Multimedia Tools and Applications, 1, 9–46 (1995)
© 1995 Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.

Production Model Based Digital Video Segmentation

ARUN HAMPAPUR, RAMESH JAIN* AND TERRY E WEYMOUTH arun@eecs.umich.edu
Computer Science and Engineering, Department of Electrical Engineering and Computer Science, University of Michigan, 1101 Beal Ave, Ann Arbor, MI 48109-2110

Abstract. Effective and efficient tools for segmenting and content-based indexing of digital video are essential to allow easy access to video-based information. Most existing segmentation techniques do not use explicit models of video. The approach proposed here is inspired and influenced by well established video production processes. Computational models of these processes are developed. The video models are used to classify the transition effects used in video and to design automatic edit effect detection algorithms. Video segmentation has been formulated as a production model based classification problem. The video models are also used to define segmentation error measures. Experimental results from applying the proposed technique to commercial cable television programming are presented.

Keywords: Digital Video, Video Segmentation, Video Indexing, Video Databases, Edit Effects, Fade In, Fade Out, Dissolve, Editing, Content based retrieval

1. Introduction

The progress of computer systems towards becoming true multimedia machines depends largely on the set of tools available for manipulating the different components of multimedia. Of these components digital video is the most data intensive. Effective tools for segmenting and content based indexing of digital video are essential in the design of the true multimedia machine. This paper presents an innovative and novel approach to modeling and segmenting digital video based on video production techniques.

The problem of segmenting digital video occurs in many different application of video. Content based access of video in the context of video databases requires access to video in a more natural unit than frames. Video segments based on edit locations provides a higher level of access to video than frames. Multimedia authoring systems which reuse produced video need access to video in terms of video shots. The edit detection algorithms presented in this paper 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 [1], [19], [24], [22] do not use explicit models. 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

* Currently at the University of California, San Diego, La Jolla, CA 92093-0407

structure provides the constraints necessary for effective video segmentation. Hampapur et al [10] have presented initial results on model based video segmentation. The work presented in this paper uses the *production model based classification* approach to video segmentation.

This paper 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. Experimental results from segmenting commercial cable television data are presented.

Section 2 discusses modeling of digital video based on video production techniques and proposes a classification of edits. Video segmentation is defined in section 3. The formulation of video segmentation as production model based classification is discussed in section 4. This section also presents the design of feature extractors. The classification and segmentation steps are discussed in 5. Section 6 presents a comparison of this work to other research in the field. Segmentation error measures are formulated in section 7. Section 8 presents the experiments performed and the results obtained. A summary of the work and future directions concludes the paper.

2. Modeling Digital Video

Video (the term video is used generically to cover movies, video, cable television programming etc) is a means of storing and communicating information. There are many different ways in which video is used and many different aspects of video [8], [14]. The two most essential aspects of video are *the content* and *the production style*. The content of video is the information that is being transmitted through the medium of video. The production style is the encoding of the content into the medium of video. The production style of video is the aspect which is directly relevant to the problem of segmenting digital video.

The process of producing a video involves two major steps, the production of shots (a sequence of frames generated by a single operation of the camera [17]), *shooting* and the compilation of the different shots into a *structured audio visual* presentation, *editing* [17], [2]. In order to segment video it is essential to have a computational model of the editing process. Figures 1, 2 illustrates the process of editing. The two phases are as follows:

Editing 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 or a model called the *Edit Decision List*[6].

Assembly This is the physical process which the *Edit Decision List* is converted into frames on the *final cut* [2]. This is the process involves taking the shots from the

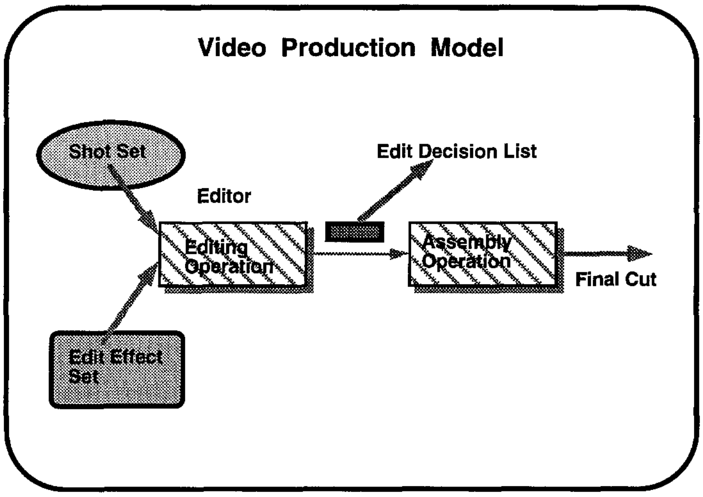


Figure 1. Video Production Model

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.

2.1. Video Edit Model

The *video edit model* presented here captures the process of editing and assembly discussed in the previous section. 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 $S = (S_1, S_2, ...S_N)$ with cardinality N. Each shot $S_i \in S$ can be represented by a closed time interval:

$$S_i = [t_b, t_e] \tag{1}$$

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 $E = (E_1, E_2, ...E_k)$ be the set edits available. Each edit E is represented by a triple:

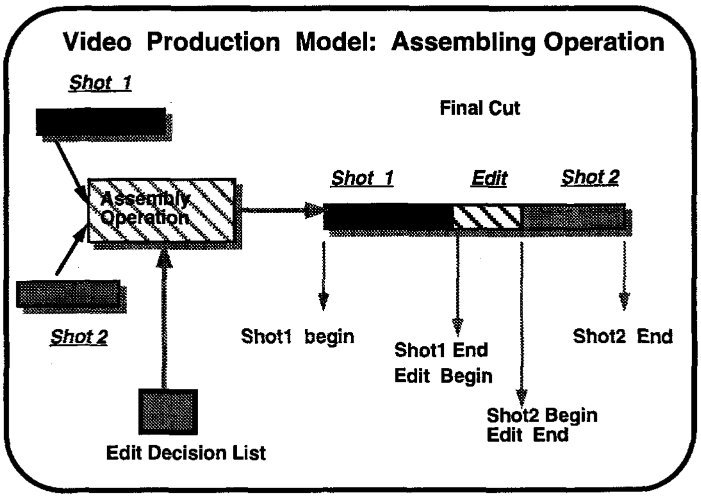


Figure 2. Video Production Model: Assembly Operation

$$E = \{[t_b, t_e], \tau, e\} \tag{2}$$

where $[t_b, t_e]$ is the duration of the edit, $\tau \in (\tau_1, \tau_2, ..\tau_n)$ is the type of the edit (typical examples include *cuts, fades, dissolves*) and e is the 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.

$$\begin{aligned} V_{edm} = \; & S_1 \circ E_{12}(S_1, S_2) \circ S_2 \circ E_{23}(S_2, S_3) \circ \\ & \ldots \circ S_{(n-1)} \circ E_{(n-1)n}(S_{n-1}, S_n) \circ S_n \end{aligned} \tag{3}$$

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:

$$V_{am} = S_1 \circ S_{e12} \circ S_2 \circ S_{e23} \circ \ldots \circ S_{(n-1)} \circ S_{e(n-1)n} \circ S_n \tag{4}$$

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:

$$\begin{aligned} V_c = s_1 \circ s_2 \circ s_3 \circ s_4 \circ \ldots \circ s_{(n-1)} \circ s_n \circ s_{(n+1)} \circ s_{(2n-1)} \\ s_i = \{[t_b, t_e], l_i\} \\ l_i \in (edit, shot) \end{aligned} \tag{5}$$

Table 1. Edit Types

Type	Name	T_s	T_c	Meaning	Duration	Examples
1	Identity	$\vec{I}$	$\vec{I}$	Concatenate Shots	Zero	Cut
2	Spatial	T_s	$\vec{I}$	Manipulate shot pixel space	Finite	Translate, Page
3	Chromatic	$\vec{I}$	T_c	Manipulate shot intensity space	Finite	Fade, Dissolve
4	Combined	T_s	T_c	Manipulate pixels and intensities	Finite	Morphing

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 in the later sections.

2.2. *Edit Effect Model*

The edit effects commonly used in video production are modeled by using **2D image transformations** [23]. The mathematical model for the process of generating *edit frames* from a pair of shots is discussed in this section. 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 $\vec{\xi} = \{x, y, t, 1\}$ and the color space be denoted by $\vec{\eta} = \{r, g, b, 1\}$. *Homogeneous Coordinates* [5] 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.

$$E(x, y, t) = S_{out}(\vec{\xi} \times T_{s1}) \times T_{c1} \otimes S_{in}(\vec{\xi} \times T_{s2}) \times T_{c2} \quad (6)$$

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

The classification presented in table 1 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.

3. Video Segmentation: Problem Definition

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 3) composed from n shots,

$$\forall\ S_i \in V \text{ locate } [t_b, t_e] \text{the extremal points of the shot} \tag{7}$$

Edit Boundary Detection: Given a video V (equation 3) composed from n shots,

$$\forall\ E_i \in V \text{ locate } [t_b, t_e] \text{the extremal points of the edit} \tag{8}$$

The above two definitions are equivalent as *edits* and *shots* form a partition of the video. These two definitions are analogous to the the *region growing* vs *edge detection* approaches to image segmentation [20].

