An Orthogonal Taxonomy for Hyperlink Anchor Generation in Video Streams Using OvalTine

Jason McC. Smith, David Stotts, Sang-Uok Kum

Department of Computer Science
University of North Carolina at Chapel Hill
Chapel Hill, NC 27599-3175, USA
Tel: 1-919-962-1833
E-mail: {smithja,stotts,kumsu}@cs.unc.edu

ABSTRACT
As dynamically linked content follows the progression of statically linked media into the realm of video, new opportunities for link creation become apparent. In this paper we describe a real-time video hypermedia system with user-definable linkage areas, in a distributed collaborative environment. We also investigate the extension of such a system to automated link creation in video streams. In the process, we identify and describe orthogonal issues of hypervideo anchor creation. An example system, OvalTine, has been produced to illustrate several potential uses through configuration of an extended video conferencing application on the SGI O2 platform.

KEYWORDS: Hypervideo, collaboration, automated anchor generation, digital video, streaming video

TRENDS TOWARD DYNAMIC CONTENT

We use the term hypervideo to refer to a displayed video stream that contains embedded user-clickable anchors. These anchors are logically attached to objects within the video environment, independent of location within the field of view [18]. For instance, a person's face in a video conferencing system may be designated as a hyperlink. As the person moves within the image, the clickable area which activates that hyperlink will move with the image of the face, so that the face itself defines the active region.

We do not specify to what the anchor may refer, nor do we limit the type of objects which may be designated as a link anchor. The goal of our work is to explore the creation and maintenance of hyperlinks in video streams, and to automate these procedures as much as possible. Before any technical discussions, we look at previous work in hypervideo.

Multimedia has shown a definite trend in computing from text to static images to video. On a parallel path, hyperlinking systems, most notably the World Wide Web, have evolved from the use of pure text such as in Lynx [10], to the incorporation of static images with embedded anchors (image maps), and now to video streams with active regions [1, 16].

All of these systems, however, are examples of manual anchor creation on a server only, and are, in addition, pre-defined for preset text, images, or video. This severely restricts the usefulness of linkages in a collaborative environment.

Every current popular method for adding active regions to video requires manual selection of video objects, on a frame by frame basis. By contrast, an automatable object tracking system is much more desirable, both for real-time applications, and for the automated addition of hyperlinks to the vast amount of archived video currently in existence.

Automation of content in hypertext is a well-researched area [2, 3, 9]. While much work has been done in the realm of context-assisted anchor creation in video, particularly in news coverage videos, these rely on a blend of modal data, including much which is manually entered by human operators [5, 14, 22, 23]. Instead, we are more interested in identifying issues related to nearly context-free object tracking within image streams. The image analysis engine selected for use on the video, or the user manually creating links, supplies the context. A facial recognition system designed to attribute a link to a database record consists of an entirely different inherent context than an engine which identifies and tracks types of automobiles on a roadway, but they both can operate equally well on the same raw video stream. This is a complementary concept to content-oriented navigation [12], where the context is provided during modal analysis. Since the context of the link data has been removed from the dimensions we define, the resultant contextual link generation engines are also close analogues to Sprocs as defined by Nürnberg, et al [17].

Text is almost exclusively an archived data source, authored once and stored, and then presented as pre-established data to the user. Research into dynamic text systems is established [3, 15, 21], but it is our opinion that text will not reach the temporally and spatially dynamic properties of video in the near future. Video allows for, and generally requires, different approaches to anchor generation than does text.

Given these trends and issues, we have identified three independent axes of interest in anchor creation for hypervideo: manual vs. automatic; server-side vs. client-side anchor creation; and real-time vs. archived video (Fig. 1).

Server-side manual anchor generation is the standard approach for authoring most hypermedia, whether video or text. We define server-side to mean any anchor creation which occurs prior to delivery of the media to a client machine, and client-side to mean any anchor generation which occurs after transmission to the client machine. Text based client-side anchor generation, more notably in automated systems, can suffer from the problem of relevant context [8, 9]. We feel that the selection of objects that are visually presented provides a cleaner approach, allowing the use of established image analysis algorithms.

These issues became apparent during the production of a video conferencing tool designed initially to research automated anchor creation in a collaborative environment. The tool was designated OvalTine (oval-tine) due to its use of oval tracking for face identification and the flexibility of video streaming capabilities, resulting in networking maps with many tines, as are found on a fork.

