KTH Computer Science
and Communication

Annotation and indexing of video content based on sentiment analysis

DANIEL VON WITTING

Master's Thesis at NADA
Supervisor: Carl Henrik Ek
Examiner: Danica Kragic

Abstract

Due to scientific advances in mobility and connectivity, digital media can be distributed to multiple platforms by streams and video on demand services. The abundance of video productions poses a problem in terms of storage, organization and cataloging. How the movies or TV-series should be sorted and retrieved is much dictated by user preferences, motivating proper indexing and annotation of video content. While movies tend to be described by keywords or genre, this thesis constitutes an attempt to automatically index videos, based on their semantics. Representing a video by the sentiment it invokes, would not only be more descriptive, but could also be used to compare movies directly based on the actual content. Since filmmaking is biased by human perception, this project looks to utilize these characteristics for machine learning. The video is modeled as a sequence of shots, attempting to capture the temporal nature of the information. Sentiment analysis of videos has been used as labels in a supervised learning algorithm, namely a SVM using a string kernel. Besides the specifics of learning, the work of this thesis involve other relevant fields such a feature extraction and video segmentation. The results show that there are patterns in video fit for learning; however the performance of the method is inconclusive due to lack of data. It would therefore be interesting to evaluate the approach further, using more data along with minor modifications.

Referat

Automatisk indexering av videomaterial baserat på värderingsanalys

Tack vare tekniska framsteg inom mobilitet och tillgänglighet, kan media såsom film distribueras till flertalet olika plattformar, i form av strömning eller liknande tjänster. Det enorma utbudet av TV-serier och film utgör svårigheter för hur materialet ska lagras, sorteras och katalogiseras. Ofta är det dessutom användarna som ställer krav på vad som är relevant i en sökning. Det påvisar vikten av lämplig notation och indexering. I dag används oftast text som beskrivning av videoinnehållet, i form av antingen genre eller nyckelord. Det här arbetet är ett försök till att automatiskt kunna indexera film och serier, beroende på det semantiska innehållet. Att istället beskriva videomaterialet beroende på hur det uppfattas, samt de känslor som väcks, innebär en mer karaktäristisk skildring. Ett sådant signalement skulle beskriva det faktiska innehållet på ett annat sätt, som är mer lämpligt för jämförelser mellan två videoproduktioner. Eftersom skapandet av film anpassar sig till hur människor uppfattar videomaterial, kommer denna undersökning utnyttja de regler och praxis som används, som hjälp för maskininlärningen. Hur en film uppfattas, eller de känslor som framkallas, utgör en bas för inlärningen, då de används för att beteckna de olika koncept som ska klassificeras. En video representeras som en sekvens av klipp, med avsikt att fånga de tidsmässiga egenskaperna. Metoden som används för denna övervakade inlärning är en SVM som kan hantera data i form av strängar. Förutom de teknikaliteter som krävs för att förstå inlärningen, tar rapporten upp relevanta andra områden, t.ex. hur information ska extraheras och videosegmentering. Resultaten visar att det finns mönster i video, lämpliga för inlärning. På grund av för lite data, är det inte möjligt att avgöra hur metoden presterar. Det vore därför intressant med vidare analys, med mer data samt smärre modifikationer.

Contents

1 Introduction **1**
1.1 Video Content Analysis 2

2 Background **3**
2.1 Video Segmentation 3
2.2 Machine Learning and Classification 3
2.3 Report Outline 4

3 Video Editing **4**
3.1 Continuity Editing 5
3.1.1 180° Rule 5
3.1.2 Matched-Exit/Entrance 6
3.1.3 Fade and Dissolve 7
3.2 Attention 7
3.3 Summary 7

4 Problem Formulation **8**
4.1 Goal 9
4.2 Expectations 10
4.3 Formal Mathematical Description 11
4.4 Prerequisites 12

5 Research and Related Work **12**
5.1 Image Recognition 13
5.2 Action Recognition 13
5.3 Machine Learning 14
5.3.1 Learning schemes 15
5.3.2 Evaluation 15
5.3.3 Related Work 16
5.3.4 Summary 16

6 Feature Extraction **17**
6.1 Feature Design 18
6.2 Static Features 19
6.2.1 HSV Histogram 19
6.2.2 Full Distribution Histogram 19
6.2.3 Edge Histogram 20
6.2.4 Fourier Transform 21
6.3 Dynamic Features 24
6.3.1 Optical flow 24
6.3.2 Motion Attention 26
6.3.3 Audio Features 28
6.4 Similarity Measures 29
6.4.1 L_1 distance 30
6.4.2 L_2 distance 30
6.4.3 χ^2 distance 31
6.4.4 Edge Histogram (EH) distance 31

7 Classification **31**
7.1 Clustering . . . 32
7.1.1 K-means . . . 32
7.1.2 Agglomerative Hierarchical Clustering . . . 33
7.1.3 Cluster Evaluation . . . 33
7.2 Support Vector Machines (SVM) . . . 33
7.2.1 Linear Support Vector Machine . . . 34
7.2.2 Dual Problem . . . 35
7.2.3 Kernels . . . 36
7.2.4 Slack variable . . . 38
7.2.5 Sequential Minimal Optimization (SMO) . . . 39
7.2.6 String Kernel . . . 39

8 Video Segmentation **41**
8.1 Shot Segmentation . . . 41
8.1.1 Shot boundary detection without thresholds . . . 42
8.1.2 Shot Feature Representation . . . 44
8.2 Scene Segmentation . . . 45
8.2.1 Scene Feature Representation . . . 45

9 Methodology **45**
9.1 Setup . . . 47

10 Results **47**
10.1 Shot Clustering . . . 47
10.2 Classification . . . 50
10.2.1 Multi-class classification . . . 52
10.2.2 Binary classification . . . 54

11 Conclusions and Future Work **55**
11.1 Shot Representation and Sequence Construction . . . 56
11.1.1 Future Work . . . 56
11.2 Classification Performance . . . 57
11.2.1 Future Work . . . 58

References **59**

List of Figures

1 Illustration of the 180 ° rule. The recording of the scene is established on the right side in the figure. By doing so, the spatial relationships have been set. Crossing the 180 ° set by the establishing shot and the characters in the scene, would reverse the positions in the scene, causing confusion. 6

2 Description of Frame-to-Scene segmentation. The frames of a movie can be segmented in to shots. A sequence of shots can describe an event, while another sequence set the atmosphere. These two different concepts of a "scene" do not necessarily coincide; an atmosphere can change in the middle of an event and vice versa. 9

3 Example of edge histogram differences, where the peaks is due to more abrupt changes from frame to frame. The y-axis is the difference magnitude, while the x-axis represent the frame identification number, i.e. time. 20

4 Illustration of the five different edge types searched for in the frame. 20

5 Different partitioning of an image. Each configuration of pixels contribute with five histogram bins, one for each edge type. . . . 21

6 Images illustrating important properties of a Fourier transformed image. First column: Image manipulations. Second column: Resulting magnitude spectrum. Translation does not affect the magnitude of the Fourier transform, and a rotation of the original image entail a rotation of the spectrum. Image reference: MUST Creative Engineering Laboratory (http://lab.must.or.kr) 23

7 A visualization of the optical flow feature. The color represent the direction of the motion, according to the HSV color wheel. The intensity represent the magnitude of the motion. a) Resulting optical flow. b) Original frame. c) HSV colorwheel. 26

8 Support Vector Machine learns a hyperplane that separates two classes, with the largest possible margin between classes. For linear SVM, the said hyperplane is a line. 35

9 The left figure show an example when the data (in Cartesian coordinates) can not be linearly separated. A transformation into Polar coordinates makes linear separation possible. 36

10 Illustration of a non-linear SVM as well as the use of slack variables. The black dotted line represent a non-linear decision boundary. Linear separation is also possible by introducing a slack variable. 38

11 An illustration of how cut candidates are represented in the shot detection algorithm. The x-axis is the frame ID, while the y-axis corresponds to the magnitude of frame differences. For the first (left) sliding window, the inter-frame differences have no isolated peaks. The second (right) sliding window includes a peak which is unquestionably the largest difference in the series, i.e. a suspected cut. 43

12 Overview of the proposed model outline. 46

13 The cluster assignments for two different features. The horizontal axis corresponds to the assigned cluster number 1-8 (A-H), while the vertical axis describe the feature values. In a) it can be seen that the range of the feature values vary for different clusters. For the feature shown in b) almost all cluster share the same range of values. 48
14 The normalized distribution of letters/characters A-H, for each labeled atmosphere. 48
15 The 10 most similar sequences for the labels eventful, gloomy and tense. The letters are represented as a colored block according to: A - red, B - blue, C - cyan, D - grey, E - magenta, F - yellow, G - green, H - black. 49
16 The 10 most similar sequences for the labels joyful, introductory, gloomy. The letters are represented as a colored block according to: A - red, B - blue, C - cyan, D - grey, E - magenta, F - yellow, G - green, H - black. 49
17 Visualization of the Gram matrix when using a string kernel, i.e. the kernel response for all pairs of string sequences. The data is sorted by class label. Blue corresponds to dissimilar sequences, while red indicate similarity. 50
18 Visualization of the Gram matrix when using a chi-square kernel, i.e. the kernel response for all pairs of letter histograms. 51
19 Visualization of the Gram matrix when using a string kernel, i.e. the kernel response for all pairs of string sequences. The data is first sorted by the movie order, followed by again sorting with respect to labels. 52
20 Visualization of the Gram matrix using a string kernel, for binary classification. The classes are sorted as *eventful* and *not eventful*. 54

List of Tables

1 Performance of the multi-class classifier evaluated on the training data. 53
2 Performance of the multi-class classifier evaluated on the test set. 53
3 Performance of the multi-class classifier evaluated on the test set. The number of training and test examples have been adjusted; 10 episodes for training and 4 episodes for testing. 54
4 Performance of the binary classifier evaluated on the training set. 55
5 Performance of the binary classifier evaluated on the test set. . . 55
6 Performance of the binary classifier evaluated on the test set. The number of training and test examples have been adjusted; 10 episodes for training and 4 episodes for testing. 55
7 The confusion matrix for a classification of the test set, using a multi-class classifier. 57

1 Introduction

The extent of using video content to mediate and express ourselves has been rapidly increasing in the modern society. Not only are videos used as entertainment and information, the technical mobility allows us to record video anywhere for any purpose. By acquiring and process information using our senses, the human brain manages to interpret and classify the events and scenarios within the video, based on our previously achieved knowledge. Research in computer science tries to model this extraordinary capacity for computers to use, in the field of machine learning. To teach computers how to mimic human perception can be useful in many regards. One would be that a well-trained computer slavishly follows the given instructions, eliminating common human errors. It would also be possible that the artificial perception notice patterns where humans do not, helping us to act and become as effective as possible.

Arguably, the resource that humans tend to value the most is *time*. Regardless of occupation, time is as valuable at work as it is at home. Obvious applications such as running the machines of an industry at optimal speed, or design schedules to be as efficient as humanly possible, can be done by the assistance of computers. The fantasy that someone or something simply could perform all of the tasks that people wish not to, is a part of human behavior. A more subtle supplement, than a self-aware work robot, is the simple but powerful task of categorization. By automatically store organized and categorized information, time is saved both by the fact that it is automatized, and that the information becomes search-able. Storing and searching is not a new concept, however doing it efficiently, in a meaningful manner and automatically is becoming more desired everywhere.

Today it is, at least for written information, expected that a simple search containing a few describing words should be enough in order to find the intended document. This is evolving to be true for text-based content by advanced search engines. For example, not long ago, librarians were essential for the gathering of any written information. While still a excellent source of knowledge and competence, librarians are today aided by technology to store and retrieve information. Additionally, the Internet allows research to be done remotely from practically anywhere.

It is increasingly popular to build similar applications for more complex data than text, for example a picture or video. A picture can be considered to be more complex in the sense that it is harder to both describe and strictly interpret the content. An image is digitally constructed by millions of individual pixels, introducing difficulties regarding how to represent and compare pictures. Additionally, in terms of art, images can be intentionally created to be interpreted by the viewer, instead of conveying the message directly. Simply put, two different images containing the same object can be interpreted entirely different, which complicates any attempt of making images search-able. One way of avoiding these difficulties is to describe the picture with words, making the problem once again text-based. Besides being conceptually different, such an approach still has the need of a human being for interpretation, inhibiting any automatic organization and categorization. Exploring human perception, with the goal to teach a computer how such an interpretation is done, has created the research field of image analysis and recognition. Remarkable progress in both text and image recognition, e.g. search engines and facial recognition, motivates

adding a new layer of complexity. How do you organize and categorize multiple images that are shown rapidly in a sequence?

1.1 Video Content Analysis

A temporally consistent sequence of images forms a video. As said, an image or *frame*, has room for interpretation that can vary dependent on the viewer. The semantic value of a picture is part of what makes painting and photography an art form; a certain constellation awakens emotions and sentiments differently for different people. Not surprisingly, video inherits this property since it is, in fact, a series of images. Aside from the interpretation of each individual frame, additional information is added to the content by showing images in a sequence. Consider a video starting with a person jumping high into the air. Played normally, the viewer might get the impression of an athletic person, who can jump that high. If the order is reversed, it instead results in a person falling, luckily landing on his feet. Mixing the frames randomly on the other hand would most likely be interpreted as nonsense.

It is evident that video- and film-making is an artistic form of conveying a message, raising an emotional and semantic response. The sentiment is additionally, compared to images, dependent on the *temporal* context, i.e. how the sequence relate to a time-line. Besides these added complications in terms of interpretation, time also introduce a new dimension in terms of description. Where images or frames are structured by pixels, a video consists of a sequence of pixels, altering frame by frame. A frame of a video, with typical resolution and color (1920x1080, RGB-space) yields over 6 million pieces of information. Considering that a two hour video consists of over 10 million frames, one starts to realize that teaching a computer to interpret correctly is harder than it might sound. To complicate the learning even more we want to enforce the interpretation that coincides with human sentiment, otherwise people still have to make an effort themselves. Simply put, when trying to perform an image- or video recognition task, the choice of information to extract is of great importance.

The information contained in a single pixel is not descriptive enough to represent the content in an image, thus even worse at resembling a video. Even when two images, or videos, showing the exact same object is compared, pixel values vary depending on the lightning conditions, camera configuration, object orientation etcetera. This variation between frames may or may not contribute to the general understanding. Besides extracting the correct information, or correct *features*, the sentiment of a video vary with time as well. It is here the focus lie for this thesis project; to explore whether it is possible to teach a computer how to interpret video similar to human sentiment. A producer of a video much often intends to stir up feelings and convey messages to the audience. Even nonsense is conceived as nonsense. More common is to express events of a certain feeling or mood, such as a sad moment or a joyful dinner party. The aim of this project is to study the behavior and characteristics of how to create such a sentiment in video, with the purpose of teaching a computer to recognize these patterns automatically. Doing so requires the knowledge of multiple fields besides information extraction as introduced earlier. The next chapter will for that reason briefly mention relevant areas, as well as the outline of the remainder of this report.

2 Background

As hinted previously, the understanding of video for us humans involves time as a parameter, thus it can be assumed to affect computer vision as well. Depending on the number of frames included, the video is interpreted differently. When examining the temporal characteristic it is therefore useful to segment the video, to form milestones for the learning task. This motivates that, when analyzing video, segmenting the video into groups of frames helps in interpreting content accurately.

2.1 Video Segmentation

Segmenting an image is important for some image recognition tasks. Consider an image of a boat out on the sea. To analyze the color of said boat, it would be beneficial to remove the background. The only colors that remain belong to the properties of the object of attention. The representation of the image has therefore been reduced, at least a fewer amount of pixels have been selected. It is thus not surprising that segmentation affects the understanding of video as well. Even though segmenting each frame can be of use for video analysis, it is of greater purpose to segment the video with respect to time, namely group frames in a structured, meaningful manner. Forming a sentiment of a video is mostly done with respect to time, thus the less importance of a single frame. Maintaining the order of frames as well as playing the video forwards is obviously a requirement for correctly interpret or *classify* the event in the video. Recall the example of a video with a jumping person, being played backwards or forwards. By grouping too few frames, parts of the conveyed message will be lost, since the event or action in fact consists of more frames. Likewise, grouping too many frames will bring the event out of it's context, merging different and (maybe) unrelated content. Additionally, since the goal is making video content searchable, such segmentation has to be both automatized and properly structured, to be able to be organized and categorized in a representative fashion.

The key word of the last sentence is **properly**. What is a proper segmentation in order to teach a computer to provide a sensible semantic label? As mentioned, segmenting in terms of the different events and actions are desirable, but it does not seize the larger semantic value. Consider again a person jumping high into the air. Without knowing why the person is jumping, it is quite irrelevant. However if seeing beforehand, an object which the person is trying to reach, the sentiment changes. A useful segmentation would thus reflect what is searched for, making each video recognition task different depending on the intention.

2.2 Machine Learning and Classification

Since organizing and cataloging is task-specific, it is important to reflect on concepts that are characteristically relevant when searching for video. Media distributors, such as Netflix[1] or Amazon Instant Video[2], are increasingly popular as a result of faster and more reliable Internet connections, as well as improved hardware. Currently people watch TV-series and movies on multiple devices such as mobile phones, computer tablets and the like. Storing and streaming videos is mostly not a problem, however the cataloging of videos is

often done manually. A YouTube video is tagged with keywords describing the content, while streaming services tend to sort the content by genre. Apart from being a vague and broad concept, genre does not necessarily work as a measure of similarity between movies. Roughly said, the enormous information contained in a longer video production is only described by a few words describing the genre. Useful social network applications such as rating systems and comments help users to identify the properties of a video. Additionally, user behavior is utilized as a suggestion system, trying to find connections between users with similar preferences.

Analyzing videos, with the purpose of classifying the characteristics inside the video, would not only automatize cataloging and organizing, but also add a more specified ability to search for videos matching the user preference. By teaching computers, using machine learning, how to recognize certain properties, would add a supplement to the existing genre specification as well as create a way to compare video productions more thorough. An example of such concept could be to search for how much action a movie contains, or to recognize "feel-good" movies. In 2009, the media distributor Netflix announced the winner of a one million dollar contest[3]. The challenge was to beat their existing recommendation system by 10%. The prediction engine should estimate whether a user will enjoy a movie or not, based on how other users have rated other movies. This shows that there is both interest and practical use of the research included in this thesis project, besides yielding information about how the video structure affects human interpretation.

2.3 Report Outline

Towards an automatic video labeling system, multiple fields have to be explored. Before getting into the details of this machine learning assignment we will first visit the world of filmmaking. To gain knowledge about video production, chapter 3 reveal common guidelines for movies and TV-series. The secrets of video production along with this introduction yields enough knowledge to formulate a problem description in chapter 4. Once the assignment is set, chapter 5 will start to unravel the task at hand, by presenting relevant research and related work. Chapter 6, 7 and 8 describes the techniques chosen and used in this thesis, in terms of feature extraction, machine learning techniques and segmentation. We will then take a step back, summarize what we have learned, in the form of a methodology outline in chapter 9. The performance is evaluated along with results in chapter 10, followed by conclusions and future work in the last section; chapter 11.

3 Video Editing

The contained information within a movie is both vast and complex. Characteristics such as color, motion and alignment with respect to time, all contribute to how the video segment is perceived. Knowledge about the content is thus stored in both the individual frames and the change over time, frame to frame. In other words, investigation about spatial as well as temporal properties is of interest when analyzing video. Since video is human made, it is created with purpose of raising an intended response by the audience, once these properties

are processed by our brains.

When creating movies and series, it is important to present the visual information in such a way that the viewer's focus and attention is maintained. Psychologists suggests that humans believe that objects continue to exist even though it has disappeared behind an obstacle, known as *existence constancy* [4]. The perception of appearance and disappearance is important for apprehending objects and events, which dictates how video content needs to be presented to avoid confusion. In general, movies and series have to present discontinuous information in order to tell a story. The concept of *continuity editing*, a common guide for film editing, is to maintain the impression of continuity even though the content is occasionally discontinuous [5]. The next section will explain some techniques to achieve this, with the motivation that it holds useful information about how movies and series are made which will contribute in terms of recognition.

3.1 Continuity Editing

In filmmaking, a shot is defined as a series of frames that is recorded uninterruptedly, thus a continuous segment of frames. The discontinuities between shots are called cuts or shot boundaries, which are considered discontinuous in at least one out of three ways; temporal, object or spatial [5].

