Multimedia structuring using trees

George Tzanetakis & Luc Julia
Computer–Human Interaction Center (CHIC!)
SRI International
333 Ravenswood Avenue
Menlo Park, CA 94025
gtzan@cs.princeton.edu julia@speech.sri.com

Abstract

Traditionally work on multimedia structuring has been centered on the creation of indices and their use for searching. Although searching is important there are many cases where the user just wants to browse through the data to find something interesting without having any particular search goal. Multimedia data exhibits hierarchical structure that can be exploited for more natural user interaction with the content.

In order to handle the large amounts of multimedia data more structure than what is currently available is required. In this paper, we have focused on structuring multimedia data using trees to describe both temporal and categorical relations. The pervasive use of trees to express hierarchies facilitates browsing, profiling, and authoring. Our main target application is the implementation of a personalized TV–guide. The constraints imposed by this application caused the development of a new simple compact graphical user interface for tree browsing.

Introduction

There is a huge increase in the amount of multimedia data on the Web and elsewhere. This increase will continue as the television, video, and music industries gradually become digital. More advanced tools than what those which are currently available are required to handle this data efficiently. Structuring the information for browsing and retrieval is an important part of this process. File systems with directories were introduced in the early days of computing as a way of organizing the increasing amounts of computer files. In a similar manner trees are representations that can be used to hierarchically organize temporal multimedia data streams like video and audio.

In this paper, we propose the use of trees as data representations that can be used for better browsing and analysis of temporal data streams. Multimedia data exhibits hierarchical structure at different time granularities. For example, a news broadcast is organized in sections, each section has different topics and each topic several scene changes or shots. As another example, music exhibits hierarchical structure at different time granularities. In our system the hierarchical structure of trees is used to express temporal and categorical relations and allows for more natural browsing, user–profiling and searching than the traditional tape–recorder user interface paradigm especially for large data sets. Traditionally, work on multimedia data has been centered on the creation of indices and their use for searching. Although searching is important there are many cases where the user just wants to browse through the data to find something interesting without having any particular search goal. In our system trees are not only used internally but are exposed and used extensively both by the multimedia author and by the end user.

A prototype implementation of a system for authoring and viewing trees containing multimedia data is described. It is implemented within the Open Agent Architecture (OAA) (Martin *et al*, 1999), a distributed multiagent framework that enables rapid integration of component technologies. Issues related to the creation, presentation and maintenance of this representation are discussed. As an application of the system we describe a personalized TV guide. A novel graphical user interface for tree viewing on a TV screen using the discrete keys of a remote control has been developed.

Motivation

The system is general and can be used for any multimedia structuring application. However the driving motivation behind it is the creation of a personalized TV guide. In our scenario, a TV provider creates a tree representation for the daily program. This authoring process is facilitated by the use of trees. By means of a user profile specific parts of the daily TV program are recorded digitally. For example a user might select to record only news and action movies. Once the profile is created, the specific instances of the daily program that fit the specified profile are recorded for subsequent browsing and playback. The end–user interacts with the data using trees both for profiling and for browsing.

This wide use of trees has helped to unify the interaction with the user and led to the creation of a simple compact graphical user interface that works using limited screen space and simple discrete buttons of a remote control. Trees also allow different levels of interaction with the user. In profiling a user might choose to be very specific and record only science fiction action movies or another one might be more general and record all movies. In browsing a user might want to view the whole news broadcast or another might choose to see only a specific topic. By means of the hierarchical structure of the trees all these operations are equally easy to do and require only five buttons of a remote control and limited screen space. In this paper the term *user* refers to the person browsing the tree and the term *author* refers to the person creating the tree.

Related Work

The evolution of video/audio browsing can be viewed as an increase in the structure of the underlying representation. Initially, the tape–recorder paradigm using fast forward and rewind was the only means to browse quickly through a video sequence. Currently most analysis systems support the notion of a single time line as a basis for non–linear skipping. Our system offers the ability of non–linear skipping and zooming.