LINKS AND ANCHORS IN VIDEO STREAMS

Our work on OvalTine began with the desire to create a hypermedia system in which video streams could be first-class data, that is, data in which link anchors could be embedded and links followed. The goal was to allow a viewer of a video stream to establish a link anchor on an object in the video frame, and as the object moved around the frame, the link anchor moved as well, tracking the object that represents it. The viewer could, at any time after creation of the link, click on the anchor and retrieve the linked information. In this way, an object in a video stream could be linked to text, to sound, even to another video stream or to an object in a video stream.

Our initial plan was to use video anchors and automated tracking to allow links to be created in video teleconferencing. This goal required us to find and employ efficient image analysis algorithms so that link anchors

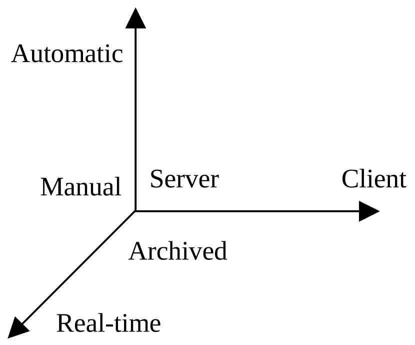

Figure 1: Dimensions of anchor generation in hypervideo

could be tracked in real-time, at frame rates of at least several per second. As the work progressed, it became apparent that the automated tracking methods we employed were useful for stored video streams as well as real-time streams (these applications are discussed later in the paper).

THE OVALTINE SYSTEM

OvalTine is a dynamic, anchor-creating video-conferencing application based on the *mediaConf* example application from Silicon Graphics for their O2 workstations [19]. *mediaConf* is based on SGI's *dmedia* libraries, which provide the basic framework for a server/client video conferencing system. This system is highly configurable, with forking of a video stream possible at any node, regardless of whether it is a video source, receiver, or relay. *mediaConf* includes a simple collaborative drawing environment which has drawing primitives such as lines, boxes, ovals, and a freeform pen. To implement tracking of video collaborators' heads, we added a real-time video analysis engine from Stan Birchfield at Stanford University which uses a mixed-mode analysis algorithm to search for ovals, such as human faces, in a video field [4]. The found faces are denoted by ovals sent across the network as drawing elements of *mediaConf*. These elements are further extended with the addition of hyperlink data. In our case, we chose simple ASCII URLs as the base case. When the user activates a hypervideo anchor, the corresponding URL information is merely displayed in a text box within the UI.

OvalTine was developed on a pair of Silicon Graphics O2 workstations, running IRIX 6.3 OS, connected with 10bT Ethernet over the standard network within the Computer Science Dept at the University of North Carolina at Chapel Hill. No special configuring of the network was done, in order to simulate more closely a natural user environment. The video input was supplied from the standard O2 video camera through the integrated ICE video hardware grabbing board. The *mediaConf* software was modified to accept the addition of the *HeadTrackerLib* library. *HeadTrackerLib* was a substantial rewrite of the *headtracker* application provided by Birchfield *et al.* on the Windows platform. All Win32 specific code was removed and replaced, resulting in a cross-platform image tracking library [20]. The *multip* C++ threading package from UNC's Graphics Lab division was used to implement our image tracking and analysis computations in a separate thread.

A standard video conferencing session in OvalTine proceeds as follows. The same application binary is started on both machines. Each user registers his or her machine with the other through the UI, which simply asks for a TCP/IP accessible machine name. Each user then selects the other machine to be its video server, so each machine is seeing the other's camera view.

In order to generate anchors, each user selects the image of the other user using an enhanced variant of the supplied drawing tools, shown in Fig 2. This triggers the building of an internal model of the image coming from the server. That model is then used to track the other person's image. The anchor region can be made visible, or not, depending on the user's preference. If it is visible, it appears as an oval overlaying the video image.

