Automatic Video Editing for Multimodal Meetings

Radek Kubicek, Pavel Zak, Pavel Zemcik, Adam Herout

Graph@FIT, Brno University of Technology, Bozetechova 2, Brno, CZ
{ikubicek, izakpa, zemcik, herout}@fit.vutbr.cz

Abstract. Meeting recording is being performed through microphones and video cameras in order to keep a permanent record of the events that are happening during the meetings. The technology to perform such recording is already mature and recording is being performed already for some time. However, efficient retrieval of the information from meeting data remains a hot topic of contemporary research. Several approaches to information retrieval exist, such as indexing the data, event semantics analysis of the data, etc. This contribution focuses on automatic video editing of the data in order to prepare audiovisual material, based on several audio and video sources, that is suitable for human users to see. The video editing takes the original audio and video data as its input as well as the results of analysis of audio and video streams and user instructions. The output of the method is a simple audiovisual stream.

1 Introduction

The technology for recording audio-visual data of meetings (such as small informal technical meetings, meetings of administration boards, conference meetings, etc.) is available and is being used in practice. However, the task of information retrieval from such collected data is an open field for research. A large European project Multi-Modal Meeting Manager [6] (MMMM, M4) has been targeting collection and processing of meeting data, both the audio and video components. The project's goals relevant for this paper included collection of well defined meeting data, annotation of the data, and research of methods to extract various events, occurrence of people and objects, and description of the processes included in the audio and video data. The records of the meetings were obtained through several synchronized video cameras and a set of microphones. Building upon the M4 project, the Augmented Multi-party Interaction project [4] (AMI) targeted interaction of the participants of meetings during the meetings. The newest, currently running Augmented Multi-party Interaction project with Distant Access [5] (AMIDA) project addes the aspect of "remote meetings" where the participants of the meeting interact with the others remotely, In all the three consequent projects, automatic video editing based on processing of the data extracted from the audio/video streams is important. The main application areas for the video editing within the above mentioned projects include:

1. "Off-line" preparation of meeting summaries in the form of single video sequence. Such sequences may include only single audio/video stream based on "intelligent" processing of several synchronous video streams and if necessary, may be shortened (in time) based on the semantic analysis of the input data and/or user requirements.
2. "On-line" (real-time) video editing in the form of continuous production of a single audio/video stream based on the input. Such audio/video stream then can be shown e.g. on a large screen display to the meeting participants to enhance their perception of the meeting.
3. "On-line" (real-time) video editing in the streaming form through Internet can also be used to transfer the audio/video information from one meeting site to another one in order to "keep in touch" the remote meeting participants.

This paper deals with the algorithms of automatic video editing that have been developed within the mentioned research projects. Section 2 summarizes the image processing and vision techniques used by the video editing engine. The automatic video editing algorithms and the proposed final solution are described in Section 3. Section 4 defines the data types and record structuring used by the central database storage used for collecting the data from different sources - both the original recording equipment and different processing and data-retrieval functional units. Section 5 summarizes the paper and suggests issues for future research.

2 Image Processing for Automatic Video Editing

The image processing methods used in automatic video editing include mostly the state of the art methods adjusted for the exploitation in meeting environments [1] [2] [7] [8]. The output of such methods needed for automatic audio/video editing includes mainly:

- presence and positioning of humans and their body parts in the video,
- presence of objects in video,
- identification of a speaker or localization of the speaker in image space,
- presence of events in video (related and non-related to human activities),
- facial expression and gesture recognition,
- tracking methods.

Methods used in the detection of presence of humans in the image and to identify the position and orientation of their body parts includes several known techniques, such as pre-processing of the image to find the skin color spots, identification of the spots (face and hand detection methods are implemented), recognition of faces through Ada-Boost based machine learning methods, and measurement of the face gaze through Gabor Wavelet Networks.

The methods of detection of the presence of objects in the video are mostly based on machine learning approach but for several specific objects, such as whiteboard and/or projection wall, specific methods are proposed.

Speaker localization is mostly done through audio signal processing (this might be done either through precise localization through microphone arrays or through processing of few microphone inputs with less precision – depending on the meeting recording equipment setup). Experiments also have been done with combination of audio and video processing through lips image detection and processing [9].

The presence of events in video are mostly recognized through detectors based on statistical processing of the scene and also on processing of lower level features through artificial intelligence methods. In fact, event detection is not only task of video processing but also of the meeting understanding research that is within the scope of the above mentioned projects but well out of the scope of this paper.

