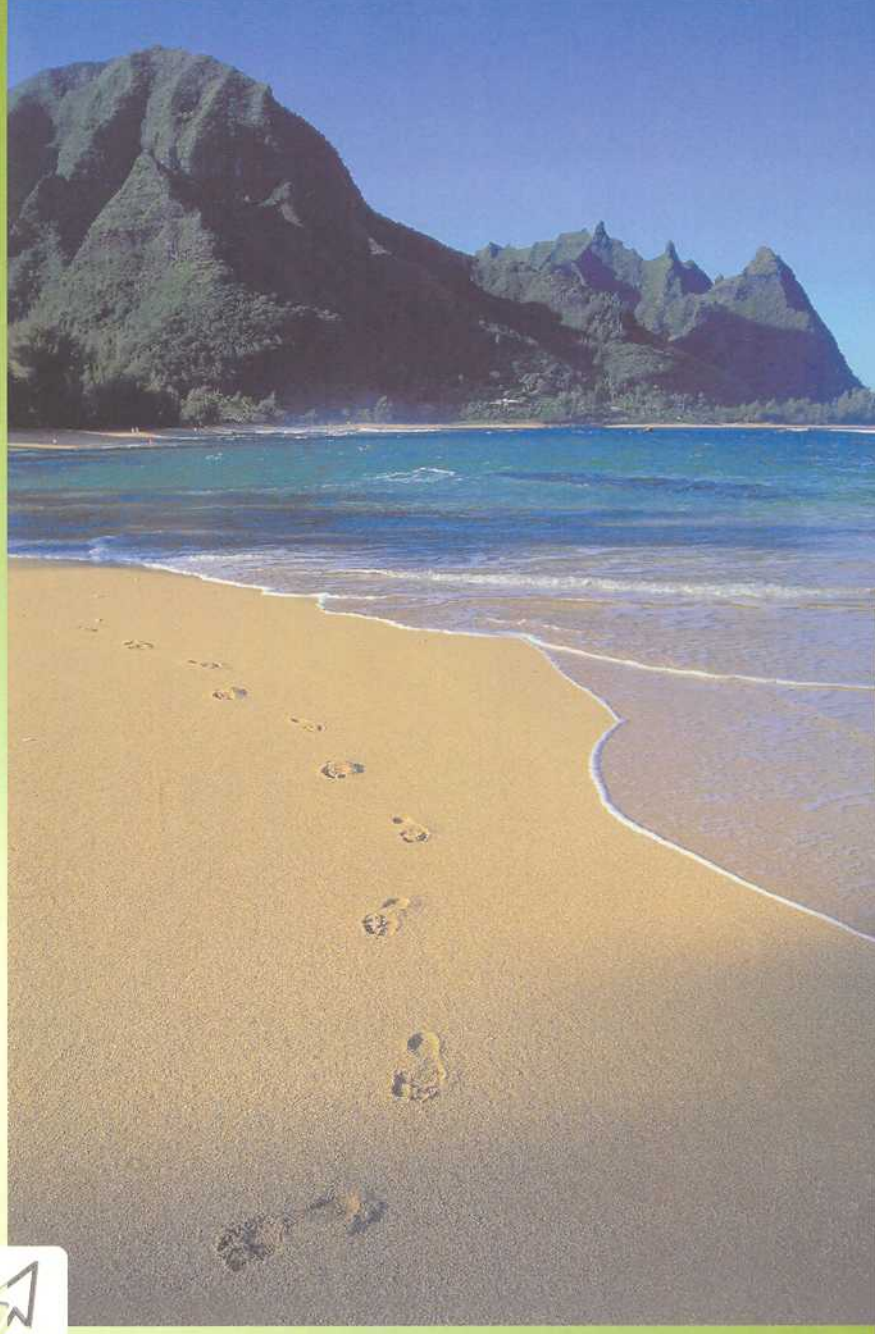
IASTED



Proceedings of the Eighth IASTED International Conference on

INTERNET AND MULTIMEDIA SYSTEMS AND APPLICATIONS

Editor: M.H. Hamza

August 16 – 18, 2004
Kauai, Hawaii, USA

A Publication of The International Association of Science and Technology for Development - IASTED

