SEQUENTIAL MONTE CARLO VIDEO TEXT SEGMENTATION

Datong Chen and Jean-Marc Odobez

Dalle Molle Institute for Perceptual Artificial Intelligence (IDIAP)
Rue du Simplon 4, CH-1920 Martigny, Switzerland
{chen, odobez}@idiap.ch

ABSTRACT

This paper presents a probabilistic algorithm for segmenting and recognizing text embedded in video sequences. The algorithm approximates the posterior distribution of segmentation thresholds of video text by a set of weighted samples. After initialization the set of samples is recursively refined by random sampling under a temporal Bayesian framework. The proposed methodology allows us to estimate the optimal text segmentation parameters directly in function of the string recognition results instead of segmentation quality. Results on a database of 6944 images demonstrate the validity of the algorithm. [1]

1. INTRODUCTION

Text recognition in video is one of the key components in the development of advanced video annotation and retrieval systems. Text characters contained in video can be any grayscale value and embedded in consecutive frames with complex changing backgrounds. To recognize these text characters, a segmentation step is necessary before applying an optical character recognition (OCR) software, even when the whole text string is well located. Therefore, a large amount of work on text segmentation from complex background has been published in recent years.

Most text segmentation methods are performed after text string is located in images. These methods assume that the grayscale distribution is bimodal and devote efforts to perform better binarization such as combining global and local thresholding [1], M-estimation [4] and simple smoothing [11]. However, the bimodal hypothesis is not always fulfilled. In [3], multiple hypotheses segmentation method, which assumes that the grayscale distribution can be k-modal (k=2,3,4), has been shown to improve the recognition performance. To exploit the temporal information provided by consecutive frames of a given text string, Sato [9] and Lienhart [7] computed the maximum or minimum value at each pixel position over consecutive frames. However, this method can only be applied on black or white characters. Li [6] proposed a multi-frame enhancement which computes the average of pre-located text regions in multiple frames for further segmentation and recognition. The average image has smaller variance of noise but may propagate blur characters in frames. A common drawback of these temporal methods is that they require accurate text image alignment at the pixel level.

In order to use the temporal information at a higher level than the pixel level, we can exploit different recognized text strings obtaining by segmenting consecutive text images of the same text string. This implies that segmentation needs to be performed on each individual text images from different frames. However, applying traditional segmentation on every frame has two problems. Firstly, it is not efficient in terms of computation cost. For a given video text string, the segmentation characteristics in different frames are varying but not completely unpredictable. Thus, the optimal parameters of the previous frame could be reused instead of performing an individual segmentation again. Secondly, traditional segmentation algorithms usually rely on predefined criterions which may not always yield segmentation results that would lead to good recognition [10]. In other words, the segmentation quality in our case should be validated using recognition results instead of any predefined criterion on grayscale values of the image. In fact, applying an OCR software on visually similar segmentation results of a same text image can lead to different recognized strings.

To address these two problems, in this paper, we present a Monte Carlo video text segmentation (MCVTS) method to segment text characters of any grayscale values by exploiting temporal information. Generally, a segmentation of text image can be regarded as a process that searches for a couple of thresholds (lower and upper) covering the grayscale values of text pixels. The MCVTS method performs a traditional segmentation of the text image in the first frame and propagate the resulting threshold couples to other frames using a particle filter approach. By introducing randomness in the exploration of the space of possible segmentation parameters in a Bayesian framework, the particle representation allows to adapt to changes of grayscale values

[1] This work has been performed in the framework of the "Combined Image and Word Spotting" project granted by the European IST Programme.

both in the text and background by maintaining multiple-hypotheses. In the presence of ambiguities and instabilities, the new method compensates OCR errors encountered when applying current OCR systems on video text. These errors are mainly due to the low resolution of characters (before resizing and interpolation), the short length of the string and their unknown font.

A detail description of the MCVTS algorithm is proposed in Section 2. In the third section, we present experiments and discussion.

2. ALGORITHM

2.1. Bayes filtering

Bayes filters address the problem of estimating the posterior probability $p(x_t|o_{1:t})$ of a dynamic state given a sequence of observations, where x_t denotes the state at time t and $o_{1:t}$ denote the observation sequence from time 1 to time t. In the MCVTS algorithm, the state $x_t = (u,l)_t$ is characterized by a upper (u) and a lower (l) segmentation threshold and the observations are the grayscale text images extracted and tracked in consecutive video frames. Thus, the goal of the video text segmentation filtering is to estimate the states that lead to an accurate segmentation of the text images, or, better, to correctly recognized strings.

