TV Watcher: Distributed Media Analysis and Correlation *

David Hilley, Ahmed El-Helw, Matthew Wolenetz, Irfan Essa, Phillip Hutto, Thad Starner, and Umakishore Ramachandran
College of Computing
Georgia Institute of Technology
801 Atlantic Dr.
Atlanta, GA 30332
{davidhi, ahmedre, wolenetz, irfan, pwh, thad, rama}@cc.gatech.edu

ABSTRACT

The explosion of available content in broadcast media has created a desperate need for applications and prerequisite system architectures to support automatic capture, filtration, categorization, correlation, and higher level inferencing of streaming data from distributed sources. We present *TV Watcher*, an archetypical example of such an application. *TV Watcher* performs user-controlled correlation of live television feeds and allows the user to automatically navigate through the available channels based on content of interest. We introduce the *Symphony* architecture for distributed real-time media analysis and delivery to meet the system requirements for applications with such needs. *TV Watcher* is built on top of the *Symphony* architecture, and currently uses closed-captioning information to correlate television programming. We present the results of a user study that shows the correlation engine is consistently able to pick significantly useful and relevant content.

Categories and Subject Descriptors

D.4.7 [**Operating Systems**]: Organization and Design—*Distributed Systems*; H.3.1 [**Information Storage And Retrieval**]: Content Analysis and Indexing

1. INTRODUCTION

Recent trends in computing and entertainment technologies have enabled users to gain access to an overwhelming amount of media. Choosing a single source is increasingly limiting since television, radio, and now the Internet provide so much constantly-available content. Television guides feature extremely limited and potentially out-dated information about the true content of shows, and many programs are broadcast live, making it impossible to discern most content before viewing. A system to support the automatic capture, filtration, categorization, correlation, and higher level inferencing of streaming data from distributed sources based on user interests is needed to combat the growing problem of information overload.

In this paper we introduce *Symphony*, a new distributed architecture designed to facilitate the real-time analysis of live, high-bandwidth media streams. Using our prototype of *Symphony* as a framework, we have implemented an application called *TV Watcher*, which provides users with a mechanism for navigating through a large number of media streams based on their interests. The *Symphony* architecture is general enough to support a multitude of different applications. For example, the *Symphony* infrastructure can be harnessed to create an automated distributed surveillance system. *Symphony*'s architecture is also well suited to applications such as the *Event Web* [12], which monitors and analyzes sensor data from sensor rich environments.

Figure 1: TV Watcher: The "Preview Mode"

There are several important aspects of *TV Watcher* that are best supported by the distributed analysis and delivery architecture of *Symphony*. For example, *TV Watcher* must provide a reasonable comparison of the semantic content of video streams in real-time. Architecturally, latency and stream synchronization are important especially with live media streams, and the sheer amount of data required for high resolution video streams necessitates an efficient transport mechanism. In addition, any reasonable analysis of high-bitrate media streams is resource-intensive. The computational cost of performing expensive analyses on live, high-bandwidth streams is mitigated by *Symphony*'s distributed architecture.

The contributions of this paper are threefold:

1. We present the *Symphony* architecture, an infrastructure for the distributed real-time analysis and delivery of media streams.
2. We introduce *TV Watcher*, a novel application built on *Symphony* for user-controlled correlation of live televi-

*The work has been funded in part by an NSF ITR grant CCR-01-21638, NSF grant CCR-99-72216, HP/Compaq Cambridge Research Lab, the Yamacraw project of the State of Georgia, and the Georgia Tech Broadband Institute. The equipment used in the experimental studies is funded in part by an NSF Research Infrastructure award EIA-99-72872, and Intel Corp.

Figure 2: TV Watcher: The "Correlation Mode"

Figure 3: TV Watcher: The Web Search Plug-in

sion feeds.

3. We present and analyze the results of an experiment conducted to assess the usefulness of the correlation engine in *TV Watcher*.

In the upcoming sections we provide technical details of the *TV Watcher* application (Section 2), the *Symphony* architecture (Section 3), the correlation engine (Section 4), and the details of our experiments (Section 5), as well as future correlation work (Section 6) and related work (Section 7).

2. THE *TV WATCHER* APPLICATION

TV Watcher allows a user to navigate through multiple televised video streams and identify currently relevant content. First, a user will select a video stream of interest (the **target stream**) from the set of available sources. The chosen stream is then correlated against all available streams (or a user-selected subset) and the results are used to display available content strongly related to the target stream. The information used for correlation can also be used to locate relevant webpages via the *TV Watcher* Web Search Plug-in.

