JOURNAL OF VISUAL COMMUNICATION AND IMAGE REPRESENTATION
Vol. 7, No. 4, December, pp. 345–353, 1996
ARTICLE NO. 0030

Abstracting Digital Movies Automatically

SILVIA PFEIFFER,* RAINER LIENHART, STEPHAN FISCHER, AND WOLFGANG EFFELSBERG

Praktische Informatik IV, University of Mannheim D-68131 Mannheim, Germany

Received April 15, 1996; accepted September 6, 1996

Large video-on-demand databases consisting of thousands of digital movies are not easy to handle: the user must have an attractive means of retrieving his movie of choice. For analog video, movie trailers are produced to allow a quick preview and perhaps stimulate possible buyers. This paper presents techniques for automatically producing such movie abstracts of digital videos. © 1996 Academic Press

1. INTRODUCTION

In the context of video-on-demand, it is vital to provide an efficient and user-friendly means for selecting a video from a large video archive. In current movie marketing, it is common to produce a *trailer* (a short summary) of a movie in order to get people interested. Similarly, television journalists can retrieve relevant clips for a report from large digital archives. With a vast number of stored videos in a video archive, it is not possible to produce by hand a trailer for every video stored. Therefore, we have thought about ways to create video abstracts automatically and have implemented a prototype system named *VAbstract*. Other work on this topic is often based on textual abstracts generated from captions (e.g., [1]), or they extract still images only (see, e.g., [2–8]). In contrast, we automatically produce a short movie abstract based on automatic content analysis of the video.

The paper is structured as follows. Section 2 defines a video abstract and explains some basic decisions behind our abstracting research. Section 3 presents some simple abstracting concepts and shows that these straightforward approaches are not very satisfactory. Section 4 describes properties of trailers and psychological experiences concerning the video perception process. These are used in Section 5 to derive our new algorithms. Section 6 presents the prototype *VAbstract* with which the experimental results of Section 7 were produced. Section 8 describes some related work and Section 9 concludes the paper.

* E-mail: pfeiffer@pi4.informatik.uni-mannheim.de, Fax: +49-621-292-5745.

2. VIDEO ABSTRACTING TERMS

2.1. Definition

We all know what an abstract of an article is: it is a short summary of the content of a longer document which is used to preselect articles relevant to the user. Analogous to this definition, we define a *video abstract* to be 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. (In the scientific literature, there is no common accepted definition of the term ''video abstract.'' We believe we have given a definition that is wide enough to comprise all former usage of this term, but narrow enough to contain no more than necessary.)

2.2. Project Decisions

Basic decisions had to be made about the type of material that will be used as input, the kind of abstracting procedure, and the type of material that will be produced as output by our abstracting system.

2.2.1. Input Material. As known from cinema trailers, different types of material can be used for the production of a common movie trailer: unchanged material from the original movie, revised material, and/or material that was not used in the original. In our project, we use only *unchanged material* from the original movie. The reason is that in a video-on-demand archive, only the original movie and no additional material from the movie production process is available in the database. Our system will work on any video archive, independent of additional sources of information.

2.2.2. Target group. From the above-given definition it follows that the quality of an abstract can only be judged in relation to a specific target group. For example, the aim of viewers of documentaries is to receive information, whereas the aim of feature film viewers is to be entertained. Therefore, these abstracts should differ: a documentary abstract should give an overview of the contents of the

1047-3203/96 $18.00
Copyright © 1996 by Academic Press
All rights of reproduction in any form reserved.

entire video, whereas a feature film abstract should be entertaining in itself and not reveal the end of the story. In feature films, the main actors and their dialogs are very important—they are of no particular interest in documentaries. The audio content, however, is of high interest in documentaries. In our project, we concentrate on abstracting *feature films*.

2.2.3. Abstracting method. There are several methods of producing an abstract from a video:

- Manual abstracting: the material for the abstract is completely chosen by a human.
- Semi-automatic abstracting: the first choice of material is made by a computer, and a human will produce the final abstract interactively.
- Automatic abstracting: the abstract is completely produced by a computer based on preset parameters.

Our abstracting system is *automatic*.

2.2.4. Still-image vs moving-image abstracts and audio abstracting. There are two fundamentally different abstracts that can be produced from the picture stream: still- and moving-image abstracts. A still-image abstract is a collection of extracted salient images. If one frame is extracted per scene, these frames are called keyframes, as they identify a scene. A moving-image abstract consists of a collection of sequences of images from the original movie and is thus a proper movie itself.