The facial expression and gesture recognition methods used in the video editing are based on machine learning approaches, such as Ada-Boost based methods for image pbject recognition and Hidden Markov Models (HMMs) for scene dynamics analysis. Tracking methods used within the project are mostly based on particle tracking approaches, where the particles are mostly seen as human face, human hand, or human torso outline. The tracking methods are needed especially to cover the gaps that occur in the video processing if single frame processing does not lead in successful recognition and understanding the scene.

3 Automatic Video Editing

The approach to automatic video editing we used is based on the following facts:

1. The video presented to the humans should be well perceived by the humans from the point of view of dynamics of the shots. A large set of rules exists for this purpose [2] [7] [8] [9].
2. The video should contain the relevant actions and objects that occur in the video. This aspect is largely dependent on the application and for the meeting data the relevant objects are humans and certain objects and actions known in advance.
3. In some cases, the video editing could be affected by the user requirements, such as "Show the activities of Mr. X in the meeting".

As the requirements can be changing over the time and can be also based on the user modifications, traditional procedural languages would not behave too well. At the same time, machine learning is not too suitable in this case, as the human perception of the video sequences are well known and well described (from the traditional film making approaches).

For the above reasons, non-procedural language Prolog was used as the implementation base of the editing decisions. The input of the editing process is a set of facts - results of audio/video stream processing, set of rules - the predefined general video editing rules and a dynamically changeable set of rules based on the details of the audio/video processing, user requirements, etc. The

output of the video editing decision process is a text description of the editing decisions that is afterwards interpreted in a video editing engine that actually processes the audio/video data. The basic scheme of the process is shown in Figure 1 below.

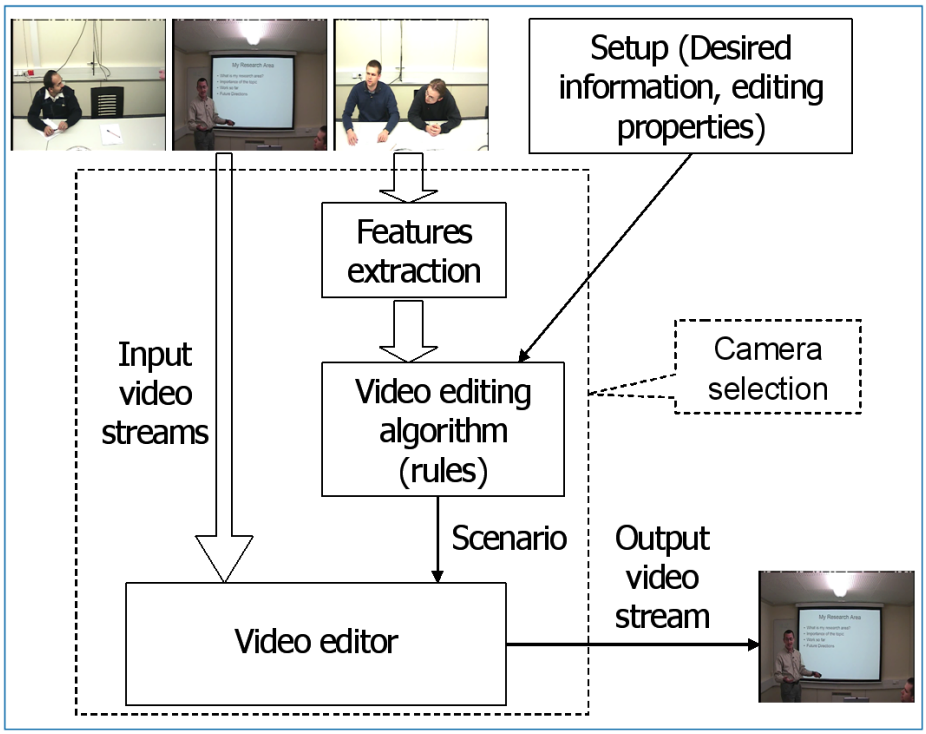

Fig. 1. The basic video editing scheme.

The main idea of building the rules for selection of views into the output video stream is that each camera should be given its "weight" that represents the importance of the activity shown by the camera. Each person, if known, should also be given the "weight" with similar meaning, and finally, a decision of what is going to the output is done using these weights and the human perception of video rules (that prevent e.g. too long and too shorts shots or too rapid changing of the shots). Additionally, virtual cameras can be defined as rectangles within the existing camera images. See Figure 2 and Figure 3 for the editing rules.

