Hierarchical Segmentation of Videos into Shots and Scenes using Visual Content

prepared by

Andrew Thompson

supervised by

Robert Laganière and Pierre Payeur

Thesis submitted to the
Faculty of Graduate and Postdoctoral Studies
In partial fulfillment of the requirements
For the M.A.Sc. degree in
Electrical and Computer Engineering

School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa

Abstract

With the large amounts of video data available, it has become increasingly important to have the ability to quickly search through and browse through these videos. With that in mind, the objective of this project is to facilitate the process of searching through videos for specific content by creating a video search tool, with an immediate goal of automatically performing a hierarchical segmentation of videos, particularly full-length movies, before carrying out a search for a specific query.

We approach the problem by first segmenting the video into its film units. Once the units have been extracted, various similarity measures between features, that are extracted from the film units, can be used to locate specific sections in the movie.

In order to be able to properly search through a film, we must first have access to its basic units. A movie can be broken down into a hierarchy of three units: frames, shots, and scenes. The important first step in this process is to partition the film into shots. Shot detection, the process of locating the transitions between different cameras, is executed by performing a color reduction, using the 4-*Histograms* method to calculate the distance between neighboring frames, applying a second order derivative to the resulting distance vector, and finally using the automatically calculated threshold to locate shot cuts.

Scene detection is generally a more difficult task when compared to shot detection. After the shot boundaries of a video have been detected, the next step towards scene detection is to calculate a certain similarity measure which can then be used to cluster shots into scenes. Various keyframe extraction algorithms and similarity measures from the literature were considered and compared. Frame sampling for obtaining keyframe sets and Bhattacharya distance for similarity measure were selected for use in the shot detection algorithm.

A binary shot similarity map is then created using the keyframe sets and Bhattacharya distance similarity measure. Next, a temporal distance weight and a predetermined threshold are applied to the map to obtain the final binary similarity map. The last step uses the proposed algorithm to locate the shot clusters along the diagonal which correspond to scenes.

These methods and measures were successfully implemented in the *Video Search Tool* to hierarchically segment videos into shots and scenes.

Acknowledgements

I would like to express my gratitude to my supervisors, Robert Laganière and Pierre Payeur, for giving me the opportunity to further my education at the University of Ottawa, and for their expertise, kindness, and patience during the process.

My gratitude goes to Adele Reinhartz for providing an interesting and challenging project, and to Diana Inkpen and Martin Scaiano for their assistance with the text-based subtitle searching procedure implemented in the Video Search Tool. Thanks also goes to Bogdan-Emanuel Ionescu for his help in implementing his shot detection algorithm and to Jason Shim for his help performing the manual segmentation of the films.

A special thanks to my family for their support and encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

With the large amounts of video data available, it has become increasingly important to have the ability to quickly search through and browse through these videos. Before performing any kind of search, the first step is usually to hierarchically segment the video into its film units. Typically, these units consist of the video frames, shots, and scenes. Frames are the single images that make up the video, shots are the contiguous sequence of frames that have been captured from a single camera, and scenes, which are much more difficult to define because of the semantics involved, can be given the general definition of being the groups that are made up of multiple shots that revolve around a single dramatic person, incidence, or location. Once the units have been extracted, various similarity measures between features that are extracted from the film units can be used to locate specific sections in the movie.

The objective of this project is to facilitate the process of searching through videos for specific content. However, the immediate goal is to be able to automatically perform a hierarchical segmentation of videos, specifically full-length movies, using only their visual content.

There have been several different approaches to video analysis and hierarchical segmentation of video in recent years. The following paragraphs will highlight various methods and techniques that have been employed by different authors, throughout the literature, for video analysis, video segmentation, and video retrieval in particular.

1.1 Video Analysis and Hierarchical Segmentation

1.1.1 Background Work

The first work to be examined is that of Zhu *et al.* [1]. The authors employ a shot-based approach to classify shots hierarchically for medical videos and to group them into different categories for retrieval.

Shot detection is performed using a MPEG compressed video method which uses the DC images and an automatic threshold detection method. Once the shots are obtained, the 10^{th} frame of each shot are selected as the representative keyframes and HSV color histograms and Tamura textures (coarseness, directionality, contrast) are extracted from each keyframe.

Shots are then compared with their neighbors to create groups of shots by clustering those which are similar spatially, with respect to similar background, or similar temporally. The resulting groups are classified as being either temporally related or spatially related.

Once the groups are obtained, a representative shot, that is the shot that best represents the content of the group, is selected. Groups are then merged into scenes if the similarity, measured as the best similarity between the groups' shots, between successive groups is greater than a threshold. A representative group for each scene is then selected. The last segmentation step is to cluster similar scenes by comparing the groups of one scene with the groups of another scene. The scene grouping precess is continued until the predetermined number of scene clusters is reached.

Having finished the video segmentation, event mining is performed to detect certain events within scenes. Several features are extracted with the goal of classifying scenes into 3 different medical categories: presentations, clinical operations, or doctor to patient dialogs. Visual features are extracted from keyframes to classify them into several different types: slides and clip-art frames, black frames, frames with faces, frames with large skin areas, and frames with blood red regions. Audio features are then used to determine when there is a speaker change between shots. The audio for each shot is then divided into clips of at least 2 seconds in length and these clips are classified into clean speech and non-clean speech segments using 14 audio features and a Gaussian Mixture Model classifier. The clip most like the speech clip is selected as the representative audio clip of the shot. These representative clips are then compared to determine if the audio from different shots belong to the same speaker.

The classification of scenes into the predefined groups is then done based on the following definitions: presentation scenes contain slide and clip-art frames, a minimum of 1 face frame, no speaker change between shots, and a minimum of 1 temporally related group. Dialog scenes contain many face frames, speaker changes at neighboring shots which both contain face frames, a minimum of 1 spatially related group, and at least 2 shots that belong to the same speaker. Clinical Operation scenes contain no speaker changes, at least one blood red or large skin area frame, and at least half of the keyframes from the shots should contain skin regions.

The authors create a tool for skimming through the videos, a user can skim through the videos using any of the videos units, either keyframes, shots, groups, scenes, or clusters of scenes and also by the type of scene.

Zhu *et al.* [2] organize videos by first segmenting them into hierarchical units then by extracting several features for retrieval purposes.

Shot detection is performed using an MPEG compressed video domain method which uses information from I, B, and P-frames. An I-frame, an Intra-coded frame, contains the entire information of a single frame from the video. The other two types contain part of the image information and require less space to store. A P-frame, a Predicted frame, only contains the information that has changed from the previous I-frame, while a B-frame, a Bi-directional frame, contains information that has changed from the previous frame or is different from the information in the following frame.

Again utilizing the information from MPEG compressed videos, camera motion classification is performed. Motion vector information is extracted from macroblocks in the video's P-frames and a 14-bin feature vector is used to characterize the motion vectors of each frame. This vector is made up of a 2-bin motion vector energy histogram, a 4-bin motion vector orientation histogram, a 4-bin motion vector mutual relationship histogram, and a 4-bin motion vector vertical mutual relationship histogram. These feature vectors are examined for certain conditions to classify them into one of the camera motion types: either pan, tilt, zoom, rolling, or still motion. The detected motion should last at least 3 P-frames to be considered and to be stored in the final motion feature vector for the shot.

Subsequently keyframes are extracted from each shot to represent its content. Keyframes are extracted for every type of motion found in the shot, so for each motion type, the motion magnitude of each P-frame is calculated and the keyframes are selected with the following criteria: for still motion, the frame with the smallest motion magnitude of the shot is chosen as the keyframe. For all other motion types, the largest motion magnitude is first located, then the frames with the minimum magnitude points to the left and right of this maximum point are selected as keyframes.

After shot detection, motion classification, and keyframe selection, hierarchical video content organization is performed using the same methods of [1] to obtain groups, scenes, and clusters of scenes.

The authors select various features to assess similarity between videos and for use in the retrieval process. A 256-bin HSV color histogram and a 10-bin coarseness texture histogram are extracted at the frame level and, at the shot-level, an average color histogram, camera motion, keyframe matching, and the shot length are used.

In [3], the authors build a content-based video browsing and retrieval system. They organize the video by performing a segmentation to locate shots and scenes, then extract keyframes from each scene to represent the content. In their application, they are able to browse by scene or keyframe and perform searches using textual indices.

The first step in the segmentation process is to divide the video into an audio stream and a video stream. Using the signal energy, non-silent segments are detected in the audio stream. These audio segments are then classified into either speech, music, or environmental sound.

Next, shot detection is performed and an expanding window grouping algorithm is used to cluster shots into scenes. Keyframes are then extracted from each scene using an adaptive keyframe extraction technique. Using this method, for each scene, frames are clustered based on the similarity of their HSV color histograms and keyframes are extracted from the clusters that have a large enough size.

Scene detection is performed by analyzing both the audio and visual segments. The authors postulate that there are 3 types of scenes that can be classified and detected based on the found audio and visual segments. The first category is audio-visual scenes which contain color and sound consistency and are detected when the audio segmentation and visual segmentation produce common boundaries. Next are the audio scenes which are detected when one audio segment contains multiple video segments. The last scenes are classified as dialog scenes. These are located when the visual similarity analysis between shots indicates a similar environment and when the entire segment contains multiple speaker changes.

Text analysis is then performed to label the located scenes. This is done by applying a text extraction and recognition algorithm on each of the scene's keyframes. The resulting vector of words is used to represent the content of a scene.

Dong and Li [4] perform hierarchical segmentation of documentary videos. Their method first employs a combination of audio and text segmentation to locate scene boundaries, scenes are then further divided into shots, keyword extraction is performed, and video scene summarization is accomplished using the extracted keywords.

The authors decide to use a top down approach at video segmentation, and find the scenes before locating the shots. Audio segmentation is first performed using signal energy to classify speech and non-speech segments. The length of the non-speech segments are calculated and used to perform a pause detection-based approach to cluster audio segments. Text segmentation using the TextTiling method is applied to the video transcript. This method locates topic boundaries in the transcript using the similarity between neighboring text blocks. The audio and text boundaries that are shared are then chosen as the scene boundaries. Once the scenes have been extracted, shots are located using the local color histogram difference between successive frames in a scene.

In order to be able to quickly locate different videos and scenes, the authors first label each scene with the keywords from the video scene's transcript that have a frequency of appearance greater than a threshold. These keywords are also used in conjunction with web-services to extract relevant, public domain documents. These documents are found with various annotations such as titles, authors, and URLs. Next, video scene summarization is performed using a text-based method. To provide an idea of the content of each scene, the first few 'utterances', which are speech units that are usually bounded by silence, are extracted from the transcript. They are located by expanding from keyword to keyword or from keyword to silence segment.

Fu *et al.* [5] have chosen to perform a hierarchical video segmentation for use in a video browsing application. They have three main stages to their video analysis procedure: first, shot detection is performed, then a good measure for video shot similarity is selected, and lastly scene changes are detected.

Before shot detection, the authors perform a color quantization by applying a color transformation from the RGB color space to the HSV color space and grouping colors that belong to the same quantization level to obtain a total of 72 colors. A 72-bin color histogram is then obtained for each of the film's frames and normalized Euclidean distance is used to obtain the similarity between neighboring frames, compared as the histogram difference.

Shot detection is accomplished using a sliding window method. The mean and standard deviation of histogram difference is calculated within each window and if the standard deviation is smaller that a predetermined threshold, a twin-threshold method is applied to determine if the current frame is a shot boundary. The two thresholds, used in the twin-threshold method, are obtained using the previously calculated mean and standard deviation. The lower threshold is equal to the mean plus twice the standard deviation, and the higher threshold is an additional 2 standard deviations away. The first threshold is used to locate potential shot transition candidates and the second threshold is used to filter these candidates to locate the actual boundary. Once the shot boundaries are located, the middle frame of each shot is selected as its representative keyframe.

Their next step is to select a shot similarity measure to be used in the scene detection process. The measure selected is a combination of color, texture, and keyword similarity. The intersection value between the color histograms of the representative keyframes is used as the color similarity value between different shots. A Homogeneous Texture Descriptor is used as the texture similarity measure. Next, keywords are extracted from the transcript of each shot, placed in a vector, then arranged by their frequency. To obtain what the authors refer to as 'semantic' similarity, the keyword vector of one shot is compared with the vector of another shot. These three similarity measures are combined into a weighted sum to represent the overall similarity between shots.

Scene detection is performed using the Splitting and Merging Forces method, which will be explained in detail later in the Scene Detection chapter, combined with their shot similarity measure.

1.1.2 System Limitations

Without performing a thorough testing of the systems presented above, certain limitations can be noticed.

A few of the systems are built for very specific types of videos, for example medical videos or educational documentary videos. These will, most likely, not perform as well on different types of videos such as full-length feature films.

Other systems select keyframes in what seem like unreliable manners, such as selecting the 10^{th} frame of the shot or the middle frame of the shot. A more extensive keyframe extraction method would help to obtain keyframes that better represent the entire content of the shot.

In the second system presented [2], before performing segmentation, the weights for the similarity measure must be manually selected, similarly in the fourth system [4], the text segmentation parameters are manually tuned. The segmentation process, including the selection of weights and parameters, should be automatic or done prior to the creation of the system.

The user interfaces for the constructed video skimming or video browsing tools appear to be, in some cases, overly complex in such a way that a first time user of the system would not be able to immediately use it. It should be easy and intuitive to use such that any user can easily employ the tool without any prior knowledge of the system.

Lastly, none of the papers presented above attempt to properly define what a scene is. They give short general definitions but do not indicate how they have constructed their scene ground truths that were used in the testing processes. This is important because without a proper definition the semantics, a scene can be interpreted very differently by different people.

These are some of the issues that will be examined when creating the *Video Search Tool*.

Having examined different approaches, it is evident that video analysis and hierarchical segmentation systems use different combinations of features extracted from images, audio, and text to achieve satisfactory segmentation and to search through the resulting film units.

1.2 Objectives

As was previously stated, the objective of this project is to facilitate the process of searching through videos for specific content by constructing a *Video Search Tool* that will perform segmentation of videos, then is able to perform keyword queries through the video's subtitle transcript. The immediate goal is to perform a hierarchical segmentation of videos into shots and scenes using visual content. It should be noted that offline methods are examined.

A thorough examination of methods for shot detection, keyframe extraction, scene detection, along with an inspection of shot similarity measures will be done. Selected shot detection and keyframe extraction methods will be chosen for implementation and a new algorithm for locating scene boundaries will be explained and tested against current methods. The chosen methods and measures will then be implemented into the *Video Search Tool.*

The works presented in this thesis were put together with the final goal of creating a *Video Search Tool* that first segments the video into shots and scenes, and can also perform keyword queries through the video's subtitle transcript.

To reiterate, the objectives are:

- Experimentally evaluate methodologies for keyframe extraction and similarity measures estimation;
- Develop an innovative technique to cluster similar shots into scenes, strictly relying on visual features from the shots;
- Merge the selected and newly developed approaches into a Video Search Tool.

1.3 Structure of the Thesis

The following chapters will involve the multiple steps taken to create a *Video Search Tool.* For performing automatic segmentation of videos, various techniques, measures, and their combinations will be presented and evaluated. The subsequent chapters will be presented in the following order: Shot Detection, Keyframe Extraction and Shot Similarity, and Scene Detection.

The document is structured in the following way:

- **Chapter 2** will present several shot detection methods that have been previously employed in the literature as a background to the shot detection method that we have chosen to implement.

- **Chapter 3** will examine different keyframe extraction methods along with several frame and shot similarity measures that have been previously employed in the literature. Certain of these methods and measures will be chosen for testing and comparison in order to select an appropriate keyframe extraction method and shot similarity measure to be used in the scene detection process.

- **Chapter 4** will present various scene detection methods found throughout the literature, followed by a detailed description of the proposed scene detection method. Several methods, along with our proposed method, are then tested and compared.

- **Chapter 5** will conclude the thesis with an overview of the works presented, our contributions, and future work.

1.4 Contributions

The following items describe the unique contributions of this thesis.

- A unique approach to evaluating similarity measures and keyframe extraction methods for their application to video segmentation;

- An innovative technique for performing scene detection based on visual content;

- An implementation of the selected and newly developed approaches and methods into a Video Search Tool.

Chapter 2

Shot Detection

In order to be able to properly search through a film, we must first have access to its basic units. A movie can be broken down into a hierarchy of three units: frames, shots, and scenes. The movie's frames are already available, so the next important step in this process is to partition the film into shots. A shot can be defined as a contiguous sequence of frames captured by a single camera, that is a same viewpoint, and so shot detection is the process of locating the transitions between each of these cameras.

Shot detection is an important first step, however, not the main focus. This chapter will first describe various previous methods that have been commonly employed to detect shots. This will serve as a preliminary introduction to the shot detection method that has been chosen, which will be subsequently presented.

2.1 Previous Methods

There have been many different methods for shot detection with most attempting to accurately locate the transitions between shots such as cuts, fades, and dissolves. Generally, cuts, which can be defined as abrupt transitions between shots, make up the vast majority of transitions in films and as such, we will focus on detecting cuts as the boundaries between shots.

Most cut detection algorithms follow a general path. First, certain parameters will be extracted from each frame or image, next these characteristics will be compared using a similarity measure, and lastly a threshold will be applied to determine the locations of cuts.

The following will present several different types of methods for cut and shot detection that follow this path. These algorithms can be divided into edge-based methods, motion-based methods, methods that use computed features of compressed video, such as MPEG, and histogram-based methods.

In [6], Zabih *et al.* apply an edge-based method to detect shot transitions. Their idea is to use the location of intensity edges in successive images to find when an old edge is far from a new edge. Their method first involves performing a Canny edge detection on neighboring frames, then determining the translation needed to align both images. Next, the proportion of entering edge pixels, ρ_{in}, and the proportion of exiting edge pixels, ρ_{out}, are determined using a threshold distance, r.

ρ_{in} is the fraction of edge pixels in the second image that are greater than a distance r from the edges in the first image. It is calculated as:

$$\rho_{in} = 1 - \frac{\sum_{x,y} \bar{E}[x + \delta x, y + \delta y] E'[x, y]}{\sum_{x,y} E[x + \delta x, y + \delta y]} \tag{2.1}$$

ρ_{out} is the fraction of edge pixels in the first image that are greater than a distance r from the edges in the second image. It is calculated as:

$$\rho_{out} = 1 - \frac{\sum_{x,y} E[x + \delta x, y + \delta y] \bar{E}'[x, y]}{\sum_{x,y} E[x, y]} \tag{2.2}$$

where E and E' are the first and second binary images obtained after edge detection, $\bar{E}$ and $\bar{E}'$ are those binary images with each edge pixel dilated by a radius r, and δx and δy are the translations required to align both images.

After these two measures are obtained, the maximum between the two is kept as its associated ρ value. Once these values are acquired for all of the film's frames, the peaks of ρ indicate the location of a shot transition.

Osian and Van Gool [7], employ a motion-based method that utilizes global motion compensation information and then apply an adaptive threshold to perform shot detection. They compare each set of frames by computing an affine transformation to align both frames before comparing them. The transformation is calculated in two steps. First each image is divided into 20 x 15 blocks, and the best motion vector is calculated for each individual block. Next, a hierarchical search is executed to determine a transformation that approximates the entire set of motion vectors. They use the average pixel difference between the motion compensated images as the similarity measure. Once this metric is obtained for the entire film, they make use of a sliding window of 16 frames to locate cuts. For every position of the sliding window, they inspect for the following conditions to determine if there is a cut between the two middle frames of the window.

1. If the current difference of the middle frames is greater than a threshold, and
2. If the current difference is the largest within the current position of the window, then
3. If the average difference of the previous frames is greater than the average difference of the next frames, then a cut is marked between the two middle frames. Otherwise
4. If the current difference is greater than the variance within the window $\times$ the average window difference, and
5. If the trend in the first half of the window is a linearly increasing difference and the trend in the second half is linearly decreasing difference, and the maximum of the two differences is not comparable to the difference of the middle frames, then a cut is marked between the two middle frames.

Wang *et al.* [8] use the information from the MPEG compressed domain, namely macroblock type information of B-frames and P-frames, to locate shot boundaries. They state that at a shot change location there is a specific type of macroblock information. Before a shot change, most macroblocks in B-frames are forward motion compensated. Macroblocks in P-frames are intra-coded before a shot change because of a change in video content. After a shot change, most of the macroblocks in B-frames are backward motion compensated. Using this knowledge, they are able to determine when a shot transition occurs.

In [9], Yeo and Liu also use information obtained from the MPEG compressed domain to perform shot detection. They extract DC images from I-frames, B-frames, and P-frames because they are spatially reduced version of the original images that retain most of the global image features and the authors believe they are ideal for shot detection. Once they have obtained the DC images, they smooth the images in order to make them less sensitive to object and camera motion. Next, pixel differences between successive images are computed and a sliding window method is applied to determine where to mark a shot boundary. There are two conditions that must be met in order for a boundary to be marked: firstly, the pixel difference should be the maximum within the sliding window and secondly, it should be n times greater than the next highest difference. The authors have experimentally determined the n value to be between 2.0 and 3.0.

Histogram-based shot detection methods are among the most popular. Rasheed and Shah [10] employ 16-bin HSV normalized color histograms, composed of 8-bins for Hue, 4-bins for Saturation, and 4-bins for Value, to represent frame content. Histogram intersection is then performed on each pair of neighboring frames to obtain a distance metric. The distances are then compared to a threshold, T_{color}, to determine the locations of cuts. A cut is declared if

$$D_{int}(f_i, f_j) = \sum_{b \in allbins} min(H_i(b), H_j(b)) < T_{color} \tag{2.3}$$

Chasanis *et al.* [11] use a very similar method but choose to use χ^2 distance to measure the difference between frames before applying a threshold, T_{sh}, which they have experimentally determined to be 0.15. In this case, a shot boundary is marked if

$$D_{\chi^2}(f_i, f_j) = \sum_{b \in allbins} \frac{(H_i(b) - H_j(b))^2}{H_i(b) + H_j(b)} > T_{sh} \tag{2.4}$$

Le *et al.* [12] expand on this method by attempting to eliminate false shot boundaries caused by large object motion. The magnitude of the motion vector is calculated for each frame and, for a previously detected cut, if the magnitude is greater than a threshold then the cut is determined to be falsely placed due to motion and is removed.

Liu *et al.* [13] also use a histogram-based method. First they divide each frame into a 4×4 grid, obtaining 16 sections of the image, before calculating a 16-bin HSV color histogram for each individual block. To obtain the distance between two frames, the Euclidean distance is calculated for each individual block and an average of these distances is used as the final distance measure. The resulting distances are then compared to a threshold, T_d, and a shot boundary is detected if

$$D_{eucl}(f_i, f_j) = C_1 \sum_{n=1}^{16} \left(C_2 \sqrt{\sum_{b=1}^{16} (H_i(n,b) - H_j(n,b))^2} \right) > T_d \tag{2.5}$$

where the normalization constants $C_1 = \frac{1}{16}$ and $C_2 = \frac{1}{\sqrt{2}}$, and n is the n^{th} sub-image, and b is the b^{th} bin in the histogram of the sub-image.

A feature tracking based method is employed in [14]. They begin by quantifying inter-frame differences by first tracking features from frame to frame, pruning false tracking with the use of a Minimum Spanning Tree, and lastly computing the square of percentage feature loss.

This measure is then used to compute a linear discriminator. A histogram of the squared percentage feature loss is obtained and the PDF (probability density function), along with the first derivative of the PDF are calculated. These are used to locate where tracking succeeded and where tracking failed, in other words, to locate the shot cuts.

2.2 Selected Method

Having presented different techniques for shot detection, cut and transition detection, the shot detection method that was selected for use in our video segmentation process will be presented. This shot detection method was taken from Ionescu, Buzuloiu, Lambert, and Coquin's "Improved Cut Detection for the Segmentation of Animation Movies" [15]. It was selected because it offers ease of implementation and has demonstrated high efficiency throughout our experimentation process.

2.2.1 Color Reduction

A typical movie frame can easily be made up of millions of different colors, and with such a large number of possible RGB combinations, calculating the color histogram for each of the film's frames becomes a very tedious task.

Thus, in order to simplify this process and greatly reduce the computation time, a color quantization, or color reduction, should be applied before any other steps are performed. Before applying a color reduction technique, a spatial sub-sampling of the original image is done such that the new image is reduced to a size 4 times smaller than the original. This step will also greatly reduce the computational complexity of the subsequent calculations.

Color reduction is then performed in the Lab space using the fixed "Webmaster" palette [16], which is composed of 216 colors. Every color of the image is then replaced by the closest palette color.

Once the colors of an image have been quantized, its color histogram is then computed for use in the cut detection algorithm.

