End-to-end subtitle detection and recognition for videos in East Asian languages via CNN ensemble

Yan Xu\textsuperscript{a,b}, Siyuan Shan\textsuperscript{a}, Ziming Qiu\textsuperscript{c}, Zhipeng Jia\textsuperscript{b,d}, Zhengyang Shen\textsuperscript{e}, Yipei Wang\textsuperscript{a}, Mengfei Shi\textsuperscript{f}, Eric I-Chao Chang\textsuperscript{b,*}

\textsuperscript{a} State Key Laboratory of Software Development Environment and Key Laboratory of Biomechanics and Mechanobiology of Ministry of Education and Research Institute of Beihang University in Shenzhen, Beijing Advanced Innovation Centre for Biomedical Engineering, Beihang University, Beijing 100191, China
\textsuperscript{b} Microsoft Research Asia, Beijing 100080, China
\textsuperscript{c} Electrical and Computer Engineering, Tandon School of Engineering, New York University, Brooklyn, USA
\textsuperscript{d} Institute for Interdisciplinary Information Sciences, Tsinghua University, Beijing 100084, China
\textsuperscript{e} Department of Computer Science, University of North Carolina at Chapel Hill, USA
\textsuperscript{f} Beijing No.8 High School, Beijing 100032, China

\begin{abstract}
In this paper, we propose an innovative end-to-end subtitle detection and recognition system for videos in East Asian languages. Our end-to-end system consists of multiple stages. Subtitles are firstly detected by a novel image operator based on the sequence information of consecutive video frames. Then, an ensemble of Convolutional Neural Networks (CNNs) trained on synthetic data is adopted for detecting and recognizing East Asian characters. Finally, a dynamic programming approach leveraging language models is applied to constitute results of the entire body of text lines. The proposed system achieves average end-to-end accuracies of 98.2\% and 98.3\% on 40 videos in Simplified Chinese and 40 videos in Traditional Chinese respectively, which is a significant outperformance of other existing methods. The near-perfect accuracy of our system dramatically narrows the gap between human cognitive ability and state-of-the-art algorithms used for such a task.
\end{abstract}

Keywords:
Subtitle text detection
Subtitle text recognition
Synthetic training data
Convolutional neural networks
Video sequence information
East Asian language

\section{1. Introduction}

Detecting and recognizing video subtitle texts in East Asian languages (e.g. Simplified Chinese, Traditional Chinese, Japanese and Korean) is a challenging task with many promising applications like automatic video retrieval and summarization. Different from traditional printed document OCR, recognizing subtitle texts embedded in videos is complicated by cluttered backgrounds, diversified fonts, loss of resolution and low contrast between texts and backgrounds [1].

Given that video subtitles are almost always horizontal, subtitle detection can be partitioned into two steps: subtitle top/bottom boundary (STBB) detection and subtitle left/right boundary (SLRB) detection. These four detected boundaries enclose a bounding box that is likely to contain subtitle texts. Then the texts inside the bounding box are ready to be recognized.

Despite the similarity between video subtitle detection and scene text detection, the instinctive sequence information of videos makes it necessary to address these two tasks respectively [2]. As illustrated in Fig. 1, for most videos with single-line subtitles in East Asian languages, texts at the subtitle region exhibit homogeneous properties throughout the video, including consistent STBB position, color and single character width (SCW). Meanwhile, the non-subtitle region varies unpredictably from frame to frame. With the assistance of this valuable sequence information, we put forward a suitable image operator that can facilitate the detection of STBB and SCW. We call this image operator the \textit{Character Width Transform} (CWT), as it exploits one of the most distinctive features of East Asian characters—consistent SCW.

Considering the complexity of backgrounds and the diversity of subtitle texts, adopting a high-capacity classifier for both text detection and recognition is imperative. CNNs have most recently proven their mettle handling image text detection and recognition [3,4]. By virtue of their special bio-inspired structures (i.e. local receptive fields, weight sharing

https://doi.org/10.1016/j.image.2017.09.013
Received 8 April 2017; Received in revised form 30 September 2017; Accepted 30 September 2017
Available online 16 October 2017
© 2017 Elsevier B.V. All rights reserved.
Our contribution can be summarized as follows:

- We propose an end-to-end subtitle detection and recognition system for East Asian languages. By achieving 98.2% and 98.3% end-to-end recognition accuracies for Simplified Chinese and Traditional Chinese respectively, this system remarkably narrows the gap to human-level reading performance.2
- We define a novel image operator whose outputs enable the effective detection of STBB and SCW. The sequence information is integrated throughout the video to increase the reliability of the proposed image operator. This module achieves a competitive result on a dataset including 1097 videos.
- We leverage a CNN ensemble to perform the classification of East Asian characters across huge dictionaries. The ensemble reduces the recognition error rate by approximately 75% in comparison with a single CNN. CNNs in our system serve both as text detectors and character recognizers.