ISBN: 0-88986-420-9
ISSN: 1482-7905

ACTA Press

Anaheim • Calgary • Zurich

Proceedings of the Eighth IASTED International Conference
INTERNET & MULTIMEDIA SYSTEMS & APPLICATIONS
August 16-18, 2004, Kauai, Hawaii, USA

MULTIMODAL TRIGGERS FOR AUTOMATED FILTERING AND REACTIVITY BASED ON WWW CONTENT

Arvind K. Bansal and Sharif Uddin
Department of Computer Science
Kent State University, Kent, OH 44242, USA
Email: arvind@cs.kent.edu and suddin@cs.kent.edu
Phone: +1.330.672.9035, FAX: +1.330.672.7824

ABSTRACT
We describe a multimodal analysis based automated triggering mechanism for automated filtering and reactivity based on the content of multiple multimedia channels over the World Wide Web. The trigger description language supports complex conditions connected through Boolean operators (logical OR, logical AND, and negation), temporal and frequency of occurrence constraints and actions. A trigger can be placed either at the client end or at the server end. We present a grammar for an XML based trigger description language, and it's corresponding implementation. Performance analysis has been presented.

KEY WORDS
Internet, multimedia, multimodal, trigger, WWW, XML

1. Introduction

In the last couple of years, the World Wide Web has become pervasive. People have become dependent on it for their daily necessities such as news, stocks, movies, person to person communication, and business conferencing. There are huge numbers of channels that continuously broadcast music, news, and movies over the Internet. Due to the increase in multimedia information over the WWW, we are facing information overload. This requires that we selectively and automatically filter out unwanted content or automatically take an action after a complex multimedia event occurs. For example, parents can exercise parental control using a preference scheme that prevents children from hearing or seeing certain audio/visual content using automated monitoring of content in real time. People can integrate work and entertainment: a channel gets automatically switched to a sports channel for a limited duration when an important event occurs. A patient's heart monitor could trigger, record, or send wireless e-mail when an unusual activity (as recognized by image or sound patterns) occurs.

There is a need for automated multimodal triggers based on content based analysis to monitor channels and to take complex reactive actions based upon user based preferences and multimedia stream characteristics [1].

The overall scheme for client end event based triggers is described in Figure 1. A background channel listener monitors multiple audio and video channels continuously. Audio and image content of the channels are continuously analyzed using an audio analyzer, a video analyzer, and an image analyzer based upon user preferences. The *trigger manager* maintains a user described complex event. An event is a temporally related or a constrained combination of audio patterns and image patterns. After an event occurs, the trigger manager triggers a corresponding action such as play, pause, delete, record, or the starting a function. This system exploits the integration of speech to text conversion and image matching to identify specific patterns of user preferences from a library or any media content, and displays a specific channel for an user preferred duration.

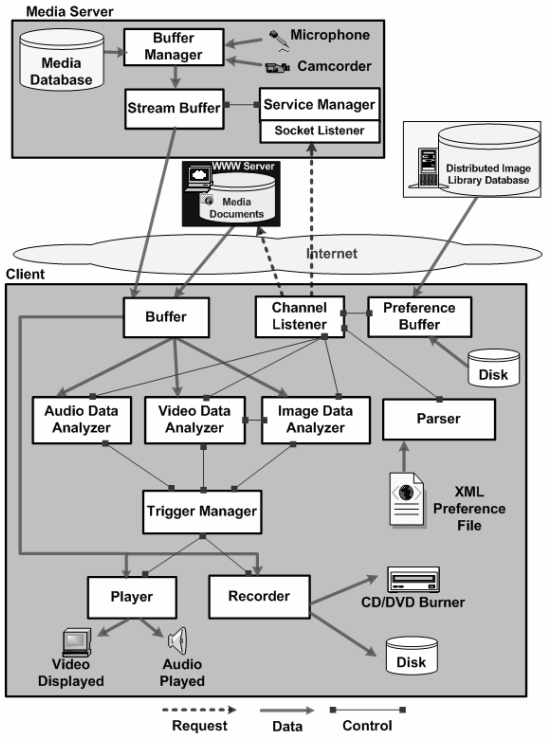


Figure 1. The overall scheme

Our system implementation benefits from previous research on multimodal content based analysis in the

context of content based image indexing and retrieval [2], shape analysis and object recognition [3], multimedia interaction and multimodal interfaces [4], speech to text conversion [5, 6], speech technology [7, 8, 9], and digital image processing [10].