- **Temporal continuity**: Temporal continuity means that the time-line of the video is followed in real time. An example of a temporal discontinuity would be a jump in time, e.g. a flashback of memories.
- **Object continuity**: The properties of the objects shown is maintained between two shots. Unexpected changes of object properties, such as the color of a car, are considered to be object discontinuities.
- **Spatial continuity**: Spatial continuity refers to the same spatial setting, e.g. location. A typical and obvious discontinuity would be moving from indoors to outdoors.

Note that these concepts are defined for video editing and filmmaking, not image- and video analysis. Spatial information may refer to location for video production, while referring to pixels properties of an image in image analysis.

The shot often describe a single action, for example a person performing a jump into midair. Recall that such a description is insufficient for describing a larger event. To fully grasp the intention of a video, a sequence of shots has to be presented in a meaningful order, thus forming a larger video segment which is called a scene. Since both shots and scenes begin and end discontinuously in some manner, it is important to minimize how these disturbances affect the viewer. The following editing techniques and concepts, which are more elaborately explained in [5], has been selected as an example of how producers and directors may avoid confusion.

3.1.1 180 ° Rule

The foundation of the continuity system is the 180 ° rule[6], which is a guideline for camera placement and editing of a scene. More specifically it is a line, splitting the three dimensional space of the current setting. The line is set by filming

an *Establishing Shot* perpendicular to this axis, with the purpose of establishing the context for a scene. Any upcoming content in the same scene should be kept within the 180 ° arc of the established line. Following the 180 ° rule ensures that relative positions are preserved, minimizing the effect of the object discontinuity.

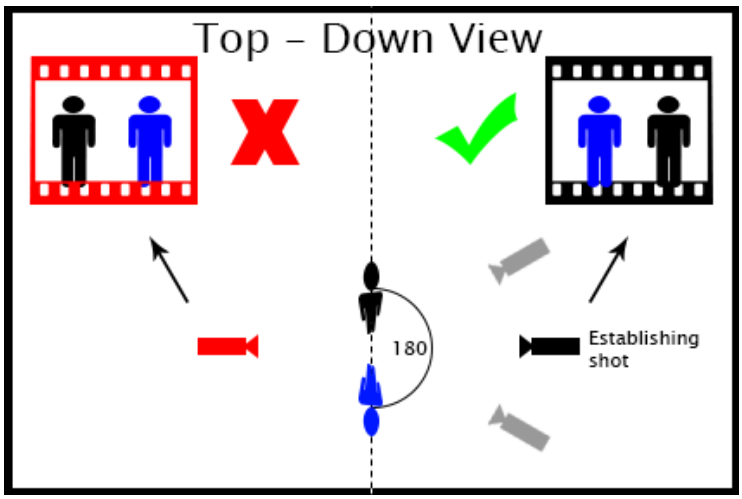


Figure 1: *Illustration of the 180° rule. The recording of the scene is established on the right side in the figure. By doing so, the spatial relationships have been set. Crossing the 180° set by the establishing shot and the characters in the scene, would reverse the positions in the scene, causing confusion.*

Imagine a dialog involving two persons as shown in Figure 1. The establishing shot shows the initial positions of the characters, relative each other and relative to the surroundings. A line connecting both persons will set the 180 ° allowed by this rule. The shots will then, most commonly, alternate between the persons depending on who is currently speaking. Crossing the line between two shots would reverse the relationships of the scene, i.e. the person on the left side of the screen will, in the next shot, appear on the right side and vice versa.

3.1.2 Matched-Exit/Entrance

Changing the location or setting between shots is as mentioned above a spatial discontinuity. This is however often needed, for example when following a traveling object from one location to the next. To create the illusion of spatial continuity, a matched-exit/entrance cut is used. Once an object exit the screen, it should enter as expected in the following shot.
Imagine a character traveling by car. If the car exit at the right side of the screen, it is expected to enter from the left side in the next shot. Even if significant time has lapsed between the shots (temporal discontinuity), the spatial discontinuity is perceived as less confusing if the expected entrance is used.

3.1.3 Fade and Dissolve

Temporal discontinuities are essential to movies and series, you occasionally want to skip time or present a character's memories. To minimize the confusion arising during cuts, certain methods are used to help the viewer apprehend the content. A common technique for letting the viewer know that time will pass is to use fade or dissolve effects. This means adding a gradual transition between either two shots or fade to a black frame, indicating the beginning of new content. Worth noting is that such effects creates expectations on the following shot, for example the viewer might expect that the main character's clothes change from one day to another.

Editing techniques, such as presented above, show how producers work in order to minimize the effect of the discontinuities present in video content. In one definition of shot properties, it is suggested that every shot can be partitioned into one out of eight different categories[5]. While cutting between shots affect how the video is interpreted, it is equally important is to know how to draw and maintain attention during the shots.

3.2 Attention

Filmmakers and editors intentionally directs the viewer's attention to achieve the desired effect, e.g. suspense or drama. Not surprisingly, some approaches of action recognition and video indexing include attention based models in an attempt to capture the essential visual information[7]. Without further explanation, the following list of visual features is considered to capture attention.

- **Abrupt appearances and disappearances of visual objects**[8]
- **Onset and noticeable motion**[9, 10]
- **Contrast or luminance changes**[11]
- **Apparent color changes**[10]
- **Looming stimulus (rapid size change)**[9]

Many of these often occur in movies and series, achieved in different ways. The recorded content itself can contain an event which captures attention, e.g. a sudden movement. In addition, one can capture attention by editing or using special effects.

3.3 Summary

Since film making and editing adapt to human perception, these kinds of characteristics can be expected to be true for most videos, at least for movies and series. Thus, using them for recognition feels natural. Commonly there exist eight different types of shots, used for building scenes. Each scene has the intention of telling a story and conveys a certain mood, which introduces a bias for film making. Certain methods (such as continuity editing) are used to properly convey the intended message, creating properties of movies and TV-series to be used in image- and video recognition. The remainder of this thesis will examine to what extent it is possible to recognize these patterns and properties, with the purpose of classifying the atmosphere or mood of a video sequence.

4 Problem Formulation

With the purpose of organizing and cataloging video productions, it is important to determine in what way the videos should be search-able. Classifying the video segments based on the stories and events, would result in a search yielding only movies telling basically the same story. While people tend to have a favorite movie or TV-show, most prefer to watch previously unseen content. A more practical measure of similarity is the resulting impression after watching the video. In other words, by classifying the mood or atmosphere of a video segment, a search is bound to suggest movies which will be interpreted similarly, possibly with an entirely different story. That being said, a scene can no longer be defined as a sequence of shots explaining a story.

Refer to Figure 2, which intend to illustrate the difference in segmentation. A shot is formed by the grouping of continuously recorded frames. An event, or an atmosphere, is formed as a sequence of shots. As the last row of the figure shows, the shot sequences for events and atmospheres does not necessarily align; an atmosphere can be changed in the middle of an event. For the purpose of this thesis, the scene delimiter is the change of atmosphere, instead of the event. Consequently we are now able to formulate the classification task at hand, along with an mathematical description, as well as initial expectations for such a classifier.

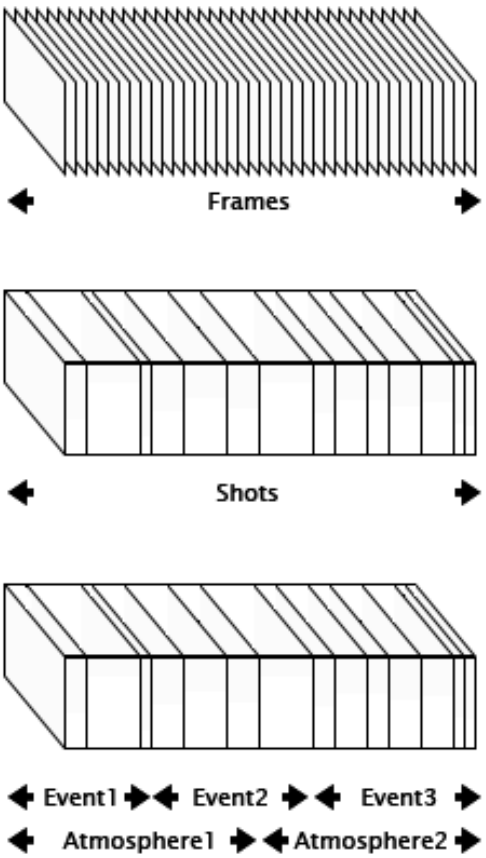


Figure 2: *Description of Frame-to-Scene segmentation. The frames of a movie can be segmented in to shots. A sequence of shots can describe an event, while another sequence set the atmosphere. These two different concepts of a "scene" do not necessarily coincide; an atmosphere can change in the middle of an event and vice versa.*

4.1 Goal

The purpose of the work in this thesis is to classify the sentiment given by a Hollywood production, more specifically a movie or TV-series. Such a video is filmed and edited to evoke the specific emotion or feeling that the creator is trying to convey. The viewer's sentiment will from this point on be called the *atmosphere* of the video. Editing techniques to achieve a certain atmosphere is expected to induce patterns and impressions within the video. By trying to find and learn these properties of a video, the following atmospheres will be classified:

- **Eventful:** A scene which includes a lot of motion, camera movement or rapid cutting. There exists a few different types of eventful scenes. One example is a fighting scene, often viewed as a video segment with high motion and frequent editing. Another would be a scene altering between

shots of many different events with the same temporal placement, i.e. events meant to be perceived as carried out simultaneously.

- **Gloomy:** Depressed and funereal moments will be labeled as gloomy. This covers the emotional state such as sad, as well as more general low-spirited ingredients. A good example would be either an event of total hopelessness or a more obvious occasion; a funeral.
- **Tense:** Scenes with the intention of creating suspense and tension such as a threatening conversation, gun stand-off and the like.
- **Joyful:** Festive events such as dancing and partying are considered joyful moments. Some video segments could also have a general "feel-good" feeling. It will mainly be used for longer sequences of joy, not just a single joke.
- **Introductory:** Often occurring story-building shots do not necessarily have any atmosphere at all, except for basic narration. Introducing characters or standard conversations that can be hard to label will be labeled as introductory, which might be a poor choice of words.
- **Emotion:** An obvious emotional scene is the romantic scene, however the label emotional will additionally include moments intended to touch the viewer emotionally.
- **Other:** This category will not be part of neither training nor testing. Parts of a movie or series such as an intro or credits will not be interesting in this thesis and is labeled for the sole purpose of deletion.

Realizing that labels of concepts like the above cannot be classified by the information contained in only one or a few frames, motivates that the video content in fact has to be structured in a more semantically meaningful manner. Partitioning a video automatically will require the extraction of information, or a set of descriptive features. It is not necessarily the same information that will be needed for classification, dividing the task slightly into two subproblems; segmentation and classification. Features will typically be extracted from each frame, while the classification refers to a collection, or segment of multiple frames. Thus segmentation introduces the need of transforming the frame based features into a group-of-frames representation, to properly prepare for teaching a machine to interpret the video. Do not be fooled, such a machine learning problem is challenging by itself.

4.2 Expectations

With this knowledge, and a somewhat more specified approach, some expectations start to develop. As mentioned, the creation procedure of filmmaking is expected to leave traces to be used for recognition. For example, a dialog between two persons in a movie is typically viewed as alternating shots of close-up images of the person speaking. These traits will hopefully be separable in terms of video content differences so that the atmosphere of a scene can be uniquely represented as a sequence of specific shots. The spatial and object properties of a video should be fairly straightforward to capture, since the change of e.g.

location or color is noticeable by just comparing two subsequent frames. Temporal elements are however harder to extract, for there are no evident way of how to relate a frame or video segment to the movie time-line.

The atmospheres to be recognized are based on sentiment which can be expected to affect the results both positively and negatively. A sentiment varies dependent on the viewer, which introduces a vagueness and ambiguity for learning. In addition it is expected that the analyzed videos are biased by the creator, e.g. the video's producer and editor. By assuming that content from the same TV-series are created similarly, any changes and correlation in the video data can be considered to contribute to learning. Consequently, an interesting analysis will be how well the classification *generalize*, i.e. the performance when trying to process dissimilar video content.

While all of the above sounds promising, maybe the biggest challenge will turn out to be how to represent a series of frames in a way that correlates with the labels correctly. Not only does this representation require proper information extraction along with a merged description (for multiple frames), but it also needs to be representative for all variations of the same atmosphere. This will add to the fact that computer vision is regarded to be a complex artificial intelligence task [12], which most often generalize poorly globally.

4.3 Formal Mathematical Description

Even though required information from each frame is to be determined, it is possible to structure the problem mathematically, at least as an initial outline. The information from each frame can be described as a row-vector, or feature vector $\mathbf{x}$. It is generated by concatenating the result of p different features, yielding M values as,

$$\mathbf{x} = [v_{11}, \ldots, v_{1q_1}, v_{21}, \ldots, v_{2q_2}, \ldots, v_{p1}, \ldots v_{pq_p}], \qquad \mathbf{x} \in \mathbb{R}^M, \tag{1}$$

where v_{pq_p} represents the *q:th* value of feature p. Notice that the number of values q_p may vary for each feature, e.g. histograms with various bin size. The dimension M of the feature vector is basically the resulting amount of information values extracted from all features. For a video consisting of N frames, one vector of features $\mathbf{x_i}$ is extracted from each frame i, creating a set of features X,

$$X = [\mathbf{x_1}, \ldots, \mathbf{x_i}, \ldots, \mathbf{x_N}]^\mathbf{T}, \qquad X \in \mathbb{R}^{N \times M}. \tag{2}$$

Earlier it has been discussed that the problem of video recognition requires some kind of segmentation of frames in an attempt to capture the temporal context. Regardless of segmentation or division of frames, it is still true that every frame will be given a class, or label, representing the atmosphere. Although a single frame does not contain the information of why it's labeled a certain way, each frame will at some point after classification be given a label, as part of a bigger concept. The set of labels Y, consisting of the corresponding label for each frame, can be formulated as

$$Y = [y_1, \ldots, y_i, \ldots, y_N]^T, \qquad Y \in \mathbb{R}^{N \times 1}. \tag{3}$$

The essence of what machine learning is trying to achieve is to learn a function F, with the purpose of finding correlations between the input X and output Y.

More specifically, assume that there is a function F that maps the elements of X into the range of Y,

$$F : X \to Y, \tag{4}$$

since Y represents the concept that is being analyzed. Each pair $(x, y) \in X \times Y$, constitute as an example of the behavior. The task is to find a hypothesis h, such that

$$h(x) = F(x), \qquad \forall x \in X, \tag{5}$$

in other words learn an approximation of F. After mentioning a couple of prerequisites for this thesis, we will start to unravel the details of how to find such function, with the intent of automatically organize and catalog video productions.

4.4 Prerequisites

Progress in the development of both digital video technology and new complex media platforms, allows the modern user to access high-definition video and audio almost everywhere. The trend is to extend the audio and video quality even further [13]. Consequently, storing and distributing of video benefits from advanced compressing and decompressing techniques and standards. An uncompressed high-definition video requires a bandwidth of 1.5 Gbps to be transmitted in real time [13]. Besides creating a field of research regarding storage and compression, it motivates that a framework is needed in order for us to analyze video productions. Erik Bodin has during the time of this thesis been researching into a image and video recognition closely related to this work, providing a collaboration in terms of video and audio rendering, as well as video analysis. The framework, developed in Java, allows the user to with simplicity construct ways of extracting information from a video. Since Erik is performing a classification analysis himself, the program aids in building classifiers, label video frames and visualize data. Throughout the rest of the thesis, this framework will be referred to as Fava: Framework for Automatic Video Annotation [14]. The power of being able to exchange knowledge and ideas, in a mutual environment, has built a well-equipped foundation for this master's thesis.

5 Research and Related Work

Instead of blindly jump into the vast high-dimensional pool of information hidden inside a video, struggling to find portions of structure and order, one can linger in uncertainty a bit longer. What if there already is a accepted way of segmenting video frames? Somebody ought to have at least extracted color information in an image before. As in all fields of research, it is vital to examine achievements in related assignments, searching for clues and hints of how to proceed with a given problem. Not only does such exploration provide the possibility of utilizing methods proven to work, it is also an aid for what to expect and suggests the focus of attention. The following section will dive briefly into previous research and commonly used concepts that can be useful.

5.1 Image Recognition

Automatic image recognition has become a popular field of research, as well as attractive for various practical applications, e.g. automatized surveillance systems. For the 10th year of annual conferences, the International Conference on Image Analysis and Recognition (ICIAR) received 177 papers from 36 countries presenting the latest research [15]. The scope of area of applications is large, covering the field of biometrics[16], medicine[17] and tracking[18], to name a few. A noticeable integration into our society is the use of advanced face recognition. Searching for faces in images automatically is one of the hardest recognition tasks, yet it can be achieved nowadays with an accuracy above 95% [19]. To account for images with unfavorable conditions, such as poor lightning, low resolution, unrepresentative camera angle, the algorithms in image recognition has become increasingly sophisticated [20]. While good accuracy is preferred, these complex methods still have to meet the application's limitations in terms of speed and computational complexity. Thus, scientists working in image recognition are also researching into ways of improvements regarding compression, image matching, searching and optimization[21, 15].

From the variety of image recognition tasks, there exist representative image descriptions to be utilized for the purpose of this thesis. For instance, a good image representation need to address the fact that images is a 2D-projection of our 3D-world. An object in an image can therefore vary in terms of scale, rotation and illumination, giving rise to complications for recognition [22]. It is in this sense important that any characteristic information, or feature, extracted from the frames in our video, account for such variations. Another thing to keep in mind is that some advanced features are too computationally heavy for use in videos; it is simply not possible for some algorithms to process every frame in reasonable time[20]. Since the task at hand is in fact video recognition, it is more interesting to read into more closely related research.

5.2 Action Recognition

Seeing the progress in image recognition, it is of course intriguing to extend the analysis to work for videos. While it is impressive to recognize, compare, sort, catalog and retrieve images, the use of action recognition in video is more connected to relevant human queries. Teaching a computer to understand the temporal properties of videos could, aside from analyze media broadcasts, aid humans in everyday situations. What first come to mind is robot vision and human-robot interaction[23]. More subtle examples of applications are sport analysis[24], security[25], medical aids[26] or behavioral understanding [27]. With focus towards video analysis, how is it that humans can interpret a simple action almost instantaneously [28]? The author of the latter draws a parallel between video frames and human visual perception, suggesting that most progress in action recognition uses too much information (number of frames). Additionally it is presented that a common way of analyzing videos are to extract a relatively local feature set, a few frames, to classify an event or action. The claimed problem, despite successful results, with such an approach is that the classification lags behind the observation. This basically means that to determine an action, these methods need to look into both the future and the past, i.e. delay the decision compared to the observation.

Regardless of the hint that the amount of collected information can be reduced for basic actions to be recognized, the progress in human action classification and behavioral recognition [29, 30, 31, 32] motivates that it is beneficial to segment and partition frames for various action and event recognition assignments. In addition, more closely related work regarding movies and TV-series [33, 34, 35, 36], all use shot segmentation as part of in larger machine learning task. The most closely related research is described in [37], where the task is to classify horror scenes in videos, based upon emotional perception. Not only does the author use segmented video sequences, it also provides experiments showing that audio features as well as emotional features increase recognition performance in this context.

All in all, for both image and action recognition, it is evident that there is valuable information hidden inside videos and their frames. Knowing what information to keep and how to represent it is seemingly dependent on what you are trying to learn. A common denominator for most of the mentioned work is the need of choosing a machine learning technique fit to process the specific data. Assuming that it is possible to extract information that correlates with the behavior of video we are studying, the next step would be to find a suitable way of learning these characteristics. In order to understand and make a valid choice of learning algorithms, first we have to go through some basics of machine learning.

