Proc. 140th SMPTE Technical Conference, 1998

Adding Hyperlinks to Digital Television

V. Michael Bove, Jr., Jonathan Dakss, Stefan Agamanolis, Edmond Chalom
MIT Media Laboratory
Cambridge, MA USA

Abstract

Hyperlinked video is video in which specific objects are made selectable by some form of user interface, and the user's interactions with these objects modify the presentation of the video. Identifying and tracking the objects remains one of the chief difficulties in authoring hyperlinked video; we solve this problem through the use of a video tracking and segmentation algorithm that uses color, texture, motion, and position parameters. An author uses a computer mouse to scribble roughly on each desired object in a frame of video, and the system generates a segmentation mask for that frame and following frames. We have applied this technique in the production of a soap opera program, with the result that the user can inquire about purchasing clothing and furnishings used in the show. We will discuss this and other uses of the technology, describe our experiences in using the segmentation algorithm for hyperlinked video purposes in both broadcast and on-demand modes, and present several different user-interface methods appropriate for hyperlinked video.

Introduction

Users of the World Wide Web have become accustomed to hyperlinks, in which "clicking" on a word or graphic selects another page, or perhaps modifies the current one. The idea of hyperlinked video, in which objects are selectable, has often been discussed as a desirable possibility – consider for instance a fashion program in which clicking on an article of apparel provides information about it, or a nature program that lets children go on "safari" and collect specimens. Playback of such material is well within the capabilities of typical digital television decoders with graphical overlay capability, but creating it has posed a challenge because of the tediousness of identifying the clickable regions in every frame manually and the difficulty of segmenting and tracking them automatically.

We have developed a method for tracking and segmenting objects which simplifies the authoring problem. During editing, the author simply uses the mouse to scribble roughly on each desired object in a frame of video, and the system generates a filled-in "segmentation mask" for that frame and for following and preceding frames, until there is a scene change or the entrance of new objects. Regions can then have an action (*e.g.* graphical overlay, switching to a different video data stream, transmission of data on a back channel) associated with them. The viewer can select objects by a variety of means, including ordinary remote controls or remote controls with inertial sensors; in our demonstrations we have set up a video projector that can identify and locate a laser pointer aimed at its screen.

Our hyperlinked video system has two main technical underpinnings. First, we apply a novel method of using color, texture, motion, and position for object segmentation and tracking of video. While the author is indicating regions of importance by scribbling with the mouse, our system uses a combination of these features to develop multi-modal statistical models for each region. The system then creates a segmentation mask by finding entire regions that are statistically similar and tracking them throughout a video scene. The second piece is a scripting language for object-based media called Isis, which enables the association of the mask with the video frames, and the association of actions with selecting regions at particular times.

We utilized this system to author a prototype hyperlinked video program which resembles an episode of a television serial drama and enables the viewer to select props, clothing and scenery and be shown purchasing information for the item, including the item's price and retailer. We produced the content used in the program, which we named "HyperSoap," with the belief that existing television material could not be adapted effectively to hyperlinked video. This entailed changes in the way HyperSoap was scripted, shot, and edited when compared to a traditional television program. We were able to learn a great deal about how people interact with hyperlinked video, and based our design of several modes of user interaction on this information.

In the next section we will describe existing hyperlinked video software. In following sections we will describe the issues we were confronted with in designing systems for authoring and playing back hyperlinked video. We then will discuss the design of video content for a hyperlinked video production, using HyperSoap as an example. Finally we will summarize our work and present other applications for this technology.

