Birzeit University

Faculty of Engineering and Technology

Department of Electrical and Computer Engineering

Video Summarization by Keyframe Extraction

Prepared by:

Ihab Abdelkareem
Bader Anini
Ahmad Yahya

Supervised by:

Mr.Aziz Qaroush

A graduation project submitted to the Department of Electrical and Computer Engineering in partial fulfillment of the requirements for the degree of B.Sc. in Computer Engineering

BIRZEIT
Jan - 2020

Abstract

Video content across the internet is growing at rapids rates. Websites which host various sorts of multimedia now facilitate the ability for their users to directly upload, view and edit videos quite easily. However, with such massive growth in video content online, it has become a tedious task to concisely manage user-generated videos. Consequently, there has developed a need to efficiently manage and process consumer videos. Multiple video management schemes have been proposed throughout previous research, one of which is video summarization. Video summarization aims to automatically convert a video into a static or dynamic skim, which in turn allows for enhancements towards more efficient video browsing and accessibility.

In this project, we approach video summarization as a keyframe extraction process. The implemented video summarization system converts input videos into a collection of static images called *keyframes*, each of which highlighting an event within the video. With keyframe extraction, we are able to provide an abstract representation of the video by a temporal sequence of images. Consequently, the video can be later referred to through its corresponding keyframes in order to facilitate efficient management of video data. To test the quality of the implemented system, the widely used VSUMM dataset was considered. Results showed a correlation between our outputs and those provided through a human's perspective, with up to 70% precision and 80% recall. Initially, the video is preprocessed and broken down into a less dense form. Secondly, the video is split into a set of scenes through visual feature analysis, as each scene generally represents a single flow of events. Finally, keyframes are chosen to summarize the video by clustering each of the extracted scenes into correlated groups of frames. To further demonstrate the use of video summarization, we have implemented a keyframe-based video browsing application. Through the application, the summarization methodology is accessible to users online, as they are allowed to upload their own content, observe its summary and browse through videos. In addition, the extracted keyframes of each video are put to use, as they build an index by which videos are searched.

ملخص

في هذه الايام يزداد محتوى الفيديو المنشور على الانترنت بمعدلات فائقة. سهلت مواقع تخزين و مشاركة محتوى الوسائط المتعددة امكانية تحميل و مشاهدة و التعديل على الفيديوهات بصورة سهلة غير مسبوقة. بالنظر الى الموضوع من جانب اخر، فانه و مع التطور الهائل في حجم المحتوى المرئي على الانترنت، اصبحت ادارة ملفات الفيديو الخاصة بالمستخدمين عملية صعبة و مرهقة، هذا بالاضافة الى انه و بالنظر من زاوية المستخدم ذاته فان التعامل مع كمية الفيديوهات واسعة الانتشار و الغير مرتبة على نسق محدد لهي تجربة غير محبذة اذا ما قارناها بالحصول على معلومات الفيديو ذاته بطريقة مختصرة. كنتيجة لذلك، كان لا بد من ايجاد طريقة لادارة و معالجة الفيديوهات. العديد من المقترحات لادارة هذه الفيديوهات تم طرحها من خلال الابحاث السابقة، و كان احدها تلخيص الفيديو الذي يتمثل في محاولة استخلاص الاجزاء المفيدة و المباشرة على شكل صور ثابتة أو مقاطع مهمة، والتي بدورها تعبر عن الجزء القيم من المحتوى و تحقق الاستفادة القصوى لدى المستخدم.

نحاول في هذا التقرير ان ننظر لمسألة تلخيص الفيديو كمهمة لاختيار مجموعة من الصور الرئيسية من داخل الفيديو. نستطيع القول إن الصور الرئيسية المستخرجة من الفيديو لها القدرة على تمثيله بطريقة بسيطة و معبرة. لفحص كفاءة الطريقة التي تم تطبيقها، تم استخدام مجموعة الفيديوهات VSUMM و المستخدمة بكثرة لفحص نظم تحليل الفيديوهات. أظهرت النتائج أن هناك تقارب بين الصور التي يختارها نظامنا و بين ما اختاره المستخدمون، بنسبة دقة تصل الى ٧٠٪ ونسبة إرجاع صور من ضمن التي اختارها المستخدمون تصل الى ٨٠٪. في البداية، نقوم بمعالجة الفيديو معالجة أولية عن طريق تقليل حجمه. نقوم ثانيا بتقسيم الفيديو الى عدة مشاهد بالاعتماد على السمات الشكلية، بعدها يمكن تحليل كل مشهد على حدة. في النهاية، نقوم باختيار الصور الرئيسة لتمثيل الفيديو بعد تجميع الصور المتشابهة مع بعضها في كل مشهد في مجموعات. بعد استخلاص الصور الرئيسية، نكون قد حصلنا على تمثيل للفيديو يتبع التسلسل الزمني للاحداث. كنتيجة لذلك، يعبر عن كل فيديو باستخدام الصور الرئيسية مما يتيح إدارة أفضل لملفات الفيديو. لتوضيح الفائدة من تلخيص الفيديو، قمنا بعمل تطبيق لتصفح الفيديوهات مبني على الصور الرئيسية المستخدمة. من خلال هذا التطبيق، يمكن للمستخدمين استخدام نظام تلخيص الفيديو، بحيث يمكنهم مشاركة فيديوهاتهم الخاصة و مشاهدة تلاخيصها هي و كل الفيديوهات الاخرى التي تم رفعها من مستخدمين اخرين. كما و تم توظيف الصور الرئيسية لتسهيل فهم الفيديو و بالتالي تقديم خدمة بحث نصي عن الفيديوهات.

Contents

List of Figures

List of Tables

Acronyms

API Application Programming Interfrace.

BOVW Bag of Visual Words.

CDN Content Delivery Network.

CNN Convolutional Neural Network.

CUS Comparison of User Summaries.

DOG Difference of Gaussian.

DT Delaunay Triangulation.

FPS Frames per Second.

HSV Hue Saturation Value.

IP Internet Protocol.

JSON JavaScript Object Notation.

NLP Natural Language Processing.

OS Operating System.

OVP Open Video Project.

PCA Principal Component Analysis.

REST Representational State Transfer.

SIFT Scale-Invariant Feature Transform.

STIMO STIll and MOving Video Storyboard.

UUID Universally Unique Identifier.

VSUMM Video SUMMarization.

Chapter 1

Introduction

1.1 Motivation and Overview

Nowadays, video capturing and generating technology is accessible to the public and at pure ease. Henceforth, large amounts of video data are being generated while disregarding effective handling of such content. For example, the YouTube video platform contains up to 50 million users creating content, with 500 hours of footage being uploaded by the minute[1]. In response to the dramatic increase of videos lacking metadata and human supervision, research has pivoted towards finding solutions to mitigate such burden. Therefore, recent research has aimed to efficiently find solutions to store, index and distribute video content effectively. As part of the schemes introduced to help manage video data, video summarization is proposed.

Video summarization is the process of synthesizing an arbitrary video into an abstract, shorter and understandable representation (Pfeiffer et al.[2]). Therefore, the term *video summary*, which we use frequently throughout this report, simply refers to the abstract and concise representation of the video. Video summaries can generally be split into two main categories, namely dynamic and static summaries(Truongand and Venkatesh[3]). With dynamic summaries, the video is converted into a group of abrupt video skims in the form of short video segments. Dynamic summaries, or *video skims*, present an enjoyable form of a video summary, as motion and audible cues are preserved. On the other hand, static summaries approach the summarization problem in the sense of converting the video into a slideshow of static images. Specifically speaking, static summaries are formed from a set of images representing key events called *keyframes*, which also convey the video's temporal evolution. Static summaries are less entertaining than their dynamic counterparts as they are limited to still images. However, they introduce a more practical approach when extending video summarization to an applied video management and indexing solution. Figure 1.1 illustrates the fundamental difference between dynamic and static video summaries for an input video.

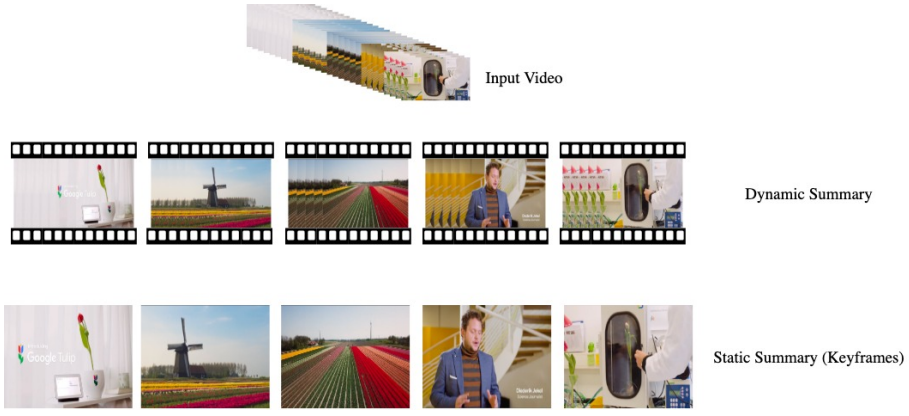

Figure 1.1: Dynamic and static video summaries

The outputs of video summarization can be put to use to satisfy the needs of many applications. For instance, dynamic video summaries can be used to facilitate the automatic generation of video trailers and skims. Likewise, video hosting platforms are able to take advantage of previewing an abstract dynamic skim for users to see before clicking on a certain video. On the other hand, static video summaries, i.e. keyframes, can be used in video storage oriented applications. Video search and retrieval are the main example of such applications, as keyframes can be used as metadata to efficiently form an index through which videos are retrieved. In addition, since keyframes generally highlight the main parts of a video, a keyframe summary is able to give insight into the main entities in the video and its general category.

Despite the benefits of video summarization and the capabilities it allows, many challenges are faced when approaching such idea. Generally, it is common that the concept of a video summary may be considered as vague, since it differs from the perspective of one person to another. Therefore, developing an efficient summarization system which fits the human's subjective perceptions is challenging. Additionally, video summarization faces challenges in choosing the proper methods in representing the video through its visual, audible and temporal elements, to later be used in extracting a representative video summary. With a variety of video types and structures, finding a general scheme to convert the video's visual content into a true summary remains among the most challenging technical problems.

In this project, we are motivated to approach video summarization by means of keyframe extraction. Not only do keyframes introduce a concise and direct video abstraction, but they can also be used to further understand and represent the video. Hence, we propose keyframe extraction methodology aimed at simplifying the video into a timeline of frames which are both relatable to a human and bring benefit to video management as well. Our methodology is then put to action through implementing video browsing and indexing by means of extracted keyframes. By doing so, we believe that video summarization serves as a building block in the task of automatically dealing with highly-scaling video content in an online video hosting platform. By automatically dedicating a static summary to each video, the video can be later found and retrieved. To provide a high-level overview of our contribution, Figure 1.2 illustrates the implemented system. The system applies our keyframe extraction methodology to an input video, and eventually uses the keyframes to represent the video within the storage to later be searched and retrieved.

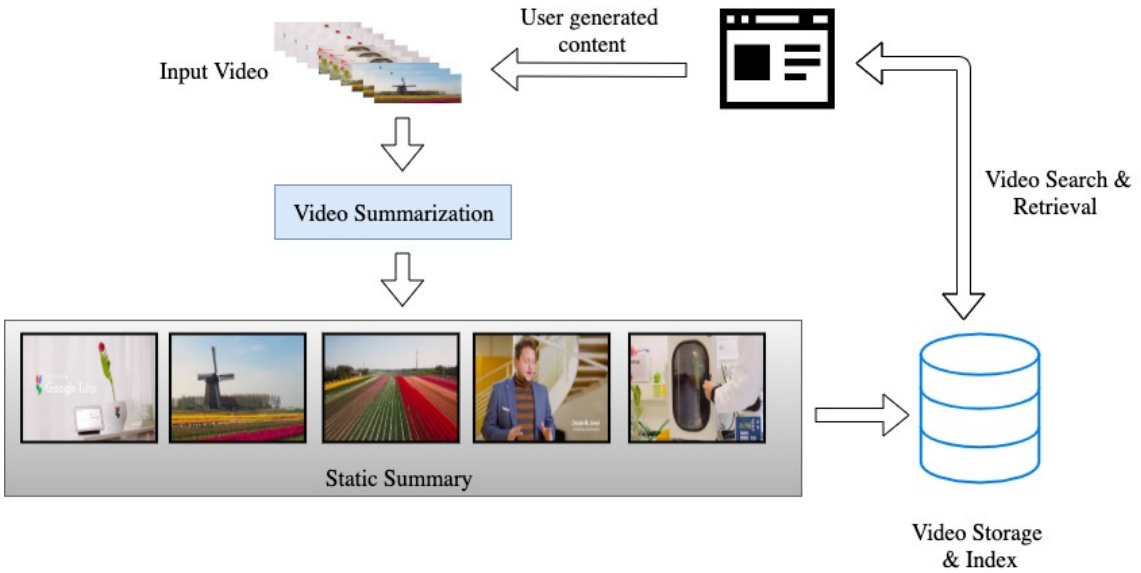

Figure 1.2: Video summarization system overview

1.2 Problem Statement

The fact that video processing tasks are complex in their nature, raises a need for efficient video understanding solutions. Hence, our focus in this project is oriented towards finding an efficient video representation methodology through video summarization and investigating what can be achieved from it.

In order to overcome the challenges that appear due to performing computationally-expensive video processing tasks, a static video summarization system will be developed. To do so, our task is to develop a system that converts a general video into a representative set of keyframes containing both the main events and temporal sequence of the video. In addition, to satisfy the requirements of extending video summarization to an applicable system, the generated keyframes from the videos need to be put into action through a video-based application. Therefore, the application of choice to demonstrate the benefits of keyframe extraction will focus on performing video representation and search within a digital library.

1.3 Contributions

- We built a static video summarization system based on local and global visual features, aiming to both highlight a video's main events and preserve its temporal sequence.
- We implemented a keyframe-based video storage and browsing application which makes our system accessible to users online, and allows for video search based on keyframes.
- We propose an approach for the evaluation of static video summarization, based on multiple types of image descriptors to guarantee precise image matching.
- Upon evaluating our work on the VSUMM dataset and comparing to previous works, we propose a more objective golden-standard summary formed from multiple user-generated summaries for the reference videos.

1.4 Report Outline

Chapter 2: Discusses the previous works on video summarization, with the common techniques and methodologies used.

Chapter 3: Elaborates on our proposed system for video summarization by keyframe extraction, including its specific specific stages and features used.

Chapter 4: Introduces our web-based application for putting keyframe extraction into action by implementing a video browsing digital library.

Chapter 5: Discusses our keyframe extraction evaluation methodology, along with the dataset considered, metrics used, obtained results and comparison to related works.

Chapter 6: Concludes our work with final remarks, presents its main limitations, and discusses our future work.

Chapter 2

Related Work

In this chapter, we will dive into the details of video summarization systems as seen in the literature. Video summarization systems have various input/output specifications depending on the internal processing requirements and on the application needs. In the case of unsupervised systems, every system must have the video as a least input requirement. However, many approaches that depend on supervised learning, which in most cases, need an external source of information that is fed along with the video as a system input. This type of information (referred to as "annotations" in literature) can be split into an external text script and metadata. The metadata can be the title, category or any other useful information that helps to select the properly supervised machine learning model or even to perform an unsupervised task more accurately.

There are two main output types of video summaries, namely the static video summary and the dynamic video summary. The static one is represented by keyframes that are selected as the most expressive frames, they are also referred to as 'storyboard' or 'still-images' in the literature. The dynamic summary consists of temporal segments that express the video the most, and they can create what's called 'video-skims' or 'moving-storyboard'. Figure 2.1 shows the two main approaches of summarization systems (supervised and unsupervised) along with their possible input/output.

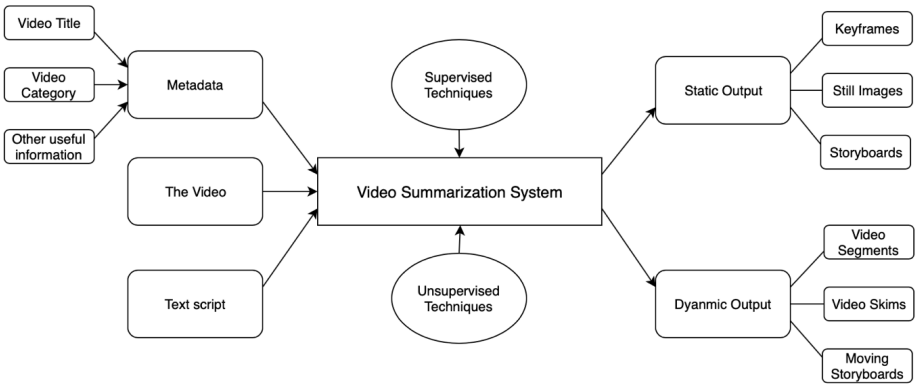

Figure 2.1: General view of video summarization systems

Regardless of the output type, video summarization systems are used for both closed domain purposes and for general uses. The choice of which domain to target depends on the application needs and data available. However, choosing a general-purpose video summarizer can build a core stage to perform other computational-expensive tasks that aren't tied to a specific class. Additionally, general purpose summarization systems are beneficial for massive amounts of user-generated videos coming from various domains and having diverse specifications.

After discussing the difference between supervised and unsupervised systems, and the difference between static and dynamic output, we focus on showing differences in unsupervised static output

approaches. It can be noticed from previous literature that such approaches modeled the problem differently. Some approaches dealt with it as a search problem where the interest is in finding keyframes among the whole set of frames. Search techniques like greedy, dynamic programming and optimization were the main focus in such methodologies.

The features used to describe frames or frame sequences also differ widely. We can split features into global and local ones. Global features are used to describe the image as a whole unit. Global features can be further divided into features based on color like color histogram, color moments, color coherence and color layout. Additionally, they can also be based on motion detected in consecutive frames like global motion features. Local features, on the other hand, are used to describe the internal semantics the image and is thus better to capture edges, keypoints and objects rather than the general description of the image. Figure 2.2 shows how approaches differ according to the discussed hierarchy.