5.3 Machine Learning

Patterns in data have been essential for human life for ages. In order to hunt animals for food, humans studied behavioral patterns of their prey. Interpreting environmental factors, such as weather or seasons, aided humans in how to successfully grow crops. Today it is ridiculously easy to gather and store data in all shapes and sizes. As the performance of hardware increases it becomes more important than ever to attempt to interpret the data:

> *We would all testify to the growing gap between the generation of data and our understanding of it. As the volume of data increases, inexorably, the proportion of it that people understand decreases, alarmingly. Lying hidden in all this data is information, potentially useful information, that is rarely made explicit or taken advantage of.[38]*

Studying data is presently used in many occupations, either by intention or unintentionally. While a statistician is employed to study data to make business changing decisions, a doctor base his assessment partly based on the experience given by treating other patients. In this report so far we have discussed the possibility of *automatically* learning from experience. The mood or atmosphere in a video is a concept that we are trying to enforce computers to learn, by considering the human perception being the ground truth. Note however, that machine learning can be used just as often for the purpose of studying patterns that machines notice, when humans do not, presenting the opportunity for people to make more informed decisions.

Differences similar to the above example, tells us that different tasks require different types of learning. The behavior to be learned is called the *concept*[38] and the output from a learning machine is said to be the *concept description*[38].

In our case, the concept is learning the atmosphere of a video by feeding examples based on video sentiment analysis.

5.3.1 Learning schemes

The existing conceptual differences split machine learning into four branches. Depending on what is desired to learn, or what the output from the learning machine should be, it is common to divide machine learning tasks into either *classification learning, association learning, clustering* and *numeric prediction*[38]. One of the main differences in each approach is what is given as input. Classification and association learning is taught by presenting training examples. Each training example consists of a set of features along with the correct class label. Classification learning slavishly searches for relations and patterns between features amongst the training examples, learning how they correlate with the given labels. For association learning, the labels are indeed given, however the learning scheme aims at finding associations not only decided by the labels, but any association in the feature space.

It is not always possible to provide the correct answer, supervising the learning. In cases where the structures of the data needs to be found in an unsupervised fashion one commonly use clustering. By studying the correlations and patterns between features, the input data is divided into clusters or regions, with examples sharing a similar feature representation. Another unsupervised technique is called numeric prediction where the output is not a class but a number. A model built with numeric prediction can be seen to provide the value of the class rather than the class itself.

5.3.2 Evaluation

Finding and learning the patterns is one thing. We have fed examples to our classifier, pressed play, taught the machine everything there is to know. But how do we see what is actually learned? Imagine that we use unsupervised clustering of all the frames of the video, with respect to their feature vectors. Would the results of the clusters be the same division as we intended? Often when using unsupervised learning, the common way to evaluate the result is to somehow visualize the connections and relations, study how well separated the data is; poke and prod too see if the patterns resembles anything useful. For supervised learning schemes at least there exist the possibility of validation, merely check the labels that have given as ground truth. However, by testing the learned classifier using the same data as for training, odds are that any calculation of error rate is misguiding. Say that, in our case, the computer is taught using a video of a TV-show that is only recorded inside, e.g. a sitcom. The classifier might predict close to 100% correctly when using the same data for training as for testing. What happens if feeding the classifier with a new, previously unseen example, like an outdoor scene? Probably, the classifier has no idea how to predict the label correctly. The performance of a classifying new data is usually known as how well a model, or classifier, *generalize*.

In terms of learning, it is apparently important to choose representative data, covering most of the possible situations for a concept. In addition we need to separate the input into a training set and a test set (at least). A learning technique suitable for the concept has to be found, and we do not yet know

what the input will consist of. This choice also has to account for other factors such as computational complexity, linearity of the data and so forth. Things get complicated quickly, why it is fortunate that there are other scientists working around us.

5.3.3 Related Work

Starting with the things we do know: the data is a video, a series of frames. It is expected that the temporal order will have an influence for learning, which will be supervised. The indication by an extensive summary of temporal video segmentation attempts, is that segmenting the video into shots is a representative way of describing temporal components [39]. It is strengthened by the fact that work regarding movies and TV-shows [37, 34] indeed uses shot segmentation as the descriptive unit. Regardless of objective, clustering is a possible way of describing similarity between shots [33, 35]. The temporal understanding or semantically larger concepts, can be analyzed by studying the ordering and altering of individual shot types [36, 33]. Sequential data in form of numbers, letters or types, is closely related to other fields, e.g. text classification tasks or DNA-sequence recognition. The study of pattern is sequences is thus not new and progress has been made using Support Vector Machines (SVM) along with string kernels [40, 41]. Glancing back at the most similar report, horror scene recognition, they in fact also make use of Support Vector Machines, although not together with string kernels [37].

5.3.4 Summary

By briefly touch upon recent successes in machine learning, a lot has been provided in terms of assignment outline and focus of interest. It is evident that the possible approaches are many and that the field of video recognition is huge. Since a thesis is a project with limited time, it is of the essence to make choices accordingly. Not only is the implementation of some algorithms time consuming and advanced, it simply would not be possible to test and evaluate every promising method qualitatively. Reading the related research articles provided valuable insight in the necessary parts of the assignments in this thesis. This allows us to specify the work flow a bit more detailed than earlier. From the research it is suggested that we:

- Extract features that characterize the content both spatially and temporally.
- Segment the videos into group-of-frames structures, e.g. shots.
- Find suitable algorithms for learning, where the Support Vector Machine (SVM) is a common choice.

From the knowledge about the assignment we can furthermore state that:

- Features will have to be designed and analyzed to match the searched atmosphere labels.
- Labels will be set based on the sentiment given from watching the video, i.e. supervised learning will be used.

As can be expected, many of the items listed involve plenty of various concepts, techniques and analysis that can be utilized. The more technical upcoming sections will present the knowledge needed in order to understand the final setup, rather than explain all possible principles.

6 Feature Extraction

No matter what the structure the videos are segmented into later on, the information contained in each frame has to be extracted. The characteristics of an image, i.e. color information, contrast or luminance, is commonly known as image features. Given an image there is an abundance of possible features to extract. Color features, edges, shapes, statistical properties [42] and much more can be computed for every pixel in the image. Furthermore one could examine portions of the image separately, either by a fixed window size or as the result of some kind of segmentation. It is possible to add filters to get rid of irrelevant information, creating yet new representations. The abundance of high-dimensional observations is a problem for many applications besides image- and video analysis, e.g. traffic prediction or advertisement optimization [43].

Assume that each pixel in an image uses three channels of colors, where the value of every channel range from [0 255]. Considering how an image with resolution 1920×1080 result in over 2 million pixels, all possible images span a huge feature space. The image data can be said to be drawn from any part of this high-dimensional space. However, imagine this short analogy. A flat surface, e.g. a pane of glass, resides in the physical space (3D). Regardless of how the pane is rotated or positioned, the internal relations of the pane remain intact. Analogously, assume that two similar pictures, for example two similar faces, are drawn from the same structure within the feature space. This would imply that the concept of faces could be expressed in a subspace of lower dimension, similar to the pane. The assumption that there exists a low-dimensional subspace containing only the structure that corresponds to a certain concept, is known as the manifold assumption. Learning the structure of the manifold would thus entail a significant dimensionality reduction.

Most successful high-level computer vision advances have been achieved by extensive analysis of features, combined with great domain knowledge, rather than explicitly learning the properties of a certain manifold. In a sense, these features could be seen as a lucky mapping to the manifold. Imagine again the face recognition task, and furthermore suppose that a feature set has been found, along with a proper learning algorithm. The classifier is trained with images containing only faces of women, and performs well for female faces. What happens if feeding the classifier a manly face? The performance of the classifier will depend heavily on how well the features follow the structure of the manifold. The training examples given will work as samples of the surface, while the classification algorithm can be seen as the interpolation method between samples.

With a large enough set of training examples, along with the proper set of features, this structure could be learned at least locally. How well a trained model will generalize, depend on how accurate the interpolation between examples turn out to be, with respect to the manifold. The authors in [12] suggest that the use of non-linear mapping, e.g. kernels, is most often only a local gener-

alization, which assumes that the target function is smooth enough. However in many tasks, for example computer vision, this assumption of a smooth function is not enough to handle the complex nature of the data, i.e. the curse of dimensionality. This basically means that the target function to learn is increasingly complex with the number of task relevant factors, thus increasingly complex for increasing dimensionality. The way to adjust is to either gather more examples, or construct features that characterize the concept in a better way.

6.1 Feature Design

We have settled that the features need to be informative for the task at hand. Equally important for features is that they are also invariant[42]. The extracted features in any machine learning task can be seen as the different variables involved in making a prediction[43]. It is possible that there are variables that should not affect the feature response. In text recognition, a word is still the same despite different coloring of the letters. Many irrelevant variables can be sorted out by domain knowledge. A video may vary in terms of resolution and quality, length or frames per second, to name a few. None of these changes should affect the output of a feature, assuming the image is the same. In addition, the content of video can differ by rotation of camera, scale, illumination etc. For images and videos it is therefore useful to build robust features for irrelevant changes, e.g. resolution. Commonly one desire invariance to image translation, rotation and scale [22, 44, 45].

The frame-to-frame differences of image histograms has been proven[46] to be well characterizing videos, motivating the use of histograms, which are invariant to translation and rotation [45]. The histogram is an estimate of the probability density function for a given data set $\mathbf{x} = [x_1, \ldots, x_n]$, by counting the number of observations of an occurrence. More specifically, suppose that k discrete intervals are formed as

$$k = \frac{b-a}{h}, \tag{6}$$

splitting the range of the data into k *bins*, with length h. The histogram is then a piecewise constant function:

$$f(x, \mathbf{p}) = \sum_{i=1}^{k} p_i \Gamma_i(x), \qquad a \leq x \leq b, \tag{7}$$

where

$$\Gamma_i(x) = \Gamma^{(h)}(x - ih), \tag{8}$$

denotes the rectangular function

$$\Gamma^{(h)}(y) = \begin{cases} 0, & y < 0 \\ 1, & 0 \leq y \leq h \\ 0, & y \geq h \end{cases} \tag{9}$$

and p_i is the number of occurrences in bin i. Unnormalized, the sum of all bins in the histogram results in the total number of observations. Using the normalization condition

$$\mathcal{N}(\mathbf{p}) = \int_a^b f(x, \mathbf{p})dx = 1, \tag{10}$$

infer that p_i is the probability for bin i. A normalized histogram can thus be seen as an estimated probability distribution, which is scale invariant [44].

6.2 Static Features

The histogram of an image, i.e. color distribution, can be separately computed for in each frame in the video. The purpose of such computations is to characterize each individual frame, which will be called static features. These features are static in the sense that only one frame is included for the computation of the feature response. This individual description of a frame can of course be part of inter-frame calculations at a later point, however each frame is in this section seen as independent images. We will start by explaining how to extract one of the most basic characteristic; color information.

6.2.1 HSV Histogram

Color can be represented in various color space systems [47]. As the application in this thesis revolves around temporal properties of video, the colors of the frames are less important than the frame-to-frame color differences. The choice of color space affect the information retrieved from frame-to-frame histogram differences [48]. Consider the default of many display systems; the RGB color space, where colors are described as a combination of the colors red, green and blue. Closely related by a simple conversion is the cylindrical HSV (Hue, Saturation, Value) color space. Imagine an image filled with a single color, with a shadow on it. In the RGB space, the area including the shadow will have a very different description compared to the rest of the image. For HSV, the image intensity is separated from the color information, thus the shadow will not affect the color (or hue) much. The point is that the choice of color space affects both how the frame histogram behave, as well as the inter-frame differences.

Additional color spaces has been invented for other purposes, e.g. model human perception (Munsell color system)[49]. A report evaluating how the choice of color space affect a video segmentation algorithm, show that HSV is an effective color space representation [48]. Creating a HSV histogram estimating the color distribution of the pixels in the image will thus be the color information feature in this project.

6.2.2 Full Distribution Histogram

According to [37], the differences and similarities from frame-to-frame comparisons are important for any video segmentation attempt. When the video is recorded continuously, within a shot, the distribution of color or edges changes slowly. Any abrupt changes, as the peaks of Figure 3, indicate a possible shot boundary. For a HSV-histogram, these abrupt differences arise also from larger differences in intensity, falsely suggesting a cut. The hue is otherwise robust to intensity changes, however once the saturation or value are very low, hue becomes unstable [39]. This motivates the creation of what we call "full distribution histogram". Each bin in the histogram symbolizes one combination of a

hue, saturation and value together. Suppose a HSV-histogram contains 10 bins for each of hue, saturation and value. The possible combinations of these bins combined results in $10 \times 10 \times 10$ bins.

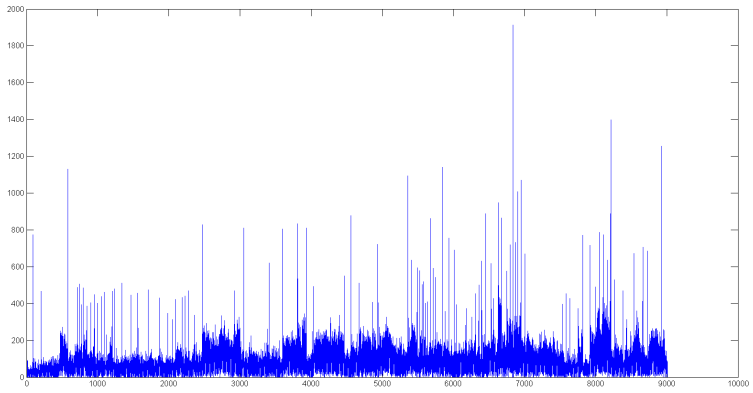


Figure 3: *Example of edge histogram differences, where the peaks is due to more abrupt changes from frame to frame. The y-axis is the difference magnitude, while the x-axis represent the frame identification number, i.e. time.*

6.2.3 Edge Histogram

Edges in images are both conspicuous and important for human perception [44]. It contains information about the image that can not be described by neither color information histograms nor texture features. An efficient way of representing the edge distribution in an image is to compute the edge histogram. The most standard[50] edge histogram for video content, contains the distribution of five types of edges, shown in Figure 4. The image is searched for vertical, horizontal, 45-degree, 135-degree and non-directional edges.

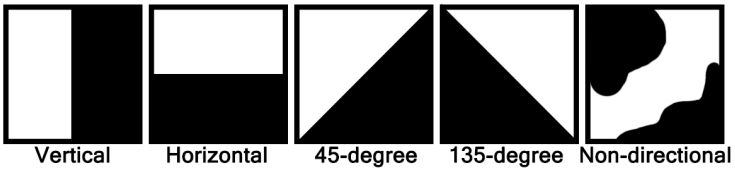


Figure 4: *Illustration of the five different edge types searched for in the frame.*

The representation of the edges in an image is achieved by dividing the image into subregions, either local, semi-global or global as can be seen in Figure 5. The different images (a - e), show different partitioning for edge extraction. Each subregion contribute with five bins to the total histogram, one for each type of edge. In (a), edges are computed for the entire image, resulting in 5 bins. In similar fashion, regions of the image is analyzed independently in a local (b) and semi-local (c - e) manner, contributing to a total edge histogram.

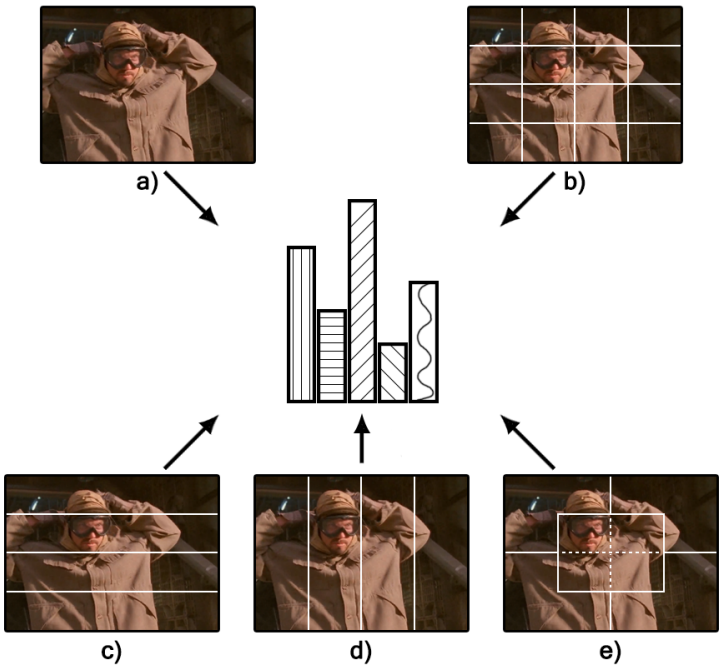


Figure 5: *Different partitioning of an image. Each configuration of pixels contribute with five histogram bins, one for each edge type.*

By counting the occurrence of the differently directed edges, a total edge distribution histogram is created by concatenating the histogram of each subregion. More details how an edge is found and how the directions are determined is described in [44]. The setup of the edge histogram used in this project results in a histogram with 150 bins, as suggested by [44]. Firstly 16 subregions to represent the local distribution, creating 16×5 bins, secondly 13 subregions for a semi-global distribution (13×5) and finally the global distribution, counting the edges present in the entire image (5 bins).

Each of the features presented above have been extracted directly from the raw data, i.e. the pixel information. It is however often valuable to observe how the data behaves when transformed into other spaces. As a basic example of such, consider data consisting of Cartesian coordinates (x, y). For certain geometries, a transformation into polar coordinates (r, θ), noticeably simplify relations. Another common transformation is the Fourier Transform.

6.2.4 Fourier Transform

The general idea of the Fourier Transform (FT) is to express a complicated function, or wave, as a combination of more simple waveforms; sine and cosine. This fact, and most information written is this section can be read in [51]. Since an image or frame can be seen as a value at a coordinate (x, y), i.e. a function of two parameters x and y, the same mathematical transformation can be performed on images. Instead of analyzing the properties of an image directly by its pixel properties, it is thus possible to study the image in another domain. By performing a 2-D Fourier Transform, the image is translated as a

summation of sine and cosine waves. In other words, the image is transformed from the spatial domain to the frequency domain, more thoroughly expressed as

$$F(u,v) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-2\pi i(xu+yv)}dxdy, \tag{11}$$

where the coordinates (u, v) are the spatial frequencies. The inverse transformation

$$f(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} F(u,v)e^{2\pi i(xu+yv)}dudv, \tag{12}$$

show that an image can be constructed by a combination of $e^{2\pi i(xu+yv)}$, weighted by the function $F(u, v)$. We know that $f(x, y)$ in the context of images is a real value, but the FT is in general complex. To process and reason about the frequency domain more easily, the Fourier transform is represented by it's magnitude and phase. The magnitude can be seen as a measure of occurrence (or strength) of a certain frequency, while the phase symbolize the frequency direction. An important realization from the last sentence is that the magnitude holds properties that can be similar for two images showing entirely different objects. The phase is crucial for proper reconstruction of the original image, however it is less useful for image comparison. With the motivation that the phase content contain too much details of just a specific image, the phase content of the Fourier transform will not play any role for the features in this project.

