A System for Information Retrieval Applications on Broadcast News Videos

Tarun Jain Sai Ram Kunala Ravi Kishore Kandala C.V. Jawahar
Center for Visual Information Technology,
International Institute of Information Technology
Hyderabad, India

***Abstract*—We present a system that is specifically designed for the information retrieval applications on broadcast news videos. The system is directly useful to an end user for easy access to the news stories of interest. It also act as a platform for convenient deployment and experimentation of various video analysis and indexing techniques on real data, and on a large scale. The system is built upon four layer architecture with certain software design choices that makes the system highly scalable, extensible and modular. The system has been in use for 20 months and has processed around 70Tb of broadcast news data till date. Extensive performance analysis of the system was done by deploying 14 state of the art desktop systems. Results of the same are reported here. This system holds immense potential for the emerging video information retrieval applications.**

I. Introduction

The increasing number of news channels results in an abundance of video data which makes it difficult for the users to go through all the available information. In India, a typical cable or satellite TV distribution network daily delivers approximately 200 hours of news in English, Hindi and regional languages. One can never afford to view this entire news broadcast. This motivated us to explore a systematic solution for information retrieval from news videos which can enable the users to access only the relevant information of their interest.

The last decade has witnessed a spurt in the production of digital video content fueled primarily by the ubiquity of digital video cameras and camera phones. Further, with the advent of personal video recorders and media center PCs, the creation of large video repositories is getting into the mainstream [1]. Technologies like IPTV as well as the commoditization of storage hardware, catalyzes this emergence. The sheer enormity of the video data along with the dearth of effective solutions to provide content based access has taken the demand for video information retrieval at an all time high. In this paper, we describe the design and implementation of a broadcast news video indexing system which is built to host an array of emerging video information retrieval techniques.

Our system is based on a model similar to that of web search engines like Google. These search engines index the content and users are pointed to the source for browsing the original content. Figure 1 depicts the system at a coarse level. The content is captured, processed, analyzed and efficient indices are then constructed from the extracted metadata. Various IR techniques then make use of this information for providing content based access to broadcast news. Details of the video processing are beyond the scope of this paper.

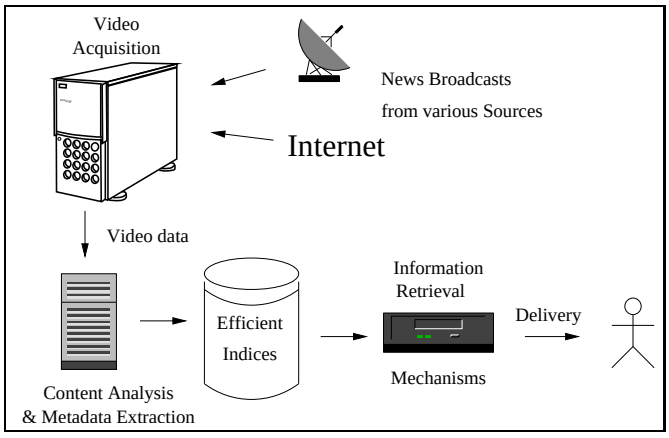

Fig. 1. The system at a coarse level

The general solution for multimedia search has traditionally been to build some feature descriptions of the content and then matching these representations [2]–[5]. A popular way to describe video content is by using text annotations. These are reasonably effective for the task of keyword based search. Annotation is usually a manually intensive and a cumbersome process. Content could also be represented using key frames [2], [5]. This reduces the problem of video retrieval into that of CBIR.

Video retrieval and analysis techniques have come a long way in recent years [6]–[8]. There have been significant advances in video indexing mechanisms which use various information cues like faces [9], location [10], audio [7], activities, etc. Lack of a specialized platform for the deployment and experimentation became a roadblock in the development of robust and scalable algorithms. Our system works as an end to end framework for the researchers wherein they can plug in their specific algorithms and get to test them with real data and other supporting mechanisms. From an end user's perspective, the system offers an intuitive mode of browsing through the news stories of their interest in a limited amount of time. They need not sequentially play all the videos to find the useful content which otherwise is the only option in absence of any such system.

The biggest challenge in designing and building such a

system is to be able to not only take into consideration the present requirements but also to predict the requirements that can come up in future while developing techniques which might not even be imagined at the moment. This is the primary factor that has motivated the architecture and design of the system. Modularization has been given paramount importance as it enables independent development of different parts of the system. The system further needs to be scalable and reliable which has been taken care of through various design decisions.

