Automatic text recognition in digital videos

Rainer Lienhart and Frank Stuber

University of Mannheim, Praktische Informatik IV, 68131 Mannheim, Germany
{lienhart, stuber}@pi4.informatik.uni-mannheim.de

ABSTRACT

We have developed algorithms for automatic character segmentation in motion pictures which extract automatically and reliably the text in pre-title sequences, credit titles, and closing sequences with title and credits. The algorithms we propose make use of typical characteristics of text in videos in order to enhance segmentation and, consequently, recognition performance. As a result, we get segmented characters from video pictures. These can be parsed by any OCR software. The recognition results of multiple instances of the same character throughout subsequent frames are combined to enhance recognition result and to compute the final output. We have tested our segmentation algorithms in a series of experiments with video clips recorded from television and achieved good segmentation results.

KEYWORDS: Character segmentation, character recognition, content-based indexing and retrieval, video processing

1 INTRODUCTION

In the age of multimedia, video is an increasingly important and common information medium. However, most current video data is unstructured, i.e. stored and displayed only as pixels. There is no additional content information such as year of production, starring actors, director, producer, costume designer, places of shots, positions and types of scene breaks etc. Therefore, the usability of raw video is limited, precluding effective and efficient retrieval. Consider the thousand of MPEG-encoded films on the Internet. Beyond the title and a short description rarely can any information be found about the content and structure of these films, therefore making it very difficult to find e.g. specific kinds of films or scenes. Information on video content would be highly desirable.

Usually, this information has to be generated manually, but manual annotation of video is very time-consuming and costly. Therefore, content-based retrieval and browsing prompt a demand for automatic video content analysis tools for indexing[2,15,16,17]. One important source of information about videos is the text contained therein. We have developed algorithms for automatic character segmentation and recognition in motion pictures. These algorithms extract automatically and reliably the text in pre-title sequences, credit titles and closing sequences with title and credits. The algorithms make explicitly use of typical characteristics of text in videos as generated by video title machines or equivalent devices/techniques, in order to enhance segmentation and, consequently, recognition performance.

The remainder of the paper is organized as follows. Section 2 reviews related work on text segmentation and text recognition in videos. We then describe the features of characters and text appearing in motion pictures and present in section 4 our feature-based approach to segmentation of character candidate regions which is based upon the character features stated in section 3. Section 5 discusses our recognition algorithms. It is followed by some information on the implementation in Section 6. In Section 7 we present empirical results as evidence that our algorithms lead to good segmentation results. Finally, we conclude our paper with a summary and outlook on future work.

2 RELATED WORK

Existing work on text recognition has focused primarily on optical character recognition in printed and hand-written documents since there exists a great demand in and market for document readers for office automation systems. These systems have attained a high degree of maturity[6]. Further text recognition work can be found in industrial applications. Most of this work concentrates on a very narrow application field. An example is the automatic recognition of car license plates[13]. The proposed system works only for characters/numbers whose background is mainly monochrome and whose position is restricted. Very little work on recognition of characters in text appearing in video broadcasts has been published.

Michael A. Smith and Takeo Kanade briefly describe in[12] a method which concentrates on extracting regions from video frames that contain textual information. However, they do not deal with the preparation of the detected text for standard optical character recognition software. In particular, they do not try to determine the characters' outline or to segment the individual characters. They keep the bitmaps containing text as they are. Human beings have to parse them. They characterize text as a "horizontal rectangular structure of clustered sharp edges"[12] and use this feature to identify text segments. We also employ this feature in our approach in the fill factor step. Unlike their approach, this feature plays only a small roll in our segmentation process of character candidate regions. We also utilize multiple instances under varying conditions to enhance segmentation and recognition performance.

Another interesting approach to text recognition in scene images is that of Jun Ohya, Akio Shio, and Shigeru Akamatsu. Characters in scene images can suffer from a variety of noise components. Text in scene images exists in a 3D space, so it can be rotated, tilted, slanted, partially hidden, partially shadowed, and it can appear under uncontrolled illumination[7]. In view of the many possible degrees of freedom of text characters, Ohya et al. restricted them to being almost upright, monochrome and not connected in order to facilitate detection. This makes the approach of Ohya et al. feasible for our aim, despite the fact that they focus on still images rather than on video streams and consequently do not utilize the characteristics typical of text appearing in video. Moreover, we focus on text generated by video title machines rather than on scene text.

