Segmentation, Index and Summarization of Digital Video Content

Di Zhong

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
in the Graduate School of Arts and Science

Columbia University

2001

ABSTRACT

Segmentation, Index and Summarization of Digital Video Content

Di Zhong

In this thesis, we propose and develop unique frameworks and methods for temporal and spatial video segmentation as well as object based video representation, indexing and retrieval at both the syntactic and the semantic level.

First we demonstrate a robust and real-time temporal scene cut detection system that combines color, edge and motion features in both compressed and uncompressed domains. Contrary to existing work, we considered comprehensive issues in practical situations, such as gradual transition, lighting change and motion. The algorithms perform very well for variant kinds of videos, including sports, sitcom, news, cartoon, movie and home videos.

Then we present an automatic region segmentation system for content-based video search. The system segments and tracks consistent regions through each video shot, and then computes visual features of extracted regions to build visual libraries that support region level search. A web-based video query system that has more than 3,000 video shots has been built. The query system allows users to do spatial-temporal search of video shots by drawing regions and specifying features. It is the first video search engine that supports automatic extraction and object-level motion-based search.

Semantic object segmentation and tracking is then studied to produce high-level object

representation and description. We introduce an integrated scheme for semantic object segmentation and content-based object search. AMOS, a unique video object segmentation system that combines low-level automatic region segmentation with user inputs is developed. An object query model is developed to effectively combine local region-level features and spatial-temporal structures. This system is very useful for MPEG-4 and MPEG-7 applications.

At the end we present a real-time framework to build semantic-level structure and event index of live sports videos. It utilized the segmentation and searching methods we have developed to detect specific scenes and events. Also it integrates knowledge about domain-specific video structures and generic machine learning algorithms. We show applications of such techniques in high-level video retrieval and browsing systems in specific domains such as sports videos. In addition, we demonstrate a summarization scheme that provides users intuitive structures of video content and statistics of game events and views.

Contents

List of Figures

List of Tables

ACKNOWLEDGEMENTS

I would like to offer my profound gratitude towards my advisor, Prof. Shih-Fu Chang, for his invaluable guidance and inspiration throughout the course of my research work. Without his continuous encouragement and advice, I could never complete this thesis.

I would like to thank Professor Dimitris Anastassiou and Professor Alexandros Eleftheriadis for their advice in the proposal of this work.

I thank all my colleagues in the ADVENT Lab, especially John Smith, Horace Meng, Luois Wang, JaeBoem Lee, Bill Chen, Hari Sundaram and Raj Kumar, with whom I have worked together for several years, for their precious suggestions, discussions and friendship.

I express my sincere thanks to my wife and my parents for their love, understanding and supports.

Chapter 1

Introduction and Motivation

1.1 Digital Video Indexing and Retrieval

Today with the progress in video compression and communication, we are able to put a large amount of digital videos online. More and more media content providers are delivering live or on-demand videos over the Internet. Home users are having high bandwidth cable or DSL connections to view TV-quality videos. While the amount of video data is rapidly increasing, multimedia applications are still very limited in content management capabilities. There is a growing demand for new techniques that can enable efficient processing, modeling and management of video contents.

A typical indexing and retrieval scenario of video content is shown in **Figure 1.1**. First, input videos and images are segmented into temporal and/or spatial consistent units. Visual features are then extracted from these segments to build indices and summaries. And finally videos or images are browsed and retrieved based on these features and structures.

Within this scenario, the content-based technique is one key component that provides feature based similarity search of pictorial data. Recently, many studies have been conducted in this area. Some examples include QBIC, PhotoBook, VisualSEEK, MARS

and VideoQ [79,85,99,81,21]. The objectives are to provide enhanced visual search capability, and to automate the conventional annotation process of videos and images.

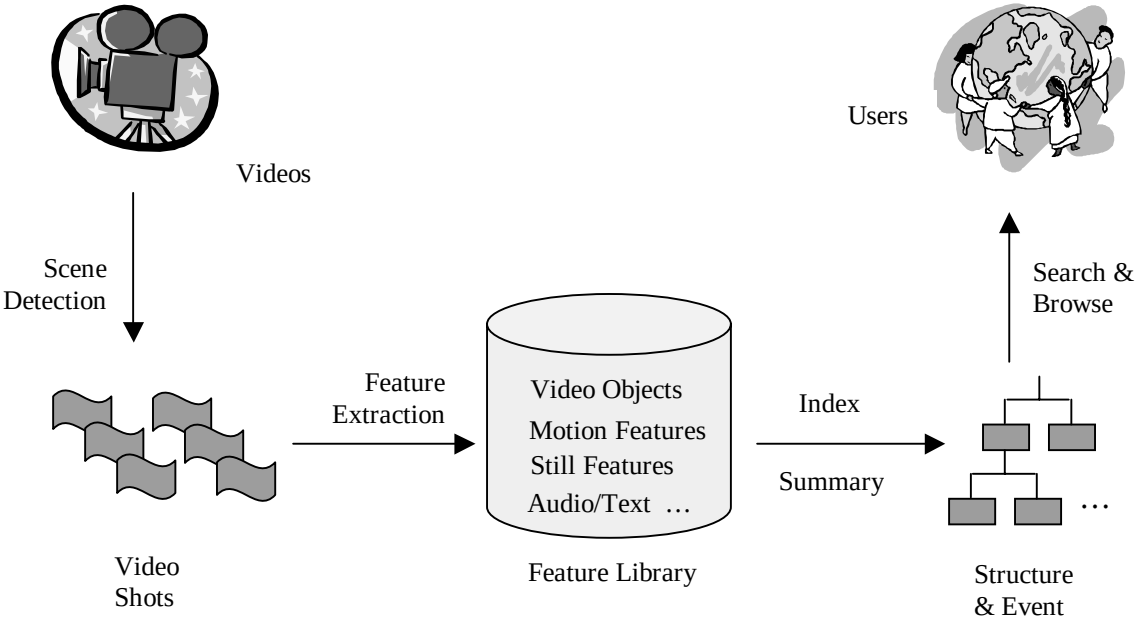


Figure 1.1 Content based video indexing and retrieval process

In addition to the feature index, another type of work aims at parsing and discovering syntactic structure [129,116,104] and semantic events [83,59] in video programs using domain knowledge and models. Results of such analysis can be used to generate semantic level or structural summaries. These enable users to have access to arbitrary segment at the detailed level based on intuitive and informative structures.

This thesis primarily presents our research work on automatic or semiautomatic extraction and indexing of visual features and semantic content in video data. Particularly, we have developed new tools and methods in handling the following unique issues.

- Robust and real time scene cut detection
- Automatic region tracking and indexing for object based video search
- Active method for semantic object segmentation and search

- Robust automatic moving object detection
- Structure parsing and event detection for structured videos

1.2 Overview of Existing Work

Images and videos are visual representations of information. During recent years, methods have been developed for retrieval of images and videos based on their visual features. Color, texture, shape, motion and spatial-temporal composition are the most common visual features used in visual similarity match [79]. Two typical query modalities include query by example and query by sketch [21]. A number of studies have been conducted on still image retrieval. Progresses have been made in areas such as feature extraction [20], similarity measurement, vector indexing [23,91] and semantic learning [74]. Studies on content-based video retrieval have been limited to scene cut detection, key frame extraction, grouping and browsing [69,122]. While image retrieval techniques can be applied to video searching, unique features of video data demand solutions to many new challenging issues.

Compared with still images, videos are dynamic data with the temporal dimensions. Videos are presented continuously at certain rates. A TV-quality video has 25 to 30 frames per second. Besides, videos consume a huge amount of storage and bandwidth. The size of a typical one-hour MPEG-1 video is more than 500M bytes.

The continuous characteristic and large data amount makes it further challenging to process and manage videos. On the other hand, as more information, particularly temporal and motion , is contained in videos, we have a better opportunity to analyze visual content inside video. Furthermore, although videos are continuous media, the content contained within a video program is hierarchical in nature. A movie video can be

divided into stories, shots, frames, as well as objects and actions. Efficient video retrieval systems require complete indexing to all these units. Extraction of constituent objects and discovery of underlying structures has been a vital but unsolved research topic. A brief overview of existing approaches are given below.

As a fundamental step of video indexing, scene cut detection algorithms have been widely studied to divide video streams into elemental units (i.e. shots). Low-level features such as color, edge and motion have been proved to be appropriate for the detection of temporal changes such as camera breaks and transitions [70,123]. Based on temporal segmentation, video data can be efficiently represented in an abstracted or summarized way. Many technologies have been developed to index segmented video shots.

One common approach that has been used in many systems is to first select one or more key frames (i.e. representative frames) for each video shot, and then use image features such as color, shape and texture to index these key frames. How to choose and organize key frames are the major issues here. Besides simple sampling methods, advanced algorithms have been developed to use color variances, camera motions, embedded texts and human faces [111] to select frames that convey the most significant information of a video shot.

Using only key frames for indexing ignores motion information contained in video shots. Moreover, as the videos are broken into individual shots, temporal relationships and events among successive shots are not explored. To enable search for events and actions, a number of methods have been proposed to include motion and temporal information into video content models. In [14,23], symbolic descriptions are used to

represent temporal relationships (e.g., before, after, etc.) and to enable match and query of such temporal structures. Motion estimation, spatial-temporal logics, object segmentation and tracking are some key techniques that have been applied in such modeling processes.

Visual features contain little semantic information, and in many circumstances, are not convenient or sufficient for users to find desired videos. High-level abstractions and summarizations, such as story, scene or action, allow users to search and browse videos at a more effective and intuitive level. As shown in **Figure 1.2**, a news story from CNN is broken down into a hierarchy of segments, stories and then individual shots [129]. This hierarchical structure provides a multiple layer abstraction that can be used to help users navigate through the long video program. In addition to detecting temporal structure, efforts have also been made to extract semantic segments from video shots.

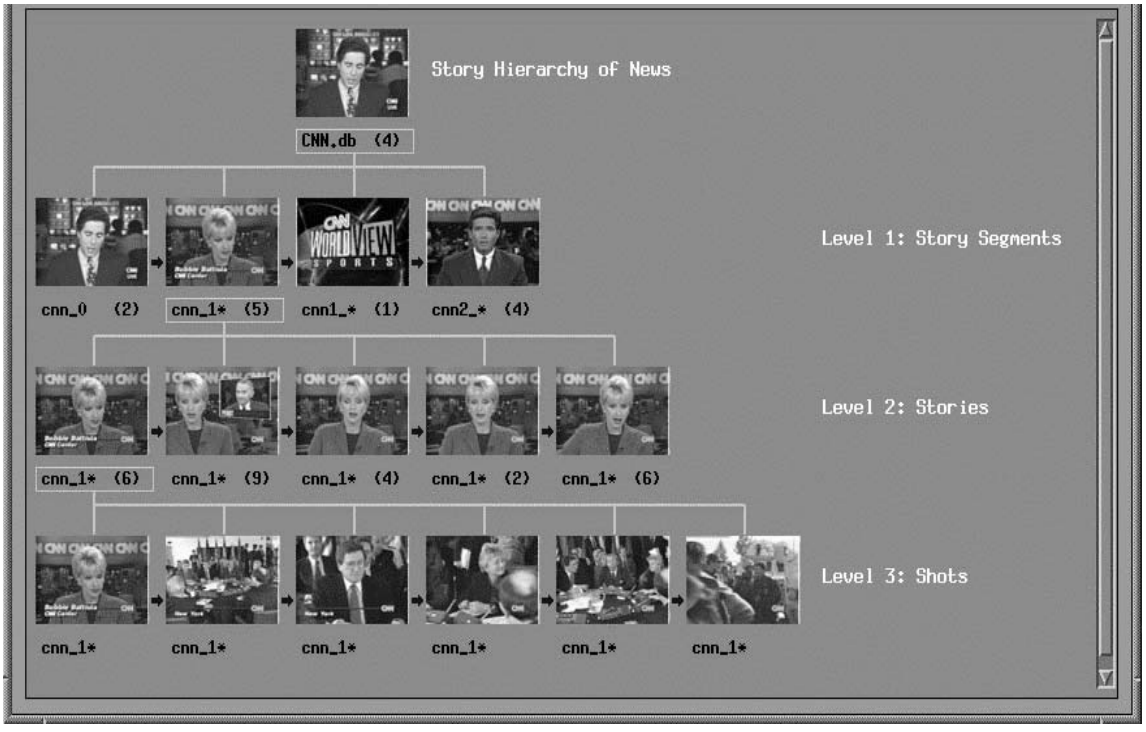


Figure 1.2 Hierarchical abstraction of a news program

In [122], a spatial structural model is used to detect anchorperson scenes. A long news program is then broken into stories based on anchorperson scenes. In [117], the scene transition graph is used to capture both the content and temporal flow of videos. It is reported to be able to detect dialogues, actions and story units.

In general, unlike elementary video shots that can be defined according to low-level features, high-level entities like story or scene are difficult to automatically extract based on only low-level visual features. As observed in [117], to properly group or classify video shots, more complicated domain models need to be built based on intermediate or high-level representations, such as regions or objects. In recent studies, several emerging video representation frameworks such as MPEG-4 and MPEG-7 have also adopted similar object-oriented models [66,67].

In conclusion, while progress has been made in the area of video indexing and summarization, many challenging issues remain to be solved. Thus, more advanced video analyzing techniques are demanded to build effective and efficient video search systems.

1.3 Problems Addressed

As we have discussed previously, video indexing involves processes of segmentation, analysis and abstraction of video content. Temporal segmentation breaks long video streams into manageable units like shots (instead of individual frames). Spatial segmentation extracts video objects contained in each video shot. These segments enable us to analyze and model video content, and to build comprehensive multi-level structures that users can efficiently search and browse, as shown below in **Figure 1.3**.

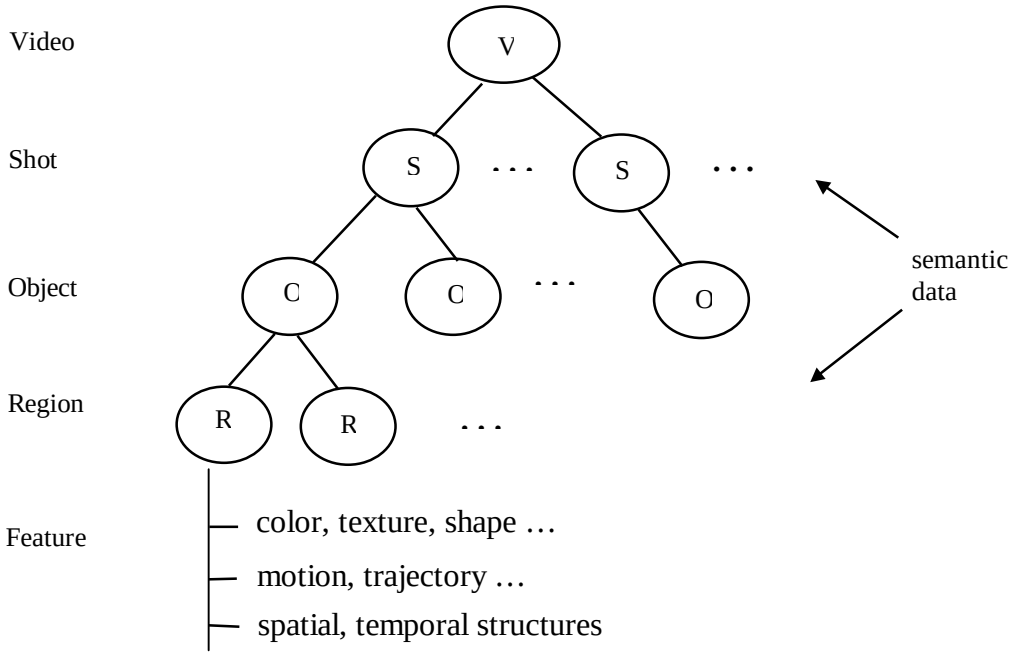


Figure 1.3 A unified object-based video representation synergying with MPEG-4 and MPEG-7 standards

Challenging issues exist in segmenting objects and building this object based representations. In the following we will review major topics that will be addressed in this thesis. They include temporal scene segmentation, video object segmentation, feature indexing and semantic summarization.

1.3.1 Temporal Scene Segmentation

Although many progresses have been made in scene cut detection, existing systems still lack the following capabilities: 1) detect gradual transitions reliably; 2) achieve real-time (or faster than real-time) processing performance; and 3) to handle special situations such as flashes or sudden lightening variances.

In building a real-time video parsing and summarization system, we developed a scheme that combines color, edge and motion features in both compressed and un-compressed domains using machine-learning techniques. In this scheme, we adopted a

new model to detect gradual transitions by examining the characteristics of their beginning and ending phases. We also successfully handled the issues of flashlights and aperture changes. Our experiment results demonstrate real-time and accurate scene cut detection performance.

1.3.2 Video Object Segmentation

Video objects are the most obvious information in video data, and are critical for human visual perception. It is intuitive for users to describe or annotate a scene or event based on objects. Recent researches on MPEG-4 have addressed the issue of object-based video compression. It is envisioned to allow the access and manipulation of audio-visual objects directly in the encoded video sequences. This provides a great potential for interactive multimedia applications, such as object based video authoring, search and edit. The emerging MPEG-7 standard, a multimedia content description interface, also defines a description scheme that attaches content descriptors to video scenes and objects.

Both the MPEG-4 and MPEG-7 standards do not standardize the methods and tools to extract objects or features [66]. As current videos are represented in the raw pixel format, objects and the information they convey have to be extracted before the construction of any object based content models. While both the MPEG-4 and MPEG-7 are becoming international standards, tools for segmenting video objects and features are still missing.

Recently, generic and flexible segmentation algorithms have been studied in many works. It has been shown that the human visual system does not compute boundaries based on any particular set of attributes [36]. One problem with the current systems is that typically only a fixed subset of features are utilized. Apart from some commonly

used features like color, edge and motion, spatial and temporal structures as well as certain underlying rules that human beings use to identify objects are also very important. Another problem is that without a good object model it is difficult to incorporate various features into object segmentation process. Different features may give different interpolations of objects. To meet these requirements, we developed a hierarchical model for object representation and a segmentation and tracking scheme that effectively fuses various visual features.

This scheme includes two related parts. We first developed an algorithm for automatic segmentation and tracking of regions based on fusion of color, edge and motion. Combined with a dominant region selection and global motion compensation process, we show that the automatically extracted video regions can be used to successfully retrieve video objects. Based on the first part, we then developed an active system for semantic video object segmentation. A semantic object typically corresponds to real-world whole body objects. Definitions of semantic objects are given by users or based on some domain knowledge. It is modeled as a multi-level hierarchical structure and includes a set of regions with associated spatial and visual features. An innovative grouping and alignment method is developed to obtain semantic objects from automatically tracked regions. Extraction of video objects at both region and semantic levels gives us rich features at different levels and great flexibility for describing video content in different applications.

1.3.3 Indexing and Summarization

The shot and object based video representation provides a flexible foundation for video

indexing and summarization. Existing video indexing techniques are mainly based on visual features extracted from key frames. Video objects allow us to describe the content more effectively. We can easily associate motions and activities with an object. By identifying relationships between objects in different shots, we can also define temporal events across the video shots. Detection of significant scenes, objects and their characteristics enable us to summarize long videos into high-level concise abstractions.

In this thesis, we studied efficient video indexing and search methods that utilize a rich set of visual features at both video object and region levels. A region based video object query model is proposed to integrate global and localized features, as well as the measurement of various spatial-temporal structures.

How to combine domain knowledge and visual features using machine learning techniques to automatically extract syntactic and semantic structures is another important issue addressed in this thesis. We demonstrate a real-time parsing system that automatically identifies important scenes and game structures based on domain models and the analysis of video segmentation results.

1.4 Summary of Contributions

The main contribution of this thesis in the field of content based video indexing and summarization is summarized as follows.

1. A robust, real-time scene cut detection scheme which combines color, edge and motion features in compressed and un-compressed domains using machine learning techniques
2. New algorithms to detect gradual transitions, flashlights and aperture changes

3. Automatic video region segmentation and tracking based on the fusion of visual features
4. Effective region aggregation and boundary alignment to track generic video objects based on the tracking of underlying regions
5. An abundant set of visual features at both object and region levels, including color, shape, texture, motion trajectory and temporal information
6. Effective integration of global and localized features, and the measurement of various spatial-temporal structures (directional, topological, temporal) in a region based video object query model
7. Automatic moving object detection using global region tracking information
8. Real-time system to extract important scenes and content structure based on the analysis of video segmentation results

1.5 Outline of Thesis

In the rest of the thesis, we first study general video segmentation techniques (including both temporal and spatial), as well as corresponding feature matching and retrieval schema. We will then discuss how to utilize domain models to discover video structures and summarize video content. The following chapters are organized as follows.

In Chapter 2, we present the shot detection algorithms. Existing work and open issues are first reviewed. We then present a new scheme that combines color, edge and motion features for real-time scene change detection. Machine learning techniques, especially decision trees, are used to help us build decision rules. We also present our algorithms for detecting gradual scene changes and some special situations such as flashlights and

aperture changes. Finally, experimental results are shown and discussed.

In Chapter 3, we present an object oriented video retrieval system using automatically tracked video regions. We first introduce an automatic video region segmentation and tracking system that fuses multiple visual features to achieve accurate and reliable tracking results. Experimental results show that our system can track salient regions properly over long video sequences. Then we discuss the construction of the visual feature library, and present efficient techniques for spatial-temporal query based on the automatically segmented video regions. Finally, some query experiments are discussed.

In Chapter 4, we present an integrated schema for semantic object segmentation and similarity-based object search based on a multi-level video object model. Here, we use the term "semantic object" to refer to whole-body objects in the image corresponding to real-world physical objects, such as cars, people and houses. We first discuss AMOS, a generic video object segmentation system which combines automatic region segmentation with user input for defining and tracking semantic video objects. The system expands the region segmentation techniques described in Chapter 3, and utilizes an iterative region aggregation and boundary alignment process to generate and track accurate semantic object boundaries. Also we discuss an automatic moving object detection approach using global region tracking information. Combination of the region and object-level representations and a rich set of features (color, texture, shape, motion, and spatio-temporal relationships) constitute a very powerful multi-level object model. Our experiments have shown great results and promise in developing advanced video search tools for semantic video representations such as those defined in MPEG-4 and MPEG-7.

In Chapter 5, we present a structural parsing and event detection system for domain specific videos (e.g., sports). The system is built using a real-time streaming framework. It utilizes the segmentation and searching methods we have developed in detecting unique scenes and events. We show that our video segmentation and indexing techniques can be easily integrated into high-level video retrieval and browsing systems in specific domains such as sports videos. We demonstrate a summarization interface that provides users overall structures of video content by showing the statistics of video shots and important views.

Chapter 6 concludes the work presented in this thesis, and includes discussion of potential applications and future work.

Chapter 2

Scene Change Detection Combining Multiple Visual Features

2.1 Overview

Shot based indexing techniques have been widely used to organize video data. Scene change detection is the most commonly used method to segment image sequences into coherent units for video indexing. A shot is a sequence of contiguous frames that are recorded from a camera. There is usually one continuous action within a shot, with no major change of scene content. However, there are still many different changes in a video (e.g. object motion, lighting change and camera motion), it is a nontrivial task to accurately detect scene changes. Furthermore, the cinematic techniques used between scenes, such as dissolves, fades and wipes, produce gradual scene changes that are harder to detect.

Scene cut detection algorithms have been studied since the early 90's. The basic method is to measure the pixel difference frame-to-frame in terms of intensity or color [126]. In [126], the number of changed pixels is counted and if the number exceeds a certain percentage, a scene cut is detected. This method is not robust due to the camera and object motions that can cause large pixel value differences.

Color histograms have been used to overcome the problem, as color distributions in

successive frames are not significantly affected by camera or object motions [121]. Assume H_i is an N-bin color histogram extracted from frame *i*, the frame difference is defined as :

$$D_i = \sum_{j=1}^{n} \left| H_i(j) - H_{i+1}(j) \right| \tag{2.1}$$

If D_i is larger than a given threshold, a scene cut is detected at the frame *i+1*. A more efficient distance measure, χ^2-test, is proposed in [78], and shown to have better performance in experiments compared to other measures. In the χ^2-test, the distance between two color histograms H_i and H_{i+1} is defined as:

$$\chi_i^2 = \sum_{j=1}^{n} \begin{cases} \dfrac{(H_i(j) - H_{i+1}(j))^2}{\max(H_i(j), H_{i+1}(j))} & \text{if } H_i(j) \neq 0 \text{ or } H_{i+1}(j) \neq 0 \\ 0 & \text{otherwise} \end{cases} \tag{2.2}$$

Although direct comparison of frame-to-frame color difference is good for direct scene changes, gradual transitions such as fade-in, fade-out, dissolve and wipe cannot be accurately detected in the same way.

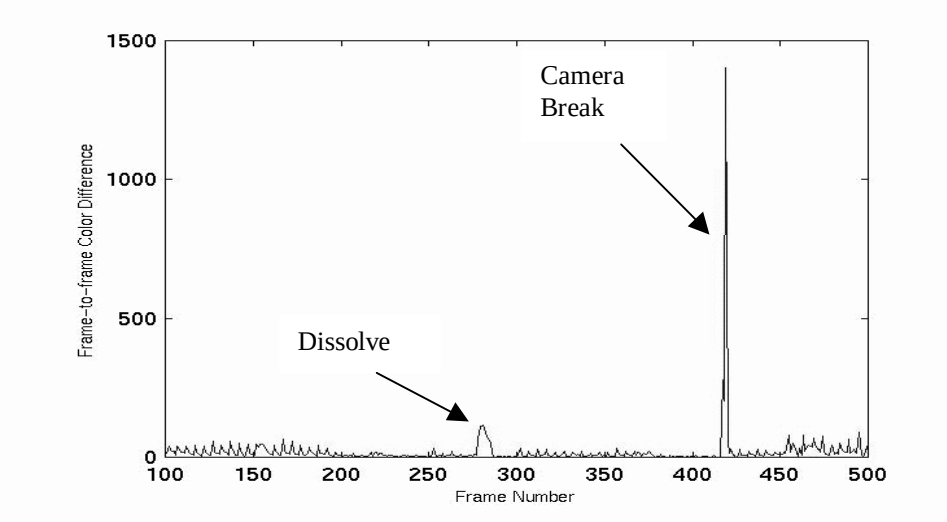


Figure 2.1 Comparison of direct scene cut and gradual

As shown in **Figure 2.1**, the frame-to-frame color differences within a gradual scene change are much smaller than that of a direct scene cut. In the meanwhile, gradual transitions last much longer (more than 1 second) compared to direct scene changes.

Because of the low difference values of gradual scene changes, a single threshold cannot distinguish them from camera or object motions. To resolve this problem, a twin-comparison algorithm is developed in [121]. This method requires two cutoff thresholds, one higher threshold for direct changes and a lower one for gradual transitions. The higher threshold is applied first. If there is no direct scene cut, the lower threshold is then used to detect potential transitions. Once a candidate transition is detected, frame-to-frame differences are accumulated for successive frames. If the accumulated distance is larger than the higher threshold, a gradual transition is declared. Note that this is based on the assumption that transitions last over a certain period, and frame-to-frame differences within the transition period do not drop below the lower threshold. In [119], an edge-based approach is proposed to detect direct scene cuts and gradual transitions at the same time. It computes the percentages of edges that enter and exit between two frames. Shot boundaries are detected when the percentage is over a given value. Dissolves and fades are detected by comparing the enter and exit percentages.

Scene cut detection often includes feature extraction and comparison for suucessive pairs of successive frames. It is a time consuming process that cannot be done in real time on a regular PC or workstation. As most digital videos are compressed in MPEG, detecting scene changes directly in the compressed domain has been studied to accomplish real time performance. In [69], statistics of motion vectors are used to detect scene cut. For a P-frame, the ratio of the number of intra-coded blocks and the number of

inter-coded blocks is computed. For a B-frame, the ratio of the number of backward motion vectors and the number of forward vector is computed. High ratio values indicate shot changes at P- or B-frames. On the other hand, a low ratio on a B frame indicates that there is a scene cut at its preceding I-frame (in transmission order).

In this chapter, we present a new scene cut detection scheme which combines multiple visual features in both the compressed and uncompressed domains. The main contribution of our work includes:

- An effective scheme to combine motion and color features in both the compressed and uncompressed domains.
- New algorithms to detect gradual transitions, as well as flash lights, lighting changes and camera motions.
- A multi-level scene change detection for browsing and correction.
- Demonstrated real time performance with high accuracy
- Extensive experiments on large amount of various types of videos

2.2 A New Scheme Combining Multiple Visual Features

Although abrupt shot boundaries have been well studied in existing works, robust detection of gradual transitions is still a challenging issue. In [43], a comparison of many existing scene cut detection algorithms is conducted. Around 90 percent accuracy is reported for direct cut detection. For gradual scene changes, the accuracy is in the range of 70 to 80 percent. While building a real-time video parsing and analysis framework that can be applied to live videos, we also met some other challenging issues. First, the real-time requirement usually conflicts with high detection accuracy. More complicated visual

features, which are used to obtain accurate scene cuts, can hardly be computed in real time. Furthermore, lighting changes, e.g., flashlights that occur often in home videos, raise another problem that causes false detection results.

To address these issues and further improve scene detection accuracy, we developed a multi-statge scene cut detection scheme that combines motion, color and edge information. Although various visual features and comparison methods have been studied, it is agreed that no single feature works better in all situations [43]. How to effectively combine visual features to obtain robust scene cuts is an open issue. We use machine-learning techniques (i.e. decision tree) to find combined measure metrics and thresholds. Compressed-domain features are applied before those in the un-compressed domain in order to accomplish real-time performance. We also developed special modules to detect flashlights and lighting changes.

The diagram of the schema is shown in **Figure 2.2**. An overview of the scheme is given below. Detailed algorithms will be described in the following sections.

Compress domain features are first extracted. Motion statistics are computed from different types of motion vectors in P or B frames. Color differences are extracted from the DC images of I- or P- frames. It only requires partial decoding without inverse DCT to extract these features, and thus the process is faster than real-time even on a regular Pentium II 300 PC. The next module detects possible flashlights. It is based on the comparison of the frame-to-frame color difference and the long term color difference. If a flashlight is detected at a frame, no scene cut detection will be performed on this frame. Otherwise, scene cut detection algorithms are applied.

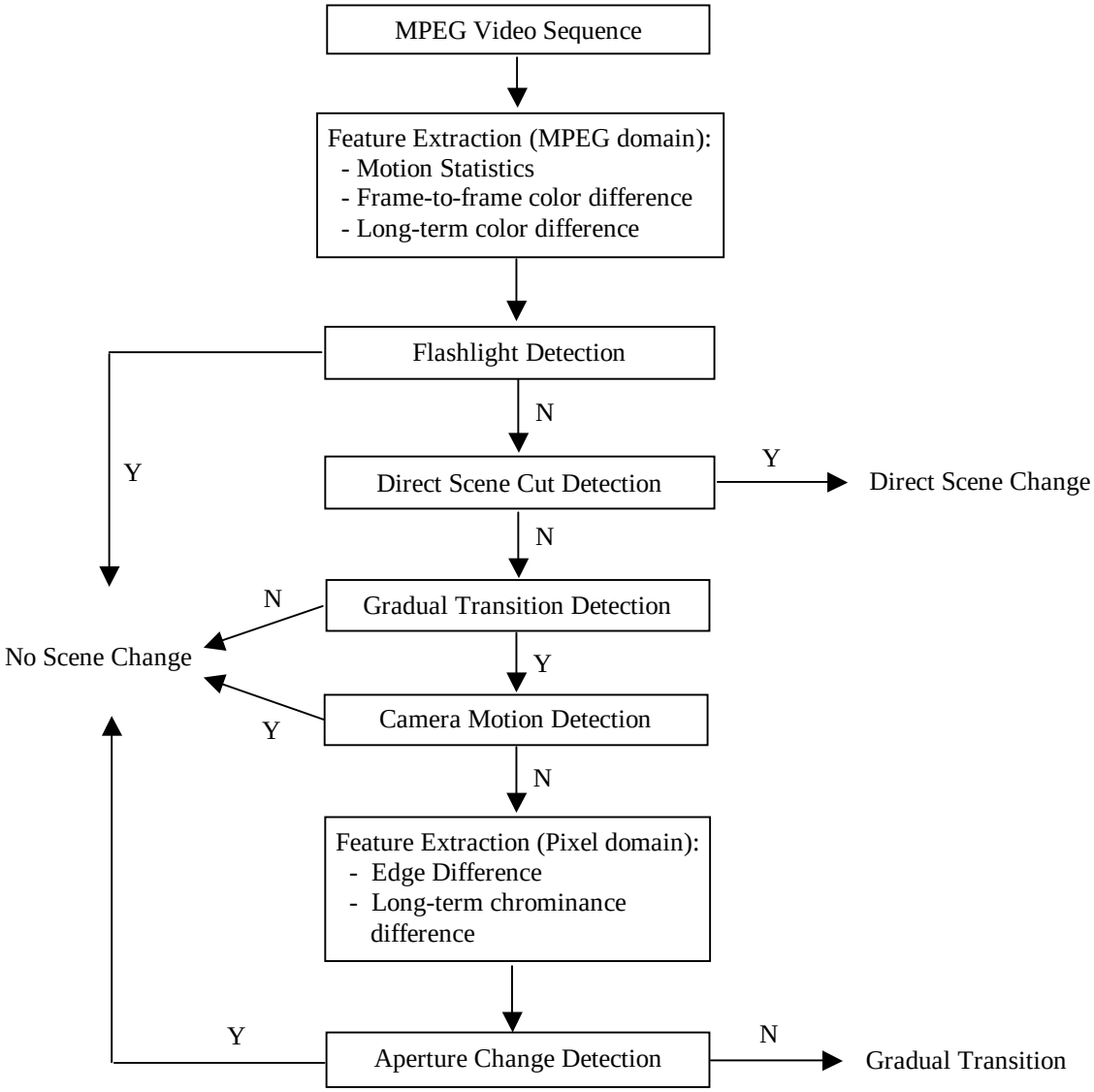


Figure 2.2 The scene cut detection schema that combines multiple visual features

Because direct scene changes can be detected accurately, we first check if there is a direct cut at the frame. If no cut is detected, we then check if there is a gradual transition by detecting starting and ending edges of transitions. As gradual scene transitions are opted to be confused with camera motions or aperture changes, when a potential gradual transition is found, we first check camera motions based on motion vectors in the MPEG compressed domain. If there is no camera motion, we extract edge and chrominance features from de-compressed frames, and compute the differences to examine whether the

difference is caused by aperture changes. In case of aperture changes, scene structure and chrominance are not affected as much as brightness. A gradual transition is declared only if image edges or chrominance are significantly changed.

In the following sections, we will give a detailed explaination of the above modules and algorithms in the order that they are applied.

2.3 Feature Extraction in the MPEG Compressed Domain

As digital video sequences are usually compressed for storage and transmission, it has been proposed in some works to detect scene cuts directly in the compressed domain without fully decoding the bitstream. Here we focus on the most popular MPEG-1 or MPEG-2 compressed videos. We adopt the motion statistic features proposed in [69], and define two more color statistic features that are also extracted in the DCT-based compressed domain. As there are three different types of compressed frames [65], different feature extraction methods are used.

MPEG video sequences are composed of three types of frames, i.e., I, P and B. An I-frame is completely intra-coded without motion prediction. A P-frame is inter-coded based on motion prediction errors from its past I- or P- frame. A B-frame is coded based on bi-directional motion prediction from its past and later I- or P- frames. I- and P-frames are also referred as anchor frames.

For an I-type frame, the frame-to-frame and long-term color differences are computed. The color difference between two frames *i* and *j* is computed in the YUV space, and is defined as follows.

$$D(i,j) = \left|\overline{Y}_i - \overline{Y}_j\right| + \left|\sigma_Y^i - \sigma_Y^j\right| + w*(\left|\overline{U}_i - \overline{U}_j\right| + \left|\sigma_U^i - \sigma_U^j\right| + \left|\overline{V}_i - \overline{V}_j\right| + \left|\sigma_V^i - \sigma_V^j\right|) \qquad (2.3)$$

where $\overline{Y},\overline{U},\overline{V}$ are the average Y, U and V values computed from the DC images of the frame *i* and *j*; $\sigma_Y,\sigma_U,\sigma_V$ are the corresponding standard deviations of the Y, U and V channels; *w* is the weight on chrominance channels U and V. The frame-to-frame color difference is computed between the I-frame and its previous P-frame (Eq 2.4). Note that the DC image of a P-frame is interpolated from its previous I- or P- frame based on the forward motion vectors.

$$D_{fram-to-frame}(i) = D(i, i - M - 1) \tag{2.4}$$

where M is the number of B-frames between a pair of successive anchor frames.

The long-term color difference is computed between the I-frame and its *k*th previous P or I frame, which is :

$$D_{long-term}(i) = D(i, i - (M + 1) * k) \tag{2.5}$$

where *k>1* and is usually set to the range from 5 to 10, which corresponding to a 0.2 second to 0.4 second time interval for typical MPEG videos.

