Video Scene Decomposition with the Motion Picture Parser[1]

E. Deardorff, T.D.C. Little, J.D. Marshall, D. Venkatesh, and R. Walzer

Multimedia Communications Laboratory
Department of Electrical, Computer and Systems Engineering
Boston University, Boston, Massachusetts 02215, USA
(617) 353-9877, (617) 353-6440 fax
tdcl@bu.edu

MCL Technical Report 01-10-1994

Abstract–A motion picture can be modeled as a composition of many scenes where each scene is comprised of multiple shots. Thus, a conventional movie is a sequential aggregation of a large number of disparate image sequences. Within each image sequence or shot, there is consistency in image content and dynamics. This consistency in dynamics can be used in identifying scene changes for video segment decomposition and for techniques to improve data compression.

We have developed an algorithm to use these dynamics for scene change detection and the decomposition of video streams into constituent logical shots. The algorithm uses intraframe image complexity and identifies scene transitions by considering short-term temporal dynamics. The algorithm has shown to be effective for detecting both abrupt scene changes (cuts) as well as smooth scene changes (fades and dissolves).

This algorithm is used in an application we have developed called the Motion Picture Parser (MPP). The MPP automates the process of tagging segments of motion-JPEG-compressed movies. Segments are also tagged for subsequent semantic content-based retrieval in units of shots and scenes. The MPP application consists of a graphical user interface with various editing controls.

Keywords: Scene change detection, movie parsing, video segmentation.

[1]In *Proc. IS&T/SPIE Symposium on Electronic Imaging Science and Technology (Digital Video Compression and Processing on Personal Computers: Algorithms and Technologies)*, San Jose, February 1994, SPIE Vol. 2187, pp. 44-55. This material is based upon work supported in part by the National Science Foundation under Grant No. IRI-9211165.

1 Introduction

Information delivery systems of the future will allow their users powerful interface and presentation flexibility. These systems will allow the viewing preferences of each user to be tailored and adapted to the available information space (e.g., one user's interest in stock trading, another's interest in children's educational programming). We envision mechanisms that provide filtering of both live broadcasts and prerecorded programming offered by video-on-demand services or other news/information sources. Such a system will require both the ability to understand a user's preferences and to filter the realm of available information.

In order to provide this functionality, a number of technical obstacles must be overcome. The challenges exist in the development of the query interface/mechanism, the query evaluation engine, and the model used to capture domain-specific semantic data. Furthermore, the query engine must be able to resolve queries in different formats (e.g., by sound, by image example, by text input), and must evaluate across multiple domain-specific semantic models (e.g., newsclip databases, personnel databases). With respect to image and video databases, two schemes appear to be evolving to achieve the content-based retrieval component. The first approach requires preprocessing the contents of the database to glean domain-specific information into readily-machine-recognizable tokens (text) for indexing and subsequent query. However, this approach cannot anticipate all possible types of queries that might be applied to the database. The second approach is dynamic. Queries are formulated in mixed formats and applied to database as a whole rather than to the pre-extracted indices. For audio, image, and video databases, both approaches require significant use of image/signal processing. For the latter approach, flexibility is traded-off for performance. A more reasonable solution would provide a balance between a priori content-skimming and dynamic content identification from the objects in the database.

In this paper we make several simplifications to this problem. First, we restrict ourselves to a single domain-specific application model (i.e., motion pictures). Second, we use a preprocessing approach to content extraction that limits the universe of subsequent queries. Third, we use a text-based query engine (i.e., we maintain semantic content in the form of text-based tokens). The preprocessing of the database contents can be simplified as well. For our application domain, motion pictures, a great deal of information can be identified without analysis of the audio/video stream. This information consists of the producer's name, actor names, etc. that are available in text form. Furthermore, access to closed-captioning can provide a running text-based representation of the dialog occurring on the soundtrack. Preprocessing can be further simplified if it involves the authoring or scripting

activities associated with movie production. That is, the content of the movie is associated with the scenes of the movie rather than being extracted later. For our application we use a semi-automated approach to movie component indexing and content assignment. First, all text-based application-specific data are collected and organized for a movie in a fixed database schema. The movie is then parsed into components which are then assigned keywords and descriptive phrases to facilitate static content-based retrieval.

The essence of this paper is the application developed to support this process. The application, called the Motion Picture Parser (MPP), parses movies into segments by identifying scene and shot boundaries. The motivation for developing the MPP was to simplify the process of populating the database used for our video-on-demand (VOD) application called the Virtual Video Browser (VVB) [7]. The VVB is a true-VOD application that permits an individual to browse an electronic video collection. Movies and scenes of movies can be identified and viewed through the VVB based on movie-specific attributes including actor names, director names, and scene characteristics. Movie and scene content indexing are facilitated by summary, keyword, and transcript searching. In addition to viewing scenes from the movies, the user can view all of a movie's textual information including summaries and transcripts.