The techniques proposed in this paper treat video segmentation as *edit boundary detection*. The reasons for this choice 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 [2]. 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 [8], [9].

4. 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 [3]. The *video edit model* (equation 3,4,5) and the *edit effect models* (equation 6) are the *data production models* in video segmentation.

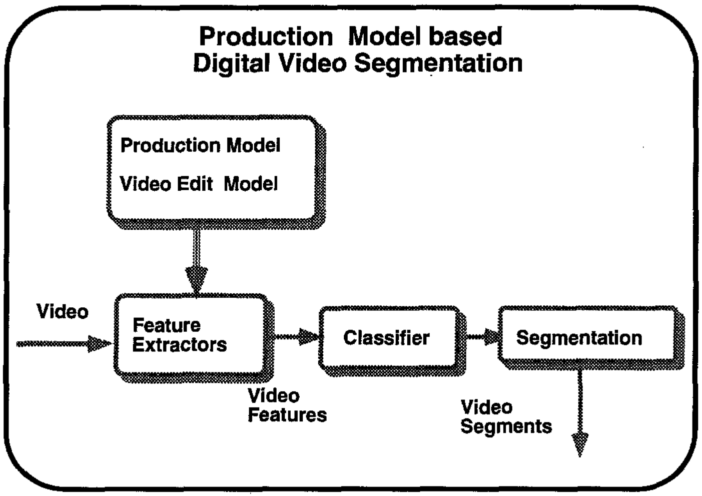


Figure 3. Production Model based Video Segmentation: Block Diagram

Figure 3 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 [3] 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 1) are the classes of interest. Feature extractors are designed for each of these classes (in this work only the first three classes are used). 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.

The following subsections present the different steps of the formulation. Section 4.1 presents the details of the *identity edit class.* The various details of the chromatic edit class are discussed in section 4.2. The spatial edit class is presented in section 4.4. The computational requirements of the feature detectors and a summary are presented in sections 4.6, 4.7. Section 5 discusses the classification and segmentation process.

4.1. 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*[2], [7] 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 [1], [19], [24], [22] 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 [21] which can be used to identify discontinuity of feature tracks in image sequences and

Table 2. Cut features

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

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. Work on modeling video for the purpose of video analysis is presented in [8]. Cut detectors used in this work are based on results presented by other researchers [1], [19], [24], [22] in the field. The operators presented by Nagasaka et al and Smoliar et al were used in an extensive experimental study using cable TV data.

Table 2 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). The two features shown in table 2 were used in the development of the segmentation system presented in this work.

4.2. *Chromatic Edit Detector*

Chromatic editing manipulates the color space of the two shots being edited (table 1). 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 discussion.

4.2.1. *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. The following discussion presents 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:

$$E(x,y,t) = g(x,y,t) \times \left(1 - \frac{t}{l_s}\right) \tag{9}$$

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 9 can be written as 10 where k indicates that the k^{th} frame of the shot is being used in the scaling.

$$E(x,y,t) = g(x,y,k) \times \left(\frac{l_s - t}{l_s}\right) \tag{10}$$

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

$$\frac{\delta E}{\delta t} = \frac{-g(x,y,k)}{l_s} \tag{11}$$

Equation 11 can be rewritten as

$$\mathcal{CI}(t) = \frac{\frac{\delta E}{\delta t}}{g(x,y,k)} = \frac{-1}{l_s} \tag{12}$$

where $\mathcal{CI}$ *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 $\mathcal{CI}$ (section 4.5).

$$F_{cs}(t) = \mathcal{U}(\mathcal{CI}(t)) \tag{13}$$

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.

4.3. *Fades and 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 modelled as chromatic scaling operations with a positive and negative fade rate. E_{fo} equation 14 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 6 and 14, for a fade out one of the shots $S_{out} = g_1$, $S_{in} = \vec{0}$ and $\otimes = +$. Similarly, E_{fi} (equation 15) 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 (14, 15) represent how the fade operation maps on to the edit effect model (equation 6). Since the operations on the individual sequences in the fade are chromatic scaling operations the chromatic scaling feature 12 can be used for detecting fades in videos.

$$E_{fo}(x,y,t) = g_1(x,y)\left(\frac{l_1 - t}{l_1}\right) + \vec{0} \tag{14}$$

$$E_{fi}(x,y,t) = \vec{0} + g_2(x,y)\left(\frac{t}{l_2}\right) \tag{15}$$

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 (16) models the process of dissolving two shots.

$$E_d(x,y,t) = g_1(x,y)\left(\frac{l_1 - t}{l_1}\right)\Bigg|_{(t_1,t_1+l_1)} + g_2(x,y)\left(\frac{t}{l_2}\right)\Bigg|_{(t_2,t_2+l_2)} \tag{16}$$

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 (15,14) and equation 16 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 13) 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 detectability of each group using the chromatic scaling operator.

Figure (4) 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.

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 the shot s, where (t_1, t_2) are the times at which **Out, In** shots begin scaling.

$$s = \textbf{In} \text{ if } t_1 > t_2 \Rightarrow \text{Fade In } \textbf{begins before} \text{ Fade Out} \tag{17}$$

$$s = \textbf{Out} \text{ if } t_1 < t_2 \Rightarrow \text{Fade Out } \textbf{begins before} \text{ Fade In.} \tag{18}$$

$$s = \textbf{Both} \text{ if } t_1 = t_2 \Rightarrow \text{Fade In, Fade Out } \textbf{begin together}. \tag{19}$$

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

$$s = \textbf{In} \text{ if } l_1 < l_2 \Rightarrow \text{In Shot } \textbf{dominates dissolve} \tag{20}$$

$$s = \textbf{Out} \text{ if } l_1 > l_2 \Rightarrow \text{Out Shot } \textbf{dominates dissolve} \tag{21}$$

$$s = \textbf{Equal} \text{ if } l_1 = l_2 \Rightarrow \text{No Shot } \textbf{dominates dissolve} \tag{22}$$

A shot is said to dominate the dissolve if its activity slope is higher. In other words, if the shot contributes more to the *interframe change [12], [13]* in the video sequence.

From figure 4 it can seen:

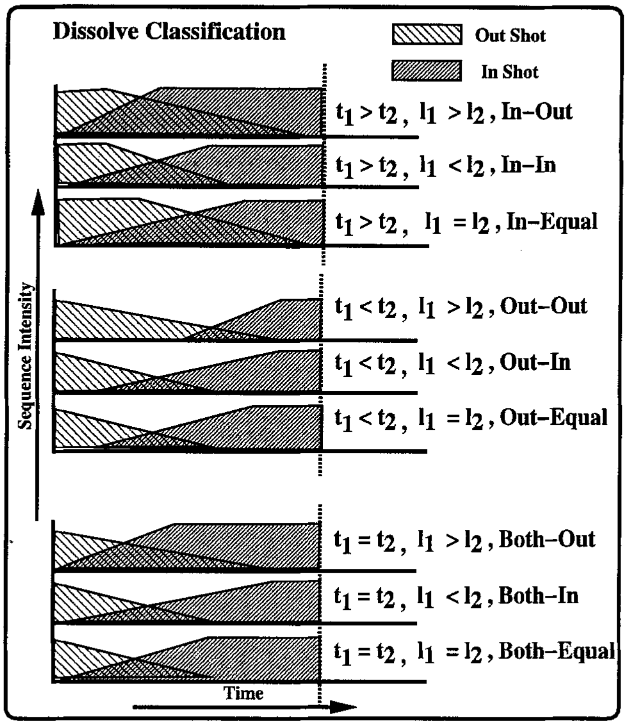
1. that except in the case of **Both-Equal** type of dissolves, all the other types have portions during which there is an exclusively single sequence chromatic scaling in progress.
2. that except in the case of **Equal Dominance Sequences** the change contribution of one sequence dominates the other.

Thus the cases in which the chromatic scaling feature (equation 13) 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. The experimental studies conducted confirm these observations over a wide range of commercial video.

4.3.1. Limitations of Chromatic Edit Detector

There are several limitations of the chromatic edit detector:

Additional Chromatic Transforms: The chromatic edit detector presented can be used for detecting chromatic edits which are scalings of the color space. There are other



Dissolve Classification
Out Shot
In Shot
$t_1 > t_2$, $l_1 > l_2$, In-Out
$t_1 > t_2$, $l_1 < l_2$, In-In
$t_1 > t_2$, $l_1 = l_2$, In-Equal
$t_1 < t_2$, $l_1 > l_2$, Out-Out
$t_1 < t_2$, $l_1 < l_2$, Out-In
$t_1 < t_2$, $l_1 = l_2$, Out-Equal
$t_1 = t_2$, $l_1 > l_2$, Both-Out
$t_1 = t_2$, $l_1 < l_2$, Both-In
$t_1 = t_2$, $l_1 = l_2$, Both-Equal
Sequence Intensity
Time

Figure 4. Sequence Activity Graph during a dissolve

possible types of chromatic transforms like chromatic translations, rotations etc which have not been addressed in this work. However this is not a serious practical limitation as chromatic transformations other than scaling are seldom used in practice.