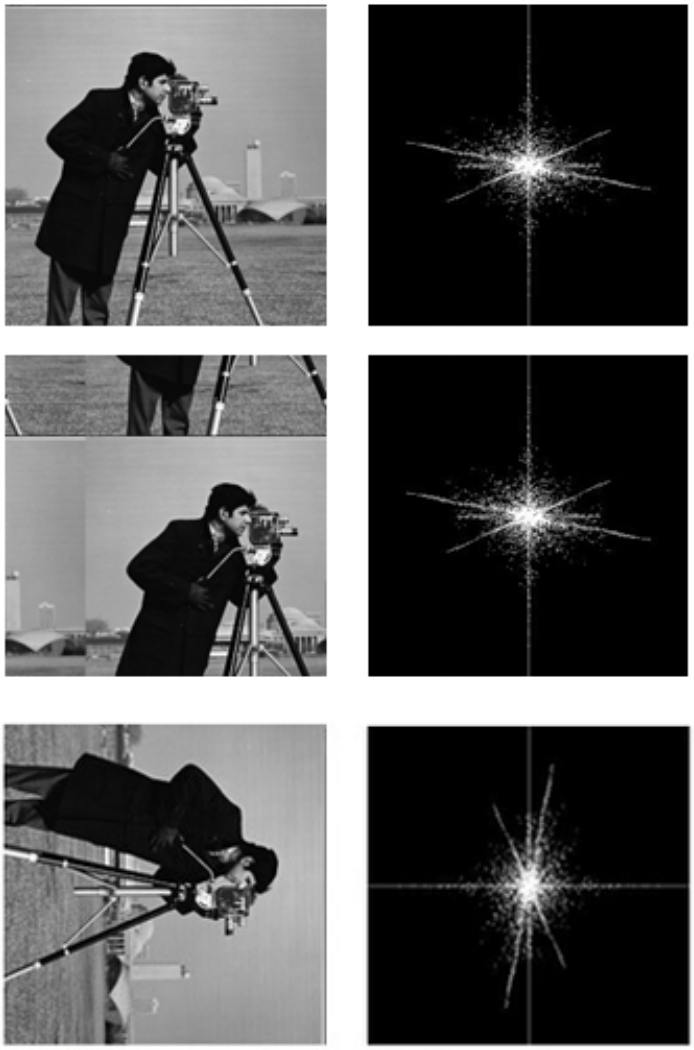


Figure 6: *Images illustrating important properties of a Fourier transformed image. First column: Image manipulations. Second column: Resulting magnitude spectrum. Translation does not affect the magnitude of the Fourier transform, and a rotation of the original image entail a rotation of the spectrum. Image reference: MUST Creative Engineering Laboratory (http://lab.must.or.kr)*

Studying the magnitude yields properties within an image regarding strong contrasts, sharp edges and shapes. Imagine a sharp edge of a gray-scale image. Crossing the edge require in a rapid change of gray values, consequently resulting in a high frequency response. As contrast and edges in an image is, as said earlier, vital in human perception, the magnitude spectrum holds interesting image properties. The component $F(u = 0, v = 0)$ is placed at the center of the magnitude image. Low frequency components are shown close to this center, while higher frequency components increase the distance to the center. Furthermore, the Fourier transform introduce other interesting properties for

image processing, e.g. translational and rotational properties. Figure 6 show some example of image manipulation with the resulting magnitude spectrum after performing the Fourier transform. As can be seen, a rotation in the image induces a rotation in the Fourier space. Other manipulations such as shifting the center of an image do not affect the frequency response. Erik Bodin[14] constructed a feature attempting to collect such information contained in the magnitude spectrum by measuring the spread and rotation of the frequency response, that has been used for this thesis as well.

6.3 Dynamic Features

The features so far have only extracted information included in each image, with no respect to the temporal nature of video. Contrary to the static features, there are some features that require a larger segment of frames to make sense, e.g. audio features. Even though audio features may be able to be extracted from a single frame, audio naturally span over larger time in the form of music or speech. Similar to audio, multiple frames is needed to be able to determine the motion or activity in the image. Even though each frame is ultimately given a feature value, the calculation for dynamic features involves multiple frames.

6.3.1 Optical flow

Recall the video production background, where it was stated that one of the key factors attracting attention in videos is motion. To determine motion, it is not surprising that more than one frame is needed in the calculations. The measurement of motion will for this work be based on optical flow, more specifically the optical flow presented in [52]. The idea of the chosen optical flow calculation, is to use the gradients of image brightness both spatially and temporally, in order to estimate the velocities of the motion. Let the brightness of a frame at time t, at position (x, y) in the image, be called $B(x, y, t)$. Assuming that brightness varies smoothly from frame to frame, the initial constraint is

$$\frac{dB}{dt} = 0. \tag{13}$$

Let the velocities $u = \frac{dx}{dt}$ and $v = \frac{dy}{dt}$ represent the spatial velocities. By applying the chain rule of differentiation, we end up with a single equation

$$\frac{\partial B}{\partial x}u + \frac{\partial B}{\partial y}v + \frac{\partial B}{\partial t} = B_x u + B_y v + B_t = 0, \tag{14}$$

consisting of three partial derivatives and two unknown velocities u, v. Approximating derivatives numerically is today common knowledge, by utilizing finite difference calculus[53]. The derivative of a function $f(x)$ in infinitesimal calculus is defined as

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{15}$$

for a infinitesimal step h. For numerical methods, the data are N equally spaced samples of a function, namely $f(x = a), f(x = (a + h)), \ldots, f(x = (a + Nh))$, with a finite step size h[53]. Since it is not possible to reach the limit, any

finite difference equation is an estimate of the actual derivate. While there exist multiple forms of computing finite differences, the partial derivatives in this thesis is estimated as

$$\begin{aligned}
B_x = \frac{1}{4}[&B(i,j+1,k) - B(i,j,k) + B(i+1,j+1,k) \\
&- B(i+1,j,k) + B(i,j+1,k+1) - B(i,j,k+1) \\
&+ B(i+1,j+1,k+1) - B(i+1,j,k+1)], \\
B_y = \frac{1}{4}[&B(i+1,j,k) - B(i,j,k) + B(i+1,j+1,k) \\
&- B(i,j+1,k) + B(i+1,j,k+1) - B(i,j,k+1) \\
&+ B(i+1,j+1,k+1) - B(i,j+1,k+1)], \\
B_t = \frac{1}{4}[&B(i,j,k+1) - B(i,j,k) + B(i+1,j,k+1) \\
&- B(i+1,j,k) + B(i,j+1,k+1) - B(i,j+1,k) \\
&+ B(i+1,j+1,k+1) - B(i+1,j+1,k)],
\end{aligned} \tag{16}$$

where $B(i,j,k) = B(x,y,t)$ for two subsequent frames.

Knowing the partial derivatives in Eq.(14), the remaining problem is to solve one equation containing two unknown variables. Skipping the details of assumptions and additional constraints discussed in [52], the final solution is an iterative scheme

$$\begin{aligned}
u^{n+1} &= \bar{u}^n - B_x[B_x\bar{u}^n + B_y\bar{v}^n + B_t]/(\alpha^2 + B_x^2 + B_y^2), \\
v^{n+1} &= \bar{v}^n - B_y[B_x\bar{u}^n + B_y\bar{v}^n + B_t]/(\alpha^2 + B_x^2 + B_y^2),
\end{aligned} \tag{17}$$

estimating the velocities u and v. The variables $\bar{u}$ and $\bar{v}$ is the local average of neighboring pixels, and α^2 an error magnitude due to approximations and noise.

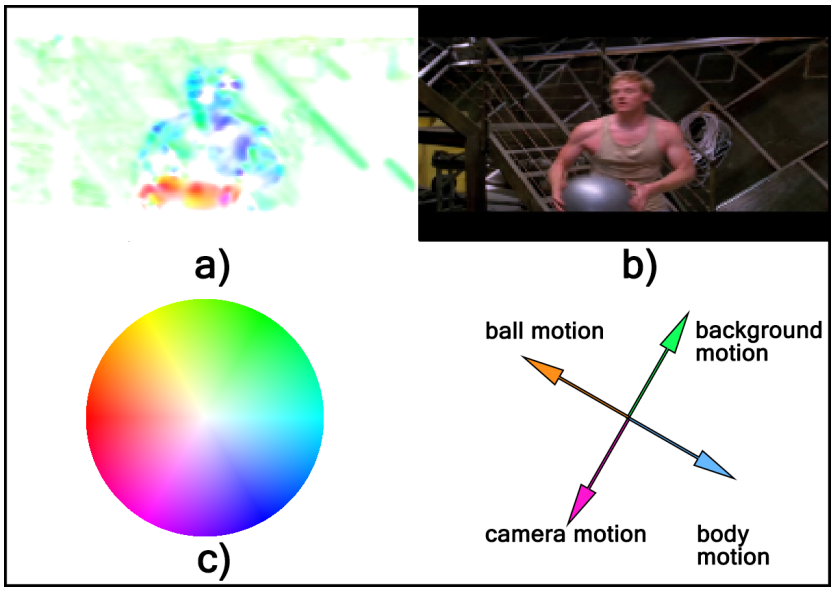


Figure 7: *A visualization of the optical flow feature. The color represent the direction of the motion, according to the HSV color wheel. The intensity represent the magnitude of the motion. a) Resulting optical flow. b) Original frame. c) HSV colorwheel.*

From the velocities u, v in the x- and y-direction it is possible to calculate a magnitude along with an angle for each pixel, symbolizing the strength and direction of the motion. To visualize the results of the algorithm, Figure 7 show the motion of an image in the HSV color space. The motion angle is represented by the color angle (hue), and the intensity (value) by the motion magnitude. In Figure 7, pixels moving left are shown as red, while pixels moving right are colored with cyan. The motion of the camera is south-west, causing the background pixels to move towards north-east. The person in the frame is moving to south-east, but the ball he is carrying is headed north-west.

Regardless of good performance for the optical flow feature, we stated multiple times earlier that a feature can not realistically be represented by a value for each pixel because it would result in too much redundant information. Now two values per pixel have been introduced, magnitude and angle of motion, extending the amount of information even more. Still, the motion of the video intuitively feels way too important for any video interpretation to be discarded, leading us right into the next feature.

6.3.2 Motion Attention

So how can one utilize the measurement of motion representatively? Reading the beginning of this section would suggest using yet another histogram. However, a histogram of for example motion orientation will still be a bit too specific to a certain situation, object or motion. In addition, any camera movement horizontally (which is common for video productions) of a static scene will most likely result in a very similar orientation histogram. Imagine sweeping the camera from left to right, e.g. showing the setting and furnishing of a room. The

optical flow will interpret this as if every object in the room is moving to the left, as if the furniture is moving. Humans will however not interpret this as walking furniture, but merely a change of camera angle. In an attempt of tuning the motion calculation to match human interpretation, a motion attention model has been developed in [7]. Besides using a different motion calculation than optical flow, the same methods can be utilized with some modifications. They use larger blocks of pixels instead of representing motion for each pixels, thus slightly different calculations, especially when calculating the entropy. The article discusses the problems of distinguishing camera motion from object motion, as well as a dimensionality reduction to finally represent visual motion attention with a single value.

The concept is to let the optical flow or any other motion vector field pass through three functions constructed to highlight and suppress different occurrences such as irrelevant camera movement. Recall that every pixels have been assigned a magnitude (strength) $M(x, y)$ and a phase (direction) $\Phi(x, y)$. The intensity function $I(x, y)$, basically the magnitude,

$$I(x, y) = M(x, y), \tag{18}$$

simply represent the motion energy. The next function utilize the phase histogram H_Φ, where each bin is an angular interval, describing the distribution of motion directions. By normalizing this histogram, each bin b can be seen as the probability $p(b)$ of a phase. The spatial consistency $C_s(x, y)$ is measured as the (binary) entropy of said probabilities, namely

$$\begin{aligned} C_s(x, y) &= -p(b_{x,y}) \log(p(b_{x,y})) - (1 - p(b_{x,y})) \log((1 - p(b_{x,y}))), \\ p(b_{xy}) &= \frac{H_\Phi(b_{x,y})}{\sum_{i=1}^{n} H_\Phi(i)}, \end{aligned} \tag{19}$$

where n is the total number of bins in the phase histogram. The bin $b_{x,y}$ refers to the bin that the pixel located at (x, y) has been assigned. For our temporal consistency function $C_t(x, y)$, the same phase histograms are computed but for multiple frames. Normalizing the histogram bins in terms of orientation during L frames, show the probability of an angle or phase over time. The temporal function can in similar fashion be expressed as

$$\begin{aligned} C_t(x, y) &= -p(b_{x,y}) \log(p(b_{x,y})) - (1 - p(b_{x,y})) \log((1 - p(b_{x,y}))), \\ p(b_{xy}) &= \frac{H_\Phi^L(b_{x,y})}{\sum_{i=1}^{n} H_\Phi^L(i)}, \end{aligned} \tag{20}$$

where H_Φ^L now is a phase histogram computed over L frames.

Now we have three matrices with the same amount of values as pixels in the image. The motion intensity generally correspond with human attention; a high motion intensity attracts more attention, while a low value of $I(x, y)$ suggest frames with static content. As mentioned, not all high motion is of interest to the viewer, leading us to the two other, entropy based functions. The Shannon entropy is a measure of uncertainty, defined as [54]

$$S = -\sum_n p_n \log p_n, \tag{21}$$

where p_n in our case is the probability of bin b. A low entropy corresponds to being certain and ordered. When the spatial entropy is high, the uncertainty is high indicating that the motion is either incoherent or disordered. Camera movement, which tend to be either horizontal or vertical, result in consistent and homogeneous optical flow, which can be utilized to separate object motion from camera motion. A high temporal entropy would thus suggest that it is in fact object motion, not camera motion.

The three functions are combined into a motion map $M_{motion}(x, y)$ as

$$M_{motion}(x, y) = I(x, y)C_t(x, y)[1 - I(x, y)C_s(x, y)], \tag{22}$$

where it can be seen that disordered temporal motion (object motion) contribute to larger values for the motion map. In the same way, disordered spatial motion will decrease the motion value. The final value of motion attention A is calculated as

$$A = \sum_x \sum_y M_{motion}(x, y)/N, \tag{23}$$

where N is the number of pixels.

6.3.3 Audio Features

Suggested by the results of related work, audio can be used as part of how to characterize a video [37]. As for video frames, the temporal alignment and order is essential to interpret the sound as intended. Assume a movie playing at 24 frames per second (FPS) with background music, sampling sound at a rate of 48000Hz. The resulting samples per frame is

$$\text{Samples per frame} = \frac{\text{Sound sample rate}}{\text{FPS}} = \frac{48000}{24} = 2000, \tag{24}$$

which is the sound for $1/24 \approx 0.0417$s of video. Analyzing 2000 samples representing 40ms will not be enough to characterize the sound as i.e. music or speech. In addition, since the audio- and video signals are processed separately, possible delays at each channel require audio-video synchronization [55]. It is obvious that the extraction of audio features differ noticeably from image processing. Luckily, with the intention of eliminating duplication of effort regarding audio extraction, there exist a framework for calculation of audio features; jAudio [56]. Since the library meets the requirements of Music Information Retrieval (MIR) researchers, it is both effective and reliable to use jAudio rather than developing new algorithms. Not only have they chosen representative algorithms for machine learning toolkits, it also provide requirements such as how many frames of sound to process. Without further introduction, the features used for this assignment are described and referenced properly in [56], leaving us with 19 ways of characterizing the audio content of the video:

Name	Short description
Root Mean Square (RMS)	A measure of the signal amplitude
Linear Predictive Coding	Calculates linear predictive coefficients of a signal
Zero Crossings	A measure of the pitch and noisiness
Method of Moments	Calculates the statistical moments of the spectrograph
Compactness	The beat sum, measuring regular beats
Strength of Strongest Beat	How strong the strongest beat is compared to other possible beats
Mel-Frequency Cepstral Coefficients	The Cepstrum coefficients from the magnitude spectrum
Spectral Centroid	Measures the center of mass of the power spectrum
Spectral Rolloff Point	Indicator of the skew of the frequencies present in a window
Spectral Deviation	The magnitude spectrum variance
Harmonic Spectral Centroid	Spectral centroid variation with respect to peaks
Harmonic Spectral Smoothness	A peak based calculation of smoothness
Strongest Frequency Via FFT Max	Find the strongest frequency component
Strongest Frequency Via Spectral Centroid	Strongest frequency component based on the spectral centroid
Strongest Frequency Via Zero Crossings	The strongest frequency with respect to zero crossings
Harmonic Spectral Flux	Correlation between adjacent peaks
Relative Difference Function	The logarithm of the derivative of the RMS
Spectral Flux	Measures the amount of spectral change of a signal
Fraction of Low Energy Frames	How quiet a signal is, relative to the rest of the signal

6.4 Similarity Measures

The features explained above provide multiple ways of describing the content in a video frame, both audio- and visual content. The hope is that the combined

information from all areas will be representative for the video. However good the representation is, a recurring fact is that frames have to be merged into larger segments to analyze the video content any further. This poses two problems, namely how to merge frames and the measure of similarity. For both problems it is important to note that each feature behaves differently, motivating the need of using proper measure of similarity. Recall the feature vector described by Eq. (1) in section 4.3. The representation concatenates the output of each individual feature, describing the frame as a single vector. This vector contains mixed concepts, e.g. probability distributions from one feature next to a single measurement of another. In order to compare the characteristics of two frames accurately, it is preferable to continue to handle each feature independently, instead of comparing the entire feature vector. In this thesis, most features output either a histogram or a single value. In order to discuss the similarity measures for histograms, some definitions need to be introduced. Let H_1 and H_2 denote two different histograms consisting of b bins, and

$$C_1 = \sqrt{\frac{N_{H_2}}{N_{H_1}}}, \qquad C_2 = \frac{1}{C_1}, \qquad N_{H_1} = \sum_{j=1}^{b} H_1(b), \qquad N_{H_2} = \sum_{j=1}^{b} H_2(b), \tag{25}$$

where we can note that $C_1 = C_2 = 1$ for normalized histograms. This definition, as well as much of the following explanations can be found in [45].

6.4.1 L_1 distance

The L_1 distance measures the difference as a sum of absolute bin-to-bin differences, mathematically described as

$$d_{L_1}(H_1, H_2) = \sum_{j=1}^{b} |C_1 H_1(j) - C_2 H_2(j)|, \tag{26}$$

which can be normalized to be in range [0 1] if divided by $2\sqrt{N_{H_1} \cdot N_{H_2}}$. Using the L_1 distance for features yielding only a singular value simply result in the absolute difference. The distance measure is also called a taxicab metric, or the Manhattan distance, because of the analogy to a taxi driving in the rectangular blocks of Manhattan.

6.4.2 L_2 distance

The "distance by air" is called the Euclidean distance or the L_2 distance. Compared to the L_1 distance, it measure the distance as the square-root of the sum of squared bin differences. The formula for the L_2 distance

$$d_{L_2}(H_1, H_2) = \sqrt{\sum_{j=1}^{b} [C_1 H_1(j) - C_2 H_2(j)]^2}, \tag{27}$$

can be normalized to the range [0 1] in the same way using $2\sqrt{N_{H_1} \cdot N_{H_2}}$. Note that for histograms with only one bin, a single value, both L_1 and L_2 return the same difference.

6.4.3 χ^2 distance

Quite different from both measures is the χ^2 distance. Remember that a normalized histogram can be seen as a probability distribution. The purpose of the χ^2 distance is to compare the histograms, determining if they belong to the same distribution function. More specifically the distance is given by

$$d_{\chi^2}(H_1, H_2) = \sum_{j=1}^{b} \frac{[C_1 H_1(j) - C_2 H_2(j)]^2}{H_1(j) + H_2(j)}, \tag{28}$$

which is normalized by $N_{H_1} + N_{H_2}$ to the range [0 1]. Since the distance measure is based upon probability distributions functions, it is hard to describe what the calculation represents for single values. Determining if two values belong to the same distribution can simply not be achieved by only one value from each distribution.

6.4.4 Edge Histogram (EH) distance

Edge histograms are developed for use in image- and video analysis. As part of the analysis of this feature in terms of image retrieval, a specific distance measure is suggested [44]. Recall that the total edge histogram included three different distributions; local, semi-global and global. For histogram H_i we define the local part as H_i^L, the semi-global as H_i^S and the global as H_i^G. The distance measure is expressed as

$$d_{EH}(H_1, H_2) = \sum_{L} |H_1^L - H_2^L| + 5 \sum_{G} |H_1^G - H_2^G| + \sum_{S} |H_1^S - H_2^S|, \tag{29}$$

giving more weight to global edge differences. Besides pointing out the strength of choosing the appropriate distance measure for each feature, this definition proves to be very useful for use in segmentation later on.

7 Classification

With the final representation yet to be decided, it has been suggested by research of related work that two different machine learning tasks are involved in this thesis. First and foremost, labels will be set for segments of video depending on the atmosphere and mood they convey. Using labels as ground truth, forming a training data set, means that the task of learning the atmosphere of a video segment will be supervised. Input to such a learning algorithm will be video segments containing multiple shots. It is expected that the temporal order and alignment of these shots form a descriptive sequence that correlates with said labels. A premise to this machine learning problem is that shots can be formulated as a sequence, which leads to the second task. The constructed sequence should reflect the audio- and video characteristics; similar shots need to be represented as similar in the sequence and vice versa. It is the extracted features that describe the characteristics of a shot, without any notion of the truth. In other words, learning how to divide and partition shots based on their similarities and dissimilarities is an unsupervised problem.

While seemingly separate tasks, it is important that the decisions of machine learning methods can coexist. For example, imagine that we can learn how to form sequences consisting of a vector with only ones and zeros. Assume that the resulting sequences can be vectors of different length, depending on the number of shots in the segment. Such a representation is of little use if the input to the next machine learning algorithm is designed to only allow vectors of equal length. With the purpose of unraveling how to solve these tasks, some technicalities of machine learning algorithms have to be presented. Besides forming a general idea how machine learning algorithms and schemes work, the upcoming section will culminate in the specific choices and assumptions made for the final project setup.

