REIHE INFORMATIK
3/97

Automatic Movie Abstracting

R. Lienhart, S. Pfeiffer, and S. Fischer
Universität Mannheim
Praktische Informatik IV
L 15, 16
D-68131 Mannheim

Automatic Movie Abstracting

Rainer Lienhart, Silvia Pfeiffer, and Stephan Fischer

University of Mannheim, Praktische Informatik IV, 68131 Mannheim, Germany
{lienhart, pfeiffer, fisch}@pi4.informatik.uni-mannheim.de

Presented is an algorithm for automatic production of a video abstract of a feature film, similar to a movie trailer. It selects clips from the original movie based on detection of special events like dialogs, shots, explosions and text occurrences, and on general action indicators applied to scenes. These clips are then assembled to form a video trailer using a model of editing. Additional clips, audio pieces, images and text, which are also retrieved from the original video for their content, are added to produce a multimedia abstract. The collection of multimedia objects is presented on an HTML-page.

1 Introduction

Common movie databases such as the internet movie database IMDb (http://uk.imdb.com/) contain mostly (manually input) textual information about a movie. Sometimes, a randomly picked short clip is also included. Although a movie contains a wealth of audio and image information, this is hardly used. The reason is being the difficulty in determining important parts of the audio and image track. Research to this aim is abundant.
In our MoCA (Movie Content Analysis) research project, we have presented algorithms to automatically retrieve content information from the video and audio tracks of digitized films [3][4][7][8][9][13]. We have also presented a video abstracting system called VAbstract [14]. In this paper, we restrict video abstracting to trailer production and expand the result by other retrieved content information like the title or main actors. The demands on a trailer depend on the video genre involved. We concentrate on feature films and adjust the trailer generation process to their needs. Besides, our algorithms analyze only the unchanged material from the original movies.
The information resulting from our video abstracting is contained in different media types like the trailer, other video clips, audio clips, text and images, and comprises therefore a multimedia trailer. It is collected on an HTML-page for presentation on the Web, but may also be stored for retrieval in a multimedia database. Our system is interesting in movie marketing, e.g. in a video-on-demand system. It is even applicable by journalists when retrieving clips relevant to a new film production, e.g. for a documentary about a movie.

The paper is structured as follows. Section 2 relates our work to the achievements of others. Section 3, defines the most frequently used basic terms of this paper. Section 4 presents the concept of our approach, which is described in detail in the subsequent three sections. The produced results are described in Section 8 and Section 9, while Section 10 concludes the paper.

2 Related Work

Research is booming in the field of video abstracting. Most systems, e.g. [1][11][18][21][22][27][28], work on the image track alone and extract key frames only. However, we regard the audio track as a very important source of information, as do Smith et. al. [19] and Taniguchi et. al. [20]. In abstracting, we therefore explore the content of the audio track as well as that of the image track. We are also convinced that in order to get a good idea of the content of a video, it is best to watch a short movie abstract of it. We therefore calculate a moving-images abstract with audio and enlargen this information by the additional extraction of special events. The system presented in this paper extracts clips for their content, an approach distinctly different from all other systems, which extract key frames on a shot basis (except for [19],[14] and [23]).
Prerequisite to the construction of a good abstract is the segmentation of the video into larger semantic units, so-called shot clusters or scenes. The proposed shot-clustering algorithm in Section 6 is similar to the time-constrained clustering proposed by Yeung et. al. in [23] but uses semantically richer features such as color coherence vectors, audio cuts and human dialog detection for clustering.
Similar work to some of the parts of our system has been produced before. However, we introduce some new approaches, especially for scene clustering and clip assembly, and a system which explores all of the content extraction techniques developed within the MoCA project for video abstracting.

3 Definitions

A *video abstract* is defined as “a sequence of still or moving images (with or without audio) presenting the content of a video in such a way that the respective target group is rapidly provided with concise information about

the content while the essential message of the original is preserved" [14].
A *video trailer* is a special type of abstract. It is a short appetizer and outlook on a broadcast, made to attract the attention of the viewers so that they become interested in the broadcast and impatient to watch the presentation. The demands on a trailer differ from film genre to film genre and depend on the specific objective. For instance, the criteria of clip selection for parts of a trailer in a periodic sitcom are usually different from that for a trailer in a feature film. Often a sitcom trailer includes material of or allusions to previous broadcasts.
A *multimedia video abstract* and *multimedia trailer* is a video abstract or trailer, respectively, which consists of different media such as video, audio, animated images, images and text. An example of a multimedia trailer is a WWW page of a famous movie which presents a short video trailer and abundant additional information.

The terms *shot* and *scene* appear frequently in this article. We use them in the sense common to the research field of automatic video content analysis and digital video libraries. A video is described at four different levels of detail: At the lowest, it consists of a set of frames which - at the next higher level - are grouped into shots according to the cuts performed during video production. Consecutive shots are aggregated into scenes based on their pertinence. All scenes together compose the video (see Figure 1). Note that in this paper a video comprises both image and audio tracks. Consequently, each frame consists of two parts: image and audio.