2.2.2 4−Histograms Method

The general idea behind the cut detection algorithm, is to measure the visual discontinuity between frames. This is done by calculating the Euclidean distance between each successive frame's color histogram. In order to reduce the effects of object movement the 4-histograms method is used. This method consists of simply splitting each image into four quadrants, calculating their respective color histograms, then calculating four Euclidean distances between the quadrants' histograms of successive images.

$$d_E^j(k) = \sqrt{\left(\sum_{c=1}^{N_c} \left[H_{(k+1)}^j(c) - H_k^j(c) \right]^2 \right)} \tag{2.6}$$

where N_c is the total number of colors, c is the index of the color, k is the image number, and j is the quadrant number.

The average of these four distances is then used as the final distance between the two frames.

$$d_E(k) = \frac{1}{4} \sum_{j=1}^{4} d_E^j(k) \tag{2.7}$$

2.2.3 2^{nd} Derivative Method

After having obtained each distance for the entire length of the film, to better locate the cuts and further reduce the influence of movement, the 2^{nd} Derivative method is applied. Since a cut is represented by a large enough dissimilarity between frames, applying a derivative to the distance vector d_E will help to distinguish these points. It has been determined that performing up to the second order derivative will effectively accomplish this without increasing or adding noise to the distance vector.

The derivatives are calculated as the difference between the distances, $d_E(k)$, estimated on successive frames. The first derivative is:

$$\dot{d_E}(k+1) = \begin{cases} d_E(k+1) - d_E(k), & \text{if } d_E(k+1) \geq d_E(k) \\ 0, & \text{otherwise} \end{cases} \tag{2.8}$$

The second derivative, $\ddot{d_E}$, is calculated in the same way by replacing d_E with $\dot{d_E}$. The negative values are set to 0 because they contain redundant information, as the locations of the cuts will always be positive peaks in the distance vector.

$$\ddot{d_E}(k+1) = \begin{cases} \dot{d_E}(k+1) - \dot{d_E}(k), & \text{if } \dot{d_E}(k+1) \geq \dot{d_E}(k) \\ 0, & \text{otherwise} \end{cases} \tag{2.9}$$

Now we have a vector with more distinguishable cut locations.

2.2.4 Automatic Threshold Computation

To calculate a proper threshold for different films, an automatic thresholding method is used. The first step is to calculate the average second derivative of the distance value from the $\ddot{d_E}$ vector.

$$ave_{\ddot{d_E}} = \frac{1}{N_k} \sum_{k=0}^{N_k-1} \ddot{d_E}(k) \tag{2.10}$$

where N_k is the size of $\ddot{d_E}$, which depends on the number of frames in the film.

This average value is then used in the definition of local maximum to locate the peaks of the distance vector $\ddot{d_E}$. A local maximum is defined as a point that meets the following conditions:

$$\ddot{d_E}(k)\text{is a local maximum if} \begin{cases} \ddot{d_E}(k) > ave_{\ddot{d_E}}, \text{ and} \\ \ddot{d_E}(k-1) < \ddot{d_E}(k), \text{ and} \\ \ddot{d_E}(k+1) < \ddot{d_E}(k) \end{cases} \tag{2.11}$$

The final threshold value, τ_{cut}, is obtained by calculating the average value of all the local maximum peaks and a cut at frame k is detected if the following conditions are met:

$$\text{cut detected if} \begin{cases} \ddot{d_E}(k-1) < \tau_{cut}, \text{ and} \\ \ddot{d_E}(k) > \tau_{cut}, \text{ and} \\ \ddot{d_E}(k+1) < \tau_{cut} \end{cases} \tag{2.12}$$

2.3 Conclusion

This chapter presented several shot detection methods from the literature. These included edge-based methods, motion-based methods, methods that use computed features of compressed video, and histogram-based methods.

The chosen shot detection method and its various procedures were also described. Ionescu's shot detection method can be broken down into several different stages. First, color reduction is applied to each of the video's individual frames using the "Webmaster" palette, which has a total of 216 colors. The $4-$Histograms method is then used to calculate the distance between each frame and to obtain the distance vector for the entire film. Once this vector is acquired, the 2^{nd} Derivative method is applied to the vector in order to more clearly distinguish the cut locations. An automatic threshold is then computed and applied to the new vector to locate the shot boundaries. A selection of shots detected using this method can be seen in Appendix A. The three films used throughout to evaluate the performance of the selected methods were chosen, due to the origins of this project, because of their religious subject matter.

Once the shot detection method is effectuated and the shots are obtained, the next step is to determine appropriate keyframe extraction methods and shot similarity measures which can be used in the scene detection process.

Chapter 3

Keyframe Extraction and Similarity Measures

Scene detection is generally a more difficult task when compared to shot detection. Shots can be defined as a sequence of frames continuously captured from a single camera whereas scenes are a series of shots that are related semantically and contain a common background, action or story. After the shot boundaries of a video have been detected, using the method described in the last chapter, the next step towards scene detection is to calculate a certain similarity measure which can then be used to cluster shots into scenes. The following examines previously employed keyframe and shot similarity measures found in the literature which were used in scene detection. This chapter is organized into four main sections: keyframe extraction, frame similarity, shot similarity measures, and the last section, experimental comparisons, will compare selected similarity measures and keyframe extraction algorithms and select one of each for use in scene detection. They will be presented in that order.

3.1 Keyframe Extraction

Keyframe extraction is performed on shots in order to save computation time by avoiding the effort of using all the frames in the video. The keyframes selected should be those frames that represent the most important content of the shot. Several different methods have been employed throughout the literature. Some authors use the middle frames of the shots as the representative frames [17][18] and others choose to select the first and last frames [8]. Another simple method used is to sub-sample the shot at a fixed rate to extract several keyframes [19][20]. The other methods that will be presented, extract the selected frames that satisfy a chosen criterion.

In [21], Detyniecki and Marsala select one keyframe from the beginning and another one from the end of the shot. To find the frames in those areas, Euclidean distance between the frame histograms are computed. Those that are significantly different from the others are selected as either the *beginning frame* or as the *ending frame*.

Le *et al.* [12] employ a *curve of cumulative frame difference* by calculating the cosine distance between successive frames. Keyframes are chosen as the frames that correspond to the midpoints between consecutive high points on the curve [21].

In [23], the first step to obtaining a set of keyframes is to select the middle frame and add it to an empty set. Next, the other frames are compared to those in the set. If the current frame's similarity with those in the set, which is computed as color histogram intersection, is below a certain threshold then it is added to the set. This is repeated until all of the shot's frames have been examined, thus obtaining a set of different keyframes to represent the shot content. Similar methods are also employed in [24][25][26].

The first step in the method presented in [13] is to compute what they refer to as the *stability* of the frames in a shot. It is calculated as follows:

$$S(f_i) = 1 - \frac{(VisSim(f_{i-1}, f_i) + VisSim(f_i, f_{i+1}))}{2} \tag{3.1}$$

where $VisSim(x, y)$ is the visual similarity between frames x and y. Once this has been calculated, the frames with local maximum *stability* are selected as candidate keyframes. If the difference between candidate keyframes is less than a threshold, the middle frame between them is chosen as the new candidate in their place. This process is repeated until no keyframes can be replaced. The next step is to divide the frames between two successive keyframe candidates into two classes. The partition position p is selected as:

$$p = argmin \left(\sum_{n=i+1}^{p} VisSim(k_i, f_n) + \sum_{n=p+1}^{j-1} VisSim(f_n, k_j) \right) \tag{3.2}$$

where f_n is a frame between keyframes k_i and k_j. Next, the new candidate keyframes are selected as the frames that are most similar to their clustering center. The final keyframes of the shot are obtained after a few iterations.

In [27], Sano *et al.* use motion information to first divide shots into two kinds of sub-shots, those that are considered *active* and those that are *non-active*. This is done by dividing the frames into 3×3 blocks in which the motion vectors are summed. Frames are deemed *active* if the sum exceeds a certain threshold. The authors claim that by selecting representative images, keyframes, from the *non-active* sub-shots, the robustness of the keyframe comparison method is improved. Toharia *et al.* [28] employ a similar method using the activity of the shot to help determine which keyframes to select. *Activity* is measured for all of the frames in the shot and the middle frame between frames that have *activity* higher than a certain threshold, or between these frames and the shot boundaries, are selected as the representative keyframes for the shot.

Ngo *et al.* [29] also use motion, which is obtained by analysis of spatio-temporal image volumes [30], in their keyframe selection algorithm. The selection of keyframes varies depending on whether the motion type is static, a pan or tilt, a zoom, or if it is viewed as tracking an object. For static shots one frame is used, for zooms the first and last frames are used. However, if the shot contains a pan or tilt, a panoramic image is obtained by warping the images together to use for comparison. If it is object tracking, the object is used to compare shots.

In [11], Chasanis *et al.* attempt to obtain unique keyframes from a shot by clustering the frames using a common spectral clustering method in which the number of clusters is determined automatically. Once the clusters are obtained, the keyframes are selected as the *medoids* of the clusters, which are defined as being the frames that have the highest similarity to all the other frames in the cluster.

Once the representative keyframes of each shot are extracted, these can be used to estimate the similarity between shots. This is discussed in the next sections.

3.2 Frame Similarity Measures

One of the most commonly used techniques to determine similarity between shots is to perform a pair-wise comparison between color histograms of keyframes extracted from the shots. This can be viewed as a frame similarity measure. In most methods, the first step of this process is to perform a color reduction, also referred to as color quantization, on the original frames. This step can greatly reduce the computational time for the steps that follow it. Next, color histograms are obtained for each of the keyframes and a visual similarity between keyframes can be calculated using several methods.

The two most commonly used methods to obtain visual similarity are Euclidean distance $VisSim_{Eucl}(x, y)$ and histogram intersection $VisSim_{int}(x, y)$. In the latter method, the degree of similarity is proportional to the region of intersection.

$$VisSim_{Eucl}(x, y) = \sqrt{\sum_{h \in bins} (H_x(h) - H_y(h))^2} \tag{3.3}$$

$$VisSim_{int}(x, y) = \sum_{h \in bins} min(H_x(h), H_y(h)) \tag{3.4}$$

where, $H_x(h)$ is the color histogram of frame x at bin h.

In [31], Sasongko, Rohr, and Tjondronegoro calculate the visual similarity between the keyframe HSV histograms of the two shots using the chi-squared test.

$$VisSim_{\chi^2}(x, y) = \sum_{h \in bins} \frac{(H_x(h) - H_y(h))^2}{H_x(h) + H_y(h)} \tag{3.5}$$

The authors also note that placing more importance on the hue has been found to be effective so they put more emphasis on the hue of each keyframe by weighting each channel as follows: 4 for hue, 2 for saturation, and unity for value.

Zhai and Shah [23] compute the visual similarity between keyframes as the Bhattacharya distance between RGB color histograms, which have 8 bins for each of the red, the green, and the blue channels.

$$VisSim_{Bhatt}(x,y) = -\ln\left(\sum_{h\in bins}\sqrt{H_x(h)H_y(h)}\right) \tag{3.6}$$

Ren *et al.* [32] decided to divide the keyframes into 9 regions, a 3x3 grid. A weight of 2 is assigned to the four corner and center regions and unity weight for the four remaining regions. A weighted sum of the intersection distances of each region is used as the similarity measure. Noguchi and Yanai [33] also divide the center frame of the shot, which they use as the keyframe, into 3x3 regions before computing RGB histograms and calculating intersection distance.

Another frame similarity measure that is employed to characterize the spatial distribution of color in a frame is the MPEG7 color layout descriptor. It is extracted from an image in four steps. First the image is divided into 8x8 blocks, then dominant colors are selected and represented in YCbCr color space. Next, an 8x8 discrete cosine transform (DCT) is applied to the Y, Cb, and Cr components to obtain three sets of DCT coefficients. Lastly, these coefficients are zigzag scanned to obtain the first few low-frequency coefficients which are then quantized to form the descriptor [34]. In [35], Naci *et al.* calculate the similarity between frames using a Gaussian function, the Euclidean distance between color layout descriptors (CLD), and a scaling parameter ϵ. It is computed as

$$VisSim_{cld}(x,y) = e^{-\frac{||CLD_x - CLD_y||_2}{\epsilon^2}} \tag{3.7}$$

where CLD_x is the color layout descriptor of frame x. A similarity matrix is created once similarities between all frames have been computed.

The authors of [36] use the same method, but take advantage of the similarity matrix to compute the minimum distance between the keyframes of the shots being compared as the final similarity measure. Color layout descriptor combined with edge histogram descriptor is used in [19].

The methods in [21], [27], [37], [38], and [39] use variations of the similarity metrics described above. They mostly use color histograms in HSV space, segments of a predefined length or keyframes to represent shots, and histogram comparison.

The next section discusses how shot similarity can be evaluated from the presented frame similarity measures.

3.3 Shot Similarity Measures

After keyframes are selected, the shot similarity measure used in [25] by Zhao *et al.*, can be obtained by calculating a pair-wise visual similarity between the keyframes of two shots. The largest obtained keyframe similarity is selected as the representative one between the shots. This can be done for each pair of shots, depending on the requirements of the method.

$$ShotSim(s_i, s_j) = max\left\{VisSim(k_m^i, k_n^j)\right\} \tag{3.8}$$

where k_m^i is the m^{th} keyframe of shot i.
In their case, they used $VisSim_{int}$, defined in eq.(3.4), as the frame similarity measure which would yield $ShotSim_{int}$ as the shot similarity measure.

Before direct use of this measure in the grouping process, the temporal distance of each shot is also weighted into the similarity measure. The middle frame numbers f_{mid} of the shots being compared, the standard deviation of shot durations σ_{sd} throughout the video and, in this specific case, the number of shots N_s, are added to the equation.

$$W(s_i, s_j) = exp\left(-\frac{(f_{mid}^i - f_{mid}^j)^2}{\sqrt{N_s}\sigma_{sd}^2}\right) \times ShotSim(s_i, s_j) \tag{3.9}$$

Ngo *et al.* [29] employed an almost identical measure, but chose to weight their similarity measure using the total number of frames, N_f, instead of shots, along with an experimentally obtained temporal parameter γ.

$$w(s_i, s_j) = exp\left\{\frac{-\gamma \times |f^j - f^i|}{N_f} \times ShotSim(s_i, s_j)\right\} \tag{3.10}$$

where $|f^j - f^i|$ is the temporal distance between shots i and j.

The final similarity measure used in [23] is computed as the maximum of a constant $\mathbb{C}$ subtracted by the visual similarity, defined in eq.(3.6), between the keyframes of the two shots being compared.

$$Sim_{Bhatt}(s_i, s_j) = \max(\mathbb{C} - VisSim_{Bhatt}(k_m^i, k_n^j)) \tag{3.11}$$

Lu *et al.* [17] suggested the use of a spatio-temporal dissimilarity function that can use any shot color histogram similarity measure. They also decided to weight this function with the temporal distance between the middle frames of the two shots.

$$Dis(s_i, s_j) = 1 - ShotSim(s_i, s_j) \times e^{-\zeta \times d_T(s_i, s_j)} \tag{3.12}$$

where ζ controls the slope of the exponential and d_T is the distance between middle frames of shots i and j.

Le *et al.* [12] extract three different features from keyframe color histograms to characterize the shot content. Keyframes are divided into regions and the mean μ_b, the variance σ_b, and the skew sk_b of the histograms are calculated as follows:

$$\mu_b = \frac{1}{N_h} \sum_{i=1}^{N_h} H_b[i] \tag{3.13}$$

$$\sigma_b = \frac{1}{N_h} \sum_{i=1}^{N_h} (H_b[i] - \mu_b)^2 \tag{3.14}$$

$$sk_b = \frac{1}{N_h} \sum_{i=1}^{N_h} (H_b[i] - \mu_b)^3 \tag{3.15}$$

where, $H_b[i]$ is the histogram of b^{th} region of the keyframe, i is the bin number, and N_h is the number of histograms.

In [40], Ren and Jiang employ a similarity measure that includes luminance histogram correlation, $c_h(i, j)$, and DC-image difference. The histograms and the DC-images are computed in the YCbCr color space. The Y, Cb, and Cr components of an image are extracted as $Y_{DC}^{(i)}$, $U_{DC}^{(i)}$, and $V_{DC}^{(i)}$ respectively, where i is the frame number. The luminance components are used to compute image difference $c_d(i, j)$.

$$c_d(i,j) = 1 - \frac{\sum_{m,n} |Y_{DC}^{(i)}(m,n) - Y_{DC}^{(j)}(m,n)|}{min\left[\sum_{m,n} Y_{DC}^{(i)}(m,n), \sum_{m,n} Y_{DC}^{(j)}(m,n)\right]} \tag{3.16}$$

$$c_h(i,j) = \frac{\sum_m h_i(m)h_j(m)}{\sum_m [h_i(m)]^2} \tag{3.17}$$

where h_i is the luminance histogram of DC-image $Y_{DC}^{(i)}$. The overall similarity metric is computed as a weighted sum of the two.

Other papers chose to incorporate several other features as well as color histograms into their similarity measures. Motion, edge information, and color layout descriptor are among the most common.

In [41], a gradient distribution descriptor is added to the similarity measure. It is calculated from histograms of the magnitude of the image intensity gradient. Shape and color similarity measures are used in [28]. The authors use Zernike invariants [42] as the shape descriptors and quantified histograms as the color feature. In [43], the authors employ object tracking and a shot color invariant feature, obtained from a color ratio gradient which is insensitive to object positions, shadows and illuminations, to represent the contents of a shot. The authors of [44] extract a 30-dimensional feature vector from the shots. It is composed of 1^{st}, 2^{nd}, and 3^{rd} order moments of each R, G, and B color histograms, along with the mean and variance values of three high-pass sub-bands for each of the R, G, and B planes, and also 1^{st}, 2^{nd}, 3^{rd} order moments of what the authors refer to as the motion magnitude histogram.

3.3.1 Dominant Color

Lin and Zhang [45] chose to use dominant color histograms to obtain a correlation measure between two shots. It is calculated as the intersection of HSV dominant color histograms. Their method for the extraction of dominant colors is detailed in the following.

The first step is to obtain the color histograms for all of the frames in the video. 10 bins are allocated for hue, 5 for saturation, and 5 for lightness (value), together they form a 3D color histogram. The next step is to identify the local maximum points and surround them with spheres of 3 units in diameter, creating color objects. The first few objects with the most pixels are the dominant objects. Using the pixels from these objects, a new 3D dominant color histogram is then obtained.

Once this is performed for all of the frames in a shot, the distances between dominant objects in adjacent frames is calculated. If this distance is smaller than a threshold, then the objects are considered to be the same. This is done for all the frames in a shot. Finally, an overall dominant color histogram is formed by using the dominant objects with the longest durations within the shot and weighting them by their duration.

The same authors along with Q.Y. Shi, expand on the methods by adding what they call *spatial structure histograms* in [46]. K-means clustering is applied to perform color quantization in order to obtain *color-blob* maps. From these maps various distribution features are extracted. Each is used to describe certain aspects of the image: 8 bin area histograms for spatial complexity, 16 bin position histograms for spatial configuration, 8 bin deviation histograms in the X and Y direction and 8 bin span histograms for the shape distribution. These measures are combined into a spatial similarity measure, $SshSim$, which is then lastly added to the measure obtained from the dominant color histograms, $DchSim$, in order to compute the final similarity measure.

$$Sim_{dcss}(s_i, s_j) = DchSim(s_i, s_j) + SshSim(s_i, s_j) \tag{3.18}$$

3.3.2 Motion

In [26], Sakarya and Telatar combine visual similarity and motion similarity into their metric. They argue that during dialogue scenes, which may span several shots, the motion content is generally low and, on the other hand, action scenes will have high motion from actor or camera movement. Therefore, they conclude that it is useful to combine the two features. The visual similarity between frames within a shot is first computed as the intersection distance between normalized 16 bin HSV color histograms, which have 8 bins for hue, 4 for saturation, and 4 for value. The motion content of the video, Mot, is then calculated using the previously calculated visual similarity feature, eq.(3.4), between frames.

$$Mot = \frac{1}{b-a}\sum_{f=a}^{b-1}(1 - VisSim_{int}(f, f+1)) \tag{3.19}$$

where a and b are the indices of the first and last frames of the shot.
The visual similarity between shots is then computed as the maximum visual similarity of all possible pairs of their keyframes and the motion similarity is computed as:

$$MotSim(s_i, s_j) = \frac{2 \times min(Mot_i, Mot_j)}{Mot_i + Mot_j} \tag{3.20}$$

where Mot_i is the motion content of shot i.
The final measure is a weighted sum of the two similarities and in this particular paper, they are both equally weighted by a factor of 0.5. It is also weighted by the temporal distance between the middle frames of each shot by multiplying it by

$$e^{\left(-\frac{1}{d}\cdot|\frac{f^i_{mid}-f^j_{mid}}{\sigma_{sd}}|^2\right)} \tag{3.21}$$

where f^i_{mid} is the middle frame of shot i, and d was manually selected to be 20.

Wang *et al.* [8] use visual and motion similarity measures almost identical to the ones from [26]. The difference is that they normalize each measure and weight the final shot similarity using their mean and standard deviations. For visual similarity, they are calculated as follows:

$$\mu_v = \frac{1}{\sum_{j=1}^{F}(N_s - j)}\sum_{j=1}^{F}\sum_{i=0}^{N_s-j-1} VisSim_{int}(s_i, s_{i+j}) \tag{3.22}$$

$$\sigma_v = \sqrt{\frac{1}{\sum_{j=1}^{F}(N_s - j)}\sum_{j=1}^{F}\sum_{i=0}^{N_s-j-1} (VisSim_{int}(s_i, s_{i+j}) - \mu_v)^2} \tag{3.23}$$

where F is the forward search range and N_s is the number of shots.

The mean and standard deviation of motion similarity, μ_m and σ_m respectively, are calculated in a similar manner. The weights are then calculated using these values.

$$W_v = \frac{\sigma_v}{\sigma_v + \sigma_m} \tag{3.24}$$

$$W_m = \frac{\sigma_m}{\sigma_v + \sigma_m} \tag{3.25}$$

The final shot similarity metric is then calculated as the weighted sum of the normalized visual and motion similarity measures.

$$\begin{aligned} ShotSim_{vm}(s_i, s_j) = & \frac{\sigma_v}{\sigma_v + \sigma_m} \times \frac{VisSim(s_i, s_j) - \mu_v}{\sigma_v} \\ & + \frac{\sigma_m}{\sigma_v + \sigma_m} \times \frac{MotSim(s_i, s_j) - \mu_m}{\sigma_m} \end{aligned} \tag{3.26}$$

Motion is also employed in the visual distance measure of [18]. A weighted summation of RGB color histogram difference, HSV color histogram difference, and motion compensated matching error (motion estimations) are used. Optical flow is another form of motion that has been employed in determining shot similarity [47]. The authors use the Lucas-Kanade method for point-based tracking to estimate the optical flow.

In [20], Chen *et al.* first segment their shots into those that contain camera motion, referred to as dynamic shots, and those without, static shots. They sample the dynamic shots by taking one frame for every ten and use a mean block histogram to represent the static ones. YUV histograms of each of the sampled frames from the dynamic shots are computed. Once this is done, normalized cosine is used as the distance measure between shots. The final similarity measure is calculated as the average of, at most, the five best similarities between the shots.

3.3.3 Edges

Another feature that has been observed in the literature is edge information. Chen *et al.* [48] use the distribution density of edges in a frame to determine its texture. The first step to obtaining their shot similarity measure is to create static background mosaics for each shot [49][50]. Once this is accomplished, several features are computed from HSV color histograms. First, each histogram bin with non-zero count is deemed a color object. Next, density of distribution λ_{i1}, compact of distribution λ_{i2}, scatter λ_{i3}, and the number of active blocks in the image frame λ_{i4} are calculated for each background. An average of a feature is used if a shot contains more than one background. The difference of spatial distribution within a shot, $D_s(c_i, c_j)$ is then calculated as the sum of the Euclidean distances between each feature, where c_i and c_j are color objects. Edges of an image are obtained by using the Canny edge detector. The image is then divided into 16x16 regions and a region is deemed textured if it contains more edge points than a set threshold. The ratio of textured blocks is then calculated for each image and averaged for the shot. The texture similarity between two shots is the minimum value of the two. Once all these measures have been calculated, the final shot similarity is calculated as:

$$ShotSim_{\lambda}(s_i, s_j) = \frac{1}{K} \sum_{(u,v) \in \Omega} Sig(D_s(u,v)) \times min(\bar{H}_i(u), \bar{H}_j(v)) + w_t \times min(t_i, t_j) \quad (3.27)$$

where

$$Sig(x) = \frac{1}{1 + e^{10 \times x - 5}}$$

is a form of the sigmoid function, and where Ω is the set of similar color objects from shots s_i and s_j whose Euclidean distance is below a certain threshold and where $\bar{H}_i$ is the average histogram of shot s_i, K is the image size, w_t is the weight of the texture feature and t is the texture value.

In [24], an edge is defined as a significant color change in the hue, lightness, or saturation histograms obtained from a sequence of shots. The authors, Truong *et al.*, use an edge detector to obtain the H, L, and S edge signals. They then combine them into a single signal before applying a threshold to extract edges. Lastly, a similarity measure that employs HLS color histograms is used to find the exact scene transition position within the temporal extension of detected edges.

