Towards semi-automatic annotations for video and audio corpora

S. Masneri[1], O. Schreer[1], D. Schneider[2], S. Tschöpel[2], R. Bardeli[2], S. Bordag[3], E. Auer[4], H. Sloetjes[4] & P. Wittenburg[4]

[1]Fraunhofer Heinrich-Hertz-Institute, Berlin, Germany
[2]Fraunhofer-Institut für Intelligente Analyse- und Informationssysteme IAIS, Sankt Augustin, Germany
[3]Max Planck Institute for Social Anthropology, Halle, Germany
[4]Max Planck Institute for Psycholinguistics, Nijmegen, the Netherlands
E-mail: [stefano.masneri,oliver.schreer]@hhi.fraunhofer.de,
[daniel.schneider,rolf.bardeli,sebastian.tschöpel]@iais.fraunhofer.de, bordag@eth.mpg.de,
[eric.auer,han.sloetjes,peter.wittenburg]@mpi.nl

Abstract

AVATecH (Advancing Video/Audio Technology in Humanities Research) is a project in which two Fraunhofer Institutes and two Max Planck Institutes collaborate in order to promote the development and application of technology for semi-automatic annotation of digital audio and video recordings. One of the aims of the AVATecH project is to implement algorithms that allow for the automatic or semi-automatic creation of pre-annotations for the video corpora, hence reducing the time needed to perform the manual annotation task. Due to the huge size of the corpora, and the extreme variety of the video content, the algorithms developed need to be fast, efficient and robust. In this paper we will present some of the algorithms currently under development, the modifications applied in order to get them working with large video corpora and how the results of the annotations are stored, as well as how they can be integrated in ELAN annotation software.

1. Problem definition

In humanities research for psycho-linguistics, very large video corpora are available in order to investigate many different kinds of research topics such as relation between spoken language and gestures or preserving endangered languages. To evaluate the huge amount of video in the most efficient manner, meaningful annotations for the entire collection are required. These annotations should be of sufficient quality to both enable the user to gain an overview of the contents, as well as select and access important parts of the content quickly. One of the aims of the AVATecH project is to implement algorithms that allow for the automatic or semi-automatic creation of pre-annotations for the video corpora, hence reducing the time needed to perform the manual annotation task.
Although the use of video analysis tools for automatic annotation has been a research field for many years, two aspects are different to common approaches. Due to the huge size of the corpora (approx. 30 TB), the algorithms need to be fast and efficient. Another important challenge is the diversity of the content. Though the most common case is that of persons being captured, almost any scenario can happen. Hence, very general, but also robust approaches need to be developed in order for the algorithms to be actually helpful in retrieving information out of the videos. A careful evaluation of robustness versus analysis quality needs to be taken into account. The almost arbitrary nature of the content does not allow an application of standard methods developed in the field of sign language recognition.

2. Video analysis algorithms

During the design of the algorithms for video analysis the focus was mainly on the efficiency and the robustness of the solution. Efficient algorithms allow for faster automatic annotation, while robustness guarantees that meaningful annotations can be achieved for the majority of the content in the corpora. The design of algorithms that can perform well in many different scenarios (in the subset of corpora used for testing there are videos shoot in interview rooms, in restaurant, in small villages, in conference rooms, each of them with a different number of people represented) is the main guideline used to adapt existing solutions to the specific problem and to develop completely new approaches to the problem. The algorithms are designed to work in a fully automatic way, i.e. without the need for human interaction, in order to guarantee the possibility to use scripts to run the program on multiple files. The implementation is done using a highly modular structure, so that future algorithms can be seamlessly integrated in the current framework, using the results provided by the previous detectors.

2.1 Shot boundary detection

Shots consist of the video frames that have been continuously recorded with a single camera operation, and therefore represent the basic unit of a video. Since different shots refer to different camera operations, all of the detectors work on a shot basis. The tool developed in (Petersohn, 2004) was used as shot boundary detector, with few changes to the I/O interface. The shot boundary detector also retrieves the position of sub-shots inside of a shot. Sub-shots are defined as a sequence of consecutive frames showing one event or part thereof taken by a single camera act in one setting with only a small change in visual content. The detection of sub-shots has proved to be useful for the development of the other detectors.

A wrapper was added to the shot boundary detector, in order to make it work with all the types of video format used in the corpora and to provide a uniform, human readable and easy to parse XML output. Since all of the video in the corpora are unedited (and therefore there are neither fades nor wipes), only hard cuts are detected in order to improve the efficiency of the algorithm. The program runs faster than real-time, processing about 80 frames per second on standard definition videos.