Multiple Sequence Scalings: The scaling detector is primarily designed to detect the scaling of single image sequences. It can be used to detect two sequence scalings as in the case of dissolves provided that the effect of *sequence domination* is present (section 4.3). 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.

4.4. Spatial Edit Detector

Spatial edits are achieved by transforming the pixel space of the shots being edited. These are **Type 2** transforms in table 1 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 preceeding the edit is translated out of the viewport 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 α_x, α_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

$$\begin{aligned} E(x,y,t) &= g_{out}(x+\alpha_x \cdot t, y+\alpha_y \cdot t, t) \\ &\quad \text{if } 0 \le x+\alpha_x \le N_x \text{ and } 0 \le x+\alpha_y \le N_y \\ &= g_{in}(x,y,t) \text{ else} \end{aligned} \tag{23}$$

Equation 23 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 [18], [4], [11] for the edit.

$$\frac{dE}{dt} = 0 \tag{24}$$

Using the chain rule for differentiation equation 24 can be rewritten as equation 25.

$$\frac{dE}{dt} = \frac{\delta E}{\delta x} \cdot \frac{dx}{dt} + \frac{\delta E}{\delta y} \cdot \frac{dy}{dt} + \frac{\delta E}{\delta t} = 0 \tag{25}$$

substituting for E from equation 23 in equation 25 and assuming that the motion in the incoming shot is negligible as compared to the motion due to the edit ($\frac{\delta g_{in}}{\delta t} = 0$) the following equation can be written.

$$\frac{dE}{dt} = \frac{\delta E}{\delta x} \cdot \frac{d(x+\alpha_x \cdot t)}{dt} + \frac{\delta E}{\delta y} \cdot \frac{d(y+\alpha_y \cdot t)}{dt} + \frac{\delta E}{\delta t} = 0 \tag{26}$$

which can be rewritten as

$$\frac{dE}{dt} = \frac{\delta E}{\delta x}\left(\frac{dx}{dt} + \alpha_x\right) + \frac{\delta E}{\delta y}\left(\frac{dy}{dt} + \alpha_y\right) + \frac{\delta E}{\delta t} = 0 \tag{27}$$

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 27 can be rewritten as follows:

$$\frac{\delta E}{\delta x} \cdot \alpha_x + \frac{\delta E}{\delta y} \cdot \alpha_y + \frac{\delta E}{\delta t} = 0 \tag{28}$$

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

$$\mathcal{SI}(t) = \alpha_x = \frac{-\frac{\delta E}{\delta t}}{\frac{\delta E}{\delta x}} \tag{29}$$

Equation 29 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 $\mathcal{SI}$, the *spatial image.* Let F_{sgx} represent the spatial translate feature.

$$F_{sgx} = \mathcal{U}(\mathcal{SI}(t)) \tag{30}$$

where $\mathcal{U}$ denotes the measure of uniformity of the scaled difference images (section 4.5). 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 translate 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 are 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.

4.4.1. Example of spatial edit detector operation

This section presents a simple example to illustrate the operation of the spatial edit detector. Let S_{in}, S_{out} be the frames from the two shots being edited using a *spatial x-translate* effect. Here the incoming shot is in the background and the out going shot moves out to the right. Let $E(0), E(1), E(2)$ be the edit frames generated by the editing, it can be seen that the pixels of the *out shot* move to the right uncovering pixels of the *in shot.* Let the spatial translate detector be applied to the edit frames at $E(1)$. Now $\frac{\delta E}{\delta t} = E(1) - E(0)$ and $\frac{\delta E}{\delta x} = E(x+1, y, 1) - E(x, y, 1)$. From table 3 we see the $\frac{\delta E}{\delta t} = \frac{\delta E}{\delta x}$, hence the ratio $\frac{\frac{\delta E}{\delta t}}{\frac{\delta E}{\delta x}}$ is a constant. For the spatial edit detector to be applicable to any translate edit, the motion of the frame must be lesser than the sampling interval in the gradient computation. This can be ensured by processing the images at an appropriately reduced spatial resolution.

Table 3. Translate Edit Example

$S_{in} = \begin{array}{\|c\|c\|c\|}\hline I_{11} & I_{12} & I_{13} \\ \hline I_{21} & I_{22} & I_{23} \\ \hline I_{31} & I_{32} & I_{33} \\ \hline\end{array}$	$S_{out} = \begin{array}{\|c\|c\|c\|}\hline O_{11} & O_{12} & O_{13} \\ \hline O_{21} & O_{22} & O_{23} \\ \hline O_{31} & O_{32} & O_{33} \\ \hline\end{array}$	
$E(0) = \begin{array}{\|c\|c\|c\|}\hline O_{11} & O_{12} & O_{13} \\ \hline O_{21} & O_{22} & O_{23} \\ \hline O_{31} & O_{32} & O_{33} \\ \hline\end{array}$	$E(1) = \begin{array}{\|c\|c\|c\|}\hline I_{11} & O_{11} & O_{12} \\ \hline I_{21} & O_{21} & O_{22} \\ \hline I_{31} & O_{31} & O_{32} \\ \hline\end{array}$	$E(2) = \begin{array}{\|c\|c\|c\|}\hline I_{11} & I_{21} & O_{11} \\ \hline I_{21} & I_{22} & O_{21} \\ \hline I_{31} & I_{32} & O_{31} \\ \hline\end{array}$
$\frac{\delta E}{\delta t}(1) = \begin{array}{\|c\|c\|c\|}\hline I_{11} - o_{11} & O_{11} - O_{12} & O_{12} - O_{13} \\ \hline I_{21} - O_{21} & O_{21} - O_{22} & O_{22} - O_{23} \\ \hline I_{31} - O_{31} & O_{31} - O_{32} & O_{32} - O_{33} \\ \hline\end{array}$	$\frac{\delta E}{\delta x}(1) = \begin{array}{\|c\|c\|c\|}\hline I_{11} - o_{11} & O_{11} - O_{12} & O_{12} - O_{13} \\ \hline I_{21} - O_{21} & O_{21} - O_{22} & O_{22} - O_{23} \\ \hline I_{31} - O_{31} & O_{31} - O_{32} & O_{32} - O_{33} \\ \hline\end{array}$	

4.4.2. *Limitations of Spatial Edit Detector*

There are several limitations of the spatial edit detector some of which are design limitations and others which are practical limitations of the detectors.

Other Spatial Transforms: The detector designed according to the previous section are capable of detecting spatial translates in various directions. However with the large scale use of digital video editors the number of different types of spatial transition effects between shots is growing. Designing detectors for many of these effects is not possible. In addition it is conceivable that in future videos will have edit information coded along with the SMPTE time code that is currently used.

Peicewise Transforms: Similarly, the application of varying spatial transforms to subregions of the image makes the design of detectors more complex. And since the properties being measured are no longer global, the detectors tend to be less reliable.

4.5. *A measure of Image Uniformity*

The output of the image manipulation operations both in the case of the *chromatic edit detector* equation 13 and the *spatial edit detector* equation 30 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 $\mathcal{XI}$ represent the image whose constancy is being measured, where $\mathcal{XI} = \mathcal{CI}$ for the chromatic scaling detector and $\mathcal{SI}$ for the spatial operator. There are two aspects that can be measured from $\mathcal{XI}$

Spatial Uniformity For the ideal case where $\mathcal{XI} = \mathcal{K}$ $\forall_{\S,\dagger}$ *all the pixels in the image will have the same value.* In the case of a real image the $\mathcal{XI}(\S, \dagger)$ is valid only if the difference pixel at that point is non-zero, because a constant set of frames in the video will yield $\mathcal{XI} = 7$. 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 $\mathcal{XI}$. 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 $\mathcal{XI} = \mathcal{K}$ $\forall_{\S,\dagger}$ the variance of $\mathcal{XI}$ 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 $\mathcal{U}$ shown in equation 31 is the measure of image constancy. The meaning of the different symbols and their computational formulas are presented in table 4.

$$\text{Uniformity Measure} = \mathcal{U} = \frac{\mathcal{A}(T(D(I(t)), 0)}{\sigma(V_{cx}(I(t))) \cdot (1.0 + \mathcal{C}(T(D(I(t), 0))) - (c_x, c_y))} \tag{31}$$

4.6. Computational Requirements of Feature Detectors

This section 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 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. More detailed discussions on each of the operations listed below can be found in [15]

From table 4 the cost of computing all the three features for detecting edits is $\Lambda_{cut} + \Lambda_{chrom} + \Lambda_{space}$ which is $\mathcal{O}(\mathcal{N})$. Thus the total feature computation cost for all the three types of edit features is $\mathcal{O}(n)$. In the experimental system that has been implemented using this approach, all the frame processing is done on reduced resolution images with about 20K pixels per image. This computational requirement of $\mathcal{O}(10^4)$ is one order of magnitude larger than the requirement reported by Arman [1]. However their technique does deals only with cuts and does not consider other types of transition effects. The current set of algorithms can be further optimized to improve the performance.