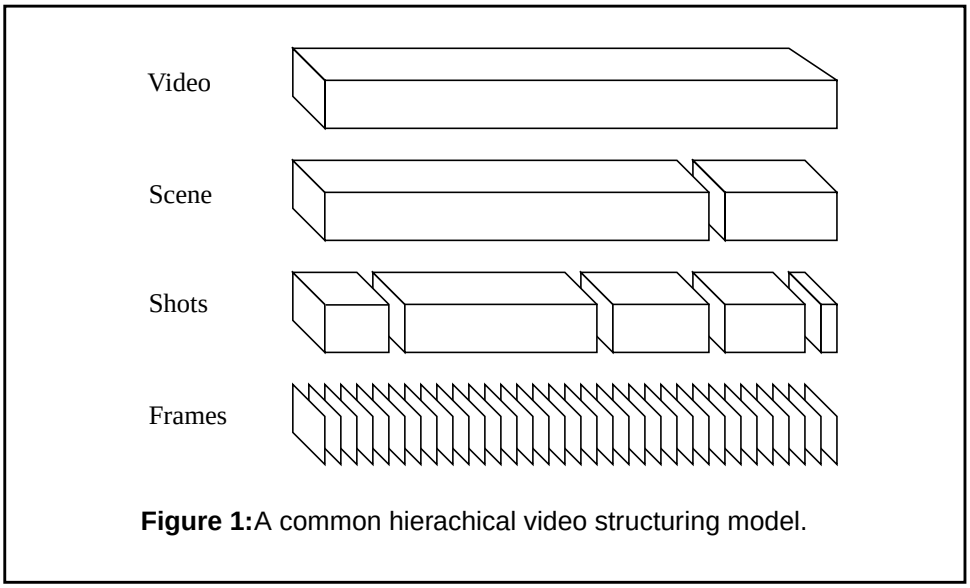

Figure 1: A common hierachical video structuring model.

The term *frame sequence* is used several times throughout the paper. It is formally defined as follows: A *frame sequence* FS_i of a video is a sequence of contiguous frames whose length is at least one frame up to a full video. It is specified by the time interval it covers within the video. That is,

$$FS_i = [t_b^i, t_e^i] \text{ with } t_b^i, t_e^i \in \Re^+, t_b^i \leq t_e^i.$$

For a fixed frame rate, the frame sequence FS_i of a video can also be specified via the start and end frames of the sequence, i.e.

$$FS_i = [f_b^i, f_e^i] \text{ with } {}_b^i, f_e^i \in N, f_b^i \leq f$$

Both representations are used interchangeably based on convenience. The frames are represented in the RGB color space together with the raw audio sample.

A *clip* denotes a frame sequence selected to become an element of an abstract by reason of its specific content. It is extracted from different types of frame sequences such as shots, scenes and frame sequences of special events. Thus, a clip may be contained within a shot or a scene, but it may also overlap the boundaries of either. A video abstract is then viewed as a collection of clips.

4 Conceptual Overview

The algorithmic abstracting approach consists of four consecutive processing steps (see Figure 2). In the first, *video segmentation and analysis*, the input video is segmented into its shots and scenes. Simultaneously, frame sequences with special events such as close-up shots of the main actors, explosions, gunfire and text occurrences during the title sequence are determined. Then, in the second step, *clip selection* and *multimedia parts selection*,

the choice is determined of video segments to compose the video abstract. These video segments are called clips. Also, additional multimedia objects such as images and text are selected for the multimedia abstract. The third step, the *video abstract assembly*, is devoted to the video abstract. It assembles the clips in their final form. This implies determining the order of the video clips, the type of transitions between them and making additional editing decisions. In the last step, *multimedia abstract assembly*, the multimedia objects (video trailer, clips, images, audio pieces and text) are assembled on an HTML-page to form the multimedia abstract.

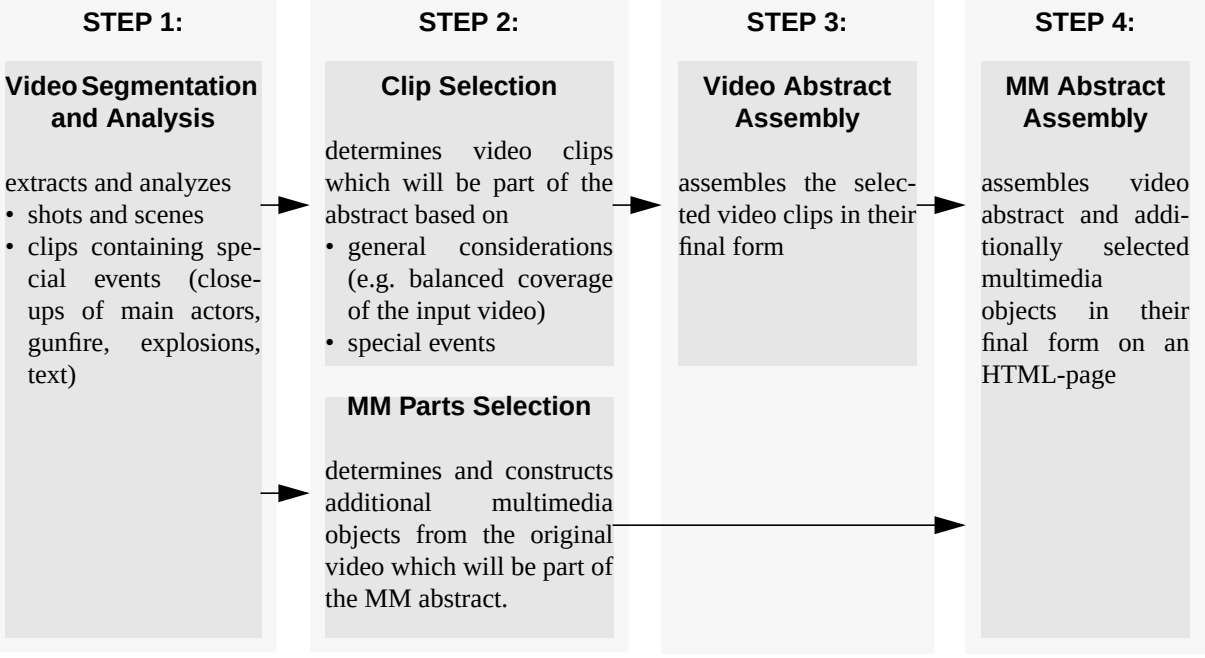

Figure 2: Overview of the four abstracting steps.

5 Video Segmentation and Analysis

In this section we describe the features which serve to determine and analyze shots, scenes and frame sequences containing special events.

