Technical University of Crete
Department of Electronics Engineering & Computer Engineering

Methods for Segmentation of Uncompressed Video Using Intensity and Motion Histograms

by

Stavros Vagionitis
October 2003

Accepted on the recommendation of

Prof. Dr. Michalis Zervakis,	Thesis Advisor
Assoc. Prof. Dr. Euripides Petrakis,	Co-Examiner
Prof. Dr. Stavros Christodoulakis,	Co-Examiner

ACKNOWLEDGEMENTS

I wish to express my sincere appreciation to Professor Michalis Zervakis for his guidance during the implementation of this thesis and his assistance in the preparation of this manuscript

In addition, special thanks to Associate Professor Euripides Petrakis for his invaluable help and suggestions of improvement this thesis.

I would also like to thank Professor Stavros Christodoulakis for the evaluation of this work and his participation in the supervisory committee.

Last, but not least, I wish to thank my parents and especially my brother Panos, for their love, support and encouragement as well as my friends for sharing my thoughts, worries and expectations during these years.

Table of Contents

TABLE OF CONTENTS **3**

CHAPTER 1 ..... **7**

INTRODUCTION 7
PROBLEMS IN DIGITAL VIDEO DATA MANAGEMENT 7
VIDEO DATA MANAGEMENT SYSTEMS, A GENERAL IDEA 9
NATURE OF VIDEO DATA 9
UNIQUE CHARACTERISTICS OF VIDEO 11
APPLICATIONS OF VIDEO 13
EXAMPLES OF VIDEOS 14

CHAPTER 2 ..... **18**

MULTIMEDIA AUTHORING & VIDEO PRODUCTION 18
TYPICAL EXAMPLES 18
Multimedia Authoring Example *19*
Video Production Example *19*
DESIGN OF DATA MODEL 19
Step 1: Choice of Video Interval *19*
Step 2: Design of Temporal Relationships *20*
Step 3: Choice of Video Features *20*
THE DATA MODEL 21
VIDEO PROCESSING APPROACHES, AN INTRODUCTION 21
VIDEO PRODUCTION 22
PHASES IN VIDEO PRODUCTION 22
PHASE 1: CONTENT CREATION: DIRECTION 23
PHASE 2: CONTENT CAPTURE: SHOOTING & RECORDING 25
PHASE 3: CONTENT ORGANIZATION: EDITING & MIXING 28
THE SHOT FRAMING PROCESS IN DETAIL 29
THE EDITING PROCESS IN DETAIL 32

CHAPTER 3 ..... **35**

VIDEO PROCESSING TECHNIQUES: A REVIEW OF UNCOMPRESSED VIDEO SEGMENTATION TECHNIQUES 35
SHOT CHANGE DESCRIPTION 35
NOTATIONS 36
PIXEL-BASED METHODS 37
Pixel comparison between two successive frames *37*
Pixel intensity time variation *39*
HISTOGRAM-BASED METHODS 41
Histogram Difference *41*
Weighted Difference *44*
Histogram Intersection *45*
Use of c^2 *Test* *46*
Similarity measures between normalized Histograms *47*
BLOCKED-BASED METHODS 48
Block Similarity *48*
Histogram Comparison *50*
Combination of Histogram differences & Likelihood rate *53*
Use of Neighborhood color ratio *53*

Evolution of block dissimilarity *54*
Temporal & Spatial sub-sampling *54*
FEATURE-BASED METHODS 55
Moment invariants *55*
Edges *56*
Feature points *58*
Planar points *59*
Color Transitions *59*
Transition modeling *59*
Bayesian approaches *62*
Statistical approaches *63*
Hidden Markov models *65*
MOTION-BASED METHODS 66
Global motion *66*
Motion vectors *67*
Optical flow *68*
Frequency domain correlation *68*

CHAPTER 4 ..... **69**

VIDEO SEGMENTATION & MODELING DIGITAL VIDEO 69
VIDEO EDIT MODEL 70
EDIT EFFECT MODEL 71
PROBLEM DEFINITION OF VIDEO SEGMENTATION 73
VIDEO SEGMENTATION USING PRODUCTION MODEL BASED CLASSIFICATION 73
CUT DETECTION 75
CHROMATIC EDIT DETECTOR 76
Chromatic Scaling *76*
FADES & DISSOLVES AS CHROMATIC SCALING 78
Limitations of Chromatic Edit Detector *81*
SPATIAL EDIT DETECTOR 82
A MEASURE OF IMAGE UNIFORMITY 84
COMPUTATIONAL REQUIREMENTS OF FEATURE DETECTORS 85
FEATURE DETECTOR SUMMARY 87
CLASSIFICATION & SEGMENTATION 88
ERROR MEASURES FOR VIDEO SEGMENTATION 90
SEGMENTATION ERROR CLASSES 90
SEGMENT BOUNDARY ERRORS: E_{sb} 91
SEGMENT CLASSIFICATION ERRORS: E_{sc} 92

CHAPTER 5 ..... **93**

FIVE APPROACHES FOR UNCOMPRESSED VIDEO SEGMENTATION 93
HISTOGRAM BASED METHODS 97
Twin Comparison *97*
Difference metric selection 97
Twin-Comparison approach for shot detection 101
Selection of Thresholds 105
Sliding Window Method *108*
Difference metric selection 108
Sliding Window (SW) approach for shot detection 108
Selection of Thresholds 111
Adaptive Method *114*
Difference metric selection 114
Adaptive approach for shot detection 114
Selection of Thresholds 117

A different value for $\underline{c}$ variable 121
INTRODUCTION TO MOTION ESTIMATION 124
Motion Models *125*
Motion Representation 125
Region of Support for Motion Representation 129
Interdependence of Motion and Image Data 132
Estimation Criteria *133*
Search Strategies *135*
Interpretation of Histogram of Motion Vector's (MV) Angle 139
Sliding Window Method for MVs *150*
Sliding Window (SW) approach for shot detection 150
Selection of Thresholds 152
Adaptive Method for MVs *155*
Adaptive approach for shot detection 155
Selection of Thresholds 157
A different value for $\underline{c}$ variable 159

CHAPTER 6 160

EXPERIMENTAL RESULTS 160
ROC Curves *160*
Internal Response Probability Density Functions 161
The Role of Criterion 162
The Receiver Operating Characteristic 164
The Role of Signal Strength 165
Varying the Noise 166
ROC CURVES OF ALGORITHMS 167
Category documentary *167*
Category news *172*
Category animation *175*
Category sports *177*
Overall Results (All Videos) *181*
CONCLUSION AND FUTURE WORK 184

REFERENCES 185

REFERENCES FOR CHAPTER 1 185
REFERENCES FOR CHAPTER 2 185
REFERENCES FOR CHAPTER 3 185
REFERENCES FOR CHAPTER 4 190
REFERENCES FOR CHAPTER 5 191
REFERENCES FOR CHAPTER 6 192

Chapter 1

Introduction

A team of surgeons preparing for a complex colon cancer surgery procedure decides to review the video files of similar procedures performed the last three years in the hospital. They search through the huge collection of video files and retrieve the relevant videos. During the review process different portions of the video are accessed based on the nature of the procedures being performed in the video with frequent comparisons between the cases (video files). While reviewing one of the video files, a surgeon recalls a different procedure that was performed on a similar anomaly. A search through the video collection yields another set of video files. At the end of the review process the team has all the background information necessary to perform the procedure. The complete review process takes up an entire day for the surgery team. More than half the time spent for review is consumed by the process of searching for the appropriate video files and navigating through the files looking for particular techniques.

The example is meant to illustrate the powerful nature of video as a medium of information representation and the effort involved in dealing with large collections of video. Utilizing video collections effectively requires the ability to access video by its content. The ability to search through video collections based on the information contained in the videos, the visual events, audio events and many such interesting patterns is termed content based access of video. There are many different applications which require such content based access of video.

A computer based system which provides the functionality necessary to manage collections of video while providing the capability of content based access of video is called a Video Data Management System (VDMS). Using a video data management system, the surgery team in the example would have been able to complete the review in less than half the time.

Problems In Digital Video Data Management

The problems involved in digital video data management arise due to three factors. The first set of problems arises from the nature of the task, ***data management***. The second set of problems is due to ***the non alpha***

numeric, ***spatio-temporal*** and ***audio-visual*** nature of video data. The final set of problems arises due to the ***digital medium of video storage***.

1) ***Data management***: The task of managing any collection of data involves four basic problems, *data modeling*, *insertion*, *organization* and *retrieval*. *Data modeling* entails the choice of data aspects most relevant to the application. Adding new data units to a collection is termed *data insertion*. *Data organization* deals with the task of suitably arranging data units with reference to one another to facilitate easy and fast access to the data. The process of requesting the data (querying for the data) and extracting it from the collection is called *data retrieval*. All the tasks mentioned above have been addressed in the context of alpha numeric data, but many of techniques applicable to alpha numeric data do not apply to video due to its non alpha numeric, spatio-temporal, audio-visual nature of video.
2) ***The non alpha numeric, spatio-temporal and audio-visual nature of video data***: The non alpha numeric nature of video makes it opaque to computer systems thus any computer system that deals with video has to manage video through an associated alpha numeric representation of video. Designing task directed representations of video is called *video data modeling*. The nature of video makes the task of incorporating new video units into an existing collection of video very cumbersome and tedious. The manual effort involved in adding to a collection of video becomes formidable as the volume of data to be managed grows. Introducing new data items into an existing collection is termed *video insertion*.
3) ***Digital medium of video storage***: The digital storage of video data provides the freedom of random access to video data at the level of individual frames of video. This increase in the accessibility of video data as compared to the traditional storage medium of video tapes gives rise to the problem of representing video at very fine temporal intervals. The fine temporal granularity of video compounds the video insertion problem making it more labor intensive.

The problem of organizing video data is more complex than that of organizing alpha numeric data. This is primarily due to the fact that the concept of equality of two data items is invalid for non alpha numeric media like video. Equality translates to similarity in the case of non alpha numeric data. The design of suitable representations can make the problem of video data organization comparable to the traditional data organization problem. The audio-visual properties of video and the spatio-

temporal nature of video give rise to new problems in specifying the video items to be retrieved from the collection. Designing methods of specifying audio-visual, spatio-temporal queries and techniques for processing such queries is termed *video data retrieval*.

Video Data Management Systems, A General Idea

Database management systems are a part of normal day to day operation of computer systems. The type of databases available on computers today is mainly limited to alpha numeric databases. In the recent past there has been some research into image data management systems. A few such systems are commercially available. With the coming of age of digital video technology video data promises to be an ubiquitous medium of information representation in computer systems. The need for managing video data on computer systems is growing. However there has been very little research on video data management systems. It is important to mention a few things about the video itself. In the following paragraph it will be presented the content of the video.

Nature Of Video Data

It is the right time to pose the question. *What is in video?* Video is an audio visual medium of information presentation. The Figure 1.0 below shows a very high level view of the content of video. In this figure the content of video has been grouped into two types:

- **Information Content**: This is the message or information conveyed by the video. For example, after watching a news story about a crime, the viewer has acquired information from the video about several aspects of the crime, like what was the crime, where did it occur, who were the victims etc. This information was conveyed to the viewer via the audio visual medium of video.
- **Audio Visual Content**: This is the audio visual content of video. This includes the video clips and audio signals. For example, in the news story on crime, the viewer sees the location of the crime, hears the associated sound track. Depending on how the video was produced, the same information content can be presented through infinitely many different audio-visual presentations.

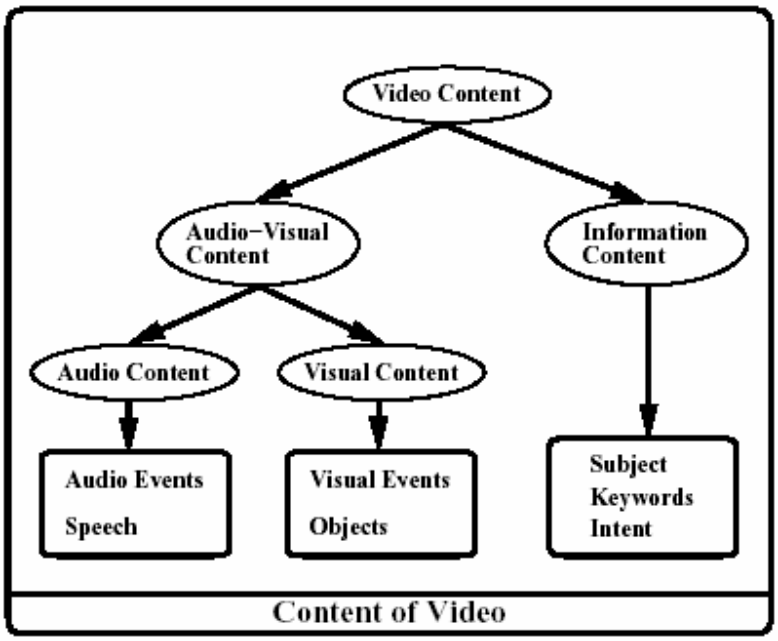


Figure 1.0: Content of Video

The key distinction between the *information content* and *audio visual content* is the amount of contextual information and knowledge required to extract each of these contents. The information content of video requires the use of contextual information along with a large body of associated knowledge where as the audio visual content is primarily oriented towards the aural and visual senses and does not require an understanding of the information. The audio visual content can be extracted from video based on the capabilities like speech recognition, image understanding and interpretation.

The management of video from an *information content* perspective has similarities to managing textual information. The management of information is addressed by the library and information sciences community. They address the issues at a very coarse grain. For example in conventional libraries the unit of information is a book, and the access patterns are by the title, author, subject etc. However the granularity of content based access envisioned for in video data management is much finer. For example, some queries, access video by content at the granularity of scenes and shots. This granularity of access corresponds to accessing books at the level of chapters and sections. Thus managing video from an information content perspective has some degree of overlap with the traditional information management problem.

The *audio-visual content* of video arises either as a visualization of a message or as a log of audio-visual activity. For example, in the case of

feature films, the *audio-visual content* is created explicitly with the purpose of visualizing the script, where as in a video taken from a security camera in a shopping mall, the video is just a log of the audio-visual activity within the range of the camera. Many different types of audio-visual events can be extracted from video. The visual medium can be used for recognizing objects, tracking objects over time, recognizing temporal events etc. The audio track can be used to recognize words and sentences, unusual sound events etc. The task of managing video based on its *audio-visual content* is a largely unaddressed research problem.

In summary, a video can be considered as an audio-visual representation of information. The information may be generated for the purpose of making the presentation or may be generated as a part of some other process. The *audio-visual content* is directly extractable from the video with minimal external knowledge, where as the information content requires the use of significant amount of contextual knowledge.

Unique Characteristics Of Video

Another important question that has to be answered is *How is video different from other classes of data?* This question is answered by classifying data into two categories, namely alphanumeric data and non alphanumeric data and comparing them. The definition of the classes and a list of comparison criteria are presented here.

- **Alphanumeric Data**: The data in this class is generated from a finite set of symbols. The symbols may be drawn from some given finite set of languages. For example, the data in a telephone directory is composed of a finite set of symbols. The symbols are drawn from the valid set of telephone numbers in a city, the possible set of names, and the valid set of addresses in the city. There are several other examples of alphanumeric data, like free text data, computer programs, product data etc.
- **Non Alphanumeric Data**: This is non symbolic data, in other words the data is not derived from a finite set of symbols. Typical examples in this class include signal level data like images, speech signals, ECG Data, MRI Data, Video Data, Weather Data etc. The key difference between symbolic and non symbolic data is that the former is essentially generated by human agency as compared to the later which is gathered by some automatic means.

The criteria for comparing alpha numeric and non alpha numeric data are listed. A short explanation of each of these criteria is presented, followed by a discussion of how the criteria vary for different types of media. The different types of media considered are a structured record, free text, line drawings, an image and a video.

- **Resolution**: The resolution of a particular media is the detail that the media provides. For example, a textual description of a scene has much less detail that an image or a video of the scene. Non alpha numeric media provides much higher detail than the alphanumeric media.
- **Production Process**: The source of data goes from being a finite symbolic alphabet as in the case of the structured record, to being generated from larger and larger sets of symbols in the case of line drawings to an infinite symbol set in the case of images and video. Alphanumeric media originates through direct human agency, i.e., it originates from a human being. Non alphanumeric data originates from sensors, i.e., it is recorded by some type of sensor. For example, a camera, a microphone, a MRI imager etc.
- **Ambiguity of Interpretation**: This criterion is a measure of the number of interpretations derivable from the data. The number of interpretations for video is much larger than for a structured data record. The interpretation process depends on the interpreting agent. For example, on viewing the video clip, a person may register the color properties of the video, while another may take note of the aural properties. But given a structured record of information the ambiguity is limited.
- **Interpretation Effort**: This is a measure of the computational effort required to interpret a given unit of information. The effort required to interpret a structured record is much smaller than that required to interpret an image or a video.
- **Data Volume**: In terms of digital storage, the volume of video is about seven orders of magnitude larger than a structured data record.
- **Similarity**: The idea of similarity between two units of information is very precisely defined in the case of structured records. This concept becomes less and less well defined as the media resolution grows higher.

Table 1.0 below summarizes the comparison between alphanumeric and non alphanumeric data. The voluminous nature of video data, its higher degree of interpretation ambiguity, interpretation effort and ill defined concept of similarity pose the most significant challenges in managing video. The greatest advantage of video as a medium is the

resolution of information, the expressive power of video surpasses that of any other medium. Video is a very natural form of communication given the audio-visual sensory capabilities of human beings.

Criteria	Alphanumeric Data	Non Alphanumeric Data
Resolution	Low	High
Production Process	Finite Symbol Set	Infinite Symbol Set
Interpretation Amguity	Low	High
Interpretation Effort	Low	High
Data Volume	Low	High
Similarity	Well Defined	Ill Defined

Table 1.0: Comparison between Alphanumeric & Non Alphanumeric Data

Applications Of Video

This section presents a detailed study of different usages of video. The goal is to understand that video plays an important role in different perspective of our life. Each of the applications is analyzed from several perspectives like *video intent*, *video content*, *video production* and *video usage*. The example applications used are feature films and news videos.

- ***Video Intent***: *Why was this video made?* This questions the purpose of producing the video. The answer to this question provides clues into the structure of video, content of the video and the organization.
- ***Video Content***: *What is the typical content?* This question probes the issue of video content for the particular class of videos. Depending on the domain of the video the predictability of the content varies.
- ***Video Production***: *How was the video made?* This addresses the issue of the nature of the production process. The answer to this question provides information about the syntactic structure, the audio-visual properties etc. Video production is viewed from the following perspectives:

- ***Script Control***: Video can be a visualization of a certain script or an audio-visual log. Script control is a measure of the degree of visualization control. For example, a feature film has multiple filming of the same scene where as a sporting event video must capture the event as it occurs.
- ***Filming Control***: The key process in making a video is the step which captures images onto the media. This involves an environment in which the video is made, a subject and the video filming parameters. Filming control is a measure of the degree of control exercised by the film maker on these parameters.
- ***Composition Control***: A video can be the result of composing many individual pieces of film footage into a temporal composition. Using this as criteria for classifying videos provides a broad classification of videos.
- ***Channel Control***: Since video is an audio-visual medium, the relative information content in the two channels can be used as criteria to classify videos. This provides clues into the best techniques for indexing a class of videos.

- ***Video Usage***: The way a video is used dictates the queries that arise in the database context. Different users of video have different query requirements. The users typically examine the video from a certain perspective. The examples illustrate this issue.

Examples Of Videos

Example 1: *Feature Films*

Video Intent: The main purpose of this class of videos is to provide entertainment. The director of the film has a message to convey to the audience. Video is used as a communication channel to communicate this message. The feature film can be considered the directors visualization of the script.

Video Content: The content of feature films is very widely varied. There are many different types of feature films. The classification scheme used for feature films is referred to as film genre' s. Extensive studies on the classification of films based on content can be found in literature. Western Movies and War Movies are examples of

classes of movies. Given a particular class of movies the content is predictable. For example, in the case of war movies, a number of things are known the subject of the movie is a war; typically it would contain a number of battle scenes.

Video Production: The production of feature films is a planned and controlled process.

> **Script Control**: The degree of control that the film maker has on the exact message to be conveyed is very high. The script can be altered and hence the nature of the video produced is very structured.
>
> **Filming Control**: The degree of control that the film maker exerts on the filming process is very high. All aspects of filming, the location, the action, and the cinematography are planned and controlled.
>
> **Composition Control**: A significant portion of film production is done through the use of editing, which plays a very important role. The degree of control on the editing process is very high.
>
> **Channel Control**: This parameter is completely under the control of the film maker. Some films have a strong visual orientation while others tend to be dominated by aural information.

Video Usage: A collection of feature films has many different groups of users. These user groups have certain queries about feature films. The following is a list of users. The list is not exhaustive but does cover the major categories of users.

> *Film Viewer*: This is a set of users who use feature films just for the purpose of entertainment. They are typically interested in a particular type of film.
>
> *Film Critic & Analysts*: This set of users, views films with the purpose of evaluating the films from many different perspectives like, general appeal, artistic appeal, director evaluation, actor evaluation, cinematographic evaluation, special effect evaluation etc. In addition to being able to locate films, this group of users will require finer grain access to the films.

Film Database Managers: These are the group of people who own and operate movie rental organizations.

Example 2: *News Video*

News video here stands for a regular television news bulletin. A news bulletin like the BBC for example.

Video Intent: The purpose of a news video is to convey the news to its audience. The news here is defined as the events that occurred over a given duration of time as observed by a certain team of people. A news video reports the events along with the necessary background information to provide a complete and understandable presentation.

Video Content: The content of a news video is unrestricted. But a news cast has a definite structuring. For example, news casts begin with the main points and have segments dedicated to politics, sports, social stories, science etc. All the news segments are presented to the viewer by the anchor person, and each individual segment has a structuring of its own, with a reporter anchoring the individual segment.

Video Production: The production of television news is less controlled than a feature film.

Script Control: The degree of control is limited to the structure of the news. Specifically, the stories that are reported on a news bulletin are controlled. However the exact content of the stories and their presentation are less controlled. With the use of satellites, news bulletins incorporate live news reports in which the degree of control on the content is very limited. Thus as compared to a feature film the degree of control exerted on the actual message conveyed by a news bulletin is smaller.

Filming Control: The environments typically involved in a news bulletin include the studio environment which tend to be well controlled and the news location environment which tends to be less controlled. The subjects involved in news reports include the news reporters and anchor persons etc, who are used to presentation and other subjects where the degree of

control is much lesser. The cinematographic aspects of news bulletins again have two distinct portions the studio segments incorporate standard cinematographic practices, while the actual news reports in many cases do not adhere to standard practices. Thus the overall degree of control that can be exercised on the filming aspects of news bulletins is lesser that of feature films.

Composition Control: News bulletins are composed video. The degree of control exercised on recorded reports is higher than that exercised on live reports or on transitions between live segments and recorded segments. As compared to feature films the degree of composition control tends to be smaller.

Channel Control: The information in news tends to be more in the audio channel; the visuals are used mainly as a enhancing mechanism for the audio report. The degree of control exercised on the distribution again tends to be smaller as compared to feature films.

Video Usage: The following is a list of news video users and the expected queries.

News Browser: This set of users is interested in news only from the perspective of getting news

News Producers & Reporters: These users reuse news for news report production. They are interested in researching facts related to a particular story. For example, the nomination of a new presidential candidate will typically result in a report with highlights in the person's life beginning from birth.

Chapter 2

Multimedia Authoring & Video Production

Multimedia authoring is the term used for the operation of composing digital documents which include different types of media objects. The typical types of media objects involved in *multimedia authoring* include text, graphics, sound, images and video. Such an authoring system will have a collection of video objects from which video clips will be included into different multimedia documents. The video collection will be updated with new video objects from time to time. During the authoring process, different clips of video will need to be retrieved from the video collection based on the content of the video. Thus a multimedia authoring system needs a video data management system to manage the video data that is associated with the authoring system.

Video Production is a term being used to cover the areas of film and video production. Specifically, editing is one of the operations that uses a large collection of video as a source for generating the final cut. In short, the video production process involves, shooting which generates a significant amount of raw footage and editing which organizes the raw footage into the final video. During the editing phase, video objects from the collection are retrieved based on content and these are organized into the final presentation. As the shooting operation progresses new clips are produced which are introduced into the video collection available to the editor. Thus the video production operation also requires a video data management system to manage the video data.

Both *multimedia authoring* and *video production* have a large degree of overlap in the access patterns they use for video collections. This is due to the fact that both the operations have the same goal, composing a presentation which includes video information. Hence both these applications are being treated as a single application for the purpose of identifying the requirements and for designing the data model.

Typical Examples

Two examples, one each from multimedia authoring and video production are considered. These examples are used to derive the typical queries in each of these applications. The examples have been chosen so that they cover the most typical access patterns that occur in each of the application. The examples are presented below:

Multimedia Authoring Example

Let the topic of the document being authored be The President's favorite sport: Golf. The document could include video clips of the white house, the president in golfing attire and the president playing golf. The following types of queries are possible:

1. Retrieve long shot of white house.
2. Retrieve shot of president on golf course.
3. Retrieve shot of president putting.
4. Retrieve tracking shot of putt.

Video Production Example

Consider a film segment which is picturing a car chase. Let the video have the following large scale structure:

- A panoramic view of the road and its surrounding area.
- The first car entering the scene.
- The second car in pursuit.
- Several shots of the chase in progress.
- The cars exiting the scene.

The following are queries which could arise while editing together such a segment:

1. Retrieve establishing shot of road scene.
2. Retrieve medium shot of car entering scene.
3. Retrieve zoom in shot of driver of car lead car.
4. Retrieve shots of cars passing the camera.

Design Of Data Model

The design of the video data model involves the following steps:

Step 1: Choice of Video Interval

It is used a video unit called shot. In film and video production terminology, the term shot refers to the sequence of images generated by the single operation of a movie camera. In all composed video productions the shot is the smallest identifiable visual unit which does not depend on the content of the video.

The video interval in the data model is chosen to be a shot. There are several implications of this decision:

- Since the shot is the finest visual granularity in a video, all other visual video units can be composed of shots. Thus using the shot as the basic temporal interval generates the finest visual index into video.
- Since the temporal interval choice is based solely on the visual property of video, this process segments the associated audio into undefined units. This is because of the reason that video and audio are not synchronized at the granularity of shots.

Step 2: Design of Temporal Relationships

Since the basic representation of video to be used is the shot, the relationships between various shots in a video and how they combine to form larger units is a question of general interest. This can be achieved by maintaining temporal relationships between the different units of video using a set of temporal relations. However, given the example query presented above, since none of the queries use the temporal relationship, the relationship need not be maintained in the data model. Thus no temporal relationships are maintained in the authoring and production support data model.

Step 3: Choice of Video Features

The video features in the video data model refer to the video interval specified in the data model. Each of the features describes a certain aspect of the video specified by the interval. In the query examples, the shot has been described in terms of some of its cinematographic properties and in terms of its content label. Thus the features to be used in the case of the authoring and production support data model are the cinematographic features of video and a set of labels which describe the content of video.

The Data Model

The previous three steps outlined the procedure for designing the video data model based on the model query. The data model for a database which supports the *multimedia authoring* and *video production* application is presented below. Let V_{ap} be the video model:

V_{ap}: *Video Interval*: Shot
: *Temporal Relations*: 0
: *Feature Count*: 2
: *Type*: (w_0, w_1)
: *Features*: (F_0, F_1)
: where F_0 = Cinematographic Label Set, F_1 = Content Labels

(Equation 2.0)

Video Processing Approaches, An Introduction

This paragraph makes an introduction to the video processing approaches which will be presented in detail to a fore come chapter. The goal of the approaches is to provide the tools necessary for reducing the effort involved in inserting video into a video database based on a video model. The tools presented here are specifically aimed at the multimedia authoring and production support database. However the ideas that are used in designing these video processing tools and the way in which the tools are applied can be used in the design of video data management systems for other applications. There are two video processing algorithms that have been presented:

Video Segmentation Algorithm: This algorithm processes a video stream and identifies in the stream the boundaries of video shots. The shot boundaries identified by the algorithm constitute the video interval in the video data model (equation 2.0). The use of such an algorithm in the video insertion process totally removes the effort necessary to manually identify video shot boundaries, the *segment location step*, in the video insertion process.

Video Indexing Techniques: The goal of the indexing algorithms is to simplify the video description process, the *segment description step* in the video insertion process. These algorithms process the individual video shots identified by the video segmentation algorithm. The shots are assigned a label based on the results of the video processing. The labels extracted by the video processing

algorithms can be used to partially automate the video description process.

Video Production

Video production is defined as the process that transforms a script into the final video. This paragraph presents a structured view of video production. The goal of the presentation is two fold: firstly, to present a clear picture of video production, with a view to identifying the various stages in the process. The second main objective is to identify the key labels or terms used to describe the videos generated by differing production parameter control strategies or styles. During the following discussion the term video represents movies, television programming, video recordings and any other form of moving image capture. Production techniques used in television production, news production and many other types of production have their origins in movie production and hence have a large degree of overlap with movie production.

The study presents a broad overview of movie production. The goal once again is to illustrate that there are clearly identifiable stages with codifiable rules in video production and that these rules can be systematically listed out and translated into constraints. The focus of this paragraph is on the editing and shot framing processes of video production.

Phases in video production

Video production is treated as a set of transformations that transform a given script to a video. Figure 2.0 shows the three main phases involved in video production. The flow indicated in the figure is a logical flow of the process; the phases have been identified based on the nature of the transformation that occurs. The first phase in the production process involves the creation of the content of the video, the second phase mainly deals with the capture of this content on to a media and the last phase involves the organization of this content into a coherent presentation. All the three phases occur simultaneously in many cases. Typically, the different shots are filmed out of order. Some of the dialogue is recorded during the shooting while others are recorded in the studios. The final organization of all the pieces is carried out in the editing and mixing rooms. For the purpose of video retrieval the temporal ordering of the phases is not very critical. Each of these phases has been discussed in further detail in the following sections. The processes that compose each phase have been represented in a diagram called the video production

diagram (figures 2.1, 2.2, 2.3). These diagrams use the following standardized notation:

Rectangle: Each rectangle in the figures represents an independent sub process. The process name is listed in bold face at the top of the box. The underlined labels within each rectangle represent the process parameters. And each process has inputs and outputs. In some rectangles composite processes are listed below the process name in smaller bold face letters.

Oval: The ovals in figures represent products or objects produced by the processes.

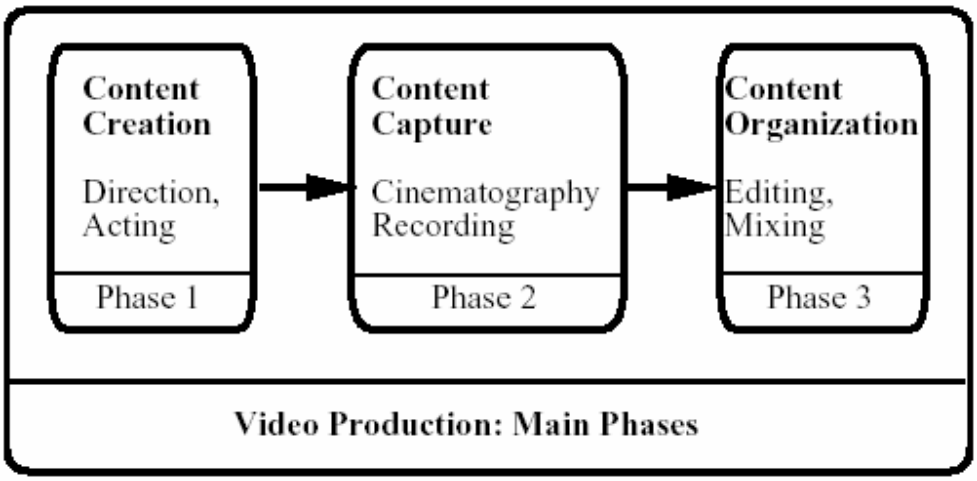


Figure 2.0: Phases in video production.

Phase 1: Content Creation: Direction

Content creation is defined as the process which transforms the script into an audio visual presentation. The process is referred to as direction in movie literature. Here the director along with the actors transforms the script into a form which can be captured onto film. This phase can be split into two sub-phases. The following discussion is intended only to bring out the organization of various aspects of content creation.

Visual Content Creation: Visual content arises from the combination of the script, actors and set. Based on the script and the directors' style, the actors' movements, expressions and gestures are controlled. At the same time visual content is very strongly influenced by the location or setting of the scene. The direction process has control over the choice of the location and on factors like lighting, background. In many cases, the visual content may be created in the studios through the use of special effects which place the actors into a non existent setting.

Aural Content Creation: Aural content is created by a combination of three distinct types of sound generation processes, dialogues, music and sound effects. The dialogue director uses the script and the actors to create dialogues. This process controls many of the parameters of dialogue delivery like pitch, intonation etc. Music is generated by the music director using the score and an orchestra. Here again several sound parameters of the music are manipulated to generate the desired effect. The effects process pertains mainly to generating different types of sound effects like background noises, special sound effects, etc. A variety of different procedures are used to generate these effects.

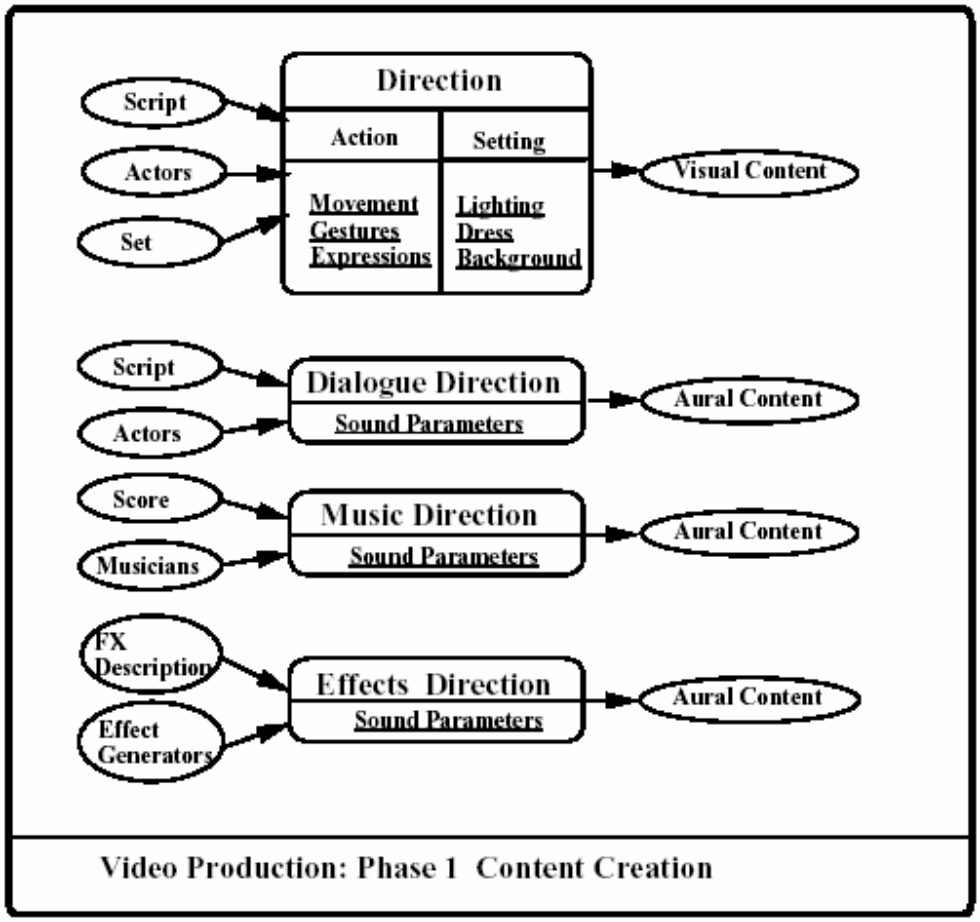


Figure 2.1: Video Production: Phase 1: Content Creation

Phase 2: Content Capture: Shooting & Recording

Content capture is referred to as cinematography in movie literature. Cinematography means writing with movement. In many cases cinematography includes the process of recording sound. Cinematography is an active process which captures the content with a unique perspective. There are several conventions and practices followed in cinematography. These conventions are applied to many different types of content. Thus from the perspective of deriving constraints usable in the database design process, cinematography is a very interesting process to study.

Visual Content Capture: Shooting: Visual content capture or shooting captures the visual content created by the director onto a medium like film or video tape or digital storage. The camera is the instrument which transforms the visible portion of the visual content into a form that can be stored onto a medium. Thus the process of shooting involves the control of various camera parameters like zoom, focus, aperture, etc. In addition to camera parameters the spatial relationship between the camera and the scene are also parameters which are constantly changed and varied during the shooting process. There are three main sets of parameters groups that are controlled during the shooting process *photographic*, *framing* and *shot time* as shown in figure 2.2. The time parameter in shooting mainly deals with the length of time for which the current visual content is captured onto film. This is referred to as *length of the take* in movie literature. The photography and framing parameters are discussed below:

Photographic Parameters: This group of parameters deals with the control of the camera optics. The goal of the control operation here is to generate a particular image quality. Image quality is normally measured in terms of the contrast of the image, quality of the colors etc. The main parameters in this group are:

- **Chromatic Parameters**: One of the important factors under this category is film speed i.e. the sensitivity of the film to light and to different frequency ranges of light. With the changes in technology, this parameter also refers to the frequency response of the CCD elements used in current video cameras. The other parameters that are grouped under this category include the lens aperture setting which controls the total amount of light entering the camera and the filters used on the lens to create various effects like haziness, fogginess etc.
- **Filming Speed Parameter**: This parameter controls the number of frames captured per second. The choice of this parameter depends mainly on the speed of the motion being filmed and the temporal resolution desired for capturing this motion.
- **Image Perspective Parameters**: This set of parameters specifies the field of view of the lens and the focal length. The typical types of lenses used include *wide angle*, *normal* and *telephoto*. A zoom lens is a lens which can be changed from a wide angle to a telephoto depending on the zoom setting.

Framing Parameters: The framing of shot deals with the relationship between the finite field of view of the camera system and the scene. This is considered a very important parameter in film production as the relationship between the frame (borders of the image) and the content of the image has a very significant effect on the way the viewer perceives the content of the image. A number of conventions are used in controlling this relationship. The following are the different parameters grouped under framing:

- **Frame Geometry**: This deal with the size and shape of the frame (image). This parameter was varied in the early days of movie making but has been standardized since. In the case of digital video applications the frame shape and size can be treated as a constant.
- **Space Definition**: This parameter deals with the relationship between the objects within the field of view and objects that are implicitly perceived by the viewer as being outside the field of view. This relationship cannot be directly used in automatic video processing in the context of VDMS as it is a *perceptual parameter*.
- **Vantage Point**: This is the set of parameters that define the relative geometry between the camera system and the scene. These set of parameters are very critical from the perspective of video analysis in VDMS. This includes parameters like relative camera angle and apparent distance.
- **Frame Motion**: This deal primarily with the way the framing changes over the duration of the shot. This depends on the camera motion and lens angle changes during a shot.

Aural Content Capture: Recording: The recording of sound is done on a number of separate tracks, each track being dedicated to a particular audio content. All these tracks are combined in stages to generate the final audio tracks which are synchronized with the video. There are three different types of audio tracks dialogue, music and effects. The dialogue tracks consist of a number of separate tracks some time the dialogue of each actor may be initially stored onto a separate track. Music is similarly recorded into a number of different tracks. Special sound effects are generated in the sound studios and stored onto different tracks as shown in figure 2.2.

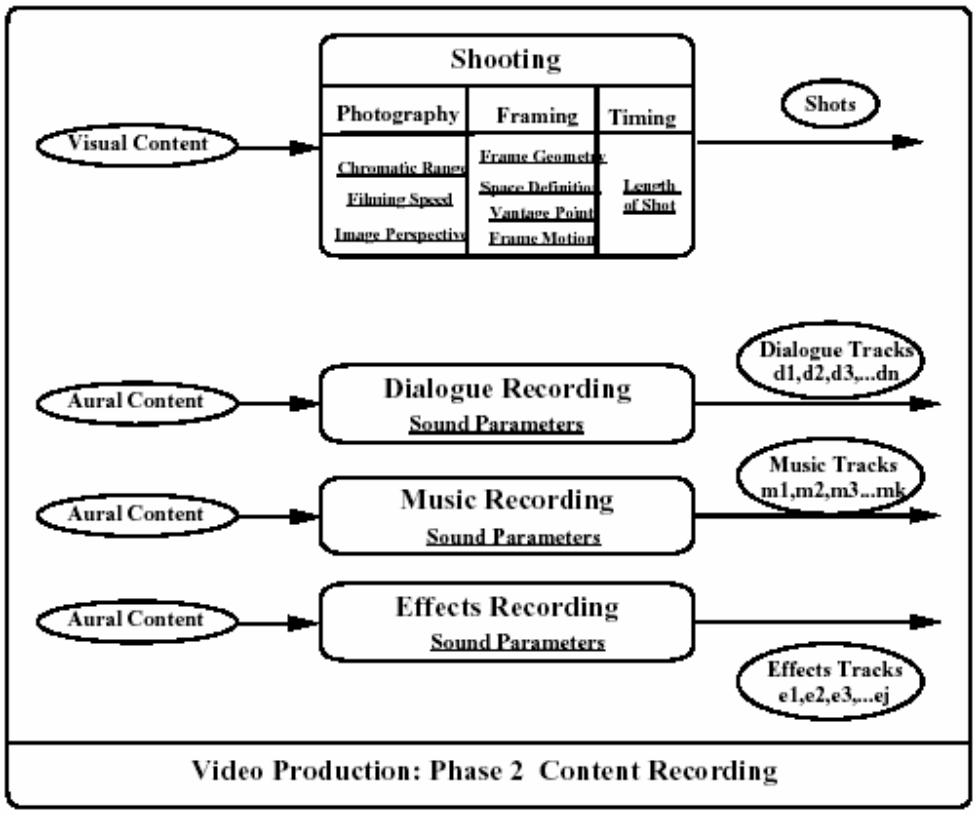


Figure 2.2: Video Production: Phase 2: Content Capture

Phase 3: Content Organization: Editing & Mixing

The final phase of video production is an organization phase, where all the content that has been captured is placed in some temporal ordering and the relative importance given to each of the different types of content is varied over time to generate the final video. This is referred to as the *final cut* in movie literature. The organization of visual content is done in the *editing process;* the organization of the audio tracks is carried out in the *pre-mixing process*. The final synchronization between the audio and visual tracks is done in the *mixing process*. This process also controls the relative importance of the audio as compared to the video, by controlling the various parameters of the audio tracks. Figure 2.3 shows the flow of the content organization phase of video production.

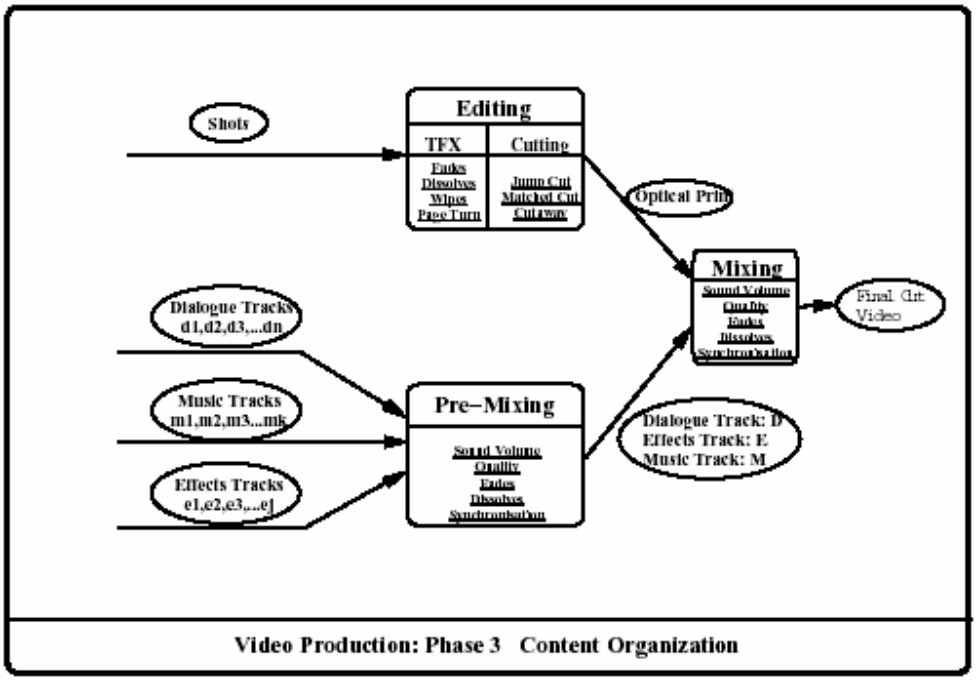


Figure 2.3: Video Production: Phase 3: Content Organization

The Shot Framing Process In Detail

Framing of the shot is one of the most important parameters to be controlled during the shooting process. The frame or the edges of the image define for the viewer the visible area of the scene. The relationship of objects to the frame significantly affects the perception of the content of the video. There is significant amount of literature which deals with the various conventions that are followed during the framing of a shot. The shot framing process is of interest from both the database perspective and from the data analysis point of view. From the database perspective being able to index video based on shot framing parameters will allow the database to support queries based on different types of shots. The conventions used in shot framing can be used to provide constraints for developing video analysis algorithms. The goal of the study is to identify different types shot labels used in movie literature based on different parameters of the shot. These labels are later used in the development video models which define the ideal index into video. Shot classes are identified based on the *vantage point* parameter group of shot framing.

Apparent Distance based labels: Apparent distance is the distance perceived between the subject and the camera. It is a function of the actual distance between the camera and the subject, the type and setting of the lens used in the shot (zoom setting) and the size of the object. The apparent distance affects the framing of the subject within the picture. The classification of shots based on this parameter uses the relative area occupied by the subject within the frame as a basis for grouping of shots. Figure 2.4 shows the relationships between the frame size and subject size assigned. The following are the apparent distance labels used:

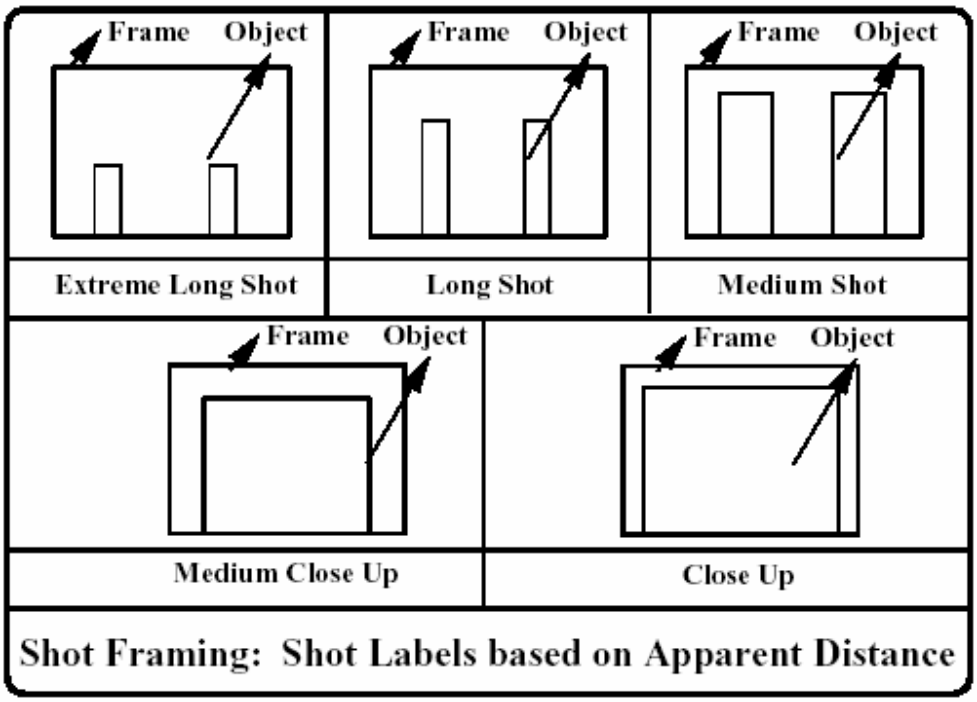


Figure 2.4: Apparent Distance based shot labels

- **Extreme Long Shot (XLS)**: A shot in which the subject is very far from the camera. Hence the subject occupies a very small portion of the frame and the shot gives an overall picture of the location.
- **Long Shot (LS)**: The apparent distance between the subject and the camera is large. The subject in general occupies about half the eight of the frame. Some area of the scene both above and below the subject is visible.

- **Medium Long Shot (MLS)**: The apparent distance between the camera and subject is lesser that a long shot but more than a medium shot. The subject completely occupies the height of the frame. The frame includes some part of the background scene in addition to the subject.
- **Medium Shot (MS)**: A shot which captures about half the length of the subject within the frame. The frame has lesser part of the background than the subject.
- **Medium Close Up Shot (MCU)**: The apparent distance between the subject and the camera lies in between that of a close up and a medium shot. The subject occupies a significant portion of the frame but not the entire frame.
- **Close Up Shot (CU)**: The apparent distance between the subject and the camera is small. The subject occupies almost the entire frame.
- **Extreme Close Up Shot (XCU)**: A shot which shows only a small portion of the subject. Hence the subject occupies the entire frame. Example: A portion of the human face like just the eyes.

Relative Camera Angle based labels: Labels assigned here depend on the angle from which the camera views the subject. They are a function of the height of the camera, the height of the subject and the orientation of the camera with reference to the ground plane. Figure 2.5 shows the three basic labels that are used. The labels from literature are listed below:

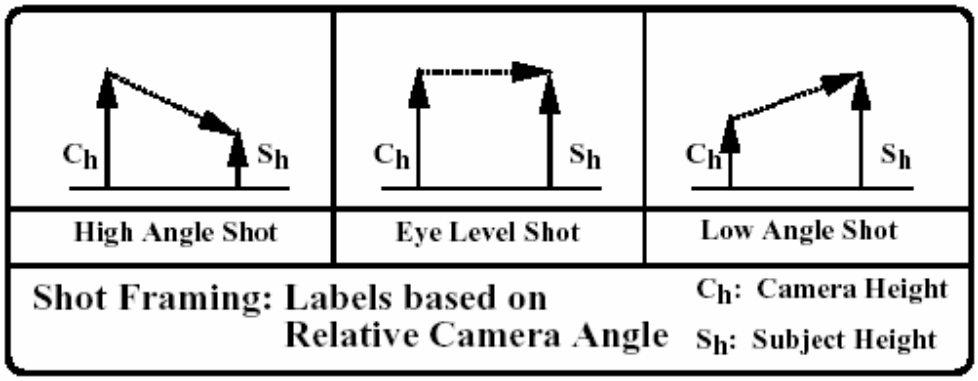


Figure 2.5: Relative Camera angle based shot labels

- **Eye Level Shot (ELS)**: The height of the camera and the height of the subject are approximately the same. The tilt of the camera with reference to the ground is typically zero.
- **Low Angle Shot (LAS)**: The height of the camera is lesser than that of the subject and the camera is tilted upwards towards the subject.
- **Extreme Low Angle Shot (XLAS)**: A shot where the camera height is very small compared to the subject and the camera is tilted up towards the top of the subject.
- **High Angle Shot (HAS)**: The height of the camera is larger than the height of the subject. The camera is tilted downwards with reference to the ground plane.
- **Extreme High Angle Shot (XHAS)**: A shot where the camera height is very large compared to the height of the subject and the camera is tilted down towards the subject. Such shots are generally acquired from helicopters, tower, etc.
-

The Editing Process In Detail

Editing is the process in which the film is composed from its component shots. This is the most important process in the content organization phase. This process involves the selection of shots from a set of shots and shaping or trimming the shots to the required length. The shots are then composed into scenes, sequences etc to match the script of the video.

Editing is the process which allows a video to present time and space at a scale different from reality. This process relies on the viewers perception and understanding to compress time and space. The editing process provides the editor with the following degrees of freedom between the two shots being edited.

- **Graphic Relations**: This is the visual relationship between the two shots being edited. This is the most important property from the perspective of segmenting video based on visual shot transitions. The editing process can minimize the visual discontinuity to the viewer across an edit; such an edit is called a graphic match. In some cases the director may maximize the visual discontinuity between shots.
- **Rhythmic Relations**: This is the relationship between the lengths of the shots being edited.
- **Spatial Relations**: This is a perceptual property where disjointed space is composed using editing.
- **Temporal Relations**: This is a perceptual property where the perceived time between two shots is controlled.

From the perspective of image based video segmentation the complete process of editing can be split into the following two phases illustrated in figures 2.6, 2.7.

- **Edit Decision**: This is the process of deciding the temporal ordering of shots. It also involves deciding on the transitions or edits to be used between different shots. The result of the editing process is a list called the Edit Decision List.
- **Assembly**: This is the physical process which the Edit Decision List is converted into frames on the final cut. The process involves taking the shots from the shot set in the specified order, and implementing the edits between the shots. The assembly process in general adds frames called edit frames to the final cut in addition to the frames from the original shots.

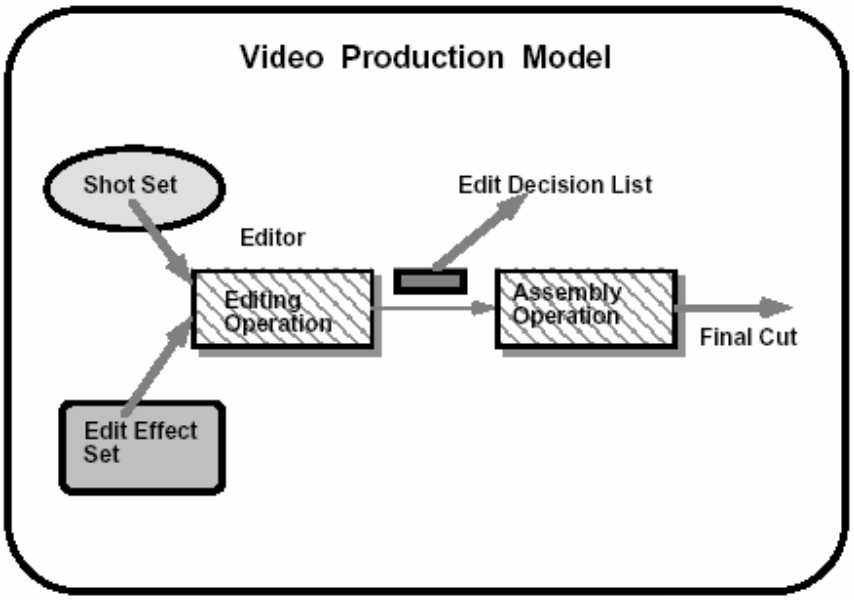


Figure 2.6: Editing Operation

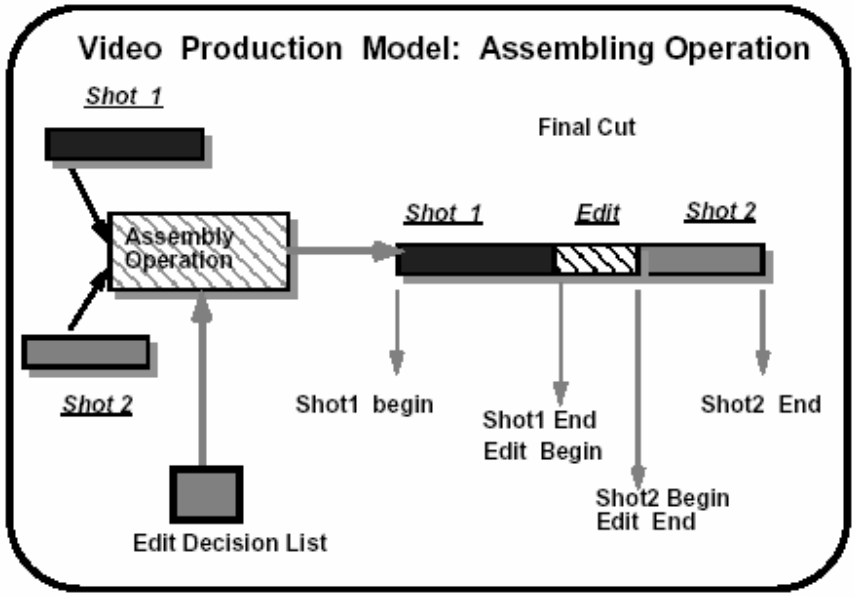


Figure 2.7: Editing Operation: Assembly Operation

Chapter 3

VIDEO PROCESSING TECHNIQUES: A Review of Uncompressed Video Segmentation Techniques

One of the components of a video database system is a set of algorithms or techniques which handle and process video. There has been an increase in the research in the area of digital video processing. One of the main processing needed when dealing with multimedia data is multimedia sequence indexing. The importance of this research field can be shown by the number of recent communications and publications on the subject. In order to index multimedia data, we may need a preprocessing the aim of which is to temporally segment the videos, that is to say detect the shot changes present in the video sequences.

