ON THE SEGMENTATION OF TEXT IN VIDEOS

Axel Wernicke and Rainer Lienhart

Microprocessor Research Lab, Intel Corporation
2200 Mission College Blvd.
Santa Clara, CA 95052-8119, USA
Rainer.Lienhart@intel.com

ABSTRACT

A new and robust multi-resolution approach of localizing and segmenting text in videos is proposed. The approach has been tested extensively on a large variety of video frame sizes such 352x240 up to 1920x1280 and a large representative set of video sequences such as home videos, newscasts, title sequences and commercials. 95% of the text bounding boxes in videos were localized correctly. 80% of all characters were segmented correctly, while 7.8% characters were damaged. 90% of the correctly segmented characters were recognized correctly by a standard OCR software.

1. INTRODUCTION

The localization, segmentation and recognition of text in videos enables many useful applications. For instance, recognized text can be used to search for specific video sequences such as jumping to news stories about a specific topic, since captions in newscasts often provide a condensation of the underlying news story. Or it can be used to identify commercials based on product names. It may also be used to enable future object-based video encoding of standard video if the text segmentation step is able to determine all text pixels of a text line over time such as our approach can do.

RELATED WORK. For an exhausting treatment of related work see *www.lienhart.de/VCAHome/Research_Topics/research_topics.html*

CONTRIBUTIONS. The main contributions of this paper are:

- A truly multi-resolution approach, working from MPEG-1 up to HDTV MPEG-2 video sequences (1980x1280) without any parameter adjustment. Character sizes can vary between 8 pixels and half the frame height. Only [5] and [9] address the problem of multi-resolution. However, they are still limited to some character size limits which is not true for our approach.
- Unlike all existing work, our approach is capable of estimating the text color reliably by using vector quantization. [7] just assumes that over time text is brighter than anything else.
- A new scheme to perfectly register moving text lines over time. This allows use of the temporal segmentation scheme first proposed by [7]. All other previous work reporting text tracking such as [3,4,5] do it more on a qualitative basis in order to identify false alarms. Text is not tracked with pixel accuracy.
- Unlike all existing work, detected text lines are scaled to an aspect-ratio preserving fixed height of 100 pixels during text segmentation. This scaling improves the segmentation of smaller font sizes and reduces complexity of larger ones.

2. TEXT LOCALIZATION

The text localization step is supposed to find the locations of text in individual video frames or images and mark them by tight text bounding boxes. These bounding boxes should only consist of one text line of a text column (see Figure 1 for an overview).

2.1 Image Feature

Artificial text occurrences have been commonly characterized in the research community as regions of high contrast and high frequency [4,7]. There are many different ways to amplify this feature. We chose to use the directional as well as the overall edge strength images as our feature images for text localization [2]:

$$E_x = \sum_{b=1}^{B} \left| D_x^b \right| \quad , \quad E_y = \sum_{b=1}^{B} \left| D_y^b \right| \quad \text{and} \quad E = 1/B \cdot \sum_{b=1}^{B} \sqrt{(D_x^b)^2 + (D_y^b)^2} \, .$$

D_x^b and D_y^b denote the directional derivation of image band b.

2.2 Fixed Scale Text Detector

The fixed scale text detector is supposed to classify each pixel in the overall edge strength image *E* based on its local neighborhood whether it is part of a text region with character heights of 8 to 12 pixels. There are many different techniques for developing a classifier, however, we decided to use a neural feed-forward network (NN) due to its proven good generalization capability.

The input layer consists of 20x10 neurons fed by the image region centered around the pixel under consideration. It is connected to 50 neurons in one hidden layer. The hidden layer is aggregated into one output neuron. The network was trained with the back propagation algorithm, using a 'bootstrap' method [8]. This method starts with a set of random patterns to initialize the non-text pattern set. Then the NN is trained and evaluated. Some of the falsely classified non-text patterns of the validation set are added to the training set and a new NN is trained and evaluated. This cycle of training, evaluation and adding new non-text patterns is repeated until the number of falsely classified patterns in the validation set does not decrease anymore. Our network needed seven training cycles. The final training set consisted of 374 text and 4601 non-text samples.

