

The Community of Multimedia Agents

Gang Wei, Valery A. Petrushin and Anatole V. Gershman

Accenture Technology Labs
161 N. Clark Street
Chicago, IL 60601
{gang.wei, valery.a.petrushin, anatole.v.gershman}@accenture.com

Abstract. Multimedia data mining requires the ability to automatically analyze and understand the content. The Community of Multimedia Agents project is devoted to creating a community of researchers and students who are interested in developing multimedia annotation algorithms. It provides an open environment for developing, testing, learning and prototyping multimedia content analysis and annotation methods. It serves as a medium for researchers to contribute and share their achievements while protecting their proprietary techniques. Each method is represented as an agent that can communicate with the other agents registered in the environment using templates that are based on the descriptors and description schemes in the MPEG-7 standard. Using the standard allows agents that are developed by different organizations to operate and communicate with each other seamlessly regardless of their programming languages and internal architecture. A development environment is provided to facilitate the construction of media analysis methods. The tool contains a workbench, which allows the user integrating agents to build more sophisticated systems, and a blackboard browser, which visualizes the processing results. It enables researchers to compare the performance of different agents and combine them to build a rapid prototype of more powerful and robust system. The Community can also serve as a learning environment for researchers and students to acquire and exchange of cutting edge multimedia analysis algorithms.

1 Introduction

The extraction of information from multimedia data is of vital importance with the explosive growth of digitized image, audio and video data. It requires the ability to automatically analyze, understand and annotate multimedia content. A large number of techniques have been proposed in this area, ranging from simple measures like color histogram for image, energy estimates for audio signal, to more sophisticated systems like speaker emotion recognition in audio [1], automatic summarization of TV programs [2] and topic detection and tracking using audio transcripts [3]. However, the capability of the current techniques is still far from the requirements of the applications in practice, especially in terms of intelligence level and robustness. For example, even the most advanced face recognition algorithms can be easily fooled by a little makeup or environmental changes. We believe that the reliable understanding

of multimedia content has to be achieved by the interaction of a number of specialized, effective and relatively simple modules (agents) that address different aspects of the content. A number of research efforts have been made in this direction, producing encouraging results, such as the TV genre classification based on face and superimposed text detection [4], and the use of both audio and video information to analyze multimedia content [5].

To enable the further progress in the field sharing and integration of algorithms developed by different researchers and organizations is required. To make this possible three major issues need to be addressed. First, it is the data compatibility, which allows exchanging information among agents. The Multimedia Content Description Interface standard (MPEG-7), which is ISO/IEC International Standard 15938 [9], promises to provide a unified base for multimedia content description for both producers and consumers. Second, agents should not expose the proprietary techniques (inventions), which they are built on. Finally, a development environment is needed to facilitate the manipulation of the agents and visualization of the processing results. The Community of Multimedia Agents project is devoted to solving the above problem.

2 Motivation

Multimedia content analysis requires expertise in a number of fields such as image and video processing, audio processing, speech recognition, linguistics, information retrieval and knowledge management. The range of expertise spans from DSP techniques for feature extraction to methods for knowledge representation, integration and inference. Unlikely a researcher or a research laboratory can cover the required range of expertise to develop a multimedia analysis system from scratch. Usually, each laboratory concentrates on its own research agenda using commercial tools (if available) or borrowing some experimental tools from other researchers to develop a rounded-up multimedia analysis prototype. Borrowing from the others is not easy due to the variety of platforms, programming languages, data exchange formats and unwillingness of companies to disseminate their intellectual property unprotected. A lucky researcher can get a tool that covers a particular task, for example, face detection; an unlucky researcher has to implement a tool by himself. In any case, the researcher will have only one (if any) face detector, in spite of his awareness that two dozens of such tools exist in the world. On the other hand, many innovative tools do not go beyond papers in conference proceedings or journals. The chances for these tools to be used by other people heavily depend on their exposure to Web search engines and other indexing tools. Many researchers would be happy to share their tools if it does not take too much effort.