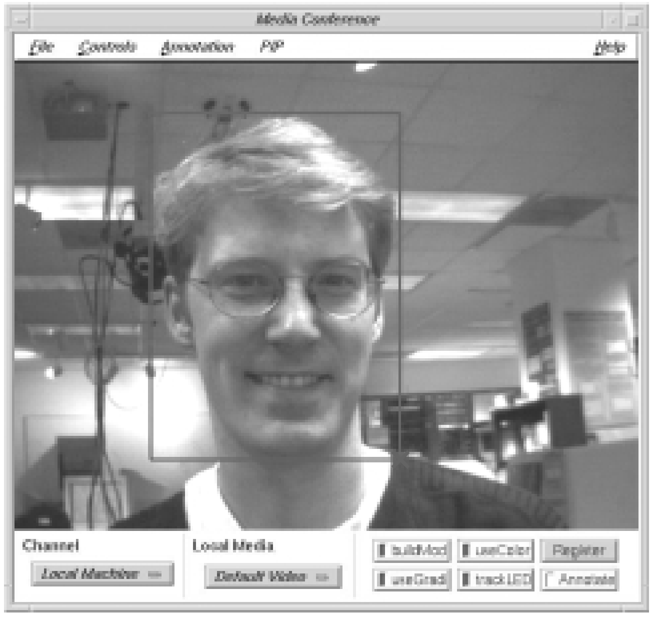

Figure 2: Selecting an area to track a face

Each machine can also act as a server of anchors, by the user selecting his or her own camera's video stream, and operating on it. Only one video stream at a time can actively be seen and have anchors generated. Thus in the current OvalTine system, a machine can act as a server-side anchor generator, or a client-side anchor generator, but not both simultaneously.

Once an anchor is established in the video frame, a link to other data can be made (to text annotations, images, web pages, other video streams). Clicking on an anchor at any time that its video object is visible in the video frame, regardless of how it has moved since anchor creation, will retrieve the linked information as one expects in a hypermedia system.

The OvalTine application was designed to exercise several key points in real-time hypervideo: 1) server-side vs. client-side anchor definition, 2) anchor propagation across a network, 3) acceptable performance for, at a minimum, near real-time execution, 4) flexibility in linked materials, 5) manual and automated anchor creation.

During development, however, it was apparent that OvalTine and its design issues were more extensible than was first thought. It also became obvious that we lacked a sufficient set of definitions for adequately discussing the research directions we wished to pursue. Subsequent background research unveiled nothing that met our needs. Because of this, we developed the taxonomy to provide us with not only a vocabulary for OvalTine, but also a well-formed set of directions for future research projects. OvalTine has provided the first step towards a rich research tool investigating the three axes of hypervideo anchor generation described earlier.

As such, we chose five example systems from current and future research directions, culminating in a business-environment oriented video-conferencing system. This provided an atmosphere that demanded all of the above requirements, as well as creating an example whose benefits can be immediately recognized by the users. These scenarios present our directions for continuing research in hypervideo linking.

EXTENDED HYPERVIDEO CONFIGURATIONS

The three axes we defined back in Fig. 1 provide a rich space in which to frame our example configurations. We specify five such cases here, ranging across the breadth of server/client and manual/automatic, with each selecting a position along the real-time/archived axis as appropriate. We are less concerned with this third axis for the moment, as the distinction between real-time or archived data is primarily one of the run-time characteristics of the image analysis required for a particular application. Our system and initial research direction lent themselves well to a real-time approach, but are, in theory, equally capable of using archived data.

Four of these models are each well defined within a single octant of our problem space, while the fifth is an interesting example of a real-time hybrid approach that is very applicable to current use. The range for each example will be illustrated with a continuous span on each of the three axes in parallel. We chose continuous ranges instead of binary positions to accommodate applications that span various styles. This will become apparent in the fifth example.

Server-side, Manual Anchors

Peter is teaching a distance learning class over the Web. Each class is broadcast by live streaming, and is also archived for later retrieval and review by students. Peter uses the whiteboard extensively for diagramming, but wants to be able to link this directly to the online texts which he has prepared. The lectures tend to be free-flowing based on feedback from the class members, so pre-selected diagrams with preset bibliographic data will not do. Also, as Peter walks around the lecture area, the camera follows him for best effect. The diagram on the whiteboard moves around the video image, so statically defined video regions are not appropriate.

Instead, Peter diagrams on the board, then adds active anchors to the corresponding objects on the video while lecturing. (A small laptop showing the outgoing video feed has been placed on the lectern for this reason.) The diagram regions with anchors attached to them are tracked as they move in the video frame. Every student can click at any time on a diagram object and be presented with the relevant linked document. In this way the students are kept up to date on the necessary accompanying texts, but can refer back to any previously referenced text in an intuitive way.

