Text Line Extraction in Natural Scene Images

汪, 留安

Kyushu University

Doctoral Thesis

Text Line Extraction in Natural Scene Images

Author:

Liuan Wang

Supervisor:

Prof. Seiichi Uchida

A thesis submitted in fulfilment of the requirements

for the degree of Doctor of Philosophy

in

Department of Advanced Information Technology

Graduate School of Information Science and Electrical Engineering

August 2018

Abstract

In this thesis, we study a text line extraction method with user-intention and complete text line extraction methods. As a crucial prerequisite, text line extraction plays an important role in Optical Character Recognition (OCR) applications. Text line extraction with user-intention focuses on user-interested text line extraction to improve the user experience. The text size, skew angle of text line, and reduction ratio are estimated to perform the text line extraction. Then, the user-intention text line path is accumulated according to the local and global permutation relationship of text characters. The complete text line extraction formulates the text line extraction problem as a global optimization solution by taking advantage of the particular structure of constructed directed graph upon the text characters. The directed graph imitates the human reading sense and left to right text spatial arrangement. The text line paths are included in the directed graph, which constructs the relationship and eliminates the disorder of candidate text characters. The text line path global optimization produces the text line paths with minimum cost value iteratively by augmenting previous text line paths. Global optimization can identify the text line number automatically to avoid exhaustive searching, and it benefits various appearances of text lines. Furthermore, we improve the text line extraction performance with complementary usage of red, green, and blue channels. Experimental results on ICDAR2011 demonstrate that we achieved a promising f-measure as high as 0.825, which is competitive with state-of-the-art methods.

Acknowledgements

The past three years have been among the most important and unforgettable times in my life. I have received extensive help from many kind people, and I would like to convey my sincere thanks to them.

First of all, I would like to convey my sincere thanks to my supervisor, Prof. Seiichi Uchida. I still cannot forget the moment he offered me the chance to join his laboratory. At the start of my research, he discussed the potential topics of my research, and his insight and kindness impressed me. Then, we decided on an interesting research topic: path global optimization-based text line extraction. During my research, I received many valuable suggestions, which were useful to my work. During the writing of papers, he also taught me how to organize an excellent paper. Aside from teaching me how to conduct research, he also taught me how to deliver an excellent presentation to an audience. His opinions and suggestions instruct me a lot for my future.

Secondly, I would like to give my sincere thanks to the committee members of my thesis defense, Prof. Taniguchi and Prof. Morooka. They spent valuable time helping me with my thesis.

Thirdly, I would like to convey my appreciation to Jun Sun. He is the pathfinder and leader for my doctorate career and life. Without his help, I would not have been able to achieve such fruitful success and enjoy a colorful life. I would like to give my thanks to everybody for their kindly help. Feng Xu helped me transfer my tuition fee. Yuan He, Wei Fan, Song Wang, and Anna Zhu gave me lots of valuable comments on my research. Mika Maeda helped me communicate with the university office. I've had a really wonderful time because of all of you!

Finally, I would like to give my great thanks to my parents, my wife, and my son. It's my parents who give me all of their love from a new born baby. They believe and support me whenever I'm in trouble. Thank you teach me how to be a kind, hardworking, and responsible man. I want to give my sincere thanks to my wife. Your encouragement and support are my greatest motivation! Life is beautiful because of you.

Contents

Chapter 1

Introduction

1.1 Background

With the growing requirement and rapid increasing usage of image capturing devices, texts embedded in images and text line extraction have received increasing attention in recent years. Up to now, a large number of approaches have been proposed to extract the text in images. As mentioned in [1, 2], text in images and videos contain exact semantic information (e.g., advertisements, street names, road signs). Sematic information can describe image content accurately and make the image "communicable" and "readable" by devices. Therefore, the text in images can be utilized in many image recognition and understanding applications, such as multilingual translation [3], book digitization [4], and image indexing and retrieval [5].

In general, a typical OCR system [6] consists of three steps. The first step is text line extraction to localize the exact text line boundary. The second procedure is to segment each symbol in the extracted text lines. Finally, the segmented symbol can be fed into an OCR recognition engine [7] to recognize and decode the text lines to

device-readable symbols. The text line extraction localizes the exact text bounding box by grouping individual characters into text lines according to character similarities. The text character similarities include color [8, 9], gradient [10–12], stroke [13–15], and texture [16–18], which can discriminate the text characters from a complex background.

However, fast and accurate text line extraction is a challenging task due to the large diversity of text fonts, orientation, low contrast, and complex background, which are difficult to handle by using conventional methods. Several text line examples from the International Conference on Document Analysis and Recognition (ICDAR) competition database are shown in Figure 1.1. Generally, text line extraction can further be decomposed into three primary steps, namely, candidate text character detection, false text candidate removal, and text line accumulation. Candidate character extraction attempts to extract the connected component of characters in different ways [13, 19, 20]. The connected component is a connected region that has only two possible values for each pixel. In false text candidate removal, noise text candidates are removed by discriminant features extracted on the image character patch [21, 22]. Finally, the text candidates are "greedy" accumulated into text lines [23–25].

The conventional text line extraction methods suffer from the following typical drawbacks:

- The user may be interested in only some text line information in the image rather than all the text line information. The text line extraction for all text lines will increase complexity and decrease efficiency. Meanwhile, the redundant text line information maybe encumbrance and confuse users.
- The conventional methods often suffer from numerous false positives in the candidate text character extraction step to achieve approved recall. Obviously, it is

FIGURE 1.1: Text line examples from International Conference on Document Analysis and Recognition (ICDAR) database.

hard to remove all the false positives for a high precision and the non-text noises will degrade the text extraction precision and recall. To make matters worse, the detected character candidates are independent of each other, leaving a challenging task to text line extraction.

- In the text line extraction step, most existing techniques accumulate the detected text components greedily to text lines by using knowledge-based heuristic rules [26]. The text line extraction may fail and affect the robustness when the scenes change for real applications.

- The conventional methods have difficulty determining the number of text lines in images due to unknown text line information. The situation is exacerbated by complex non-character noises, and exhaustive searching will lead to a low text extraction precision.

In conclusion, due to the drawbacks and challenges in text line extraction, new methods expect text line extraction with user-intention, construction of relationship among character candidates, text line paths optimization, and automatic determination of the

number of text lines. With these issues in mind, we firstly perform text line extraction with user-intention. The character size, skew angle of text line, and reduction ratio are estimated to extract the text line in a sub-region of an image. Then, text candidate components are accumulated to a text line according to local and global relationships.

We also propose a complete text line extraction method based on the global optimal of text line paths algorithm. Complete text line extraction extracts all the text lines automatically without any user interaction. In globally optimal text line extraction, unlike the conventional methods with unstructured text candidates and exhaustive searching, we integrate all the candidate text components into a directed graph inspired by the human reading sense. The directed graph can efficiently describe the human reading habit and the corresponding relationship of each characters in a text line. Besides, text line paths optimization can avoid exhaustive searching and ignore the potential noise paths similar to the text line path planning. Through these operations, the proposed globally optimal text line extraction is supposed to solve the problems of text line extraction.

1.2 Conventional Methods — Region-based Methods

In the literatures, a large number of state-of-the-art text line extraction methods have been proposed. These methods can be roughly classified into two categories [27]: region-based and connected components (CC)-based methods.

Region-based methods attempt to extract discriminative hand-crafted features in sliding windows and classify the local windows into text and non-text regions by using well-trained classifiers. Ye et al. [28] combined detection process, color, texture, and OCR statistic features to discriminate texts from non-text patterns. These text properties are used to group text pixels into candidate text lines. Hanif et al. [29] proposed a complete

text localization boosting framework integrating feature and weak classifier selection based on computational complexity to construct the text detectors, and a neural network to learn the necessary rules for localization. Minetto et al. [30] described a robust and accurate multi-resolution approach to detect and classify text regions in scenarios. The segmented regions are filtered out by using shape-based classification, and neighboring characters are merged to generate text hypotheses.

Li et al. [31] implemented a scale-space feature extractor that feeds an artificial neural processor to detect text blocks, which can be used for tracking and refinement. Zhong et al. [32] localized captions in JPEG compressed images and the I-frames of MPEG compressed videos by using the intensity variation information encoded in the DCT domain. Gllavata et al. [33] applied a wavelet transform to images and calculated the distribution of high-frequency wavelet coefficient to statistically character text and non-text areas. Ye et al. [34] calculated a wavelet energy feature to locate text pixels and connected these pixels into regions by a density-based region growing method. Pan et al. [35] extracted candidate text regions by a boosted region filter in each pyramid layer. Then, the text lines are segmented by multi-orientation projection analysis.

Shivakumara et al. [36] presented new Fourier transform-based features in RGB space with statistical features, which are subjected to K-means clustering. Kim et al. [37] generated a transition map based on transient colors for overlay text regions between inserted text and its adjacent background. Goto et al. [38] grouped homogeneous regions to avoid multiple and redundant speech syntheses or braille conversions to a wearable camera system for the blind. Ye et al. [39] leveraged both the appearance and consensus of connected components, which were represented with color and spatial layout features. Gomez et al. [40] built a perceptual organisation framework. It exploited collaboration of proximity and similarity laws to create text hypotheses. Sung et al. [41]

performed extremal region tree construction, sub-path partitioning, sub-path pruning, and character candidate selection sequentially. Then, heuristics were used to group the character candidates into text lines. Risnumawan et al. [42] introduced an ellipse growing process based on a nearest neighbor criterion to extract text components.

Zhang et al. [43] exploited the symmetry property of character groups and allowed for direct extraction instead of single characters or strokes properties. Huang et al. [44] claimed a novel framework to tackle the low-quality texts by taking advantage of the Maximal Stable Extremal Region (MSER) and sliding window-based method. The MSERs operator can dramatically reduce the number of windows scanned, and the sliding window with convolutional neural network (CNN) can correctly separate the connections of multiple characters in text components. Lee et al. [45] extracted six different classes features of text, and used AdaBoost with multi-scale sequential search. Wang et al. [46] presented an end-to-end scene text recognition with a two-stage pipeline consisting of text detection followed by a leading OCR engine. Coates et al. [47] applied large-scale algorithms to learn the features automatically from unlabeled data to construct highly effective classifiers for detection. Wang et al. [48] combined the representational power of large, multi-layer neural networks to train highly accurate text detector modules.

In the above trials of region-based methods, a large number of windows need to be classified to achieve approved recall. Therefore, the methods are often computationally expensive with complex classification methods. Moreover, region-based methods are very difficult for scene text images with complex backgrounds to remove all the false positives to achieve a high precision, and non-text noises will degrade the text extraction precision. To make matters worse, there is no relationship and structure for independent text candidates, it will leave another heavy task for text line extraction.

1.3 Conventional Methods — Connected Components-based Methods

As opposed to the region-based methods, CC-based methods attempt to extract connected components from images, and then group them into text lines. Wang et al. [49] generated the text lines from hierarchical edges reconstruction and cut by local linearity in the MSER spanning tree, and then the cut multi-components are re-connected based on the text line energy minimization in terms of text line consistency and fitting error. Yin et al. [50] extracted MSERs as character candidates using minimizing regularized variations, and then the candidates are grouped into text lines by the single-link clustering algorithm. Tian et al. [51] proposed a unified scene text detection system named text flow by utilizing the minimum cost flow network model, which can solve the error accumulation problem at both character and text line levels effectively.

Wang et al. [52] segmented character connected components by markov random field with local contrasts, colors, and gradients in multi-channels. Liolios et al. [53] improved the connected components method for skew detection accuracy. Fletcher et al. [54] generated the connected components and applied Hough transform to group them into logical character strings that can be separated from the graphics. Tsai et al. [55] generated a count of connected components extracted from the respective scale set for each spatial bin from the multiple level image.

Shi et al. [56] formulated the text detection as a bi-label (text and non-text regions) segmentation problem. It used a undirected graph model built upon MSERs to incorporate various text information sources into one framework, and the cost function could be optimally minimized via graph cut algorithm to get the final MSERs labeling results. Lee et al. [57] generated text lines through a modified K-means clustering algorithm

by referring texture features, edge information and color information. Pan et al. [58] designed a text region detector to segment and estimate the text confidence, and then grouped the text components into text lines with a learning-based energy minimization method. Koo et al. [59] trained an AdaBoost classifier to determine the adjacency relationship and cluster CCs by using their pairwise relations. Neumann et al. [60] applied an exhaustive search with feedback loop to group extremal regions into words and select the most probable character segmentation. Li et al. [61] separated the text and non-text interferences by leveraging the surrounding context in an information-theoretic fashion, and then yielded the text lines by minimizing the energy cost function.

Although CC-based methods have been proposed with promising performance, those methods still have drawbacks when applied to text line extraction. Firstly, the CC-based methods group the text components into text lines greedily until all positives are identified, it is hard to determine the number of text lines in images due to the unknown text line information, and the exhaustive searching will lead to a low precision of text line extraction. Secondly, the "greedy" grouping usually applies geometrical spatial arrangement or heuristics rules of text components. It may fail when the scenes change, which will impact the robustness in real applications. Therefore, the new methods should overcome the mentioned drawbacks in text line extraction.

1.4 Contribution and Organization of the Thesis

Unlike conventional trials of text line extraction, a text line extraction with user-intention and complete text line extraction methods are proposed to address these limitations. The text line extraction with user-intention focuses on user interested text line only. While, the complete text line extraction is to extract all the text lines in the image

without user interaction. The text lines are extracted by a path optimization strategy instead of extracting individual text line one by one through geometric grouping. We can expect that this approach is beneficial to avoid inconsistent extraction results.

The contributions include the following:

- The text line extraction with user-intention focuses on user interested text information, thereby decreasing the complexity of text line extraction, and the user interaction will improve the performance of text line extraction.
- The human reading sense-based directed graph can effectively construct the spatial relationship structure among adjacent character components and eliminate the disorder of candidate character components in the graph model.
- The path optimization of text lines is similar to the human reading ability in planning of a text line path sequentially. We demonstrate that the text line extraction can be solved by text line paths optimization with intrinsic consistency of the same text line.
- The text line paths optimization can produce the text lines iteratively to avoid exhaustive searching and obtain the global optimized number of text lines.

In summary, the text line extraction user-intention and complete text line optimization are good solutions for the limitations in the text line extraction. Experimental results demonstrate the effectiveness of the text line paths optimization method in comparison with state-of-the-art methods. In the following part, the text line extraction with user-intention and global text line paths optimization-based methods will be introduced, and the analysis will also be conducted.

The remainder of this thesis is organized as follows: Chapter 2 describes the text line extraction with user-intention on an image, which focuses on a single user interested text line. Chapter 3 introduces the directed graph construction upon the extracted text candidates for text line extraction. The detailed analysis of directed graph construction is presented in order to find out how it eliminates the disorder of characters and builds the relationship of candidates. In Chapter 4, we describe the k-shortest paths global optimization based text line extraction on the constructed text directed graph. The performances on widely used ICDAR2011 and ICDAR2013 database are demonstrated in order to show the effectiveness of the global text paths optimization based method in the text line extraction task. In Chapter 5, the text line path optimization is extended to different channel combinations to improve the performance. In Chapter 6, the optimization is further applied to improve the channel combination efficiency of text line extraction. In Chapter 7, we conclude thesis.

Chapter 2

Text Line Extraction with User-intention

2.1 Introduction

Text line extraction with user-intention pays attention to user interested text line rather than all appearance of text lines in the image. The user-guided text line extraction plays an important role in human computer interaction. Commercial mobile OCR translation App [62] has been released with user-interested text line. Text line extraction with user-intention is the first step for these applications.

Our particular focus in this chapter is reliable scene text line extraction with user-intention in terms of swipe and tap gesture as shown in Figure 2.1. In case of swipe gesture, the bounding box of swipe gesture is used as an initialized user-intention text line region. In case of tap gesture, the tap gesture is converted into a swipe gesture with default width of 100 pixels and height of 1 pixel when the width of an image is larger

FIGURE 2.1: Text line extraction examples with user-intention of tap and swipe gesture.

than the default size. The text line extraction provides user interested text line rather than detecting the whole text content.

In the followings, a fast and accurate text detection with user-intention is introduced, the flowchart of which is presented in Figure 2.2 (tap gesture is used for demonstration). Firstly, we design a sub-region detector to extract a sub-region of the target image based on the estimated size of character, skew angle of text line, and reduced ratio. The yellow mask in Figure 2.2 illustrates the sub-region. Secondly, the sub-region is decomposed into connected components, whose heuristic and texture features are fed into a cascade Gentle Adaboost classifier to eliminate the non-text elements. Then, the connected component overlapping with user-interaction is defined as seed character component. If no seed character component is extracted, the width of the user-interaction gesture will be enlarged until at least one seed connected component is extracted. Finally, candidate texts that satisfy the local and global permutation relationship with seed character components, are accumulated to a user-intention text line.