For a P-type frame, the computation of frame-to-frame and long-term color differences is the same as an I-type frame. Color statistics are extracted from interpolated DC images. In addition, the motion measure *Rp* is computed. *Rp* is the ratio of intra coded blocks to forward motion vectors in the P-frame (detail can be found in [69]). Here forward motion vectors in a P-frame refer to motion estimation from its former I or P frame.

For a B-type frame, we only compute two motion-based measures, *Rf* and *Rb*. *Rf* is the ratio between forward and backward motion vectors in the B-frame. *Rb* is the ratio between backward and forward motion vectors in the B-frame (again, detail can be found in [69]). Note that backward motion vectors in a B-frame refer to motion estimation from

its next anchor frame, while forward motion vectors refer to motion estimation from its former anchor frame, all in the display order.

2.4 Flashlights Detection

Flashlights occur frequently in home videos (e.g. ceremonies) and news programs (e.g. news conferences). They cause abrupt brightness changes of a scene and will be detected as false scene changes if not handled properly. We apply a flash detection module before the scene change detection process. If a flashlight is detected, the scene cut detection is skipped for the flashing period. As we will demonstrate, when a scene cut happens at the same time of a flashlight, our algorithm will not detect the flashlight and can still detect the scene cut correctly.

Flashlights usually last less than 0.02 second. Thus for normal videos with 25 to 30 frames per second, one flashlight will affect at most one frame. An example of flashlights is show in **Figure 2.3**. As we can see, the affected frame has a very high brightness.

a. the frame before a flashlight

b. the frame of a flashlight

Figure 2.3 The effect of flashlights on a scene (video shot by Prof. Shih-Fu Chang)

Based on our observation of home videos, a flashlight causes the following changes in

a recorded video sequence. First, it may generate a bright frame. Note that because the frame interval is longer than the time of flashlights, a flashlight does not always generate the bright frame. Secondly, a flashlight often causes the aperture change of a video camera, and generates a few dark frames in the sequence right after the flashlight. The average intensities over the flashlight period in the above example are shown in **Figure 2.4**.

As shown in **Figure 2.4**, the intensity jumps to a high level at the frame where the flashlight occurs. The intensity goes back to normal after a few frames (e.g., 4 to 8 frames) due to aperture change of video cameras. On the contrary, for a real scene cut, the intensity (or color) distribution will not go back to the original level. Based on this feature, we use the ratio of the frame-to-frame color difference and the long-term color differences, to detect the flashes. The ratio is defined as follows.

$$Fr(i) = D(i, i-1) / D(i+\delta, i-1) \tag{2.6}$$

where i is the current frame, and δ is the average length of aperture change of a video camera (e.g. 5). If the ratio $Fr(i)$ is higher than a given threshold (e.g. 2), a flashlight is detected at the frame i. Obviously, if we use the long term color difference at frame $i+\delta$ to detect flashlight at frame i, this will become a non-causal system. In actual implementation, we need to introduce a latency not less than δ in the detection process. Also, in order to determine the threshold value, we use a local window centered at the frame being examined to adaptively set thresholds.

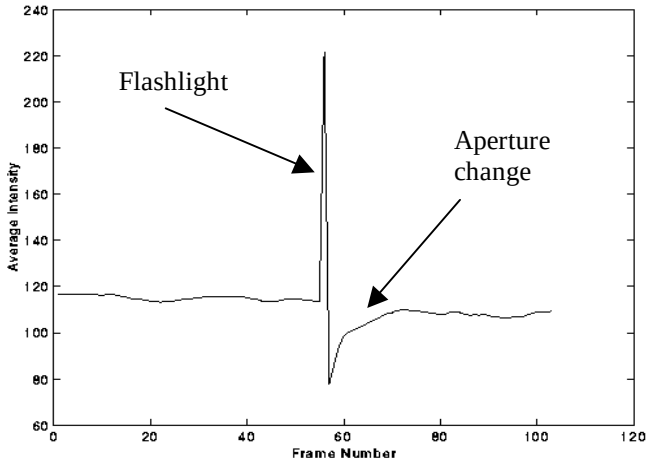


Figure 2.4 Typical intensity changes in a video sequence due to a flashlight

Note that the above flash detection algorithm only applies to I- and P- frames, as we do not extract color features at B frames. However, a flashlight occurring at a B-type frame (i.e. bi-direction projected frame) does not cause any problem in the scene cut detection algorithm we adopted and modified in [69]. This is because a flashed frame is almost equally different from its former and successive frames, and thus forward and backward motion vectors are equally affected [65].

In occasions where a scene cut happens at or right after a flashlight, the flashlight will not be detected because the long-term color difference is also large due to the scene cut. As our goal is to detect actual scene cuts, misses of flashlights are acceptable. Furthermore, as we will discuss in the next section, our algorithms are able to pick the right scene cut position under this situation.

2.5 Scene Cut Detection

Given the color and motion measures of the frame-to-frame differences, the scene cuts can be detected by identifying peak values of these measures. As scene changes in videos

from different sources usually have different characteristics, it is hard to set a global threshold that can detect peak values in different videos. Even within the same video (e.g., a news program), different parts may have different levels of peak values. To solve the problem, we use a local window to detect peak values. The size of the window is usually 30 to 60 frames, and is centered at the frame that is being examined for scene cuts.

Assume the size of window is $2*\delta+1$, feature values for each frame are divided by their corresponding average values over the window $[i-\delta, i+\delta]$. These new peak-to-average ratios (*PA*) are defined as follows.

$$PA_T(i) = \frac{T(i)}{\sum_{k=i-\delta}^{i+\delta} T(k) \Big/ (2*\delta+1)}, \quad \text{where } T \in (D_{frame-to-frame}, R_p, R_b, R_f) \tag{2.7}$$

Note that because different frame types have different features, the sum in the above equation is conducted only over the frames where the corresponding feature is available. For an I frame, *PA* of $D_{frame-to-frame}$ is computed. For a P frame, *PA's* of $D_{frame-to-frame}$ and R_p are computed. For a B frame, *PA's* of R_b and R_f are computed.

Given all these *PA* ratios within a local window, it is not trivial to combine them in a single decision process that detects scene cut on a given frame. One approach is to try different combinations manually, and then compare their performances to find the most appropriate model. Considering there are many thresholds involved in various combinations, such manual selection process are complex and time-consuming. In this work, we adopt a decision tree based learning process to find proper decision models and approximated thresholds.

2.5.1 Combining Color and Motion Features Using Decision Tree

Decision tree is a popular, simple machine learning technique. It involves a tree in which a non-leaf node is labeled with a feature. The branches at the non-leaf node correspond to the possible values or ranges of the feature. As an example in **Figure 2.5**, the feature at the top level is frame-to-frame color difference, the branches below the node are the possible value ranges of the feature, e.g., more than 100 and less or equal 100. Leaf nodes are labeled with a class, i.e. scene cut or no scene cut. Decision trees are used for classifying instances - one starts at the root of the tree, then, taking appropriate branches according to the feature at each branch node, and eventually comes to a leaf node. The label on that leaf node is the class for that instance.

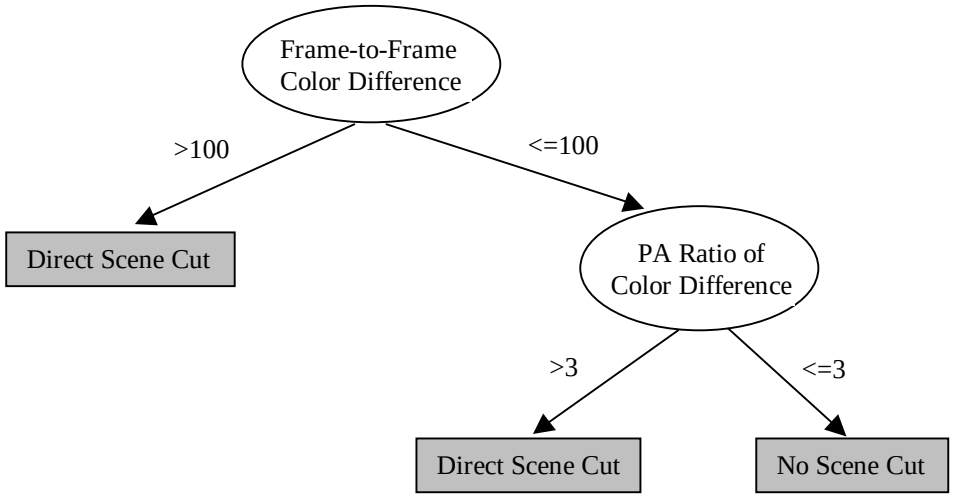


Figure 2.5 An example of decision tree

Similar to other machine learning techniques, feature selection is the key to success in developing decision tree techniques. Simply putting all measures together without identifying proper features will result in ineffective solutions. Based on the characteristics of compressed data, we choose to build scene cut decision trees for I, P and B frames separately. This is because different types of frames have different characteristics as well

as features. We expect that decision rules are different for different types of frames. Furthermore, direct scene cuts and gradual transitions are also handled separately because their detection models are different. Gradual transitions last longer than direct scene cuts, and will be detected only based on color differences. Thus we need to build and train four different decision trees: three for direct scene cut detection and one for gradual transition detection.

Our training videos include baseball, news, sitcom and home videos. We manually labeled each scene cut and its corresponding frame type. We use the public domain induction tool OC1 (Oblique Classifier 1 [77]) to build our scene cut classifier. Oblique decision tree methods are tuned especially for domains in which attributes are numeric. After the training, we manually prune and merge deep level nodes in the output trees to obtain simplified final decision models. These models are discussed in the following sections, i.e., 2.5.2 to 2.5.4.

2.5.2 Direct Scene Cut Detection

Direct scene cuts are detected at all three types of frames. As we mentioned before, detection of direct scene cuts is relatively easy. If we detect a peak difference within a local window, it indicates a scene cut. Here we examine the PA ratios to find peak values. The decision tree is trained to learn the order of the color and motion features to be compared in the detection process.

For the *k*th frame, if it is an I frame, we use the following PA ratios : $PA_{D_{frame-to-frame}}(k)$, $PA_{D_{frame-to-frame}}(k+M+1)$ and $PA_{Rb}(k+j)$ where *j=1…M*. Here *M* is the number of B frames between the I frame and its next P frame. Note here the frame numbers are in

transmission order. An example of *M=2* is shown in **Figure 2.6**. Note here frame *k+1* and *k+2* are B frames displaying before the I frame (*k*), but being transmitted afterwards due to the frame prediction procedure used in MPEG.

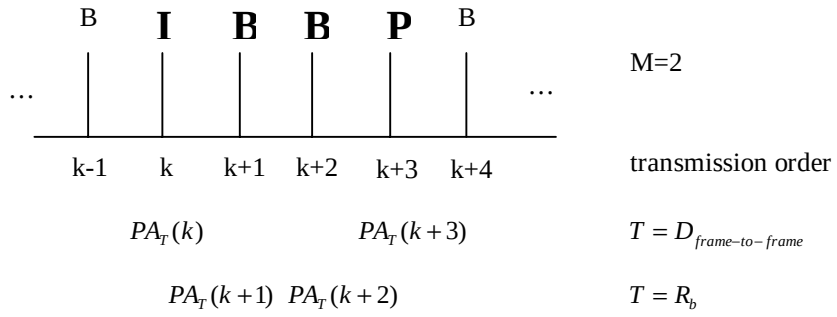


Figure 2.6 PA ratios used in I frame scene cut detection

In addition to $PA_{D_{frame-to-frame}}(k)$ that is used to check if there is a peak at frame *k*, $PA_{D_{frame-to-frame}}(k+M+1)$ is checked to make sure that there is no peak value at frame *k+M+1* (as we can safely assume there is no two scene cuts with a few frames). This is mainly to handle very fast camera motions that result in large frame differences in consecutive frames (instead of only in one frame for regular direct scene cut). $PA_{Rb}(k+j)$ is checked to see if the scene cut occurs at the frame *k+j* [69], which is displayed before the frame *k*. This is because the frame *k* is compared with its former P frame to obtain the frame-to-frame color difference, and will also have a peak difference when a scene cut actually occurs at one of the B frames before it (in the display order). The decision tree derived is shown in **Figure 2.7**.

The threshold *TH* to detect peak PA ratios of the frame-to-frame color difference is about 5 to 6, which are obtained from the training process. The *TH_Rb* to detect peak PA

ratios of R_b is around 2 to 3. As discussed in [69], a large R_b indicates that there is a direct scene cut at the B frame. The optimal thresholds are slightly different for different type of videos (e.g., baseball and home video). As we will discussed in the section 2.5.4, in practice, a general multi-level schema can be used to enable users to easily correct false alarms and misses.

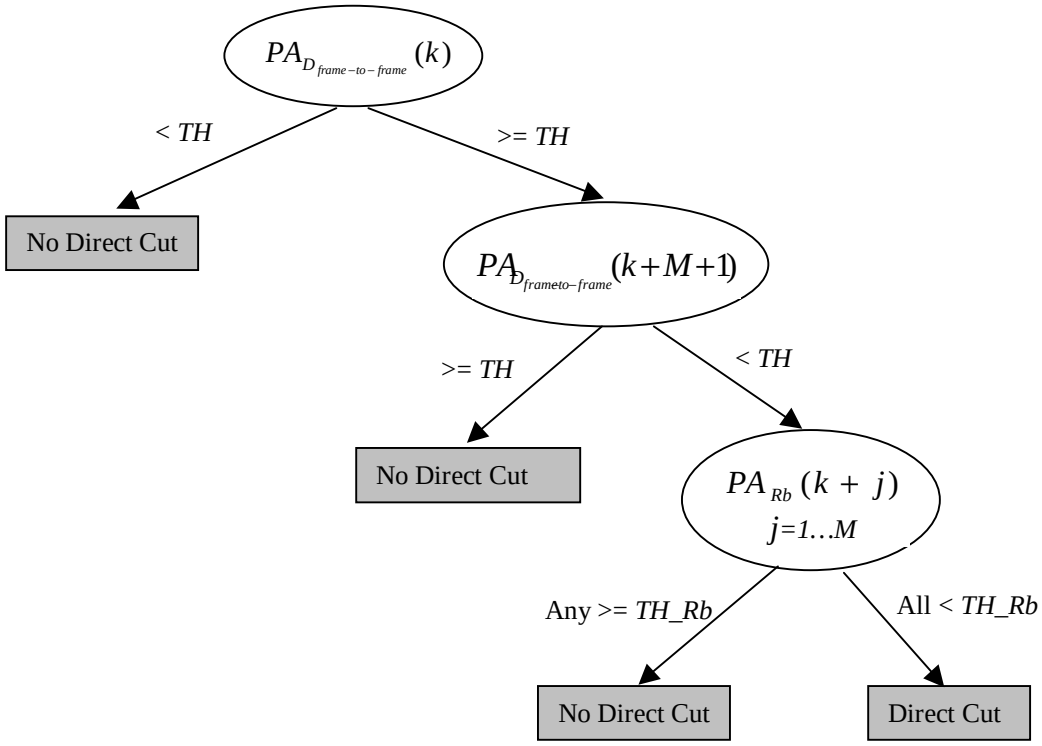


Figure 2.7 Direct scene cut detection at I-type frames

If the *k*th frame is a P frame, $PA_{Rp}(k)$ is checked in addition to the features that are checked for an I frame, including $PA_{D_{frame-to-frame}}(k)$, $PA_{D_{frame-to-frame}}(k+M+1)$ and $PA_{Rb}(k+j)$ where *j=1…M*. Its decision rules are summarized in **Figure 2.8**.

Similarly, a peak *Rp* or $D_{frame-to-frame}$ ratio indicates a potential direct scene cut. The thresholds are *TH_Rp* and *TH* for motion and color respectively. In addition, if the two relatively lower thresholds, *TH_Rp1* and *TH1* are both satisfied, a possible scene cut is

also declared. This is because the DC image of a P frame is interpolated from its leading I or P frame, and its computed frame-to-frame color difference may be smaller than the actual difference. The same validation process is followed to make sure there is no peak at the frame *k+M+1,* and there is no scene cuts at its following B frames. Here *TH_Rp* is in the range of 15 to 25.

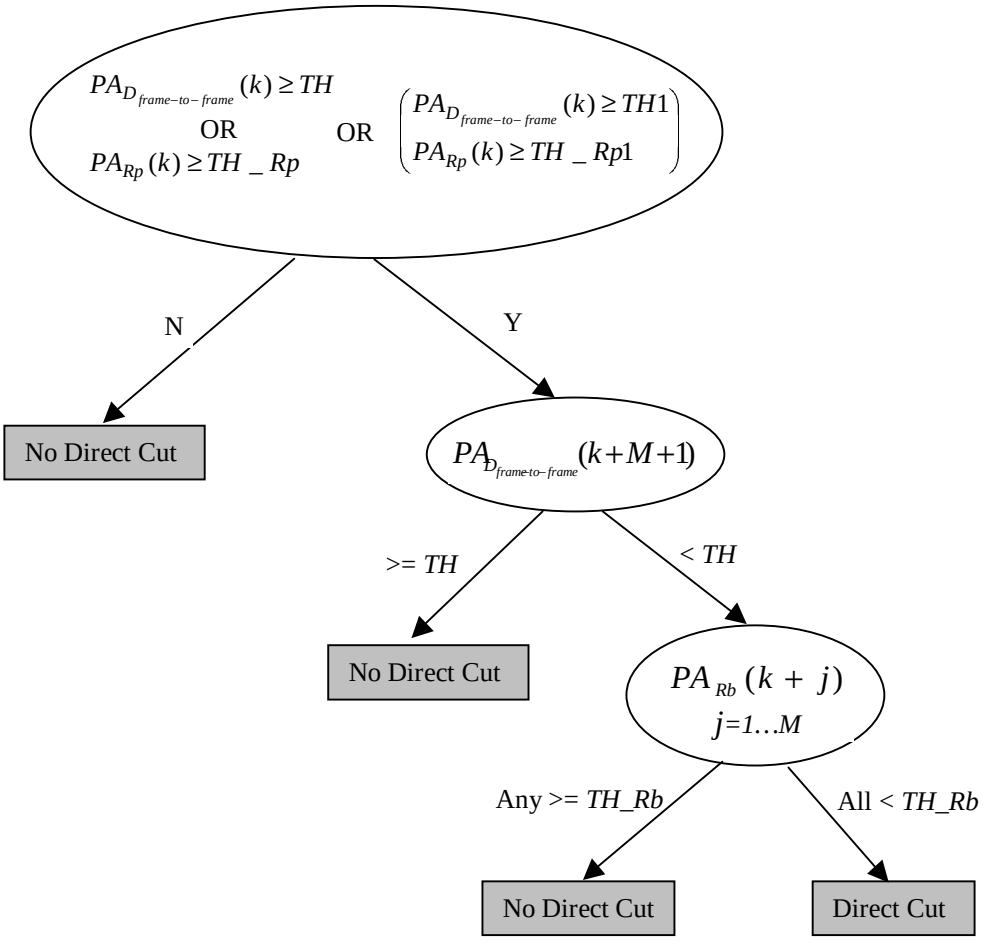


Figure 2.8 Direct scene cut detection at P-type frames

For a B-frame, we check PA_{Rb} and PA_{Rf} ratios at the frame *k* and its successive B frames. As shown in **Figure 2.9**, assume *L* is the frame number of the last B frame before the next anchor frame, the *k*th frame is a direct scene cut point if PA_{Rb} value of the frame

k and all its following B-frames are larger that the given threshold *TH_Rb* . If the PA_{Rf} values of the frame *k* and all its following B-frames are larger than the threshold *TH_Rf*, its past I or P frame (in transmission order) is detected as a scene cut. This approach is an enhanced version of what has been proposed in [69]. Instead of only examining one B-frame, the new approach checks more B-frames to improve accuracy and robustness.

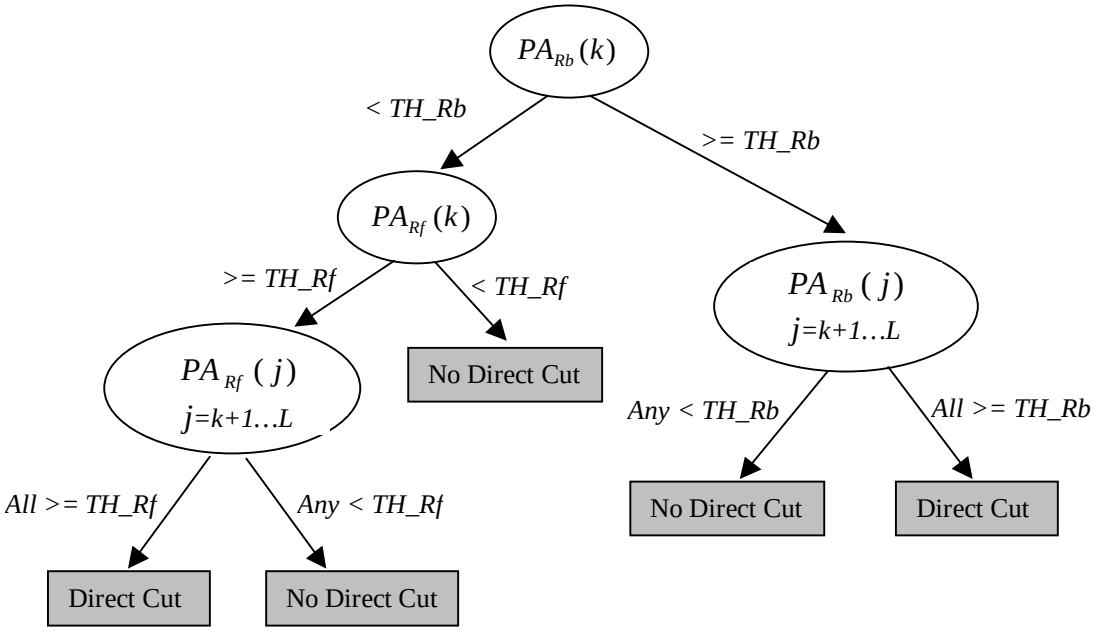


Figure 2.9 Direct scene cut detection at B-type frames

2.5.3 Gradual Transition Detection

If no direct scene change is detected, the algorithm checks for gradual transitions that do not show high peak values in above modules. The widely used twin-comparison algorithm is designed to track a transition assuming that frame-to-frame differences do not drop below a threshold within the whole period. However, for long transitions in some videos (e.g. sports or sitcom), differences may drop to a very low level for a few frames within transitions. This will cause misses and/or falses in the twin-comparison

method. Other researches have developed algorithms to compute the intensity variance and then detect parabolic curves to find dissolve or fade related transitions. Due to noise and motion, it is hard to find desired parabolic shapes without introducing many false alarms.

Using the learning method of decision tree, we find it is rather simple and robust to detect the beginning and ending edges of transitions, which have the shape of up and down steps respectively. An example transition is shown in **Figure 2.10**, b1-b6 and e1-e6 are PA ratios of frame-to-frame color differences computed at anchor frames (Eq 2.4).

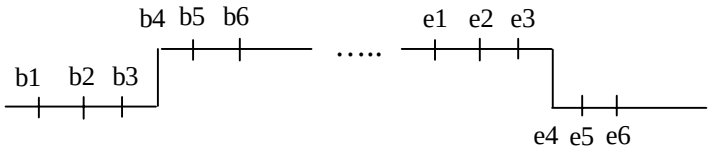


Figure 2.10 The beginning and ending edges of a gradual scene change

Based on induction results from the decision tree learning, we use color differences at six successive anchor frames to detect the beginning and ending steps (note that we do not compute color measures at B-frames). The final decision rules are summarized as follows.

- If b1, b2 and b3 are smaller than *TH_G1*, and b4, b5 and b6 are larger than *TH_G2*, a beginning edge is detected at the frame of b4
- If e1, e2 and e3 are larger than *TH_G2*, and e4, e5 and e6 are smaller than *TH_G1*, an ending edge is detected at the frame of e4

The above rules are inducted from MPEG-1 sequences with 30 frames per second and two B-frames between each pair of anchor frames. Thus, in general, we need to examine a window of about half second to properly detect steps. From our experiments, the threshold *TH_G1* is around 1.1 to 1.3; the threshold *TH_G2* is from 0.7 to 0.8. As frame-

to-frame variances always exist at the boundaries of a transition (which may not be true within a transition), and are usually noticeable (i.e. large), our detection method can pick beginning and ending edges with a high recall rate.

The remaining problem, which also exists in other scene cut detection algorithms, is that fast camera or object motions may produce similar up and down steps, which will cause false positives and therefore reduce precision. To alleviate the problem, we check the distance between a pair of beginning and ending steps. In detail, when a ending step is detected, the distance with the last beginning step, *L*, is computed, and

- If *L>L1 and L<L2,* then there is a transition. Otherwise there is no transition.

Here *L1* is the minimum length of a transition (e.g. 10 frames or 0.3 second). *L2* is the maximum length of a transition (e.g. 60 frames or 2 seconds).

The length constraint removes most false alarms since up and down steps caused by motions typically do not come up in pairs within short periods. One most likely false situation is when there is a sudden camera motion and the motion also stops suddenly in a short time. Slow camera motions usually do not create step-like changes. This is acceptable in many applications because the above camera action quickly changes the recording view and may be considered as a “true’ scene cut.

2.5.4 Multi-Level Scene Cut Detection

A scene cut detection with 100 percent accuracy is not realistic, even though we have used various methods to help us choose effective decision models and threshold values and consider many important issues. In practice, we have noticed that given a good browsing interface (such as the one we will show in Chapter 5), it is easy for users to

identify and correct false alarms (assume there are a limit amount of errors). On the other hand, it is hard to correct a missed scene cut without playing and watching the whole video again. Adjusting some decision thresholds to lower values can minimize misses, but it usually causes many false alarms.

To solve this problem, we adopt a multi-level scheme for exploring this tradeoff situation. In this scheme, multiple sets of thresholds are used instead of just the optimized threshold values. Scene cuts are detected at different levels. The multi-level schema is shown in **Figure 2.11**.

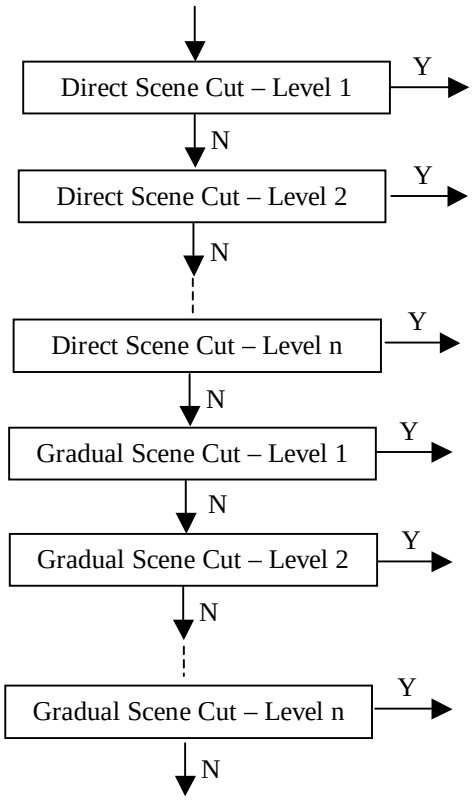


Figure 2.11 The multi-level scene cut detection scheme

For each frame, the detection process goes from a higher level to a lower level when a scene cut is not detected at the higher level. The process stops whenever a scene cut is detected. Because obviously, scene cuts at one level are also scene cuts at the lower

levels, the output of a scene cut includes the frame number as well as its detecting level.

In practice, because the direct scene cut detection is usually accurate, we only apply more than one levels in the gradual scene cut detection stage. Gradual scene changes, such as dissolve and fade, are likely to be confused with fast camera panning/zooming, motion of large objects and light variance. A high threshold will miss scene transitions, while a low threshold may produce too many false alarms. Our multi-level approach generates a hierarchy of scene cuts. Users can quickly go through the hierarchy to see positive and negative errors at different levels, and correct them.

2.6 Verification Using Complex Features

As we discussed before, the main problem in the scene cut detection is to distinguish between real scene breaks (especially gradual transitions) and normal changes, such as camera or object motions and lighting changes. Many features and methods have been studied and are proven to improve the detection accuracy. However, most of these approaches require more complicated feature extraction algorithms, such as motion estimation and edge detection. Typically, these algorithms are computation intensive and cannot be performed in real-time on a regular workstation or PC.

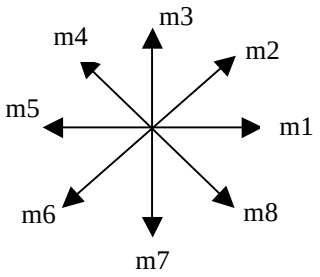
Here, we develop a verification mothod to increase detection accuracy without losing the real-time performance. In this scheme, extraction and comparison of more detailed visual features are applied only when a potential gradual transition is detected. Note that this process can also be included in our multi-level scene cut detection scheme discussed in the previous section. Complex models and features are applied only to the candidate frames.

In the following sub-sections, we will present two methods for final verification - camera motion detection and aperture change detection. Typically, these methods require higher resolution and accuracy of extracted features.

2.6.1 Camera Motion Detection

In this work, we focus on the detection of camera panning operations. Compared to object motion and camera zooming, panning operations usually produce much larger visual content changes (e.g. color), and thus cause more false alarms. On the other hand, panning can also be detected more reliably compared with other motion activities. The method we utilized here looks for dominant direction of motion vectors as an indication of the camera panning.

We first compute the histogram of eight motion directions for every P-frame based on motion vectors that are available in the MPEG compressed domain (**Figure 2.12**). This feature extraction process does not require much computation as the number of motion vectors (i.e. macro blocks) is small (330 for a CIF size frame). The detection of panning at P-frames is applied only when a potential gradual scene change is found.



m9 : number of intra coded macro-blocks
m0 : number of macro-blocks with zero motion

Figure 2.12 Panning detection based on histogram of motion vectors

The algorithm to detect panning in a P-frame is given as follows.

```
Let i be the direction with maximum number of motion vectors (1
to 8)
    j be the direction with second most motion vectors (1 to 8)

if (mi>m9) and (mi>m0) and (mi/mj>2)  then
    panning
else
  no panning
```

We find the dominant motion directions. If there are more motion vectors in the dominant direction than the intra coded blocks as well as the blocks with zero motion, we compare the dominant motion direction with the second dominant one. If the former contains a lot more vectors than the later one does (e.g. twice, as we used in our implementation), a panning is detected.

At the ending edge of a potential gradual transition, the above panning detection method is applied to all the P-frames within the transition period. If pannings are detected in P-frames more than a certain percentage (e.g. 50%), we treat this change as a panning instead of a gradual transition.

Note that there are special edit operations in some movies or sports that result in effect similar to the camera panning. One example is the wipe with a new scene coming in and an old scene going out. In such cases, camera motion detection can be used as an optional module when needed in specific domain.

2.6.2 Aperture Change Detection

Aperture changes of a video camera happen occasionally in movies, news programs, and especially home videos. Aperture changes are usually caused by changes of lighting condition. For example, when a camera is panned from a bright scene to a dark scene, the aperture will gradually open wider to receive more light. This process causes image

intensity changes over a short period in the recorded video (**Figure 2.13**).

a. panning to dark scene **b**. statrt of aperture change **c**. the changed aperture

Figure 2.13 An example of aperture change that generates two potential transitions (videos shot by Prof Shih-Fu Chang)

The example in **Figure 2.13** causes two falsely gradual transitions, one from bright to dark and the other from dark to bright. To solve this problem, we compare the chrominance and edge features of an I-frame right after a transition (potential) with features of an I-frame before the transition. If the differences are smaller than a threshold, the transition is ignored.

Chrominance histograms and edge direction histograms are computed from the two decoded I-frames. The reason to use I-frames is because an I-frame requires less decoding computation and it does not depend on other frames. The chrominance histogram is calculated in the HSV space using only H (hue) and S (saturation) values. The luminance values are not compared as they are sensitive to lighting changes. We use a 36-bin histogram that has 12 levels of H and 3 levels of S. The color difference is computed using the L1 distance measure.

$$D_{color} = \sum_{k=1}^{36} \left| H_1(k) - H_2(k) \right| \tag{2.8}$$

Edge histograms are counted at 16 equally divided directions by calculating the gradient of each edge pixel. Here edge pixels are detected using Sober operator. For an

edge pixel (i,j), let dx=f(i)-f(i-1) and dy=f(j)-f(j-1), its gradient is arctan(dy/dx). The distance between two edge histograms E1 and E2 is defined as follows.

$$D_{edge} = \sum_{k=1}^{16} \left| E_1(k) - E_2(k) \right| \tag{2.9}$$

Given D_{color} and D_{edge} between two I-frames at the begin and end of a potential transition, if they are both under given thresholds (e.g. around 0.5 in our implementation), an aperture change is declared and the transition is ignored. This is based on the consideration that edge and chrominance features are less affected by aperture changes. For a real transition, these differences are expected to be larger. The other illumination invariant features can also be used [107].

2.7 Results and Discussions

We developed a real-time scene cut detection system using the scheme described in previous sections on a Pentium-III 600 PC. The PC has a FutureTel MPEG compression card that can capture and compression live video feeds from VCR or cable. This system allows us to test our scene change detection algorithms on a large amount and wide variety of video sources.

In our experiments, we have used a total of around 5.5 hours videos from different sources. As listed in Table 1, there are two half-hour baseball videos from ESPN and Fox respectively. The two tennis videos are from games at two different places. Home videos are obtained from two different personal owners. There are also half hour videos from sitcom, news, cartoon, movie and trailer. Most of these videos contain commercials. They are recorded to VHS tapes from TV broadcasting channels, and then digitized and

compressed to MPEG-1 streams at 30 frames per second and CIF resolution (i.e., 352x240) using the FutureTel MPEG encoding card. The movie file is obtained from a VCD and its original format is MPEG-1. The bit rates of these MPEG videos are from 1.2Mbps to 1.5 Mbps.

Table 2.1. Description of the 6 hours of videos in the experiment dataset

Video Type	Number of Sequences	Length
Baseball	2 (ESPN and Fox)	30x2 minutes
Tennis	2 (two different games)	30x2 minutes
Sitcom	1 (Senfield)	30 minutes
News	2 (CNN)	30 minutes
Cartoon	1 (animals)	30 minutes
Movie	1 (comedy)	30 minutes
trailer	1 (Hot Shots)	30 minutes
home video	2 (two different people)	30x2 minutes

The baseball and tennis videos contain fast object and camera motions. In the meanwhile, different scenes in a game have similar backgrounds as one game is played at the same stadium or court. Home videos are usually very jerky. There are non-smooth camera motions and unexpected aperture changes. Sitcom videos include many back and forth camera angle switches. It is challenging to pure color based approaches because many angle changes only slightly change the color distribution of a scene. News programs have been widely studied. A news video often contains various scenes in different stories. Cartoons are a special type of videos that are usually consisted of computer-generated graphics. Here a scene cut means significant changes of graphical objects within a scene. Commercials exist in all types of videos except the movie and trailer. The half hour movie is extracted from a comedy that is originally recorded on a VCD in MPEG-1 format. The trailer is a promotion video provided by Hot Shots & Cool

Cuts Inc., a stock footage company. It contains a lot of short unrelated shorts that are sampled from the company's large video collections.

We manually identify the ground truth by using a MPEG player with frame accuracy. In our experiments, the scene cut detection results are compared with the ground truth in terms of precision and recall. Assume N is the ground truth number of scene cuts, M is the number of missed cuts and F is the number of false alarms, the recall and precision are defined as follows :

$$\text{Recall} = \frac{N - M}{N} \tag{2.10}$$

$$\text{Precision} = \frac{N - M}{N - M + F} \tag{2.11}$$

These two measures are both important. We certainly do not want to miss any critical scene changes. On the other hand, too many false alarms will compromise the efficiency of indexing and summarization. In practice, as we mentioned before, it is relatively easy for users to manually identify false alarms and improve the precision. Thus in the training stage when the thresholds are selected, we prefer lower threshold values, which tend to give more scene cuts, if recalls are not significantly affected (e.g. reduced less than 5%).

We use two sets of thresholds in the experiments. One is for home videos (i.e. amateur). The other one is for the rest of videos (i.e. professional). Amateur and professional videos have very different characteristics. The former have jerky camera motions compared to smooth camera motions in the professional one. On the other hand, gradual transitions or editions in home videos are simpler and are not as many as those in professional videos. The two sets of thresholds are chosen based on the consideration of these characteristics. At the training stage, we use a small amount of video data (around 1

hour) digitized from sources (including news and home videos) that are different from the above testing videos. Note in most of our detection modules, we do not compare absolute value of a specific measure (e.g., frame to frame difference or motion vector percentage) against a fixed threshold. Instead, we normalized these measures with the local window average and use machine learning tools to automatically choose optimal thresholds. By doing these, we were able to generalize the algorithms and make them applicable to different type of videos.

The detailed experiment results are shown in the following tables (Table 2.2 to 2.6).