The effectiveness of intelligent time skipping has been demonstrated in SpeechSkimmer (Arons, 1998), where the user can audition spoken documents at several times real time, using time–compression techniques and segmentation based on pitch. Similarly, there are commercial products like Virage (Virage, 2000) that use segmentation to a single time line for video browsing and retrieval. Marsyas (Tzanetakis & Cook, 2000) is a music browser based on multiple feature automatic segmentation and classification. In (Chiu & Wilcox, 1998) a method for automatically building a tree using agglomerative clustering for structuring Ink and Audio notes is described. A review of multimedia authoring tools is given in (Bulterman & Hardman, 1995). The Amsterdam Hypermedia Model (Hardman et al, 1993) describes extending hypertext to support real multimedia. In this model structures very similar to those described in this paper are used for authoring multimedia presentations. Finally our novel graphical user interface for tree browsing is related to work in alternative tree browsers like (Johnson & Shneiderman, 1991).

This work builds upon experience gained with the development of two systems for multimedia analysis at SRI. MVIEWS (Cheyer & Julia,1998) is a system for annotating, indexing, extracting, and disseminating information from video streams for surveillance and intelligence applications. MAESTRO (Rivlin et al, 2000) uses multiple time lines resulting from different analysis techniques, in a display similar to a music score for content–based indexing, archiving, and retrieval of video. Unlike MAESTRO, where each time line is kept separate we attempt to combine them in one hierarchical structure. All the analysis agents developed for these applications can be used by our system. In addition it is also easy to incorporate the results of external analysis tools.

The important features of our system are the following: (1) pervasive use of trees at all stages of authoring, profiling and browsing, (2) integration of automatic and manual authoring in one semi–automatic environment and (3) a new graphical user interface for tree browsing.

Architecture

In this section we define some of the basic concepts underlying the architecture of our system in order to define the terminology for the rest of the paper. These concepts correspond to objects in our system and are stored and communicated between the various components using OAA. These are:

- ***TimeRegion*** is the basic entity of the architecture. Each node of the *TimeTree* is a *TimeRegion* and it corresponds to a segment in time of a multimedia stream. The basic fields are a symbolic name, a link to the external multimedia file(s) together with starting and ending times and finally a *class_id* which provides the link from *TimeTree* to *ClassTree*. Basically the *class_id* corresponds to the type of the segment. For example, a particular segment of a news broadcast would have type *News*. As will be explained in the *ClassTree* definition, in reality more information is carried with the *class_id.* There are also other additional fields that are not important for viewing and interacting with the tree. Finally it is easy to add new fields without affecting the rest of the system.
- ***TimeTree*** is a tree where each node is a *TimeRegion.* There is no restriction on the ordering of the nodes although typically the lower levels will be sorted based on time and the higher levels based on *class_id*. This tree is the center of the system and it is used for authoring and browsing. The playback for an inner node corresponds to playing sequentially all the leafs of its subtree.
- ***ClassEntry*** corresponds to a class or type for a multimedia segment. It defines a mapping from a particular *class_id* to a symbolic name and color coding. For example one could define a ClassEntry for the type *News* and define a colormap scheme for drawing the corresponding nodes. Each node of the *ClassTree* is a *ClassEntry.*
- ***ClassTree*** is a tree where every node is a *ClassEntry.* This tree is used for profiling, editing and rendering the *TimeTree*. It is important to understand that this tree is abstract in the sense that it is not tied to any specific multimedia instance. Since the *class_id* refers to a specific node in this tree, it actually represents a whole path from the root to that particular node. For example a *class_id* could represent a type that would be News−>International−>Politics.

Although conceptually different, the two trees are internally represented in the same format and the viewing and browsing tools are the same for each. This not only saves programming effort but unifies the interface for browsing and profiling, simplifying the interaction with the user.

Authoring

An important issue for our system is the creation of the tree representations. Although it is possible to imagine that in the future this process could be completely automated, the current level of technology is inadequate. On the other hand complete manual authoring is very time−consuming and there are various multimedia analysis technologies that, although not perfect, yield satisfactory results. Therefore to achieve the best result with minimum human effort a semi−automatic approach is needed.