The major contributions of this paper are as follows:

1. XML has been extended for content based triggering based on multimodal analysis of audio and video streams over the World Wide Web. The use of XML provides extensibility and flexibility.
2. The trigger supports logical operators, frequency of word occurrences and temporal constraints to define an event.
3. The system monitors and performs multimodal analysis on multiple channels simultaneously.

The system and the corresponding user-interface has been implemented to run on Microsoft Windows using C#, C++, Intel's OpenCV [11] (for image analysis) and Microsoft speech recognition SDK [7].

The paper is organized as follows: Section 2 describes the background needed for speech to text conversion, text matching, and image matching. Section 3 describes a grammar for the trigger description language. Section 4 describes the architecture and implementation of the system. Section 5 describes the performance evaluation. Section 6 talks about related work. Section 7 concludes the work.

2. Background

Speech recognition [5] analyzes a spoken audio signal to determine words. During this process, the audio signal is analyzed to identify *phonemes* – smallest unit of sound – and silence. Each phoneme has a spectral shape [5]. A *signal analyzer* converts an audio stream to spectral shapes, and these spectral shapes are matched to identify phonemes. Silence is used as a word boundary. Phonemes together with word boundaries are analyzed to identify spoken words, perform segmentation, labeling, and higher level processing.

There are many commercial and free speech recognition systems [7, 12, 13] capable of huge vocabulary and continuous speech recognition. We have used Microsoft's Speech SDK [7] in this work.

Content-based image retrieval [14, 15] uses features such as color histograms [10], color moments [14], pattern matching, as well as shape and contour matching [16].

Color histograms rely on the overall distribution of colors in an image. Increasing the number of bins in a color histogram enhances the discrimination power. A common problem with color histograms is that different images can have similar color distributions. Color moments [14] rely on the mean, variance, and skew of a color distribution. Image segmentation divides an image into sub-areas and calculates a histogram for each sub-area to reduce the ambiguity caused by multiple images mapping to one histogram distribution.

One technique used in the OpenCV library [11] is to compare two histograms p and q. A correlation function is used given by the following equation, where p_i is the i^{th} component of histogram p, q_i is the i^{th} component of histogram q and N is the number of histogram bins. The result is 1 for a 100% match.

$$\frac{\sum_i p_i q_i - \frac{\sum_i p_i \times \sum_i q_i}{N}}{\sqrt{\left(\sum_i p_i^2 - \frac{\sum_i p_i \times \sum_i q_i}{N}\right) \times \left(\sum_i q_i^2 - \frac{\sum_i p_i \times \sum_i q_i}{N}\right)}} \quad (1)$$

Shape matching involves edge detection [17, 18], contour detection, and region based matching. One or more contours represent a shape. The edge detection algorithms involve smoothening of images using a Gaussian filter [10], partially differentiating the images on the x and y axes to get the gradient, non-maximum suppression to compute individual edge points, and the use of a threshold to remove undesired edges. An edge point is maximal locally in the direction of the gradient. A threshold based upon color or luminosity is used to filter regions of interest. Figure 2 shows an original image to the left, the contours extracted in the middle, and the grayscale histogram to the right.

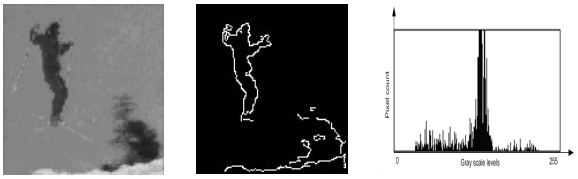


Figure 2. Image analysis technique

3. A Simplified Grammar

In this section, we describe a simplified grammar for the multimedia trigger description. This grammar describes four components: channel description for monitoring, trigger and event description, multimedia item description, and actions. See Figure 3.

A *preference* is a set of monitored *channels*. A channel is abstracted as a quadruple of the form (*channel-id, channel-name, location, status*). The attribute *location* is an *address* followed by a *port* or an *uri*. An *address* represents a domain name or an IP address. An *uri* represents a WWW document or media file. A channel can have a *status* "*active*" or "*inactive*". Only *active* channels are monitored. A user can deactivate or delete a channel from a preference list.