Similarly, two types of audio abstracts can be distinguished: single-word and continuous-audio abstracts. It is possible to translate the spoken words through speech recognition into text and then use the classic methods of information retrieval from text [9]. Such methods usually produce a list of keywords identifying the content. In a movie there are also other important sounds in addition to speech. In order to keep all the important sounds, we decided to abstract the audio by *extracting continuous parts of the sound track*.

The most important decision was whether our abstracting system should produce a still- or moving-image abstract, the differences between which are great. A still-image abstract can be built much faster, as less data has to be handled. Once composed, it is displayed more easily, as there are no timing or synchronization constraints. It is even possible to display simultaneously several images next to each other such that the temporal order of events is displayed in a spatial order and can be grasped more quickly by the user. Such a collection of stills can also be printed out very easily.

There also are advantages to a moving-image abstract. Compared to a still-image abstract, it makes much more sense to use the original audio information for a moving-image abstract. The possibly higher computational effort during the abstracting process pays off during playback time: the abstract can be displayed on common TV sets, its playback is much more natural for the user, as watching a trailer (i.e., a moving-image abstract) is much more exciting than watching a slide show (i.e., a still-image abstract), any kind of blurriness that turns up is caused by real movement and not by an accidental choice of an unfocused image, and in many videos the motion is information bearing. These very important advantages led us to produce a *moving-image* abstracting system.

3. STRAIGHTFORWARD CONCEPTS

3.1. Express Run-Through

The easiest way to get a fast overview of a video is with fast-preview mode, as known from VCRs. When a video database user wants to get a quick overview of several longer videos, however, this is not very satisfying for several reasons:

- The maximum compression factor possible with this approach is between 3 to 1 and 6 to 1, as otherwise the plot happens too quickly to be grasped. Therefore, it usually takes much longer than desired to browse a video.
- The cut frequency, i.e., the amount of cuts per time unit, is raised by the compression factor. This heavily overstrains human concentration because of a much elevated cut frequency, becoming unbearable especially when watching more than two or three abstracts.
- The audio of the original cannot be used, as it would make no sense to the viewer when played back fast. Therefore a user can only make use of the information available from the picture track.

All in all, this is a method for situations requiring a detailed overview of a short video, e.g., for finding a desired point in the video. This method is easily implemented. It is, however, unsuitable for our aim of composing a trailer automatically.

3.2. Random Choice of Scenes or Images

Instead of using an intelligent method to select scenes or images for an abstract, one might propose that a random choice may result in an abstract of just as good a quality. However, for representing the characteristic contents of an original, the compression factor must be kept low; i.e., a vast amount of scenes/images must be extracted. Such a sample survey will almost certainly include irrelevant scenes or images with little information. Thus, this method could only be used for semi-automatic abstracting. The information content of the abstract will often be mediocre. Nevertheless, this method is very easy to implement.

4. GOOD-QUALITY ABSTRACTS

When watching cinema trailers, one becomes aware of some properties of good video abstracts. We feel that these are the following:

[P1] *Important objects/people.* The most important objects and/or actors appearing in the original also appear in the trailer.

[P2] *Action*. Many scenes in a trailer include a lot of action.

[P3] *Mood*. A trailer represents a mood very similar to that of the original movie. Emotionally important scenes are included.

[P4] *Dialog.* Dialog scenes often include important information such as a change of action or atmosphere and are therefore often included in trailers.

[P5] *Disguise end.* The end of the movie is never revealed, as the story's thrill is often released there.

VAbstract aims at extracting scenes from the original movie according to these rules. Taking a look at some psychological experiences about the perception of video helps us in converting the above-mentioned properties into algorithms:

[E1] *Contrast perception.* Processing of visual stimuli within the human visual system is mainly based on perception of contrast [10]. Lines and contours are determined from the contrasts, and only on their basis is the perception of forms and objects performed. The brain goes through a multistage pattern-matching process with stored patterns in memory. It can even complete missing parts of an image in order to recognize shapes. This process is much faster with images or scenes containing high contrast.

[E2] *Color perception.* The perception of colors depends on the context of their appearance: especially their perceived brightness and saturation depend on the relative difference of brightness and saturation from those of other colors in the same picture or in adjoining pictures. For example, a dark color may appear relatively bright next to an even darker color, or very dark next to bright colors. Colors are fundamentally important to the transfer of emotions. Thus, it is important that the basic color contents of a video are preserved to represent the mood of a video (imagine an abstract of the movie *Waterworld* without scenes of the ocean).

[E3] *Context perception.* Pryluck *et al.* [11] have shown that the sequence of scenes or images in a video has a strong influence on their impression on the viewer. In their book, they refer to an experiment by H. D. Goldberg with a scene of a child riding a tricycle on the road. He showed this scene together with images either of a happy family life or of busy roads. Depending on the context in which the tricycle scene was shown, people associated different meanings with the scene. The sequence in which scenes are put together strongly influences the perceived content of the abstract. Therefore, we have to be very careful not to present scenes in our trailer in a misleading context.

