A VIDEO SEGMENTATION AND ANNOTATION TOOL FOR PARLIAMENTARY RECORDINGS AND TRANSCRIPTIONS

Luis M. de Campos, Juan M. Fernández-Luna, Juan M. García,
Francisco Gómez, Juan F. Huete, Carlos J. Martín-Dancausa
Department of Computer Science and Artificial Intelligence. E.T.S.I. Informáticas y de Telecomunicación
Universidad de Granada, C.P. 18071, Granada, Spain

ABSTRACT

The Parliament of Andalusia records all the parliamentary sessions as well as generates files with the exact transcription of the files. With these two types of media, a search engine, starting from a user's query, would return the relevant documents for that query, but also a link to the corresponding portion of the video where the speech is played. To achieve this goal, a previous tool to segment the videos and later to annotate or synchronize text and video must be developed. In this paper we describe the tool for segmentation and annotation, more specially the simple but effective and efficient segmentation algorithm developed exclusively for the special features of the videos from the Parliament of Andalusia.

KEYWORDS

Video segmentation, annotation, parliamentary sessions.

1. INTRODUCTION AND MOTIVATION

One of the main objectives of a democracy is that citizens know what their representatives in the parliament are dealing with in each moment. In this line, national and regional parliaments have to spread the works developed in these chambers of members of parliament in order to make public all the matters being discussed. So the Parliament of Andalusia, the southern region of Spain, generates a group of electronic documents in PDF format called session diaries, published in the www.parlamentodeandalucia.es site. They store all the discussed matters in every session and the participations of the members of parliament for these matters. These session diaries belong to different legislatures, composed of, at most, four years of politics activity. Nowadays, since the creation of the Parliament of Andalusia in 1982, there have been seven legislatures with more than 1500 PDF documents. Moreover, the sessions are recorded in video, so additionally to the transcriptions, the digital library of the Parliament is complemented with the videos.

In the session diaries, and therefore, in the videos, we can find all the participations of the members of parliament, and also all the agreements achieved in the plenary sessions of the Permanent and Commission Delegation passing laws or celebrating informative sessions with members of the regional Government. The session diary and its corresponding video are published in the website after the meetings of the deputies.

These documents could be accessed through by means of a search engine that works with a representation in XML of the PDF documents where the internal structure of the session diaries could be exploited. Then the user formulates a query and gets the relevant documents (Baeza-Yates, Ribeiro-Neto, 2001), or parts of them (Chiaramella, 2001). But, this is not the case of the videos, which may be accessed by date, basically. There is no link between the document of the session diary and the video. But it could be very useful for the user that when she/he retrieves a relevant document (the text), or a portion of it, she/he could watch the associated video at the same time. Then the structured information retrieval field (Chiaramella, 2001; de Campos et al., 2006) gives the possibility of making the decision of determining the type of XML element containing relevant information, so the user does not have to inspect the whole document to find the requested information. But also, the user could watch only the portion of the video in real time. This feature could be an added value for the search engine.

But before the information retrieval system could use the videos to be presented to the user, there is a previous work of synchronizing the text contained in the session diary and the corresponding video. The output of this stage would be the XML elements containing text from a speech marked up with time tags, corresponding to the beginning and end points in the associated video. To facilitate this annotation stage, a segmentation of the video is performed, by means of an automatic process. The video is partitioned in segments of similar content, detecting the boundaries. When there is a change of camera, a new segment is created. Finally, these segments are associated with the textual transcription of the speeches. Basically the requirements of our software are: 1) Automatic segmentation of videos; 2) Manual edition of the segmentation; 3) Annotation of the XML documents containing the transcriptions, synchronizing them with the segments of the videos.

Most of the existing segmentation algorithms found in the specific literature are designed for general videos (Camastra, F. and Vinciarelli, A., 2007). This means that they are complex algorithms prepared to detect the boundaries of the segment in all conditions. But in our context, a simple algorithm based on histogram comparison could work very well, as the case is. In this paper we describe the algorithm itself, how it has been improved, as well as the main features of the segmentation and annotation tools. Therefore, this paper is organised as follows: In the next section, a brief overview of existing annotation tools is presented. Section 3 will describe the segmentation tool, focussing on the algorithm as well as in the improvements. In Section 4, the annotation tool is explained. The next section, 5, will deal with the player used to show the videos. Finally, Section 6 will conclude and present future works.