5.1 Color Histograms and Color Coherence Vectors

The color contents are an important feature of individual frames and frame sequences. They can be viewed as a compact summary. In practice, the color contents are most often measured by *color histograms*. The color histogram H_i of a frame sequence FS_i is defined as the vector $\langle (h_1), \ldots, (h_n) \rangle$, where h_j specifies the number of pixels of color *j* normalized by the total number of pixels in FS_i. Typically, only a few of the most significant bits of each RGB color component are used to calculate the color histogram. Two color histograms are compared using the root of the sum of squared differences (Euclidean distance).

The *color coherence vector* (CCV) [16] is related to the color histogram. However, instead of counting only the number of pixels of a certain color, the color coherence vector additionally distinguishes between coherent and incoherent pixels within each color class *j* depending on the size of the color region they belong to. If the region (i.e. the connected 8-neighbor component of that color) is larger than a threshold t_{ccv}, a pixel is regarded as coherent, otherwise, as incoherent. Thus, there are two values associated with each color *j*:

- α_j, the number of coherent pixels of color *j* and
- β_j, the number of incoherent pixels of color *j*.

Then the color coherence vector CCV_i is defined as the vector $\langle (\alpha_1, \beta_1), \ldots, (\alpha_n, \beta_n) \rangle$. Before it is calculated, the input image is scaled to 240x160 pixels and smoothed by a Gaussian filter of sigma 1 as proposed in [16]. t_{ccv} is set to 300, and only the two most significant bits of each RGB color component are used. Two CCVs are compared using the Euclidean distance.

5.2 Shot Boundary Detection

Shot boundaries such as hard cut, fades, and dissolves are detected according to the edge change ratio with global

motion compensation as proposed by Ramin Zabih et. al. [26].

5.3 Frontal Face Detector

In most video genres the actors are one essential or even the most important part of a broadcast. This is particularly true for feature films and cannot be ignored when constructing an abstract or trailer. We should know who the main actors are. Thus, a face detector and a method of identifying faces of the same actor within the same shot and across shots/scenes is highly desirable. Moreover, it is a valuable source of semantics.

One of the most reliable *face detectors* in digital images was developed by Rowley, Baluja, and Kanade [25]. Their system recognizes about 90% of all upright and frontal faces while only sporadically detecting non-faces as faces. Therefore, we have recreated their proposed neural network-based frontal face classification system for arbitrary images (e.g. photos, newspapers, and single video frames) as a basis for our frontal face detector in video sequences. To widen the range of detectable faces, our detector also searches for slightly tilted/rotated faces (∓ 30 degree). This is necessary, because the faces of the actors in motion pictures are always moving, in contrast to the faces in typical still images such as portrait and sports team photographs which are usually depicted upright. However, this more general face search increases by a factor of three the number of patterns which have to be tested in each image. To speed up processing, the candidate frontal face locations are drastically reduced by an extremely fast pre-filtering step: Only locations whose pixel colors approximate human skin colors [5] and which show some structure (such as nose, mouth and eyes) in their local neighborhood are passed to the face detector. This pre-filter reduces the number of candidate face locations by 80%. Moreover, only every third frame of the video sequence is investigated.
Each face detected is described by the vector $(t_j^i, x_{pos}, y_{pos}, s, \gamma)$. It specifies the frame t_j^i, in which a face of size s (in pixels) was detected, as well as the x- and y-coordinates (x_{pos}, y_{pos}) of its center and its angle of incline γ.

So far, each detected face is still completely unrelated to every other face in the frame sequence. Consequently, the next task is to put the frames with faces into relation to each other in order to find groups with the same actors. Such contiguous groups of related faces is called a *face group*. We are only interested in the main actors and therefore consider only faces larger than 30% of the frame width (i.e. faces in all varieties of close-up shots). In a first step, faces within shots are related to each other according to the similarity of their position and size in contiguous frames, assuming that these features change only slightly from frame to frame for the same face. This is especially true for dialog scenes. In addition, we dispose of accidental mis-classifications of the face detector by discarding all face groups with fewer than 3 faces and by allowing up to 2 dropouts in the face-tracking process. In a second step, temporally non-overlapping face groups with similar CCVs are merged in order to get as large face groups as possible. This simple grouping algorithm works very well within shots and is computationally cheap. It does not demand complex face recognition algorithms such as described in [6].

CCV comparison is also used to merge temporal non-overlapping face groups of the same actors across shots, resulting in so-called *face sets*. It is capable of merging face groups of the same actors under similar illumination, e.g. a dialog within a scene. However, it cannot guarantee that all face groups of the same actors merge together. An actor's face color varies so much throughout a video. For instance, the CCV of an actor's face at night differs significantly from its CCV in daylight, generally even more than the CCVs of faces of different actors in the same scene. Thus, our grouping algorithm splits the main actors generally into a few distinguished face sets with different actors. Retrieval of only one face set per actor, would require complex face recognition algorithms under unconstrained illumination, a feature not yet available in our system.

5.4 Dialog Detection

A sequence of contiguous shots of a minimum length of two shots is denoted as a *dialog frame sequence* if
(1) at least one face set is present in 75% of the shots and for sequences of at least 4 shots,
(2) the CCV-based shot-overlapping relations between face groups within face sets interlink shots by crossings and
(3) the first and last shots contain at least one face set with shot-overlapping relations.

An example of a detected dialog is shown in Figure 3.