The proposed method is supposed to have the following advantages in comparison with conventional methods:

- Text line extraction can incorporate with user interaction by different gestures in

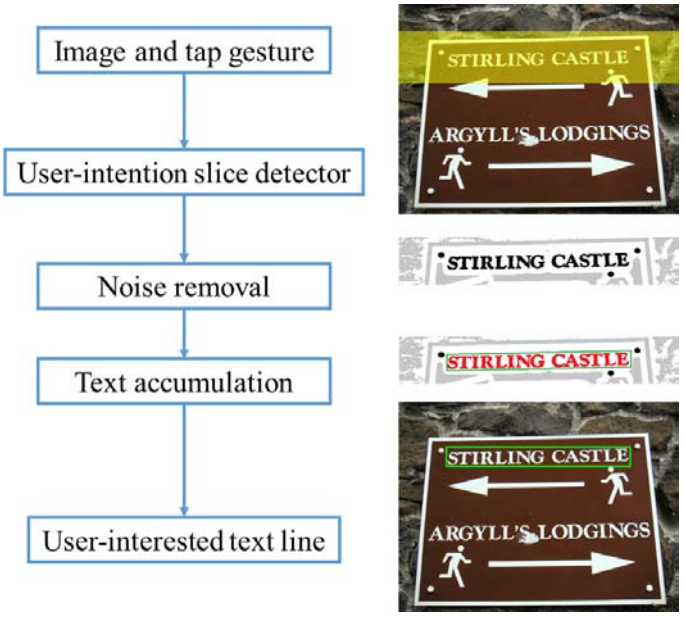


FIGURE 2.2: Flowchart of the proposed system and samples in each step.

terms of swipe and tap, which will focuses only on the user-interested text line to improve the user experience.

- Text line extraction with user-intention produces text lines in a local region of the target image, which will decrease the computation complexity of text line extraction.

- Text line extraction with user-intention will generate promising text line extraction performance with human help and interaction.

2.2 Methodology of Text Line Extraction with User-intention

With the improving performance of mobile devices, high-resolution images are captured, and fast text line extraction becomes a challenging task due to the limited computing ability of mobile devices and large variation of text size, especially small text. Users do not want to receive any delayed response from the service. Therefore, we aim to cut

a sub-region of an image based on user-intention to provide a fast and accurate text detection service without reducing the resolution of images for mobile devices.

2.2.1 Sub-region detector

Given a captured image and a user tap or swipe interaction, the top and bottom locations of a sub-region are calculated according to the center of user-intention and the skew angle of the target text line. Then, we expand the top and bottom region by the increment and the character size of the target text line. The sub-region detector extracts a sub-region R_s with user-intention interaction

$$R_s = p(s, \varphi) + \xi, \tag{2.1}$$

where $p(\cdot)$ defines a region according to the given parameters. s, φ, and ξ denote the estimated character size, skew angle of the target text line, and increment, respectively.

We estimate these text line parameters on local user-intention regions rather than the whole image to speed up the operation. Firstly, edge connected components are generated with adaptive threshold based on the initialized user-intention text line region. Two advantages exist; the edge character component is suitable for two polarity texts, namely, black text on white background and white text on black background. Meanwhile, the adaptive threshold is appropriate for different quality images, especially low contrast images. Double-edge symmetry of edge [63] is utilized to filter out the non-text edges.

We employ the cascade windows to estimate the text size as shown in Figure 2.3. The cascade windows comprise a series of windows with various levels. If no text is detected

FIGURE 2.3: Double edge symmetry with cascade window.

in the current level, then, the size of window is enlarged, which is adaptive for various text scales. All the cascade windows are normalized to a resolution 100 × 100. The procedure is summarized as follows:

- Step 1: Initialize the first level of the cascade window and user region to a normalized size.
- Step 2: Normalize the current window and extract the edge character components using adaptive canny threshold.
- Step 3: Obtain the edge character components overlapping with user region.
- Step 4: When the detected texts touch the border, enlarge cascade window, and go to step 2 until the detected text is located far from the border.

2.2.2 Estimation of skew angle

The skew angle is a crucial parameter for text line with user-intention. The optimal skew angle is calculated by minimizing an energy cost function.

Firstly, the image coordinate is converted to a world coordinate with a central point of detected edge character component set as zero coordinate, and then rotate the coordinate axis to calculate the energy of each rotation. Given a candidate edge x and the detected text character components $X(x \in X)$, we define the energy $E_x(\theta)$ as the projection

location of x in the given skew angle θ. The estimated skew angle of the target text line φ is minimized as follows:

$$\varphi = \arg\min_{\theta} \sum_{i=-L}^{L} |i| \cdot N(E_x(\theta)), \tag{2.2}$$

where L denotes the interval of projection, and $N(\cdot)$ calculates the number of detected text edges that fall to a corresponding projection axis.

On the basis of the estimated size, orientation, and skew angle with increment ξ, we can extract the user-interested text line in the sub-region of an image, and the sub-region contains all the user-interested text. Finally, the sub-region of the image is reduced to a normalized text size by the reduction ratio r defined as follows:

$$r = \frac{\min(w_s, h_s)}{\|S\|}, \tag{2.3}$$

where $\|S\|$ denotes the normalized text size, and w_s and h_s denote the width and height of the detected seed character components, respectively.

Through this operation, the processing time of text line extraction is reduced considerably in comparison with text detection on the whole image without accuracy loss. The larger resolution of image and text size become, the more we can speed up the processing time. This characteristic is suitable for high-resolution image and low computing ability of mobile device.

2.3 Elimination of Noise Connected Components

A set of candidate character components is extracted by using Adaptive Sauvola method [20], and then all the candidate character components are classified into two categories: text character and non-text noise. Two-level features, fast heuristic features and accurate texture feature, are fed into Gentle Adaboost classifier, which was selected because of its fast and good classification performance.

2.3.1 Heuristic and texture features

The heuristic features are designed to effectively discard a large proportion of non-text candidates with very small computational expense, while the texture feature is useful to remove noise non-texts accurately. Heuristic features comprise the aspect Ratio, occupy ratio, margin ratio, mean gray value ratio, stroke width ratio, and variance of stroke width. The texture feature of Histogram of Oriented Gradient (HOG) is extracted for the Gentle Adaboost classifier.

- Aspect Ratio (AR): AR is the ratio between the width and height of a connected component and defined as:

$$AR = \max(w/h,\, h/w), \tag{2.4}$$

  where h and w are the height and width of the connected components, respectively. For a character component, h has no huge difference from w and thus $AR \sim 1$. For noise, there might be a large deviation from 1.

- Occupy Ratio (OR): OR is the ratio between the number of foreground pixel and the number of total pixels, and defined as:

$$OR = N(f)/N(c), \tag{2.5}$$

where $N(f)$ and $N(c)$ represent the number of foreground pixels and the number of total pixels of the connected component, respectively. A character component has no huge difference of OR. For noise, there might be a large deviation from OR.

- Margin Ratio (MR): text character components often share a uniform margin ratio, which is similar to occupy ratio:

$$MR = N(m)/N(f), \tag{2.6}$$

where $N(m)$ and $N(f)$ represent the number of margin pixels and the number of foreground pixels, respectively.

- Mean Gray value Ratio (MGR): MGR represents the difference between the foreground and the background

$$MGR = \max(\overline{g(f)}/\overline{g(b)}, \overline{g(b)}/\overline{g(f)}), \tag{2.7}$$

where $\overline{g(f)}$ and $\overline{g(b)}$ are the average gray value of foreground and background pixels, respectively.

- Stroke Width Ratio (SWR): Text character is composed of strokes, and the Mean value of Stroke Width (MSW) is often stable with the width and height of the connected component.

$$SWR = \frac{MSW}{w + h}. \tag{2.8}$$

- Variance of stroke width: Candidate character components with large stroke width variance are likely to be non-text noise.

- Histogram of Oriented Gradient (HOG): The good performance of HOG has been demonstrated in many fields by characterizing the distribution of local intensity gradient. The HOG feature in terms of 144 dimensions is sent to Gentle Adaboost to classify the candidate character components accurately.

2.3.2 Gentle Adaboost

To reduce computation and ensure good classification ability, the Gentle Adaboost classifier is used to discriminate the text and non-text noise with the extracted heuristic and texture features. Instead of building a complex classifier, the boosting algorithm attempts to select a cascade of better performing weak classifiers from a pool of weak classifiers and construct a strong classifier by using a linear combination of simple classifiers. The candidate CC is discarded when any classifier rejects it. Therefore, the overall detection rate D and false positive F can be summarized as follows:

$$\begin{cases} D = \prod_i^M d_i \\ F = \prod_i^M f_i, \end{cases} \tag{2.9}$$

where d_i and f_i denote the detection rate and false alarm rate of the i^{th} layer with M classifiers in the cascade, respectively.

The Gentle Adaboost uses weighted least-squares regression rather than fitting a class probability to provide a more stable and reliable ensemble. The Gentle Adaboost classification algorithm [64] is described as Algorithm 1.

(a) (b) (c)

FIGURE 2.4: Noise candidate character components elimination. (a) Binarization image; (b) Fast noise removal by Gentle Adaboost with heuristic features; (c) Accurate noise removal by Gentle Adaboost with HOG feature.

Algorithm 1 Gentle Adaboost classification algorithm.

Input:

$S = \{s_1, s_2, \cdots, s_n\}$ with $s_i = (x_i,\ y_i)$ as the training data and the maximum number of classifiers M.

Output:

A classifier $H(x)$ suited for the training data.

1: Initialize the weights $w_i = 1/n, i \in \{1, 2, \cdots, n\}$.

2: **for** classifier $i = 1$ to M **do**

3: Train $H_m(x)$ by weighted least-squares of $(x_i,\ y_i)$ with weight w_i.

4: Update $H(x) = H(x) + H_m(x)$.

5: Update $w_i \leftarrow w_i \cdot e^{(-y_i H_m(x_i))}$ and normalize weights to $\sum_i^n w_i = 1$.

6: **end for**

7: **return** Output: $H(x)$ =sign($\sum H_m(x)$)

We first use the boosting training scheme to train a strong classifier by using a linear combination of heuristic features. These simple heuristic features can filter out a large amount of non-text noises roughly with less computation, and each weak classifier responds to only one heuristic feature. For the past text character components, another cascade classifier trained by HOG feature is utilized to remove the non-text character components for further accuracy improvement. Note that only text connected components approved by all the cascade classifiers can be regarded as text character components.

2.4 Text Line Accumulation with User-intention

Candidate text character components are retained and non-text noises are filtered out by Gentle Adaboost classifier using heuristic and texture features. The text line accumulation with user-intention aims to accumulate candidate character components and adjacent seed character components together into a meaningful text line.

For initialization, candidate character components overlapping with user-intention are set as the seed character components, and then character components sharing the same similarity with seed character components are assembled together based on the local and global constraints.

2.4.1 Text character similarity

The text characters in a text line should share similarities of stroke width and gray value information. To exploit this observation for the separation of candidate character components in the same line, we define the gray value similarity $f(cc)$ of current character components cc with seed connected components sc. The similarity can classify the candidate components to text line character components s or non-text line character components $\overline{s}$.

$$f(cc) = \begin{cases} s & |\overline{GV_{cc}} - \overline{GV_{sc}}| \leq |\overline{GV_{cc}} - \overline{GV_b}|, \\ \overline{s} & \text{otherwise}, \end{cases} \tag{2.10}$$

where $\overline{GV_{cc}}$, $\overline{GV_{sc}}$, and $\overline{GV_b}$ denote the mean gray value of current character component, seed character components, and background, respectively. The stroke width similarity

between current character component $\overline{sw_c}$ and seed character components $\overline{sw_s}$ is defined as $|\overline{sw_c} - \overline{sw_s}|/\overline{sw_s}$.

2.4.2 Text line accumulation

The characters in meaningful text line share similarities, namely, character distance and overlapping ratio in the line direction, which can be defined as local constraint. Each character is one member of a text line, and we can use the character similarity in one text line to define the global constraint. All candidate character components that satisfy the local constraint and global constraint are accumulated to a text line.

2.4.2.1 Local constraint

Local constraint rules are designed to update the candidate character components to text line character components. The updating operation is repeated until no candidate text character component can be accumulated in text lines. The parameters in the local constraint are decided experimentally.

- The distance d_s^c between the candidate character component and the seed character components should be smaller than twice maximum distance d_s^a of neighboring seed character components. Thus, we impose the following constraint:

$$d_s^c < 2 \cdot d_s^a. \tag{2.11}$$

- The vertical overlapping area O_s^c between candidate character component and seed character component should be smaller than half of the candidate character component area A_c. Thus, we impose the following constraint:

$$O_s^c < 0.5 \cdot A_c. \tag{2.12}$$

- The maximum width and height of the candidate character component should be larger than half of the maximum of the width and height of the seed character components. Thus, we impose the following constraint:

$$\max(w_c, h_c) > 0.5 \cdot \max(w_s, h_s). \tag{2.13}$$

2.4.2.2 Global constraint

The global constraint attempts to restrict every character to the text line. As shown in Figure 2.5, two parallel lines $\overline{l_1, l_2}$, which are close to the text foreground with the slope set as skew angle of the text line, are defined to present the global constraint. All the text character components in the text line "RIVERSIDE" should be restricted in the estimated border text lines l_1 and l_2.

Given a set of seed character components S and parallel lines $\overline{l_1, l_2}$, we define its text line region R_{l_1,l_2} that encloses S. The probability of the candidate character component P_{cc} belonging to the text line is defined by the ratio of character component foregrounds falling into the text line region. The candidate character component belongs to the text line if the probability is larger than the setting threshold

$$P_{cc} = \frac{F_{cc} \cap R_{l_1, l_2}}{F_{cc}}. \tag{2.14}$$

where F_{cc} denotes the foreground pixels of candidate character component. After new candidate character components are grouped into the text line, the new parallel lines

FIGURE 2.5: Global constraint for user-intention text line between two green parallel lines l_1 and l_2.

$\overline{l_1, l_2}$ should be updated until no candidate can be grouped. Finally, we obtain the whole user-intention text line.

2.5 Experiments and Analysis

To evaluate the effectiveness and robustness of the proposed text line extraction with user-intention algorithm in terms of swipe and tap gesture, we apply the proposed method to the public ICDAR2003 database [65], which contains 258 images in the training set and 251 images in the test set. The Gentle Adaboost classifiers are trained on the training database with heuristic and HOG features, and the performance of the proposed method is evaluated on the test database. The user-intentions of swipe and tap gestures are made by volunteers to indicate the target text line randomly on the ICDAR2003 database.

2.5.1 Experimental results and analysis

The traditional precision criterion [66] is used to evaluate the performance of text line extraction with user-intention. It should be noted that the user-intention target texts

are text lines rather than a single character. The precision criterion compares the match similarity between the detected text line bounding rectangle and the annotated ground truth text line bounding box. The text line region matching area, $m(i, j)$, between the i^{th} ground-truth text line region and the j^{th} extracted text line region is given as follows:

$$m_r(G_i, E_j) = f(G_i \cap E_j \cap I), \tag{2.15}$$

where I is the image region, and G_i and E_j denote the i^{th} ground truth region and the j^{th} extracted text line region, respectively. $f(\cdot)$ is a function computes the intersection area of the text line region. Then, the best matching of the extracted text line region and ground truth text line region can be computed by the following equation:

$$Match_E(E_j) = \max_{i=1,\ldots,|G|} \frac{2 \times m_r(G_i, E_j)}{|G_i| + |E_j|}. \tag{2.16}$$

The precision criterion is calculated on each image independently, and then an average value over all images is calculated as the performance of the proposed method. We count the number of matches according to the overlapping of the labeled text lines and the ground truth text lines

$$precision = \frac{\sum_j^{|E|} Match_E(E_j)}{|E|}. \tag{2.17}$$

Table 2.1 illustrates the performance of scene text detection with user-intention in terms of tap gesture. Our method achieves the precision of 0.81 on ICDAR2003 test database, which is 9.0% higher than Du's method. Compared with state-of-the-art method, the proposed method achieved encouraging performance.

TABLE 2.1: Precision of the proposed text line extraction with user-intention in terms of tap gesture on the ICDAR2003 database.

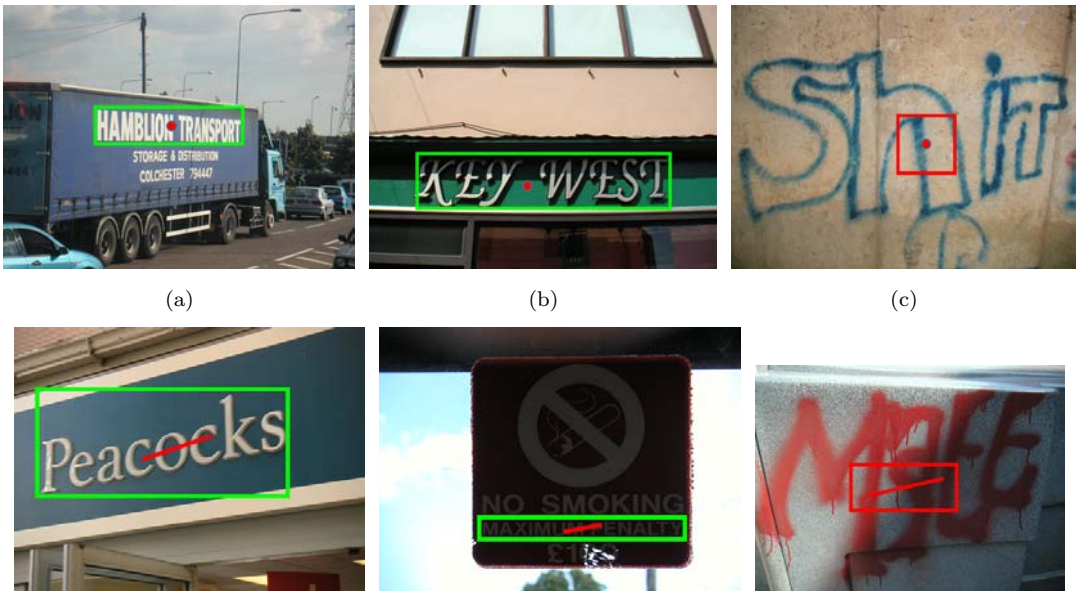
Method	precision
Du [62]	0.72
The proposed method	0.81

TABLE 2.2: Precision of text line extraction with user-intention in terms of swipe gesture on the ICDAR2003 database

Method	precision
Du [62]	0.77
Sun [67]	0.79
The proposed method	0.84

Table 2.2 illustrates the precision performance of scene text detection with user-intention in terms of swipe gesture. Our method achieves a precision of 0.84 on the ICDAR2003 test database, which is improved by 5.0% in comparison with Sun's method. Meanwhile, the performance of text line extraction with user-intention by swipe gesture is 3.0% higher than that of tap gesture, this performance is reasonable because more information is captured than tap gesture.