4.7. Feature Detector Summary

The above sections presented a systematic approach to designing features. The feature design was based on the edit effects models. Figure 5 shows a flow diagram for extracting

Table 4. Computational Cost Table

Operation	Symbol	Computation Formula	Cost	Order
Histogram	$H(I(t))$	$\forall_{x,y} H(I(x,y,t)) ++$	$N * (k_{add})$	$\mathcal{O}(\mathcal{N})$
Difference	$\mathcal{D}(I(t))$	$\forall_{x,y} I(x,y,t+1) - I(x,y,t)$	$N * (k_{sub} + k_{abs})$	$\mathcal{O}(\mathcal{N})$
Division	$\mathcal{S}(I(t))$	$\forall_{x,y} I(x,y,t+1) \div I(x,y,t)$	$N * (k_{div})$	$\mathcal{O}(\mathcal{N})$
Gradient	$\frac{\delta}{\delta x}(I(t))$	$\forall_{x,y} I(x+1,y,t) - I(x,y,t)$	$N * (k_{sub})$	$\mathcal{O}(\mathcal{N})$
Threshold	$T(D(t),\mathcal{T})$	$\forall_{x,y} B(x,y,t) = 1$ if $I(x,y,t) \geq \mathcal{T}$	$N * k_{comp}$	$\mathcal{O}(\mathcal{N})$
Mean	$\mu(I(t))$	$\frac{\sum_{x,y} I(x,y,t)}{N}$	$N * k_{add} + k_{div}$	$\mathcal{O}(\mathcal{N})$
Variance	$\sigma(I(t))$	$\sum_{x,y}(I(x,y,t) - \mu(I(t)))^2$	$N * (k_{mult} + K_{sub})$	$\mathcal{O}(\mathcal{N})$
Area	$\mathcal{A}(B(t))$	$\sum_{x,y} B(x,y,t)$	$N * k_{sum}$	$\mathcal{O}(\mathcal{N})$
Centroid	$\mathcal{C}(\mathcal{B}(\sqcup))$	$C_x = \frac{\sum_{x,y} x\, I_d(x,y)}{\mathcal{A}}$ $C_y = \frac{\sum_{x,y} y\, I_d(x,y)}{\mathcal{A}}$	$N * (2 * k_{mult}) + k_{div}$	$\mathcal{O}(\mathcal{N})$
Chromatic Img	$\mathcal{CI}$	equation 12	$\Lambda_{diff} + \Lambda_{div}$	$\mathcal{O}(\mathcal{N})$
Spatial Img	$\mathcal{SI}$	equation 29	$\Lambda_{diff} + \Lambda_{grad} + \Lambda_{div}$	$\mathcal{O}(\mathcal{N})$
Uniformity	$\mathcal{U}(\mathcal{XI},\mathcal{D})$	equation 31	$\Lambda_{diff} + \Lambda_{thresh} +$ $\Lambda_{area} + \Lambda_{var} +$ $\Lambda_{centroid} + \Lambda_{thresh} +$ $k_{div} + k_{add}$	$\mathcal{O}(\mathcal{N})$
Cut	$\mathcal{F}_{cut}$	$\sum_{i=0}^{G} \frac{(H_t(i) - H_{t+1}(i))^2}{H_{t+1}(i)}$	$3 * N * k_{add} +$ $G * (k_{add} + k_{div} + k_{mult})$	$\mathcal{O}(\mathcal{N} + \mathcal{G})$
Chromatic	$\mathcal{F}_{chrom}$	$\mathcal{U}(\mathcal{CI})$	$\Lambda_{uniformity} + \Lambda_{\mathcal{CI}}$	$\mathcal{O}(\mathcal{N})$
Spatial	$\mathcal{F}_{space}$	$\mathcal{U}(\mathcal{SI})$	$\Lambda_{uniformity} + \Lambda_{\mathcal{SI}}$	$\mathcal{O}(\mathcal{N})$

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 χ^2 comparison of the histograms. The *chromatic* feature requires the computation of a difference image, an image division and a 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 Computations: 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.

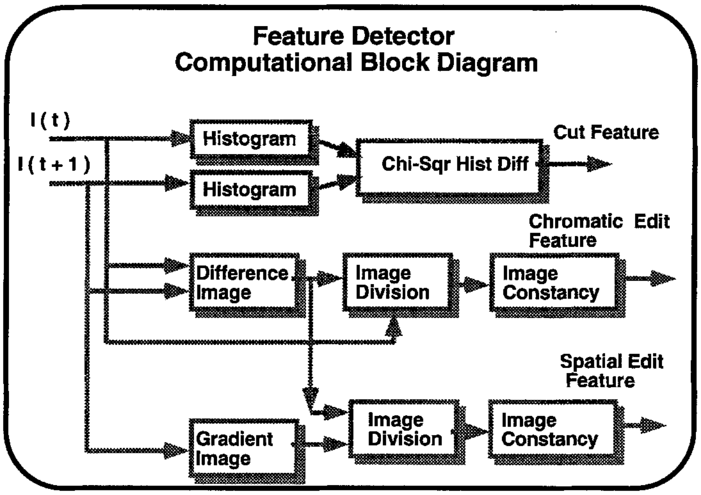


Figure 5. Feature Extractors: Computational Block Diagram

5. Classification and Segmentation

The features extracted from the video undergo several steps of processing before the video segments are identified. This section presents the various steps involved. Figure 6 provides an illustration of the details involved in the classification and segmentation process. A modification of a standard two class discrete classifier [3] 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.

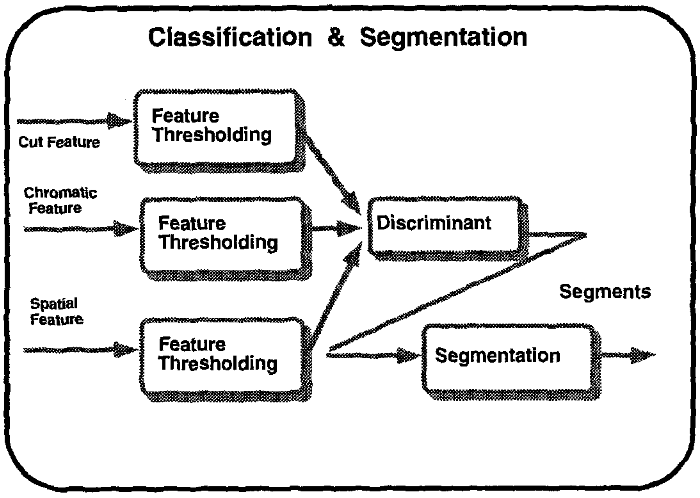


Figure 6. Steps in classification and segmentation

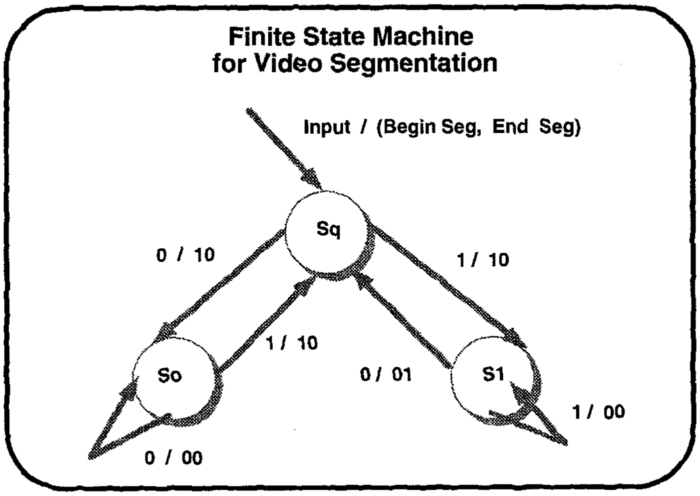

Finite State Machine for Video Segmentation
Input / (Begin Seg, End Seg)
Sq
0 / 10
1 / 10
So
1 / 10
0/ 01
S1
1 / 00
0 / 00

Figure 7. Finite State Machine for Segmentation

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 6. The notation used is the standard notation of finite state machines[16]. The circles indicate states and the arrows indicate the transition between states. The machine for segmentation has 3 states S_q, S_0 and S_1 where S_q is the quiescent state, S_0 is the *shot segment state*, S_1 is the *edit segment state*. The machine has two outputs namely *begin segment, end segment*.

6. Comparison to existing work

Most of the existing work in the area of video segmentation has used a *bottom up approach* to the problem of segmentation resulting in segmentation being viewed as a problem of effect detection. The lack of explicit edit effect models has been instrumental in the limited success that other techniques have had with detecting *extended edit effects*. The work presented in this paper has taken the *top down approach* to video segmentation resulting in the use of effect models which have allowed the design of simple and effective techniques for segmenting digital video.

Comparison of cut detection techniques in videos essentially has to depend on rigorous experimental comparisons performed on well characterized data, due to the fact that cuts are *null edits*. Such a comparison is not possible at this time. Hence the comparison presented below compares the proposed segmentation technique without considering cut detection.