The remainder of this paper is organized as follows. Section 2 reviews related works. Section 3 describes the synthetic data generation scheme, the CNN ensemble and the end-to-end system. In Section 4, the proposed system and each module in it are evaluated on a large dataset, and the experimental results are presented. In Section 5, observations from our experiments are discussed. A conclusion and discussion of future work are given in Section 6.

2. Related work

In this section, we focus on reviewing relevant literature on image text detection and recognition. As for other text detection and recognition methods, several review papers [1,7–10] can be referred to.

2.1. Image text detection

Generally, text detection methods are based on either connected components or sliding windows [4]. Connected component based methods, like Maximally Stable Extremal Regions (MSER) [11–13], enjoy their computational efficiency and high recall rates, but suffer from a large number of false detections. Methods based on sliding windows [3,4,14–17] adopt a multi-scale window to scan through all locations of an image, then apply a trained classifier with either hand-engineered features or learned features to distinguish texts from non-texts. Though this kind of method produces significantly less false

---

1. https://drive.google.com/file/d/0B0x5IW_m4AC5M0RuY1JiUWJcUU/view?usp=sharing.

2. Human-level reading performance is 99.6% according to the experiment in Section 4.1.
detections, the computational cost of scanning every location of the image is unbearable. Therefore, connected component-based methods and sliding-window-based methods are often utilized together for text detection [6,13,18,19], where the former generate text region proposals and the latter eliminate false detections. This text detection scheme is also adopted in this paper, but our text region proposal method is based on the sequence information of video and thus not comparable to existing methods designed for scene text detection [20–22]. Hence, we focus on reviewing methods based on video sequence information and text region verification works that aim to eliminate false detections.

2.1.1. Methods incorporating video sequence information

Tang et al. [23] analyze the difference of adjacent frames to detect the subtitle text based on the assumption that in each shot the scene changes more gradually than the subtitle text. Wang et al. [24] exploit a multi-frame integration technique within 30 consecutive frames to reduce the complexity of backgrounds before the text detection process. Liu et al. [25] compare the distribution of stroke-like edges between adjacent frames and segment the video into clips in which the same caption is contained. Then they adopt a temporal “and” operation to identify caption regions. However, contrary to the proposed method in this paper, these existing methods rarely exploit temporal information throughout the video.

2.1.2. Text region verification based on hand-engineered features

Traditional methods harness manually designed low-level features such as SIFT and histogram of oriented gradients (HOG) to train a classifier to distinguish texts from non-texts. For instance, Wang et al. [26] propose a new block partition method and combine the edge orientation histogram feature with the gray scale contrast feature (EOH-GSC) for text verification. Neumann et al. [19] adopt the SVM classifier with a set of geometric features for text detection. Wang et al. [15] and Jaderberg et al. [6] eliminate false text detections by Random Ferns with HOG features. Minetto et al. [27] propose a HOG-based texture descriptor (T-HOG) that ameliorates traditional HOG features on the text/non-text discrimination task. Liang et al. [28] propose a multi-spectral fusion method for enhancing low resolution text pixels and use MSER for text detection. Yin et al. [29] adopt a pruning algorithm to extract MSERs and detect text in natural scene images. Effective as these handcrafted features are to describe image content information, they are suboptimal to represent text data due to their heavy dependence on priori knowledge and heuristic rules.

2.1.3. Text region verification based on feature learning

In contrast to these traditional methods, more advanced methods take advantage of high-capability feature learning to automatically learn a more robust representation of text data, hence possessing a powerful discrimination ability to eliminate false text detections. Yao et al. [30] use Fully Convolutional Network to localize texts in a holistic manner. Delakis and Garcia [16] train a CNN to detect texts from raw images in a sliding window fashion. Wang et al. [3] and Huang et al. [13] utilize a multi-layer CNN for both text detection and recognition, and the first layer of the network is trained with an unsupervised learning algorithm [14]. Ren et al. [17] are the first to tackle Simplified Chinese scene text detection. They propose an algorithm called convolutional sparse auto-encoder (CSAE) to pre-train the first layer of CNN on unlabeled synthetic data for Simplified Chinese scene text detection.

Both the above-mentioned methods and our approach are based on feature learning, comparing favorably against methods based on hand-engineered features. We further promote East Asian text detection performance by training a CNN ensemble in an end-to-end manner on labeled synthetic data.

2.2. Image text recognition

Similar to Section 2.1 where the importance of features is addressed, existing image text recognition methods are also classified into those based on hand-engineered features [15,19,31–54] and those based on feature learning [3,4,6,14,18,35–42].

2.2.1. Image text recognition based on hand-engineered features

Bissacco et al. [33] propose a scene text recognition system by combining a neural network trained on HOG features with a powerful language model. Lee et al. [31] present a new text recognition method by merging gradient histograms, gradient magnitude and color features. Khare et al. [43] propose a novel blind deconvolution method for deblurring the blur image and improving text recognition performance. Bai et al. [34] use HOG features, artificially generated training data and a neural network classifier for Simplified Chinese image text recognition. Though state-of-the-art performance was achieved, its 85.44% recognition accuracy still impedes its practical application.

