Summarizing Video Scenes

Anne Kathrein
288875
anne.kathrein@rwth-aachen.de

Justus Lauten
289077
justus.lauten@rwth-aachen.de

ABSTRACT
In this paper we address the problem of analyzing and representing complex video data, resulting in different representations of video summaries. To perceive the main content of a video, it must be watched at full length, which is a very time consuming process. People are also challenged by selecting and browsing through video clips in order to find interesting content. Videos are often presented by arbitrarily chosen keyframes of a video sequence, which do not necessarily reflect its content very well. Techniques that give concise summarizations would be desirable, in order to be able to select the video that contains the desired information and shortens browsing time. It might occur that a user needs to gain information from various sources of video material. With a good video summary it would not be necessary to watch all videos at their entire length, since relevant scenes would be presented. One problem is to decide which scenes contribute to a good summary, since an arbitrary choice of scenes would result in a set of unimportant segments. Another challenge of video summarization software is to detect shot and scene boundaries. These are needed to divide the video sequence into reasonable parts to reduce redundancy in a keyframe representation. A last issue is to generate the video summary as a visually attracting presentation, which is enjoyable for the user.

Author Keywords
Video Summarization, Skimming, Storyboarding, Comic Book Representation

INTRODUCTION

We present methods for summarizing videos to dynamic and static representations. First we employ algorithms to analyze video data by detecting shots and scenes. Techniques are described to rate the relevance of shots and frames. We present techniques that assign importance scores to shots and frames, based on factors like human perception and different properties of the video. An algorithm is presented that calculates a continuous image from the background images of a shot. In the course of this paper we discuss three different approaches for video summarization, where the described techniques are applied. The first technique is *Dynamic Video Skimming* which generates a concise video of the input video by merging the most important scenes together. Next *Comic Representation* introduces a technique which detects single shots of a video sequence, represents them as keyframes and aligns them in a visual layout reminiscent of a comic book style. Last *Schematic Storyboarding* uses motion arrows and captions adopted from traditional storyboards to depict contents of motions and camera movements in a scene. The presented approaches can be adapted in many areas of application. These include film analysis, video editing, browsing of large video databases or reviewing meeting- and lecture videos. We conclude with a comparison of the different approaches and a brief outlook.

SHOT AND SCENE ANALYSIS

A video contains several scenes. These are parts of the video which contain similar semantical content. Scenes consist of one shot or a series of shots. These shots are bordered by camera breaks, i.e cuts, whips or dissolves. Shots are uninterrupted continuous video sequences, which consist of a series of frames. These are single photographic images of a motion picture. A keyframe of a shot is a frame which represents the content of the shot best. A proper video summary contains shots and scenes as well, since a randomly cut video summary is rarely understandable. To generate a feasible video summary, shots and scenes need to be detected. We will describe a graph modeling technique presented by Ngo et. al. ([1]), which is a method to detect scenes, and

a shot detection technique presented by Cernekova et. al. ([3]), which is based on singular value decomposition.

Graph modeling

Graph Modeling is an effective scene detection technique based on temporal graph analysis. Both visual content and temporal structure of a given video contribute to the evaluation and detection of scenes. A series of graph-construction steps is applied, where clustering is based on visual content and scene detection on the temporal structure (cf. [1]).

Shot Detection and Clustering

Initially a set of shots S is obtained. Color, texture and statistical information of the input video is examined and spatio-temporal slices are generated to detect camera breaks. These are 2D-slices with dimension (x, t) or (y, t),where t denotes the time where a certain frame is shown and x and y represent the frame coordinates. To detect cuts, wipes and dissolves a pattern matching of these slices is done [13]. Based on these shots a keyframe is selected for further analysis. The gained set of shots represents the set of nodes of the first graph, which is constructed for the analysis. Let $G = (V, E)$, where G is a complete undirected graph with vertices $V = S$ and edges E with weight $w(i, j)$. The weight describes the similarity of shots i and j and is defined as:

$$w(i,j) = \exp\left\{\frac{-k \cdot \|f_j - f_i\|}{T} \cdot \mathrm{Sim}(i,j)\right\}$$

$\|f_j - f_i\|$ describes the temporal distance between shots j and i and T is the total number of frames in a video. The value k is an emphasis parameter. $\mathrm{Sim}(i, j)$ takes the color similarity of the keyframes of two shots into account. Let s_i and s_j denote two shots with keyframes $(r_{t1}, \ldots, r_{tk_t})$, $t \in \{i, j\}$, then $\mathrm{Sim}(i, j)$ is defined as (cf. [14]):

$$\begin{aligned}
\mathrm{Sim}(i,j) &= \frac{1}{2} \cdot (M(s_i, s_j) + \hat{M}(s_i, s_j)) \\
M(s_i, s_j) &= \max_{p=1,2,\ldots} \max_{q=1,2,\ldots}(\mathrm{intersect}(r_{ip}, r_{jq})) \\
\hat{M}(s_i, s_j) &= \hat{\max}_{p=1,2,\ldots} \max_{q=1,2,\ldots}(\mathrm{intersect}(r_{ip}, r_{jq}))
\end{aligned}$$

The function $\mathrm{intersect}(r_{ip}, r_{jq})$ compares the color histograms of the two frames r_{ip} and r_{jq}. For the similarity evaluation two values M and $\hat{M}$ are taken into account. $\hat{M}$ calculates the value of the second largest intersection ($\hat{\max} S$ computes the second largest value of the set S). This is done for reasons of robustness, since similar color histograms indicate related color distributions although their frames might have different contents. For clustering, the normalized cut algorithm is applied. This algorithm recursively decomposes G into subgraphs by partitioning it in each step into two new disjoint subgraphs A and B with $A \cup B = V$. A partition is to be found that minimizes the following value:

$$\mathrm{Ncut}(A,B) = \frac{\mathrm{cut}(A,B)}{\mathrm{assoc}(A,V)} + \frac{\mathrm{cut}(A,B)}{\mathrm{assoc}(B,V)}$$

where $\mathrm{cut}(A, B) = \sum_{i \in A, j \in B} w(i, j)$ and $\mathrm{assoc}(A, V) = \sum_{i \in A, j \in V} w(i, j)$. Two partitions with the least similarity are gained. Clusters are obtained by deleting the edges between the subgraphs in each step. The algorithm recursively partitions the graph until the similarity between all pairs of subgraphs drop under an adaptive threshold $T_s = \mu + \sigma$ which is the sum of the average and standard deviation of shot similarities. The last state of subsets describe the sought clusters.