Zhang et al [24]: This work addresses the problem of cut detection and the problem of detecting extended edit effects in video. Their approach to extended edit effect detection suffers from several draw backs. The principal problem is that they do not have an explicit model of the effects. The effect is defined in terms of the interframe difference measure. The solution proposed is based on a *two pass, dual threshold* algorithm. They have also used optical flow based computations for detecting other imaging effects like zooms and pans. Contrasted with the approach presented in this work, which uses models of the effects resulting in the systematic design of effect detectors. The approaches presented in this paper rely on the edit effect models to design detectors which use consecutive pairs of frames to detect extended effects. The detectors designed in this work provide *single pass, two frame* analysis approach to detecting effects as opposed to the *two pass, dual threshold, multi-frame* analysis approach proposed by Zhang et al. The approaches presented in this work are computational much more efficient than the approach presented by Zhang et al.

Nagasaka et al [19]: The work presented detects cuts in the context of partitioning video to perform search on the segments for certain objects. The paper presents an excellent comparison of various frame comparison algorithms. This work however does not address the complete segmentation problem as it does not consider the detection of other extended scene transition effects.

Akutsu et al [22]: This work addresses video segmentation as a part of the larger problem of structured video computing. They use an interframe differencing operation for detecting cuts. The authors point out that they perform gradual transition detection by performing frame comparisons over extended time intervals. However they do not provide details on the exact nature of the algorithm.

Arman et al [1]: This work on has adopted the novel approach of dealing with video in the compressed form. Thus the techniques developed here are very computationally efficient. They have presented techniques for detecting cuts based on the inner product of the DCT coefficient representation of frames. They have however not addressed the problem of detecting extended edit effects in video.

The production model based segmentation approach presented in this work has the additional advantage of using the feature based classification formalism. This formalism separates out the feature extraction and the decision processes. Thus additional features for detecting new effects or for improving the system performance can be accommodated into the system by modifying the discriminant function.

7. Error Measures for Video Segmentation

The video computational model (equation 5) 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

$$V = S_1 \circ S_2 \circ S_3 \circ S_4 \circ \ldots \circ S_{(n-1)} \circ S_n \tag{32}$$

$$V^{'} = O_1 \circ O_2 \circ O_3 \circ \ldots \circ O_k \tag{33}$$

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:

$$E(V, V^{'}) = E_{sb}(V, V^{'}) \times W_{sb} + E_{sc}(V, V^{'}) \times W_{sc} \tag{34}$$

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.

7.1. The Segment Correspondence Problem in Video

Given two video computational models, the process of measuring errors involves comparing the individual segments in the two models. This requires a mapping between the individual segments of the two models. The process of generating this mapping is referred to as the **segment correspondence** problem.

Definition: A correspondence C between videos V and $V^{'}$ can be defined as a mapping $C : V \rightarrow V^{'}$ where $\forall v_i \in V \quad \exists \quad v_i^{'} \in (V^{'} \cup \emptyset)$ and $\forall v_i^{'} \in V^{'} \; \exists \; v_i \in (V \cup \emptyset)$. A correspondence set optimizes some given *correspondence criteria.*

The correspondence criteria used to assign the correspondence between two videos can vary widely depending on the application. If the purpose of assigning the correspondence is to compare the story content of two videos, the correspondence criteria will be some semantic property of the video. For the purpose of computing the segmentation error, a simple criterion like the *temporal segment overlap* can be used. Correspondence criteria can be grouped into *local criteria* where local information from the intervals in question is sufficient to determine the correspondence and *global criteria* where the correspondence between two segments is dependent on the complete temporal structure of the two videos under consideration. The correspondence criteria needed for measuring the video segmentation error is a global correspondence criterion defined as follows:

Maximal Global Overlap Given two videos $V, V^{'}$ the correspondence set $C : V \rightarrow V^{'}$ must maximize the total temporal overlap between the corresponding segments in the two videos.

The following is an algorithm for computing the correspondence between two videos $V, V^{'}$ using the maximal global overlap criterion:

Step 1: Intersection Computation For every segment $S \in V$ and $O \in V^{'}$, compute the intersecting intervals from $V^{'}$ and V respectively. The temporal ordering of intervals in a video can be used to efficiently compute this intersection set.

Step 2: Overlap Computation For every intersection computed in Step 1 the overlap must be computed. Given two intervals $T_1 = [t_{1b}, t_{1e}]$ and $T_2 = [t_{2b}, t_{2e}]$ the overlap O is given by $O(T_1, T_2) = |t_{1b} - t_{2e}| - (|t_{1b} - t_{2b}| + |t_{1e} - t_{2e}|)$

Step 3: Overlap Ordering Order all the intersections based on a descending order of the overlap values.

Step 4: Correspondence Assignment Choose the correspondences (intersections) which yields the maximum total temporal overlap while ensuring that each interval is assigned to only one other corresponding interval.

Step 5: Null Assignment Assign all the remaining segments in either video to a null segment, indicating that it has no corresponding counter part in the other video.

7.2. *Segmentation Error Classes*

Given a correct video model V (equation 32) with n segments and an output model V' (equation 33) with k, let n' and k' be the number of unassigned segments in V and V' after computing the 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. This causes the under segmentation error to be bounded to a much smaller number than the over segmentation error. For example given a 10 segment video with a 1000 frames, a video algorithm that results in an under segmented video can yield outputs with 1 to 9 segments, while an algorithm which over segments video can output 11 to 1000 segments. This effect is accounted for by the use of different scaling factors in the definition of the error measures.

7.3. *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.

$$E_{sb}(V, V') = \frac{\sum_{i=1}^{n} e_i(S_i, O_i)}{Length(V)} + \frac{n' + k'}{\lambda} \tag{35}$$

where $\lambda = n$ for under and equal segmentation and $\lambda = Length(V)$ 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:

$$e_i(T_1, T_2) = |t_{1b} - t_{2b}| + |t_{1e} - t_{2e}| \tag{36}$$

7.4. Segment Classification Errors

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

$$e_{sc} = 1 \;\; if \; l_{s1} \neq l_{s2} \tag{37}$$
$$= 0 \;\; if \; l_{s1} = l_{s2} \tag{38}$$

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

$$E_{sc}(V, V') = \frac{\sum_{i=1}^{n} e_{sc}(S_i, O_i)}{n} + \frac{n' + k'}{\lambda} \tag{39}$$

where $\lambda = n$ for under and equal segmentation, $\lambda = Length(V)$ for over segmentation, e_{sc} is the classification error between S_i and O_i. The video segmentation error between V and V' can now be defined using equation 34.

7.5. Behavior of Error Measure

The goal of defining the error measures is to evaluate the performance of the segmentation system as a function of various parameters of the system. In order to achieve this the error measure must be monotonic. Given a reference video R and a comparison video V, the error measure e should be a monotonic function of the actual error, as V moves away from R, e should increase. The boundary error measure was applied to a set of simulated videos to verify these properties. The reference video contained 32 equal segments. The under segmented videos were generated by deleting segments from the reference videos while keeping the length constant. The over-segmented videos were generated by adding segments to the reference video without changing the locations of existing segments. Figure 8 shows the variation of the **segment boundary error** across the set of simulated videos. The plot shows the error as a function of the number of segments in the video. The error is zero at the location of the reference video and monotonically increases as the comparison video gets increasingly under (over) segmented to the left (right) of the reference video location. This is the error measure used to evaluate the performance of the segmentation system in the next section.

8. Experimental Results

This section reports on the experiments performed based on the techniques proposed in this paper. The goals of the experimental evaluation have been to study the performance of the detectors (sections 8.2, 8.3), to tabulate the performance of the overall segmentation system (section 8.4) and to analyze the sensitivity of the entire system to variations in the threshold (section 8.5). The experimental results presented in here are a snapshot of

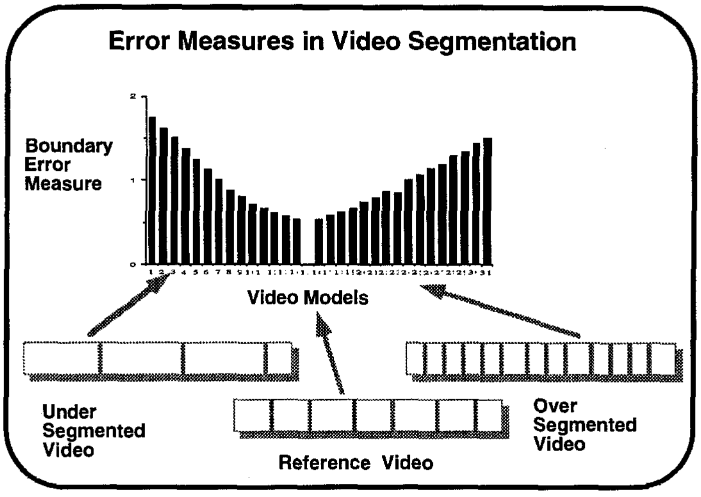


Figure 8. Error Simulation for Video Segmentation

the ongoing work. Experimental characterization on much larger video data collections are currently in progress.