To derive a recursive update equation for the posterior, we exploit Bayes rule :

$$p(x_{t+1}|o_{1:t+1}) = \alpha p(o_{t+1}|x_{t+1}, o_{1:t})\, p(x_{t+1}|o_{1:t})$$

where $\alpha = p(o_{t+1}|o_{1:t})^{-1}$ is a normalization constant. The prediction term $p(x_{t+1}|o_{1:t})$ can be expanded by marginalizing over the state at time t:

$$p(x_{t+1}|o_{1:t}) = \int_{x_t} p(x_{t+1}|x_t, o_{1:t})\, p(x_t|o_{1:t})\, dx_t.$$

Assuming independence of observations conditioned on the states (i.e. $p(o_{t+1}|x_{t+1}, o_{1:t}) = p(o_{t+1}|x_{t+1})$) and a first order Markov model for the sequence of states (i.e. $p(x_{t+1}|x_t, o_{1:t}) = p(x_{t+1}|x_t)$). we obtain the following recursive equation for the posterior :

$$p(x_{t+1}|o_{1:t+1}) = \alpha p(o_{t+1}|x_{t+1}) \int_{x_t} p(x_{t+1}|x_t)\, p(x_t|o_{1:t})\, dx_t$$

The exploitation of the last equation requires the definition of two conditional densities: the transition probability $p(x_{t+1}|x_t)$ and the data likelihood $p(o_t|x_t)$. Both models are typically time-invariant so that we can simplify the notation by denoting these models $p\left(x'|x\right)$ and $p(o|x)$ respectively. They are presented in the next Subsection, while the description of the particle filter implementation of the above equation is deferred to the end of the Section.

2.2. Probabilistic models for segmentation

Transition probability

In the context of video text segmentation, the transition probability $p\left(x'|x\right)$ is a probabilistic prior on text threshold variations. We assume, in this paper, that the variations are mainly due to noise, and the transition model is therefore modeled as Gaussian process.

Data likelihood

The data likelihood $p(o|x)$ provides an evaluation of the segmentation quality of the observed image o given a pair of thresholds $x = (l,u)$. This evaluation could rely on the segmented image. However, computing accurate measures of segmentation quality in term of character extraction is difficult without performing some character recognition analysis. Besides, visually well segmented image does not always lead to a correct recognition. The OCR may produce errors due to the short length and the unknown font of the text string. Therefore, since ultimately we are interested in the recognized text string, the data likelihood will be evaluated from the output T of the OCR.

To extract the text string T, we first binarize the image o using x, and remove non-characters using a connected component analysis step [2]. Then, a text string T is produced by applying an OCR software on the resulting binary image.

To evaluate the data likelihood using string T, we exploit prior information on text strings and on the OCR performance based on language modeling and OCR recognition statistics respectively.[2] Let us denote the string T as $T = (T_i)_{i=1,\ldots,l_T}$ where l_T denotes the length of the string and each character T_i is an element of the character set C_s :

$$C_s = (0,\ldots,9,a,\ldots,z,A,\ldots,Z,G_b)$$

in which G_b corresponds to any other garbage character. Finally, let us denote by H_a (resp. H_n) the hypothesis that the string T or the characters T_i are generated from an accurate (resp. a noisy) segmentation. The data likelyhood is defined as the probability of the accurate segmentation hypothesis H_a given the string T:

$$\begin{aligned} p(o|x) &\propto p(H_a|T) = \tfrac{p(T|H_a)p(H_a)}{p(T)} \\ &= \tfrac{p(T|H_a)p(H_a)}{p(T|H_a)p(H_a)+p(T|H_n)p(H_n)} \\ &= \tfrac{1}{1+\frac{p(T|H_n)p(H_n)}{p(T|H_a)p(H_a)}}. \end{aligned}$$

We estimated the noise free language model $p(.|H_a)$ by applying the CMU-Cambridge Statistical Language Modeling (SLM) toolkit on Gutenberg collections[3]. A bigram model

[2] From a qualitative point of view, when given text-like background or inaccurate segmentation, the OCR system produces mainly garbage characters like ., ,!, & etc and simple characters like i,l, and r.

