropped image of the ranking area. It's important to note that this check is only performed when, at least, one of the elements Rank 1 or 2 is detected.

The process begins by detecting the ranking's position within the video frame. The position (and height) of the ranking differs with the appearance of certain elements, such as flags and safety car elements.

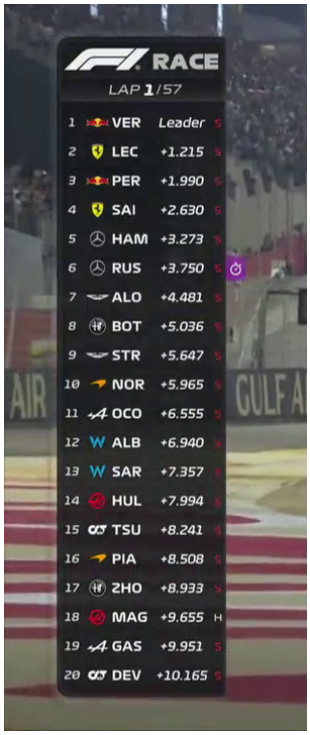

Figure 4.33: Ranking.

Figure 4.34: Ranking with a flag element.

Both images (Figures 4.33 and 4.34) were cropped with their left corners aligned to

the left edge of the video frame. Here, the ranking information is identified as being shifted vertically and shortened.

After extracting the ranking area from the video frame, the preprocessing steps are the same as the ones described before. The area of interest here is the vertical area containing all the names of the racers (Figure 4.35).

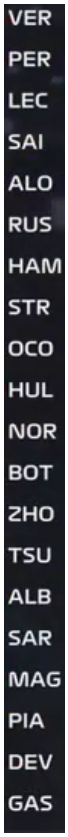

Figure 4.35: Ranking area.

Figure 4.36: Ranking area after preprocessing.

The output consists of a list with all the racers' names in order. This can be achieved by parsing the output text. If the order of the names in that list differs from the previous one, it indicates an overtake has occurred, and the event is extracted. The overtake event will store the information of the racers whose position has changed.

4.3 Events Output

When extracting events from a Formula 1 video, JSON (JavaScript Object Notation) has been chosen as the data format for storing this information. JSON offers several distinct advantages over XML (eXtensible Markup Language) in this specific context.

Firstly, JSON is more human-readable and concise than XML. Its syntax is straightforward and easier to comprehend, making it accessible not only to developers but also to individuals who may need to work with the data without extensive technical knowledge. This readability can be particularly valuable in the broadcasting system, where quick access to event data is essential.

Furthermore, JSON typically results in smaller file sizes compared to equivalent XML data. In a domain where events are frequently updated, such as Formula 1 races, compact data representation can significantly reduce storage requirements and improve data transmission efficiency.

JSON's ease of parsing is another key advantage. Most modern programming languages provide native support for JSON, simplifying the process of reading and manipulating the data. This streamlined handling of data can save valuable development time and resources.

The efficiency of JSON extends to its compatibility with web technologies, making it an ideal choice for integration into web applications and APIs. It aligns seamlessly with the technologies commonly used in Formula 1 data systems, facilitating the exchange of event data between different parts of the infrastructure.

In summary, our decision to use JSON for storing Formula 1 event data is driven by its advantages in terms of readability, compactness, ease of parsing, compatibility with modern programming languages and web technologies, and its status as a widely accepted standard for data interchange. This choice enhances the efficiency and accessibility of the data extraction process within the dynamic world of broadcasting.

4.3.1 Events as JSON Objects

The JSON object contains key information about a particular event in the context of Formula 1 racing. The "type" key indicates its category. The "title" key offers a descriptive title, potentially for identification purposes (must be unique). The "offset" is another key, denoting the timestamp or time offset when the event occurred. Finally, the "duration" key provides the duration of the event in seconds. These keys collectively organize and present vital details about the event, facilitating its interpretation and analysis within the Formula 1 racing context. The JSON object, can additionally have a "metadata" key, that can save all the additional information. All keys are mandatory except for the "metadata" key. Storing the information like this is important for future work that may be developed. Since one of the goals is to provide this information to a broadcasting network, all the keys are crucial to further utilization.

For example, to create a chronological order of events.

Here is an example of a *Formation Lap* event:

```
{
    "type":"FormationLap",
    "title":"FormationLap1",
    "offset":"5:5",
    "duration":"1:54"
}
```

There is an example of each event in Annex (Section 7).