The advantages of this scenario include single-source control of anchor creation, single-point image analysis load, and minimized total cost of analysis resources over a large and wide network. Disadvantages include limited anchor creation, which may not match the needs of the client user, and a large overhead for the creation of a richly linked environment.

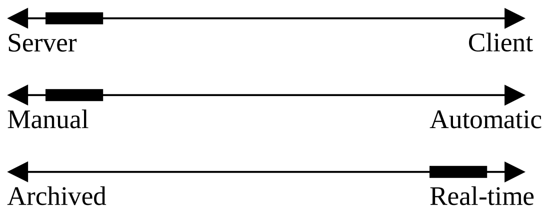

Figure 3: Tracked whiteboard in taxonomic space

When multiple servers are involved, each one acting as a client to the others, a peer-to-peer system is created. This would be analogous to a traditional video conferencing system, where each node acts as both producer and consumer of video streams. During the video production process, however, we add the ability to include hyperlinks as noted above.

Fig. 4 illustrates this type of architecture. OvalTine was configured for this type of system, if not this exact situation, by registering the *Server* machine from the *Client* machines. The *Client* machines then received the anchor embedded video stream solely from *Server*. The *Server* displayed only its own source video.

The video stream was analyzed on *Server* to find human faces, and the anchor information along with the region data was passed to *Client1* and *Client2* as an extension to the supplied collaborative drawing layer included in *mediaConf*. A user sitting at either *Client1*, *Client2* or *Server* had access to the linked data (in this case, simple URL support for the user's home page). Performance was measured on *Server* as 10-15 FPS after tracking was initiated for one face, and performance on the client machines was nominal as compared to when no image tracking was being performed, as one would expect.

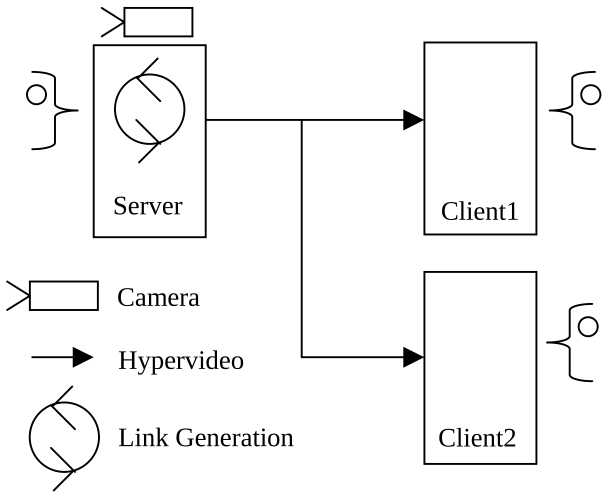

Figure 4: Server-side real-time manual

Server-side, Automated Anchors

FOX Television decides to augment their NHL hockey broadcasts with a new feature for Web deployment: players' statistics on demand. A viewer will be able to click on any player currently in the video stream, and have instant access to their career statistics, game statistics, and any other information relevant to the current situation.

Because FOX already has experience with tracking the puck in real-time and highlighting it on-screen through their FoxTrax system [13], they decide to extend a similar system to the players. Each player will wear a radio transmitter that uniquely identifies them, and a series of radio receivers stationed around the rink will pick up and triangulate the signals. The video cameras are also registered in the rink space, so that a properly calibrated system will know to a fair degree of accuracy where each player is on-screen. An image analysis system will then refine the player's position within the current frame by searching for edges, jersey numbers, etc. Now that the image regions are specified, URL data can be tagged to each, and the user's browser handles the user events to display the requested data and video via standard mechanisms such as JavaScript.

Fully automated anchor generation is similar to a server-side system, where a single analysis engine is used to inject anchors into the video stream. An automated system, however, has the added benefit of being a repeatable behavior without the need of a human operator. Only particular situations will properly be addressed with this setup, but for those requiring it, it is a powerful tool.