Table 2.2 Detection results of all scene cuts for 8 different types of videos

Video	# of scene cuts (ground truth)	# of false	# of missed	recall	precision
baseball	891	74	37	96%	92%
tennis	622	50	39	94%	92%
sitcom	461	35	23	95%	93%
news	276	32	7	97%	89%
cartoon	313	42	5	98%	88%
movie	225	18	31	86%	92%
trailer	655	22	18	97%	97%
home video	184	65	9	95%	73%
TOTAL	3627	338	169	95%	91%

Table 2.3 Total missing number of direct scene cuts

# of direct cuts (ground truth)	# of missed	Recall
3263	107	97%

Table 2.4 Total missing number of gradual transitions

# of transitions (ground truth)	# of missed	recall
364	62	83%

Table 2.5 Two different reasons that cause false alarms

Total false # in test videos	Camera or object motion	Lighting or Aperture Change
338	266 (79%)	72 (21%)

Table 2.6 Flashlights detection results of home videos

# of flashlights (ground truth)	# of missed	recall
64	18	72%

For the baseball videos, we have good recall and precision results. Considering the total length of baseball videos is about 60 minutes, there are about 1.2 false alarms per minute (i.e. per 1800- frames), and about 1 miss every two minutes. Many false alarms (~50%) come from close-up scenes when a person suddenly appears in front of the camera, which totally changes a scene. The rest of false alarms are mainly from panning and zooming when cameras are following player in the field. Most misses occurred at gradual transitions that are inserted between games and replays. Similar precision and recall results are obtained for tennis. Shots in tennis videos are longer (~ 1.5 times) than those in baseball videos. Camera and object motions in long shots causes most falses, as the large changes are more likely to happen in a long panning or zooming shot. Again, the wiping of company logos as transitions between game and replays accounts for many misses.

Our algorithms perform well on sitcom videos. Many direct scene changes in sitcoms do not have large color differences when two successive shots are taken from the same scene. Motion features are helpful in detecting such changes without bringing in too many false alarms. The precision rate is relatively low for news videos (about 89%). Most false positives are caused by camera motions during the field reporting segments. Although news stories are taken by professionals, the recording situations in the field are

often not good, and thus we see some jerky camera motions. The recall rate is very high for cartoon videos, while many computer-simulated camera and object motions tend to be confused with scene changes. The detection results for the trailer video are very good due to the fact that most scene changes are direct scene cuts between un-related video shots. The movie sequence has a lower recall than precision. Dark scenes taken at night account for many missing scene cuts. Also there are more advanced special effects and edits being used in the movie. Note all the above results are obtained by using the same set of threshold values.

Higher motion thresholds (*TH-Rb, TH-Rf and TH-Rp*) are used for home videos (about 25% higher) in order to tolerate jerky camera motions. As expected, we have a high precision recall, but a low precision rate (72%). Our testing home videos have many long shots (e.g., a few minutes long), in which the camera often changes from one view to another several times. If we compute the number of false alarms over time, it is about 2 falses every minute. Typical camera operations that produce false scene cuts include 1) fast moving from one angle to another; 2) zooming in too close to an object; and 3) following a fast moving object.

As mentioned, direct scene cuts and gradual scene cuts have completely different characteristics. It is appropriate to evaluate their performances separately. When we count the number of missing scene cuts, we also label each change as either direct or gradual. The final results are shown in Table 2.3 and 2.4 respectively. For direct scene cuts, we achieved a very high 97% recall rate on average. The recall rate is 83% for gradual scene changes. Our experiments show that to improve the later recall rate to 90% introduces too many false alarms.

For false scene cuts, we classify them into two classes: motion-related and lighting-related. The motion-related refers to both camera and object motions, which often occur at the same time. The lighting-related includes real lighting changes as well as aperture changes. As shown in Table 2.5, 80% false alarms are motion-related.

Detection of flashlights in one of our home videos is shown in Table 2.6. The recall rate is 72%. Most misses, which result in false scene cuts, are caused by subsequent aperture changes after flashlights (**Figure 2.4**). When these changes are longer than the size our detection buffer window (about half second), the ratio Fr in Eq. 2.6 falls below our threshold and causes miss detection of a flashlight. We can increase the length of local window to improve flashlight detection accuracy.

In summary, we performed extensive tests on our scene cut detection scheme. It achieves the best results for sports and sitcom videos. News and cartoons are slightly less accurate, and the most challenging ones are home videos. This ranking is consistent with the degree of irregularity of camera motions in different types of videos. It indicates that motion is the main issue, and we need advanced methods that can capture motions more accurately. Utilizing better motion estimation techniques and exploiting more complicate features (e.g., region level features) are promising directions, although at the cost of computational complexity.

Chapter 3

Video Region Segmentation and Search

3.1 Introduction

Temporal video segmentation using scene cut detection algorithms have been widely applied in online video indexing and retrieval systems [6,126]. As long video sequences are broken into elementary units (i.e. shots) of contiguous frames, compared to traditional frame based fast forward and backward approaches, users can retrieve desired content more efficiently. However, for most video contents, the temporal segmentation produced by scene cut detection algorithms are still at a very low level. There are usually a large number of shots in a one-hour video (e.g., ~ 1000). Furthermore, users often want to browse and search videos at a higher semantic level. For a news program, it is more helpful to divide videos into stories instead of elementary shots.

Some researches have been done to group video shots into logical scenes or stories [118,129]. In [129], we used a hierarchical clustering method to organize video shots according to their visual and motion features. In [126] and [53], anchor and commercial shots are recognized to find news story boundaries. The transition graph is proposed in [118] to simulate the content flow of a video sequence, and a temporal constrained clustering algorithm is used to combine shots into “scenes”.

When many different models can be used to conduct semantic level temporal

segmentations for specific domains, extraction of meaningful intermediate or high level features, such as regions or objects, from video data is a fundamental problem that needs to be addressed in nearly all video indexing and summarization systems. As current videos are represented only in raw pixels with color or brightness information, spatial segmentation is a necessary step to support efficient feature extraction and analysis. Recently, for the purpose of object based video representation and indexing, generic and flexible segmentation algorithms are being studied by many researchers [47,28,40]. How to achieve consistent segmentation results and how to build correct mapping between image partitions at individual frames are two main issues to be addressed.

In this chapter, we will study an automatic video region segmentation method, and demonstrate how it is applied in a content-based video retrieval system. The general system flow is shown in **Figure 3.1**.

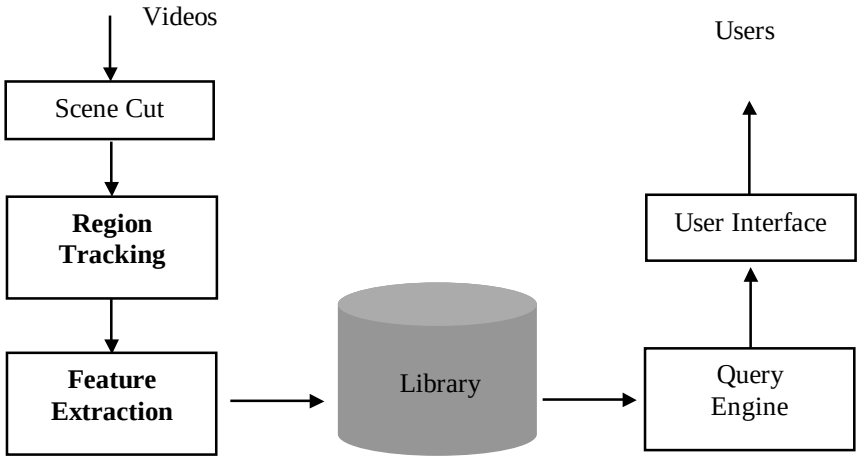


Figure 3.1 Automatic region tracking for content based video search

Here we will focus on the issues of region tracking and feature extraction. The issue of semantic video object tracking and indexing will be addressed in chapter 5, and a high level video structure parsing and event detection system for efficient video retrieval will

be discussed in chapter 6. Following we give an overview of regions segmentation and indexing techniques we will study in this chapter. Detailed descriptions will be given in the rest sections.

3.1.1 Image Segmentation Techniques

Image segmentation is one of the most important steps for feature extraction and analysis. The goal is to segment images into parts with homogeneous properties over space and time, such as color, texture, motion, and spatial-temporal structures. This can be formally defined as follows: if *F* is the set of all pixels in an image, and *M()* is the measure of homogeneity of an area of connected pixels, then segmentation is a partition process which divides *F* into a subsets of connected pixels $F_1, F_2, ..., F_n$ such that

$$M(F_i) = true, \quad \forall i \in \{1,..n\} \tag{3.1}$$

$$M(F_i \cup F_j) = false \text{ if } F_i \text{ is adjacent to } F_j \tag{3.2}$$

$$\bigcup_{i=1}^{n} F_i = F \quad \text{and} \quad F_i \cap F_j = \phi \quad \text{if } i \neq j \tag{3.3}$$

According to [37], image segmentation can be divided into at least two phases that have different definitions of homogeneity. The first phase is a partial segmentation, in which extracted regions with homogenous features do not correspond directly to semantic objects. The second phase is a complete segmentation, which results in semantic objects with their underlying regions. While partial segmentation can be achieved using some general algorithms, it usually requires specific domain knowledge to obtain a complete segmentation.

Basic image segmentation techniques can be generally classified into three categories, which are thresholding, edge detection and region growing. Threshold-based

segmentation transforms an input image *f* into an output binary image *g* based on a given threshold *T* as follows:

$$g(i,j) = \begin{cases} 1 & for\ f(i,j) \geq T \\ 0 & for\ f(i,j) < T \end{cases} \tag{3.4}$$

where *i* and *j* are X and Y coordinates, *g(i,j)=1* indicates the pixel *(i,j)* belongs to the object, and *g(i,j)=0* means it belongs to the background (or vice versa). *T* is a threshold value in the selected feature space. Given multiple thresholds, we can also segment an image into multiple objects.

Figure 3.2 Edge detection results of an image

Edge-based segmentation uses edge pixels to find object boundaries. Edges are high frequency data that indicate discontinuities of color in an image. How to distinguish between edges and noise (**Figure 3.2**) is the main concern of an edge detection algorithm. A variety of methods have been proposed to properly trace object borders [82] or to match desired shape models [57].

Due to the noisy nature of edge detection, it is hard to construct objects by finding their borders. Region growing methods directly produce partitions based on given homogeneity criteria (e.g, color, **Figure 3.3**). A bottom-up approach groups pixels into regions as far as homogeneity measures are satisfied (Equation 3.1). On the contrary, a

top-down approach split images into regions until homogeneity measures are satisfied. Combined split and merge approaches try to take advantages of both approaches.

Figure 3.3 An example of region growing segmentation (region are shown in random colors)

Advanced segmentation techniques have been developed in all three categories to achieve better performances. Some well-known examples include watershed algorithms that use mathematic morphology for region growing [94], active contour (or snake) approaches that use energy minimizing splines [57], neural network methods that learn context information from training data, and region clustering based on the fuzzy theory [75].

3.1.2 Segmentation and Tracking of Video Regions

Although much work has been done in decomposing images into regions with uniform features, we are still lacking robust techniques for segmenting video data in general video sources, especially when relatively accurate region boundaries are needed. Moving objects segmentation using the motion field or optical flow has been the main focus of many researches. As motion fields are usually noisy for real-world scenes, direct segmentation of them is erroneous and not stable. Model-based motion estimation and segmentation methods are more robust. Recently, Wang and Adelson in [112] presented

an affine-clustering based algorithm. Motion layers are extracted from the original motion field by iteratively estimating and refining affine models. In [8], instead of using optical flow, Ayer and Sawhney proposed a method to estimate motion models and their layer support simultaneously. The number of layers is decided by a formulation based on a maximum likelihood estimation of mixture models and the minimum description length encoding principle. In [73], Meyer and Bouthemy developed a pursuit algorithm to track an object based on the multi-resolution estimation of the affine model from the motion field within the object. In general, the above methods concentrate on segmenting moving objects and cannot track static objects or objects with intermittent motions (e.g., people crossing the street). Furthermore, due to the accuracy limitation of motion estimation, motion segmentation may not give clear object boundaries.

Methods have also been proposed to track feature points or contour segments [31,52]. These methods generate good results when moving objects have strong and stable (over time) localized features such as corners and edges. However, they are sensitive to object deformation as well as object occlusion and image noise. To track non-rigid objects, deformable models have been widely studied. Active contour (i.e., snakes) [57] is one of the basic energy-minimizing elastic contour models. Given the energy function, an elastic contour at the current frame can be moved and deformed iteratively to the best position at the next frame using the snake algorithm. Such methods are able to generate accurate object boundaries for relatively simple background. As snakes require very accurate initialization (thus can handle only slow motion) and are sensitive to textured image regions, many improvements [10,15] as well as new models such as MRF models [58,109] are studied.

Despite these progresses, region segmentation remains a hard problem for video analysis and understanding. One problem with many existing approaches is that segmentation results are sensitive to noises and/or slight variances of features, especially at places around segmentation boundaries. In region tracking, the problem may cause different segmentations at successive frames. When video sequence is short, boundary errors usually do not hurt overall tracking performance seriously. However, when a region needs to be tracked over a long period, accumulated boundary errors are likely to completely break the tracking process. To increase the stability of region segmentation, fusion of various visual features in the segmentation process is an essential approach. As an example, edge-based methods may produce accurate object boundaries but are sensitive to noises. On the contrary, color based region growing methods are robust to noises but usually results in over-segmented regions, and may not be able to generate accurate boundaries (e.g., due to color blur). An efficient method to combine these two features is highly desired to achieve more consistent segmentations.

Another problem is that the mapping between regions at successive frames is not reliable when these regions are segmented independently. Because similar regions often exist within even small local windows, minor segmentation differences and/or motion estimation errors could cause region mismatches. To address this problem, an inter-frame segmentation process needs to be developed to partition an intermediate frame consistently with segmentation results of its preceding frame. This approach avoids the non-reliable afterwards mapping between successive frames.

In this chapter, we will present an automatic video region segmentation and tracking method based on the fusion of color, edge, motion and temporal features. This method

can track video regions stably over a long period, and is especially useful to build visual index of a large video collection.

3.1.3 Region Based Spatial-Temporal Retrieval

While content-based video retrieval has been focused on shot segmentation and key frame based search, some attempts have been made to index videos through object tracking. In [62] and [54], block-based motion estimation methods are used to roughly compute motion trajectories of moving objects. Moving objects are either manually defined or extracted by grouping (i.e., clustering) blocks with similar moving vectors. In [62], moving objects are represented and compared according to their moving directions. In [54], a dynamic programming based method is applied to efficiently compare two motion trajectories. An MPEG-domain moving object tracking method is presented in [32], which directly uses macro-block displacements in P- and B-frames to track macro-blocks.

In above researches, although authors tried to utilize motion information for video indexing, as video objects were not well segmented and tracked by grouping motion vectors, trajectory accuracy is rather limited. In this chapter, we demonstrate a spatial-temporal indexing method using automatically tracked video regions. Region motion trajectory is generated through a global motion estimation process. In addition, other visual features such as color, texture and shape are also captured for each region. A query model using filtering and validation, as proposed in [99], is applied to perform region feature based video search.

3.2 Region Segmentation and Tracking Using Feature Fusion

We define an image region as a contiguous area of pixels with consistent features (e.g., color) in an image frame. It may correspond to part of a physical object, like a car, a person, or a house. A video region is a sequence of instances of the tracked image region in consecutive frames. The region segmentation and tracking process is applied within a video shot to obtain video regions.

3.2.1 Overview

The segmentation and tracking of feature regions is based on the fusion of color, edge and optical flow. Color is chosen as the major segmentation feature because of its consistency under varying conditions, such as change in orientation, shift of view, partial occlusion or change of shape. Compared with other features such as edge, shape and motion, colors (or more precisely, mean colors) are more stable. Edge features are complementary to color information: color captures low frequency information (means) while edge captures high-frequency details (edges) of an image. Thus fusion of them greatly improves segmentation results, especially region boundaries. Different from old merge-and-split methods where the edge is applied after color-based region merge, we propose a new method to fuse edge information directly in the color merging process. Affine motion model is estimated for each region based on the computation of optical flow. It is utilized to track color regions through a video shot.

The basic region segmentation and tracking procedure is shown in **Figure 3.4**. Projection and segmentation is the major module in which different features are fused for region segmentation and tracking. This module is described in **Figure 3.5** and will be

further discussed in detail below. Optical flow of current frame n is derived from frame n and n+1 in the motion estimation module using a hierarchical block matching method. Different from simple block matching methods which estimate motion solely based on minimum mean square errors, this technique yields reliable and homogeneous displacement vector fields, which are close to the true displacements. Taken color regions and optical flow generated from above two processes, a linear regression algorithm is used to estimate the affine motion for each region. The affine transformation is a good first-order approximation to a distant object undergoing 3D translation and linear deformation. Affine motion parameters are further refined by using a log(D)-steps region matching method in the six-dimensional affine space. Through above modules, color regions with affine motion parameters are generated for frame n. Similarly, these regions will be tracked in the segmentation process of frame n+1.

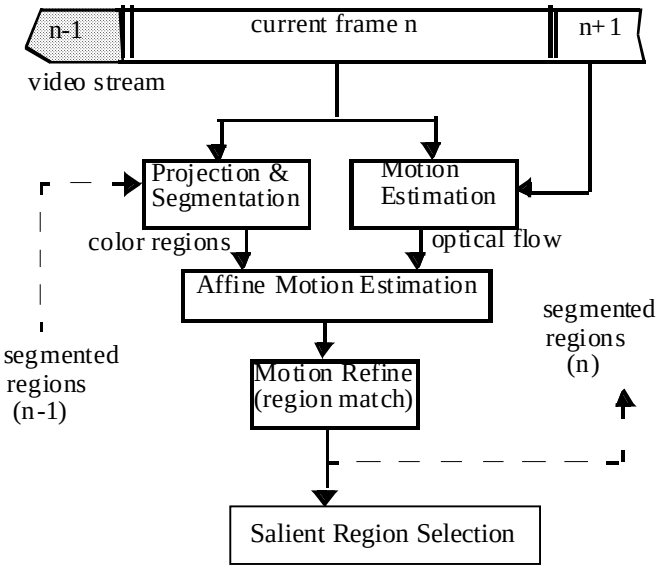


Figure 3.4 The diagram of region segmentation and tracking

A salient region selection module is applied at the final stage to automatically associate regions with high-level semantic objects. Several criteria are adopted to group

or identify major interesting regions. Sizes and durations are utilized to eliminate noisy and unimportant regions: regions with both small size and small duration are eliminated or merged with its neighbor regions. When object motion exists, affine motion models are used to group adjacent regions with similar motions to obtain the moving object.

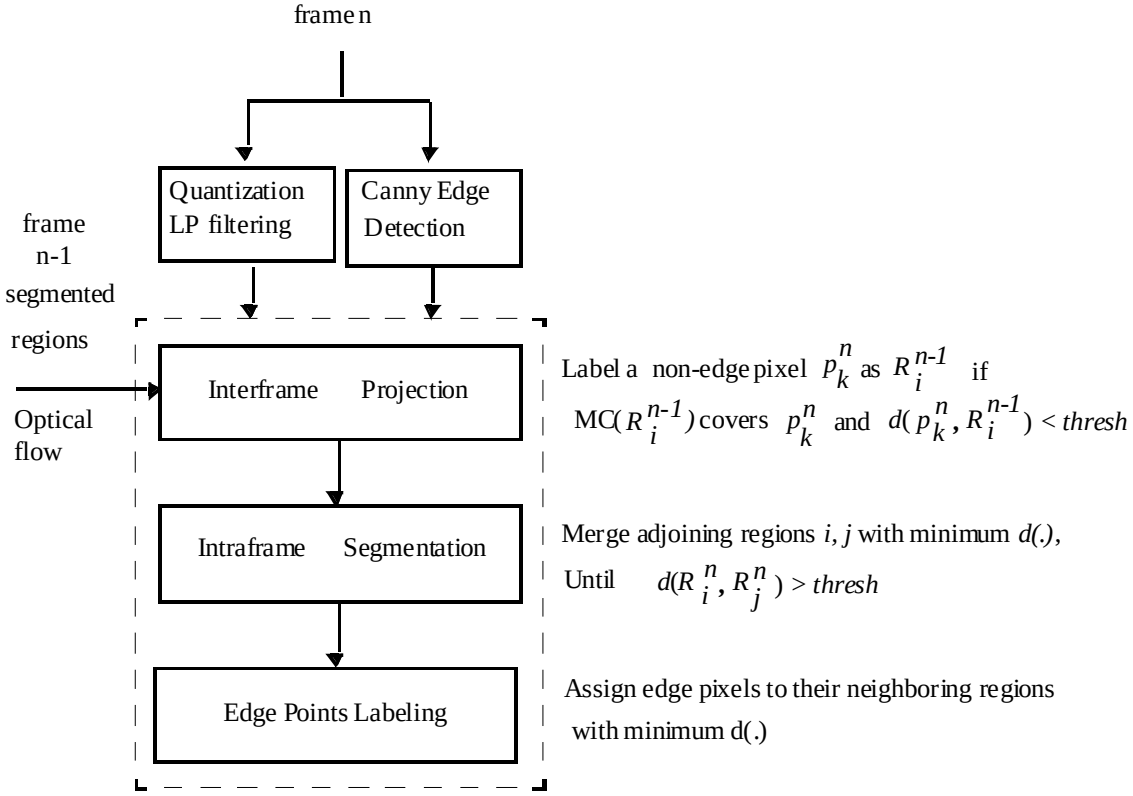


Figure 3.5 The motion projection and segmentation module

3.2.2 Generation of Feature Maps

The segmentation algorithm is developed based on three feature maps: color map, edge map and motion field (i.e. optical flow). These maps are computed at each frame as follows.

The color map is the major feature map in the following segmentation module. We have developed a novel process to generate it. This process contains the following steps:

First, the original image is converted into CIE L*u*v* color space. It is well known that perceptually non-uniform color spaces such as RGB are not suitable for color segmentation, as distance measure in these spaces is not proportional to the perceptual difference. L*u*v* is a perceptual uniform color space which divides color into one luminance channel (L*) and two chrominance channels (u* and v*). The separation also gives us the ability to put different weights on luminance and chrominance in distance measurements. This is an important option that users can choose according to the characteristics of given video shots. The color difference is thus given by the weighted Euclidean distance in the three dimensional space, i.e.

$$\Delta s = \sqrt{w_L (\Delta L^*)^2 + w_u (\Delta u^*)^2 + w_v (\Delta v^*)^2} \tag{3.5}$$

where w_L, w_u and w_v are weights given by users. Our experiments show that it generally generates better results to put higher weights on chrominance channels (e.g., two times higher than that of the luminance channel).

Then the L*u*v* image is smoothed to remove possible noise, as well as tiny detail, for the purpose of region merging. This is done by an adaptive quantization and median filtering process. Quantization is performed to produce homogeneous regions in which pixels having similar colors are mapped into a single bin. To quantize an image to a limited number of colors(e.g., 64 or 32 bins), the common fixed-level quantization method cannot always preserve color information under variant situations. Thus, we use a clustering-based (e.g., K-Means) method to analyze the input image and determine an adaptive quantizor in the L*u*v* space. The above weighted Euclidean distance (Eq. 3.4) is used as the distance measure in the clustering process. After quantization, a median filtering process is applied on each of the L*u*v* channels. The non-linear

median filtering eliminates image noises and emphasizes prominent color regions while preserving salient image features like edges. It produces the final color map.

The edge map is a binary mask where edge pixels are set to 1 and non-edge-pixels are set to 0. It is generated by applying the CANNY edge detection algorithm. The CANNY edge detector performs 2-D Gaussian pre-smoothing on the image and then takes directional derivatives in the horizontal and vertical directions. These derivatives are used to calculate the gradient. Local gradient maximums are defined as candidate edge pixels. This output is run through a two-level threshold synthesis process to produce the final edge map. A simple algorithm is developed to automatically choose the two threshold levels in the synthesis process based on the histogram of the edge gradient.

The motion field is generated by a hierarchical block-matching algorithm [13]. This algorithm uses distinct sizes of measurement windows for block matching at different grid resolutions, and thus is able to generate a homogeneous motion field that is closer to the true motion than the usual block-based estimation algorithms. We adopted a 3-level hierarchy as suggested in [13].

3.2.3 Region Segmentation and Tracking

For the first frame in the sequence, the system will go directly to intra-frame segmentation. For intermediate frames, as a group of regions with image features and motion information are available from frame n-1, an interframe projection algorithm is used to track these regions. Conceptually, all existing regions at frame n-1 are first projected into frame n according to their motion estimations (i.e. affine parameters). For every pixel in frame n, if it is covered by a projected region and the color difference

(weighted Euclidean distance in L*u*v* space) between the pixel and the mean color of the region is under a given threshold, it is labelled as the same region. When there are more than one regions satisfy the above condition, the one with the smallest color difference will be chosen. If no region satisfies the condition, the pixel will remain un-labeled at this point.

The tracked regions together with un-labeled pixels (for the first frame, all pixels are un-labeled) are further processed by an intraframe segmentation algorithm. New uncovered regions will be detected in this process. An iterative clustering algorithm is adopted: two adjoining regions with the smallest color distance are continuously merged until the difference is larger than the given threshold.

In the following parts, we will discuss the segmentation process first, and then describe the region projection and tracking in detail.

A. Segmentation

First, a color-based pixel labeling process is applied to the color map. Labeling is a process where one label is assigned to a group of neighboring pixels with the same (or very similar) color. The labeling process being used here is standard at the starting frame, but different from the standard method for rest frames. We will discuss this "inter-frame" labeling process in the sub-section B after we explain the region projection process.

To prevent assigning one label to two regions with the same color but separated by edge pixels, only non-edge pixels (i.e. pixels that are set to 0 in the edge mask) are labeled in the process. Edge pixels remain un-labeled. This process generates an initial group of regions (i.e. pixels with the same label) as well as their connection graph. Two

regions are linked as neighbors if pixels in one region have neighboring pixels (4-connection) in another region. As edge-pixels are not labeled, two regions separated by edge pixels are not linked as neighboring regions.

The color merging is an interactive spatial-constrained color clustering process (**Figure 3.6**). Color distances (Eq. 3.5) between every two connected regions are computed. Two connected regions (e.g., i and j) are merged if the color distance between them is (1) smaller than the given color threshold; and (2) the local minimal, (i.e. it is smaller than all the other distances between these two regions and their neighbors). Once a new region is generated from two adjoining regions, its mean color m is computed by taking weighted average of the mean colors of the two old regions (m_1, m_2),

$$m^c = \frac{m_1^c s_1 + m_2^c s_2}{s_1 + s_2}, \quad \text{where } c \in \{L^*, u^*, v^*\} \tag{3.6}$$

where sizes of the two old regions are used as weights. The region connections are also updated for all neighbors of the two old regions. The new region takes the label of the larger one of the two merged regions. Then the two old regions are dropped.

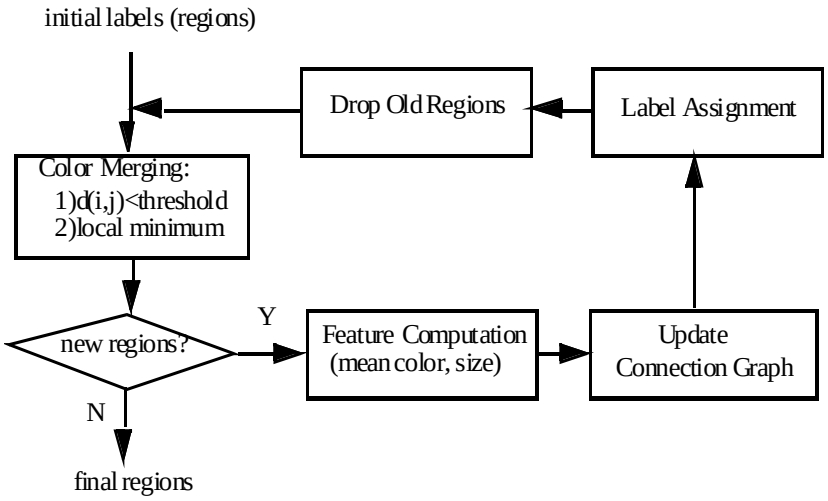


Figure 3.6 Region segmentation and projection at frame n

The merging is iterated until color distances between every two connected regions are above the color threshold. As edge pixels are not labeled, two regions separated by edge pixels are not connected as neighbors. Thus, the growth of each region is naturally stopped at its edge pixels. In the meanwhile, short or weak edges which are usually inside one color region will not affect the region merging process. After the color merging process, edge pixels are simply assigned to their neighboring regions with the smallest color distances.

To ensure homogeneous motion, an optional motion-based segmentation using dense optical flow is applied to the segmented color regions to check the uniformity of the motion distribution. An iterative spatial constrained clustering process similar to that discussed above is used to group pixels inside a color region according to their motion vectors and the given motion threshold.

Finally, tiny regions (e.g., for CIF-size images or regions with less than 50 pixels) are merged with their neighboring regions based again on the color distance. This is mainly to simplify the segmentation result and to reduce the computation complexity.

B. Tracking

A linear regression algorithm is used to estimate the affine motion model (i.e., 8-patameter egomotion model) for each segmented region based on the optical flow computed from frame *n-1* and frame n. As shown in **Figure 3.7**, all segmented regions from the previous frame *n-1* are first projected (virtually) onto the current frame *n* using their individual affine motion models. Projected regions keep their labels in the former frames. The projected regions may not cover the full frame, and holes are potential new

regions.

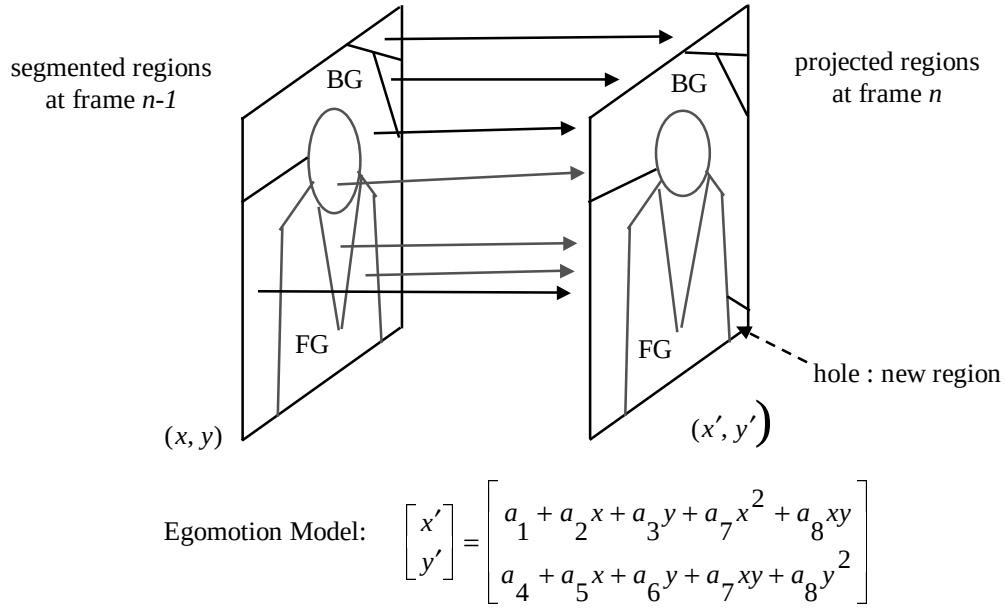


Figure 3.7 Affine model based region projection

As we mentioned before, an inter-frame labeling process is developed to obtain initial segments at frame *n* (**Figure 3.8**).

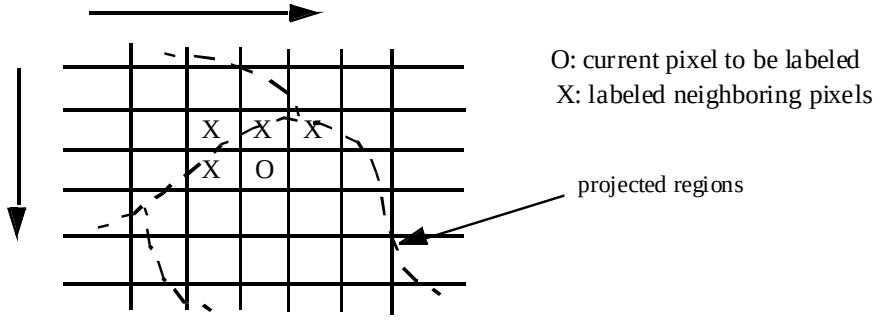


Figure 3.8 Illustration of the inter-frame labeling process

First, in the inter-frame labeling process (**Figure 3.8**), non-edge pixels are walked through and labeled one by one from left to right and top to down. As in the common labeling process, if a pixel has the same color as that of its labeled neighboring pixels, it is assigned the label as its neighbors. When the color of a pixel is different from the colors of its labeled neighboring pixels, its color is compared with all projected regions

that cover its coordinate at the current frame. If its color distance to the closest region is below the given color threshold, this pixel is "tracked", and assigned the label of the closest projected region. Otherwise, a new label is generated and assigned to the pixel.

3.2.4 Post Merging Process

The above segmentation method sometimes generates over-segmented regions. An example is shown in **Figure 3.9**. The brightness is high at the center and gradually goes down to borders. As color is the feature used in the region growing process, and region merging is based on averaged colors within each region, a region with gradual color changes may be broken into multiple regions when accumulated changes are above the given threshold.

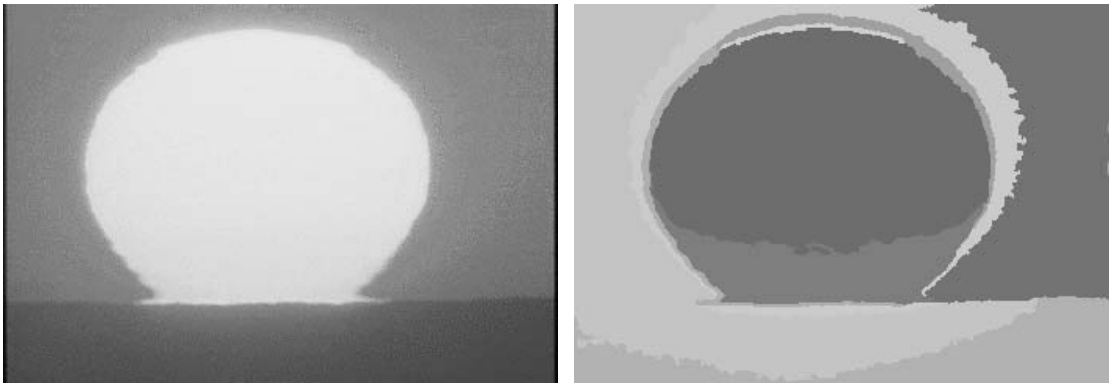


Figure 3.9 Sun and background sky are divided into multiple regions due to gradual color changes

This problem can be solved in some different ways. For example, we can change the difference measure in the merging process from using average color to using average color around region boundaries. Here we present a post processing method based on the edge map that we already extracted. Because this problem is caused by gradual color changes, we can assume there is no edge between these wrongly segmented regions. Therefore, by checking the edge pixels around region boundaries, we can merge two

neighboring regions that are not divided by enough edges.

Assume that region *i* and *j* both exist in N successive frames. Note that each region has multiple instances at a number of successive frames. $B(i^k, j^k)$ is the number of neighboring pixels between region *i* and *j*. $E(i^k, j^k)$ is the number of edge pixels along the border between region *i* and *j*. The edge distance between region *i* and *j* is define as

$$C_k(i,j) = \begin{cases} \dfrac{E(i^k, j^k)}{B(i^k, j^k)} & if\ B(i^k, j^k) > 0 \\ 0 & if\ B(i^k, j^k) = 0 \end{cases}, \quad \text{where } k=1,\ldots,N \tag{3.7}$$

Here $B(i^k, j^k) = 0$ means region *i* and *j* are not adjacent at frame *k*. Given a threshold *T*, local decisions, $D_k(i,j)$'s, are made at each frame to decide if two instances should be merged. $D_k(i,j) = 1$ means region *i* and *j* can be merged at frame *k*.

$$D_k(i,j) = \begin{cases} 1 & if\ C_k(i,j) \geq T \\ 0 & if\ C_k(i,j) < T \end{cases}, \text{ where } k=1,\ldots,N \tag{3.8}$$

The final global decision is made combining local decisions at individual instances. Region *i and j* are merged if

$$\frac{\sum_{k=1}^{N} D_k(i,j)}{N} > P \tag{3.9}$$

where *P* is the percentage of majority required for a merging (a typical value is 50%). Note that when two regions are merged, their instances are merged at all corresponding frames.

Here we discussed an edge-based merging method. Other visual features such as texture and motion can also be used in this post processing process, and the process is not limited to merging only. Splitting can also be an option when the automatic segmentation

process results in too few regions (e.g., only one region). We will study a motion based grouping method in the next chapter for automatic moving object detection.

3.2.5 Salient Video Region Selection

The purpose of our automatic region segmentation and tracking system is to build a feature library for content-based video retrieval. What we want to extract are dominant regions that are able to represent the visual and motion information of semantic video objects. For this reason, we do not want to include small or short regions that are usually caused by noises or segmentation errors, and are not interesting to end users. Furthermore, to include these random behaved small regions in the index will cause many false returns to users' queries.

To detect semantic video objects is certainly the best way to solve the above problem. However, with our current technology, it is not realistic to find semantic objects in general video sources, where there is no domain knowledge or object model. When object motion exists, we can use motion information to find a moving object. As we will discuss in the next chapter, this is also a complex and computation expensive process. Here for the purpose to process large amount of general videos, we adopt a simple solution by computing the volume of each video region.