5.5 Text Recognition and Title Extraction

In the opening sequence of a feature film two important things appear in letters: the title and the names of the main actors. Both pieces of information should be contained in the video abstract as well as be available as a

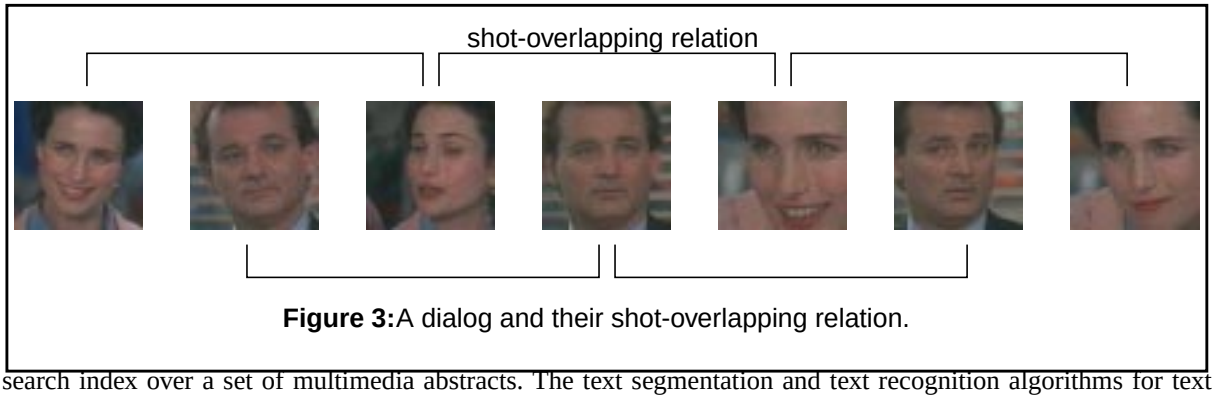

Figure 3: A dialog and their shot-overlapping relation.

search index over a set of multimedia abstracts. The text segmentation and text recognition algorithms for text appearances in digital videos presented in [7] and [9] are deployed to extract the bitmaps of the different text appearances and to translate their content into ASCII for retrieval purposes.

The *text segmentation* step results in a list of character regions per frame and a list of their motion paths throughout the sequence. These motion paths interconnect the regions of same characters over time [7]. In order to be able to extract the bitmaps of the title and the names of main actors, character regions within each frame are clustered into words/text lines based on their horizontal distance and vertical alignment. Next, those clusters which are connected via the motion path of at least one character region are combined into a text line representation over time. For each text line representation, a time-varying (one per frame) bounding box is calculated. The content of the original video framed by the largest bounding box is chosen as the representative bitmap of the text line representation. This method works correct under the following assumptions:

- the text line is stationary or moving linearly and
- all characters of a cluster are contained in the segmented character regions within at least one frame.

If these assumptions hold true, the largest bounding box will enclose the text and we perform a *text recognition* on it. In our experience these assumptions are true for most of the opening sequences in feature films.

Assuming that the *title* distinguishes itself by being centered on the screen and either by the largest size of characters in the opening sequence or, if no such can be found, by the length of the text line, the title bitmap candidate is determined. The font size is largest for short titles, while for longer titles the font size used decreases with increasing title length.

We do not analyze the closing sequence containing title and credits since - in contrast to TV production - in most movies these are illegibly small in their video version and thus can neither be segmented nor recognized by our algorithms.

5.6 Audio Cuts and Audio Feature Vectors

One way to cluster shots into meaningful units like scenes is by analysis of the audio track. We determine audio cuts, i.e. time instances which delimit time periods with similar sound, in order to explore the similarity of the audio track of different shots. To that aim, we compare frequency and intensity composition of the audio frames of the shots. These parameters are extracted via Fourier transform, where we choose a certain analysis window size (between 100 and 200 ms) and a certain windowing function (usually a Hamming window) as parameters. The resulting complex values are converted into real values by calculation of decibels (other methods like simple addition, addition of squares or calculation of amplitude are possible). Such a collection of real values for a specific analysis window is called an *audio feature vector* x_t.

In order to analyze a frame sequence, consecutive audio feature vectors are calculated on overlapping analysis windows (usually the overlap amounts to 1/4 of the analysis window's size). While no cut is found, calculation of a *forecasting vector* is performed by exponential smoothing Forecasting: $F_{t+1} = \alpha x_t + (1-\alpha)F_t$, where $0 < \alpha < 1$ is called the smoothing constant (here set to 0.2). The forecast is the weighted sum of the last observation and the previous forecast. The forecasting vector therefore contains representative feature values from previous feature vectors. The speed at which the forecasting vector adapts to new feature values is controlled by the smoothing constant. The forecasting vector compares the next feature vector with the afore analyzed feature vectors by using the Euclidean distance.

The decision about an *audio cut* is based on the difference between a new feature vector and the forecasting vector. We have two difference thresholds: a high threshold which directly determines a cut (because there was a significant difference) and a lower threshold which determines similarity (i.e. the difference is hardly significant). If, however, too many consecutive feature vectors are just similar, we also deduce a cut.

As calculation of an audio cut refers to about 100 to 200 ms, an audio cut is related to several frames. Therefore, an audio cut is attributed to a frame sequence and denoted by $AC_j = FS_i$.

5.7 Audio Events

Attractiveness to the user is an important design criterion for a trailer. Action films often contain *explosions* and *shots* which can be recognized automatically. Indicative of action, these elements have to be included in a trailer. The algorithm to recognize these events is the following:

1. For a time-window calculate distribution of indicators loudness, frequencies, pitch, fundamental frequency, onset, offset and frequency transition.
2. Compare indicator distributions with database set of precalculated distributions of explosions and shots.
3. Recognize shot or explosion if distribution can be found in database.
4. Move time-window forward. Goto 1.

Pitch is calculated by the method of Meddis and Hewitt [10]. Onset and offset are a measurement of how fast a signal level changes. Considering a shot, the loudness and frequency levels both change rapidly, resulting in a high onset value. Frequency transitions measure glides in frequency as angles. Explosions and shots show steep angles of frequency glides. The exact theory of and weights for the different indicators can be found in [13]. The detection process for an explosion can be seen in Figure 4. The y-axis shows the Euclidean difference between the indicator distribution of the actual film and a pre-calculated distribution of an explosion. The location of the explosion is clearly visible.