3 FEATURES OF CHARACTERS IN PRE-TITLE SEQUENCES, CREDIT TITLES AND CLOSING SEQUENCES

Text in videos serves many different purposes: At the beginning and/or the end of a broadcast it informs the audience about the title, director, actors, producer, etc. of these. Within a broadcast text also gives important information about the subject currently presented. For instance, text in sportscasts often informs about the score, while in newscasts and documentary films speaker's name and place and/or important information about the current subject is presented. Text in commercials informs about the message, the product name or the company name. These text appearances all have in common that they are carefully directed. They do not appear by accident, they are overlaid over the scene and are made to be read.

In addition, text can also appear in scenes as part of the scenes: For instance, in a scene of a shopping mall many shop names can be seen in the video. Such text in scenes is very difficult to detect or recognize: It may appear under any tilt and slant, in any lighting, and upon straight or wavy surfaces (e.g. text on a T-shirt).

We do not deal with scene text in this paper, rather we concentrate exclusively on text added to the video artificially, especially by means of video title machines. The reason being that text which is superimposed over a scene is fundamentally different from text contained in scenes, and we did not want to deal with two different problems at once. Thus, in the following the word "text" and "character" will exclusively refer to those produced by video title machines or by equivalent devices/techniques.

Before characters and, thus, words and text can be recognized, the features of their appearance have to be analyzed. Our list of features includes:

- Characters are monochrome. Only a very small percentage are polychrome and are of no further interest here.
- Characters are rigid. They do not change their shape, size or orientation from frame to frame. Again the very small percentage of characters that do change size and/or shape are of no further interest here.
- Characters have size restrictions. A letter is not as large as the whole frame nor are letters smaller than a certain number of pixels as they would otherwise be illegible to human beings.
- Characters are either stationary or linearly moving. Stationary characters do not move. Their position remains fixed over multiple subsequent frames. Moving characters move steadily and

also have a dominant translation direction: Generally, they move either horizontally or vertically. Moreover, many just move from right to left or bottom to top.
- Characters contrast with their background. Artificial text is designed to be read and, thus, must be in contrast to its background. But as we will see later, due to the narrow bandwidth of the TV signal, this point does not hold true for all character outlines.
- The same characters appear in multiple consecutive frames (temporal relation).
- Characters often appear in clusters at a limited distance aligned to a horizontal line (spatial relation), since that is the natural method of writing down words and word groups. But this is not a prerequisite, just a strong indicator. From time to time just a single character might appear on one line.
- Character outlines/borders are degraded by current TV technology and digitizer boards. Characters often blend into the background, especially on their left side. Monochrome-designed characters do not appear to be monochrome any more. The color is very noisy and sometimes changes slightly spatially and temporally e.g. by interference with the colors of the surroundings. Even stationary text may jump around by a few pixels. Those are typical analog television/ video artifacts.

Any (artificial) text segmentation and recognition approach has to be based upon these observed features. Next we describe our use of them.

4 SEGMENTATION OF CHARACTER CANDIDATE REGIONS

Theoretically the segmentation step extracts all pixels belonging to text appearing in a video. However, this cannot be done without knowing where and what the characters are. Therefore, the actual aim of the segmentation step is to divide the pixels of each frame of a video into two classes:
- regions which do not contain text and
- regions which possibly contain text.

Regions which do not contain text are discarded, since they cannot contribute anything to the recognition process, and regions which might contain text are kept. We call them *character candidate regions* since they are (not exactly) a superset of the character regions. They will be passed on to the recognition step for evaluation.

Here we describe the segmentation process. It can be divided into three parts, each part increasing the set of non-character regions of the previous part by further regions which do not contain text, thereby reducing the character candidate regions, approximating them more and more to real character regions. We first process each frame independently of the others. Then, we try to profit from the multiple instances of identical text in consecutive frames. Finally, we analyze the contrast of the remaining regions in each frame to further reduce the number of candidate regions and to build the final character candidate regions. In each part we utilize the character features as described in Section 3.