Assume for a video region *i*, its duration is N_i, its size at each instance is S_i^k where *k=1,…,N*. The volume of region *i* is then $V_i = \sum_{k=1}^{N} S_i^k$. After computing volumes of all video regions, we start to remove regions with the smallest volumes until :

1) the smallest volume is larger than a threshold,

or

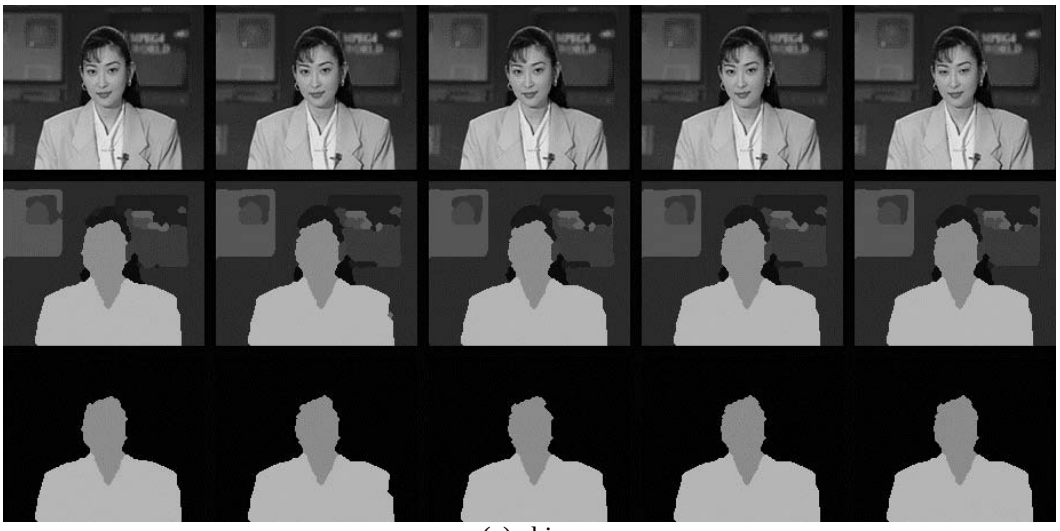
2) the number of remaining region in a video shot is below a given value (e.g., 10).

3.2.6 Experiment Results

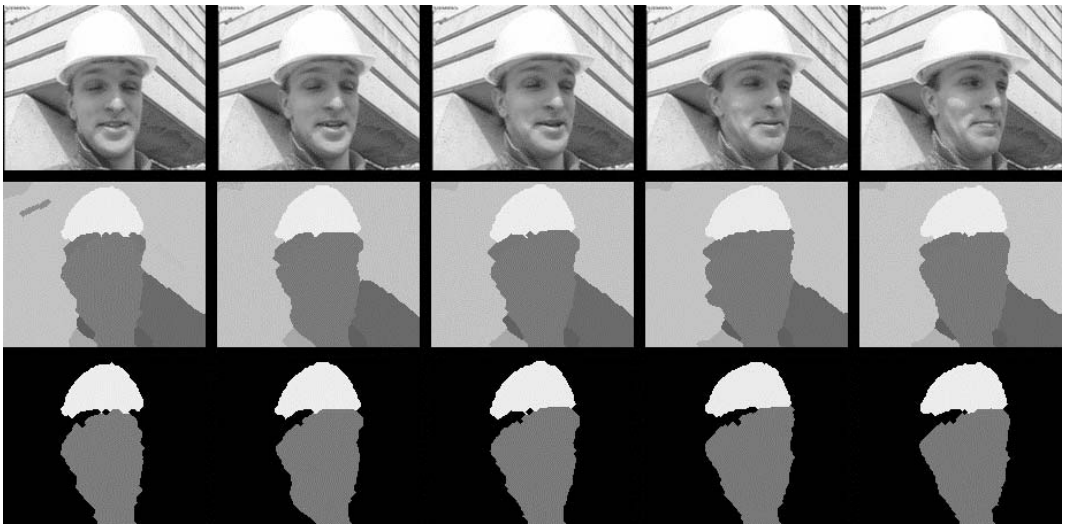
The region segmentation and tracking results of three testing sequences are given in **Figure 3.10**. In each case, the top row shows original sequence. The second row shows all extracted video regions. The bottom row shows a selected subset of automatically segmented regions being tracked. Regions are shown with their representative (i.e. average) colors. The first sequence (akiyo) contains an anchorperson with small motion and strong color difference with background. The second sequence (foreman) contains an object with relatively large motion and very similar brightness to background. In the third sequence, a baseball player is running in the field. The scene contains fast object and camera motion. In all three sequences, major regions are well segmented and tracked through the whole sequence (40 frames are shown). Also the segmented regions have very accurate boundaries.

The segmentation system was used to segment about 200 video shots, covering variant kinds of contents, such as sports, nature, science and history. Experiments show that our method is robust for the tracking of salient color regions through video shots under different circumstances, such as multiple objects, fast/slow motion and region covering/uncovering.

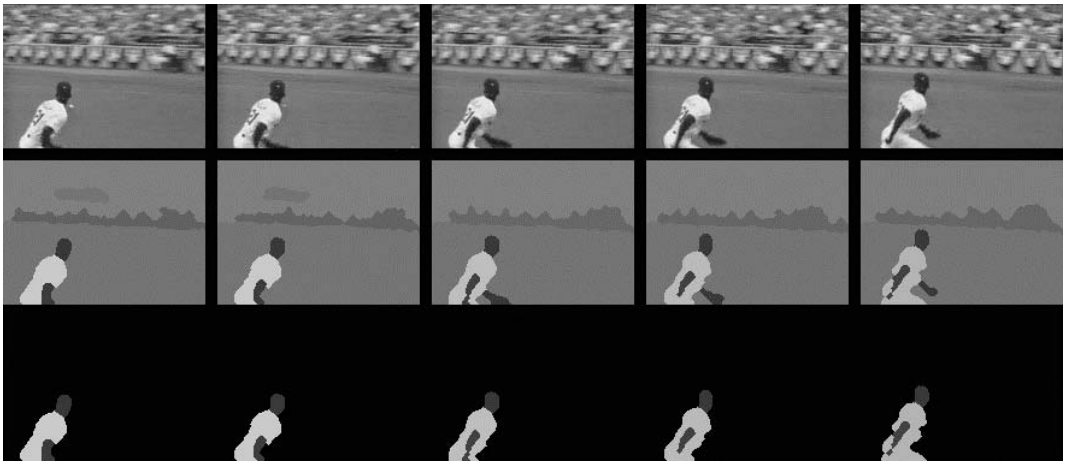
We have noticed that the key to track a region through a long sequence is the inter-frame segmentation process using on region projection. It naturally links regions segmented between two successive frames. On the contrary, if segmentations are fulfilled



(a) akiyo



(b) foreman



(c) baseball player

Figure 3.10 Region segmentation and tracking results of three testing sequences (at frame 1,10,20,30,40)

in each frame independently, it is not robust to match regions from different frames unless for large regions with slow motions.

The combination of color and edge features allows us to achieve accurate region boundaries. Different from old merge-and-split methods where the edge is applied after color-based region merge, we use a new method to fuse edge map directly in the color merging process. This approach is not sensitive to image noises and allows us to use more aggressive merging thresholds in the region growing process to obtain large and stable regions.

The region segmentation scheme we developed is a computation intensive approach. On a regular Sun UltraSparc 5 workstation, it takes about 2-3 days to process a one-hour video (CIF size, 30 fps). If fast processing speed is desired, down sampling at a certain rate (spatially and/or temporally) reduces the processing time by the same rate. Our experiments show that the segmentation algorithm works well for QCIF size video at 15 fps.

3.3 Build of Feature Library

For the aforementioned more than 200 video shots, we obtained more than 2000 salient video regions. Visual features are automatically extracted from the video regions and stored for subsequent queries.

3.3.1 Observations of Segmentation Results

Some segmentation results of sports videos are shown in **Figure 3.11**. People are the main objects within these videos. These images give us a general idea of what regions are

automatically extracted. The results show that our algorithm can correctly identify salient region such as body and face, while ignoring detailed features like eyes. The region boundaries are accurate, which allows us to define shape features.

Figure 3.11 More region segmentation results shown in random colors

One un-desired characteristic is that one semantic object is usually divided into multiple regions, due to the fact that a semantic object does not have homogenous visual

features. Thus it is important to store spatial positions or spatial structures of video regions, so that users can search an object by specifying a set of regions with certain spatial locations.

Background regions also raise a problem. They are usually large regions that cannot be easily removed as noise regions. If background regions and object regions are stored together without being distinguished, users will not always be able to form an effective query. This is because certain features that are significant to objects, such as shape and motion trajectory, may be meaningless to background regions. For a query including these features, if it is matched against background regions, un-predictable results will be returned. Therefore, it is critical to distinct between background and object regions, and to search them differently in the query process.

3.3.2 Visual Feature Extraction

We compute and store the following visual features for each video region: color, texture, shape, spatial location and motion trajectory. In the experiments, we used the mean (i.e., representative) color and the Tamura texture. For shape features, we use global descriptors such as normalized area, aspect ratio, circularity and orientation. 2D-strings are used to capture the spatial arrangement of feature regions. Motion trajectory is stored as a list of a region's coordinates at successive frames.

As we discussed before, we need to distinct background and object region for features like shape, spatial location and motion trajectory. To obtain the trajectory of a region, we again need to compensate camera motions from background regions. Here we use affine motion models to identify background regions.

We first estimate the global affine model A_0 from optical flow around frame borders. For each region, its motion compensation error using A_0 is computed. If the error is smaller than a threshold, the region is declared as a background region. Using optical flow in all background regions detected in the previous step, a new affine model A_1 is computed. Then all regions are classified again according to their motion compensation error with A_1. This process is iterated until the global affine model converges.

The above iterative process only provides a local decision on one instance of a video region. A majority counting procedure similar to the one described in 3.2.4 is utilized to decide if a region belongs to background or objects. While local decisions at individual frames are sensitive to motion estimation errors, global decision is quite robust at shot level where a region is tracked.

3.4 Spatial-Temporal Search of Video Content

We will first discuss distance metrics used in matching different visual features, and then present the search process. Finally, experimental results and analysis are provided.

3.4.1 Feature Matching Metrics

The distance metrics we employed for each feature is discussed as follows.

Color - The color of a query region is matched with the mean color of a target region stored in the database using weighted Euclidean distance as follows:

$$D_C = \sqrt{(L_q - L_t)^2 + 4*(u_q - u_t)^2 + 4*(v_q - v_t)^2} \tag{3.10}$$

where *q* and *t* refer to the query and target respectively.

Texture – The Euclidean distance weighted along each dimension with corresponding variance is defined as follows:

$$D_T = \sqrt{\frac{(\alpha_q - \alpha_t)^2}{\sigma_\alpha^2} + \frac{(\beta_q - \beta_t)^2}{\sigma_\beta^2} + \frac{(\phi_q - \phi_t)^2}{\sigma_\phi^2}} \tag{3.11}$$

where α, β and ϕ refers to coarseness, contrast and orientation respectively. σ_α^2, σ_β^2 and σ_ϕ^2 are the variances of corresponding features.

Aspect Ratio – The L1-distance is used.

$$D_R = \left| R_q - R_t \right| \tag{3.11}$$

Size – The distance is defined as a ratio between the sizes of query and target regions:

$$D_A = 1 - \frac{\min(A_q, A_t)}{\max(A_q, A_t)} \tag{3.12}$$

where A_q and A_t are normalized areas.

Trajectory – We compare two trajectories by uniformly sampling them at a fixed rate, and then computing the average of Euclidean distances between each pair of sampling positions.

$$D_M = \frac{1}{N} \sum_{i=1}^{N} \sqrt{(x_q^i - x_t^i)^2 + (y_q^i - y_t^i)^2} \tag{3.13}$$

where N is the number of samples; (x_q, y_q) and (x_t, y_t) are coordinates along query and target trajectories; and i's are the sampling frames. Note that we divide the duration of trajectories into three categories, i.e., long, medium and short. Users need to specify the length of a query trajectory so it can be aligned with target trajectories. To provide partial matching, we resort to the dynamic programming approach proposed in [62]. It provides a fast searching method to match a short trajectory against a longer one.

3.4.2 Video Query Model

When a query only contains one region, the search process is to compute a total distance of all the selected features and sort the total distances in ascend order. When users provide multiple regions in a query, we use the same filtering and validation process that is used in [99]. Each query region is searched independently first, and then results of individual searches are "intersected" to produce the candidate video list, from which spatial-temporal relationships among video objects are verified based on the above 2D-string based techniques [23].

Generalization of 2D-strings to perform image region similarity matching has been successfully realized in the VisualSEEk system [99]. It provides a framework for searching for and comparing images by the spatial arrangement of automatically segmented feature regions. Multiple regions can be queried either by their absolute or relative locations. Its query strategy can be extended to spatial-temporal query of video objects. Simply a sequence of 2D-strings can be used to represent significant changes of spatial structures in a video shot. 2D-string based query can also be extended to 3D-strings. Video objects maybe projected to x, y, and time dimensions to index their absolute central position, 3-dimensional support, and relative relationships. More sophisticated variations of 3D strings can be used to handle complex relationships such as adjacent, contain, overlap. Combination of 2D strings and trajectory matching is another possibility to explore spatial-temporal structures.

3.4.3 Experiment Results

In our experimental setup, there are 200 video shots, covering variant kinds of contents, such as sports, nature, science and history. By applying the object segmentation and tracking algorithm to the video shots, more than 2000 salient video regions are generated and stored in our server. We build a web-based video search interface called VideoQ, which allows users to form visual queries by drawing feature regions. The interface of VideoQ system is shown in **Figure 3.12**.

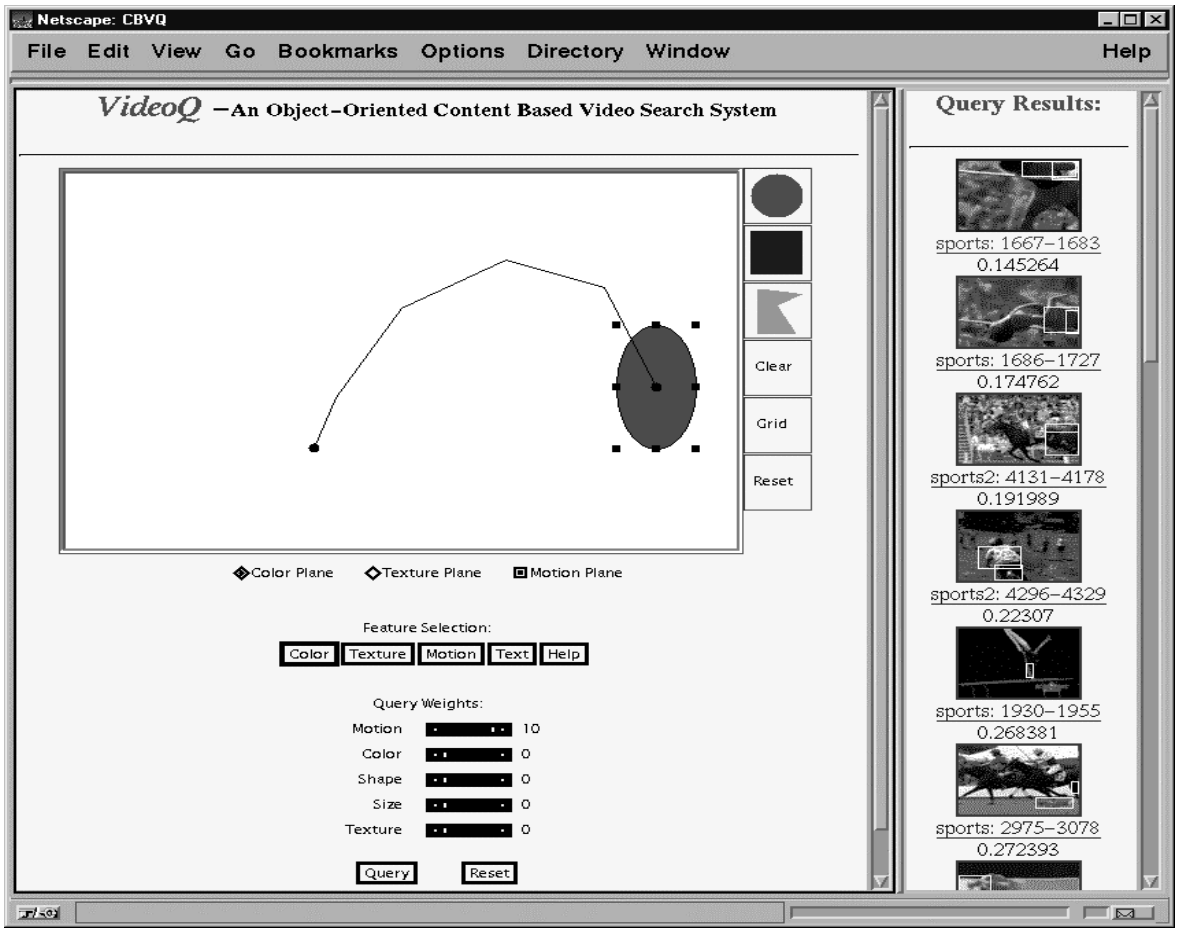


Figure 3.12 The web interface of VideoQ

When a query is submitted to the server, our searching engine will find video shots with similar objects and spatial-temporal structures in the database, and return the icons

of these shots to users. Users may then click on any icon to play a corresponding video shot.

To evaluate the system, precision-recall metrics are computed based on some sample queries. Before each sample query, we establish a ground truth by choosing a set of relevant video shots from the database.

$$\text{Recall} = \frac{\textit{relevant returns}}{\textit{all relevant in database}} \tag{3.14}$$

$$\text{Precision} = \frac{\textit{relevant returns}}{\textit{all returns}} \tag{3.15}$$

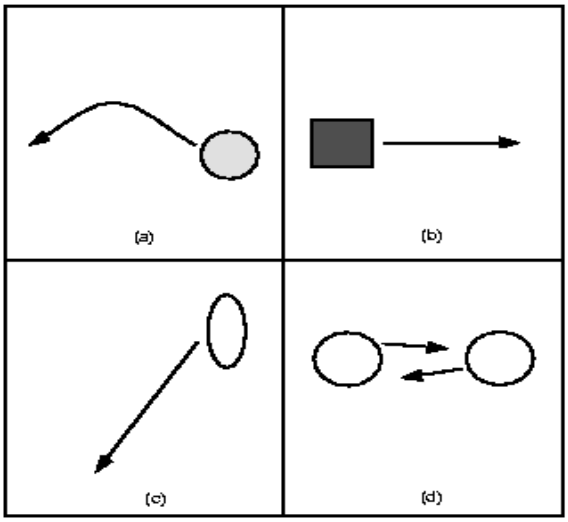


Figure 3.13 Four query examples used in precision-recall experiments

Four sample queries were performed as shown in **Figure 3.13**. Example (a) and (b) highlight motion, color and size. Query (c) uses motion and size. Query (d) highlights motion in addition to size and motion.

The average precision-recall curve is calculated and plotted in **Figure 3.14**. In the experiments, sample queries (a), (c) and (d) performed well, as their motion trajectories are well-defined and not easily confused with camera motions. The query (b) did not

perform well because the global motion compensation has some unavoidable errors and our database contains many videos with camera panning from right to left. As we can see from the curve, to reach a high recall rate, we will have many false alarms. In a practical system, this problem can be alleviated when visual features are used together with text indices or classifications.

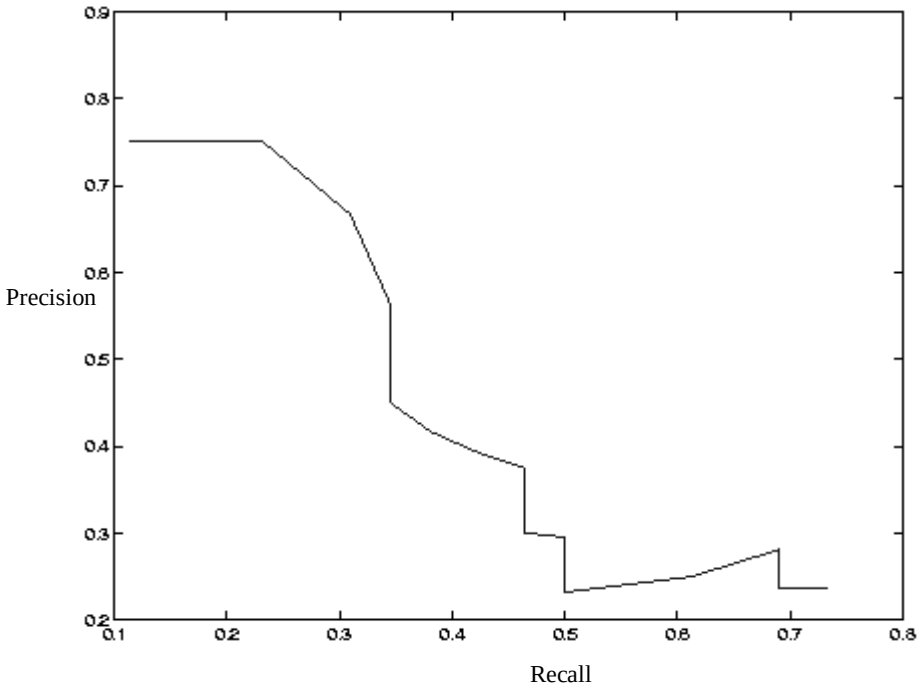


Figure 3.14 The average precision-recall curve over four query

Another test we have performed is to see how many queries does a user need to find a desired video. Twenty target or "desired" video shots are randomly selected, and sample queries are performed. The curve of query number versa query return size is shown in **Figure 3.15**. A great number of queries are needed for small return size. On average for a return size of 14, only two queries are needed to reach a video.

In summary, through VideoQ system, we have shown that our fully automatic region segmentation and feature extraction method can be used to generate visual indices to large amount of video data. It is shown that motion trajectory is the most useful feature in

searching video objects. This again proved that our region-tracking algorithm can follow salient regions stably through a video shot.

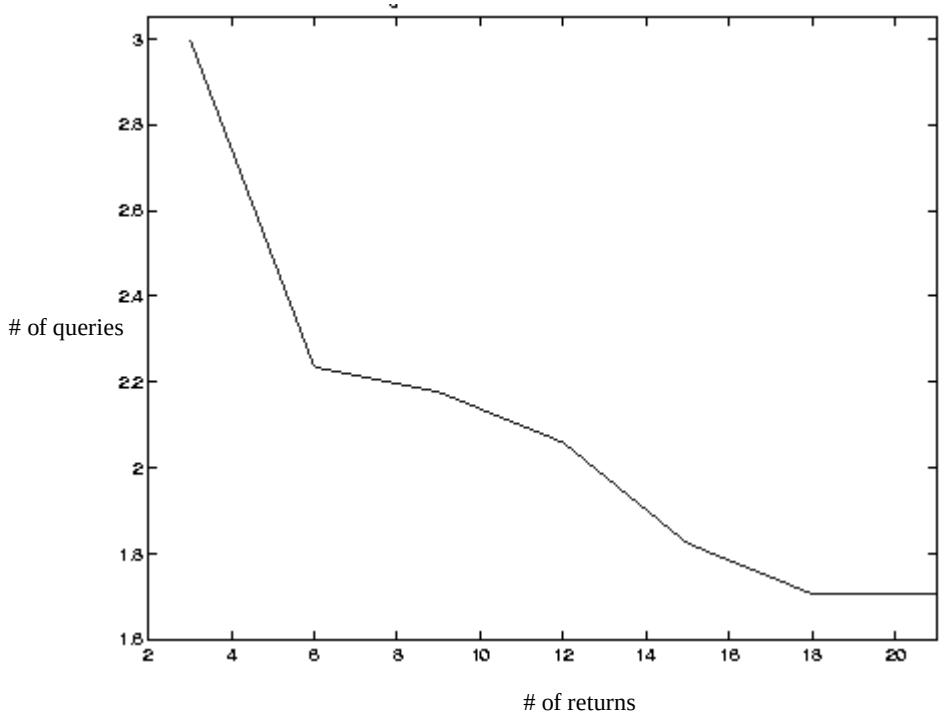


Figure 3.15 Average number of queries needed to reach a video shot (accounting for successful classes only)

We also noticed that most search errors (i.e., false alarms) come from noise regions (e.g., background regions) and global motion compensation errors. The former one introduces many unwanted candidates, and the later one gives us incorrect motion trajectories. This leads us to a high segmentation level for semantic objects. As we will discuss in the next chapter, despite of decades of research, grouping feature regions into objects is still an open issue in computer vision. In general, we need to limit our system to specific domains or sacrifice the degree of automation by asking for user inputs. Based on the region segmentation techniques we have developed, we will show an active system that can help users to effectively identify video objects.

Chapter 4

Semantic Video Object Segmentation and Search

4.1 Introduction

To meet the challenges of future multimedia applications, the newly established MPEG-4 standard has proposed an object-based framework for more efficient multimedia representation. This representation enables unprecedented, flexible access and manipulation functionalities of multimedia content. Similarly, the upcoming MPEG-7 standard, which aims to aims at offering a comprehensive set of audiovisual description tools to enable the needed quality access to multimedia content, will also adopt an object-oriented model to capture events and relations within a scene. In both standards, the production of objects is out of the scopes and is left to content providers and researchers. Thus, the success of object-based media representation and description depends largely on semantic object segmentation.

As we discussed in Chapter 3, image and video object segmentation has been a challenging task over decades. Although much work has been done in decomposing images into regions with uniform features, we are still lacking robust techniques for segmenting semantic video objects in general video sources. In this chapter, we will

address effective semantic object segmentation, representation and search methods in general video sequences. We use "semantic object" to refer to video objects corresponding to meaningful real-world objects, in contrast with lower-level regions, which correspond to image areas with homogeneous features. In the rest of the chapter, we use "semantic object" and "object" interchangeably.

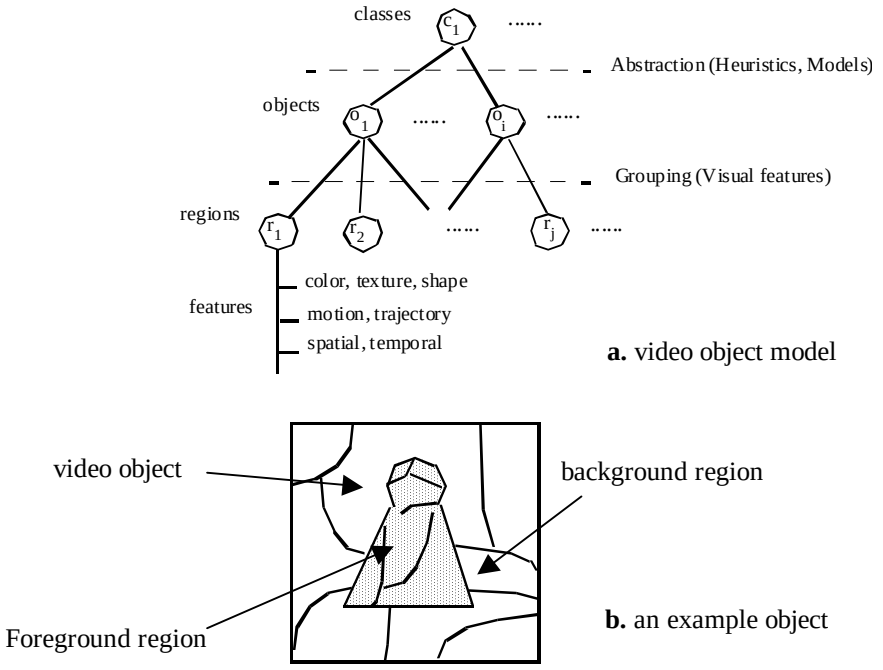


Figure 4.1 Hierarchical representation of video objects

We proposed an integrated approach to address both video object segmentation and content-based retrieval. In this approach, a semantic object is modeled as a set of regions with corresponding spatial and visual features (**Figure 4.1)**. This model directly links the semantic object to its underlying feature regions. For segmentation, the region-based method generates more accurate object boundaries and is also more robust in handling various real-world situations, including complex objects, fast and/or intermittent motion, complicated background, multiple moving objects and partial occlusion. For content-based similarity search, underlying regions provide localized features as well as the

spatial-temporal structure of a video object.

At the bottom level are primitive color regions. These regions are segmented according to color, edge, motion and some other features. Automatic segmentation of homogeneous feature regions has been studied in the last chapter. Various visual features such as color, texture, shape, motion, life span and so on, are then extracted and stored together with these regions for indexing, search and object extraction purposes.

Video objects are extracted at the second level by further grouping primitive regions using different features, such as color, motion vector, spatial or temporal connectivity and long term motion trends, as well as domain models or user inputs. When motion exists, it usually provides a strong indication of entire objects and can be applied to extract moving objects.

The top level includes links to conceptual abstraction of video objects. For example, a group of video objects may be classified to moving human figure by identifying color regions (skin tone), spatial relationships (geometrical symmetry in the human models), and motion pattern of component regions. Usually, accuracy of automatic methods decreases as the level goes up. Fully automatic mapping of video objects to semantic concepts for unconstrained domains is still difficult. In order to solve this problem, several general approaches are taken in recent works. First, some minimal level of user input is used to label example video objects. The system then propagates the user-assigned labels to other video objects in the repository based on visual similarity. The second approach applied unsupervised or supervised clustering of video objects based on their visual features and then tries to map the clusters to subjective concepts [117]. Finally, higher accuracy is achieved by constraining the systems to specific application

domains and thus benefit from the use of domain knowledge (e.g., news video, sports video).

We propose the above hierarchical video object schema for content-based video extraction and indexing. One challenging issue here is to maximize the extent of useful information obtained from automatic image analysis tasks. This general schema can be adapted to different specific domains efficiently and achieve higher performance.

In the following parts, we first present an active system that combines low-level automatic region segmentation with active user inputs for defining and tracking semantic video objects. After discussing the algorithms, extensive experiments and evaluation are presented and analyzed. Then we present another region-based approach for automatic moving object detection. In the last section, we will present a content-based searching system that specifically addresses search and retrieval of semantic objects. We present our work on feature extraction at multiple levels, construction of the visual feature library, and an innovative searching framework for searching video objects.

4.2 Active Video Object Segmentation

Recently, with the demand for object segmentation in general videos and the requirement of more accurate segmentation boundaries, region-based tracking algorithms, which combine common image segmentation techniques with motion estimation methods, have been reported in [33,48,120]. These methods explore more region features and are more robust in handling real-world situations (e.g., occlusion) than those methods using only feature points, segments or boundaries. In [33], Dubuisson and Jain presented an approach to combine motion segmentation using image subtraction with static color

segmentation using the split-and-merge paradigm. Color segmentation regions are examined against motion masks to form the final object mask based on certain confidence measures. In [120], an algorithm is developed to match edge detection and line approximation results with motion segmentation, and to determine the final object boundary. In [48], Gu and Lee proposed a semantic object tracking system using mathematical morphology and perspective motion. Their system uses a modified morphological watershed procedure to segment uncertain areas between the interior and exterior outlines. Flooding seeds are sampled on both interior and exterior outlines. Regions growing from interior seeds define the segmented object boundary. In the first frame, the interior outline is defined by users, and the exterior outline is obtained by dilation of the interior one. For the subsequent frames, the interior and exterior outlines are created by erosion and dilation of the motion-projected boundary from the previous frame.

Satisfactory results from the aforementioned region-based work were reported for certain type of video content, e.g., those with rigid objects and simple motions. However, these techniques usually track a single contour or video object, ignoring complex components and their associated motions within the object. In real-world video sources, an object usually contains several parts with different motions (sometimes non-rigid and with rapid changes). One single motion model is not adequate to track a semantic object. Meanwhile, these techniques still use the motion field as the main feature for tracking purposes. Static color or gray-level segmentation is fulfilled separately, and the fusion of the two segmentation results is done only at the final stage using certain heuristic rules. Due to the noisy nature of the motion field in real-world scenes, tracking results may also

be error prone. As there are no constraints being applied between motion and static segmentation, when the two results are different from each other, it is hard to align them to generate the final object mask. Furthermore, these techniques tend to ignore the background content during the tracking process. This may cause problems in tracking regions near the boundary of the object.

To solve the above problems for real-world video sources, we developed an active system (AMOS) which uses an innovative method for combining low level automatic region segmentation and tracking methods with an active method for defining and tracking video objects at a higher level. A semantic object is modeled as a set of regions with corresponding spatial and visual features. This model directly links the semantic object to its underlying regions. It can handle various real-world situations, including complex objects, fast or intermittent motion, complicated backgrounds, multiple moving objects and partial occlusion.

4.2.1 System Overview

Our initial object definition comes from the user input, i.e. users identify semantic objects in the starting frame by using tracing interfaces (e.g., mouse or optical pen). Given an initial object boundary, the goal of our system is to construct a semantic object model at the starting frame and then track the movement of the object in the subsequent frames.

Generally, a semantic object does not correspond to simple partitions of an image based on one or a few features like color, shape or motion. As the temporal and spatial consistency of one or more visual features is the basic requirement of automatic object

segmentation and tracking methods, it is hard to track a semantic object directly at object level for general video sequences. Decomposing and representing a semantic object in a set of homogeneous feature regions which have temporal and spatial consistency is a necessary step. In our system, the aforementioned hierarchical object model is applied: a semantic object is created and tracked as a set of underlying regions which are homogeneous in terms of color, edge and motion characteristics.

We developed an active system (AMOS) that effectively combines automatic region segmentation methods with the active method for defining and tracking video objects at a higher level (**Figure 4.2**). The system contains two stages: an initial object segmentation stage where user input at the starting frame is used to create a semantic object with underlying homogeneous regions; and an object tracking stage where homogeneous regions and the object are tracked through the successive frames. Note that the segmentation process is applied within individual video shots where there are no scene cuts.

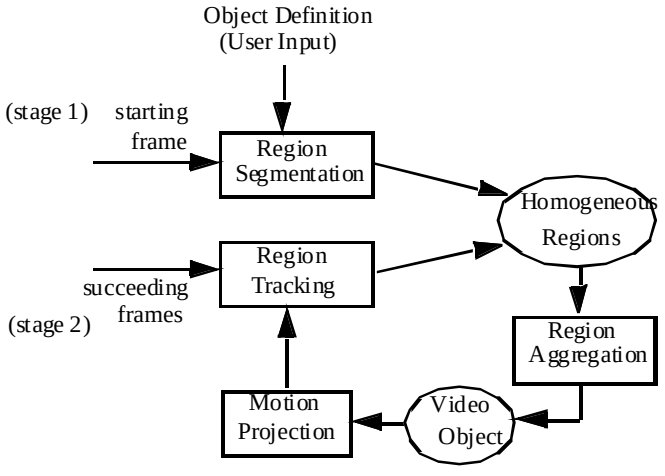


Figure 4.2 The architecture of AMOS system

In the first stage, based on a rough object boundary given by users at the starting frame, the semantic object is generated through a region segmentation and aggregation

process. An innovative region segmentation is applied within the slightly expanded bounding box of the user-specified object. It effectively fuses color and edge features in a region-growing process that produces homogeneous color regions with accurate boundaries. Motion segmentation based on a dense motion field is used to further split the color regions to extract homogeneous regions in both color and motion. Region aggregation is based on the coverage of each region by the given object mask: regions that are covered more than a certain percentage are grouped into the foreground object. The final contour of the semantic object is computed from the aggregated foreground regions. Both foreground region and background regions are stored and are tracked over time in the successive frames.

Tracking at both the region and object levels is the main task in the second stage. As shown in **Figure 4.2**, segmented regions from the previous frame are first projected to the current frame using their individual affine motion models. An expanded bounding box including all projected foreground regions is computed. Then the area inside the bounding box is split to homogeneous color and motion regions following a region tracking process. Unlike existing approaches, projected regions are not used directly as the new segmentation, but as seeds in another color based region growing process to track existing regions. Pixels that cannot be tracked from any old regions are labeled as new regions. Thus the resulting homogeneous regions are tagged either foreground (i.e. tracked from a foreground region), or background (i.e. tracked from a background region), or new. They are then passed to an aggregation process and classified as either foreground or background regions. Instead of the user input, the approximated object boundary is computed from the projected foreground regions.

Our experimental results and extensive performance evaluation have shown very good results. A detailed explanation of the two stages is given in the following two sections.