2.2.2. Image text recognition based on feature learning


Works tackling East Asian image text recognition with CNNs are relatively rare. Zhong et al. [41] adopt a CNN with a multi-pooling layer on top of the final convolutional layer to perform multi-font printed Simplified Chinese character recognition, which renders their method robust to spatial layout variations and deformations. Bai et al. [39] propose a CNN architecture for Simplified Chinese and English character recognition, and the hidden-layers are shared across these two languages. However, both works [39,41] can only recognize an isolated character as opposed to a text line. Besides, the work of Bai et al. [39] can only recognize 500 Simplified Chinese characters, though there are thousands of characters commonly used [45]. Therefore, to the best of our knowledge, the system proposed in this paper is the first to leverage high-capability CNNs to recognize image text lines in Simplified Chinese (and also other East Asian languages) with a comprehensive alphabet consisting of 7008 characters.

3. Method

In this section, we will describe the synthetic data generation pipeline, the CNN ensemble and the end-to-end system in detail. As illustrated in Fig. 2, the end-to-end system consists of three modules including STBB and SCW detection, SLRB detection and subtitle recognition.

3.1. Synthetic data generation

As it is easy to simulate the generation pipeline of subtitles, training data are synthetically generated in a scheme similar to [46,47]. The labeled synthetic data in Simplified Chinese (SC), Traditional Chinese (TC) and Japanese (JP) are generated to train CNNs in SC, TC and JP respectively.

(1) Dictionary construction: three comprehensive dictionaries that respectively cover 7009 SC characters, 4809 TC characters and 2282 JP characters are constructed. A space character is included in each dictionary.

(2) Font rendering: 22, 19 and 17 kinds of font for SC, TC and JP are collected respectively for introducing more variations to the training data.

(3) Random selection of background and character: 45,441 frames are randomly extracted from 11 news videos downloaded from the Internet. Afterwards, small background patches are randomly cropped from these frames. The size of every background patch is determined with regard to a random combination of a character and a font. 200,000
machine-born white characters with dark shadows are generated by repeatedly selecting a random combination of a font and a character from the dictionary.

(4) Random shift and Gaussian blur: every randomly generated machine-born character is superimposed on a randomly selected background patch with a random shift of \( \theta \) pixels, where \( \theta \) is drawn from a uniform distribution on the interval \([-2, 2]\). Then every image is convolved with a Gaussian blur at the scale of \( \sigma \) pixels, where \( \sigma \) is drawn from a uniform distribution on the interval \([0.5, 1.6]\). The convolved images are then converted to grayscale images and resized to 24 \times 24. Therefore, 200,000 samples are generated for SC, TC, and JP respectively.

The procedure of generating training samples for the text/non-text SVM classifier is almost the same, except that the same number of background patches without characters are also stored as non-text training examples. Fig. 3 presents some of the training data.

3.2. Convolutional neural networks ensemble

CNNs have been recently applied to recognize image texts with great success [3, 4, 6, 18]. The architecture of our CNN model is mainly inspired by [48], in which a four-layer CNN with local response normalization achieved an 11% test error rate on the CIFAR-10 dataset [49]. As delineated by Table 1, the configuration of our net is derived from the code shared by Krizhevsky [50]. Our CNN takes as input a character image rescaled to the size of 24 \times 24 pixels and returns as output a vector of \( z \) values between 0 and 1. The input image is converted to grayscale image so as to reduce the susceptibility of our model to variable text colors and alleviate the computational burden. Ten parallel CNNs as described above form the CNN ensemble. They are independently trained and their outputs are averaged to get the final recognition results.

Note that we do not perform the data augmentation as proposed by [48], in which 24 \times 24 patches are randomly cropped from the original 32 \times 32 images in CIFAR-10 [49] to prohibit overfitting. The reason behind this is twofold. On the one hand, the loss of critical information, including radicals and strokes in characters, is inevitable if the original images are randomly cropped. On the other hand, we are not concerned about overfitting because our synthetic dataset can be arbitrarily large.

3.2.1. Details of learning

Stochastic gradient descent with a batch size of 128 images is used to train our models. Parameters like learning rates, weight decay and momentum are concurrent with the shared code [51]. 195,000 images are used for training while the remaining 5000 images are used for
validation. We train each model for only one epoch on the training set, which takes approximately two hours on one NVIDIA Tesla K20Xm GPU.

3.2.2. Visualization
In Fig. 4, we visualize the learned CNN ensemble using the technique as demonstrated [52,53]. It can be observed that the appearance of different shifts and fonts of a specific category is captured in a single image, and ten CNN models in the CNN ensemble learn something slightly different from each other albeit the overall similarity. The visualization indicates that the CNN ensemble has captured distinctive features of characters.

3.2.3. Training the text/non-text SVM classifier
We adopt a linear SVM classifier [54] to determine whether there is a character in a given image patch. The SVM takes the outputs of the local4 layer of the CNN ensemble as its features. The local4 layer of every CNN outputs a 6 × 6 × 32 feature map, which is 1152-dimensional after concatenation. The CNN ensemble consists of 10 CNNs, thus the feature vector of the SVM is 11,520-dimensional. The parameter C of the SVM controls the trade-off between margin maximization and errors of the SVM on training data. C is optimized on the synthetic validation set.

