BRNO UNIVERSITY OF TECHNOLOGY
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

FACULTY OF INFORMATION TECHNOLOGY
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

VIDEO SUMMARIZATION
SUMARIZACE VIDEA

BACHELOR'S THESIS
BAKALÁŘSKÁ PRÁCE

AUTHOR **ROMAN ŠTAFL**
AUTOR PRÁCE

SUPERVISOR **Ing. VÍTĚZSLAV BERAN, Ph.D.**
VEDOUCÍ PRÁCE

BRNO 2022

Brno University of Technology
Faculty of Information Technology

Department of Computer Graphics and Multimedia (DCGM) Academic year 2021/2022

Bachelor's Thesis Specification

Student: **Štafl Roman**
Programme: Information Technology
Title: **Video Summary**
Category: Image Processing

Assignment:

1. Get acquainted with image and video processing methods focused on image feature extraction and optical flow.
2. Design a system that will extract image features from video frames, evaluate the activity in adjacent frames, search for similar frames in the video and generate output with only representative video parts.
3. Prepare a suitable annotated dataset.
4. Implement the system using the available libraries.
5. Evaluate the system on a prepared data set.
6. Document the methods, design, implementation, dataset and results in technical report.

Recommended literature:

- M. Sonka, V. Hlaváč, R. Boyle. *Image Processing, Analysis, and Machine Vision*, CL-Engineering, ISBN-13: 978-0495082521, 2007.
- Gary R. Bradski, Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*, ISBN 10: 0-596-51613-4, September 2008.

Detailed formal requirements can be found at https://www.fit.vut.cz/study/theses/

Supervisor: **Beran Vítězslav, Ing., Ph.D.**
Head of Department: Černocký Jan, doc. Dr. Ing.
Beginning of work: November 1, 2021
Submission deadline: May 11, 2022
Approval date: November 1, 2021

Abstract

Long video recordings from surveillance cameras often need to be meticulously reviewed. However, this is an impractical task for a human, as it would not be time-efficient. The person would be prone to a significant amount of potential mistakes due to suboptimal attention span and limited ability to spot every error. An application facilitating a search for changes in a video would save hours of labour-intensive work, which will be the main focus of this thesis.

Abstrakt

Dlouhé videozáznamy z bezpečnostních kamer je často třeba pečlivě zkontrolovat. To je však pro člověka nepraktický úkol, neboť by to nebylo časově efektivní. Lidé jsou náchylní ke značnému množství potenciálních chyb kvůli omezené pozornosti a schopnosti odhalit každou událost. Aplikace umožňující vyhledávání změn ve videu by ušetřila hodiny intenzivní práce, což bude hlavním předmětem tohoto díla.

Keywords

video summarization, footage analysis, image processing, video processing

Klíčová slova

sumarizace videa, analýza záznamu, zpracování obrazu, zpracování videa

Reference

ŠTAFL, Roman. *Video summarization.* Brno, 2022. Bachelor's thesis. Brno University of Technology, Faculty of Information Technology. Supervisor Ing. Vítězslav Beran, Ph.D.

Video summarization

Declaration

I hereby declare that this Bachelor's thesis was prepared as an original work by the author under the supervision of Ing. Vítězslav Beran Ph.D. I have listed all the literary sources, publications and other sources, which were used during the preparation of this thesis.

.....
Roman Štafl
May 6, 2022

Contents

Chapter 1

Introduction

The modern world offers countless ways of using video capture devices. Surveillance cameras, car dashboard cameras, or any other devices that record video for an extended period of time generate large amounts of footage. It would mean countless hours spent reviewing the footage for a human to analyze this data, let alone to edit it. On top of that, placing a task of reviewing hours of footage on a human would bring in a big possibility of human error, whether it be them being tired and not paying enough attention or being distracted and overlooking details among other things.

This has led to a drastic increase in demand for video processing tools and an effective way of storage, as people all over the world are capturing immense amounts of video data on a daily basis. For this reason, video summarization is an important topic.

Video summarization is a way to summarize the contents of a video by selecting the most informative and relevant parts and presenting them to the user. Video summarization is able to have varying requirements depending on the user. Among the most general aims of video summarization are reducing the space required to store the data, increasing the speed at which humans are able to analyze the footage or generating a short overview of the content of the video. This thesis focuses on design and implementation of a video summarization application, which would help with analysis and storage of mainly surveillance camera footage.

Chapter 2

Theoretical Background

This chapter introduces the reader to the basic concepts necessary for understanding the methods and techniques used in the following chapters. Aside from explaining the methods and techniques chosen in this work, this chapter also explores other possible approaches.

2.1 Video Structure

Videos and their processing is the main focus point of this thesis, therefore fully understanding the manner in which videos are structured is essential before exploring the more complex subjects mentioned later on. Videos are structured into scenes, shots and frames, as is depicted in the illustration in figure 2.1. A frame is a single still image taken at a point in time. A shot is a set of continuously recorded frames, which can be several seconds or minutes long. A scene is composed of several shots. In the case of narrative films, sequences are used, which are composed of scenes. [6]

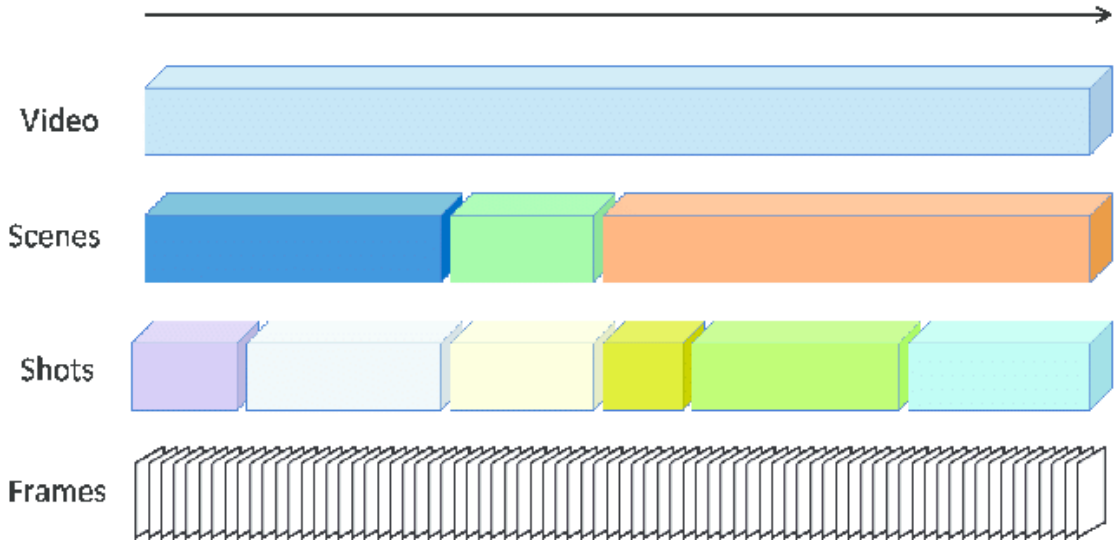

Figure 2.1: An illustration of video structure viz. [6]

2.2 Video Summarization

Video summarization is a process that summarizes the contents of a video by selecting the most informative and relevant parts and presenting them to the user typically in the form of a shorter version of the recording. An illustration is presented in figure 2.2. Video summarization not only solves the problem of reducing the space required to store long videos, but also greatly increases the speed at which humans are able to analyse them. This comes useful for example in the context of looking through surveillance records.

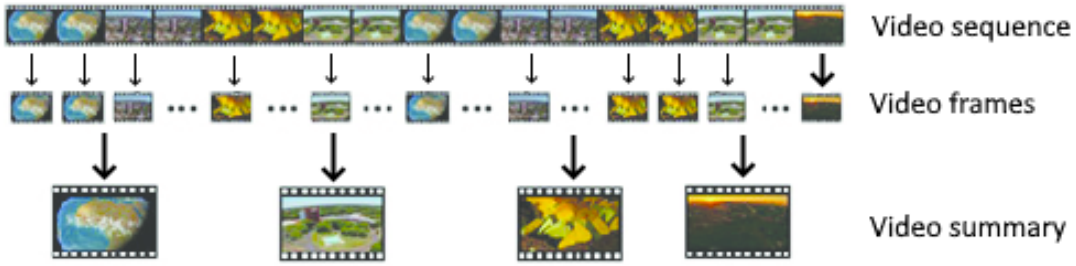

Figure 2.2: Video Summarization illustration viz. [2]

Types of Video Summarization

Videos can mainly be divided into two types, a static camera video or a dynamic camera video. As such, the type of the summary depends on the nature of the video on which the summary is done. Videos produced by surveillance, traffic and dashboard cameras are captured by static cameras and together with other footage similar in nature fall under static camera videos. Nearly all of other videos with a lot of camera movement and activity would classify as dynamic camera videos. For instance a video of a television show would be classified as a dynamic camera video and therefore a dynamic camera summary would be made out of them.

A static camera summary, also called key frame based video summarization or a storyboard, is a set of selected keyframes. It is made up of a number of frames representing every shot.

The idea of dynamic summarization, also called video skimming, is a short video composed by concatenating short, most informative scenes from the original video, while still retaining the important content.

Video Summarization methods

Several methods have been developed for assessing video content and generating a precise video summary. Each of these methods is unique in their criteria of choosing the shots for the final summary. [7, 8, 15] Generally, these methods are able to be classified into three categories based on their characteristics.

Feature based

Videos contain various types of information. This information is able to be described with image features, such as colour, motion, event, etc. After extracting these features, they are able to be processed and evaluated and a video summary can be made by selecting shots based on the most interesting features. An illustration is able to be seen in figure 2.3. [8] This approach is especially relevant when considering static camera footage, which is often produced by surveillance cameras. In these videos it is typical that the majority of the shots are the same or very similar, with the scene barely changing or not changing at all, and the few remaining shots are instances of activity. For this reason an approach that finds these instances by searching for motion, changes in colour or other events is applicable, that is assuming these instances of activity are what the user is looking for in the final summary. [8] Working with image features often requires a lot of computational time, as large amounts of data need to be processed, which poses a challenge in creation of a video summarization system.