Research related to this effort includes modeling of unstructured data for content-based retrieval [12, 14, 20, 21, 22, 24], modeling of the temporal component of multimedia data [5, 8], and modeling of application-specific multimedia data (movies) [4, 10, 11, 16, 17, 18]. Matthews et al. [13] describe a prototype video editing tool called the "VideoScheme." This system is simple and does not provide extensive functionality beyond the basic editing functions of cutting and merging different video clips. There is very little context information that is retrieved in this system. Otsuji and Tonomura [15], Zhang et al. [25] and Arman et al. [1] discuss schemes for identifying scene changes using image processing techniques. A similar approach has been applied by Swanberg et al. [21, 22] and MacNeil [12]. In the former work, application domain-specific models are used to capture extracted information and to support content-based retrieval. In Steven's work in the ALT project at CMU [20], knowledge of content, structure, and use of the content is embedded in video objects. This approach is applied to the generation of video sequences using a rule base and leads to the selection of seamless concatenation of video sequences from many small video shots.

The remainder of this paper is organized as follows. In Section 2 we describe a model for digital movies. In Section 3 we describe the image dynamics for compressed video and their application in identifying scene transitions. In Section 4 we describe the working of

the MPP. In Section 5 the MPP and proposed future enhancements are discussed. Section 6 concludes the paper.

2 Modeling of Motion Pictures

In a conventional database management system (DBMS), access to data is based on distinct attributes for well-defined data developed for a specific application. For unstructured data such as audio, video, or graphics, similar attributes can be defined. However, the information contained in the data can be diverse and difficult to characterize within a single application [24]. Two primary problems must be solved in order to provide content-based retrieval for unstructured data of these types. First, a means of extracting information contained in the unstructured data (e.g., an image) is required, and second, this information must be appropriately modeled in order to support both user queries for content and data models for storage.

The first problem can be approached by segmentation and feature extraction. Content of unstructured data such as imagery or sound is easily identified by human observation; however, few attributes lead to machine identification. For still and moving images, it is possible to apply segmentation to yield features such as shape, texture, color, relationships among objects, and events [3]. This process is very difficult to apply to complex scenes; however, it can be simplified with *site models* which characterize images a priori and allow detection of scene changes. The second problem of managing the diverse query types and data models is simplified by semantic nets.

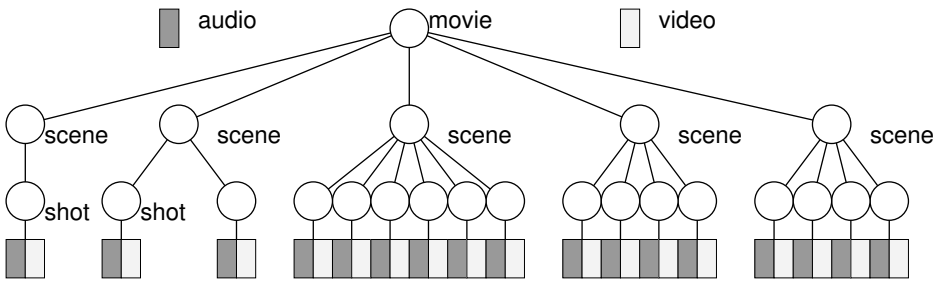

Figure 1: Temporal Schema For a Motion Picture

From a structural perspective, a motion picture modeled as data consists of a finite-length of synchronized audio and still images. This model is a simple instance of more general models for heterogeneous multimedia data objects. Davenport et al. [4] describe the

fundamental film component as the *shot*: a contiguously recorded audio/image sequence. To this basic component, attributes such as content, perspective, and context can be assigned, and later used to formulating specific queries on a collection of shots. Such a model is appropriate for providing multiple views on the final data schema and has been suggested by Lippman and Bender [6], Bender et al. [2], and by Loeb [10, 11]. The model that we use for motion pictures consists of a hierarchy of shots and scenes as illustrated in Fig. 1. This temporal model is also associated with a more application-specific one that captures movie names, actors, etc.

Aguierre Smith and Davenport [17, 18] use a technique called *stratification* for aggregating collections of shots by contextual descriptions called *strata*. These strata provide access to frames over a temporal span rather than to individual frames or shot endpoints. This technique can then be used primarily for editing and creating movies from source shots. The current implementation of this work requires manual semantic analysis to generate the contextual information.