[3] www.gutenberg.net

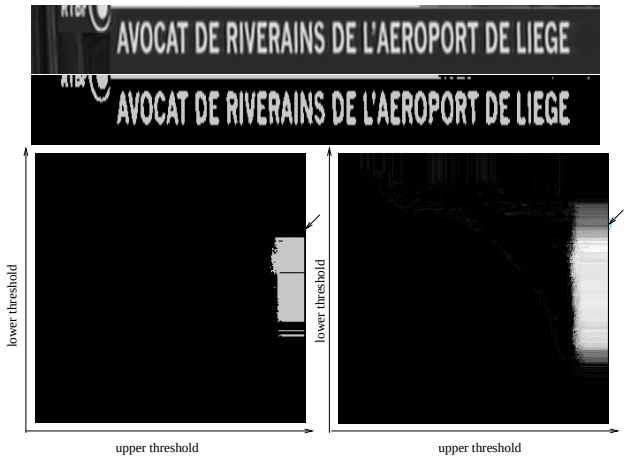

Fig. 1. Data likelihood approximation: the observed text image is displayed at the top. The second image displays the results of applying Otsu binarization, which corresponds to OCR output "V AVOCAT DE RIVERAINS DE L AEROPORT DE iIEGE". In the last row, the left image shows the states that lead to the recognition of all the words in the ground truth, and the right image displays the proposed data likelihood at all the states.

was selected. Cutoff and backoff techniques [5] were employed to address the problems associated with sparse training data for special characters (e.g. numbers and garbage characters). The noise language model $\mathrm{p}(.|H_n)$ model was obtained by applying the same toolkit on a database of strings collected from the OCR system output when providing the OCR input with either badly segmented texts or text-like false alarms coming from the text detection process. Only a unigram model was used because the size of the background dataset was insufficient to obtain a good bigram model. The prior ratio on the two hypotheses is modeled as $\frac{p(H_n)}{p(H_a)} = b$, where b [4] is a bias that can be estimated from general video data. The data likelihood is then given by:

$$p(o|x) \propto \frac{1}{1 + \frac{\Pi_{i=1}^{l_T} \mathrm{p}(T_i|H_n)}{\mathrm{p}(T_1|H_a)\Pi_{i=2}^{l_T} \mathrm{p}(T_i|T_{i-1},H_a)} * b}.$$

Figure 1 shows the ground truth data likelihood, which is defined as $p(o|x) = 0$ if not all the words in the ground truth are recognized, otherwise $p(o|x) = 1$. The figure also shows the proposed data likelihood of the image at all the possible states, illustrating that our probabilistic model is accurate. Even if the initial state (here provided by an Otsu algorithm [8] and shown with an arrow in the images) leads to an incorrectly recognized text string, the Bayesian filtering methodology, thanks to the introduction of random perturbation and our data likelihood model, will still be able to find a state that provides the correct string. The Bayesian filtering is implemented by a recursive particle filter that is described below.

[4] We have chosen $b = b_0^{l_T}$ to take into account string length in the prior.

Fig. 2. Examples of located embedded text in video.

2.3. Particle approximation

The idea of the particle filter is to approximate the posterior $p(x_t|o_{1...t})$ by a set of N weighted samples $X_t = (x_t^j, w_t^j)_{j=1...N}$ such that:

$$p(x_t|o_{1...t}) \approx \sum_{j=1}^{N} w_t \delta\left(x_t^j - x_t\right),$$

where δ is the mass choice function ($\delta(0) = 1$, otherwise $\delta(x) = 0$). The initial set of samples represents the initial knowledge $p(x_1|o_1)$ and can be initialized using an Otsu algorithm applied on the first image. The recursive update is realized in three steps. First, sample $\tilde{x}_t^i$ from the approximated posterior X_t. Then, sample x_{t+1}^i from the transition probability $p\left(x_{t+1}|\tilde{x}_t^i\right)$. Finally, assign $w_{t+1}^i = p\left(o_{t+1}|x_{t+1}^i\right)$ as the weight of the new sample (x_{t+1}^i, w_{t+1}^i). In our case, since the number of samples per image is low, we add the new particles to the set X_{t+1} of samples instead of replacing the old values with the new ones.

1. initial X_1 using an Otsu algorithm;
2. for each frame $t = 1,\ldots,n$ do step 3 and 4;
3. for $i = 1$ to m do sample $\tilde{x}_t^i \sim X_t$; sample $x_{t+1}^i \sim p\left(x_{t+1}|\tilde{x}_t^i\right)$; set $w_{t+1}^i = p(o_{t+1}|x_{t+1}^i)$;
4. $X_{t+1} = X_t$, add the m new samples (x_{t+1}^i, w_{t+1}^i) to X_{t+1}.
5. output the text string that corresponds to the segmentation with the highest data likelihood.

