Automatic Video Segmentation by Polygon Evolution

Dissertation
zur Erlangung des Doktorgrades
des Fachbereiches Mathematik
der Universität Hamburg

vorgelegt von
Daniël de Wildt
aus Heerlen (Niederlande)

Bremervörde
2004
Version 0.9.4 vom 08.01.2004

Als Dissertation angenommen vom Fachbereich
Mathematik der Universität Hamburg

auf Grund der Gutachten von
und

Hamburg, den

Prof. Dr. Sprecher des Fachbereichs Mathematik

Contents

1 Introduction **3**
- 1.1 Introduction . . . 3
- 1.2 Motivation . . . 3
- 1.3 Classical temporal video segmentation and indexing algorithms 6
- 1.4 Subject of this work . . . 8

2 Fundamental **10**
- 2.1 Discrete Curve Evolution . . . 10
- 2.2 Video processing terminology . . . 13
- 2.3 Quality Measurement . . . 17

3 Key Frame Detection **19**
- 3.1 Key frames requirements . . . 19
- 3.2 Image and Video Descriptors . . . 22
- 3.3 Relevance Measure . . . 24
- 3.4 Conclusion . . . 28

4 Closed Videos **29**
- 4.1 Abstract review of existing applications . . . 29
- 4.2 Algorithm and software analysis . . . 31
- 4.3 Dominant Colors and Optimal Color Composition . . . 57

5 Experiments on closed videos **65**
- 5.1 YUV vs. RGB . . . 65
- 5.2 Different Cost Functions for the centroid feature . . . 67
- 5.3 OCCD only . . . 67

6 Comparison **79**
- 6.1 Algorithm comparison with Kumar et. al . . . 79
- 6.2 Algorithm comparison with Zhu et. al. . . . 81
- 6.3 Experimental comparison with David Rossiter et. al. . . . 84

6.4 Experimental comparison with Drew 84

7 Video stream **87**
7.1 Window analysis 88
7.2 Window Position 91
7.3 Window width 94
7.4 Window relevance 96
7.5 Event detection threshold 97
7.6 Filtering 102

8 Results **109**
8.1 Comparison 109
8.2 Summary 109
8.3 Conclusion 110
8.4 Future 110

A Video sets **111**
A.1 self made video sets 111
A.2 Third party video sets 118

B Software **121**
B.1 Applications 123
B.2 Application interaction diagram 125
B.3 File formats 127

List of figures **131**

Bibliography **136**

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. Key frames are very adequate for human visual perception, since we can easily judge the content of a video by viewing its representation by a small set of key frames. It is be 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 is an useful representation of the video by a small set of frames is required.

1.2 Motivation

Key frame detection is widely used in the literature, applications and in practice. Radke [44] has a reference list with a wide range of different applications for these class of video sequences like *Video Surveillance* [43, 47], *Remote Sensing* [7, 9], *Medical Diagnosis and Treatment* [12, 38], *Civil Infrastructure* [23, 27], *Underwater Sensing* [30, 10] or *Driver Assistance Systems* [8, 50].

Videos contains 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 a few minutes of the video if the content is desirable or not which results in waste of time, bandwidth and money.

There exist many content searching techniques in other parts of multimedia, 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) are available in the internet for their costumers to get an impression of the content. We have also 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 needs often more time as it saves.

The "normal" way to get information about the video content is viewing trailers and skip 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 implies the availability of the content and an abstract. An alternate possibility is to read an abstract of the video which is sometimes emphasized with images of the described part of the video. A both cases is 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 which selection could be based on the key frames.

All these components to create an overview of an unknown video assumes that an usable abstraction with key frames of the video exists which describes the desired and expected content. Different purposes assumes different kind of abstracts which matches the aim of the abstraction to be created. A video trailer should sometimes create a curiosity for the video and sometimes it should create an abstract with all important parts of the video. These aims are not always identical.

For the creator of the abstraction is it important to select different abstraction levels to have the possibility to select the desired parts of the content to create an abstract that matches the expectation of the abstraction.

An other example application for video key frames is the representation for a video search result like it is done by the video engine of altavista [3] which could be found at *http://www.altavista.com/video/default*. Altavistas search

engine provides a search for videos by entering key words which are associated to videos on website. A semantic search engine searches for these key words and presents the found videos by representing a frame of video[1]. It will happen frequently by this ineffective presentation of video content that the searcher does not find the needed information because a wrong, ineffective or insufficient video content is presented. Either someone need many attempts or he/she loads also video which *could* be useful. These results or often poor because the content could be everything.

It would be more effective for the seeker to get a list of key frames of the video presented to get a better impression of the expected content of the video. Figure 1.1 shows the resulting frame as originally presented by altavista for a search on personal security. It is not apparent what exact the topics of the video are. Someone could get the impression that the video only has ear protection information. Figure 1.2 explains that also other content like head protection and appropriate clothes are also necessary.

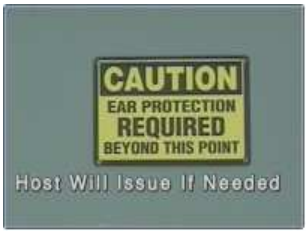

Figure 1.1: Original example frame for a short video introducing visitor protection.

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

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

1.3 Classical temporal video segmentation and indexing algorithms

There exist many techniques for key frame extraction. Most of them first segment videos into shots. Shot or video scene change detection is a well-studied topic [2, 25, 31]. These algorithms are based on fast and large changes between successor frames, but also gradient transitions between shots are detectable with additional analysis of the found shots [25, 26]. Such shots are represented by frames inside these shots. This could a simple process which is implemented into the detection algorithm or by an additional process that analyses single shots and picks more or less intelligent a representative frame. The quality of these detection algorithms rises and falls with the ability to detect and separate shots which is based on the comparison of frames in a local context. Higher abstraction levels are found by merging separate shots together to groups [53]. This process need a higher processing overhead, due to the fact that here shots are compared. Also depends this 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 joins single frames to frame groups which will result in groups of individual frames which are representative for the video sequence [45, 22].

The first category traditional algorithms for temporal video segmentation and indexing for lower abstraction levels are based on shot and cut detection [31, 52] and for the higher abstraction level on scene detection [2, 53].

The low level algorithms are mostly based on three steps; The extraction of image features, the calculation of neighbor image differences and the detection of shot boundaries. In an abstraction process are these detected shots 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 likely to other frames in the shot).

The second category are clustering algorithms [45].

Algorithms for higher abstraction levels are joining shots together to get higher abstraction levels. These algorithms need more information about the content of the scenes.

1.3.1 Algorithm of Kumar et. al

Rajeev Kumar and Vijay Devatha describes in their manuscript "A Statistical Approach to Robust Video Temporal Segmentation" an algorithm for video shot boundary detection [31]. They uses a key frame detection which is based on a shot detection algorithm with frame descriptors based a subdivided weighted grey color histogram. The pixel colors of the histogram are in such a way that pixels in the inner areas are higher and outer areas are lower weighted to show the importance of the inner parts of the viewable area, where 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 detect differences in two images, based on the histograms. They used the Bhattacharrya metric [5], which is a generalized χ^2 measure and is defined as the sum of the dot product of all histogram bins.

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 a approximated interval should not be approximated by the polynomial curve. Also polynomial curves introduce false minima near these points. This does not happen with B-Spline Curves. The real shot detection is based on the minima of the B-Spline curve which are potential candidates for a detected new shot. The similarity curve should fall below $T_h = \mu - \alpha \cdot \sigma$ where μ and σ are 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 [25]. The features are DC coefficient based histogram from the intra frame of a MPEG-video stream. With the gauss distance between adjacent intra frames and a dynamic generated threshold are shots detected. After a new detected shot are the previous P- and B-frames searched to refine the cut. With a statistical algorithm is detected if inside a shot gradual transition between shots exists, which will result in a new detected shot

boundary. For simplification reasons are key frames for shots represented by the 10th frame of a shot. The paper goes into the problem of grouping shots together to get a higher layer of representing video context. Spatial shot clustering algorithm groups visual 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 kind 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 which contains the a refinement of the content. The creation of the different abstraction levels is based on a merge process which joins shots together to groups of shots. Different comparison algorithms are used between single frames, a frame and a group of frames and between different frame groups. An entropy based threshold technique as used to define a dynamic (adaptive) threshold 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 the two dimensional polygon line by successive removing vertices from the polygon [36]. 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. has used this algorithm to segment videos [21, 33, 13]. This work follows the idea and concept as presented by Daniel DeMenthon.

It describes a video as a sequences by polygon trajectories with frame descriptors. Frames which are nearly equal to there neighborhood are found and gradual eliminated from the polygon. The algorithm does not reduce the key frame detection to a shot or gradual change detection as it is normally done. It allows applications to implement an own definition of importance by implementing the a relevance measure for frames depending on the aim of the key frame detection. We test the algorithm with many different classes of videos and shows improvements and refinement on the selection and implementation of the centroid based frame descriptors features which describes the frames and the temporal order of the frames. The improvements includes

the implementation of different kind of filters which are applied to the descriptors of a single frame and temporal in relation to neighbor frames. The refinement include the number of frame describing features as well the selecting of color space and the class of features. New at this place is also the usage of dominant colors in the discrete curve evolution.

Chapter 2 contains fundamental information about the used algorithms like the discrete curve evolution, feature filtering, the quality measurement and the video terminology. Chapter 3 absorbed the term *video key frames* and the requirements of a detection algorithm. Chapter 4 contains more considerations about *key frames*, the problems and requirements. The also implies the implementation of these requirements in the *discrete curve evolution* process. Chapter 4 contains improvements for the discrete curve evolution implementation as made by Daniel deMenthon. This includes the selection, the weighting and filtering of the frame descriptors. We compare different color spaces and features based on dominant colors and an optimal color composition. The experiments are continued in chapter 5. In chapter 6 we compare our algorithm with actual algorithms and results with third parties. 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 which is introduced in chapter 7. This is useful for real time key frame detection. Our algorithm does not only detect fast movements in sequence but also smooth changes inside the window. Chapter 8 contains a summary about the presented information and experiments and a conclusion of the work in this paper. The appendixes contains additional information about the used videos and applications. Appendix A contains an abstract about the videos used in this paper with background facts like duration, number of scenes and the expected ground truth result. Appendix B contains a description of the applications I have used and written to perform the experiments. This includes the free available MPEG1-player of the Berkeley university [39] which is used to create the frame descriptors and to extract the calculated key frames. We have also developed a helpful tool which joins the key frames at different abstraction levels and the video into a remote control and navigation tool from which it is easy to to navigate inside the video at different abstraction levels.

The work is partially based on previous published work [32, 35].

Chapter 2

Fundamental

The following sections of this chapter describes the fundamental definition and descriptions of the video processing terminology which is used in this paper. Also described are the mathematical basics behind the algorithms, the filters and the quality measurements.

2.1 Discrete Curve Evolution

The advantages of the discrete curve evolution are it's flexibility, the speed performance and robustness. The basic work is based on polygon simplification algorithm which simplifies a polygon by an iterative process which is called the *Discrete Curve Evolution*. In this iterative process are vertices of the polygon removed which are mostly constant in it's *local context*. 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.

The first appliance of the discrete curve evolution was in the context of shape similarity of planar objects [37]. Figure 2.1 shows a few stages of the evolution applied on a edge of a fish drawing. Notice that the most relevant vertices of the curve and the general shape of the picture are preserved even as most of the vertices have been removed.

In the following subsections are the different parts of the discrete curve evolution discussed. In section 2.1.1 is the algorithm described and in section 2.1.2 is the elimination criteria discussed. The application of the DCE as a key frame detection algorithm is discussed in chapter 3.

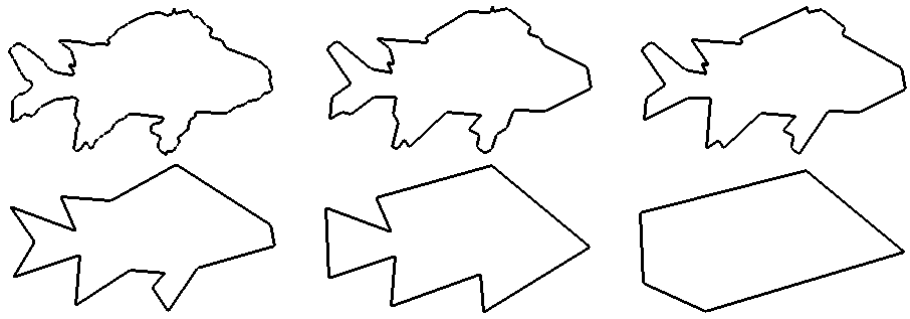

Figure 2.1: 6 stages of the discrete curve evolution

2.1.1 Evolution Process

Definition: Discrete Curve Evolution:
Let P be a (not necessary closed) endless polygon $P = (v_0, ..., v_n) \subset \mathbb{R}^m$ with $n \leq \|\{v|v \in vertices, \mathcal{C}(v)\text{is defined}\}\|$. The *discrete curve evolution* is a sequence of polylines

$\wp = (P = P^0, ..., P^m)$,
$|Vertices(P^m)| \geq \|\{v|v \in vertices, \mathcal{C}(v)\text{is defined}\}\|$,
where$|\ .\ |$is the cardinality function

with $P^{i+1} \subset P^i$ where $P^{i+1} = \mathcal{A}(P^i)$. $\mathcal{A}$ is the best abstraction.

Each vertex v_j in P^i (except the first and the last) is assigned a relevance measure K which is defined on v and its neighbor vertices in P^i.

The algorithm behind the discrete curve evolution could be defined by

Definition: Discrete Curve Evolution Algorithm
k=0;
while $|Vertices(P^k)| \geq \|\{v|v \in vertices, \mathcal{C}(v)\text{is defined}\}\|$
Find p_i: $\mathcal{C}_{P^k}(p_i) = min_j\{\mathcal{C}_{P^k}(p_j), p_j \in P^k\}$
$P^{k+1} := P^k \mid_{p_i}$
increase k by one
repeat

$\mathcal{C}$ is the *cost function* which measures the *relevance* of a frame to it's neighborhood.

The remaining frames are defined as the most important frames. For ordering purposes is the first of these frame the most important and the last frame

is lesser important frame. We didn't test other boundary frame but other possible boundary settings are

1. Linking the start and end frame together, so the polygon is a closed polygon.
2. Add additional frames to the beginning and the end of the video. This could be empty (black) or duplicate frames. These additional frames should not be a part of the resulting key frame set.

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.

We need, to perform a curve evolution process, a polygon and a cost function. One of our requirements in chapter 3.1.1 is a symbolic description of the video frames. The polygon could be represented by the numerical values of the symbolic frames descriptors. These vertices are linked to each in order of the represented frames.

comment: The original *polygon* was defined on $\mathbb{Z}^2$ as a closed polygon. We uses an open polygon in $\mathbb{R}^n$. The start and end vertices are fixed and not merged together.

2.1.2 Cost Function

The *cost function* is performed for every step of the curve evolution process for the whole new polygon P^k. After each step changes the value of the cost function for every vertice which neighborhood also contained the removed old vertice.

In the original is the cost function $\mathcal{C}$ defined on only the vertice and both neighbor vertices. In our case defines the cost function the content in relation to the local context, which is the neighborhood of the vertice. An better understand of the information content includes a look of view on a bigger context and so the definition of the cost function on more neighbor vertices is necessary. For this reason are considerations necessary to define a cost function on more then three frames.

We will see in the further chapters that a larger size of the local context could be used by only defining the cost function on the direct neighbor vertices and by adding an additional preprocessing step.

2.2 Video processing terminology

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

- A *frame* 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 could also other subsets of $\mathbb{R}^n$ used for the domain space. $f(x, y) \rightarrow \overline{\text{color}(x, y)}$
- A *frame* is a video image 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 in a time unit like seconds or an abstract time unit like frame number (in which case t would be integer).
- A *video* is a temporal ordered sequence of frames. $Video = \{f_1, ..., f_n\}$
- 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[1]. This will varying for different kind video recoding media and standards (like cinema, DVD, television, PAL, NTSC, digital photo camera or video camera). Normally are frame rates between 25 and 30 fps. Digital photo cameras uses sometimes a rate of 15 fps and webcams sometimes less than 1 fps.
- The *frame or video resolution* is the resolution of a single frame in pixel's width and pixel's height. This could varying on the video standard.
- The *resolution ratio* is the ratio of the resolution width:height. Normally is this for a television 3:4 (0.75). Cinematic films has normally ratios higher as 1.5 (like 1.85 or 2.35). The films we have used has often the same ratio as used for television. The most common resolutions[2] are 320×240 or 160×120.

[1] The measure unit is *fps* for frames per second

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

- A *pan* is either a horizontal move of the camera or a rotation in the vertical axis. For example is the scenery is recorded from left to right from right to the left.
- A *tilt* is a rotation of the camera around the axis in the viewing direction.
- A *shot* is the smallest video unit with more than one frame and 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 between two shots least at less for one frame between the shotF frames. There exists different kind of transitions. A *gradient transition* is a linear switch between two shots resulting a blending between these 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. Often [25] a *key frame* is defined as a representative frame of a single shot.

The reduction of *key frames* to a single representative frame of a *shot* will reduce the meaning and the abstraction functionality of a shot to the content of a single shots. There exists many various kind of examples that shows that it is not possible to reduce a single shot to one frame. By the association of key frames to shots is automatically the abstraction level of the video predefined and there is no way to modify this level. Even if it is needed.

I agree that there are many applications for which a reduction of shots to a single frame will be correct. But there will exist at least even more applications for which this will 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 an approximation of the original source by the remaining frames. The expectation for this approximation is very simple. It should be not too bad. The approximation

must be good enough and should describe the content of the video. These last two phrases could be interpreted as

1. The remaining frames must match the *content* of the original frames as good as possible as expected by the observer.

2. The information reduction should between an expected minimum and maximum level. If the number of frames is below the minimum number, then the information reduction is to high and a number above the maximum will result in reducible information.

Which are the key frames?
If someone asks this question to somebody, then the answers will varying like "The content must be described by the key frames" or "The key frames should be a summary of the video". Also the answers if a set of key frames represents will match their expectation are varying from "Yes, but that frame is missing" or "No, that information is not important for me".

Different people will expect for the same content different key frame sets as a result of how they have for them self defined the *best approximation* for the given content. It is possible that depending on the mood, the time and the situation of the same person the same video will be described by different *key frames*.

Consequence:
The consequence is that we have an reduction and abstraction level of the video (which will be reflected by the key frames), that depends on the view of an observer and it's expectation which will also depend on the application in which the reduction is needed.

A given set of key frames will result in a fix definition of the application and also the role of the observing user. The quality range of acceptable key frames will be more or less vague. Only this observer could define the expected abstraction level which results are suitable for his application. The quality of key frames rises and falls with the correct numerical definition of key frames and abstraction level.

This consideration results in the fact that there will be no suitable definition of key frames and abstraction level for every application. Key frames will only be available if this unknown information of the application is given. There are two questions that should be answered before we could give a key frame result for a given video.

1. In what kind of information are we interested?

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

For example: These kind of problems also exists for other domains 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 [49] are learning algorithms. Frames are compared on base of image descriptors. The learning process in- or decreases the weighting of the descriptors by user interaction to best match the expected result. This results in a description of which image descriptors are important which reflects in the expected information of the user. These kind of algorithms should lead to good results if the image descriptors are suitable for the class of information that is expected in the application. There will also be a value for the equality between frames at which the results are not good enough to match the users expectation. This value is a description of the reduction factor which fulfills the application for which the search was performed.

These two open points in the definition of key frames has great demands on the key frame detection algorithm. It should be flexible to even fulfill the definition of important information as the possibility to define an abstraction level.

Due to the fact, that we could not define a general suitable definition of key frames, we try to define a general information description of key frames, to reach a wide class of applications. It is not our intend to find a suitable algorithm to fulfill any application as mentioned in chapter 3 and match any expected result for any situation.

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 of the context in which this frame appears. Changes in this local context will reflect in changes of the key frames. These changes are possible by one of the following reasons which could be interpreted as request to the definition of key frames.

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

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

The definition behind 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 of these results are *good*, *representative*, *complete* or even *suitable*. There exists several [6] measurements that tries to describes 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 match exactly the key frames as expected by the key frame definition. This includes either the 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 requirement 2 of section 2.2.1.

The experimental results are compared to the ground truth results and with statistical information about those comparisons is the quality of the experimental result measured.

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

The ground truth results are self with a hand held camera 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 objects (this is a noise in the object motion in the time) and also we have a background noise in frame colors self. These videos could be found in the appendix A.

The compared experiments are created by other people. They are also often self recorded videos, where it is not directly obvious which key frames are expected. Sometimes is a trailer or a mixture of cuts used as a comparison result. These videos could be found in the appendix A.2. A small set of the videos contains a set of *ground truth results* as they are selective defined by a larger group of people [22].

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 for 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 for 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 number of correct and false detected key frames.

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

.

Chapter 3

Key Frame Detection

We have seen in the previous chapter that the definition and also the detection of key frames is not trivial. In this chapter we will discuss 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 reaches from the classical shot representation over to the detection of different but important information inside a shot. Also non linear changes like unpredictable events changes in action inside shots like the starting or stopping of a car should be detected because these kind of information could also important and should be detected.

Our requirements on a key frame detection algorithm are

1. **Shot detection**
   A change of the local context is possible by a cut or a blending like gradient transition between shots. Such a shot and cut detection is a well studied topic [4, 25]. Also it is no problem to find key frames for different shots with and without blendings.

2. **Change in content**
   It is possible that one shot contains different kind of information because on of the background or foreground objects changes. This is

possible by objects which enters the area of view or objects which hide or does not hide an other object. For example a 360 degree panorama view of a beach will show a beach and a few seconds later the beach or something else in the background like a city or mountains. The content of this panorama view will not be representable by one key frame. The examples from 1 tries to represent the detected shots by a single frame which is sometimes a random selected.

3. **Event detection**
   Also a change in the "action" of a scene could 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 representated by key frames.

1 and 2 reflects 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 the algorithm concept and the fact that a shot has two boundary shots, these changes are relative and not local practicable.

3 reflects a change in either in the *local context* as between the frames directly. The local context for these events is represented by changes in the linear difference between the potential key frame and it's previous and following frame. For example: In the ideal case of a stopping object are the potential key frame and the following frame identical (no difference) but the potential key frame and previous frame are different.

3.1.1 Requirements on a frame comparison algorithm

The key frame requirements above has some influence on the comparison between frames in a video. A direct one to one comparison between frames which is done like in [25, 31] is not possible, because this will not fulfill the requirement 3 which need more information of the local context in which the frame appears. Either we need frame descriptors which reflect this local content or we need a frame criteria measurement which considers this. I took the decision to only define frame descriptors for the frame self without considering the local context. This is done to make the algorithm as general as possible for existing frame descriptors which exists 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. How better the FD matches our expectation how 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. How better we are be able to compare frames, how better our results will be.

3. **Context Dependency**
   The comparison between single frames does not consider the *local context*, so we do not need a frame to frame similarity, but a similarity of a frame to it's *local context*. This comparison will be defined later, but it will be based on a frame to frame comparison. So 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 depends on the frame comparison.

The first two requirements 1 and 2 are an important element in the *CBIR*. Considerations from that part of the video signal processing should be also important for our key frame results, as far as they match our exceptions of the key frames.

The consequence of requirement 3 is

1. **Usage of local context**
   Our key frame definition should notice the *local context* as it include the content of the surrounding frames for it's relevance calculation. The simplest kind of such a context would be the previous and following frame.

2. **Detection of non linear temporal changes**
   A key frame selection should be based on the similarity of previous and following frames and a change in this similarity.

3.2 Image and Video Descriptors

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

But that shows how important the selection of correct frame descriptors would be. The quality of the frames feature descriptors has a direct influence on the quality of the algorithm which are based on these descriptors. The fine art is it to reduce the number of needed descriptors without reducing the amount of information in it.

