Automatic Video Segmentation by Polygon Evolution

Dissertation
zur Erlangung des Doktorgrades
der Fakultät für Mathematik, Informatik
und Naturwissenschaften
der Universität Hamburg

vorgelegt
im Department Mathematik
von

Daniël de Wildt

aus Heerlen (Niederlande)

Hamburg
2006

Als Dissertation angenommen vom Fachbereich
Mathematik der Universität Hamburg

auf Grund der Gutachten von Prof. Dr. U. Eckhardt
und Prof. Dr. L. J. Latecki

Hamburg, den 30. Juni 2005

Prof. Dr. A. Kreuzer Dekan des Fachbereichs Mathematik

Contents

Chapter 1

Introduction

1.1 Introduction

Key frames are the most natural and convenient representation of video content since they reduce video information to a concise form of a small number of still images. Key frames are used in many domains of the video content analysis. They are very useful for human visual perception since one can easily judge the content of a video by viewing its representation through a small set of key frames. It is obvious that key frames are used for a video representation like summaries or for search operations like Content Base Image Retrieval. In all of these cases, a representation of the video by a small set of frames is required.

1.2 Motivation

Key frame detection is widely used in literature, applications and in practice. Radke [47] has a reference list with a wide range of different applications for these classes of video sequences like *Video Surveillance* [10, 50], *Remote Sensing* [7, 9], *Medical Diagnosis and Treatment* [6, 46], *Civil Infrastructure* [33, 43], *Underwater Sensing* [22, 40] or *Driver Assistance Systems* [25, 30].

Videos contain a very large amount of data even if they are reduced with modern compression algorithms like MPEG. It is not always applicable to view a video directly online or search it's contents in an appropriate way. A long download time and bandwidth are necessary to get all of the content in an acceptable time. It is not obvious, with the availability of only a few

minutes of the video, whether the contents are desirable or not. This results in a waste of time, bandwidth and money.

There exist many well-known multimedia content searching techniques, like skipping of CD tracks or DVD chapters. Playing the first few seconds of music tracks as an introduction to the song is a normal feature for CD players and audio applications. Also some online shops make parts of songs (samples) available for their customers on the internet to give an impression of the content. We also have the same possibilities for the media video. Playing the first few seconds of a DVD track is not always representative for the track and downloading a sample from the internet often requires a lot of time.

The "normal" way to get information about the video content is by viewing trailers or skipping through the video with a remote control. The user is searching for important events or shot changes and if such an event is found, he will view the video at normal speed to get an impression of the content of the event. This kind of searching through the video content implies the availability of the content and an abstract of it. An alternate possibility is to read a review of the video which could be emphasized with images of the described to get an impression of the content. In both cases, a short representation of the video is necessary. We distinguish two basic types of such representations: dynamic — a video trailer, and static — a video summarization or storyboard composed of a few key frames. The key frames are the fundamental part of a summarization because a trailer could be based on shots while are based on the key frames.

All these components of creating an abstract of unknown video content assume that a usable abstraction with key frames of the video exists describing the desired and expected content. Different purposes of abstract creation assume different kinds of abstract content. A video trailer should consist of an abstract with all important parts of the video and it should also often create an inquisitiveness for the video. These aims are not always identical or representable by an identical set of key frames. For the creator of an abstraction, it is important to have the ability to select different abstraction levels, from which he can select the desired parts of the content to match the expectations of the abstraction that should be created.

Another example of an application in which video key frames are used is the representation of a video search result like it is done by the video engine of altavista [2][1]. One of the possibilities for altavista's search engine is to search for videos by entering key words. Altavista's search engine searches

[1]This page is available at *http://www.altavista.com/video/*

internet for videos which are linked to or integrated in websites and which associate the words of the website to the video. A semantic search engine searches for these key words and presents the found videos with representing a frame of the video[2]. It happens frequently with this kind of video content presentation that the searcher does not find the correct video because it is represented by incorrect, ineffective or insufficient video content. Either someone needs many attempts to get a representable frame or this person loads every video of the search result. These results are often poor because the content could be anything.

It would be more effective for the searcher to get a list of key frames of the video presented to get a better impression of the expected content of the video[3]. Figure 1.1 shows the resulting frame for a search on personal security as shown by altavista. It is not apparent what exactly the content of the video is. Someone could get the impression that the video has only ear protection information. Figure 1.2 also shows us that other content like head protection and appropriate clothes is contained.

Figure 1.1: Original example frame for a short video introducing visitor protection.

Figure 1.2: Example of three representative frames of the same video.

[2]This seems to be the center frame of the video.

[3]The search engine company *google* has an online news broadcast search engine which provides a result with different key frames. The website is *http://video.google.com/* and is in a beta test stadium.

1.3 Classical temporal video segmentation and indexing algorithms

There exist many techniques for key frame extraction. Most of them segment videos at first into shots. Shot or video scene-change detection is a well-studied topic [1, 24, 31]. These algorithms are based on fast changes between successor frames but gradient transitions between shots are also detectable with additional analysis of the found shots [24, 26]. Such shots are represented by frames from inside these shots. This could be a simple process which is implemented into the detection algorithm or which is implemented by an additional process that analyses single shots and picks more or less intelligently a representative frame. The quality of these detection algorithms depends on the quality of the shot detection algorithm and the merging process. Higher abstraction levels are found by merging separate shots together into groups [53]. This process has a higher processing overhead due to the shots comparison. This also depends on the quality of the shot detection process. A group representation by a few frames also needs an additional frame selection process to find representative key frames for this group of shots. Other efforts for key frame extraction are clustering algorithms which join single frames to frame groups. This results in groups of individual frames which are representative for the video sequence [21, 48].

The low-level algorithms are based mostly on three steps: the extraction of image features, the calculation of neighbor image differences and the detection of shot boundaries. In an abstraction process, these detected shots are represented by representative images, the key frames. The selection algorithm could vary from easy (select the n-th frame of a shot as key frame) to complex algorithms (for example, select that image from the shot which is most similar to the images of the other frames in the shot).

1.3.1 Algorithm of Kumar et. al

Rajeev Kumar and Vijay Devatha describe in their manuscript, "A Statistical Approach to Robust Video Temporal Segmentation", an algorithm for video shot boundary detection [31]. They use a key frame detection which is based on a shot detection algorithm with frame descriptors based on a subdivided weighted grey color histogram. The histogram values are based on weighted pixel values. Pixels in the inner areas are higher weighted and in the outer areas are lower weighted to show the importance of the inner

parts of the viewable area. The sum of the weighted pixels is normalized. A Gaussian window filter is applied over the histogram intersections to achieve a robustness in the histogram-matching algorithm. The likeness between images is calculated with a matching metric which detects differences in two images based on the histograms. Kumar and Devatha used the Bhattacharrya metric [4], which is a generalized χ^2 measure and is defined as the sum of the dot product of all histogram bins between two frames.

The shot detection algorithm is based on the minima detection of a matching curve which contains the metric values of successor frames over the time. The curve-matching algorithm approximates the calculated image distance curve by a B-Spline curve. Normal polynomial curves are not used because edge points of an approximated interval should not be approximated by the polynomial curve. A polynomial curve introduces false minima near these points which does not happen with B-Spline curves. The real shot detection is based on the minima of the B-Spline curve. These minima are potential candidates for a detected new shot. The similarity curve should fall below a dynamic generated threshold T_h which is based on the original data.

1.3.2 Algorithm of Zhu et. al.

Xingquan Zhu, Jianping Fan, Ahmed K. Elmagarmid and Xindong Wu described in their article, "Hierarchical video content description and summarization using unified semantic and visual similarity", algorithms for video summaries [53]. The base of their algorithms is a shot detection algorithm which is presented in [24]. The features are DC coefficient-based histograms from the intra-frame of a MPEG-video stream. With the Gauss distance between adjacent intra-frames and a dynamic generated threshold, shots are detected. After a new detected shot, the previous P- and B-frames are searched to refine the cut. Within a shot, there can be detected with a statistical algorithm whether a gradual transition exists between shots. This will then result in a new detected shot boundary. For simplification reasons, there are key frames for shots represented by the 10th frame of a shot. The paper deals with the problem of grouping shots together to get a higher layer of representing video context. Spatial shot clustering algorithm groups visualize same shots together but context information is lost. Video scenes are semantic units, so it is very difficult to define boundaries for such video parts. Video group detection algorithms are based on threshold selection.

They merge temporally and spatially related shots into groups. Four different kinds of abstraction levels are implemented to let user select different parts

of the video content based on a single frame. Each part is refined by the next lower abstraction level thus refining the content. The creation of the different abstraction levels is based on a merging process, which joins shots together to groups of shots. Different comparison algorithms are used between single frames, between a frame and a group of frames and between different frame groups. An entropy-based threshold technique is used to define a dynamic (adaptive) threshold in order to detect usable thresholds.

1.4 Subject of this work

Discrete Curve Evolution

Longin Jan Latecki and Rolf Lakämper developed a greedy algorithm (the *Discrete Curve Evolution*) to simplify a two-dimensional polygon line of a contour by removing successive vertices from the polygon [38]. The simplification is done in such a way that not too much important contour information of the original line is lost. Daniel deMenthon et. al. used this algorithm to segment videos [19, 20, 37]. This work follows the same idea and the same concept to segment videos and create indices.

The content

The content describes a video as a sequences of polygon trajectories with frame descriptors. Frames which are nearly equal to their neighborhood are found and gradually eliminated from the polygon. The algorithm does not reduce the key frame detection to a shot detection algorithm nor does it reduce it to a gradual change detection algorithm as it is normally done. It allows applications to implement their own definition of importance by implementing the relevance measure for frames depending on the aim of the key frame detection. The algorithm is tested with different classes of videos. We tested the implementation of the centroid based frame descriptors and their temporal order, and we improved and refined the selection of the features. The improvements include the implementation of a filter which is applied to the descriptors of a frame and its neighbor frames. The refinement includes the number of frame descriptors as well as the selection of color space and the class of features. New in this area is also the usage of dominant colors in the discrete curve evolution.

Chapter 2 contains fundamental information about the used algorithms such as the discrete curve evolution, feature filtering, quality measurement and video terminology. In Chapter 3, the term *video key frames* and the requirements for a detection algorithm are worked out. The necessary requirements and implementation of the *Discrete Curve Evolution* process are also implied. Chapter 4 contains improvements for the existing frame descriptors as introduced by Daniel DeMenthon. These include the selection, the weigh[-]ting and filtering of the frame descriptors. New experiments with different color spaces and different features, based on dominant colors and an optimal color composition, are done in Chapter 5. Chapter 6 contains an experimental and algorithm comparison with others who have done research on key frames extraction. New is the application of the *discrete curve evolution* on video streams without a well-defined start and end frame by defining an analysis window. This is useful for real time key frame detection and is introduced in Chapter 7. Chapter 8 contains a summary of the presented information and experiments and a conclusion of the work in this paper. The appendices contain additional information about the used videos and applications. Appendix A contains an abstract of the videos used in this paper, with background facts like duration, number of scenes and the expected ground truth result. Appendix contains a description of the applications I have used and written to perform the experiments. Appendix B includes the free-available MPEG1-player of the Berkeley University [8] which was used to create the frame descriptors and to extract the calculated key frames. We have also developed a helpful tool which shows the key frames at a specific abstraction level and controls the video with a remote control. It is easy to to navigate inside the video and select different abstraction levels with this navigation tool.

This work is partially based on previous published work [35, 36].

Chapter 2

Fundamental

The following sections of this chapter will describe the fundamental definition of the video processing terminology which is used in this paper. Also described is the mathematics behind the algorithms, the filters and the quality measurements.

2.1 Discrete Curve Evolution

The advantages of the *Discrete Curve Evolution* (DCE) is the flexibility, the speed performance and robustness. The basic work is based on an iterative polygon simplification algorithm which is called the *Discrete Curve Evolution*. In this iterative process, vertices of the polygon are removed which are mostly constant in comparison to the neighbor vertices. The algorithm belongs to the class of greedy algorithms which implies that it is a local optimization algorithm.

The first appliance of the discrete curve evolution was in the context of shape similarity of planar contour objects [39]. Figure 2.1 shows a few stages of the evolution applied on an edge of a fish drawing. Notice that the most relevant vertices of the curve and the general shape of the picture are preserved even though most of the vertices have been removed. In the geometric language for the polyline trajectory, these vertices are the most linear ones. Consequently, the remaining vertices of the simplified polygon line are frames that are more different than the deleted ones.

In the following subsections, the algorithm and the elimination process of the discrete curve evolution will be discussed. The applicability of the dis-

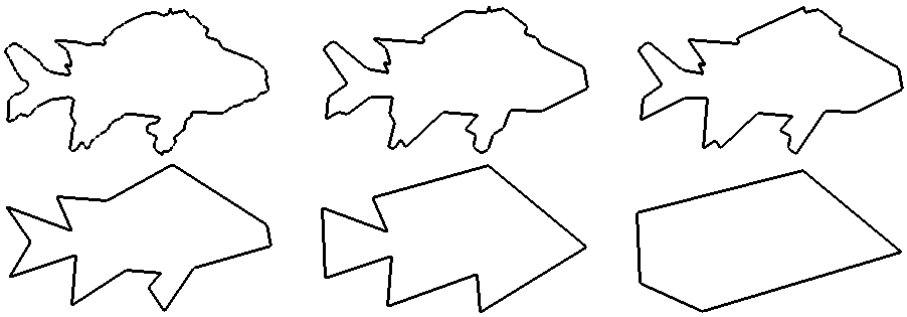

Figure 2.1: Six stages of the discrete curve evolution for a planar object

crete curve evolution as a video segmentation algorithm will be discussed in Chapter 3.

2.1.1 Evolution Process

Lakämper uses the term *nearest abstraction* to define the *discrete curve evolution* which uses a measure named *cost function* to select the vertex of the polygon that should be removed [32]. The original definitions are used for two-dimensional contour in $\mathbb{Z}^2$. Our implementation of the discrete curve evolution is used for video segmentation for which we need a higher feature space $\mathbb{R}^m$ and changes in the definition of the cost function. We introduced a *local context* to define a subset of $\mathbb{R}^m$ in which the cost function is applied to a vertex of the polygon.

Definition: Local Context
Let P be an endless polygon $P = (v_0, ..., v_n) \subset \mathbb{R}^m; v_i \in P$.
The *Local Context* Loc_c of v_i in a given neighborhood $c \in \mathbb{N}$ is defined by
$Loc_c : \mathbb{R}^m \rightarrow P_{loc} \subset P$

$$Loc_c(v_i) = \{v_j \mid v_j \in P \wedge \|i - j\| < c\} \tag{2.1}$$

In the following sections, $Loc_c(v_i)$ is substituted by P_{loc} for the sake of simplicity.

Definition: Nearest Abstraction
Let P be an endless polygon $P = (v_0, ..., v_n) \subset \mathbb{R}^m; \mathcal{I} : \mathcal{I}(\mathbb{R}^m) \rightarrow \mathbb{R}$ is a measure for the information content of the polygon P. The nearest abstraction of P is $P' = \mathcal{A}(P)$, which is a polygon with

- $P' \subset P$
- $\mathcal{I}(P') \leq \mathcal{I}(P)$
- $\forall P'' \subset P : \mathcal{I}(P'') \leq \mathcal{I}(P')$

The information content measure $\mathcal{I}$ is defined as sum of the relevances of all vertices of the polygon. The relevances are measured by the *cost function*.

$$\mathcal{I}(P) = \sum_{i=0}^{|P|-1} \mathcal{C}(v_i, P_{loc}) \tag{2.2}$$

The cost function $\mathcal{C}$ describes how much content information will be lost if the vertex v is removed from P. The meaning of $\mathcal{C}$ will be described later. It is defined by

$$\mathcal{C} : \mathbb{R} \times P_{loc} \rightarrow \mathbb{R}; P_{loc} \subset P \subset \mathbb{R}^m \tag{2.3}$$

The definition of $\mathcal{I}$ and the requirements of $\mathcal{A}$ are as follows:

$|P| - |P'| = 1; P \backslash P' = \min_{v_i \in P} \mathcal{C}(v_i, P_{loc})$ where $|\ .\ |$ is the cardinality function and P_{loc} is a local context of v_i.

Definition: Discrete Curve Evolution
Let P be an endless (not necessary closed) polygon $P = (v_0, ..., v_n) \subset \mathbb{R}^m$ in such a way that $\exists v \in Vertices(P) \exists \lambda \in \mathbb{R} : \mathcal{C}(v, P_{loc}) = \lambda$.
The *Discrete Curve Evolution* (DCE) is a sequence of polylines $\wp = (P = P^0, ..., P^m), \#\{v | v \in Vertices(P^m) \exists \lambda \in \mathbb{R} : \mathcal{C}(v, P_{loc}) = \lambda\} = 1$ (where $|\ .\ |$ is the cardinality function) and with $P^{i+1} \subset P^i$ where $P^{i+1} = \mathcal{A}(P^i)$. $\mathcal{A}$ is the nearest abstraction.

The algorithm behind the discrete curve evolution could be defined as follows:

Definition: Discrete Curve Evolution Algorithm
k=0;
while $\exists v \in Vertices(P^k) \exists c \in \mathbb{R} : \mathcal{C}(v, P_{loc}) = c$
Find $v_i : \mathcal{C}(v_i, P_{loc}) = \min_j \{\mathcal{C}(v_j, P_{loc}), v_j \in P^k\}$
$P^{k+1} := P^k \mid_{v_i}$
increase k by one
repeat

2.1.2 Cost Function

The *cost function* is a function that measures how much information a vertex inside the polygon contains and is lost if the vertex is removed from it. The cost function is performed for every step of the curve evolution process for the whole new polygon P^k. After each step, the values of the cost function change for every vertex where its neighborhood also containes the removed vertex.

The *cost function* used by Lakämper et. al. [39] measures the information content of a vertex in relation to both neighbor vertices. It is possible that the information content of a vertex depends on a larger neighborhood and not only on the direct neighbor vertices. This limitation is avoided by defining a larger measure context for the cost function $\mathcal{C}$ which we called the *local context*.

For our case, we defined the cost function on a polygon vertex in relation to a *local context*, which contains a subset of the neighbor vertices surrounding the measured vertex.

2.1.3 Polygon

The *polygon* used by Lakämper et. al. [39] was defined on $\mathbb{Z}^2$ as a closed polygon.

One of our requirements of Chapter 3.1.1 will be a symbolic description of the video frames. The polygon vertices could be represented by the symbolic description of the frames descriptors. These vertices are linked to each other in the order of their appearance in the video. We used an open polygon with static start and end vertices. The frame descriptors will not be necessary Euclidean.

2.2 Video processing terminology

In this paper, video technology terms are used. This section contains definitions of the terms which are used in this paper.

- An *image* is a function of $f \subset \mathbb{Z}^2$ into a color space $\mathbb{R}^n$. Normally a subset of $\mathbb{R}^n$, like $[0, 255] \subset \mathbb{Z}$ for grey color images or $[0, 255]^3 \subset \mathbb{Z}^3$

for RGB or YUV colors, is used. Through color space transformations, other subsets of $\mathbb{R}^n$ could also be used for the domain space. $f(x, y) \rightarrow \overline{\text{color}(x, y)}$

- A *video* is a temporal ordered sequence of images. $Video = \{f_1, ..., f_n\}$

- A *frame* is an image of a video at a specific time step.
  $f_t : \mathbb{Z}^2 \Rightarrow \mathbb{R}^3$
  $f_t(x, y) \rightarrow \overline{\text{color}_t(x, y)}; t \in \mathbb{R}$
  Time t is application dependent and could be a real existing time with a time unit like seconds or an abstract time unit like frame number (in which case t would be a natural number).

- A *video stream* is a video with an undefined or infinite number of frames. $Video = \{f_1, ..., f_\infty\}$

- A *closed video* is a video with a defined number of frames. $Video = \{f_1, ..., f_n\}, n \in \mathbb{N}$

- The *frame rate* is the rate of played frames in a video per second (measured frames per second (unit [fps]). This will vary for different kinds of video recording media and standards (like cinema, DVD, television, PAL, NTSC, digital photo camera or video camera). Normally frame rates vary between 25 and 30 fps. Digital photo cameras sometimes use a lower rate and webcams sometimes use less than 1 fps.

- The *frame or video resolution* is the resolution of a single frame in pixel's width and pixel's height. The dimensions could vary depending on the used video standard.

- The *resolution ratio* is the ratio of the resolution width:height. This is normally for a television 3:4 (0.75). Cinematic films have normally ratios higher than 1.5 (like 1.85 or 2.35). The films we have used have often the same ratio as used for televisions. The most common resolutions[1] are 320×240 or 160×120.

- A *pan* is either a horizontal move of the camera or a rotation around the vertical axis. For example, the scenery is recorded from left to right or from right to the left.

- A *tilt* is a rotation of the camera around the axis in the viewing direction.

[1] Expected is, that the size of the pixel's are squares

- A *shot* is the smallest video unit of continuous frames with the same content. Shots are separated by a *cut* or a *transition*.
- A *cut* is a hard switch between different shots. Ideally a cut happens between exactly two frames. The first frame before the cut and the first frame after the cut are normally completely different.
- A *transition* is a soft switch between different shots. A transition is a change between two shots with at least one frame showing parts of the content of both shots. Different kinds of transitions exist. A *gradient transition* is a linear change switch between two shots resulting in a blending between the shots.

2.2.1 Key Frames

The term *key frame* is in the video signal processing literature not well defined. It is an abstract description of representative frames of a video sequence. Sometimes a *key frame* is defined as a representative frame of a single shot [24].

The reduction of *key frames* to a single representative frame of a *shot* will reduce the meaning and the abstraction of those shots. There is no information about the quality of this abstraction. Different shots could have the same content; however, a single shot could have a different kind of content. There exist many various kinds of examples which show that it is not possible to reduce a single shot to one frame. By the association of key frames to shots, the abstraction level of the video is automatically predefined, and there is no way to modify this level even if it is necessary to define different kinds of abstraction levels. I agree that there are many applications for which a reduction of shots to a single frame would be correct, but there will exist even more applications for which this would be wrong.

What are key frames representing?
Key frames will be a representation of the video content by single frames. This implies a reduction of frames and information, and a representation of the original source by the remaining frames from the view of an observer. The expectations of this representation are simple. The approximation must be good enough and should describe the content of the video. This could be interpreted as follows:

1. The information reduction should be between an expected minimum and maximum abstraction level. If the number of frames is below the

minimum level, then the information reduction is too high, and an abstraction level which is too low will result in reducible information. These upper and lower abstraction levels could vary depending on the observer and his expectation and interpretation of the abstraction.

2. The remaining frames must match the *content* of the original frames as well as possible. This could also vary depending on the observer and his interpretation of "matching the content".

Which are the key frames?
If someone asks this question, then the answer will vary such as "The content must be described by the key frames" or "The key frames should be a summary of the video". If someone asks whether a set of key frames will match their expectation or not, the answer will also vary from "Yes, but that frame is missing" or "No, that information is not important for me".

For the same content, different people will expect different key frame sets as a result of how they have defined for themselves the similarity of the key frames and the associated content. It is possible that depending on the mood, the time, the situation and the reason for the key frame creation, the same person will describe the same video by different *key frames*.

Consequence:
The consequence is that we have a reduction and abstraction level of the video (which is reflected by the key frames) that depends on the view of an observer and his expectation which will also depend on the application for which the reduction is needed.

A given set of key frames from an observer will identify the application and the role in which the observer acts. The quality and range of acceptable key frames will be more or less vague in this situation. Only the observer himself can define the expected abstraction level wether results are suitable for his application or not. The quality of key frames rises and falls with the correct numerical definition of key frames and abstraction level as used by the observer.

This consideration results in the fact that there will be no suitable definition of key frames and abstraction level for each application (in which an observer acts). Key frames will only be clearly available if this unknown information of the application and user is given.

We need answers to the following two questions before we can give the key frames for a given video.

1. What kind of information of the video is the observer interested in?

2. How far or to which abstraction level should this information be reduced?

These kinds of problems also exist for example in other parts of the video signal processing like Content Based Image Retrieval. The search for specific images is based on the expected information of the images. Projects like Viper [28] use learning algorithms to detect the requirements for frames as required by user. Frames are compared on base of image descriptors. The learning process increases or decreases the weighting of the descriptors by user-interaction in order to find the best match for the expected results. This results in a description in which the image descriptors user is interested. These kinds of algorithms should lead to good results if the image descriptors are suitable for the class of information that is expected by the users application. A similarity value gives information about the quality of an image and the trained images. This makes it possible to differentiate between a good and a bad content similarity.

These two open questions make great demands on the key frame detection algorithm. It should be flexible to fulfill the definition of important information and the possibility of defining an abstraction level.

Due to the fact that we could not define a suitable general definition of key frames, we tried to define a specific but general description of key frames, in order to reach a wide class of applications in which this description could be used. It is not our intent to find a suitable algorithm to fulfill all expected results for any situation and application.

It is more intuitive that key frames should contain important information about a local part of a video sequence that is representative for it and not only for shots. Different sequences should also contain different key frames. Such a local sequence (or part of a video) should contain nearly the same frames with the same content of information (entropy) or significance. This definition of *key frames* is very depending on the context in which this frame appears. Changes in this local context will reflect in changes of the key frames. These changes are possible due to one of the former reasons which could be interpreted as a request to the definition of key frames.

These different criteria for key frames come from the different applications in which key frames are used or should be detected. For simple video summaries, a representation of shot key frames could be enough [31]. More complex

summaries use grouping algorithms or other algorithms, which need more information about the context [53].

The definition behind the term *key frame* will be discussed in the next chapter.

2.3 Quality Measurement

A single key frame algorithm with results could be useful but there is no information whether these results are *good*, *representative*, *complete* or even *suitable*. Several [5] measurements exist that try to describe the quality of the results which are returned from the key frame detection algorithm.

To test the quality of the experimental results, there must be *suitable experiments* with *predefined representative results*. These *ground truth results* should exactly match the key frames as expected by the key frame definition. This includes either an ordered list of key frames in order of their importance or a fix set of key frames with the number of frames that is expected. This is expected in accordance to the answer for question 2 in section 2.2.1. We need either an abstraction level (which directly or indirectly includes the number of key frames) or the list of all key frames, in order of their importance, in order to select an abstraction level at another time.

The experimental results are compared to the ground truth results. The statistical information from those comparisons is a quality measure of the experimental result.

1. *recall*
   The recall describes the completeness of the responding functionality/algorithm.
   $Recall := \frac{|\{expected\ response\} \cap \{response\}|}{|\{expected\ response\}|}$
   The recall is a measure of how many correct answers we have. A zero means that we have no correct answer and a "one" means that all of our answers are correct.

2. *precision*
   The precision describes the accuracy of the responding functionality/ algorithm. $Precision := \frac{|\{expected\ response\} \cap \{response\}|}{|\{response\}|}$
   The precision is a measure of how many of our answers are wrong. A zero means that all of our answers are wrong and a "one" means that all of our answers are correct.

3. *precision versus recall graph*
   The precision-recall graph shows the relation between the number of correct and false detected key frames.

The *expected response* is the expected result of an algorithm or function. The *response* is the effective response of an algorithm or function.

Besides these objective quality measures, we also have subjective quality measures. We compared our algorithm directly with the results of other key frames detection algorithms and analysed the resulting key frames. Due to the nature of those key frame definitions and algorithm implementations, it is difficult to make a direct conclusion of these results.

The ground truth results are self-recorded videos, based on simple scenarios with predefined changes in the environment. The advantage of a real camera recording is the non-artificial move of the camera (pan) and the objects (these are noises in the object motion during the time). Also we have a background noise in the frame colors themselves.

The comparison experiments are created by other people and it is not always obvious which key frames are expected. Sometimes a trailer or a mixture of cuts is used for comparison results. A small set of the videos contains one set of ground truth results because they are selective defined by a larger group of people [21]. A description of the videos can be found in Appendix A.2 and are available at the homepage [12].

Chapter 3

Key Frame Detection

We have seen in the previous chapter that the definition and therefore the detection of key frames is not trivial. In this chapter we will discuss the requirements for a reasonable key frames detection algorithm and how this could be implemented in the Discrete Curve Evolution.

3.1 Key frames requirements

Our definition of key frames should fulfill several criteria, which range from the classical shot representation further to the detection of different but important information inside a shot. Other criteria are also non-linear changes such as motion changes of objects, camera pan or other unpredictable events because this kind of information could also be important for an observer, and thus should be detected.

In summary, our requirements of a key frame detection algorithm are as follows:

1. **Shot detection**
   A change in the local information content is possible by a cut or a blending like gradient transition between shots. Such a shot and cut detection is a well-studied topic [3, 24]. There is also no problem in finding key frames for different shots with and without blendings.

2. **Change in content**
   It is possible for one shot to contain different kinds of information

because of background or foreground object changes. This is possible through objects which enter the area of view or through objects which hide or do not hide another object. For example, a 360-degree panorama view of a beach will first show a beach and then a few seconds later, the beach or something else in the background such as a city or mountains. The content of this panorama view cannot be representable by a single key frame.

3. **Event detection**
   A change in the "action" of a scene could also reflect a change in the local context. A moving car contains other information as a parking car. The *events* "start" or "stop" could be important for our (imagine) application and should also be represented by key frames. This also implies that linear changes should not be detected.

Points 1 and 2 reflect changes between frames that should be detected. In terms of shot detection, a key frame should represent the detected shot and the changes of the shot to a neighbor shot. However, due to the concept of the algorithm and the fact that a shot has two boundary shots, these changes are relative and not always practicable. (It makes, for example, no sense to represent two very similar shots by two very similar key frames.) Point 3 reflects a non-linear change in the *local context*.

3.1.1 Requirements of a frame comparison algorithm

The requirements above have influence on the comparison between the frames in a video. A direct frame-to-frame comparison (such as is done in [24, 31]) is not possible for this will not fulfill the requirement 3 because more information of the context is needed in which the frame appears. Either we need frame descriptors, which reflect this local content, or we need a frame similarity measure which considers this. It makes sense to define frame descriptors for the frame itself without considering the information about the neighbor frames. This is done to make the algorithm as general as possible for existing frame descriptors, which exist for single frames only.

1. **Symbolic Description**
   We need a symbolic description of frames to represent these frames. These *frame descriptors* (short *FD*) should be able to represent the different information content of frames. The selection of these *FD* has

directly influence on our application for which we need the key frames. The better the FD matches our expectation, the better the results will be.

2. **Frame Comparison**
   We need the possibility to compare frames based on the symbolic description in relation to each other. The better we are be able to compare frames, the better our results will be.

3. **Context Dependency**
   The comparison between single frames does not consider the *context*, so we do not need a frame-to-frame similarity but a similarity of a frame to its *local context*. This comparison will be defined later and it will be based on a frame-to-frame comparison. Therefore, the first two requirements are the most important because a comparison of frames is based on these frame descriptors, and the quality of the key frame algorithm will depend on the frame comparison.

3.2 Image and Video Descriptors

The classification of images and video sequences is based on descriptors that are extracted from the video frames. All detection algorithms for video streams only are based on the video content. Other ideas are to develop algorithms using other kinds of information like audio to extend the available information in order to get better results in the video processing. This will not be a part of this work. A video processing algorithm based only on the video content should be able to fulfill our requirements for a key frame detection algorithm because we only used the video content in our considerations.

