Image Feature Matching for Scene Extraction from Single-Camera Videos

by

Hemanchal Joshi

A Thesis Submitted to

Saint Mary's University, Halifax, Nova Scotia
in Partial Fulfillment of the Requirements for
the Degree of Master of Science in Computing and Data Analytics

December, 2021, Halifax, Nova Scotia

Approved by:

Approved: Dr. Yasushi Akiyama,
Supervisor

Approved: Dr. Sageev Oore,
External Examiner

Approved: Dr. Jiju Poovvancheri,
Examiner

Approved: Dr. Hai Wang,
Examiner

December 21, 2021

ABSTRACT

Image Feature Matching for Scene Extraction from Single-Camera Videos

By Hemanchal Joshi

Template matching is a widely-used technique for finding an image or a part of an image that matches the specified reference image, called a template. A tool was implemented using this technique to find template images in a soccer video. In the initial implementation of this tool, one approach of the template matching, template-based matching, was used but it lacked versatility and had some limitations for analyzing single-camera videos.

In this thesis, we first analysed the existing template matching approach and identified its limitations. We then explored alternate approaches that could potentially address the identified limitations and found that feature-based template matching produced most optimal results for the tested videos. Feature-based template matching uses keypoints localization to detect the template in an image. The results showed that feature-based template matching could detect templates despite variances in the video quality and overcome some of the identified limitations of the existing approach.

December 21, 2021

Contents

List of Figures

Chapter 1

Introduction

With the wide usage of computer and mobile device technologies that enable easy ways to create, store, and share multimedia data such as pictures, videos, and sound, enormous multimedia data is accumulated to be handled. While approaches introduced in research areas such as data mining play a significant role in understanding collective data, especially for narrowing down the search space [3], we need more assistance to understand and manipulate individual multimedia data using data analysis approaches.

The databases have become an abundant knowledge base of learning for all kinds of trades, study, or businesses for data modeling and future prediction in respective fields. It also helps in improved decision making [4]. To conduct data analysis, we analyze the pre-existing data to observe useful information and introduce systems to find certain patterns within the available dataset. Some such examples are using data mining techniques to diagnose and treat heart disease [5], using big data for research in psychology [6], and so on. In the present context, with the increasing use of mobile phone cameras and recording devices, multimedia data is recorded and distributed in abundance in day-to-day life, but multimedia data analysis is still a challenge because of the complex nature of multimedia data [7]. Images are a type of multimedia data that can be collectively presented as a video. Analysis of a set of images is a complex process, and we can use them in data analysis to achieve different goals, ranging from object detection to modelling data in a business environment.

In some scenarios, like in a sports videos of soccer or hockey, it is not always ideal to go through the whole video when you want to watch the highlights of the game or analyze the performance of a single player in the team. Watching the whole video is time-consuming if all you want is a summarized detail of the games key moments. While the video highlights are created by experienced human video editors in the industry using multiple cameras and a large amount of data, efforts have been made to automate the editing process. We can take the example of USTA (United States Tennis Association), which uses IBM Watson to analyze various match data, including extraction of highlights based on the sound perceived from the crowd and many other parameters [8]. Such efficient data analysis is possible using video streams from multiple cameras and the noise from spectators, that results in the collection of a large amount of raw data. For simple scenarios like a practice match where a coach can be using a single-camera (mobile phones and a consumer-level camcorder) to capture the whole match. Such videos have no crowd or other parameters that can ease the data analysis and help the user to extract the highlights (video frames containing key moments). Considering all these shortcomings, including the availability of raw data for single camera videos can create a challenge in multimedia data analysis and information retrieval. In this thesis, we will try to overcome some of these challenges and then analyze the goalkeeper's performance for single camera video under various circumstances.

1.1 Background and Motivation

A single image consists of many pixels, each holding a pixel value, and all these pixel values play a decisive role in forming the picture [9]. Therefore, a single image in a multimedia database has countless pixel values that define the particular data. For example, an image of a dog may be a single image, but it may carry millions of pixel values that combine and define that image singularly. Images, videos and other similar entities fall under the multimedia data category. Multimedia data nowadays is also used as a tool for learning in many sectors. The availability of various multimedia tools tends to enhance the understanding and accelerate the knowledge intake if used appropriately. Unlike the fully automated multimedia data analysis tool, an interactive design seeks to maintain a balance between human control and skillful action [10]. Interactivity is where the term Human in the Loop (HITL) comes into existence. HITL is a progressive way associated with modeling and simulation [11]. It allows the users to experience an interactive communication phase live with a computerized interface. Under certain circumstances, the input provided by a user can tweak the overall performance of the system. In HITL, humans taking part in tuning, training, and testing various algorithms in a machine learning environment. HITL reflects the influence of the interactive interfacing between the computer and the human users.

Understanding the concept of multimedia data can be time-consuming and challenging.

Common ways to understand the multimedia content are listening to the sound, viewing the images, or playback video and animation. One encounters this type of issue in different tasks, such as selecting several pictures from a few hundred candidate pictures for certain multimedia tasks. Typically, one would manually go through the thumbnails of all these pictures. We do this while capitalizing on particular heuristic filtering criteria such as dates and locations. A similar task with video data (e.g., a videographer is trying to find certain video segments to be used for a documentary film) can exponentially increase the time spent on the search. The thumbnail view for videos typically only shows a single video frame that may or may not be the most relevant frame to represent the video content. In these cases, one would need to open each candidate video to play it back and search its content to find relevant video sections. Similar and related issues can occur when navigating through a music library or video streaming services.

This study is a part of a larger research program on the interface designs of video analysis tools [12]. The template-based template matching approach is used on a goalkeeper performance analysis tool that detects the goalkeeper involved in certain plays in a soccer game video. There are numerous challenges while using single-camera videos as input for multimedia data analysis. Some of these challenges include image quality, resolution, steadiness, poor-visibility conditions, and transformation (zoomed and rotated image frames). Single-camera videos are shot in a single angle at a time, while in fact, the use of multiple cameras can give the same play in multiple angles simultaneously, potentially with better quality images despite requiring calibrated setups and a large cost to implement it [13]. The study presented in this thesis investigates approaches that address these limitations posed by single-camera videos.

1.2 Problem Statement

This research is a part of an existing research where we try to overcome the limitations of the current template matching approach. The existing system currently uses a template-based template matching algorithm to extract the soccer goalkeeper interaction moments. In the existing system, the user selects a rectangular reference frame called template, and the algorithm runs a search on all the image frames of the input video based on this template as shown in Figure 1.1.

The system then compares every video frame with the template to check if that frame contains the template and calculates the likelihood. The matching process repeats for every video frame in the video by overlapping and calculating the maximum obtained value for each frame. The system then plots the obtained values for all the video frames for users to select an optimal threshold as shown in Figure 1.2. The threshold is a user-selected value from the graph that separates the likely and unlikely frames to contain the template. The user is responsible for observing each frame's likelihood values and selecting a threshold. After a threshold is selected, the video frames holding equal or higher values than the threshold are compiled and extracted as a single short output video. To allow the user to select the template is a significant component of the study that focuses on interactive interfacing for multimedia data analysis. Although the matching process is simply done by overlapping and matching the template with a target frame, a few limitations can hinder the system from extracting optimal output. These limitations are visual aspects such as

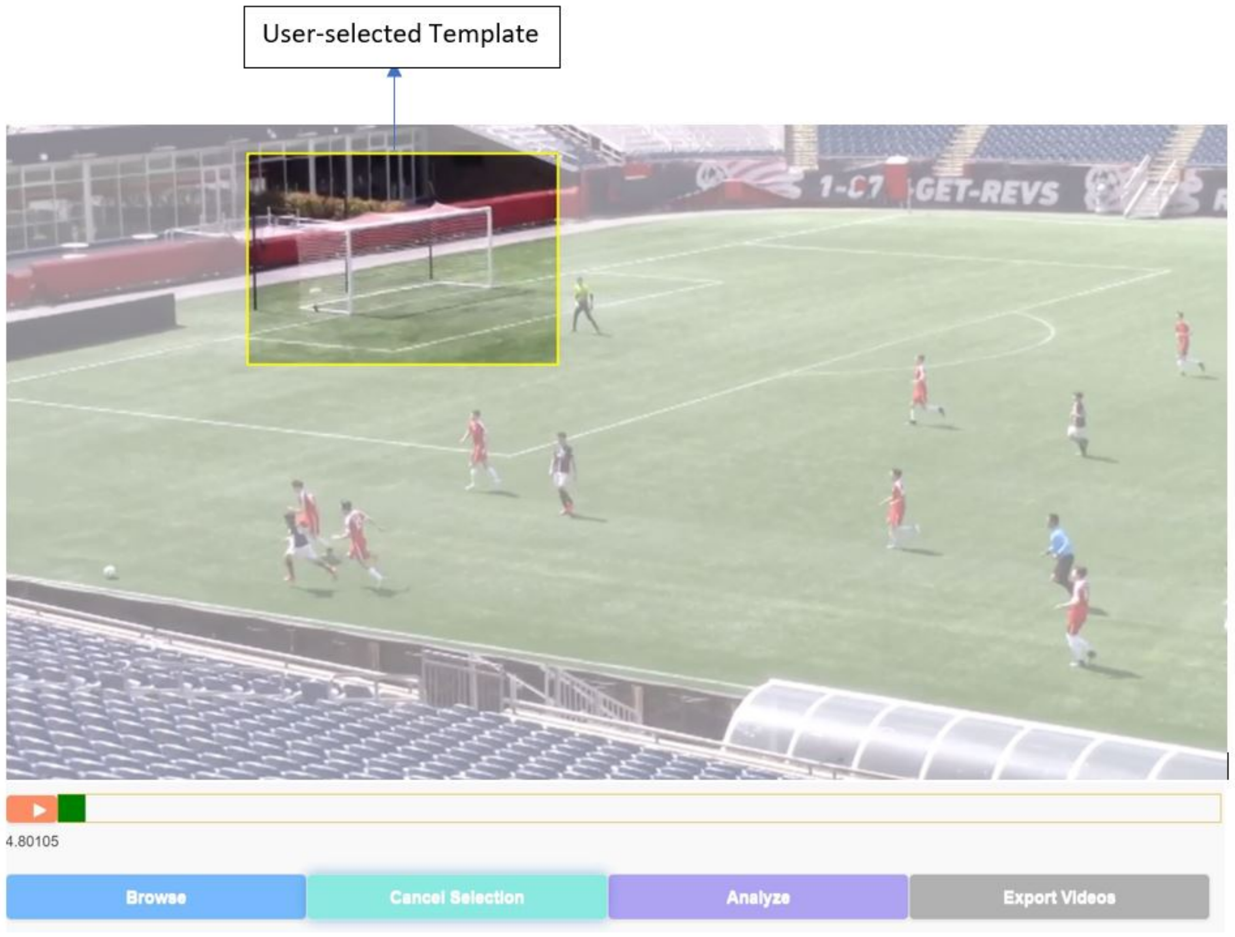


Figure 1.1: Interactive User-Interface (UI) for the existing system.

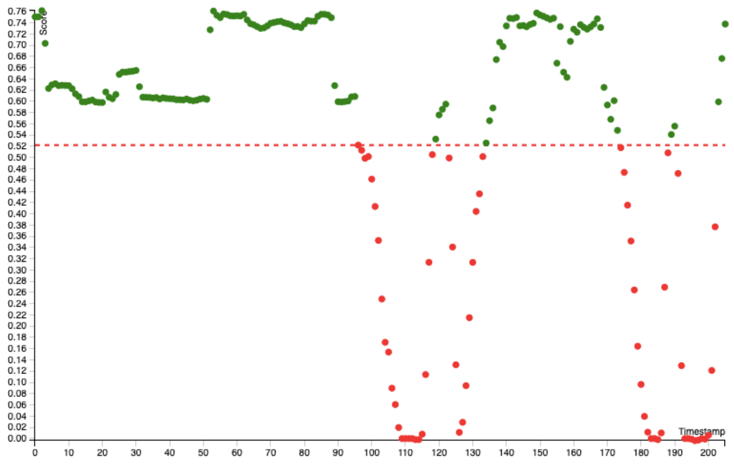


Figure 1.2: User-selected a threshold that separates the likely and unlikely frames to contain the template image.

unstable camera position causing rotated and scaled frames, lighting conditions, and environmental factors like rain. These limitation factors affect the computation of the matching value and make the video input difficult to deal with, creating anomalies in the resultant output video. It is thus necessary to study the factors affecting single-camera videos to overcome the existing limitations.

Since the objective of the existing approach was to analyze goalkeeper performance by extracting the goalkeeper moments, we continue to experiment on the same domain. We use a single-camera soccer video as our primary input and set goalkeeper performance analysis by overcoming the existing limitations as the primary motivation for this research. Once the proposed approaches are discussed and experimented with using the soccer videos, we also examine their potential scalability; we try to expand our area of research by implementing the performance analysis for other single-camera sports videos like hockey and basketball and note the observations and we will discuss them later in Chapter 6.

When we conducted this thesis research, we observed that the existing template-based template matching was not scale or rotation invariant. We investigated ways to overcome the transformation (scale and rotation) invariance and possibly certain other issues related to poor visibility due to lighting or weather conditions. As we were studying the expanded research area of the existing system, we aimed to answer one fundamental question: ”In what ways can we improve the existing system’s performance to allow users to more easily identify the video frames of their interest (i.e., the video segments that contains the goal area)?”. We also wanted to investigate if any available approaches can surpass the current approach regarding optimization (i.e., time or performance). Hence, the set of questions that I wanted to answer through the research presented in this thesis are enlisted as follows.

- RQ1: How can we improve efficiency and accuracy for analyzing multimedia data while allowing the users to handle the interactive interface and control the results?
- RQ2: What algorithms work well with the video analysis for videos with different conditions, like a rotated or zoomed frame or low lighting, etc., and provide optimal results? Is there an algorithm for template matching that helps achieve transformation (i.e., zooming and scaling) invariance and provides optimal results if we use an inconsistent varying quality single-camera video?
- RQ3: Are there enough datasets to train a predictive model for the single-camera video extraction? If not, is it possible to use supervised machine learning algorithms without the presence of an optimal dataset for single-camera videos?

The search for the answers to all these questions led us to a set of experimentation, findings, and conclusion.

1.3 Thesis Structure Overview

The requirement to overcome the limitation of the existing approach, as discussed in previous sections, and the rise of research questions have encouraged us to set milestones and organize our research. The overall thesis research is divided into five chapters. In the next chapter (i.e., Chapter 2), we discuss all the available resources, including the approaches, algorithms, and their functions related to this research, in detail. After discussing the working mechanism of all the algorithms, in Chapter 3, we discuss the full detailed planned work-flow of the research. The existing research and the methodologies used for the complete research is discussed in Chapter 4. We then started our qualitative analysis with surveys and user feedbacks and began experimenting on different supervised and unsupervised approaches, discussed in Chapter 5. We conclude a potential approach with a possible algorithm that has the potential to overcome some of the limitations (discussed in Section 1.2). We tested and reformed the proposed approach with the algorithm that gives the optimal results (in Chapter 6). In Chapter 7, we discussed the summary of the findings of this research as well as limitations of the proposed approach. We also suggested the potential future directions that one may undertake.

Chapter 2

Literature Review

Multimedia data analysis is challenging in many levels, and understanding and making use of existing resources and tools were the crucial steps toward achieving the goals of the research presented in this thesis. A significant development is visible in the field of multimedia data analysis in terms of using deep learning and AI advancements subsequently [14]. In this chapter, we will discuss the existing multimedia extraction tools, multimedia information retrieval tools and the algorithms relevant for this research in detail. We begin by discussing the background study for the existing multimedia tools in Section 2.1. Next, we discuss about the research on interactive multimedia editing tools and their interactive interfacing techniques. The discussion will be followed by the detailed explanation of the algorithms relevant to this research (Section 2.2).

2.1 Interactive Information Retrieval

One of the fundamental tasks while analyzing multimedia data is to extract relevant information from raw target data. While there are different approaches to information retrieval, this section focuses on the approaches and tools for *interactive* information retrieval. Rather than relying on the fully automated method of information retrieval, the approaches discussed in Section 2.1.1 are examples of systems that involve human interactions for multimedia data retrieval using different interaction styles.

In this section, we will discuss the existing multimedia data extraction tools that can be used for interactive editing. Study of the existing tools can provide an insight on the possibility of using different algorithms to extract optimal results for an interactive multimedia data analysis.

2.1.1 Multimedia Data Extraction for Interactive Editing

Image and video property information such as basic color correction properties (e.g., color channels, brightness, contrast, exposure, hue, saturation, opacity) are often available for viewing and manipulation in most popular commercial image and video editing softwares (e.g., Adobe Photoshop, Adobe Premier Pro, Apple Final Cut Pro). In addition to these basic types of image information, advances have been made, especially in the study area of computer vision that focuses on image feature extraction for such purposes as content-based image search and object recognition/detection [15, 16].

Image or video editing typically requires users commands in order to extract the desired output. One of the most common interaction techniques for video segmentation is by allowing the users to specify target objects and regions in an image by drawing or scribbling on one video frame and extrapolating to other frames. The scribbles can be single or multiple based on the approaches used for segmentations. Graphical Annotation Tool [17] allows users to select a video frame and draw a boundary around an object. In general, tracking is done by the temporal segmentation of the frames and generation of an object mask to initialize a video tracker. This tracker then invokes a tracking algorithm [18] based on the frame to frame update of a set of Gaussian distribution for both the foreground and the background pixels categorized by a Bayesian decision. This technique is robust to camera movements as the user-generated mask is used to initialize the object tracker. In the Geodesic Framework [19] tool, the users interact by using the mouse by scribbling on the target object with one color and around the object in another color. It helps the algorithm to segregate the background and the object. The system then performs the optimal, linear time computation of weighted geodesic distances to those scribbles to detect the object boundary and segment the object. This process is depicted in Figure 2.1.

Another similar tool is LiveCut [20], which implements a graph-cut optimization framework [21]. The system may extract many cues on the object like color and gradient, adjacent color relationships, spatiotemporal coherence, and motion. These cues that can differ alongside the frame are collected and are locally weighed to resolve them using graph-cut optimization. It then allows the user to correct the errors by adjusting the weights and learns to optimize its performance. Grabcut [22] is another graph-cut-based segmentation tool that allows its users to draw scribbles on the object (shown as the red lines in Figure 2.2) and the background (shown as the blue lines). The scribbles result in a rectangular box drawn around the object (shown as the green rectangle) to indicate the

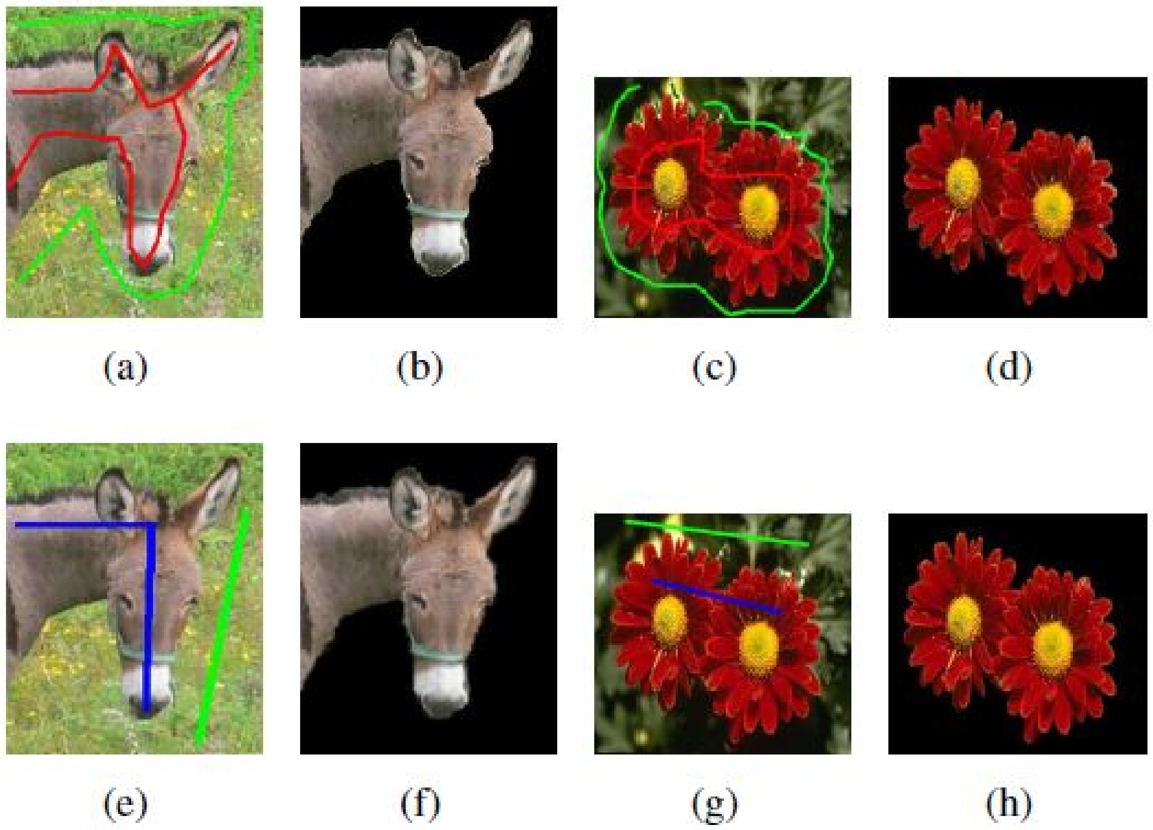


Figure 2.1: Figures (a)-(d) shows the user inputs as scribbles as used in systems. Figures (e)-(h) shows proposed approach leading to better results with less scribbles [19].

estimated foreground area (i.e., the user-selected area). It then slices the frame by grab cut operator after estimating the probable foreground region and rectifies the interested region by making sure the object lies in the green rectangle. Rather than scribbling over the

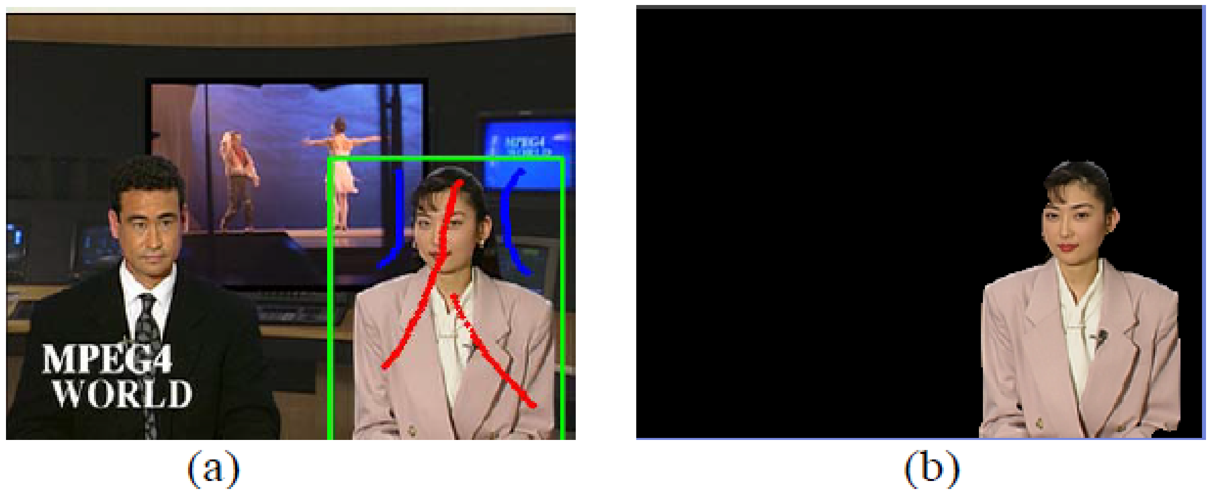


Figure 2.2: (a) User Scribbles on the object and boundary (b) segmented object [22]

.

surface and creating a boundary, an approach to video segmentation with only a few strokes [23]. It only requires a single disjoint scribble on the specified object. It uses a combination of motion from point trajectories and integrates a constraint to enforce color consistency by combining the user input with long-term motion cues. FOMTRACE [24] uses a delineation algorithm derived from Optimum-Path Forest (OPF) [25] to trace a fuzzy object model based on the input mask layer created by tracking the scribbles.

More recent approaches for object segmentation may also include machine-learning algorithms using artificial neural networks. One such tool, Deep Extreme Cut [26] works by adding an extra channel to the image in a convolutional neural network (CNN). The user specifies the accurate bounding box with the mouse clicks as the extreme endpoints of the objects in the frame. It initiates the segmentation process, which contains a centralized Gaussian as an input to the extra channel of the neural network that later segments the layer. In Mindcamera [27], the user draws a rough outline of an object, and the system uses alpha matting to allocate real-time foreground object extraction to return the scene images that consist of the potential objects. Mindcamera uses contour extraction to filter the backgrounds and Gradient Field HOG (GF-HOG) [28] to add spatial information to Bag of Visual Word (BoVW) as a description.

All these interactive video segmentation tools use similar user interactions despite being based on different approaches. These approaches are constantly evolving with advances in machine learning. A different approach to video segmentation is using a template-based template matching algorithm [12] which uses the template matching algorithm for relevant video scene and summary extraction. According to the rectangular frame selected by the user, the system extracts and matches the video frames with the input. If the video frame contains the template, it appends the frame into the output video.

After discussing some of the existing interactive multimedia tools and their base working mechanism, we learned the basic concept of multimedia data analysis in an interactive environment. The ever-evolving multimedia data analysis techniques over the years enlisted some of the possible applications of multimedia analysis and gave us a brief insight into the work that is yet to be done. We came across various applications for interactive multimedia data analysis which depicted the impact of users selection on the obtained results. The similarities between the existing tools and the existing research highlighted the relevance of user input in multimedia data analysis. It further shed light on possibility of the multimedia data analysis with different algorithms and approaches. This background study of the existing system emphasized the need of background study to understand the multimedia data analysis trends and also backed the idea of interactive interfaces in the analysis. We further proceed to study the existing research and to find the alternatives to produce optimal output.

In the next section we first discuss the existing template-based template matching approaches. We then continue to discuss the feature-based template matching apprroaches in Section 2.2.2. We then discuss the potential supervised algorithms in Section 2.2.3

2.2 Template matching

The use of template matching allows the system to search based on a predefined template because of which template matching algorithm is effective in building an interactive system. We study and discuss template matching approach as the base approach for the research. The overall working mechanism of template-based matching is explained in Figure 2.3. Here, template matching takes a reference area as input. This region is called

a template, and the algorithm runs to find the presence of that particular template in another image (called the target image) using different approaches and algorithms. Some of these approaches are template-based template matching and feature-based template matching. The values using the approaches are based on many metrics, for example, calculating maximum or minimum distance or finding relevance between the descriptors between template and the target image.

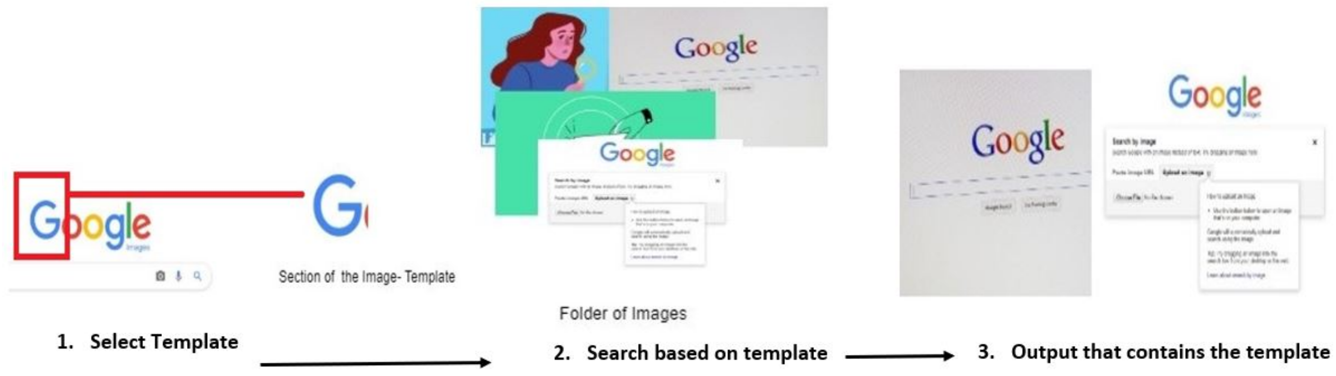

Section of the Image- Template
Folder of Images
1. Select Template
2. Search based on template
3. Output that contains the template

Figure 2.3: Working mechanism of template Matching algorithm in detail. Step-1: A part of the image is selected as a reference area or the template. Step-2: The system takes the reference area and searches for a matching area among a set of images (called as target images). Step-3: The backend algorithm matches the template and target image and exports the images containing the template.

As discussed in Chapter 1, some issues with template matching are background changes, detection of non-rigid transformations, and scale changes. The obtained results using template matching vary based on the approach that we use. There are a few common template matching approaches that we discuss in Sections 2.2.1 and 2.2.2. We also discuss the working mechanism of supervised learning approaches in Section 2.2.3.

2.2.1 Template-based Template matching

Template-based template matching takes in a user-fed input, a smaller part of an image known as a template, and then scans a target image pixel by pixel to locate the template in the image. The resultant image containing the template is based on Region of Interest (ROI). The template, also known as a patch, is placed over the image. The template and image overlap to calculate the numeric compatibility based on the similarity between the template and the image as shown in Figure 2.4.

The patch now moves and overlaps the region of the target on the right and slides over

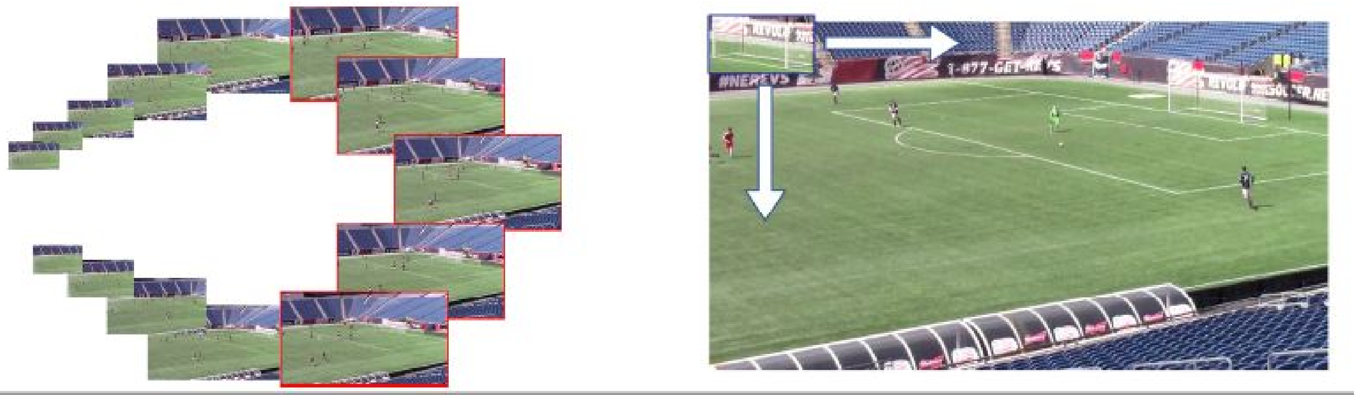


Figure 2.4: Template and target image mapping and likelihood value calculation [12].

each and every pixel of the target image calculating the value for that pixel until the patch reaches the right end. This approach is called the sliding window approach. The maximum value among the obtained values is assigned as the likelihood value of the frame. Template matching can be performed using two methods, grayscale-based matching or edge-based matching. For finding the similarity between the patch and image using grayscale matching, numerical computations need to be done based on correlation. There are three functions that we used in this research to calculate the distance metric between the template and the target image using template-based template matching. They are discussed in detail below.

Cross-correlation Function

The cross-correlation function compares a template with a target image for all the overlapped regions. When the template is placed over the target as shown in the Figure 2.5, the cross-correlation value is calculated at pixel P. This function matches the highly correlated points to each designated key point in the system. The distance value between the overlapped regions of the template and the target image for each pixel is calculated using sliding window approach for each designated video frame. Each correlation value is saved in the array of correlation values alongside the value for every pixel until the last pixel. While using the Correlation function, the maximum value is selected as the likelihood value for the respective frame.

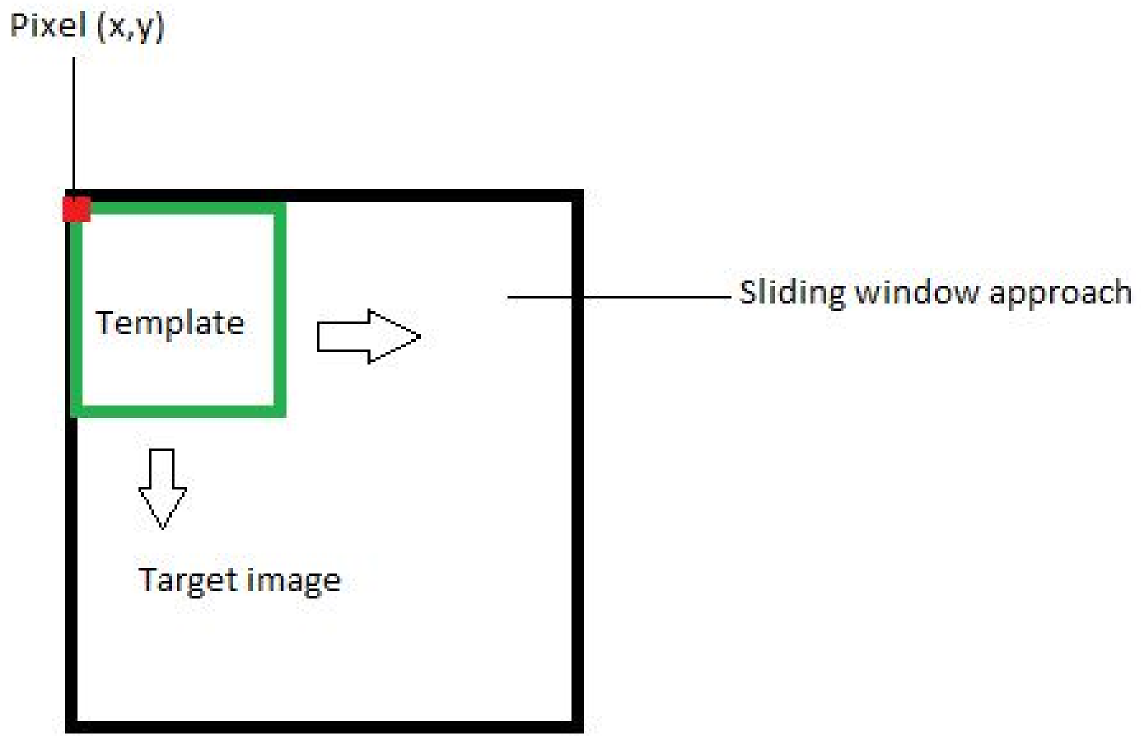


Figure 2.5: The template image overlapping the target image and the sliding window approach.

Let $C(x, y)$ be the cross-correlation of the two images at a pixel (x, y), where (x, y) are the coordinates of the pixel, $T(x, y)$ the pixel value of the target frame at (x, y), and $R(x, y)$ the pixel value of the reference area (or template) at (x, y), the metric is calculated by the following formula.

$$C(x, y) = \sum_{x',y'} \{T(x', y') \cdot R(x + x', y + y')\} \tag{2.1}$$

For every video frame, the values for $C(x, y)$ were obtained by using Equation 2.2 (also called as the normalized cross-correlation function) to mitigate any substantial values. This equation divides the obtained correlation value with the dot product of energy in the overlapped area and the energy of the template itself is then divided by the determinant of itself which gives us our final correlation value.

$$C(x,y) = \frac{\sum_{x',y'} \left\{T(x',y') \cdot R(x+x', y+y')\right\}}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} R(x+x',y+y')^2}} \tag{2.2}$$

We calculate $C(x, y)$ for all the pixels given by Equation 2.2, and then use the maximum value in $C(x, y)$ as the highest likelihood of the reference area matched in the target frame. We repeat this process for each reference area and calculate the overall likelihood of the reference areas appearing in the frame.

To keep the values obtained by Equation 2.2 consistent and to present the obtained results in an optimal way, we used the data normalization equation (Equation 2.3) to distribute the obtained values within a range of 0 to 1, 1 being the highest and 0 being the lowest likelihood of the respective frame containing the template. For X is the array of the obtained values and the X' is the normalized array, the normalized value for each X in the array is calculated using Equation 2.3 where i=0, 1,....., N.

$$X_i' = \frac{X_i - X(min)}{X(max) - X(min)} \tag{2.3}$$

Coefficient of Cross-correlation Function

The mean-shifted cross-correlation algorithm, also called Pearson's correlation coefficient, is a function of template-based template matching. The function uses the correlation coefficient as a measure of distance between the template and the target image. The obtained values for the correlation segregates the pixels into two categories, dark and light pixels. Unlike in cross-correlation, correlating the brighter patches is done by subtracting the mean before calculating the correlation.

Let $C(x, y)$ be the cross-correlation of the two images at a pixel (x, y), where (x, y) are the coordinates of the pixel, $T'(x, y)$ the coefficient of the pixel value of the target frame at (x, y) (where T' is the shifted value or the coefficient value obtained after subtracting the mean from the original value), and $R'(x, y)$ the coefficient of the pixel value of the reference area (or a template) at (x, y), the metric is calculated by the following formula.

$$C(x,y) = \sum_{x',y'} \{T'(x',y') \cdot R'(x+x', y+y')\} \tag{2.4}$$

We use the normalized coefficient of cross-correlation function to avoid any exponential values. In this step, we used the correlation of coefficient function displayed in Equation 2.5. This equation divides the obtained correlation value with the dot product of energy in the overlapped area and the energy of the template by the determinant of itself which gives us our final correlation value. The value indicates the likelihood of the frame containing the template. Furthermore, the obtained value distinguishes the likelihood of the frame containing the template and helps counter the observed disparities of the pixel values likely caused by different environmental conditions (e.g., weather, etc.)

$$C(x,y) = \frac{\sum_{x',y'} \{T'(x',y') \cdot R'(x+x',y+y')\}}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} R'(x+x',y+y')^2}} \tag{2.5}$$

To keep the values obtained by Equation 2.5 consistent and to mitigate substantial values, we used the data normalization equation (Equation 2.3) to distribute the obtained values within a range of 0 to 1, 1 being the highest and 0 being the lowest possibility of the respective frame containing the template.

For every video frame, the values for $C(x,y)$ was obtained by using the Equation.2.5 for every pixel in the video frame. The obtained values for both images are compared and the overlapping of both the template and target bright pixels gives you a determined similarity.

Sum of the Squared-differences Function

The SSD function calculates the sum of squared differences between the template and target image pixels. It uses a brute force approach to calculate the value for the difference for the pixel over a range of displacements and then finds the minimum sum of squared differences of the obtained value. The differential value of the overlapped image regions is used to match the template with the target instead of multiplied value, unlike in correlation. The sum of squared differences (SSD) is equivalent to the squared Euclidean distance, the distance function for the L2 norm when used for computation.

Let $C(x,y)$ be the differential value for the two images at a pixel (x,y), where (x,y) are the coordinates of the pixel, $T(x,y)$ the pixel value of the target frame at (x,y), and $R(x,y)$ the pixel value of the reference area (or a template) at (x,y), the metric is calculated by the following formula.

$$C(x,y) = \sum_{x',y'} \left\{ T(x',y') - R(x+x', y+y') \right\}^2 \tag{2.6}$$

Squaring the differences, instead of taking their absolute value, penalizes values that are further from what you expect. It makes the images more distant as the difference in value grows. It maps more to possible estimate as being way off, even if it's not actually that distant. We use the normalized SSD function shown in Equation 2.7 to avoid exponential values and to avoid inconsistencies in the values.

$$C(x,y) = \frac{\sum_{x',y'} \left\{ T'(x',y') \cdot R'(x+x', y+y') \right\}^2}{\sqrt{\sum_{x',y'} T(x',y')^2 \cdot \sum_{x',y'} R'(x+x',y+y')^2}} \tag{2.7}$$

For every video frame, the values for $C(x,y)$ was obtained by using the Equation.2.7 for every pixel in the video frame. The values obtained by SSD calculated the minimum values, as SSD calculates the darker patches from the detected image when compared to correlation functions that detects the lighter patches. To use the minimum values in the higher likelihood range for the output graph, we needed to pre-process the values before plotting graph. So to keep the values obtained by Equation 2.7 consistent with the other two functions and to mitigate substantial values, we first calculated the reflection of the values around the horizontal line at y=0.5 and then used the data normalization equation (Equation 2.3) to distribute the obtained values within a range of 0 to 1, 1 being the highest and 0 being the lowest possibility of the respective frame containing the template. The normalized final values are then plotted in the graph. Normalization of obtained values distinguishes the likely and unlikely frame distinctly and helps counter the observed disparities of the pixel values likely caused by different environmental conditions (e.g., weather, etc.).

2.2.2 Feature-based Template Matching

A feature in an image can represent certain aspects like the objects, points, or any specific regions of the objects in the image. In any template matching approach, the first image is the user-selected template image that has the desired object to be matched. There is a second image called target image that contains the same object but that can be in a different orientation. The first step in any feature-based template matching algorithms is to allocate the keypoints and define their respective descriptors for both the template and the target images. In template matching, the system needs to recognize the similarities between the input and the resultant image. Feature matching is a computational technique that creates a reference from the template based on which the resultant object is detected.

To overcome the observed shortcomings of the current approach, we studied feature-based template matching. The two matching approaches used in feature-based template matching for this thesis dissertation are Brute Force(BF)-based matching and Fastest Nearest Neighbour(FLANN)-based matching.

Feature detectors for template matching

There are five feature detecting algorithms, among which three are used for this research using various distance measures to match the features. The five algorithms are:

1. FAST (Features from Accelerated Segment Test)

2. BRIEF (Binary Robust Independent Elementary Features)
3. ORB (Oriented FAST and Rotated BRIEF)
4. SIFT (Scale Invariant Feature Transform)
5. SURF (Speeded Up Robust Feature)

The algorithms used for this research study are Oriented FAST and rotated BRIEF (ORB), scale-invariant feature transform (SIFT) and SURF (Speeded Up Robust Feature).

- **Oriented FAST and Rotated BRIEF (ORB)** is the cost free feature extraction method, and it is available to the public contrary to the patent SIFT and SURF methods, all implemented in the OpenCV library. ORB contributes to detecting the keypoints in a fast and efficient manner.

  When a template is run through an ORB object, initially, the goal is to pinpoint the keypoints as fast as possible. For this, the ORB objects go through two processes. First, it uses FAST algorithm [29] to extract the features of the input image. As the template runs through the ORB, it creates numerous FAST arrays, each consisting of 16 pixels p_1, p_2, ..., p_n with a center p. Each array is then sorted into three categories determined by the pixel value. The categories are brighter than p, darker than p, similar to p. If eight or more pixels have a comparatively higher or lower value than p(i.e., brighter or darker), then p is selected as a keypoint in the image. Similarly, the remaining pixels are compared and keypoints are detected.

  However, the points detected using a FAST algorithm lack the multi-scaled and oriented features. The ORB algorithm uses a multi-scale image pyramid to overcome this, representing the down-sampled version of the image at various levels and resolutions. When passed through ORB, the keypoints are plotted over all the pyramid layers, due to which ORB tends to partially scale-invariant. After identifying the keypoints, it then assigns orientation to those keypoints based on which side they are facing, i.e., left or right. This orientation allocation is done by detecting intensity change using the ORB intensity centroid. Once the centroid is calculated, and orientation is determined, the keypoints can now be turned to obtain rotation invariance. The moments are computed with x and y within a circular region of radius r, where r is the size of the patch, to improve the rotation.

Now for descriptors, ORB uses Binary Robust Independent Elementary Feature (BRIEF) descriptors. Since BRIEF performs poorly with rotation, ORB steers BRIEF according to the orientation of keypoints. First, for any feature set of n binary tests at the location (x_i, y_i), ORB defines a 2 by n matrix, S, which contains the coordinates of these pixels. Then, using the patch orientation, θ, its rotation matrix is found. The S is then rotated to get steered (rotated) version S_θ. Finally, ORB discretizes the angle to increments of $(2\pi)/30$ (12 degrees) and constructs a lookup table of precomputed BRIEF patterns. As long as the keypoint orientation θ is consistent across views, the correct set of points S_θ will be used to compute its descriptor.

BRIEF has an important property that each bit feature has a large variance and a mean near 0.5. But once it is oriented along keypoint direction, it loses this property and becomes more distributed. Moreover, high variance makes a feature more discriminative since it responds differentially to inputs. Therefore, another desirable property is to have the uncorrelated tests. Since then, each test will contribute to the result. To resolve all these, ORB runs a greedy search among all possible binary tests to find the ones that have both high variance and means close to 0.5, as well as being uncorrelated. The result is called rotated-BRIEF (rBRIEF).

BRIEF, on the other hand, starts soothing images using a Gaussian kernel to avoid high-frequency noise. It takes the neighboring pixels around the keypoint in a random pair (first pixel in the pair selected by a standard deviation or spread of sigma around the keypoint and second pixel by multiplying sigma by half). If the first pixel is brighter than the second, then the binary feature vector is assigned a 1 or 0 that is added as a descriptor for the keypoint. It is repeated 128 times for each keypoint, and a short feature vector is created. But since BRIEF is not rotation invariant, ORB initiates Rotation-aware BRIEF to allow it to be rotation invariant. Overall, the BRIEF takes all the keypoints allocated by the FAST algorithm and converts them to a binary feature vector, representing the points as an object.

The descriptors are the numerical values that define the keypoints. The ORB object's keypoints and descriptors are allocated using the ORB algorithm then the distance is calculated using the Brute Force matcher and FLANN-based matcher to determine the matching keypoint pairs of the template and the second image. The features also match the rotated and scaled target images.

- **Scaled Invariant Feature Transform (SIFT)** SIFT is a transformation invariant algorithm and is also patented to include in a Non-free module in OpenCV only available for study and research purposes [30]. SIFT provides distinctiveness to each feature in an image where they can be matched with a larger dataset. SIFT starts by localizing the keypoints in an image by pinpointing the image's distinct features. Since SIFT is scale invariant, this process is done in a scale-space peak selection. The scale-space tries to replicate the concept of the multi-scale nature of objects in a digital image by dividing them into different scale-space octaves. The number of octaves in a scale space is dependent on the image size of the original image, and this octave's size is half of the previous octave.

The scale-space of an image is defined by a function $F(x, y, \sigma)$ where F is produced by convolution of a Gaussian kernel at different octave scales as defined in Equation 2.8. The convolution of the Gaussian Kernel is also called blurring of the image in a multi-scale space. The images in the different octaves are blurred when the Gaussian operator $G(x, y, \sigma)$ is applied to each pixel multiplied by I, intensity at location (x,y).

$$F(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{2.8}$$

The function F then results in a blurred image in each pixel where σ is a scaling parameter. For this process, the blurring of an image is directly proportional to σ. The blurred images are then subtracted using σ and $k\sigma$ and used to generate the set of images called Difference of Gaussians (DoG), where k is a scale space constant. This is done to localize the interesting regions which will help find keypoints in the image which can be observed in the Figure 2.6. A similar keypoint localization method from

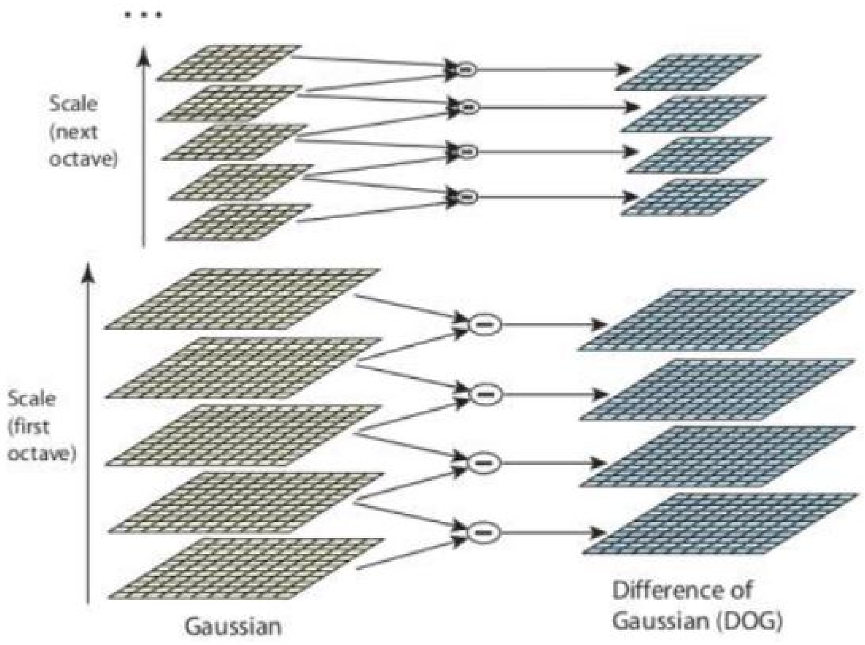


Figure 2.6: Difference of Gaussian Kernel(DoG) [1].