This scarcity of media analysis tools and difficulty finding and sharing them motivated our project. The project's general objective is to create a virtual community of researchers, who exchange their multimedia analysis tools and test data. The Community's objective is to consolidate efforts and expedite research and education in multi-

media analysis field. To facilitate exchanging and combining media analysis tools the following requirements are held:

- The Community provides a library of multimedia analysis agents. Any community member can submit and download agents. Every Community member gets a license for free usage of agents for research and educational purposes.
- Agents exist in formats that can be directly used as modules to build larger systems, however the proprietary techniques are hidden from the users.
- All rights belong to the agents' authors or their organizations.
- The Community is located on the World Wide Web and agents are accessible from any Internet workstation.
- The Community provides MPEG-7 based templates for agents' outputs that facilitate communication among agents and allow building hierarchies of agents.
- The Community provides open source tools for creating agents and visualizing their performance. These tools can be freely downloaded from the Community Web site.
- The Community also is a learning environment. It provides information about related business and academic news, overviews of achievements of lead laboratories and researchers, event and job announcements, book and paper recommendations, tutorials, and glossary of specialized terms. It also includes a directory of community member e-mail addresses and chat rooms for real-time discussions. Altogether the tools and information form a socio-technical learning environment that could be beneficial for researchers, teachers and students.

Several questions about the above sketch of the Community design arise.

Why should a researcher or a student join the Community?

As a Community member a researcher has the following benefits.

- The researcher can develop a new media transformation or annotation algorithm and compare it to the best algorithms (agents) in the library. The Community provides testing data and evaluation tools. Now the researcher will be prepared to answer the question whether he or she compared his/her algorithm to the best ones.
- The researcher can combine agents to create a number of rapid prototypes of an annotation system to choose the most efficient solution.
- The researcher can use the generic agents, such as a Gaussian Mixture Model agent or a Hidden Markov Model agent, to train models and use them for prototyping.
- The researcher can use the agents' results in the form of MPEG-7 annotations for further processing outside the Community tools.

A student can use the Community resources for:

- Learning media processing and annotation algorithms.
- Developing his/her own algorithms and compare their performance to the best ones.
- Take part in competitions for developing agents, which will be conducted by the Community and its sponsors.

- Communicate with the Community members discussing the issues of interest.

What is an agent? Why is it a good idea to use agents?

According to [6], agents “are defined as active, persistent software components that perceive, reason, act, and communicate”. This definition being projected to the multimedia annotation field covers the wide spectrum of applications from a feature extraction routine to an autonomous system that searches for a particular multimedia content on the Web. However, currently a Community agent is a program in executable form that has media file(s) and/or MPEG-7 file(s) as input and output. Currently, agents work asynchronously and interact via annotation and/or media files. Such architecture does not support real-time processing, but it is appropriate for many media annotation prototyping tasks. The agent-based approach proved to be very useful in many applications. We found that the concept of agent is highly valuable for multimedia analysis. Most of the multimedia processing systems uses agents (in the above mentioned sense) implicitly or explicitly [7, 8].

Why should agents use MPEG-7?

The Multimedia Content Description Interface Standard (MPEG-7) [9] is an XML-based language that is developed for multimedia annotation. It is the hope and future of the multimedia processing and using industries. Using this standard as an interface language for agents allows easily converting media annotation prototypes into industrial applications, which suppose to be an MPEG-7 based systems. To facilitate development of agents using MPEG-7, the Community tools provide templates, which could be considered as macro definitions of MPEG-7 descriptions. For example, a template for object description allows the user to read and write an instantiation of the object by specifying only the object’s type, its location and frame number. These parameters are translated into a well-formed MPEG-7 description.

How are the Community tools different from the MPEG-7 eXperimentation Model?

The eXperimentation Model (XM) is not a development tool. It is the tool for evaluating and selecting descriptors and description schemes that belong or eventually will be included in the MPEG-7 Standard. This tool is restricted to implementation of the MPEG-7 descriptors and descriptor schemes only. It is an open tool and a researcher can use some pieces of its code to develop an agent. But the XM supports neither agent development and aggregation nor visualization of agents’ results. Another difference is that each MPEG-7 descriptor or description scheme is represented in the XM by only one piece of code while the Community welcomes multiple implementation of the same descriptor/ description scheme or functionality. We believe that having many redundant agents for the same task can improve results and increase robustness of aggregated systems.

What is the future development of the Community?

Currently, we foresee the following stages in developing the Community.

Stage 1. Simple agents. At this stage the Community Agent Library will accumulate a number of simple agents, which perform simple annotations and transformations. The major tasks at this stage are the following.

