RESPONSIVE TELEVISION

V. M. Bove, Jr. and S. Agamanolis

Massachusetts Institute of Technology Media Laboratory, USA

ABSTRACT

Responsive Television consists of a stream of media objects and procedural metadata that describes responses to viewer actions, identity, context, and equipment. Such a system enables providing personalized or responsive content even given a broadcast or multicast environment, and doesn't require potentially sensitive information about the viewer to be transmitted back to the source. A barrier to such development has been the difficulty of authoring the associated metadata. We are developing a programming-by-example editing tool that allows the user to specify a few examples of variations associated with variables known to the display; the system then can generate generalized rules and embody them in software that accompanies the media objects.

INTRODUCTION

The Internet has brought the ability to provide personalized content to the consumer, and indeed Internet users have become accustomed to media experiences that better match their needs and circumstances. But what does this mean to television? The Broadercasting project at the MIT Media Laboratory raises several questions in connection with the intersection of television and the Internet:

- ***Scalability of the system:*** With very large numbers of simultaneous viewers it may not be practical for a server to provide an individual stream for each, though distributed caching and dynamic resource reallocation can alleviate this problem.

- ***Rapid responsiveness:*** In some scenarios (***e.g.*** a program that adapts to the actions of individual viewers) it is necessary for the content to respond to rapidly changing situations. This requirement may not be compatible with latencies of up to several seconds that can occur because of either very long physical channels (such as satellite links) or cascaded memory buffers throughout the distribution system.

- ***Privacy concerns:*** Consumer preference or even legislation may prevent transmission of viewer information to a content server which would permit useful personalization.

- ***Shared experience versus complete personalization:*** To what degree will the viewers of a personalized television program still have a common experience, and who controls the limits – the viewer or the content creator?

Our responsive television research notes that significant computational intelligence now exists throughout the chain from production to display, and further that in many cases there is "extra" bandwidth available to the viewer's receiver. Thus we consider the possibility of client-side personalization or responsiveness, in other words moving some of the final editing process to the receiver and allowing for adaptation of the content to a specific viewing situation. We incur the cost of the added bandwidth occupied by the material that any particular viewer doesn't see, but we save complexity at the source server. In this work we do not assume a particular distribution channel, but we do target a broadcast model, either standard digital television via terrestrial/cable/satellite channels or broadband Internet multicast.

Providing at least a coarse degree of personalization such as advertisement targeting is relatively easy given currently-deployed infrastructure (1), but finer-grained personalization or real-time responsiveness requires writing a fairly complex piece of software describing mappings between response states and the output video sequences. Further, this approach does not match well with the thinking and working styles of most traditional video producers and editors, who are used to manipulating audio and video source material in a more direct and hands-on fashion.

Our model of a responsive television program consists of a stream of media objects and procedural metadata that describes how the program should play out in different situations, in response to a number of possible factors like the profile or preferences of the viewer, the equipment being used to display the program, or live feedback from sensing devices in the presentation environment. Adapting a program in these ways can help to better achieve the goals of its creator and can foster a richer connection with its viewers. Such a system also enables providing personalized or responsive content even given a broadcast or multicast environment, and doesn't require potentially sensitive information about the viewer to be transmitted back to the source. Possible applications include advertisements with different versions for particular people or situations, news programs that take into account a viewer's background, educational programs that adjust in real time to a student's attention level and other emotional responses, or environmentally responsive video installations in public spaces.

Our Viper system is a tool for creating responsive video programs. The system consists of three main components, all of which incorporate a graphical user interface where information is displayed or manipulated visually. The first component provides the ability to import source material and trim it into individual clips in a fairly traditional way. The second component provides an interface to allow the producer to annotate each video clip with important information that will be used to drive the automatic editing portion of the system. The final part of the system provides a means for building and visualizing abstract editing guidelines that will control how clips are selected and sequenced to create full video programs. The author may create several sets of editing guidelines for different situations, and the system will attempt to generalize from these specific cases to develop guidelines for other situations not yet considered.

Our work thus sits in the middle ground between standard editing tools (whether for linear or interactive playback), and totally automated generation of content (***e.g.*** (2)).

In the following sections, we describe the functionality offered by our tool, which is still under development, and we describe a documentary video program we were able to create with an early version of the tool.