ORB is then used to represent the main points. This process is done in a multi-scale, i.e., compared with one lower and one upper-scale octave of the image itself. Instead of comparing each pixel value with the regular 8 surrounding the pixel, SIFT uses eight surrounding pixels, nine neighboring upper-scale, and nine-neighboring lower scale pixels of the same image in multi-scale. While comparing each pixel with 26 pixels, potential keypoints are then generated and pointed in the image. SIFT also uses the Harris Corner Detector [31] to eliminate the outliers among the keypoints and localize keypoints. The outlier may consist of the points lying along the edge and lacking in color, density, or contrast which are not useful for feature detection. The DoG is more responsive to the edges and flats, so edges should be removed from the detected keypoints.

The multi-scale blurring and keypoint localization ensures the scale invariance. In addition to this, to add the rotation invariance, an orientation should be assigned to each detected keypoints. The gradient magnitude and direction, both are calculated in the region around the keypoints depending on the scale. Once calculated for all the keypoints, a histogram is generated representing all the values. The peaks of the histograms (i.e., above 80 percentile), are used to calculate the orientation. Each keypoint now has a scale, location, and orientation which needs highly distinctive descriptors. A 16x16 window surrounding the keypoint is taken and split into 16, 4x4 sub-blocks for which an eight bin histogram is created. The feature vector uses the 4x8 direction from the bin values to form a keypoint descriptor. After obtaining the keypoints and the descriptors, they are matched based on the nearest neighbor or closest match.

- **Speeded Up Robust Feature (SURF)** SURF is a patented feature detection algorithm. SURF, like its other counterparts, ORB and SIFT, works in two steps, keypoint detection, and its descriptor extraction.

  For keypoint localization, unlike in ORB and SIFT, SURF uses Hessian matrix-based keypoints. It depends on the matrix's determinant value to define the keypoints location and scale. To adapt to any scale, the image is filtered using a Gaussian kernel σ. For a given point $X = (x, y)$,L being a second order derivative at a point, the Hessian matrix $H(x, \sigma)$ in X at scale σ is defined as:

$$H(x,\sigma) = \begin{bmatrix} L_{xx}(x,\sigma) & L_{xy}(x,\sigma) \\ L_{xy}(x,\sigma) & L_{yy}(x,\sigma) \end{bmatrix} \tag{2.9}$$

  We need to apply convolution with Gaussian kernel, then second-order derivative to calculate the determinant of the Hessian matrix (calculated by Equation 2.9). SURF pushes the approximation, both convolution, and second-order derivative even further with box filters. These approximate second-order Gaussian derivatives can be evaluated at a very low computational cost using integral images. These derivatives are also evaluated irrespective of their size. These are some reasons that make SURF a little fast. SURF uses the image pyramids to represent the scale space. The scale space is analyzed by up-scaling the filter size instead of iteratively reducing the scale size. It is because of the use of box filters and integral images.

To extract the respective descriptors, SURF calculates Haar-wavelet responses [32] in x and y-direction for the current scale. The Haar-wavelet is a sequence of rescaled square-shaped functions that form a wavelet family or basis. For scale-based filtering at a higher scale, integral images are used again when the wavelet size increases. We calculate all wavelet responses for both the x and y directions and then change the orientation and re-calculate until we reach the maximum value for the orientation. The maximum value is the orientation for the respective keypoint.

The algorithm then centres the keypoints in a square region for the calculated orientation. The region is further divided into smaller square sub-regions, and the feature for each sub-region is computed. The features are computed in both the directions as dx and dy and are the first feature vectors. It is followed by calculating the sum of the absolute value of responses $|dx|$ and $|dy|$. Hence, all the subregions have a four-dimensional descriptor vector with a total length of 64-D, unlike in SIFT with the descriptor vector of length 128-D vector, making SIFT slower than SURF.

Template matching approaches for feature based templates

There are two basic matchers to use feature-based template matching. Brute Force(BF) based matcher is the conventional method that directly uses the distance metrics to localize the keypoint and to match the frame [33]. On the other hand, the FLANN-based approach filters the good matches with the nearest neighbour ratio-filter to optimize the results. Both the approaches are discussed in detail below.

BF-based Matcher Brute Force matching is a feature-based feature matching approach that takes the descriptor of the feature in the template image and matches it with the closest similar feature in the second set of images. Initially, we take two-image sets, the first one is the template, and the second one is the image that contains the template in a different photographic condition. Brute force matching defines the keypoints in the template image and a descriptor, a numeric value given to each keypoints.

The features of the second set of image is also determined by defining its keypoints and descriptor by comparison of the pixel density. If the template has an array of keypoints k_1 = i_1, i_2, i_3, i_4, ..., i_n and the second image has an array of keypoints k_2=j_1, j_2, j_3, j_4,....., j_m, for every i in k_1,the distance between i and all the points in K_2 are calculated (i.e. the distance between i_1 and j_m^1 will be calculated as an array of distance d = d_1, d_2, d_3, ..., d_o).

The keypoint with the shortest distance from d will be the j_1 pair for the first keyponts i_1. The same process will follow for all n keypoints in k_1 until we have a set of final keypoints k = (i_1, j_1),(i_2, j_2),(i_3, j_3),.....,(i_n, j_n). In general, the keypoint from the second image is only accepted if it is closest to the keypoint from the first image. The final pair of keypoints are matched from the template to the other image to identify the presence of the template in the second image.

The formula for calculating distance is determined by the function used for the process. For example, the NORM_L2 function calculates the Euclidean distance between the points, whereas the NORM_L1 function calculates the distances in each dimension. The use of each function shows a variation in the detection of the features.

FLANN-based matcher Fast Library for Approximate Nearest Neighbors or FLANN is an optimized algorithm that uses the FAST (Features from Accelerated and Segments Test) library for locating the keypoints as the features of the image for high dimensional features in large datasets. It requires more computational time but is efficient than the BF-based matcher for larger datasets and is light and scale-invariant for most of the feature-based algorithms it uses. In FLANN-based matcher, two dictionaries in K-Dimensional tree need to be passed for deciding the algorithm that can be used for matching. The obtained distance between the template and target keypoint is compared with its neighbors using a ratio filter which eliminates the false matches (if the distance is longer than the filter ratio). Then, the calculated shortest distance between the keypoints allocates the nearest neighbors for all the keypoints, amongst which the best ones are chosen as a good match based on the algorithm used.

FLANN-based matcher uses the different distance algorithms to match the descriptors.The first dictionary is the algorithm passed.

The second dictionary is the search parameters that specify the number of times the distance function is traversed recursively to obtain precision. We use the K Nearest Neighbor algorithms to match the descriptors from the template to the target. The number of times the distance algorithm is traversed, it increases the value precision for ORB, SIFT, and SURF.

After the distance between the keypoints is calculated, the ratio filter is used to remove false matches and possible outliers. For example, if the ratio filter is 80/20, it means that a keypoint and its neighbor is matched only if the distance between them is less than 4 times the keypoint. If the distance is larger than that value, the keypoints are discarded as false matches. The matched keypoints are called good matches.

2.2.3 Supervised Learning Approaches

Supervised learning can also be used for identifying video frames with target objects. It creates and trains a model to classify an object based on the training data. The training data are labeled and fed into the system to train the supervised learning model. Among many approaches to supervised learning, some of the algorithms that we considered to experiment for this thesis dissertation are:

- Convolutional Neural Network (CNN)
- You Only Look Once (YOLO)

Convolutional Neural Networks (CNN)

Convolutional Neural Networks is a deep neural network algorithm that creates a model that can detect an object in an image and can classify it based on the models' training labels. CNN's are based on working patterns of neurons in the human body and their data transmission technique, inspired by the organization of the visual cortex. There is very little pre-processing in CNN as these networks are engineered to learn the characteristics of the image on their own. A Convolutional Neural Network or a ConvNet works as a multilayer deep learning model that initiates by reducing an image into an easier form for processing while preserving its critical features.

The base layer is followed by the convolutional layer that acts as a kernel or a matrix acting as a filter. The layers are shown in the Figure 2.7. After that, the box filter moves through its right and then down until the whole image is traversed. This extract was highlighting features of the images such as edge or color depth. Finally, the image is passed through the pooling layer to reduce the computational complexity. It also decreases the spatial size of the results from the previous layer. There are two types of pooling techniques, max pooling and average pooling. Max pooling performs better than average pooling for our purposes because it gives the maximum value within the portion of the image and discards the noisy activations alongside the de-noising process. On the other hand, average pooling provides the average of all values within the portion of the image overlapped by the kernel and undergoes dimensionality reduction for de-noising.

The output is flattened and fed to a regular Neural Network for classification, called a full-connected layer. Once we train the CNN model with labeled data containing a fixed number of classes, we can feed input data to the model for classifying the input into a predefined class. There are various architectures of the neural networks. Some modified CNN algorithms worked on this research project are discussed below.

Faster Region-based CNN (Faster RCNN) RCNN is a region-based bounding-box selection algorithm for a faster CNN approach [35]. Choosing desired region method is the initial step in the RCNN approach. After that, RCNN is used to train the model in a step-by-step approach. For example, using a bounding box to select regions in an image to pass it through the convolutional layers is a proposed method in RCNN. Figure 2.8 shows the workflow of an RCNN model. RCNN can also be accomplished using a brute force sliding window approach, which provides fewer regions than the bounding box region

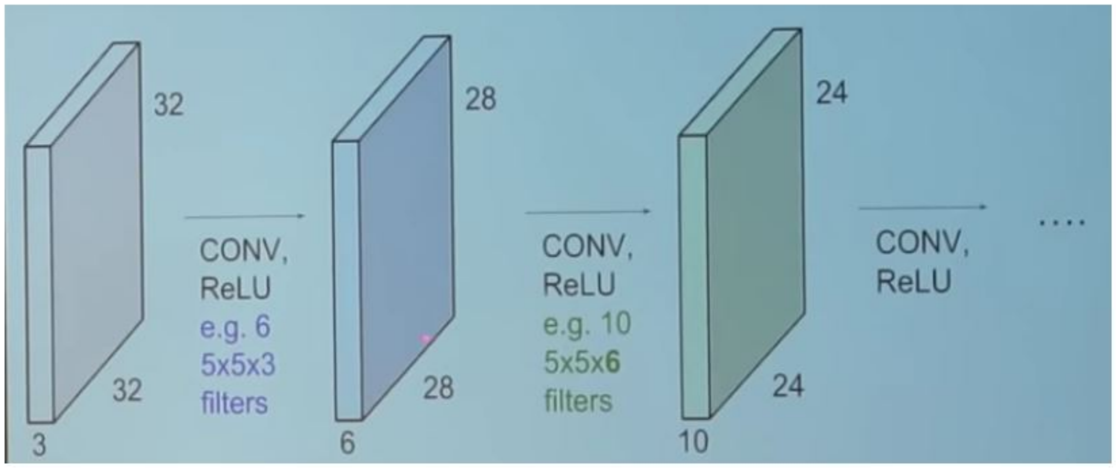


Figure 2.7: Layers in Convolutional Neural Networks [34].

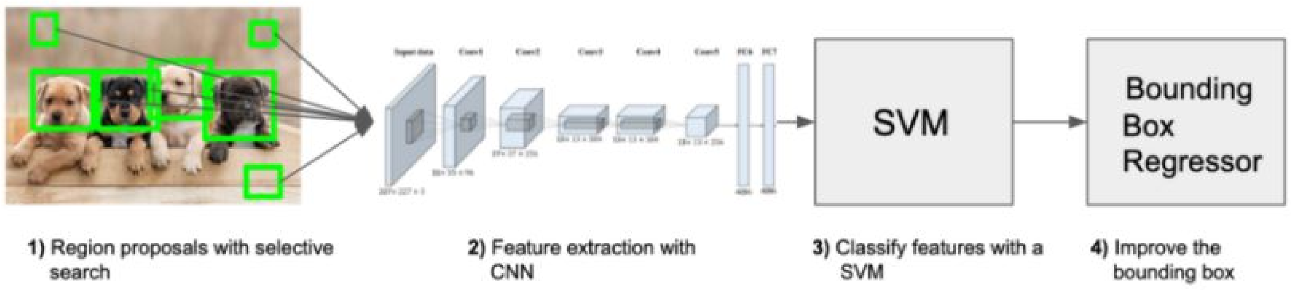

SVM
Bounding Box Regressor
1) Region proposals with selective search
2) Feature extraction with CNN
3) Classify features with a SVM
4) Improve the bounding box

Figure 2.8: Steps in an RCNN model [2].

detection. The regional proposal is then used to create a corresponding vector using CNN representing the same image in a smaller dimension. The vectors are classified and used to identify the class of objects that the respective vector represents. The Support Vector Machine (SVM) [36] classifier is used for each object class and generates the confidence score for each vector. This resultant confidence represents the confidence of the vector belonging to a particular class. All the resultant confidence and the classification of the classes are then brought together as a single image using greedy-non maximum suppression [37].

YOLO- You Only Look Once

YOLO was introduced in ”You Only Look Once: Unified, Real-Time Object Detection” as a research topic presented in 2015 [38]. Classification can be done with single images classified and labeled as a single object, but when it comes to many objects in a single image, it isn’t easy to classify different objects in a single image. YOLO introduced this approach for multiple object classification based on the location of an object in an image using CNN for object detection in real-time. The algorithm initiates a single neural network to the full image and categorizes the image into different regions. All the regions provide the bounding boxes that carry the probability of holding an object weighed by predicted probabilities. Despite training on single object labeled images, YOLO classifies multiple objects on the testing image based on the bounding box classification from the training data. It uses a single network for the entire process. It is faster than CNN, provides a general representation of objects, and outperforms other object detection methods [39]. The available labeled datasets can train the model and run on real-time images or videos.

The YOLO V3 architecture (as shown in Figure 2.9) is called darknet-53 and is inspired by ResNet [40] and Feature-Pyramid Network (FPN) [41].

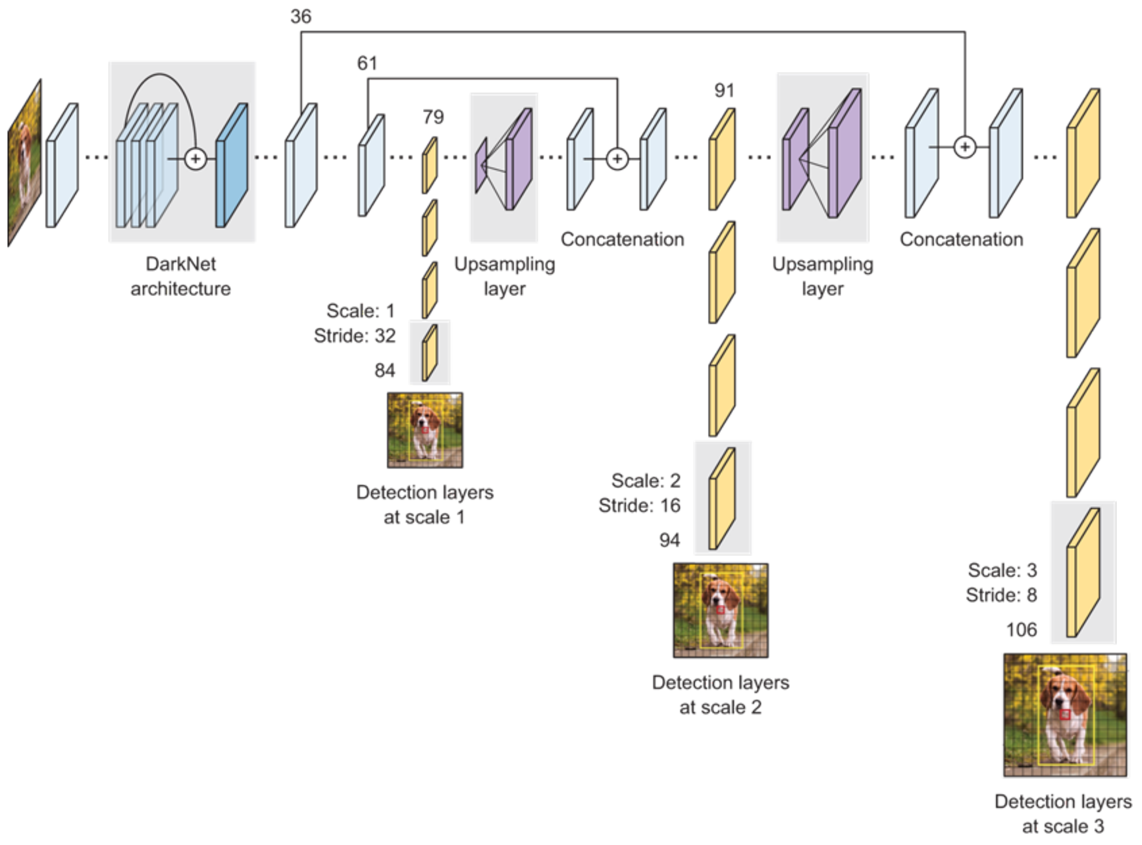


Figure 2.9: High level architecture diagram for YOLO [42].

2.3 Summary

In this chapter, we discussed the approaches and algorithms studied for this research. We started by discussing various available approaches of template-based template matching algorithm. All the algorithms and approaches that were discussed (in Section 2.2) will be experimented on to finalize a robust algorithm that can overcome certain limitations of the existing template-based template matching approach. The background studies helped us to understand the working mechanism of all the algorithms. Understanding the relevant approaches also gives an insight on the key elements in the existing approach that can be focused using alternative algorithms for optimal results.

In Chapter 3, we ill discuss the methodologies we used for this research. We then laid out a careful step-by-step plan to proceed towards our objective to overcome the limitations of template-based template matching.

Chapter 3

Research Methodologies

This chapter outlines the research methodology for the current study. The standard User-Centred Design framework is loosely followed in this study. The main focus is on investigating the back-end video processing algorithms to address issues observed in the template-based template matching approaches. In this chapter, we will lay out milestones for the research that will investigate and discuss the available back-end algorithms to overcome the limitations of the current approach.

The overview of the problem domain along with the core research questions is discussed first. The details of the research methodology is discussed in the later sections. The overall methodology for this study can be broken down in detail as follows:

- The detailed background study and the motivation of the research is discussed in Chapter 3.
  - The discussion of the scope of our study and the objectives of this research in detail in Section 3.1.

  - The background/original idea of this research and the relevant tools are discussed in Section 3.1.1.
  - We concluded this chapter by discussing the methodologies selected for the overall study in Section 3.2.

- A detailed explanation of our existing video anlysis tool and the observations are discussed in Chapter 4.
  - The research questions on the existing template-based template matching approach are discussed in Section 4.1.

- Chapter 5 describes a series of the preliminary investigations that we conducted.
  - The details on the survey are listed in Section 5.1. This survey allowed us to understand the user requirements necessary to set the goals while designing an appropriate prototype tool for the research.
  - The existing tool was experimented on different types of supervised and unsupervised algorithms in Section 5.3.
  - A prototype tool was designed and experimented. A possible solution was proposed to the existing limitations in Section 5.3.2. We also proposed a feasible solution to these research questions based on our observation.

- Chapter 6 describes the results with the observations.
  - The parameters in the feature-based template matching approaches that could be fine-tuned for optimal results are discussed in Section 6.2.1.

  - A fine-tuned algorithm that gives optimal results while overcoming the limitations of the existing tool was finalized in Section 6.3.
  - A user study to evaluate the effectiveness of the proposed tool, and the approach was conducted in Section 6.4.

- Th observed limitations and insights on the future work is discussed in detail in Chapter 7.

3.1 Problem Domain

The existing template-based template matching used a single-camera soccer video to analyze goalkeeper's performance [12]. We further analyzed and observed that under a certain variations in the video quality, like the illuminance or the transformation of the video frames (scaling, rotation and so on), the current template-based templating matching does not provide optimal results. It is mainly because the conventional template-based template matching approaches are not rotation or transformation invariant. The limitations observed while using the template-based template matching approach led us to explore and narrow down the problem domain for this research. Algorithms like Fast screening algorithm for rotation invariant template matching [43] and FRoTeMa: Fast and Robust Template Matching [44]; present the possibility of using template-based template matching to achieve transformation invariance. It motivates us to study similar computational algorithms that can help us achieve the transformation invariance for single camera video.

In many cases, one non-professional person is responsible for capturing most single-camera videos, often with their mobile phones. One such example is a soccer video captured by a coach or a parent to analyze the performance of their players. Such videos are often captured by their mobile phones likely without a tripod and they can often lack the video quality with uneven illuminance. The resultant video can be shaky and can have an inconsistent quality throughout the video. This research is a search for an optimal algorithm to be used in an interactive tool for such inconsistent single-camera videos that we will use for the video scene extraction. The target audience can be anyone capturing single-camera videos and extracting certain scenes based on their input from a single-camera videos. The initial idea was to use supervised learning to achieve rotation and transformation invariance for single-camera videos as they help construct a predictive model based on its training [45]. The lack of databases that can be used with our existing interactive approach and user-interactive interfacing motivated us to study the possible alternative approaches.

3.1.1 Template Matching

We need to be aware of the relevant ongoing studies to understand the problem domain of our research. This study arose from the research questions obtained after analyzing the limitations of the template-based template matching approach. We studied the working mechanism of the tool based on the template-based template matching in detail to understand these limitations, which built the foundation for the preliminary study. A thorough background study of similar interactive tools enabled us to to analyze the significance of interactive multimedia tools over automated ones as discussed in Section

2.1.1. For this we reviewed academic papers and studied different tools used in multimedia data analysis to compare our existing approach with the other similar tools.

The conventional template matching algorithm is simple and uses three simple functions for calculating the distance measure. The distance values can be thought as the likelihood values calculated between the template and the target image that indicates whether the target image contains the template. The functions for calculating the likelihood value using template-based template matching are:

- Cross-correlation function
- Mean-shifted Cross-correlation function
- Sum of Squared Differences function

The general definitions of these functions are provided in Section 2.2.1. We investigated all the three functions on a manually rotated and scaled single-camera soccer video in Chapter 5, and we provided the dicussions of the observations in detail.

3.2 Methodologies Selected

We started the preliminary study with a qualitative study that solidified our research objectives and problem domain. We chose a qualitative study as the main method of the study, especially for preliminary investigations, because it incorporates human experience in real-world scenarios. The feedback we receive from real-world scenarios boosts the research's motive, and in our case, users are an integral part of our research. In an interactive interface where users are in control, the results are directly or indirectly dependent on the input given by the user.

The first step to our qualitative analysis is discussed in detail in Section 5.1. We used an online questionnaire for the qualitative study and recruited the participants via online medium due to various unavoidable circumstances including the Covid related restrictions that voided our possibilities to meet them in person. While we have investigated and collected relevant statistical data, the obtained data were analyzed and used as descriptive data to understand and evaluate any possible new approaches. We concluded that users are interested in being in control in an interactive video analysis tool. The analysis from the survey helped shape our research questions and solidify our objectives. Based on the preliminary analysis, we created a prototype tool and experimented on various subject-relevant existing and new approaches to overcome the existing limitations. The obtained results using different approaches and algorithms were compared and presented to the users for the final qualitative study. The detail discussion occurs in Chapter 6.

The final study (as discussed in Section 6.4) was conducted to evaluate the approaches from the human perspective as its sole purpose is to be used by humans for applications like video extraction, annotation, dataset production, etc. in a user-interactive environment. Here again, we used the online questionnaires (with a series of questions and video outputs as multiple choices) to reach out to the participants and we collected their responses. The participants response enabled us to analyze qualitative feedback on the proposed tool, whereas the statistical data was summarized and discussed in detail in Section 6.4. The user responses helped us conduct a descriptive analysis from the feedbacks collected in order to compare the obtained results and to draw a conclusion.

Chapter 4

Overview of the main project

The existing research used template-based template matching to extract the goalkeeper moments for single-camera soccer videos based on the user-selected template [12]. The existing approach has an interactive interface that allows users to upload the single-camera video they want to analyze. The user selects the template image with a mouse selection and that template is the reference area to extract relevant frames from the input video. The template is matched with each target video frame using the cross-correlation which gives the correlation value as the likelihood. The obtained likelihood values will then be displayed in a scatter graph as shown in Figure 4.3.

This section discusses the current approach and the detailed working mechanism of the template-based template matching. We then discuss the limitations and possible modifications that can possibly overcome these limitations with the questions discussed in Section 4.1. The working mechanism of the existing template matching approach is shown as a flowchart in Figure 4.1. Our current approach works in five basic steps in the current

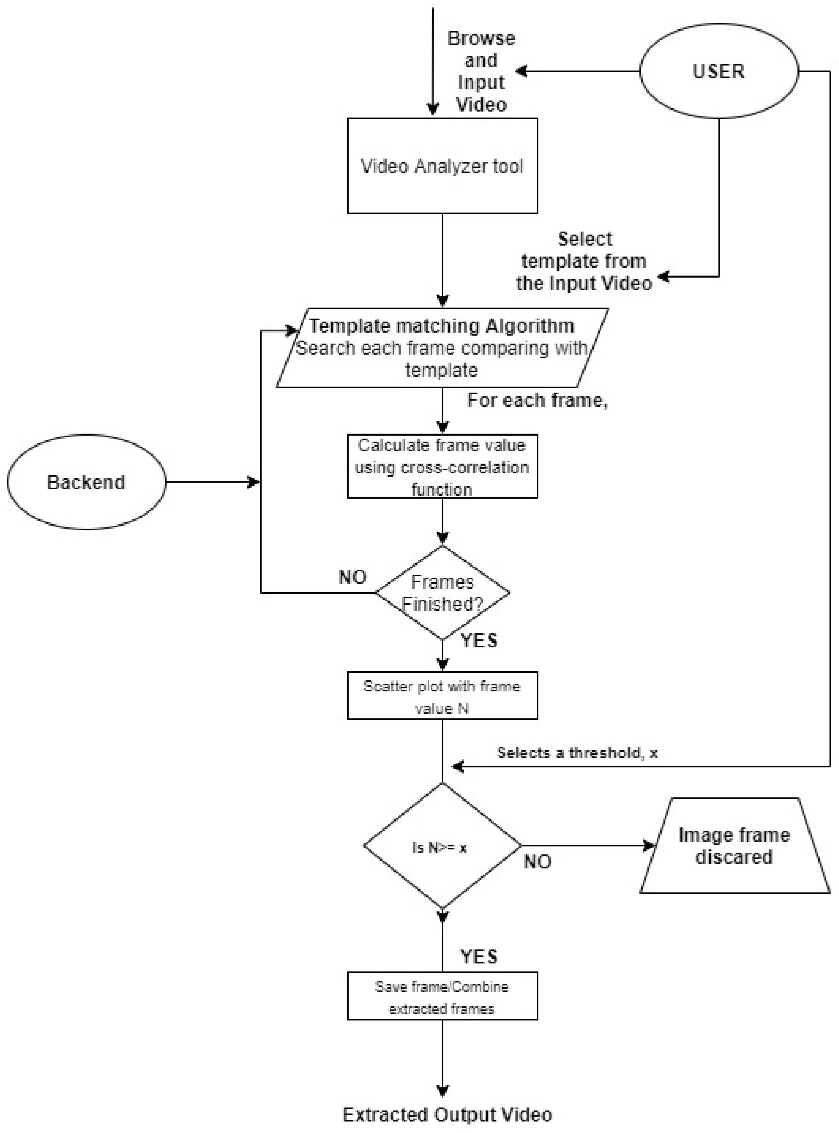

Browse and Input Video
USER
Video Analyzer tool
Select template from the Input Video
Template matching Algorithm Search each frame comparing with template
For each frame,
Backend
Calculate frame value using cross-correlation function
NO
Frames Finished?
YES
Scatter plot with frame value N
Selects a threshold, x
Is N>= x
NO
Image frame discared
YES
Save frame/Combine extracted frames
Extracted Output Video

Figure 4.1: The workflow of template-based template matching approach.

system when the users use the interactive system.

1. The user uploads a video. The user selects the frame from which they want to select a template. A template is a rectangular reference area chosen by a user as shown in Figure 4.2. In this step, the rectangular template is fed to the system as a reference. The uploaded video contains many video frames, called the target frames that will be compared with the template one by one.

2. The template is labeled as an $N \times N$ reference area by the system. The reference area is overlapped with the first $N \times N$ pixels of the target video frame for comparisons.

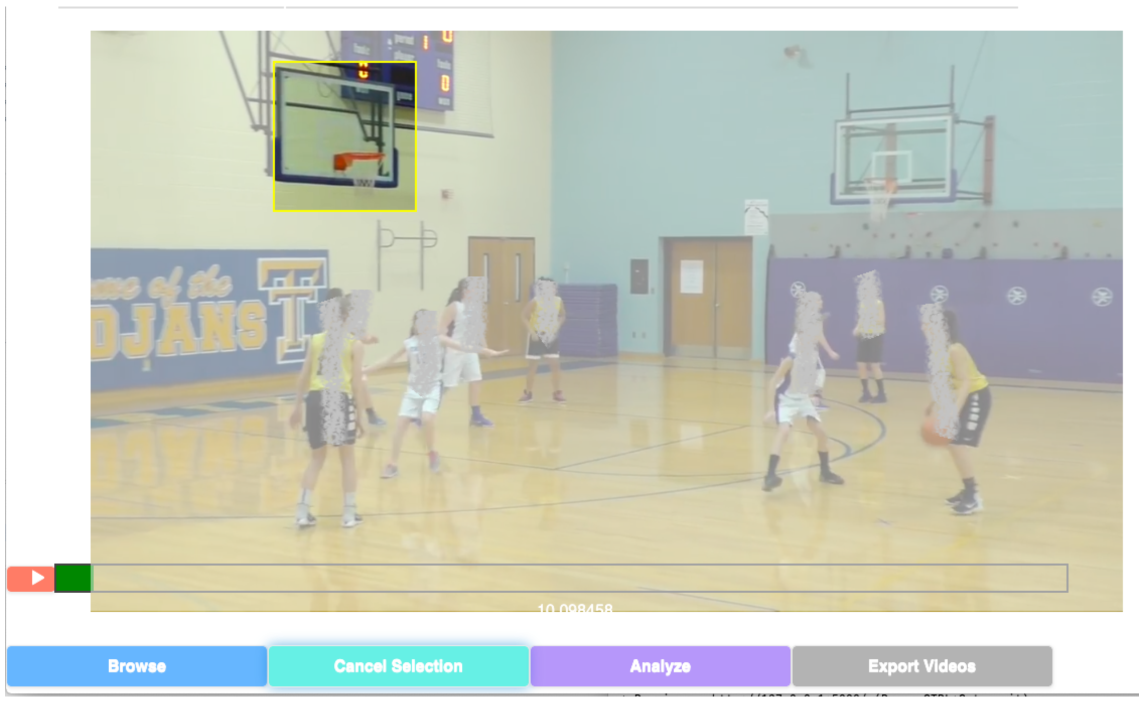


Figure 4.2: Selecting a template after the video is uploaded in the existing system based on template-based template matching.

3. The system calculates the value for each target frame in the original video with the template resulting into a likelihood matching value for each target frame. In this step,

the system currently uses a template matching algorithm with the cross-correlation function. The cross-correlation function multiplies the template and the target image coordinate's pixel colour values together and assigns the likelihood value for the target coordinate (i.e., the likelihood of the target image matching the template at the coordinate). The target frame now slides right by one pixel (and down by one and move back to the left-most pixel when it reaches the right-end) in the target frame, and the likelihood value for that pixel is calculated. The process continues until all the pixel values in that frame are calculated.

We can imagine that if the two signals line up exactly, multiplying them together will square the template. If they are not lined up, then the product will be smaller. So, the value where the maximum is located is where the template and target possibly overlap. Hence, the maximum value among obtained values is assigned as the frames' likelihood value. This process continues for all the frames until each frame holds a likelihood value. We then use a normalization function to normalize the obtained frame value for each frame between 0 and 1, 1 being the highest likelihood of the frame containing the template. The system calculates the normalized value for all the frames. The normalized value for each frame is displayed as a scatter plot graph as shown in Figure 4.3. This graph allows the users to identify the frames of their interest and distinguish them from the other frames. The users are now able to use this graph (As shown in Figure 4.3), to visually investigate and determine the potential threshold, that segregates the likely and unlikely frames to contain a template.

4. From the given graph, we aim to allow the users to observe the distinctiveness between

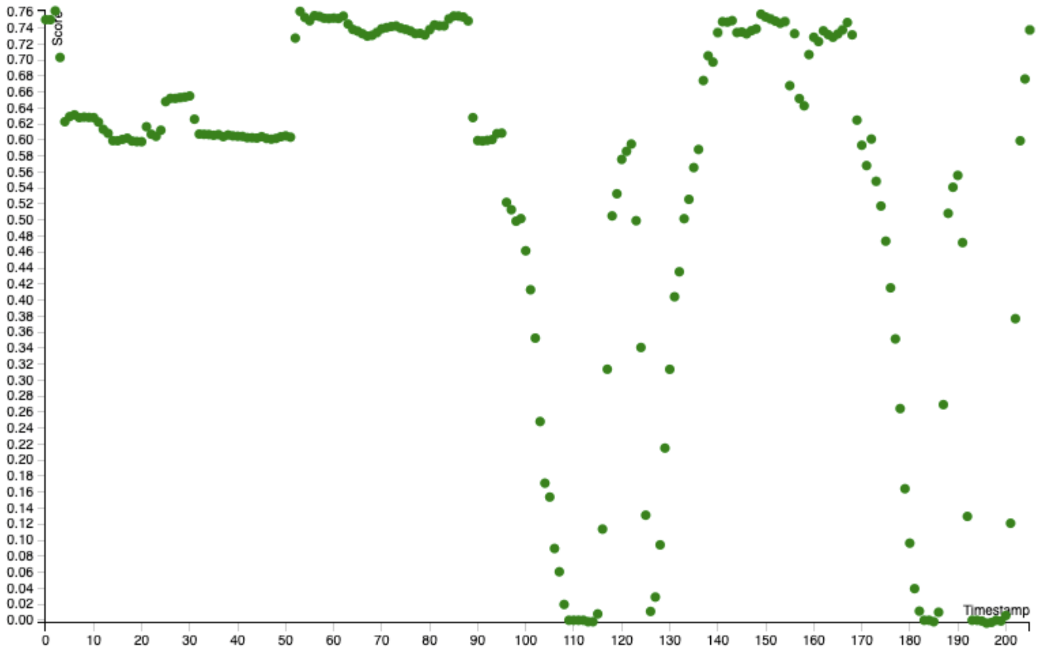


Figure 4.3: The obtained likelihood values using normalized cross-correlation are plotted in the graph.

the likely and unlikely frames from the scattered likelihood value and to aid them to select an appropriate threshold, without actually inspecting the actual video frames. In the Figure 4.4, the resulting likelihood values is plotted in the graph with green points, where the select threshold of 0.52 is used to distinguish the likely and unlikely frames. The frames holding values equal or greater than 0.52 (i.e., represented by the green data points in Figure 4.4) are the ones that are selected to contain the template, whereas the frames represented by the red data points are unlikely to contain the template and will thus be discarded from the final video output.

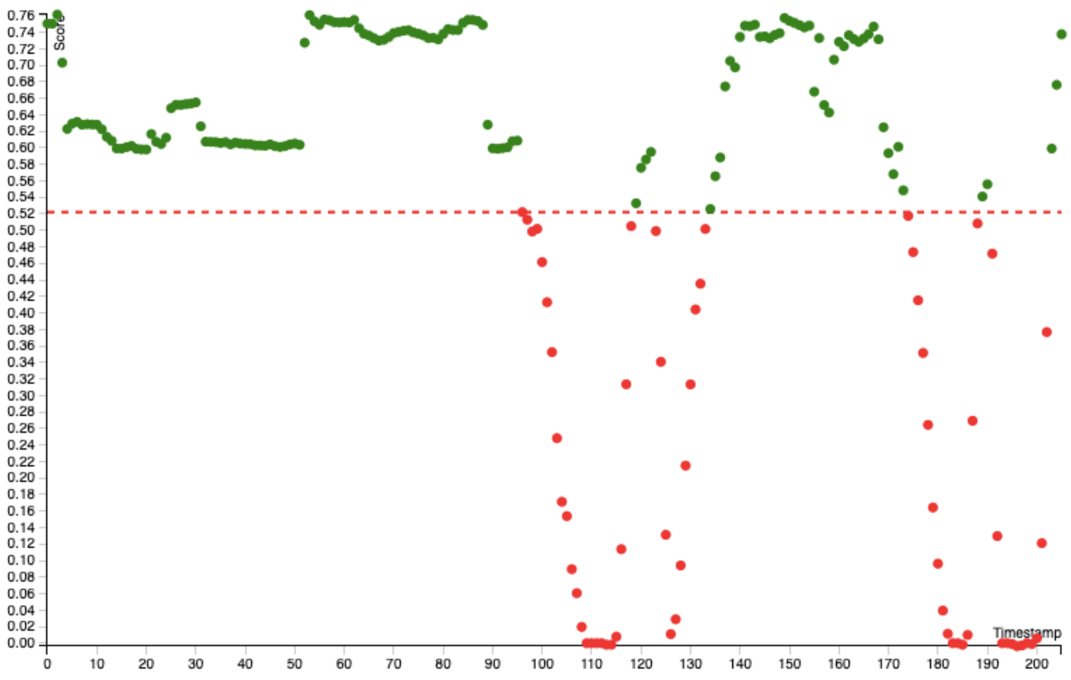


Figure 4.4: The user-selected threshold that separates the likely and unlikely frames to contain the template.

5. This final step, the system now extracts video segments with the frames with likelihood value equal to or higher than a threshold value i.e., values higher than 0.52 as in the

case shown in Figure 4.4. Once the user selects the threshold, the resultant frames are exported as a video with all the frames the system assumed to contain the template.

In the existing template-based template matching research studied in this thesis, our focus was to investigate alternate approaches for Step 3 by exploring a set of algorithms to address observed limitations (like rotated, scaled frames with uneven illuminance and so on). The existing template-based template matching has limitations on the rotated and scaled videos because the approach itself is not transformation (scaling or rotation) invariant. To overcome these issues, we studied and experimented on the possible alternative approaches to overcome these limitations later in Chapter 5.

4.1 Research Questions

Limitations of template-based template matching like rotation or scaling variance can result into the loss of a potential output frame that affects the systems optimization. We have to focus on minimizing data loss by overcoming the limitations for single-camera videos. A thorough background study of the working mechanism and outcomes of the template-based template matching approach and the existing system raises many questions that require feasible solutions. These solutions can help us fix the problem domain of the existing issues to simplify further study. We define these problem domian related questions as the research questions, and their motive is to help us brainstorm to come up with a possible solution to these questions. The research questions that can highlight the importance of this study are discussed below.

- RQ1: How can we improve efficiency and accuracy for analyzing multimedia data while allowing the users to handle the interactive interface and control the results?

- RQ2: Are there enough datasets to train a predictive model for the single-camera video extraction? If not, is it possible to use supervised machine learning algorithms without the presence of an optimal dataset for single-camera videos?

- RQ3: What algorithms work well with the video analysis for videos with different conditions, like a rotated or zoomed frame or illuminance, etc., and provide optimal results? Is there an algorithm for template matching that helps achieve transformation (i.e., zooming and scaling) invariance and provides optimal results if we use an inconsistent varying quality single-camera video?

All these questions are the backbone of our research and the answers to these questions can help us to justify the reasons for conducting this study. For this, we start a preliminary study in Chapter 5, optimize our results in Chapter 6 and disucss the observed answers to these research questions in Chapter 7.

Chapter 5

Data Preparation and Preliminary Analyses

Both supervised and unsupervised algorithms can be used for video extraction depending upon the availability of raw data. In cases using multimedia data, when there is a lack of training data, analysis using supervised learning algorithms may become difficult. In this chapter, we first experiment with the possibility of using supervised machine learning algorithms (as explained in detail in Section 5.3.1) without a proper training dataset to determine the validity of our previous statement. After validating our base argument, we try to find a feasible algorithm that helps answer our research questions (as discussed in Section 1.2 and 4.1).

The initial research questions are derived from the limitations of the template-based template matching algorithm for object detection. The existing tool that uses the template-based template matching algorithm is also used as a reference to design a prototype for this research. The template-based template matching approach does not detect the zoomed and scaled video frames. In this chapter, we will first experiment to verify if the existing template-based template matching approach works poorly with videos under certain photographic conditions such as illuminance, environmental obstructions like rain, and poor visibility conditions. All these are the limitations of the template-based template matching approach while extracting the single-camera videos. We will then experiment with different approaches to overcome the limitations of the existing tool.

We started the preliminary investigations with a survey targeting a potential user group to understand the problem domain of the research (discussed in detail in Section 5.1). We collected and analyzed their responses and then discussed whether the results from the survey supports the research idea. We then investigated the current approaches to identify potential bottlenecks that may be hindering better performance results in Section 5.2. Next, we experimented with the single camera-video extraction using different algorithms that support the research domain. We started by studying the template-based template matching functions for a manually rotated and scaled single-camera video. We noted the results and analyzed their observed practical limitations in Section 5.2.4. We then explored a set of potential alternative approaches that may be used to address all or some of those issues identified (discussed in detail in Section 5.3). We have also experimented with different supervised machine learning algorithms (in Section 5.3.1). After a thorough investigation, we dropped the idea of using supervised learning algorithms and studied feature-based template matching instead. We used feature-based template matching because it is transformation invariant (as explained in Section 2.2.2). We finally compared the preliminary results from all the experiments to solidify an algorithm that is most effective to achieve the transformation invariance. We will show how these new algorithms could improve the existing approach by addressing some of its limitations. We finally compare and study the proposed algorithm and its optimization in Chapter 6.

5.1 Preliminary User Studies

We conducted an online survey to better understand the user perspectives on the research idea. While we investigated and collected certain statistical data, they were used as descriptive data to understand the problems and propose solutions as the new approaches.

Initially, we created a questionnaire with a group of questions focused on the usability and the suitable application of the proposed research. The online questionnaire began with asking for the consent of the users. If they were willing to provide their consent, they could go ahead and respond to the questionnaire, which had a few open-ended and close-ended questions. After completion of the form but before submitting their responses, if they still wanted to withdraw their participation, they had an option in the end, which enabled them to cancel their participation any time and erase all their data.

There were 12 participants for the preliminary online survey, among which 6 (50%) participated in the follow-up survey. The follow-up survey is a survey for those who are currently participating or have participated in organized sports in the past. The results obtained from the participants participating in the initial study as well as the questionnaire's answers are discussed in this section.

5.1.1 General Survey Analysis

The general survey contained questions related to video extraction from the users' perspectives. The users are given a scenario on which they miss a sport game/match and are given different options to watch the same game again, e.g., watching only highlights, watching the whole game's re-play, etc. The user's feedback is then recorded, analyzed, and discussed in detail here to understand the relevance of this research domain. The questions we asked are displayed on top of the graph, followed by the analysis of the responses for each question.

QUESTION 1: If you miss a sports match, do you want to watch the replay in order to get the updates?

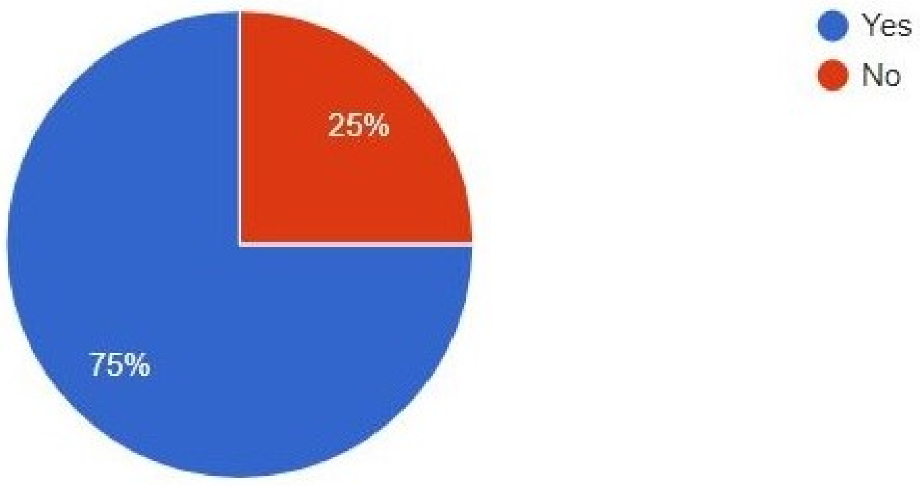


Figure 5.1: participant's responses when asked if they want to re-watch the match in case they missed it.

Response:

As shown in the Figure 5.1, out of the 12 participants who participated in the general survey, 9 participants (75%) would like to re-watch the match even though they miss it. i.e., a short highlight or a full-recorded version of the game. In contrast, 3 participants (25%) don't want to re-watch the game.

Analysis:

The answers show the interest of most of the participants and their eagerness to watch the game despite missing it. The participants are willing to dedicate the time and effort to know what happened during the match.

QUESTION 2: If you could not manage time to watch a live match or its replay, how do you get the updates?

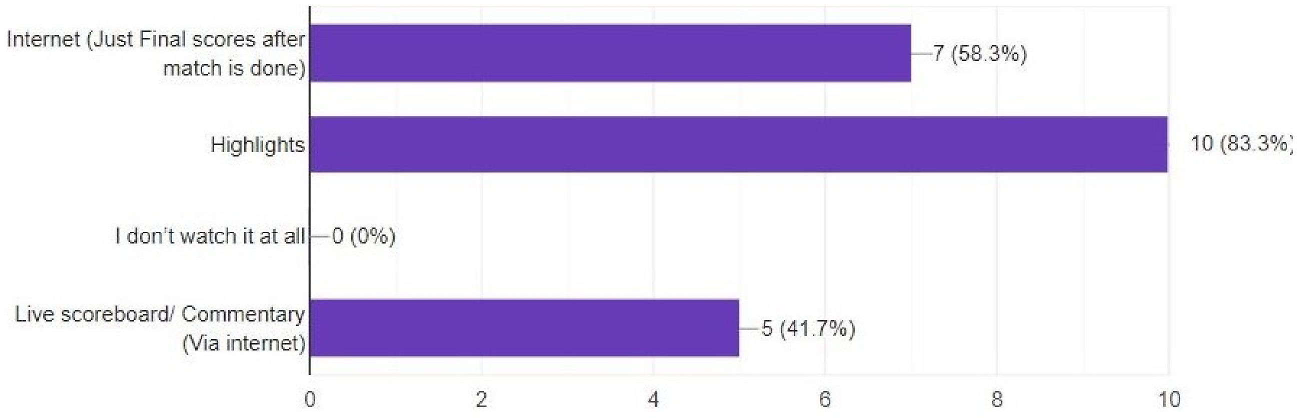


Figure 5.2: Participants responses when asked how do they get the updates for the match they missed.

Response:

We asked the participants how they received their updates for the match they missed and gave them multiple options to choose from as shown in Figure 5.2. We observed that 10 participants (83.3%) watch the highlights whereas 7 participants (58.3%) go through the internet for the final scores.

Analysis:

We observed that most participants rely on the highlights of the game. This may be because highlights cover the important happenings in any sport game. It also looks like when the participants wants to find the score live, they rely on the live score board either using internet or presumably radio or a podcast.

QUESTION 3: Are you satisfied by watching just the highlights of the match?