The number of shot change detection methods is now important and several reviews of these methods have been done. These reviews often present the different methods and their efficiency based on some quality measures. So they are very useful when one wants to select and implement a shot change detection method for a global video processing which could be done off-line. When processing has to be done on-line, the selection of a particular method should also consider computation time. This is especially true when dealing with uncompressed video sequences which contain a huge quantity of data. If the method has to be implemented on common hardware architecture, computation time is directly linked with complexity of the method.

Before it is presented a large number of methods, it is necessary to recall and describe the different forms a shot change can take. It is also defined the notations used in this chapter. The classification of the presented methods it is based on the basic elements used in the segmentation process: *pixels*, *histograms*, *blocks*, *features*, *motion* and combination of several approaches.

Shot Change Description

A shot is defined as a continuous video acquisition (with the same camera). When the video acquisition is done with another camera, there is a shot change. The simplest way to perform a change between two shots is called a *cut*. In this case, the last frame of the first video sequence is directly followed by the first frame of the second video sequence. This kind of shot change is also called abrupt change. Because of their simplicity, cuts are often the easiest shot changes to be detected.

More complex shot changes are now available for video editing, thanks to improvement of the video production software. Instead of cutting and pasting the second video next to the first one, it is possible to insert an effect, as a *wipe*, a *fade*, or a *dissolve*. A wipe is obtained by progressively replacing the old image by the new one, using a spatial basis. A dissolve is a transition where all the images inserted between the two video sequences contain pixels whose values are computed as linear combination of the final frame of the first video sequence and the initial frame of the second video sequence. Fades are special cases of dissolve effects, where a monochrome frame replaces the last frame of the first shot (*fade in*) or the first frame of the second shot (*fade out*). There are also other kinds of effects (combining for example wipe and zoom), but actually most of the shot change detection methods are concerned only by the effects described previously in their indexing task.

Notations

Video sequences are composed of successive frames or images. We define I_t the frame of the video obtained at time t. So it is possible to define $P(I_t, i, j)$ the intensity of the pixel with coordinates i and j in the frame I_t. We assume that the size of the images is X-by-Y pixels, so we have $1 \le i \le X$ and $1 \le i \le Y$.

When methods are dealing with color images, the notation $P(I_t, C_k, i, j)$ will be used. C_k represents the color component numbered k. As an example, we can consider that C_1, C_2, and C_3 are respectively representing the R, G, and B components in the RGB color space. So $P(I_t, C_k, i, j)$ is representing the value of the color component C_k for the pixel with coordinates i and j in frame I_t.

Some methods are dealing with histograms. So we define $H(I_t, v)$ the number of pixels of the image It with an intensity equal to v, with $v \in [0, V]$ where V is the maximum gray-level value. If we consider color images, indexing methods can use several histograms, one for each color component. We then use the notation $H(I_t, C_k, v)$ to define the number of pixels with an intensity value of v for the color component C_k in the image I_t.

Another common approach for video segmentation is to use block-sampled images. Let us note B the number of blocks b in each frame.

Finally, because a lot of methods are using some thresholds for shot change detection, we have also noted T some threshold fixed by the user. Several authors propose a learning procedure in order to use an appropriate threshold value.

As can easily be imagined from this introduction part, the works dealing with video sequence segmentation are quite numerous. Even if the complexity of the methods is naturally increasing along time we have not chosen a chronological thread to present the various methods. Rather we have sorted them according to the basic elements they are relying on. We have organized them from the most simple; the pixel in the image to the most sophisticated ones, those that are using a combination of methods. More precisely we have distinguished 6 large categories characterized by the respective use of:

- Pixel characterization.
- Histogram of the frames.
- Partition of the image in blocks.
- Features.
- Motion during the sequence.
- Combination of approaches.

Pixel-based Methods

Shot change detection can be performed by comparing successive frames. The simplest way to compute the dissimilarity between two frames is to compare corresponding pixels from two successive images. As we will see, some improvements of the initial pixel comparison have been proposed. First we present the methods considering two consecutive frames and then those that extend the study to a longer temporal interval.

Pixel comparison between two successive frames

One of the first methods described in literature was from Nagasaka et al in 1991. Shot changes are detected using a simple global inter-frame difference measure, defined as:

$$\text{Detection if } \left(\left| \sum_{i=1}^{X} \sum_{j=1}^{Y} P(I_t, i, j) - \sum_{i=1}^{X} \sum_{j=1}^{Y} P(I_{t-1}, i, j) \right| \right) > T \quad \text{(Equation 3.0)}$$

Nagasaka et al also introduced a shot change detection method based on pixel pair difference called template matching. For every two successive frames, differences of intensities are computed on pixels having the same spatial position in the two frames. Then the cumulated sum of differences is compared to a fixed threshold in order to determine if a shot change has been detected:

$$\text{Detection if } \left(\sum_{i=1}^{X} \sum_{j=1}^{Y} \left| P(I_t, i, j) - P(I_{t-1}, i, j) \right| \right) > T \quad \text{(Equation 3.1)}$$

A color version has also been presented:

$$\text{Detection if } \left(\sum_{i=1}^{X} \sum_{j=1}^{Y} \sum_{k=1}^{3} \left| P(I_t, C_k, i, j) - P(I_{t-1}, C_k, i, j) \right| \right) > T \quad \text{(Equation 3.2)}$$

A couple of years later, Zhang et al were comparing the pixels of two successive frames on a Boolean basis. The fact that pixels are different is noted:

$$D(I_t, I_{t-1}, i, j) = \begin{cases} 1, & \text{if } P(I_t, i, j) \neq P(I_{t-1}, i, j) \\ 0, & \text{Otherwise} \end{cases} \quad \text{(Equation 3.3)}$$

for the gray-level case and requires one operation per couple of pixels. Definition is quite similar for color images. In order to allow some variations on pixel intensities, a better (but more complex as it needs three operations instead of one) definition is given:

$$D(I_t, I_{t-1}, i, j) = \begin{cases} 1, & \text{if } \left| P(I_t, i, j) - P(I_{t-1}, i, j) \right| > T_D \\ 0, & \text{Otherwise} \end{cases} \quad \text{(Equation 3.4)}$$

where T_D is considered as the tolerance value. The amount of different pixels is computed and is compared to a given threshold, which results in the detection or not of a shot change:

$$\text{Detection if } \left(\sum_{i=1}^{X} \sum_{j=1}^{Y} D(I_t, I_{t-1}, i, j) \right) > T \quad \text{(Equation 3.5)}$$

In order to avoid false detections due to motion in the video sequence, they also propose to smooth the images with a filter of size 3 × 3 before computing the D values. The filter limits the effects due to noise and camera motion.

Several other statistical measures have been proposed in the literature. The normalized difference energy and the normalized sum of absolute differences can be used for shot detection, as shown by the following equations:

$$\text{Detection if } \left(\frac{\sum_{i=1}^{X}\sum_{j=1}^{Y}\left(P(I_t,i,j)-P(I_{t-1},i,j)\right)^2}{\left(\sum_{i=1}^{X}\sum_{j=1}^{Y}P(I_t,i,j)^2\right)\left(\sum_{i=1}^{X}\sum_{j=1}^{Y}P(I_{t-1},i,j)^2\right)} \right) > T \quad \text{(Equation 3.6)}$$

$$\text{Detection if } \left(\frac{\sum_{i=1}^{X}\sum_{j=1}^{Y}\left|P(I_t,i,j)-P(I_{t-1},i,j)\right|}{\sum_{i=1}^{X}\sum_{j=1}^{Y}P(I_t,i,j)+\sum_{i=1}^{X}\sum_{j=1}^{Y}P(I_{t-1},i,j)} \right) > T \quad \text{(Equation 3.7)}$$

Pixel intensity time variation

The previous two frame study can be generalized by analyzing variations of intensities through time. Taniguchi et al label pixels with respect to the evolution of their intensities on several successive frames. The labels used are "constant", "step(I_t)", "linear(I_{t_1}, I_{t_2})", and "no label". These labels represent respectively pixels with constant values, pixels with a change in value at frame It, pixels with a progressive change in value between frames I_{t_1} and I_{t_2}, and finally pixels with random values due to motion. Two Boolean conditions $\Theta_1(I_{t_1},I_{t_2},i,j)$ and $\Theta_2(I_{t_1},I_{t_2},i,j)$ are introduced in order to define the constancy of a set of pixel values $P(I_t,i,j)$ with $t_1 \le t \le t_2$:

$$\Theta_1(I_{t_1},I_{t_2},i,j)=\begin{cases}\text{true if } \left(\max_{t_1\le t\le t_2}P(I_t,i,j)-\min_{t_1\le t\le t_2}P(I_t,i,j)\right)<T \\ \text{false} \qquad\qquad \text{Otherwise}\end{cases} \quad \text{(Equation 3.8)}$$

$$\Theta_2(I_{t_1},I_{t_2},i,j)=\begin{cases}\text{true if } \left(\max_{t_1\le t\le t_2}\left(P(I_t,i,j)+(t-t_1)J_{t_1,t_2}\right)-\min_{t_1\le t\le t_2}\left(P(I_t,i,j)+(t-t_1)J_{t_1,t_2}\right)\right)<T \\ \text{false} \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{Otherwise}\end{cases}$$

(Equation 3.9)

with J_{t_1,t_2} defined as:

$$q_{t_1,t_2}=\frac{\left|P(I_{t_1},i,j)-P(I_{t_2},i,j)\right|}{t_2-t_1} \quad \text{(Equation 3.10)}$$

These similarity conditions Θ_1 and Θ_2 are then used to determine the label $L(I_{t_o}, I_{t_f}, i, j)$ of each pixel of a video sequence involving $f+1$ frames, using the following scheme:

$$L(I_{t_o}, I_{t_f}, i, j) = \begin{cases} \text{constant} & \text{if } \Theta_1(I_{t_o}, I_{t_f}, i, j) \\ \text{step}(I_t) & \text{if } \left(\Theta_1(I_{t_o}, I_{t-1}, i, j) \wedge \Theta_1(I_t, I_{t_f}, i, j) \wedge \neg\Theta_1(I_{t-1}, I_t, i, j)\right) \\ \text{linear}(I_{t_1}, I_{t_2}) & \text{if } \left(\Theta_1(I_{t_0}, I_{t_1}, i, j) \wedge \Theta_1(I_{t_2}, I_{t_f}, i, j) \wedge \neg\Theta_1(I_{t_1}, I_{t_2}, i, j) \wedge \Theta_2(I_{t_1}, I_{t_2}, i, j)\right) \\ \text{no label} & \text{Otherwise} \end{cases}$$

(Equation 3.11)

which can also be defined as:

$$L_{label}(I_{t_0}, I_{t_f}, i, j) = \begin{cases} 1 \text{ if } L(I_{t_0}, I_{t_f}, i, j) \text{is of kind "label"} \\ 0 \quad \text{Otherwise} \end{cases} \quad \text{(Equation 3.12)}$$

Quantities of pixels associated with each label are computed. Cuts (respectively dissolves) are detected thanks to the analysis of the ratio between quantity of pixels labeled "step" (respectively "linear") and quantity of pixels labeled (i.e. with a label different of "no label"). A cut is detected at frame I_t if:

$$\text{Detection if } \left(\frac{\sum_{i=1}^{X}\sum_{j=1}^{Y} L_{step(I_t)}(I_{t_0}, I_{t_f}, i, j)}{XY - \sum_{i=1}^{X}\sum_{j=1}^{Y} L_{no\ label}(I_{t_0}, I_{t_f}, i, j)} \right) > T \quad \text{(Equation 3.13)}$$

A dissolve is detected between frames I_{t_1} and I_{t_2} if:

$$\text{Detection if } \left(\frac{\sum_{i=1}^{X}\sum_{j=1}^{Y} L_{linear(I_{t_1}, I_{t_2})}(I_{t_0}, I_{t_f}, i, j)}{XY - \sum_{i=1}^{X}\sum_{j=1}^{Y} L_{no\ label}(I_{t_0}, I_{t_f}, i, j)} \right) > T \quad \text{(Equation 3.14)}$$

Lawrence et al use evolution of temporal derivative of the pixel intensities as a criterion for shot change detection. First pixels with high spatial derivative are discarded in order to avoid motion effect. A pixel $P(I_t, i, j)$ is considered if and only if the following condition holds:

$$\max\left(\left|P(I_t,i,j)-P(I_t,i-1,j)\right|,\left|P(I_t,i,j)-P(I_t,i,j-1)\right|\right)<T \quad \text{(Equation 3.15)}$$

A convolution process involving remaining pixels and a Gaussian mask is then performed to obtain temporal derivative of $P(I_t,i,j)$. Absolute values of these derivatives are summed up in order to define the distance measure which will be analyzed through time. Shot boundaries correspond to local maxima of this distance measure. False detections due to noise or motion are limited if the neighborhood of the local maxima obtained previously are further analyzed.

Histogram-based Methods

The previous section was dedicated to pixel-based methods. It is also possible to compare two images based on global features instead of local features (pixels). Histogram is a global image feature widely used in image processing. The main advantage of histogram-based methods is their global aspect. So they are more robust to camera or object motion. The main drawback appears when we compare two different images having a similar histogram. It will often results in missing a shot change.

Different uses of the histogram can be distinguished. Some methods only compute differences between histograms and then the quality of the result is linked to the kind of histogram considered. A first extension is the use of weighted differences between histograms. Another approach consists in the definition of an intersection operator between histograms or the definition of different distances or similarity measures.

Histogram Difference

Tonomura et al proposed a method based on gray-level histograms. Images are compared by computing a distance between their histograms, as shown in the following equation:

$$\text{Detection if } \left(\sum_{v=0}^{V}\left|H(I_t,v)-H(I_{t-1},v)\right|\right)>T \quad \text{(Equation 3.16)}$$

Nagasaka et al propose a similar method using only 64 bins for color histograms (2 bits for each color component of the RGB space). Using the notation $H_{64}(I_t,v)$, the detection is defined by:

$$\text{Detection if } \left(\sum_{v=0}^{63}\left|H_{63}(I_t,v)-H_{63}(I_{t-1},v)\right|\right)>T \quad \text{(Equation 3.17)}$$

Gargi et al apply histogram difference to other color spaces (HSV, YIQ, L*a*b*, L*u*v*, and Munsell). More precisely, only non-intensity components are used (i.e. Hue and Saturation for HSV, I and Q for YIQ, a* and b* for L*a*b*, u* and v* for L*u*v*, and hue and chroma for the Munsell space). Shot change detection is then defined by:

$$\text{Detection if } \left(\sum_{k=1}^{2} \sum_{v=0}^{V} \left| H(I_t, C_k, v) - H(I_{t-1}, C_k, v) \right| \right) > T \quad \text{(Equation 3.18)}$$

Pye et al compute three histogram differences, considering separately the three color components of the RGB space. The highest value is compared to a threshold for shot change detection:

$$\text{Detection if } \left(\max_{k \in \{R,G,B\}} \sum_{v=0}^{V} \left| H(I_t, C_k, v) - H(I_{t-1}, C_k, v) \right| \right) > T \quad \text{(Equation 3.19)}$$

Ahmed et al present several shot change detection algorithms using color histograms. The first algorithm compares two frames using histograms computed on the Hue component C_H. So can be represented by:

$$\text{Detection if } \left(\frac{\sum_{v=0}^{V} \left| H(I_t, C_H, v) - H(I_{t-\Delta}, C_H, v) \right|}{\sum_{v=0}^{V} H(I_{t-\Delta}, C_H, v)} \right) > T \quad \text{(Equation 3.20)}$$

where Δ is the temporal skip between two frames.

The second algorithm by Ahmed et al is based on reduced RGB space histograms. As in Nagasaka et al, histograms are composed of only 64 bins, using 2 bits for each color component. The detection is done through a computation similar to the previously mentioned method:

$$\text{Detection if } \left(\frac{\sum_{v=0}^{63} \left| H_{64}(I_t, v) - H_{64}(I_{t-\Delta}, v) \right|}{\sum_{v=0}^{63} H_{64}(I_{t-\Delta}, v)} \right) > T \quad \text{(Equation 3.21)}$$

O'Toole et al detect shot changes using a cosine similarity measure computed between two histograms. First three 64 bin histograms representing respectively the Y, U, and V components are obtained from each frame. Next the three histograms are concatenated into a single one in order to get only one 192 bin histogram per frame. Then two successive frames are compared based on their histogram using a cosine similarity measure to perform shot change detection:

$$\text{Detection if } \left(1 - \frac{\sum_{v=0}^{V} \left(H_{YUV}(I_t, v) H_{YUV}(I_{t-1}, v) \right)}{\sum_{v=0}^{V} H_{YUV}(I_t, v)^2 \sum_{v=0}^{V} H_{YUV}(I_{t-1}, v)^2} \right) > T \quad \text{(Equation 3.22)}$$

Chiu et al rely their video segmentation on a genetic algorithm using color histogram differences. Possible shot boundaries are evaluated with similarity adjacency functions. In order to limit the optimization cost of these functions, a genetic algorithm is used instead of traditional methods. A video sequence is encoded as a string of binary values, 1 and 0 representing respectively the presence or not of a shot boundary in the current frame. The fitness function used in the algorithm is defined as a similarity adjacency function based on color histogram differences. Finally crossover and mutation processes are derived from classical genetic algorithms in order to be adapted to video segmentation task.

Zhang et al propose a method called twin comparison. Successive frames are compared using a histogram difference metric. The difference values obtained are compared with two thresholds. Cuts are detected when difference is higher than a high threshold T_H. Possible starts of gradual transition are detected when difference is higher than a low threshold T_L. In this case, an accumulated difference is computed until the current difference is below T_L. Finally the accumulated difference is compared to the high threshold T_H for shot change detection. The two thresholds can be automatically set using standard deviation and mean of the inter-frame differences in the whole video sequence.

Li et al use also a two step method, detecting successively the location of the end of the transition and its start. Frames are compared using the color ratio histogram metric. First two frames I_{t_1} and I_{t_2} (with $t_2 = t_1 + \Delta$) are compared using this metric. While the difference is below a given threshold T, t_2 is set to $t_2 + 1$. When the difference is above T, the transition end has been obtained. In order to determine the transition start, t_1 is set to $t_2 - 1$. The difference between frames I_{t_1} and I_{t_2} is then computed and compared to the threshold T. While the difference is below T, t_1 is set to $t_1 - 1$. When the difference is above T, the transition start has also been obtained.

Several other statistical measures have been reviewed in . The quadratic histogram difference can be computed between histograms from two successive frames, whereas the Kolmogorov-Smirnov statistic is computed between cumulative histograms from two successive frames. These two measures are detailed below, using the notation $H_C(I_t, v)$ to represent the cumulative histogram up to bin v for the frame I_t.

$$\text{Detection if } \left(\sum_{v=0}^{V} \frac{(H(I_t, v) - H(I_{t-1}, v))^2}{(H(I_t, v) + H(I_{t-1}, v))^2} \right) > T \quad \text{(Equation 3.23)}$$

$$\text{Detection if } \left(\max_{v \in [0,V]} \left(\left| H_C(I_t, v) - H_C(I_{t-1}, v) \right| \right) \right) > T \quad \text{(Equation 3.24)}$$

Weighted Difference

In color images, some color components may have a bigger influence than others. So it is possible to detect shot changes by weighting the histograms of each color component depending on their importance:

$$\text{Detection if } \left(\sum_{k=1}^{3} \sum_{v=0}^{V} \frac{L(I_t, C_k)}{L_{mean}(I_t)} \left| H(I_t, C_k, v) - H(I_{t-1}, C_k, v) \right| \right) > T \quad \text{(Equation 3.25)}$$

where $L(I_t, C_k)$ and $L_{mean}(I_t)$ are respectively the luminance for the k^{th} color component of the frame I_t and the average luminance of the frame I_t considering all the color components.

Zhao et al use a learning procedure to determine the best weight values for weighted histogram difference computation. They first compute the original histogram difference defined by equation 3.16. Then a learning step formulated as a min-max optimization problem is performed

in order to select the best weights to use in weighted histogram differences. The detection process relies finally on the following equation:

$$\text{Detection if } \left(\sum_{k=1}^{3} \sum_{v=0}^{V} w(k,v) \left| H(I_t, C_k, v) - H(I_{t-1}, C_k, v) \right| \right) > T \quad \text{(Equation 3.26)}$$

where $w(k,v)$ represents the best weight selected after the learning step.

Gargi et al presented a method based on the difference of average colors of a histogram, which can be as well considered as a histogram weighted difference. The shot change detection can then be represented by:

$$\text{Detection if } \left(\sum_{k=1}^{3} \left(\sum_{v=0}^{V} H(I_t, C_k, v)v - \sum_{v=0}^{V} H(I_{t-1}, C_k, v)v \right)^2 \right) > T \quad \text{(Equation 3.27)}$$

Another method by Gargi et al using color histograms has also been proposed. More precisely, it uses a reference color table as a frame difference measure. Reference color table can be seen as a coarse quantization of RGB color space into 27 different color triples which are used as bins for a 3-D color histogram H_{ref}. The shot change detection can be represented by:

$$\text{Detection if } \left(\sum_{v=0}^{V} w(v,t) \sqrt{\left(H_{ref}(I_t, v) - H_{ref}(I_{t-1}, v) \right)^2} \right) > T \quad \text{(Equation 3.28)}$$

where the weight $w(v,t)$ is defined as:

$$w(v,t) = \begin{cases} H_{ref}(I_{t-1}, v) \text{ if } \left(H_{ref}(I_t, v) > 0 \right) \wedge \left(H_{ref}(I_{t-1}, v) > 0 \right) \\ 1 \qquad \text{Otherwise} \end{cases} \quad \text{(Equation 3.29)}$$

Histogram Intersection

Similarity between two images can also be evaluated thanks to histogram intersection. Histogram intersection is computed using different operators, for example a min function. Similarity ratio belonging to interval [0, 1] is then compared to a given threshold. This comparison allows the detection of shot changes:

$$\text{Detection if } \left(1 - \frac{1}{XY} \sum_{v=0}^{V} \min\left(H(I_t, v), H(I_{t-1}, v) \right) \right) > T \quad \text{(Equation 3.30)}$$

where XY represents the number of pixels in frames processed.
Another version of the histogram intersection-based shot change detection method is defined by the following equation:

$$\text{Detection if } \left(1-\frac{1}{XY}\sum_{v=0}^{V}\frac{\min(H(I_t,v),H(I_{t-1},v))}{\max(H(I_t,v),H(I_{t-1},v))}\right)>T \quad \text{(Equation 3.31)}$$

Haering et al apply histogram intersection defined in equation 3.30 to HSV (Hue, Saturation, Value) color space, using 16 bins for Hue component and 4 bins each for Saturation and Value components.

An extension of the above method has been proposed by Javed et al. Hue is represented using only 8 bins. Instead of thresholding the histogram intersection of two successive frames, they compute the difference between two successive histogram intersection values and compare this derivative to a threshold.

Use of c^2 Test

Nagasaka et al have also proposed a 64 bin histogram comparison based on 2 test. The shot change detection is then defined by:

$$\text{Detection if } \left(\sum_{v=0}^{63}\frac{(H_{64}(I_t,v)-H_{64}(I_{t-1},v))^2}{H_{64}(I_t,v)}\right)>T \quad \text{(Equation 3.32)}$$

with the assumption $H_{64}(I_t,v)\neq 0$. If this assumption does not hold, we use the following equation instead:

$$\text{Detection if } \left(\sum_{v=0}^{63}\frac{(H_{64}(I_t,v)-H_{64}(I_{t-1},v))^2}{H_{64}(I_{t-1},v)}\right)>T \quad \text{(Equation 3.33)}$$

with the assumptions $H_{64}(I_{t-1},v)\neq 0$ and $H_{64}(I_t,v)=0$. This method is considered as more efficient than simple histogram comparison-based methods.

A modification has been proposed by Dailianas et al where the detection is represented by:

$$\text{Detection if } \left(\sum_{v=0}^{V}\frac{(H(I_t,v)-H(I_{t-1},v))^2}{\max(H(I_t,v),H(I_{t-1},v))}\right)>T \quad \text{(Equation 3.34)}$$

Gunsel et al perform a K-means clustering algorithm to determine the location of shot boundaries. Successive frames are compared using 2 test or histogram difference on YUV histograms. Every inter-frame difference value is classified into shot change or non-shot change.

Similarity measures between normalized Histograms

Several measures computed on normalized histograms have been reviewed by Ren et al and by Kim and Park . Using the notation $H_N(I_t, v)$ to represent the probability of intensity v in the frame I_t, cross entropy, divergence, Kullback Liebler distance, and Bhattacharya distance are respectively defined as:

$$\text{Detection if } \left(\sum_{v=0}^{V} \left(H_N(I_t, v) \log \frac{H_N(I_t, v)}{H_N(I_{t-1}, v)} \right) \right) > T \quad \text{(Equation 3.35)}$$

$$\text{Detection if } \left(\sum_{v=0}^{V} \left(H_N(I_t, v) \log \frac{H_N(I_t, v)}{H_N(I_{t-1}, v)} \right) + \sum_{v=0}^{V} \left(H_N(I_{t-1}, v) \log \frac{H_N(I_{t-1}, v)}{H_N(I_t, v)} \right) \right) > T$$

(Equation 3.36)

Detection if

$$\left(\sum_{v=0}^{V} \left(H_N(I_t, v) \log \frac{H_N(I_t, v)}{H_N(I_{t-1}, v)} \right) + \sum_{v=0}^{V} \left((1 - H_N(I_t, v)) \log \frac{1 - H_N(I_t, v)}{1 - H_N(I_{t-1}, v)} \right) \right) > T$$

(Equation 3.37)

$$\text{Detection if } \left(-\log \left(\sum_{v=0}^{V} \sqrt{H_N(I_t, v) H_N(I_{t-1}, v)} \right) \right) > T \quad \text{(Equation 3.38)}$$

All these methods are based on a uniform process all over the image. The heterogeneity present within a frame led to use block-based methods.

Blocked-based Methods

Block sampling of the video frames can be performed in order to increase the quality of shot change detection but also to decrease the computation time. Once block representation has been obtained from original images, it is possible to perform some algorithms derived from pixel or histogram-based methods presented previously. Use of blocks allows a processing which is intermediate, between local level like pixel-based methods and global level as histogram-based methods. Main advantage of block-based methods is their relative insensitivity to noise and camera or object motion. We have distinguished between several approaches all working on blocks.

Block Similarity

Kasturi et al perform a similarity test on block-sampled images. Like in pixel-based methods, pairs of blocks (with same spatial coordinates) from two successive frames are compared. The similarity is based on block features like mean and variance. The likelihood rate L is defined for a block b as:

$$L(I_t, I_{t-1}, b) = \frac{\left(\frac{s^2_{t,b} + s^2_{t-1,b}}{2} + \left(\frac{m_{t,b} - m_{t-1,b}}{2} \right)^2 \right)^2}{s^2_{t,b} s^2_{t-1,b}} \quad \text{(Equation 3.39)}$$

where $m_{t,b}$ and $s^2_{t,b}$ are respectively the mean and the variance of block b pixel values in image I_t. Then thresholded values L_D of L are defined by the equation:

$$L_D(I_t, I_{t-1}, b) = \begin{cases} 1 \text{ if } L(I_t, I_{t-1}, b) > T_D \\ 0 \quad \text{in other cases} \end{cases} \quad \text{(Equation 3.40)}$$

where T_D is considered as a tolerance value. Detection is obtained when:

$$\text{Detection if } \sum_{b=1}^{B} c_b L_D(I_t, I_{t-1}, b) > T \quad \text{(Equation 3.41)}$$

where c_b is used to give more or less importance to block b. Most of the time c_b is set to 1 for all the blocks. This likelihood ratio can also be used directly on full-frames.

Another well-known measure involving variance is the Yakimovsky likelihood ratio which can be applied also on blocks or frames directly. For each block this ratio is computed as:

$$L'(I_t, I_{t-1}, b) = \frac{\left(s^2_{\{t,t-1\},b}\right)^2}{s^2_{t-1,b} s^2_{t,b}} \quad \text{(Equation 3.42)}$$

where $s^2_{t,b}$ and $s^2_{t-1,b}$ represent the variances of the pixel intensity values in the frames I_t and I_{t-1} considering a block b. The notation $s^2_{\{t,t-1\},b}$ is used to denote the variance of the pooled data from both frames for a block b.

Freund statistic can also be used to detect shot changes. Distance measure is then defined by:

$$L''(I_t, I_{t-1}, b) = \frac{m_{t,b} - m_{t-1,b}}{\sqrt{\frac{s^2_{t,b} + s^2_{t-1,b}}{XY}}} \quad \text{(Equation 3.43)}$$

Lee et al perform shot change detection using block differences computed in the HSV color space. First RGB images are converted to HSV in order to avoid camera flashes. Then the mean values of Hue and Saturation components are computed for each block. Two successive blocks are compared using these mean values:

$$D_H(I_{t_1}, I_{t_2}, b) = \left| m(I_{t_1}, b, C_H) - m(I_{t_2}, b, C_H) \right| \quad \text{(Equation 3.44)}$$

$$D_S(I_{t_1}, I_{t_2}, b) = \left| m(I_{t_1}, b, C_S) - m(I_{t_2}, b, C_S) \right| \quad \text{(Equation 3.45)}$$

where $m(I_{t_1}, b, C_H)$ is the mean of a block b in the frame I_t considering the color component C_k. D_H and D_S represent respectively differences for Hue and Saturation color component. These two distances are used to determine if each block has changed:

$$D(I_{t_1}, I_{t_2}, b) = \begin{cases} 1 \text{ if } \left(D_H(I_{t_1}, I_{t_2}, b) > T_H\right) \vee \left(D_S(I_{t_1}, I_{t_2}, b) > T_S\right) \\ 0 \quad \text{Otherwise} \end{cases} \quad \text{(Equation 3.46)}$$

Finally the ratio between the number of changed blocks and the total number of blocks is compared to another threshold in order to detect shot changes:

$$\text{Detection if } \frac{1}{B}\sum_{b=1}^{B} D(I_t, I_{t-\Delta}, b) > T \quad \text{(Equation 3.47)}$$

where Δ represents the temporal skip used in the shot change detection process.

Histogram Comparison

Swanberg et al present a method detecting shot changes thanks to the comparison of color histograms computed on the blocks of the images noted $H(I_t, b, C_k, v)$. The detection process is then defined as:

$$\text{Detection if } \left(\sum_{k=1}^{3} \sum_{b=1}^{B} \sum_{v=0}^{V} \frac{\left(H(I_t, b, C_k, v) - H(I_{t-1}, b, C_k, v)\right)^2}{H(I_t, b, C_k, v) + H(I_{t-1}, b, C_k, v)} \right) > T \quad \text{(Equation 3.48)}$$

Nagasaka et al extend their histogram comparison to images divided in 4x4 blocks. Every pair of blocks from two successive frames is compared using the c^2 test on 64 bin histograms:

$$c(I_t, b) = \left(\sum_{v=0}^{63} \frac{\left(H_{64}(I_t, b, v) - H_{64}(I_{t-1}, b, v)\right)^2}{H_{64}(I_t, b, v)} \right) \quad \text{(Equation 3.49)}$$

which requires for each block 4 operations per histogram bin. The values obtained are then sorted in an ascending way and the 8 lowest are kept. The sum of these values is computed and compared to a threshold to detect shot changes:

$$\text{Detection if } \left(\sum_{b=1}^{8} c_s(I_t, b) \right) > T \quad \text{(Equation 3.50)}$$

where the c_s values represent ascending sorted values of c (i.e. for $b \in [1,16]$ we have $c_s(I_t, 1) \le c_s(I_t, b) \le c_s(I_t, 16)$).

Ueda et al proposed to use the rate of correlation change instead of the magnitude of correlation change proposed by Nagasaka. Each value $c(I_t, b)$ obtained from a pair of blocks is compared to a threshold.

The detection depends on the number of significant values $c(I_t,b)$ instead of the sum of the highest $c(I_t,b)$.

Ahmed et al propose also a block-based version of their method using the 6 most significant RGB bits as described in equation 3.21. They compare histograms computed on blocks instead of global histograms. The sum of the histogram differences obtained for each block is computed and compared to a predefined threshold in order to detect shot changes, as shown in:

$$\text{Detection if } \left(\sum_{b=1}^{B} \frac{\sum_{v=0}^{63} \left| H_{64}(I_t,b,v) - H_{64}(I_{t-\Delta},b,v) \right|}{\sum_{v=0}^{63} H_{64}(I_{t-\Delta},b,v)} \right) > T \quad \text{(Equation 3.51)}$$

where Δ represents the temporal skip between two successive frames to be analyzed.

Ahmed et al proposed an improved version of their algorithm described previously. Instead of comparing two frames considering a fixed temporal skip, the method is based on an adaptive temporal skip. First, two images I_{t_1} and I_{t_2} are compared according to equation 3.51. Then if the difference is greater than a threshold, t_2 is replaced by $\frac{t_1+t_2}{2}$ and the frames are again compared. If the difference is still greater than the threshold, t_1 is also set to $\frac{t_1+t_2}{2}$ (considering the current values of t_1 and t_2) and frames are compared. This process is repeated until $t_1+1=t_2$ which represents a shot change between frames t_1 and t_2.

Lee et al introduce a selective HSV histogram comparison algorithm. First, pixels are classified with respect to their color level. If a pixel is characterized by high values for V and S, it is classified into a discrete color using H component. Otherwise the classification is based on the intensity (or gray-level) value. For a given pixel $P(I_t,b,i,j)$, two complementary states are defined:

$$S_{hue}(I_t,b,i,j) = \begin{cases} 1 \text{ if } \left(\left(P(I_t,b,C_S,i,j) > T_S \right) \wedge \left(P(I_t,b,C_V,i,j) > T_V \right) \right) \\ 0 \qquad \text{Otherwise} \end{cases} \quad \text{(Equation 3.52)}$$

$$S_{gray}(I_t,b,i,j) = 1 - S_{hue}(I_t,b,i,j) \quad \text{(Equation 3.53)}$$

Computation of these states requires 4 operations per pixel. For each block two selective histograms $H_{hue}(I_t,b)$ and $H_{gray}(I_t,b)$ are then computed in a classical way considering the two states previously defined. The notation $H_{hue}(I_t,b,v)$ (respectively $H_{gray}(I_t,b,v)$) represents the number of pixels in block b of frame I_t with state S_{hue} (respectively S_{gray}) equal to 1 and with hue (respectively intensity or gray-level) value equal to v. These histograms are used for shot change detection:

Detection if

$$\left(\sum_{b=1}^{B}\left(\sum_{v=0}^{V}\left|H_{hue}(I_t,b,v)-H_{hue}(I_{t-1},b,v)\right|+\sum_{v=0}^{V}\left|H_{gray}(I_t,b,v)-H_{gray}(I_{t-1},b,v)\right|\right)\right)>T$$

(Equation 3.54)

Bertini et al compare in the HSI color space histograms of successive frames divided in 9 blocks. In order to improve robustness to change in lighting conditions, the Intensity component is not used. The detection can then be represented by the following equations:

$$D_{HS}(I_t,I_{t+1},b)=\sum_{k\in\{H,S\}}\sum_{v=0}^{V}\left(H(I_t,b,C_k,v)-H(I_{t+1},b,C_k,v)\right) \quad \text{(Equation 3.55)}$$

$$D'_{HS}(I_t)=\sum_{b=1}^{B}D_{HS}(I_t,I_{t+1},b)-\sum_{b=1}^{B}D_{HS}(I_{t-1},I_t,b) \quad \text{(Equation 3.56)}$$

Detection if $(((D'_{HS}(I_t)>0)\wedge(D'_{HS}(I_{t+1})<0))\vee((D'_{HS}(I_t)<0)\wedge(D'_{HS}(I_{t+1})>0)))$

(Equation 3.57)

Chahir et al [19] based their method on histogram intersection computed on frames divided in 24 blocks. The color space used in their method is L*u*v*. For each block, color histogram intersection is computed between two successive frames requiring 12 operations per bin. A comparison with a threshold allows determining whether the block has been changed or not. The number of changed blocks is then compared to another threshold in order to detect a shot change.

Combination of Histogram differences & Likelihood rate

This method proposed by Dugad is based on two successive steps to detect cuts and other transitions. Shot changes are detected using successively histogram difference and likelihood ratio. In this method three thresholds have to be set. Histogram difference step is defined as in equation 3.16 and is compared with two thresholds. The difference is first compared to a high threshold in order to avoid false alarms. If it is lower than this threshold, it is then compared to a low threshold. If it is higher than this low threshold, the final decision is taken by computing likelihood ratio values. In this case, the two frames to be compared are divided in 64 blocks and the 16 central blocks are kept. For each block, the likelihood ratio is computed between the block $P(I_t, b)$ and the blocks $P(I_t, b')$ where b' belongs to the neighborhood of b, and the minimum of the likelihood value is kept. Then a mean of the 16 minimum likelihood ratios is computed and is compared with the third threshold, which may result in shot change detection.

Use of Neighborhood color ratio

Adjeroh et al compare two successive frames using neighborhood color ratios. A local averaging step is first performed in order to obtain one value $P'(I_t, b)$ per block:

$$P'(I_t, b) = \prod_{i=2}^{X-1} \prod_{j=2}^{Y-1} \frac{1}{4P(I_t, b, i, j)} \left(P(I_t, b, i-1, j) + P(I_t, b, i+1, j) + P(I_t, b, i, j-1) + P(I_t, b, i, j+1) \right)$$

(Equation 3.58)

Pairs of blocks from two different frames are then compared using this measure:

$$D'(I_t, I_{t-\Delta}, b) = 1 - \min\left(\frac{P'(I_t, b)}{P'(I_{t-\Delta}, b)}, \frac{P'(I_{t-\Delta}, b)}{P'(I_t, b)} \right)$$ (Equation 3.59)

where Δ represents the temporal skip. Shot changes are finally detected if the number of significant D' values for all selected blocks is higher than a fixed threshold, or if the average value of D' is higher than another threshold.

Evolution of block dissimilarity

Shot changes can also be detected by analyzing the evolution of block dissimilarity. Demarty et al compute locally a distance criterion in RGB color space between blocks of two successive images. Result obtained consists in distance values between the two images for every block. Then the sum of these values is computed and an evolution curve of this sum is built. This evolution curve is filtered using a top-hat morphological operation and is finally compared with a threshold in order to detect shot changes.

Lefevre et al proposed a method using HSV color space on block-sampled images in order to avoid false detection due to illumination effects. A value is obtained for each block from Hue and Saturation components. Then a block-based difference is computed between two frames based on the block values. This difference is tracked through time, as well as its derivative. Analysis of this derivative allows cut detection, whereas the initial (non-derivated) difference values are used to initialize a cumulative sum computation of the derived values. This allows detection of gradual transitions.

Temporal & Spatial sub-sampling

Xiong et al propose to sub-sample the video sequence in both space and time. An abrupt change is detected between two frames I_t and $I_{t+\Delta}$ if:

$$\text{Detection if } \left(\sum_{b \in B'} D_m\left(I_t, I_{t+\Delta}, b\right) \right) > T \quad \text{(Equation 3.60)}$$

where B' represents a set of a priori selected blocks and $D_m\left(I_{t_1}, I_{t_2}, b\right)$ is defined as:

$$D_m\left(I_{t1}, I_{t2}, b\right) = \begin{cases} 1 & \text{if } \left| m_{t_1,b} - m_{t_2,b} \right| > T_m \\ 0 & \text{in other cases} \end{cases} \quad \text{(Equation 3.61)}$$

Gradual transitions are detected using an edge-based frame-to-frame difference measure. If a shot change is detected, a binary search is performed reducing Δ to determine the exact shot boundaries. The method proposed is called "Net Comparison" and has also been tested in HSV color space.

Feature-based Methods

All the methods we have already presented were using features, but they can be qualified of trivial features. Here we are considering more sophisticated ones. We consider:

- The moments computed on the image.
- The contour lines extracted from the image.
- Some feature points extracted using Hough Transform.
- The planar points.
- Color transition.
- Modeling of the video transition effects.
- The use of some decision process as Bayesian methods.
- Features computed from classical statistical approaches.
- The use of Hidden Markov Models.

Moment invariants

Arman et al use moment invariants combined with histogram intersection to detect shot changes. Moment invariants have properties such as invariance to scale change, rotation, and translation. The moments of a frame I_t are defined as:

$$m_{p,q} = \sum_{i=1}^{X}\sum_{j=1}^{Y} i^p j^q P(I_t, i, j) \quad \text{(Equation 3.62)}$$

Dailianas proposed that shot changes are detected thanks to the computation of the usual Euclidean distance between two frames using a vector composed of the first three moment invariants, defined as:

$$\vec{\Phi} = \begin{pmatrix} m_{2,0} + m_{0,2} \\ (m_{2,0} - m_{0,2})^2 + 4m_{1,1}^2 \\ (m_{3,0} - 3m_{1,2})^2 + (3m_{2,1} - m_{0,3})^2 \end{pmatrix} \quad \text{(Equation 3.63)}$$

Considering equation 3.62, all moments used in equation 3.63 require 3 operations per pixel; expecting $m_{1,2}$ and $m_{2,1}$ which need 4 operations per pixel. The detection can be finally defined as:

$$\text{Detection if } \left(\left\| \overrightarrow{\Phi(I_t)} - \overrightarrow{\Phi(I_{t-1})} \right\|^2 \right) > T \quad \text{(Equation 3.64)}$$

Edges

Zabih et al use edge extraction to detect shot changes. Global motion compensation is per-formed on successive frames. Next, edges are extracted using Canny algorithm and dilated. Normalized proportions of entering edges and exiting edges are then computed using the following equations:

$$P(C_{out}, I_t) = 1 - \frac{\sum_{i=1}^{X}\sum_{j=1}^{Y} E_d(I_{t-1}, i + a_{t-1,t}, j + b_{t-1,t}) E_d(I_t, i, j)}{\sum_{i=1}^{X}\sum_{j=1}^{Y} E(I_{t-1}, i, j)} \quad \text{(Equation 3.65)}$$

$$P(C_{in}, I_t) = 1 - \frac{\sum_{i=1}^{X}\sum_{j=1}^{Y} E_d(I_{t-1}, i + a_{t-1,t}, j + b_{t-1,t}) E(I_t, i, j)}{\sum_{i=1}^{X}\sum_{j=1}^{Y} E(I_{t-1}, i + a_{t-1,t}, j + b_{t-1,t})} \quad \text{(Equation 3.66)}$$

where E and E_d are respectively the contour image and its dilated version, and $(a_{t-1,t}, b_{t-1,t})$ represents the global motion translation vector between the two successive images I_t and I_{t-1}. Then a dissimilarity measure $ECF(I_t)$ called edge change fraction is computed by:

$$ECF(I_t) = \max(P(C_{out}, I_t), P(C_{in}, I_t)) \quad \text{(Equation 3.67)}$$

Finally this value is compared to a threshold to detect shot changes:

$$\text{Detection if } ECF(I_t) > T \quad \text{(Equation 3.68)}$$

Smeaton et al proposed an evolution of the previous method where the detection is based on the evolution of the edge change fraction on several frames instead of the analysis of this dissimilarity measure on only one frame. Detection can then be defined by:

$$\text{Detection if } (ECF(I_t) - ECF(I_{t-1})) > T \quad \text{(Equation 3.69)}$$

Lienhart also uses edge information to perform dissolve detection. First edges extracted with the Canny edge detector are confronted with two thresholds to determine weak and strong edges:

$$E_w(I_t,i,j)=\begin{cases}E(I_t,i,j) & \text{if } T_w \le E(I_t,i,j) \le T_s \\ 0 & \text{in other cases}\end{cases} \quad \text{(Equation 3.70)}$$

$$E_s(I_t,i,j)=\begin{cases}E(I_t,i,j) & \text{if } T_s \le E(I_t,i,j) \\ 0 & \text{in other cases}\end{cases} \quad \text{(Equation 3.71)}$$

where T_w and T_s are respectively the lowest and highest thresholds for detecting weak and strong edges. E_w and E_s represent the weak and strong edge images. Then the edge-based contrast EC is obtained for a frame I_t according to the equation:

$$EC(I_t)=1+\frac{\sum_{i=1}^{X}\sum_{j=1}^{Y}E_s(I_t,i,j)-\sum_{i=1}^{X}\sum_{j=1}^{Y}E_w(I_t,i,j)-1}{\sum_{i=1}^{X}\sum_{j=1}^{Y}E_s(I_t,i,j)+\sum_{i=1}^{X}\sum_{j=1}^{Y}E_w(I_t,i,j)+1} \quad \text{(Equation 3.72)}$$

Finally dissolves are detected when the current value of EC is a local minimum.

Yu et al use edge information to detect gradual transitions. Cuts are first detected using a histogram difference measure computed between two successive sub-sampled frames. Then a second pass is necessary for detecting gradual transitions. For every frame I_t between two successive cuts at time t_1 and t_2, the number $Q_E(I_t)$ of edge points present in the image is computed and temporally smoothed. Then for every local minima $Q_E(I_{t'})$ which is below a predefined threshold, a search of the two closest local maxima $Q_E(I_{t_1'})$ and $Q_E(I_{t_2'})$ is performed with $t_1' < t' < t_2'$. A fade out effect is detected between t_1' and t' if:

$$\text{Detection if } \left(\frac{1}{t'-t_1'}\sum_{t=t_1'}^{t'}\left|Q_E(I_t)-Q_E(I_{t-1})\right|\right) \in \left[T_{\text{low}}^{\text{fade out}}, T_{\text{high}}^{\text{fade out}}\right] \quad \text{(Equation 3.73)}$$

Similarly, a fade in effect is detected between t' and t_2' if:

$$\text{Detection if } \left(\frac{1}{t_2'-t'}\sum_{t=t'}^{t_2'}\left|Q_E(I_t)-Q_E(I_{t-1})\right|\right) \in \left[T_{\text{low}}^{\text{fade in}}, T_{\text{high}}^{\text{fade in}}\right] \quad \text{(Equation 3.74)}$$

For dissolve effect detection, a new measure called double chromatic difference is computed for every frame belonging to the interval $[t_1', t_2']$:

$$DCD(I_t) = \sum_{i=1}^{X}\sum_{j=1}^{Y}\left|\frac{1}{2}P(I_{t_1'},i,j)+\frac{1}{2}P(I_{t_2'},i,j)-P(I_t,i,j)\right| \quad \text{(Equation 3.75)}$$

If the frame number $t_{\min}$ corresponding to the global minimum value $DCD(I_{t_{\min}})$ is close enough to t' ($|t'-t_{\min}|<e$ with e defined as a small number), a dissolve effect is assumed to be found between frames $I_{t_1'}$ and $I_{t_2'}$.

Heng and Ngan also propose a method based on edge information. They introduce the notion of edge object, considering the pixels close to the edge. Occurrences of every edge object are matched on two successive frames. Shot changes are detected using ratio between the amount of edge objects persistent over time and the total amount of edge objects.

Nam and Tewfik propose a coarse-to-fine shot change detection method based on wavelet transforms. Image sequences are first temporally sub-sampled. Frames processed are also spatially reduced using a spatial 2-D wavelet transform. Intensity evolution of pixels belonging to coarse frames is analyzed using a temporal 1-D wavelet transform. Sharp edges define possible shot change locations. Video frames around these locations are further processed at full-rate. Temporal 1-D wavelet transform is applied again on the full-rate video sequence. Edge detection is also performed on every coarse frame and the number of edge points is computed on a block-based basis. Difference between two successive frames is computed using the number of edge points for each block. True shot boundaries are located on sharp edges in the 1-D wavelet transform and high values of inter-frame difference considering block-based amount of edge points. An extension to wipe transitions detection has been proposed by Nam and Tewfik also.

Feature points

Ardebilian et al detect shot changes by comparing between feature points extracted from two successive images. They use Hough transform to extract feature points. Success or not of the feature points matching between two successive frames results directly in cut detection.

Planar points

Silva et al perform shot change detection using spatio-temporal representation of an image sequence. A video sequence is represented in $\Re^4$ as a hyper-surface:

$$V = \{i, j, t, P(I_t, i, j)\} \quad \text{(Equation 3.76)}$$

The amount of planar points in every frame is considered as the measure for detecting cuts. We recall that planar points are points contained in a flat neighborhood of the hyper-surface. For a given frame I_t, planar points are determined using the characteristic polynomial coefficients of the Hessian matrix of $P(I_t, i, j)$. Then the percentage of planar points is computed. A cut is detected (in a four frame interval) when this value is greater than three times the temporal variance of the percentage of planar points.

Color Transitions

Sanchez et al compare between two successive frames using color histograms computed on specific regions. These regions are defined from the most significant color transitions of the image, considered as high values of its multi-spectral gradient and computed with Sobel approximation. Color histograms are compared between regions of two successive frames to determine the coherence of the region through time. Shot changes are finally detected if the amount of changed regions is above a given threshold.

Transition modeling

Some shot changes are created from production effects. These transitions can be modeled explicitly with mathematical tools in order to be detected. Several methods using these assumptions are presented below.

Hampapur et al model several kinds of fades and wipes with mathematical functions. Knowing the two last shots and their respective durations, it is possible to estimate the duration of the current shot. Detection of shot changes can rely on a constancy measure defined for frame I_t as:

$$C(I_t) = \frac{\sum_{i=1}^{X}\sum_{j=1}^{Y} S_{binary}(I_t, i, j)}{s_t\left(1 + \left|X_C(I_t) - \frac{X}{2}\right| + \left|Y_C(I_t) - \frac{Y}{2}\right|\right)} \quad \text{(Equation 3.77)}$$

where s_t represents the standard deviation of pixel intensities in frame I_t. The binary state $S_{binary}(I_t, i, j)$ of a pixel is defined as:

$$S_{binary}(I_t, i, j) = \begin{cases} 1 & \text{if } P(I_t, i, j) \neq 0 \\ 0 & \text{otherwise} \end{cases} \quad \text{(Equation 3.78)}$$

The components of the centroid of image I_t are noted $X_c(I_t)$ and $Y_c(I_t)$. They are computed as:

$$X_C(I_t) = \frac{\sum_{i=1}^{X}\sum_{j=1}^{Y} iP(I_t, i, j)}{\sum_{i=1}^{X}\sum_{j=1}^{Y} P(I_t, i, j)} \quad \text{(Equation 3.79)}$$

$$Y_C(I_t) = \frac{\sum_{i=1}^{X}\sum_{j=1}^{Y} jP(I_t, i, j)}{\sum_{i=1}^{X}\sum_{j=1}^{Y} P(I_t, i, j)} \quad \text{(Equation 3.80)}$$

Adami et al perform dissolve detection applying a model of dissolve effects on frame histograms. For every frame, two specific histograms are computed:

$$\overline{H}(I_t) = H_{\frac{1}{2}}(I_t) * H_{\frac{1}{2}}(I_t) \quad \text{(Equation 3.81)}$$

$$H_{\Delta}(I_t) = H_{\frac{1}{2}}(I_{t-\Delta}) * H_{\frac{1}{2}}(I_{t+\Delta}) \quad \text{(Equation 3.82)}$$

where $H_{\frac{1}{2}}(I_t)$ represents the histogram of the frame I_t scaled by half, Δ is a fixed parameter, and the operator $*$ figures a convolution. It is then possible to compute a dissimilarity measure using these histograms:

$$R(I_t) = \frac{c^2\left(H(I_t), \overline{H}(I_t)\right)}{c^2\left(H(I_t)H_\Delta(I_t)\right)} - 1 \quad \text{(Equation 3.83)}$$

where the c^2 operator is computed between two histograms. Maxima values of $R(I_t)$ indicate dissolve locations:

$$\text{Detection if } R(I_t) > R(I_{t'}) \forall t' \text{ in the neighborhood of } t \quad \text{(Equation 3.84)}$$

Aigrain et al detect shot changes using the assumption of the absence of important motion in the different shots. Their method is based on a motion differential model and uses density function estimation as the difference between two images. First, two successive images are reduced spatially and normalized using histogram equalization. Then the histogram of pixel-pair difference is computed and is simplified to two values, which are respectively the amount of differences belonging to the interval [128, 255] computed on normalized images and the amount of differences belonging to the interval [1, 40] computed on non-normalized images. Cut and gradual transition detections are respectively based on local maxima of the first and second value described previously.

Lienhart relies on fade modeling from Hampapur to perform fade detection. The proposed algorithm uses the standard deviation of pixel intensities as an estimation of the scaling function introduced in fade effects. First, all monochrome frames are located as they are fade start or end candidates. These frames are characterized by a standard deviation $s(I_t)$ close to zero. Fades are then detected by searching in both temporal directions for a linear increase in the standard deviation.

Alattar bases also the shot change detection on variance of pixel intensities. Fades are detected using a two steps scheme. First local minimum values of the second order difference of the pixel intensity spatial variance time series are obtained. Then a test is performed to determine whether the first order difference of the pixel intensity mean is relatively constant in the neighborhood of the local variance minimum or not. In the positive case, a fade is assumed to be found. A similar method by Alattar also has been proposed for dissolve.

Truong et al combine approaches from Lienhart and Alattar. First, all monochrome frames are detected. Then only monochrome frames, which are next to a local minimum of the intensity variance second order difference, are kept. Finally a test is performed on the first order difference of the mean intensity. If this value is constant through time and other conditions are satisfied, a fade has been detected. The other conditions correspond to comparison between thresholds and the absolute value of the first order difference and the intensity variance of the first or last frame. Dissolve detection is performed using the evolution of the variance first order difference through time. This difference value should be monotonically increasing from a negative value up to a positive value. So zero crossings are used to locate dissolve frames.

Fernando et al use also mean and variance of the luminance signal to determine fade and dissolve transitions locations. For every frame, the mean and the variance of the luminance are computed. The ratio between second temporal derivative of the variance to the first temporal derivative of the mean is then compared between two successive frames. Shot changes are located when this ratio is constant through time.

Bayesian approaches

Vasconcelos et al propose a segmentation method using a Bayesian model of the editing process. For each frame a local activity measure is computed based on a tangent distance. In order to detect a shot change, this measure is compared (following a Bayesian framework) to an adaptive threshold, depending on the a priori duration of a shot and on the time elapsed between the previous shot change and the current frame I_t.