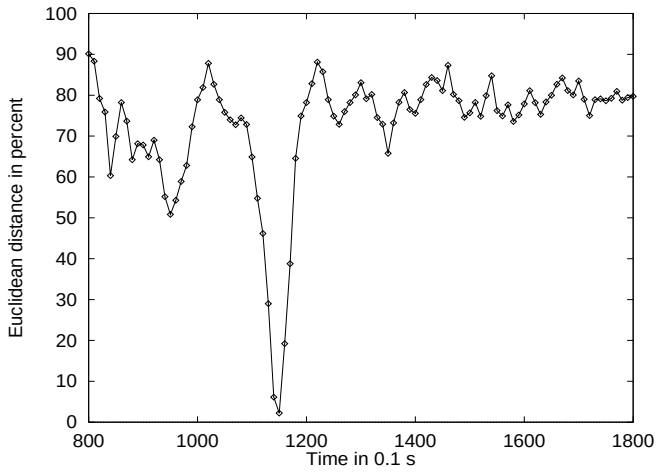

Figure 4: Recognition of explosion.

6 Shot Clustering

Shots are defined as “one uninterrupted image with a single static or mobile framing” [2]. That means a shot is a frame sequence $S_j = FS_i$ in which the contiguous frames do not differ very much. Usually, several (contiguous) shots are used to build one larger semantic unit in the video. They are then called *scene*, *act* or just a *cluster of shots*. The clustering of shots depends on the criterion which stands in the foreground. Here, we describe a general clustering algorithm for segmenting a feature film into its larger semantic units. We use it only to determine scenes, but other clustering approaches are possible. The scenes are used in Section 7 as the basic unit to determine clips so as to generate a balanced trailer. These, together with those of special events, make up the trailer clips.

6.1 General Algorithm

At first, for each shot S_j, the required feature values $F_k(S_j)$ are determined (possible features have been presented in Section 5). These values are then used to compare and group shots. Therefore, for each feature F_k we must define a metrics d_k, which results in a distance measure of shots such that we can determine the amount of similarity between them. This enables shots to be clustered according to one or several features.

If we choose to cluster shots based only on one feature F_k, then shots are grouped together based solely on this feature’s distance metrics: Shots S_{j_1} and S_{j_2} are grouped together, if $d_k(F_k(S_{j_1}),F_k(S_{j_2})) < threshold_k$.

If we choose to cluster shots based on several features, we first determine separately the distances of each feature and then combine the results by a function. For example, we may determine separately clusters based on each feature and then apply a simple "OR"-function:
If $[d_{k_1}(F_{k_1}(S_{j_1}),F_{k_1}(S_{j_2})) < threshold_{k_1}]$ or $[d_{k_2}(F_{k_3}(S_{j_1}),F_{k_2}(S_{j_2})) < threshold_{k_2}]$, S_{j_1} and S_{j_2} belong to one cluster.

Another possibility would be to define a new metrics $d_{k_{1,2}}$, interweaving the feature values of F_{k_1} and F_{k_2}.

We have implemented a heuristics to determine shot clusters. It is described in the following paragraph. Other clustering algorithms are of course possible and may be included into our system in the future.

6.2 Scene Determination

In order to determine scenes, we had to consider what constitutes a scene and have come up with the following heuristics:

1. Sequential shots with very similar color content usually belong to a common scene, because they share a common background. The color content changes more dramatically at the end of a scene than within it. A change of camera angle usually retains the main background colors.
2. Shot boundaries not identical to a (very close) audio cut do not coincide with a scene boundary, since the audio contents before and after the shot boundary are similar. In different scenes, however, the audio differs significantly different. Hardly any change in audio implies that no change of scene has occurred.
3. A third heuristic composes together consecutive shots which are interweaved by a dialog.

Therefore, the algorithm to determine scene boundaries is based on three features: color histograms (See "Color Histograms and Color Coherence Vectors" on page 5.), audio cuts (See "Audio Cuts and Audio Feature Vectors" on page 7.) and dialogs (See "Dialog Detection" on page 6.). The algorithm proceeds as follows:

1. If for two consecutive shots S_i and S_{i+1}:

   $\sqrt{CCV_i^2 + CCV_{i+1}^2} < threshold_{CCV}$, then these shots belong together.
2. If for a shot boundary between shots S_i and S_{i+1}, described by the delimiting frame f_i, there is no audio cut $AC_j = [f_b^j, f_e^j]$ such that $b - c \le i \le e + c$, where c is a constant describing the closeness of the audio cut to the shot boundary, then shots S_i and S_{i+1} belong together.
3. If for a shot boundary between shots S_i and S_{i+1}, by the delimiting frame f_i, there is a dialog frame sequence $DFS_j = [f_b^j, f_e^j]$ such that $b \le i \le e$, then shots S_i and S_{i+1} belong together.

For all shots and shot boundaries, the above three steps must be tested in order to determine which of the shot boundaries coincide with scene boundaries.

7 Trailer Generation

Trailers in feature films are made to interest people in a movie without revealing the story. Such trailers require inclusion of eye-catching, curiosity-causing video clips into the abstract.

7.1 Basic Properties

Before we can address the question of how to use the information from the video segmentation and analysis to generate a feature film trailer, the question of its basic properties must be answered. Because there is no general rule, which clips have to be included in such a trailer, we decided to deem the following necessary:

(1) *Important objects/people*: The most important objects and/or actors appearing in the original should appear in the trailer. Especially the appearance of the starring actor(s) is important, because very often people base their decision on which performance to watch more on the main/starring actors than on the story. If these are their favorite movie stars, they will want to see the film in any case.

(2) *Action*: If the film contains explosions, gunfire, car chases or crime, some of these should be in the trailer. They attract attention and make viewers curious as to who/what was blown up; who shot whom? To answer these questions, they must see the movie.

(3) *Dialog*: Short extracts from important dialog scenes with a starring actor stimulate the watchers' fantasy and guessing about the story. Hence, some of them should be part of the trailer.

(4) *Disguise end*: The end of the movie is never revealed, as the story's thrill is often released there.

(5) *Title and title music:* Title and parts of the title music should be contained in the trailer. Optionally, the names of the main actors should be shown.

(6) *No abstract/summary*: A trailer is not a video abstract in the sense that it preserves the essential message of the original full-length feature film. Nevertheless, it should consist of a balanced set of some "highlights" from different parts of the feature film.