3.3.4 Other Measures

Others who chose to employ support vector machines [51] or hidden Markov models [52] for scene detection, do not directly compute a similarity measure. They do, however, extract many features that they believe to be useful to describe and classify shots. Examples of audio features extracted include short-time zero crossing rate, bandwidth, mean of the spectral flux, silence ratio, harmonic ratio, high zero-crossing rate ratio, low shot-time energy ratio, volume standard deviation, volume dynamic range, volume undulation, non-silence ratio, standard deviation of zero crossing rate, standard deviation of pitch, non-pitch ratio, and frequency centroid. Color features can include color histograms, dominant colors, mean and standard deviation of colors. Motion and edge information is also used.

Dynamic programming was performed in two different papers to obtain a similarity between shots. In [11], Chasanis *et al.* perform a local sequence alignment of the keyframes within two shots using the Smith-Waterman algorithm. A visual similarity matrix is first computed using HSV color histogram difference, with 8, 4, and 4 bins for HSV respectively.

$$VisSim_{algn}(f_i, f_j) = 1 - \sum_{h=1}^{128} \frac{(H_i(h) - H_j(h))^2}{H_i(h) + H_j(h)} \tag{3.28}$$

This matrix is then used in the alignment algorithm and the normalized resulting score is the final similarity measure.

Liu *et al.* in [13], perform a global alignment using the Needleman-Wunsch algorithm. Keyframes of two shots are aligned and it is determined whether they are matched, partly matched, or not matched at all. To determine if they are matched two scores which use HSV color histogram difference are calculated and compared.

In [53], Bredin *et al.* use principal component analysis (PCA) or linear discriminant analysis (LDA) to create a $footprint$ of a shot. D-dimensional color histograms H_j^i are obtained for every frame j within the shot i as well as an overall set of feature vectors $\mathcal{H}$.

$$\mathcal{H} = \bigcup_{i=1}^{N_s} \{H_j^i\}_{j \in [1, N_f^i]} \tag{3.29}$$

where N_s is the total number of shots and N_f^i is the number of frames in the shot i and H_j^i is the color histogram of frame j of shot i.

PCA is then apply to $\mathcal{H}$ in order to reduce the dimension and maximize the variance of the data set. Only the first two principal components are kept.

$$\mathcal{V} = PCA(\mathcal{H}) = \bigcup_{i=1}^{N_s} \left\{ v_j^i \right\}_{j \in [1, N_f^i]} \tag{3.30}$$

Here v_j^i is a 2-dimensional vector composed of the first two principal components.

LDA can also be applied to minimize the overall intra-class variance and to reduce the dimensions of the feature vector. Before this is done, the shot numbers are added to $\mathcal{H}$.

$$\bar{\mathcal{H}} = \bigcup_{i=1}^{N_s} \left\{ (H_j^i, i) \right\}_{j \in [1, N_f^i]} \tag{3.31}$$

LDA is then applied to $\bar{\mathcal{H}}$ and, as with PCA, the first two components are kept.

$$\bar{\mathcal{V}} = LDA(\bar{\mathcal{H}}) = \bigcup_{i=1}^{N_s} \left\{ \bar{v}_j^i \right\}_{j \in [1, N_f^i]} \tag{3.32}$$

where v_j^i and $\bar{v}_j^i$ are the resulting 2D vectors from PCA and LDA respectively.

Next, a 2D 30-bin histogram can be computed from the resulting sets $\mathcal{V}$ or $\bar{\mathcal{V}}$. The histogram can then be arranged into a binary *footprint* of the shot. The x and y components represent the first and second dimension of the 2D vectors v_j^i or $\bar{v}_j^i$ that form the sets previously mentioned. In a footprint, a bin is set to 0 (white) if it is empty or set to 1 (black) if it is not. These footprints can then be used as a similarity measure between shots.

3.4 Experimental Comparisons

Having presented various methods for keyframe extraction, frame and shot similarity, selected methods will be chosen for comparison and review. The ground truth, along with several other measures, will be employed to establish an appropriate threshold for each method and, subsequently, to select the similarity measure and keyframe extraction method which perform best.

3.4.1 Frame Similarity Measures

Because most keyframe extraction methods require the use of a frame similarity measure, the first step will be to compare some frame similarity measures.

As seen in section 3.3, these measures perform pair-wise comparisons of the keyframes extracted from the shots. Intersection, eq.(3.4), Euclidean Distance, eq.(3.3), and Bhattacharya Distance, eq.(3.6) were chosen for comparison. These measures were chosen because Euclidean distance is a typical distance measure, Intersection is commonly used when employing histograms, and Bhattacharya distance measures the similarity between two probability distributions, which in our case are the color histograms since they represent the distribution of color in an image.

Comparison Measures

The creation of a ground truth is essential to the process of comparing different measures. A ground truth of four segments from the movie *The Shawshank Redemption* has been constructed. For all four segments, each shot within the segment has been manually compared with each of the others within the same segment. If two shots are judged to be similar, the similarity point on the map is assigned a value of 1, shown as a black point on the map; otherwise it is assigned a value of zero, shown as a white point on the map. Using these values, similarity maps, or similarity matrices, of all four segments were created and combined into a single map. The combined segments, each individually outlined, can be seen in Figure 3.1. The segment boundaries on the ground truth similarity map are bounded by $(1, 35)$, $(36, 43)$, $(44, 59)$, and $(60, 105)$. They are composed of 8157, 410, 1358, and 7647 frames respectively. Shots from one segment were not compared to the shots of another and as such when calculating the comparison measures, the areas outside the segment boundaries were also ignored.

The first segment is composed of a scene in the movie where the main character Andy is accused of murder and is sent to jail. It occurs in a court room with some flashback shots to other areas. Examining Figure 3.2, it was determined that shots 6 and 7 are not similar and that shots 6 and 8 are similar.

The second segment is where another character, Red, is being evaluated by a group for parole. Figure 3.3 shows that shots 36 and 37 are not similar, shots 36 and 38 are similar, and that shot 43 is not similar to either 36, 37, or 38.

In the third segment, Andy enters a bank and has dealings with the bankers. In this segment, we have an example of where cuts have been

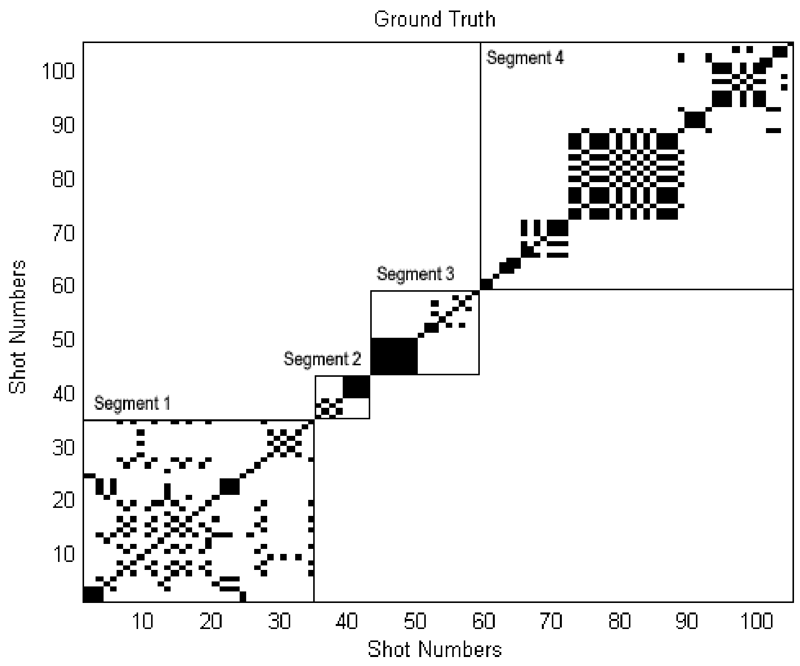


Figure 3.1: Manually created ground truth for the four segments of the movie. A black dot appears when the shot on the x-axis is similar to the shot on the y-axis

falsely detected because of fast motion of a large object in the middle of the frame. However, the resulting "sub-shots" should still be deemed to be similar as we can see with shots 44 and 45. Figure 3.4 shows that shots 44 and 45 are similar, and shots 44 and 51 are not similar.

In the last segment Red is searching for something outdoors. This is a long outdoor segment, that includes an area with roads, one in a field, and another in the woods. Figure 3.5 shows as sample of frames taken from the segment. Shots 59 and 60 are similar, another frame taken from shot 60 shows that shots 60 and 61 are not similar, shots 65 and 66 are similar, and shots 66 and 68 are not similar.

An NxN similarity matrix is computed for each of the four segments using one of the three frame similarity methods, Intersection, Euclidean Dis-

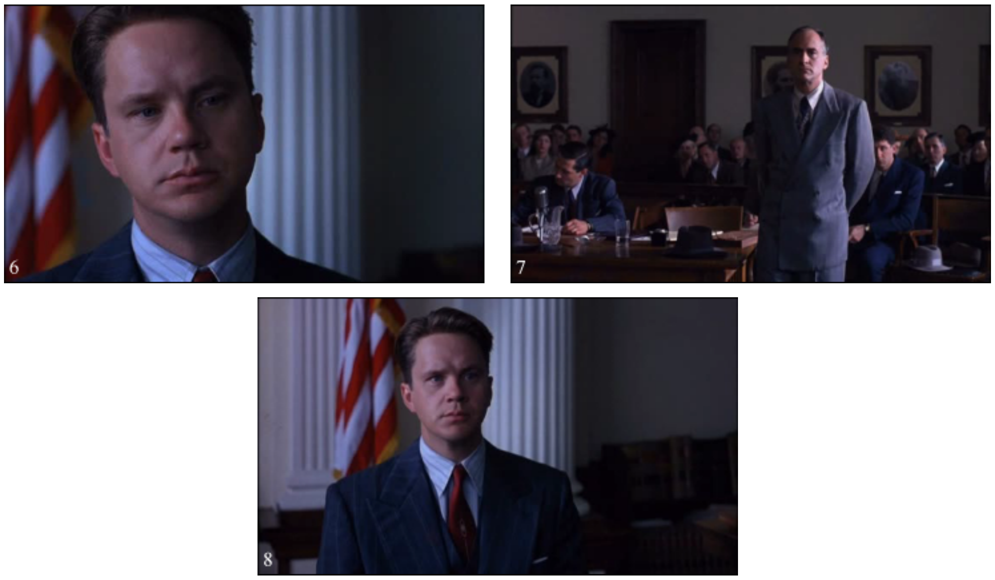


Figure 3.2: Frames from shots 6, 7, and 8

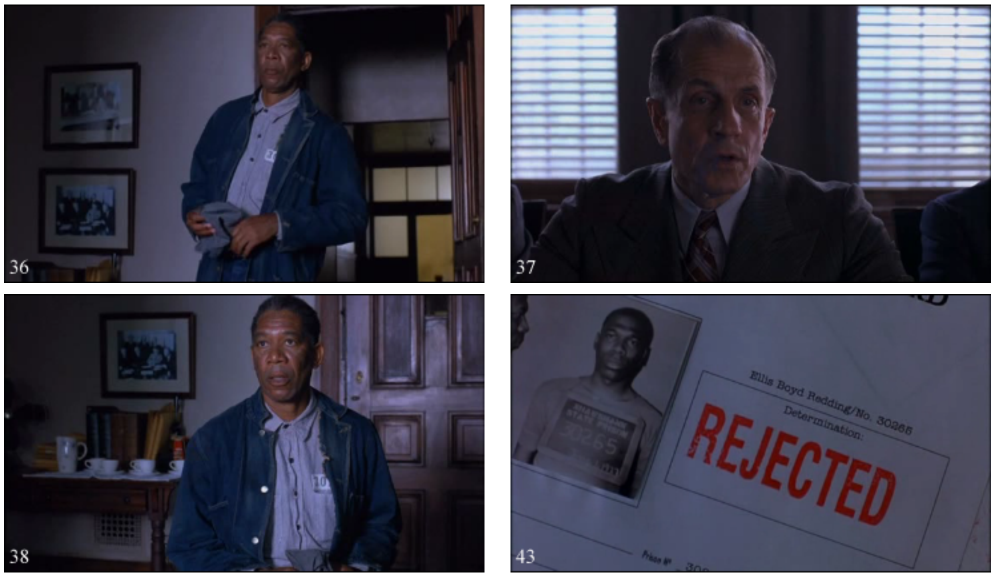


Figure 3.3: Frames from shots 36, 37, 38, 43

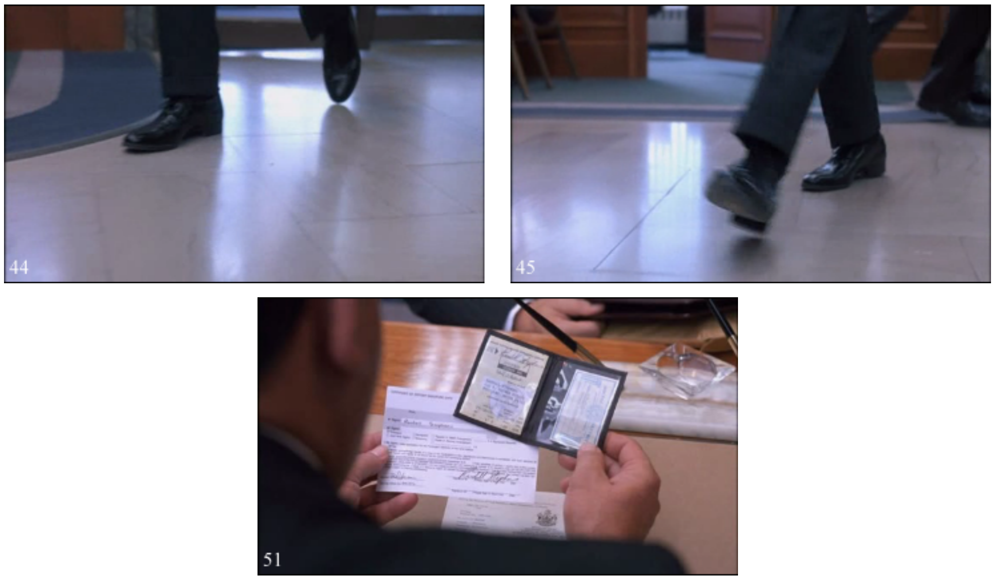


Figure 3.4: Frames from shots 44, 45, and 51

tance, and Bhattacharya Distance. Their resulting similarity matrices are aptly named *simMapInter*, *simMapEucl*, and *simMapBhatt*. The keyframe sets used are obtained by sampling the shots every 10 frames, since this method does not require the use of frame similarity to extract the keyframes (keyframe extraction methods are evaluated in the next section). Frame similarity is rather used to construct the shot similarity matrix according to equation 3.8. The matrices are then normalized such that 1 is the most similar and 0 is not similar at all.

They are normalized as follows:

For Intersection, we simply divide by the maximum value.

$$simMapInter_n = \frac{simMapInter}{max(simMapInter)} \tag{3.33}$$

For Euclidean distance usually a distance of 0 indicates the highest similarity, so we first divide by the maximum value then invert the results.

$$simMapEucl_n = 1 - \frac{simMapEucl}{max(simMapEucl)} \tag{3.34}$$

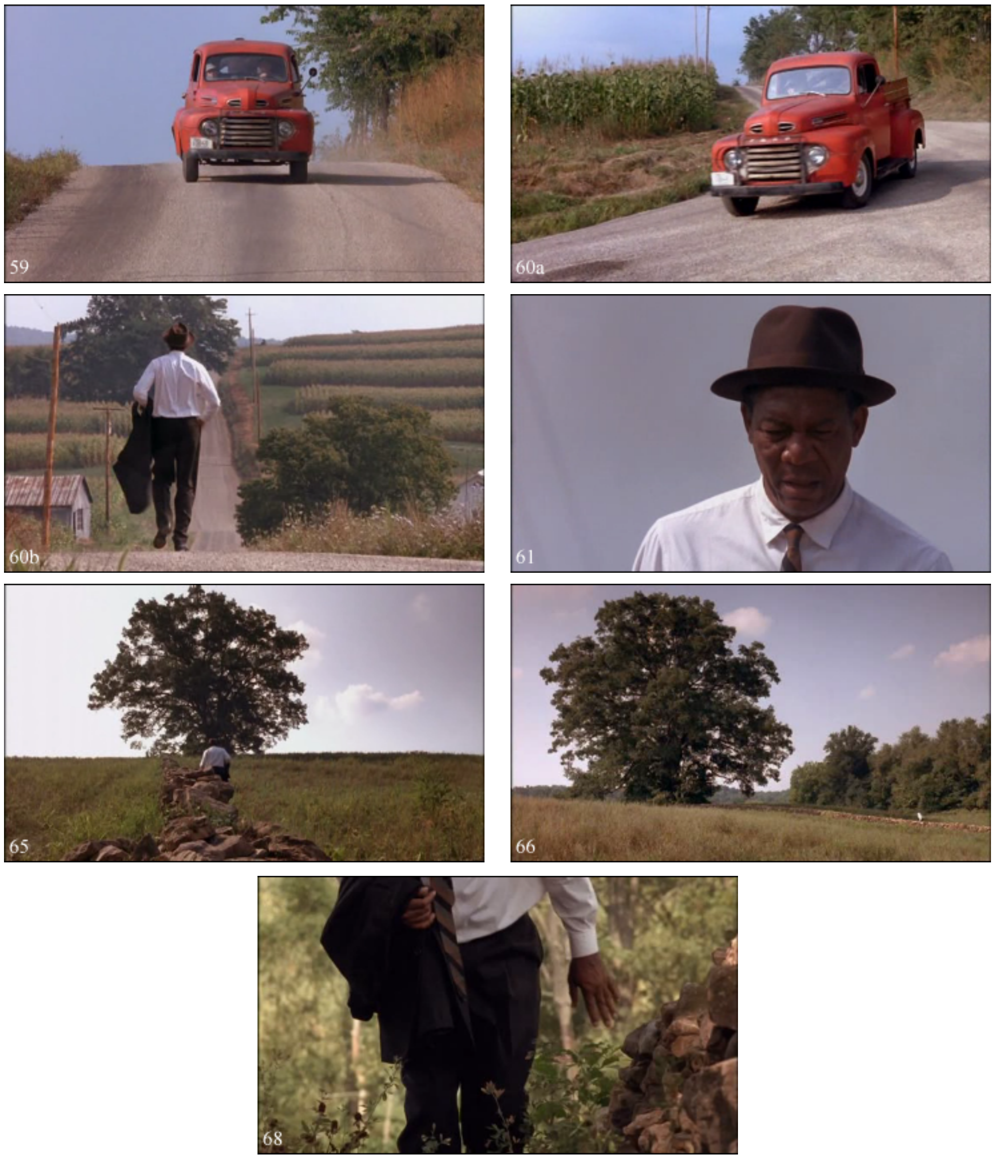


Figure 3.5: Frames from shots 59, 60(a), 60(b), 61, 65, 66, and 68

For Bhattacharya distance the first step is to make the smallest distance 0, then divide by the maximum value.

$$simMapBhatt_{temp} = 1 - \frac{simMapBhatt}{max(simMapBhatt)} \tag{3.35}$$

$$simMapBhatt_{n} = \frac{simMapBhatt_{temp}}{max(simMapBhatt_{temp})} \tag{3.36}$$

These normalized matrices are then thresholded and the resulting matrices are compared with the ground truth to obtain true positive (TP), false positive (FP), true negative (TN), and false negative (FN) values. This is done for multiple thresholds, with the threshold varying from 0 to 1.

These values are then used to compute several measures that can be used to compare and evaluate the results. These measures include *Precision*, *Recall*, *F-Measure*, F_2-*Measure* [54], and the Normalized Detection Cost Rate ($NDCR$) [55].

Precision can be defined as the percentage of true detections with respect to the overall declared event. It is calculated as:

$$Precision = \frac{TP}{TP + FP} \tag{3.37}$$

Recall can be defined as the percentage of true detections with respect to the overall events actually present in the sequence. It is calculated as:

$$Recall = \frac{TP}{TP + FN} \tag{3.38}$$

F-Measure can be used to combine the two previous measures and is defined as the harmonic mean of *Precision* and *Recall*. It is a measure of accuracy that considers both *Precision* and *Recall*, it is a weighted combination of the two. In this form, it is equally balanced between both and is calculated as:

$$F\text{-}Measure = \frac{2(Precision \times Recall)}{Precision + Recall} \tag{3.39}$$

There is also the F_2-*Measure*, which weights *Recall* twice as much as *Precision*.

$$F_2\text{-}Measure = \frac{5 \times (Precision \times Recall)}{4 \times Precision + Recall} \tag{3.40}$$

In order to select a proper threshold, two error rates, the probability of a miss (P_{miss}) and the false alarm rate (R_{fa}), are combined into a single detection cost rate. A cost is then associated to each error depending on the type of problem (C_{miss}, C_{fa}). Here, misses indicate that two shots were not deemed to be similar, when in fact they should have been, and false alarms are the opposite. In our case, a higher cost for misses has been selected because we would prefer to obtain more similarities and not miss potentially important similarities.

P_{miss} is calculated as:

$$P_{miss} = \frac{FN}{N_{target}} \tag{3.41}$$

where N_{target} is the number of similar shots in the ground truth.

R_{fa} is calculated as:

$$R_{fa} = \frac{FP}{T_{queries}} \tag{3.42}$$

where $T_{queries}$ is the total number of shots being compared.

A Normalized Detection Cost Rate is then calculated using R_{target}, which has the effect of balancing the error types with respect to the priors, and the cost for each error. R_{target} is the a priori target rate and is estimated as the number of similar shots over the total number of shots in the ground truth. Choosing a cost for false alarms and for misses (C_{fa}, C_{miss}) the normalized detection cost rate is then calculated as:

$$NDCR = P_{miss} + Beta \times R_{fa} \tag{3.43}$$

where $Beta = \frac{C_{fa}}{C_{miss} \times R_{target}}$

The minimum $NDCR$ across all threshold values is selected as the decision threshold for its associated similarity measure.

One way of visualizing the capabilities and the effectiveness of the similarity measures is to plot a Detection Error Tradeoff curve (DET) [56]. Generally, the task of detecting similarities involves balancing failures to detect similarities and false similarities. The DET curve plots error rates on both axes, miss probability versus false alarm rate, using a logarithmic scale. This allows for a better viewing of the performance of each system as the area between each plot is increased and the tradeoff between the error types can be better observed. The DET curves will plot the resulting miss probability versus the false alarm rate for threshold values between 0 and 1 with a step size of 0.0001. The minimum NDCR points are also indicated on the curve for each system, one showing an equal cost between misses and false alarms (C_{miss}=1, C_{fa}=1, 1:1) and the other with the selected cost ratio (C_{miss}=10, C_{fa}=1, 10:1). A higher cost is associated to misses because we would prefer to have some extra false similarities and not miss potentially important ones. The optimum threshold value, the normalized threshold value, will be the one that results in the minimum NDCR point (10:1). It should also be noted that on these plots, straight lines correspond to normal likelihood distributions, and that the diagonal $y = -x$ represents random performance.

Results

When selecting the thresholds for shot similarity with each measure, the threshold at the minimum NDCR mark is used as a good initial value. Before choosing the cost for a false alarm and missed similarity, the problem must first be examined. In the case of detecting shot similarities in movies, we selected a slightly higher cost associated to misses than to false alarms because for this situation it should be less harmful if there are a few more similarities than if important similarities are missed. In other words, we want to be certain to find all similar shots, even with the possibility of a few false similarities.

The resulting DET curves and the associated *Precision*, *Recall*, *F-Measure*, and F_2-*Measure* values, obtained by comparing the results from each method to the four segment ground truth, can be seen in Figure 3.6 and Table 3.1. It should be noted that it has been observed that the minimum NDCR with both costs set to 1 also corresponds to the minimum $FP + FN$ point.