A *trigger* is a triple of the form (*trigger-id, event-description, action*). A trigger is activated after an event takes place. An event includes the positive multimedia matches or the absence of positive matches during a specified duration. An action could be a computation, rendering (*play*), suppression of rendering (*delete*),

temporal suspension of rendering (*pause*), archiving (*record*), or deleting the channel (*shutoff*).

```
/* Channel and preferences */
<preference>::= '<pref>' {<channel>}+  '</pref>'
<channel> ::=  '<chan' <chan_attr> '>' {< trig>}* '</chan>'
<chan_attr> ::= <chan-id> [<chan-name>] <loc> [<status>]
<loc>::= <address> [<port>] | <uri>
<address> ::= <domain-name> | <ip-address>
<status>::= "active" | "inactive"

/* Trigger and  Event Description */
<trig> ::= '<trigger' <trigger-id> '>'  {<event>}*  <action>
           '</trigger>'
<event> ::= '<event' <event-id> '>' <matches> '</event>'
<matches> ::= '<matches>' <match> |<match> <logical-op>
                 <matches> '</matches>'
 <match> ::= '<match>' <match-type> [<percent-match>]
               [<threshold>] [<occurrence>] [duration-frequency]
               '</match>'
<duration-frequency> ::= '('<show> <duration>')'
<match-type> ::= <text> | <audio> |   <image> |<video>
<text> ::= '<text>' <text-item> '</text>'
<audio> ::= '<audio>' <audio-item> [chan-type] '</audio>'
<video> ::= '<video>' <video-item> '</video>'
<chan-type> ::= "mono" | "dual" | "stereo" | "joint"
<image> ::= '<image>' <image-item> '</image>'
<percent-match> ::= '<percent>' <integer> '</percent>'
<threshold> ::= '<threshold>' <integer> '</threshold>'

/* Multimedia item description */
<text-item> ::= <phrase> | <word>
<phrase> ::= '<phrase>' {<word>}+  '</phrase>'
<word> ::= '<word>' {char}+  '</word>'

<audio-item> ::= {<audio-feature>}+ | <text>
<audio-feature> ::= '<audio-feature' '>' [<frequency> ] |
                      {<pitch>] |  [<sample-rate>]
                      '</audio-feature>'
<frequency> ::= '<freq>' <range> | <integer> '</freq>'
<occurrence> ::= '<occur>' <range> | <integer> '</occur>'
<sample-rate> ::= '<rate>' <range> | <integer> '</rate>'

<video-item> ::= {<image-item>}+
<image-item> ::= '<image' <loc-type> '>' <image-desc>
                 '</image>'
<image-desc> ::= '<image-feature>' <image-type> <file-path>
                  '</image-feature>'
<image-type> ::= "shape" | "face" | "color" | "similarto"
<loc-type> ::= <filename> | <database-imageid>
<database-imageid> ::=  '<database-imageid'
                          <database-location> '>' <integer>
                          '</database-imageid>'

/* Comparison, duration and visibility description */
<range> ::= <rangeop> <integer> '-' <integer> |
            <comparator>  <integer>
<show> ::= "inside" | "outside"
<comparator> ::=  == | > | < | = | =
<duration>::= '<duration>' forever | never | <integer> | <range>
              '</duration>'

/* Action description */
<action> ::= "play" <qual> | "record" <path> <qual> |
             "replay" <qual> | "suspend" <qual> |  "shutoff"
<qual> ::= "for" <duration> | "until" <time> |
           "for" <duration> "from" <time> |
```

Figure 3. A simplified grammar

A complex event is a composition of simple events. A simple event could be the occurrence of a media item during a time range for a specific number of times or the absence of a specific media objects for a certain duration. Simple events are connected through logical operators such as logical-AND, logical-OR, negation, temporal constraints and frequency (or a range of frequencies) of occurrence during a duration. The temporal constraints used are the presence or absence of media for a specified duration (possibly indefinite). The time could be either absolute or relative to a previous matching event.