2.2 Key-frames extraction

Because of the huge amount of data, it is extremely important to provide the user the possibility to efficiently browse the content of the videos. Since watching each video in the corpora would require hundreds of hours, the best option is to select an adequate number of images to represent all the content of a video.
The simplest (and somewhat obvious) approach is to extract an image every *n* frames, but it can lead to miss some important information or, on the other hand, extracting lots of almost identical images, when the scene is static and few changes happen. The new approach is to use the results of the shot-cut detection algorithm, extracting an image every time a new sub-shot is detected.

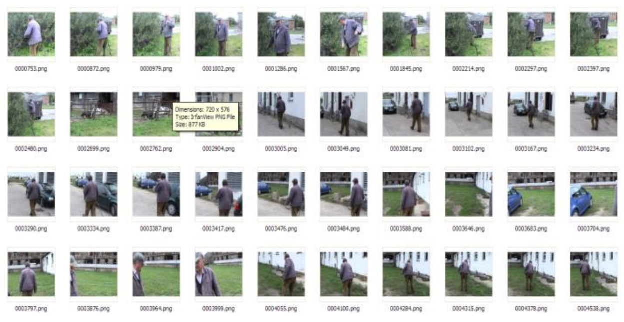
Figure 1: Results of key-frames extraction tool.

An additional image is extracted at the midpoint of a shot, to ensure that every shot in the video is represented by at least one image. With this approach, all the relevant information in a video is captured. Various options allow the user to decide the quality and the size of the output images. A typical use case (see **Figure 1** for an example) is to save the images as compressed thumbnails, allowing the user to grasp the content of the whole video with just a few glimpses.
The execution is usually five to ten times faster than real-time.

2.3 Global motion detection

The motion inside of a shot is another useful feature that can provide plenty of information to the user. An accurate motion analysis allows distinguishing between different types of video content. For example, an interview will have a static camera and a low amount of total movement inside the scene, while the video of a carnival will have lots of pans (i.e. camera motion) and motion inside of the scene. Further than that, an accurate motion analysis can provide helpful information for many other detectors.

The algorithm performs then a frame-based motion analysis and detects when global motion (pan or tilt) occurs inside of a shot. Our work is based on the *Hybrid Recursive Matching* algorithm (Atzpadin, Kauff & Schreer, 2004). For each frame in the shot, it extracts a motion map, representing the motion of a grid of pixels inside of the frame. The absolute value (i.e. the speed, calculated as L^2 norm) and the orientation are then computed, in order to obtain a vector field representing the total motion for that particular frame. An analysis of the motion map allows then to distinguish between camera motion and motion inside of the scene. This is particularly useful because in this way other than detecting camera motion, there is also the possibility to compensate motion when, as done in different detectors, moving objects inside of the scene needs to be tracked.
Once the motion for each frame in the shot has been analyzed, a post-processing step is performed, in order to reduce the fragmentation of the results. This step is necessary because the algorithm detects all kind of camera movements (even very short ones, e.g. when the person shooting the videos is adjusting the camera), while most of the time the user is interested only in longer camera movements. There is a set of parameters that can be tweaked by the user so that he can choose whether to have a very detailed but fragmented motion analysis or a coarser but less fragmented one. At the end of the algorithm a list of the camera motions occurring in each shot is obtained, with initial and final frame, as well as the direction of motion.
Working with standard definition video and taking motion vectors every 8x8 pixel, the program is able to process about 30 frames per second, slightly faster than real-time.

2.4 Skin colour estimation

In order to be able to successfully track objects inside of a scene the motion detector alone is not enough. To describe the object of interest other information needs to be extracted from the video. Since the user is typically interested in gesture annotation, it has been decided to focus on the detection and tracking of hands and heads. In order to perform the detection the first thing to do is *skin colour estimation*, that is to find the colour and luminance ranges that best represent the human skin. This is not an easy task, since for different videos skin is represented with different colours (skin colours vary depending on the person filmed, the luminance condition, the quality of the camera and other factors). Hence, the need is to develop an algorithm that allows accurate skin colour estimation without *a-priori* information.
It was chosen to work with the YUV colour space, instead of the more common RGB, for two reasons: the first one is that, under equal conditions, the U and V component tend to be similar for all kind of people filmed (e.g. the skin colour component of a Caucasian person and that of an Asiatic person are similar). The second one is that the biggest amount of information is contained in the luminance component Y, allowing us to use sub-sampled version for the U and V colour components, hence

