LOCAL INFORMATION BASED OVERLAID TEXT DETECTION BY CLASSIFIER FUSION

Ahmet Ekin

Video Processing Group, Philips Research, Eindhoven, The Netherlands
ahmet.ekin@philips.com

ABSTRACT

When implemented in hardware, image-processing algorithms should be robust to memory limitations because some hardware architectures may not have memory size as large as the whole frame size. Although this is not generally a problem for low-level processing, higher-level understanding, such as object detection, demands novel solutions because the available information may, in some cases, be very local, e.g., only a partial view of the object could fit in the available memory size. In this paper, we propose a novel hardware-oriented overlaid text detection algorithm that can detect text with height as large as five times the memory size. The algorithm integrates a connected component (CC)-based algorithm with a texture-based machine learning approach. The CC-based algorithm uses character-level features in the horizontal direction whereas the texture-based algorithm extracts block-based features to integrate information from all directions. Furthermore, the texture-based algorithm employs a support vector machine (SVM) to benefit from the strength of machine learning tools. In order to detect text of large font size, we also propose a novel hardware-oriented, height-preserving multi-resolution analysis. Finally, the results of the two classifiers as well as color and edge cues are used for the final pixel-based text/non-text decision.

1. INTRODUCTION

Text overlays are added onto TV broadcast to supplement the audio-visual content with additional metadata. Because of its information value, overlaid text detection has mainly been considered in the context of video indexing and retrieval. In contrast, our main application is visual quality improvement. This difference is important because most storage and retrieval applications can afford to have offline and distributed (in the sense that the task can be assigned to a non-TV device) processing whereas visual quality improvement should often be performed online by using very limited computational and memory resources of a TV set. In this context, memory size refers to the number of image lines available for processing at a single instant. In software, whole frame or video information can usually be used; but only a few lines of image data are available in hardware. In the following, we will mainly focus on the implications of the memory limitations to overlaid text detection algorithms.

In general, overlaid text detection involves two steps: 1) text candidate extraction, and 2) text verification. In the first stage, the smallest spatial processing units, such as pixels and blocks, are processed independently from each other and are assigned as text candidate or not. In the second stage, spatially connected detections are merged for region-level morphological analysis, e.g., by using region bounding box information. In one or both of these steps, the state-of-the-art text detection algorithms rely on non-local information, which can even be at the frame level. In this paper, we define the local region as lines of data supported by the hardware memory. For example, Figure 1 shows the local region for the yellow line in red. This value is 11 lines in our case. The memory size could be much less than the height of text, but the width is the same as the image width. The problem we would like to solve in this paper is to be able to robustly detect text of any size by processing only limited information available in the memory. This is a challenging problem especially when the text height is greater than the memory size as shown in Figure 1.

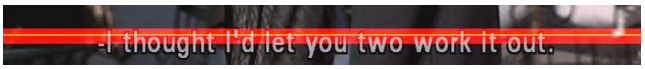

Figure 1: The local area for the line in yellow is highlighted in red (total of 11 lines); we use only the information from the local region for text detection and verification.

The existing text detection algorithms exploit color connectedness (connected component, CC-, based approaches) and/or textural features [1][2]. Recently, machine learning tools, such as neural nets [3], or SVMs [4], are also applied to determine the text/non-text boundary. In these algorithms, non-local information is needed mainly for multi-resolution image analysis (for large-sized text detection), for text segmentation (for extraction of word and text line boundaries), and finally for suppression of false alarms.

Having constraints not applicable to the existing approaches, we propose an integrated framework that employs a CC-based and a texture-based algorithm. CC-based algorithm utilizes character-based features in the horizontal direction; hence, it processes each image line independently from the others. In contrast, texture-based algorithm uses block-based features to use information from all directions. In order to take advantage of machine learning tools, an SVM with a linear kernel has been trained to make text/non-text decision for the texture-based algorithm. We also propose a novel height-preserving horizontal scaling method for multi-resolution analysis. In the verification stage, we also use color and edge features to fully utilize the existing information. As a result, the proposed algorithm is able to robustly detect text whose height can be five times bigger than the memory size.

Section 2 explains the text candidate extraction that is comprised of edge-based preprocessing, CC-based detection, and texture-based algorithm. After that, Section 3 describes the text verification stage where the individual detections are fused. Section 4 presents the experimental results while Section 5 concludes the paper.