Hanjalic et al use also a statistical framework for the shot change detection, which is modeled as a probability minimization problem of the average detection error. Detection criteria are linked with visual content discontinuity (based on motion compensation) and knowledge about shot length distribution.

Han et al base their detection on gray-level or color histogram differences computed between successive frames using equation 3.16 or equation 3.18. A filtering step combining an average clipping operation and a subsequent local convolution is used to improve the shot change detection. The evolution curve of the filtered histogram difference value is analyzed and decision for the detection of a shot change is taken following a Bayesian framework. Detections of a cut or a gradual effect are respectively linked with the presence in the evolution curve of a rectangular or triangular shape.

Statistical approaches

Yilmaz et al use Principal Coordinate System to perform shot change detection on RGB frames. First, image rows are concatenated in order to obtain only one row vector per color component for each frame. We use the notations $V(I_t, C_R)$, $V(I_t, C_G)$ and $V(I_t, C_B)$ for the row vectors associated with the Red, Green and Blue components. Then the 3x3 covariance matrix $M(I_t)$ of the RGB color space is computed following:

$$M(I_t) = \begin{pmatrix} V(I_t, C_R) \\ V(I_t, C_G) \\ V(I_t, C_B) \end{pmatrix} \left(V(I_t, C_R)^T V(I_t, C_G)^T V(I_t, C_B)^T \right) \quad \text{(Equation 3.85)}$$

Next the vector representing the principal axis is selected and noted $V_{l_{\max}}(I_t)$. We recall this vector is the eigenvector associated with the maximum eigenvalue $l_{\max}$ of the covariance matrix. Finally two successive frames are compared with respect to the angle between their respective principal axes following the equation:

$$\text{Detection if } \left(1 - \frac{V_{l_{\max}}(I_t)^T V_{l_{\max}}(I_{t-1})}{\left\| V_{l_{\max}}(I_t) \right\| \left\| V_{l_{\max}}(I_{t-1}) \right\|} \right) > T \quad \text{(Equation 3.86)}$$

Han et al use also a principal component analysis to perform shot change detection. First frames are sub-sampled and represented as column vectors. Then successive frames are grouped in a temporal window. The mean vector m and the empirical covariance matrix M of this window are computed. Then the unique set of orthonormal eigenvectors of M and their associated eigenvalues are obtained. Each frame in the window is then projected onto the K eigenvectors corresponding to the K largest eigenvalues. Finally shot changes are detected using the temporal variations of angle and length of the K first principal components.

Li et al based their algorithm on the computation of joint probability images between frames. They use a spatio-temporal representation of the successive joint probability images obtained in order to detect shot changes. First a joint probability image is computed between two frames, which consist in the frequency of the co-occurences of intensity or chrominance values. Two similar images I_{t_1} and I_{t_2} will be characterized by a joint probability image $J\left(I_{t_1}, I_{t_2}\right)$ composed of non-zero values on the diagonal. A distance measure is then defined between two frames using the joint probability image:

$$D_J\left(I_{t_1}, I_{t_2}\right) = 1 - \frac{\sum_{(i,j)\in\Im'} J\left(I_{t_1}, I_{t_2}, i, j\right)}{\sum_{(i,j)\in\Im} J\left(I_{t_1}, I_{t_2}, i, j\right)} \quad \text{(Equation 3.87)}$$

where $\Im$ and $\Im'$ represent respectively the set of all pixels (i, j) in the joint probability image and the set of all pixels near the diagonal line with a given tolerance d (i.e. $\Im' = \{(i, j) : |i - j| < d\}$). If the value D_J obtained is higher than a fixed threshold, several algorithms are used in order to confirm the presence of a shot change. Dissolve effects are detected using histogram intersection performed on spatio-temporal representations of joint probability images.

Gong and Liu perform shot change detection using the Singular Value Decomposition. Every frame is divided in 3×3 blocks on which a 3-D color histogram composed of 125 bins is computed. A vector of 1125 components is then obtained for every frame. The video sequence is represented by a matrix which is processed by a singular value decomposition algorithm. The K largest singular values are kept and are used to compute a similarity measure between two frames. Detection of a shot change is done by comparing the similarity computed between the two frames I_{t_1} and I_{t_2} with a low and a high threshold. If the similarity measure is below the low threshold, no shot change has been detected. On the contrary, if the measure is higher than the high threshold, a shot change is assumed to be found. In the last case (i.e. the similarity measure is between the two thresholds), a refinement step is performed involving frames between I_{t_1} and I_{t_2}.

Hidden Markov models

Eickeler et al use Hidden Markov Models to perform video indexing. Some of the classes represent shot boundary frames. Several features are defined to describe each frame, but only some of them characterize shot boundary frames:

$$d_1(I_t) = \min\left(\frac{\sum_{i=1}^{X}\sum_{j=1}^{Y}\left|P(I_t,i,j)-P(I_{t-1},i,j)\right|}{XY}, \frac{\sum_{i=1}^{X}\sum_{j=1}^{Y}\left|P(I_{t+1},i,j)-P(I_{t-2},i,j)\right|}{XY}\right) \quad \text{(Equation 3.88)}$$

$$d_2(I_t) = \sum_{v=0}^{V}\left(\left|H(I_t,v)-H(I_{t-1},v)\right| - median\left(\begin{matrix}\left|H(I_{t-1},v)-H(I_{t-2},v)\right|,\\ \left|H(I_t,v)-H(I_{t-1},v)\right|,\\ \left|H(I_{t+1},v)-H(I_t,v)\right|\end{matrix}\right)\right) \quad \text{(Equation 3.89)}$$

$$d_3(I_t) = \sum_{i=1}^{X}\sum_{j=1}^{Y}\frac{P(I_t,i,j)-P(I_{t-1},i,j)}{P(I_t,i,j)-0{,}5P(I_{t-1},i,j)+P(I_{t+1},i,j)} \quad \text{(Equation 3.90)}$$

where $d_1(I_t)$, $d_2(I_t)$, and $d_3(I_t)$ represent respectively the intensity of motion, the median filtered intensity of difference histograms, and the ratio between the difference pixel and the difference from interpolated pixel. After a learning step using the Baum-Welch algorithm, segmentation is performed using the Viterbi algorithm.

A similar approach using Hidden Markov Models has been proposed by Boreczky . The model is based on image, audio, and motion features. Classification of a frame into a shot boundary class is done using only the image feature of the frame. This feature is defined as a luminance 64 bin histogram difference similar to the one described in equation 3.16.

Motion-based Methods

As the nature of motion is usually continuous in a video sequence, it can also be used as a criterion to detect shot changes. Based on this fact, several approaches using motion information were proposed in the literature. We review here methods based on global (or camera) motion, motion vectors, optical flow, and correlation in the frequency domain.

Global motion

Cherfaoui and Bertin detect shot changes in two steps. First the global motion parameters are estimated using an affine transformation model. The estimated motion is then used to classify a shot as fixed, pan, or zoom. If the motion is not coherent through time, a shot change is assumed to be found.

Bouthemy et al based their detection on dominant multi-resolution motion estimation. This estimation uses a global 2-D parametric model composed of 6 parameters. Once the dominant motion has been estimated, a coefficient $w_{i,j}$ is also available for every pixel (i, j). It represents the coherence of the pixel with the dominant motion estimated. Using a predefined threshold, it is possible to define the set of dominant motion-coherent pixels in each frame. The evolution of the set size through time is analyzed in order to detect shot changes.

Zugaj et al extend the previous method to wipe detection. Here only pixels which are non-coherent with the estimated dominant motion are considered. For each frame, two histograms are computed based on the number of non-coherent pixels along horizontal and vertical axes. For every couple of frames, absolute differences between corresponding histograms are computed and result in two other histograms. The correlation between two successive absolute difference histograms is

then measured along the two axes. If one of the two correlation values exceeds a predefined threshold, a horizontal or vertical wipe is detected.

Mann et al proposed a method where global motion estimation is performed using a 8 parameter model. They define "video orbits" as collection of pictures starting from one picture and applying all possible projective coordinate transformations to that picture using the 8 motion parameters. Two frames belonging to the same scene will lie in the same orbit or nearly so. So shot changes are detected when the distance between the orbits of successive frames is higher than a threshold.

Motion vectors

Akutsu et al use a motion smoothness measure to detect shot changes. The indexing method uses sub-sampled video sequences and processes only one frame every k frames. Then the selected frame is divided into 8x8 blocks and each block is matched to a block in the next chosen frame. Motion vector is estimated thanks to the closest matched neighboring block, which is also used to compute the value of the correlation coefficient. An inter-frame similarity measure can be defined as the average of these correlations. Another measure called motion smoothness is defined as the ratio between the number of blocks which have significantly moved and the displacement of these blocks. Shot changes are finally detected in presence of local extrema in the correlation and motion smoothness ratio values.

Shahraray proposed a similar method. Sub-sampled video sequences are used and every frame is divided in 12 blocks. A research is performed to match each block from one frame to the most similar block (in a spatial neighborhood) in the next frame. Motion vector and correlation value are computed in a way similar to the Akutsu et al method. Main difference with the previous method is the use of a nonlinear digital order statistic filter. Correlation values are sorted into ascending order and the first values and their respective motion vectors are used for the computation of an average value which is considered as a similarity measure. As in Akutsu et al, a local temporal extremum in the similarity measure means shot change detection. A motion-controlled temporal filter is used to avoid false detection due to motion.

Liu et al based their method on motion compensated images obtained from motion vector information. First motion vectors of frame I_{t-1} are used to create a motion compensated version I'_t of the frame I_t. The next step is luminance normalization. The motion compensated frame I'_t is normalized in order to get the same energy as the original frame I_t. Normalized image is noted I''_t. The original frame I_t is then compared to

the two modified frames I'_t and I''_t using a modified version of the c^2 test applied on their histograms. The result $c(I_t, I'_t)$ is compared to an adaptive threshold in order to detect cuts. Fade detection is based on the comparison between $c(I_t, I'_t)$ and $c(I_t, I''_t)$ which are the two histogram differences computed previously.

Optical flow

Fatemi et al use optical flow as information to detect shot changes. First the video sequence is divided into overlapping subsequences, defined as 3 consecutive frames and a fourth predicted frame. Every frame is then divided into B blocks, which are predicted from the first frame to the second one and from the second frame to the third one. Finally a set of 3 matching blocks from the first three frames is used for block prediction into the last frame. If the block prediction does not correctly estimate the content of the last frame, a shot change is assumed to be found.

Frequency domain correlation

Porter et al propose a technique inspired by motion-based algorithms. Correlation between two successive frames is computed and used as a shot change detection measure. In order to compute the inter-frame correlation, a block-based approach working in the frequency domain is taken. Frames are divided into blocks of 32×32 pixels. Every block in a frame I_{t-1} is matched with a neighboring block in frame I_t by first computing the normalized correlation between blocks and then seeks and located the correlation coefficient with the largest magnitude. The normalized correlation is computed in the frequency domain instead of the spatial domain to limit computation time. The average correlation is then obtained for a couple of frames. Shot changes are detected in presence of local minima of this value.

Chapter 4

Video Segmentation & Modeling Digital Video

A video data model was presented in a previous chapter. This data model uses a temporal interval as the basic unit of video. The process of determining the temporal interval of a video data model based on some criteria is termed video segmentation. Several different types of segmentation criteria can be used to segment a video. Subjective criteria like change in emotion to objective criteria like change in audio volume levels are all valid criteria. Video is treated as a visual medium, so the goal of segmentation is to identify the smallest visual unit of a video, the shot.

The problem of segmenting digital video occurs in many different application of video. Multimedia authoring systems which reuse produced video need access to video in terms of video shots. The edit detection algorithms presented in this chapter can be used in digital video editing systems for edit logging operations. There are several other applications in video archiving and movie production which can use the segmentation techniques presented here.

Video segmentation requires the use of explicit models of video. Most of the current approaches to video segmentation do not use explicit models. A review of these techniques has been presented in the previous chapter (chapter 3). They pose the problem as one of detecting camera motion breaks in arbitrary image sequences. The solutions that have been presented typically involve the application of various low level image processing operations to the video sequences. These approaches have not utilized the inherent structure of video. Defining models of video which capture the structure provides the constraints necessary for effective video segmentation. The work presented in this chapter uses the production model based classification approach to video segmentation.

This chapter presents a video edit model which is based on a study of video production processes. This model captures the essential aspects of video editing. Video features extractors for measuring image sequence properties are designed based on the video edit model. The extracted features are used in a production model based classification formulation to segment the video. The models are also used to define error measures, which in conjunction with test videos and correct video models are used to evaluate the performance of the segmentation system.

Video Edit Model

The video edit model presented here captures the process of editing and assembly discussed in chapter 2. The model has three components, the **Edit Decision Model** which models the output of the editing, the **Assembly Model** which represents the assembling phase of video production and the **Edit Effect Model** which describes the exact nature of the image sequence transformation that occurs during the different types of edit effects.

Consider a set of shots $S = (S_1, S_2, \ldots S_N)$ with cardinality N. Each shot $S_i \in S$ can be represented by a closed time interval:

$$S_i = [t_b, t_e] \quad \text{(Equation 4.0)}$$

where t_b is the time at which the shot begins and t_e is the time at which the shot ends. Before the final cut is made $t_b = 0 \forall S_i \in S$ since there is no relative ordering of the shots. Let $E = (E_1, E_2, \ldots E_k)$ be the set edits available. Each edit E is represented by a triple:

$$E = \{[t_b, t_e], t, e\} \quad \text{(Equation 4.1)}$$

where $[t_b, t_e]$ is the duration of the edit, $t \in (t_1, t_2, \ldots t_n)$ is the type of the edit (*cuts*, *fades*, *dissolves*) and e is the mathematical transformation that is applied during an edit or the **edit effect model**. Consider a video V. Let V be composed of n shots taken from the set S. Then the **Edit Decision Model** V_{edm} can be represented as follows:

$$V_{edm} = S_1 \circ E_{12}(S_1, S_2) \circ S_2 \circ E_{23}(S_2, S_3) \circ \ldots \circ S_{(n-1)} \circ E_{(n-1)n}(S_{(n-1)}, S_n) \circ S_n \quad \text{(Equation 4.2)}$$

where $S_i \in S$, the subscript i denotes the temporal position of the shot in the sequence (i.e. if $i < j$, shot S_i appears before shot S_j in the *final cut*) and E_{ij} denotes the edit transition between shots S_i and S_j. The $\circ$ denotes the concatenation operation and $E_{ij} \in E$. The **Assembly Model** for V is given by:

$$V_{am} = S_1 \circ S_{e12} \circ S_2 \circ S_{e23} \circ \ldots \circ S_{(n-1)} \circ S_{e(n-1)n} \circ S_n \quad \text{(Equation 4.3)}$$

where S_x represent the shots used to compose the video and S_{exx} represent the edit frames. The **assembly model** can be rewritten as follows:

$$V_c = s_1 \circ s_2 \circ s_3 \circ s_4 \circ \ldots \circ s_{(n-1)} \circ s_n \circ s_{(n+1)} \circ s_{(2n-1)} \quad \text{(Equation 4.4)}$$
$$s_i = \{[t_b, t_e], l_i\}$$
$$l_i \in (edit, shot)$$

V_c is called the **video computational model.** In this representation, every segment of video is a temporal interval with a label indicating the content. Segmentation of video requires two labels namely *(shot, edit)*. It is important to keep in mind that the *edit frames* are also image sequences, they differ from shots in the production technique used. Cuts are a special type of interval with zero length. This computational model of video is used to define error measurements.

Edit Effect Model

The edit effects commonly used in video production are modeled by using **2D image transformations**. The mathematical model for the process of generating edit frames from a pair of shots is discussed in this paragraph. Consider an image sequence, where the pixel positions are denoted by x, y and the frame number is denoted by t. Let r, g, b denote the three (red, green, blue) color values assigned to each pixel. Let the image space be denoted by $\vec{x} = \{x, y, t, 1\}$ and the color space be denoted by $\vec{h} = \{r, g, b, 1\}$. *Homogeneous Coordinates* are used for representing both the image and color spaces in order to accommodate affine transformations. Using these notations and assuming linear affine transformations, the possible set of edit frames $E(x, y, t)$ given two shots $S_{out}(x, y, t)$ and $S_{in}(x, y, t)$ can be denoted as follows:

$$E(x, y, t) = S_{out}\left(\vec{x} \times T_{s1}\right) \times T_{c1} \otimes S_{in}\left(\vec{x} \times T_{s2}\right) \times T_{c2} \quad \text{(Equation 4.5)}$$

Here S_{out} represents the shot before the edit or the out going shot and S_{in} represents the shot after the edit or the incoming shot, T_s and T_c denote the transformation applied to the pixel and color space of the shots being edited and $\otimes$ denotes the function used to combine the two shots in order to generate the set of edit frames. In general T_s and T_c can be any type of transformation (linear, non-linear) and $\otimes$ can be any operation. In practice however, the transformations are either linear or piecewise linear and the operation $\otimes$ is addition. The remainder of the discussion assumes this simplified edit effect model. Edit effects are classified into four types based on the nature of the transformation used during the editing

process. Table 4.0 shows a classification of edit effects. $\vec{I}$ denotes the identity transformation.

Type	Name	T_s	T_c	Meaning	Duration	Examples
1	*Identity*	$\vec{I}$	$\vec{I}$	Concatenate Shots	Zero	Cut
2	*Spatial*	T_s	$\vec{I}$	Spatial Changes	Finite	Translate, Page
3	*Chromatic*	$\vec{I}$	T_c	Intensity Changes	Finite	Fade, Dissolve
4	*Combined*	T_s	T_c	Spatial & Intensity Changes	Finite	Morphing

Table 4.0: Edit Types

The classification presented in table 4.0 has a simple physical explanation. The classes correspond to the different types of operations that an editor can perform when editing two shots. The options are:

Type 1: Identity Class: Here the editing process does not modify either of the shots. No additional edit frames are added to the final cut. Cuts comprise this class of edits.
Type 2: Spatial Class: Only the spatial aspect of the two shots is manipulated by this class of edit effects. The color space of the shots is not affected. This class is comprised of effects like page translates, page turns, shattering edits and many other digital video effects.
Type 3: Chromatic Class: Edits in this class include fade in's, fade out's and dissolves. Here the edit effect manipulates the color space of either of the shots without changing the spatial relation of any of the pixels.
Type 4: Spatio-Chromatic Class: Here both the space and color aspects of the shots is simultaneously manipulated during the editing process. This class consists of effects like image morphing and wipes.

Problem Definition of Video Segmentation

Video segmentation is the process of decomposing a video into its component units. There are two equivalent definitions of video segmentation:

- **Shot Boundary Detection**: Given a video V (equation 4.2) composed from n shots,

  $\forall S_i \in V$ locate $[t_b, t_e]$ the external points of the shot (Equation 4.6)

- **Edit Boundary Detection**: Given a video V (equation 4.2) composed from n shots,

  $\forall E_i \in V$ locate $[t_b, t_e]$ the external points of the edit (Equation 4.7)

The above two definitions are equivalent as edits and shots form a partition of the video. These two definitions are analogous to the *region growing versus edge detection* approaches to image segmentation.

The techniques presented in this chapter treat video segmentation as **edit boundary detection**. The reason is the relative simplicity of edit effect models as compared to shot models. Edits are simple effects that are artificially introduced using an editing suite to compose a video. Shots on the other hand incorporate all the factors that affect the static image formation process and the changes of these factors over time. This makes shots much harder to model analytically as compared to edits.

Video Segmentation using Production model based classification

Video segmentation is formulated as a production model based classification problem. In production model based classification the essential aspect is the existence of a computational model of the data production process. This data production model is used in designing feature extractors which are used in the automatic analysis of the data which is being classified. The use of the data production model distinguishes production model based classification from the traditional feature based classification. The *video edit model* (equation 4.2, 4.3, 4.4) and the *edit effect models* (equation 4.5) are the *data production models* in video segmentation.

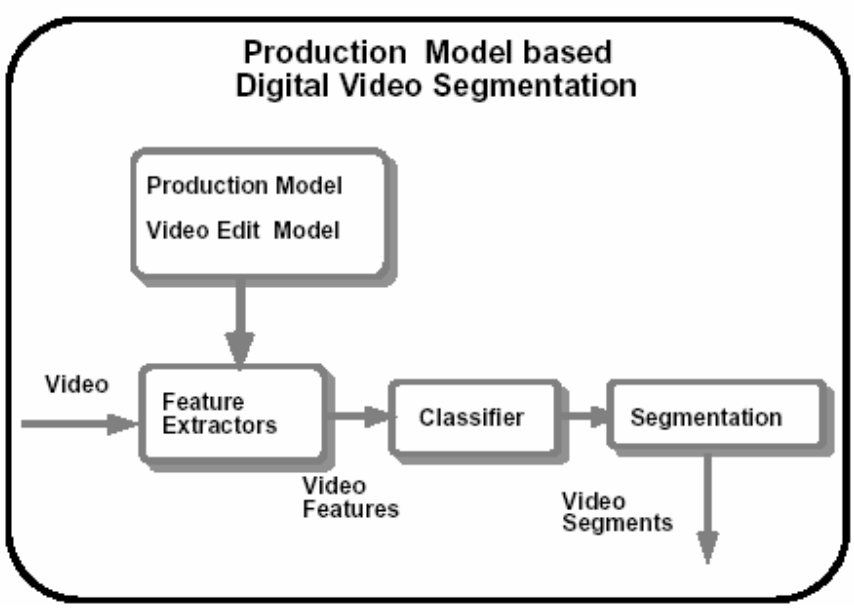


Figure 4.0: Block Diagram of Production Model based Video Segmentation

The above figure shows the stages in model based classification. The design phase of the problem has the following steps:

- **Production Model Formulation**: This step involves the study of the data production process (in this case video and film editing) to develop a model of the process. The first step is to isolate the essential steps used in data production. These steps are then translated into computational models.
- **Feature Extractor Design**: Here the production models developed in the previous step are used to systematically design features extractors which can be applied to the data in the automatic analysis phase.

The analysis or online portion of the production model based classification approach is the feature based classification process where the feature extractors previous designed are used. The steps in feature based classification are:

- **Feature Extraction**: The data to be classified is processed by the feature extractors to generate features which are indicators of the data classes of interest.

- **Classification**: The different features extracted in the previous step are combined using a discriminant function to assign a class label to the data set being classified.

In the current formulation of video segmentation as production model based classification, the four edit classes (table 4.0) are the classes of interest. Feature extractors are designed for each of these classes. The feature extractors are applied to the individual frames of the video. The features are combined using a discriminant function to assign to each frame of video the label *edit* or *shot*. The segmentation block essentially groups the frames into segments based on the labels assigned. A finite state machine is used to achieve this segmentation step.

Cut Detection

A cut is an identity edit and unlike other edits cannot be modeled or defined independently of the two shots it concatenates, since it does not contribute any edit frames to the video. Cuts can be categorized in terms of the shots that they concatenate. When a video with a cut is presented to a viewer, the viewer will experience a sudden transition (or discontinuity) of visual properties across the transition. Visual properties of a shot include factors like speed and direction of motion of objects and camera, shapes, color, brightness, distribution, etc. During the editing phase of video production, the director controls various visual property transitions across a cut. Some directors try to minimize the visual discontinuity experienced by the viewers across a cut. This criterion is termed as a *graphic match* in editing literature; others maximize the visual discontinuity across the cut to evoke a specific viewer reaction.

A cut detector is an algorithm which can detect the discontinuity of a certain visual property across two consecutive frames in a video. Most of the cut detectors used in literature rely on the color space of the frames to identify a discontinuity. The techniques also have an implicit shot model in terms of the expected variation of the visual property within the bounds of the shot. The performance of these detectors is fairly acceptable and accuracies in the range of 90% to 95% have been reported. There are two ways of achieving better cut detection, use *additional visual properties* and *use explicit models of shots*. There are several techniques which can be used to identify discontinuity of feature tracks in image sequences and other such properties. The problem of developing general models of video shots is an extremely difficult problem due to the number of factors that affect the nature of the video data. However different aspect of video can be individually modeled and used in developing tuned cut detectors. Cut

detectors presented in this chapter are based on results presented by other researchers in the field.

Name	Measurement Formula	Description
Template Matching	$\sum_{x,y} \lvert S(x,y,t) - S(x,y,t+1) \rvert$	Intensity Difference
Histogram c^2	$\sum_{i=0}^{G} \frac{(H_t(i) - H_{t+1}(i))^2}{H_{t+1}(i)}$	c^2 Histogram Comparison

Table 4.1: Cut features

Table 4.1 shows two features for cut detection that have been proposed in literature. Consider a shot S. Let the pixel size of the frames be $N_x \times N_y$. Let G be the number of gray levels. Let t denote the time or the frame number. Let the shot length be L. Let x, y denote pixel location within a frame. Let H_t be the histogram of the t^{th} frame of the shot. These features were extracted from videos with cuts between different types of shots (object motion shots, camera motion shots, etc).

Chromatic Edit Detector

Chromatic editing manipulates the color space of the two shots being edited (table 4.0). The goal of the chromatic edit detector is to discriminate between intensity changes due to chromatic editing as opposed to intensity changes due to scene activity. The intensity changes in the image sequence resulting from chromatic editing have a particular pattern which is modeled analytically by the edit effect model. The chromatic edit detector analyses the video data and detects the presence of the data patterns conforming to the edit effect model. **Fades** and **Dissolves** are the two most prevalent types of chromatic edits. These edit effects are used as the focus of the rest of the paragraph.

Chromatic Scaling

Fades and dissolves can be modeled as chromatic scaling operations. During a fade one of the shots being edited is a constant (normally black). A dissolve typically involves the scaling of two shots that are being edited. Thus a detector which can detect chromatic scaling in a video can be used to detect both fades and dissolves. Following it is

presented the model for a chromatic scaling of an image sequence, and derives a feature for detecting such an effect in a video. Once the scaling detector is designed, the use of that detector for detecting fades and dissolves is discussed.

Consider a gray scale sequence $g(x,y,t)$. Let the color space of the sequence be scaled out to black over the length of l_s frames. Then the model $E(x,y,t)$ for the output video of the scaling operation is:

$$E(x,y,t) = g(x,y,t) \times \left(1 - \frac{t}{l_s}\right) \quad \text{(Equation 4.8)}$$

During a single frame scaling, only the last frame of the shot is used in the scaling operation, i.e. the last frame of the shot is frozen and the intensity is scaled to zero (or the intensity is scaled from zero in the case of a fade in). Thus during a single frame scaling there is no motion within the sequence. In this case equation 4.8 can be written as 4.9 where k indicates that the k^{th} frame of the shot is being used in the scaling.

$$E(x,y,t) = g(x,y,k) \times \left(\frac{l_s - t}{l_s}\right) \quad \text{(Equation 4.9)}$$

Differentiating $E(t)$ with respect to time:

$$\frac{dE}{dt} = \frac{-g(x,y,k)}{l_s} \quad \text{(Equation 4.10)}$$

Equation 4.10 can be rewritten as

$$CI(t) = \frac{dE/dt}{g(x,y,k)} = \frac{-1}{l_s} \quad \text{(Equation 4.11)}$$

where CI *chromatic image*, is the scaled first order difference image. This image is a constant image with the constant value being proportional to the fade rate. A simple function based on the distribution of intensities can be designed to provide a measure of the constancy of an image. Let $F_{CS}(t)$ be the chromatic scaling feature which is a measure of constancy of CI.

$$F_{CS}(t) = U(CI(t)) \quad \text{(Equation 4.12)}$$

For multi-frame scaling, the scaling is done over a sequence of frames. Thus there is scene action in progress during the scaling edit. Here the differences in the pixel values during a scale will be due to a combination of the motion generated intensity difference and the scale generated intensity difference. The chromatic scaling feature, F_{CS}, is applicable if the change due to the editing dominates the change due to motion.

Fades & Dissolves as Chromatic Scaling

The fade and dissolve operations can be represented as some combination of chromatic scaling operations.

Fade Detection: Two types of fades are commonly used in commercial video production, fade in from black and fade out to black. In both these cases the fades can be modeled as chromatic scaling operations with a positive and negative fade rate. E_{fo} in equation 4.13 represents the sequence of images generated by fading out a video g_1 to black. l_1 is the fade out rate in terms of the number of frames. $\vec{0}$ represents the black image sequence. Comparing equations 4.5 and 4.13, for a fade out one of the shots $S_{out} = g_1$, $S_{in} = \vec{0}$ and $\otimes = +$. Similarly, E_{fi} (equation 4.14) represents the images generated by fading in a sequence g_2, at the rate of l_2. Here $S_{out} = \vec{0}$ and $S_{in} = g_2$. The equations 4.13, 4.14 represent how the fade operation maps on to the **edit effect model** (equation 4.5). Since the operations on the individual sequences in the fade are chromatic scaling operations, the chromatic scaling feature 4.11 can be used for detecting fades in videos.

$$E_{fo}(x,y,t) = g_1(x,y)\left(\frac{l_1 - t}{l_1}\right) + \vec{0} \quad \text{(Equation 4.13)}$$

$$E_{fi}(x,y,t) = \vec{0} + g_2(x,y)\left(\frac{t}{l_2}\right) \quad \text{(Equation 4.14)}$$

Dissolve Detection: A dissolve is a chromatic scaling of two shots simultaneously. Let E_d be the set of edit frames generated by dissolving two shots $S_{out} = g_1$ and $S_{in} = g_2$. Equation 4.15 models the process of dissolving two shots.

$$E_d(x,y,t) = g_1(x,y)\left(\frac{l_1 - t}{l_1}\right)_{(t_1, t_1 + l_1)} + g_2(x,y)\left(\frac{t}{l_2}\right)_{(t_2, t_2 + l_2)} \quad \text{(Equation 4.15)}$$

Here (t_1, t_2) are the times at which the scaling of g_1, g_2 begin and (l_1, l_2) are the duration for which the scaling of g_1, g_2 lasts. The relative values of these parameters can be used to group dissolves into different classes. Such a grouping can be used to analyze the effectiveness of the detection approach. Comparing equations 4.14, 4.13 and equation 4.15 it can be seen that the dissolve is a combination of the fade in and fade out operation occurring simultaneously on two different shots.

A dissolve is a multiple sequence chromatic scaling. Designing the dissolve detector can now be treated as a problem of verifying that the chromatic scaling detector (equation 4.12) can be used to detect the dissolve. The approach followed in this work is to classify the dissolves into groups based on their qualitative properties and to verify the detect ability of each group using the chromatic scaling operator.

Figure 4.1 presents the *sequence activity graph* (SAG) during a dissolve edit. It shows the *qualitative labeling* of dissolves. The hatched area indicates the *Out Shot* activity and the filled area the *In Shot*. A positive slope in the SAG indicates a fade in operation and the negative slope indicates a fade out operation. A zero slope in the SAG indicates no sequence activity due to editing.

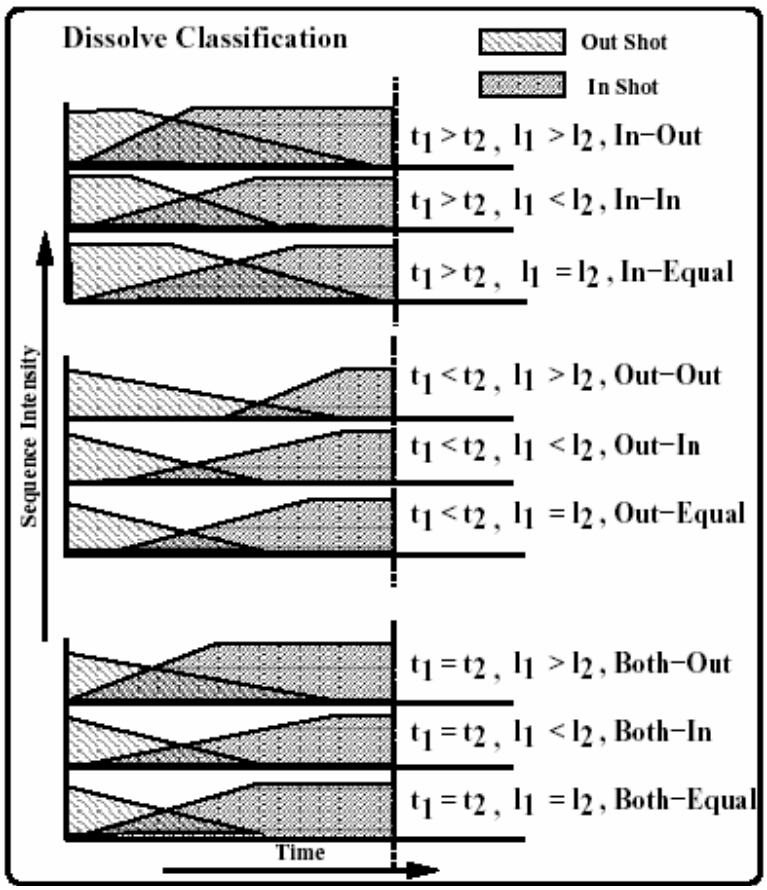


Figure 4.1: Sequence Activity Graph during a dissolve

The basis used for the qualitative labeling is the start time and the dissolve length. The labels are based on **Shot of Initial Activity**-**Dominating Shot** attributes of the sequence which are defined as follows:

Shot of Initial Activity: This is defined as shot s, where (t_1, t_2) are the times at which **Out, In** shots begin scaling.

s = **In** if $t_1 > t_2 \Rightarrow$ Fade In **begins before** Fade Out. (Equation 4.16)
s = **Out** if $t_1 < t_2 \Rightarrow$ Fade Out **begins before** Fade In. (Equation 4.17)
s = **Both** if $t_1 = t_2 \Rightarrow$ Fade In, Fade Out **begins together.** (Equation 4.18)

Dominating Shot: This is defined as shot s, where (l_1, l_2) are the dissolve lengths of the **Out, In** shots begin respectively.

s = **In** if $l_1 > l_2 \Rightarrow$ In Shot **dominates dissolve**. (Equation 4.19)

s = **Out** if $l_1 < l_2 \Rightarrow$ Out Shot **dominates dissolve**. (Equation 4.20)

s = **Equal** if $l_1 = l_2 \Rightarrow$ No Shot **dominates dissolve.** (Equation 4.21)

A shot is said to dominate the dissolve if its activity slope is higher. In other words, if the shot contributes more to the *inter frame change* in the video sequence.

Observing figure 4.1 two things can be reported:

1. Except in the case of **Both-Equal** type of dissolves, all the other types have portions during which there is an exclusively single sequence chromatic scaling in progress.
2. Except in the case of **Equal Dominance Sequences** the change contribution of one sequence dominates the other.

Thus the cases in which the chromatic scaling feature (equation 4.12) will not respond to the dissolve are those in which very similar sequences are being dissolved with precisely equal fade rates over the dissolve. In most commercially produced video sequences dissolves are seldom executed with such precision and dissolving very similar sequences are avoided. Hence the chromatic scaling feature can be used to detect most dissolves.

Limitations of Chromatic Edit Detector

There are several limitations of the chromatic edit detector which follows below:

- **Additional Chromatic Transforms**: The chromatic edit detector presented can be used for detecting chromatic edits which are scaling of the color space. There are other possible types of chromatic transforms like chromatic translations, rotations etc. However this is not a serious practical limitation as chromatic transformations other than scaling are seldom used in practice.
- **Multiple Sequence Scaling**: The scaling detector is primarily designed to detect the scaling of single image sequences. It can be used to detect two sequences scaling as in the case of dissolves provided that the effect of *sequence domination* is present. If there are segments of video with dissolves of more than two sequences the chromatic scaling detector cannot be guaranteed to respond.
- **Piecewise Transformations**: The detectors are designed to respond to global image transforms. The detector responses become

unpredictable if the chromatic transforms are applied to spatial sub windows in the sequence.

- **Transformation Rates**: This is the inverse of the duration for which a particular transformation is applied to create an edit. When the transformation rate is very high the *extended edit* approaches the *cut or identity*. In this case the change will be large enough for the cut detector to respond. In the case of extremely small transformation rates (dissolves and fades over 100's of frames), the change due to the effect between frames may be too small. In such cases detecting these effects based on two frames of video will not be possible and approaches involving extended set of frames will have to be adopted.

Spatial Edit Detector

Spatial edits are achieved by transforming the pixel space of the shots being edited. These are **Type 2** transforms in table 4.0 where $T_C = \vec{I}$ and T_S has different values depending on the exact nature of the spatial effect used, takes on different values depending on the specific type of spatial edit. One of the most commonly used edits is the page translate, where the shot preceding the edit is translated out of the view port uncovering the shot that follows the edit. This type of edit is used as an exemplifier of the class of spatial edits and a feature derivation is presented.

Consider a video $E(x,y,t)$ with a translate spatial edit. In such an edit the first shot translates out, uncovering the second shot. Let $E(x,y,t)$ be a gray scale sequence for notational simplicity. Let $g_{out}(x,y,t), g_{in}(x,y,t)$ be the gray scale models of the incoming and outgoing shots in the edit. Let a_x, a_y be the translation coefficients in the x and y directions respectively. Let N_x, N_y be the number of pixels in the x and y dimensions of each frame. The translate edit can now be modeled as:

$$E(x,y,t) = \begin{cases} g_{out}(x + a_x t, y + a_y t, t) & \text{if } 0 \le x + a_x \le N_x \text{ and } 0 \le y + a_y \le N_y \\ g_{in}(x,y,t) & \text{else} \end{cases}$$

(Equation 4.22)

Equation 4.22 represents the process where a pixel in the final cut is taken either from g_{out} if it lies within the bounds of the frame, or from g_{in}. In the case of a pure spatial edit the brightness of a particular point does not change over time, the change in the video is caused by the motion of the

point due to the edit. This fact can be used to write down the constant brightness equation for the edit:

$$\frac{dE}{dt} = 0 \quad \text{(Equation 4.23)}$$

Using the chain rule for differentiation equation 4.23 can be rewritten as equation 4.24:

$$\frac{dE}{dt} = \frac{dE}{dx} \cdot \frac{dx}{dt} + \frac{dE}{dy} \cdot \frac{dy}{dt} + \frac{dE}{dt} = 0 \quad \text{(Equation 4.24)}$$

substituting for E from equation 4.22 in equation 4.24 and assuming that the motion in the incoming shot is negligible as compared to the motion due to the edit $\frac{dg_{in}}{dt} = 0$ the following equation can be written:

$$\frac{dE}{dt} = \frac{dE}{dx} \cdot \frac{d\left(x + a_x \cdot t\right)}{dt} + \frac{dE}{dy} \cdot \frac{d\left(y + a_y \cdot t\right)}{dt} + \frac{dE}{dt} = 0 \quad \text{(Equation 4.25)}$$

which can be rewritten as

$$\frac{dE}{dt} = \frac{dE}{dx}\left(\frac{dx}{dt} + a_x\right)\frac{dE}{dy}\left(\frac{dy}{dt} + a_y\right) + \frac{dE}{dt} = 0 \quad \text{(Equation 4.26)}$$

Assuming that there is no scene action in progress during the edit (i.e. the first shot freezes before the translation begins) there will be no relative changes in the image due to scene motion. Hence $\frac{dx}{dt} = \frac{dy}{dt} = 0$. Therefore equation 4.26 can be rewritten as follows:

$$\frac{dE}{dx} \cdot a_x + \frac{dE}{dy} \cdot a_y + \frac{dE}{dt} = 0 \quad \text{(Equation 4.27)}$$

For the case of pure translation in the x direction $a_y = 0$. Hence

$$SI(t) = a_x = \frac{-\frac{dE}{dt}}{\frac{dE}{dx}} \quad \text{(Equation 4.28)}$$

Equation 4.28 shows that in the case of the edit being a pure translation in the x direction, the scaling of the difference image by the X gradient image results in a constant image SI, the *spatial image*. Let F_{sgx} represent the spatial translate feature:

$$F_{sgx} = U(SI(t)) \quad \text{(Equation 4.29)}$$

where U denotes the measure of uniformity of the scaled difference images. Thus the feature F_{sgx} can be used as an indicator of the spatial translate in x. Similar features can be designed for detecting spatial translates edits in different directions. This is feasible given that the cost of computing each feature is limited and the number of directions in a quantized image is small. Many other types of spatial transforms like the page turn and several other digital editing effects can be modeled as piece wise transforms applied to image windows. A similar process can be used to design detectors for these various types of edits. However as the edits effects become more complex with significant local effects the design of effective detectors becomes more difficult.

A Measure of Image Uniformity

The output of the image manipulation operations both in the case of the *chromatic edit detector* equation 4.12 and the spatial edit detector equation 4.29 are gray scale images with a constant value. In the case of real images this will seldom be the case□a constant image will have a uni-modal histogram. The following is a measure which responds to images with a uni-modal histogram. Let $I(x,y,t), I(x,y,t+1)$ be the two consecutive frames in the video for which the features are being computed. Let $x \in (1, N_x)$ and $y \in (1, N_y)$ where $N = N_x * N_y$ is the total number of pixels in the image. Let XI represent the image whose constancy is being measured, where $XI = CI$ for the chromatic scaling detector and SI for the spatial operator. There are two aspects that can be measured from XI.

- **Spatial Uniformity**: For the ideal case where $XI = K\forall_{x,y}$ all the pixels in the image will have the same value. In the case of a real image the $XI(x,y)$ is valid only if the difference pixel at that point is non zero, because a constant set of frames in the video will yield $XI = \vec{0}$. Hence the uniformity measure is directly weighted be the number of non zero difference pixels or the *Area of the non zero difference*

image. Also a computationally cheap measure of the spatial uniformity is the location of the centroid of the valid pixels of XI .In the ideal case the centroid should be located at the physical center of the image. Thus the uniformity measure can be inversely weighted by the distance centroid from the physical image center.

- **Value Uniformity**: For the ideal case where $XI = K\forall_{x,y}$ the variance of XI will be zero. Hence inversely weighting the uniformity measure by the variance guarantees that the measure will have a strong positive response for a constant image.

Based on the above consideration U shown in equation 4.30 is the measure of image constancy. The meaning of the different symbols and their computational formulas are presented in table 4.2:

$$\text{Uniformity Measure} = U = \frac{A(T(D(I(t)),0)}{s\,(V_{cx}(I(t))) \cdot (1.0 + C(T(D(I(t),0))) - (c_x, c_y))}$$

(Equation 4.30)

Computational Requirements of Feature Detectors

This paragraph presents the computational requirement analysis of the three feature detectors used. Let N be the number of pixels per frame in the digital video sequence being segmented. Let $k_{add}, k_{sub}, k_{div}, k_{mult}, k_{abs}, k_{comp}$ be the cost of performing one operation of *addition, subtraction, division, multiplication and absolute value, comparison* respectively. Table 4.2 lists the costs of various operations in the feature extraction process and estimates the complexity of the computation. Let Λ represents the cost of a compound operation in terms of the basic image computations.

Operation	Symbol	Computation Formula	Cost
Histogram	$H(I(t))$	$\forall_{x,y} H(I(x,y,t)) + +$	$N * (k_{add})$
Difference	$D(I(t))$	$\forall_{x,y} I(x,y,t+1) - I(x,y,t)$	$N * (k_{sub} + k_{abs})$
Division	$S(I(t))$	$\forall_{x,y} I(x,y,t+1) \div I(x,y,t)$	$N * (k_{div})$
Gradient	$\frac{d}{d_x}(I(t))$	$\forall_{x,y} I(x+1,y,t) - I(x,y,t)$	$N * (k_{sub})$
Threshold	$T(D(t),T)$	$\forall_{x,y} B(x,y,t) = 1$ if $I(x,y,t) \geq T$	$N * k_{comp}$
Mean	$m(I(t))$	$\frac{\sum_{x,y} I(x,y,t)}{N}$	$N * k_{add} + k_{div}$
Variance	$s(I(t))$	$\sum_{x,y} (I(x,y,t) - m(I(t)))^2$	$N * (k_{mult} + k_{sub})$
Area	$A(B(t))$	$\sum_{x,y} B(x,y,t)$	$N * k_{sum}$
Centroid	$C(B(t))$	$C_x = \frac{\sum_{x,y} xI_d(x,y)}{A}$ $C_y = \frac{\sum_{x,y} yI_d(x,y)}{A}$	$N * (2 * k_{mult}) + k_{div})$
Chromatic Img	CI	Equation 4.11	$\Lambda_{diff} + \Lambda_{div}$
Spatial Img	SI	Equation 4.28	$\Lambda_{diff} + \Lambda_{grad} + \Lambda_{div}$
Uniformity	$U(XI, D)$	Equation 4.30	$\Lambda_{diff} + \Lambda_{thresh} + \Lambda_{area} + \Lambda_{\text{var}} + \Lambda_{centroid} + \Lambda_{thresh} + k_{div} + k_{add}$
Cut	F_{cut}	$\sum_{i=0}^{G} \frac{(H_t(i) - H_{t+1}(i))^2}{H_{t+1}(i)}$	$3 * N * k_{add} + G * (k_{add} + k_{div} + k_{mult})$
Chromatic	F_{chrom}	$U(CI)$	$\Lambda_{uniformity} + \Lambda_{CI}$
Spatial	F_{space}	$U(SI)$	$\Lambda_{uniformity} + \Lambda_{SI}$

Table 4.2: Computational Cost Table

Feature Detector Summary

The above paragraphs presented a systematic approach to designing features. The feature design was based on the edit effects models. Figure 4.2 shows a flow diagram for extracting all the features from a pair of images $I(t)$ and $I(t+1)$. The *cut* feature involves the computation of a histogram for each image and a c^2 comparison of the histograms. The *chromatic* feature requires the computation of a difference image, an image division and an image constancy computation, while the *spatial* feature requires an image difference, image division and a constancy computation. The most important aspects of the feature detectors designed are:

- **Local Computation**: The extended edit effects like fades and dissolves are being extracted based on using just two consecutive frames in the sequence. This is a very efficient method of extracting extended edit effects.
- **Algorithm Simplicity**: The computations needed to accomplish the task of extracting extended edits are very simple and hence reliable.
- **Modularity**: The set of operations necessary to compute all the features is modular and the results of some computations can be used in more than one feature detector.

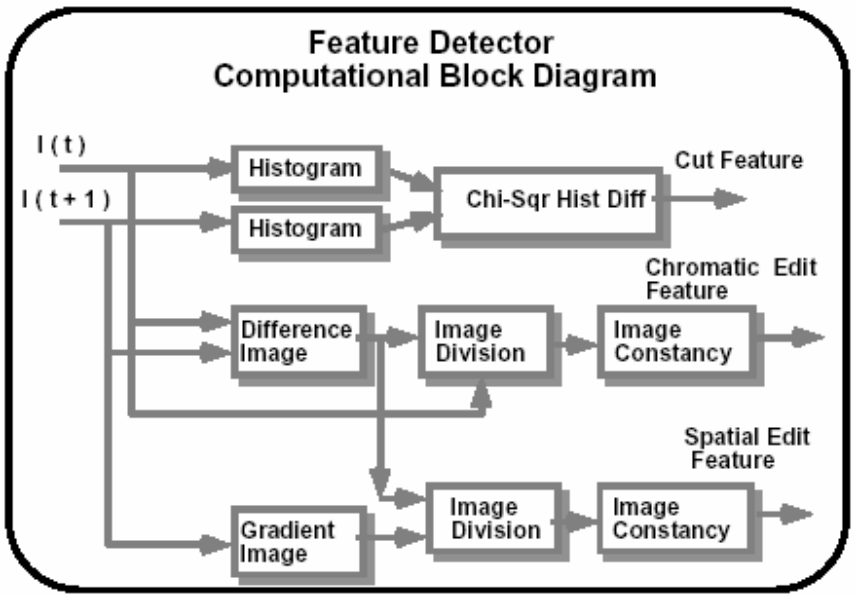


Figure 4.2: Feature Extractors: Computational Block Diagram

Classification & Segmentation

The features extracted from the video undergo several steps of processing before the video segments are identified. This paragraph presents the various steps involved. Figure 4.3 provides an illustration of the details involved in the classification and segmentation process. A modification of a standard two class discrete classifier has been used to achieve the segmentation process. The lack of prior probability density functions for the various features makes the use of standard Bayesian decision models unsuitable for this application.

- **Feature Thresholding**: The first step in the classification and segmentation process is feature thresholding. The response space of each of the features, namely cut, chromatic edit and spatial edit, are discretized into a number of regions. A single threshold is used to categorize the response as either positive or negative. The thresholds T_i for the different features are chosen based on the conditional probability distributions of the features over an experimental data set.
- **Discriminant Function**: The discriminant function is a function designed to combine the feature responses of the three features. The output of the discriminant function thus assigns to each frame in the video one of two labels **edit** or **shot**. The label assignment takes into account the correlation that exits between the features and the conditional distributions of the features. The discriminant function used in this system is $D(f_c, f_{cs}, f_{sp}) = f_c \wedge f_{cs} \wedge f_{sp}$ where $\wedge$ is the logical OR operator. Thus the output of the discriminant function is a two label pulse train that needs to be segmented.
- **Segmentation**: The two label pulse train (i.e. each frame is either called an *edit* or *shot*) is segmented by using a finite state machine show in Figure 4.4. The notation used is the standard notation of finite state machines. The circles indicate states and the arrows indicate the transition between states. The machine for segmentation has 3 states S_q, S_o and S_1 where S_q is the *quiescent state*, S_o is the *shot segment state*, S_1 is the *edit segment state*. The machine has two outputs namely *begin segment*, *end segment*.

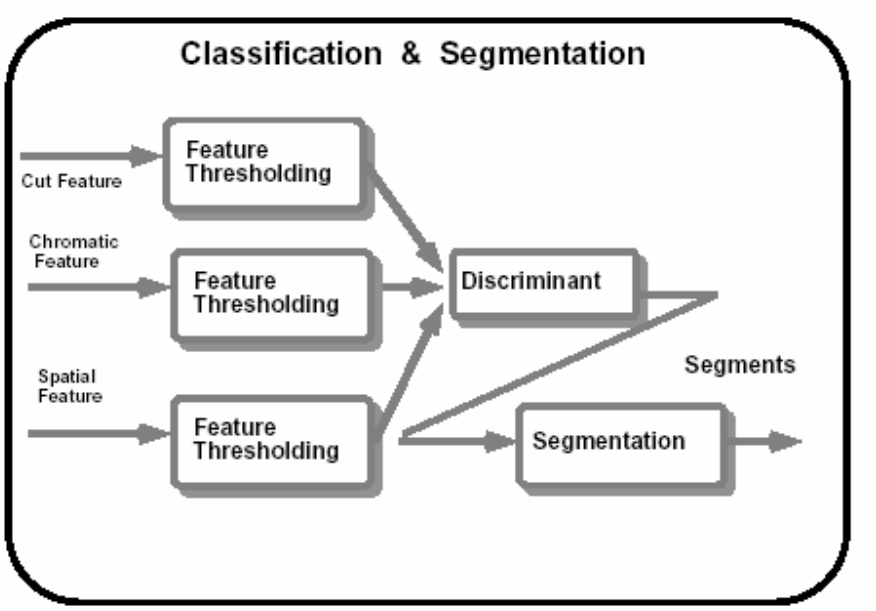


Figure 4.3: Steps in Classification & Segmentation

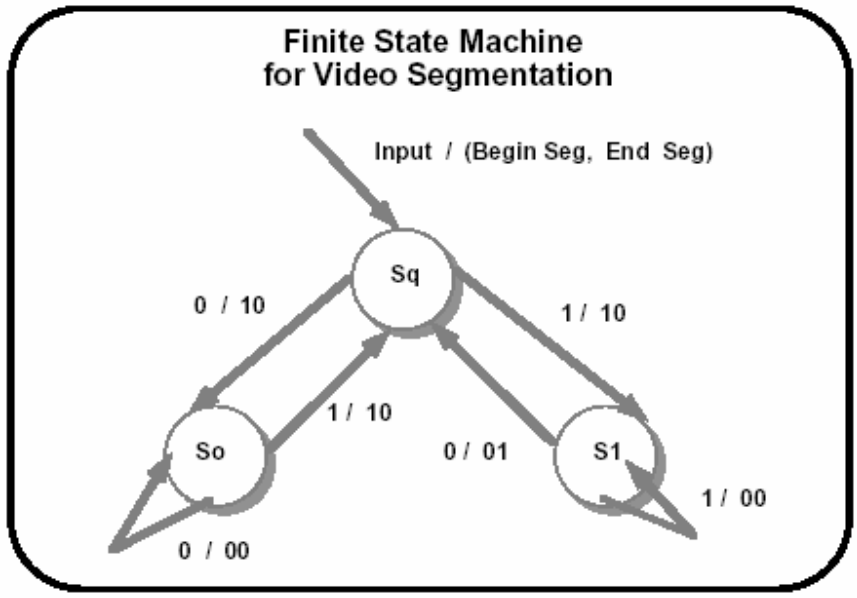


Figure 4.4: Finite State Machine of Segmentation

Error Measures for Video Segmentation

The video computational model (equation 4.4) is used for the purpose of measuring segmentation error. Shots and extended edits (fades, dissolves, spatial edits etc) are closed intervals with non zero length. A cut however is not an interval. But for the sake of consistency cuts can be treated as a closed interval of length zero. The error in segmenting a video is the difference between the correct model of the video V and the output of a segmentation algorithm V'. Let V have n segments. Let V' have k segments. Then

$$V = S_1 \circ S_2 \circ S_3 \circ S_4 \circ ... \circ S_{(n-1)} \circ S_n \quad \text{(Equation 4.31)}$$

$$V' = O_1 \circ O_2 \circ O_3 \circ ... \circ O_k \quad \text{(Equation 4.32)}$$

where S_i and O_i are segments in the correct video computational model and the output of the segmentation algorithm. The difference between two videos with reference to segmentation has the following two error components, the *Segment Boundary Error* due to the improper location of the segment boundaries, and the *Segment Classification Error* due to the improper labeling of the segments. Thus the overall segmentation error E can be defined as follows

$$E(V,V') = E_{sb}(V,V') \times W_{sb} + E_{sc}(V,V') \times W_{sc} \quad \text{(Equation 4.33)}$$

E_{sb} represents the segment boundary error and E_{sc} represents the segment classification error. The weights W_{sb} , W_{sc} allow the error measure to reflect the bias of the application, and can be set based on which error involves more cost to the user.

Segmentation Error Classes

Given a correct video model V (equation 4.31) with n segments and an output model V' (equation 4.32) with k, let n' and k' be the number of unassigned segments in V and V' after computing the correspondence (*Given 2 video computational models, the process of measuring errors involves comparing the individual segments in the 2 models. This requires a mapping between the individual segments of 2 models. The process of generating this mapping is called* **Segment Correspondence**). Then the segmentation can be classified as follows:

- **Under Segmentation** $k' < n'$: The number of unassigned segments in the output model are fewer than in the correct model. This implies that the segmentation algorithm has missed a number of boundaries and that the number of false positives in the edit detection is lesser than the number of missed edits.
- **Equal Segmentation** $k' = n'$: The number of segments unassigned in both the output and the correct model are the same. This implies that the number of false positives and missed edits are the same.
- **Over Segmentation** $k' > n'$: The number of unassigned segments is greater in the output model as compared to the correct model. This implies that the video has been broken up into more segments than necessary or that the number of false positives in the edit detection is greater than the number of missed edits.

The classification of a segmentation error provides a qualitative labeling of the error. In addition to this the error classes are used in the definition of the error measure. In most real videos the number of segments in the video tends to be much smaller than the number of frames.