1. Developing tools for creating agents and visualizing their work.
2. Developing MPEG-7 based templates for agents’ outputs.
3. Accumulating initial "critical mass" of agents.

4. Developing XML-based internal representations for agents, scripts and results.
5. Developing and launching the Community's Web site.

Some items of the above agenda are already accomplished. The Accenture Technology Labs completed the first version of the Development Environment platform for Windows 2000/XP, developed internal representations for agents and scripts, developed some templates and the Community Web site, which is available at <http://community.techlabs.accenture.com>. Researchers and students from several universities contributed several dozens agents to the starting version of the Agent Library.

Stage 2. Intelligent Agents. The next step is to create more sophisticated agents that are built by aggregating low-level agents. Some of these agents or tools could be able to automatically combine (synthesize) agents to solve a specified problem. This will require representing the functions of the agents at the semantic level, and such techniques as Resource Description Framework (RDF) [10] or the emerging DARPA Agent Markup Language (DAML) [11] could be valuable for representing the ontology of the agents.

Stage 3. Distributed Intelligent Agents. The further step is to develop formal specifications, interfaces and tools that allow distributed agents to find each other on the Web and to communicate and solve a specified problem. At this stage the Community of researchers will be extended to the Community of Multimedia Agents to justify its name. Some steps toward creating simple business-oriented agents that work on the Semantic Web have already been made [12].

Why the Community is an open environment?

First, the Community is open to everyone who is able and would like to contribute. Second, the Community's tools are open too. There are several levels of openness. The Agent Library is open and anybody can contribute as many agents as he or she can. The set of templates is open to researchers who would like to extend agent developing tools. Finally, the source code of development tools can be downloaded and modified to create more efficient tools.

3 Architecture

The architecture of the Community's tools is presented on Figure 1. The Development Environment consists of two tools: a Workbench and a Blackboard Browser, which are responsible for the creation of multimedia analysis processes using agents and the visualization of the results, respectively.

The user starts using the Development Tool with selection of a multimedia file or a collection of files to be processed. Three types of media files are allowed: still images, audio files, and video files. Then the user builds a multimedia analysis algorithm by loading and connecting agents from the Agent Library to the Workbench. Each media object has an associated "Metadata Sheet" in XML format, which is the directory of the processing results produced by the agents. When the agents are invoked, the Workbench coordinates the data flow between the agents and updates the Metadata

Sheets of the media files. The Blackboard Browser visualizes the results generated by the agents providing a summary of the media content.

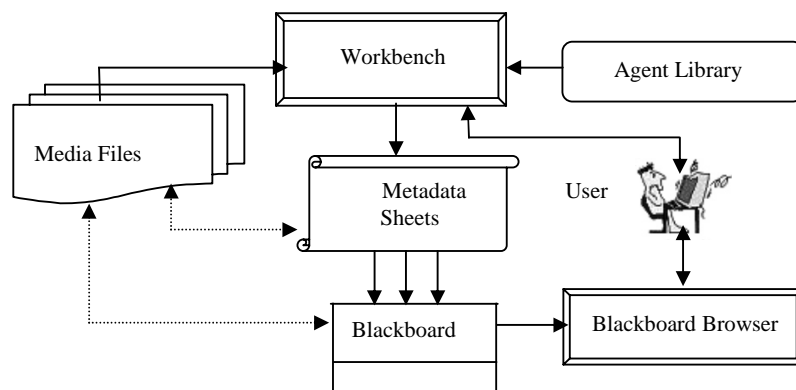


Fig. 1. The Community's Tools Architecture.

4 Development Tool

The Development Tool consists of two major parts: the Workbench and the Blackboard Browser. The Workbench allows a user to select and combine existing agents as building blocks to construct multi-agent systems. The user starts by selecting a media file. The media file is represented as a rectangle with a number of dots at the bottom (Figure 2). The largest dot corresponds to the raw media data. The other smaller dots, if any, are the processing results previously produced by the agents. Those results are recorded in the Metadata Sheet for the media file and can be used as inputs to other agents to avoid repeated computation and significantly reducing overhead, especially for time-consuming video processing algorithms. Some of the dots can represent "ground truth" data, which is manually prepared data that presents expert knowledge about a particular aspect of the media file. For example, for the face detection problem the ground truth data contains a list of frame numbers and coordinates of the areas that contain faces. The ground truth data serves for both intuitive and formal comparison of an agent performance with an expert's opinion.