A *multimedia match* has optional attributes *match-percent* and *duration-frequency*. The *match-percent* is a threshold supporting approximate matching to compensate for the presence of noise and the incompleteness of recognition of the objects caused due to the absence of all multimedia modes. The attribute *duration-frequency* is a pair of the form (*show*, *duration*). The *show* field determines if the media will be *present* or *absent* for a particular duration. A *duration* is an exact count or inequality (less than, greater than, less than or equal to, greater than or equal to) with respect to a given value. Different types of multimedia matches have additional requirements. For example, in the audio-match, *chan-type* could be *mono*, *dual*, *stereo* or *joint*.

A *text-item* is a *word* or a *phrase*. A *phrase* has one or more words in it. An *audio-item* is a bag of *audio-features* or *text-match*. An *audio-feature* could be *frequency*, *pitch* or *sample rate* of audio. An *audio-match* is either a *digital waveform match* or a *text-match* form. In digital waveform match, digitized sound is matched. In text-match form, incoming digital sound is transformed to phonemes, phonemes are transformed to text form, and the resulting incoming text is matched with the given text.

The next type of item is a *video-item*, which is a sequence of one or more *image-items*. Since image matching is approximate due to the incompleteness of segmentation algorithms and due to the change in image due to the lighting and shadow effect, a *threshold* is needed. A video match occurs above this threshold.

Finally, an *image-item* is a bag of *image features*. An *image-feature* could be "*shape*", a "*face*", a "*color*", or "*similarto*", which matches two similar images. An *image-feature* has either a *filename* or a *database-imageid*, describing the location of the image features.

4. Architecture and Implementation

In this section we describe an abstract model for the client end architecture for monitoring and filtering multimedia streams. We have intentionally omitted the server side description as we use regular Internet sites.

The first step is to parse the XML preference file and spawn a *channel listener* process thread for each active channel preference. A *trigger manager* process listens to the audio, video or image data analyzer processes for a preferred event (refer to Figure 1). Once a match is found, the *trigger manager* triggers an action to play or record the media for a duration based on the preference. Figure 4 gives the abstract process for monitoring channels.

```
Algorithm: monitor-channels
Input: XML preference file;
for each preference channel   {
    check location of channel;
    if (location is WWW document) {
      download the document;
      parse the downloaded document to find media;
      buffer manager fills the stream buffer;}
    if (location is media server) {
      buffer manager gets the media data from media
          server and fills the stream buffer;}
    parse preferences for the channel;
    if (preference is audio) {call audio data analyzer; }
    if (preference is video) {call video data analyzer; }
    if (preference is image) {call image data  analyzer; }
}
```

Figure 4. Monitoring channels

4.1 Audio Analysis

The audio data analyzer processes media streams in three steps. First, media streams are converted to text. Existing Automated Speech Recognition (ASR), has been extended to handle temporal constraints and frequency occurrence of words. Second, word boundaries, word offsets and time offsets are stored in a history table. Finally, the incoming text stream is matched using approximate string matching with the information in the associative preference table. An abstract description is given in Figure 5.

A history table maintains information about incoming words in an audio stream. Each word is associated with a *stream offset*, an *absolute time offset*, a *time offset from the last occurrence of the same word*, and *total occurrence of the word* in a *rolling time window*. Some commonly used words such as 'a' and 'the' are pruned. The system maintains an excluded word list in a hash table to prune such words.

The current and previous matches are maintained to test for various operators such as *and*, *or* and *time duration*. A *current-pointer* is used to keep the information about the last matched item in a preference list. The *current-pointer* is incremented only after a match is found. A trigger is activated when the *current-pointer* reaches the end of the preference list. A match state is also maintained. Previous match is reset if the match does not occur within the constrained time limit.

```
Abstract Process: analyze-audio-data
Input: audio preference ;
while not end of stream {
    if  (stream data present in stream buffer)   {
            read data from stream buffer;
            convert stream data to text;
            parse audio data preference;
            if  (match found) {
              call trigger manager to play or record the
              audio from stream buffer;}
    } else wait for buffer manager to fill the stream buffer;
}
```