However, this shows us how important the selection of frame descriptors is. The quality of these frame descriptors has a direct influence on the quality of the algorithm which is based on these descriptors. It is important to know, based on the defined expectations, which kind of information should be stored in the descriptors. For example,

1. If the position of objects has an influence on the ability to detect this frame as a key frame, then it is logical that some of the position information must be included in our frame descriptors. Such information could be represented by different kinds of data. This could be directly

accessed by the coordinates of the objects but also indirectly accessed by weighting (non-position) depending descriptors based on the area in which they appear; therefore, these frame descriptors have got position information. For example will a histogram based on colors which are weighted by a factor depending on the pixels appearance, will contain other information as a straight-forward built histogram.

2. Speed changes are reflected by an acceleration or deceleration which in turn reflect a non-linear position change of an object over time. Time and position depending information must be included in the frame descriptors in order to be able to detect such kinds of events. The time factor could be directly represented by the time of the frame in the video or by its frame number. The time factor could also be indirectly stored in terms like speed factor or motion vectors such as used in MPEG video stream B- and P-frames.

As we can see, the available amount of information could be represented by a wide range of frame and video descriptors. It is assumed that *frame descriptors* are descriptors which are available within a frame but which did not have information about the context of the frame in the video. *Video descriptors* are descriptors which contain information about the frame inside the video. This could be the mentioned time information but it also could be information about the other frames in the video. An idea is to detect whether the importance of frame descriptors should be increased, and whether if they are non-constant in the context. With this amount of information and also the amount of possibilities of getting the information, the storing and the processing of the information will be increased. This will cause a confused result and thus will lead to a reduction of the information into an easily comprehensive number of descriptors without an important loss of necessary information. There exist many more kinds of information inside frames, such as the following:

- Texture information which could be represented by the relationship between the lightest and darkest pixels inside different-sized areas around a single point [53]. Several kinds of texture information descriptors are described in [23].
- Different color spaces such as $\{R, G, B\}$, $\{Y, U, V\}$, $\{L, a*, b*\}$
- Color moments like the centroid coordinates.
- Motion vectors of frame components as used in MPEG video streams.

- Hidden Markov Model
- Training based feature weighting

Our expectation of the *events* requirement assumes that we can detect at least moving objects and changes in their speed. This assumes the availability of object position and time.

3.2.1 Buckets

Features based on histograms are widely used in the literature [18, 52]. Daniel DeMenthon et. al. developed histogram-based features called *buckets* [18] which are not directly based on the histogram but based on centroids, that are based on the histogram. The histogram of each Y-, U- and V-color component, with a range from 0 to 255 is subdivided into four bins with a linear range of 64 colors. The pixels with the color from a single bin represent a centroid.

The coordinates x, y and *area* of the centroid are used as the features for a bin. This feature selection is done for every color component and every histogram bin. A single bucket contains the features of a single centroid. The only necessary missing feature is the time which is added as an additional feature component and is represented by the frame number. Together with the time factor, we will have
$3[colors] \times 4[\frac{intervals}{color}] \times 3[\frac{features}{interval}] + 1[(time)feature] = 37[features]$. These features are represented by a 37 dimensional vector which reflects the information content in our frame: $f_t \rightarrow \mathbb{R}^{37}$

3.2.2 Dominant Colors

Another and a more intuitive feature is *dominant colors*. Hu et. al. has described a model and comparison algorithm for *Content Based Image Retrieval* [29]. The advantage is that the descriptors and the comparison algorithm are more based on the human vision model, so the results for a frame-to-frame comparison will match our expected results easier if we compare two frames. (The easiest way to understand this is with a cut detection algorithm which is based only on a frame-to-frame comparison algorithm. If the human vision detects a large difference then the algorithm should also detect a large difference.)

This algorithm detects the most important colors from a code book by associating the smaller-sized color areas to the larger-sized color areas. Lesser important colors are displaced by more important colors. The descriptors contain the code book color number and the percentage of its area. The matching algorithm searches the optimal color matching such that the difference is minimal. The disadvantage is that neither area information nor time information is stored in frame descriptors.

3.3 Relevance Measure

As we have seen, the *cost function* describes how much information of the content of the neighbor vertices is contained in a specific vertex. For example, a car drives from Point A over Point B to Point C, and Point B lies exactly between A and C. Then it could be expected that if the movement of the car is linear, without loss of information, then the car will drive from A to C and it will pass point B. The information that the car will pass Point B is redundant. The cost function at Point B should be zero because no additional information is contained in it.

The consequence is that more important frames will contain a higher value for the cost function. In terms of key frame importance, the cost function measures how important a frame is in relation to the neighbor frames.

The considerations of the previous chapters lead to the following consequences. A video is mapped into $\mathbb{R}^m$ through presentation of the frames by their frame descriptors. The frame descriptors are linked to each other resulting in a polygon. Neighbor frames represent neighbor vertices.

Definition: Relevance Measure
Let $P = (f_0, ..., f_n)$ by a polygon. A cost function $M_{rel}(f_i, Loc_c(f_i))$ is a *relevance measure* for a given *key frame definition* if it satifies the following:

1. If f_i is more *similar* to $Loc_c(f_i)$, then f_j is similar to $Loc_c(f_j)$, then $M_{rel}(f_i, Loc_c(f_i)) < M_{rel}(f_j, Loc_c(f_j))$
2. The "similarity" term above should match our expectation of the *key frame definition*.

M_{rel} is not necessarily positive definite. So there could exist some f_j, such that $M_{rel}(f_i, Loc_c(f_i)) < 0$.

The requirement to detect shots implies that abrupt changes to at least one neighbor frame should be detected, and that slow changes or nearly equal frames should not be detected. For example, if frame f_t is nearly equal to f_{t+1}, and f_{t+1} is very different from f_{t+2}, then

$$M_{rel}(f_{t+1}, \{f_t, f_{t+1}\}) < M_{rel}(f_{t+1}, \{f_{t+1}, f_{t+2}\}) \tag{3.1}$$

The requirement to detect non-linear events is described in the following example. It is assumed that if an object moves linear from frame f_t over frame f_{t+1} to frame f_{t+2} then in frame f_{t+2} appears an event: The object travels with another speed, then it moves slower over frame f_{t+3} to frame f_{t+4}. It is expected that there is a linear movement in f_{t+1} and therefore its relevance $M_{rel}(f_{t+1}, \{f_t, f_{t+1}, f_{t+2}\})$ should be low. In f_{t+1} there is a change in the speed, and therefore it's expected that the relevance $M_{rel}(f_{t+2}, \{f_{t+1}, f_{t+2}, f_{t+3}\})$ should be higher.

$$M_{rel}(f_{t+1}, \{f_t, f_{t+1}, f_{t+2}\}) < M_{rel}(f_{t+2}, \{f_{t+1}, f_{t+2}, f_{t+3}\}) \tag{3.2}$$

In f_{t+3}, the movement is again linear and therefore $M_{rel}(f_{t+3}, \{f_{t+2}, f_{t+3}, f_{t+4}\})$ should be low again.

$$M_{rel}(f_{t+2}, \{f_{t+1}, f_{t+2}, f_{t+3}\}) > M_{rel}(f_{t+3}, \{f_{t+2}, f_{t+3}, f_{t+4}\}) \tag{3.3}$$

The frame f_{t+2} in which the event occurs, is expected to be more relevant than the other frames.

3.3.1 Image comparison

Point 2 in our relevance measure definition could be the most important requirement because it defines how close our results would be to the expected key frame definition; however it also could be the most difficult requirement to be satisfied.

Normally [31, 53] a distance measure $d(\ .\ ,\ .\ )$ would be the easiest way to compare images with each other in order to get a similarity measure between two images.

1. Identity: $d(x, x) = 0$

2. Positive definition: $d(x, y) > 0, \forall x \neq y$

3. Symmetry: $d(x, y) = d(y, x)$

4. not the triangle inequality $d(x, y) + d(y, z) \geq d(x, z)$ i.e., there can exist some y's such that $d(x, z) > d(x, y) + d(y, z)$

Number 1 means that two identical frames should have no differences and should also have a distance from 0. Number 2 means that the distance between two different frames is positive definite. Number 3 means that it makes no difference in which order we compare two images.

If the *local context* is only one neighbor frame, then we could use this metric for a relevance measure. Unfortunately this will not fulfill our requirements in detecting non-linear changes; therefore, we need another measure. A previous and a successor frame are at least necessary to detect non-linear changes. So M_{rel} should be defined on at least three frames. The easiest way to define such a measure is to use the metric in order to define such a relevance measure.

Some examples for "potential" relevance measures based on frame-to-frame comparison metric are as follows:

1. $M_{rel}(f_1, \{f_0, f_1, f_2\}) := d(f_0, f_1) + d(f_1, f_2)$

2. $M_{rel}(f_1, \{f_0, f_1, f_2\}) := |d(f_0, f_1) - d(f_1, f_2)|$

3. $M_{rel}(f_1, \{f_0, f_1, f_2\}) := d(f_0, f_1) + d(f_1, f_2) - d(f_0, f_2)$

3.4 Conclusion

The frame descriptor selection based on histogram centroids is very suitable for our expected key frames. It contains all the necessary information that is used in our key frame definition. The *discrete curve evolution* is useful because the relevance measurement rates those frames with a higher value which match our key frame definition.

Only information descriptions by frame descriptors and information analyses by the relevance measure are not guarantees for a good key frame extraction. Every kind of information which is included in all videos and video frames[1]

[1]If a frame contains $x \times y$ pixels in a 3 dimensional color space, then we will have information in $\mathbb{R}^{3xy+1}$, this means for a frame resolution of 320 $\times$ 240 a dimension of 230401

is compressed into $\mathbb{R}^{37}$. Everyone will see that such a compression also leads to a reduction of information. An important question however is how stable the features are. Will they be easily disturbed by small changes in the frame like a little bit more or less brightness? What happens if the frames are scaled? Whether the different kinds of information which occur in the "real" is distinguished well enough in this 37 dimensional space, and whether the contained information can also be interpreted by the detection algorithm will be tested and verified by experiments in the next chapters.

Chapter 4

Closed Videos

In this chapter, we will analyse the applicability of the Discrete Curve Evolution with closed videos. As we have seen in the previous chapter, Daniel DeMenthon has used a 37 dimensional feature vector based on video frames to describe the content of a closed video. The key frame detection is split into different steps which are performed by individual applications. The first step is the creation of the frame descriptors, which implies the extraction of frames, the associated histogram bins and time information. The second step is the *Discrete Curve Evolution* as described in chapter 2.1. The third step is the extraction of the key frames from the video.

In this chapter we will see how these frame descriptors can be optimized. We will show the flexible extensibility of our algorithm by changing the frame descriptors.

4.1 Abstract review of existing applications

The applications used by Daniel DeMenthon were written and developed by the LAMP division at the Maryland University [34]. The application *Merit* was used to analyze the DCT (*Discrete Cosinus Transformation*) in MPEG video streams. This application is used for the *DCE key frame* algorithm to extract the color information of each frame. An MPEG-1 video stream stores data in DCT blocks of 8x8 values (which are the *macro blocks*). The most upper left values of such a DC block contain the average color intensity of the block that is available for each the YUV color components. This (average) color information is used to calculate the centroids which gives us

the necessary information. This data is stored in a DCT-File[1] used by the next application.

The *bucket* application, together with the frame number and the DCT-data creates the frame descriptor vector. Its data is stored in a BFT-file. This file is used by the discrete curve evolution to perform the key frame detection.

The histogram is subdivided into four equidistant parts and for each part there is a *bucket* with centroid data calculated. (From these buckets is calculated the frame descriptor for each frame f_t.) The vector elements are ordered by the colors Y, U, V (in this order). Inside each color component is stored the data from the lightest color intensity to the darkest color intensity. The data is (in this order) the "x" and "y" value of the centroid in DCT block coordinates and the area in number of DCT blocks.

$$
\begin{array}{lllllllll}
f_t \rightarrow FD(f_t) = (& t, & & & & & & & \\
 & b^x_{Y_1}, & b^y_{Y_1}, & b^\#_{Y_1}, & ..., & b^x_{Y_4}, & b^y_{Y_4}, & b^\#_{Y_4}, & \\
 & b^x_{U_1}, & b^y_{U_1}, & b^\#_{U_1}, & ..., & b^x_{U_4}, & b^y_{U_4}, & b^\#_{U_4}, & \\
 & b^x_{V_1}, & b^y_{V_1}, & b^\#_{V_1}, & ..., & b^x_{V_4}, & b^y_{V_4}, & b^\#_{V_4} &)^T
\end{array}
$$

The Discrete Curve Evolution is performed by the *curve evolution* application which is a C++ program. This application reads the BFT-file and produces three output files, although only the EVO-file and the TOL-file are mentioned here. The relevance measure is as follows:

$$
\begin{array}{rcll}
M_{rel}(f_t, \{f_{t-1}, f_t, f_{t+1}\}) & := & d(FD(f_{t-1}) & , FD(f_t)) \\
 & + & d(FD(f_t) & , FD(f_{t+1})) \\
 & - & d(FD(f_{t-1}) & , FD(f_{t+1}))
\end{array}
\tag{4.1}
$$

Where the metric is the Euclidean metric as defined by:

$$
d(FD^1, FD^2) := \|FD^1 - FD^2\|_2 \tag{4.2}
$$

The EVO-file contains frames in the order when they were removed from the polygon line. Additional information stored with the frame is both the relevance number and the relevance value of the frame as it was removed. The TOL-file contains a (hypothetic) number of relevant frames which are representative for the video. The algorithm behind this value is discussed in section 4.2.6.

[1] See appendix B.3 for a description of the file formats.

The extraction of the key frames is done with a *modified MPEG player* from the Berkeley University [8], written in C. This MPEG-player is freely available in several Linux distributions and could also be downloaded from the internet. The modifications include an option for reading a list with frame numbers from a file. These numbers are extracted and stored in the PPM-format which is a well-known file format for UNIX based computer operating systems. This frame list was created from the EVO- and the TOL-file.

The list of applications was finished with a JAVA-applet which joins all files to the video viewer. The viewer could be used as a standalone application or as client application in a browser like Firefox [44] or Internet Explorer. The application has a slider from which it's possible to easily define the needed abstraction level. The default abstraction level is the number of frames as defined in the TOL-file. The order of the frames in the different abstraction levels is defined by the EVO-file. The available frames in the abstraction level are shown in a time-line which represents the video. With the mouse, each key frame is selected and is then shown in the viewer.

4.2 Algorithm and software analysis

We tested the functions on our test videos [13, 14, 15]. These results were not bad but some improvement was necessary. Our improvements included changes in the data source on which the frame descriptors were based, the content of the frame descriptors and additional filtering functionality of the frame descriptors. We tested the algorithms and programs on other videos and scenes in order to test the applicability for different kinds of videos.

4.2.1 Feature Extraction

The existing features are based on the average color intensity of a macro block. It automatically implements a filter because the average color doesn't necessarily have anything to do with the existing pixels, and precise analysis of a frame is not possible due to the loss of information. For example, if the macro block contains 32 black and 32 white pixels, then the average color would be grey. These three colors exist in three different parts of the histogram. It is possible that our results will be disturbed. Also an eventuality analysis based on the texture is not possible because these fine contours do not exist in this part of the macro blocks.

It is possible that the dimensions of a frame not always be a multiple of 8, due to the nature of macro blocks which **always** have a size of 8 pixels. In that case, parts of the frame are either removed or either add or stretched and in both cases the processed information is not correct.

The third problem is that small position changes of objects inside a macro block are not detectable. In the worst case, those images could be identically identified if the average color value is not changed.

Also the time factor (frame number) is not scaling invariant. When we have another frame rate, this could lead to other results.

Feature normalization

The idea behind these features is good because it could contain much important information about objects in the scenario and even the whole scenario itself. It is possible to detect or identify objects if information is stored in other buckets than the background. Also it is easy and fast to extract these features from the video sequence.

The problem we have is that these features are not directly usable for different kind of frames and videos because the weighting and importance of the different features depends on the frame format. This will make it difficult to compare and analyze the results for our videos. For example, this will directly affect different scaled videos which can not give the same results for different sizes. The features are frame size and rate dependent.

It was necessary to scale the features into a well-defined domain before we could make tests. As we have seen in chapter 4.2.1, information could be lost due to the fact that features depend on MPEG1-macro blocks instead of on the pixels of the image. It would be an improvement if we would not use the macro blocks as a base for the features but instead the pixels themselves. This would reduce the amount of lost information. Other video sources and sizes which have no average color information stored in macro blocks could also be used. The disadvantage is that we have 64 times higher data content which must be processed because instead of one feature for each 8x8-sized macro block we will then have 64 features (one for each pixel). The time used by the bucket-filling process is linear to the amount of pixels, and the computer speed is increased rapidly in the past. Due to simplicity of the filling process, it is not expected that the processing time for this amount of data will increase significantly. If we have timing problems, then we will decode the MPEG data to frame data, but this is no problem for modern

computers and video cards which directly support MPEG decoding in real time.

Our idea is to scale closed intervals of the centroid data to a static-defined interval, independently of the source value interval in the buckets. We will scale our values to the interval $[0, 1]$. These intervals are selected because they could be represented by a linear normalization of the origin intervals. Different importance of frame descriptors could be realized by different weighting of these frame descriptors in the image comparison functionality; therefore, it is not necessary to implement a weighting of the different frame descriptors at this stage.

Our proposal is to scale centroid coordinate x from $x \in [0, x_{max}]$ to $x_{scaled} \in [0, 1]$. This is done by dividing x through x_{max}. The centroid coordinate y is scaled from $y \in [0, y_{max}]$ to $y_{scaled} \in [0, 1]$. This is done by dividing y through y_{max}. And the area is scaled from $Area \in [0, x_{max}y_{max}]$ to $Area_{scaled} \in [0, 1]$. This is done by dividing $Area$ through $x_{max}y_{max}$.

$$b^x_{centroid} \rightarrow b^x_{centroid}/b^{x_{max}}_{centroid}$$
$$b^y_{centroid} \rightarrow b^y_{centroid}/b^{y_{max}}_{centroid}$$
$$b^{\#}_{centroid} \rightarrow b^{\#}_{centroid}/b^{x_{max}}_{centroid}b^{y_{max}}_{centroid}$$

The problem is the time factor. Same video intervals in different video sequences should always have the same importance. One frame with the same pixel neighborhood should always give us the same relevance value, independent of how long the video sequence is and at which position these frames occur. If we make the decision to scale the time to a fix interval, we will have problems with cuts for the same frames of the video. If we have videos with different frame rates, we will also have problems if we do not scale the time to a clearly defined interval. We scaled the number of frames of one second to the interval $[0, 1]$.

Another problem with the time value is its weighting. What importance has the time factor in relation to the other relevance factors of the video content? The relative importance of the time factor in relation to total number of the other relevance values should also be constant. If we double the number of buckets to refine their content, then this bucket doubling should have no influence on the importance of the time. We did many experiments with a histogram subdivision of 4, 8 and 16 parts. Through the increase of the number of centroids, the importance of a single centroid is also decreased just as is the importance of the time factor. If the importance of the time is constant by increasing the number of features, then the time factor should be weighted by the same multiple of the number of increased features. So it

makes sense to multiply the time factor with a factor C which depends on the number of the frame descriptors and is defined as $C(x) = x/Constant$. For example, C will be equal to one if we have four buckets and a value of 13 for $Constant$[2].

$t_{framenumber} \rightarrow t_{framenumber} \cdot C(1 + 3 \cdot \#centroids)/framerate$

Our frame descriptors FD for the frame F_t, as used by the curve evolution, are defined by

$$
\begin{aligned}
f_t \rightarrow FD(f_t) = (\quad & \frac{t \cdot C(1+3 \cdot \#centroids)}{framerate} \quad , \\
& \frac{c_x^{Y_1}}{c_{x_{max}}^{Y_1}}, \quad \frac{c_y^{Y_1}}{c_{y_{max}}^{Y_1}}, \quad \frac{c_\#^{Y_1}}{c_{x_{max}}^{Y_1} c_{y_{max}}^{Y_1}}, \\
& \ldots, \\
& \frac{c_x^{Y_4}}{c_{x_{max}}^{Y_4}}, \quad \frac{c_y^{Y_4}}{c_{y_{max}}^{Y_4}}, \quad \frac{c_\#^{Y_4}}{c_{x_{max}}^{Y_4} c_{y_{max}}^{Y_4}}, \\
& \frac{c_x^{U_1}}{c_{x_{max}}^{U_1}}, \quad \frac{c_y^{U_1}}{c_{y_{max}}^{U_1}}, \quad \frac{c_\#^{U_1}}{c_{x_{max}}^{U_1} c_{y_{max}}^{U_1}}, \\
& \ldots, \\
& \ldots, \\
& \frac{c_x^{V_4}}{c_{x_{max}}^{V_4}}, \quad \frac{c_y^{V_4}}{c_{y_{max}}^{V_4}}, \quad \frac{c_\#^{V_4}}{c_{x_{max}}^{V_4} c_{y_{max}}^{V_4}} \quad)^T
\end{aligned}
\tag{4.3}
$$

The algorithms are more correct because the whole image is analyzed and not only the macro blocks. The application accesses the pixel colors directly from the output frame buffer of the MPEG-player.

4.2.2 Loss of information

It was not so clear and easy to detect the problems in the ground truth results, so we tried other improvements. As we mentioned before, the amount of information and the measurability of the changes inside this information are important. An increase in the amount of information could also increase the quality of the features and therefore the quality of the results. The amount of information which is used in the frame comparison algorithm has a direct influence on its quality. If in this process, too much information is lost, then this will result in a bad comparison result. This could result in frames that are

[2]Opposite to the derived frame rate, we has to use the frame number divided by 1000 because the correct frame rate in some of our videos was missing and a down-scaling of the frame number was necessary. In the case of 4 buckets and a frame rate of 25 fps the *Constant* will be 520.

bin	X	Y	Area
bin 1	20	173	13
bin 2	64	78	9
...	...	...	...
bin 16	0	0	0

Table 4.1: Table Y buckets of "Mov1" with 16 bins

bin	X	Y	Area
bin 1	20	155	8
bin 2	20	209	4
bin 3	58	120	3
bin 4	67	56	6
...	...	...	...
bin 32	0	0	0

Table 4.2: Table Y buckets of "Mov1" with 32 bins

too easily detected as equal but in reality they are unequal; therefore resulting in removed frames that are more different than the remaining frames. The idea is to add more histogram subsets in order to increase the amount of information but without getting too much redundant information.

The idea is to increase the subdivision of the color histograms on which the centroids are based. We will double the histogram subdivision. The amount of possible colors in each of the resulting histogram parts will be equal due to the fact that the number of colors is a power of two. The original number of buckets used by Daniel DeMenthon was 4 [18]. As we have seen, this leads to a total amount of 37 frame descriptors which includes a very large reduction of information. The idea is that a larger amount of buckets could lead to a better improvement of the results because the information reduction is reduced. We tried a histogram subdivision of 16 bins, which results in a feature vector with 145 frame descriptors (including the time feature) and a subdivision of 32 bins resulting in 289 frame descriptors.

Table 4.1 contains the frame descriptors for the Y buckets. The values are multiplied by 1000 and rounded to integer, so some rounding errors are possible. For a comparison table 4.2 contains the same frame descriptors for 32 buckets. In the second table, bin 1 and 2 represent the same content as in table one. Bin 1, and bin 3 and 4 of the second table represent the same content as bin 2 of the first table.

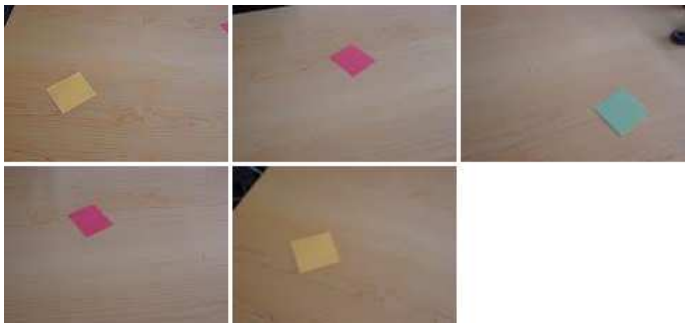

Figure 4.1: Expected five key frames of the ground truth video “Mov1”.

Figure 4.2: The best five key frames with 145 features for “MOV1”

One of our ground truth experiments is “Mov1”[3]. The video shows a table with three different colored papers. The camera starts at the first paper, moves to the second paper, to the third paper and back to the second and first paper. We expected those frames as the key frame result, in which a single paper is shown. This should be the five frames with the first paper, the second paper, the third paper, the second paper and the first paper. Figure 4.1 shows the expected key frames of video “Mov1”.

We tested the different bucket sizes with this video without filters to get comparable results. We saw in image 4.2 that the real results of the best five images did not match the best five images of the expected ground truth results[4]. Key frame number two was missing (it was leafed blank in the image set). In image 4.3, we saw that the missing image was inserted as the sixth best image. The frame descriptors use 8 bins (subdivisions) for each histogram color component to fill the buckets. For the results of figure 4.2, we used 16 bins for each color component. The results matched the expected

[3]Information about the used videos is available in appendix A.

[4]The small colored squares inside the images reflect the position and color of the buckets.

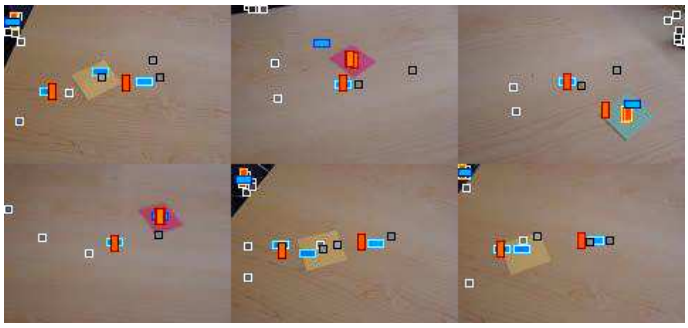

Figure 4.3: The best six key frames with 145 features for “MOV1”

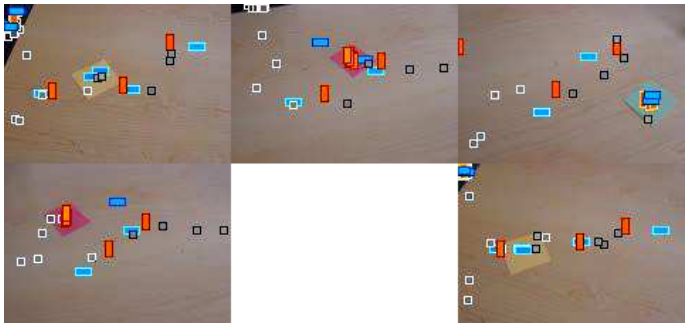

Figure 4.4: The best five key frames with 289 features for “MOV1”

ground truth results. The increase of the number of buckets will increase the quality of our key frame result.

Results are available for video sequences “House Tour”, “Kylie”, “Mov1” and “mrbeantu”. More information about these videos can be found in appendix A and on the homepage [12].

4.2.3 Nearly empty centroids

Another problem is video sequences where “nothing” important is happening, but key frames are detected. For example “Mov1”. We tested our applications in the default configurations on “Mov1”.

We have drawn the frame descriptors in the frames to get a visual idea of which frame descriptors exist and where they are located. The frame descriptor representation is done by drawing squares and rectangles for the represented centroids in the frames. The Luminance (Y) colors are represented by grey-colored squares. The inner part of the square shows the color intensity and a black or white border in order to get an acceptable contrast

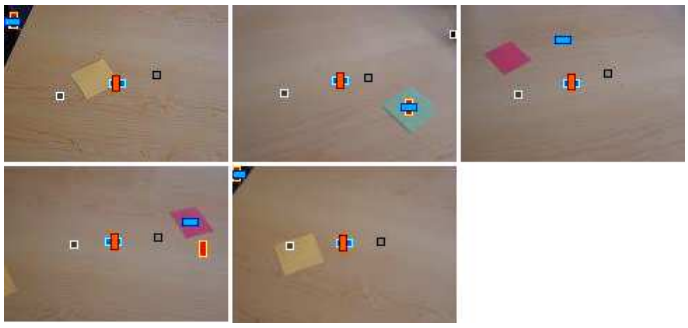

Figure 4.5: Resulting five key frames of video “Mov1” with normalized features. Frames 1, 197, 263, 313 and 378.

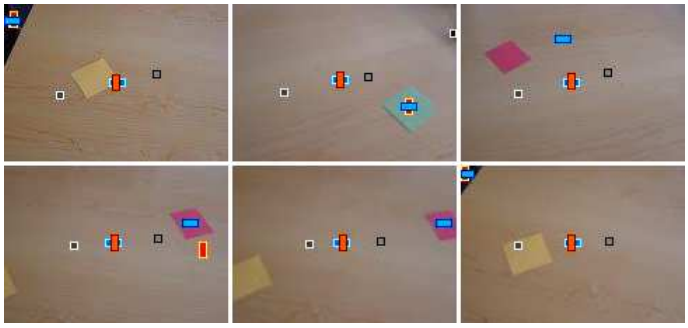

Figure 4.6: Best six key frames of video “Mov1”.
Frames 1, 197, 263, 313, 319 and 378

to improve the visibility. The position of the squares represents the position of the centroids. The size of the centroid is not represented. The red chrominance (U or Cr) colors are represented by vertical red-colored rectangles. The Luminance shows brightness of the inner part for the intensity of the representing bucket. The border is also black or white in order to create a contrast. The same is done for the blue chrominance (V or Cb) for which centroids are represented by horizontal blue colored rectangles. The only features not shown are the time and centroid sizes. 65 % of the information content is represented by these squares and rectangles.

The result of our applications for five key frames is in figure 4.5.

As can be seen, some of the correct key frames are missing; therefore we looked at the smallest detected key frame set in which the ground truth key frame set is included. With six frames we got the result set of figure 4.6 and with seven frame we got the result set of figure 4.7.

It is possible that the sixth frame is equal to one of the other five frames

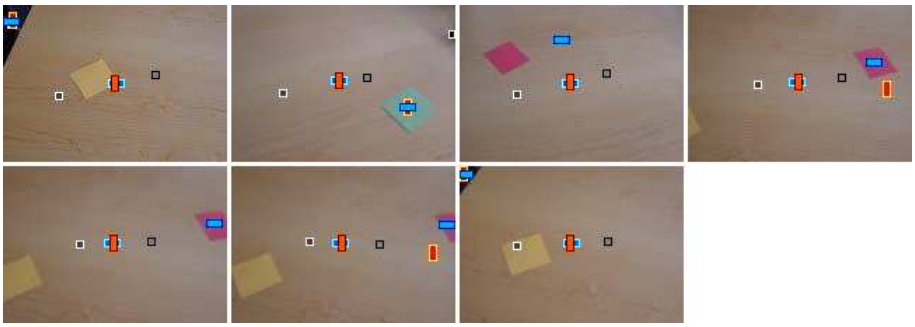

Figure 4.7: Best seven key frames of video “Mov1”.
Frames 1, 197, 263, 313, 319, 320 and 378

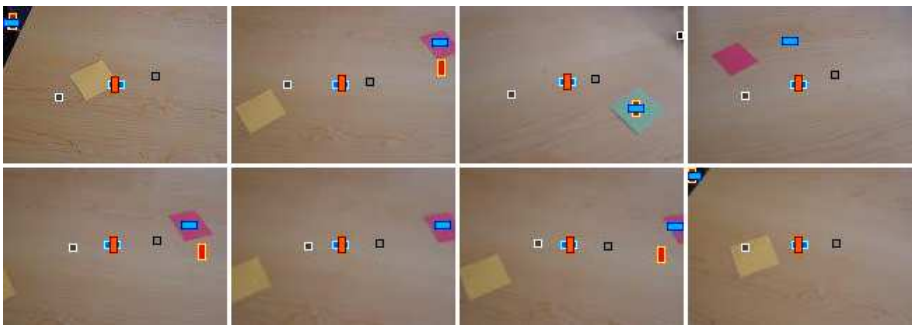

Figure 4.8: Best eight key frames of video “Mov1”.
Frames 1, 55, 197, 263, 313, 319, 320 and 378

because the seventh frame lies between those two frames and is different. We had not expected that the seventh frame nearly equal is to two of the neighbor frames.