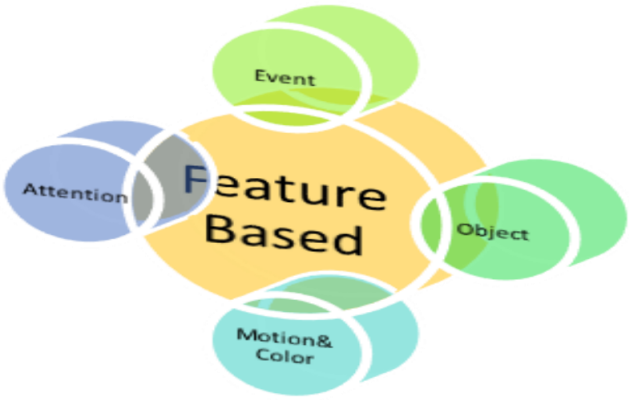

Figure 2.3: Classification of feature based approaches viz. [8]

Distribution based

In some cases, taking the content of the video into account would not be of much help, or would be only the secondary element when creating an interesting summary. A good example of this case are videos captured by dynamic cameras. In dynamic camera footage, it is typical to register a significant amount of motion, change in colour and the scenes changing quite often. While it is still possible to extract features from a video of this sort, in a video summary created from a dynamic footage the features are usually not the main concern. Instead, a lot of importance is placed on the story flow and rhythm in dynamic summaries. [15, 7] A summary containing evenly distributed shots covering all parts of

the video is more desirable. In this approach the system does not work with features, therefore majority of the tasks requiring a lot of computational time is skipped, making this a relatively fast method. A system using a uniform distribution of key shots has been tested in paper [14].

Distribution and Feature based

When maximizing the information in form of interesting shots is relevant in the summary, but even distribution is also desired, a combination of Distribution based and Feature based approaches can be used. We take interesting content in the video into consideration by extracting image features and processing them, finding the most interesting shots. We then pick the most relevant shots in such a way that we get a good distribution of shots from the video in the final summary. [8, 15]

2.3 Image Feature Extraction

Generally, a feature is any piece of information which is relevant for solving the computational task related to a certain application.

As this work focuses on video summarization, the main type of features that are relevant for this work are image features. Image features may be specific structures in the image such as points, edges or objects. They may also be the result of a general neighborhood operation or feature detection applied to the image. Generally, features are able to be described as interesting parts of an image, as illustrated in figure 2.4. [13, 24]

Image processing has a very sophisticated collection of features. The feature concept is generic and the choice of features in a particular computer vision system may be highly dependent on the specific problem at hand.

Feature Extraction is a dimensionality reducing process where an initial set of raw data is given a more manageable form for processing. The most important characteristic of these large data sets is that they have a large number of variables. A lot of computing resources are required to process these variables, which is why feature extraction is introduced to get the best feature from those data sets by selecting and combining variables into features, reducing the amount of data. These features are easier to process and still retain important information about the original data set.

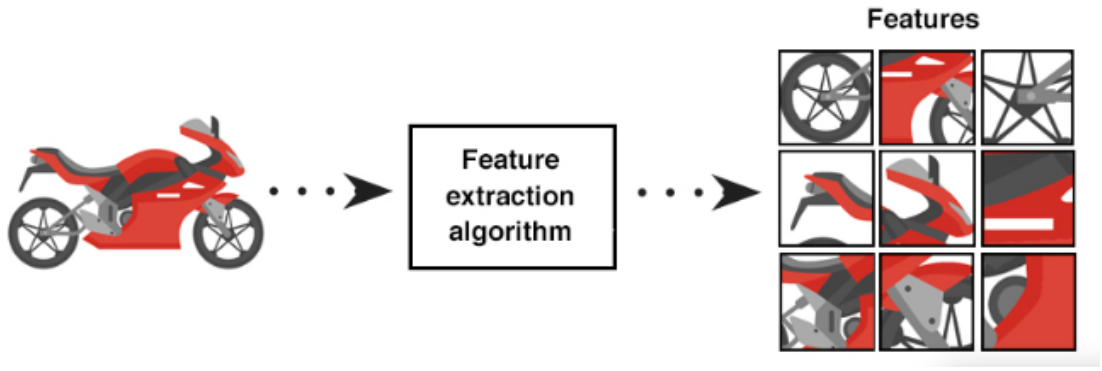

Figure 2.4: Feature extraction illustration viz. [23]

Once features have been detected, a local image patch around the feature is able to be extracted. This extraction may involve considerable amounts of image processing. The result is known as a feature descriptor or feature vector. A feature descriptor is a representation of an image or an image patch that simplifies the image by extracting useful information and throwing away extraneous information. In addition to such attribute information, the feature detection step by itself may also provide complementary attributes, such as the edge orientation and gradient magnitude in edge detection and the polarity and the strength of the blob in blob detection. [10]

2.4 Selected Image Features and Metrics

This section examines a chosen number of image feature types and descriptors possible to use in a video summarization system and their advantages and disadvantages.

Edges

A concise definition would say that edge features are points in an image where there is an edge between two regions. A very basic definition of edges is that they are a representation of change in luminosity levels of the neighbouring pixels or sets of points in the image which have a strong gradient magnitude. An edge is able to take virtually any shape. One method to detect this change is to use gradients. They will not only detect the change but also the direction of the change. This result of this operation is a large number of pixels that can all possibly be edges. Processing all of these pixels would take a lot of time, which is why additional methods have been developed to cut down the number of chosen pixels. [13]

In the ideal case, the result of applying an edge detector to an image may lead to a set of connected curves that indicate the boundaries of objects, the boundaries of surface markings as well as curves that correspond to discontinuities in surface orientation. The result of an edge detection algorithm is able to be seen in figure 2.5.

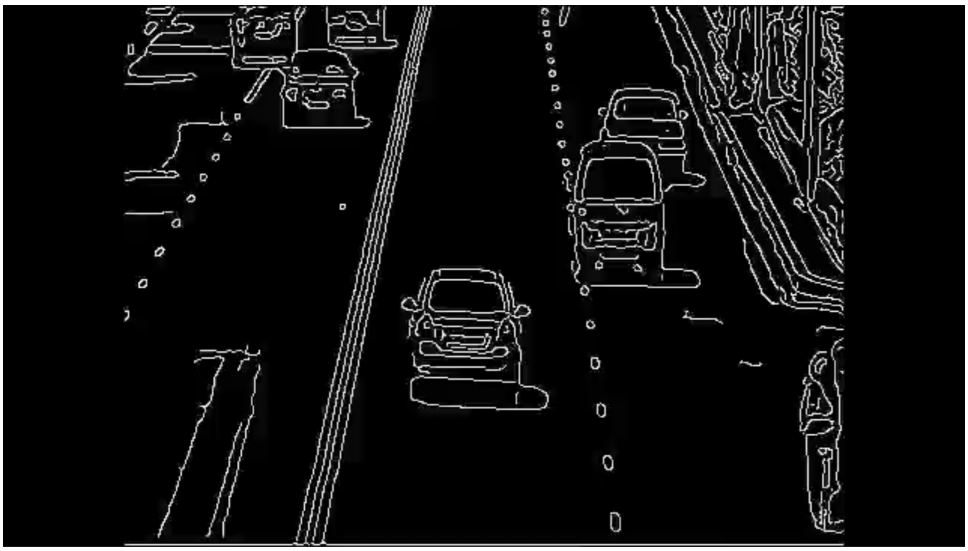

Figure 2.5: Edge detection example

The edge histogram descriptor captures the spatial distribution of the edges. The process of edge orientation histogram creation is nicely explained in a paper on colour and texture descriptors. [17] The image is divided into 16 parts and local histograms are computed for these parts. Each histogram has five bins for each category of edge orientation. The categories of the orientations in question are horizontal, vertical, 45°, 135° and isotropic. Each sub-image is then further divided into blocks and the blocks are then divided into even smaller 4 pixel blocks, as can be seen in the illustration of the process in figure 2.6. Edge operators are then applied to the 4 pixel blocks. The results are then compared to a certian minimum threshold and those blocks that exceed that threshold are used in computing the histogram. [17]

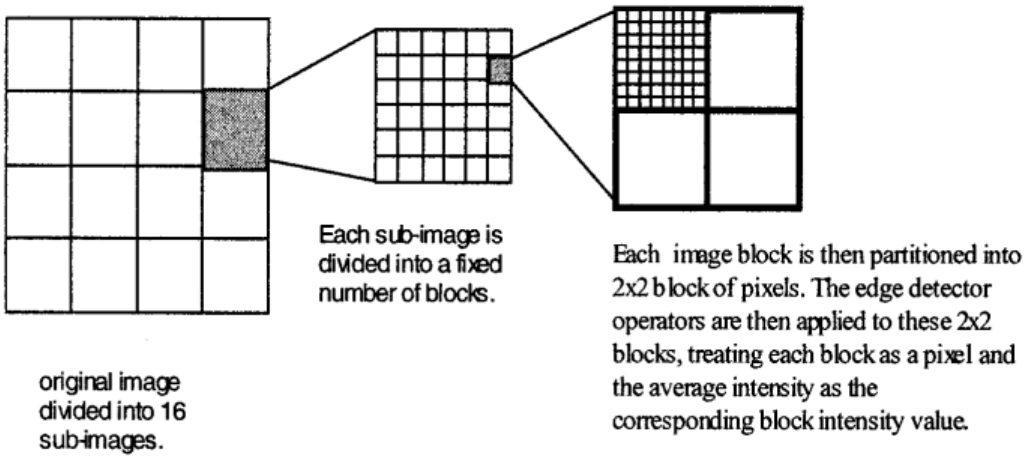

Figure 2.6: Computing of the edge histogram descriptor viz. [17].

Corners

Detection of corners, also known as interest points, in images has been a long-standing problem in computer vision, having applications in a wide variety of pattern recognition and computer vision tasks such as feature detection. [4]

A corner can be defined as the intersection of two edges. A corner can also be defined as a point for which there are two different edge directions in the local neighbourhood of the point. Corners are the important features in the image, and they are generally termed as interest points which are invariant to translation, rotation, and illumination.

One of the earliest corner detection algorithms is the Moravec corner detection algorithm that defines a corner to be a point with low self-similarity. The algorithm tests each pixel in the image to see if a corner is present, by considering how similar a patch centered on the pixel is to nearby, largely overlapping patches. [19] One of the most popular approaches for corner detection is the Harris corner detector for which there now exist many variants. [25] It has improved upon Moravec's corner detector by considering the differential of the corner score with respect to direction directly, instead of using shifted patches. [9] An example of corner detection is able to be seen in figure 2.7.