The trained NN is used to classify an input feature image *E*. A 20x10 pixel window slides over *E* from left to right and top to bottom, evaluating the network at each position and collecting the response of the NN in a so-called response image. If the output of the NN exceeds $th_{nn}=0.85$, a box of 20x10 filled by the NN's output value is added to the associated position in the response image. Since a step size of one is computationally prohibitive for large images or frames, we used a step factor of 3/2 in *x/y* direction. Experiments have shown, that this subsampling causes no decrease in accuracy but reduces computational complexity by 83%.

2.3 Scale Integration

The raw fixed-scale text detection results at all scales must be integrated into one saliency map of text in order to construct initial text bounding boxes. As you can observe from Figure 1 column 4, text locations identify themselves as correct hits at multiple scales, while false alarms appear less consistently over multiple scales. Similar experience have been observed by Rowley et. al for their NN-based face detector [6]. Therefore, a salience map is created by

projection the confidence of being text (here: the NN output) back to the original scale of the image. Hereto, the salience map is initialized by zero. For each detected bounding box at each scale its confidence value of being text is added to the saliency map over the size of the bounding box at the original image scale. Figure 1 column 5 gives an example.

2.4 Extraction of Text Bounding Boxes

INITIAL TEXT BOUNDING BOXES. The algorithm starts with searching for the next not yet processed pixel in the saliency map with a value larger then th_{core}=5.0 . The choice of the threshold's value is determined by the goal to avoid the creation of text boxes for non-text regions. Non-text regions should be less salient. Once such a pixel is found, it is taken as a seed for a new text box of height and width 1. This new text box is then expanded iteratively.

The average intensity of the pixels of the adjacent row above the total width of the box in the overall edge strength image is taken as the criterion for growing in that direction. If the average intensity is larger than th_{region}=4.5 , the row is added to the box. Next, the same criterion is used to expand the box to the left, bottom, and right. This iterative box expansion repeats as long as the bounding box keeps growing.

REVISED TEXT BOUNDING BOXES. The initial bounding boxes often do not optimally frame the text in the image: Some boxes contain no text; others span more than one line and/or column of text, and in many the background make up a large portion of the pixels. Fortunately, these shortcomings can be overcome by an iterative post-processing procedure utilizing the information contained in so-called projection profiles [7].

A horizontal/vertical projection profile of an image region is defined as the vector of the sums of the pixel intensities over each column/row. Upper/lower boundaries of text lines can be identified by steep rises/falls in the vertical projection profile. Similarly, the right and left boundaries of text objects are indicated by steep rises and falls in the horizontal projection profile. These steep rises and falls are identified as locations where the profile graph crosses th_{text}=.82*$min_{profile}$+.18*$max_{profile}$, where $max_{profile}$ / $min_{profile}$ are the maximum/minimum values in the profile. The factor of 0.18 was chosen experimentally. Every line with a vertical profile value exceeding th_{text} is classified as containing text.

Similarly, a horizontal segmentation algorithm is applied to ensure that text in one line which does not belong together is separated. However, there are two minor but important differences:

1. A factor of 0.25 instead of 0.18 is used. Experimentally this value has proven to be better for the horizontal segmentation.
2. Individual words in the same column should not be split up due to small gaps between them. Therefore, if a gap between two contiguous pairs of down-up and up-down transitions is smaller than th_{gap}, the two transitions in the middle are ignored.

Many times one pass of vertical and horizontal segmentation cannot resolve complex layouts. Thus, a few cycles of vertical and horizontal segmentations are applied to each text box.

Next, boxes with *height*<8*pixels* or *height*>$\text{image}_{height}/2$ are regarded as non-text regions and thus discarded. Boxes with *height*>*width* are also discarded, since horizontal segmentation assures that each text box contains word(s) of only one text line. Finally, text boxes with the same upper and lower boundaries or which touch or overlap each other are joined into one text box.

TEXT AND BACKGROUND COLOR. Estimates of the text and background color for each text box are needed later to determine whether a text bounding box contains normal (i.e., dark text on bright background) or inverse text (i.e., bright text on dark background). Given that images are colorful and that even a visually single-colored region like a character in a video frame consists of pixels of many different but similar colors, the complexity of the color distribution in each text bounding box is reduced by quantizing the colors to the four most dominating colors using the fast vector quantizer proposed by Wu [10]. Next, two color histograms are calculated: One describing the four center rows of the text box and another one describing two rows directly above and underneath the text box (four rows together). The latter histogram describes an image region that contains no or only little text while the first histogram should be dominated by the text color. Taking the difference between the first and second histogram, the maximum of the difference histogram is very likely to correspond to the text color and the minimum to the dominating background color. This methodology has proved experimentally to be very reliable for homogeneously colored text. It may fail for multi-colored text which, however, is rare. We assume normal text, if the grayscale value of the text color is lower than the one of the dominant background color, otherwise inverse text.