7.1 Clustering

Collected data does not always provide a label or ground truth to assist in the learning of patterns. While there of course is the possibility of manually labeling the data, chances are that the amount of data is simply too large to label. Additionally, some applications does not require the data to be classified, but merely to analyze the data to find correlations and differences. Suppose that a data set can be easily and effectively divided into two partitions that can be analyzed separately, the task of learning relationships have been divided into two smaller subproblems. Clustering is an unsupervised learning method with the purpose of descriptively categorize data with respect to their similarities and differences [57]. To illustrate the idea of clustering, we present a small toy example:
A database have records of the coordinates (x, y) for every home of the citizens in Stockholm. The information about where the municipality borders are drawn is lost, so they decide to divide Stockholm into eight new areas based upon the resident data. Clustering can be used to find the best eight clusters of houses, with respect to a chosen distance measure. The concept is fundamental in Data Mining tasks, creating the opportunity to separately analyze data instances that share the same characteristics. While it is an efficient and extensively used method, it is most often important to choose the correct clustering technique depending on the data itself. Different techniques use different measures of fit and similarity. Since the work in this report includes clustering, some of the most well-known clustering algorithms will be visited.

7.1.1 K-means

One of the easiest and most used clustering technique is called K-means. The algorithm belongs to the *error minimizing algorithms* which basically minimizes the distance between a data instance and a cluster center, for all data instances. Cluster centers are either chosen randomly or according to some predefined process, assigning certain instances as centers. K-means clustering is an iterative algorithm that for each iteration assigns each data point to the nearest cluster center. Then an update of the centers follows, by calculating the mean of all data points assigned to the same cluster. Once again the instances are matched to the closest cluster, repeating until a fixed amount of iterations has been reached, or the designated number of clusters. Common distance measures for similarities are the Euclidean distance or the Manhattan distance. K-means

and similar error minimizing algorithms usually perform well if the data forms isolated and compact clusters [57]. It will be used later as part of the shot detection algorithm, which is preemptively structured to create two isolated clusters. Two big problems with K-means are the selection of the initial partitions as well as noisy data and outliers. For the shot detection algorithm, there is prior knowledge aiding in selecting proper cluster centers. However, noisy data and outliers can always increase the cluster errors substantially.

7.1.2 Agglomerative Hierarchical Clustering

While K-means partition instances by measuring the distances between the data and the clusters, another approach to clustering is Hierarchical Clustering methods. Common denominator is that the clusters are formed by recursively partition the instances into clusters based on their similarities [57]. Specific for Agglomerative Hierarchical Clustering is that, each object is initially assigned to belong to its own cluster. By recursively merging similar clusters, either up to a set similarity threshold, or until a specified number of clusters [35]. Agglomerative Hierarchical Clustering will, for the purpose of this machine learning task, be used for merging frames into shots.

7.1.3 Cluster Evaluation

It has been said that initialization and outliers affect cluster errors, without mentioning what is meant by an error. Since clustering is an unsupervised machine learning concept, the way to evaluate the learning is to create some form of criterion to evaluate with respect to. For K-means, which is an error minimizing algorithm, the measure of cluster performance is the Sum of Squared Errors (SSE). It is defined as

$$SSE = \sum_{k=1}^{K} \sum_{\forall x_i \in C_k} ||x_i - \mu_k||^2, \tag{30}$$

where C_k represent the set of instances belonging to cluster k. The instance x_i is compared to the vector mean of cluster k (centroid), namely μ_k. While this is only one possible criterion, the point is that clustering algorithms are constructed by formulating criteria measures suitable for the data itself. Whether a clustering is "good" or not, is purely decided by domain knowledge.

As said in the introduction of this section, the task in thesis involves labels of the concept we are trying to learn. Clustering has been used as a tool in order to form a valid representation, which can be used as a step in learning the atmosphere of a video. Since the concept of atmospheres is manually labeled, it is now time to head into the supervised learning method used in this project. Motivated mostly by related work described in section 5, the Support Vector Machine is a valid point of entry.

7.2 Support Vector Machines (SVM)

Support Vector Machines (SVM) are supervised learning models, commonly used in machine learning for classification and regression[58]. The general idea of SVM is to find a hyperplane that separates the input data. In addition SVM

tries to maximize the margin between different classes, gaining robustness for new, and unseen data. This chapter will present the support vector machines in more detail, along with possible extensions in order to deal with non-linearly separable data. Both structure and content of this section is highly inspired by the detailed description of SVM:s found in [58].

7.2.1 Linear Support Vector Machine

The most basic SVM is the linear SVM, which relies on linearly separable input data. Recall the mathematical description in 4.3. For this example, $\mathbf{x_i}$ is the same, a feature vector $\mathbf{x_i} \in \mathbb{R}^M$, while the label is assumed to be either *positive* or *negative*; $y_i \in \{-1, 1\}$. The goal is to learn a separating hyperplane

$$f(\mathbf{x}) = sgn(\mathbf{w} \cdot \mathbf{x} - b), \tag{31}$$

where *sgn* is the sign function, $\mathbf{w}$ a weight vector and b the offset parameter. The role of the latter term is elaborately described in [59] and will be overlooked in this thesis, to be handled by chosen algorithms. The learning is accomplished by forcing the weight vector $\mathbf{w}$, as well as b, to conform to three hyperplanes, namely

$$H_0 : \quad y = \mathbf{w} \cdot \mathbf{x} - b = 0 \tag{32}$$
$$H_1 : \quad y = \mathbf{w} \cdot \mathbf{x} - b = 1 \tag{33}$$
$$H_2 : \quad y = \mathbf{w} \cdot \mathbf{x} - b = -1 \tag{34}$$

These hyperplanes should meet six conditions:

- All hyperplanes H are to be parallel,
- H_1 and H_2 should be equally distanced from H_0,
- The *positve* data point closest to H_0, $\mathbf{x_{sp}}$, coincides with H_1,
- The *negative* data point closest to H_0, $\mathbf{x_{sn}}$ coincides with H_2,
- All other data points should then satisfy $y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1$,
- The distance between H_1 and H_2 are to be maximized.

Data points on respective hyperplane, $\mathbf{x_{sp}}$ and $\mathbf{x_{sn}}$, are called *support vectors*, which are the only vectors contributing when calculating $\mathbf{w}$. The last condition can be expressed mathematically as

$$\frac{|(\mathbf{w} \cdot \mathbf{x_{sp}} - b) - (\mathbf{w} \cdot \mathbf{x_{sn}} - b)|}{\|\mathbf{w}\|} = \frac{|1 - (-1)|}{\|\mathbf{w}\|} = \frac{2}{\|\mathbf{w}\|}. \tag{35}$$

Thus minimizing $\|\mathbf{w}\| = \mathbf{w^T w}$ equals maximizing the margin between the hyperplanes, yielding the optimal separation of data points. The idea is further illustrated in Figure 8. As the figure show, the found hyperplane of the SVM algorithm, is the line separating the support vectors of each class, with the maximum distance between classes. Learning can be formulated as

$$\min_{\mathbf{w},b} \mathbf{w^T w}, \qquad s.t. \quad y_i(\mathbf{w} \cdot \mathbf{x_i} - b) \geq 1, \tag{36}$$

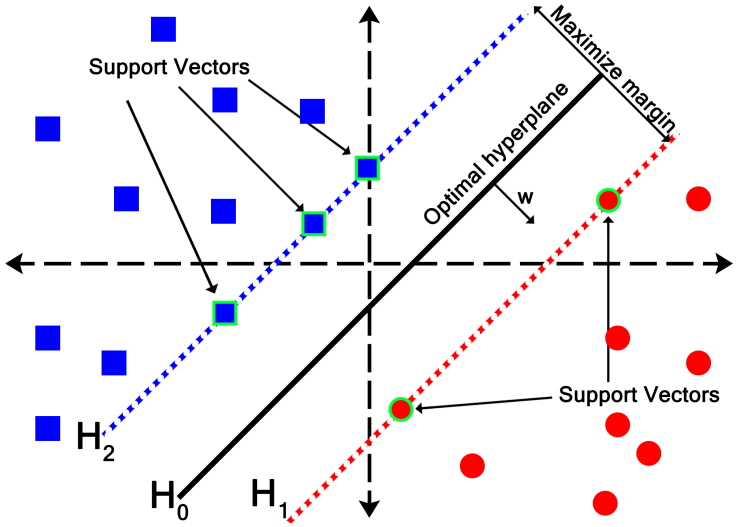


Figure 8: *Support Vector Machine learns a hyperplane that separates two classes, with the largest possible margin between classes. For linear SVM, the said hyperplane is a line.*

which belong to a special type of mathematical optimization problems. The goal is to minimize a quadratic function with respect to several variables, which are subject to linear constraints. This is also known as a Quadratic Programming (QP) problem [60].

7.2.2 Dual Problem

The optimization problem in the previous section is a convex quadratic programming problem in a convex set. Introducing the Lagrange multipliers $\boldsymbol{\alpha} = \alpha_1, \ldots, \alpha_i, \ldots, \alpha_N$, i.e. one multiplier for each inequality in Eq. (36). This expresses the problem in terms of the Lagrangian function [60]

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}) = \mathbf{w^T w} - \sum_{i=1}^{N} \alpha_i y_i (\mathbf{w} \cdot \mathbf{x_i} - b) + \sum_{i=1}^{N} \alpha_i, \tag{37}$$

which can be solved by doing the following:

$$\max_{\boldsymbol{\alpha}} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\alpha}),$$

$$s.t \quad \frac{\partial \mathcal{L}}{\partial \mathbf{w}} = 0 \qquad \frac{\partial \mathcal{L}}{\partial b} = 0 \qquad \boldsymbol{\alpha} \geq \mathbf{0}. \tag{38}$$

Substituting Eq. (38) in Eq. (37) yields

$$\begin{aligned} &\max_{\boldsymbol{\alpha}} \mathcal{L}_D = \max_{\boldsymbol{\alpha}} \left[\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x_i} \cdot \mathbf{x_j} \right], \\ &s.t. \sum_i \alpha_i y_i = 0 \end{aligned} \tag{39}$$

as the new optimization problem. This is known as the *Wolfe's reduced gradient method* [61]. The initial variables $\mathbf{w}$ and b is currently not needed for further solving. However, the weight matrix can be calculated from the Lagrange multipliers as

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i y_i \mathbf{x_i}, \tag{40}$$

which will be used to classify unseen data. Before solving Eq. (39) any further, please notice that the only operation concerning the feature vectors $\mathbf{x_i}$ is a dot product, as well as a multiplication with its corresponding label. By abusing this property, the SVM can be modified handle non-linearly separable data, also known as the *kernel trick*.

7.2.3 Kernels

Assume that the feature vectors $\mathbf{x_i}$ requires a non-linear surface for separation. As an example of when this might occur, see the data presented at the left part of Figure 9. The two classes are represented by their Cartesian coordinates. Separating them would require a circular hyperplane, i.e. a non-linear hyperplane. Expressing the data in Polar coordinates (right side of the figure), transform the data to a feature space where linear separation is possible.

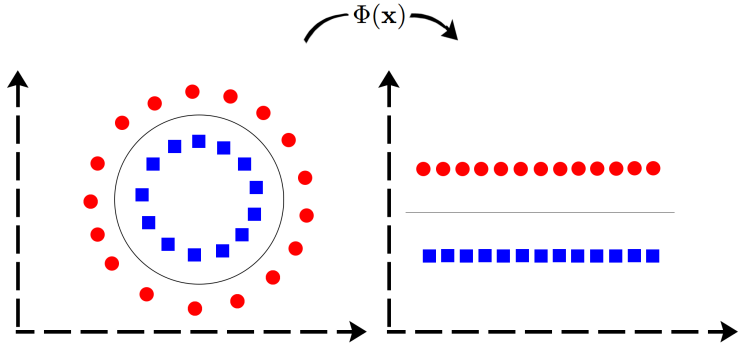


Figure 9: *The left figure show an example when the data (in Cartesian coordinates) can not be linearly separated. A transformation into Polar coordinates makes linear separation possible.*

By transforming the feature from the input space into a, possibly higher-dimensional, feature space, we convert the data into a form that is separable. In fact, N data points will be separable in spaces of $N-1$ dimensions or less,

with very few exceptions [58]. Consider a *kernel map* $\Phi(\cdot)$, transforming the data point into an arbitrary dimension. The optimization problem then becomes

$$\max_{\boldsymbol{\alpha}} \mathcal{L}_D = \max_{\boldsymbol{\alpha}} \left[\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \Phi(\mathbf{x_i}) \cdot \Phi(\mathbf{x_j}) \right], \tag{41}$$

assuming the dot product is justified for $\Phi(\cdot)$. Suppose that the dot product in this high dimensional space is equal to the result of a kernel function, i.e. $K(\mathbf{x_i}, \mathbf{x_j}) = \Phi(\mathbf{x_i}) \cdot \Phi(\mathbf{x_j})$. Ultimately this means that, as long as the kernel function $K(\mathbf{x_i}, \mathbf{x_j})$ satisfies certain conditions, the dot product is calculated without knowing either the representation of the higher-dimensional space nor the explicit transformation. Said conditions is given by Mercer's condition [58]; There exists a mapping and an expansion

$$K(\mathbf{x}, \mathbf{z}) = \sum_i \Phi(\mathbf{x})_i \Phi(\mathbf{z})_i \tag{42}$$

if and only if , for any $g(\mathbf{x})$ such that,

$$\begin{aligned} &\int g(\mathbf{x})^2 d\mathbf{x} \qquad \text{is finite, then,} \\ &\int K(\mathbf{x}, \mathbf{z}) g(\mathbf{x}) g(\mathbf{z}) d\mathbf{x} d\mathbf{z} \geq 0. \end{aligned} \tag{43}$$

Even though this sounds promising, the choice of kernel dictates how well the model generalize, i.e. how the learned model will perform when classifying unseen data. A higher dimension increases the complexity of the separating surface, which in many cases result in learning a too specific behavior, i.e. over-fitting. Presume a data set looking as in Figure 10, where there is one point which can be considered as an outlier that are unrepresentative for the rest of the data. In the figure, an alternative non-linear hyperplane has been drawn to include the added data point. Learning to include this data point by increasing the dimensionality, can create an over-fitted model, performing poorly once given new examples. Failing to learn any separating hyperplane is likewise called an under-fitted model.

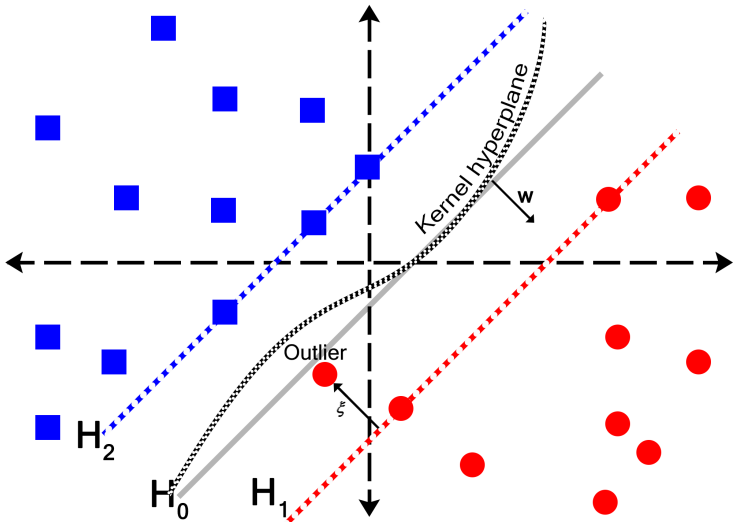


Figure 10: *Illustration of a non-linear SVM as well as the use of slack variables. The black dotted line represent a non-linear decision boundary. Linear separation is also possible by introducing a slack variable.*

7.2.4 Slack variable

There are other methods to handle outliers and noisy data than non-linear transformations. Consider again the data set in Figure 10. The points are in fact still separable using a linear SVM, although with a smaller margin. Since the margin should be maximized for more robustness, it is not always beneficial to use the hyperplane providing perfect separation. By introducing a slack variable ξ_i,

$$\begin{aligned} H_1 &: y = \mathbf{w} \cdot \mathbf{x} - b = 1 - \xi_i \\ H_2 &: y = \mathbf{w} \cdot \mathbf{x} - b = -1 + \xi_i \\ \xi_i &\leq 0, \qquad \forall i, \end{aligned} \tag{44}$$

we allow data instances to exist between H_1 and H_2, i.e. imperfect separation. Allowing misclassification is penalized by a factor C, altering the optimization problem in Eq. (36) to become

$$\begin{aligned} &\min_{\mathbf{w},b,\xi_i} \mathbf{w^T w} + C \sum_{i=1}^{N} \xi_i \\ &s.t. \quad y_i(\mathbf{w} \cdot \mathbf{x_i} - b) + \xi_i - 1 \geq 0, \qquad 1 \leq i \leq N \end{aligned} \tag{45}$$

,

where ξ_i has been as added as a optimization variable. It can be shown that the introduction of ξ_i only affect the dual problem, Eq. (39), by adding an upper constraint for $\boldsymbol{\alpha}$ [61]. The resulting optimization problem including the slack variable can be expressed as

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \mathcal{L}_D &= \max_{\boldsymbol{\alpha}} \left[\sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j K(\mathbf{x_i}, \mathbf{x_j}) \right], \\ s.t. \qquad & \sum_i \alpha_i y_i = 0 \\ & 0 \leq \alpha_i \leq C \end{aligned} \tag{46}$$

Letting the penalizing term $C = \infty$, equals the original optimization problem, without the slack variable.

7.2.5 Sequential Minimal Optimization (SMO)

As a result of adding non-linear properties for the SVM, the final expression to be optimized is formulated in Eq. (46). There are at least three different methods available for the optimization of the Lagrange multipliers α_i; Chunking[62], Osuna's[63] and SMO[64]. The latter, Sequential Minimal Optimization (SMO), is the fastest algorithm of the three. The SMO method searches heuristically for two multipliers to optimize, iteratively finding the optimal values for all α_i. This method is based on the fact the solving the optimization problem for two Lagrange multipliers can be done analytically, instead of the implementation of any numerical routines. In order to avoid tedious calculations in the field of optimization, details of how SMO work can be read about in [64], including the presented facts.

So far in the discussion and presentation of Support Vector Machines, it has been referred to as a binary classification problem. It is binary in the sense that only two classes are present as possible output, a positive or negative instance. True for the learning task of atmosphere classification is that we have six different classes, i.e. a multi-class classification problem. The implementation of SMO used in this thesis has been made by [65], states that multi-class problems are solved in a 1-vs-1 fashion. This means that the Lagrange multipliers are optimized by classify two classes at the time, for all combinations. To be precise, recall the atmosphere labels presented in section 4.1. Possible combinations are for example *Eventful-Tense*, *Eventful-Joyful*, *Joyful-Tense*, and so forth. Another possible way is to optimize in a 1-vs-rest manner, resulting in a *Eventful-Not Eventful* classification where all other classes are considered as negative examples. Discussing the differences is valuable since the two approaches are conceptually different. Either we choose to comfortably build one classifier, predicting all possible classes, or one binary learning machine for each class.

7.2.6 String Kernel

As the introduction to this chapter mentioned, the scenes will be analyzed as a sequence of shots. A scene can contain different number of shots, thus the sequence can have varying length. Examples of other areas including dynamically sized inputs are text classification and DNA-sequencing. Most learning algorithms cannot handle input of different length [41], partly motivating the extension of kernels. Besides dealing with non-linear data as mentioned earlier, kernels can be utilized to transform the input into a feature space where the input appears to have fix length. Successful use of the so called String Kernel,

along with SVM, has been shown in [66, 41, 67]. In addition, a scene represented by a sequence of letters are the essential part of the scene segmentation presented in [33]. So how does one build a string kernel?