It is important to know, based on the defined expectations, which kind of information should be contained in the descriptors. For example

1. If the position of objects has an influence an the detection ability, than it is logical that some of the position depending information must be included in our frame descriptors. Such information could be represented by different data classes. This could be direct by coordinates of objects, but also indirect by weighting different kind of (non position) depending descriptors, based on the area in which they appears. For example will a histogram based on colors weighted by a factor which depends on the area in which they appears be different if some parts of the image are per mutated.

2. Speed changes are reflected by a acceleration or deceleration which reflects a non linear position change of an object over time. This includes that time and position depending information must be included in the frame descriptors, to be able to detect such kind of events. The time factor could be directly represented by the time of the frame in the video or by it's frame number. The time factor could also be indirectly stored in terms like speed factor, or motion vectors as used in MPEG video stream B- and P-frames.

As we see the available mount of information could be represented by a varying 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 contains information about the frame inside the video. This could be the mentioned time information, but also information about the other frames in the video. An idea is to detect if the importance of frames descriptors should be risen, if they are non constant in the context. With the amount of information, also the amount of possibilities to get the information, storing the information and the processing of the information will be increased. This will result in a confused results which leads towards a reduction of the information to an easily comprehensive number of descriptors without an important lost of necessary information. There exists much more kind of information inside frames, like

- Texture information which could be represented by the relation between the lightest and darkest pixel's inside different sized areas around a single point [53]. Several kind of texture information descriptors are described in [24].
- Different color spaces like $\{R, G, B\}$, $\{Y, U, V\}$, $\{L, a*, b*\}$
- Color moments like the centroid coordinates.
- Motion vectors of frame components like used in MPEG video streams.
- Hidden Markov Model
- Training based feature weighting

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

3.2.1 Buckets

An often [11, 52] used feature in frames are histograms. Daniel de Menthon et. al. has developed histogram based features called *buckets* [11]. Instead of the histogram self, are histogram based centroid used as features. The histogram of each Y-, U- and V-color component with a range from 0 to 255 is subdivided in four bins with a linear range of 64 colors. The pixels with the color from inside such a bin representing a centroid.

The coordinates x, y and *area* of the centroid is used are used as the feature. This is done for every color component and every histogram bin. So, each bucket contains the features of a centroid based on the pixels which values are inside of a single histogram bin. The only needful 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 $3colors \times 4\frac{intervals}{color} \times 3\frac{features}{interval} + 1(time)feature = 37features$. These features are represented by a 37 dimensional vector which represents the information content of our frames: $f_t \rightarrow \mathbb{R}^{37}$

It could be sacrificed to implement an complex object detection algorithms like image segmentation. It is expected, that the influence of an object on the associated centroid is large enough to be detected and processed.

3.2.2 Dominant Colors

An other and more human intuitive feature are 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 comparison algorithm are more based on the human vision model, so the results for a frame to frame comparison will faster match our expected results, if we compare two frames. (This easiest reason to understand this, is a cut detection algorithm, which is only based on a frame-to-frame comparison algorithm. If the human vision detects a large difference, then also the algorithm should 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. The descriptors contains the code book color number and it's procentual 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 describes the *cost function* 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, when the movement of the car is linear, without lost of information, that the car drives over a and c that it then also should past point b. The information, that the car will past point b is redundant. The cost function in point 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 in relation to the neighbor frames is.

The considerations of the previous chapters leads to the following consequences.

$$f_i \in f_1, ..., f_n; n = \#\text{frames in video} \tag{3.1}$$

The local context Loc_c is given by the neighbor frames of a specific frames. This *neighborhood* is defined by a constant c.

$$Loc_c(f_i) := \{f_j \mid f_j \in video\ frames \wedge \|i - j\| < c\} \tag{3.2}$$

The searched relevance measure M_{rel} is defined on the frame f_i and it's local context $Loc_c(f_i)$.

$$M_{rel}(f_i, Loc_c(f_i)) \in \mathbb{R} \tag{3.3}$$

We call such a measure a "Relevance Measure" for a given "Key Frame Definition" if it satisfies

1. If f_i is more *similar* to $Loc_c(f_i)$ then f_j's *similarity* to $Loc_c(f_j)$, then $M_{rel}(f_i, Loc_c(f_i)) \leq M_{rel}(f_j, Loc_c(f_j))$
2. The "similarity" term above, should match our expectation of the *Key Frame Definition*.

M_{rel} is not necessary 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 implements, that abrupt changes to at least one neighbor frame should be detected, and slow changes or nearly equal

frames should be not detected. For example, if frames f_t is nearly equal to f_{t+1} and f_{t+1} is very different to f_{t+2}, then

$$M_{rel}(f_{t+1}, \{f_t, f_{t+1}\}) < M_{rel}(f_{t+1}, \{f_{t+1}, f_{t+2}\}) \tag{3.4}$$

The requirement to detect non linear events is a little bit more difficult to describe. For example, it is expected, that if an object moves linear from frame f_t over frame f_{t+1} to frame f_{t+2}. In frame f_{t+2} appears an event; The object travels with an other speed, it moves slower over frame f_{t+3} to frame f_{t+4}. It is expected, that

$$M_{rel}(f_{t+1}, \{f_t, f_{t+1}, f_{t+2}\}) < M_{rel}(f_{t+2}, \{f_{t+1}, f_{t+2}, f_{t+3}\}) \tag{3.5}$$

$$M_{rel}(f_{t+2}, \{f_{t+1}, f_{t+2}, f_{t+3}\}) > M_{rel}(f_{t+3}, \{f_{t+2}, f_{t+3}, f_{t+4}\}) \tag{3.6}$$

The frame in which the events occurs is expected to be more relevant then in the other frames.

3.3.1 Image comparison

Step 2 could be the most important requirement because it defines how near our results would be to the expected key frame definition. Also step 2 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 to got an equality measurement 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 $d(x, z) > d(x, y) + d(y, z)$

1 means that two identical frames should have no differences and also a distance from 0. 2 means that the distance between two different frames is positive definite. 3 means that it makes no sense 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, so we need an other measurement. The easiest way to define such a measure is to use the metric to define this.

Even with the limitation of formulae 3.4 applied to the metric is this not possible. *If frame f_t is nearly equal to f_{t+1} and f_{t+1} is very different to f_{t+2}, then is*

$$d(f_{t+1}, \{f_t, f_{t+1}\}) < d(f_{t+1}, \{f_{t+1}, f_{t+2}\}) \tag{3.7}$$

Formulaes 3.5 and 3.6 are not possible, due to the definition of M_{rel} on a *local context*. At less a (in terms of time) equidistant previous and a successor frame are necessary to detect changes in a linear change. So M_{rel} should be defined on at least three frames.

3.3.2 Examples

Candidate "Relevance Measure" for a Loc_1 could be

1. $M_{rel}(f_1, \{f_0, f_1, f_2\}) := d(f_0, f_1) + d(f_1, f_2)$
   This measure does not achieve 3)

2. $M_{rel}(f_1, \{f_0, f_1, f_2\}) := d(f_0, f_1) + d(f_1, f_2) - d(f_0, f_2)$
   This measure does achieve 3)

3. $M_{rel}(f_1, \{f_0, f_1, f_2\}) := |d(f_0, f_1) - d(f_1, f_2)|$
   This measure does achieve 3)

with neighbor frames a,c as the local context of frame b. The relevance value defines the importance of a single frame b in relation to it's neighbor frames.

3.4 Conclusion

The frame descriptor selection, based on histogram centroids are very suitable for our expected key frames. It contains all necessary information that is used in our key frame definition. The *discrete curve evolution* is only useful because the relevance measurement of the frames weights exactly those frames which matches our key frame definition.

Only an information description by frame descriptors and an information analyzes by the relevance measure is not a guarantee for a good key frame extraction. Every kind of information which is included in all videos and video frames[1] are compressed into $\mathbb{R}^{37}$. Everyone will see that such a compression will also leads to a reduction of information. Also an important question is how stabile are the features. Will they easily disturbed by small changes in the frame like a little bit more or less brightness? What happens of the frames are scaled? If the different kind of information which occurs in the "real" is good enough separated in this 37 dimensional space and if the contained information could also be interpreted by the detection algorithm will be tested and verified by experiments in the next chapters.

[1] If a frame contains $x \times y$ pixel's 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

Chapter 4

Closed Videos

As we have seen in the previous chapter has Daniel DeMenthon used a 37 dimensional feature vector based on video frames to describe the content of a closed video. In this chapter are the used algorithms reviewed and improvements for presented.

The key frame detection is splitted in 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*. The third step is the extraction of the key frames from the video.

4.1 Abstract review of existing applications

The applications used by Daniel DeMenthon are written and developed by the LAMP division on the Maryland University (USA) [34]. The application *Merit* which is 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 for each frame. An MPEG-1 video stream stores data in DCT blocks of 8x8 pixel (This are the *macro blocks*). The most upper left values of such a DC block contains the average color intensity of the block, which is available for each the YUV color components. These (average) color information is used the calculate centroids which gives us the needed information. These data is stored in a DCT-File[1] which is used by the next application.

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

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

The histogram is subdivided 4 equidistant parts and for each part is a *bucket* (centroid data) calculated. From these buckets is for each frame f_t the frame descriptor frame calculated. The vector elements are ordered in by the colors (in this order) Y, U, V. Inside each color is the data stored from the lightest color intensity to the darker color intensity. The data are (in this order) the x and y value of the centroid in DCT block coordinates and the area in number of DCT blocks.

$$
\begin{array}{lll}
f_t \rightarrow FD(f_t) = (& t, & \\
& b^x_{Y_1}, \quad b^y_{Y_1}, \quad b^\#_{Y_1}, & \\
& \ldots, & \\
& b^x_{Y_4}, \quad b^y_{Y_4}, \quad b^\#_{Y_4}, & \\
& b^x_{U_1}, \quad b^y_{U_1}, \quad b^\#_{U_1}, & \\
& \ldots, & \\
& \ldots, & \\
& b^x_{V_4}, \quad b^y_{V_4}, \quad b^\#_{V_4} \quad)^T &
\end{array}
$$

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, which only the EVO-file and the TOL-file are mentioned here. The relevance measure is the in chapter 3.3 developed

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

The used metric is the euclidean metric

$d(FD^1, FD^2) := \|FD^1 - FD^2\|_2$

The EVO-file contains frames in order when they are removed from the polygon line. Additional information stored with the frame is 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.

The extraction of the key frames is done with a *modified MPEG player* from the Berkeley University [39], which is written in C. This MPEG-player is freely spread with several Linux distributions and could also downloaded from the internet. The modifications includes an option to read a list with

frame numbers from a file which are extracted and stored the PPM-format which is a well known file format on UNIX based computer operating systems. This frame list created from the EVO- and the TOL-file.

The list of applications is finished with a JAVA-applet which joins all files to video viewer. The viewer could be used as a standalone application or as client application in a browser like Firefox [41] or Internet Explorer. The application has 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 represent the video. With the mouse each key frame selected and is then shown in the viewer.

4.2 Algorithm and software analysis

We tested these functionality on some small test videos [16, 15, 17]. These results are from a good quality, so the decision to make more tests to improve the curve evolution, features and functionality of the applications was a good idea. We tested the algorithms and programs on other videos and scenes to test the applicability for different kind of videos.

4.2.1 Feature Extraction

The existing features are all based in the average color in a macro block. This automatic implements a filter as the average color could have nothing to do with the existing pixel's and a precise analyze is not possible due the important lost of information. For example, if the macro block contains 32 black and 32 white pixel's, then the average color would be grey. These three color's resists in three different parts of the histogram. It could be possible that our results will be disturbed. Also an eventually analyze based on the texture is not possible, because those fine contours does not exist in this part of the macro blocks.

Due the nature of macro blocks, which are **always** a block with a size of 8 pixel's, is it possible that the dimensions of a frame is not a multiple of 8. In that case are either parts of the frame removed or add or stretched, in all cases is the processed information not correct.

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

Also the time factor (frame number) is not scaling invariant if we have an other frame rate.

Feature normalization

The idea behind these features is good because it gives us necessary information about objects (which should have a measurable contrast to the background and which is expected to exists in an other centroid). 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 makes it difficult to compare and analyze the results for our videos. This will for example directly effect different scaled videos which *could* not give the same results for different sizes.

It is necessary to scale the features into a well defined domain before we could make tests. As we have seen in 4.2.1 could information be lost due the fact that features depends one MPEG1-macro blocks instead of the pixel's of the image. It would be an improvement if we will not use the macro blocks as a base of the features but the pixel's self. A lost of information could be surpressed and other video sources that does not have average color information based on macro blocks could also be used. The disadvantage as a 64 times more information that must be processed because instead of one feature for each 8x8 sized macro block we have 64 features (for each pixel one). This could be surpressed by an improvement of the computer speed and the simplification of the bucket filling process. This part of the analyzing process does not importantly increase the whole processing time. The more time for the processing is linear ($O(h)$).

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 it could be represented by a linear normalization of the origin intervals. Different importances of features could be realized by different weighting's in the image comparison calculation.

The centroid coordinate x is scaled from $x \in [0, x_{max}]$ to $x_{scaled} \in [0, 1]$. This is done by dividing x through x_{max}.

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

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}.

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

The are is scaled from $Area \in [0, x_{max}y_{max}]$ to $Area_{scaled} \in [0, 1]$. This is done by dividing $Area$ through $x_{max}y_{max}$.

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

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 occurs. If we take the decision to scale the time to a fix interval, we will have problems with cuts with the same frames of the video. If we have videos with different frame rates, we also get problems if we not scale the time to a clearly defined interval. We scaled the number of frames of one second to the interval $[0, 1]$.

An other problem with the time value is the weighting. How important should the time factor in relation to the other relevance factors of the video content? Also the relative importance of the time factor in relation to total number of the other relevance values should be constant. So, it makes sense to divide the time factor by the number of the other relevance factors.

$t_{framenumber} \rightarrow t_{framenumber}/(framerate \times 3 \times \#centroids)$

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

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

The algorithms are exacter because to the whole image is analyzed and not only the macro blocks. The program access directly the (RGB/YUV) output frame buffer of the mpeg player. It was interest to conclude, that the results are nearly equal. So the algorithms are stabile.

4.2.2 Lost of information

It is not so clear and easy to detect the problems in not ground truth results, so we tried other improvements. As we mentioned before is the amount of information and the measureability of the changes in these information important. An increase of the amount of information coudld also increase the quality of the features and so the quality of the results. The amount of information which is used in the frame comparison algorithm has a direct influence on the it's quality. If in this processing to much information is lost, then we this will result in a bad comparison result. This could result in frames that are to easy detected as equal but in realty they are to unequal, which will result in removed frames that are more different as the remaining frames. The idea is to add more histogram subsets to increase the amount of information, but with got not to much unnecessary information.

Our idea is to increase the subdivision of the color histograms on which the centroid's 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 are a power of two. The original number of buckets used by Daniel DeMenthon is 4 [11]. As we have seen leads to a total amount of 37 feature 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. We tried to an histogram subdivision into

eight parts, what will result in 145 features descriptors (included the time feature) and with sixteen, which results 289 feature descriptors.

We made several experiments with a histogram subdivision of 4, 8 and 16 parts. The test video we used is the small version of "Mr. Beans Christmas" [48]. Through the increase of the number of centroid's are also the importance of a single centroid decreased even as the importance of the time factor. If the importance of the time should be constant by increasing number of features then the time factor should weighted by the same multiple of the number of increased features. (In our example a multiplication with two or our should be necessary[23].)

Other experiments are the ground truth results. For this example we used video "Mov1" as in the subsection before for this experiment. We also tested this video without the filters to get comparable results. It could be possible that through the color merging in to histogram parts more or less equal information is merged into different parts

As we can see in image 4.1, the real results of the best five images does not match the best five images of the expected ground truth result. Key frame number two is missing (which is leafed blank in the image set). In image 4.2 we see that the missing image is inserted as the sixth best image. The frame descriptors uses 8 bins (subdivisions) for each histogram color component to fill the buckets.

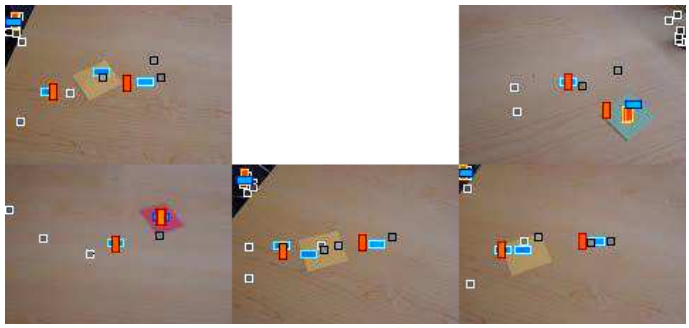

Figure 4.1: The best five key frames with 145 features for "MOV1"

We uses for the results of figure 4.1 16 bins for each color component. The results matches the expected ground truth results.

Results are available for video sequences "House Tour", "Kylie", "Mov1" and "mrbeantu". More information about these videos could be found in

[2]This fact is not mentioned in our applications.

[3]Also we have not tested a change in weighting the time factor at all.

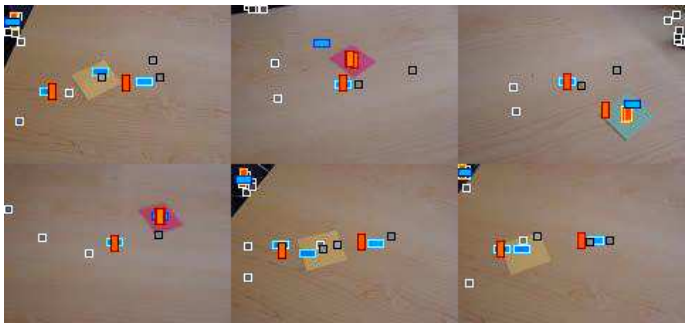

Figure 4.2: The best six key frames with 145 features for "MOV1"

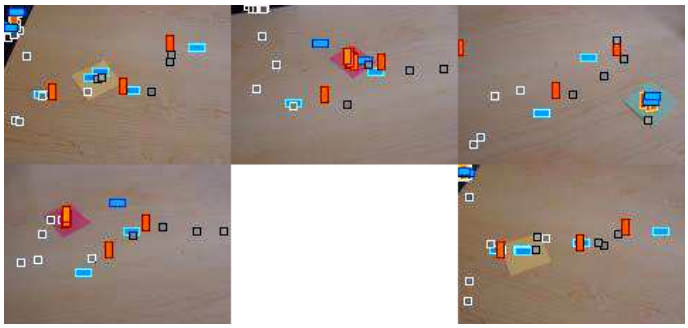

Figure 4.3: The best five key frames with 289 features for "MOV1"

appendix A and on the homepage [14]. These experiments are made without any filtering of the data sources. More information about the application versions is available in appendix B.

4.2.3 Problems with nearly empty centroids

An other problem are video sequences where "nothing" important are happening, but key frames are detected. For example "Mov1". We tested our applications in the default configurations on "Mov1". Figure 4.4 shows the expected ground truth result of this video[4].

We have drawn the frame descriptors in the frames to get a visual idea which frame descriptors exists and where they are located. The frame descriptor representation is done by drawing squares and rectangular's for the represented centroid's into the frames. The Luminance (Y) colors are represented by grey colored squares. The inner part of the square show the color intensity

[4]The videos and the expected ground truth results are described in appendix A.

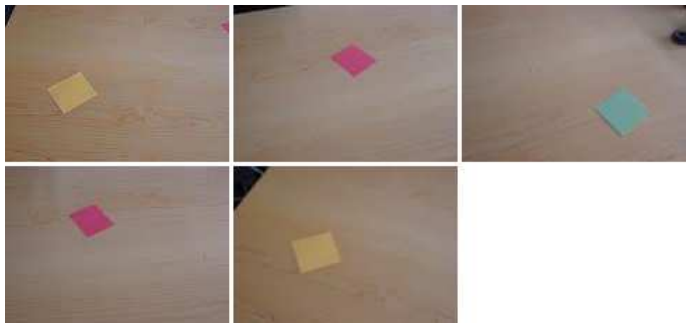

Figure 4.4: The five key frames of the ground truth video “Mov1”.

and the border black or white to get an acceptable contrast to improve the visibility. The position of the squares represent the position of the centroid's. The size of the centroid is not represented. The red chrominance (U or Cr) colors are represented by vertical red colored rectangles. As for the Luminance shows brightness of the inner part the intensity of the representing bucket. The border is also black or white to create a contrast. The same is done for the blue chrominance (V or Cb) which centroid's are represented by horizontal blue colored rectangles. The only not shown features 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.

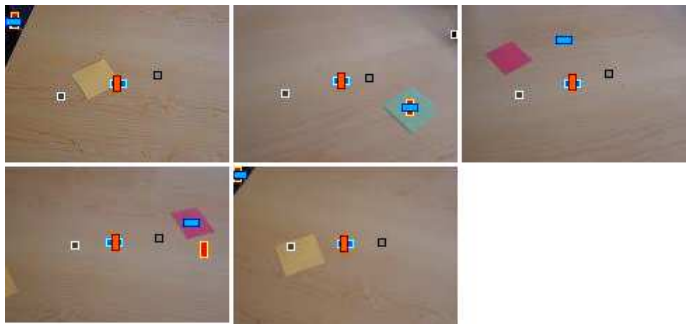

Figure 4.5: Resulting five *key frames* of our experiments with normalized features. This are frames 1, 197, 263, 313 and 378 of video “Mov1”.

As we see are some of the correct key frames missing and so we looked as much frames we the ground truth key frames are a subset of the detected key frames. 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,

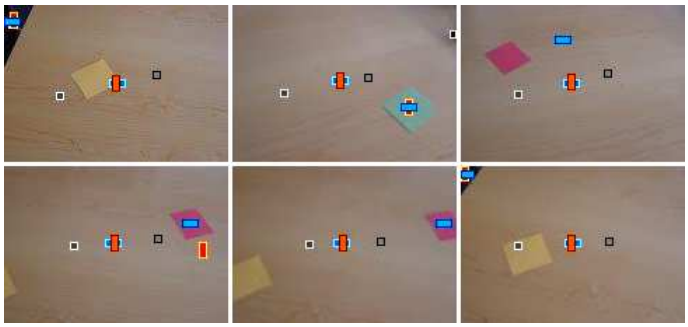

Figure 4.6: Best six frames 1, 197, 263, 313, 319 and 378 of video “Mov1”.

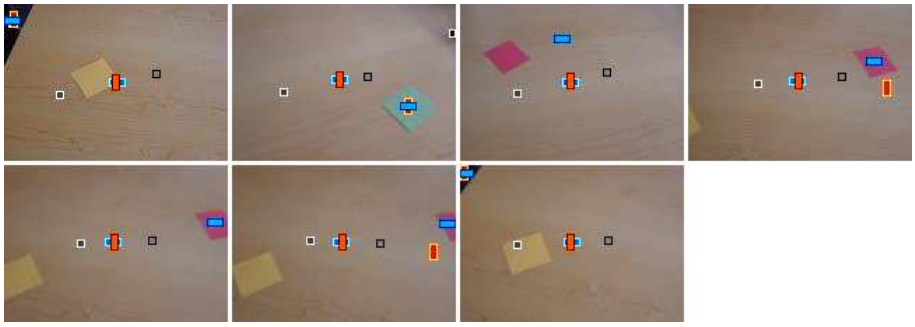

Figure 4.7: Best seven frames 1, 197, 263, 313, 319, 320 and 378 of video “Mov1”.