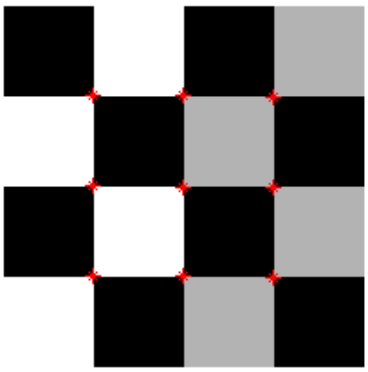

Figure 2.7: Corner detection example

Colour Features

Colour is one of the most important features of images. Colour features are defined based on a number of colour spaces or models. RGB, HSV, LCh, CMYK and various other colour spaces are known in literature. [22] After specification of the colour space, colour image features can be extracted from the image. One of the known colour features is a colour histogram. [1, 21]

A colour histogram in image processing represents the distribution of colours in three-dimensional RGB space. For a typical imaging system with 8 bits for each colour band, there are 256^3 bins, spanning most of the possible colours in the image, into which a pixel may fall. The number of image pixels for every bin is then counted and a histogram of colours is produced. An illustration of a colour histogram can be seen in figure 2.8.

An advantage to colour histograms is that they are intuitive and more importantly simple to compute, as all it takes is to assign every pixel value to it's corresponding bin. The main drawback of colour histograms in terms of video summarization is that they are dependent only on the distribution of colours in the image and they ignore shapes and textures in the object. Colour histograms can potentially be identical for two images with different texture and shape content, but happen to contain identical distribution of colour.

Another problem is that colour histograms have high sensitivity to noisy interference such as lighting intensity changes and quantization errors, as they heavily change the distribution of colours in the image. [21]

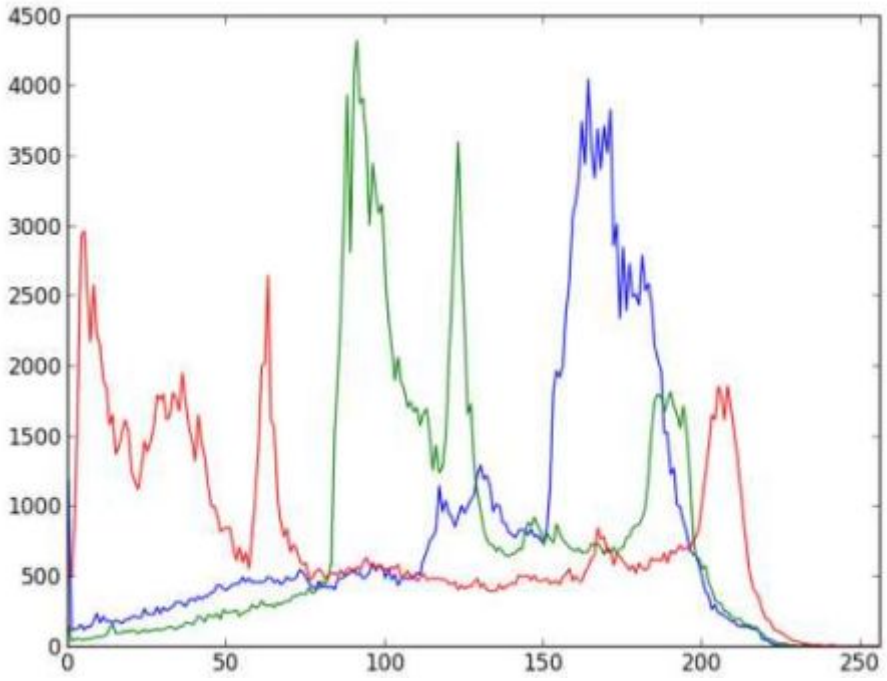

Figure 2.8: RGB Histogram example

Histogram of Oriented Gradients

The histogram of oriented gradients, or HOG for short, is a feature descriptor used in computer vision and image processing and is often used to extract features from image data. This method is similar to edge orientation histograms, as it also focuses on the structure or shape of objects. The difference is that while histogram of oriented gradients detects an edge, it also finds the direction of the edge. This is done by extracting the gradient and orientation of the edges.

The image is divided into smaller regions or cells, on which the orientations, also called localized portions, are compiled. An example of this is able to be seen in figure 2.9. For each of these regions, a histogram is generated using the gradient magnitudes and orientations. The resulting histogram is essentially a vector of a number of bins corresponding to angles 0 to 160. For improved accuraccy, the local histograms are normalized by calculating a measure of the intesity across a larger section of the image, called a block, that contains a number of cells, and after that using this value to normalize all cells within the block. The normalization results in better invariance to changes in illumination and shadowing.

The descriptor of this method is the concatenation of the histograms calculated for the individual cells. There are a few known advantages the HOG descriptor has over other descriptors. As it operates on local regions of the image, it is invariant to geometric and photometric transformations, except for object orientation.

In many feature detectors, the first calculation step is the normalization of colour and gamma values. In the original paper it is pointed out that in the case of HOG descriptor,

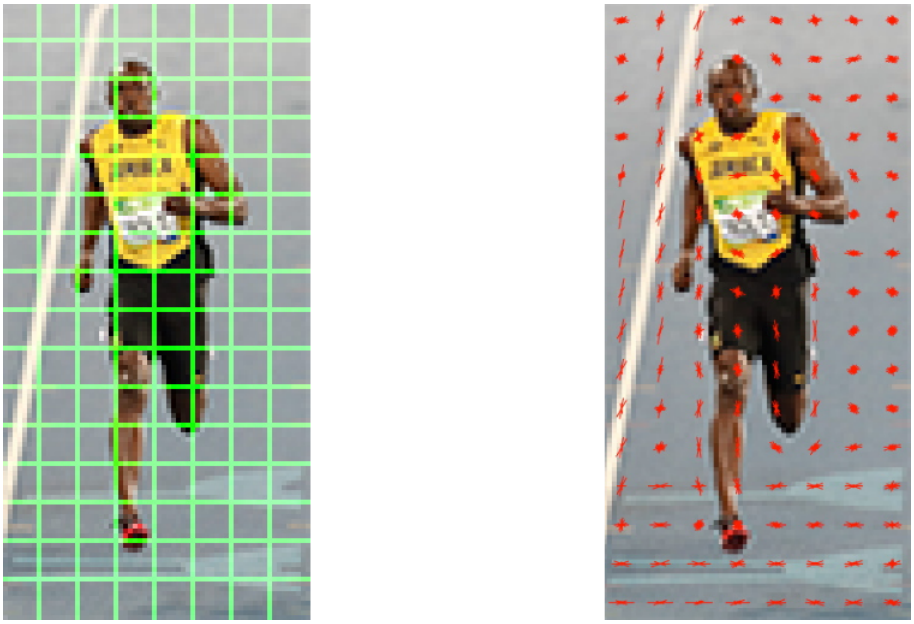

Figure 2.9: Division of an image into cells and HOG descriptor visualization viz. [16]

colour and gamma normalization has only a modest effect on performance for the most part, because the subsequent descriptor normalization achieves similar results. [5] However, the original paper focuses on human detection in images, therefore I have decided to not omit this step in the process of calculating the descriptor for the purpose of video summarization.

Next step of the calculation is computing the gradient values. The most common method is to apply a one dimensional centered point discrete derivative mask in one or both of the horizontal and vertical directions. The creators of the original paper found no performance benefit in using more complex masks.

Figure 2.10: 1-D horizontal and vertical filter kernels

The next step of the process is generating the cell histograms. Each pixel within a cell contributes to the value of a orientation-based histogram channel based on the values received from the gradient computation. The histogram channels are evenly spread over 0 to 180 degrees or 0 to 360 degrees, depending on whether they are signed or unsigned and they are divided into a number of channels. In experiments performed in the original HOG paper, the best performance was achieved with 9 histogram channels together with unsigned gradients. The value that every pixel contributes is the gradient magnitude.

Gradients of an image are sensitive to overall lighting. By making the image darker by dividing all the pixel values in half, the gradient magnitude will change by half and so will the histogram values. To account for changes in illumination and shadowing, the gradient strengths are locally normalized, which requires grouping the cells together into larger connected blocks. Two main geometries for the blocks are usually followed: rectangular and circular. As these blocks typically overlap, each cell contributes to more blocks at once. After that the descriptor is formed by concatenating the components of the normalized cell histograms from all of the block regions. An image summary of the steps of the HOG algorithm is visualized in figure 2.11.

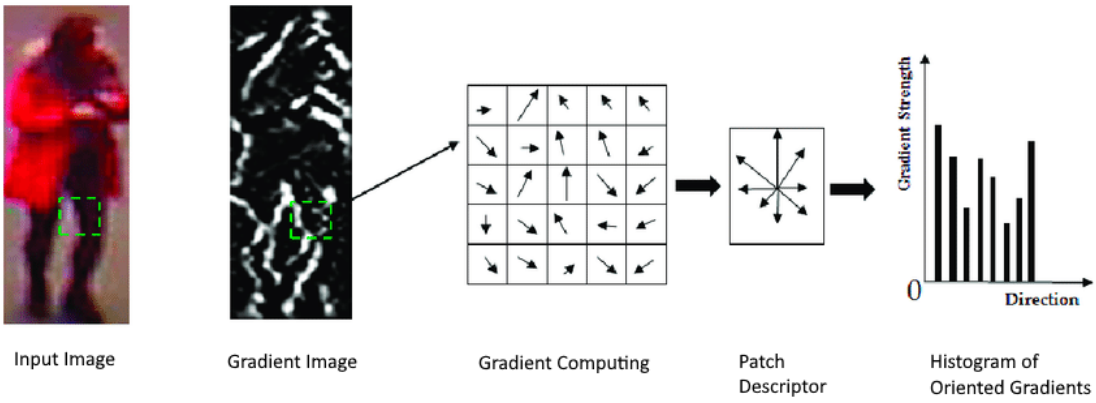

Figure 2.11: HOG methodology viz. [20]

2.5 Feature Comparison

For the purpose of comparing image features highlighted in section 2.4, distance calculating methods are able to be used. This section exposes different distance methods possible for the comparison of image features in their varying structures, for instance histograms.