Let $s = s_1, \ldots, s_{|s|}$ denote a string, with the length $|s|$. The string is a set of characters or letters belonging to a finite alphabet $\mathcal{A}$. All finite strings with length n is denoted $\mathcal{A}^n$, creating a feature space $\mathcal{F}^n = \mathbb{R}^{\mathcal{A}^n}$. Let u be a substring of length n, namely $u \in \mathcal{A}^n$. A string is transformed to feature space $\mathcal{F}^n$ by the feature map $\Phi_u(s)$. The feature mapping is calculated as

$$\Phi_u(s) = occ_u(s), \tag{47}$$

where $occ_u(s)$ represent the number of occurrences of subsequence u in s. The number of occurrences is weighted according to their contiguity by a decaying factor $\lambda \leq 1$. In other words, the substring u might occur, only with multiple other characters in between, creating a gap. The length of the the matching sequences, including the gap, is given by $l(\cdot)$. The target string u is only a subsequence of s if there exist indices $\mathbf{i} = (i_1, \ldots, i_n)$ such that $u_j = s_{i_j}$ for all $j = 1, \ldots, n$. We call this $u = s[\mathbf{i}]$ for simplicity, yielding the mathematical expression as

$$occ_u(s) = \sum_{u=s[\mathbf{i}]} \lambda^{l(\mathbf{i})}, \tag{48}$$

where the length of the subsequence $l(\mathbf{i}) = i_{|u|} - i_1 + 1$. Forming the kernel $K_n(s,t)$ of two strings is thus a measurement of the amount of shared subsequences, with length n:

$$\begin{aligned} K_n(s,t) &= \sum_{u \in \mathcal{A}^n} \Phi_u(s) \cdot \Phi_u(t) = \Big[\text{Equation (47) and (48)}\Big] = \\ &= \sum_{u \in \mathcal{A}^n} \sum_{u=s[\mathbf{i}]} \lambda^{l(\mathbf{i})} \sum_{u=s[\mathbf{k}]} \lambda^{l(\mathbf{k})} = \\ &= \sum_{u \in \mathcal{A}^n} \sum_{u=s[\mathbf{i}]} \sum_{u=s[\mathbf{k}]} \lambda^{l(\mathbf{i})+l(\mathbf{k})}. \end{aligned} \tag{49}$$

To illustrate how this works in practice, consider the string "up" and "upp". Furthermore, let $\lambda = 0.5$ and the sequence length $n = 2$. The resulting matches are:

String	u	p	u	p	p	Length
Match	×	×	×	×		$l = 4$
	×	×	×		×	$l = 5$

We have two matches, one with a combined length of 4, the other with length 5. The string kernel results will be $\lambda^4 + \lambda^5 = 0.5^4 + 0.5^5 = 0.09375$. The decay parameter decides how to penalize a gap between the matching characters, i.e. a smaller value of λ decreases the similarity if there is a gap. In order to express the similarity in range [0 1], a normalization is needed, where 1 is a perfect match. A normalized kernel $\hat{K}(s,t)$ is achieved by computing

$$\hat{K}(s,t) = \frac{K(s,t)}{\sqrt{K(s,s)K(t,t)}}, \tag{50}$$

according to calculations found in [41].

The details and mathematical description might come off as complicated but the important fact to assimilate is that similarity between strings of different length can be measured. The measurement account for the temporal alignment and order of the sequences, as well as the length between matching sequences. There is however a potential problem with using string kernels for the work in this thesis. There is no notion of what the letter between matchers are, just that there is a gap between matching sequences. Consider the two words "up" and "usp". The string kernel will return the same value of similarity as between the words "up" and "ump". For the purpose of sequences of video shots, it is not unreasonable to assume that it will matter whether the letter in between is "m" or "s".

The presented technicalities in this chapter provides us with a method for learning classes or concepts, which are represented as a string. Specifically for this project, this imply that the video content needs to be transformed into a string descriptor. Forming these strings is, in our context, an unsupervised learning task which could be aided by the use of clustering. The strings should represent the characteristics of the video segments involved, including the temporal order and alignment. Despite frequently mentioning the importance of video segmentation, it has not yet been presented how to segment a video.

8 Video Segmentation

The last piece of information, to be able to summarize what we have learned, is the video segmentation. Time and time again it has been stated that we both need and expect the video frames to be clumped into shots, with the purpose of maintaining the temporal elements and characteristics of video productions. To motivate again why this is useful, please consider the following situation. Two shots taken from entirely different movies, shows a conversation between two people. Visually, the content may vary substantially; one could be recorded outside in the sun and the other one in dark quarters of a spaceship. By analyzing and comparing the color information from a single frame taken from each shot, the shots will be considered dissimilar. For this thesis it is requested that both situations should be expressed as similar. It is thus important to compare larger segments of frames, containing a wide range of characteristics, including the propagation through time. These shots should furthermore be part of a larger video segment describing the atmosphere or mood. The core of this assignment is to investigate whether a sequence of shots can describe the certain atmosphere we interpret when watching a video. In order to learn anything from the sequences, they have to be of adequate length, correlating the sequence to a set label. This chapter will present how to deal with the challenges of video segmentation.

8.1 Shot Segmentation

Quickly browsing the Internet, a common shot length is about 5-10 seconds long. It is instantly realized that segmenting a video manually is far too time-consuming. Thus numerous attempts of automatic shot detection has been made so far, many presented and evaluated in [39]. By inspecting the differ-

ences between successive frames, e.g. color information or edge information, cuts can generally be detected easily. The hard part is to detect *all* cuts and furthermore, avoid to falsely detect shot boundaries. In addition, video productions occasionally use fade or dissolve effects to alert the viewer of temporal story changes. Simply put, we need a reliable shot detector that automatically detects how to segment the video.

8.1.1 Shot boundary detection without thresholds

The shot boundary detection (or cut detection) algorithm chosen in this project is using k-means clustering, based on the approach described in [68]. It is beneficial that the cut detection algorithm is entirely unsupervised both in terms of parameter tuning and thresholds, since every analyzed video is different. Additionally, for the same reason, it is preferable to choose an algorithm which excludes a training stage. More details regarding clustering techniques in general has been presented in section 7.1. The proposed method works as follows, reproduced from the original report:

Using the edge histogram mentioned previously along with the chosen distance measure for the same, the inter-frame differences are calculated for all frames. The distance array D is formed, containing distances, d_i, representing the frame difference between frame i and frame $i+1$. From D, the maximum difference value, d_{max}, is extracted which will be used to compute a normalized feature vector for clustering.
A non-overlapping sliding window of size $2m+1$ is applied with the purpose of creating a series of frame differences f_k as:

$$f_k = [d_{k-m}, ..., d_k, ..., d_{k+m}]. \tag{51}$$

Each frame series f_k is assumed to contain one cut candidate. The representation of this candidate $v(f_k)$ is formed by letting

$$v(f_k) = \left[\frac{f_{max}}{d_{max}}, \frac{f_{sec}}{f_{max}}\right], \tag{52}$$

where f_{max} and f_{sec} is the largest and second largest value within each series. This way, the data is normalized with respect to d_{max}. Have a look at Figure 11, aiming at illustrating how cut candidates are represented. The first series of frame differences show multiple large differences, however low compared to the maximum difference d_{max}. In addition, the peaks are not isolated, suggesting that the differences are due to rapid movement or similar, not a cut. The second sliding window includes an isolated difference measure, close to the largest possible difference for the entire video, suggesting a cut. Since a cut is expected to be represented as such isolated large values in the difference array D, an ideal cut would yield $v(f_{ideal}) = [1, 0]$. The corresponding vector for an ideal non-cut would be $v(f_{non}) = [0, 1]$.

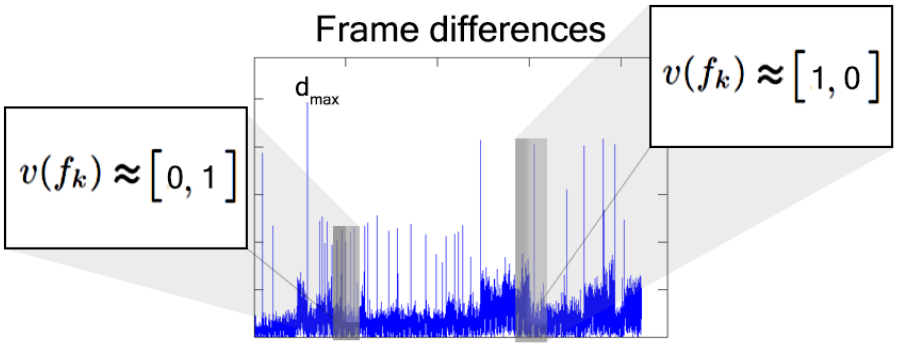


Figure 11: *An illustration of how cut candidates are represented in the shot detection algorithm. The x-axis is the frame ID, while the y-axis corresponds to the magnitude of frame differences. For the first (left) sliding window, the inter-frame differences have no isolated peaks. The second (right) sliding window includes a peak which is unquestionably the largest difference in the series, i.e. a suspected cut.*

This "ground-truth" is used as the initial clusters for the k-means algorithm. The clustering will assign each candidate to either of these clusters, segmenting the video.

The performance of detecting the shot boundaries is dependent on the size of the sliding window, thus the value of m. To improve performance, the above procedure is done for different values of m, $(1 < m < m_{max})$, followed by a cluster evaluation for each window size. The suggested evaluation technique utilizes the silhouette coefficient $SC(v)$ of a feature vector v, as a quality measurement. More specifically, for each candidate classified as a cut, v_c, the SC is calculated as

$$SC(v_c) = \frac{b(v_c) - a(v_c)}{max(b(v_c), a(v_c))}, \tag{53}$$

where $a(v_c)$ is the average Euclidean distance between v_c and the other feature vectors belonging to the same cluster[68]. Similarly, $b(v_c)$ is the average distance to the members of the other cluster. The cluster quality CQ is then given as

$$CQ = \frac{1}{N} \sum_{v_c} SC(v_c), \tag{54}$$

where N is the number of candidates classified as cuts. The algorithm will thus find the most suitable value of m ranging from 1 to m_{max}. In a way m represents a limit of how often a cut can occur, since only one cut candidate is chosen for each window. Tests of how to set m_{max} have been performed in [68]. In general, the algorithm is able to find a suitable m regardless of the maximum window size m_{max}. As long as m_{max} is reasonably high $(m > 12)$, the evaluation of the clusters each iteration, will make sure that the m with the best cluster quality is chosen.

8.1.2 Shot Feature Representation

Revisiting the mathematical description stated earlier in section 4.3, the feature representation for each frame needs to be transformed to characterize an entire shot. The feature vector in Eq. (1) can be rewritten as

$$\mathbf{x} = [\mathbf{v_1}, \mathbf{v_2}, \dots \mathbf{v_p}], \qquad \mathbf{x} \in \mathbb{R}^M, \tag{55}$$

where $\mathbf{v_p}$ is feature p containing q_p columns. Let a shot X_n be a set of frames,

$$X_n = [\mathbf{x_1}, \dots, \mathbf{x_n}]^T, \qquad X_n \in \mathbb{R}^{n \times M}. \tag{56}$$

In order to receive a more manageable shot representation, a transformation $T : X_n \rightarrow S$ is needed, which reduces the dimensionality of a shot without loosing vital information. This reduction is made by agglomerative hierarchical clustering of frames within a shot, inspired by previous work in the similar field of video retrieval in [35]. The idea is based upon that frames can be merged if similar enough and works as follows:
Each cluster in the algorithm is considered to be a video segment. Initially all frames is set to be its own cluster, i.e. a video segment containing one frame only. The L_1-distance between features is then computed between successive clusters within the shot. If the minimum distance of all clusters is below a threshold γ, the two clusters merge to a single cluster. When merging two clusters, the feature vector needs to be updated in order to compute new distance for next iteration. Similar to [35], the average histogram will be used to represent a video segment, averaging each histogram bin for all frames in the segment. The average value for a histogram bin is given as [45]

$$AvgHist(b_i) = \frac{1}{m} \sum_{j=s_f}^{e_f} H_j(b_i), \tag{57}$$

where b_i is the bin, s_f the starting frame, e_f the ending frame, H_j the *j:th* histogram and m the total number of frames in the cluster. This way, frames are clustered into video segments until the minimum distance exceeds the threshold. Once the clustering stops, the remaining clusters represent a summarized description of the shot.

The above clustering method is used separately for each feature $\mathbf{v_p}$ in the feature vector. For each feature the ratio

$$v_s(p) = \frac{\text{number of remaining clusters for feature p}}{\text{initial number of clusters}}, \tag{58}$$

is formed, yielding a variation parameter for each feature in the range [0 1]. The final shot representation has been greatly reduced to

$$S = [v_s(1), v_s(2), \dots, v_s(p)], \qquad S \in \mathbb{R}^{1 \times p}, \tag{59}$$

which in a way is a measure of how each feature vary during the shot. If $v_s(p)$ is low, the feature p change less between the frames in the shot. A feature column close to 0 means that the entire shot could be summarized by only one frame, according to that feature.

8.2 Scene Segmentation

Once shot boundaries have been found, the next task would be to segment the shots into scenes or similar larger segments. The advances and successes in automatic scene detection lag behind in terms of accuracy and reliability [33]. Using them would introduce further uncertainties to this experiment. Besides, the definition of a scene in this thesis varies from the general usage, since it is partitioned according to the atmosphere. This motivates using the scene delimiters directly from the training data, which has been labeled manually. Of course, further use of the classifier created from this work cannot assume the input to be manually labeled. The idea is that once the classifier has been trained successfully, it can perform predictions of segments either from a constant partitioning or any attempt of automatic scene detection. It is merely for the purpose of training the classifier, that the scenes are divided by manual labels.

8.2.1 Scene Feature Representation

Recall the shot representation in section 8.1.2. The shot feature vector now contains a measure of the importance of every feature initially extracted from the frames. Even though [33] is not used for scene detection, it suggest an approach for how to symbolize a scene. In the report, the similarities between all shots are computed forming a similarity matrix. After clustering this matrix, each shot is given a letter corresponding to the assigned cluster. A scene will ultimately be a string containing a sequence of letters. By doing this the problem has been reduced to again match the initial structure in section 4.3, suitable for machine learning. Instead of a description of frames along with their features, the problem has been transformed to

$$X = [s_1, \ldots, s_i, \ldots, s_N]^T, \tag{60}$$

$$Y = [y_1, \ldots, y_i, \ldots, y_N]^T, \tag{61}$$

where $\mathbf{s_i}$ instead contains a string of letters of different length.

9 Methodology

In an attempt to summarize the huge variety of components connected to this project, this section will try to interweave the different parts into a model. Additional knowledge for all the relevant steps in the procedure of indexing videos, allows for a detailed declaration of tasks to complete. Refer to Figure 12, which describe the model outline. With the purpose of declaring the choices made when trying to implement this model for video indexing and classification. We will walk through the descriptive picture step by step:

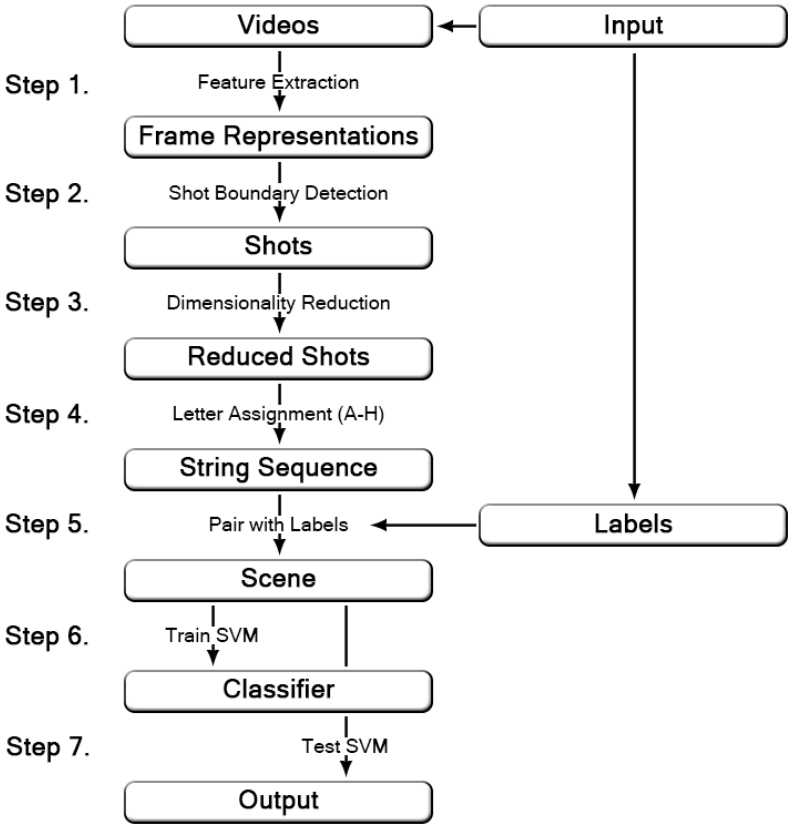


Figure 12: *Overview of the proposed model outline.*

- **Step 1:** All of the features in chapter 6 have been extracted from each frame in the video. Calculations involving several frames, such as motion, will still yield one output per frame.
- **Step 2:** Using only the edge histogram, the shot boundaries are detected by the cut detector algorithm presented in section 8.1.1.
- **Step 3:** For each shot, Agglomerative Hierarchical Clustering is performed, reducing the representation from $\mathbb{R}^{n \times M}$ to $\mathbb{R}^{1 \times p}$ (as in section 8.1.2). n is the number of frames in each shot, M the total length of the feature vector for each frame and p the number of features.
- **Step 4:** Once all videos are processed, the set containing all shots are clustered into 8 clusters using K-means. Each shot is thus given a letter "A-H", representing a class of shots. The number 8 is motivated by [5] suggesting all shots can be divided into 8 shots, as well as [33] where 8 is an experimentally good number for scene segmentation.
- **Step 5:** The strings are paired with their corresponding label, forming a data set.
- **Step 6:** A Support Vector Machine, using sequential minimal optimization along with the string kernel, is trained.

- **Step 7:** For testing, Step 1-5 is repeated for a new set of videos. The label will now be used for evaluation (Step 5), and the SVM is naturally tested instead of trained (Step 6).

9.1 Setup

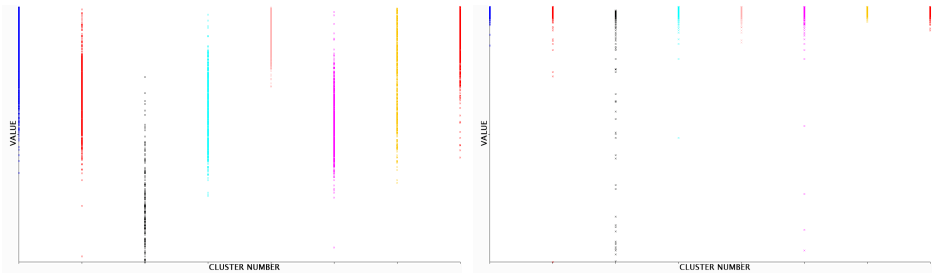
The material used for analysis is the underrated but much appreciated TV-series Firefly (2002-2003). A total number of 12 episodes were used for training, while 2 are used for testing. The videos have been analyzed and processed in JAVA, using the framework simultaneously developed in collaboration with Erik Bodin [14]. Any other used and altered source code has been provided by [69, 70, 56, 71]. For evaluative purposes, both MATLAB and WEKA [65] have been used.

10 Results

Due to the fact that the assignment has evolved into two machine learning tasks, there is need for evaluation of both how the string sequences behave, and the results of building a classifier upon these sequences. Thus, this section will first present the results of clustering shots, assigning letters representatively. Understanding the characteristic of the created sequences is vital for any interpretation of the classifier results.

10.1 Shot Clustering

The goodness of fit for the resulting 8 clusters, by performing k-means clustering, can be measured by for example sum of squared errors or silhouette coefficients. However, such a number tells us nothing about if a shot has been given the correct letter or character. Better is to examine to what extent features are separated in the different clusters. Consider Figure 13a and 13b, where the first show an example of good separation characteristic. At least three different ranges of feature values are common, divided amongst the 8 clusters. In comparison with the second figure, where most clusters share the same range of feature values, there is better separation for the Hue histogram feature than the LPC feature.



(a) *Example of the cluster assignment for the feature based on the Hue histogram.*

(b) *Example of the cluster assignment for the LPC (Linear Predictive Coding) feature.*

Figure 13: *The cluster assignments for two different features. The horizontal axis corresponds to the assigned cluster number 1-8 (A-H), while the vertical axis describe the feature values. In a) it can be seen that the range of the feature values vary for different clusters. For the feature shown in b) almost all cluster share the same range of values.*

Performing these kind of visualization forms an idea of which features are more important, however ruling any feature as irrelevant is premature, since clustering in eight dimensions is hard to imagine. All in all, the performed clustering manages to separate the shots. To show the distribution of letters in the formed sequences, a histogram of the letters is computed for each atmosphere. The histograms, found in Figure 14, show interesting similarities for some atmospheres.