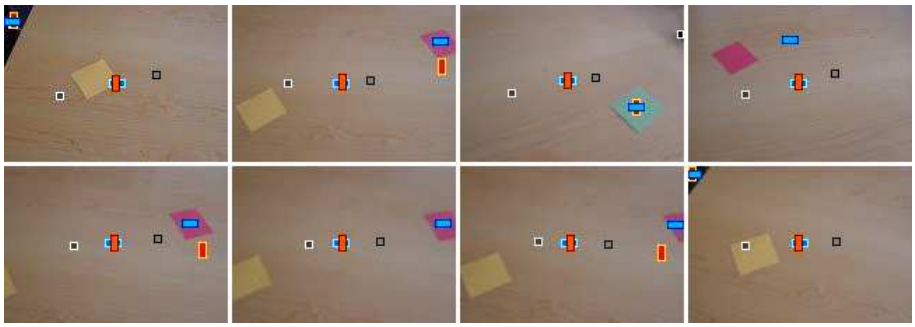

Figure 4.8: Best eight frames 1, 55, 197, 263, 313, 319, 320 and 378 of video “Mov1”.

because the seventh frame lies between those two frames and is different. What we not expected is that the seventh frame is nearly equal as two of the neighbored frames

The problem that we got here is why is the seventh frame nearly equal to two other frames? Also we observed this behavior with 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 we can see in figure 4.7. As we can see in the middle frame is at the right side a centroid missing which 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 approximates 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, which format is described 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 like 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.

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

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

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%), which will expect that the second bucket of frames 312 and 319 is not exact but nearly zero but 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 both.

The problem here is that the coordinates for not existent centroid's is undefined but handled as if they are zero, what is wrong. The first idea is either not to use this part of the centroid data if one of the involved centroid's does not exist or to set it equal to the compared centroid. But what happens if every frame has always one pixel at the lower left coordinate (nearly) (0,0) set? This will normally, depending on the frame scale, 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 halve 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 on. The real amount of information which is represented by the coordinates is based on the number of pixel's which builds the centroid. A better idea is to multiply the coordinates by size of the centroid. The difference between two less important centroid's will be always small. It is only possible that these values are large when at less one centroid is large enough. We called this coordinate multiplication *dynamic weighting* because the importance and so the weighting of the coordinates are dynamical modified.

The maximum possible X- and Y-range are halved, so the X- and Y-values should be doubled when it's 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 will features which are 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 (static). As we can see has we some important improvements

1. Fast "jumps" in the feature are completely eliminated. This multiplication acts as a kind of filter which flatten abrupt changes in pixel's which switches between different centroids.

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

3. As a result of the previous two points is "pixel noise" which will be reflected in the randomly added and removed of pixel's in centroid's also removed.

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

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

The in the *Discrete Curve Evolution* used feature vector is

$f_t \rightarrow FD(f_t) = (t, c_x^{\hat{Y_1}}, c_y^{\hat{Y_1}}, c_\#^{Y_1}, ..., c_x^{\hat{Y_4}}, c_y^{\hat{Y_4}}, c_\#^{Y_4}, c_x^{\hat{U_1}}, c_y^{\hat{U_1}}, c_\#^{U_1}, ..., ..., c_x^{\hat{V_4}}, c_y^{\hat{V_4}}, c_\#^{V_4})^T$

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

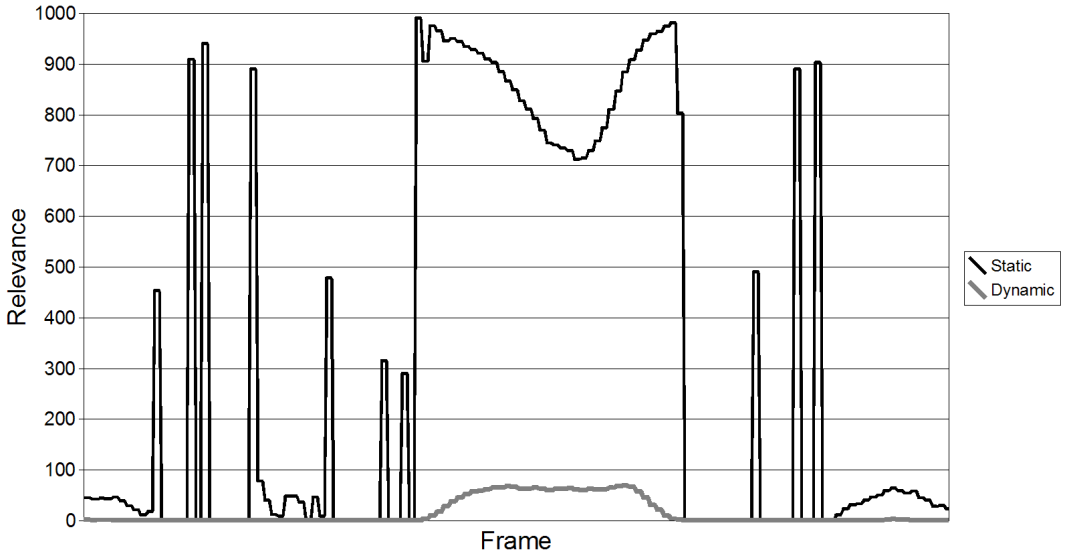

Figure 4.10: Comparison of the X-component of the second U-centroid of the video "Mov1".

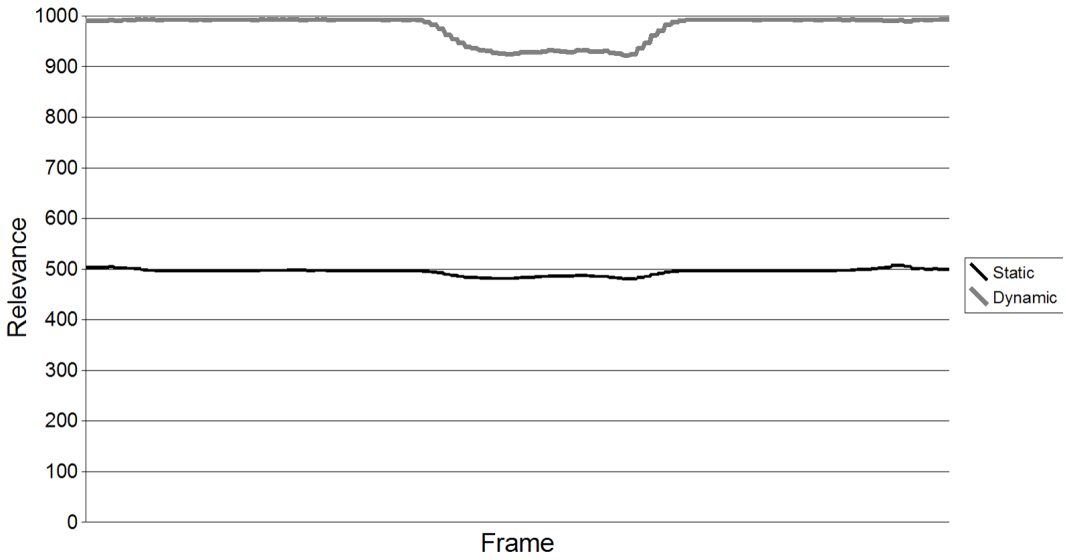

Figure 4.11: Comparison of the X-component of the third U-centroid of the video "Mov1".

The result is not very bad but also the fourth key frame is missing and the last key frame twice. What we see is that in the last key frame on the upper left the table edge with the black background appears. So we will have new buckets (as we can see on the drawn rectangles in the upper left).

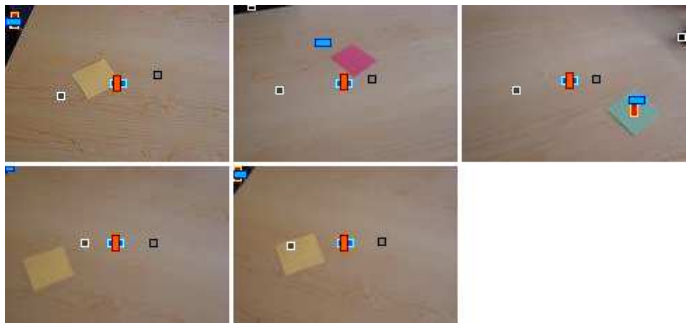

Figure 4.12: Best five frames 1, 100, 200, 328 and 378 of video “Mov1” with the weighting modification for the centroid coordinates.

The argument that the area is small and so they should not be important in the difference weighting will not be correct. That part of the frame is not very large, but more then only a *few* pixel's as in the examples above. The features are

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 are the differences more than only a few pixel's. What happen with six frames could we see in image 4.13.

This time the second key frame is doubled. Also this the difference is 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
0  0 0   0   0   0 496 495 1000 0 0 0
0  0 0 499 502 972 401 246   27 0 0 0
```

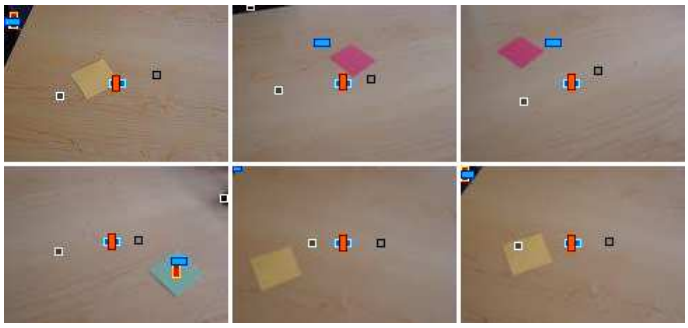

Figure 4.13: Best six frames 1, 100, 127, 200, 328 and 378 of video “Mov1” after the weighting modification.

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
```

With seven frames

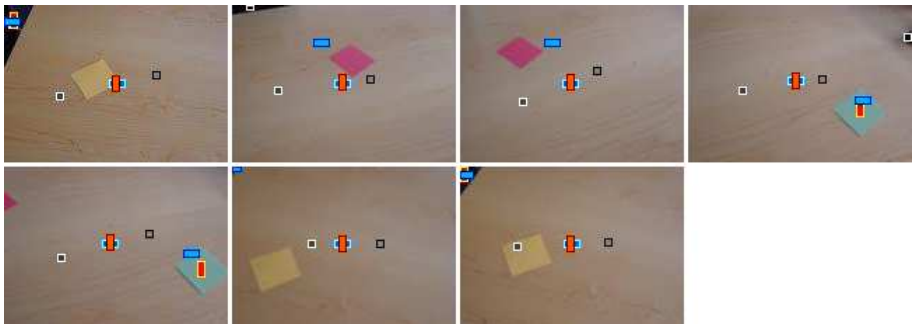

Figure 4.14: Best seven frames 1, 100, 127, 200, 236, 328 and 378 of video “Mov1” after the weighting modification.

With eight frames

As we see are our ground truth video not as perfect as we have hoped.

On the first look (see images 4.8 and 4.15) is seems that our improvement with an area weighted centroid coordinates is not much better as before. But on the second look we see improvements in the changes of the “wrong” images a kind of “quality” improvement. With a numerical analyze of the features and a visual analyze of the frames it looks like the key frame detection of

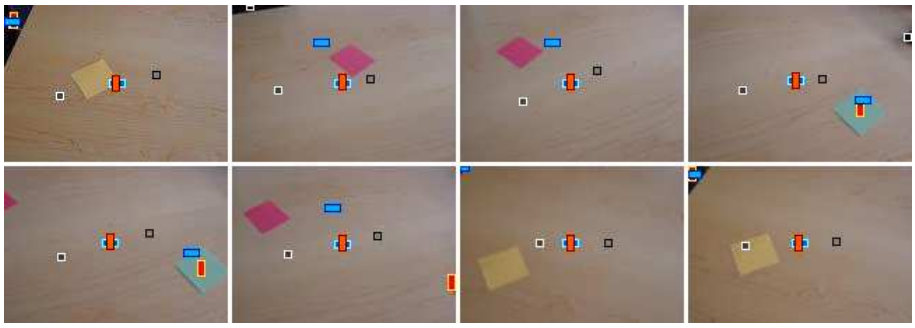

Figure 4.15: Best eight frames 1, 100, 127, 200, 236, 259, 328 and 378 of video “Mov1” after the weighting modification.

the improved algorithm is reasonable. The wrongly detected key frames has 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 could be found in appendix A and on the homepage [14].

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

Problem:
Our reflections for the dynamic area weighting has only an influence on temporal changes between frames, but it has no influence if larger homogeneous areas changes bins inside a single frame or between frames. Such areas will still abruptly changes the centroid data which will reflect in abrupt changes in the features without necessary a 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 should be for each possible pixel color equal to 1. At this moment is the weighting only for one predefined bucket one and for each other bucket 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 bucket when the color is changing. Figure 4.17 shows a suggested weighting for the colors and the associated buckets.

Conclusion:
Dynamic weighting of the X- and Y-coordinates of centroid’s could useful and an alternative of static filters. Objects moving (slowly) into or from

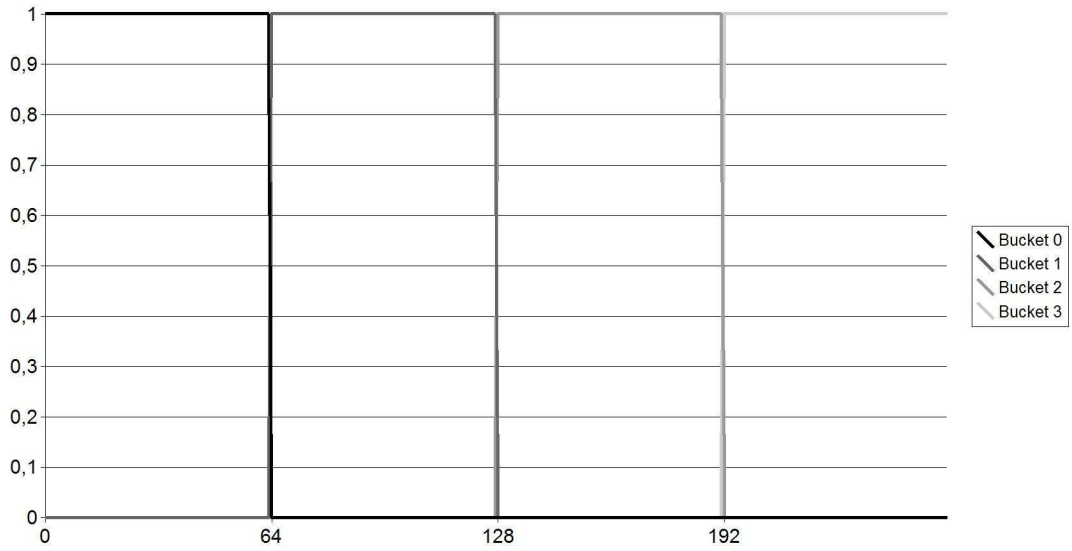

Figure 4.16: Weighting of pixels for associated centroids as implemented.

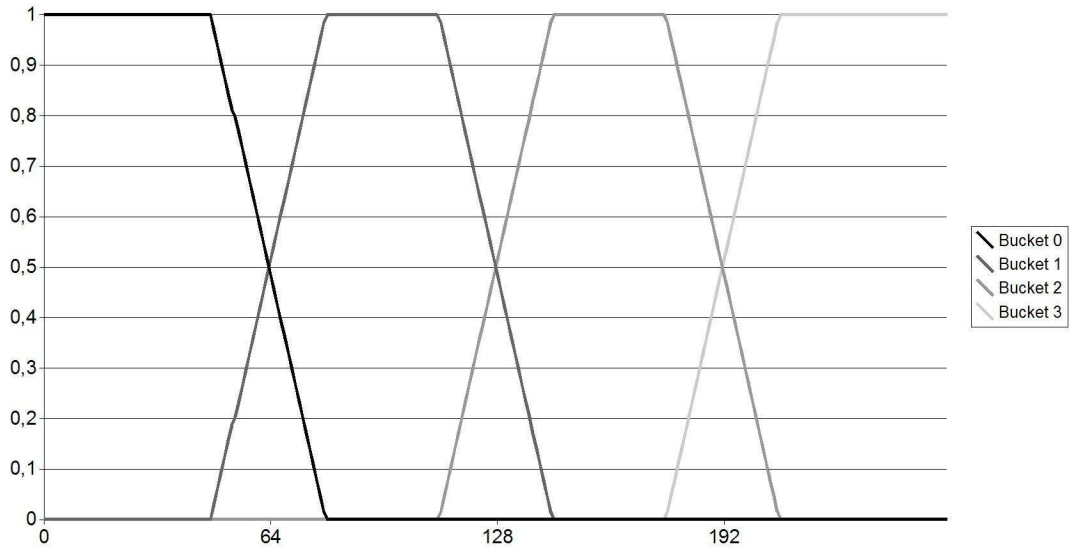

Figure 4.17: Proposal weighting of pixels for associated centroids bins.

a centroid are well scaled to their importance which depends on their size so that abrupt bucket changes has not an so importance influence on the features as before without the dynamic weighting. But it is not a mean solution for every color transitions between different centroids.

4.2.4 Fast frame changes

Events in Mov1

As we have seen in the previous section are our results not so good as expected. The reason for this problem are features of new or moved centroids which depends on variations in image which results in (too) large variations in the of the frame descriptors. This seems to be correct because as we see exists these changes in the frames. The problem is our definition of "event". An event is an event if the state of the video is *stabile*. eg nothing importance changes to the previous state happens. As we have described in chapter 3.1 are the features and the relevance measure depending on the application.

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

Events in halloween

An other kind of problems we have sometimes are with videos with fast and very short frame switching. This is for example used in the video "halloween" as a kind of video teaser[6]. In the introduction sequence of the video is a short sequence of 12 different images with 2 frames each, which results in 12 image changes in approximately 1 second. These images are a kind of abstract of the video content. The algorithm correctly detects these (at a highly level of evolution process) frames as key frames due to the fact that these frames contains different kind of information and acts as a new shot each, which should be detected as.

Also contains the video a scene in which a person on a stretcher is moved through a floor with fast light flashes. Due to the frame descriptor concept[7] are each flashes detected as a key frame[8]. Figure 4.18 shows these 9 frames.

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

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

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

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

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

the changes in the video. The color changes in the left part (after the begin) of the figure shows the fast changes in the introduction[9].

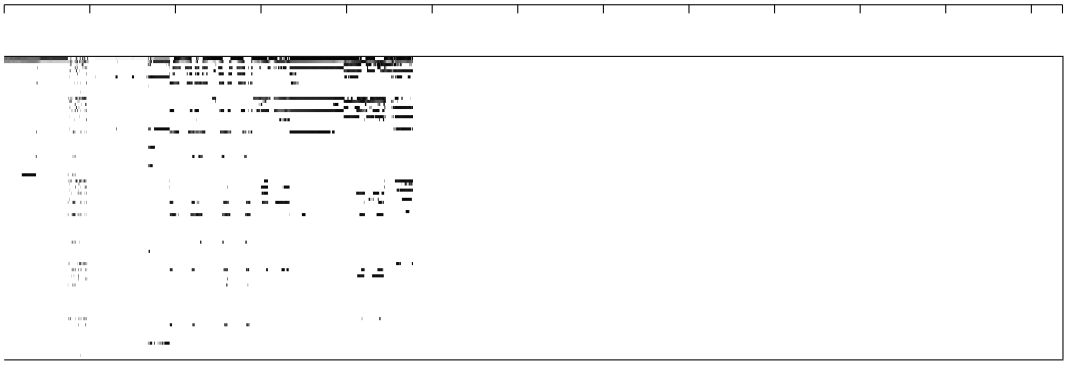

Figure 4.19: Diagram with color intensity over the time. Step width 500 frames. (Approx. 207")

Either we defined the requirements or we implemented the key frame implementation wrong. A dark/bright switch in the video could be interpreted as an event that occurs or, in the case of the "halloween abstract" in the start sequence of the video, as separate cuts which should be detected. In both cases are our definition of the expected key frame wrong. What seems to be wrong with our expectation of the key frames? In both cases are the changes to fast or the context in which the key frames occurs is to short.

There seems to be two possible solutions

[9]We do not have such diagrams for the buckets self, but as we have seen before could intensity diagrams be created from the feature files.

1. We could change the importance of the time. Frames which are temporal to near to each other are to difficult.

2. An increase of the local context makes it possible to measure the state of a frame to the neighbor frames and detect to fast changes in the frame descriptors.

Proposal 1 will bring us in trouble if we implement this .

Proposal 2 seems to be more contestant .

Important is, that these events are very short in the time and they will not result in a permanent (stabile) situation of the scenery. We could add a limitation to the *event* requirement 3 in chapter 3.1 by defining a least durability of the situation to be detected as an *event*.

The easiest way to fulfil this limitation is by filtering features that does 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 take a situation to be detected as a *stabile event*. In terms of the filter, is the question *"How width should the filter be implemented?"*.

Also important is how important are the information that we lost?

In some video scenes we had specks, 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 for comparison purposes also a gaussian filter implemented.

Grey-Level morphological filter

Morphological filters are widely used in the image processing to filter single pixel's in an image depending on the neighborhood pixel's. There exists different and wide varying kind of pixels. Some which only used the horizontal or vertical pixel's or a combination of this as the neighborhood , some with a local area completely surrounding the filtered pixel and sometime in a video sequence are also the pixel's of neighbor frames used to filter a single pixel.

Morphological filter's are introduced by J. Serra [46]. Morphological filter's are implemented by an erosion and a dilation function. Out intend is not to use the filter for pixel's 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:

$$f(z) = -\infty \text{ if } z \text{ is outside the definition domain of } f \tag{4.2}$$

$$\text{Domain:} D(f) = \{z | f(z) > -\infty\} \tag{4.3}$$

$$\text{Translation of f by x:} f_x(z) = f(z - x) \tag{4.4}$$

$$\text{Domain translation of f by y:} (f + y)(z) = f(z - x) + y \tag{4.5}$$

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

$$(f \Theta g)(x) = min\{f(z) - g_x(z)\}; z \in D(f) \cap D(g_x), x \text{ such that } D(g_x) \subseteq D(f) \tag{4.6}$$

In our case is

$$f(t) := FD^n(frame(t)) \tag{4.7}$$

$$g : \{-i, ..., i\} \rightarrow 0 \tag{4.8}$$

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

$$(f \oplus g)(x) = max\{f(z) + g_{-x}(-z)\}; z \in D(f) \cap D(g_{-x}(-z)) \tag{4.9}$$

The from us used operations 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

$$A \circ B = (A \Theta B) \oplus B \tag{4.10}$$

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

$$A \bullet B = (A \oplus B) \Theta B \tag{4.11}$$

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

Our application should implement 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 $2i + 1$:
  $dilate(FD(t_n)) = min(FD(t_n - i), ..., FD(t_n), ..., FD(t_n + i))$

With these implementations are the morphological filters opening and closing implemented. The opening reduces all local maxima inside the filter based on the width of the filter function g and it's values. The value 0 implements no difference to minimum of f inside the filter width. This decision is taken because it is expected, that the feature descriptors comes fromSo the local maxima are completely eliminated. The closing operation reduces all remaining[10] 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 and contains 73 features. The parts when Rustam is weaving are represented by a clear change in the feature. The four marked areas shows some changes in the feature which are a result of the filter.

The light gray curve is the unfiltered original feature, the dark grey curve is the filtered feature with $i = 2$ which results in 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 could we see that peaks in the curve are flatten. This could also observed in the not marked middle peak of the curve.

In area two and three in the upper right part of the image could we see that smaller peaks (area 3) are completely removed with a smaller filter window and widther peaks are remaining (area 2). These peaks are removed by the widther filters.