3. INFORMATION REDUNDANCY

Video distinguishes itself from images by temporal redundancy. Each text line appears over several contiguous frames. This temporal redundancy can be exploited to

- increase the chance of localizing text since the same text may appear under varying conditions from frame to frame,
- remove false text alarms in individual frames since they are usually not stable throughout time,
- interpolate the locations of 'accidentally' missed text lines in individual frames, and
- enhance text segmentation by bitmap integration over time.

Complete text objects, which describe text lines over time by their text bitmaps, sizes and positions in the various frames as well as their temporal range of occurrence, are extracted in a two-stage process in order to reduce computational complexity.

3.1 Stage 1: Video Monitoring For Text Occurrences

Video is monitored for text occurrences at a coarse temporal resolution. For this purpose, the image-based text localizer is only applied to an evenly spaced frame subset of the video. The maximum possible step size is given by the minimum assumed temporal duration of text lines occurrences, which we assume to be one second. Thus, it seems reasonable to assume that text should appear clearly for at least 2/3 of a second in order to be easily readable.

If the image-based text localizer does not find any text line in $frame_t$, the monitor process continues with $frame_{t+20}$. If, however, at least one text line is found, the image-based text localizer will be applied to $frame_{t-1}$ and $frame_{t+1}$. Next, for each text line

in $frame_t$ the algorithm searches for a corresponding text line in $frame_{t-1}$ and $frame_{t+1}$. Correspondence between to text lines is defined as an area overlap of at least 80% of their respective bounding boxes at their frame locations. If corresponding boxes in $frame_{t-1}$ and $frame_{t+1}$ are found for a text box in $frame_t$, a new text object comprising these text boxes is created and marked for tracking in time.

3.2 Stage 2: Fast and Precise Text Tracking

Each text object must now be extended to all frames containing the respective text line based on the information contained in the text objects created in the video monitoring stage. Obviously, text tracking must be performed backwards and forwards in time. However, we restrict our description to forward tracking only since backward tracking does not differ from forward tracking except in the direction you go through the video.

The basic idea behind our fast text tracker is to take the text line in the current video frame, calculate a characteristic signature which allows us to distinguish this text line from text lines with other contents and search for the image region of same dimension in the next video frame which best matches the reference signature.

The vertical and horizontal projection profile serve as a compact and characteristic reference signature, and the center of a signature is defined as the center of the associated text bounding box. Similarity between two signatures is measured by signature intersection, i.e., by the sum of the minimum between respective elements in the signatures. To find the precise position of a text line in the next frame, all signatures whose centers fall into a search window around the center of the reference signature, are calculated and compared to the reference signature. If the best match exceeds a minimum required similarity, the text line is declared to be found and added to the text object. If the best match does not exceed a minimum required similarity, a signature-based drop-out is declared. The size of the search radius depends on the maximum assumed velocity of text. In our experiments we assumed that text needs at least 2 seconds to move from left to right in the video.

The signature-based text line search can track zooming text only over a very short period of time. To overcome these limitations, the signature-based search is replaced every 5-th frame by the image-based text localizer in order to re-calibrate locations and sizes of the text lines. Newly detected text boxes, however, are not considered here.

Due to imperfection in the video signal continuous recognition of text objects in every frame is often not possible. Therefore two thresholds $max_{DropOut}^{signature\text{-}based}=4$ and $max_{DropOut}^{image\text{-}based}=3$ are defined. Whenever a text object cannot be extended to the next frame, the respective counter is incremented by one. The respective counter is reset to zero whenever its related search method succeeds. The tracking process is aborted, as soon as one of both counters exceeds its respective threshold.

3.3 Postprocessing

To prepare a text object for text segmentation, it should be trimmed down to the part which has been detected with high confidence. Therefore, each text object is temporally trimmed down to the first and last frame in which the image-based text localizer detected the text line. Next, all text objects are discarded which occur less than a second or show a drop-out rate of more than 25%. A few global features are determined for each text object:

1. Text color: The text color of a text object is determined as the median of all determined text colors per frame.
2. Text size: If the size of the text bounding box is fixed, we determine its width and height by means of the median over the set of widths and heights.
3. Text position: A text line is regarded as static in the x and/or y direction if the average movement per frame is less than 0.75 pixels. If the text line is static, we replace all text bounding boxes by the median text bounding box. The median text bounding box is the box which left/right/top/bottom border is the median over all left/right/top/bottom borders. If the position is only fixed in one direction, the left and right or the top and bottom are replaced by the median value, respectively.

