A Generic Tool for Content-Based Multimedia Browsing

Bernard Mérialdo
Florence Dubois
Institut Eurécom
2229, route des Crêtes
BP 193
06904 Sophia-Antipolis Cedex
FRANCE
(33) 93.00.26.29
merialdo@eurecom.fr

Abstract

The automatic analysis of the contents of multimedia documents requires to combine informations coming from various data types (audio, video, text...). In this paper, we propose an architecture that describes agents for processing flows of information. These agents can be applied to elementary data types (audio, video...), but also on the results produced by other agents. The architecture includes a Multimedia Flow Browser, which is able to display simultaneously visual representations of the various information flows produced by agents, and an Agent Editor, which provides a graphical interface to create new agents by combining existing agents. The architecture is open, so that it is possible to add new data types (and the procedures to visualize them) and new agents. A simulated example is presented to show the possible usage of this tool in an application based on TV News recordings.

1. Introduction

A multimedia application mixes speech, audio, image, video and text processing and navigation, in order to offer an improved and transparent interface and provide a communication that is as natural as possible. The basic functions that can be performed on a multimedia document mainly use its syntactic structure and not its semantic content: operations such as cut-and-paste, play-back, go-to, fast-forward and rewind are only possible on indices such as byte count, time stamp, etc... More elaborate operations such as a search for a particular event or the generation a summary of the document, require an analysis of the contents of

the document. This is a very difficult task, because of the variety and complexity of the recognition techniques that are involved (speech recognition, image analysis, natural language understanding). Hence there is a need for tools which facilitate the development, experimentation, combination and integration of various indexing techniques.

As an example, suppose that we are interested in a given topic in a TV news recording. If we don't know the time where the presentation of this topic took place, we must playback the whole recording (even at higher speed) to find the desired spot. This would be time consuming. If an automatic system is able to detect, for example, events such as *˝Mister X is talking about topic Y˝*, then it will be possible to directly access the appropriate location. This event can be detected using information extracted from different components of the recording. For example, face recognition on the video part can be used to detect the presence of Mister X (another possibility would be speaker identification on the audio channel), word spotting on the audio part can identify the utterance of word Y, and captions could be searched for certain string patterns.

We propose an architecture which facilitates the creation of and experimentation with such detectors. They are built as a combination of agents. Some agents will process elementary data types (audio, video...), while some agents will combine the results obtained by other agents to create complex detectors. A first tool, the Multimedia Flow Browser, allows us to visualize simultaneously the original data and the results of the processing by different agents. A second tool, the Agent Editor, provides a graphical interface to easily construct new agents as the combination of existing agents. These tools are extensible, and it is possible to incorporate new visualization procedures for new data types, and new agents in the agent library. This architecture is inspired from the image processing Khoros software [Konstantinides94].

The paper is organized in the following way. Section 2 presents related research conducted in video indexing and retrieval. Section 3 gives an example of an application of the Browser. Section 4 gives a general overview of the architecture; we present the Browser, the different types of visualization that it currently provides, the search features that are available, and describe the Agent Editor. Conclusions are presented in section 5.

2. Related Work

A major issue in multimedia document processing is the construction of indices that are representative of the content of the document. This involves pattern recognition techniques, including speech, image and natural language processing, analysis and understanding. Such techniques have been extensively studied for a long time, however, for the most part, each media was studied in isolation. With the development of multimedia technologies and the wider usage of multimedia documents such as video recordings, a strong emphasis has emerged on the intelligent processing of multimedia documents.

Because text indices are easier to manipulate, many approaches use techniques derived from textual Information Retrieval (IR) [Salton89], by processing either the textual component of the document (for example the caption of a picture[Ogle95]), or textual annotations that have been manually added to the document [Weber94]. Although it is harder to define indices in non-textual data, such indices are potentially so useful that many projects tackle this difficult problem. Using speech recognition techniques such as word spotting, it is possible to detect the utterance of keywords in the audio component so that IR techniques can be used. The Video Mail project [Brown94, Jones95] at Cambridge, GB, is an example of this approach. [Schauble95] uses specific techniques to handle the problem of inaccurate keyword recognition. [Blum95] uses specially computed parameters to handle an audio database of sounds. Other approaches exploit the image component of the document, for example by computing and comparing textures. This is the case of the QBIC (Query by Image Content) Project at IBM [Flickner95, Petkovic95], and the PhotoBook Project at MIT [Pentland93]. Many approaches analyze the video component, where the basic operation is a segmentation of the video into consecutive shots using a cut detection algorithm [Arman94][Merialdo95]. This step often is followed by a more elaborate processing such as macro-segmentation [Aigrain95] or parsing [Zhang95].

While many projects use indices from a single media to process multimedia documents, some effort is also done to combine indices from a variety of different media, such as the multimedia episodes defined in [Gabbe94]. [Srihari95] combines natural language processing and image understanding to

create an automatic indexing system for captioned pictures of people, and [Rowe95] uses a related approach.