Our system supports this semi−automatic approach mainly in two ways: exposing the tree structure in the authoring graphical user interface and integrating automatic analysis technologies. As an example of the first, if during the authoring process a node is labeled with a certain *class_id* then all the nodes that are ancestors of that node are marked with the same class id. This subtree semantics (explained in more detail in the user interface section) allows gradual refining of the structure and allows editing of large parts of the tree with one operation.

Typically, different analysis techniques work at different time granularities. For example a speech recognition analysis based on words works on a different time scale than a video shot detection. The tree representation allows the easy integration of these different techniques. Currently the lower levels of the tree corresponding to finer time scales are created automatically using multimodal analysis tools. The resulting lower levels must then be grouped to gradually build the *TimeTree.* This task is difficult to automate, therefore a human author is required. Since the temporal segmentation is very time consuming if done manually, having only to group the nodes saves significant amounts of authoring time.

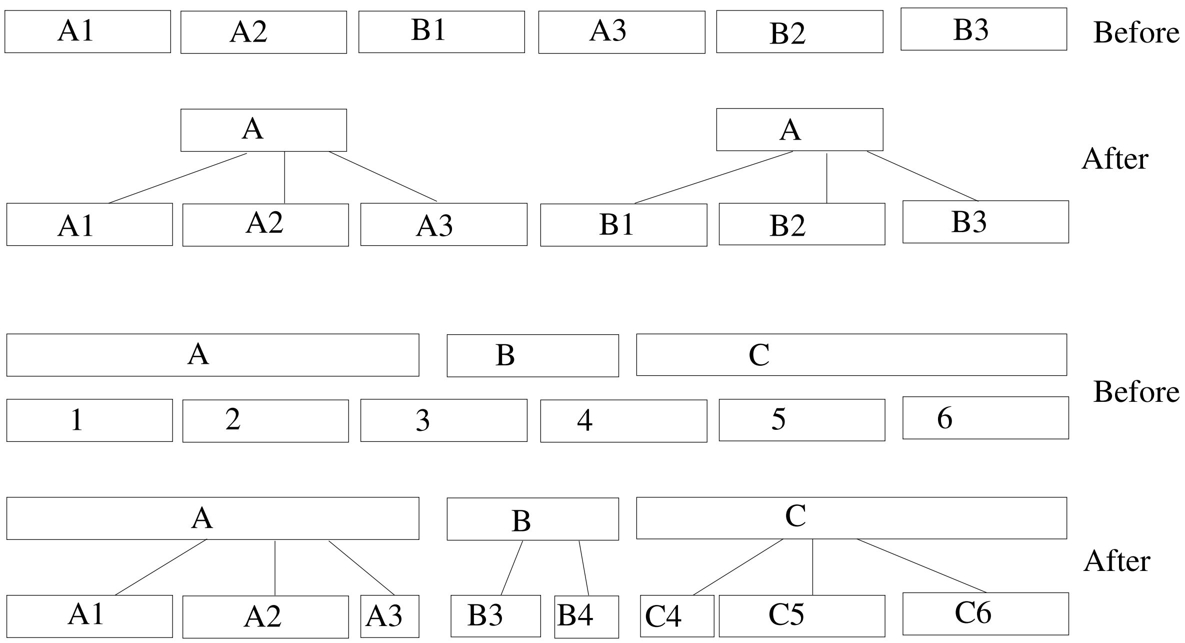


Figure 1. Class and dominant grouping

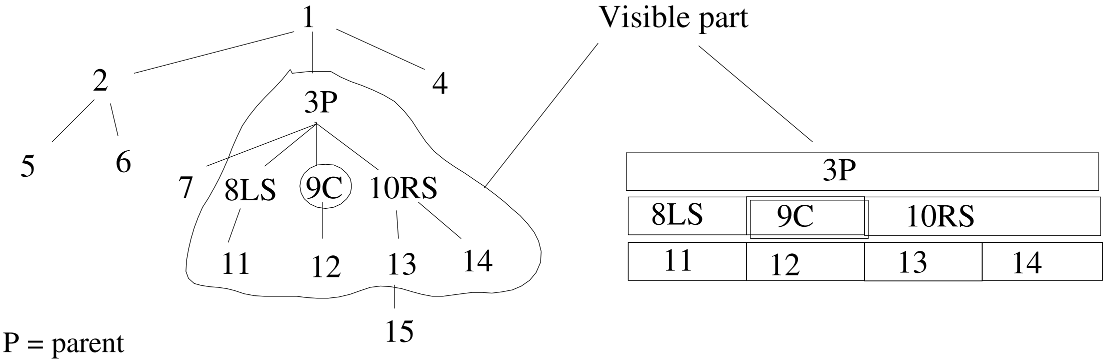
Although automating grouping is difficult there are some forms that are more easy to automate and can be used to assist the author :

- ***Class grouping*** is simply the creation of one additional level of tree from a time line by grouping together regions that have the same *class_id*.
- ***Dominant grouping*** is a heuristic that can be used to create two levels of a tree from two time lines resulting from different analysis tools. In this case, one time line is considered to have more on average coarser granularity and is retained as the dominant upper level while the other is oversegmented in time so that it can properly be grouped.
- ***Agglomerative clustering*** is described in more detail in (Chiu & Wilcox, 1998). In this algorithm, the two pairwise closest regions are grouped together to create a new level in the tree. The choice of the distance function is very important. We have used the distance between the color histograms of the regions for video and the distance between spectral features for audio. Other schemes are also possible.

