

Detecting Moving Text in Video Using Temporal Information

Weihoa Huang, Palaiahnakote Shivakumara and Chew Lim Tan
School of Computing, National University of Singapore
{huangwh, tancl}@comp.nus.edu.sg

Abstract

This paper presents our work on automatically detecting moving rigid text in digital videos. The temporal information is obtained by dividing a video frame into sub-blocks and calculating inter-frame motion vector for each sub-block. Text blocks are then extracted through both intra-frame classification and inter-frame spatial relationship checking. Unlike previous works, our method achieves both detection and tracking of moving text at the same time. The method works very well detecting scrolling text in news clips and movies, and is robust towards low resolution and complex background. The computational efficiency of the method is also discussed.

1. Introduction

Extraction and recognition of text from video becomes more and more important when the web based video content providers are very popular and the amount of online multimedia information is explosively increasing. Text extracted from video allows automatically indexing and summarization, which is very helpful for online searching and content management.

Although many works have been proposed on extracting text from video, most of them focus on features within a single video frame, such as connected components, texture or edge information [1]. In fact, a big difference between text extraction from images and video text extraction is that the former deals with a single input image and the latter deals with a set of consecutive frames that gives temporal information and redundant information. The temporal information in video frames mainly refers to the motion of objects or pixel blocks across neighboring frames, while the redundant information refers to reappearing objects and textual content across neighboring frames. The

temporal and redundant information can be used to enhance both still text detection and moving text detection. So far, very few works reported in the literature made use of such information. Li and Doermann used Least-squared-error search to estimate the trajectory of an existing text block in the future frames [2, 3]. Kasturi et al made use of the MPEG motion vectors with refinements to tracking existing text blocks [4]. In both cases, text blocks are detected from a single video frame first, and then trajectory and velocity of each block is estimated based on the temporal information to track these text blocks, as shown in Figure 1(a). To deal with new text blocks entering a frame, an additional step is required to check the features of the regions near the existing text blocks. Thus the temporal information is applied for tracking existing text blocks only.

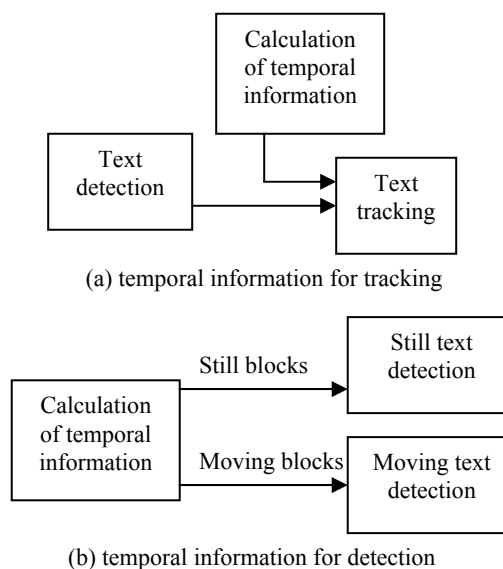


Figure 1. Using temporal information for different purposes.

In our approach, the temporal information is applied to help with text detection, as shown in Figure 1(b). By checking the temporal information beforehand, we are able to divide a frame into regions where text remains still and regions where text may be moving. This narrows down the search range of both still text detection and moving text detection. Furthermore, such division allows redundant information to be applied accordingly. In this paper, the focus is on the detection of moving text.

The major concern here is the computational cost. Detecting of text followed by tracking reduces the number of pixels to be considered during motion vector calculation. On the other hand, calculation of motion vector for the whole video frame is more costly. However the computational cost can be reduced to some extent by pre-filtering some part of a frame that is definitely non-text.

Before we describe the details of our method, we make two assumptions. First of all, text is treated as rigid objects in the video frames, which means it does not scale or deform during its appearance in the video frames. Slight changes in the color of pixels due to anti-aliasing are allowed. Secondly, the components in a text block moves with the same trajectory and velocity. These two assumptions are reasonable as the text in many video of common interests satisfy these constraints. However there are exceptions, especially in TV commercials where many artistic effects are introduced to text characters.

The remaining sections of the paper present further details. Section 2 introduces the calculation of motion vectors, text block classification and the refinement of results. Section 3 presents experiments conducted and the results obtained. Section 4 discusses the performance of the current method and the efficiency issue. Section 5 gives a conclusion to the whole paper.