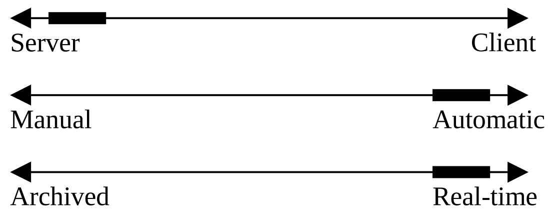

Figure 5: FoxTrax for players in taxonomic space

Note that we do not specify the automated system linking content to video objects. Much work has already been done in this area [6, 11, 22, 23], we are simply investigating the ramifications of various types of anchor creation environments. Automated systems are but one of the possibilities.

In our example, we address only automated anchor generation in a real-time system, but this approach also has merit in other areas. In-depth analysis of archived data in such areas as law enforcement, for example, where a crowd at a suspicious blaze could be scanned using facial recognition algorithms and a database of known arsonists. Multiple individuals could be tracked through the event as needed.

Fig. 6 illustrates this architecture. OvalTine's application design already performs much of the functionality of a fully automatic system. Fully automated image analysis engines under will be used in further research.

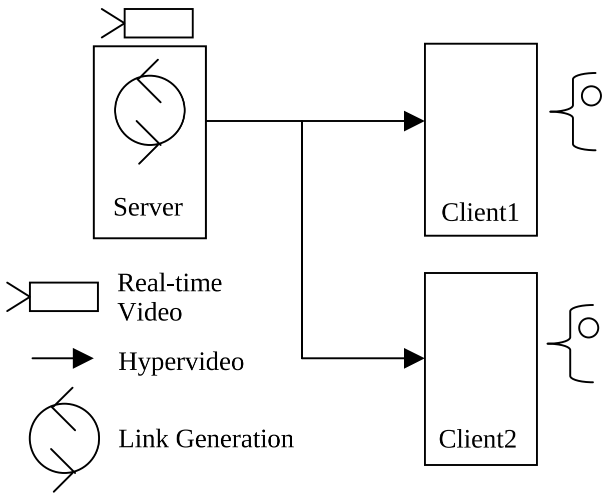

Figure 6: Server-side real-time automated

Client-side, Manual Anchors

Jane is an anthropologist on a team studying urban walking patterns in downtown Seattle. A series of cameras have been placed at various points throughout downtown and are broadcasting over the Web, allowing her a real-time view of pedestrian flow. The camera placement was carefully designed to create a large area of coverage with minimal stitching at the borders of the fields of view. This facilitates tracking one individual or group of pedestrians through the larger region, since the software knows where one image view's edge is registered with respect to its neighbors.

The system allows Jane and the other researchers each to select a person to track from either the real-time view or previously archived video, and tag them with data such as date, time, weather conditions, or whether they exited a parked car or bus. The team can then let the image analysis software compile the information and data for them. The accumulated data from all researchers will be analyzed at a later date. Once the data has been collected, the video is no longer needed, and can be disposed of.

A client-side anchor creation system resembles a dynamic pre-recognized search for information on the part of the user. This approach offers immense flexibility at the expense of requiring the user manually to add anchor information.

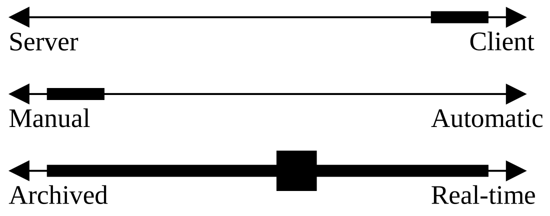

Figure 7: Urban walking patterns in taxonomic space

It has long been recognized, however, that user-defined anchors are often a superior approach to content linking [9, 2], and provide a natural complementary approach to a fully automated system.

Fig. 8 illustrates this architecture. Registering the machine *Server* as in the first configuration set up OvalTine's client-side linking system, but now the user at machine *Client1* performed the anchor creation. *Client1* performed the image analysis to track heads designated by the user and the user then attached the anchor data relevant to him or her.

At no time can one client of the video stream access another client's locally created links. Each client has a private and inviolate database of links particular to that client's desires or needs.