Segment Boundary Errors: E_{sb}

Once the corresponding segments have been assigned between V and V' the boundary error can be computed as the absolute difference of the corresponding intervals scaled by the length of the video. An additional penalty is added for the unassigned segments from both the correct and output models.

$$E_{sb}(V,V') = \frac{\sum_{i=1}^{n} e_i(S_i, O_i)}{Length(V)} + \frac{n' + k'}{l} \quad \text{(Equation 4.34)}$$

where $l = n$ for under and equal segmentation and $l = Length(V)$ for over segmentation, e_i is the interval error between S_i and O_i. The error between two intervals $T_1 = [t_{1b}, t_{1e}], T_2 = [t_{2b}, t_{2e}]$ is defined as follows:

$$e_i(T_1, T_2) = |t_{1b} - t_{2b}| + |t_{1e} - t_{2e}| \quad \text{(Equation 4.35)}$$

Segment Classification Errors: E_{sc}

Given two corresponding segments s_1 and s_2 with labels l_{s1} and l_{s2} the segment classification error is defined as follows

$$e_{sc} = \begin{cases} 1 \; if \; l_{s1} \neq l_{s2} \\ 0 \; if \; l_{s1} = l_{s2} \end{cases} \quad \text{(Equation 4.36)}$$

The overall segment classification error for the entire video is given by

$$E_{sc}(V,V') = \frac{\sum_{i=1}^{n} e_{sc}(S_i, O_i)}{n} + \frac{n' + k'}{l} \quad \text{(Equation 4.37)}$$

where $l = n$ for under and equal segmentation and $l = Length(V)$ for over segmentation, e_{sc} is the classification error between S_i and O_i.

Chapter 5

Five Approaches for Uncompressed Video Segmentation

In chapter 3 several methods for segmentation of uncompressed video were presented. In this chapter three histogram based and two motion based approaches will be examined, for the segmentation of the uncompressed video, extensively.

The first histogram based approach is called *Twin Comparison* (TC) and was introduced by Zhang et al. TC can detect both camera cuts and gradual transitions with a double thresholding, using global thresholds. More details for TC will be discussed next on this chapter. The second histogram based method is similar to the first one. In the second method we use a local window based threshold (SW). This threshold is calculated like one of the thresholds in TC. The third histogram based method is an adaptive algorithm using adaptive mean and adaptive standard deviation to calculate the threshold. The last two methods will be broadly examined next on this chapter.

Both motion based methods use *motion vectors* to segment the uncompressed video. After the calculation of motion vectors, the histogram of their angle is produced. The histogram difference of motion vector's angle is used finally in these two methods. The first one is the same with the second of the histogram based methods with the difference that now we have angle histogram difference and not intensity histogram difference. The second is the same with the third of the histogram based methods but, as in the first motion based method, the histogram difference is for angle and not for intensity.

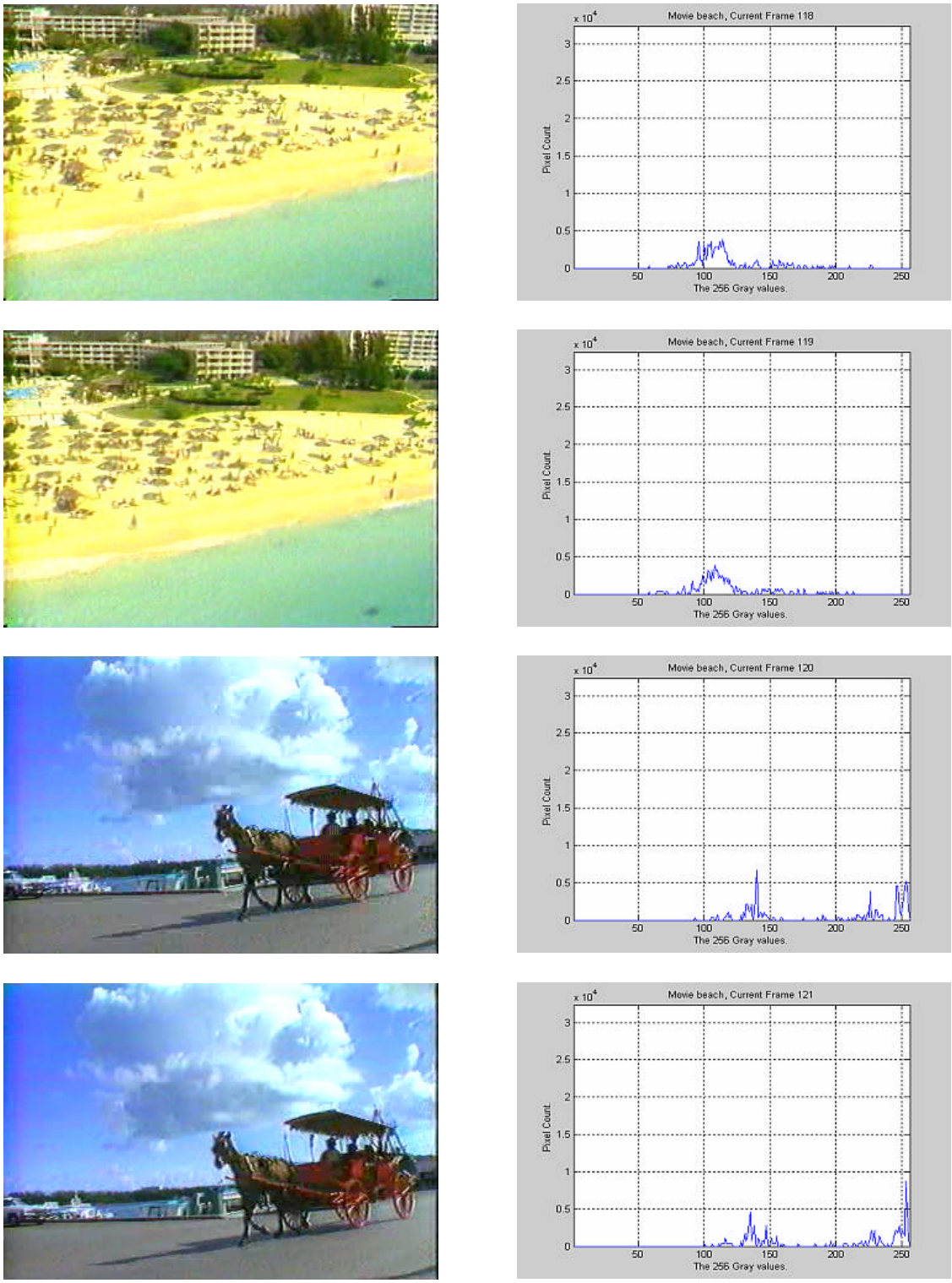
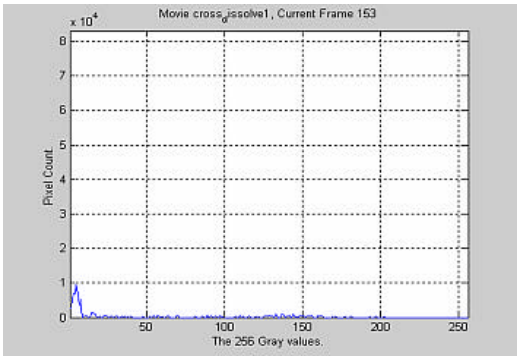
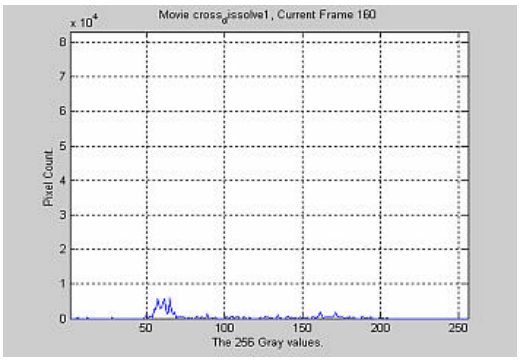
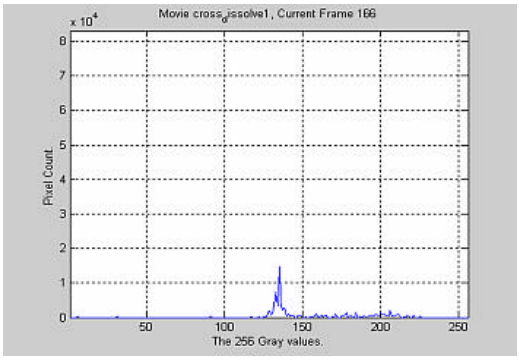
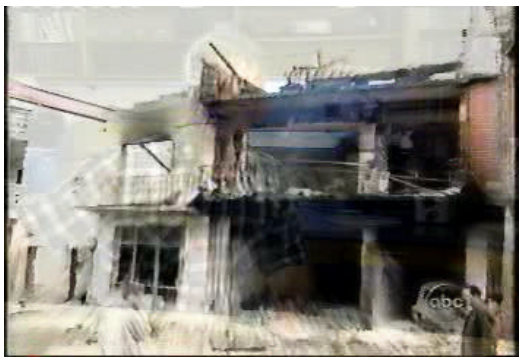


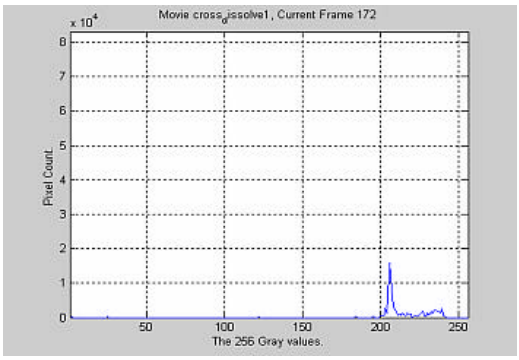
Figure 5.0: Four frames across a camera break from a documentary video with their intensity histograms. The first two frames are in the first camera shot. The third and fourth belong to the second camera shot. There are significant content changes between the second and the third frame. This change can be seen from their intensity histograms.


Movie cross_issolve1, Current Frame 153
x 10^4
Pixel Count
The 256 Gray values.


Movie cross_issolve1, Current Frame 160
x 10^4
Pixel Count
The 256 Gray values.


Movie cross_issolve1, Current Frame 166
x 10^4
Pixel Count
The 256 Gray values.




Movie cross_issolve1, Current Frame 172
x 10^4
Pixel Count
The 256 Gray values.

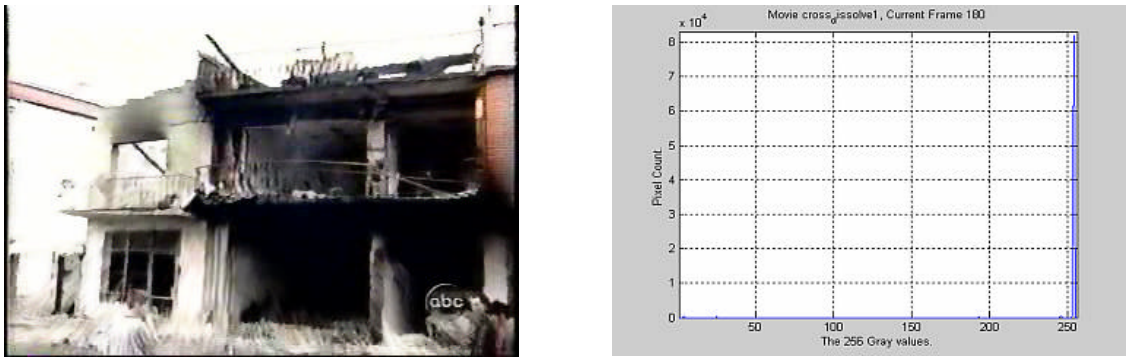


Figure 5.1: Five frames across a *cross dissolve*. The first frame is the one just before the dissolve starts, and the last one is the frame immediately after the end of the dissolve. The rest are the frames within the dissolve. In the right column intensity histograms for each frame can be seen.

Histogram Based Methods

Twin Comparison

Difference metric selection

The detection of transitions involves the quantification of the difference between two image frames in a video sequence. To achieve this, we need first to define a suitable metric, so that a segment boundary can be declared whenever that metric exceeds a given threshold. As mentioned in chapter 3 difference measures used to partition video can be divided into two major types: the *pair-wise comparison* of pixels or blocks, and the comparison of the histograms of pixel values (intensity histograms).

Difference Measure Name	Difference Measure Formula	Advantages	Disadvantages
Pair-wise Comparison	$DP_i(k,l) = \begin{cases} 1 \text{ if } \lvert P_i(k,l) - P_{i+1}(k,l) \rvert > t \\ 0 \quad \text{otherwise} \end{cases}$ $\frac{\sum_{k,l=1}^{M,N} DP_i(k,l)}{M * N} * 100 > T$	Simple implementation, easy understanding	Sensitive to camera movement
Likelihood Ratio	$\frac{\left[\frac{S_i * S_{i+1}}{2} + \left(\frac{m_i - m_{i+1}}{2} \right)^2 \right]^2}{S_i * S_{i+1}} > t$	Raise of the level of tolerance to slow & small object motion from frame to frame.	If 2 sample areas to be compared have the same mean & variance, but different probability density function, no change will be detected.
Histogram Comparison	$SD_i = \sum_{j=1}^{G} \lvert H_i(j) - H_{i+1}(j) \rvert$	Less sensitive to object motion, since it ignores spatial changes.	Two images have similar histograms but different content.

Table 5.0: Difference metrics that are discussed in Zhang et al paper. In this table also appears the mathematic formula that is used to implement metrics and the most important advantages/disadvantages.

Notation & Explanation of *Pair-wise comparison*: In grayscale images, a pixel is judged as changed if the difference between its intensity values in the two frames exceeds a given threshold t. This metric can be represented as a binary function $DP_i(k,l)$ over the domain of 2-D coordinates of pixels, (k,l) where the subscribe i denotes the index of the frame being compared with its successor. $P_i(k,l)$ denotes the intensity value of the pixel at coordinates (k,l) in frame i. The pair-wise segmentation algorithm simply counts the number of pixels changed from one frame to the next. A segment boundary is declared if more than a given percentage of the total number of pixels, given as a threshold T, have changed. The total number of pixels in a frame of dimensions M by N is $M * N$.

Notation & Explanation of *Likelihood Ratio*: To make the detection of camera breaks more robust, instead of comparing individual pixels, we can compare *corresponding regions* (*blocks*) in two successive frames on the basis of second order statistical characteristics of their intensity values. Let m_i and m_{i+1} denote the mean intensity values for a given region in two consecutive frames, and let S_i and S_{i+1} denote the corresponding variances. Camera breaks can now be detected by first partitioning the frame into a set of sample areas. Then a camera break can be declared whenever the total number of sample areas whose likelihood ratio exceeds the threshold t is sufficient large.

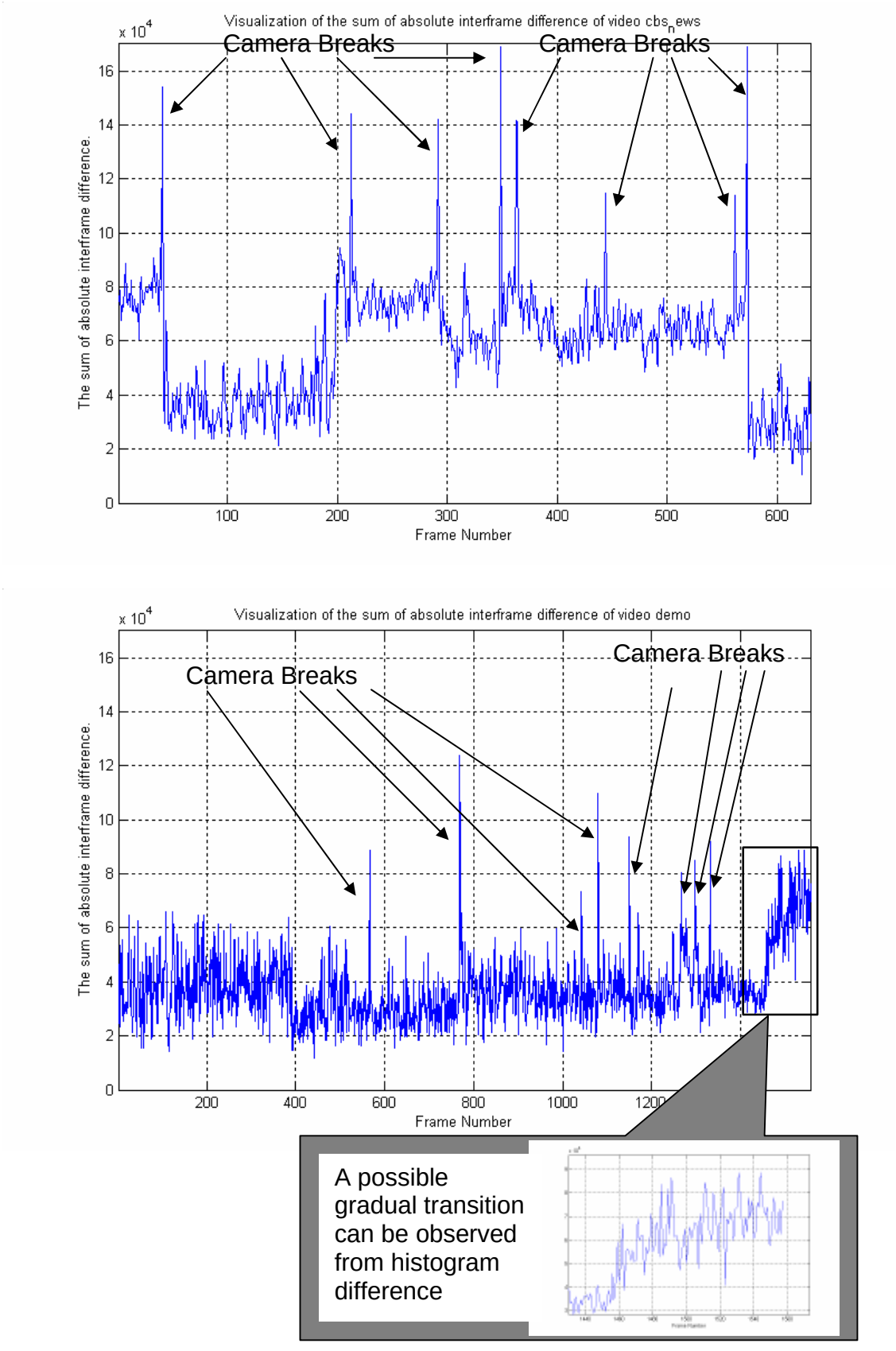
Notation & Explanation of *Histogram Comparison*: An alternative approach is to compare some feature of the entire image. One such feature that can be used in segmentation algorithms is a histogram of intensity levels. Two frames having an unchanged background and unchanged objects will show little difference in their respective histograms. Let $H_i(j)$ denote the histogram value for the i-th frame, where j is one of the G possible grey levels (NOTE: The number of histogram bins can be chosen on the basis of the available grey-level resolution and the desired computation time). If the overall difference SD_i is larger than a given threshold T, a segment boundary is declared. This metric was chosen to implement because of the advantage that is less sensitive to object motion and because of the low probability of the fact that there may be cases in which two images have similar histograms but completely different content.

Figures 5.0 and 5.1 shows in the right column grey-level histograms of the frames from the left column. In figure 5.0 it is obvious the difference between the histograms across the camera break between the second

and the third frame, while the histograms of the first and second frame are almost identical. The histograms of the third and fourth frame are also identical. Figure 5.2 illustrates the application of histogram comparison to three news videos. The three plots display the sequence of SD_i values, defined in table 5.0, between every two consecutive frames over the entire video. The formula was applied to grey-level intensities computed from the intensities of the three color channels Red, Green and Blue by the conversion formula which is displayed below:

$$GREY = 0.211 * R + 0.715 * G + 0.074 * B$$

In this formula R, G and B stand for intensities of the red, green and blue, respectively. This formula also gives the impact of individual color changes on the overall grey level and indicates the significance of the green component. The high pulses in the plots correspond to camera breaks.


Visualization of the sum of absolute interframe difference of video cbs_news
x 10^4
Camera Breaks
Camera Breaks
The sum of absolute interframe difference.
Frame Number
Visualization of the sum of absolute interframe difference of video demo
Camera Breaks
Camera Breaks
A possible gradual transition can be observed from histogram difference

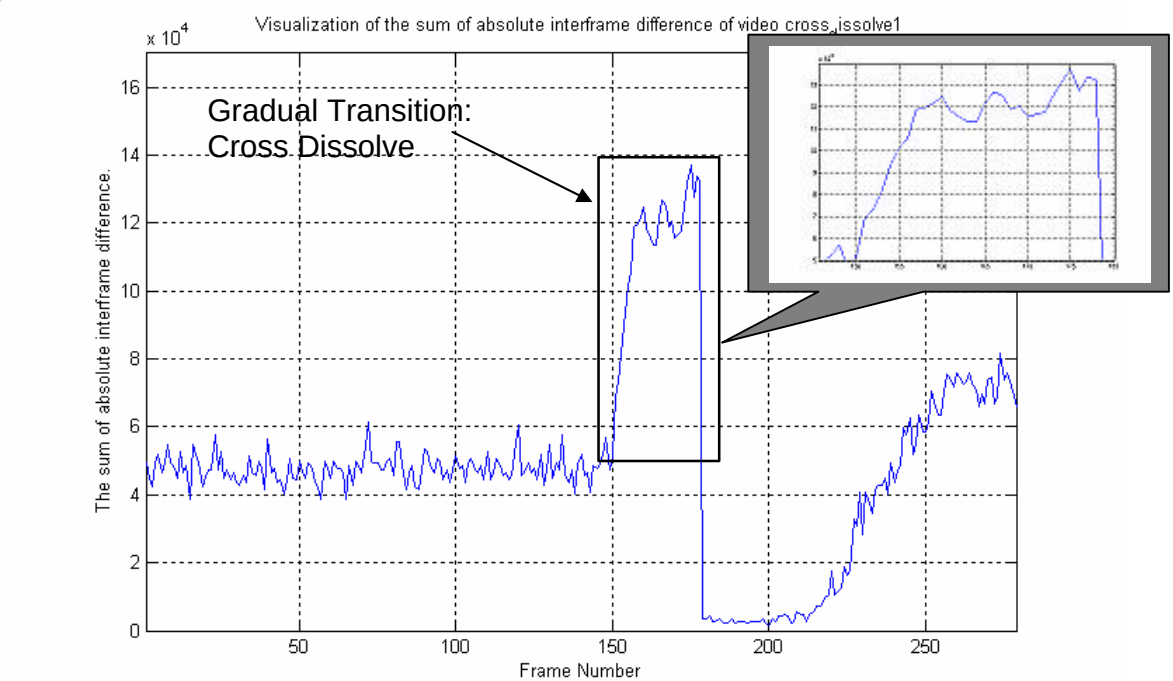


Figure 5.2: Three histogram difference plots from different video files. Camera breaks and gradual transitions can be observed

In addition to the weaknesses of each of the individual metrics that have been cited in table 5.0, all three types of difference metric face a severe problem if there are moving objects, of either large size or of high speed, or a sharp illumination change between two frames in a common shot, resulting in a false detection of a camera break. Flashing lights and flickering objects (such as video & computer monitors) are common sources of errors.

Twin-Comparison approach for shot detection

The TC approach adapts a difference metric to accommodate gradual transitions. The histogram comparison difference metric is used in the TC approach. As one can see from figure 5.2, there are three plots of the frame-to-frame histogram differences. In the first plot there are eight high pulses that correspond to camera breaks. There are also eight high pulses in the second plot. It is easy to select a suitable cutoff threshold value (such as 100000 and 70000 for first and second plot respectively) for detecting these camera breaks. However, the inset of both the first and second plot displays sequences of pulses the values of which are higher

than those of their neighbors but are significantly lower than the cutoff thresholds.

The simplest approach to this problem would be to lower the thresholds. Unfortunately, lower thresholds cannot be effectively employed, because the difference values that occur during the gradual transition implemented by a special effect may be smaller than those that occur during between the frames within a camera shot. For example, object motion, camera panning and zooming also entail changes in the computed difference value. If the cutoff threshold is too low, such changes may easily be registered as "false positives". The problem is that a single threshold value is being made to account for all segment boundaries, regardless of context.

In figure 5.1 it is obvious that the first and last frames are different, even if all consecutive frames are very similar in content. The problem is to detect these first and last frames. If they can be determined, then each of them may be interpreted as a segment boundary and the period of gradual transition can be isolated as a segment unto itself. The inset of the third plot in figure 5.2 illustrates that the difference values between most of the frames during the dissolve are higher, although only slightly, than those in the preceding and following segments. What is required is a threshold value that will detect a dissolve sequence and distinguish it from an ordinary camera shot. A similar approach can be applied to transitions implemented by other types of special effects.

TC requires the use of two cutoff thresholds: T_b is used for camera break detection and T_s is used for special effect detection (such as cross dissolve). The detection process begins by comparing consecutive frames using *histogram comparison* metric. Whenever the difference value exceeds threshold T_b, a camera break is declared, for example F_B in figure 5.3. However, the TC also detects differences that are smaller than T_b but larger than T_s. Any frame that exhibits such a difference value is marked as the potential start of a gradual transition F_S. Such a frame can be seen in figure 5.3. This frame is then compared to subsequent frames as shown in figure 5.3 in second plot. This is called *accumulated comparison* since, during a gradual transition, this difference value will normally increase. The end frame F_E of the transition is detected when the difference between consecutive frames decreases to less than T_s, while the accumulated comparison has increased to a value larger than T_b. The accumulated comparison is only computed when the difference between consecutive frames exceeds T_s. If the consecutive difference value drops below T_s before the accumulated comparison value exceeds T_b, then the potential start point is dropped and the search

continues for other gradual transition. A problem with TC is that there are some gradual transitions during which the consecutive difference value does fall below T_s. This problem is solved by permitting the user to set a tolerance value that allows a number of consecutive frames with low difference values before rejecting the transition candidate.

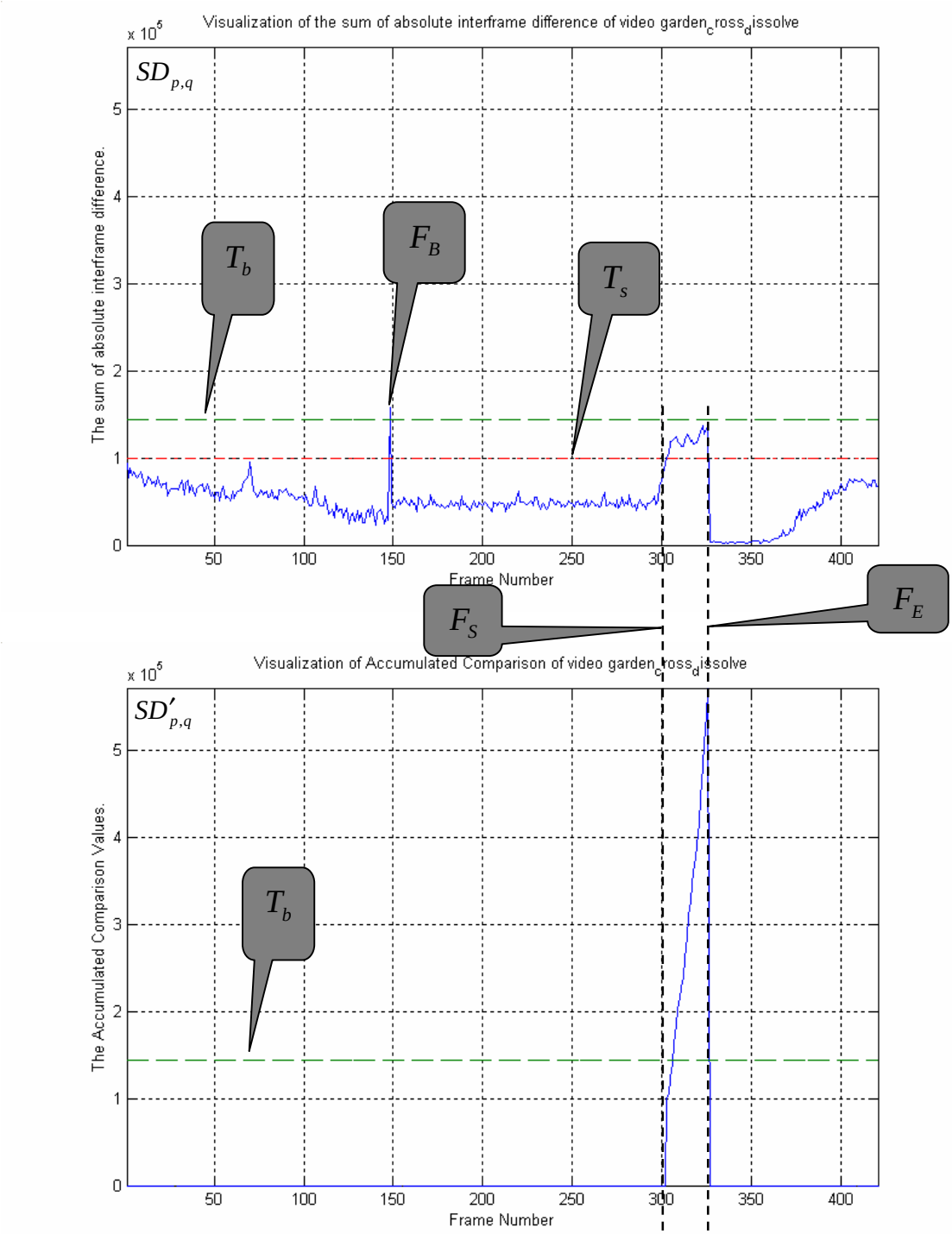


Figure 5.3: Illustration of TC. The first plot $SD_{p,q}$ is the difference between consecutive frames defined by the difference metric; the second plot $SD'_{p,q}$ is the accumulated difference between the current frame and the potential starting frame of a transition; T_s is the threshold used to detect the starting frame F_S of a transition; T_b is the threshold used to detect the ending frame F_E of a transition. T_b is also used to detect camera breaks such as F_B. $SD'_{p,q}$ is only calculated when $SD_{p,q} > T_s$.

Selection of Thresholds

Selection of appropriate threshold values is a key issue in applying the segmentation algorithms. Thresholds must be assigned that tolerate the variations in individual frames while still ensuring a desired level of performance. A "tight" threshold makes it difficult for "impostors" to be falsely accepted by the system, but at the risk of falsely rejecting true transitions. Conversely, a "loose" threshold enables transitions to be accepted consistently, at the risk of falsely accepting "impostors". In order to achieve high accuracy in video partitioning, an appropriate threshold must be found.

Considerable research has been done on the selection of thresholds for the spatial segmentation of static images. A good summary can be found in Digital Picture Processing by Rosenfeld & Kak (1982). Typically, selecting thresholds for such spatial segmentation is based on the histogram of the pixel values of the image. The conventional approaches include the use of a single threshold, multiple thresholds and variable thresholds. The accuracy of single threshold selection depends upon whether the histogram is bimodal, while multiple threshold selection requires clear multiple peaks in the histogram. Variable threshold selection is based on local histograms of specific regions in an image. In spite of this variety of techniques, threshold selection is still a difficult problem in image processing and is most successful when the solution is application dependent. In order to set an appropriate threshold for temporal segmentation of video sequences, we draw upon the same feature, the histogram of the frame-to-frame differences. It is necessary to know the distribution of the frame-to-frame differences across camera breaks and gradual transitions.

The automatic selection of threshold T_b is based on the frame-to-frame differences over a video source. The red curve in figure 5.4 shows a typical distribution of difference values (The graph of these values is the first plot in figure 5.3). This example is based on the difference metric for comparison of grayscale histograms obtained from a news video. The range of difference values is given on the horizontal axis, and the frequency of occurrence of each difference value is given on the vertical axis. This distribution exhibits a high and sharp peak on the left (near 50000) corresponding to a large number of consecutive frames that have a small difference between them. There is also a small peak near vertical axis (near 12000) corresponding to a small number of consecutive frames between which a very small or zero difference occurs. There is a small peak and a long tail on the right (near 125000) corresponding to a small number of consecutive frames that have a significant difference. Because this histogram has only a single modal point, the approaches for threshold selection already mentioned are not applicable.

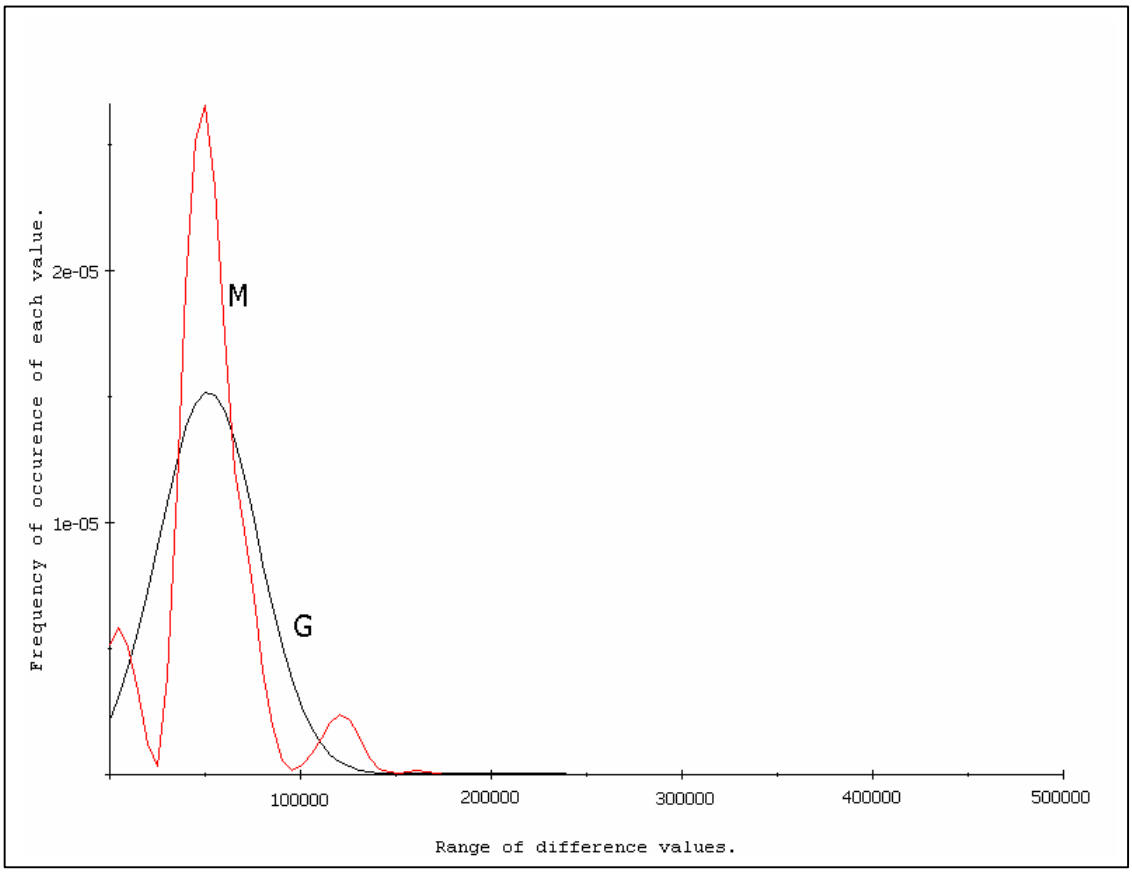


Figure 5.4: **M**, distribution of computed frame-to-frame differences based on a grayscale histogram for all frames of the news video; **G**, Gaussian distribution derived from the mean and variance of distribution **M**.

If there is no camera shot change or camera movement in a video sequence, the frame-to-frame difference value can only be due to three sources of noise:

1. Noise from digitizing the original analog video signal.
2. Noise introduced by video production equipment.
3. Noise resulting from the physical fact that few objects are perfectly still.

All three sources of noise can be assumed to be Gaussian. Thus, the distribution of frame-to-frame differences can be decomposed into a sum of two parts: the Gaussian noises and the differences introduced by camera breaks, gradual transitions and camera movements. Obviously, differences due to noise have nothing to do with transitions.

Let s be the standard deviation and m the mean of the frame-to-frame differences. If the only departure from m is due to Gaussian noise, then the probability integral

$$P(x) = \int_0^x \frac{1}{\sqrt{2p}\, s} e^{-\frac{|x-m|^2}{2s^2}} dx \quad \text{(Equation 5.0)}$$

will account for most of the frames within a few standard deviations of the mean value. In other words, the frame-to-frame differences from the non-transition frames will fall in the range of 0 to $m + as$ for a small constant value a. The black curve **G** in figure 5.4 shows the Gaussian distribution obtained from s and m for the frame-to-frame differences from which the **M** curve was calculated. Therefore, the threshold T_b can be selected as

$$T_b = m + as \quad \text{(Equation 5.1)}$$

That is, difference values that fall out of the range from 0 to $m + as$ can be considered indicators of segment boundaries. From experiments, the value a should be chosen between 3 and 4. Under a Gaussian distribution, the probability that a non-transition frame will fall out of this range is practically zero.

For detecting gradual transitions, another threshold T_s also needs to be selected. Experiments have shown that T_s should be selected along the right slope of the **M** distribution shown in figure 5.4. Furthermore, T_s should generally be larger than the mean value of the frame-to-frame differences of the entire video. Therefore, the threshold T_s can be selected as

$$T_s = bm \quad \text{(Equation 5.2)}$$

From experiments, the value b should be chosen between 1.5 and 2.

Sliding Window Method

Difference metric selection

To partition the video, we should first define suitable metrics, so that a shot boundary is declared whenever that metric exceeds a given threshold. It is been used histogram difference because histogram is less sensitive to object motion than other two metrics which presented in table 5.0.

Sliding Window (SW) approach for shot detection

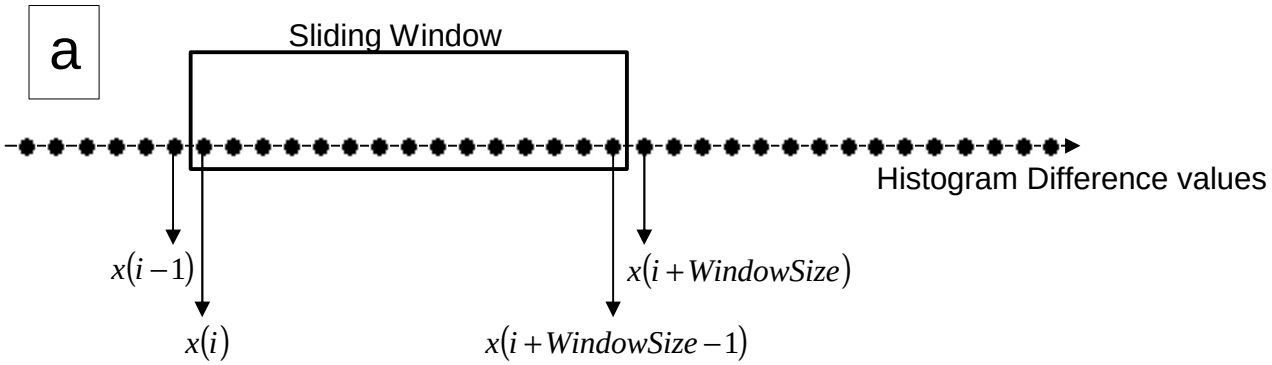
As mentioned before the metric which will be used for SW approach, will be the histogram difference. Some example graphs can be seen in figure 5.2. Next follows how SW approach is used with histogram difference.

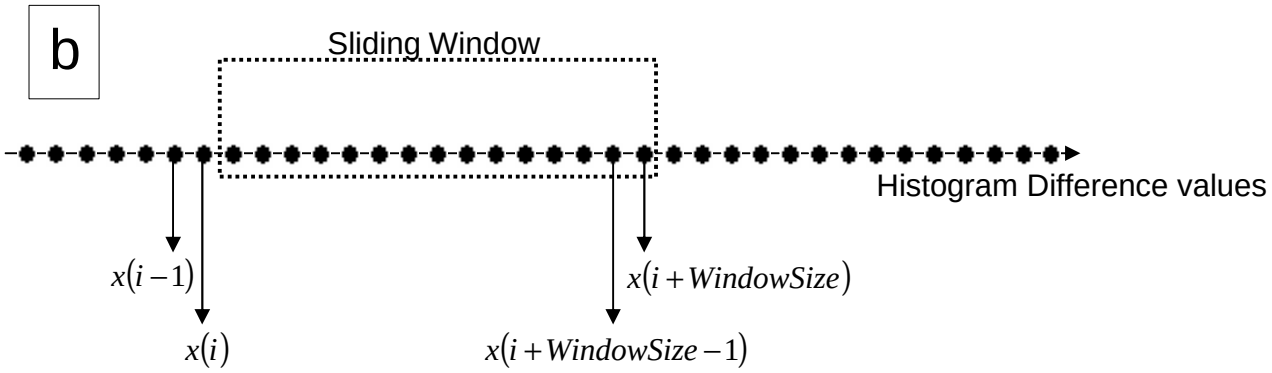
Before the explanation of the SW algorithm, it is necessary to give some notations. The sliding window which is used has a size of **WindowSize** frames. The value of **WindowSize** is standard to 15 which do not change during the process of the entire video. $x(i)$ is the frame-to-frame histogram difference value of i-th frame. The video has a size of **TotalFrames** frames. So we have for variable i, $0 \le i \le (TotalFrames - 1)$. $m_W(j)$ is the mean value of the data within the sliding window. $s_W(j)$ is the variance of the data within the sliding window. $T_W(j)$ is the threshold of the data within the sliding window. The variable j denotes how many instances of the sliding window can be produced during the processing of the entire video. So the limits of variable j are $0 \le j \le ((TotalFrames - WindowSize) - 1)$.

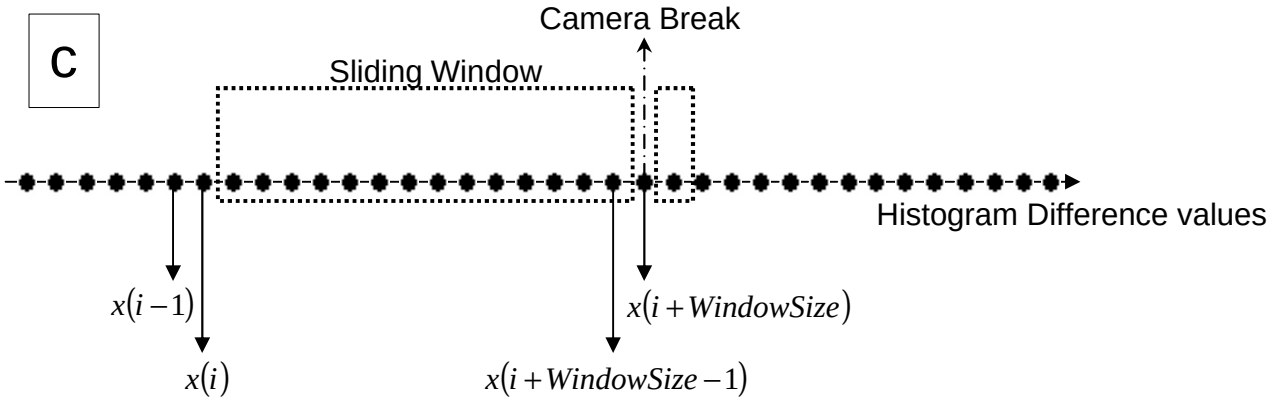
We build a sliding window preceding the current frame. The next step is to calculate the mean $m_W(j)$ and variance $s_W(j)$ value of the histogram difference data in the window. The threshold $T_W(j)$ is calculated by the formula $T_W(j) = m_W(j) + a_W s_W(j)$, where a_W is a constant variable for the entire video. The value of variable a_W is changing according to video type. A typical value, for video type news is between 3.5 and 5. The following step is to compare the threshold $T_W(j)$ with the histogram difference value which is next to the right of the sliding window. In other words, if $x(i),\ldots,x(i + WindowSize - 1)$ are in the sliding window, the threshold $T_W(j)$, which calculated from the above values, is compared to $x(i + WindowSize)$. If $T_W(j) > x(i + WindowSize)$, the sliding window is moving one frame to the right, so $x(i)$ is getting out from the left of the window and $x(i + WindowSize)$ is getting in from the right of the window. If

$T_W(j) \leq x(i + WindowSize)$, the sliding window is not moving and none value get in the window if the first condition, $T_W(j) > x(i + WindowSize)$, is not satisfied. The value $x(i + WindowSize)$ is declared as Camera Break. The value of $T_W(j)$ will be the same while the condition $T_W(j) \leq x(i + WindowSize)$ is satisfied.

One threshold need to be calculated in SW approach in order to partition video in shots. The camera breaks happens in one frame and gradual transitions occupy more than one frame. So, if the condition $T_W(j) \leq x(i + WindowSize)$ is satisfied for one frame, then we have a Camera Break. If the previous condition is satisfied for more than one continuous frame, then we have Gradual Transition. Some possible instances of sliding window can be seen in figure 5.5.


a
Sliding Window
Histogram Difference values
$x(i-1)$
$x(i)$
$x(i+WindowSize)$
$x(i+WindowSize-1)$


b
Sliding Window
Histogram Difference values
$x(i-1)$
$x(i)$
$x(i+WindowSize)$
$x(i+WindowSize-1)$


c
Camera Break
Sliding Window
Histogram Difference values
$x(i-1)$
$x(i)$
$x(i+WindowSize)$
$x(i+WindowSize-1)$

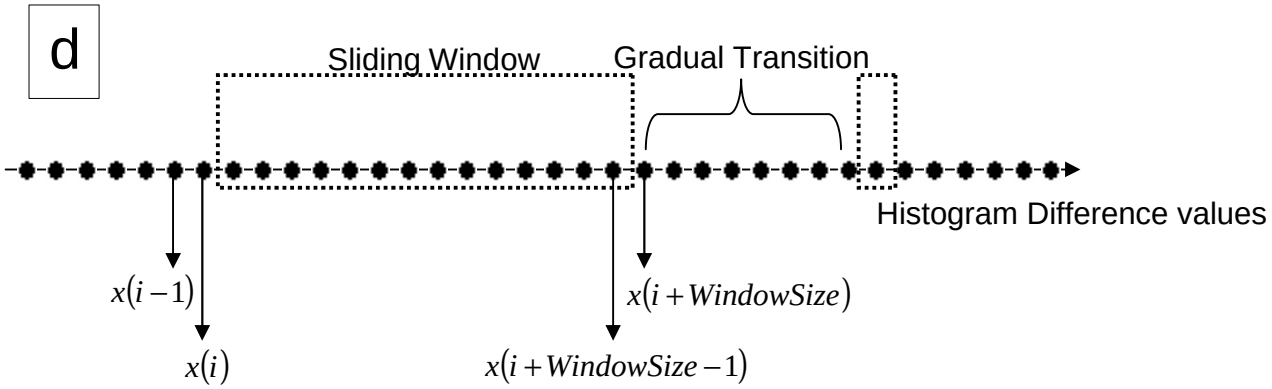
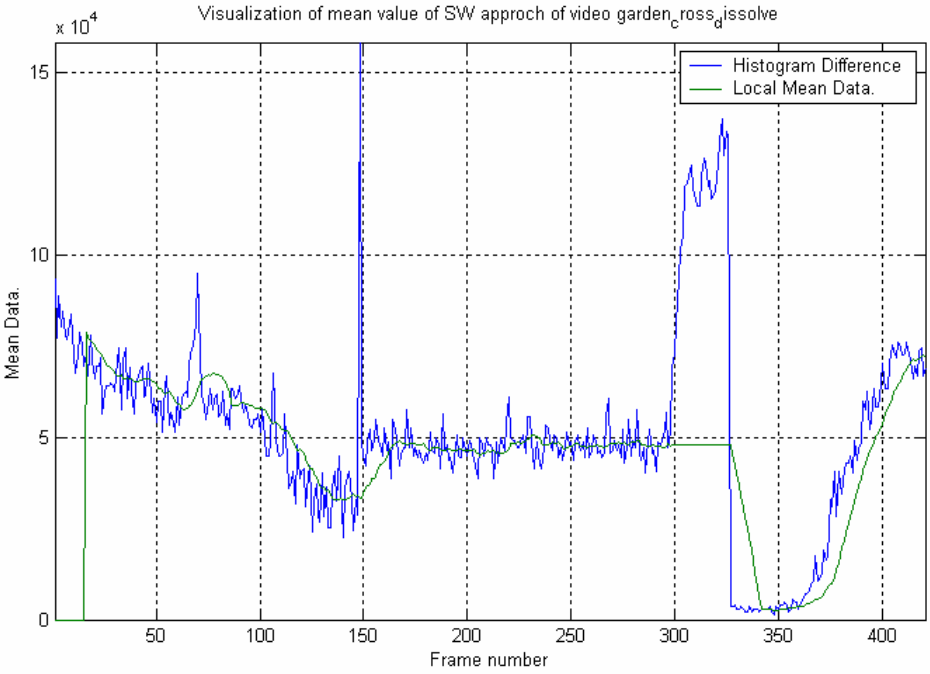
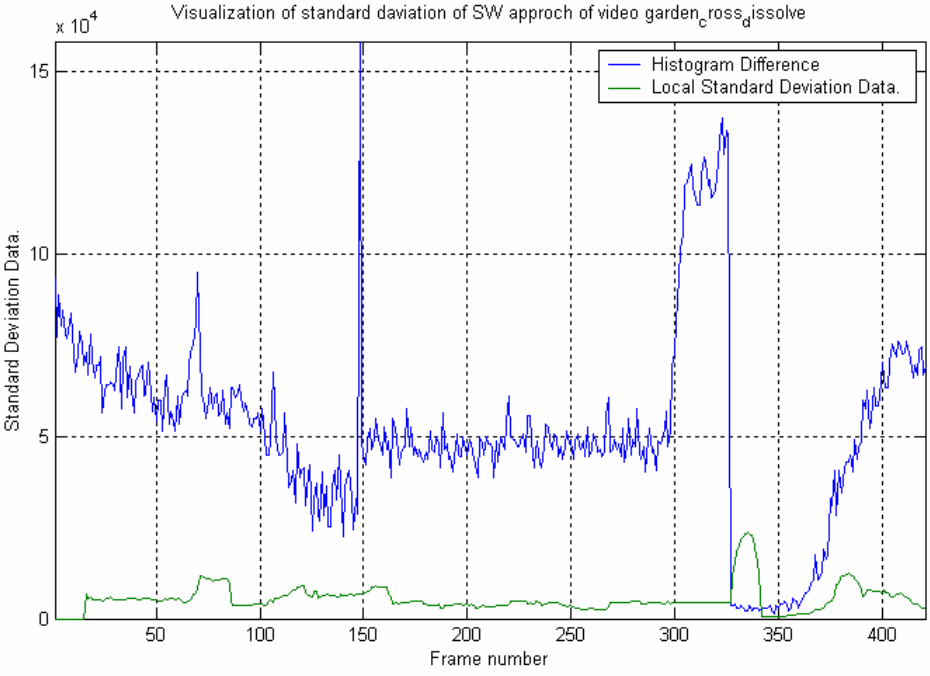


Figure 5.5: (**a**) Illustration of a sliding window instance j in histogram difference graph. (**b**) Illustration of a sliding window instance $j+1$, if condition $T_W(j) > x(i+WindowSize)$ is satisfied. (**c**) Illustration of a sliding window instance $j+1$, if condition $T_W(j) \le x(i+WindowSize)$ is satisfied for one frame. In this case we have Camera Break. (**d**) Illustration of a sliding window instance $j+1$, if condition $T_W(j) \le x(i+WindowSize)$ is satisfied for more than one frame. In this case we have Gradual Transition.

Selection of Thresholds

As mentioned before, the threshold T_W is calculated by the formula $T_W = m_W + a_W s_W$. The reason of selection this value for threshold is explained in TC approach. One threshold is used in SW approach because this threshold is adapted to the local variations of histogram difference. In other words, the graph of the threshold follows the variations of the histogram difference graph. An example of m_W, s_W and T_W graphs with histogram difference graph can be seen in figure 5.6.


Visualization of mean value of SW approch of video garden_cross_dissolve
x 10^4
Histogram Difference
Local Mean Data.
Mean Data.
Frame number
0
5
10
15
50
100
150
200
250
300
350
400


Visualization of standard daviation of SW approch of video garden_cross_dissolve
x 10^4
Histogram Difference
Local Standard Deviation Data.
Standard Deviation Data.
Frame number
0
5
10
15
50
100
150
200
250
300
350
400

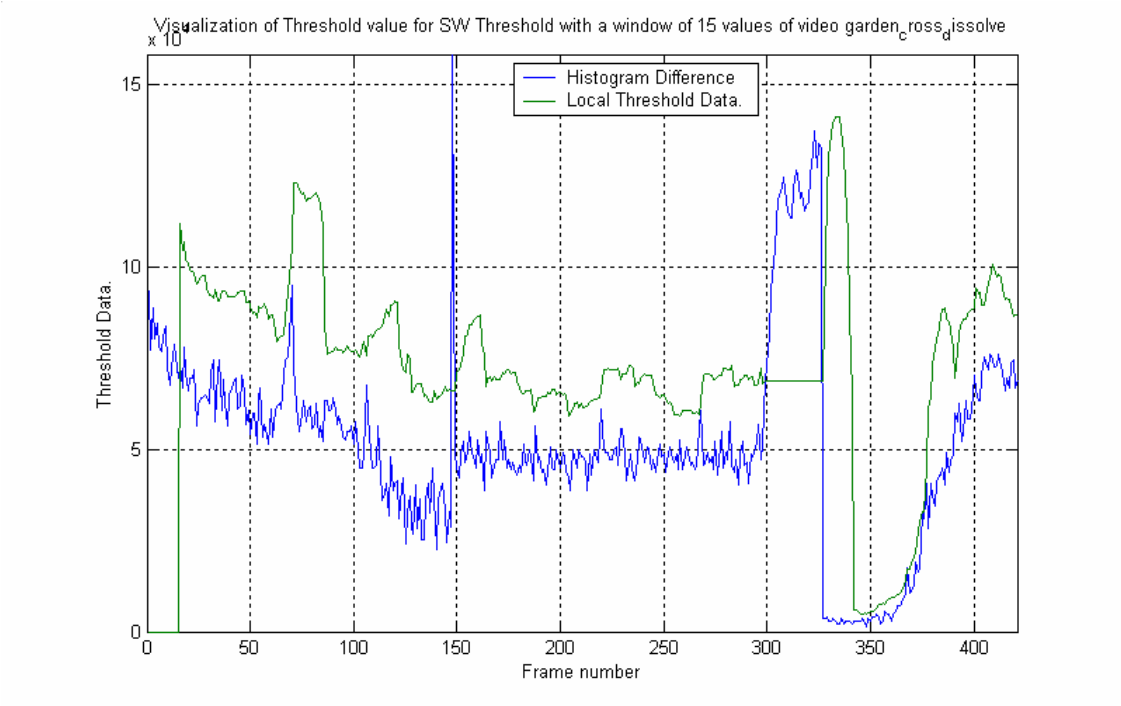


Figure 5.6: The three plots are for $m_W(j)$, $s_W(j)$ and $T_W(j)$, respectively. It is worth mention that the first 15 values of histogram difference are used to calculate the first value of $m_W(j)$, $s_W(j)$ and $T_W(j)$. Before the first value, we can not calculate $m_W(j)$, $s_W(j)$ and $T_W(j)$, so their values are zero.

Adaptive Method

Difference metric selection

For video partition, it is necessary to define suitable metrics, so that a shot boundary is declared whenever that metric exceeds a given threshold. It is been used histogram difference because histogram is less sensitive to object motion than other two metrics which presented in table 5.0.

Adaptive approach for shot detection

As mentioned before the metric which will be used for Adaptive approach, will be the histogram difference. Some example graphs can be seen in figure 5.2. Next follows how Adaptive approach is used with histogram difference.