Because of the potential interest of users, broadcast news are the subject of many investigations to create, manipulate and process indices allowing intelligent processing such as filtering and retrieval. Examples of such systems are presented in [Brown95] [Hauptman95] [Mani95].

Finally, multi-agent systems are the focus of a great interest in the field of Artificial Intelligence [Lewis93] because they facilitate the implementation of complex behaviors. In particular, the coordination between agents, the informations they exchange, the protocols they should use, are extensively studied.

3. An Example of an Application

Assume that you want to know if a given topic has been discussed in yesterday's TV news. You can get the recording and process it with agents that are available in the architecture. For example, you might start by using a word spotting agent to locate the relevant utterances in the audio part. The Multimedia Flow Browser will allow you to visualize the video and audio part, together with the results of the word spotting agent. The resulting display will be similar to the one shown in Figure 1.

Suppose now that you want to detect a complex event such as *"Person X is talking about topic Y"*. This can be accomplished with a complex agent built by combining simpler agents, such as:

- Face Location (FL) agent: takes a video as input and produces a list of regions in the image that are classified as faces of people;

- Face Identification (FI) agent: takes as input an excerpt of the video which has been recognized as a human face and identifies the person (with reference to a given database);

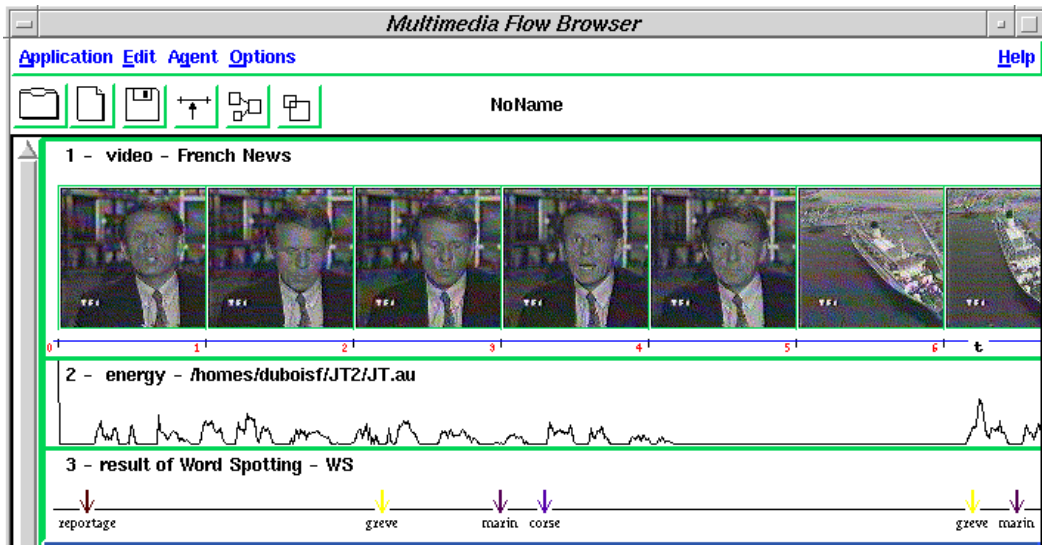


Figure 1 - An example of application using a Word Spotting agent.

- Face Movement (FM) agent: takes an extract of the video which has been recognized as a human face and checks if the person is currently speaking or not (for example by analyzing lip movement);

- Speaker Identification (SI) agent: takes as input an audio signal and determines the instants where a given speaker (from a set of known speakers) is talking;

- Word Spotting (WS) agent: takes as input an audio signal and determines the instants where certain words (from a predefined vocabulary) are pronounced;

- Caption Word (CS) agent: takes as input a text stream (words with time stamps) and determines the instants at which certain words appear.

Assuming that X is a particular person, and that topic Y is illustrated by a keyword, a possible structure of such a system is indicated in Figure 2.

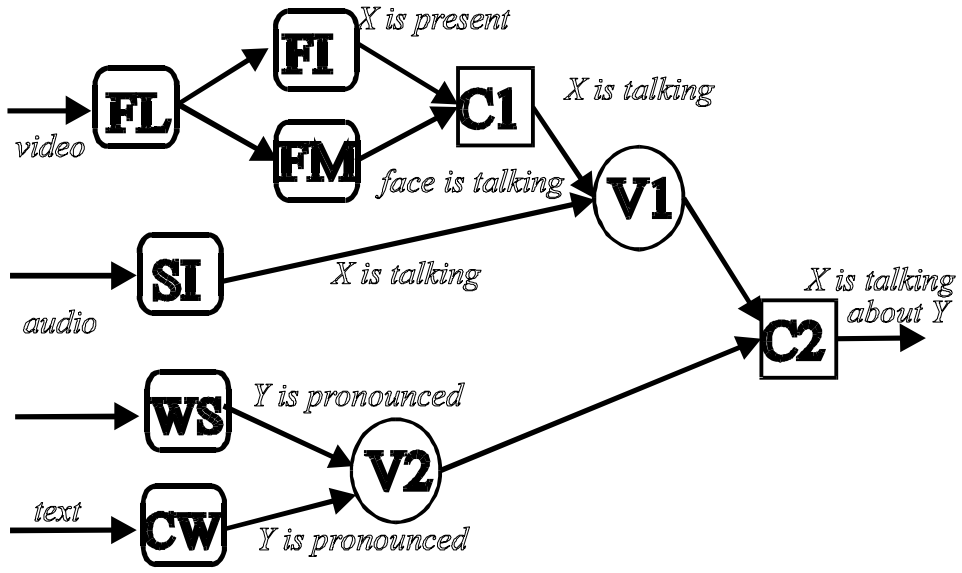


Figure 2 - A multi-agent architecture for detecting the complex event *"Person X is talking about topic Y"*