The problem that we got here is: “Why is the seventh frame nearly equal to two other frames?” We also observed this behavior in some parts of our ground truth results. The best 7 images of Mov1 are frames 1, 197, 263, 313, 319, 320 and 378. Frames 313, 319 and 320 are nearly the same as can be seen in figure 4.8. As we can see in the middle frame, on the right side is a centroid missing that exists in the two frames. The frame descriptors for these frames 312, 318 and 319 are listed below. The frame descriptor values are natural values in the range $[0, 1000]$ and approximate the real values in the range $[0, 1]$.

frame 313:
`288995 312 1 832 00001212`
`313`

```
0 0 0 317 517 517 689 471  482 0 0 0
0 0 0 890 540   0 496 495  999 0 0 0
0 0 0 489 498 979 835 366   20 0 0 0
```

frame 319:

```
290626 318 1 808 00001218
319
0 0 0 349 501 533 665 488  466 0 0 0
0 0 0   0   0   0 496 495 1000 0 0 0
0 0 0 488 497 982 943 370   17 0 0 0
```

frame 320:

```
290626 319 2 792 00001218
320
0 0 0 351 490 535 663 501  464 0 0 0
0 0 0 903 558   0 496 495  999 0 0 0
0 0 0 492 496 989 968 375   10 0 0 0
```

The data comes directly from the EFT file, with the described format in appendix B.3. The data is the same as for the BFT file with some additional parameters which are represented by the first lines. This first line of each data block contains additional video stream information such as frame offset, MPEG frame type, time code etc., which is used by one of our tools[5]. The second line is the time feature which is represented by the frame number. The third line contains the luminance (Y) color information. The most bright and dark buckets are empty which is reflected by (0 0 0). The fourth line contains the chrominance (U) buckets and the fifth line contains the chrominance (V) buckets.

Figure 4.9: Frames 313, 318 and 319 of video "Mov1" showing the centroid problem.

The most significant difference for these (nearly) equal frames is the second chrominance (U) bucket.

[5]This is the Smart Fast Forward viewer which is described in chapter B.1.5.

frame 313:	890	540	0
frame 318:	0	0	0
frame 319:	903	558	0

The third bucket contains for frames 312 and 319 an area size of 999 (0.999%). Then it could be expected that the second bucket of frames 312 and 319 is not exact (nearly zero); however, the area size of the second bucket of frame 318 is exactly zero. The problem is either in the frame descriptors, in the frame comparison algorithm or in both.

The problem here is that the coordinates for not-existent centroids are undefined but they are handled as if they are zero, which is wrong. The first idea is either not to use this part of the centroid data, if one of the involved centroids does not exist, or to set it equal to the compared centroid. But what happens if every frame always has set one pixel nearly at the lower left coordinate (0,0)? Depending on the frame scale this will normally not have much effect on the other frame descriptors but this chrominance bucket is well-defined for every frame. The values for frame 318 are still (nearly) the same. The only thing that could happen (in the worst case), is that the coordinates of frame 312 and frame 319 will be half so large.

frame 313:	445	270	0
frame 318:	0	0	0
frame 319:	451	278	0

In this case, the frames are nearly identical but still very different in this feature component. The real problem is that the importance of the coordinates is always constant, independent of the amount of information on which it depends. The real amount of information, which is represented by the coordinates, is based on the number of pixels that build the centroid. A better idea is to multiply the coordinates by the size of the centroid. The difference between two less important centroids will always be small. It is only possible for these values to be large when at least one centroid is large enough. We called this coordinate multiplication *dynamic weighting* because the importance and thus the weighting of the coordinates are dynamically modified. The maximum possible X- and Y-range are halved, so the X- and Y-values should be doubled when they are multiplied by the area.

$\hat{c_x} = 2 \times c_x \times c_\#$
$\hat{c_y} = 2 \times c_y \times c_\#$

In our example are features which will be used for frame comparison something like:

frame 313:	0	0	0
frame 318:	0	0	0
frame 319:	0	0	0

So these differences will only have less importance in the frame comparison.

Figure 4.10 shows a comparison of the frame descriptor component of the X-component of the second U-centroid with the dynamic weighting (dynamic) and without the dynamic weighting (static). As can be seen, we have some important improvements as follows:

1. Fast "jumps" in the feature are completely eliminated. This multiplication acts as a kind of filter flattening abrupt changes in pixels that switch between different centroids.

2. The importance of this feature is extremely reduced due to the small size of the centroid.

3. "Pixel noise", which is shown as the randomly added and removed pixels in the centroids, is also removed as a result of the previous two points.

In addition figure 4.11 shows the X-component of the third centroid of the U-color. As a result the importance of this centroid is raised because it contains nearly all pixels in this color component. The quality of the feature seems to be better for use.

We have not implemented the coordinates and size scaling in the feature extraction algorithm. The coordinates are multiplied when they are imported into the discrete curve evolution algorithm. This is done in order to be compatible with older extracted features.

The dynamic weighted feature vector used in the *Discrete Curve Evolution* is defined by the following:

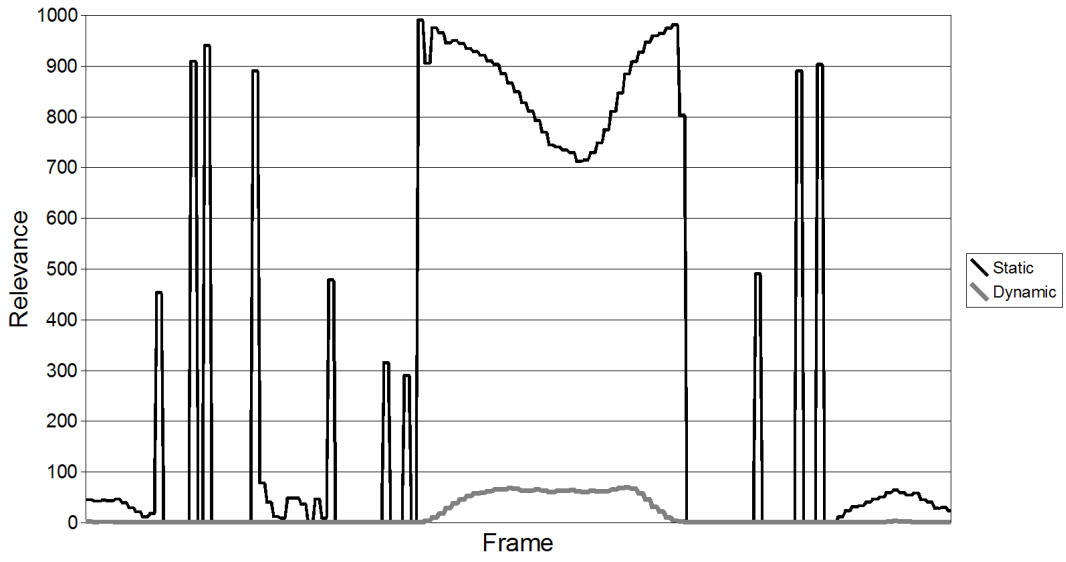

Figure 4.10: Comparison of the X-component of the second U-centroid of the video "Mov1".

$$
\begin{aligned}
f_t \rightarrow FD(f_t) = (\; & \frac{t \cdot C(1+3 \cdot \#centroids)}{framerate}, \\
& \frac{2c_{\#}^{Y_1} c_x^{Y_1}}{c_{x_{max}}^{Y_1} c_{y_{max}}^{Y_1} c_{x_{max}}^{Y_1}}, \quad \frac{2c_{\#}^{Y_1} c_y^{Y_1}}{c_{x_{max}}^{Y_1} c_{y_{max}}^{Y_1} c_{y_{max}}^{Y_1}}, \quad \frac{c_{\#}^{Y_1}}{c_{x_{max}}^{Y_1} c_{y_{max}}^{Y_1}}, \\
& \ldots, \\
& \frac{2c_{\#}^{Y_4} c_x^{Y_4}}{c_{x_{max}}^{Y_4} c_{y_{max}}^{Y_4} c_{x_{max}}^{Y_4}}, \quad \frac{2c_{\#}^{Y_4} c_y^{Y_4}}{c_{x_{max}}^{Y_4} c_{y_{max}}^{Y_4} c_{y_{max}}^{Y_4}}, \quad \frac{c_{\#}^{Y_4}}{c_{x_{max}}^{Y_4} c_{y_{max}}^{Y_4}}, \\
& \frac{2c_{\#}^{U_1} c_x^{U_1}}{c_{x_{max}}^{U_1} c_{y_{max}}^{U_1} c_{x_{max}}^{U_1}}, \quad \frac{2c_{\#}^{U_1} c_y^{U_1}}{c_{x_{max}}^{U_1} c_{y_{max}}^{U_1} c_{y_{max}}^{U_1}}, \quad \frac{c_{\#}^{U_1}}{c_{x_{max}}^{U_1} c_{y_{max}}^{U_1}}, \\
& \ldots, \\
& \ldots, \\
& \frac{2c_{\#}^{V_4} c_x^{V_4}}{c_{x_{max}}^{V_4} c_{y_{max}}^{V_4} c_{x_{max}}^{V_4}}, \quad \frac{2c_{\#}^{V_4} c_y^{V_4}}{c_{x_{max}}^{V_4} c_{y_{max}}^{V_4} c_{y_{max}}^{V_4}}, \quad \frac{c_{\#}^{V_4}}{c_{x_{max}}^{V_4} c_{y_{max}}^{V_4}} \quad)^T
\end{aligned}
\tag{4.4}
$$

With this change the result for five frames are shown in figure 4.12.

The result is not too bad but the fourth key frame is also missing and the last key frame is missing twice. We can see in the last key frame that the table edge appears on the upper left in the black background. Therefore we will have new buckets as can be seen on the drawn rectangles in the upper left. The argument that the area is small and therefore they should not be important in the difference weighting is not correct. That part of the frame is not very large but there are more then only a few pixels as shown in the examples above. The features are as follows:

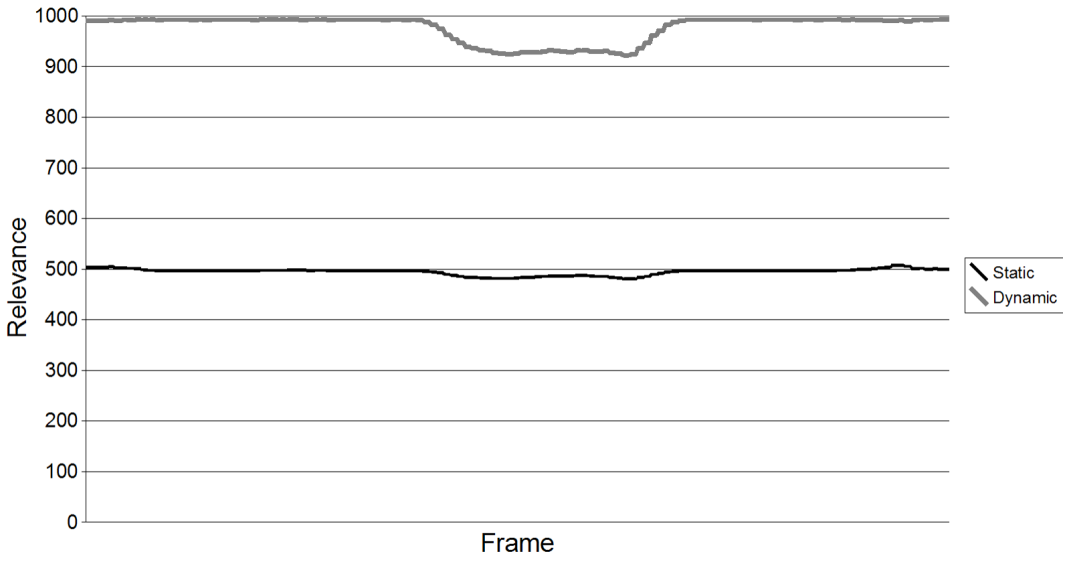

Figure 4.11: Comparison of the X-component of the third U-centroid of the video “Mov1”.

frame 328:

```
298968 327 1 908 00001302
328
  6 16 0 359 493 547 665 498  451 0 0 0
  0  0 0   0   0   0 496 495 1000 0 0 0
  0  0 0 497 495 999   7  10    0 0 0 0
```

frame 378:

```
340340 377 2 4 00001500
378
 30 57 8 260 516 400 663 487  591 0 0 0
  0  0 0  23  50   4 499 497  995 0 0 0
  0  0 0 500 499 992  30  54    7 0 0 0
```

As we can see, the differences are more than only a few pixels. What happens with six frames can be seen in image 4.13.

This time the second key frame is doubled. The difference is also the disappearing edge of the table and background.

frame 100:

```
93483 99 1 940 00000324
100
 86 16 8 208 549 293 623 478  697 0 0 0
```

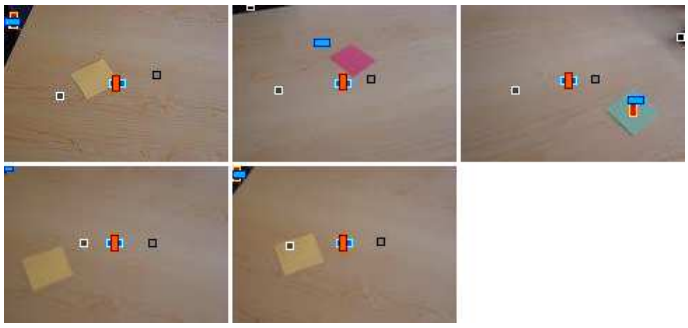

Figure 4.12: Best five frames 1, 100, 200, 328 and 378 of video “Mov1” with the weighting modification for the centroid coordinates.

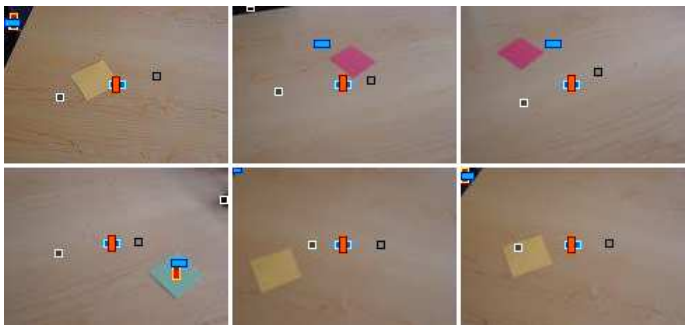

Figure 4.13: Best six frames 1, 100, 127, 200, 328 and 378 of video “Mov1” after the weighting modification.

```
  0   0 0    0    0    0 496 495 1000 0 0 0
  0   0 0 499 502 972 401 246   27 0 0 0
```

frame 127:

```
118005 126 1 992 00000501
127
  0   0 0 284 622 359 616 424  640 0 0 0
  0   0 0    0    0    0 496 495 1000 0 0 0
  0   0 0 498 501 974 417 247   25 0 0 0
```

Figure 4.14 shows the result with seven frames and figure 4.15 with 8 frames.

As can be seen, our ground truth video is not as perfect as we had hoped. At first (see images 4.8 and 4.15) is seems that our improvement with area weighted centroid coordinates is not much better than before. But on the second look, we see improvements in the changes of the “wrong” images - a kind of “quality” improvement. With a numerical analysis of the features and a visual analysis of the frames, it looks like the key frame detection of

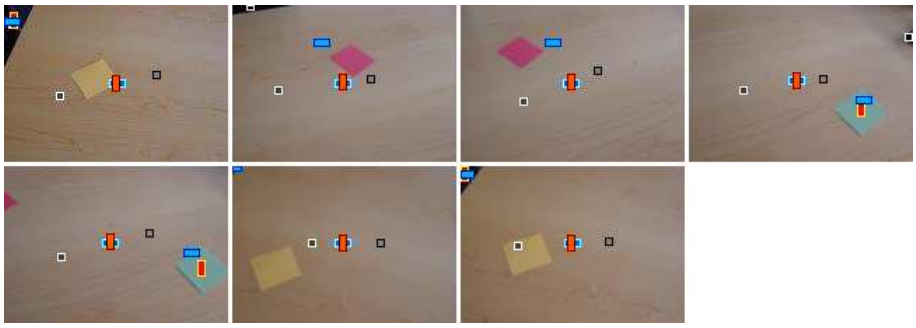

Figure 4.14: Best seven frames 1, 100, 127, 200, 236, 328 and 378 of video "Mov1" after the weighting modification.

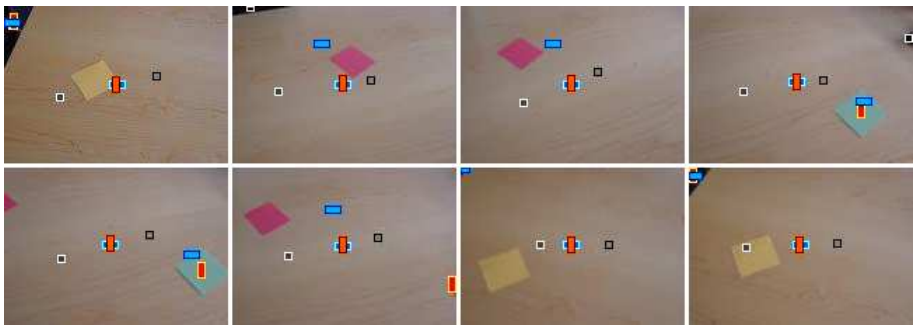

Figure 4.15: Best eight frames 1, 100, 127, 200, 236, 259, 328 and 378 of video "Mov1" after the weighting modification.

the improved algorithm is reasonable. The wrongly detected key frames have some features that could increase their importance.

Results are available for video sequences "Mov1", "Mov3", "Mov00085", "security1", "security7", "mrbeantu", "House Tour" and "Kylie". More information about these videos can be found in appendix A and on the homepage [12].

The frame descriptors contain 37 features which is a histogram subdivision into 4 bins.

Problem:
Our reflections for the dynamic area weighting only have an influence on temporal changes between frames but our reflections have no influence if larger homogeneous areas move are change bins inside a single frame or between frames. Such areas will still abruptly change the centroid data, thus reflecting abrupt changes in the features without a necessary visible change.

A solution could be the introduction of *bucket* weighting for each color value of

a pixel and bucket. The sum of all weights for each possible pixel color should be equal to one. At this moment, the weighting for **only** one predefined bucket is one and for each other bucket is zero.

Figure 4.16 shows the weighting of the color values for each bucket. Our problem could be avoided by creating more linear changes between the buckets when the color is changing. Figure 4.17 shows a suggested weighting for the colors and the associated buckets.

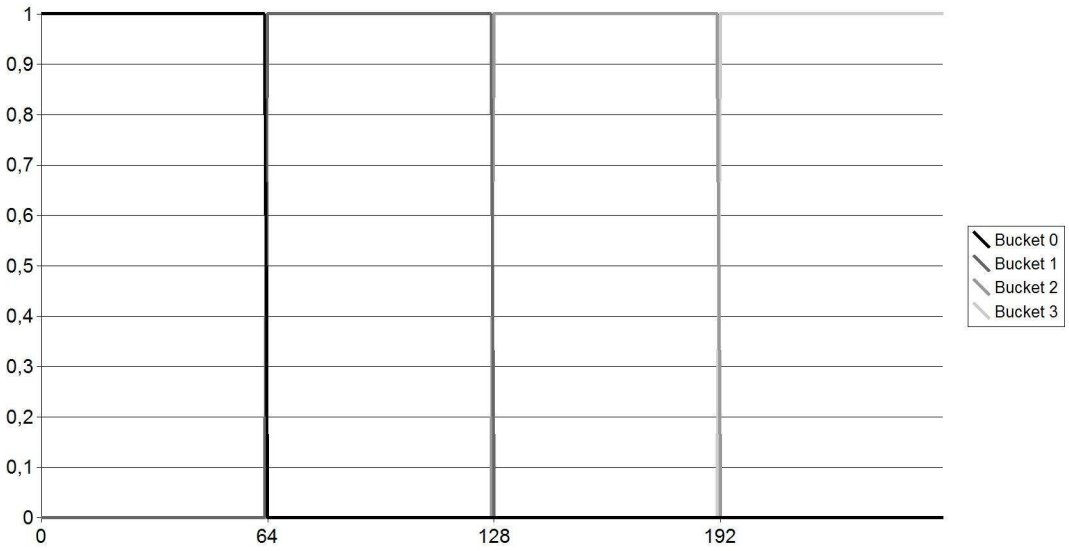

Figure 4.16: Weighting of pixels for associated centroids as implemented.

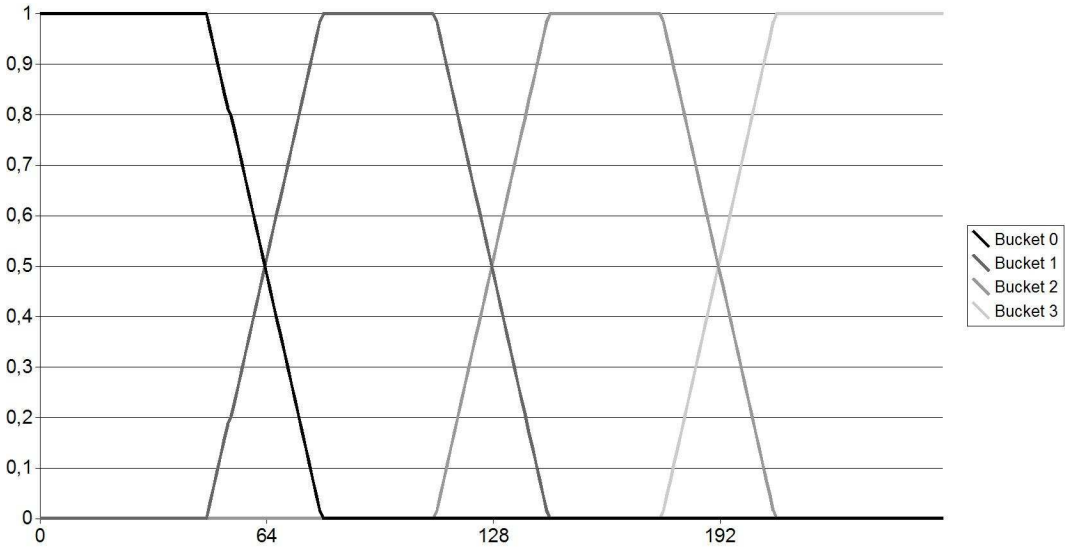

Figure 4.17: Proposal weighting of pixels for associated centroids bins.

Conclusion:
Dynamic weighting of the X- and Y-coordinates of centroids could be useful and an alternative to static filters. Objects moving (slowly) into or from a centroid are well-scaled to their importance depending on their size, so that abrupt bucket changes haven't such an important influence on the features as before and without the dynamic weighting. however, it is not a mean solution for every color transition between different centroids.

4.2.4 Fast frame changes

Events in Mov1

As we have seen in the previous section, our results are not as good as expected. The reason for this problem is features of new or moved centroids which depend on variations in image thus resulting in (too) large variations of the frame descriptors. This seems to be correct because we see that these changes exist in the frames. The problem is our definition of "event". An event is an event if the state of the video is *stable*. (eg. no important changes to the previous state happen). As we have described in chapter 3.1, the features and the relevance measure depend on the application.

In our case, a filter would eliminate the features with a *short* endurance. Our mentioned feature descriptor FD depends on an area of the frames directly around the frame. The result is the filtered features. The window width will depend on our definition (or expectation) of *event*.

Events in halloween

Another kind of problem with videos that we sometimes had was with fast and very short frame switching. This is used for example in the video "halloween", as a kind of video teaser[6]. In the introduction sequence of the video, there is a short sequence of 12 different images with 2 frames each, resulting in 12 image changes in approximately 1 second. These images are a kind of abstract of the video content. The algorithm correctly detects these frames (at a high level of the evolution process) as key frames due to the fact that these frames contain different kinds of information and act as a new shot each; therefore, they should be detected as such.

[6]See appendix A.1.3 for a description of the video properties.

The videos also contains a scene with fast light flashes in which a person on a stretcher is pushed along a hospital floor. Due to the frame descriptor concept[7], each flash is detected as a key frame[8]. Figure 4.18 shows these 9 frames.

Figure 4.18: Key frames (out of 20) from the hospital floor scene at approx. 19", from the video "Halloween".

Figure 4.19 shows the intensity of some colors over the time. The diagram was created with a color code book of 99 colors. It is used here to visualize the changes in the video. The frames 950 to 1550 are underlined in bold.

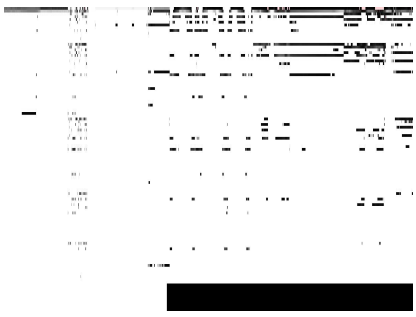

Figure 4.19: Diagram with color intensity over the time of the video "Halloween" with 6182 frames.

Either we defined the requirements or we implemented the key frame implementation incorrectly. A dark/bright switch in the video could be interpreted as an event that occurs or as separate cuts which should be detected such

[7]The buckets subdivided the different colors in different parts with different luminance. This is an argument to use YUV colors because this color space is more like human perception. In such scenes, the luminance only make the most changes. In the RGB color space, all color components should change the buckets.

[8]Note: 9 of the 20 best frames are from this scene (Frames 977, 1044, 1106, 1247, 1294, 1357, 1420, 1454, 1537).

as in the case of the "halloween abstract" in the start sequence of the video. In both cases, our definition of the expected key frame was wrong. What seems to be wrong with our expectation of the key frames? In both cases, the changes are too fast or the context in which the key frames occur is too short.

There seems to be two possible solutions:

1. We could change the importance of the time. Frames which are temporally too near to each other are too difficult.
2. An increase in the local context makes it possible to measure the state of a frame according to the neighbor frames and to detect changes which are too fast in the frame descriptors.

Proposal 1 will get us in trouble if we implement this. Therefore, proposal 2 seems to be more preferable.

It is important that these events be very short in the time and that they not result in a permanent (stable) situation of the scenery. We could add a limitation to the *event* requirement 3 in chapter 3.1 by defining the shortest durability of the situation before it is detected as an *event*.

The easiest way to fulfill this limitation is by filtering features that do not match this. We have decided to remove such features completely by implementing a morphological filter, which results in a new question: "How long should it take for a situation to be detected as a *stable event*?" In terms of the filter the question is, "How width should the filter be implemented?".

Also necessary to consider is how important was the information that we lost.

In some video scenes, we had speckles, noise and very fast image changes without any information (to confuse the watcher). It makes sense to remove/filter such features/frames from the video sequence. This is implemented by creating a morphological filter. We have also implemented a Gaussian filter for comparison purposes.

Grey-Level morphological filter

Morphological filters are widely used in the image processing to filter single pixels in an image depending on the neighborhood pixels. There exist

different and wide-varying kinds of pixels, some of which are only used as horizontal or vertical pixels, or as a combination of both of these as the neighborhood. Some pixels have a local area completely surrounding the filtered pixel, and sometimes in a video sequence, there are also the pixels of neighbor frames used to filter a single pixel.

Morphological filters were introduced by J. Serra [49] and are implemented by an erosion and a dilation function. Out intent is not to use the filter for pixels but to filter each frame descriptor FD^n of the frame f_t in the time domain t. We used grey-scale version of the erosion and dilation functions.

Definition:

$$f(z) = -\infty \text{ if } z \text{ is outside the definition domain of } f \tag{4.5}$$
$$\text{Domain:} D(f) = \{z | f(z) > -\infty\} \tag{4.6}$$
$$\text{Translation of f by x:} f_x(z) = f(z - x) \tag{4.7}$$
$$\text{Domain translation of f by y:} (f + y)(z) = f(z - x) + y \tag{4.8}$$

Erosion:
The erosion of f by a structuring element g at point x is defined as

$$(f \Theta g)(x) = min\{f(z) - g_x(z)\}; z \in D(f) \cap D(g_x), x \text{ such that } D(g_x) \subseteq D(f) \tag{4.9}$$

In our case,

$$f(t) := FD^n(frame(t)) \tag{4.10}$$
$$g : \{-i, ..., i\} \to 0 \tag{4.11}$$

The total definition domain of g and also the resulting filter width is $2i + 1$.

Dilation:
The dilation of f by structuring element g at point x is defined as

$$(f \oplus g)(x) = max\{f(z) + g_{-x}(-z)\}; z \in D(f) \cap D(g_{-x}(-z)) \tag{4.12}$$

The operation used by us is the *opening* operation, which is defined as a dilation followed by an erosion. The opening of an input signal A by a structuring element B is defined by

$$A \circ B = (A \Theta B) \oplus B \tag{4.13}$$

The second used operation is the *closing* operation, which is defined as an erosion followed by a dilation.

$$A \bullet B = (A \oplus B) \Theta B \tag{4.14}$$

The filter width should be free definable to get asmuch flexibility as possible.

Our application implements the erode and dilate operations as basic functions. The opening and closing operations are implemented by applying these basic functions in the correct order.

- erode function with filter width $2i + 1$:
  $erode(FD(t_n)) = min(FD(t_n - i), ..., FD(t_n), ..., FD(t_n + i))$
- dilate function with filter width $2i + 1$:
  $dilate(FD(t_n)) = max(FD(t_n - i), ..., FD(t_n), ..., FD(t_n + i))$

With these implementations, the morphological filters perform opening and closing operations. The opening operation reduces all local maxima inside the filter based on the width of the filter function g and its values. The closing operation reduces all remaining[9] local minima. Minima and maxima outside the definition width of g still exists. A full morphological filter performs an opening operation followed by a closing operation. (The inverted order closing followed by an opening operation would also be possible.)

Figure 4.20 shows an example of the intensity values of a filtered and an unfiltered feature over the time. The frame descriptors are from the video "Mov3" and contain 73 features. The parts where Rustam is weaving are represented by a clear change in the feature. The four marked areas show some changes in the feature which are a result of the filter.

The light gray curve is the unfiltered original feature and the dark grey curve is the filtered feature with $i = 2$ which results in a window width of 5. The black curve is the filtered feature with $i = 5$ and a window with of 11 frames.

In area one, in the upper left part of the image, we can see that peaks in the curve are flattened. This can also be observed in the unmarked middle peak of the curve.

In areas two and three, in the upper right part of the image, we can see that smaller peaks (area 3) were completely removed with a smaller filter window

[9]It is possible that after the opening operation no values are left for which local minima exists.

and that wider peaks are remained (area 2). These peaks are removed by the wider filters.

In area four, in the lower left part of the image, we can see that some noise was removed by the filter. A value of $i = 2$ seems to be enough for this effect.