This system is in existence for more than 20 months and has consistently been evolving to come to its current form. Almost 50 hours of broadcast news video data in 3 different languages (English, Hindi and Telugu) gets recorded and processed each day by the system. Almost 70Tb of videos have been processed in all by the system till date. There are 1000 valid users which have password protected access to the processed content over our university's LAN. Performance studies conducted on a cluster composing of 14 state of the art personal computers is reported in Section IV. The results clearly depict the success of the system in being robust and reliable against common causes of failure.

II. Architecture

The architecture of the system is primarily influenced from our experiences with a preliminary version of the system. The video data being bulky in nature, poses a number of challenges for its capture, flow over the network and of course processing. To get an idea of the scale, typically 100 GB of data gets captured and processed by the system on any given day. Moreover, video processing is computationally very intensive. This is due to data size as well as the complex nature of the typical video processing algorithms. Processing often needs to be parallelized and it needs to be ensured that it always keeps up with the data acquisition rates. The system continuously records and stores the broadcast videos. It needs to be ensured that failure of the storage machines does not propagate to the recording machines. Similarly, any crashes in the processing machines should not have any negative effect on the recording or storage tasks. The system must be capable of being extended frequently even though the algorithms and the skills of the people interacting are quite diverse. These issues result in the following set of goals for the system.

A. Goals and Challenges

1) *Modularity:* A clear demarcation of responsibility among the different modules with minimal coupling and standard interfaces between them was the primary requirement for the system. As different people work on different parts of the system, modularization is of utmost importance for the smooth evolution of the system. Moreover, there must be scope for modifying the implementation of a module without affecting other modules.
2) *Scalability:* The entire framework is targeted to keep growing in scale over time. Over time, the system needs to scale up in terms of the amount of video data being stored and processed, number of parallel streams being captured, number of IR applications deployed, number of users, etc. The entire infrastructure should be able to scale up incrementally. Also, the individual sub systems should be capable of being scaled independent of other parts. For example, the processing capabilities of the system could be increased significantly without changing anything with the other parts.
3) *Extensibility:* The system requires frequent addition and alteration of software modules. The newly added modules must work seamlessly with the already existing modules performing other tasks. This enables frequent upgradation of the software. The process of substituting alternate modules in place of existing ones to perform the same task should be easy and repeatable. This allows developers of the modules to test various alternates and select the best possible combination of modules.
4) *Reliability:* The various components of the system run round the clock and interact with each other. Multiple machines interacting over network are part of the overall process. It must be ensured that even if one or more components fail, the rest of the system should continue to function.

A *Four-Layered* architecture is chosen to achieve high modularity and independence in design. The overall function of the system is divided into 4 layers of independent modules which interact via pre-decided interfaces. Figure 2 shows the different layers with their constituents and the interactions between them. To ensure *Loose Coupling* in the system, it is ensured that each layer has well defined and independent responsibilities. The organization of modules into separate layers also ensures that any of them can be scaled independent of the other layers.

Throughout the system, we have chosen *Horizontal Scaling* over *Vertical Scaling*. This means that for the scaling up of any of the Video Acquisition, Data Management or Processing layers, we chose to add more servers to the layer rather than increasing the capacity of the existing servers. It increases the reliability and availability of the system as dependence on single server decreases and failure at a single point is less likely to lead to failure of the system as a whole. It also allows increase in performance by introducing *Load Balancing*. The *Peer-to-peer* nature of the system also gets a boost because of horizontal scaling. The recording and processing nodes can establish individual connections with the relevant storage nodes for data transfer. This distributes the load over the system and allows higher reliability and availability of the overall system. The users accessing processed content over web also get automatically redirected to videos residing at different locations which prevents clogging of the central web server.