4.1 Segmentation of character candidate regions in single frames

Monochromaticity

We start with the original frame (Figure 1). Due to the assumed monochromaticity of characters we segment the frame into homogeneous gray scale segments in a first processing step. We employ the *Split and Merge* algorithm proposed by Horowitz and Pavlidis[4] to perform the segmentation. It is based on a hierarchical decomposition of the frame. According to Horowitz and Pavlidis, the split process begins with the entire image as the initial segment, which is then split into quarters. Each quarter is tested against a certain homogeneity criteria to determine whether the segment is “homogeneous enough”. If not homogeneous enough, the segment is split again into quarters. This process is applied recursively until only homogeneous segments are left. We use the standard homogeneity criterion: The difference between the largest and smallest gray tone intensities must be lower than a certain threshold. We call the threshold *max_split_distance*. A homogeneous segment is assigned its average gray level. Next, in the merge process, adjacent segments are merged together if their average gray tone intensity difference is less than a parameter called *max_merge_distance*. As a result, all monochrome characters appearing in the image should be contained in some of the monochrome segments. For our example frame, the Split and Merge algorithm outputs the image shown in Figure 2.

Size restrictions
The segmented image now consists of regions homogeneous according to their gray tone intensity. Some regions are too large and others are too small to be instances of characters. Therefore, monochrome segments whose width and height exceed *max_size* are removed, as are connected monochrome segments whose combined expansion is less than *min_size*. The impact on our example image can be seen in Figure 3 (deleted segments are black).

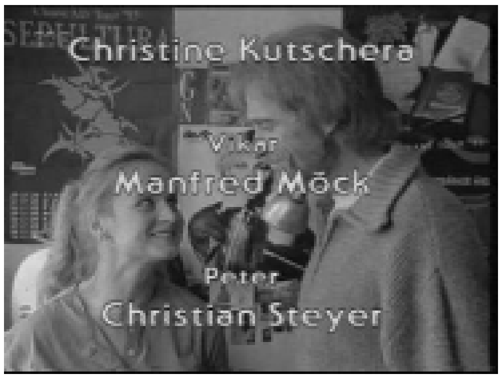

Figure 1: Original video frame.

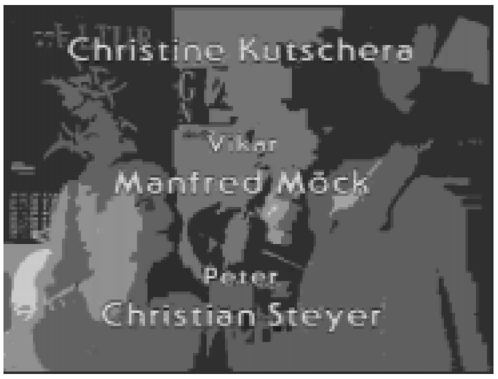

Figure 2: Applying modified split and merge algorithm to Figure 1.

4.2 Enhanced segmentation based on consecutive frames

Since we are analyzing text in videos generated by video title machines the same text typically appears in a number of consecutive frames. It is obvious that the segmentation result can be improved by using these multiple instances of the same text because each character of the text often appears somewhat altered from frame to frame due to noise, changing background and/or changing position. Thus, we have to detect corresponding character candidate regions in consecutive frames.

Motion analysis
As already mentioned in Section 3 the text considered here is either stationary or linearly moving; and even stationary text may move by some pixels around its original position from frame to frame. Consequently, we have to perform motion analysis in order to detect corresponding character candidate regions in consecutive frames. Motion is estimated by means of block-matching, since block-matching is suitable for rigid objects and characters are assumed to be rigid, changing neither their shape, orientation nor color. Moreover, block-matching is very popular and used for motion compensation in the international standards for video compression such as H.261 and MPEG[3]. Our matching criterion is the minimum mean absolute difference criterion[14]. The mean absolute difference (MAD) is defined as

$$MAD(d_1, d_2) = \frac{1}{|R|} \cdot \sum_{(x,y) \in R} \left| g(x,y) - g(x+d_1, y+d_2)) \right|$$

R specifies the block for which the translation vector has to be calculated. The displacement estimate $(\hat{d}_1, \hat{d}_2)$ for block R is given as the displacement where the MAD value is minimal. The search area is restricted to $|d_1|, |d_2| \le search\_range$ and derived from the speed of fast-scrolling credit titles.