In mathematics, the Euclidean distance between two points in Euclidean space is the length of a line segment between the two points. It is able to be calculated from the Cartesian coordinates of the points using the Pythagorean theorem, therefore occasionally being called the Pythagorean distance. The distance between two objects that are not points is usually defined to be the smallest distance among pairs of points from the two objects. For the purposes of calculating the distance of two histograms, equation defined in figure 2.12 where `u` and `v` are one dimensional vectors and `w` is an optional parameter that can be used to change the weight of the numbers in the vectors, is used.

$$||u - v||_2$$
$$\left(\sum (w_i |(u_i - v_i)|^2)\right)^{1/2}$$

Figure 2.12: Euclidean distance [11]

Manhattan distance or the City block distance is calculated as the distance along the x axis plus the distance along the y axis, which is similar to the way you move in a city (like Manhattan) where you have to move around the buildings instead of going straight through. For one dimensional arrays, vectors, or in our case histograms it can be defined as in figure 2.13 where `u` and `v` are one dimensional vectors.

$$\sum_i |u_i - v_i|$$

Figure 2.13: City block distance [11]

Chebyshev distance is a metric defined on a vector space where the distance between two vectors is the greatest of their differences along any coordinate dimension. For one dimensional arrays it can be defined as in figure 2.14 where `u` and `v` are one dimensional vectors. [18]

$$\max_i |u_i - v_i|$$

Figure 2.14: Chebyshev distance [11]

Figure 2.15: Visual difference of the three distance methods in 2-D space viz. [3]

Chapter 3

Problem Definition and Proposed Solution

3.1 Problem Structure

The whole point of video summarization begins with a user whose intention is to analyse one or several long videos. Doing this without any help would take an immense amount of time and would carry a high possibility of human error. The problem does not end there, as some users require to keep these videos archived. Without any editing, archiving large amounts of these long videos in their raw form places a burden of requiring immense amount of storage space on the user.

For this reason, it would be beneficial to people to have a solution to this problem. A system that automates video analysis and ultimately helps them save time, energy and storage space, while also reducing the possibility of human error during the analysis.

Ideally, a video summarization system allows the user to input a video, extracts all shots from the video that could be of importance for the user and generates a shorter version of the video, excluding all of the unimportant shots that bring no useful information to the user. The user would be left with a summary, much shorter and smaller in size than the original video, yet still retaining all the important shots the user could need.

However, creating a video summarizing application that fits the needs of every user would be impossible, as every user has a unique idea of what is important to extract from their video footage, or requires a video summary for a different purpose. For instance, for a user whose intention is to review long surveillance camera footage in order to search for suspicious behavior, a system that extracts every shot with even a hint of activity would be suitable. Alternatively, for a user whose intention is to create a slideshow to present the contents of a long video as quickly as possible, a well distributed collection of shots from every scene would be a much more desirable outcome of the system.

For the purpose of generating a suitable summary, this project needs to allow the user to select the target video for summarization. Afterwards, the video input by the user must be processed and interesting shots need to be detected. The system then needs to determine which of the shots are the most interesting and finally the chosen shots need to be extracted from the video and concatenated to form the final summary. Assuming the user also cares about the distribution of shots throughout the video, the system will not select shots for the final summary solely based on how interesting they are, but also based on their coverage of the original video in the final summary.

3.2 System requirements

To cover all of the defined problems with this solution, firstly the system needs to be able to analyze an input video and extract image features for each frame. I will address this part as the extraction phase. The system then determines which shots to include in the summary by comparing the image features and evaluating the distance between the most interesting frames, so that even distribution of chosen shots is taken into consideration. I will address this part as the analysis phase. After analysis, the data representing the chosen shots are saved and ready for the summary generator. The file with the chosen shots is passed into the summary generator which generates the final summary video.

3.3 System phases

In the whole summarization process, the system would go through several phases. These phases would handle all the steps necessary for video summarization, from the beginning, where the user inputs a video, to the end, where the system provides a summarized version of the original video to the user.

1. Preprocessing phase - To extract image features from the video frames, preprocessing would be done to reduce the computational cost of the following phases. Each frame would be resized to a uniform size and transformed to a grayscale colour-space.

2. Extraction phase - This phase of the system would focus on the extraction of image features from the input video. For that, the system would need to handle extracting a suitable image feature type from each of the frames, and have a suitable way to save the data.

3. Analysis phase - The extracted image features would then need to be analyzed. A score needs to be given to each frame that highlights an interesting point in the video. The value of the score is determined by how interesting the shot is. To also take even distribution of shots into account, another score indicating the distance from other shots is introduced. The system needs to be robust and detect corrupted frames in the original footage to not include them in the summarized video. The two scores for each shot are then evalued and shots for the summary are selected based on the final score.

4. Output Generation - The selected shots are then extracted from the original footage and concatenated in the final summary video.

Preprocessing Phase

Firstly, the system handles the frame preprocessing. The frames need to be adjusted and prepared before they are able to be worked with in the following steps.

Preprocessing

Several operations are performed on each frame to reduce the computational cost of feature extraction and following parts of the system.

1. Frame resizing - Each frame is resized to a predetermined resolution.

2. Grayscale - Each frame is transformed into a grayscale colour-space.

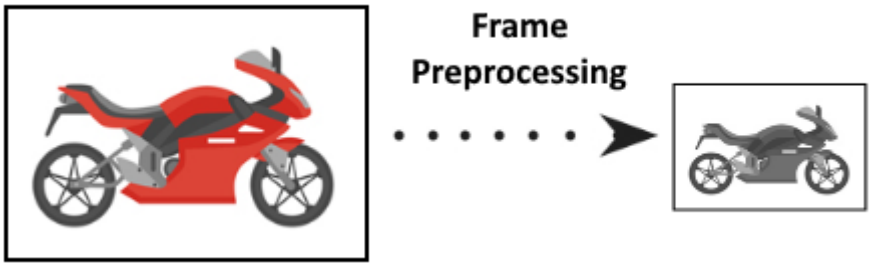

Figure 3.1: Frame preprocessing illustration

Extraction Phase

This is the first phase of the system. The main point of this phase is the extraction of image features from the input video. The input of this phase are the preprocessed frames from the previous phase. The system needs to describe each frame in a way that allows easier analysis of the frame's contents. For this purpose, the system will extract image features from each frame in the input video. Varying types of image features exist, each describing the video in their own way. For this reason it is essential to select an image feature type that is most suitable for this system. The output of the extraction phase is a file containing the extracted image features from the input video.

Analysis Phase

This is the second phase. Here the system processess the extracted image features and based on them evaluates which shots from the original video are to be included in the summary. To form a summary of interesting parts of the video, the system needs to evaluate which parts are interesting.

Inputs

The primary input of this phase are the image features from the previous phase. The system also allows the user to change the secondary input parameter, which is the number of shots in the final summary. With this parameter, the user will be able to regulate the system sensitivity by choosing how many shots are to be included in the final summary and therefore also affecting the length of the output video.

Inner Processes

- To select the desired shots, the system calculates the relation of each frame, forming a Similarity Matrix. The system then calculates the Difference Vector from the Similarity Matrix.
- The system selects the most interesting shots from the Difference vector. A scoring system is then used to assure an even distribution of shots has an impact on selecting the shots for the final summary. A score is assigned to each selected shot, it's value

based on the distance from the nearest other selected shot. Based on the difference value and the distance score, the shots for the final summary are selected. The system then represents frame indexes of those shots as ranges.

- Those ranges are then saved in a suitable way to be later passed to the summary generator.

Outputs

The output of this phase is a file containing a list of frame index ranges.

Generation Phase

This is the last phase of the system. The system goes through the input video and extracts the required shots described by the analyzer. The shots are then concatenated and a summary video is formed.

Inputs

The inputs are the list of ranges that describe the indexes of frames in the original video that are to be included in the final summary and the original video, from which the shots will be extracted.

Inner Processes

1. The system loads the input video and extracts shots described by the frame range list.
2. Extracted shots are then concatenated to form the summary video.
3. The summary video is saved.

Outputs

The output of this phase, and at the same of the entire video summarization system, is the summarized video.

3.4 System Design

For every system phase described in 3.3 corresponding parts were designed. These parts solve the problems defined by each system phase and together form the video summarization system.

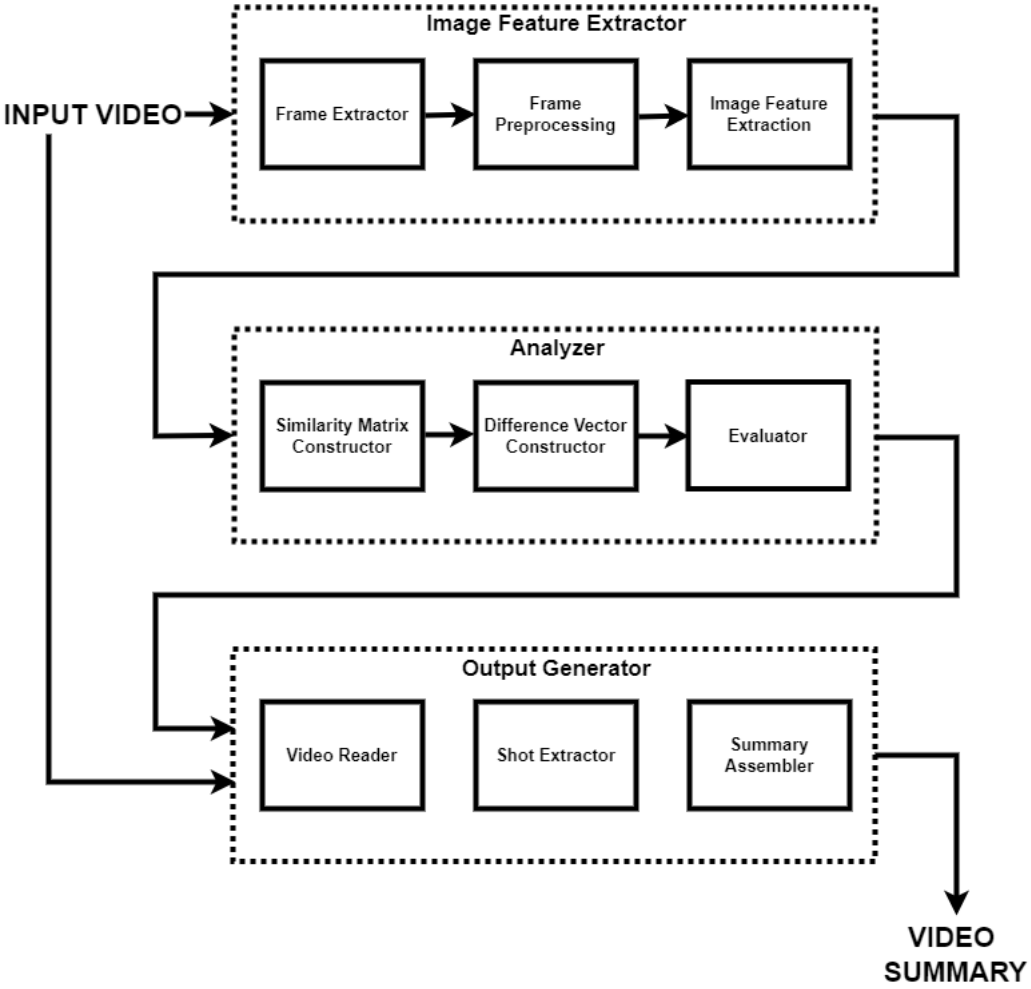

Figure 3.2: System Design

Image Feature Extractor

For the purpose of handling the problems of the Extraction phase explained in section 3.3, the image feature extractor was designed. It extracts image features from every frame of the input video and saves them to be used by the following parts of the system.