2. BRIEF REVISION OF EXISTING ANNOTATION TOOLS

Having a look at the available tools for video annotation, we could find, among others, the followings: *IBM MPEG-7 Annotation Tool* (http://www.alphaworks.ibm.com/tech/videoannex): it allows annotating videos using the metadata associated to a MPEG-7 video. This is segmented automatically, and a key frame extracted. On this representative, the user can make the pertinent annotations about events of static scenes and key objects. For our purposes, the annotation is very restrictive, because only can be done using predefined scenes and objects. Also it is in MPEG-7, which is effectively a standard, but we are not considering these multimedia format. Another problem is that the segmentation can not be edited. *IBM Multimedia Annotation Tool* (www.alphaworks.ibm.com/tech/multimodalannotation) is very similar to the previous one, but also can annotate audio, background and foreground sounds. Both tools are free. *Virage VideoLogger* (www.virage.com) analyses the video and generates automatically a structured index about the content. It allows making manual annotations, but it is not free. This tool is really complex because of its high functionality. iFinder (www.imk.fhg.de/de/ifinder) is another tool which works with the MPEG-7 standard. It is able to segment videos, recognize faces, as well as to make speech recognition. It is a server client application. But it is not free and really complex. Ricoh MovieTool (www.ricoh.co.jp/src/multimedia/MovieTool/about/index.html) allows automatically segmenting MPEG-7 videos, but giving the possibility of refining the output, making it manually. A very interesting feature is the possibility of creating a hierarchy of segments. *ZGDV VideTo* (www.zgdv.de/zgdv/departments/zr4/Produkte/VIDETO) presents the interesting feature that the annotation could be stored in XML format, without any restriction (later it could be converted to MPEG-7). Patterns about specific domains could be created, facilitating the annotation. The segmentation is fixed and can not be edited. With the analysis of all these tools we realise that there is no tool that fulfils all our requirements.

3. THE VIDEO SEGMENTATION TOOL

When the annotation of a video is carried out, we have to present the information to the user in such a way that all the content is summarized in a small amount of information. An adequate way is to split the video in segments (Camastra, F. and Vinciarelli, A., 2007). A segment is a fraction of the video that has continuity. Once the video has been divided, it must be presented to the user in such a way that with only a look she/he could have an idea of the content of each segment. This is made showing a key frame, a representative of the content.

The segmentation based on difference between shots (Jain A. and Chaudhuri, S. 2002; Hanjalic, A., Lagendijk, R. and Biemond, J., 2006; Sáez, E. et al., 2003) obtains some features from each shot and compares them with those from the following shot. If the difference is greater enough, then it is considered that there is a change of scene. The most used features are the colours (Godil, A., 2004) or the edges of the image (Gastaud, M. and Barlaud, M., 2002). To obtain the differences of the shots, the format of the video storage could be considered (Calic, J. et al., 2002): shots storing the full image, or those storing only the change with respect to the previous shot. The segmentation based on flow of movement tries to detect the object in the scene (foreground and background objects) and analyses the movements in the following shots (Kobla, V. et al, 1997; Apostoloff, N. and Fitzgibbon, A., 2006; Lu, Y., Gao, W. and Wu, F., 2002). There also exist hybrid techniques which combines both methods (Liu, L. and Fan, G., 2005).

Our objective is to find the method that is able to segment the videos of the Parliament of Andalusia, in a correct and efficient way, without forgetting the complexity of the implementation, which has to be low. These videos show long scenes, with few movements, and sudden changes. This means that if the analysis of objects in the scene is not going to help, because the speakers are usually static in the talk. In the chamber of the Parliament, there are four fixed cameras, focussing to the speaker, a general view of the chamber, and two centred in the seats of the deputies. Therefore, the realization of these videos is usually very static, and therefore very easy to detect the changes of cameras. Considering the segmentation using shot differences, we shall notice that in the changes of camera, these differences are large, and inside the same segment, the differences are low, because the cameras are static and the movement is almost null. Therefore, we shall use this last method, considering the colour as the feature in which the method will be based on, basically because of the simplicity of the method and good results.