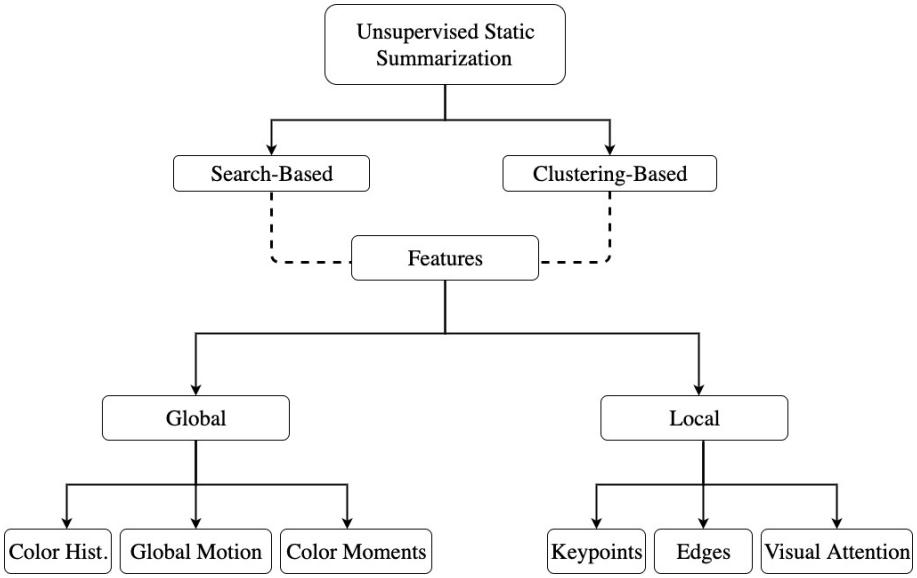

Figure 2.2: General methods and features used in summarization systems

In order to provide clear and formalized insight into the literature, the methodologies discussed in this chapter shall be divided into accordance with common unifying general frameworks.

2.1 Unsupervised Video Summarization

In this section, we present the main unsupervised methodologies and frameworks in static video summarization. Unsupervised methods aim to leverage the semantics of local and global visual features in the process of key-frame extraction, namely the visual concepts present within each frame. Prior knowledge or manual annotation of the input frame sequence is not required in such methodologies. The features used in such methods are many variants of color-based, key-point, and optical-flow features. Additionally, unsupervised methods can be categorized into two main approaches, which are search-based and clustering-based.

2.1.1 Search-Based Approaches

Search-based techniques hold a main role in the literature, since the video-summarization problem itself can be viewed as a search problem. Therefore, some approaches formalize it entirely as a search-based problem. In this section, the pure search approaches are addressed and discussed, each one with its advantages and disadvantages.

Search-based static video summarization methodologies mainly obey a similar workflow to a certain extent. The search goal can be to find the critical changes between the frames, finding

local-extrema points among them, or looking for a pre-defined pattern that can maximize or minimize some criterion. Search-based methods start by prepossessing the video to reduce the workload by sampling the video or by changing its resolution or both. The next step is to do frame-measure calculations in which features are used to describe frames. After that, a local or global search starts. Local search depends on greedy algorithms to find local frames that satisfy some criteria and thus are faster but not optimal. Global searches on the other hand optimize for the whole video. After the search is done, approaches are left with candidate keyframes. At this point, the last stage which is of keyframe selection is applied. The four general stages of search-based techniques are illustrated in Figure 2.3.

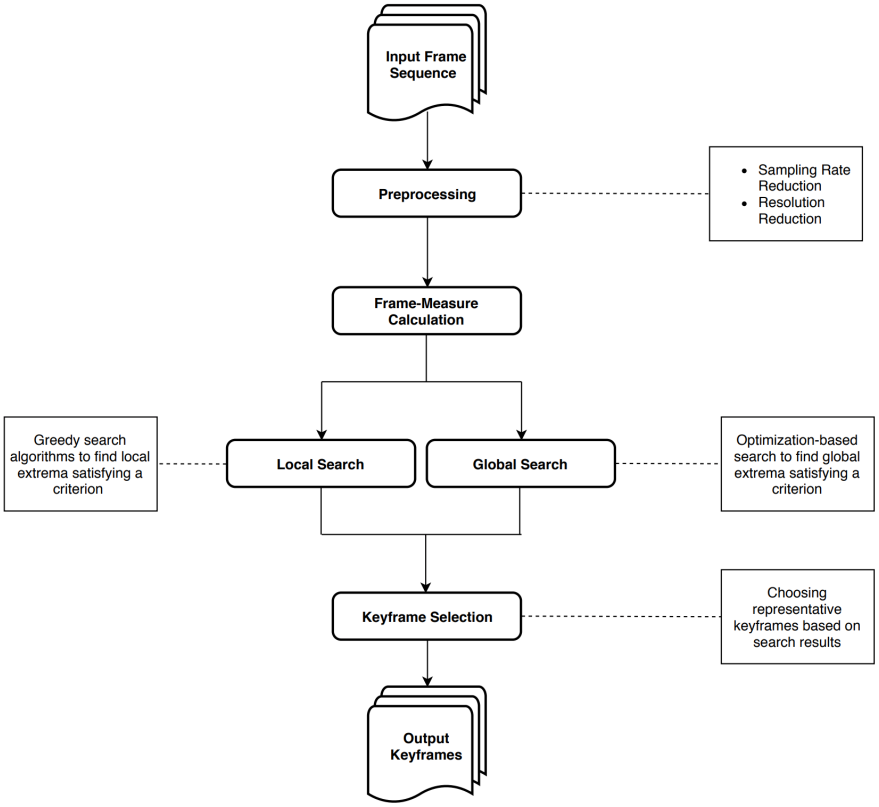

Figure 2.3: Search-based video summarization general flow

In Wolf[4], the authors presented a keyframe selection approach based on the 'stillness' of either the camera or the actor movement. They extract the optical flow features for every frame in the video according to the proposed work of Horn and Schunck[5]. For each frame, they computed a metric called $M(t)$ that represents the optical flow magnitude of the frame. After that, they used a local search method to find the local minima. Therefore, they defined two pointers, $m1$ and $m2$, where $m1$ points to an earlier frame, such that the value of $M(t)$ at $m1$ varies at least of some pre-defined percentage of the value of $M(t)$ at $m2$. The keyframe is the local minima in the mentioned range between $m1$ and $m2$. The search continues for frames after the frame at position $m2$ in order to find other keyframes. The approach was near-realtime, also, the number of keyframes is not fixed and a shot can contain multiple keyframes. However, their method was not proved or evaluated by any metric or on any dataset as it was only tested on two movies. In addition, a variation percentage threshold is needed for their local searching.

In Mendi et al.[6], the authors developed a system for video summarization based on global features and two optical flow algorithms. The video is initially segmented into shots using the color histogram of successive frames. Next, they applied a three-phase method: pre-processing, applying a motion function and the keyframe extraction. For pre-processing, they scaled each

frame down and then converted it to grayscale for lighter computational cost, as the shape of the motion remains the same. The next step was to apply local and global optical flow algorithms. The proposed optical-flow methodology of Lucas, Kanade, et al.[7], which is a local algorithm, and the global method of Horn and Schunck[5], each were used and ended with a value that represents a frame. The last step was the keyframe selection, in this step they tried several methods. First, searching for the local extrema points in each shot (local minimum and maximum) and marking them as keyframes. Another method was to find the maximum two consecutive points in the search to infer the keyframes. Finally, they propose that for each maximum point they search for, the next point that varies at least times N times is marked as a keyframe. The experiments in this approach focused on 5 variations of the above methods, each one using one optical flow algorithm or just selecting the middle frame of the shots. They created a user interface to let the users evaluate each method as ('good', 'reasonable' or 'bad') on their own collection. Most of the methods were parameter-free, and the time was reasonably good after the pre-processing step. However, evaluation wasn't applied on a dataset, and no method got 50% or higher of the good rate from users voting.

Guan et al.[8] was the first methodology that used local features to extract keyframes, their approach didn't divide the video into shots, but relied completely on the keypoints. The keypoints extractor and descriptor used was Scale-Invariant Feature Transform (SIFT), and they focused on two objectives, maximizing the coverage of the keyframes, and minimizing their redundancy. They first constructed a universal keypoints pool by scanning the whole video frame by frame and then searching for keypoint chains by finding the matching frames within an adaptive window W. After the keypoint chains were constructed, an aggregation step is needed to create the universal chain, which will later present the keypoints pool. Then, the keypoints pool will was split into covered and uncovered groups. When a keypoint is chosen, it will go to the covered group to measure and minimize the redundancy. Initially, all the keypoints in the pool are in the uncovered group. The coverage of a frame $C(f_i)$ is the intersection of its keypoints with the uncovered group. Meanwhile, the redundancy $R(f_i)$ is the intersection of the frame's keypoints with the covered group. Finally, the influence of a frame was expressed as $Influence(f_i) = C(f_i) - \alpha R(f_i)$.

The approach proposed by Guan et al.[8] was robust and was tested among different case studies on the Open Video Project (OVP) video repository. Its results outperformed the ones that depended only on the global feature. However, its computational complexity was an issue, since it depends heavily on the extraction and matching of the local features. Another main limitation is the number of parameters needed. The approach needs a threshold for the radius of matching the keypoints, an initial value of the threshold W, and α for measuring the influence. Many testing steps needed tuning for the previously mentioned thresholds.

Liu et al.[9] formalized the problem as maximizing a pattern. First, they created a graph by representing each frame as a node and from each node there's an outgoing edge to the next temporal nodes (frames). Hence, they ended up with a directed acyclic graph. Then, they defined the weight of an edge as the visual difference between the two nodes. In their case, such weight was a color histogram difference between the two frames. The pattern they defined was finding the maximum path of a certain length. Consequently,the frames selected among the path will have the maximum diversity. However, this problem is an NP-hard problem of complexity $O(nCk) = O(n!)$. To resolve this, they proposed an optimal dynamic programming solution of complexity $O(n^3)$. Then for further improvements, they proposed a greedy algorithm with optimizations, and they showed how the greedy algorithm had close results to the optimal dynamic programming one with $O(nlog(n))$ complexity. Finally, they proposed a hybrid model that combines both the dynamic and greedy solutions with results more closely to the dynamic programming solution, with a complexity $O(n^2)$. The greedy and hybrid algorithms were fast and on-the-fly, and the proposed solutions outperformed the previous techniques. However, changing the value of k isn't possible for the greedy and hybrid algorithms, so the number of keyframes were strictly parameterized.

Furini et al.[10] used the k-center problem to solve the video summarization problem, as selecting k keyframes so that the maximum distance between a keyframe and an original frame is minimized. However, the k-center problem is NP-hard problem, so they performed greedy farthest point first solution with customizable optimizations. In their tool, the users can specify the time they are

willing to wait, and the number of keyframes. Hence, the results depend on the time the users specify. Even with the greedy solution in their approach, the computational cost still is high and the number of keyframes was parameterized.

Lastly, the early approach proposed by DeMenthon et al.[11] represents the video as a curve approximation by a chain of line segments. To select feature vectors in which frames are represented, feature vectors of 37 dimensions containing both time and pixel-luminance information were considered. Finally, keyframes are chosen by noticing large variations in the curve representing the video's frames.

All the previously mentioned work have drawbacks. In finding the local-extrema points among the frames, a lot of semantic information are lost because only some peaks are considered. The advanced search algorithms were also NP-hard problems, so their computational cost was dramatically high. Even with dynamic programming solutions, the resulting polynomial complexity is also quite high. Finally, for most of the previously search-based approached, the number of keyframes was predetermined as well for the thresholds used.

2.1.2 Clustering-Based Approaches

A good number of clustering algorithms were applied in detecting shots and scenes inside the videos as a step before keyframe selection. Popular algorithms like k-means, x-means, c-means, Delaunay clustering, Hierarchical clustering and much more proved their ability to cut the videos into smaller units according to changes in frame features. Clustering is the common unsupervised approach in the literature to segment videos for its ease of use and the great results it achieves. Some algorithms need predetermined parameters to function such as the number of the resulting clusters or some kind of thresholds while others don't and are fully automated. In this section, we will discuss the popular cluster-based approaches in the literature. Nevertheless, the majority of clustering-based approaches comply with a unified workflow in the keyframe extraction process as illustrated in Figure 2.4.

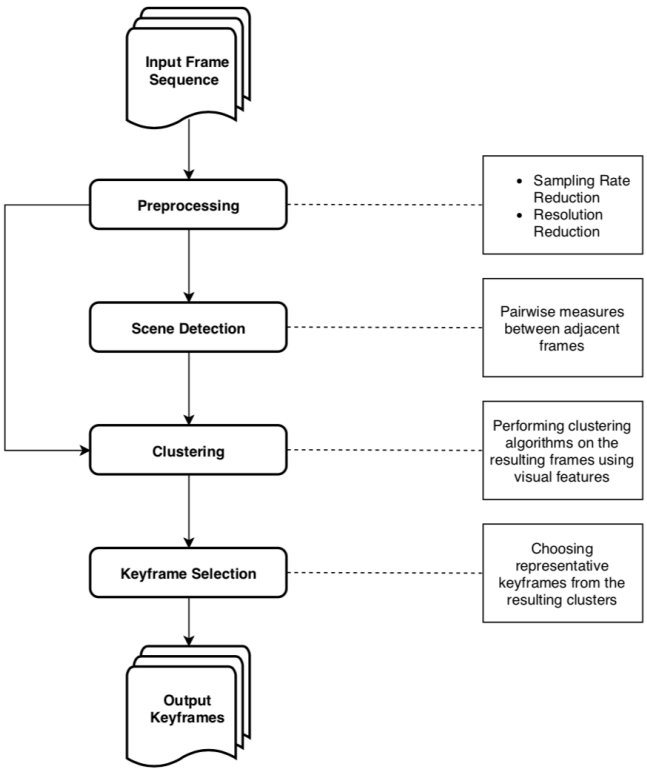

Figure 2.4: Clustering-based video summarization general flow

Multiple previous works used color histogram feature to describe frames. De Avila et al.[12] used k-means clustering to segment videos after sampling it the input at 1 fps and removing meaningless and redundant frames.The number of clusters, k, was determined by thresholding according to the pairwise distance of adjacent frames. Frames closest to cluster centroids are sorted in temporal order and selected to be keyframes. This approach was efficient as it was automated and fast. However, directly clustering the video mainly causes a loss in the temporal significance of the resulting keyframes.

In Mundur et al.[13], Delaunay clustering is used on frames in order to extract length variable clusters. A centroid of each cluster is then selected as a keyframe. Frames are represented as a vector in an m-dimensional space initially where m was set to 256 (color histogram). But because this vector is sparse, Principal Component Analysis (PCA) is used to reduce its size to d such that the chosen principle components are the components that cause 90% of the variation, $d = 7$ was found to be sufficient. Delaunay triangulation is used to draw edges between points such that only rectangles are formed with its circumcircle not containing other points inside it. For each point, local mean length (mean of all lengths of point edges) and local deviation (standard deviation of the lengths of the point edges) are calculated. Then, the global deviation, which is the mean of all of the local deviations was found. If the edge length is greater than the global standard deviation plus local mean length, then it was considered to be an inter-cluster edge and was removed. On the other hand, if its length is less than local mean length minus the global standard deviation then it is an intra-cluster edge and it is retained. This method is fast as the triangulation can be done in $O(nlogn)$ and the identification of inter/intra cluster edges can be done in $O(n)$, and it is fully automated. It also does a good job in compression as it selects fewer keyframes in general than test dataset (OVP dataset) yet achieves high overlap percentage and performs better than k-means clustering. However, it only makes use of one feature which is the color histogram. More features could be used to achieve better results such as text, audio, or motion features (if they are available).

Panda et al.[14] constructs a complete weighted graph where vertices are all frames and edge weights are histogram intersection distances. This graph is then reduced to a stilton graph based on a threshold to remove less-valued edges and vertices according to video length. Mimimum spanning tree is applied to the graph such that an edge between two vertices with weight more than the average weight of edges for the first or second vertex plus their standard deviation is removed. After that, random walker algorithm is applied to cluster the graph, then k clusters are found from it. This method was time costly and required a predetermined value of k.

Asadi et al.[15] and Cayllahua-Cahuina et al.[16] use c-means to cluster frames. But Asadi et al.[15] samples the video with 1 fps to reduce size and redundancy first. They both tackled the problem of needing cluster count by detecting content change according to the pairwise distance of every frame and its adjacent one in Asadi et al.[15] and by using fuzzy-art algorithm in Cayllahua-Cahuina et al.[16]. The frame with the highest membership inside each cluster is selected to be a keyframe. After that redundant keyframes are removed according to Euclidian distance in Asadi et al.[15].

A yet more intriguing methodology which uses a clustering algorithm is that of Chu et al.[17]. The authors are motivated by the fact that videos of the same topics share similar video contents, hence they propose a video co-summarization technique, based on summarizing individual videos using the visual content they share with other videos of the same topic. Hence, they model such similarity by computing co-occurrence matrices between the input frame sequence, and the retrieved video using normalized local SIFT features, along with color histograms. Using the co-occurrence matrices, bipartite graph is then used to model the similarity between the input video and the set of retrieved videos. Next, k-means is used to partition the graph into co-clusters of highly commonly occurrent visual content between shot pairs. Co-clusters which resemble high correlation are only chosen, by ruling out co-cluster which correspond to high dominance of a single video. The method presented by Chu et al.[17] stands out compared to other unsupervised techniques as it takes advantage of the co-occurrence of visual concepts amongst videos of the same topic, hence generating more efficient summaries. However, its weaknesses were the time-consuming graph construction. Also, the complexity of considering processing among a set of

videos in order to generate a summary for a single instance only was costly, in addition to the complexity of finding maximum correlating co-clusters. Moreover, the static parameter of the number of clusters is also an issue (predefined k).

Color histogram clustering approaches show fast performance and minimal parametrization. However, using the color feature alone does not allow precise frame description, since the global feature lacks to describe the semantic content within a given frame, and is prone to high rates of redundancy within a specific scene within the frame sequence. Bag of Visual Words (BOVW) feature vectors, which are analogous to bag of words in Natural Language Processing (NLP), are widely used in computer vision applications. The vector itself is constructed of a histogram describing the frequency of features among a known amount of codewords(codebook). Bag of visual words is adopted in several video summarization methodologies.

In Ainasoja et al.[18], the input video is divided into one second long windows, SIFT features are then used in the c-means algorithm to generate a BOVW codebook for each window. The scene boundaries are then detected based on existing a significant difference between adjacent Bow Histograms. Additionally, the authors introduce dynamic generation for the BOVW codebook, where it is re-generated as scenes are being detected, which introduce a form of dynamicity in the computation of the feature vectors, as they became relevant to the specific scene. The keyframes are finally chosen based on the resulting clusters. Several approaches are implemented in selecting keyframes from the resulting clusters, such as the middle frame or the frame with the most/least similarity to the others.

The previous approach stood out with its dynamic codebook generation, along with its relatively fast execution time. However, it lacks complete automation as human intervention is required, along with a manual parameter tuning. Meanwhile, Guan et al.[19] proposes a methodology where similar to Ainasoja et al.[18] in terms of the codebook construction via SIFT features, but different in the sense where following the application of X-means clustering. Each frame's BOVW vector is compared with the uncovered key-points of the scene in order to find the frame's coverage, and conversely, the vector is compared with the scenes covered key-points to find its redundancy. Despite its distinct approach, the previous methodology is flawed due to the requirement of cluster-related parameters.