8.1. Experimental Setup and Data Description

The experiments were conducted on video data stored on a laser disk player. The laser disk player was remote controlled from the host computer via a serial link. Video was digitized on the fly, the laser player was synchronized to the desired frame and the image was digitized using a digitizer card on the host machine. The combination of the host computer and laser disk player essentially provided *random frame access* to the video stored on the video disk. The host computer ran the segmentation software which had a graphical user interface front end which allowed easy handling of video and facilitated experimental analysis of the segmentation system.

The data used for experiments was gathered from the local cable television system here in Ann Arbor. The types of programming included in the data include news footage, music videos, commercials, sitcoms, sports casts, etc.

8.2. Experiment 1: Large Scale Feature Plots

This set of experiments presents feature plots of the chromatic and spatial features over a relatively large time spans. The plots show the detector responses to different types of video content like object motions, camera motions and edit effects. The goal of this presentation is to illustrate to the reader the *large scale qualitative behavior* of the detectors. The graphs have been annotated with the various effects that are in progress. The corresponding video sequences have been presented in figure 9

Example 1: Fade Sequence: Figure 9 (left) a fade in sequence (top to bottom). There is a significant amount of object motion in progress close to the camera as the image sequence is being faded in. Figure 10 shows the response of the chromatic scaling feature to this sequence. The detector responds positively during the fade in part of the sequence. The response drops off considerably after the fade in, although the same object motion continues. The detector also responds to a cut. The cut is the last frame in figure 9 (left)

Example 2: Dissolve Sequence: Figure 9 (middle) shows a dissolve sequence. The response of the chromatic scaling feature is shown in figure 11. The first peak in the response corresponds to the dissolve shown in figure 11. It should be noted that the detector misses a dissolve and gives a spurious response. Images corresponding to the missed dissolve and the false positive are not presented. On examining the sequences carefully it was found that the spurious response was due to a large object moving very close to the camera in a low contrast scene, and the missed response was due to a dissolve between very similar sequences in the music video.

Example 3 Translate Edits: Figure 9 (right) a page translate edit between two shots, the first shot is a zoom in shot and the second shot is a pan shot. Figure 12 show the output of the translate edit detector. An observation of the result shows that the detector responds well to the edit while suppressing both pan and zoom,

8.3. Experiment 2: Feature Switching Behavior

The experiments in this section were performed to compare the chromatic and spatial detectors proposed in this work to features that have commonly been used in segmenting video [19], [24], [22]. The features used for comparison are the *template matching* (sum of difference pixels) feature and the χ^2 *histogram* feature shown in table 2. The experiments had the following steps.

Step 1: Representative sequences were chosen for each of the extended edit effects (fade in, fade out, dissolve, spatial translate).

Step 2: The feature responses of the proposed feature and the existing features for these sequences was stored by extracting the features over all the frames that comprised the edit effect. Let these feature responses be denoted by Effect_i, i.e. the response of feature i over the frames of the *edit effect* under consideration.

Step 3: The feature responses of each of the detectors for a cut across the edit effect was measured. This measurement indicates the detector response if the edit effect under consideration had been replaced by a *cut* between the two shots. Let this measurement be referred to as Cut_i, i.e. the response of feature i to a cut between the shots.

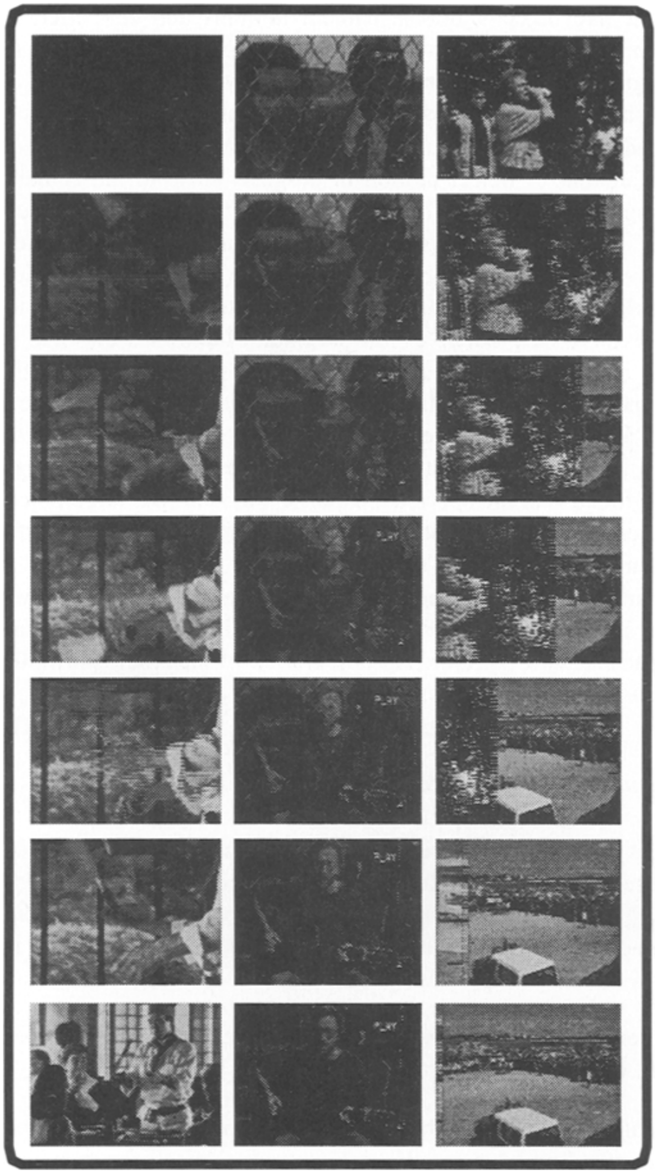


Figure 9. **EXPERIMENT 1: Left:** Fade Out **Middle:** Dissolve **Right:** Translate

Step 4: Based on the measurements of the previous two steps, the normalized feature values were computed. These feature values are shown in figures 13,14,15.

$$\text{Normalized Feature Values} = \frac{\text{Effect}_i}{\text{Cut}_i} \tag{40}$$

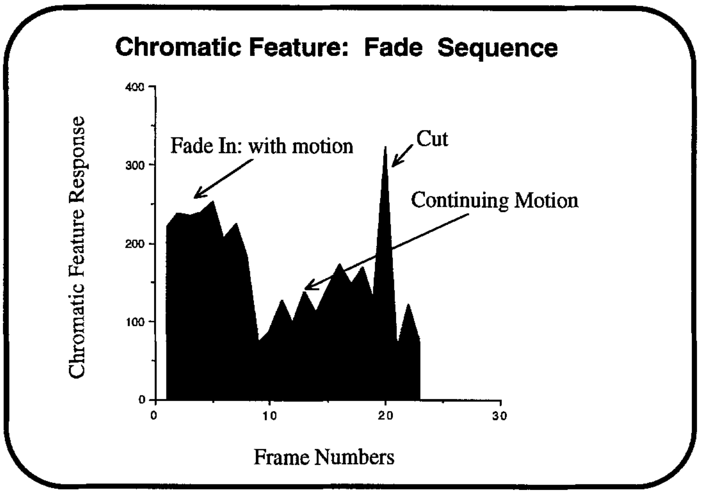


Figure 10. **EXPERIMENT 1:** Chromatic Scaling Feature Response: Fade In Sequence

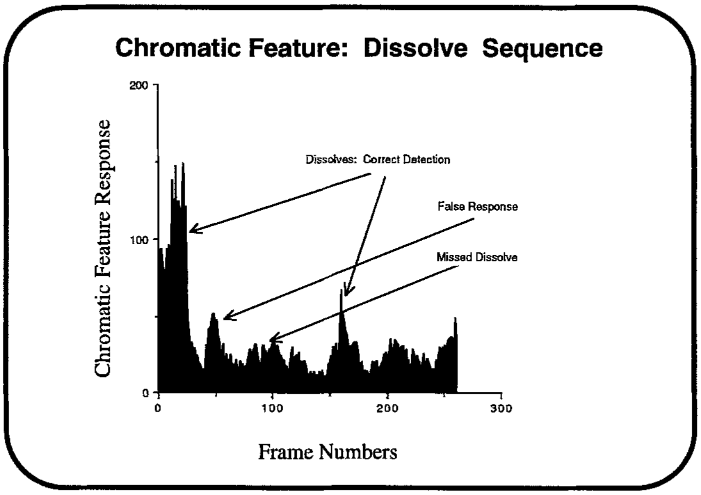


Figure 11. **EXPERIMENT 1:** Chromatic Scaling Feature Response: Dissolve

8.3.1. Interpretation of Feature Comparison

Figure 13: This plot shows the variation of the chromatic feature when it is applied to a fade out followed by a fade in. The ideal response graph shows how an ideal