Scene Detection

For scene detection a new graph is constructed. Let

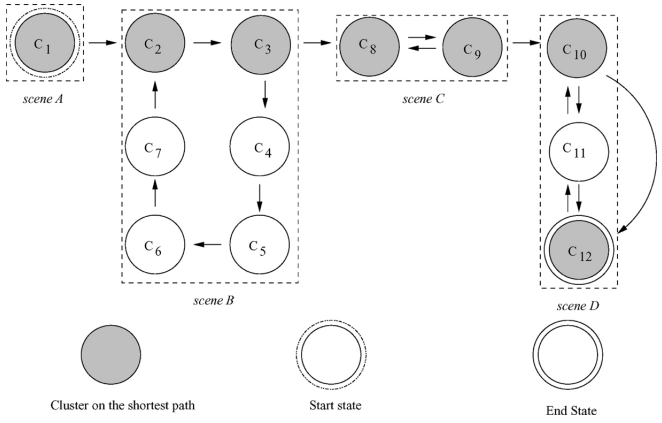

Figure 1. Temporal graph and scene change detection, [1]

$\bar{G} = (\bar{V}, \bar{E})$ denote a directed graph, where $V = \{C_1, \ldots C_n\}$ is the set of clusters obtained in the previous step and E consists of temporal connections between the clusters $\{C_i\}_{i \in \{1,\ldots,n\}}$. I.e. if there exists an order of scenes $\{\ldots, s_i, s_{i+1}, \ldots\}$ with $s_i \in C_x$ and $s_{i+1} \in C_y$ then $e_{x,y} \in E$ holds true. The cluster-node containing the first shot is defined as start state and the one containing the last shot is the end state. Such a temporal graph is shown in Figure 1. Scene detection is based on the interaction of clusters in time. If clusters belong to one scene, they will probably have some intertwined connection other than a pure sequential course of action. E.g. *scene B* in Figure 1 starts in cluster C_2. After various shots, a shot of cluster C_2 is shown again, although content of other clusters had been shown in between. The visited clusters probably have a semantical connection and therefore belong to one scene. According to this assumption a shortest path search from the start state to the end state based on the Dijkstras's algorithm is applied, by weighting the edges to 1. Now the edges connecting the shortest path are removed and the cluster-nodes that are still connected by some edges represent a scene. Graph modeling is an effective scene detection technique, but it still fails at some points. The clustering of shots is based on the similarity of the color histograms of frames and on the temporal distance between shots. If the input video shows the same visual content (e.g. a video of a speech or a talk) in the whole video, clustering is only dependent on the temporal distance of shots, which does not offer valuable clues to the semantical connection of the clusters. Another point where scene detection fails is when the visual content of a scene

does not alternate in turns, i.e. if a plot does not return to a cluster visited before. The problem which occurs here is that scenes are video sequences fragmented from the narrative point of view, which are hard to identify. A first step to solve these special cases could be the application of language understanding techniques.

Singular Value Decomposition

Different shots of a video can be detected by analyzing visual changes of the feature space. Each feature space contains many frames, which ought to be clustered. The problem we are facing at this point is that a video shows about 20-30 fps. A refined feature space is to be constructed from the large amount of frames. Singular Value Decomposition(SVD) is a technique to refine the feature space of a video based on the analysis of visual changes (cf. [2],[3]). The input of SVD is a Matrix $A \in \mathbb{R}^{m \times n}$, $n \geq m$. For each frame i a column-vector a_i is constructed, with $A = [a_1, a_2 \dots a_n]$, which represents a histogram in the RGB-color space. A frame is divided into several blocks and each block has its "color-bins". Each column of the matrix A describes a frame and each row describes the development of one "color-bin".

Matrix Decomposition

The assumption of the SVD-Technique is that linearly dependent or semi-dependent column vectors of A are frames of high similarity. The refined feature space consists of frames which have linearly independent color histograms. Given the matrix A, the SVD of A is defined as (cf. [2]):

$$A = U \Sigma V^T$$

U is an $m \times n$ column-orthonormal matrix, $\Sigma = diag(\sigma_1, \dots, \sigma_n)$ is an $n \times n$ diagonal matrix with $rank(A) = r$ and $\sigma_1 \geq \sigma_2 \cdots \geq \sigma_r \geq \sigma_{r-1} = \cdots = 0$, which are the singular values. V is an $n \times n$ orthonormal matrix. The smaller a singular value of a specific row is, the smaller is the difference to the other vectors. The singular values, which are multiplied with the rows have great affect on the similarity of the vectors. If you decide to refine your feature space to the size of κ you only use the κ largest singular values and set the other values to 0. Then again you discard the linearly dependent vectors.

Shot Detection(Clustering)

After obtaining the refined feature space the clustering is done. For clustering basically two values are important: a threshold and some sort of comparison value between two frames. There are several ways to define such a value. In [2] a distance metric is defined:

$$D(\psi_i, \psi_j) = \sqrt{\sum_{l=1}^{\kappa} \sigma_l (v_{il} - v_{jl})^2},$$

where $\psi_i = [v_{i1}\, v_{i2} \dots v_{ir}]^T$ of V^T, while Cernekova et. al. ([3]) use the cosine value of the angle between vectors v_i and v_j, so they obtain a value $\Psi \in [0, 1]$. Now an initial cluster is defined and each of the following vectors are "compared" to which cluster it fits best, based on the comparison value. If this vector does not satisfy the given threshold for any cluster a new cluster is defined containing this vector. One drawback of scene detection by SVD, which is based on the comparison of color histograms, is that color histograms are evaluated to be similar if their frames have related color distributions. This technique is not sensitive to local changes. E.g. a cut in a sports match could easily be missed, since the frames have similar color distributions. A dissolve could also be missed, since this technique is not sensitive to changing resolutions. On the other hand a shot could be detected by mistake, if many color changes occur in a shot e.g. a light is switched on.

Comparison between SVD and Graph Modeling

Both SVD and graph modeling are effective shot and scene detection techniques. SVD is a very precise method to evaluate similarity between frames. While SVD puts a lot of effort into shot detection by analyzing the feature matrix, graph modeling concentrates more on clustering and scene detection. Like the SVD technique, clustering relies on visual similarities, but the scene detection finally takes the temporal structure of the video and the interaction of the clusters into account. Scenes where visual content is repeatedly changed are still clustered to one scene, which is a great advantage of this technique. Shot detection relies on detecting cuts, wipes and dissolves by spatio-temporal slices. Since only few slices are used to evaluate the occurrence of camera breaks, this method lacks in precision. The generated patters, especially of wipes and dissolves, are easily confound with the motion of objects in a shot. Color information is very important to detect shots and it is used in the SVD technique in a very precise manner. But drawbacks mentioned in the previous section (SVD) could be improved by applying spatio-temporal slices. Nevertheless SVD is computationally intensive ($O(n^3)$), because it operates directly on the video's frames. By implementing the Dijkstras's algorithm graph modeling achieves scene detection in $O(n + e)$. Even the partitioning is easily calculated by turning the Ncut computations into a standard eigen system. SVD is much more precise in shot detection while graph modeling has advantages in runtime.