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

How width should the filters be selected?
In imitation to our efforts in section 4.2.1 to make a framerate independent

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

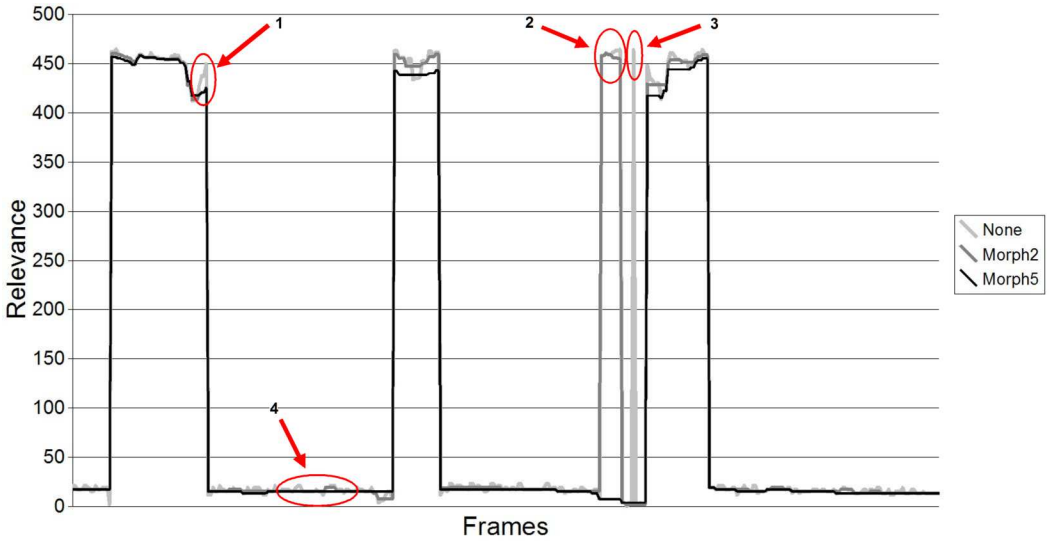

Figure 4.20: Diagram of a morphological (un)filtered dynamic weighted feature from video "Mov3".

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 4.1: Table with effective temporal filter with different frame width and different frame rates.

algorithm, we should avoid any kind of frame numbers in the filter expression. Our key frame expectation are based on our impression of time context in which they appears and it seems to be logical to define a suitable filter in time even if it is based on frames. In that case should the frame width 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 is applied, the frame rate not available is, could we not implement this requirement.

The most newer 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 are

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

Other filters

There are several possible filters that could be used to detected 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 filter width $2i+1$ are constants C_{-i}^{gauss} till C_{i}^{gauss} calculated and normalized so that the sum of the constants is equal to one.

$$\begin{array}{l} C_x^{gauss_i} := \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \\ \text{with } x \in \{-i, ..., i\} \\ \text{and appropriated } \sigma \text{ and } \mu \end{array} \tag{4.12}$$

$$\begin{array}{rcl} FD^{gauss_i}(t_n) & := & C_{-i}^{gauss_i} \cdot FD(t_{n-i}) \\ & + & ... \\ & + & C_{0}^{gauss_i} \cdot FD(t_n) \\ & + & ... \\ & + & C_{-i}^{gauss_i} \cdot FD(t_{n+i}) \end{array} \tag{4.13}$$

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

Conclusion

Filters could be used at various abstraction levels of the feature creation process. Not implemented and tested are filters at the lower levels. This could be for example pixel filters, which filters the data directly inside the images, depending of the surrounding content. Possible filters are 2D-filters which only uses the content of that frame but also 3D-filters are imaginable which uses 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 compontents which is done by [11]. The proposed weighting for the pixels which are assigned to centroids could be distinguished as filter at a higher level.

As we can see, is it possible to eliminate short “jumps” of larger areas between different buckets with morphological filtering. The width of the window

should depend on the application depending expectation of the definition of event.

Not tested and implemented are filters defined for a single frame or for the histograms. Image filters could 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 Problems selecting key frames from the frame list

An other kind of not yet answered questions is not only the abstraction level (how many frames should we need) but also *which* frames are needed. The last question could be somehow strange because our evolution process defines the order of the frames and with the abstraction level is it possible to define which frames are needed from the list.

The problem is that we didn't define before what we understand under *the* abstraction level. Also this is a definition which depends on the observer which 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] makes it sense to define a different abstraction levels and expectations on it. Selecting a different abstraction level will result in a different amount of frames which belongs to the abstraction level.

Our Curve Evolution process gives us some information to create and define some different abstraction levels. The main information are 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 importance (relevance) of the key frames in the evolved frame list

Also sometimes an important key frames is 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.

Sometimes not every frame at the end of the evolved frame list would be a key. We would see that not every frame would be a key frame. If a key frame

resists between two identical frames, then one of these surrounding frames should be a key frame, if the inner frame is removed of the key frame list.

For this problem not trivial problem 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 define application depending a useful and suitable boundaries for the number of key frames.

For example: A decrease in the relevance value in relation to the previous key frame could be for example interpret as a less important frame in the new context. This removed frame makes more sense in the context of key frame removed before. An idea is to merge these two frames together. Abstraction levels could then be defined on merged key frame groups instead of 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 are for some of comparison videos a full set of key frames needed to make comparison possible. So we have tried some algorithms to find a usable number key frames. These number should be not a perfect value, but more rough approximation of the expected result sets. Maybe it is possible to find usable algorithm that will match the number of frames in the result set with this background information.

As we have described, the number of key frames will very depends on the expectation of what kind of frame differences will be *important* enough to be marked as such. Our goal here is to found such an importance level.

For example someone could define in "Mov3" (appendix A.1.1) only the frames where Rustam is waving as key frames and the frames between these frames are no key frames for the observer. If we take a look in the key frame list . We will got these frames if we define the number of key frames as the best five frames. On the other side an observer could also define the frames between the waves, where Rustam does nothing, are also key frames (we a lesser relevance). We will get these result if we raises the number of key frames to seven . So our evolved frame list will give different key frame lists for different importance definitions of different observers.

Figure 4.21: The best three key frames for “MOV3”

Figure 4.22: The best six key frames for “MOV3”

The evolved list contains all frames of the video sequence in the order of it’s relevance. Only the frames with the highest order in the evolved list are key frames. The problem is to define the a number of frames which are key frames. We have several information like number of the frame in the evolved frame list, the relevance when it’s removed . We have the number of the frame in the video. We did not found a useful or calculate able value for the maximum number of frames which should be used to define the frames as key frames.

4.3 Dominant Colors and Optimal Color Composition

The histogram based frame descriptors are easy implemented and the information content seems to be good for key frame extraction. Also the results makes 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 human visual system performs identification of dominant colors by eliminating fine details and averaging colors within small areas [1]. Consequently, on the global level, humans perceive images only as a combination of few most prominent colors, even though the color histogram of the observed image might be very *busy*. Based on these findings, we perform extraction of perceived colors through the following steps. First a color image is transformed from the RGB space into the perceptually more uniform Lab color space. This color space is designed such way that the color distances computed with $||\ .\ ||_2$ matches the subjective impression of color likeness [51]. The set of all possible colors is then reduced to a subset defined by a compact color codebook with a size of 16 to 512 colors [1]. This code book has in our case 99 colors. Finally a statistical method is 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 is considered to be dominant if P_i exceeds a threshold (typically $2-3\%$). The result is 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 by 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

each column being the color composition of a singe 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 shows a visual interpretation of the frame descriptors $CoCo(V)$ for the Mr. Bean's clip. The vertical dimension of the images is 99 and the horizontal is the number of frames pixel's. 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 the bottom.)

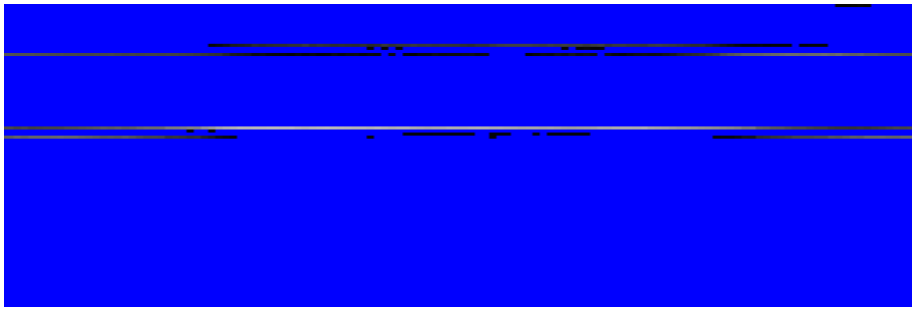

Figure 4.23: Intensity composition representation of Mov1 video clip.

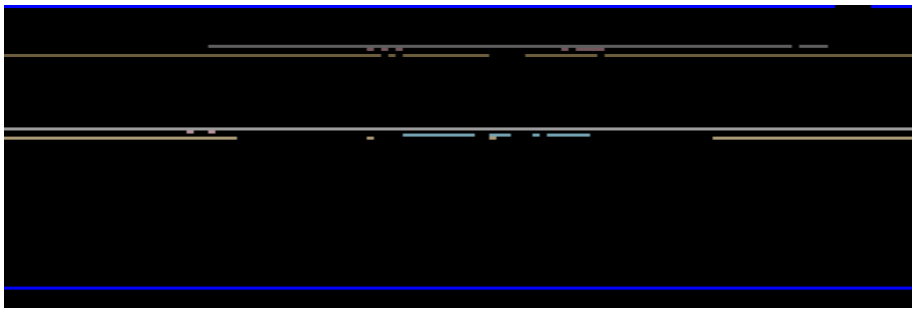

Figure 4.24: Color composition representation of Mov1 video clip.

4.3.2 Optimal Color Composition metric

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

similar colors also occupy similar area percentage [1]. To compare 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 denoted $d(A, B)$.

The representation $CoCo(V)$ of a video sequence V can contain eventual instabilities due to instabilities of the color segmentation in single frames. We use time (frame number) dependences of the frames to filter the instabilities. We apply morphological opening and closing to each row of $CoCo(V)$ with support size of 11 frames. This allows us not only to filter out the instabilities, but also to eliminate the outlier images like completely white (e.g., due to blinding light) or black images (e.g., luck of light) that last just a few frames. Such images belong to common effects of movie makes. It does not make sense to consider such images ascandidates for key frames, since they do not contain any information about video sequences. After applying the morphological opening and closing to $CoCo(V)$, we obtain a filtered representation of the video sequence, which we will denote by $CoCo'(V) = \{CoCo'(I_t)^T\}$. We will apply 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 obtain key frames of video V by simplifying the polyline $\{CoCo'(I_t)^T\}$.

A semi metric called Optimal Color Composition Distance (OCCD) to capture both criteria is developed [1]. To compute OCCD the set of color components of each image is 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 recast as the problem of minimum cost graph matching, for which there exist well-known solutions with $O(n^3)$ com-

plexity [28]. Note that here 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. It is completely independent of and usually much smaller than the color code book size.

Figure 4.25 shows the best five frames for the sequence Mov1. The bad results for the last two images could be explained by the fact that the camera moves from right to the left and the upper left part of the video becomes black. The part at the right that removes from the camera sight is very bright. These area changes and the very intense difference in the brightness seems to be important enough to become a significant difference between those frames.

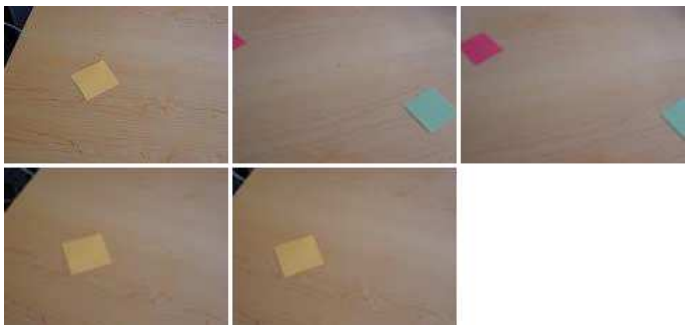

Figure 4.25: Key frame result of video Mov1 with 5 frames. We used Dominant Colors as frame descriptors, without filtering and relevance measure 4.1

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

4.3.3 Filtering

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

We have used the same idea and algorithms here as for the centroid features. Due to this fact, we will also have discovered some of these problems with the dominant colors . We hope that these problems could also be removed by morphological filters. We use the same idea here to apply the morphological filter on the dominant colors. Each of the possible 99 code book colors is handled as separate feature with a given intensity. These intensities are filtered over time for each color code. The total percentual area for each

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

frame is scaled to 100%. Figure 4.27 shows the unfiltered color intensities and figure 4.28 the available colors of video Mov1.

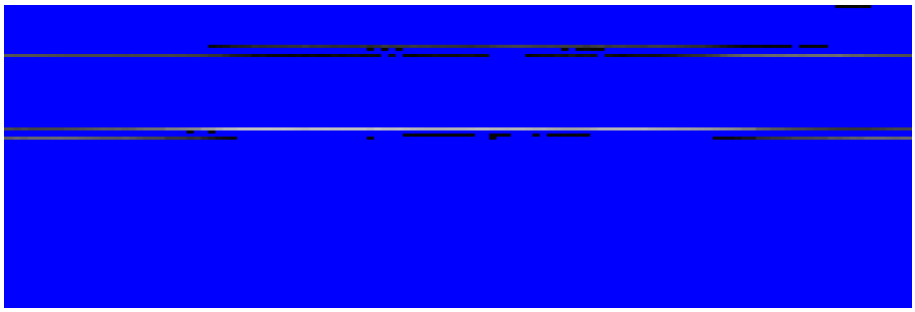

Figure 4.27: Intensity Mov1 without any filter

Figure 4.29 shows the morphological filtered, with a filter width of 7 frames, color intensities and figure 4.30 the available colors of video Mov1.

Figure 4.31 shows the morphological filtered, with a filter width of 11 frames, color intensities and figure 4.32 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 lost of additional information we had in the bucket definition. There is not position information about the colors available. Mirrored images are with this definition identical. Even completely different images with the same color usage

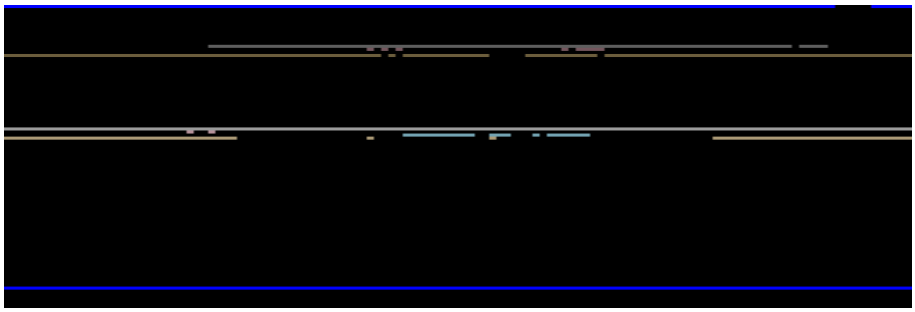

Figure 4.28: Color Mov1 without any filter

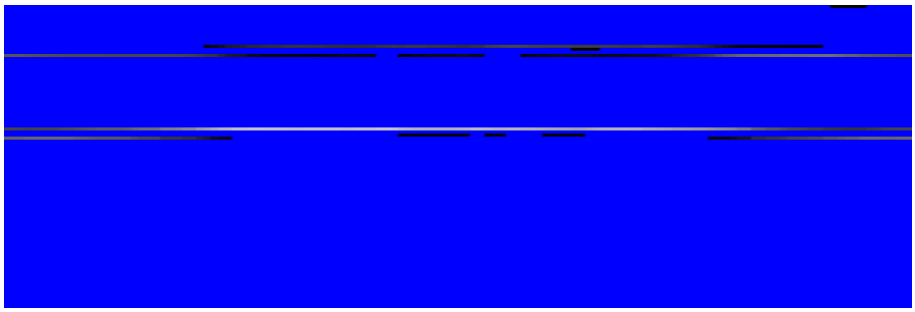

Figure 4.29: Intensity Mov1 - Morph3

are identified as identical. Also slow movements of an object behind a homogene background are not well detected .

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

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

The euclid distance between these coordinates are calculated and added as a second component. These components are 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 changes. The OCCD supports this correct by matching the colors. The coordinates doesn't reflect this color likeness

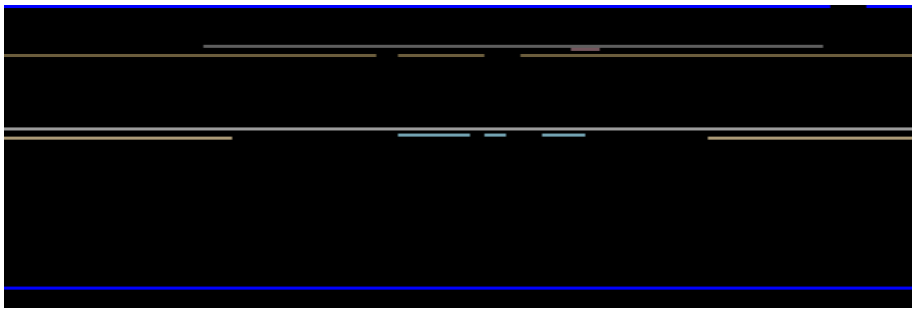

Figure 4.30: Color Mov1 - Morph3

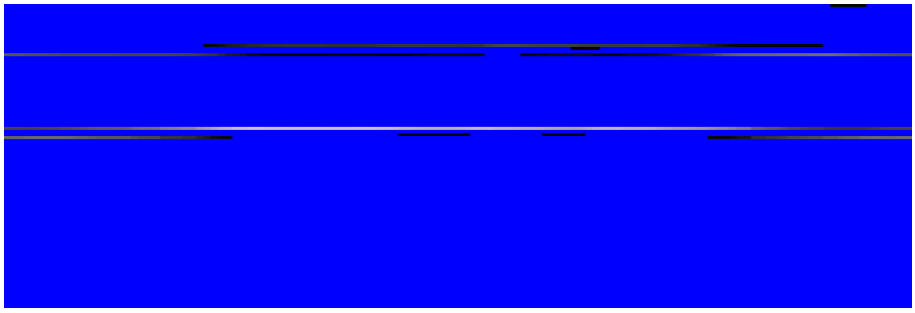

Figure 4.31: Intensity Mov1 - Morph5

2. New parts of objects moves into the image. The OCCD supports this correct by matching the area of the colors. Smaller objects are less important as bigger objects. The coordinates does not reflect this area factor.

The badness is that 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 not well scaled in relation to the first part (color distance).

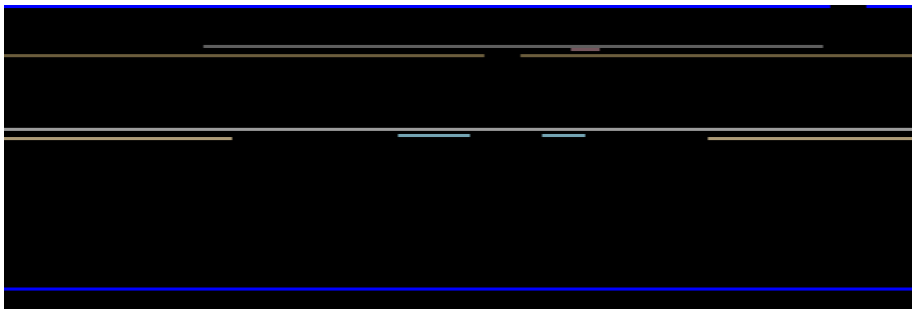

Figure 4.32: Color Mov1 - Morph5

Chapter 5

Experiments on closed videos

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

We have also tested cost functions that contains a larger local context $context_2(f_t) = \{f_{t-2}, ..., f_{t+2}\}$ 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 contains only subsets of these videos and applications.

5.1 YUV vs. RGB

This experiments shows 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 a YUV histogram subdivided in 16 bins which results in 145 features. As we can see appears the last key frame 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, but created with a RGB histogram subdivision of also 16 parts. As we can see contains the resulting frame set expected key frames .

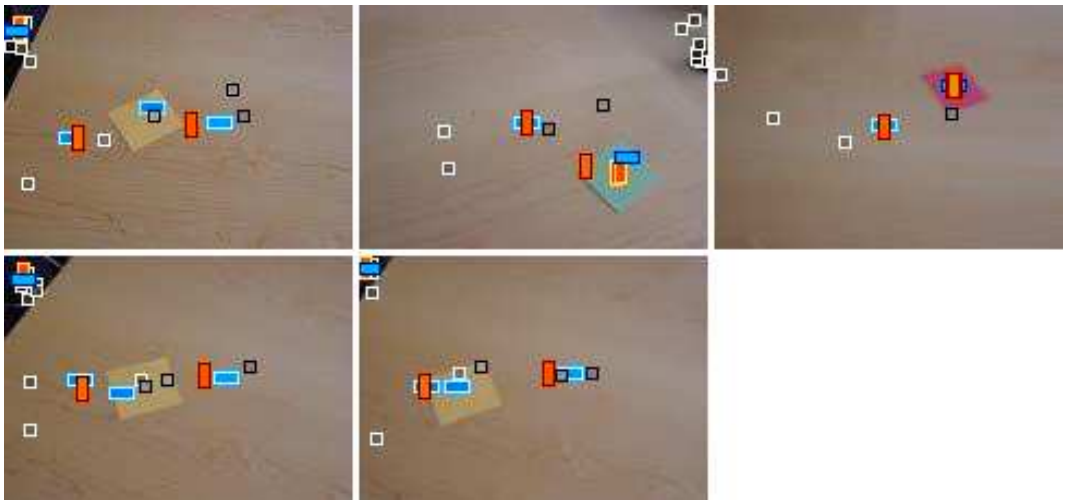

Figure 5.1: Figure with the best five YUV frames of Mov1 without filtering

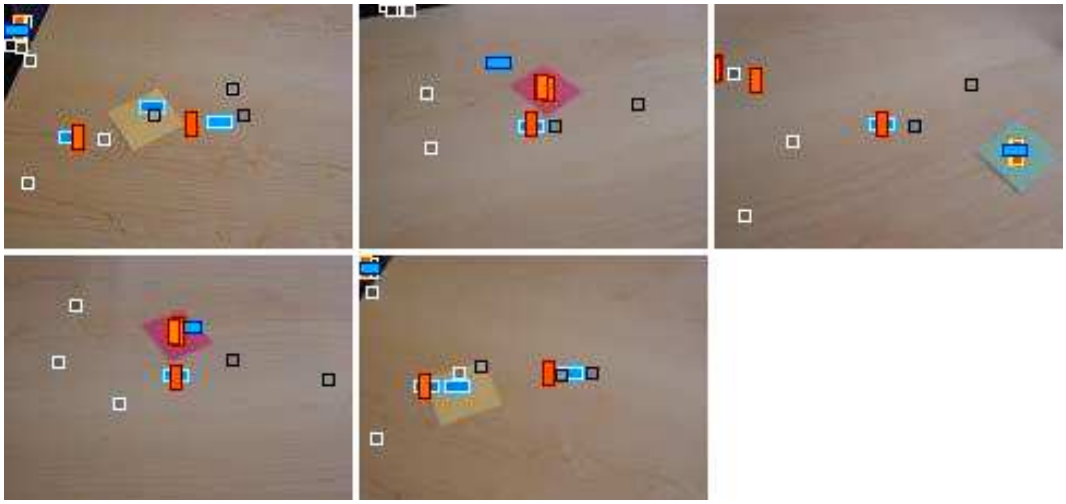

Figure 5.2: Figure with the best five RGB frames of Mov1 without filtering

It seems that for this experiment the RGB color space is the better choice because the results matches exactly the expected key frames.

5.2 Different Cost Functions for the centroid feature

Our algorithm has the possibility to define different kind similarity functions. So could for example

$$\begin{aligned} Rel(f_t, context_2(f_t)) &= d(f_{t-2}, f_{t-1}) \\ &+ d(f_{t-1}, f_t) \\ &+ d(f_t, f_{t+1}) \\ &+ d(f_{t+1}, f_{t+2}) \\ &- d(f_{t-2}, f_{t+2}) \end{aligned}$$

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

Other ideas are

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

5.3 OCCD only

5.3.1 Comparison YUV centroids vs. OCCD

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

Video sequence "Mov1"

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 frame descriptors (37) is low. In this result set is key frame four missing and frame number five is twice avilable. 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 are frames two and three two times mismatched to one single

frame (frames two and three). Frame two could be more associated to key frame three and frame number three to the expected key frame number four. Key frame number two is completely missing. The last key frame is twice, unless with the centroids is the upper left part with the table edge and the background not the problem because these images are nearly the same.