5

Experiments

In this experiments chapter, the results and analysis from two Formula 1 videos from the same season (2023) will be presented, all of which with a resolution of $720p$. Tipically, during the seasons, the information displayed doesn't change throughout the whole year.

The application was developed in C++ on a *64-bit Ubuntu Linux* system, within a virtual machine (*Orable VM Virtual Box* [16]) with a base memory of *5073MB* and utilizing two processors.

The Table 5.1 displays the number of laps, duration and execution time for each race. The **Laps** row indicates the total number of laps, the **Duration** row provides the race duration in the format of $HH:MM:SS$, while the **Execution Time** row represents the time the program required to process the entire video, also presented in the same time format.

	Race	
	Bahrain	**Australia**
Laps	57	58
Duration	02:03:29	03:04:29
Execution Time	01:00:18	01:53:04

Table 5.1: Races information.

The information presented in Table 5.1 shows that the execution time is approximately

half the duration of the video. One of the goals was to reduce this execution time. This is very important when discussing about broadcasting. Several options were considered, such as selecting specific search areas (Figure 4.9), implementing template matching for elements only when that may appear on the screen (Figure 4.8), optimizing Tesseract OCR to use only the necessary information, processing methods in 1-second intervals (which is sufficient, as elements typically remain on the screen for several seconds), and exploring various other strategies.

To evaluate the success of the program, the number of events was initially determined by manually counting them while viewing the races. However, due to the unreliability of this process, as well as its significant time-consuming nature, the actual number of events was subsequently omitted for the two races.

The Table 5.2 displays the events for the Bahrain race, dividing them as **Detected** (events detected automatically) and **Verified**, which encompasses all the events captured manually. The **Overtake** event was excluded because it's challenging to count it manually, given that the positions are updated in a new event every second whenever there are changes in the ranking. The row **Total** is the sum of the number of the events above. The events **Formation Lap**, **Starting Grid**, **Race Start**, and **Classification** were detected once, while the events **Lap** and **Fastest Lap** were detected in every lap of each race.

		Race	
		Bahrain	
		Detected	**Verified**
Events	**Green Flag**	2	2
	Yellow Flag	2	2
	Red Flag	0	0
	Safety Car	2	1
	Replay	19	19
	Radio	28	30
	In Pit	41	48
	Pit Exit	26	46
	Out	3	3
	Total	123	151

Table 5.2: Detected and verified events in the Bahrain race.

This gives an accuracy of 80%, sensibly. Particularly, the event **Pit Exit**, has a lot of failures since it's only being extracted when the pit exit element is detected. Sometimes, this element isn't present on the video and it's assumed it happened. The process to assume when it happened wasn't yet developed. The other event that shows a failure in the program is the **Safety Car**. In the Table 5.2 this event was detected two times. This

happened because the safety car element disappeared to show a replay and appeared again after that. The process to determine if the safety car's elements are related wasn't yet developed also.

An interesting experiment involved manually creating a small video summarization. This approach, when automated, proves to be particularly valuable in broadcasting scenarios when users may have missed specific events and wish to review them. This would be accomplished by placing the video at the beginning of the shot where an event occurred to ensure a smoother and more engaging user experience when revisiting a particular event.

For example, when focusing on the event mentioned in Listing 5.1 and examining the timestamps of the shots around that time (Listing 5.2), the chosen timestamp for "rewatching" is the one corresponding to the first shot that occurred just before the event. In this case, would be the shot whose title is "Shot9460". The duration could be the time between the considered shot ("Shot9460") and the shot "Shot10467", since the time between these shots is higher than the duration of the event.

```
{
    "type":"YellowFlag",
    "title":"YellowFlag1",
    "offset":"10:19",
    "duration":"0:29"
}
```

Listing 5.1: Event Yellow Flag 1.

```
...
{
    "type":"Shot",
    "title":"Shot9460",
    "offset":"10:18",
    "duration":"0"
}
{
    "type":"Shot",
    "title":"Shot10106",
    "offset":"10:39",
    "duration":"0"
}
{
```

```
15          "type":"Shot",
16          title":"Shot10202",
17          "offset":"10:43",
18          "duration":"0"
19      }
20      {
21          "type":"Shot",
22          "title":"Shot10467",
23          "offset":"10:51",
24          "duration":"0"
25      }
26      ...
```

Listing 5.2: Part of Shot Events.