[E4] *Stimuli processing.* Inputs from the eye first go to the sensory memory, where they are available for analysis for about 150 ms. During this time, the next image has to arrive such that an impression of movement will result [10]. Psychological experiments have proven that the human visual system takes about 400 to 700 ms to process one visual stimulus. If there are several stimuli that must be processed together, some kind of parallel processing is possible, but each additional stimulus takes about 100 to 300 ms longer. As the short-term memory cannot register more than about five to nine items, we can now calculate the time that a human needs to completely analyze a scene, i.e., to recognize all objects that he/she is able to recognize jointly:

$$\begin{aligned} &150 \text{ ms recognition of movement} \\ &+ 700 \text{ ms first stimulus} \\ &+ 2400 \text{ ms eight additional items} \\ &= 3.25 \text{ s} \end{aligned}$$

Therefore, a scene must be at least 3.25 s long to get completely analyzed (we assume that there are usually at least nine items to be recognized within a scene). If, however, all extracted scenes are that short, this strains the concentration of the viewer because he must be 100% concentrated on each scene. To limit the demand upon a viewer's full mental capacity to at most 50%, we conclude that the average of the lengths of all extracted scenes for the abstract should exceed 6.5 s. We are aware of the fact that these arguments are very vague but we use them as a first approximation.

5. ALGORITHMIC REALIZATION

The design of our algorithms is based on the aforementioned properties of cinema trailers taking into account the psychological experiences about human video perception.

The basic entity for automatic abstracting is a scene: our goal is to extract a number of scenes from the original fulfilling properties P1 to P5. Therefore, *cut detection* has to be performed first (see, e.g., [12–16]). As we intend to include audio, an *audio cut detection* [17] has to be performed as well so as not to include meaningless audio pieces. Scene limits are now defined by an audio and a video cut.

For the extraction of important objects/people (property P1), we can make an essential simplification: It is not necessary to explicitly recognize objects or people in order to include them in the abstract. When enough scenes or still images are extracted from the original, it is statistically

unlikely that an object turning up often in the original would not be part of the abstract. Recognition of objects or people by the viewer is also ensured from parts of a video, and when images/scenes with high contrast are chosen, objects are recognized more quickly, according to rule E1. Therefore, an algorithm for extraction of *high-contrast* scenes is implemented (see, e.g., [18]). Object/people recognition itself is left to the human.

For the identification of high-action scenes (i.e., property P2), it suffices to define "action" through "motion" in our context. It makes no difference whether we find a lot of camera or object motion—both have their part in determining "action." Therefore, simple frame difference calculations suffice (see, e.g., [19]). If a high degree of *motion* is recognized in a scene, we conclude that it should be included in the abstract.

As rule E2 tells us, colors are an important component for the perception of a video's mood (property P3). As we want to represent the basic mood of the original movie in the abstract, it is necessary to extract scenes that have a *basic color composition* similar to the average color composition of the whole movie. Color composition is calculated based on brightness, color composition, and saturation (see, e.g., [18]).

Recognition of *dialog* scenes (property P4) can be performed heuristically on the audio track. The audio signal is transferred to a short-term frequency spectrum and the fundamental frequency is extracted [20]. After normalizing, the spectrum is compared to the spectrum of a spoken "a" in order to distinguish between speech and other sounds or noise. A dialog is then characterized by the existence of two such "a"s with significantly different fundamental frequencies, identifying scenes with at least two different speakers [21, 22]. The "a" is used because it occurs frequently in most languages and because it is usually spoken as a long sound, improving the rate of recognition. This method is much easier to implement than a complete speaker recognition.

The end of a feature film can be disguised (property P5) by partitioning the video into *several parts* of equal length and not using the last part for abstracting. In our experiments, we have subdivided feature films into 10 parts, not using the last one for abstracting. This should suffice for disguising the end.

In order to reduce the possibility of a misleading context change for a scene between the original movie and the abstract, the selected scenes should at least be presented in their *original sequence*. An exception can be made for extracted title frames which may be appended to the feature film abstract, as this is a common habit in trailer production. However, when selected scenes are simply concatenated, a new, unintended context can still result, as scenes which were shown in a certain context now appear with different neighbor scenes. Therefore, the abstract synthesis should include some *black frames with silence* between the selected scenes in order to clearly indicate to the viewer that these are single scenes "without context."