The upcoming question is, how to determine the location and size of the blocks to be used for motion estimation. It is obvious that the quality of the displacement estimation depends on the position and size of the block we try to match with its instance in the successive frame. For instance, if the chosen block is too big, it may be impossible for the algorithm to find an equivalent block since parts of the block may have left the frame (this can happen with scrolling titles) or parts of the block in the subsequent frame may have been correctly recognized as background while in the first frame they were not filtered out and remained part of the character candidate region.

To avoid these problems we take advantage of the fact that characters appear as words and are therefore placed in rows. We select our block R by means of the following algorithm: The input image is binarized (background =

black, rest = white) and each white pixel is dilated by a circle of a specified radius. As can be seen from Figure 4, the characters and words now constitute a compact region. We frame each connected cluster in a rectangle and define it as block R. If the fill factor is above a certain threshold the block is used for motion analysis. If the fill factor is below a certain threshold, the block is divided recursively into smaller blocks until the fill factor for a resulting block exceeds the threshold. For each resulting block which meets the required fill factor, block-matching motion analysis is performed.

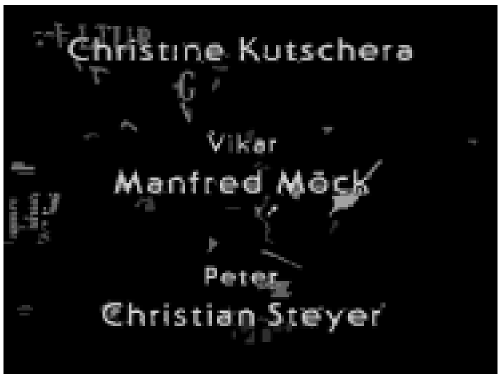

Figure 3: Applying size restrictions to Figure 2.

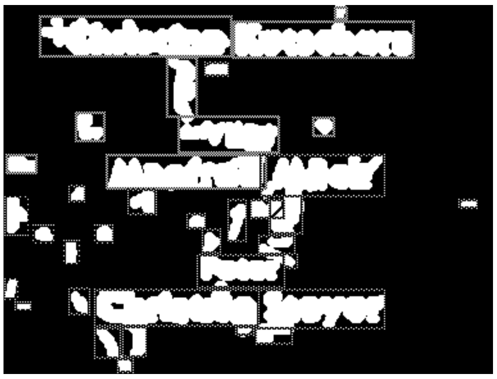

Figure 4: Binarized and dilated Figure 3. Blocks are marked by rectangles.

Blocks without an equivalent in the subsequent frame are discarded. Also, blocks which have an equivalent in the subsequent frame but show a significant difference in their average gray tone intensity are discarded. The resulting image is passed on to the next segmentation step (Figure 5).

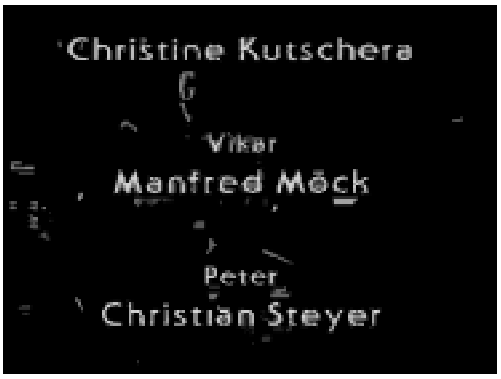

Figure 5: Image result after applying motion analysis to two consecutive frames of Figure 3.

4.3 Enhanced segmentation of character candidate regions by contrast analysis

Contrast analysis

Characters generated by video title machines usually contrast with their background. So this is also a necessary condition for character candidate regions. Therefore, each region left from the previous segmentation step is checked to see whether its outline partially contrasts strongly with the background and/or other remaining regions. Especially the dark shadows often laid below characters to enhance legibility should result in a very strong contrast between the character regions and parts of their surrounding. If no such contrast is found for a region, we conclude that it cannot belong to a character and discard the region.