The video part of the recording is analyzed by the Face Location agent to detect face locations, which are then identified by the Face Identification agent. At the same time, the Face Movement agent will detect when the person is talking. The combination of the identity of the person and its lip movement gives an indication of the event *"X is talking"* based on the video source. In parallel, the Speaker Identification agent analyses the audio signal to detect when *"X is talking"*. The video and audio information are then combined to formulate an evidence for *"X is talking"*. Similar operations occur with other agents. This example shows that there are three different ways by which simple agents can be combined together:

- *succession*: when the output of one or several agents are used as input to another agent. This is the case when the Face Location agent provides information to the Face Identification agent;

- *validation*: when several agents provide information about the same event, and that their advice have to be combined in a single hypothesis. This is the case when we want to combine the output of the video and audio identification agents;

- *composition*: when the hypotheses formulated by various agents have to be combined to form a more complex hypothesis. This is the case when we combine the information *"X is talking"* with the information *"word Y is being pronounced"*.

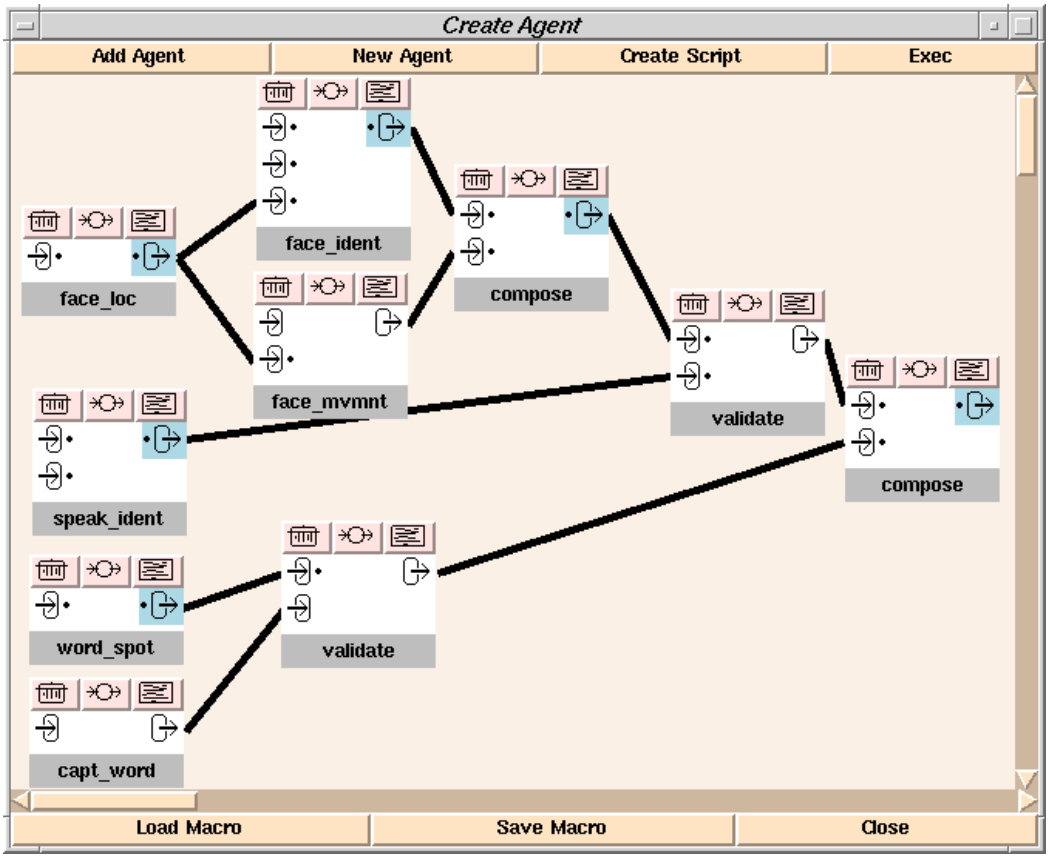


Figure 3 - Construction of the architecture with the *Agent Editor*.

The complex agent can be built with the *Agent Editor*, which offers a graphical interface to select and link together simple agents. Figure 3 contains an example of the display when creating the complex agent previously described.

Then, you can use the Browser to display some of the flows computed by the various agents, as shown in Figure 4.