TV Watcher's user interface currently supports two basic modes of operation: **preview mode** and **correlation mode**. After starting the application, a user selects a stream of available media that is interesting to him or her. Correlation mode can be entered by double-clicking on a stream of interest or selecting the "Correlation" option from a chosen stream's options menu.

***TV Watcher* Preview Mode:** Pictured in Figure 1, *TV Watcher*'s preview mode shows all of the available media streams at thumbnail size (and full frame rate if possible). In addition, the closed captioning text of a single stream is displayed in the lower portion of the interface – the selection is changed by clicking on a video window. Each individual media stream may have many configurable parameters (based on the limitations of the media provider), and a per-stream popup menu enables the user to change these parameters. For instance, the capture servers utilizing the Brooktree capture card (many consumer-level video capture cards use Brooktree chipsets) allow the user to select the cable television tuner or video input, and the video can be streamed at a custom resolution as uncompressed raw frames or motion-JPEG compressed video. When necessary, the user can also change the current channel of a television stream.

***TV Watcher* Correlation Mode:** In correlation mode, *TV Watcher*'s interface is as shown in Figure 2: the user-selected stream of interest is now displayed at full size, and thumbnail-sized video streams from the other sources are arranged around the stream of interest. Each video window has a colored border with a label; the color is a visual indicator of the correlation strength and the label gives the numeric percentage. This correlation measure is currently based on the content of a stream's closed captioning text. Additionally, the most important words (as determined by our text correlation algorithm) in the stream of interest can be sent to a plug-in application that performs web searches on these terms.

***TV Watcher* Web Search Plug-in:** The Web Search plug-in for *TV Watcher*, pictured in Figure 3, is intended to provide additional information when presenting the correlation results. This application performs limited web searches on the most important words appearing in the stream of interest (the algorithm for finding these words is covered in Section 4) and displays the top results in embedded browser windows. Although the plug-in is intended to run simultaneously with *TV Watcher*, it can also be used to harvest links relevant to keywords in documents.

3. THE *SYMPHONY* ARCHITECTURE

Figure 4 shows the elements of the *Symphony* architecture. The five entities of the system are as follows: capture servers, stream servers, the media analysis engine, clients, and the distributed programming framework. In the system diagram, each rectangle represents one of these components, all of which interact through the distributed programming framework. The media providers, called the **capture servers**, simply provide various media streams for use. A **stream server** provides stream location facilities including searching and filtering based on metadata. The media analysis system runs on a cluster of workstations and handles the analysis, correlation and interpretation of media; the type of

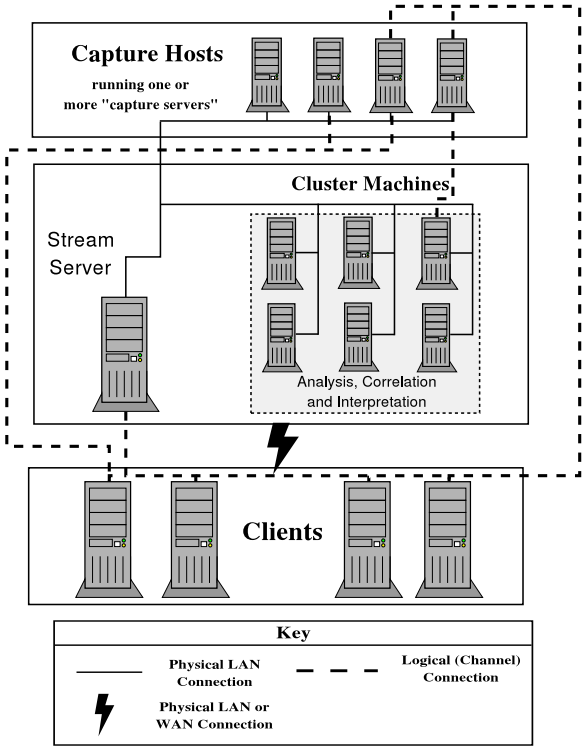

Figure 4: A configuration of the *Symphony* architecture

analysis performed is, of course, specific to the application using *Symphony*. The clients utilize media from the capture servers and the results provided by the media analyses: multiple clients may exist in the system at any given time, and can share media and correlation resources via *Symphony*. Finally, the distributed programming framework provides the glue connecting all of the other components.