3.3. STBB and SCW detection
In this section, we describe the proposed image operator CWT and how it is applied with the sequence information to detect STBB and SCW.

3.3.1. Character width transform
One feature that distinguishes East Asian text from other elements of a video frame is its consistent SCW. SCWs of East Asian characters are identical as long as their font styles and font sizes are set the same. In this work, we leverage this fact to define CWT, which recovers regions that are likely to contain texts.

CWT is a local image operator. At each local region, CWT generates a histogram that estimates the distribution of SCWs of the subtitle text in this region. SCW is estimated by detecting pixels that are likely to locate at the space between characters and calculating the pairwise distances between these detected pixels. As illustrated in Fig. 5, the randomness at non-subtitle regions makes the pairwise distances distribute uniformly. Meanwhile, at subtitle regions, more pairwise distances come from the space between characters, leading to the emergence of a local peak in the vicinity of the SCW. Based on the distribution patterns of histograms constructed at different local regions, we predicate that the STBB and the SCW can be determined simultaneously.

Detecting pixels at the space between characters requires the binarization of frames extracted from videos (see Fig. 6(b) for illustration). Firstly, each RGB frame with the size of $H \times W$ is transformed into LAB color space to avoid the illumination inference [55]. Then, Sauvola algorithm [56] is adopted to separate text components from background (binarization) for its robustness to the uneven illumination and noise. This algorithm performs local thresholding with $\mu$-by-$\nu$ neighborhood. Both $\mu$ and $\nu$ are set to 150 pixels and the threshold is set to 0.34.

CWT is then applied to every local region in a sliding-window manner. Concretely, a $h \times W$ sliding window (as shown in Fig. 6(c))
is adopted, where $h$ is a variable less than $H$ and determined according to the resolution of videos. This window scans each frame by moving vertically from top to bottom at stride 1, and $H - h + 1$ window regions can be obtained. Finally we acquire $H - h + 1$ histograms by applying CWT at every window region.

Let $x^k_{i,j} \in (0, 1)$ denote a pixel in the binarized frame $k$ where $(i, j)$ are the coordinates. Values of most text pixels are 1 after the binarization. We take the sliding-window region whose top boundary is at position $i$, and the sum of elements in its each column is:

$$v^k_{i,j} = \sum_{j=1}^{i+h-1} x^k_{i,j}.$$  

(1)

After that, pixels that are likely to locate at the space between characters are detected by local-minimum points ($LMPs$). We denote a set of $LMPs$ by $\mathcal{L}_i^k$, where $\mathcal{L}_i^k = \{ x^k_{i,j} \mid v^k_{i,j} < \min(v^k_{i,j-1}, v^k_{i,j+1}) \text{ or } v^k_{i,j} = 0 \}$.

As illustrated by Fig. 7, the majority of $LMPs$ are interspersed among backgrounds as well as the space between characters. If more than 30 $LMPs$ are connected (i.e., $\forall j, 3M \geq 30, x^{k}_{i,j}, x^{k}_{i,j+1}, \ldots, x^{k}_{i,j+M-1} \in \mathcal{L}_i^k$), they will be removed, which can effectively eliminate $LMPs$ from backgrounds while reserve $LMPs$ from the space between characters. The rationality of this constraint is that more than 30 connected $LMPs$ could only come from backgrounds. Then all pairwise distances between $LMPs$ are calculated and stored in a set $D_i^k$:

$$D_i^k = \left\{ [m-n] \mid x^{k}_{i,m}, x^{k}_{i,n} \in \mathcal{L}_i^k, |m-n| < w_{\text{min}} < |m-n| < w_{\text{max}} \right\},$$  

(2)

where $w_{\text{min}}$ and $w_{\text{max}}$ denote the minimum and the maximum SCW respectively.

It is noteworthy that since the statistical information derived from a single frame is too coarse to provide a reliable estimation of SCW, we cannot construct a histogram directly from $D_i^k$ in the next step. This is when the sequence information of video comes in handy. As STBB

Fig. 4. Visualization of 5 character classes learned from the Traditional Chinese character classifier. There are 10 visualization results corresponding to 10 CNN models in each line. These images are generated by numerically optimizing the input image which maximizes the score of a specific character category [52,53].

Fig. 5. Illustration of the distribution patterns of histograms at a subtitle region (window region 2) and non-subtitle regions (window region 1 and 3).

136
and SCW are consistent throughout the video, we assume that values in \( D_1, D_2, ... D_T \) are drawn from the same underlying distribution, where \( T \) represents the number of frames in the video. Based on this assumption, histograms \( U_i(\hat{w}) \) can be constructed from frames throughout the video:

\[
U_i(\hat{w}) = \sum_{k=1}^{T} \sum_{t} I_0(t),
\]

where \( I_0(t) \) equals 1 if \( r = \hat{w} \) and 0 otherwise. In order to alleviate the computational burden, videos are downsampled to 0.0625 fps without compromising the STBB detection performance.

