2011 International Conference on Document Analysis and Recognition

A Gradient Vector Flow-Based Method for Video Character Segmentation

Trung Quy Phan, Palaiahnakote Shivakumara, Bolan Su and Chew Lim Tan
School of Computing, National University of Singapore
{phanquyt, shiva, subolan, tancl}@comp.nus.edu.sg

***Abstract*—In this paper, we propose a method based on gradient vector flow for video character segmentation. By formulating character segmentation as a minimum cost path finding problem, the proposed method allows curved segmentation paths and thus it is able to segment overlapping characters and touching characters due to low contrast and complex background. Gradient vector flow is used in a new way to identify candidate cut pixels. A two-pass path finding algorithm is then applied where the forward direction helps to locate potential cuts and the backward direction serves to remove the false cuts, i.e. those that go through the characters, while retaining the true cuts. Experimental results show that the proposed method outperforms an existing method on multi-oriented English and Chinese video text lines. The proposed method also helps to improve binarization results, which lead to a better character recognition rate.**

Keywords-Video character segmentation; Curved segmentation path; Gradient vector flow; Minimum cost path finding

I. INTRODUCTION

With the proliferation of videos on the Internet, there is an increasing demand for search and retrieval. In addition to image features, e.g. colors and shapes, semantic features, e.g. text, play an important role in video retrieval. Both caption text and scene text (which appears on billboards, road signs and so on) can be used to improve the retrieval of relevant videos, or even relevant frames.

According to [1], a video text detection and recognition system consists of five steps: (1) Detection, (2) Localization, (3) Tracking, (4) Extraction and enhancement, and (5) Recognition. The first three steps focus on detecting and localizing text lines in video frames. The text lines' locations are usually represented by their rectangular bounding boxes. However, these boxes still contain both text and background pixels so the fourth step aims to make text easier to recognize, e.g. by binarization. The fifth and final step typically uses an optical character recognition (OCR) engine to produce the final output as a string of characters.

OCR engines work well for document images, most of which contain monochrome text on a plain background. However, it does not produce satisfactory results for video images due to the poor resolution, low contrast and unconstrained background of the text lines. Moreover, scene text is affected by lighting conditions and perspective distortions. Hence, in this paper, we use our text detection method [2] to get multi-oriented text lines in a video frame and focus on the fourth step, extraction and enhancement, to improve the performance of OCR for video text.

In particular, we propose a method for video character segmentation, i.e. splitting a detected text line into individual character images. The motivation is two-fold. First, it is an important step if a custom-built OCR with its own feature extraction scheme, e.g. [3], is used instead of a commercial OCR. Second, even if a traditional OCR engine is used, this step can still help to improve the recognition rate by performing enhancement methods, e.g. binarization, on individual character images instead of on the whole text line.

Character segmentation is a well-known problem in document analysis, especially for handling touching handwritten characters. Casey and Lecolinet [4] provided a comprehensive literature survey of character segmentation methods for document images. There are three main approaches mentioned in the paper: *dissection*, the decomposition of a text line image into individual character images, *recognition-based*, the use of recognition results to provide feedback for segmentation, and *holistic*, the direct recognition of words (without segmentation) through matching features such as ascenders and descenders.

Many of the methods mentioned in the above paper were designed solely for document images and thus rely on connected component analysis. However, this is not suitable for video images because text pixels cannot be reliably extracted as complete connected components due to the low contrast and complex background of the text lines. Therefore, a number of methods have been proposed for video characters and most of them belong to the first approach. The second approach is not common because video character recognition itself is a challenging problem, while the third approach is limited to predefined lexicons.

A common video character segmentation method is projection profile analysis [5]. Edge information (or other kinds of energy) in each column was analyzed to distinguish between columns that contain text pixels and gap columns, based on the observation that the former had higher energy compared to the latter. Heuristic rules were proposed to further split and merge the segmented regions based on assumptions about the characters' widths and heights [6], [7]. Although these methods are simple and fast, it is difficult to determine a good threshold for images of different contrast (Fig. 1). In addition, because they work based on columns, they can only produce vertical cuts.