Figure 2.6 demonstrates successful and failure examples of text line extraction with user-intention on different images with different orientation and skew. The text line extraction focuses on a user-interested text line rather than the whole text lines in the image. The red point and skew line denote the tap and swipe gesture, respectively. The green box and red box present the successful and failure examples of text line extraction with user-intention, respectively. The images in the first row give the text line extraction with tap gesture, and the images in the second row present the text line extraction with swipe gesture. We observe that the proposed method is accurate and robust for complex natural scene images with various fonts and sizes, and low contrast. The proposed method can also be applied in the handhold devices for human-computer



(a) (b) (c)

(d) (e) (f)

FIGURE 2.6: Successful and failure examples of text line extraction with user-intention on the ICDAR2003 competition database.

interaction.

Although text line extraction with user-intention achieves competitive results, some failures still exist. Some user-interested text lines are not successfully extracted, as shown in Figure 2.6(c) and (f), due to the failure of text character extraction. This limitation can be corrected by user-labeling the complete text line.

2.6 Conclusion

The purpose of this chapter is to extract text line with user-intention in terms of tap and swipe gesture. The proposed method makes full use of the user-intention and designs a sub-region detector by using the estimated text property. The decomposed character components are discriminated into text character components and noises through the cascade of Gentle Adaboost classifiers with extracted heuristic and texture features. Finally, text candidate character components, which share the same similarity of gray

value and stroke width constraints, are accumulated to a text line according to the local and global relationship with seed character components. The experimental results on the public ICDAR2003 competition database demonstrate the effectiveness of our text line extraction with user-intention in natural scene images.

Chapter 3

Character Candidates Extraction and Directed Graph Construction for Multiple Text Line Extraction

3.1 Introduction

As mentioned in Chapter 1, most of the conventional methods [50, 57, 59, 60] usually group the text components into text lines exhaustively without any structure due to the difficulties of the construction of text candidate structure. Some of methods [51, 56, 58] build a undirected graph to simply integrate the text components. These difficulties include text line alignment, text variation, and scene complexity, making the text candidates relationship construction a challenge task.

- Text line alignment: the text lines can show horizontal, skew, or non-planar alignment, and various perspective distortions. As a result, no priori knowledge can be

FIGURE 3.1: Construction of the directed graph based on the spatial arrangement of connected components.

used to construct the text component relationship.

- Text variation: the size and font of text characters can greatly vary even in the same text line, making it difficult to build relationship. To make matters worse, the broken or touching text components will increase the difficulties of the relationship construction of text components.

- Scene complexity: in natural scene environments, the text characters are surrounded by numerous text-like false positives, such as bikes, tires, leaves, and fences, which are very difficult cases for building relationships among text components.

This chapter focuses on the disorder elimination of text components and attempts to discover possible solutions by observing and analyzing the pattern of local text line arrangement. Inspired by how humans read text lines from left to right, we construct a directed graph from the extracted text candidates as shown in Figure 3.1. The red bounding boxes and green arrows denote the vertices and edges in the directed graph. The vertices comprise all the connected components that may be single characters, touching characters, broken characters, and noises due to uncorrect character extraction process. The directed edges connect the vertices based on the arrangement of the vertices from left to right. The correct text lines are assumed to be included as paths

in the directed graph. We can obtain a globally optimal text line extraction results by extracting those paths that give the minimum overall cost.

This method is supposed to offer the following advantages over the conventional methods:

- The directed graph designed from the candidate text component nodes can be used to construct the relationship structure and eliminate the disorder of candidate text components in the graph model.

- Compared with the undirected graph, the directed graph designed from the candidate text component nodes can greatly reduce the computational complexity for text line path global optimization.

- The maximal stable extremal region-based text components extraction method can extract most text components even from low-quality images to achieve the approved recall. Meanwhile, the designed Convolutional Neural Networks (CNN) can effectively remove the noisy false positives to achieve a promising precision.

3.2 Text Candidate Extraction

3.2.1 Extraction of maximal stable extremal region

In OCR technologies, connected components of foreground pixels are extracted to be text character candidates. This is so-called text character extraction. Text character extraction is conducted by discriminating the text components from image backgrounds. Such extraction should be robust to the effects of color, illumination, degradation, and texture variation in the foreground and background. The extracted text components can be utilized for text localization, verification, or recognition in each OCR step.

FIGURE 3.2: Example of text character components extraction from background.

Each image has two text character polarities, namely, black text characters on white background and white text characters on black background. A color image is initially converted into the gray channel. Then, the two polarities of text characters are extracted by using the MSER method. The black text characters on a white background indicate that the pixel intensity of text characters in the image is lower than the intensity of background boundary pixels. Similarly, the white text characters on a black background indicate that the pixel intensity of text characters in the image is larger than the intensity of background boundary pixels. Both of black text characters and white text characters must be extracted for text line extraction in OCR application.

Given an input image I, the text character extraction binarizes a set of contiguous regions $R = \{r_1, r_2, ..., r_n\}$ on an input. The bounding box of each region can be used as a text character node in the construction of directed graph and as text nodes in a text line. Figure 3.2 illustrates the extraction of text characters from the target image. As can be seen in the figure, a set of text characters {"P", "e", "a", "c", "o", "c", "k", "s"} are discriminated from the texture background.

In the text components extraction, we employ the MSER method to extract the text character candidates. The MSER based character region extraction method [19] has been considered as one of the best candidate connected components extraction method

[68–70], which has won the first place in the ICDAR2011 [71] and ICDAR2013 [72] competitions and achieved promising performance. The MSER can be regarded as a virtually unchanged local binarization over a large range of thresholds. The MSER method shows high efficient in both multi-scale detection and low quality connected components with near-linear complexity [73].

An Extremal Region (ER), whose special case results in MSER, is well defined as a connected component region R_i if the intensity values of all sets are either larger or smaller than its outer boundary intensity. The local minima intensity will appear and two local minima regions will merge into a new extremal region with the threshold level increasing. Let Δ be the parameter of threshold step, the variation of an ER R_i is defined as follows:

$$v(R_i) = \frac{|R_{i+\Delta} - R_i|}{|R_i|}. \tag{3.1}$$

An ER R_i is an MSER when it is maximally stable, that is, it has a minimum of variation and if its variation is lower than that of its parent R_{i-1} and child R_{i+1}.

Figure 3.3 illustrates an example of the MSER root tree with an increasing threshold level. Firstly, the text characters in "Home" are adhesive in one connected component in the assigned threshold. Secondly, as the threshold increases, the text character "H" is separated from the adhesive connected component, and then the touching connected component consists of the three character "ome". Finally, the characters "o," "m," and "e" are discriminated from the adhesive connected component "ome" in the MSER rooted tree.

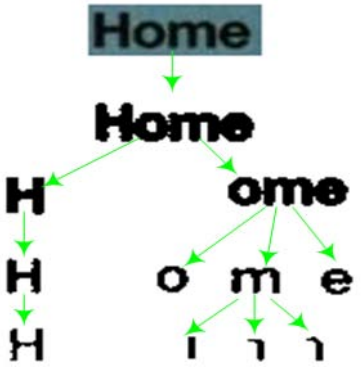


FIGURE 3.3: Text candidates extraction in the MSER rooted tree.

The performance of an MSER is controlled by five parameters, which are defined as below:

(a) maxWH: it prunes those component regions whose maximum of width and height bigger than "maxWH",

(b) minWH: it prunes those component regions whose minimum of width and height smaller than "minWH",

(c) maxVariation: it prunes those regions with sizes similar to those of its children,

(d) minDiversity: it cuts off those MSERs with diversity values below the set value, and

(e) Δ: it represents the parameter for the threshold step.

In the MSER rooted tree, any connected component that satisfies the MSER definition will be extracted as a text candidate. However, text characters have exclusive characteristics that discriminate text characters from other MSERs. Therefore, we apply the aspect ratio characteristic to penalize too large or too small aspect ratio [50]. Suppose that v_r is the variation of region r and a_r is the aspect ratio of region r, we can define the regularized variation v_r^* with the text character aspect ratio falling into $[a_{min}, a_{max}]$ as follows:

$$v_r^* = \begin{cases} v_r - \lambda \cdot (a_r - a_{max}) & a_r > a_{max} \\ v_r - \lambda \cdot (a_{min} - a_r) & a_r < a_{min} \\ v_r & \text{otherwise.} \end{cases} \tag{3.2}$$

3.2.2 Character and non-character classification

Although the MSER method can achieve a promising recall and extract most of the candidate text components even in low quality images, this technique suffers from many false positives, which result a low precision. Any non-text objects, such as bikes, leaves, and tires, will generate noisy MSER connected component regions. In this section, character and non-character classification is described to identify and eliminate the non-character noises for the text line extraction. The false positive MSER components are removed by using CNN classifiers [74].

CNN is regarded as one of the most promising techniques that achieve state-of-the-art performance in solving many data-driven problems, such as character recognition [75–77], face recognition [78–80], speech recognition [81–83], and natural language understanding [84–86]. The CNN is a hierarchical architecture with multiple layers of feature convolutions. Unlike conventional hand-engineered features [87, 88], the CNN is capable of meaningful high-level features and semantic representation for text character components automatically.

We design one CNN to discriminate the text characters and non-characters as shown in Figure 3.4. The designed CNN consists of three convolutional layers, each of which is followed by a Rectified Linear Units (ReLU) layer and a max-pooling layer. Then, a fully-connected layer is followed by a softmax classifier. The convolutional layer can be regarded as a feature extractor that converts the text candidate regions into "feature

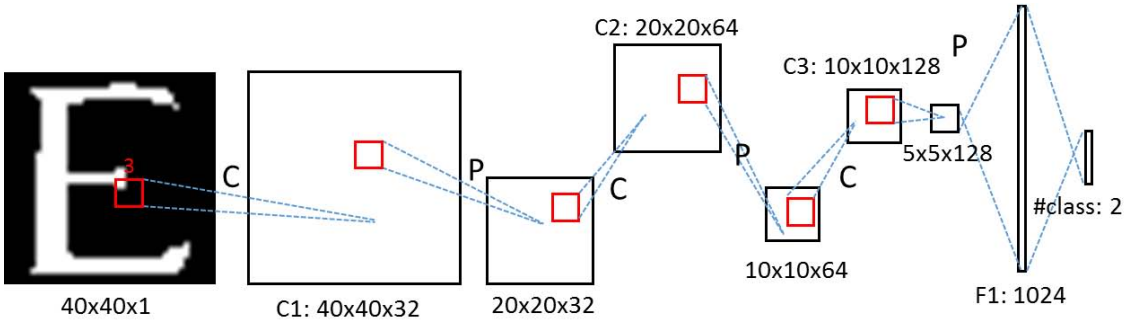


FIGURE 3.4: The designed CNN structure for character and non-character classification.

maps" that represent various text features, including strokes, edges, spatial distributions or textures. Three convolutional layers produce 32, 64, and 128 feature maps, respectively, which are convolved by a shifting window with 3×3 kernels in one stride. The output feature maps will be reshaped as input feature maps for the next convolutional layer.

The ReLU layer is the non-saturating activation function $f(x) = \max(0, x)$ to enhance the non-linear properties of the feature maps and to accelerate the CNN training on large and complex datasets.

In the max-pooling layers, the non-linear down-sampling strategy is employed to reduce the spatial size of the feature maps and to control the over-fitting. For each 3×3 sub-region in the feature map, we apply the overlapped pooling strategy with a stride of 2. The maximum value of the sub-region is then outputted for successive layers in the CNN architecture.

Each neuron in the fully-connected layer has connections to all activations with the previous layer. Two fully-connected layers with 1024 and 2 neurons, respectively, are then concatenated in sequence after the convolutional layers. The fully-connected layer that is followed by a softmax classifier computes the score probability of classifying one candidate region into either text character or non-text noise.

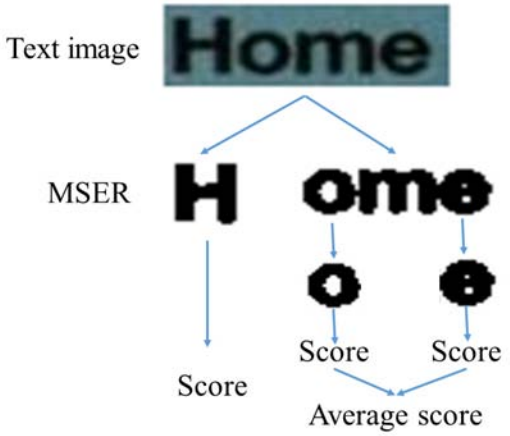


FIGURE 3.5: Character and non-character classification strategy for touching characters and non-touching characters.

Given an extracted binary MSER with an aspect ratio smaller than 2.0, it is normalized into 40x40 pixels while maintaining the aspect ratio. The minimum width and height are padded by the bounding pixels. The resized regions are fed into the designed well-trained CNN to predict the character or non-character score probability. Those character regions whose score probabilities exceed the set threshold $T = 0.9$ are retained as true positives, while those character regions with score probability smaller than threshold are removed as noises.

For the touching characters (with an aspect ratio larger than 2.0 in the experiment), the minimum width and height is normalized to 40 pixels. The left and right patch segments are recognized afterward for average voting. Figure 3.5 presents an example of the touching character classification strategy. The characters in "ome" are touching together in one MSER, the left segment "o" and right segment "e" are segmented from the touching characters according to the height of MSER. These two segments are fed into the designed CNN to calculate the score probability and the probability of touching character is calculated by the average probability score of these two segments.

3.3 Directed Graph Construction

The directed graph construction aims to build the structure of a graph like Figure 3.1. The vertices of the graph comprise the candidate text characters which are extracted by the procedure of Section 3.2. The directed edges of the graph connect the candidate text characters which satisfy some constraints. The text lines are assumed to be path segments in the directed graph.

3.3.1 Constraints of the potential directed graph edges

The text characters in an image are spatially arranged in a text line, and the neighboring text characters share similar spatial properties. Obviously, we can use these constraints to filter out the unnecessary edges in the directed graph. Two constraints, namely, distance constraint and overlapping constraint, are defined for the potential directed graph edges, and any two text candidate components in the directed edge must satisfy the criteria.

3.3.1.1 Distance constraint

Given that two neighboring text characters, m_i and m_j, are located close to each other in a text line, we restrain the Euclidean distance $D(m_i, m_j)$ between two neighboring text characters lower than the threshold t_d as follows to avoid unnecessary edges in the directed graph:

$$t_d = \alpha \cdot \min(\max(w_{\mathrm{i}}, h_{\mathrm{i}}), \max(w_{\mathrm{j}}, h_{\mathrm{j}})), \tag{3.3}$$

FIGURE 3.6: Distance constraint for the directed graph construction.

where w_i, h_i and w_j, h_j denote the width and height of the bounding boxes of the m_i and m_j, respectively, and α is a control parameter for the distance constraint.

Figure 3.6 illustrates the effectiveness of the distance constraint. Each character is set as one vertex of the directed graph. The green and red arrows denote the potential directed edges and impossible directed edges, respectively. For the source vertex of text character "C" with blue rectangle, we can build the potential directed edges from the source character "C" to characters "O", "L", and "C" due to short distance. Obviously, building directed edges from the source vertex "C" to characters "H", "E", "S", "T", "E", and "R" is unnecessary because the source character "C" is located far from the target characters.

3.3.1.2 Overlapping constraint

We observe that the text characters are spatially arranged as a straight, skewed or curved text line. The overlapping between two neighboring text characters exists between these two neighboring characters. Therefore, we apply the overlapping constraint in the assigned direction from left to right as shown in Figure 3.7. The length of the overlapping

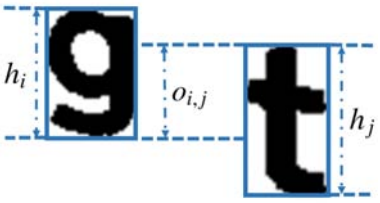


FIGURE 3.7: Overlapping constraint between two neighboring text connected components.

range $o_{i,j}$ must be larger than 0, which is considered reasonable for the arrangement of the text lines. Any two MSER components that satisfy these two constraints are potentially linked in the text line directed graph.

3.3.2 Directed graph construction

Given that any two or more text characters can be treated as a text line in an image, and the disorder in the candidate characters can make text line extraction a challenging task. Meanwhile, the heuristic rules can decrease the robustness of text line extraction when the text line situation changes. Therefore, we construct one directed graph upon the candidate text characters by applying the weak universal knowledge of the text line in order to effectively build the relationship structure and eliminate the disorder of text characters. The text lines are assumed to be text line paths in the directed graph, and the text line extraction can be formulated as a solution to the optimization of paths in the directed graph.