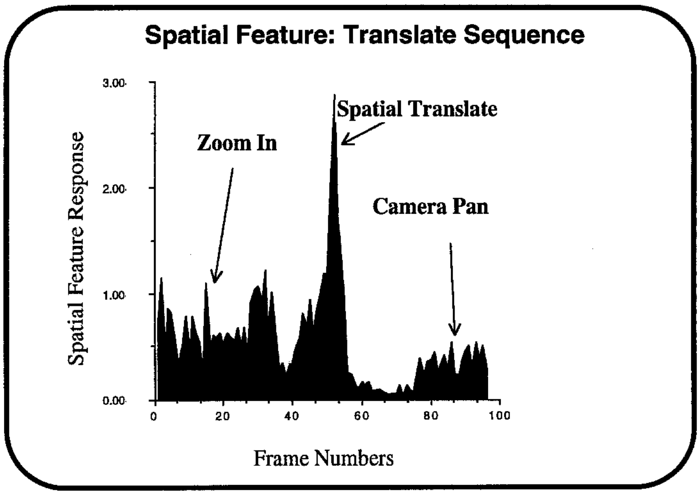


Figure 12. **EXPERIMENT 1:** Spatial Translation Feature Response

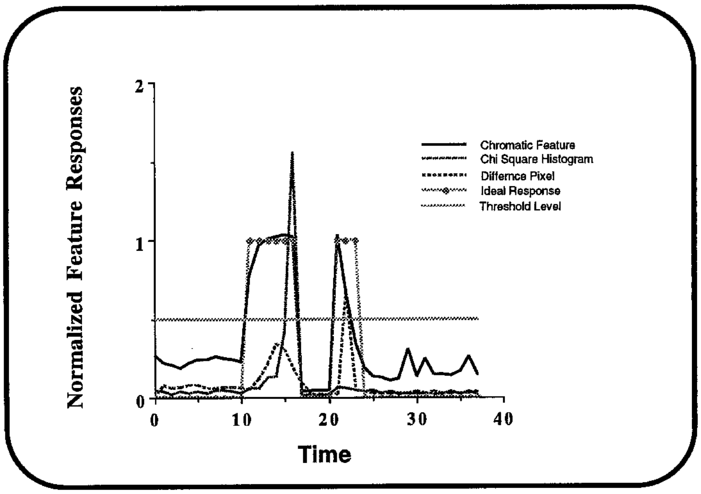


Figure 13. **EXPERIMENT 2:** Chromatic Scaling Feature applied to fades

fade detector would have responded to the sequence. The other three graphs show the

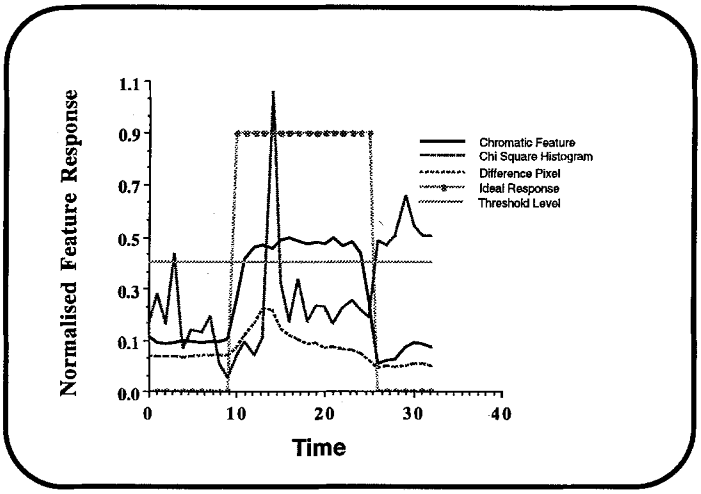


Figure 14. **EXPERIMENT 2:** Chromatic Scaling Feature applied to dissolves

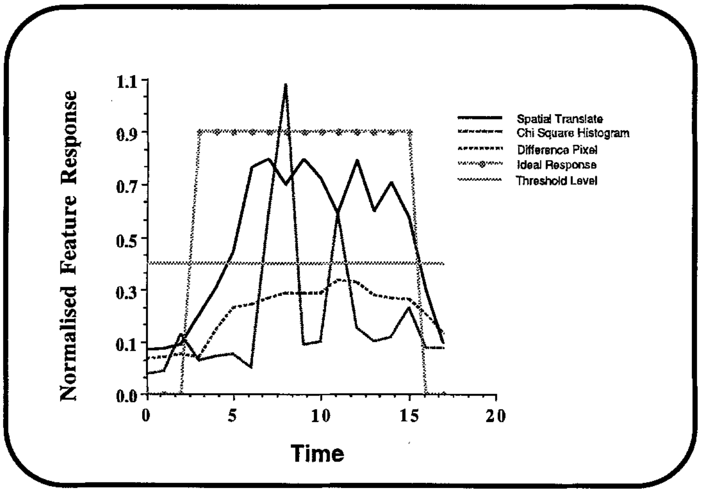


Figure 15. **EXPERIMENT 2:** Spatial Feature applied to spatial translate

response of the chromatic feature, the chi-square feature and the difference pixel feature.

Table 5. **EXPERIMENT 3:** Segmentation Performance

Type	Correctly Detected	False Detection	Total Number	% Correct	% False
Cut	145	16	159	91%	10%
Fade In	3	1	4	75%	25%
Fade Out	3	0	3	100%	0%
Dissolve	14	4	19	73%	21%
Spatial	3	3	5	60%	60%
TOTAL	168	24	190	88 %	12 %

All the graphs are normalized with reference to the cut feature value. Assuming that the threshold is nominally set at 50% of the response to a typical cut, the chi-square feature would pick up the *end of the fade out* and would *miss the fade in*, the difference pixel feature would *miss the fade out* and pick up the *the middle frame of the fade in*. The chromatic edit detector picks up the *complete fade in* and *the entire fade out* except for the last frame.

Figure 14: This plot shows the variation of the chromatic feature when it is applied to dissolve. The ideal feature response is shown. Assuming 50% of the cut feature value as the threshold, it can be seen that the chromatic feature picks up the *entire dissolve* except for frames the beginning and end frames of the dissolve. The χ^2 feature would pick up a cut in between the dissolve and the difference pixel feature would entirely miss the dissolve.

Figure 15: This figure shows the variation of the spatial edit feature applied to a spatial translate edit. When compared to the ideal response and a threshold level of 50% of the cut response, it can be seen that the difference pixels does not pick up any edit, while the χ^2 picks up two frames in the middle portion of the translate edit. The spatial edit feature misses out on frames in the beginning and end part of the edit.

8.4. Experiment 3: Segmentation Performance

Here the segmentation system was used to segment the test video data. The thresholds were picked empirically and the results were tabulated based on the number of edits detected and missed. Table 5 summarizes the results obtained using this algorithm. The result summary indicates an 88% correct segmentation which implies that 12 out of every 100 edits were missed, while about 12 edits were falsely detected.

8.5. Experiment 4: Sensitivity Analysis

The goal of this set of experiments is to measure the stability of the segmentation system with reference to variation in the thresholds. The experiment has the following steps:

Step 1: An experimental video is chosen which incorporates the set of transition effects being considered. The video is manually segmented and a reference model of the video is developed according to equations 3,4,5. The results presented here are for a CNN Headline news video which incorporated fades,dissolves, cuts and translate edits.

Step 2: The segmentation algorithm is run on the test video with a set of thresholds T_c, T_ch, T_sp for the cut, chromatic and spatial features respectively.

Step 3: The segmentation error (equation 34) is measured for the segmentation results with reference to the reference model developed in step 1.

Step 4: The segmentation error is plotted as a function of each of the three feature thresholds that are used in the segmentation process. The thresholds are varied about the initial point at which the segmentation error was measured.

Figure 16 shows the variation of error. The thresholds were varied from -90% to +165% of their initial value. The video was taken from the initial part of the CNN Headline News Cast which includes the logo of the headline news, an anchor person shot, and several indoor and outdoor news reports with cuts, fades, dissolves, page and other edits.

The threshold sensitivity graph shown in figure 16 has a region of *low error* from +25% to +100% of the quiescent threshold value. This region is the **stable operating region** for the system when using either the cut, chromatic or spatial feature. The wide range of this region indicates that the performance of the system is independent of the threshold chosen over a wide range of thresholds. From the graphs it can be seen that the errors for the cut feature vary more widely than for the other two, indicating that the cut feature is more sensitive to the values of the threshold used. This is due to the fact that the cut detector as pointed out earlier is detecting a *null edit* where as the other detectors are designed to identify the occurrence of a certain data model.

8.6. Summary of Experimental Results

The following is an evaluation and interpretation of the results presented in this section along with ideas for improving the results.

Feature Detectors Three different edit classes were described and video features which respond to these edits were designed. The response of the features to the edit classes is very good as is illustrated in the sample measurements presented. The following experimental observations were made about the feature detectors on applying them to different sets of video sequences.