To overcome this problem, a number of papers, inspired by works on touching handwritten characters, modeled the segmentation problem as a minimum cost path finding problem. Kopf, Haenselmann and Effelsberg [8] used Dijkstra's algorithm to perform path finding from the top row to the bottom row of the input image. A path's cost was

1520-5363/11 $26.00 © 2011 IEEE
DOI 10.1109/ICDAR.2011.207

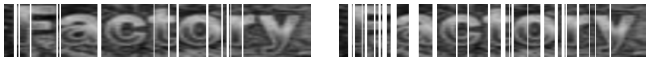


Figure 1. The results of projection profile analysis are sensitive to threshold values. With a high threshold, true gaps are missed (left), while with a low threshold, many false gaps are detected (right).

defined as the cumulative absolute difference in grayscale intensities between consecutive pixels, based on the assumption that the background region had little variation in intensity. This method may not work well for images with complex backgrounds. In a similar approach, Tse, Jones, Curtis and Yfantis [9] applied path finding recursively until the segmented regions met the stopping criteria, e.g. their widths were below a threshold. However, this method requires binarization to get connected components, which is extremely difficult to do reliably, as aforementioned.

Different from the previous methods, Saidane and Garcia [10] used a machine learning technique, convolutional neural networks, for segmentation. The input was the three images corresponding to three color channels of the text line image. The output was a vector which classified whether each column was a border between consecutive characters. This method requires training data and thus may not generalize well to different datasets. It also allows only vertical cuts.

In this paper, we propose a new method for video character segmentation by formulating the problem as a path finding problem. The cost function employs gradient vector flow (GVF) to define the criteria of a good path based on the gradient information. Our method has two advantages over existing methods in the literature. First, it allows curved segmentation paths, which are traditionally only required for touching handwritten characters. However, they are also important for video characters because touching characters can occur due to the poor resolution, the low contrast, and the variety of fonts of the text lines, including stylized text used in commercials. Second, our method does not require many thresholds for classifying gap columns (as in methods based on projection profile analysis) or extracting complete connected components from text line images (as in document analysis methods and binarization methods).

Finally, it should be noted that in the literature, there is another school of thought that leaves character segmentation to OCR engines and instead focuses on extracting character pixels more accurately, e.g. [11], [12], [13]. Hence, in the experiment section, we show that our method can, in fact, help to improve the binarization results.

II. Proposed Approach

We use our text detection method [2], which is capable of detecting both horizontal and non-horizontal text, to extract the text lines from a video frame. They then become the input images of the proposed character segmentation method, which consists of three steps: cut candidate identification, minimum cost path finding and false positive elimination. The first step employs GVF to identify pixels that are potentially part of non-vertical cuts. In the second step, we find multiple least cost paths from the top row to the bottom row of the image. The third step helps to remove false cuts that go through the middle of the characters.

A simple preprocessing step is also performed on the detected text lines: they are rotated back to horizontal orientation, if they are non-horizontal, and normalized to a fixed height of 128 pixels because some lines are too small to be readable at their original sizes, e.g. 10-pixel height.

A. Cut Candidate Identification

GVF [14] is a popular method that is often used together with active contour [15] for non-rigid registration and motion tracking. With the normal gradient, there is only information at the edges, and not in homogenous regions. GVF helps to overcome this problem by propagating the gradient information, i.e. the magnitude and the direction, into homogenous regions. As a result, there are enough forces to attract the snake into concave regions. The propagation is done by minimizing the following energy functional:

$$\varepsilon = \iint \mu\left(u_x^2 + u_y^2 + v_x^2 + v_y^2\right) + |\nabla f|^2 |g - \nabla f^2|\, dxdy \quad (1)$$

where $g(x,y) = \big(u(x,y), v(x,y)\big)$ is the GVF field and $f(x,y)$ is the edge map of the input image [14].

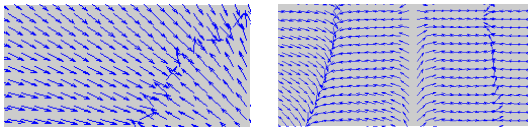
In this paper, we propose using GVF in a novel way for character segmentation (instead of for registration or tracking). A gap between two characters can be thought of as a collection of points that lie in the middle of two edges, one from the character on the left hand side and the other from the character on the right hand side. Within a gap, there is more than one segmentation path that can separate the two characters. One way to define a good path is that it should stay as far as possible from the two character edges to allow room for errors if the edge information is not accurate or the character contours are partly broken due to low contrast.