Before the explanation of the Adaptive algorithm, it is necessary to give some notations. $x(i)$ is the frame-to-frame histogram difference value of i-th frame. The video has a size of **TotalFrames** frames. So we have for variable i, $0 \le i \le (TotalFrames - 1)$. $m_A(i)$ is the mean value of the data which is calculated by the adaptive algorithm. $s_A(i)$ is the variance of the data which is calculated by the adaptive algorithm. $T_A(i)$ is the threshold of the data which is calculated by the adaptive algorithm.

The calculation of adaptive mean value $m_A(i)$ and adaptive standard deviation $s_A(i)$ is being by the following formulas:

$$m_A(i) = m_A(i-1) - c(m_A(i-1) - x(i)) \Leftrightarrow m_A(i) = (1-c)m_A(i-1) + cx(i) \quad \text{(Equation 5.3)}$$

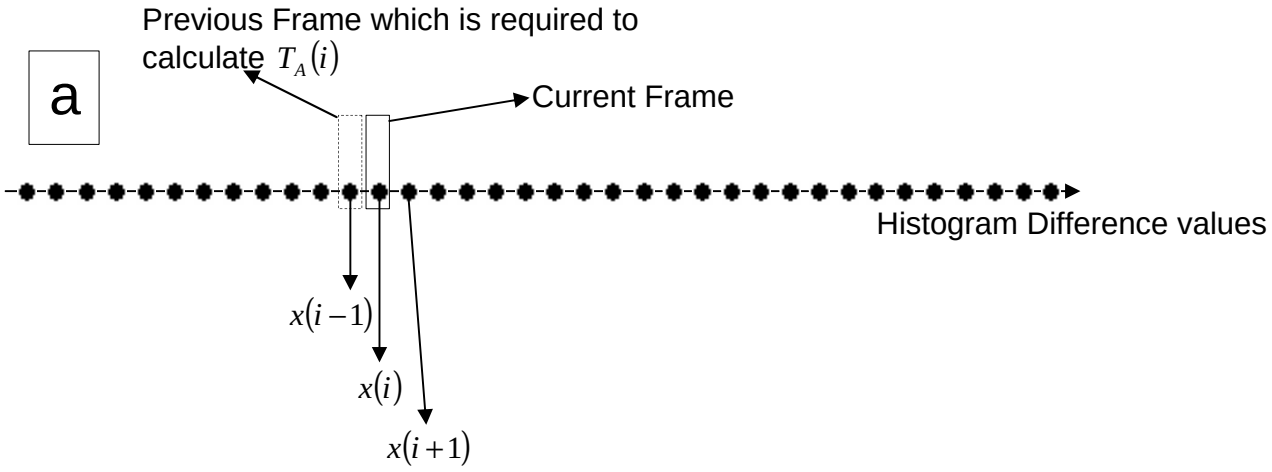
$$s_A(i) = \sqrt{m_A(i)^2 - l_A(i)} \quad \text{(Equation 5.4)}$$

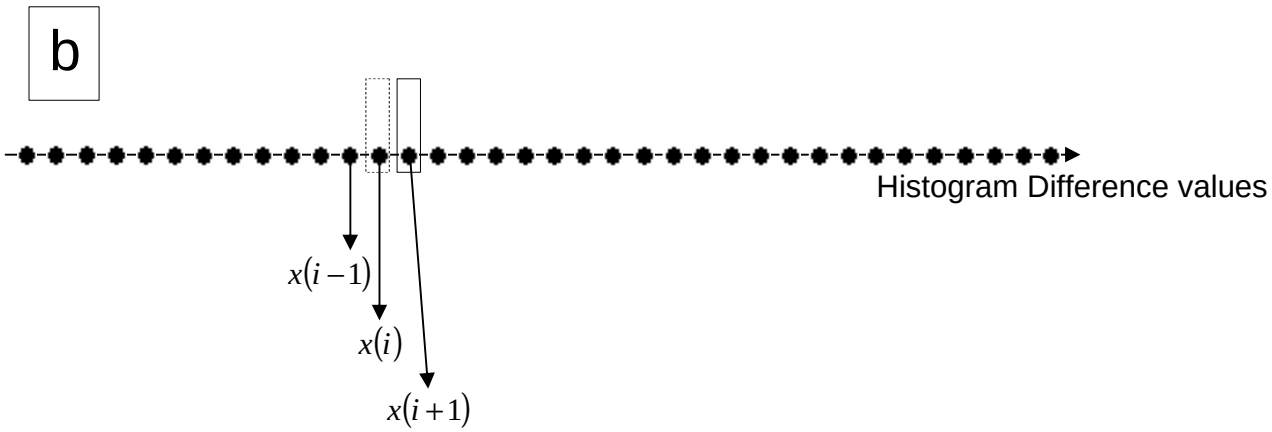
where c is a controlling coefficient $c \in [0,1]$ and $l_A(i)$ is the adaptive second moment

$$l_A(i) = l_A(i-1) - c\left(l_A(i-1) - x(i)^2\right) \Leftrightarrow l_A(i) = (1-c)\,l_A(i-1) + cx(i)^2 \quad \text{(Equation 5.5)}$$

The initial values are $m_A(0) = x(0)$ and $l_A(0) = x(0)^2$. Default value for $c = 0.05$. The threshold $T_A(i)$ is calculated by the formula $T_A(i) = m_A(i) + a_A s_A(i)$, where a_A is a constant variable for the entire video. The value of variable a_A is changing according to video type. A typical value, for video type news is between 4 and 5. The next step is to compare the threshold $T_A(i)$ with the next histogram difference value. In other words, the threshold

$T_A(i)$ is compared to $x(i+1)$. If $T_A(i) > x(i+1)$, then we continue with the calculation of $T_A(i+1)$. If $T_A(i) \le x(i+1)$, then $T_A(i+1) = T_A(i)$ until the first condition, $T_A(i) > x(i+1)$, is satisfied. The value $x(i+1)$ is declared as Camera Break.





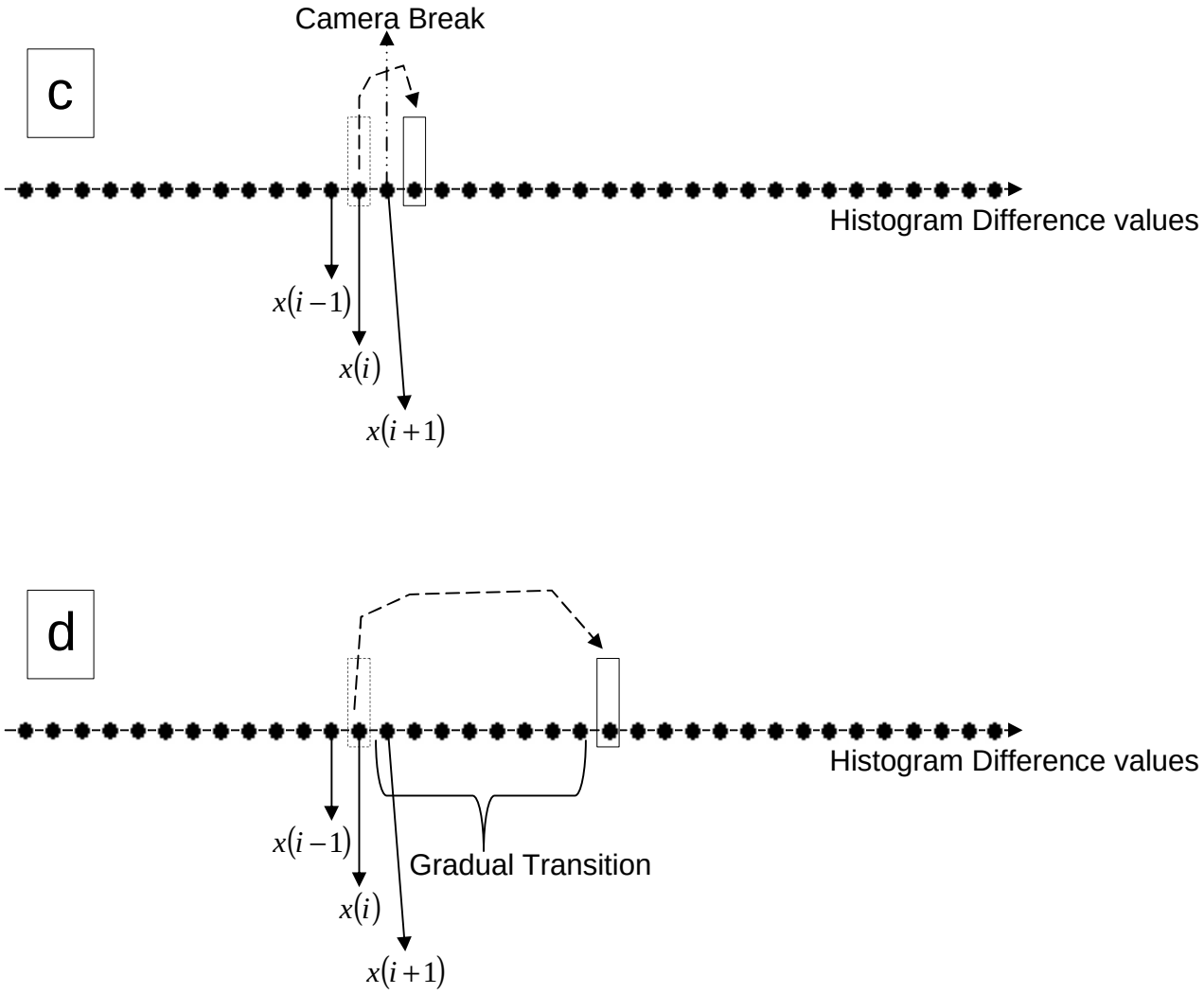
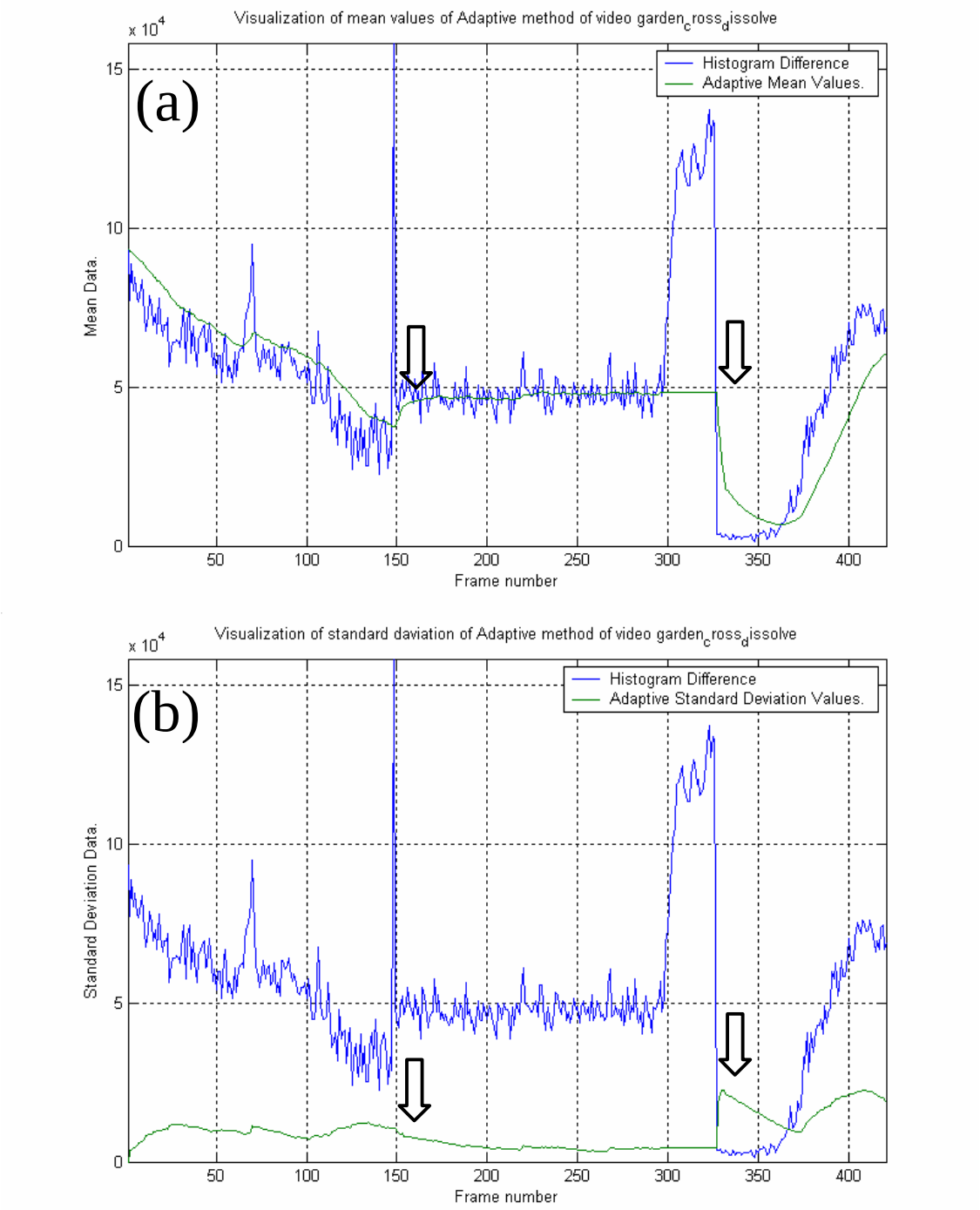
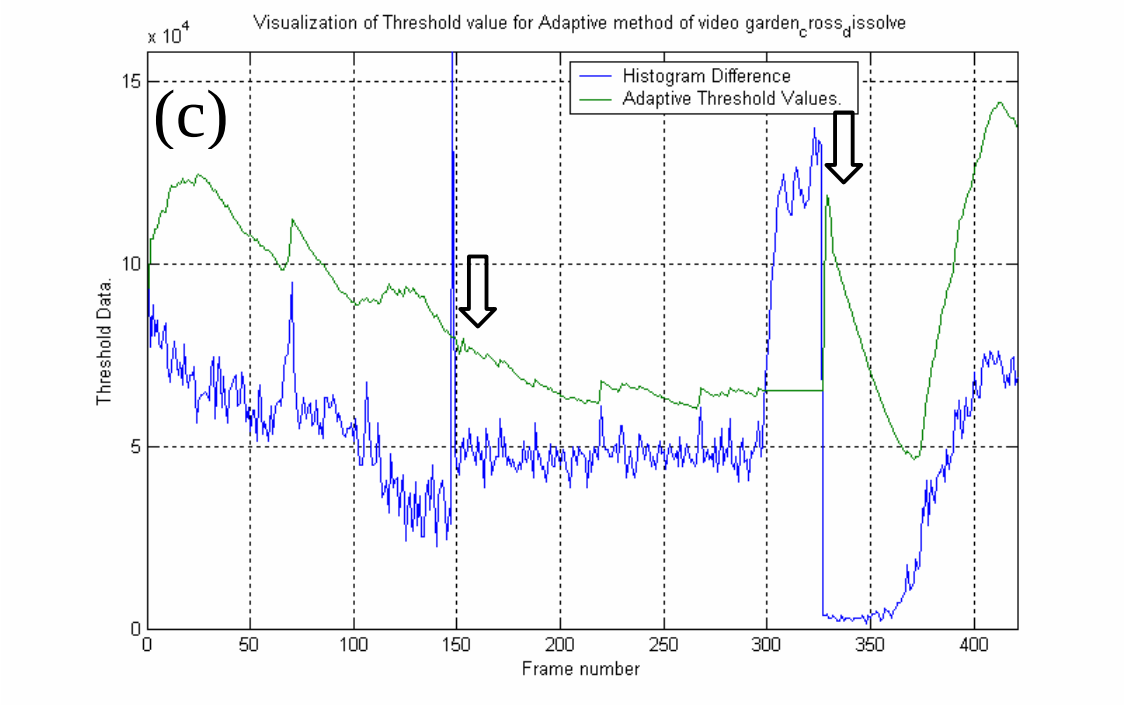


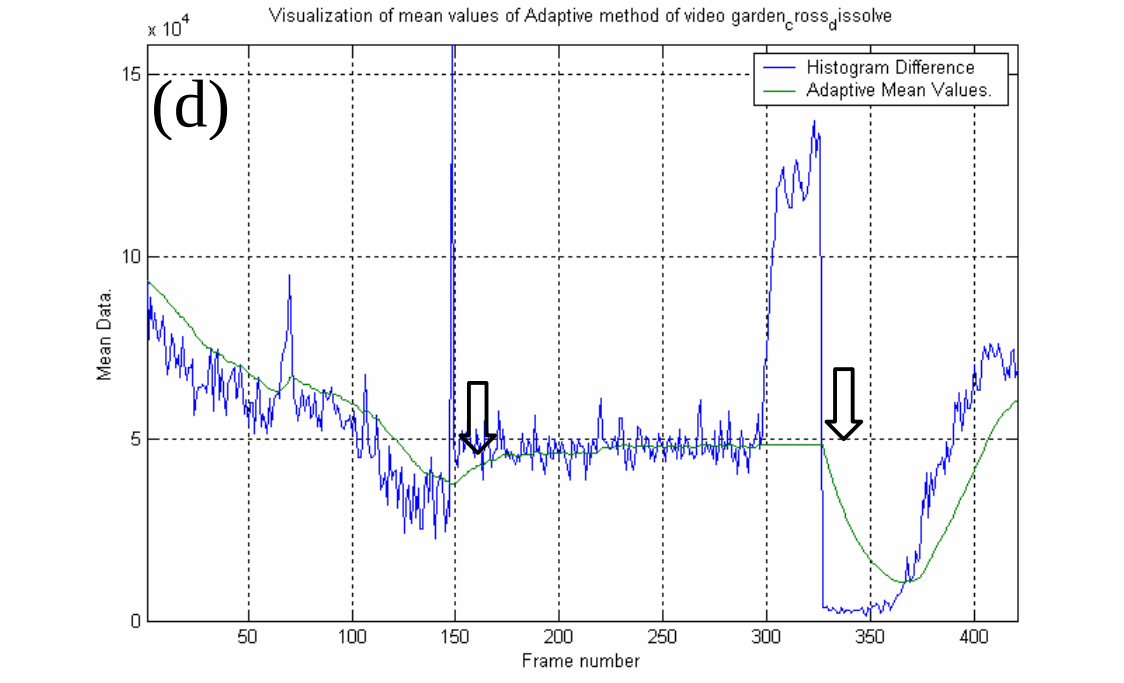
Figure 5.7: (**a**) Illustration of adaptive method in a histogram difference graph. (**b**) Illustration of adaptive method, if condition $T_A(i) > x(i+1)$ is satisfied. (**c**) Illustration of adaptive method, if condition $T_A(i) \leq x(i+1)$ is satisfied for one frame. In this case we have Camera Break. (**d**) Illustration of adaptive method, if condition $T_A(i) \leq x(i+1)$ is satisfied for more than one frame. In this case we have Gradual Transition.

Selection of Thresholds

As mentioned before, the threshold T_A is calculated by the formula $T_A = m_A + a_A s_A$. The reason of selection this value for threshold is explained in TC approach. One threshold is used in Adaptive approach, as in SW approach, because this threshold is adapted to the local variations of histogram difference. In other words, the graph of the threshold follows the variations of the histogram difference graph. An example of m_A, s_A and T_A graphs with histogram difference graph can be seen in figure 5.8.


Visualization of mean values of Adaptive method of video garden_cross_dissolve
x 10^4
(a)
Histogram Difference
Adaptive Mean Values.
Mean Data.
Frame number
Visualization of standard daviation of Adaptive method of video garden_cross_dissolve
x 10^4
(b)
Histogram Difference
Adaptive Standard Deviation Values.
Standard Deviation Data.
Frame number


Visualization of Threshold value for Adaptive method of video garden_cross_dissolve
x 10^4
(c)
Histogram Difference
Adaptive Threshold Values.
Threshold Data.
Frame number
0
5
10
15
0
50
100
150
200
250
300
350
400


Visualization of mean values of Adaptive method of video garden_cross_dissolve
x 10^4
(d)
Histogram Difference
Adaptive Mean Values.
Mean Data.
Frame number
0
5
10
15
50
100
150
200
250
300
350
400

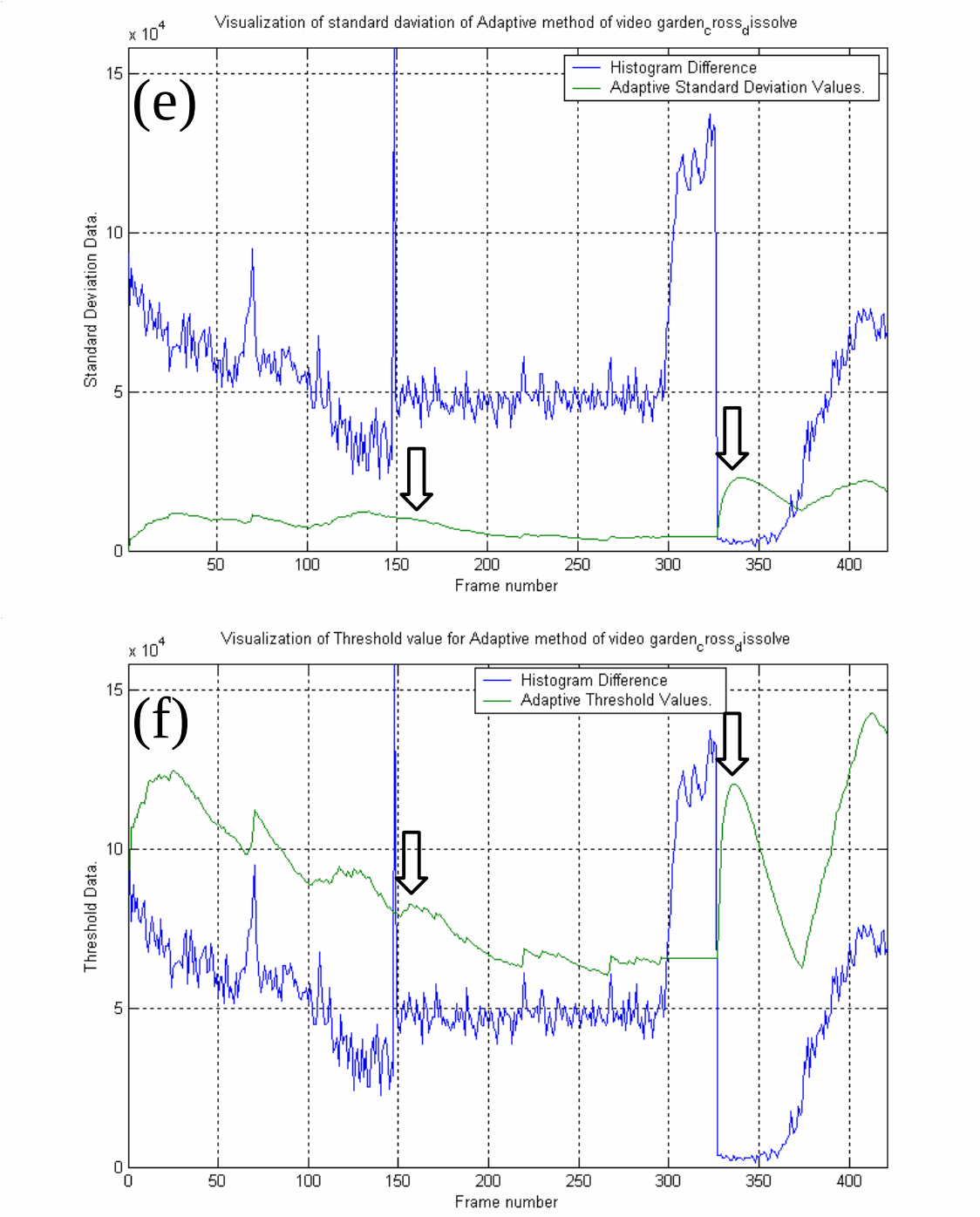


Figure 5.8: The six plots are for $m_A(i)$, $s_A(i)$ and $T_A(i)$ with different c, respectively. (**a**) It is $m_A(i)$ with $c = 0.05$. For five frames after a Camera Break or Gradual Transition $c = 0.20$. (**b**) It is $s_A(i)$ with $c = 0.05$. For five frames after a Camera Break or Gradual Transition $c = 0.20$. (**c**) It is $T_A(i)$ with $c = 0.05$. For five frames after a Camera Break or Gradual Transition $c = 0.20$. (**d**) It is $m_A(i)$ with $c = 0.05$ in whole video. (**e**) It is $s_A(i)$ with $c = 0.05$ in whole video. (**f**) It is $s_A(i)$ with $c = 0.05$ in whole video. The arrows show the graph after a Camera Break or Gradual Transition.

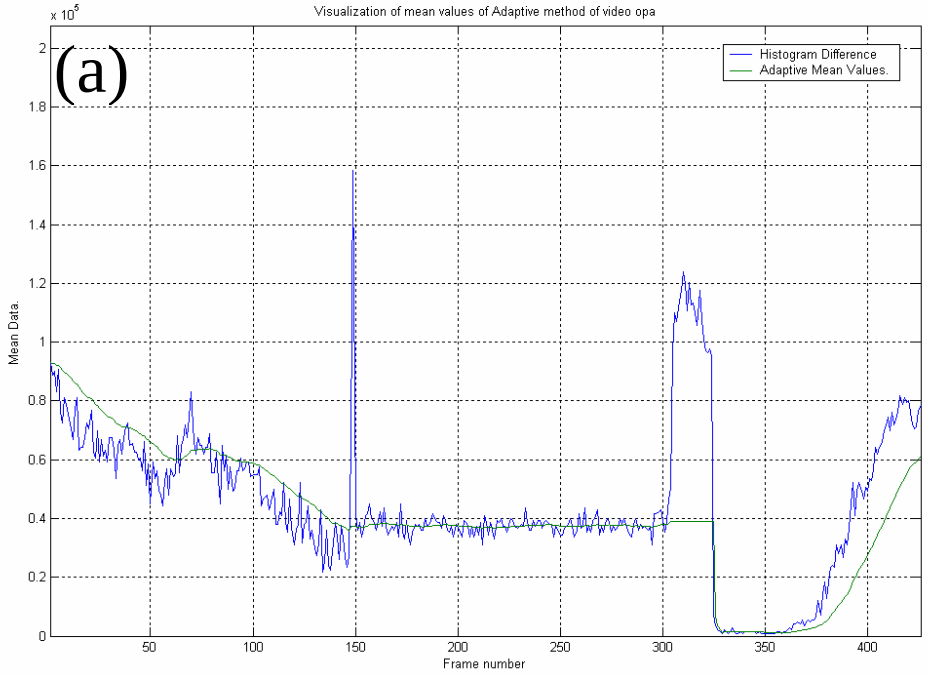
It is necessary to make a small introduction about motion estimation before we continue to motion based methods.

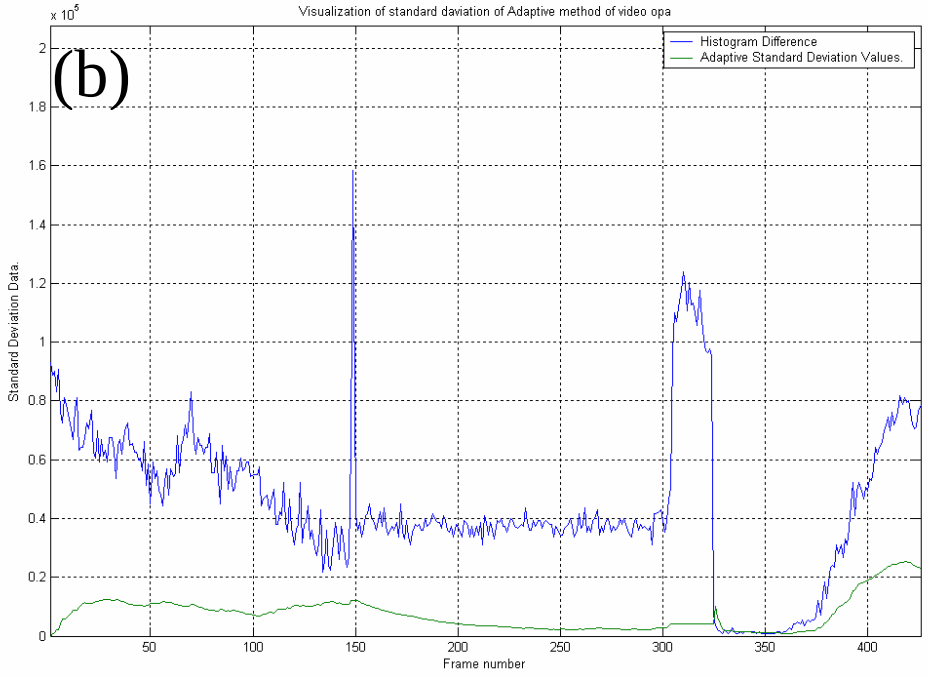
A different value for c variable

We observed in (**c**) and (**f**) graphs in figure 5.8 that when $c = 0.20$ after five frames of a Camera Break or Gradual Transition, we have a peak that decrease faster than the peak when $c = 0.05$. In both cases, the peak has a big value and it is necessary to decrease that value. The value is big because of the standard deviation (we have big difference values in standard deviation) as we can see in (**b**) and (**e**) graphs in figure 5.8. In order to achieve small values we cannot change the equations 5.3 and 5.5, but we can change the value of c in a way that can be adaptive to the values of histogram difference. So, the value of c can be

$$c = \frac{\left(m_{\mathrm{A}}(i-1) - x(i)\right)}{\max\left\{m_{\mathrm{A}}(i-1), x(i)\right\}} \quad \text{(Equation 5.6)}$$

With the above equation, the threshold adjusts with the signal (histogram difference) for five frames after a Camera Break or Gradual Transition. Figure 5.9 shows a graph with the above improvement.


(a)
Visualization of mean values of Adaptive method of video opa
x 10^5
Histogram Difference
Adaptive Mean Values.
Mean Data.
Frame number
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
50
100
150
200
250
300
350
400


(b)
Visualization of standard daviation of Adaptive method of video opa
x 10^5
Histogram Difference
Adaptive Standard Deviation Values.
Standard Deviation Data.
Frame number
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
50
100
150
200
250
300
350
400

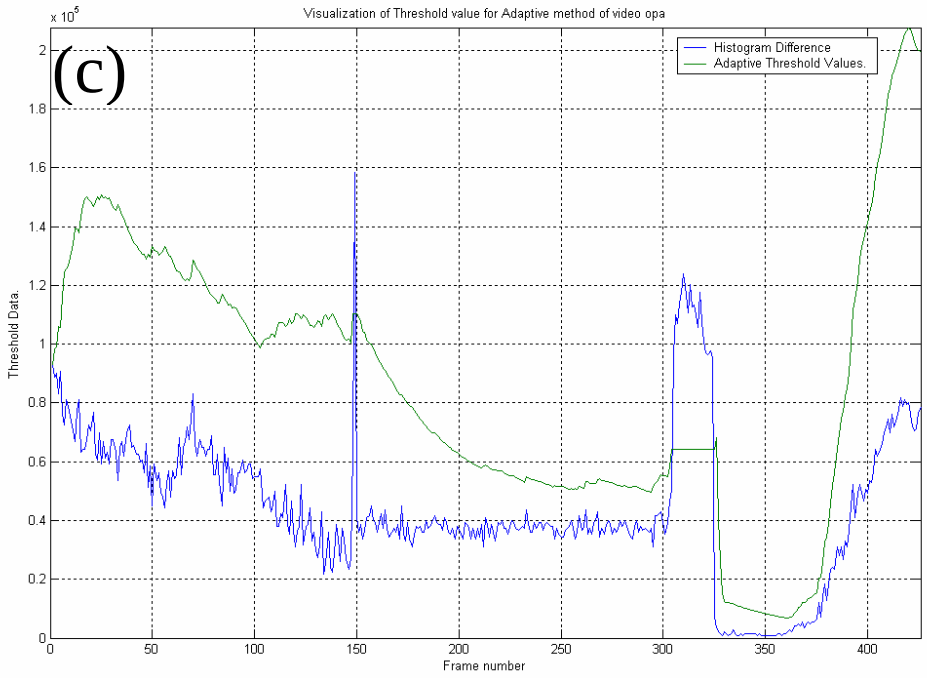


Figure 5.9: The three plots are for $m_A(i)$, $s_A(i)$ and $T_A(i)$ with c calculated by equation 5.6, respectively. (**a**) It is $m_A(i)$ with $c = 0.05$. For five frames after a Camera Break or Gradual Transition $c = \frac{(m_A(i-1) - x(i))}{\max\{m_A(i-1), x(i)\}}$. (**b**) It is $s_A(i)$ with $c = 0.05$. For five frames after a Camera Break or Gradual Transition $c = \frac{(m_A(i-1) - x(i))}{\max\{m_A(i-1), x(i)\}}$. (**c**) It is $T_A(i)$ with $c = 0.05$. For five frames after a Camera Break or Gradual Transition $c = \frac{(m_A(i-1) - x(i))}{\max\{m_A(i-1), x(i)\}}$.

Introduction to Motion Estimation

Motion is a prominent source of temporal variations in image sequences. In order to model and compute motion, we need to understand how images (and therefore image motion) are formed. Motion in image sequences acquired by a video camera is induced by the movements of objects in a 3D scene and by camera motion. Thus, camera's parameters, such as its 3D motion (rotation, translation) or focal length, play an important role in image motion modeling. If we know these parameters precisely, only object motion needs to be recovered. However, this scenario is rather rare, and both object and camera motion usually needs to be computed. The 3D motion of objects and cameras induces 2D motion on the image plane via a suitable projection system. It is this 2D motion, also called *apparent motion* or *optical flow*, which needs to be recovered from intensity and color information of a video sequence. 2D motion finds diverse applications in video processing and compression as well as in computer vision, primarily because the temporal correlation of intensities in an image sequence is very high in the direction of motion.

In video compression, the knowledge of motion helps remove temporal data redundancy and therefore, attain high compression ratios. Motion estimation became a fundamental component of such standards as H.261, H.263 and the MPEG family. Although motion models used by the older standards are very simple (one 2D vector per block), the new MPEG-4 standard offers an alternative (region-based) model that allows increased efficiency and flexibility. In video processing, motion information is used for standards conversion (motion-compensated 3D sampling structure conversion), noise suppression (motion-compensated filtering), or deblurring (motion-compensated restoration). In computer vision, 2D motion usually serves as an intermediary in the recovery of camera motion or scene structure.

To compute motion trajectories, three basic elements need to be specified. ***First***, underlying **models** must be selected, e.g. the motion model (*representation, region of support*), motion and image data relationship model (observation model), motion boundary model and occlusion model. The choice of models and their parameters is application dependent. For example, the occlusion model may not be relevant for block-based compression, whereas it would be essential in image analysis. ***Second***, an **estimation criterion** must be identified. Such a criterion may take different forms, such as a simple mean-squared error over a block, a robust criterion (e.g. with saturation for large errors), or a complex rate-distortion or Bayesian criterion involving multiple terms. ***Third***, a **search strategy** must be implemented to determine the motion parameters that optimize the selected criterion. In general, by a suitable

selection of search strategy, one can trade, to a large extent, optimization performance against computational load. The strategy may be deterministic or stochastic in nature. Exhaustive and simplified search methods as well as deterministic relaxation belong to the most popular schemes and include, as special cases, block matching and gradient-based methods. Among the best-known deterministic relaxation methods are *Iterated Conditional Modes* and *Highest Confidence First*. Mean-field techniques stemming from statistical mechanics are important deterministic optimization techniques based on the approximation of a partition function. Stochastic relaxation techniques, including simulated annealing, are dominant among the stochastic approaches. An important element of the optimization strategy is its hierarchical implementation in order to avoid the violation of some underlying assumptions (e.g. local intensity linearity) and/or reduce the computational complexity of the algorithm.

Motion Models

Motion Representation

Consider a point on an object moving in 3D space. Let its position at time t be

$$\mathbf{X} = \mathbf{X}(t) = (X(t), Y(t), Z(t))^T \in \Re^3 \quad \text{(Equation 5.7)}$$

expressed in camera coordinates. $(\mathbf{X}(t), t)$ defines a curve in 3D space over time which we refer to as the *world motion trajectory*. For any two time instants t and t', the world motion trajectory identifies a 3D displacement in position

$$D_{t,t'}(\mathbf{X}) = \mathbf{X}(t') - \mathbf{X}(t) \quad \text{(Equation 5.8)}$$

An image acquisition system projects the 3D world onto a 2D image plane with image coordinates $\mathbf{x} = (x, y)^T \in \Lambda$, where Λ is a sampling grid, usually an orthogonal lattice. Upon this projection, the world motion trajectories result in motion trajectories $(\mathbf{x}(t), t)$. We adopt the definition of a 2D motion trajectory proposed in Dubois & Konrad: a trajectory is defined only in the time interval in which the associated point is visible in the image. Thus, assuming that we are dealing with non-transparent objects, each spatio-temporal position $(\mathbf{x}, t)$ belongs to a motion trajectory of only one visible point. As depicted in figure 5.10, the 2D

displacement can be expressed, similarly to the 3D displacement, as follows

$$\mathbf{d}_{t,t}(\mathbf{x}) = \mathbf{x}(t') - \mathbf{x}(t) \quad \text{(Equation 5.9)}$$

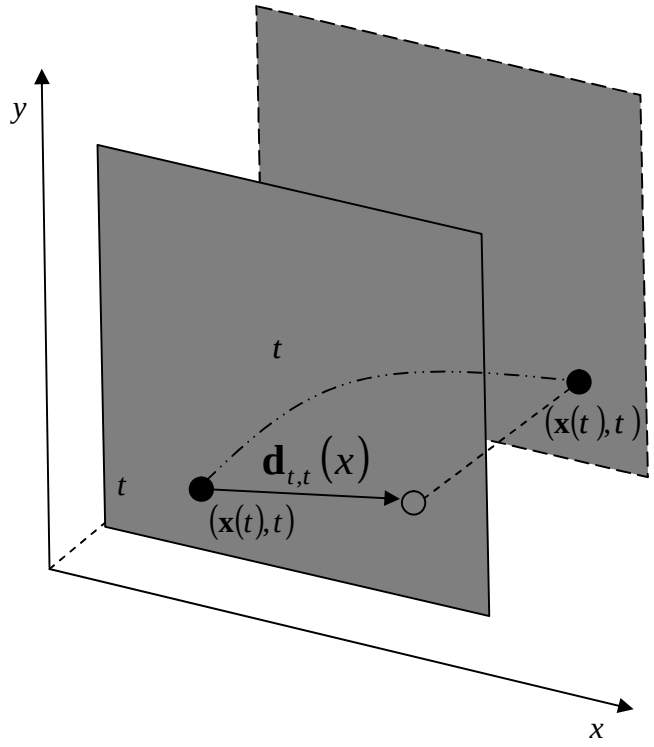


Figure 5.10: Motion trajectory $\mathbf{x}(t)$ and associated displacement vector $\mathbf{d}_{t,t}(x)$

In general, a motion field is a vector-valued function of continuous spatial coordinates. In practical applications, this function is often described in a parametric form using a finite, usually small, number of parameters.

Since 2D motion results from the projection of moving 3D objects onto the image plane, a model for 2D motion fields can be derived from the models describing 3D motion, 3D surface function and camera projection geometry. If these models are parametric, the resulting 2D motion model will be parametric as well. As an example, consider a 3D planar patch undergoing 3D affine motion under orthographic projection. The 3D affine motion can be written as follows

$$D(\mathbf{X}) = (\mathbf{R} - \mathbf{I})\mathbf{X} + \mathbf{s} \quad \text{(Equation 5.10)}$$

In general, the 3x3 matrix $\mathbf{R} = (r_{ij})$ has nine degrees of freedom and the translational motion vector $s = (s_1, s_2, s_3)^T$ has another three degrees of freedom. Equation 5.10 includes rigid motion as a special case. Then, $\mathbf{R}$ is a rotation matrix, i.e. its columns are orthonormal, thus allowing only three degrees of freedom corresponding to the three rotation axes.

Let the planar patch be specified by three parameters a, b, g as follows

$$aX + bY + gZ = 1 \quad \text{(Equation 5.11)}$$

The camera model provides two additional scalar equations mapping 3D world coordinates onto 2D coordinates of the image plane. For orthographic projection, the following relationship holds:

$$x = cX, y = cY; c \in \Re \quad \text{(Equation 5.12)}$$

Substituting equations for the camera model (Equation 5.12) and for the 3D surface (Equation 5.11) into equation 5.10, we readily obtain a model for 2D motion which, for the given example, becomes the 2D affine model

$$\mathbf{d(x)} = (\mathbf{A} - \mathbf{I})\mathbf{x} + \mathbf{b} \quad \text{(Equation 5.13)}$$

with

$$\mathbf{A} = \begin{pmatrix} r_{11} - \frac{a}{g} r_{13} & r_{12} - \frac{b}{g} r_{13} \\ r_{21} - \frac{a}{g} r_{23} & r_{22} - \frac{b}{g} r_{23} \end{pmatrix}, \mathbf{b} = \begin{pmatrix} \frac{c}{g} r_{13} + cs_1 \\ \frac{c}{g} r_{23} + cs_2 \end{pmatrix} \quad \text{(Equation 5.14)}$$

Clearly, a 2D motion model does not uniquely correspond to one 3D model; identical 2D motion models may result from different assumptions about 3D motion, surface and camera projection models.

Table 5.1 summarizes some parametric models for 2D motion and provides possible underlying assumptions.

Table 5.1: Motion Models

2D Model			3D Model		Camera Model
	Number of parameters	Motion field	3D surface function	3D motion	
Translational	2	$d(x) = (a_1, b_1)^T$	Arbitrary	Rigid 3D translation	Orthographic
Affine	6	$d(x) = \begin{pmatrix} a_1 & a_2 \\ b_1 & b_2 \end{pmatrix} x + \begin{pmatrix} a_3 \\ b_3 \end{pmatrix}$	Planar	3D affine	Orthographic
Projective linear	8	$d(x) = \begin{pmatrix} \frac{a_1 + a_2 x + a_3 y}{1 + a_4 x + b_4 y} \\ \frac{b_1 + b_2 x + b_3 y}{1 + a_4 x + b_4 y} \end{pmatrix} - x$	Planar	3D affine	Perspective
Quadratic	12	$d(x) = \begin{pmatrix} a_1 + a_2 x + a_3 y + a_6 x^2 + a_5 xy + a_4 y^2 \\ b_1 + b_2 x + b_3 y + b_6 x^2 + b_5 xy + b_4 y^2 \end{pmatrix}$	Parabolic	3D affine	Orthographic
Sampled	2 per Δ^2 pixels	$d(x) = \sum_{i,j} \begin{pmatrix} a_{ij} \\ b_{ij} \end{pmatrix} H(x - \Delta \cdot i, y - \Delta \cdot j)$	"Smooth" as specified by interpolation kernel H		Arbitrary
Polynomial	$2\lvert K \rvert$ Motion-adaptive	$d(x) = \sum_{(i,j) \in K} \begin{pmatrix} a_{ij} \\ b_{ij} \end{pmatrix} x^i y^j$	"Smooth" as specified by K		Arbitrary

The first four models are illustrated in figure 5.10. The simplest, translational, model for 2D motion is used in the existing coding standards H.261, H.263 and MPEG family. It accounts for a rigid translational 3D motion under orthographic projection, resulting in a spatially constant 2D motion. Clearly, motion compensation with such fields preserves any 2D shape. This model was used in this work.

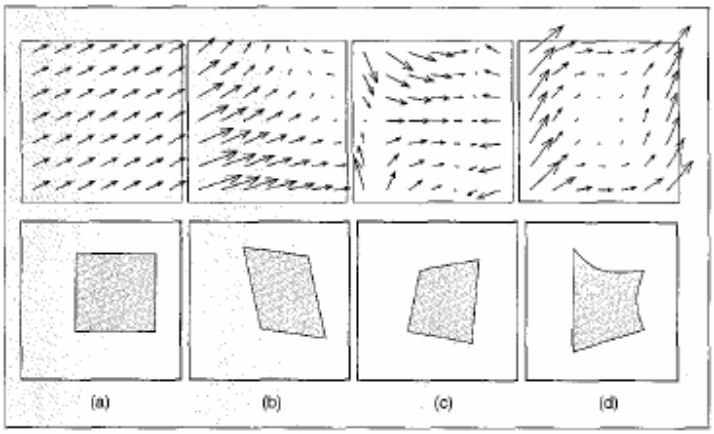


Figure 5.11: Examples of parametric motion vector fields & corresponding motion-compensated predictions of a center square: (a) Translational, (b) affine, (c) projective linear & (d) quadratic. The motion models are illustrated in table 5.1.

Region of Support for Motion Representation

As discussed in the preceding section, 2D motion in an image can be described spatially by a model form from table 5.1. Models differ in terms of the number of parameters and in terms of the functional dependence of $\mathbf{d(x)}$ on those parameters. In general, the higher the number of parameters, and thus the higher the function order, the more precise the description of the motion field. At the same time, however, an excessive number of parameters may result in motion "overmodeling" (excessive number of degrees of freedom – important in video processing & computer vision) and increased coding cost (important in video coding). In this case, the motion estimation accuracy may actually decrease. This is due to ill-posed ness of motion estimation; for example, no unique solution may exist. The precision of the motion field also depends on the region of support $R \subset \Lambda$ for the model, i.e. the set of image points to which the model applies. Since the true motion field $\tilde{\mathbf{d}}$ is rarely purely translational or divergent or exhibits other regularity, the smaller the region of support R, the better the approximation. The quality of approximation for a given motion field $\mathbf{d}$ can be measured, for example, by mean-square error

$$E_d^2 = \frac{1}{|R|}\sum_{\mathbf{x}\in R}\left\|\tilde{\mathbf{d}}(\mathbf{x}) - \mathbf{d}(\mathbf{x})\right\|^2 \quad \text{(Equation 5.15)}$$

Thus, for a given number of parameters the precision of a motion field can be adjusted by choosing a suitable region of support. Unfortunately, the mean-square error can be measured only if $\tilde{\mathbf{d}}$ is known, i.e. for computer generated images. There are different support regions with both fixed and variable size. There are four kinds of region of support:

1. Global Motion.
2. Motion of Individual Image Points.
3. Motion of Regions.
4. Hierarchical Motion Models.

A brief description of the above kinds of region of support follows.

Global Motion: The most constrained, yet simplest case is global motion, motion such that all image points are displaced in a uniform manner. The region of support for such models consists of the whole image

$$R = \Lambda \quad \text{(Equation 5.16)}$$

where it is assumed that the sampling grid Λ is an orthogonal lattice: $\Lambda = \{1,\ldots,K\}\times\{1,\ldots,L\}$ with K, L being the number of columns and lines in the image. See figure 5.12a.

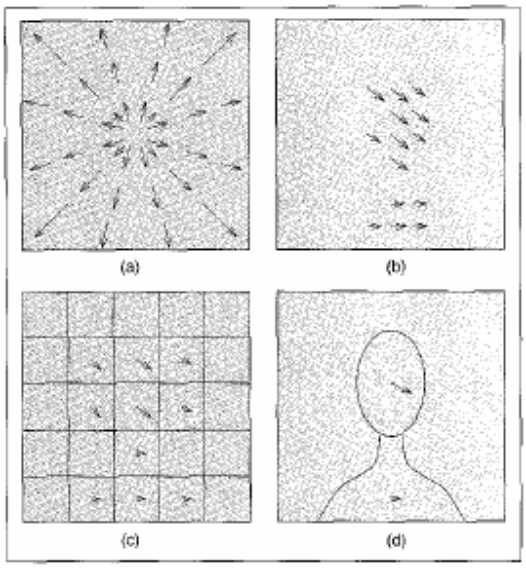


Figure 5.12: Various regions of support for a motion model: (a) global, (b) dense, (c) block-based and (d) region-based. The implicit underlying scene is of "head-and-shoulders" type as captured by the region based model.

Motion of Individual Image Points: At the other extreme of the spectrum, the region of support may consist of a single image point:

$$R_x = \{\mathbf{x}\}, \mathbf{x} \in \Lambda \quad \text{(Equation 5.17)}$$

Then, motion of each image point can be described by a set of parameters, such as displacement in the case of linear motion, or velocity in the case of quadratic trajectories. The pixel-based or dense motion representation is the least constrained one since at least two parameters describe movement of each image point, and thus at least $2\times K\times L$ parameters are used to present motion in an image. See figure 5.12b.

Motion of Regions: Between the two extremes above, one can find methods that apply motion models from table 5.1 to image regions. The motivation is to ensure a more accurate modeling of motion fields than in the global motion case and a reduced number of parameters in comparison with the dense motion. The simplest image partitioning is into non-overlapping rectangular regions R_{mn} of fixed size $B_K \times B_L$, referred to as blocks, whose union covers the whole image:

$$R_{mn} = \left\{ \mathbf{x} = (i, j)^T \in \Lambda : (m-1) B_K < i \le m B_K, (n-1) B_L < j \le n B_L \right\} \quad \text{(Equation 5.18)}$$
$$m = 1, \ldots, K / B_K, n = 1, \ldots, L / B_L$$

Block partitioning with simple translational motion is used today in all digital video compression standards such as H.261, H.263 and MPEG family. See figure 5.12c.

For better results a general image partitioning is necessary. The reasoning is that for objects with sufficiently smooth 3D surface and 3D motion, the induced 2D motion fields in the image plane can be suitably described by models from table 5.1 if applied to the area of object projection. A natural image partitioning can be provided by the image acquisition process itself. Because several 3D objects typically move in front of a camera, it is straightforward to group all pixels arising from one surface of a 3D object into one region. It is more interesting to find image partitioning such that all image points in a region arise from objects that undergo one motion. Then motion parameters can be estimated from all the image points in a moving region. In both cases, a region is described as follows

$$R_n = x_n \subset \Lambda \quad \text{(Equation 5.19)}$$

where all arbitrarily shaped regions x_n are non-overlapping and their union covers the complete image.

Hierarchical Motion Models: The practical concept of a variable size block for motion models can be regarded as a special case of hierarchical representation that has often been exploited in computer vision applications. In such a representation, the estimate (in this case motion) can be modeled at multiple levels of detail, making it possible to extract coarse characteristics first and add finer details later. In figure 5.13, we show a multiresolution representation of a motion field in dual form. On the left, a motion field is represented at multiple resolutions and scales at the same time. On the right, is shown an equivalent representation that can be obtained from the left representation by upsampling and interpolation. This representation is at multiple resolutions, but at a single scale.

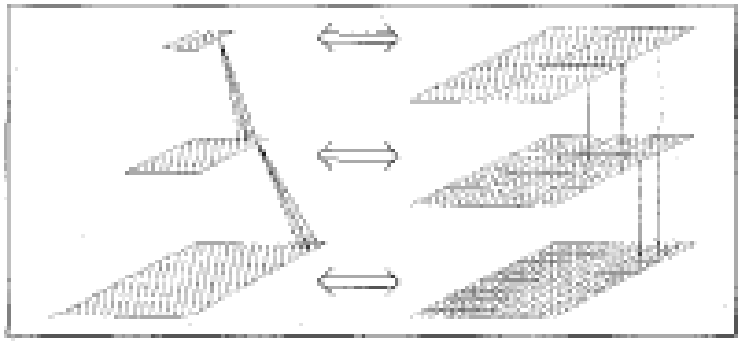


Figure 5.13: Dual representation of a motion field at multiple resolutions: at multiple scales (left) and at a single scale (right). The representations are equivalent since one can be obtained from the other by filtering/downsampling or upsampling/interpolation operators.

Interdependence of Motion and Image Data

At the very essence of every motion estimation algorithm lie assumptions about the relationship between motion parameters and image intensity. Let $g_t(\mathbf{x})$ be the image intensity at position $(\mathbf{x}, t)$. The usual and reasonable assumption made is that image intensity remains constant along the motion trajectory. This assumption implies, among others, that any intensity change is due to motion, that scene illumination is constant and that object surfaces are opaque. Although these constraints are almost never satisfied exactly, the constant intensity assumption approximately describes the dominant properties of natural image sequences and motion estimation methods based on it work well.

Let s be a variable along a motion trajectory. Then, the constant intensity assumption translates into the following constraint equation

$$\frac{dg}{ds} = 0 \quad \text{(Equation 5.20)}$$

By applying a chain rule, the above equation can be written as the *motion constraint equation*

$$\frac{\partial g}{\partial x}u + \frac{\partial g}{\partial y}v + \frac{\partial g}{\partial t}u = (\nabla g)^T \mathbf{v} + \frac{\partial g}{\partial t} = 0 \quad \text{(Equation 5.21)}$$

where

$$\nabla = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y} \right)^T \quad \text{(Equation 5.22)}$$

denotes the spatial gradient and $\mathbf{v} = (u, v)^T$ is the velocity. The above constraint equation, whether in the continuous form or as a discrete approximation, has recently served as the basis of many algorithms estimating linear motion. Note that equation 5.21 applied at one position $(\mathbf{x}, t)$ is underconstrained, since it only determines the component of velocity $\mathbf{v}$ in the direction of image gradient. Due to this aperture problem, additional constraints must be used to uniquely solve for velocity.

Estimation Criteria

Various motion representations as well as the relationship between motion and images discussed in the previous section can be used now to formulate an estimation criterion. There is no unique criterion for motion estimation. The difficulty in establishing a good criterion is primarily caused by the fact that motion in images is not directly observable and that particular dynamics of intensity in an image sequence may be induced by more than one motion. Another problem is that most of the models discussed above are far from ideal. Therefore, all attempts to establish suitable criteria for motion estimation require further implicit or explicit modeling of the image sequence. There are four kinds of estimation criteria:

1. DFD-Based Criteria.
2. Frequency Domain Criteria.
3. Regularization.
4. Bayesian Criteria.

A brief description of the above kinds of estimation criteria follows.

DFD-Based Criteria: An important class of criteria arising from the constant-intensity assumption aims at the minimization of the following error

$$e_{t,t}(x) = g_t(\mathbf{x}) - \hat{g}_{t,t}(\mathbf{x}), \forall x \in R \quad \text{(Equation 5.23)}$$

where

$$\hat{g}_{t,t}(\mathbf{x}) = g_t(\mathbf{x} + \mathbf{d}_{t,t}(x)) \quad \text{(Equation 5.24)}$$

is called a motion compensated prediction of $g_t(x)$. If R is a complete image ($R = \Lambda$), this error is called a *displaced frame difference* (DFD). However, when R is a block or an arbitrary shaped region, the corresponding error is called a *displaced block difference* or a *displaced region difference*.

Motion fields calculated solely by minimization of the magnitude of the prediction error (Equation 5.23) are, highly sensitive to noise if the number of pixels in the region of support R is not large compared to the number of motion parameters estimated, or if the region is poorly textured. However, such a minimization may yield good estimates for parametric motion models with few parameters and a reasonable region size.