Figure 1 graphically depicts class and dominant grouping. In class grouping A and B are different *class_ids,* for example *News* and *Commercials* and A1, A2, B1,etc. are specific instances. In dominant grouping the letters are the dominant analysis time line and the rectangle width corresponds to duration.

Another important aspect of authoring especially for viewing purposes is the ordering of nodes at each level of the tree. The system provides various forms of automatic sorting to assist the author. Typically sorting based on time is used on the lower levels of the tree and thematic sorting based on the *class_id* is used for the higher levels. In addition the author or the user can reorder the nodes of a particular level in any desired way, such as to accommodate his personal viewing preferences.

During the authoring process both the *TimeTree* that is being created and the *ClassTree* are visible and used. New categories and hierarchical relations can easily be added to the *ClassTree.* When the author adds a new entry to the *TimeTree* by clicking at the *ClassTree* he can easily establish the *class_id* linking. Many different editing operations are supported that allow the author to correct mistakes of the automatic analysis techniques and build the tree with minimal effort.



P = parent
C = current node
LS,RS = siblings
view_depth = 1, view_width = 1

Figure 2. Tree Browser and rectangular layout

User interface

Browsing and navigating through a tree data structure has been an old problem in user interface design and many different solutions have been proposed. The three most common ones are: (1) the expanding nodes tree browser (Explorer, Java tree), (2) the directory path of some shells and finally, although not typically thought this way, (3) Web pages. The expanding node browser is the most complicated and flexible allowing the user to adjust his view from global to local features in many different ways. Its main disadvantage is the large screen space it requires and the complex interaction with the user. Trying to control an expanding node browser without using the mouse is very difficult.

On the other extreme a directory path does not provides any information except the parents of the current node. The Web page provides only the siblings of the current node. Our graphical user interface tries to combine these two approaches providing a compact, intuitive adjustable view of the local neighborhood of the current node. The design of our interface was driven by the main constraints of our target application which are very limited screen space and discrete buttons of a remote control.