Comparison experiments with enabled filters shows that the results are not better as without filters. In some cases is only the quality of the selected key frame improved. For the dominant color is for example one mismatched frame replaced with a frame with exactly one paper.

Table 5.1 shows the frame number of the results.

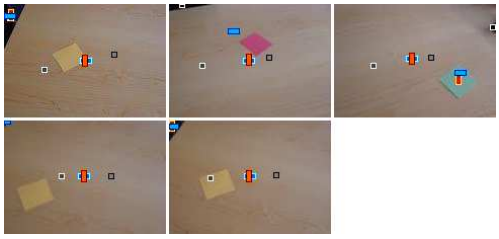

Figure 5.3: Centroid based buckets with 37 frame descriptors, without filter.

Figure 5.4: Dominant Colors, without filter.

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*.

Video sequence "Mov3"

The results for "Mov3" are in images 5.5 and 5.6.

As we can see is the dominant color version better then the centroid based features.

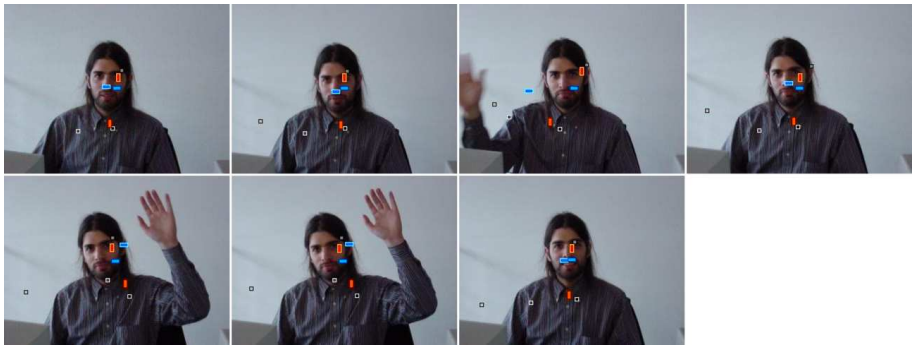

Figure 5.5: Centroid based buckets with 37 frame descriptors, without filter.

Figure 5.6: Dominant Colors, without filter.

Video sequence “Mov00085”

The results for “Mov00085” are in images 5.7 and 5.8.

As we can see is the quality of the dominant color frames version better then the centroid based features.

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.

But the discrete evolution algorithm of the centroid based features performs better.

order	Expected (range)	Resulting frames	
		YUV	OCCD
1	0-	0	0
2	34- 58		36
3	63- 127	72	63
4	142- 163	159	144
5	163- 222	167	181
6	238- 274	238	240
		241	
7	322- 377	377	377

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

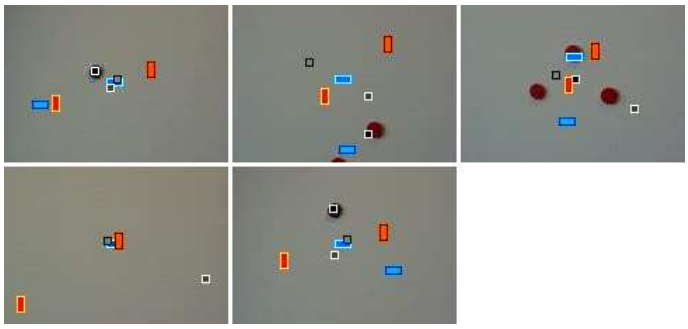

Figure 5.7: Centroid based buckets with 37 frame descriptors, without filter.

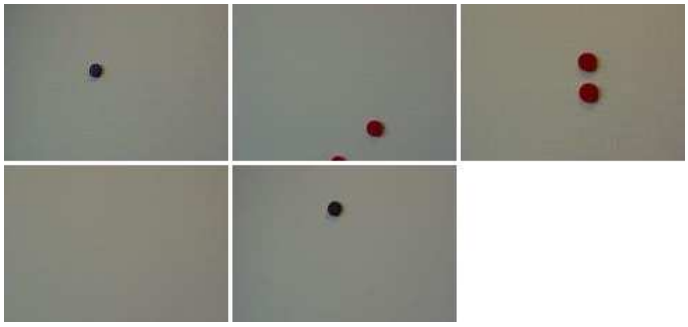

Figure 5.8: Dominant Colors, without filter.

Figure 5.9 shows the key frame results of the *Morph5* filtered centroid based features. Figure 5.10 shows the key frames of the also *Morph5* filtered dominant colors. The performance quality of the centroid based features are not better. The performance of the dominant colors is optimal.

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*.

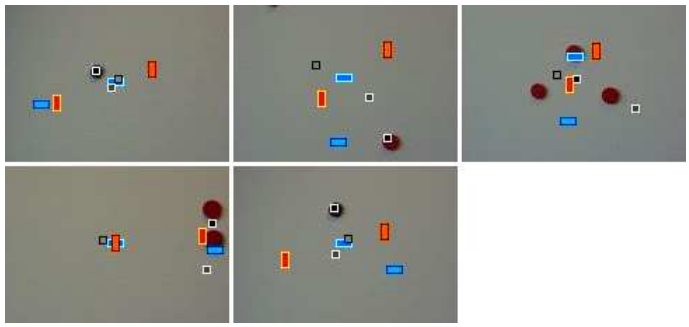

Figure 5.9: Centroid based buckets with 37 frame descriptors, with filter *morph5*.

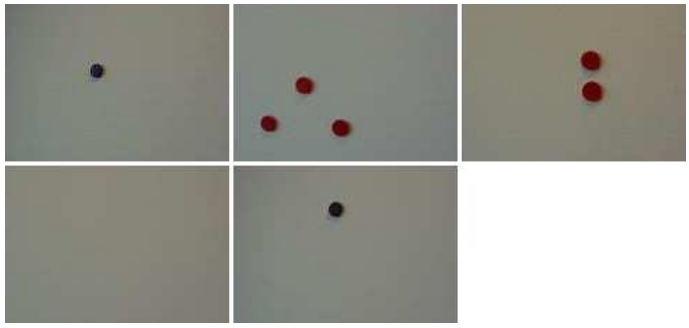

Figure 5.10: Dominant Colors, with filter *Morph5*.

5.3.2 Different image scalings

We have developed histogram and centroids based frame descriptors in 4.2.1 which are invariant to image scaling. Also the Dominant Color frame descriptor from 4.3.1 are scaling invariant.

The video we used for the scaling invariance test is a video from Mr. Bean

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*.

known as "Mr. Beans Christmas". The large and original version of the video is called "MrBeanTurkey". From this video is with a smaller lower (down) scaled version called "mrbeantu" created. Information about both videos are available in chapter A.

We have a lost of information due to the scaling of the video frames because pixels are changed or completely removed. This lost of information is reflect in different content and different information even if our frame descriptors are scaling invariance. The result is that we could not expect identical results for a video and it's down scaled version version. But for a robust key frame algorithm will we expect that we got nearly the same results for visual (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 to $[0, \frac{1}{2}]$ instead of the developed $[0, 1]$. Nevertheless is the quality if the results very good and nearly identical. The frame descriptors are histogram based centroids and time with 37 components. The cost function is that of formulae 4.1.

Figure 5.11 shows the result of the best nine key frames of the large version of "Mr. Beans Christmas" and figure 5.12 shows the results of the small version. They contain the best 9 key frames for the dynamic weighted and unfiltered frame descriptors.

As we can see are the results nearly the same. A detailed frame comparison is in table 5.5. We made also made the same experiments with filtered features. Figure 5.13 shows also the result of the best nine key frames of

Figure 5.11: Result with the best nine frames of the large version of the video Mr. Beans Christmas with unfiltered frame descriptors.

Figure 5.12: Result with the best nine frames of the small version of the video Mr. Beans Christmas with unfiltered frame descriptors.

the large version of "Mr. Beans Christmas" but this time are the features filtered with a morphological filter with a width of 11 (Morph5). Figure 5.14 shows the results of the small version of Mr. Bean with Morph5 filtered.

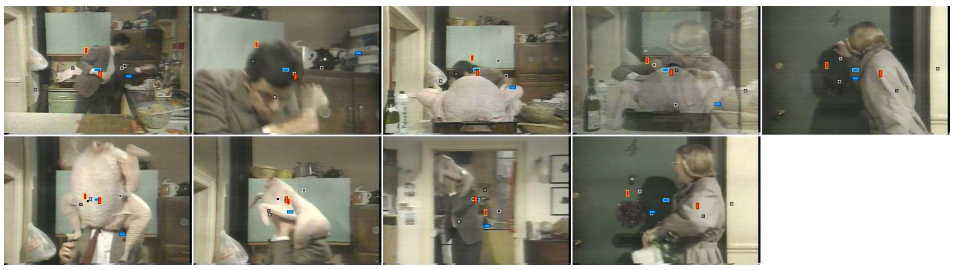

Figure 5.13: Result with the best nine frames of the large version of the video Mr. Beans Christmas with filtered frame descriptors.

Also these experiments has nearly identical results. Frame number four is a little blurred in the large 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 small version of the video. Table 5.5 shows detailed information about the detected key

Figure 5.14: Result with the best nine frames of the small version of the video Mr. Beans Christmas with filtered frame descriptors.

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 of the video segmentation in relation to the expected key frames[1] are subjective viewed not optimal, frames of shot four are completely missing and shot seven has twice key frames, but the results in relation to the scaling invariance are very good, both filtered and unfiltered. For each detected key frame in the large version exists a counterpart in the small version and vice versa.

	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.

[1]These key frames are available in appendix A

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 view the results of the same experiments as made for the centroid based features we take look on the feature vector components of the dominant colors for the different videos. Figure 5.15 shows the available dominant colors of the 99 possible code colors of the large version of the Mr. Bean video. The X-component of the figure represents the time line of the video. The most left column of pixels is frame number 0 and the most right column 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 indicates that the representing code book color in that frame was not available. As we can see are both figures nearly identical.

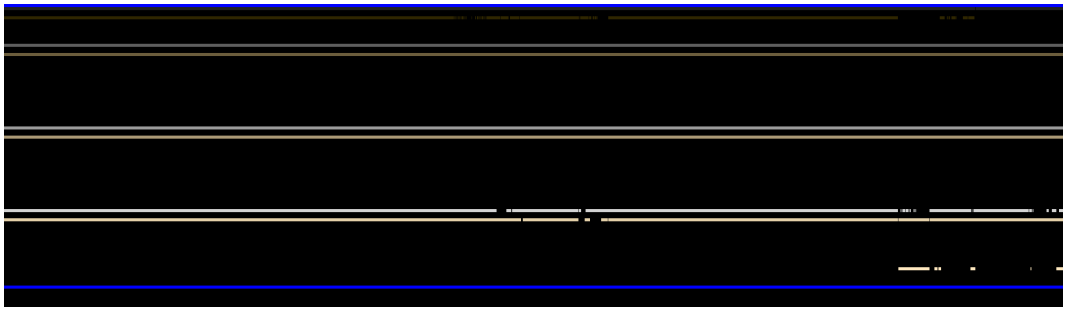

Figure 5.15: Dominant Color Availability image for the large version “Mr. beans Christmas“.

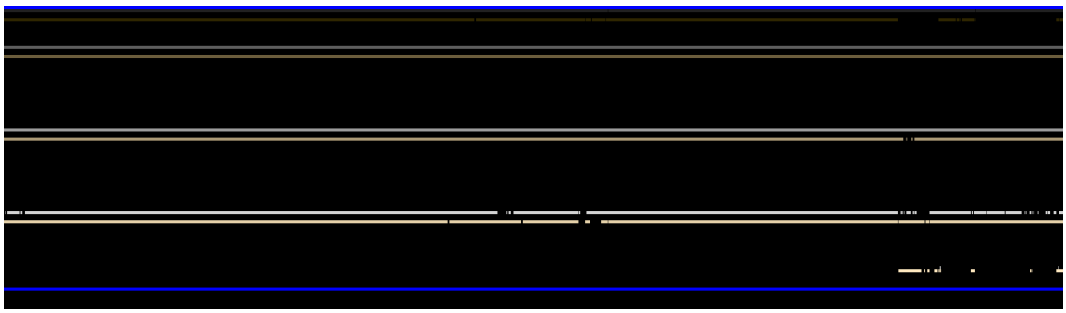

Figure 5.16: Dominant Color Availability image for the small version “Mr. beans Christmas“.

In some frames are some colors in only one of the video frames avialable but not in the other. This is not very as long as the area of the frame is not

very large or as long as a similar other code book color is available (maybe in other intensity of the color). Figure 5.15 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 indicates a larger availability of that color in that frame. Figure 5.16 shows the same area information for the small version of the video.

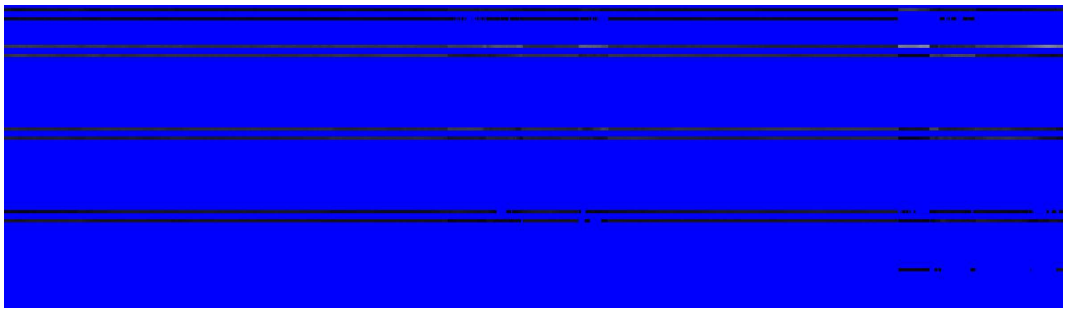

Figure 5.17: Dominant Color Intensity image for the large version “Mr. beans Christmas“.

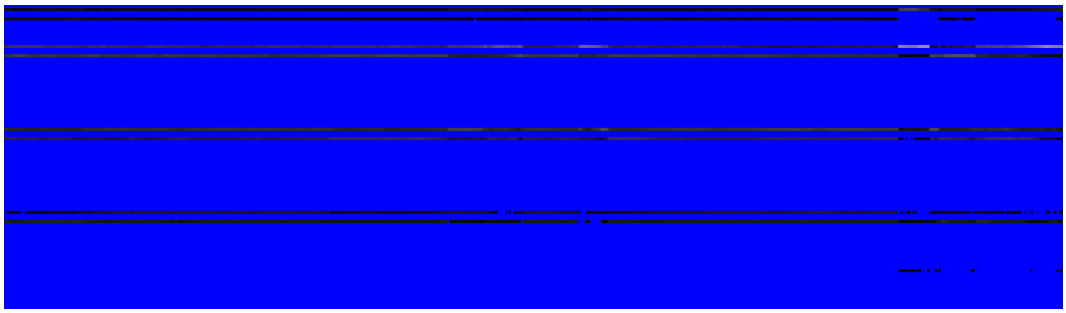

Figure 5.18: Dominant Color Intensity image for the small version “Mr. beans Christmas“.

As we can see are the areas in the different frame parts (nearly) black, so that the misgivings are arbitrary. Figure 5.19 shows the result of the best nine key frames of the large version of ”Mr. Beans Christmas“ and figure 5.20 shows the results of the small version of the video. They contain the best 9 key frames for the unfiltered dominant colors frame descriptors.

Even as the results for the centroid features comes the results near to the expected key frames and also are the results of the scaled videos nearly the same. Figure 5.21 shows the result of the Morph5 filtered features. Figure 5.22 shows also the results of the small version with Morph5 filtered.

Figure 5.19: Result with the best nine frames of the large version of the video Mr. Beans Christmas with unfiltered dominant colors.

Figure 5.20: Result with the best nine frames of the small version of the video Mr. Beans Christmas with unfiltered dominant colors.

Figure 5.21: Result with the best nine frames of the large version of the video Mr. Beans Christmas with filtered dominant colors.

Table 5.6 shows a direct comparison between the expected key frames for the shots and the results key frame numbers. Also in this case are the results nearly the same with an exception of the key frames for shot 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 5.22: Result with the best nine frames of the large version of the video Mr. Beans Christmas with filtered dominant colors.

	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.

Chapter 6

Comparison

We have seen in chapter 1.3 the functionality of two other algorithms [31, 53] to create a temporal segmentation of video sequences.

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 content important areas are in the centre of the frame, this is reflected by weighting of the pixels, depending on 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 bins and inside the window is the gauss filter 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.

The most interesting idea in this manuscript is the possibility to create an "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. What I miss is the lost of region information.

This algorithm is probably only good to detect hard cuts as shown in figures 1 and 2 on page 10. Due the lack of a comparison in a local context, slow

movements will not (or badly) detectable because the frame differences are to 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 this kind of frames are not directly those kind of (key) frames that should be detected, because shot boundaries normally happens with faster transition between shots, but this is relative and should be invariant to the frame rate. The only probability to detect such slow small differences is the dynamic shot detection level. The variation in the algorithm is flexible enough to fulfill the expected shot detection.

I don't understand the reason why they have used a curve approximation to detect minima in the curves. 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, but the implemented filter does not consider this.

As every error propagation, the filter will reduce the amount of information. This could result in not detected shot boundaries without clearly transitions. The only nice feature is the possibility to define different frame width for the filter window.

The base algorithm is very simple and this one of (many) possible implementations. I think the implementation details has some interesting aspects like "field of view" selection of the features and the dynamic detection level. These advantages are not new. What I'm missing are region information in the features and a better implementation of the filter. Also I think the curve approximation is not necessary and could be implemented by directly compare the differences in the inter-frame distances.

Comparison with the DCE

Due to the eliminating of unimportant frames, we will have only potential key frames, that not only will be compared to its neighborhood, but also to the neighbor Comparison of their algorithm with our algorithm Disadvantage

An interesting idea is to define a *field of interest* which could result (application depending) in better features which could result in better key frames. (I think our mov1-video could one of the videos which will show better results.)

We also filtering our features, but in the time and not in the frame which could also be a good idea to test. But I think that our starting-point for the filter is better. On page is described that over a time of 9 frames 8 false positive key frames are eliminated by the the error propagation. With our possibility to define the filter over the we would be able to eliminate all frames if this desired for this application. I expect that due the concept of our algorithm we would also detect only one key frames (depending on the abstraction level) for this part of the video.

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

The Discrete Curve Evolution (DCE) is not designed for special features or frame comparison metrics. I think the DCE is a better algorithm as those in the manuscript, because it is easier to eliminate unimportant frame as to define important frames (as they have done). Our problem is, that we doesn't know exactly which part of our frame hierarchy are the important frames. This is better solved in the manuscript. Maybe we could also use such a dynamic analyses of the relevance curve to find the minimum relevance value.

The comparison of a frame to it's neighbor frames in the DCE is not directly based on a metric between two frames, but indirectly in the relevance measure. I would say, in comparison with our selected features and filters, that we have more information with the centroid coordinates of the histogram colors. Our histograms is also flexible (or variable) as in the manuscript. I think the only better part of the feature selection is the pre processing with the "field of interest" areas.

Our filter is better implemented because we a selection between different filters (Gaussian and morphological) and different filter width (like the manuscript) but it is done in the time dimension. Our error propagation consider the temporal characteristic of the relevance measurement.

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].

I think the abstraction ideas and algorithms are good and functional. The idea to predefine some abstraction layers is a good idea and it's easier to work towards algorithms that will implement the abstractions. The disadvantage is that it could be stiff for some kind of videos. All work is based on the shots, which fundamentally depends on it's detection algorithm. The disadvantage of their algorithm is that it only depends on the intraframes of a MPEG videostream and the features are only the values of the color histogram. A distance metric will only detect fast frame changes. The gradual transition algorithm also only detects one transition between two cuts. What happens if more then one of such a transition exists? Shots will not be detected and the higher level algorithms will fail. Even if a shot is detected correctly, then the (visual) shot comparison is based on a frame comparison of a specific frame of the shot, even if the shot contains more gradual transients or a pan over a scenario with different information contents. In that case are the frames (and also the frame or shot comparisons) not significant or even invalid for that shot and also the higher abstraction levels are not usable. The frame comparison is based on other frame descriptors as used for the shot detection self. It is not impossible that one algorithm detects a new shot even if the other algorithm does not detect this. Bad detected shots be the first algorithm could result in a bad calculated threshold for (shot group detection) in the other algorithm. It would make more sense to use the same frame descriptors and comparison metric for both algorithms (shot detection and shot grouping algorithm).

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

A good idea is 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 compared algorithm. In the DCE we do not have predefined abstraction levels, we only none specific abstraction levels. Due to our algorithm, we could also be able to find scenes, groups and single shots, but it is not possible to say at which abstraction level we are.

Comparison of the algorithm features:

Frame descriptors
The used frame comparison algorithm is based on two kind of features. It has a histogram based part of features and a texture based part. I think for the histogram based features are we better, because we are 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 the coarseness texture.

Low level key frame detection
The used shot detection algorithm is not the simplest algorithm because it also detects gradual transitions. But this algorithm could fail if to much gradual transitions are used or if the transitions are to slow or not existent. (Like a long camera move across the horizon with different content.) Also the selected key frame (which is important for the later analyze) is more a random as a content depend selection. I think that our algorithm will detect such shots better and the key frames safer, because the temporal distance of the shots has no influence on our key frame selection process and this selection is based on the conspicuous of a key frame in relation the neighbor key frames. (And this "conspiciuty" 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 build a hierarchical tree of the frames in the order of the abstraction levels. Due this fact, the DCE could group shot (key frames) to a higher abstraction like shot groups. Therefore exists different strategies[1] which are not further pursued, so a comparison at this level with these algorithms is not possible. This was also not our intention 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 as only detecting (shot) key frames.

[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 are shots in a "shot group" successively removed until one or two frames are left. These remaining frames of this "group" could be interpreted as key frames for this group.*

6.3 Experimental comparison with David Rossiter et. al.

David Rossiter has on his Homepage [45] three videos with his own key frame creation perl script. The three videos he used are a short sequence from the movie "The blade runner", a tour through a house and collation about Kylie Minogue. The videos could be found on his homepage.

In appendix A are the videos and ground truth results presented.

6.4 Experimental comparison with Drew

Drew et. al has developed an efficient clustering method [22] 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 are such result sets not always comparable with other result sets created by other methods due to the background information what is expected as a key frame.

These ground truth key frames are compared with the results which are 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 which is artificial created by human and the result set of an other segmentation algorithm.

Figure 6.1 shows the expected ground truth key frame result set as expected. The four frames are from the video *beachmov* which is a small video clip of approx. 30 seconds with four shots. Shot one is a larger shots showing with a left-right-left pan over a beach with water, a city in the background and a forrest. This shots blends over into shot two showing a beach volleyball game. With an other blending is this shot transformed in shot three which shows a few people in a swimming pool. This shots moves with a blending into shot four which shows the edge of a swimming pool. Figure 6.2 shows the result of the Drews 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 ok. But as we described before does 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: Drews key frame result for video *beachmov*

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

This example shows even the generell problem that it is not trivial to define which frames are key frames even as the problem how many frames should be left in a resulting key frame set. But the our results shows, beside these questions, that the algorithm gives good results for a temporal video segmentation algorithm which not only detects shot detections. 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 could be found on the origin website of Drew [22]. All comparison between the clusterinjg Drews algorithm and the *discrete curve evolution* is available on *http://www.videokeyframes.de/Information/Doc/GlobalResults.pdf* [20].

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 quantity of false detected key frames.