Before image features are able to be extracted from the video frames, preprocessing must be done on each of the frames. This process includes resizing the frame and transforming it into a grayscale colour-space. These steps are done to reduce the computational cost of extracting the image features.

After frame preprocessing, the system is ready to extract image features. The proposed image feature type in this solution is histogram of oriented gradients that is described in section 2.4. It has been chosen as for the purpose of our solution it is more robust and

offers more advantages than the other explored features, such as invariance geometric and photometric changes. Getting an image feature for every frame in a video is one of the most time-consuming processess in the entire system. Upon encountering a problem in the later phases of the system, starting over from the beginning is something the system should avoid. That is why the system needs to save the results of this operation to prevent wasting time in case of an error.

Analyzer

To transform the extracted image features into a form better suitable for further evaluation, following part of the system is proposed. Here, the image features are examined so the system is able to work with values instead of histograms.

The system calculates the difference of each pair of frames using their extracted image features. In the end, this gives us a matrix of calculated differences, illustrated in figure 3.3. One row in the matrix contains the differences of one frame from every other frame. I will refer to this matrix as the Similarity Matrix.

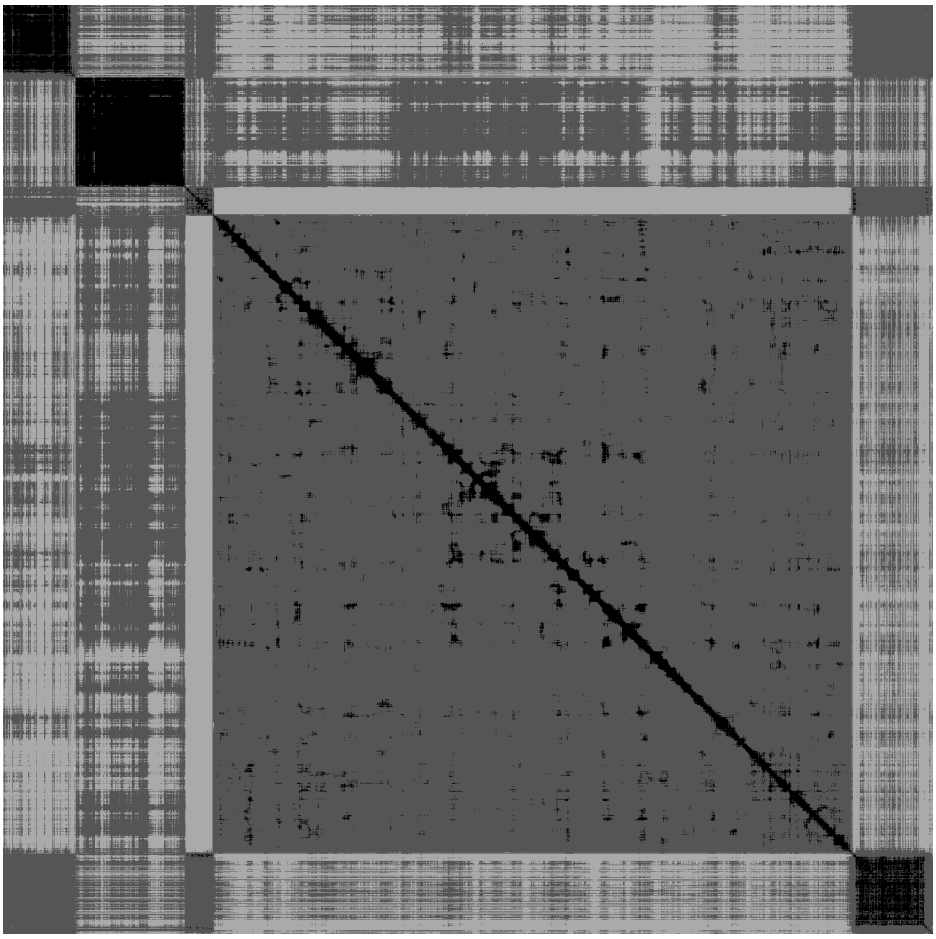

Figure 3.3: Visualization of the Similarity Matrix

From the Similarity Matrix, the system calculates the sum of all the difference values in each row, resulting in one score for every frame in the video which describes how different the frame is from every other frame in the video. I will address this score as the Difference

score. The resulting vector containing the Difference scores is visualized in figure 3.4 and I will refer to this vector as the Difference Vector.

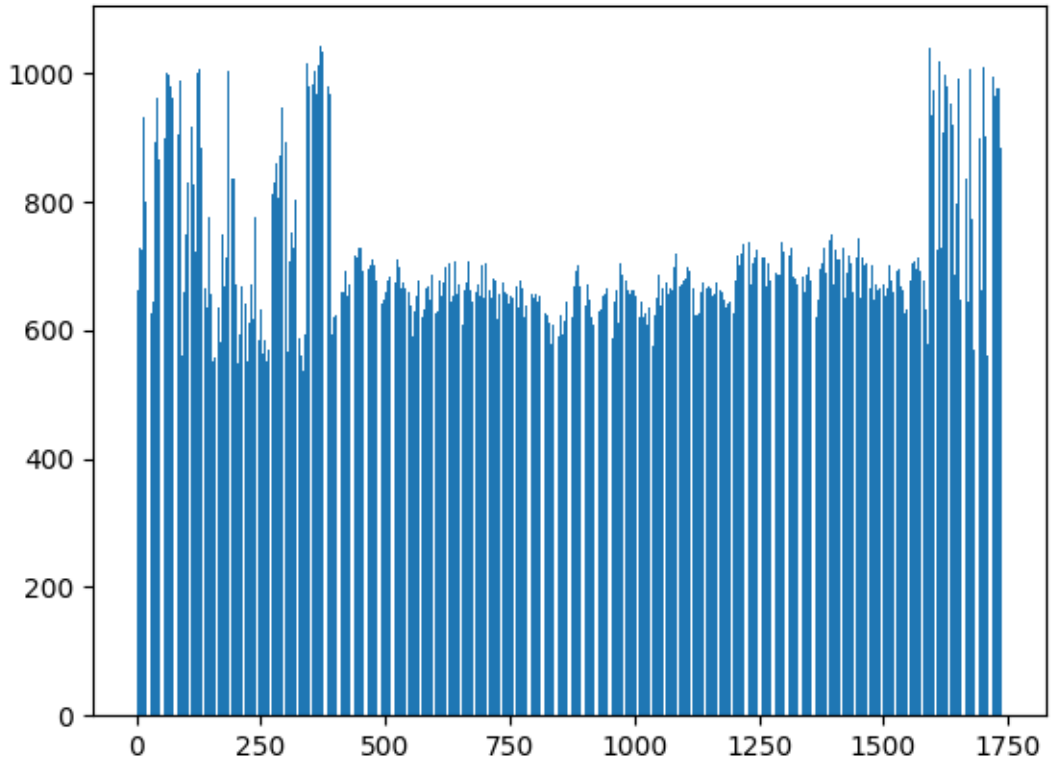

Figure 3.4: Visualization of the Difference Vector

The Evaluator loads the file containing the Difference Vector. The system then selects the most distinct frames and their close area, that also meet the required distribution of shots in the video.

The evaluator's tasks is to determine which shots to include in the final summary based on the data provided in the form of the Difference vector. To do this, the Evaluator loads the file containing the Difference Vector and evaluates which frames are interesting and then writes down the shots where the frames belong.

In this solution, the system assumes that interesting frames are frames most different from other frames. The system also takes into consideration the final distribution of chosen shots in the video and assigns a score to each of the shots. This score is based on the distance from the other nearest shot, and I will address this score as the Distance score.

From the Difference score and the Distance score, the system then computes the final score. I will address this final score as the Importance score. The system then selects frames with the highest Importance score. Those frames and shots they belong to are then chosen for the summary video. The amount of selected shots depends on the number of shots parameter, which is able to be changed by the user. The output of this part is a list of ranges of frames to be included in the summary video.

Output Generator

The system loads the input video and extracts important properties such as resolution and framerate. The system goes through the input video frame by frame while checking them against the expected shots. Shots that are described in the ranges acquired by the Analyzer are extracted from the input video. The summary video is assembled.

Chapter 4

Implementation, experiments and assessment

This chapter focuses on the technologies used in the implementation of the solution described in chapter 3.

4.1 Tools and Libraries

This section focuses on specific approaches I have chosen for the implementation of the system and states the software, tools and libraries selected for this purpose.

Used Technologies

- Microsoft Windows 10 was chosen as both the target and development platform, as it is widely used and I have years of experience using it.
- Python 3.9.0 was chosen for its ease of use, my previous experience with it and support of helpful mathematic and video editing tool libraries.
- OpenCV 4.4.0 is an open-source library that includes several hundreds of computer vision algorithms and image and video editing tools and was chosen as the core library with video and image editing tools that are essential for the implementation of this project.
- Atom 1.15.0, together with the `script` package, was chosen as the development environment, as it is very simple, user friendly and a more complex environment is not necessary.

Other Used Libraries and Modules

- Scipy 1.6.0 is an open-source Python library used for scientific computing and technical computing. It was chosen for it's distance calculating functions.
- The `pickle` module implements binary protocols for serializing and de-serializing a Python object structure and was used for it's ease of saving and loading data structures into a file.
- `filedialog` module from tkinter library was used for the file selection window.

- Pathlib is a Python module which provides an object API for working with files and directories.
- NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. This project works a lot with arrays, which is why NumPy is almost essential to work with.
- PIL was used for visualization of the Similarity Matrix.
- Matplotlib was used for visualization of the Difference Vector figures.

4.2 System parts

The system is implemented as several independent scripts, that together as a whole form the video summarization application.

1. Image Feature Extractor
2. Difference Vector Constructor
3. Evaluator
4. Output Generator

Image Feature Extractor

This part of the system represents the implementation of the Image Feature Extractor that has been designed in chapter 3.4 and is responsible for reading and extracting image features from the video input by the user.