Authoring Hyperlinked Video

Several companies have announced products for hyperlinked video authoring. VisualSHOCK MOVIE from the New Business Development Group of Mitsubishi Electric America, Inc. (http://www.visualshock.com) concentrates on Web and local playback (*e.g.* CD-ROM, DVD-ROM). The authoring tool, called the Movie MapEditor, requires specifying a rectangular bounding box around the location of a desired object in the starting and ending frames of a video sequence, and a tracking algorithm estimates the position of the "hot button" in intermediate frames; manual tracking by the operator is also supported. HotVideo from IBM's Internet Media Group (http://www.software.ibm.com/net.media/solutions/hotvideo) has a similar approach, but supports a larger set of geometrical object shapes. Intermediate locations of an object may be specified by multiple keyframes or located by a tracking algorithm. Another current authoring tool to incorporate an object-tracking algorithm is Veon's V-Active (http://www.veon.com).

We have found that this method for associating links or actions to objects is too coarse for the objects found in many varieties of video sequences. To enable the association of links with arbitrarily shaped regions, a system is needed which can differentiate among objects with higher precision.

Segmenting Objects

If we want to enable the author to specify clickable regions that follow object outlines rather than simple geometrical shapes, and if we don't want to require manual outlining in every frame, some automated segmentation method is necessary. For example, the VisualSEEk system [1] automatically identifies regions based on color and texture distribution. This and other automatic systems are limited by the ability of the system to generate an internal model that corresponds to actual objects as required by the author. Good quality is important for situations in which the labels the system generates for pixels (known as a "segmentation mask") are rendered for the viewer, for example to indicate the clickable regions. It must be noted, of course, that playing back video with a detailed segmentation will require much more information to be transmitted or at least retained on the server than would be the case with geometrical regions that can be described in terms of their coordinates.

We have developed a novel segmentation system which classifies every pixel in every frame of a video sequence as a member of an object, or group of objects, as defined by an author [2][3]. The system enables the author to define objects by labeling representative clusters of pixels in a single frame. The pixels which the author has classified are then tracked through the remaining frames of the sequence. The system computes multiple features for all of the pixels in the sequence, which include color, texture, motion and position. For each frame, statistical models for each object are created based on the features of the classified pixels. These models are then used to classify each unlabelled pixel to the object with highest statistical similarity.

We have created a simple drawing tool which enables the author to indicate regions of importance within a video sequence. The user selects a single representative frame from the sequence (typically one in which the important regions are fully visible) and uses the mouse to highlight representative pixels for each desired object. When the author has finished labeling pixels in the frame, our system estimates the location of these pixels within each of the remaining frames in the sequence using block matching. When this stage has completed, there are pixels in each frame which have been classified to correspond to objects. The next stage of processing will assign the rest of the pixels in the image to one of these objects.

Starting from small subset of pixels that the author has indicated in one frame and classifying every pixel in every frame is done by calculating a multi-modal statistical model based on a variety of different features including the color, motion, and position of a pixel and the texture and motion properties of a small neighborhood centered upon it. We have found that this yields a more robust segmentation, whose quality is minimally affected by misleading data from a single feature. Because a region's model is multi-modal, it is for instance possible to specify a region that has red and yellow pixels, and the system will not assign intermediate shades of orange to that object.

After we have calculated a multi-dimensional feature vector for every pixel in the video sequence, we use the feature information for the tracked pixels to form a statistical model for each user-identified region. We then compare the feature information of each unclassified pixel to these models and label the pixel as a member of the region to which it bears the greatest statistical similarity. Basically, we use feature information to express each region as a mathematical function such that we can input feature values of unclassified pixels and receive as output the probability that the pixel corresponds to that region.

Because only statistical methods are employed to classify pixels, it is likely that there will be small aberrations in otherwise consistent classification, such as a small group of mislabeled pixels surrounded by properly classified pixels. We use two algorithms during a postprocessing stage to help rectify these aberrations. To help remove small groups of misclassified pixels, we apply a process which "unclassifies" each pixel which was originally classified with a certainty below a set threshold. A second algorithm, known as "K-nearest neighbor" (KNN), reassigns the unclassified pixels to the region which is most popular amongst its neighboring pixels.