Figure 5. Audio analysis

Example 1: An XML snippet for an audio matching preference is shown in Figure 6. The word "hungry" has an "included" *qualifier*, which states that the match will occur and the result will be played since the *trigger* has an *action* of "play". The next attribute of the word "hungry" is *occurs*, which states that the word "hungry" occurs at least once and at most twice in the audio stream. The time distance between two "hungry" words should be between *0* and *20* seconds, which is denoted by the *min-max-time* attribute. The next word following "*hungry*" is "*people*", and is connected by "*and*" operator.

```
<channel channel-id="1" channel-name="chn1"
      location="www.cs.kent.edu" port="7001" status="active">
    <trigger trigger-id="1" action="play"
          action-qualifier="forever"> …
          <text-match>…
                <word qualifier="included" occurs="1-2"
                      min-max-time="0-20">hungry</word>
                <op>and</op>
                <word>people</word>
          </text-match>
    </trigger>
</channel>
```

Figure 6. An audio preference example snippet

4.2 Video and Image Analysis

The video analyzer reads the video stream buffer. A frame grabber [19] is used to grab frames from video data. A *grab-buffer* event creates a Device Independent Bitmap (DIB) image from the media buffer and calls *process-image* in a thread in order to match the grabbed image with the preference expression. Figure 7 has the abstract description.

```
Abstract process: analyze-video-data
Input: video preference ;
while not end of stream {
    if (stream data present in stream buffer)   {
            read data from stream buffer;
            extract frames from stream buffer;
            parse video data preference;
            for each frame {call 'analyze-image-data' ;}
    } else wait for buffer manager to fill the stream buffer;
}

Abstract process: analyze-image-data
Input: image preference;
while not end of stream {
    if  (stream data present in stream buffer)   {
            read data from stream buffer of an image;
            extract image features from the image;
            parse image data preference;
            if (match found) {
               call trigger manager to play or record the
               video from stream buffer for preferred; }
            else
               wait for buffer manager to fill the stream buffer;
    }
```

Figure 7. Video analysis

We have used color histograms, and shape matching using functions given in Intel's OpenCV [11] to match images. Finding contours requires a *chain code* algorithm

[20] on the edges detected by the *Canny* algorithm [18]. In order to find the chain encoding of an enclosed contour, a grid is first superimposed on the contour. The contour is then sampled based on the grid and an ordered list of points on and near the boundary is represented using directional codes (see Figure 8).

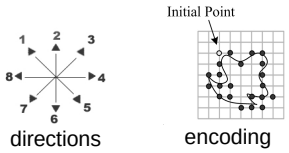


Figure 8. Contour making

OpenCV's contour matching is based upon the equation shown below.

$$\text{result} = \sum_{i=1}^{7} \frac{| m_i^A - m_i^B |}{| m_i^A |} \quad (2)$$

$$\text{where } m_i^A = sign\ (h_i^A) \log_{10} | h_i^A |$$

$$m_i^B = sign\ (h_i^B) \log_{10} | h_i^B |$$

The terms h_i^A and h_i^B are *Hu Moments* [21] for the contours *A* and *B*. Hu has shown that a set of the seven features derived from the second and third moments of contours is invariant to translation, rotation, and scaling.

Example 2: Figure 9 shows a tested example of shape matching using contours. The user provides the name of the image file. The image is analyzed and the contour (see Figure 9b.) is stored in the database. Figure 9c shows one frame of the image from the video. The gray color version of the frame is analyzed to derive the edges. Figure 9d shows the image after passing it through the edge detector, and Figure 9e shows the corresponding contour. The region of interest is extracted using neighboring pixel contrast, and the contours are matched using an OpenCV [11] function.

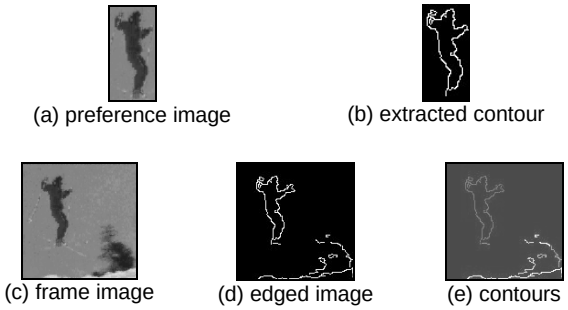


Figure 9. An example of shape matching

5. Performance Evaluation

We investigated the overall delay and change in buffer size during rendering as the number and complexity of multimedia objects being analyzed were increased. This analysis is important to assess real time rendering capability in devices with limited buffer sizes such as PDAs and medical devices.