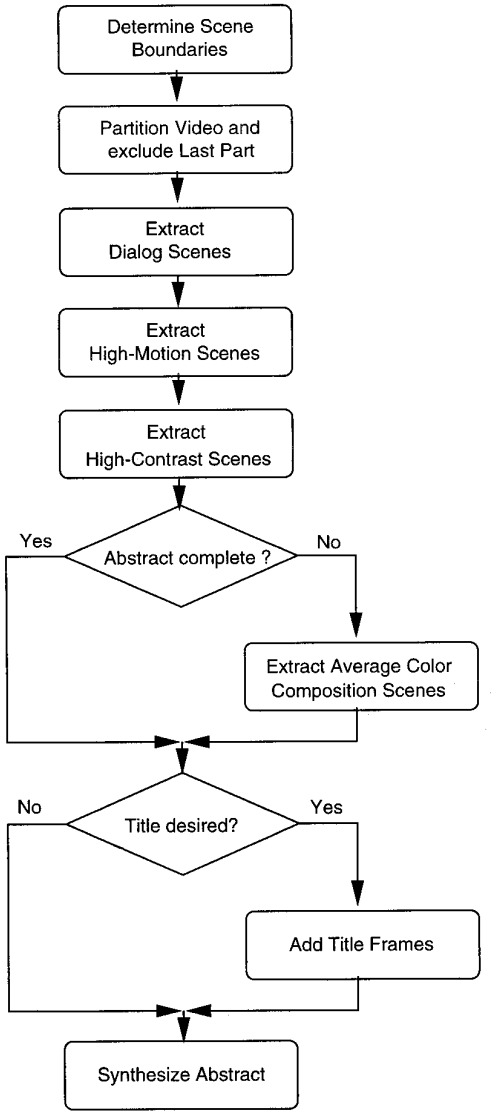

FIG. 1. Abstracting algorithm.

6. *VAbstract:* THE PROTOTYPE VIDEO ABSTRACTING SYSTEM

We have implemented a prototype video abstracting system called *VAbstract* following the scheme outlined above. Although this system is still at an early stage, it is already usable for producing abstracts automatically.

6.1. *The Concept*

Figure 1 shows a flow chart of the implemented system. The extracted scenes must each be longer than 3.25 s. In

our first approach, we select at most one dialog, one high-motion, and one high-contrast scene from each of the examined parts of the video, leaving out the last part when analyzing a feature film. The user determines the desired length of the abstract, e.g., 120 s. If with the selected scenes, the abstract is not complete, it is filled up with scenes whose average color is close to the average color composition of the whole video. If too many scenes were chosen, less suitable ones are dropped again. Afterwards, those frames from the video bearing the title can be selected. Finally, the abstract is synthesized by composing the selected scenes with their audio, separating them by black frames with silence and either appending or prepending the title frames.

VAbstract is also usable as a still-image abstracting system. It determines *important* scenes to be extracted. It is then possible to simply select one (e.g., the center) frame from each scene for an abstract. Such a still-image abstract would be much denser in content than a simple still-image abstract that includes one picture from *every* scene of the video, as it is composed only of pictures from important scenes.

6.2. *Status of Implementation*

VAbstract was implemented in about 1500 lines of ANSI C using the Vista library V2.1.3 [23, 24]. The audio modules currently are still missing. Title extraction must be done by hand, but we are close to an automatic solution [25].

Two example application interfaces were built in about another 2500 lines of Tcl/Tk code on top of *VAbstract*: a user interface assisting a video librarian in selecting a video (see Fig. 2) and a ''provider'' interface supporting a video library provider in the construction and administration of a video library (see Fig. 3).