The example of the output of video editing is shown in Figure 4 below (along with the three input video streams).

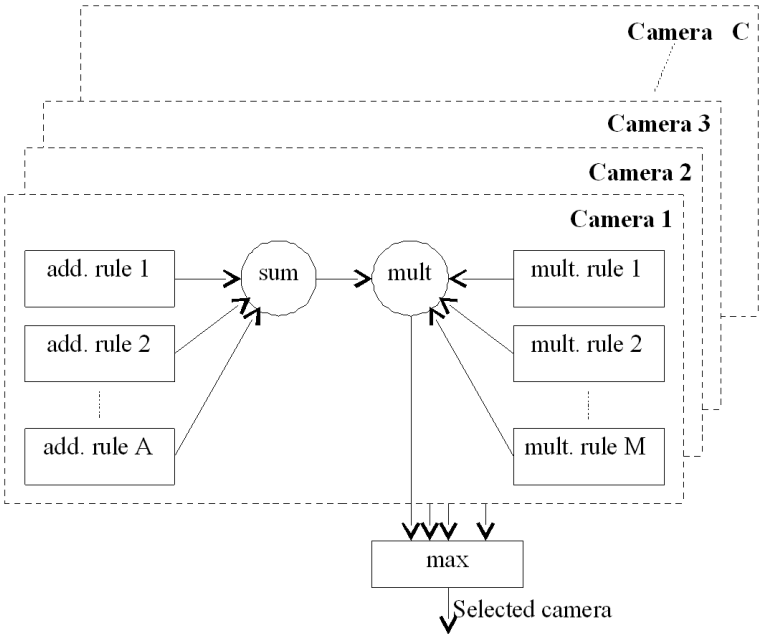

Fig. 2. The weight definition video rules for cameras.

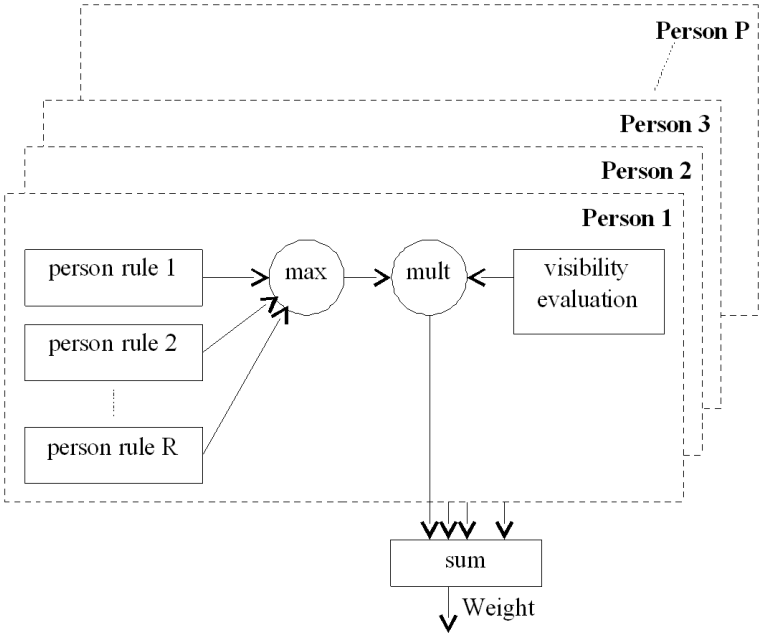

Fig. 3. The weight definition video rules for meeting participants (persons).