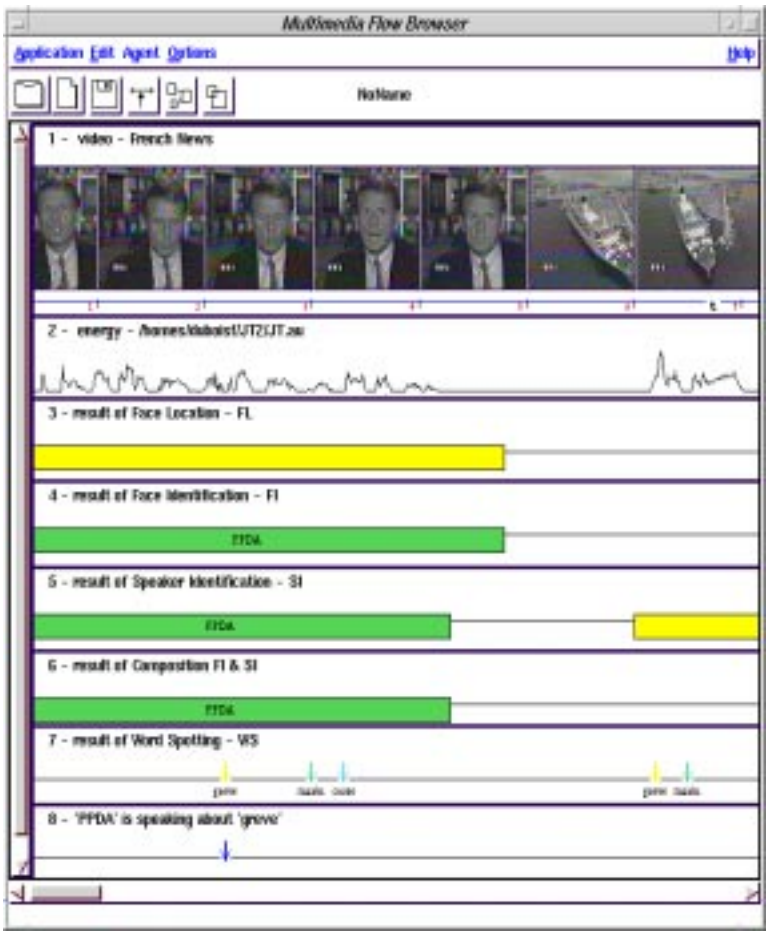


Figure 4 - The results can be visualized on the main window of the Multimedia Flow Browser.

4. The Multimedia Flow Browser

The role of the Multimedia Flow Browser is to visualize simultaneously several flows of data. Those flows might be of various data types, either continuous (such as audio, video) or discrete (events characterized by an instant or an interval, such as captions), but we assume that they are all related to the same timeline. They might have been either recorded from external sources (audio, video), or have been produced as the result of the processing of other flows by agents.

4.1 Visualization formats

The Browser provides a number of visualization formats for the standard data types. When the user wants to visualize a data flow, he can choose among the formats that are available. chooses which flows he wants to display on the Browser. Depending on the format that has been chosen, some computation will be performed by the Browser to compose the visualization window from the data flow. We give some examples of visualization formats that are currently implemented in the Browser.

A video flow can be represented as a sequence of consecutive images. These images can be placed either at regular time intervals or at each new cut detected (using a cut detection algorithm). A comparison of these formats is given in Figure 5.

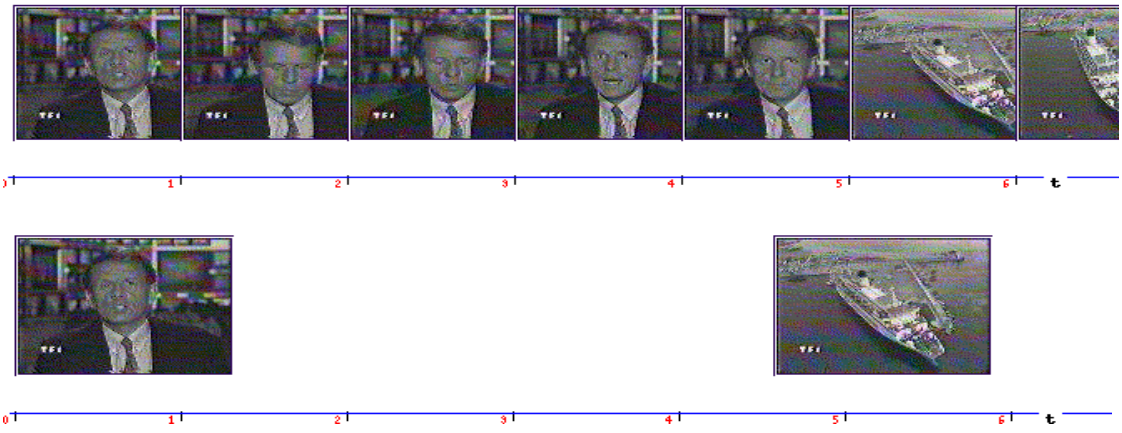


Figure 5 - Representation of video flows

The audio flow can be displayed as signal or energy (the energy is automatically computed from the signal). An example is provided in Figure 6.