B. Video Acquisition

This layer includes all the elements of the system that are employed for capturing the broadcast video signal and encoding. The videos thus recorded are cached locally and

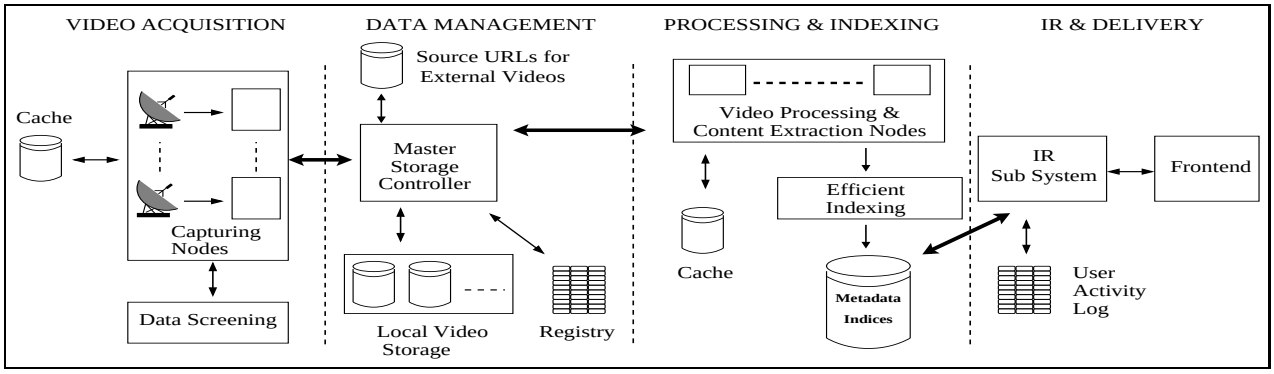

Fig. 2. A four-layered architecture is chosen to achieve high modularity and independence of design. The overall function of the system is divided into 4 layers of independent modules which interact via pre-decided interfaces.

then transferred to the central storage in the data management layer from where they are accessed for processing.

An interface needs to be provided between the Video Acquisition layer and the Data Management layer for transfer of chached videos. We evaluated and experimented with several options to establish a reliable method for this data transfer. Writing over shared storages suffers from low reliability as problems like one of the machines going down can propagate to other parts of the system and cause unrecoverable errors. Also, dynamically providing direct access of storage locations to recording systems over network file systems is cumbersome. This motivated us to put a *Caching* mechanism in place. The video when first captured is cached and then transferred to the storage in the Data Management layer. There is a *Data Screening* component in this layer. It acts as a filter to discard low quality videos.

C. Data Management

The task of this layer is to manage all the video data, thus becoming a central point for any data access. It consists of a local storage for maintaining the unprocessed videos. Typically, the storage is distributed over multiple storage nodes in order to achieve high capacity of storage incrementally. As the storage location is not static, any data access request to the Data Management layer has to be controlled centrally. Moreover, space management must also be done centrally. These requirements give rise to the *Master Storage Controller*. The master controller interfaces with the other layers through a web service. A *registry* of all the videos in the storage is also maintained in this layer. Each entry of the registry contains basic information like record time, source, etc. required to uniquely identify the video along with its storage location. The timely deletion of old videos and their corresponding entries is also handled by the controller.

When a write request is made to the master controller from a capturing node in the Video Acquisition layer, it selects a storage location with enough free space and minimal load. A peer to peer connection is established for the data transfer. Once the file transfer is done, the master controller is notified. The master controller then adds this as an entry into the registry. In case, the storage location gets down due to any reason anytime before the completion of the data transfer, the capturing node requests for a new location for storing the video and starts the transfer again. A log of all such incomplete transfers is maintained by the master storage controller. These incomplete files are cleaned up by the master controller at regular intervals. Similarly, for a read request from the processing layer, master controller looks up the registry based on identifying information and finds out the storage location for the requested broadcast. The download URL is then returned back as response. This layer also contains a database of the access URLs of the processed videos.

D. Processing and Indexing

The task of this layer is to perform the entire processing and indexing operations. There are 3 major components of this layer.

1) *Video Processing:* The standard pre-processing and video processing operations are performed by this component. These include noise removal, shot detection and advertisement removal.
2) *Content Extraction:* Various feature extraction and content analysis algorithms form a part of this component. These include key frames, color histograms, motion vectors and face videos.
3) *Indexing:* Efficient indexing schemes are employed to build index structures of all the metadata extracted. There can be multiple indices for different metadata information. These indices are used by the IR & Delivery layer for providing quick content based access to the users looking for some specific information.

The system supports various types of metadata for the videos:

1) Source information about the videos like channel, broadcast date and time, resolution, frame rate and language.
2) Low level features like color, motion, compressed domain features and audio stream information.
3) Object level annotation information like locations, faces, activities and events.
4) Direct and indirect user feedback for the videos.