The result is summarized in Figures 10 through 13. Figure 10 shows the time delay performance for a phrase. Words in a phrase are connected through 'logical-AND'. However multiple simple events may be connected through 'logical-OR' as well as 'logical-AND'. In case of logical-OR, the presence of any event will be sufficient. If two events have been declared to be '*t*' seconds apart, then the total time taken would be (time for event 1 + *t* + time for event 2).

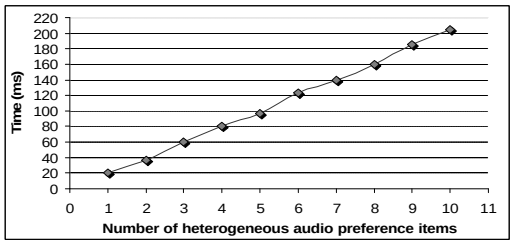


Figure 10. Number of audio items vs. time delay

Figure 11 shows the time delay for the same phrase with different sampling rate. Since the information for each sample has the same phrase, there is a near linear increase in time as the sampling rate increased for the same phrase.

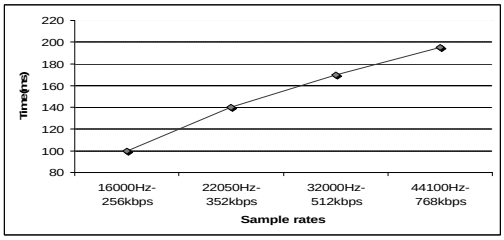


Figure 11. Time delay vs. sampling rate

Figure 12 shows time delay vs. number of homogeneous images in the same frame, and time delay vs. enhancing the resolution. There is an initial overhead of deriving histograms and obtaining gray color images for shape matching. The graph shows that delay increases with the number and resolution of images.

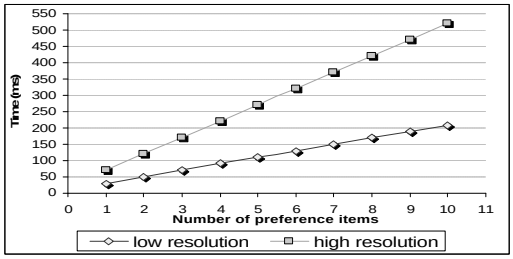


Figure 12. Time delay vs. image items

Figure 13 shows a graph for buffer size vs. total number of characters in a preference. Different curves represent different sampling rates. The graph shows that the buffer size increases almost linearly as the number of homogeneous image items are increased. The buffer size also increases as the sampling rate for an audio item increases. Slight nonlinearity in the graph is due to the

non-uniform spectra for different characters during speech to text conversion.

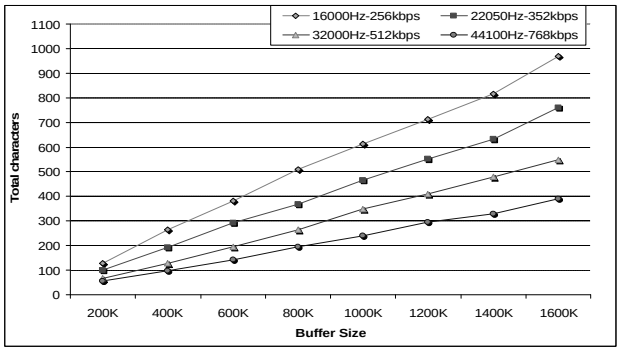


Figure 13. Buffer size vs. number of characters

6. Related Work

There has been some work on the use of XML to index and retrieve the multimodal media extracted using content based analysis [22]. Their focus is on content based analysis and detecting scene changes. In contrast to our research, their implicit trigger lacks the notion of temporal constraints and logical connectivity and handling the absence of media objects during a time range in a trigger.

7. Conclusion and Future Work

We have described an XML based system and the corresponding implementation for automated multimedia content analysis based triggering. The system continuously monitors user preferred channels over the World Wide Web, and takes an action when a user defined event occurs. A multimedia event is a composition of the presence (or absence) of multiple images, text, or spoken words and phrases during a time frame. The action could be visualization, computation, or communication of vital information.