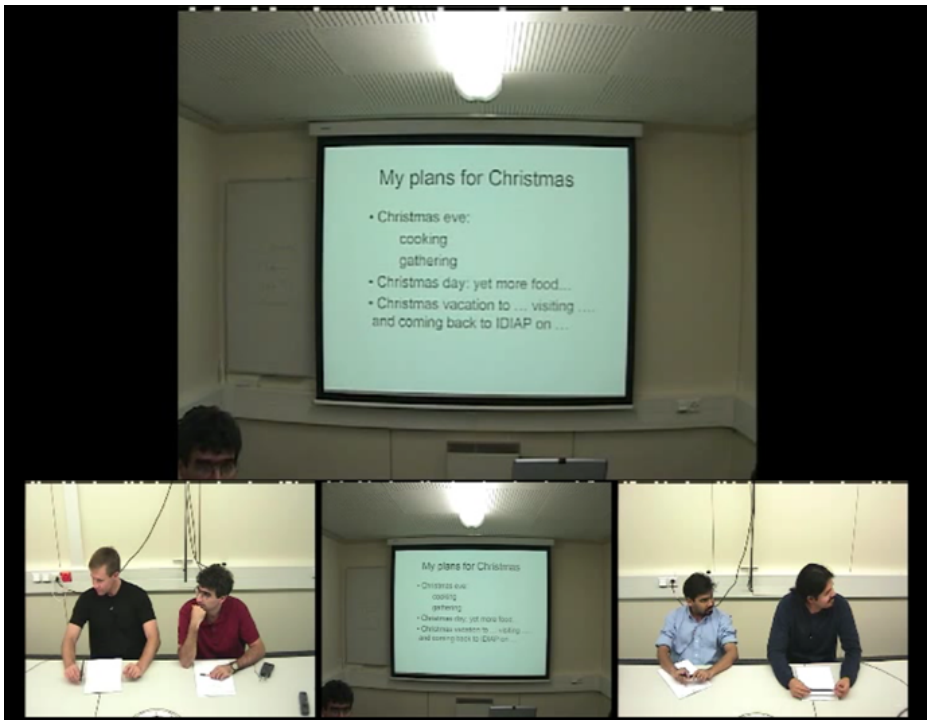

Fig. 4. Example of the output of the video editing algorithm.

4 Remote Access to the Audio/Video Data HUB

The input of the audio-video editing system is constituted by the multimedia audio-visual data themselves with metadata describing their properties and structure. As an input serve also events appearing in the data. These events – that are potentially somehow important for the video editing – are either manually annotated by a tool or are extracted automatically. In case of manual annotation, the data are collected before the editing process starts, the automatic event extraction can happen during the video processing, whose output is the edited video sequence. All the manually annotated and automatically extracted events need to be stored for later use and for the editing process.

To allow this, a component serving as the data storage of the annotations has been defined, that allows a unified way of storing and retrieving the data. This complete infrastructure is referred to as the Hub.

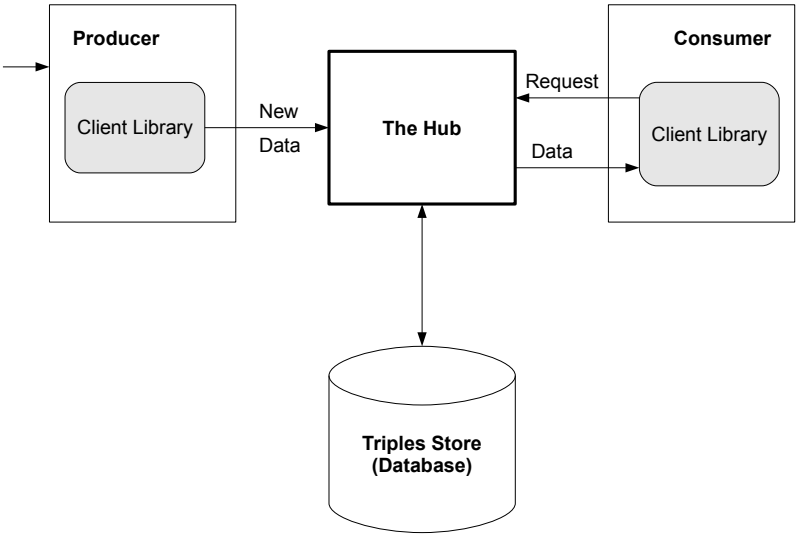

Fig. 5. Producer-consumer scheme of Hub.

The Hub is intended to provide all of the storage that a group or a company needs for annotations about their archived meetings in one place. Each such Hub contains a database, which keeps the stored annotations and serves requests to the data. Along with the annotation/extracted data it contains all related documents, presentation slides and notes shown during the meeting. The Hub is based on the producer-consumer model, see Figure 5, where the producer sends

information to the Hub, which transforms them to its internal data format and stores into the database. The client consuming the data sends a request to the Hub, which processes it and if data matching the request are found, they are sent to the client. Such requests can be specified either to match an exact data entry or regular expressions, time intervals etc. can be used. At the same moment when one event-extracting process inserts data to the Hub, the consuming application can retrieve it. This allows the Hub to be used both in real-time on a pending meeting to provide a newly connected client with all available information about the meeting as well as to post-processing the meeting data and production of summaries and similar material.