3.1 The Segmentation Algorithm

The segmentation algorithm that we present in this section is based on detecting differences between shots. More specifically, these differences will be given by the different colors of the shots. We shall adopt a grey scale representation. From the RGB images of the video, we get that representation using a simple transformation: $I(i,j) = (R(i,j) + G(i,j) + B(i,j))/3$, where I is the matrix that represents the image in grey scale; R, G and B are the matrices that represent the level of red, green and blue of the image, respectively, and i and j are the indexes of a pixel. The histogram of the image represents the number of times that a specific colour occurs in it. As we are working in grey scale, and considering that 8 bits are used to represent the intensity of 256 tones, we could represent the image by means of vectors of 256 elements. If we note H the vector, and H(i) the number of occurrences of the colour i in the image, then, in order to determine if there has been a change between two shots S1 and S2, with histograms H1 and H2, respectively, we could compute the difference of both vectors: $(H1 - H2)[i] = | H1[i] - H2[i] |$. Computing the difference for each colour, and summing up all of them, we have a scalar value of the difference between both. If this value is greater than a certain threshold, then there is a change in the shots.

This is a really simple algorithm as a set of shots are considered included in the same segment if the difference between their histograms is low. But experimenting with the videos of parliamentary sessions, we realised that the changes of cameras origins that the difference of histograms is high, so they could easily be detected. But it is also high (although a little bit smaller) for shots in which there is no such change in the camera. Therefore, mistakes might be made. This is due to a minimum variation of the luminosity of the image, which considerably alters the histogram of the image. For example, a simple movement of the deputy who is speaking could provoke that his/her jacket reflects the light in a different way. This could make that the colour represented in a shot is lightly different to the previous shot. Although the difference is not noticeable for the human eye, a minimal variation in the histograms increases the difference between both of them. In order to solve this problem, we have to achieve that between close histograms the difference is not so important, so we could soften the histogram, in such a way that each element depends on the neighbours, and therefore a light change does not affect the difference. A solution is the application of a convolution filter, which makes that each element of the vector is the sum of those closest elements. We have carried out such convolution using the vector [0.1, 0.2, 0.4, 0.2, 0.1]: $H1'[i] = 0.1*H1[i-2] + 0.2*H1[i-1] + 0.4*H1[i] + 0.2*H1[i+1] + 0.1*H1[i+2]$.

An important decision that will clearly have a great influence in the performance of the segmentation is the selection of the threshold value. It will depend on the resolution of the video, as in shots with a higher

resolution the difference of their histograms will be proportionally larger. Moreover, images with a higher number of colours will also present a higher difference among shots, so we will have to consider the number of tones contained in the images. Therefore, the threshold used in our algorithm takes into account the width of the image, its height, as well as the number of colours. It is defined as: Threshold = (Width · Height · No. of colours) / Factor. Factor is a parameter decisive to get an optimal segmentation.

3.1.1 Optimization of the Algorithm Efficiency

The first parameter that must be estimated is 'Factor', used to compute the threshold just mentioned. For the type of considered videos, its value has been obtained empirically studying several of them. The process which has lead to get it has been the following: first of all, we have obtained the difference for each pair of contiguous shots in each video; secondly, we have localized manually the cuts (changes of scene) that the segmentation algorithm should detect; and finally, once we have studied the values of differences in the shots in which there is a cut, we have selected a value for Factor such as the threshold value is sufficiently low to detect all the real cuts, and sufficiently high to not detect cuts which does not exists. With a threshold of 16,000 all the cuts are detected, and nothing except real cuts will be detected.

This basic segmentation algorithm works properly, but not efficiently. As the videos from the sessions of the Parliament of Andalusia are very long (about 5 hours), it is required to improve the segmentation speed, but without worsening the effectiveness. The first attempt is to reduce the number of shots to be considered. In the original algorithm, the histograms of each pair of shots are compared. But it is noticed that if there is a cut between the shots i and i+1, this cut also will be place between the shots i-x and i+y+1. This means that we could detect the cut analysing the difference of histograms between two non contiguous shots surrounding shots i and i+1. Therefore, instead of analysing all of them, we will discard x shots between each studied pair. The next step will be to refine the segmentation to locate exactly when the cut is produced, i.e. if the algorithms find a cut between shots x and y ($x < y$), then it will visiting each pair of shots i and i+1, with $x < i < y$, contained in the interval, to detect the cut. This process is much faster than to compare each single pair of shots and also offers the optimal result. For videos in which there is not an excessive number of cuts, as the case is, this is an appropriate method.