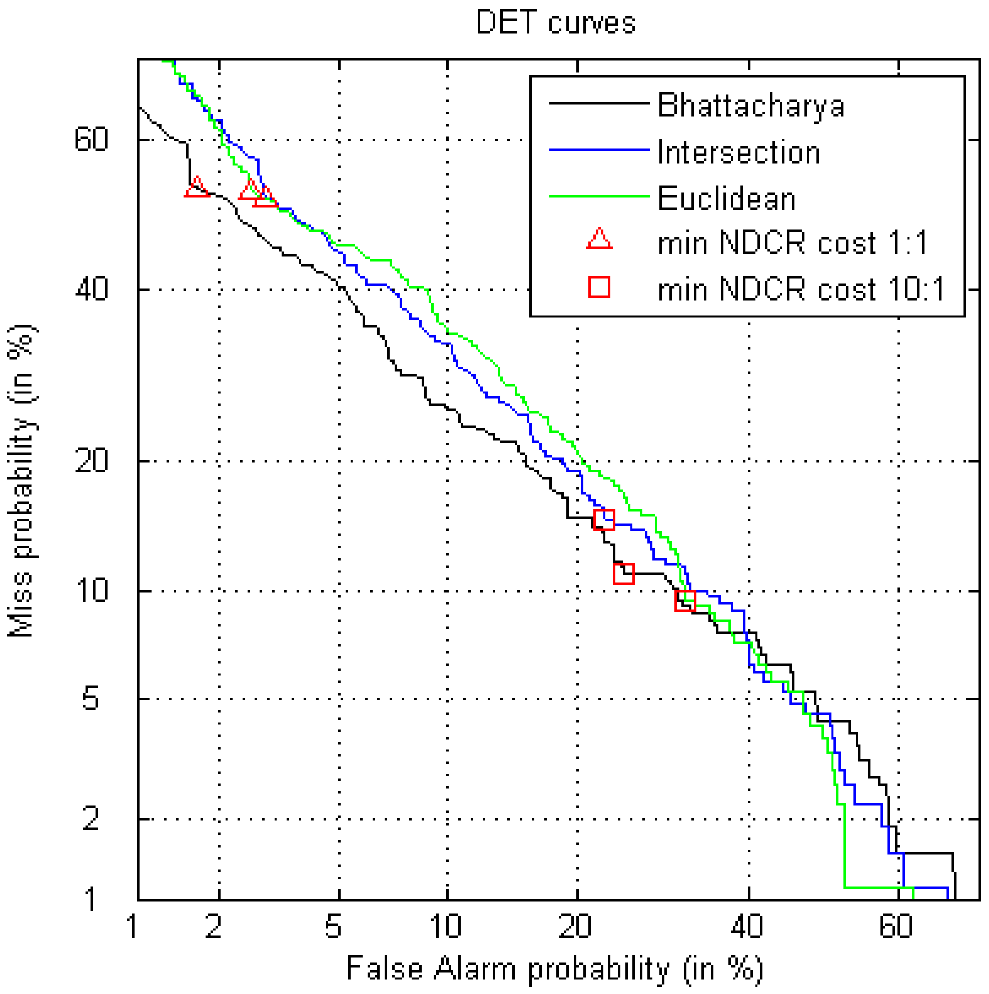


Figure 3.6: DET curves resulting from each frame similarity method being applied to the four selected movie segments

Table 3.1: Precision, Recall, F-Measures at each threshold (minNDCR 10:1) for the compared frame similarity measures

Method	Normalized Threshold Value	Precision	Recall	F-Measure	F_2-Measure
Intersection	0.8184	0.3535	0.8521	0.4997	0.6646
Euclidean	0.8452	0.2948	0.9057	0.4448	0.6403
Bhattacharya	0.9709	0.3436	0.8891	0.4956	0.6748

Decision

Comparing each similarity method using costs of 10 for a missed similarity and 1 for a false alarm, the minimum NDCRs are calculated for the three methods and the DET curves are plotted in Figure 3.6. The Normalized Threshold Values listed in the table correspond to the minimum NDCR values.

Examining the DET curves, one can see that the performance of these $VisSim$ methods do not differ by much. At the minimum NDCR value, with the selected cost of 10:1, the Euclidean Distance measure has the largest False Alarm probability (P_{fa}) of approximately 30%. It also, however, has the smallest Miss probability (P_{miss}), which should be expected, at around 9.5%.

The Intersection method resulted in a P_{fa} of about 23% with a P_{miss} of 14.9%. The Bhattacharya distance method gave a P_{fa} of around 25% and a P_{miss} of 11.2%. Taking into consideration the fact that we have weighed more heavily on the probability of a miss, after observing the DET curves and the results at the minimum NDCR the methods would have to be ranked in the order of: Bhattacharya Distance, Intersection, and Euclidean Distance.

It is difficult to decide on a measure when examining the similarity maps, because the methods perform so similarly, therefore the Precision and Recall values from Table 3.1 will also be examined. It can be seen that the Recall values are all fairly high, and looking at the F-$Measure$, which is a 1-to-1 balance between $Precision$ and $Recall$, it is difficult to select between

Table 3.2: Precision, Recall, F-Measures at each threshold for the compared keyframe extraction methods

Method	Normalized Threshold Value	Precision	Recall	F-Measure	F_2-Measure
Beginning and Ending Region Keyframes	0.9618	0.3520	0.8706	0.5013	0.6725
Middle Keyframes	0.9741	0.3132	0.7652	0.4444	0.5938
Sampled Keyframes	0.9709	0.3436	0.8891	0.4956	0.6748

Intersection and Bhattacharya but if *Recall* is given more weight, as we have also chosen to employ a higher cost for misses, and employ the $F_2\text{-}Measure$ then Bhattacharya Distance slightly edges out Intersection. For this reason, Bhattacharya Distance will be chosen as the frame similarity measure.

3.4.2 Keyframe Extraction Methods

Three keyframe selection methods from section 3.1 have been chosen for comparison. The first method involves simple sampling of frames across all shots every 10 frames (as we did in section 3.4.1), the second involves selecting one keyframe from the beginning and another from the end regions of the shot [21], and the third method begins with selecting the middle frame of the shot, adding it to a set, and recursively comparing the other frames of the shot with those in the set and adding them to it if they are different enough from those in the set [23].

The keyframe extraction methods are evaluated using the same measures, previously described in section 3.4.1, that were used to select a frame similarity method.

Results

The resulting DET curves and the associated *Precision*, *Recall*, *F-Measure*, $F_2\text{-}Measure$, and Normalized Threshold Value can be seen in Figure 3.7 and Table 3.2.

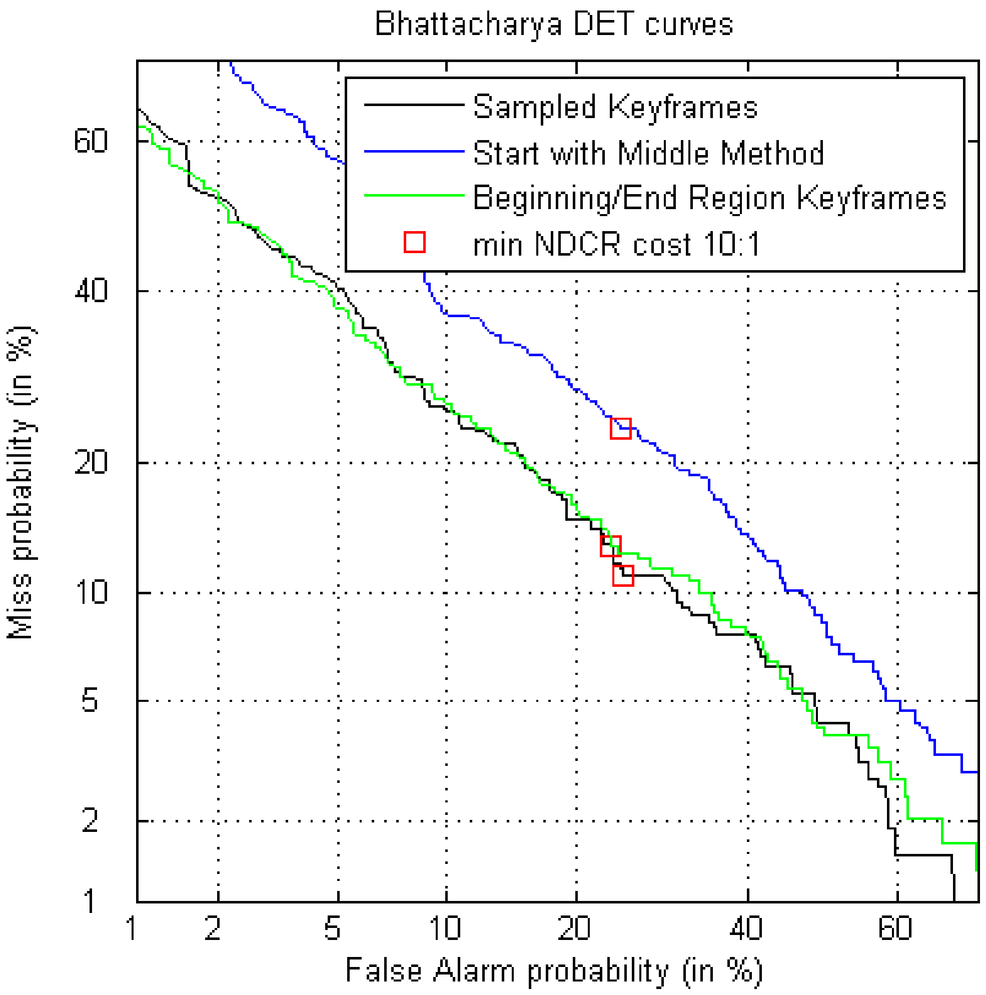


Figure 3.7: DET curves of each keyframe selection method

Table 3.3: Time to Compute

Method	Time to Select Keyframes (sec.)	Time to Compute Similarities (sec.)	Total Time (sec.)
Sampled Method	9.859186	342.147752	352.0069
Beginning and Ending Region Method	988.740170	6.482721	995.2229

Decision

Examining the DET curves shown in Figure 3.7, it is clear that the Middle keyframe selection method has the worst performance compared to the other two methods of sampling the frames and selecting a keyframe from the beginning and the end region of a shot. The main reason for its bad performance is most likely because the method, more often than not, only selects a single keyframe per shot and so does not properly represent the full content of the shot.

The other two methods perform similarly and it can be shown that the major difference is computation time. The Sampling method is quick to select keyframes but quite slow to compute shot similarity as there are more keyframes to examine, especially for longer shots. On the other hand, the Beginning and Ending region method takes longer to select keyframes but is much faster to compute shot similarity, having only 2 keyframes per shot to examine. If we examine the total time taken (see Table 3.3), the Sampled method is almost 3 times faster, therefore it will be chosen as the keyframe selection method.

3.5 Conclusion

This chapter presented various keyframe extraction methods and shot similarity measures used in scene detection within the literature. Keyframe extraction methods, frame similarity measures, shot similarity measures, including those that employ dominant color, motion or edges, were examined.

Intersection, Euclidean distance, and Bhattacharya distance were chosen for comparison to select the most appropriate similarity measure. The DET curves of each method is the first to be examined, and when taking into consideration that the probability of a missed similarity carries a greater cost, Bhattacharya distance was ranked first after this test. Next, *Precision*, *Recall*, *F-Measure*, and the F_2-*Measure* are evaluated for each of the three similarity measures. Intersection and Bhattacharya distance were very close when looking at the first three comparison measures but with the F_2-*Measure*, Bhattacharya distance slightly edges out Intersection as the similarity measure of choice.

Keyframe extraction methods were the next to be compared. Three methods were selected: frame sampling to obtain keyframe sets, selecting keyframes from the beginning and ending regions of shots, and the last method that first adds the middle frame of the shot to the keyframe set and recursively adds more depending if they are different enough from the current frames in the set. The resulting DET curves show the first two methods are very close in performance and eliminate the Middle keyframe method from the mix. In order to choose between the two remaining methods time to select keyframes and time to compute similarity were examined. The Sampled keyframe method finished with the highest rank by taking a third of the time to be executed.

The different frame similarity measures are relatively close in terms of effectiveness but Bhattacharya achieves an overall better performance. For shot similarity, the most important aspect is to select several keyframes. Simple sub-sampling performs as well as the more complex methods and is very efficient.

Having selected Bhattacharya distance as the similarity measure and the Sampled keyframe method to obtain keyframes, the next step is to employ these with an efficient scene detection method.

Chapter 4

Scene Detection

Once the similarity measure and keyframe extraction methods have been determined, the next step is to use them to group or cluster similar shots into scenes.

4.1 Previous Methods

From the literature examined, a graph-based shot clustering approach was among the most common methods. The authors of [17], [25], [26], [29], and [35] all used a slight variation of the same method. In [25], Zhao *et al.* create an undirected graph $G = (V, E)$ where the vertices V, the nodes of the graph, represent the shots, and the edges E have a weight $w(s_i, s_j)$ assigned to them which is the similarity value between the shots i and j. Once the graph is complete, it is divided into clusters, which are groups of shots, using the Normalized Cuts algorithm. The algorithm removes edges between clusters C_n and C_m by minimizing the $Ncut(C_n, C_m)$ value which is defined as:

$$Ncut(C_n, C_m) = \frac{cut(C_n, C_m)}{assoc(C_n, V)} + \frac{cut(C_n, C_m)}{assoc(C_m, V)} \tag{4.1}$$

where

$$cut(C_n, C_m) = \sum_{s_i \in C_n, s_j \in C_m} w(s_i, s_j) \tag{4.2}$$

$$assoc(C_n, V) = \sum_{s_i \in C_n, s_j \in V} w(s_i, s_j) \tag{4.3}$$

The authors approximate a discrete solution to the minimization of $Ncut(C_n, C_m)$ by solving the equation:

$$min_x Ncut(x) = min_y \frac{y^T(D-W)y}{y^T Dy} \tag{4.4}$$

where

$$d_i = \sum_j w(s_i, s_j)$$

$$D = diag(d_1, d_2, ..., d_n)$$

$$W(s_i, s_j) = w(s_i, s_j)$$

$$y = (1+x) - \frac{\sum_{x_i>0} d_i}{\sum x_i < 0 d_i}(1-x)$$

and x is an indicator vector where $x = 1$ if node i is in C_n or -1 otherwise. V is the set of all shots contained in C_n and C_m.

The algorithm is repeated on the clusters until the original graph is partitioned into M parts by $M-1$ operations. In this paper, the number of clusters, or number of scenes, M is either set manually or determined by two other methods. The first is to stop the partitioning operations once the $Ncut$ value is above a threshold T_{cut}, which is determined by $T_{cut} = a\sqrt{N_s} + c$, where a=0.02, c=0.l3, and N_s is the number of shots. The second is to make use of the Q-function [57]:

$$Q(p_m) = \sum_{c=1}^{m} \left[\frac{assoc(C_c, C_c)}{assoc(V, V)} - \left(\frac{assoc(C_c, V)}{assoc(V, V)} \right)^2 \right] \tag{4.5}$$

where p_m is a partition of the shots into m sub-groups by $m-1$ cuts. The number of scenes is then determined by $M = argmax(Q(p_m))$.

Naci *et al.* [35] use Spectral clustering by Normalized cuts and minimize the value of $Ncut(C_n, C_m)$ [58] by finding the eigenvalues and eigenvectors of the similarity matrix, which is composed of pairwise similarities between shots, keyframes of shots, or of all frames. They determine that the second smallest eigenvalue is the minimal value of $Ncut(C_n, C_m)$. Every entry $x^{(2)}(i)$

of the second eigenvector $x^{(2)}$ corresponds to a point i in the data set and can be clustered as follows:

$$\text{if } x^{(2)}(i) < 0 \; i \in C_n$$
$$\text{if } x^{(2)}(i) > 0 \; i \in C_m$$

The graph is repeatedly partitioned until $Ncut(C_n, C_m)$ is smaller than a preset threshold $NCUT$.

In [29], Ngo *et al.* minimize $Ncut(C_n, C_m)$ using a standard eigensystem

$$D^{-\frac{1}{2}}(D - W)D^{-\frac{1}{2}}z = \lambda z \tag{4.6}$$

where D is a diagonal matrix with $\sum_j w(s_i, s_j)$ along the diagonal and W is the similarity matrix. As previously seen, the second smallest eigenvalue is used to find the sets C_n and C_m. The graph is repeatedly partitioned until $Ncut(C_n, C_m)$ is smaller than a threshold $Ts = \mu_{sim} + \sigma_{sim}$, where μ_{sim} is the average shot similarity and σ_{sim} is the standard deviation of shot similarity.

Once this has been accomplished, time order information is added to create a directed temporal graph $TG(V, E)$, where vertices V represent the clusters and the edges E are weighted by the transition probabilities between clusters. A directed edge is added from cluster C_n to cluster C_m if shot s_i is in the first and shot s_{i+1} in the second. The transition probabilities can be computed as:

$$P_r(C_m|C_n) = \frac{1}{|C_n|} \sum_{s_i \in C_m} \sum_{s_j \in C_n} \tau(i - j) \tag{4.7}$$

where N_s is the number of shots, $|C_n|$ is the number of shots in cluster C_n, and

$$\tau(x) = \begin{cases} 1, & \text{if } x = 1 \\ 0, & \text{otherwise} \end{cases}$$

The last step is to divide the latest graph into scenes. The authors note that there must be at most one edge between scenes and that a scene is composed of at least one cluster. Keeping these in mind, the scenes are detected by first computing the shortest paths, using Dijkstra's algorithm, between the cluster with the first shot and the cluster with the last shot. All edges have unity weight. Edges between clusters C_i and C_j are then removed if $i = j + 1$

and if there is no path between C_i and C_j, these clusters belong to different scenes.

Sasongko *et al.* [31] also use a graph-based approach. They build an undirected graph as in the previous methods, then a minimum spanning tree is built and the edges that have a distance, which is in terms of similarity, greater than a threshold are removed. This creates clusters which represent the scenes. A tree-based peeling strategy is also employed with a graph-based method in [27].

Another clustering method that has been observed is a window type algorithm where the current shot is compared to certain shots ahead of it and some behind. In [45], Lin and Zhang define an attraction ratio $R(s_i)$ for a current shot i towards a new scene. Using a window of size 3, they calculate the similarity of the shot with those ahead and behind. If the attraction ratio is greater than a threshold and if $R(s_i) > R(s_{i-1})$ and $R(s_i) > R(s_{i+1})$, then the attraction is stronger from the *right side* and the shot starts a new scene, otherwise it is absorbed into the current scene. The attraction ratio is calculated as:

$$R(s_i) = \frac{(right(s_i) + right(s_{i+1}))}{(left(s_i) + left(s_{i+1}))} \tag{4.8}$$

where,

$$left(s_i) = max\,\{sim(s_i, s_{i-1}), sim(s_i, s_{i-2}), sim(s_i, s_{i-3})\} \tag{4.9}$$

$$left(s_{i+1}) = max\,\{sim(s_{i+1}, s_{i-1}), sim(s_{i+1}, s_{i-2})\} \tag{4.10}$$

$$right(s_i) = max\,\{sim(s_i, s_{i+1}), sim(s_i, s_{i+2}), sim(s_i, s_{i+3})\} \tag{4.11}$$

$$right(s_{i+1}) = max\,\{sim(s_{i+1}, s_{i+2}), sim(s_{i+1}, s_{i+3}), sim(s_{i+1}, s_{i+4})\} \tag{4.12}$$

The method is improved upon in [46] by dividing the attraction ratio into a splitting force $Fs(s_i)$ and a merging force $Fm(s_i)$. The splitting force for the current scene boundary candidate is calculated as:

$$Fs(s_i|s_{i+1}) = \frac{(Fs(s_i) + 1)/(Fs(s_{i+1}))}{2} \tag{4.13}$$

where,

$$Fs(s_i) = \frac{left(s_i)}{right(s_i)} \tag{4.14}$$

and $left(s_i)$ and $right(s_i)$ are the same as previously defined.

The merging force for the current scene boundary candidate is calculated as:

$$Fm(s_i|s_{i+1}) = \frac{(Fm(s_i) + Fm(s_{i+1}))}{2} \tag{4.15}$$

where,

$$Fm(s_i) = \frac{1}{3}\sum_{j=i-3}^{i-1} max\,\{sim(s_j, s_{i+1}), sim(s_j, s_{i+2}), sim(s_j, s_{i+3})\} \tag{4.16}$$

The forces are normalized into $Fm'(s_i|s_{i+1})$ and $Fs'(s_i|s_{i+1})$ using their respective means and standard deviations. The authors note that at an ideal boundary, the splitting force and merging force should be at a maximum and minimum respectively. Thus, they specify two conditions upon which a scene boundary should be declared.

1. $Fs'(s_i|s_{i+1})$ reaches a maximum and is greater than a threshold T_1, and $Fm'(s_i|s_{i+1})$ reaches a minimum at the same time.
2. $Fs'(s_i|s_{i+1})$ reaches a maximum or $Fm'(s_i|s_{i+1})$ reaches minimum, and $Fs'(s_i|s_{i+1})$ is greater than a threshold T_2 and $Fs'(s_i|s_{i+1}) - Fm'(s_i|s_{i+1})$ is greater than a threshold T_3.

It should be noted, that the authors experimentally determined the values of the thresholds.

Lu *et al.* [59] also attempt to create comparable measures. They employ what they refer to as a *continuity* measure. It is a weighted sum of *cohesion*, which points out consistency across shots, and *disjunction*, which points out inconsistencies. They notice that any number of similarity measures can be used for *cohesion* and features such as the length of black frames, silences, or the appearance of wipes or fades can be used for *disjunction*.

Wang *et al.* [8] employ a window-based method to perform shot grouping. They refer to their method as the *overlapping links method*. It consists of three steps which are a forward search, a backward search, and a step to declare a scene boundary. The method uses a forward search with $fw =$

$min(F, CtFutureShot)$, a backward search with $r = min(R, CtPreShot)$, where R and F are the backward and forward search range respectively and $CtPreShot$ and $CtFutureShot$ are the shots before and after the current shot, respectively.

1. First, find the most similar shot within the forward search range. If such a shot exists, group the paired shots into the same scene and set the similar shot as the current shot, otherwise go to step 2.

2. Second, set shot i as the current one and find the most similar previous shot. If it exists group the paired shots into the same scene and go to step 1, otherwise, decrease the current shot i by 1 and repeat this step until the r^{th} previous shot is the current shot.

3. The third step is to declare a scene boundary immediately after the current shot, set the next shot as the current, and go to step 1.

In [48], Chen *et al.* first compute similarity between neighboring shots and if the similarity is smaller than the mean μ of all similarities then a potential scene boundary is marked between the shots. Usually this method tends to over segment the video so the authors propose another step to further merge the candidate scenes. A scene similarity measure is proposed as

$$SceneSim(G_1, G_2) = \frac{1}{m \times n} \sum_{i=1}^{m} \sum_{j=1}^{n} ShotSim(s_{1i}, s_{2j}) \tag{4.17}$$

The candidate scenes G_1 and G_2 are similar if $SceneSim(G_1, G_2)$ is greater than $\mu - \frac{\sigma}{2}$, where σ is the standard deviation of all similarities between shots. The merging of candidate scenes is then performed with a five step algorithm.

1. Set the expanding scene to be the first scene, say G_i.

2. Compare the scene with two following scenes G_{i+1} and G_{i+2}.

3. If the expanding scene is similar to scene G_{i+2}, merge all three scenes, set the new scene as the expanding scene and go to 2.

4. If the expanding scene is similar to scene G_{i+1}, group the two scenes, set the new scene as the expanding scene, and go to 2.

5. If none of the scenes are similar, set scene G_{i+1} as the expanding scene and go to 2.

This is continued until all of the scenes have been examined.

Several other papers employ similar shot grouping methods [11][13]. In [11], each shot is compared with the next 3, if one of the shots is deemed similar then the two similar shots and all the shots in between are grouped together. This is repeated until all shots have been inspected. The similarity measure is compared with a threshold to determine if the shots are similar. A comparable algorithm, referred to as *bi-directional searching*, is used in [32].

One very simple, and most likely error-prone method, used to group shots [21][24][27][28], is to perform pair-wise comparison of shots and to consider the end of the first as a scene boundary if the similarity measure between the two is below a certain threshold.

A k-means clustering approach is used in [18] and [33]. The general idea is to further divide shots into sub-shots, by sub-sampling them based on shot duration, then apply k-means clustering to find and merge similar sub-shots. This does not, however, take the temporal layout of the shots into consideration which is a measure that should be incorporated into shot clustering methods for scene detection. The authors of [44] apply a modified k-means algorithm for shot clustering.

Hierarchical agglomerative clustering is performed in [20] and [38]. It begins by creating as many clusters as there are segments, or shots, and at each step the closest two clusters are merged until there remains a single cluster. Euclidean distance between HSV color histograms is used as the distance measure. Once the final tree is obtained, it is then cut at a level where the number of leaves equals the chosen number of clusters or scenes.

In [47], Yamasaki *et al.* employ Gaussian Mixture models (GMM) in their method for clustering shots. Initially, the number of clusters is equal to the number of shots. The next 5 steps are detailed in the following:

1. The distances between all clusters are calculated.

2. The two closest clusters are modeled with a single Gaussian.

3. If the number of clusters N_c is smaller than a maximum T_{max}, return to 1.
   If $N_c < T_{max}$ and greater than a minimum T_{min}, go to 4.
   Otherwise, if $N_c < T_{min}$ go to 5.

4. Form a GMM from the Gaussians obtained in the previous steps and compute description length DL.

$$DL(N_c) = -l + w_{sc}p \tag{4.18}$$

   where

$$p = \frac{1}{2}\left[(c-1) + c\left\{n + \frac{1}{2}n(n+1)\right\}\right] logN_f \tag{4.19}$$

   and l is the likelihood of the GMM, N_f is the total number of frames, c is the mixture of the GMM, n is the dimension of the features, P is a penalty with respect to the number of scenes, and w_{sc} is a predetermined weight to adjust the estimated number of scenes.