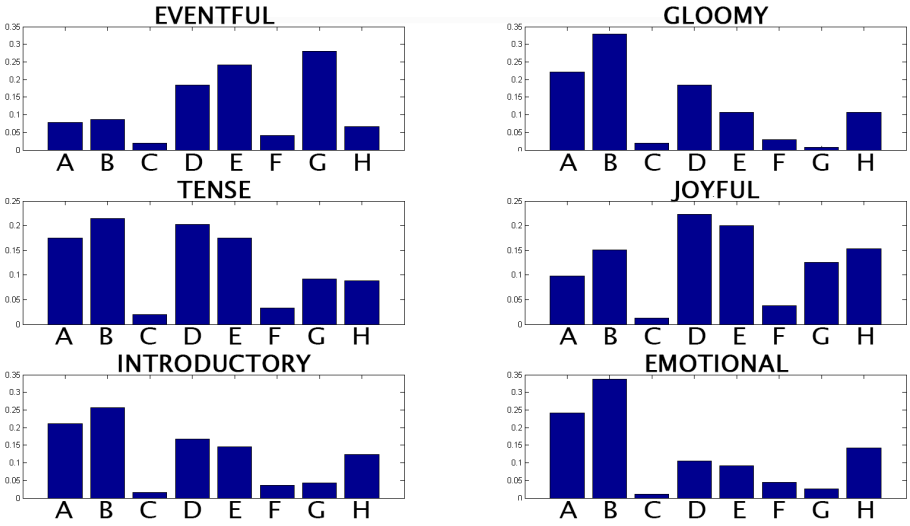


Figure 14: *The normalized distribution of letters/characters A-H, for each labeled atmosphere.*

The sequences labeled as *eventful* clearly differ from the rest of the distributions, with the frequent use of the letter G. Furthermore, *gloomy* and *emotional*

have very similar distributions of letters. Slightly less similar are *tense* and *joyful*, although still resembling one another. The *introductory* distribution share traits with most labels, with the exception of *eventful*. Even though this raises questions about how well such similarities will be separated by the classifier, it is important to note that the labels are expected to depend on the ordering of the sequence as well.

In order to examine the order of the letters in the sequences, the 10 most similar sequences for each label have been visualized in Figure 15 and 16. More specifically, the sequences are the five pairs of sequences that yield the highest kernel response.

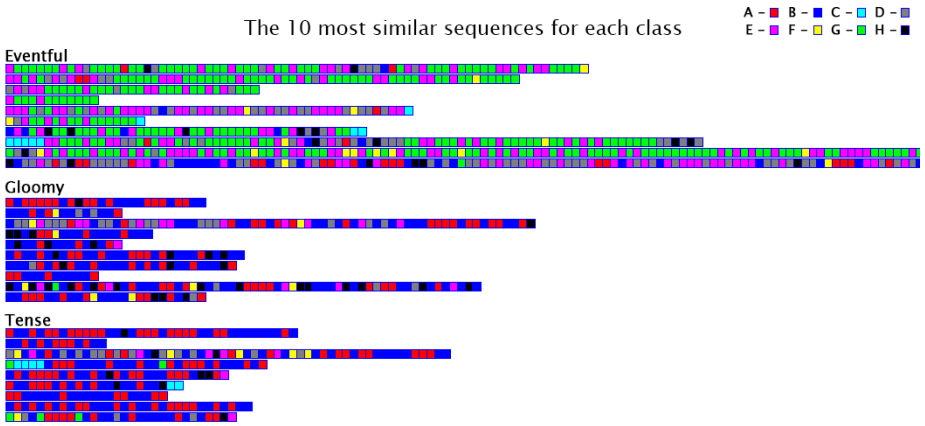


Figure 15: *The 10 most similar sequences for the labels eventful, gloomy and tense. The letters are represented as a colored block according to: A - red, B - blue, C - cyan, D - grey, E - magenta, F - yellow, G - green, H - black.*

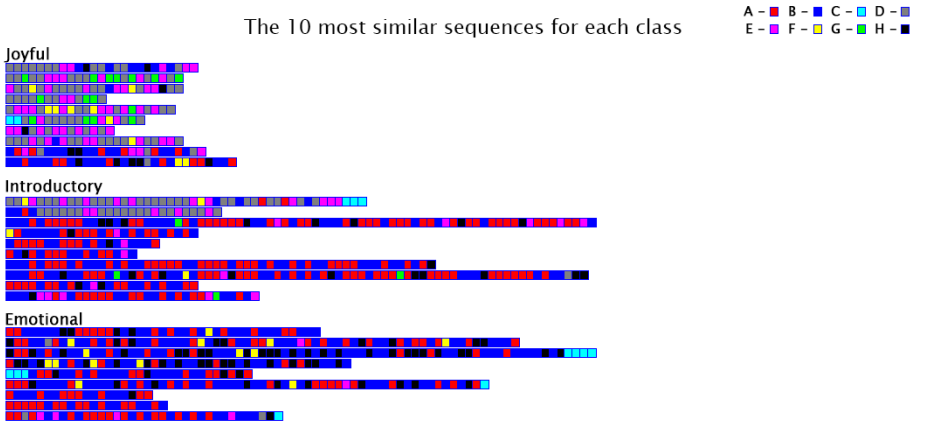


Figure 16: *The 10 most similar sequences for the labels joyful, introductory, gloomy. The letters are represented as a colored block according to: A - red, B - blue, C - cyan, D - grey, E - magenta, F - yellow, G - green, H - black.*

The figures show that *eventful* is one again the most unique. Besides the

frequency of the letter G, it also contains sequences involving the letter G and E (green and magenta). Most other examples containing green blocks, are part of sequences with other letters, e.g. D (grey) for the *joyful* sequences. Other characteristics worth mentioning are that *gloomy* and *emotional* commonly involve sequences with large portions of black blocks, and that subsequences containing blue and red is common for most labels. There is much more to be said about these sequences, which will be resumed in the discussion chapter, however no conclusions should be drawn without first examining the classifier results.

10.2 Classification

Before evaluating the performance of the classifier, many characteristics can be shown by observing the result of using a kernel. The Gram matrix G is defined as

$$G = \begin{pmatrix} \hat{K}(s_1, s_1) & \hat{K}(s_1, s_2) & \cdots & \hat{K}(s_1, s_N) \\ \hat{K}(s_2, s_1) & \hat{K}(s_2, s_2) & \cdots & \hat{K}(s_2, s_N) \\ \vdots & \vdots & \ddots & \vdots \\ \hat{K}(s_N, s_1) & \hat{K}(s_N, s_2) & \cdots & \hat{K}(s_N, s_N) \end{pmatrix}, \tag{62}$$

where $\hat{K}(s_i, s_j)$ is the normalized inner product between two strings, calculated as in Eq. (50). The resulting gram matrix when using the string kernel is presented in Figure 17. The color indicate how similar two strings are, ranging from blue (dissimilar) to red (similar). The input has been sorted with respect to the labeled class, allowing us to see the similarity between strings given the same label.

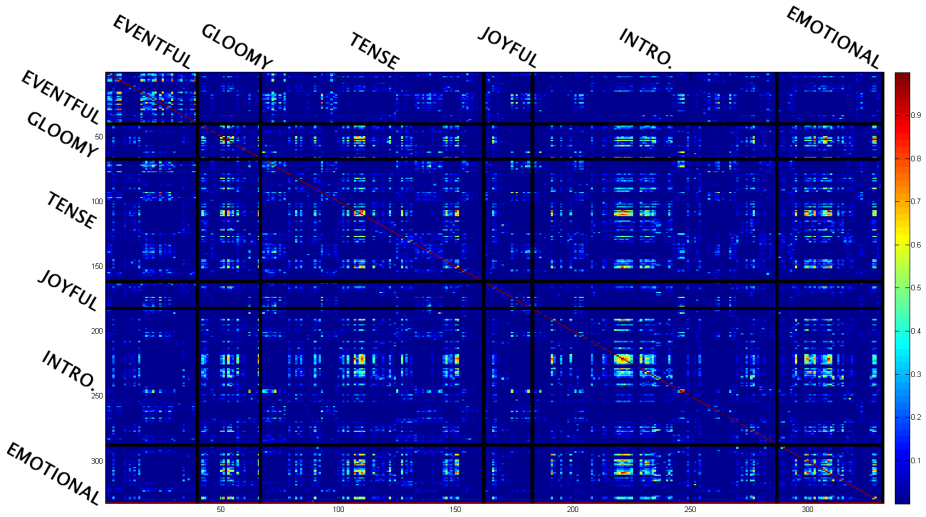


Figure 17: *Visualization of the Gram matrix when using a string kernel, i.e. the kernel response for all pairs of string sequences. The data is sorted by class label. Blue corresponds to dissimilar sequences, while red indicate similarity.*

Desired would be that the diagonal squares are similar, i.e. red, while the rest are blue, or dissimilar. This would indicate that only strings labeled as

equal are similar, which is the behavior we are trying to learn. As seen in the figure, strings labeled *eventful* are the best example of this, with high similarity for the same label and low similarity with other labels. Additionally it can be seen that most other classes, unfortunately, are similar to many different classes. It is therefore indicated that learning a multi-class classifier might prove to be difficult with the current setup.

To motivate that the string kernel is a good choice of method, we do the same computations for another kernel, namely the chi-square kernel. Input for a such a kernel is a histogram, with the distance measure as presented in section 6.4.3. The histograms in question are computed by once again counting the occurrence of letters in each string, as in the previous section.

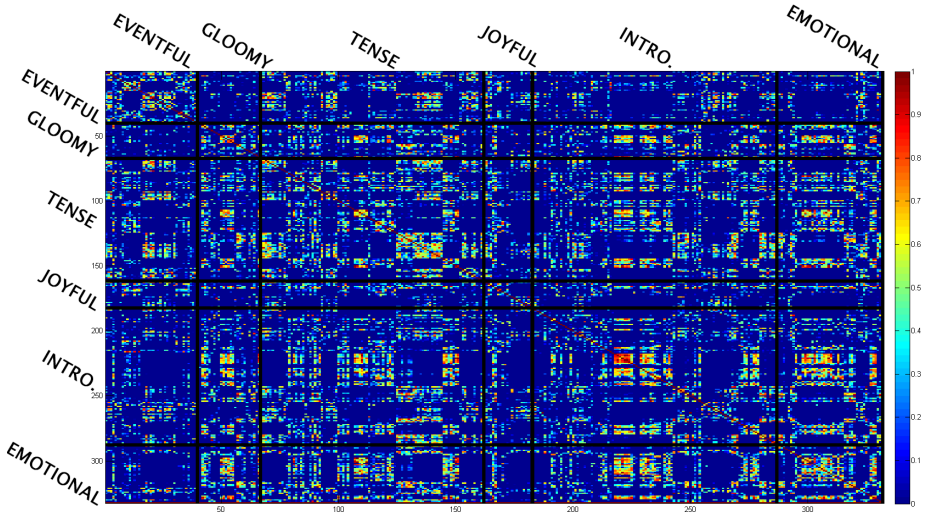


Figure 18: *Visualization of the Gram matrix when using a chi-square kernel, i.e. the kernel response for all pairs of letter histograms.*

The kernel response using the chi-square kernel is shown in Figure 18, where we can see that simply computing the histogram of letters perform way worse in terms of similarity. This indicates that the ordering of the letters in the sequence are important to the similarity measure, as both expected and desired.

An additional matter that can be investigated by examining the kernel response is how well the strings represent the content of a video. Consider Figure 19 where the input is instead sorted with respect to the order of the video episodes. After sorting by movie order, the segments are additionally sorted by class as previously. Each square in the figure thus represent similarity of the episodes, based on their string representation.The worst case scenario is if the diagonal show great similarity, while other square are dissimilar, since that would imply that one episode is only similar to itself.

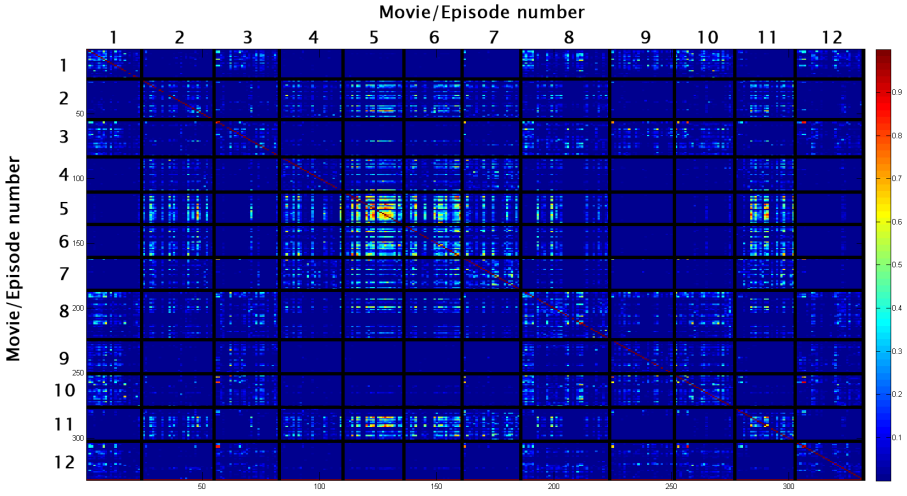


Figure 19: *Visualization of the Gram matrix when using a string kernel, i.e. the kernel response for all pairs of string sequences. The data is first sorted by the movie order, followed by again sorting with respect to labels.*

In this case, the Gram matrix show widely spread similarities, for instance episode number 5 is similar to episode 6 and 11. Without any further way of examining this until the classifier is proven reliable, the characteristics of the Gram matrix look promising for use as a computation of movie similarity.

We arrive at the moment of truth, namely the performance of the classifier. The classifier was trained using 332 sequences from 12 episodes, followed by a test set containing 53 sequences (2 episodes). The parameter C, regulating the allowed slack was set to $C = 8$. Furthermore, the kernel decay parameter $\lambda = 0.5$ was used, along with the subsequence length $n = 6$. The relevant parameters have been chosen as a combination of trial and error and the suggestions from related work. Tuning parameters for improved performance is generally interesting for a SVM, however in this case it showed unnecessary. Tests in WEKA show that these parameters affect the results scarcely, why such analysis has been excluded from this report.

10.2.1 Multi-class classification

Starting with the evaluation of the multi-class classifier, where a single classifier is created, taught by performing all combinations of 1-vs-1 classification. The performance will be measured by the Recall, Precision, Accuracy F_1-score, evaluated for both the training data and the test set. The Recall is given as

$$\text{Recall} = \frac{\text{True positive}}{\text{True Positive} + \text{False Negative}}, \tag{63}$$

where true positive means the correct classification. False negative is equivalent with a miss (which should have been detected). Precision is in similar fashion

computed as

$$\text{Precision} = \frac{\text{True positive}}{\text{True Positive + False Positive}}, \tag{64}$$

where a false positive can be seen as false alarm. The F_1-score is expressed as the calculation of

$$F_1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision + Recall}}, \tag{65}$$

which is a common way of merging Precision and Recall into one performance factor. Accuracy is simply calculated as the ratio of correctly classified instances.

The performance of the multi-class classification is shown in Table 1, where it has been evaluated using the same data as for training. The point is to show that there is in fact possible to train a classifier with the current setup.

Class	**Precision**	**Recall**	**F_1-score**	**Accuracy**
Eventful	0.667	1.0	0.8	
Gloomy	1.0	0.889	0.941	
Tense	1.0	0.968	0.984	
Joyful	1.0	0.952	0.976	
Introductory	1.0	0.875	0.933	
Emotional	0.978	0.978	0.978	
All				93.6747%

Table 1: *Performance of the multi-class classifier evaluated on the training data.*

The performance might come off as promising; over 90% accuracy should be considered good performance. However, a classifier is of poor use if not able to generalize well, i.e. perform satisfactory for previously unseen data instances. The next result, presented in Table 2, is with the purpose of measuring the performance for new data examples.

Class	**Precision**	**Recall**	**F_1-score**	**Accuracy**
Eventful	0.0	0.0	0.0	
Gloomy	0.0	0.0	0.0	
Tense	0.4	0.5	0.444	
Joyful	0.0	0.0	0.0	
Introductory	0.136	0.231	0.171	
Emotional	0.0	0.0	0.0	
All				24.5283%

Table 2: *Performance of the multi-class classifier evaluated on the test set.*

As can be seen, the performance is not very good for new, unknown data points. Not many of the few (53) data instances were correctly classified. As stated earlier, sequences from 12 episodes were used for training, while 2 for testing. To address the possibility of over-fitting or under-fitting, we attempt to shift this partitioning to see the effect in terms of performance. Table 3 shows the result of a test performed with sequences from 10 episodes for training, and 4 for testing. The performance is slightly increased, but altogether still low.

Class	Precision	Recall	F_1-score	Accuracy
Eventful	0.444	0.308	0.364	
Gloomy	0.0	0.0	0.0	
Tense	0.429	0.474	0.45	
Joyful	0.0	0.0	0.0	
Introductory	0.239	0.393	0.297	
Emotional	0.0	0.0	0.0	
All				31.4286%

Table 3: *Performance of the multi-class classifier evaluated on the test set. The number of training and test examples have been adjusted; 10 episodes for training and 4 episodes for testing.*

Further parameter tuning, shifting data sets and other kinds of manipulations seems rather pointless due to the poor performance. Instead, we will try to train multiple binary classifiers in a 1-vs-rest manner.

10.2.2 Binary classification

We begin in the same way as in the previous chapter, by revisiting the kernel response. Since the most prominent of the classes is the *eventful* atmosphere, this will be used for the example of a binary classification. The Gram matrix illustrated in Figure 20 is the same as in Figure 17, however sorted only with respect to the *eventful* label.

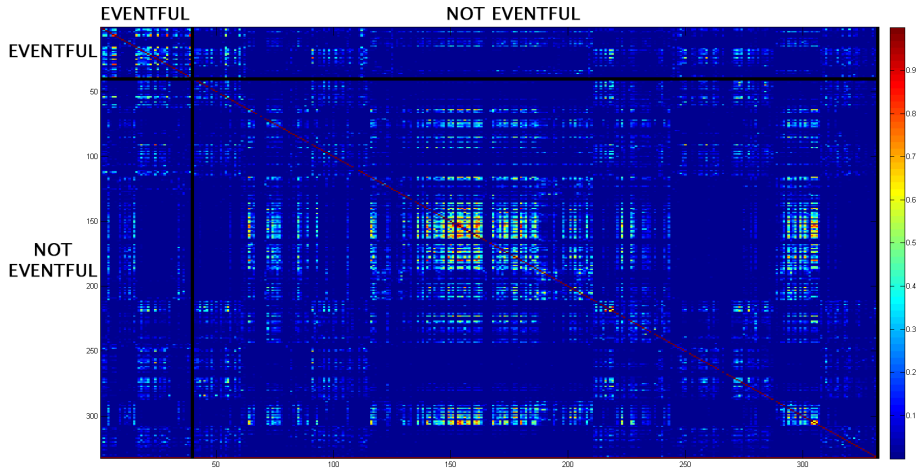


Figure 20: *Visualization of the Gram matrix using a string kernel, for binary classification. The classes are sorted as eventful and not eventful.*

The new figure show even more clearly than before that, generally, examples are the most similar to its own class. The binary classification performance will once again be evaluated both by using the training data and the test set. Furthermore, the sizes of the data sets will be adjusted similarly to the previous

results. Starting with the evaluation using training data, found in Table 4, we see that the overall accuracy is about the same as before, which is not that surprising. When using test data for evaluation, the performance is still poor, as Table 5 show. The overall accuracy is substantially increased, note however that the eventful class is still not recognized.

Class	**Precision**	**Recall**	**F_1-score**	**Accuracy**
Eventful	0.667	1	0.8	
Other	1	0.932	0.965	
All				93.9759%

Table 4: *Performance of the binary classifier evaluated on the training set.*

Class	**Precision**	**Recall**	**F_1-score**	**Accuracy**
Eventful	0	0	0	
Other	1	0.849	0.849	
All				84.9057%

Table 5: *Performance of the binary classifier evaluated on the test set.*

Instead of using 12 episodes for training, we adjust the data sets according to earlier, resulting in a training data set of 10 episodes, evaluated on 4 episodes. The classification results are summarized in Table 6. The binary classification perform better in terms of accuracy, however the *eventful* class is worse than for multi-class classification.