In our tests, the clients are easily run on machines only connected via a consumer-level Internet connection (1.5Mbit / 256Kbit ADSL). The connections between the "cluster machines" should be high-speed links to facilitate the distributed computation, and the connections between **capture servers** and the cluster machines should be relatively fast if the streaming media is bandwidth-intensive.

The dotted lines represent logical connections between the different entities of *Symphony*. Although the stream server is logically a central point of control in the system, the actual media streams are fully distributed. The clients receive their streams of interest directly from the appropriate capture server and only use the stream server to locate the appropriate media. Likewise, the cluster machines directly retrieve the necessary media streams from the capture servers. The lack of a centralized media provider allows the cluster machines and clients to receive different streams from the capture servers; this is extremely useful because the cluster machines may require higher resolution streams for media analysis, while the clients show thumbnail media streams or aggressively compressed content (as may be necessary for servicing clients across lower-bandwidth Internet links).

Capture Servers: The capture servers provide media streams to interested entities in the system and are the sources of data for distributed analysis. Additionally, the hosts running the capture servers may be physically located outside of the media analysis cluster and may depart the computation at any time. If real-time analysis of streams is to be performed, however, it is advantageous if the media providers are physically located on the same network as the media analysis machines (reducing latency and increasing host-to-host bandwidth). The ability to add and remove stream producers (and consumers) at any time is key to the flexibility to the system. A single capture server may provide multiple media streams – for instance, in the standard testing setup of *TV Watcher*, each capture server provides a stream of audio, video and text from the same source (broadcast television or video input).

Capture servers also provide static and dynamic meta-data used to locate and classify media streams. Upon startup, a capture server will provide some static stream information to the stream server which is used to by clients to locate relevant streams. For instance, the supported output resolutions, available video sources, supported compression methods, and other capabilities are examples of static meta-data. Dynamic meta-data varies based on the stream source but frequently includes attributes like the current television channel, video source, output resolution, compression statistics and the name of the content (if available).

Stream Servers: The stream server is a central component of the *Symphony* system and provides an interface between media producers and consumers as well as a mechanism for controlling distributed media analysis. Although the system diagram shows the stream server located on the same high-speed local network as the cluster machines, this is not an architectural requirement (in fact, the stream server uses relatively small amounts of bandwidth): our prototype implementation of *Symphony* simply assumes this particular configuration.

Different distributed entities contact the stream server to locate media streams, and the stream server's internal list of available content contains meta-data which may be used to locate appropriate entries. Each capture server maintains a status register that provides the dynamic stream meta-data (such as current television channel, resolution, source, etc.). The stream server observes the status of all known capture servers and provides a single data stream of changes to each client.

Media Analysis System: One of the key features of *Symphony* is the ability to perform distributed real-time media analysis. Resource-intensive computation for media interpretation and correlation is distributed over a cluster of workstations, enabling an application to perform many expensive analyses on live streaming media. The details of the media analysis engine are specific to each application using *Symphony*.

Typically, clients will control the parameters of the analysis based on user input: for instance, with the *TV Watcher*, a user selects a specific television program and the system performs various correlation operations to find related content. In other cases, a capture server may utilize the media analysis system directly for gathering statistics about streams.

Clients: Clients in *Symphony* are specific to a particular application and are typically used for user-interaction: the

previously described *TV Watcher* user interface and web search plug-in are both instances of clients. Clients are simply applications within the *Symphony* framework that use the streaming media analysis infrastructure. Client applications are not required to be media consumers, as some may use available media streams indirectly by requesting media analysis results on available sources.

TV Watcher's client is an application that displays the video streams and analyzes the available closed captioning text streams. It only interacts with the stream server upon startup when retrieving descriptions of the currently available media streams. After receiving the static stream information, the application also initializes a thread to monitor the stream server's meta-data channel for updates to streams' dynamic meta-data. Through this mechanism, the *TV Watcher* application can robustly handle a stream provider going off-line while still maintaining connections to the other stream sources.

Distributed Programming Framework: Applications implemented on top of *Symphony* utilize a distributed programming framework that provides the interface between the aforementioned system components. To properly support real-time media analysis applications, this framework must provide distributed transport of stream-based data with high throughput and reasonable latency. In addition, *Symphony* requires high-level abstractions suitable for temporal correlation of media streams, such as time-stamped data items and stream synchronization. As previously noted, the ability to dynamically add and remove stream producers or consumers is also needed to flexibly support many different applications. Our current *Symphony* prototype uses *D-Stampede* [1], a distributed cluster-oriented programming system, but *Symphony* can utilize any transport satisfying the properties described above.