6

Conclusion

In conclusion, this study delved into the realm of Formula 1 event extraction, focusing on the analysis of three Formula 1 videos from the 2023 season. The objective was to develop a program that could automatically identify and summarize key events in more kinds of motorsport races, but the focus relied on Formula 1 races, contributing to an enhanced viewer experience and potential applications in broadcasting.

One of the paramount goals of this research was to reduce the execution time of the program, a critical factor for real-time broadcasting applications. Various strategies were explored, such as specific search areas, template matching, and optimized OCR usage, demonstrating the commitment to enhancing program efficiency.

The results showcased the program's capability to detect a diverse array of events. While the program demonstrated impressive event detection capabilities, the verification process revealed an accuracy rate of approximately 80%. Some challenges were encountered in certain event categories, notably the **Pit Exit** event, where occasional absence of visual cues led to assumptions. Further refinement in event verification processes is necessary for improved accuracy. The study acknowledged the complexities associated with Formula 1 event extraction, particularly the need to address scenarios where elements briefly disappear and reappear, as seen with the **Safety Car** event. The development of another algorithms to handle such cases is a future research direction.

In summary, this research represents a step toward automating Formula 1 event extraction, demonstrating the potential for creating racing content and enhance the broadcasting experience. Despite the challenges encountered, the study lays the foundation

for continued advancements in this field, with opportunities to refine event detection algorithms and enhance the accuracy of event extraction processes. As Formula 1 and similar sports continue to captivate global audiences, the pursuit of innovative technologies to improve content summarization remains critical for both broadcasters and fans alike.

7

Future Work

As part of future research, the objective is to broaden the testing scope by incorporating a more extensive selection of races, including those from different seasons. The current testing phase has been limited to videos exclusively from the 2023 season, with all template data sourced from that specific timeframe. However, the potential lies in diversifying the dataset by including races from various seasons.

Furthermore, the project's aspirations go beyond the limits of a single sport. The plan is to expand the program's applicability to encompass a wider range of sports, acknowledging that similar video analysis techniques could find utility in various athletic disciplines.

The initial objective is to assess the program's performance when applied to race broadcasts. This evaluation will serve to measure both the effectiveness and efficiency of the project in this context. By systematically examining how well the program detects and analyzes race events, the aim is to refine and enhance its capabilities, ultimately making it a versatile tool for enthusiasts and sports broadcasting analysis across different domains.