Extracting the image features of every frame in the video consists of several steps. In my implementation, I create the OpenCV `VideoCapture` object using the input video. With that, the system is able to read the input video frame by frame. The image feature type I have chosen to describe my frames with are histograms of oriented gradients, or HOG for short. To extract these image features, the system first initializes a `HOG Descriptor` available in the OpenCV library. To reduce the computational cost of feature extraction, each frame is then resized to `256x256` dimensions and transformed into a grayscale colour-space. The preprocessed frame is then passed to the descriptor for extraction of HOGs. The image is divided into `64x64` sized cells. The reasoning behind this is that HOGs were originally used for human detection and `8x8` cells were used to capture details of their facial features, which is not necessary in this work. The descriptor then gives us histograms with 9 bins made from the gradient magnitude and orientation all concatenated into one histogram. Once features were extracted from every frame they are dumped into a file using the `pickle` module.

Difference Vector Constructor

This part of the system represents the implementation of the Similarity Matrix Constructor and Difference Vector Constructor from the Analyzer described in chapter 3.4. The extracted image features from Image Feature Extractor are loaded from the file. The system then initializes an empty two dimensional matrix with the length of both of the

dimensions being the number of frames in the video. For every frame, the system computes the difference of the frame from every other frame, using the euclidean distance function mentioned in section 2.12 from the `SciPy` library. The HOGs of the frames are compared by the distance function, resulting in difference values for each pair of frames. The computed values are saved into the matrix, resulting in the Similarity Matrix. Values in the matrix are then normalized. Values on each row are then summed up, resulting in the Difference vector. The Difference Vector is then saved into a file.

Evaluator

Evaluator represents the last part of the Analyzer described in chapter 3.4. The evaluator loads the Difference vector from the file. The system then finds frames with the highest Difference score in the Difference vector. The amount of selected frames is ten times the value of the number of shots parameter. Distance score is then assigned to the selected frames. Distance score is based on the distance from the other nearest selected frame and the way the distance is calculated is the number of frames between the two selected frames.

Both the Difference scores and the Distance scores for the selected frames are then normalized and multiplied together, resulting in an Importance score for each of the selected frames. The frame with the highest Importance score is then chosen together with a padding of 30 frames in both directions, giving us a shot for the final summary. The shot is then saved into a list of chosen shots and is removed from the Difference Vector. This step is repeated until the amount chosen shots is equal to the number of shots parameter. The number of shots parameter determines the sensitivity of the system and how long the final summary will be and is able to be changed to better fit the nature of the input video. The result of this part are sorted ranges of frames defining the chosen shots, and are saved into a file.

Output Generator

The Output Generator loads the input video and the file containing the selected shots. It then opens a new video and writes the selected shots into the new video.

For this, the output generator loads and unpickles the file with ranges. Just like in Image Feature Extractor, the input video is loaded and read with the `VideoCapture` object. Properties are extracted from the original video, such as framerate, resolution, number of frames and length. After properties have been extracted, `VideoWriter` object can be initialized with framerate and resolution as parameters.

The system then goes through frames in the input video and those that fall into the selected shots are written into the output video.

4.3 Datasets

A large amount of annotated datasets exist and are able to be found online. These datasets highly vary in the nature of the content they can be used for. This work mainly focuses on summarizing surveillance camera footage or footage of static cameras in general. A summary of a video captured by a non-static camera is also able to be generated, however the results may not be as precise as with the static ones.

For this purpose, a dataset with videos falling into both categories, static camera and dynamic camera footage is required. Ideally the data in the dataset will be separated

into surveillance footage and footage of other nature, so that we are able to assess the overall precision of the application's results, but also precision resulting from the system's execution on only the static footage type of data.

Chosen Dataset

The annotated dataset that was chosen for the purpose of testing this system is one called Visiocity. This dataset is a collection of several long videos spanning across different categories, such as TV shows, sports, surveillance, etc. [12]

Each video is uniformly segmented into snippets. Videos in different categories have different snippet sizes. One of the main characteristics of Visiocity dataset is that the annotations of shots are not directly marked as ground truth. Instead, concepts are marked for each shot. Those concepts contain categories such as entity, action, number of people etc. This way the user is able to generate a different ground truth based on what purpose their system is going for.

For the purpose of testing this video summarization system, the main focus will be the surveillance category of the Visiocity dataset. While the general result of testing on the entirety of the dataset will be also included, the most important will be the surveillance category, as the topic of surveillance footage is what our system is built around.

4.4 Experiments

This chapter focuses on testing the precision of the designed and implemented system and also uncovering possible shortcomings of the system or exposing issues connected to the system's implementation. For this purpose a dataset is required, one with videos that fit the nature of the videos that this paper has focused on. This is achieved by analyzing the results after executing the system on a dataset of annotated videos. Some of the interesting experiments, which have been selected based on their results where the behavior of the results deviates from the average or has an interesting character, have been examined and will be highlighted in this chapter. The nature of these experiments will be introduced and the problematics of summarizing these videos will be explained.

System Testing

The parameter that affects the sensitivity of the system is the number of shots to be included in the final summary. A number of experiments have been done by executing the system on the annotated surveillance videos from the chosen dataset. Tests of the system have been done and repeated while changing the number of shots parameter. Performance of the implemented system has been evaluated using receiver operating characteristic curve analysis. In figure number 4.1 is the receiver operating characteristic curve of the implemented system.

As can be observed, with a low number of shots included, the precision is high, however the recall is minimal. This is caused by the chosen shots covering minimal portion of the video. As these are videos taken by surveillance cameras, their length is in the matter of hours. For a small number of shots to include all the activity in an hour long video, the video would have to have minimal amount of activity and therefore require only a very short summary.

By increasing the number of shots, recall rises significantly, as the system is able to cover a larger portion of the video, but the precision begins to decrease. In most cases, the decrease in precision may happen for two reasons. First, the system places too much value on distribution and selects well distributed shots over the more interesting shots. Second reason is that with increasing recall, most of the interesting shots in the video are taken already. Due to this, there are either no interesting shots left in the video, or the system is not robust enough to extract the remaining few interesting shots. Either way, this leads to uninteresting shots to be included in the summary and therefore reducing the precision of the system.

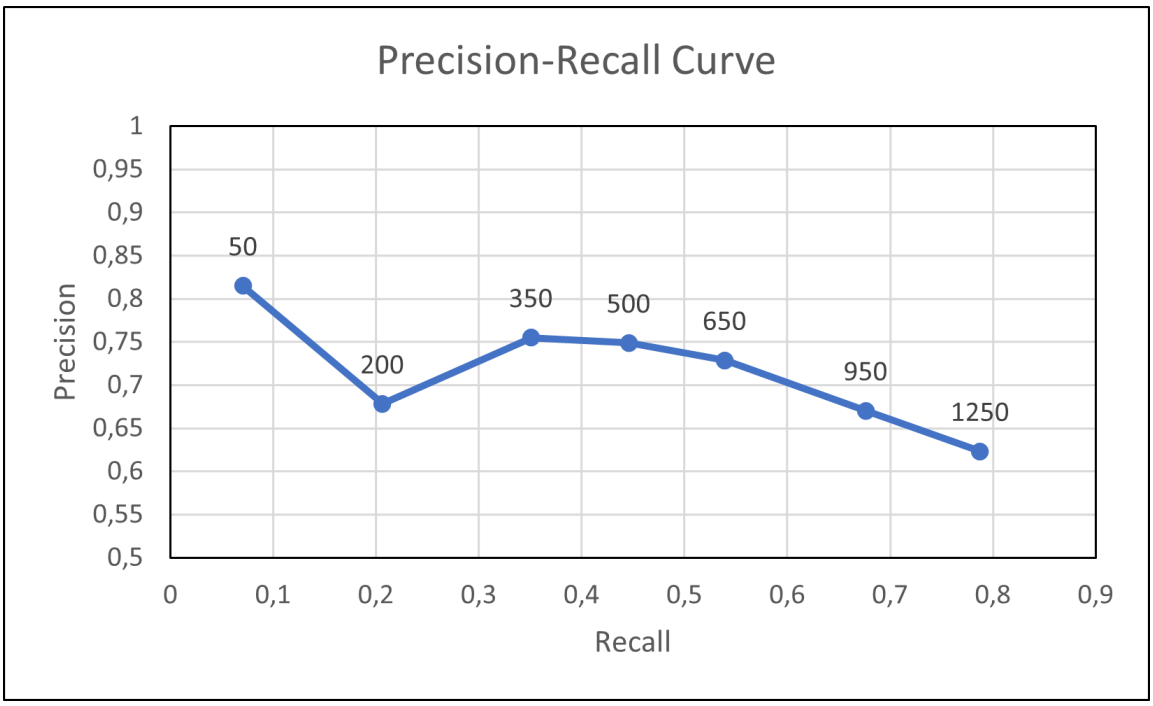

Figure 4.1: ROC Curve of the system tested on surveillance camera footage

Experiment 1

One of the executed experiments was done on a video labeled `surveillance_12` from the annotated dataset. This video shows the view of a surveillance camera likely installed on some house's porch, overlooking together the driveway and the yard, as can be seen in figure 4.2.

Figure 4.2: A single frame from surveillance video 12

The overall precision remains high for varying number of shots, while the recall is rising with increasing number of shots. That is because there is a lot of activity in the video with only a few shots where nothing is happening, therefore there is a lot of shots that are to be included in the summary. This can be seen in figure 4.3, where even with a high number of extracted shots and high precision the system did not reach maximum recall. This experiment demonstrates a nearly ideal case of the system's function.

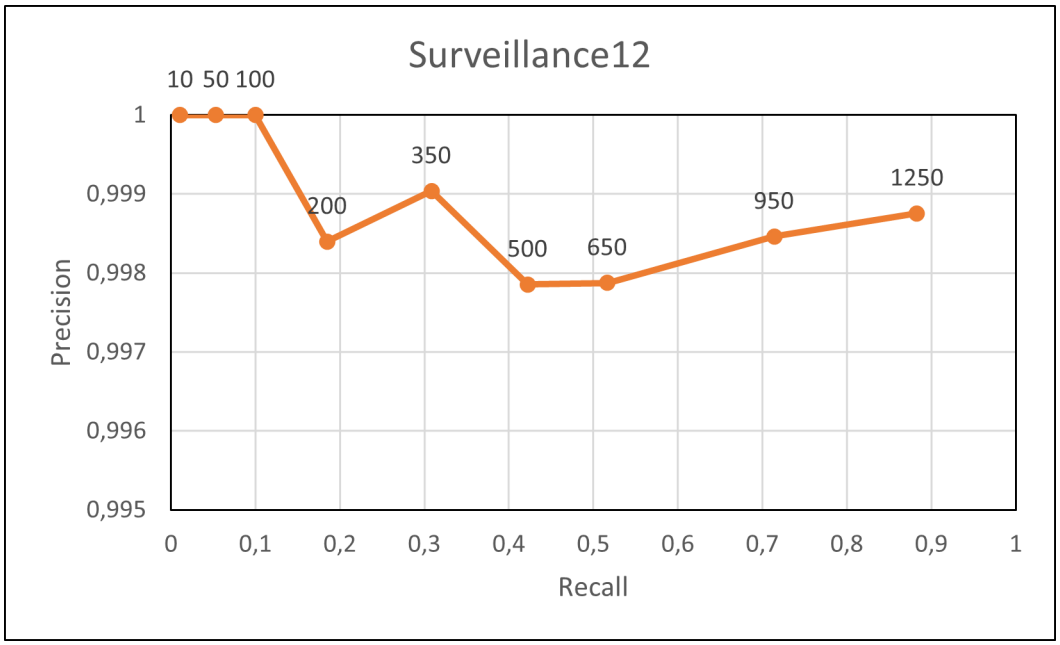

Figure 4.3: ROC Curve of the surveillance video 12.