For the VVB application we currently use manual feature extraction of well-defined attributes that fit into a fixed data schema.

3 Video Scene Dynamics

The data comprising a video stream can be modeled as a sequence of still frames that are displayed sequentially. The digital still frames are comprised of pixels. A 640x480 still contains 307,200 such pixels. If pixels were used to compare successive frames for the purpose of detecting scene transitions,[2] from the resulting process would be extremely slow and requires significant computing power. However, when a compression scheme such as JPEG [23] is applied to the frames, a significant amount of information about the shots in the frames can be extracted.[3] This is due to short-term temporal consistency within the frames of a shot. Because there is little change in the overall image content of a shot, the compressed frames of the shot have similar size (in bytes)i.e. frames within a single scene exhibit consistent intraframe image complexity. Therefore, by studying the file size dynamics of the compressed frames, we can find transitions from one shot to the next. However, a filtering scheme based purely on size will fail when two shots of similar complexity follow

[2]We use "scene transition" to refer to any change (scene or shot) in the video data stream.

[3]Although we use the JPEG compression scheme, the results of the scene transition identification can be applied to MPEG compression since we can readily covert JPEG-compressed data to a MPEG-compressed format.

each other. It may also detect erroneous changes when none exist. When such situations occur, more complex algorithms must be employed. The image size dynamics for two sample JPEG-compressed video sequences are shown in Fig. 2.

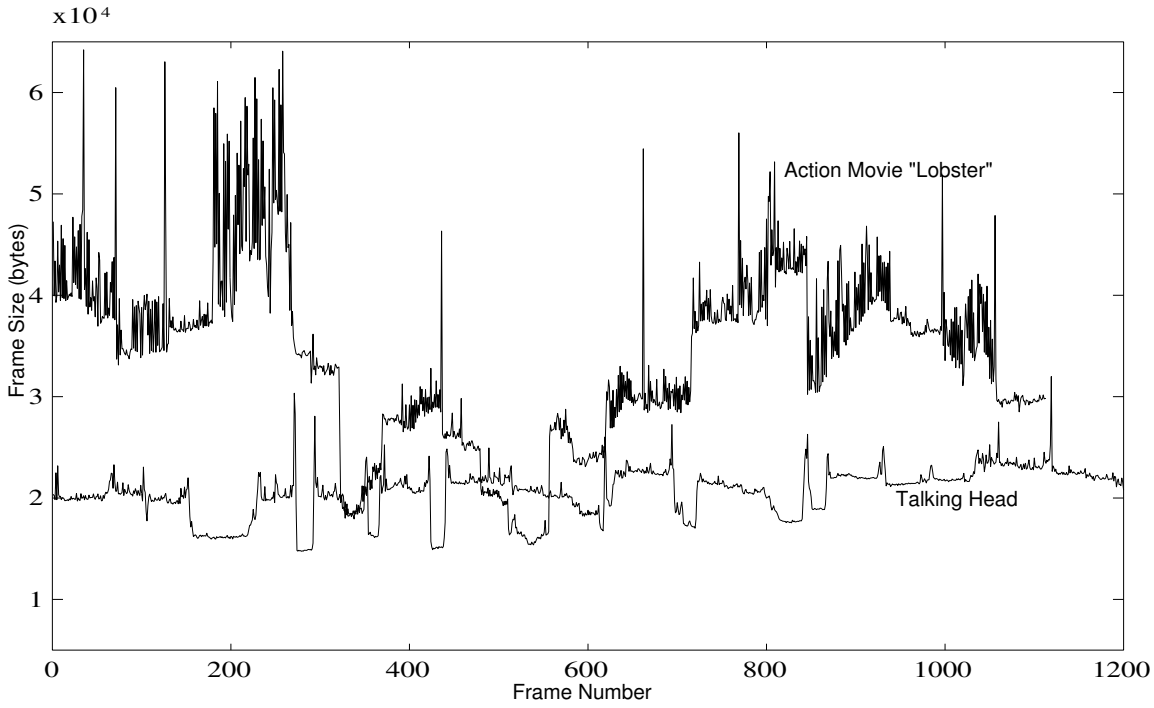

Figure 2: Image Size Dynamics for JPEG-Compressed Movies

Fig. 2 illustrates the differences in the characteristics of compressed video for two different videos sequences. The more dynamic (bursty) sequence is from a movie clip with significant action (motion and complex image sequences shot over water). The second, less bursty sequence is from a video-taped classroom and is representative of "talking head" video dynamics. This sequence has little motion as compared to the action movie. Consequently, the scene and shot transitions are more easily identifiable in the first case, where changes are abrupt and quite prominent. Two different kinds of scene changes are illustrated in Fig. 3. The first is a "merge," "fade," or a "dissolve" where the images are superimposed over each other at a scene boundary. These yield a composition of two images at the scene transition which is difficult to compress, and result in a spike in the compressed frame size plot. The second type of change is a "cut." In this case, an abrupt change from one frame to the next due to the cut causes a positive or negative step in the compressed frame size plot.