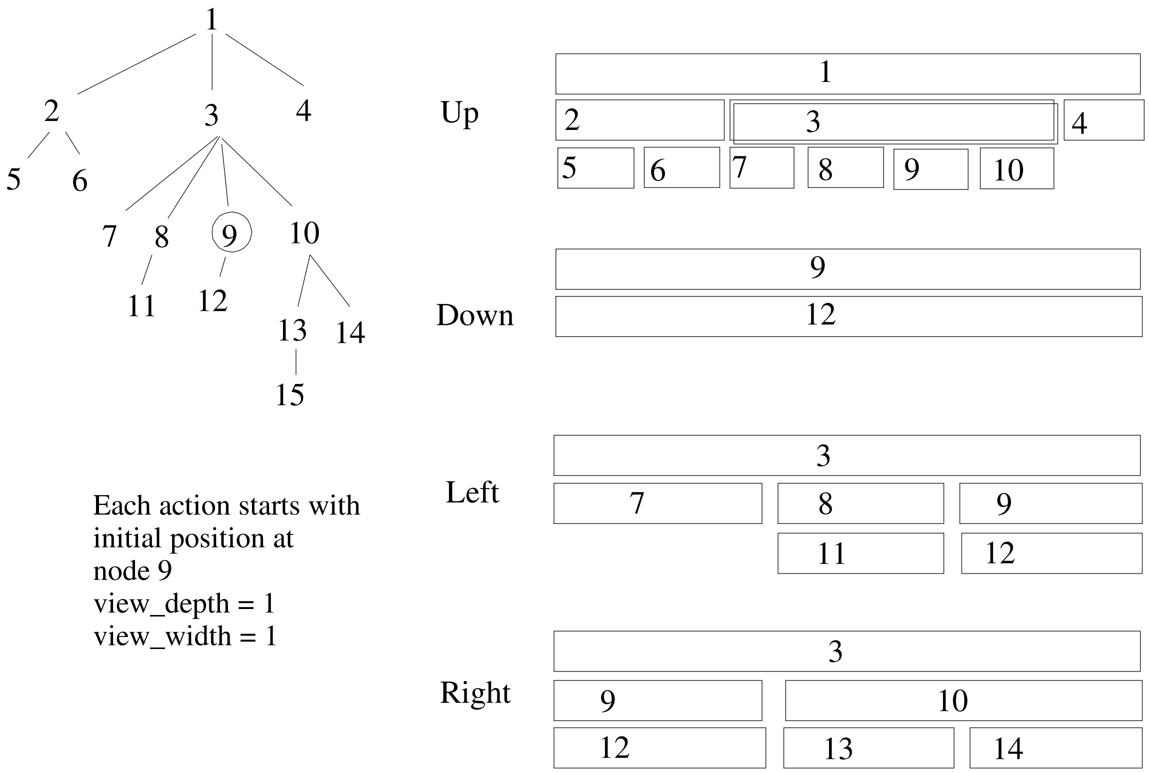
One way to think about our tree browser is that the user is essentially moving a local window around the current node. The visible nodes are only the nodes within this local window. There are two parameters that control the size of this moving window. The *view_depth* controls how many levels of the tree under the currently selected node are visible and the *view_width* controls how many siblings (and their subtrees) of the currently selected node are visible. Thus for the currently selected node the following nodes are displayed:

- The parent of the node
- The node itself (marked selected) together with its subtree up to *view_depth.*
- *view_width* siblings together with their subtrees up to level *view_depth*.

In terms of graphic layout we have chosen to use colored and labeled rectangles of different sizes. A parent rectangle has width equal to the sum of widths of its children and levels are displayed top down. Avoiding special cases and details the basic algorithm for constructing the layout is:

1. Make a regular grid where the cell size is equal to the desired screen width divided by the number of nodes of level *view_depth* plus the number of leaves that belong to the subtree of the current node and the subtrees of its *view_width* siblings.
2. Starting bottom up, incrementally build each level of the tree from the deeper level by making the width of each cell equal to the width of its children.

Our new tree browsing scheme and the rectangular layout are shown in Figure 2.