A second optimization is related to the size of the image. If the difference of histograms with the full image it is enough to differentiate shots, we could suppose that the histogram of only a portion of the image could offer enough information to perform this action, adjusting the threshold to the corresponding size. Then the reduction will improve in the efficiency of the process, as the number of computations is lower. This is not only interesting in terms of efficiency but also in term of efficacy of the segmentation. In most of the videos, there are movements but inside a scene without a compulsory change of shot. In this case, the algorithm could make a mistake because, in these cases, the movement originates a change in the colours of the image. In the case of the videos of the Parliament of Andalusia, where the location of the cameras is known, we could know the part of the image that will suffer less interferences of this type. If we only use the pixels of the area to compute the histograms, we reduce the risk of finding cuts where there are not. In the case of study, the lower left corner of the image is considered, as empirically it is the area where fewer movements are observed.

Finally, once the segmentation is finished, we have to select a representative shot of each segment (key frame), in order to present them to the user. There are several techniques to select the key frame (Avrithis Y., 1999; Calic, J. and Thomas B., 2004; Calic J. and Izquierdo, E. 2002). But in the case of the videos of the Parliament, and due to their nature, it is not required a sophisticated technique as the user is going to use it as a guide to the annotation phase. Then, the shot which is in the 10% of the total length of the segment is selected. With this selection there are no mistakes, as the segment is supposed to be accurate in terms of very similar contents of all its shots.

3.1.2 Experimentation and Evaluation

In this section, we shall show how the selection of the best values for the two main parameters of the segmentation algorithm has been made, so the best performance is reached. In this line, we have to decide the size of the window used to compute the histograms and later the difference between them (reduction factor), and the number of shots ignored in each stage of histogram comparison, and the posterior refinement to find the exact point of change. We have created a set of 6 videos of different length. Their durations as well as the

number of exact segments, obtained manually when there is a change in the scene are the followings: V5(5m,5), V9(9m, 7), V20(20m 26s, 27), V30(30m 13s, 36), V40(40m 8s, 29), VF(5h 46m 36s, 433).

The first experiment tries to show how important is to use a reduction factor. In Table 1 we could show the accuracy of the segmentation considering the first 5 videos, with and without refinement (reduction factor of 2 and 24 shots between two compared shots). As it may be noticed, the refinement is configured as an essential technique to maintain a high accuracy.

Table 1. Percentage of correct segment detected (accuracy) with and without refinement

Videos	Accuracy with	Accuracy without
V5	100.00	27.78
V9	100.0	87.50
V20	92.59	20.0
V30	88.89	25.53
V40	100.00	76.32

With respect to the reduction factor, we have tested that same set of videos with several values, from 1 to 9. In Figure 1, we can observe the graphical representation of the reduction factor and segmentation time. Figure 2 shows the reduction factor and the percentage of correct segments. Initially, these experiments have been carried out with the number of shots jumped set to 24 and refinement applied.

Figure 2. Time vs. Reduction Factor

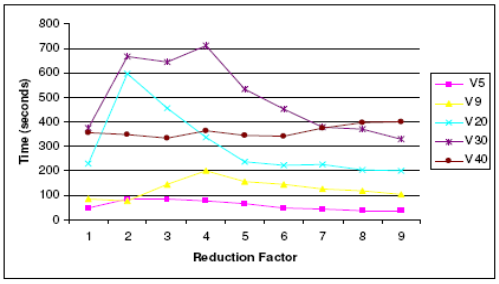
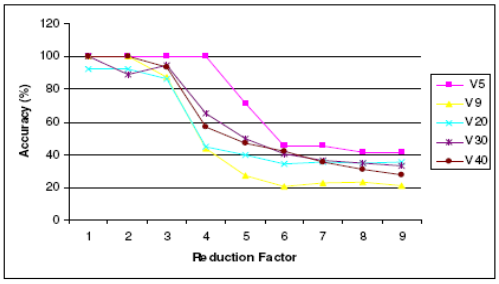


Figure 2. Accuracy vs. Time



The segmentation time is the sum of times of different tasks: the shot extraction process from the video, the computation of its histogram, the convolution, the comparison with the previous shot, and the refinement, in case of needing it. In affirmative case, the time would increase considerably, as the algorithm has to look for the change from the previous shot until the shot where the real change is produced. When the reduction factor is increased, the time required to compute the histogram is reduced. But, as a smaller portion of the image is used, there is less information available and therefore there will me more segmentation errors.