Similar to the approach before, Cahuina et al.[20] also utilizes the X-means clustering algorithm, where a BOVW vector calculated from local features -such as Difference of Gaussian (DOG) and SIFT variants-is used to generate the codebook. The X-means parameters were set to static values which assure that each scene within the video results in 5 to 10 key-frames. The approach also addresses the issue of frame dissolving, which is the phenomena of frames emerging upon smooth transition from one scene to another.

Generally, clustering methods which employ BOVW aim to utilize the key-point based features in order to feature vectors used in clustering algorithms, namely codeword frequency histograms. Their use of local features allows for robust techniques in semantic representation of each frame, along with scene detection algorithms. However, their limitation lies within the need for explicit codebook parameters, along with specifying the local feature descriptor of use, whose efficiency may vary from one frame to another. Another limitation here is the high time complexity that results from finding the global BOVW and matching global with local BOVW.

2.1.3 Unsupervised Video Summarization Drawbacks and Limitations

Despite the ease of application of unsupervised approaches and the good time complexity they show in general (especially when using clustering), a number of its noticeable weaknesses cannot be ignored. For a good number of clustering approaches, the desired number of clusters must be determined previously, making the approach less automated. For search-based approaches which traverse the frames looking for optimal satisfaction for one or more metrics, it is usually costly regarding the time spent to process all frames. In both approaches, the selection of keyframes is actually a blind work in the sense that it tries to satisfy objectives based on general frame-level visual features which do not necessarily result in representative key-frames. Given these limitations,

unsupervised approaches are also not capable of summarizing domain-specific videos since it cannot take into advantage any type of domain-specific elements, such as object detection/tracking which give specific insight into the video. Compared to supervised approaches, unsupervised methods do not make use of user-generated summaries. Therefore, they cannot learn and improve their accuracy to generate summaries that are as close as possible to what people try to do when they create their own summary.

Chapter 3

Keyframe Extraction Methodology

In this chapter, we discuss our methodology to achieve video summarization by keyframe extraction. Throughout our proposed methodology, we introduce the stages in which the system swiftly converts an input video into a set of keyframes. By summarizing a video into keyframes, we dramatically obtain a shrunk video representation in order to enhance the understanding, browsing and indexing of video content.

3.1 Methodology Overview

In order to generate an efficient summary, the video summarization system was designed to mimic a human's general approach in choosing a set of keyframes based on their visual attention. Therefore, we've divided our methodology in tackling the summarization problem into a sequence of complementary stages. Each of the implemented stages is devoted to partially carrying the input video towards its final keyframe-based summary. The raw input video is first broken down into a more processing compatible version through preprocessing. To semantically represent the video, both color and deep learning based features are extracted. Afterwards, the video's scenes are detected in order to break down video summarization, which in turn orients keyframe extraction to focus on each event separately. Finally, each scene is clustered and split into highly correlated groups, from which keyframes are chosen. Figure 3.1 shows a high level overview of the proposed system as it converts a user's video to a timeline of keyframes. Through this methodology, the temporal sequence of events was preserved and conveyed through the extracted keyframes.

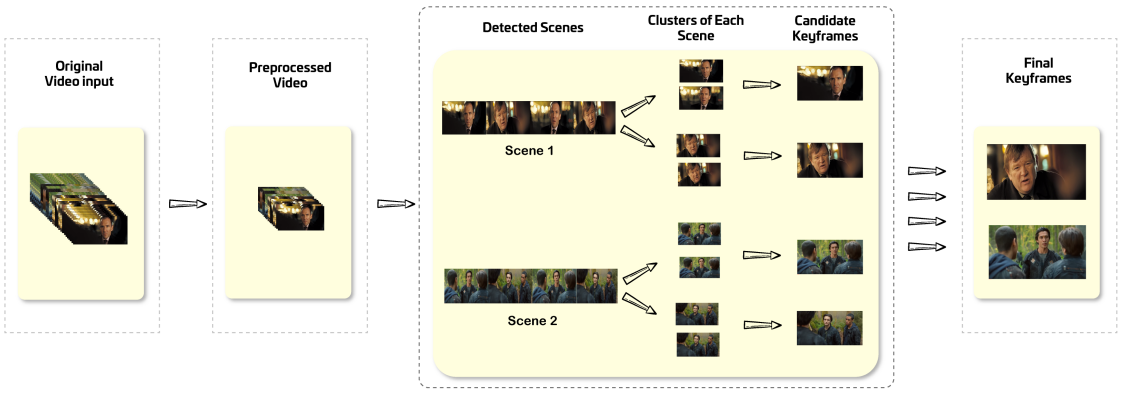

Figure 3.1: Proposed keyframe extraction approach

3.2 Video Preprocessing

In our methodology, video preprocessing is a preparatory yet critical stage. Due to the variability in real-world videos found online, many issues could arise due to inconsistency in video characteristics. Consequently, we propose preprocessing as a stage to transform raw videos into ones which are compatible with our propose summarization technique. By doing so, the processing time of our methodology is significantly improved while preserving the raw video's visual content. In this stage, the raw video undergoes two main processes, namely uniform sampling and resizing.

3.2.1 Uniform Sampling

Frame-rates of videos across the web generally vary due to several factors which trace back to the video's origin. Nevertheless, the average video found online through sources such as YouTube or social-media platforms tends to fall in a wide range between 24 to 60 Frames per Second (FPS)[21]. While a high frame-rate is a key element in a video's quality and enjoyability, it results in a high level of visual redundancy from a computational perspective. Hence, due to the large presence of redundancy in the frames of a raw video, a part of the initial stage for video summarization was to scale down the amount of frames to be processed. The sampling frame-rates were experimentally chosen in a way which significantly reduces processing time while preserving the video's visual content. Frame-rates between 3 and 5 were observed and showed the most satisfactory results through experimentation in achieving such task. Figure 3.2 shows how a 25 FPS video was sampled down to 5 FPS.

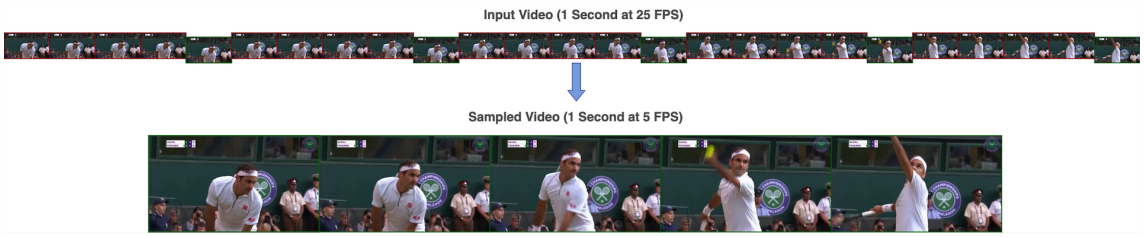

Figure 3.2: Sampling a video at 5 FPS

3.2.2 Frame Resizing

Similar to high frame-rates, large frame sizes are also a burden. In order to remedy the problem of varying frame sizes, a reference frame size was enforced. Videos that were fed into the summarization system were eventually resized to guarantee the efficiency of processing while maintaining to preserve visual content. Choosing a suitable frame size to use throughout the system's stages is not a clear-cut task. Nevertheless, referring to the feature extraction used in the methodology, discussed in 3.3, we find that frames are implicitly resized to 299x299 in the case of Convolutional Neural Network (CNN) feature vectors. To assure that this size is appropriate, it is checked to also match its original sized counterpart through the Hue Saturation Value (HSV) color histogram and SIFT[22] keypoints. Figure 3.3 shows the keypoint matching between resized image and its original version, while their HSV color histogram match reached 99%.

Figure 3.3: Frame equivalence after Resizing (299x299 vs 1280x270)

3.3 Scene Detection

Scene detection is a key process in our proposed summarization system. By dividing the pre-processed video into multiple temporally and semantically correlated segments, extracting keyframes then becomes intuitively focused on each scene separately. Additionally, this in turn helps video coverage, as each scene introduces an elementary event occurring within the video. In this sense, splitting a video into scenes can be considered as the first major step towards converting it into a timeline of keyframes. At a high level, scene detection is performed by first selecting the appropriate feature vectors in which to represent the video's frames. Afterwards, the differences between feature vectors of adjacent frames are calculated. Finally, using the calculated pairwise differences, continuous segments are chosen as scenes based on a threshold-based technique. Figure 3.4 shows the high-level flow of the proposed scene detection technique.

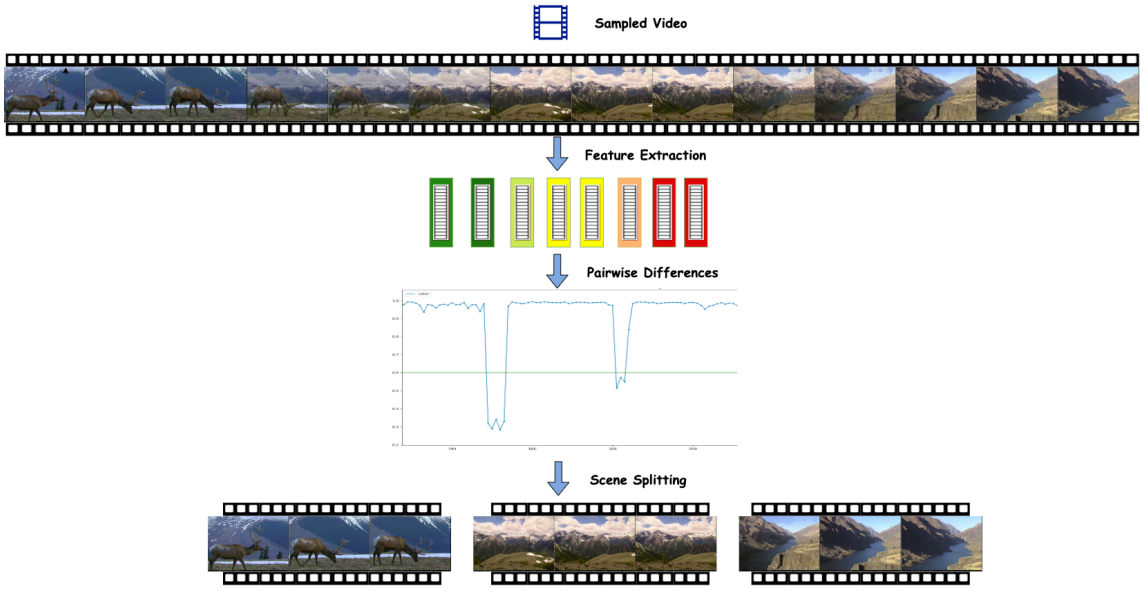

Figure 3.4: High-level flow of scene detection

3.3.1 Feature Extraction

In order to visually represent the many frames of an any input video, we extract both global and local features. Consequently, throughout our discussion of the multiple stages of system implementation, the term *feature vector* is frequently mentioned. Mainly three types of widely used features are extracted to perform image representation, namely HSV color histograms, feature vectors extracted from CNN and BOVW models. First, the HSV color histogram is used as a global feature vector, as each of the 3 HSV channels are discretized to 16 bins. Secondly, to take advantage of the local features and keypoints of each frame , the bag of visual words model was employed. In BOVW, a code-book of 50 entries is created from clustering all SIFT[22] keypoints among the whole video. Finally, we leverage the concept of transfer-learning for feature extraction. Transfer-learning aims to utilize pre-trained neural networks for later use in other applications. In our case, we extract meaningful feature vectors from the final convolution of the Inception-V3[23] CNN, pre-trained on the ImageNet[24] dataset (14 Million images for more than 20000 classes).

Feature Vector	Speed	Captured Features	Similarity-Metric	Dimensions
Color Histogram	Fastest	Color distribution	Correlation	4096(16x16x16)
BOVW	Fast	Local Keypoints and Edges	Cosine-Similarity	50-100 clusters
Pre-trained CNN	Slow	Objects	Cosine-Similarity	2048

Table 3.1: Feature-vector comparison

In order to benefit from the previously mentioned feature extraction techniques, we define image similarity measures. Consequently, the difference or similarity between two arbitrary images based on the feature vector of choice can be calculated. To compare two HSV color histograms, the correlation metric is used. Assuming H_1, H_2 to be the 4096 dimensional color histograms for two arbitrary images, their correlation is given as:

$$Corr_{1,2} = \frac{\sum_{i=1}^{N}(H_1(i) - \bar{H_1})(H_2(i) - \bar{H_2})}{\sqrt{\sum_{i=1}^{N}(H_1(i) - \bar{H_1})^2 \sum_{i=1}^{N}(H_2(i) - \bar{H_2})^2}} \tag{3.1}$$

where N is the number of bins and $\bar{H_k} = \frac{\sum_{i=1}^{N} H_k(i)}{N}$. As for CNN and BOVW feature vector similarities, the cosine similarity between two vectors V_1,V_2 representing two arbitrary images is given as:

$$cos(\boldsymbol{V}_1, \boldsymbol{V}_2) = \frac{\boldsymbol{V}_1 \cdot \boldsymbol{V}_2}{||\boldsymbol{V}_1|| \cdot ||\boldsymbol{V}_2||} \tag{3.2}$$

In order to proceed with scene detection, a certain feature vector of the discussed candidates needed to be chosen. Scene detection relies mainly on the pairwise differences between adjacent frame feature vectors. Hence, the choice of the feature vector was influenced by the specific behavior of adjacent frame differences. To obtain insight on such behavior, plots of adjacent differences were made using the mentioned feature descriptors for the same video.

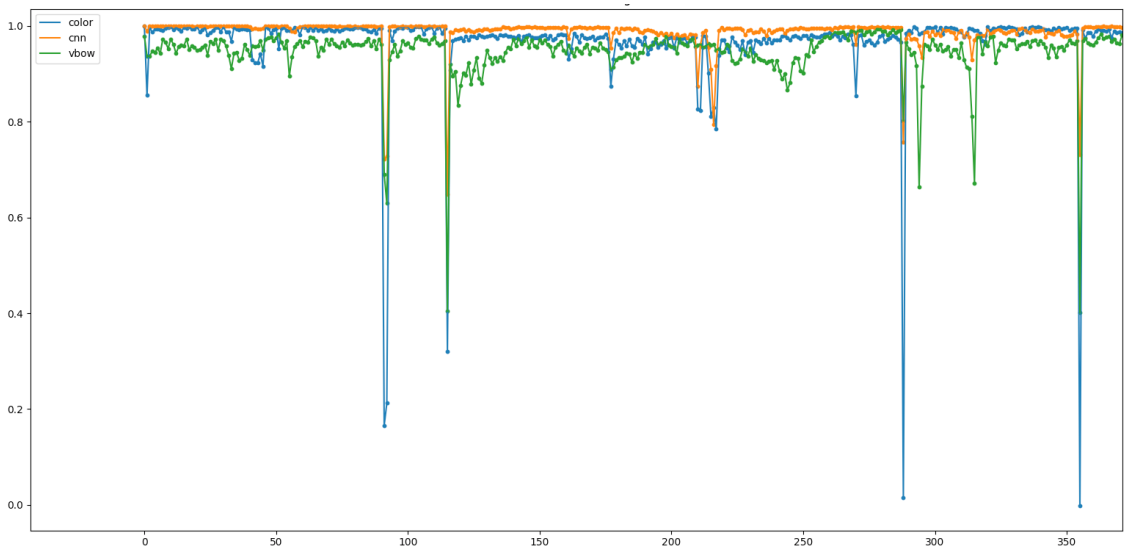

Figure 3.5: Multiple frame difference plots using different feature-vectors

Figure 3.5 shows how the different feature vectors behave when calculating the differences of adjacent frames in the video. The x-axis denotes the frame position i within the sequence of the video, while the y-axis shows the difference introduced by frame i and its preceding frame (at position $i-1$) . From these plots, We're able to reach a justification for the suitable image descriptors fit for scene detection. It is noticed that all three plots have a similar behavior to some extent. However, starting with the plot for BOVW(green), we notice that the frame differences within the same scene are far less stable than those in the remaining plots. As mentioned previously, a BOVW vector contains the frequency of each cluster of SIFT[22] keypoints within a certain frame. Consequently, this leaves much room for error as a frame may inaccurately be assigned SIFT[22] keypoints it does not contain upon determining the nearest cluster. As for the differences of vectors extracted from the CNN(orange), high stability within the same scene is shown. A specific scene is generally concerned with a set of objects or entities. Hence, the feature vector of a pretrained CNN describing the existence of a scene and its objects tends to be stable despite the overall frame

differences within the scene. However, the setback of using CNN feature vectors in scene detection was the fact that scene transitions are not nearly as evident as they are in color histograms(blue). Therefore, color histograms were mainly used, alongside CNN feature vectors, in the process of scene detection as they showed steep scene transitions along with noticeable internal scene stability.

3.3.2 Adjacent Frame Difference Computation

After defining the feature vectors, scene detection requires analyzing the difference of adjacent frames within the video's stream. It is expected that scenes are generally correlated and present small changes throughout their corresponding frames. Furthermore, by initially sampling the video, transitions from one scene to another become sharper and easier to detect. In the scene detection technique, and regardless of the features used, the difference metric of two arbitrary frames is normalized to be in the range [0,1]. The cosine similarity was used in the case of both CNN and BOVW features, while the correlation metric was used for HSV color histograms. Figure 3.6 illustrates the behavior of the mentioned difference metrics for each feature vector as a transition between consequent frames occurs.

HSV	0.95	0.94	0.35	0.17	0.30	0.95
CNN	0.99	0.98	0.75	0.55	0.79	0.96
BOVW	0.99	0.99	0.37	0.8	0.95	0.99

Figure 3.6: Adjacent frame differences

3.3.3 Difference Threshold

Following the calculation of frame differences for the whole video, we define a reference threshold in which we use for scene detection. The threshold is generally used for determining scene boundaries, as the behavior of the adjacent differences dictates the detection of scenes. By using a predefined threshold, a generalization of when the descriptor indicates nearly-different images is achieved. Although an exact value indicating a mismatch between two frames cannot be determined, a rough estimate suitable for scene detection was established. The designated thresholds for CNN vectors and HSV histograms were specified as in Table 3.2. The use of the predefined threshold is illustrated throughout the examples of scene detection. The values of the thresholds were chosen through comparing image similarities from the applied feature vectors with a subjective perspective.

Descriptor	Threshold	Metric
CNN Feature Vector	0.8	Cosine Similarity
HSV Histogram	0.6	Correlation

Table 3.2: Predefined thresholds of scene detection

After obtaining a reference threshold for scene splitting, a visualization is used for analysis. Figure 3.7 shows a plot of the correlation(y-axis) between adjacent frames(x-axis) for an arbitrary video. With a clear visualization of how adjacent frame similarities change, we are able to detect dramatic changes which lead to finding scene boundaries. The plot is a generalization among the whole video of how the pairwise distances behave, similar to what is specifically shown in Figure 3.6.