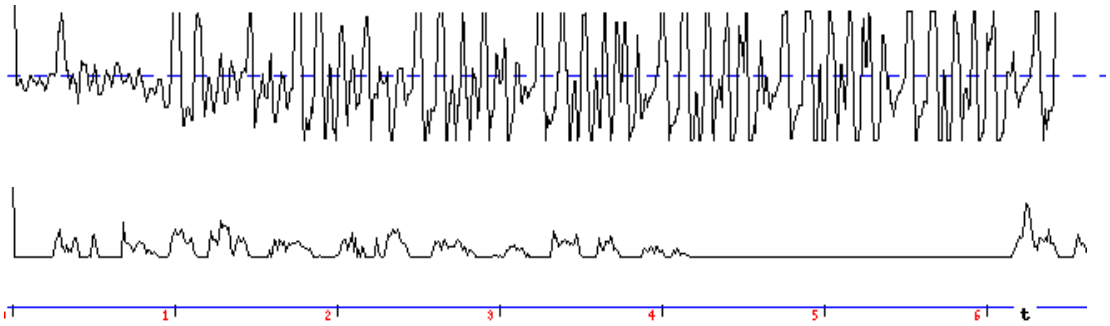
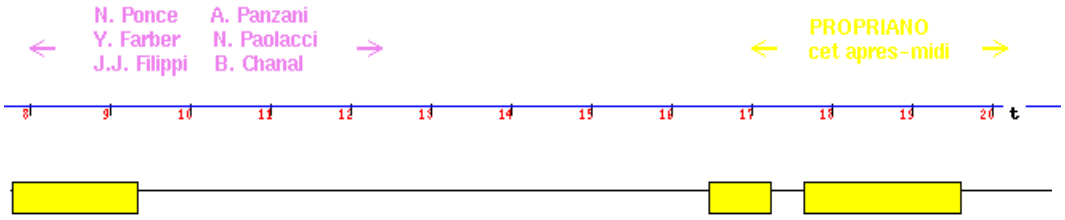


Figure 6 - Representation of audio flows

Representation of discrete flows is also possible. We have defined a few data types (and file formats) for describing such flows which contain text labels associated with either time spots or time interval definitions. Several visualization formats are provided to visualize these data types. They are constructed using arrows or boxes (to indicate starting and ending time) located on a timeline, text labels, and eventually colors to differentiate different labels. Examples illustrating these possibilities are provided in Figure 7.



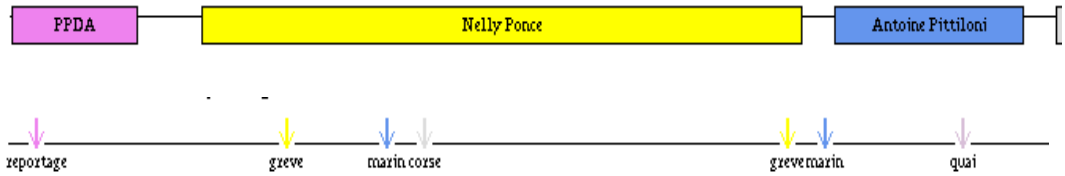


Figure 7 - Representation of discrete events

The Browser has been designed so that it is easy to add new data types, new visualization formats, and the procedures that are needed to implement them. When creating new agents, it is preferable to use an existing data type, for example, a word spotting agent can produce a data flow composed of time-stamped text labels. In the case where no existing data type is suitable, for example if we want to visualize the Fourier transforms of images, new visualization procedures will have to be written and added to the Browser.

The first function of the Browser is visualization. The user can select the data flows he wants to see, choose the visualization formats that will present these flows, and move the cursor on the time axis to explore the contents of these flows. He can also directly provide the time he wants to visualize. However these operations use only the syntactic structure of the data (perhaps with the exception of the cut detection algorithm for the video). We now describe other functionalities of the Browser that rely more on the content of the document.

4.2 Search and Indices

The Browser also provides Search and Index functions. The Search function is given a pattern by the user, searches for the next occurrence of this pattern in the data and positions the time cursor at the first instance found. The Index function displays an ordered list of possible patterns for a data flow, together with indications on the number and position of the occurrences of these patterns in the data.

Each data type has its own implementations of the Search and Index functions (we currently do not allow complex searches combining several data types to be performed; rather, we would create a new agent for this combination and search the data flow produced by this agent). It is also possible to have several implementations of a single data type, if the data contained in the flow can be observed from different perspectives. For certain data types, these implementations are straightforward. For example, data flows containing time-stamped text labels (like captions), can be easily searched for certain keywords in the text, or for certain constraints on the duration of the time interval. For other data types, the definition of patterns is more difficult. For example, how to define patterns that can be used to describe an image in a video is far from obvious.