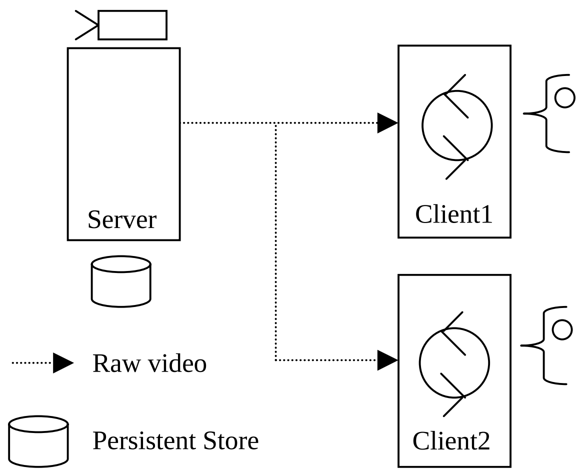

Figure 8: Client-side archived manual

Client-side, Automated Anchors

A series of suspicious fires have plagued suburban Chicago, and the FBI is certain that they are the work of one or two arsonists. In an effort to reduce the number of possible suspects, they have acquired video footage of the crowds surrounding each fire, and are in the midst of cross-referencing them. This is a massive task, however, and is beyond the capabilities of one machine.

In an effort to reduce the cost required by a single server capable of such a job, they have designated multiple client machines, each addressed with the task of finding and identifying only a few persons within a small area of the initial image. Once people are acquired and identified, tracking is a relatively low-cost task that can be done over the entire frame region by each machine.

In this way, the investigators are able to automatically cross-reference hours of video footage from many crime scenes and produce a concise and relevant suspect list.

As an alternate illustration, consider another scenario from broadcast news. In an effort to make their newscasts more appropriate for Web broadcast, CNN has decided to create anchor-rich context within the video. While an off-line analysis engine is appropriate for archived video, it is unlikely that a sufficiently powerful platform will be found to produce all the content needed on one single machine.

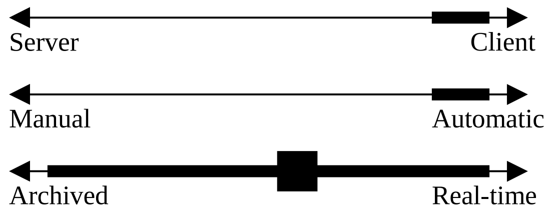

Figure 9: Video scanning in taxonomic space

Instead, live video will be fed from a single server to multiple client machines, each acting within a particular context to analyze the audio and video. One may be translating the audio into closed captioning text, another may be identifying persons shown, paying special attention to those in close-ups, while another may be performing database searches based on the results of the other two engines to create the appropriate links on the anchors identified.

We have now taken the idea of client and server and pushed into the realm of distributed computing, a natural progression. The *Server* is providing the same video stream to multiple *Clients*, but is now allowing each *Client* to perform the desired context analysis.

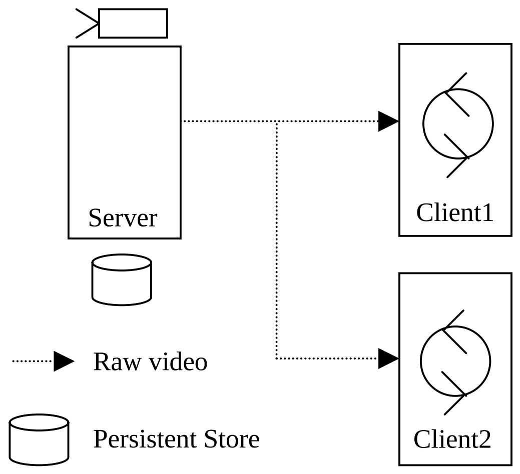

Figure 10: Client-side archived automated

Fig. 10 illustrates this architecture. A future extension to OvalTine would allow it to be configured for this situation by setting up a multiple *Client* system as in the previous model, but as the context of the surrounding video changed, the URL associated with the person might change.

This scenario requires image analysis algorithms more powerful than those with which we are currently experimenting. In particular, it requires automated image recognition and image comparison/matching capabilities. As researchers in this area develop them, however, we can incorporate them.

Hybrid Scenario

Joe is in an important video conference with a prospective client. The software on his workstation is tracking Joe's face, and transmitting an associated URL to request his personal biographical data from his website. During the course of the conference, a new team member walks into the view at the remote site. Joe is positive he should remember her name, and she greets him as if they have met. Without panicking, Joe calmly clicks on her face and looks quickly at the data that comes up, transmitted from the client company's personnel database. He promptly says hello to

Sally, never realizing that she also performed the same action to remember his name.