2. The Proposed Method

2.1. Calculating the motion vectors

A motion vector specifies two features: the direction of movement and the magnitude in terms of pixels per frame. There is motion vector information provided by MPEG encoder [5]. But as Kasturi et al pointed out [4], this motion vector information is not reliable as it is used as motion compensation for video compression, instead of accurately tracking specific objects in the video frames. Thus the motion vectors are too noisy to be directly applied. Secondly, the size of macroblocks in MPEG encoder is pre-determined, either 16×16 or 4×4 pixels, making it hard to be

customized. Finally the motion vectors are not available for I frames. Although I frames are relatively fewer, text can move into or out from I frames.

In our case, we calculate the motion vectors by ourselves. The frame is divided into $N \times N$ sub-blocks ($N = 15$ in our case). For each sub-block, we search for the best correspondence in the next frame, assuming maximum velocity is V pixels/frame. The searching is done using a sliding window, thus the number of block-wise comparisons for a sub-block is $(2V+1)^2$. To accelerate the process, we assume that text is moving either in a horizontal direction or a vertical direction. This assumption can immediately reduce the number of block-wise comparisons to $(4V+1)$. Of course the assumption will decrease the generality of the method, which then only detects text blocks with horizontal or vertical movement.

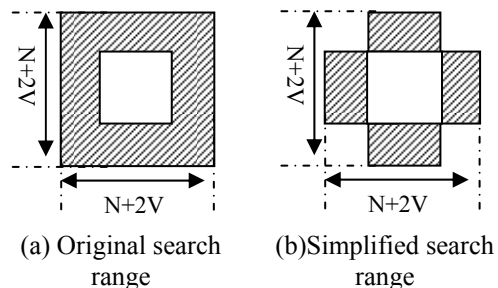


Figure 2. Search range of a sub-block

Instead of *Least Mean Square Error (LMSE)*, we use *Sum of Absolute Differences (SAD)* to reduce number of square operation. The *SAD* compares the intensity of the pixels in a sub-block B_1 and the target sub-block B_2 using formula (1).

$$SAD(B_1, B_2) = \sum_{x=1}^W \sum_{y=1}^H (I_1(x, y) - I_2(x, y)) \quad (1)$$

The sliding window starts from center outwards. If the *SAD* of a sub-block at the same location in the next frame is zero or the smallest within the search window, then the magnitude of the motion vector of the sub-block is set to zero. If the *SAD* value is larger than a threshold value, then there is no matching sub-block in the next frame, and the magnitude of the motion vector is set to negative. For other sub-blocks, the direction of the motion vector is set as the displacement between the matching block and the current block, and the magnitude of the motion vector is set as the distance between the centers of the two blocks.

The process can be speed up by ignoring sub-blocks that have nearly uniform color and are unlikely

to contain text that has significant contrast from the background. The calculation of color variation is done to each sub-block before the motion vector calculation. If the color variation is very small, the block is skipped.

Because text is assumed to be moving continuously across a number of frames, the calculation of motion vector is done between frame F_i and F_{i+1} , also between F_i and F_{i+2} , resulting in two motion vectors. If the magnitude in any of the two motion vectors is zero, then the magnitude of the first motion vector is zero. In this way, the motion vector between F_i and F_{i+2} provides supplementary information on the movement of the sub-blocks in future frames. It is noted that we can even further calculate motion vectors between F_i and F_{i+3} , F_{i+4} etc. to get even more information about the future movement. However the cost of such calculation needs to be well controlled.

2.2. Classification of Sub-blocks

As we are interested in finding text blocks that are moving, sub-blocks with motion vector of zero or negative magnitude are ignored. For the remaining sub-blocks, we apply Sobel edge detector to each block. Then two edge-based features are examined to estimate if the block contains text components. They are the density of edge pixels and the maximum number of edge pixels per scan line. These two features are also used in [4], based on the fact that a text line contains significant number of strokes that are contrasted from the background. If the values of these two features in a block are high enough, the sub-block is labeled as a candidate text block.

2.3. Forming text blocks and text groups

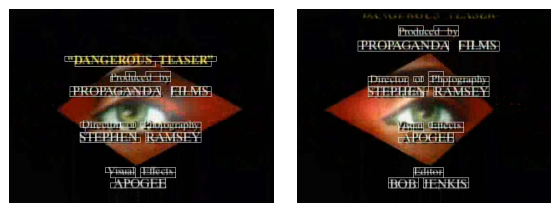
A complete text block is formed by merging bounding boxes of neighboring candidate text blocks that are moving together with the same direction and magnitude. As the motion vector may not be calculated perfectly accurately, there may be some minor error in the magnitude. Thus we give a tolerance of ± 1 pixels.

A text group contains text blocks that moves with the same direction and same magnitude. The text blocks may not be near each other. The concept of text group is to handle different text blocks moving in different directions in the same video sequence. A group contains the block index and the common motion vector for the whole group.