D-Stampede provides a distributed, high-level transport on which the *Symphony* prototype is built. The *D-Stampede* system presents a set of abstractions through which distributed applications share data and synchronize activities between nodes in a cluster of workstations (as well as client devices outside of the logical "cluster"). *D-Stampede* supports globally visible data abstractions called **channels**, which provide for transport of time-sequenced data. Channels are convenient for media streams because they allow retrieval of data based on temporal information. Our prototype also uses **registers** as containers for dynamic status data, such as stream resolution and compression information. *D-Stampede*'s real-time guarantees are also useful for supporting media-oriented applications. Finally, *D-Stampede*'s support for heterogeneous computing environments adds flexibility to *Symphony*.

4. TEXT CORRELATION

The *Symphony* architecture provides a general real-time media analysis infrastructure, but one of the most resource-intensive and useful forms of analysis is stream correlation. While gathering statistics about a single stream is also possible, *TV Watcher* requires the comparison of multiple video streams. Since the general goal of the application is automatic selection of content suited to a user's viewing preferences, comparing a "reference stream" to other available media sources is a natural way of finding other interesting content.

Because *TV Watcher* focuses on standard television streams, the natural components to consider for correlation are video, text and audio. Currently, text correlation is the most complete form of correlation implemented in the application: closed captioning text is decoded and analyzed as a text document (with timing information). Video correlation techniques are currently being developed and integrated into *TV Watcher*. Efforts on other modes of correlation are covered in Section 6.

Although audio and video are promising candidates for correlation of television streams, textual transcripts are a simple starting point and provide a wealth of easily-extracted semantic information. Text correlation algorithms such as TF/IDF (term frequency / inverse document frequency) [20] are well established, and the popularity of the World Wide Web fuels continual development and refinement of text correlation techniques. As noted earlier, the text correlation results can also be used to search for relevant documents on the World Wide Web.

The most obvious form of text available with television media streams is closed captioning. Not all television shows are captioned, but captioning is an increasingly common practice; in fact, the FCC requires standard, over-the-air broadcasters to meet captioning quotas during a majority of the day (6am to 2am) [7]. Even live news broadcasts are often captioned, though mistakes (misspellings, omissions, etc.) do occur and may affect the quality of the correlation results.

Correlation Algorithm: The text correlation system utilized in *TV Watcher* is a flexible, stand-alone correlation engine based on the standard TF/IDF algorithm [20]. Initially, the system calculates the frequencies of each term in the two text streams being compared and creates a dictionary of all relevant terms. Next, the inverse document frequency is calculated and normalization factors and weights are determined. The final computation involves taking the dot product of the weighted term vectors to find the overall similarity of the two documents. The resultant value provides a relative measure of the pairwise correlation between two text documents. Also, since the process includes weighting of terms, we can easily obtain the most heavily weighted terms that contribute to this similarity; ideally, the heavily weighted terms represent the most "important" words in a document.

Correlation Implementation: In addition to the basic algorithm above, our correlation engine implements several enhancements to improve the results. First, we use a **stoplist** to remove words that provide little semantic meaning or are extremely common in everyday speech. We have also integrated the Porter stemming algorithm [16] to reduce words into their stems (for instance, "eating" would be stemmed to "eat").

One of the challenges inherent in *TV Watcher* is maintaining usefulness and relevance of correlation results over long time periods. For instance, after capturing hours of text, the results of a comparison between channels may be unreasonably biased by history (much of which is irrelevant to the current content). To avoid this problem, our correlation engine allows correlation queries based on limited windows of time. For instance, a user could request the correlation score obtained by comparing the last 30 seconds of CNN with the last 2 minutes of CBS. This process can be exploited in the

future to allow for correlation of a particular prerecorded video clip against material on live television. Additionally, our correlation engine admits multiple simultaneous queries over a particular dataset. Overall, these specific features of our implementation allow for a highly flexible text correlation engine for *TV Watcher*.

Closed Captioning Text: While the text correlation algorithm implemented in *TV Watcher* is designed for generic document correlation, several issues specific to closed captioning affect the correlation process. Closed captioning text is digital data encoded in an undisplayed portion of the video content. Although closed captioning text is generally a good transcript of the audible dialogue (as well as important sound effects), there are several problems inherent to the closed captioning system. First, since closed captioning is transmitted as analog video data, the quality of the decoded text is dependent on the quality of the video feed: noise and interference on the television feed will cause errors in the closed captioning. The standard for closed captioning transmission includes only one parity bit for error checking and no method of error recovery, so closed captioning decoders must drop characters which fail the parity check.