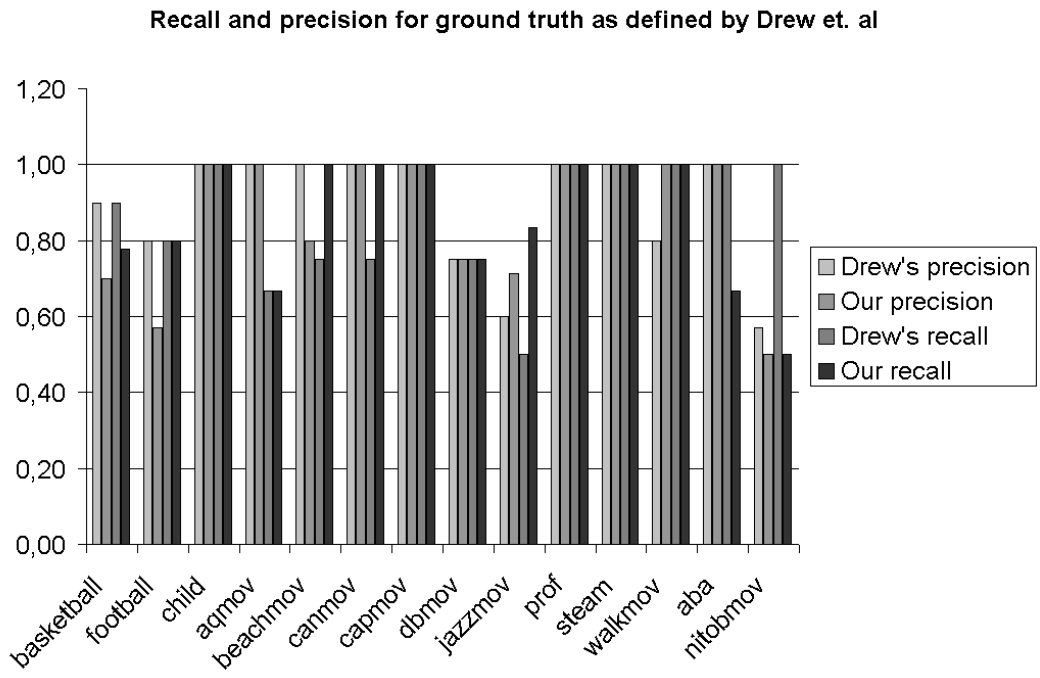

Figure 6.4: Precision and recall for Drew's and our results

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 is the difference from a data technical point of view is the open characteristic of the video material. The video stream didn'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 is analyzed.

The target is to detect unpredictable changes in the scenario like the appearance or disappearance of persons and objects [42] or an other change in the scenario.

These kind of detection is not trivial because there are many not static parts in a video stream which make it difficult to clearly detect a change. In outdoor applications exists many environmental features which is an important influence on the appearance of the video content. Wind moves trees, sun and clouds creates light and shadow etc. Also the camera self has an important influence on the content. Where is the camera mounted? Is it static or dynamic motor driven? Which direction is recorded? Is zoomed into the scenario? All changes in these features will change the content of the same scenario.

It is nearly unimpossible 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 eliminating the environmental noise. Due to the amount of external influence is this not trivial.

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

Open video streams are for example in real time video streams which does not have a well defined start and end frame. An example for such kind of video streams could be surveillance videos to observe areas and entrances and detect access to these. Security personal could be automatic alarmed if something or someone will enter a not cleared area.

Also video streams are used in quality assurance and in processing processes to observe the texture of an product.

Video stream data analysis are also used in quality control of surfaces like rails and streets.

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

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

7.1 Window analysis

The solution is to make a local analysis on a connected subset of frames of a video stream. 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 are the three frames shown of the local maxima of figure 7.1.

The analysis of video on a subset is done with the same algorithm as we have used for the analysis of closed videos with a fix start and end frame. The fragmentation is done at the higher feature level inplace of the lower video level. This could be done because the frame descriptor creation and filtering applications are local operations on the video stream which didn't need global information about the whole video. The advantage is a minimum amount

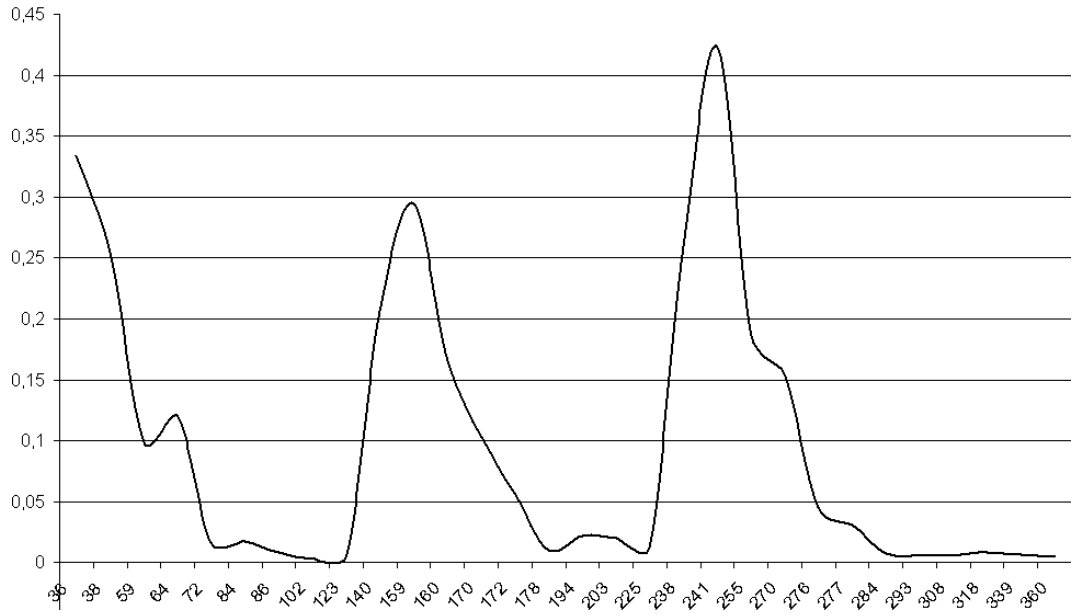

Figure 7.1: A relevance Curve of "Mov3"

Figure 7.2: Frames at the local maxima of figure 7.1

of changes in the existing applications, so the frame descriptor extraction and also the filter applications could be reused without any limitations. A new fragmentation application creates from the (un)filtered frame descriptors the requested subset a new subset on which the discrete curve evolution is applied. With the result of the DCE is a relevance value for the window calculated and associated to the most important key frame of the window. This pair of window/frame and 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 to define the window relevance to the relevance value of most important key frame inside the window? The window relevance is introduced to have the ability to adapt the importance of the *whole* video content in a window on our requirements. A window in which more than one event happens could be more important than a window at which only one event

appears. The usage of a single frame relevance value will only shows the availability of at least a single event. More than one important key frame inside the window would not be reflected by a window relevance which is based on a single frame.

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

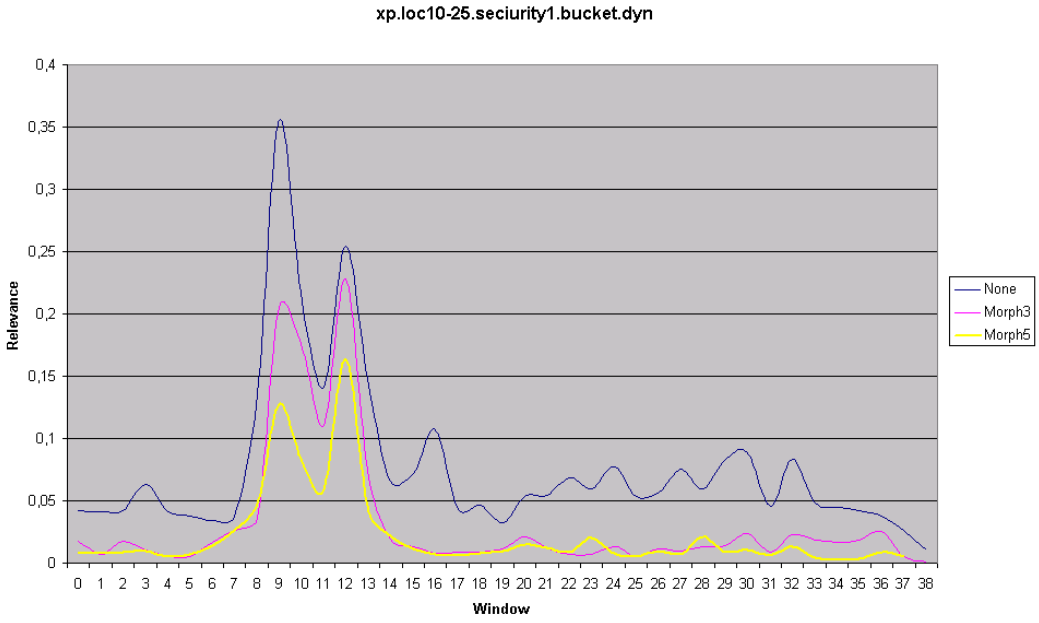

Figure 7.3: Example with relevance values as produced by our local curve evolution process. The used video *security1*

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

Our intend is not only to find the best key frame over the time but also detects events. It is necessary to measure the quality of windows and to fulfill this requirement it is usefull to detect important windows rather then only a single frame inside the window. This frame becomes be important to give a hint to 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 "window". The first questions here is how we should define the width and the location of the window.

Let us take a look on 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 are started with the frame before the 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 no key frames of the analyze, but they are by defined for the global video key frame extraction as key frames. For the local analyze this is not desired, because we only want real calculated key frames.

	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 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 are more then 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 sense of their relevance values).

	0 1 123456789012
window 1	sfkfffe.....
window 2	.....skffffe

Table 7.2: Not all potential key frames could be detected

New window start 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 are started in the middle the previous window. So each frame of the video stream has two changes to be a key frame. The should only be an example for each new window position which started at a static position inside the previous window.

	0 1 123456789012
window 1	sffffe.....
window 2	...sffffe..

Table 7.3: New window starts inside the previous window

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

Disadvantage:
Each other following key frames from different windows are not necessary ordered in the time. More time is spend to perform an analysis of the same number of frames.

	0 1 1234567890
window 1	**sfffke**...
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 are 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.

	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 then one good key frame in each window. Subsequent key frames are in the order of there appearance.

If an important event happens that is distributed of more then one window, the key frames then also occurs in a short time period, frames of that are closed to (or closed in) this event are more often analysed and also to become a key frame. In table 7.2 is the key frame in window 3 in 3 windows and also has 3 changes 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 of the step width from 1 to n frames will decrease the analysing data volume to $1/n$, because we only take every n-th value. Also window curve diagrams has the same value domain.

7.3 Window width

The width of the window reflects the are in which an important event is expected. If an event subdivides over some windows, then it is expected that the maximum relevance value of a window key frame is lower then that of a key frame where the event fits in a window.

If a window is to width, then it's possible that more then one event could be in the frame and only one could be detected. It is possible to lower this risk by setting the window intelligent. The step width should be less or equal then the window width, else are some frames not in a window, which could result in a not detected key frame. Also a step width which is too small could lead to double detected key frames and/or key frames 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 have used for our experiments. Also the width of the window for the dynamic step width is given.

Window width defined on time analysing time window

It is not possible to define a fix width that will match all scenarios. We use a width of approximately 2 seconds. The most of the sequences uses a frame rate of approx. 25 fps, so this results in a window of 50 frames.

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.

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 it's 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 in the next window also nothing importance will happen, so we could increase the window width. (n is the increasing factor) If something importance 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

It's possible that, if the relevance value of the key frame is increasing (relative to the previous key frame), that an important event will happen. It makes sense to decrease the window width in such cases. At the other side, 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 different static window width.

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.

Due to a larger window width, the focus will be lay longer on really important key frames and those will be detected "earlier". Also less important key frames between two important key frames will be skipped.

shows the results for the different window width merged together.

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 which 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^{window}

The advantage is that information about the events could be stored inside the window relevance value. Short events, not more like a noise, could be reduced in their importance, but longer events could be raised in their importance.

We have for example used for our following experiments the sum of the best C relevances.

$$\begin{aligned} &M^{window}(w) := \textstyle\sum_{i=0}^{C} M_{rel}(f_i) \\ &\text{with constant } C < |\{f | f \in window\}| \\ &f_i \in window \text{ ordered:} M_{rel}(f_i) > M_{rel}(f_{i+1}) \end{aligned} \tag{7.1}$$

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.

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 be automaticely a key frame. It makes sense to define a tolerance or detection level which a relevance must reach 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 even as on the video source and window step width and size self. As we have seen will result a filter in smaller relevance values, but also in a smaller noise level, so a static level of the relevance value could make sense, but a dynamic solution should be preferred. A video with a higher background noise will make too much false positive detections.

Many 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 are interest us are frames 97 (relevance of 0.33) and 135 (relevance of 0.28). All other extrema are noise with a relevance between 0.01 (frame 241) and 0.14 (frame 199). This results in a worst ratio of 2.01 between good and bad frames.