Because objects in video are likely to reoccur in several shots and the system is run on shots individually, at this stage the author may wish to adjust region labels which have been set locally to values which represent a region or object globally. This could also mean adjusting mask values to a special number (*i.e.* 0) which corresponds to regions for which there is no action or link desired.

At this stage the system has generated a second video sequence, which contains the segmentation information for every frame of the input sequence.

Associating Actions with Objects

If pixel-level identification of objects is possible (as it is with our system), a person can interact with the video during playback by indicating one or more pixels which comprise a particular object; the playback system can then cause a corresponding event or action to occur. This event could be the display of additional information about the object, such as the rendering of text or an image, or perhaps retrieving information from a URL on the World Wide Web. An event could also be a change in the way the video is presented, such as showing scenes from alternate camera angles or playing back a different part of the video which is relevant to the selected object. For example, the hyperlinked video "HyperCafe" [4] allows a person to navigate through various video clips of conversations occurring in a cafe by selecting actors in the video or icons which appear at certain times during the presentation.

One possible implementation for enabling this association is to provide a database in which object labels can be cross-referenced with descriptions of the objects (if one of the desired actions is to render object information) or instructions for executing actions. Many of the existing hyperlinked video systems utilize this infrastructure; for example, VisualSHOCK maintains an "anchor list," accessible via a drop-down menu system, where data structures containing link information are stored and referenced. The playback system for HyperSoap, built in a scripting environment called Isis [5], maintains an internal database consisting of a list of lists, in which each list corresponds to a particular object and contains text strings which include the name of the object, as well as actions associated with the object.

Interacting with Hyperlinked Video

In this section, we describe the various issues which confronted us when we designed our playback system. Important user interface issues, including methods in which the presence of links are indicated in video, mechanisms used by the viewer to select objects, and approaches to initiating link events are described in detail. In many cases, our solutions to these issues vary greatly from existing implementations.

One reason for this difference is that we focused our attention on user interaction mechanisms apart from the desktop PC and instead within the television paradigm. In some ways using a mouse on a PC may be natural for hyperlinked video, considering that people are used to using a mouse to activate hyperlinks on the World Wide Web. However, the desktop is not an ideal place to view video; additionally, many of the genres of content suitable for hyperlinked video are programs which people are accustomed to watching in their living rooms.

We designed our system with the intent of simulating a future television viewing scenario, one in which a device with significant computational ability (*e.g.* a set-top box, enhanced television receiver or DVD player) could render video content while receiving directions about playback from both the viewer and the content itself. For our implementation we used a projection screen and a video projector which displayed video data transmitted by a workstation running Isis, a scripting environment which we used to create our playback software.

When displaying hyperlinked video, we feel it is important to indicate the presence of hyperlinks to viewers in a manner which does not distract the viewer from the content of the video. This also enables a scenario in which the viewer chooses to watch the presentation in a passive manner, as one might read a document on the World Wide Web without choosing to click on hyperlinked words. Ideally, viewers might be able to adjust the manner in which hyperlink opportunities are presented to suit their own preferences.

Many of the existing hyperlinked video playback tools utilize a common and minimally distracting method for presenting hyperlink opportunities to the viewer. Since the viewer will be interacting with the video using a pointing device (a mouse or its equivalent), when the viewer moves the cursor over a hyperlinked region, the cursor changes shape. At the same time, a name or description for the link is displayed in an area of the screen. IBM's HotVideo software goes a step further and changes the cursor to an icon which corresponds to the type of information contained in a link, such as a small document if the link is text or a film reel if the object links to another video stream. HotVideo also displays an indicator at all times which changes color to indicate when there are hyperlink opportunities within the frames of video that are being shown to the viewer. This method is also used by WebTV's WebPIP (http://developer.webtv.net/docs/itv/Default.htm) as there is always only one hyperlink opportunity in a video segment.