IMPORTANCE MEASUREMENT

The goal is to obtain a meaningful summary of a given video. A measurement is required which rates if shots and scenes should be preserved in the summarization and are important or attracting or if they do not contribute to the user's understanding of the movie and can be left out.

Simple Approach

A simple approach is to weigh the importance of a shot by taking length and uniqueness of this particular shot into account. To gain information about the shot and its context it is required to cluster the shots. This can easily be done with the graph modeling technique described earlier. Once n clusters have been detected, we can describe a measurement for the normalized weight of a cluster as

$$W_i = \frac{C_i}{\sum_{j=1}^{n} C_j}$$

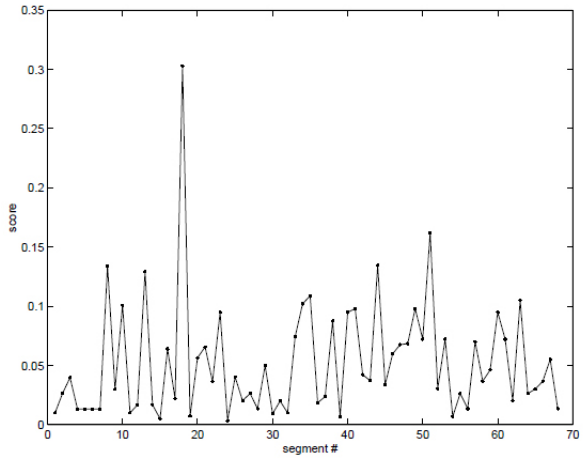

Figure 2. An example for the importance score, where the measurement is shown in respect to the according segment number, *[6]*

where C_i is the total length of the cluster i, received by summing the length of the single shots in that cluster. W_i is the proportion of shots from the whole video that are in cluster i. Now the importance for a single shot can be measured. W_i is used to describe the uniqueness of a shot assigned to cluster i. In fact the inverse of W_i is used for the calculation so that the importance measurement becomes smaller for a large W_i (meaning the shot is very common in the video sequence) and larger for a small W_i (meaning the shot is unique in his context). Assuming that long shots are important for a video sequence we also want to consider the length L_i of a shot i. These considerations lead to the importance measurement I_j for a shot j with the following equation:

$$I_j = L_j \cdot log \frac{1}{W_k}$$

The logarithm is a transformation commonly used in statistics to better fit the data. In Figure 2 you can see an example of a video sequence which is rated with an importance score for each of its segments. I_j can also easily be amplified with an additional factor. For example you could identify human faces in video shots, utilize the audio or even take the audiences' mean heart rate, if it can be detected, to amplify the importance measure.

Attention Models

Attention models are an importance measurement technique which are based on the users perception. Assuming that loud noises, strong motion changes in scenes or shots containing people raise the user's attention Ma et. al. ([4]) describe various attention models to detect the important parts.

Visual Attention Models

The visual attention model can be influenced by motion, camera action, contrasts or people appearing on the screen. We will introduce four of the visual attention models proposed in [4]. The *Motion Attention Model* is based on the construction of Motion Vector Fields (MVF). MVFs describe the movement from the current frame to the next one. Figure 3 shows motion vectors pasted on a car driving scene. You can easily make out the direction the cars are driving in. This image illustrates the difference between moving and static objects, since vectors with higher length, e.g. those positioned at the center of a car, point out that the objects have moved to a perceivable distance in a few frames. Static areas, like the street, are associated by points or very short vectors, since they do not provide any movement. This MVF is passed to three inductors, which project the MVF to one of three maps. First the intensity inductor is applied, which

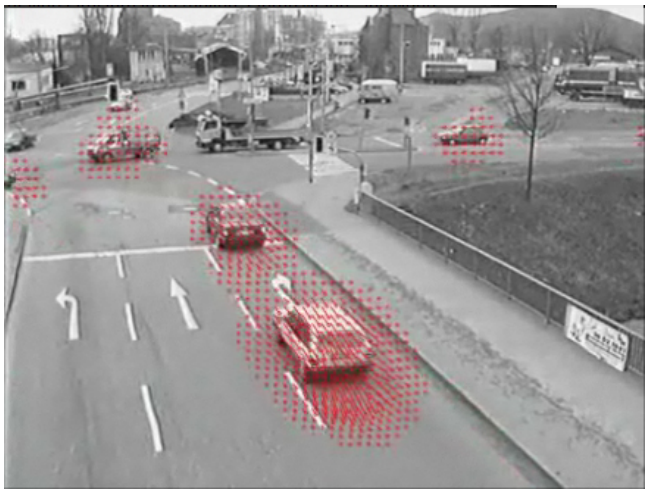

Figure 3. Motion Vector Fields, *www.aharmat.com*

relies on the intensity of movement, activity and motion energy. A good measurement for the intensity of movement is the length of the vectors. The intensity inductor is defined as:

$$I(i,j) = \frac{\sqrt{(\mathrm{d}x_{i,j}^2 + \mathrm{d}y_{i,j}^2)}}{\mathrm{MaximumMagnitude}}$$

The maximum magnitude is the maximal length of a vector in the MVF. The spartial coherence inductor extracts which areas are consistent and might belong to one object and which are inconsistent and probably are located at the edge of an object. Here a Matrix C_s is computed, which holds information on the consistency of an area. The temporal coherence inductor, represented by a matrix C_t, holds information on the movement in an area over a given set of frames. The three inductors are fused to a saliency map, as seen in Figure 4. Salient regions in a saliency map appear brighter, so the average brightness of a saliency map is computed in order to find out which frames contain the most motion activity. Not only motion may attract the user's attention. The *Static Attention Model* analyses color contrasts, intensity contrasts and orientation contrasts by creating a saliency map based on these parameters. Human faces draw the user's attention, so that a *Face Attention Model* needs to have impact on visual attention modeling. A face detection software is applied, which returns information on size, position and number of faces detected in a frame. Camera motion is utilized to attract the viewers attention to certain things. This is the next important visual attention model: The *Camera Atten-*

Figure 4. Motion attention detection. From left: intensitiy inductor I, spartial coherence inductor C_s, temporal coherence inductor C_t, concluding saliency map, original video frame with the attention area marked by the bounded box. [1]

tion Model. Camera motion is divided into the following classes: Camera rotations: Panning and tilting(x- and y- coordinates), rolling (z-axis), camera displacement: Tracking and booming (x, y-axis), dollying(z-axis), zooming and still. The camera attention model is multiplied with other attention models and used as a magnifier. Camera actions are assigned to specific values. For example, since zooming is most likely used to attract attention, zooming and especially fast zooming is assigned to a value greater than one. On the other hand horizontal panning neglects objects shown in the scene, so its assinged value is less than one.