With this motivation, we use the GVF field to identify candidate cut pixels. It is observed that for edges, there are often two "arrows" (gradient directions) pointing towards each other while for gaps, the arrows usually point away from each other (Fig. 2). This implies that on the left hand side of a gap, the pixels are closer to the character on the left and thus attracted to that side, and similarly for the right hand side. Because the gap pixel is equally far from both characters, it satisfies the criterion mentioned above. Therefore, pixel (x, y) is a candidate cut pixel if and only if:

$$\begin{cases} u(x,y) < 0 \\ u(x+1,y) > 0 \\ angle\big(g(x,y), g(x+1,y)\big) > \theta_{min} \end{cases} \quad (2)$$

where *angle*(.) returns the angle between two vectors. In other words, the GVF vector at pixel (x, y) should point to the left hand side, the GVF vector at pixel (x + 1, y) should point to the right hand side and the angle between these two vectors should be sufficiently large, e.g. 15 degrees.

Fig. 3 shows the candidate cut pixels of a text line with complex background. GVF is able to detect pixels in the gaps between consecutive characters. Although these pixels do not form complete cuts yet, they play an important role in the path finding process, which is described in the next section, where the segmentation paths are encouraged to go



(a) Edge (b) Gap between two edges

Figure 2. GVF fields around an edge and a gap.

through these pixels instead of other pixels in the same gap.

A side effect of (2) is that it also captures "medial" pixels, i.e. those that are in the middle of the character strokes (Fig. 3). However, it is still possible to distinguish between candidate cut pixels and medial pixels. Since medial pixels are part of a character, if a segmentation path wants to go through these pixels, it has to make several background-to-character and character-to-background transitions. This is not the case for candidate cut pixels because the segmentation path would only stay in the background.

In the next section, we explain how the cost function is designed to both encourage paths going through candidate cut pixels and discourage those going through medial pixels.

B. Minimum Cost Path Finding

Inspired by a method for segmenting merged characters in document images [16], we formulate character segmentation as a minimum cost path finding problem where from the top row, it costs less to go through a gap and reach the bottom row than cutting through a character.

The input image can be considered as a graph where the vertices are the pixels, and pixel (x, y) is connected to neighboring pixels in the left-down, down and right-down directions, i.e. pixels $(x - 1, y + 1)$, $(x, y + 1)$ and $(x + 1, y + 1)$. The minimum cost paths are found by dynamic programming as follows.

Let $I(x, y)$ be the grayscale input image, p_0 be a starting pixel on the top row, $c(p_1, p_2)$ be the cost of moving from pixel p_1 to pixel p_2, and $d(p)$ be the cumulative cost of the minimum cost path from pixel p_0 to pixel p.

Initialization:

$$d(p) = \begin{cases} 0, & if\ p = p_0 \\ +\infty, & otherwise \end{cases} \quad (3)$$

Update rule:

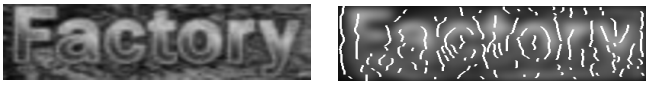
$$d(p) = min \begin{cases} d(p_{left-up}) + c(p_{left-up}, p) \\ d(p_{up}) + c(p_{up}, p) \\ d(p_{right-up}) + c(p_{right-up}, p) \end{cases} \quad (4)$$

where $p_{left\text{-}up} = (p.x - 1, p.y - 1)$, $p_{up} = (p.x, p.y - 1)$ and $p_{right\text{-}up} = (p.x + 1, p.y - 1)$. The cost function is defined as:

$$c(p_1, p_2) = \begin{cases} 0 & if\ candidate(p_2) \\ (I(p_1) - I(p_2))^2 & if\ p_1.x = p_2.x \\ k \times (I(p_1) - I(p_2))^2 & otherwise \end{cases} \quad (5)$$

where $candidate(p)$ returns true if p is a candidate cut pixel and k is the diagonal move penalty (to be explained later).