These two situations are found in videos V30 and V40. In the former, as there are lots of cuts, the reduction factors, 2 to 7 make the algorithm to spend more time testing the differences between histograms when refining than the time saved using image areas smaller. In the latter, the cuts are less frequents, so better times are obtained with factors 2 to 6.

With respect to the accuracy of the algorithm, we may notice that, in most of the cases, from a reduction factor greater than 2, it decreases considerably. This is due to the fact that there is less information about the change of images testing a small area, so the algorithm makes more mistakes. As a conclusion, we shall use a reduction factor of 2 because the accuracy is usually very close to the 100%, and the segmentation time is usually one of the shortest.

Fixed the value of this parameter, next we shall study the number of shots ignored between two shots to be compared. In this case, the analysis will be done with VF in two steps. Figure 3 shows the segmentation time for values 1, 12, 22, 32. We can observe how the value 12 gets a better performance. In a second step, we have tried with values 10, 12, 14, 16 and 18. In this case, 14 is the best. Until there the segmentation time decreases. From it, this time increases.

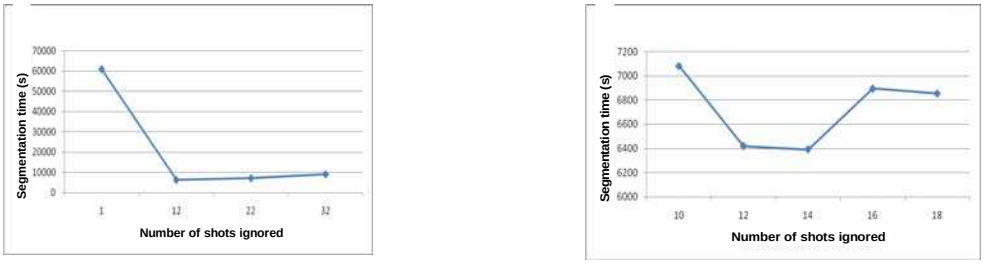


Figure 3. Segmentation time for different numbers of shots ignored.

3.2 Features of the Segmentation Application

Before starting with the main features of the segmentation tool, just to mention that it has been developed in Java, using the Java Media Framework (JMF) library to incorporate multimedia elements and tasks in our application in a comfortable and easy way.

There exists the possibility of making a batch segmentation of several videos. This facility is implemented because the segmentation time is usually long taking into account the average duration of them. Then the user selects a group of videos to be segmented, and the application will process each one, given as an output the segments of each video.

Once the automatic segmentation of a video has been carried out, the software offers the possibility of editing the segmentation manually. The output of this process is a set of segments, represented by a key shot. The user may need to adjust the segmentation to prepare the posterior process of annotation, in order to be more accurate. An example of this need is the case in which a deputy is speaking, and a camera focusing her/her. Then there is a change of camera, and that focuses another deputy. In spite of this change of scene, the first speaker is continuing with her/his talk. The segmentation algorithm would give as an output two segments (correct), but the user may wish to join both of them, because the current speech is being given in both of them. Therefore, the user is allowed to edit the segments, combining them (two segments s1 and s2 could be joined if they are contiguous) or dividing segments in two.

In the application, all the segments found by the algorithm are shown in a window (see the left figure). More specifically, the key shot of each segment. If we click in one of them, then all the shots contained in it are shown in a separate window. By means of submenus activated by the left bottom of the mouse, the user could edit the segments. A viewer allows playing any segment.