3. EXPERIMENTS AND DISCUSSION

The MCVTS algorithm was tested on text regions located and extracted from one hour of video provided by the CIM-

methods	Ext.	CRR	Prec.	WRR
Baseline	3664	88.9%	80.1%	70.8%
MCVTS	3637	93.9%	85.3%	83.8%

Table 1. Performance comparison between the MCVTS (m=3) and the baseline method is average image method [6]: character recognition rate (CRR), precision (Prec.), word recognition rate (WRR).

WOS project, using the algorithm presented in [2]. The whole database consists of 250 different text strings (3301 characters) in 6944 text images (about 28 images per text string in average). Figure 2 shows some image examples.

Performances are evaluated using character recognition rates (CRR) and precision rates (Prec) that are computed on a ground truth basis as :

$$CRR = \frac{N_r}{N} \quad and \quad Prec = \frac{N_r}{N_e}$$

N is the true total number of characters in the ground truth, N_r is the number of correctly recognized characters and N_e is the total number of extracted characters. Additionally, we also compute word recognition rate to get an idea of the coherency of character recognition within one solution.

Table 1 lists the results of the average image method [6] and the MCVTS algorithm with $m = 3$. It illustrates that the MCVTS algorithm significantly improves the character recognition, the precision and the word recognition rate. Especially, the MCVTS algorithm performs better when the text image is noisy or the grayscale of characters span a wide range as shown in Figure 2. From the computation complexity point of view, the CPU cost of the MCVTS algorithm depends on the number of samples, the thresholding operation and OCR CPU cost. The algorithm can handle 3-5 text strings in real time with $m = 3$. Note that using more than $m = 3$ particles per image does not improve the performance of the algorithm. The average number of samples per text string is thus around 85.

In this paper, we proposed a Monte Carlo method for segmenting and recognizing embedded text of any grayscale value in image and video based on particle filter. The MCVTS algorithm has four main advantages for segmenting video text. Firstly, the algorithm proposes a methodological way to search for segmentation parameters that lead to accurate results. Secondly, the algorithm adapts itself to the data by sampling in proportion to the posterior likelihood. This enable us to propose an accurate probability model based on OCR results instead of estimating the posterior of segmentation based on segmented images. Thirdly, the algorithm does not require precise tracking of text images among video frames at the pixel level. Finally, the MCVTS algorithm is very easy to implement and also easy to be extended to other state spaces, such as parameters of local thresholding techniques (e.g. Niblack binarization).

4. REFERENCES

[1] H. Kamada amd K. Fujimoto. High-speed, high-accuracy binrization method for recognizing text in images of low spatial resolutions. In *Int. Conf. on Document Analysis and Recognition*, pages 139–142, Sept. 1999.

[2] D. Chen, H. Bourlard, and J-Ph. Thiran. Text identification in complex background using SVM. In *Int. Conf. on Computer Vision and Pattern Recognition*, pages 621–626, Dec. 2001.

[3] D. Chen, J.M. Odobez, and H. Bourlard. Text segmentation and recognition in complex background based on markov random field. In *Int. Conf. Pattern Recognition*, volume 2, 2002.

[4] O. Hori. A video text extraction method for character recogntion. In *Int. Conf. on Document Analysis and Recognition*, pages 25–28, Sept. 1999.

[5] S.M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Trans. on Acoustics, Speech and Signal Processing*, 35:400–401, 1987.

[6] H. Li and D. Doermann. Text enhancement in digital video using multiple frame integration. In *ACM Multimedia*, pages 385–395, 1999.

[7] R. Lienhart and A. Wernicke. Localizing and segmenting text in images and videos. *IEEE Trans. on Circuits and Systems for Video Technology*, 12(4):256–268, 2002.

[8] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Trans. on Systems, Man and Cybernetics*, 1(9):62–66, 1979.

[9] T. Sato, T. Kanade, E. K. Hughes, M. A. Smith, and S. Satoh. Video OCR: indexing digital news libraries by recognition of superimposed caption. In *ACM Multimedia System Special Issue on Video Libraries*, pages 52–60, Feb. 1998.

[10] Christian Wolf, Jean-Michel Jolion, and Francoise Chassaing. Text localization, enhancement and binarization in multimedia documents. In *Int. Conf. on Pattern recognition*, pages 1037–1040, Quebec City, Canada., August 2002.

[11] V. Wu, R. Manmatha, and E. M. Riseman. Finding text in images. In *Proc. ACM Int. Conf. Digital Libraries*, pages 23–26, 1997.