As the conference continues, Joe wishes to make notations on each new team member, and have them always available for future video conferences. Joe begins an annotation session, where the notes are directly linked to the individual's face, much as the original URL was. Joe's workstation software is creating a hybrid pool of data, taking the video region selection and URL sent by the peer server at Sally's end, and adding a URL to Joe's own notes. This chained data will be saved to disk. During a future session, Joe's software will, upon a click on Sally, dutifully send a request through the URL supplied by Sally's workstation, and in addition will look up the relationship between Sally's URL and the URL to the local annotation data, displaying both.

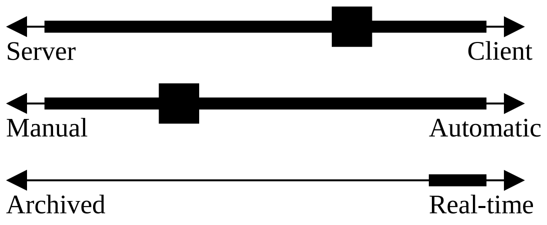

Figure 11: Business video conferencing in taxonomic space

This is perhaps one of the most interesting uses of hypervideo, a blend of pre-selected data anchors and user defined anchors of personal interest. In the above case, Joe has previously chosen his own biographical data to accompany his video to the various sites, and in addition, Joe has been adding information to the links received from the peers. Both Joe and Sally were recognized by the software on their respective servers, identified as to who they were, and the bibliographic information was automatically linked to their faces in the video stream. This is a one-time detailed analysis, as simpler object tracking algorithms can be used as each participant stays within the image.

Note that this is not just a multiple peer server configuration, where every node is a server to the others, but no client data is being added. That is more properly a subset of the server-side system described above, with the natural extension of multiple servers.

Fig. 12 illustrates this architecture. OvalTine could, in the future, support this scenario by merging the server-side and client-side systems as described above. A multiple server system would be instantiated by registering each of the nodes with each other, such that every node is a potential source for any other node. Currently only one source can be viewed at any time, but the selection can be changed to any of the registered servers. (A picture-in-picture mode is available, but only the main image is capable of carrying hyperlinks.)

By selecting the local video source as the display, the user could choose the data to be transmitted with his or her tracked face. By selecting a remote video source from the server list, the user could attach additional data to the incoming link. A single UI button *'Add URL'* would accomplish both tasks, intercepting the video stream internally.

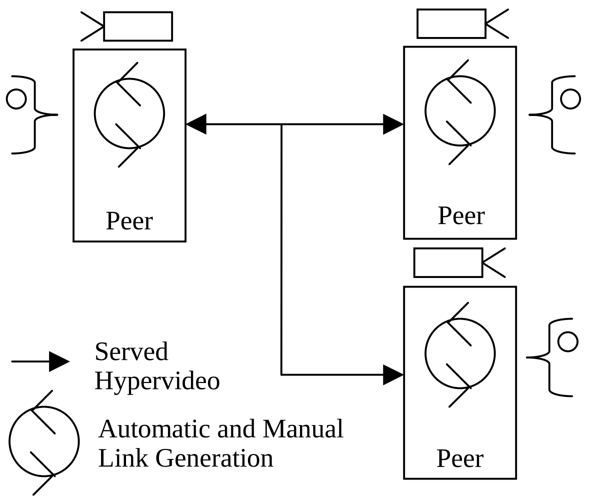

Figure 12: Hybrid real-time

CONCLUSIONS

In the process of implementing a video collaboration system involving anchor creation, we identified three key orthogonal issues in anchor creation for hypervideo. These issues are particularly relevant to the attributes of hypervideo as opposed to other more static media such as text.

The three dimensions of server/client anchor creation, manual/automated anchor creation, and real-time/archived raw data provide a rich definition space in which to address future work.

While most current hypermedia falls into a small subspace of the region defined by these axes (static + manual + archived), hypervideo has the capability of spanning the entire spectrum. With new capabilities come new requirements, approaches and restrictions. It is our intent to continue investigating those boundaries.

OvalTine is our primary tool for such research, currently providing exclusively server-side or client-side manual anchor generation, but was designed to accept fully automatic image analysis engines now under development. OvalTine is implemented primarily as a real-time experimental platform. Adding the capability to work with archived data is a fairly trivial change, and is currently underway.

ACKNOWLEDGEMENTS

Thanks to Hans Weber for use of his *multip* thread package in the production of OvalTine, and to Matt Cutts for critical reading.