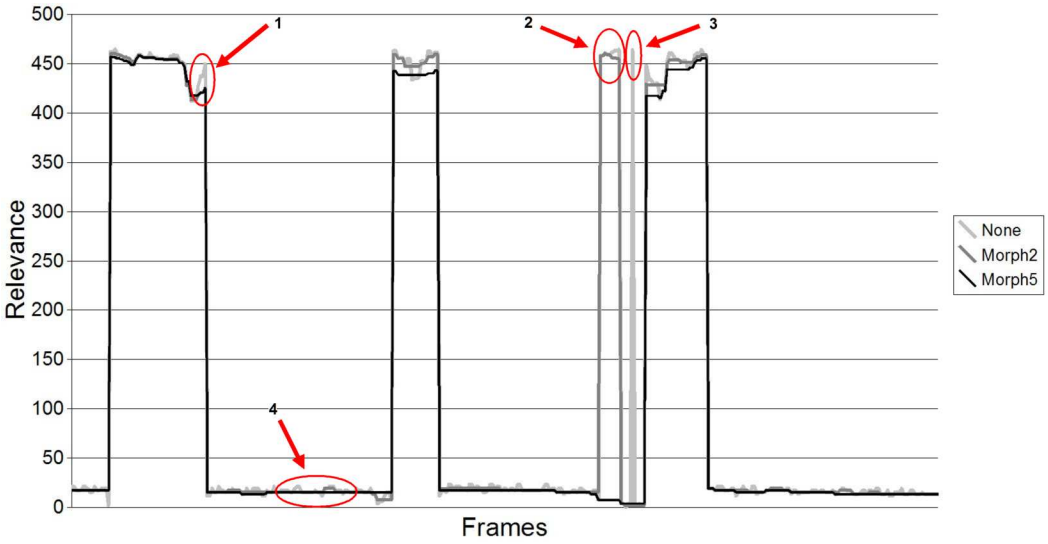

Figure 4.20: Diagram of a morphologically (un)filtered dynamic weighted feature from video "Mov3".

How wide should the filters be?
In imitation of our efforts in section 4.2.1 to make a framerate independent algorithm, we should avoid any kind of frame numbers in the filter expression. Our key frame expectation is based on our impression of the time context in which they appear and it seems to be logical to define a suitable filter in time even if it is based on frames. In that case, the frame width should be broken down to frames (based on the expected filter width in seconds and the frame rate in frames per second).

Due to the problem that in our scripts, at the time that the filter was applied, the frame rate was not available, therefore we could not implement this requirement.

The newest versions of the experiments are all made with different versions of the morphological feature filter. "Morph$n$" stands for a filter width of $2n + 1$ pixel's (n pixel left and n pixel to the right of the origin pixel). The used filter width and frame rate are shown in table 4.2.4.

naming	filter width	10 fps	15 fps	25 fps	30 fps
Morph1	3 frames	0.30”	0.20”	0.12”	0.10”
Morph2	5 frames	0.50”	0.33”	0.20”	0.17”
Morph3	7 frames	0.70”	0.47”	0.28”	0.23”
Morph5	11 frames	1.10”	0.73”	0.44”	0.37”
Morph11	23 frames	2.30”	1.53”	0.92”	0.76”

Table 4.3: Table of effective temporal filters with different frame widths and different frame rates.

Normally “None”, “Morph3” and “Morph5” are used. Sometimes “Morph1” and “Morph2” are also used but seldom “Morph11”.

Other filters

There are several possible filters that could be used to detect key frames with the discrete curve evolution. The usability of these filters depends on the application and environment in which they are used. We have for example also implemented a flexible gaussian filter.

For a Gaussian filter width $2i+1$, the constants C_{-i}^{gauss} till C_{i}^{gauss} are calculated and normalized so that the sum of the constants is equal to one.

$$\begin{aligned} &C_x^{gauss_i} := \tfrac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \\ &\text{with } x \in \{-i, ..., i\} \\ &\text{and appropriated } \sigma \text{ and } \mu \end{aligned} \tag{4.15}$$

$$\begin{aligned} FD^{gauss_i}(t_n) \;:=\; & C_{-i}^{gauss_i} \cdot FD(t_{n-i}) \\ +\; & ... \\ +\; & C_{0}^{gauss_i} \cdot FD(t_{n}) \\ +\; & ... \\ +\; & C_{-i}^{gauss_i} \cdot FD(t_{n+i}) \end{aligned} \tag{4.16}$$

Other filters not implemented could be for example the meridian filter.

Conclusion

Filters can be used at various abstraction levels of the feature creation process. Not implemented and not tested are filters at the lower levels. This could be for example pixel filters, which filter the data directly inside the

images depending on the surrounding content. Possible filters are 2D-filters which only use the content of that frame, and also 3D-filters which use the pixels of previous and successor frames. Also accommodated at this level are filters which intervene directly in the MPEG-datastream by filtering for example the DC components which is done in [18]. The proposed weighting of the pixels assigned to centroids could be distinguished with a filter at a higher level.

As we can see, it is possible to eliminate short "jumps" of larger areas between different buckets with morphological filtering. The width of the window should depend on the duration of the event definition of the application.

Not tested and not implemented are filters defined for a single frame or for the histograms. Image filters can be used to eliminate noise which will have an effect on the histogram and thus the centroids which are based on these. Histogram filters could be implemented by moving a filter window over the histogram values [31]. This could be, for example a window filter or an image filter.

4.2.5 Selecting key frames from the frame list

Another kind of question not yet answered is not only the abstraction level (how many frames do we need) but also the frames which are needed. The last question is somewhat strange because our evolution process defines the order of the frames. With the abstraction level it is possible to define the frames which are needed from the list.

The problem is that we didn't define before-hand what we understood under *the* abstraction level. This is also a definition depending on the observer who selects the abstraction levels. The easiest way to define the abstraction level is to say how many (either absolute or relative) number of frames are expected or needed. For other applications and users like [53] it makes sense to define different abstraction level and expectations. Selecting a different abstraction level will result in a different amount of frames belonging to the abstraction level.

Our Curve Evolution process gives us some information to create and define some different abstraction levels. The main information is the frame number in the video sequence, the relevance level at the frames removal time and the position number when it was removed.

The "Rustam" experiments from section A.1.1 shows that it is also important to know the relevance of the key frames in the evolved frame list.

Sometimes an important key frame is also removed during the discrete curve evolution and the resulting key frame is not important, e.g. it's like the other neighbor of the removed key frame.

Not every frame at the end of the evolved frame list is sometimes a key frame. We could see that not every frame was a key frame. If a key frame exists between two identical frames, then one of these surrounding frames should be a key frame, only if this inner frame is removed from the key frame list.

For this important problem there is more work necessary. It seems to be possible to create with the relevance values and the available number of frames a logic to join frames to groups and to define the application depending on useful and suitable boundary for the number of key frames.

For example: A decrease in the relevance value, in relation to the previous key frame, is for example interpreted as a less important frame in the new context. This removed frame makes more sense in the context of the key frames removed before-hand. A solution is to merge these two frames together. Abstraction levels could then be defined on merged key frame groups instead of on single key frames.

4.2.6 Number of key frames

As we have seen in chapter 2.2.1 the number of a key frames or an lower boundary for the relevance value is needed to make it possible to define the result frame set. It is not possible to define the size of the expected result set without these information.

Nevertheless for some of the comparison videos a full set of key frames are needed to make comparison possible. So we tried some algorithms to find a usable number of key frames. This number should not be a perfect value, but a more rough approximation of the expected result sets. Maybe it is possible to find usable algorithm that will match with this background information the number of frames in the result set.

As we have described, the number of key frames will depend greatly on the expectation of the kind of frame differences that are *important* enough to be marked as such. Our goal here is to find such an importance level.

For example, someone could define in "Mov3" (appendix A.1.1) only the frames, where Rustam is weaving, as key frames but the frames between these frames are no key frames for the observer. When we take a look in the key frame list, we will find these frames, if we define the number of key

frames as the best five frames. On the other hand, an observer could also define the frames between the waves, where Rustam does nothing, also as key frames (we however see a lesser relevance). We will get this result if we raise the number of key frames to seven. So our evolved frame list will give different key frame lists for the different importance definitions of different observers.

Figure 4.21: The best three key frames of "MOV3"

Figure 4.22: The best six key frames of "MOV3"

The evolved list contains all frames of the video sequence in the order of their relevance. Only the frames with the highest order in the evolved list are key frames. The problem is to define the number of frames which are key frames. We have various information such as the number of the frames in the evolved frame list, the relevance when they are removed and the frame number of the frame self in the video. We did not find a useful or calculable value for the maximum number of frames which could be used to define the key frames.

4.3 Dominant Colors and Optimal Color Composition

The histogram based frame descriptors are easily implemented and the information content seems to be good for key frame extraction. The results also make sense on the base of the information content. The idea is to use features which are more intuitive for the human understanding of the used frame information.

4.3.1 Dominant Colors as Frame Descriptor

Hu et. al [29] has used *Dominant Colors* of frames as a frame descriptor. It has been shown that, in the early perception stage, the human visual system performs identification of dominant colors by eliminating fine details and averaging colors within small areas [41]. Consequently, on the global level, humans perceive images only as a combination of the few most prominent colors, even though the color histogram of the observed image might be very *busy*. Based on these findings, we performed extraction of perceived colors through the following steps. First a color image was transformed from the RGB space into the perceptually more uniform Lab color space. This color space was designed in such a way that the color distances computed with $||\ .\ ||_2$ matches the subjective impression of color likeness [51]. The set of all possible colors was then reduced to a subset, defined by a compact color codebook with a size of 16 to 512 colors [41]. This code book has, in our case, 99 colors. Finally, a statistical method was applied to identify colors of speckle noise and remap them to the surrounding dominant color (see [29] for details). A color component with index i was considered to be dominant if P_i exceeds a threshold (typically $2-3\%$). The result was a rough segmentation of an image with just a few colors.

The comparison contains the following steps. The dominant colors of the images are lined up into a vector. Each value represents a fixed area size of the image. (3 when using a vector with 33 components) $(a1, ..., an) and (b1, ..., bn)$

Once the perceptually dominant colors are extracted, we represent a color composition of an image I by vector of areas occupied with dominant colors $(CoCo(I) =< P_1^I, P_2^I, ..., P_{99}^I >$, where P_i is the area percentage occupied by the color with index i).

A 2D representation of a video sequence $V = \{I_t\}$ is the sequence $CoCo(V) = \{CoCo(I_t)^T\}$, where t is the frame index. $CoCo(V)$ is a 2D array with

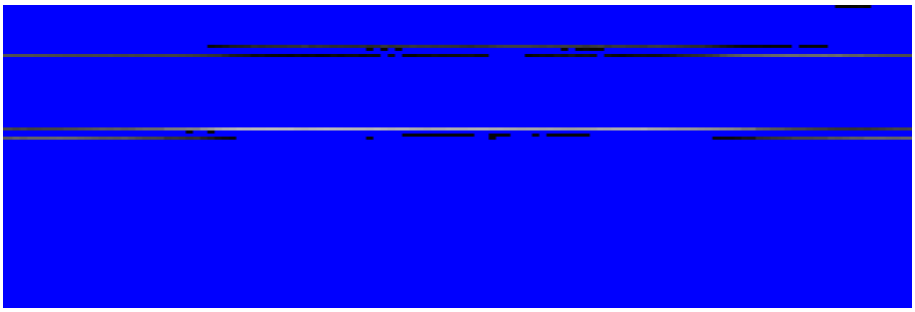

Figure 4.23: Intensity composition representation of Mov1 video clip.

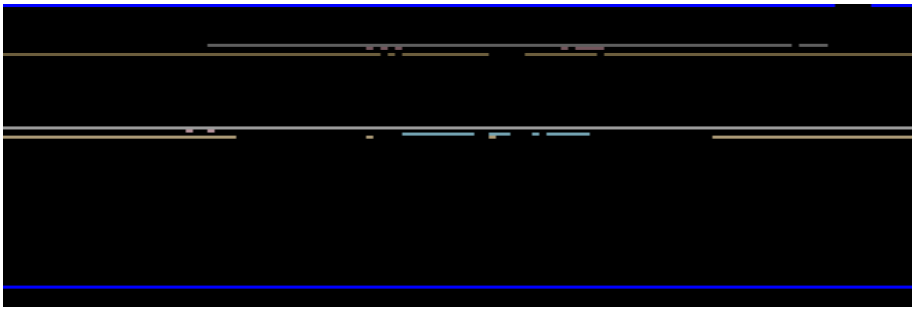

Figure 4.24: Color composition representation of Mov1 video clip.

each column being the color composition of a single image ($CoCo(I_t) =< P_1^{I_t}, P_2^{I_t}, ..., P_{99}^{I_t} >$). Consequently, row i (for $i = 1, ..., 99$) represents the area distribution of color, with index i, over the whole video sequence.

Figures 4.23 and 4.24 show a visual interpretation of the frame descriptors $CoCo(V)$ for the "Mr. Bean" clip. The vertical dimension of the images is 99 and the horizontal is the number of the frames in pixels. Figure 4.23 shows the intensity of the different color components - the brighter the color of the pixel (t, i), the higher the area percentage of the color with index i in the frame t. Figure 4.24 shows the available colors with intensity unequal to zero. Black means that the color is not available in the frame. Not available black colors are represented by the color blue. (This is the first line and line 93 nearly at the bottom.)

4.3.2 Optimal Color Composition distance

Based on human perception, two images are considered similar in terms of color composition if the perceived colors in the two images are similar, and if similar colors also occupy a similar area percentage [41]. To compare the

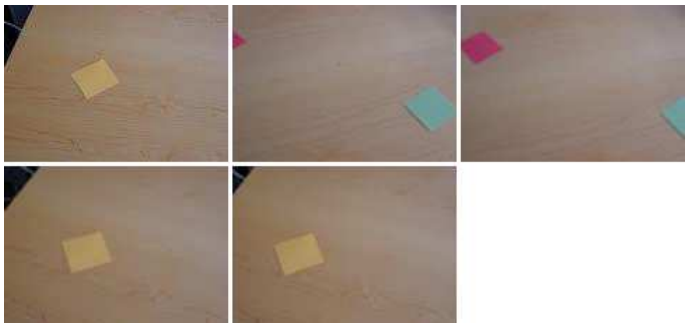

Figure 4.25: Key frame result of video "Mov1" with 5 frames. We used Dominant Colors as frame descriptors, without filtering, and relevance measure of formulae 4.1.

two images A and B, we use the optimal mapping function from [29] that minimizes the overall mapping distance between representations $CoCo(A)$ and $CoCo(B)$. It is called Optimal Color Composition Distance (OCCD) and is denoted $d(A, B)$.

A semi-metric called Optimal Color Composition Distance (OCCD) was developed to capture both criteria [41]. To compute the OCCD, the set of color components of each image was first quantized into a set of n (typically $20-50$) color units, each with the same area percentage p, where $n \times p \approx 100$. We call this set, the quantized color component (QCC) set. Suppose we have two images A and B, with QCC sets $\{C_A \mid U_A^1, U_A^2, ...U_A^n\}$ and $\{C_B | U_B^1, U_B^2, ...U_B^n\}$. Let $I(U_x^k)$, $x = A$ or B, $k = 1..n$, denote the color index of unit U_x^k, and $\{M_{AB} \mid m_{AB} : C_A \rightarrow C_B\}$ be the set of one-to-one mapping functions from set C_A to set C_B. Each mapping function defines a mapping distance between the two sets: $MD(C_A, C_B) = \sum_{i=1}^{n} W(I(U_A^i), I(m_{AB}(U_A^i)))$, where $W(i, j)$ is the distance between color i and color j in a given color code book. Our goal is to find the optimal mapping function that minimizes the overall mapping distance. The distance $d(A, B)$, between the images A and B, is then defined to be this minimal mapping distance.

This optimization problem can be described as the problem of minimum cost graph matching, for which there exist well-known solutions with $O(n^3)$ complexity [27]. Note here that n is the number of quantized color components, which roughly corresponds to the maximum number of dominant colors a human being can distinguish within *one* image. n usually much smaller than the color code book size.

Figure 4.25 shows the best five frames of the sequence Mov1. The bad results of the last two images can be explained by the fact that the camera moves

Figure 4.26: Key frame result of video "Mov3" with 7 frames. We used Dominant Colors as frame descriptors, without the filtering, and relevance measure of formulae 4.1.

from right to left, and then the upper left part of the video becomes black. The part at the right that is removed from the camera sight is very bright. These area changes and the very intense difference in the brightness seem to be important enough to become a significant difference between those frames.

Figure 4.26 shows that results for the video Mov3 are acceptable. We got a slight difference in the expected key frames when we increased the number of frames, as we can be seen in figure 4.26.

4.3.3 Filtering

We filtered the dominant colors of the time for each of the colors in the code book.

We have used here the same idea and algorithms as for the centroid features. Due to this fact, we also discovered some of these problems with the dominant colors. We hoped that these problems could also be removed by morphological filters. We used the same idea here to apply the morphological filter on the dominant colors. Each of the possible 99 code book colors was handled as a separate feature with a given intensity.

The representation $CoCo(V)$ of a video sequence V can contain eventual instabilities due to instabilities of the color segmentation in single frames. We used time (frame number) dependencies of the frames to filter the instabilities. We applied morphological opening and closing to each row of $CoCo(V)$ with the support size of 11 frames. This allowed us not only to filter out the instabilities but also to eliminate the extraordinary images like completely

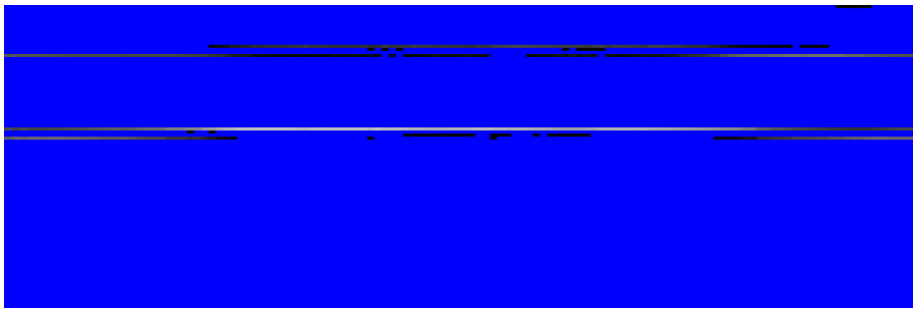

Figure 4.27: Intensity Mov1 without any filter

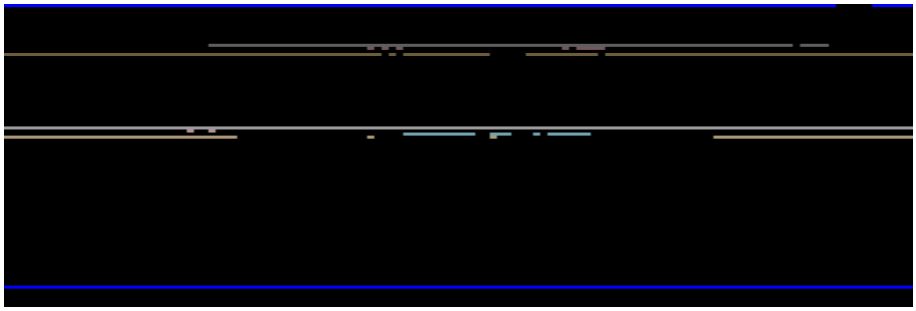

Figure 4.28: Color Mov1 without any filter

white (e.g., due to blinding light) or black images (e.g., lack of light) that last just a few frames. Such images belong to the common effects of movie productions. It does not make sense to consider such images as candidates for key frames, since they do not contain any information about video sequences. After applying the morphological opening and closing to $CoCo(V)$, we obtained a filtered representation of the video sequence, which we will denote by $CoCo'(V) = \{CoCo'(I_t)^T\}$. We applied the distance function d to the filtered representations of images, i.e., in the following, $d(A, B)$ denotes the optimal mapping function applied to $CoCo'(A)$ and $CoCo'(B)$.

The set of vectors $CoCo'(V) = \{CoCo'(I_t)^T\}$, together with the distance function d, form a metric space. In this space, the sequence $\{CoCo'(I_t)^T\}$ is the trajectory of video V that can be viewed as a polyline. We obtained key frames of video V by simplifying the polyline $\{CoCo'(I_t)^T\}$.

These intensities are filtered over time for each color code. The total percentual area for each frame is scaled to 100%. Figure 4.27 shows the unfiltered color intensities and figure 4.28 shows the available colors of video Mov1. Figure 4.29 shows the morphological filtered color intensities, with a filter width of 7 frames, and figure 4.30 shows the available colors of video Mov1. Figure 4.31 shows the morphological filtered color intensities, with a

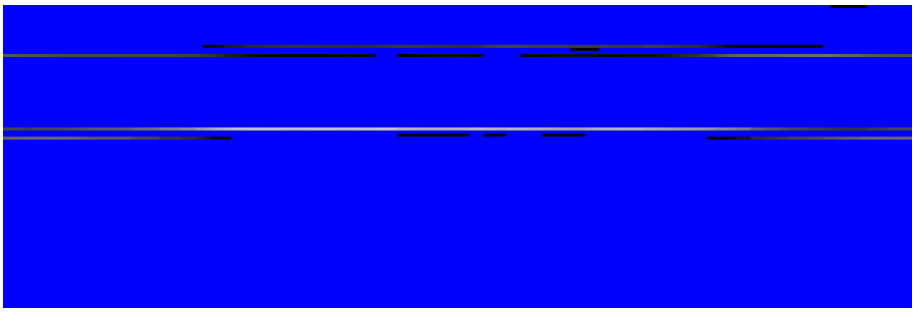

Figure 4.29: Intensity Mov1 - Morph3

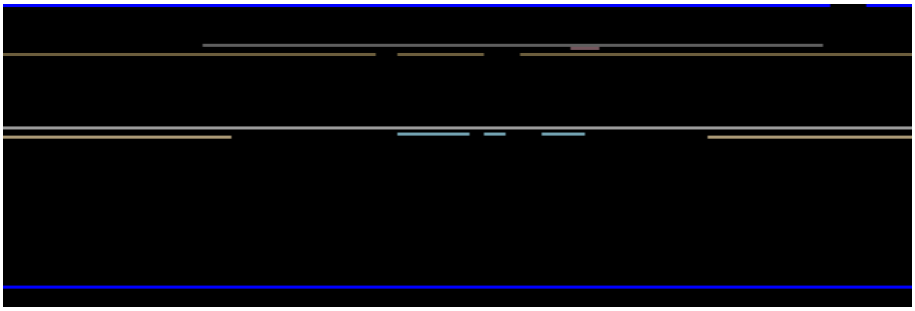

Figure 4.30: Color Mov1 - Morph3

filter width of 11 frames, and figure 4.32 shows the available colors of video Mov1.

4.3.4 Coordinates of the dominant colors

With this definition use of the dominant colors (DC), we got a lot more additional information than we had in the bucket definition. There is no position information about the colors available. Mirrored images with this definition are identical. Even completely different images with the same color usage are identified as identical. Slow movements of an object behind a homogene background are also not well-detected.

We hoped that adding the centroid coordinates of the dominant colors would improve our results. The probability of having different images with the same dominant colors and the same centroid coordinates would be definitely lesser than without the centroid coordinates.

We changed the algorithm in such a way that the coordinates of the dominant colors centroid are also stored. (See appendix B for a detailed format description).

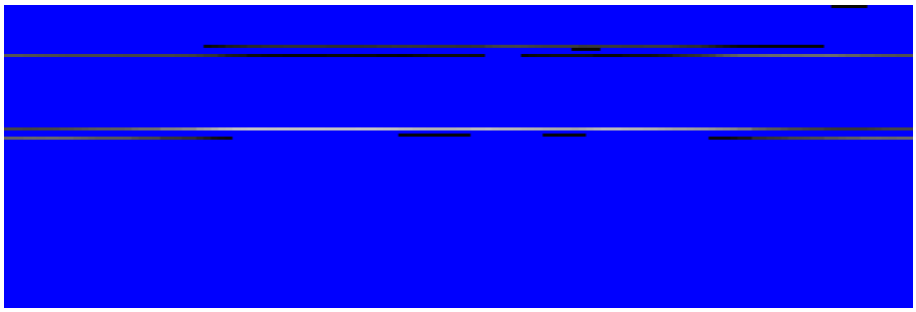

Figure 4.31: Intensity Mov1 - Morph5

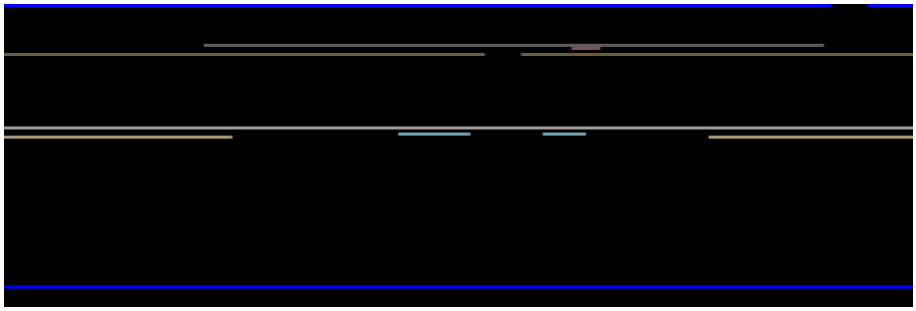

Figure 4.32: Color Mov1 - Morph5

The euclidian distance between these coordinates was calculated and added as a second component. These components were added with different weights as shown in formulae 4.3.4.

$f \in [0,1]$; $d_{combined}(I_1, I_2) = f d_{OCCD}(I_1, I_2) + (1-f) d_{coordinates}(I_1, I_2)$

Problem:

1. The shading of a color change. The OCCD supports this correctly by matching the colors. The coordinates don't reflect this color likeness.
2. New parts of objects move into the image. The OCCD supports this correctly by matching the area of the colors. Smaller objects are less important than bigger objects. The coordinates do not reflect this area factor.

Unfortunately we could not find a usable value for f. The best values would be very small values. The conclusion is that either the second part (coordinate distance)of the formulae is very large (which could be possible by the suggested problem above) or it is not well-scaled in relation to the first part (color distance).

Chapter 5

Experiments on closed videos

In this chapter are the discussed improvements tested with our experimental videos. The experiments we have done contain a variable and wide spectrum of tested components. These contain various number of buckets (as defined by Daniel DeMenthon). This chapter contains experiments with the two weights of the centroid coordinates and different versions of the cost function. It also contains different feature color spaces and different morphological feature filters.

We also tested cost functions that contain a larger local context, $Loc_2(f_t) = \{f_{t-2}, ..., f_{t+2}\}$. Our experiments are made with several video sets and different application configurations. An exact content description and statistical summarization of the videos is in Appendix A. The experiments are often based on different application configurations. These applications and configurations are described in Appendix B. The experiments in the following subsections contain only subsets of these videos and applications.

5.1 YUV vs. RGB

These experiments show the flexibility in the selection of different color space for the experiments. Figure 5.1 shows the best five key frames of the video sequence Mov1. It is made with the frame descriptors based on a YUV histogram subdivided into 16 bins, which result in 145 features. As we can see, the last key frame appears twice and the frame before the last key frame is missing. Figure 5.1 shows the best five key frames of the same video sequence created with a RGB histogram subdivision of also 16 parts. As we can see, the resulting frame set contains the expected key frames.

As is shown in this experiment, the RGB color space, depending on the situation in which it is used, could be better than the YUV color space. The results of the RGB colors match our expected key frames exactly. Other experiments have shown that the decision to use RGB colors is not always the better choice. This shows us that not only the kind of selected features is important but also the color space on which they are based.

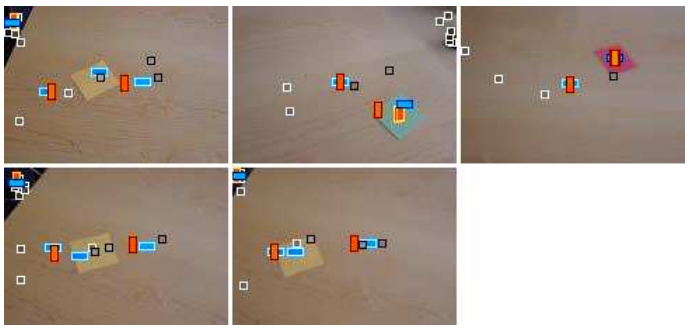

Figure 5.1: Figure with the best five YUV frames of "Mov1" without filtering

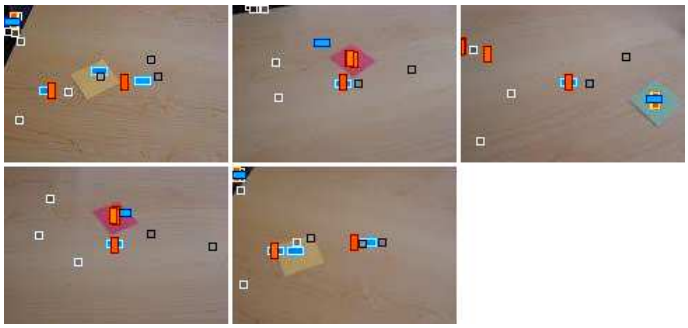

Figure 5.2: Figure with the best five RGB frames of "Mov1" without filtering

5.2 Different Cost Functions for the centroid feature

Our algorithm has the possibility to define different kinds of similarity functions. So could, for example,

$$\begin{aligned} M_{rel}(f_t, Loc_2(f_t)) &= d(f_{t-2}, f_{t-1}) \\ &+ d(f_{t-1}, f_t) \\ &+ d(f_t, f_{t+1}) \\ &+ d(f_{t+1}, f_{t+2}) \\ &- d(f_{t-2}, f_{t+2}) \end{aligned}$$

define a relevance measure with a larger local context. Thus so it could first detect changes which are either very slow or which occur over a longer frame period.

Another idea is, for example,

$$\begin{aligned} M_{rel}(f_t, context(f_t)) &= d(\overline{f_{t-1}, f_{t+1}}, f_t) \\ &= \frac{\langle f_t, f_{t+1} - f_{t-1} \rangle}{||f_t||} \end{aligned} .$$

5.3 Comparison of YUV centroids vs. OCCD

Our experiments are made with centroid based features including 37 components, YUV buckets and dominant colors. The cost function is in both cases the same as in formulae 4.1. We will present here the ground truth video sequence "Mov1", "Mov3" and "Mov00085".

Video sequence "Mov1"

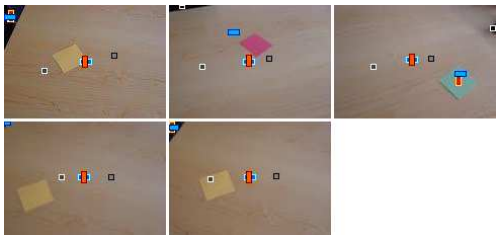

Figure 5.3: Result for "Mov1" with centroid based buckets with 37 frame descriptors, without filter.

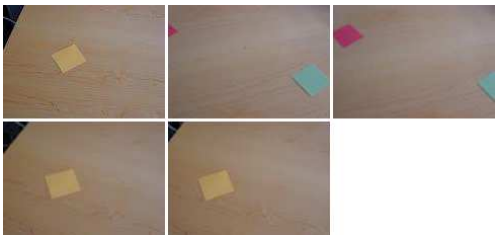

Figure 5.4: Result for "Mov1" with Dominant Colors, without filter.

Figure 5.3 shows the well-known best five images of the video sequence *Mov1* with unfiltered features based on the centroids with YUV colors. The number of 37 frame descriptors is low. In this result set, key frame four is missing and frame number five is available twice. This is a result of the upper left table border and the black background. The image quality of the resulting frames in comparison to the expected frames is good. The papers are well-placed and they are completely visible.