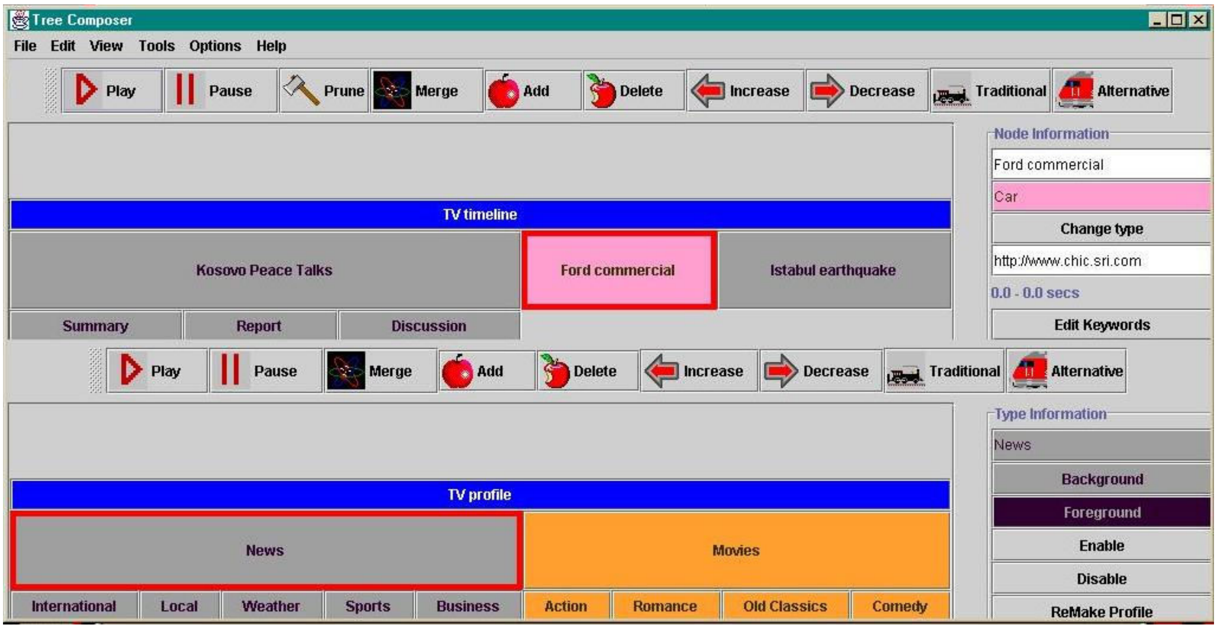
Each action starts with
initial position at
node 9
view_depth = 1
view_width = 1

Figure 3 TreeBrowser Actions

Of course, other layouts that retain the essential topological characteristics of our tree browsing are possible. Every time a user changes the currently selected node the view is adjusted accordingly. The possible actions are limited and can be performed with only four arrow buttons and one select button. Their semantics are:

- **Up** make the parent of the selected node the new selected node.
- **Down** make the leftmost child of the selected node the new selected node.
- **Left** make the left sibling of the selected node the new selected node. If there is no left−sibling perform an **Up** action.
- **Right** make the right sibling of the selected node the new selected node. If there is no right−sibling perform an **Up** action.
- **Select** is any action that operates on the currently selected node without changing it. Depending on the mode it can have different meanings. For example, in browsing **Select** would correspond to playback of the corresponding multimedia stream, in authoring it could correspond to editing the particular node and in profiling it could correspond to marking/unmarking a particular type. In some cases **Select** follows what we call subtree semantics, i.e. the operation is not only applied to the currently selected node but to all the nodes of its subtree. For example if a *News* type is marked in the profile then all the nodes of its subtree like *International*, *Local, Sports, etc.* are also marked. Similarly in authoring if a node is marked with a particular *class_id* its subtree is also marked with the same *class_id.* The subtree−semantics save user time by allowing mass operations on the tree that can subsequently be refined if necessary.

The system has been used by a limited number of users. The idea of using trees to express hierarchies is natural and all the users thought it helps with the interaction with the system. The new tree scheme seems to require some time to become familiar with it. Using a remote control and a real TV screen instead of a computer monitor makes the justification of the interface clearer to the users.