The data in the Hub are stored as timed-triples. The triple contains entries: *object*, *attribute* and *value*, and is accompanied by a *timestamp*. But the application for automatic video editing (VideoEditor) uses a XML file as the source, which contains relevant information about the input data, annotated and detected events. A communication protocol was defined to interface the Hub with this application. It defines the format of separate tags stored in the triplets in the Hub and it handles the transformation of data read from the XML file into the corresponding timed triplets and the following communication with the Hub by the means of requests. These requests are processed by a XML parser, which can construct the request and parse the corresponding response to the relevant data entries. This parser can also handle an invalid XML, which for example does not contain the termination tags or does not meet the defined DTD.

A comparison of the XML structure with the modified communication protocol for the Hub is in the following listing. `[MID]` denotes a unique identifier of the meeting, `[N]` is represented by all indices from 0 to *Count* of the corresponding section and `[...]` denotes all different parameters of the respective section. The output data marking the places for the shot boundary are stored with identical names, only instead of `[MID]` they use `[MID-out]`.

4.1 Data Path from the Event Producer to the Editing Process

Either during the recording or after it has been finished, the video data are sent to the annotation software and to the automatic event extractors. These processes extract relevant events and send them to the Hub. Such data are requested by the VideoEditor as a bulk. The block of data replied from the Hub is parsed from the XML into the timed-triples in the internal format of the application. Complete data path of the video editing process is shown in the Figure 7.

The actions of the automatic video editing process can be described by the following pseudo-code.

```
1  get all appropriate data from hub
2  if received Data
3    parse Data into TimedTriples
4  else
5    end application
```

```
<?xml version="1.0"?>
<AVEvents>
 <EventGroups>
  <Group>
   <ID>0</ID><Name>individual_positions</Name>
   <Meaning>State</Meaning><Enabled>1</Enabled>
  </Group>
  <Group>
   <ID>1</ID><Name>Speaking</Name>
   <Meaning>State</Meaning><Enabled>1</Enabled>
  </Group>
 </EventGroups>
 <EventTypes>
  <Type>
   <ID>0</ID><Name>off_camera</Name><Offset>0</Offset>
   <Parameters individual="PM"/>
   <Secondary>
    <Key></Key><Offset>0</Offset><Parameters individual="ID"/>
   </Secondary>
   ...
  </Type>
  ...
  <Type>
   <ID>8</ID><Name>Start speaking</Name><Offset>0</Offset>
   <Group>1</Group><GroupIndex>0</GroupIndex>
  </Type>
 </EventTypes>
 <File>
  <Source Camera="3">video\ES2003a.Corner_orig.avi</Source>
  ...
  <Source>audio\ES2003a.Mix-Headset.wav</Source>
  <TimeFormat>Milliseconds</TimeFormat>
  ...
  <Event>
   <ID>8</ID><Time>1110415</Time>
   <Text>Yeah? Okay.</Text>
   <Parameters Person="Closeup4"/>
  </Event>
  ...
 </File>
</AVEvents>
```

Fig. 6. Example structure of XML input file for video editing application.

object	attribute
[MID].EventGroups.Group	Count
[MID].EventGroups.Group.[N]	[...]
[MID].EventTypes.Type	Count
[MID].EventTypes.Type.[N]	[...]
[MID].EventTypes.Type.[N]	Parameters.Count
[MID].EventTypes.Type.[N]	Parameters.[N].[...]
[MID].EventTypes.Type.[N]	SecondaryKeys.Count
[MID].EventTypes.Type.[N]	SecondaryKeys.[N].[...]
[MID].EventTypes.Type.[N]	SecondaryKeys.[N].Parameters.Count
[MID].EventTypes.Type.[N]	SecondaryKeys.[N].Parameter.[...]
[MID].Sources.Source	Count
[MID].Sources	TimeFormat
[MID].Sources.Source.[N]	[...]
[MID].Events.Event	Count
[MID].Events.Event.[N]	[...]
[MID].Events.Event.[N]	Parameters.Count
[MID].Events.Event.[N]	Parameter.[N].[...]

Table 1. Rules for saving XML events into Hub timed-triples.

```
 6  for each InternalStructure find RelevantData
 7    if RelevantData in TimedTriples
 8      fill InternalStructure with RelevantData
 9    else
10      process next InternalStructure

11  run video editing process
```

After the editing process is finished, the resulting set of events – marking the shot boundaries – is sent back to the Hub to be used for later editing or for comparison/evaluation with other means of editing.