### 3.3.2. Detecting the STBB and SCW

Given histograms \( U_i, U_{i+1}, ... U_{H+1} \), the STBB and the SCW can be determined. Concretely, if the local peaks (see Fig. 5) of several adjacent histograms \( U_i, U_{i+1}, U_{i+2} \) all locate near \( \hat{w} \), \( t \) and \( b \) will be regarded as positions of a set of candidate STBB, and \( \hat{w} \) will be the corresponding SCW. Our algorithm is presented in Algorithm 1, of which the output \( P \) contains several candidate sets of STBB and estimated SCW.

Note that elements contained in \( P \) are raw candidates, some of which might come from non-subtitle regions and should be eliminated. A post processing algorithm are adopted to remove these false-positive candidates: (1) if two candidates with a similar SCW are overlapped, we eliminate the one whose subtitle height is smaller. (2) If two candidates have a similar STBB and the SCW of one of them is approximately two times larger than the other one, the candidate with the larger SCW is eliminated. (3) Candidates whose STBB locate at the upper half of the frame are eliminated due to the fact that most of subtitles are superimposed on the bottom half of the frame.

This post processing algorithm eliminates almost all false detections, and a small amount of surviving false-positives will be further removed by the text/non-text classifier in the step following.

### 3.4. SLBB detection

Raw subtitle regions \( RS \) bounded by the detected STBB and the left/right boundary of original frames are cropped from original frames. The size of \( RS \) is \( h_j \times W \), where \( h_j \) represents subtitle height. Then, SLBB are detected in a sliding-window manner: a \( h_j \times (w - 1) \) window, a \( h_j \times w \) window and a \( h_j \times (w + 1) \) window that respectively slide from left to right across \( RS \) with stride 1 are adopted, where \( w \) is the determined SCW. Then, every window region is classified as either text or background.

![Algorithm 1 STBB and SCW determination](image)

**Algorithm 1 STBB and SCW determination**

**Input:** histograms \( \{U_i, U_{i+1}, ... U_{H+1}\} \),

- maximum SCW \( w_{max} \), minimum SCW \( w_{min} \),
- minimum subtitle height \( min_{height} \)

**Output:** candidate STBB and SCW \( \{P\} \)

1. for \( i = 1 \) to \( H - h + 1 \) do
2. for \( j = w_{min} \) to \( w_{max} \) do
3. \( q_{ij} = 0 \)
4. if \( \max(U_j(i+1), U_j(i+2)) \leq U_j(i) \) then
5. \( \hat{q}_{ij} = j + \frac{1}{2} \times \frac{U_j(i+2) - U_j(i)}{U_j(i+2) - U_j(i+1)} \)
6. end if
7. end for
8. end for
9. \( Q \leftarrow \emptyset \), \( P \leftarrow \emptyset \)
10. for \( i = 1 \) to \( H - h + 1 \) do
11. for \( j = w_{min} \) to \( w_{max} \) do
12. if \( \hat{q}_{ij} > 0 \) then
13. \( Q = Q \cup \{q_{ij}\} \)
14. for \( k = i + 1 \) to \( H - h + 1 \) do
15. \( C = \{x \mid x \in \{q_{ij} - h, q_{ij} + h\}, x > 0\} \)
16. if \( C = \emptyset \) then
17. break for
18. end if
19. \( e = \arg \max_{x \in C} |x - \text{median}(Q)| \)
20. \( Q = Q \cup \{e\} \)
21. end for
22. if \( k - i - h/2 \) is \( \geq \text{min}_{height} \) then
23. \( P = P \cup \{i, k + h/2, i + 1, \text{median}(Q)\} \)
24. end if
25. end for
26. end for
27. end for

![Fig. 7.](image)

The majority of LPM’s are interspersed among backgrounds (denoted by red asterisks) and the space between characters (denoted by green asterisks). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)
region or non-text region by the SVM classifier described in Section 3.2.3. Supposing that \( a_i \) and \( b_i \) respectively denote the left boundary position and the right boundary position of the ith window region predicted as a text region, and there are \( n \) window regions predicted as text regions. Algorithm 2 is designed to merge overlapping window regions predicted as text regions together and subsequently determine the SLRB. According to the output LeftBound and RightBound of Algorithm 2, subtitle region \( S \) is detected by further removing non-subtitle regions on two sides of \( RS \). This process is illustrated in Fig. 8. The parameter \( \beta \) of Algorithm 2 is determined according to the resolution of videos. \( \beta \) being too large would cause the real subtitle region to be easily connected with non-subtitle regions that are incorrectly predicted, while being too small, an integral sentence might be easily broken into pieces.

### 3.5. Subtitle recognition

Now that the subtitle region \( S \) has been successfully detected, we will describe the proposed subtitle recognition scheme with three steps including sliding window based segmentation, window region recognition and dynamic programming determination.