Contrast analysis is performed by the following processing queue: we calculate the Canny edge map[1] and apply a significantly high threshold (called *canny_threshold*) to limit response to strong edges. The resulting edge image is dilated by *dilation_radius*. Then, regions from the motion analysis segmentation step are discarded if they do not intersect with any dilated edge. For our example this leads to the result shown in Figure 6.

Fill factor and width-to-height ratio
The blocks and their respective fill factors are calculated again for each remaining character candidate region as described in the motion analysis section above. If the fill factor is too low, the corresponding regions are discarded. Next, the width-to-height ratio of the blocks is calculated. If it exceeds certain limits, i.e. does not lie between *min_ratio* and *max_ratio*, the corresponding regions are also discarded. This process leads to the final segmentation image. In Figure 7 it is shown for our sample video frame.

Segmentation Result
So far, the character candidate regions for each frame have been extracted. The regions are stored in new frames, thereby producing a new video. In these frames pixels belonging to character candidate regions retain their original gray level. All other pixels are marked as background. Segmentation is now finished and the new video can be analyzed frame by frame by any standard OCR software.

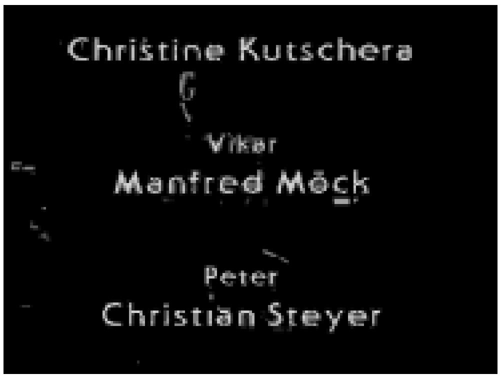

Figure 6: Image result after contrast analysis of Figure 5.

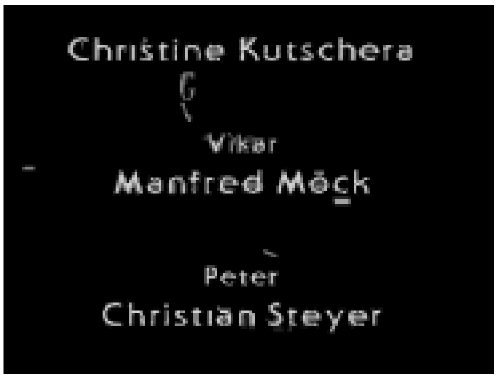

Figure 7: Final segmentation image.

5 CHARACTER RECOGNITION

Segmentation delivers a video showing character candidate regions. In character recognition each frame has to be parsed by an OCR software. We implemented our own OCR software using a feature vector classification approach as described in[11]. However, this software is far from perfect and use of a commercial software package should result in better recognition rates.

Since we are analyzing video, each character appears in multiple consecutive frames. Thus, all instances of recognition of the same character have to be combined into one single recognition result. Corresponding characters and character groups are identified by motion analysis as described in Section 4.2. Thus, we are able to relate the multiple independent recognition results to the same character and word. The most frequent result constitutes the final recognition result.

6 IMPLEMENTATION

The segmentation algorithms were implemented on a SUN SPARCstation 5 under Solaris 2.4 and on a DEC ALPHA 3000 under Digital Unix 3.2 with 2300 lines of C code. They are part of the MoCA Workbench[5] and require the Vista library 1.3 as a basis[9,10]. The OCR software is implemented in C with 1200 lines of C code and trained with 14 different postscript fonts. However, the second part of the character recognition process, i.e. the combination of all OCR recognition results into one final text output, is still under implementation and will soon be finished.

7 EXPERIMENTAL RESULTS

We have tested our segmentation approach on 8 digital video samples. The video data was digitized from several

German and international TV broadcasts as 24-Bit JPEG images at a Q-factor of 50[8], a size of 384 by 288 pixels and at 14 fps. All JPEG images were decoded only as gray scale images. We have two samples for each of the following classes:

- stationary text, stationary scene;
- stationary text, moving scene;
- moving text, stationary scene; and
- moving text, moving scene.

Moving text means that the text moves across the scene, e.g. from bottom to top or right to left. Equivalently, moving scenes denote scenes with significantly high motion or, more generally speaking, significantly strong changes. A stationary scene is either a still image or a very static scene such as speaker scenes in newscasts. Stationary text remains at a fixed position. Moreover, the characters in the video samples vary in size, color and shape.