Other approaches can be more distracting. Whenever a frame contains hyperlinked objects, HotVideo can also render the wireframe shapes which the author used to indicate the regions. This method is suitable when there are only a small number of objects present in each frame of video, but can cause the video to become quite cluttered for higher numbers of objects. It can also be confusing to the viewer if the linked object occupies a large portion of the video frame and may surround other objects, such as a wall in an interior shot of a house. Another potentially distracting display method which a HotVideo author can choose is to render the pixels within the wireframe shape in a different manner than the ones outside the shape, such as adjusting the brightness or color tint of those pixels. Many of these systems do not allow the viewer to choose the method in which all links are indicated, in favor of having the author determine this method per link.

Since our method allows for identifying objects at a pixel level, we are able to show the user the presence of hyperlinks by highlighting all of the pixels which comprise a selected object. This highlighting effect is achieved by using Isis' ability to overlay an image buffer with a video stream in real time. During playback, there is an image buffer whose pixels are all set to a solid color. When the viewer selects a pixel which corresponds to a hyperlinked object, the playback script retrieves the segmentation mask for the next frame and sets the alpha channel of the pixels in the image buffer to correspond to the selected object's pixels in the segmentation mask. In this manner, the only pixels in the image buffer which will be rendered are the ones which correspond to the selected object. Then, when the next frame of video is ready to be shown, Isis composites this frame of video with the image buffer (keeping the image buffer slightly transparent) thus indicating the presence of a hyperlink. If the viewer continues to select pixels from the same object, the playback script updates the alpha channel of the pixels in the image buffer. When the user stops selecting pixels, the script continues to update and render the image buffer, but adjusts the transparency of the buffer over time. This gives the viewer the impression that the link is "fading" out of the picture. We have also implemented two of these buffers simultaneously, which allows the system to show the presence of another object while the previously selected object may still be fading.

The means by which the viewer of hyperlinked video selects links should be simple to use and feel as natural to the viewer as possible. Since people are comfortable using a remote control device to browse television programming, it makes sense for them to use this device to browse through hyperlinks in that content. WebTV provides buttons on the remote which can be pressed when the icon appears indicating the presence of a link for the present segment. The viewer can then choose to view the web page content referenced by the link or archive the link for later viewing. When several links are present, we envision a button on the remote control which would allow the viewer to cycle through the hyperlinked objects on the screen while a graphical overlay would be provided to indicate the location of each object as it is selected during the cycle. The remote might also incorporate a position touch pad, a joystick, or an inertial sensor

For our demonstrations, we used a hand-held laser pointer as the interaction mechanism and set up a projection device which could both project frames of video and track the position of the laser on the screen. To do this, we added a small digital camera to a regular video projector and built the functionality into the Isis playback script to use the camera's output. Since the laser pointer forms a red dot with significantly greater intensity than any of the red pixels of the projected video, the system looks at only the red channel of the camera's output and applies a bounding-box algorithm to determine the center of the laser spot. We then apply a linear transformation to correct for possible keystone distortion, and compute a calibrated (x,y) position for the laser in units of screen pixels.

Since the viewer may have no prior knowledge of which objects in the video are hyperlinked, there must be some method to allow the user to browse through objects shown on the screen before selecting one. To achieve this with our laser pointer interface, our playback script keeps track of the number of frames in which the viewer has pointed to the same object. If the viewer selects the object for more than a set number of frames, the script assumes the viewer has intended to select that object. To show the user that the object has been selected, the pixel values in the image buffer for highlighting are set to a solid color, thus modifying the appearance of the highlighting.

The temporal nature of video raises an interesting issue regarding the display and timing of events associated with embedded hyperlinks. Whereas hyperlinked documents on the World Wide Web are stationary and can be read in any desired fashion, there is a flow to video content which is generally not controlled by the viewer. While all of the existing systems direct web browsers to video links immediately, we have found that displaying lengthy or detailed information while the video continues to run may cause the viewer to feel distracted from the video content by having to read or examine an image simultaneously. Likewise, a delay in showing the link information may blur the context of the information such that it is no longer relevant to the video presentation. Thus events triggered by selecting hyperlinks must be presented in a way which maintains the continuity of the video while still presenting the desired link information to the viewer in a timely and meaningful fashion.