5. The optimal number of scenes is then determined by

$$\hat{N}_c = argmin_{N_c} DL(N_c) \tag{4.20}$$

They also change some of the parameters in the algorithm in order to cluster similar scenes.

The authors of [51] and [52] use similar techniques for scene boundary detection. Zhu and Ming [51] first combine audio and visual features into a single feature vector then employ a Support Vector Machine to classify segments into either news, commercial, cartoon, music, weather, tennis, basketball, or football. Once this has been performed, if the classification process has been accurate, the scene transitions can easily be found. In [52], the same method is used but with a Hidden Markov model to classify shots.

Zhai and Shah [23] remark that scene boundaries are located where there are changes in the properties of what they refer to as the *central concept*, which can be environment, topics or sub-themes. This can be seen as a change-point problem and they use a Markov chain Monte Carlo in their approach to scene detection since it has been commonly employed to solve these sorts of problems. Other methods that have been used for grouping or finding similar video segments include a frame and shot labeling method [37], one that employs a Longest Common Subsequence model [19], a Greedy RSC clustering algorithm [12], and a k-nearest neighbors method is employed in [60].

4.2 Proposed Method

After having examined a number of different scene detection methods found in the literature, our proposed method will be presented in detail in the following paragraphs.

Using the previously selected frame similarity measure and keyframe extraction method (see section 3.4) a shot similarity map is created. The full $N \times N$ similarity map, where N is the number of shots in the film, is not constructed because of the long computation time that it would require. Instead, after manually testing different lengths, it was decided that each shot should be limited to a window of 60 shots in the forward direction and 60 shots in the backwards direction. The similarity map uses the normalized color histogram values to obtain a similarity value between two shots.

For every similarity point between shot s_i and shot s_j:

$$ShotSim_{map}(s_i, s_j) = min\left\{VisSim_{bhatt}(k_m^i, k_n^j)\right\} \tag{4.21}$$

A temporal distance weight is also added to the similarity value, such that the further apart the shots are the less likely they are to be similar. This weighting function is essentially an inverse Gaussian with its lowest point at 0 and its width determined by the total number of shots:

$$W_{td}(s_i, s_j) = r \times \left[\left(1 - \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(s_i - s_j)^2}{2\sigma^2}}\right) + \left(\frac{1}{\sigma\sqrt{2\pi}}\right) - 1\right] \tag{4.22}$$

where $r = 0.049 \times \sigma\sqrt{2\pi}$, $\sigma = round\left(\frac{round(N)}{100} + 1\right)$, $round(N)$ rounds the value of N to the nearest integer, and N is the number of shots.

The final similarity measure used in the computation of similarity points for the map is:

$$Sim_{map}(s_i, s_j) = ShotSim_{map}(s_i, s_j) + W_{td}(s_i, s_j) \tag{4.23}$$

Once the temporal distance weight is added to the similarity values, the threshold value for Bhattacharya distance is adjusted for non-normalized Bhattacharya distance values and applied to the similarity map such that similarity values less that the threshold are deemed to be similar and are assigned a value of 1 (black), otherwise they are assigned a value of 0 (white).

Now with a binary shot similarity map, clusters of similar shots, depicted as black regions, can be observed on the map. These clusters are located along the diagonal and are in a square-like shape. The objective is to obtain the boundaries of these clusters to use as scene boundaries. A section of the binary shot similarity map obtained from the movie *The Shawshank Redemption* is shown in Figure 4.1.

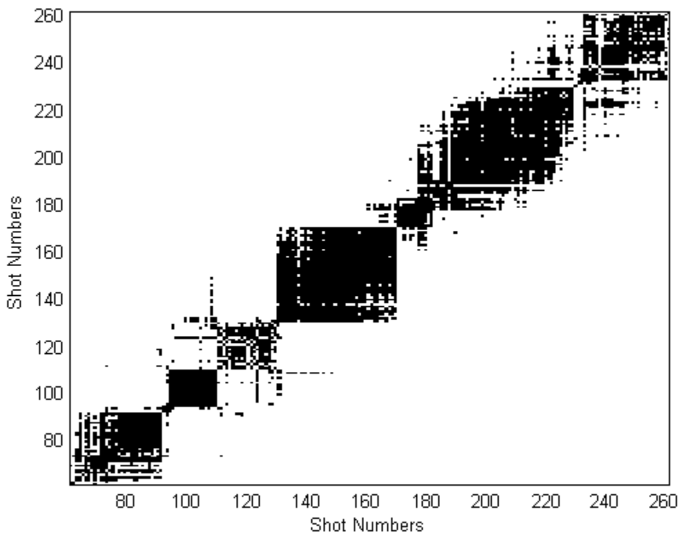


Figure 4.1: A Section of the movie *The Shawshank Redemption*'s Binary Shot Similarity Map

Since the map is symmetrical, we only need to examine one half of the map, either side of the diagonal, and locate the triangular clusters.

The first step is to find the starting point, the first shot of the cluster. Generally, the starting shots will have a large number of similar shots ahead of them, that is a large number of forward similarities. Forward shots can be defined as the shots that succeed the current shot but are within the window of 60 shots in the forward direction. This is due to the similarity map having been constrained to this amount in both directions and as such the similarities beyond these limits are not calculated.

For each shot, the sum of forward shot similarities is calculated using the binary similarity map. The shot with the largest sum is then chosen as the starting point to find the first cluster, see Figure 4.2. More often than not, the first starting point, the one with the overall largest sum, will mark the beginning of the biggest cluster.

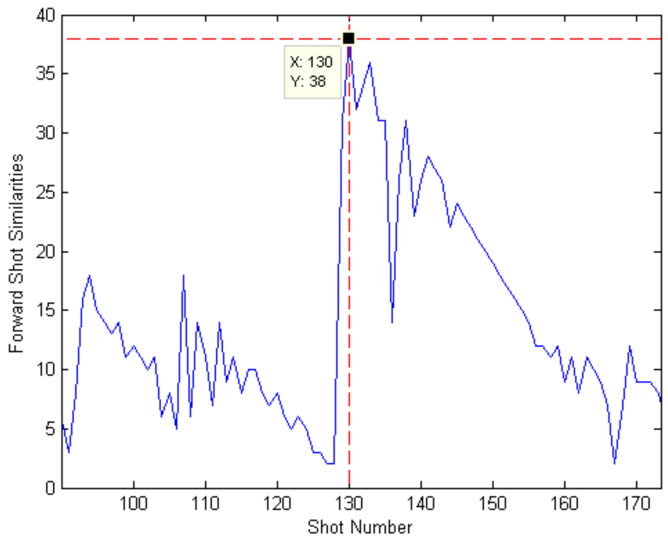


Figure 4.2: Largest Sum of Forward Shot Similarities, example

From the found starting shot, the distance, in terms of shots, to its furthest similar shot is obtained. This distance is used to create the first triangle whose boundaries will mark the initial cluster, see Figure 4.3.

Once those initial boundaries are located, the number of similar shots inside the triangle, N^-, and the number of shots above the triangle, N^+, are obtained and the ratio $\frac{(N^- - N^+)}{Area of Triangle}$ is used as the measure which determines

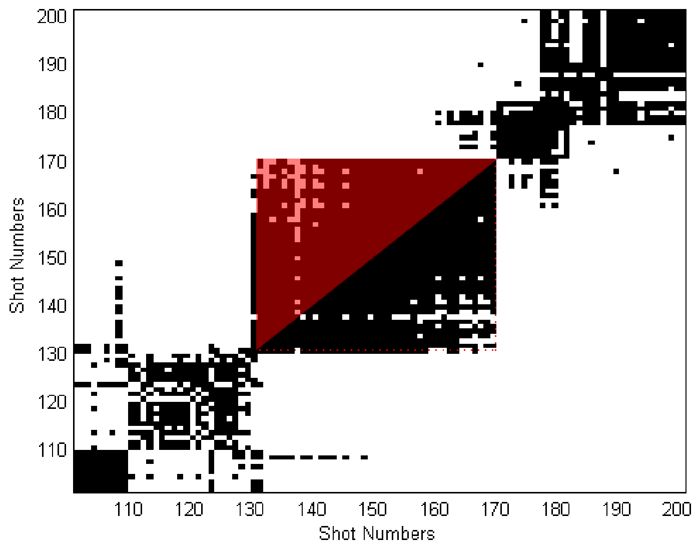


Figure 4.3: Triangle (red) on the Full Similarity Map, example

the quality of the selected boundaries, Figure 4.4. This ratio is chosen because the clusters are not always filled with similar points, so this ratio will ensure that the most appropriate cluster will be found, rather than the one with the greatest number of similar shots within.

The size of the cluster is then reduced by one shot in each direction and the ratio is calculated for these triangle boundaries. This is repeated, a reduction by one shot and the calculation of the ratio, until the minimum size of 3 shots is attained. This size is chosen because it is assumed that scenes should contain a minimum of 3 shots.

When this has been done and all the ratios have been calculated, the triangle (boundaries) that results in the highest ratio is selected as the chosen boundaries for the cluster. The sums of forward similarities within the found cluster are set to 0 so they will not be considered in the next runs. The above procedures are then repeated, finding the highest sum, creating the triangle and calculating its ratio, selecting the highest ratio and clearing the sums within the chosen boundaries, until the highest sum value is less than a threshold.

Once this is done and no further clusters can be found, overlapping po-

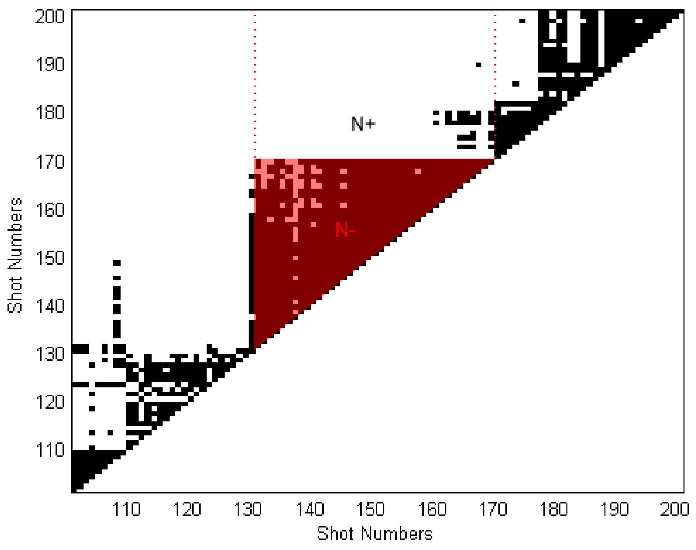


Figure 4.4: Formed Triangle (red) from Furthest Shot and Ratio Calculation, example

tential scenes are dealt with using the following guidelines: If the overlap is greater than a threshold number of shots, or if the overlap is for more than 50% of one of the potential scenes, then the overlapping potential scenes are merged into a single scene. Figure 4.5. Otherwise the intersecting region is examined and for each of the possible non-overlapping combinations of the two potential scenes, the ratios are calculated for both sections and added together: $\frac{(N_1^- - N_1^+)}{AreaofTriangle_1} + \frac{(N_2^- - N_2^+)}{AreaofTriangle_2}$. The combination that yields the highest ratio value of this sum is selected as the cutting point between the scenes. Figure 4.6.

To summarize, the proposed scene detection method can be broken down into the following steps. First, calculate the similarity map using eq.(4.23) and the limiting window of 60 shots in the forward and backward direction. Next, threshold the map to obtain a binary similarity map. The general algorithm for locating scene clusters is then applied.

1. For each shot, calculate the sum of forward shot similarities.

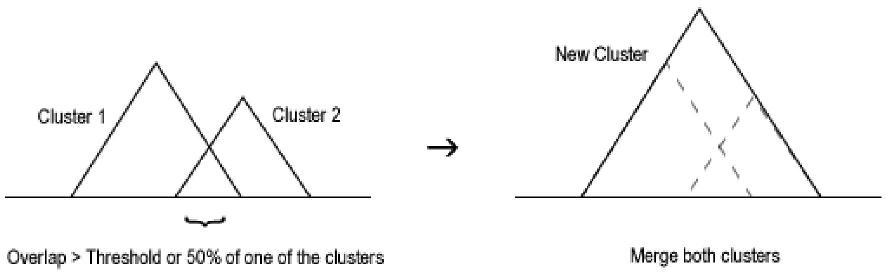


Figure 4.5: Overlap Condition 1

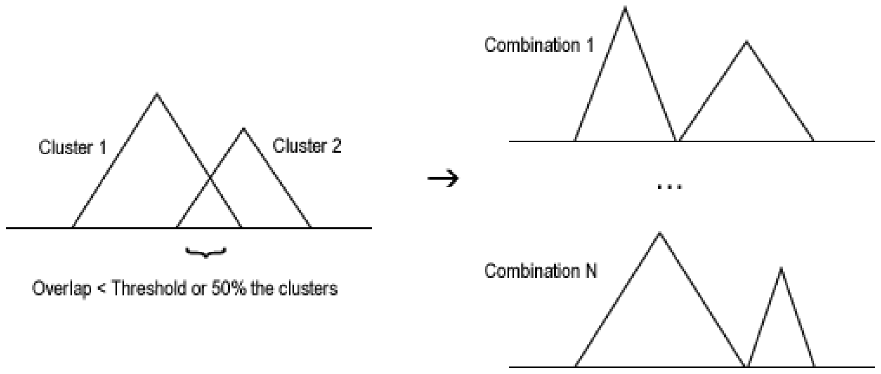


Figure 4.6: Overlap Condition 2

2. Find the shot with the largest sum, shot s_i.
3. Create a triangle from the furthest similar shot to shot s_i.
4. Count the number of similar shots inside the triangle, N^-, and the number of similar shots above the triangle, N^+. Calculate the ratio $\frac{(N^- - N^+)}{AreaofTriangle}$.
5. Repeat from step 3 while reducing the triangle size to a minimum size of 3 shots.
6. Select the triangle with the highest ratio value.

7. Set the sums (of forward shot similarities) of the shots within this potential scene to 0. Repeat from 2 until the highest sum value is less than a threshold.

8. Manage overlapping clusters. If the overlapping region is greater than a threshold or contains 50% of one of the clusters, merge the two. Otherwise examine each of the possible non-overlapping combinations and select the one which results in the highest ratio value.

4.3 Experimental Comparison

Having presented various methods for scene detection, selected methods, along with the proposed method, will be chosen for comparison and review.

4.3.1 Comparison Measures

In order to be able to determine the efficiency of various scene boundary and clustering algorithms, ground truths must first be constructed. It should be noted that when we refer to Ground Truth for scenes, it is in fact a relative term. This is because ideally the *Director's Cut* is the absolute ground truth. However, it is difficult to obtain this for each film and, as such, our "relative" ground truths are used. Three films were selected for testing and two ground truths were created manually for each of the following films: *The Shawshank Redemption*, *3:10 to Yuma*, and *Gran Torino*. Two different subjects, or educated users, each created their own ground truths. They separately marked the scene boundaries for the entirety of each movie while using the scene definition from [24] as a guideline.

1. When there are no two interwoven parallel actions: a change in location, or time, or both defines a scene change.

2. An establishing shot, although maybe different in location to its corresponding scene, is considered part of that scene, as they are unified by dramatic incidence.

3. When parallel actions are present and interleaved, and there is a switch between one action to another, a scene boundary is marked if and only if the duration of the action is shown for at least 30 seconds.

4. A montage sequence that involves “dynamic cutting”, where shots containing different spatial properties are rapidly joined to convey a single dramatic event, forms a single scene.

While the usual measures applied to evaluate the efficiency of information retrieval methods are *Precision* and *Recall* they tend to not be sensitive to near misses. In the context of scene boundary detection, near misses should not be penalized as much as complete misses or boundaries detected much further away and, for this reason, the *WindowDiff* evaluation metric [61] has been chosen to compare scene boundaries.

This metric calculates the error using a moving window with a length of k, that is set to half the average ground truth segment size, in other words half the average size of a scene. For each position of the window, the number of ground truth boundaries, r_i, and algorithm boundaries, a_i, are compared. A penalty is added if the number of boundaries are not equal.

$$WindowDiff(gt, alg) = \frac{1}{N-k} \sum_{i=1}^{N-k} \Big(|b(gt_i, gt_{i+k}) - b(alg_i, alg_{i+k})| > 0 \Big) \tag{4.24}$$

where, $b(i, i+k)$ is the number of boundaries between position i and $i+k$, N is the total number of shots in the movie, gt_i is the i^{th} boundary from the ground truth, and alg_i is the i^{th} boundary found using the algorithm to be tested.

WindowDiff is similar to a distance measure, the smaller the *WindowDiff* value the closer the two scene boundary results are to each other.

4.3.2 Results

A number of shot clustering and scene boundary detection methods were selected and implemented, based on the details found in their respective papers, to be evaluated alongside our method. The first is a simple Neighboring method [27] (page 51) which places a scene boundary between two shots if they are not considered similar. The Neighbor Merge method expands on the neighboring method by further merging those potential scenes [48] (page 51). The Attraction Ratio method [45] (page 49) and its improved version the Splitting and Merging Forces method [46] (page 49), the Overlapping Links method (OLM) [8] (page 50), a Modified Overlapping Links method

Table 4.1: Scene Detection *WindowDiff* Values, The Shawshank Redemption

Scene Boundary Method	Number of Scene Boundaries (SB)	$WindowDiff_{GT1}$ (122 SB)	$WindowDiff_{GT2}$ (123 SB)
Neighbor	511	1.9075	1.8535
Neighbor Merge	149	0.7790	0.7303
Attraction Ratio	518	1.8086	1.7705
Splitting and Merging Forces	274	1.0037	0.9646
Overlapping Links	144	0.6965	0.6679
Modified OLM	102	0.6245	0.5970
Window-of-3	128	0.6510	0.6298
Our Method	99	0.5315	0.5050
Other Ground Truth	-	0.1957	0.1957

SB : Scene Boundaries
$WindowDiff_{GT1}$: WindowDiff value when compared to Ground Truth 1
$WindowDiff_{GT2}$: WindowDiff value when compared to Ground Truth 2

[62], and a Window-of-3 method, which clusters using a forward search with a window of 3 [11] (page 52), were also selected. The similarity measure and keyframe extraction method employed for each of these measures, if required, are the previously selected (see section 3.4) Bhattacharya distance and Sampled keyframe method with their appropriate thresholds.

These methods were compared with both of the ground truths of each film. The two different ground truths were also compared (*Other Ground Truth* in the tables) to obtain an idea of the variations due to semantics even when employing a standard scene definition.

As was previously explained, the *WindowDiff* distance method was selected as the scene boundary comparison method because of its ability to incorporate near misses into the metric and not simply apply a penalty for a direct true or false. When examining the result tables, Table 4.1, Table 4.2, and Table 4.3, the number of boundaries detected along with the *WindowDiff* value of each different method can be observed.

The *WindowDiff* results between the pair of ground truths for each film

Table 4.2: Scene Detection *WindowDiff* Values, 3:10 to Yuma

Scene Boundary Method	Number of Scene Boundaries (SB)	$WindowDiff_{GT1}$ (47 SB)	$WindowDiff_{GT2}$ (35 SB)
Neighbor	964	9.9591	14.4523
Neighbor Merge	317	3.0427	4.5514
Attraction Ratio	778	8.0291	11.7297
Splitting and Merging Forces	561	5.6316	8.2740
Overlapping Links	365	3.6176	5.3310
Modified OLM	303	3.0037	4.4209
Window-of-3	339	3.3541	4.9327
Our Method	116	1.0640	1.4786
Other Ground Truth	-	0.2241	0.3166

Table 4.3: Scene Detection *WindowDiff* Values, Gran Torino

Scene Boundary Method	Number of Scene Boundaries (SB)	$WindowDiff_{GT1}$ (82 SB)	$WindowDiff_{GT2}$ (95 SB)
Neighbor	498	2.8970	2.3952
Neighbor Merge	87	0.6659	0.6504
Attraction Ratio	440	2.4360	1.9397
Splitting and Merging Forces	171	1.0122	0.9050
Overlapping Links	72	0.6396	0.6291
Modified OLM	35	0.5793	0.5822
Window-of-3	66	0.6018	0.6084
Our Method	84	0.4494	0.4915
Other Ground Truth	-	0.2250	0.1998

averages at a distance of about 0.22, this difference is caused by the slight differences in semantic understanding from the ground truth creators even when each is employing the same standard definitions of a scene.

The number of boundaries detected can usually be used as an early indicator of the efficiency of the detection method. The Neighbor method generally returns a very high number of scene boundaries because of its simple and straightforward approach of marking boundaries only where the similarity between neighboring shots does not satisfy the threshold. Examining the three result tables, this method has the highest *WindowDiff* values for each of the tested movie ground truths. Continuing from the Neighboring method to the Neighbor Merge method by grouping the similar scenes, from those that were previously found, brings the number of scene boundaries down to a more reasonable amount which is much closer to the ground truth numbers. As a result of the merging, the *WindowDiff* values are also much improved but the method still sits 5^{th} best out of 8 methods.

The Attraction Ratio method produces the second highest amount of boundaries and the *WindowDiff* values obtained also reflect this and are relatively high as well. A slight variation of this method, the Splitting and Merging Forces method produces approximately half the number of boundaries and also improves the *WindowDiff* distance values but they are still not within a satisfactory range.

The Overlapping Links method uses a combination of forward and backward searches to locate scene boundaries. For all of the tested movies, with the exception of *3:10 to Yuma*, which turned out to be a fairly difficult movie to segment, the number of boundaries detected with this method are fairly close to the ground truth numbers. The *WindowDiff* results obtained are decent but not outstanding. The modified Overlapping Links method, modified in a way to avoid the backward search component, generally generated a number of scene boundaries a fair bit below the Overlapping Links numbers and the ground truth numbers. Its calculated *WindowDiff* distance results, however, are better than the ones obtained in the non-modified method. This would indicate that even though a small number of boundaries are found, the locations of these boundaries are much closer to the locations of those in the ground truths.

The last method tested before our proposed method, is the Window-of-3 method. This method uses a sliding window with a size of 3 shots to group similar shots until a similar shot is not found in the window. The number of boundaries located varies quite a bit from movie to movie but the *WindowDiff* values are generally in the top half with regards to performance.

Finally, our proposed method for scene detection finds a number of boundaries which is close to the movie's ground truth numbers except for *3:10 to Yuma* where the number of boundaries is quite a bit higher than its ground truth's but are still closer to the actual number than the other methods tested. The *WindowDiff* results for this method show its superiority as they are all the top ranked result for each ground truth. In addition to the *WindowDiff* values, the found scene boundaries for the first 800 shots can be seen in Appendix B. When examining these bar graphs, it is relatively clear that the boundaries found with our proposed method are consistently the closest to those observed in the ground truths. The scene boundaries for our proposed method can also be observed on the similarity maps, containing the first 400 shots, in Appendix C. In addition, the first few scenes detected using our proposed method can be seen, with the middle keyframes used to represent the shot content, in Appendix D.

4.4 Conclusion

This chapter presented various scene detection methods from the literature. Several different types were examined including graph-based methods, window-type methods, shot grouping methods, and other clustering-based methods that employed different algorithms such as k-means clustering, Gaussian mixture models, and support vector machines.

The proposed scene detection method and its various procedures were also described. The first step is to create a similarity map of the film using the previously chosen keyframe extraction method and shot similarity measure. The similarity map is limited to 60 shots in the forward direction from the diagonal and 60 shots in the backward direction to avoid the tedious process of calculating the entire map. A temporal distance weight is then applied to the map and a threshold, the minimum NDCR value for Bhattacharya distance, is applied to determine if shots are similar or not and to change the map into its binary form, 1 for similar or 0 otherwise. The last step is to process this similarity map by locating the square clusters along the diagonal which correspond to the movie's scenes.

Several different scene detection methods were then applied to three films, *The Shawshank Redemption*, *3:10 to Yuma*, and *Gran Torino*, and compared using the *WindowDiff* method, in order to determine their efficiency at locating scene boundaries. Eight methods were compared: the Neighbor method, the Neighbor Merge method, the Attraction Ratio method, the Splitting and Merging Forces method, the Overlapping Links method, a modified version of the Overlapping Links method, a Window-of-3 method, and the proposed method. Since it is difficult to define what a scene is and because scene detection is semantics oriented, two ground truths were created, each by different subjects, for all three films.

The results vary slightly per movie in terms of method efficiency and number of boundaries. It can be seen that the segmentation for *3:10 to Yuma* is much more difficult than the other movies. Examining this movie's scene detection results, it is evident that it is a complex case as performance is low for all techniques. For our proposed method in particular, this can be explained by inspecting the Figures C.4, C.5, and C.6 of Appendix C. The similarity maps are very diffuse and without several clear clusters which explains why the method over-segments this movie. The diffuse similarity maps is likely due to the fact that the locations, which define a large percentage of the scenes, are generally large ones such as valleys or small towns. Even with this difficulty, the proposed method still outperforms the other scene boundary detection methods and gives results that are most similar to each film's ground truth.