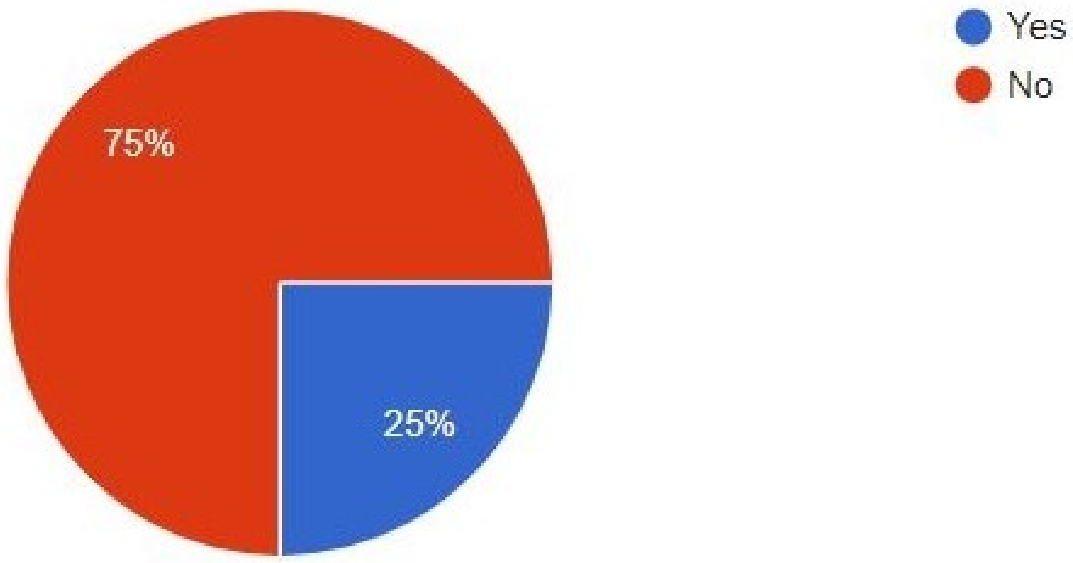


Figure 5.3: participant's responses when asked if they are satisfied just watching the highlights.

Response:

As seen in the Figure 5.3, 9 participants (75%) are not satisfied watching just the highlights of the game they missed. 3 participants (25%) are satisfied just watching the highlights.

Analysis:

This may be because the participants are willing to watch their favorite player and other moments normally not shown in the highlights. We followed up with the next question to observe their satisfaction with our research idea.

QUESTION 4: Do the highlights provided by the Sports Channel (or similar media) provide enough insights into the match?

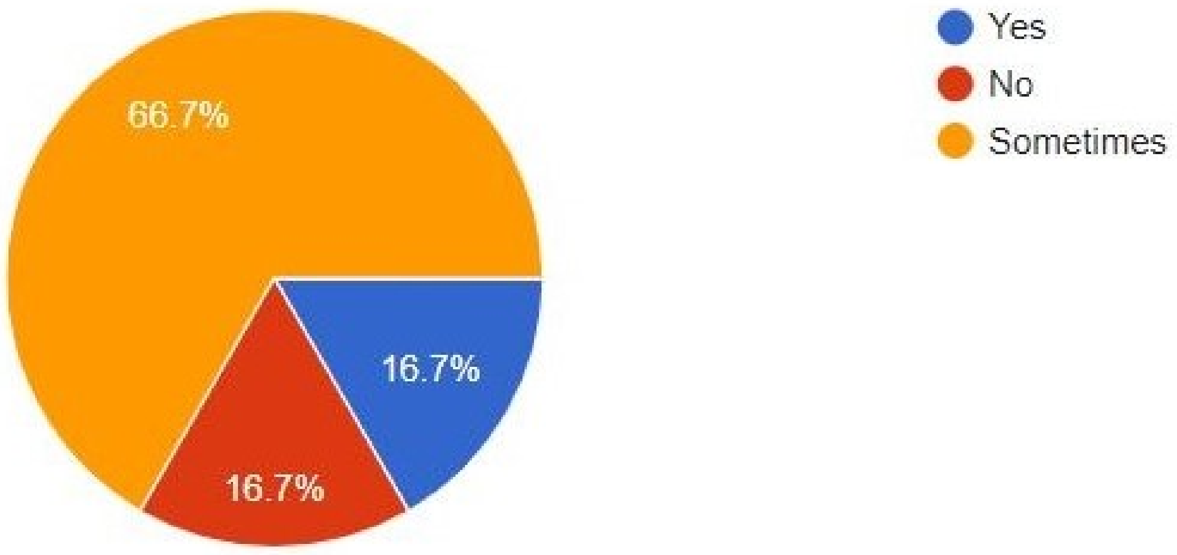


Figure 5.4: participant's responses when asked if the sports channel provide enough insights on the whole match.

Response:

As shown in Figure 5.4, 2 participants (16.7%) believed that the channel does provide enough insights, and 2 participants (16.7%) believed that the channel does not provide enough insights on the whole match in their highlights. The remaining 8 participants (66.7%) mentioned that the sports can occasionally give enough insights into the match.

Analysis:

Most users find that the highlights can only provide enough details sometimes because the participants are presumably looking for their preferred key moments in the game that the highlights may sometimes skips.

QUESTION 5: In addition to what's provided, in your opinion what do you think can make the highlights approachable and interesting?

Response:

This was a subjective question. We asked the participants their views on what can make the highlights more interesting. Only 7 participants chose to answer the open-ended questions, and their responses are noted as follows.

- 1 participant stressed that multiple focus points on different individuals at times could improve the highlights.
- 1 participant mentioned that the highlights should not miss any of the significant moments of the match.
- 1 participants also noted that the camera should always be focused on the ball.

- 1 participant mentioned that the short clips of important moments of the game and keeping it within a time frame of 10min could also make the highlights interesting.
- 1 participant said High Definition (HD) video quality, English language, and being uploaded right after the match also could help improve the highlights.
- For 1 participant, except for the baskets/goals/catches, wickets, sixes, fours, the funny moments between the match would make the highlight more interesting ("Like when Kobe and Michael were playing against each other, Michael didn't let him take the ball").

Analysis:

All the user response points towards the elements that can ensure quality video extraction for sports videos. According to the participants, the extracted highlights video should have almost all key moments for any game. The important moments may include players interactions, video clarity, focus on funny moments etc.

QUESTION 6: Would you like it if you could control what portions of the match you want to watch as highlights?

Response:

As shown in Figure 5.5, 8 participants (66.7%) agreed that they wanted to interact with the system to control what they can see for the highlights. On the other hand, 4 participants (33.3%) are partially interested, and there are no participants that would not prefer the interactive interface.

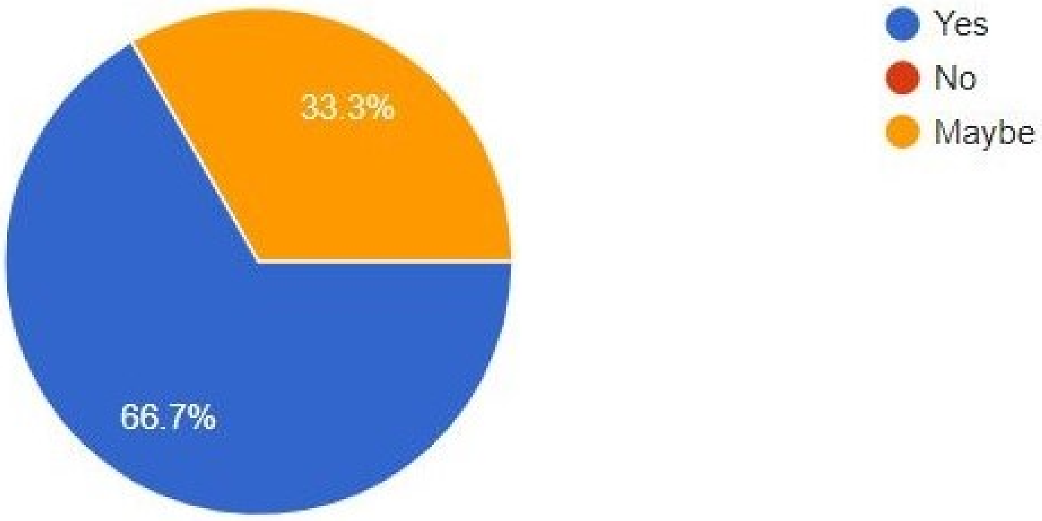


Figure 5.5: participant's responses when asked if they want to be able to control what goes in the highlights of a match.

Analysis:

The responses shows the willingness of the users to control the extracted highlights. It would be interesting to see if providing a tool like the one we developed in this research may help those who are unsure to see how it would be like to control the video highlights interactively.

QUESTION 7: Is there a situation where you watch a whole match just for few reasons i.e. watching a single-player play?

Response:

As seen in Figure 5.6, 6 participants (50%) watch the match to observe a single player. 5 participants (41.7%) were interested in watching the whole match instead of focusing on a

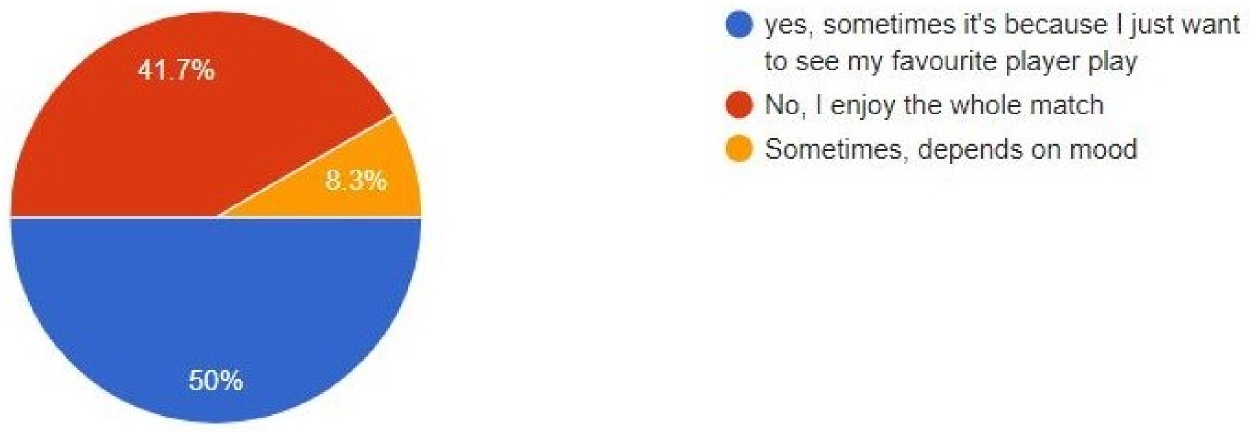


Figure 5.6: participant's responses when asked if they watch the whole match for any specific reasons.

single player. The remaining 1 participant (8.3%)answered it depends on their mood.

Analysis: We observe that half of the participants seem okay with only watching some video fragments based on their player choice. It shows the user's desire to control what and how much they want to watch.

QUESTION 8: What are other reasons that make you want to watch a match even though you are not interested in the whole game? (Hint- answers can be.. m favourite player, team, the opening act, etc.)

Response:

We asked the users what they thought were some interesting moments in a sports game they do not want to miss. It was an open-ended question and we received 12 responses noted below.

- 8 participants (66.7%) did not want to miss their favorite player or team, or they wanted to watch the interesting interactions between players, among which one mentioned they also did not want to miss the opening ceremony.
- 2 participants (16.67%) wanted to be up to date with the league standing, i.e., to mention which rank the team holds in the league.
- 1 participant (8.3%) liked to keep track of how the formation keeps changing during the gameplay. The participant mentioned, "it was watching your favorite player struggle and get out of the big defensive moves. Sometimes silly moments like watching Neymar cry at small things" as an interesting moment.
- 1 participant (8.3%) replied there was none.

Analysis:

We observed that there are a lot parameters in the game/play that makes user want to watch the whole match despite not being interested in a particular game. The responses give us an insight on the user's desired key moments in a sport game/play.

QUESTION 9: Do you have any idea about the available software that are used in sports video editing/extraction (shortening the video; extracting highlights of the match)?

Response: As shown in the Figure 5.7, 7 participants (58.3%) did not know about the available software for video editing, whereas 5 participants (41.7%) do.

Analysis:

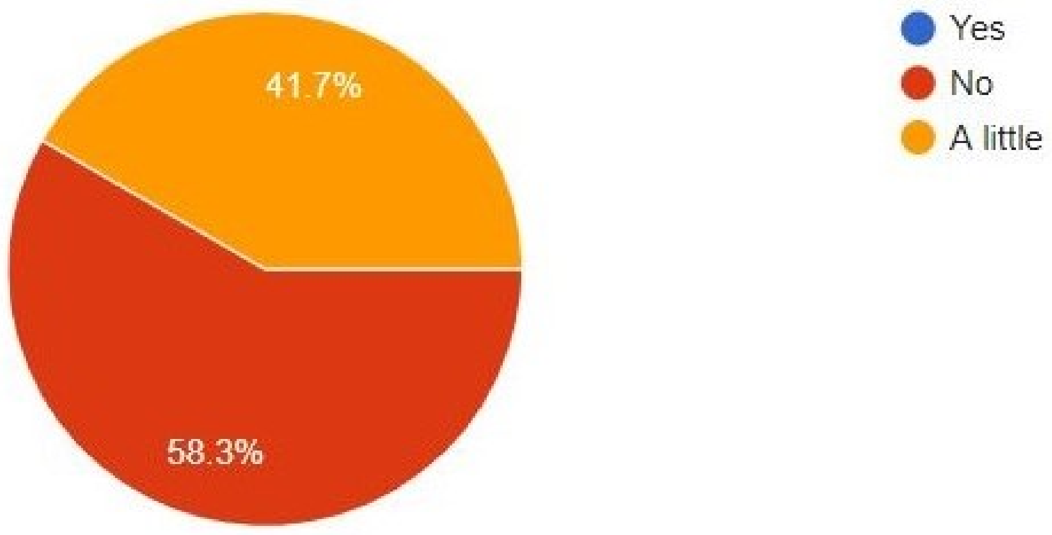


Figure 5.7: participant's responses when asked if they knew any video editing software.

Since video editing software is usually used professionally and often comes with a cost (like Adobe creative cloud), the participants were less likely to use it.

QUESTION 10: Have you ever been in a situation where you were supposed to watch a whole long video (maybe a show, a party, or a wedding or any sports event)?

Response:

It was an open-ended question but most participants gave a yes/no response. As shown in Figure 5.8, 8 participants (66.7%) have been in a place where they had to watch a long video, whereas 1 participant (8.3%) have not faced that situation.

Analysis:

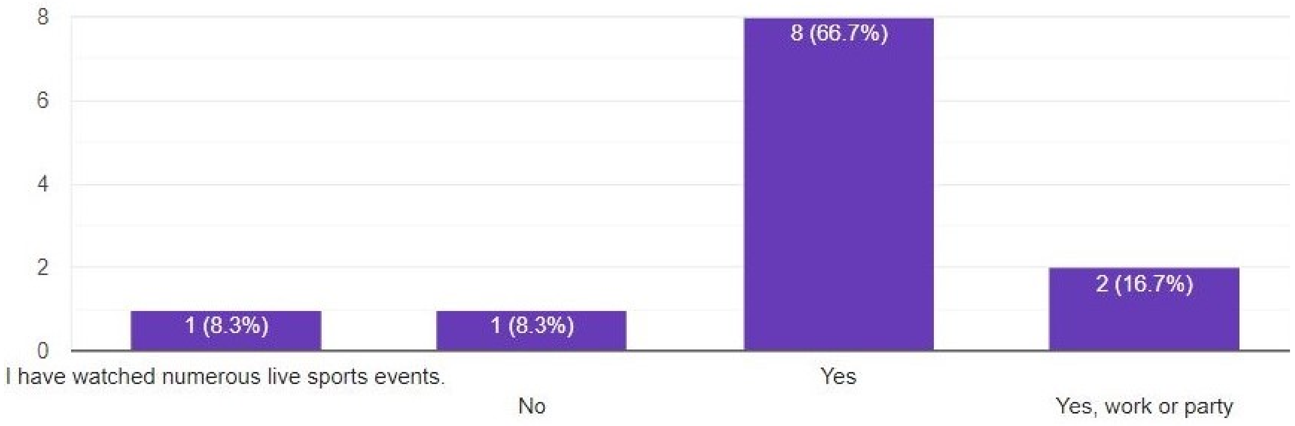


Figure 5.8: participant's responses when asked if they had been in a situation where they had to watch a long video.

We observe that most users are likely to face situation where they will have to watch a long video without having an alternative to watch the key moments. This observation supports the motivation of this research idea.

QUESTION 11: Can you explain a situation where you felt like you were wasting your time and want a shortened version of this video you are watching?

Response:

We asked the participants their views on watching the long videos and got many answers as it was an open-ended question. We recorded 9 responses and they are noted as follows:

- 1 participant (11.11%) was tired and could not focus on the match at all.
- 1 participant (11.11%) mentioned that they find many unnecessary segments they would like to skip, especially while watching Youtube videos.
- The pre and post-match analyses are the clips that 1 participant (11.11%) would like

to skip while re-watching a video.

- 2 participants (22.22%) feel the need to skip some segments while watching a long birthday or a wedding video.
- 1 participant (11.11%) mentions there was no such moment.

Analysis: We observed that the users, in one way or another, would want to be able to control what they watch. From the participants responses, we observed that while long videos, there are so many instances when the users would like to skip many sections to watch the main content. It follows that the user's desire to segment and shorten a video to get only the content of their interest.

After the general survey ended, 6 participants decided to continue the next survey focused on the sportsperson. We conducted this second survey to understand the usability and application of the end goal for small-scale video extraction, like individual player performance analysis for scrimmages or practice sessions. We noted the responses and analyzed the user feedbacks below.

Follow-up Survey: responses from the Sportsperson

QUESTION 1: What sport do you play?

Response:

Among the participants, only 1 participant (16.7%) played soccer, and the rest 5 participants (83.3%) played or taught other games as observed from Figure 5.9.

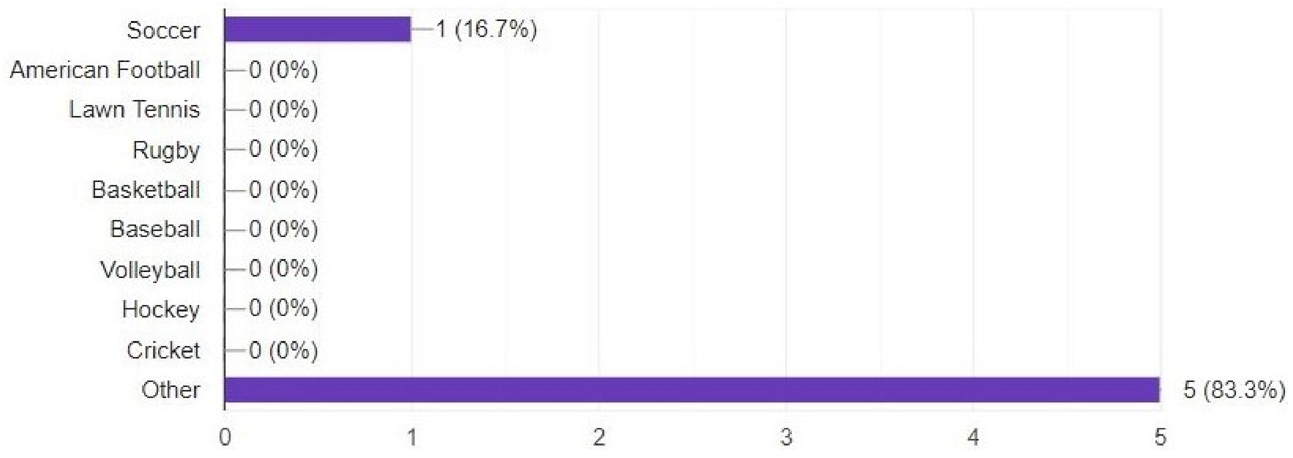


Figure 5.9: We asked the participants what did they play or teach.

Analysis:

We observe that participants play soccer and other games presumably badminton, table tennis, etc.

QUESTION 2: While you have practice matches, how do you evaluate your own or team's performance (Student's performance in case of a coach)?

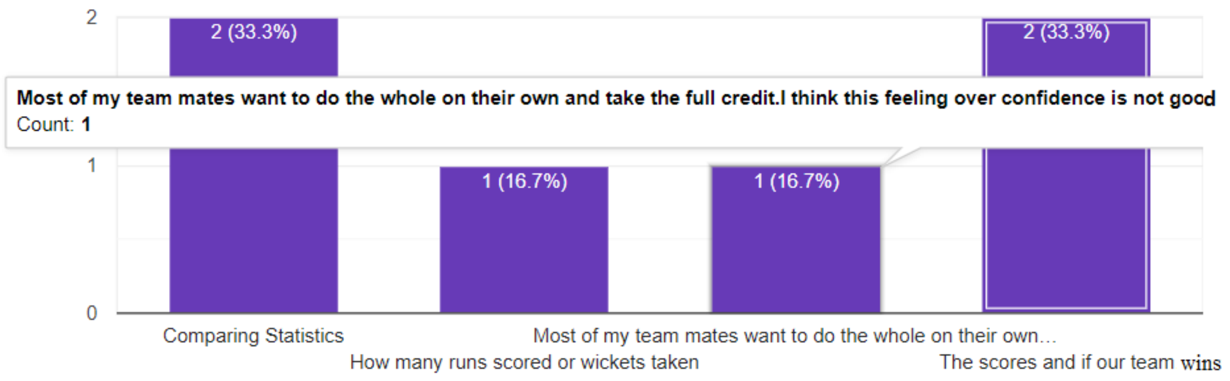


Figure 5.10: We asked the participants how they evaluate their own or their team's performance.

Response:

We can observe from the Figure 5.10, the participants mentioned they did different things to evaluate their and their teammate's performance. The players compared statistics of the practice match based on individual performance and goals or scores attained. Different performance metrics like wickets taken, fouls committed or total time on the field, etc., help keep track of the player's performance.

Analysis:

Based on the responses, we observed that different performance metrics can be used as input to extract relevant parts of a video. We discuss more on this matter in upcoming sections.

QUESTION 3: Do you record any videos in order to analyze your (or your player's) performance?

Response:

From Figure 5.11, we can observe that 4 participants (66.67%) use camera sometimes to record their performance. Remaining 2 participants (33.33%) do not use cameras. To the 4 participants who do use cameras, we asked a follow up question.

Analysis:

Based on the responses, we observed that most of the users use a single camera video like mobile phone. This may indicate that the use of single camera video to capture games in a small scale sport game is very common.

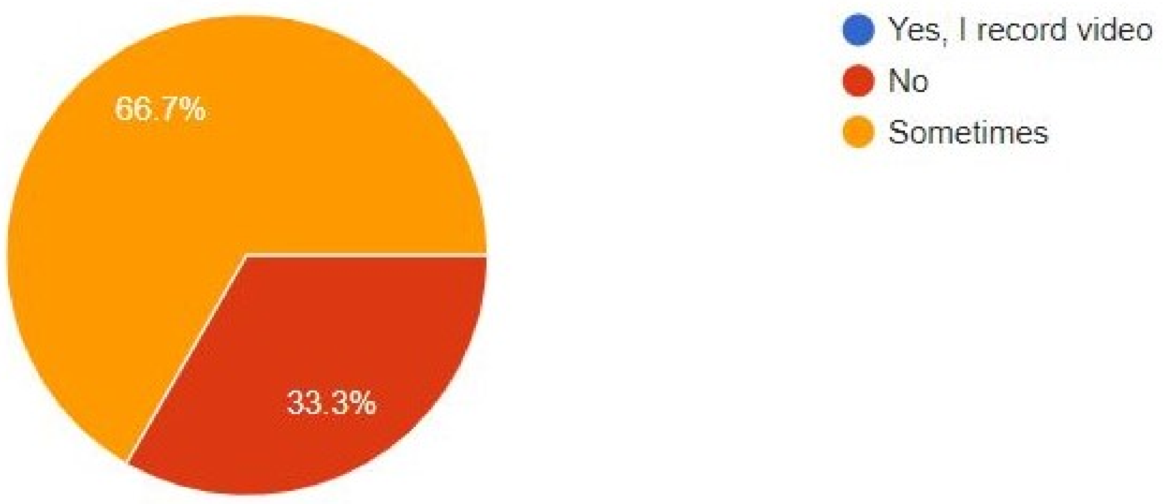


Figure 5.11: Participant's responses when asked if they used any cameras to record their performance.

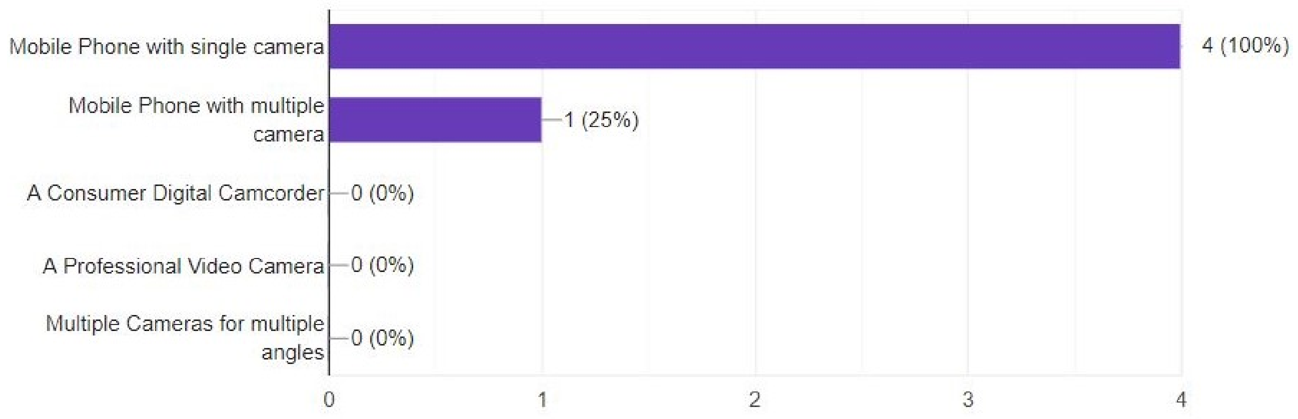


Figure 5.12: Participant's response when asked what type of camera do they use for recording videos.

QUESTION 4: What type of camera do you use during recording?

Response:

As shown in Figure 5.12, we asked a multiple choice question where users could choose more than one response. We can observe that all the 4 participants (100%) used a mobile phone camera for recording the videos. 1 participant also uses phone with a multiple camera to record the videos.

Analysis:

We observed that the use of single-camera video seems common in sports video analysis, this response can be taken as a motivation for this research.

QUESTION 5: What do you generally analyze from such videos (i.e. what specifically do you look for when you analyze such videos)?

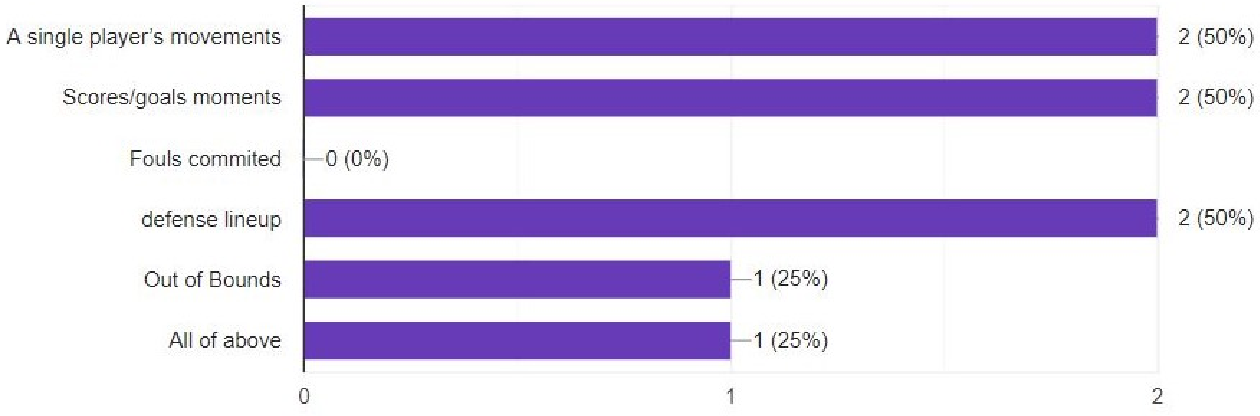


Figure 5.13: Participant's response when asked what do they look for in the recorded video to analyze the performance of the player or the team.

Response:

We asked a question where the participants could select multiple answers as shown in Figure 5.13. We observe that 2 participants tend to analyze a single player's movements and goal moments to analyze the performance. 2 participants are also interested in the defense line-up and out of bounds and other key moments to analyze the performance.

Analysis:

This indicates the possible parameters the participants are willing to use a key to watch and analyze the video they have recorded.

QUESTION 6: If you watch a video recorded by a single camera/mobile phone, what issues do you face while analyzing the performance of all the players?

Response: This was a subjective question. We got 4 responses where 2 participants mentioned that the video is too long, and sometimes even when they can figure out the problem, they can't improve it. 1 participant mentioned that they get "unclear images if they zoom and the video capturing a camera range is not wide enough."

Analysis:

The participants faced some difficulties while analyzing their recorded videos. These difficulties give an idea of the problem domain and the possible ways of improving user participation for multimedia data analysis.

QUESTION 7: Do you use any kind of software for video extraction or video editing (like extracting video highlights and video shortening)?

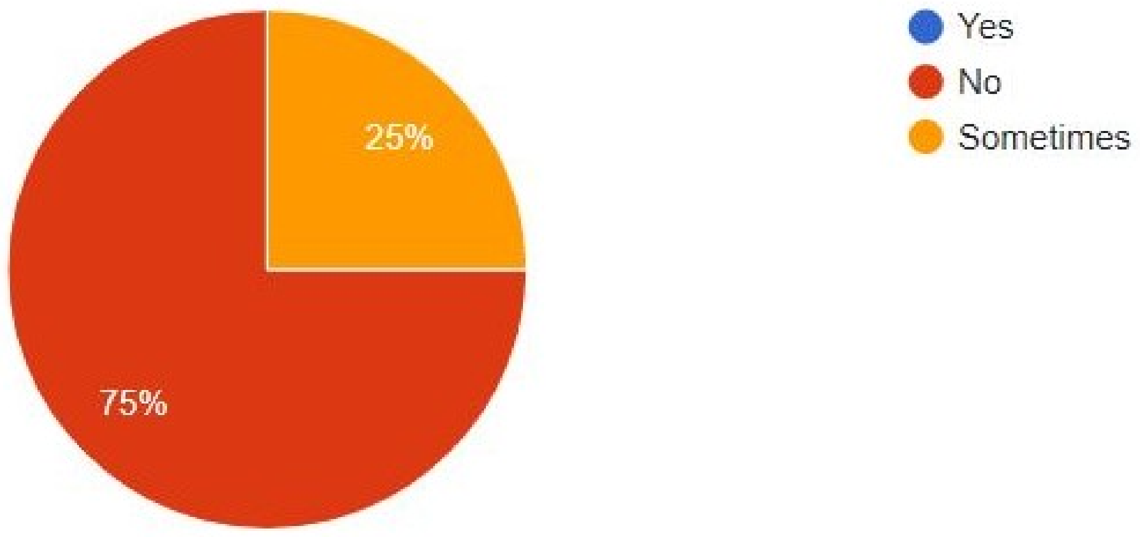


Figure 5.14: We asked the participants if they used any video editing software, and we recorded their responses.

Response:

As shown in Figure 5.14, we had 4 responses, among which 3 participants (75%) of the participants do not use any video editing software, whereas 1 participant (25%) answered sometimes.

Analysis:

The participants who take videos to analyze the player's performance try to track the single player's movement and use parameters such as score moments, defense line-ups, and out-of-bounds to analyze the overall performance.

QUESTION 8: Video extraction tool allow you to select a specific portion of a lengthy video so that you do not have to go through the whole video just to find the certain parts.

Hoe exactly would you use such a video extraction tool to analyze a practice match?

Response:

When asked how the participants would like to use the tool, 1 participant mentioned that they would like to focus on their interest, such as a player or certain plays, and extract only the relevant video segments. 1 participant mentioned that they want to concentrate on the parts where the team lacks strategy, thus, it helps the team and players to avoid making redundant mistakes and overcome that challenge.

Analysis:

The user responses shows the ways using which the users can benefit from the proposed research idea.

The survey helped us better understand the background of the problem domain of this research. We also analyzed the background for the real-world application of our research idea. The users willing to use video editing software for video extraction will likely benefit from this research. The sportsperson who wants to extract the video segments based on their interest seems interested in this research idea. This research will also likely open numerous possibilities for research on single-camera videos by allowing users to control what they want to extract.

We analyzed the preliminary survey and set the user responses as achievable milestones for the research. The user requirements and findings, as observed from the survey, are:

1. People would be interested in controlling which part of the video they want to watch as a highlight. They seem interested in the idea of interactive interfacing.

2. People mentioned situations where they are compelled to watch long videos, and they would want to avoid such situations and watch only the video segments they are most interested in. Understanding the possible key parameters for the extraction can be a motivation for video segment extraction.

3. Sportspersons mostly use their mobile phones to capture games and analyze player performance. They use different performance metrics like the player's movement or key moments like goals or fouls to analyze their overall performance. Allowing users to choose these parameters should be kept in mind while creating an interactive user interface.

After setting the milestones, we created a detailed study flow for this research and drafted a set of experiments with different algorithms. Once we propose a prototype and identify a suitable algorithm for this research, we again conduct a user-evaluation study (in Section 6.4) to compare our findings with user feedback. In the next section, we start our experimentation process by discussing the potential issues and limitations of the current template-based template matching functions with several single-camera soccer video.

5.2 Identifying Issues of Current Approaches

The template-based template matching algorithm motivated us to conduct this research. A template is a rectangular reference area from a video frame selected by a user, and the algorithm examines its presence in each video frame based on distance values (as explained in Section 2.2.1). For the template-based template matching experiments, we used a short (three-and-a-half-minute) single-camera soccer video with the template-based template matching approach and its distance measure functions. We manually rotated and/or scaled some video frames of the soccer video (as shown in Figure 5.2) before using it to input our experiment. The user-selected template for this research is the goalpost area, and one example of template selection is shown in Figure 5.16. The algorithm uses the sliding

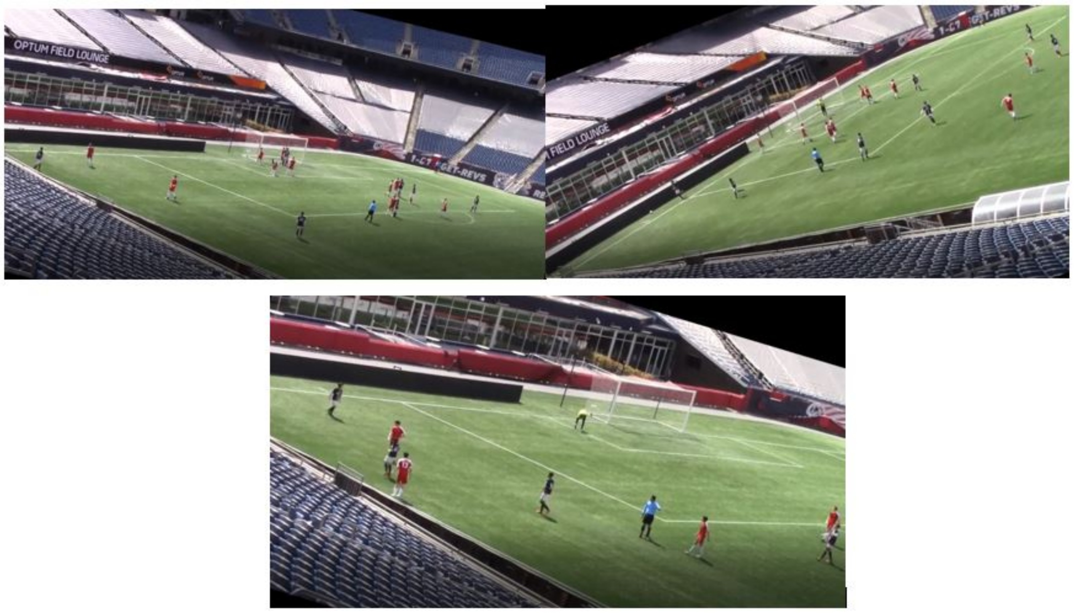


Figure 5.15: Some of the frames in the soccer video are manually scaled and rotated as shown in figure.

window approach [46] (as shown in Figure 5.17) to find the location of the adjacent pixel. An $N \times N$ sliding window is selected based on the template, and then it starts moving horizontally and overlaps a region referencing each pixel in the target video frame. The function slowly moves through the image and compares the template with the overlapped patches of a definite size. For each overlapped area, the tool calculates and saves a correlation (or distance based on the algorithm to be used) value for the pixel. The window continues to move.

After obtaining the values for all the pixels in the frame, the algorithm will help locate the

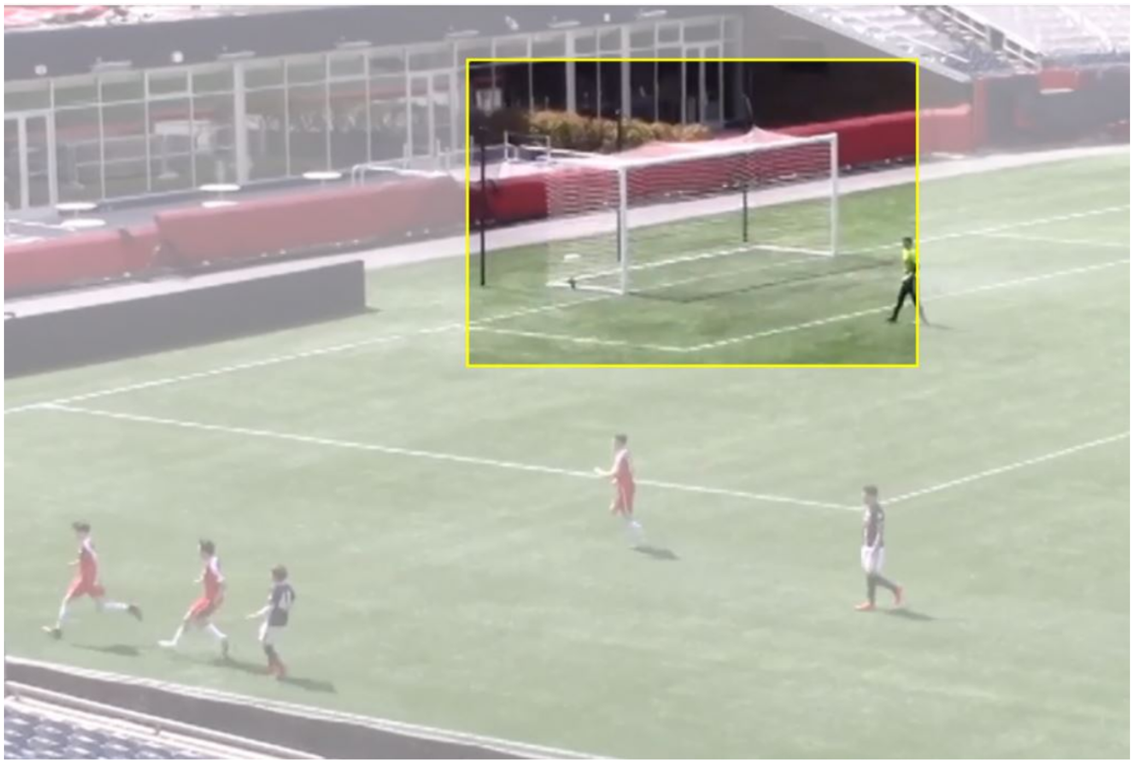


Figure 5.16: The entire area with goalkeeper area selected is the template is shown with a rectangular yellow bounding box.

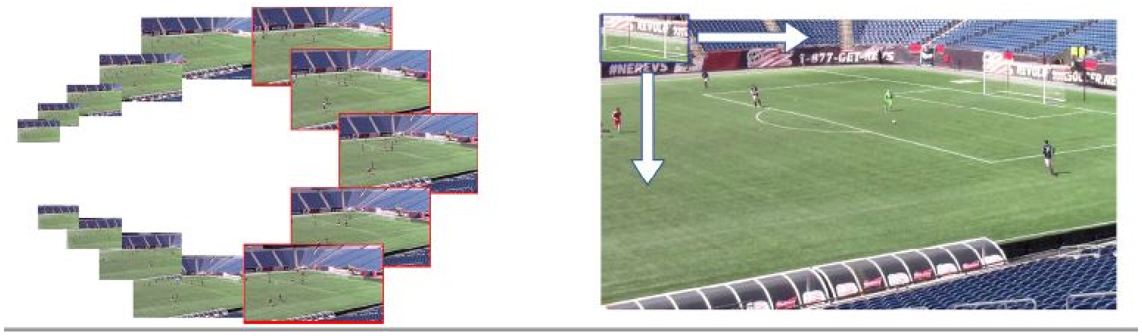


Figure 5.17: The sliding window mechanism matching through all the video frames [12].

matching pixel in the template based on the minimum or maximum value. For instance, the distance measure function extracts the maximum (or minimum) value in the output array and selects it as the pixel location for that frame. The maximum (or minimum) value for the location denotes the pixel location top-left corner of the matched rectangle. The max value (or min value) also denotes the likelihood of that particular frame containing the template. The sliding window then slides and overlaps the second video frame, follows the same mechanism, and extracts its maximum (or minimum) as shown in Figure 5.18. This process continues for all the video frames until the algorithm obtains the correlation value for each frame. The sliding window approach is time-consuming, so the users can choose to skip a few pixels to speed up the process for research purposes. For the initial study in the existing approach, we skipped four pixels in the sliding window to decrease the run-time without affecting the effectiveness of the algorithm. The correlation value calculation for every fourth pixel saves the program run-time without compromising the obtained results. For a video with 900 video frames, for example, if every video frame is a 300×300 image, instead of calculating an array of 90,000-pixel values for that individual frame, we calculate pixel values for every fourth frame, i.e., an array of 22,500-pixel values. The minimum or maximum value in the array based on the function used is extracted as the respective frame's value from this array. Similarly, to further reduce run-time, the algorithm skips frames based on their frame rate (frame per second). The frame rate is 30 fps for the soccer video, so the system skips every 30 frames. It means that instead of calculating 900 frame values, one for each video frame, the algorithm calculates only 30 frame values, one for every 30 frames. The obtained likelihood values are visualized in a scatter plot, the frame number in the x-axis, and the obtained value of the respective frame at that

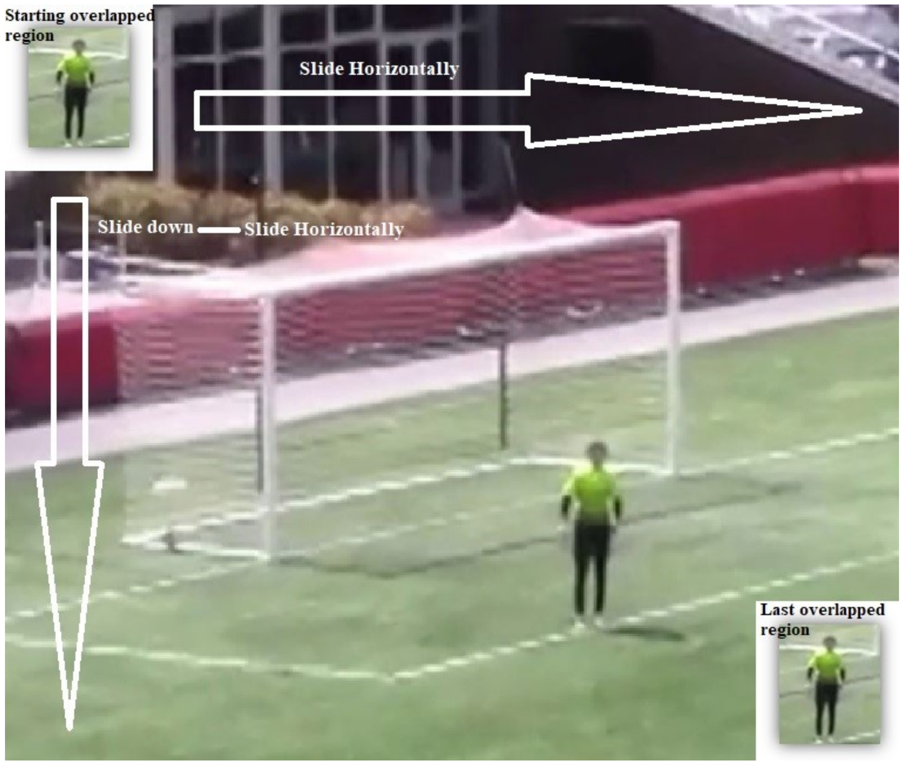


Figure 5.18: The sliding window mechanism when compared to the template [12].

time-stamp in the corresponding y-axis. The scatter plot displays all the calculated values and lets the user determine the threshold. A threshold is a distinct value selected by a user that separates the frame values that are more likely to contain the template from the unlikely frame values. When the user chooses the threshold, all the video frames with values greater than the threshold are extracted as a single output video, assuming the frames contain the template. The green straight line in Figure 5.19 represents a user-selected threshold. The values above the threshold are noted in green data points with a higher likelihood of containing the template. The red data points are values below the threshold and are less likely to contain the template. As shown in the sample of the frames in Figure 5.19, all the green data points represent a frame that will be further extracted as an output video containing a template. In this section, we discuss three main functions

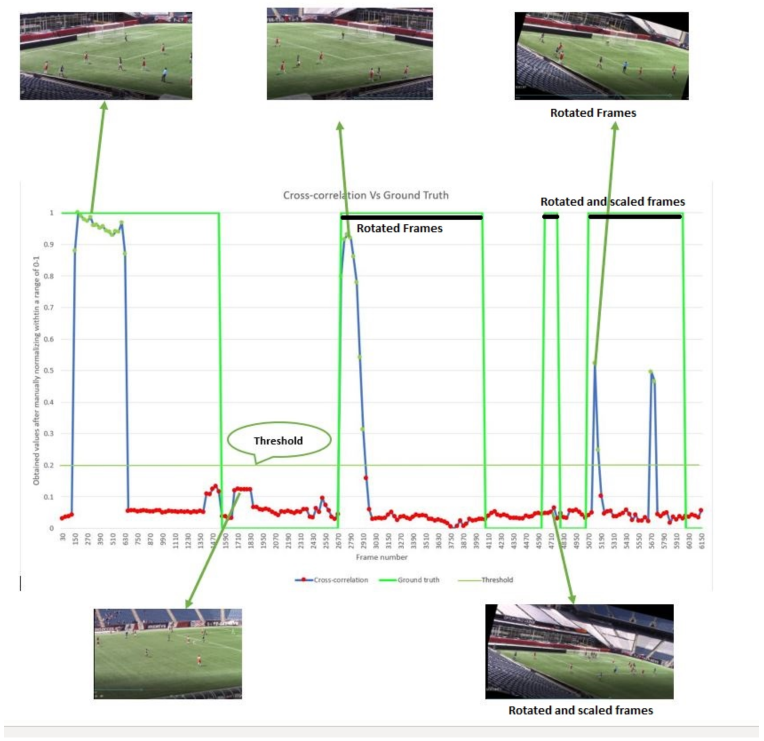


Figure 5.19: The obtained likelohood results with the relevant frames (using threshold value 0.2).

used by the OpenCV for the template-based template matching algorithm. They are:

- Cross-correlation function
- Coefficient of the cross-correlation function (Mean-shifted cross-correlation function)
- Sum of the squared difference function

All template matching functions follow the same mechanism to calculate the distance values. This investigation examined all three different likelihood measures for comparing the template and the video frames. In this section, we will discuss the details of the investigation of these likelihood measures, run on a manually transformed single-camera videos

5.2.1 Template Matching Using Cross Correlation Function

We began the investigation with the cross-correlation function for the likelihood measure (discussed in detail in Section 2.2.1 for the detailed explanation of this function).