Additionally, the system architecture and the capabilities of the viewer's equipment may affect the degrees of freedom available. If the video is playing on-demand from a server, or locally from a DVD, random access and pausing are possible, and it's not necessary to do much caching of content. The segmentation mask doesn't necessarily even need to be transmitted, or it could be transmitted only when the user is actually pointing at something on the screen. In a broadcast environment, different content design might be needed, since pausing isn't possible; also the display device needs the segmentation mask at all times. We've experimented with strategies appropriate for each case. In an early experiment, selecting a linked caused the video to be interrupted immediately to show the viewer the link content. After a set period of time, the link content would disappear and the video would resume from the last frame shown before the interruption. While some users enjoyed the instant response from the system, others found it somewhat troubling as their interactions would interrupt important moments within the content, such as cutting an actor off while they were delivering a line. Also, people tended to forget the context of the video content before the interruption and found themselves somewhat lost in the plot of the video when it resumed.

Our next implementation yielded a pleasanter manner of interaction: once a viewer selected a hyperlink, the script would wait until the video reached a natural breakpoint (for instance, after an actor finishes speaking a set of lines) before interrupting and showing the link's information. This yielded a much more favorable reaction from users of the system, although the initial reaction of many first-time users to waiting for information was that the playback mechanism was not working properly. Others who felt more interested in the link information than in the video content expressed that they would have preferred instant feedback.

In addition to this mode of interaction, we decided to implement two additional modes for our system and gave the user the capability to toggle between all three modes during playback. The first additional mode overlaid an abridged version of the link information while the video continued to run. Based on the area of the screen where the user selected an object, the system would try to render the information such that it would be less likely to occlude other hyperlinked objects. This could be combined with a feature similar to WebTV's such that each link was archived to enable the viewer to see the complete information after the end of the program.

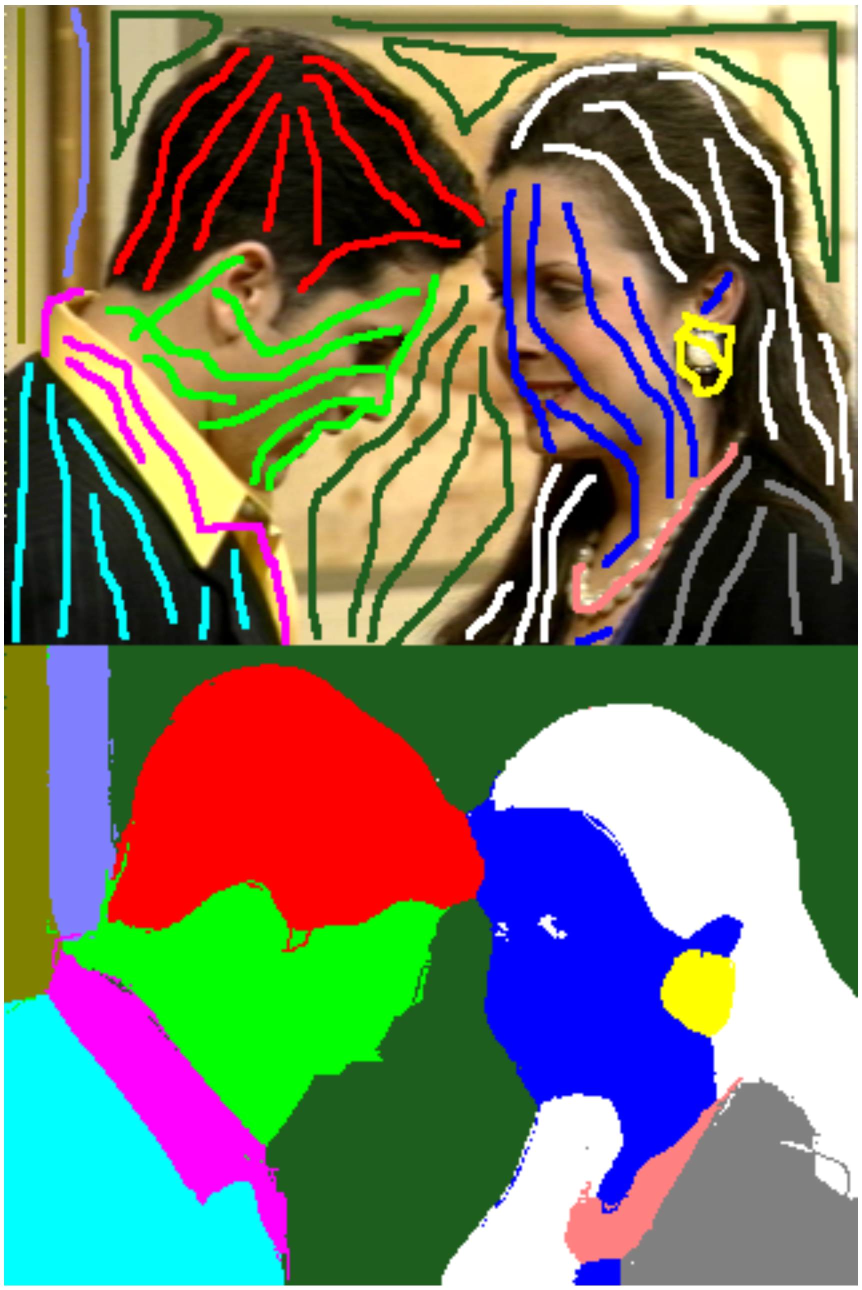
Figure 1: A video frame with training points overlaid on it (top). The resulting segmentation (bottom).