#### 3.5.1. Sliding window based segmentation

In order to recognize each single character in the subtitle, the subtitle region \( S \) must be properly segmented (i.e. split the image text line into patches that each of which contains a single character). This step is challenging due to touching characters and the inherent structure of separation from the left and right sides of many East Asian characters. Unlike other methods where potential segmentation points must be determined precariously \([33,34,37,57]\), our method obviates this step since the SCW is known, which is an inborn advantage of our system. Three sliding windows identical to those in Section 3.4 are adopted again to slide from left to right across \( S \) at stride one, and each window region is fed into the CNN ensemble for recognition.

#### 3.5.2. Window region recognition

Given a window region \((a_i, b_i)\), the softmax layer of each CNN model outputs the probability of each category, and categories whose probabilities are among the top 20 are reserved. Then, probabilities of these reserved categories are averaged across 10 CNN models. If the largest average probability is greater than a threshold (i.e. 0.2), candidate categories of \((a_i, b_i)\) with the top 5 average probabilities will be recorded before moving to the next window position \((a_{i+1}, b_{i+1})\).

### Algorithm 2 SLRB determination

**Input:** \( n \) predicted text window regions \((a_1, b_1), (a_2, b_2), \ldots, (a_n, b_n)\), parameter \( \beta \) controlling the maximum gap between two clauses separated by space, the determined SCW \( w \)

**Output:** the left and the right boundaries of subtitle \( \{ \text{LeftBound}, \text{RightBound} \} \)

```
1: i ← 1, k ← 1
2: LeftCandidate ← ∅, RightCandidate ← ∅
3: while i < n do
4: j ← i + 1
5: right ← \( b_i \)
6: while j ≤ n and \( a_j \) ≤ \( a_i \) do
7: right ← \( \max(\text{right}, b_j) \)
8: j ← j + 1
9: end while
10: if \( j - i > 3 \) then
11: if \( \text{LeftCandidate} = \emptyset \) then
12: RightCandidate[k] ← right
13: LeftCandidate[k] ← \( a_i \)
14: \( k ← k + 1 \)
15: else
16: if \( a_j ≤ \text{RightCandidate}[k - 1] + \beta × \text{SCW} \) then
17: RightCandidate[k - 1] ← right
18: else
19: RightCandidate[k] ← right
20: LeftCandidate[k] ← \( a_i \)
21: \( k ← k + 1 \)
22: end if
23: end if
24: end if
25: \( i ← j \)
26: end while

27: Z ← \( \arg \max (\text{RightCandidate}[i] - \text{LeftCandidate}[i]) \)
28: \( \text{LeftBound} ← \text{LeftCandidate}[Z] \)
29: \( \text{RightBound} ← \text{RightCandidate}[Z] \)
```

Otherwise, the window region \((a_i, b_i)\) would probably reside between two adjacent characters. In this case, it will be abandoned and the next window region \((a_{i+1}, b_{i+1})\) will be examined. Finally, those recorded 5 candidate categories whose probabilities are greater than 0.05 will be stored with their associated recognition probabilities \( R_{\text{prob}} \) and the window position \((a_i, b_i)\).

#### 3.5.3. Dynamic programming determination

The final recognition results are determined by a dynamic programming algorithm. From the leftmost window \((a_1, b_1)\) step by step all the
way to the rightmost window \((a_j, b_j)\), this algorithm builds the whole sentence by repeatedly appending the character in the next window position (i.e. \(w - 2, w - 1\) or \(w\) pixels rightward) to the previously recognized sentence. In each step from the window \((a_i, b_i)\) to the next window \((a_j, b_j)\), every previously recognized sentence that arrives to \((a_j, b_j)\) is processed by a character based 3-gram language model. For every unique 3-gram word group consisting of the newly appended character and two former characters, a recognition probability \(\text{Rprob}\) and a 3-gram language probability \(L\) are recorded, based on which the total score of the word group is calculated as:

\[
groupscore_{i,j} = \gamma \times \log(Lscore) + (1 - \gamma) \times \log(\text{Rprob}), \tag{4}
\]

\(\gamma\) is the proportion of the language score and the recognition score which is 0.3 in our experiment. Since the sliding window has three widths (i.e. \(w - 1, w\) and \(w + 1\)), it is possible to obtain several identical word groups that arrive at \(b_j\) but with different scores during the building process. Therefore, a pruning strategy that only reserves the word group with the highest score is applied to reduce the redundancy and improve the efficiency. The building process terminates when \(b_j\) approaches the right boundary of the image, and the total score of the \(k\)th possible sentence is:

\[
totalscore_k = \sum \text{groupscore} \times \text{windows}(k), \tag{5}
\]

where \(\sum \text{groupscore}\) represents the sum of all \(\text{groupscore}\) in the \(k\)th candidate sentence and \(\text{windows}(k)\) represents the number of windows (i.e. characters) in the \(k\)th candidate sentence. The sentence with the highest total score is selected as the final recognition result.

4. Experiments

We conduct ample experiments to evaluate each component of the proposed system. The end-to-end performance of our system is also reported in this section.

4.1. Dataset

As listed in Table 2, an extensive dataset containing 1097 videos in Simplified Chinese, Traditional Chinese and Japanese is constructed. These videos exhibit a wide range of diversity in TV program genres, including talk shows, documentaries, news reports, etc.