Response to other effects The edit detection features presented are specifically designed to respond to specific models of the edit effect frames like chromatic scaling. However both the chromatic scaling feature and the spatial edit feature respond to other effects. The false positive responses are mainly to sequences

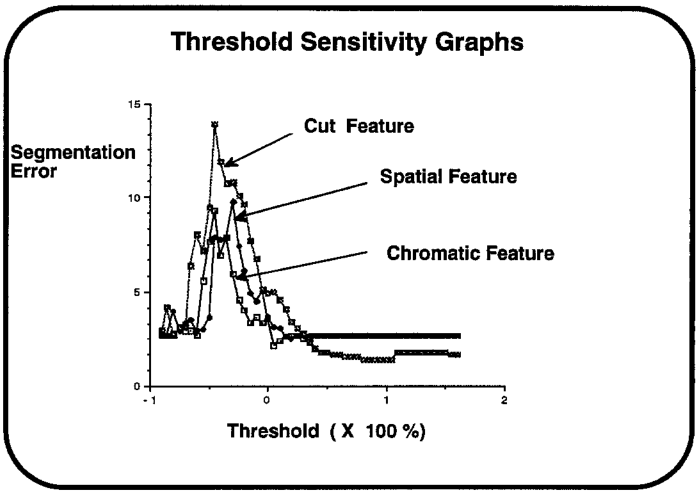


Figure 16. **EXPERIMENT 4:** Threshold Sensitivity Plots

with *large zoom's* and to sequences in which there is *object motion very close to the camera.* In both these cases the image sequence activity tends to be spread out uniformly across most of the image. Since the chromatic and spatial features use *uniformity of change* as the main indicator of edit activity, they respond falsely to zoom's and large object motions. The false responses can be suppressed by altering the uniformity measure to incorporate image motion information.

Partial Fades Application of the chromatic scaling feature to a wide variety of sequences and analysis of the results, shows that the measure is very sensitive to fades. The feature provided good responses to the case where the sequence incorporated partial fades. A typical example of such a sequence is the anchor person shot in a news video, where an icon (map etc) fades into the scene as an inset.

Feature Correlations The spatial and chromatic edit features also display a strong response to cuts. From the perspective of video segmentation this correlation between the three detectors is beneficial as the outputs can be used to improve the confidence in the final segmentation. The reason for the strong response is the fact that many cuts (between sufficiently different scenes) cause a large change in the chromatic and spatial domains of the video.

9. Summary and Future Work

A technique for segmenting digital video is developed. This technique relies on the idea of using explicit models of video production to design feature extractors. The use of explicit models of video makes the technique very effective in dealing with a variety of transition effects. The formalism of production model based classification makes the segmentation system extendable to newer effects. The technique is applied to real video data taken from cable television programming and the results presented.

Future work in the segmentation of video will include the design of *tuned cut detectors*. The edit effect models can be extended to handle a wider range of edits effects. The idea of modeling the video production process and using the models to design feature extractors can be extended to characterizing video shots [8].

Acknowledgments

The first author would like to thank Clint Bidlack, Gopal Pingali and Sandra Bartlett for a number of discussions which have contributed to improving the quality of this paper. The first author would also like to thank Kusuma A Rao for reviewing the final versions of the paper. The authors would also like to acknowledge the reviewers for thier insightful comments

References

1. Farshid 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. David Bordwell and Thompson Kristin. *Film Art: An Introduction*. Addison-Wesley Publishing Company, 1980.
3. Richard O Duda and Peter E Hart. *Pattern Classification and Scene Analysis*. A Wiley-Interscience Publication. John Wiley and Sons, 1973.
4. Claude L Fenema and William B Thompson. Velocity determination in scenes containing several moving objects. *Computer Graphics and Image Processing*, 9:301–315, 1979.
5. Foley, vanDam, Feiner, and Hughes. *Computer Graphics: Principles and Practice*. The Systems Programming Series. Addison Wesley, 1990.
6. Anderson H Gary. *Video Editing and Post Production: A Professional Guide*. Knowledge Industry Publications, 1988.
7. Laurence Goldstein and Jay Kaufman. *Into Film*. E P Dutton and Co, 1976.
8. Arun Hampapur. *Designing Video Data Management Systems*. PhD Thesis, The University of Michigan, 1994.
9. Arun Hampapur, Ramesh Jain, and Terry Weymouth. Digital video indexing in multimedia systems. In *Prooceedings of the Workshop on Indexing and Reuse in Multimedia Systems*. American Association of Artificial Intelligence, August 1994.
10. Arun Hampapur, Ramesh Jain, and Terry Weymouth. Digital video segmentation. In *Prooceedings Second Annual ACM MultiMedia Conference and Exposition*. Association of Computing Machinery, October 1994.
11. Berthold K. P. Horn and Brian G. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

12. R Jain. Difference and accumulative difference pictures in dynamic scene analysis. *Image and Vision Computing*, Vol 2(No 2):99–108, May 1984.

13. R Jain and H H Nagel. On the analysis of accumulative difference pictures from image sequences of real world scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol 1(No 2):206–214, April 1979.

14. Ramesh Jain and Arun Hampapur. Metadata in video databases. In *To appear Sigmod Record: Special Issue On Metadata For Digital Media*. ACM: SIGMOD, December 1994.

15. Ramesh Jain, Rangachar Kasturi, and Brian G Schunck. *(To be published) Introduction to Machine Vision*. McGraw Hill, 1995.

16. Zvi Kohavi. *Switching and Finite Automata Theory*. Computer Science Series. McGraw Hill, 1978.

17. Ira Konigsberg. *The Complete Film Dictionary*. Penguin Books, 1989.

18. J O Limb and J A Murphy. Estimating the velocity of moving images in television signals. *Computer Graphics and Image Processing*, 4:311–327, 1975.

19. 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.

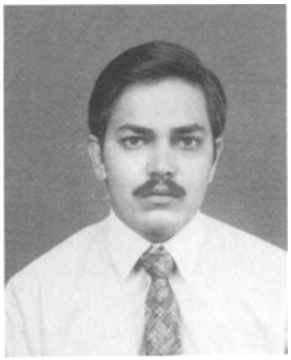
20. Azriel Rosenfeld and Avinash C Kak. *Digital Picture Processing Vol 1,2*. Academic Press, 1976.

21. 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.

22. Yoshinobu Tonomura, Akihito Akutsu, Yukinobu Taniguchi, and Gen Suzuki. Structured video computing. *IEEE Multimedia*, 1(3):34–43, Fall 1994.

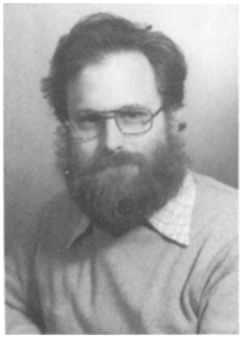
23. George Wolberg. *Digital Image Warping*. IEEE Computer Society Press, 1992.

24. H J Zhang, A Kankanhalli, and S W Smoliar. Automatic partitioning of video. *Multimedia Systems*, 1(1):10–28, 1993.



Arun Hampapur recieved his B.Eng. in Electronics and Communication Eng from the University of Mysore, India in 1987. He received his Masters degree in Electronics and Control Eng from the Birla Institute of Technology and Science, Pilani, India in 1989. He received his M.S. in Computer Eng from Louisisana State University in 1991. He is currently a Doctoral Candidate in Computer Science and Eng at the University of Michigan, Ann Arbor. His dissertation work is on the Design of Video Data Managment Systems. His research interests include video databases, digital video processing, image databases, multimedia systems, computer vision, image processing and robot navigation systems. He is a student member of IEEE.

Ramesh Jain is currently a Professor of Electrical and Computer Engineering, and Computer Science and Engineering at University of California at San Diego. Before joining UCSD, he was a Professor Electrical Engineering and Computer Science at the University of Michigan, Ann Arbor. He has also been affiliated with Stanford Unviersity, IBM Almaden Research Labs, General Motors Reserach Labs, Wayne State University, University of Texas at Austin, University of Hamburg, West Germany, and Indian Institute of Technology, Kharagpur, India. His current research interest are in multimedia information systems, image databases, machine vision, and intelligent systems. He is the Chairman of Imageware Inc., an Ann Arbor based company dedicated to revolutionzie software interfaces for emerging sensor technologies. Ramesh is a Fellow of IEEE, AAAI, and Society of Photo-Optical Insturmentaion Engineers, and member of ACM, Pattern Recognition Society, Cognitive Science Society, Optical Society of America, and Society of Manufacturing Engineers. He has been involved in organization of several professional conferences and workshops, and served on editiorial boards of many journals. Currently, he is the Editor-in-Cheif of IEEE Multimedia, and is on the editorial boards of Machine Vision and Applications, Patter Recognition, and Image and Vision Computing. He received his Ph.D. from IIT, Kharagpur in 1975 and his B.E. from Nagpur University in 1969



Terry E. Weymouth received his PhD in May of 1986 from the University of Massachusetts, where he was involved in computer vision research. Dr. Weymouth is a Senior Research Scientist in the Electrical Engineering and Computer Science department of the University of Michigan. He is a member of IEEE, ACM and AAAI. His reserach interests include video and image databases, computer vision, robot navigation, biomedical image processing and distributed collaboratory systems.