HyperSoap

We used our authoring and playback systems to create HyperSoap, a four-minute prototype hyperlinked video drama whose content was closely patterned after daytime soap operas. We oversaw every aspect of the production of HyperSoap content, including scriptwriting, storyboarding and directing the shoot and postprocessing the video and audio. Every item on the set, including clothes, props and furnishings, was taken from the JCPenney catalog and was "for sale." Pointing the laser-equipped remote at the screen highlighted selectable objects, and keeping the pointer fixed on one object for an extended period selected it. In instant-feedback mode (simulating a broadcast), a small text window would be displayed on the screen containing the item's description, brand name, price and a logo for the distributor. In breakpoint mode, more descriptive text boxes would be shown, as well as a still image resembling a photo of the item from a catalog. This same information could also be rendered at the end of the scene.

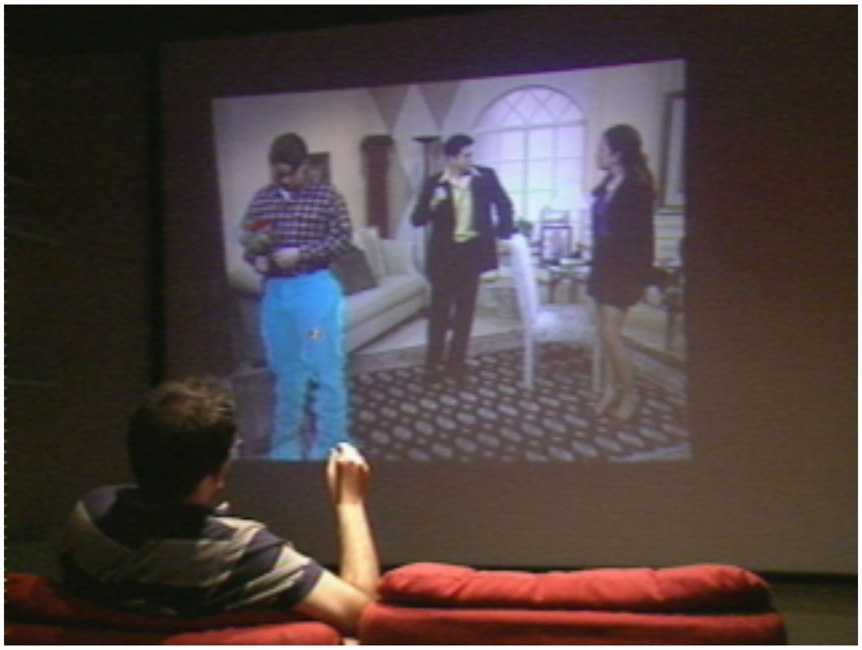
Figure 2: A viewer indicates a desired object by pointing at it with a special remote control.