After obtaining the likelihood for all the video frames, we display the results as shown in Figure 5.20. The distinct distribution of values on the graph enables the user to select a threshold like shown in Figure 5.20. Figure 5.20 shows that the values obtained by using

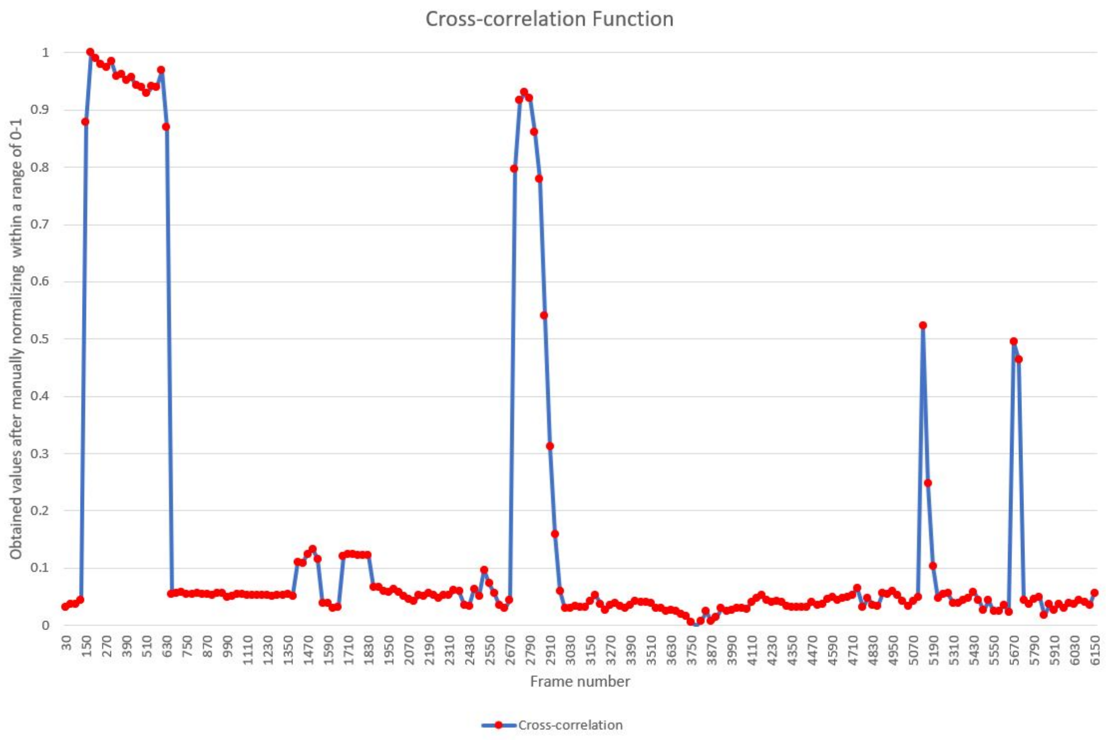


Figure 5.20: The obtained likelihood values for an input video are presented in the graph using the cross-correlation function in template matching.

cross-correlation can be distinguished into likely and unlikely frames to contain the template. Thus, it is easy for the users to select a reasonable threshold value by visually investigating the normalized graph. The visible segregation of values and the proper selection of threshold between the likely and unlikely frames matching the template minimizes the selection of false positives and prevents data loss. To calculate the accuracy of the obtained values, we compared the obtained values of each frame with their respective ground truth. As the values lie between 0 and 1, 0 being unlikely and 1 being the highest likelihood of the frame containing the template, we manually encoded the ground truth for each frame. We observed the ground-truth value on the input video, and if the frame included the template, we encoded the frame's value as 1 or else 0. Finally, we compare the values and discuss the accuracy of the cross-correlation distance function. As observed in

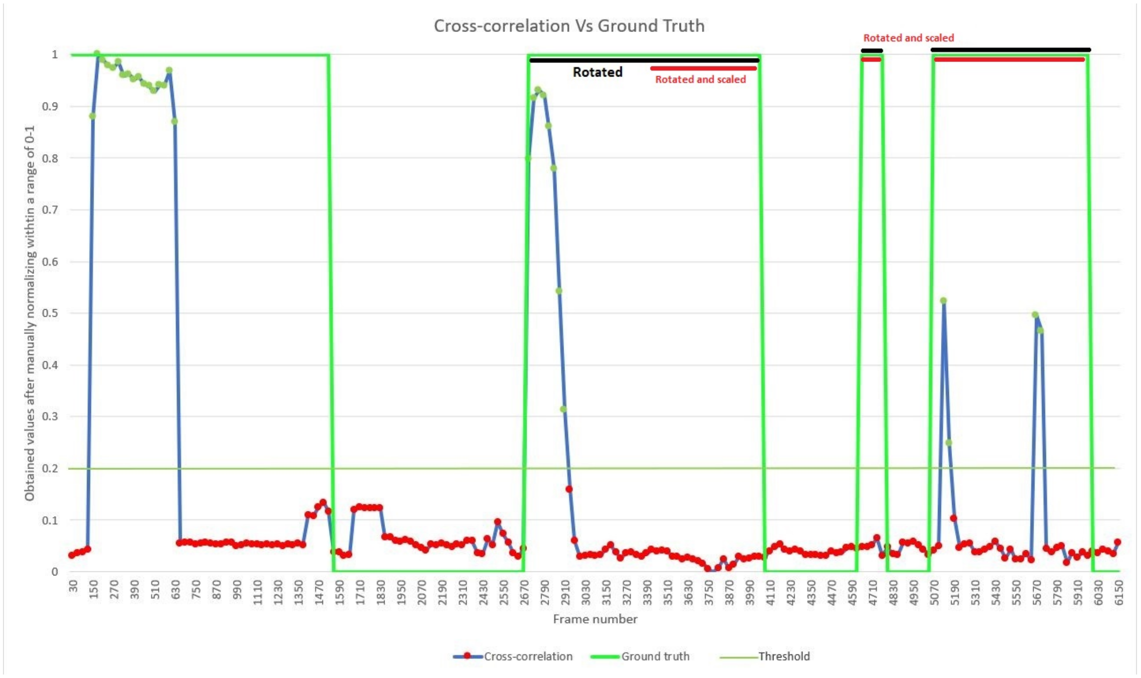


Figure 5.21: The frames with values equal or more than threshold 0.2 are distinguished as the frames that are likely to contain the template. The obtained values are compared with manually entered ground truth results as shown in the figure.

Figure 5.21, the obtained values are normalized between 0 to 1 using a normalization equation (Equation 2.3). The obtained values are scattered within 0 to 1 as shown in the Figure 5.21 and a clear threshold can be selected by the user. Presumably, if the user selected 0.9 as the threshold for Figure 5.21, it can distinguish the likely and unlikely frames, as the frames with values higher than 0.9 are more likely to contain the template. When the obtained value is compared with the ground truth as shown in the figure, it is evident that the cross-correlation distance measure fails to detect most of the rotated and scaled frames of the input video between the frames 3000-4110, 4590-4630, and most of the frames between 5070-6090.

Findings We can observe the limitations like missing to detect a scaled object that we noted while using the rotated and scaled soccer video as an input. Surprisingly, we can also observe that the correlation value increases between frames 5100 and 5190 and then between 5670 and 5790. Since these frames contained transformed video frames, we investigated the reason behind the increased likelihood within these frames. In both instances, the likelihood has increased when the camera reaches 0 degree angle while rotating or scaling, i.e., those are the frames when the camera comes to original orientation and scale for a brief moment before rotating and scaling it to another side. The cross-correlation function shows that the actual rotated, scaled, and even tilted frames are undetected. We conclude that values obtained by the cross-correlation function are not transformation invariant. The cross-correlation function results in false-negative for almost all the transformed video frames. Furthermore, the likelihood value using cross-correlation is neither calculated in scale space nor is angle oriented, so it is not rotation or transformation invariant.

5.2.2 Template Matching Using Coefficient of Cross-correlation Function

We used coefficient of cross-correlation function to compare the obtained likelihood values with the cross-correlation function (discussed in detail in Section 2.2.1). The obtained values are then normalized using the Equation 2.3 and plotted in the Figure 5.22 and discussed in the upcoming sections. To determine the working accuracy of the function, we

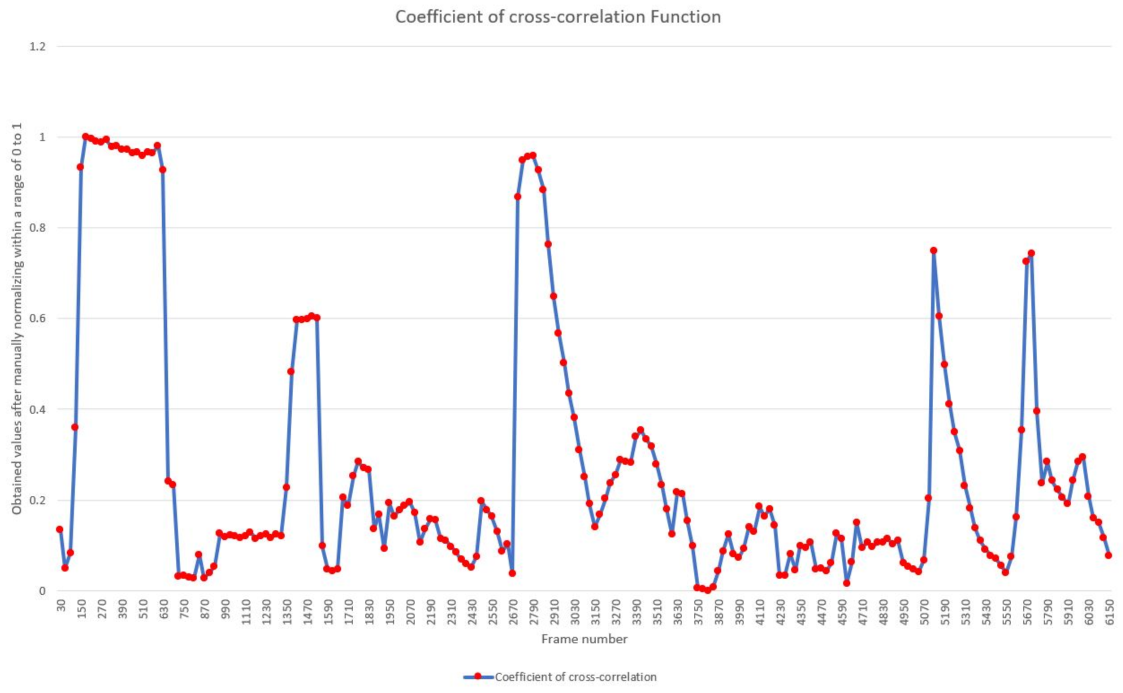


Figure 5.22: Matching data points in the input video using the coefficient of cross-correlation function in template matching.

plot the ground truth for each frame and compare it with the obtained results as shown in Figure 5.23. When we compare obtained values with the ground truth values in Figure 5.23, we observe that the mean-shifted cross-correlation function seems to have similar results to cross-correlation (as shown in Figure 5.21). The detected frames are similar to cross-correlation, including the frames 2670-3030 where the frames have higher likelihood value,and the value suddenly drops after 3030 as the frame starts rotating.

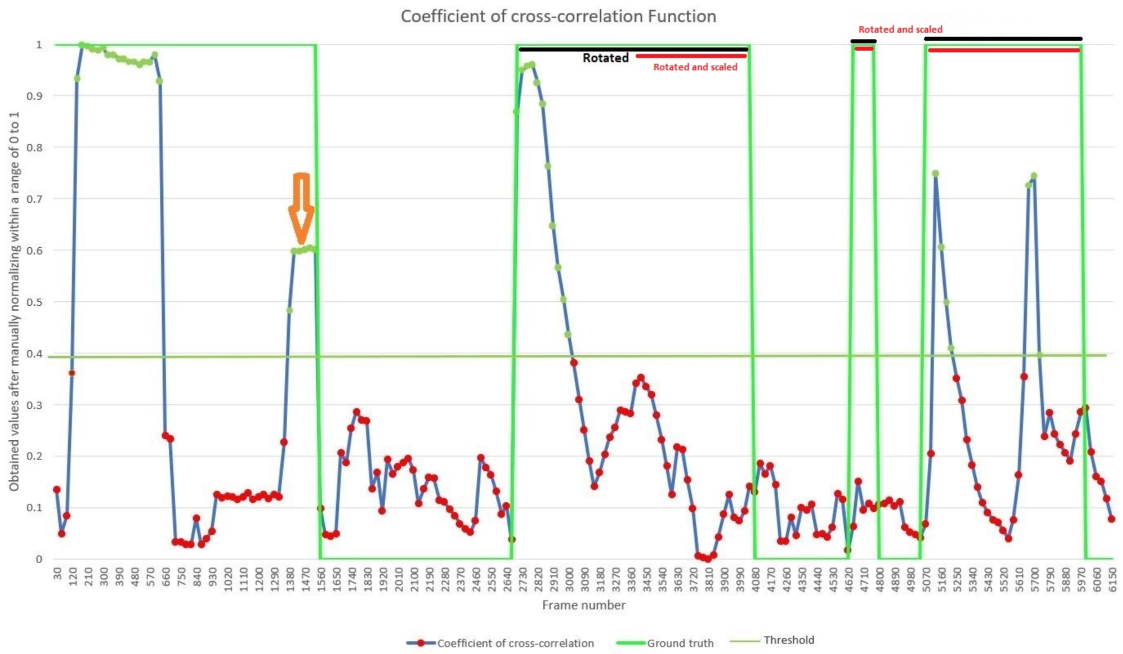


Figure 5.23: The frames with values equal or more than threshold 0.4 are distinguished as the frames that are likely to contain the template. The obtained values are compared with manually entered ground truth results as shown in the figure.

Findings From the Figure 5.23 and comparing the values, we note that the obtained values are similar to that of the cross-correlation function (as explained in 5.2.1), but with some frames that were even less distinctive than the cross-correlation function. We thus conclude that the mean-shifted cross-correlation function is not transformation invariant when compared to the ground truth. Minor differences between the two functions include the detection of frames 1350 to 1550 maked with an orange arrow in Figure 5.23. It may be because of the computer rounding functions as the coefficient of correlation function generally operates with the numbers closer to 0, coefficient being the sharpness of the cross-correlation function. This may lead to match the template with the target resulting in false-positive likelihood. Also, frames between 5100-5310, and 5650-5700 may contain likely frames because those are the frames when the camera comes to a steady position for a brief moment before rotating scaling it to another side.

Both cross-correlation and mean-shifted cross-correlation functions did not provide optimal results required to achieve transformation invariance. In the next section, we experiment and try to overcome the limitations by using the function, sum of squared differences between the template and the target.

5.2.3 Template Matching Using Sum of Square Difference Function

Finally, we have tested the template-matching algorithm with the third likelihood calculation function, Sum of Squared Differences (SSD). The results, along with the manually annotated ground truth values, are shown in Figure 5.24.

Findings The obtained results, as shown in Figure 5.24, provides likelihood value equivalent or greater to the cross-correlation function. Although clustered into likely and unlikely values, if a user selects a threshold between 0.2-0.3, we can observe that the output values may presumably contain false negatives within frames 0-150, 750-1150, 3250-3550, and 3650-3900. The scattered graph enables the user to extract a threshold and gives similar values when compared to cross-correlation and coefficient of cross-correlation function. Frames between 5100-5310, and 5630-5790 are detected because those are the frames when the camera comes to a steady position for a brief moment before rotating scaling it to another side. Also, while observing the undetected frames between 4620-4800,

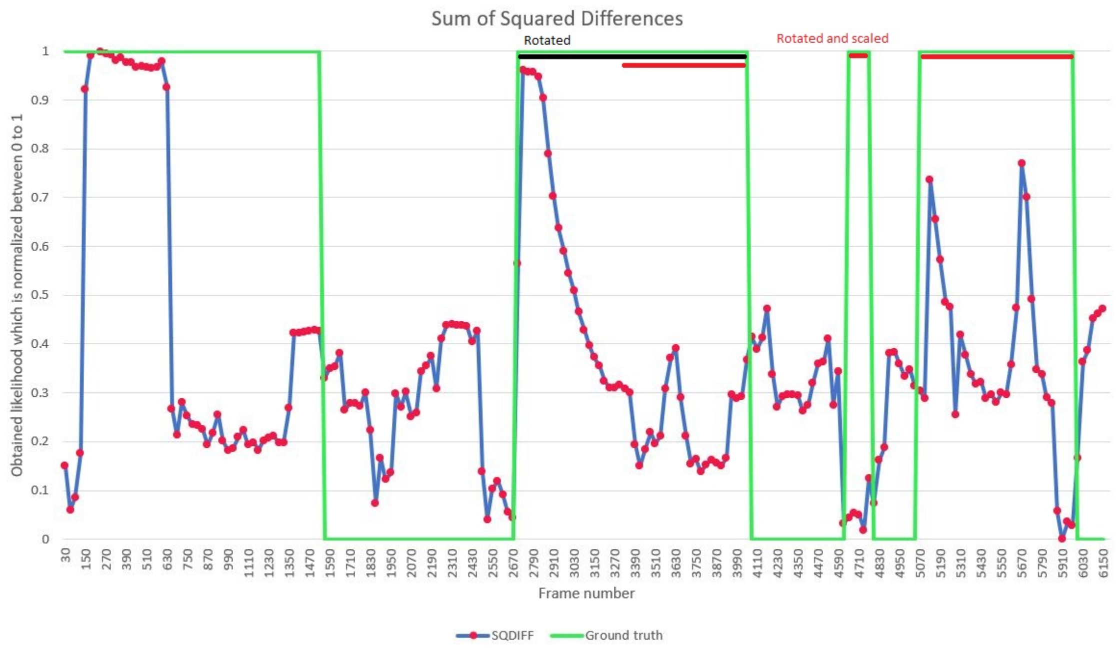


Figure 5.24: The values obtained by using SSD function. The obtained values are compared with manually entered ground truth results as shown in the figure.

which are the scaled frames, it is easier to assume that SSD function is not scale invariant.

While observing frames 1590-2500, which are supposed to be "unlikely" frames but the SSD may detect most of them as potential false positive frames when a threshold between 0.2-0.3 is selected in the Figure 5.24. Each point in the template calculates a difference between the template pixel and the target pixel, and when squared, the distance becomes greater, which may have potentially caused many false positives and true negatives among the obtained values. Some possible reasons behind the inaccurate obtained values can also be redundant background objects, illuminance, the transformation of the video frame and so on. This is because under the illuminance differences, there may be outliers (like color shade differences) which can result into a greater distance between similar objects within the template and the target.

We observe that this function result in many false positives and false negatives unlike in the cross-correlation approach which confuses the user while trying to determine a threshold without losing data. This comparison led us to conclude that the existing Cross-correlation is already a better choice for the existing template-based template matching approach. We will also discuss more observations in detail in Section 5.2.4. Until this point in research, we have not found any optimal solutions for the transformation invariance for single-camera videos.

5.2.4 Comparison of all Three Template-based Template Matching Functions

The obtained results are affected by many circumstances. Some of them are the scale and position of the video, illuminance, the fast movement of the players, video quality and so on. The use of different functions and computer computational abilities also fluctuates the effectiveness of different approaches in determining the likelihood of the image. Despite their ineffectiveness towards transformed image detection, we compare the outcome of all the three algorithms with the user-defined and observed ground truth values in Figure 5.25. From the Figure 5.25, we can conclude that neither of the template-based template matching functions worked as expected. We were unable to justify any of the template-based template matching as an optimal result for image transformation based on template matching using the template-based approach. Among all the three functions, SSD provided optimal results by matching more likely frames between 30-1550,2670-3990, and 5050-6000 with the single-camera soccer video. SSD also showed partial rotation invariance with some false positives which may be sufficient to question the effectiveness of the

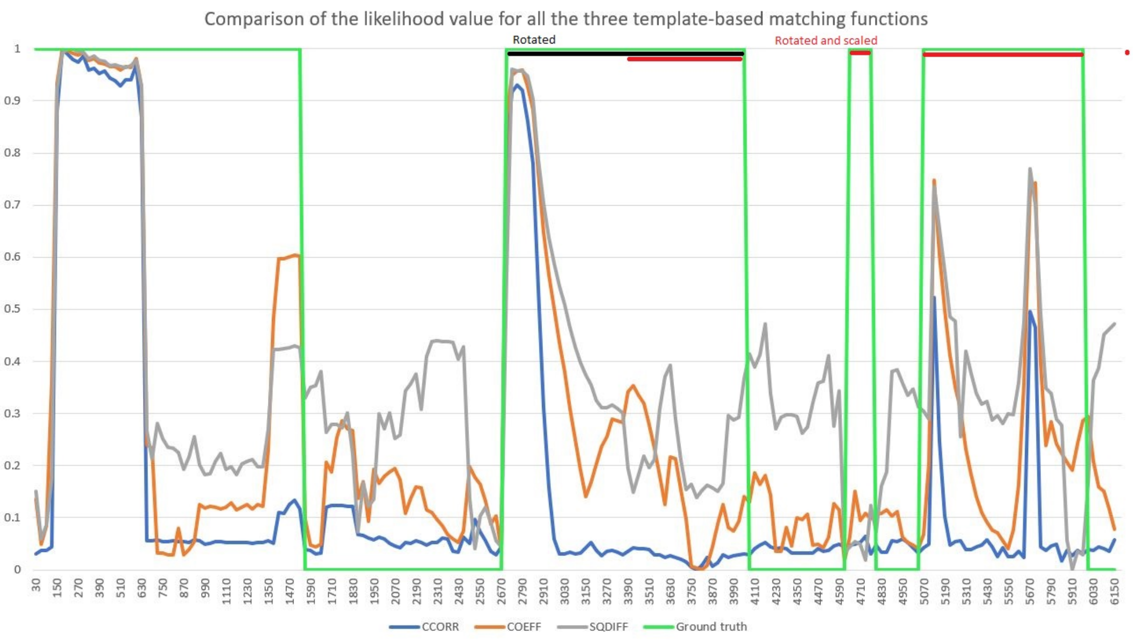


Figure 5.25: Comparison of the normalized likelihood values. Values obtained by using all three functions (cross-correlation, coefficient of cross-correlation and SSD) are compared with the ground truth values.

function. In an attempt to overcome the limitations, we also explored the possibility of using supervised learning algorithms for single-camera videos in the next section.

5.3 Exploration of Alternate Approaches

As discussed in Section 5.2.4, all three functions (cross-correlation, coefficient of cross-correlation, and SSD) based on template-based template matching algorithms could not achieve scale and rotation invariance for single-camera videos. We also explored the idea of using various unsupervised classification algorithms like Known Nearest Neighbour (KNN), Support Vector Machine (SVM), and decision trees/random forest, etc. But since the objective was to provide a tool for the users to determine the threshold, the idea of using unsupervised classification algorithms like clustering can cause ambiguous data while designing an interactive user system. Some fuzzy algorithms may cause ambiguity in the classification results, but ultimately, we intended to give the users full control for the final video extraction. So we dropped the idea of using an unsupervised classification algorithm from the preliminary study.

We also considered the usability of the self-supervised learning approach for this research alongside the supervised learning algorithms. Self-supervised learning approach is an unsupervised learning approach where the model creates the supervised learning task out of the unlabelled input data. For e.g., giving the model upper half of an image so that the model predicts the lower half of the image. But the self-supervised learning requires a lot of training data which is why we decided we would only consider further studying self-supervised learning algorithm if we could find datasets relevant to this research, like for supervised learning approaches. We wanted the proposed tool to support and ease the user's decision making processes while extracting the relevant video frames, which is why we experimented with the discussed alternate approaches.

5.3.1 Supervised Learning Algorithms

Supervised learning algorithms can classify data which we can possibly use to our advantage and the algorithms work the best when much training data is available. Supervised learning algorithms (as explained in section 2.2.3) train a learning model based on the labeled data. The supervised learning algorithm trains a model, and the model's accuracy may higher or lower based on the training provided to the model. In the absence of an adequate amount of training data, the model often fails to provide optimal results.

For this research, as training data for the related single-camera videos is not readily available, we explored the availability of the relevant training data.

Convolutional Neural Network(CNN)

CNN is a class of artificial neural network used to analyze visual data like images. The CNNs are the powerful machine learning models trained for object detection and identification at various levels [47]. Here, we used Tensorflow [48], an open-source platform that facilitates CNN, Faster RCNN, and other algorithms to train a model. For this experiment, the faster RCNN algorithm was used, which is an improved version of CNN. The RCNN is faster because it generates a feature map from only one image and avoids unnecessary convolutional repetitions. More detailed discussions were provided in Section 2.2.3.

For this study, the lack of training data is one of the main problems in machine learning algorithms. Data augmentation is the first solution that comes to mind to maximize the size and increase the effectiveness of the training [49]. Although data augmentation in machine learning allows the expansion of the current dataset, it was difficult to collect even an initial smaller dataset that we require for this study in the first place. The idea of data augmentation is better used when we have to create a dataset with minimum data to start with. For this thesis, finding relevant data and labelling a small dataset seemed time-consuming for single-camera soccer videos. We searched the online repositories and dataset libraries but did not find an appropriate dataset to train our predictive model. The datasets were either for popular classification research problems like object detection or contained unrelated angles. One example can be the images in a professional soccer game where multiple cameras is used from different angles to better capture the game and this was unrelated to this research as shown in the Figure 5.26. Because of all these reasons, we sidelined the idea of data augmentation and started taking screen shots of the relevant single-camera videos to extract the images. These object in these images are now labeled to create a custom dataset and train a supervised learning model. The Figure 5.26 is an

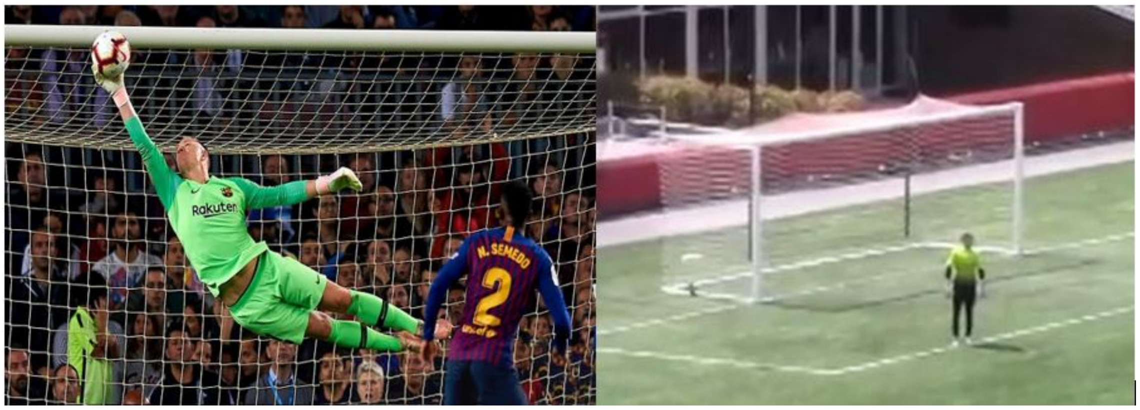


Figure 5.26: Here we compare an image downloaded from the internet with a single-camera soccer video frame that is representative for this research.

typical example of the labeled data found on the internet when compared to the frame excerpts from our single-camera videos. We can observe that the example data found were not good representatives of the views of the goals in our video frames. Our earlier tests show that these example data did not produce any acceptable results of locating the goals in our video frames. Therefore, we decided to move ahead with the idea of creating our custom dataset by labeling them manually. The first step to create a dataset was to collect the relevant data from any sources available. We needed different kinds of images related to soccer from single-camera videos, relevant angles, goalpost-focused images, and goal moments. We collected the single-camera videos, mostly of a local club game or a school

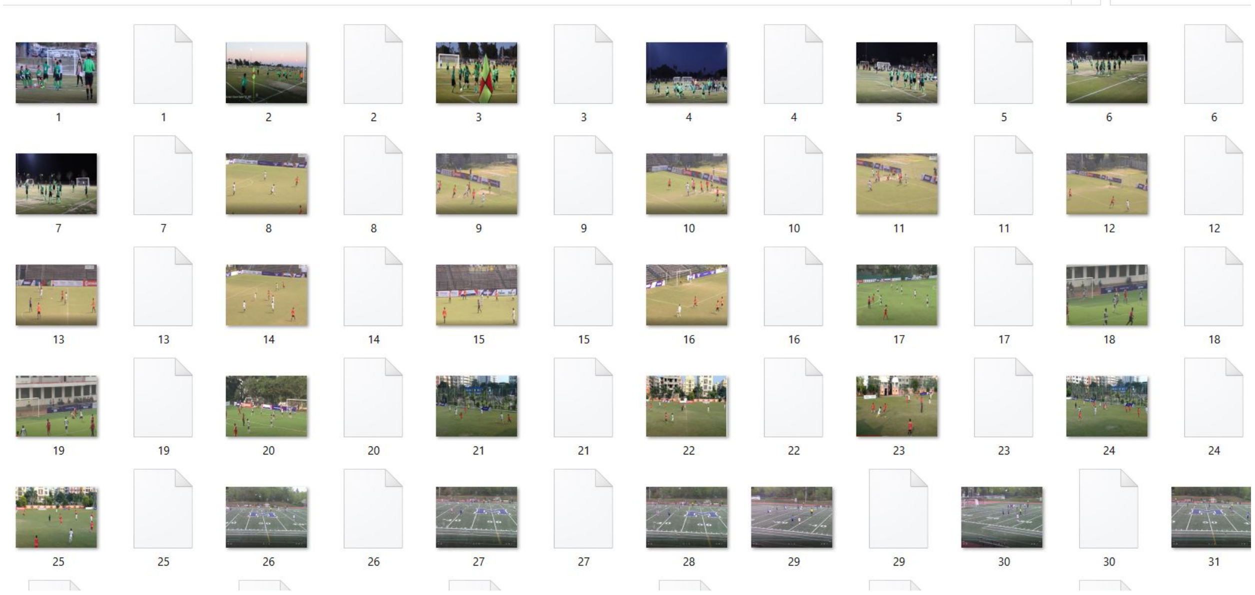


Figure 5.27: Set of training images that were collected by screen shots of the single-camera soccer videos. Each image contains multiple objects that were labeled and trained to create a custom model.

game captured by a single-camera video to extract the relevant image frames. From 10 single camera videos, we took 50 screen shots to extract them as images. A single image now contained multiple objects like players, balls, the referee and so on. We labeled all objects possible in each image which would give us multiple labeled objects to train the model better. We used labelImg [50], an annotation tool, to create the labeled dataset as shown in Figure 5.29. We labeled the dataset in PascalVoc format [51]. We used three distinct object labels; player, goalpost, and ball for the labeling process as shown in Figure 5.28. We used these labels because one of the application areas of our study was to analyze the goalkeepers performance in a single-camera soccer video while overcoming the limitations of template-based template matching algorithm. For this, we trained our custom model to be able to classify an input object into these three labels based on its training. A single image contained multiple objects to be labeled as shown in the labels in

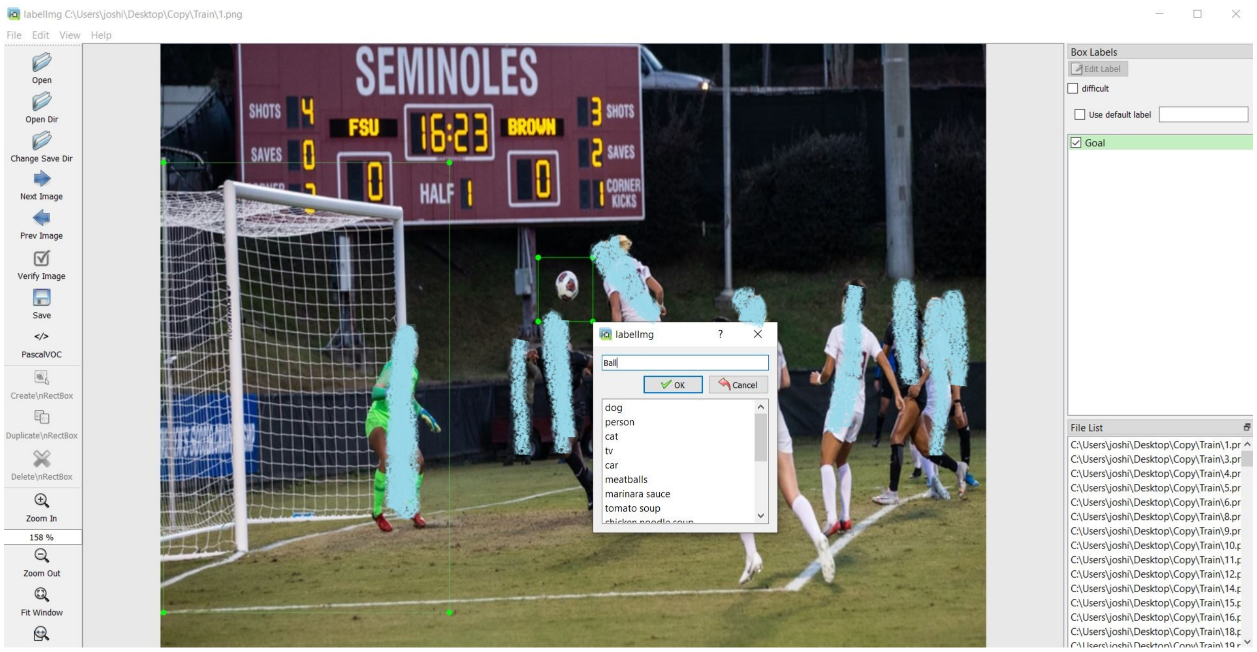

labelImg C:\Users\joshi\Desktop\Copy\Train\1.png
File
Edit
View
Help
Open
Open Dir
Change Save Dir
Next Image
Prev Image
Verify Image
Save
PascalVOC
Create\nRectBox
Duplicate\nRectBox
Delete\nRectBox
Zoom In
158 %
Zoom Out
Fit Window
Box Labels
Edit Label
difficult
Use default label
Goal
labelImg
Ball
OK
Cancel
dog
person
cat
tv
car
meatballs
marinara sauce
tomato soup
File List
C:\Users\joshi\Desktop\Copy\Train\1.pr
C:\Users\joshi\Desktop\Copy\Train\3.pr
C:\Users\joshi\Desktop\Copy\Train\4.pr
C:\Users\joshi\Desktop\Copy\Train\5.pr
C:\Users\joshi\Desktop\Copy\Train\6.pr
C:\Users\joshi\Desktop\Copy\Train\8.pr
C:\Users\joshi\Desktop\Copy\Train\9.pr
C:\Users\joshi\Desktop\Copy\Train\10.p
C:\Users\joshi\Desktop\Copy\Train\11.p
C:\Users\joshi\Desktop\Copy\Train\12.p
C:\Users\joshi\Desktop\Copy\Train\14.p
C:\Users\joshi\Desktop\Copy\Train\15.p
C:\Users\joshi\Desktop\Copy\Train\16.p
C:\Users\joshi\Desktop\Copy\Train\18.p
SEMINOLES
SHOTS 4
FSU
16:23
BROWN
3 SHOTS
SAVES 0
0
HALF 1
0
2 SAVES
1 CORNER KICKS

Figure 5.28: Interface of the LabelImg [50] anootation tool where we can choose to label multiple objects at once using different labels.

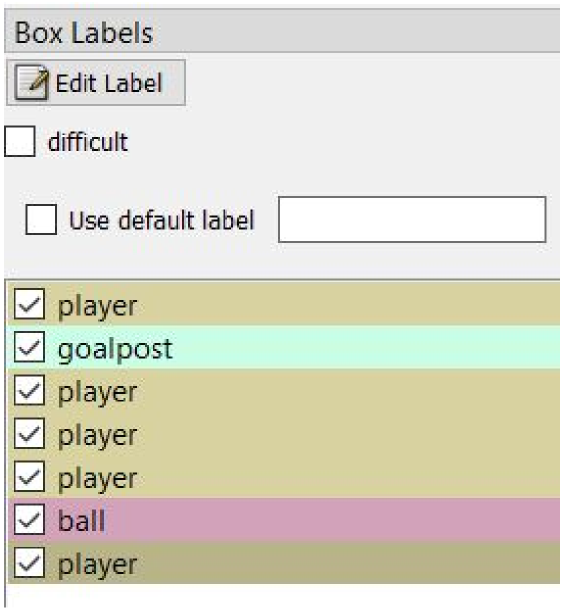
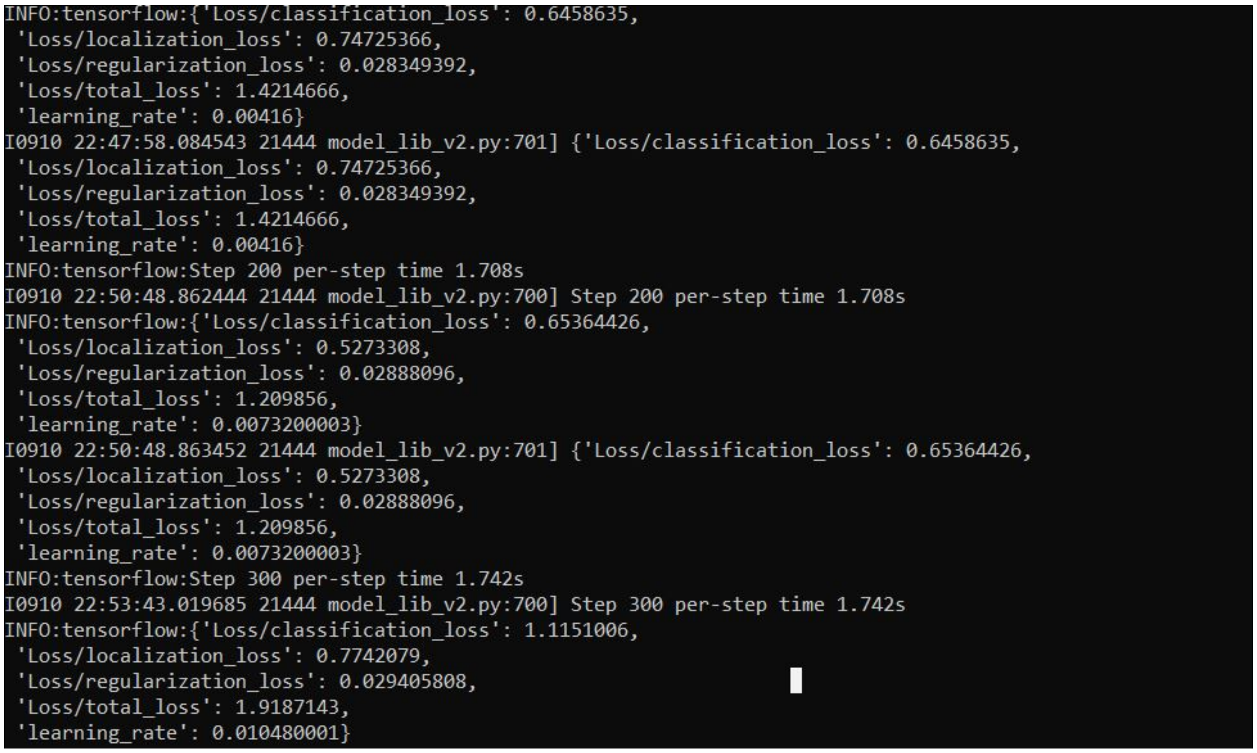


Figure 5.29: The four labels used to annotate the training images

Figure 5.29. The process was slow and time-consuming but we labeled 463 objects within the 50 images using the 3 labels (player, goalpost, and ball) for the preliminary study. To train the model, we split the labeled data into training and testing data in a rough 80:20 ratio. So we had 40 images with 374 labeled objects in the training dataset and 10 images with 89 labeled object in the testing directory. We started training the model with an initial learning rate of 0.00416 as shown in the Figure 5.30. The initial loss Alongside training, the model's performance increases directly with the availability of trained data. We changed the configuration for training based on EfficientDet – a recent family of SOTA



```
INFO:tensorflow:{'Loss/classification_loss': 0.6458635,
 'Loss/localization_loss': 0.74725366,
 'Loss/regularization_loss': 0.028349392,
 'Loss/total_loss': 1.4214666,
 'learning_rate': 0.00416}
I0910 22:47:58.084543 21444 model_lib_v2.py:701] {'Loss/classification_loss': 0.6458635,
 'Loss/localization_loss': 0.74725366,
 'Loss/regularization_loss': 0.028349392,
 'Loss/total_loss': 1.4214666,
 'learning_rate': 0.00416}
INFO:tensorflow:Step 200 per-step time 1.708s
I0910 22:50:48.862444 21444 model_lib_v2.py:700] Step 200 per-step time 1.708s
INFO:tensorflow:{'Loss/classification_loss': 0.65364426,
 'Loss/localization_loss': 0.5273308,
 'Loss/regularization_loss': 0.02888096,
 'Loss/total_loss': 1.209856,
 'learning_rate': 0.0073200003}
I0910 22:50:48.863452 21444 model_lib_v2.py:701] {'Loss/classification_loss': 0.65364426,
 'Loss/localization_loss': 0.5273308,
 'Loss/regularization_loss': 0.02888096,
 'Loss/total_loss': 1.209856,
 'learning_rate': 0.0073200003}
INFO:tensorflow:Step 300 per-step time 1.742s
I0910 22:53:43.019685 21444 model_lib_v2.py:700] Step 300 per-step time 1.742s
INFO:tensorflow:{'Loss/classification_loss': 1.1151006,
 'Loss/localization_loss': 0.7742079,
 'Loss/regularization_loss': 0.029405808,
 'Loss/total_loss': 1.9187143,
 'learning_rate': 0.010480001}
```

Figure 5.30: The model enters the training mode.

models [52]. We trained the model until it reached a steady low loss rate as shown in the Figure 5.31. For the preliminary investigation, the learning batch was reduced to 1 to reduce the overall run-time. From the Figure 5.31, we can observe that the loss value increase slightly at the first few epochs. The loss function then decreases after a few epochs. It can mean that the model was overfitted by the training data. We made sure to not use the same data for testing and training, the overfitting may also be because of the smaller and less diverse dataset. We can only determine if the model was overfitted after we test the model. We use a single-camera soccer video to test the model and the results

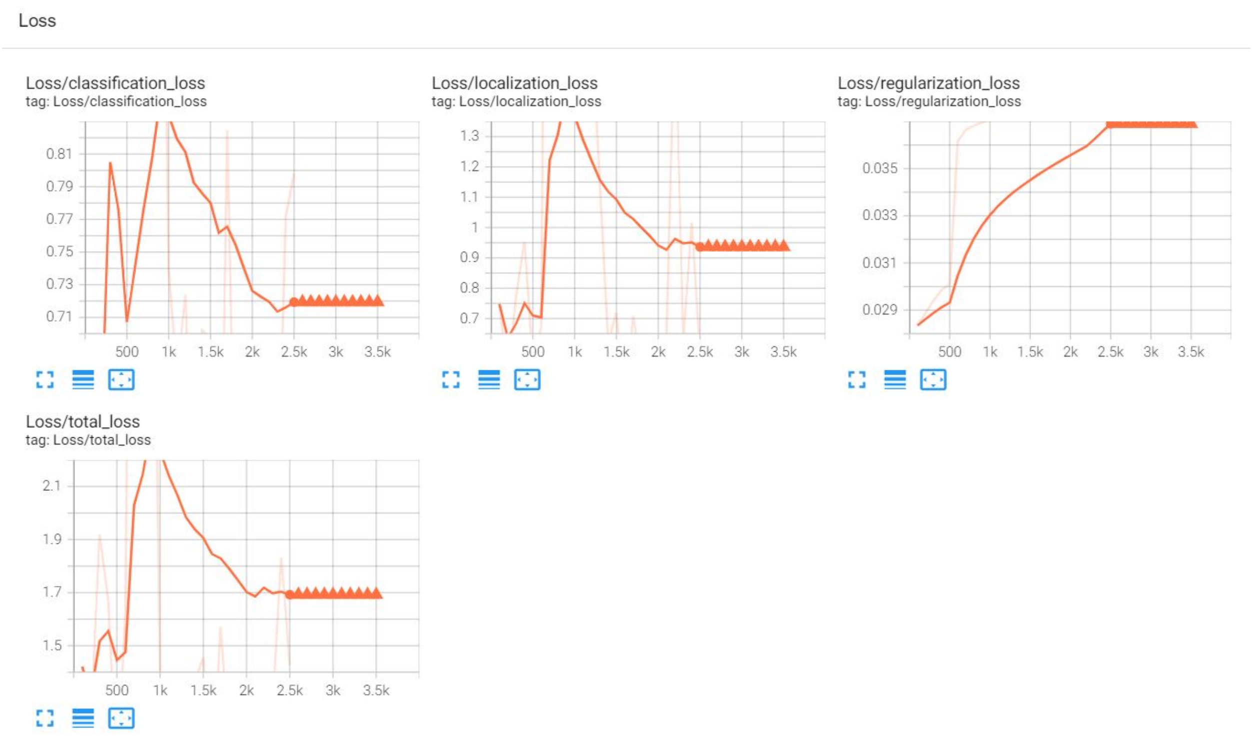


Figure 5.31: The model enters the training mode.

are shown in the Figure 5.32 and 5.33. We observe that, for the training model the players are detected with an accuracy of average accuracy of 97% and the goalpost is detected with an accuracy of 63% (shown in Figure 5.32). The testing data shows a lower accuracy of 60-80% for players and the goalpost is also undetected. This may have happened because of the overfitting. One way to avoid overfitting is by training the model with a larger and diverse dataset. Another way we can prevent overfitting is by using data augmentation.

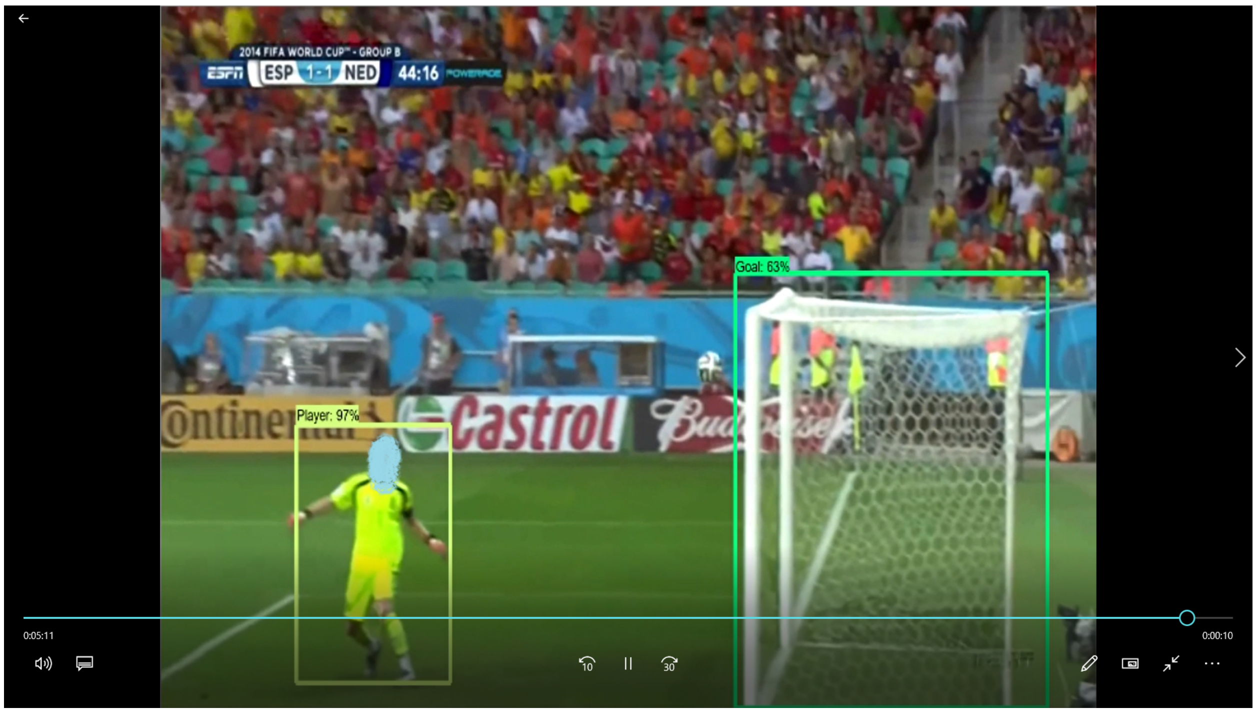


Figure 5.32: Detection of the goalpost using the RCNN model in tensorflow

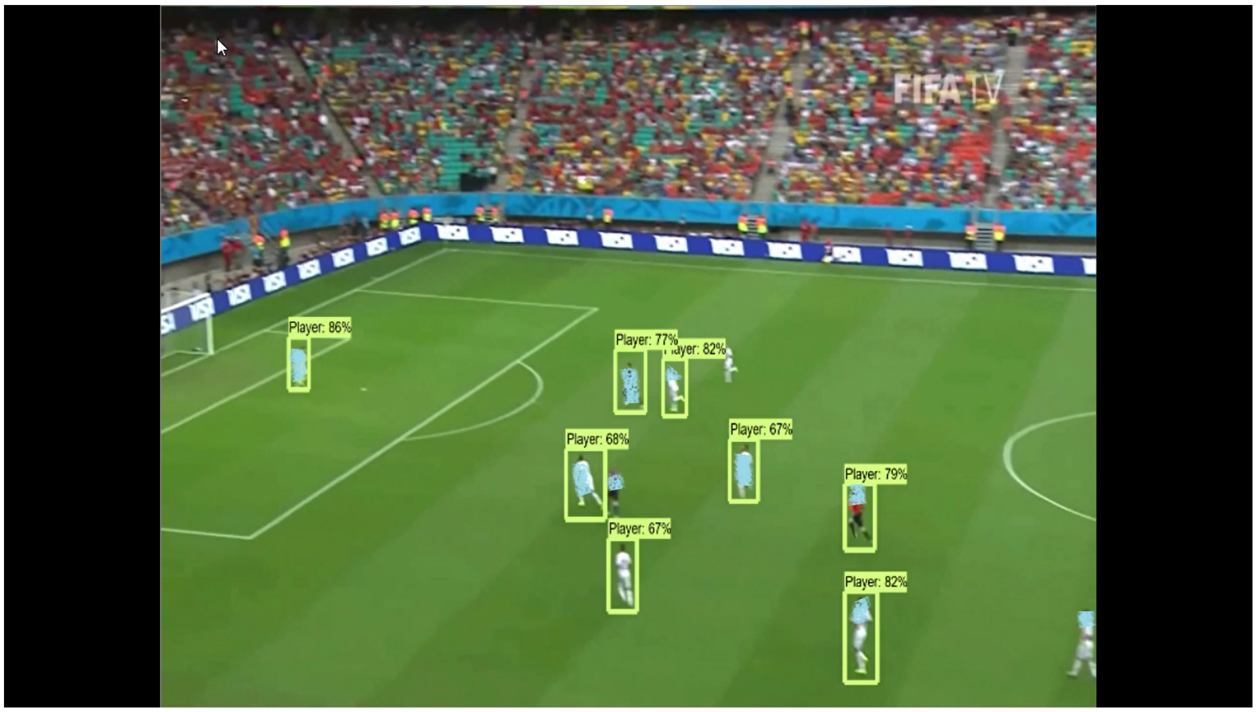


Figure 5.33: Object detection using CNN in tensorflow

Due to the lack of relevant dataset, however, excessive training and augmentation is not feasible for this research. Even after preventing the overfitting and being able to classify the objects with adequate training, challenges to keep the interactive interfacing intact still exists.