The Workbench filters the agent library and displays only the agents that can process the selected media file. The agents are organized by their functionality in a tree structure in the top-left area as it is depicted on Figure 2. This figure shows that a video file in the MPEG-1 format is selected, and a number of agents of the following five categories Object Detection, Conversion (Transformation), Feature Extraction, Event Detection and Classification are available. The user can search for desired agents by expanding the tree. When an agent is highlighted, its description is shown in the text box below the agent tree so that the user can learn more about the agent. By clicking the "Load" button, the highlighted agent is loaded to the working space, and

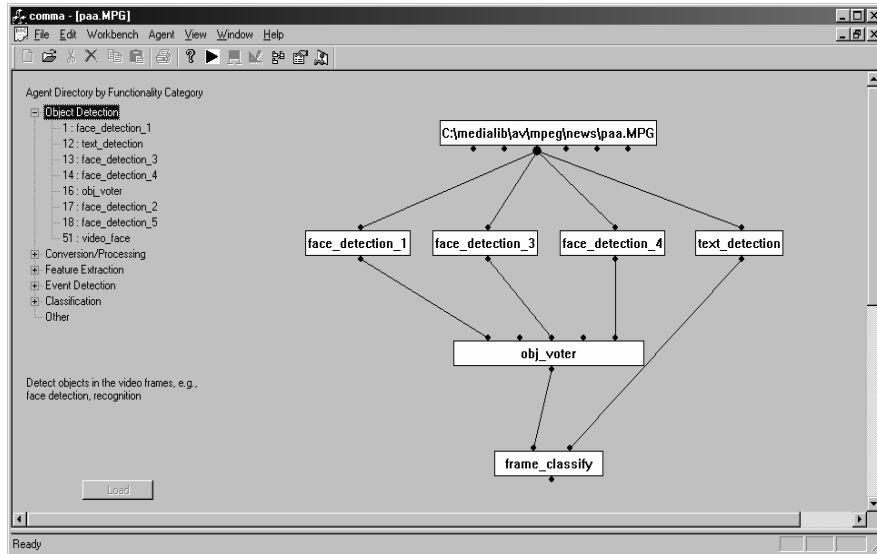


Fig. 2. Workbench window of the Development Environment

presented as a rectangle. The dots above and under the rectangle correspond to input and output pins, respectively. The user can build media annotation processes by connecting the media and agents. Figure 2 gives an example of how to integrate agents to build a more sophisticated agent. Let us consider the scenario where a researcher needs to create an agent that assign the video frame into predefine categories (e.g., “frame with face only”, “frame with text only”, “frame with both text and face”, and “frame without faces or text”). Without the Community of Multimedia Agent, the researcher may have to re-implement some face and text detection algorithms or create his/her own ones. Using the Community’s tools the researcher can simply design an agent that takes the results of face and text detection agents as input, and produces classification labels, such as the “*frame_classify*” agent that is depicted in Figure 2. Compared with developing every component from the scratch, a lot of time and efforts can be saved. The user can also save the system composed of agents as a script and later load it as a “macro-agent”.

On the other hand, with the availability of more than one face detection agents, their results can be combined to obtain more reliable performance. Since the face detection agents may employ various algorithms, e.g., neural network, color-shape analysis, etc., each may have its own strengths and weaknesses at different occasions. The overall accuracy can be improved by having a voting committee among the several agents. This can be accomplished by the “obj_voting” agent in Figure 2, which is a simple agent that accepts the results of up to five object-detection agents. It has a parameter (tuner) that specifies the mode of voting, which could be “or” (a frame has a face detected if at least one of the agents detects a face), “and” (if all agents detect a

face) or “majority” (if the majority of agents detect a face). It has been proved that a voting committee can produce more accurate results than any of its members when the errors of the members are uncorrelated with each other [13].

By clicking the triangle button on the toolbox bar the user can run the script. The Workbench coordinates the script execution and manages the annotations. Currently the tool does not support parallel or real-time execution of agents.

The user can use any programming language to develop an agent if the final result is an executable module. The system also allows using any interpretive language for agent development, but installation of the interpreting program should be done separately. To facilitate agent creation the Community provides tools and tutorials for using C++ and Perl as the development languages. We also plan to include Java and MATLAB to the list.