In addition to the bizarre misspellings and spliced words created by dropped letters and whitespace, live captioning often suffers from similar mistakes caused by human error (and thus inherent in a correctly received stream). During live news broadcasts, proper nouns are occasionally misspelled and long words may be abbreviated. Since real-time captioning must occur extremely quickly, systems for creating captions are based on gesture recognition [19], so an incorrect or unclear gesture will often cause a completely different word to be substituted, making sentences incoherent. Additionally, captions generated in real-time are almost always entirely in uppercase, and this eliminates semantic information for determining names and other proper nouns.

Pre-generated captions, such as those on commercials, movies and other non-live television programming, are extremely unlikely to contain the aforementioned errors (other than bad decoding due to poor reception). Most "offline" captioning contains proper, mixed-case text and may also include positioning data, color tags and other potentially useful extra information. Although our captioning decoder currently discards color and positioning data, it would not be difficult to utilize it as extra semantic information to improve correlation results when available.

5. CORRELATION PERFORMANCE

In order to assess the usefulness of the *TV Watcher* text correlation using closed captioning, we conducted an experiment to determine the user-rated quality of the correlation algorithm results. We hypothesize that, using the Web Search Plug-in, the existing prototype will retrieve web pages rated as both relevant to a video segment and useful in synthesizing new reports based on the topic that segment. Our experiment design is similar to that of the text correlation experiments in [18].

Experiment Methodology: We selected and recorded ten news stories at random from CNN Headline News: CNN is a particularly good choice for our purposes because the beginning of each new story is clearly delimited in the closed captioning stream. While future versions of *TV Watcher* may segment content using cues from the video and audio, the pre-segmented CNN stories allow us to isolate the effectiveness of the engine without the potential for segmentation errors. The closed captioning text from each story was used as input to the correlation engine; the five most heavily-weighted words were then extracted from each stream and sent to *TV Watcher*'s Web Search plug-in. From these results, we picked the top three web pages reported by our system. As a control, three web pages for each story were retrieved by choosing five random words from the article and using the same search mechanism. Each of the web pages was then assigned a unique ID.

Ten subjects unfamiliar with this project were recruited for our experiment. The subjects were instructed to watch each of the ten news stories in turn; after watching a news story, they were told to rate each of the six retrieved web pages for both relevancy and "usefulness" on a scale from 1 to 7 – weakest to strongest. A webpage is defined to be relevant if it covers the same topics as the news story. Usefulness, in this case, is defined as how beneficial the web page would be if the subject was going to create a report based on the topic of the video news story. The distinction between relevancy and usefulness is important. For example, an exact transcript of the news anchor's dialog would be relevant to the video, but not useful because it presents no new information. Similarly, a webpage with a one sentence summary of the news story is also strongly relevant but not useful.

For convenience, both the video segments and links to the web sites were presented on a webpage personalized for each subject – the order of the hyperlinks was randomized per-subject. Subjects could review a video segment or change their evaluation ratings at any time during the experiment, though most did not. This process required approximately one hour per subject, and the subjects were compensated with $10 for their time. Each subject watched 10 news stories and rated six web pages per news story per metric for a total of 120 ratings per subject.

Experimental Results: On average, the subjects rated the web pages suggested by *TV Watcher* at a relevancy level of 4.6 versus 1.3 for the web pages retrieved by searching on random words from the article. The mean usefulness rating is 4.3 for our engine versus 1.4 for the control. For nine of the ten subjects, the web pages retrieved by *TV Watcher* were rated more relevant than the control with a significance level of $p < 0.00000011$ or less when using an unpaired, two-tailed t-test on distributions with unequal variances. Similarly, these subjects ranked our system's results as more useful with a significance level of $p < 0.000004$. The first subject rated both sets of web pages at an average level less than 2 for both relevancy and usefulness, so the test did not show significance between the two conditions. We believe this anomaly is caused by the test subject's misunderstanding of the directions. It may also be a limitation of our experimental methodology, which can be refined.

Given the above results, we can conclude that *TV Watcher*'s correlation engine is indeed successful in retrieving relevant and useful web pages for users given the closed captioning text of a news segment. In addition, the results suggest that the text correlation engine's selection of the most salient words for the web search is significantly more successful than selecting random words from the article for

the search. While this last result may seem obvious, closed captioning contains many mispellings and noise errors that could have significantly affected the performance of the system.