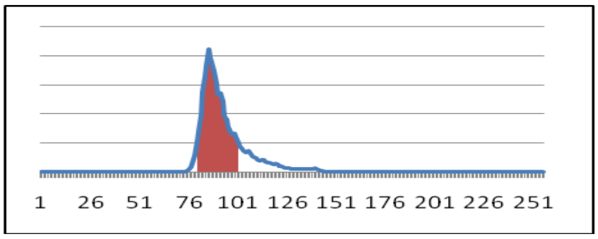

Figure 2: Histogram for U component. The range representing skin colour is highlighted

reducing the amount of memory needed and speeding up the computation.
The main goal of the skin colour estimation algorithm is then to narrow, for each one of the Y, U and V component, the range from [0...255] to the actual range corresponding to skin colour. An early, rough, estimation of the U and V ranges can be done by excluding the values that surely do not represent skin colour. Of course this narrowing operation is too approximate and must be further refined.
To refine the U and V skin parameter estimation the results of the motion detector were used. In fact, it is assumed that the motion inside of a scene is most of the time due to the movement of a person, and therefore by identifying the motion inside a scene the presence of skin can be automatically identified as well. Of course there can be other moving objects in a scene, other than hands and heads, but these objects most of the time can be discarded since they are outside the colour ranges fixed before at the beginning of the motion analysis.
By repeating this procedure for each frame in the shot it is possible to keep track of all the pixels in the shot that most likely represent skin. By representing all the pixel values in a histogram the desired colour range can then be identified with a high degree of precision. **Figure 2** shows the histogram of the U values of the selected pixel for a shot: the highlighted range shows the interval that represents the skin range for this colour component.
Once the estimation of the U and V colour components is done, a new processing step is performed in order to estimate the luminance component. The execution time, for standard definition video, is comparable to real-time.

2.5 Hands and head detection and tracking

Once the colour and luminance ranges representing skin have been estimated accurately, the detection and tracking of hands and heads in the video is performed. The first step of the detection process involves the search of *seed* points where the hands and heads regions most likely occur. **Figure 3** shows how the seed points are selected: Histograms along the horizontal and vertical directions compute the number of pixels with luminance and colour values within the desired interval; the pixels where a maximum occur in both the directions are selected as seed points.
Starting from the seed points a region growing algorithm is applied, that selects all the points in the neighbourhood within the colour and luminance ranges found in the previous step. For each region found a different label is applied, allowing us to track the movement of different regions along the timeline. Each region found is then approximated with an ellipse, characterized by the position of the centre, the orientation and the length of the axes.
In **Figure 4** the result of the detection is shown: the pixels marked as skin are highlighted, and the ellipses approximating the region found are shown.
To discriminate between hands and heads a face detection algorithm (Viola & Jones, 2001) is applied and some basic geometric considerations (position of the heads, size of the region) are made in order to increase the robustness of the system.
The tracking is performed by considering the change of orientation and position of the ellipses along the timeline.

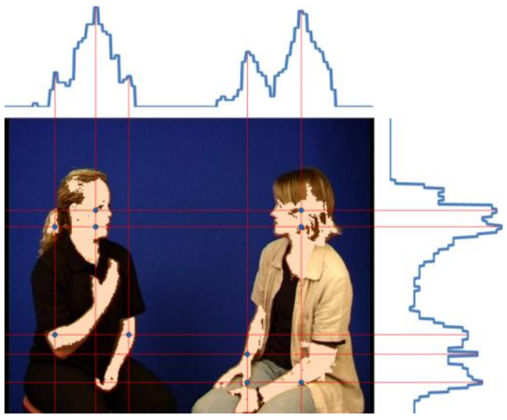
Figure 3: Detection of seed points

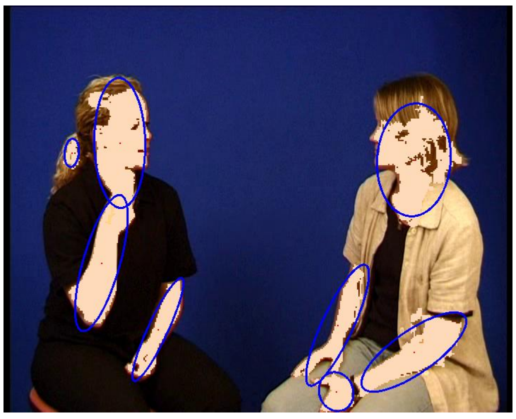
Figure 4: Hand and head tracking

The execution time depends upon the number of objects tracked, but is usually 3 to 5 times faster than real-time.