Currently, the MPP uses a scheme which estimates shot changes based on shifts in successive compressed frame sizes. The **Scene Transition Detection Algorithm** used for

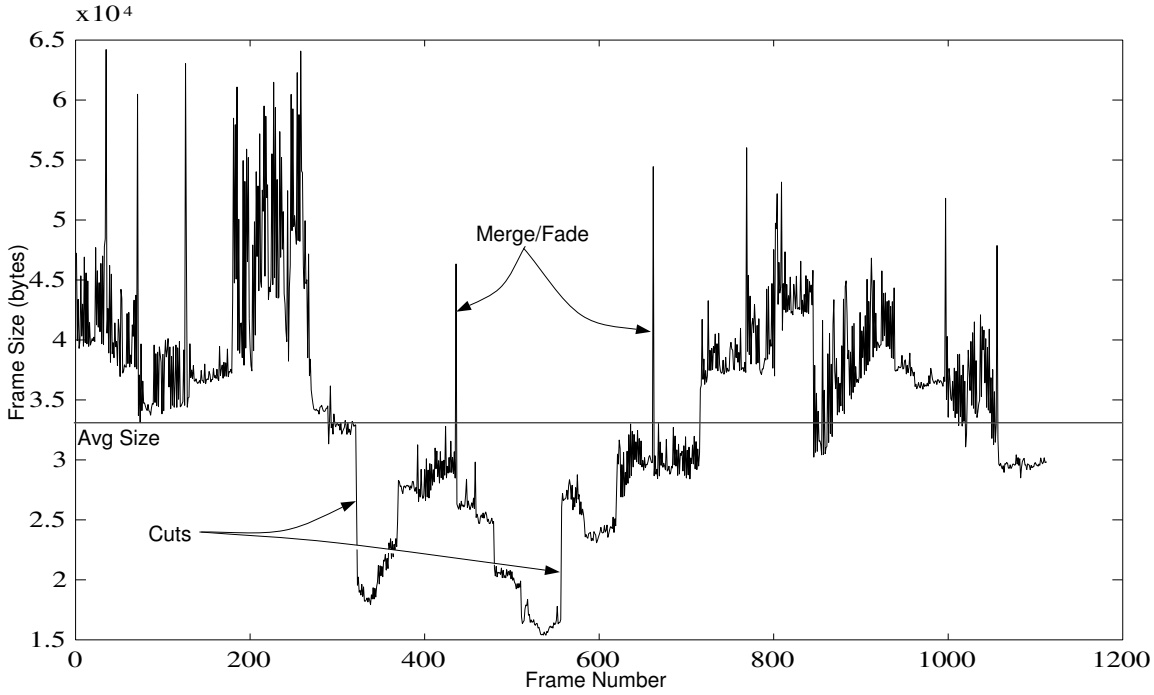

Figure 3: Illustration of Various Scene Transitions

this purpose is summarized below.

1. while $i < last\_frame\_no$ do
   (a) $forward\_increase = frame\_size[i+1] - frame\_size[i]$
   (b) $backward\_increase = frame\_size[i] - frame\_size[i-1]$
   (c) if $|forward\_increase| > \Delta$ and $|backward\_increase| > \Delta$ then
      - {flag as possible transition due to fade/merge/dissolve}
   (d) else if $|forward\_increase| > \Delta$ or $|backward\_increase| > \Delta$
      - {flag as possible transition due to cut}
   (e) end if
2. end while

This algorithm operates by comparing the changes in frame size around sets of three consecutive frames. If a spike is detected, it is flagged as a fade/merge/dissolve. Otherwise, a step change indicates a cut. The sensitivity parameter of the algorithm (Δ) is selectable by

the MPP user interface. In the current implementation, this parameter is manipulated by the user. We are examining techniques to dynamically adapt the sensitivity to the characteristics of the video data stream. The algorithm has shown to be quite robust usably accurate.