Suppose each MSER component is a vertex v_i, and a directed edge $e_{i,j}$ is located between v_i and v_j in the directed graph. We can construct a directed graph $G_D = (V, E)$ composed of text character vertices V and directed edges E that connect vertices of text characters. The vertices and edges set of the directed graph can be expressed as $V = \{v_i\}_{i=1}^{n}$ and $E = \{e_{i,j}\}$, respectively, where n denotes the number of MSER components vertices.

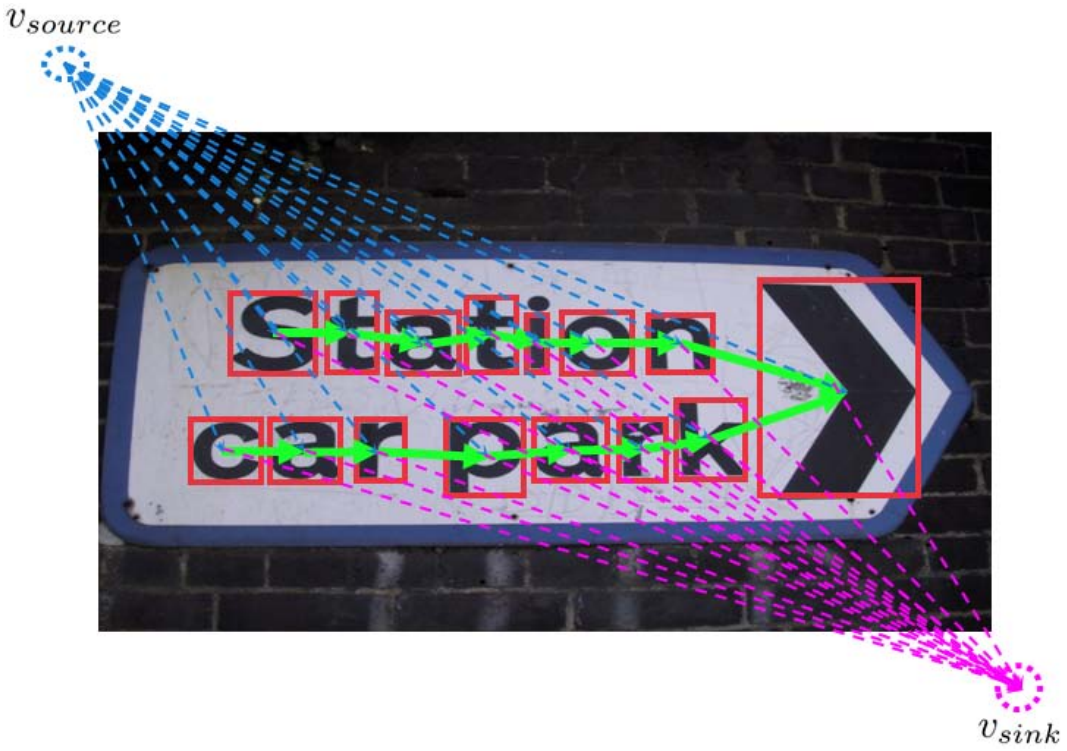


FIGURE 3.8: Construction of directed graph upon the MSER text character components.

The vertices of the directed graph represent the locations of MSER components, and each vertex can be the starting or the ending location of a text line. Therefore, two additional virtual vertices v_{source} and v_{sink} are added to the directed graph, where v_{source} is the possible graph source point and v_{sink} is the text line sink point. A directed edge $e_{source,i}$ is located from the virtual source vertex to each MSER component and a directed edge $e_{i,sink}$ is located from each MSER component vertex to the sink vertex. The cost value of these edges are set to 0 to allow each MSER component to be the start or end location of the text line at no cost.

The directed graph edges are built in the assigned text line direction from left to right according to the spatial position relationship of the center MSER vertices. This arrangement mirrors the behavior of humans in reading text character by character. For vertex v_i, we firstly check the text line direction from left to right. The x-coordinate cx_j of vertex v_j must be larger ($cx_j > cx_i$) than the x-coordinate cx_i of vertex v_i. We also

verify the distance and overlapping constraint to build the candidate vertex pair $p_{i,j}$ from vertices v_i to v_j. Finally, we identify the nearest MSER component vertex v_j from the set of candidate vertex pair $\{p_{i,j}\}$ and build one directed edge $e_{i,j}$ from the current vertex v_i to the selected MSER vertex v_j. The details of the directed graph construction are given in Algorithm 2.

Algorithm 2 Directed graph construction.

Input:

The set of extracted text character component vertices, $V = \{v_i\}_{i=1}^{n}$.

Output:

Constructed directed graph, $G_D = (V, E)$.

for $i = 1$ to n **do**
  Build an directed edge $e_{source,i}$ from virtual source vertex v_{source} to vertex v_i;
  Build an directed edge $e_{i,sink}$ from vertex v_i to virtual sink vertex v_{sink};
  for $j = 1$ to n, $j \neq i$ **do**
    if $cx_j > cx_i$ **then**
      Verify the text character pair by the distance and overlapping constraints, described as 4.1;
      Build the component pair $p_{i,j}$ between vertex v_i to v_j, and add it to the candidate set $\{p_{i,j}\}$;
    end if
  end for
  Find the nearest vertex v_j with minimum distance in the set of $\{p_{i,j}\}$;
  Build an directed edge $e_{i,j}$ from vertex v_i to vertex v_j;
end for
return $G_D = (V, E)$.

Figure 3.8 presents an example of the built directed graph. In the figure, the blue and purple dashed arrows denote the directed edges from the virtual source vertex to the MSER vertex and the MSER vertex to the virtual sink vertex, respectively, while the green arrows represent the directed edges from the start to end vertices, The text lines "Station" and "car park" are included in the directed graph in the form of directed

paths p_1=$v_{\text{“}S\text{”}}$, $v_{\text{“}t\text{”}}$, $v_{\text{“}a\text{”}}$, $v_{\text{“}t\text{”}}$, $v_{\text{“}i\text{”}}$, $v_{\text{“}o\text{”}}$, $v_{\text{“}n\text{”}}$ and p_2=$v_{\text{“}c\text{”}}$, $v_{\text{“}a\text{”}}$, $v_{\text{“}r\text{”}}$, $v_{\text{“}p\text{”}}$, $v_{\text{“}a\text{”}}$, $v_{\text{“}r\text{”}}$, $v_{\text{“}k\text{”}}$, respectively.

3.3.3 Cost function of graph edges

Each directed edge is assigned an appropriate cost value in the directed graph to measure the importance of the relationship among the text components, and the directed edge cost function $c(e_{i,j})$ calculates the cost value from one MSER vertex to another in the directed graph edge. Specifically, the following unary and pairwise cost functions are employed to calculate the cost of each directed edge.

1. Unary cost function: the unary cost function measures the cost value of classifying the candidate MSER vertex into text. The probability of recognition engine and variation of MSER are used to define the unary cost function.

   (a) Probability of recognition engine: the probability given by the CNN-based recognition engine in Section 3.2.2 is a strong feature that discriminates the text character from noises.

   (b) Variation of MSER: the text MSER component is virtually unchanged over a large range of thresholds to distinguish it from the background. Therefore, the text MSERs tend to show small variations of MSER components.

2. Pairwise cost function: the pairwise cost function measures the cost value of the discontinuity of two linking text candidates in one text line. The location distance, overlap, and gray value similarity of the adjacent vertices are applied to define the pairwise cost function.

(a) Location distance of adjacent vertices: the text line comprises some individual characters that are arranged in a specific order. The adjacent MSER components in the text line are separated by a small distance, while the noises do not follow any regular pattern.

(b) Overlap of adjacent vertices: all components in the text line are arranged straightly, and an overlap is observed between the adjacent text components as shown in Figure 3.7. The overlap between adjacent vertices is calculated as:

$$ro_{i,j} = \frac{o_{i,j}}{h_i + h_j}. \tag{3.4}$$

(c) Gray value similarity of adjacent vertices: adjacent text vertices in the same text line have a similar gray vale, which is calculated by the mean gray value of foreground:

$$g_{i,j} = |g_i - g_j|. \tag{3.5}$$

The directed edge $e_{i,j}$ from vertices v_i to v_j is assigned the cost value $c_{e_{i,j}}$, which is calculated as:

$$c_{e_{i,j}} = -\log\left(\frac{c_t}{1 - c_t}\right), \tag{3.6}$$

where c_t is calculated by the above unary cost and pairwise cost with equal weight. Note that there is no cost for directed edges between each MSER vertex and virtual source vertex, and the same as sink vertex.

TABLE 3.1: Parameter setting of the text character extraction by using the MSER method.

Parameter	Δ	maxWH	minWH	maxVariation	minDiversity
Setting	4	10	600	0.5	0.5

3.4 Experiments and Analysis

The experiment results of the text character extraction and directed graph construction are demonstrated in this section to show the effectiveness of the proposed method. The analysis of the experimental results highlights the importance of constructing the directed graph construction on the text characters.

3.4.1 Experiment setting

We use five parameters to control the text character extraction performance. These experiment parameters are listed in Table 3.1. Given that the MSER regions are being produced in the root tree, we increase the threshold step by 4 pixels and prune those regions with a "maxVariation" larger than 0.5 and a "minDiversity" smaller than 0.5. The maximum width and height in the experiment are both set to 600, thereby limiting the extraction to those text characters whose maximum width and height are smaller than 600 pixels. Meanwhile, the minimum width and height are set to 10, thereby further limiting the extraction to those text characters which width and height do not go below 10 pixels. These settings are deemed reasonable because too large or too small text character candidates are tend to be noisy and need to be filtered out.

We observe that the aspect ratio of text characters are often within a certain range. Having a too large or too small aspect ratio indicates that the MSER region tends to be noisy or touching each other. Obviously, we can penalize the variations in the MSER with

TABLE 3.2: Parameter setting of variation penalization in the MSER root tree.

Parameter	λ	a_{max}	a_{min}
Setting	0.33	3	0.33

a too large or too small aspect ratio. The parameters of the MSER variation penalization are listed in Table 3.2. Suppose that the regular aspect ratio of text characters are located in $[0.33, 3]$, we can penalize the too large or too small MSER regions with the coefficient of 0.33.

The neighboring text characters in the text line are adjacent to one another and the Euclidean distance of potential linking text characters is small enough for the text line arrangement. Therefore, the noisy directed edges can be effectively filtered out from the constructed graph by the distance constraint. We set the parameter k to 3.0 to remove the potential directed edges as defined in Equation 3.3.

To train the CNN parameters, we harvest the training samples by applying the MSER text character detector on the ICDAR2011 [71] and SVT [89] training databases. For the positive samples, we manually label the text character regions extracted by the MSER method in each image. Similarly, the MSER regions that do not overlap with the ground truth text characters are harvested as negative samples. Each sample is padded by 2 pixels to avoid losing the boundary features. Figure 3.9 presents some negative samples from the non-character region in the training database, while Figure 3.10 presents some positive text character samples that can either be single text characters or touching characters.

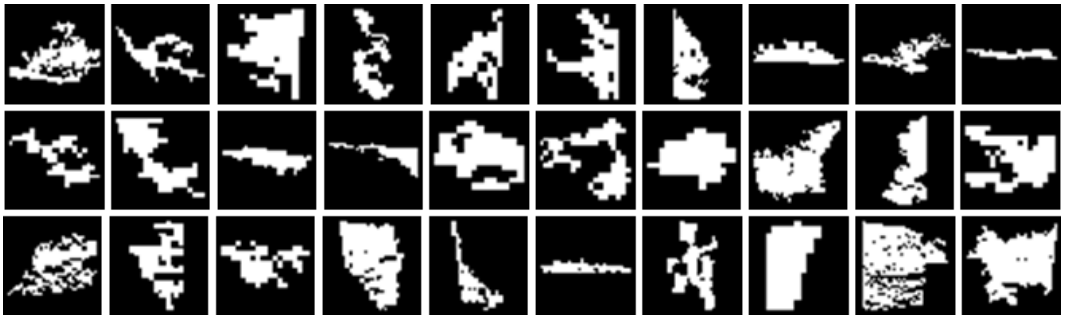


FIGURE 3.9: Negative CNN training samples for character and non-character classification.

3.4.2 Results of character candidates extraction

Figure 3.11 presents some results of text character extraction by using the MSER method. The images on the first, second, and third rows are the source images, the results of black text character extraction on a white background, and the results of white text characters extraction results on a black background, respectively. The images on the first column show that the text characters have various sizes and styles. For instance, the text characters "JAVA" are much larger than "3RD" and "EDITION." As for the images on the second column, the intensity of the text characters "go create" is much higher than that of the adjacent characters because of the highlights. The images on the third column show that the MSER method has successfully extracted the two polarities text characters. The images on the fourth column reveal that the contrast between the text characters and background is fairly small to discriminate these two from each other. Consequently, the MSER method can effectively extract text characters with various sizes, highlights, different polarities and low contrast text characters in images.

Figure 3.12 illustrates some examples of CNN-based false positive removal. The green, blue, and red bounding boxes represent the retained black text characters, white text

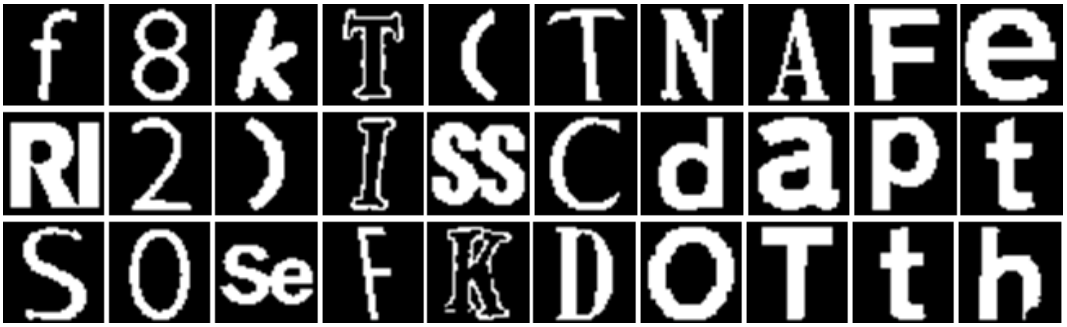


FIGURE 3.10: Positive CNN training samples for character and non-character classification.

FIGURE 3.11: The results of extracting text characters from different images by using the MSER method. The images on the first, second, and third rows are the gray image, the extracted black text characters, and the extracted white text characters, respectively.

characters, and removed noise non-characters, respectively. Most of the noise non-characters are removed except for some noise regions with features that are similar to those of text characters. As shown in the figure, the text character can be retained due to the high classification ability of CNN. The outputs of CNN-based non-character removal are fed into the directed graph construction procedure to build the relationship among the text characters.

As shown in Figure 3.12, the MSER-based text character extraction method can achieve

FIGURE 3.12: Examples of text character and non-character classification by a well-trained CNN.

a promising recall and extract most text characters even in low quality images. However, this method can produce many noises and reduce the precision of text character extraction. In the text line extraction procedure, it will leave a challenge task of solving the paths optimization in the directed graph. Therefore, we employ a CNN to effectively discriminate the text characters from the non-characters.

3.4.3 Result of directed graph construction

Figure 3.13 illustrates the construction of the directed graph based on all extracted regions. The virtual source and sink vertex are ignored to simplify the explanation. The red arrows denote the directed path from one character to an adjacent text character. The false positive regions will also be added into the directed graph and assumed to be one vertex that will be solved in the text line paths optimization procedure. The candidate text characters that are located far away from the other regions are isolated and have no connections.

As depicted in the figure, the directed graph can construct the relationships. After the CNN-based non-character removal, the directed graph constructs the relationship among

FIGURE 3.13: Construction of the directed graph based on all the candidate character regions.

all candidate regions according to the spatial arrangement of text lines in an image. The directed edge connects two vertices of text characters from left to right. Obviously, the vertices and directed edges will produce the directed graph among characters and include the text line paths for optimization. There are still spurious vertices and edges. These vertices and edges will be added into the directed graph, and the k-shortest paths optimization can automatically remove these noisy vertices in the optimized text line paths.

3.5 Conclusion

This chapter aims to construct the relationships among the candidate text characters and to eliminate the disorder of candidate text characters. The key idea is to construct a directed graph based on the text character nodes that are spatially arranged in the text lines. Through the observation of human reading sense from left to right, we consider the spatial arrangement of adjacent text characters in constructing the directed graph.

In the directed graph, the text line paths are also included as paths that flow from the virtual source vertex to the sink vertex. Our experiments demonstrate that the directed graph can be constructed successfully based on well-extracted text characters and the text lines can be extracted by solving the text line paths global optimization problem in the graph.

Chapter 4

K-Shortest Paths Optimization for the Extraction of Multiple Text Lines

4.1 Introduction

Text line extraction is a crucial step for OCR applications where the text character candidates are concatenated into text lines based on the extracted text character components. As mentioned in Chapter 1, most of the conventional methods [50, 57, 60] group the text character components into text lines. However, given that such "greedy" grouping often employs the heuristics rules of text characters, the robustness of the text line extraction will be decreased when the scenes change. Moreover, the lack of text line information, it is hard to determine the number of text lines in an image, and the exhaustive search leads to low precision until all text characters are identified in the text lines.