4.2.2 Object Segmentation in Initial Frame

The system diagram of semantic object segmentation at the starting frame is shown in **Figure 4.3**. It consists of four major processes as follows:

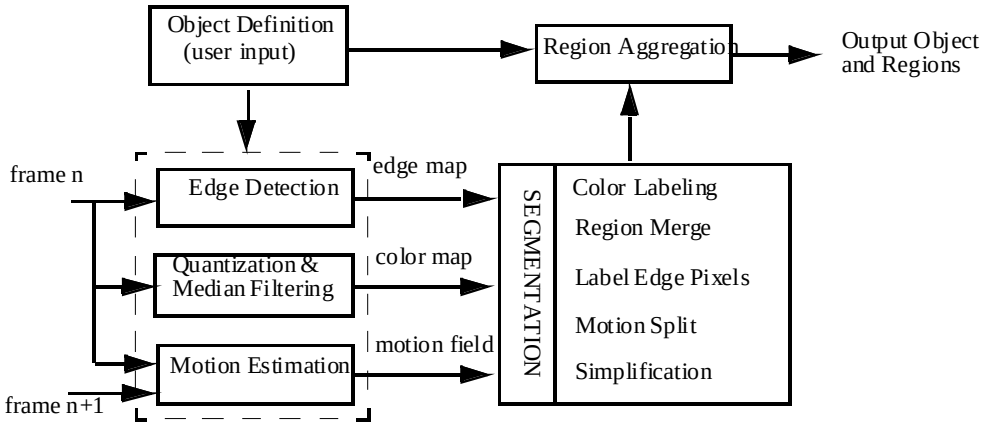


Figure 4.3 Object segmentation in initial frame

1) Object Definition and Threshold Specification

First, the user identifies a semantic object by using tracing interfaces (e.g., mouse). The input is a polygon whose vertices and edges are roughly along the desired object boundary. To correct user-input errors, a snake algorithm [57] is used to align the user-specified polygon to the actual object boundary (i.e. edges). The snake algorithm is based on minimizing an energy function associated with edge pixels. Users may also choose to skip the snake module if the initial outline is satisfactory. The final object boundary will be obtained in the subsequent region segmentation and aggregation process.

After the object definition, users can specify a set of parameters to start the tracking

process. These parameters include a color-merging threshold, weights on the three color channels, a motion merging threshold, and a tracking buffer size. Usually users may just rely on the default values for these parameters. Determination of these parameters will be explained in more detail in the following sections. These parameters can be optimized based on the characteristic of a given video shot and experimental results. For example, for a video shot where foreground objects have similar luminance with background regions, users may put a lower weight on the luminance channel. Users can start the tracking process for a few frames with the default thresholds which are automatically generated by the system, and then adjust the thresholds based on the segmentation and tracking results. The segmentation results are not sensitive to slight changes of parameter values. However, optimization may be used to reduce the computation complexity. Our system also allows a user to stop the tracking process at any frame, modify the object boundary that is being tracked, then restart the tracking process from the modified frame.

Given the initial object boundary from users (or the snake module), an expanded (~15 pixels) bounding box surrounding the arbitrarily shaped object is computed. All the following segmentation procedures are performed inside this bounding box to reduce computation complexity.

2) Generation of Feature Maps

Within the bounding box, the three feature maps, edge map, color map and motion field are created from the original images. We take the same approaches described in 3.2.2.

3) Region Segmentation

The segmentation algorithm is developed based on the three feature maps: color map, edge map and motion field. Departing from old merge-and-split methods where the edge is applied after color-based region merge, we propose a new method to fuse edge information directly in the color merging process. Detailed description is given in the section 3.2.3. In the previous region tracking process, the goal is to track salient regions reliably through an image sequence. Here what we want to robustly track are video objects, and it is not necessary to trace regions over a long period. Therefore, we use a set of relatively low thresholds, which produce more detailed regions, to get better object boundaries.

4) Region Aggregation

The region aggregation module takes homogeneous regions from the segmentation and the initial object boundary from the snake (or user input directly). Aggregation in the starting frame is relatively simple compared with that for the subsequent frames, as all regions are newly generated (not tracked) and the initial outline is usually not far from the real object boundary. A region is classified as foreground if more than a certain percentage (e.g., 90%) of the region is included by the initial object boundary. On the other hand, if less than a certain percentage (e.g., 30%) of a region is covered, it is considered as background. Regions between the low and high thresholds are split into foreground and background regions according to the intersection with the initial object mask.

Finally, affine motion parameters of all individual regions, including both foreground

and background, are estimated using a multivariate linear regression process over the dense optical flow inside each region. In our system, a 2-D affine model with 8 parameters is used. It is a good first-order approximation to a distant object undergoing 3-D translation, rotation and linear deformation.

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} a_1 + a_2 x + a_3 y + a_7 x^2 + a_8 xy \\ a_4 + a_5 x + a_6 y + a_7 xy + a_8 y^2 \end{bmatrix} \tag{4.1}$$

where $[x, y]$ and $[x', y']$ are the original and transformed coordinates respectively, a_1 to a_8 are 8 affine transformation parameters. These affine models will be used to track the regions and object in the future frames, as we will discuss in the next section.

4.2.3 Object Tracking in Successive Frames

Given the object with homogeneous regions constructed in the starting frame, tracking in both the region and object levels is the main task in the successive frames. The objective of the tracking process is to avoid losing foreground regions and also avoid including false background regions. In our system, this task is accomplished through the following steps. First, an inter-frame segmentation process is used to segment a frame into homogeneous regions. Unlike at the starting frame where all regions are tagged as new, these regions are classified in the inter-frame segmentation process as either foreground, or background or new according to their relationship with the existing foreground and background regions in the previous frame. Then, in the region aggregation process, the estimated (projected) object boundary is used to group these tagged regions into the object and background. For regions around the object boundary with the tag "new", their visual features are also examined to decide whether they belong

to the object or not.

1) Region Projection

As shown in **Figure 4.4**, segmented regions from the previous frame, including both foreground and background, are first projected onto the current frame (virtually) using their individual affine motion models. Projected regions keep their labels and original classifications. For video shots with static or homogeneous background (i.e. only one moving object), users can choose not to project background regions to save time. An expanded bounding box of all projected foreground regions is computed. Similar to the process in the first frame, all the following segmentation and aggregation processes are only applied to the area inside the bounding box.

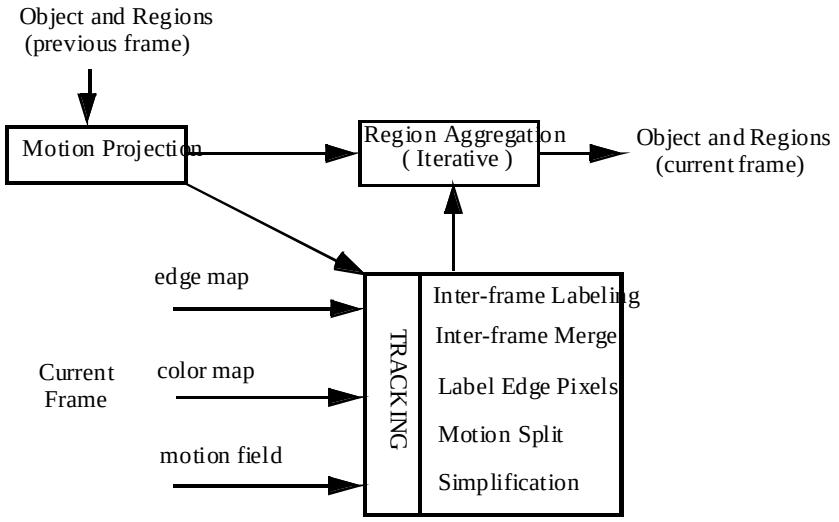


Figure 4.4 Automatic semantic object tracking process

2) Generation of Feature Maps

Generation of the three feature maps (color, edge and motion) utilizes the same methods as we described in the previous section. The only difference is that in the quantization step, the existing color palette computed at the starting frame is directly used

to quantize the current frame. Using a consistent quantization palette enhances the color consistency of segmented regions between successive frames, and thus improves the performance of region tracking. As object tracking is limited to single video shots, in which there is no abrupt scene change, using one color palette is generally valid. Certainly, a new quantization palette can be automatically generated when a large quantization error is encountered.

3) Region Tracking

In the tracking module, based on the projected foreground and background regions, three feature maps are fused to track existing regions and segment the current frame. The inter-frame labeling process is described in section 3.2.3. Note the difference here is that regions are classified as *foreground*, *background* or *new*.

The subsequent color merge, edge labeling, motion split and small region elimination processes are similar to those at the initial frame with some additional constraints. Foreground or background regions tracked from the previous frame are only allowed to be merged with regions of the same class or new regions, but merging between a foreground region and a background region is forbidden. New regions can be merged with each other or merged with foreground/background regions. When a new region is merged with a tracked region, the merged result inherits its label and classification from the tracked region. In motion segmentation, split regions remain in their original classes. After this inter-frame tracking process, we obtain a list of regions temporarily tagged as either foreground, background, or new. They are then passed to an iterative region aggregation process.

4) Region Aggregation and Object Composition

As shown in **Figure 4.5**, region aggregation module includes two inputs: the homogeneous region and the estimated object boundary. The object boundary is estimated from projected foreground regions. Foreground regions from the previous frame are projected independently of one another and the combination of projected regions forms the mask of the estimated object. The mask is refined with a morphological closing operation (i.e. dilation followed by erosion) with a size of several pixels in order to close tiny holes and smooth boundaries. To tolerate motion estimation errors, which are common for general video sources, the resulting mask is further dilated for the tracking buffer size, which is specified by users at the beginning of tracking. Generally a larger buffer size is required for objects with fast motion or abrupt shape change.

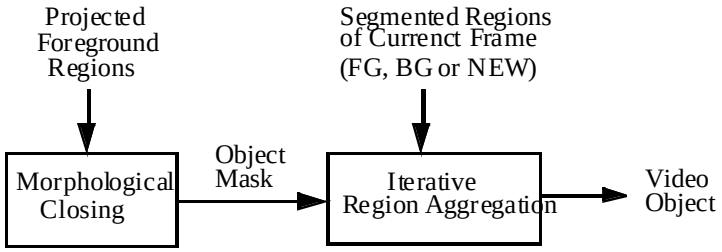


Figure 4.5 Region aggregation using projected objects

The region aggregation module implements an iterative region grouping and boundary alignment algorithm based on some distance measures between a region and the projected object mask. Currently we defined three distance metrics as follows.

$$D_1(r) = \frac{\left| r \cap Obj^{Project} \right|}{|r|} \tag{4.2}$$

$$D_2(r1, r2) = Edge(r1, r2) \tag{4.3}$$

$$D_3(r1, r2) = \left\| MV^{r1} - MV^{r2} \right\| \tag{4.4}$$

where $D_1(r)$ is the intersection ratio of region *r* with the projected object mask $Obj^{\Pr oject}$. $D_2(r1, r2)$ is the number of edge pixels between region *r1* and *r2*. $D_3(r1, r2)$ is the Euclidean distance between the mean motion vectors MV^{r1} and MV^{r2} of region *r1* and *r2* respectively.

The basic algorithm is described as follows. For every segmented region, if the region is tagged as background, keep it as background without checking any distance measures. If it is a foreground or new region, we compute the intersection ratio $D_1(r)$. If a foreground region is covered by the object mask by more than certain percentage (e.g., 80%), it is kept as foreground; otherwise, it is intersected with the object mask and split into one foreground region and one new region.

For a new region, if its D_1 is larger than certain percentage (e.g.,80%), it is grouped into the object as foreground; if the intersection ratio is very small (e.g., less than 30%), it is kept as new. Otherwise, visual similarity (including D_2 and D_3) between this region and its neighbors is examined. The region is grouped into the object as foreground if the region is separated from background regions by more edge pixels than foreground regions (or this region is not connected to any background regions), and its closest neighbor according to the motion feature (e.g., mean motion vector) is a foreground region.

The above aggregation and boundary alignment process is iterated multiple times (e.g., 2 or 3) to handle possible motion projection errors, especially for fast motion. Also

we use a relatively lower ratio (80%) here to include a foreground or new region. At the end of the last iteration, all remaining new regions are classified into background regions.

Finally, affine models of all regions, including both foreground and background, are estimated by a linear regression process over the optical flow. As described before, these affine models are used to project regions onto the future frame in the motion projection module.

4.2.4. Segmentation Results and Performance Evaluation

The AMOS system has shown very good segmentation and tracking results on general video sources. As shown in **Figure 4.6**, five different types of video sequences are used to do subjective and objective evaluation of our system. The first sequence (akiyo) contains an anchorperson with small motion and strong color difference with background. The second sequence (foreman) contains an object with relatively large motion and very similar brightness to background. In the third sequence, the skater has many different fast motion parts (i.e. body, arms and legs). The characteristic of the fourth sequence is that the shape of the flying bird changes abruptly from frame to frame. In the last sequence, a plane is flying away from the camera. 100 successive frames are used for each sequence in the experiments. The last three sequences are decoded from MPEG-1 videos.

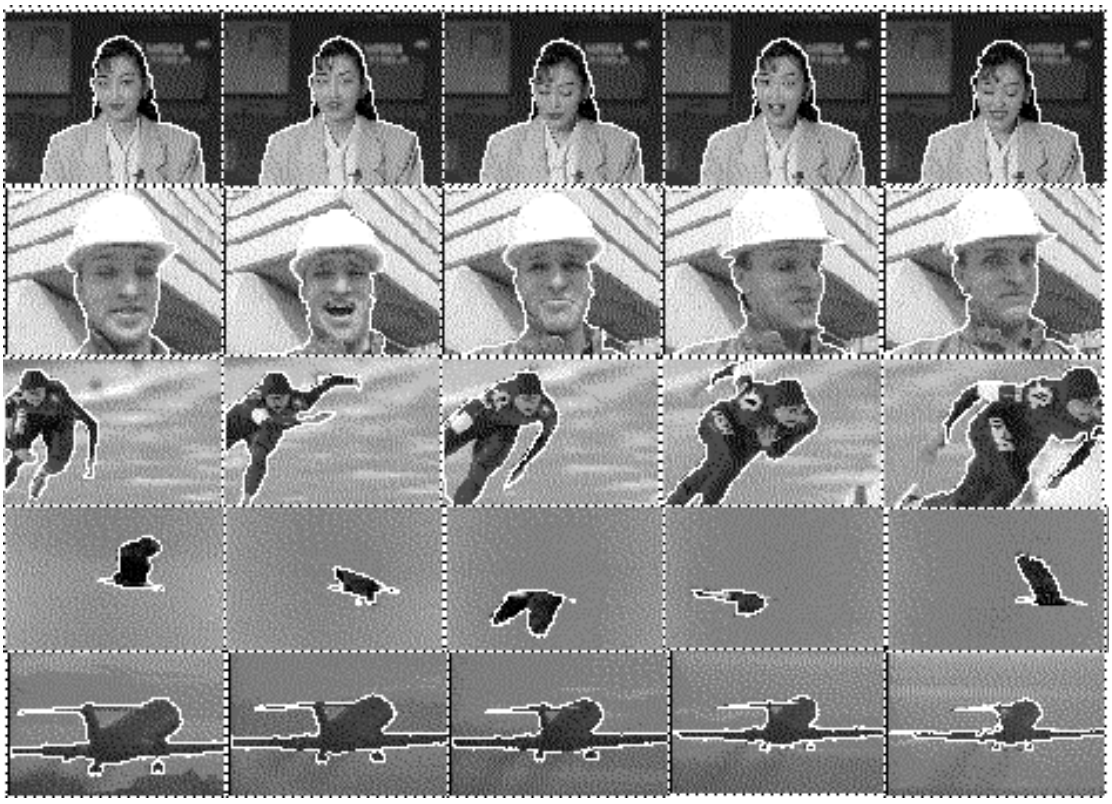


Figure 4.6 Object tracking results of five sequences after three user inputs
(from left to right, frame #1, 25, 50, 75, 100)

After the first user input at the starting frame, more user inputs are applied at subsequent frames where the largest tracking errors occur or start to occur to refine the tracking results. Notice that the effort required to correct boundary errors (i.e. subsequent inputs) is much less than the initial object definition. As we will discuss below, major tracking errors come from uncovered background or foreground regions, as such errors usually propagate to the subsequent frames **Figure 4.6** shows object tracking results of the five testing sequence after 3 user inputs, which gave us acceptable results for all five sequences. For subjective evaluation, segmented objects are superimposed onto gray background with random noise and played in real time for users to see whether there are observable errors. To remove boundary jitter (i.e. high frequency noise), a temporal median filtering process (with a radius of 2 frames) is applied to the binary object masks

before they are composed with the background with random noise. For the akiyo sequence, there are no noticeable errors after only one input at the starting frame. For the foreman, bird and plane sequences, three user inputs give us outputs without noticeable errors. The skater sequence, which has fast and complex motion, requires 4 user inputs to remove noticeable errors. These subjective evaluations are confirmed and further explained in detail in the following objective evaluation experiments (**Figures 4.7 –4.10**).

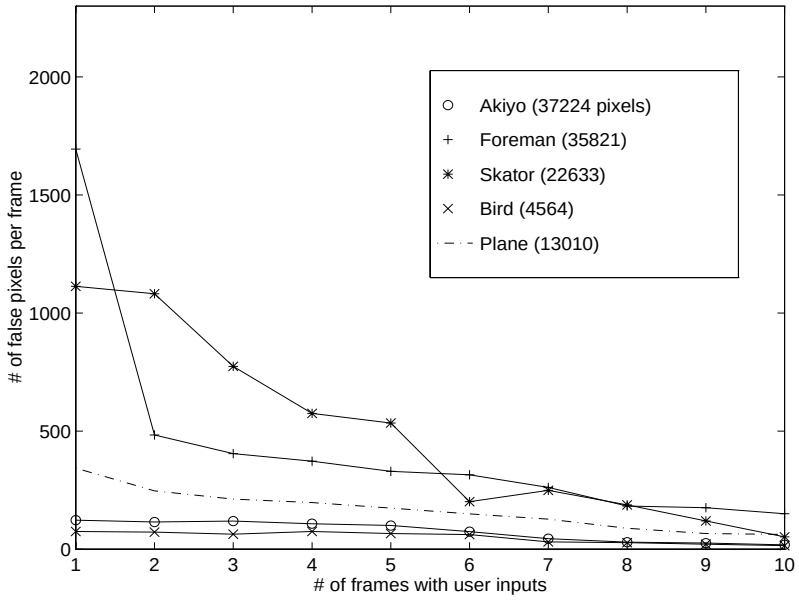


Figure 4.7 Average number of false pixels over 100 frames (numbers in the legends are sizes)

In the objective evaluation, we manually extracted semantic objects in each frame over 100 successive frames (with the help of the snake algorithm), and considered these as the ground truth. We then computed the average numbers of missing pixels, false pixels and maximum boundary deviations between the ground truth and the segmentation results. The performance results are shown with different numbers of user inputs in Figures 4.7,4.8 and 4.10 respectively. Here the first user input is the object outlining at the starting frame; and the following inputs are boundary adjustments that users made to correct tracking errors. The legends show the average size (number of pixels) of each

object. We could plot normalized curves for missing and false pixels (e.g., normalize the size or perimeter of an object), but each method has different drawbacks. As we will show, the maximum boundary deviation curves provide size invariant evaluation results.

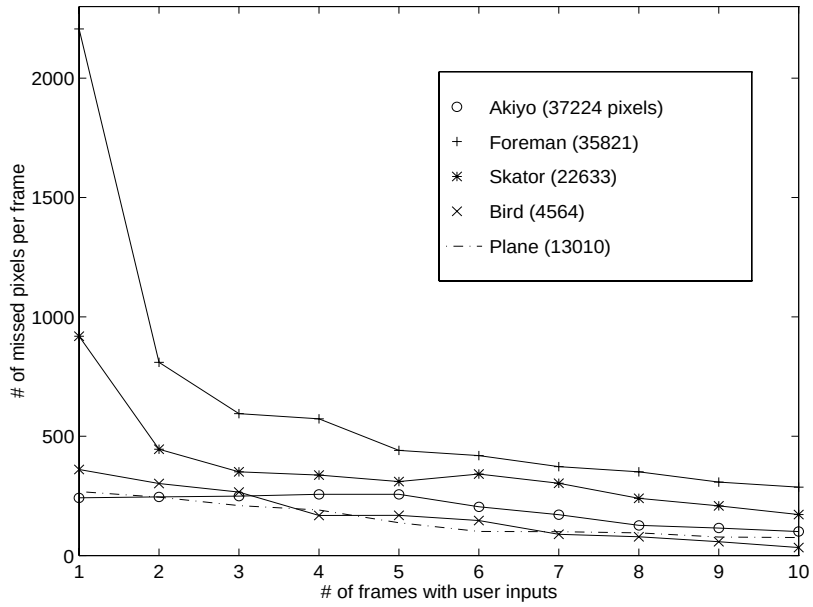


Figure 4.8 Average number of missed pixels over 100 frames (numbers in the legends are sizes)

Numbers of missing and false pixels are simply computed by comparing a segmented object mask with its related ground truth. While there are different ways to define the boundary deviation, we define the deviation for a missing pixel, as the distance between this pixel and its nearest foreground pixel in the segmented object mask; and for a false pixel, as the distance between this pixel and its nearest foreground pixel in the ground truth mask (**Figure 4.9**).

$$dev^{miss}(p) = \min_{q}(\|p-q\|) \quad \text{and} \quad q \in Obj^{seg} \tag{4.5}$$

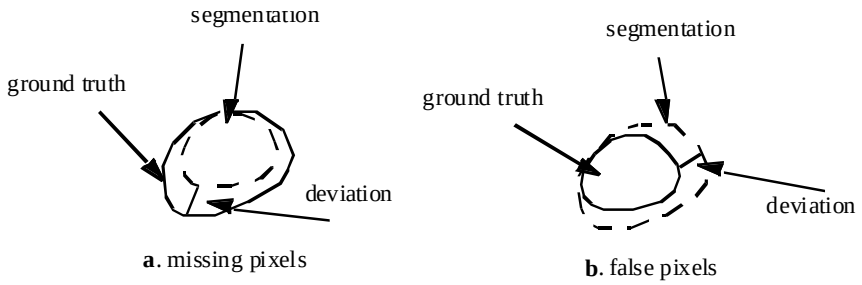
$$dev^{false}(p) = \min_{q}(\|p-q\|) \quad \text{and} \quad q \in Obj^{true} \tag{4.6}$$

where p and q are coordinates of corresponding pixels. Obj^{seg} and Obj^{true} are the segmented object mask and the ground truth mask respectively. The maximum boundary

deviation is the maximum value of the deviations of all missing or false pixels.

$$dev^{\max} = \max(\max_{p \in miss}(dev^{miss}(p)), \max_{p \in false}(dev^{false}(p))) \quad (4.7)$$

where *miss* and *false* refer to the sets of missing and false pixels respectively.



a. missing pixels **b**. false pixels

Figure 4.9 Definition of boundary deviations

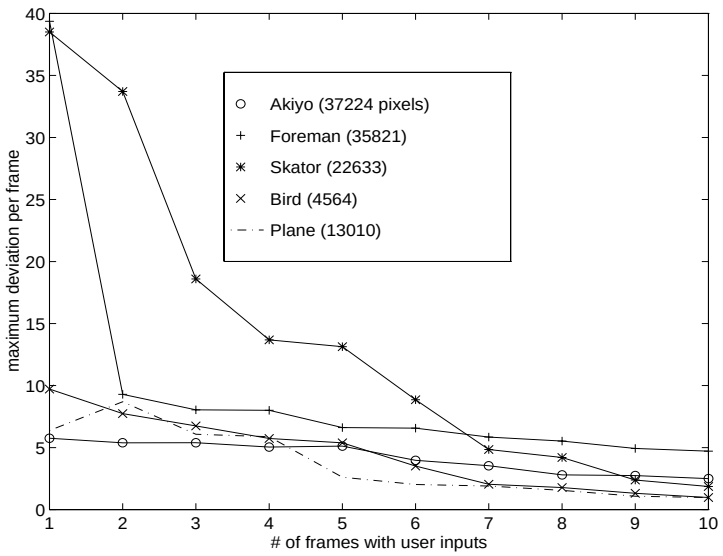


Figure 4.10 Maximum boundary deviations of the five tracked objects (averaged over 100 frames)

Main tracking errors are usually caused by new or uncovered background or foreground regions. This is clearly reflected in **Figure 4.10**, where the maximum deviations are around 40 pixels when a large region is missed or falsely included. As shown in the plots, such errors are corrected after 2 or 3 user inputs. The remaining errors may be considered an accuracy limitation of our segmentation and tracking algorithm. The number of false and missing pixels can be reduced from 30 to 200 pixels

depending on the object size. The maximum deviation curves show that these missing and false pixels are less than 5 pixels away from the ground truth. Considering inherent errors caused by boundary blur (especially for the MPEG-1 sequences with fast motion) and manual segmentation, our system generates very good tracking results.

The Akiyo sequence only requires one user input to obtain very good tracking result over 100 frames. Missing and false pixels are mainly around the hair, which is not clearly separated from background by color or by edge. Additional user inputs bring very small improvement. The bird sequence contains a small, fast-moving object. The abrupt shape change between successive frames causes the missing of part of the wings in some frames. These errors are corrected after 2-3 more user inputs. For the plane sequence, after the first input, most false pixels come from a black strip at the left frame border. Part of the strip is merged into the plane in some successive frames. This error is easily corrected after 1-2 more user inputs.

The foreman and skater sequences have relatively large errors at the first user input, due to complex object motion. Missing pixels are mainly caused by the emerging of new foreground regions around the frame border (e.g., the left shoulder in the foreman sequence). These new regions connect with old object regions only by a few pixels when they first appear and they also abruptly change the shape of object. Thus they are classified as background regions during the tracking process. False pixels are included mainly because of uncovered background regions enclosed by the object. In the skater sequence shown in **Figure 4.6**, frame 50 has a background region included in the object. The background region is uncovered when the two legs move apart. As it is enclosed by the object and the frame border and thus not connected to any exist background regions,

the new background region is falsely classified as foreground. The above two types of errors can be corrected by one user input at the frame where the error starts to happen. Restarting the tracking process after the correction will correct errors in the successive frames. For the foreman and skater sequence, the numbers of missing and false pixels drop rapidly after the first 2 user inputs. And after this, maximum boundary deviations are generally within 10 pixels.

The computational complexity and speed of the AMOS system are not currently our main focus. Generally the segmentation speed depends on the object size and the complexity of the scene. For a typical object like akiyo in CIF size images, it takes around 20 seconds per frame on a SUN UltraSparc-2 workstation. Note this is based on a pure JAVA implementation of all segmentation and tracking processes in the system without speed optimization. Also this includes all the computation processes described in the system architecture. Optimization may be performed to determine critical processes and ignore non-critical processes to reduce computations. Parallel processing can greatly improve the speed of our system. As one can see, the computations of three feature maps, which are the most computation intensive parts of the system, can easily be done in parallel. The main region segmentation and tracking algorithm can also have parallel implementation, as region merging is accomplished by examining local minimums.

In summary, our extensive experiments and evaluation have shown that the AMOS system can be effectively used to segment generic video objects with accurate boundaries. The system fuses the color, edge and motion information in its region segmentation, and utilizes an iterative region aggregation and boundary alignment process to generate and track accurate object boundaries. As a semi-automatic system, it

also allows users to easily correct tracking errors. We have been using this system to extract more than 100 video objects, and building a video object database for many object-based video applications such as MPEG-4 compression, content-based retrieval (which we will discussed in the next part of this paper), and network transmission.

4.3 Automatic Moving Object Segmentation

The AMOS system is a general tool developed for interactive semantic object segmentation. It can be used in offline application where object-based compression and indexing is needed. In the case when real-time processing is required, user inputs are not feasible or very limited. Typical examples include broadcast sports or news programs. If we want to parse and summarize these videos in real time, automatic object extraction methods need to be developed. In chapter 3, we have presented an automatic video region tracking system. Given domain specific models, these regions can be grouped into high-level objects.

In this section, we will look at a rather general constraint, which is the motion of objects. Using this constraint, objects in a scene are classified into foreground and background according to their motion characteristics. Moving objects (after global motion compensation) are defined as foreground objects, which are the targets we want to extract. Static regions (e.g., ground) and objects (e.g., a still car) are both considered as background. In addition, we also assume background regions are more likely around the corners and borders, while foreground regions are likely in the middle of a scene.

4.3.1 Overview

The 2D motion observed in an image sequence is caused by 3D motions of : 1) objects, and 2) camera. When there is no camera motion, we can easily find moving objects by grouping all moving regions together. However, except some special cases (e.g., surveillance videos), common TV programs and home videos usually contain camera motions. In these situations, to detect moving objects, we firstly need to compensate motions caused by camera operations.

As pointed out in [2], the camera induced image motion depends on the ego-motion parameters (i.e., rotation, zoom and translation) of the camera and the depth of each point in the scene. In general, it is an inherently ambiguous problem to estimate all these physical parameters. Existing camera motion detection approaches can be generally divided into two classes: 2D algorithms that assume the scene can be approximated by a flat surface, and 3D algorithms that work well only when significant depth variations are preserved in the scene. It has been noticed that in 2D scenes when the depth variations are not significant, the 3D algorithms are not robust or reliable. On the contrary, 2D algorithms using a 2D global parametric model (e.g., affine model) cannot handle 3D scenes where there are multiple moving layers.

In ordinary videos we see in our daily life (e.g., those used in our experiments), as depth information is not well preserved, 2D algorithms are used more widely than 3D algorithms. When the scene is far from the camera and/or the camera only conducts rotation and zoom, a single affine motion model can be used to model and compensate the camera-induced motion. However, when the scene is close to the camera and the camera is translating, multiple moving planar surfaces may be produced in the image

sequence. In general videos, the above two scenarios can both exist in one video shot with gradual transitions between them. To manage this problem, many approaches have been proposed to use multiple 2D parametric models to capture multiple motion layers.

In [113], affine motion parameters are first estimated from the optical flow by linear regression, and then spatio-temporal segmentation is obtained by a clustering in the affine parametric space. In [16], a dominant motion is first estimated by means of a merge procedure. Then motion vectors that can be well represented by this dominant motion model are identified and excluded, and affine parameters are estimated from remaining blocks. This procedure is repeated until all motion layers are detected. Similar approaches are also reported in [76]. These methods rely only on motion information in grouping image pixels or blocks into motion layers, and thus usually result in inaccurate segmentation on motion boundaries. As there is a strong dependence between motion estimation and layer segmentation, bad segmentation results will reduce the accuracy of motion estimation. Another problem is that the object tracking is not really addressed here. It is assumed that moving objects detected at individual frames automatically form the temporal object track. In real-world scenes, objects and camera usually do not have uniform temporal motion. Captured image sequence will show obvious object motion in some frames, but slight or even no motions in other frames. This introduces inconsistent detection results in individual frames.

To overcome these problems, applying our region segmentation and tracking algorithms proposed in chapter 3, we developed a two-stage moving objects detection method. This method uses regions with accurate boundaries to effectively improve motion estimation results. Furthermore, we explore the temporal constraint in a video

shot to achieve more reliable object detection results. The two general stages of our algorithm is shown in **Figure 4.11**.

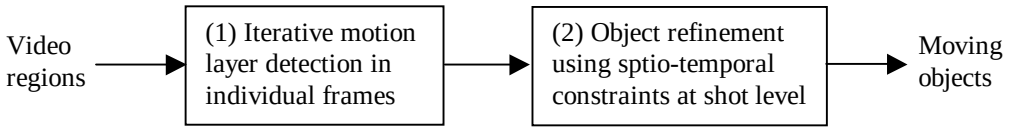


Figure 4.11 Two-stage moving object detection based on region segmentation and tracking results

In the first stage, we apply an iterative motion layer detection process based on the estimation and merging of affine motion models. Each iteration generates one motion layer. The difference with existing methods is that motion models are estimated from spatially segmented color regions instead of just pixels or blocks. In the second stage, temporal constraints are applied to detect moving objects in spatial and temporal space. Layers in individual frames are linked together based on characteristics of their underlying regions. One or more layers will be declared as motion objects according to certain spatio-temporal consistency rules.

4.3.2 Iterative Motion Layer Detection

The iterative layer detection is applied to each individual frame as shown in **Figure 4.12**. First, non-background regions[1] are merged into motion layers according their affine models. Because different regions that belong to the same motion layer may have different estimated parameters due to inaccuracy in the initial dense motion field, a simple clustering approach in the affine parametric space usually does not work well. To solve this problem, we use the following distance measure to compare two neighboring

[1] All regions are non-background regions in the first iteration.

regions R_i and R_j.

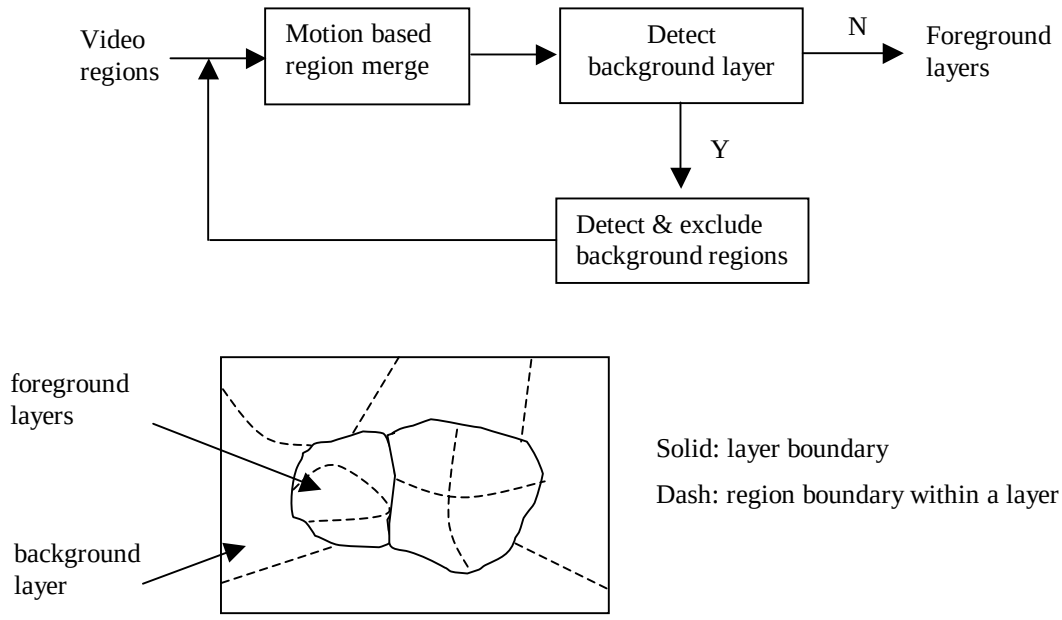


Figure 4.12 Iterative motion layer detection procedure

$$D(i, j) = \min(MCErr(R_i, M_j), MCErr(R_j, M_i)) \tag{4.8}$$

where M_i and M_j are the affine motion models of region R_i and R_j respectively. $MCErr(R, M)$ is the motion compensation error of region R under motion model M. A region *i* is merged with its closest neighbor if their distance is below a given threshold *TH_AFF*.

$$Merge(i,k) = \begin{cases} 1 & D(i,k) < TH\_AFF \\ 0 & D(i,k) \geq TH\_AFF \end{cases} \quad where\ D(i,k) = \min_j (D(i, j)) \tag{4.9}$$

where *j* refers to all neighboring regions of region *i*.

After regions are merged into motion layers, we try to identify one background layer in each iteration[1]. This is based on the assumption that a foreground layer must have discontinued motion fields around its outer boundaries. Boundaries of a layer are

consisted of pixels that have at least one neighboring pixel not belonging to the layer. Outer boundary is the outmost closed curve that contains the whole layer. Assume $b_1,\ldots,b_n$ are the *n* points along the outer boundary of a layer l[1], we define the following energy function to measure its boundary discontinuity.

$$E_l = \frac{1}{n}\sum_{p=b1}^{bn} G(p) \quad and \quad G(p) = \max(|p_1 - p_8|, |p_2 - p_7|, |p_3 - p_6|, |p_4 - p_5|) \tag{4.10}$$

where $p_1,\ldots,p_8$ are motion vectors of the 8 neighbors of point *p*, as shown in **Figure 4.13**.

p_1	p_2	p_3
p_4	p	p_5
p_6	p_7	p_8

Figure 4.13 Neighoring motion vectors of point *p*

A layer *l* is detected as a potential background layer only when E_l is smaller than a threshold (e.g., 0.4). If no background layer is detected, the algorithm stops and all remaining regions belong to foreground layers. When there are more than one possible background layers, the largest one is chosen as the background, and its affine motion model is used to compensate non-background regions. Those regions with small compensation errors are classified as background, and excluded from the next iteration of layer merging and detection. After multiple iterations, multiple background layers and multiple foreground layers may be produced.

[1] A background layer may be broken into disjoint pieces, and detected in several iterations.

4.3.3 Moving Object Detection Using Temporal Constraints

The foreground layers detected at individual frames are not reliable due to: 1) noisy nature of the motion field and inaccurate motion models; and more importantly, 2) a moving object may have noticeable motions in some frames where it can be easily detected, but small motions in other frames where it may be mistakenly treated as background. A global decision through an entire shot is necessary to remove noises and achieve reliable results.

To apply temporal constraints, we first link foreground layers (i.e., to track them) in individual frames according to their underlying regions. A foreground layer L_i^m in frame *m* is linked with a layer L_j^n in frame *n*, if the following condition is satisfied:

$$\left| L_i^m \cap L_j^n \right| = \max_{k,l} \left(\left| L_k^m \cap L_l^n \right| \right) \tag{4.11}$$

where $k \in \{ \textit{foreground layers in frame } m \}$ and $l \in \{ \textit{foreground layers in frame } n \}$. The intersection of two layers is defined as the common regions they both contain. In addition, we also define the link as a conductive relationship, which means if layer A and B, B and C are linked respectively, then A and C are also linked. This ensures that each local motion layer belongs to one and only one temporal layer.