Related work in scene change detection includes the work by Zhang et al. [25], Otsuji and Tonomura [15], and Arman et al. [1]. The former schemes [15, 25] require a pixel by pixel comparison between frames and are therefore computationally intensive. The latter scheme [1] uses DCT (Discrete Cosine Transform) coefficients to estimate scene transitions and is computationally less complex. These fine-tuning algorithms can easily be included in the MPP for cases for which our scheme fails. A uniform approach for detecting scene transitions cannot be applied to all videos. However, by specifying a smaller search space for these algorithms, the MPP can speed up the process of decomposing long shots of video.

4 The Motion Picture Parser

The Motion Picture Parser is intended to partially automate the process for entering data into a database used by our Virtual Video Browser application. It is a standalone application, operating independently of the VVB. The MPP generates an intermediate text file that can be applied in the population of the VVB database.[4] The basic functions of the MPP are scene and shot identification. A movie consists of one or more scenes. Each scene has one or more shots. Shots are comprised of one or more frames. The frames which correspond to a shot must be identified, as must the shots which correspond to a scene. The MPP achieves its overall functionality by the following operations.

1. display_movie – displays a selected JPEG-compressed (interleaved audio and video) video file to the user. It supports VCR functionality such as play, fast-forward, rewind, still-frame, and jump. It uses find_scenes_and_shots to locate scene and shot changes.

2. find_scenes_and_shots – automates the process of determining where scene and shot changes occur in the digital video. It communicates with the user to verify its selections and offers the user the opportunity to refine selections.

3. tag_shot/scene For each scene and shot in the movie, associate relationships with previous and subsequent shots/scenes. Identify and index keywords (manual).

[4]We use the POSTGRES relational database management system [19].

4. build_shots_file – creates an intermediate ASCII file known as the `shots.txt` file based on the input received from find_scenes_and_shots.

5. generate_temporal_schema – processes the `shots.txt` file into a temporal schema file (`postgres.ts`) and a schema interface file (`postgres.si`).

6. generate_shot_files – processes the `shots.txt` file and extracts the individual shots from the original JPEG movie file into individual files.

In the remainder of this section we describe these activities in detail.

4.1 Scene and Shot Decomposition

The MPP interface consists of two menus: a text based menu for entering processing commands and a graphical interface for video editing. The initial input to the MPP consists the filename of a selected JPEG movie file (`movie_name`). Other inputs include the selection of frame numbers belonging to shots and the selection of shots belonging to scenes.

When identify_shots_and_scenes is selected from the MPP main menu, a device control panel and movie window appear as shown in Fig. 4. This function displays the digital video movie to the user and allows VCR functions of play, fast-forward, rewind, still-frame and jump. This function creates an intermediate ASCII file called `shots.txt` based on input received from the user.

When the initialize button is pressed, the first frame of the selected movie is displayed in the movie viewer window. The panel can be used to control any editing and movie display. The following controls are are used specifically for scene and shot identification (in addition to the VCR controls).

- Mark Scene
- Mark Start Shot
- Mark End Shot
- Find Next Shot
- Go To Frame #

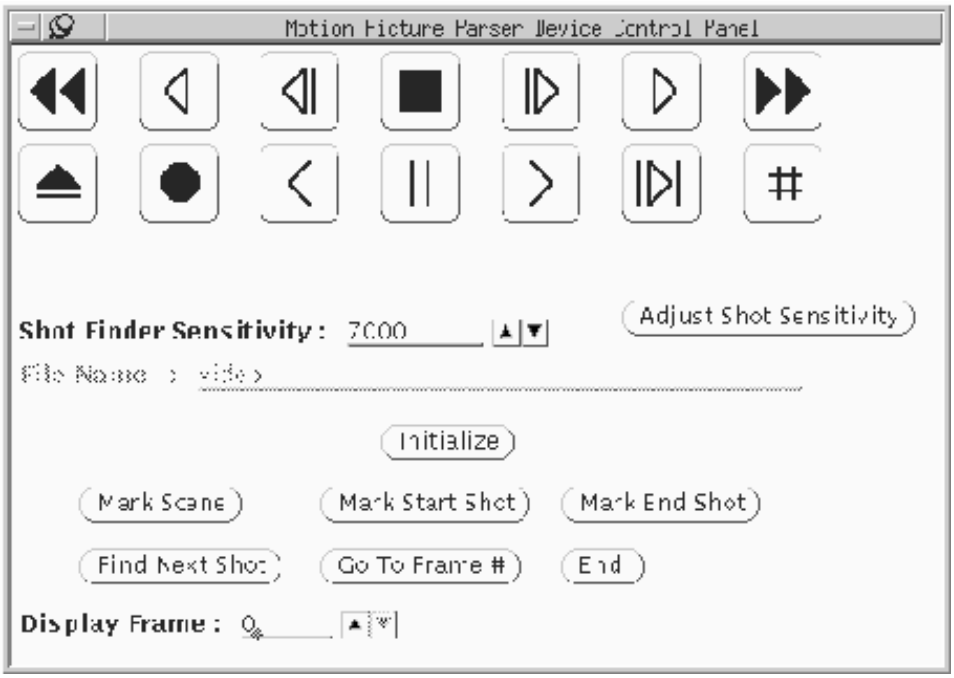

Figure 4: The MPP Device Control Panel

- Adjust Shot Sensitivity
- End

These controls are outlined as follows.