7.2 Clip Selection

We now design the clip selection algorithm. It extracts various short clips from a full-length feature film according to the properties stated in (1)-(6). Their assembly is postponed to the next section. To this objective, the following questions must be answered:

1. What is the target length of the trailer?
2. Which scenes should the trailer contain?
3. Which sub-sequences of a selected scene should be in the trailer?

The abstracting algorithm starts with a user interactions: The user (e.g owner of a video-on-demand server or user of a video-on-demand preview tool) needs to specify the target length of the trailer. Typical values are 30 to 120 seconds.

Next, the abstracting algorithm must determine computationally the scenes from which short subsequences should be selected for the trailer. Initially, all scenes of the feature film are in the scene candidate set. Then, in obedience to rule (4) all scenes from the last 20% of the film are removed. Before the scene candidate set can be narrowed down further, one must be aware that scene and sub-sequence selection are somehow related, since it makes no sense to select a scene to be a portion of the trailer without having any specific appealing sub-sequence in mind. At the same time, the decision process is restricted by the limited semantics of features which can be extracted automatically and reliably from the feature film (see Section 5). Therefore, two different mechanisms are used to select scenes from the scene candidate set. Both have in common that the decision is made simultaneously for a scene and its appealing sub-sequence(s).

On the one hand, specific events such as shots, explosions, cries, close-up shots, dialogs of main actors and appearance of the title are located. They can be extracted automatically and are suited to attract people's attention (properties (1)-(3),(5)). Thus, the associated scenes and sub-sequences are marked for inclusion in the trailer up to a certain percentage of its length. This share of special event in percent is a parameter the user must specify. In our experiments it was set to 50%. If the total length of located special events is longer than the proportional trailer length, scenes and clips are chosen uniformly and randomly from the different types of special events up to their proportional trailer length. This guarantees the greatest variety of special event types and increases the attractiveness of the trailer.

On the other hand, property (6) demands a balanced abstract. Therefore, the remaining time in the trailer (i.e. 50% to 100%) should be filled with shots (as parts of a scene) which are well distributed over the length of the feature film. The question to be answered is which shots in the original scenes should be included in the trailer or equivalent to that, how the importance of a scene can be measured quantitatively.

We determine the importance of scenes according to the criteria of action: the amount of action within each shot is determined, which leads to a ranking of the shots. The ranking is needed to gain knowledge about the relevance of a shot in the selection process. The user may specify whether he would like to have the trailer filled with shots with a lot of action or little action.

The principal idea we use in order to determine the amount of action contained within a shot is the analysis of the distribution of a set of indicators including color, motion, loudness, frequency and perceptional information (for further details see [3]). In previous work, we have used those indicators to recognize film genres (see [3] and [4]). There, a grouping process which determines the homogeneity of a set of similar films yields the unknown weights of the indicators. In this paper, we use the weights of the indicators of action-films (see Figure 5) to classify scenes accoring to action. The indicators *loudness*, *onset* and *offset* obtain high values for action-films. This could be expected, as action films often contain peaks in the audio spectrum as well as shots and explosions..

The indicator distribution of the video is calculated for all two subsequent time-windows (typically a per-frame basis for video and 30 ms for audio). In the case of a great amount of action, the Euclidean distance between the distribution and the indicator weights of action-films will be low, otherwise high. Using the difference metric, a quantitative importance for the abstract can be assigned to the shots. Possible strategies are the use of a nearest-neighbor-criterion, the calculation of distances to other film sets and the homogeneity in a certain set of similar films [3]. The difference metric D_t used is expressed as

$$D_t = \sum_{i=1}^{N} w_i (I_{i,t+1} - I_{i,t})$$

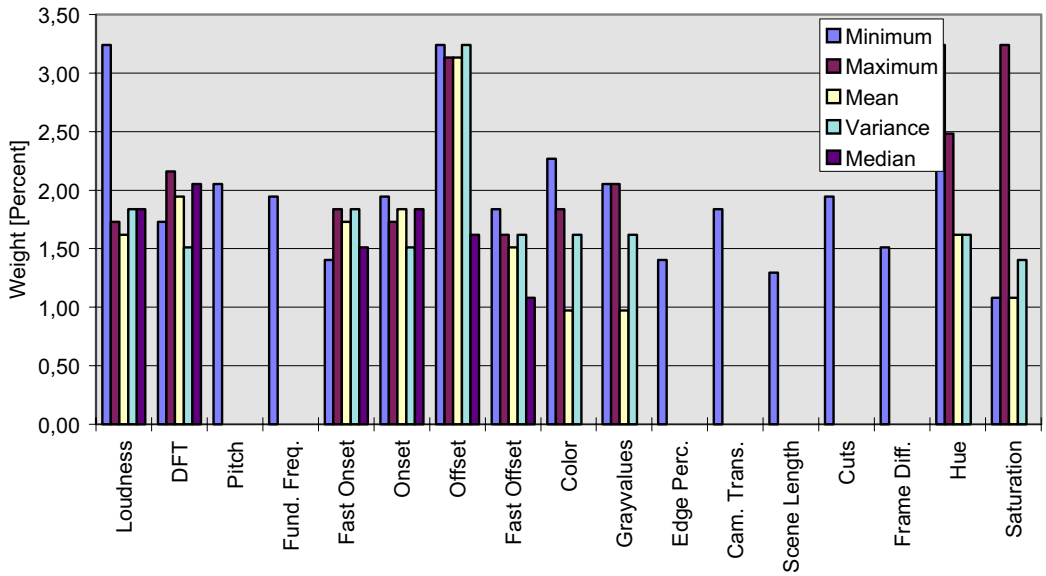

Figure 5: Indicator weights for a set of action-films

where N is the number of indicators i, w_i the indicator weight and $I_{i,t}$ the value of indicator i at time tThe importance of a specific shot within a scene can now be measured in three ways. The use of the calculated set of difference metrics without further transformation yields an *absolute importance* of a shot. The maxima of the difference metrics in a scene are indicative for action as the distribution of many indicators changes from one time window to the next. The disadvantage of this approach is that the importance of sequences of very small action followed by such of only a little bit more cannot be recognized. Often these sequences are important. For this reason we also use the *relative importance* of a shot expressed as the ratio of two subsequent difference metric values. The relative importance exactly solves the described problem. Comparing the absolute and the relative importance, it is obvious that only the use of both yields good results. We therefore combine both using the standard deviation of both importance values as a weight. In case of high action, the change ratio of the absolute importance is often very high while that of the relative importance is low which results in a high weight for the absolute importance. The contrary is the case for little action. The mathematical definition of the combined importance I_t at time t is hence