2. TEXT CANDIDATE EXTRACTION

In this section, we first explain the edge-based preprocessing step that determines the region-of-interest for the CC- and texture-based algorithms that are explained in Sections 2.2 and 2.3, respectively.

2.1. Edge-based preprocessing

The insertion of text onto video should result in large intensity and color differences from the background so that the viewers can read the overlaid text easily. In this section, we use this feature to eliminate non-text regions to speed up the overall processing.

We first detect horizontal, $G_h(x,y)$, and vertical, $G_v(x,y)$, derivatives at each image location (x,y) by applying a *2x2* mask. After that, the edge strength, $ES(x,y)$, is computed for the pixel (x,y) as in Equation 1, where we prefer L1 norm to the Euclidean distance because of the computational reasons. The pixels having edge strength greater than the adaptively computed threshold value, $GThr_{High}$, are assumed to include text regions if there are any. The value of $GThr_{High}$ is determined as a function of the average edge strength as shown in Equation 2, where k is a constant coefficient (We found $k = 5$ as an appropriate value not to lose any pixels at the text boundaries), M and N are image width and height, respectively. Figure 2 demonstrates the output of this stage proving that strong edges should exist at the transitions between text and natural video content.

$$ES(x,y) = |G_h(x,y)| + |G_v(x,y)| \quad (1)$$

$$GThr_{High} = \frac{k}{M \times N} \sum_{x=1}^{M} \sum_{y=1}^{N} ES(x,y) \quad (2)$$

Figure 2: Strong edges (shown in green) accumulate around the text region.

2.2. CC-based character-level analysis

When analyzed at a character level, text regions show: 1) large horizontal gradients that have opposite signs at character-to-background and background-to-character transitions, and 2) a horizontally smooth single colored region within a character stroke (the color of the text character may be varying in the vertical direction). This section presents a method that exploits these observations to detect text regions by processing image intensity line-by-line.

Wong and Chen [5] proposed *maximum gradient difference* (MGD) feature to boost the first characteristics of the overlaid text. This feature computes the difference between the maximum and the minimum horizontal gradient values in a window. The key idea behind the feature is that when a window is large enough to include at least one background-to-character and one character-to-background transition, there will be a large positive and a large negative gradient. This will be valid independently from the relative brightness of the text with respect to the background, that is, text can be brighter or darker than background and MGD is invariant to this. The window length should be determined as a function of the largest expected font size. In [5], text regions are identified as those having large MGD values by comparing them with a pre-defined threshold value. However, this tends to merge the text regions with the line structures in the background. This might not have been a problem if we had access to the whole frame information; however, because of our constraints, it causes problems for the morphological analysis in the verification stage. Furthermore, the maximum and the minimum operations are sensitive to noise.

To remedy the above problems, we first modify the MGD according to the second observation, also suggested in [6], and then use relative location and magnitude of the positive and negative gradients. We integrate the second observation into MGD by requiring low gradient values at the locations between the large gradients. However, this alone will not suffice to prevent the merging of characters and line structures. Furthermore, noise-related large MGD values still survive. Therefore, we also add the location and the magnitude constraints to the MGD. The location constraint makes sure that the large positive and negative gradients occur at the opposite sides of the window center. The magnitude constraint verifies that there is not a big discrepancy between the magnitudes of the largest positive and the largest negative gradients by checking that the smaller one, in the absolute sense, is bigger than the half of the larger one. Figure 3 shows a sample result of this step where the pixels having high text likelihood are highlighted in green. Please note that only predefined neighborhood of each strong edge is processed for CC-based analysis.

Figure 3: The result of the CC-based character-level analysis is highlighted in green.

2.3. Texture-based text detection by SVM

Character-level analysis, explained in the previous section, analyzes each line independently from the vertical neighbors. As a result, some false alarms may occur at locations that do not have horizontal textures but have only strong vertical edges. To incorporate local texture information in other directions, this section introduces block-based texture analysis. The text likelihood of a block is computed by a pre-trained SVM. This type of analysis

adds the power of the machine learning tools to our algorithm. Because of the constraints about the available information and the compute-power, we also introduce some novel aspects to the texture-based text detection.