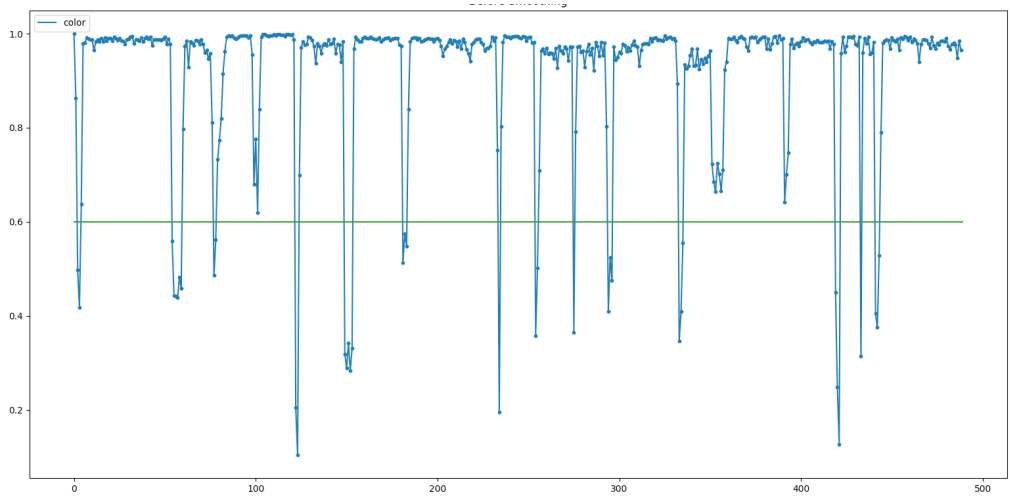

Figure 3.7: Adjacent frame differences for a complete video (HSV histogram)

3.3.4 Scene Boundaries

Following the definition of a scene-cut threshold on which scene detection mainly relies, we begin to determine scene boundaries. Specifically speaking, scene boundaries are specific video frames which indicate the beginnings and endings of scenes. Initially, all scene boundaries are found by detecting points which transitions below the defined threshold. A set of scene boarder boundaries for a portion of the frame differences is shown in Figure 3.8.

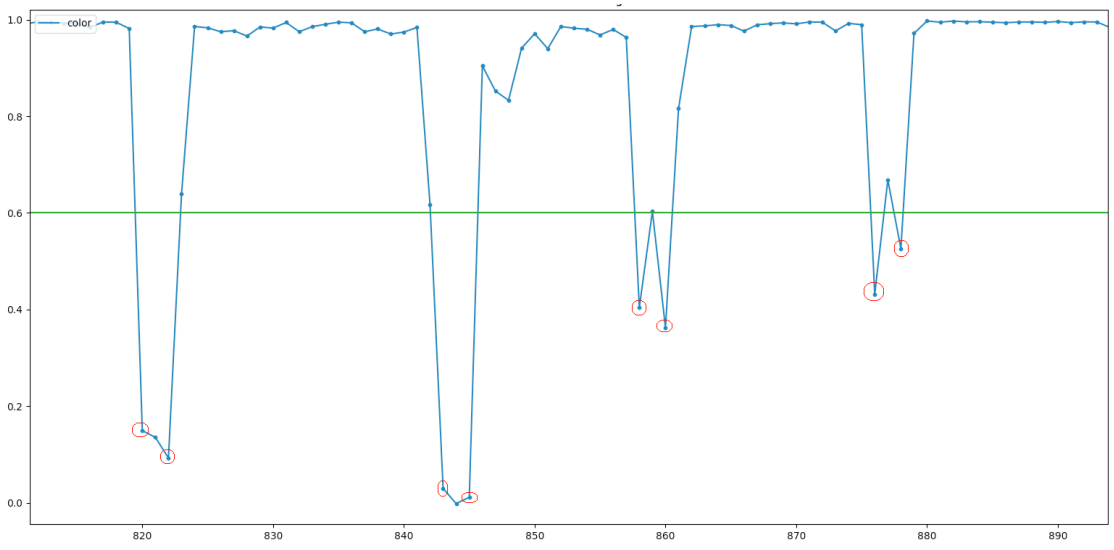

Figure 3.8: Finding scene boundaries from a difference plot

The points below the threshold in Figure 3.8 correspond to pairs of frames which introduce a significant difference that leads to the detection of new scenes. Therefore, we mainly rely on these pairs as the significant base from which beginnings and endings of new scenes can be found. Figure 3.9 shows a visualization of a scene boundary pair that suggests a boundary between two adjacent scenes.

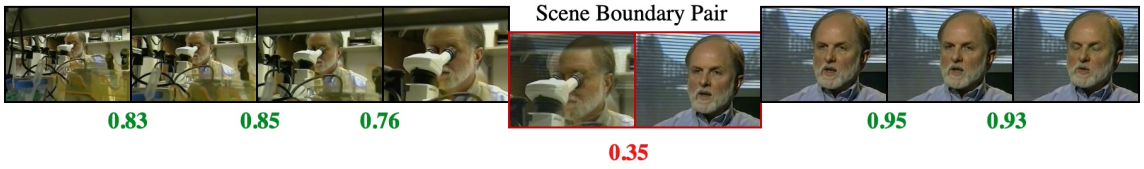

Figure 3.9: Scene boundaries visualization

3.3.5 Dissolving Frames and Frequent Transitions

Before proceeding in converting the generated boundaries into a set of practical scenes, we must address the issues to take into account while doing so. Namely, dissolving frames and frequent transitions around the threshold. It is worth mentioning that finding transitions around the scene-cut threshold is not enough to be certain of a new scene being introduced. One of the main challenges in doing so was the *dissolving frames* problem. A dissolving frame is an intermediate frame between two scenes used to introduce a smooth transition. Although such frames may be an element of entertainment within the video, they are quite problematic in scene detection. Figure 3.10 shows a group of dissolving frames with their corresponding difference sequence from Figure 3.7. Generally, dissolving frames are concentrated in groups of consecutive frames which fall below the cut threshold. This is due to the nature in which each dissolving frame introduces significant a difference compared to its preceding frame as it carries the frame sequence into a new scene.

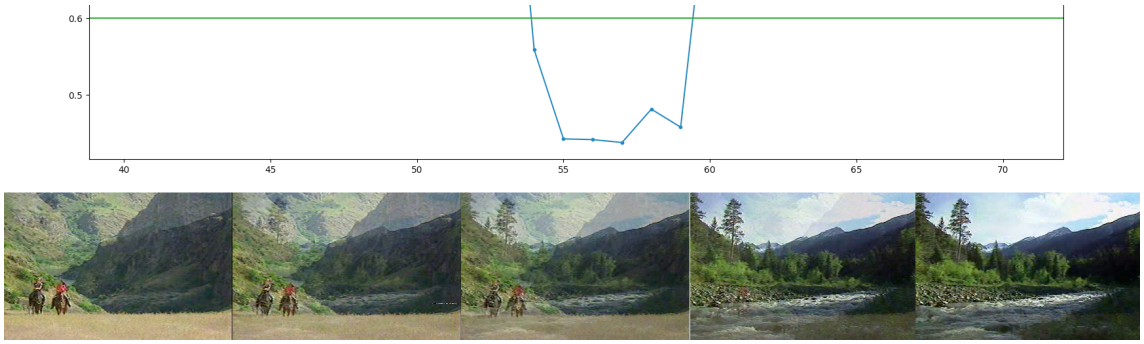

Figure 3.10: Dissolving frames

The final phenomenon to be addressed concerns frequent transitions across the threshold within a short period of time as shown in Figure 3.11.

Figure 3.11: Frequent transitions near the threshold

3.3.6 Scene Splitting

With the extracted scene boundaries, along with the defined issues to be cautious of, the video segments to be used as scenes can be inferred. Generally, scenes are found by taking segments between two boarder boundaries. However, two main constraints are enforced as scenes as generated from the boundaries. First, to prevent the case of an insignificant segment (a few frames) as in Figure 3.11, a minimum scene length is enforced. By setting a minimum length to the gap between two scene boarder boundaries, the frequent transition problem is eliminated. Scenes are quite variant in duration; however, it was found that an informative scene within the testing environment was no shorter than 2 seconds. Second, segments below the threshold which correspond to dissolving frames are neglected.

Algorithm 1 Generating Scenes from Boundaries

Data: Scene boundary points: List of frame positions
Result: Resulting scene segments: Pairs of boundaries
initialization
Scene Pairs: [ ]
Scene Boundaries: Threshold transition positions (Figure 3.8)
Min Scene Length: 2 Seconds(2*FPS Frames)
while *(B1,B2) as the next boundary pair* **do**
  if *(B2 - B1 > Min Scene Length))* **then**
    Add (B1,B2) to Scene Pairs
  end
  B1 <= B2
  B2 <= next Scene Boundary
end

Algorithm 1 generates the final scenes as they are represented as lists of beginning and ending frame positions. Furthermore, by taking pairs of boundaries at the boarders of segments below the threshold, groups of dissolving frames are eliminated. Figure 3.12 illustrates the final results of scene detection.

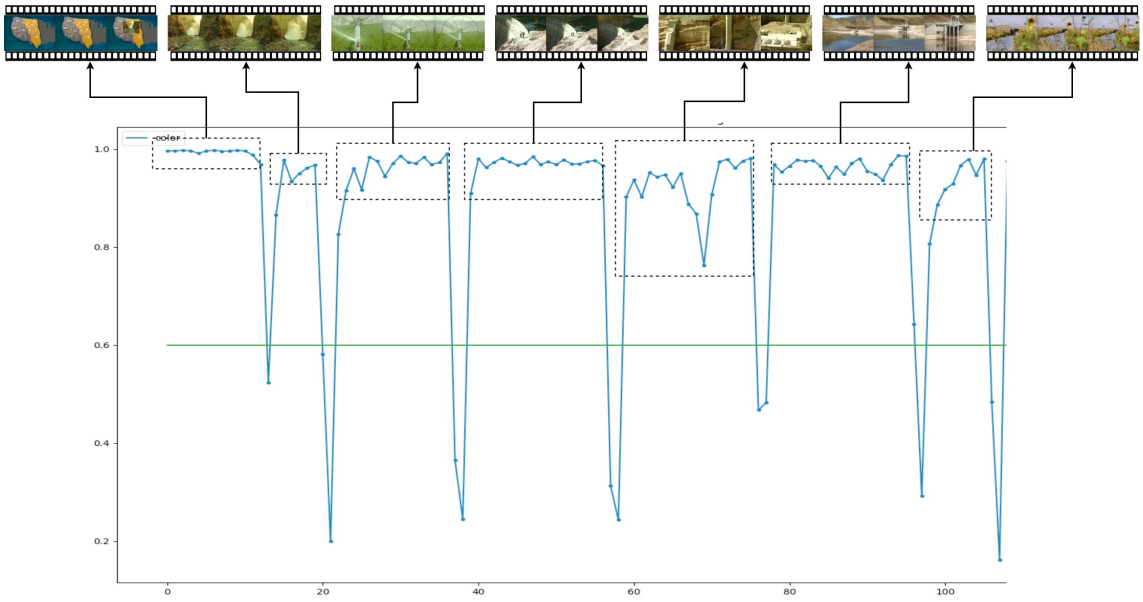

Figure 3.12: Selecting the final scenes

3.3.7 Limitations of Scene Detection

The proposed scene detection technique is able to generally split the video into a set of visually correlated segments. However, it is insufficient for keyframe extraction, as it suffers from evident limitations. By applying a single threshold to control scene boundaries, minor transitions which do not meet such threshold are therefore ignored. Furthermore, determining frame similarity in its nature is a subjective issue and cannot be performed directly through a single metric . It may occur, due to these limitations, that a detected scene may hold within it slight variations which could also be classified as other scenes. To illustrate, Figure 3.13 shows how an imprecise cut threshold prevents the detection of a possible scene. We notice that the adjacent frame similarities between the first scene undergo a change that could infer a new scene. However, since all of this was above the predefined threshold, both parts were considered in the same scene.

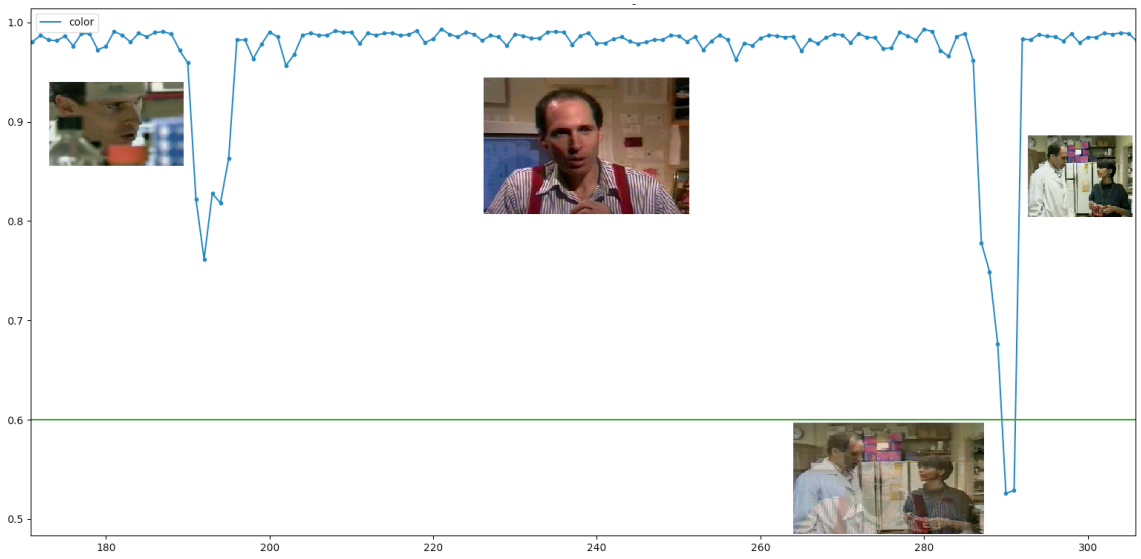

Figure 3.13: Scene detection limitations

Despite limitations, the degree of temporal segmentation performed by scene detection for the video is sufficient to carry on with keyframe extraction. To build on the detected scenes and reach the final keyframe set, an internal scene processing scheme is introduced. The final phase manages to implicitly solve the threshold limitation of scene detection, as multiple keyframes may be generated for a single scene.

3.4 Internal Scene Processing

At the beginning of this stage, we have a set of scenes that form the video ready from the previous stage. Now we want to process each scene to get a set of candidate keyframes that will represent it. To do so, our approach first begins by clustering the frames of each scene into a group of semantically similar frames. Second, following the grouping of the scene's frames into multiple clusters, each cluster then nominates a single keyframe. Finally, unique keyframes from the resulting candidate keyframes are chosen and are considered as the resulting keyframes for the corresponding scene. Furthermore, Since the processing of one scene is not related to the processing of others, each scene was processed separately and contributed to a portion of the video's keyframe set. In Figure 3.14, a high-level flow of how a scene is converted to keyframes by clustering is shown.

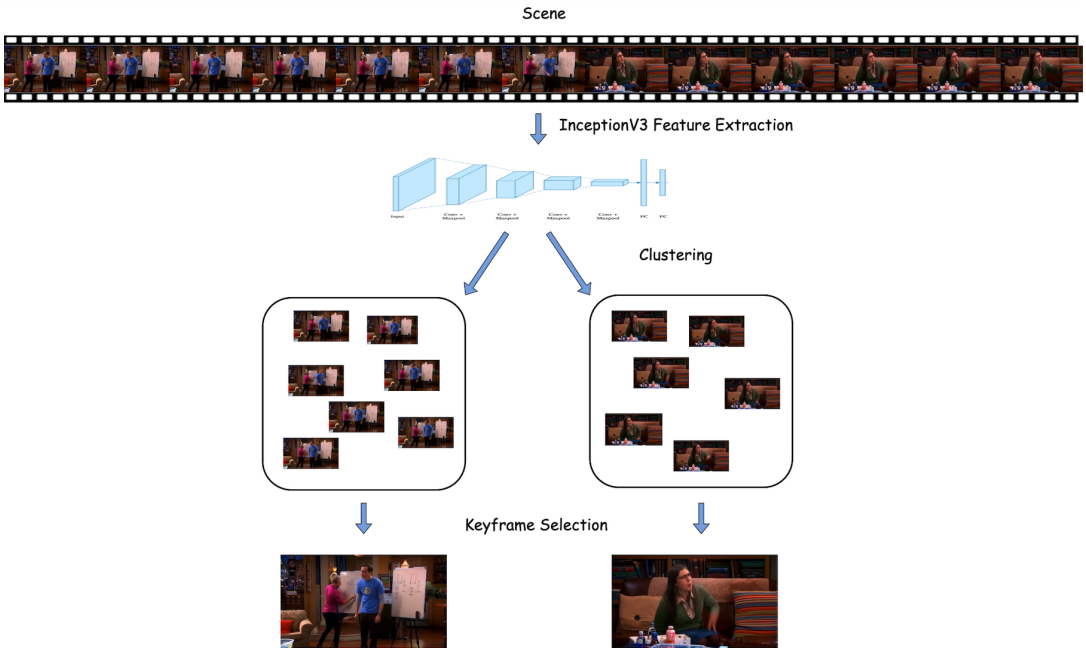

Figure 3.14: Scene-based keyframe extraction

3.4.1 Scene clustering

We aim from this step to group the frames that look semantically like each other and to choose one from each group or cluster. A scene may contain frames that vary semantically as scene detection may fail to distinguish between two or more consecutive scenes. Likewise, a scene may contain more than one important shot worth taking candidate keyframes from. Consequently, we perform clustering on the scene's corresponding frames using the previously extracted CNN feature vectors. The choice of using the CNN feature vectors in this stage rather than the previously used HSV color histograms was due to its their precise representation. Since the scenes themselves came from segmenting the video into parts based on the HSV color histograms of its frames in a sequential manner, this indicates that frames of the same scene are very much alike in terms of color. On the other hand, the feature vectors extracted from the pretrained CNN are able to describe more semantic representations including the existence of different objects within the different frames of each scene.

The first clustering technique of choice was k-means. K-means aims to group a set of scene frames into k semantically correlated groups. For k-means to properly function, a predetermined value of k needed to be defined, in order to specify the number of cluster for each scene. Furthermore, the euclidean distance was used to calculate the differences between the CNN feature vectors in order to group them in the correct clusters. A single keyframe is chosen for each cluster, where the most cluster-representative keyframe has maximum similarity with others of the same cluster. Nevertheless, with k-means representing each cluster with a center/mean vector, and therefore being sensitive to outliers, we extend our experiments to k-medoids clustering.