Although we observed in this preliminary experiment that it was quite difficult to train an optimal and useful model with the limited amount of training data, the current research and its resulting tool can help the users label video objects and thus create such datasets. Finding an algorithm that narrows down the template containing frames (like template-based template matching does) can help increase the precision of data augmentation if we decide to use machine learning algorithms in near future.

You Only Look Once (YOLO)

YOLO (You Only Look Once) is a real-time deep learning approach discussed in detail in Section 2.2.3. YOLO does not require any training for the custom model as we can download the trained model from the respective repositories on the internet. YOLO works faster with darknet that runs only in Linux OS. As our problem domain states the lack of training data, we also studied YOLO as an alternative to the CNNs. The YOLO model is trained on labels in the COCO dataset and the pre-trained model comes with a weight file along with the configuration file that runs the algorithm. We only have to create a deep learning framework for the model, providing support for the system to run the algorithm. We used the OpenCV framework and made sure we have at least the 3.4.2 version to move forward with YOLO. We must use CPU instead of the GPU on this model as the videos will be processed in real-time, which make the processing light on our system.

Since we are unable to custom train a model due to the lack of relevant dataset, we used the pre-trained model to extract the detected object labels which could help us extract the relevant frame. We used the yolov3 weights, yolov3 configuration, and COCO names files. The weight file contains the trained model as the configuration file holds the algorithm with the dataset names containing the labels trained on the model. The COCO dataset contains two pre-trained labels, person and ball which is relevant to our study. However, the pre-trained model is missing the label related to goalpost (or goalkeeper/ goalkeeper area) which strays this approach from our study. When we run the YOLO model with our single-camera soccer video the object labels are detected in each video frame as shown in the Figure 5.34.

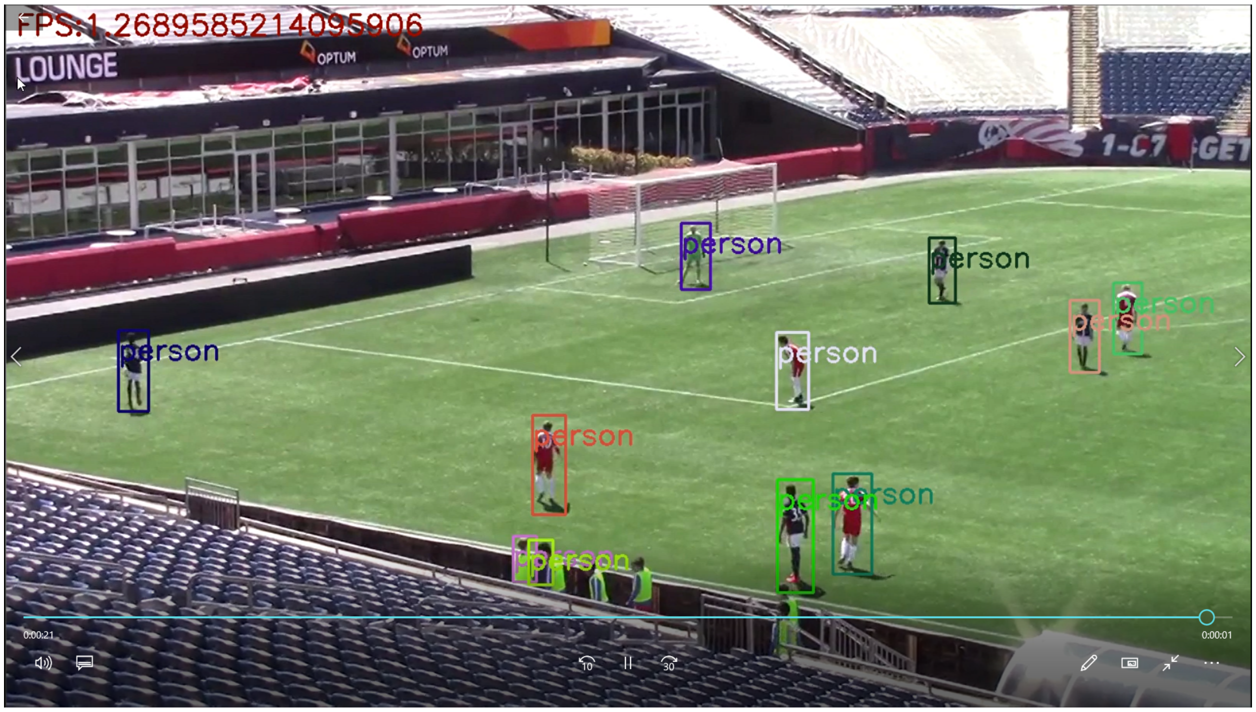


Figure 5.34: Object detection using YOLO pre-trained model.

We observe that all the players are detected with the label person but the possible relevant label (which is the goalpost area for this research) is missing in the coco dataset, which is why the system is unable to implement the pre-trained model to extract the single camera soccer videos for goalkeeper performance analysis. We have to custom train the model, for which we lack relevant dataset.

Findings The initial problem while working with a supervised algorithm is the unavailability of the relevant dataset. But while working with a pre-trained model like YOLO, it is only important to test the input video on the model. If the labels from the COCO dataset are detected accurately with the test video on an initial run then the next step is to determine if the subject relevant label is included within the trained labels. If not, you have to train a custom model like with CNNs and it was not feasible to do so for this research due to the lack of relevant training dataset. We conducted an initial test on the soccer video which detected the players as persons and then extracted the labels into a file. If the goalpost area or the goalkeeper (soccer related) was one of the labels in the pre-trained models, we might have been able to detect and extract the relevant frames containing the goalpost area or the goalkeeper. In this experiment, we did not find a goalpost or goalkeeper relevant pre-trained label because of which we were unable to proceed with the YOLO approach.

Final Observation

In this preliminary investigation, we examined a supervised learning algorithm by training a model by splitting the dataset into training and testing data. It then trains the model with training data and then feeds testing data to calculate its accuracy. The training process was the main hurdle for the single-camera videos due to the lack of labeled training datasets available even after a thorough search of the available resources. As discussed in detail in this section, using the supervised learning algorithms based on the limited amount of training dataset failed to provide the knowledge base used to build a solid framework for this research. We concluded that the required dataset for the research should be created, organized, and labeled manually. It took 15 hours just to find the single-camera videos and 5 hours to create and analyze the screenshots that were to be used for creating the custom dataset. It took another 15 hours to clean, scale and label all the data before splitting the dataset into training and testing data. Finally, we trained the dataset for 10 hours until we got the steady loss function to find the overfitting issue.

YOLO approach lacked pre-trained labels relevant to our problem domain (the goalkeeper performance). From the preliminary results, we were unable to reach a feasible solution with the supervised learning algorithms and we concluded that supervised learning algorithm would work better in scenarios with abundant training data to avoid any overfitting. There are other algorithms based on template matching that can give effective results for our subject matter and can contribute to the creation of a relevant dataset for the supervised machine-learning approach in the near future. Such approaches may potentially provide an insight to create a precise narrowed down dataset for data augmentation.

Due to the lack of relevant dataset, it became certain that we could not proceed with either supervised or self-supervised learning approaches. We lack the labelled data that self-supervised learning approaches needs to infer and fine-tune the obtained results for self-learning. Next, we decided to study feature-based template matching to find the traditional computation algorithms closest to template-based matching approaches that may work and will also keep the existing template-based user-interactive system intact. If the feature-based template matching was successful in overcoming some of the limitations of the template-based template matching, we could analyze the matched frames to create a better dataset for machine-learning algorithms eventually. In the next section, we discuss the use of feature-based template matching in an attempt to overcome the existing limitations (scaled and rotated frames, illuminance, fast movement of players and so on).

5.3.2 Feature-based template matching

Feature-based template matching is a similar approach to template-based template matching, and its approaches and algorithms are known to offer distinctive transformation invariance (as discussed in Section 2.2.2). To address the issues of the current template-based template matching approaches that we identified in Section 5.2.4 and Section 5.3.1, we explored feature-based approaches of template matching. The objective of this investigation of feature-based matching was to achieve transformation invariance while keeping in mind the interactive interface where the users will choose the template. There are two approaches of feature-based template matching that show promising results to fulfill our objective. They are:

- Brute Force Matcher (BF-based matcher) [33]
- Fast Library for Approximate Nearest Neighbors Matcher (FLANN-based matcher) [53]

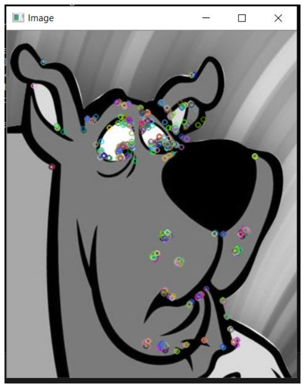
Object detection using feature-based template matching uses many algorithms as discussed in section 2.2.2. We only discuss three selective algorithms that have transformation invariance for this research, and they all work under both the BF-based and FLANN-based approaches. The algorithms are:

- Oriented FAST (Features from Accelerated and Segments Test) and Rotated BRIEF (Binary Robust Independent Elementary Feature) (ORB)
- Scale-Invariant Scale Transform (SIFT) and using Ratio Descriptor Test
- Speeded-Up Robust Features (SURF) using Ratio Descriptor Test

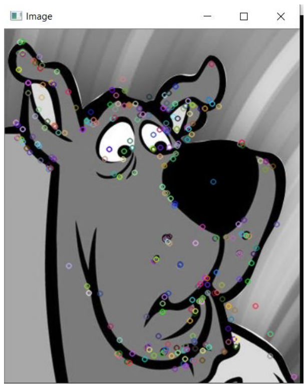
We will start this section with the experiment of all the three feature-based template matching algorithms, using the BF-based matcher approach. We will discuss the obtained results and list all our observations in detail. After experimenting with the BF-based matcher, we experiment with the same three algorithms using a FLANN-based matcher and discuss the observations. We also include a detailed research on feature-based template matching for the rotated and scaled (transformed) video frames. While studying feature matching, we try to overcome the limitations of the existing tool (such as scale and rotation invariance and illuminance issues).

All the three algorithms, ORB, SIFT, and SURF, have a similar working mechanism. We initially experimented with still images and then moved on to videos and video frames. When we run each algorithm, the keypoints are detected in both the template and target images. The keypoints are determined based on pixel depth (as explained in Section 2.2.2). We noted the difference in keypoint detection based on the algorithm used. The default value for the number of keypoints is 500 unless encoded otherwise (explained in detail in

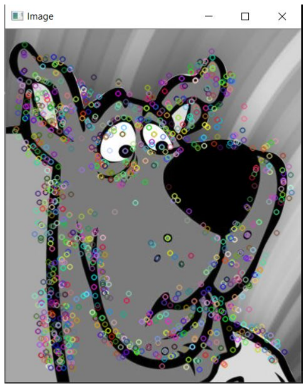
Section 6.2.2). The detected keypoints also depend on the approaches (i.e., BF-based matcher vs. FLANN-based matcher) used with each of the algorithms as shown in Figure 5.35. ORB uses pixel density to determine the keypoints whereas SIFT uses cascaded filters



(a) The circular multicolor keypoints in the image are detected using ORB.



(b) The circular multicolor keypoints in the image are detected using SIFT.



(c) The circular multicolor keypoints in the image are detected using SURF.

Figure 5.35: Comparison of number of keypoints detected in the same template by ORB, SIFT and SURF.

SURF with Gaussian filters to determine the keypoints and their descriptors. It happens partially because SIFT and SURF are scale-invariant, unlike ORB, which is only partially scale-invariant. Each keypoint has their unique descriptor that defines that particular keypoint. Descriptors are the visual features or the feature vector, which is 128–512 bits string, and they represent the image's distinct feature (i.e., an object, its edge, motion, or color). Finally, ORB, SIFT, and SURF algorithms all use different methods to define all the keypoints and their respective descriptors as explained in Section 2.2.2. All the keypoints are evaluated to carry a binary feature which is called a hash value which is stored in the descriptor buckets. For every keypoint descriptor, its matching buckets are retrieved depending on the distance between them. For each descriptor, now the bucket contains descriptors with a common sub signature (which is a binary value). The distance between the descriptors with common sub-signature is called the hamming distance [54]. The final likelihood value for each frame depends on the use of BF-based matcher and FLANN-based matcher (as explained in detail in Section 2.2.2). As in the case of the template-based approaches, the obtained likelihood values over all the frames are then normalized within 0 and 1, 1 being the highest possibility of the frame containing the template.

We start experimenting on feature-based template matching with the algorithms using BF-based approach, first on images. We then clip a short portion of the transformed soccer video to observe the preliminary results. We discuss the obtained results and observations in the next section.

Brute Force Matching

Brute Force Matcher is the feature-based template matching approach that uses feature detection to match the template with the target image. Initally, the keypoints and descriptors for each pixel in the template and the target image are defined. Each of the descriptors carry a binary value in the hashing table that acts as its signature. The descriptors are then compared with the target descriptors and their hash value. The descriptors containing similar values are paired and the distance between the pair and remaining pairs in the hashing table is calculated. The distance is called the hamming distance and it is calculated by using the Euclidean distance formula for the BF-based matcher. The similar descriptor pairs in the target and template image with the shortest hemming distance between them will be a final match.

When BF-based matcher is first initialized, all the algorithms (ORB, SIFT or SURF) define the keypoints and their respective descriptors for both the template and the target image (as explained in 2.2.2). We now start experimenting the BF-based matching while using ORB, SIFT and SURF algorithms respectively.

ORB using BF-based matcher

Oriented and rotated Fast (ORB) is comparatively faster than other two algorithms for brute force matching because it does not have to go through the filter ratio descriptor. Alongside using a pyramid to produce multiscale features, the BF-based matcher then applies Harris corner detection [55] to find the top N keypoints among the detected keypoints. One limitation of using FAST is that it does not compute the orientation. To overcome this, ORB was modified to calculate the intensity weighted centroid of the center-corner located patch that provides the direction vector to define orientation as explained in Section 2.2.2.

We matched the obtained descriptors of the template with the corresponding descriptors in the target image. The matched keypoints of the template and target images can be observed in Figure 5.36. Initially detecting 500 keypoints, ORB shortlisted about 20 best matching keypoints based on the shortest distance of the overlapping descriptors. The process was quick, and the observed results were rotation invariant. There were a few false positive matches presumably because of the similarity in lighting and color gradient, which happens often. While experimenting, we decided to use a real-time video and its frames as

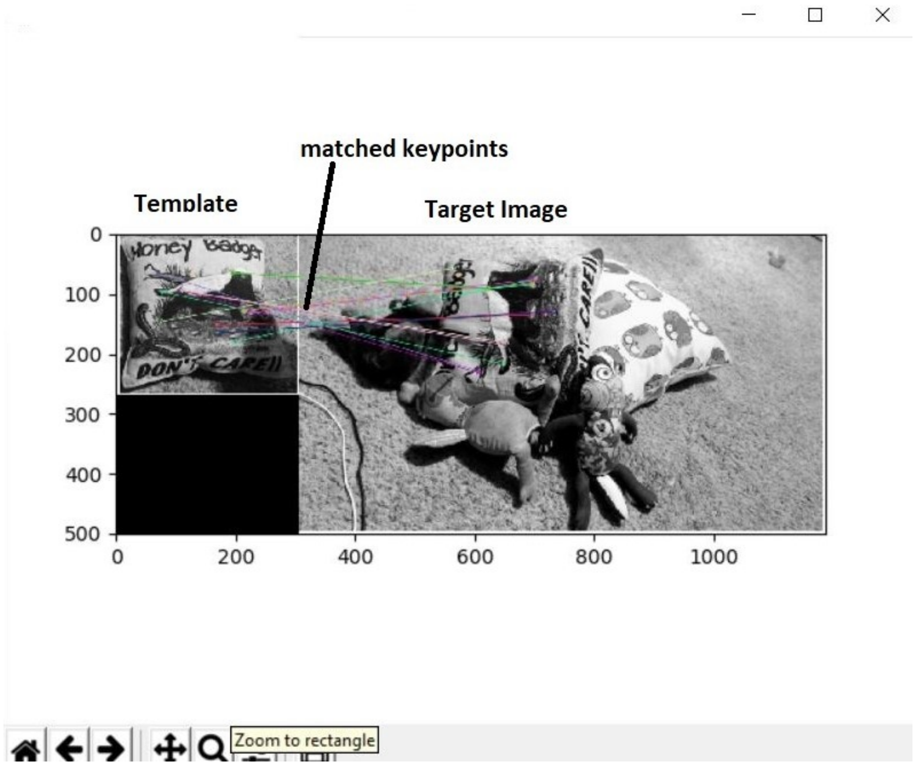


Figure 5.36: Rotated and scaled keypoints matches using the BF-based matcher with ORB descriptors. The multicolor lines represents the matched similar keypoints among the template on the left and the target image on the right.

the target image. We used a still image of a notebook as a template and fed a live video as the target video frames. When we opened the webcam holding the notebook and moved around (to scale and rotate the target image), the system failed to detect the notebook in the resultant video, which is the template. We observed that the model has difficulty matching the template with the corresponding target. We did this experiment to see if the BF-based matcher works well with video as much as it does with the still images. We observe that there were no feature that were matched when we used the real-time video to move around. It may be because of video's constant movement causing the object to scale and rotate, all at once. It can also be because of illuminace, and the difference in resolution between the template and the target frame.

The matched lines in Figure 5.36 BF-based matcher using ORB was successful to match still images but failed to match a video. We moved on to study the Scaled Invariant Feature Transform (SIFT) using BF-based matcher to observe if the number of detected keypoints is increased which can possibly contribute to more keypoints matching.

SIFT/SURF and descriptor test using BF-based matcher

Scale Invariant Feature Transformation algorithm can detect keypoints accurately and their respective descriptors for an image/object via cascaded filters (as explained in detail in Section 2.2.2). Cascaded filters are interconnected filters, where the settings of the parent filter limit the passing of features available on the child filter on the same dashboard. These filters get utilized to detect the scale-invariant characteristics of an image. These keypoints determined in the space show where the Difference of Gaussian (DoG) is calculated in scale space progressively. We summarize the detailed work-flow of the algorithm in the flowchart below (in Figure 5.37). Conducting the ratio test on all the keypoints matches the keypoints between two images. Here, we used the Euclidean distance formula to calculate the distance between all the keypoints and compare the obtained values. If the nearest distance and second-nearest distance are too close, it may indicate an inconsistent value or a noise. In such a case, we take the ratio of second-closest and closest-distance. We discard the rest of the values after taking the ratio value. It helps preserve 95% of the values that are true while discarding 90% of false positives [56]. This

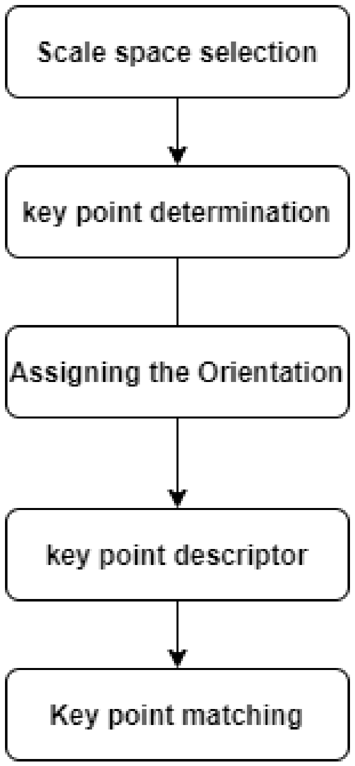


Figure 5.37: Steps involved in feature matching using SIFT

algorithm works slower than ORB because of the ratio test run-time, but it detects effective keypoints that identify the features more precisely. SIFT does not lead to prior segmentation of the object and preserves the local features, making it robust to occlusion and clutter, and we can observe this in Figure 5.38. We repeat the same process as ORB while testing the SIFT descriptors. The first input was a random template image that created a feature object. This object was the tool that matched the descriptors of the template image with the target image, as shown in the given figure. We also observed rotation and illumination invariance if the threshold from the bin values gets changed into larger numerical values. For example, if number out of the 128 bin values holds a value greater than 0.2, the threshold = 0.2. Here, the algorithm matched more keypoints than ORB with minimum false matches, as shown in Figure 5.38.

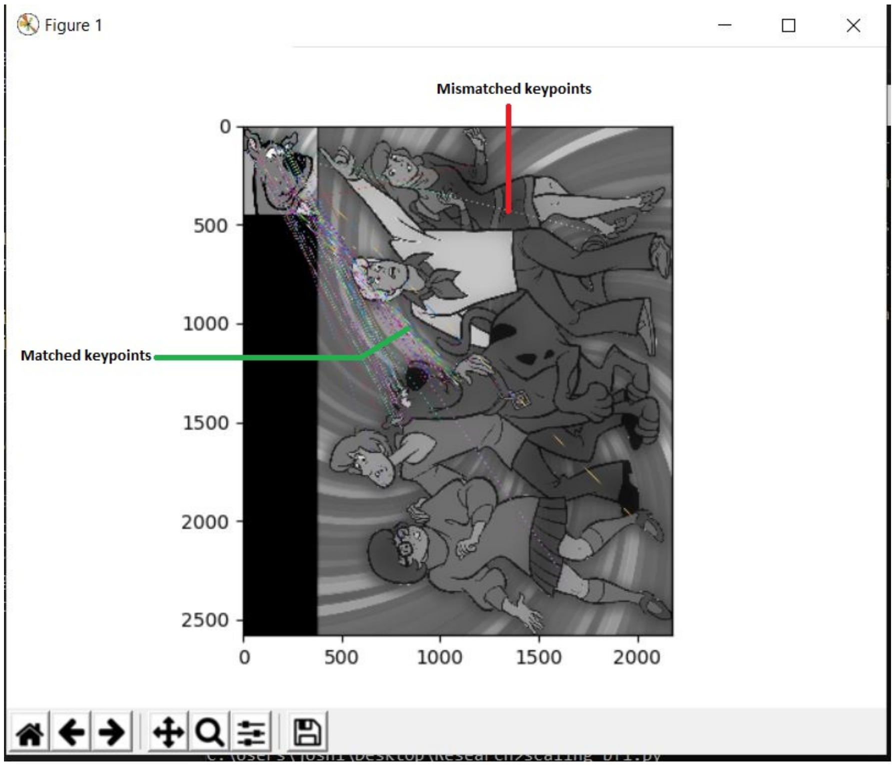


Figure 5.38: BF-based matcher with the matched and mismatched keypoints between the template and the target image. We can observe that there are only a few mismatched keypoints.

Findings Running a series of images through the BF-based approach for feature-based template matching shows promising results for still target images. For video frames, BF-based matcher encountered difficulty in execution due to numerous reasons, as explained in 2.2.2. For SIFT algorithm, we observe that the Brute Force approach is not executed as efficiently on videos. It may be because of the fast and multi-directional movement of the object in videos, unlike in still images. It may also be due to illuminance, resolution, and many other factors. We also observed that only a few frames matched the template with the input video, whereas most frames missed the descriptor entirely. From these observations, we concluded that the keypoint localization entirely depends on the algorithm complexity and the keypoint orientation. According to the observations, the BF-based matcher approach is not a feasible solution to achieve transformation invariance for single-camera videos.

Furthermore, we also noted that SIFT works slower but exceeds in precision than ORB. Both ORB and SIFT are rotation invariant, and SIFT is scaling invariant, which makes it transformation invariant. SIFT and SURF use different keypoint localization method but both undergo ratio test to filter the good matches which is why testing either algorithm shows the basic feasibility and complexity of both SIFT and SURF. Although the BF-based matcher helps achieve transformation invariance, the results are not satisfactory when run on videos. To achieve better results using single-camera videos, we decided not to use the conventional BF-based matching but to focus our research on advanced FLANN-based approach.

FLANN-based Matching

FLANN (Fast Library for Approximate Nearest Neighbors) is an algorithm that uses the nearest neighbour technique in OpenCV. It is compiled as a library for performing a search based on the nearest neighbors in a larger dataset (as explained in section 2.2.2). For image recognition and data compression, the nearest neighbor allows the search queries to skim through the bulky datasets by filtering for the nearest probable value. Multimedia data contains image that contains pixels carrying a value that specifies the color, density, region of interest for that image. The presence of all these parameters can make the multimedia data analysis slightly more complex than numeric data. With numeric data and a fewer image frames, BF-based matcher can be significantly efficient. A FLANN-based matcher is an alternative approach to a BF-based matcher that is efficient for high-volume complex big data [57]. In high-dimensional spaces, despite being a time-consuming algorithm, nearest-neighbor matching preserves the data resulting into the minimal loss of accuracy [58]. FLANN approach also uses the three feature detecting algorithms (i.e., ORB, SIFT, and SURF algorithms) to find the closest proximity and increased accuracy with less time consumption.

FLANN-based matcher approach uses a K-Dimensional (KD) tree to locate and match the corresponding descriptors between a template and a target image. The KD tree stores the trained descriptors so that it is faster to find the most similar descriptor when performing the matching to give optimal results (as explained in section 2.2.2). In this step, we first used the FLANN-based matcher on the same image as we did for the BF-based matcher (discussed in Section 2.2.2). Among all the three algorithms used, we found a significant increase in the precision of the keypoints and descriptors detected and matched when SIFT and SURF are used with the FLANN-based matcher, as shown in the Figure 5.39. The lines drawn from the template to target images shows the matched keypoints when using SIFT algorithm with FLANN-based matcher. We observe and note that the rotation invariance is also achieved with the target image from the given figure.

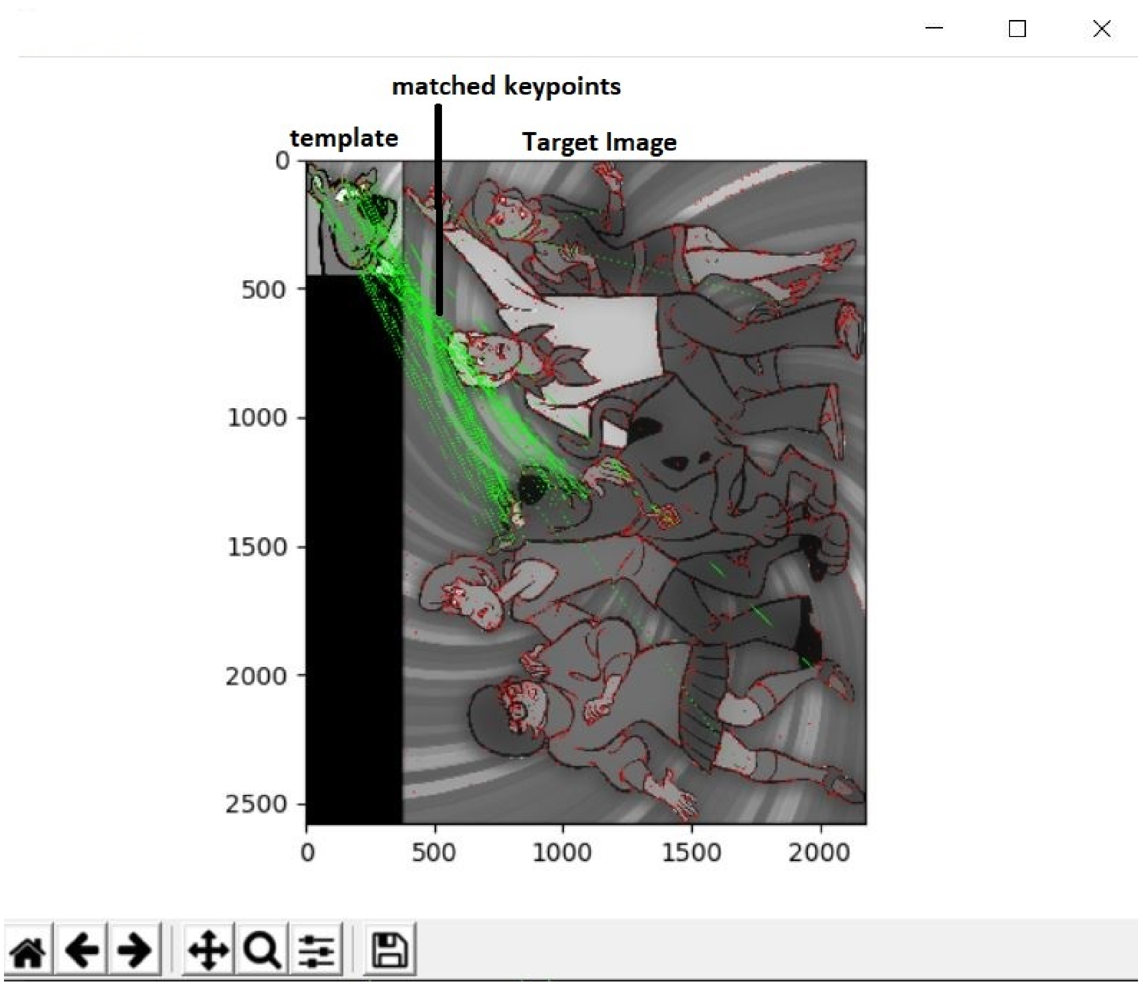


Figure 5.39: The green lines from the template to the target images are the matched keypoints while using SIFT algorithm with FLANN-based matcher.

After overcoming the limitations for images, it was necessary to achieve the same results for single-camera videos to attain our objectives for this research. We decided to use the manually transformed soccer video that we experimented on in Section 5.2.1. For this preliminary research, we decided to clip the soccer video into a short video of 42 seconds, containing 1260 frames. We did not skip any frames to obtain efficient preliminary results and also to observe each algorithm's precision. The user selects the goalkeeper areas as the template for all experiments, as shown in the Figure 5.40. We used the same video for all the three algorithms (i.e., ORB, SIFT, and SURF) and the observed results are discussed in detail in rest of this section.

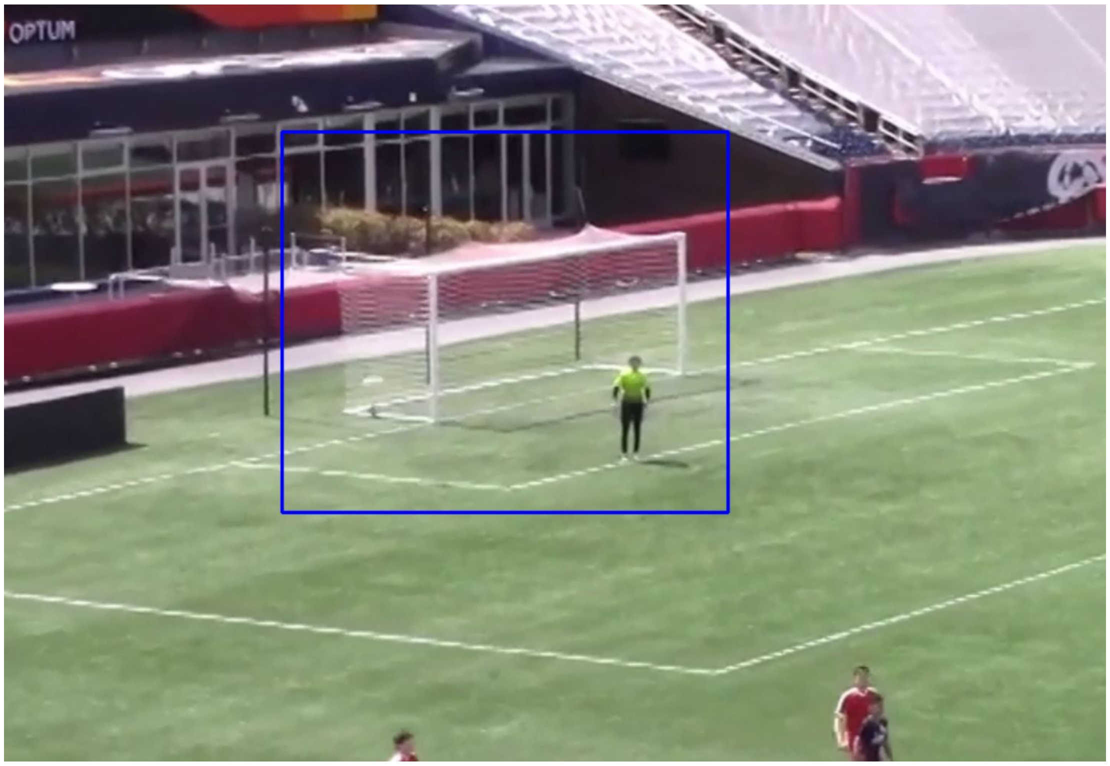


Figure 5.40: The blue area represents the template selected by the user. The user uses a mouse to select a rectangular area as a template fed to the system. As we are focusing on the soccer videos we select the goalpost area as the template as shown in the figure.

ORB using FLANN-based approach We use ORB algorithm to detect the keypoints and define their respective descriptors. Multiple keypoints are then matched through a FLANN-based matching approach. The keypoints are now filtered with a ratio (discussed in detail in Section 6.2.3) and only good matching pairs are now extracted as the final matching keypoints. We noted that the algorithm matches the features holding the minimum distance accurately from the template to the target image as shown in the Figure 5.41. The colored lines represent the matched features from the template to the target frame on the right-hand side. It then draws the blue rectangular area that is the detected user-defined template in that particular video frame.

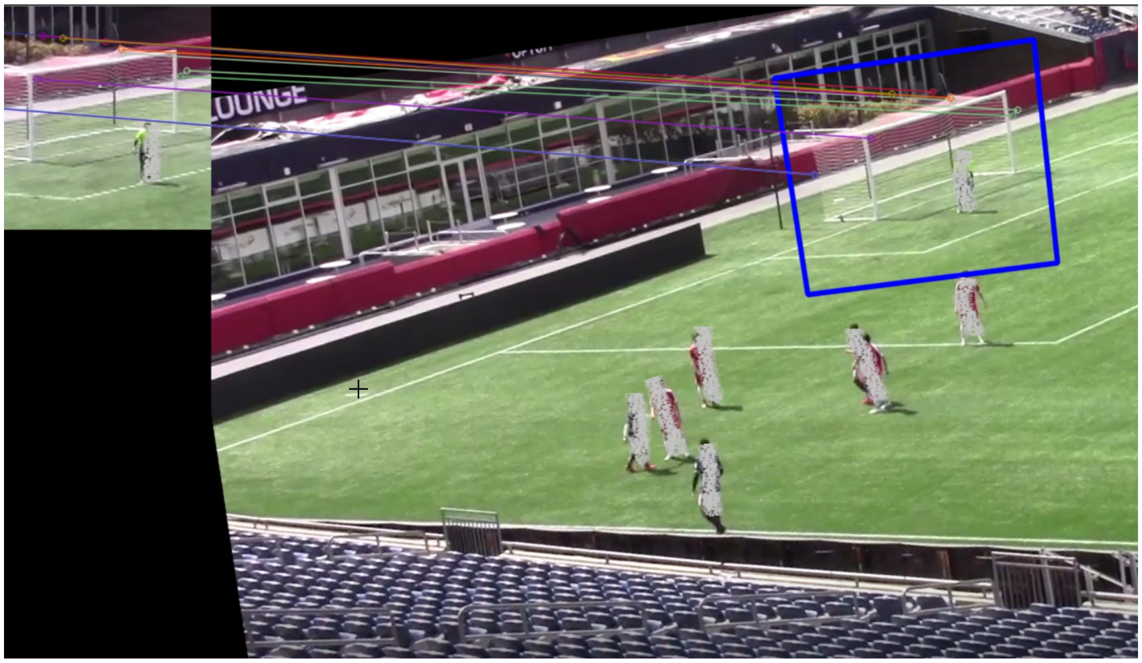
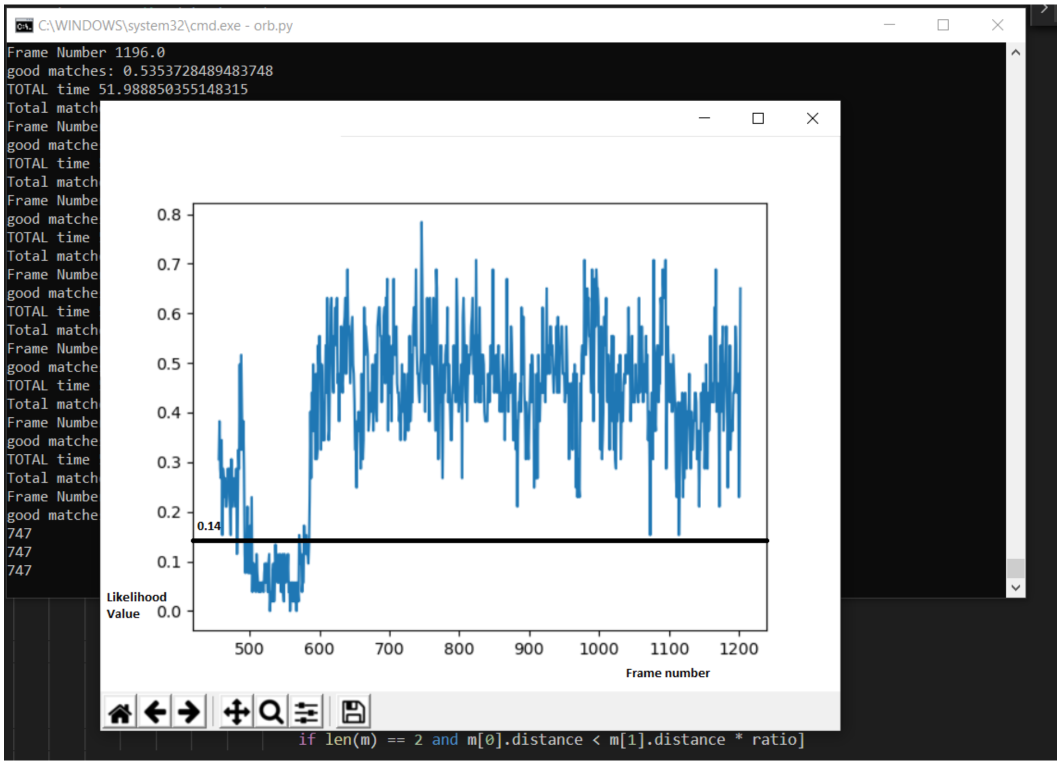


Figure 5.41: The multicolor lines represent the matched keypoints from the template to the target frame. We can observe that a few keypoints matches the template to the target image even when the target image is rotated, increasing the possibility of rotation invariance.

The tool runs ratio filters based on user input (70/30 or 90/10) (discussed in detail in Section 6.2.3) and extracts good matches for each frame among all the detected features (as explained in Section 2.2.2). The good matches are the matches with the nearest Hemming distance value between the template and the target image. The obtained values are normalized between 0 to 1 over all the frames, and those normalized values for each frames are the likelihood of the target value containing the template. While using ORB algorithm with FLANN-based matcher on the soccer video, we obtained likehood values for all 1260 frames that we plotted in the Figure 5.42. Figure 5.42 shows the likelihood value in the y-axis and the respective frame number in the X-axis, higher the likelihood value for a frame, the higher chances of the template being present in the target frame. As shown in the Figure 5.42 for obtained values, if we selected a threshold as 0.14 to separate the likely and unlikely frames, all the frames holding values greater than 0.14 are assumed to be containing the template. To study the accuracy of the results obtained, we compared the



```
Frame Number 1196.0
good matches: 0.5353728489483748
TOTAL time 51.988850355148315
Total match
Frame Numbe
good matche
TOTAL time
Total match
Frame Numbe
good matche
TOTAL time
Total match
Frame Numbe
good matche
TOTAL time
Total match
Frame Numbe
good matche
TOTAL time
Total match
Frame Numbe
good matche
TOTAL time
Total match
Frame Numbe
good matche
747
747
747
```

```
if len(m) == 2 and m[0].distance < m[1].distance * ratio]
```

Figure 5.42: The frame number is represented in the x-axis with its corresponding likelihood value in y axis. We observe and select a threshold of 0.14 to avoid loosing true positives and to avoid matching false positives.

obtained values with the given ground truth values for each frame as we observe in Figure 5.43. The ground truth values are included in the output graph alongside the obtained results. In this figure, the blue lines represents the manually labelled data with the ground truth values for the frames, and the red line represents the results obtained when using ORB algorithm with FLANN-based matching. While comparing the ground truth with obtained results for each frame presumably 0.14 as the threshold (as shown in Figure 5.43 using a black line), we can observe that the system fails to detect only the frames between 370-390. Hence, with the careful threshold selection and separation, it misses around 20 frames but matches the rest of 1220 frames which gives it an accuracy of roughly 96% for a short manually rotated soccer video. It shows the effectiveness of using the ORB algorithm for feature-based template matching using FLANN-based matcher.

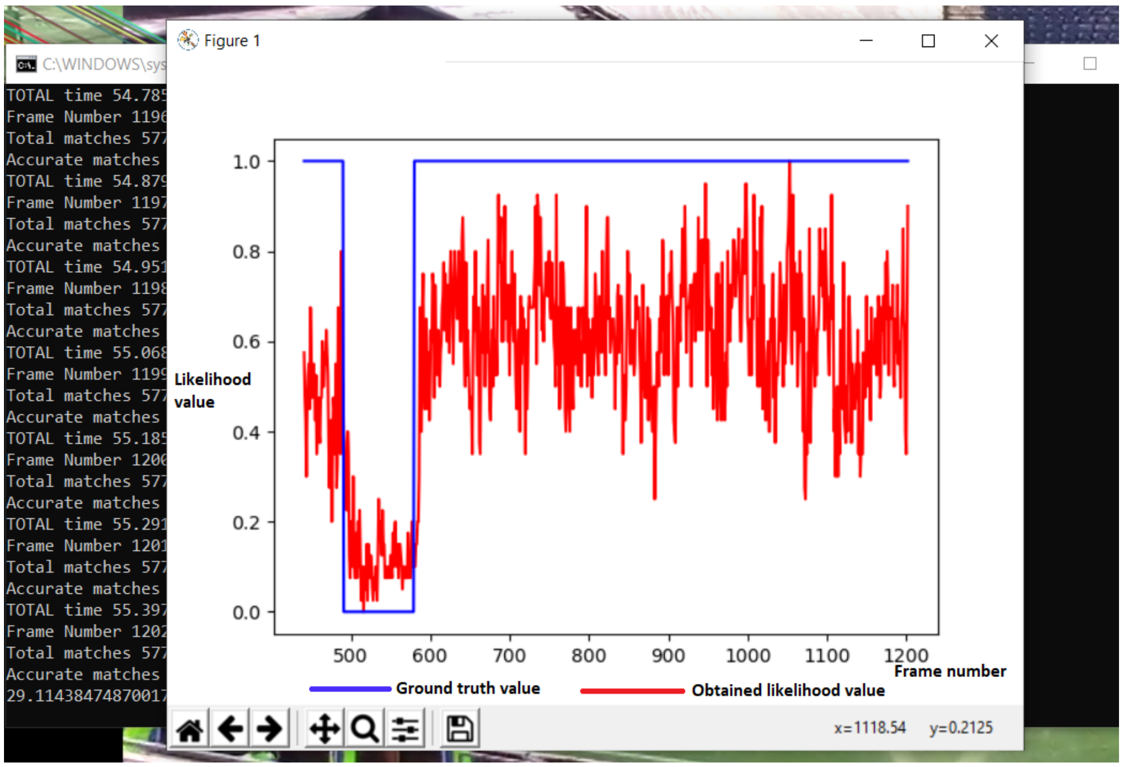

```
C:\WINDOWS\sys
TOTAL time 54.785
Frame Number 1196
Total matches 577
Accurate matches
TOTAL time 54.879
Frame Number 1197
Total matches 577
Accurate matches
TOTAL time 54.951
Frame Number 1198
Total matches 577
Accurate matches
TOTAL time 55.068
Frame Number 1199
Total matches 577
Accurate matches
TOTAL time 55.185
Frame Number 1200
Total matches 577
Accurate matches
TOTAL time 55.291
Frame Number 1201
Total matches 577
Accurate matches
TOTAL time 55.397
Frame Number 1202
Total matches 577
Accurate matches
29.11438474870017
```

Figure 5.43: The red lines in the Y-axis represents the obtained values of likelihood for each frame number in the X-axis. The blue line is the manually plotted ground truth value. We compare and visualize the segregation of the likely and unlikely values as well as the accuracy of the algorithm using FLANN-based matcher.

After confirming its prospects of effective performance on a short video, we executed the same algorithm on a longer video with rotated and scaled scenes. For the next step, to determine the time taken by the system to execute a longer video, we used the original six minutes and 47 seconds long soccer video (with 12,210 frames). This was the same single-camera soccer which we initially used in Section 5.2.1 and it was manually rotated and scaled to study ways overcome the limitations. When we used ORB as the back-end algorithm, the system took 150 seconds to execute and the obtained results were plotted in a graph as shown in the Figure 5.44. When we choose a threshold among the obtained likelihood values, we can observe that the system detects some rotated frames and some of the scaled frames. The likelihood drops where they are scaled and rotated simultaneously like labelled in the Figure 5.44.

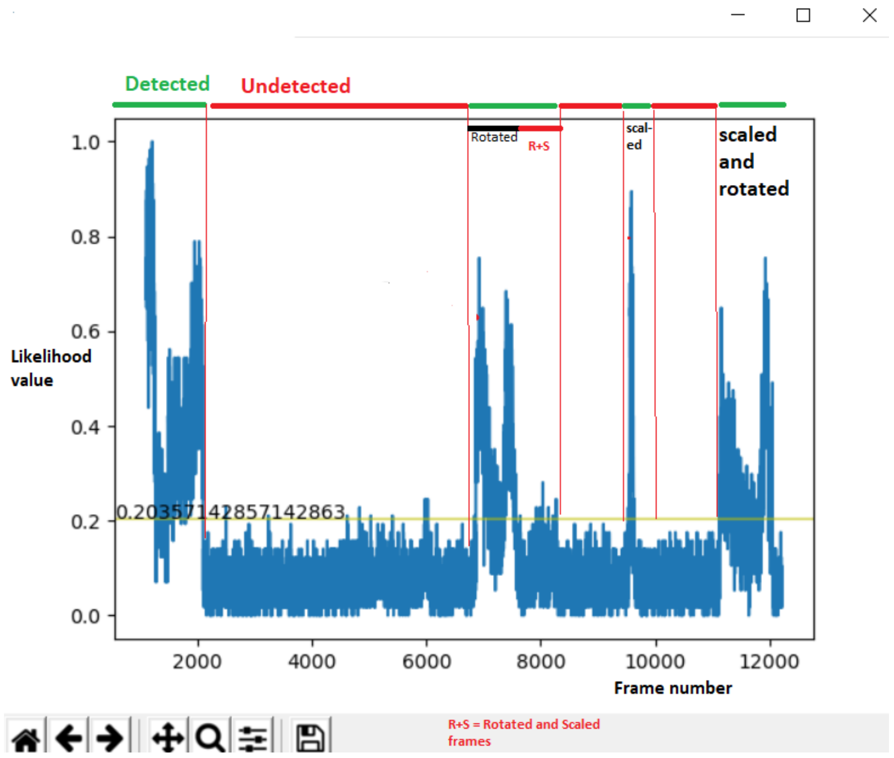


Figure 5.44: The user selected 0.2 as the threshold for the likelihood value, we can observe that the obtained values in blue lines can be compared to the actual values represented by the horizontal sectioning of the frames. The rotated and scaled frames are also labelled in the graph for easy comparison.