1. Mark Scene is used to indicate the start of a new scene. After the video is displayed, to begin the shot and scene selection process, Mark Scene must be selected. Once Mark Scene is activated, Start and End Shots selections for the scene can be entered. A new scene selection is started by simply depressing the Mark Scene button.

2. Mark Start Shot identifies the start of a new shot. After this selection is initiated, the application waits to receive a Mark End Shot selection. If Mark End Shot is not selected and Mark Start Shot is reselected, the previous Mark Start Shot selection is lost and the current selection is used for to mark the start of a shot selection.

3. Mark End Shot identifies the end of a shot selection. After the end of a shot is marked, Mark Scene or Mark Start Shot can be selected. If Mark End Shot is selected and Mark Start Shot has not been selected, then an error message is displayed. If the location of

the selected end of shot is before the location of the selected start of the shot, then an error message is displayed. One of the following operations must then be selected.

   (a) Select a new shot start location. A new start position for the shot can be selected by depressing the Mark Start Shot button.

   (b) Find an end shot location after the start shot location. Advance the movie to an end position after the selected start position by depressing the Fast Forward control button or using the Go To Frame option. Then select Mark End Shot.

4. The Find Next Shot button advances the movie to the next shot. The location of the next shot is determined by the location of the displayed image. The movie is advanced forward to the estimated next shot and the first frame of the next shot is displayed. This function uses the algorithm specified in the earlier sections to find the location of the next shot.

5. Go To Frame and Display Frame options advance the movie playback to a desired (valid) frame number. A valid frame number is in the range of zero to the last frame of the movie.

6. Adjust Shot Sensitivity changes the shot finder sensitivity value for the shot identification routine. The minimum, default, and maximum sensitivity values are $1,000$, $7,000$ and $100,000$, (bytes) respectively. Since these values represent the difference in frame sizes for a JPEG movie file, smaller values are more sensitive to differences. The shot sensitivity can be changed by entering a new value via text input or via an increment button. Once the shot change estimates are complete, the first frame of the movie is redisplayed.

7. End. This option quits the session.

The sequence for identifying a shot is to (1) load the movie, (2) view it via the movie viewing window, and (3) isolate a specific shot. When the desired start of a shot is displayed in the movie viewer, depressing the Still control button on the device panel pauses the movie playback. At this point the scene transition can be explored by forward or backward stepping through the frame sequence. After the desired cut point is marked with Mark Start Shot, the next transition point (in the forward direction) can be identified. After a similar transition point exploration, the end of shot can be marked with Mark End Shot. After all such shots have been isolated they are recorded as frame numbers to `shots.txt`. The form of the `shots.txt` file is shown below.

```
--------------------
Scene
         0         493
       493         817
       818         927
       928        1128
      1086        1211
--------------------
Scene
      1212        1306
      1307        1419
      1420        1602
      1603        1765
--------------------
Scene
      1752        1795
      1796        2005
--------------------
Scene
      2005        2239
      2240        2446
      2466        2561
      2561        2827
--------------------
Scene
      2828        3024
      3025        3098
      3099        3215
      3216        3799
      3800        3889
      3890        4653
--------------------
```

4.2 Copying and Decomposing Movie Segments

When Copy video from movie to shots is selected from the main menu of the MPP, the copy operation begins parsing the scene and shot data collected by the Identify_Scenes_and_Shots function described in the previous section. This function processes the `shots.txt` file to extract the individual scenes and shots from the original digital video file into separate files. The extraction and copying of the digital video to files is a batch-mode operation. Generation of the file names of the new JPEG movie files is is performed automatically. This batch process take two inputs:

1. The shots file, `shots.txt`, which lists the frame numbers of each shot and scene.
2. The video file of the movie.

For each shot in the shots file the copying and decomposition function creates an appropriate file name, extracts from the video file the frames which correspond to this shot, and stores these frames in the created file. The user does not interact with this operation. However, several useful status messages are displayed to the user as the operation progresses. These messages are displayed to the terminal from which the MPP is invoked. A sample of these messages is included below.