Audio Attention Models

Loud noises, music or e.g. human laughter highly attract the viewer's attention. Another important attention model needs to be mentioned: The *Audio Attention Model*. There are three important audio attention models mentioned in [4]. The *Audio Saliency Model* expresses the loudness of sound since it is the most effective way to measure salience in sound. Loudness of sound can be represented by energy. People pay attention to both overall loud scenes and sudden decreases or increases of loudness. The average energy of the given audio segment is computed and energy peaks are detected to be merged to the audio saliency attention model. The audio saliency model is defined as (cf. [4]):

$$M_{as} = \bar{E_a} \cdot \bar{E_p}$$

$$\bar{E_a} = \frac{E_{\text{avr}}}{\text{MaxE}_{\text{avr}}}$$

$$\bar{E_p} = \frac{E_{\text{peak}}}{\text{MaxE}_{\text{peak}}}$$

E_{avr} denotes the average energy and E_{peak} the energy peak of an audio segment. MaxE_{avr} denotes the maximum average energy and $\text{MaxE}_{\text{peak}}$ the maximum energy peak of the entire audio sequence. Thus $\bar{E_a}$ and $\bar{E_p}$ are the normalized average energy and the normalized energy peak in an audio segment. The audio saliency model is used as a magnifier for other attention models. Speech and music attract the viewers so that a *Speech Attention Model* and a *Music Attention Model* is generated. Music is often used to underline a certain atmosphere in a video sequence or to emphasize important scenes. Speech often contains important information, which contributes to the content of the video. Sequences with high rates in speech and music need to be found. This can be measured by the ratio between speech and music and other sounds. For speech and music recognition the audio-stream has to be segmented and analyzed. This analysis leads to a classification into speech, music, silence or other sounds. On that basis the rate of speech or music in a scene can be evaluated.

Attention Curve

To evaluate the given shots and frames of a video, the mentioned attention models are fused to a single attention model A by linearly combining them. All described models are weighted by a factor, which influences the impact a single model has on the whole model, and are added to each other. Now each frame is assigned to an attention value based on A. These attention values are fused to an attention curve, by smoothing the connections between the single attention values, and are mapped to the time when corresponding frame is shown in the video. Since high attention values assign important scenes, crests of the attention curve are detected. This is done by analyzing the curves derivative to find zero-crossing points.

Comparison between Attention Models and Simple Approach

Attention Models and the Simple Approach both rate the importance of a scene. A great advantage of the simple approach is that it is rather easy to calculate, since it is merely based on cluster length and uniqueness. On the other hand these values are not very reliable to evaluate importance. The attention model is a more sophisticated approach and has many computation steps. A lot of information needs to be gathered and analyzed. But the attention models refer to the user's perception, so it is more likely that the attention models release a satisfying result. But they still remain models, which means that their calculated results may not necessarily be most interesting to the user. Imagine a video sequence where only the shadow of a person is depicted to show his presence. This sequence would not get high attention values, although it might be very important for the content.

EXTENDED FRAME

A video shot is not a static image. Camera movements like pans or zooms make the background vary over time. In these cases it might be helpful for video summarization to capture the background of a shot in one picture. Goldman et. al. ([8]) developed a technique that generates a continuous picture out of frames from the same shot. For this approach, user input is required for selecting key frames and identifying and labeling according features in the frames. The goal now is to transform images selected by the user with uniform scales and translations so that they result in a seemingly continuous image. Rotation is intentionally omitted in the transformation, as rotations of the frames can lead to a

false impression of the camera movement. An example of this is shown in Figure 5, where you get the impression that the camera is rolling with respect to the horizon, whereas it moves constantly. This gives poorer geometrical alignment but a more accurate impression of the camera movement in the end. A user has to pick different features of

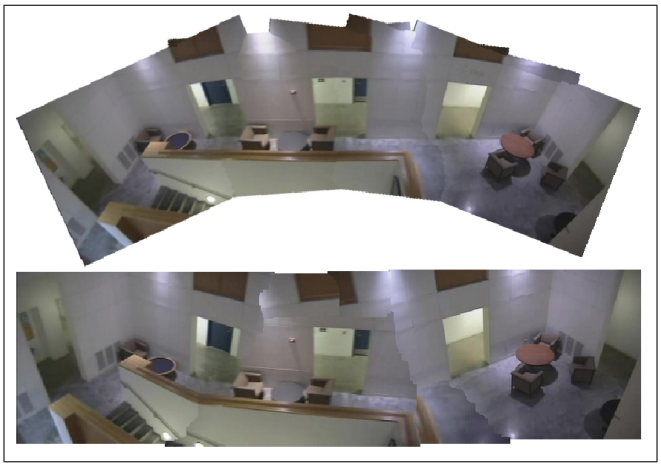

Figure 5. Top: Transformation with rotation. Bottom: Transformation without rotation, *[8]*

an image that belong to the background. The location of these features are denoted as f_{ik}, where k is the label of the feature and i the corresponding frame. To align the frames best, we aspire a transformation between the sets of points $f_i = \{f_{ik} \forall k\}$ and $f_j = \{f_{jk} \forall k\}$ with the lowest standard deviation. Applying only a uniform scale and translations can cause problems. Imagine a scenario where two points lie on a horizontal line in the first frame and on a vertical line in the second, so that the camera undergoes a 90 degree rotation. The scale factor that provides the best results in the least-squares sense would be 0, although the images were not necessarily changed in size. To prevent this undesired effect, a refinement of a method introduced by Horn [9] and refined by Umeyama [10], in which the optimal rotation R is substituted with the identity matrix, is used. The transformation $M_{i \to j}$ for a frame i consists of a scale-function $s_{i \to j}$ and a translation function $t_{i \to j}$. For these, the centroids $\bar{f}_i$ and the standard deviation σ_i of the selected feature points are needed:

$$\bar{f}_i = \sum_k \frac{f_{ik}}{n}$$

$$\sigma_i = \frac{\sum_k \|f_{ik} - \bar{f}_i\|^2}{n}$$

Now the relative scale $s_{i \to j}$ between frames i and j can be computed as the the quotient of σ_j and σ_i (the standard deviations of the feature points). With the translation function $t_{i \to j}$ we shift frame j in the coordinate system of the extended frame, so that the centroids of frame i and frame j lie on top of each other. We subtract the centroid of frame j with the scaled centroid of frame i. These considerations lead to the following equations:

$$s_{i \to j} = \frac{\sigma_j}{\sigma_i}$$

$$t_{i \to j} = \bar{f}_j - s \cdot \bar{f}_i$$

The transformation $M_{i \to j}$ consists of these two functions, where i and j are two temporarily adjacent frames. $\bar{f}$ and σ are recomputed for each pair of frames with the intersection of feature points in i and j. The transformation M_i, that positions the frame j in the coordinate system of the extended frame, can be denoted as follows, where I is the identity matrix:

$$M_0 = I$$

$$M_i = M_{(i-1) \to i} \circ M_{i-1}$$

This is not an optimal solution, as an error will be reproduced in the following frames (cf. [8]). But tests have shown that it is acceptable for such a small amount of images. To composite the different frames the Photomontage approach of Agarwala et. al. [11] is used. This determines the locations of seams between frames.