To measure the magnitude of the prediction error e, a common choice is an L_p norm. For the L_2 norm, this corresponds to the mean-squared motion compensated prediction error:

$$J_1(\mathbf{d}) = \sum_{x \in R} \left(g_t(\mathbf{x}) - g_t(\mathbf{x} + \mathbf{d}(\mathbf{x})) \right)^2 \quad \text{(Equation 5.25)}$$

This criterion, although very often used, is unreliable in the presence of outliers; even for a single large error $e(x)$, $e^2(x)$ is very large and by overcontributing to J_1 it biases the estimate of $\mathbf{d}$. Therefore, a more robust mean absolute error criterion

$$J_2(\mathbf{d}) = \sum_{x \in R} \left| g_t(\mathbf{x}) - g_t(\mathbf{x} + \mathbf{d}(\mathbf{x})) \right| \quad \text{(Equation 5.26)}$$

is the criterion of choice in practical video coders today. This criterion is less sensitive to bias due to the piecewise linear dependence of J_2 on e, and at the same time is less involved computationally. There are two more DFD-based criteria, the median squared error criterion

$$J_3(\mathbf{d}) = \underset{x \in R}{med} \left(g_t(\mathbf{x}) - g_t(\mathbf{x} + \mathbf{d}(\mathbf{x})) \right)^2 \quad \text{(Equation 5.27)}$$

and a criterion based on the Lorentzian function

$$J_4(\mathbf{d}) = \sum_{x \in R} \log\left(1 + \left(g_t(\mathbf{x}) - g_t(\mathbf{x} + \mathbf{d}(\mathbf{x})) \right)^2 / 2s^2 \right) \quad \text{(Equation 5.28)}$$

Frequency Domain Criteria: Another class of criteria for motion estimation uses transforms, such as Fourier transform. For example, due to its shift property, the 2D Fourier transform of an image undergoing spatially constant motion.
Regularization: Instead of dealing with the unconstrained nature of equation 5.21 by restricting the motion model to a few parameters, another approach is to explicitly model additional constraints. This can be done by a weak constraint on the estimate itself, reflecting the empirical observation that typical motion fields are spatially smooth.
Bayesian Criteria: A general framework for motion-field estimation is provided by Bayesian methods.

Search Strategies

With models expressing out knowledge about motion and images specified, and an estimation criterion selected, what remains is to identify an estimation procedure. This procedure involves an optimization of the selected criterion with respect to the parameters of the chosen model. For dense motion fields, both the number of unknowns and the state space for each of them may be large as their state spaces; an exhaustive search over the complete state space is, with rare exception, computationally prohibitive. Below, we briefly discuss some search strategies.

Matching: For a small number of motion parameters and a small state space, the most common search strategy, when minimizing a prediction error, is matching. In this approach, motion compensated predictions for various motion candidates are compared with the original image within the region of support of the motion model. The candidate yielding the best match for a given criterion becomes the optimal state. For small state spaces, as is the case in block-constant motion models, the full state space of each motion vector can be examined. This leads to exhaustive-search block matching. Figure 5.14 shows the principles of block matching estimation.

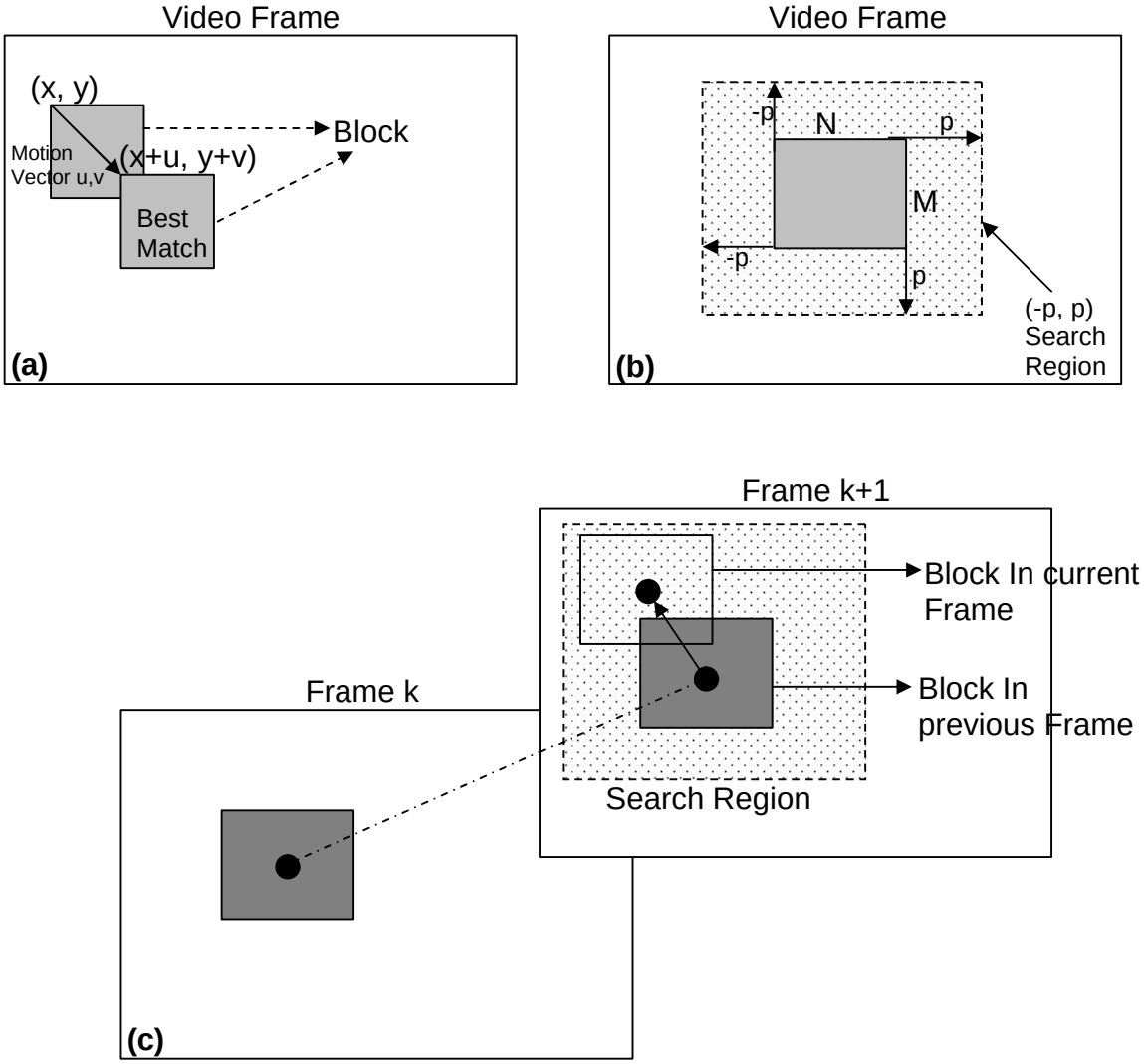


Figure 5.14: The three figures illustrate the principles of block-matching estimation. (a) Find one displacement vector for each block, (b) Within a search range, find a best "match" that minimizes an error measure such as $J_2(\mathbf{d}) = \sum_{x \in R} \left| g_t(\mathbf{x}) - g_t(\mathbf{x} + \mathbf{d}(\mathbf{x})) \right|$ and (c) Shows the above principles using a point (small black hole) in two video frames. The *exhaustive-search block matching* algorithm compares all possible displacements within the search range, for that reason is computationally expensive.

Relaxation: For dense motion fields based on a non-causal model, simultaneous optimization of all parameters may be computationally prohibitive. To alleviate the problem, relaxation techniques construct a sequence of estimates such that consecutive estimates differ in one variable at most.

HCF Method: Another deterministic optimization technique for Markov random fields that update a single site in each step is the *highest confidence first* (HCF) algorithm. In contrast to relaxation schemes, its site visiting schedule is not fixed, but is driven by the input data. Initially, all the sites are marked as "uncommitted". At the beginning, the HCF algorithm selects sites with a "peaked" likelihood function, which is typically the case for highly structured regions. Later, the algorithm includes more and more sites that may not possess such an ideal likelihood function, and thus builds on the neighborhood information of already estimated sites. Since only variables at committed sites influence the optimization, and initially all the sites are uncommitted, the estimated field is independent of the initial state.

Gradient-Based Optimization: Gradient-based techniques require an estimation criterion $J(d)$ that is differentiable. Because this criterion depends on motion parameters via the image function g, such as in equation 5.25, it is usually approximated by a Taylor expansion with respect to motion parameters. Then, the differentiation of the Taylor approximated criterion involves differentiation and interpolation of image intensities. Due to the Taylor approximation, the model is applicable only in a small vicinity of the desired motion estimate.

Mean-Field Techniques: Much work on the theoretical analysis of Gibbs/Markov random fields has been performed in equilibrium statistical mechanics. Mean-field approaches have proven a powerful tool for the approximation of the mean of each field. The motivation for mean-field techniques is based on the important result from statistical mechanics stating that mean values of a Gibbs/Markov random field can be obtained from its partition function. For this purpose, the partition function is considered to be a function of the data. Therefore, mean-field approaches first formulate the desired mean-field through the partition function and then approximate the partition function by assuming that this sum is governed by realizations near the equilibrium state. Then one can benefit from the property that typical optimization criteria exhibit fewer local optima at higher temperatures. Hence, one can design deterministic optimization procedures that find initial estimates at high temperatures and improve them by decreasing the temperature (annealing).

Hierarchical Optimization: The search strategies presented in the preceding paragraphs are often computationally expensive. To lower this computationally burden, the hierarchical motion representations discussed in "Hierarchical Motion Models" are often exploited. In the multiresolution/multiscale approach (figure 5.13, left), the motion field is represented over a multiresolution pyramid. In the multiresolution/multiscale motion estimation, motion parameters are

computed at the lowest resolution first. The computational load of this task is low as compared to the estimation at full resolution because the dimension of the state space of motion vector fields is reduced by 2^{2v} and the amplitude of motion is reduced by 2^{v}. Also, due to the scale change between levels of the motion pyramid, methods based on a spatial smoothness constraint converge much faster than their non-hierarchical counterparts. Consequently, a coarse estimate is found very rapidly at the highest level, especially by fast schemes such as the deterministic relaxation. By a suitable projection, this estimate is decreased in scale to serve as an initial state for the motion estimate at the next lower level of the pyramid. More detailed information is added at this level by the same or another optimization scheme. This procedure is repeated until an estimate at the lowest level of the pyramid is found.

In this work the motion model which was used is the first one in the table 5.1, translational. Motion of Regions was selected as region of support. For estimation criteria the DFD-based criteria was selected with equation 5.26 as error criterion. Finally, the exhaustive-search block matching algorithm was selected as a search strategy. All the models and methods mentioned before are used today in all digital video compression standards such as H.261, H.263 and MPEG family.

As mentioned before the two methods, which will be presented below, use the histogram of the angle of motion vectors. The size of each block in a frame is 16x16 (Width x Height) pixels. The number of motion vectors in each frame is stable, for example if we have a video with frame size 352x240 (Width x Height) pixels, the number of motion vectors in each frame will be 330 =22x15 (Width x Height). In figure 5.15 some information about motion vector and histogram can be seen.

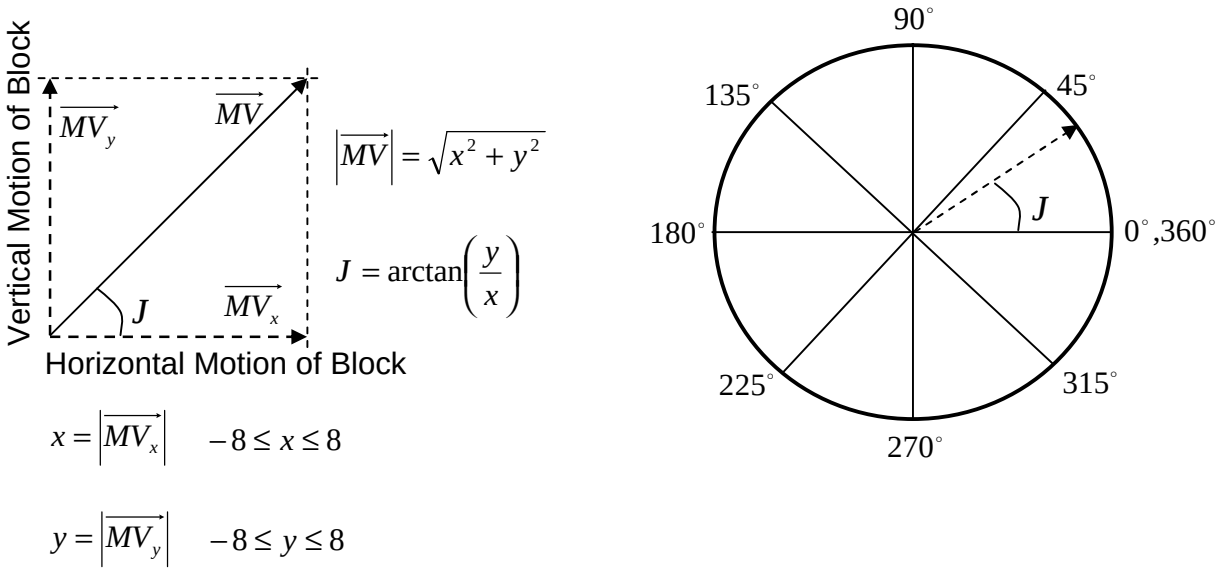


Figure 5.15: In the left side of this figure is a motion vector $\overrightarrow{MV}$ and how is produced by its horizontal and vertical component ($\overrightarrow{MV_x}$, $\overrightarrow{MV_y}$). The length of the vector is given by $\left|\overrightarrow{MV}\right| = \sqrt{x^2 + y^2}$ and the angle by $J = \arctan\left(\frac{y}{x}\right)$ in radians (in degrees the formula is $J^\circ = \frac{180}{p}\arctan\left(\frac{y}{x}\right)$). In the right side of this figure is the trigonometric circle divided in eight regions (eight basic directions that a block can move). The division in eight regions was made to help us out with the calculation of histogram of motion vector's angle. In other words, we use eight bins to calculate histogram and we do not use 360 bins which is unwieldy.

Interpretation of Histogram of Motion Vector's (MV) Angle

Motion Vectors (MV) are divided in two categories, *Static* MVs and *Dynamic* MVs. Static MVs are those of which the length is smaller than one, $\left|\overrightarrow{MV}\right| < 1$. Their length is so small that do not affect the total motion. Dynamic MVs are those MVs with length bigger than one, $\left|\overrightarrow{MV}\right| \ge 1$. The maximum length of a MV is $\left|\overrightarrow{MV}\right|_{\max} = \sqrt{8^2 + 8^2} \Leftrightarrow \left|\overrightarrow{MV}\right|_{\max} = 8\sqrt{2} \approx 11.314$ if search region is (-8, 8). Figure 5.16 shows a typical histogram of MV angles.

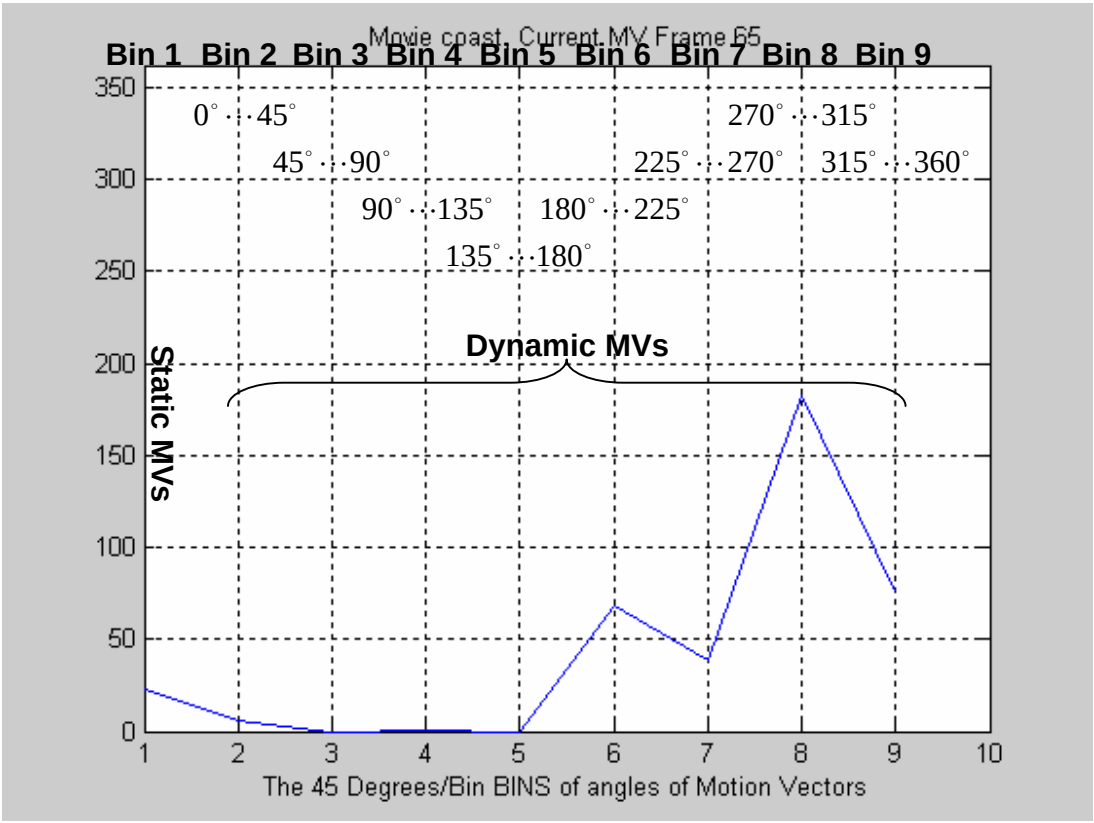


Figure 5.16: This figure illustrates a histogram of MV angles. We divide the histogram in two regions. In the first region we can see the number of Static MVs. The first region is Bin 1. The second region is from Bin 2 to Bin 9. This region shows the number of dynamic MVs according to their angle.

Next follows some typical exams of camera panning, tilting, zoom-in and zoom-out and how the histogram behaves. Figure 5.17 illustrates a camera pan and figure 5.18 illustrates a camera tilt, zoom-in and zoom-out.

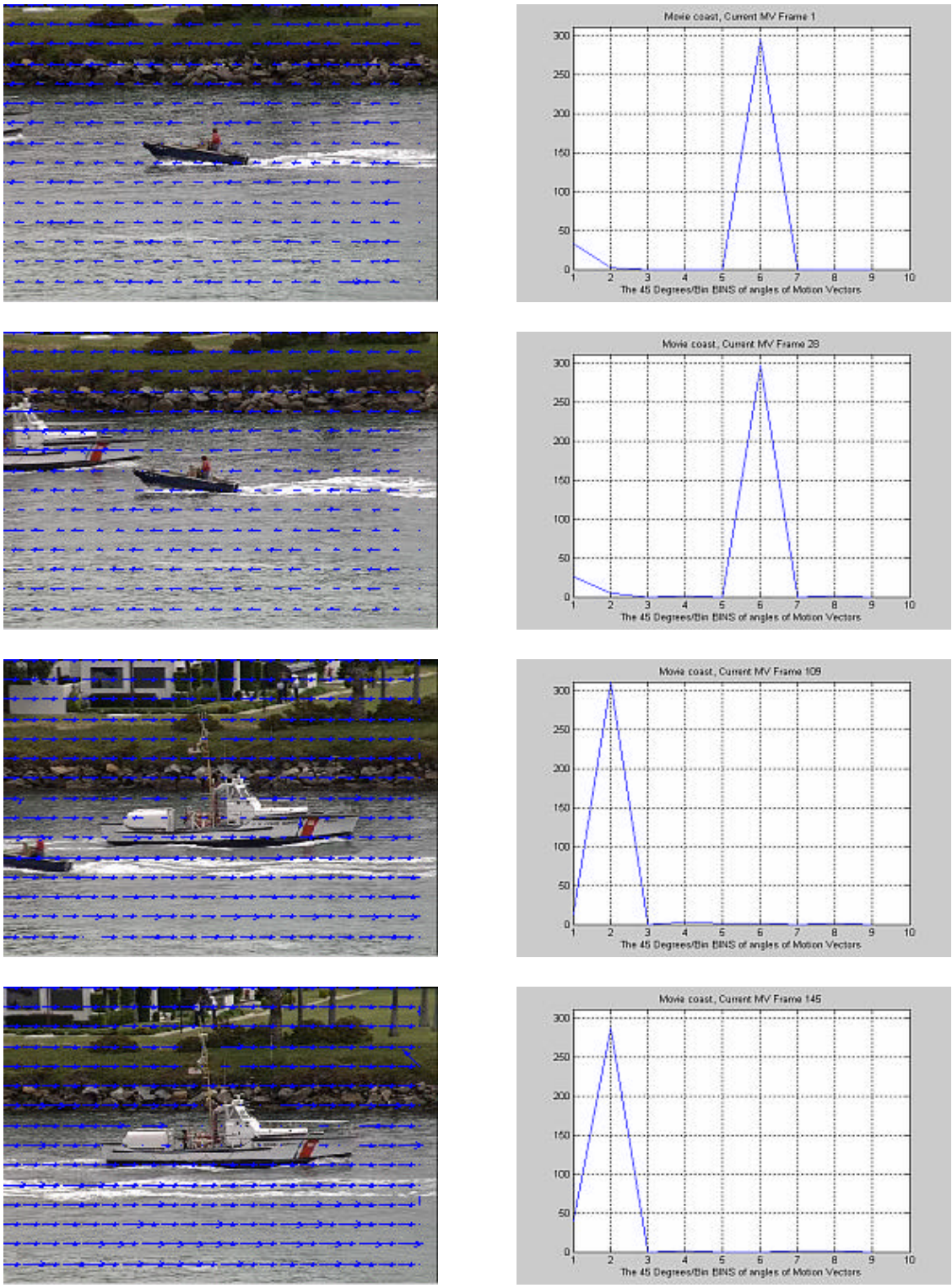
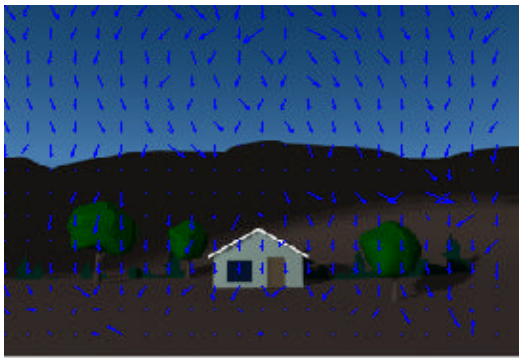
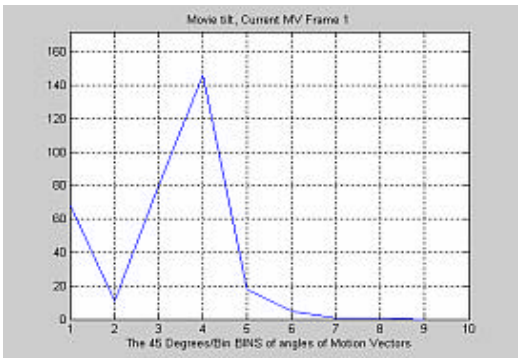
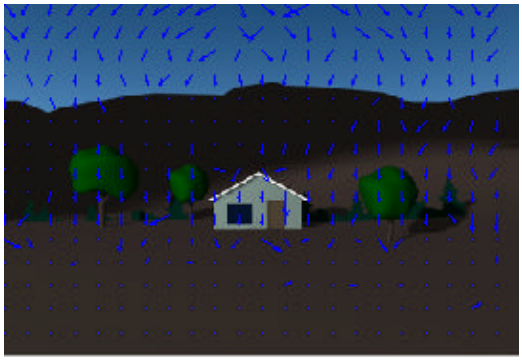
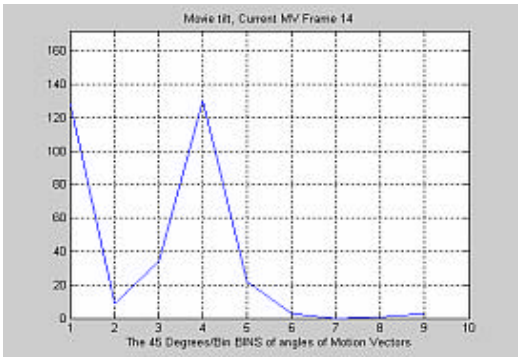


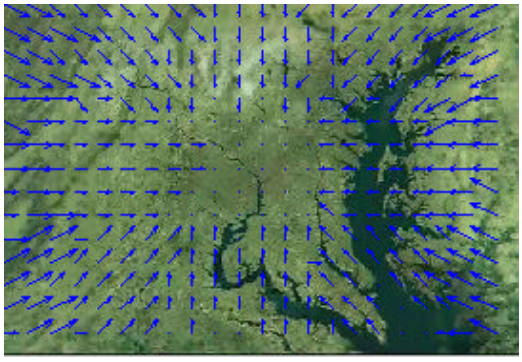
Figure 5.17: In the left column there are four frames from video “coast guard” (first two frames a camera pan to the left & next two to the right) with their motion vectors. In the right column there are the histograms of MV angles for each frame. We observe that when we have movement to the left, there is a peak in bin 6 (180^o-225^o). When we have a movement to the right, there is a peak in bin 2 (0^o-45^o).

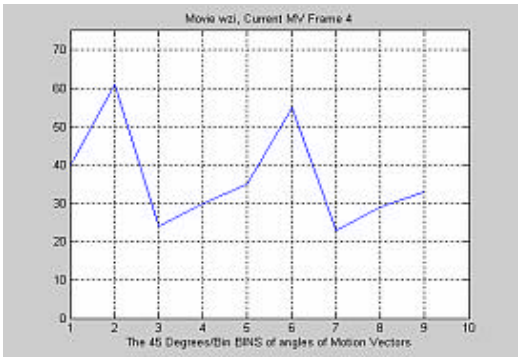


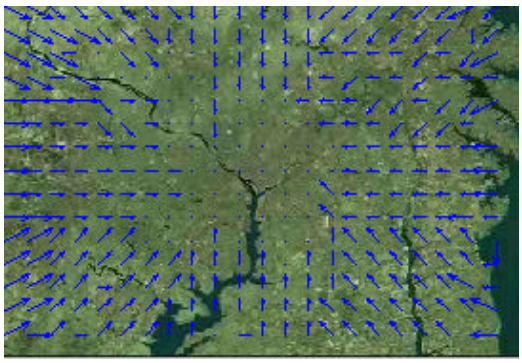

Movie tilt, Current MV Frame 1
The 45 Degrees/Bin BINS of angles of Motion Vectors

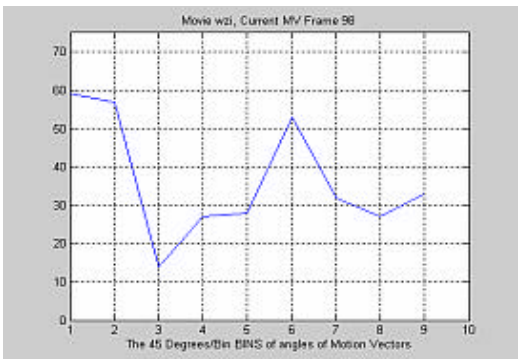



Movie tilt, Current MV Frame 14
The 45 Degrees/Bin BINS of angles of Motion Vectors




Movie wzi, Current MV Frame 4
The 45 Degrees/Bin BINS of angles of Motion Vectors




Movie wzi, Current MV Frame 98
The 45 Degrees/Bin BINS of angles of Motion Vectors

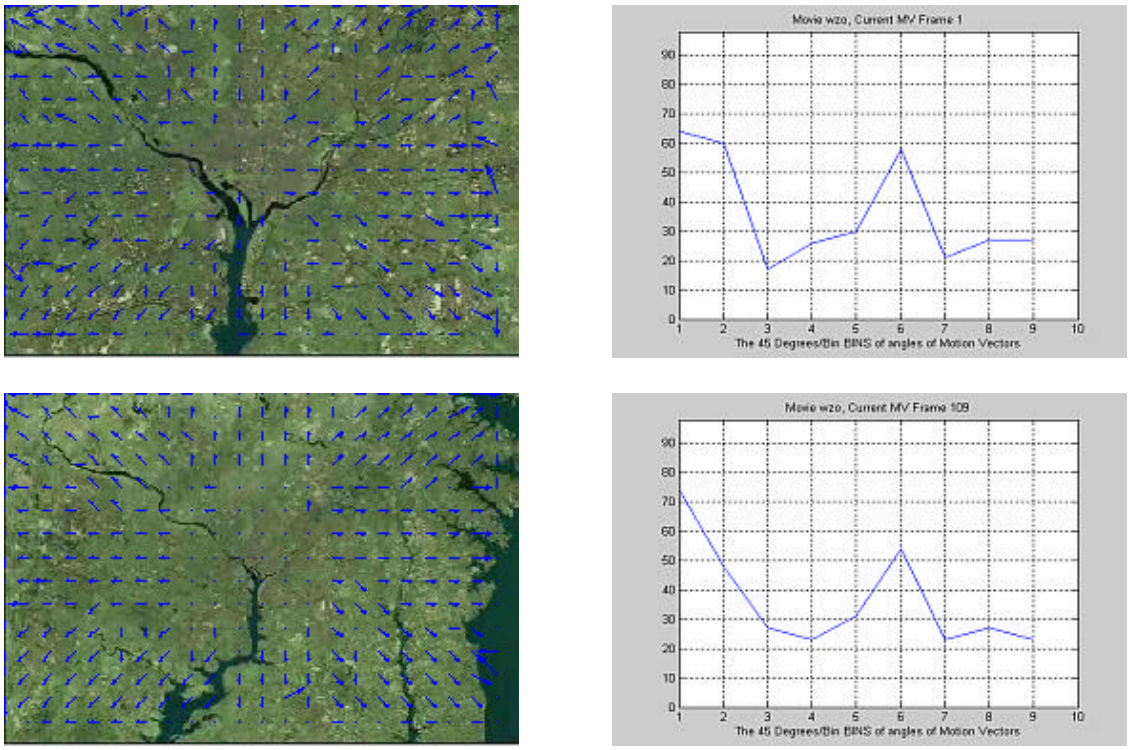
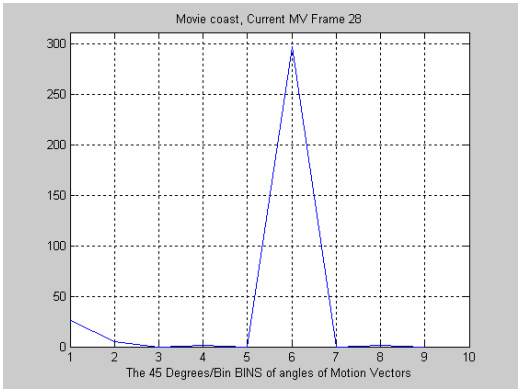
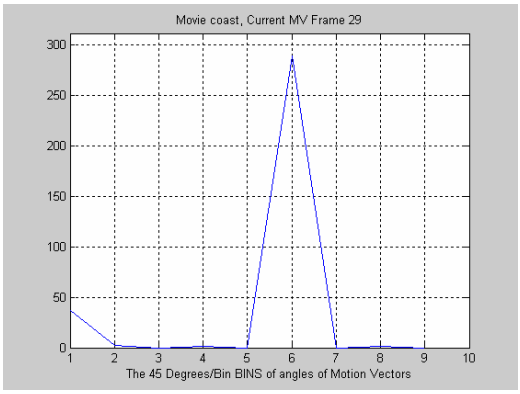
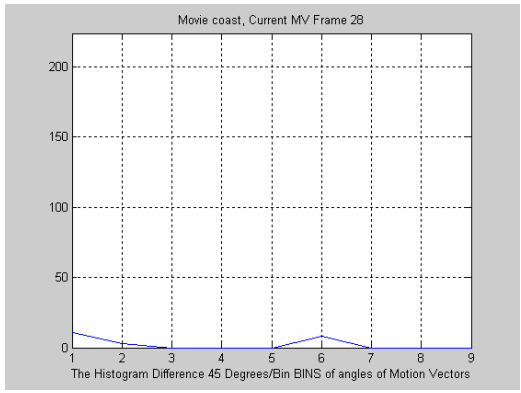


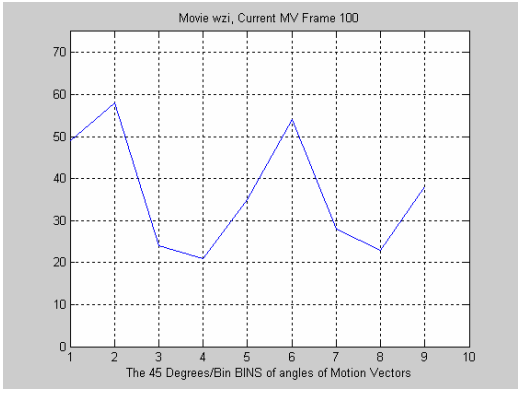
Figure 5.18: In the left column there are two frames from a camera tilt, two frames from a zoom-in (from NASA's satellite) and two frames from a zoom-out (from NASA's satellite). In the right column there are the histograms of MV angles for each frame. We observe that in camera tilt, we have a peak in bin 4 (90^o-135^o). In zoom-in and zoom-out the histograms are almost the same, tend to be flat.

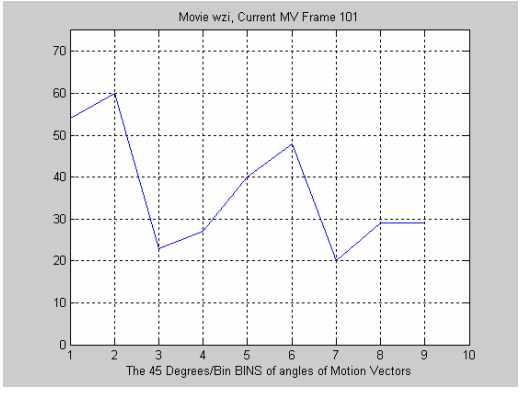
The following figure contains histogram differences of several videos. We can observe that the histogram difference behaves the same when we have a constant motion no mater what motion we have (tend to be a flat line). So we cannot draw safe conclusions from a histogram difference about the kind of motion. In a histogram difference, we can see clear the change of the motion. All the above are illustrated in figure 5.19.

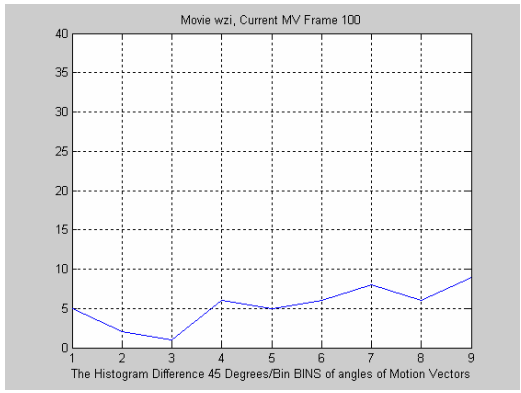

Movie coast, Current MV Frame 28
The 45 Degrees/Bin BINS of angles of Motion Vectors


Movie coast, Current MV Frame 29
The 45 Degrees/Bin BINS of angles of Motion Vectors


Movie coast, Current MV Frame 28
The Histogram Difference 45 Degrees/Bin BINS of angles of Motion Vectors


Movie wzi, Current MV Frame 100
The 45 Degrees/Bin BINS of angles of Motion Vectors


Movie wzi, Current MV Frame 101
The 45 Degrees/Bin BINS of angles of Motion Vectors


Movie wzi, Current MV Frame 100
The Histogram Difference 45 Degrees/Bin BINS of angles of Motion Vectors

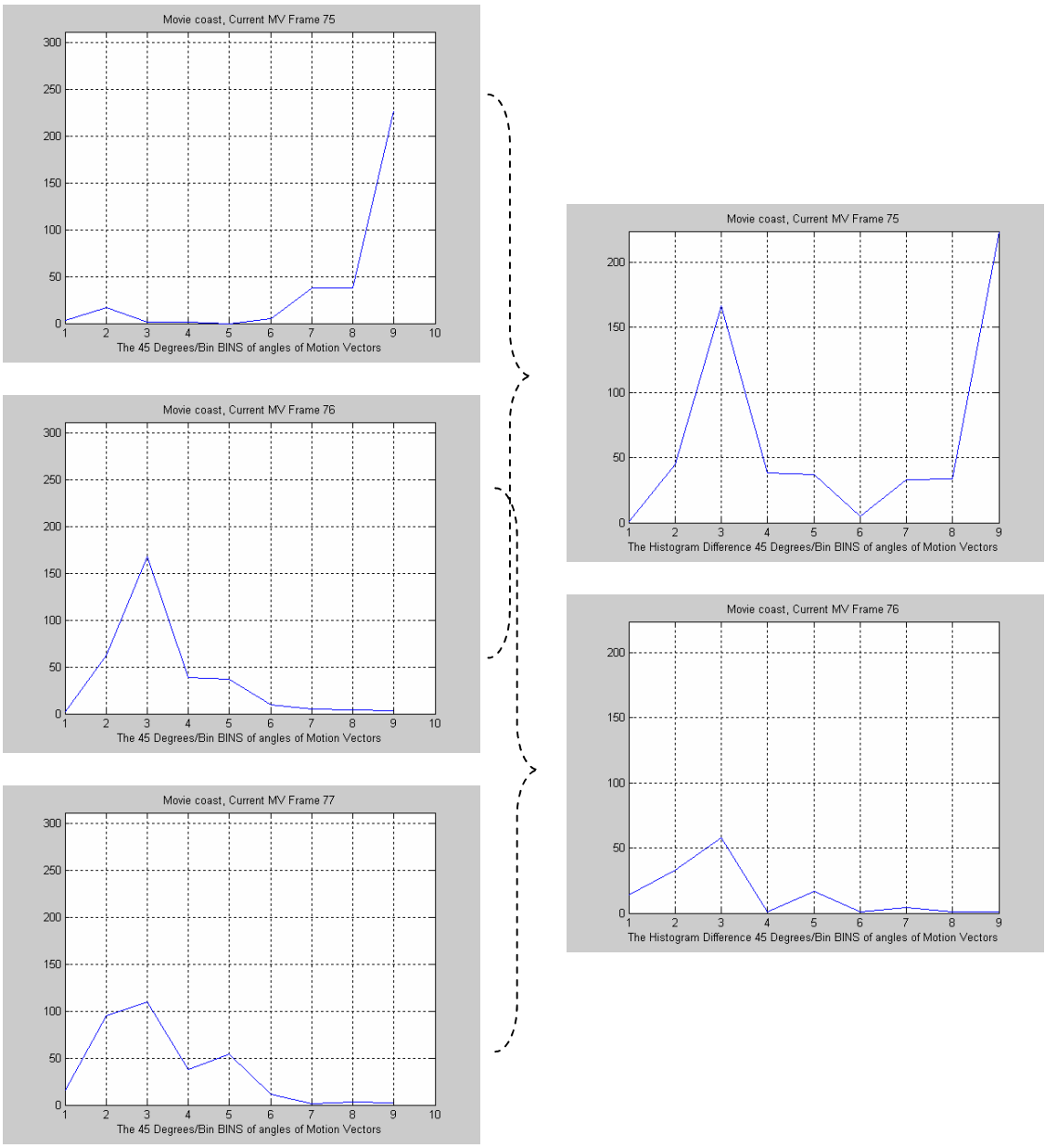
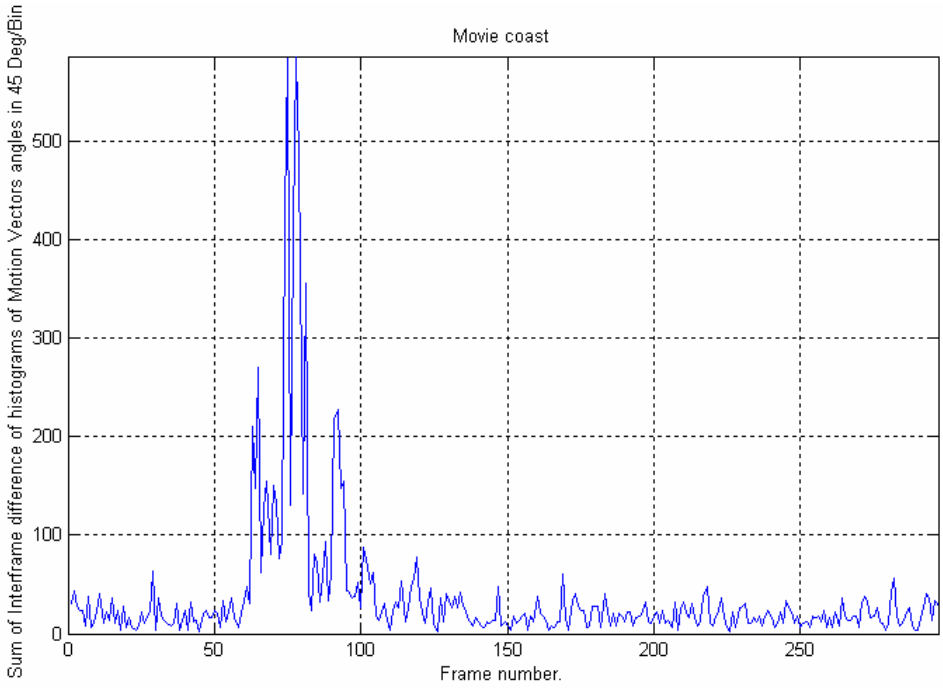
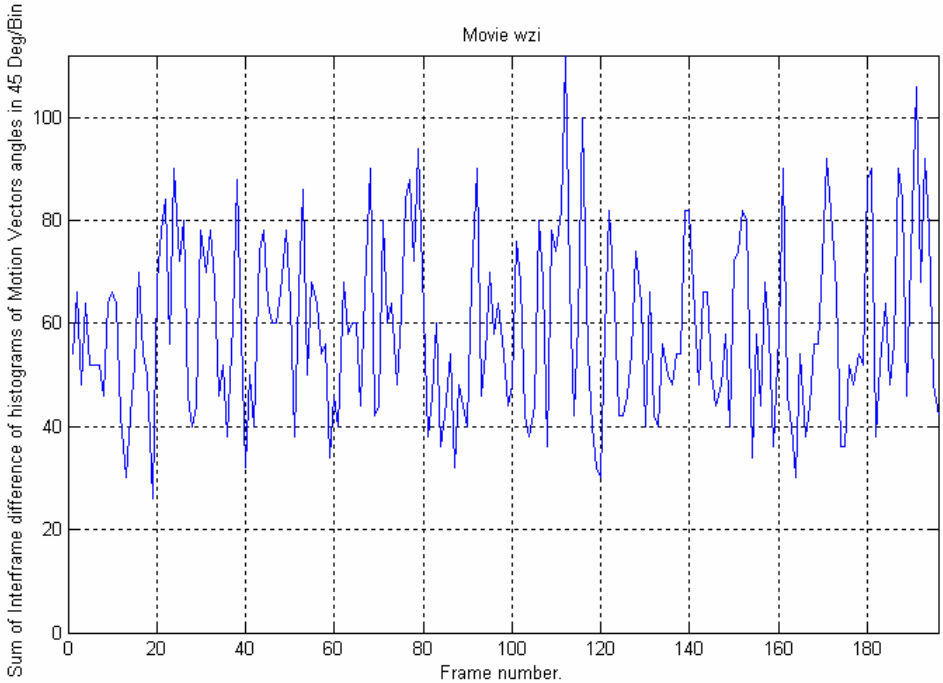


Figure 5.19: In the left column there are two histograms of consecutive frames and in the right column there are their histogram differences. The first two histograms are from a camera pan and the next two are from a zoom-in. We observe that in these two kinds of motion, the histogram differences tend to be flat because we have a constant motion without change. The final three histograms are from a camera pan which changes direction. We can observe the distinct difference of histogram differences in the right column.

Finally, before the two methods will be presented, in figure 5.20 there are some typical sum of histogram differences.


Movie coast
Sum of Interframe difference of histograms of Motion Vectors angles in 45 Deg/Bin
0
100
200
300
400
500
0
50
100
150
200
250
Frame number.


Movie wzi
Sum of Interframe difference of histograms of Motion Vectors angles in 45 Deg/Bin
0
20
40
60
80
100
0
20
40
60
80
100
120
140
160
180
Frame number.

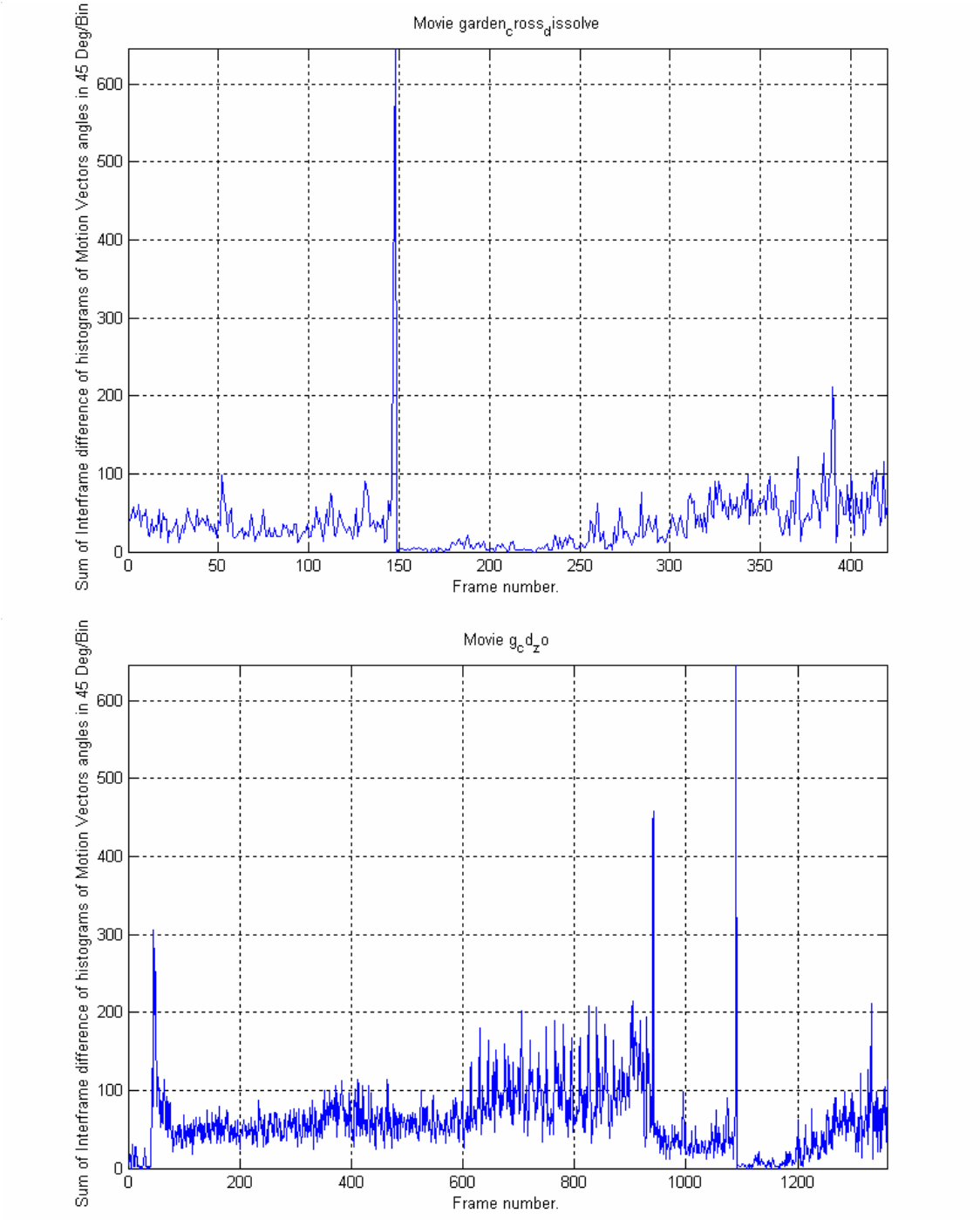


Figure 5.20: The first graph has a camera pan, the second has a zoom-in, the third has a camera pan with a cross dissolve and the fourth has no motion, camera pan, zoom-out and cross dissolve.

Sliding Window Method for MVs

Sliding Window (SW) approach for shot detection

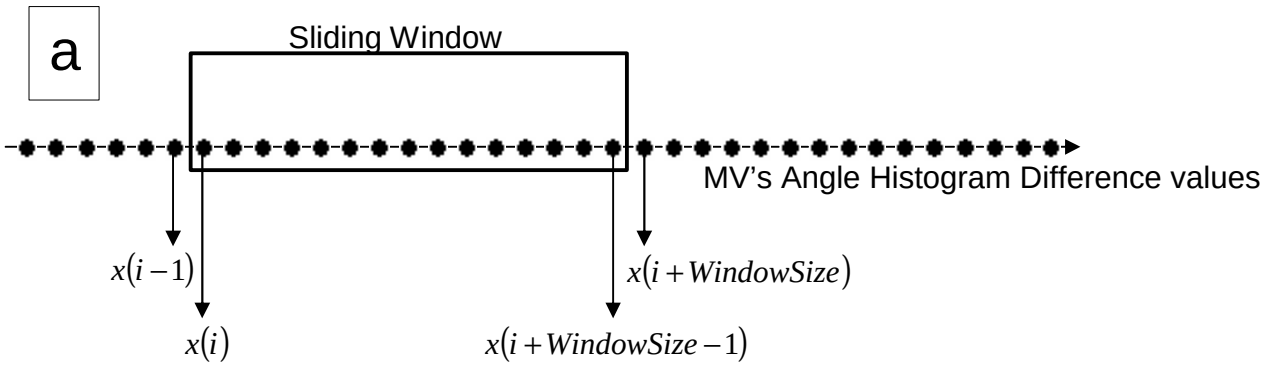
As mentioned before the metric which will be used for SW approach, will be the MV's angle histogram difference. Some example graphs can be seen in figure 5.20. Next follows how SW approach is used with MV's angle histogram difference.

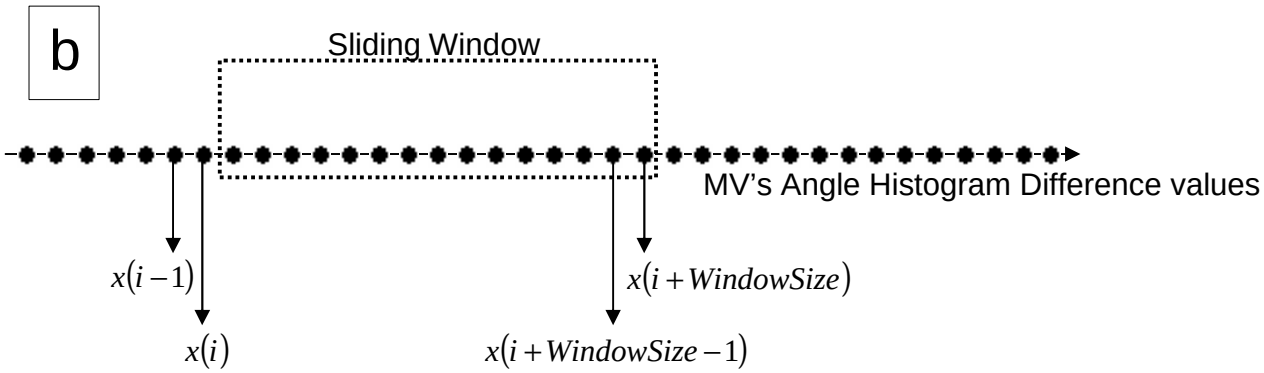
Before the explanation of the SW algorithm, it is necessary to give some notations. The sliding window which is used has a size of **WindowSize** frames. The value of **WindowSize** is standard to 15 which do not change during the process of the entire video. $x(i)$ is the frame-to-frame MV's angle histogram difference value of i-th frame. The video has a size of **TotalFrames** frames. So we have for variable i, $0 \le i \le (TotalFrames - 2)$ (we have *TotalFrames-2* because for two frames we have a MV field). $m_W{}^{MV}(j)$ is the mean value of the data within the sliding window. $s_W{}^{MV}(j)$ is the variance of the data within the sliding window. $T_W{}^{MV}(j)$ is the threshold of the data within the sliding window. The variable j denotes how many instances of the sliding window can be produced during the processing of the entire video. So the limits of variable j are $0 \le j \le (((TotalFrames - 1) - WindowSize) - 1)$.

We build a sliding window preceding the current frame. The next step is to calculate the mean $m_W{}^{MV}(j)$ and variance $s_W{}^{MV}(j)$ value of the histogram difference data in the window. The threshold $T_W{}^{MV}(j)$ is calculated by the formula $T_W{}^{MV}(j) = m_W{}^{MV}(j) + a_W{}^{MV} s_W{}^{MV}(j)$, where $a_W{}^{MV}$ is a constant variable for the entire video. The value of variable $a_W{}^{MV}$ is changing according to video type. The following step is to compare the threshold $T_W{}^{MV}(j)$ with the histogram difference value which is next to the right of the sliding window. In other words, if $x(i),\ldots,x(i + WindowSize - 1)$ are in the sliding window, the threshold $T_W{}^{MV}(j)$, which calculated from the above values, is compared to $x(i + WindowSize)$. If $T_W{}^{MV}(j) > x(i + WindowSize)$, the sliding window is moving one frame to the right, so $x(i)$ is getting out from the left of the window and $x(i + WindowSize)$ is getting in from the right of the window. If $T_W{}^{MV}(j) \le x(i + WindowSize)$, the sliding window is not moving and none value get in the window if the first condition, $T_W{}^{MV}(j) > x(i + WindowSize)$, is not satisfied. The value $x(i + WindowSize)$ is

declared as Camera Break. The value of $T_W{}^{MV}(j)$ will be the same while the condition $T_W{}^{MV}(j) \le x(i + WindowSize)$ is satisfied.

One threshold need to be calculated in SW approach in order to partition video according to his motion. The motion change, usually, happens in one frame. So, if the condition $T_W{}^{MV}(j) \le x(i + WindowSize)$ is satisfied for one frame, then we have a motion change. Some possible instances of sliding window can be seen in figure 5.21.