The raw data of the curve evolution is below. 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
```

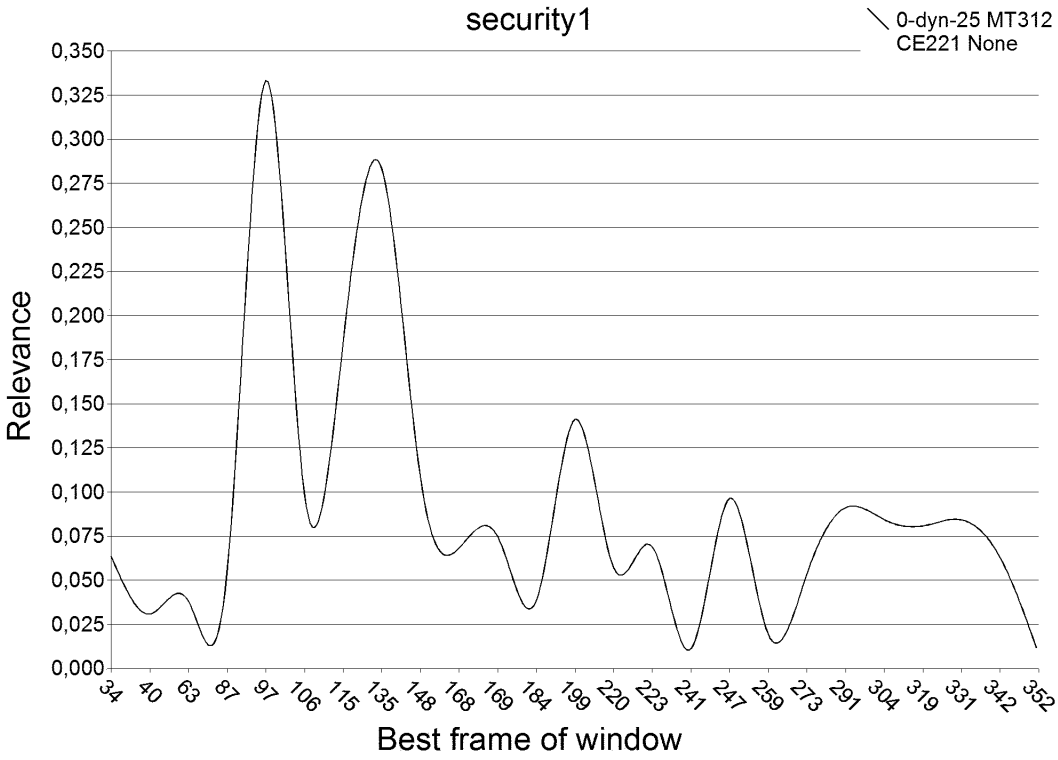

Figure 7.4: Video security1. Dynamic weighted centroids with 37 frame descriptors, no filtering. Window position starts of the best key frame of the previous window. The window relevance is the relevance of the best key frame in the window.

Figure 7.5 shows the window relevance features for the data of video security1 with 37 features, no filtering and 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 are noise with a relevance between 0.08 (frame 184) and 0.20 (frame 304). This results in a worst 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 frame in the window. The maxima are frames 97 (relevance of 0.33) and 135 (relevance of 0.28). All other extrema are noise with a relevance between 0.04 (frame 259) and 0.14 (frame 199). This results in a worst ratio of 2.01 between good and bad frames.

The following result is that of the same experiment before with the exception,

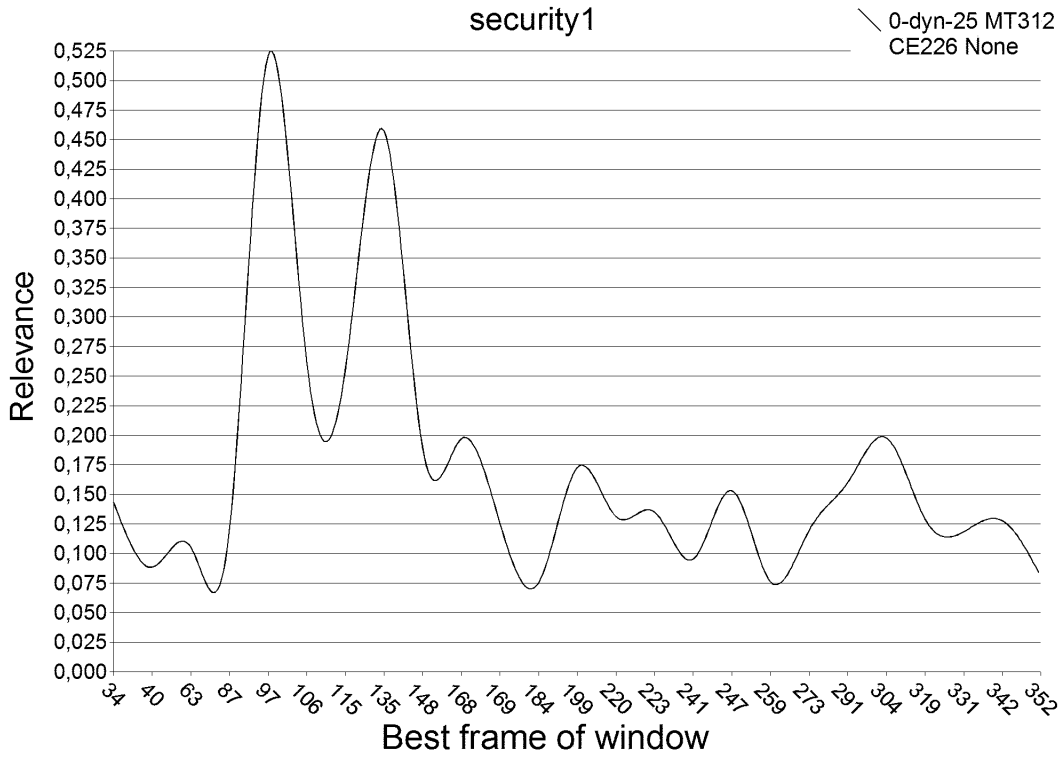

Figure 7.5: Video security1. Dynamic weighted centroids with 37 frame descriptors, no filtering. Window position starts of the best key frame of the previous window. Window relevance is the sum of the relevances of the three best key frames.

that the best key frame of the window is defined as the key frame at which the maximum relevance appears. This is not necessary 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 are noise with a relevance between 0.04 (frame 258) and 0.14 (frame 199). This results in a worst ratio of 2.01 between good and bad frames.

First algorithm

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

Algorithm:

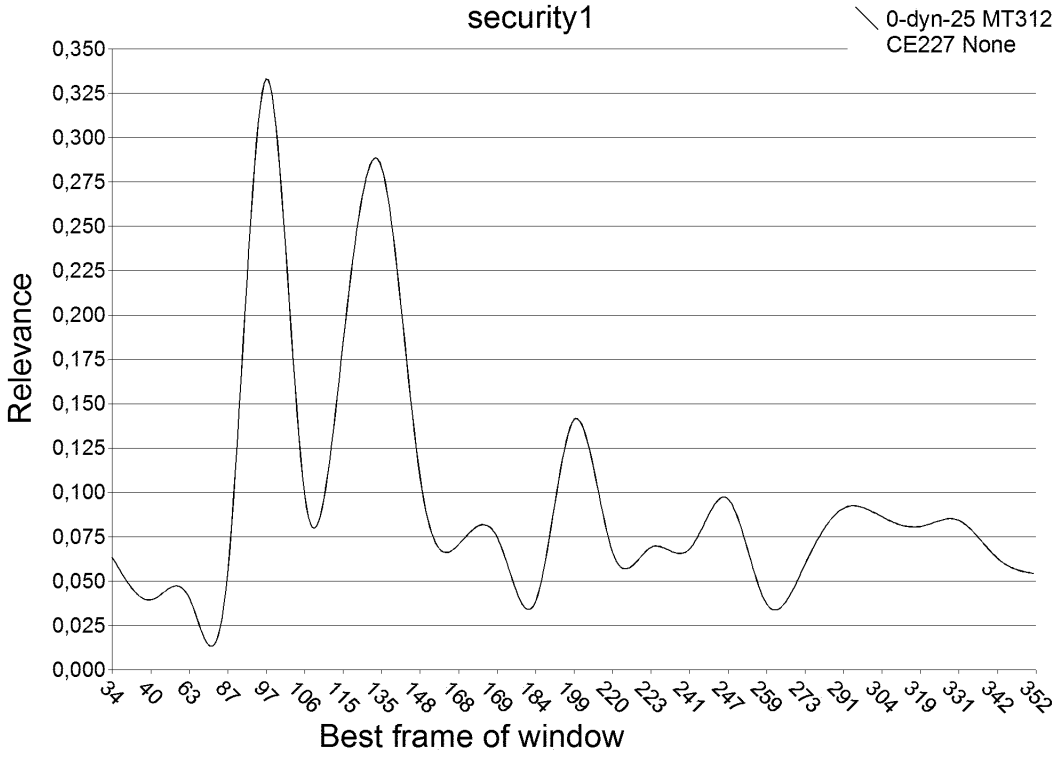

Figure 7.6: Video security1. Dynamic weighted centroids with 37 frame descriptors, no filtering. Window position starts at the best key frame of the previous window. Window relevance is maximum of the key relevances.

$relevance(window_{t-2}) < relevance(window_{t-1}) \wedge$
$relevance(window_{t-1}) \geq relevance(window_{t-1})$
$\Rightarrow$ Potential key frame of window $window_{t-1}$ is key frame

Second algorithm

The idea is to define a static relevance level from which on frames are important.

Algorithm:

$relevance(window_{t-2}) < relevance(window_{t-1}) \wedge$
$relevance(window_{t-1}) \geq relevance(window_{t-1}) \wedge$
$relevance(window_{t-1}) \geq T_{constant}$
$\Rightarrow$

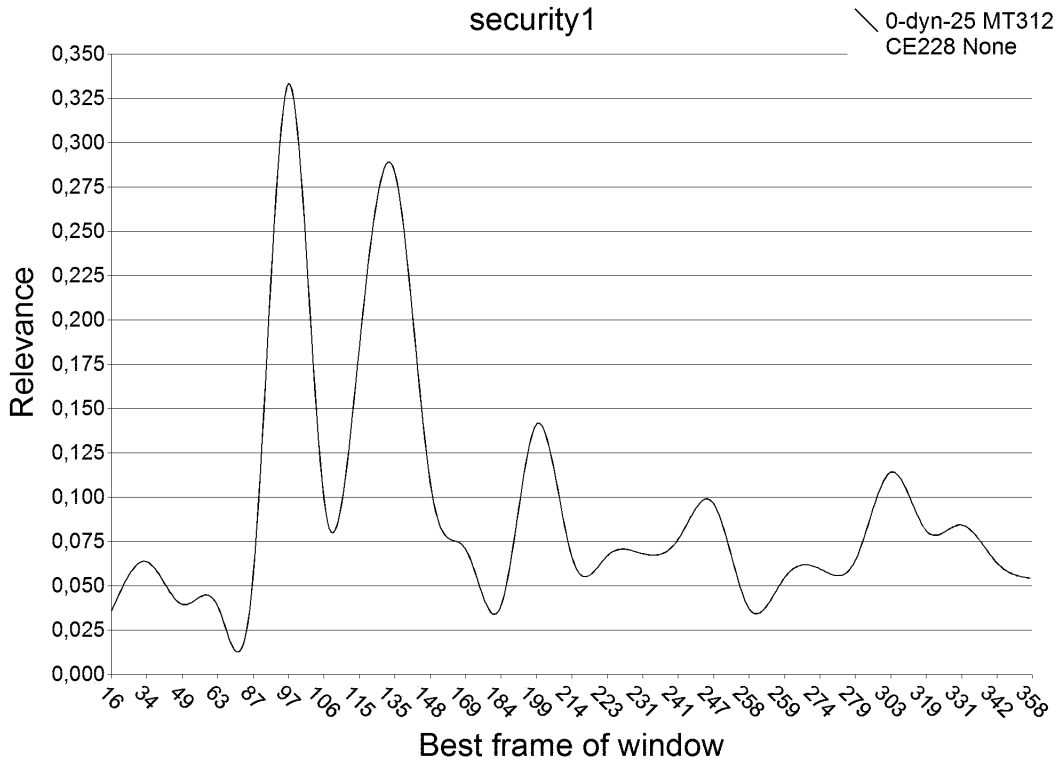

Figure 7.7: Video security1. Dynamic weighted centroids with 37 frame descriptors, no filtering. Window position starts at the key frame with the best relevance value of the previous window. Window relevance is maximum of the key relevances.

$T_{constant}$ is a pre defined constant threshold. The event appears in window $window_{t-1}$.

Third algorithm

The idea is to define a dynamic relevance level from which on frames are important.

$relevance(window_{t-2}) < relevance(window_{t-1}) \wedge$
$relevance(window_{t-1}) \geq relevance(window_{t-1}) \wedge$
$relevance(window_{t-1}) \geq T_{dynamic}$
$\Rightarrow$

$T_{dynamic}$ is a dynamic generated threshold. The event appears in window $window_{t-1}$.

7.6 Filtering

The following figures shows the usage of different morphological filters applied to the video *security1*. As we see is no different between the *none* filtered and the *Morph1* filtered features. The source and the filtered features of digital camera are identical. It seems that the digital camera triples the recorded frame to get the necessary video frame rate. Figures 7.8, 7.9, 7.10, 7.11, 7.12 and 7.13 shows the influence of morphological filters with different filter window width. The range of the relevance is reduced to lower levels. Different details of the curve disappears, but as we can see in figures 7.9 and 7.10 also are some details not reduced which results in a raise of the importance at these places (around window number 199).

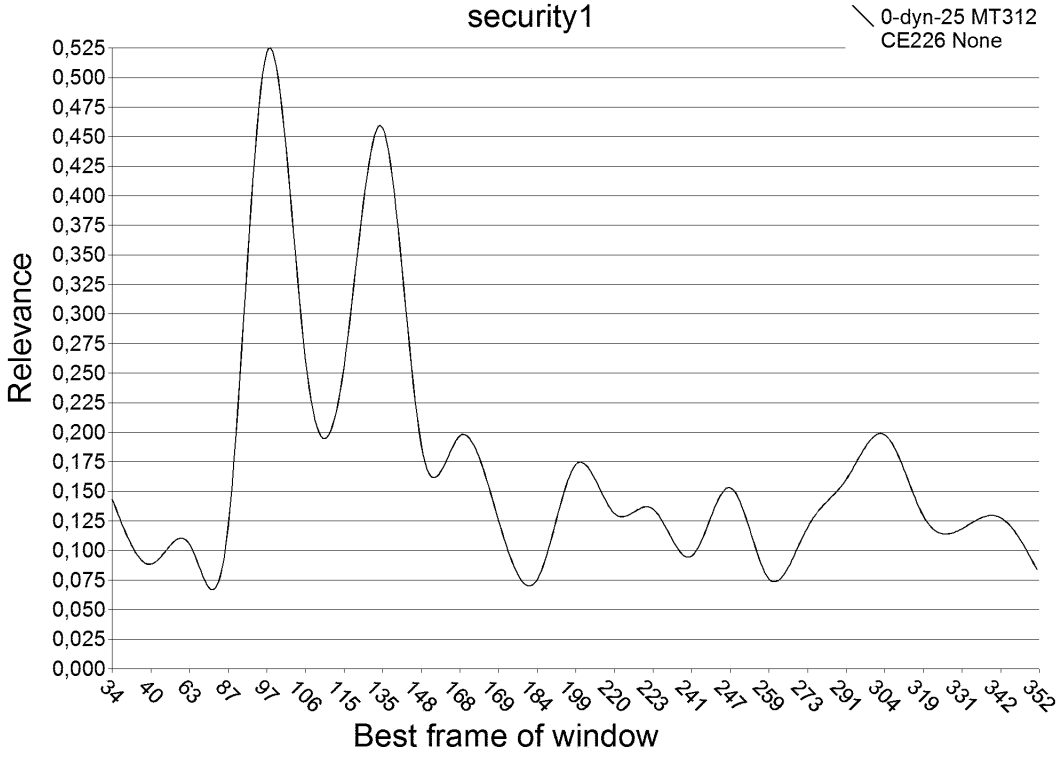

Figure 7.8: *None* filtered video *security1* with 73 features and curve evolution *CE226*.

Figure 7.14 shows the influence of different morphological filter width on 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.

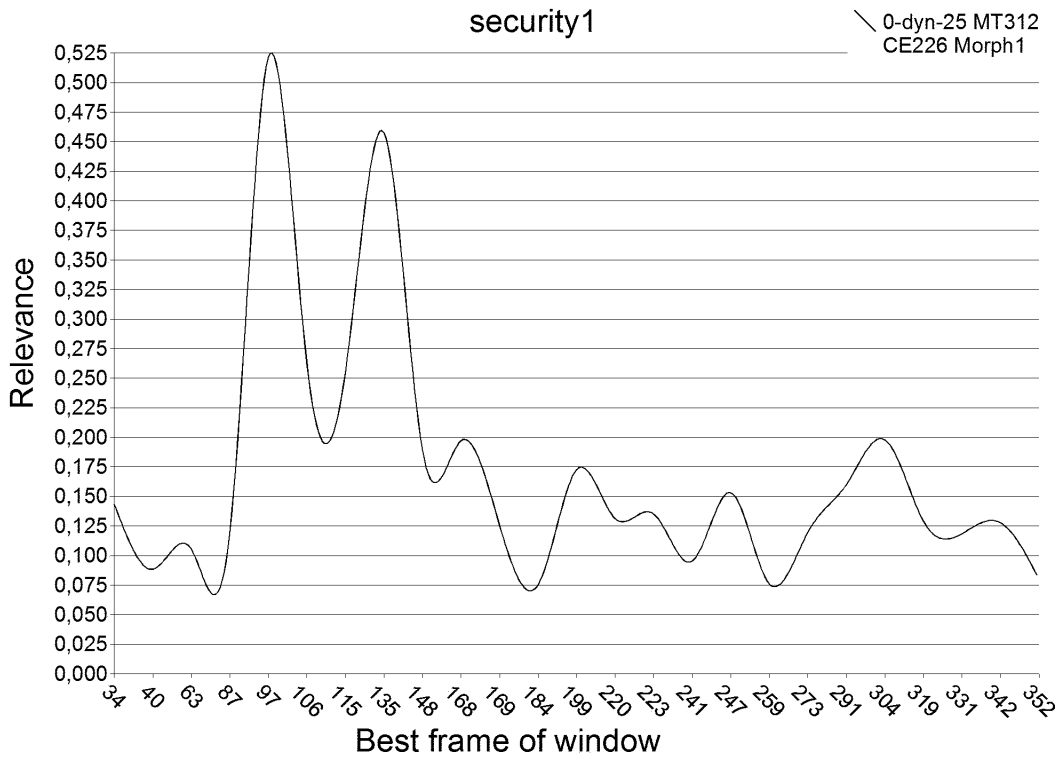

Figure 7.9: *Morph1* filtered video *security1* with 73 features and window relevance of the sum of the best three key frame relevances.

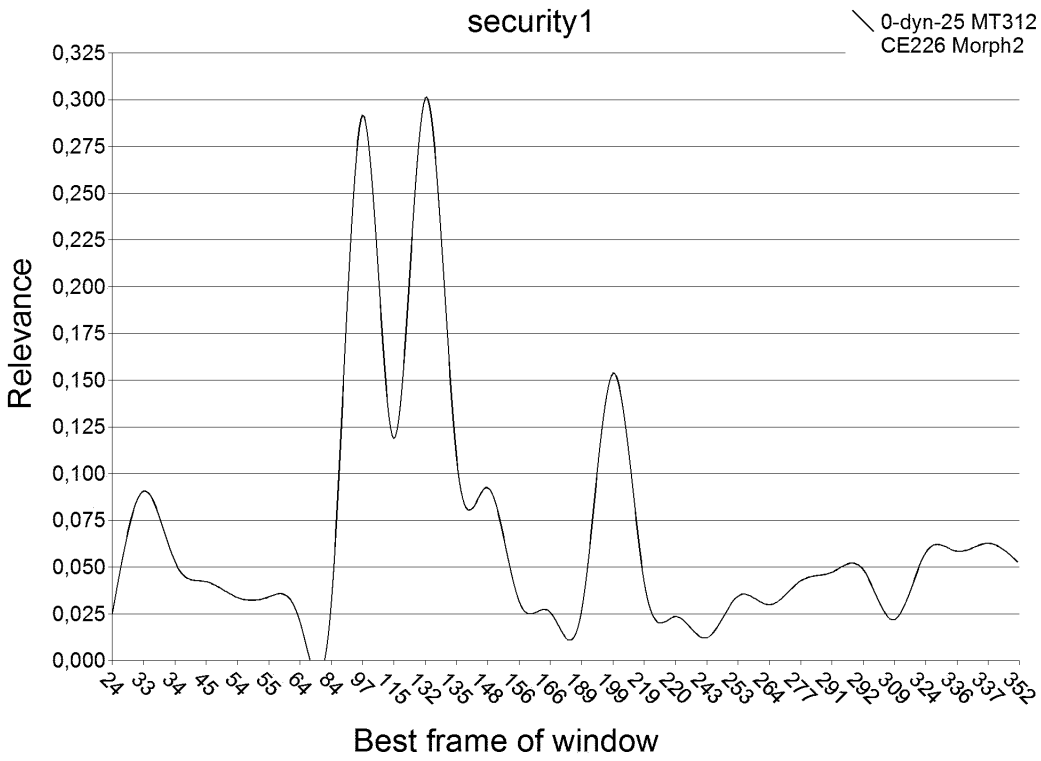

Figure 7.10: *Morph2* filtered video *security1* with 73 features and window relevance of the sum of the best three key frame relevances.

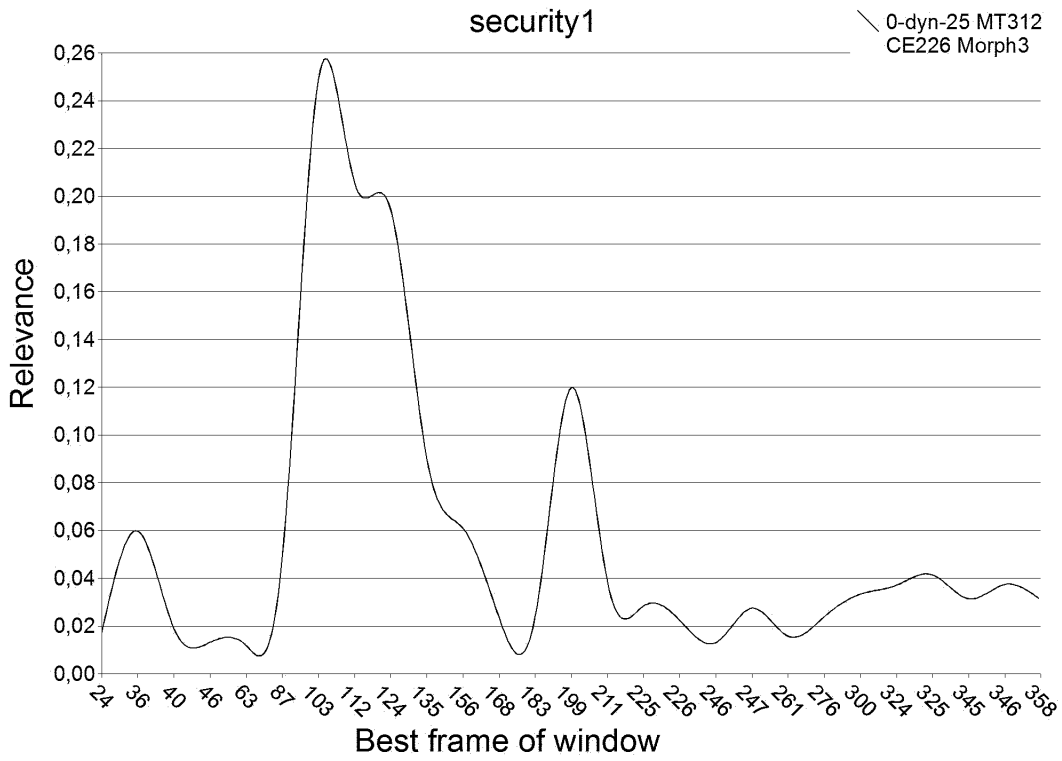

Figure 7.11: *Morph3* filtered video *security1* with 73 features and window relevance of the sum of the best three key frame relevances.

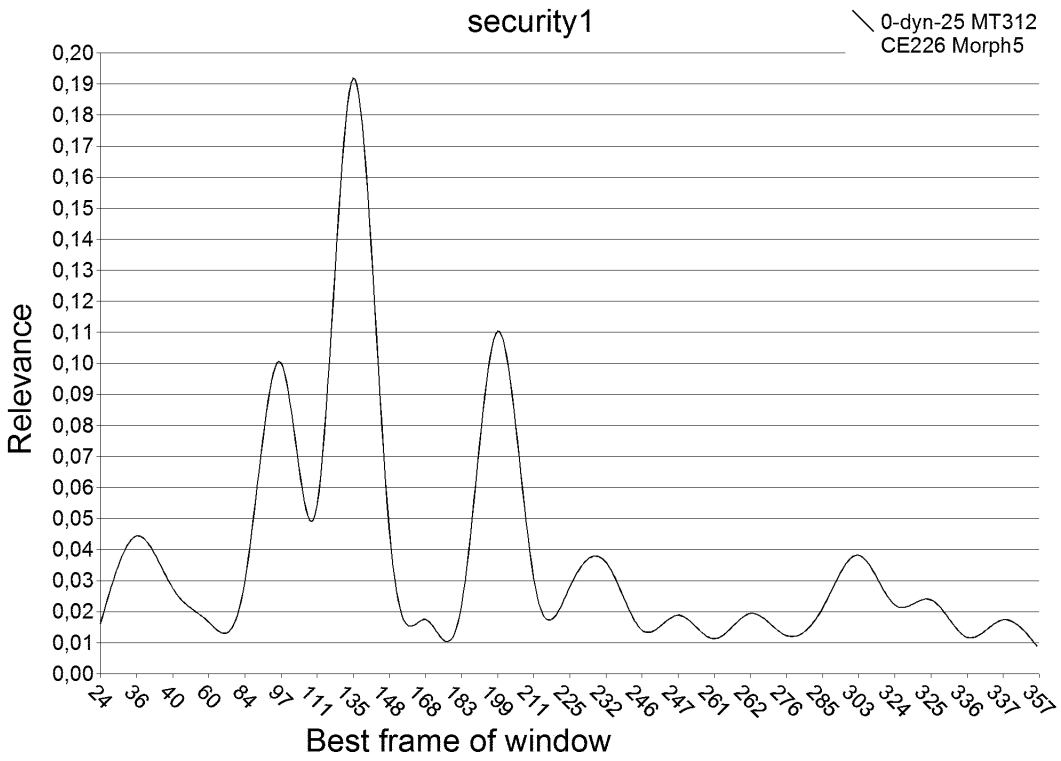

Figure 7.12: *Morph5* filtered video *security1* with 73 features and window relevance of the sum of the best three key frame relevances.

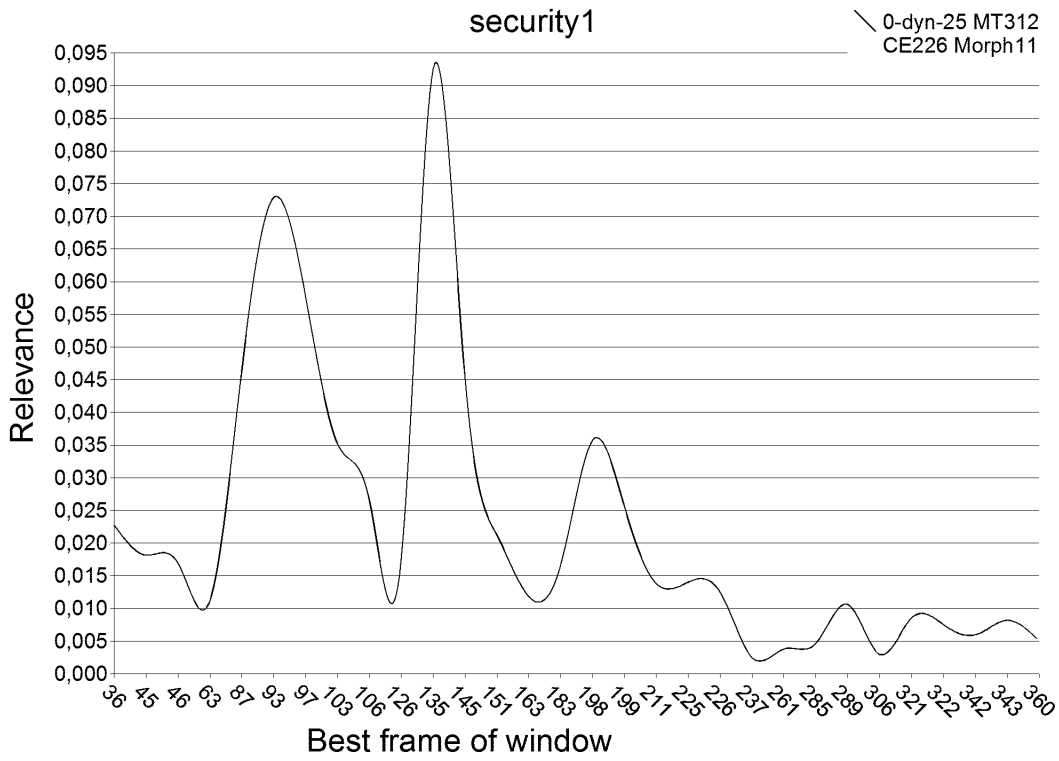

Figure 7.13: *Morph11* filtered video *security1* with 73 features and window relevance of the sum of the best three key frame relevances.

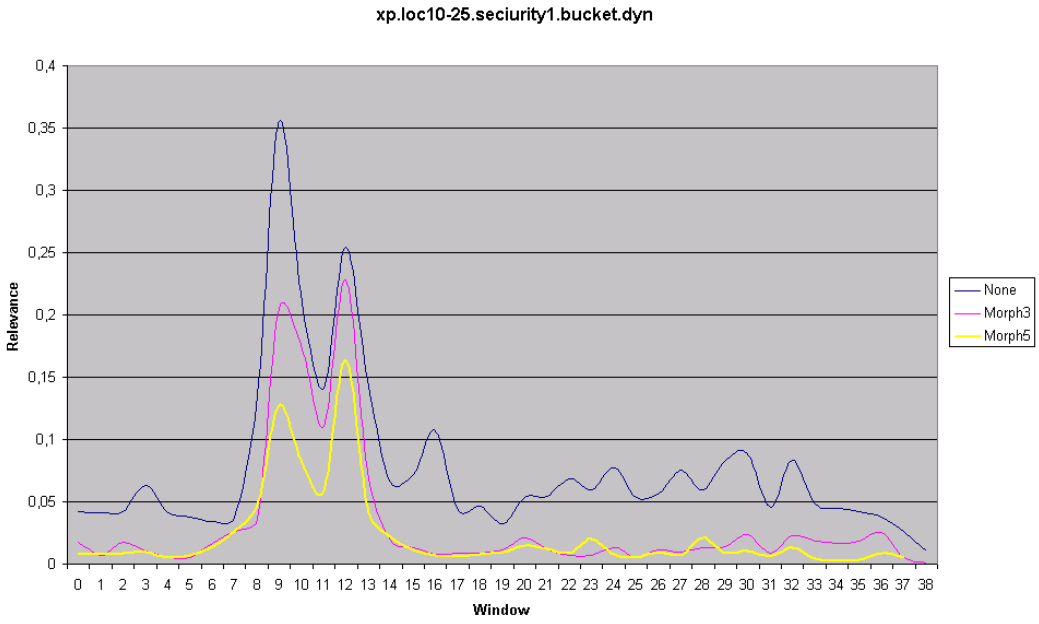

Figure 7.14: Relevance curve with different morphological filter width

Chapter 8

Results

8.1 Comparison

A complete comparison with results could be found on the homepage [14]. The results contain a comparison with our algorithm for all available videos as used by [22, 45].

8.2 Summary

The discrete curve evolution is a flexible greedy algorithm for video segmentation, to extract key frames and create video summaries. The flexibility is inside the ability to define frame descriptors for the appropriate purpose. The key frame measure functionality is not simply reduced to a metric, which results in a more flexibility 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 none key frames. This makes it possible to detect key frame on 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. 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.

The flexibility to analyze frames in bigger context is also the disadvantage of the key frame detection algorithm, which makes it unusable for the analysis

of video streams. We bypassed this disadvantage by performing a window analysis of the video stream to detect events. The window positioning algorithm is optimized to detect important once.

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, we have optimized them for best results. These optimization is improved by optimizing the frame descriptors implementing a filter for the frame descriptors.

8.4 Future

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

Also interesting is the research for usable frame descriptors, frame comparison metrics and the resulting cost measures for predefined key frame requirements. Interesting at this point are trained frame descriptors and comparison metrics which are used in CBIR [49].

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

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 availableat homepage [14].

A.1 self made video sets

A.1.1 Ground Truth

Important for the ground truth video's is that they are simple and the results must be identical for all tests (even person 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 move slowly over all papers, ending with the first. The movie is known as "MOV1".

Facts of video *Mov1*

video size	160 x 112 pixel's
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.

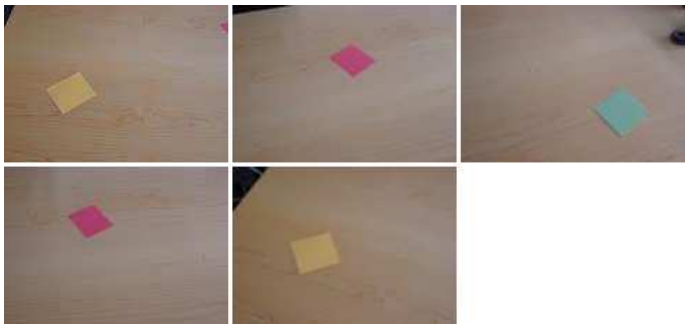

Figure A.1: Collation of the sequence "MOV1" with the expected ground truth results.

Dots

Movie "MOV00085" [15] 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 pixel's
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 results we got also shows the cleared board between or the that part between the groups which included more then one group. Figure A.2 shows the expected key frames.

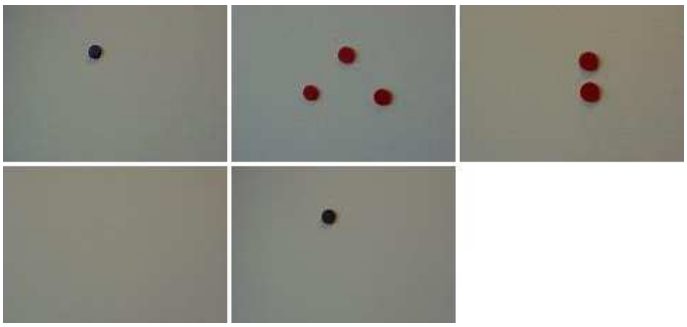

Figure A.2: Collation of the sequence “MOV00085” with the expected ground truth results.

Rustam

Movie “MOV3” [17] 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 weaves his left hand (ca. frame 36), sits still, and again weaves with his left hand (ca. frame 159). After a few seconds he weaves his right hand (ca. frame 241).

Facts of video *Mov3*

video size	320 x 240 pixel's
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	Weaving Rustam.

Shot description

1. Rustam weaves (from view of the observer) 2 times with his left hand and 2 times with his right hand. Rustams hand did not disappears between the last two weaves.

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 are recorded with two different handheld cameras. The first one recorded small videos with 25 fps. These videos are in reality recorded with approx. 8 fps, but every frame is tripled. The second camera recorded larger videos with 25 fps. These videos are in reality recorded with approx. 12 fps, but every second frame is doubled.

Security 1

Movie “security1” [18] is a low resolution video of 160x112 pixels and has 386 frames. With a frame rate of 25 fps it lasts for 15 seconds.

It 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, stand up and leaves the view of the camera at the right border. Image A.4 shows the expected key frames.

Facts of video *security1*

video size	160x112 pixel’s
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 the camera view from the right side. He squats and disappears the view at right side.

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

Security 7

Movie "security7" [19] is a low resolution video of 160x112 pixel's and has 386 frames. With a frame rate of 25 fps it lasts for 15 seconds.

It 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 disappears 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 appears. The guest disappears the camera view at the right side.

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

A.1.3 Television and existing video's

Halloween

This is one of the first not self made video's. It's one of the first minutes of a Video CD named "halloween". This video is only used for performance tests of and not used for comparison. Because this are not shot description and ground truth available.

Facts of video *Halloween*

video size	352 x 288 pixel's
ratio	9:11
number of frames	6182 frames
frame rate	29.7 fps
video length	206.3 seconds
number of shots	??? shot

Mr. Beans Christmas (large)

This video is the large version of "Mr. Beans Christmas"

Facts of video *Mr. Beans Christmas*

video size	352 x 240 pixel's
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 have 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 this shots put Mr. Bean the turkey into the view of the camera.

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

Mr. Beans Christmas (small)

The small version of the movie "Mr. Beans Christmas" (mrbeantu.mpg) is the identical version of the large version with an exception in the resolution which has a size of 112x80 pixels.

Figure A.6: Collation of the sequence “Mr. Beans Christmas” with the expected ground truth results

Facts of video *Mr. Beans Christmas*

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

(frames

A.2 Third party video sets

A.2.1 Rossiter et. al.

We used for our comparison experiments a set of 3 videos which are available at the hompage of Rossiter [45]. They contain a collation of clips of the motion picture "The Blade Runner", a collation of a few shots of through a house and collation of a scenes of the singer Kylie Minogue. A description of the videos could be found at [14].

Example information of "Kylie"

Information about the video *Kylie* as used in chapter 6. The video recorded with at a slower frame rate, which results fast playback of the content.

Facts of video *Kylie*

video size	192 x 144 pixel's
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	An other interview with Kylie Minogue.
6	190	205	Lead out of the BBC television.

The expected ground truth result matches 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 and contains one to seven shots. Some shots has also pans and blendings between them. The two downloaded videos comes from the university of Kansas The

first clip is scene of a football match scene match and the second clip is a collation 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 pixel's
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 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 own programs to verify and get knowledge of it. Even the programming language makes it possible for me to alter the algorithms I have implemented the algorithms in other programs based on C/C++.

The feature extraction is implemented in an MPEG player. The MPEG player is a C program inwhich I implemented the bucket extraction for each color space (RGB and YUV). The dominant colors are 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 is 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 is implemented in a separate program which is written in C++. For each kind of feature and filter 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 is also implemented in a separate program which is written in C/C++. The original version of this program is written by Daniel de Menthon. For each kind of feature is also an own program version written with a different version number. The distance measure for the buckets are directly implemented by a vector implementation. The distance measure for the dominant colors are implemented by a library which was supported by Hu. The different versions of the feature filter are described in appendix 2.1.

The player is a new application, implemented in C++ by Jan Erik Hoffmann. The player is written in C++. It's a graphical user interface (GUI) between the discrete curve evolution, the key frames and a 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 easy changed by a slider. The player also controls a commercial MPEG player with sound support. Clicking on an images skips to the same image in the MPEG stream at which the video could viewed.

The interaction between the programs are realized data files and scripts. The input and output for each program are data files like

- 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 performs

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

In appendix B.2 is described how these program are interacting with each other for the different algorithms. In appendix B.3 are the different file formats described which are produced by the programs.

B.1 Applications

B.1.1 Feature Extraction

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

B.1.2 Feature 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 with 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 with the dominant color frame descriptors which are used the curve evolution algorithm.
- *-o < output filename>*
  The parameter *< output filename >* specified 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 used to calculate an offset in the MPEG-stream on which a MPEG-viewer could start playing. This makes 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 is used for the curve evolution. It is possible with this option to perform the curve evolution only on a subset frames of the video. A

value larger or equal two will use all intra (I), predictable (P) and between (B) frames. An one will skip the between frames and a zero will only use the intra frames.

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

- *-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>* are used to simulated the curve evolution an endless video, which are used for the local analysis. Only the subset of the features with these frame boundaries are used for the curve evolution.

- *-mpg <MPEG filename>*
  The parameter *<MPEG filename>* is written in the output file as reference to the origin MPEG video. This information is used by our Smart Fast Forward Viewer to load the appropriated 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
3. a remote control for an external MPEG-player [40]

B.1.6 MPEG-Player

We used a commercial MPEG player [40] to show the video content from the Smart Fast Forward viewer. This decision was taken because other MPEG-player at that moment didn't support audio and suitable remote control mechanism.

B.1.7 Scripts

The complete creation process of the key frames, started with the feature extraction and ended with the key frame extraction is controlled by shell scripts to make an autonomous system with any unnecessary user interaction.

B.2 Application interaction diagram

The different kind of applications interacts with each other and exchanges data. The following two subsections shows the interactions between the application 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 where made in by several programs. Image B.1 shows the different steps of the key frame extraction. In the first step are the frame features extracted from the video sequence. The features are extracted to an intermediate file. The input file is only the video sequence. The output file format is described in chapter B.3.1.

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

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

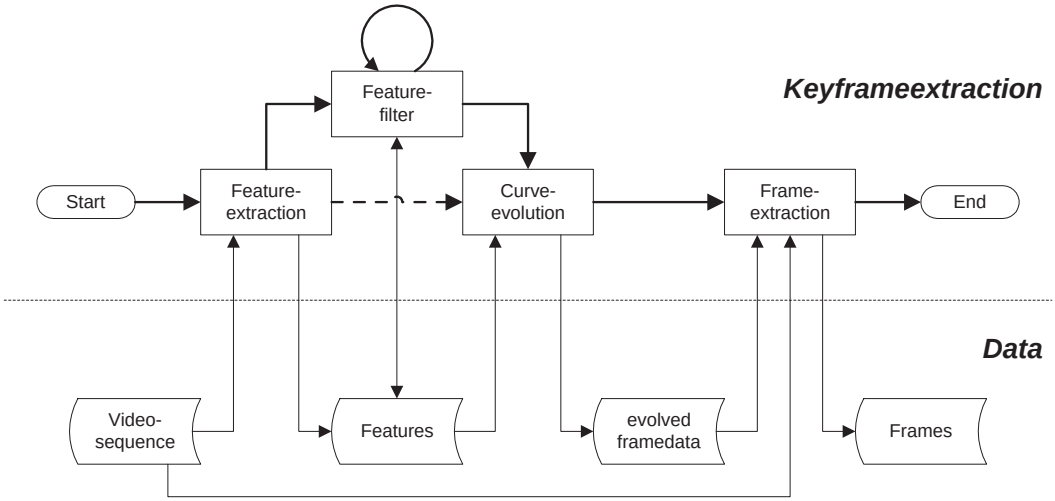

Figure B.1: Framework of the key frame extraction process.

In step five could the key frames and video sequence viewed with the smart fast forward viewer (SFF viewer) or with in an other application for example in an server/client environment. Image B.2 shows the framework for the SFF viewer and an example of an embedded client/server environment with a HTTP web server and a browser. An interactive example could be found on http://www.videokeyframes.de/ [14]. The input files for SFF player are 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 where made in by several programs.bImage 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.

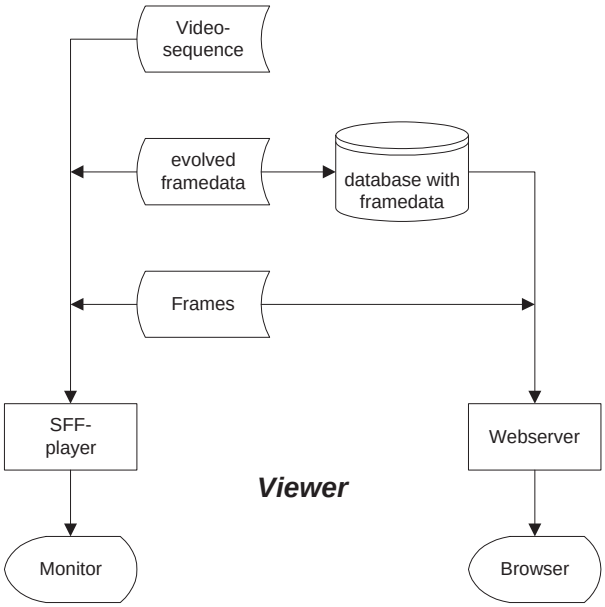

Figure B.2: Framework of the smart fast forward viewer

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 contains the features in space separated columns. Line one contains the number of frames. Line two contains number of columns which should be skipped. Line three contains the number of feature column. The first data column is the time (frame number). The following columns are ordered by the color component (red, green, blue or $Chrominance$, $Luminance_r$, $Luminance_b$), then the bucket number ordered from lower to the higher values and then bucket self with the bucket x coordinate, 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]
```