Class	**Precision**	**Recall**	**F_1-score**	**Accuracy**
Eventful	0.375	0.231	0.286	
Other	0.897	0.946	0.921	
All				85.7143%

Table 6: *Performance of the binary classifier evaluated on the test set. The number of training and test examples have been adjusted; 10 episodes for training and 4 episodes for testing.*

The presented results, for both classification attempts, indicate that there is need for improvements, which will be discussed in the next chapter. The approach behaves as desired, as can be seen in the gram matrices, however the results are poor. As stated earlier there are methods of tweaking the model further, such as including a validation set for training, however it is unrealistic that the model can be improved enough be useful.

11 Conclusions and Future Work

At first glance, the results are a combination of two separate, however connected, issues. The letter assignment could be tweaked to provide more representative sequences. More importantly, it seems that training is faulted by the lack of data, hindering us from discarding the sequences preemptively. Despite the

problems, the results show that the method account for the temporal alignment and ordering of sequences, as desired. This section will discuss and analyze these factors more thorough, along with presenting possible extensions to improve the setup.

11.1 Shot Representation and Sequence Construction

First and foremost, the sequences representing an atmosphere are for some classes similar. The histogram of letters in Figure 14 along with the visualization of similar sequences in Figure 15 and 16, show that sequences share similarities both in distribution of letters and the internal order. The Gram matrices also confirm that, for example *gloomy* and *emotional*, turn out to be similar. Maybe the most interesting about this is that, when manually labeling the videos, it was often difficult to separate the two atmospheres. Likewise, it was also difficult to point out the differences between *tense* and *introductory*. Considering that the human interpretation had difficulties learning these concepts, it is not unreasonable to assume that the sequences could be similar also in a digital interpretation.

The same figures also show that the *eventful* class differ significantly from the rest of the classes, indicating that the features and shot representation favor the distinction of high activity content such as motion and chaotic camera movement. The used set of features, and the clustering of frames and shots, could thus be useful for training an "action", "no-action" classifier. It would though have to be analyzed and tested more thoroughly with much more data instances. The Gram matrix sorted by episodes, Figure 19, suggest that such a setup could be used for measuring movie similarities with respect to the eventful content.

If not satisfied with only an action-classifier, it is likely that the k-means clustering of shots is too simple for this cause. The clustering results, for example Figure 13b show that there is more to be desired in terms assigning letters to shots. Besides having features that separate poorly, there is no way of showing if the assigned clusters represent the kind of video production characteristics that we are looking to describe. K-means clustering is often a long shot, and difficult to analyze depending on the domain and application.

11.1.1 Future Work

The easiest addition to determine the validity of the sequences would be to simply use more data. In the data set containing 332 sequences for training, not a single sequence was equal. Odds are that there are so many kinds of examples of a certain atmosphere that we have merely been able to collect a small fraction. An important note is that collecting more data is both time consuming (labeling) and computationally heavy. The shot clustering should preferably involve all shots used for training at once, even for different movies or episodes. In this project, the 12 episodes were on the verge of what a modern standard computer could handle. In addition, the project processed episodes from the same TV-series, hoping to address any bias in filmmaking as much as possible.

A more interesting approach would be to turn the unsupervised clustering task to a supervised machine learning assignment. The report presenting video

production techniques [5] suggest that all shots can be categorized into eight different types. If instead of clustering, one should try to learn how to recognize the types of shots, the need for many shots at once would diminish. The videos could then be processed separately, returning only the shot sequence as a string. For example, a shot involving a conversation between two people looks fairly similar for most video productions, with the camera alternatively switching between two faces. Besides time better spent, the shots would reduce the amount of data needed simultaneously as well as increase the knowledge about how the sequences are constructed.

11.2 Classification Performance

Regardless of how the sequences shape out to be, the specifics of the machine learning process also have to be analyzed. Both the multi-class and the binary classifier perform rather equally bad with the current setup. Even if Table 5 and Table 6 show high accuracy, the classifiers fail to recognize the classes. The accuracy in the binary case is thus just the same as a function repeatedly guessing *not eventful* or *other* every time. The successful separation for training data, Table 1 and 4, show that the model do not generalize well. The main thing to notice at this point is the lack of data. For 14 episodes, only 385 examples were extracted. The small data set result in badly fitted models, which is manifested by change in performance by adjusting the training and testing portions of the data. Even though the sizes of the different portions could be additionally analyzed, 385 examples do not allow much manipulation. In some of the classification attempts, over 100 support vectors were used, about a third of the entire training data set. This is partly because of the small sample of data, which obviously does not cover enough of the possible instances.

Another related matter is that, although the *eventful* class seems to be unique, a binary classifier still fail to separate eventful examples from the rest. To examine this phenomena more closely, we will look at the confusion matrix. A confusion matrix is basically a table showing the predicted class label, along with the correct answer, which in this way give more insight in what the classifier has learned. In Table 7, the confusion matrix is shown for the multi-class classifier, trying to predict the entries in the test set. The classes given as rows should be seen as the truth label, while the columns represent the predicted class. Thus, the entirely correct prediction would be a filled diagonal, while the rest of the entries are zeros.

Predicted class							
Eventful	Gloomy	Tense	Joyful	Intro.	Emotional		
4	0	2	0	6	1	Eventful	**T**
1	0	3	0	4	1	Gloomy	**r**
1	0	18	0	11	0	Tense	**u**
0	1	2	0	1	0	Joyful	**t**
3	1	12	0	11	1	Intro.	**h**
0	1	5	0	7	0	Emotional	

Table 7: *The confusion matrix for a classification of the test set, using a multi-class classifier.*

The matrix indicate that most instances are classified as *introductory* and *tense*. Not only are these two the most represented classes in the data set, but also the ones that share similarities with many different classes. The conclusion to be drawn from this is that the SVM has been unable to separate very similar concepts. The both classes mentioned constitute 232 out of the 385 data instances, allowing them to be classified correctly more frequently by pure luck.

One could argue that the binary *eventful* classifier should perform better, since the sequences differ quite a lot when looking at Figure 15 as well as the Gram matrices. Note however that the 10th most similar sequence in the figure starts to show similarities with sequences from other classes. Odds are that the 40 positive examples of eventful scenes are not nearly enough data to separate correctly. All in all, it is for both classification attempts difficult to draw any more conclusions without more data. The string kernel approach does in general show relevant behavior, although it cannot be properly evaluated at this point.

11.2.1 Future Work

As mentioned frequently by now, the greatest need in order to improve and properly evaluate this method, is to gather more examples. Doing this would be interesting both with and without improvements for the construction of data sequences. Seeing how the sequences are characterized, more data would additionally allow us to test different setups of classes, for example merge similar classes such as *Emotional* and *Gloomy* to a single class, or remove the *Introductory* class entirely. Many occurrences of the latter class was labeled when other labels did not fit, corrupting the data to some extent.

Considering how the string kernel responds, it would be exciting to see further experiments of the performance as a pure similarity measure for video. The sequences does not necessarily have to be as complex as in this thesis. Suppose an efficient computer vision algorithm that, e.g. searching for vertical edges. Building sequences based upon the quantity and magnitude of these edges might prove useful to compare the setting and environment of movies and TV-series. My guess is that similar research has been set in motion already, in the strive to properly index, annotate and categorize video content.

References

[1] "Netflix." http://www.netflix.com/. Accessed: 2014-06-14.

[2] "Amazon Instant Video." http://www.amazon.com/Instant-Video/b/ref=sa_menu_aiv?ie=UTF8&node=2858778011. Accessed: 2014-06-14.

[3] "Netflix prize." http://www.netflixprize.com/. Accessed: 2014-06-13.

[4] A. Michotte, "Perception et cognition [perception and cognition]," *Acta Psychologica*, vol. 11, p. 70–91, 1955.

[5] J. T. Smith, *An Attentional Theory of Continuity Editing*. PhD thesis, University of Edinburgh, 2005.

[6] D. Bordwell and K. Thompson, *Film art: An introduction*. New York: McGraw-Hill, 8. ed. ed., 2008.

[7] Y.-F. Ma, X.-S. Hua, L. Lu, and H.-J. Zhang, "A generic framework of user attention model and its application in video summarization," *Multimedia, IEEE Transactions on*, vol. 7, pp. 907–919, Oct 2005.

[8] J. R. Brockmole and J. M. Henderson, "Prioritizing new objects for eye fixation in real-world scenes: Effects of object–scene consistency," *Visual Cognition*, vol. 16, no. 2-3, pp. 375–390, 2008.

[9] S. Franconeri and D. Simons, "Moving and looming stimuli capture attention," *Perception & Psychophysics*, vol. 65, no. 7, pp. 999–1010, 2003.

[10] C. Folk and J. Remington, Rogerand Wright, "The structure of attentional control: Contingent attentional capture by apparent motion, abrupt onset, and color," *Journal of Experimental Psychology: Human Perception and Performance*, vol. 20, no. 2, pp. 317–329, 1994.

[11] J. T. Enns, E. L. Austen, V. D. Lollo, R. Rauschenberger, and S. Yantis, "New objects dominate luminance transients in setting attentional priority," *Journal of Experimental Psychology: Human Perception and Performance*, pp. 1287–1302, 2001.

[12] Y. Bengio, A. C. Courville, and P. Vincent, "Unsupervised feature learning and deep learning: A review and new perspectives," *CoRR*, vol. abs/1206.5538, 2012.

[13] N. Ling, "Expectations and challenges for next generation video compression," in *Industrial Electronics and Applications (ICIEA), 2010 the 5th IEEE Conference on*, pp. 2339–2344, June 2010.

[14] E. Bodin, "FAVA: Framework for Automatic Video Annotation," 2014.

[15] M. Kamel and A. Campilho, *Image Analysis and Recognition: 10th International Conference, ICIAR 2013, Póvoa do Varzim, Portugal, June 26-28, 2013. Proceedings ...* Berlin, Heidelberg: Springer Berlin Heidelberg, 2013.

[16] A. Mourão, P. Borges, N. Correia, and J. Magalhães, "Facial expression recognition by sparse reconstruction with robust features," in *Image Analysis and Recognition* (M. Kamel and A. Campilho, eds.), vol. 7950 of *Lecture Notes in Computer Science*, pp. 107–115, Springer Berlin Heidelberg, 2013.

[17] S. Bertoluzza, G. Maggi, and S. Tomatis, "A modular registration algorithm for medical images," in *Image Analysis and Recognition* (M. Kamel and A. Campilho, eds.), vol. 7950 of *Lecture Notes in Computer Science*, pp. 467–474, Springer Berlin Heidelberg, 2013.

[18] C. Craye and F. Karray, "Multi-distributions particle filter for eye tracking inside a vehicle," in *Image Analysis and Recognition* (M. Kamel and A. Campilho, eds.), vol. 7950 of *Lecture Notes in Computer Science*, pp. 407–416, Springer Berlin Heidelberg, 2013.

[19] A. Savchenko, "Directed enumeration method in image recognition," *Pattern Recognition*, vol. 45, no. 8, pp. 2952 – 2961, 2012.

[20] A. Savchenko, "Adaptive video image recognition system using a committee machine," *Optical Memory and Neural Networks*, vol. 21, no. 4, pp. 219–226, 2012.

[21] J. Ladan and E. Vrscay, "The discrete orthonormal stockwell transform and variations, with applications to image compression," in *Image Analysis and Recognition* (M. Kamel and A. Campilho, eds.), vol. 7950 of *Lecture Notes in Computer Science*, pp. 235–244, Springer Berlin Heidelberg, 2013.

[22] D. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[23] V. I. Pavlovic, R. Sharma, and T. S. Huang, "Visual interpretation of hand gestures for human-computer interaction: A review," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 19, pp. 677–695, July 1997.

[24] M.-C. Roh, B. Christmas, J. Kittler, and S.-W. Lee, "Robust player gesture spotting and recognition in low-resolution sports video," in *Computer Vision – ECCV 2006* (A. Leonardis, H. Bischof, and A. Pinz, eds.), vol. 3954 of *Lecture Notes in Computer Science*, pp. 347–358, Springer Berlin Heidelberg, 2006.

[25] W. Hu, T. Tan, L. Wang, and S. Maybank, "A survey on visual surveillance of object motion and behaviors," *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, vol. 34, pp. 334–352, Aug 2004.

[26] H. Zhou and H. Hu, "A survey - human movement tracking and stroke rehabilitation," 2004.

[27] M. A. R. Ahad, *Computer Vision and Action Recognition.* Atlantis Press, 2011.

[28] K. Schindler and L. Van Gool, "Action snippets: How many frames does human action recognition require?," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pp. 1–8, June 2008.

[29] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri, "Actions as space-time shapes," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 29, pp. 2247–2253, Dec 2007.

[30] A. Efros, A. Berg, G. Mori, and J. Malik, "Recognizing action at a distance," in *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*, pp. 726–733 vol.2, Oct 2003.

[31] J. Niebles and L. Fei-Fei, "A hierarchical model of shape and appearance for human action classification," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pp. 1–8, June 2007.

[32] H. Jhuang, T. Serre, L. Wolf, and T. Poggio, "A biologically inspired system for action recognition," in *Computer Vision, 2007. ICCV 2007. IEEE 11th International Conference on*, pp. 1–8, Oct 2007.

[33] V. Chasanis, A. Likas, and N. Galatsanos, "Scene detection in videos using shot clustering and symbolic sequence segmentation," in *Multimedia Signal Processing, 2007. MMSP 2007. IEEE 9th Workshop on*, pp. 187–190, Oct 2007.

[34] Z. Rasheed and M. Shah, "Scene detection in hollywood movies and tv shows," in *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, vol. 2, pp. II–343–8 vol.2, June 2003.

[35] N.-X. Lian, Y.-P. Tan, and K. L. Chan, "Efficient video retrieval using shot clustering and alignment," in *Information, Communications and Signal Processing, 2003 and Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint Conference of the Fourth International Conference on*, vol. 3, pp. 1801–1805, Dec 2003.

[36] Z. Rasheed and M. Shah, "Detection and representation of scenes in videos," *Multimedia, IEEE Transactions on*, vol. 7, pp. 1097–1105, Dec 2005.

[37] J. Wang, B. Li, W. Hu, and O. Wu, "Horror movie scene recognition based on emotional perception," in *Image Processing (ICIP), 2010 17th IEEE International Conference on*, pp. 1489–1492, Sept 2010.

[38] I. H. Witten and E. Frank, *Data mining : practical machine learning tools and techniques*. Amsterdam: Morgan Kaufmann, 2. ed. ed., 2005.

[39] I. Koprinska and S. Carrato, "Temporal video segmentation: A survey," in *Signal Processing: Image Communication*, pp. 477–500, 2001.

[40] K. Rieck, "Similarity measures for sequential data," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 1, no. 4, pp. 296–304, 2011.

[41] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins, "Text classification using string kernels," *J. Mach. Learn. Res.*, vol. 2, pp. 419–444, Mar. 2002.

[42] P. Dollar, Z. Tu, H. Tao, and S. Belongie, "Feature mining for image classification," in *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pp. 1–8, June 2007.

[43] D. Kong, *Image Annotation And Feature Engineering Via Structural Sparsity And Low Rank Approximation.* PhD thesis, University of Texas at Arlington, 2013.

[44] C. S. Won, D. K. Park, and S.-J. Park, "Efficient use of mpeg-7 edge histogram descriptor," *ETRI Journal*, vol. 24, pp. 23–30, February 2002.

[45] A. Ferman, A. Tekalp, and R. Mehrotra, "Robust color histogram descriptors for video segment retrieval and identification," *Image Processing, IEEE Transactions on*, vol. 11, pp. 497–508, May 2002.

[46] J. S. Boreczky and L. A. Rowe, "Comparison of video shot boundary detection techniques," 1996.

[47] P. Kay, B. Berlin, L. Maffi, and W. R. Merrifield, *World Color Survey.* CSLI Publications, July 2009.

[48] D. K. S. D. R. K. U. Gargi, S. Oswald, "Evaluation of video sequence indexing and hierarchical video indexing," in *Storage and Retrieval for Image and Video Databases III*, vol. 2420, pp. 144–151, March 1995.

[49] M. Miyahara and Y. Yoshida, "Mathematical transform of (r, g, b) color data to munsell (h, v, c) color data.," in *Proc. SPIE 1001, Visual Communications and Image Processing '88: Third in a Series*, October 1988.

[50] P. Salembier and T. Sikora, *Introduction to MPEG-7: Multimedia Content Description Interface.* New York, NY, USA: John Wiley & Sons, Inc., 2002.

[51] M. Sonka, V. Hlavac, and R. Boyle, *Image processing, analysis and machine vision.* London: Chapman & Hall, 1. ed. ed., 1993.

[52] B. K. Horn and B. G. Schunck, "Determining optical flow," *Artificial Intelligence*, vol. 17, no. 1–3, pp. 185 – 203, 1981.

[53] R. W. Hamming, *Numerical methods for scientists and engineers.* New York: Dover, 2. ed. ed., 1986.

[54] C. E. Shannon, "A mathematical theory of communication," *SIGMOBILE Mob. Comput. Commun. Rev.*, vol. 5, pp. 3–55, Jan. 2001.

[55] A. Younkin and P. Corriveau, "Determining the amount of audio-video synchronization errors perceptible to the average end-user," *Broadcasting, IEEE Transactions on*, vol. 54, pp. 623–627, Sept 2008.

[56] D. Mcennis, C. Mckay, and I. Fujinaga, "Jaudio: A feature extraction library," in *International Conference on Music Information Retrieval*, 2005.

[57] O. Maimon and L. Rokach, *Data Mining and Knowledge Discovery Handbook.* Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2005.

[58] C. J. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, pp. 121–167, 1998.

[59] T. Poggio, S. Mukherjee, R. Rifkin, R. A, and A. Verri, "b," tech. rep., Massachusetts Institute of Technology, Cambridge, MA, July 2001. CBCL Paper 198/AI Memo 2001-011.

[60] R. Fletcher, *Quadratic Programming*, pp. 229–258. John Wiley and Sons, Ltd, 2000.

[61] S. M. Sinha, *Mathematical Programming : Theory and Methods*. Burlington, MA, USA: Elsevier Science and Technology, 2006.

[62] T. Kudo and Y. Matsumoto, "Chunking with support vector machines," in *Proceedings of the Second Meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies*, NAACL '01, (Stroudsburg, PA, USA), pp. 1–8, Association for Computational Linguistics, 2001.

[63] E. Osuna, R. Freund, and F. Girosi, "Support vector machines: Training and applications," tech. rep., Cambridge, MA, USA, 1997.

[64] J. Platt, "Fast training of support vector machines using sequential minimal optimization," in *Advances in Kernel Methods - Support Vector Learning* (B. Schoelkopf, C. Burges, and A. Smola, eds.), MIT Press, 1998.

[65] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, vol. 11, pp. 10–18, Nov. 2009.

[66] S. Tian, J. Yu, and C. Yin, "Anomaly detection using support vector machines," in *Advances in Neural Networks – ISNN 2004* (F.-L. Yin, J. Wang, and C. Guo, eds.), vol. 3173 of *Lecture Notes in Computer Science*, pp. 592–597, Springer Berlin Heidelberg, 2004.

[67] S. Liao and M. Duan, "Sketch recognition via string kernel," in *Natural Computation (ICNC), 2012 Eighth International Conference on*, pp. 101–105, May 2012.

[68] R. Ewerth and B. Freisleben, "Video cut detection without thresholds," in *Proc. of the 11th Workshop on Signals, Systems and Image Processing*, pp. 227–230, 2004.

[69] M. Qian, "LAML: Linear Algebra and Machine Learning." `http://sourceforge.net/projects/lamal/files/`, 2014. Version 1.4.

[70] P.-D. Belanger, "EKmeans: Enhanced/Equal K-means Clustering." `https://code.google.com/p/ekmeans/`, 2012. Version: svn-r27.

[71] M. Lux and S. A. Chatzichristofis, "Lire: Lucene image retrieval: An extensible java cbir library," in *Proceedings of the 16th ACM International Conference on Multimedia*, MM '08, (New York, NY, USA), pp. 1085–1088, ACM, 2008.