Annex

Events as JSON Objects


```
{
    "duration":"1:54",
    "offset":"5:5",
    "title":"FormationLap1",
    "type":"FormationLap"
},
{
    "duration":"0",
    "offset":"8:18",
    "title":"RaceStart0",
    "type":"RaceStart"
},
{
    "duration":"0",
    "metadata":[{"position":"1","racer":"ALB","tires":"HARD"},
                {"position":"2","racer":"ALO","tires":"MEDIUM"}
  ,
                {"position":"3","racer":"BOT","tires":"HARD"},
                {"position":"4","racer":"DEV","tires":"HARD"},
                {"position":"5","racer":"GAS","tires":"SOFT"},
                {"position":"6","racer":"HAM","tires":"HARD"},
                {"position":"7","racer":"HUL","tires":"HARD"},
                {"position":"8","racer":"LEC","tires":"HARD"},
```

```
                {"position":"9","racer":"MAG","tires":"SOFT"},
                {"position":"10","racer":"NOR","tires":"HARD"},
                {"position":"11","racer":"OCO","tires":"HARD"},
                {"position":"12","racer":"PER","tires":"HARD"},
                {"position":"13","racer":"PIA","tires":"HARD"},
                {"position":"14","racer":"RUS","tires":"HARD"},
                {"position":"15","racer":"SAI","tires":"HARD"},
                {"position":"16","racer":"SAR","tires":"HARD"},
                {"position":"17","racer":"STR","tires":"HARD"},
                {"position":"18","racer":"TSU","tires":"HARD"},
                {"position":"19","racer":"VER","tires":"HARD"},
                {"position":"20","racer":"ZHO","tires":"HARD"}]
    ,
    "offset":"5:49",
    "title":"StartingGrid0",
    "type":"StartingGrid"
},
{
    "duration":"0",
    "metadata":[{"position":"1","racer":"VER","tires":"HARD"},
                {"position":"2","racer":"ALO","tires":"HARD"},
                {"position":"3","racer":"BOT","tires":"HARD"},
                {"position":"4","racer":"DEV","tires":"HARD"},
                {"position":"5","racer":"GAS","tires":"HARD"},
                {"position":"6","racer":"HAM","tires":"HARD"},
                {"position":"7","racer":"HUL","tires":"HARD"},
                {"position":"8","racer":"LEC","tires":"HARD"},
                {"position":"9","racer":"MAG","tires":"SOFT"},
                {"position":"10","racer":"NOR","tires":"HARD"},
                {"position":"11","racer":"OCO","tires":"HARD"},
                {"position":"12","racer":"PER","tires":"HARD"},
                {"position":"13","racer":"PIA","tires":"HARD"},
                {"position":"14","racer":"RUS","tires":"HARD"},
                {"position":"15","racer":"SAI","tires":"HARD"},
                {"position":"16","racer":"SAR","tires":"HARD"},
                {"position":"17","racer":"STR","tires":"HARD"},
                {"position":"18","racer":"TSU","tires":"HARD"},
```

```
                {"position":"19","racer":"VER","tires":"HARD"},
                {"position":"20","racer":"ZHO","tires":"HARD"}]
   ,
    "offset":"16:50",
    "title":"6",
    "type":"Lap"
},
{
    "duration":"0:3",
    "offset":"75:3",
    "title":"YellowFlag2",
    "type":"YellowFlag"
},
{
    "duration":"0:3",
    "offset":"75:3",
    "title":"RedFlag1",
    "type":"RedFlag"
},
{
    "duration":"0:3",
    "offset":"76:3",
    "title":"Green2",
    "type":"YellowFlag"
},
{
    "duration":"0:3",
    "metadata":[{"racer":"STR", "lap":"2"}]
    "offset":"75:3",
    "title":"FastestLap1",
    "type":"FastestLap"
},
{
    "duration":"0:15",
    "offset":"75:21",
    "title":"SafetyCar1",
    "type":"SafetyCar"
```

```
},
{
    "duration":"0:12",
    "metadata":[{"racer":"STR"}],
    "offset":"7:5",
    "title":"Replay1",
    "type":"Replay"
},
{
    "duration":"0",
    "metadata":[{"racer":"HUL"}],
    "offset":"76:45",
    "title":"InPit36",
    "type":"InPit"
},
{
    "duration":"0",
    "metadata":[{"racer":"GAS"}],
    "offset":"24:25",
    "title":"PitExit0",
    "type":"PitExit"
},
{
    "duration":"0:6",
    "offset":"26:55",
    "title":"Radio5",
    "type":"Radio"
},
{
    "duration":"0",
    "offset":"33:5",
    "title":"PIA",
    "type":"Out"
},
{
    "duration":"0",
    "metadata":[{"position":"9","previousPosition":"10",
```

```
                    "racer":"RUS"},
                {"position":"10","previousPosition":"9",
                    "racer":"MAG"}],
    "offset":"33:33",
    "title":"Overtaking30",
    "type":"Overtaking"
},
{
    "duration":"0",
    "offset":"71:33",
    "title":"Classification",
    "type":"Classification"
}
```

References

[1] S. Jothi Shri. "Crowd video event classification using convolutional neural network". (Nov. 2019), [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S014036641930595X.

[2] Sandra E. F. de Avila, Antonio da Luz Jr., Arnaldo de A. Araujo, and Matthieu Cord, *VSUMM: An Approach for Automatic Video Summarization and Quantitative Evaluation*. XXI Brazilian Symposium on Computer Graphics and Image Processing IEEE, 2008.

[3] Geraci Marco and Montenegro, *STIMO: Still and Moving video storyboard for the Web Scenario*. Journal Multimedia Tools et al., 2010.

[4] Azra Nasreen and Dr Shobha G, *Key Frame Extraction using Edge Change Ratio for Shot Segmentation*. International Journal of Advanced Research in Computer and Communication Engineering, Vol 2., 2013.

[5] Karim M. Mohamed, Mohamed A. Ismail, and Nagia M. Ghanem, *VSCAN: An Enhanced Video Summarization using Density-based Spatial Clustering*. Computer and Systems Engineering Department Faculty of Engineering, Alexandria University Alexandria, Egypt, 2014.