Similar problems arise in the implementation of the Index function. Not only should we define the patterns that have to appear in the Index, but we should also define an ordering on these patterns to build the index. In the case of time-stamped labels, the patterns might be the labels themselves, listed in order of occurrence. Figure 8 is an example of an index created from the results of a word spotting agent. In the case of video, we can provide a list of different images appearing in the video (of course images that are only slightly different are considered identical), sorted by decreasing number of occurrences.

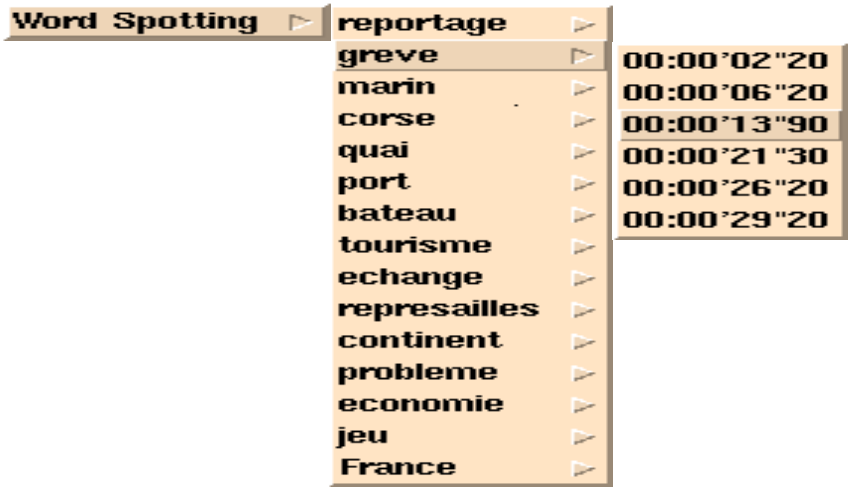


Figure 8 - Example of indices resulting from word spotting

4.3 The Agent Editor

Agents process one or several input data flows and provide their results as one or several output data flows. Agents can be of two types : simple or complex. Simple agents are stand-alone procedures (such as word-spotting). The architecture provides conventions for defining their input and output parameters. Complex agents are combinations of simple or complex agents. They are created using the Agent Editor which provides a graphical interface to select and link agents together. The user will select the agents he wants to use in the library of available agents, and they will appear as boxes in the graphical interface with slots indicating the possible input and output flows. To connect agents together, he will then link the output to certain agents to the input of the adequate agents by drawing lines from/to the corresponding boxes.