- scene_num indicates the scene number being parsed.
- shot_num indicates the shot number within the scene being parsed.
- time indicates the duration of the shot in seconds.[5]
- total indicates the total number of frames being extracted from the video file.

This last step of extracting the video frames is disk access intensive and can take quite some time depending on the total number of frames being extracted. When the extraction of each shot is complete, an appropriate message is generated. When all of the scenes and shots have been extracted, the "Copy Operation Complete!" message is displayed. If any of the scene or shots exist as files, Copy will detect that the JPEG shot video files have already been extracted and the program will terminate.

[5]The timing is applied to the temporal schema of the VVB database.

4.3 Generating the Temporal Schema and DBMS Population

The generate_temporal_schema processes the `shots.txt` file into POSTGRES-compatible database population scripts. This function builds a temporal schema file (`postgres.ts`) and a schema interface file (`postgres.si`). Details of the database schemata and relations are described elsewhere [7]. An example of the output produced by this process for the movie "Dr. No" is shown below (partial listing of `postgres.ts`).

```
/* nodes for Dr. No */
append node(
           node_id         = 001,
           node_type       = "non-terminal",
           duration         = 7200.00,
           title            = "movie-Dr. No"
           )

/* subnodes for Dr. No */
append subnode(
              parent_id       = 001,
              child_id        = 002,
              index_f         = 1,
              index_r         = 3,
              delay_f         = 0,
              delay_r         = 0,
              )

/* terminals for Dr. No */
append terminal(
               node_id         = 005,
               medium           = "video",
               encoding         = "JPEG",
               filename         = "Dr_No_term1"
               )
```

5 Discussion and Future Work

The MPP system was developed on a Unix workstation equipped with a video capture and compression card manufactured by Parallax Graphics, Inc. The user interface for the MPP runs within the X11 window system under the Motif window manager. The MPP has been tested using different video segments with varied image dynamics. The MPP is an easy application to use and is very efficient in comparison to the manual movie decomposition and indexing that we have performed for the VVB.

We are pursuing a number of extensions to this work to (1) facilitate more interesting content extraction and (2) to apply the scene dynamics and decomposition to bandwidth management in a VOD system. For the first item, we are investigating the use of the audio domain to provide speaker identification and to augment the detection of transitions. The second item is motivated by the desire to control the quality of audio and video data delivery in the path from storage, across a network, and to the user. By having a priori information about the characteristics of the delivered data, the system as whole can better allocate its resources among many connections and users. At a different scale, the relative bandwidth requirements of shots (or scenes) can be used in allocating resources to guarantee smooth, gap-free playout throughout their duration. Transitions points can be used to introduce small timing adjustments that cannot be tolerated during shot delivery. Finally, we are investigating the use of the scene transitions as a means for optimally placing standalone frames (I frames) for the MPEG compression scheme.

6 Conclusion

In this paper we have described a prototype application for decomposing compressed video data sequences for indexing purposes. The application, called the Motion Picture Parser, takes a video file as input and through a semi-automated process, produces script files for populating the database of our VOD application. It also produces individual files corresponding to the decomposed shots of the original video file. Our experience with the MPP has shown it to be a valuable tool. However, significant research is necessary to achieve the objective of automating the process of video content extraction for database indexing.

References

[1] F. Arman, A. Hsu, and M-.Y. Chiu, "Image Processing on Compressed Data for Large Video Databases," *Proceedings of the 1st ACM International Conference on Multimedia (ACM Multimedia'93)*, Anaheim CA, August 1993, pp. 267-272.

[2] W. Bender, H. Lie, J. Orwant, L. Teodosio, and N. Abramson, "Newspace: Mass Media and Personal Computing," *Proc. Summer 1991 Usenix Conf.*, Nashville, Tennessee, June 1991, pp. 329-348.

[3] M.J. Carlotto, "Image Understanding Now and in the Future," *Advanced Imaging*, Vol. 7, No. 8, August 1992, p. 26.

[4] G. Davenport, T.G. Aguierre Smith, and N. Pincever, "Cinematic Primitives for Multimedia," *IEEE Computer Graphics & Applications*, July 1991, pp. 67-74.

[5] R.G. Herrtwich, "Time Capsules: An Abstraction for Access to Continuous-Media Data," *Proc. 11th Real-Time Systems Symp.*, Lake Buena Vista, FL, December 1990, pp. 11-20.

[6] A. Lippman and W. Bender, "News and Movies in the 50 Megabit Living Room," *Globecom '87*, Tokyo, Japan, November 1987, pp. 1976-1981.