Texture-based analysis involves first extraction of a set of features in a window of fixed size, *11x11* in our case because the memory size is 11 image lines. We extract two features per pixel: the magnitudes of the horizontal and vertical gradients. When a machine-learning tool, such as an SVM, is used, a supervised training stage is needed to train the classifier with the selected features. This is done offline and only once. After the training of the classifier, when a new image is presented, the same set of derivative features are extracted and fed into the classifier that determines the text likelihood of the input feature vector. In order to detect text of varying size, multi-resolution analysis of the image, as shown in Figure 4, is needed. The results across multiple scales are integrated to decide whether a site can be text or not.

In our case, the usual multi-resolution analysis is impossible because the memory size does not allow scaling in the vertical direction. This presents a unique challenge for font-size-independent detection of text. To this effect, we propose to scale the group of lines only in the horizontal direction. This makes scaling independent of memory size. Figure 5 shows the proof of the height-preserving property of the proposed scaling method. In contrast to the usual multi-resolution as shown in Figure 4, this is a sub-optimal solution. However, our thorough analysis has shown that the proposed scaling method robustly works for text detection when the length of a word is long enough (3-4 letters). The reason for such robustness stems from the fact that the classifier identifies large variations in texture (measured by the derivatives) as text. When the text is large, the window in the original image is not big enough to capture those variations; hence, it is necessary to do the scaling. The proposed horizontal scaling fits close characters into the window so that they can be identified as text.

Figure 4: Multi-resolution view of the input image (the spatial resolution decreases 2x2 to the right).

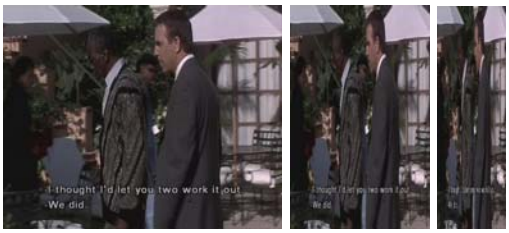

Figure 5: Height-preserving image scaling is optimal for hardware implementations (the spatial resolution decreases only in the horizontal direction).

For the detection, we use three scales as shown in Figure 5. The detection result from the first scale is accepted as text when the SVM result is greater than zero. The second and third scales are only accepted if the SVM detection result is greater than 0.5. In this way, we can detect text whose height is as large as five times the memory size. Figure 6 shows the result of texture-based detection with SVM. In order to speed up the processing, we use linear kernel for the SVM. This makes it possible to define the classification with a cross-correlation operation. Furthermore, we apply the SVM only to the strong edges that are detected in Section 2.1 to further speed up the processing.

Figure 6: The result of the SVM-based text detection algorithm

3. TEXT VERIFICATION

In the text verification stage, we aim to decide whether individual detections from character and texture analysis can be accepted as text and, at the same time, we attempt to determine the most accurate boundary of the text region. In addition to the detection results, we also use color cues and strong edge count for the final decision.

Because we have to decide locally, the steps below are followed for each group of lines (a group of lines is shown in red in Figure 1). We first check for whether there is any detection from both stages. If there is detection, we apply the following steps:

1. Reduce the color of the image to 8 levels. These levels are determined adaptively for each local region.
2. Find the regions both texture-based and character-based algorithms have classified as text.
3. Compute the start and the end points of each text region determined in the texture-based analysis. This involves morphological dilation operation to connect close but disconnected regions.
4. Determine the text color from the results of character-based detection and color segmentation in step 1. Text color is assigned as the color index that has the largest number of pixels in the text mask created in Section 2.2.
5. Starting from the center of each region found in step 3, use character-based algorithm to refine the start and the end of the selected region, compute the length of the region, and if the length is large enough (greater than 10 pixels), accept the region as text.
6. Find the regions that are identified as text by the character-based algorithm, but not supported by the SVM result. Re-run SVM over the scaled version of each such region with a lower threshold. If identified as text by the rerun of SVM, accept each such region as text.

The above verification stage uses our observation that character-based algorithm results in more accurate text boundaries whereas texture-based algorithm is more effective in the reduction of false alarms.

4. RESULTS

In order to evaluate the proposed algorithm, we have used data downloaded from the Open Video Project web page [7] and captured within Philips. The videos from [7] are mostly of 352x240 size whereas those in the Philips set are of size 720x576. The Open Video Project set has compression artifacts; however, Figure 7, where the detected text regions are highlighted in green, shows the robustness of the proposed algorithm to such artifacts as well as variations in background. Figure 8 shows some detection results with few false positives in highly textured areas. The result on the right in Figure 8 demonstrates that the scene text can also be detected with the proposed algorithm. Figure 9 is an example to the detection of large-sized text. The text segments "Markte" and Morgen" in Figure 9 are more than five times larger than the available memory size for processing. The detection of text with large font size is made possible by the height-preserving multi-resolution analysis explained in Section 2.3 and exemplified in Figure 5.