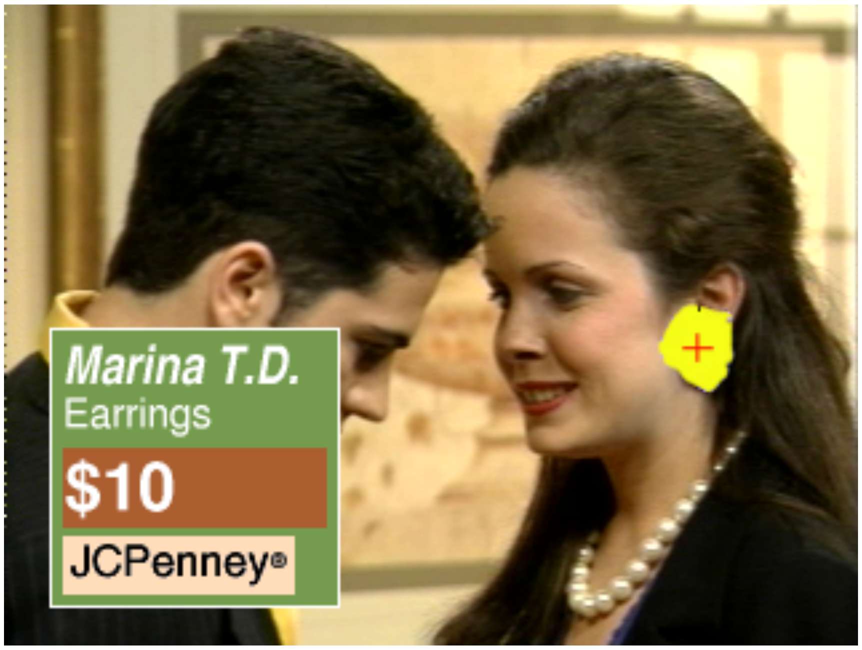

Figure 3: Information called up by selecting an object.

In breakpoint mode, we used music to help maintain continuity between the video and the hyperlink information. A soundtrack was composed for HyperSoap with the unique quality that it could continue despite interruptions and still help to underscore the action being played out in the video. This was achieved by having the composer write pieces of music for specific parts of the scene which could be seamlessly looped by the playback script. When the scene changed from one part to the next, the music for the previous part would fade out while the new one began. As a result, when the user selected a link, the background music for the scene would continue through the interruption. Many users commented that the music helped to bridge the interruptions of the program content when objects were selected.

Our authoring system produced good enough pixel-level segmentation of the 40 objects which appear in HyperSoap to permit the use of the segmentation mask for highlighting. There were a total of 45 shots which needed to be run through the system individually, where the number of linkable objects per shot ranged from 10 to 20. We found that to maintain a consistently high quality of segmentation, we needed to retrain the system after approximately 30 frames, or approximately one second of video. This was largely done to prevent the propagation of errors during the tracking of the author-specified pixels. However, this still provided an exceptional level of automation of the segmentation process; typically an average of 5000 pixels were classified in each training frame by the author, meaning that for each 30-frame sequence the authoring system was required to classify 99.8% of the total pixels. Additionally, we found we were able to reduce the amount of data in the segmentation mask by subsampling the data vertically and horizontally by a factor of 2. This had a minimal effect on the quality of the upsampled mask when it is rendered.