4.4.1 Video Search Tool

Having selected a shot detection method, a keyframe extraction method, a shot similarity measure, and a scene boundary detection algorithm, they were implemented to segment videos in the *Video Search Tool.*

The objective was to create a tool that facilitates browsing and searching through a video by hierarchically organizing the video into its film units. The *Video Search Tool* is readily capable of this. The initial step, when using the tool for the first time, is to load the video and perform shot and scene detection. The respective boundaries are then saved for future use. Next, the subtitles should be loaded, in order to perform keyword searches. Once a search has been carried out, the results will be displayed in the form of subtitles that contain the keyword or a synonym of the keyword. These subtitles can be selected (clicked) to view the single subtitle, the entire shot which contains the subtitle, or the entire scene that contains the subtitle. See Appendix E for a detailed description of the *Video Search Tool.*

Chapter 5

Conclusion

The works presented in this thesis were put together with the final goal of creating a *Video Search Tool* that first segments the video into shots and scenes strictly using visual content, and can also perform keyword queries through the video's subtitle transcript.

5.1 Summary

First, various types of shot detection algorithms were examined such as edge-based methods, motion-based methods, MPEG-based methods, and histogram-based methods before presenting the selected method. The shot detection is done by first performing a color reduction from RGB to the Lab color space using the 216 colors of the "webmaster" palette, calculating the average Euclidean distance between the four regions of the color histograms of neighboring frames, applying a second order derivative to the resulting distance vector, and finally using the automatically calculated threshold to locate shot boundaries.

In the second chapter, keyframe extraction methods, frame similarity, and shot similarity measures were presented. Keyframe extraction methods such as selecting the middle frame, and frame clustering to select keyframes, frame similarity measures including Intersection, Euclidean distance, and Bhattacharya distance, and shot similarity measures including histogram-based, dominant color-based, motion-based, and edge-based measures, were presented. Selected similarity measures were tested and compared using DET curves and certain statistical measures. Bhattacharya distance provided the best results and was chosen for use. Next, keyframe extraction methods were compared in order to select a method, and also with the purpose of determining whether a complex method would give better results than simple frame sampling. In the end, obtaining keyframe sets by sampling frames was selected for use.

Having chosen the similarity measure and keyframe extraction method, the next chapter considered scene detection methods. Graph-based, window-based, and shot clustering-based methods were among the previously used methods that were examined. An original method, initially for use with feature films but which can also be used on any video, was then presented with a detailed procedure. A shot similarity map of the film is first constructed using the Bhattacharya distance metric to compute the visual similarity between shot's keyframes. A window of 60 shots in the forward direction and 60 shots in the backwards direction is used to limit the size of the similarity map and save on computation time. A temporal distance weight is then added to the shot similarity values, in order to ensure that shots which are a great temporal distance from each other, are less likely to be deemed similar. Next, a threshold is applied to create a binary shot similarity map which consists of similar shots, with a value of 1, and non-similar shots, with a value of 0. The last step involves locating the square clusters of similar shots along the diagonal of the binary map, which are taken as the scenes for the film.

A selection of scene detection methods were then tested alongside our proposed method. Experimentation with 3 movies demonstrated that the proposed method outperforms every other, in terms of scene boundary locations with respect to those of the ground truths, measured using the *WindowDiff* metric.

The chosen approaches were implemented in a *Video Search Tool*, see Appendix E.

5.2 Contributions

The following items describe the unique contributions of this thesis.

- Shot similarity measures along with keyframe extraction methods were examined and compared. Using a manually constructed ground truth, composed of four segments from the movie *The Shawshank Redemption*, DET curves were used to select appropriate thresholds and several measures were calculated. It was found that the combination of Bhattacharya distance along with the Sampled keyframe selection method was best suited for this problem.

- A new approach to scene detection, which is based strictly on visual features from the shots, was developed. It first involves creating a shot similarity map using the selected shot similarity and keyframe extraction method. Next, a temporal distance weight and a predetermined threshold are applied to the map to obtain the final binary similarity map. The last step uses the proposed algorithm to locate the shot clusters along the diagonal which correspond to scenes.

- The proposed method, along with the selected shot detection and keyframe selection methods, were implemented and are applied in the created *Video Search Tool*. It first segments the video into shots and scenes, then is able to perform keyword queries through the video's subtitle transcript.

5.3 Future Work

The works presented in this thesis were done so for the goal of implementing a hierarchical video segmentation and searching system. There are several improvements that are being looked into to ameliorate current methods and to add extra functionalities to the program.

One improvement being examined is the application of multimodal scene detection using a combination of textual segmentation, M. Scaiano's method [63], along with the visual segmentation method that is currently used. The idea is to employ a text segmentation method to locate topical scene breaks in the film subtitles. Once these boundaries are located, they can be used to reinforce the scene boundaries found with the current method. This could prove to be an effective method for enhancing the precision of the scene detection method on movies which are difficult to segment using visual methods, such as *3:10 to Yuma*.

Among other additions under consideration, is the option to include the ability to add notes to a time segment on a video timeline, in other words performing video annotation. This could be done for a frame, shot, scene, or a video segment of a manually selected length. In addition, the annotations would be included in the search process, thus adding an extra layer of searchable text information to the video.

Another interest is to add to the searching capabilities by performing a search to locate particular visual cues throughout the movie. This is a specific application which would have to be predefined before its addition to the system. For example, for a user whose interests lie in the field of Religious studies, it may be interesting to locate the sections in the video where a person is walking by or in water, or when a person is with their arms wide open standing in a cross or t-shape position. One example of this situation is the escape scene in *The Shawshank Redemption*.

The possibility of using an automatic thresholding method, to calculate the threshold for similarity for each different movie, should also be explored to determine if it would enhance scene detection performance.

In addition to the proposed improvements above, the ability to save video clips and to sort the results returned from the search by different categories, such as relevance and time, can be added.

Bibliography

[1] X. Zhu, W.G. Aref, J. Fan, A.C. Catlin, and A.K. Elmagarmid, "Medical Video Mining for Efficient Database Indexing, Management, and Access", In Proceedings of the 19th International Conference on Data Engineering, pp. 569-580, 2003.

[2] X. Zhu, A.K. Elmagarmid, X. Xue, L. Wu, and A.C. Catlin, "Insight Video: Toward Hierarchical Video Content Organization for Efficient Browsing, Summarization, and Retrieval", IEEE Transactions on Multimedia, Vol. 7, No. 4, pp. 648-666, 2005.

[3] Y. Zhu and D. Zhou, "Video Browsing and Retrieval Based on Multimodal Integration", In IEEE/WIC International Conference on Web Intelligence, pp. 650-653, 2003.

[4] A. Dong and H. Li, "Educational Documentary Video Segmentation and Access through Combination of Visual, Audio, and Text Understanding", IEEE International Symposium on Signal Processing and Information Technology, pp. 652-657, 2005.

[5] C.J. Fu, G.H. Li, X.W. Xu, and K.X. Dai, "Mining Video Hierarchical Structure for Efficient Management and Access", Proceedings of the Fifth International Conference on Machine Learning and Cybernetics, pp. 1013-1018, August 2006.

[6] R. Zabih, J. Miller, and K. Mai, "A Feature-Based Algorithm for Detecting and Classifying Production Effects", Multimedia Systems, Vol. 7, No. 2, pp. 119-128, March 1999.

[7] M. Osian and L. Van Gool, "Video Shot Characterization", Machine Vision and Applications, Vol. 15, No. 3, pp. 172-177, July 2004.

[8] X. Wang, S. Wang, H. Chen, and M. Gabbouj, "A Shot Clustering Based Algorithm for Scene Segmentation", International Conference on Computational Intelligence and Security Workshops, pp. 259-262, Dec. 2007.

[9] B.L. Yeo and B. Liu, "Rapid Scene Analysis on Compressed Video", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 5, No. 6, pp. 533-544, December 1995.

[10] Z. Rasheed and M. Shah, "Scene Detection in Hollywood Movies and TV Shows", IEEE Computer Society Conference on Computer Vision and Patter Recognition, Vol. 2, pp. 343-348, 2003.

[11] V. Chasanis, A. Likas, and N. Galatsanos, "Video Rushes Summarization Using Spectral Clustering and Sequence Alignment", In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 75-79, 2008. ACM.

[12] D.-D. Le, N. Putpuek, N. Cooharojananone, C. Lursinsap, and S. Satoh, "Rushes Summarization Using Different Redundancy Elimination Approaches", In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 100-104, 2008. ACM.

[13] Y. Liu, Y. Liu, and T. Ren, "Rushes Video Summarization using Audio-Visual Information and Sequence Alignment", In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 114-118, 2008. ACM.

[14] A. Whitehead, P. Bose, and R. Laganiere, "Feature Based Cut Detection with Automatic Threshold Selection", In Proceedings of Content Based Image and Video Retrieval CIVR, pp. 410-418, 2004.

[15] B. Ionescu, V. Buzuloiu, P. Lambert, D. Coquin, "Improved Cut Detection for the Segmentation of Animation Movies", IEEE International Conference on Acoustics, Speech and Signal Processing, vol. 2, pp. 641-644, 2006.

[16] Visibone, "Webmaster Palette", http://www.visibone.com/colorlab, 2006.

[17] S. Lu, I. King, and M.R. Lyu, "Video Summarization by Video Structure Analysis and Graph Optimization", IEEE International Conference on Multimedia and Expo, vol. 3, pp. 1959-1962, June 2004.

[18] Z. Liu, E. Zavesky, B. Shahraray, D. Gibbon, and A. Basso, "Brief and High-Interest Video Summary Generation: Evaluating the AT&T Labs Rushes Summarizations", In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 21-25, 2008. ACM.

[19] W. Bailer and G. Thallinger, "Comparison of Content Selection Methods for Skimming Rushes Video", In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 85-89, 2008. ACM.

[20] F. Chen, J. Adcock, and M. Cooper, "A Simplified Approach to Rushes Summarization", In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 60-64, 2008. ACM.

[21] M. Detyniecki and C. Marsala, "Adaptive Acceleration and Shot Stacking for Video Rushes Summarization", In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 109-113, 2008. ACM.

[22] C. Gianluigi and S. Raimondo, "An innovative algorithm for key frame extraction in video summarization", In Journal of Real-Time Image Processing, vol. 1, no. 1, pp. 69-88, 2006.

[23] Y. Zhai and M. Shah, "A General Framework for Temporal Video Scene Segmentation", IEEE International Conference on Computer Vision, vol. 2, pp. 1111-1116, Oct. 2005.

[24] B.T. Truong, S. Venkatesh, and C. Dorai, "Scene Extraction in Motion Pictures", IEEE Transactions on Circuits and Systems for Video Technology, vol. 13, no. 1, Jan. 2003.

[25] Y. Zhao, T. Wang, W. Hu, Y. Du, Y. Zhang, and G. Xu, "Scene Segmentation and Categorization Using NCuts", IEEE Conference on Computer Vision and Pattern Recognition, pp. 1-7, June 2007.

[26] U. Sakarya and Z. Telatar, "Graph Partition Based Scene Boundary Detection", 5th International Symposium on Image and Signal Processing and Analysis, pp. 544-549, Sept. 2007.

[27] M. Sano, Y. Kawai, and N. Yagi, "Video Rushes Summarization Utilizing Retake Characteristics", In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 95-99, 2008. ACM.

[28] P. Toharia, O. D. Robles, L. Pastor, and A. Rodriguez, "Combining Activity and Temporal Coherence with Low-level Information for Summarization of Video Rushes", In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 70-74, 2008. ACM.

[29] C.W. Ngo, Y.F. Ma, and H. J. Zhang, "Video Summarization and Scene Detection by Graph Modeling", IEEE Transactions on Circuits and Systems for Video Technology, vol. 15, no. 2, pp. 296-305, Feb. 2005.

[30] C.W. Ngo, T.C. Pong, H.J. Zhang, and R.T. Chin, "Motion-based Video Representation for Scene Change Detection", In the International Journal of Computer Vision, vol. 50, no. 2, pp. 127-142, Nov. 2002.

[31] J. Sasongko, C. Rohr, and D. Tjondronegoro, "Efficient Generation of Pleasant Video Summaries", In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 119-123, 2008. ACM.

[32] R. Ren, P. Punitha, and J. Jose, "Video Redundancy Detection In Rushes Collection", In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 65-69, 2008. ACM.

[33] A. Noguchi and K. Yanai, "Rushes Summarization Based on Color, Motion and Face", In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 139-143, 2008. ACM.

[34] E. Kasutani and A. Yamada, "The MPEG-7 Color Layout Descriptor: a compact image feature description of high-speed image/video segment

retrieval", International Conference on Image Processing 2001, vol. 1, pp. 674-677, Greece, 2001.

[35] S. Naci, U. Damnjanovic, B. Mansencal, J. Benois-Pineau, C. Kaes, and M. Corvaglia, "The COST292 Experimental Framework for RUSHES Task in TRECVID 2008", In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 40-44, 2008. ACM.

[36] V. Valdes and J. M. Martinez, "Binary Tree Based On-line Video Summarization", In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 134-138, 2008. ACM.

[37] D. Gorisse, F. Precioso, S. Philipp-Foliguet, and M. Cord, "Summarization Scheme based on Near-duplicate Analysis", In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 50-54, 2008. ACM.

[38] E. Dumont and B. Merialdo, "Sequence Alignment for Redundancy Removal in Video Rushes Summarization", In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 55-59, 2008. ACM.

[39] R. Laganière, R. Bacco, A. Hocevar, P. Lambert, G. Pais, and B. Ionescu, "Video Summarization from Spatio-Temporal Features", In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 144-148, 2008. ACM.

[40] J. Ren and J. Jiang, "Hierarchical Modeling and Adaptive Clustering for Real- time Summarization of Rush Videos in TRECVID08", In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 26-30, 2008. ACM.

[41] V. Beran, M. Hradis, P. Zemcika, A. Herout, and I. Reznicek, "Video Summarization at Brno University of Technology", In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 31-34, 2008. ACM.

[42] P. Toharia, O.D. Robles, A. Rodriguez, and L. Pastor, “A Study of Zernike Invariants for Content-Based Image Retrieval”, In Proceedings of the 2007 IEEE Pacific Rim Symposium on Image Video and Technology, PSIVT2007, volume 4872 of Lecture Notes on Computer Science, pp.944-957, Santiago, Chile, Dec. 2007.

[43] C.-R. Huang and C.-S. Chen, “Video Scene Detection by Link-Constrained Affinity-Propagation”, IEEE International Symposium on Circuits and Systems, pp. 2834-2837, May 2009.

[44] P.P. Mohanta and S.K. Saha, “Semantic Grouping of Shots in a Video using Modified K-Means Clustering”, Seventh International Conference on Advances in Pattern Recognition, pp. 125-128, February 2009.

[45] T. Lin and H.J. Zhang, “Automatic Video Scene Extraction by Shot Grouping”, IEEE International Conference on Pattern Recognition, vol. 4 , pp. 39-42, Sept. 2000.

[46] T. Lin, H.J. Zhang, and Q.Y. Shi, “Video Scene Extraction by Force Competition”, IEEE International Conference on Multimedia and Expo, pp. 753-756, Aug. 2001.

[47] K. Yamasaki, K. Shinoda, and S. Furui, “Automatically Estimating Number of Scenes for Rushes Summarization”, In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 129-133, 2008. ACM.

[48] L.H. Chen, Y.C. Lai, and H.Y. Liao, “Video Scene Extraction Using Mosaic Technique”, 18th International Conference on Pattern Recognition, vol. 4, pp. 723-726, 2006.

[49] L.H. Chen, C.W. Su, H.Y. Liao, and C.C. Shih, “On the Preview of Digital Movies”, Journal of Visual Communication and Image Representation, vol. 14, no. 3, pp.357-367, Sept. 2003.

[50] L.H. Chen, Y.C. Lai, C.W. Su, and H.Y. Liao, “Extraction of Video Objects with Complex Motion”, Pattern Recognition Letters, vol. 25, no. 11, pp.1285-1291, Aug. 2004.

[51] Y. Zhu and Z. Ming, “SVM-based Scene Classification and Segmentation”, International Conference on Multimedia and Ubiquitous Engineering, pp. 407-412, April 2008.

[52] J. Huang, Z. Liu, and Y. Wang, “Joint Scene Classification and Segmentation Based on Hidden Markov Model”, IEEE Transactions on Multimedia, vol. 7, no. 3, June 2005.

[53] H. Bredin, D. Byrne, H. Lee, N.E. O’Connor, and G.J.F. Jones, “Dublin City University at the TRECVid 2008 BBC Rushes Summarisation Task”, In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 45-49, 2008. ACM.

[54] C.J. van Rijsbergen, “Information Retrieval”, Butterworths, London, 1979.

[55] TRECVID 2009, “CBCD Evaluation Plan TRECVID 2009 (v1)”, http://www-nlpir.nist.gov/projects/tv2009/Evaluation-cbcd-v1.3.htm, 2009.

[56] A. Martin, G. Doddington, T. Kamm, M. Ordowski, and M. Przybocki, “The DET Curve in Assessment of Detection Task Performance”, In Proc. Eurospeech ’97, pp. 1895-1898, 1997.

[57] S. White and P. Smyth, “A Spectral Clustering Approach to Finding Communities in Graphs”, In SIAM International Conference on Data Mining, pp. 274-285, 2005.

[58] T. Wang, S. Feng, P. P. Wang, W. Hu, S. Zhang, W. Zhang, Y. Du, J. Li, J. Li, and Y. Zhang, “THU-Intel at Rush Summarization of TRECVID 2008”, In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 124-128, 2008. ACM.

[59] X. Lu, Y.F. Ma, H.J. Zhang, and L. Wu, “An Integrated Correlation Measure for Semantic Video Segmentation”, IEEE International Conference on Multimedia and Expo, vol. 1, pp. 57-60, Nov. 2002.

[60] G. Quenot, J. Benois-Pineau, B. Mansencal, E. Rossi, M. Cord, F. Precioso, D. Gorisse, P. Lambert, B. Augereau, L. Granjon, D. Pellerin, M.

Rombaut, and S. Ayache, “Rushes Summarization by IRIM Consortium: Redundancy Removal and Multi-Feature Fusion”, In Proceedings of the TRECVID BBC Rushes Summarization Workshop (TVS 2008) at ACM Multimedia, New York, NY, USA, pp. 80-84, 2008. ACM.

[61] L. Pevzner and M.A. Hearst, “A Critique and Improvement of an Evaluation Metric for Text Segmentation”, Computational Linguistics, Vol. 28, No. 1, pp. 19-36, March 2002.

[62] A. Hanjalic, R.L. Lagendijk, and J. Biemond, “Automated High-Level Movie Segmentation for Advanced Video-Retrieval Systems”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 9, No. 4, pp. 580-588, June 1999.

[63] M. Scaiano, D. Inkpen, R. Laganière, “Automatic Text Segmentation for Movie Subtitles”, In Proceedings of the Canadian AI 2010 Conference, pp. 295-298, 2010.

Appendix A

Shot Detection Examples

The following are examples of shots detected using the selected method, which was described in section 2.2. The shots were sampled every 25 frames, however some frames were omitted in order to avoid too much repetition. Frames from the same shot are placed on the same line and shots that are not temporally neighboring the previous or next shot are separated by a horizontal line.

It should be noted that because most of the transitions between shots are cuts, the detection of fades and dissolves, among other transitions, were not used in the shot detection algorithm. The results validate this decision as there are few observed incidents of false detections due to those transitions.