Therefore with the growth of the agent library, the Community users will be better equipped to address a variety of problems, and can use the Development Tools for rapid prototyping of media annotation systems.

5 Blackboard Browser

The Blackboard Browser visualizes the results produced by the agents to provide insight about the media content and allows the user having an intuitive evaluation of the performance of the agents. Each agent can generate one or more XML files through its output pins, and the data formats conform to the MPEG-7 based templates associated with the pin types. The references to these result files are recorded in the Metadata Sheet of the media file, and the Blackboard Browser parses the Metadata Sheet to retrieve and visualize them.

Figure 3 shows a Blackboard window for a video file. It contains video browser on the right side, a current frame image on the left side that presents agents’ results, and a summary of agents’ findings for the current frame in the middle of the screen. The user can watch the results for any frame using the navigation buttons. The tool allows the user to view the agent results at different granularities using “Zoom” buttons. Below the frame and time scales there are the summaries of agents’ findings for the whole clip or for the zoomed portion of it. Figure 3 presents the output of six agents. It shows the waveform as an output of a simple audio processing agent in a form of a graph. It also presents the output of five classification agents. Three agents of five are real media processing agents (face detection, text detection, and gender classification), two agents are ground truth agents for face detection and text detection tasks. The agents’ results are presented in a form of the colored horizontal bars aligned with the timeline. For example, the summary of a face detection agent is presented as a three-value color bar. Each frame can be categorized as “no faces detected” (white color), “one face detected” (blue color), and “multiple faces detected” (red color). The same color code is used for the text detection agent’s results. The results of speaker gender classification agent [14] also are represented as three-value color bar (see the bottom color bar), where the segments corresponding to “no speaker” are painted in white, “male speaker” - in yellow, and “female speaker” - in red. A user can explore how a

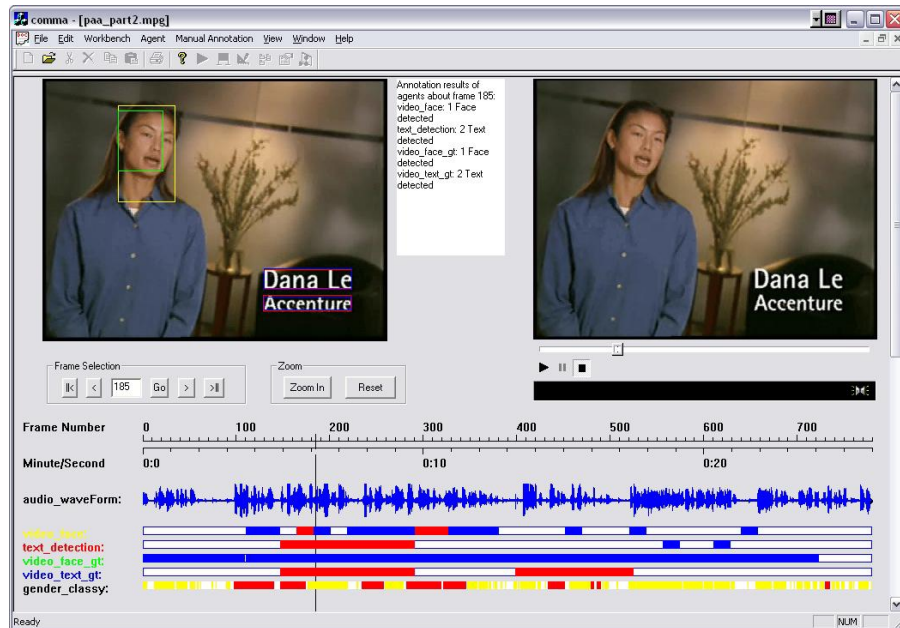


Fig. 3. Blackboard Browser for Result Visualization

particular detection agent works by clicking on the agent's summarization strip and watch the results represented on the current frame picture as a rectangular that frames a detected face or text. Or by clicking on the time scale the user can watch the results of all agents simultaneously on the same picture. The user also can compare agents' performance with ground truth data. For example, comparing the performance of a simple frame-based face detection agent (represented by the top bar) with the ground truth data (the third bar) the user can see that the agent missed about 60% of frames with faces and found multiple faces in frames where only one face is present (false alarm). Comparing the performance of the video text detection agent (the second bar from top) with the ground truth data (the fourth bar) the user can see that the agent perfectly found multiple texts in the first shot, but completely missed texts in the second shot and introduced some false alarms at the end of the video clip.