The above linking or tracking process results in a number of groups of foreground layers. We will refer these groups as *temporal layers* below. We use some spatio-temporal constraints to validate these temporal layers. The first one is the duration of a temporal layer. As dominant moving objects are usually followed by the camera, layers with short duration are likely to be background regions, and thus are dropped. Secondly,

---

[1] Boundary points along the frame borders are not included.

the frame-to-frame changes of center coordinates and sizes of a temporal layer are examined. If there are large and abrupt changes, the temporal layer is not a valid tracking and will not be detected as a foreground object.

Finally, a morphological open and close procedure is applied at individual frames to remove small isolated regions and to fill holes within a moving layer.

In summary, our method is designed to automatically detect and track salient moving objects. By using temporal constraint, we can robustly and accurately segment moving objects over a long period. Our method can also handle objects with discontinuous motions (i.e., moving in some frames and still in other frames). There are some challenging issues that are not addressed in our approaches.

For example, the temporal occlusion is not considered here. When one moving object is first moving, then occluded by another object or background, and later appear as a separate moving object again, it will be treated as a new moving object. However, we can use region based object matching to detect reoccurrence of the same object after occlusion. Another case is that because small regions cannot be tracked reliably over a long period, the algorithm is not suitable to segment small moving objects (e.g., less than 500 pixels). On the other hand, large foreground objects with uniform motion may cause background regions to be wrongly detected as foreground (as shown in **Figure 4.14**). In this case, background regions only spatially connect with foreground objects. Therefore, the criteria based on motion discontinuity in boundary pixels cannot distinguish the background from foreground. We handle this issue based on the assumption that background regions are more likely to be located around the corners and borders. In **Figure 4.14**, while no background regions are detected in the above iterative process, the

region that has maximum number of pixels along frame borders is declared as the initial background.

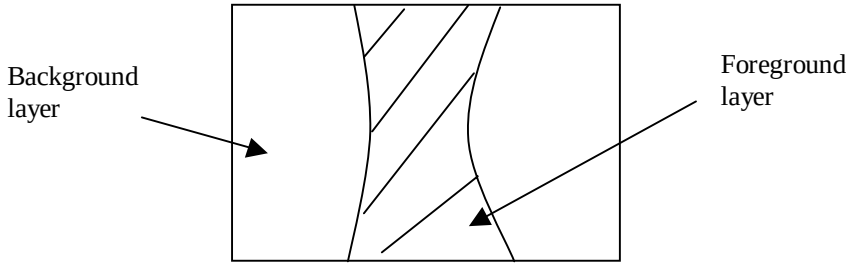


Figure 4.14 When background layers connect only to a large foreground layer, background is detected based on spatial locations of these motion layers

Since our algorithm does not require consistent motion model for a moving object[1], multiple moving objects will be declared as one single object if they are spatially connected in some frames within a shot. To handle this potential issue, our algorithm can be modified to include object-matching techniques (e.g., region-based matching method that will be discussed in Section 4.4), as well as rules to split one merged motion layer into multiple moving objects.

4.3.4 Results and Discussion

The object detection results of 5 sequences are presented in **Figure 4.15**. In **Figure 4.15**, each row we first show the image of frame #1, and then show the moving object tracking results at frame #1, #10, #20 and #30. They all have depth variance and camera motion (i.e. following the moving objects) in the scenes, resulting in multiple motion layers. The first sequence contains a skater running towards the camera. The ice field has a gradual

[1] One moving object may contains multiple parts with different motions.

depth change from near to far. In the second sequence, a person is working away from the camera in an office. Cubic walls exit at different depths. The third sequence is a bird-eye's view of a soccer player running in the field. Sequence 4 contains three background layers, which are the ground, wall and crowd. The last sequence contains the sky, the stage and the jumping skier. Note that regions within segmented objects are shown in random colors to demonstrate region segmentation results. One region being tracked at different frames is shown with the same color.

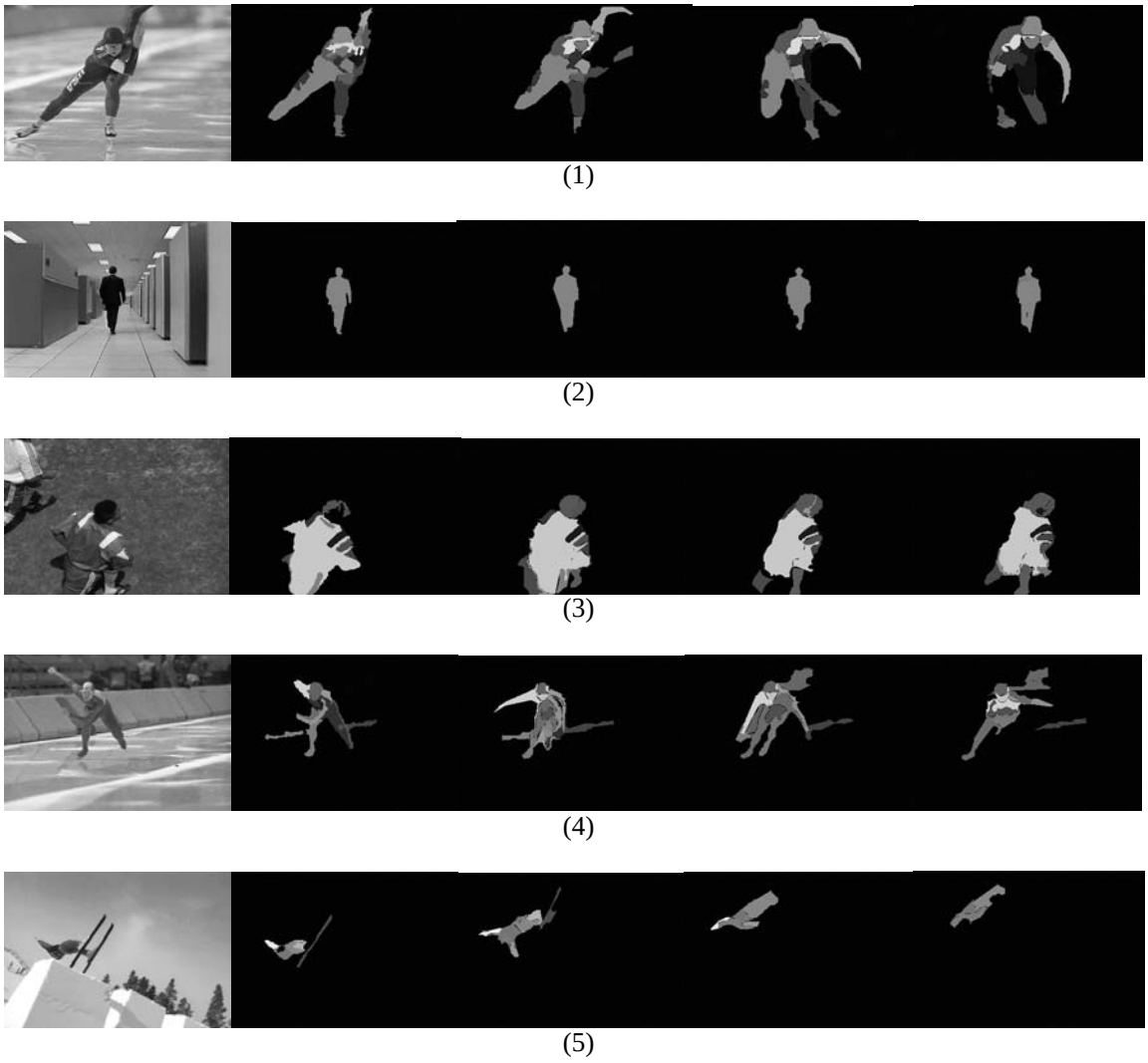


Figure 4.15 Moving object detection and tracking results of five image sequences (detected objects are show at frame #1, #10,#20,#30)

The gradual depth change in the sequence 1 does not cause much problem as the ground is merged into one large region in the first color based region segmentation stage. In sequence 2, the cubic walls are tracked as separated regions. Although these regions are classified as foreground objects in some frames, their temporal durations are short[1] and thus are considered as background. In the third sequence, both the player and the grass field have gradual depth variances. Similar to the first sequence, color segmentations are proven to be useful in handling such situations. Another player at the left-upper corner exists for a short period and is not detected as a target object. All the above three sequences show good tracking results. Some small background regions are falsely included in the sequence 4. These regions are mainly from the connecting parts of two background regions, and usually have inaccurate motion fields. Some foreground pixels are missed in (5) is because small isolated regions are removed in the final morphological operations.

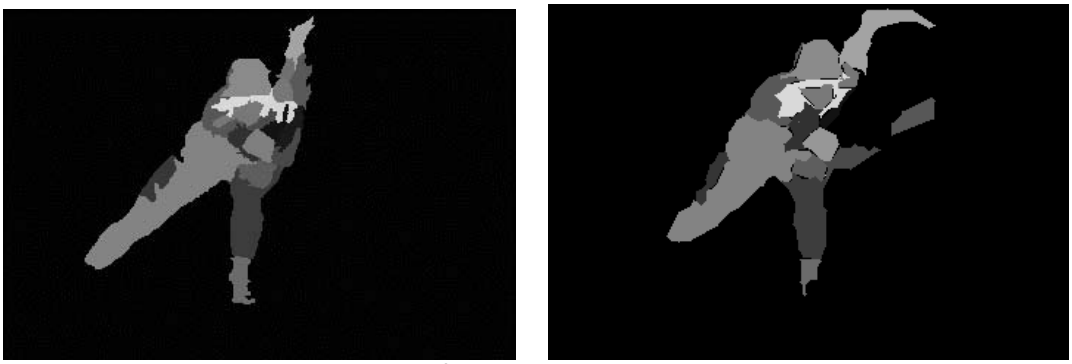
To show the temporal consistency constraints greatly improve moving object detection results, we present some intermediate detection results in **Figure 4.16a** (final detection results are shown in **Figure 4.16b** for comparison). It contains local motion layers detected at two frames (frame 1 and 10) in the sequence (1). As we can see, some foreground regions are merged into background because they have estimated motion models similar to that of background at these particular frames. By applying temporal constraint over an entire shot, we are able to recover missing foreground regions.

In summary, we demonstrated that long-term region based moving object detection approach is more robust and reliable compared to existing approaches that only uses local motion information (e.g., frame-to-frame motion field). In chapter 5, we will show an

[1] The camera lens is moving forward to follow the person

application system using this technique for real-time detection and tracking of tennis players.

(a) local detection result at individual frames



(b) final detection results

Figure 4.16 Moving layer detection result at individual frames

4.4 Region Feature Based Video Object Search

In the last chapter, we introduced the VideoQ system that is built upon our automatic video region segmentation and feature extraction algorithms. The system supports sketch based visual queries in which users can define one or more query objects with various visual features. Although VideoQ supports localized feature matching, it mainly uses low-level uniform feature regions. These regions do not necessarily correspond to meaningful real-world objects. While it has the advantage of being fully automatic, further efforts have to be made to support high-level semantic queries.

Semantic video objects introduce one more description level in the visual feature library of VideoQ. One might suppose that similarity matching of video objects is similar to matching of multiple regions with spatial-temporal relationships. However, several new critical issues emerge. First, underlying regions of a video object are tightly connected with each other, and thus similarity matching of the spatial-temporal structure become more important and should be performed more precisely. There are also important visual features that are unique in the object level, such as the temporal variances of size and orientation, repeated appearances at the different times. Finally, the spatial-temporal information of video objects available in MPEG-4 (i.e., BIFS) can be used to support high-level content-based search with multiple objects.

Here as the first step, we would like to address the problem of similarity matching of video objects using localized visual features. Based on the hierarchical object model and the segmentation framework in the AMOS system, we extend the VideoQ system to develop a new content-based search system for semantic objects. It utilizes an abundant set of visual features at both the object and region levels, including color, shape, texture, motion trajectory and temporal information. We also developed an effective integration of global and localized feature matching, as well as the measurement of various spatial-temporal structures (directional, topological, temporal).

Similar to the VideoQ system, there are three steps to build the video object indexing and search system. The first step is to a salient region extraction process within video objects. In the second step, visual features at both the object and region level are then extracted. This process builds a visual feature library for segmented video objects. Finally, an object searching model is developed, which effectively searches the visual

feature library, combines global and local features and examines the spatial-temporal structures of video objects.

4.4.1. Generating Salient Feature Regions

As we discussed before, salient feature regions are desired for the content based searching purpose. Although in the AMOS system, the underlying regions of video objects are created and tracked during the object segmentation process, these regions are usually small and short (see 4.2.2). Extra efforts (e.g., grouping) are required to create salient feature regions that can facilitate efficient object retrieval. For simplicity, here we apply the salient region segmentation algorithms (refer chapter 3) to do a second pass region tracking inside object masks. This is based on the consideration that the second pass salient region tracking process is relatively simple and fast. Furthermore, as an added benefit, the second-pass region tracking process can be used to help users refine previously segmented objects, as we will discuss next.

Although the above automatic region segmentation and tracking process can generate satisfactory salient regions, sometimes users may still want to define higher level regions which are not uniform in terms of visual features. An optional user interactive process is provided to further merge or eliminate tracked regions. Users can click on two neighboring regions at any frame, and ask the system to merge them in all the frames where these two regions have been tracked. In general, this allows flexible user control to create desirable region segmentation within the video object. This control of each individual tracked region also allows users to refine object masks by dropping a falsely included background region, which provides another way for users to correct uncovered

background regions in AMOS. Instead of stopping the object tracking process to correct errors, users can drop such background regions after the second-pass region tracking.

After the above salient region segmentation process, each segmented region is stored as a list of masks (approximated using polygons) with its starting and ending frame number in a video sequence. Video objects are stored in the same way. All these data will be used for the following feature extraction process.

4.4.2 Building the Visual Feature Library

A large set of visual features of video objects and their underlying regions are computed and stored for the subsequent queries. In the following, we will describe these features and show how they are computed at both the object and region levels. Note that to preserve temporal variations, ideally all feature data can be computed at every frame of an object or a region. However, this is unrealistic and also unnecessary for the purpose of searching. Thus currently we compute frame-by-frame values for only certain important features that are needed for critical temporal structured analysis. Other features are computed only at the starting frame where an object or region appears. Another approach to capturing temporal information is to store feature data at fixed sampled frames or at frames where feature differences exceed certain thresholds.

- Temporal Position - starting and ending frames of a video object or region in a video sequence. This information is available from the tracking processes.
- Motion Trajectory - the motion compensated centers of a video object or region at all of its successive frames. At each frame, a global motion model of camera (affine

model) is estimated from background pixels (pixels outside the object mask) and used to compensate the position of the object or region.

- Color - the representative color (in CIE L*u*v* space) of a region. It is passed from the region segmentation process. The overall color histogram is extracted as the color feature of an object.
- Texture - Tamura model of objects and regions, including three measures: coarseness, contrast and orientation.
- Shape (global) - global shape descriptors are computed for a video object and its regions at a number of sampling frames. Currently four descriptors are included: normalized area, aspect ratio, circularity, and orientation.
- Shape (local) - boundary polygon of a video object. This feature is not computed for regions.
- Shape Variance - currently the system detects the size change (increase/decrease) and rotation (left/right) of video objects and regions. They are both two-dimensional feature vectors with 4 possible values (0,0), (1,0), (0,1) and (1,1), where 1 in the first dimension stands for increase or left, and 1 in the second dimension stands for decrease or right.
- Spatial Connection graph - each region has a list of neighboring regions. This feature is generated during the region tracking process.

Besides the above features, more importantly, we also propose some structure features that capture the spatial and temporal relationship among feature regions. These features are pre-computed and stored so that they can be quickly accessed and examined in the similarity matching process.

There are several different ways to compare spatial structure, such as 2D-Strings [23] and spatial-temporal logical [14]. We choose a relatively fast and simple algorithm, the spatial-orientation graph, which represents spatial relationships between a set of regions as edges in a weighted graph [49]. This graph can be easily constructed by computing the orientation of each edge between the centers of a pair of query regions, and stored as a n*(n-1)/2-dimension feature vector, where n is the number of query regions (Figure 4.17a). As the spatial-orientation graph cannot handle the contain relationship, we extend it with a topological graph (Figure 4.17b), which defines the contain relation between each pair of regions with three possible values: contains(1), is-contained(-1) or not containing each other(0). Similarly, the temporal graph (Figure 4.17c) defines whether one region starts before(1), after(-1) or at the same time(0) with another region. Note here for simplicity we only define the temporal order according to the first appearing time (or starting time) of each region. By taking the ending time, a more complicated temporal relation graph can also be generated. As shown in **Figure 4.17**, feature data of the two graphs are also represented as n*(n-1)/2-dimension vectors.

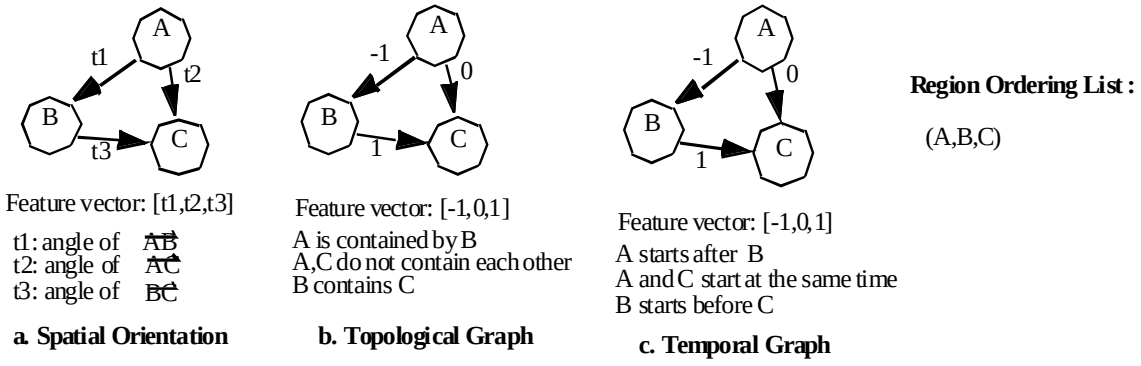


Figure 4.17 Examples of the three spatial-temporal relationship graphs

4.4.3 Region based Object Query Model

Given a query object (composition of a set of query regions), there are generally two searching approaches. One is to directly match query object against objects in the database. One example is the R-Tree based search methods is used in [86]. However, R-tree is not suitable for indexing of a large set of high-dimension features.

The other searching approach is to first find a matching region list for each query region based on visual features, and then "join" these region lists to find the best matched video objects by combining visual and structure similarity metrics. This approach is used in our earlier work, VisualSEEK [99]. In [63], Li and Smith further proposed a querying scheme that divides a composite query into a sequential of representation of sub-queries. A fast dynamic programming method is then used to retrieve the best matches for a composite object. However, sequentialization of a large set of visual and structure features is difficult. Here we use a parallel query and join scheme which supports partial matches. The object-searching diagram is shown in **Figure 4.18**.

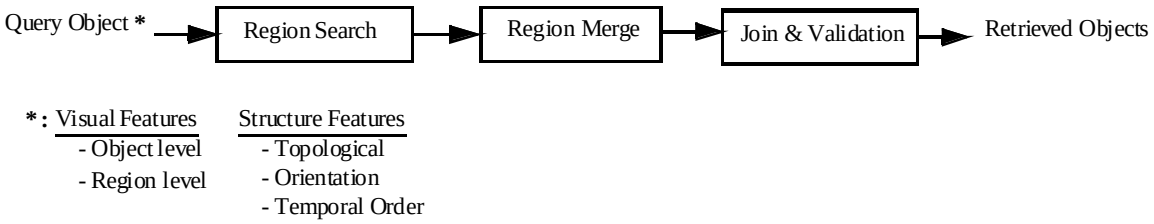


Figure 4.18 The parallel object query model

Given a query object with N regions, the object searching process consists of three stages. The first stage, region search, is to find a candidate region list from the database for each query region. The second stage, region merge, is to merge regions from a same video object into a large "virtual region". The final stage, join & validation, is to join the

candidate region lists to produce the object candidate list and to compute the final global distance measure. The detailed procedure is given as follows:

1) For every query region, find a candidate region list based on the weighted sum (according to the weights given by users) of distance measures of different visual features[1] (e.g., shape or trajectory). Only regions with distances smaller than a threshold are added to a candidate list. Here the threshold is a pre-set value used to empirically control the number or percentage of objects the query system will return. For example, a threshold 0.3 indicates that users want to retrieve around 30 percent of video objects in the database[1]. The threshold can be set to a large value to ensure completeness, or a small value to improve speed.

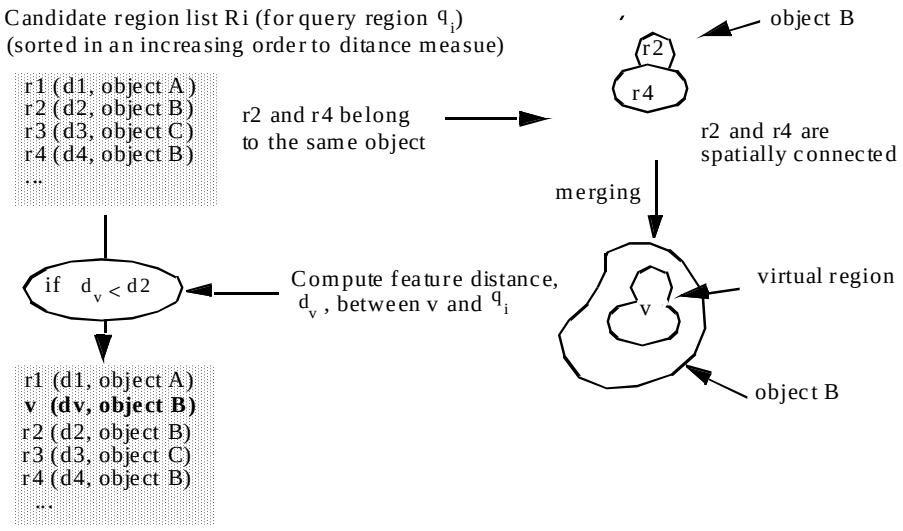


Figure 4.19 Query Time Region Merging Process

2) Sort regions in each candidate region list by their ObjectID's[2], and perform query time region merge. As users don't know exactly which segmented regions are included in the database, query regions may not match regions in the database on some visual features,

[1] Individual feature distances are normalized to [0,1]

e.g., size and shape. Furthermore, due to the nature of automatic region segmentation process (i.e., consistency of certain visual features), segmented regions are sometime smaller than query ones. To alleviate this problem and improve searching accuracy, we also develop a novel query time region merging process (**Figure 4.19**). For each candidate region list, the query system will try to merge regions from a same video object into a large "virtual region" if 1) they spatially connected with each other (the region connection graph); 2) the merged region is closer to the query region (feature distance). When a "virtual region" is generated, it is added to the current candidate region list for the following validation processes.

3) Perform join (outer join) of the region lists on ObjectID to create a candidate object list. Each candidate object in turn contains a list of regions. A "NULL" region is used when :

- a region list doesn't contains regions with the ObjectID of a being-joined object
- a region appears (i.e. matched) more than once in a being-joined object

4) Compute the distance between the query object and each object in the candidate object list as follows:

$$D = w_0 \sum_i FD(q_i, r_i) + w_1 SD(sog_q + sog_o) + w_2 SD(topo_q + topo_o) + w_3 SD(temp_q + temp_o) \quad (4.12)$$

where q_i is the ith query region. r_i is the ith region in a candidate object. FD(.) is the feature distance between a region and its corresponding query region. If r_i is NULL,

---

[1] Assume the feature vectors in the database have normal distribution in feature spaces
[2] Each video object or region has a unique ObjectID or RegionID.

maximum distance (i.e., 1) is assigned. sog_q (spatial orientation), $topo_q$ (topological relationship) and $temp_q$ (temporal relationship) are structure features of the query object. sog_o, sog_o and sog_o are retrieved from database using indices on ObjectID, RegionID and temporal positions. When there is a NULL region (due to the above join process), the corresponding dimension of the retrieved feature vector will have a NULL value. SD(.) is the L1-distance and a penalty of maximum difference is assigned to any dimension with a NULL value.

5) Sort the candidate object list according to the above distance measure D and return the result.

4.4.4 Experimental System and Results

A prototype system has been developed to demonstrate and evaluate our proposed visual similarity searching method which integrates matching of localized feature matching and spatial-temporal structures. We created a semantic video object database with 104 video objects from different types, including people, sports, animal, flowers and transportation. About 500 salient regions and their visual features are extracted and stored in the database.

Figure 4.20 shows the system's user interface. On the left panel, users can browse a video object database (arbitrarily shaped objects) and view the subsequent query results. To compose a visual query on the right canvas, users can draw regions from scratch or load underlying regions of an example video object from the database. Once a set of regions have been created on the canvas, users can assign and change their visual attributes to form a query object. These attributes include size, shape, spatial positions,

color, texture, motion trajectories, temporal position and temporal shape changes (e.g., a growing or rotating region). The global features of the query object, such as size, shape, spatial position, can be easily derived from the drawn regions. Motion trajectory and temporal shape change of a video object need to be specified separately. As users may want to search video objects without specifying any local region or only specify a subset of regions, the system also allows users to specifically define one area as the global object boundary that has its own size and shape.

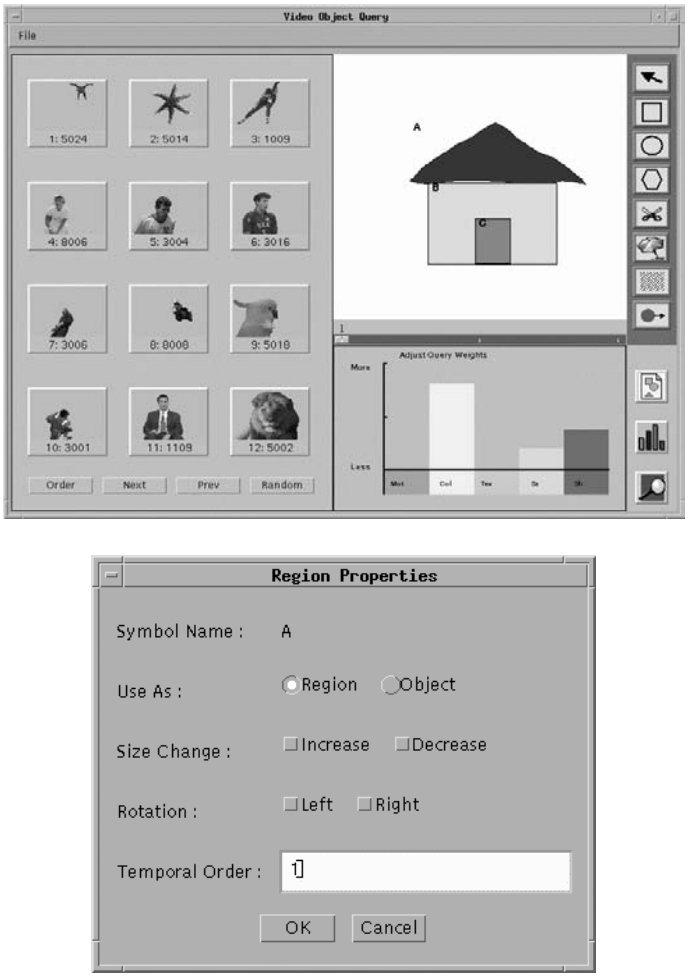


Figure 4.20 User interface of the object query system

With all these properties, users can create a query object with specific visual features and spatial-temporal structures, and then choose a set of weights to start the search

process. These weights define how important different region features and object features are in the similarity matching functions. Users also have the option of specifying matches of spatial and/or temporal structures in the query. For object shape matching, users can choose to use either global descriptors or local descriptors (e.g. polygon matching [73]).

Our query experiments have shown encouraging results. The matching of structure features has been proved to be critical in finding objects with multiple parts and special relationships (e.g. human body). Four query examples are shown in **Figure 4.21**. In the first example (Figure 4.21a), we are trying to find flowers by specifying two regions with color, shape and spatial relationship. Higher weight is put on the shape feature. Our experiments show that the shape matching of the stem part plays a key role in successfully finding flowers. In the second example (Figure 4.21b), we define a more complex query object, which has four parts with specific spatial relationship. While color and shape features are important in finding similar feature regions for each part, the matching of spatial structure eliminates false candidates and brings correct objects to the top of the return list. Figure 4.21c shows an example of partial matching. Users draw a face with two eyes. In the feature library, small regions like eyes are usually not extracted. Using the outer join and partial match method, we are able to find similar faces including those do not have extracted eye regions (results except the first one shown in Figure 4.21c). Figure 4.21d shows an example of trajectory matching. The retrieved objects are moving to the left-bottom direction on the screen.

(a) flowers

(b) body

(c) face

(d) motion

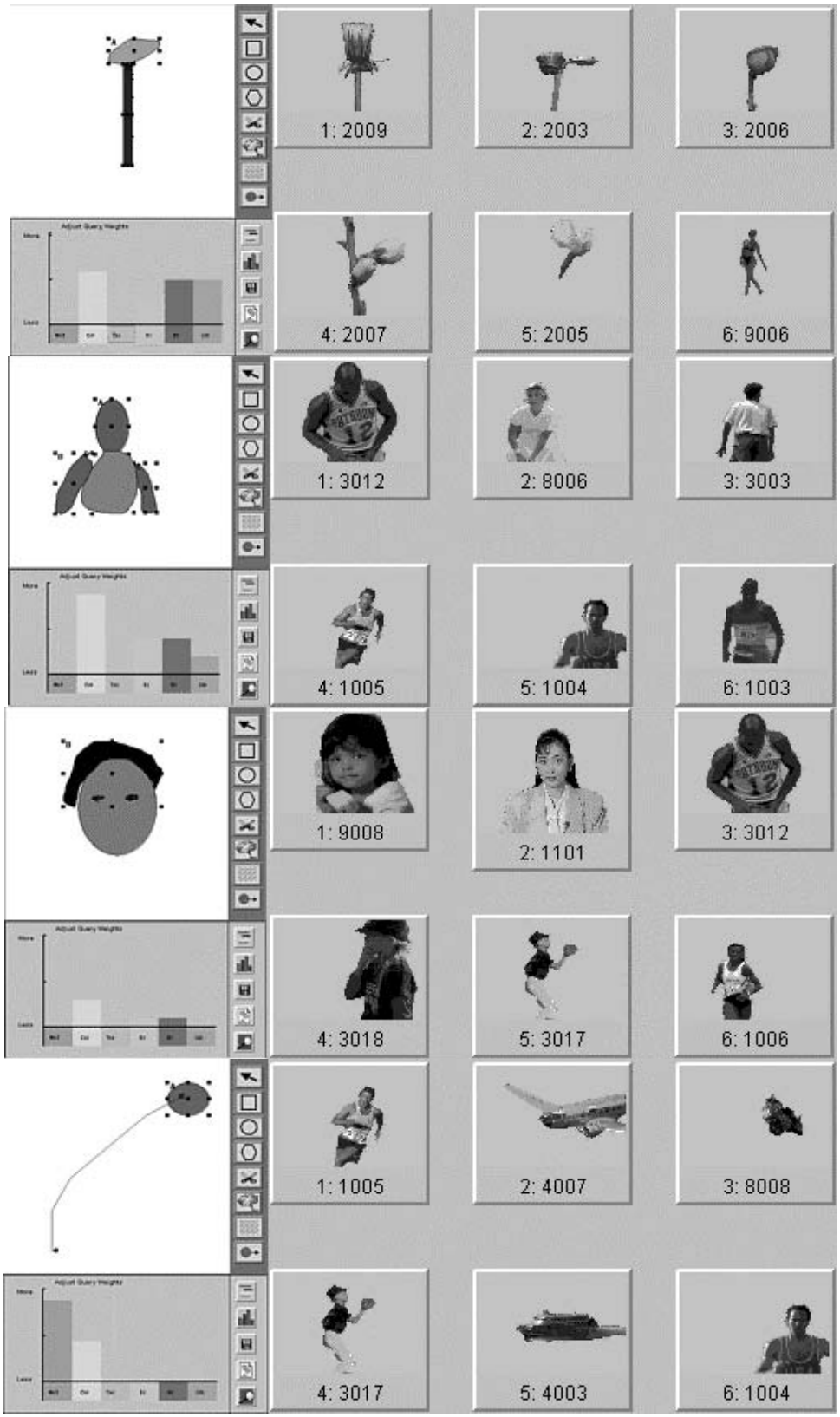


Figure 4.21 Examples of Video Object Search

Chapter 5

Structure Parsing and Event Detection for Sports Video

5.1 Introduction

For video indexing, typically videos are first processed to obtain constituent objects and features. These extracted entities provide an intermediate content model to effectively describe videos. In earlier chapters, we have studied temporal segmentation that generates elementary video shots, and spatial-temporal segmentation that extracts video regions and objects. We also developed feature matching algorithms and query methods to support visual similarity search. These works provide very useful tools for accessing online digital videos.

However, the objects and features we have extracted so far contain little information at the semantic level. To solve this problem, we can explore the knowledge and constraints in specific domain and apply domain-specific rules and/or certain machine learning techniques. In this chapter, we present an event detection and structure parsing system for sports videos. This system is built on top of segmentation and search techniques we have presented in prior chapters. We also demonstrate a summarization and browsing interface that allows users to easily access content structure and event index of video data.

Combining domain knowledge and automatically extracted low-level features have been seen in some works on news and movie videos. In [123], an edge model is used to match anchorperson views in specific news programs. The story structure is then reconstructued by finding anchorperson views as well as commercials. In [50], logical story units (LSU's) are extracted from movies according to global temporal consistency. The consistency is based on the assumption that an event is related to a specific location and to certain characters.

In this chapter, we will present a structure analysis framework for sports video using domain models. Compared with other types of videos, sports videos have different characteristics. A sports game usually occurs in one specific playground, contains abundant motion information and has well-defined content structures. Event detection in basketball and tennis videos has been studied in [96] and [102] respectively, but without attempts to reconstructing high-level structures. We will further discuss these two in later sections.

In this chapter, we present a real-time structure parsing and event detection system for sports videos. Compared to existing work, our solution has the following unique features.

- A general framework for video structure discovery and event detection.
- Combination of domain-specific knowledge and generic machine learning techniques.
- Multi-stage scene detection algorithms using our unique moving object segmentation and feature extraction methods.
- Real time processing performance by exploring redundancy in the compressed video streams.

- Higher accuracy demonstrated in specific sports domains such as tennis and baseball.

In the rest of this chapter, we will present our real-time prototype systems that automatically analyzes sports videos using tennis video as an example, and demonstrate experimental results in tennis and baseball videos.

5.2 Semantic Content in Sports Video

Sports video is a major part in most broadcasting TV programs, and has a large number of audience. Compared to other videos such as news and movies, sports videos have well-defined content structure and domain rules.

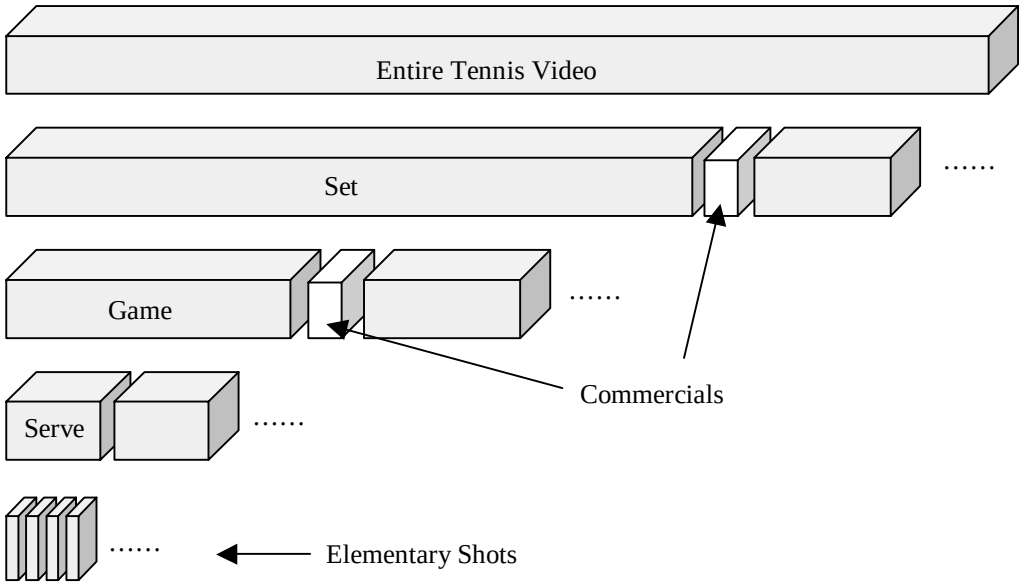


Figure 5.1 The temporal structure of a typical tennis video

First, sports videos are structured. A long sports game is often divided into a few segments. Each segment may again contain some sub-segments. Furthermore, there are a fixed number of cameras in the field that result in unique scenes during each segment.