Figure 5.4 is the counter-part of this video for the dominant colors. In this result set, frames two and three are mismatched twice to one single frame. Frame two could be associated more to key frame three and frame number

order	Expected (range)	Resulting frames	
		YUV	OCCD
1	0- 22	0	0
2	82- 141	99	
3	192- 231	199	
none			237
none			249
4	267- 309		
5	340- 377	327	348
		377	377

Table 5.1: Key frame number and the resulting frame numbers of the different features for the video *Mov1*.

three to the expected key frame number four. Key frame number two is completely missing. The last key frame appears twice, despite the appearance of the table edge and black background, which appear in the upper left-hand corners.

Comparison experiments with enabled filters shows that the results are no better than without filters. In some cases, only the quality of the selected key frame is improved. For the dominant color, one mismatched frame is for example replaced with a frame with exactly one paper.

Table 5.1 shows an overview of the exact results of both features for the sequence "Mov1". The results for both kinds of frame descriptors are not acceptable. It could be that either not enough information or the wrong information was available. For the buckets, it could have been too few buckets or too small color space range. For the dominant colors, it could have been a wrong code book or the size of the dominant colors was too large, so that small parts of other pieces of paper were not detected.

Video sequence "Mov3"

Figure 5.5 shows the ground truth result of the video "Mov3". Figure 5.6 shows the key frame result of the video segmentation based on buckets with 37 frame descriptors including dynamic area weighting without filtering. Figure 5.7 shows the result for the dominant colors.

As can be seen, the results based on dominant color have a better quality than the centroid based features. Table 5.2 shows the exact results for the comparison experiment. Both kinds of frame descriptors give good results. It

Figure 5.5: Ground truth result for “Mov3”

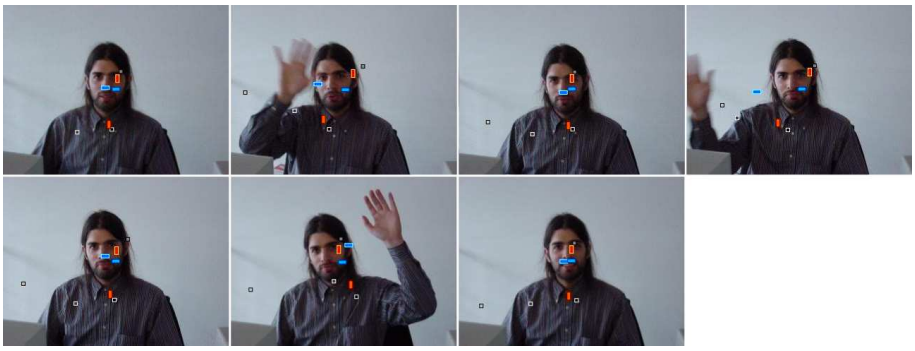

Figure 5.6: Result for “Mov3” with centroid based buckets with 37 frame descriptors, without filter.

Figure 5.7: Result for “Mov3” with Dominant Colors, without filter.

seems that, in contrast to the “Mov1” experiment, the amount of information in the frame descriptors is enough to detect the key frames.

order	Expected (range)	Resulting frames	
		YUV	OCCD
1	0-	0	0
2	34- 58	46	36
3	63- 127	72	63
4	142- 163	159	144
5	163- 222	167	181
6	238- 274	238	240
7	322- 377	377	377

Table 5.2: Key frame number and the resulting frame numbers of the different features for the video *Mov3*.

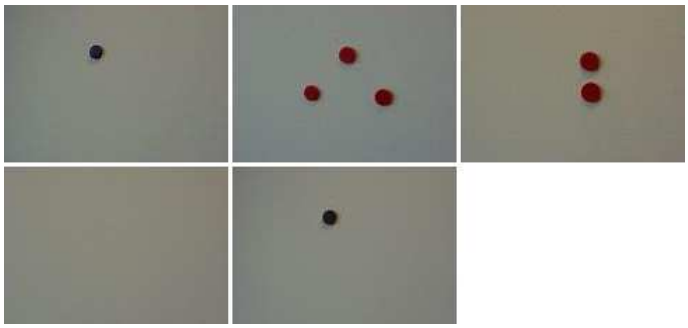

Figure 5.8: Ground truth result for "Mov00085"

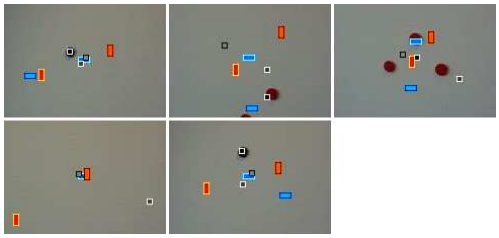

Figure 5.9: Result for "Mov00085" with centroid based buckets with 37 frame descriptors, without filter.

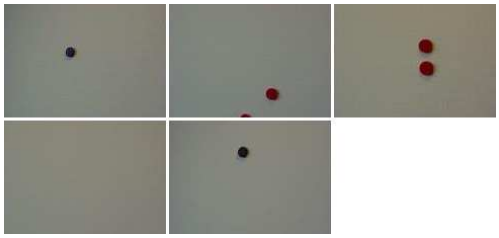

Figure 5.10: Result for "Mov00085" with Dominant Colors, without filter.

Video sequence "Mov00085"

Figure 5.8 shows the ground truth result of the video "Mov00085". Figure 5.9 shows the key frame result of the video segmentation based on buckets with 37 frame descriptors including dynamic area weighting without filtering. Figure 5.10 shows the result for the dominant colors.

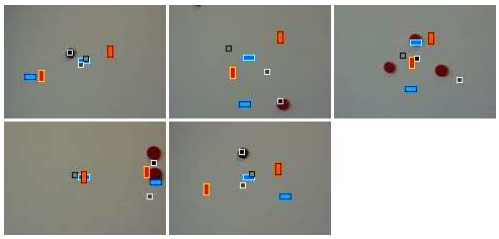

Figure 5.11: Centroid based buckets with 37 frame descriptors, with filter *morph5*.

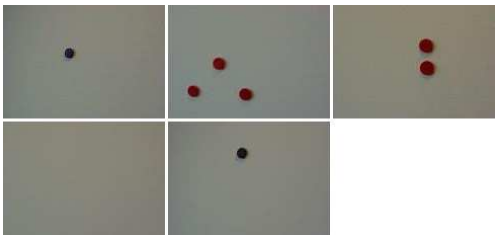

Figure 5.12: Dominant Colors, with filter *Morph5*.

order	Expected (range)	Resulting frames	
		YUV	OCCD
1	0- 51	0	0
none		70	72
2	78- 150	109	
3	214- 273		246
4	279- 298	279	297
5	310- 386	386	386

Table 5.3: Key frame number and the resulting frame numbers of the different features for the video *Mov00085*.

order	Expected (range)	Resulting frames	
		YUV	OCCD
1	0- 51	0	0
none		64	
2	78- 150	109	136
3	214- 273	273	247
4	279- 298		295
5	310- 386	386	386

Table 5.4: Key frame number and the resulting frame numbers of the different features for the video *Mov00085*.

As can be seen, the results of both kinds of frame descriptors are very bad and not acceptable. Table 5.2 shows the exact results for the comparison experiment. As we can see, the quality of the dominant color frames version is a little bit better than the centroid based features. Only the frame with two dots is not completely visible. In the centroid based key frames, the frame with three dots is available twice and the frame with two dots is completely missing. The sixth frame for the YUV features is frame number 214, which

is the missing key frame number three. The missing key frame number two of the OCCD frames appears as key frame number eight.

Figure 5.11 shows the same experiment, but this time the features are filtered with *Morph5* for the centroid based frame descriptors. Figure 5.12 shows the key frames of the dominant colors also filtered with *Morph5*. The performance quality of the centroid based features is not better. The performance of the dominant colors is optimal. Table 5.4 shows the exact results for the Morph5 filtered "Mov00085" comparison experiments. It seems that the dominant color based frame descriptors are very good for this kind of experiment.

5.4 Different image scalings

We developed histogram and centroid based frame descriptors in 4.2.1, which are invariant to image scaling. The Dominant Color frame descriptor from 4.3.1 is also scaling invariant.

The video we used for the scaling invariance test is a video from Mr. Bean, known as "Mr. Bean's Christmas". The original full-size version of the video is named "MrBeanTurkey". From this video, a smaller down-scaled version named "mrbeantu" was created. Information about both videos are available in Chapter A.

We had a loss of information due to the scaling of the video frames because pixels were changed or completely removed. This loss of information was reflected in different content and different information, even though our frame descriptors are scaling invariance. Therefore is that we could not expect identical results for a video and its down-scaled version. However for a robust key frame algorithm we expect that we got nearly the same results visually for (nearly) the same content.

Dynamic buckets

This experiment was made with a previous version of the correct scaling of the dynamic weighted X- and Y-color component. The scaling of these coordinates is $[0, \frac{1}{2}]$ instead of the developed $[0, 1]$. Nevertheless, the quality of the results was very good and nearly identical. The frame descriptors are histogram based centroids and time with 37 components. The cost function is from formulae 4.1.

Figure 5.13: Result with the best nine frames of the full-sized version of the video “Mr. Bean’s Christmas”, with unfiltered frame descriptors.

Figure 5.14: Result with the best nine frames of the down-scaled version of the video “Mr. Bean’s Christmas”, with unfiltered frame descriptors.

Figure 5.13 shows the result of the best nine key frames of the large-sized version of “Mr. Bean’s Christmas” and figure 5.14 shows the results of the smaller version. They contain the best 9 key frames for the dynamic weighted and unfiltered frame descriptors.

As we can see, the results are nearly the same. A detailed frame comparison is in Table 5.5. We also did the same experiments with filtered features. Figure 5.15 also shows the result of the best nine key frames of the full-sized version of “Mr. Bean’s Christmas” but this time the features were filtered with a morphological filter with a width of 11 (Morph5). Figure 5.16 shows the results of the down-scaled version of Mr. Bean, with Morph5 filtered.

These experiments also have nearly identical results. Frame number four is a little blurred in the fill-sized version. This frame is in the transition of two shots and the frame contains content from both shots, but we see that it also contains information as shown in the fourth key frame of the down-scaled version of the video. Table 5.5 shows detailed information about the detected key frames, the frame range in which we expected the key frames and the resulting frame numbers for the different videos and filters. The total quality

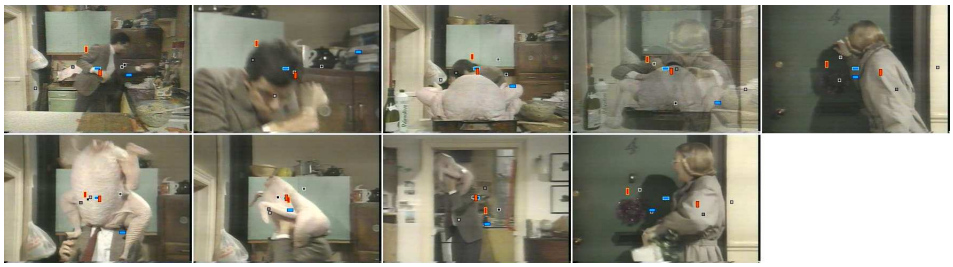

Figure 5.15: Result with the best nine frames of the full-sized version of the video "Mr. Bean's Christmas", with filtered frame descriptors.

Figure 5.16: Result with the best nine frames of the down-scaled version of the video "Mr. Bean's Christmas", with filtered frame descriptors.

of the video segmentation in relation to the expected key frames[1] is not optimal when viewed subjectively because frames of shot four are completely missing and shot seven has two key frames. However the results in relation to the scaling invariance are very good, both filtered and unfiltered. For each detected key frame in the full-sized version, there exists a counterpart in the down-scaled version and vice versa.

Dominant Colors

The frame descriptors for the following experiments are dominant colors as described in the previous chapter. The cost function is that of formulae 4.1.

Before viewing the results of the same experiments as done for the centroid based features, we will take a look at the feature vector components of the dominant colors for the different videos. Figure 5.17 shows the available dominant colors of the 99 possible code colors of the full-sized version of the Mr. Bean video. The X-component of the figure represents the time line of the video. The column of pixels most to the left is frame number 0 and

[1]These key frames are available in Appendix A

	Frame range		Frame numbers			
Features:			dynamic weighted centroids			
Filter:			None		Morph5	
Version:			Large	Small	Large	Small
Shot 1:	0-	992	0	0	0	0
Shot 2:	997-	1165			1162	1157
Shot 3:	1166-	1291	1179	1223	1207	1186
Shot 4:	1291-	1357	1346	1312		
Shot 5:	1357-	2009	1357	2008	2009	2008
Shot 6:	2009-	2079	2072	2015	2043	2073
Shot 7:	2080-	2182	2102	2101	2107	2106
			2180	2177	2179	2179
Shot 8:	2183-	2363	2184	2200	2205	2196
Shot 9:	2364-	2379	2379	2379	2379	2379

Table 5.5: Table with the frame range of the shots and the frame numbers of the resulting key frames. Created with dynamic weighted centroids.

the column most to the right is frame number 2379. The Y-component of the figure represents the different available colors. The top row of pixels is the first code book color and the bottom row of pixels of the last code book color (99). Black pixels indicate that the representing code book color in that frame was not available. As we can see, both figures nearly identical.

In some frames, some colors are available in only one of the video frames but not in the other. This is not very long, as the area of the frame is not very large or not as long as a similar other code book color available (maybe in another intensity of the color). Figure 5.17 shows the area information of used dominant colors in the same way as the available dominant colors. Black pixels indicates that the color is either not available or the representing area is very small. Brighter pixels indicate a larger availability of that color in that frame. Figure 5.18 shows the same area information for the down-scaled version of the video.

As can be seen, the areas in the different frame parts are (nearly) black, so that the misgivings are arbitrary. Figure 5.21 shows the result of the best nine key frames of the full-sized version of "Mr. Bean's Christmas" and figure 5.22 shows the results of the down-scaled version of the video. They contain the best 9 key frames for the unfiltered dominant colors frame descriptors.

Just like the results for the centroid features, these results are also similar to the expected key frames, and the results of the scaled videos are also nearly

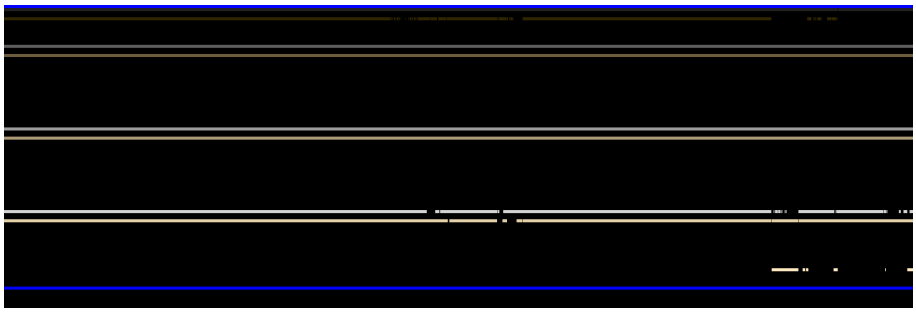

Figure 5.17: Dominant Color Availability image for the large version “Mr. Bean’s Christmas”.

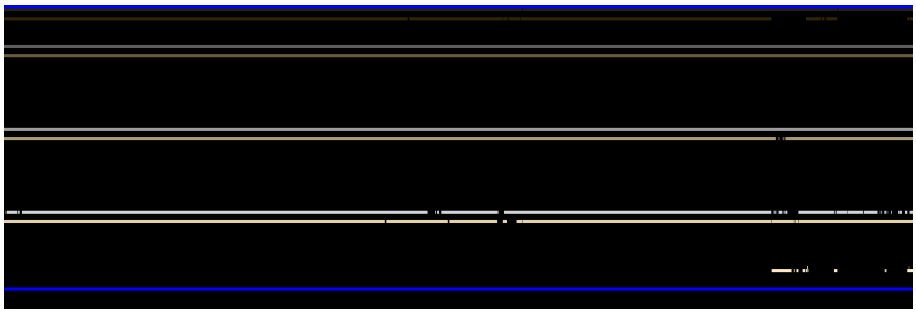

Figure 5.18: Dominant Color Availability image for the small version “Mr. Bean’s Christmas”.

the same. Figure 5.23 shows the result of the Morph5 filtered features. Figure 5.24 also shows the results of the down-scaled version with Morph5 filter.

Table 5.6 shows a direct comparison between the expected key frames for the shots and the results key frame numbers. In this case, the results are also nearly the same with an exception of the key frames for shots 7 and 8. A comparison between the centroid based features and dominant colors results shows that the results of the dominant colors are a little better in relation to the expected key frames, as shown in the appendix.

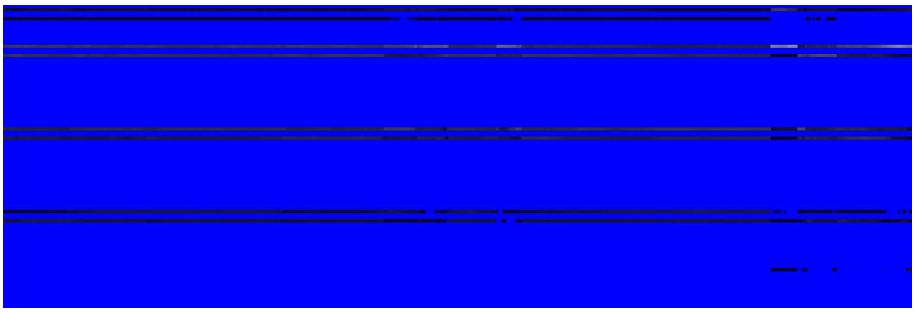

Figure 5.19: Dominant Color Intensity image for the large version "Mr. Bean's Christmas".

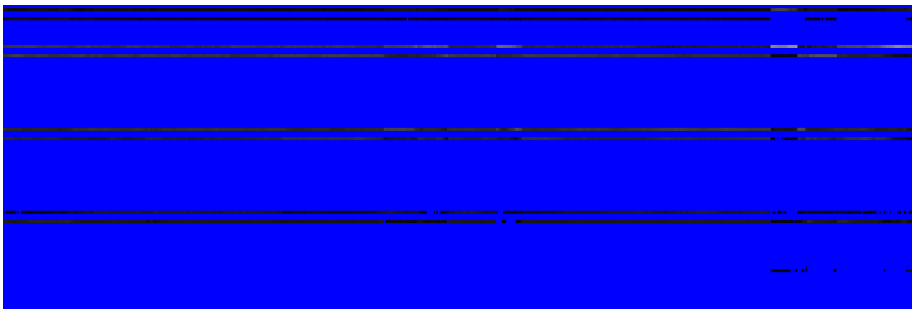

Figure 5.20: Dominant Color Intensity image for the small version "Mr. Bean's Christmas".

	Frame range		Frame numbers			
Features:			Dominant Colors			
Filter:			None		Morph5	
Version:			Large	Small	Large	Small
Shot 1:	0-	992	0	0	0	0
					613	614
Shot 2:	997-	1165	1109	1162	1154	1154
Shot 3:	1166-	1291	1225	1221	1212	1226
Shot 4:	1291-	1357	1304	1304	1302	1297
Shot 5:	1357-	2009	1785	1788	1806	1802
Shot 6:	2009-	2079	2042	2045	2041	2025
Shot 7:	2080-	2182	2105	2095		2105
Shot 8:	2183-	2363	2183	2183	2206	
Shot 9:	2364-	2379	2379	2379	2379	2379

Table 5.6: Table with the frame range of the shots and the frame numbers of resulting key frames. Created with dynamic weighted centroids and Dominant Color features.

Figure 5.21: Result with the best nine frames of the full-sized version of the video “Mr. Bean’s Christmas” with unfiltered dominant colors.

Figure 5.22: Result with the best nine frames of the down-scaled version of the video “Mr. Bean’s Christmas” with unfiltered dominant colors.

Figure 5.23: Result with the best nine frames of the full-sized version of the video “Mr. Bean’s Christmas” with filtered dominant colors.

Figure 5.24: Result with the best nine frames of the full-sized version of the video “Mr. Bean’s Christmas” with filtered dominant colors.

Chapter 6

Comparison

We have seen in Chapter 1.3 that the function of the two algorithms [31, 53] was to create a temporal segmentation of video sequences. In this chapter, we will make a comparison between the basics of the discrete curve evolution algorithm and the other two algorithms.

6.1 Algorithm comparison with Kumar et. al

The segmentation algorithm developed by Rajeev Kumar and Vijay Devatha [31] is a shot detection algorithm based on a neighbor frame to frame comparison algorithm. The used frame descriptors are based on flexible histogram bins. They assume that important content areas are in the centre of the frame. This is reflected by weighting of the pixels depending on the pixels position inside the frame. These histogram bins are filtered by a Gaussian filter to achieve a robustness of the frame descriptors. A window is moved over all the bins and inside the window the Gauss filter is applied.

The frame comparison is based on the Bhattacharrya metric, which is a generalized χ^2 measure and is defined as the sum of the dot product of all histogram bins.

An interesting idea in this manuscript is the possibility to create a "Field of Interest View" (which should depend on the application). Single moving objects on the screen will be detected due to the weighted histogram (as long as objects are not moving in the same weighted area). Unimportant information on the frame border will be filtered. Missing is the loss of regional information.

This algorithm is probably only useful to detect hard cuts as shown in figures 1 and 2 on page 10 of their manuscript. Due to the missing of a comparison in a local context, slow movements will not be detected (or poorly detected) because the frame differences could be too small to be detectable. If changes are so slow that the histogram differences are near zero, then it could be possible that these changes are not detected or are lost in the background noise. Maybe these kinds of frames are not directly the kind of (key) frames that should be detected because shot boundaries normally happen with faster transition between shots. However, this is relative and should be invariant to the frame rate. The only probability to detect such slow and small differences is the dynamic shot detection level. The variation in the algorithm is flexible enough to fulfill the expected shot detection. In addition to the previous comments, if a video frame rate is stretched by identical interframes, then these frames have an influence on the approximation curve, which also have an influence on the minima and the detected shot boundaries.

Error propagation to eliminate noise in the histogram is a good idea but the image equality algorithm is based on images at different times. The implemented filter however does not consider this.

As with every error propagation, the filter will reduce the amount of information. This could result in undetected shot boundaries without clear transitions. The only useful feature is the possibility to define different frame width for the filter window.

Comparison with the DCE

As already mentioned, an interesting idea is to define a *field of interest*, which could result (depending on the application) in better features and in better key frames. It is possible that the "MOV1" video could give better results with the usage of a "field of interest".

We also filtered our features only in the time and not in the frame. (It may be a good idea to test filtering our features also inside of the frame.) Such kinds of filters could be implemented for the buckets by filtering the content between the neighbor buckets. I think that the starting-point for the filter in the time domain is a better choice.

We considered the local context by the definition of the relevance measure. The area of the local context was raised by each removed frame. Due to this increase in the comparing area, we can also detect slow movements, which is not possible with the shown algorithm in this manuscript.

The discrete curve evolution is not designed for special features or frame comparison metrics. We have the flexibility to be able to define different kinds of frame descriptors. It would be possible to implement the features as proposed by Kumar et. al in the discrete curve evolution. What we need in that situation is an algorithm to detect the expected abstraction level. This is solved better in the manuscript by performing a dynamic analyses of the relevance curve in order to find the minimum relevance value.

The shot detection algorithm of Kumar et. al is based on a single frame to frame comparison. The discrete curve evolution has the ability to define more complex comparison functions and it would also be possible to implement the comparison metric of Kumar in the discrete curve evolution.

The discrete curve evolution seems to be more flexible than the proposed algorithm. We have the flexibility to be able to define different frame descriptors and frame comparison functions. Our implemented frame descriptors are also based on the histogram but we extract on the other hand more information than the proposed frame descriptors. However, any other kind of information is not used.

6.2 Algorithm comparison with Zhu et. al.

Xingquan Zhu, Jianping Fan, Ahmed K. Elmagarmid and Xindong Wu described in their article, "Hierarchical video content description and summarization using unified semantic and visual similarity", algorithms for video summaries [53].

The idea to predefine different abstraction layers is a good idea and it's easier to work towards developing algorithms that will implement these abstractions. The disadvantage is that it may not be flexible enough for some kinds of videos. All work is based on shots depending fundamentally on its detection algorithm. The distance metric will only detect fast frame changes. The gradual transition algorithm also only detects one transition between two cuts.

The detected shots are merged into the different kinds of abstraction levels. The merging process uses different kinds of comparison algorithms between the different groups of shots in the different abstraction levels. Finally they are based on frame comparison and dynamic generated thresholds. The frame comparison is based on other frame descriptors as used for the shot detection itself.

Another problem, but also an advantage too, is the user interaction to define keywords for shot groups. An user interaction is not always possible and in those cases only the non-supervised part of the group algorithm is suitable (and comparable).

It is a good idea to use different algorithms for different abstraction levels which are based on the next lower abstraction level.

Comparison with the DCE

A direct one to one comparison is not possible due to the specialized functionality of the proposed algorithm. In the DCE we do not have predefined, only non-specific abstraction levels. Due to our algorithm, it could also be possible to implement shot grouping algorithms based on our detected key frames, the location and the relevance.

Comparison of the algorithm features:

Frame descriptors
The used frame comparison algorithm is based on two kinds of features. It has a histogram-based part of features and a texture-based part. I think our histogram-based features are better for us because we are then able to detect the position of the objects. The advantage of the proposed algorithm is the ability to use a more content-based descriptor with coarseness texture.

Low-level key frame detection
The used shot detection algorithm is not the simplest algorithm because it also detects gradual transitions. However this algorithm could fail if too many gradual transitions are used or if the transitions are either too slow or non-existent. (Like a long camera move across the horizon with different content.) Also the selected key frame (which is important for the later analysis) is more a random than a content dependent selection. I think that our algorithm will detect such shots better and the key frames will be safer because the temporal distance of the shots has no influence on our key frame selection process, and this selection is based on the conspicuity of a key frame in relation to the neighbor key frames. (This "conspicuity" feature between frames is used for the higher detection algorithms.) The problem of the DCE is that we have no usable (static or dynamic) threshold to detect key frames (or shots).

High level key frame detection
The advantage of the DCE is that it builds a hierarchical tree of the frames in the order of the abstraction levels. Due to this fact, the DCE could group shots (key frames) to a higher abstraction like shot groups. Therefore,

different strategies[1] exist which are not further pursued, so a comparison at this level with these algorithms is not possible. This was not our intention anyway because more information about the content and the application of the resulted information is necessary. Nevertheless, the DCE sometimes produces a result which looks like the grouping algorithm[2]. Even if the DCE is not directly suitable for higher detection algorithms, it contains much more potential than only for detecting (shots) key frames.

6.3 Experimental comparison with David Rossiter et. al.

David Rossiter has (on his Homepage [48]) three videos with his own key frame creation perl scripts. The used videos are a short sequence from the movie, "The blade runner", a tour through a house and a presentation of Kylie Minogue. The videos can be found on his homepage.

In Appendix A, the videos and ground truth results are presented.

6.4 Experimental comparison with Drew

Drew et. al have developed an efficient clustering method [21] to create key frames for video segmentation. They created several ground truth experiments. A group of people created representative key frames for these videos as a ground truth result set. As discussed in Chapter 3, such result sets are not always comparable with other result sets created by other methods, due to the background information that is expected as a key frame.

These ground truth key frames are compared with the results created by their clustering algorithm. These videos are very useful to test the performance of the *discrete curve evolution* with a ground truth result set. This was artificially created by humans and is the result set of another segmentation algorithm.

[1] It could be possible to join lower abstracted key frames to higher key frames by analyzing the key frame intervals, the neighbor key frames and the relevance values.

[2] At a specific *but lower* abstraction level in the DCE, shots in a "shot group" are successively removed until one or two frames are left. These remaining frames of this "group" could be interpreted as key frames for this group.

Figure 6.1 shows the expected ground truth key frame result set. The four frames are from the video *beachmov* which is a short video clip of approx. 30 seconds with four shots. Shot one is a larger shot taken with a left-right-left pan over a beach with water, a city in the background and a forrest. This shot blends over into shot two showing a beach volleyball game. With another blending, this shot is transformed into shot showing a few people in a swimming pool. This shot then moves with a blending into shot four which shows the edge of a swimming pool. Figure 6.2 shows the result of Drew's clustering algorithm for this video. The frames two and three are the same as in the expected result and frame four is missing but the first frame seems to be completely different. Both frames are from the first shot which is alright. However, as we described before, different people expect different results for the same video (or shot) if the shot contains different content. Figure 6.3 shows our result which also includes both frames of the first shot.

Figure 6.1: Expected ground truth results for video "beachmov"

Figure 6.2: Drew's key frame result for video "beachmov"

Figure 6.3: Our result with the *DCE* for video "beachmov"

This example shows not only the general problem of defining which frames are key frames but also the problem of how many frames should be left in a resulting key frame set. Despite these problems, our results show that the algorithm gives good results for a temporal video segmentation algorithm that not only detect shots. It seems logical, at this abstraction stage, that the great content difference between those first frames is reason enough to

present two frames for a video abstract.

The complete video set contains 14 videos which can be found on the original website of Drew [21]. All comparison between Drew's clustering algorithm and the *discrete curve evolution* is available on *http://www.videokeyframes.de/Information/Doc/GlobalResults.pdf* [11].

Figure 6.4 shows an overall comparison of Drew's and our results for the videos used by Drew. The precision is used to measure the quantity of correct detected key frames, and the recall measures the quantity of false detected key frames.

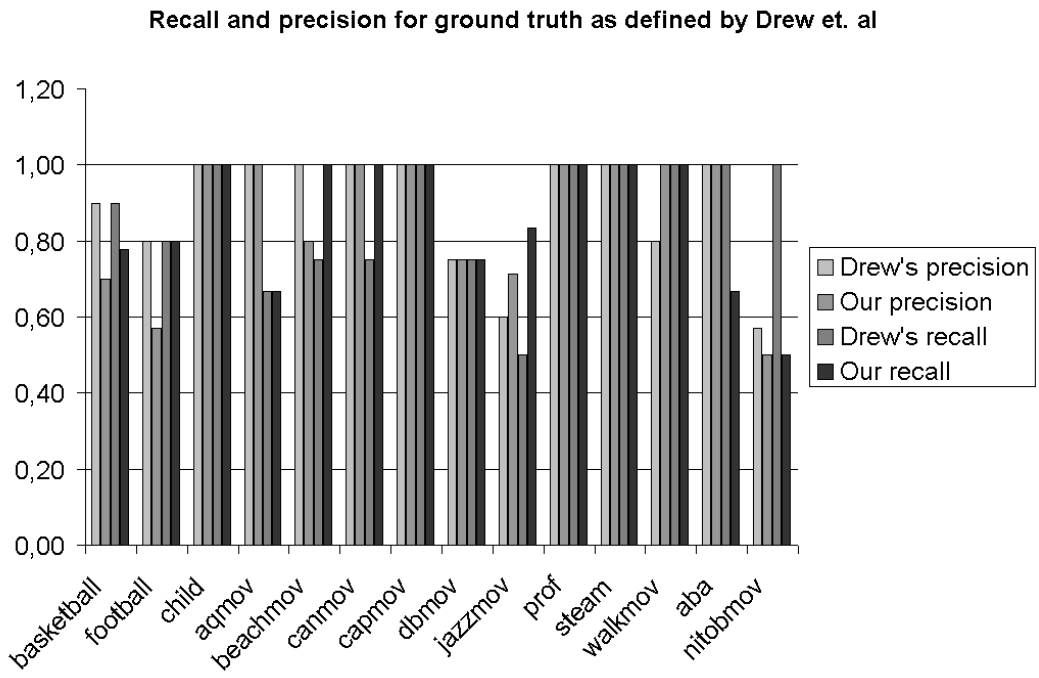

Figure 6.4: Precision and recall for Drew's and our results

Chapter 7

Video stream

In this chapter we will discuss the applicability of the discrete curve evolution on video streams. Video streams are presented for example by video cameras and are used for video monitoring or in video surveillance applications. In comparison with closed videos, the difference from a data technical point of view is the open characteristic of the video material. The video stream doesn't have normally a fix defined start or end frame. An analysis like we have done for closed videos is not possible. The second difference is the aim of the application for which the video was analyzed.