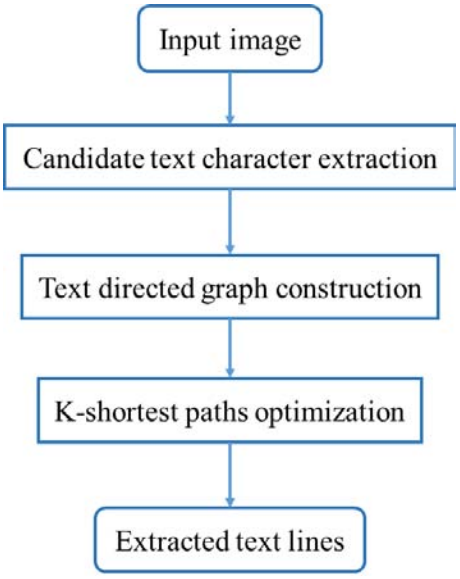


FIGURE 4.1: Text line extraction system based on text line paths global optimization.

This chapter focuses on the global optimization of text line paths by taking advantage of the constructed directed graph described in Chapter 3. Figure 4.1 presents the flowchart of text line extraction based on text line paths optimization. We formulate the text line paths optimization problem as k-shortest paths optimization in the directed graph. The text line paths are similar to human reading ability in planning of a text line path sequentially [90–92].

The text line paths global optimization tries to discover the global solution of multiple text lines in an image. As shown in Figure 4.2, two paths with thicker arrows p_1=v_{source}, $v_{\text{“}S\text{”}}$, $v_{\text{“}t\text{”}}$, $v_{\text{“}a\text{”}}$, $v_{\text{“}t\text{”}}$, $v_{\text{“}i\text{”}}$, $v_{\text{“}o\text{”}}$, $v_{\text{“}n\text{”}}$, v_{sink} and p_2=v_{source}, $v_{\text{“}c\text{”}}$, $v_{\text{“}a\text{”}}$, $v_{\text{“}r\text{”}}$, $v_{\text{“}p\text{”}}$, $v_{\text{“}a\text{”}}$, $v_{\text{“}r\text{”}}$, $v_{\text{“}k\text{”}}$, v_{sink} are extracted based on global optimization. The global optimization can exclude the vertex of noises and the directed edge from the vertex of character “n” to the vertex of noise automatically from the text lines. This method initially produces one text line with a minimum cost in the directed graph. Afterward, the text line paths are iteratively augmented to multiple text lines, similar to the behavior of humans reading ability in planning text line paths sequentially.

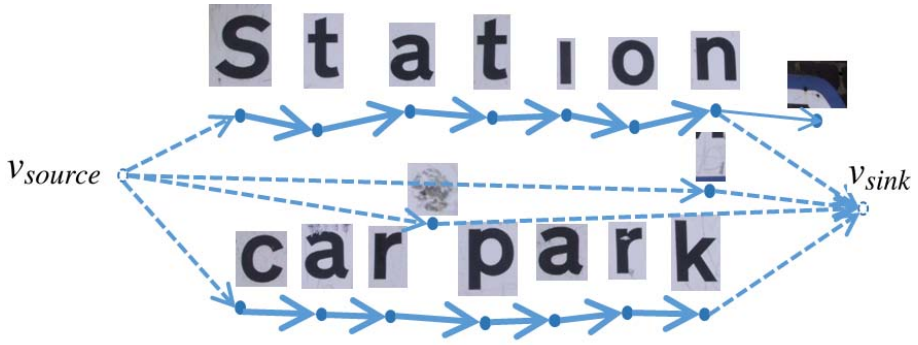


FIGURE 4.2: Text line extraction based on k-shortest paths optimization.

This chapter discusses the global optimization of text line paths upon the constructed directed graph. This procedure offers the following advantages over the conventional methods:

- The k-shortest paths global optimization can produce the text lines iteratively to avoid an exhaustive search. Therefore, the text line paths optimization-based method is suitable for various appearances of text lines in a given image.

- The number of text lines can be obtained automatically through the optimization process. The noisy candidate components can be excluded in the global optimization, thereby reducing the number of extracted noise text lines.

- As revealed in the later experimental results, we demonstrate that the text line extraction problem can be solved by using k-shortest paths global optimization in the constructed directed graph.

4.2 Methodology of the K-Shortest Paths Optimization

A text line can be considered as a path flow from the source text character node to the sink text character node in the constrict direction. The directed text line path is

determined by a sequence of text character vertices $V = v_1, v_2, ..., v_n$. The directed edges in the text line path connect these vertices into the sequence $E = e_{v_1,v_2}, e_{v_2,v_3}, ..., e_{v_{n-1},v_n}$.

Figure 4.2 presents an example of the text line paths. Two paths p_1=v_{source}, $v_{\text{“S”}}$, $v_{\text{“t”}}$, $v_{\text{“a”}}$, $v_{\text{“t”}}$, $v_{\text{“i”}}$, $v_{\text{“o”}}$, $v_{\text{“n”}}$, v_{sink} and p_2=v_{source}, $v_{\text{“c”}}$, $v_{\text{“a”}}$, $v_{\text{“r”}}$, $v_{\text{“p”}}$, $v_{\text{“a”}}$, $v_{\text{“r”}}$, $v_{\text{“k”}}$, v_{sink} arrange from the virtual source vertex to the sink vertex. The character “p” is treated as a single vertex in the text line path p_2 but cannot be treated as a single vertex in p_1. Given that no overlapping vertex is observed in either of these two text line paths, thus each character cannot be shared by two or more text lines, and the text line paths are node-disjoint in the given image. In sum, we can formulate the text line extraction problem as a results of the global path flow optimization on the constructed directed graph.

The text line paths can be optimized by using the generic Linear Program (LP) method [93]. However, this approach is only suitable for moderately-size problems with a high time complexity. The k-shortest paths optimization is more efficient than the LP method by taking advantage of the particular structure of the constructed directed graph. The average time complexity of this method is nearly linear with the number of nodes.

The k-shortest paths optimization aims to find a given number k of these paths, and the minimum total cost is subjected to the node-disjointness path by utilizing a minimal cost flow algorithm [94]. The node disjoint paths can be also used in communication networks to facilitate the transmission between a given source and sink nodes. The K shortest paths are found inductively from previous optimal path solutions $P = p_1, p_2, ..., p_{k-1}$ by using a shortest path algorithm at each step. This method is not just a greedy algorithm that finds the shortest path in a one-by-one way. In each iteration, the previously detected paths still have a chance to be revised for the globally optimal solution.

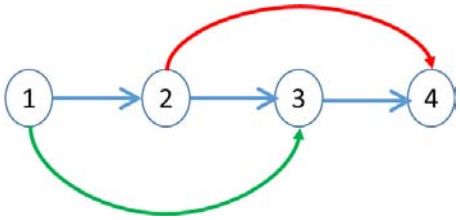


FIGURE 4.3: Node disjoint paths.

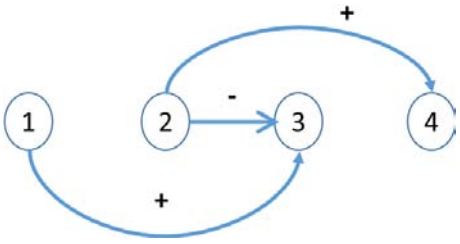


FIGURE 4.4: Path difference between P_2 and P_1 in Figure 4.3.

Figure 4.3 and Figure 4.4 present the example of the path difference between two paths. Suppose that the shortest path is $p_1 = v_1, v_2, v_3, v_4$ from vertex v_1 to vertex v_4 and that the optimal node disjoint paths are v_1, v_2, v_4 and v_1, v_3, v_4 (as shown in Figure 4.3). A signed positive label "+" is assigned to edge whose direction coincides with the direction of the shortest paths, while all the other edges are assigned with the negative label "-". The logical difference $P_2 \ominus P_1$ turns out to be a path from v_1 to v_4 in Figure 4.4 when the different signs of directed edges are cancelled. The path cost in Figure 4.4 represents the cost difference of P_2 and P_1.

In the text line path extraction problem, the text line path difference always includes the virtual source v_{source} and sink v_{sink} vertices with "+" and "-" labels, respectively. The total cost of a signed text line path is the total cost of its positively labeled edges subtracting the total cost of its negative labeled edges. The basic idea of k-shortest paths optimization is to augment the text line paths in the minimum flow problem. After extracting the shortest text line path, a successive labeling procedure produces an optimal of node disjoint path P_2 and continuing inductively. Conclusively, the optimal

$K-1$ node disjoint text line paths produce an optimal set of K node disjoint text line paths.

In initialization, the Bellman-Ford algorithm [95] is used to generate the single shortest text line path. At the K^{th} iteration, the K shortest text line paths are updated by the previous $K-1$ shortest text line paths in three steps, namely, graph transformation, interlacing generation, and path augmentation. Algorithm 3 summarizes the text line extraction based on k-shortest paths optimization. The basic idea is to find the optimal text line paths $P = p_1, p_2, ..., p_k$ inductively by applying a shortest path algorithm. At each iteration, the text line paths are augmented by the previous text line paths and the interlacing of the transformed directed graph. Finally, the global optimal text lines can be achieved when the total cost changes its sign, and then the optimal text line number k can be obtained.

4.2.1 Directed graph transformation

The directed graph transformation is to define an equivalent network G^* from the original directed graph G_D. This procedure can be used to find $n+1$ disjoint text line paths inductively from the optimal P_n disjoint text line path solution by using a shortest path algorithm at each iteration.

Each vertex v_i in P_n, except for the virtual source and sink vertices, is split and introduced to an auxiliary vertex v_i'. All outputs on v_i are then assigned as outputs on v_i'. The cost value of the auxiliary directed edge $e_{i,i'}$ from vertex v_i to v_i' is set to 0. Then, the direction and algebraic sign of edge cost in P_n, including the auxiliary edges, are subsequently revised. It leaves all paths and edges exactly the same as in the directed graph, but the path cost in the directed graph is now defined strictly in terms of edge

Algorithm 3 Text line extraction based on k-shortest paths global optimization.

Input:

The set of extracted text character vertices, $V = \{v_i\}_{i=1}^{n}$.

Output:

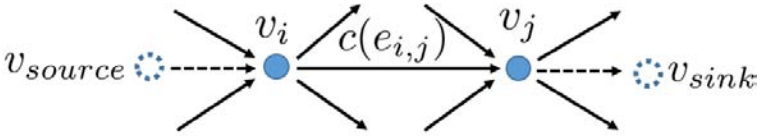
The optimal set of text line paths $P_i = p_1, p_2, ..., p_i$ between the virtual source v_{source} and sink vertices v_{sink}.

Construct the directed graph G_D based on the extracted text characters and calculate the cost value of each edge
Calculate the shortest path p_1^* between v_{source} and v_{sink} in the directed graph
$P_1 \Leftarrow p_1^*$
for $i = 2$ to n **do**
    Transform the directed graph $G^* \Leftarrow G_D$
    Transform the cost of the edges $G_c^* \Leftarrow G^*$
    Calculate the interlacing p^* of the transformed directed graph
    Augment the text line paths $P_{i+1} \Leftarrow P_i \bigoplus p^*$
    if $cost(P_{i+1}) \leq cost(P_i)$ **then**
        return $P_i = p_1, p_2, ..., p_i$
    end if
end for

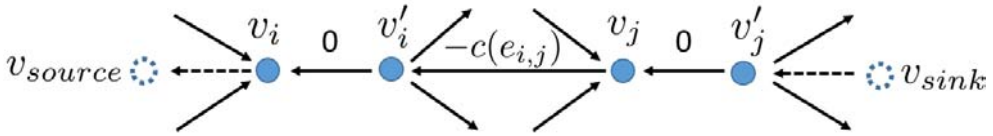
cost. The vertex split addresses the node disjoint criteria, which are relaxed to edge disjointness. The direction and algebraic signed reversion indicate a transformation from signed paths to directed unsigned paths.

One example of directed graph transformation is shown in Figure 4.5. In the original directed graph, the shortest path $p = v_{source}, v_i, v_j, v_{sink}$ is extracted in the first iteration. We split the vertices v_i and v_j to auxiliary vertices v_i' and v_j' after excluding the virtual source and sink vertices, and then revise the direction and algebraic sign of the edge cost including the auxiliary edges.

Additionally, we apply the directed edge cost transformation to graph G^* by revising the negative cost of directed edges to a non-negative edge cost. This procedure generates



(a) One shortest path in the original graph



(b) Graph transformation result

FIGURE 4.5: An example of directed graph transformation.

a canonic equivalent graph G_c^*. For the cost value of an edge cost $c_{e_{i,j}}$ from vertex $v_i \in V$ to vertex $v_j \in V$, suppose that the transformed cost value $c^*_{e_{i,j}}$ is calculated by the shortest path cost from v_{source} to v_i, v_j and that the cost value $c_{e_{i,j}}$ before graph transformation can be computed as

$$c^*_{e_{i,j}} = c_{e_{i,j}} + c_{p_{source,i}} - c_{p_{source,j}}, \tag{4.1}$$

where $c_{p_{source,i}}$ denotes the cost value from v_{source} to v_i. Conclusively, the cost of transformed graph G^* can be calculated by the shortest path cost of the vertices in the original graph G. The signed path cost after the transformation is unchanged between the source and sink vertices. As one of its advantages, the cost transformation reduces the complexity of shortest path computation by converting the unsigned directed graph into a signed directed graph.

4.2.2 Interlacing generation

Let P_i be the optimal set of i paths at iteration i. The interlacing generation aims to extract a node-simple shortest signed directed path p^*, called interlacing of P_i. The signed path p^* interlaces P_i in the given transformed graph and comprises i node disjoint

directed paths form the virtual source vertex to the sink vertex at iteration i. This path must satisfy the following two criteria:

- the interlacing directed edge should be common to both p^* and P_i if and only if it has a negative label; and
- the interlacing vertex should be common to both p^* and P_i if and only if it is incident to a directed edge with a negative label.

The first criterion is necessary to obtain the edge disjoint text line paths in i-th iteration yet is not sufficient to extract the node disjoint text line paths. The second criterion can exclude the signed text line paths with a single vertex overlapping with P_{i-1}, which is a complement for node disjoint paths.

In our implementation, we apply the Dijkstra's single source shortest path algorithm to generate the interlacing p^*. The shortest path algorithm is suitable for the transformed directed graph due to the non-negative directed edge cost. Afterward, we can augment K-shortest paths in the k iteration by using the $K-1$ shortest paths and the interlacing of the transformed directed graph in iteration $k-1$.

4.2.3 Path augmentation

Path augmentation aims to produce the $k+1$ shortest text line paths P_{k+1} based on interlacing p^* and k shortest paths P_k in the previous iteration. This procedure adds edges to P_k with positive interlacing labels and removes those edges from P_k with negative interlacing labels:

$$P_{k+1} = p^* \oplus P_k. \tag{4.2}$$

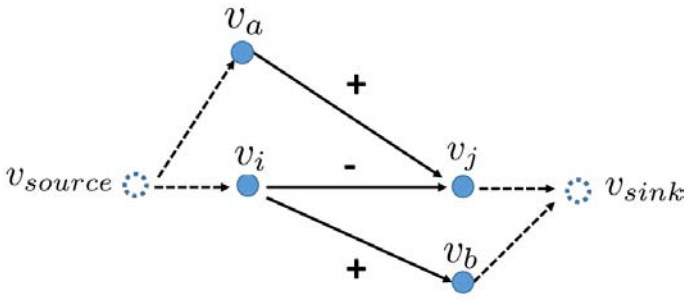


FIGURE 4.6: An example of path augmentation based on interlacing and previous shortest paths.

Figure 4.6 illustrates an example of shortest path augmentation. Assume that the shortest path is $p_1 = v_{source}, v_i, v_j, v_{sink}$ and the shortest interlacing is p^*=v_{source}, v_a, v_j, v_i, v_b, v_{sink} with the edge label $\{+, +, -, +, +\}$. The directed edges e_{v_a,v_j} and e_{v_i,v_b} will be added into the shortest path P_1, and the edge e_{v_i,v_j} will be removed from the shortest path P_1. Finally, the updated paths $v_{source}, v_a, v_j, v_{sink}$ and $v_{source}, v_i, v_b, v_{sink}$ can be obtained via path augmentation as defined in Equation 4.2.

4.2.4 Text line paths global optimization

Our text line extraction problem can be treated as a minimum cost flow problem with a 0-1 flow constraint. If 1, it suggests that the edge is part of a text line. The k-shortest paths algorithm is well-studied and widely applied [96] for path selecting and routing in the directed graph. We take advantage of the particular structure of the constructed directed graph to obtain the global optimal by using the k-shortest paths optimization algorithm. The target text line extraction problem aims to find the optimal solution that minimizes the cost function between the source v_{source} and sink vertices v_{sink} in the directed graph. The optimal solution f of the k-shortest paths optimization can be defined as:

$$f = \underset{G_D=(V,E)}{\arg\min} \sum c(e_{i,j}) \cdot l(e_{i,j}), \tag{4.3}$$

where $c(e_{i,j})$ and $l(e_{i,j})$ denote the cost function and label of edges $e_{i,j} \in E$ for vertices $i \in V$ and $j \in V$ in the directed graph G_D, respectively.