[7] T.D.C. Little, G. Ahanger, R.J. Folz, J.F. Gibbon, F.W. Reeve, D.H. Schelleng, and D. Venkatesh, "A Digital Video-on-Demand Service Supporting Content-Based Queries," *Proceedings of the 1st ACM International Conference on Multimedia (ACM Multimedia'93)*, Anaheim CA, August 1993, pp. 427-436.

[8] T.D.C. Little and A. Ghafoor, "Interval-Based Temporal Models for Time-Dependent Multimedia Data," *IEEE Transactions on Data and Knowledge Engineering*, Vol. 5 No. 4, August 1993, pp. 551-563.

[9] T.D.C. Little and A. Ghafoor, "Synchronization and Storage Models for Multimedia Objects," *IEEE J. on Selected Areas in Comm.*, April 1990, pp. 413-427.

[10] S. Loeb, "Delivering Interactive Multimedia Documents over Networks," *IEEE Communications*, Vol. 30, No. 5, May 1992, pp. 52-59.

[11] S. Loeb, R. Hill, and T. Brinck, "Lessons from LyricTime: A Prototype Multimedia System," *Proc. 4th IEEE ComSoc Intl. Workshop on Multimedia Communications (Multimedia '92)*, Monterey, CA, April 1992, pp. 106-113.

[12] R. MacNeil, "Generating Multimedia Presentations Automatically using TYRO, the Constraint, Case-Based Designer's Apprentice," *Proc. 1991 IEEE Workshop on Visual Languages*, Kobe, Japan, October 1991, pp. 74-79.

[13] J. Matthews, P. Gloor, and F. Makedon, "VideoScheme: A Programmable Video Editing System for Automation and Media Recognition," *Proceedings of the 1st ACM International Conference on Multimedia (ACM Multimedia'93)*, Anaheim CA, August 1993, pp. 419-426.

[14] M.H. O'Docherty and C.N. Daskalakis, "Multimedia Information Systems - The Management and Semantic Retrieval of All Electric Data Types," *The Computer Journal*, Vol. 34, No. 3, June 1991, pp. 225-238.

[15] K. Otsuji and Y. Tonomura, "Projection Detecting Filter for Video Cut Detection," *Proceedings of the 1st ACM International Conference on Multimedia (ACM Multimedia'93)*, Anaheim CA, August 1993, pp. 251-257.

[16] L.A. Rowe and B.C. Smith, "A Continuous Media Player," *Proc. 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, La Jolla, CA, November 1992.

[17] T.G. Aguierre Smith and G. Davenport, "The Stratification System: A Design Environment for Random Access Video," *Proc. 3nd International Workshop on Network and Operating System Support for Digital Audio and Video*, La Jolla, CA, November 1992.

[18] T.G. Aguierre Smith and N.C. Pincever, "Parsing Movies in Context," *Proc. Summer 1991 Usenix Conf.*, Nashville, Tennessee, June 1991, pp. 157-168.

[19] M. Stonebraker and G. Kemnitz, "The POSTGRES Next Generation Database Management System," *Communications of the ACM*, Vol. 34, No. 10, October 1991, pp. 78-92.

[20] S.M. Stevens, "Embedding Knowledge in Continuous Time Media," *Proc. 2nd Intl. Workshop on Network and Operating Support for Digital Audio and Video*, Heidelberg, Germany, November 1991.

[21] D. Swanberg, C.F. Shu, and R. Jain, "Architecture of a Multimedia Information System For Content-Based Retrieval," *Proc. 3rd International Workshop on Network and Operating System Support for Digital Audio and Video*, La Jolla, CA, November 1992.

[22] D. Swanberg, T. Weymouth, and R. Jain, "Domain Information Model: An Extended Data Model for Insertions and Query," *Proc. 1992 Workshop on Multimedia Information Systems*, Tempe, Arizona, February, 1992, pp. 39-51.

[23] G.K. Wallace, "The JPEG Still Picture Compression Standard," *Communications of the ACM*, Vol. 34, No. 4, April 1991, pp. 30-44.

[24] T.M. Wittenburg, "A Reconfigurable Multimedia Document Management System Based on an Extended Object-Oriented Data Model," *M.S. Thesis*, Dept. of Electrical, Systems and Computer Engineering, Boston University, December 1992.

[25] H.J. Zhang, A. Kankanhalli, and S.W. Smoliar, "Automatic Partitioning of Full-Motion Video," *Multimedia Systems*, Vol. 1, No. 1, 1993, pp. 10-28.