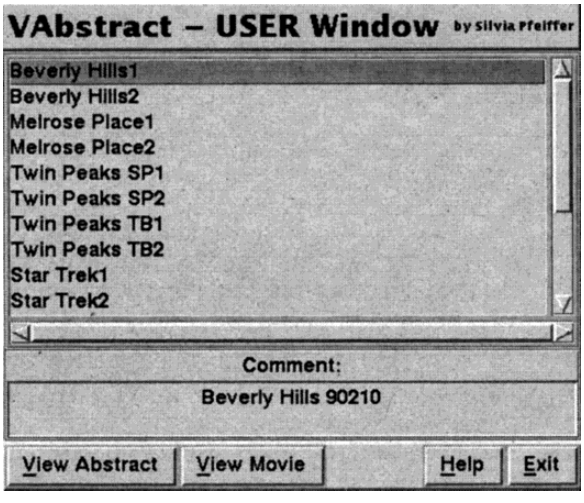

FIG. 2. User interface for *VAbstract.*

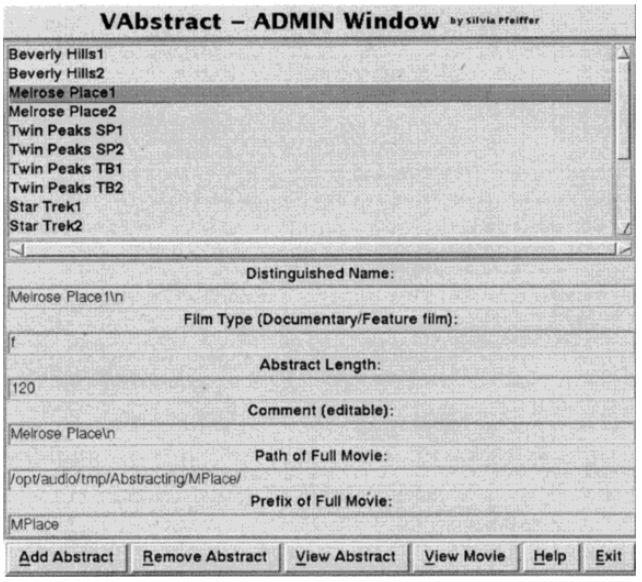

FIG. 3. Provider interface for *VAbstract.*

7. EXPERIMENTAL RESULTS

The movies for our experiments were recorded from German television, digitized by a Parallax video board, and stored as a collection of JPEG frames. We reduced the frame grabbing rate to about 4 frames/s so that we were able to keep all the analyzed videos on disc. For 1 h of video, digitizing and preprocessing took about 4 h. Experiments were performed on a cartoon video, on different TV series, and on a feature film.

Abstracting of a documentary was tried as well. In this case, we also included the last part of the video because documentaries do not have a ''showdown.'' We found out that documentaries on TV often have a predefined structure similar to chapters in a book: a group of scenes is devoted to one topic. There are clear semantic boundaries, e.g., speaker scenes, between the ''chapters.'' In this case, we have provided the possibility to manually prepartition the video before using our abstracting algorithms on each part. This guarantees that the abstract contains relevant scenes from each part even if some of them are low in motion or contrast.

For our experiments, we chose movies for which a manually produced abstract (trailer) was available. For example each episode of the TV series *Twin Peaks* started with a short (human-generated) summary of the previous episode. Thus we were able to compare the man-made abstracts with our automatically produced ones. Table 1 gives an overview of the seven videos which we analyzed.

Our experiences with trailers have shown that some of the aforementioned properties of good abstracts do not apply to all kinds of trailers: *previews* usually concentrate

TABLE 1
Overview of Analyzed Videos and Their Original Trailers

Video	Video length [min]	Number of scenes in video	Number of frames in digitized video	Trailer type	Trailer length [s]	Number of scenes in trailer
Beverly Hills	45	488	9,707	Preview	38	9
Melrose Place	47	510	10,046	Preview	34	11
Twin Peaks (SP)	48	264	10,326	Summary	64	21
Twin Peaks (TB)	46	313	9,965	Summary	48	21
Star Trek	51	425	10,874	Summary	120	45
Flintstones	35	218	7,550	None	—	—
Indiana Jones	115	761	24,492	None	—	—

on the one most important scene of the next episode, whereas *summaries* include the last part of the abstracted video, as this usually contains the foundation for the action of the next episode. TV series trailers therefore fundamentally differ from movie trailers. *VAbstract* extracts scenes a lot more evenly spread over the whole length of the original than they are in the man-made preview or summary. To give an example, the automatically generated abstract of *Beverly Hills* picked 13 scenes between frame numbers 253 and 7924. In contrast, the man-made preview only included scenes between frame numbers 430 and 521, leaving out all other scenes.

We have also discovered that manually produced abstracts usually do not contain scenes in their original time order. Often, such a change of sequence is performed deliberately to produce a new context, which did not exist in the original video. The reason for this might be to compress a long and complicated story into a shorter but still suitable version which is more impressive to the viewer.

Another basic difference between manually produced trailers and our approach is that scenes for trailers are often not included as a whole. A human can easily extract the important part of a scene for the story, whereas our system is more or less ignorant of the main action stream. Although we compare the results of our system to the manually produced trailers, one has to keep in mind that our algorithms are based on ''syntactic'' parameters and not on the story. Our prototype is still based only on the picture stream, and the mainline of the story often occurs in the audio stream. Future versions of *VAbstract* will include audio analysis, and we expect them to work better.

In Table 2 we give an overview of the abstracts that we produced with *VAbstract*. From the shorter movies we produced 2- and 4-min-long abstracts; *Star Trek* had a longer trailer, so we produced an additional 8-min abstract, and from *Indiana Jones*, we produced 6-, 8-, and 10-min abstracts.