3. Experimental Results

We collected 8983 frames from 20 video clips, including 2 music videos, 8 movies and 10 news clips. The languages of the movies include English, Chinese, Korean and Thai. The frame rate is 25 frames per second for all the videos. The movie frames are selected towards the end of each movie so that the movie credits are included. The scrolling text in the music videos and movies are the credits going upwards on the screen, as shown in Figure 3(a) and (b). On the other hand, the news clips have text scrolling from right to left at the bottom, as shown in Figure 3(c).

The system automatically examines all the video frames in a batch process. The detection of text blocks is performed by checking 3 consecutive frames at once, and the resulting text blocks are indicated using their bounding boxes, as can be seen in Figure 3.



(a) An English music video



(b) A Chinese Movie



(c) A news clip

Figure 3. Sample results of text detection

To evaluate the performance of the system, we manually count the number of false positives that are non-text blocks wrongly treated as text-blocks by the system. We also count the number of misdetections that are text blocks not detected by our system. A text block is considered correctly detected if the bounding box covers all the characters in the same text block. If a text block has some characters not covered by the detected bounding box, it is considered as a misdetection as well. The false positive rate and

detection rate are summarized in Table 1. For the calculation, the 2 music videos are categorized as movies as well.

Table 1. Performance of the proposed system

Testing data	Detection rate	False positive rate	Avg. Processing time (ms)
Movies	98.41%	2.54%	93.05
News	96.85%	17.47%	51.73
Overall	98.32%	3.44%	83.81

4. Discussions

False positives mostly occur where there are rigid textured objects or some textured part of an object moving, as the example in Figure 4 shows. This is because detection is based on temporal information among 3 neighboring frames only. If we enforce constraints on a larger number of consecutive frames, then the false positives can be further eliminated.



Figure 4. Examples of false positive

Some misdetections are due to the setting of parameters. For example, text with too low resolution or contrast is not detected. Furthermore, text blocks beyond the search range of the motion vectors are not detected. However, as text detection is done to all the frames, redundant information is there to recover such missing text strings. In fact, through our observation, most of the missing text blocks are detected in nearby frames. Thus if we consider the redundant information, the detection rate is close to 100%. Another cause of misdetection is the nature of language. In languages like Chinese, characters in a word can be separated far from each other. Some characters are in very simple shape and contain only a few strokes, causing the edge based feature not working well for such characters. This is a limitation of the existing method that it does not work well for some languages in large font size.

Based on our observation, a text block stays on screen for at least 2 seconds for the readers to capture the semantic meaning. This applies to both still text and moving text. Thus the maximum moving velocity

V (pixels/frame) of a text block can be estimated by

$$V = W / (FR \times 2) \quad (2)$$

Where W is the width of frame, FR is the frame rate. This determines the search range for calculating the motion vectors.

The system was written as a Windows application using Visual C++. The experiment was run on a PC with Pentium IV 2.6GHz processor. The average processing time per frame is also shown in Table 1. It can be seen that detection based on 3 consecutive frames does not result in very significant amount of processing time. Through investigation, we found that the speed up step actually skipped 83.76% of the blocks before motion vector calculation. Furthermore, another 10.95% of the blocks give velocity of zero, thus are ignored during inter-frame classification.

5. Conclusion

This paper presents our work on automatically detecting moving rigid text in digital videos. Unlike previous works, our method makes use of the temporal information for text detection instead of tracking, which significantly improves both detection rate and detection accuracy. Experimental results show that the method works very well for detecting scrolling text in news clips and movie credits. The next step is to further apply the temporal information in actual recognition of the text detected.

Acknowledgment

This research is supported in part by IDM R&D grant R252-000-325-279.

References

- [1] K. Jung, K. I. Kin and A. K. Jain. Text information extraction in images and video: a survey. *Pattern Recognition*, 37(5): 977-997, 2004.
- [2] H. Li and D. Doermann. Automatic text tracking in digital videos. *IEEE Workshop on Multimedia Signal Processing*, pp. 21-26, 1998.
- [3] H. Li, D. Doermann and O. Kia. Automatic Text detection and tracking in digital video. *IEEE Trans. Image Processing*, 9(1): 147-156, 2000.
- [4] D. Crandall, S. Antani and R. Kasturi. Extraction of special effects caption text events from digital video. *IJDAR'03*, 5: 138-157, 2003.
- [5] J. L. Mitchell, W. B. Pennebaker, C. E. Fogg and D. J. LeGall. *MPEG Video Compression standard*. Digital Multimedia Standards Series. Chapman and Hall, London, 1997.