There are multiple processing tasks running simultaneously in the processing. Each processing task maintains an independent schedule of videos to be processed according to which

it accesses the raw data from the data management layer. The processing layer interacts with the IR and Delivery layer for providing access to the metadata index. There is support for indexing the metadata into an XML database. This ensures flexibility to the processing modules for modular processing of videos and the metadata can be used partially. Moreover, this XML based structuring enables sharing of the metadata across different processing tasks in a standardized manner. The access to this XML metadata and output content of pre-processing such as keyframes, is also enabled through network APIs. This ensures higher reliability and decoupled storage of the metadata. This also ensures reliable sharing of metadata across various processing tasks.

The XML schema for the metadata consists of primarily two parts. First is the basic information about the video that is captured at the time of recording such as source name, language, recording timestamp, resolution and duration. Second part contains support for storing the metadata extracted by various feature extraction and processing modules. Data types of some of the elements are custom defined to impose certain restrictions on values they can assume. For example, minimum and maximum possible values, enumerated types, etc. Following is an example fragment from the schema.

```
<xs:element name="shotsInfo">
  <!--List of Shots-->
  <xs:list>
    <xs:complexType>
    <xs:sequence>
      <xs:element name="startFrame"
      type="xs:integer"/>
      <xs:element name="endFrame"
      type="xs:integer"/>
      <xs:element name="keyframe"
      type="xs:integer"/>
    </xs:sequence>
    </xs:complexType>
    <xs:minLength value="1"/>
  </xs:list>
</xs:element>

<xs:element name="featureVector">
  <xs:complexType>
  <xs:sequence>

  <xs:element name="featureName"
  type="xs:string"/>
  <!--Feature extracted over this time
  interval-->
  <xs:element name="timeInterval"
  type="timeIntervalType"/>
  <!--Low Level Feature Vectors like Color
  and Motion-->
  <xs:list name="vector"
  itemType="xs:decimal"/>
  <!--Support for Semantic Level Text
  Annotations-->
  <xs:element name="annotation"
  type="xs:string"/>

  </xs:sequence>
  </xs:complexType>
</xs:element>
```

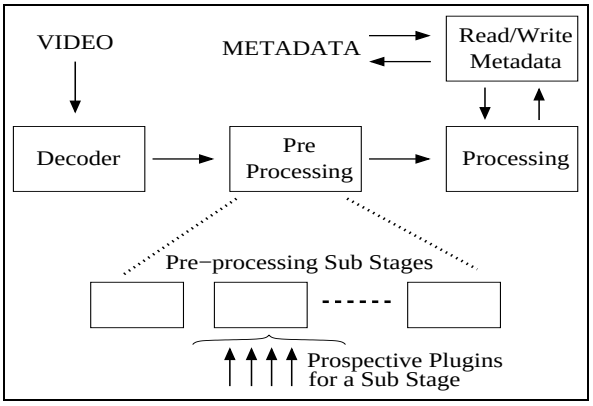

Fig. 3. Various stages of the processing pipeline. Each stage or a sub-stage might have several implementations which can be selected and sequenced at run time due to the plugin architecture.

E. IR Support and Delivery

Users are generally interested in looking for the most important news of the day, tracking a particular news story, avoiding watching irrelevant videos like commercials, watching related news across different channels, etc. Some of the successfully employed techniques in text domain can directly be mapped to the video domain. In a broadcast news delivery system, the users are interested to watch the videos sorted according to their importance. The user feedback can be used for providing personalized access to the news stories and also for collaborative filtering and categorization of the news content. Another responsibility of this layer is to deliver the output of these IR modules to the clients through a front end. A user activity log is also maintained which is a useful input for any kind of personalization. This layer is designed to vary the content presented and the user interface according to the client capabilities and requirements.

III. Software Design

The software has been designed based on the goals and requirements we started with. The primary design decision is to model the different stages of the software as a pipeline. Modeling as a pipeline requires the standardization of data structures throughout the different stages. Moreover, each stage in the pipeline needs to be independent of the internals of the other. This requires standardization of the interfaces among the different stages. To fulfill this, we chose a *Plugin architecture*. Any stage or a sub-stage can be manipulated within itself as long as the interfaces are conformed with. The plugin architecture allows us to have multiple implementations for a module. This enables run time selection of the best suited

Fig. 4. Screen Shot of the UI element designed for presenting a news story

algorithm depending on configuration setting and user input. The design is pictorially shown in Figure 3.