[6] Tinumol Sebastian and Jiby J. Puthiyidam. "A survey on video summarization techniques". (Dec. 2015), [Online]. Available: https://www.ijcaonline.org/research/volume132/number13/sebastian-2015-ijca-907592.pdf.

[7] Muhammad Ehsan Anjum, Syed Farooq Ali, Malik Tahir Hassan, and Muhammad Adnan. "Video summarization: Sports highlights generation". (Dec. 2013), [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6731340.

[8] OpenCV. "Histogram comparison". (), [Online]. Available: https://docs.opencv.org/3.4/d8/dc8/tutorial_histogram_comparison.html.

[9] Xiaocui Liu, Jiande Sun, and Ju Liu. "Shot-b ased temporally respective frame generation algorithm for video hashing". (Nov. 2013), [Online]. Available: https://www.researchgate.net/publication/261336199_Shot-based_temporally_respective_frame_generation_algorithm_for_video_hashing#fullTextFileContent.

[10] Azra Nasreen and Dr Shobha G. "Key frame extraction using edge change ratio for shot segmentation". (Nov. 2013), [Online]. Available: https://www.ijarcce.com/upload/2013/november/52-s-Azra_Nasreen-key_frame.pdf.

[11] OpenCV. "Canny edge detection". (), [Online]. Available: https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html.

[12] Reginald Best Jr. "Building a dataset and developing a video event classifier for football". (Jun. 2023), [Online]. Available: https://dspace.mit.edu/bitstream/handle/1721.1/151510/best-regbest-meng-eecs-2023-thesis.pdf?sequence=1&isAllowed=y.

[13] MathWorks. "Psnr compute peak signal-to-noise ratio (psnr) between images". (), [Online]. Available: https://www.mathworks.com/help/vision/ref/psnr.html.

[14] OpenCV. "Template matching". (), [Online]. Available: https://docs.opencv.org/4.x/d4/dc6/tutorial_py_template_matching.html.

[15] Filip Zelic and Anuj Sable. "How to ocr with tesseract, opencv and python". (), [Online]. Available: https://nanonets.com/blog/ocr-with-tesseract/.

[16] Oracle. "Oracle vm virtual box". (), [Online]. Available: https://www.virtualbox.org/.

[17] OpenCV. "Operations on arrays". (), [Online]. Available: https://docs.opencv.org/3.4/d2/de8/group__core__array.html#gae15fc2d956eb3a93ec65339a50dc7b6a.

[18] Tao Sun. "A deep learning based real-time pedestrian recognition system". (Jun. 2021), [Online]. Available: https://dspace.mit.edu/bitstream/handle/1721.1/139317/sun-taosun-sm-sdm-2021-thesis.pdf?sequence=1&isAllowed=y.

[19] "Darknet-53". (Apr. 2018), [Online]. Available: https://paperswithcode.com/method/darknet-53.

[20] Leonardo Moraes, Ricardo Marcondes Marcacini, and Rudinei Goularte. "Video summarization using text subjectivity classification". (Nov. 2022), [Online]. Available: https://dl.acm.org/doi/pdf/10.1145/3539637.3556998.

[21] M. Said Aydemir, Ugur Ergul, and Adem Gucluand M. Elif Karsligil. "Video summarization using simple action patterns". (Nov. 2012), [Online]. Available: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6460562.

[22] N. Dalaland B. Triggs, *Histograms of oriented gradients for human detection*. CVPR 2005. IEEE Computer Society Conference, 2005.

[23] Mrinal Tyagi. "Hog (histogram of oriented gradients): An overview". (Jul. 2021), [Online]. Available: https://towardsdatascience.com/hog-histogram-of-oriented-gradients-67ecd887675f.

[24] Padmavathi Mundur, Yong Rao, and Yelena Yesha, *Key frame-based video summarization using Delaunay clustering*. International Journal on Digital Libraries April 2006 et al., 2006.

[25] Ray Smith, "An overview of the tesseract ocr engine", in *ICDAR '07: Proceedings of the Ninth International Conference on Document Analysis and Recognition*, Washington, DC, USA: IEEE Computer Society, 2007, pages 629–633, ISBN: 0-7695-2822-8. [Online]. Available: https://storage.googleapis.com/pub-tools-public-publication-data/pdf/33418.pdf.