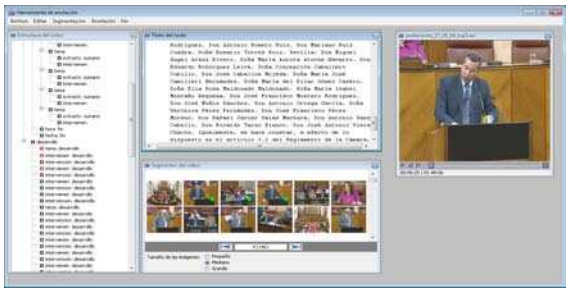
In order to show a shot, the application must access the video. This means a high access, the player must be placed in the correct position, the file in the hard disk must be accessed and copy the contents in main memory. For a real time usage, this would mean that there is a high response time. To avoid this, we have implemented a cache module, in which we store the shots of the video in a cache memory (those which are more probable to be shown), so their access will be much faster.

4. THE ANNOTATION TOOL

Once the segmentation of a video has been performed, and the posterior edition, the user is ready to carry out the annotation stage. The input of this process will be the sets of segments found in the video corresponding to a parliamentary session and the transcription of the speeches given in the chamber for that video. This transcription is represented by means of an XML document, which contains the structure of the session, as well as the text itself. The output will be the XML document containing the transcription synchronized with

the video by means of time stamps in the elements of the document. The segment of the video related to a specific text could be easily access.

The annotation tool will consist of a manual association of segments with the corresponding elements in the XML document, so each tag will have a link to the part of the video where this text is played.



The figure on the left shows the user interface of the annotation tool. It is composed of four windows. The left window shows the tree representation of an XML document containing session diary. If a leaf node in clicked, then the text contained in it is shown in the central uppest windows. The window below contains the segments found in the first part of the process. Finally, a player is included in the right part of the interface, in order to help the annotation.

The annotation process is as follows: the user selects a segment in the video, then find the node in the XML document containing the transcription of the audio of that segment, and by means of a drag and drop action, associate the former with the latter. These steps are repeated until all the segments have been assigned to a node of the document.

Actually, with the association of a segment to an XML element of the document, we introduce a pair of attributes to the corresponding tags, containing the beginning and ending times of the segment. This information will be enough to access the portion of the video in retrieval time.

Once all the segments have been assigned to leaf nodes of the XML tree, and therefore, all the affected tags have been complemented with temporal attributes linking the text with the video, it is necessary to propagate the times to upper nodes until reaching the root node. In our context of the parliamentary sessions, the text is usually contained in 'paragraph' tags. These are contained in 'speech' tags (the speech of a deputy), which are included in the debate of a parliamentary initiative (several MPs usually takes part in the discussion of a point of the agenda). Finally, several points of the agenda, plus general information of the session, are included in the 'session' tag, the root of the tree. Then if the search engine retrieves a whole point of the agenda, it should also allow the access to all the segments related to the elements contained in that tag. Therefore, a propagation of times is performed from leaf nodes to the root node.

5. THE VIDEO PLAYER

Finally, once the transcriptions in form of XML documents have been completed with time stamps, the search engine, once it has indexed the parliamentary digital library, is ready to be used. When a user query is formulated to the system, it ranks all the elements from all the documents according to their relevance with respect to the query, and presents this ranking to the user. Then she/he could click on a link, in which she/he is interested, to inspect the text or to watch the video segment, which is played by means of a player.

An important element in the retrieval stage is the video player. We have to take into account two important features in order to make the decision of what type of player to use: first of all, it must be an accessible element for any type of user, independent on the platform used; secondly, in order to save broad band in the video transmission, we only have to send to the player from the server the portion of the video in which the user is interested in, not the whole video. Usually the videos we are working with are very heavy, so the user can not be waiting to download 40 Mbytes of video to watch only 30 seconds. This must be fast.

For this reason, the player is implemented using the Flash technology, which allows creating vector animations, easily dimensionable, as well as with low storing requirements. This space optimization together with the possibility of the feature of loading the image at the same time it is shown in the browser, allows adding visual elements that animate a web avoiding the fact of a really long load of the web page. Moreover, Flash implements a programming language called ActionScript. One of the things that allow this language, by means of its libraries, is to incorporate a multimedia player inside a Flash animation.