The k-shortest paths global optimization algorithm tries to find k paths $p_1, p_2, ..., p_k$ iteratively with the minimum total cost value in the constructed directed graph, where k is settled. Any path between v_{source} and v_{sink} in the directed graph represents a feasible path flow of a text line. In our text line extraction case, no text MSER component can be shared by two text lines, thereby suggesting that the extracted k paths must be vertex disjoint and that each vertex must be included in one path at most. We assume that $P_k = p_1, p_2, ..., p_k$ is the set of all k shortest paths calculated in iteration k and p_l is one of the shortest paths in the text line set of P_k. The cost value of the single shortest text line path p_l can be calculated as follows by the cost of all graph edges $e_{i,j}$ belonging to the l^{th} shortest path:

$$c(p_l) = \sum_{e_{i,j} \in p_l} c(e_{i,j}). \tag{4.4}$$

We can calculate the cost value of each shortest text line path $l = 1, 2, 3, ..., k$ at iteration k. The total cost value $c(P_k)$ of the k-shortest paths at the k^{th} iteration is formulated as follows:

$$c(P_k) = \sum_{i=1}^{k} c(p_i). \tag{4.5}$$

We compare the total cost value $c(P_n)$ of the new iteration n with the cost of the previous iteration $c(P_{n-1})$ and observe that the total path cost is monotonically increasing. The

FIGURE 4.7: Text line extraction based on k-shortest paths optimization.

global minimum is achieved when the cost value changes its sign and becomes decreasing. Meanwhile, the optimal parameter is obtained as n which satisfies the following conditions:

$$\begin{cases} c(P_{n-1}) \leq c(P_n) \\ c(P_n) \geq c(P_{n+1}). \end{cases} \tag{4.6}$$

Figure 4.7 illustrates an example of the text line paths global optimization between the virtual source and sink vertices. The blue and purple dashed lines denote the virtual directed edges from the virtual source vertex to the text character, and from the text character to the virtual sink vertex, respectively. The cost value of the virtual edges is set to 0. The green path flows that connect the bounding box of the text character components denote the route of text lines.

4.3 Experiments and Analysis

In this section, we demonstrate the experimental results of k-shortest paths global optimization to show the effectiveness of the proposed method. By analyzing these experimental results, we find that the k-shortest paths optimization demonstrates competitive performance in text line extraction.

4.3.1 Experiment setting

To evaluate the effectiveness and robustness of the proposed text line extraction method, we perform the text line extraction experiments on the public benchmark scene text detection database ICDAR2011 [71] and ICDAR2013 [72]. Afterward, we compare the proposed method with the other methods for scene text detection on these two benchmarks.

The ICDAR2011 database is extended from the earlier ICDAR2003 [65] and ICDAR2005 [97] databases. This database is used in competitions along with ground truth in XML format and is publicly available at "http://algoval.essex.ac.uk/icdar/Datasets.html". The images in the database are captured by a digital camera using auto focus and natural lighting, and the text in these images is presented in various colors and fonts, on many different complex backgrounds, and in various orientations. The ground truth includes the bounding boxes of the coordinates of the text regions and is stored in a separate file that corresponds to an image file. The ICDAR2011 database contains 484 images comprised of 229 images for training and 255 images for testing.

The ICDAR2013 database, which is a subset of ICDAR2011 database, comprises 229 training images and 233 testing images. This database removes a small number of

duplicated images over the training and testing sets. Additionally, the ICDAR2013 database revises several ground truth annotations the in ICDAR2011 database.

4.3.2 Experimental results and analysis

The traditional performance metrics, namely, precision, recall, and f-measure [66], are adopted to quantify the performance based on text line region matching. The metrics compare the detected text lines with the annotated ground truth text lines. The text line region matching $m_r(G_i, E_j)$ is given by Equation 2.15. The best matching of the extracted text line region and ground truth text line region can be computed as:

$$Match_G(G_i) = \max_{j=1,\ldots,|E|} \frac{2 \times m_r(G_i, E_j)}{|G_i| + |E_j|}, \tag{4.7}$$

$$Match_E(E_j) = \max_{i=1,\ldots,|G|} \frac{2 \times m_r(G_i, E_j)}{|G_i| + |E_j|}, \tag{4.8}$$

where, I is the image region, and G_i and E_j denote the i^{th} ground truth region and the j^{th} extracted text line region, respectively.

All performance measures are independently calculated on each image, and the average value over all images is calculated as performance of the proposed method. Afterward, we count the number of matches according to the overlap between the labeled text lines and ground truth text lines. Two performance measures, namely, recall and precision, are calculated as

$$recall = \frac{\sum_i^{|G|} Match_G(G_i)}{|G|}, \tag{4.9}$$

$$precision = \frac{\sum_{j}^{|E|} Match_E(E_j)}{|E|}. \tag{4.10}$$

Recall measures the ratio between the extracted true positives and all true text lines, while precision measures the ratio between the extracted true positives and all extracted text lines. Both of these measures are defined based on the text line area matching. The f-measure, as an overall measurement, is computed as follows by combining the recall rate and precision rate for equal weight:

$$f = \frac{2 \times recall \times precision}{recall + precision}. \tag{4.11}$$

The performances of the proposed method on the ICDAR2011 database are shown in Table 4.1. The proposed method achieves a recall performance of 0.716, precision performance of 0.902, and f-measure of 0.798. Compared with state-of-the-art methods, the proposed k-shortest paths optimization based text line extraction method achieves a promising performance in terms of recall, precision, and f-measure. Although the recall performance is 4.6% lower than the best performance, the precision is 4.0% higher than the best performance. The over-all f-measure is only 1.1% lower than the best performance.

Figure 4.8 illustrates the iterative extraction of text lines based on k-shortest paths optimization. The target image is given in Figure 4.8(a). In the first iteration, the path "Mastering the JFC EDITION" (Figure 4.8(b)) is extracted from the directed graph and the text lines "Mastering the JFC" and "EDITION" are combined into one path due to minimum cost optimization. The k-shortest paths optimization updates the text line paths iteratively. The new text line paths "graphic", "JAVA", and "3RD" (Figure 4.8(c)-(e)) are updated successively from the second iteration to the fourth iteration by the

TABLE 4.1: Experimental results of k-shortest paths optimization-based text line extraction on the ICDAR2011 database.

Method	recall	precision	f-measure
[98]	0.581	0.672	0.623
[60]	0.647	0.731	0.687
[56]	0.631	0.833	0.718
[59]	0.644	0.812	0.719
[99]	0.75	0.82	0.73
[50]	0.683	0.863	0.762
[43]	0.760	0.860	0.800
[51]	0.762	0.862	0.809
K-shortest paths optimization method	0.716	0.902	0.798

interlacing and previous path. In fifth iteration, the text line paths "Mastering the JFC" and "3RD" (Figure 4.8(f)) are correctly revised from the first text line path "Master the JFC EDITION" for the globally optimal solution. The total cost value of these paths is monotonically increasing from the first iteration to the fifth iteration. We achieve the global optimal and the optimal path flow number of text lines when the total cost value changes its sign. In Figure 4.8, the five text lines "Mastering the JFC," "graphic," "JAVA," "3RD," and "EDITION" are extracted.

The proposed method can extract the text line paths iteratively to produce the text line paths as shown in Figure 4.9. Each character only appears in a single text line path and cannot be shared by two text lines because of the node-disjoint of the k-shortest paths optimization algorithm. Each ground truth text line path can be extracted by an average of 1.94 iterations in the gray scale channel, thereby validating the effectiveness of the proposed method.

Table 4.2 presents the performance of the proposed method on the ICDAR2013 database.

(a) Target image (b) #iter 1

(c) #iter 2 (d) #iter 3

(e) #iter 4 (f) #iter 5

FIGURE 4.8: Iterative extraction of text lines based on k-shortest paths optimization.

We produce recall, precision, and f-measure values of 0.703, 0.898, and 0.789, respectively. In sum, compared with state-of-the-art methods, the proposed text line extraction method achieves a competitive performance in terms of recall, precision, and f-measure.

Figure 4.10 presents some successful and failure examples of successful and failed text

FIGURE 4.9: Examples of text line paths extraction results.

TABLE 4.2: Experimental results of k-shortest paths optimization based text line extraction on ICDAR2013 database.

Method	recall	precision	f-measure
[100]	0.651	0.840	0.734
[101]	0.687	0.854	0.762
[102]	0.743	0.858	0.797
[43]	0.740	0.880	0.800
[51]	0.759	0.852	0.803
K-shortest paths optimization method	0.703	0.898	0.789

line extractions. The green, blue, and red text line bounding boxes represent the correct, false positive, and failed of text line extraction, respectively. Our method is robust to skew text lines due to the overlap between the adjacent characters in the same text line as shown in Figure 4.10(e). Meanwhile, the experimental results illustrate that the proposed method can also extract multiple (Figure 4.10(a)-(d)), low contrast (Figure 4.10(f)), highlighted (Figure 4.10(g)), and complex background (Figure 4.10(h)) text lines.

Although the proposed method demonstrates promising performance, some limitations need to be considered. Firstly, the k-shortest paths optimization-based text line extraction method is incapable of dealing with single character text lines (Figure 4.10(i)) due to no path existing in the constructed directed graph. Secondly, some background regions that resemble text characters can lead to a high text probability. This is because

(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

FIGURE 4.10: Examples of the text line extraction results.

these false text characters appears as one of the text line path and consequently degrade the precision. Thirdly, some text characters may not be extracted due to highlight or complex backgrounds, which will lead to the failure (red text line in Figure 4.10(b)) of text line extraction.

4.4 Conclusion

This chapter aims to optimize the text line paths by taking advantage of the particular structure of the constructed directed graph. The basic idea is to formulate the text line

extraction problem as k-shortest paths global optimization in the directed graph. This procedure iteratively generates the text line paths with minimum cost by augmenting the previous iteration. Our experimental results reveal that the text line paths global optimization can successfully extract the text lines iteratively. The global optimization can also identify the number of the text line automatically to avoid an exhaustive search, which is suitable for various appearances of text lines in a given image. This method can also reject the false positive noises in all of the text line paths, thereby reducing the number of noisy text lines.

Chapter 5

Multi-Channel Paths Optimization for Text Line Extraction

5.1 Introduction

As mentioned in [60], only 85.6% of text characters can be extracted as MSERs in the single gray channel due to the complex situation of natural scene images. Text character extraction is the first step of text line extraction, and its performance determines the text line extraction performance. Obviously, the recall measure is less than 85.6%, and single channel characters is insufficient for text line extraction. The text characters exhibit intensity disparity in different channels, and the intensity disparity can contribute to extract various text character components in different channels. If other channels of image can be combined, then the character recall performance can be significantly

FIGURE 5.1: The disparity of pixel intensity in different channels.

improved up to 94.8%. Meanwhile, the text line extraction performance can be improved further.

The global optimization of multi-channel text line paths aims to integrate various image channels into one framework and extract the text line paths in a global way. As shown in Figure 5.1, the text line "LION WALK" failed in gray channel will be successfully extracted in blue channel. Therefore, the properties of multi-channel text characters seem beneficial for high performance text line extraction.

This chapter introduces a method to optimize the multi-channel text line paths by integrating the text character components of various channel into one directed graph. As shown in Figure 5.2, we firstly extract the text character in gray, red, green, and blue channels. Secondly, the multi-channel directed graph is built to incorporate all the text characters from all channels. Thirdly, we explore the multi-channel text line paths global optimization for text line extraction. Finally, the extracted text lines are certified by various channels, and the noise duplicated text lines are removed.

The proposed method is supposed to have the following merits:

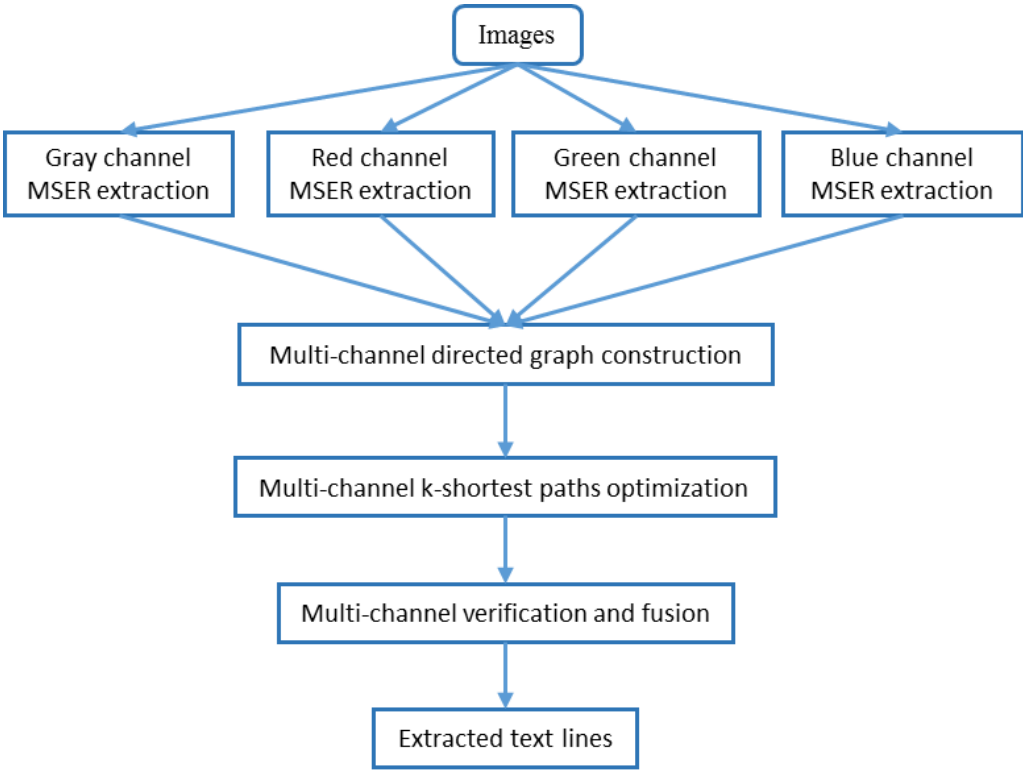


FIGURE 5.2: Flowchart of our proposed multi-channel k-shortest paths optimization for text line extraction.

- The constructed multi-channel directed graph can effectively combine the text characters of various channel into one directed graph, and describe the relationship structure of text character components between different channels.

- The text line paths can be globally optimized between different channels by the multi-channel k-shortest paths optimization. Furthermore, the extracted text lines can be verified by duplicated text lines from other channels.

- We demonstrate that the multi-channel k-shortest paths optimization-based text line extraction method can effectively improve the performance of text line extraction by incorporating candidate text components of various channels into one framework.

5.2 Multi-channel Text Character Extraction

We extract the candidate text components by the MSER method in gray, red, green, and blue channels [60]. Each channel consists of two polarized text characters, which are black and white text character embedded in the opposite background. Then, the non-characters noises are removed by the designed CNN filter described in 3.2.2.

MSER extraction in red, green, and blue channels can make up for the deficiency of the text character extraction of the gray image. Several missing text characters can be extracted in the red, green, and blue channels due to the intensity disparity in various channels. Thus, the channel combination is capable of improving the recall performance of the candidate text components. Furthermore, the candidate text components can be certified by various channels.

5.3 Multi-channel Directed Graph Construction

We incorporate the text characters of gray, red, green, and blue channels into one multi-channel directed graph. Not only can the relationship of neighboring text characters from each channel built, but the relationship of different channel text characters can also be incorporated into one framework. Due to the absence of interaction of text characters from various channels, the constructed multi-channel directed graph is channel-disjoint and no directed edge connects different channels. The text line paths and their duplicated paths are assumed to be directed path segments in the constructed multi-channel directed graph.

We supposed that the constructed multi-channel directed graph is $G_m = (V_m, E_m)$, where V_m and E_m are the set of vertices and directed edges in the multi-channel directed

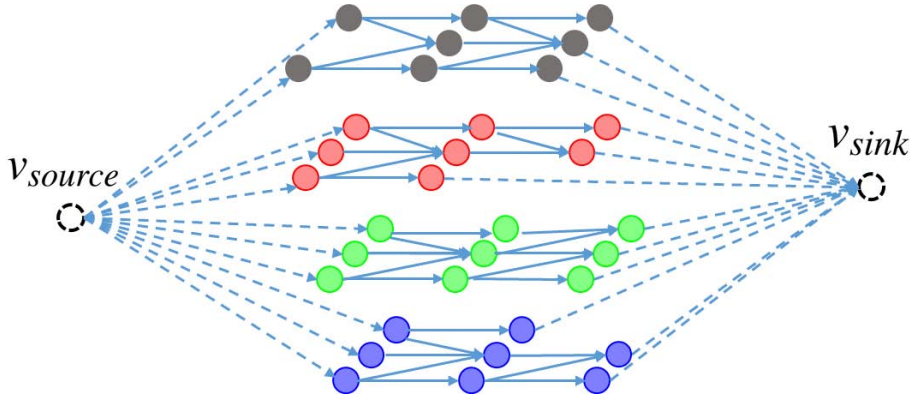


FIGURE 5.3: Multi-channel directed graph comprised of gray, red, green, and blue channels.

graph. The vertices $\{V_m | m = \{gray,\ red,\ green,\ blue\}\}$ are comprised of text character candidates from the gray, red, green, and blue channels. The directed edges build the linking relationship of adjacent text characters at each channel.

Figure 5.3 illustrates the simplified results of the constructed multi-channel directed graph. A total of four text character layers corresponding to gray (gray nodes), red (red nodes), green (green nodes), and blue (blue nodes) channels. The text characters that fail to be extracted in the gray channel may be extracted in the red, green, or blue channels due to the intensity disparity at various channels. Note that the text characters are extracted at each channel respectively, any text character cannot be shared by other channels, and no interaction occurs between different channels.

Similar to the single channel directed graph construction, our method adds another additional virtual source vertex v_{source} and sink vertex v_{sink} to the multi-channel directed graph. Our method allows each vertex to be the start or end location of one text line. The vertex v_{source} is the directed graph source node and the first virtual character of the text line. The vertex v_{sink} is the directed graph sink node and last virtual character of the text line. We build an directed edge $e_{source,i}$ from the virtual source vertex v_{source} to vertex v_i and $e_{i,sink}$ from vertex v_i to vertex v_{sink} for each vertex in all channels.