HyperSoap raised several interesting issues regarding the differences between traditional television programming and hyperlinked video content. A traditional television soap opera is designed in such a way that simply adding hyperlinks and allowing interruptions would likely detract from its effectiveness because they would obstruct the flow of one of the most important components of this format, the story.

In designing the content for HyperSoap, we had to move away from a form where the story is central and any product placement or viewer interaction are decorations, to a new form where all of these components are equally important and dependent on each other for the survival of the presentation. The knowledge that everything would be for sale and that viewers would be interacting often was part of the program's design from the very beginning. This entailed several changes in the way the scene was scripted, shot, and edited when compared to a traditional TV program.

For example, shots were designed to emphasize the products for sale while not drawing too much attention away from the flow of the story. This often required closeups of certain objects which would have otherwise been difficult to see or select with our interaction device; actions needed to be written in the script so these closeups would not seem unnatural. Similarly, the scene was edited so that shots were either long enough or returned to often enough for viewers to spot products and decide to select them. Also, the story itself involved the products in ways that might increase their value to the consumer. Integrating the story with the products made interruptions to show product information less jarring overall.

Summary and Applications

We found that the knowledge that all of the objects in HyperSoap were for sale and that the program was designed for the dual purpose of entertainment and shopping motivated viewers to "stay tuned" and to interact with the products. As an additional way to motivate viewers to select products, we also implemented another version of HyperSoap in which selecting objects revealed details about the plot and characters in the drama which pertained to those items. We feel that both of these types of programming would create an intriguing form of interaction, were they to be implemented in a broadcast model. Other potential types of content which could benefit from "hyper" capabilities include how-to videos, particularly ones which involve complicated components, such as repairing an automobile or learning to fly an airplane. Likewise, training videos which demonstrate complex processes in which several people perform tasks simultaneously (*e.g.* a surgical procedure or a football play) could allow viewers with different backgrounds to learn more about the tasks or people whose purpose particularly interests them. This same idea could be applied to documentaries and educational programming, for example enabling children to learn more about particular aspects of nature by selecting animals and plants observed in "safari" footage, as if they were collecting specimens on a real safari.

Although we used the Isis language for our production, it is quite possible to use more standard tools (*e.g.* Java). While the computational demands of segmentation are quite high, display is well within the capabilities of a typical PC or a set-top box with graphical overlay capabilities.

Acknowledgments

We would like to thank Michael Ponder, JCPenney, and the JCPenney Headquarters Production Staff for their invaluable assistance with the HyperSoap shoot. We would also like to thank Kevin Brooks, Paul Nemirovsky and Alex Westner for providing their scriptwriting, scoring and Foley sound expertise. Also a special thank you to Julia Ann Lewis, Matthew Rupe and Andrew T. Chandler, for their tremendous acting talents. This work has been supported by the Digital Life Consortium at the MIT Media Laboratory.

References

[1] J. R. Smith and S.-F. Chang, "Local Color and Texture Extraction in Spatial Query," *IEEE Proc. Int. Conf. Image Processing*, pp. III-1011—III-1016, 1996.

[2] E. Chalom and V. M. Bove, Jr., "Segmentation of an Image Sequence Using Multi-Dimensional Image Attributes," ," *IEEE Proc. Int. Conf. Image Processing*, pp. II-525—II-528, 1996.

[3] E. Chalom, "Statistical Image Sequence Segmentation Using Multidimensional Attributes," PhD thesis, MIT, Cambridge MA, 1998.

[4] N. Sawhney, D. Balcom, and I. Smith, "Authoring and Navigating Video in Space and Time," *IEEE Multimedia*, 4:4, pp. 30-39, Oct. 1997.

[5] S. Agamanolis and V. M. Bove, Jr. "Multilevel Scripting for Responsive Multimedia," *IEEE Multimedia*, 4:4, pp. 40-49, Oct. 1997.