It is quite interesting to examine the duration of the production process for these abstracts. The numbers are of course dependent on the speed of the machine, the amount of main memory available, and the average load of the machine, but they indicate the duration of each abstracting step. When an abstract is produced for the first time, this takes about 6 to 7 times the length of the original film because cut detection has to be performed, and our cut detection algorithm is not very efficient. The results from cut detection are saved in a file such that subsequent abstracting runs are performed much faster. There is a basic abstracting duration of about 10 min for each video. For each 120 s of abstract length, about 5 min abstracting duration have to be calculated additionally. Our results were produced on a DECalpha 3000 running OSF/1 V3.2.

Table 3 compares the automatically produced abstracts with the man-made trailers (where available). Although some scenes are obviously identical, there is still a large amount of difference, for the reasons mentioned above. The difference when viewing the abstracts may not be as large, because there are always similar scenes in a video from which both automatic and manual abstracter have to choose, and they usually do not decide on the same scene.

Some descriptive words on the subjective quality of the abstracts may be allowed here. We believe that the abstracts are already quite good for a prototype system: usually, all important places of action are extracted. The abstracts are better for feature films than for TV series, as the series usually contain more than one action stream, which makes it more difficult for a user to follow. When watching the abstracts, a lack of thrill is noticeable. The

TABLE 2
Number of Scenes of Each Generated Abstract

Video	120-s abstract	240-s abstract	360-s abstract	480-s abstract	600-s abstract
Beverly Hills	13 scenes	23 scenes	—	—	—
Melrose Place	6 scenes	11 scenes	—	—	—
Twin Peaks (SP)	10 scenes	18 scenes	—	—	—
Twin Peaks (TB)	6 scenes	12 scenes	—	—	—
Star Trek	9 scenes	22 scenes	—	35 scenes	—
Flintstones	9 scenes	20 scenes	—	—	—
Indiana Jones	—	—	28 scenes	31 scenes	34 scenes

TABLE 3
Comparison of Abstracts and Trailers: Number of Identical Scenes

Video	120-s abstract	240-s abstract	360-s abstract	480-s abstract	600-s abstract
Beverly Hills	1	3	—	—	—
Melrose Place	0	0	—	—	—
Twin Peaks (SP)	2	3	—	—	—
Twin Peaks (TB)	3	5	—	—	—
Star Trek	3	4	—	6	—

reasons for this may be that too many uninteresting scenes for the mainline story were also extracted. More important is the missing audio: it is well known that background music is the most important stylistic device for the creation of thrill [26]!

As the experiments have shown, the system already produces acceptable abstracts even though some algorithms are not yet implemented. It has been shown that a proper person/object recognition is not necessary, as the most important people/objects did appear in the abstract thanks to our action- and motion-detection algorithms. An additional implementation of person identification could be used to avoid a strong concentration on a single, very active person. It has also been shown that the missing dialog and title recognition are very important; they are currently being implemented.

8. RELATED WORK

8.1. Still-Image Abstracting Systems

The first digital video browsers reported in the literature collected still images, usually extracting frames at equal distances (see, e.g., [27]). These images were then reduced in size and displayed in sequence. No attention was paid to semantic issues.

There are now more advanced systems of this type (see, e.g., [2–8]). Most newer systems first perform a segmentation of the video based on the detection of scenes (shots) and then select a frame from each scene (shot). Sometimes not merely the first frame is chosen, but the first one of good quality or the first one that satisfies certain color or motion requirements. This ensures a more content-related extraction of still images. As there are usually still a vast number of extracted images which need to be handled, some systems try to reduce this number by making a selection based on camera movement or on content. There are even some systems that enhance their extracted still images by some means to playback small sequences of images as moving images. As they usually select a block of frames around an extracted frame for displaying or let the user choose the scenes via extracted frames, their result is not considered to be a real moving-image abstract.

Most current still-image abstracts serve as an index for users, for easier selection of the viewing start position. Therefore, the browsing character of such systems is very distinct, whereas their abstracting character is usually only rudimentary. Usually, audio is not included in still-image abstracts. However, Taniguchi *et al.* [4] describe a system extracting keyframes for each scene. Their ''abstract'' is called Mini-Video and consists of these keyframes plus audio data. The length of the audio track is not shortened at all, as the intention is to keep a record of a lot of video and audio data using as little memory space as possible. Therefore, this system is not regarded as one abstracting the audio.