4. TEXT SEGMENTATION

4.1 Resolution Adjustment

All subsequent text segmentation steps are performed on text box bitmaps rescaled by cubic interpolation to a fixed height of 100 pixels, while preserving the aspect ratio for two reasons:

1. Resolution enhancements of small font sizes for better segmentation results since it a) enables sub-pixel precise text alignment for small text occurrences and b) the usage of standard OCR software for recognition.
2. Computational savings for large font sizes (e.g. for HDTV video sequences at 1920x1280). A text height larger than 100 pixels does not improve segmentation nor OCR performance.

4.2 Removing Complex Backgrounds

The temporal redundancy is exploited to remove complex backgrounds surrounding the actual characters. The method applied here was first proposed by [7] for static text. In our work, it is also applied to moving text since we solved the problem of sub-pixel accurate text alignment. The basic idea works as follows: A text object's bitmaps are piled up such that the characters are aligned perfectly with each other. Looking though a specific pixel in time, you may notice that pixels belonging to text vary only slightly, while background pixels often change vastly through time. Since the text's location is static due to its alignment its pixels are not supposed to change. Background pixels are likely to change due to motion in the background or motion of the text line.

Given the pile of perfectly aligned bitmaps, the maximum/minimum operator is applied through time on the grayscale images for normal/inverse text [7]. The only serious problem we have to solve is the perfect alignment of the text bitmaps: All bounding text boxes of a text object are extended horizontally by 20% and vertically by 40%. Next, all bitmaps are converted to grayscale since grayscale is not vulnerable to color compression artifacts. Let $B_0(x,y)$, ..., $B_T(x,y)$ denote the $T+1$ bitmaps under consideration and $B^r(x,y)$ the representative bitmap which is to be derived and initialized to $B_0^r(x,y)=B_0(x,y)$. Then, for each bitmap $B_i(x,y)$, $i \in \{1,\ldots,T\}$, we search for the best displacement (dx,dy) which minimizes the difference between $B^r(x,y)$ and $B_i(x,y)$ with

respect to the text color, i.e.,

$$(dx_t^{opt}, dy_t^{opt}) = \arg\min \sum_{\substack{(x,y)\in B^r \\ B^r_{i-1}(x,y)\subseteq textColor}} \left| B^r_{i-1}(x,y) - B_i(x+dx, y+dy) \right|$$

This kind of block matching search works because only pixels with text color are taken into account. A pixel is defined to have text color if it does not differ more than a certain amount from the text color determined for the text object. The color distance is calculated based on the RGB values.At each iteration $B^r(x,y)$ is updated to $B^r_i(x,y) = max(B^r_{i-1}(x,y), B_i(x+dx_t^{opt}, y+dy_t^{opt}))$ for normal text. For inverse text *max* is replaced by *min*. If a text object has been identified to be static in Section 2.4, we do not have to search for the perfect translations. Instead, the translations between the various bitmaps are all set to zero.

Next, each pixel on the boundary of a text bounding box is taken as a seed to fill all pixels with the background color (black for inverse text and white for normal text) which do not differ more than $th_{seedfill}$ from the seed color. The seed fill algorithm uses a 4-neighborhood [1]. Since the pixels on the boundary do not belong to the text and the text contrasts with its background, the seed-fill algorithm will never remove any character pixel. We call this newly constructed bitmap $B^r(x,y)$ again.

This procedure might not delete all background pixels, Therefore, each non-background pixel is taken as a seed pixel for a 8-neighborhood seed-fill. The algorithm is only applied hypothetically to $B^r(x,y)$ in order to determine the dimension of the region that could be filled. All hypothetical regions with a height less than min_{height} pixels and a width less than min_{width} or larger than max_{width} pixels are set to the background color.