For each stage of the pipeline, a factory class is implemented which encapsulates the logic of selecting the appropriate algorithm implementation. Based on our experience with the initial versions of the system, we noticed that the memory allocation and deallocation appeared to be a significant overhead in the performance of the system since typical video processing tasks generally require huge chunks of memory. This issue was rectified with a simple design modification wherein we implemented *Object Pooling and Reuse*. Memory is allocated at the system initiation itself and the objects are reused over all subsequent requests.

Another major challenge was the design of the user interface. Users access video segments via web browsers. An effective and intuitive way of visualizing the videos is designed such that the user gets a feel of the content without actually needing to stream the videos and see them. Users are typically presented news broadcasts automatically segmented into individual stories. These stories are of few minutes duration each. Browsing this by playing the video requires significant time. We extract the critical frames called key frames and form a visual summary which is presented in an innovative manner along with the metadata. It allows users to know the relevant meta information (eg. language, channel, time of news, importance, etc.) associated with each story. The key frames from the story are arranged left to right and move in a slide show. Figure 4 shows a screentshot of this UI element. Any keyframe zooms out on mouse hover by the user. This enables the user to study the content in detail. Figure 5 shows a screenshot of this feature in action.

Fig. 5. The thumbnail zooms out on hovering the mouse pointer.

IV. Implementation and Performance Analysis

In its current form, each day the system indexes 100 news broadcasts from 8 different channels spanning 3 languages. The video is encoded in MPEG-1 format which is kept as the standard format throughout the system. Two processing servers with an AMD Athlon 3200+ processor and 1G of RAM each are in use. 1500GB of total storage spread over 5 storage nodes is in use. *LAMP* (Linux, Apache, MySQL, PHP) set of free software programs are employed to build the underlying software infrastructure.

There is an administrative interface to the whole system for configuring the system. The administrator can add or remove recording, storage and processing nodes on the system. The recording and processing schedules can be adjusted. The system sends out daily reports to the administrators and developers of the system which can be configured from here. The system also sends out email alerts to the administrators in case of failure of any node which can also be configured from this interface. Table II depicts the evolution of the system through last 20 months.

We performed extensive experiments and simulations for observing the performance of the system. A network of 14 state of the art desktop systems was employed for conducting the experiments. These systems acted as recording, storage and processing nodes. Configuration was changed according to each experiment and readings were automatically collected and stored in a database.

The first simulation was performed to analyze the capability of the system to handle large number of parallel video captures. The system was put under observation for a day and the average load on the maximally loaded storage location was plotted over every 10 minute interval. First three graphs in figure 6 show the results of the simulation. The three cases correspond to the central storage consisting of 6, 4 and 2 storage locations respectively. Each of these graphs contain different plots showing the load corresponding to the varying number of parallel video streams being captured. It can be seen that the load always remains below the line indicating the maximum possible load that can be handled by a single disk. This means that all the storage disks were having load lower

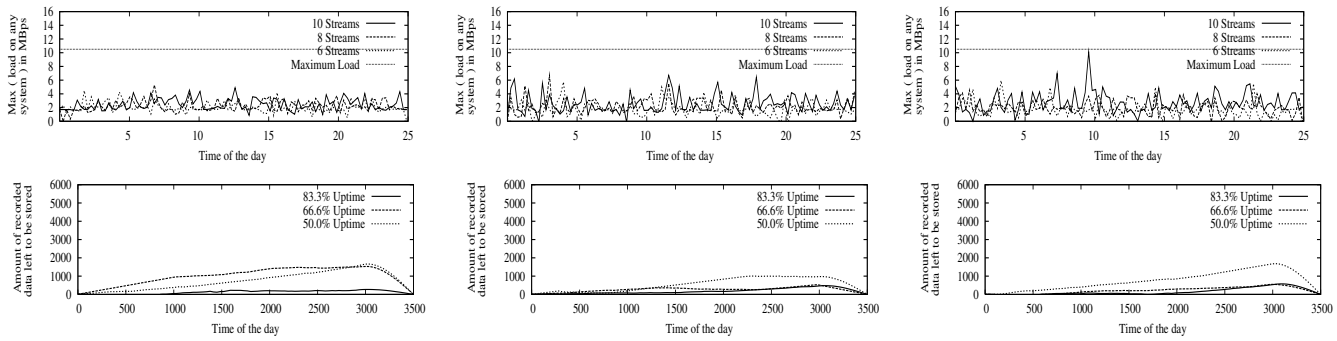

Fig. 6. Results for the Simulations.