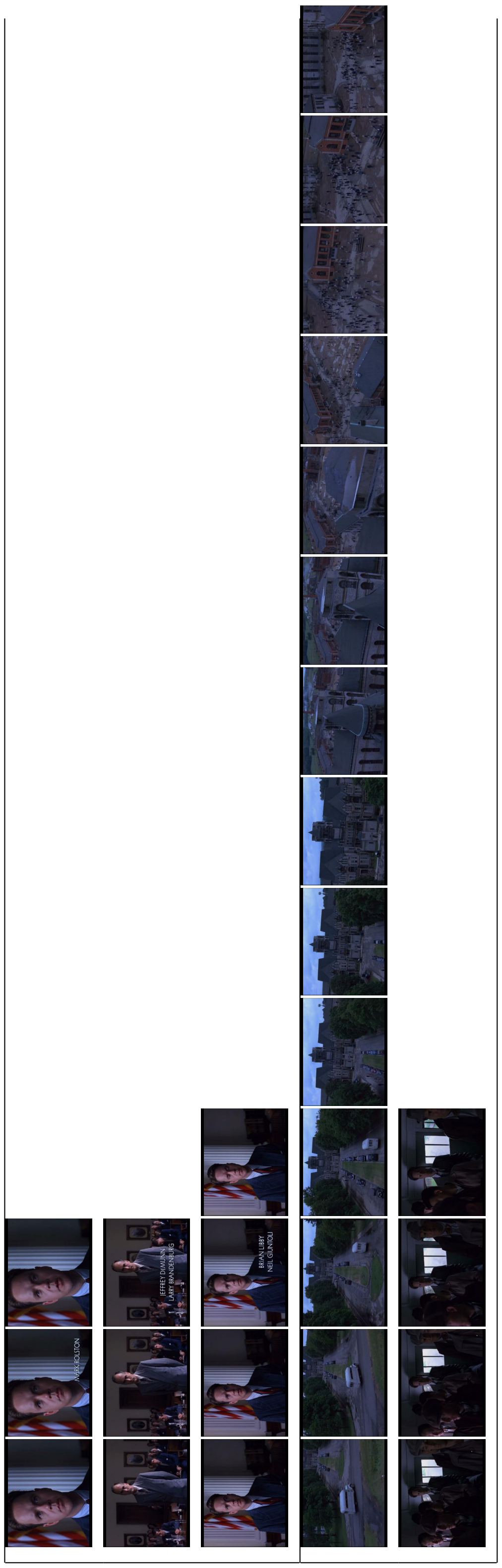


Figure A.1: The Shawshank Redemption shots

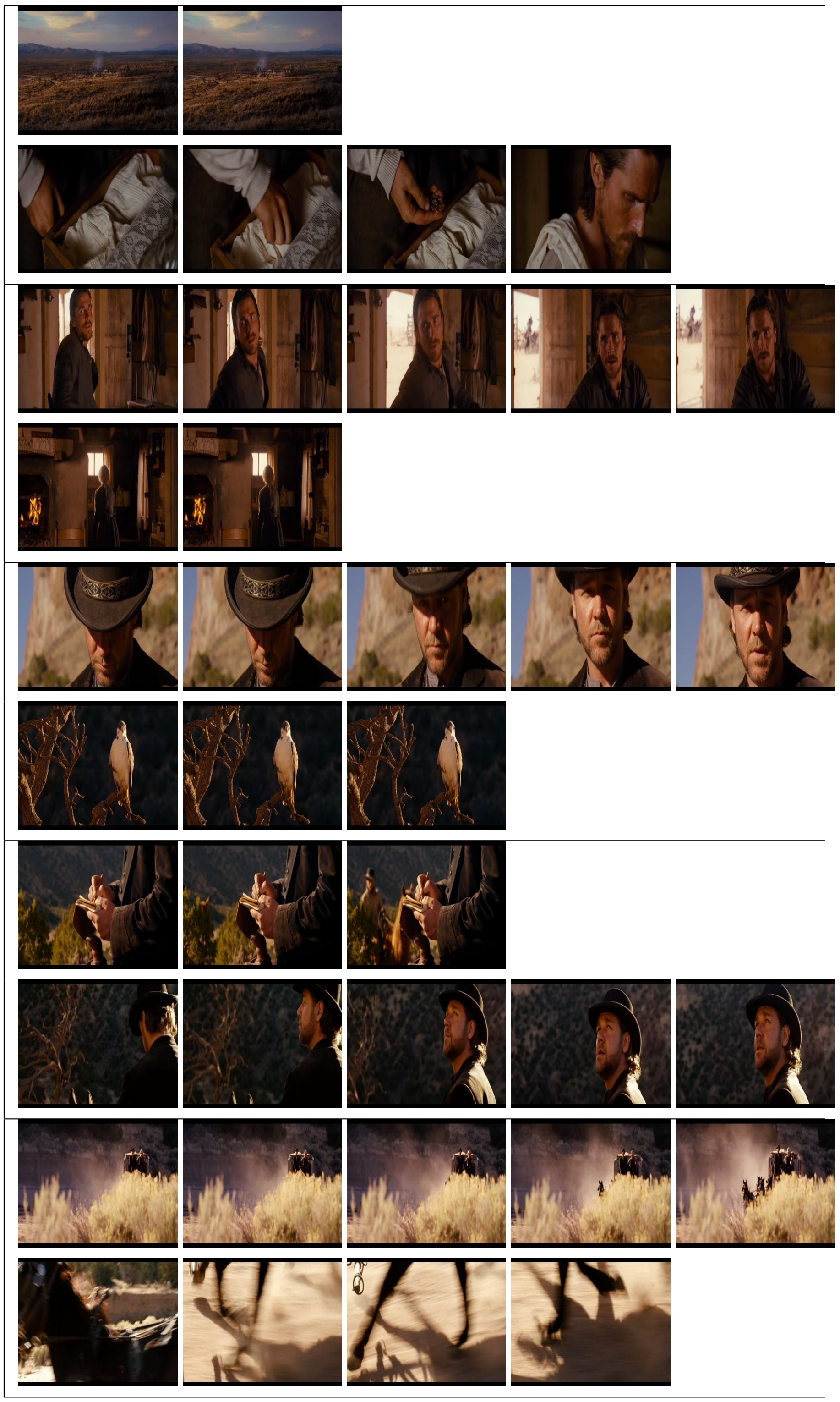


Figure A.2: 3:10 to Yuma shots

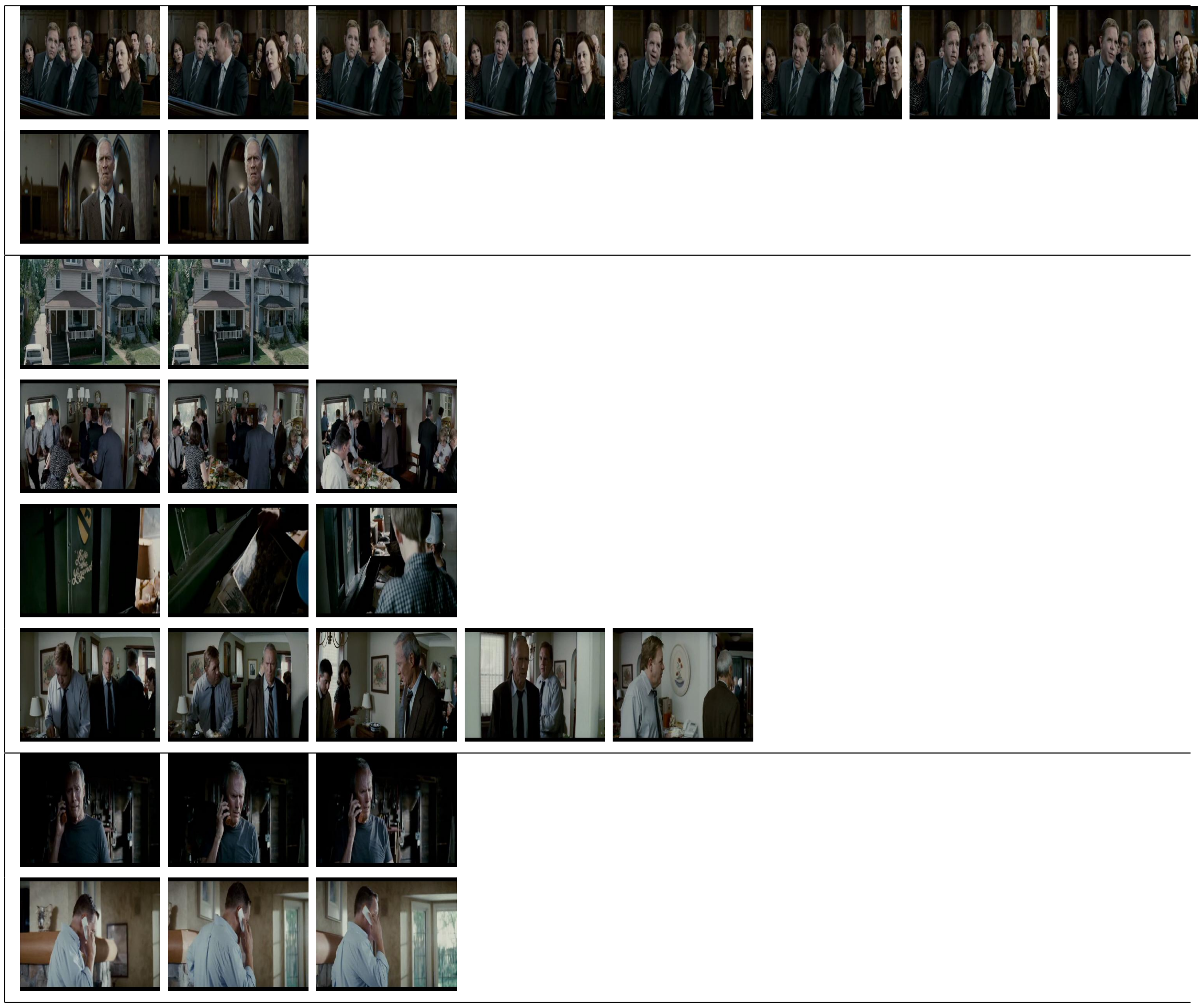


Figure A.3: Gran Torino shots

Appendix B

Scene Detection Results: Bar Graphs

In order to visualize the scene boundary results in a different manner, the boundary locations for the ground truths and the resulting boundaries for each tested method will be displayed as bar graphs for the first 800 shots of each movie. The ground truth's graphs will be presented first and will be followed by the tested method's graphs placed in order of increasing $WindowDiff$ values ($WindowDiff_{GT1} + WindowDiff_{GT2}$).

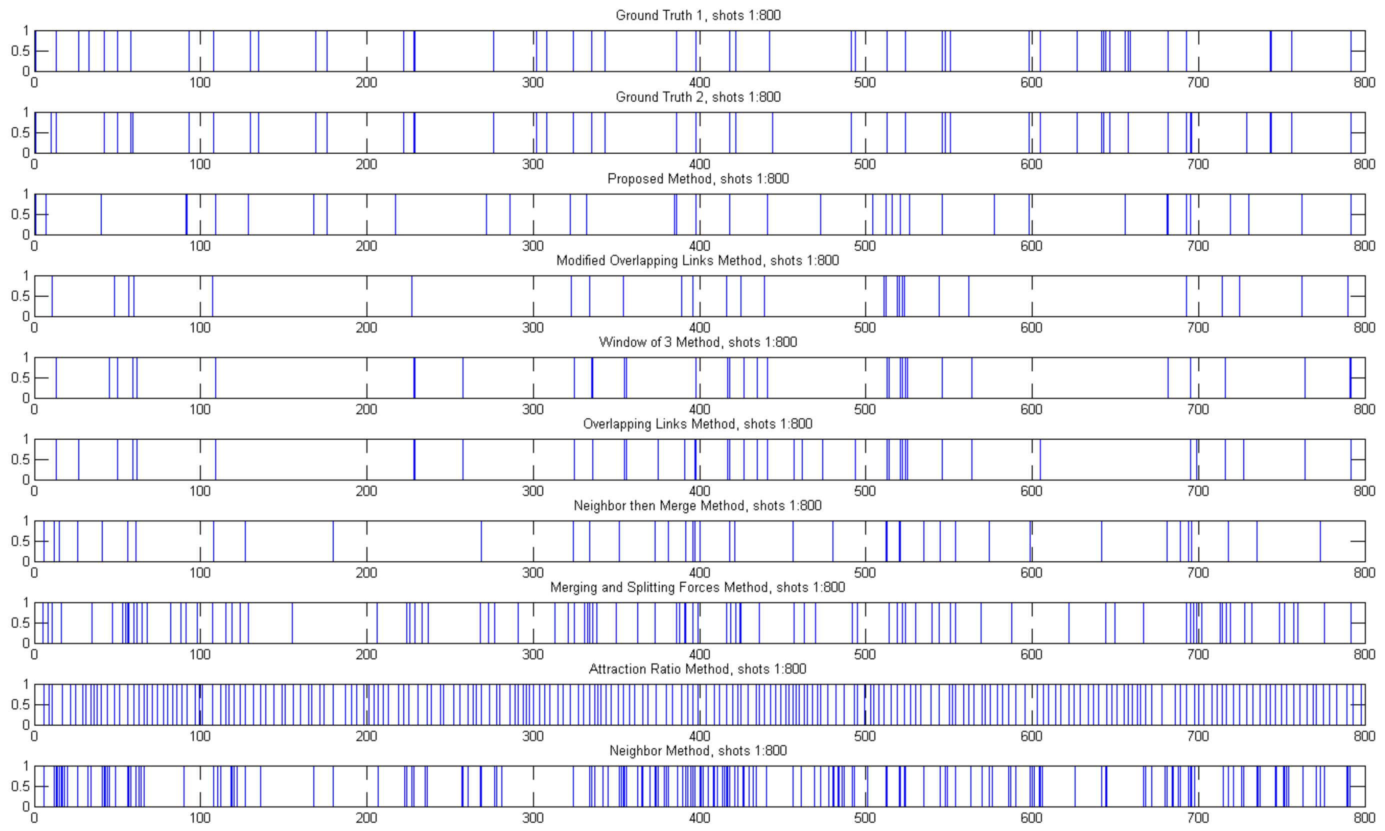


Figure B.1: The Shawshank Redemption Bar Graphs

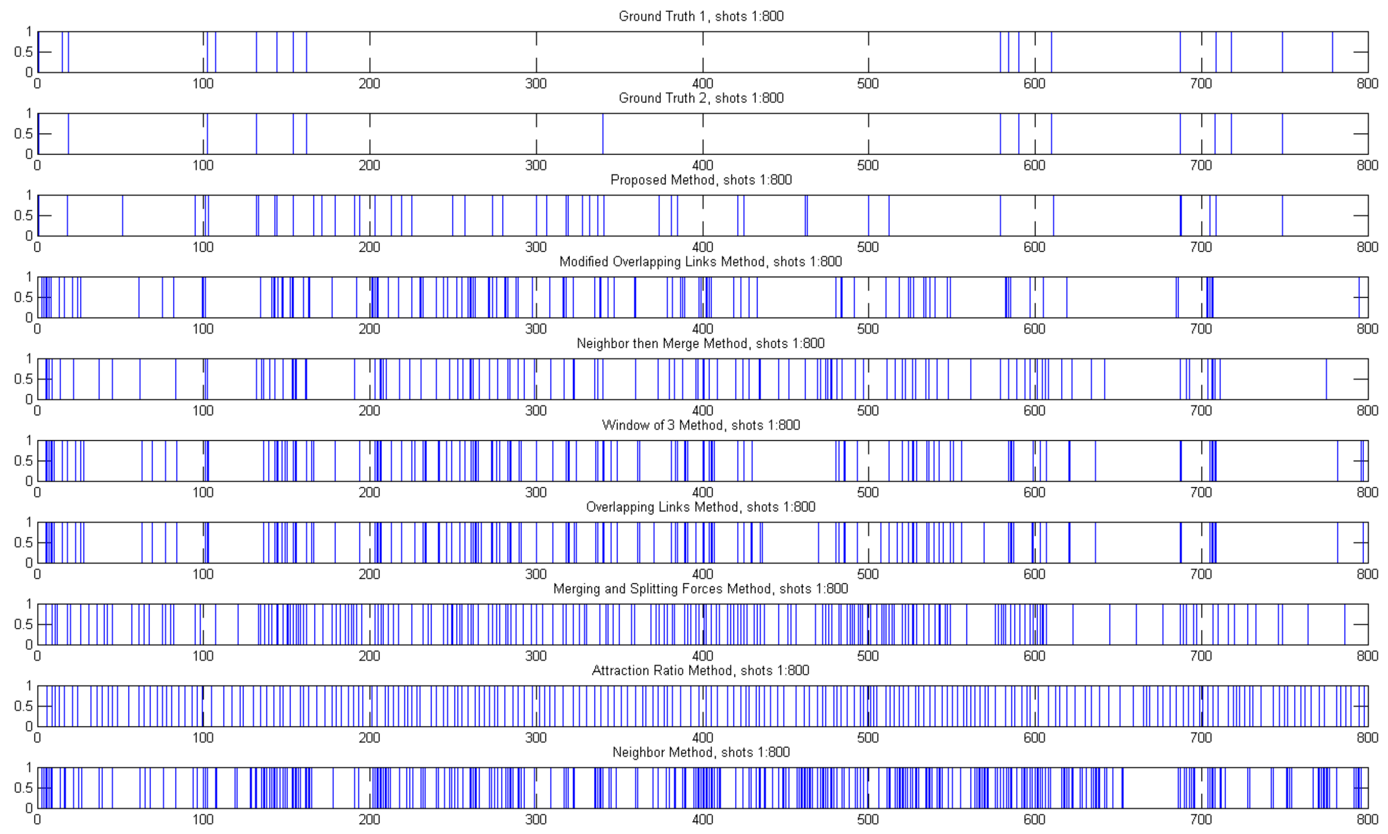


Figure B.2: 3:10 to Yuma Bar Graphs

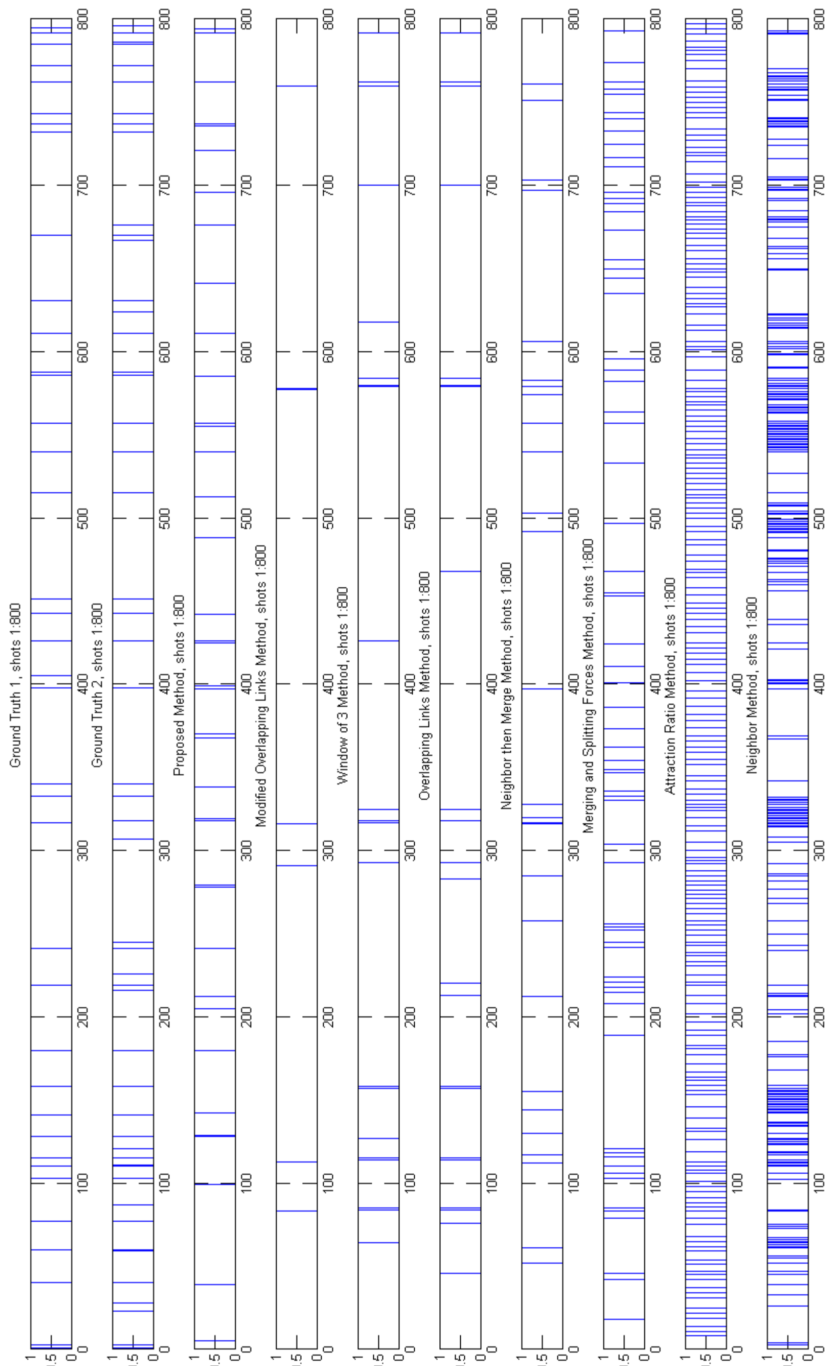


Figure B.3: Gran Torino Bar Graphs

Appendix C

Scene Detection Results: Similarity Maps

In order to better visualize the scene boundary results, the boundary locations for the first ground truth (GT1), the second ground truth (GT2), and for the proposed method will be marked on a similarity map for the first 400 shots.

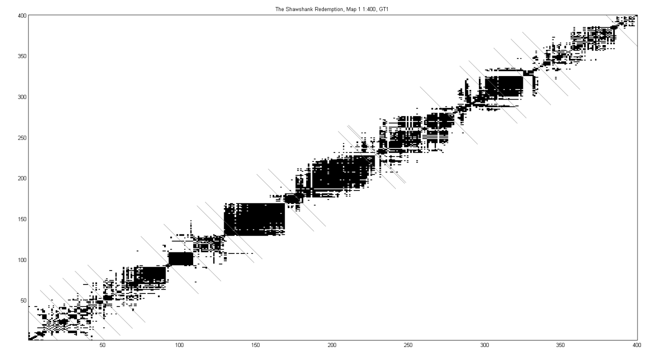


Figure C.1: The Shawshank Redemption GT1

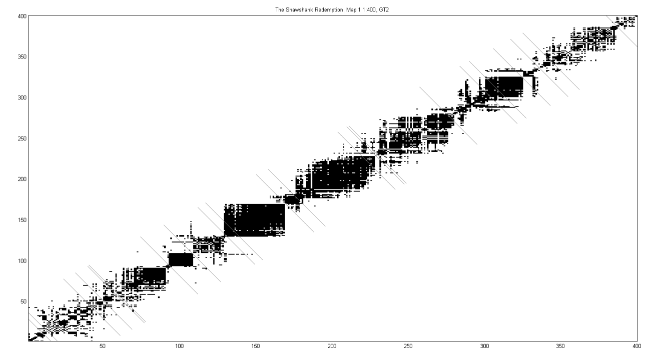


Figure C.2: The Shawshank Redemption GT2

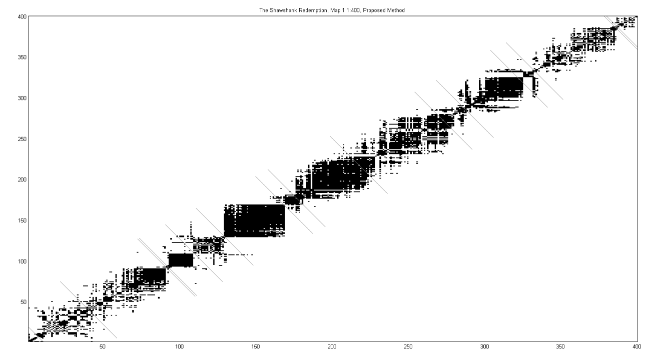


Figure C.3: The Shawshank Redemption Results, Proposed Method

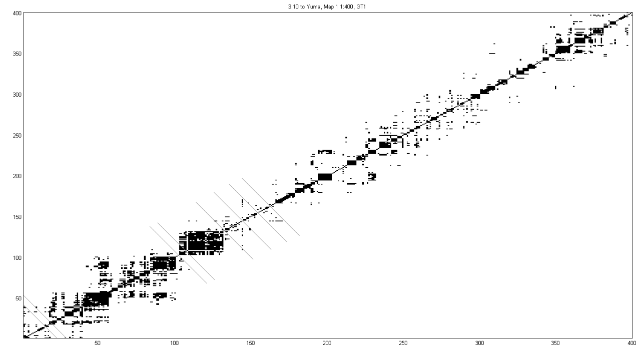


Figure C.4: 3:10 to Yuma GT1

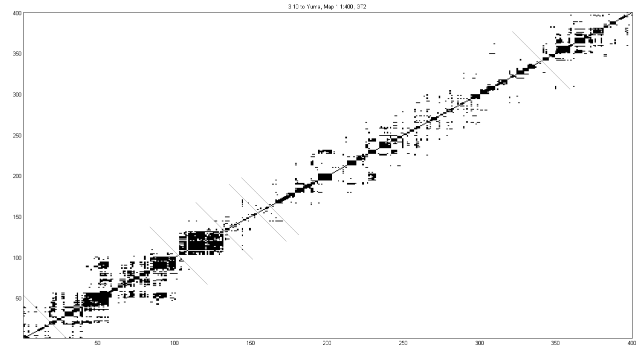


Figure C.5: 3:10 to Yuma GT2

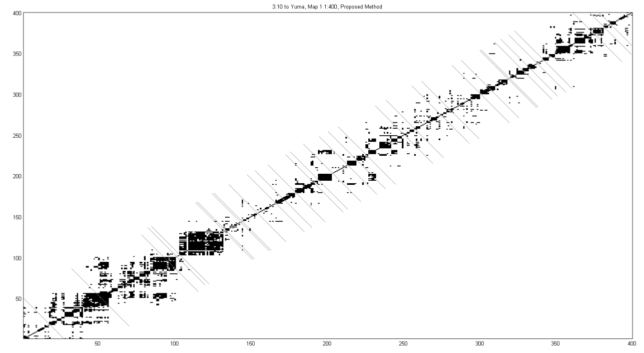


Figure C.6: 3:10 to Yuma Results, Proposed Method

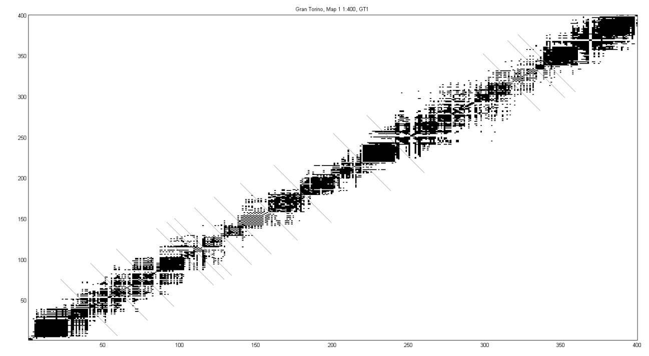


Figure C.7: Gran Torino GT1

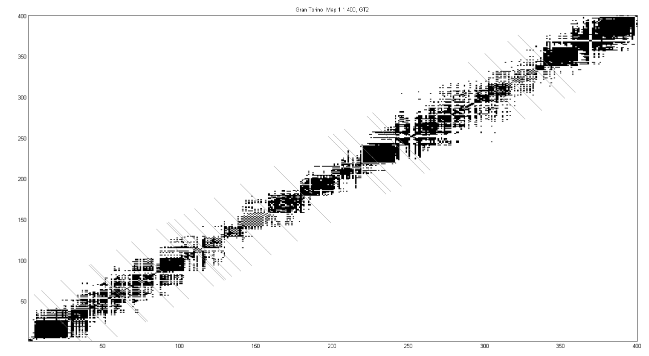


Figure C.8: Gran Torino GT2

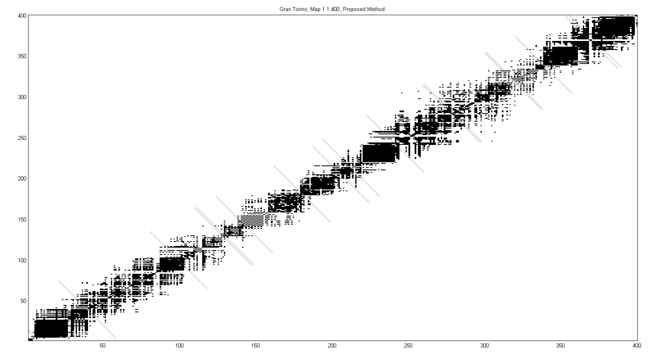


Figure C.9: Gran Torino Results, Proposed Method

Appendix D

Scene Detection Examples

The following are examples of scenes detected using our proposed scene detection method. The first few scenes of each film are shown, separated by a horizontal line. The middle frames of the shots were chosen to represent the shots contained within each scene.