6 Agent Library

The Community provides a library of multimedia processing and annotation agents that serve as building blocks for more sophisticated, powerful and robust systems. Each agent is an independent application and can be developed by different researchers or organizations.

Table 1. Classification of Agents

	Video	Image	Audio
Object Detection	Detect visual objects in video frames	Detect visual objects in image	Detect a type or source of the sound such as speech, music, noise, etc.
Object Tracking	Track and model the movement of visual objects in video sequence	N/A	Tracking the source of a sound
Classification	Assign video segments into predefined categories	Assign image into predefined categories	Assign video segments into predefined categories
Event Detection	Detect specific visual events that take place during certain period or at certain instant	N/A	Detect specific audio events
Conversion / Transformation	Apply operations or filters to the video	Apply operations or filters to the image	Apply operations or filters to the audio
Feature extraction	Extract features that can be represented numerically	Extract features that can be represented numerically	Extract features that can be represented numerically
Summarization	Summarize video content	Summarize image content	Summarize audio content
Interpretation	Inference about a situation, commercial detection, etc.	Inference about depicted situation, etc.	Speech recognition, topic detection, etc.

The agents in the Library are organized by the media they can process and their functionality. Table 1 presents a classification of agents based on agent functionality and media category. For each agent class, the Community provides open-source tools to facilitate developing agents in different programming languages.

Each agent is represented internally in the XML format using an Agent Description Schema (ADS). The Development Environment provides a GUI tool, which allows the agent contributor to register new agents by filling out a form. The tool automatically encodes the information provided into the XML description.

According to the ADS, each agent has a unique numerical *agent ID* for retrieval purpose. Three other high-level attributes include the *functionality*, *media category* and the *signature interface*. The *functionality* stands for the operation performed by the agent, such as classification (assign media data into predefined categories) and event detection (find certain events in video or audio segments). The *media category* illustrates the general aspect of media the agent deals with, e.g., video, audio or image. In addition to the above features, the ADS contains the following elements: the *agent location* is the path and filename of the executable file corresponding to the agent. The

remark attribute provides a free-text annotation of the agent that allows the user to learn about the agent’s functionality. The *media format* attribute indicates the format of multimedia that the agent can handle, e.g. MPEG, BMP, WAV.

The agent interface specification consists of two levels, namely the syntactic interface and the signature interface. The former addresses the lower-level characteristics while the latter represents relatively higher-level features of the agents.

The syntactic interface requires each agent to be an application that can be invoked through a command line, e.g., a console executable program. Any programming language can be used for developing an agent. The system allows also using any interpretive language for agent development, but installation of the interpreting program should be done separately. An illustration of the Syntactic Interface of a typical agent is shown in Figure 4.

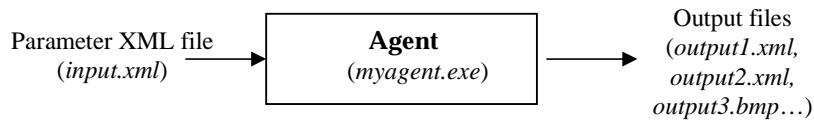


Fig. 4. The Syntactic Interface of an agent

Seen at the signature level, an agent in the Community is a black box that either takes the raw data of the media directly or the processing results produced by other agents as input, and generates its own processing results that can be used by the other agents. As shown in Figure 5, the signature interface of an agent contains three visible parts, namely Input Pins, Output Pins and Tuners.

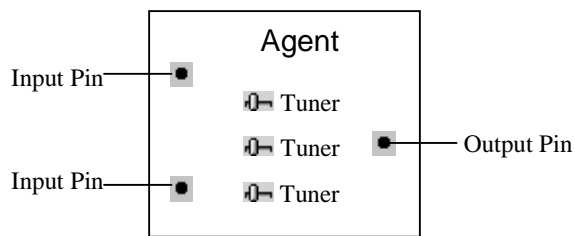


Fig. 5. Signature Interface of an Agent

An agent must have one or more input pins and output pins for data flow. There are different types of pins depending on the natures of the data. For example, if an agent performs face detection on MPEG video, it has one input pin of type “MPEG” and an output pin of type “Object Information”. Each pin belongs to a certain type, and pins of the same type are considered to be compatible with each other. Table 2 gives an exemplary list of the pin types. Users can build multi-agent systems by connecting the input pin of one agent to a compatible output pin of another agent. Thus the agents can