As previously mentioned, the cost function is designed to encourage paths that go through candidate cut pixels. It is thus set to be zero at these pixels. For other pixels, the cost function is set to the squared difference between two gray intensities because we assume that for text to be readable, there should be some contrast between the characters and the background. (We use the squared difference to penalize large differences more, instead of penalizing the differences linearly.) A large difference may indicate transitions between the background and the characters, i.e. cutting through the characters. Therefore, paths that go through medial pixels are discouraged by this cost function.



(a) Input (b) Candidate cut pixels

Figure 3. Candidate cut pixels of a sample image. In (b), the image is blurred to make the (white) cut pixels more visible.

Curved segmentation paths are naturally allowed. However, in many cases, vertical paths are sufficient so k is set to $\sqrt{2}$ to avoid paths with excessive curvature.

An advantage of the proposed cost function is that it works directly on grayscale images and does not require binarization like many document analysis methods. In addition, by using the squared difference, it is able to handle text of different polarities, i.e. both bright and dark text.

Note that the above algorithm finds the best path for only one starting point on the top row. To segment all the characters, we run it multiple times with different starting points. Ideally, we only need to put a starting point every w pixels where w is the estimated character width (based on the height of the input image). However, because the characters have variable widths, e.g. 'i' versus 'm', and furthermore, the gaps between the words may not be a multiple of w, more frequent starting points are required. In our implementation, a starting point is placed every $w / 4$ pixels.

C. False Positive Elimination

In the previous section, the cost function is carefully designed to discourage segmentation paths that cut through the characters. However, these false cuts may still occur for various reasons, e.g. low contrast which leads to a small difference in grayscale intensities of two consecutive pixels on the path. In this step, we aim to remove these false cuts.

It is interesting to observe that if there are more starting points than required in a gap, the minimum cost paths usually converge to the same end point (Fig. 4a). This suggests that end points are more reliable than the starting points, especially because the latter are placed according to a heuristic rule based on the estimated character width.

In order to verify whether a segmentation path is a true cut or a false cut (going through a character), we perform backward path finding from the end points to the top row (similar to forward path finding, except that the directions of the edges are reversed). For true cuts, it is likely that the forward path and the backward path are close to each other because they both aim to pass through the candidate cut pixels in the background. However, for false cuts, instead of going the same route as the forward path, the backward path may switch to either side of the character because the cost would be lower since there are no background-to-character

(a) Forward path finding (b) Backward path verification

Figure 4. Two-pass path finding algorithm. In (a), different starting points converge to the same end points. In (b), the false cuts going 'F' have been removed while the true cuts are retained.

and character-to-background transitions (Fig. 4b).

The proposed method can be considered as a two-pass path finding algorithm where the forward direction locates potential cuts and the backward direction verifies them.

III. Experimental Results

Since there is no standard dataset for video text, we have used our text detection method [2] to extract a variety of text lines from TRECVID videos [18], including news programmes, commercials and movie clips. The text lines are divided into 4 datasets: English horizontal (200 images), English non-horizontal (100 images), Chinese horizontal (200 images) and Chinese non-horizontal (100 images).

For comparison purpose, we have implemented an existing method [8], denoted as *Kopf's method*. As mentioned in the introduction section, this method also performs minimum cost path finding. It uses a similar graph structure as our method but with a different cost function. It makes a simple modification from document analysis methods by using the absolute difference in grayscale intensities between consecutive pixels on a path. On the other hand, the proposed method defines the cost function based on GVF and also employs path verification.

A. Sample Segmentation Results

Fig. 5 shows sample segmentation results of the existing method and the proposed method. The image on the left hand side contains English characters of very low contrast. The existing method misses 2 cuts (between 'U' and 'I', and between 'I' and 'T') and gives 4 false cuts while the proposed method identifies all the cuts correctly, without any false cuts. In the image on the right hand side, the Chinese characters are also of low contrast. The proposed method detects all the cuts correctly while the existing method misses one cut between the second and the third characters. Similar to the previous image, the existing method produces many more false cuts than the proposed method (6 versus 2). Fig. 6 shows more results of the proposed method.

B. Segmentation Accuracy

We use recall (R), precision (P) and F-measure (F) as the performance measures, and make the following definitions:

- *Actual Cuts* (AC): Ground truth cuts in the images, which are counted manually.
- *True Cuts* (TC): Detected cuts that only pass through the background region.
- *False Cuts* (FC): Detected cuts that go through the characters.