The cost value of the directed edge with virtual vertices is set as 0 to allow each text character to be the starting or ending location of the text line at no cost in all channels.

In the multi-channel directed graph, we also build the directed edge in the assigned text line direction from left to right according to the spatial location of the text character region, which is the same as the single channel directed edge. For each vertex v_i in the gray channel, the distance constraint and overlapping constraint are checked with other vertices in the same channel to remove unnecessary computation. According to the assigned text line direction, the x-coordinate cx_j of target vertex v_j should be larger than that of source vertex v_i in the directed edge. Then, the nearest text character vertex v_j is selected from all the candidate pairs of vertex $p_{i,j}$ to build the directed edge $e_{i,j}$. Finally, the directed edges in the red, green, and blue channels can be built by the same method. Algorithm 4 provides the details of the multi-channel directed graph construction.

Each directed edge in the multi-channel directed graph is attached a cost function comprised of unary cost function and pairwise cost function stated in Chapter 3.3.3. The appropriate cost of each edge is set according to a probability that two vertices are neighboring characters in a text line. Similarly, the cost value of directed edges with virtual source and sink vertex is set to 0 to allow each text character to be the start or end text at no cost in the multi-channel directed graph. The normalized item of unary cost and pairwise cost is based on the total cost value of directed edge in the multi-channel directed graph

Algorithm 4 Multi-channel directed graph construction.

Input:

Set of extracted MSER vertices in the gray, red, green, and blue channels.

Output:

Constructed multi-channel directed graph, $G_m = (V_m, E_m)$.

for $channel = \{gray,\ red,\ green,\ blue\}$ **do**
  for $i = 1$ to n_c **do**
    Build an directed edge $e_{source,i}$ from virtual source vertex v_{source} to vertex v_i;
    Build an directed edge $e_{i,sink}$ from vertex v_i to virtual sink vertex v_{sink};
    for $j = 1$ to n, $j \neq i$ **do**
      if $x_j > x_i$ **then**
        Verify the pair by the overlapping and distance constraints in the corresponding channel;
        Build the component pair $p_{i,j}$ between vertex v_i to v_j, and add it to the candidate set $\{p_{i,j}\}$;
      end if
    end for
    Find the nearest vertex v_j with minimum distance in the set of $\{p_{i,j}\}$;
    Build directed edge $e_{i,j}$ from vertex v_i to vertex v_j;
  end for
end for
return $G_m = (V_m, E_m)$.

5.4 Multi-channel K-shortest Paths Optimization

We can assume that the correct text lines are included as paths in the multi-channel directed graph, and the target text line extraction problem can be formulated as a minimum cost flow optimization with a 0-1 constraint in the multi-channel directed graph. Then, we can obtain a globally optimal extraction result of the text lines. If 1, then the edge and two neighboring text character vertices are part of one text line. Otherwise, these two text characters cannot be in one text line. The optimal solution

f of the multi-channel k-shortest paths optimization between the source vertex v_{source} and sink vertex v_{sink} is defined as follows:

$$f = \underset{G_m=(V_m,E_m)}{\arg\min} \sum_{e_{i,j} \in E_m} c(e_{i,j}) \cdot l(e_{i,j}), \tag{5.1}$$

where $c(e_{i,j})$ denotes the cost value of edge $e_{i,j} \in E_m$ for vertex $i \in V_m$ and vertex $j \in V_m$ in the multi-channel directed graph. The $l(e_{i,j})$ denotes the label of edge $e_{i,j} \in E_m$ and the value of $l(e_{i,j})$ is 0 or 1 for optimization.

We employ the same k-shortest paths optimization algorithm as single channel text line extraction. This algorithm can extract k text line paths iteratively with minimum total cost in the multi-channel directed graph. In the multi-channel text line extraction case, no text character vertex can be shared by two channels and two text lines, it means the extracted text line paths are vertex and channel disjoint. At each iteration, the total cost value of extracted paths is compared with the previous one to find the global optimal when the cost value changes sign and becomes decreasing.

5.5 Fusing Text Lines in Multiple Channels

We produce the text line paths in gray, red, green, and blue channels by the k-shortest global optimization algorithm in the multi-channel directed graph. Some repeating text lines may exist in other channels and the repeating text lines should be removed to avoid repeat extraction. Therefore, we fuse the repeating text lines in different channels into one text line. If all of the text components in the text line are included in the text lines from other channels, then the included text line should be removed as repeating noise.

Finally, only one text line will be retained for the repeating text lines in gray, red, green, and blue channels.

5.6 Experiments and Analysis

In this section, the multi-channel text line extraction method is compared with gray channel k-shortest paths optimization method and state-of-the-art methods on the ICDAR2011 and ICDAR2013 databases. The experimental results demonstrate that although the precision result of the multi-channel k-shortest paths optimization method is lower than that of the gray channel text line extraction, it achieves much higher recall and f-measure performance. The results prove that the performance of the k-shortest paths optimization for text line extraction can be improved by the combination of red, green, and blue channels.

5.6.1 Experiment setup

In the multi-channel text line extraction experiments, we also use the same parameter settings for MSER text character extraction, which is given in Table 3.1. Five parameters named "Δ", "maxWH", "minWH", "$maxVariation$", and "$minDiversity$" are used to control the text character extraction performance. The threshold step is set as 4 pixels for region production in the MSER root tree. The minimum and maximum width and height of the text character region are restricted between 10 and 600 pixels, respectively. Meanwhile, the regions with "maxVariation" larger than 0.5 and "minDiversity" smaller than 0.5 are pruned to obtain the text character with high probability.

To fairly compare the gray channel text line extraction and multi-channel text line extraction method based on the k-shortest paths optimization, other parameters in the

multi-channel text line extraction are set the same as the gray channel text line extraction. The coefficient λ of MSER variation penalization and distance parameter α of potential neighboring vertices are set as 0.33 and 3, respectively. We evaluate these two methods on the ICDAR2011 and ICDAR2013 databases. Furthermore, the same performance metrics, namely, precision, recall, and f-measure, which are described in 4.3.2, are applied to quantify the performance.

5.6.2 Experiment results and analysis

The experimental results of the multi-channel text line extraction on the ICDAR2011 database are shown in Figure 5.4. The single gray channel text line extraction can achieve a recall of 0.716. When we add one red, green, or blue channel, the recall of the two channel combination text line extraction can increase from 0.729 to 0.739, and the recall performance can be improved by 1.8% on average. When we add two channels of red, green, and blue channels, the recall of the three channel text line extraction can be achieved from 0.745 to 0.753, and the recall performance can be improved by 3.3% on average. After we integrate all the red, green, and blue channels for multi-channel combination, we can achieve the recall performance of 0.781, which represents a 6.4% improvement.

Among all the results, the single channel text line extraction has the lowest f-measure. The multi-channel text line extraction achieves the highest f-measure by combining gray, red, green, and blue channels, and the recall improves significantly by 6.4%. The reason is that the multi-channel text line extraction can extract more text character candidates and optimize the combined multi-channel directed graph. Through the comparison, we may conclude that the multi-channel text line extraction method can improve the performance of single channel text line extraction.

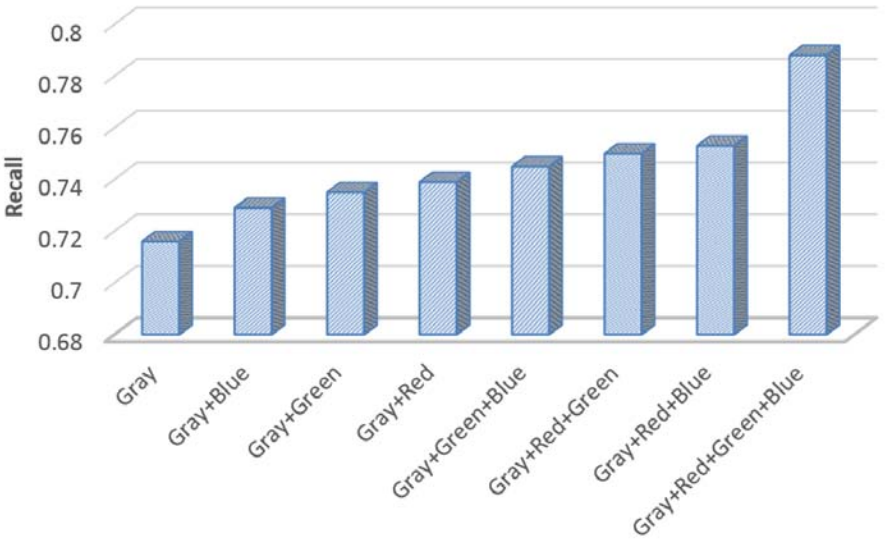


FIGURE 5.4: Comparison of experimental results of multi-channel paths optimization-based text line extraction on ICDAR2011 database.

Figure 5.5 illustrates an example of the comparison between single channel text line extraction and multi-channel text line extraction. The target image has a total of three text lines: "BRITAIN's FAVOURITE DEPARTMENT STORE", "www.debenhams.com", and "DEBENHAMS" in the target image. In the single channel text line extraction, the text line "BRITAIN's FAVOURITE DEPARTMENT STORE" can not be extracted due to the enfoldment and light reflection. While in the multi-channel text line extraction, we can extract the missing text line successfully in the red, green, or blue channels. Obviously, the red, green, and blue channels can be supplements of gray channel text line extraction.

Table 5.1 presents the performance comparison between the proposed multi-channel text line extraction and other methods. We should note that the [60] method is multi-channel method, and other methods in the table are not multi-channel methods. The text line extraction in the gray channel is easy to be extended to the multi-channel mode by combining the red, green, and blue channels. The performance of text line extraction is expected to be improved by the multi-channel combination as a supplementary. Our

(a) (b)

FIGURE 5.5: Text line extraction results of (a) gray channel k-shortest paths optimization and (b) multi-channel k-shortest paths optimization method.

TABLE 5.1: Experimental results of k-shortest paths optimization-based multi-channel text line extraction on ICDAR2011 database.

Method	recall	precision	f-measure
[98]	0.581	0.672	0.623
[60]	0.647	0.731	0.687
[56]	0.631	0.833	0.718
[59]	0.644	0.812	0.719
[99]	0.75	0.82	0.73
[50]	0.683	0.863	0.762
[43]	0.760	0.860	0.800
[51]	0.762	0.862	0.809
Multi-channel combination method	0.781	0.863	0.820

k-shortest paths optimization-based multi-channel text line extraction method achieves a recall of 0.781, and precision performance of 0.863 and f-measure of 0.820. Compared with state-of-the-art methods, our multi-channel text line extraction method performed better performance in terms of recall, precision, and f-measure. This result proves the advantage of our method and encourages us to apply the proposed method in text line extraction.

Table 5.2 shows the performance of the multi-channel text line extraction on the ICDAR2013 database. The multi-channel text line extraction method achieves a recall of

TABLE 5.2: Experimental results of k-shortest paths optimization-based multi-channel text line extraction on ICDAR2013 database.

Method	recall	precision	f-measure
[100]	0.651	0.840	0.734
[101]	0.687	0.854	0.762
[102]	0.743	0.858	0.797
[43]	0.740	0.880	0.800
[51]	0.759	0.852	0.803
Multi-channel combination method	0.772	0.856	0.812

0.772, precision of 0.856, and f-measure of 0.812. These encouraging results proved that our method outperforms the others.

Figure 5.6 presents the comparison examples between single channel and multi-channel text line extraction. The green, blue, and red text line boxes denote the correct, false positive, and failure of text line extraction results, respectively. The first and second row images provide the examples of single channel text line extraction and multi-channel text line extraction. In Figure 5.6(a) and (b), the text lines of "LION WALK" and "BRITAIN's FAVOURITE DEPARTMENT STORE" are missed due to the failure of character extraction in the gray channel. The text line "ONLY" in Figure 5.6(c) is partly extracted because of the failure of characters extraction of "ON". However, these text lines can be successfully extracted by the multi-channel text line extraction method. Obviously, the multi-channel text line extraction can improve the text line extraction performance of single channel text line paths optimization.

Although the proposed multi-channel text line extraction based on k-shortest paths optimization achieves promising results, it has several limitations. Firstly, the multi-channel text line extraction will decrease the precision performance due to additional noises introduced in the red, green, and blue channels. Secondly, this method is also

FIGURE 5.6: Examples of comparison between single channel and multi-channel text line extraction by the k-shortest paths global optimization.

incapable of dealing with text lines with a single character because such a text line is no path existing in the multi-channel directed graph.

5.7 Conclusion

In this chapter, the multi-channel text line extraction based on k-shortest paths optimization is investigated in depth. The multi-channel text line extraction can achieve promising performance through the complementary usage of the red, green, and blue channels in the directed graph. The comparative study of various channel combinations also shows the potential of the channel combination text line extraction method. With various channel combinations, the multi-channel text line extraction can present a different performance. In our experiments, we show that the recall performance can be

improved significantly when the red, green, and blue channels are combined in the text line extraction.

Chapter 6

Efficiency Improvement of Path Optimization in Multiple Channels

6.1 Introduction

As mentioned in Chapter 5, the multi-channel text line paths optimization can combine the characters of various image channels and extract the text line paths in a global way. The multi-channel combination can be beneficial for improving text line extraction accuracy. However, the repeating text characters introduced in the red, green, and blue channels increase the complexity of the directed graph and text line paths optimization. For the extracted text line in the gray channel, similar duplicate text lines or text line segments may exist in the red, green, and blue channels. We remove the repeating the path segments as noises to avoid repeating extraction. Thus, the number of vertices and

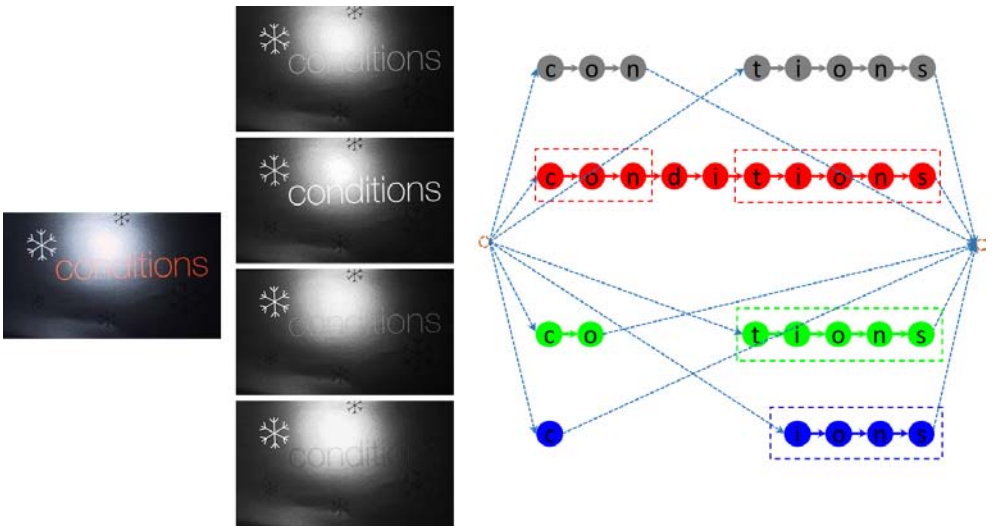


FIGURE 6.1: The directed graph construction of multi-channels.

edges can be reduced and the efficiency of paths optimization in multiple channels can be improved.

As shown in Figure 6.1, the characters in the text line "conditions" may be extracted several times in multiple channels. The number of vertices and edges will be much larger than that of the previous single channel directed graph. Suppose that we extract the guiding text lines "con" and "tions" in the gray channel, the repeating unambiguous path segment $v_{\text{“}t\text{”}}$, $v_{\text{“}i\text{”}}$, $v_{\text{“}o\text{”}}$, $v_{\text{“}n\text{”}}$, $v_{\text{“}s\text{”}}$ can be reduced to $v_{\text{“}t\text{”}}$, $v_{\text{“}s\text{”}}$ in red and green channels. The repeating unambiguous path segment $v_{\text{“}i\text{”}}$, $v_{\text{“}o\text{”}}$, $v_{\text{“}n\text{”}}$, $v_{\text{“}s\text{”}}$ can be reduced to $v_{\text{“}i\text{”}}$, $v_{\text{“}s\text{”}}$ in blue channel. The repeating unambiguous path segment $v_{\text{“}c\text{”}}$, $v_{\text{“}o\text{”}}$, $v_{\text{“}n\text{”}}$ can be reduced to $v_{\text{“}c\text{”}}$, $v_{\text{“}n\text{”}}$ in red channel. Therefore, the complexity of vertices and edges in the directed graph can be decreased, and the global k-shortest paths optimization in the reduced directed graph can be facilitated.