STBBs of all videos and SLRBs of videos marked by \(\dagger\) are annotated manually. As our recognition module is almost error-free, the recognition results of videos marked by \(\dagger\) are annotated by a human annotator “A” on the basis of the outputs of the proposed system. The annotations obtained in this manner are regarded as ground truth. To test the quality of the ground truth annotations, we randomly select 400 frames containing 4494 characters from the already annotated frames and employ another two human annotators “B” and “C” to annotate these frames independently again. By comparing the annotations from “B” and “C”, the final agreement on the result is reached, based on which the annotations from “A” are examined. The annotations from “A” achieve 99.8% accuracy, indicating that the ground truth annotations are of high quality.

We also measure the human-level reading performance on these 400 frames. A human annotator “D” is employed to annotate these frames manually, and the annotations from “D” are examined based on the final agreement mentioned-above. The human-level reading performance is estimated by the performance of “D”, of which the reading accuracy is 99.6%.

### Table 2

Our dataset configuration. All videos are utilized to evaluate the STBB detection module, while only videos marked by \(\dagger\) are randomly selected to evaluate the remaining modules and the end-to-end system.

<table>
<thead>
<tr>
<th>Language</th>
<th>#Videos</th>
<th>Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traditional Chinese</td>
<td>1015(40)</td>
<td>480 \times 320</td>
</tr>
<tr>
<td>Traditional Chinese</td>
<td>40</td>
<td>852 \times 480</td>
</tr>
<tr>
<td>Simplified Chinese</td>
<td>40(40')</td>
<td>852 \times 480</td>
</tr>
<tr>
<td>Japanese</td>
<td>2</td>
<td>480 \times 320</td>
</tr>
</tbody>
</table>

### Table 3

Parameter \(h\) optimization. STBB detection precision is not presented for the reason that false-positives are subsequently removed by the text/non-text classifier. Therefore, every video only has one final subtitle location. Note that the correctness of STBB determination always entail the correctness of SCW determination, hence only the former is reported. This step is not compared to any baseline, as there is no previous work tackling the STBB and SCW determination problem to the best of our knowledge.

<table>
<thead>
<tr>
<th>Video resolution</th>
<th>Number of videos</th>
<th>h</th>
<th>Number of videos whose STBB are correctly detected</th>
<th>Recall</th>
</tr>
</thead>
<tbody>
<tr>
<td>480 \times 320</td>
<td>1017</td>
<td>1</td>
<td>972</td>
<td>95.6%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
<td>980</td>
<td>96.4%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>951</td>
<td>93.5%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>934</td>
<td>91.8%</td>
</tr>
<tr>
<td>852 \times 480</td>
<td>80</td>
<td>3</td>
<td>73</td>
<td>91.3%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>5</td>
<td>75</td>
<td>93.8%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>7</td>
<td>75</td>
<td>93.8%</td>
</tr>
</tbody>
</table>

4.2. Experiments on STBB and SCW detection

In order to demonstrate the efficacy of our method, all videos in the dataset are selected for evaluation. In the experiment, the height of the vertical sliding window \(h\) is optimized with regard to videos with 480 \times 320 resolution and videos with 852 \times 480 resolution respectively.

The CNN ensemble trained on synthetic data with random shift empowers our system with high robustness even if the STBB are not precisely detected. For this consideration, our evaluation method is defined as follows: the STBB of a video are detected correctly if

\[-3 \leq T_g - T_{gt} \leq 2\text{ and } -2 \leq B_{gt} - B_g \leq 3,\]

where \(T_g\), \(T_{gt}\), \(B_g\) and \(B_{gt}\) denote positions of detected top boundary, ground-truth top boundary, detected bottom boundary and ground-truth bottom boundary respectively.

We perform a series of tests to determine the optimal value of parameter \(h\) (the height of the proposed vertical sliding window in Section 3.3.1) by 5-fold cross validation on the whole dataset. The input variables \(w_{min}, w_{max}\) and \(\text{min height}\) of Algorithm 1 are also chosen by 5-fold cross validation and set to 5, 40 and 12 respectively. Table 3 shows the performance of our STBB detection module with regard to different \(h\). The variable \(h\) actually controls the trade-off between the STBB detection accuracy and the tolerability to noise. From our experiments, we observe that when \(h\) is too small, the histogram becomes more susceptible to background noise as well as strokes inside characters that do not reflect SCW. But \(h\) being too large would compromise the STBB detection accuracy.

4.3. Experiments on SLRB detection

In this section, the performance of our SLRB detection module is evaluated against two baseline methods based on hand-engineered features: T-HOG [27] and EOH-GSC [26]. The input parameter \(\beta\) of Algorithm 2 is set to 0.7/2.5 for videos in 480 \times 320/852 \times 480 resolution respectively.