$$I_t = \frac{\sigma_A}{\sigma_A + \sigma_R} \cdot D_{A,t} + \frac{\sigma_R}{\sigma_A + \sigma_R} \cdot D_{R,t}$$

where σ_A is the standard deviation of the absolute differences and σ_R that of the relative differences in a scene.

The selection of a shot to be included in the trailer is then quite simple. For a scene the highest value I_t is determined and the shot including time window t is selected for the trailer generation.

7.3 Clip Assembly

In the assembly stage, the selected video clips and their respective audio parts are put together in their final form. We have experimented with two different degrees of freedom to arouse the watchers' curiosity and fantasy:

- ordering and
- the type of transitions (edits) between the clips.

Ordering

Pryluck et. al [17] showed that the sequencing of clips strongly influences the viewers' judgement about their meaning. Therefore, the clips are concatenated according to the following rules: The video clips are grouped into four classes. The first class or the *event class*, consists of the special events: cries, shots and explosions. The second class consists of *dialogs*, while the *remaining clips* (i.e. the fill-up clips with action) constitute the third class. The extracted *text* occurences in the form of bitmaps and ASCII text are treated separately. Within each class the temporal order is preserved since it seems impossible to find reasonable resequencing rules with so little informa-

tion about the video content. Thus, it is best to preserve the temporal order. To arouse the watchers' curiosity and fantasy, dialogs and event clips are assembled in turn into several so-called *edited groups*. The length of an edited group is restricted to a quarter of the length of the total share of special events. The gaps between them are filled up with the remaining clips resulting in a preliminary trailer.
The text occurrences usually show the film title, the names of the main actors and the production company. From them, the guessed title bitmap is always appended to the end of the trailer and cut to a length of one second. Optionally, the other text occurrences (actors' names and production company) can be added to the trailer. If so, these text bitmaps are scattered over the full length of the trailer. Since their size is generally small with regard to the frame size, they are superimposed on the trailer either at the left or right side and thus consume no additional time of the trailer's target play time.

Edits
Three different types of video edits are considered: hard cuts, dissolves, and wipes. Their usage should be based on rules derived from knowledge elicited from professional cutters and a model of editing [12]. However, this is a research field in its own right. As a preliminary solution we found it reasonable to concatenate event clips with every other clip by means of hard cuts and leave the soft cuts (dissolves and wipes) to the calmer clips such as dialogs. Table 1 shows the actual usage in the different cases. If more than one type of edit is specified, one is chosen randomly or by user specification for each trailer. A much more sophisticated approach for automatic video editing of humorous themes can be found in [12].

	Event Clips	**Dialog Clips**	**Other Clips**
Event Clips	hard cut	hard cut	hard cut
Dialog Clips	hard cut	dissolve, wipe	hard cut, dissolve, wipe
Other Clips	hard cut	hard cut, dissolve, wipe	hard cut, dissolve, wipe

Table 1: Editing rules

Independently of the type of video edit, the audio clips are always concatenated by a short dissolve between the outgoing and incoming audio clips.

8 HTML-Page
The automatically extracted information together with the manually specified movie title is collected automatically on a HTML-page (multimedia trailer). The top of the page shows the film title, an animated gif image constructed from the extracted text bitmaps (including the title), and the video abstract. These are followed by a randomly selected subset of the special events detected. A click plays the corresonding frame sequence. The special events are followed by a list of the scenes constructed by our shot clustering algorithms. The bottom part of the HTML-page is devoted to the creation parameters of the trailer such as creation time, target length, and share of special events and to some statistical data such as the total number of shots and scenes, as well as the number of shots/scenes with faces, shots/explosions and dialogues. A screen shot is shown in Figure 6. A set of Web pages can be searched on the basis of the recognized text and the manually specified title.

9 Experimental Results
While it would be desirable to have an absolute measure of the quality of a trailer, this is not possible. To decide about the quality of a trailer, therefore, we used two methods: We compared the automatically produced trailer with other man-made trailers, and we asked test persons for their opinion. The first is a difficult method, because every manual abstract of a video is produced by different people with different aims. Therefore, its results were poor. Personal opinions of test people ,however, were very encouraging. Currently, we are running a series of experiments to broaden our empirical basis. An earlier prototype has also shown very promising results (see [14]).

10 Conlusion and Outlook
We have presented an algorithm for automatic production of a video trailer of a feature film. It is based firstly on the segmentation of the input video into larger semantic units, so-called shot clusters or scenes, and secondly on the detection and extraction of semantically rich special events such as dialogs, shots, explosions and text of the title sequence. Additional video clips, audio pieces, images and text are added and combined automatically on a HTML-page. We call such a movie page a multimedia trailer.

Our inital experiments show promising results. For the future we plan to add face recognition algorithms in order to recognize the same actors reliably throughout the whole feature film as the same actors and not only within the

Figure 6: Screenshot of the HTML-page of the multimedia abstract.

same scene. We also plan to automatically extract title music in order to use it as background music for the trailer. Another directive is to enhance the shot clustering algorithms such that is does not only segment a video into scenes but also relates the scenes for their context, thus providing additional information in the clips selection process. Also, the model of editing should be improved. So far, only the relationship between the four classes of clips and their original order in the feature film determine the assembly. There should also be an analysis whether two contiguous clips fit to each other e.g. regarding color composition or other features. Moreover, we plan to abstract a bunch of feature films and to inquire test people about their opinion regarding quality.