collaboratively process the media content by sharing data and perform more complex tasks that any single agent cannot accomplish. For example, a license plate detection agent combined with an OCR agent can be used to help the police identify stolen vehicles. We created templates for the pin types using the description schemes and descriptors in MPEG-7 standard so that agents developed by different researchers can communicate with each other as long as their outputs conform to the templates. For example, when the foregoing license plate detection agent is applied to a video clip on a frame-by-frame basis, the result is represented as a “VideoSegment” description scheme in MPEG-7, while each detected face corresponds to a “StillRegion” description scheme. The agent developer may first decide on the types of the pins of the agent, and then select the appropriate encode/decode tools.

Tuners are used for adjusting technical settings of agents to allow their flexibility. Such settings could include threshold values, operation types or any other configurations. For example, in a speaker gender identification agent, a numerical value can be used to favor the decision toward male or female. An agent may include zero or more tuners. Each tuner has a default value recommended by the inventor of the agent to ensure good performance in general cases, while the users can change it to meet their particular needs. For example, when a researcher designs an agent that detects stop signs on the road for driving assistance, he may prefer to have a balanced recall (the probability of a stop sign to be detected) and precision (the probability of a detected stop sign to be indeed a stop sign), while in practice it is usually desirable to detect as many sign as possible, even though at the cost of producing more false alarms.

Table 2. Examples of Pin Types

Pin Type	Explanation
Raw Data	The original media data stream, e.g., BMP image, MPEG video clip, MP3 audio clip
Object Information	Visual objects in video frames or images, e.g., signs, cars, human faces, animals, etc.
Event Information	Certain visible events that take place during a period of time or at some instant, e.g., car accident, scene cut
Label	Predefined class labels, e.g., movie ratings: G, PG, PG-13 or R, audio classification: music, speech, noise
Feature	Low-level features that can be represent by numerical scalars or vectors, e.g., color histogram (images), fundamental frequency
Text	Free text, e.g., the results generated by closed caption capture agents, OCR agent or speech recognition agents

7 Agent Development

To facilitate agents development the following methodology has been proposed. We assume that a researcher already has an algorithm and the problem is to convert it into an agent. If the algorithm is implemented as an executable program, then the researcher has to create a “wrapper” program that prepares data for the processing program, calls the processing program and converts resulting files into MPEG-7 annotation files. If the algorithm is implemented as a procedure then the researcher has to create an executable program that reads input data, calls the procedure and generates results. In any case an agent consists of the following blocks (Figure 6):

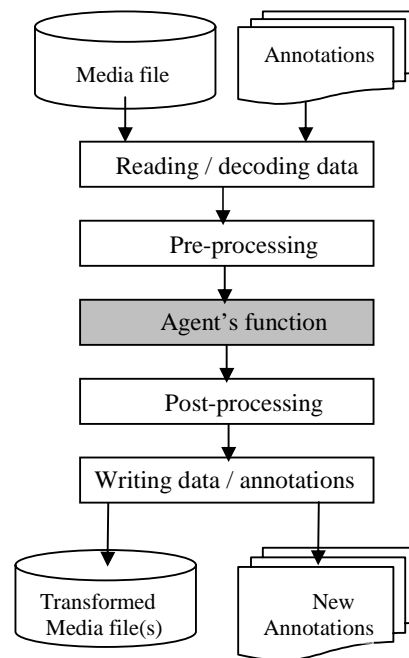


Fig. 6. Agent's structure.

- Reading and decoding raw data (signal) and annotations (features). The complexity of this step depends on the complexity of data coding method, for example, such coding standards as MPEG-1, MPEG-4 or JPEG are rather sophisticated.
- Do pre-processing, such as filtering, normalizing or pre-emphasizing the signal, or re-arranging annotations.
- Do the main agent's function, such as face detection in image or gender identification in audio stream.

- Do post-processing, such as calculate feature statistics that is required by the MPEG-7 standard.
- Write transformed media data and/or annotation files.

The Community provides tools that help the researcher to accomplish all data input/output and pre- and post-processing tasks and allows the researcher focusing on the main function of the agent, which is depicted by a shaded box on Figure 6.