For example, in football, a game contains two halves, and each half has two quarters. Within each quarter, there are many plays, and each play starts with the formation in which players line up on two sides of the ball. A tennis game is divided first into sets, then games and serves (**Figure 5.1**). Usually when a serve starts, the scene is switched to court views (**Figure 5.2**). In addition, for TV broadcastings, there are commercials or other special information (e.g., score board, player's name) inserted between segments.

The above characteristics of specific domains provide opportunities for developing new algorithms and tools to parse video structures, and to reconstruct some kinds of build table-of-contents for sports video indexing. As the structure is fixed given a type of sports, domain models can be used to greatly improve detection accuracy. The resulting systems and tools will enable users to have access any arbitrary segment at the desired level based on the intuitive and informative structure.

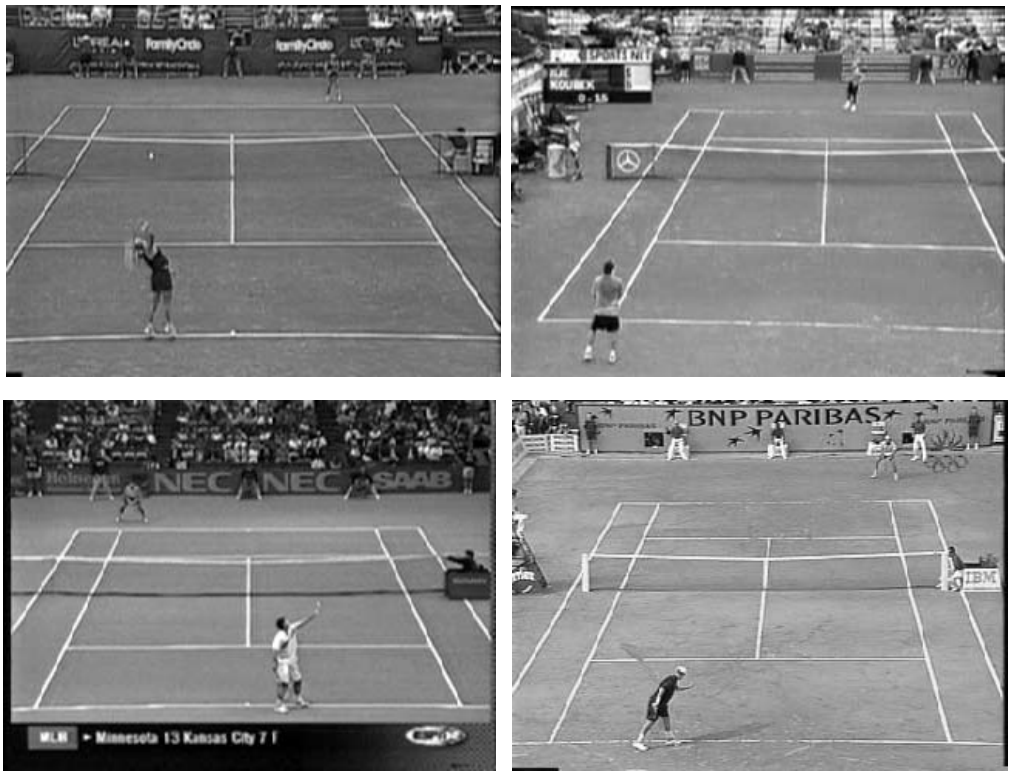


Figure 5.2 Scenes from four different tennis games

Another property of sports videos is that there are many special events in the video that are interesting to most audience. Typical examples include touch-downs in football, shots in basketball, home-runs in baseball, aces in tennis and so on. Within one sports, an event usually has similar visual-audio attributes that do not vary greatly from game to game. Baseball games are played in grass fields, and when there is a home-run, the camera will quickly follow the ball across the field and there will be loud screams from the audience. Tennis courts typically have uniform ground colors and white court lines, and within a serve, the camera is typically placed at shots from the top-rear position (e.g., a few scenes from different games are shown in **Figure 5.2**). These characteristics allow us to adopt some domain-specific rules and apply machine-learning techniques using automatically segmented objects and features to detect interesting events. Hence, given a sports video, we can identify some important events that are interesting to most audience, and automatically or semi-automatically build an index to all the occurrences of these events. Similar to the keyword index in a book, this event index enables viewers to access desired scenes much more intuitively and efficiently compared to traditional fast forward and backward search.

A few works have been conducted to analyze sports videos. In [96], a basketball annotation system is presented. Using prior knowledge of basketball games, the system detects wide-angle shots versus close-up shots. Camera motions are further analyzed within wide-angle shots and used to detect possible fast break, steal and probable shots. When users want to retrieve one type of event (e.g., fast break), only sequences satisfying corresponding motion criteria are shown. Tennis videos are studied in [102]. The approach is based on the extraction of a model for the tennis court-lines from videos. A

player tracking algorithms is developed to track players, and then a reasoning module is used to map low-level positional information to high-level tennis play events.

The issue addressed in above researches is to detect certain events using low-level features such as motion and position. The well-defined temporal structures were not explored. Another challenging problem that has not been well addressed is the real time constraint. Sports video broadcasting is mostly live, and interests of audience go down greatly after the results are known. The real-time capability in generating event indices is critical in practical applications for sports videos.

In this chapter, we present a general framework to parse temporal structure of live broadcasted sports videos. In the proposed approach, analyzing rules are first built through a training phase. In the operation phase, these rules are updated to adapt to new data in live inputs and used to select candidate scenes. Scene verification approaches using our segmentation techniques are then applied to reduce false positives. Within the detected scenes, we will also show innovative event detection tools using segmentation techniques we have developed in previous chapters.

5.3 Real Time Video Analysis System

We briefly describe the system architecture of the real time sports video analysis platform. Details about how segmentation and feature extraction algorithms are adapted to meet the real-time requirement will be included in later sections where structure parsing and event detection are discussed.

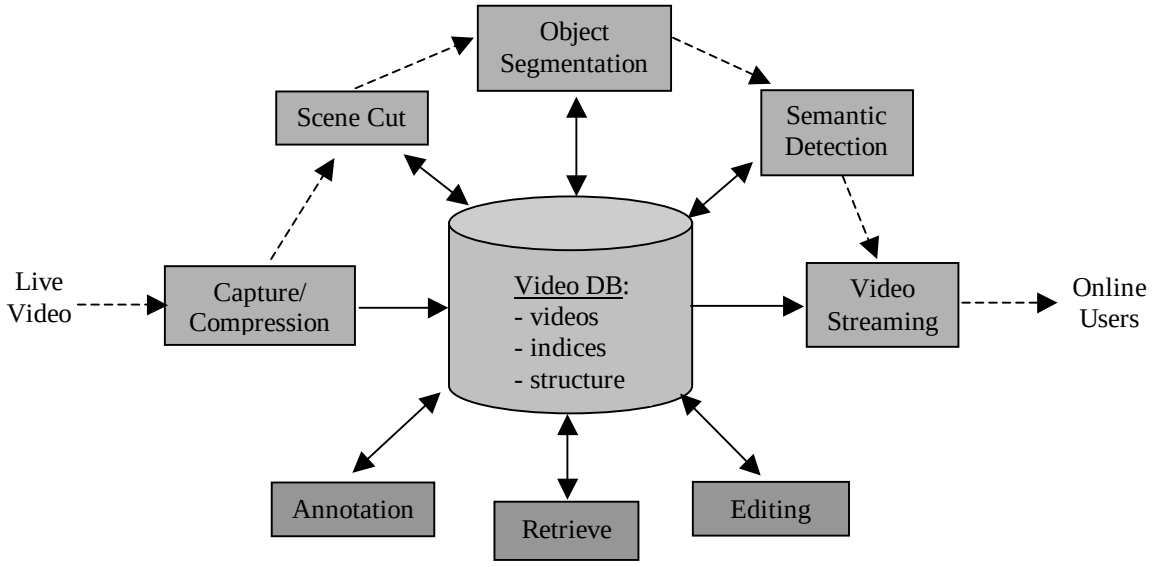


Figure 5.3 System architecture of the real time video analysis platform

The system is built on a personal computer with Microsoft Window NT 4.0. Real-time performance is reached on a Pentium-III 600 dual CPU machine. The system allows users to build digital video database taking regular analog video sources such as TV broadcast, cable, VCR, or video camera. As shown in **Figure 5.3**, the dashed arrows indicate real-time flow of video data. Some main function modules are as follows.

- Capture and Compression – An MPEG compression card is used to capture and compress video and audio signals. Text decoder card can be added to obtain closed caption signal.
- Scene Cut Detection – The scene cut detection scheme presented in chapter 2 is used. It detects scene cuts first in MPEG compressed domain with partial decoding. For gradual scene changes, chrominance and edge features are compared in the un-compressed domain. To achieve real-time performance, we only decompress and analyze I-frames. The output scene cut positions and representative frames (e.g., the first I-frame after a scene cut) are stored in the video database.

- Object Segmentation and Feature Extraction – Object segmentation and feature extraction are the most time-consuming processes, and cannot be applied to all frames. To spot important scenes, representative frames are always examined first. When a potential candidate is located, segmentations and more detailed features are computed and used in the subsequent semantic detection module. Thus, although the mission of this module is to extract visual features, it requires a semantic filtering component. The component depends on domain knowledge and should use only simple features that can be quickly computed, such as global color histograms.
- Semantic Detection – The detection tasks can be classified into two categories: one is to parse the structure information; another is to find interesting events. The detection process can also be divided into 1) local, which is applied to individual frames; 2) temporal, which is applied to one or more videos shots. The former one usually precedes the later one. All detected results are stored into the video database to construct structure table or event index.
- Video Streaming –Videos are delivered to users in real-time together with structure parsing and event detection results. The semantic and event information can be used to filter the video content and send a personalized version of the video to each individual user.
- Annotation, Editing – These are usually post processing with manual interactions. However, they may be applied in real time too. For example, when certain events or objects are detected, additional information such as hyperlinks or highlights can be immediately inserted or overlaid to the original streams. It is also possible to cut

and/or replace some portions of broadcasted videos in real time (as in advertisement insertion).

- Retrieve – With our real-time analysis system, a live video can be searched right after, or even during the broadcasting time.

In summary, our scene analysis algorithms work as modules along a streaming pipeline. Such a stream-line architecture is important in applications that may need real-time performance or application specific customization.

The above real-time parsing and indexing system has a lot of applications. It can be used to create large searchable video archives. It can also be used in web-casting of sports programs in which game outlines and scene highlights can be rapidly developed on web sites. The system can also be combined with existing sports annotation software, which allow reporters to annotate and log a game in real time. In interactive applications, interactive user interface with virtual-reality like environments can be enhanced with simultaneous displays of graphics and synchronized video clips.

5.4 Structure Parsing

Given the structure model of a game, we need to detect visual cues that indicate the beginning and/or ending of each section or sub-section. Some common features occurring between top-level sections are commercials, embedded texts and special logos. Many methods have been proposed for commercial and text detection [64,95]. Here we will study the detection of basic units within a game, such as serves in tennis. These units usually start with a special scene. Simple color based approaches have been suggested [102]. Based on our experiments to be described later, this type of approaches can only

reach about 80 percent accuracy. Furthermore, as color information varies from game to game, adaptive methods need to be exploited to handle such variation. In our real-time framework, we first use a fast adaptive color filtering method to select possible candidates, then apply more complicated features to verify them against domain specific rules or models. In the following we present the system using serves in tennis as an example.

5.4.1 Color Based Adaptive Filtering

Color based filtering is applied to key frames of video shots. It contains two stages as shown in **Figure 5.4**, the training stage and then the operation stage.

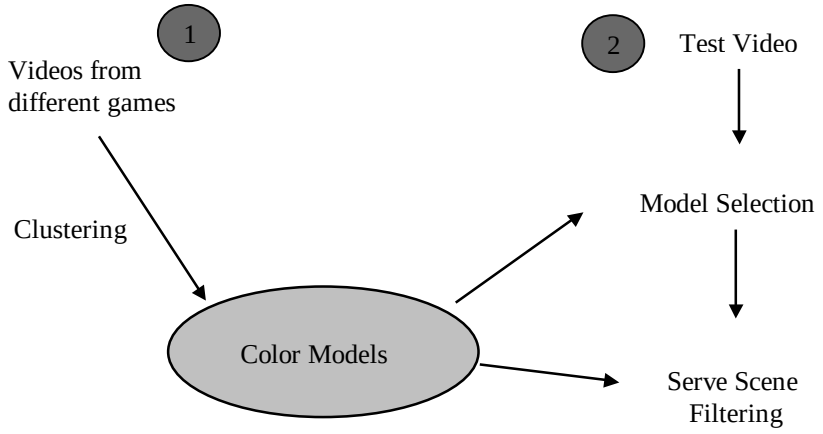


Figure 5.4 The color based adaptive filtering process

First, the filtering models are built through a clustering based training process. The training data should contain enough games so that a new game will be similar to some in the training set. Assume h_i, $i=1\ldots,N$ are color histograms of all serve scenes in the training set. A k-means clustering is used to generate K models (i.e., clusters), $M_1,\ldots,M_K$, such that,

$$h_i \in M_j \ , \quad if \ \ D(h_i, M_j) = \min_{k=1}^{K}(D(h_i, M_k)) \tag{5.1}$$

where $D(h_i, M_k)$ is the distance between h_i and the mean vector of M_k, i.e. $H_k = \frac{1}{|M_k|}\sum_{h_i \in M_k} h_i$ and $|M_k|$ is the number of training scenes being classified into the model M_k. This means that for each module M_k, H_k is used as its the representative feature vector.

When a new game coming in, proper models need to be chosen to spot serve scenes. This raises a typical egg-and-chicken problem, as we need to know serve scenes to select a correct model. To solve the problem, we detect the first *L* serve scenes using all models, $M_1, \ldots, M_K$. That is to say, all models are used in the filtering process. If one scene is close enough to any model, the scene will be passed through to subsequent verification processes (Eq 5.2).

$$h_i^{'} \in M_j, \quad if \ D(h_i^{'}, M_j) = \min_{k=1}^{K}(D(h_i^{'}, M_k)) \ \ and \ \ D(h_i^{'}, M_j) < TH \tag{5.2}$$

where $h_i^{'}$ is the color histogram of the *i*th shot in the new video, and *TH* is a given filtering threshold to accept shots with enough color similarity. Shot *i* is detected as a serve scene if the segmentation based verification, which will be described in 5.4.2, is also successful, and we mark this serve as being founded by model M_j (i.e., classify the scene into the model M_j). If the verification fails, $h_i^{'}$ is removed from the set of M_j.

After *L* serve scenes are detected, we find the model M_o, which leads to the search for the model with the most serve scenes.

$$|M_o| = \max_{k=1}^{K}(|M_k|) \tag{5.3}$$

where $|M_k|$ is the number of incoming scenes being classified into the model M_k. In the filtering process for subsequent shots, model M_o and a few of its most close neighboring models are applied in a same way as define in Eq 5.2.

5.4.2 Segmentation Based Verification

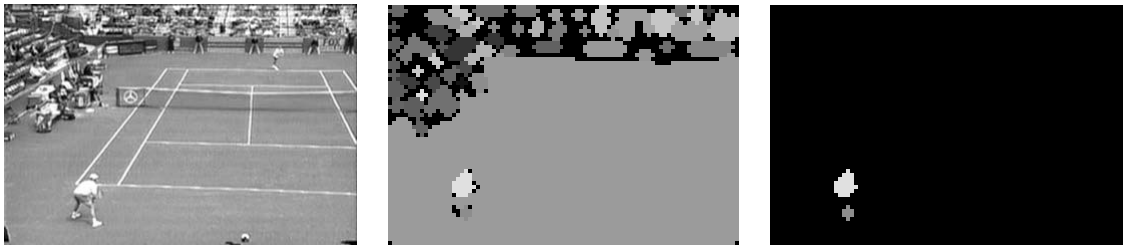
Color histograms are global features that can be computed faster than real time. However, using only color the detection accuracy is less than 80%. Many close-up scenes of playgrounds and replay scenes are likely to be detected as false positives.

To improve detection accuracy, the salient feature region extraction and moving object detection we proposed in chapter 3 and section 4.3 can be utilized here to produce localized spatial-temporal features. Compared with global features, such as color histograms, spatial-temporal features are more reliable and invariant to detect given scene models. Especially in sports videos, special scenes are often made of several objects at fixed locations. The similarity matching scheme of visual and structure features we studied in previous chapters can also be easily adapted here for model verification.

To verify serve scenes, the moving object detection algorithm (section 4.3) is applied. To achieve real-time performance, segmentation is performed on the down-sampled images of the key frame (which is chosen to be an I-frame) and its successive P-frame. The down-sampling rate used in our experiment is 4, both horizontally and vertically, which results in images with size 88x60. Motion fields are estimated using the hierarchical approach as proposed in [13]. An example of segmentation and detection results is shown in **Figure 5.5**.

Figure 5.5 (b) shows the region segmentation result. The court is segmented out as one

large region, while the player closer to the camera is also extracted. The court lines are not preserved due to the down-sampling. Black areas shown in (b) are tiny regions being dropped at the end of segmentation process. Figure 5.5 (c) shows the moving object detection result. In this example, the result is quite satisfactory, and only the desired player is detected. Sometimes a few background regions may also be detected as foreground moving object. We will address this issue in section 5.5 when discussing the player tracking algorithm. Here for verification purpose, as we will describe below, the important thing is not to miss the player.



a. original key frame b. region segmentation c. moving object detection

Figure 5.5 An example of automatic region segmentation and moving object detection

Following rules are applied in scene verification. First, there must be a large region (e.g. larger than two-thirds of the frame size) with consistent color (or intensity for simplicity). This large region corresponds to the tennis court. The uniformity of a region is measured by the intensity variance of all pixels within the region (Eq 5.4).

$$Var(p) = \frac{1}{N}\sum_{i=1}^{N}[I(p_i) - \bar{I}(p)]^2 \tag{5.4}$$

where N is the number of pixels within a region *p*. $I(p_i)$ is the intensity of pixel *I and* $\bar{I}(p)$ is the average intensity of region *p*. If *Var(p)* is less than a given threshold, the size of region *p* is examined to decide if it corresponds to the tennis court.

Secondly, the size and position of player are examined. The condition is satisfied if a moving object with proper size is detected within the lower half part of the previously detected large “court” region. In a downsized 88x60 image, the size of a player is usually between 50 to 200 pixels. As our detection method is applied at the beginning of a serve, and players are always at the bottom line to start a server, the position of a detected player has to be within the lower half part of the court.

5.4.3 Edge Based Verification

One unique characteristic of serving scenes in tennis game is that there are horizontal and vertical court lines. Ideally if a camera shots straightforward from top-rear point of the court and all court lines are captured (**Figure 5.2**), rules for a complete court can be used to verify serve scenes with high precision. However, in a real scene, due to the camera panning and zooming, or object occlusion, usually not all court lines are viewable. Trying to match a full court will result in a low recall rate of serve scenes.

Since we already go through color based filtering and region based verification processes, relatively loosen constraints are enforced on court lines. An example of edge detection using the 5x5 Sobel operator is given in **Figure 5.6**.

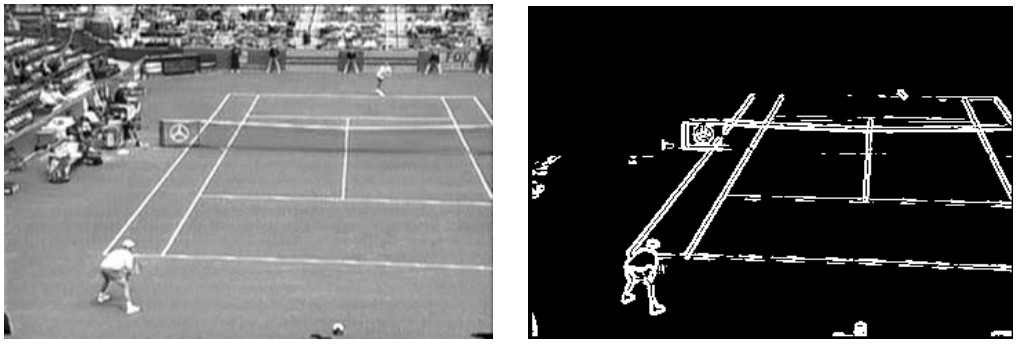


Figure 5.6 Edge detection within the court region

Note that the edge detection is performed on a down-sampled (usually by 2) image and inside the detected court region. Hough transforms are conducted in four local windows to detection straight lines (**Figure 5.7**). Windows 1 and 2 are used to detect vertical court lines, while windows 3 and 4 are used to detect horizontal lines. It greatly increases the accuracy in detecting straight lines to use local windows instead of a whole frame. As shown in the figure, each pair of windows roughly cover a little bit more than half of a frame, and are positioned somewhat close to the bottom border. This is based on the observation of the usual position of court lines within court views.

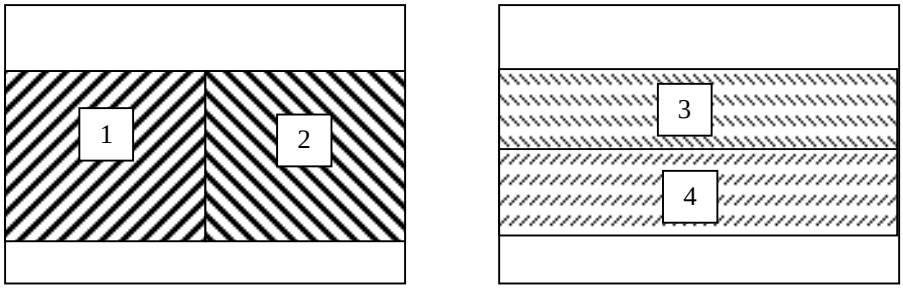


Figure 5.7 Local windows for hough-transform based line detection

The verifying condition is that there are at least two vertical court lines and two horizontal court lines being detected. Note these lines have to be apart from each other for a certain distance, as noises and errors in edge detection and Hough transform may produce duplicated lines. This is based on the assumption that despite of camera panning, there is at least one side of the court, which has two vertical lines, being captured in the video. On the other hand, camera zooming will always keep two of three horizontal lines, i.e., the bottom line, middle court line and net line, in the view.

5.4.4 Experiments and Discussion

We applied the above filter and verification scheme to tennis and baseball videos, with

different color models and verification rules respectively. Our experiment results on a one-hour tennis and a one-hour baseball video is shown in Table 5.1. Our scheme shows very good performance on both videos. The overall recall and precision rates are about 95%.

Table 5.1 Detection results for serve and pitch scenes

	Ground truth	# of Miss	# of False
Tennis (serve)	89	7	2
Baseball (pitch)	93	3	4

This result is very good compared to approaches using only colors. Based on our experiments, previously proposed approaches using color histogram filtering can only achieve about 80% precision rate in order to obtain near 100% recall. No serve scenes should be dropped in the first filtering stage (i.e., recall rate is maximized), thus a low filtering threshold is used. Furthermore, despite of using advanced segmentation and feature extraction, our proposed scene detection and verification process is performed in real time.

One thing to point out here is that the color models used in the first filtering stage include clusters extracted from the first 10 minutes of the testing video as well as some videos from some other games (from different channels or games). As different games may have different color distributions, in real applications, we need to have a fairly comprehensive color models extracted from different games in order to properly process new incoming games. While this is a critical issue for color only approaches, our approach is quite flexible in handling such situations. Considering there are only limited types of play fields, we can lower our filtering threshold to allow more false alarms in the first stage, and rely on the feature based verification process to improve precision rate.

5.5 Event Detection

In parsing the game structures, we have presented a scene detection approach using spatial-temporal features. In this section, focus will be on detecting and summarizing what happened in an interesting scene. Especially, we will adapt our moving object detection algorithm to track a tennis player in real-time, and analyze his or her trajectory to obtain certain play facts.

5.5.1 Player Tracking

In chapter 4, we presented an automatic moving object detection method that contains two stages: an iterative motion layer detection step being performed at individual frames; and a temporal detection process combining multiple local results within an entire shot. Here we adapt this approach to track tennis players within court view in real time. Currently we focus on the player who is close to the camera. The player at the opposite side is smaller and not always in the view. It is harder to track small regions in real time because we need to down-sample spatially to reduce computation complexity.

The local motion layer detection process is similar to what we previously explained in 5.4.2 (**Figure 5.6**), down-sampled I- and P-frames are segmented and compared to extract motion layers. The reason to skip B-frames is because bi-direction predicted frames require more computation to decode. To ensure real-time performance, only one pair of anchor frames are processed every half second. For a MPEG stream with a GOP size of 15 frames, the I-frame and its immediate following P-frame are used. Motion layer detection is not performed in later P frames in the GOP. This change requires a different temporal detection process to detection moving objects. The process is described as

follows.

As half second is a rather large gap for the estimation of motion fields, motion-based region projection and tracking from I frame to another I frame are not reliable, especially when a scene contains fast motion. Thus, a different process is required to match moving layers detected at individual I-frames. We use the following temporal filtering process to select and match objects that are detected at I frames.

Assume O_i^k is the *kth* object (*k=1,…,K)* at the *ith* I-frame in a video shot, $\vec{p}_i^k$, $\vec{c}_i^k$ and s_i^k are the center position, mean color and size of the object respectively. We define the distance between O_i^k and another object at *jth* I-frame, O_j^l, as weighted sum of spatial, color and size differences.

$$D(O_i^k, O_j^l) = w_p \left\| \vec{p}_i^k - \vec{p}_j^l \right\| + w_c \left\| \vec{c}_i^k - \vec{c}_j^l \right\| + w_s \left| s_i^k - s_j^l \right| \tag{5.5}$$

where w_p, w_c and w_s are weights on spatial, color and size differences respectively. If $D(O_i^k, O_j^l)$ is smaller than a given threshold, *O_TH*, objects O_i^k and O_j^l match with each other. We then define the match between an object with its neighboring I-frame $i+\delta$ as follows,

$$F(O_i^k, i+\delta) = \begin{cases} 1 & \exists\, O_{i+\delta}^l,\ D(O_i^k, O_{i+\delta}^l) < O\_TH \\ 0 & otherwise \end{cases} \tag{5.6}$$

where $\delta = \pm 1, ..., n$. Let $M_i^k = \sum_{\delta=\pm1,...,n} F(O_i^k, i+\delta)$ be the total number of frames that have matches of object O_i^k (*k=1,…,K)* within the period $i-\delta$ to $i+\delta$, we select the object with maximum M_i^k. This means that if $M_i^r = \max_{k=1,...,K}(M_i^k)$, the *rth* object is kept at the *ith* I-frame. The other objects are dropped. The above process can be considered as a general

temporal median filtering operation.

After the above selection, we obtain the trajectory of the lower player by sequentially taking the center coordinates of the selected moving objects at all I-frames. There are some issues that need some discussion here. First, if no object is found in a frame, linear interpolation is used to fill the missing point. When there are more than one objects being selected at a frame (in the situation when more than one objects have the same maximum number), the one that is spatially close to its precedent is used. In addition, for speed reason, instead of using affine model to compensate camera motion, here we use the detected net lines to roughly align different instances.

Experimental tracking results of one sever scene is shown in **Figure 5.8**.

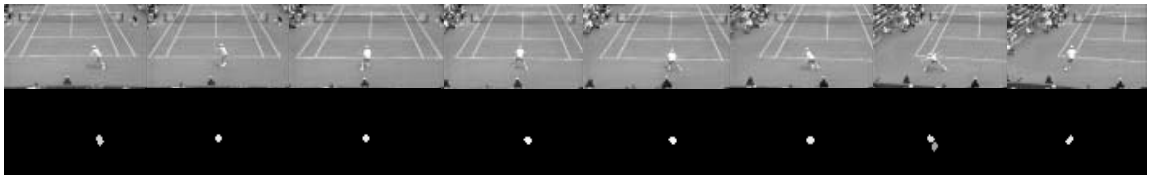


Figure 5.8 Tennis player tracking within a serve scene (results are shown on 8 I-frames with a 15-frame interval)

The first row shows down sampled frames. The second row contains final player tracking results. The body of the player is well tracked and detected. Successful tracking of tennis players provides a foundation for high-level semantic analysis. Compared with the tracking algorithm in [102], which computes residual errors to find moving objects and then searches players in pre-defined windows, our approach provides more accurate as well as real time performance.

5.5.2 Trajectory Analysis

The extracted trajectory is analyzed to obtain play information. Presently, we focus on

two aspects. The first one is the position of a player. As players usually play at bottom lines, we want to find cases when a player moves to the net zone. The second one is to estimate the number of strikes a player conducted within a serve. Users who want to learn stroke skills or play strategies may be interested in serves with more strikes.

Given a trajectory containing *K* coordinates, $\vec{p}_k$ *(k=1,…,K),* at *K* successive I-frames, we first detect "still points" and "turning points". $\vec{p}_k$ is a still point if,

$$\min(\|\vec{p}_k - \vec{p}_{k-1}\|, \|\vec{p}_k - \vec{p}_{k+1}\|) < TH \tag{5.7}$$

where *TH* is a pre-defined threshold. Furthermore, two consecutive still points are merged into one. If point $\vec{p}_k$ is not a still point, the angle at the point is examined. $\vec{p}_k$ is a turning point if

$$\angle(p_k p_{k-1}, p_k p_{k+1}) < 90^o \tag{5.8}$$

An example of object trajectory is shown in **Figure 5.9**. After detecting still and turning points, we use them to judge the player's positions. If there is a position close to the net line (vertically), the serve is classified as net-zone play. The estimated number of strokes is the sum of the numbers of turning and still points.

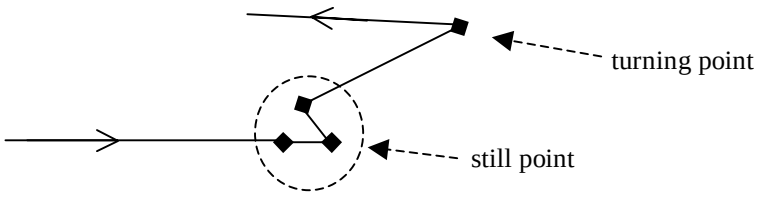


Figure 5.9 Detection of still and turning points in object trajectory

Experiment results of the above one-hour video are given in Table 5.2. In the video, the ground truth includes 12 serves with net play within about 90 serve scenes (see Table

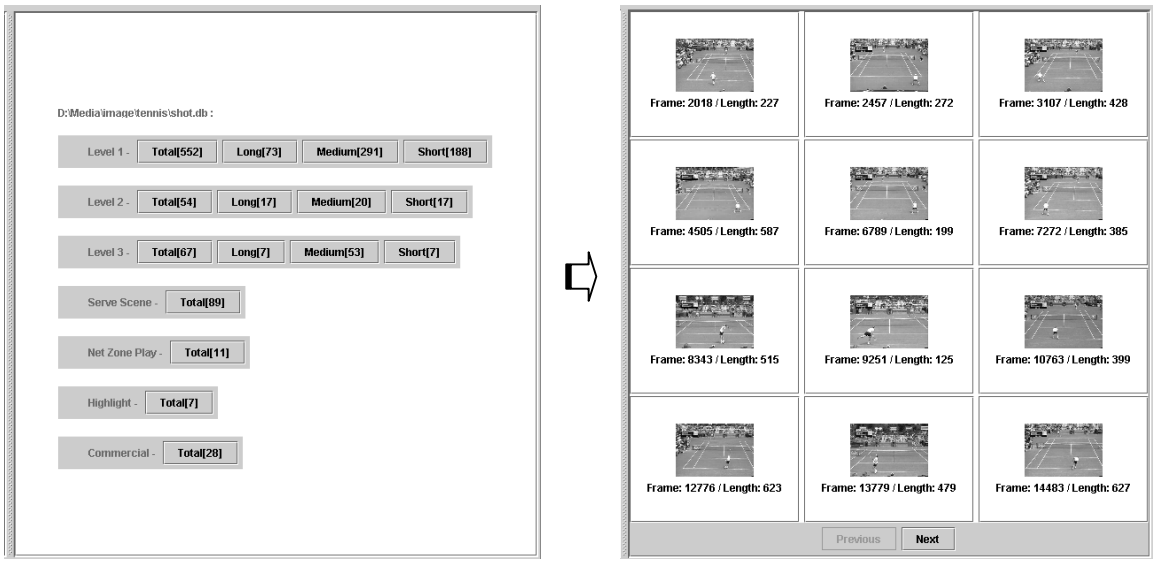
5.1), and totally 221 strokes in all serves. Most net plays are correctly detected. False detection of net plays is mainly caused by incorrect extraction of player trajectories or court lines. Stroke detection has a precision rate about 72%. Beside the reason of incorrect player tracking, some errors are caused by limitations of our estimation model. First, at the end of a serve, a player may or may not strike the ball in his or her last movement. Many serve scenes also show players walking in the field after the play. In addition, a sever scene sometimes contain two serves if the first serve failed. These may cause problems since currently we detect strokes based on the movement information of the player. To solve these issues, more detailed analysis of motion such as speed, direction, repeating patterns in combination with audio analysis (e.g., hitting sound) will be very useful.

Table 5.2 Trajectory analysis results for one hour tennis video

	# of Net Plays	# of Strokes
Ground Truth	12	221
Correct Detection	11	216
False Detection	7	81

5.6 A Summarization and Browsing Interface

As we have observed, video data contain large amount of visual and semantic information. Even after content indices are generated, how to show these indices in a limited display is still a challenging issue. In this section, we present a system for video browsing and summarization using the structure parsing and event detection results presented in this thesis. The system has two unique access methods for users to find desired video shots.



(a) (b)

Figure 5.10 Summarization interface providing scene index to video

First, the system provides a summarization interface (**Figure 5.10a**). It shows the statistics of video shots, such as number of long, intermediate and short shots. It also shows the number of some common kinds of scenes in a specific domain. For instance, in tennis, there are serve, net-zone play, commercial and etc.). Seeing these summaries, users may follow up with more specific request by choosing a category (e.g., serve). As shown in **Figure 5.10b**, from here users can directly go to any interesting scenes.

The second interface combines the sequential temporal order and the hierarchical structure among all video shots within a video. As shown in **Figure 5.11**, the structure tree is in the left window. In this example, games are listed at the top level. There are commercial breaks between games. Under each game, there are many serves. Each serve contains the serving shots and a few follow-up shots.

In the video shown in **Figure 5.11**, the first game includes 16 serves. Each serve segment is labeled with the length of the segment, type of play in this serve and the

approximate number of strokes in a serve. For example, a label “(L) S B 4” means long segment, server, base-line play and approximately 4 strokes.

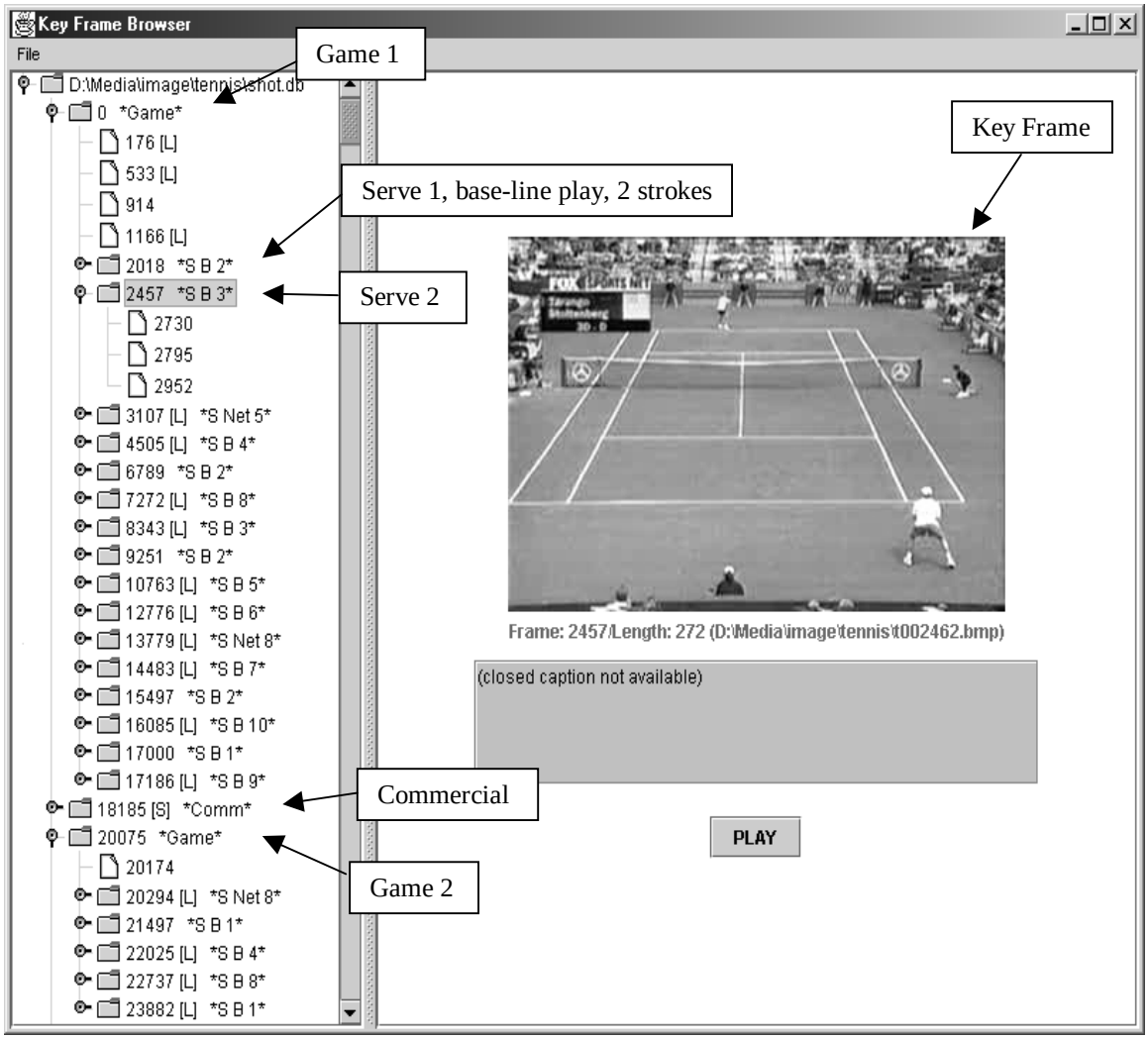


Figure 5.11 Browsing interface providing hierarchical structure

All these elements are organized as nodes in a tree. This allows users to navigate from more abstracted top levels to detailed levels. When users click on any nodes, the corresponding key frame will be shown in the right window, and users can start to play the video at the corresponding moment.