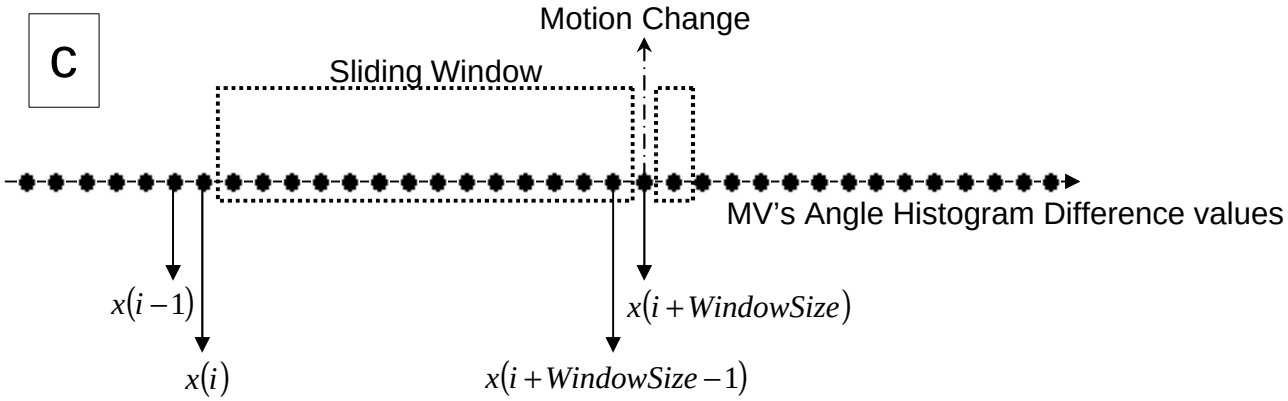
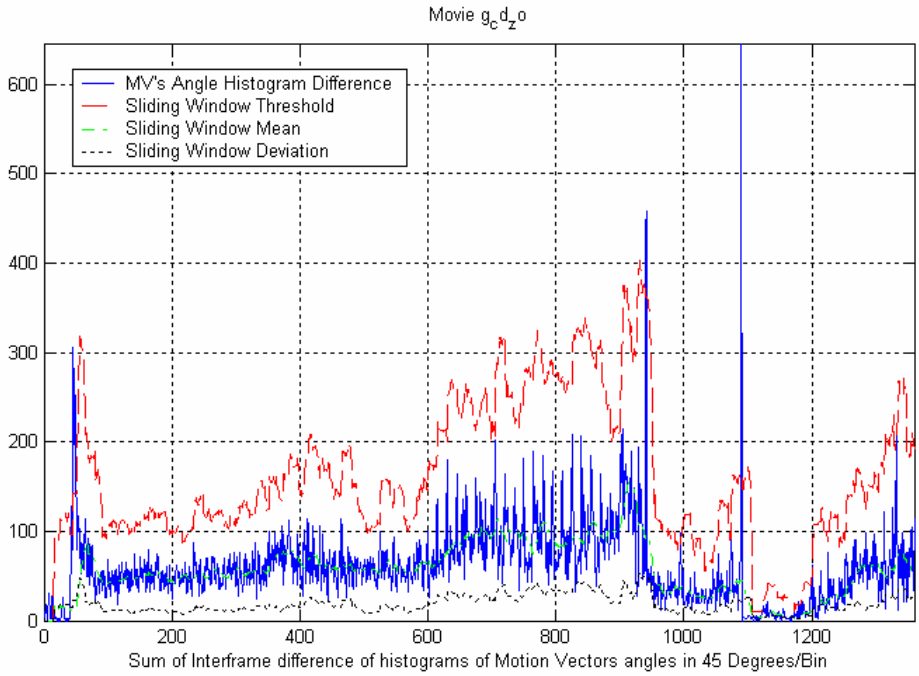
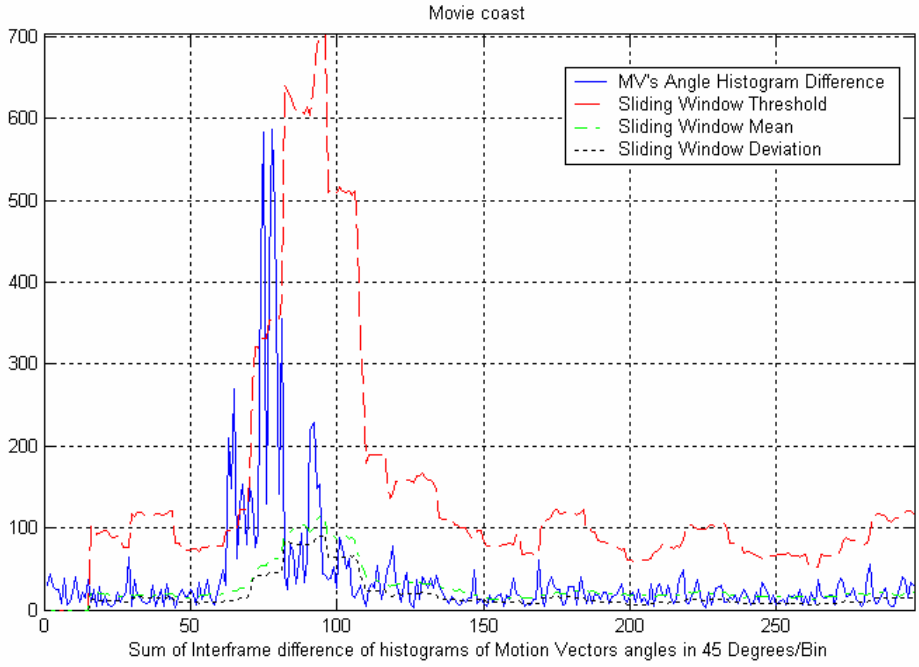


Figure 5.21: (**a**) Illustration of a sliding window instance j in MV's histogram difference graph. (**b**) Illustration of a sliding window instance $j+1$, if condition $T_W{}^{MV}(j) > x(i+WindowSize)$ is satisfied. (**c**) Illustration of a sliding window instance $j+1$, if condition $T_W{}^{MV}(j) \le x(i+WindowSize)$ is satisfied for one frame. In this case we have Motion Change.

Selection of Thresholds

As mentioned before, the threshold $T_W{}^{MV}(j)$ is calculated by the formula $T_W{}^{MV} = m_W{}^{MV} + a_W{}^{MV} s_W{}^{MV}$. The reason of selection this value for threshold is explained in TC approach. One threshold is used in SW approach because this threshold is adapted to the local variations of MV's histogram difference. In other words, the graph of the threshold follows the variations of the MV's histogram difference graph. An example of $m_W{}^{MV}, s_W{}^{MV}$ and $T_W{}^{MV}(j)$ graphs with histogram difference graph can be seen in figure 5.22.


Movie g_c d_z o
MV's Angle Histogram Difference
Sliding Window Threshold
Sliding Window Mean
Sliding Window Deviation
600
500
400
300
200
100
0
0
200
400
600
800
1000
1200
Sum of Interframe difference of histograms of Motion Vectors angles in 45 Degrees/Bin


Movie coast
MV's Angle Histogram Difference
Sliding Window Threshold
Sliding Window Mean
Sliding Window Deviation
700
600
500
400
300
200
100
0
0
50
100
150
200
250
Sum of Interframe difference of histograms of Motion Vectors angles in 45 Degrees/Bin

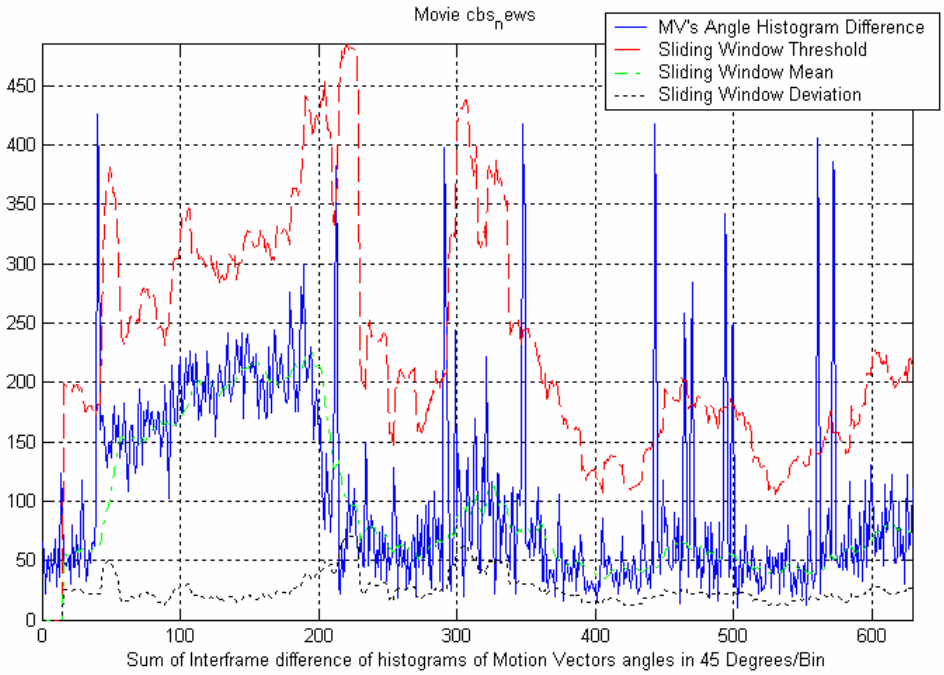


Figure 5.22: Three graphs, each one have the MV's angle histogram difference, the SW Threshold $T_W{}^{MV}(j)$ (Red line), the SW Mean $m_W{}^{MV}(j)$ (Yellow line) and SW Deviation $s_W{}^{MV}(j)$ (Black line). It is worth mention that the first 15 values of MV's histogram difference are used to calculate the first value of $m_W{}^{MV}(j)$, $s_W{}^{MV}(j)$ and $T_W{}^{MV}(j)$. Before the first value, we cannot calculate $m_W{}^{MV}(j)$, $s_W{}^{MV}(j)$ and $T_W{}^{MV}(j)$, so their values are zero.

Adaptive Method for MVs

Adaptive approach for shot detection

As mentioned before the metric which will be used for Adaptive approach, will be the MV's angle histogram difference. Some example graphs can be seen in figure 5.20. Next follows how Adaptive approach is used with MV's angle histogram difference.

Before the explanation of the Adaptive algorithm, it is necessary to give some notations. $x(i)$ is the frame-to-frame histogram difference value of i-th frame. The video has a size of **TotalFrames** frames. So we have for variable i, $0 \le i \le (TotalFrames - 2)$ (we have *TotalFrames-2* because for two frames we have a MV field). ${m_A}^{MV}(i)$ is the mean value of the data which is calculated by the adaptive algorithm. ${s_A}^{MV}(i)$ is the variance of the data which is calculated by the adaptive algorithm. ${T_A}^{MV}(i)$ is the threshold of the data which is calculated by the adaptive algorithm.

The calculation of adaptive mean value ${m_A}^{MV}(i)$ and adaptive standard deviation ${s_A}^{MV}(i)$ is being by the following formulas:

$$ {m_A}^{MV}(i) = {m_A}^{MV}(i-1) - c\left({m_A}^{MV}(i-1) - x(i)\right) \Leftrightarrow {m_A}^{MV}(i) = (1-c){m_A}^{MV}(i-1) + cx(i) $$

(Equation 5.28)

$$ {s_A}^{MV}(i) = \sqrt{{m_A}^{MV}(i)^2 - {l_A}^{MV}(i)} $$ (Equation 5.29)

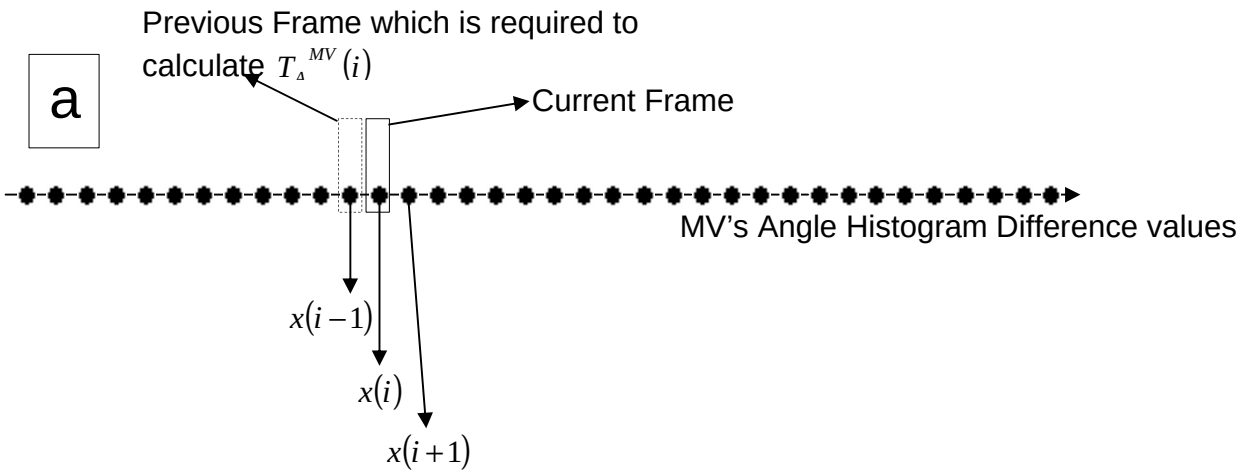
where c is a controlling coefficient $c \in [0,1]$ and ${l_A}^{MV}(i)$ is the adaptive second moment

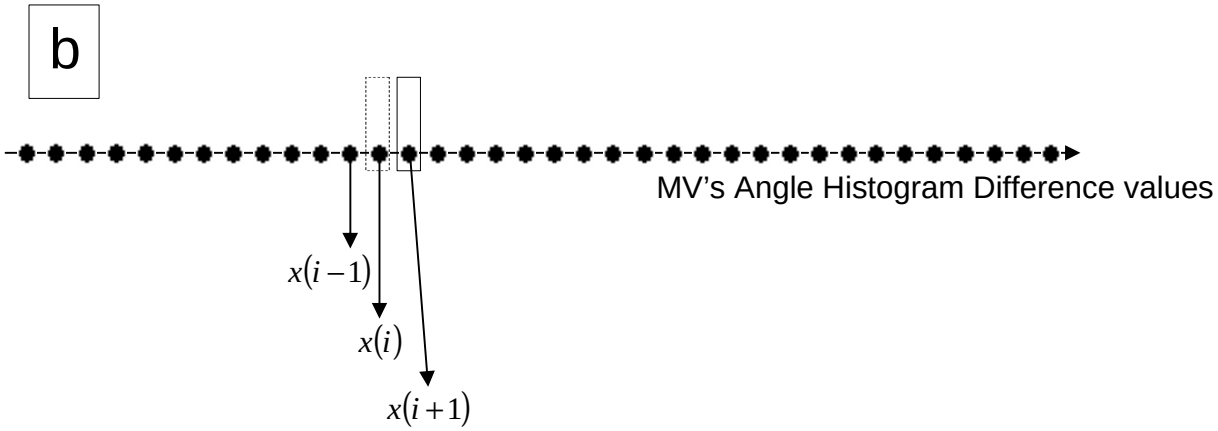
$$ {l_A}^{MV}(i) = {l_A}^{MV}(i-1) - c\left({l_A}^{MV}(i-1) - x(i)^2\right) \Leftrightarrow {l_A}^{MV}(i) = (1-c){l_A}^{MV}(i-1) + cx(i)^2 $$

(Equation 5.30)

The initial values are ${m_A}^{MV}(0) = x(0)$ and ${l_A}^{MV}(0) = x(0)^2$. Default value for $c = 0.05$. The threshold ${T_A}^{MV}(i)$ is calculated by the formula ${T_A}^{MV}(i) = {m_A}^{MV}(i) + {a_A}^{MV}{s_A}^{MV}(i)$, where ${a_A}^{MV}$ is a constant variable for the entire video. The value of variable ${a_A}^{MV}$ is changing according to video type. The next step is to compare the threshold ${T_A}^{MV}(i)$ with the next histogram difference value. In other words, the threshold ${T_A}^{MV}(i)$ is compared to $x(i+1)$. If ${T_A}^{MV}(i) > x(i+1)$, then we continue with the calculation of ${T_A}^{MV}(i+1)$. If ${T_A}^{MV}(i) \le x(i+1)$, then ${T_A}^{MV}(i+1) = {T_A}^{MV}(i)$ until the

first condition, $T_A^{MV}(i) > x(i+1)$, is satisfied. The value $x(i+1)$ is declared as Motion Change.





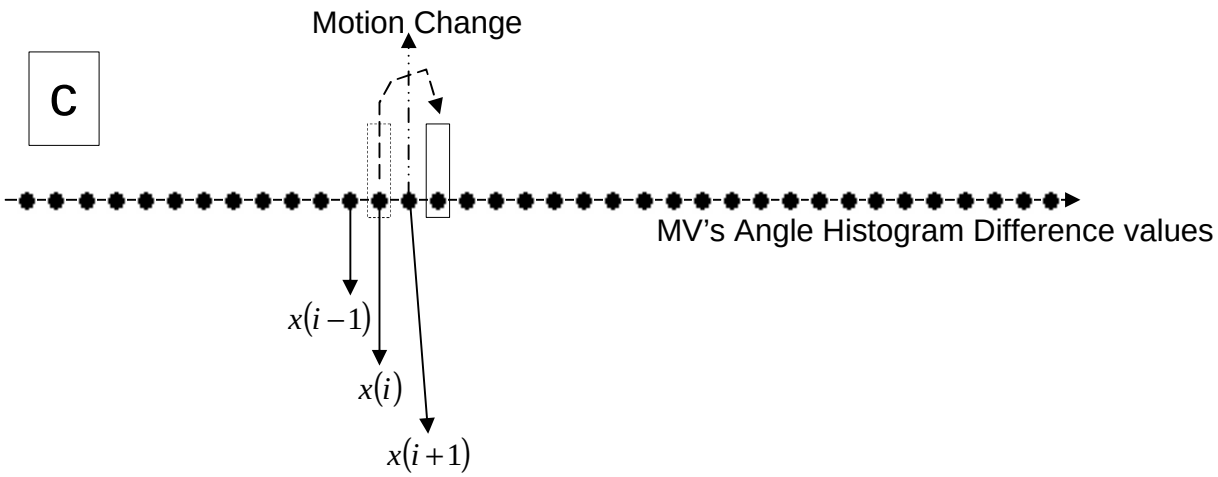
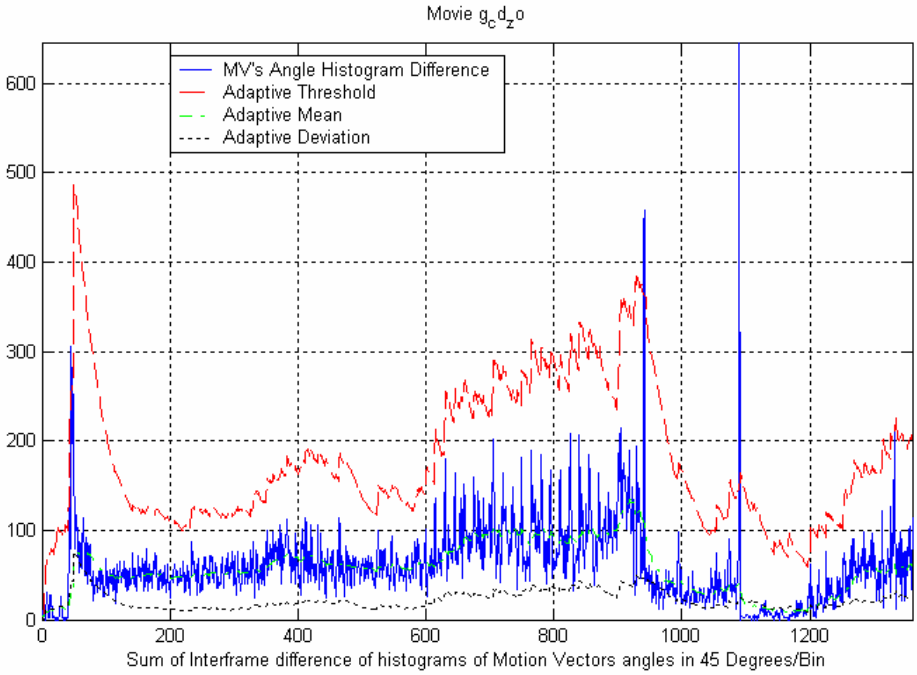
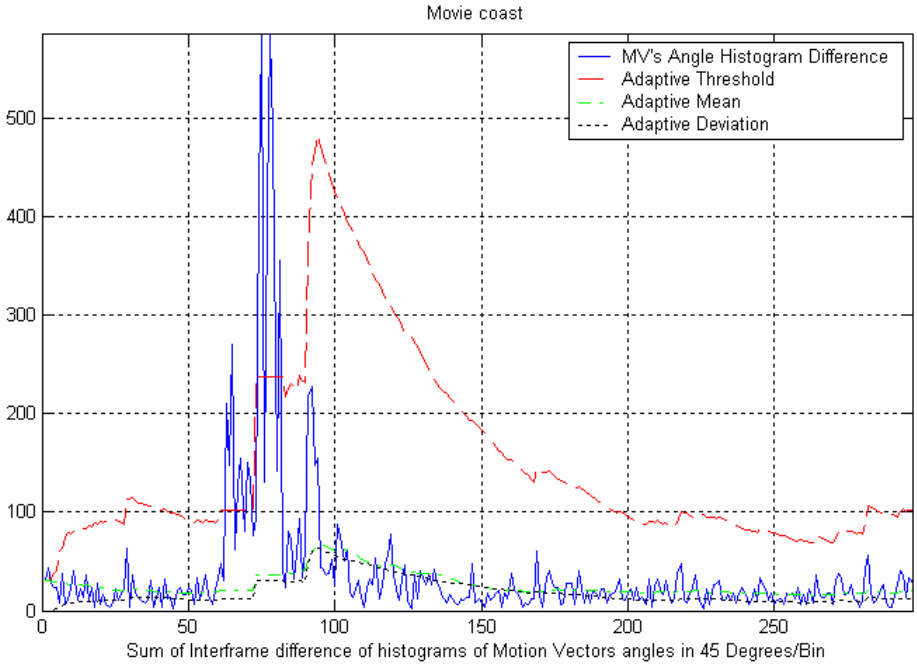


Figure 5.23: (**a**) Illustration of adaptive method in a MV's angle histogram difference graph. (**b**) Illustration of adaptive method, if condition $T_A^{MV}(i) > x(i+1)$ is satisfied. (**c**) Illustration of adaptive method, if condition $T_A^{MV}(i) \le x(i+1)$ is satisfied for one frame. In this case we have Motion Change.

Selection of Thresholds

As mentioned before, the threshold T_A^{MV} is calculated by the formula $T_A^{MV} = m_A^{MV} + a_A^{MV} s_A^{MV}$. The reason of selection this value for threshold is explained in TC approach. One threshold is used in Adaptive approach, as in SW approach, because this threshold is adapted to the local variations of histogram difference. In other words, the graph of the threshold follows the variations of the histogram difference graph. An example of m_A^{MV}, s_A^{MV} and T_A^{MV} graphs with MV's angle histogram difference graph can be seen in figure 5.24.


Movie g_c d_z o
MV's Angle Histogram Difference
Adaptive Threshold
Adaptive Mean
Adaptive Deviation
0
100
200
300
400
500
600
0
200
400
600
800
1000
1200
Sum of Interframe difference of histograms of Motion Vectors angles in 45 Degrees/Bin


Movie coast
MV's Angle Histogram Difference
Adaptive Threshold
Adaptive Mean
Adaptive Deviation
0
100
200
300
400
500
0
50
100
150
200
250
Sum of Interframe difference of histograms of Motion Vectors angles in 45 Degrees/Bin

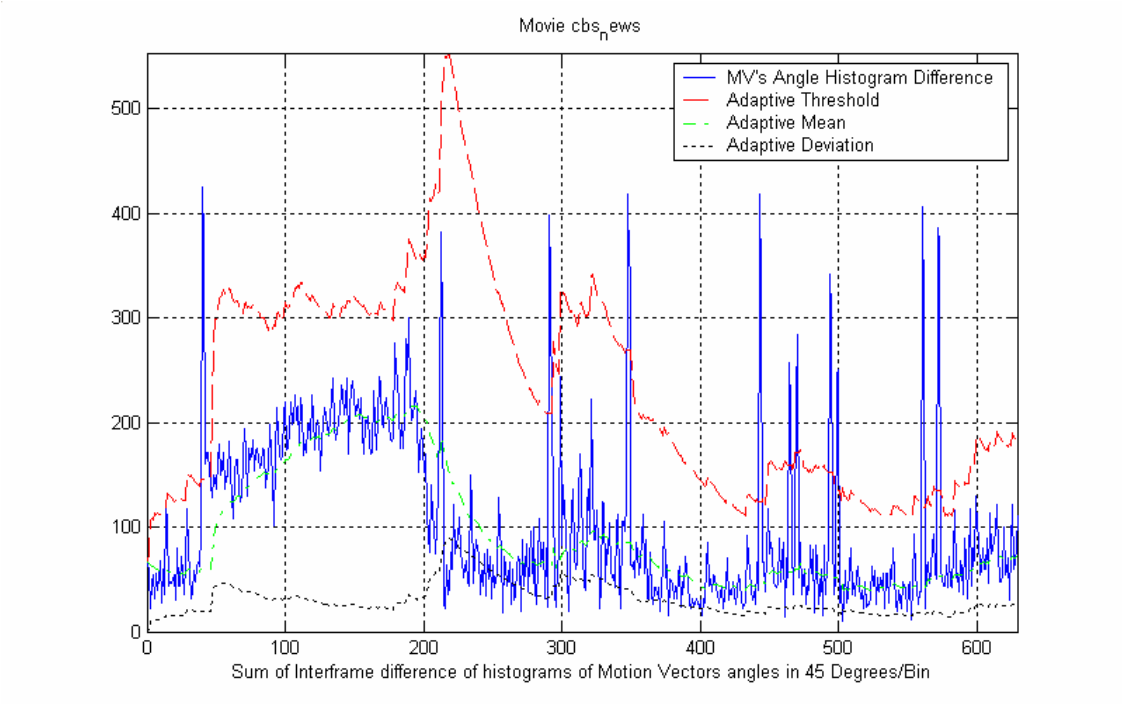


Figure 5.24: Three graphs, each one have the MV's angle histogram difference, the Adaptive Threshold $T_A^{MV}(i)$ (Red line), the Adaptive Mean $m_A^{MV}(i)$ (Yellow line) and Adaptive Deviation $s_A^{MV}(i)$ (Black line).

A different value for c variable

We can put the value of c according to equation 5.6.

Chapter 6

Experimental Results

The algorithms described before, the three algorithms for intensity and the two for motion, have been tested in a number of video sequences which categorized in *news*, *animation*, *sports* and *documentary*. In order to evaluate their performance we use ROC (Receiver Operating Characteristics) curves.

ROC Curves

The starting point for signal detection theory is that nearly all decision making takes place in the presence of some uncertainty. Signal detection theory provides a precise language and graphic notation for analyzing decision making in the presence of uncertainty. Next follows an example which will help understand ROC curves.

We want to detect brief, dim light in a dark room. Imagine we use a simple-force choice method in which the light is flashed on half of the trials, randomly interleaved. On each trial, the subject must respond "yes" or "no" to indicate whether or not they think the light was flashed. We assume that the subjects' performance is determined by the number of photon absorptions/photopigment summarizations on each trial. There are two kinds of noise factors that limit the subject's performance: *internal noise* and *external noise*.

External noise: There are many possible sources of external noise but the main source of external noise is the quantal nature of light. On average, the light source is set up to deliver certain stimulus intensity, say 100 photons. A given trial, however, there will rarely be exactly 100 photons emitted. Instead the photon count will vary from trial to trial following a Poisson distribution.

Internal noise: Internal noise refers to the fact that neural responses would be noisy, even if the stimulus was exactly the same on each trial. Some of the emitted photons will be scattered by the cornea, the lens and the other goopy stuff in the eye. The number of scattered photons will vary randomly from trial to trial. Of the photons that reach the photoreceptors, not all of them will be absorbed by the photopigments.

In practice, it would be impossible to measure the number of photons absorbed on any given trial because we would have to record simultaneously from all the rods in the retina. It is possible to characterize the probability that a certain number of photons will be absorbed. Each

of the relevant factors (number of photons emitted, number of photons scattered, and number of photons absorbed) can be modeled as a Poisson process. A sequence of Poisson processes behaves altogether like a single Poisson process with an overall rate constant equal to the products of all of individual constants. For example, assume that 100 photons are emitted on average that 10% of those photons pass through the eyes' optics on average, and that 10% of those are absorbed by photoreceptors on average. Then there will be 1 photon absorbed on average for each trial on which the light is flashed, and this number will vary from trial to trial following a Poisson distribution.

On trials for which no light is flashed, there will typically be some non-zero level of response, due to thermal isomerizations of photopigment molecules. Barlow called this the "dark light" because a spontaneous isomerization will lead to the same neural signal as if a proton was actually absorbed. The subject will not be able to tell the difference between real light and dark light.

Internal Response Probability Density Functions

Because the task is so hard, there is always some uncertainty as to what was there or not. Either there was a flash (signal plus noise) or there was no flash (noise alone). Either the subject saw the flash (respond "yes") or did not (respond "no"). So, there are four possible outcomes:

1. hit (signal present and subject says "yes")
2. miss (signal present and subject says "no")
3. false alarm (signal absent and subject says "yes")
4. correct rejection (signal absent and subject says "no")

Hits and correct detections are good, contrary to false alarms and misses. Figure 6.1 shows a graph of two hypothetical internal response curves. The curve on the left is for the noise-alone trials, and the curve on the right is for the signal-plus-noise trials. The height of each curve represents how often that level of internal response will occur. Notice that we never lose the noise. The internal response for the signal-plus-noise case is generally greater but there is still a distribution (a spread) of possible responses. Notice also that the curves overlap, that is the internal response for a noise-alone trial may exceed the internal response for a signal-plus-noise trial.

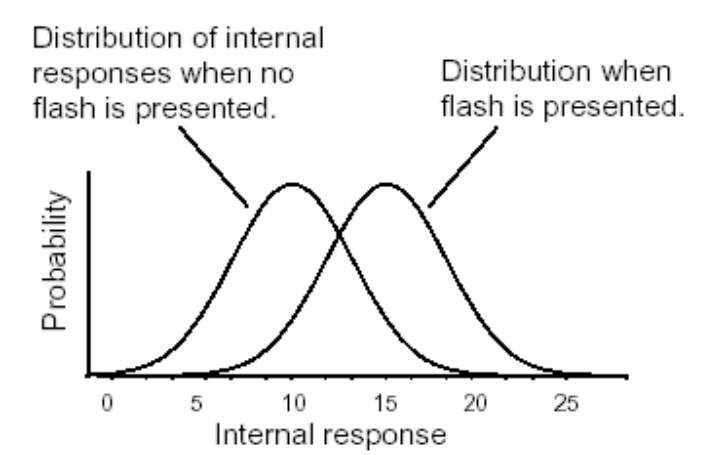


Figure 6.1: Internal response probability density functions for noise-alone and for signal-plus-noise trials. On noise-alone trials, in this example, there will generally be about 10 units of internal response (i.e. 10 photopigment isomerizations). However, there will be some trials with more (or less) internal response because of the internal and external noise.

The Role of Criterion

There are two main components to the decision making process: *stimulus strength* and *criterion*. The stimulus strength affects the probability density functions in the obvious way: a stronger signal (brighter flash) will shift the signal-plus-noise curve to the right.

The second component of the decision process is quite different. The subject is being asked to use their own judgment in making a decision. Different subjects may feel that the different types of errors are not equal. Perhaps the simplest strategy that the subject can adopt is to pick a *criterion* location along the internal response axis. Whenever the internal response is greater than this criterion they respond "yes". Whenever the internal response is less than this criterion they respond "no".

An example criterion is indicated by the vertical lines in figure 6.2. The criterion line divides the graph into four sections that correspond to: hits, misses, false alarms and correct rejections. On both hits and false

alarms, the internal response is greater than the criterion, because the subject is responding "yes". Hits correspond to signal-plus-noise trials when the internal response is greater than criterion, as indicated in the figure. False alarms correspond to noise-alone trials when the interval response is greater than criterion, as indicated in the figure.

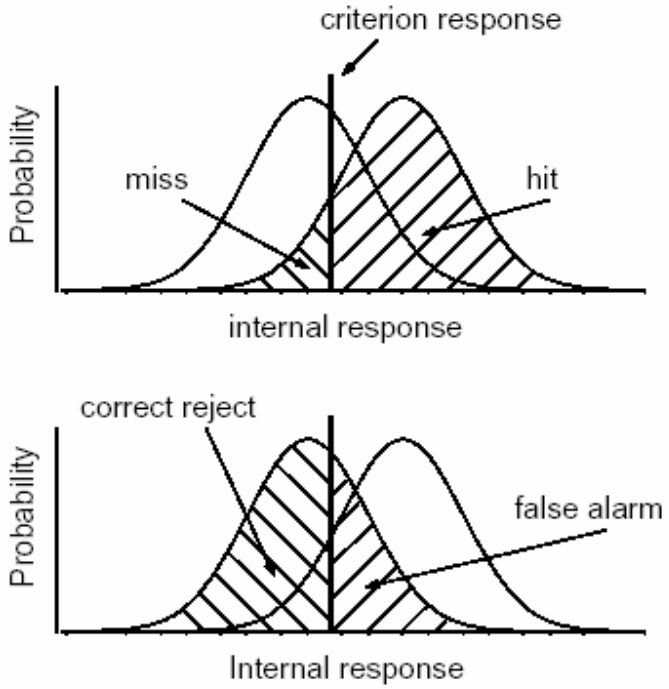


Figure 6.2: Internal response probability density functions for noise-alone and for signal-plus-noise trials. Since the curves overlap, the internal response for a noise-alone trial may exceed the internal response for a signal-plus-noise trial. Vertical lines correspond to the criterion response.

Suppose that the subject chooses a low criterion, as shown in figure 6.3 at the top, so that they respond "yes" to almost everything. Then they will never miss a flash when it is present and they will therefore have a very high rate. On the other hand, saying "yes" to almost everything will greatly increase the number of false alarms. Thus, there is a clear cost to increasing the number of hits, and that cost is paid in terms of false alarms. If the subject chooses a high criterion, as shown in figure 6.3 at the bottom, then they respond "no" to almost everything. They will rarely make a false alarm, but they will also miss many real flashes.

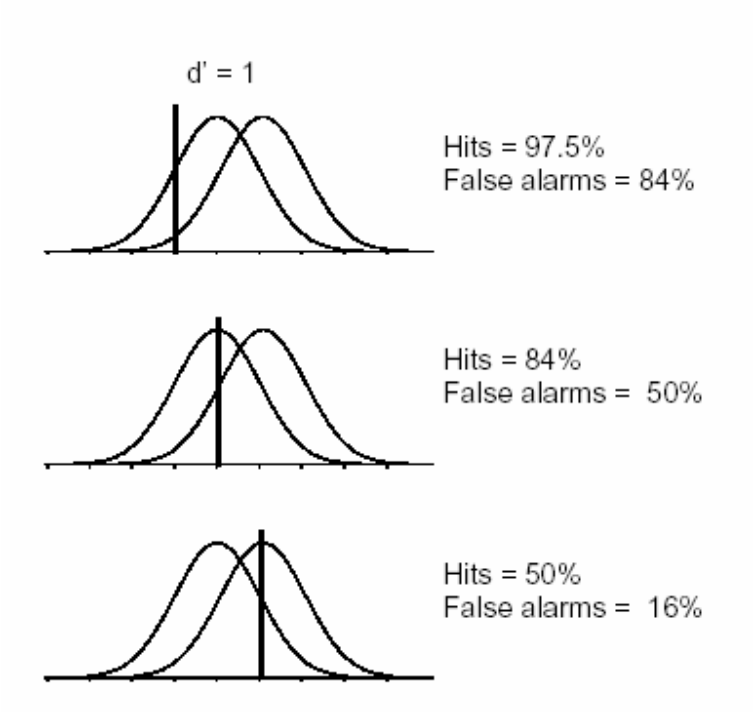


Figure 6.3: Effect of shifting the criterion

There is no way that the subject can set their criterion to achieve only hits and no false alarms. It is inevitable that some mistakes will be made. Because of the noise it is simply a true, undeniable, fact that the internal responses on noise-alone trials may exceed the internal responses of signal-plus-noise trials, in some instances. Thus the subject cannot always be right. They can adjust the kind of errors that they make by manipulating their criterion, the one part of this diagram that is under their control.

The Receiver Operating Characteristic

We can describe the full range of the subject's options in a single curve, called an *ROC curve*. The ROC captures, in a single graph, the various alternatives that are available to the subject as they move their criterion to higher and lower levels.

ROC curves, which shown in figure 6.4, are plotted with the false alarm rate on the horizontal axis and the hit rate on the vertical axis. We already know that if the criterion is high, then both the false alarm rate and the hit rate will be very low. If we move the criterion lower, then the hit rate and the false alarm rate both increase. So the full ROC curve has an upward sloping shape. Notice also that for any reasonable choice of criterion, the hit rate is always larger than the false alarm rate, so the ROC

curve is bowed upward. The subject may set the criterion anywhere, but any choice that they make will land them with a hit and false alarm rate somewhere on the ROC curve.

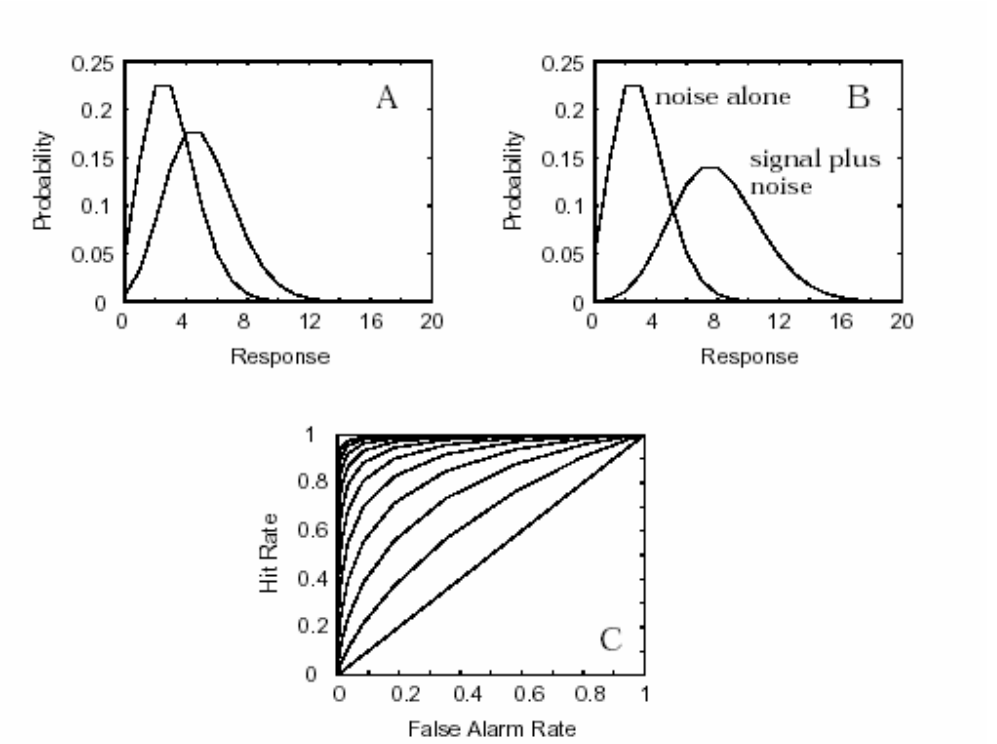


Figure 6.4: Internal response probability density functions and ROC curves for different signal strengths. When the signal is stronger there is less overlap in the probability of occurrence curves, and the ROC curve becomes more bowed. **A**: Probability density functions when the signal evokes an average of 2 photon absorptions per trial. **B**: Probability density functions when the signal evokes an average of 5 photon absorptions per trial. **C**: ROC curves for a series of signal strengths that evoke an average of n=0, 1, 2, 3, …, 10 photon absorptions per trial. In all cases the dark noise (average number of spontaneous isomerizations per trial) was 3.

The Role of Signal Strength

If we present a brighter flash (e.g. with 200 photons emitted per flash on average rather than 100), then the subject's internal response strength will, on average, be stronger. Pictorially, this will have the effect of shifting the probability density function for signal-plus-noise trials to the right, a bit further away from the noise-alone probability density.

Figure 6.4 shows two sets of probability densities and two ROC curves. When the signal is stronger there is less overlap between the two probability density curves. When this happens the subject's choices are not so difficult as before. They can pick a criterion to get nearly a perfect hit, with almost no false alarms. ROC curves for stronger signals bow out further than ROC curves for weaker signals.

Varying the Noise

There is another aspect of the probability densities that also determines detectability: the *spread* of the curves. For example, consider the two sets of probability densities in figure 6.5. The separation between the peaks is the same but the second set of curves is much skinnier. Clearly, the signal is much more discriminable when there is less spread (less noise) in the probability densities. So the subject would have an easier time setting their criterion in order to be right nearly all the time.

In our example, we have assumed Poisson noise so the absorption count variance is proportional to the mean absorption count. However, one can easily imagine situations in which the response variance depends on factors that are independent of the mean response.

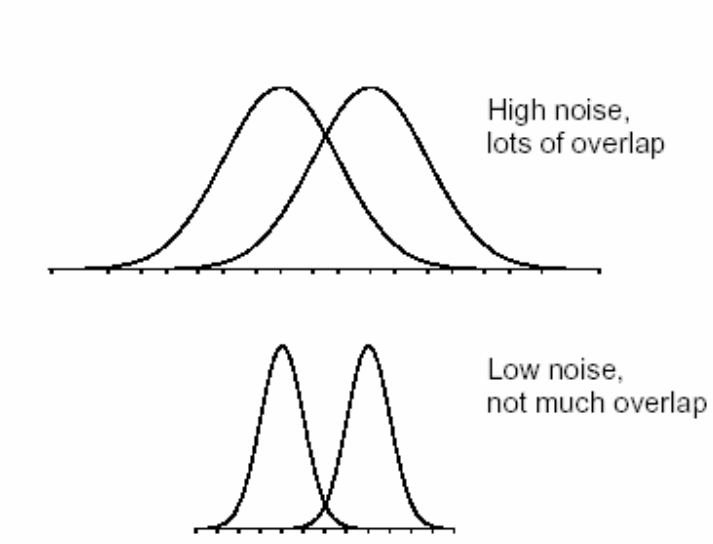


Figure 6.5: Internal response probability density functions for two different noise levels. When the noise is greater, the curves are wider (more spread) and there is more overlap.

ROC Curves of Algorithms

Next follows ROC curves of each algorithm for the four categories of videos. At the end we have the overall results for all videos.

Category documentary

In the next figure, figure 6.6, we can see the ROC curves of Camera Break detection for category documentary.

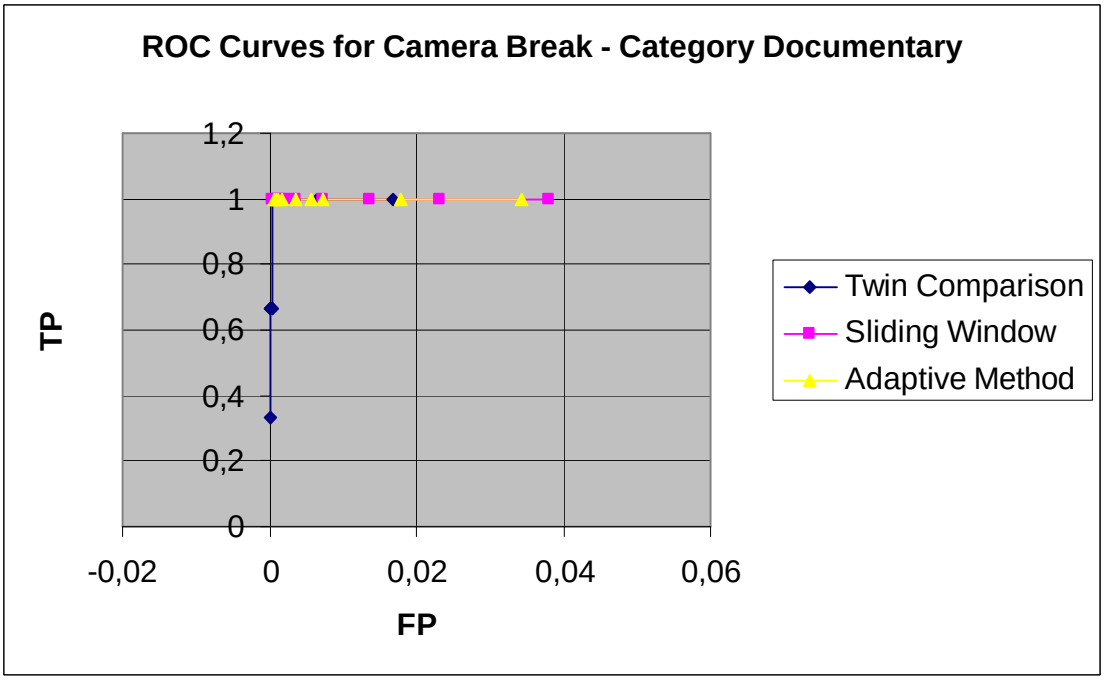


Figure 6.6: ROC curves for algorithms of Twin Comparison, Sliding Window and Adaptive Method for Camera Break detection in videos of category documentary. We used variable a of threshold as a criterion for these curves. The values of a which were used are 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5 and 6.

Observing the above figure we can notice that the curves of Sliding Window and Adaptive Method are parallel to x-axis. X-axis is FP (False Positives) and y-axis is TP (True Positives). Next follows the mathematic formula of FP and TP for Camera Break detection.

$$FP = \frac{\text{Number of frames which are False Detected as Camera Breaks}}{\text{Number of frames which are not Camera Breaks}} \quad \text{(Equation 6.1)}$$

$$TP = \frac{\text{Number of frames which are Correct Detected as Camera Breaks}}{\text{Number of frames which are Camera Breaks}} \quad \text{(Equation 6.2)}$$

The curves which are parallel to x-axis denote that for all values of variable a the algorithms can detect a standard number of frames which are Camera Breaks. In our case the value of TP is 1, so all the frames which are Camera Breaks are detected. We also observe that while the value of variable a increasing, the curves, including the curve of Twin Comparison, are directed to a zero value of FP. This means that while the threshold increasing, the number of frames which are False Detected as Camera Breaks decreasing. For curve of Twin Comparison we observe that for big values of a not only we don't have False Detected frames but also the number of Correct Detected frames is decreasing. Apparently this happens to the other two algorithms, but for bigger values of a which are not shown in this graph. In other words, the algorithm of Twin Comparison reach the zero value of this graph (FP=0, TP=0) much faster than the other two algorithms. So, the derivative of curve of Twin Comparison is smaller than the derivative of the other two algorithms. The area below the graph is smaller than the algorithms of Sliding Window and Adaptive Method meaning that the other two algorithms are better for this category.

The duration of Camera Break is varying from algorithm to algorithm. So, in Twin Comparison the duration is one frame and in the other two algorithms the duration is less than or equal three frames. We chose the different duration in Sliding Window and Adaptive Method because the threshold has not a stable value as in Twin Comparison and the values of threshold are sensitive to histogram difference values.

In the next figure, figure 6.7, we can see the ROC curves of Gradual Transition detection for category documentary.

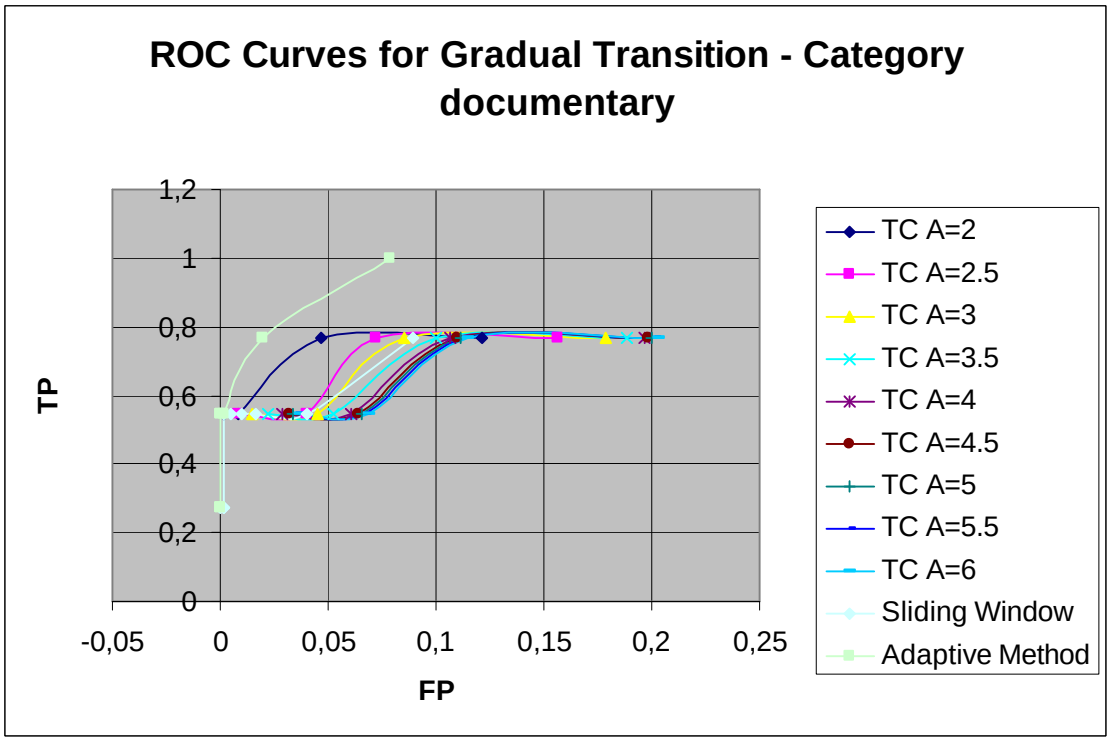


Figure 6.7: ROC curves for algorithms of Twin Comparison, Sliding Window and Adaptive Method for Gradual Transition detection in videos of category documentary. We used variable a of threshold as a criterion for algorithms Sliding Window and Adaptive Method. For the algorithm of Twin Comparison we used variable b as a criterion for the curves. The values of b which were used are 1.5, 1.7, 1.9 and 2.1. For Twin Comparison we kept the variable a stable (which is used for Camera Break detection) and for several values of a we increase b .

Observing the above figure we can notice that the curves of Twin Comparison for several values of a have the same form and while the a increasing, the curves are directed to a standard value which will be the upper limit. Above that value, there will be no curve. This is logical for Twin Comparison because while the a value increasing, the first threshold increasing (the threshold for Camera Break detection) to a value that will not detect any Camera Break (the value of the threshold will be bigger that of the histogram difference). This is the upper limit and above that value there will be no change in the results for any value of b for Gradual Transition detection. X-axis is FP (False Positives) and y-axis is TP (True Positives). Next follows the mathematic formula of FP and TP for Gradual Transition detection.

$$FP = \frac{\text{Number of frames which are False Detected as Gradual Transition}}{\text{Number of frames which are not Gradual Transition}}$$ (Equation 6.3)

$$TP = \frac{\text{Number of frames which are Correct Detected as Gradual Transition}}{\text{Number of frames which are Gradual Transition}}$$ (Equation 6.4)

It is worth mention that for algorithms of Sliding Window and Adaptive Method the duration of a Gradual Transition is larger or equal three frames. In Twin Comparison the duration is larger or equal one frame. We chose the different duration because the threshold of Sliding Window and Adaptive Method has not a stable value and the values are sensitive to histogram difference. Observing the diagram above, we can notice that the curve of Adaptive Method is above all the others, so this method has better results in this category. The curve of Sliding Window is between the Twin Comparison curve of value $a = 3$ and $a = 3.5$. We observe that the curve of Sliding Window is similar to the curves of Twin Comparison with the difference that for big values of b the curve is directed to zero value (FP=0, TP=0). The Twin Comparison curves for big values of b are parallel to x-axis, in other words the FP are decreasing and the FP have a stable value.

In the next figure, figure 6.8, we can see the ROC curves of Camera Motion Change detection for category documentary.

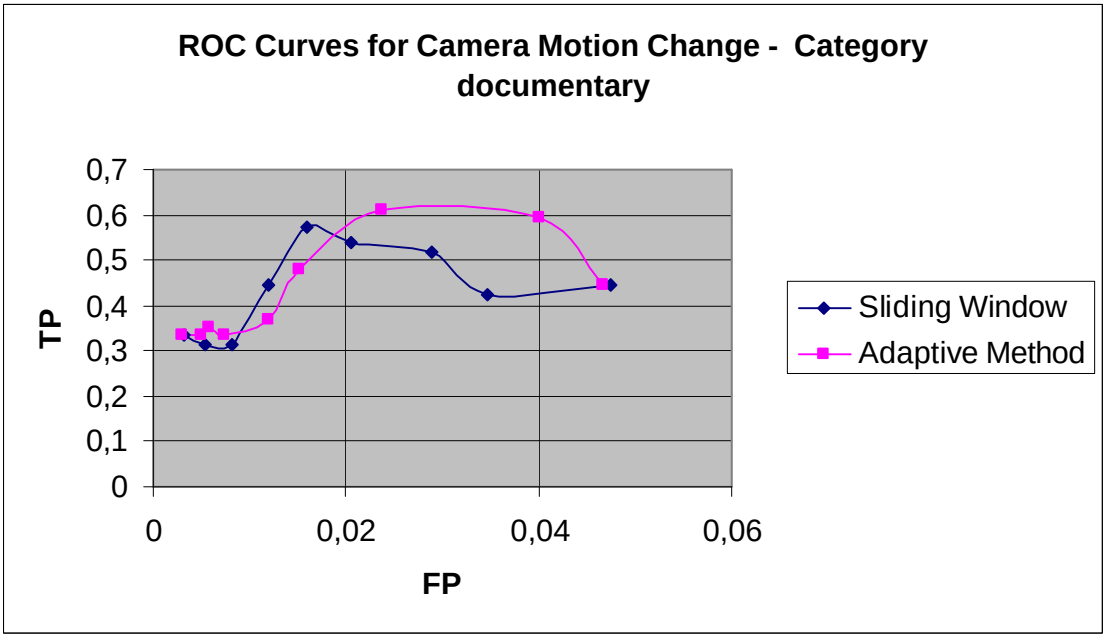


Figure 6.8: ROC curves for algorithms of Sliding Window and Adaptive Method for Camera Motion change detection in videos of category documentary. We used variable a of threshold as a criterion for algorithms Sliding Window and Adaptive Method.

X-axis is FP (False Positives) and y-axis is TP (True Positives). Next follows the mathematic formula of FP and TP for Camera Motion Change detection.

$$FP = \frac{\text{Number of frames which are False Detected as Camera Motion Change}}{\text{Number of frames which are not Camera Motion Change}}$$

(Equation 6.5)

$$TP = \frac{\text{Number of frames which are Correct Detected as Camera Motion Change}}{\text{Number of frames which are Camera Motion Change}}$$

(Equation 6.6)

Observing the above two curves we can notice that for small values of a (the right most values) the value of ROC curve is small and while the a is increasing the value of ROC curve increasing too. After a value of a and while a is still increasing the value of ROC curve is decreasing directed to zero value. Before we give an explanation of these curves it is worth mention that the duration of a Camera Motion Change is less or equal three frames, just as Camera Break. We chose this duration because threshold is sensitive to histogram difference values. If

we had chosen the duration as one frame, we would have a lot of false and missed detections.

While the a value is small, the threshold is also small and a lot of histogram difference values are bigger than threshold. In other words the duration is bigger than three frames and so these frames are false detected. While the value of a increasing, the threshold increasing too and we have less false detected frames and much more correct detected. For big values of a , the threshold is big enough than most of the histogram difference values, so we have not only less false detections but also less correct detection. For that reason the two curves are directed to zero value (FP=0, TP=0).

Category news

In the next figure, figure 6.9, we can see the ROC curves of Camera Break detection for category news.

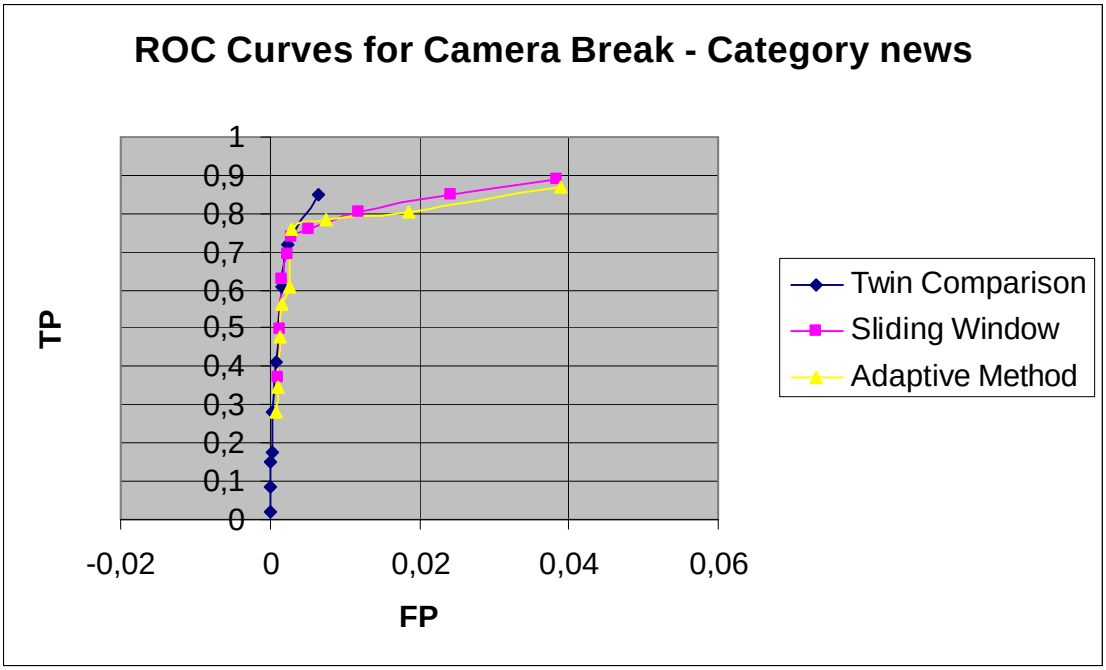


Figure 6.9: ROC curves for algorithms of Twin Comparison, Sliding Window and Adaptive Method for Camera Break detection in videos of category news. We used variable a of threshold as a criterion for these curves. The values of a which were used are 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5 and 6.