6. EXTENSIONS TO CORRELATION

6.1 VBI Data

Closed captioning text is transmitted in the vertical blanking interval (VBI) of NTSC video. The vertical blanking interval is the initial, undisplayed 21 lines of video which do not contain picture content: closed captioning text is only encoded in the last line of the vertical blanking interval, leaving extra bandwidth in the VBI for data transmission. The first nine lines of the VBI are reserved for video synchronization, but the other eleven lines frequently carry interesting information. For instance, the 10th line is reserved for silent radio – silent radio carries information frequently displayed on LED boards at bars, airports and hotels such as sports scores, weather information, stock quotes, and lottery numbers. The 12th, 13th and 15th lines are reserved for PBS datacast, and the 14th line is reserved for a television guide service called Starcast. Cable and satellite providers may also place other data signals in the VBI information. Although much of the available data is not relevant to the programming, the most promising non-captioning VBI data is called XDS.

XDS stands for "eXtended Data Service" and is a stream of data transmitted on line 21 of the second NTSC field (closed captioning text is transmitted in the first field). The XDS stream contains various data related to the current program, including relevant web links (utilized by WebTV), the program's name, episode information, ratings, the station's call-sign, and other potentially useful meta-data. Some devices, most notably the "V-Chip", may use XDS information to censor content that is offensive or inappropriate for children [19].

Although the *TV Watcher* application currently discards the XDS information, rudimentary XDS decoding is available in the closed captioning decoder and the capture servers could easily be modified to provide XDS streams. This information could be used to augment the text correlation, and the provided web page addresses could be sent directly to the *TV Watcher* web search plug-in.

6.2 Video Correlation

Video correlation is potentially the most unique and interesting feature of *TV Watcher*: the ability to perform real-time video correlation is explicitly enabled by the *Symphony* architecture. More importantly, video correlation is an archetypical example of the type of intensive media analysis which actually prompted the development of the *Symphony* architecture. Presently, the extensive analysis of multiple real-time video streams is computationally infeasible on a single personal computer. The approach of *Symphony* allows computationally expensive media analysis to be distributed over a cluster of available machines.

The biggest challenge of video correlation is the lack of common or general-purpose algorithms; even the general comparison of still images is an open problem in the field of computer vision. Part of the work in creating *TV Watcher* is developing metrics by which television video streams may be compared. Video correlation is currently under development for *TV Watcher*, and several different algorithms have been implemented but it is still unclear whether video information can be used as a primary method of stream comparison or if it is better suited to simply augment the results of the text correlation. Currently we are exploring video correlation algorithms which add information to strengthen the results of the existing closed captioning correlation.

Figure 5: Face Detection, Extraction and Temporal Density Plots. Top – an image from video with faces marked. Bottom – density of faces over time (left to right) in a stream.

Face Detection and Recognition: Some of the most visually obvious aspects of the television programs are related to faces: face density, frequency, location, size, motion and identity are all properties that provide semantic information about the content of a media stream. Certain types of television programs exhibit specific patterns of face appearances. For instance, daily news broadcasts in which one or more anchors read the news often feature centrally positioned, large faces that typically do not change location. Other heuristics provide semantic information about the content of the program (as opposed to information about the program itself). As a very simple example, high facial density can indicate a crowd such as an audience.

In addition to detecting faces, identifying and recognizing faces can also provide information about the content of streams. For news-related television programming, finding the same face in two different streams often indicates that the streams are related. Immediately following an important political speech, for example, news programs and political commentary shows will typically cover the speech, often showing video clips. In this case, detecting the face of the speaker on multiple channels may indicate related content.

Obviously this mechanism is sensitive to some specific quirks of television: actors and prominent people frequently

covered in many news stories may appear in multiple streams in different, unrelated contexts. Detecting the face of a popular actor in two streams does not necessarily indicate content-level correlation. Similarly, detecting the face of the president on two different channels is not indicative of strongly related content. However, face recognition can be used to augment the results of text correlation, and matching against a database of known news anchors, talk show hosts, actors and other celebrities may provide useful auxiliary information about the network and type of show.

Currently, we have prototype implementations of face recognition and detection algorithms running on video streams and are in the process of integrating them into *TV Watcher*'s correlation engine. Figure 5 shows a standalone prototype application that detects and extracts faces while displaying a graphical representation of face density over time. The face detection algorithm is a fast, free implementation [11] of the Viola-Jones object detection algorithm [24] with a classifier cascade trained for faces. The face recognition is performed by our own implementation of a PCA (principle component analysis) face recognition algorithm [23].