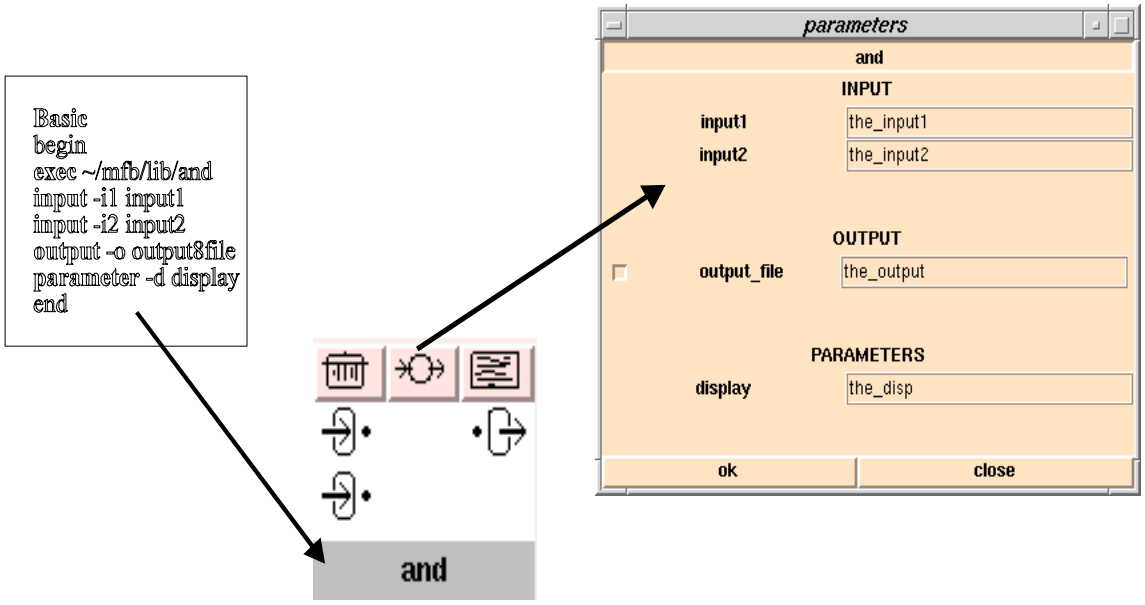


Figure 9 - Agent Representation

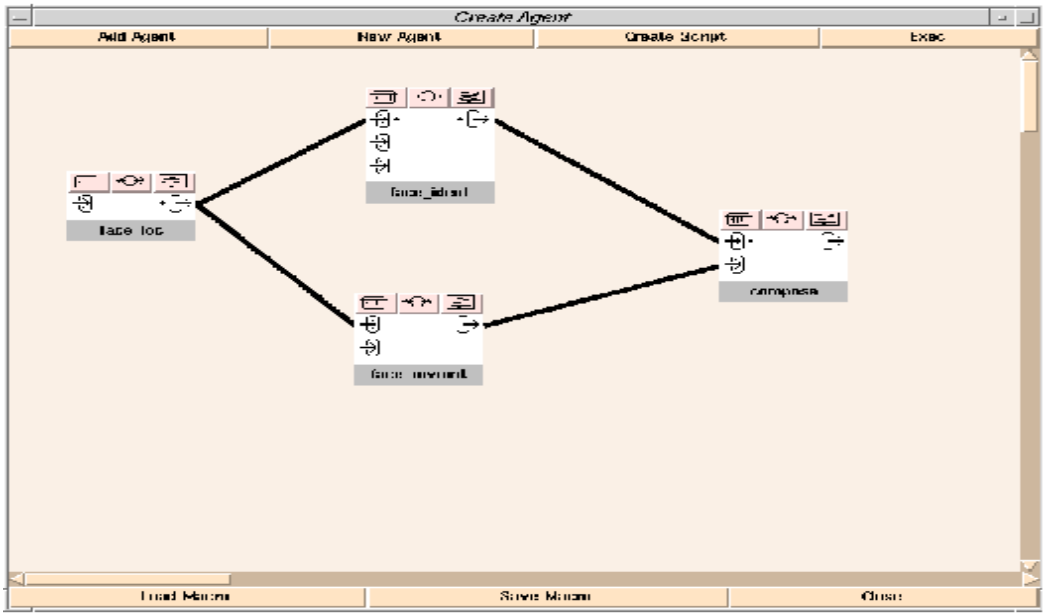


Figure 10 - Example of combination of agents with the *Agent Editor tool*.

Once they are created, complex agents are included in the library of available agents, and can be modified or reused to build other agents.

The Agent Editor and the Browser are linked together. It is possible to call the Agent Editor from the Browser to display or modify an existing agent, create a new one. It is also possible to select intermediate data flows that will be visualized in the Browser window. The Browser is able to run agents to construct

new data flows. Before running an agent, the user has to define the data flows that will be used as input, and the output flows that he wants to visualize. If the agent is a simple one, the Browser will simply start the corresponding program with the adequate parameters. If the agent is a complex one, the Browser will

decompose it into simpler agents that it will call sequentially, so that an agent is started only when its input data flows have been created.

4.3 Implementation and limitations

Prototypes of the Multimedia Browser and the Agent Editor have been built using the Tcl/Tk language/toolkit [Ousterhout94], so that adding new procedures is easy. In the first version, the processing of agents was simulated (results files were created by hand or by random generation). Some simple agents, such as face recognition [Clergue95] or word spotting [Gelin96], are now being built. A number of visualization procedures have been defined for the basic data types (video as sequence of shots, audio as signal or energy graphs), together with results indicating labeled intervals, and labeled time spots. Search and Index procedures have been implemented for time-stamped text labels, and experiments are being conducted on video data flows. The table below contains a brief summary of our experimentations.

Data type	*Visualization format*	*Search procedure*	*Index procedure*
video	images at fixed rate	image similarity	list of different images
	images at cuts		
audio	signal	amplitude or energy threshold	*none*
	energy graph		
text label	arrows	keyword search	list of keywords
	boxes		

A strong limitation of this version is that it assumes that all the flows have the same timeline, so that when using the Browser, all the visualization windows are updated simultaneously. This restricts the usage of the Browser to a single " document ". The user interface is designed for experimentation by agent developers, so that it is not appropriate for the potential end-user of such agents.

Another limitation is that there is currently no synchronization mechanism between the agents when they process data. When a complex agent is applied on some data, the simple agents are simply ordered and called one after each other, so that an agent is called when the flows that it should process are available. Ideally, we could think of a more elaborate strategy where the results found by certain agents would influence the processing of others, so that, for example the agent do not look for a face on the video when nobody is talking on the audio.

5. Conclusion

The Multimedia Flow Browser and the Agent Editor are research tools that are currently being developed. They facilitate the experimentation of agents which analyze the contents of multimedia documents to detect the occurrence of complex events. They have been designed with an open architecture that allows us to easily include new features: adding a new data types, new visualization formats, new agents, new indexing schemes, etc...

The development of the Browser raises a number of issues concerning the processing of various data types, for example on the possible ways to implement search and index functions on non-textual data types.

Our intention is to build an basic library of agents that users will be able to extend and combine to fulfill their needs. We are currently starting to use the tool to experiment on real-life situations, such as TV news indexing.

6. References

[Aigrain95] Phillipe Aigrain, Philippe Joly, and Veronique Longueville. Medium knowledge-based macro-segmentation of video into sequences. In IJCAI'95, Workshop on Intelligent Multimedia Information Retrieval, August 1995.