K-medoids chooses one of the keyframes (the medoid) as the center of each group of the k groups in each iteration rather than the mean. Consequently, each cluster is implicitley represented by its corresponding medoid frame. Additionally, k-medoids minimizes the sum of dissimilarities between frames in a cluster and the center frame of it thus results in a final medoid that is as similar to the rest of the frames in the group.

Regardless of the chosen clustering algorithm, the number of clusters (k) needs to be previously defined. In our experiments, k was varied such that $k \in (2, 3, 4)$. To choose the optimal value of k, the *silhouette*[25] score was calculated at each value of k. The silhouette cluster analysis method is a common technique used to select the optimal number of clusters for a certain clustering method. In silhouette analysis, each cluster is represented by a silhouette, which indicates its internal purity and correlation. The average silhouette score is calculated by considering each of the clusters. The value of k producing the maximum silhouette score was used in both cases.

3.4.2 Candidate Keyframe Selection

To build on the resulting clusters, each cluster nominates a single keyframe to contribute to the final set. In order to do so, the most representative frame out of each cluster is chosen. In the case of k-medoids, the medoid frame is directly chosen since it has a guaranteed minimal dissimilarity compared to the remaining frames in its cluster. On the other hand, k-means represents each cluster with the mean of the cluster's feature vectors. Consequently, choosing a candidate keyframe from a k-means cluster was done by finding the frame with the minimal dissimilarity sum compared to the remaining frames. Let a cluster of n frames be denoted through it feature vectors as $C = \{v_i\}_{i=0}^{i=n}$, then the candidate keyframe for the cluster maximizes sim_j such that:

$$sim_j = \sum_{i=0, i \neq j}^{i=n} cossim(v_i, v_j) \tag{3.3}$$

where $cossim(v_i, v_j)$ is the similarity between frames at position i, j respectively.

3.4.3 Per-Scene Redundancy Removal

To guarantee that the resulting keyframes from the set of candidates are unique, near-redundant keyframes are removed. Consequently, each scene will generate from 1 to 4 final keyframes. Cosine similarity was used as the distance metric on the CNN feature vectors for the keyframes with a threshold of 0.8. To illustrate, Figure 3.15 shows a set of 3 clusters resulting in 2 final keyframes for the extracted scene.

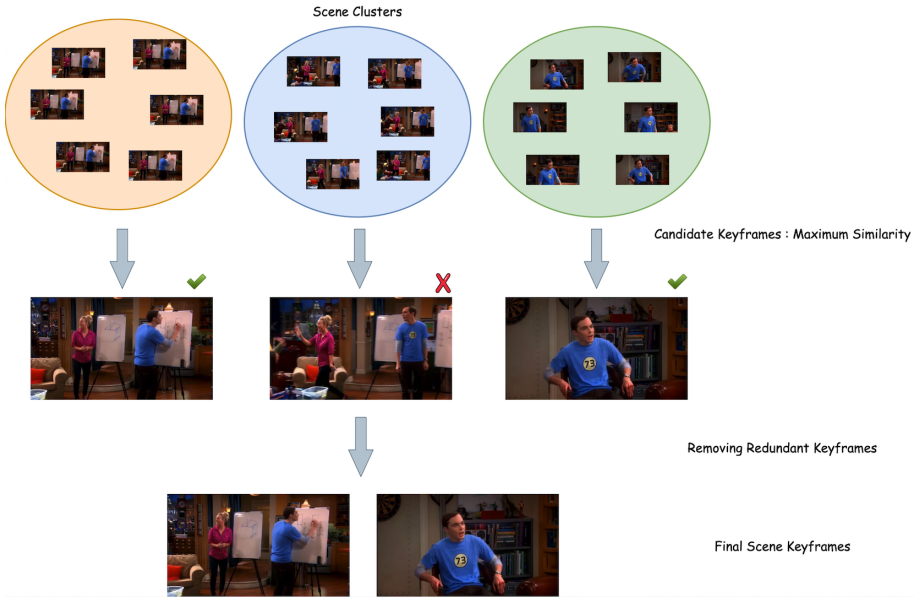

Figure 3.15: Removing repetitive keyframes from cluster-candidate keyframes

3.5 Collecting the Final Summary

To put the previously discussed stages together for the complete video, the final summary is collected from the results of processing each scene individually. Consequently, the final output is a temporally successive set of keyframes, highlighting the detected scenes. Figure 3.16 shows the process of collecting all keyframes from each scene to finally form the video's static summary.

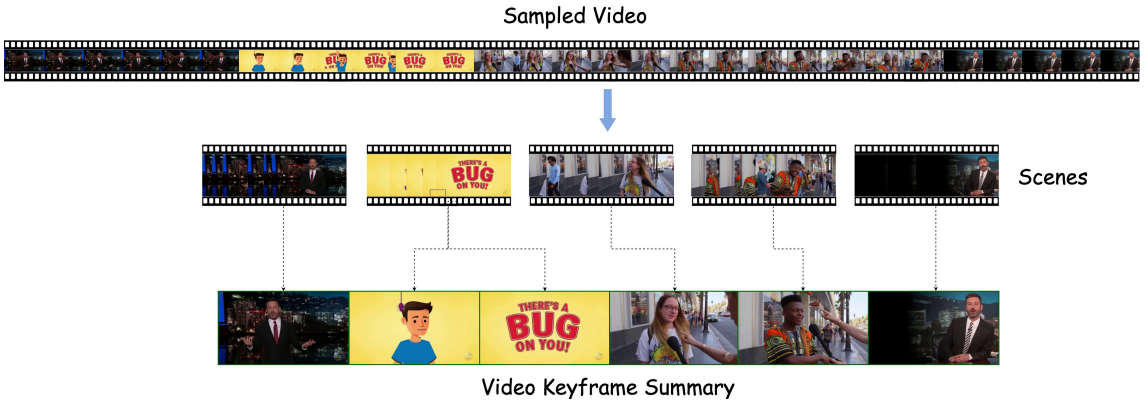

Figure 3.16: Collecting the final video summary from the extracted scenes

3.6 Tools

To conclude our discussion on the implementation of the keyframe extraction methodology, we will elaborate on the tools used to achieve it. In this section, we discuss the used programming languages, tools and libraries used in the implementation of the project's core summarization system.

3.6.1 Python

Due to the vast amount of support which Python provides for computer vision and data oriented processing, Python was our undebatable language of choice. Additionally, Python was found efficient with our extensive use of file-system based operations of reading and writing images, scenes and videos. Furthermore, Python supported the core image processing library used (OpenCV)[26] as well as easy vector and array manipulation through data analysis libraries.

3.6.2 OpenCV

The OpenCV(Open source computer vision)library mainly facilitated the required image processing operations dealt with in the project. Concerning the discussed keyframe extraction methodology, OpenCV was used to extract SIFT[22] keypoints, compute HSV histograms, debug the system's flow and much more.

3.6.3 Machine Learning Libraries

To cover the project's dependency on clustering and feature extraction, several machine learning libraries were put to use. First, SciKit-learn[27], which is a python machine learning library, was used to cluster the scenes as discussed in 3.4. Meanwhile, Keras[28], which is a neural-network library available in Python, was used in order to load the pre-trained InceptionV3[23] model and extract feature vectors from the fully-connected layer of the network.

Chapter 4

Application

To enable users to summarize their own videos and to demonstrate the power of extracted keyframes in understanding the contents of the video, an end to end web application was implemented. Through demonstrating video summarization in an accessible way, we believe that we can portray the benefits of keyframe extraction on video management.

4.1 Application Overview

Our web app is mainly divided into its frontend and backend sides. With the implemented app, users can upload a video from their device and get it summarized and analyzed. Furthermore, the keyframes and insights for each one are displayed. Also, users can browse and play the videos uploaded by others and search them using text queries.

4.2 Backend - Server Side Summarization

The backend side of the project serves all the modules vital for our video summarization process and its usage in understanding the contents of the video. This enables users to search through the uploaded videos, which are persisted by the system and to get insights of them as well. The application's backend allows for multiple features which leverage the concept of video summarization. First, the upload process and preprocessing of the video is facilitated. Additionally, the video undergoes the core summarization system for keyframe extraction. Furthermore, the generated keyframes are understood through their insight, such as labels and web entities. Uploaded videos, along with their persisting their keyframes, are presisted in a Content Delivery Network (CDN). Meanwhile, video metadata is persisted in a database. Through the previously mentioned storage scheme, video search in the system was allowed. Figure 4.2 shows the backend system architecture.

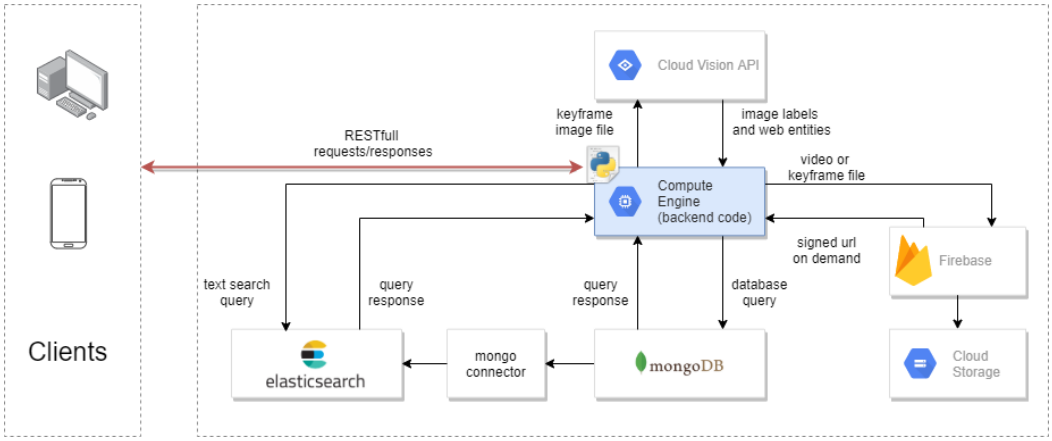

Figure 4.1: Application's backend system architecture

4.2.1 Handling upload

The upload Application Programming Interfrace (API) end-point receives a request from the client-side which contains the video's title, description and byte stream. There is a variety of video extensions, some of them can be played on browsers like .mp4 and .ogg while others can't. To overcome this limitation and to eliminate any effort that has to be done by the user, we convert any video with format other than .mp4 to .mp4 with h.264 encoding.

4.2.2 Storage of videos, Keyframes, and Metadata

In order to serve the videos so that they are streamed by the frontend, a content delivery service is needed. Our choice was Firebase storage which is a part of the Google Cloud Platform[29](see 4.4.4). Each video is stored in a folder along with its extracted keyframes. Whenever the frontend requests videos or keyframes, a new signed source link with a token is generated for each of them, so that the frontend does not handle the authentication process to Firebase. For the metadata of each video, we used a non-relational database which stores data in documents with JavaScript Object Notation (JSON) format, our choice for it was MongoDB[30](see 4.4.2). This made the process of managing the database a lot easier since no schema is required and the data is stored with the same way it is dealt with objects in code. The stored information are a unique Universally Unique Identifier (UUID), a timestamp, the original filename of the video, its duration, its provided title, its provided description, links to both video and its thumbnail, and a list of keyframes. Each keyframe contains a designated link(reference in Firebase) and a list of extracted tags. Figure 4.2 shows the exact structure of the stored metadata.


```
_id: "84606e1b-2e29-40e3-8d4e-ad9a0a9c2549"
name: "small.mp4"
title: "toys"
timestamp: "1357466440"
description: "a child playing with his toys"
firebase_url: "https://storage.googleapis.com/video-summarization-project.appspot.com..."
keyframes: Array
  0: Object
  1: Object
      firebase_file_name: "84606e1b-2e29-40e3-8d4e-ad9a0a9c2549%2F1.jpg"
      firebase_url: "https://storage.googleapis.com/video-summarization-project.appspot.com..."
    cloud_vision_labels: Array
        0: "Toy"
        1: "Lego"
        2: "Vehicle"
        3: "Space"
  2: Object
thumbnail_url: "https://storage.googleapis.com/video-summarization-project.appspot.com..."
duration: "2:30"
```


Figure 4.2: Video metadata in database

4.2.3 Understanding of Generated Keyframes

After we get the final set of keyframes, we use them as a representation of the whole video. Each keyframe is used to get insights about the video so that the content of it is understood. We used the Google Cloud Vison(part of Google Cloud Platform[29], see 4.4.4) service to get two types of tags: keyframe labels and keyframe web entities. Keyframe labels can be thought of as the classification of the image from millions of classes that span a lot of things in the world. This includes products, general objects, activities, animal types, locations and more. The returned response from google cloud vision API contains a set of labels with degree of confidence for each. We limit the stored labels to 70% or more degree of confidence so that we don't get unrelated labels that may affect the search performance. Figure 4.3 shows a demo of how google cloud vision detected labels such as 'Child', 'Play' and 'Toy' among others from the uploaded image.

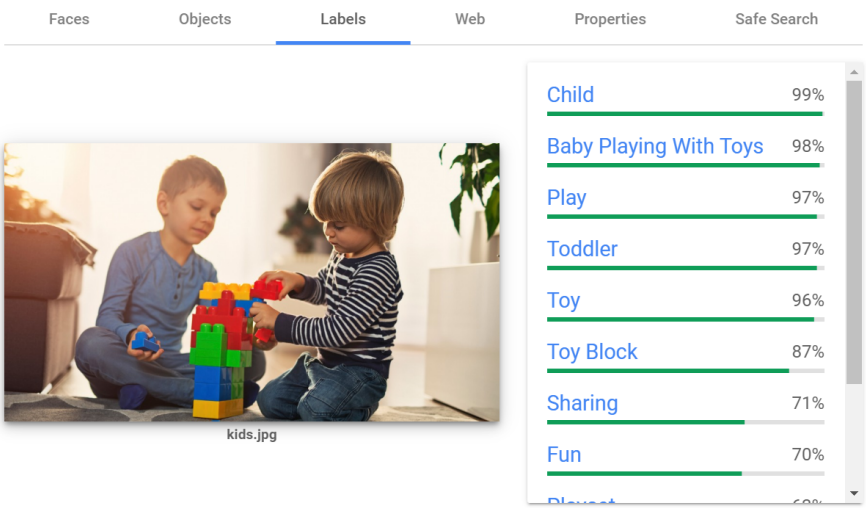

Figure 4.3: Cloud vision labels

Keyframe web entities are generated by Google Cloud Vision after searching the web for the image and returning what people wrote along with this image. This is very useful in the cases of recognizing named entities such as celebrities, TV series and movies, specific places such as universities and much more. A set of entities are returned with their degree of confidence, again we limit the stored entities to 70% or more. This restriction may sometimes lead to keyframes with are tags which is acceptable, since it only occurs when there is no obvious tags for the image. Figure 4.4 shows how google cloud vision detected the name of the place in the uploaded image which is Birzeit University.

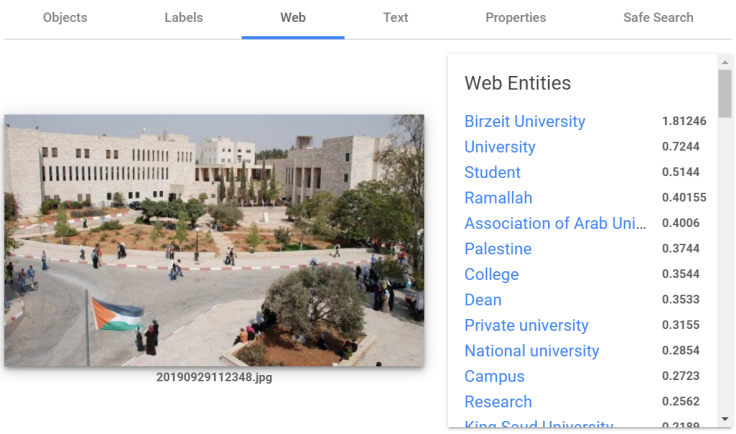

Figure 4.4: Cloud vision web entities

4.2.4 Video search

After analyzing the keyframes and generating tags from each of them, we are able to provide a search service on the uploaded videos. The supported type is a text query and the returned response is a set of video metadata with features that match the query. In our system, the matching depends on the title of the video and on the set of generated tags (labels and web entities). The search is not strict and accepts fuzzy searches, for example: 'Vehicle', 'vehicles' and 'vehicls' in a query match a video with 'Vehicle' tag. And "barcelona vs liverpool" query will match a video with "Barcelona game against Liverpool" title. We used Elasticsearch[31](see 4.4.3) to match user queries and synced it with MongoDB. In that way, whenever a metadata of a video is added to the database, it will be indexed by Elasticsearch.

4.3 Frontend

For the client side, we introduced the services we have in a user friendly experience, so that the user can upload, review, skim and watch videos in an easy way. The front-end was written using the ReactJS[32] library(see 4.4.5), as the application's features are all accessible through the user's browser. To further illustrate, Figure 4.5 shows a high-level architecture of the client-side, along with its communication with the server-side components.

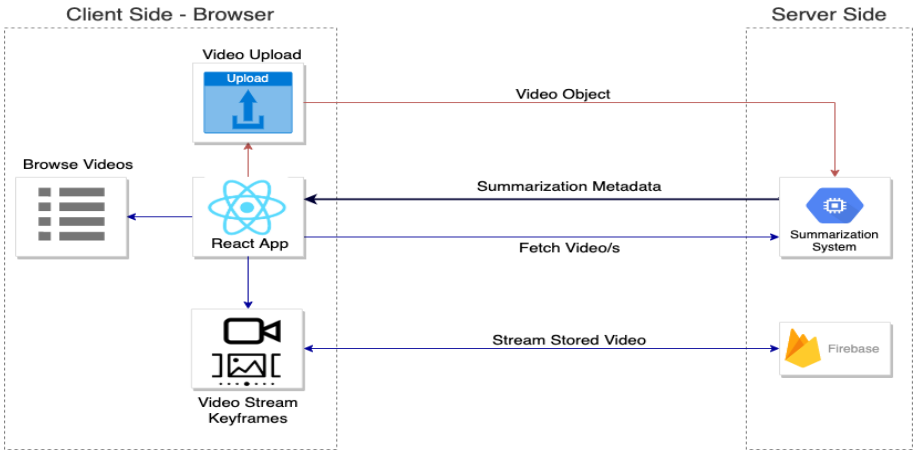

Figure 4.5: Client-side architecture

4.3.1 Video Browsing

Initially, the main application view shows all the stored videos, with 7 videos per page. Each video is abstractly described through its title, thumbnail, description, and duration. While browsing, the user has the option to navigate to the summarization details of any video. Figure 4.6 shows an example of how the stored videos are browsed through the application.