CREATING CLIPS

The first component of our tool allows the video editor to import source video footage and split it into individual video clips. The graphical user interface for this component is much like a traditional interface for setting in and out points in a video sequence and maintaining a database of clips. However, the clips are not manually assembled into complete programs, as they would be in a traditional editing system. The goal of this stage is simply to create a large database of video clip objects, different subsets of which will be included in different versions of the final program. The author can also indicate a "critical section" for each clip that may be shorter than its full duration, allowing the system appropriately to vary the length of the clip if it is later selected for inclusion in the program.

ANNOTATING CLIPS

After creating a database of video clips, the author uses the second component of the tool to annotate each clip with information that will be needed by the system later to generate full edits of a program. This process happens in another graphical interface in which several types of annotations may be added to clips. Each clip may be rated on one or more arbitrary user-defined scales, such as "importance", "energy", or "closeup", to indicate the degree to which each clip reflects a certain characteristic. Clips may also be annotated with key words or other user-defined boolean variables. Another interface allows ordered or unordered groupings of clips to be expressed more explicitly. Yet another interface permits the author to create relationships among specific video clips, such as clip A "is a reaction to" clip B, or clip A "provides additional detail to" clip B.

The goal is to annotate the database of clips not with exhaustive or generic pieces of information (3) but with specific kinds of information that might be most helpful to generate meaningful results in the automatic editing stage. Some of this information might be generated automatically, by analyzing the video clip to detect certain features (4), but much of what is likely to be useful may be too content-specific or subjective to sensed by automatic means. The MPEG-7 standard outlines a format in which these annotations might be stored along with each clip, although the current version of the tool does not use the MPEG-7 format.

MAKING EDITING GUIDELINES

The third and most important part of our tool provides a means for building abstract editing guidelines that control how clips are chosen from the database and ordered to form a full video

program. The interface allows the user to create a hierarchy of program "atoms". Each atom represents a selection of one or more clips from the database, or an ordering of one or more atoms to form a sequence.

Clips are selected from the database by specifying properties and constraints, based on the annotations made in the previous stage, that should hold for whatever is chosen ("not yet selected", "high importance", "is a reaction to the previous clip"). Collections of clips or other atoms are put into a certain order by specifying similar sorts of rules and constraints ("increasing energy", "chronological order", "alternate closeups with wide-angles"). The selection and ordering guidelines can range from being somewhat fuzzy and abstract, giving the system a wide berth in how it makes its decisions, to being very rigid and explicit, calling for particular clips or arrangements by name.

It is at this stage that the actual responsiveness of the program is established. A set of parameters which will be known at the time of viewing must be declared. These can correspond to any number of response factors, such as viewer profile, equipment, or sensor outputs, and will drive the editing of the program. The author creates a template for a full program by assembling program atoms for an example set of parameter settings. Several templates of different structures may be created, each for a different presentation situation. Alternatively, a single template can be established, the parameters of which can change for different situations.

The system can generalize from a set of templates for specific situations to create new templates for other situations not yet considered. Since a group of templates defined for a particular program may contain many internal structural similarities, it is possible to formulate an algorithm that can appropriately interpolate among the tree structures to generate new templates. These system-generated templates incorporate, to different degrees, elements of the other author-defined templates, based on the "nearness" of the new situation to those that are already known. If the response variables for the presentation are modeled as real-valued continua, the system can detect numerical trends in their values over a group of templates and appropriately interpolate and extrapolate those values when considering a new situation.

PLAYBACK

In the playback phase, the system assembles a program based on the information in the chosen or generated template, attempting best to satisfy all of the constraints specified therein, and presents it to the viewer. If the situation incorporates live feedback mechanisms, the system can update the parameters (and thus the template) while the program is in progress, allowing real-time responses to be reflected immediately in the playout.

This automated assembly process could happen on the server side, where the response data needed to generate the program would be provided by the client through a back-channel mechanism, and final edited material would be streamed to the receiver using a standard delivery method. However, in situations where enough computational power and bandwidth are available, it is possible to perform the final assembly and personalization of the content entirely on the client side, thereby eliminating the need for any information to be released from the household or other viewing area. This ensures the protection of personal preferences, viewing habits, and other privacy-sensitive data from unauthorized use, and it lessens the need for complex on-demand video server architectures.