Experiment 2

Another interesting experiment was done on video `surveillance_3`. This video captures what seems to be the front row of chairs in a classroom. At the beginning of the video, there are people discussing something at the front of the classroom. After they leave, the classroom is empty for the rest of the video, with only one instance of a person briefly entering the classroom. A picture of the classroom for reference can be seen in figure 4.4. The results in this experiments are very similar to Experiment 1.

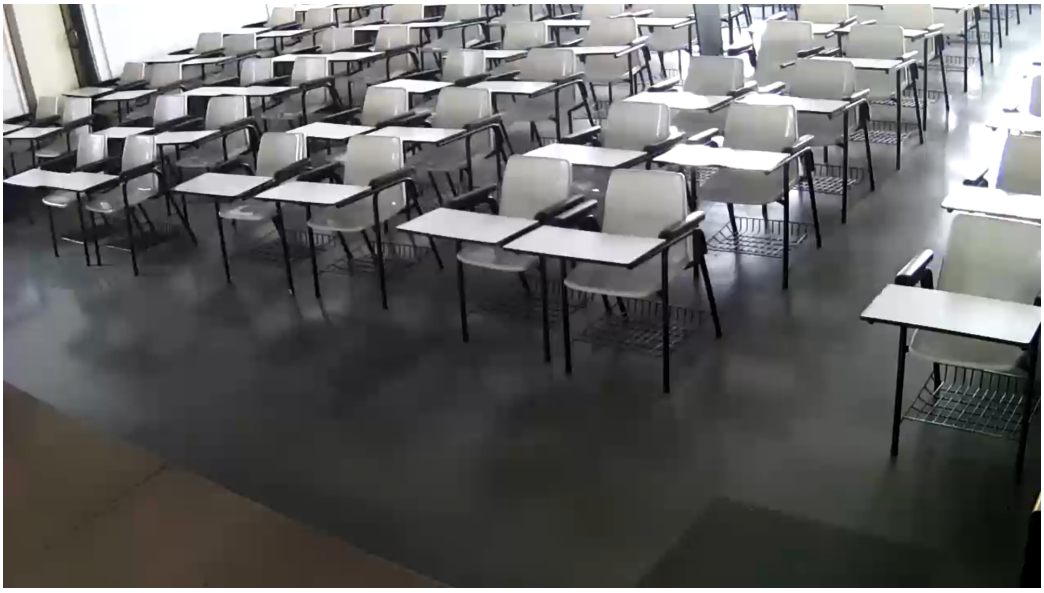

Figure 4.4: A single frame from surveillance video 3

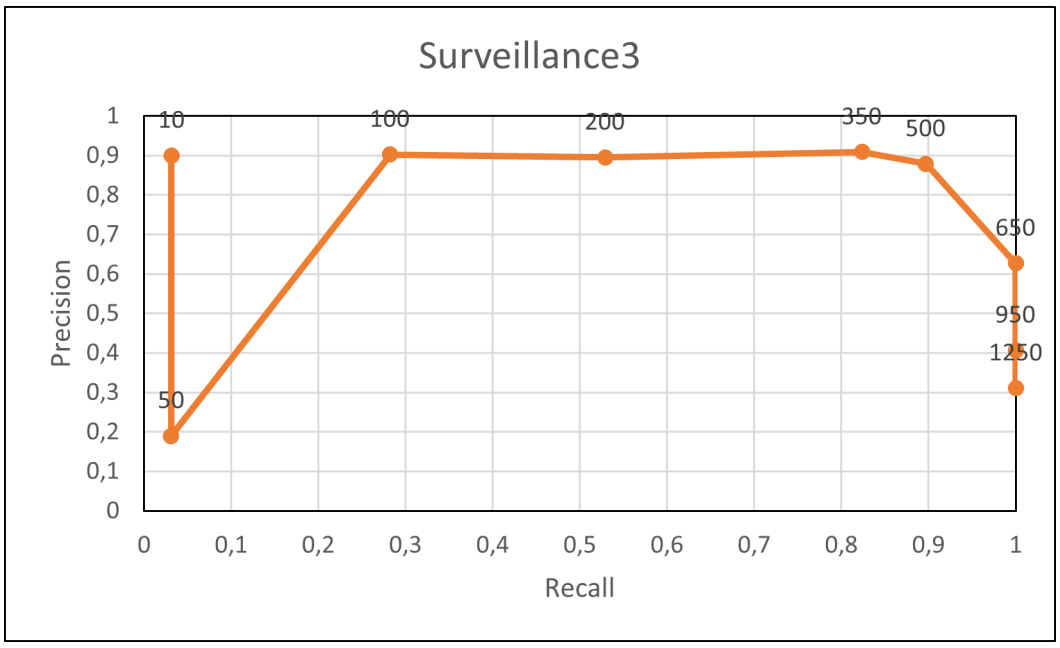

Figure 4.5: ROC Curve of the surveillance video 3

One of the differences that can be observed is that with the number of shots parameter being set to 650, the system has already achieved maximum recall. This means that all the shots annotated as relevant have been included in the summary, and additional unnecessary shots are being included, which naturally causes the precision to decrease.

The second observation is the anomaly that occurs while number of shots is set to 50. Both precision and recall are considerably low, though recall being low is almost definitely due to the fact that only 50 shots are extracted. The low precision can be explained by the system placing too much value on the Distribution score. As can be seen, the precision is high in the previous testing cycle, where number of shots was set to 10. As the system's number of shots parameter increased from 10 to 50, the system analysed more shots and in the end selected different shots that scored higher because of the Distribution score. This may have caused the shots selected in the previous cycle not to be included, leading to the drop in precision.

Experiment 3

When testing video `surveillance_7`, the precision was very low, and while recall increases with increasing shots, precision stays low for all number of shots.

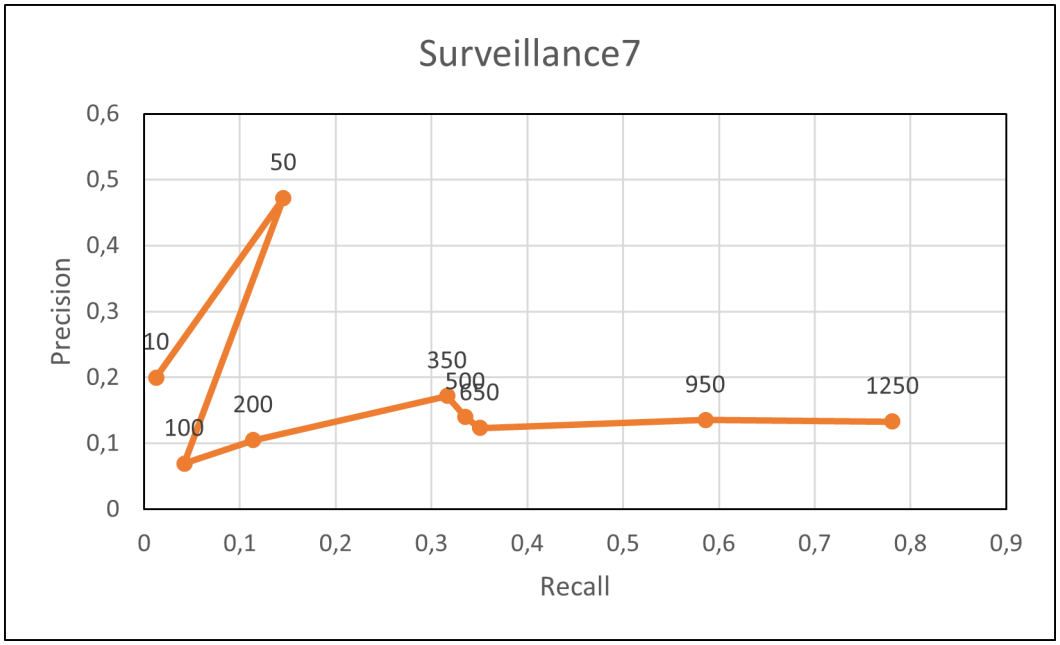

Figure 4.6: ROC Curve of the surveillance video 7

After examining the results and the original video, I have observed that from a certain point in the video, night vision gets turned on and the frames from then on are evaluated as highly different from the previous frames. The video captures a lightly illuminated room. Upon closer examination of the processed frames, I have noticed a significant increase in illumination of the room upon activation of the camera's night vision. This suggests that the camera in question is using an infrared illuminator, which activates together with night vision for better visibility at night. With the IR illuminator active, illumination increased to a degree where much more detail is visible in certain parts of the room. This has caused

the system the system to evaluate the frames after night vision has been turned on as highly different, as a significant amount of detail appeared in the video.

Figure 4.7: Surveillance video 7 before and after night vision

This exposes an issue with the system. Upon the occurence of a significant change in the video that remains there for the remainder of the video or a long amount of time, the Difference vector is split into two or more parts. The frames in the part that contains fewer frames then score high in the Analyzer and are more prone to be selected for the summary, as they are very different from the frames in the part containing more frames, giving them a high difference value.