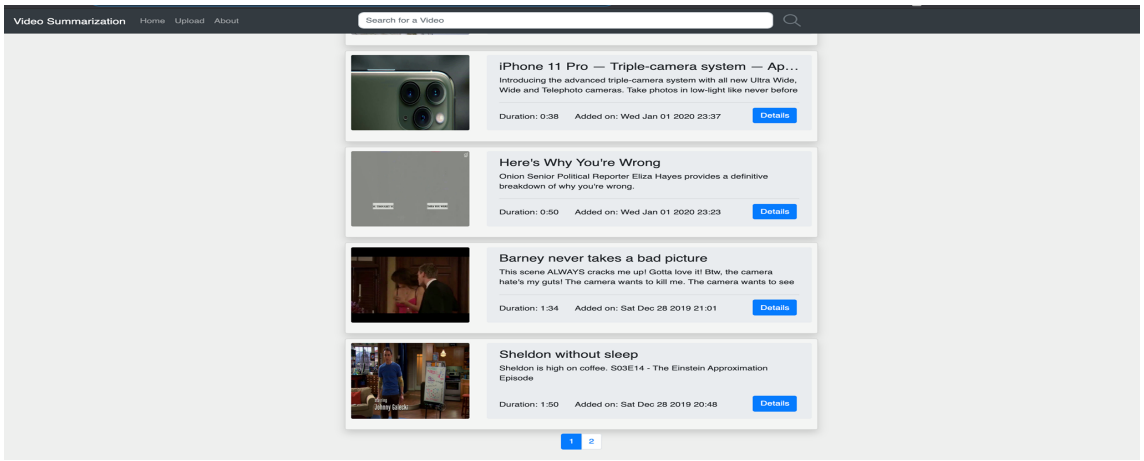

Figure 4.6: Home view of the application

4.3.2 Video Search

As previously mentioned, with video summarization, insights drawn from keyframes now form a powerful index in which we can search a video management system. In our application's user interface, we allow users to search for stored videos through both their titles and keyframe insights. Additionally, with video search being tolerant to minor errors, users are able to formulate their queries flexibly. For example,the search for the phrase: "Jimmy Kimmel" yields the results that

contain such information about the query, in this case, the video title. The example is show in Figure 4.7

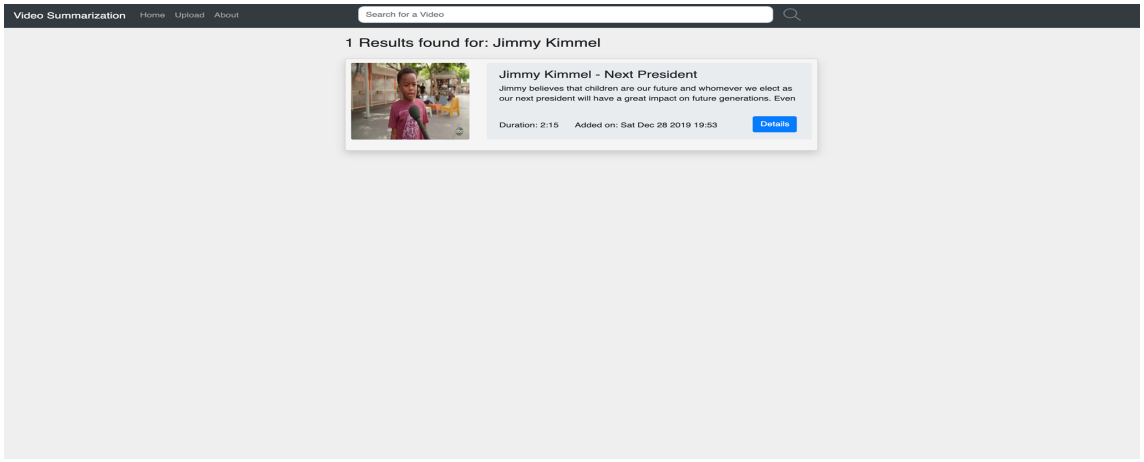

Figure 4.7: Search result for the phrase: "Jimmy Kimmel"

On the other hand, queries matching keyframe insights were also found to be a form of search enhancement. Figure 4.8 shows how the video "Sheldon without sleep" was returned when searching with the query "Big Bang Theory".

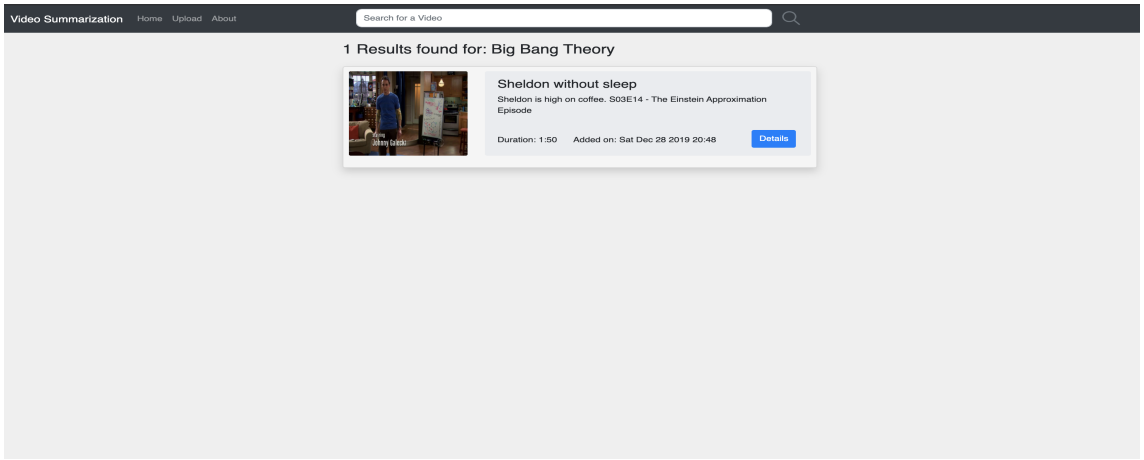

Figure 4.8: Search by keyframe insights

4.3.3 Video Upload

The upload feature allows users to add the video they want to summarize and extract information from. It requests from user to optionally enter the video name and description then the user uploads a video locally from his/her device. After uploading a video the user will have the option to see the summarization details including the keyframe insights. Uploading a video is demonstrated in Figure 4.9.

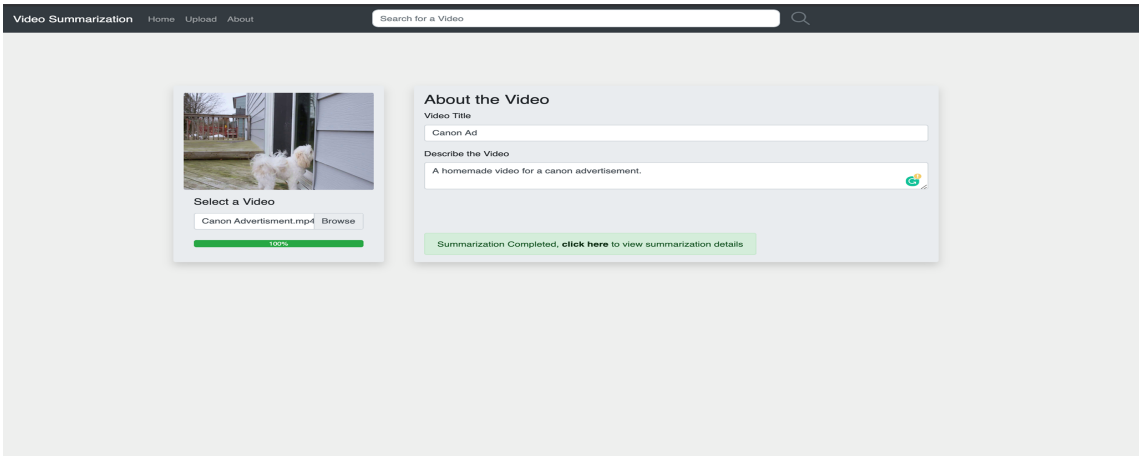

Figure 4.9: The upload view in the application

4.3.4 Summarization Details

This view provides the core of our project and its results for each video. It's responsible for viewing the static summary of the video as a slide show of keyframes which gives the user the ability to skim the video by looking at the generated keyframes. Another functionality is when the user selects a keyframe to view its insights and related tags. All of this is also wrapped with a video player including the video name, duration, upload date and the description similar to normal video content provider websites. Figure 4.10 demonstrates the video details view features.

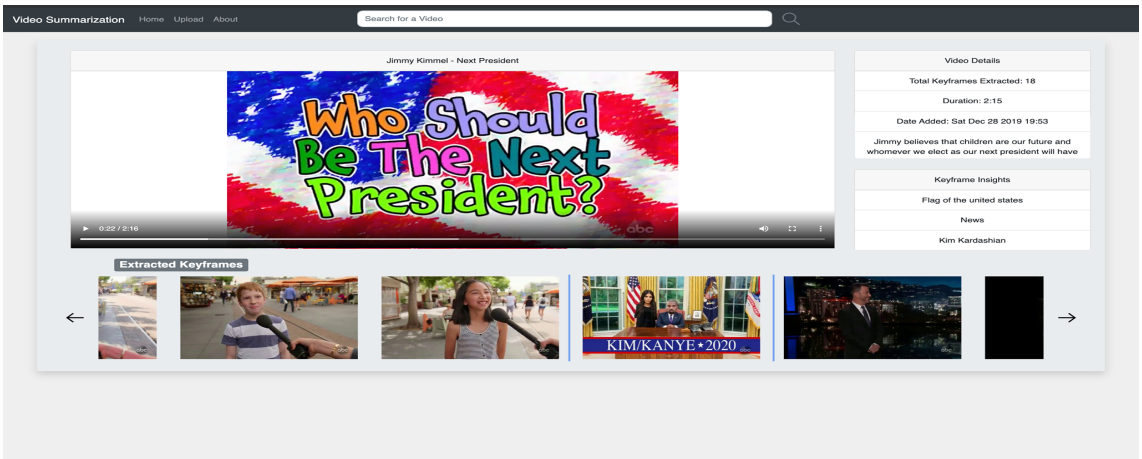

Figure 4.10: The summarization-details view in the application

4.4 Tools and Technologies

In order to provide video summarization as a user-friendly web application, multiple tools and technologies were used. The tools which the application is mainly composed of were the base of implementing end-to-end video summarization.

4.4.1 Flask

Flask[33] is a lightweight web application framework that is used to make web applications with ease. Flask is written in python and is one of the most popular frameworks because of its ability to get a working application quickly and easily. Flask also has the ability to scale to complicated projects. Since the video summarization module is written in python, flask is a perfect fit to

serve it. Flask was used in our application to make RESTful APIs and expose them using the development server provided with it. A web application written in flask uses a defined port to receive requests and send responses from it. The APIs are then accessible by the frontend using the public IP of the server and that port number, 8080 in our case.

4.4.2 MongoDB

MongoDB[30] is one of the most popular databases used in the world. It does not depend on relations between tables as in the traditional relational databases. For that reason, it does not require schema and thus it is easier to begin with since no complicated planning of the relations is required in advance. MongoDB depends on what is called collections which can be thought of as tables in relational database. Each collection is dedicated to serve an important entity. Collections contain multiple documents that can have different structure and are analogous to entries of tables. Documents use JSON like structure to store data in keys and values. Values can be nested into other set of keys: values and can also be arrays. The files of the database are stored in the filesystem of the OS in a directory of user choice. A standalone database or a replica database can be created. Standalone database will result in one database operating while the replica will result with multiple replicas of the same database that are efficiently maintained using an advanced algorithm. MongoDB has a community server that serves the requests to the database and operates on a defined port, 27017 in our case.

4.4.3 Elasticsearch

Elasticsearch[31] is a very common and very fast search and analytics engine. It is used to search in large texts such as logs or in huge databases with a lot of documents and can work with both structured and unstructured data. Its algorithm uses advanced techniques to index data and retrieve them in realtime. Elasticsearch operates on a defined port which is 9200 by default. It receives RESTful requests on the default port which can be index creation or data modification or search query requests. The Index can be directly created from a pre-existed MongoDB replica database. The search queries define the fields to be matched and support wildcards, Regexes and Fuzzy Searching options. One can also define (AND, OR and NOT) between different queries in one request. The returned response contains the documents as stored in the original source (MongoDB for example) with a score for each one.

4.4.4 Google Cloud Platform

Google cloud platform[29] offers a suite of services for many purposes, including hosting, storage and machine learning. As far as our web application is concerned, our usage of google cloud platform services focused on three main services: Compute Engine, Firebase and Cloud Vision.

Compute engine provides persistent, high-performance virtual machines. By using compute engine, we were able to have our web application hosted on a server with suitable specifications for handling user requests and heavy video processing. Firebase, on the other hand, was used so that we could store the uploaded and managed video content. By assigning a unique reference ID to a video, it became accessible through multiple components in the application. Lastly, Cloud Vision is a service used for image understanding by extracting metadata such as labels,objects and related web-entities. Cloud Vision metadata was used excessively to enhance video search in our application.

4.4.5 ReactJS

ReactJS[32] is a JavaScript library used for building user interfaces. The client-side implementation of our application depended mainly on the react library, as it was used to introduce video summarization as a user interface.

Chapter 5

Evaluation and Results

Similar to any proposed research methodology, keyframe-based video summarization techniques need to be evaluated and assessed. Hence, by evaluating a video keyframe summary, we are able to gain direct insight on the summary's quality and representation of a video. In this chapter, we discuss the evaluation of our proposed methodology, and the results that have been achieved on a reference dataset. Furthermore, we demonstrate the challenges faced when objectively measuring the quality of a system-generated video summary.

5.1 Evaluation Overview

The evaluation our video summarization system depends mainly on comparing summaries to their user generated equivalents. By doing so, the quality of a system-generated summary is assessed in a human's perspective. Therefore, we carry out our system's evaluation through comparisons to real user judgements. Unfortunately, the correctness of a video summary is vague and presents challenges in objective calculation. Additionally, resources in which we can compare our results to other methods are scarce and require manual data collection and annotation. The first step of evaluating our work was by bearing in mind the usage a valid keyframe-based video summarization dataset. Figure 5.1 shows the general flow of the our evaluation methodology.

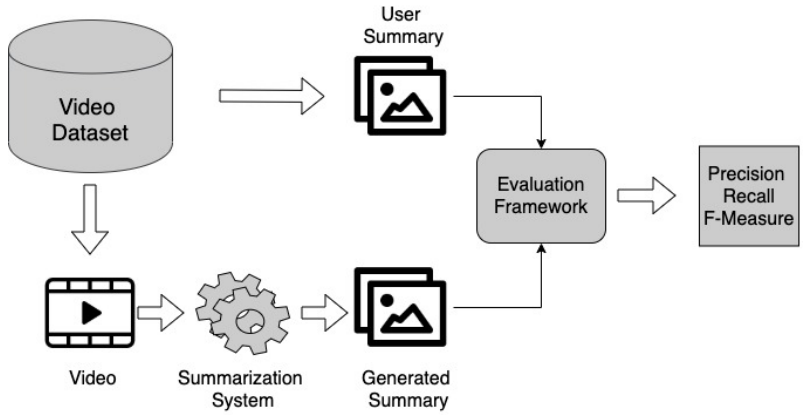

Figure 5.1: System evaluation methodology

5.2 Evaluation Dataset

Generally, evaluation aims to verify the quality of an approach in solving a problem as well as comparing it to previous work. Unfortunately, keyframe based video summarization research suffers from the scarcity of public datasets and results. The majority of previous work concerning keyframe extraction does not publicly disclose of exact an testing dataset, annotations, or precise evaluation technique. Consequently, evaluating our proposed work was a challenge and required implementation. Additionally, objective evaluation of a video summary requires user-created annotations which act as ground-truth(reference) summaries.

To evaluate our proposed work, in a way that satisfies the previously mentioned constraints, we test on the VSUMM dataset. The VSUMM dataset was introduced in De Avila et al.[12] as a set of videos from the Open Video Project(OVP)[34] repisitory were chosen. Furthermore, the dataset consists of videos previously used by approaches such as DT (Mundur, Rao, and Yesha[13]), STIMO (Furini et al.[10]) and the OV(Open-Video) summaries by (DeMenthon, Kobla, and Doermann[11]). Therefore, we are able to compare our approach with previous ones on the same dataset. For each video present in the dataset, VSUMM provides 5 manually created summaries, thus offering multiple ground-truth summaries of various perspectives. In addition, the VSUMM dataset was found to be a suitable fit for our proposed evaluation method for its support to general activity videos and being not tied to a specific domain. Another frequently used dataset in the literature is the Kodak[35] dataset. However, publicly distributed annotations and results are not provided in the form of reference keyframes, making it inadequate for our evaluation methodology. Table 5.1 shows the main attributes of the VSUMM dataset which was mainly used throughout our system evaluation methodology.

Reference	Total Videos	Video Length	Users Involved
De Avila et al.[12]	50 videos(from OVP)	1-4 Min	50 Users (5 videos per user)

Table 5.1: VSUMM dataset characteristics

Figure 5.2 shows an example of two manually created ground-truth summaries for one of the videos from OVP[34] used in the VSUMM dataset.

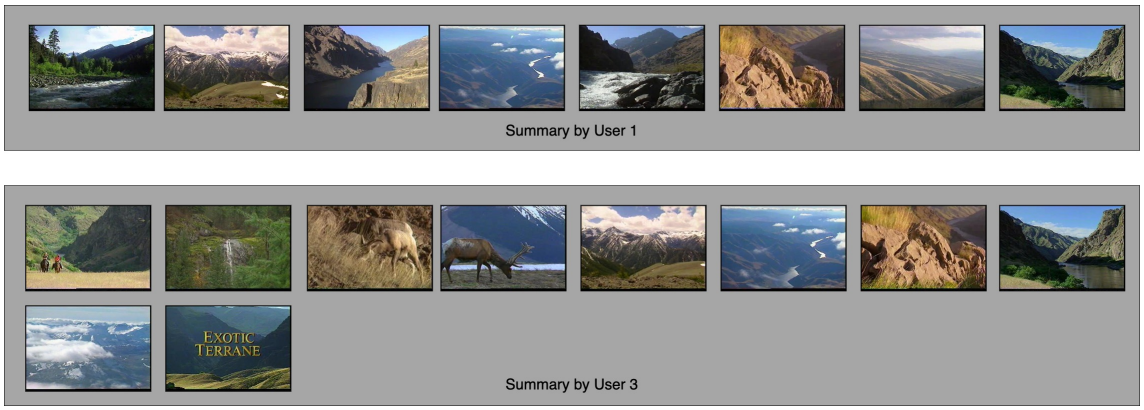

Figure 5.2: Ground-truth summaries created by users (VSUMM dataset)

The authors of VSUMM proposed an evaluation method called Comparison of User Summaries (CUS), and derived two main metrics from it, namely, CUSa and CUSe. CUSa measures the accuracy and it's the fraction of common keyframes between the two summaries over the number of keyframes in the target summary, see Equation 5.1. Where the CUSe measures the error and it's the fraction of number of non-matching keyframes from the extracted summary over the number of keyframes in the target summary, see Equation 5.2.

$$CUSa = \frac{|Common\ Keyframes|}{|Target\ Summary|} \tag{5.1}$$

$$CUSe = \frac{|Non - common\ Keyframes\ in\ extracted\ summary|}{|Target\ Summary|} \tag{5.2}$$

The CUSa acts like the recall in any information retrieval assessment system which ranges from 0 to 1. However, the CUSe ranges from 0 to the ratio of keyframes between the two summaries. Therefore, the comparison between different approaches is difficult as metrics are not normalized. Furthermore, in the VSUMM evaluation framework, two keyframes are matched if their HSV similarity is above 0.5. Hence, their evaluation framework is prone to error in determining a match between two keyframes. For this, we introduced our evaluation framework that handles the errors and drawbacks in the VSUMM's framework, and discuss it in details (5.3).