Currently, the playback system (like the authoring tool itself) runs on an interpreter for our Isis programming language (5), but the same basic engine could be used to generate output for other playback platforms.

AN EXAMPLE

Although it is still under development as of this writing, an early version of the tool has already been tested with moderate success in making a documentary program about a popular annual festival at an MIT dormitory. This event has been the subject of controversy throughout the MIT community in recent years because of the outre nature of some of the activities that are traditionally part of it (very loud music, mud wrestling, et cetera). The subject matter presented many opportunities to experiment with different forms of responsiveness.

We had several goals in developing a responsive video program about the event. First of all, we wanted to be able to tell the story in a suitable manner for different audiences (age ratings, preferences about emphasis on music versus action). We also wanted to be able to vary the duration of the show, from a quick summary to something very long and detailed, a feature that might become particularly useful for receivers equipped with disk-based personal video recorders (PVRs).

Given the range of situations we wished to explore, we decided to cover the event as exhaustively as possible with a single camera, paying special attention to the behaviors and reactions of the many

different attendees: organizers, students, campus police officers, performers. We digitized this source material into our tool and formed individual clips from anything that we thought might be useful in any possible final cut of the program. In the annotation phase, we rated each clip on various scales that we thought would be useful later, such as "importance" (to help in later deciding what to include in longer or shorter editions) and "close-up" (to help in controlling the level of intimacy and the balance of different kinds of shots). We also rated the clips for content and other subjective features like sex, energy, and so on.

While our tool does not yet provide functionality for more complex elements like L-cuts or transitions of different sorts, the editing guidelines we developed for the program outline a basic framework upon which clips are selectively added or trimmed to generate programs with different durations and different levels (for example) of sex, violence, and music.

Using various versions of these editing guidelines, we could experiment with many different presentation scenarios. For example, we could give the viewer manual control over the length and detail of the program, or we might use some kind of affect recognition system (6) to detect the viewer's attentiveness and involvement, appropriately controlling the amount of detail shown in each section of the program. We could take into account preferences of the viewer and censor material that might offend. We could also create versions that exhibit different behavior based on equipment factors like the size or aspect ratio of the screen (7).

In any of these cases, two key points are important to keep in mind. First, it is the author of the program, not the viewer or a third party, who maintains the control in deciding how and to what extent the program will be assembled differently for different situations and what freedoms the viewers will have to interact with and personalize the program to suit their desires. Secondly, no information about the viewer's preferences or interaction habits ever needs to be transmitted outside of the viewing area in order to accomplish the personalization and final assembly of the video program.

Several productions, including advertising, informational programming, and entertainment, are in the planning stages so that we can further test the tool and refine the direction of its development. For more information and the latest results, go to http://isis.www.media.mit.edu and click on "projects."

REFERENCES

1. Simons, D., 2000. Digital TV Wins. Forbes. 28 June 2000.

2. Dalal, M. ***et al***., 1996. Negotiation for Automated Generation of Temporal Multimedia Presentations. Proc. ACM Multimedia 96. pp. 55 to 64.

3. Davis, M., 1993. Media Streams: An Iconic Visual Language for Video Annotation. Telektronikk, 4.93. August 1993. pp. 59 to 71.

4. Vasconcelos, N., 2000. A Probabilistic Architecture for Content-based Image Retrieval. Proc. CVPR '00.

5. Agamanolis, S. and Bove Jr., V. M., 1997. Multilevel Scripting for Responsive Multimedia. IEEE Multimedia. 4:4. October-December 1997. pp. 40 to 50.

6. Picard, R., 2000. Toward Computers that Recognize and Respond to User Emotion. IBM Systems Journal. 39:3-4. To appear.

7. Bove Jr., V. M., 1995. Object-Oriented Television. SMPTE Journal. 104. December 1995. pp. 803 to 807.

ACKNOWLEDGMENTS

We wish to thank Matthew Palmer for assisting in the video production and helping to test the tool. The research described in this paper has been supported by the Digital Life Consortium and the Broadercasting Special Interest Group at the MIT Media Laboratory.

Figure 1: A screen from the annotation mode of the Viper tool, showing parameters associated with the shot.