8 Summary and Future Work

The Community of Multimedia Agents is a community of researchers and an open environment that allows researchers to share their achievements in multimedia annotation field while protecting their intellectual property. The Community provide the following components:

- The Agent Library gives researchers access to the variety of tools to handle the complexity of multimedia data and to develop new more efficient tools.
- The Development Environment facilitates the development and rapid prototyping of multimedia analysis systems.
- The open set of MPEG-7 templates allows unifying interfaces between agents and developing a hierarchy of concepts for creating ontologies in the field.
- The Community Web site provides the Community's resources and serves as a collaborative working and learning environment that consolidates efforts and expedites research and education in multimedia analysis field.

The Community's short-term objective is to create the "critical mass" of agents and a set of reliable tools to be useful for researchers and students. We are currently collaborating with several research labs and Universities to develop more agents.

The Community's long-term objective is to develop more sophisticated agents and tools that can process media streams rather than media files in real-time, generate sequences of agents to solve a formally specified problem, and find agents on the Web that can solve a problem.

9 Acknowledgements

The authors are thankful to Dr. Ishwar K. Sethi, and researchers of the Intelligent Information Engineering Laboratory at the Oakland University, Rochester, MI for their contribution to the core version of the Community's Agent Library.

References

1. Petrushin, V.A. Emotion Recognition in Speech Signal: Experimental Study, Development, and Application, *Proc. 6th International Conference on Spoken Language Processing (ICSLP 2000)*, Beijing, 2000. Vol. IV, pp 222-228.
2. M.T. Maybury (Ed.) *Intelligent Multimedia Information Retrieval*, AAAI Press/MIT Press, Menlo Park, CA / Cambridge, MA, 1997.
3. Ibrahimov, O.V., Sethi, I.K. and Dimitrova, N.. Clustering of Imperfect Transcripts using a Novel Similarity Measure, In A.R. Coden, E.W. Brown and S. Srinivasan (Eds.), *Information Retrieval: Techniques for Speech Applications, LNCS*, vol. **2273**, Springer-Verlag, 2002, pp. 23-35.
4. Dimitrova, N., Agnihotri, L. and Wei, G., Video Classification using Object Tracking, *International Journal of Image and Graphics*, **1**, No. 3 (2001), pp. 487-505.
5. Yao Wang, Zhu Liu, and Jin-Cheng Huang, Multimedia Content Analysis Using both Audio and Video Clues, *IEEE Signal Processing Magazine*, **17**, No 6, November 2000, pp. 12-36.
6. Huhns, M.N. and Singh, M.P., Agents and Multiagent Systems: Themes, Approaches, and Challenges, In M.N. Huhns and M.P. Singh (Eds.), *Readings in Agents*, Morgan Kaufman, San Francisco, CA, 1998.
7. Hauptmann, A.J. and Witbrock, M.J., InforMedia: News-on-Demand Multimedia Information Acquisition and Retrieval, In M.T. Maybury (Ed.) *Intelligent Multimedia Information Retrieval*, AAAI Press/MIT Press, Menlo Park, CA / Cambridge, MA, 1997, pp. 215-239.
8. Merialdo, B. and Dubois, F., "An Agent-based Architecture for Content-Based Multimedia Browsing", In M.T. Maybury (Ed.) *Intelligent Multimedia Information Retrieval*, AAAI Press/MIT Press, Menlo Park, CA / Cambridge, MA, 1997, pp. 281-294.
9. Martínez, José M., Overview of the MPEG-7 Standard, <http://mpeg.telecomitalia.com/standards/mpeg-7/mpeg-7.htm>
10. W3C Candidate Recommendation, *Resources Description Framework (RDF) Schema Specification 1.0*, March 2001.
11. W3C Notes, *DAML+OIL (March 2001) Reference Description*, March 2001.
12. Heflin, J. and Hendler, J., A Portrait of the Semantic Web in Action, *IEEE Intelligent Systems*, vol. 16, No. 2, pp. 54-59, March/April 2001.
13. Hansen, L. K. and Salomon, P. Neural network ensembles, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1990.
14. M. Li. *MPEG-7 Audio Analysis Agents for Community of Multimedia Agents*, Technical Report, Accenture Technology Labs, August 2002.