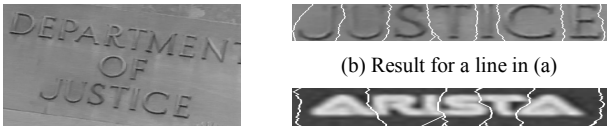
The performance measures are calculated as follows:

- $R = TC / AC$
- $P = TC / (TC + FC)$
- $F = 2 \times P \times R / (P + R)$

(a) Segmentation results by Kopf's method

(b) Segmentation results by our method

Figure 5. Results of the existing method and the proposed method.



(a) Video frame with scene text (b) Result for a line in (a) (c) Result for logo text

Figure 6. Results of the proposed method for non-horizontal text (b) and logo text with touching characters (c). In (c), the gap between 'R' and 'I' is missed because the touching part is quite thick.

Table I shows the performance of Kopf's method and our method on English horizontal and non-horizontal text lines. Although both methods have similar recall, the proposed method has significantly higher precision and F-measure. The existing method produces many false cuts for images with complex background. On the other hand, by using GVF and backward path verification, the proposed method is able to stay as far as possible from the character edges (to allow room for errors) and remove the majority of the false cuts.

Similarly, our method achieves higher precision and F-measure for Chinese horizontal and non-horizontal text lines, although Kopf's method has a slightly higher recall for Chinese horizontal text lines (Table II). The recall of both methods increases, compared to English text. The English datasets are more challenging than the Chinese datasets because they have more variety of text lines, including stylized text used in commercials. Another reason is that Chinese characters have more regular widths than English characters and thus it is easier to detect the gaps. In terms of precision, both methods degrade in performance. A Chinese character typically consists of multiple sub-components and furthermore, there are gaps between these components. Therefore, both methods produce more false cuts.

Both methods also have a lower precision for non-horizontal text, compared to horizontal text. Multi-oriented text is often stylized text or scene text. In both cases, the background is complex; and the contrast is low in the second case. Hence, both methods are more likely to make mistakes. The degradation in the proposed method's performance, however, is less than that of the existing method.

C. Recognition Accuracy

To show that character segmentation helps to improve the recognition rate, we use a recent binarization method [17], which outperforms traditional methods such as Otsu's method and Niblack's method on the dataset of the Document Image Binarization Contest 2009, at two different levels: the text line level and the character level (i.e. the individual characters segmented by the proposed method). The performance measure is the character recognition rate (CRR) using Tesseract [19], Google's OCR engine. (For this

TABLE I. SEGMENTATION RESULTS ON ENGLISH TEXT

Method	English Horizontal			English Non-horizontal		
	R	P	F	R	P	F
Kopf's method	0.89	0.76	0.82	0.88	0.62	0.73
Our method	0.89	0.91	**0.90**	0.91	0.85	**0.88**

TABLE II. SEGMENTATION RESULTS ON CHINESE TEXT

Method	Chinese Horizontal			Chinese Non-horizontal		
	R	P	F	R	P	F
Kopf's method	0.96	0.60	0.74	0.95	0.57	0.71
Our method	0.95	0.81	**0.87**	0.96	0.74	**0.84**

(a) Input (b) Line level 'TO' (c) Character level 'TONIGH §'

Figure 7. Binarization results without segmentation (b) and with segmentation (c), together with the recognition results. The proposed method helps to improve the binarization and recognition results in (c).

experiment, we have considered only English text lines.)

To ensure a fair comparison, for the character level, we put the binarized results together into a line so that in both cases, the OCR engine can utilize its language model to better recognize the characters. Fig. 7 shows the binarization results of a challenging image with a complex and uneven background. Without segmentation, the last four characters are not binarized well because the binarization method fails to choose the appropriate parameters for both characters with clean background and characters with complex background (in the same text line). Only two characters are recognized correctly. On the other hand, with segmentation, the binarization result is significantly improved and six characters are recognized correctly.

Table III shows that it is better to perform binarization at the character level than at the text line level. Part-by-part binarization helps to reduce the problem of complex and uneven background by using local information.