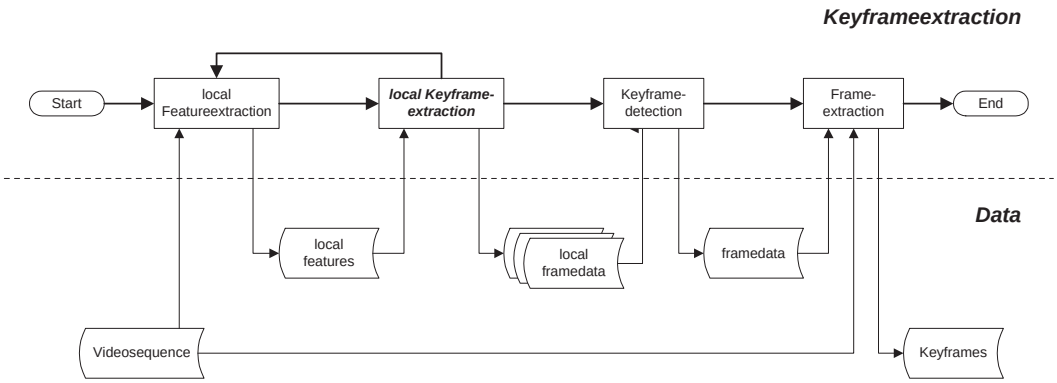

Figure B.3: Framework of local key frame algorithm

Frame Data:
<no. extended columns> * [extended data] <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 starts at column 6. (This is the column after "000000".) We have 37 features. This implies 4 buckets for each color. 16 is the x coordinate of the first bucket in the luminance color. 55 is the y coordinate of the first bucket in the luminance color. 997 is the area of the first bucket in the luminance color. 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 pixel's, which means that the image is very dark.

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

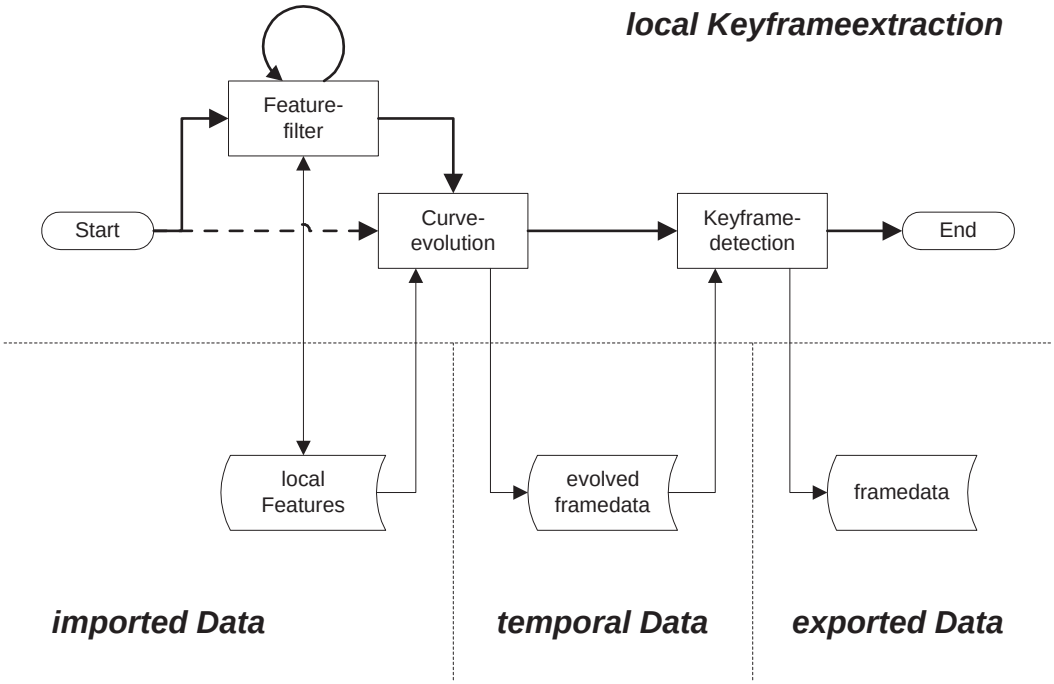

Figure B.4: Framework of the potential key frame detection part

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 contains the features in frame blocks 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 contains information of 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 to 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

1.1 Original example frame for a short video introducing visitor protection. 5
1.2 Example of three representative frames of the same video. . . 5

2.1 6 stages of the discrete curve evolution 11

4.1 The best five key frames with 145 features for "MOV1" . . . 35
4.2 The best six key frames with 145 features for "MOV1" 36
4.3 The best five key frames with 289 features for "MOV1" . . . 36
4.4 The five key frames of the ground truth video "Mov1". 37
4.5 Resulting five *key frames* of our experiments with normalized features. This are frames 1, 197, 263, 313 and 378 of video "Mov1". 37
4.6 Best six frames 1, 197, 263, 313, 319 and 378 of video "Mov1". 38
4.7 Best seven frames 1, 197, 263, 313, 319, 320 and 378 of video "Mov1". 38
4.8 Best eight frames 1, 55, 197, 263, 313, 319, 320 and 378 of video "Mov1". 38
4.9 Frames 313, 318 and 319 video "Mov1" showing the centroid problem. 40
4.10 Comparison of the X-component of the second U-centroid of the video "Mov1". 42
4.11 Comparison of the X-component of the third U-centroid of the video "Mov1". 42
4.12 Best five frames 1, 100, 200, 328 and 378 of video "Mov1" with the weighting modification for the centroid coordinates. . . . 43
4.13 Best six frames 1, 100, 127, 200, 328 and 378 of video "Mov1" after the weighting modification. 44
4.14 Best seven frames 1, 100, 127, 200, 236, 328 and 378 of video "Mov1" after the weighting modification. 44
4.15 Best eight frames 1, 100, 127, 200, 236, 259, 328 and 378 of video "Mov1" after the weighting modification. 45

4.16 Weighting of pixels for associated centroids as implemented. . 46
4.17 Proposal weighting of pixels for associated centroids bins. . . 46
4.18 Key frames (out of 20) from the hospital floor scene (approx. 19") of the video "halloween". 48
4.19 Diagram with color intensity over the time. Step width 500 frames. (Approx. 207") 48
4.20 Diagram of a morphological (un)filtered dynamic weighted feature from video "Mov3". 52
4.21 The best three key frames for "MOV3" 56
4.22 The best six key frames for "MOV3" 56
4.23 Intensity composition representation of Mov1 video clip. . . . 58
4.24 Color composition representation of Mov1 video clip. 58
4.25 Key frame result of video Mov1 with 5 frames. We used Dominant Colors as frame descriptors, without filtering and relevance measure 4.1 60
4.26 Key frame result of video Mov3 with 7 frames. We used Dominant Colors as frame descriptors, without filtering and relevance measure 4.1 61
4.27 Intensity Mov1 without any filter 61
4.28 Color Mov1 without any filter 62
4.29 Intensity Mov1 - Morph3 62
4.30 Color Mov1 - Morph3 63
4.31 Intensity Mov1 - Morph5 63
4.32 Color Mov1 - Morph5 64

5.1 Figure with the best five YUV frames of Mov1 without filtering 66
5.2 Figure with the best five RGB frames of Mov1 without filtering 66
5.3 Centroid based buckets with 37 frame descriptors, without filter. 68
5.4 Dominant Colors, without filter. 68
5.5 Centroid based buckets with 37 frame descriptors, without filter. 69
5.6 Dominant Colors, without filter. 69
5.7 Centroid based buckets with 37 frame descriptors, without filter. 70
5.8 Dominant Colors, without filter. 70
5.9 Centroid based buckets with 37 frame descriptors, with filter *morph5*. 71
5.10 Dominant Colors, with filter *Morph5*. 71
5.11 Result with the best nine frames of the large version of the video Mr. Beans Christmas with unfiltered frame descriptors. 73

5.12 Result with the best nine frames of the small version of the video Mr. Beans Christmas with unfiltered frame descriptors. 73
5.13 Result with the best nine frames of the large version of the video Mr. Beans Christmas with filtered frame descriptors. . 73
5.14 Result with the best nine frames of the small version of the video Mr. Beans Christmas with filtered frame descriptors. . 74
5.15 Dominant Color Availability image for the large version "Mr. beans Christmas". 75
5.16 Dominant Color Availability image for the small version "Mr. beans Christmas". 75
5.17 Dominant Color Intensity image for the large version "Mr. beans Christmas". 76
5.18 Dominant Color Intensity image for the small version "Mr. beans Christmas". 76
5.19 Result with the best nine frames of the large version of the video Mr. Beans Christmas with unfiltered dominant colors. . 77
5.20 Result with the best nine frames of the small version of the video Mr. Beans Christmas with unfiltered dominant colors. . 77
5.21 Result with the best nine frames of the large version of the video Mr. Beans Christmas with filtered dominant colors. . . 77
5.22 Result with the best nine frames of the large version of the video Mr. Beans Christmas with filtered dominant colors. . . 78

6.1 Expected ground truth results for video *beachmov* 85
6.2 Drews key frame result for video *beachmov* 85
6.3 Our result with the *DCE* for video *beachmov* 85
6.4 Precision and recall for Drew's and our results 86

7.1 A relevance Curve of "Mov3" 89
7.2 Frames at the local maxima of figure 7.1 89
7.3 Example with relevance values as produced by our local curve evolution process. The used video *security1* 90
7.4 Video security1. Dynamic weighted centroids with 37 frame descriptors, no filtering. Window position starts of the best key frame of the previous window. The window relevance is the relevance of the best key frame in the window. 98
7.5 Video security1. Dynamic weighted centroids with 37 frame descriptors, no filtering. Window position starts of the best key frame of the previous window. Window relevance is the sum of the relevances of the three best key frames. 99

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 maximum of the key relevances. 100
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 maximum of the key relevances. 101
7.8 *None* filtered video *security1* with 73 features and curve evolution *CE226*. 102
7.9 *Morph1* filtered video *security1* with 73 features and window relevance of the sum of the best three key frame relevances. . 103
7.10 *Morph2* filtered video *security1* with 73 features and window relevance of the sum of the best three key frame relevances. . 104
7.11 *Morph3* filtered video *security1* with 73 features and window relevance of the sum of the best three key frame relevances. . 105
7.12 *Morph5* filtered video *security1* with 73 features and window relevance of the sum of the best three key frame relevances. . 106
7.13 *Morph11* filtered video *security1* with 73 features and window relevance of the sum of the best three key frame relevances. . 107
7.14 Relevance curve with different morphological filter width . . . 108

A.1 Collation of the sequence "MOV1" with the expected ground truth results. 112
A.2 Collation of the sequence "MOV00085" with the expected ground truth results. 113
A.3 Collation of the sequence "MOV3" with the expected ground truth results. 114
A.4 Collation of the sequence "security1" with the ground truth results 115
A.5 Collation of the sequence "security7" with the expected ground truth results 116
A.6 Collation of the sequence "Mr. Beans Christmas" with the expected ground truth results 118
A.7 Collation of the sequence "Kylie" with the expected ground truth results. 119
A.8 Ground truth results of video "beachmov" 120

B.1 Framework of the key frame extraction process. 126
B.2 Framework of the smart fast forward viewer 127
B.3 Framework of local key frame algorithm 128

B.4 Framework of the potential key frame detection part 129

Bibliography

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

[2] 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.

[3] Search engine. http://www.altavista.com.

[4] Michèle Bassevile and Igor V. Nikiforov. Detection of abrupt changes: Theory and application. http://. Previously published by Prentice-Hall, Inc.

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

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

[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] S.-W. Chen C.-Y. Fang 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.

[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] K. Salamy D. Edgington, W. Dirk, C. Koch, M. Risi, and R. Sherlock. Automated event detection in underwater video. In *Proceedings MTS*. IEEE Oceans, 2003.

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

[12] H. Delingette D. Rey, G. Subsol 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.

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

[14] D. de Wildt and L. J. Latecki. Project homepage for temporal video segmentation. http://www.videokeyframes.de/.

[15] D. de Wildt and L. J. Latecki. Video "mov00085". http://www.videokeyframes.de/Mov00085.mpg.

[16] D. de Wildt and L. J. Latecki. Video "mov1". http://www.videokeyframes.de/Mov1.mpg.

[17] D. de Wildt and L. J. Latecki. Video "mov3". http://www.videokeyframes.de/Mov3.mpg.

[18] D. de Wildt and L. J. Latecki. Video "security1". http://www.videokeyframes.de/seciurity1.mpg.

[19] D. de Wildt and L. J. Latecki. Video "security7". http://www.videokeyframes.de/seciurity7.mpg.

[20] Daniel de Wildt and Longin Jan Latecki. Comparison between a clustering and the discrete curve evolution algorithm. http://www.videokeyframes.de/Information/Doc/GlobalResults.pdf.

[21] A. Rosenfeld D.F. DeMenthon, L.J. Latecki 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.

[22] 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/.

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

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

[25] 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).

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

[27] W. Franklin G. Nagy, T. Zhang, 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.

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

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

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

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

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

[34] Language and Media Processing Laboratory; University of Maryland;. 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 and R. Lakämper. Convexity rule for shape decomposition based on discrete contour evolution. *Computer Vision and Image Understanding*, 73:441–454, 1999.

[37] 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.

[38] J. P. Armspach M. Bosc, F. Heitz, 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.

[39] http://bmrc.berkeley.edu/frame/research/mpeg/mpeg-play.html.

[40] mtv - mpeg player for linux & unix. http://www.mpegtv.com/.

[41] Mozilla Organisation. http://www.mozilla.org/products/firefox/.

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

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

[44] 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.

[45] Rossiter. http://www.cs.ust.hk/~rossiter/mm_projects/video_key_frame/video_key_frame.html.

[46] J. Serra. *Image Analysis and Mathematical Morphology.* Academic Pres, 1982.

[47] 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.

[48] Mr. bean's x-mas.

[49] Homepage. http://viper.unige.ch/demo/php/demo.php.

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

[51] G. Wyszecki and W. S. Stiles. *Color Science: concepts and methods, quantitative data and formulae.* John Wiley and Sonsa, 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.

[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.