5.3 Evaluation Methodology

5.3.1 Keyframe Matching

Determining if two keyframes are equivalent or not is not a straightforward task. Some small changes between two frames may lead to a completely different feature vector. Therefore, two main feature vectors were tested, namely the HSV color histogram and the CNN feature vectors. None of the feature vectors showed robustness on all the areas, for instance, the HSV histogram showed robustness regarding the image skewing and inner objects position changes. Meanwhile, the CNN feature vector showed more robustness regarding to contrast changes. Figure 5.3 shows an example of how both metrics were combined to determine precise image matching.

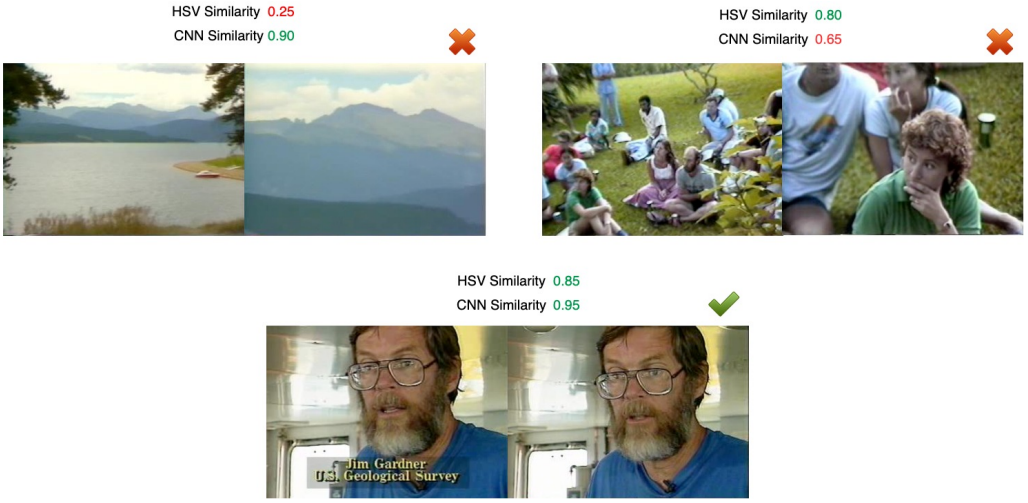

Figure 5.3: Keyframes similarity using HSV and CNN

Therefore, we chose to determine the similarity of two images based on both the HSV and CNN feature vectors. The match occurs if the HSV match is above 0.5 and the CNN match is above 0.8. In the subsections to follow, the terms *relevant*, *similar* and *matched keyframes* are used interchangeably.

5.3.2 Evaluation Metrics

In order to thoroughly asses our system and see how it compares to real summaries, we defined two main types of evaluation metrics, namely, precision and recall. The defined evaluation metrics generally measure the relevance between the extracted summary and a target summary. This subsection will demonstrate both of them and mention their drawbacks and then introduce the measure which combines both of them.

Precision measures the fraction of common keyframes between the extracted summary and the target summary with respect to the extracted summary. In mathematical representation, it's the number of relevant keyframes between the two summaries over the size of the extracted summary, see Equation 5.3.

$$Precision = \frac{|Common\ Keyframes|}{|Extracted\ Summary|} \tag{5.3}$$

As seen, precision would be affected badly if the size of the extracted summary is large compared to the target summary. Hence, a fewer number of extracted keyframes results in a better value for precision. However, if the system extracted only a single relevant keyframe, then the precision would be 100%, which is the major drawback in the precision.

Recall measures the fraction of the common keyframes between both of the summaries with respect to the target summary. In mathematical representation, it's the number of relevant keyframes between the two summaries over the size of the target summary, see Equation 5.4.

$$Recall = \frac{|Common\ Keyframes|}{|Target\ Summary|} \tag{5.4}$$

Because the size of extracted summary is fixed, a larger extracted summary is a better result achieved for recall, as one can get 100% recall if the extracted summary composed of all the frames. As concluded from both the precision and recall, none of them can stand alone as a good evaluation assessment. Therefore, the F-score is used. The F-score is the harmonic mean of precision and recall, and expressed in Equation 5.5.

$$F - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{5.5}$$

Both of precision, recall and f-score are used in our evaluation assessment as each one exposes and reveals different aspect about our summarization system. As seen, all the evaluation metrics depend on the number of common keyframes between the two summaries. The methodology of extracting the common keyframes is disscussed in the next subsection.

5.3.3 Summaries Comparison

Comparing the extracted summary with a target summary is done by comparing each keyframe from the extracted summarfy with every other keyframe in the target summary. At each iteration, a keyframe from the extracted summary is compared with all other keyframes in the target summary. If two frames are matched based on the definition in 5.3.1, then the maximum CNN similarity is taken into consideration. After the iteration, if the maximum CNN similarity value was greater than 0.8, then we record a match and discard the keyframes from the both the summaries.

Figure 5.4 shows the mapping of common keyframes between the extracted summary and a target summary, along with the evaluation metrics. In addition, Algorithm 2 discusses the procedure of finding the number of common keyframes between two summaries.

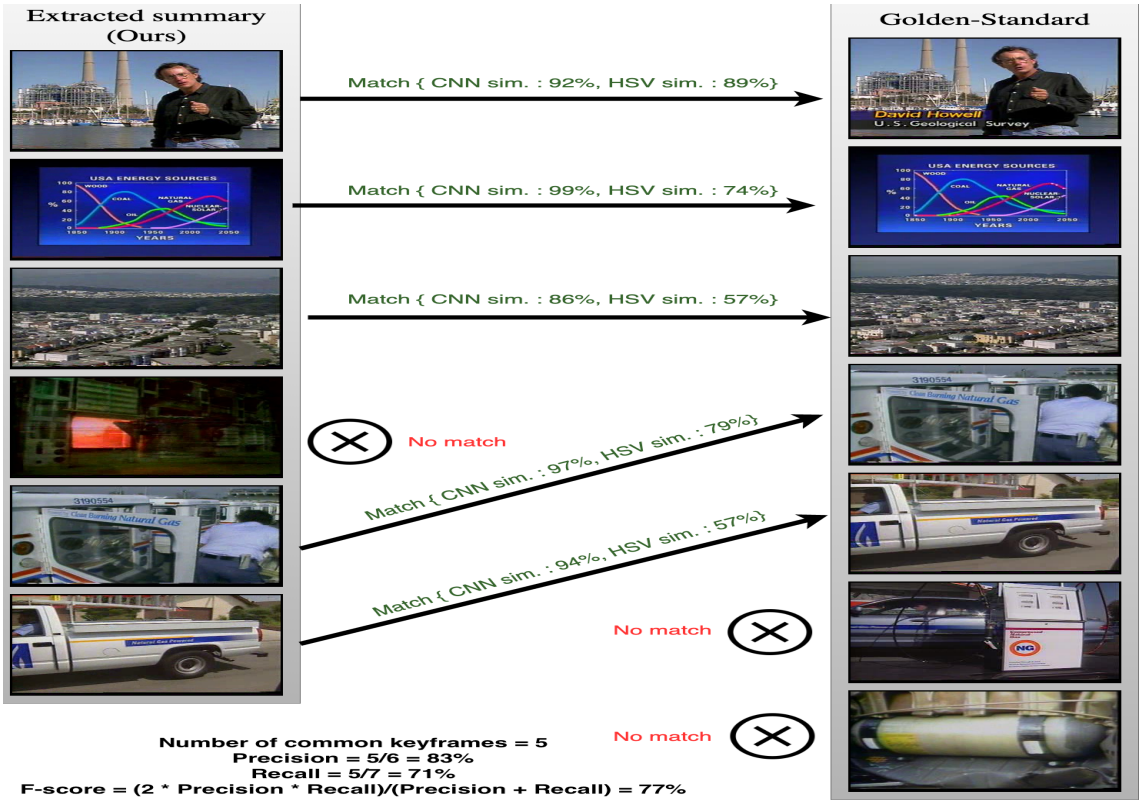

Figure 5.4: Finding common keyframes between reference and generated summaries

Algorithm 2 *Finding the number of common keyframes algorithm*

Data: The two summaries: the extracted summary (S1) and the target summary (S2)
Result: The number of common keyframes between the summaries
initialization
CommonKeyframesNumber = 0

for $frame_i$ *in* $S1$ **do**
  $maxMatchingFrame = null$
  $maxMatchPercentage = 0$

  for $frame_j$ *in* $S2$ **do**
    if $frame_i$ *match with* $frame_j$ **then**
      $maxMatchingFrame = frame_j$
      $maxMatchPercentage = MatchCNN(frame_i, frame_j)$
      end
    end
    if $maxMatchPercentage > 0.8$ **then**
      increment CommonKeyframesNumber
    end
  end

5.4 Evaluation Results

Having defined the reference ground-truth summaries for each video, along with the evaluation procedure, we can finally conduct experiments to measure the quality of our work. In this section, we discuss the achieved results under the implemented evaluation framework. Furthermore, we split the experiments into two main categories. First, all user-created summaries are taken into account as we evaluate our system-generated summaries. Second, golden-standard summaries are used as the reference summary for each video in the dataset.

5.4.1 User-Based Evaluation

As previously discussed, a completely correct video summary is a vague idea and differs from one person's perspective to another. The VSUMM dataset was no exception to such challenge, as each of the five users gave a unique group of keyframes by which they believed the video had been summarized. In this set of experiments, we aim to compare the output summary of each video with the each of its 5 ground-truth summaries. Furthermore, we study the variation of parameters and implemented methodology such as clustering methods, feature-extraction and thresholds.

Since user-based evaluation takes into account all manually created summaries for each video, the final results of the discussed evaluation metrics consider the average values from all users. To illustrate user-based evaluation, Figure 5.5 shows a system-generated summary from our implemented system in comparison with its corresponding user-made summaries.

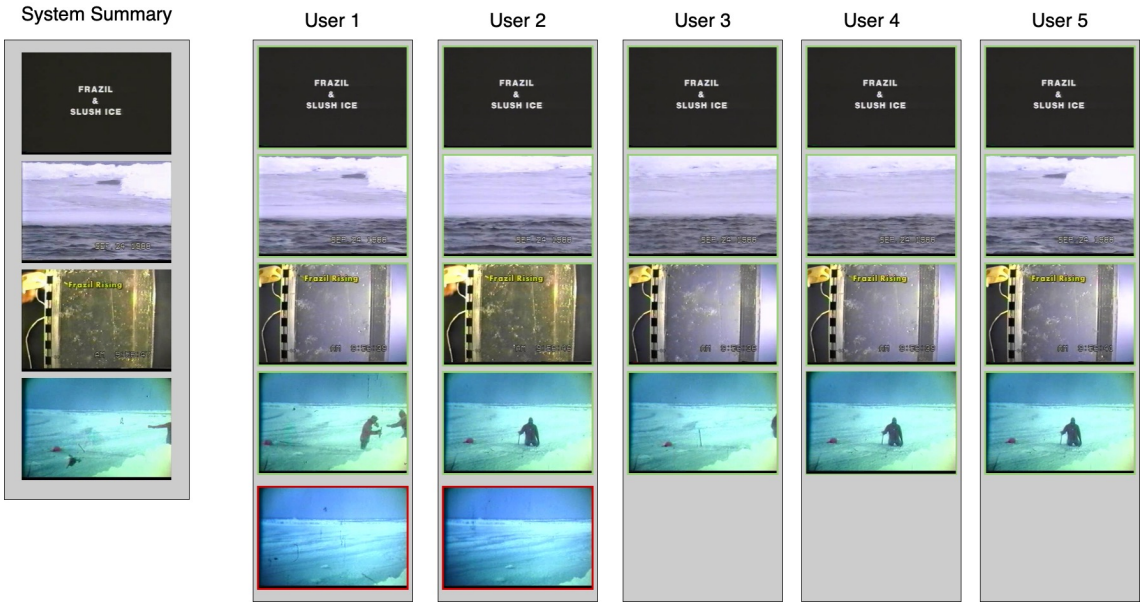

Figure 5.5: User-based evaluation (video 70)

Additionally, Table 5.2 shows the resulting metrics for the same evaluation. The final result is shown in the Avg column, as each of the users is taken into account.

Metric/User	1	2	3	4	5	Avg
Precision	1.0	1.0	1.0	1.0	1.0	$P_i = 1.0$
Recall	0.8	0.8	1.0	1.0	1.0	$R_i = 0.92$
F-Measure	0.88	0.88	1.0	1.0	1.0	$F_i = 0.95$

Table 5.2: User-based evaluation (video 70)

From Table 5.2, we notice that the average metrics are taken into consideration when evaluating a video. The previous procedure is done for each of the 50 videos. Furthermore, the global values of precision, recall, and f-measure for the whole dataset are the average of the final results

of the 50 videos. Assuming that the results of each video V_i are denoted as P_i,R_i,F_i, for precision,recall and f-measure respectively, the global evaluation values for the complete dataset as seen in Equations 5.6, 5.7, 5.8.

$$P_g = (\sum_{i=1}^{i=50} P_i)/50 \tag{5.6}$$

$$R_g = (\sum_{i=1}^{i=50} R_i)/50 \tag{5.7}$$

$$F_g = (2 * P_g * R_g)/(P_g + R_g) \tag{5.8}$$

To calculate the global evaluation metrics, multiple experiments were conducted to better analyze the effect of each system parameter. In each experiment, parameters mentioned in 3.3 and 3.4 were varied. The parameters included the implemented clustering technique, the scene detection threshold and extracted features used in the multiple stages of processing.

Scene Det.	Sampling Rate	Clustering	Min Scene Length	P_g	R_g	F_g
HSV Hist.	2-FPS	K-Means	2-Sec	0.62	0.85	0.72
HSV Hist.	3-FPS	K-Medoids	1-Sec	0.69	0.8	0.74
HSV Hist.	3-FPS	K-Means	1-Sec	0.58	0.88	0.70
CNN.	3-FPS	K-Means	1-Sec	0.7	0.7	0.70

Table 5.3: User-based evaluation results

From the results in Table 5.3, we can infer that the average in recall each of the experiments is generally higher than the average precision. With the implemented methodology, and regardless of the specific parameters, most of the events are highlighted through the keyframe summary. Hence, extracted keyframes which indicate change of events but have not been frequently chosen by the annotating are most likely to be present in the system-generated summary. Hence, a decrease in precision is expected to occur.

5.4.2 Golden-Standard Evaluation

As previously mentioned, the vagueness of what the correct video summary should be results in differences between manually-created summaries in each video. Therefore, a golden-standard summary for each video was created in order to obtain a generalized benchmark. By using a golden-standard summary, a more valid, and less subjective ground-truth summary is created. Our criteria of creating a golden-standard summary by considering a keyframe to be relevant(a correct keyframe) in the case of its presence in the majority(3 or more) of the user summaries. Consequently, system-generated summaries were compared to a single summary, representing the 5 previously mentioned manually-created summaries for each video in the dataset. Figure 5.6 shows an example of creating a golden-standard summary from a set of manually-created summaries, and the keyframe matching with the system-generated summary that follows.

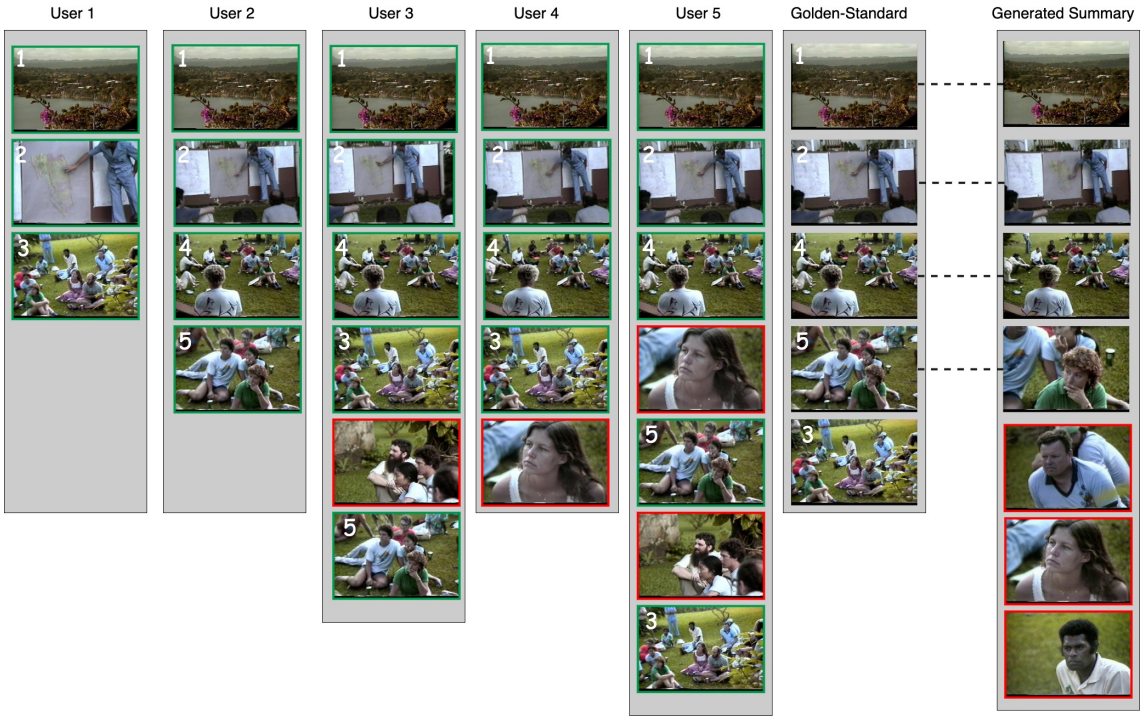

Figure 5.6: Golden-standard summary evaluation(video 61)

As mentioned , keyframes which are common between 3 or more summaries were chosen. In addition, the golden-standard summary as seen in Figure 5.6 is able to provide a more generalized user perspective, giving a less subjective comparison. In our case, since the VSUMM dataset did not provide a such benchmark, 50 golden-standard summaries were manually created.