IV. CONCLUSION AND FUTURE WORK

We have proposed a new method for video character segmentation, which is able to produce curved segmentation paths and works directly on grayscale images, i.e. no binarization is required. GVF is used in a novel way to identify candidate cut pixels. A two-pass path finding process is then employed where the forward direction helps to locate potential cuts and the backward direction serves to verify the true cuts and remove the false cuts, i.e. those that go through the middle of the characters. Experimental results show that the proposed method performs well for both English and Chinese text lines of horizontal and non-horizontal orientation. It also helps to increase the character recognition rate by improving binarization results.

In the future, we plan to do a complete work, from text detection to enhancement and recognition. For example, after segmentation, it may be necessary to reconstruct the shapes of the characters, if there are broken edges, before sending them for recognition. The language model can also be implemented outside of the OCR engine, e.g. as in [13].

TABLE III. RECOGNITION RATES WITH AND WITHOUT SEGMENTATION

Method	CRR
Binarization (line level)	59.1%
Segmentation + binarization (character level)	**66.6%**

ACKNOWLEDGMENT

This research is supported in part by A*STAR grant R252-000-402-305.

REFERENCES

[1] K. Jung, K.I. Kim, and A.K. Jain, "Text information extraction in images and video: a survey", Pattern Recognition, vol. 37, no. 5, 2004, pp. 977–997.

[2] P. Shivakumara, T. Q. Phan, and C. L. Tan, "A Laplacian approach to multi-oriented text detection in video", IEEE Transactions on PAMI, vol. 33, no. 2, 2011, pp. 412–419.

[3] M. Mori, M. Sawaki, and N. Hagita, "Video text recognition using feature compensation as category-dependent feature extraction", In Proc. ICDAR, 2003, pp. 645–649.

[4] R. G. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation", IEEE Transactions on PAMI, vol. 18, no. 7, 1996, pp. 690–706.

[5] R. Lienhart and A. Wernicke, "localizing and segmenting text in images and videos", IEEE Transactions on Circuits and Systems for Video Technology, vol. 12, no. 4, 2002, pp. 256–268.

[6] X. Huang, H. Ma, and H. Zhang, "A new video text extraction approach", In Proc. ICME, 2009, pp. 650–653.

[7] G. Miao, G. Zhu, S. Jiang, Q. Huang, C. Xu, and W. Gao, "A real-time score detection and recognition approach for broadcast basketball video", In Proc. ICME, 2007, pp. 1691–1694.

[8] S. Kopf, T. Haenselmann, and W. Effelsberg, "Robust character recognition in low-resolution images and videos", Technical report, University of Mannheim, 2005.

[9] J. Tse, C. Jones, D. Curtis, and E. Yfantis, "An OCR-independent character segmentation using shortest-path in grayscale document images", In Proc. International Conference on Machine Learning and Applications, 2007, pp. 142–147.

[10] Z. Saidane and C. Garcia, "An automatic method for video character segmentation", In Proc. International Conference on Image Analysis and Recognition, 2008, pp. 557–566.

[11] W. Kim and C. Kim, "A new approach for overlay text detection and extraction from complex video scene", IEEE Transactions on Image Processing, vol. 18, no. 2, 2009, pp. 401–411.

[12] Z. Saidane and C. Garcia, "Robust binarization for video text recognition", In Proc. ICDAR, 2007, pp. 874–879.

[13] D. Chen and J. Odobez, "Video text recognition using sequential Monte Carlo and error voting methods", Pattern Recognition Letters, vol. 26, no. 9, 2005, pp. 1386–1403.

[14] C. Xu and J. L. Prince, "Snakes, shapes, and Gradient Vector Flow", IEEE Transactions on Image Processing, vol. 7, no. 3, 1998, pp. 359–369.

[15] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: active contour models", Int. Journal of Computer Vision, vol. 1, no. 4, 1987, pp. 321–331.

[16] J. Wang and J. Jean, "Segmentation of merged characters by neural networks and shortest path", In Proc. ACM/SIGAPP Symposium on Applied Computing, 1993, pp. 762–769.

[17] B. Su, S. Lu, and C. L. Tan, "Binarization of historical document images using the local maximum and minimum", In Proc. Int. Workshop on Document Analysis Systems, 2010, pp. 159–166.

[18] TRECVID. http://trecvid.nist.gov/

[19] Tesseract. http://code.google.com/p/tesseract-ocr/