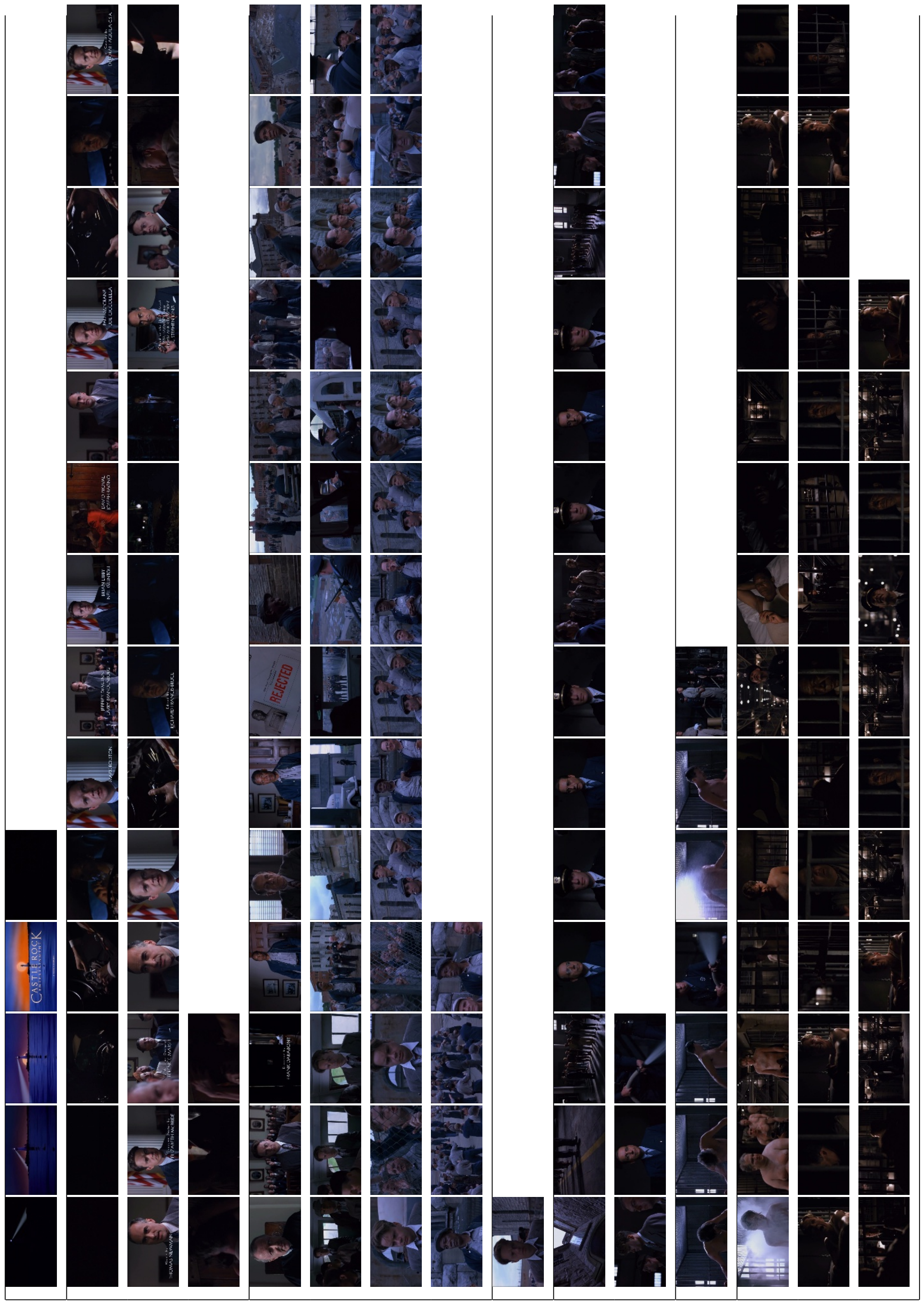


Figure D.1: The Shawshank Redemption Scenes Proposed Method

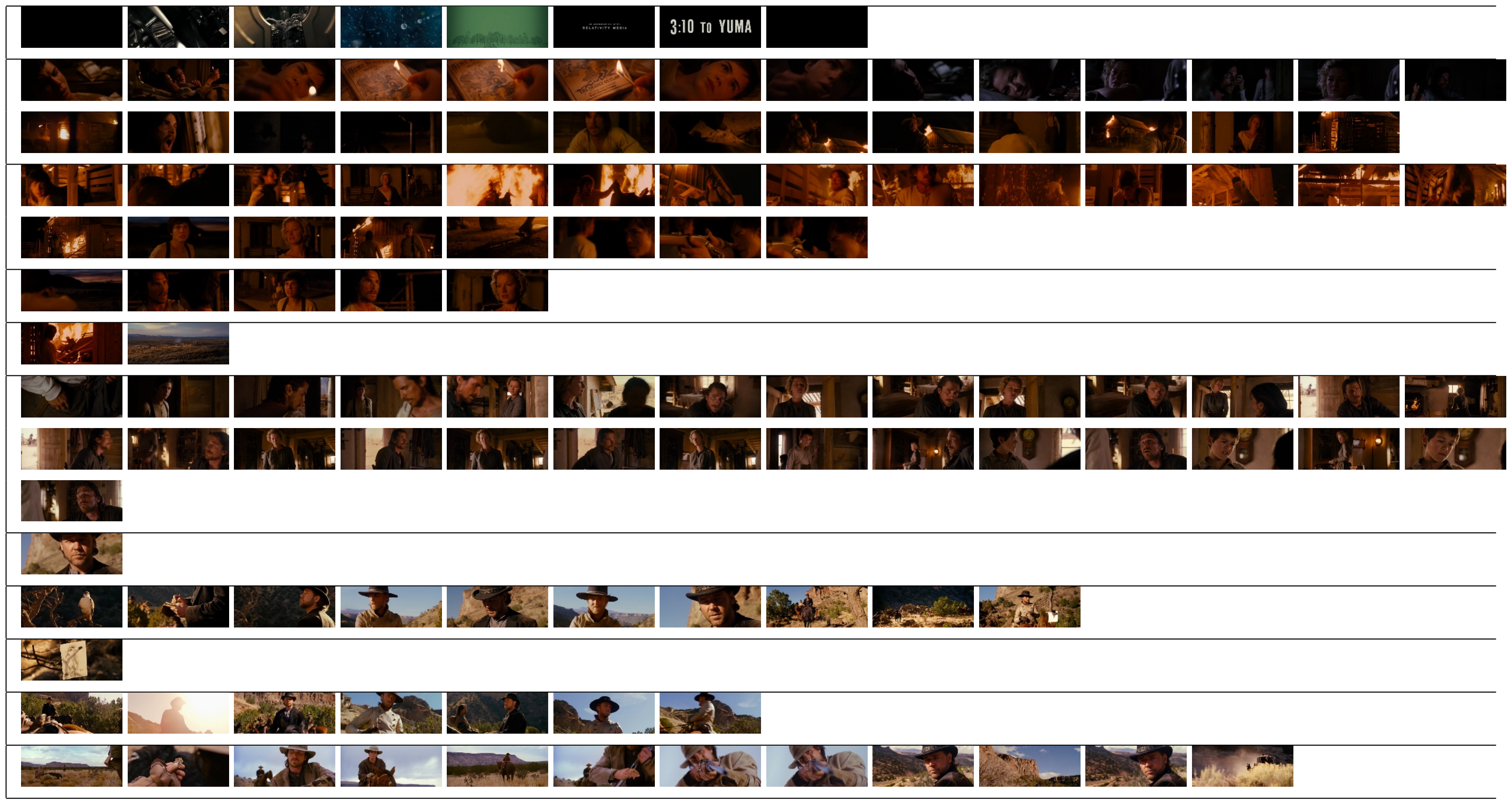


Figure D.2: 3:10 to Yuma Scenes Proposed Method

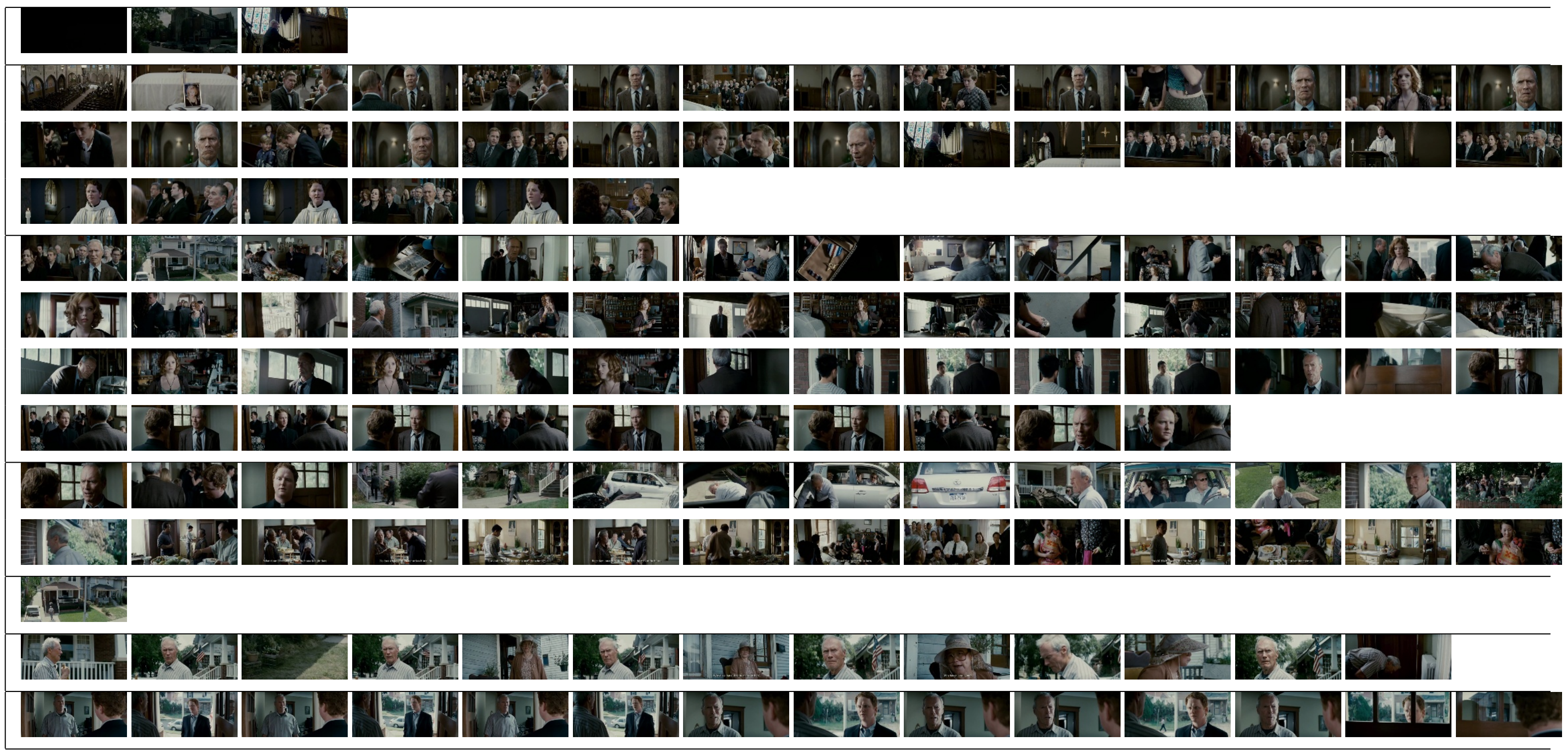


Figure D.3: Gran Torino Scenes Proposed Method

Appendix E

Video Search Tool

The goal of this project was to create a tool that would facilitate the process of browsing and searching through a movie. The Video Search Tool, shown in Figure E.4, was created using Microsoft Visual C++ and Qt, a cross-platform application and UI framework. The following paragraphs will present a step by step guide to using the program.

The first step, before browsing or searching, is to perform video segmentation on the desired video to obtain its video units: its shots and scenes. Open a movie by selecting *File→Open→Video*.

If the shot boundaries for the film were acquired in a previous run, they can be loaded by going to *File→Load Cuts* and selecting the appropriate *.cuts* file. Otherwise, begin shot detection by pressing the first *Go* in the *Segmentation* group box, this will initialize the process. The progress bar indicates at what stage the shot detection is at.

The shot detection process is made up of several steps. To begin, color reduction and color histogram extraction is carried out for each individual frame. Once the histograms are all obtained, the distance is calculated as the average Euclidean distance between the four quadrants of each neighboring frame's color histogram. Next, a second order derivative is applied to the distance vector and a threshold is used to locate the shot boundaries.

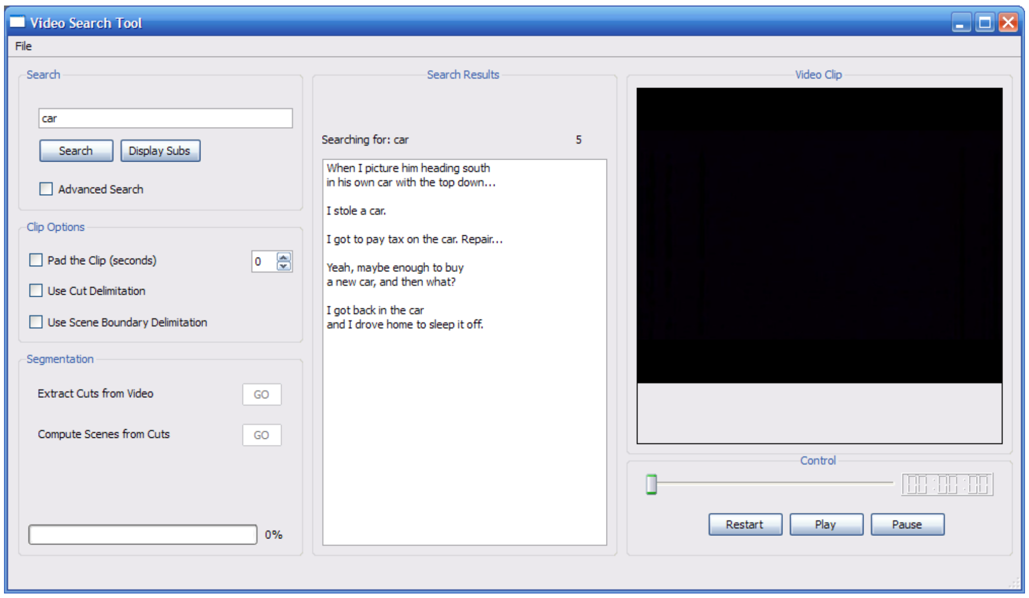
If the scenes were obtained in a previous run, they can be loaded by going to *File→Load Scene Boundaries* and selecting the appropriate *.sceneBoundary* file. Otherwise, if the shot boundaries were obtained in a previous run but the scenes were not, the histograms file must first be loaded before scene detection can be performed by going to *File→Load Histograms* and selecting the appropriate *.histograms* file. To complete the segmentation, scene detection can be performed by pressing the second *Go*.

The scene detection process is done in the following steps. First, the keyframe sets are obtained for every shot by sampling every 10 frames. The similarity map will then be calculated and a threshold applied to obtain a binary map and finally the scenes are located on the map using our proposed method.

After video segmentation, the subtitles can be loaded by going to *File→Open→Subtitle* and selecting the appropriate *.srt* file. The subtitles will then be indexed and stored with their associated time stamps. The indexing needs to be performed once and is saved in a *.index* file. The subtitles will be displayed once the indexing process is finished.

The video is now ready for browsing and searching. To perform a keyword search, simply type it into the search text box and press *Search* or hit Enter on the keyboard. The resulting subtitles which contain this keyword will be displayed in the results box and the number of returned results will also be displayed, Figure E.1. An advanced search can also be done by clicking the *Advanced Search* check box before pressing *Search* or Enter. By performing an advanced search, subtitles that contain synonyms of the keyword will also be added to the results box, Figure E.2.

Clip options can be modified before selecting a subtitle to view. The clip can be extended by selecting *Pad the Clip* and choosing how many seconds to add to the beginning and the end of the clip. Another option is to choose *Use Cut Delimitation*, which if selected when viewing a subtitles, will show the entire shot that contains the chosen subtitle. Similarly, if *Use Scene Boundary Delimitation* is selected, the entire scene, which contains the subtitle, will be viewed, Figure E.3. Cut delimitation and scene boundary delimitation cannot be selected together, since scenes are made up of multiple shots, but padding can be applied to the shot or scene delimitation clips.



car

☐ Advanced Search

☐ Pad the Clip (seconds) 0

☐ Use Cut Delimitation

☐ Use Scene Boundary Delimitation

0%

Searching for: car 5

When I picture him heading south
in his own car with the top down...

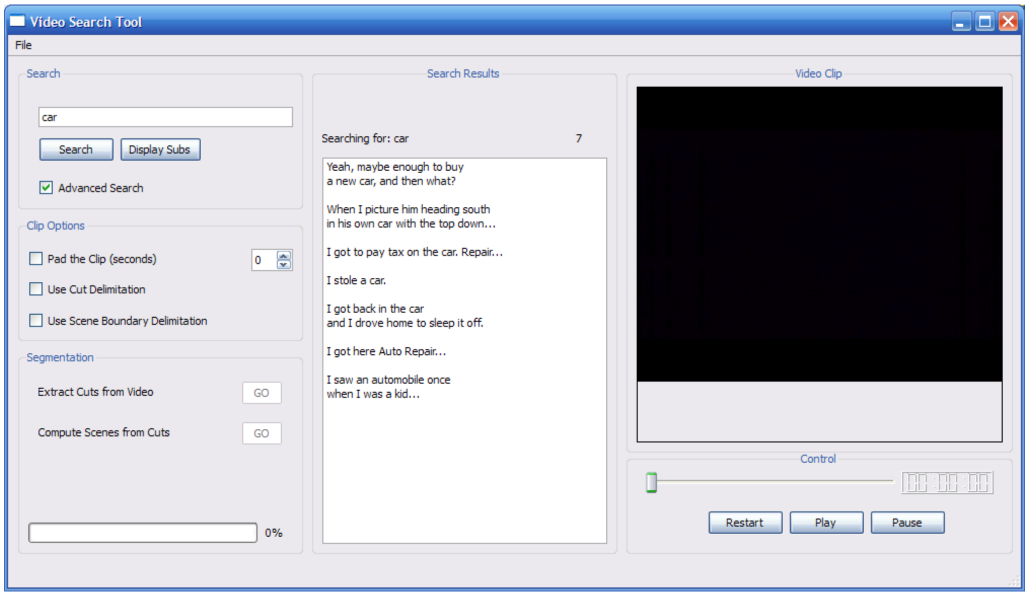
I stole a car.

I got to pay tax on the car. Repair...

Yeah, maybe enough to buy
a new car, and then what?

I got back in the car
and I drove home to sleep it off.

Figure E.1: Basic Keyword Search



car

☑ Advanced Search

☐ Pad the Clip (seconds) 0

☐ Use Cut Delimitation

☐ Use Scene Boundary Delimitation

0%

Searching for: car 7

Yeah, maybe enough to buy
a new car, and then what?

When I picture him heading south
in his own car with the top down...

I got to pay tax on the car. Repair...

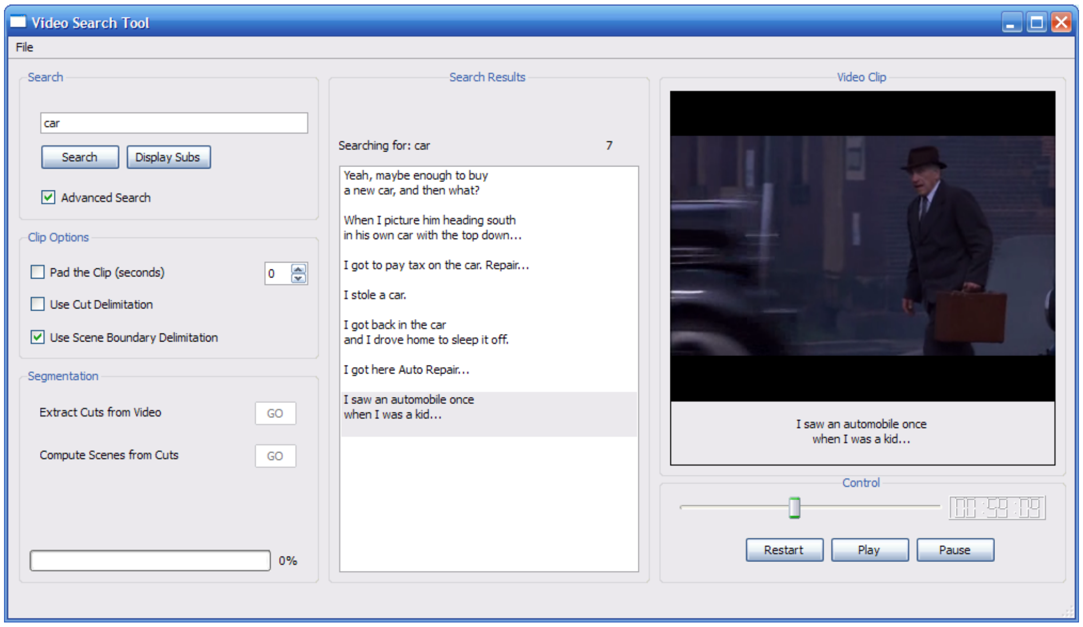
I stole a car.

I got back in the car
and I drove home to sleep it off.

I got here Auto Repair...

I saw an automobile once
when I was a kid...

Figure E.2: Advanced Keyword Search



car

[✓] Advanced Search

[ ] Pad the Clip (seconds) 0

[ ] Use Cut Delimitation

[✓] Use Scene Boundary Delimitation

0%

Searching for: car 7

Yeah, maybe enough to buy
a new car, and then what?

When I picture him heading south
in his own car with the top down...

I got to pay tax on the car. Repair...

I stole a car.

I got back in the car
and I drove home to sleep it off.

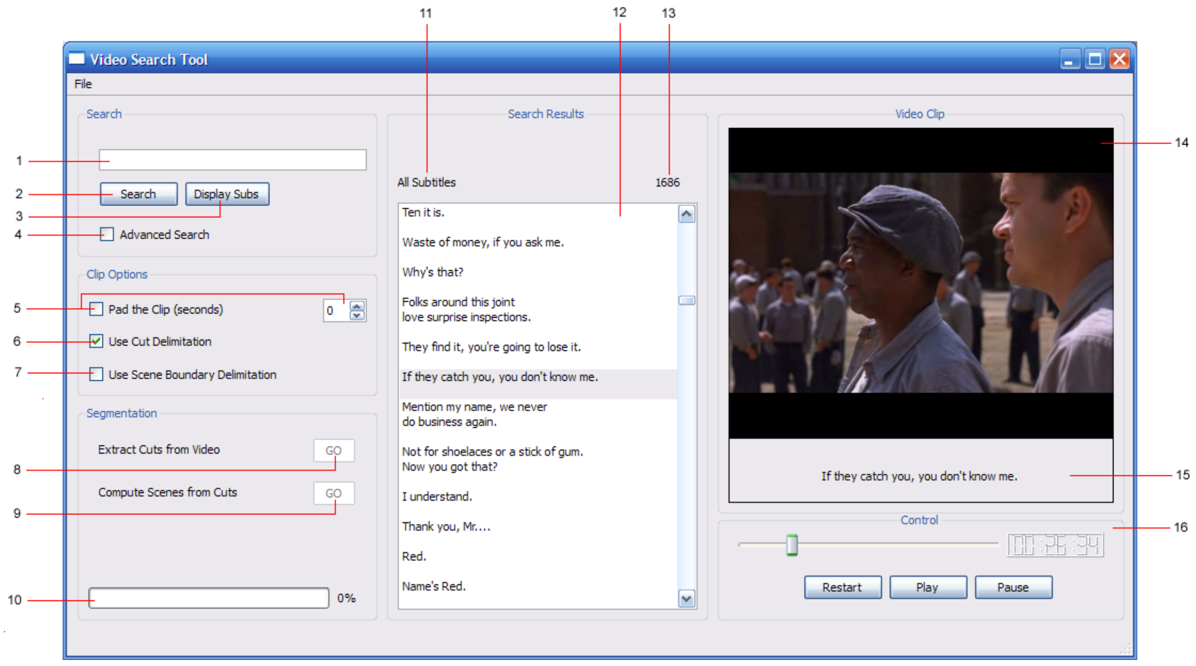
I got here Auto Repair...

I saw an automobile once
when I was a kid...

I saw an automobile once
when I was a kid...

Figure E.3: Clip Options

After selecting the clip options, the clip can be viewed by double clicking the desired subtitle in the *Search Results* box and pressing *Play* in the *Control* box. The clip will automatically pause once the selected delimitation has been reached and to continue viewing from this point, simply press *Play* again. The loaded video can also be browsed in the traditional manner by navigating with the slider and using the control buttons.



Ten it is.

Waste of money, if you ask me.

Why's that?

Folks around this joint
love surprise inspections.

They find it, you're going to lose it.

If they catch you, you don't know me.

Mention my name, we never
do business again.

Not for shoelaces or a stick of gum.
Now you got that?

I understand.

Thank you, Mr....

Red.

Name's Red.

If they catch you, you don't know me.

Figure E.4: Video Search Tool

1. Search Text Box
2. Search button to initiate the keyword search
3. Display Subtitles button to display the entire subtitles list in the results box
4. Advanced Search checkbox to include synonyms of the keyword in the search process
5. Pad the Clip checkbox and spinbox to add time to the beginning and the end of the video clip
6. Use Cut Delimitation checkbox to begin and end the video clip at shot boundaries
7. Use Scene Boundary Delimitation checkbox to begin and end the video clip at scene boundaries
8. Go button to perform Shot Detection
9. Go button to perform Scene Detection
10. Progress Bar indicating the stages of Shot Detection or Scene Detection
11. Indicating what is displayed in the search results box
12. Search Results box containing the results of the search or all of the film's subtitles
13. Number of subtitle results in the box
14. Video
15. Subtitles
16. Video Control box with slider bar and time display, restart, play, and pause buttons