APPROACHES

There are two main techniques to summarize a video: Static and dynamic summarization. In our research we found "Comic Book Representation" and "Storyboarding" as representatives of static summaries and "Video Skimming" as a dynamic technique.

Dynamic Video Skimming

Dynamic video skimming is a video summarization technique which shortens a long video to a concise version. This is done by attaching seemingly important and informative scenes to another. The summarization of these short clips is supposed to give an overview of the video so that the viewer knows the important facts of the video without watching it in full length. A skimmed video needs to fulfill certain criteria. If a user decides to watch the concise version of the video, his goal probably is to gain as much information as possible in short time. It is important that the selected scenes contribute to the content or the atmosphere of the video and are enjoyable to watch. E.g. a viewer would probably prefer to watch a discussion between two people than a landscape shot. It is important that we do not cut out random parts of a shot or a scene and fuse them to a summarization. This would result in a fully not understandable collection of shots. You cannot simply shorten a video at a certain skim ratio. Shot and scene boundaries need to be detected and the most affecting scenes will be selected. We will discuss how the described techniques can be applied to summarize a video which fulfills the given criteria as good as possible. A video is skimmed with respect to its shot boundaries, so that the skimmed version is not only a collection of shots. Thus shot and scene boundary detection is applied. In the first section we described two clustering techniques: The very precise SVD technique and the temporal graph analysis. Both can be applied to find such boundaries. You can even find scene boundaries with graph analysis. The next

important aspect is the attraction or importance of the selected scenes. With the help of an attention model or an attention curve, which is shown in Figure 6, it is possible to detect these shots. Either peaks in the timeline are chosen and the shots or scenes that refer to these crests will contribute to our summary, or attention models are fused directly to the clusters without constructing the attention curve. What is still needed as user input is the skim ratio, which has great affect on the summarization. A high skim ratio may exclude relevant content from the summarization while a low skim ratio has nearly no effect. User studies have found out that a skim ratio of 25% conclude in convincing results. Once the skim ratio r is defined, the most important and attracting $r\%$ of the video are selected. We will present two skimming techniques that fulfill the given criteria rather well. The first technique is based on the attention curve (cf. [4]). Here every selected sequence for the summarization is supposed to have about the same length. Once the number of peaks n is found in the curve, we can evaluate how long every sequence shall be. Lets say L is the total length of a shot and r is the skim ratio, then the sequence-length surrounding a crest l is computed the following way: $l = L \cdot \frac{r}{n}$. The example shown in Figure 6, where two peaks have been found, defines a skimming rate $r = 25\%$. We compute $l = L \cdot \frac{25\%}{2} = L \cdot 12,5\%$. This may lead to very short sequences if the number of crests is high enough. A constant l_{min} is defined as the minimum length of a sequence. To achieve this length, keyframes with the smallest attention values are removed until the proposed length is reached. So parts of shots are cut out. To avoid annoying effects, sentence boundaries are detected and cuts are only applied after a sentence is finished, thus the detected boundaries may be moved as you can see in Figure 6. The second technique we will present refers to the temporal graph analysis and attention values described in [1]. By applying the graph-analysis technique, the video is segmented into shots, clusters and scenes. To skim this video to a skim ratio r, four algorithmic steps are applied. If the resulting video length is conform to the skim ratio, the algorithm terminates. First a skimming on scene-level is done by summing up all attention values

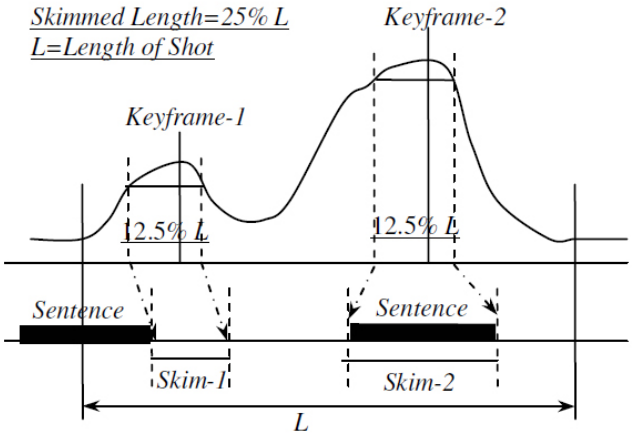

Figure 6. Attention curve with skimming rate of 25%. Shows a video sequence of length L with two peaks and the resulting skim-sequences Skim-1 and Skim-2, which are adjusted to the sentence boundaries.*[4]*

a_w, with $w \in \{1, \ldots, |C| \, | C \in S\}$ of the clusters contained in a scene, which are weighted by the percentage the cluster has in the video, and divided by the number of clusters in this scene, to a value Q. All scenes S, which's $Q(S)$ values are less than $\alpha \times \mu \times (1 - r)$ are discarded, where μ is the average Q value of all scenes and α is a control parameter. If the video is still too long we move to the cluster-level. Here the quality of a cluster is evaluated by

$$QC_w = \frac{a_w}{\sum_{C_j \in S_i} a_j},$$

where S_i is the current scene. Now a subset of clusters in S_i, ordered by its QC values, is constructed, which satisfies

$$\sum_{C_j \in S_i} QC_j \geq (1 + r) \cdot \frac{Q(S_i)}{\sum_k Q(S_k)}.$$

The subset of clusters that fulfills this inequation is selected, the other clusters are discarded. Again the most important clusters are selected, regarding the importance of the scene. If a scene has a higher quality it will contain more clusters than a less important one. Since we assume that we have not achieved the skim ratio r yet, we move on to the next step, the shot level. Again it should be avoided losing the connection to the previous level, so the cluster quality is taken into account. The attention values $a(s_j)$ of all shots s_j of a cluster are ordered in a descending manner by their attention values. As before a subset of shots is chosen which satisfies

$$\sum_{s_j \in C_i} a(s_j) \geq (1 + r) \cdot QC_i.$$

In the last step subshots are discarded. Shots are ordered ascending by their attention value. One after another the shots are skimmed so that only the most important subshots remain until the desired skim ratio is achieved. In this approach the qualities are measured by attention values. You could also use the simple approach, which we described in one of the previous sections. In order to obtain more sophisticated results, the attention models were chosen in this algorithm.