8.2. Video Skimming

Smith and Kanade from Carnegie Mellon University propose a method for extracting the significant audio and video information from a video, creating a *skim* video representing a short synopsis of the original [16]. They perform this extraction based on the significance of objects or words appearing in each scene and on the structure of the video scene. The context of their work is easy usage of digital video libraries, storing full-featured videos in content-specific segments. A user's query to such a video library will extract such segments from videos, so-called *video paragraphs*. These paragraphs are then *skimmed*, i.e., significant images and words from a paragraph are extracted to produce a short synopsis for easy browsing and retrieval.

We view a video skim as a moving-image abstract, although it consists of only a few consecutive frames from a scene and is therefore similar to some of the still-image abstracts. What makes a video skim different from a keyframe abstracting system is that the number of extracted frames is not constant but is dependent on the image and audio content of the scene, and, more importantly, not all scenes must be used for an abstract, thus eliminating scenes of low interest.

9. CONCLUSION

Video abstracting will gain increasing importance in the future when video-on-demand services are started in metropolitan area networks. Automatic abstracting services will be necessary in order to enable users to select the desired videos with a high hit ratio.

We have presented several fundamental ideas for automatically abstracting digital videos based on video content characteristics and a prototype system called *VAbstract* based on these ideas. We also have presented a series of abstracting experiments which have given us more insight

into techniques of manual abstract production. The largest deficit of our system is the missing audio analysis—important pieces of the story are often contained in dialog on the audio track. Currently, we are implementing the audio modules and title-extraction algorithms for improving our results. Future work will concentrate on gaining more knowledge about abstracting other types of video genres and better usage of predefined structure in videos. Also, we expect considerable runtime improvements for *VAbstract* as we implement more efficient algorithms for cut detection, motion intensity, and color composition analysis. We are aware of the fact that our basic rules are empirical; there is no formal proof of correctness. We can only evaluate them empirically, including human viewers as our judges of quality. We intend to broaden our empirical basis with more human users.

ACKNOWLEDGMENT

Many thanks to Helge Blohmer [28], who developed and implemented a first prototype of the *VAbstract* system in C.

REFERENCES

1. A. P. de Vries, Television information filtering through speech recognition, in *Interactive Distributed Multimedia Systems and Services* (B. Butscher, E. Moeller, and H. Pusch, Eds.), pp. 59–69, Lecture Notes in Computer Science, Vol. 1045, Springer-Verlag, Berlin/New York, 1996.
2. F. Arman, R. Depommier, A. Hsu, and M.-Y. Chiu, Contest-based browsing of video sequences, in *Proc. ACM Int. Conf. Multimed., San Francisco, CA, Oct. 1994*, pp. 97–103.
3. M. E. Rorvig, A method for automatically abstracting visual documents, *J. Amer. Soc. Inform. Sci.* **44** (1993), 1.
4. Y. Taniguchi, A. Akutsu, Y. Tonomura, and H. Hamada, An intuitive and efficient access interface to real-time incoming video based on automatic indexing, in *Proc. ACM Int. Conf. Multimed., San Francisco, CA, Nov. 1995*, pp. 25–33, ACM Press, New York.
5. Y. Tonomura, A. Akutsu, Y. Taniguchi, and G. Suzuki. Structured video computing, *IEEE Multimed. Mag.* **1** (1994), 34–43.
6. M. M. Yeung, B.-L. Yeo, W. Wolf, and B. Liu, Video browsing using clustering and scene transitions on compressed sequences, in *Multimed. Comput. and Network (San José, Feb. 1995)* (A. Rodriguez and J. Maitan, Eds.), pp. 399–414, Proc., SPIE, Vol. 2417, Int. Soc. Opt. Eng., Bellingham, WA, 1995.
7. H. Zhang, S. W. Smoliar, and J. H. Wu, Content-based video browsing tools, in *Multimed. Comput. and Network, (San José, Feb. 1995)* (A. Rodriguez and J. Maitan, Eds.), pp. 389–398, Proc. SPIE, Vol. 2417, Int. Soc. Opt. Eng., Bellingham, WA, 1995.
8. H. Zhang, C. Y. Low, S. W. Smoliar, and J. H. Wu, Video parsing, retrieval and browsing: An integrated and content-based solution, in *Proc. ACM Int. Conf. Multimed. (San Francisco, CA, Nov. 1995)*, pp. 15–24, ACM Press, New York.
9. G. Salton and M. J. McGill, *Introduction to Modern Information Retrieval*, McGraw–Hill, New York, 1983.
10. P. H. Lindsay and D. A. Norman, *Introduction into Psychology—Human Information Reception and Processing*, Springer-Verlag, Berlin/Heidelberg/New York, 1991. [in German]
11. C. Pryluck, C. Teddlie, and R. Sands, Meaning in film/video: Order, time and ambiguity, *J. Broadcasting* **26** (1982), 685–695.
12. H. Zhang, A. Kankanhalli, and S. W. Smoliar, Automatic partitioning of full-motion video, *Multimed. Syst.* **1** (1993), 10–28.
13. R. Zabih, J. Miller, and K. Mai, A feature-based algorithm for detecting and classifying scene breaks, in *Proc. ACM Int. Conf. Multimed. (San Francisco, CA, Nov. 1995)*, pp. 189–200.
14. A. Hampapur, R. Jain, and T. E. Weymouth, Production model based digital video segmentation, *Multimed. Tools Appl.* **1** (1995), 9–46.
15. F. Arman, A. Hsu, and M.-Y. Chiu, Image processing on encoded video sequences, *Multimed. Syst.* **2** (1994), 211–219.
16. M. Smith and T. Kanade, Video skimming for quick browsing based on audio and image characterization. Computer Science technical report, Carnegie Mellon University, Jul. 1995.
17. C. Gerum, *Automatic Recognition of Audio-Cuts*, Master's thesis, University of Mannheim, Germany, Jan. 1996. [in German]
18. A. K. Jain, *Fundamentals of Digital Image Processing*, Information and System Science Series, Prentice Hall, Englewood Cliffs, NJ, 1989.
19. D. Murray and A. Basu, Motion tracking with an active camera, *IEEE Trans. Pattern Anal. Mach. Intell.* **16** (1994), 449–459.
20. Structure of music. Festschrift, Fritz Winckel zum 75. Geburtrstag am 20.June 1982, Technical University and Academy of Arts, Berlin, 1982. [in German]
21. W. A. Ainsworth, *Mechanisms of Speech Recognition*, Pergamon, Oxford/New York/Ontario, 1976.
22. B. Eppinger and E. Herter, *Speech Processing*, Hauser, Munich/Vienna, 1993. [in German]
23. A. R. Pope, D. Ko, and D. G. Lowe, Introduction to vista programming tools. Tech. rep., Department of Computer Science, University of British Columbia, Vancouver, 1995.
24. A. R. Pope and D. G. Lowe, A software environment for computer vision research. Tech. rep., Department of Computer Science, University of British Columbia, Vancouver, 1995.
25. R. Lienhart and F. Stuber, Automatic text recognition in digital videos, in *Image and Video Processing IV,* Proc. SPIE, Vol. 2666-20, Int. Soc. Opt. Eng., Bellingham, WA, 1996.
26. D. Bordwell and K. Thompson, *Film Art: An Introduction*, 4th ed., McGraw–Hill, Englewood Cliffs, NJ, 1993.
27. M. Mills, J. Cohen, and Y. Y. Wong, A magnifier tool for video data, in *Proc. ACM Computer Human Interface (CHI), May 1992.*
28. H. Blohmer, *Video-abstracting*, Master's thesis, University of Mannheim, Germany, Jul. 1995. [in German]