REFERENCES

1. Apple Computer, Introduction to Wired Movies, Sprites, and the Sprite Toolbox, http://developer.apple.com/techpubs/quicktime/qtdevdocs/REF/refWiredIntro.htm

2. Bernstein, M., An apprentice that discovers hypertext links, Hypertext: Concepts, systems and applications: Proceedings of the European conference on Hypertext, INRIA, France, 1990, pp. 212-223.

3. Bernstein, M., J. D. Bolter, M. Joyce, E. Mylonas, Architectures for Volatile Hypertext, Hypertext ‘91 Proceedings, ACM, San Antonio, TX, 1991, pp. 243-260.

4. Birchfield, S., Elliptical Head Tracking Using Intensity Gradients and Color Histograms, IEEE Conf on Computer Vision and Pattern Recognition, Santa Barbara, CA, June 1998.

5. Boissière, G., Automatic creation of hypervideo news libraries for the World Wide Web, Hypertext ’98 Proceedings, ACM, Pittsburgh. PA, 1998.

6. Burril, V., T. Kirste, et al, Time-varying sensitive regions in dynamic multimedia objects: a pragmatic approach to content-based retrieval from video, Information and Software Technology Journal Special Issue on Multimedia, Vol. 36, No 4, Jul 1994, pp. 213-224.

7. Elliot, E. and G. Davenport, Video Streamer, CHI ’94 Proceedings, ACM, 1994, pp. 65-66.

8. Engebretsen, M., Hyper-news: revolution or contradiction?, Hypertext ‘97 Proceedings, ACM, Southhampton, UK, 1997, pp. 222-223.

9. Glushko, R.J., Design issues for multi-document hypertexts, Hypertext ’89 Proceedings, ACM, Pittsburgh, PA, 1989, pp. 51-60.

10. Grobe, M. An Early History of Lynx: Multidimensional Collaboration, http://www.cc.ukans.edu/~grobe/early-lynx.html

11. Hampapur, A., et al, Digital Video Segmentation, Multimedia ’94 Proceedings, ACM, 1994, pp. 357-364.

12. Hirata, K., Y. Hara, H. Takano, S. Kawasaki, Content-oriented Integration in Hypermedia Systems, Hypertext ’96 Proceedings, ACM, Bethesda, MD, 1996, pp. 11-21.

13. IEEE Computer Graphics and Applications Vol. 17, No 2. March - April 1997, Applications: The FoxTrax Hockey Puck Tracking System

14. Ip, H. H., S. Chan, Hypertext-Assisted Video Indexing and Content-based Retrieval, Hypertext ‘97 Proceedings, ACM, Southhampton, UK, 1997, pp. 232-233.

15. Kendall, R. Hypertext dynamics in *A Life Set for Two*, Hypertext ’96 Proceedings, ACM, Bethesda, MD, 1996, pp. 74-84.

16. Liestøl, Gunnar, Aesthetic and Rhetorical Aspects of Linking Video in Hypermedia, Hypertext ’94 Proceedings, ACM, 1994, pp. 217-223.

17. Nürnberg, P. J., J. J. Leggett, E. R. Schneider, As We Should Have Thought, Hypertext ’97 Proceedings, ACM, Southhampton, UK, 1997.

18. Sawhney, N., D. Balcom, I. Smith, HyperCafe: Narrative and Aesthetic Properties of Hypervideo, Hypertext ‘96 Proceedings, ACM, Washington, D.C., 1996, pp. 1-10.

19. Silicon Graphics, Inc., mediaConf videoconferencing, https://toolbox.sgi.com/src/demos/O2/mediaConf/, July 1997. Note: Requires a free registered login and password for SGI’s developer program.

20. Smith, J. McC., HeadTrackerLib: Generalized head tracker, http:://www.cs.unc.edu/~smithja/HeadTrackerLib/, Oct 1999.

21. Stotts, P. D., R. Furuta, Dynamic adaptation of hypertext structure, Hypertext ’91 Proceedings, ACM, San Antonio TX, 1991, pp. 219-232.

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

23. Zhang, H.J., C.Y. Low, S.W. Smoliar, J.H. Wu, Video Parsing, Retrival and Browsing: An Integrated and Content-Based Solution, Multimedia ’95 Proceedings, ACM, 1995, pp. 15-24.