Comic Representation

One approach to represent a summary of a video is a representation in a comic book style. Single keyframes taken from the video are ordered and resized in a two dimensional layout that is similar to a comic book. Based on the different sizes of the keyframes, the viewer sees which of the according shots are relevant for the whole video and which are not at first glance. The comic book representation uses the simple approach, discussed before, to rate the shots by their relevance. This score is used to resize the keyframes accordingly, where large keyframes stand for high scores and small sizes for low scores. In this approach three graduations of keyframe sizes are used (the smallest, twice the smallest and three times the smallest). Following thresholds are used to classify the shots: A score under $\frac{1}{8}$ of the maximum score is considered irrelevant and does not show up in the summarization. For a shot with an importance score between $\frac{1}{8}$ and $\frac{1}{4}$ of the maximum score, the smallest keyframe size is chosen. Between $\frac{1}{4}$ and $\frac{1}{2}$ of the maximum score, the keyframe

Figure 7. Interface of the Comic Book Representation, *[6]*

is sized twice the smallest and between $\frac{1}{2}$ of the maximum score and the maximum score, the biggest keyframe size is chosen: three times the smallest. The keyframe that represents a shot is simply taken from the middle of that shot. With only three different sizes of keyframes, it can be ensured that they can be efficiently packed in the layout with as little whitespace as possible. A grid system is used, where one grid can hold the smallest keyframe. An algorithm then tries all possible keyframe packings for a row (temporal order must be observed) and picks the packing with the least keyframe resizing. This exhaustive algorithm is not very efficient but easy to implement. Considering the limited number of keyframes, it is sufficient for this purpose. Because of the importance threshold some keyframes are not shown in the summary and this can generate redundancy.

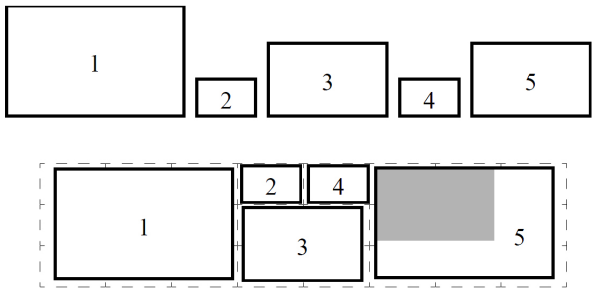

Figure 8. How keyframes are packed: As much whitespace as possible is avoided, in this example keyframe 5 is enlarged additionally in the final packing. *[5]*

A frame between two keyframes from the same cluster is not listed because of its low importance score, for example. The two keyframes which are essentially depicting a likewise image, as they are from the same cluster, are then standing next to each other, creating redundant information. To prevent this scenario a heuristic is employed that iteratively deletes the duplicate image (the smaller or latter image will be deleted). Another undesired scenario for example would be a long dialogue, where redundant small images from two clusters would alternate in the representation. If two keyframes from the same cluster are divided only by one keyframe, the smaller or latter keyframe will be deleted. If the keyframes were of the same size, the remaining keyframe will be enlarged if possible. These two heuristics make the representation more concise without compromising the effectiveness.

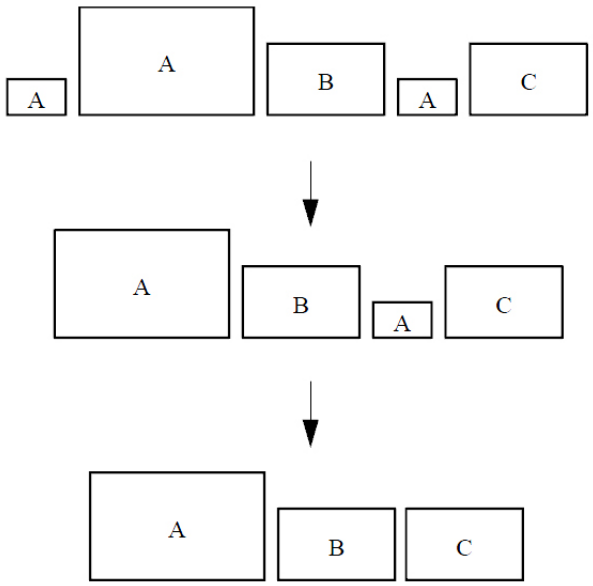

Figure 9. The two heuristics are applied to reduce further redundancy, *[7]*

Storyboarding

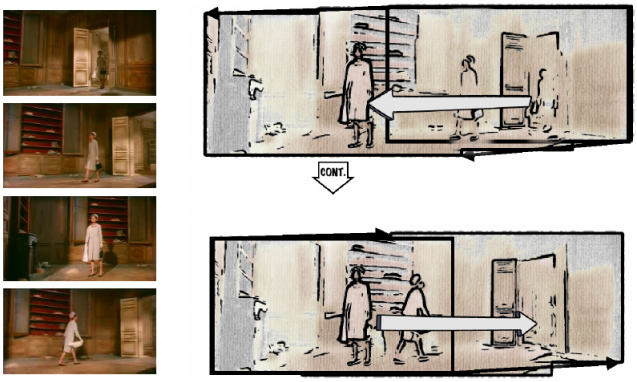

Figure 10. A computed storyboard, where additionally a sketch filter is performed, *[8]*