References

[1] F. Arman, R. Depommier, A. Hsu, and M.-Y. Chiu. Context-based browsing of video sequences. *Proc. ACM Int. Conf. Multimed.*, San Francisco, CA, Oct. 1994, pp. 97-103.

[2] D. Bordwell, K. Thompson. Film Art: An Introduction. *McGraw-Hill*, Inc., 4th ed., 1993.

[3] S. Fischer. Indicator combination for content analysis of digital film. PhD thesis. University of Mannheim, 1997 (In German).

[4] S. Fischer, R. Lienhart, and W. Effelsberg. Automatic Recognition of Film Genres. *Proc. ACM Multimedia 95*, San Francisco, California, pp. 295-304, Nov. 1995.

[5] M. Hunke. Locating and tracking of human faces with neural networks. Master's thesis, University of Karlsruhe, 1994. http://ernie.sfsu.edu/~hunke/

[6] S. Lawrence, C. L. Giles, A. C. Tsoi, and A. D. Back. Face Recognition: A Convolutional Neural Network Approach. *IEEE Transactions on Neural Networks, Special Issue on Neural Network and Pattern Recognition*, accepted for publication.

[7] R. Lienhart. Automatic Text Recognition for Video Indexing. *Proc. ACM Multimedia 96*, Boston, MA, pp. 11-20, Nov. 1996.

[8] R. Lienhart, S. Pfeiffer, and W. Effelsberg. The MoCA Workbench: Support for Creativity in Movie Content Analysis. *Proc. IEEE Conference on Multimedia Computing & Systems*, Hieroshima, Japan, pp. 314-321, June 1996.

[9] R. Lienhart and F. Stuber. Automatic Text Recognition in Digital Videos. In *Image and Video Processing IV 1996*, Proc. SPIE 2666-20, pp. 180-188, Jan. 1996.

[10] R. Meddis and M. Hewitt. Virtual pitch and phase sensitivity of a computer model of the auditory periphery: pitch identification, *Journal of the Acoustical Society of America*, Vol. 89, pp. 2866-2882, 1991.

[11] M. Mills, J. Cohen, and Y. Y. Wong. A magnifier tool for video data. *Proc. ACM Computer Human Interface (CHI)*, May 1992.

[12] F. Nack and A. Parkes. The Application of Video Semantics and Theme Representation in Automated Video Editing. *Multimedia Tools and Applications*, Vol. 4, No. 1, pp. 57-83, 1997.
[13] S. Pfeiffer, S. Fischer, and W. Effelsberg. Automatic audio content analysis. *Proc. ACM Multimedia 96*, Boston, MA, pp. 21-30, Nov. 1996.
[14] S. Pfeiffer, R. Lienhart, S. Fischer, and W. Effelsberg. Abstracting digital movies automatically. In *Journal of Visual Communication and Image Representation*, Vol. 7, No. 4, pp. 345-353, December 1996.
[15] C. A. Poynton. A Technical Introduction to Digital Video. *John Wiley & Sons*, Inc. 1996.
[16] G. Pass, R. Zabih, and J. Miller. Comparing Images Using Color Coherence Vectors. *Proc. ACM Multimedia 96*, Boston, MA, pp. 65-73, Nov. 1996.
[17] C. Pryluck, C. Teddlie, and R. Sands. Meaning in film/video: Order, time and ambiguity. *J. Broadcasting* **26** (1982), 685-695.
[18] M. E. Rorvig. A method for automatically abstracting visual documents. *Journal of the American Society on Information Science* **44** (1993) 1.
[19] M. Smith and T. Kanade. Video skimming for quick browsing based on audio and image characterization. Computer Science *Technical Report, Carnegie Mellon University*, July 1995.
[20] Y. Taniguchi, A. Akutsu, Y. Tonomura, and H. Hamada. An intuitive and efficient access interface to real-time incoming video based on automatic indexing. *Proc. ACM Multimedia 1995*, San Francisco, CA, Nov. 1995, pp. 25-33.
[21] Y. Tonomura, A. Akutsu, Y. Taniguchi, and G. Suzuki. Structured video computing. *IEEE Multimedia Magazine* **1** (1994), 34-43.
[22] M. Yeung, B.-L. Yeo, W. Wolf, and B. Liu. Video browsing using clustering and scene transitions on compressed sequences. *Multimedia Computing and Networking* 1995, Proc. SPIE 2417, San Jose, pp.399-414.
[23] M. Yeung, B.-L. Yeo, and B. Liu. Extracting Story Units form Long Programs for Video Browsing and Navigation. *Proc. IEEE Multimedia Computing & Systems*, Hiroshima, Japan, pp. 296-305, June 1996.
[24] M. Yeung, B.-L. Yeo. Video Content Characterization and Compaction for Digital Library Applications. In *Storage and Retrieval of Image and Video Databases 1997*, Proc. SPIE 3022, pp. 45-58, Jan. 1997.
[25] H. A. Rowley, S. Baluja, and T. Kanade. Human face recognition in visual scenes. *Technical Report Carnegie Mellon Uuniversity*, CMU-CS-95- 158R, School of Computer Science, November 1995.
[26] R. Zabih, J. Miller, and K. Mai. A Feature-Based Algorithm for Detecting and Classifying Scene Breaks. *Proc. ACM Multimedia 95*, San Francisco, CA, pp. 189-200, Nov. 1995.
[27] H. Zhang, S.W. Smoliar, and J. H. Wu. Content-based video browsing tools. *Multimedia Computing and Networking* 1995, Proc. SPIE 2417, San Jose, pp. 389-398.
[28] H. Zhang, C. Y. Low, S. W. Smoliar, and J. H. Wu. Video parsing, retrieval and browsing: An integrated and content-based solution. *Proc. ACM Multimedia 1995*, San Francisco, CA, Nov. 1995, pp.15-24.