4.3 Binarization

The text bitmap $B^r(x,y)$ is now prepared for recognition by standard OCR software. Hereto, the grayscale text bitmaps must be converted to black on white background. From Section 2.4 we know the text color and whether we have to deal with normal or inverse text. A good binarization threshold is the average between the intensity of the text and the background color. Each pixel in the text bitmap which is higher than the binarization threshold is set to white for normal text and black for inverse text. Each pixel in the text bitmap which is lower or equal than the binarization threshold is set to black for normal text and white for inverse text. Finally, it is recommended to clean-up the binary bitmap by discarding small regions.

5. RESULTS

The algorithms have been tested extensively on a large variety of video sizes (from 352x240 up to 1920x1280) and a large representative and difficult set of video sequences such as home videos, newscasts, title sequences and commercials (10 minutes together). The text detection system missed approximately 38 of all text boxes containing 15% of all text pixel. The localization performance could be boosted up to 94.7% by exploiting the temporal redundancy in video sequences. 80% of all characters were segmented correctly and 7.8% characters were damaged. 90% of the correctly segmented characters were recognized correctly by a standard OCR software. These performance numbers are above the ones reported for existing systems.

ACKNOWLEDGEMENT. We would like to thank Boon-Lock Yeo and NSTL for the wonderful MPEG Library that made this work possible and which decodes MPEG videos incredibly fast.

References

[1] J. D. Foley, A. Dam, S. K. Feiner und J. F. Hughes. Computer Graphics: Principles and Practice. Addison-Wesley, Reading, MA, USA, 1990.

[2] Bernd Jaehne. Practical Handbook on Image Processing for Scientific Applications. CRC Press, Boca Raton, 1997.

[3] R. Lienhart. Automatic Text Recognition for Video Indexing. Proc. ACM Multimedia, Bosten, MA, Nov. 1996, pp. 11-20.

[4] R. Lienhart and W. Effelsberg. Automatic Text Segmentation and Text Recognition for Video Indexing. ACM/Springer Multimedia Systems Magazine. to appear.

[5] H. Li, D. Doermann, and O. Kia. Automatic Text Detection and Tracking in Digital Video. IEEE Trans. on Image Processing. to appear.

[6] H. A. Rowley, S. Baluja, and T. Kanade. Neural Network-Based Face Detection. IEEE PAMI, vol. 20, no. 1, pp. 23-38, January 1998.

[7] T. Sato, T. Kanade, E. K. Hughes, M. A. Smith. Video OCR for Digital News Archives. IEEE Int. Workshop on Content-Based Access of Image and Video Database, 1998.

[8] K.-K. Sung. Learning and Example Selection for Object and Pattern Detection. PhD Thesis, MIT AI Lab, January 1996.

[9] V. Wu, R. Manmatha and E. M. Riseman. Finding Text in Images. In Proc. of Second ACM International Conference on Digital Libraries, Philadelphia, PA, pp. 23-26, July 1997.

[10] X. Wu. YIQ Vector Quantization in a New Color Palette Architecture. IEEE Trans. on Image Processing, vol. 5. no. 2, p. 321-329, 1996.

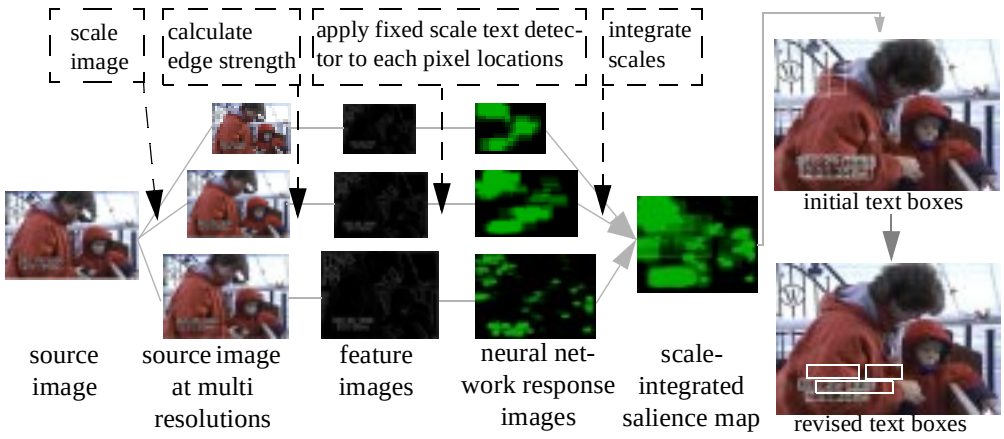

Figure 1. Overview over the text localization step