3. Annotation based on video analysis

Each one of the tools developed produces as output an xml file. The xml schema defining the syntax of the output files is the same for each program, in order to simplify the integration of the different tools and the addition of new ones.
For the shot boundary detection module the output xml contains a list of *shot* elements, each one of them has, as attributes, the start of the shot, the end of the shot and the position of the representative frame. Each shot element

can also have zero to many *subshot* children elements, specified by the relative representative frame. The subsequent annotation tools simply use the xml produced by the shot boundary detector as input, and add new children elements to each shot. The global motion detector adds *motion* elements, with attributes specifying the starting and ending frames, the type and the direction of motion. At last, the skin colour estimation tool adds one *segmentation parameters* element for each shot, with six attributes that specify the estimated intervals for the Y, U and V components.
The results of the hand and head tracking tool are not yet propagated to the output xml file, since various output options are currently being examined and no definitive choice has been made.
Below an example of a typical *shot* element, with all its child elements, is shown.

```
<shot start="0" end="501" keyFrame="250">
  <subshots>
    <additionalShot keyFrame="174"/>
    <additionalShot keyFrame="217"/>
    <additionalShot keyFrame="457"/>
  </subshots>
  <segmParams>124 133 134 81 10 16</segmParams>
  <motion>
    <cameraMotion direction="right" start="127" end="136"/>
    <cameraMotion direction="right" start="158" end="162"/>
    <cameraMotion direction="right" start="164" end="220"/>
  </motion>
</shot>
```

A *shot* element produced by the detectors

4. Integrating detectors in ELAN

The annotation software ELAN 3.6 introduced a Recognizer API for extending the program with pattern recognition components. The first implementation had limited functionality and only offered support for audio recognizers. The first attempts to integrate audio detectors proved to be particularly useful and generated a list of new wishes and requirements.
In the meantime, support for video recognizers has been added as well. To a large extend the interface for video recognizers follows the lines of that of audio recognizers, but the data structures for video differ slightly (no distinction between channels, provision for 2D area markers) and so does the user interface to interact with the recognizers. It is expected that more modifications of and extensions to the framework will prove necessary in the course of the project.
It needs to be noted that integration in ELAN is just one of the ways in which detectors will be made available; other solutions are developed as well. In the case of recognizers that are not deployed as extension of ELAN, the resulting XML can be imported into ELAN.

5. Future developments

Future developments point in two different directions. On one hand there is the possibility to improve the current detectors by increasing the richness of semantics. One example is offered by the hand and head detection and tracking tool, for which it could be explored the possibility to map the tracked movement (e.g. rotation or movement in a certain direction) into actual hand gestures, in order to add to the framework gesture-recognition capabilities too.
On the other hand there is the development of tools that can extract other features (for example, a tool for tracking of other objects inside of the scene, for the segmentation of the video to allow distinguishing between background and foreground, for the creation of summaries of the videos).

6. Conclusion

A new framework for automatic annotation was developed. The framework is designed to be as robust and efficient as possible. It implements algorithms for fast browsing of the video corpora, motion detection, hand and head detection and tracking.

7. References

Petersohn, C. (2004). *Fraunhofer HHI at TRECVID 2004: Shot Boundary Detection System*, TREC Video Retrieval Evaluation Online Proceedings, TRECVID.

Petersohn C. (2007). *Sub-shots - basic units of video*. In EURASIP Conference Focused on Speech and Image Processing, Multimedia Communications and Services, Maribor, Slovenia.

Boreczky J. & Rowe A. (1996). *Comparison of video shot boundary detection techniques*, Journal of Electronic Imaging, 5(2), 122-128.

Atzpadin, N., Kauff, N. & Schreer, O. (2004). *Stereo Analysis by Hybrid Recursive Matching for Real-Time Immersive Video Conferencing*, Trans. on Circuits and Systems for Video Technology, Special Issue on Immersive Telecommunications, Vol.14, No.3, pp. 321-334.

Askar, S., Kondratyuk, Y., Elazouzi, K., Kauff P. & Schreer, O. (2004). *Vision-Based Skin-Colour Segmentation of Moving Hands for Real-Time Applications*, Proc. of 1st European Conf. on Visual Media Production (CVMP 2004), London, United Kingdom.

Lausberg, H., & Sloetjes, H. (2009). *Coding gestural behavior with the NEUROGES-ELAN system*. Behavior Research Methods, Instruments, & Computers, 41(3), 841-84.

Crasborn, O., Sloetjes, H., Auer, E., & Wittenburg, P. (2006). *Combining video and numeric data in the analysis of sign languages with the ELAN annotation software*. In C. Vetoori (Ed.), Proceedings of the 2nd Workshop on the Representation and Processing of Sign languages: Lexicographic matters and didactic scenarios (pp. 82-87). Paris: ELRA.

Viola P. & Jones, M. (2001). *Robust real-time object detection*. Second International Workshop on Statistical and Computational Theories of Vision.