Rather than considering P_i,R_i,F_i for an arbitrary test video V_i to be the average of comparing with each of the five users, we compare the generated summary directly with the corresponding golden-standard reference. Nevertheless, the global values of evaluation metrics for the whole VSUMM dataset remain to follow Equations 5.6, 5.7, 5.8. Table 5.4 shows the results of evaluating our approach with referring the the golden-standard summary for each video, the results are for the same experiments discussed in the user-based evaluation scheme.

Scene Det.	Sampling Rate	Clustering	Min Scene Length	P_g	R_g	F_g
HSV Hist.	2-FPS	K-Means	2-Sec	0.65	0.85	0.74
HSV Hist.	3-FPS	K-Medoids	1-Sec	0.69	0.8	0.74
HSV Hist.	3-FPS	K-Means	1-Sec	0.58	0.88	0.70
CNN.	3-FPS	K-Means	1-Sec	0.7	0.7	0.70

Table 5.4: Golden-standard evaluation results

We can observe that the behavior of both precision and recall has remained similar to that in the case of user-based evaluation. Nevertheless, with golden-standard summaries now capturing more events as the majority of users is represented, an increase in recall was allowed.

5.4.3 Discussion and Mitigation

To elaborate on our achieved results, we discuss the effects of each stage and its possible mitigation. Firstly, the preliminary preprocessing stage had further effects on its following stages. Choosing a critically low frame-rate such as 1 FPS caused relatively higher differences in adjacent frames and hence decreasing the effectiveness of scene detection. On the other hand, high frame-rates increased smoothness in frame transitions and made those between scenes less apparent. Frame-rate should therefore be set to both keep critical transitions while maintaining internal scene similarity.

Scene detection and scene clustering worked closely together in extracting keyframes. The possible errors of scene detection had an effect on clustering. Most importantly, finding a generalized threshold from which scenes were inferred was the main challenge, which led to slight errors in detected scenes. A part of mitigating this problem was proposing the clustering scheme; by doing so, multiple keyframes could be extracted from a scene holding within it other sub-scenes. However, with the main issue with clustering being the predefined value of k(number of cluster), came the possible mitigation of choosing the optimal value through the silhouette method. Nonetheless, possible imprecision in feature vectors used caused both inaccurate approximations of k and faulty redundancy removals. With this in mind, it was possible that resulting keyframes were either falsely removed, hence decreasing recall, or falsely duplicated, hence decreasing precision.

5.5 Comparison with Related work

In this section, we compare the results of our methodology to others from those who have tested on the OVP[34] dataset. To do so, we run both user-based and golden-standard evaluation on each of the publicly available results of previous work on video summarization. Using the avaiable outputs, we were able to compare our work with VSUMM(De Avila et al.[12]), DT(Mundur, Rao, and Yesha[13]), STIMO(Furini et al.[10]) and the OV(Open-Video) summaries by (DeMenthon, Kobla, and Doermann[11]). Figure 5.7 shows the keyframe summaries of the different approaches on the OVP[34] dataset for an arbitrary video, including our approach.

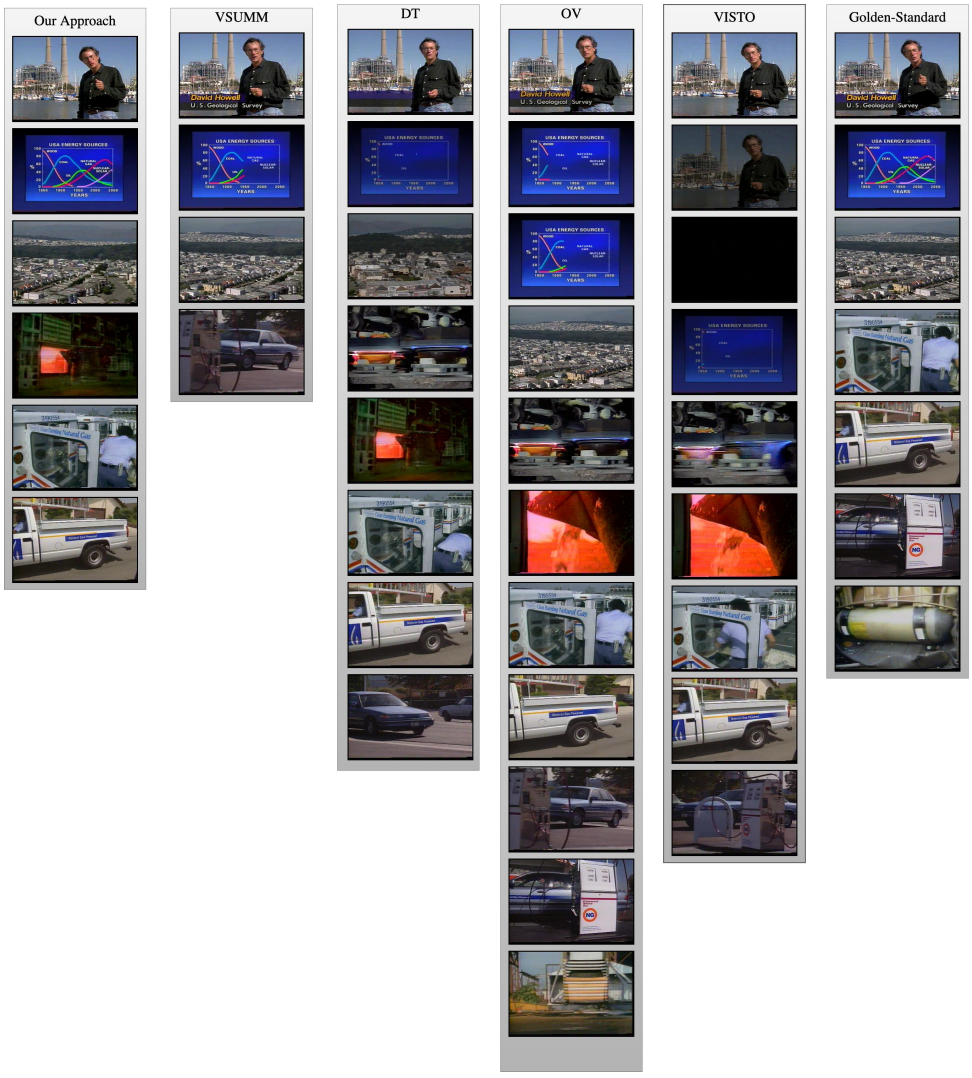

Figure 5.7: Summaries of different methods (v51)

To further compare between each method, Table 5.5 shows the differences in the used methodology for each of the mentioned previous works along with our approach.

Approach	Features	Keyframe Selection
VSUMM	HSV histogram(16-bins)	Frame closest to cluster centroid
DT	HSV histogram(256-bins)	Frame closest to cluster centroid
STIMO	HSV histogram and audio features	k-center/farthest point first
OV	luminance and chrominance of pixels	Finding centroid vectors in the curve
Our Approach	HSV histogram,CNN and BOVW Vector	K-Means and K-Medoids Centroids

Table 5.5: Main differences between related work

5.5.1 User-Based Evaluation

Using the discussed evaluation technique in 5.3, we compare the results of the our methodology with the previous approaches. First, the user-based evaluation method is applied as discussed in 5.4.1. The average precision, recall and f-measure values are shown in Table 5.6 for each method.

Approach	P_g	R_g	F_g
VSUMM2	0.74	0.84	0.79
VSUMM1	0.77	0.69	0.71
DT	0.65	0.51	0.57
STIMO	0.57	0.69	0.62
OV	0.64	0.71	0.67
Our Approach	0.7	0.8	0.75

Table 5.6: Related work comparison (User-Based)

5.5.2 Golden-Standard Evaluation

In addition to evaluating on the regular user-based annotations provided by the VSUMM dataset, we also evaluate the previous approaches using the chosen golden-standard summaries for each video. The results of golden standard evaluation are shown in Table 5.7.

Approach	P_g	R_g	F_g
VSUMM2	0.79	0.87	0.83
VSUMM1	0.82	0.72	0.77
DT	0.69	0.51	0.59
STIMO	0.59	0.68	0.63
OV	0.67	0.71	0.69
Our Approach	0.65	0.85	0.74

Table 5.7: Related work comparison (Golden-Standard)

Chapter 6

Conclusion, Limitations and Future Work

6.1 Conclusion

In this project, video summarization was implemented through extracting keyframes which highlight a video's main events. Through our contribution, the use of high level CNN features, along with global color descriptors was put to use to divide the video into scenes. Furthermore, the final keyframe summary was collected by processing each scene separately, and nominating representative and unique shots through the clusters of each scene. To build on the used image descriptors, we proposed a keyframe summary evaluation technique, based on finding the common keyframes between a reference and generated summary.

To make use of the extracted keyframes for each video, we showed how they can be used as the base of a video browsing application. The application was able to receive video content from users in order to increase and develop the used video storage and index. Consequently, videos were found and identified by their extracted keyframe insights.

6.2 Limitations

Multiple limitations where faced while implementing and evaluating the proposed methodology. For instance, the main elements which were used in both scene detection and scene processing were prone to various levels of error. The use of both types of image descriptors led to difficulties at times in detecting significant changes of events through the video. Specifically, solely relying on visual descriptors in determining scenes and important moments was not sufficient. Furthermore, the same issues were found in determining the exact amount of relative keyframes to be extracted from a certain video scene. Although the main assumption of relying on cluster purity metrics such as the silhouette was found to be effective, errors in correct predictions led to keyframe redundancy within the same scene. In addition, the choice of relevant keyframes based on its uniqueness within a scene, with the considered features, may not always fit with a user's subjective choice, as shown in Figure 6.1.

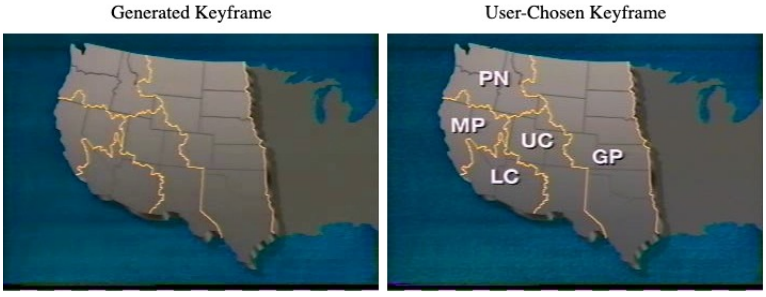

Figure 6.1: Generated vs. subjectively-chosen Keyframe

In addition to the previously discussed limitation, objectively evaluating keyframe extraction faced various challenges. For example, the scarcity of publicly distributed outputs for previous works led to limiting the choices of datasets to evaluate our work on. Hence, the VSUMM dataset, which was chosen due to its available results and annotations, faced major issues in video quality and content. As demonstrated previously, the subjectivity in the user-made summaries also made determining the quality of generated summaries a hard task. Despite using alternatives such as the golden-standard, which was a universal summary generation strategy, its agreement with the average user-based summaries reached 84%. Therefore, the ability to reach a satisfactory summary for all users was limited. Finally, since automatically determining if two images were a match was prone to error and is unable to reach a human's perspective, precisely evaluating a summary with its reference was also subject to challenges.

6.3 Future Work

Much can be done as future work regarding our our approach in video summarization and its use in a keyframe-based application. With the concept of image matching being a fundamental component in both keyframe extraction and its evaluation method, we aim to develop a more accurate and human-similar image match detection model. Furthermore, regarding the stages of keyframe extraction, future work includes a more thorough investigation into visual features to increase the system's performance and computation time. More importantly, a contribution can be done by collecting a new and higher quality video summarization dataset, and providing its corresponding user-based and golden-standard annotations. Finally, we plan on enhancing the application by enlarging and improving its video storage repository, and improving the keyframe-based video search and making it a more user-customized environment.

Bibliography

[1] *Omnicore: YouTube by the Numbers*. https://www.omnicoreagency.com/youtube-statistics/. Accessed: 2020-2-23.

[2] Silvia Pfeiffer et al. *Abstracting Digital Movies Automatically*. Tech. rep. 1996.

[3] Truongand and Venkatesh. "Video Abstraction: A Systematic Review and Classification". In: *ACM Trans. Multimedia Comput. Commun. Appl.* 3.1 (Feb. 2007), 3–es. ISSN: 1551-6857. DOI: 10.1145/1198302.1198305. URL: https://doi.org/10.1145/1198302.1198305.

[4] Wayne Wolf. "Key frame selection by motion analysis". In: *1996 IEEE International Conference on Acoustics, Speech, and Signal Processing Conference Proceedings*. Vol. 2. IEEE. 1996, pp. 1228–1231.

[5] Berthold Horn and Brian Schunck. "Determining Optical Flow". In: *Artificial Intelligence* 17 (Aug. 1981), pp. 185–203. DOI: 10.1016/0004-3702(81)90024-2.

[6] Engin Mendi, Hélio B Clemente, and Coskun Bayrak. "Sports video summarization based on motion analysis". In: *Computers & Electrical Engineering* 39.3 (2013), pp. 790–796.

[7] Bruce D Lucas, Takeo Kanade, et al. "An iterative image registration technique with an application to stereo vision". In: (1981).

[8] Genliang Guan et al. "Keypoint-based keyframe selection". In: *IEEE Transactions on circuits and systems for video technology* 23.4 (2012), pp. 729–734.

[9] Tiecheng Liu and John R Kender. "Optimization algorithms for the selection of key frame sequences of variable length". In: *European conference on computer vision*. Springer. 2002, pp. 403–417.

[10] Marco Furini et al. "STIMO: STIll and MOving video storyboard for the web scenario". In: *Multimedia Tools and Applications* 46.1 (2010), p. 47.

[11] X DeMenthon, A Kobla, and X Doermann. "Video Summarization by Curve Simplification". In: (Oct. 1999). DOI: 10.1145/290747.290773.

[12] Sandra Eliza Fontes De Avila et al. "VSUMM: A mechanism designed to produce static video summaries and a novel evaluation method". In: *Pattern Recognition Letters* 32.1 (2011), pp. 56–68.

[13] Padmavathi Mundur, Yong Rao, and Yelena Yesha. "Keyframe-based video summarization using Delaunay clustering". In: *International Journal on Digital Libraries* 6.2 (2006), pp. 219–232.

[14] Rameswar Panda, Sanjay K Kuanar, and Ananda S Chowdhury. "Scalable video summarization using skeleton graph and random walk". In: *2014 22nd International Conference on Pattern Recognition*. IEEE. 2014, pp. 3481–3486.

[15] Ebrahim Asadi and Nasrolla Moghadam Charkari. "Video summarization using fuzzy c-means clustering". In: *20th Iranian Conference on Electrical Engineering (ICEE2012)*. IEEE. 2012, pp. 690–694.

[16] EJY Cayllahua-Cahuina, G Cámara-Chávez, and David Menotti. "A static video summarization approach with automatic shot detection using color histograms". In: *Proceedings of the International Conference on Image Processing, Computer Vision, and Pattern Recognition (IPCV)*. The Steering Committee of The World Congress in Computer Science, Computer ... 2012, p. 1.

[17] Wen-Sheng Chu, Yale Song, and Alejandro Jaimes. "Video co-summarization: Video summarization by visual co-occurrence". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3584–3592.

[18] Antti E Ainasoja et al. "Keyframe-based Video Summarization with Human in the Loop." In: *VISIGRAPP (4: VISAPP)*. 2018, pp. 287–296.

[19] Genliang Guan et al. "A top-down approach for video summarization". In: *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 11.1 (2014), p. 4.

[20] Edward JY Cayllahua Cahuina and Guillermo Camara Chavez. "A new method for static video summarization using local descriptors and video temporal segmentation". In: *2013 XXVI Conference on Graphics, Patterns and Images*. IEEE. 2013, pp. 226–233.

[21] *Lumen5: YouTube Video Dimension and Size*. `https://lumen5.com/learn/youtube-video-dimension-and-size/`. Accessed: 2020-2-23.

[22] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International Journal of Computer Vision* 60 (2004), pp. 91–110.

[23] Christian Szegedy et al. "Rethinking the Inception Architecture for Computer Vision". In: *CoRR* abs/1512.00567 (2015). arXiv: `1512.00567`. URL: `http://arxiv.org/abs/1512.00567`.

[24] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[25] Peter Rousseeuw. "Rousseeuw, P.J.: Silhouettes: A Graphical Aid to the Interpretation and Validation of Cluster Analysis. Comput. Appl. Math. 20, 53-65". In: *Journal of Computational and Applied Mathematics* 20 (Nov. 1987), pp. 53–65. DOI: `10.1016/0377-0427(87)90125-7`.

[26] G. Bradski. "The OpenCV Library". In: *Dr. Dobb's Journal of Software Tools* (2000).

[27] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[28] François Chollet et al. *Keras*. `https://github.com/fchollet/keras`. 2015.

[29] *Google Cloud Platfrom*. `https://cloud.google.com/`. Accessed: 2019-12-20.

[30] *MongoDB*. `https://www.mongodb.com/`. Accessed: 2019-12-10.

[31] *Elasticsearch*. `https://www.elastic.co/`. Accessed: 2019-12-10.

[32] *ReactJS*. `https://reactjs.org/`. Accessed: 2019-12-10.

[33] *Flask Web Framework*. `http://flask.palletsprojects.com/en/1.1.x/`. Accessed: 2019-12-10.

[34] *Open Video Project Dataset*. `https://sites.google.com/site/vsummsite`. Accessed: 2019-9-15.

[35] Jiebo Luo, Christophe Papin, and Kathleen Costello. "Towards extracting semantically meaningful key frames from personal video clips: from humans to computers". In: 19 (2009), pp. 289–301.

Draft 1

ORIGINALITY REPORT

5%	2%	3%	2%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Lecture Notes in Computer Science, 2014. Publication	<1%
2	Padmavathi Mundur, Yong Rao, Yelena Yesha. "Keyframe-based video summarization using Delaunay clustering", International Journal on Digital Libraries, 2006 Publication	<1%
3	Submitted to University of Sydney Student Paper	<1%
4	epdf.tips Internet Source	<1%
5	Studies in Computational Intelligence, 2016. Publication	<1%
6	Submitted to The Hong Kong Polytechnic University Student Paper	<1%
7	www.ece.cmu.edu Internet Source	<1%
	www.ijetcse.com	