TABLE I
SOME QUICK STATISTICS ABOUT THE SYSTEM

No. of different channels being captured	10
Number of news broadcasts recorded and processed each day	100
Approximate amount of Video Data processed till date	70 Tb

TABLE II
THE TABLE TRACKS SYSTEM'S CONTINUOUS EVOLUTION OVER LAST 20 MONTHS

	No. of System Failures per month	Total Storage Size (in GB)	Data Captured and Processed per day
Jan '06	20	200	10 hrs.
July '06	5	500	25 hrs,
Jan '07	0	1200	25 hrs.
July '07	0	1500	50 hrs.

than the maximum that they can handle. The results clearly show that the system is scalable and the system can handle increasing number of parallel recordings with the growing storage locations in the central storage.

The second simulation was performed to test the system's behavior in situations when the reliability of the central storage is low. The system is designed to ensure no loss of captured video data even when the central storage suffers from high average downtimes. The caching mechanism ensures safety of data even if storage is not immediately available at the time of recording. Last three graphs in figure 6 shows results of this simulation. Each graph shows the amount of recorded video data pending to be transferred on to the central storage over a day. The downtime of the storage systems has been simulated in three different ways to imitate different possible real life scenarios. In first case, a percentage of the storage locations were forced to be down permanently for the day. In second case, all the storage disks were forced to be down for a percentage of the duration in a day. In the last case, storage disks were randomly and alternately forced to be down for small durations so as to end up with the mentioned overall average uptime. Within each graph, there are multiple plots corresponding to the different average uptimes of the storage locations. The graphs clearly indicate that even in low uptime situations for central storage, there is no loss in data. At the end of the plot, the amount left to be transferred corresponding to the day of observation eventually reaches zero. There is a only a lag which is introduced in the data transfer from the recording cache to the central storage because of unavailability of enough free storage nodes. That is expected to happen as the transfer rate is limited by the total bandwidth of the central storage.

V. CONCLUSIONS

It is quite evident that the goals and requirements we started with are satisfactorily met because of the layered architecture and various design choices made. The system has been time tested over 20 months. The extensive performance analysis through stress simulations depict the robustness of the system. The system holds great potential for the development of innovative IR techniques. The system is shown to be capable of evolving into a much bigger platform hosting a variety of content based video access algorithms.

REFERENCES

[1] Youtube. [Online]. Available: http://www.youtube.com
[2] A. Jain, A. Vailaya, and W. Xiong, "Query by video clip," in *Proc. IEEE International Conference on Pattern Recognition (ICPR'98)*, vol. 1. Washington, DC, USA: IEEE Computer Society, 1998, pp. 909–911.
[3] A. Hampapur, A. Gupta, B. Horowitz, C.-F. Shu, C. Fuller, J. R. Bach, M. Gorkani, and R. C. Jain, "Virage video engine," January 1997, pp. 188–198.
[4] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker, "Query by image and video content: The QBIC system," *Computer*, vol. 28, no. 9, pp. 23–32, September 1995.
[5] J. R. Smith and S.-F. Chang, "Visualseek: A fully automated content-based image query system," in *ACM Multimedia*, 1996, pp. 87–98.
[6] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Proc. IEEE International Conference on Computer Vision (ICCV'03)*, 2003, pp. 1470–1477.
[7] Y. Zhai, J. Liu, and M. Shah, "Automatic Query Expansion for News Video Retrieval," in *IEEE International Conference on Multimedia and Expo*, 2006, pp. 965–968.
[8] A. F. Smeaton, "The físchlár digital library: Networked access to a video archive of TV news," in *TERENA Networking Conference*, 2002.
[9] O. Arandjelovic and A. Zisserman, "Automatic Face Recognition for Film Character Retrieval in Feature-Length Films," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 860–867.
[10] J. Yang and A. G. Hauptmann, "Annotating News Video with Locations." in *Proceedings of the International Conference on Image and Video Retrieval*, 2006, pp. 153–162.