From the initial observation, we can assume that ORB may be rotation invariant and partial scale invariant (as explained in Section 2.2.2). The graph in Figure 5.44 is manually sectioned into detected and undetected frames. Detected frames are the frames with higher likelihood of containing the template and the undetected frames are the ones that are unlikely. The red lines parallel to the X-axis indicate the actual sections (i.e. observed from the video) where the frames do not contain a template and the green lines indicate the sections where the frames should contain a template. The horizontal red lines indicate the sudden change in likelihood value which sections the obtained values of the graph, i.e., points where there is either a gradual increase or decrease in the likelihood value. We select a threshold based on these increasing or decreasing trends to observe and separate the detected and undetected frames. For example, from the obtained graph, if we choose 0.2 as the threshold, then the system understands that any template with a frame value is 0.2 or higher contains the template. All the frames higher than the chosen threshold are extracted as frames likely to contain the template.

Finding We can observe that feature based-template matching with ORB using FLANN-based matcher takes double the time of execution when compared to the template based matching. This is because existing template-based matching system skips every four pixels and skips through every 30 frames (as explained in Section 5.2.1) whereas the feature-based template matching scans each and every pixels in each input frame. The run-time may be reduced by further experimenting while skipping frames which is what we experimented in the further experiments (discussed in detail in Chapter 6). Another interesting observation is that ORB matches more of the obtained likelihood values (as shown in Figure 5.44) between the rotated frames 7000-8300, when compared to the scaled frames between 9200-10000 which shows that ORB is likely more rotation invariant than scale-invariant. We also observed that ORB works poorly while comparing the frames 11000-12210, which were both scaled and rotated simultaneously.

SIFT and SURF using FLANN-based approach While using ORB, the feature descriptors are defined using the Euclidean distance formula, while for the SIFT and SURF algorithm descriptors represent the oriented gradient [59]. In this section, we first used SIFT on the short soccer video like we did while experimenting on ORB based on the FLANN approach. When the system matches the keypoints and descriptors using the ratio test, we noted that the system detected more keypoints than ORB as shown in Figure 5.46. The system also detects more frames with the template without losing most of the likely frames. The ground truth values are shown alongside the obtained values, and we can observe the distinction between the likelihood values that determines if the frame contains the template. The red lines are the obtained likelihood which can be compared to the blue ground truth value. This clear distinction allows the user to choose a template without hesitation and with minimum errors. After obtaining optimal results from a short video, we also tested on the longer soccer video we used in the experiments earlier (as we did with ORB). SIFT algorithm takes 390 seconds to execute on the long soccer video, which is almost three times the time ORB takes. This is understandable because more steps included in execution of the algorithm for defining the descriptor and the ratio filter process for the matching (as discussed in Section 2.2.2). When we run SIFT algorithm on the back-end, all the obtained likelihood values are noted and presented in a graph. We

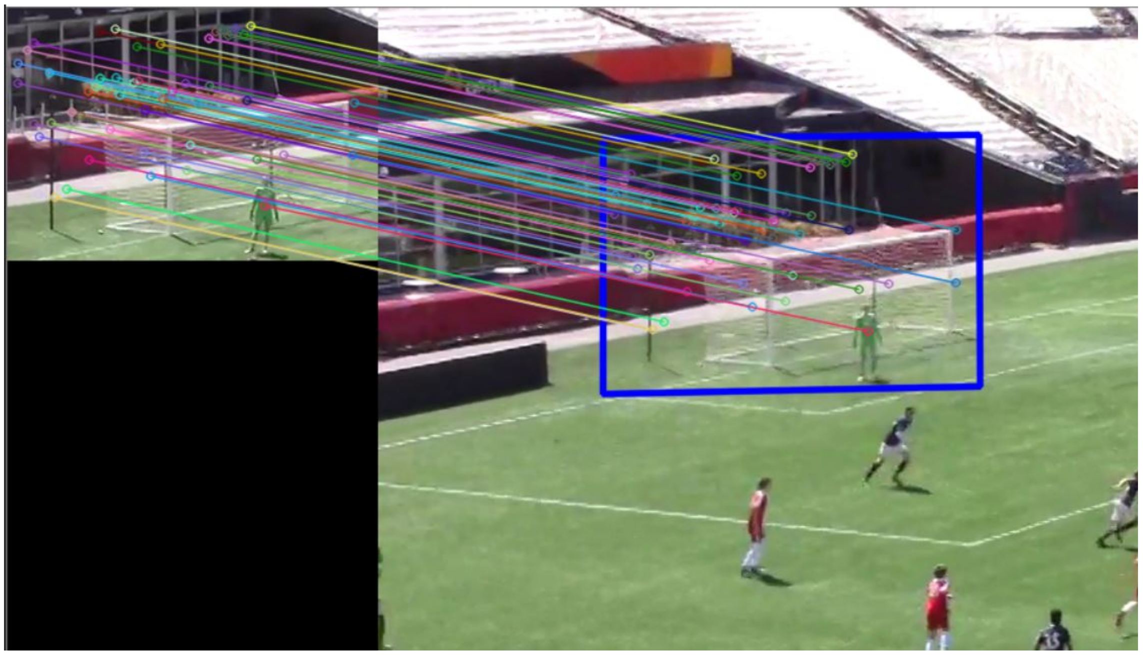


Figure 5.45: The multicolor lines are the indicators of the matched keypoints between the user-selected template on the left and the video frame on the right. We observe that more keypoints are matched when compared to the ORB as shown in Figure 5.41.

observe that most of the rotated frames, and some simultaneously scaled and rotated frames are matched and detected (as shown in Figure 5.48). The observation shows that SIFT algorithm based on FLANN is likely to overcome some of the limitations of template-based template matching and provide us an optimal solution to template matching, especially with the adjustments of certain parameters (as discussed in Section 6.2). We observe similar findings when we use SURF as the back-end algorithm for the long soccer video. We also observed that SURF took 250 seconds, which is higher run-time than ORB for execution but lower run-time than SIFT. This may be because of the speeded up and robust keypoints and descriptor matching SURF offers (as explained in detail in 2.2.2). The obtained likehood values when using SURF are plotted in Figure 5.49. We observe that the obtained values are segregated as likely and unlikely frames based on their likelihood value when we select a distinctive threshold as shown in the figure. From

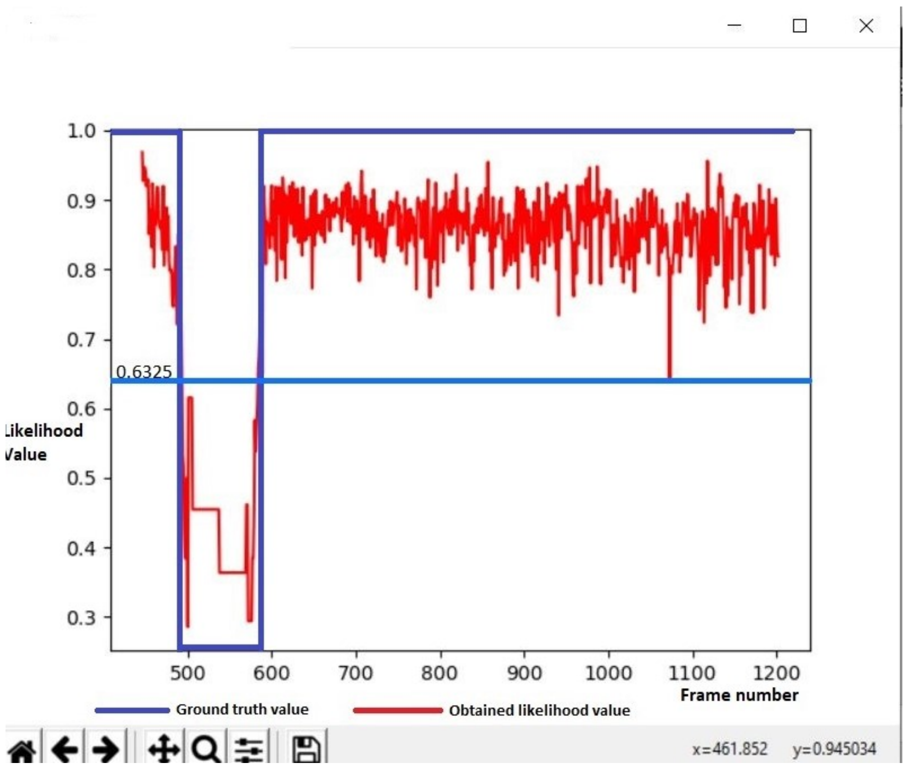


Figure 5.46: Comparison of the likelihood values obtained by running SIFT descriptors test on a 42 seconds soccer video with the ground truth values. The dark blue line indicates the manually entered ground truth values whereas the red lines indicate the obtained likelihood values. The light blue line with a numerical value indicates the user-selected threshold.

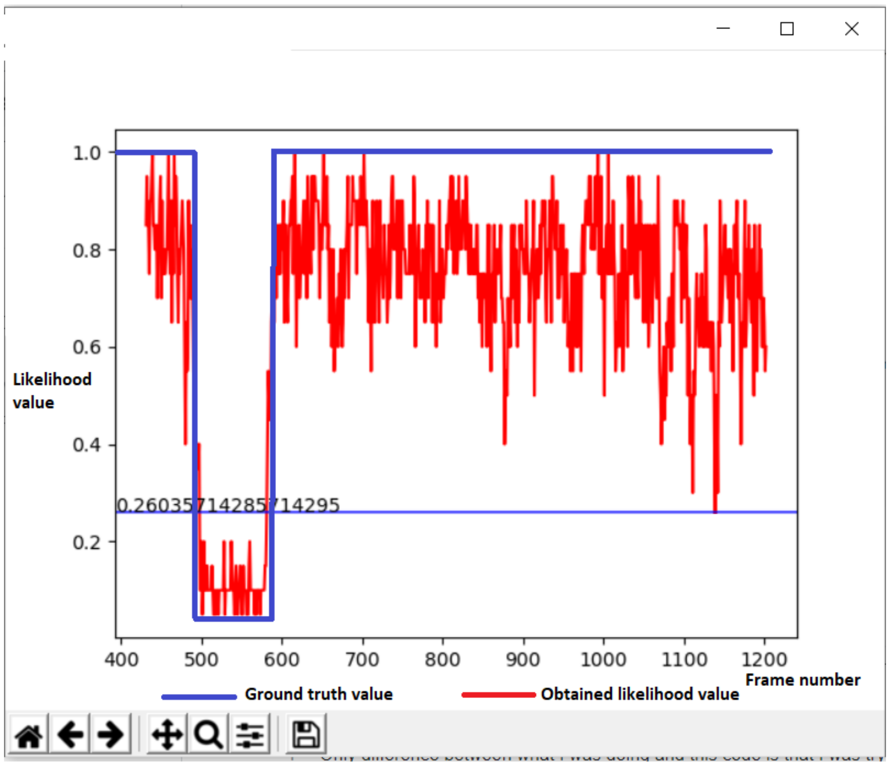


Figure 5.47: Comparison of the likelihood values obtained by running SURF descriptors test on a 42 seconds soccer video with the ground truth values. The dark blue line indicates the manually entered ground truth values whereas the red lines indicate the obtained likelihood values. The light blue line with a numerical value indicates the user-selected threshold.

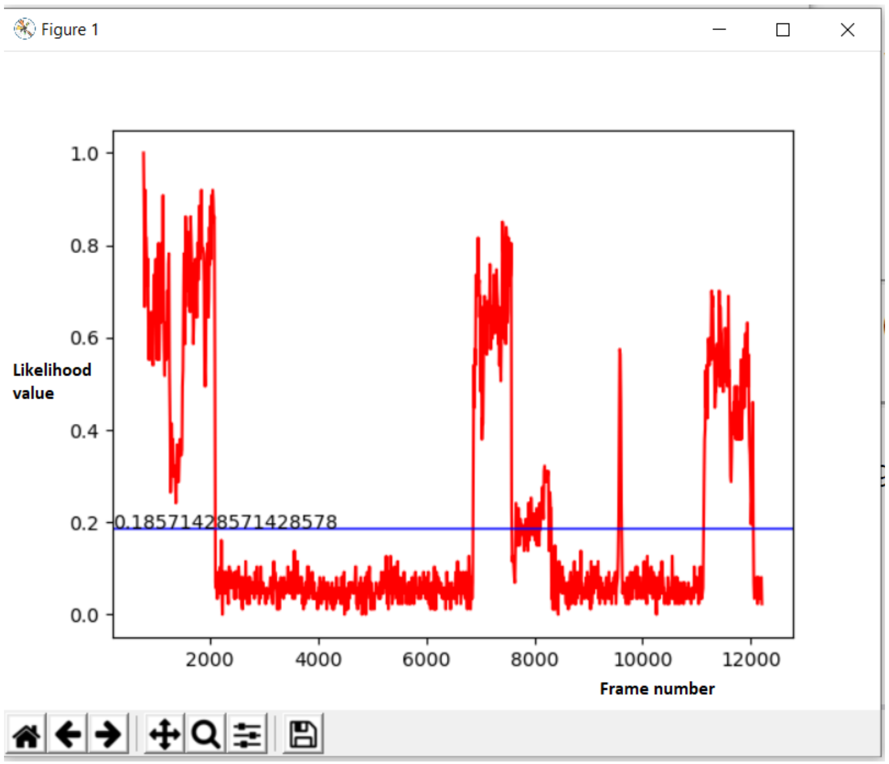


Figure 5.48: Visualization of values obtained by running FLANN based SIFT descriptors test while skipping 10 frames. As shown in the figure, all the frames with likelihood values below the user-selected threshold 0.1857 are unlikely to contain the template.

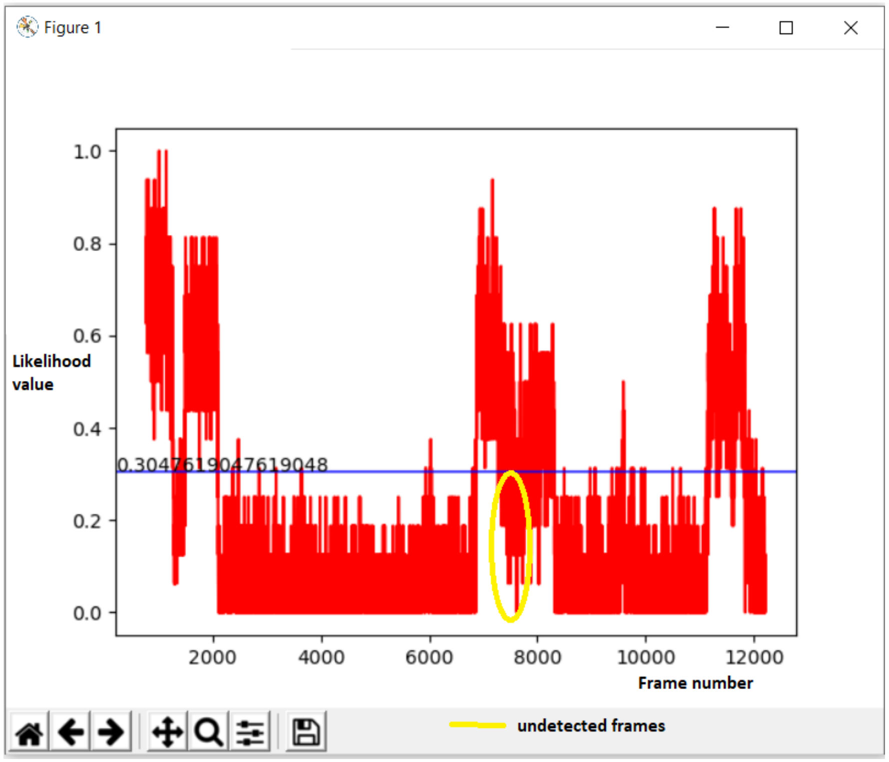


Figure 5.49: Visualization of values obtained by running FLANN based SURF descriptors test on a longer 6 minutes 47 seconds soccer video. As shown in the figure, all the frames with likelihood values below the user-selected threshold 0.3047 are unlikely to contain the template.

the Figures 5.44, 5.48 and 5.49, we can observe that SIFT and SURF seem to detect more likely frames and can be perceived as more rotation and scale-invariant. In addition to this, we observe that SIFT tends to provide distinct values and likelohood segregation despite the higher runtime whereas the fluctuation in the likelihood values in SURF can confuse the user when trying to identify a single threshold value for all the video frames (as shown in Figure 5.49). These are just the preliminary observations from our initial investigation which provided us insight on the feasibility of feature-based template matching algorithms. The obtained results and their optimization will be discussed in detail in Chapter 6. The manually transformed video frames helped us analyze the results clearly and study the outcomes of each algorithm. Based on this preliminary research, we concluded that FLANN-based feature matching is suitable for the proposed tool to overcome some limitations of template-based template matching.

The objective of the preliminary investigation was to study and observe the current issues of the template-based template matching using the three functions (as explained in 3.1.1) and to explore alternate approaches that would potentially provide us with optimal results. The functions of template-based template matching did not completely overcome the limitations (scaling, rotation, illuminance and so on). While exploring alternate approaches, the supervised algorithms based on the limited amounts of training dataset theoretically showed promising results, but it was difficult to build a solid framework working with such limited training data. The supervised algorithms are run by training the available datasets and then testing on the remaining data. It was not possible for the single-camera videos because there were minimum relevant labeled datasets available for training models for the proposed study even after a thorough search on the available resources and over the Internet. Dataset creation, annotation, and labeling is a time-consuming process, and while working with an interactive user interface, it was not easy to incorporate the idea of machine-learning algorithms.

After deciding not to use supervised learning algorithms for this research, we experimented with feature-based template matching which used keypoints as key features in an image and maps the similar keypoints from a template image to a target image (explained in detail in Section 2.2.2). We used Brute Force-based matching and Fast Library for Approximate Nearest Neighbors (FLANN)-based matching approach for feature-based matching, using three algorithms ORB, SIFT and SURF as the back-end video analysis module of the existing system. From the preliminary investigation, we observed the FLANN-based approach to give optimal results. The results obtained are compared against each other and then fine-tuned to provide optimal results in Chapter 6 in detail.

Chapter 6

Results and Interpretation

This research project aims to study and addressing the limitations of template-based template matching. In Chapter 5, we experimented with different approaches to overcome the existing limitations. As discussed, the supervised learning algorithms did not provide satisfactory results that could have been extrapolated to overcome the existing limitations. After analyzing the possibility of using a supervised learning model and finding it difficult to plausibly apply in our research domain, we conducted background research to find alternate approaches. We began experimenting with the feature-based template matching approaches. Initial findings in Section 5.3.2 indicate that feature-based template matching fits our problem domain and scope of work. This chapter investigates, compares, and finally shortlists a robust algorithm based on the Fastest Nearest Neighbour(FLANN) approach among the feature-based template matching approaches.

Based on the observations of the preliminary investigation results as discussed in Chapter 5, we decided to use the FLANN-based approach for the study and compared the results obtained using Oriented FAST and Rotated BRIEF (ORB), Scale Invariant Feature Transform (SIFT), and Speed Up Robust Features (SURF) algorithms, all based on FLANN. The proposed prototype was built using python and we ensured that complete control of template selection was given to the users, just like in the template-based template matching approach. We ran the prototype several times with the soccer video (the same video as the one used in Section 5.3.2) on all the three algorithms. The results are noted and discussed in detail in Section 6.2. The parameters used in these feature-based template matching algorithms (such as the number of keypoints to be detected, ratio-filters to exclude the false matches, and assigning a minimum value to filter the good matches) are adjusted, and the optimal results obtained are compared and discussed in Section 6.3. We also explored the possibilities of using the feature-based template-matching on other sports videos in Section 6.3.3. After discussing the observed results, finally we conducted a user study with our obtained output videos and analyzed the responses in Section 6.4. In this user study, we compared our observations and user responses and discussed the effectiveness of the tool and noted the feedbacks.

6.1 Reiteration of Study Overview

The motivation of this study was to find an algorithm that supports the single-camera video extraction process overcoming the existing limitations. The designated algorithm should also be able to detect the transformed video frames (like rotated and scaled frames). The basic idea of template matching is extracting video frames that have a likelihood of matching the user-selected template (as explained in Section 2.2.1). This research aims to help users distinguish between the likely and unlikely frames that match the template. This research's general idea of interactivity is to plot values for each frame in an output graph and to help users choose a distinct threshold to separate the likely frames from the unlikely ones. The likely frames are then collectively extracted as a short output video containing the template. The template for this research is a goalpost/keeper area and the extracted video is a short highlight video with all frames containing the template.

When the proposed system runs on three different algorithms, ORB, SIFT, and SURF, it calculates three different values for each frame depending on the algorithm used. We can refer to this value in our study as the likelihood value. The obtained values can depend on the parameters that you can tune in the algorithm. In Section 6.2, we discuss how we first executed all three algorithms, with default parameter values, to observe the obtained values. In Section 6.2.1, we discuss how adjusting some values of the dependent parameters for all three algorithms can affect the results. We finalized the best values for the parameters from the observations that optimized the obtained results. We plotted a graph and compared the optimal results of all three algorithms based on the fine-tuned values of the parameters. We then discuss the obtained results of all three algorithms in detail in Section 6.3. We also analyzed the results for all three algorithms when affected by various external factors. For this, we used input videos of other games in different illuminance and visibility conditions to obtain, compare, and analyze the results further as discussed in Section 6.3.3. Finally, we conducted a qualitative user-evaluation study (as discussed in Section 6.4), with the obtained results for all the three algorithms using three different input videos, including the main soccer video (used in Section 3.1.1). We analyzed the responses from user-evaluation study and compared the obtained insights with our experimentally obtained results.

6.2 Adjusting Variables and Function Parameters

We ran our system using all three algorithms, ORB, SIFT, and SURF, one at a time, and we analyzed the obtained results. The system uses various parameters with pre-defined values to run the algorithms. If we change the value of some of the input parameters (parameters with flexible input values that the users can change), the obtained results can be different from those obtained using default values. The obtained results with the default values of the input parameters are discussed in Section 6.2.1. In some cases, changing the values of those parameters may increase or decrease the overall process's efficiency. We tested the algorithms in an ad-hoc manner, by systematically changing the parameter values at arbitrary intervals to observe the differences in results while adjusting the input value of some parameters in Sections 6.2.2, 6.2.3, and 6.2.4. We noted any noticeable differences in the obtained results. We justified the reasons for using specific input values for each parameter, and then discussed all the observations in detail in each of the sections.

6.2.1 Obtained Results with Default Parameter Values

When we run the proposed system based on feature-based template matching, we observed that the results depended on different factors such as data quality, value for the paramaters, and human-computer interaction (the interaction methods, such as mouse selection, keys and so on). In Chapter 5, we found the initial results from the feature-based template matching to be rotation and scale invariant. For the final result calculation we used the longer version of the video (3 minutes 25 seconds of the rotated and transformed soccer video) for detailed observation and to obtain optimal results. To get the output graph that allows users to visually recognize the likely and unlikely frames while reducing the run-time for analysis, we first skipped through every 10 frames of the video. As seen is Figure 6.1, the obtained likelihood value after execution for all three algorithms (ORB, SIFT, and SURF) did not undergo huge performance changes despite iterating through every 10 frames. This assures us that skipping 10 frames decreases our run-time while preserving the accuracy of the results. The obtained results also seemed to overcome some of the limitations while showing promising results.

We ran both the feature-based and the template-based algorithm and discussed the obtained results here. The results are shown in Figure 6.1. The figures can be considered as a comprehensive and perceivable entity to analyze the results for our study as it makes it easier to distinguish the likely and unlikely frames. The X-axis in Figure 6.1 represents the frame number, whereas the y-axis represents the normalized obtained likelihood values for each frame. The obtained values are normalized within a range of 0 to 1, 1 being the highest likelihood that the frame contains the template. The blue line in the graph represents the values obtained using ORB, orange represents using SIFT, and the silver lines are plotted by using SURF as the system algorithm.

When we compare the values obtained by these algorithms, we observe that SIFT and SURF detect more frames when compared to ORB. TO compare the template-based template matching using cross-correlation function with the feature-based template

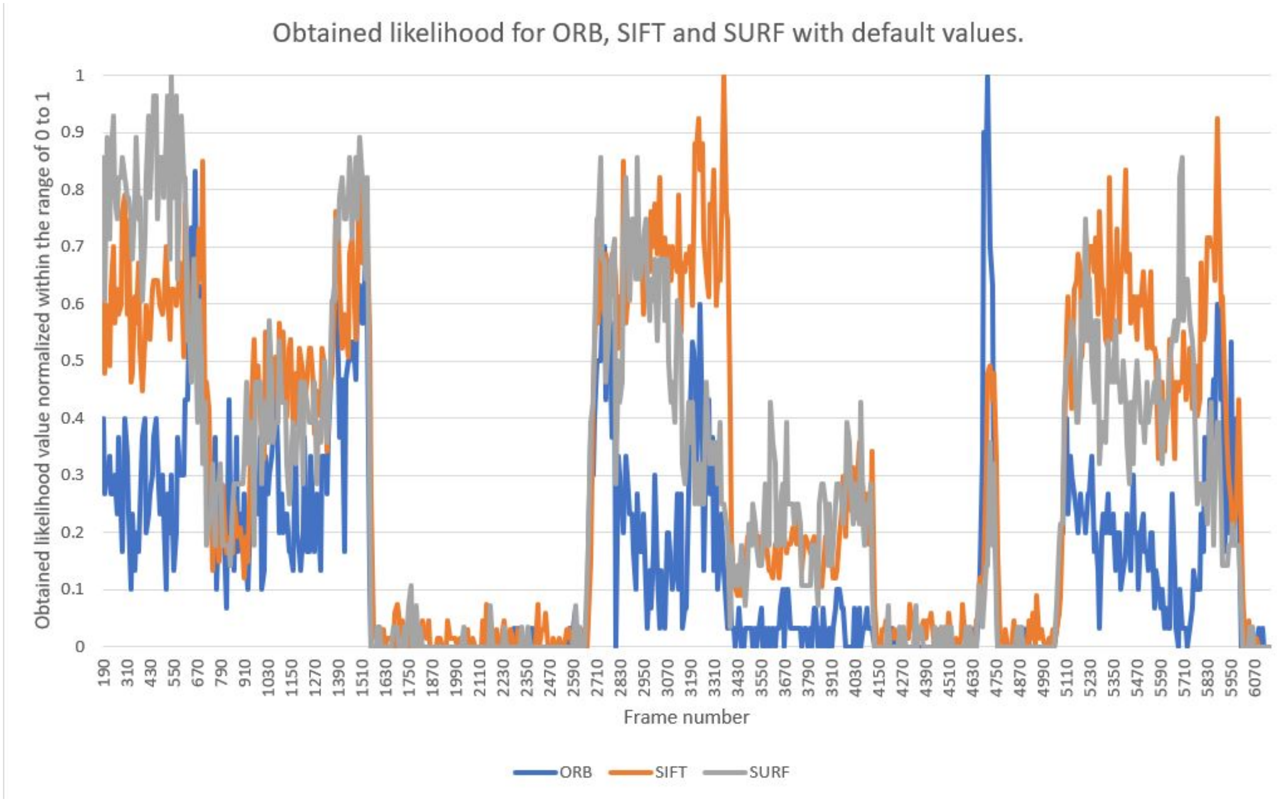


Figure 6.1: Initially obtained likelihood values for the frames based on default parameter values on a 3 minutes 25 seconds long video. The obtained values are for every 30 frames, as frames were skipped for reducing the run-time and to offer a distinctive selection to the users.

matching algorithms as shown in Figure 6.2, we skipped matching for every 30 frames. Despite skipping 30 frames, the accuracy of the feature-based template matching was not affected. It also ensured that the likelihood values obtained by template-based and feature-based matching were now comparable (as template-based matching skipped every 30 frames to calculate the likelihood). The obtained values plotted in Figure 6.2 can now be compared with the template-based template matching. The blue line represents likelihood values obtained using ORB, orange represents SIFT, grey represents SURF, and yellow line represents the likelihood values obtained using existing template-based template matching. All four algorithms are compared with the ground truth values represented by the green line in Figure 6.2. The ground truth value for each frame is assigned manually to either 0 or 1. We label the value 0 for the frames that do not contain the template, 1 for the frames that do. The obtained values for template-based and feature-based template matching are now compared to the ground truth values to determine the efficiency of the system. We start the comparison by first comparing the run-time of the algorithms. We noted the run-time for each algorithm as shown in Table 6.1. Despite the template matching algorithm and its methods taking shorter run-time than feature matching, from Figure 6.4, we observed that feature-based template matching distinctly finds more relevant video frames (including the rotated and scaled frames) than template-based template matching.

From Figure 6.2, when we assume the user selects a distinctive threshold of between 0.4-0.6 (as per our observation, the frames can be separated into the likely an unlikely frames distinctively if we use threshold within 0.4-0.6 range). We can observe that template-based matching shows that the template-based matching detected a fewer video frames of a video

Method	Time Taken (seconds)
Template-based	60.1
ORB	110.22
SIFT	232.27
SURF	175.72

Table 6.1: Table for run-time of each of the algorithms for the 3 minutes 47 seconds soccer video (while skipping 30 frames) in seconds.

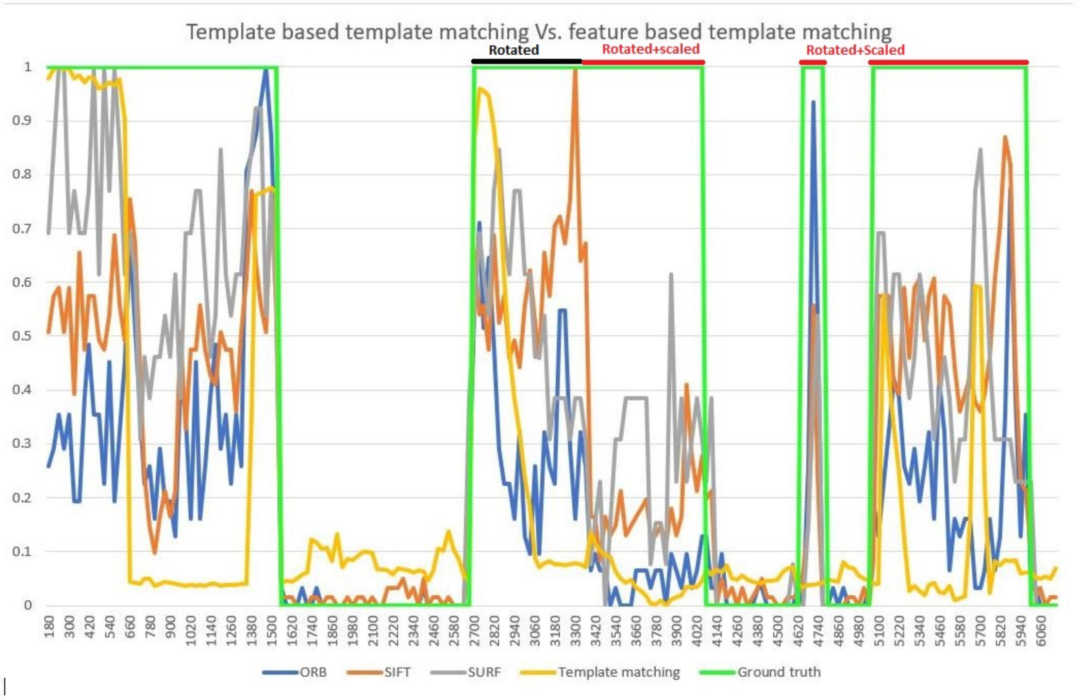
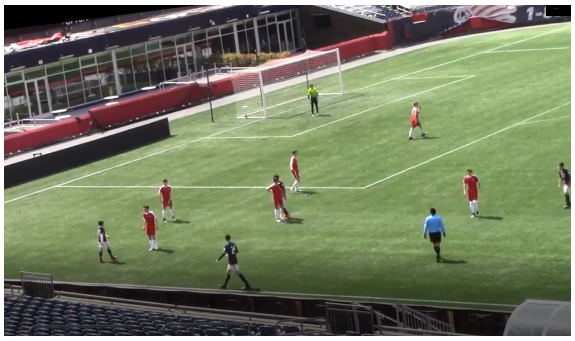


Figure 6.2: Comparision of the existing tool with the proposed feature matching approaches

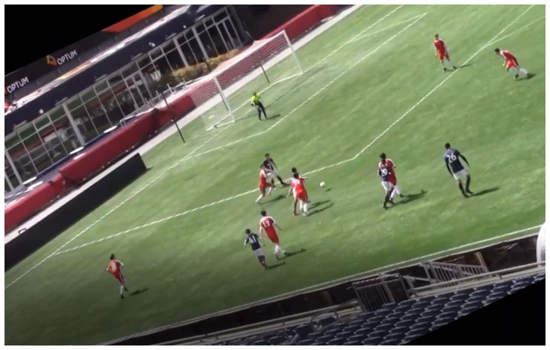
with transformed (rotated and scaled) video frames. Between frames 2640 and 4080, we observe that the template-based correlation method (represented by the yellow line) only has higher values for frames up to 3120, after which the values gradually fall. We went back to the original video to see what was happening and observed that the video starts rotating after frame number 3120, which is why template-based matching fails to detect the template in the following video frames. Similarly, we also observe that the frames between 5030 to 5240 and between 5600 and 5700 have higher values. We investigated and found that these were the frames where the camera rests on the original scale and angle (i.e., when the scale and angle return from one extreme values to the original ones, on the way to the other extreme values.). Frames from 3390-3520 were undetected by any algorithm. We investigated and found the video frames were the ones where the rotation or scaling just begins. This may be because the algorithm failed to detected the first few frames when the frame just starts rotating or scaling being unable to match from a specific angle. Again, the frames 3520-4080 were matched and detected by SURF algorithm only. This may be because SURF uses wavelet responses in both vertical and horizontal distances for orientation calculation by applying Gaussian weight making the orientation calculation and roated frames matching quick and precise.

Figure 6.3 shows samples of scaled and rotated frames for the single-camera video we used for this research. For each of the similar frames, obtained likelihood value using all the primary algorithms for this research is compared with the ground truth value in Figure 6.2. We can observe the difference in the matched likelihood value is higher between using the feature-based matching and the template-based matching when the scaled and rotated

(a) Frame 3210 was manually rotated for this research.



(b) Frame 4690 was manually scaled for this research.



(c) Frame 5260 was manually rotated and scaled simultaneously.

Figure 6.3: Representation of the soccer video frames that were manually rotated and scaled for the research purpose.

frames are used.

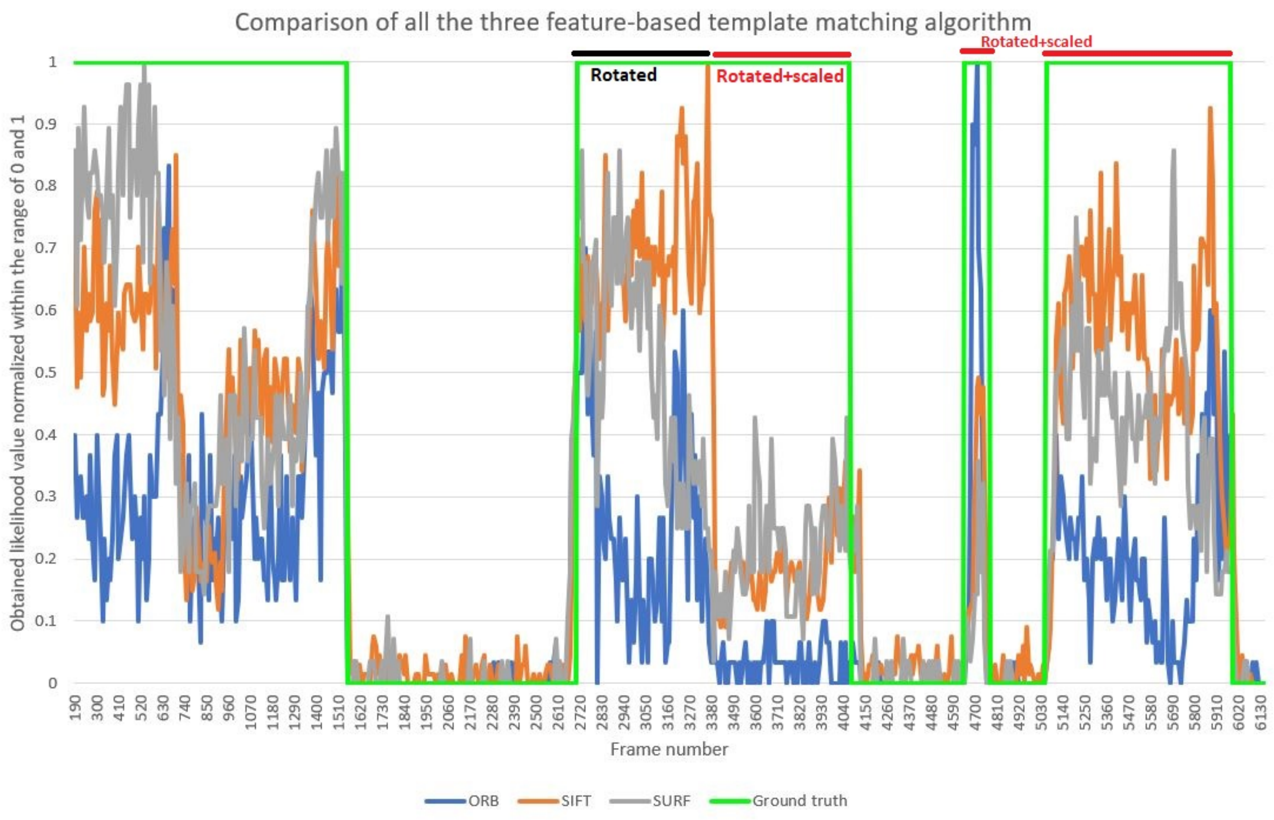


Figure 6.4: Comparison of the likelihood values obtained by the system with the proposed feature-based template matching algorithm.

We can observe the initial results obtained by using all the three algorithms based on the default values, and compare them to the ground truth as shown in Figure 6.4. We observed that, when the threshold value lies between 0.2-0.3, the system can possibly extract more frames with the higher likelihood value. We can also observe that ORB has the many outliers that have caused the minimum true positives to be detected, and SURF has the maximum number of true positives detected. We also observe that despite being scale-invariant, SIFT fails to detect some of the rotated and scaled frames between 3370 to 3970. We investigated and found that those were the frames that were scaled and rotated simultaneously. SURF allows the parallel execution for scaling approaches, whereas SIFT executes scale-space first and then calculates the keypoints orientation. That may be why SIFT failed to detect some of those frames. From our initial investigation, we concluded that SURF worked the best for our soccer video.

We observed that the critical keypoints (i.e., the values around whatever threshold the user selects) can vary when different algorithms are used. To observe the reason behind these differences, we changed the values of the input parameters for feature-based template matching in the proposed tool. We first modified some input parameter values and discussed the obtained results starting in Section 6.2.2 up until Section 6.2.4. We then finalized an optimal value for each of the parameters we used to experiment to achieve effective performance in our obtained likelihood values. We then compared the obtained results with the preliminary results (with default values from Section 6.2.1) and then provided the detailed discussions in Section 6.3.

To optimize the performance of the proposed feature-based matching, we investigated ways to determine appropriate values for a set of parameters. Fine-tuning is the process of adjusting the parameter values for any model or system in ways to improve the performance of the system observed in the preliminary results so that the model works at its best. We fine tune the parameters also to avoid using inconsistent values for the input parameters that may affect the effectiveness of the system. For this system, we changed the values of some parameters and studied their respective roles to adjust the input parameters. The overall process is shown in Figure 6.5 and explained in detail below. Three major input parameters defined for this existing feature-based template matching research are:

- The maximum number of keypoints detected (discussed in Section 6.2.2)

- Selection ratio and good matches (discussed in Section 6.2.3)
- The value for minimum matches (discussed in detail in Section 6.2.4)

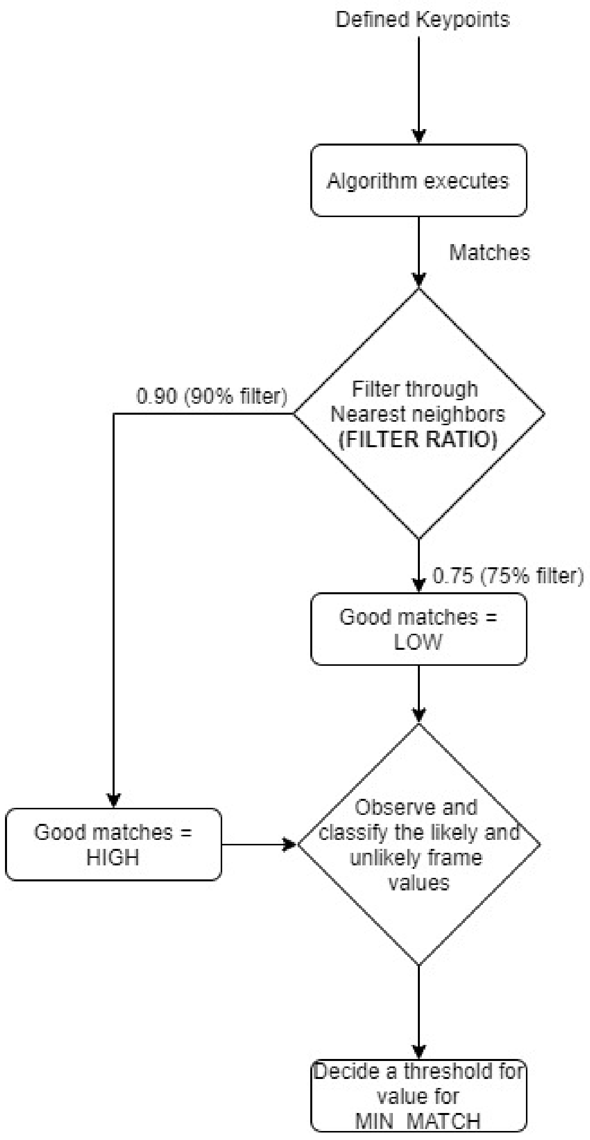

Defined Keypoints
Algorithm executes
Matches
Filter through Nearest neighbors **(FILTER RATIO)**
0.90 (90% filter)
0.75 (75% filter)
Good matches = LOW
Good matches = HIGH
Observe and classify the likely and unlikely frame values
Decide a threshold for value for MIN_MATCH

Figure 6.5: Diagram showing the working mechanism of the prototype tool. The first parameter is the number of maximum keypoints to be detected which determines the value for good matches and the minimum matches eventually leading to a fine tuned system. As seen in the yes/no filter (in diamond figure), the parameters are interdependent and change in value of one parameter will change the value of the overall system.

After the user selects the template (as explained in Section 2.2.1), the first video frame is target image for the tool. For the number of frames N in the input video, the template image and the first target image are the first template-target pair and are the initial input for the algorithm. The system loops through all the remaining target frames skipping every 30 frames, comparing the target frame with the template image. It then calculates the likelihood value for each pair in the order template-target(1), template-target(2),......, template-target(N). For comparing each template-target pair, the system first detects several keypoints for each pair using feature-based template matching algorithms (as explained in Section 2.2.2). In the Figure 6.5, we can observe that the first parameter to be fine-tuned is the number of keypoints to be detected. It is discussed in detail in Section 6.2.2. The system then uses the ratio filter to filter only the good matches among the matched keypoints for that pair. The selection of the appropriate ratio filter is thus the second parameter to be fine-tuned and is discussed in detail in Section 6.2.3.

Based on the number of obtained good matches, we need to decide the value of minimum matches. The number of minimum matches acts as a threshold that determines if, for that template-target pair, the template gets detected in the target frame and is discussed in detail in Section 6.2.4. If the pair's good matches value is greater than defined minimum-matches, the template is present in the target image. For example, suppose a template-target pair has 11 good matches, and the defined minimum match value is 10. Since good matches are greater than defined minimum matches ($11 \geq 10$), the system declares that the same target frame contains the template. The value for minimum matches is the third parameter to fine-tune in this system and is dependent on the values of the first two fine-tuning parameters as shown in Figure 6.5. We will now discuss the process of adjusting and observing the parameters and their resultant values in detail.

6.2.2 Value for the Maximum Number of Keypoints Detected

Feature from accelerated segment test (FAST) is used for keypoint localization in ORB, Difference of Gaussians (DoG) in SIFT, and Hessian matrix in SURF (as discussed in detail in Section 2.2.2). Keypoints are the features highlighting light or dark pixel patches of any object that determines its important structure, like edges and shapes. The intensity of each pixel in an image is compared to all its surrounding neighboring pixels. The comparison also occurs in different scales and dimensions while locating potential keypoints for an image. For different algorithms, it is important to understand how the number of keypoints detected affects the overall performance of the feature-based template matching. The number of input keypoints determines the number of features to be matched for locating an object in an image. The value for keypoints is set at 500 by default but can be modified while defining the number of regions to be detected. We define the number of keypoints in our tool as follows:

$$detector = cv2.ORBcreate(numberofkeypoints) or detector = cv2.ORBcreate()$$

Here, thenumberofkeypoints is set 500 by default. As we increase the value for the number of keypoints to be detected, it can generally increase the precision of the keypoint localization and detection. However, it can slow down the overall process by increasing the run-time for the algorithm. To find the average number of keypoints that may drive the system towards optimal results, we start executing the process with the default value of 500. We note the number of detected keypoints and execute the computation again by increasing the value for keypoints to 2500 and get the keypoints detected are shown in Figure 6.6. We observed that the increase in the number of keypoints also increased the number of good matches for the given video frames. We continually increased the value to

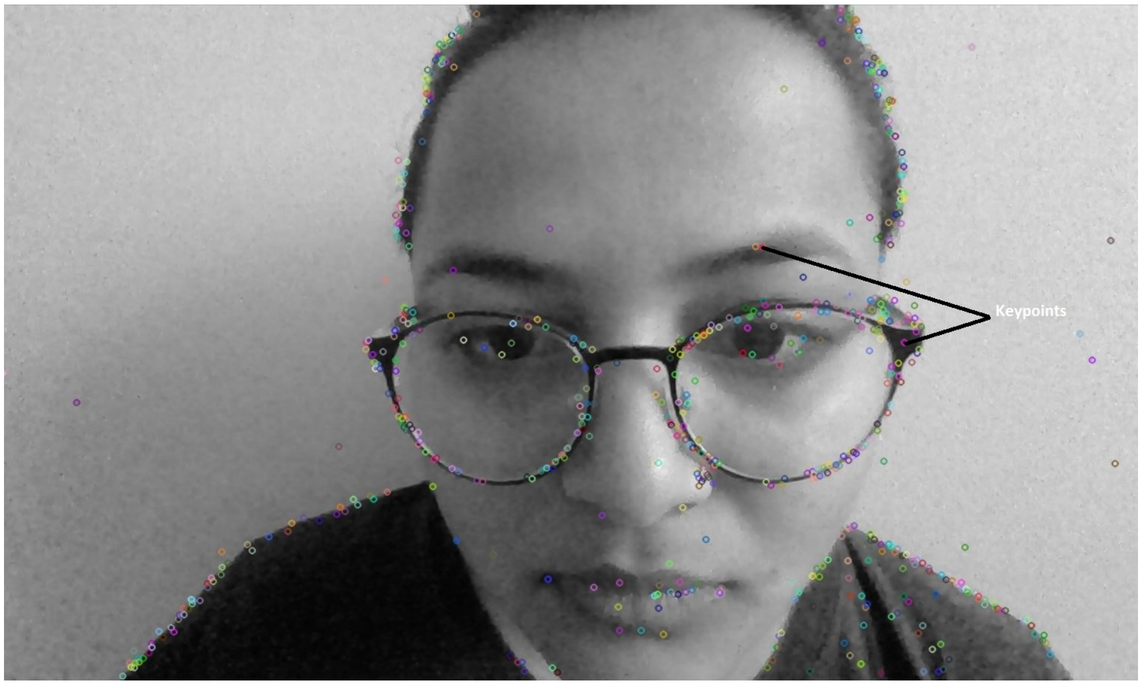


Figure 6.6: Detected keypoints when the value for number of keypoints is 2500.