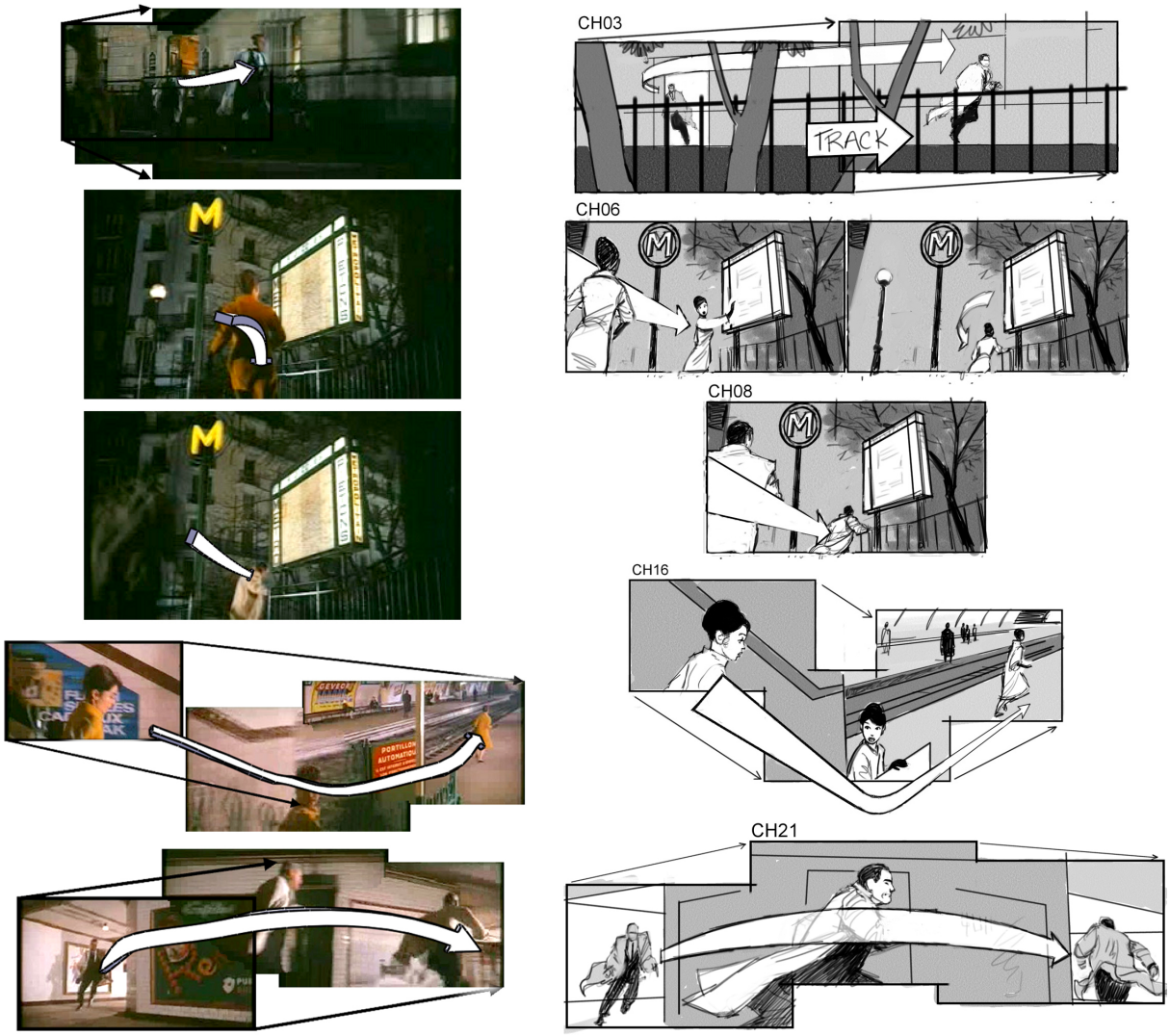

Figure 11. Comparison between computed and hand drawn storyboard, *[8]*

For nearly every video that is being produced professionally a storyboard is drawn beforehand. A storyboard can be compared to a large comic depicting what you actually want to show in your video. Besides the background and actors themselves, this includes arrows and captions that indicate motion and camera settings. Normally a shot is represented by not more than one or two storyboard panels, so you can absorb the motion and setting of the shot at one glance. Thus generating a storyboard of an existing video should give a pretty concise summary, compared to a single frame representing the whole shot for example. As a foundation for the storyboard representation, an extended frame, as described previously, is created with some supplements. The first supplement is to analyze moving subjects in the shot, so that they can be ordered according to their depth. This means that if a person for example appears in multiple images of which the extended frame is compounded, we want to make sure that the image of the person standing nearer to the camera overlaps an image of the person standing more in the background. The first problem is to extract the person from the background and the second is to find he right z-order between the different images of that person. As compositing is addressed as a labeling problem in this approach, user input is utilized to solve these problems. To segment subjects that appear in the foreground, the GrabCut matting algorithm (cf. [12]) is used. GrabCut is an algorithm that segments an image using an optimized graph-cut algorithm on the texture (color) and contrast information of the image. The user only needs to draw a rectangle around the subject and an algorithm computes the desired matte. Now that the subject is exempted from the background, the size is used as a heuristic to determine the z-order of the subject, assuming that larger objects are nearer to the camera than smaller ones. This is not generally true if you think of a flat screen for example. Viewed frontally its size is much bigger than viewed from the side, although the distance to the camera might not have changed. But this fact is negligible for this approach, as it works just fine in most cases. The next sup-

plement added is a system to split an extended frame of one video shot into more parts, if necessary, to make sure that the extended frame stays viewable. There are two scenarios we want to avoid when generating a continuous image of frames. It can happen that the camera performs large zooms, meaning that the size difference between the largest and the smallest frame becomes very big, even as big that the content of the smallest frame couldn't be perceived anymore. To avoid this a scale test is applied where the extended frame is split as soon as the size ratio between the largest and smallest frame is greater than a certain threshold. The second undesired scenario is when a subject crosses its path in the extended frame coordinate system so that different moments in time would obscure each other. This would happen if a person for example walks from left to right and then back from right to left. To determine the scenarios where a path would cross itself, an overlap test is performed along the path. The test fails when a certain amount of pixels of the successive time intervals overlap. In [8] they used $T_s = 1.5$ as a threshold for the scale test and $T_o = 0.25$ for the overlapping test. The tests are performed after every new frame that is added to the extended frame. If one of them fails, the current frame is set as the last frame of the current extended frame and also as the first frame of the new composite. This is a help for the user to keep the thread. Additionally an arrow is added between the two extended frames that points on the successive frame and is labelled accordingly to show users the following frame. To understand in which direction the camera moves, an algorithm additionally outlines the first frame with a boundary and draws arrows from each corner of the first frame to the last frame. Arrows are omitted if they would intersect lines of the boundaries. An essential part of