The target is to detect any unpredictable changes in the scenario like the appearance or disappearance of persons and objects [45] or any other change in the scenario.

This kind of detection is not trivial because there are many non-static parts in a video stream, thus making it difficult to clearly detect a change. In outdoor applications, many environmental features exist which are an important influence on the appearance of the video content. The wind moves trees, sun and clouds create light and shadow etc. The camera itself has also an important influence on the content. Where is the camera mounted? Is it statically or dynamically motor driven? Which direction is recorded? Is a zoom being used? All changes in these features will change the content of the same scenario.

It is nearly impossible to create a detection algorithm that is flexible enough to detect events in a scenario for any possible environmental situation. Normally a frame-to-frame based algorithm is used for the detection events but this depends on the environmental noise and due to the great amount of external influence, is this not trivial.

Our aim is to use the discrete curve evolution as a flexible detection algorithm for unpredictable events in data streams.

Open video streams are found for example in real time video streams, which do not have a well-defined start and end frame. An example for the use of such kinds of video streams could be in surveillance videos in order to observe areas, entrances and to detect access to these. Security personnel could be automatically alarmed if something or someone enters a restricted area.

Video streams are also used in quality assurance and in manufacturing processes to observe the texture of a product. Video stream data analysis is also used in the quality control of surfaces like rails and streets.

We have seen that our algorithm is applicable in closed videos with a predefined start and end frame. Video streams from live cameras have no predefined (start and) end frame. Our algorithm would be very useful in finding key frames in the motivation examples above. The problem we had at this point was that those videos did not have a well-defined start and end frame.

How can we use the algorithm to analyze such (infinite) video streams?

7.1 Window analysis

The solution is to make a local analysis of a video stream on a connected subset of frames. The best key frame of this subset is stored and it's relevance value is drawn over the time in which the key frame appears. The video stream is fragmented in consecutive windows with a frame subset and each of these subsets is analysed for potential key frames.

The relevance curve of the potential key frames is analysed and important changes inside the curve are detected as key frames.

Figure 7.1 shows such a relevance curve for the video "Mov3". In figure 7.2, the three frames at the local maxima of figure 7.1 are shown.

The analysis of video on a subset is done with the same algorithm that we used for the analysis of closed videos with a fixed start and end frame. The fragmentation is done at the higher feature level instead of the lower video level. This may be done because the frame descriptor creation and filtering applications are local operations on the video stream which don't need global information about the whole video. The advantage is a minimum amount of changes in the existing applications, so that the frame descriptor extraction and also the filter applications can be reused without any limitations. With

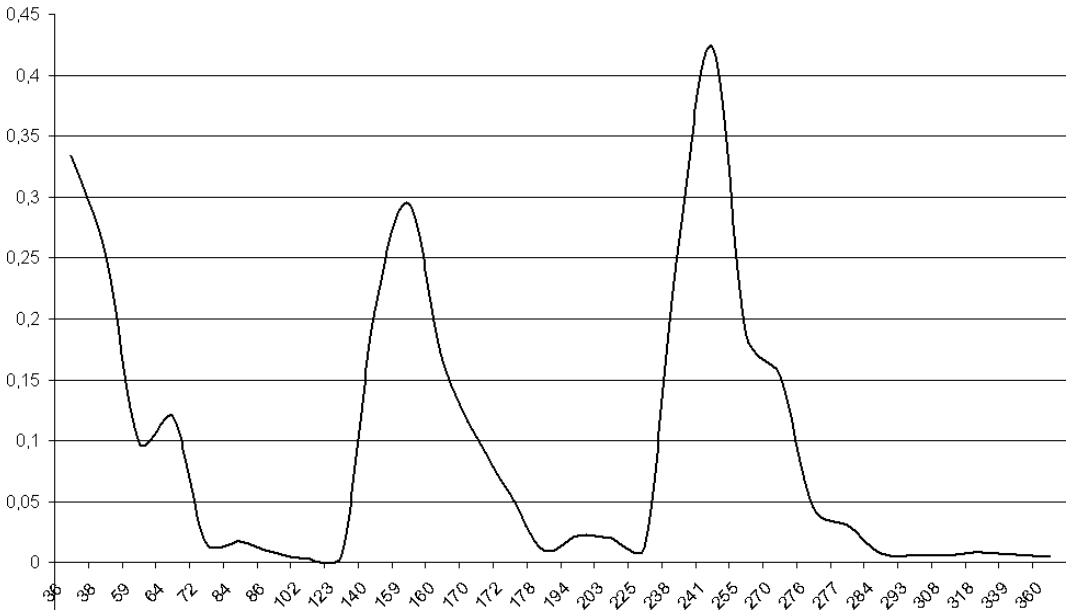

Figure 7.1: A relevance Curve of "Mov3"

Figure 7.2: Frames at the local maxima of figure 7.1

the (un)filtered frame descriptors, a new fragmentation application creates from the requested subset a new subset to which the discrete curve evolution is applied. With the result of the DCE, a relevance value for the window is calculated and associated to the most important key frame of the window. This pair of window/frame values and the relevance values are merged with the previous results into a two dimensional polygon line. A local analysis of the polygon line gives us the ability to detect events inside the video stream.

Why not define the window relevance to the relevance value of the most important key frame inside the window? The window relevance was introduced because it has the ability to adapt the importance of the *whole* video content in a window to our requirements. A window in which more than one event happens could be more important than a window in which only one event appears. The usage of a single frame relevance value will show the availability of at least a single event. More than one important key fwhich is based

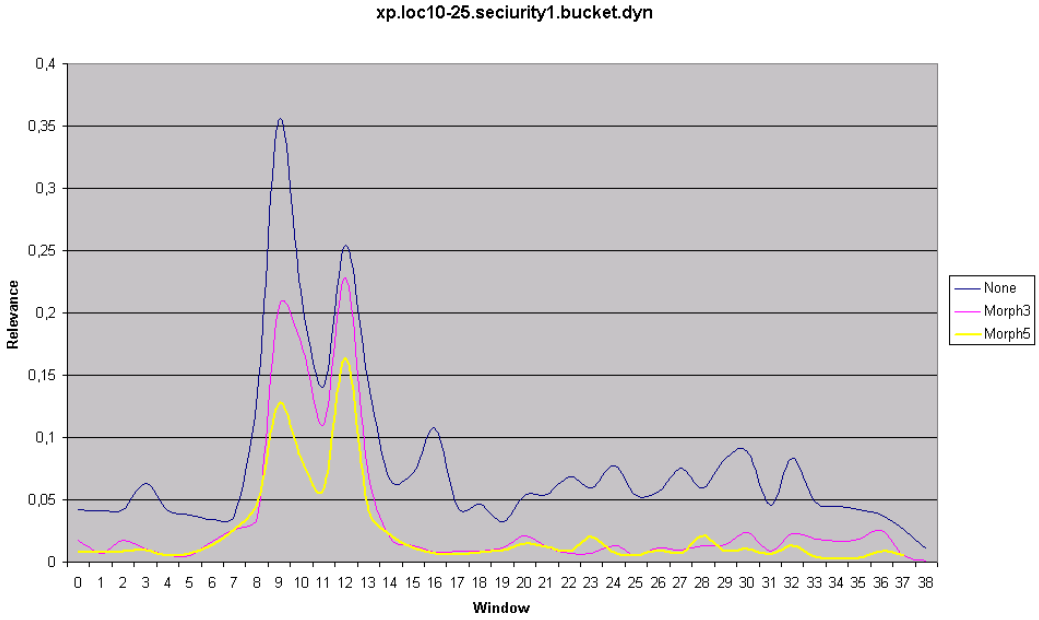

Figure 7.3: Example with relevance values as produced by our local curve evolution process. The video used is "security1"

on a single frame.

The application was modified in such a way that it wass possible to define a start and end frame. We used scripts to split the features into small parts. These same scripts also joined the results of the different curve evolution application running into a single file, which could be analyzed or visually represented like it is done in figure 7.3.

The easiest way to find the optimal key frame inside a window would be to simply search each frame for the frame with the best relevance value for the given cost function.

Our intent is not only to find the best key frame over the time but also to detect events. It is necessary to measure the quality of windows and in order to fulfill this requirement, it is useful to detect important windows rather than only a single frame inside the window. This frame becomes an important indicator of the event. Also a list of best key frames from inside the window or neighbor windows is possible.

A *window relevance* is a relevance value which depends on the frames inside the window.

7.2 Window position

The analyzed subsets of the video stream are called "windows". The first question here is how we should define the width and the location of the window.

Let us take a look at what kind of results we could expect if we change the width of the window.

New window starts after previous window

The first window was started with the first available frame of the video stream. The best key frame of this window was selected and the relevance of this key frame was stored. The next windows were started with the second-last frame of the previous window. So each frame of the video stream could be a key frame. The first and last frame of an analyzed sequence are not key frames of the analysis but they are defined as key frames for the global video key frame extraction. For the local analysis this is not desired because we only want real calculated key frames. Table 7.2 shows this window positioning algorithm.

	0 1 123456789012
window 1	sfffffe.....
window 2	.....sfffffe

Table 7.1: New window starts after previous window.

"s" means the start frame of the window (which is excluded from the possible key frames in the window).
"e" means the end frame of the window (which is excluded from the possible key frames in the window).
"f" means the detectable frames in the window.
"k" means the best key frame in the window.
"." are all other frames outside the window.

Advantage:
We have a key frame in each window.

Disadvantage:
There is exactly one key frame in each window, even if there is more than one

good key frame in that window. Table 7.2 shows the problem. If in the first window, frame 3 is detected as the best key frame, then the next possible key frame starts in window 2 at frame 7, even if one or more of the frame numbers 4, 5 or 6 are better frames (in the sense of their relevance values).

	0 1 123456789012
window 1	sf**k**fffe.....
window 2	.....s**k**fffe

Table 7.2: Not all potential key frames could be detected.

New window starts inside the previous window

The first window was started with the first available frame of the video stream. The best key frame of this window was selected and the relevance of this key frame was stored. The next windows were started in the middle the previous window. So each frame of the video stream has two possibilities at being a key frame. This is only an example of each new window position which starts at a static position inside the previous window. Table 7.2 shows this window positioning algorithm.

	0 1 123456789012
window 1	sfffffe.....
window 2	...sfffffe..

Table 7.3: New window starts inside the previous window.

Advantage:
We have a key frame in each window. There could be more than one good key frame in each window, which is detected by the next window.

Disadvantage:
The following key frames from different windows are not necessarily ordered in the time. More time is spent to perform an analysis of the same number of frames. Table 7.2 shows this window positioning problem.

	0 1 1234567890
window 1	**sffffke**...
window 2	...**skfffe**

Table 7.4: Wrong order of the key frames.

New window starts at the previous key frame

The first window was started with the first available frame of the video stream. The best key frame of this window was selected and the relevance of this key frame was stored. The next windows were started with the key frame of the window before. So it is guaranteed that after a key frame the next best possible frame becomes a key frame. Table 7.2 shows this window positioning algorithm.

	0 1 1234567890
window 1	**sfkfffe**...
window 2	..**skfffe**.
window 3	...**sffkffe**

Table 7.5: New window starts after the key frame of the previous window.

Advantage:
We have a key frame in each window. There could be more than one good key frame in each window. Subsequent key frames are in the order of their appearance.

If an important event happens that is distributed over more than one window, the key frames then occur in a short time period. Frames that are near to (or even closed into) this event are more often analysed and also become key frames. For example, in table 7.2, the key frame in window 3 was used in the analysis of 3 windows and had 3 changes in order to become a key frame.

This selection of the window position is the most useful.

Table 7.6 shows which experiments are made with dynamic and static window positioning for a window width of 25 frames. The increase in the step width from 1 to "n" frames will decrease the analysing data volume to $\frac{1}{n}$ because we only take every n-th value. Window curve diagrams also have the same value domain.

Step width	Window width					
	10	25	30	45	60	90
1		+				
5		+				
10		+				
15						
30						
45						
dyn		+				

Table 7.6: The "+" shows which step width/window width were made with "security1" and "security7" for the step width tests.

7.3 Window width

The width of the window reflects the area in which an important event is expected. If an event occurs over a few windows, then it is expected that the maximum relevance value of these window key frames is lower than that of a key frame where the event occurs in one window.

If a window is too wide, then it's possible that more than one event could be in the frame but only one could be detected. It is possible to lower this risk by setting the window more intelligently. The step width should be less than or equal to the window width, otherwise some frames are not in the window; therefore, resulting in a not-detected key frame. A step width which is too small could lead to double-detected key frames and/or key frames directly before an already detected key frame. A "too small" step width could decrease the performance of the algorithm. This is discussed in the next sub-sections.

The best window width would be a variable width that depends on the maximum relevance of the previous key frames.

Table 7.7 shows which combinations between static window step width and window width we used for our experiments. The width of the window for the dynamic step width is also given.

Window width based on a time window

It is not possible to define a fixed width that will match all scenarios. We uses a width of approximately 2 seconds. Most of the sequences use a frame

Step width	Window width					
	10	25	30	45	60	90
1		+				
5	+	+	+	+		
10		+	+	+	+	
15			+	+	+	+
30					+	+
45						+
dyn		+				

Table 7.7: The "+" shows which step width/window width experiments we made.

rate of approx. 25 fps resulting in a window of 50 frames.

Window width depends on the position of the previous key frame

We could define the width of a window as the $2+n$ multiple of the distance of the previous key frame to its window boundary (with $n > 0$). It's expected that if nothing important happens, the best key frame is somewhere in the middle of a window. In that case, we expect that also in the next window nothing important will happen, so we could increase the window width (n is the increasing factor). If something important happens, the key frame will be near the edge of a window. The next window width will be smaller. A starting, a minimal and a maximal window width should be defined too.

Window width depends on the relevance of the previous key frame

If the relevance value of the key frame is increasing (relative to the previous key frame), then it is possible that an important event will happen. It makes sense to decrease the window width in such cases. On the other hand, it makes sense to increase the window width if the relevance value is decreasing (or is nearly constant). A starting, a minimal and a maximal window width should be defined too.

Experiments

Table 7.8 shows the available experiments with a given window step width (this is discussed later) and with a different static window width.

Step width	Window width					
	10	25	30	45	60	90
1						
5		+	+	+		
10						
15						
30						
45						
dyn						

Table 7.8: The "+" shows which step width/window width were made with "security1" and "security7" for the window width tests.

Due to a larger window width, the focus will be longer on really important key frames and these will be detected "earlier". Less important key frames between two important key frames will also be skipped.

7.4 Window relevance

The next analysis is not based on a single frame of a window but on the whole window. The resulting window-relevance curve is the important feature that will be analysed. The idea is to assign the window not the relevance value of the last frame but a value which is based on a couple of frames of the window.

We call this the *window relevance* measure M^{win}. The window relevance is defined at time t for a window width w and depends on the frame relevances

Definition: polygon window
Let $P = (v_0, ..., v_n)$, a (not necessary endless) polygon. A *polygon window* of width $w \in \mathbb{N}$ at time $t \in \mathbb{N}_0$ is a polygon $P_{win(w,t)} \subset P$, defined by

$$P_{win(w,t)} = (v_{t+0}, ..., v_{t+w-1}) \tag{7.1}$$

Definition: window relevance
Let $P = (v_0, ..., v_n)$, a (not necessary endless) polygon, and $P_{win(w,t)}$, a polygon window. The window relevance $\mathcal{W} : P_{win(w,t)} \rightarrow \mathbb{R}$ of the polygon window $P_{win(w,t)}$ depends on the results of the discrete curve evolution $\wp = (P^0_{win(w,t)}, ..., P^m_{win(w,t)})$ applied to the window polygon, where $\mathcal{W}(P_{win(w,t_2)}) < \mathcal{W}(P_{win(w,t_1)}$ if the events in polygon window $P_{win(w,t_2)}$ are less important than the events in polygon window $P_{win(w,t_1)}$.

The concequence is that either a specific implementation $\mathcal{W}$ matches a specific definition of *important events*, or the implementation itself implies the definition of *important events*.

We have for example used (for our following experiments) the sum of the most relevant C vertices of the polygon window. These vertices are the last C vertices removed in the discrete curve evolution process.

$$\begin{aligned} &\mathcal{W}(P_{win(w,t)}) := \sum_{i=m-C}^{m} \mathcal{C}(v_j, P_{loc}) \\ &\text{where } v_j = P^i|_{P^{i+1}} \in P^i \\ &\text{and } P_{loc} = Loc_c(v_j) \subset P^i \end{aligned} \tag{7.2}$$

The advantage of this window relevance is that information about more than one event could be contained in the value of the window relevance. It can be expected that each (by the DCE) detectable event in a window will contain a vertex with a *high* relevance value at the time that the vertex is removed. The value of C should be selected in such a way that there is at least the number of expected events in the window.

$$C \geq \text{ maximum number of expected events in a window} \tag{7.3}$$

7.5 Event detection threshold

Key frames in the "local context" of open scenes are defined by maxima in the window relevance curve. Not every maxima should automatically be a key frame. It makes sense to define a tolerance or detection level that must be reached by a relevance before it is accepted as a key frame. Such a level could be predefined and static. It also could be dynamically calculated depending on the previous level.

The detection depends on the application as also on the video source and the window step width and size itself. As we have seen, a filter will result in

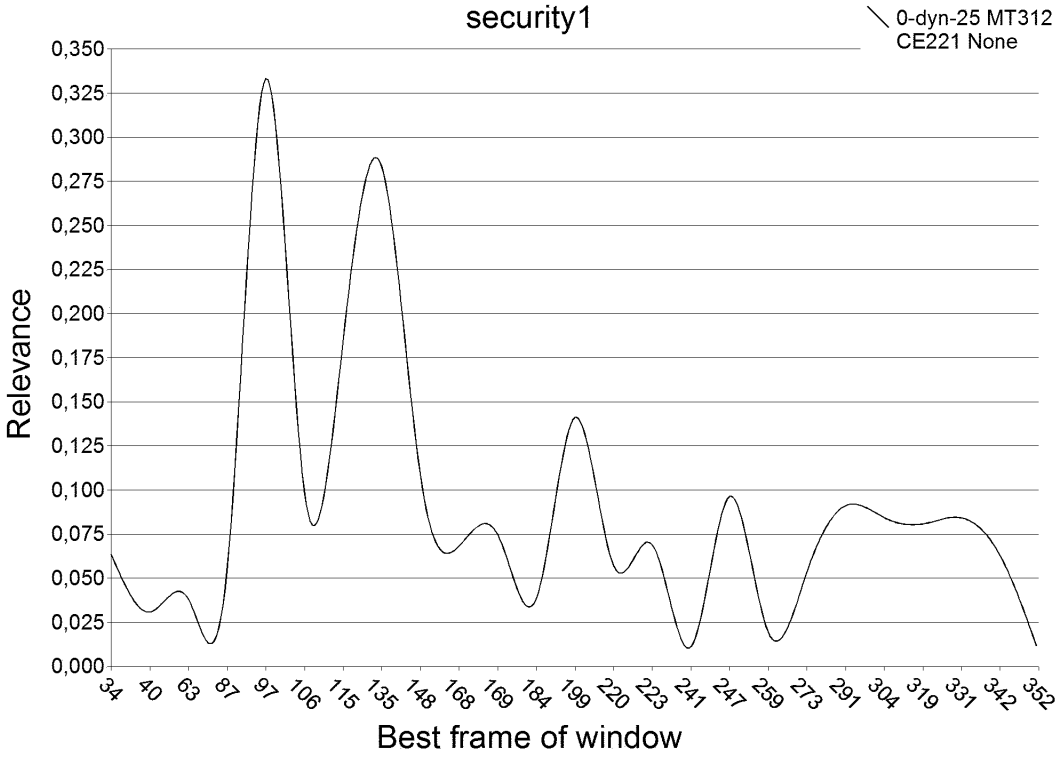

Figure 7.4: Video "security1". Dynamic weighted centroids with 37 frame descriptors, no filtering. Window position starts at the best key frame of the previous window. The window relevance is the relevance of the best key frame in the window.

smaller relevance values but also in a lower noise level, so that a static level of the relevance value could make sense, but a dynamic solution would be preferred. A video with a higher background noise will make too many false positive detections.

Much local maxima could appear, but which of these are *really* important?

Figure 7.4 shows the window relevance features for the data of video security1 with 37 features, no filtering. The window relevance is that of the best key frame in the window. The local maxima, which is interesting to us, are the frames 97 (relevance of 0.33) and 135 (relevance of 0.28). All other extrema is noise, with a relevance between 0.01 (frame 241) and 0.14 (frame 199). This results in a worse ratio of 2.01 between good and bad frames.

Table 7.9 shows the raw data of the curve evolution. The data is described in Appendix B.3. Column one contains the window number, column two is the window relevance and column three is the best key frame number of the

video which is inside the window of column one.

0 0.063478 34	13 0.057065 220
1 0.030814 40	14 0.068978 223
2 0.038028 63	15 0.011096 241
3 0.055547 87	16 0.096287 247
4 0.333187 97	17 0.019384 259
5 0.098694 106	18 0.053408 273
6 0.184939 115	19 0.091015 291
7 0.283159 135	20 0.084368 304
8 0.108204 148	21 0.080704 319
9 0.068149 168	22 0.084167 331
10 0.074726 169	23 0.062894 342
11 0.038537 184	24 0.007884 352
12 0.141034 199	

Table 7.9: Data created by the discrete curve evolution

Figure 7.5 shows the window relevance features for the data of video "security1" with 37 features, no filtering. The window relevance is the sum of the relevances of the three best key frames. This curve evolution assigns the sum of the relevances of the last three frames to the last frame. The peaks are frames 97 (relevance of 0.52) and 135 (relevance of 0.46). All other extrema isre noise with a relevance between 0.08 (frame 184) and 0.20 (frame 304). This results in a worse ratio of 2.31 between good and bad frames.

Figure 7.6 shows the window relevance features for the data of video "security1" with 37 features, no filtering. The window relevance is defined as the maximum of all frames in the window. The maxima is frames 97 (relevance of 0.33) and 135 (relevance of 0.28). All other extrema is noise with a relevance between 0.04 (frame 259) and 0.14 (frame 199). This results in a worse ratio of 2.01 between good and bad frames.

The following result is that of the same experiment before with the exception, that the best key frame of the window is defined as the key frame at which the maximum relevance appears. This is not necessarily the last key frame. Figure 7.7 shows the window relevance features for the data of video security1 with 37 features, no filtering and Curve Evolution 2.28. The peaks are frames 97 (relevance of 0.33) and 135 (relevance of 0.28). All other extrema is noise with a relevance between 0.04 (frame 258) and 0.14 (frame 199). This results in a worse ratio of 2.01 between good and bad frames.

It is not trivial to define a detection algorithm for depending on the win-

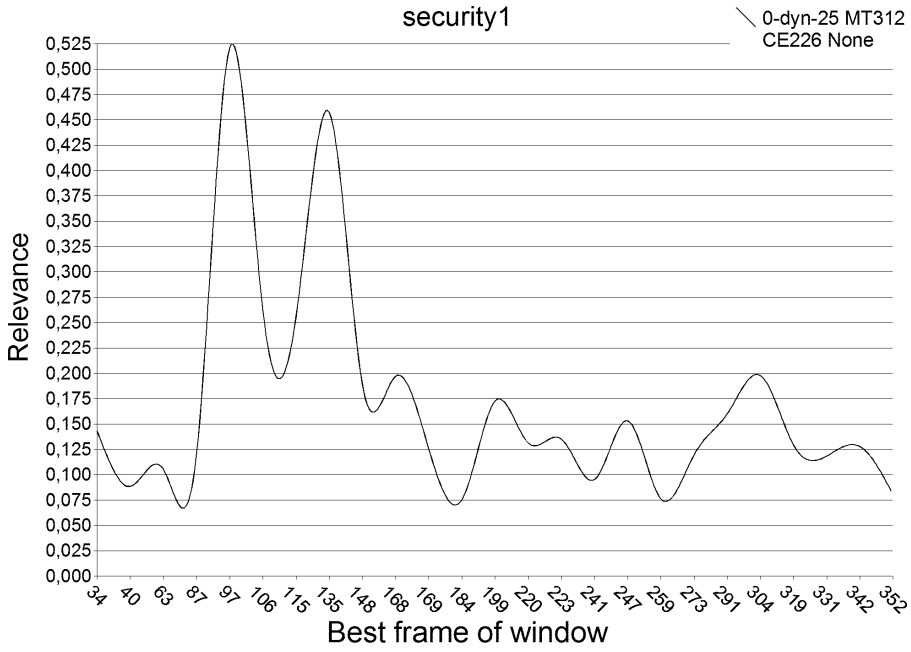

Figure 7.5: Video "security1". Dynamic weighted centroids with 37 frame descriptors, no filtering. Window position starts at the best key frame of the previous window. Window relevance is the sum of the relevances of the three best key frames.

dow relevance levels to detect important windows. The following algorithms are examples of how such a detection algorithm could be implemented for different kinds of situations.

First algorithm

The idea is to define local maxima in the window relevance curve as a window with important events. If the previous relevance was higher and the relevance before that was lower, then we have a local maximum.

Algorithm:

$\mathcal{W}(P_{win(w,t-2)}) < \mathcal{W}(P_{win(w,t-1)}) \wedge \mathcal{W}(P_{win(w,t-1)}) \geq \mathcal{W}(P_{win(w,t)})$
$\Rightarrow P_{win(w,t-1)}$ has at least one important event

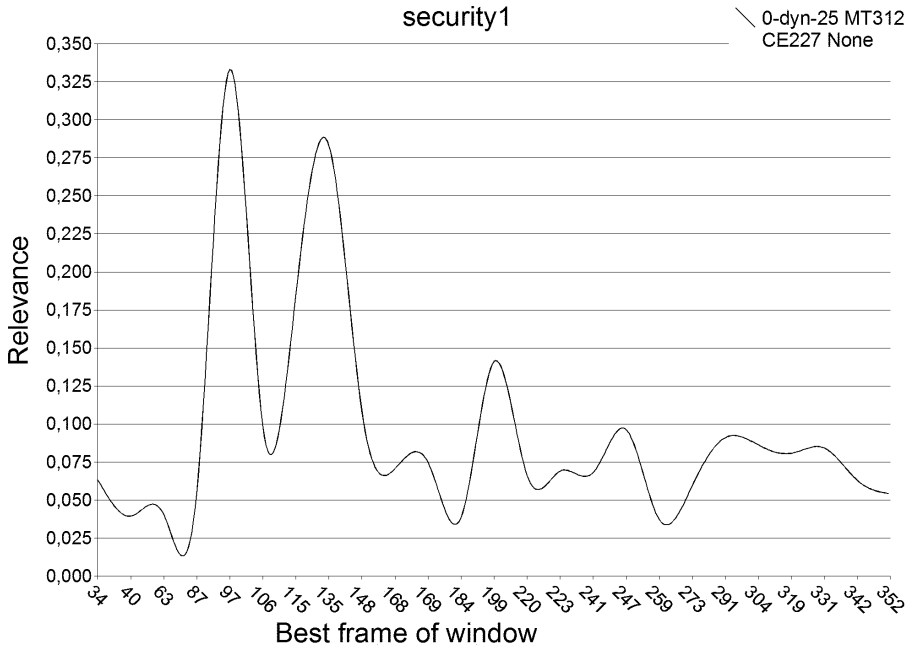

Figure 7.6: Video "security1". Dynamic weighted centroids with 37 frame descriptors, no filtering. Window position starts at the best key frame of the previous window. Window relevance is the maximum of the key relevances.

Second algorithm

This algorithm is a refinement of the previous event detection algorithm. This algorithm could used when it is expected that the window relevance level should have a minimum value. The idea is to define a static threshold T_{const}, thus determining that a window is important.

Algorithm:

$\mathcal{W}(P_{win(w,t-2)}) < \mathcal{W}(P_{win(w,t-1)}) \wedge$
$\mathcal{W}(P_{win(w,t-1)}) \geq \mathcal{W}(P_{win(w,t)}) \wedge$
$\mathcal{W}(P_{win(w,t-1)}) \geq T_{const}$
$\Rightarrow P_{win(w,t-1)}$ has at least one important event

Third algorithm

This detection algorithm is the same as the previous algorithm, however the thresholds T_{dyn} are dynamically calculated for each window, depending on the previous windows. This could be useful in cases for which it is not

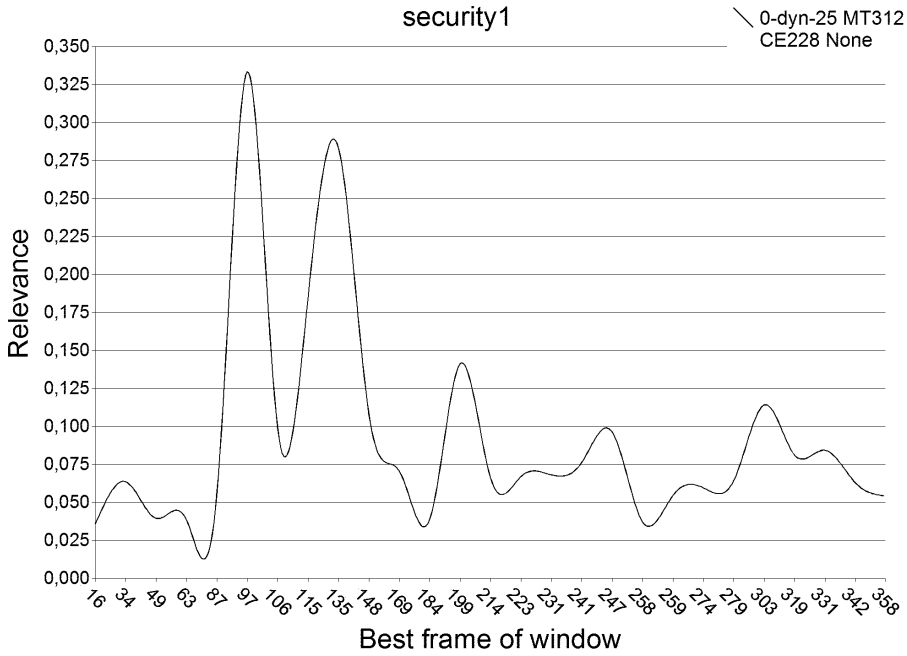

Figure 7.7: Video "security1". Dynamic weighted centroids with 37 frame descriptors, no filtering. Window position starts at the key frame with the best relevance value of the previous window. Window relevance is the maximum of the key relevances.

possible to define a static threshold. This is for example useful in situations when the filter width is changed, which could result in lower relevance values of the cost function, thus influencing the window relevance values.

$\mathcal{W}(P_{win(w,t-2)}) < \mathcal{W}(P_{win(w,t-1)}) \wedge$
$\mathcal{W}(P_{win(w,t-1)}) \geq \mathcal{W}(P_{win(w,t)}) \wedge$
$\mathcal{W}(P_{win(w,t-1)}) \geq T_{dyn}$
$\Rightarrow P_{win(w,t-1)}$ has at least one important event

Fourth algorithm

Another idea is to define the window relevance in such a way that each value above a dynamic or static threshold T is defined as important.