5000, 7500, 10000, and 12,500. We maintain an interval of 2500, executed the process and noted the results along with the observed good matches. We observed the number of good matches obtained to determine the best input value for several keypoints. For values 500 and 2500, we observe that there are not enough good matches to match the template with the target video frame. While using 5000 or larger value, we observe minimum increase in run-time but enough good matches as shown in Table 6.2. From Figure 6.7, we plot a graph and use the elbow plot method to determine the best value to use for the parameter. To maintain a balance between the detected good matches and run-time, we decided to take an average of 5000 features after analyzing the Figure 6.7.

Keypoints	Good matches (ORB)	Good matches (SIFT)	Good matches (SURF)
500	11	45	24
2500	28	79	41
5000	39	95	52
7500	43	102	55
10000	46	107	61

Table 6.2: Number of input keypoints with the number of good matches detected.

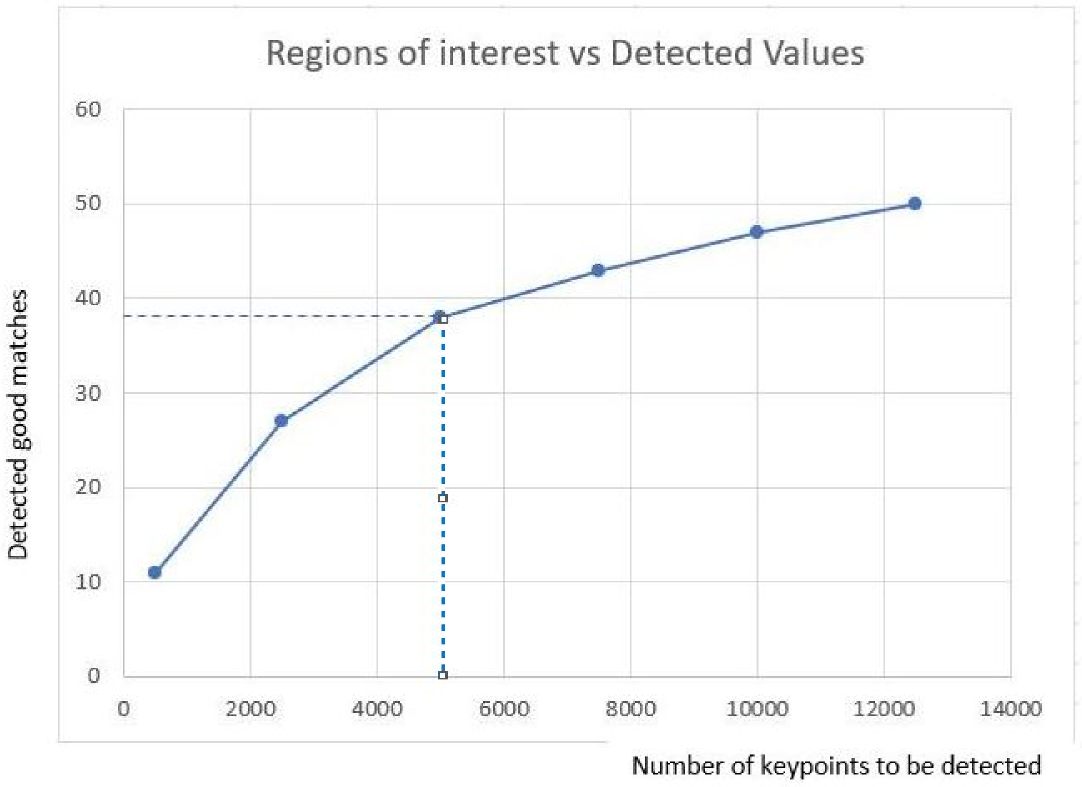


Figure 6.7: Graph plot for different values for number of keypoints and the dotted line shows the 5000 is the optimum value for the of number of keypoints.

6.2.3 Good Matches and Selection Ratio

As explained in the previous section, we now initialize the value for the number of keypoints to be detected in the image at 5000. After the system has detected the keypoints, the system uses the detect and compute functions to locate the keypoints and define their respective descriptors.

$$kp1, desc1 = detector.detectandcompute(gray1, None) \text{ for template}$$

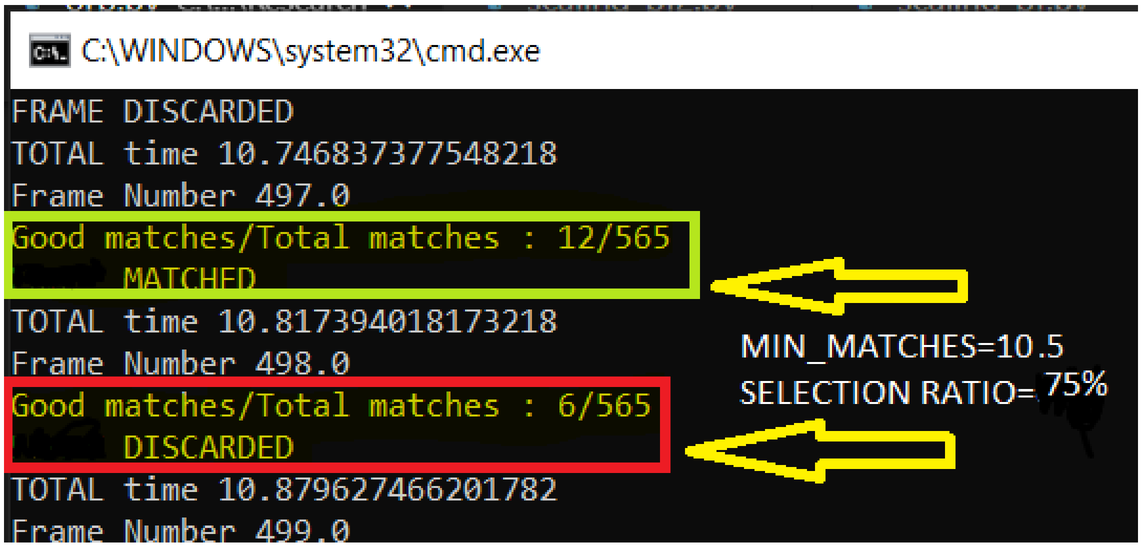
$$kp2, desc2 = detector.detectandcompute(gray2, None) \text{ for target frame}$$

where kp1, desc1 are keypoints and their respective descriptors of the template and kp2, desc2 are the keypoints and their respective descriptors for the target image frame. These descriptors are the feature vectors carrying 0 or 1 values as a range of 128-512 bits string. For example, the descriptor denotes whether the keypoint represents the edge or a dark and lightly shaded area in the image. After a value is assigned to the keypoint, the algorithm finds the good matches by calculating the Euclidean distance between the template and the target image keypoints for the template-target pair. Good matches are shortlisted from the initial pool of matched keypoints. For example, if 500 keypoints are detected in the template and 510 keypoints are detected in the target frame, we can assume around 200 keypoints will be matched among the 5000 keypoints for the template-target pair. Then we apply a ratio filter to these 200 matched keypoints based on the nearest neighbor algorithm to extract a fewer optimal good matches.

Mapping all matched keypoints makes the program prone to potentially extracting inaccurate or anomalous data. Among those 200 initial matches, we filter out a fewer good matches by segregating the matched keypoints based on the shortest distance between the keypoints. First, the system determines the keypoint with the shortest distance and then determines the second keypoint with the second shortest distance. It then compares the distance between these two neighboring keypoints to determine if they are the nearest neighbors. Good matches are determined by how close the minimum matched keypoints and the second best-matched keypoints are. The filter ratio threshold is a numeric value that segregates the obtained matches into good matches and bad matches. The filter ratio threshold is a value between 0 and 1 (0 to 100%) and is defined after the keypoints are detected and their descriptors initialized. The matched keypoint is called a good match if it has the shortest distance with the neighboring matched keypoint. The matched keypoints with higher distance between the neighboring matched keypoints are the bad matches and they are discarded.

For example, if the filter percentile is 0.75, the system defines the keypoints with the distance value three-quarters (75%) time closer to another keypoint as a possible good match. Thus, the system finally generates the 50 good matches, where the distance between the neighboring keypoints had a maximum distance ratio of .75. This means each of the keypoints that was closer than or at most three-quarters closer to the neighbouring keypoint were considered as good matches. For each target frame in the video, FLANN generates its number of good matches obtained when the template is compared with each target image. For example, if there are 1000 frames in a video, each frame will have a different number of good matches, but it will be under a similar range as shown in Figure 6.8. Extracting a number of good matches optimizes our resultant likelihood value by neglecting the keypoints that had greater distance between them. Determining the number of good matches takes us one step closer to decide if that target frame contains the template.

To obtain robust resultant data for this research, we tested the prototype tool with two values for the filter ratio, 75% (0.75) and 90% (0.90). We used 0.75 because there were only few to no values for good matches when we tested with 0.50 (50%) and the filter ratio as the distance value would be very small between the keypoints. If the threshold for the ratio filter decreases, so does the maximum distance between the matched neighbors lowering the value of good matches. After passing the obtained matches through these ratio filters, we get the final number of good matches for that target frame. For this, we use the FLANN approach where the initially matched keypoints loop through the array where each keypoint compares its distance value with its neighboring keypoints using the ratio filter. For example, if we filter the obtained matches through 0.75 (75%), we can achieve a short array of compatible and good matches as most of the matched keypoints. The keypoints with longer distances between them get eliminated, so we can have a small optimal set of good matches. We can observe the total number of detected good matches in Figure 6.8 when the ratio was 0.75 (75%). Similarly, while the same algorithm on the same input video by defining the value for the selection ratio as 0.90, a higher number of good matches (i.e. 116 good matches out of 504 matches) is observed in Figure 6.9. It happens due to lower compactness of the filter. The compactness loosens with the increase in filter value as the matches with distance values as low as one-ninth times closer than the nearest keypoint can be passed and fitted into the good matches array. As a result even the keypoints with a greater distance (up to 0.90 times) between the neighbours are considered as good

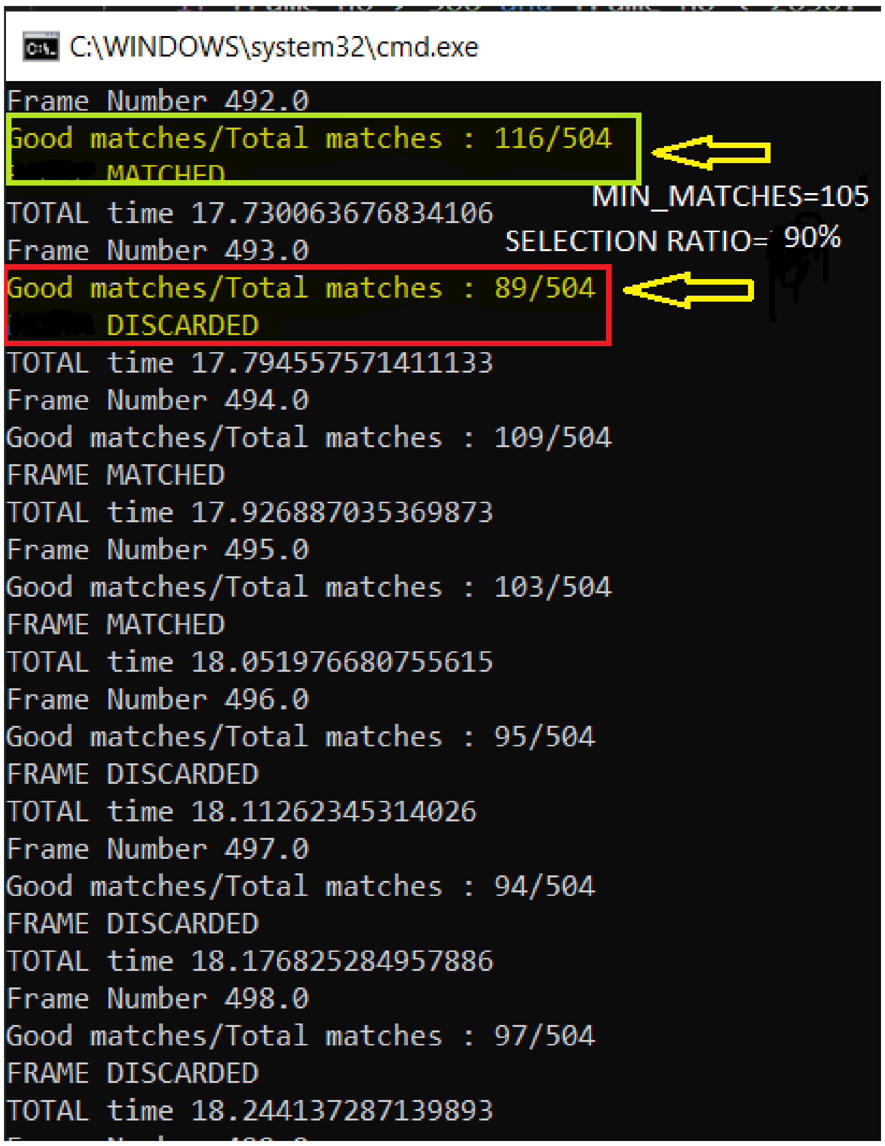


```
FRAME DISCARDED
TOTAL time 10.746837377548218
Frame Number 497.0
Good matches/Total matches : 12/565
[redacted] MATCHED
TOTAL time 10.817394018173218
Frame Number 498.0
Good matches/Total matches : 6/565
[redacted] DISCARDED
TOTAL time 10.879627466201782
Frame Number 499.0
```

Figure 6.8: The highlighted values for good matches while the ratio is 0.75. We can observe that the highlighted values in the green box shows the number of good matches out of the total matches i.e., for target frame number 497, out of 565 detected keypoints, only 12 were good matches.

matches when compared to the shorter (up to 0.70) distances when a 75% filter was used. The higher ratio filter widens the filter letting in more good-matches. The number of good matches detected when passing the value 0.90 (90%) as selection filter for this research is shown in the Figure 6.9.

Ideally, having a higher number of good matches should increase the likelihood of the target frame containing the template but in this case it can lead to lowered accuracy. Increasing the number of good matches may also increase the chances of detecting a false positive because the filter is loosely fitted, allowing false positive frames (keypoints with higher neighboring distance or no matched keypoints) to filter through. For further observation, we plot the obtained results when using the 0.75 (75%) ratio filter in Figure 6.10. The results get narrowed down as the good matches by discarding keypoints with distances larger than 75%. It discards the disputed values and their keypoints to shorten



C:\WINDOWS\system32\cmd.exe

```
Frame Number 492.0
Good matches/Total matches : 116/504
[redacted] MATCHED
TOTAL time 17.730063676834106
Frame Number 493.0
Good matches/Total matches : 89/504
[redacted] DISCARDED
TOTAL time 17.794557571411133
Frame Number 494.0
Good matches/Total matches : 109/504
FRAME MATCHED
TOTAL time 17.926887035369873
Frame Number 495.0
Good matches/Total matches : 103/504
FRAME MATCHED
TOTAL time 18.051976680755615
Frame Number 496.0
Good matches/Total matches : 95/504
FRAME DISCARDED
TOTAL time 18.11262345314026
Frame Number 497.0
Good matches/Total matches : 94/504
FRAME DISCARDED
TOTAL time 18.176825284957886
Frame Number 498.0
Good matches/Total matches : 97/504
FRAME DISCARDED
TOTAL time 18.244137287139893
```

MIN_MATCHES=105

SELECTION RATIO= 90%

Figure 6.9: The highlighted values for good matches while the ratio is 0.90. We can observe that the highlighted values in the green box shows the number of good matches out of the total matches i.e., for target frame number 492, out of 565 detected keypoints, 116 were good matches. Due to the higher value for ratio filter, keypoints with higher distance values between their neighbours are also extracted as good matches.

the potential matches list. It is the first step to determine if the template matches the target. We can observe a clear distinction between the likely and unlikely frames while using the 75% filter ratio as shown in Figure 6.10. In contrast, despite obtaining more good

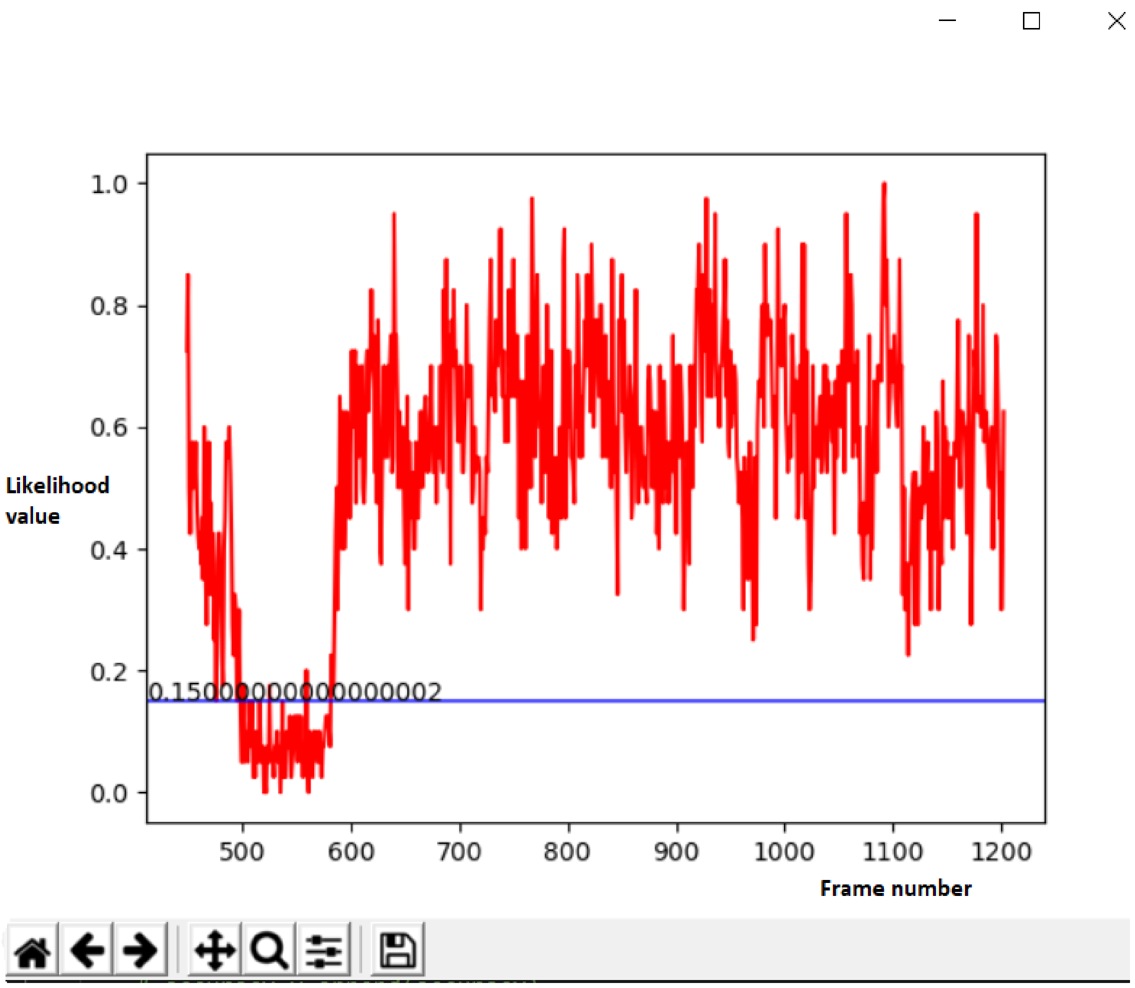


Figure 6.10: Graph plot for likelihood values obtained using a 0.75 ratio filter. We can observe that the likelihood values are well distributed for the user to easily choose a threshold. As seen in the figure, the user selects 0.15 as the threshold that distinctly segregates the likelihood values predicting likely and unlikely frames.

matches when using 0.90 as a filter, the system matches the keypoints with values holding longer distances between them that may be extracted as false-positive keypoints. Matching excessive keypoints also increases the chances of error and may cause difficulty while choosing a threshold, as shown in Figure 6.11. While comparing the ratio filter that compares the neighboring keypoints distance, the 0.75 ratio gives distinctive segregation of likely and unlikely values minimizing the false positive when compared to 0.90. Using a narrow filter allows keypoints with higher likelihood to filter through when compared to using a wider filter. It can increase inconsistency in the matched values. We observed and decided to use 0.75 as the ratio filter for this system. If we use a smaller value than 0.75, say 0.50, it filters only 0-2 good matches which is an inadequate metric value for the system to analyze the likelihood of the target frames. The number of good matches obtained helps determine the value of the next parameter, minimum matches threshold which is discussed in detail in the next section.

6.2.4 Value for Minimum Matches

Minimum-match is a numeric value that acts as a threshold to let the program decide if the target contains the template or not. In the feature-based template matching algorithms, the term MIN_MATCH defines the number of minimum matches, and the user can determine the minimum matches value observing the range of values for the good matches. It is the minimum number of good matches a template-target pair should have for the detected keypoints to match. The minimum matches value determines if the template-target pair matches the keypoints by comparing the obtained good matches values to the given value of minimum matches. From examples in Section 6.2.2 and 6.2.3 and Figure 6.8, we observed that the number of good matches was between 0-20 (Let this value be N) while using the 0.75 ratio filter so we defined the value of MIN_MATCH as the median calculated as follows:

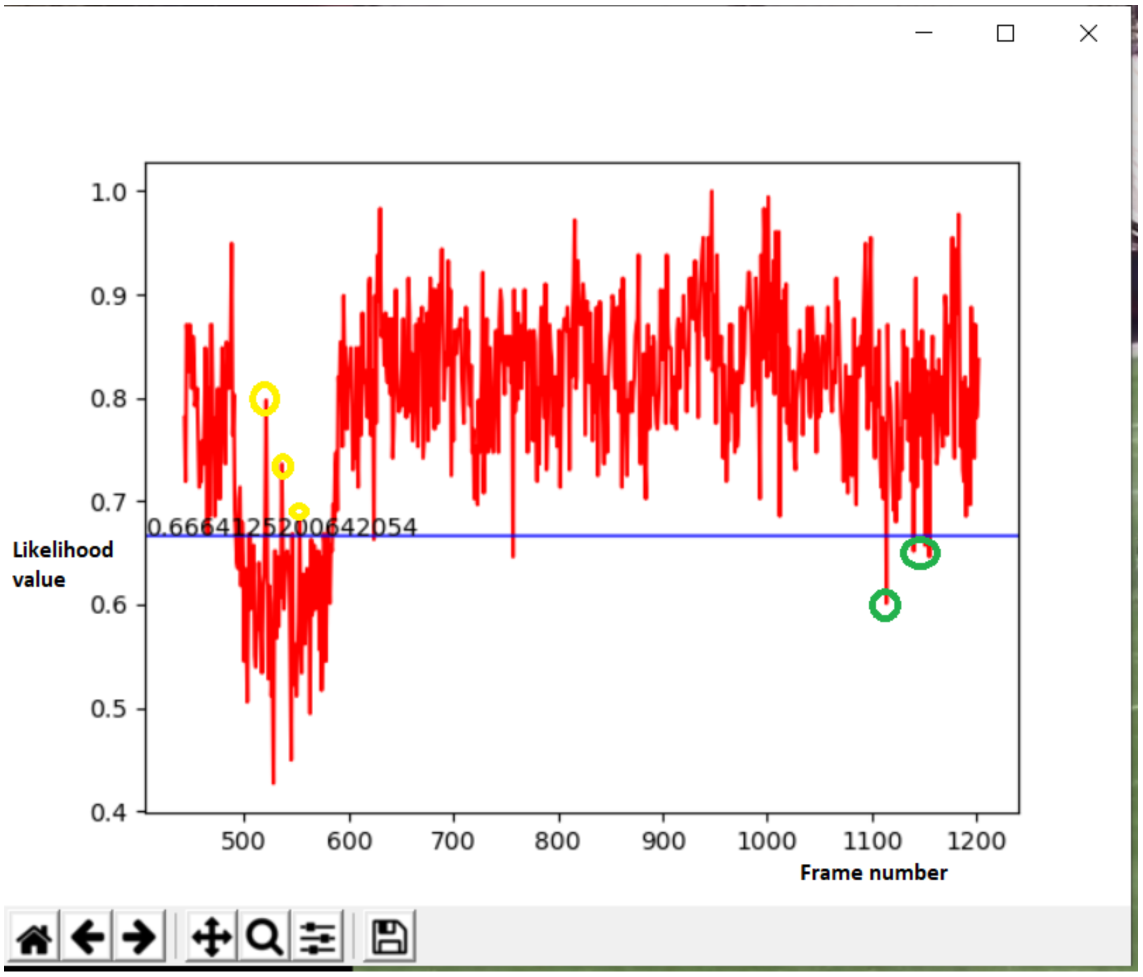


Figure 6.11: Graph plot for the likelihood values obtained using a 0.90 ratio filter. We can observe that the obtained values are not distinctly segregated. If the user selects a threshold of 0.66 as shown in the figure, some false positives (circled in yellow) are extracted as likely frames that are considered as the false positives. The green circles on the other hand are the false negatives (false negatives here are the likely frames that are discarded assuming they don't contain the template).

observed min (N)-observed max (N) i.e., 10.5 in this case (as shown in Figure 6.8)

Similarly, when we used a 0.90 (90%) ratio filter, the number of good matches was between 0-200, so we defined the value of MIN_MATCH to be 100 (as shown in Figure 6.9).

If a template-target pair contains fewer good matches than the defined minimum, the keypoints are discarded (or the keypoints do not match) by giving a lower likelihood value. The system labels such target frame as 0 (template absent). Similarly, if a template-target pair contains more good matches than the defined minimum matches, the system marks the target frame as 1 (template present). For example, in Figure 6.8, we observe that the ratio filter used is 0.75, and the obtained good matches range between 6 to 13. Therefore, we can finalize the input value of the minimum matches as the median value of 10.5, i.e., MIN_ MATCH = 10.5. This value acts as a threshold. Like in the example from Figure 6.10, the system calculates good matches for each template-target pair so the system detects the template in the target frame when the good matches value is 12 and discards the frame when the good matches value is 6. We can also observe a similar example in Figure 6.9. Hence, a minimum match is a pre-defined number based on the number of good matches detected.

For example, if we use any of the ORB, SIFT, or SURF algorithms to pass the matches through a 0.75 filter and observe the value of the good matches, we set the value for minimum matches to the median (i.e., 10.5). We choose 10.5 as the median value that possibly separates the likely and unlikely target frames to contain a template (as shown in the Figure 6.8). Using the appropriate value for the minimum matches prevents data loss and discards the undetected target frames with projected confidence. Contrary to this, if the minimum-matches value is not carefully decided, e.g., if we randomly choose the value to be 13 in the above example (for the Figure 6.8), we will discard almost all of the likely frames containing the template, and it will severely impact the results. But if we observe that the likely target frames are between 15 and 25, the unlikely target frames are between 7-18. In such a case, we can decide to select the median of range 7-25, i.e., 16 as the threshold. Minimum matches value acts as a threshold to avoid false positives and to preserve the likely frames. In the proposed prototype, we analyzed all our observations and defined the values for the discussed parameters. The input values for the parameters also depend on the algorithm used to run the system.

For the SIFT and SURF algorithm, the system gives the value for minimum matches according to the number of detected keypoints. We can also use mean of the observed obtained value of good matches instead of median to determine the minimum-matches value input. When we used the manually scaled and rotated soccer video, we observed that the number of good matches for each frame increased with the use of the SIFT and SURF algorithm as the algorithms are more rotation and scale-invariant. The input for all the three parameters discussed in Section 6.2.2, 6.2.3 and 6.2.4 should be analyzed and fine-tuned based on the algorithm used (ORB, SIFT or SURF). For example, if we observe the maximum value for the good matches while using the SIFT algorithm is 65, and the observed minimum value is 10 for the unlikely frames, the value of minimum matches can be defined as 37.5 (the median of range 10-65).

6.3 Results with Adjusted Parameters

We used the minimum keypoints as 5000, and the ratio filter as 75% quantile and a fine-tuned minimum matches (based on algorithm used), to optimize the feature-based template matching algorithm used by the proposed system. We used the same manually rotated and scaled single-camera soccer video and adjusted the parameters of the proposed system as discussed in Section 6.2. We now observe and discuss our observations for the soccer video.

6.3.1 Results on Soccer Video

When we ran our fine-tuned system, we plotted the obtained likelihood values in the Figure 6.12. Comparing this to Figure 5.26, we can observe that the keypoints detected provided fewer good matches for ORB because it uses FAST algorithm to determine the keypoints. The orientation is given by the direction of the vector from the N detected keypoints, which convert into moments that make ORB rotation invariant. The keypoints are rotation invariant but fail to be scale-invariant, so we use SIFT and SURF to overcome the limitation of ORB. SIFT and SURF produce different distinctive results than ORB justifying to be rotation and scale-invariant. In Figure 6.12. the Y-axis shows the likelihood value for its respective frame in the X-axis. To observe the accuracy of the obtained results, we compared the obtained values with the ground truth values.
The green line in Figure 6.12 is the manually labelled ground truth value for the frame. In the given graph, according to the ground truth values, the frames between 3450 and 4070 are rotated as well as scaled. As we can observe for a short soccer video, despite obtaining

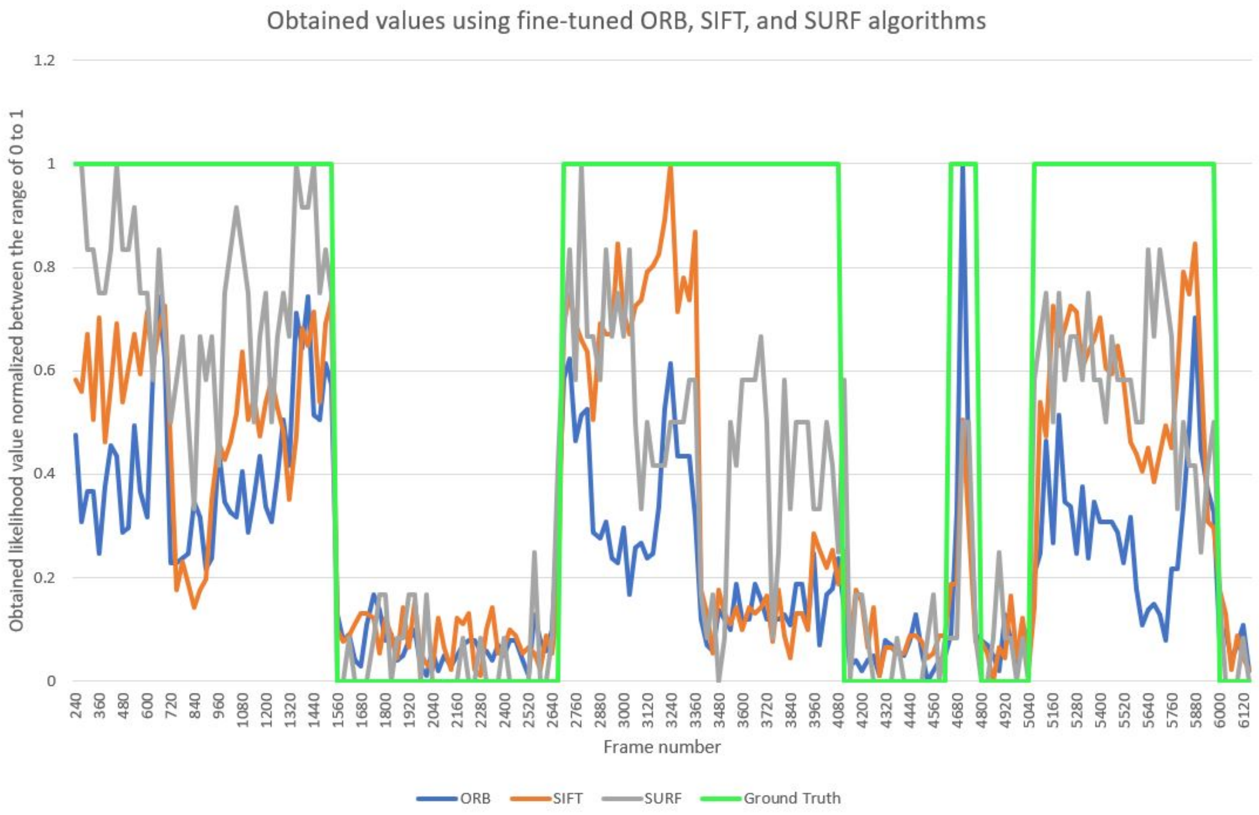


Figure 6.12: Visualization of the values obtained by running FLANN based algorithms in comparison to original values.

lower likelihood values than other algorithms, ORB detect most of the frames except the scaled frames. SIFT detects more scaled frames in addition to what ORB detects. SIFT is rotation and scale-invariant but fails to detect some scaled frames of the input video. SURF locates fewer keypoints, making the algorithm faster to process than SIFT, but detects more of the rotated and scaled frames between frames 3450 to 4070.

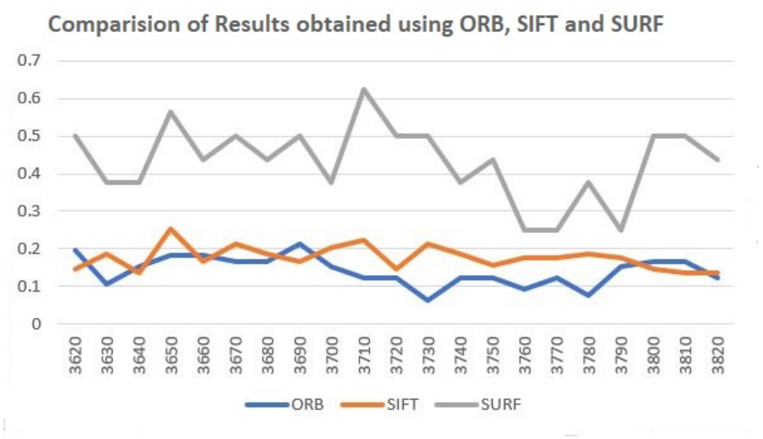


Figure 6.13: comparison of obtained likelihood values with all three algorithms within frames 3620-3820 in detail.

We can observe the obtained value for the transformed frames based on ORB, SIFT, and SURF in the given figures above. We can also observe that SURF received more accurate values within the frames than other methods within frames 3620-3820 in Figure 6.13, potentially due to various reasons. The reasons include algorithm used to detect keypoints, number of detected regions, and number of matched keypoints. Unlike the other two methods, SURF uses the Hessian matrix approximation method to use box filters for feature extraction despite fewer keypoints allocation. It uses the second-order Gaussian derivatives and is analyzed at a lower computational cost independent of image size. These are a few likely reasons why SURF is faster and gives approximate, rounded values with increased accuracy than SIFT and ORB. With the use of SURF, the obtained likelihood values can be distinctly segregated into likely and unlikely frames, as shown in Figure 6.13. We conclude that SURF gives the optimal solution for transformation invariance of a manually rotated and scaled single-camera soccer video.

To analyze the effectiveness of the algorithms and discuss other limitations like illuminance, video quality and environmental conditions like weather that may affect the single-camera video, next we used a single-camera soccer video played in the football field and captured in rain. The rain obstructs the frames and hampers the clarity of the video frame creating potential challenges for template matching.

6.3.2 Invariance to Video Frame Transformation and Environmental Hindrances

Many factors affect the results obtained by a video analyzing system, including the video quality and angle of the video view. While using feature-based template matching, we also analyzed the single-camera videos with variable visibility and weather conditions during this study. Any single-camera video quality can be affected by environmental factors, including the weather, the recording skills of the person, focus, and camera movements. To study the difference between the results obtained based on these factors, we used an input video that contained such factors that can affect the outcome. We used a similar transformed 174 second long soccer video played on a football field as shown in Figure 6.14. It was raining, which blurred some of the pixels of the image frames, degrading the quality of the picture. We then used this video as an input for the proposed tool and analyzed the obtained results. We came across various interesting observations while using SIFT and

SURF for the new soccer video, which we will discuss in this section.

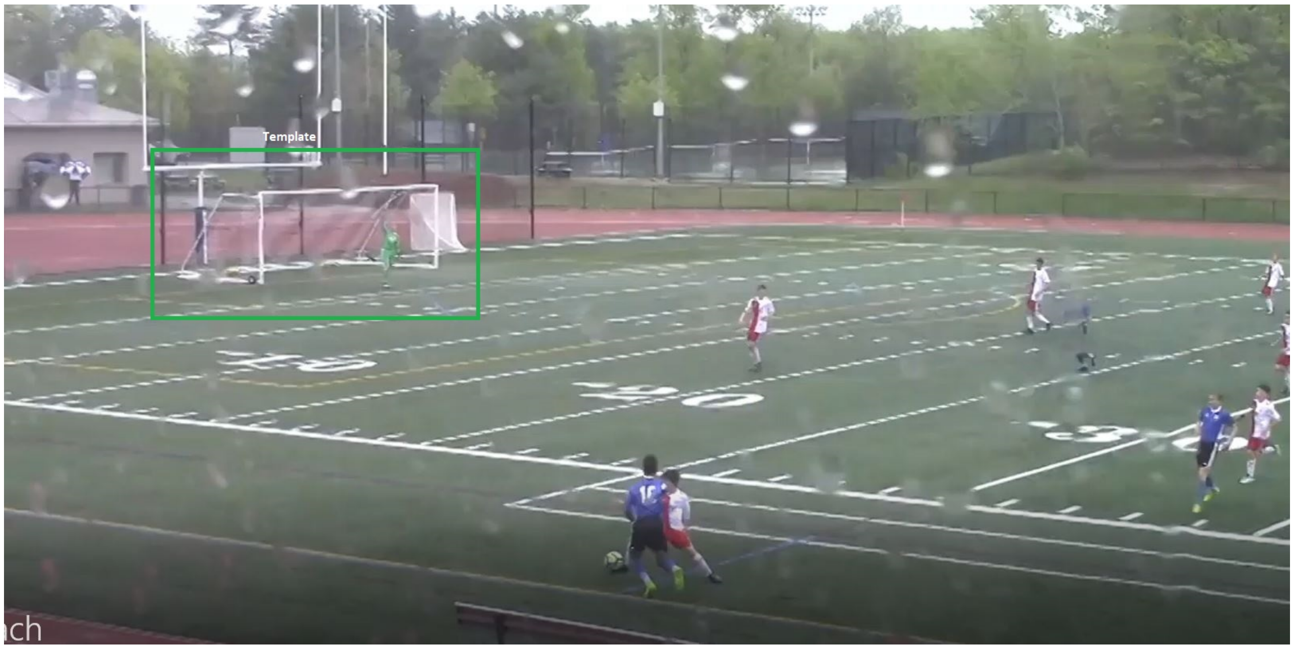


Figure 6.14: The soccer video captured in a football field when it is raining. The rain affects the image quality and the additional white lines of the football field may confuse the matching algorithm. The green rectangular selected area in the image frame is the selected template for the resultant observations.

Figure 6.15 shows contradicting results to the soccer video that we used initially for this research. SURF performed the best for the primary soccer video under normal circumstances regarding prominent run-time and increased accuracy when we compared the obtained likelihood values with the ground truth values. Contrary to that, SURF gives inconsistent likelihood values for this second soccer video with poor visibility when compared to ORB and SIFT (ORB and SIFT seem to perform similarly as observed from Figure 6.15). Since this video is shorter than our primary soccer video, we skipped the algorithm to run through every 10 target frames. The obtained results are presented and individually labelled in Figure 6.16.

While the input video scans through SIFT as shown in the Figure 6.16, the obtained

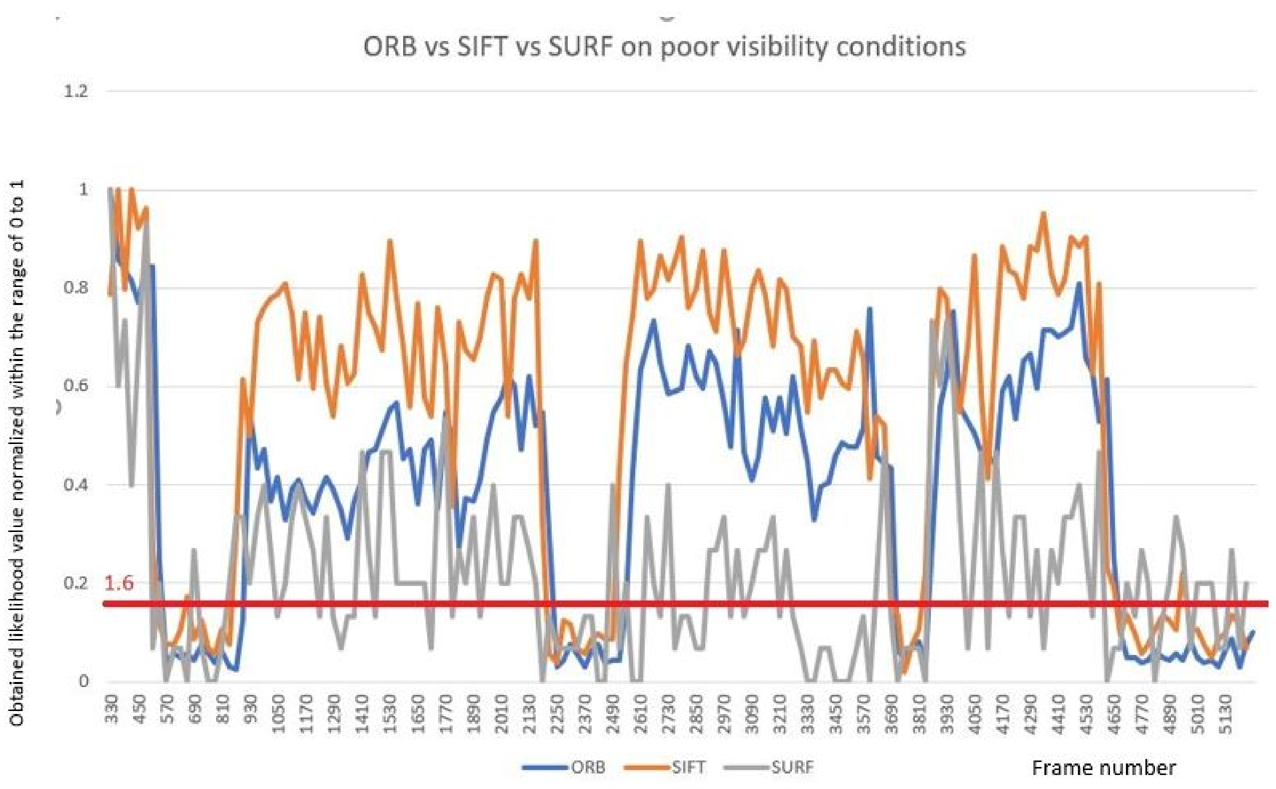


Figure 6.15: Comparison of the obtained results when ORB, SIFT and SURF algorithms are used on a single-camera video with poor visibility conditions.

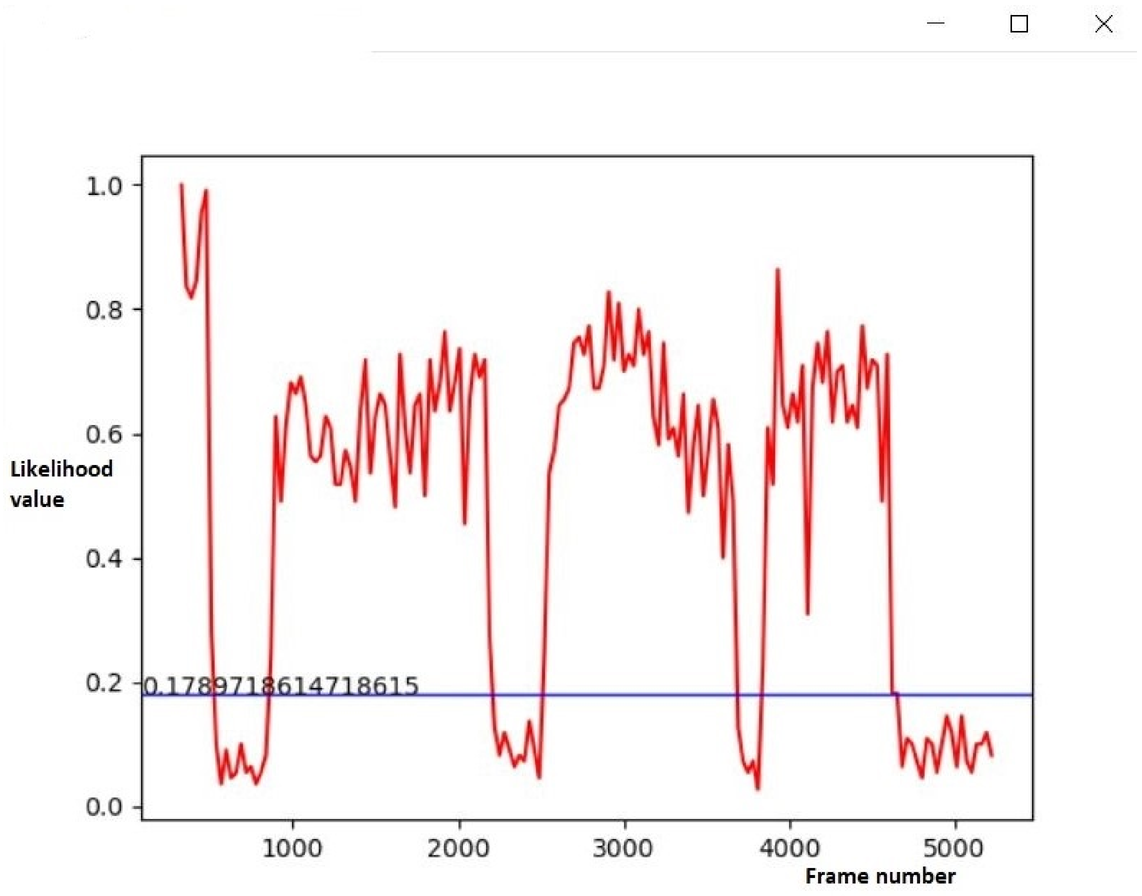


Figure 6.16: The obtained likelihood value for the poor quality soccer video with visibility conditions using SIFT.

likelihood values are distinctive. All the transformed frames are also detected that resemble the detection pattern of the initial soccer video. Similarly, when we run the same input video on SURF, we observe inconsistent values as shown in Figure 6.17. The inconsistency can be a result of various factors, including the weather conditions that affected the video quality in the first place. Based on the keypoint localization by determining the Hessian matrix, SURF detects fewer keypoints. These keypoints fail to match and decrease the number of resulting good matches, decreasing the likelihood. It eventually leads to the detection of false-negative values in each frame, and an inconsistent series of values is plotted, as shown for frames 800-1200 in the Figure 6.17.

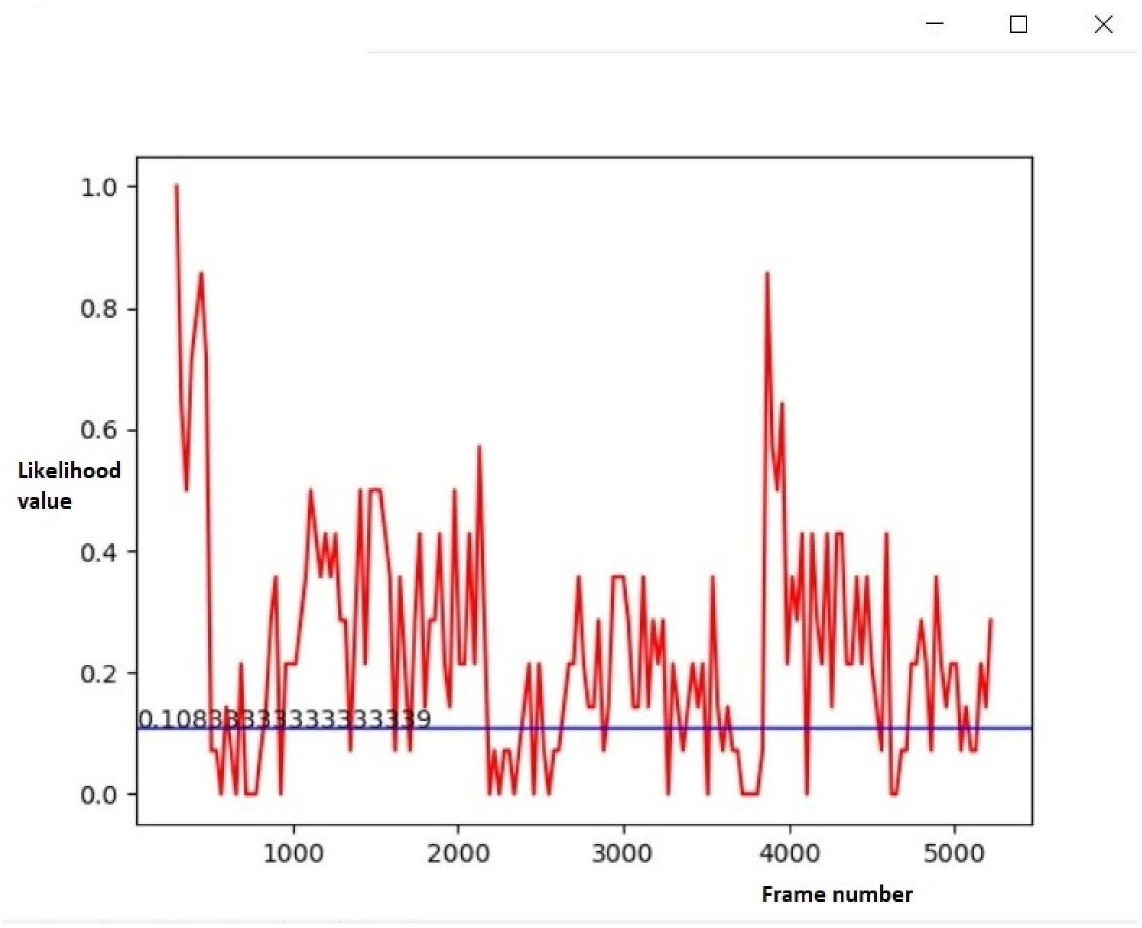


Figure 6.17: The obtained likelihood value for the poor quality soccer video with poor visibility conditions using SURF.