An important additional feature by which we made the decision of using this technology to develop the player used in the search engine for the digital library of the Parliament of Andalusia, is the possibility of playing a piece of a video in any time of it, without downloading the whole video in the computer.

6. CONCLUSION AND FURTHER RESEARCH

In this paper we have presented a video segmentation and annotation tool for the videos and the documents belonging to the digital library of the Parliament of Andalucía. The videos records the sessions of debate, while the documents, represented in XML, stores the exact transcriptions of the speeches given in the session. From the point of view of the access to this material from the user, it would be very interesting that, as the output of a query, the system returned the text associated with its corresponding portion of video. To implement this, we need a first step in which the video is segmented in pieces, and later, each interval is assign to a text element belonging to the XML document. Therefore, in this paper, we have presented a very simple but effective and efficient segmentation algorithm to carry out this task, as well as an annotation tool to synchronize both media. Both tools have been designed thinking in the easiness of use.

With respect to further research, we think that the human work in the process of synchronization is too much, so we would like to reduce it, changing the philosophy of the tool: instead of segmenting video and later manually annotate it, we are researching on a process in which automatically the audio of the session would be synchronizing with the text of the transcriptions. We think this would improve the process, alleviating the human intervention.

ACKNOWLEDGEMENT

Work jointly supported by the Spanish 'Ministerio de Educación y Ciencia' (TIN2005-02516), 'Consejería de Innovación, Ciencia y Empresa de la Junta de Andalucía' (TIC-276), and Spanish research programme Consolider Ingenio 2010: MIPRCV (CSD2007-00018).

REFERENCES

Apostoloff, N. and Fitzgibbon, A., 2006. Automatic video segmentation using spatiotemporal T-junctions". BMVC.

Avrithis Y., 1999. *A Stochastic Framework for Optimal Key Frame Extraction from MPEG Video Databases*. Computer Vision and Image Understanding, 75(1-2), pp: 3 – 24.

Baeza-Yates, R.; Ribeiro-Neto, B., 1999. *Modern information Retrieval*, Addison-Wesley.

Calic, J. and Thomas, B., 2004. *Spatial analysis in key-frame extraction using video segmentation*. Proc. of 5th Int. Workshop on Image Analysis for Multimedia Interactive Services.

Calic, J. et al., 2002. *Temporal video segmentation for real-time key frame extraction*. *ICASSP*.

Calic J. and Izquierdo, E. 2002. *Efficient key-frame extraction and video analysis*. ITCC, Las Vegas, USA.

Camastra, F. and Vinciarelli, A., 2007. *Video Segmentation and Key frame Extraction*. In Advanced Information and Knowledge Processing. Springer, pp. 413-430.

Chiaramella, Y., 2001. *Information retrieval and structured documents*, Lectures on IR, Springer, 286-309.

de Campos, L. et al., 2006. *Garnata: An Information Retrieval System for Structured Documents based on Probabilistic Graphical Models*. 11th IPMU Conference. 1024 - 1031,2006.

Gastaud, M. and Barlaud, M., 2002.*Video segmentation using active contours on a group of pictures*. ICIP, 2, pp. 81-84.

Godil, A., 2004. *ViSA: Video Segmentation and Annotation*. UPA Conference.

Hanjalic, A., Lagendijk, R. and Biemond, J., 2006. *Automated Segmentation of Movies into Logical Story Units*. Information Systems, 31(7), 638 – 658.

Jain A. and Chaudhuri, S. 2002. *A Fast Method for Textual Annotation of Compressed Video*. ICVGIP.

Kobla, V., Doermann, D., Rosenfeld, A. Compressed Domain Video Segmentation.

Kobla, V. et al., 1997. *Compressed domain video indexing techniques using DCT and motion vector information in MPEG video. SPIE conference on Storage and Retrieval for Image and Video,* pp. 200-211.

Liu, L. and Fan, G., 2005. Combined Key-frame Extraction and Object-based Video Segmentation. IEEE Transactions On Circuits And Systems For Video Technology, 15(7).

Lu, Y., Gao, W. and Wu, F., 2002. *Automatic video segmentation using a novel background model*. ISCAS.

Sáez, E. et al., 2003. *New edge-based feature extraction algorithm for video segmentation*. 15th Ann. Symp. Electronic Imaging, Science and Technology.