$\mathcal{W}(P_{win(w,t-1)}) \geq T$
$\Rightarrow P_{win(w,t-1)}$ has important events

7.6 Filtering

The following figures show the usage of different morphological filters applied to the video *security1*. As can be seen there is no difference between the *non-*filtered and the *Morph1* filtered features. The source and the filtered features of the digital camera are identical. It seems that the digital camera triples the recorded frame to get the necessary video frame rate. Figures 7.9, 7.10, 7.11, 7.12, 7.13 and 7.14 show the influence of morphological filters with different filter window widths. The range of the relevance is reduced to lower levels. Different details of the curve disappear, but as we can see in figures 7.10 and 7.11, some details are also not reduced, which results in a raise of the importance at these places (around window number 199).

Figure 7.8 shows the influence of different morphological filter widths on the relevance level of the windows. The used video is "security1" (Appendix A.1.2), with a window width of 25 frames and a static repositioned window. The repositioning step width of the window is 10 frames.

The missing differences in figures 7.9 and 7.10 are the result of the tripled frame[1]. The video consists of blocks with three successive identical frames. A morphological filter with the same width (or less) as the width of these frame blocks will not change anything.

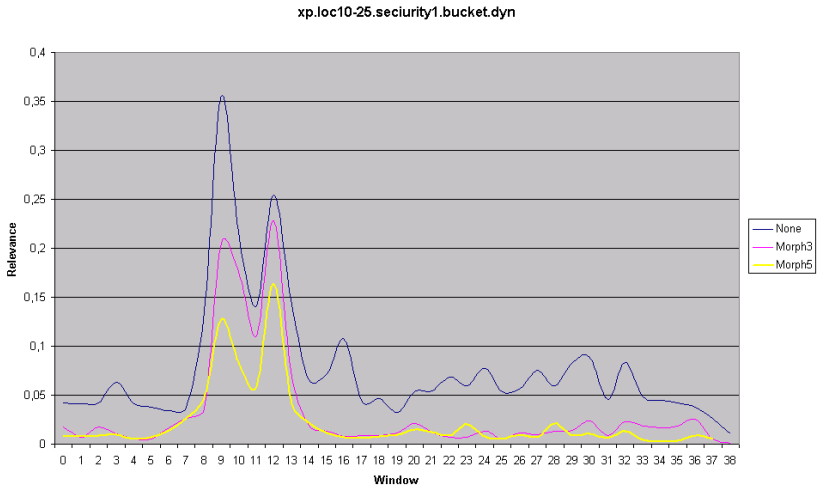

Figure 7.8: Relevance curve with different morphological filter width.

[1]As described in the appendix, a new frame is followed by two identical frames. This is a recording feature of the used hand-held camera.

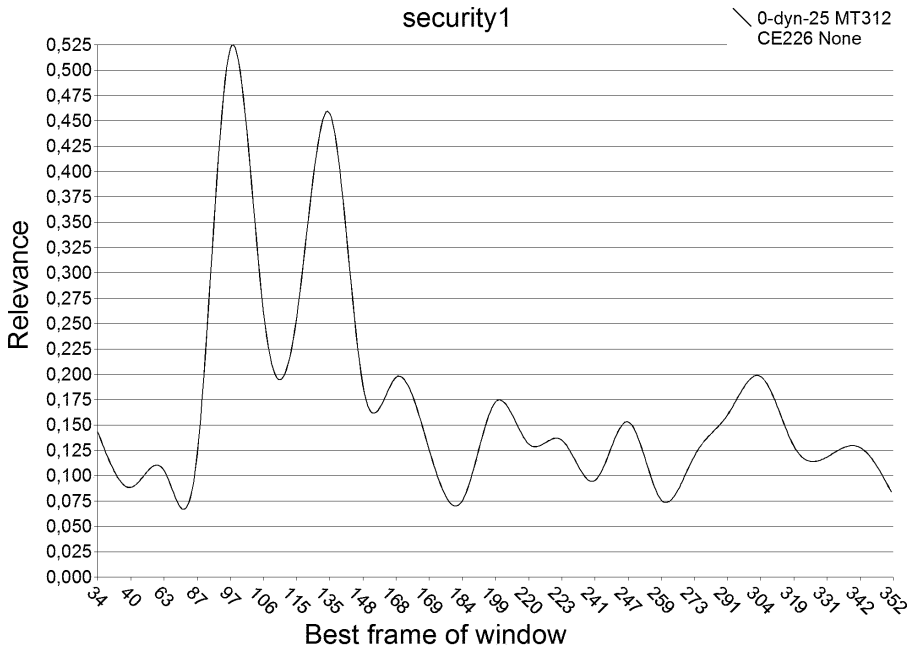

Figure 7.9: *Non*-filtered video "security1" with 73 features, and window relevance defined as the sum of the relevances of the three best key frames.

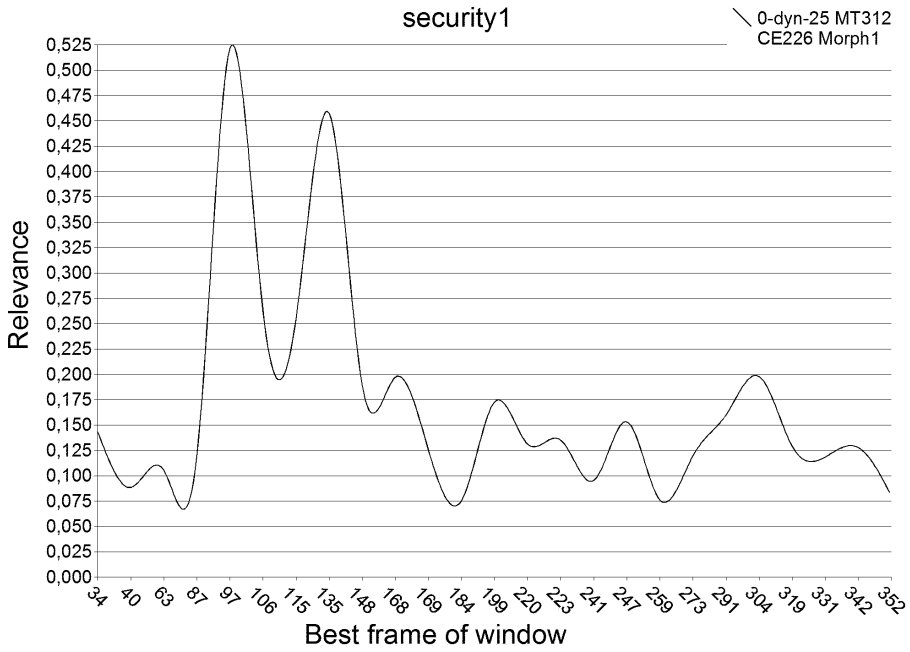

Figure 7.10: *Morph1* filtered video "security1" with 73 features, and window relevance defined as the sum of the relevances of the three best key frames.

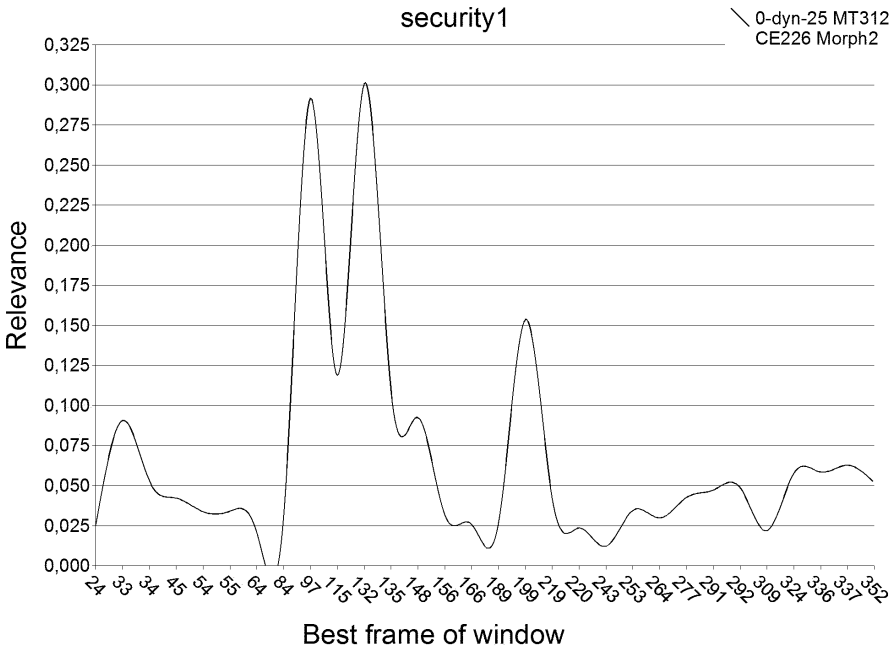

Figure 7.11: *Morph2* filtered video "security1" with 73 features, and window relevance defined as the sum of the relevances of the three best key frames.

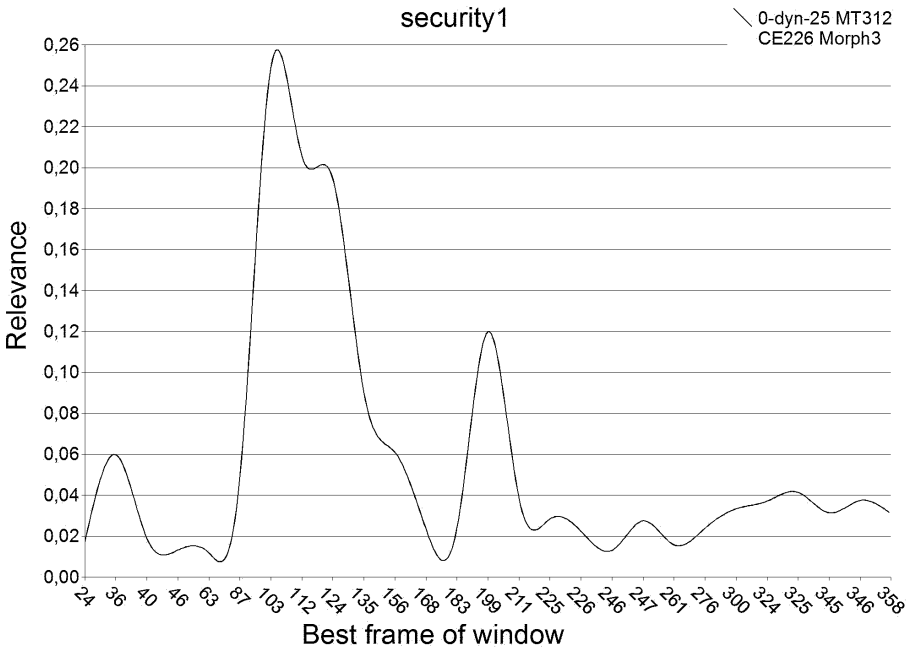

Figure 7.12: *Morph3* filtered video "security1" with 73 features, and window relevance defined as the sum of the relevances of the three best key frames.

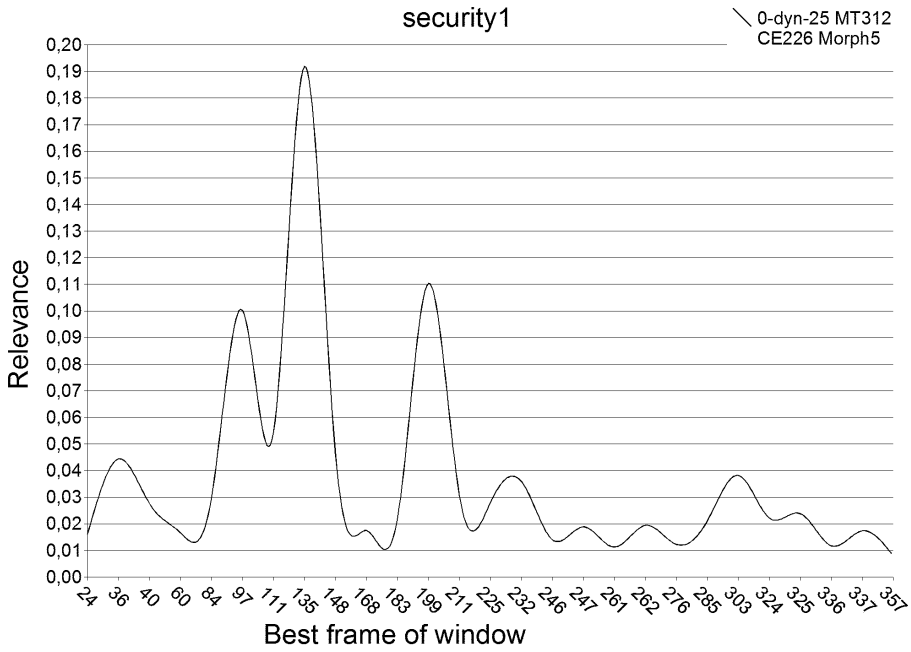

Figure 7.13: *Morph5* filtered video "security1" with 73 features, and window relevance defined as the sum of the relevances of the three best key frames.

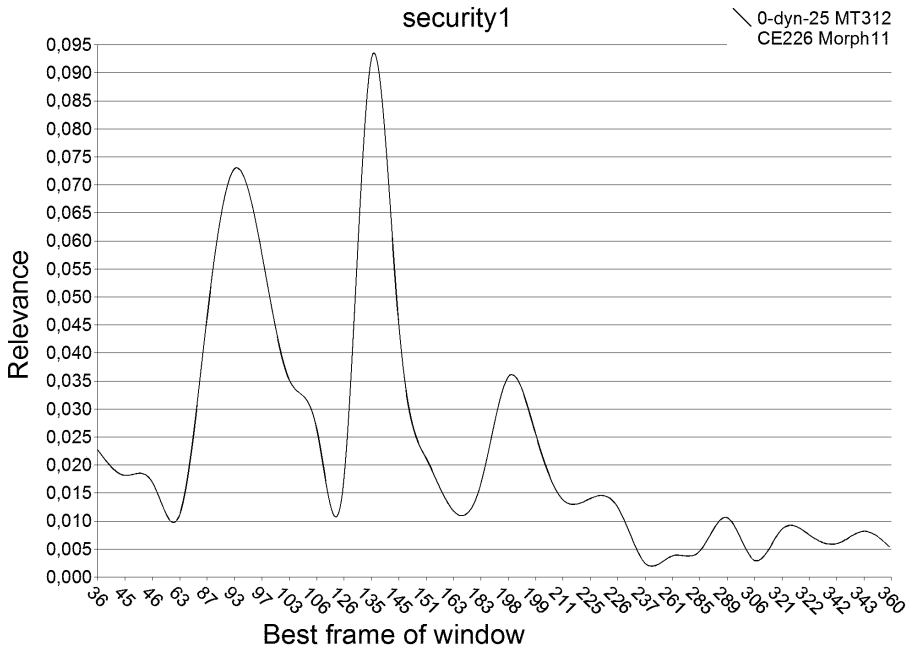

Figure 7.14: *Morph11* filtered video "security1" with 73 features, and window relevance defined as the sum of the relevances of the three best key frames.

Chapter 8

Results

8.1 Comparison

A complete experimental comparison with results can be found on the homepage [12]. The results contain a comparison with our algorithms for all available videos as used by [21, 48].

8.2 Summary

The discrete curve evolution is a flexible greedy algorithm for video segmentation and indexing, in order to extract key frames and create video summaries.

The flexibility is integrated into the ability to define frame descriptors for the appropriate purpose. The key frame measure functionality is not simply reduced to a metric; therefore resulting in more flexibility in order to adapt the key frame relevance measure to the definition of expected key frames. The strength of the algorithm is not the detection of potential key frame candidates but the detection of non-key frames. This makes it possible to detect a key frame on a variable context of the video and not only inside a static environment.

The existing centroid based frame descriptors are optimized by changing the size of histogram bins behind the centroids. We changed the centroid features in such a way that they scale invariance. We added a dynamic weighting of some components of these frame descriptors, depending on their importance

in the frame. This will avoid an incorrect detection of key frames. A filter gives use the latitude to define a time frame in which key frames are expected. We have shown that the algorithm is not only based on feature spaces with a defined metric, but the non-euclidean spaces are possibly also, as shown with the Dominant Colors and Optimal Color Composition Distance.

We have also shown that the Discrete Curve Evolution, with optimized frame descriptors based on centroids, is comparable with other video segmentation algorithms and results.

The flexibility to analyse frames in bigger context, as is necessary in video streams, is also a disadvantage for this video segmentation algorithm, thus making it unusable for the analysis of video streams. We bypassed this disadvantage by using the Discrete Curve Evolution for a window analysis of the video stream. This allowed us to detect important events inside the video stream. The window positioning algorithm is optimized to detect important events only once, however without leaving other less important events undetected.

8.3 Conclusion

In this paper, we have seen that the Discrete Curve Evolution is a suitable and flexible algorithm for video segmentation and indexing.

We developed applications to extract different video frame feature descriptors and we optimized them for best results. These optimizations were improved by optimizing the frame descriptors with the implementation of a filter.

The Discrete Curve Evolution gives us the ability to define frame descriptors, depending on the application in which a video segmentation algorithm is used. It also gives us the ability to implement a measure that meets the requirements of the video segmentation algorithm.

8.4 Future

The discrete curve evolution contains more potential for video segmentation than is shown in this paper. Some disadvantages of the used features are not discussed, such as slight changes in the brightness with a large impact on the frame descriptors. A suggestion of how this could be solved was already made.

More research is needed for usable frame descriptors, frame comparison metrics and for the cost measures for predefined key frame requirements. Trained frame descriptors and comparison metrics as used in CBIR, for example [28], are also interesting ideas.

Other research required is for upper and lower boundaries of useful abstraction levels. Merging differences between abstraction levels to groups could be also interesting in order to define application depending abstraction levels like it is done in [53].

For the video stream analysis, research is necessary for more intelligent window width and positioning, and for the implementation of an automatic detection algorithm.

Appendix A

Video sets

This appendix contains an overview of the videos I used for experiments as described in this paper. A complete list of videos is available at homepage [12].

A.1 Self-made video sets

A.1.1 Ground Truth

It is important for the ground truth videos that they are simple and the results must be identical for all tests (even persons and experiments).

Papers

We created ground truth experiments to verify the results. The first experiment shows 3 different colored papers on a white board. The camera moved slowly over all papers, ending with the first. The movie is known as "MOV1".

Facts of video *Mov1*

video size	160 x 112 pixels
ratio	0.7
number of frames	378 frames
frame rate	25 fps
video length	15.1 seconds
number of shots	1 shot

Video and shot summary of *Mov1*

Shot	start	end	Description
1	1	378	Table with 3 different colored pieces of paper.

The expected key frame results are the images of the different colored papers as shown in Image A.1.

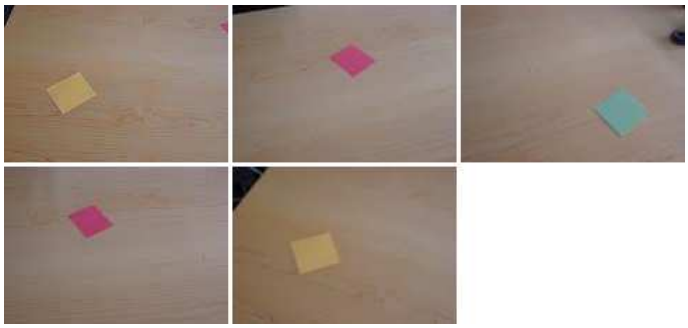

Figure A.1: Collation of the sequence "MOV1" with the expected ground truth results.

Dots

Movie "MOV00085" [13] is also known as "Dots". A hand-held camera shows 3 groups of magnetic dots on a white board. The camera moves slowly over a single black dot, to a group with 2 red dots, over to a group with 3 dots and back to the single black dot.

Facts of video *Mov00085*

video size	160 x 112 pixels
ratio	0.7:1
number of frames	387 frames
frame rate	25 fps
video length	15.5 seconds
number of shots	1 shot

Shot summary

Shot	start	end	Description
1	1	387	Whiteboard with different groups of magnetic dots.

The expected key frame results are images of the magnetic dots. The resulting key frame set contains images showing the cleared board between the dot groups or frames which included more than one group. Figure A.2 shows the expected key frames.

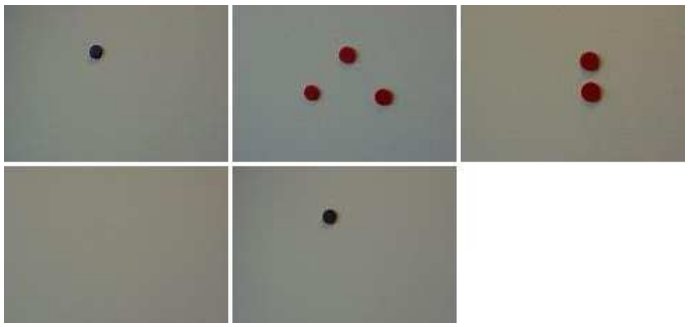

Figure A.2: Collation of the sequence "MOV00085" with the expected ground truth results.

Rustam

Movie "MOV3" [15] is also known as "Rustam". A hand-held camera shows the upper part of a man sitting. First he sits "still" making only small body movements. Then he waves his left hand (approx. frame 36), then he sits still, and then again he waves with his left hand (approx. frame 159). After a few seconds he waves his right hand (approx. frame 241).

Facts of video *Mov3*

video size	320 x 240 pixels
ratio	3:4
number of frames	386 frames
frame rate	25 fps
video length	15.4 seconds
number of shots	1 shot

Shot summary

Shot	start	end	Description
1	1	387	Waving Rustam.

Shot description:

1. Rustam waves (from the observers point of view) 2 times with his left hand and 2 times with his right hand. Rustam's hand did not disappear between the last two waves.

Figure A.3 shows the expected key frames.

Figure A.3: Collation of the sequence “MOV3” with the expected ground truth results.

A.1.2 Camera recordings

These videos were recorded with two different hand-held cameras. The first camera recorded small videos with 25 fps. These videos were in reality recorded with approx. 8 fps but every frame was tripled. The second camera recorded larger videos with 25 fps. These videos were in reality recorded with approx. 12 fps but every second frame was doubled.

Security 1

Movie “security1” [16] is a low resolution video of 160x112 pixels and has 386 frames. With a frame rate of 25 fps, it lasts for 15 seconds.

The video was taken with a fixed hand-held camera pointing at a closed door. It shows a room with white walls. A person dressed in white enters the room from the right. The person turns to the camera sits down, then stands up and leaves the view of the camera at the right border. Image A.4 shows the expected key frames.

The camera records the video at a lower rate of aprox. 8 fps. These frames are tripled which results in a frame rate of 25 fps. The first frame is always an I-frame and the next two frames are always B-frames, which are identical to the I-frame.

Facts of video “security1”

video size	160x112 pixels
ratio	0.7
number of frames	386 frames
frame rate	25 fps
video length	15 seconds
number of shots	1 shot

Shot summary

Shot	start end	Description
1		Longin Jan Squatting.

Shot description

1. Longin Jan appears in the camera view from the right side. He squats and disappears from the view to right side.

Figure A.4: Collation of the sequence "security1" with the ground truth results

Security 7

Movie "security7" [17] is a low resolution video of 160x112 pixels and has 386 frames. With a frame rate of 25 fps, it lasts for 15 seconds.

This video was taken with a fixed hand-held camera pointing at a closed door. It shows a room with a white wall and a door. At about frame 160, the door opens and the person enters the room. The person closes the door and walks towards the camera, passing it to the right and disappearing at about frame 255. Figure A.5 shows the expected key frames of this sequence.

Facts of video "security7"

video size	160x112 pixels
ratio	0.7
number of frames	386 frames
frame rate	25 fps
video length	15 seconds
number of shots	1 shot

Shot summary

Shot	start end	Description
1		Guest entering the room.

Shot description:

1. This video recorded a door through which a guest appeared. The guest disappeared from the camera view to the right side.

Figure A.5: Collation of the sequence “security7” with the expected ground truth results

A.1.3 Television and existing videos

Halloween

This is one of the first videos not self-made. It shows the first minutes of a Video CD named “Halloween”. This video was used only for performance tests and not for comparison purposes because there was no shot description nor ground truth available.

Facts of video “Halloween”

video size	352 x 288 pixels
ratio	9:11
number of frames	6182 frames
frame rate	29.7 fps
video length	206.3 seconds
number of shots	unspecified

Mr. Bean’s Christmas (full-sized)

This video is the full-sized version of “Mr. Bean’s Christmas”.

Facts of video “Mr. Bean’s Christmas”

video size	352 x 240 pixels
ratio	11:16
number of frames	2379 frames
frame rate	30 fps
video length	80 seconds
number of shots	9 shots

Shot summary

Shot	start	end	Description
1	0	994	Kitchen with Mr. Bean and a turkey
2	995	1165	Close-up of Mr. Bean
3	1166	1291	Kitchen with Mr. Bean and a turkey
4	1291	1357	Close-up of Mr. Bean
5	1357	2009	Kitchen with Mr. Bean and a turkey
6	2009	2079	Woman at the door
7	2080	2182	Kitchen with Mr. Bean and a turkey
8	2183	2363	Living room with Mr. Bean and a turkey
9	2364	2379	Woman at the door

We detected:
Frame 0 (per definition), which is representative for shot 1.
Frame 613 is also contained in shot 1 (Approx. 20.4”).
Frame 1021 is contained in shot 2 (Approx. 34.0”).
Frame 1154 is also contained in shot 2 (Approx. 38.5”).
Frame 1212 is contained in shot 3 (Approx. 40.4”).
Frame 1302 is contained in shot 4 (Approx. 43.4”).
Frame 1806 is contained in shot 5 (Approx. 60.2”).
Frame 2041 is contained in shot 7 (Approx. 68.0”).
Frame 2206 is contained in shot 8 (Approx. 73.5”).
Frame 2379 (per definition), which is representative for shot 9.

Shot description:

1. Shot one shows the kitchen without the turkey. In these shots, Mr. Bean brings the turkey into the view of the camera.

2. In shot two there is a zoom and a pan onto Mr. Bean. Due to this camera action, the turkey disappears from the camera view.

3. Shot three shows the same content as shot one.

4. Shot four shows the same content as shot two.

5. Shot five shows the same content as shot one.

6. Shot six shows a woman knocking on the front door.

7. Shot seven shows the same kitchen as in shot one. Mr Bean has his head in a turkey.

8. Shot eight shows the living room as Mr. Bean enters it from the kitchen.

9. Shot nine shows the woman from shot six waiting at the front door.

Figure A.6: Collation of the sequence "Mr. Bean's Christmas" with the expected ground truth results

Mr. Bean's Christmas (down-scaled version)

The down-scaled version of the movie "Mr. Bean's Christmas" (mrbeantu.mpg) is the identical version of the full-sized version with an exception in the resolution, which has a size of 112x80 pixels.

Facts of video "Mr. Bean's Christmas"

video size	112 x 80 pixels
ratio	0.7
number of frames	2379 frames
frame rate	30 fps
video length	80 seconds
number of shots	9 shots

A.2 Third party video sets

A.2.1 Rossiter et. al.

For our comparison experiments, we used a set of 3 videos which are available at the hompage of Rossiter [48]. The videos contain a series of clips of the motion picture "The Blade Runner", a series of shots through a house and a series of scenes of the singer, Kylie Minogue. A description of the videos can be found at [12].

Example information of "Kylie"

The video with information about "Kylie", as used in chapter 6, was recorded at a slower frame rate, resulting in a fast playback of the content.

Facts of video "Kylie"

video size	192 x 144 pixels
ratio	3:4
number of frames	205 frames
frame rate	25 fps
video length	8.2 seconds
number of shots	6 shots

Video and shot summary

Shot	start	end	Description
1	1	41	Interview with Kylie Minogue.
2	42	80	Dancing performance.
3	81	101	Other dancing performance.
4	102	105	A dancer.
5	106	189	Another interview with Kylie Minogue.
6	190	205	Lead out of the BBC television.

The expected ground truth result is shown in figure A.7. It contains one frame of every shot.

Figure A.7: Collation of the sequence "Kylie" with the expected ground truth results.

A.2.2 Drew et. al.

The experiment set of Drew contains 12 self-created and two downloaded videos. The self-created videos are short clips less than one minute in length and contain one to seven shots. Some shots also have pans and blendings between them. The two downloaded videos come from the university of Kansas. The first clip is a scene of a football match and the second clip is a series of basketball shots.

Example information of "beachmov"

Information about the video "beachmov" as used in chapter 6

Facts of video "beachmov"

video size	321 x 240 pixels
ratio	3:4
number of frames	738 frames
frame rate	25 fps
video length	29.5 seconds
number of shots	4 shots

Video and shot summary

Shot	start	end	Description
1	1	300	A pan over a beach
2	275	535	Beach volleyball
3	510	700	Four people in a swimming pool
4	675	739	People on the edge of a swimming pool

Special shot characteristics

- Shot one shows a beach, and the camera makes a pan from left to right showing water, a beach and a forest in the background.

The expected ground truth result is shown in figure A.8. It contains two frames for the first shot and one frame for each of the following shots.

Figure A.8: Ground truth results of video “beachmov”

Appendix B

Software

It was necessary to implement these algorithms in our own programs in order to verify them and get knowledge of them. Tthe programming language made it possible for me to alter the algorithms. I have implemented the algorithms in other programs based on C/C++.

The feature extraction was implemented in an software MPEG-player. The MPEG-player is a C-program in which I implemented the bucket extraction for each color space (RGB and YUV). The dominant colors were implemented by a library of Hu. I have created for each kind of modification to the feature extraction algorithm an own program with a different version. The different versions of the feature extraction are described in Appendix B.1.1. The features are directly extracted from the video memory. The MPEG player also extracts the key frames if this is necessary.

The feature filtering was implemented in a separate program which is written in C++. For each kind of feature and filter, there is an own program version written with a different version number. The different versions of the feature filter are described in Appendix B.1.2.

The discrete curve evolution was also implemented in a separate program which was written in C/C++. The original version of this program was written by Daniel de Menthon. For each kind of feature was also an own program version written with a different version number. The distance measure for the buckets were directly implemented by a vector implementation. The distance measure for the dominant colors were implemented by a library which was supported by Hu. The different versions of the feature filter are described in Appendix 2.1.

The player was a developed application and implemented in C++ by Jan Erik Hoffmann. The player was written in C++. It is a graphical user interface (GUI) between the discrete curve evolution, the key frames and an MPEG player. The player is called "*Smart Fast Forward Player*" (SFF-player). The player reads the evolved evo file and shows the images from the MPEG player extracted images at a specific abstraction level. The level could easily be changed by a slider. The player also controls a commercial MPEG player with sound support. When selecting an image, it will skip into the same image in the MPEG video stream. From this point on, the video can be played normally.

The interaction between the applications is realized by data files and scripts. The input and output for each program are data files such as:

- MPEG1-video/system Stream
- Feature Files
- Frame List
- Key Frames

The scripts are the "glue" between the separate programs and the data files. The scripts perform:

- Directory creation
- Extraction of the features
- Filtering
- Discrete Curve Evolution
- Frame Extraction.

Appendix B.2 describes how these programs are interacting with each other for the different algorithms. In Appendix B.3, the different file formats are described which are produced by the programs.

B.1 Applications

B.1.1 Feature Extraction

The features are extracted within the MPEG player. There exist different versions of the player for different features. These player versions are explained in the following sub-sections.