Scene Detection: Scene and program segmentation also provide valuable information to assist the closed-captioning text correlation. One of the factors affecting the accuracy of text correlation is the overall history of a particular stream: the content of live television obviously changes dramatically from program to program, and potentially even between segments of a program (news stories, chapters, etc.). Also, most television networks feature intermittent commercials, which may also be closed captioned, but this text should not be included in the correlation results.

Segmenting television content based on the captioning alone is a difficult problem: although some shows may add delimiting information to the closed captioning feeds, there is no common convention for doing so and it is relatively rare. For example, captioning on CNN news broadcasts typically contains the string "`>>>>`" before each individual news story, but the same string may also appear to separate the dialog of different speakers or in other contexts. On the other hand, commercial detection using video analysis is relatively straightforward. Some consumer video recording products, such as the ReplayTV, feature or have featured extremely accurate commercial detection. Scene segmentation in video is also an open problem, but several reasonably accurate algorithms already exist [10, 17], and we are currently investigating the feasibility of integrating a general purpose one-pass scene detection algorithm into *TV Watcher*'s correlation engine.

Other Video Characteristics: In addition to face recognition and scene detection, many other aspects of a video stream provide semantic information that can be utilized. For instance, motion in video can provide useful clues to the type of program: news programs tend to have less camera motion, causing the background to remain relatively static between editing cuts. Statistics about color content can also be analyzed to predict the type of television content. Cartoons, filmed content and content produced on video all have different color characteristics which are easily recognizable to the average viewer. Other features to consider adding to *TV Watcher* include object detection, gesture/action recognition and figure tracking.

7. RELATED WORK

TV Watcher utilizes the abstractions provided by the *Symphony* architecture. The *Media Broker* [13] architecture provides facilities for distributed stream registration, discovery, and transformation. The functionality of the stream server in *Symphony* is similar to that provided in the *Media Broker*. Our future plans include exploring this overlap and using the higher-level, media-friendly abstractions of the *Media Broker* in future prototypes of *Symphony*. In addition, the *AuraRT* system [4, 5] is similar to *Symphony*, but targeted towards more controlled environments and applications requiring low-latency delivery of streams.

The Informedia project [25] is concerned with video indexing, information extraction and content-based queries: in particular, the News-On-Demand project [9] performs speech-based content queries on a database of stored news broadcasts. Both IBM's CueVideo effort [2] and the Altavista Media Search project [6] attempt to index video streams offline for future searching and browsing by content. Although we are primarily concerned with video, much work has been done on querying still images by content, such as QBIC (query by image content) [14], VisualSEEk [21], and Photobook [15]. Additionally, [22] provides a good overview of current techniques used in multi-modal video indexing systems.

Many of the aforementioned projects use available textual information when indexing media content; some use the text from captioning data while others attempt to transcribe speech present in the audio streams. Closed captioning indexing and segmentation as it pertains to a specific subset of television content (late-night talk shows) is extensively covered in [3]. In [10], closed captioning data is used to segment video content from television news broadcasts, and the related News-On-Demand project [9] uses closed-captioning to extract "keywords" for semantic categorization (similar to the function of the *TV Watcher* web search utility).

8. CONCLUDING REMARKS

Distributed capture, correlation, and dissemination of multimedia streams is an important avenue of system exploration with the potential for use in many application scenarios. In this paper, we have given a concrete instance of such an application, *TV Watcher*, and have presented the requirements of that application. We have developed a distributed architecture, *Symphony*, to support such applications. A prototype of *TV Watcher* has been implemented on the *Symphony* architecture, and we have conducted user studies on text-based correlation and demonstrated its effectiveness in enabling the selection of related content.

Future avenues of exploration for the *TV Watcher* project follow two basic paths: work on *TV Watcher*'s correlation engine, and new developments for the *Symphony* architecture. Advancements in the fundamental architecture of *Symphony* will not only facilitate the development of new distributed media analysis applications, but also improve the potential range of media analyses feasible in the *TV Watcher* application. Potential additions to the correlation system have already been mentioned, and future architectural improvements include building *Symphony* on top of the *Media Broker* [13] infrastructure, and using Grid computing [8] for more flexible resource management. In addition, augmenting *Symphony* with historical stream persistence would allow

the infrastructure to support more media analysis applications.

9. REFERENCES

[1] Adhikari, S., Paul, A., and Ramachandran, U. D–Stampede: Distributed Programming System for Ubiquitous Computing. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS).* (July 2002), pp. 209–216.