X-axis is FP (False Positives) and y-axis is TP (True Positives). The mathematic formula is given by equations 6.1 and 6.2. We can observe that while the value of a increasing the curves are directed to zero value (FP=0, TP=0). The curves of Sliding Window and Adaptive Method are similar. The curve of Twin Comparison is above the other two curves so it has better results in this category. As we mentioned in the explanation of figure 6 the duration of Camera Break for Sliding Window and Adaptive Method is less or equal three frames. So, while the value of a is decreasing, the threshold decreasing and a lot of histogram difference values are bigger than threshold, in other words we have a lot of false detections. While the a increasing the threshold increasing and so we have less false detected frames and much more correct detected. For big values of a the value of threshold is much bigger than of histogram difference values, so we have not only less false detections but also less correct detection.

In the next figure, figure 6.10, we can see the ROC curves of Gradual Transition detection for category news.

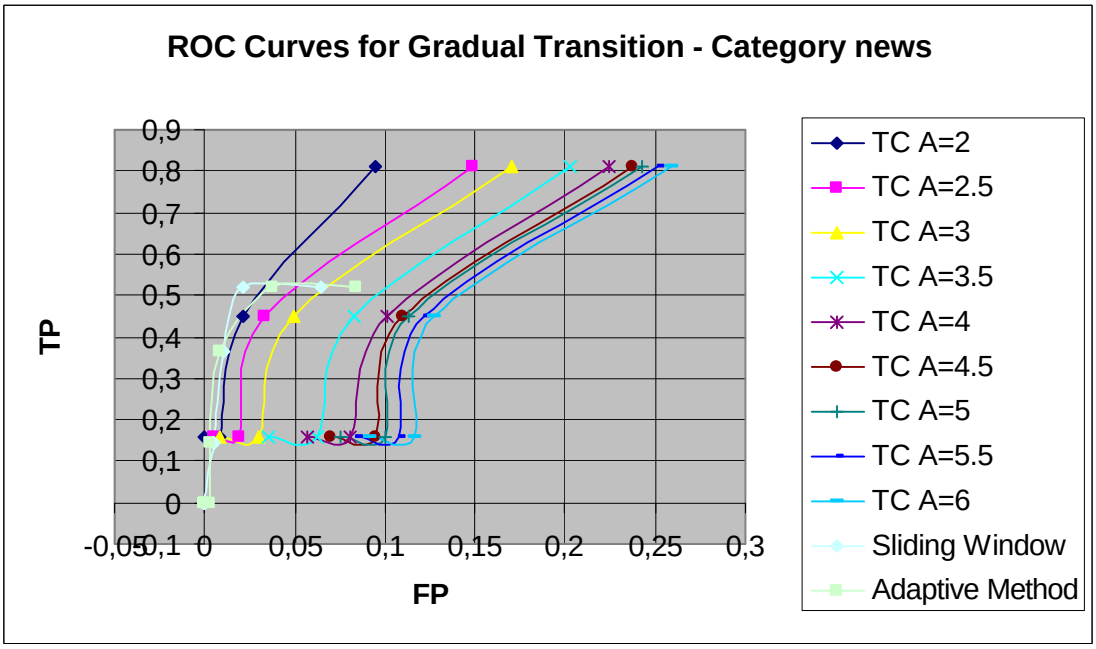


Figure 6.10: ROC curves for algorithms of Twin Comparison, Sliding Window and Adaptive Method for Gradual Transition detection in videos of category news. We used variable a of threshold as a criterion for algorithms Sliding Window and Adaptive Method. For the algorithm of Twin Comparison we used variable b as a criterion for the curves. The values of b which were used are 1.5, 1.7, 1.9 and 2.1. For Twin Comparison we kept the variable a stable (which is used for Camera Break detection) and for several values of a we increase b .

X-axis is FP (False Positives) and y-axis is TP (True Positives). The mathematic formula is given by equations 6.3 and 6.4. We observe that for small values of a the curves of Sliding Window and Adaptive Method are parallel to x-axis for value of TP=0.52. Observing the two curves, the curve of Sliding Window is above the other so has better results from the Adaptive Method for this category. Twin Comparison has better results from Sliding Window as it can be seen from the ROC curves. In this category and for Twin Comparison, for small values of b we have big values for ROC curves. For big values of b the values of the curves are parallel to x-axis, in other words the FP are decreasing and the FP have a stable value.

In the next figure, figure 6.11, we can see the ROC curves of Camera Motion Change detection for category news.

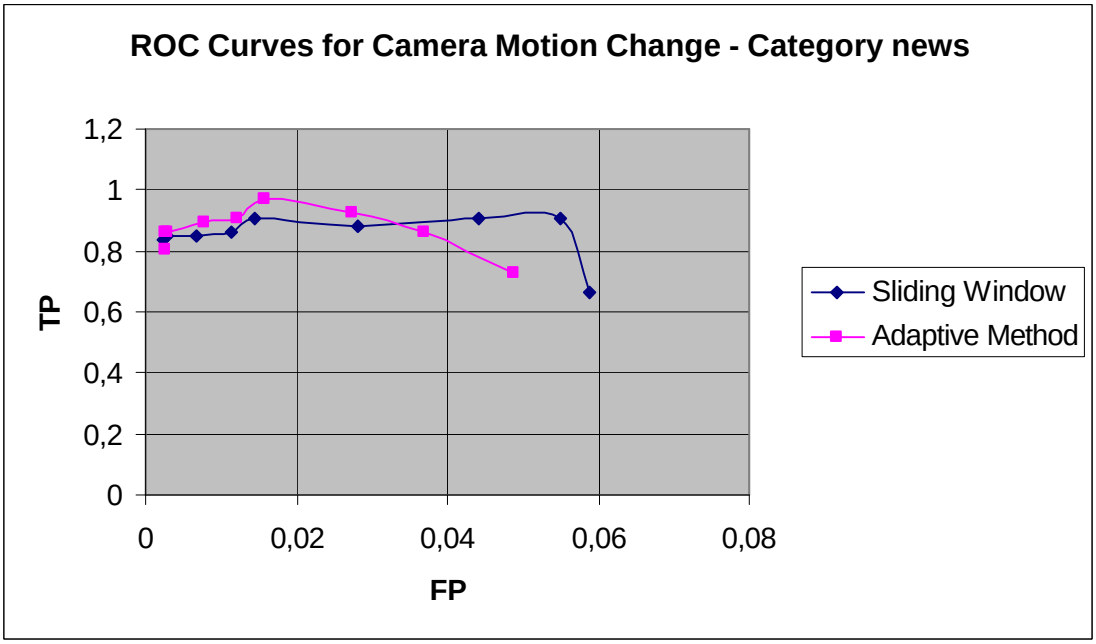


Figure 6.11: ROC curves for algorithms of Sliding Window and Adaptive Method for Camera Motion change detection in videos of category news. We used variable a of threshold as a criterion for algorithms Sliding Window and Adaptive Method.

X-axis is FP (False Positives) and y-axis is TP (True Positives). The mathematic formula is given by equations 6.5 and 6.6. Observing the above two curves we can notice that for small values of a (the right most values) the value of ROC curve is small and while the a is increasing the value of ROC curve increasing too. After a value of a and while a is still

increasing the value of ROC curve is decreasing directed to zero value. While the a value is small, the threshold is also small and a lot of histogram difference values are bigger than threshold. In other words the duration is bigger than three frames and so these frames are false detected. While the value of a increasing, the threshold increasing too and we have less false detected frames and much more correct detected. For big values of a, the threshold is big enough than most of the histogram difference values, so we have not only less false detections but also less correct detection. For that reason the two curves are directed to zero value (FP=0, TP=0).

Category animation

In the next figure, figure 6.12, we can see the ROC curves of Camera Break detection for category animation.

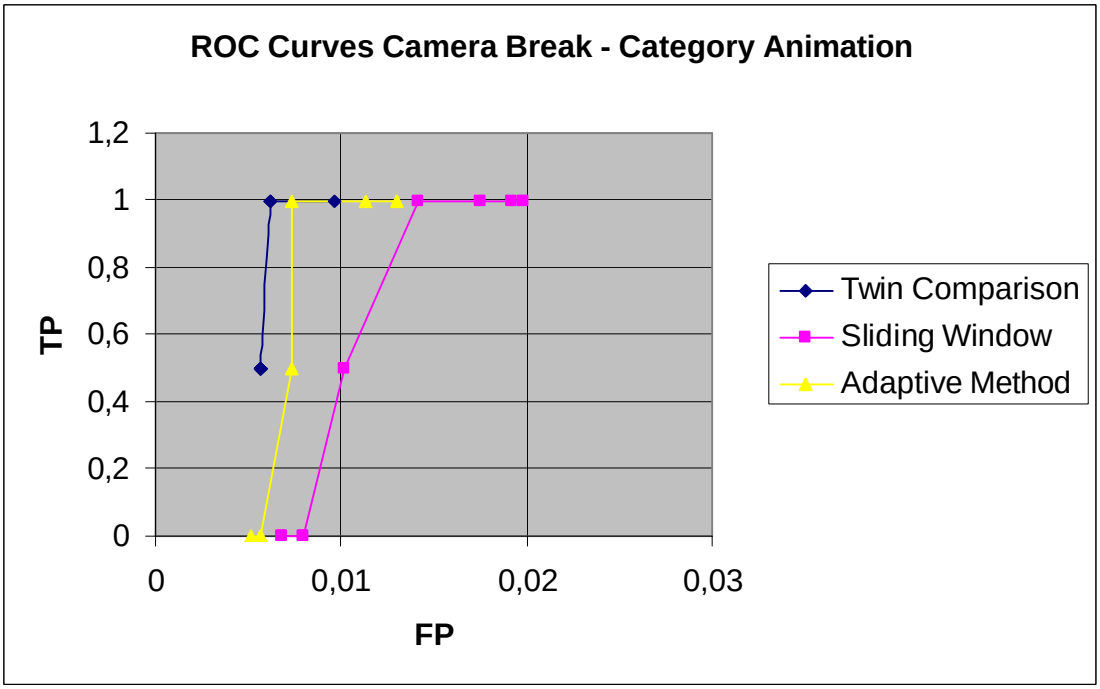


Figure 6.12: ROC curves for algorithms of Twin Comparison, Sliding Window and Adaptive Method for Camera Break detection in videos of category animation. We used variable a of threshold as a criterion for these curves. The values of a which were used are 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5 and 6.

X-axis is FP (False Positives) and y-axis is TP (True Positives). The mathematic formula is given by equations 6.1 and 6.2. Observing the above graph we can notice that the curve of Twin Comparison is above

the other two curves, which means that has better results from the other two algorithms. Next follows the Adaptive Method and then Sliding Window. It is worth mention that for three values of a ($a = 2,2.5,3$) we have TP=1. The same happens for four values of a ($a = 2,2.5,3,3.5$) for Sliding Window and Adaptive Method.

In the next figure, figure 6.13, we can see the ROC curves of Gradual Transition detection for category animation.

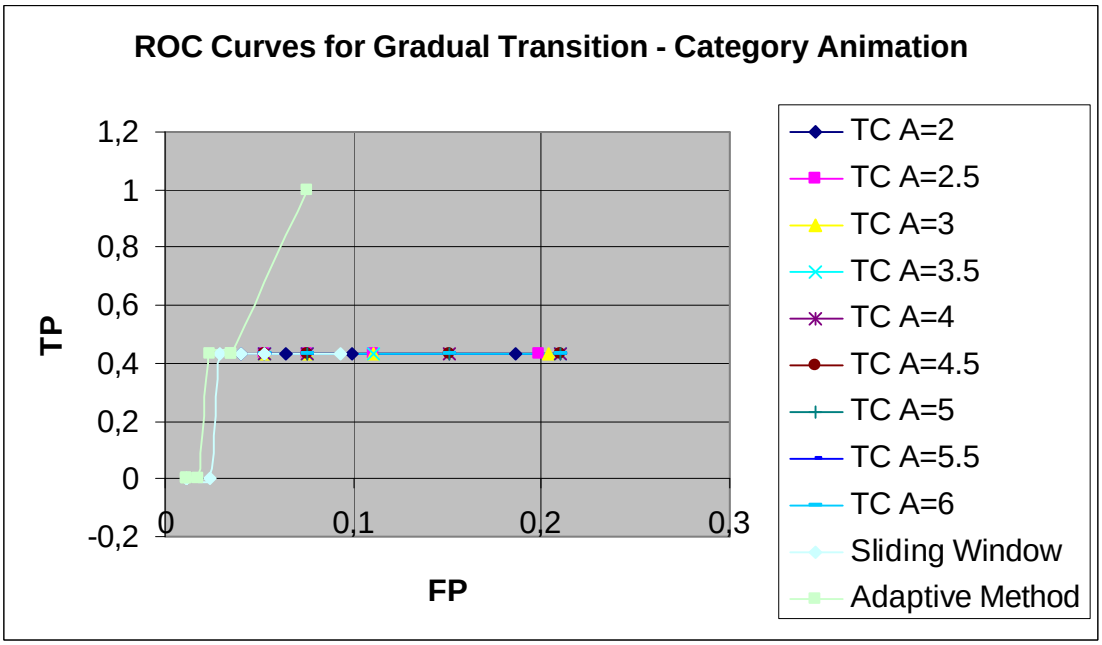


Figure 6.13: ROC curves for algorithms of Twin Comparison, Sliding Window and Adaptive Method for Gradual Transition detection in videos of category animation. We used variable a of threshold as a criterion for algorithms Sliding Window and Adaptive Method. For the algorithm of Twin Comparison we used variable b as a criterion for the curves. The values of b which were used are 1.5, 1.7, 1.9 and 2.1. For Twin Comparison we kept the variable a stable (which is used for Camera Break detection) and for several values of a we increase b .

X-axis is FP (False Positives) and y-axis is TP (True Positives). The mathematic formula is given by equations 6.3 and 6.4. We observe that all the curves of Twin Comparison are parallel to x-axis indicating that while b is increasing we have a stable value for TP which is TP=0.43 and a decrease of false positive. The curve of Adaptive Method is above the other curves, so it has better results in this category. The curve of Sliding Window for small values of a ($a = 2,2.5,3,3.5$) has the same value of TP as Twin Comparison.

In the next figure, figure 6.14, we can see the ROC curves of Camera Motion Change detection for category animation.

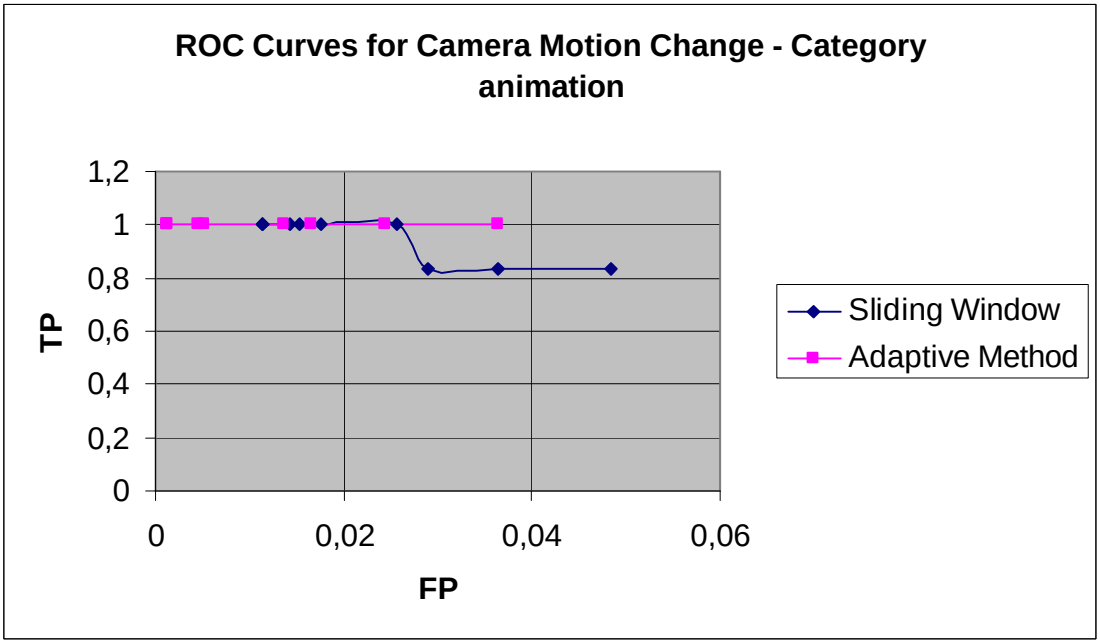


Figure 6.14: ROC curves for algorithms of Sliding Window and Adaptive Method for Camera Motion change detection in videos of category animation. We used variable a of threshold as a criterion for algorithms Sliding Window and Adaptive Method.

X-axis is FP (False Positives) and y-axis is TP (True Positives). The mathematic formula is given by equations 6.5 and 6.6. Observing the above diagram we can notice that the curve of Adaptive Method is parallel to x-axis and the value of TP is one. For small values of a ($a = 2,2.5,3$) the values of ROC for Sliding Window are the same TP=0.83. When a is increasing, TP values are increasing to value one while FP are decreasing. So the best results come from Adaptive Method.

Category sports

In the next figure, figure 6.15, we can see the ROC curves of Camera Break detection for category sports.

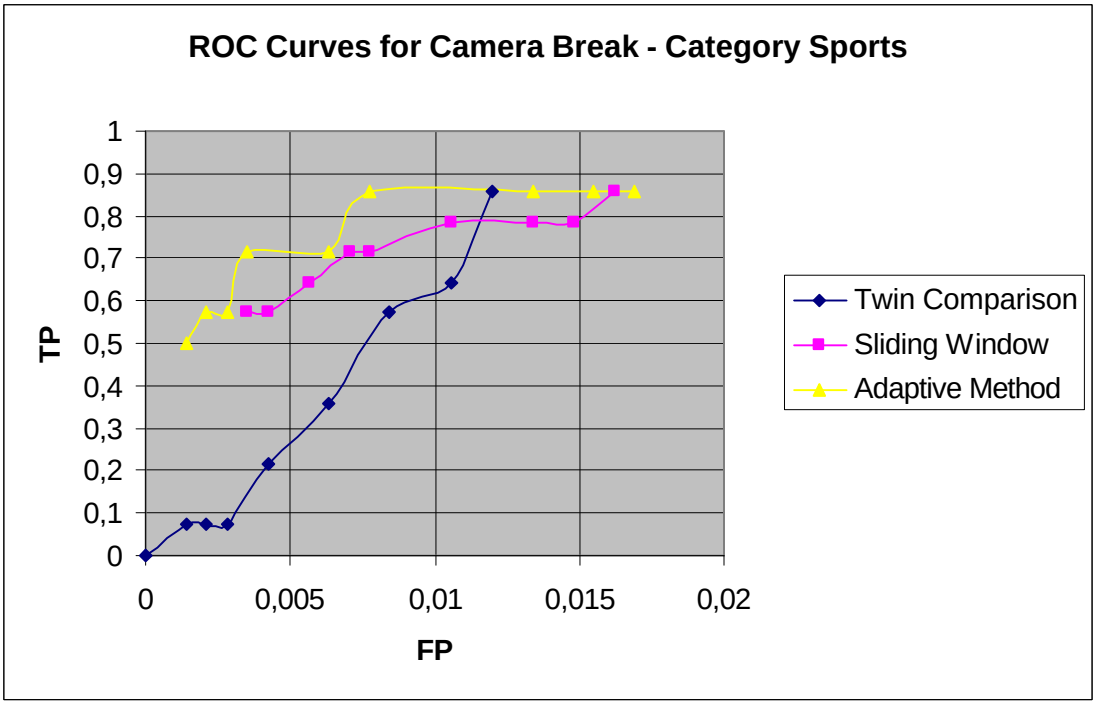


Figure 6.15: ROC curves for algorithms of Twin Comparison, Sliding Window and Adaptive Method for Camera Break detection in videos of category sports. We used variable a of threshold as a criterion for these curves. The values of a which were used are 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5 and 6.

X-axis is FP (False Positives) and y-axis is TP (True Positives). The mathematic formula is given by equations 6.1 and 6.2. We observe that the curve of Adaptive Method is above the other curves, so we have better results with this method for this category. Next follows Sliding Window and then Twin Comparison.

In the next figure, figure 6.16, we can see the ROC curves of Gradual Transition detection for category sports.

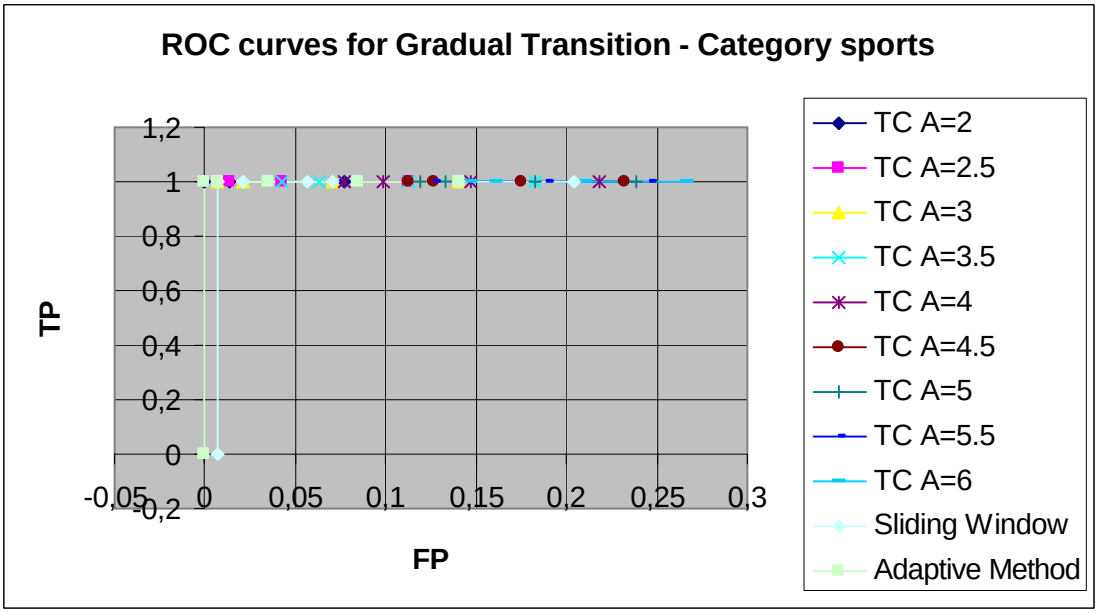


Figure 6.16: ROC curves for algorithms of Twin Comparison, Sliding Window and Adaptive Method for Gradual Transition detection in videos of category sports. We used variable a of threshold as a criterion for algorithms Sliding Window and Adaptive Method. For the algorithm of Twin Comparison we used variable b as a criterion for the curves. The values of b which were used are 1.5, 1.7, 1.9 and 2.1. For Twin Comparison we kept the variable a stable (which is used for Camera Break detection) and for several values of a we increase b .

X-axis is FP (False Positives) and y-axis is TP (True Positives). The mathematic formula is given by equations 6.3 and 6.4. We observe that the curves of Twin Comparison are parallel to x-axis with a value of TP=1. The same happens for the Sliding Window and Adaptive Method with the difference that in the last value of a the value of TP=0.

In the next figure, figure 6.17, we can see the ROC curves of Camera Motion Change detection for category sports.

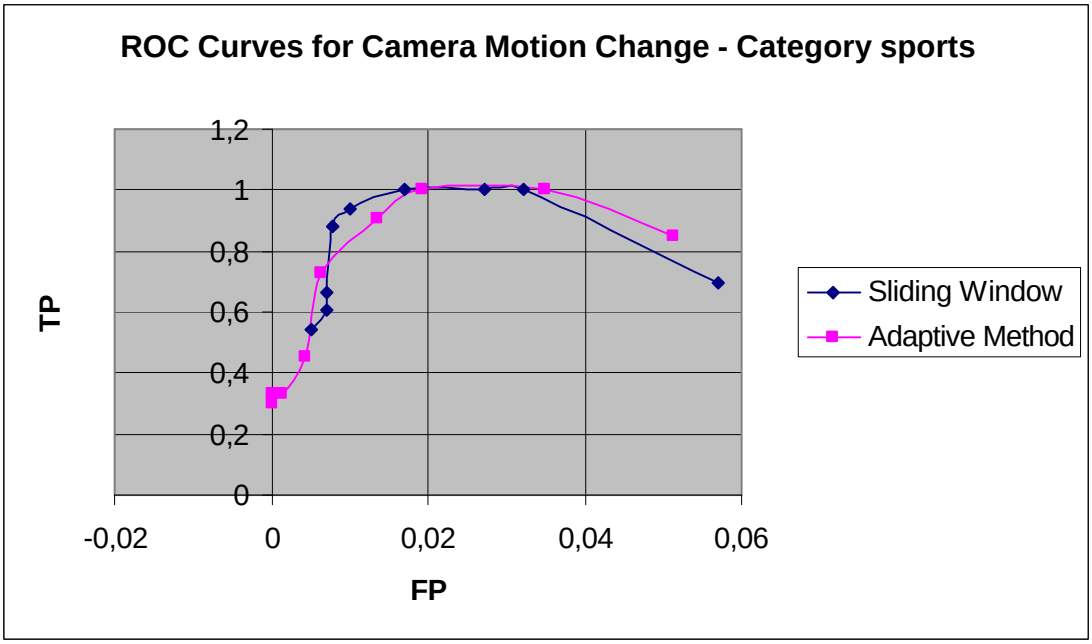


Figure 6.17: ROC curves for algorithms of Sliding Window and Adaptive Method for Camera Motion change detection in videos of category sports. We used variable a of threshold as a criterion for algorithms Sliding Window and Adaptive Method.

X-axis is FP (False Positives) and y-axis is TP (True Positives). The mathematic formula is given by equations 6.5 and 6.6. Observing the above two curves we can notice that for small values of a (the right most values) the value of ROC curve is small and while the a is increasing the value of ROC curve increasing too. After a value of a and while a is still increasing the value of ROC curve is decreasing directed to zero value. While the a value is small, the threshold is also small and a lot of histogram difference values are bigger than threshold. In other words the duration is bigger than three frames and so these frames are false detected. While the value of a increasing, the threshold increasing too and we have less false detected frames and much more correct detected. For big values of a, the threshold is big enough than most of the histogram difference values, so we have not only less false detections but also less correct detection. For that reason the two curves are directed to zero value (FP=0, TP=0).

Overall Results (All Videos)

In the next figure, figure 6.18, we can see the ROC curves of Camera Break detection for all videos.

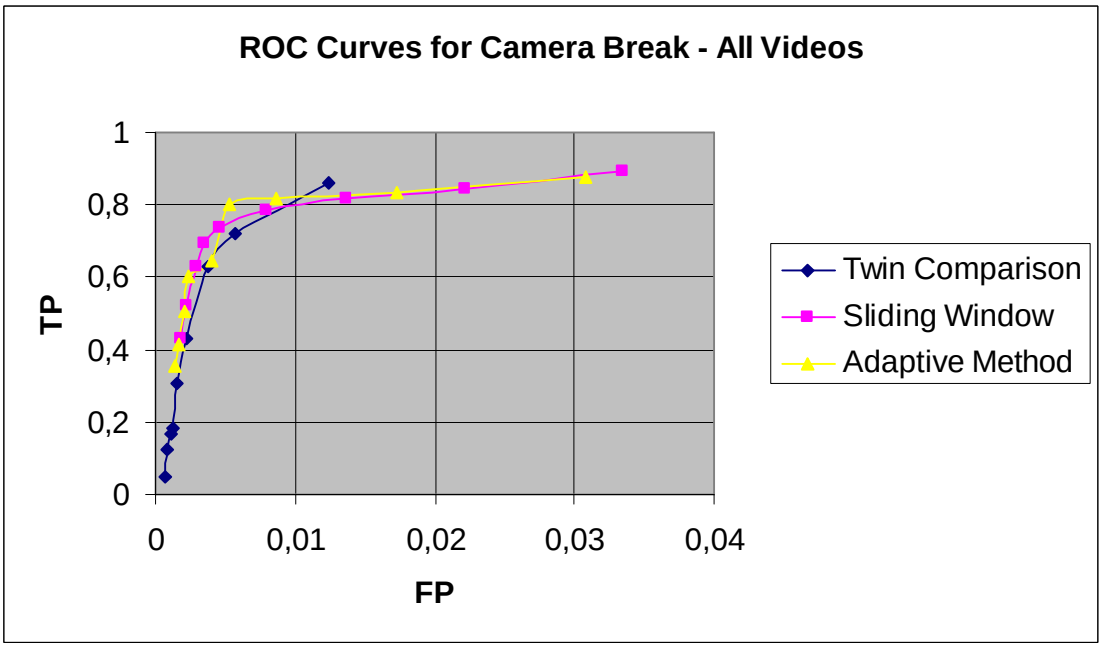


Figure 6.18: ROC curves for algorithms of Twin Comparison, Sliding Window and Adaptive Method for Camera Break detection in videos of all videos. We used variable a of threshold as a criterion for these curves. The values of a which were used are 2, 2.5, 3, 3.5, 4, 4.5, 5, 5.5 and 6.

X-axis is FP (False Positives) and y-axis is TP (True Positives). The mathematic formula is given by equations 6.1 and 6.2. We observe that the curves of Sliding Window and Adaptive Method for small values of a have almost the same ROC values. When a increasing, the two curves are directing to zero value (FP=0, TP=0). The derivative of the curve of Sliding Window is smaller than the derivative of Adaptive Method. The derivative of Twin Comparison is smaller than the derivative of Sliding Window. So, the algorithm with the best result is Adaptive Method then follows Sliding Window and Twin Comparison.

In the next figure, figure 6.19, we can see the ROC curves of Gradual Transition detection for all videos.

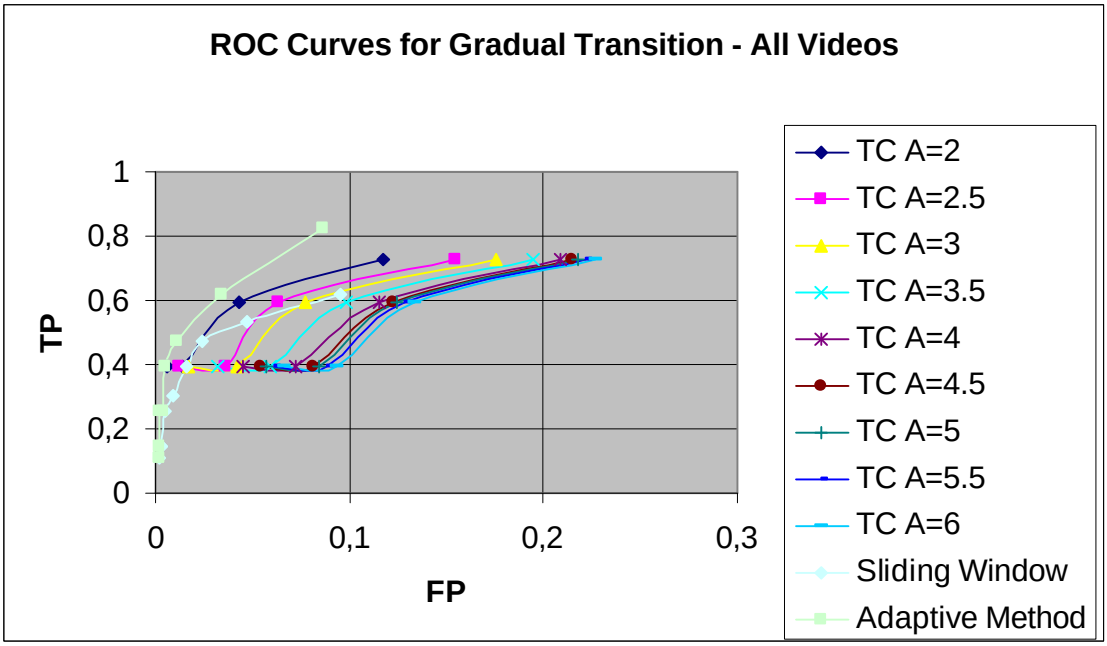


Figure 6.19: ROC curves for algorithms of Twin Comparison, Sliding Window and Adaptive Method for Gradual Transition detection in videos of all videos. We used variable a of threshold as a criterion for algorithms Sliding Window and Adaptive Method. For the algorithm of Twin Comparison we used variable b as a criterion for the curves. The values of b which were used are 1.5, 1.7, 1.9 and 2.1. For Twin Comparison we kept the variable a stable (which is used for Camera Break detection) and for several values of a we increase b .

X-axis is FP (False Positives) and y-axis is TP (True Positives). The mathematic formula is given by equations 6.3 and 6.4. We observe that the curve of Adaptive Method is above all the other algorithms, so it has the best results. Next follows the curve of Twin Comparison for $a = 2$. The curve of Sliding Window follows next and finally the rest curves. It is worth looking that while the curves of Sliding Window and Adaptive Method directed to zero value, the curves of Twin Comparison are directed to a stable value parallel to x-axis.

In the next figure, figure 6.20, we can see the ROC curves of Camera Motion Change detection for all videos.

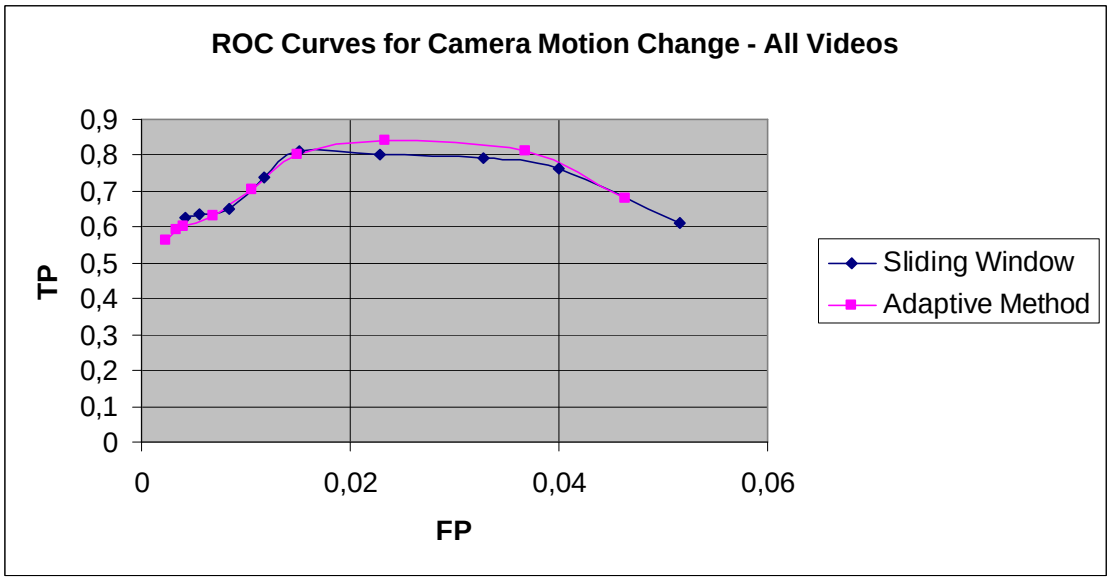


Figure 6.20: ROC curves for algorithms of Sliding Window and Adaptive Method for Camera Motion change detection in videos of all videos. We used variable a of threshold as a criterion for algorithms Sliding Window and Adaptive Method.

X-axis is FP (False Positives) and y-axis is TP (True Positives). The mathematic formula is given by equations 6.5 and 6.6. Observing the above two curves we can notice that for small values of a (the right most values) the value of ROC curve is small and while the a is increasing, the value of ROC curve increasing too. While a is increasing, the value of ROC curve is decreasing directed to zero value. While the a value is small, the threshold is also small and a lot of histogram difference values are bigger than threshold. In other words the duration is bigger than three frames and so these frames are false detected. While the value of a increasing, the threshold increasing too and we have less false detected frames and much more correct detected. For big values of a, the threshold is big enough than most of the histogram difference values, so we have not only less false detections but also less correct detection. For that reason the two curves are directed to zero value (FP=0, TP=0).

Conclusion and Future Work

In this work we implemented five algorithms for Video Segmentation. Three of the algorithms use the histogram intensity information of the video and the other two the motion information.

From the above results we can notice that the behavior of algorithms in four categories of video is different. For Camera Break Detection, Twin Comparison gives best results in categories of news and animation. Adaptive Method gives best results in the other two categories, sports and documentary. Sliding Window has generally good results except the category of animation. For Gradual Transition Detection, Twin Comparison gives best results in categories of news and sports. Adaptive Method gives best results in the remaining categories, documentary and animation. Sliding Window has good results. For Camera Motion Change Detection, Adaptive Method gives best results in categories of documentary, news and animation. Sliding Window gives best results in category of sport. In the overall results for Camera Break Detection, Adaptive Method gives best results. Adaptive Method gives also best results for Gradual Transition Detection and Camera Motion Change Detection.

It is worth mention that in ROC curves, of both Adaptive Method and Sliding Window, of Camera Motion Change Detection, for small values of a the curve falls, directing to TP=0. This is something that we don't want to happen. So we can put a threshold that a should not fall below this.

Generally the overall results are not so important as the results by category. The overall results give a general idea about the performance of the algorithms. On the other hand, the results by category give an idea about the performance of the algorithms for a specific portion of videos. If we had, for example, a Video database, we could categorize the videos while inserting them in the database. Then we could implement the appropriate algorithm in each category.

The above algorithms were tested separately for each video category. The combination of the above algorithms may lead to much better results, if the execution time is not so important factor for our application.

References

References for Chapter 1

1. Arun Hampapur, "Designing Video Data Management Systems", Doctor of Philosophy Thesis, University of Michigan, 1995
2. Anastasia Analyti and Stavros Christodoulakis, "Multimedia Object Modeling and Content-Based Querying", Proceedings of the Advanced Course Multimedia Databases in Perspective, University of Twente, pp. 213-238, 1995

References for Chapter 2

1. Arun Hampapur, "Designing Video Data Management Systems", Doctor of Philosophy Thesis, University of Michigan, 1995
2. Anastasia Analyti and Stavros Christodoulakis, "Multimedia Object Modeling and Content-Based Querying", Proceedings of the Advanced Course Multimedia Databases in Perspective, University of Twente, pp. 213-238, 1995

References for Chapter 3

1. S. Lefevre, J. Holler and N. Vincent, "A Review of Real-Time Segmentation of Uncompressed Video Sequences for Content Based Search and Retrieval", RFAI Publication, Real Time Imaging, To appear.
2. A. Nagasaka and Y. Tanaka, "Automatic Video Indexing and full-search for Object Appearances", In IFIP Working Conference on Visual Database Systems, pages 113-127, Budapest, Hungary, October 1991
3. W. Ren, M. Sharma and S. Singh, "Automated Video Segmentation", In International Conference on Information, Communication and Signal Processing, Singapore, October 2001
4. Y. Taniguchi, A. Akutsu and Y. Tonomura, "Panorama Experts: Extracting and Packing Panoramas for Video Browsing", In ACM International Conference on Multimedia, pages 427-436, Seattle, WA, November 1997
5. S. Lawrence, D. Ziou, M. F. Auclair-Fortier and S. Wang, "Motion Insensitive Detection of Cuts and Gradual Transitions in Digital Videos", To appear in Pattern Recognition Letters, 2002

6. Y. Tonomura and S. Abe, "Content Oriented Visual Interface Using Video Icons for Visual Database Systems", Journal of Visual Languages and Computing, 1(2):183-198, June 1990
7. U. Gargi and R. Kasturi, "An Evaluation of Color Histogram-Based methods in Video Indexing", In International Workshop on Image Databases and Multimedia Search, pages 75-82, Amsterdam, The Netherlands, August 1996
8. M. Ahmed, A. Karmouch and S. Abu-Hakima, "Key Frame Extraction and Indexing for Multimedia Databases", In Vision Interface Conference, pages 506-511, Trois-Rivieres, Canada, May 1999
9. C. O'Toole, A. Smeaton, N. Murphy and S. Marlow, "Evaluation of Automatic Shot Boundary Detection on a Large Video test suite", In Conference on Challenge of Information Retrieval, Newcastle, UK, February 1999
10. H. J. Zhang, A. Kankanhalli and S.W. Smoliar, "Automatic Partitioning of full motion Video", Multimedia Systems, 1(1):10-28, 1993
11. A. Dailianas, R. B. Allen and P. England, "Comparison of Automatic Video Segmentation Algorithms", In SPIE Conference on Integration Issues in Large Commercial Media Delivery Systems, vol. 2615, pages 2-16, Philadelphia, PA, October 1995
12. W. Zhao, J. Wang, D. Bhat, K. Sakiewics, N. Nandhakumar and W. Chang, "Improving Color-Based Video Shot Detection", In IEEE International Conference on Multimedia Computing and Systems, vol. 2, pages 752-756, Florence, Italy, June 1999
13. N. Haering, N. da Victoria Lobo, R. Qian and I. Sezan, "A framework for Designing Event Detectors", In Asian Conference on Computer Vision, Taipei, Taiwan, January 2000
14. O. Javed, S. Khan, Z. Rasheed and M. Shah, "A framework for Segmentation of Interview Videos", In IASTED International Conference on Internet and Multimedia Systems and Applications, Las Vegas, CA, November 2000
15. B. Gunsel, A. M. Ferman and A. M. Tekalp, "Temporal Video Segmentation Using Unsupervised Clustering and Semantic Object Tracking", Journal of Electronic Imaging, 7(3):592-604, July 1998
16. S. Kim and R. H. Park, "A novel Approach to Scene Change Detection Using a Cross Entropy", In IEEE International Conference on Image Processing, vol. 3, pages 937-940, Vancouver, Canada, September 2000
17. R. Kasturi and R. C. Jain, "Dynamic Vision", In R. Kasturi and R. C. Jain, editors, Computer Vision: Principles, pages 469-480, IEEE Computer Society Press, Washington, 1991
18. M. S. Lee, Y. M. Yang and S. W. Lee, "Automating Video Parsing Using Shot Boundary Detection and Camera Operation Analysis", Pattern Recognition, 34(3):711-719, March 2001

19. D. Swanberg, C. F. Shu and R. Jain, "Knowledge guided Parsing and Retrieval In Video Databases", In SPIE Conference on Storage and Retrieval for Image and Video Databases, vol. 1908, pages 13-24, San Jose, CA, February 1993
20. H. Ueda, T. Miyatake and S. Yoshizawa, "Impact: An Interactive Natural-Motion-Picture dedicated Multimedia Authoring System", In ACM Conference on Human Factors in Computing Systems, pages 343-350, New Orleans, LO, April 1991
21. M. Ahmed and A. Karmouch, "Video Segmentation using an Opportunistic approach", In Multimedia Modeling, pages 389-405, Ottawa, Canada, October 1999
22. C. M. Lee and M. C. Ip, "A Robust Approach for Camera Break Detection in Color Video Sequences", In IAPR International Workshop on Machine Vision Applications, pages 502-505, Kawasaki, Japan, December 1994
23. M. Bertini, A. Del Bimbo and P. Pala, "Content Based Indexing and Retrieval of TV news", Pattern Recognition Letters, 22(5):503-516, April 2001
24. Y. Chahir and L. Chen, "Automatic Video Segmentation and Indexing", In SPIE Conference on Intelligent Robots and Computer Vision, vol. 3837, pages 345-356, Boston, MA, September 1999
25. R. Dugad, K. Ratakonda and N. Ahuja, "Robust Video Shot Change Detection", In IEEE Workshop on Multimedia Signal Processing, pages 376-381, Rodondo Beach, CA, December 1998
26. D. A. Adjeroh, M. C. Lee and C. U. Orji, "Techniques for fast Partitioning of Compressed and Uncompressed Video", International Journal of Multimedia Tools and Applications, 4(2):225-243, March 1997
27. C. H. Demarty and S. Beucher, "Morphological Tools for Indexing Video Documents", In IEEE International Conference on Multimedia Computing and Systems, vol. 2, pages 991-992, Florence, Italy, June 1999
28. S. Lefevre, J. Holler and N. Vincent, "Real-Time Temporal Segmentation of Compressed and Uncompressed Dynamic Color Image Sequences", In International Workshop on Real Time Image Sequence Analysis, pages 56-62, Oulu, Finland, August 2000
29. W. Xiong and J. C. M. Lee, "Efficient Scene Change Detection and Camera Motion Annotation for Video Classification", Journal of Computer Vision and Image Understanding, 71(2):166-181, August 1998
30. F. Arman, R. Depommier, A. Hsu and M. Y. Chiu, "Content-Based Browsing of Video Sequences", In ACM International Conference of Multimedia, pages 97-103, San Francisco, CA, November 1994

31. R. Zabih, J. Miller and K. Mai, "A Feature-Based Algorithm for Detecting and Classifying Production Effects", Multimedia Systems, 7(2):119-128, March 1999
32. R. Lienhart, "Comparison of Automatic Shot Boundary Detection Algorithms", In SPIE Conference on Storage and Retrieval for Image and Video Databases, vol. 3656, pages 290-301, San Jose, CA, January 1999
33. H. Yu, G. Bozdagi and S. Harrigton, "Feature-Based Hierarchical Video Segmentation", In IEEE International Conference on Image Processing, vol. 2, pages 498-501, Santa Barbara, CA, October 1997
34. W. J. Heng and K. N. Ngan, "Integrated Shot Boundary Detection using Object-Based Technique", In IEEE International Conference on Image Processing, vol. 3, pages 289-293, Kobe, Japan, October 1999
35. J. Nam and A. H. Tewfik, "Combined Audio and Visual Streams Analysis for Video Sequence Segmentation", In International Conference on Acoustics, Speech and Signal Processing, vol. 4, pages 2665-2668, Munich, Germany, April 1997
36. J. Nam and A. H. Tewfik, "Wipe Transition Detection Using Polynomial Interpolation", In SPIE Conference on Storage and Retrieval for Media Databases, vol. 4315, pages 231-241, San Jose, CA, January 2001
37. M. Ardebilian, X. Tu and L. Chen, "Robust 3rd Clue-Based Video Segmentation for Video Indexing", Journal of Visual Communication and Image Representation, 11(1):58-79, March 2000
38. R. Silva, J. Gomes and L. Velho, "Segmentation of Video Sequences using Volumetric Image Processing", In Eurographics Workshop on Multimedia, Milan, Italy, September 1999
39. J. M. Sanchez, X. Binefa, J. Vitria and P. Radeva, "Local Color Analysis for Scene Break Detection applied to TV commercials recognition", In International Conference on Visual Information Systems, pages 237-244, Amsterdam, The Netherlands, June 1999
40. A. Hampapur, R. C. Jain and T. Weymouth, "Production Model-Based digital Video Segmentation", International Journal of Multimedia Tools and Applications, 1(1):9-46, March 1995
41. N. Adami and R. Leonardi, "Identification of Editing Effect in Image Sequences by Statistical Modeling", In Symposium on Picture Coding, pages 157-160, Portland, OR, April 1999
42. P. Aigrain and P. Joly, "The Automatic Real-Time Analysis of Film Editing and Transition Effects and Its Applications", Computer and Graphics, 18(1):93-103, January/February 1994

43. A. M. Alattar, "Detecting Fade Regions in Uncompressed Video Sequences", In International Conference on Acoustics, Speech and Signal Processing, pages 3025-3028, Munich, Germany, April 1997
44. B. T. Truong, C. Dorai and S. Venkatesh, "New Enhancements to Cut, Fade and Dissolve Detection Processes In Video Segmentation", In ACM International Conference on Multimedia, pages 219-227, Los Angeles, CA, October 2000
45. W. A. C. Fernando, C. N. Canagarajah and D. R. Bull, "Fade and Dissolve Detection in Uncompressed and Compressed Video Sequences", In IEEE International Conference on Image Processing, page 27AP2, Kobe, Japan, October 1999
46. N. Vasconcelos and A. Lippman, "Statistical Models of Video Structure for Content Analysis and Characterization", IEEE Transactions on Image Processing, 9(1):3-19, January 2001
47. S. H. Han, I. S. Kweon, C. Y. Kim and Y. S. Seo, "Bayesian Shot Detection using Structural weighting", In IAPR International Workshop on Machine Vision Applications, pages 95-98, Tokyo, Japan, November 2000
48. A. Hanjalic and H. J. Zhang, "Optimal Shot Boundary Detection based on Robust Statistical Models", In IEEE International Conference on Multimedia Computing and Systems, vol. 2, pages 710-714, Florence, Italy, June 1999
49. A. Yilmaz and M. Shah, "Shot Detection using Principle Coordinate System", In IASTED International Conference on Internet and Multimedia Systems and Applications, Las Vegas, CA, November 2000
50. K. J. Han and A. H. Tewfik, "Eigen-Image based Video Segmentation and Indexing", In IEEE International Conference on Image Processing, pages 538-541, Santa Barbara, CA, October 1997
51. Z. N. Li and J. Wei, "Spatio-Temporal joint Probability Images for Video Segmentation", In IEEE International Conference on Image Processing, vol. 2, pages 295-298, Vancouver, Canada, September 2000
52. Y. Gong and X. Liu, "Video Shot Segmentation and Classification", In IAPR International Conference on Pattern Recognition, vol. 1, pages 860-863, Barcelona, Spain, September 2000
53. S. Eickeler and S. Muller, "Content-Based Video Indexing for TV Broadcast News using Hidden Markov Models", In International Conference on Acoustics, Speech and Signal Processing, pages 2997-3000, Phoenix, AZ, March 1999
54. J. S. Boreczky and L. D. Wilcox, "A Hidden Markov Model Framework for Video Segmentation using Audio and Image features", In International Conference on Acoustics, Speech and Signal Processing, vol. 6, pages 3741-3744, Seattle, WA, May 1998

55. P. Bouthemy, M. Gelgon and F. Ganansia, "A Unified Approach to Shot Change Detection and Camera Motion Characterization", IEEE Transactions on Circuits and Systems for Video Technology, 9(7):1030-1044, October 1999
56. M. Cherfaoui and C. Bertin, "Temporal Segmentation of Videos: A New Approach", In SPIE Conference on Digital Video Compression: Algorithms and Technologies, vol. 2419, pages 38-47, San Jose, CA, February 1995
57. D. Zugaj and P. Bouthemy, "Wipe Detection in the Temporal Segmentation of Video", In European Workshop on Content Based Multimedia Indexing, Toulouse, France, October 1999
58. S. Mann and R. W. Picard, "Video Orbits of the Projective Group: A simple Approach to Featureless Estimation of Parameters", IEEE Transactions on Image Processing, 6(9):1281-1295, September 1997
59. A. Akutsu, Y. Tonomura, H. Hashimoto and Y. Ohba, "Video Indexing Using Motion Vectors", In SPIE Conference on Visual Communications and Image Processing, vol. 1818, pages 1522-1530, Boston, MA, November 1992
60. B. Shahraray, "Scene Change Detection and Content-Based Sampling of Video Sequences", In SPIE Conference on Digital Video Compression: Algorithms and Technologies, vol. 2419, pages 2-13, San Jose, CA, February 1995
61. T. Liu, X. Zhang, D. Wang, J. Feng and K. T. Lo, "Inertia-Based Cut Detection Technique: A step to the integration of Video Coding and Content-Based Retrieval", In IEEE International Conference on Signal Processing, pages 1018-1025, Beijing, China, August 2000
62. O. Fatemi, S. Zhang and S. Panchanathan, "Optical Flow based model for Scene Cut Detection", In Canadian Conference on Electrical and Computer Engineering, vol. 1, pages 470-473, Calgary, Canada, May 1996
63. S. V. Porter, M. Mirmehdi and B. T. Thomas, "Video Cut Detection using Frequency Domain Correlation", In IAPR International Conference on Pattern Recognition, vol. 3, pages 413-416, Barcelona, Spain, September 2000

References for Chapter 4

1. Farsid Arman, Arding Hsu and Ming-Yee Chiu, "Image Processing on Compressed Data for Large Video Databases", In Proceedings of the ACM Multimedia, pages 267-272, California, USA, June 1993, Association of Computing Machinery.
2. Akio Nagasaka and Yuzuru Tanaka, "Automatic Video Indexing and full Video Search for Object appearances", In 2nd Working

Conference on Visual Database Systems, pages 119-133, Budapest, Hungary, October 1991, IFIP WG 2.6
3. H. J. Zhang, A. Kankanhalli and S.W. Smoliar, "Automatic Partitioning of full motion Video", Multimedia Systems, 1(1):10-28, 1993
4. Akihito Akutsu and Yoshinobu Tonomura, "Video Tomography: An efficient method for Camerawork Extraction and Motion Analysis", In Proceedings Second Annual ACM Multimedia Conference, Association of Computing Machinery, October 1994
5. Arun Hampapur, Ramesh Jain and Terry Weymouth, "Digital Video Segmentation", In Proceedings 2nd Annual ACM Multimedia Conference and Exposition, Association of Computing Mahinery, October 1994
6. George Wolberg, "Digital Image Warping", IEEE Computer Society Press, 1992
7. Foley, VanDam, Feiner and Hughes, "Computer Graphics: Principles and Practice", The Systems Programming Series, Addison Wesley, 1990
8. Azriel Rosenfeld and Avinash C. Kak, "Digital Picture Processing Vol. 1, 2", Academic Press, 1976
9. David Bordwell and Thompson Kristin, "Film Art: An Introduction", Addison-Wesley Publishing Company, 1980
10. Richard O. Duda and Peter E. Hart, "Pattern Classification and Scene Analysis", A Wiley-Interscience Publication, John Wiley and Sons, 1973
11. Laurence Goldstein and Jay Kaufman, "Into Film", E. P. Duton and Co, 1976
12. Ishwar K. Sethi and Ramesh Jain, "Finding Trajectories of Feature Points in a Monocular Image Sequence", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 9 (No 1): 56-73, January 1987
13. Ramesh Jain, Rangachar Kasturi and Brian G. Schunck, "Introduction To Machine Vision", McGraw Hill, 1995
14. Zvi Kohavi, "Switching and Finite Automata Theory", Computer Science Series, McGraw Hill, 1978

References for Chapter 5

1. H. J. Zhang, A. Kankanhalli and S.W. Smoliar, "Automatic Partitioning of full motion Video", Multimedia Systems, 1(1):10-28, 1993
2. N. Wessel, A. Voss, H. Malberg, C. Ziehmann, H. U. Voss, A. Schirdewan, U. Meyerfeldt and J. Kurths, "Nonlinear Analysis of Complex Phenomena in Cardiological Data", Herzschr Elektrophys 11:159-173, Steinkopff Verlag, 2000

3. Cristoph Stiller and Janusz Konrad, "Estimating Motion In Image Sequences: A Tutorial on Modeling and Computation of 2D Motion", IEEE Signal Processing Magazine, pages 70-91, July 1999

References for Chapter 6

1. David Heeger, "Signal Detection Theory", Handout for New York University Department of Psychology and Neural Science, November 1997
2. http://www.anaesthetist.com/mnm/stats/roc/
3. http://www.poems.msu.edu/InfoMastery/Diagnosis/ROC.htm