Ford commercial

Car

http://www.chic.sri.com

0.0 - 0.0 secs

News

Figure 4 TreeComposer screenshot

Implementation

The whole system is implemented using JAVA and OAA is used for communication between the components. The use of OAA (Martin et al, 1999) facilitates the integration of new analysis agents and allows the use of existing multimodal agents for video analysis, like those described in MAESTRO (Rivlin et al, 2000). In addition it allows various distributed configurations like running the analysis tools on a different machine from the authoring tool or the browser. Finally a standard format is defined for importing analysis data from other applications and it is easy to write wrappers for importing data from other applications. As an example, a utility for importing indices from Virage (Virage 2000), a commercial package for video indexing, has been written.

The system has the following components:

- *TreeComposer* is the authoring tool for creating and editing trees. A *TimeTree* and a *ClassTree* are displayed at the same screen and various editing, grouping and sorting operations are supported. Using a standard API the results of external automatic analysis agents can be incorporated. This integration is even easier in the case of OAA agents that can communicate directly to the system. In our application scenario, this would be the component used by the authoring agency. Both traditional expanding–nodes tree (Explorer, Java style) and our tree viewing scheme is supported.
- *TreeBrowser* provides viewing, profiling, and browsing capabilities (no editing). It is used by the end user for example in a Web page. As in *TreeComposer,* both tree viewing techniques are supported.
- *WebTVBrowser* is similar to the TreeBrowser, but works over WebTV (WebTV, 2000) to display the trees. A series of specialized WebTV html pages are created that can be used to navigate through the tree. Obviously only the new tree browsing scheme is used.

Future work

The main direction for future work is the development of more segmentation, annotation, and grouping agents. The dynamic nature and flexibility of OAA will help the integration of these new agents. In order to evaluate the graphical user interface, more careful user studies need to be conducted. It should be noted, however, that for the WebTV application, most other candidates for tree browsing would simply not work. Of course other new alternative tree browsers might be applicable. An especially interesting direction is the investigation of other graphical layouts that retain the basic topological features of our novel tree browsing scheme. Finally we plan to investigate using our tree browsing scheme in other application areas like Web browsing.

Summary

In this paper, a system for structuring multimedia data based on trees has been described. The trees are used extensively in all stages of authoring and presentation and express both temporal and categorical relations. The authoring is done semi−automatically and the dynamic nature of OAA allows the easy integration of new components. The driving application of the system is the creation of a personalized TV guide and browser. To accommodate the special constraints imposed by this application, a novel tree browser that uses compact space and discrete control was developed.

References

Arons, B. SpeechSkimmer: A system of interactively skimming recorded speech. *ACM Transactions Computer Human Interaction,* 1997, 4, (pp. 3−−38).

Bulterman, D., Hardman, L. (1995) Multimedia authoring tools: state of the art and research challenges in *Computer Science today: recent trends & developments* edited by Jan Van Leeuwen, Springer Verlag, Lecture notes in Computer Science.

Cheyer, A., and Julia (1998), L. MVIEWS: Multimodal tools for the video analyst, In *Proceedings of UIST 98* (pp. 55−62) ACM, San Fransisco.

Chiu, P., and Wilcox (1998), L. A dynamic grouping technique for ink and audio notes. In *Proceedings of UIST 98* (pp. 195−202) ACM, San Fransisco.

Harman, D., Bulterman, D and van Rossum, G (1993) The Amsterdam hypermedia model: extending hypertext to support real multimedia, *Hypermedia Journal* 5(1), (pp. 47−69)

Johnson, B., Shneiderman, B. (1991) TreeMaps: A space filling approach to the visualization of hierarchical information structures. In Proceedings of the 2nd *International IEEE Visualization Conference* (pp 91−128) San Diego, Oct 1991.

Martin, D., Cheyer, A. and Moran, D.(1999) The open agent architecture: A framework for building distributed software systems. *Applied Artificial Intelligence* 13(1−2) (pp.91−128).

Tzanetakis, G. and Cook P. (2000) MARSYAS: A framework for audio analysis. To appear in: *Organised Sound,* Cambridge University Press.

Rivlin,Z.et al (2000) MAESTRO: Conductor of multimedia analysis technologies, *Communications of the ACM,*43(2), (pp. 57−74).

Virage (2000): http://www.virage.com

WebTV (2000): http://www.webtv.com