In our experiments we used the following values for the different parameters:

- *max_split_distance* = 30
- *max_merge_distance* = 30
- *max_size* = 70 pixels
- *min_size* = 5 pixels
- *search_area* = 20 pixels
- *dilation radius* = 3 pixels
- *fill_faktor_threshold* = 0.7 (Section 4.2) and 0.3 (Section 4.3), respectively
- *canny_threshold* = 80
- *min_ratio* = 0 and *max_ratio* = 6

The experimental results can be found in Table 1. The first column specifies the type of the video sample, followed by its length measured in frames. The third column contains the actual number of characters in the corresponding video sample. It is measured by writing down all video title text appearing in the sample video and counting the characters. Thus, the character number refers to the text in the video not to the sum over the number of characters appearing in all frames. The fourth column gives the number and percentage of characters segmented as character candidate regions by our segmentation algorithms. Segmentation performance in our experiments is always very high ranging from 86% to 100%, and thus providing experimental evidence of the quality of our algorithms. For video samples with moving text and/or moving scene the segmentation performance ranges even from 97% to 100%. These performance measurements are consistent with our approach; it cannot profit from the multiple instances in a stationary scene with stationary text, since all instances of the same character have the same background. Thus, the segmentation performance is lower. Readers interested in seeing the eight video clips can retrieve them from *http://www.informatik.uni-mannheim.de/~lienhart/MoCA_TextRecognition/*

The quality of the character candidate regions for the recognition process cannot be evaluated here since we are dealing only with character segmentation. Such evaluation can only be done in combination with an OCR software and has to be investigated in future experiments.

Another important quality factor in the segmentation process is the average reduction factor of relevant pixels. It specifies the reduction of the number of relevant pixels by our segmentation process decreasing the workload for the recognition process. Moreover, the higher the reduction factor, the fewer non-character regions that are still part of the character candidate region, thereby reducing mis-recognition by an OCR software. The average reduction factor is given by

$$\text{average reduction factor} = \frac{1}{\text{\# of frames in video}} \cdot \sum_{f \in \text{video}} \frac{\text{\# of pixels in all character candidate regions of frame f}}{\text{\# of pixels in original frame f}}$$

The last column of Table 1 shows the characters per frame for the video samples. It correlates with the average reduction factor.

video type	frames	characters	thereof contained in character candidate regions		reduction	characters per frame
stationary text, stationary scene	400	137	131	96%	0.058	0.34
stationary text, stationary scene	400	92	79	86%	0.028	0.23
stationary text, moving scene	116	21	21	100%	0.035	0.18
stationary text, moving scene	400	148	144	97%	0.037	0.36
moving text, stationary scene	139	264	264	100%	0.065	1,90
moving text, stationary scene	190	273	273	100%	0.112	1.44
moving text, moving scene	202	373	372	99,7%	0.130	1.85
moving text, moving scene	400	512	512	100%	0,090	1.28

Table 1: Segmentation results.

To give also empirical evidence of the stability of our algorithm we tested it with a ninth video sample without any text. The video sample consisted of 500 frames, and the average reduction factor was 0.038. This value is very low in comparison to the ones for the video samples containing text. Thus, our algorithm is also able to detect scenes which probably contain no or only little text. But, the final decision is left to the OCR tool.
Some readers might ask what about text as part of a scene? Does it distort the algorithm?. In general, scene text is not extracted. But if it has the same features as artificial text it is extracted. This usually happens for scene text which is used in videos for similar tasks as artificial text, for instance a close shot of a city name.

8 CONCLUSIONS AND OUTLOOK

We have presented algorithms for automatic character segmentation in motion pictures which extract automatically and reliably the text in pre-title sequences, credit titles and closing sequences with title and credits. Experimental results from 8 digital video samples comprising a total of 2247 frames are very promising. Our algorithms extracted between 86% and 100% of all add-on text appearances in our digital video samples. For video samples with moving text and/or moving scene the segmentation performance ranges even from 97% to 100%. The resulting character candidate regions can easily be parsed by standard OCR software. Our recognition algorithms then combine all instances of recognition of the same character into a single recognition result.