SILVIA PFEIFFER is a research assistant at the professorship for computer science of Professor Effelsberg in Mannheim, Germany, pursuing her Ph.D. in computer science within the MoCA (Movie Content Analysis) project. Her interests include audio processing, especially digital music processing, psychoacoustics, and information retrieval from multimedia documents. She received her Master's Degree in computer science and business administration from the University of Mannheim, Germany, in 1993.

RAINER LIENHART is a research assistant at the professorship for computer science of Professor Effelsberg in Mannheim, Germany, pursuing his Ph.D. in computer science within the MoCA (Movie Content Analysis) project. His interests include image/video processing, computer vision, interactive video, and information retrieval. In 1994 he joined the Tenet Group at the International Computer Science Institute in Berkeley for six months and received his Master's Degree in computer science and business administration from the University of Mannheim, Germany, in the same year.

STEPHAN FISCHER is a doctoral candidate in the Department of Mathematics and Computer Science at the University of Mannheim, Germany. His research focuses on multimedia content processing within the MoCA (Movie Content Analysis) project. Mr. Fischer received his Master's Degree in Computer Science and Business Administration from the University of Mannheim, Germany, in 1993.

WOLFGANG EFFELSBERG received his Master's Degree in electrical engineering in 1976 and Dr.-Ing. degree in computer science in 1981 from the Technical University of Darmstadt, Germany. From 1981 to 1984 he was an assistant professor at the University of Arizona in Tucson, then a post-doctoral fellow at IBM Research in San Jose, California. From 1984 to 1989 he was the leader of a research group on the design and implementation of communication protocols and applications for the emerging OSI networks at IBM's European Networking Center in Heidelberg. Since 1989 he has been a full professor of computer science at the University of Mannheim, Germany. His research interests include communication protocols for multimedia systems, protocols for high-speed networks, distributed multimedia applications, and multimedia content processing.