[2] Amir, A., Srinivasan, S., Ponceleon, D., and Petkovic, D. CueVideo: Automated indexing of video for searching and browsing. In *Proceedings of SIGIR'99* (1999).

[3] Bacher, D. R. Content-based indexing of captioned video. Submitted to the Department of Electrical Engineering and Computer Science at MIT in partial fulfillment of the requirements for the degree of B.S. in Computer Science and Engineering, May 1994.

[4] Dannenberg, R. B. Aura as a platform for distributed sensing and control. In *Symposium on Sensing and Input for Media-Centric Systems (SIMS 02).* (2002), Santa Barbara: University of California Santa Barbara Center for Research in Electronic Art Technology, pp. 49–57.

[5] Dannenberg, R. B., and van de Lageweg, P. A system supporting flexible distributed real-time music processing. In *Proceedings of the 2001 International Computer Music Conference* (2001), San Francisco: International Computer Music Association, pp. 267–270.

[6] Eberman, B., Fidler, B., Iannucci, R., Joerg, C., Kontothanassis, L., Kovalcin, D., Moreno, P., Swain, M., and Thong, J. M. V. AltaVista media search: Indexing multimedia for delivery over the internet. In *Third International Conference on Visual Information Systems* (June 1999).

[7] Federal Communications Commission. Closed Captioning: FCC consumer facts, September 2003.

[8] Foster, I., Kesselman, C., and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of Supercomputer Applications 15*, 3 (2001).

[9] Hauptmann, A., and Witbrock, M. Informedia News on Demand: Multimedia information acquisition and retrieval. In *Intelligent Multimedia Information Retrieval*, M. T. Maybury, Ed. AAAI Press/MIT Press, 1996.

[10] Hauptmann, A. G., and Witbrock, M. J. Story segmentation and detection of commercials in broadcast news video. In *Advances in Digital Libraries* (1998), pp. 168–179.

[11] Intel. OpenCV: Open source computer vision library. http://www.intel.com/research/mrl/research/opencv/.

[12] Jain, R. Experiential computing. *Communications of the ACM 46*, 7 (2003), 48–55.

[13] Modahl, M., Bagrak, I., Wolenetz, M., Hutto, P., and Ramachandran, U. Media Broker: An Architecture for Pervasive Computing. In *Proceedings of the IEEE Conference on Pervasive Computing and Communications* (Orlando, FL, March 2004).

[14] Niblack, W., Barber, R., Equitz, W., Flickner, M., Glasman, E., Pektovic, D., Yanker, P., Faloutsos, C., and Taubin, G. The QBIC project: querying images by content using color, texture, and shape. In *Storage and Retrieval for Image and Video Databases, Proc. SPIE 1908* (1993), pp. 173–187.

[15] Pentland, A., Picard, R. W., and Sclaroff, S. Photobook: Tools for content-based manipulation of image databases. In *Proceedings of the SPIE Conference On Storage and Retrieval of Video and Image Databases* (February 1994), vol. 2185, pp. 34–47.

[16] Porter, M. F. An algorithm for suffix stripping. In *Program* (1980), vol. 14, pp. 130–137.

[17] Rasheed, Z., and Shah, M. Scene Detection In Hollywood Movies and TV shows. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.* (June 2003).

[18] Rhodes, B. J. *Just-In-Time Information Retrieval.* PhD thesis, MIT Media Laboratory, Cambridge, MA, May 2000.

[19] Robson, G. D. Closed captions, V-Chip, and other VBI data. *Nuts & Volts* (January 2000).

[20] Salton, G., Allan, J., and Buckley, C. Automatic structuring and retrieval of large text files. *Communications of the ACM 37*, 2 (1994), 97–108.

[21] Smith, J. R., and Chang, S.-F. VisualSEEk: A fully automated content-based image query system. In *ACM Multimedia* (1996), pp. 87–98.

[22] Snoek, C., and Worring, M. Multimodal video indexing: A review of the state-of-the-art. *Multimedia Tools and Applications* (2004). In press.

[23] Turk, M., and Pentland, A. Eigenfaces for recognition. *Journal of Cognitive Neuro Science 3* (1991), 71–86.

[24] Viola, P., and Jones, M. Rapid object detection using a boosted cascade of simple features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2001).

[25] Wactlar, H. D., Kanade, T., Smith, M. A., and Stevens, S. M. Intelligent access to digital video: Informedia project. *Computer 29*, 5 (1996), 46–52.