6.3.3 Application to Other Sports

We selected a similar template for the hockey video to make observations. As in the soccer video, the goal was the template for the hockey video based on which the tool ran the feature-based matching as shown in Figure 6.18. We ran the video on all three algorithms, ORB, SIFT, and SURF. The obtained results on all three algorithms gave different insights than the results obtained from the soccer video. The inconsistent results may be because of various prominent factors affecting the input video, including the video resolution and other external factors that we discussed in detail here. While running the system using all three algorithms (from Figure 6.19), we observed that SURF detects more false-positive frames when compared against ORB and SIFT. The method SURF uses to detect the initial keypoints, differs from the keypoint localization method of ORB and SIFT. Other reasons include the template and its feature matching ability when the template itself is conflicting. The goalkeeper in hockey quite often occludes the view of the goal because of the heavy gears and giant structures and the smaller goal, unlike in the soccer game where the goal is much larger and is reasonably visible. The result is a conflict in the template itself as many players are detected as the positive frames, which are the false positive values.

As shown in Figure 6.19, we observe that SIFT and ORB detect the frames better than SURF, which detects many frames that do not include the template compared with the ground-truth values represented by the green line. We conclude, at instances where the template is not fairly distinct for feature matching, SURF can detect false positives that overlap the actual values for a few frames. The Figure 6.19 shows that any of the algorithms do not detect the frames between 3780 and 4000. On the other hand, all the

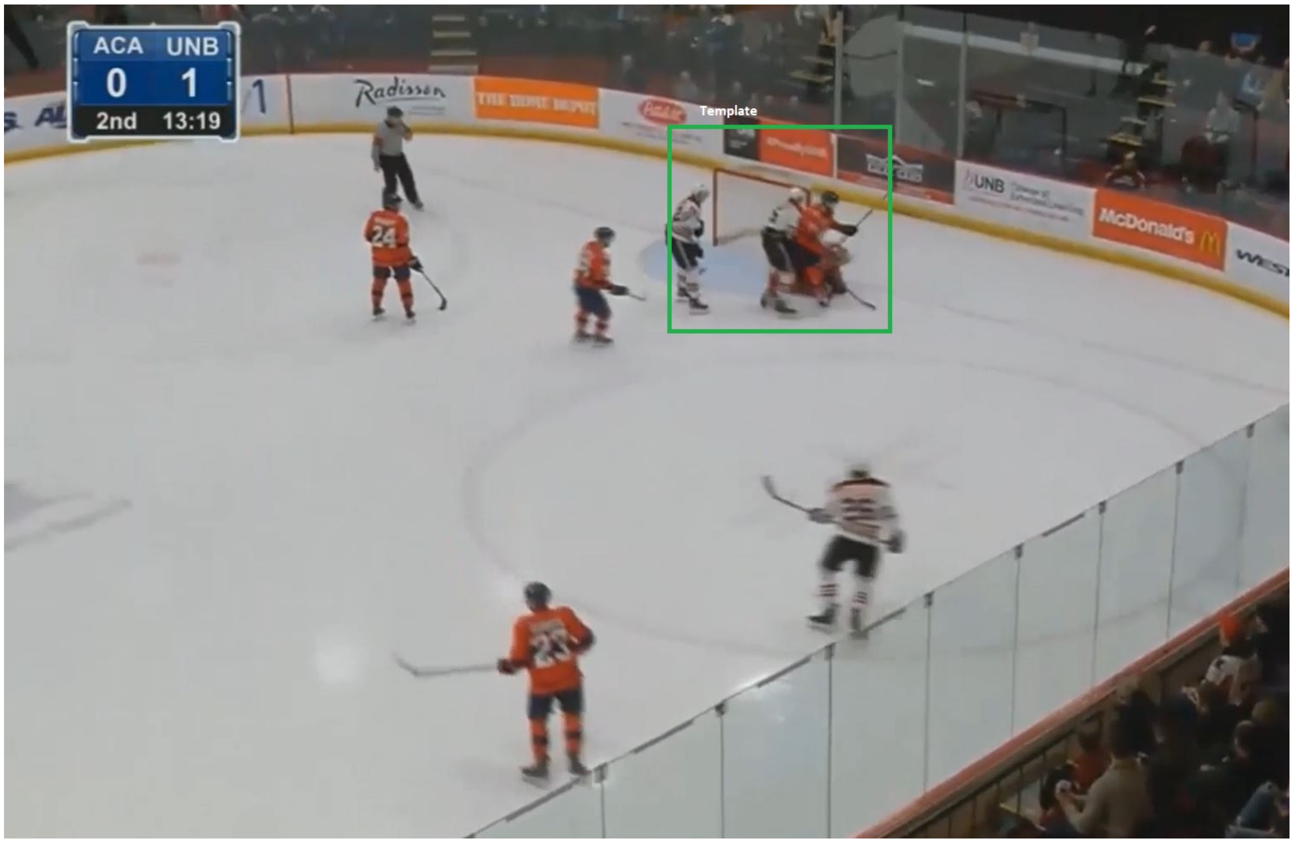


Figure 6.18: The green rectangular selected area in the image frame is the selected template for the resultant observations.

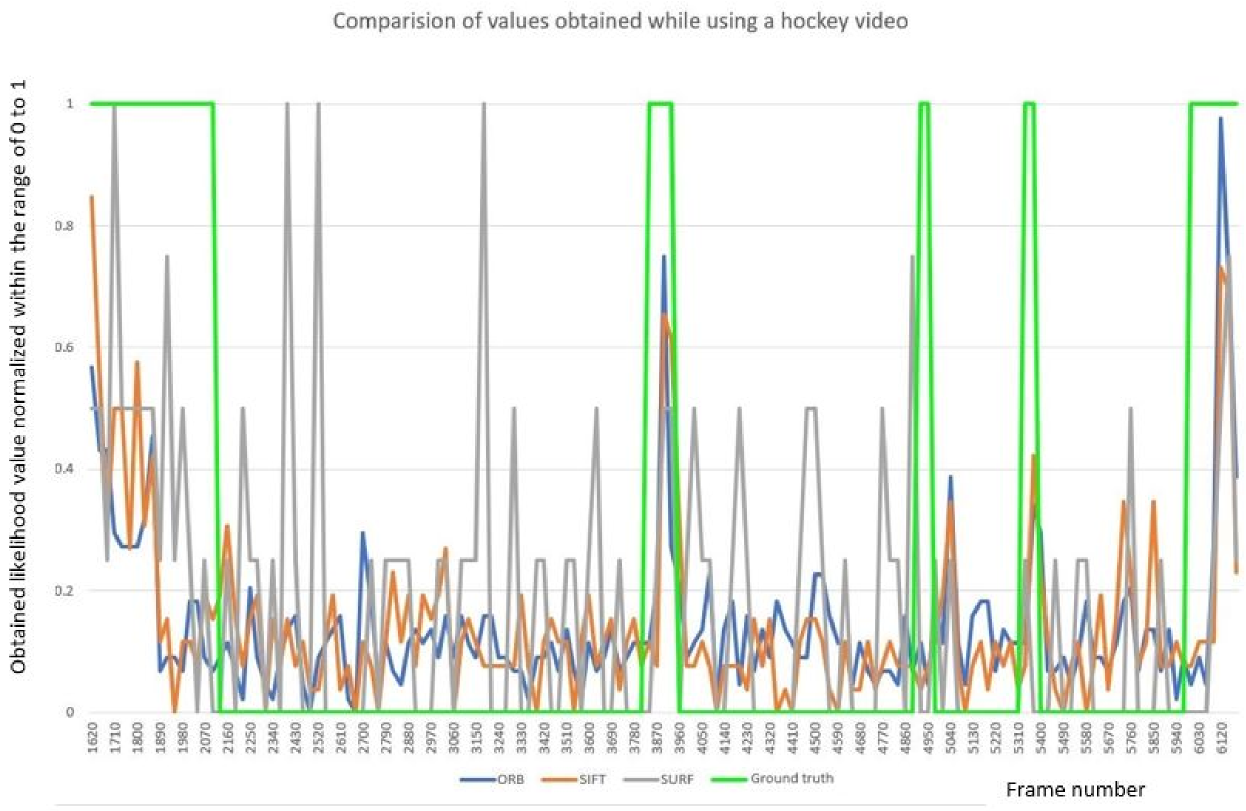


Figure 6.19: The green line is the ground truth values for the hockey video. We can observe that the obtained likelihood values are inconsistent using all three algorithms which impacts the user's decision of selecting a threshold that segregates the likely and unlikely frames.

frames between 5300 and 5400 are detected by SIFT, SURF and ORB at some levels.
Overall, we can visualize that SIFT works distinctly well when it comes to the circumstances and the external factors that may hamper the tool's feature matching abilities like under different environmental condition. We observed that SURF did not comprehend the obtained likelihood accurately for a video with limitations like illuminance, video quality and weather, despite giving distinctive likelihood values for the primary rotated and scaled soccer video.

What we found from this and all the previous experiment is that we cannot safely determine which of those 3 algorithms to use for which conditions as all the three algorithms are rotation and scale invariant at some levels. All the three algorithms give distinctive results driven by numerous factors as discussed in this section. The solution for this is to choose one specific algorithm that gives closest results which is easy for the users to comprehend.

We also experimented on a single-camera basketball video as shown in Figure 6.20.

As observed for obtained results in Figure 6.21, there are some inconsistencies in the likelihood value due to many possible reasons. One of them is many similar-looking baskets in the background as seen in Figure 6.20, alongside the fast movement of the players.
When we used the basketball video, we observed that SIFT detects most of the true positives resulting into one of the optimal algorithms for this type of video. SURF on the other hand has lower likelihood values which may be because matching the features are compared only if they have same type of contrast (based on signs) which allows faster but non optimal matching. We also observe that the frames 4680-4920 are not detected by any

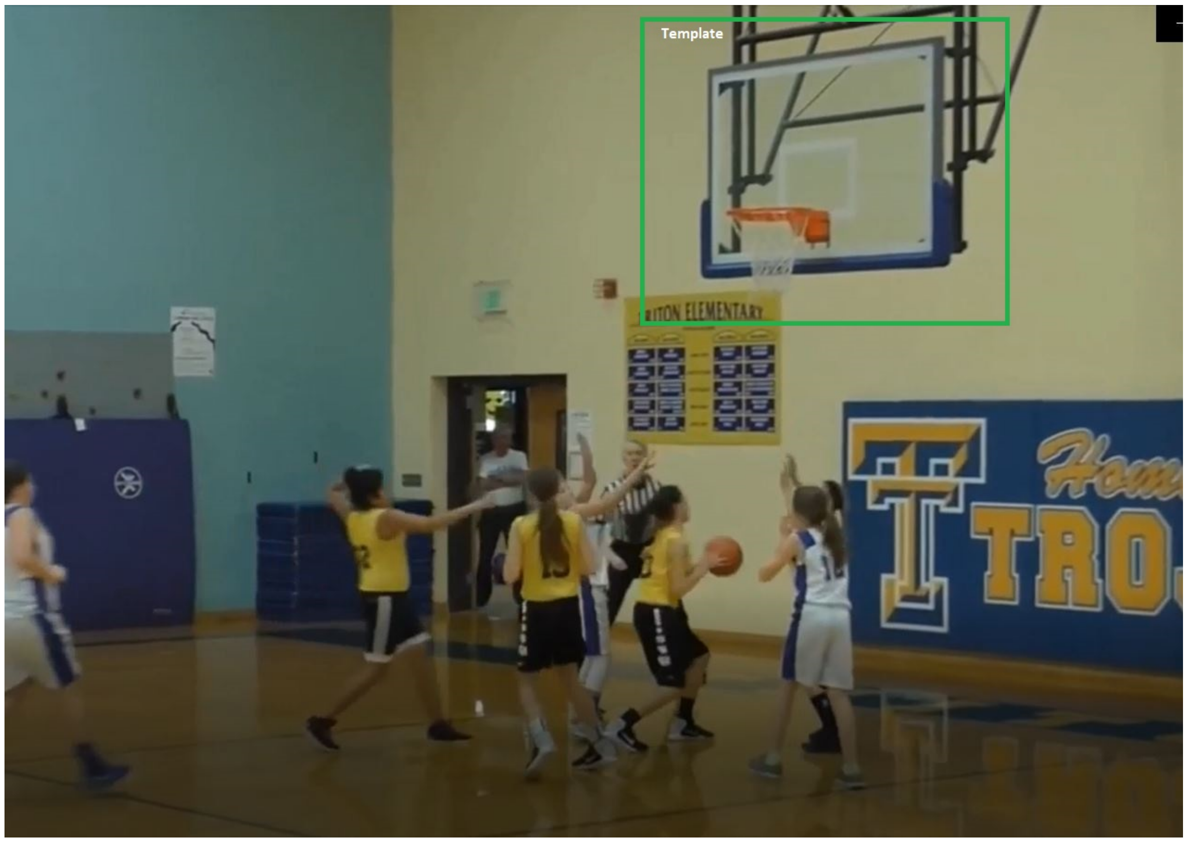


Figure 6.20: The basket ball video frame with the green rectangular area showing the user-selected template for the video.

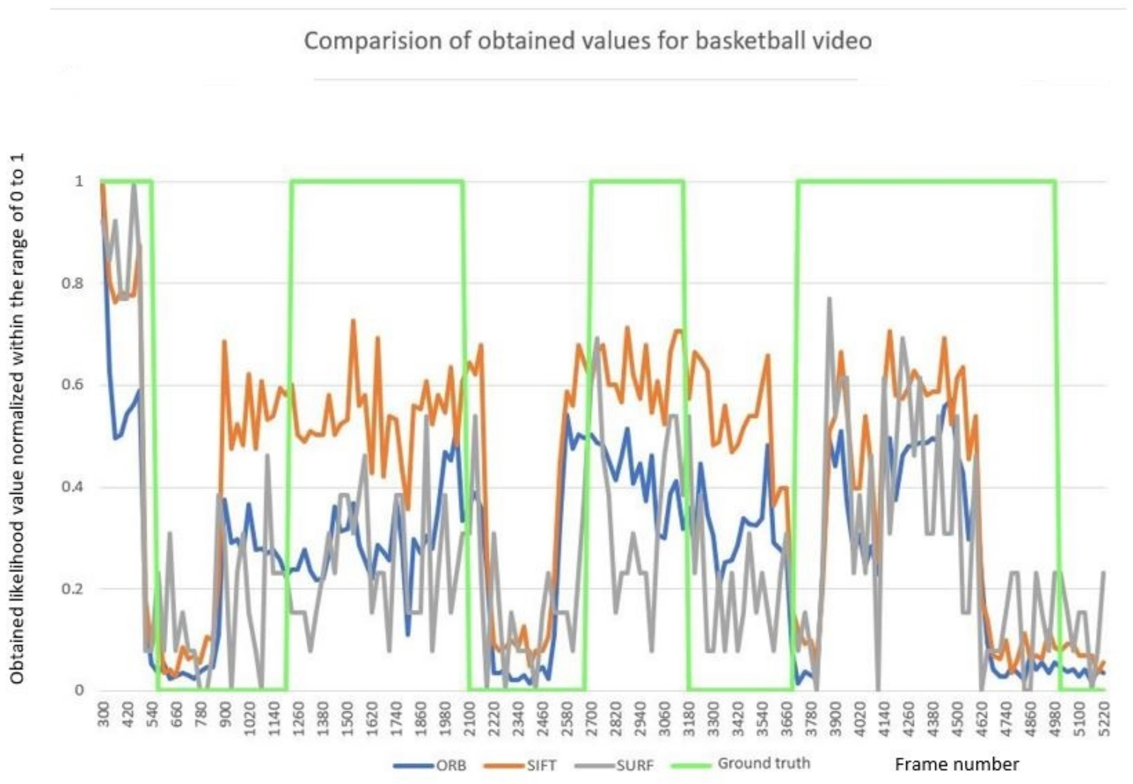


Figure 6.21: Comparison between the obtained likelihood value using all three different functions of template matching for a basketball video.

of the algorithms because the template is barely present in the frame as the object is projected towards the holder (i.e. the referee in this case). Although the multiple occurrences of the template object in the target frame helped to obtain distinct likelihood value, the distinction between the likely and unlikely frames from the obtained value is still difficult for the users using SURF algorithm when compared to SIFT and ORB.

6.4 User Evaluation studies

We evaluated the performances of the template matching algorithms by conducting a set of user studies. For preliminary testing and usability study, it was necessary to understand the user perspectives on the outcomes because this interactive video analysis tool is designed to support humans, so we decided to ask the users to observe the output videos and note their observations. The users' ratings were then compared against our detailed analyses of these feature-matching algorithms. Note that the sampling size of this user study was very small, and thus the results obtained here do not carry any statistical significance. We, however, included the results nonetheless to qualitatively illustrate how human evaluations of these algorithms may differ from what we observed. In order to analyze more reliable statistical significance of the data, we will need to conduct larger scale studies with more detailed comparisons of these algorithms.

Feature based template matching was the algorithm we chose to overcome the existing limitations discussed in Chapter 6. Specifically, we compared the three algorithms for the feature-based template matching algorithm, namely, ORB, SIFT, and SURF, on the three different videos (i.e., single-camera soccer, hockey and basketball videos with rotated and scaled frames). For each of the three videos, we created three output videos, based on the three algorithms (i.e., the total of nine videos). We then asked the participants to compare the algorithms by viewing the resulting output videos. The end-users are the stakeholders who can provide their honest opinion without any detailed knowledge of the backend components. It allowed us to analyze our observations that can sometimes get influenced by a lot of work on the backend. In our case, we needed an unbiased third party to observe our results. It also helped us decide if we chose the appropriate method for feature-based feature matching. We gave the users some measurable targets, and they tried to test the performance of the prototype. The investigator observed, noted, and recorded all the responses.

We started the user evaluation study with a questionnaire. The number of questions depended on the number of input videos i.e., three for this study. The users were to observe the input video, the template that was used to extract video segments, and the three output videos. They were then asked to choose one output video that they thought was the best one (based on the performance/accuracy of the goal frame detection). Their selection was then recorded and then compared and analyzed together with our findings. The participants finally answered the questionnaire asking about the tool and we noted their responses.

Eight participants participated in this user-study. It included 3 responses from new participants and 5 responses from the participants that took part in the preliminary studies.

The first input video was the primary soccer video for this research (used in Section 6.2.1).

In Figure 6.22 , we observed that 6 participants (75%) selected the video extracted by SIFT as the best video. We monitored the user responses and compared it with our observations as discussed in this section.

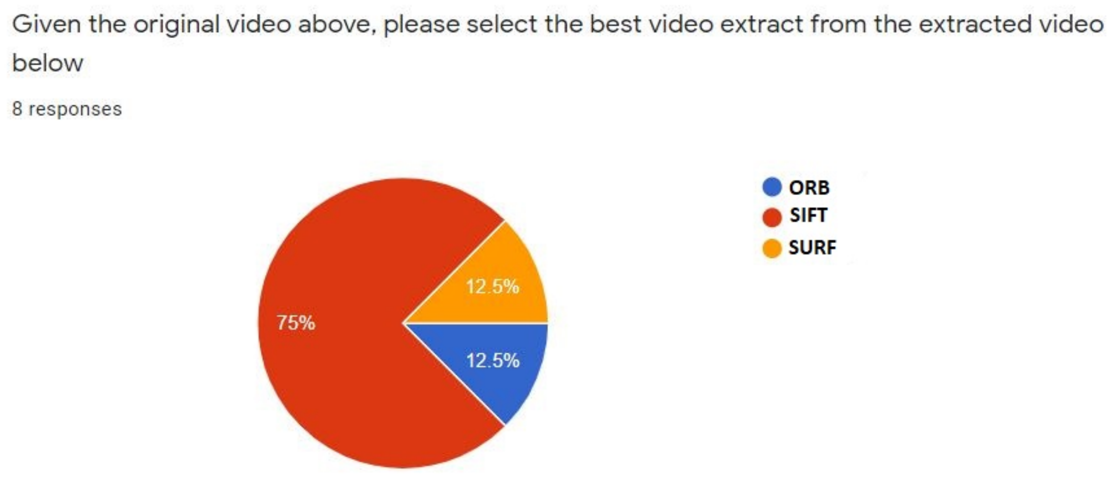


Figure 6.22: Analyzing the participants responses on the feature-based template matching algorithm for soccer video for scaled and rotated frames.

We observed that only 1 participant (12.5%), selected the output video extracted by ORB and SURF each. We previously deduced that SURF was the best method to work with the transformed single-camera video based on the optimal results and runtime as discussed in Section 6.2. The user evaluation in Figure 6.22 shows the user's inclination towards SIFT while used on a transformed soccer video. It is because of the scale and rotation invariant properties of SIFT. The users may have been unable to perceive some of the missed frames when using SIFT.

The second input video was a soccer video captured in poor environmental conditions with rain and wind, played in a football field (the same video used for analyzing hockey video in Section 6.3.3). It included the frames with disrupted visibility due to rain and had multiple goalposts that could impact the resultant video. We provided the three output videos extracted using ORB, SIFT, and SURF accordingly and found a little diverse result than from the first input video. 4 participants (50%) found the video resulted using SIFT to be the best, and 2 participants (25%) of the users found SURF to work the best, and the remaining 2 participants (25%) found the video extracted by ORB to be the best as shown in Figure 6.23. The lower responses for ORB shows the ineffectiveness of ORB for poor

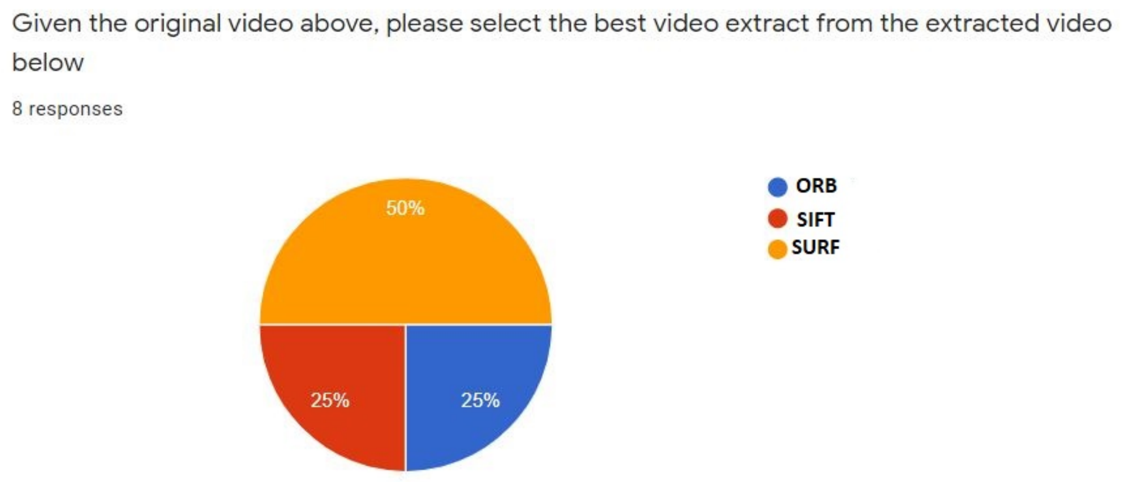
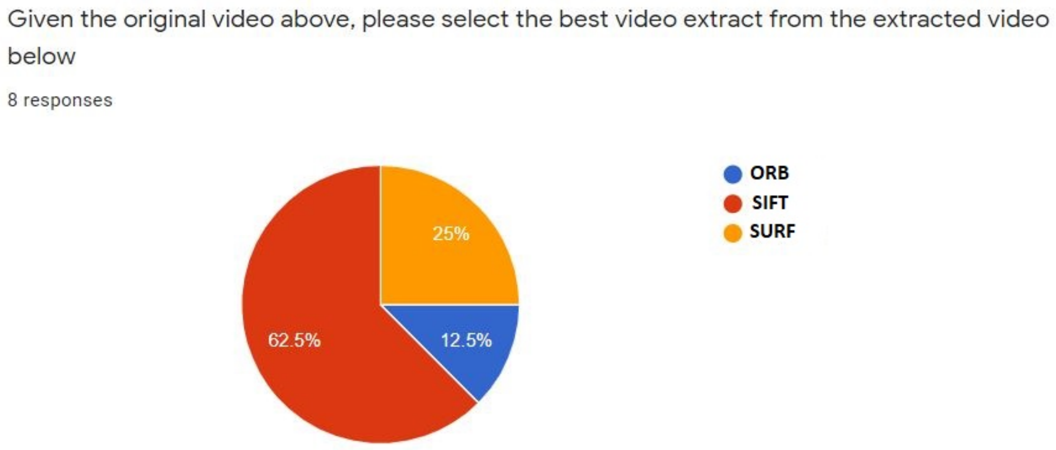


Figure 6.23: Analyzing the participants responses on the feature-based template matching algorithm for soccer video on poor video visibility.

visibility conditions from the user perspective. This finding in fact contradicts with our research findings which is discussed in detail in Section 6.3.2. It may be because SURF speeds up the localization process and detects a lot of false-positive frames, which ends up with giving a longer output video that may distract the user.

We now move on to our third and final video that is a hockey video because for the fast-paced games like hockey, it is difficult to interact with the input video. It is hard to select a clean template for the system. We used a hockey video as a reference to other fast-paced games that may benefit from this research. From user responses, we observed that only 1 participant (12.5%) selected video extracted using ORB, 2 participants selected extracted video using SURF and 5 participants (62.5%) selected the video extracted by using SIFT algorithms as shown in Figure 6.24. We noted that the user evaluation and the


Given the original video above, please select the best video extract from the extracted video below

8 responses

Figure 6.24: Analyzing the participants responses on the feature-based template matching algorithm for a hockey video.

research study gave similar results for SIFT on this video. Despite many visible shortcomings for the fast-paced game, SIFT works best for the limitations surrounding invariances than other feature-based template matching algorithms. The user response in Figure 6.24 also justifies this conclusion.

The overall comparison of the obtained result with the user-studies is shown in Table 6.3. From the user observation and obtained feedback presented in the Table 6.3, the results for the illuminance and other invariances matched the user responses as highlighted by the green boxes in the table. The only result that contradicts is our primary soccer video for which the user observed SIFT to be the algorithm giving optimal result for the soccer video. We decided to experiment on the poor visibility video one more time by fine-tuning

	Video type	**ORB**	**SIFT**	**SURF**	**Template-based**
Analysis on the frame detection	**Soccer video (transformation)**	POOR	GOOD	GREAT	POOR
	Soccer video (poor visibility)	GOOD	GREAT	POOR	POOR
	Hockey video	GOOD	GREAT	POOR	POOR
	Basketball video	GOOD	GREAT	POOR	GOOD
User Evaluation	**Soccer video (transformation)**	POOR	GREAT	POOR	(N/A)
	Soccer video (poor visibility)	GOOD	GREAT	GOOD	(N/A)
	Hockey video	POOR	GREAT	GOOD	(N/A)

Table 6.3: Comparisons of the three feature-based matching algorithms and the template-based matching algorithm on different single-camera video types. The results obtained using feature-based template matching is also compared to the user-evaluation studies. In the figure, the green box denotes the combined results and the red box shows the contradictions.

the parameters (results discussed on Section 6.2), which gave us the same results. SIFT worked best for the video with poor visibility conditions as well. As 2 out of three questions of the overall user-evaluation study overlapped the ongoing research and there are factors that may have influenced the user-studies, we concluded that feature-based template matching using SIFT or SURF tended to produce better results than with ORB for the videos we have tested in this research.

From the detailed observations and discussions of the obtained results in Section 6.2.1, we observed that SURF is the most efficient feature-based template matching algorithm. To justify our observation, we also compared the obtained results for all the three algorithms, ORB, SIFT, and SURF, based on the user evaluation studies in Section 6.4. The results obtained from the user evaluation studies support our research findings and indicates that SIFT provides optimal results for transformation invariance. Despite achieving transformation invariance, there is still a lot of room for improvement in the current research. We further discuss the findings, the limitations, and possible future work based on this study in Chapter 7.

Chapter 7

Conclusion and Future Work

In this chapter, we will summarize all our findings, point out the limitations of the study while discussing the obtained results, and list the possible future direction for the research.

The surge in the use of multimedia data has created challenges for useful knowledge extraction from multimedia data analysis with the rise in multimedia databases [4]. The study presented in this thesis attempted to take on one such challenge: achieving the transformation invariance using a template matching algorithm for single-camera videos. We started this thesis research by conducting a general survey (as discussed in Section 5.1) to understand our problem domain and to analyze the user's perspective on the feasibility of this research. The survey results and its analysis helped us to lay out a workflow and the measurable targets on the way for this research. We experimented with different supervised algorithms, like CNN and YOLO (as discussed in Chapter 5, Section 2.2.3), but we did not proceed with supervised algorithms partly due to the lack of relevant training datasets despite some positive initial results. Unavailability of huge diverse dataset was also one of the reasons we did not move forward with self-supervised learning approaches. This direction may still have the potential to be pursued further for the purposes similar to our domain. Instead, we found a closely related solution using feature-based template matching to overcome some of the limitations of template-based template matching.

The background study of the algorithms based on the template-based approach led us to the detailed analysis of the feature-based template matching approach (as discussed in Section 5.3.2). We narrowed down the relevant algorithms of feature-based template matching as ORB, SIFT, and SURF to be the most effective for this research. The three algorithms helped us achieve some level of transformation invariance (by matching zoomed and rotated frames). ORB helped achieve rotation invariance only. SIFT and SURF, on the other hand, helped achieve rotation and scaling invariance, while SURF also working effectively for light and poor visibility conditions. We will now discuss the summary of the obtained results, the future work, and possible research in the upcoming sections.

7.1 Research Questions and Findings

This research initially investigated matters in the problem domain of analysis tools for single-camera videos. In particular, we tried to answer the research questions as listed in Section 4.1. First, we studied and experimented with all three feature-based template matching algorithms, ORB, SIFT, and SURF, to achieve the zoomed and scaled invariance. In the process, we also observed that the proposed feature-based matching algorithms can overcome some of the limitations while keeping the user's ability to choose the threshold intact by providing distinct likelihood values for the frames. These observations based on the raised research questions (Section 4.1) are answered and discussed here.

7.1.1 Research Question 1

How can we improve efficiency and accuracy for analyzing multimedia data while allowing the users to handle the interactive interface and control the results? The first step to increase the efficiency and to analyze any interactive interface is by conducting a qualitative study. Human participation for the background study of the research and collecting feedback from them on the subject matter was what allowed us to understand the problem domain. Once we understood the problem domain, the user feedback modified and improved our proposed approaches based on usability. It also helped us understand the application and feasibility of the single-camera video extraction in real-time. We also noted that the accuracy of the user-interactive system by comparing the obtained likelihood values to actual ground truth. This helped improves the effectiveness of the algorithm. The final results should be analyzed by the human participants to determine if the goal is met (as we discussed in Section 6.4). For this research, we conducted a human-centered qualitative study (as discussed in in Section 5.1) and once again collected user feedbacks from user-evaluation studies as discussed in Section 6.4. As this study was conducted with a small sample of users, a larger-scale study will be needed to further investigate the human users' perception of the output videos as well as the tool's usability.

7.1.2 Research Question 2

Are there enough datasets to train a predictive model for the single-camera video extraction? If not, is it possible to use supervised machine learning algorithms without the presence of an optimal dataset for single-camera videos? Our study is based on the extraction of a single-camera soccer video, and when searched for the relevant training dataset, we did not come across any dataset pool. As explained in Section 5.3.1, even the available relevant data has some shortcoming. Some of the images have mismatched angles, i.e., too zoomed in/out, shot from a different angle, a professional game shot which makes it even difficult to find a specific dataset to effectively train a model. It is possible to create your own dataset for optimal results and increased accuracy, or you can use whatever dataset is available, but the results would not be the same. You can also create your own dataset, which will be a time-consuming process.

Despite overcoming some of the limitations of the template-based template matching, there is still a lot of room for improvement in the current approach that we further discuss in detail in the next section.

7.1.3 Research Question 3

What algorithms work well with the video analysis for videos containing different circumstances, like a rotated or zoomed frame or illuminance, etc., and provide optimal results? Is there an algorithm for template matching that helps achieve transformation (i.e., zooming and scaling) invariance and provides optimal results if we use an inconsistent varying quality single-camera video? After the thorough background study and experimentation, we found different approaches that can possibly provide optimal results under the noted limitations (i.e., scaled and zoomed frames, illuminance, visibility conditions and so on). When the relevant training dataset was available, the supervised learning algorithms would potentially have helped achieve transformation invariance with increased accuracy. In the absence of the sufficient dataset, it was not optimal for us to use supervised learning algorithms given our timeframe and resources that were available; instead, we can use feature-based template matching approaches. Among the feature-based approaches, the FLANN-based matcher approach worked the best for achieving the transformation invariance as explained in Section 5.3.2. ORB is rotation invariant and partially scale-invariant, and the SIFT and SURF both are scale and rotation invariant. SIFT and SURF also showed promising results in poor resolution and poor visibility conditions (as discussed in Section 6.3.2). There are other approaches that were not feasible with this research as discussed in Section 5.3.

As discussed in Chapter 6, feature-based template matching is transformation invariant. Feature-based template matching will use the keypoints and its descriptors to detect the template instead of using the value for the pixel's location unlike in the template-based template matching approach. ORB and SIFT also worked effectively under poor visibility conditions and other factors such as illuminance as discussed in Section 6.3.2.

7.2 Limitations of the study

There are some notable limitations of the feature-based feature matching, which can motivate further research and shape the future direction for a relevant study. System run-time is one of the first areas in the current study that needs to be worked on for feature-based template matching. Executing feature-based template matching consumes more time than the existing template-based approaches do because the parameters pass through multiple processes. For matching the keypoints in feature-based approach, each pixel must be compared with all the nearest surrounding pixels in the matching window to get the optimal keypoint. The algorithm then calculates the value in scale space for each pixel. The keypoint will then be used as a reference in the sliding. The feature-based matching process involves keypoint localization and ratio filtering for SIFT and SURF, which overall consumes more processing time than template-based template matching. The processing time can be reduced by skipping a few windows without impacting the results. We will need to study how the efficiency of the current approach can be optimized while retaining the accuracy of the results.

We used different arbitrary parameter values to build the proposed system. Some of the parameters are pre-defined, like minimum value and good matches as explained and the parameters are fine-tuned in Section 6.2. The inability to use ad-hoc parameters to fine-tune the current approach is another visible limitation. We will thus need to further explore algorithms or systematic methods to determine the parameter values based on types and quality of input videos.

We experimented mostly on soccer, and a few other sports videos, which makes this research is limited to a specific problem domain. While it may be easier to imagine how we may extrapolate this technique to other similar sports that may have distinct templates such as soccer, basketball, and hockey, the scalability to other types of sports and activities such as dancing or wedding videos is unknown. It would be interesting to explore and understand the scalability and applicability of the current approach to other domains.

Matching the template with every pixel and calculating the value of each video frame among the pixels is time-consuming for longer videos with numerous video frames. The efficiency of the overall research is limited because the algorithm that we use passes through every 30 frames calculating the likelihood values for the frames is not in real-time (online). However, since each frame's analysis is independent on the previous frames, the real-time analysis while the user watches the video for the first time can be accomplished. Once it is done, the analysis data is then stored and attached to the video, thus provided as the off-line data analysis results for the future use. That is, the analysis needs to be done only once for each video, and the likelihood plots can be shown for any future usages.

7.3 Future work

The first possibility observed is creating a relevant training dataset for using supervised learning algorithms. It can replenish the availability of the relevant single-camera video training dataset for the supervised algorithm as discussed in Chapter 5. Feature-based template matching extracts narrowed down video frames containing the template, and those frames can be augmented to create a bigger training dataset. The current approach of feature-based template matching can be used to first narrow down the search space of relevant video frames that contain target objects. This elimination of irrelevant video frames can significantly reduce the amount of time to find target frames. The extracted frames can then be used to label objects in these videos in order to create a dataset to be used as training data. This training data can then be augmented to create an efficient dataset that can be used to train single-camera object detection models using different supervised learning algorithms.

Another scope of focus for further study of this research is aspects of the human-computer interaction. In the current research, the focus was heavily on the data analytics but designing interactive tools such as the one presented in this thesis typically requires the user-centred design approaches. While we continue to improve the back-end video data analytics algorithms, the front-end design of the tool can be improved simultaneously by involving potential target user groups such as sport coaches and players who may use such tools and following the common user-centred design methodologies.

7.4 Conclusion

In conclusion, we have investigated and explored a set of analytical approaches for single-camera videos to identify video segments that contain user-defined objects/scenes. We also tried to observe if the approaches helped to overcome other environmental factors like poor visibility conditions. After a series of experiments, we found that the three feature-based template matching algorithms had the most optimal performance results, especially for videos with objects with variations in sizes and rotations. The three algorithms were Oriented FAST and Rotated Brief (ORB), Scale-Invariant Feature Transform (SIFT), and Speeded-Up Robust Features (SURF). Among them, SIFT and SURF improved the template-based template matching approach by overcoming the rotation and scaling limitations.

Once this tool is developed and implemented, we hope it helps to study the importance of interactive interfacing for multimedia data analysis. We also hope to improve this approach to test and modify the prototype and to analyze the videos from the larger sports community. This approach can also be modified for amateur players to help them analyze their performance when professional tools are unavailable. Finally, we hope that the implemented tool will reduce the time and increase the ease of creating new datasets for annotating video frames by narrowing down the likely frames that contain the target objects to be annotated. Such new datasets can then be used to improve the machine learning models for template-matching and video scene extractions.

Bibliography

[1] D. Tyagi, “Introduction to SIFT (Scale Invariant Feature Transform),” Apr 2020. [Online]. Available: https://medium.com/data-breach/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40

[2] S. Elfouly, “R-CNN (object detection). A beginners guide to one of the most… — by sharif elfouly — medium,” 2020. [Online]. Available: https://medium.com/@selfouly/r-cnn-3a9beddfd55a

[3] T. Li, M. Ogihara, and G. Tzanetakis, *Music Data Mining*, ser. Chapman & Hall/CRC Data Mining and Knowledge Discovery Series. Taylor & Francis, 2011. [Online]. Available: https://books.google.ca/books?id=_zc3vKDLUNIC

[4] C. A. Bhatt and M. S. Kankanhalli, “Multimedia data mining: state of the art and challenges,” *Multimedia Tools and Applications*, vol. 51, no. 1, pp. 35–76, 2011.

[5] M. Shouman, T. Turner, and R. Stocker, “Using data mining techniques in heart disease diagnosis and treatment,” in *2012 Japan-Egypt Conference on Electronics, Communications and Computers*. IEEE, 2012, pp. 173–177.

[6] I. Adjerid and K. Kelley, "Big data in psychology: A framework for research advancement." *American Psychologist*, vol. 73, no. 7, p. 899, 2018.

[7] L. Gao, J. Song, X. Liu, J. Shao, J. Liu, and J. Shao, "Learning in high-dimensional multimedia data: the state of the art," *Multimedia Systems*, vol. 23, no. 3, pp. 303–313, 2017.

[8] IBM, "2020 us open a grand slam for remote fans." [Online]. Available: https://www.ibm.com/case-studies/united-states-tennis-association-us-open/

[9] A. B. Watson and C. H. Null, "Digital images and human vision," 1997.

[10] J. Heer, "Agency plus automation: Designing artificial intelligence into interactive systems," *Proceedings of the National Academy of Sciences*, vol. 116, no. 6, pp. 1844–1850, 2019.

[11] L. F. Cranor, "A framework for reasoning about the human in the loop," 2008.

[12] Y. Akiyama, R. G. Barrantes, and T. Hynes, "Video scene extraction tool for soccer goalkeeper performance data analysis." in *IUI Workshops*, 2019.

[13] R. Aggarwal, A. Vohra, and A. M. Namboodiri, "Panoramic stereo videos with a single camera," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 3755–3763.

[14] P. M. Kamde, D. Algur *et al.*, "A survey on web multimedia mining," *arXiv preprint arXiv:1109.1145*, 2011.

[15] A. Goodrum, “Image information retrieval: An overview of current research,” *Informing Science*, vol. 3, p. 2000, 2000.

[16] W. Zhou, H. Li, and Q. Tian, “Recent advance in content-based image retrieval: A literature survey,” *CoRR*, vol. abs/1706.06064, 2017.

[17] X. Giró-i Nieto and M. Martos, “Interactive segmentation and tracking of video objects,” in *2012 13th International Workshop on Image Analysis for Multimedia Interactive Services.* IEEE, 2012, pp. 1–4.

[18] J. Gallego, M. Pardas, and M. Solano, “Foreground objects segmentation for moving camera scenarios based on scgmm,” in *Internatinoal Workshop on computational Intelligence for Multimedia Understanding.* Springer, 2011, pp. 195–206.

[19] X. Bai and G. Sapiro, “A geodesic framework for fast interactive image and video segmentation and matting,” in *2007 IEEE 11th International Conference on Computer Vision.* IEEE, 2007, pp. 1–8.

[20] B. L. Price, B. S. Morse, and S. Cohen, “Livecut: Learning-based interactive video segmentation by evaluation of multiple propagated cues,” in *2009 IEEE 12th International Conference on Computer Vision.* IEEE, 2009, pp. 779–786.

[21] D. Greig, B. Porteous, and A. Seheult, “Exact maximum a posteriori estimation for binary images,” *Journal of the Royal Statistical Society, Series B*, vol. 51, p. 271–279, 01 1989.

[22] L. Yang, X. Wu, Y. Guo, and S. Li, “An interactive video segmentation approach based on grabcut algorithm,” in *2011 4th International Congress on Image and Signal Processing*, vol. 1. IEEE, 2011, pp. 367–370.

[23] N. Shankar Nagaraja, F. R. Schmidt, and T. Brox, “Video segmentation with just a few strokes,” in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 3235–3243.

[24] T. V. Spina and A. X. Falcao, “Fomtrace: Interactive video segmentation by image graphs and fuzzy object models,” *arXiv preprint arXiv:1606.03369*, 2016.

[25] G. H. de Rosa, J. P. Papa, and A. X. Falcão, “Opfython: A python-inspired optimum-path forest classifier,” *arXiv preprint arXiv:2001.10420*, 2020.

[26] K.-K. Maninis, S. Caelles, J. Pont-Tuset, and L. Van Gool, “Deep extreme cut: From extreme points to object segmentation,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 616–625.

[27] J. Wang, Y. Zhao, Q. Qi, Q. Huo, J. Zou, C. Ge, and J. Liao, “Mindcamera: Interactive sketch-based image retrieval and synthesis,” *IEEE Access*, vol. 6, pp. 3765–3773, 2018.

[28] T. Bui and J. Collomosse, “Scalable sketch-based image retrieval using color gradient features,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2015, pp. 1–8.

[29] D. G. Viswanathan, “Features from accelerated segment test,” in *Proceedings of the 10th workshop on Image Analysis for Multimedia Interactive Services, London, UK*, 2009, pp. 6–8.

[30] D. G. Lowe, “Distinctive image features from scale-invariant keypoints,” *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004.

[31] J.-H. Hou and Y. Lin, “Adaptive harris x-corner detection algorithm,” *Computer Engineering and Design*, vol. 30, no. 20, pp. 4741–4743, 2009.

[32] Ü. Lepik and H. Hein, *Haar wavelets: with applications.* Springer Science & Business Media, 2014.

[33] A. Jakubović and J. Velagić, “Image feature matching and object detection using brute-force matchers,” in *2018 International Symposium ELMAR.* IEEE, 2018, pp. 83–86.

[34] H. Pokharna, “The best explanation of convolutional neural networks on the internet! — by harsh pokharna — technologymadeeasy — medium,” 2016. [Online]. Available: https://medium.com/technologymadeeasy/the-best-explanation-of-convolutional-neural-networks-on-the-internet-fbb8b1ad5df8

[35] R. Girshick, “Fast R-CNN,” in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[36] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, “Support vector machines,” *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[37] J. Hosang, R. Benenson, and B. Schiele, “A convnet for non-maximum suppression,” in *German Conference on Pattern Recognition.* Springer, 2016, pp. 192–204.

[38] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[39] B. Benjdira, T. Khursheed, A. Koubaa, A. Ammar, and K. Ouni, "Car detection using unmanned aerial vehicles: Comparison between faster R-CNN and YOLOV3," in *2019 1st International Conference on Unmanned Vehicle Systems-Oman (UVS)*. IEEE, 2019, pp. 1–6.

[40] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[41] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2117–2125.

[42] "7 object detection with R-CNN, SSD, and YOLO - deep learning for vision systems," 2021. [Online]. Available: https://livebook.manning.com/book/grokking-deep-learning-for-computer-vision/chapter-7/v-5/367

[43] B. Liu, X. Shu, and X. Wu, "Fast screening algorithm for rotation invariant template matching," in *2018 25th IEEE International Conference on Image Processing (ICIP)*. IEEE, 2018, pp. 3708–3712.

[44] A. M. Moussa, M. Habib, and R. Y. Rizk, "Frotema: Fast and robust template matching," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 6, no. 10, 2015.

[45] Z.-H. Zhou, "A brief introduction to weakly supervised learning," *National science review*, vol. 5, no. 1, pp. 44–53, 2018.

[46] L. Talker, Y. Moses, and I. Shimshoni, "Efficient sliding window computation for NN-based template matching," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 404–418.

[47] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[48] P. Goldsborough, "A tour of tensorflow," *arXiv preprint arXiv:1610.01178*, 2016.

[49] D. A. Van Dyk and X.-L. Meng, "The art of data augmentation," *Journal of Computational and Graphical Statistics*, vol. 10, no. 1, pp. 1–50, 2001.

[50] Tzutalin, "Tzutalin/labelimg: Labelimg is a graphical image annotation tool and label object bounding boxes in images." [Online]. Available: https://github.com/tzutalin/labelImg

[51] S. Vicente, J. Carreira, L. Agapito, and J. Batista, "Reconstructing pascal voc," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 41–48.

[52] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 781–10 790.

[53] M. Muja and D. G. Lowe, "Scalable nearest neighbor algorithms for high dimensional data," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 11, pp. 2227–2240, 2014.

[54] A. Bookstein, V. A. Kulyukin, and T. Raita, "Generalized hamming distance," *Information Retrieval*, vol. 5, no. 4, pp. 353–375, 2002.

[55] Y. Pei, H. Wu, J. Yu, and G. Cai, "Effective image registration based on improved harris corner detection," in *2010 International Conference on Information, Networking and Automation (ICINA)*, vol. 1. IEEE, 2010, pp. V1–93.

[56] T. Lindeberg, "Scale Invariant Feature Transform," *Scholarpedia*, vol. 7, no. 5, pp. 10 491–, 2012.

[57] M. A. Belarbi, S. Mahmoudi, G. Belalem, and S. A. Mahmoudi, "Web-based multimedia research and indexation for big data databases," in *2017 3rd International Conference of Cloud Computing Technologies and Applications (CloudTech)*. IEEE, 2017, pp. 1–7.

[58] M. Muja and D. Lowe, "FLANN-fast library for approximate nearest neighbors user manual," *Computer Science Department, University of British Columbia, Vancouver, BC, Canada*, vol. 5, 2009.

[59] V. Vijayan and P. Kp, "FLANN based matching with SIFT descriptors for drowsy features extraction," in *2019 Fifth International Conference on Image Information Processing (ICIIP)*. IEEE, 2019, pp. 600–605.