[Benedetti96] Gerard Benedetti, Benoit Bodin, Franck Lhuisset, Olivier Martineau, and B. Merialdo. A structured Video Browsing Tool, pages 17-26. Chapman & Hall, 1996.

[Blum95] Thom Blum, Douglas Keislaer, James Wheaton, and Erling Wold. Audio databases with content-based retrieval. In IJCAI'95, Workshop on Intelligent Multimedia Information Retrieval, August 1995.

[Brown94] Martin Brown, Jonathan Foote, Gareth Jones, Karen Sparck-Jones, and Steve Young. Video mail retrieval by voice: An overview of the cambridge/olivetti retrieval system. ACM Multimedia Conference, Workshop on Multimedia Database Management Systems, October 1994.

[Brown95] Martin Brown, Jonathan Foote, Gareth Jones, Karen Sparck-Jones, and Steve Young. Automatic content-based retrieval of broadcast news. ACM Multimedia Conference, November 1995.

[Clergue95] E. Clergue, M. Goldberg, N. Madrane, and B. Merialdo. Automatic face and gestual recognition for video indexing. IWAFGR95 Workshop, June 1995.

[Flickner95] Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, Dragutin Petkovic, David Steele, and Peter Yanker. Query by image and video content: the QBIC system. IEEE Computer, 28(9):23-32, September 1995.

[Gabbe94] John D. Gabbe, Allen Ginsberg, and Bethany S. Robinson. Towards intelligent recognition of multimedia episodes in real-time applications. ACM Multimedia Conference, pages 227-235, October 1994.

[Gelin96] Philippe Gelin and Christian Wellekens. Keyword spotting for video soundtrack indexing. In Proceedings of the IEEE ICASSP96, 1996.

[Hauptman95] Alexander G. Hauptman and Michael J. Witbrock. Informedia: News-on-demand multimedia information acquisition and retrieval. In IJCAI'95, Workshop on Intelligent Multimedia Information Retrieval, August 1995.

[Jones95] Gareth J. F. Jones, Jonathan Foote, Karen Sparck Jones, and Steve J. Young. The video mail retrieval project: Experiences in retrieving spoken documents. In IJCAI'95, Workshop on Intelligent Multimedia Information Retrieval, August 1995.

[Konstantinides94] K. Konstantinides and J. R. Rasure. The khoros software development environment for image and signal processing. Ieee Transactions on Image Processing, 3(3):243-52, 1994.

[Lewis93] A. H. Lewis. Modeling and Design of Artificial Intelligence Systems. Kluwer Academic Publishers, 1993.

[Mani95] Interjeet Mani, Morgan Green, David House, and Mark T. Maybury. Towards content-based browsing of broadcast news video. In IJCAI'95, Workshop on Intelligent Multimedia Information Retrieval, August 1995.

[Ogle95] Virginia Ogle and Michael Stonebreaker. Chabot: Retrieval from a relational database of images. IEEE Computer, 28(9):40-48, September 1995.

[Ousterhout94] J. K. Ousterhout. Tcl and the Tk toolkit. Addison-Wesley, 1994.

[Pentland93] A. Pentland, R. Picard, and S. Sclaroff. Photobook: Tools for content-base manipulation of image databases. Technical report, MIT Media Lab Vismod, 1993.

[Petkovic95] Dragutin Petkovic, Myron Flickner, Harpreet Sawhney, Wayne Niblack, Jonathan Ashley, Qian Huang, Byron Dom, Monika Gorkani, Jim Hafner, Denis Lee, David Steele, , and Peter Yanker. Query by image and video content: The qbic system. In IJCAI'95, Workshop on Intelligent Multimedia Information Retrieval, August 1995.

[Roth95] Steven F. Roth, Mei C.Chuah, John Kolojejchick, Joe Mattis, and Octavio Juarez. Sagebook: Searching data-graphics by content. In IJCAI'95, Workshop on Intelligent Multimedia Information Retrieval, August 1995.

[Rowe95] Neil C. Rowe and Brian Frew. Automatic retrieval of objects in captioned depictive photographs. In IJCAI'95, Workshop on Intelligent Multimedia Information Retrieval, August 1995.

[Salton89] Gerard Salton. Automatic Text Processing. Addison-Wesley, 1989. iindexSalton, G.

[Schauble95] Peter Schauble and M. Wechsler. First experiences with a system for content-based retrieval of information from speech recordings. In Proceedings of the IJCAI95, Montreal, Canada, pages 59-69, 1995.

[Srihari95] Rohini K. Srihari. Automatix indexing and content-based retrieval of captioned images. IEEE Computer, 28(9):49-56, September 1995.

[Weber94] Karon Weber and Alex Poon. Marquee: A tool for real-time video logging. In Proceedings of ACM CHI'94 Conference on Human Factors in Computing Systems, volume 2 of PAPER ABSTRACTS: Multimedia in Use, page 203, 1994.

[Zhang95] H. J. Zhang, S. y. Tan, S. smoliar, and G. Yihong. Automatic parsing and indexing of news video. Multimedia Systems, 2:256-266, 1995.