The current implementation cannot discriminate multiple voices. The image analysis cannot handle partial occlusion of images, and the modeling of complex events needs to be studied further. The threshold to prune undesired edges (during image matching) needs domain specific intelligent processing. We are currently extending our system to include separation of voices in spoken words.

References

[1] A. Guercio and A. K. Bansal, A Model for Integrating Deterministic and Asynchronous Events in Reactive Multimedia Internet Based Languages, *Proc. 5th Intl. Conf. on Internet Computing,* Las Vegas, June 2004, to appear

[2] R. Gonzalez and A. Wardhani, Resource Discovery and Content Based Retrieval of Visual Data, *Proc. Australian Telecommunication Networks & Applications Conf., ATNAC '96*, Melbourne, Australia, 1996, 369-374.

[3] M. H. Safar and C. Shahabi, *Shape analysis and retrieval of multimedia objects* (Boston: Kluwer Academic Publishers, 2003)

[4] S. Oviatt, *Multimodal interfaces, the human-computer interaction handbook* (Mahwah, NJ: Lawrence Erlbaum Associates Publishers 2003), 286-304.

[5] J. Bernstein and H. Franco, *Speech recognition by computer, principles of experimental phonetics*, (St. Louis, MO: Mosby, 1996), 408-434

[6] P. P-Nga, S. R. Subramanya, N. A. Alexandridis, S. Srakaew and G. Blankenship, Content-Based Audio Retrieval Using a Generalized Algorithm, *Proc. Advances in Intelligent Systems: Concepts, Tools, and Applications*, Kluwer Academic, 1998.

[7] Microsoft Speech and Speech SDK 5.1, Available online: *http://www.microsoft.com/speech/*

[8] S. Srinivasan and E. Brown, Is Speech Recognition Becoming Mainstream?, *IEEE Computer*, April, 2002, 38-41

[9] A. Syrdal, R. Bennett and S. Greenspan (editors), *Applied speech technology* (Boca Raton: CRC Press, 1995), Chapters 4,5,6 and 9.

[10] J. C. Russ, *The image processing handbook* (Boca Raton: CRC Press, 2002)

[11] Open Source Computer Vision Library, Available online: *http://www.intel.com/research/mrl/research/opencv/*

[12] Dragon NaturallySpeaking, Available online: *http://www.scansoft.com/naturallyspeaking/*

[13] Nuance, Available online: *http://www.nuance.com*

[14] D. Long, H. Zhang and D. D. Feng, Fundamentals of Content-based Image retrieval, *Multimedia Information Retrieval and Management - Technological Fundamentals and Applications*, Berlin, NY, Springer, 2003, 1-26.

[15] A. Smeulders, M. Worring, S. Santini, A. Gupta and R. Jain, Content-Based Image Retrieval at the End of the Early Years, *IEEE Trans. on Pattern Analysis and Machine Intelligence, PAMI*, Washington, DC, December, 2000, 22(12), 1349-1380.

[16] S. Belongie, J. Malik, and J. Puzicha, Shape Matching and Object Recognition Using Shape Contexts, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Washington, DC, 2002, 509-522.

[17] D. Marr and E. Hildreth, Theory of edge detection, *Proc. Royal Society of London*, 207, Series B, 1980, 187-217.

[18] J. F. Canny, A Computational Approach to Edge Detection, *IEEE Trans. on Pattern Analysis & Machine Intelligence*, 8(6), Washington, DC, November 1986, 679-698.

[19] Microsoft DirectX SDK, Available online: *http://www.microsoft.com/directx*

[20] H. Freeman, L. S. Davis, A corner-finding algorithm for Chain-coded curves, *IEEE Trans. Computer*, 26(3), 1977, 297-303.

[21] M. Hu, Visual Pattern Recognition by Moment Invariants, *IRE Trans. on Information Theory*, 1962, 8(2), 179-187.

[22] C. H. Ngai, P. H. Chan, E. Yau, M. R. Lyu, XVIP – An XML Based Video Information Processing System, *Proc. 26th Annual Intl. Computer Software and Applications Conf. (COMPSAC2002)*, Oxford, England, August 26-29 2002, 173-178.