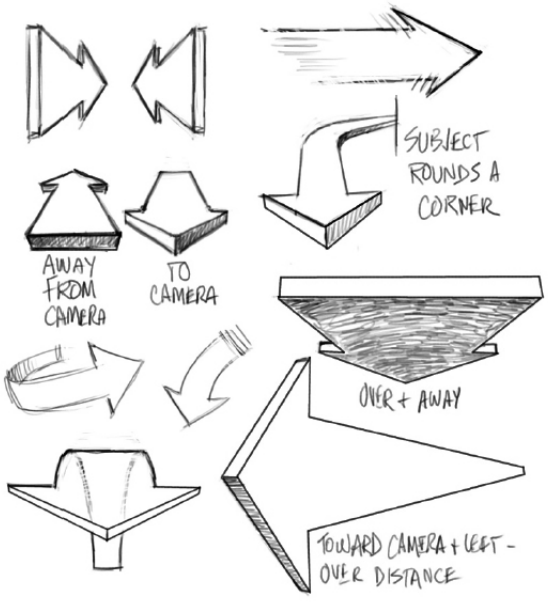

Figure 12. Examples of different arrow styles utilized in storyboard, *[8]*

a storyboard are arrows that describe the motion of a subject in a shot. These arrows are normally drawn in a three dimensional space. They should give an overview of the main motion that is happening in the shot. Small motions that are not noticeable or self-explanatory are not reflected in an arrow. You can see some examples of such arrows in Figure 12. The only data available to draw such arrows onto the extended frame is gained from the input provided by the user. The user must label the selected subject features if not belonging to the background, but specifically to this subject. A walking person can be considered as one subject, but also a large group of people moving in the same direction would be one subject. As no values for the subject's position on the z-axis are available, a pseudo-3D estimation will be generated. For this purpose we can make the assumption that an object is large if it is close to the camera and smaller if it is farther away from the camera. As a value for size we can use the standard deviation of the subject's feature points. If the distance of the subject to the camera is known for one frame, the distance in the other frames can be computed for the subject, using similar triangles: $\frac{d_j}{d_i} = \frac{\sigma_j}{\sigma_i}$, where σ is the standard deviation and d_i the distance of our subject to the camera in frame i. Only the distance of the subject in one frame must be declared now. In the schematic storyboarding approach they used a constant of $d_{min} = 500$ pixels for the smallest distance. As x and y values in the extended frame coordinate system the corresponding values of the centroid $\bar{f}_i$ can be used. Now a B-spline, which has a very "smooth" look, is laid along the 3D coordinates of the subject and builds the backbone for a motion arrow. To give these images the feeling of a storyboard even more, a non-photorealistic filter can be applied to give it a sketchy look. The storyboard approach gives users much information about a video shot at a glance. But utilizing the arrows as a timeline, the storyboard can also be used to scrub through a video. Dragging along an arrow for example gives you the feeling as if you would direct the camera.

EVALUATION OF APPROACHES

So far we have presented three different approaches to summarize a video. Now we will evaluate the different techniques and weigh the pros and cons. In contrast to the other approaches, dynamic video skimming transfers the most content information. Nevertheless this also makes it more time consuming since you still have to watch the whole skimmed video. Also if content was left out in the summary it can only be watched in the original video. On the other hand audio and interaction sequences and discussions can be perceived in a dynamic summarization and skimming is applyable to all kinds of videos. The more static comic book representation and the storyboarding both give an overview of the video at a glance. But on behalf of the static keyframes of the comic book representation, you can only assume what will happen. Storyboarding on the other hand gives information on interaction through motion arrows. Both approaches provide printable versions. The comic book representation provides the facility to be able to select the parts a user wants to see by himself. Important scenes are highlighted through their size, which makes it easier for the user to decide which scene he wants to watch. However comic book representation is only useful for long videos with alternating visual content. Nevertheless it has an appealing layout and pro-

vides a comic book interface that users are familiar with. Storyboarding has a very intuitive and pleasing visual representation. In contrast to the comic book representation, a whole time interval is transferred to one image and thus it contains more content. But storyboarding needs a lot of user input, and therefore it is very time consuming to generate this kind of summary. This approach is only worth applying if the summarization is seen more often. A very useful feature is that storyboarding provides the opportunity to use the motion arrows for scrubbing through the videos. All in all we think that for general purpose (videos with a lot of content and and alternating scenes) the comic book representation is the most useful tool, because it gives a concise overview on the content and still leaves the freedom to access all of the video's content.

OUTLOOK

Concluding we would like to point out which features can be improved. A big disadvantage of storyboarding is the large amount of user input. Hence an automated approach where algorithms complete the user's work should be constructed. The static keyframes in the comic book representation are not very efficient. If they where for example replaced by the storyboard view of that shot, the user could better decide which clip to watch. Furthermore the interaction between dynamic and static view could be more blurred, e.g. playing a skimmed version of a shot as the user hovers over the corresponding keyframe with his mouse. Skimming on the other side could offer an option for playing skipped scenes, since the skimming technique is fully automated and it is not guaranteed to provide the optimal summarization. You could think of a specialized timeline for example. Furthermore user studies on the skimming rate could be accomplished to be able to define a skimming rate for special purpose, fitting to the according application area. For example lecture videos might need a higher skimming rate than some hollywood movie.

REFERENCES

[1] Chong-Wah Ngo, Yu-Fei-Ma, Hong-Jian Zhang "Video Summarization and Scene Detection by Graph Modeling," *IEEE Transations on circiots and systems for video Technology, VOL. 15,NO.2, Feb. 2005*

[2] Yihong Gong, Xin Liu "Video Summarization Using Singular Value Decomposition," *C&C Research Laboratories, NEC USA, Inc.*

[3] Z. Cernekova, C. Kotropoulos, I. Pitas, "Video Shot Segmentation using Singular Value Decomposition," *Department of Informatics, Aristotle University of Tessaloniki*

[4] Yu-Fei Ma, Lie Lu, Hong-Jiang Zhang and Mingjing Li, "A User Attention Model for Video Summarization," *Microsoft Research Asia*

[5] John Boreczky, Andreas Girgensohn, Gene Golovchinsky, and Shingo Uchihashi, "An Interactive Comic Book Presentation for Exploring Video," *FX Palo Alto Laboratory, Inc.*

[6] Shingo Uchihashi and Jonathan Foote, "Summarizing Video Using a Shot Importance Measure and a Frame-Packing Algorithm," *FX Palo Alto Laboratory, Inc.*

[7] Shingo Uchihashi, Jonathan Foote, Andreas Girgensohn, John Boreczky, "Video Manga: Generating Semantically Meaningful Video Summaries," *FX Palo Alto Laboratory, Inc.*

[8] Dan B Goldman, Brian Curless, David Salesin, Steven M. Seitz, "Schematic Storyboarding for Video Visualization and Editing Video Summaries," *University of Washington, Adobe Systems*

[9] Berthold K. P. Horn, Hugh M. Hilden, Shahriar Negahdaripour, "Closed-Form Solution of Absolute Orientation Using Orthonormal Matrices," *University of Hawaii at Manoa*

[10] Shinji Umeyama, "Least-Squares Estimation of Transformation Parameters Between Two Point Patterns" *IEEE Transactions on Pattern Analysis and Machine Intelligence 13, 4, 376-380*

[11] Aseem Agarwala, Mira Dontcheva, Maneesh Agrawala, Steven Drucker, Alex Colburn, Brian Curless, David Salesin, Michael Cohen "Interactive Digital Photomontage" *University of Washington, Microsoft Research*

[12] C. Rother, V. Kolmogorov and A. Blake, "GrabCut - interactive foreground extraction using iterated graph cuts" *ACM Transactions on Graphics*

[13] Chong-Wah Ngo, Ting-Chuen Pong, Roland T. Chin, "Video Partitioning by Temporal Slice Coherency" *IEEE transactions on circuits and systems for video technology, vol. 11, no. 8, aug. 2001*

[14] Chong-Wah Ngo, Ting-Chuen Pong, Hong-Jiang Zhang "Motion-Based Video Representation for Scene Change Detection", *International Journal of Computer Vision 50(2)*