Our evaluation method is quite similar to the ICDAR’03 detection protocol [58]. Let \(r\) denote the ground-truth SLRB, and \(r'\) denote the corresponding detected SLRB. The average match \(m_{ave}\) between all \(r\)
and \(r'\) in a video is defined as twice the length of intersection divided by the sum of the lengths:

\[
m_{\text{ave}}(r, r') = \frac{2 \sum_{E(r) \cap E(r')} (L(r) + L(r'))}{\sum_{E(r) \cap E(r')} (L(r) + L(r'))},
\]

(7)

where \(L(r)\) is the distance between a set of left and right boundaries and \(E\) denotes all the ground-truth SLRBs in a video.

Table 4 lists the statistics of \(m_{\text{ave}}\) of 80 videos and shows the superiority of our CNN features over T-HOG [27] and EOH-GSC [26] features on the text/non-text classification task.

### 4.4. Experiments on subtitle recognition

This section measures the performance of our character recognition module. For comparison, we test the same 80 videos in the previous section with Grayscale based Chinese Image Text Recognition (gCITR) [34] as well as another two commercial OCR software: ABBYY FineReader 12 [59] and Microsoft OCR library [60]. gCITR [34] is the previous state-of-the-art system for Simplified Chinese subtitle recognition, where 85.44% word accuracy is achieved on another dataset. Besides, the performance of a single CNN is also reported in order to manifest the efficacy of the CNN ensemble. Two annotators spend one week, eight hours a day, labeling the ground truth recognition results of these 80 videos.

The performance of our subtitle recognition module is evaluated by the word accuracy \(W_{\text{acc}}\) that is defined as:

\[
W_{\text{acc}} = \frac{N - E_{\text{dis}}}{N},
\]

(8)

where \(N\) is the number of ground-truth words and \(E_{\text{dis}}\) represents Levenshtein edit distance [61] to change a recognized sentence into ground-truth.

Tables 5 and 6 show the performance of ABBYY [59], gCITR [34], Microsoft OCR library [60], our single CNN and the CNN ensemble on the Simplified Chinese and Traditional Chinese text line recognition task. The performance of the proposed method exceeds other baselines by a large margin. In order to demonstrate the efficacy of our system on other languages, we also test it on two videos in Japanese, and an average 97.4% \(W_{\text{acc}}\) is achieved.

### 4.5. End-to-end performance

The same 80 videos in the previous section are selected for evaluating the end-to-end performance. Table 7 compares the end-to-end performance of the proposed system with ABBYY [59], gCITR [34], Microsoft OCR [60].

### 5. Discussion

The dataset used for the experiment contains extreme cases like cluttered backgrounds, illumination changes and loss of resolution that are encountered in real-world videos. Although the STBB detection...
module has achieved competitive performance, there is still room for improvement. We observe that a majority of incorrectly detected STBBs locate near the ground-truth boundaries (Fig. 9). Actually, more accurate boundary positions can be obtained if some regression methods like the one in [6] are adopted. In the SLRB detection module, it is observed that specific characters are sporadically misclassified as non-texts. We find the strokes of these characters are all very sparse, which can be easily confused with edge or texture features at backgrounds (Fig. 10). Confusion and loss of radicals and strokes are two major mistakes made by the CNN character recognizer (Fig. 11). Character categories that are misclassified more than three times are examined and the causes of the errors are scrutinized. We find that 45.5% of the errors are caused by resemblances between two characters, 33.2% are caused by cluttered backgrounds, 18.2% are caused by the incorporation of the language model and 3.2% are caused by large vertical shifts of characters.

6. Conclusion

In this paper, we exploit the distinctive features of East Asian characters (consistent character width, subtitle top and bottom boundary position, and color) and present an novel end-to-end subtitle text detection and recognition system specifically designed for videos with subtitles in East Asian languages. By applying CWT and integrating the sequence information throughout the video, we are able to detect STBB and SCW simultaneously. This represents a departure from scene text detection problem where sophisticated methods are designed to detect texts in a single image. A CNN ensemble is leveraged to classify East Asian characters into thousands of categories. Our models are trained purely on synthetic data, which makes it possible for our system to be retrained on other languages without requiring human labeling effort. Our system, as well as each module in it, compares favorably against existing methods on an extensive dataset. The near-human-level performance of our system qualifies it for practical application. For example, our system can provide accurate and reliable text labels for speech recognition researches, since video subtitles are synchronous with speech in videos.

In future work, this system will be tested on videos in Korean or other languages with consistent SCW.

Acknowledgments

This work is supported by Microsoft Research under the eHealth program, the National Natural Science Foundation in China under Grant 81771910, the National Science and Technology Major Project of the Ministry of Science and Technology in China under Grant 2017YFC0110903, the Beijing Natural Science Foundation in China under Grant 4152033, the Technology and Innovation Commission of Shenzhen in China under Grant shenfagai2016–627, Beijing Young Talent Project in China, the Fundamental Research Funds for the
Central Universities of China under Grant SKLSE-2017ZX-08 from the State Key Laboratory of Software Development Environment in Beihang University in China, the 111 Project in China under Grant B13003. We would like to thank Jinfeng Bai for conducting the gCTR baseline experiment.

References