B.1.2 Feature Filter

Version 2.x and 3.x of the "Feature Filter Application" is used to filter the buckets. These applications accept the following parameters:

- *-i <FFT filename>*
  The parameter *<FFT filename>* specifies the input filename for the filter.
- *-o <FFT filename>*
  The parameter *<FFT filename>* specifies the output filename for the filter.
- *-FilterBase <filter width>*
  The parameter *<filter width>* specifies the base width of the min/max filter. The total filter width is "$1 + 2* < filterwidth >$".
- *-FilterType <filter type>*
  The parameter *<filter type>* specifies if the minimum or maximum filter operation is done. 0 implies the minimum filter and 1 implies the maximum filter.

Version 4.x of the "Feature Filter Application" filters the dominant color feature file as used by the OCCD routines of Hu. The applications accepts the following parameters:

- *-i <FTR filename>*
  The parameter *<FTR filename>* specifies the input filename for the filter with the dominant color features.
- *-ippm*
  This parameter specifies if a feature image PPM-file should be created from the input FTR-file.

- *-o <FFT filename>*
  The parameter *<FFT filename>* specifies the output filename for the filter.

- *-oppm*
  This parameter specifies if a feature image PPM-file should be created from the output FTR-file.

- *-FilterBase <filter width>*
  The parameter *<filter width>* specifies the base width of the min/max filter. The total filter width is $1 + 2* < filterwidth >$.

- *-FilterType <filter type>*
  The parameter *<filter type>* specifies if the minimum or maximum filter operation is done. 0 implies the minimum filter and 1 implies the maximum filter.

B.1.3 Curve evolution

Curve Evolution Application version 3.23 is the most used application for features based on the Dominant Colors. The application accepts the following parameters:

- *-i <EFT filename>*
  The parameter *<EFT filename >* specifies the input file for the extended centroid features with the video control information. This information is used to associate the frame number to the MPEG-video.

- *-i2 <FFT filename >*
  The parameter *<FFT filename >* specifies the input file for the dominant color frame descriptors which are used in the curve evolution algorithm.

- *-o < output filename>*
  The parameter *< output filename >* specifies the output filename with the resulting data of the curve evolution.

- *-n <number of I-frames>*
  The parameter *<number of I-frames >* contains the number of intra-frames before this frame. This was used to calculate an offset in the MPEG-stream on which a MPEG-viewer could start playing, thus making it possible to start the MPEG a few seconds before the frame appears.

- *-f <frame type>*
  The parameter *<frame type>* specifies which of the frames of the MPEG video are used for the curve evolution. It is possible with this option to perform the curve evolution on only a subset of frames of the video. A value larger or equal to two will use all intra (I), predictable (P) and between (B) frames. A value of one will skip the between frames, and a value of zero will only use the intra frames.

- *-start <start frame number>*
  The parameter *<start frame number>* specifies which frame the curve evolution is applied to.

- *-end <end frame number>*
  The parameter *<end frame number>* specifies until which frame the curve evolution is applied.

  With the parameters *<start frame number>* and *<end frame number>*, an endless video is used to simulate the curve evolution for the local analysis. Only the subset of the features with these frame boundaries is used for the curve evolution.

- *-mpg <MPEG filename>*
  The parameter *<MPEG filename>* is written in the output file as reference to the original MPEG video. This information is used by our Smart Fast Forward Viewer to load the appropriate video.

B.1.4 MpegInfo

This application extracts the total number of frames in a video and is used by the scripts to control the extraction process in which information about the total number of frames is used.

B.1.5 Smart Fast Forward Viewer

The Smart Fast Forward Viewer was developed by Jan Erich Hoffman as a Graphical User Interface to join the results of the Key Frame Extraction Algorithms and the original content of the video together. It acts as:

1. an abstraction level selector

2. an abstraction viewer with preservation of the temporal information, and

3. a remote control for an external MPEG-player [42].

B.1.6 MPEG-Player

We used a commercial MPEG player [42] to show the video content from the Smart Fast Forward viewer. This decision was made because other MPEG-players didn't support audio and suitable remote control mechanisms at that time.

B.1.7 Scripts

The complete creation process of the key frames, starting with the feature extraction and ending with the key frame extraction, is controlled by shell scripts to make an autonomous system without any necessary user interaction.

B.2 Application interaction diagram

The different kinds of applications interact with each other and exchange data. The following two subsections show the interactions between the applications as described in the previous section for the global analysis of the videos and for the local analysis inside video streams.

B.2.1 Global key frame extraction

Our experiments for the video scene key frame extraction were made by several programs. Image B.1 shows the different steps of the key frame extraction. In the first step, the frame features are extracted from the video sequence to an intermediate file. The input file only contains the video sequence. The output file format is described in Chapter B.3.1.

In the second step, the filters applied to the features. The input and the output file format are identical.

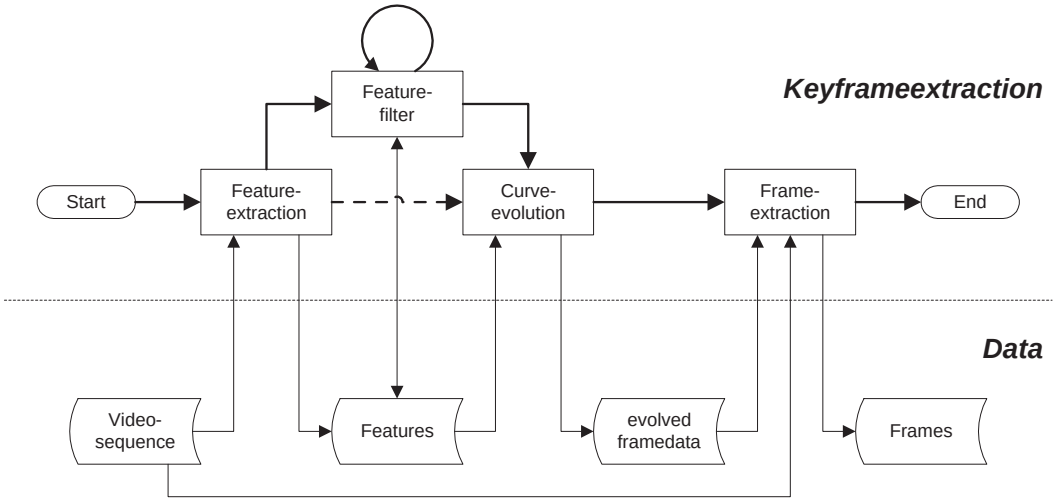

Figure B.1: Framework of the key frame extraction process.

The third step is the digital curve evolution. The input file is the intermediate file. The output file contains the relevance order and the relevance of the filtered frames. The file format is described in Chapter B.3.3.

In the fourth step, the most important images are extracted from the video sequence. The input files contain the video sequence from step one and the evolution file from step three. The output files contain the key frames.

In step five, the key frames and video sequence can be viewed with the smart fast forward viewer (SFF-viewer) or with another application for example in a server/client environment. Image B.2 shows the framework for the SFF viewer and also an example of an embedded client/server environment with a HTTP web server and a browser. An interactive example can be found on http://www.videokeyframes.de/ [12]. The input files for the SFF-player contain the video sequence from step one, the key frames from step four and the evolution file from step three.

B.2.2 Local key frame extraction

Our experiments for the video scene key frame extraction were made by several programs. Image B.3 shows the different steps of the key frame extraction in open scenes.

The key frame detection is done with the framed version of the key frame extraction. Image B.4 shows the key frame.

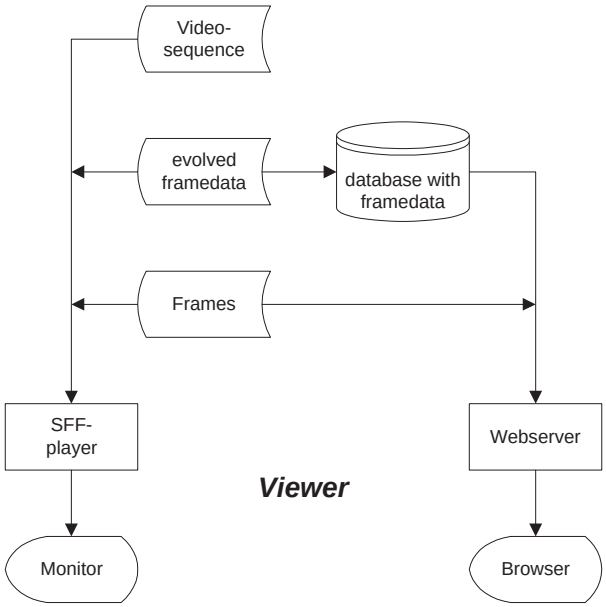

Figure B.2: Framework of the smart fast forward viewer

B.3 File formats

B.3.1 Bucket Features

The bucket features are stored in a file with the suffix ".eft". The file contains a header of 3 lines. All other lines contain the features in space-separated columns. Line one contains the number of frames. Line two contains the number of columns which should be skipped. Line three contains the number of feature columns. The first data column is the time (frame number). The following columns are in order of the color component (red, green, blue or $Chrominance$, $Luminance_r$, $Luminance_b$). Then the bucket numbers are in order from the lower to the higher values and then the bucket itself with the bucket "x" coordinate, the bucket "y" coordinate and the area. The values are integer promille values in the value range.

Data File:

```
[no.  frames] <new line>
[no.  extended columns] <new line>
[no.  feature columns] <new line>
<no.  frames> * [frame data]
```

Frame Data:

```
<no.  extended columns> * [extended data]
```

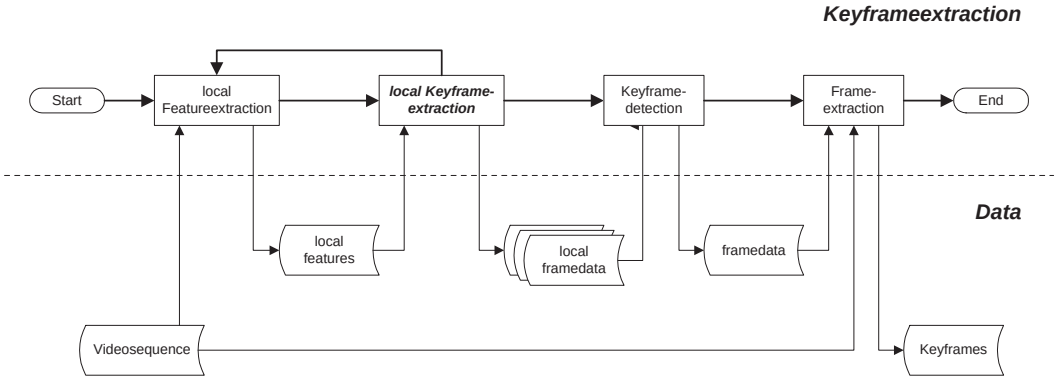

Figure B.3: Framework of local key frame algorithm

```
<no.  feature columns> * [feature data]
<new line>
```

Example:

```
6182
5
37
87064 0 1 2064 00000000 1 16 55 997 470 796 2 904 746 0 0 0 ↩
↪ 0 0 0 0 173 54 683 58 63 316 0 0 0 0 0 0 527 328 132 75 ↩
↪ 11 863 576 934 3
...
```

This file contains features for 6182 frames. The features start at column 6. (This is the column after "000000".) We have 37 features. This implies 4 buckets for each color. The "x" coordinate of the first bucket in the luminance color is 16. The "y" coordinate of the first bucket in the luminance color is 55. The area of the first bucket in the luminance color is 997. Because this is the first bucket of the luminance, it contains the darker pixels of the image. This bucket contains 99,7 percent of all pixels, which means that the image is very dark.

The "x" coordinate of the second bucket in the luminance color is 470. The "y" coordinate of the second bucket in the luminance color is 796. The area of the second bucket in the luminance color is 2.

B.3.2 Dominant color Features

The dominant color features are stored in a file with the suffix ".ftr". The file contains a header of 1 line. All other lines contain the features in frame blocks

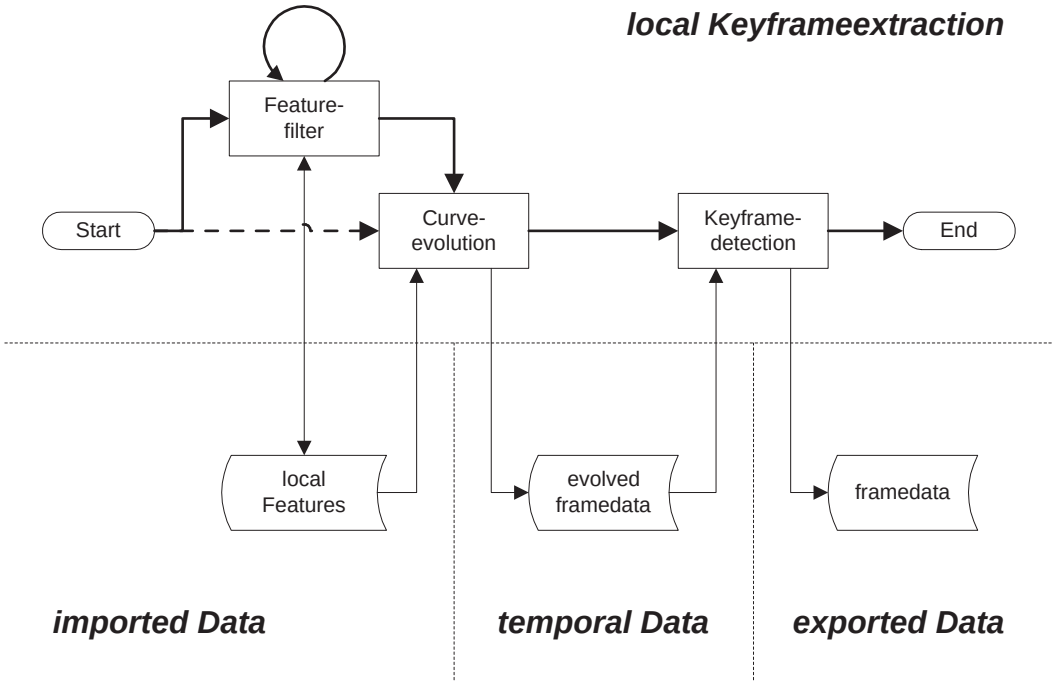

Figure B.4: Framework of the potential key frame detection part

and space-separated columns. Line one contains the number of frames. All other lines are blocks with dominant color information for one frame. Line one contains 2 values. Value one is the frame number. Value two is the number of dominant colors in this frame. Each of the following lines in the block contain information about one dominant color. A line with dominant color information contains seven values. Values one to three are the values for "L"*, "a*" and "b*". Value number four is the color number in the codebook. Value number five is the area of the color in the image. Value number six and seven are the x- and y-coordinates of the centroid.

Data File:

```
[no.  frames] <new line>
<no.  frames> * [frame data]
```

Frame Data:

```
[frame number] [#colors] <new line>
<no.  colors> * [color data]
```

Color Data:

```
[L*] [a*] [b*] [codebook color no.]  [area] [x] [y] <new line>
```

Example:

```
572
0 4
40.939472 35.434769 -76.906685 38 0.414575 0.412636 0.466091
0.000000   0.000000   0.000000  0 0.410669 0.593721 0.483514
40.000000 15.242203 -41.250500 24 0.033440 0.592466 0.793416
40.000000 34.271545 -53.427330 32 0.023773 0.547657 0.631149
1 ...
```

B.3.3 Evolved frame information

The result of the curve evolution is written in a file with the suffix ".evo".

Data File:

```
[Video Filename] <new line>
[Image Path] <new line>
[] <new line>
[] <new line>
[no.  frames] <new line>
<no.  frames> * [Frame Data]
```

Data File:

```
[Entry no.]  [relevance] [Frame no.]  [Frame no.]
[Video Offset] [Intraframe no.]  [Video Offset]
[Intraframe no.]  <new line>
```

Example:

```
/home/Danny/Promotion/Programme/Data2/BladeRunner.mpg
./
79
2
572
0 0.000000 94 94 291772 93 275200 90
1 ...
```

List of Figures

Bibliography

[1] G. Ahanger and T. Little. A survey of technologies for parsing and indexing digital video. *J. of Visual Communication and Image Representation*, 7:28–43, 1996.

[2] Internet search engine.
Internet: http://www.altavista.com, October 2005.

[3] Michle Basseville and Igor Nikiforov. *Detection of Abrupt Changes - Theory and Applications*. Information and System Sciences Series. Prentice Hall, Englewood Cliffs, 1993.

[4] A. Bhattacharyya. On a measure of divergence between two statistical populations defined by their probability distributions. *Bull. Calcutta Math. Soc.*, 35:99–110, 1943.

[5] J. Boreczky and L. Rowe. Comparison of video shot boundary detection techniques. In *Proc. SPIE Storage and Retrieval for Image and Video Databases.*, 1996.

[6] M. Bosc, F. Heitz, J. P. Armspach, I. Namer, D. Gounot, and K. Rumbach. Automatic change detection in multimodal serial mri: application to multiple sclerosis lesion evolution. *Neuroimage*, 20:643–656, 2003.

[7] L. Bruzzone and D. F. Prieto. An adaptive semiparametric and context-based approach to unsupervise change detection in multitemporal remote-sensing images. *IEEE Transactions on Image Processing*, 11(5):452–466, April 2002.

[8] Berkeley Multimedia Research Center. The berkeley mpeg player. Internet: http://bmrc.berkeley.edu/frame/research/mpeg/mpeg_play.html, October 2005.

[9] J. B. Collins and C. E. Woodcock. An assessment of several linear change detection techniques for mapping forest mortality using multitemporal landsat tm data. *Remote Sensing Environment*, 56:66–77, 1996.

[10] R. Collins, A. Lipton, and T. Kanade. Introduction to the special section on video surveillance. *IEEE Transaction Pattern Anal. Machine Intelligence*, 22(8):745–746, August 2000.

[11] D. de Wildt and L. J. Latecki. Comparison between a clustering and the discrete curve evolution algorithm. Internet: http://www.videokeyframes.de/Information/Doc/GlobalResults.pdf, October 2005.

[12] D. de Wildt and L. J. Latecki.
project homepage for temporal video segmentation.
Internet: http://www.videokeyframes.de/, October 2005.

[13] D. de Wildt and L. J. Latecki. Video "mov00085".
Internet: http://www.videokeyframes.de/Mov00085.mpg, October 2005.

[14] D. de Wildt and L. J. Latecki. Video "mov1".
Internet: http://www.videokeyframes.de/Mov1.mpg, October 2005.

[15] D. de Wildt and L. J. Latecki. Video "mov3".
Internet: http://www.videokeyframes.de/Mov3.mpg, October 2005.

[16] D. de Wildt and L. J. Latecki. Video "security1".
Internet: http://www.videokeyframes.de/seciurity1.mpg, October 2005.

[17] D. de Wildt and L. J. Latecki. Video "security7".
Internet: http://www.videokeyframes.de/seciurity7.mpg, October 2005.

[18] D. F. DeMenthon, V. M. Kobla, and D. Doermann. Video summarization by curve simplification. In *Proc. ACM Multimedia*, pages 211–218, 1998.

[19] Daniel DeMenthon, Longin Jan Latecki, and Azriel Rosenfeld. Video summarization by polygon simplification. In *IEEE PAMI*, pages 1185–1190, october 2000.

[20] D.F. DeMenthon, L.J. Latecki, A. Rosenfeld, and M. Vuilleumier Stückelberg. Relevance ranking and smart fast-forward of video data by polygon simplification. In *Proc. Int. Conf. on Visual Information Systems*, pages 49–61, 2000.

[21] M. S. Drew and J. Au. Video keyframe production by efficient clustering of compressed chromaticity signatures. In *Proc. ACM Multimedia*, 2000. http://www.cs.sfu.ca/~mark/ftp/AcmMM00/.

[22] D.R. Edgington, K.A. Salamy, M. Risi, R.E. Sherlock, D. Walther, and Christof Koch. Automated event detection in underwater video. In *OCEANS 2003. Proceedings*, volume 5, pages 2749– 2753, 22-26 Sept. 2003.

[23] H. Tamura et. al. Texture features corresponding to visual perception. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-8(6):460–473, 1978.

[24] Jianping Fan, Walid G. Aref, Ahmed K. Elmagarmid, Mohand-Said Hacid, Mirette S. Marzouk, and Xingquan Zha. Multiview: Multilevel video content representation and retrieval. *Jounal of Electronic Imaging*, 10(4):895–908, October 2001. 10(4).

[25] C.-Y. Fang, S.-W. Chen, and C.-S. Fuh. Automatic change detection of driving environments in a vision-based driver assistence system. *IEEE Transactions on Neural Networks*, 14(3):646–657, May 2003.

[26] A.M. Ferman and A.M. Tekalp. Efficient filtering and clustering methods for temporal video segmentation and visual summarization. In *J. Visual Communication and Image Representation*, 9, pages 336–351, 1998.

[27] H. Gabow. *Implementation of algorithms for maximum matching on nonbipartite graphs*. PhD thesis, Standford University, 1973.

[28] Viper group on Multimedia Information Retrieval. Live internet demonstration:. http://viper.unige.ch/demo/php/demo.php, October 2005.

[29] J. Hu and A. Mojsolovic. Optimal color composition matching of images. In *Proc. Int. Conf. on Pattern Recognition*, volume 4, pages 47–51, 2000. Barcelona.

[30] W. Y. Kan, J. V. Krogmeier, and P. C. Doerschuk. Model-based vehicle tracking from image sequences with and application to road surveillance. *Opt. Eng.*, 35(6):1723–1729, 1996.

[31] Rajeev Kumar and Vijay Devatha. A statistical approach to robust video temporal segmentation. In *Proc. on Int. Conf. on Computer Visual Graphics and Image Processing (ICVGIP)*, December 2002.

[32] Rolf Lakämper. *Formbasierte Identifikation zweidimensionaler Objekte*. PhD thesis, University of Hamburg, Hamburg; Germany, 1999.

[33] E. Landis, E. Nagy, D. Keane, and G. Nagy. A technique to measure 3d work-of-fracture of concrete in compression. *Journal Engineering Mechanics*, 126(6):599–605, June 1999.

[34] Language and Media Processing Laboratory; University of Maryland. Internet homepage:. http://www.cfar.umd.edu/.

[35] L. J. Latecki and D. de Wildt. Automatic recognition of unpredictable events in videos. In *Proc. of Int. Conf. on Pattern Recognition (ICPR)*, Quebec City, August 2002.

[36] L. J. Latecki, D. de Wildt, and J. Hu. Extraction of key frames from videos by optimal color composition matching and polygon simplification. In *Proc. Multimedia Signal Processing*, pages 245–250, Cannes, France, October 2001.

[37] L. J. Latecki, D. DeMenthon, and A. Rosenfeld. Automatic extraction of relevant frames from videos. In *Proc. German Conf. on Pattern Recognition*, pages 412–419. DAGM, 2000.

[38] L. J. Latecki and R. Lakämper. Convexity rule for shape decomposition based on discrete contour evolution. *Computer Vision and Image Understanding*, 73:441–454, 1999.

[39] L. J. Latecki and R. Lakämper. Shape similarity measure based on correspondence of visual parts. *IEEE Trans. Pattern Analysis and Machine Intelligence*, 22:1185–1190, 2000.

[40] K. Lebart, E. Trucco, and D. M. Lane. Real-time automatic sea-floor change detection from video. *MTS/IEEE OCEANS 2000*, pages 337–343, September 2000.

[41] A. Mojsilovic, J. Kovacevic, J. Hu, R. Safranek, and K. Ganapathy. Matching and retrieval based on the vocabulary and grammar of color patterns. *IEEE Transactions on Image Processing*, 9(1):38–54, 2000.

[42] mtv - mpeg player for linux & unix. Internet homepage: http://www.mpegtv.com/, October 2005.

[43] G. Nagy, T. Zhang, W. Franklin, E. Landis, E. Nagy, and D. Keane. Volume and surface area distributions of cracks in concrete. *Visual Form 2001 (Springer LNCS 2059)*, pages 759–768, 2001.

[44] Mozilla Organisation.
Internet: http://www.mozilla.org/products/firefox/, October 2005.

[45] Second ieee international workshop on performance evaluation of tracking and surveillance. Internet: http://visualsurveillance.org/PETS2001, December 2001.

[46] D. Rey, G. Subsol, H. Delingette, and N. Ayache. Automatic detection and segmentation of evolving processes in 3d medical images: Application to multiple sclerosis. *Medical Image Analysis*, 6(2):163–179, June 2002.

[47] Omar Al-Kofahi Richard J. Radke, Srinivas Andra and Badrinath Roysam. Image change detection algorithms: A systematic survey. Technical report, Department of Electrical, Computer, and Systems Engineering Rensselaer Polytechnic Institute, 110 8th Street, Troy, NY, 12180 USA, 2004.

[48] Dr David Rossiter. Department of computer science, hong kong university of science and technology, clear water bay, kowloon, hong kong. Internet homepage: http://www.cs.ust.hk/~rossiter/mm_projects/video_key_frame/video_key_frame.html, October 2005.

[49] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Press, 1982.

[50] C. Stauffer and W. E. L. Grimson. Learning patterns of activity using real-time tracking. *IEEE Transactions on Pattern Anal. Machinne Intelligence*, 22(8):747–757, August 2000.

[51] G. Wyszecki and W. S. Stiles. *Color Science: concepts and methods, quantitative data and formulae.* John Wiley and Sons, New York, 1982. ISBN 0-471-02106-7.

[52] Di Zhong and Shih-Fu Chang. Content based video indexing techniques. http://www.ctr.columbia.edu/~dzhong/Papers/hier.html, October 2005.

[53] Xingquan Zhu, Jianping Fan, Ahmed K. Elmagarmid, and Xindong Wu. Hierarchical video content description and summarization using unified semantic and visual similarity. *Multimedia Systems*, 9:31–53, 2003.

abstract

This work treats the segmentation of videos with pre-defined start and end frame, as well as video data streams without a firmly defined start and end frame. The video segmentation consists of the time analysis of the video data material and the determination of striking segments in this material, as well as the representation of the segments by single frames.

The video segmentation is used in multiple applications, e.g. in the production control, monitoring as well as in the automatic interpretation of videos.

The fundamental concepts of the used image processing are explained. Since there are no objective and clear criteria for the definition of video segments, subjective criteria are deduced, which a video segmentation should fulfill.

As foundation for the video segmentation the discrete curve evolution is used, which turned out as a flexible and versatile applicable algorithm. It was developed at the University of Hamburg by Mr. Latecki and Mr. Lakaemper and used by Mr. DeMenthon at the University of Maryland (USA) for video segmentation. The used frame descriptors are based on the statistical analysis of pixels, whereby a reduction of the picture information in $\mathbb{R}^{37}$ is reached. These frame descriptors and algorithms are further analyzed and verified by experiments. This leads to the normalization of the basic video material concerning playing speed of the video and the dimensions of the display format. Additionally the selected number of frame descriptors is improved, normalized and weighted. The use of morphological filters on the characteristics round off the improvement of these characteristics. A further beginning off the improvement of video segmentation is the use of other frame descriptors, e.g. the RGB-color space and/or dominating colors, on which will be get into deeper.

It will be shown, how the video segmentation algorithm can be applied to video data streams by the use of a window. The problems and possible proposals for solution arising here are addressed.

The work is rounded off by a description of the accomplished experiments.

Zusammenfassung

Die Arbeit behandelt die Segmentierung von sowohl Videos mit vordefinierten Anfangs- und Endbildern als auch Videodatenströme ohne ein fest definiertes Anfangs- und Endbild. Die Videosegmentierung besteht aus der Zeitanalyse des Videodatenmaterials und der Bestimmung von markanten Abschnitten in diesem Material, sowie der Repräsentation der Abschnitte durch Einzelbilder.

Die Videosegmentierung findet vielfache Anwendung z.B. in der Produktionskontrolle, der Überwachung sowie in der automatischen Interpretation von Filmen.

Es werden die Grundbegriffe der verwendeten Bildverarbeitung erklärt. Da es keine objektive und eindeutige Kriterien für die Definition der Videosegmente gibt, werden subjective Kriterien hergeleitet, die eine Videosegmentierung erfüllen sollte.

Als Grundlage für die Videosegmentierung wird die Diskrete Kurvenevolution verwendet, die sich als ein flexibler und vielseitig anwendbarer Algorithmus herausgestellt hat. Sie wurde an der Universität Hamburg von Herrn Latecki und Herrn Lakämper entwickelt und von Herrn DeMenthon an der Universität Maryland (USA) für die Videosegmentierung angewandt. Die benutzten Bilddeskriptoren basieren auf der statistischen Analyse von Bildpunkten, wodurch eine Reduzierung der Bildinformationen auf $\mathbb{R}^{37}$ erreicht wird. Diese Bilddeskriptoren und Algorithmen werden näher analysiert und durch Experimente verifiziert. Die daraus gewonnenen Informationen werden benutzt um die Deskriptoren und Algorithmen zu verbessern. Dies führt zur Normalisierung des zugrunde liegenden Videomaterials bzgl. Abspielgeschwindigkeit des Videos und der Dimensionen des Bildformates. Zudem wird die gewählte Anzahl der Bilddeskriptoren verbessert, normalisiert und gewichtet. Die Anwendung von morphologischen Filtern auf den Merkmalen rundet die Verbesserung dieser Merkmale ab. Ein weiterer Ansatz für die Verbesserung der Bildsegmentierung ist die Verwendung von anderen Bilddeskriptoren, wie z.B. der RGB-Farbraum bzw. dominierende Farben, worauf vertieft eingegangen wird.

Es wird gezeigt, wie der Videosegmentierungsalgorithmus durch die Verwendung eines Fensters auch auf Videodatenströme angewandt werden kann. Die hier auftretenden Probleme und mögliche Lösungsvorschläge werden angesprochen.

Die Arbeit wird durch eine Beschreibung der durchgeführten Experimente abgerundet.

Lebenslauf

Persönliche Angaben

Name:	Daniël de Wildt
Geburtstag:	29. Oktober 1969
Geburtsort:	Heerlen (Niederlande)
Familienstand:	verheiratet

Schulausbildung

1975-1977:	Niederländische Grundschule in Zeven
1982-1986:	Niederländische Realschule in Zeven
1986-1989:	Sankt-Viti Gymnasium Zeven, Abiturabschluss

Ausbildung

1989-1997:	Studium der angewandten Mathematik an der Universität Hamburg.
1996-1997:	Diplomarbeit bei Siemens AG München, Abteilung Zentrale Forschung und Entwicklung.
1997:	Diplomabschluss im Fach Mathematik. Titel der Diplomarbeit "Segmentrandverfolgung durch ein aktives Konturenmodell".
2003-2005:	Promotionsstudium Mathematik an der Universität Hamburg.

Beruflicher Werdegang

1990-1997:	Freier Mitarbeiter bei dataplan Beratungsgesellschaft mbH in Hamburg.
1997-2001:	Technischer Angestellter bei Stelljes Media Line GmbH & Co. KG in Bremervörde.
2001-2002:	EDV-Leiter bei Druckhaus Stelljes GmbH & Co. KG in Bremervörde.
seit 2002:	Softwareingenieur bei benntec Systemtechnik GmbH in Bremen.

Veröffentlichungen

1. L.J. Latecki, D. de Wildt und J. H. Hu. "Extraction of Key Frames from Videos by Optimal Color Composition Matching and Polygon Simplification". *Proc Multimedia Signal Processing*, pages 245-150 Cannes, France, October 2001.
2. L.J. Latecki, D. de Wildt. "Automatic Recognition of Unpredictable Events in Videos". *Proc. Int. Conf. On Pattern Recognition (ICPR)* Quebec City, August 2002.