A streaming media server is being developed that will allow automatic editing on a running meeting. At present state, the data needs to be stored in a media file in the file system with the editing tool. When such streamed media is available, the Hub's potential will be fully utilized.

5 Conclusions and Future Work

This paper presented an approach to automatic video editing, which builds upon a central data-collection system which collects data from different sources - both from the original audio/video streams from the meetings and from different metadata and extracted data as retrieved by a number of data processing units.

The described system proved to be functional in a demonstration sample. However, interconnection with more data retrieval units will be sought in the

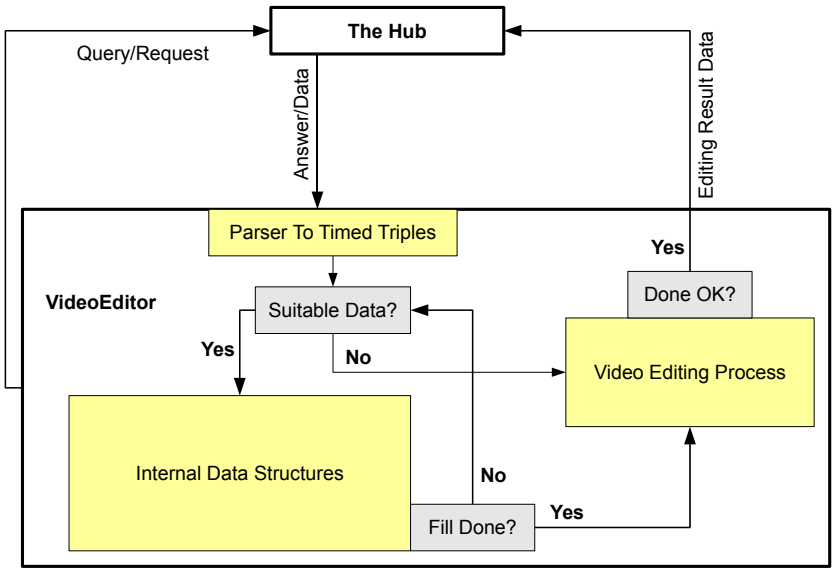

Fig. 7. Data path of video editing process.

future, to further extend the possibilities of the automatic video editing algorithms. The system would also benefit from an extended set of video-editing rules, which will be defined based on real usage on actual data acquired from model and real meetings.

The results of the AMI and AMIDA projects can be found on the web ([5]) and are available for further use.

Acknowledgements

This work has been supported by the research grant project GA AVCR 1ET400750408 "Rapid prototyping tools for development of HW-accelerated embedded image- and video-processing applications.", by the project EU-6FP-IST, IST-033812-AMIDA, and by the research project "Security-Oriented Research in Informational Technology" CEZMSMT, MSM0021630528.

References

1. Ashby, S. et al.: *The AMI Meeting Corpus*, In: Measuring Behavior 2005 Proceedings Book, Wageningen, NL, 2005
2. Kadlec, J., Potucek, I., Sumec, S. Zemcik, P.: *Evaluation of Tracking and Recognition Methods*, In: Proceedings of the 11th conference EEICT, Brno, CZ, 2005, ISBN 80-214-2890-2

3. Porter, S., Mirmehdi, M., Thomas, B.: *Temporal video segmentation and classification of edit effects*, Image and Vision Computing, Volume 21, Issues 13-14, 2003
4. *Augmented Multi-party Interaction project*, http://www.amiproject.org/ (available as of 2008-07)
5. *Augmented Multi-party Interaction with Distant Access*, http://www.amidaproject.org/ (available as of 2008-07)
6. *Multi-Modal Meeting Manager project,* http://www.m4project.org/ (available as of 2008-07)
7. Chambela, T. et al.: *Creating Video Art with Evolutionary Algorithms*, In: Computers & Graphics, Volume 31, Issue 6, December 2007
8. Bocconi, S., Nack, F., Hardman, L.: *Automatic Generation of Matter-of-Opinion Video Documentaries*, Web Semantics: Science, Services and Agents on the World Wide Web, Volume 6, Issue 2, April 2008
9. Sumec, S., Kadlec, J.: *Event Editor - The Multi-Modal Annotation Tool*, In: Workshop on Multimodal Interaction and Related Machine Learning Algorithms (MLMI), Edinburgh, GB, 2005