5.7 Conclusion and Open Issues

In this chapter, we presented a general framework for structure parsing and high-level event detection for sports videos, using tennis as an example. We have also tested this performance in baseball videos. It combines domain-specific knowledge with automatically segmented objects and features using generic machine learning techniques. Real time processing performance is achieved by exploring redundancy in the compressed video streams. Our experiments have demonstrated high accuracy in specific sports domains such as tennis and baseball.

Under the framework we have developed, there are many more issues that can be studied to produce a comprehensive summary of sports videos. For tennis videos, we can include audio information to detect the number of strokes within each serve more accurately. Score boards are common in all broadcasted sports programs. By detecting these text boxes and recognizing scores, we can understand the status of a game. For example, in a baseball game, we will be able to know when there is a new player entering the field. We can also exam object and/or camera motion at or around special scenes (e.g., serve in tennis, pitch view in baseball) to understand if any interesting events happen. In baseball, when a batter hits the ball, the camera will usually follow the flying ball or the running players.

By analyzing structures and detecting interesting events in videos, we will able to provide efficient browsing, searching and summarization functions to users, possibly in real time for live contents. We can also generate short highlights of important portions within a long game.

Chapter 6

Conclusion and Future Work

In this thesis, we present our work on segmentation, feature extraction, index and summarization of digital video content. A unified object-based video representation model is proposed and applied to build efficient video indexing and search systems. Our work addressed the following five basic issues involved in typical video indexing scenarios.

- The first one is temporal segmentation (i.e., scene cut detection), which breaks video streams into temporally consistent units.
- The second is spatial-temporal segmentation that extracts and tracks objects contained in videos.
- The third one is to define and compute effective features of extracted objects.
- The fourth one is to build an indexing and retrieval schema that supports good feature based matching.
- Finally, domain knowledge and production rules should be included and linked to low-level features to index high level events and structures in videos.

Following the natural processing flow of digital video content, we first demonstrated a robust and real-time scene cut detection schema that combines color, edge and motion

features on both compressed and uncompressed domains. The decision tree based learning method is used to build detection rules. We developed a new approach to find gradual transitions by examining their starting and ending edges. Special situations such as flashlights and aperture changes are studied. Our extensive experiments show that the algorithms perform well for different kinds of videos, including sports, sitcom, news, cartoon, movie and home videos.

Then we presented an automatic region segmentation system for content-based video search. The system segments and tracks spatial consistent regions through each video shot. It then computes visual features of extracted regions to build visual libraries that support region-based search for video shots. To track salient regions reliably through video shots, we developed new intra-frame and inter-frame segmentation methods using fusion of color, edge and motion features. A web-based video query system that has more than 3,000 video shots is built. Both visual and motion features are computed and stored. The query system allows users to do spatial-temporal search of video shots by drawing regions and specifying features. Promising results are observed in both segmentation and query experiments.

Based on the robust region segmentation and tracking techniques we have developed, semantic object segmentation and tracking is studied to support higher-level object-based video representation and description. We introduced an integrated schema for semantic object segmentation and content-based object search using the region-based video object model. A semi-automatic video object segmentation system, AMOS, which combines low level automatic region segmentation with user inputs, is developed to define and track semantic video objects. The system utilizes an iterative region aggregation and

boundary alignment process to generate and track objects. Our experiments show that our approach can accomplish fairly accurate object boundaries. Besides we also presented a fully automatic moving object detection method using global temporal consistency of tracked objects. Using the region-based video object model, an object query model, which effectively combines local region-level features and spatial-temporal structures, is then presented. Our experiments have shown promising results and great promise in developing advanced video search tools for semantic video representations. Such tools are critical in emerging applications and standards such as MPEG-4 and MPEG-7.

Finally, we present a real-time framework to reconstruct high-level structures and event index for live sports videos using domain specific models. We explore the production rules and regular transition structures in sports videos. This system utilized the segmentation and searching modules we have developed in earlier chapters to find unique scenes and events. We show that our video segmentation and indexing techniques can be effectively integrated into high-level video retrieval and browsing systems in specific domains such as sports videos. Good experiment results are demonstrated on tennis and baseball videos. Our system also includes new summarization tools and interface that allow users to comprehend the video structures and event statistics efficiently. Such components can be used to develop other functionalities such as semantic compression, content-based streaming and so on.

In conclusion, we have proposed and developed various models and methods for segmenting, indexing, searching and summarizing videos at low levels as well as semantic levels. One key concept in our works is the object-based representation. Recent MPEG-4 and MPEG-7 standards both utilize an object based content encoding and

description scheme. Our works in this thesis have shown promising results and huge potential in this direction.

Unlike old standards where a scene is represented as a composition of pixels, MPEG-4 adopts the concept in which the scene is represented as a composition of objects. The ongoing MPEG-7 standard also tries to describe video content in terms of objects and features. With these new standards, segmentation based video analysis and indexing tools are gaining more and more interest. As a freely distributed software, our AMOS system has been distributed to dozens of researchers and companies.

The methods and algorithms presented in this thesis are quite general. For different practical applications, we expect these algorithms to be improved by introducing domain models and knowledge. A smart MPEG-4 video camera for video conferencing would use the head-and-shoulder model to improve object segmentation/tracking accuracy and speed. A surveillance video indexing system would focus on detecting, tracking and summarization of moving objects with static backgrounds. For other unique specific domains, as we demonstrated for sports videos, semantic structures and events should be defined and extracted to support efficient browsing, searching, and summarization at the semantic level.

Chapter 6

Conclusion and Future Work

In this thesis, we present our work on segmentation, feature extraction, index and summarization of digital video content. A unified object-based video representation model is proposed and applied to build efficient video indexing and search systems. Our work addressed the following five basic issues involved in typical video indexing scenarios.

- The first one is temporal segmentation (i.e., scene cut detection), which breaks video streams into temporally consistent units.
- The second is spatial-temporal segmentation that extracts and tracks objects contained in videos.
- The third one is to define and compute effective features of extracted objects.
- The fourth one is to build an indexing and retrieval schema that supports good feature based matching.
- Finally, domain knowledge and production rules should be included and linked to low-level features to index high level events and structures in videos.

Following the natural processing flow of digital video content, we first demonstrated a robust and real-time scene cut detection schema that combines color, edge and motion

features on both compressed and uncompressed domains. The decision tree based learning method is used to build detection rules. We developed a new approach to find gradual transitions by examining their starting and ending edges. Special situations such as flashlights and aperture changes are studied. Our extensive experiments show that the algorithms perform well for different kinds of videos, including sports, sitcom, news, cartoon, movie and home videos.

Then we presented an automatic region segmentation system for content-based video search. The system segments and tracks spatial consistent regions through each video shot. It then computes visual features of extracted regions to build visual libraries that support region-based search for video shots. To track salient regions reliably through video shots, we developed new intra-frame and inter-frame segmentation methods using fusion of color, edge and motion features. A web-based video query system that has more than 3,000 video shots is built. Both visual and motion features are computed and stored. The query system allows users to do spatial-temporal search of video shots by drawing regions and specifying features. Promising results are observed in both segmentation and query experiments.

Based on the robust region segmentation and tracking techniques we have developed, semantic object segmentation and tracking is studied to support higher-level object-based video representation and description. We introduced an integrated schema for semantic object segmentation and content-based object search using the region-based video object model. A semi-automatic video object segmentation system, AMOS, which combines low level automatic region segmentation with user inputs, is developed to define and track semantic video objects. The system utilizes an iterative region aggregation and

boundary alignment process to generate and track objects. Our experiments show that our approach can accomplish fairly accurate object boundaries. Besides we also presented a fully automatic moving object detection method using global temporal consistency of tracked objects. Using the region-based video object model, an object query model, which effectively combines local region-level features and spatial-temporal structures, is then presented. Our experiments have shown promising results and great promise in developing advanced video search tools for semantic video representations. Such tools are critical in emerging applications and standards such as MPEG-4 and MPEG-7.

Finally, we present a real-time framework to reconstruct high-level structures and event index for live sports videos using domain specific models. We explore the production rules and regular transition structures in sports videos. This system utilized the segmentation and searching modules we have developed in earlier chapters to find unique scenes and events. We show that our video segmentation and indexing techniques can be effectively integrated into high-level video retrieval and browsing systems in specific domains such as sports videos. Good experiment results are demonstrated on tennis and baseball videos. Our system also includes new summarization tools and interface that allow users to comprehend the video structures and event statistics efficiently. Such components can be used to develop other functionalities such as semantic compression, content-based streaming and so on.

In conclusion, we have proposed and developed various models and methods for segmenting, indexing, searching and summarizing videos at low levels as well as semantic levels. One key concept in our works is the object-based representation. Recent MPEG-4 and MPEG-7 standards both utilize an object based content encoding and

description scheme. Our works in this thesis have shown promising results and huge potential in this direction.

Unlike old standards where a scene is represented as a composition of pixels, MPEG-4 adopts the concept in which the scene is represented as a composition of objects. The ongoing MPEG-7 standard also tries to describe video content in terms of objects and features. With these new standards, segmentation based video analysis and indexing tools are gaining more and more interest. As a freely distributed software, our AMOS system has been distributed to dozens of researchers and companies.

The methods and algorithms presented in this thesis are quite general. For different practical applications, we expect these algorithms to be improved by introducing domain models and knowledge. A smart MPEG-4 video camera for video conferencing would use the head-and-shoulder model to improve object segmentation/tracking accuracy and speed. A surveillance video indexing system would focus on detecting, tracking and summarization of moving objects with static backgrounds. For other unique specific domains, as we demonstrated for sports videos, semantic structures and events should be defined and extracted to support efficient browsing, searching, and summarization at the semantic level.

References

1. E. H. Adelson and J. R. Bergen, “Spatio-temporal Energy Models for the Perception of Motion”, Journal of the Optical Society of America A 2 (2), pp. 284-299, 1985.
2. G. Adiv, “Inherent ambiguities in recovering 3D motion and structure from a noisy flow field”, IEEE Trans. On Pattern Analysis and Machine Intelligence, 11:447-489, May 1989.
3. Gulrukh Ahanger, Dan Benson, and T.D.C.Little, "Video Query Formulation", Storage and Retrieval for Images and Video Databases II, IS&T/SPIE Symposium on Electronic Imaging Science & Technology, San Jose, CA, Feb 1995.
4. Edoardo Ardizzone, Mohand-Said Hacid: A Semantic Modeling Approach for Video Retrieval by Content. IEEE International Conference on Multimedia Computing and Systems, ICMCS 1999, 7-11 June, 1999, Florence, Italy, Proceedings, Volume II. IEEE Computer Society, 1999 ,Vol. 2 1999: 158-162.
5. F. Arman, A. Hsu, and M.-Y. Chiu, “Feature Management for Large Video Databases”, Proc. IS&T/SPIE Conf. on Storage and Retrieval for Image and Video Databases, San Jose, CA, 1993, pp. 2-12.
6. F. Arman, R.Depommier, A.Hsu, and M-Y.Chiu, "Content-Based Browsing of Video Sequences", Proceedings of ACM international Conference on Multimedia '94, Oct 15-20, San Francisco, CA, USA.
7. Y. Alp Aslandogan and Clenment T. Yu, “Techniques and systems for image and video retrieval”, IEEE Transactions on Knowledge and Data Engineering, Vol. 11, No. 1, January/February 1999
8. A. Ayer, and H. S. Sawhney, "Layered Representation of Motion Video Using Robust Maximum-Likelihood Estimation of Mixture Models and MDL Encoding", Proc.Fifth Int'l Conf Computer Vision, 1995.
9. J.L. Barron, D.J. Fleet, and S.S. Beauchemin, “Performance of optical flow techniques”, International Journal of Computer Vision, vol. 12, 1994, 43-77.
10. A.Bascle, P.Bouthemy, R.Deriche and F. Meyer, "Tracking complex primitives in image sequence", Proc. 12th International. Conference on Pattern Recognition, 1994.
11. J.R.Bergen, P.Anandan, K.J.Hanna and R.Hingorani, "Hierarchical model-based motion estimation", ECCV-92. pp237-252, Santa Margarita Ligure, May 1992.

12. J. Ross Beveridge, Joey Griffith, Ralf R. Kohler, Allen R. Hanson and Edward M. Riseman, “Segmenting images using localized histograms and region merging”, International Journal of Computer Vision, 2, 311-347, 1989.

13. M. Bierling, "Displacement Estimation by Hierarchical Block Matching", SPIE Vol 1001, Visual Communication & Image Processing, 1988.

14. A.D. Bimbo, E. Vicario and D. Zingoni, "Symbolic description and visual querying of image sequences using spatio-temporal logic", IEEE Transactions on Knowledge and Data Engineering, Vol 7, No. 4, August, 1995.

15. A.Blake, R. Curwen and A. Zisserman, "A framework for spatiotemporal control in the tracking of visual contours", International Journal of Computer Vision, 11(2):127-145, 1993.

16. Georgi D. Borshukov, Gozde Bozdagi, Yucel Altunbasak and A. Murat Telalp, “Motion segmentation by multistage affine classification”, IEEE transaction on image processing, Vol 6, No 11, Nov 1997.

17. P. Bouthemy and E. Framcois, “Motion segmentaion and qualitative dynamic scene analysis from an image sequence”, Int. Journal of Computer Vision, vol. 10, no. 2, pp157-182, 1993.

18. Charles S. Carman and Michael B.Merickel, "Supervising ISODATA with an Information Theoretic Stopping Rule", Pattern Recognition , Vol 23, No 1/2 pp.185-197, 1990.

19. Hyun Sung Chang, Sanghoon Sull and Sang Uk Lee, “Efficient Video Indexing Scheme for Content-Based Retrieval”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 9, No.8. Dec 1999.

20. Shih-Fu Chang and John R. Smith, "Extracting Multi-dimensional Signal Features for Content-Based Visual Query," SPIE Symposium on Visual Communications and Image Processing, May 1995.

21. S.-F. Chang, W. Chen, H. Meng, H. Sundaram, and D. Zhong, "VideoQ: An Automated Content-Based Video Search System Using Visual Cues", ACM 5th Multimedia Conference, Seattle, WA, Nov. 1997.

22. S. K. Chang and A. Hsu, "Image information systems: Where do we go from here?," IEEE Trans. on Knowledge and Data Engineering, Vol. 4, No. 5 (October 1992), pp. 431-442.

23. S.-K. Chang, Q. Y. Shi, and C. Y. Yan, "Iconic indexing by 2-D strings", IEEE Trans. Pattern Anal. Machine Intell., 9(3):413-428, May 1987.

24. J. G. Choi, Y.-K. Lim, M. H. Lee, G. Bang, J. Yoo, "Automatic segmentation based on spatio-temporal information", ISO IEC JTC1/SC29/W11 MPEG95/M1284, Sept 1996.

25. J.-G. Choi, Y.-K. Lim, M.H. Lee, G. Bang, J. Yoo, "Automatic segmentation based on spatio-temporal information", ISO IEC JTC1/SC29/W11 MPEG95/M1284, Sept

1996.

26. C.-T. Chu, D. Anastassiou and S.-F. Chang, "Hybrid Object-Based/Block-Based Coding in Video Compression at Very Low Bitrate", J. of Image Communication-Signal Processing, Special Issue on MPEG-4, 1997.

27. Tat-Seng Chua and Li-Qun Ruan, “A Video Retrieval and Sequencing System”, ACM Transactions on Information Systems, Vol 13, No. 4, pp373-407, Oct. 1995.

28. S.Colonnese, A. Neri, G.Russo and P.Talone, "Moving objects versus still background classification: a spatial temporal segmentation tool for MPEG-4", ISO/IEC JTC1/SC29/WG11 MPEG96/571.

29. Csinger, A., Booth, K. S., & Poole, D. A. “Reasoning about video: Knowledge-based transcription and presentation”, Proceedings of the 27th Annual Hawaii International Conference and System Sciences (pp. 599-608). Maui, Hawaii, 1994.

30. Serhan Dagtas, Wasfi AL-Khatib, “Models for Motion-Based Video Indexing and Retrieval”, IEEE Transactions on Image Processing, Vol. 9 No.1, Jan 2000.

31. R. Deriche and Olivier Faugeras, "Tracking line segments", Image and Vision Computing, 8(4):261-270, 1990.

32. N.Dimitrova and F.Golshani,,"Rx for Semantic Video Database Retrieval,", Proc. ACM Multimedia'94, pp.219-226, San Francisco, Oct 1994.

33. M.P. Dubuisson and Anil K. Jain, "Contour Extraction of Moving Objects in Complex Outdoor Scenes", International Journal of Computer Vision, Vol. 14, pp. 83-105, 1995.

34. C.Faloutsos, R Barber, M. Flickner, J. Hafner, W. Niblack, D. Peetkovic and W. Equitz, "Efficient and Effective Querying by Image Content", Journal of Intelligent Information Systems(1994), p231-262.

35. Andras Farago, Tanas Linder, and Gabor Lugosi "Fast Nearest-Neighbor Search in Dissimilarity Spaces", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 15, No.9, Sep 1993.

36. Martin A. Fischler and Oscar Firschein, "Readings in Computer Vision", Morgan Kaufmann Oublishers, Inc,, Los Altos, CA (1987).

37. K. S. Fu and J.K. Mui, “A survey on image segmentation”, Pattern Recognition 13, 3-16 (1981)

38. John M. Gauch, Susan Gauch, Sylvain Bouix, Xiaolan Zhu, “Real Time Video Scene Detection and Classification.”, Information Processing and Management 35(3): 381-400 (1999)

39. R. Geman and D.Geman, "Stochastic relaxation, Gibbs distributions and the Bayesian Restoration of Images", IEEE Trans. Pattern Analysis and Machine Intelligence, Vol PAMI-6, No. 11, pp. 721-741, 1984

40. P. Gerken, M.Wollborn and S.Schultz, "An object-based layered codec as proposal

for MPEG-4 scalability and compression tests - technical description", ISO/IEC JTC1/SC29/WG11 MPEG95/359.

41. Y. Gong, H. J. Zhang and T. C. Chua, "An Image Database System with Content Capturing and Fast Image Indexing Abilities", Proc. IEEE International Conference on Multimedia Computing and Systems, Boston, 14-19 May, 1994, pp.121-130.

42. William I Grosky and Zhaowei Jiang, "Hierarchical Approach to Feature Indexing", Image and Vision Computing Vol.12 , Jun 1994

43. Ullas Gargi, Rangachar Kasturi, and Susan H. Strayer, “Performance Characterization of Video-Shot-Change Detection Methods”, IEEE Transactions on Circuits and System for Video Technology,Vol 10, No. 1 Feb 2000

44. James Griffioen, Rajiv Mehrotra and Rajendra Yavatkar, “An Object-Oriented Model for Image Information Representation”, CIKM 1993, Proceedings of the Second International Conference on Information and Knowledge Management, Washington, DC, USA, November 1-5,

45. W. I. Grosky, “Multimedia Information Systems”, IEEE Multimedia, Spring, 1994, pp12-24.

46. J.J De Gruijter and A.B. McBRATNEY, "A Modified Fuzzy K-Means Method for Predictive Classification", Classification and Related Methods of Data Analysis,1988

47. Chuang Gu, T. Ebrahimi, M. Kunt, "Morphological Moving Object Segmentation and Tracking for Content-based Video Coding", Multimedia Communication and Video Coding, Plenum Press, New York, 1996

48. Chuang Gu and Ming-Chieh. Lee "Semantic Video Object Segmentation and Tracking Using Mathematical Morphology and Perspective Motion Model", ICIP'97, October 26-29, 1997 Santa Barbara, CA.

49. V.N. Gudivada and V.V. Raghavan, "Design and Evaluation of Algorithms for Image Retrieval by Spatial Similarity", ACM Transaction on Information Systems, Vol.13,No,2,April 1995, pp.115-144.

50. A. Hanjalic, R.L. Lagendijk, J. Biemond: *Automated High-Level Movie Segmentation for Advanced Video Retrieval Systems*, IEEE Transactions on Circuits and Systems for Video Technology, Vol.9, No.4, June 1999

51. S.Horowitz and T. Pavlidis, "Picture Segmentation by a directed split-and-merge procedure", Proc. 2nd Intl. Joint Conf. on Pattern Recognition, pp. 424-433, 1974

52. V.S.Hwang, "Tracking feature points in time-varying images using an opportunistic selection approach", Pattern Recognition, 22(3):247-256, 1989

53. Qian Huang, Zhu Liu, Aaron Rosenberg, “Automated Semantic Structure Reconstruction and Representation Generation for Broadcast News”, IS&T/SPIE Conference on Storage and Retrieval for Image and Video Database VII, San Jose, California, Jan 1999.

54. Mikihiro Ioka and Masato Kurokawa "A Method for Retrieving Sequences of Images on the basis of Motion Analysis", SPIE Vol. 1662 Image Storage and Retrieval Systems (1992)/35

55. D.-S. Jang, G.-Y. KIM and H.I. CHOI, "Model-based Tracking of Moving Object", Pattern Recognition, Vol 30, No 6, pp999-1008, 1997

56. Anilk Jain, Richard C Dubes, “Algorithms for Clustering Data”, Englewood Cliffs, NJ : Prentice Hall, 1988

57. M. Kass, A. Witkin and D. Terzopoulos, "Snakes: Active contour models", International Journal of Computer Vision, pp 321-331, 1988.

58. B.Kervrann and F. Heitz, "A hierarchical framework for the segmentation of deformable objects in image sequences", Proc. IEEE Conference on Computer Vision and Pattern Recognition, Seattle, pp. 724-728, 1994

59. Milind R. Naphade, Igor Kozintsev, Thomas S. Huang and Kannan Ramchandran, “A Factor Graph Framework for Semantic Indexing and Retrieval in Video”, IEEE Workshop on Content-based Access of Image and Video Libraries (CBAIVL-2000), Hilton Head, South Carolina USA, June, 2000

60. David A. Langan, James W. Modestino and Jun Zhang, "Cluster Validation for Unsupervised Stochastic Model-Based Image Segmentation", IEEE Transactions on Image Processing, Vol 7, No. 2, Feb, 1998

61. Denis Lee, Ron Barber, Wayne Niblack, Myron Flickner, Jim Hafner, and Dragutin Petkovic ,"Query By Image Content Using Multiple Objects and Multiple Features: User Interface Issues", Proceedings of 1st Intelnational Conf. on Image Processing (ICIP), Vol II, Nov. 1994

62. Suh-Yin Lee and Huan-Ming Kao, "Video Indexing - An Approach Based on Moving Object and Track", SPIE Vol 1908 (1993)/25

63. C.-S. Li, J.R. Smith, L. Bergman and V. Castelli, "Sequential Processing for Content-based Retrieval of Composite Objects", SPIE Storage&Retrieval of Image/Video DB, 1998.

64. Rainer Lienhart, Christoph Kuhmunch and Wolfgang Effelsberg, “On the Detection and Recogonition of Television Commercials”, Proc. of IEEE International Conference on Multimedia Computing and Systems, June, 1997, Ottawa, Canada

65. ISO/IEC 13818 –2 Committee Draft (MPEG-2)

66. "Description of MPEG-4", ISO/IEC JTC1/SC29/WG11 N1410, MPEG document N1410 Oct. 1996.

67. “Introduction to MPEG-7 (version 1.0)”, ISO/IEC JTC1/SC29/WG11 N3545, Beijing, July 2000

68. Rainer Lienhart, Silvia Pfeiffer, and Wolfgang Effelsberg, “Video Abstracting”, Communications of The ACM December 1997/Vol. 40, No.12

69. J. Meng, Y. Juan and S.-F. Chang, "Scene Change Detection in a MPEG Compressed Video Sequence, " SPIE Symposium on Electronic Imaging: Science & Technology-Digital Video Compression: Algorithms and Technologies, SPIE vol. 2419, San Jose, Feb. 1995.

70. J. Meng and S.-F. Chang, "Tools for Compressed-Domain Video Indexing and Editing", SPIE Conference on Storage and Retrieval for Image and Video Database, San Jose, Feb. 1996.

71. J. Meng and S.-F. Chang, "CVEPS: A Compressed Video Editing and Parsing System," ACM Multimedia Conference, Boston, MA, Nov. 1996.

72. Rajiv Mehrotra and James E. Gary, "Similar-Shape Retrieval In Shape Data Management", IEEE Computer, Vol. 28, No. 9, September 1995.

73. Francois G. Meyer and Patrick Bouthemy, "Region based tracking using Affine Motion Models in Logn Image Sequences", CVGIP: Image Understanding, Vol. 60, No. 2, Spet, pp. 119-140, 1994

74. T.P. Minka and R. W. Picard, "Interactive Learning using a Society of Models", MIT Media Lab Perceptual Computing TR #349, 1996.

75. A. Moghaddamzadeh and N. Bourbakis, “A Fuzzy Region Growing Approach for Segmentation of Color Images”, Pattern Recognition, Vol 30, No 6 , pp 867-881,1997

76. F. Moscheni, F.Dufaux and M.Kunt, “A new two-stage global/local motion estimation based on a background/foreground segmentation”, IEEE Proc ICASSP’95, Detroit, MI, May 1995

77. S.K. Murthy, S. Kasif, and S. Salzberg, “A System for Induction of Oblique Decision Trees”, Journal of Artificial Intelligence Research 2:1 (1994), 1-32.

78. Akio Nagasaka and Yuzuru Tanaka,"Automatic Video Indexing and Full-Video Search for Object Appearances", Visual Database Systems II, Elsevier Science Publishers B.V.1992

79. W. Niblack, et, al, “The QBIC Project: Querying Images by Content Using Color, Texture and Shape”, Proc. IS&T/SPIE. on Storage and Retrieval for Image and Video Databases I, San Jose, CA, Februrary 1993.

80. B.C. O'Connor, “Selecting Key Frames of Moving Image Documents: A Digital Environment for Analysis and Navigation”, Microcomputers for Information Management, 8(2), pp. 119-133, 1991.

81. Michael Ortega, Yong Rui, Kaushik Chakrabarti, Sharad Mehrotra and Thomas Huang, "Supporting Similarity Queries in MARS", ACM Multimedia '97, Seattle, WA, Nov. 1997.

82. Nikhil R. Pal and Sankar K. Pal, "A review on Image Segmentation Techniques", Pattern Recognition, Vol 26, No. 9, pp1277-1294, 1993

83. Seungyup Paek and Shih-Fu Chang, "The Case for Image Classification Systems

Based on Probabilistic Reasoning", IEEE International Conference on Multimedia and Expo. July 30 - August 2, 2000. New York City, NY, USA.

84. Soo-Chang Pei and Yu-Zuong Chou, “Efficient MPEG Compressed Video Analysis Using Macroblock Type Information”, IEEE Transactions on Multimedia, Vol. 1, No. 4 Dec 1999

85. A.P. Pentland, R. W. Picard and S. Scarloff, “Photobook: tools for content-based manipulation of image database”, Proc. IS&T/SPIE. on Storage and Retrieval for Image and Video Databases II, San Jose, CA, 1994, pp.34-47.

86. E.G.M. Petrakis and C. Faloutsos, "Similarity Searching in Large Image Database", TR# 3388, Dept. of Computer Science, University of Maryland.

87. Pfeiffer, S.; Lienhart, R.; Fischer, S.; Effelsberg, *W.* “Abstracting Digital Movies Automatically”, Technical Report at the University of Mannheim, April 1996

88. R. Polana and R.C. Nelson, "Recognition of motion from temporal texture", Proceedings CVPR '92 (IEEE Computer Society Conference on Computer Vision and Pattern Recognition), Champaign, IL, June 15-18, 1992 (IEEE Computer Society Press), 129-134.

89. R.W.Picard and T.Kabir, “Finding Similar Patterns in Large Image Databases”, Proc. IEEE conference on Acoustics, Speech and Signal Processing, Minneapolis, MN, April 1993.

90. R. W.Picard, T. Kabir and F. Liu "Real-time Recognition with the entire Broads Texture Database", CVPR, 1993.

91. L. A. Rowe, J. S. Boreczky, and C. A. Eads, “Indexes for User Access to Large Video Databases”, Proc. IS&T/SPIE Conf. on Storage and Retrieval for Image and Video Databases II, San Jose, CA, 1994, pp. 150-161.

92. E. Saber, A.M. Takalp, & G. Bozdagi, "Fusion of Color and Edge Information for Improved Segmentation and Edge Linking," in IEEE ICASSP'96, Atlanta, GA, May 1996.

93. Hiroaki Sakamoto, Hideharu Suzuzi, and Akira Uemori, "Flexible Montage Retrieval for Image Data", Storage&R for I&V Databases II , SPIE Vol.2185/25

94. Philippe Salembier and Montse Pardas, "Hierarchical Morphological Segmentation for Image Sequence Coding", IEEE Transactions on Image Processing Vol 3, No. 5, Sept 1994

95. Toshio Sato, Takeo Kanade, Ellen K. Hughes, Michael A.Smith, “Video OCR for Digital News Archives”, Proc. Of the 1998 International Workshop on Content-based Access of Image and Video Database, January 3, 1998 Bombay, INDIA

96. Drew D.Saur, Yap-Pen Tan etc. “Automated Analysis and Annotation of basketball Video”, Proceedings of SPIE's Electronic Imaging conference on Storage and Retrieval for Image and Video Databases V, Feb 1997.

97. J.C.Scholtes, "Unsupervised Learning and the Information Retrieval Problem", IJCNN, Vol 1, 1991, pp95-100

98. Eero P.Simoncelli, Edward H. Adelson, David J. Heeger "Probability Distributions of Optical Flow", Proc. Conf. Comput .Vis. Patt. Recog. Maui, pp 310-315

99. J.R. Smith and S.-F. Chang, "VisualSEEk: a fully automated content-based image query system", ACM Multimedia 96, Boston, MA, Nov 20 1996.

100. J. R. Smith and S.-F. Chang, "Searching for Images and Videos on the World-Wide Web," IEEE Multimedia Magazine, 1996.

101. M. Stricker and M. Orengo, “Similarity of Color Images”, Proc. IS&T/SPIE. Conf. on Storage and Retrieval for Image and Video Databases, San Jose, CA, 1993.

102. G. Sudhir, John C.M. Lee and Anil K. Jain, “Automatic Classification of Tennis Video for High-level Content-based Retrieval”, Proc. Of the 1998 International Workshop on Content-based Access of Image and Video Database, January 3, 1998 Bombay, INDIA

103. H. Sundaram and S.F. Chang, "Efficient Video Sequence Retrieval in Large Repositories," Proc. SPIE Storage and Retrieval for Image and Video Databases VII, San Jose CA, Jan 23-29 1999.

104. H. Sundaram and S.F. Chang, “Determining Computable Scenes in Films and their Structures using Audio Visual Memory Models”, ACM Multimedia 2000, Oct 30 - Nov 3, Los Angeles, CA.

105. C.Swanberg, C.-F. Shu, and R. Jain, “Knowledge Guided Parsing in Video Databases”, Proc. IS&T/SPIE Conf. on Storage and Retrieval for Image and Video Databases, San Jose, CA, 1993.

106. Mark Tabb and Narendra Ahuja, "Multiscale Image Segmentation by Integrated Edge and Region Detection", IEEE Transactions on Image Processing, Vol 6, No. 5, May 1997

107. Daniel Toth, Til Aach, and Volker Metzler, “Illumination-Invariant Change Detection”, Proceedings of the 4th IEEE Southwest Symposium on Image Analysis and Interpretation, Austin Texas, Apr 2000

108. Alain Tremeau and Nathalie Borel, "A Region Growing and Merging Algorithm to Color Segmentation", Pattern Recognition, Vol 30, No. 7, pp 1191-1203, 1997

109. N.Ueda and K. Mase, "Tracking moving contours using energy minimizing elastic contour models", Computer Vision-ECCV'92, Vol 588, pp453-457, Springer-Verlag, 1992.

110. Luc Vincent and Pierre Soille, "Watersheds in Digital Spaces: An efficient Algorithm Based on Immersion Simulations", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol 13, No. 6, June, 1991.

111. H. Wang and S.-F. Chang, “Automatic Face Region Detection in MPEG Video

Sequences”, SPIE's Photonics China '96 - Electronic Imaging and Multimedia Systems, Beijing, China, November 1996.

112. J.Y.A. Wang and E. H. Adelson, "Representing Moving Images with Layers", IEEE Transactions on Image Processing, Vol 3, No. 5, Sept 1994.

113. J.Y.A Wang and E. H. Adelson, “Spatio-temporal segmentation of video data”, SPIE Proc Image and Video Processing II, Vol 2182, San Jose, CA, Feb 1994.

114. Y. Weiss and Edward. H. Adelson, "Perceptually organized EM: A framework for motion segmentation that combines information about form and motion", MIT media Lab Perceptual Computing TR. #315.

115. J. K. Wu and A. D. Narasimhalu, "Identifying Faces Using Multiple Retrievals", IEEE Multimedia, Summer, 1994, pp27-38.

116. Boon-Lock Yeo and Bede Liu, “On the extraction of DC Sequence From MPEG Compressed Video”, International Conference on Image Processing, Washington, D.C., USA, Oct. 1995

117. M.M. Yeung, B.L. Yeo and B. Liu, "Extracting Story Units from Long Programs for Video Browsing and Navigation", International Conference on Multimedia Computing and Systems, June 1996.

118. Minerva M. Yeung, Boon-Lock Yeo, Wayne Wolf and Bede Liu, “Video Browsing using Clustering and Scence Transitions on Compressed Sequences”, Multimedia Computing and Networking, San Jose, Feb. 1995.

119. Ramin Zabih, Justin Miller, Kevin Mai, ”A Feature-Based Algorithm for Detecting and Classifying Scene Breaks”, ACM Multimedia, 1995. pp189-200.

120. Kui Zhang, Miroslaw Bober and Josef Kittler, "Motion Based Image Segmentation for Video Coding", Proceedings of the International Conference on Image Processing, Washington (ICIP'95) Los Alamitos, CA, USA, pp.476-479, 1995.

121. H. J. Zhang and S. W. Smoliar, “Developing Power Tools for Video Indexing and Retrieval”, Proc. IS&T/SPIE. on Storage and Retrieval for Image and Video Databases II, San Jose, CA, 1994, pp.140-149.

122. HongJiang Zhang, Yihong Gong, etc. "Automatic Parsing of News Video", Proc. IEEE Int'l Conf. Multimedia Computing and Systems, IEEE Computer Society Press, Los Alamitos, Calif.,1994.

123. H. J. Zhang et al., “Automatic Parsing and Indexing of News Video”, Multimedia Systems, 2 (6), pp. 256-266, 1995.

124. H. J. Zhang, S. W. Smoliar, and J. H. Wu, “Content-Based Video Browsing Tools”, Proc. IS&T/SPIE Conf. on Multimedia Computing and Networking 1995, San Jose, CA, 1995.

125. H. J. Zhang and D. Zhong, "A scheme for visual feature based image indexing", Proc. IS&T/SPIE Conf. on Storage and Retrieval for Image and Video Databases III,

February 1995, San Jose, pp. 36-46.

126. H.J.Zhang, C.Y.Low, S.W.Smoliar and J.H.Wu "Video Parsing, Retrieval and Browsing: An Intergrated and Content-Based Solution", Proc.ACM Multimedia'95, San Francisco, Nov 5-9, 1995.

127. Di Zhong and S.-F.Chang, "Video Object Model and Segmentation for Content-Based Video Indexing", ISCAS'97, HongKong, June 9-12, 1997.

128. Di Zhong and S.-F.Chang, "Spatio-Temporal Video Search Using the Object Based Video Representation", ICIP'97, October 26-29, 1997 Santa Barbara, CA.

129. Di Zhong, H. J. Zhang and S.-F.Chang, "Clustering methods for video browsing and annotation", Storage & Retrieval for Still Image and Video Databases IV, IS&T/SPIE's Electronic Imaging: Science & Technology, Feb. 96.

130. Di Zhong and Shih-Fu Chang, "AMOS - An Active MPEG-4 Video Object Segmentation System", ICIP-98, Chicago, Oct. 1998.