We quantified the performance of the algorithm as a function of text height (in pixels) when the memory size is 11 lines. For this, we define the TFM measure as the absolute count of true detections-false alarms-missed text regions. We also report the performance in precision and recall measures, defined in Equation 3. For text having height smaller than 12, the TFM rate is 99-9-7 (91.6% precision, 93.4% recall); for text having height between 12 and 25, the TFM rate is 213-15-22 (93.4% precision, 90.6% recall); and for text having height less than 35, the TFM rate is 89-4-14 (95.6% precision, 86.4% recall). As text height increases, the precision rate improves at the expense of a decrease in the recall rate. The proposed algorithm was able to detect 12 regions that have text height larger than 60 pixels without any false alarms. The main reasons for the missed text are small width of the text block, such as single digits, and low contrast between the background and the text. The false alarms have mainly resulted from the textured background.

$$\textit{Precision} = (\text{the \# of correct texts}=T) / (\text{the \# of detections}= T+F)$$
$$\textit{Recall} = (\text{the \# of correct texts}=T) / (\text{the \# of texts} = T+M) \quad (3)$$

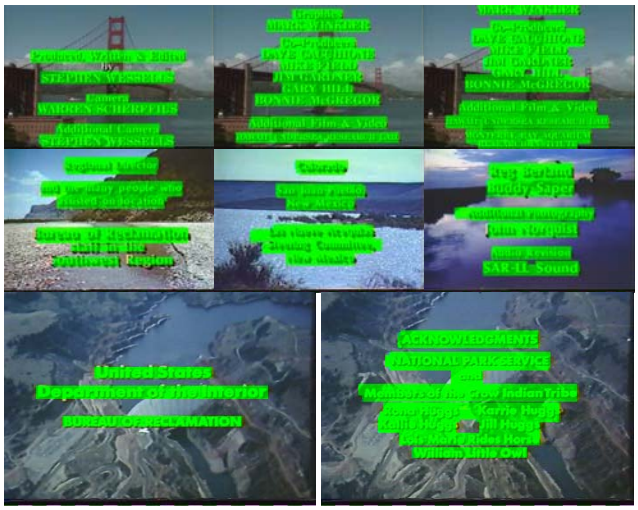

Figure 7: Examples of detections of text on natural scenes (Data are from the Open Video Project [7]).

Figure 8: Examples from the Philips set; scene text can also be detected (right), occasional false alarms in textured areas (left)

Figure 9: Text of large size is still detectable; the height of text is more than five times larger than the memory size.

5. CONCLUSIONS

In this paper, we proposed an integrated, hardware-oriented text detection algorithm. Among other things, the main contribution of this paper is the robust detection of text with very limited local information. To achieve this, we introduced an integrated algorithm that exploits both character-level and block-based text features. The proposed algorithm can handle large-sized text as long as the text height is not larger than five or six times the number of available image lines for processing. The proposed algorithm shares some of the weaknesses of the existing algorithms. Textured background may sometimes cause false alarms while characters with single or two letters and the text regions having little contrast may be missed.

6. REFERENCES

[1] K. Jung, K.I. Kim, A.K. Jain, "Text information extraction in images and video: a survey," *Pattern Recognition,* 37:977-997, 2004.

[2] R. Lienhart, "Video OCR: A Survey and Practitioner's Guide," In *Video Mining*, Kluwer Academic Publisher, pp. 155-184, Oct. 2003.

[3] R. Lienhart and A. Wernick, "Localizing and Segmenting Text in Images, Videos and Web Pages," *IEEE Trans. on CSVT,* 12:4:256 -268, Apr. 2002.

[4] K.C. Kim, H.R. Byun, Y.J. Song, Y.W. Choi, S.Y. Chi, K.K. Kim, Y.K. Chung, "Scene text extraction in natural scene images using hierarchical feature," In *Proc. IEEE ICPR,* 2005.

[5] E. K. Wong and M. Chen, "A new robust algorithm for video text extraction," *Pattern Recognition,* 36:1397-1406, 2003.

[6] Personal communication with Paola Carrai, Philips, Monza, Italy, Jan. 2004.

[7] The Open Video Project, Online, http://www.open-video.org/