Currently, our algorithms process gray scale image. This makes it difficult to detect e.g. yellow text upon a blue-gray background since these colors do not contrast in gray scale images. Consequently, our approach was unable to segment such text reliably. Our future plan is to extend the algorithm to operate on color images in an appropriate color space and calculate contrast in these color images.

In the future, we are also planning to incorporate the text segmentation and text recognition module into our automatic video abstracting system, so as to be able to extract the movie title and the most important actors of a movie, since they are an essential part of the abstracting. The algorithms will also be built into our automatic video genre recognition system[2] with an eye towards improved performance, since certain text might be characteristic for certain genres.

References

[1] John Canny, “A Computational Approach to Edge Detection”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 8, No. 6, pp. 679-697, Nov. 1986.

[2] Stefan Fischer, Rainer Lienhart, and Wolfgang Effelsberg, “Automatic Recognition of Film Genres”, Proc. ACM Multimedia 95, San Francisco, CA, Nov. 1995, pp. 295-304.

[3] D.L. Gall, “MPEG: A Video Compression Standard for Multimedia Applications”, Communications of the ACM, 34, 4, April 1991.

[4] S. L. Horowitz and T. Pavlidis, “Picture Segmentation by a Traversal Algorithm”, Comput. Graphics Image Process. 1, pp. 360-372, 1972.

[5] Rainer Lienhart, Silvia Pfeiffer, and Wolfgang Effelsberg, “The MoCA Workbench”, University of Mannheim, Computer Science Department, Technical Report TR-34-95, November 1996. (*http://www.informatik.uni-mannheim.de/pub/techreports/tr-95-034.ps.gz)*

[6] Shunji Mori, Ching Y. Suen, Kazuhiko Yamamoto, “Historical Review of OCR Research and Development”, Proceedings of the IEEE, Vol. 80, No. 7, pp. 1029-1058, July 1992.

[7] Jun Ohya, Akio Shio, and Shigeru Akamatsu, “Recognizing Characters in Scene Images”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 16, No. 2, pp. 214-220, 1994.

[8] William B. Pennebaker and Joan L. Mitchel, “JPEG Still Image Data Compression Standard”, Van Nostrand Rheinhold, New York, 1993.

[9] Arthur R. Pope, Daniel Ko, David G. Lowe, “Introduction to Vista Programming Tools”, Department of Computer Science, University of British Columbia, Vancouver.

[10] Arthur R. Pope and David G. Lowe, “Vista: A Software Environment for Computer Vision Research”, Department of Computer Science, University of British Columbia, Vancouver.

[11] Alois Regl, “Methods of Automatic Character Recognition”, Ph. D. thesis, Johannes Kepler University Linz, Wien 1986 (in German).

[12] Michael A. Smith and Takeo Kanade, “Video Skimming for Quick Browsing Based on Audio and Image Characterization”, Carnegie Mellon University, Technical Report CMU-CS-95-186, July 1995.

[13] M. Takatoo et al., “Gray Scale Image Processing Technology Applied to Vehicle License Number Recognition System”, in Proc. Int. Workshop Industrial Applications of Machine Vision and Machine Intelligence, pp. 76-79, 1987.

[14] A. Murat Tekalp, “Digital Video Processing”, Prentice Hall Signal Processing Series, ISBN 0-13-190075-7, 1995.

[15] Ramin Zabih, Justin Miller, and Kevin Mai, “A Feature-Based Algorithm for Detecting and Classifying Scene Breaks”, Proc. ACM Multimedia 95, San Francisco, CA, pp. 189-200, Nov. 1995.

[16] H. J. Zhang, C. Y. Low, S. W. Smoliar, and J. H. Wu, “Video Parsing, Retrieval and Browsing: An Integrated and Content-Based Solution”, Proc. ACM Multimedia 95, San Francisco, CA, pp. 15-24, Nov. 1995.

[17] Hong Jiang Zhang and Stephen W. Smoliar, “Developing Power Tools for Video Indexing and Retrieval”, Proc. SPIE Conf. on Storage and Retrieval for Image and Video Databases, San Jose, pp. 140-149, CA, 1994.