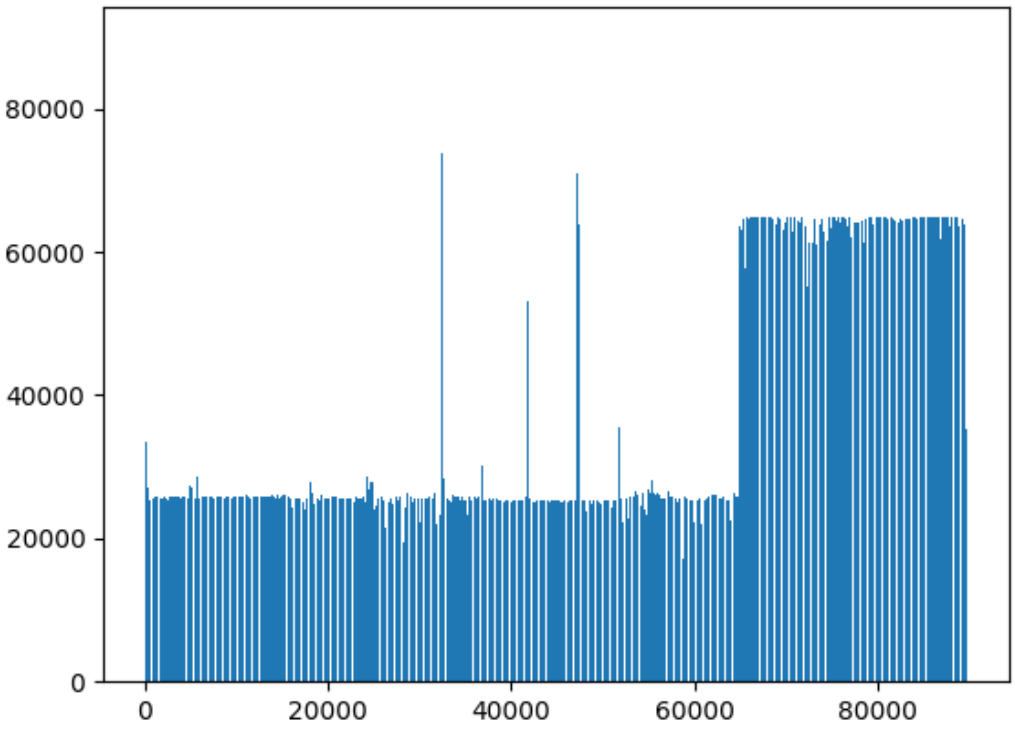

Figure 4.8: Difference vector of Surveillance video 7

4.4.1 Discovered implementation problems

One of the biggest difficulties I have come across during the testing of the implemented system is the difficulty of summarizing long videos.

For the purpose of finding the most interesting shots in the video, this solution describes a method where an image feature is extracted from every single frame in the video. As shown in section 3.3, the image features are then compared against each other to create the Difference vector, which is then further processed. The construction of the Difference vector is the most computational time demanding part of the system. The problem in this solution is that the system analyses every frame of the input video regardless of the video frame rate. In most cases, analysing every single frame of the video is unnecessary. For instance, if the system were to run on a video with a frame rate of 60 frames per second, it would be ultimately irrelevant in terms of video summarization whether the system was analysing every fourth frame instead of every single one. However in terms of effectivity and speed of the system, in this case skipping most of the frames would make the system about 15 times faster.

For the purpose of analysing the extracted image features, I have chosen the approach of constructing a similarity matrix by comparing each of the features, as described in 3.3. The execution of this approach has been very demanding on computer memory. When considering long videos this matrix could reach very large sizes. Consider an hour long video with a framerate of 24 frames per second. That gives us 86400 frames. In the matrix, each frame has to be compared with every other frame, that is $86400x86400$, multiplied by the size of the data type. Even if we use Numpy's 2 byte float instead of Python's inbuilt 8 byte float, that gives us a need for 16GB of memory. Even if a system with this much memory was available, the summarizing system would fail on any video longer than an hour or with a higher framerate than 24 frames per second. As allocating the whole matrix at once is simply not a possible solution, I have come up with a simple workaround of skipping the construction of the whole similarity matrix, and instead after computing each row of the matrix summing the values up resulting in a value for the Difference vector. This change has increased the time required to process the image features, as the optimalization the Similarity Matrix brought is gone, but the memory usage has been significantly reduced and allowed me to analyse several videos at once. In retrospect, a more effective solution to the problem would be to gradually save the computed data for the similarity matrix into a file instead of putting the burden on memory alone.

Chapter 5

Conclusion

This project aimed to study video summarisation techniques and design and implement a system capable of video summarisation. I have researched various techniques of video summarisation and studied different types of image features, how they can be used in computer vision, and how to work with them and evaluate them. I have recognised that the difficulty of video summarisation lies not only in the methods used to achieve the final summary but also in optimisation and the system's use of computation time, and the user's definition of what is essential in regards to the final summary. For the reason of the varying natures of videos a video summarisation system can be used on, it would be very difficult or nearly impossible to design a system that would work on all types of videos and fit the needs of all users. For this reason, in this thesis, I have decided to focus on static camera videos, specifically surveillance camera footage. Based on the researched materials, I have designed a video summarising system, which uses histograms of oriented gradients extracted from frames in the video and a difference score, which is based on the distribution of shots to determine which shots are interesting. I have implemented this system mainly using the OpenCV library available for Python, which focuses on computer vision and offers video processing and editing tools. The testing of the system has been done using the Visiocity annotated dataset. The category of videos in the dataset that the system was tested on was surveillance. The testing has revealed the overall precision of the system, as well as certain anomalies, which has led to the discovery and identification of some system shortcomings and weak points.

Bibliography

[1] Alsmadi, M. Content-Based Image Retrieval Using Color, Shape and Texture Descriptors and Features. *Arabian Journal for Science and Engineering*. february 2020, vol. 45. DOI: 10.1007/s13369-020-04384-y.

[2] Belo, L., Caetano, C., Patrocínio Jr, Z. and Guimarães, S. Summarizing video sequence using a graph-based hierarchical approach. *Neurocomputing*. january 2016, vol. 173, p. 1001–1016. DOI: 10.1016/j.neucom.2015.08.057.

[3] Chatterjee, A. *OpenGenus: <Euclidean vs Manhattan vs Chebyshev Distance>*. 2022. Information available from opengenus.org. Available at: https://iq.opengenus.org/euclidean-vs-manhattan-vs-chebyshev-distance/.

[4] Chen, L., Guo, H., Wang, H., Chen, Y.-L. and Wu, X. The visual location of workpiece based on Hermite Interpolation and mapping for robot arms. In:. April 2015, p. 171–176. DOI: 10.1109/ICIST.2015.7288962.

[5] Dalal, N. and Triggs, B. Histograms of oriented gradients for human detection. In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. 2005, vol. 1, p. 886–893 vol. 1. DOI: 10.1109/CVPR.2005.177.

[6] Duc, T., Le, D.-D. and Satoh, S. Scalable approaches for content based video retrieval. *Progress in Informatics*. march 2014, p. 31. DOI: 10.2201/NiiPi.2014.11.5.

[7] Elkhattabi, Z., Tabii, Y. and Benkaddour, A. Video Summarization: Techniques and Applications. *World Academy of Science, Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering*. 2015, vol. 9, p. 928–933.

[8] Haq, H. B., Asif, M. and Bin, M. Video Summarization Techniques: A Review. *International Journal of Scientific & Technology Research*. february 2021, vol. 9, p. 146–153.

[9] Harris, C. and Stephens, M. A Combined Corner and Edge Detector. In: *Proceedings of the Alvey Vision Conference*. Alvety Vision Club, 1988, p. 23.1–23.6. Doi:10.5244/C.2.23.

[10] Hassaballah, M., Ali, A. and Alshazly, H. Image Features Detection, Description and Matching. In:. February 2016, vol. 630, p. 11–45. DOI: $10.1007/978\text{-}3\text{-}319\text{-}28854\text{-}3_2. ISBN ISBN 978-3-319-28852-9$.

[11] Jones, E., Oliphant, T., Peterson, P. et al. *SciPy: Open source scientific tools for Python*. 2001-2022. Available at: http://www.scipy.org/.

[12] Kaushal, V., Kothawade, S., Tomar, A., Iyer, R. and Ramakrishnan, G. *How Good is a Video Summary? A New Benchmarking Dataset and Evaluation Framework Towards Realistic Video Summarization*. 2021.

[13] Kunaver, M. and Tasic, J. Image feature extraction - an overview. In:. February 2005, p. 183 – 186. DOI: 10.1109/EURCON.2005.1629889. ISBN 1-4244-0049-X.

[14] Li, X., Zhao, B. and Lu, X. A General Framework for Edited Video and Raw Video Summarization. *IEEE Transactions on Image Processing*. 2017, vol. 26, no. 8, p. 3652–3664. DOI: 10.1109/TIP.2017.2695887.

[15] Li, Y., Merialdo, B., Rouvier, M. and Linarès, G. Static and dynamic video summaries. In:. November 2011, p. 1573–1576. DOI: 10.1145/2072298.2072068.

[16] Mallick, S. *LearnOpenCV: Histogram of Oriented Gradients explained using OpenCV*. 2022. Information available from learnopencv.org. Available at: https://learnopencv.com/histogram-of-oriented-gradients/.

[17] Manjunath, B., Ohm, J.-R., Vasudevan, V. and Yamada, A. Color and texture descriptors. *IEEE Transactions on Circuits and Systems for Video Technology*. 2001, vol. 11, no. 6, p. 703–715. DOI: 10.1109/76.927424.

[18] Mistry, Y., Ingole, D. and Ingole, D. Content based image retrieval using hybrid features and various distance metric. *Journal of Electrical Systems and Information Technology*. january 2017, vol. 5. DOI: 10.1016/j.jesit.2016.12.009.

[19] Moravec, H. *Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover*. Pittsburgh, PA: Carnegie Mellon University, September 1980.

[20] Nguyen, D., Hong, H., Kim, K. and Park, K. Person Recognition System Based on a Combination of Body Images from Visible Light and Thermal Cameras. *Sensors*. march 2017, vol. 17, p. 605. DOI: 10.3390/s17030605.

[21] Novak, C. and Shafer, S. Anatomy of a color histogram. In: *Proceedings 1992 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. 1992, p. 599–605. DOI: 10.1109/CVPR.1992.223129.

[22] Stanchev, P., Green, D. and Dimitrov, B. High level color similarity retrieval. *International Journal*. january 2003, vol. 10.

[23] Team, E. *educative: What is feature extraction?* 2022. Information available from educative.io. Available at: https://www.educative.io/edpresso/what-is-feature-extraction.

[24] Tian, D. A review on image feature extraction and representation techniques. *International Journal of Multimedia and Ubiquitous Engineering*. january 2013, vol. 8, p. 385–395.

[25] Willis, A. and Sui, Y. An algebraic model for fast corner detection. In: *2009 IEEE 12th International Conference on Computer Vision*. 2009, p. 2296–2302. DOI: 10.1109/ICCV.2009.5459443.

Appendix A

Contents of DVD

The attached DVD contains the following items:

1. Digital version of this text.
2. Source code of this text.
3. Source code of the implemented software.
4. A Readme file with steps to launching the software.