In this chapter, we propose a novel integrated paths optimization in the integrated directed graph to improve the efficiency. The integrated directed graph construction aims to integrate different channels into one framework and eliminate the duplicate path segments. The integrated text line extraction aims to extract text lines in the

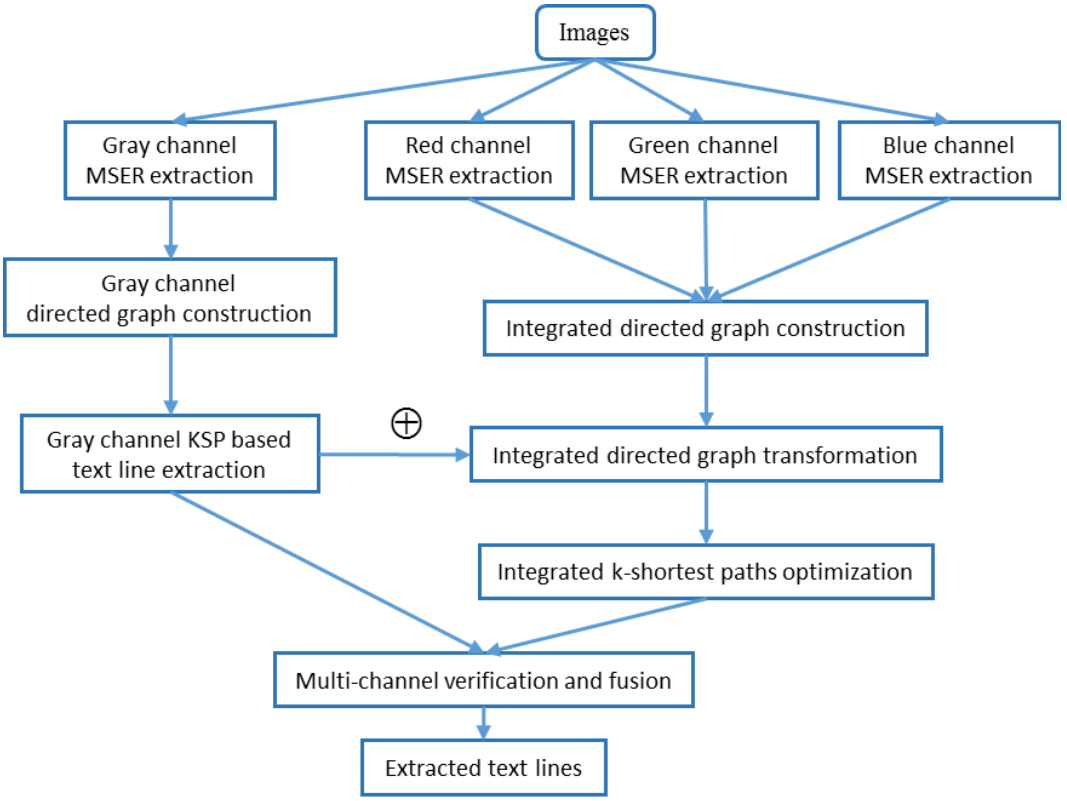


FIGURE 6.2: Flowchart of our proposed integrated k-shortest paths optimization-based text line extraction.

integrated directed graph by paths optimization. To improve the efficiency and decrease the complexity of the directed graph, we transform the directed graph of multi-channels into a reduced graph guided by the extracted text lines of the gray channel. Finally, the k-shortest paths optimization algorithm is utilized to extract the text lines by taking advantage of the particular structure of the integrated directed graph. The flowchart of the proposed method is shown in Figure 6.2.

In this chapter, we discuss the details of integrated directed graph transformation based on multi-channel directed graph for text line extraction. The discussion discovers how it performs to improve the efficiency of text line extraction. The main contributions are described as follows in comparison with the text line extraction of multi-channels:

- The integrated directed graph can effectively describe the relationship structure and eliminate the disorder of text components among different channels.

- The integrated directed graph can be transformed guided by the extracted text lines in the gray channel, which can improve the efficiency and decrease the complexity of the directed graph.

6.2 Methodology of Efficiency Improvement

For integrated text line extraction, we apply the same MSER and noise removal methods as the multi-channel text line extraction in the gray, red, green, and blue channels to extract the text character candidates. To incorporate different channels into one reduced framework, we construct a reduced integrated directed graph on the red, green, blue channels without the gray channel. The reduced directed graph is transformed based on the certificated text line segments compared with the text line extraction results in the gray channel. Therefore, the directed edges can be original or transformed edges. The target text lines are assumed included as paths in the reduced integrated directed graph and gray channel directed graph.

6.2.1 Integrated directed graph construction

The integrated directed graph $G_I = (V_I, E_I)$ is constructed upon the extracted text components in the red, green, and blue channels. The assigned text line direction is from left to right according to the text character space relationship. Firstly, the gray channel directed graph is constructed and the text lines are extracted based on the k-shortest paths optimization method as described in Chapter 4. Secondly, we construct the integrated directed graph upon the text characters of the red, green, and blue channels. Then, the integrated directed graph is reduced and transformed based on the gray

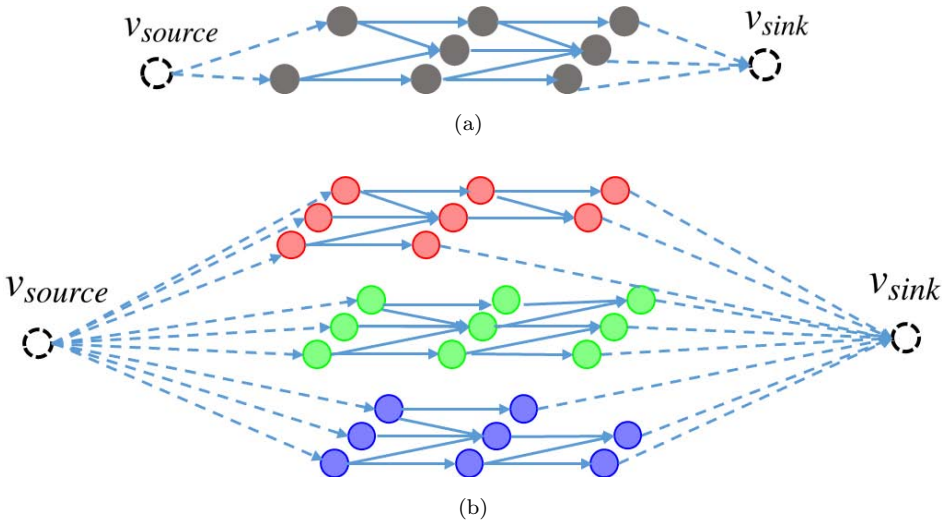


FIGURE 6.3: The proposed text line extraction involves one gray-channel directed graph and one integrated directed graph. (a) provides one example of the gray-channel directed graph. (b) illustrates the integrated directed graph.

channel text lines verification. Algorithm 5 provides the directed graph construction details.

Before the integrated directed graph reduction and transformation, the integrated directed graph is constructed upon text characters in red, green, and blue channels. The vertices of directed graph $\{V_I | I = \{red,\ green,\ blue\}\}$ are comprised of location of text components at the target red, green, and blue channels. The directed edges are built between any two linked vertices for each channel in the integrated directed graph. As text cannot exchange channels and be shared by two channels, no directed edges exist between the channels, and the integrated directed graph is made of disconnected channels as shown in Figure 6.3.

The directed edge linking method is the same as the multi-channel directed edges presented in Chapter 5. For each vertex v_i, we firstly check any other vertex v_j whose x-coordinate x_j is larger ($x_j > x_i$) than the x-coordinate x_i of vertex v_i. Then, the horizontal overlapping constraint shown in Figure 3.7 is certified to build the candidate vertex pair $p_{i,j}$ from vertex v_i to v_j. Finally, we find the nearest text component vertex

v_j from the candidate pair set $\{p_{i,j}\}$ to build the directed edge $e_{i,j}$ from current vertex v_i to selected vertex v_j. To allow each text component to be the start or end location of the text line, two additional virtual vertices v_{source} and v_{sink} are also added to the integrated directed graph, where v_{source} and v_{sink} denote the possible source and sink point, respectively. Meanwhile, we build the directed edges $e_{source,i}$ from the virtual vertex v_{source} to each vertex v_i and $e_{i,sink}$ from each vertex v_i to the sink vertex v_{sink}. To prevent overloading, the virtual source and sink vertices are connected to each vertex in every channel, as illustrated in Figure 6.3.

6.2.2 Directed graph transformation

In the integrated directed graph, the computational complexity can be significantly reduced by grouping unambiguously text line vertices into tracklet. Firstly, the consistent text paths are generated by the k-shortest paths global optimization algorithm on the single gray channel commodity. We then reduce the integrated directed graph on the red, green, and blue channel commodities. The key idea is to group the unambiguous nodes into tracklets according to the extracted text lines in the gray channel. The transformed directed graph is treated as a reduced directed graph with tracklets vertices and individual vertices. Meanwhile, the cost value of directed edges in the reduced graph is calculated based on previous edges. Finally, we apply the k-shortest paths optimization algorithm on the reduced integrated directed graph to extract text line paths in the red, green, and blue channels.

We assume that a text line path $p_1 = v_1, v_2, v_3, v_4, v_5, v_6, v_7$ (Figure 6.4(a)) is extracted in gray channel by the k-shortest paths optimization algorithm. Then, we track the text line path in the integrated directed graph. We suppose that a directed path segment $p_2 = v_8, v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_9$ exists in one channel of the directed graph and part of

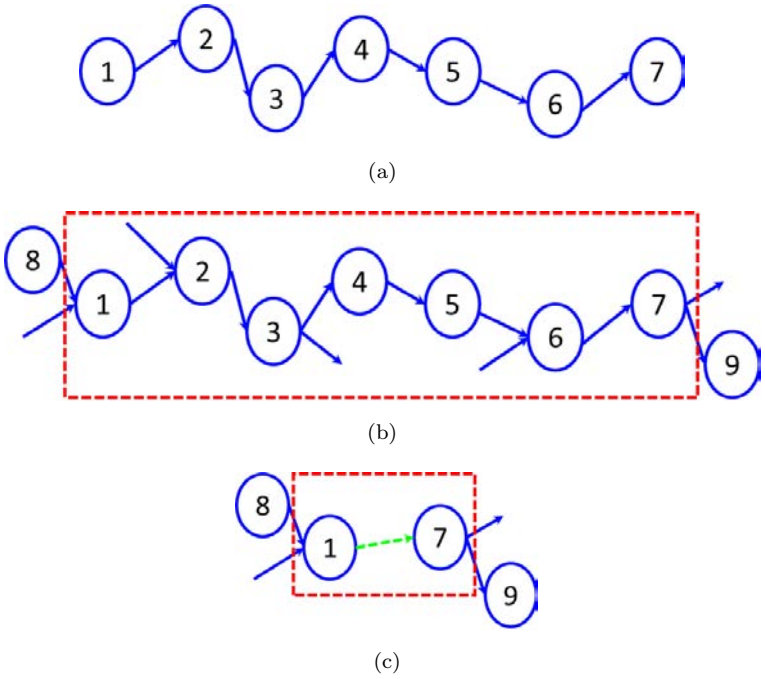


FIGURE 6.4: The directed graph transformation. We suppose the guiding text line is extracted in gray channel and presented in (a). (b) presents the unambiguous path segment tracked and found with a dashed bounding box. (c) illustrates the tracklet with a dashed bounding box.

path segment (dashed bounding box in Figure 6.4(b)) is the same as text lines p_1 in the gray channel. In the transformation, we group the repeating path segment into tracklet $p_1^* = v_1, v_7$ (dashed bounding box in Figure 6.4(c)). The reduced directed path segment $p_2^* = v_8, v_1, v_7, v_9$ can be obtained by the transformation.

In the consistent text line paths generation step, we can obtain text line paths $p_1, p_2, ..., p_k$ in the gray channel, which are set as the guiding text lines. We track each guiding text line in each layer of the integrated directed graph to find the duplicate text line segments. Furthermore, the guiding text lines can be split into connected text line segments to match the duplicate text line segments in the integrated directed graph.

We track the gray channel text lines in the integrated directed graph, and the tracked text path segments are defined as unambiguous text line tracklets due to exact verification in various channels. For each unambiguous text line tracklet, we disconnect all

the directed edges between the first and last vertices, and build a virtual directed edge $e_{tracklet}$ from the first to the last vertex. The cost value of the directed edge $c_{e_{tracklet}}$ can be calculated as follows by the cost of all edges E from the first to the last vertex in the integrated directed graph:

$$c(e_{tracklet}) = \sum_{e_{i,j} \in E} c(e_{i,j}). \tag{6.1}$$

Algorithm 5 Integrated directed graph construction.

Input:

The set of extracted MSER vertices in gray, red, green, and blue channels: $V = \{v_c^i\}_{i=1}^{n_c}$.

Output:

Constructed integrated directed graph, $G_I^* = (V_I^*, E_I^*)$.

1: Build an directed graph for gray channel, described as Algorithm 2;
2: Extract n guiding text lines $P = \{p_1, p_2, ..., p_n\}$ by using the k-shortest paths optimization, described as Algorithm 3;
3: Build the integrated graph $G_I = (V_I, E_I)$ for red, green, and blue channels, descried as Algorithm 4;
4: **for** $i = 1$ to n **do**
5: Track the text line path p_i in $G_I = (V_I, E_I)$ and find out the matched text path segment p_s;
6: Disconnect all the directed edges between the first and last vertices in p_s;
7: Build one virtual directed edge $e_{tracklet}$ from the first to the last vertex;
8: Calculate the cost value of directed edge $c(e_{tracklet})$;
9: **end for**
10: **return** $G_I^* = (V_I^*, E_I^*)$;

6.2.3 K-shortest paths optimization in reduced directed graph

Similar to the multi-channel text line extraction, the k-shortest paths global optimization method is applied a 0-1 constraint in the reduced directed graph. Note that the tracklet

segments in the reduced directed graph should be recovered with original text character vertices for text line paths. The cost function is minimized to find the optimal solution in the directed graph from the virtual source vertex v_{source} to the sink vertex v_{sink} by the following equation. The optimal solution f of the k-shortest paths optimization in the reduced directed graph can be written as follows:

$$f = \underset{G_I^*=(V_I^*,E_I^*)}{\arg\min} \sum_{e_{i,j}\in E_I^*} c(e_{i,j}) \cdot l(e_{i,j}), \tag{6.2}$$

where $c(e_{i,j})$ denotes the cost value of edge $e_{i,j} \in E_I^*$ for vertex $i \in V_I^*$ and vertex $j \in V_I^*$ in the reduced directed graph. The $l(e_{i,j})$ denotes the label of edge $e_{i,j} \in E_I^*$, and the value of $l(e_{i,j})$ is 0 or 1 for optimization.

In the integrated text line extraction, the text line paths in the gray, red, green, and blue channels are produced by the k-shortest global optimization algorithm. The additional channels not only produce the failure text lines in the gray channel but also produce duplicate text lines. We remove the duplicate text lines as noises to avoid repeat extraction. If all of the text characters in the text line are included by other text lines, then we remove the included text line as duplicate noise.

6.3 Experimental Results and Analysis

In this section, we compare the integrated text line extraction with the multi-channel text line extraction and state-of-the-art methods on the ICDAR2011 and ICDAR2013 databases. The experimental results present that the vertices and directed edges can be reduced significantly, and the processing time of the k-shortest paths optimization can be accelerated. These results prove that the complexity of multi-channel text line extraction

can be reduced significantly without sacrificing the performance by the directed graph transformation.

6.3.1 Experiment setup

In the integrated text line extraction, we apply the same experimental settings as those in the multi-channel text line extraction. The settings include the MSER text character extraction, noise text character removal, directed graph construction, and k-shortest paths optimization. Furthermore, we quantify the performance by the same metrics: precision, recall, and f-measure to ensure fair comparison.

6.3.2 Experiment results and analysis

We firstly extract the guiding text lines on gray channel by applying the k-shortest paths optimization algorithm, which is set as the baseline compared with the proposed integrated path flow method. The baseline recall, precision, and f-measure performance are 0.716, 0.902, and 0.798, respectively. As shown in Figure 6.5(a), the text line "DEBENHAMS" and "www.debenhams.com" are extracted successfully by the k-shortest paths optimization method. However, we cannot extract the text line "BRITAIN'S FAVOURITE DEPARTMENT STORE" due to highlight and complex background on the gray channel.

In the integrated directed graph, the guiding text lines "DEBENHAMS" and "www.debenhams.com" are employed to deal with the transformation. The directed graph path p=$v_{\text{“}D\text{”}}$, $v_{\text{“}E\text{”}}$, $v_{\text{“}B\text{”}}$, $v_{\text{“}E\text{”}}$, $v_{\text{“}N\text{”}}$, $v_{\text{“}H\text{”}}$, $v_{\text{“}A\text{”}}$, $v_{\text{“}M\text{”}}$, $v_{\text{“}S\text{”}}$ is transformed into p^*=$v_{\text{“}D\text{”}}$, $v_{\text{“}S\text{”}}$ on all integrated channels. In each channel, the directed graph nodes are reduced from nine to two, and the edges are reduced from eight edges to one edge. Meanwhile,

(a) (b)

FIGURE 6.5: Text line extraction results. (a) shows the gray channel k-shortest paths optimization result and (b) shows the text line extraction results of the integrated k-shortest paths optimization method.

the cost value of directed edge $c(e_{D,S})$ can be calculated by all the cost values of edges from vertex $v_{\text{“}D\text{”}}$ to $v_{\text{“}S\text{”}}$. We can perform a similar transformation for the guiding text line “www.debenhams.com” in the integrated directed graph. Obviously, the integrated directed graph transformation can effectively degrade the computation complexity. We also observed 1.5 times acceleration of processing time in the integrated k-shortest paths optimization compared with separated channels. Finally, we can extract the text lines in the integrated directed graph comprised of different channels, as shown in Figure 6.5(b).

Table 6.1 presents the reduction ratio of the number of vertices and edges between multi-channel directed graph and reduced integrated directed graph on the ICDAR2011 database. In the red channel, the number of vertices and edges can be decreased by 56.7% and 66.0% compared with the multi-channel directed graph, respectively. In the green channel, the number of vertices and edges can be reduced by 62.8% and 72.8% similar to the red channel. In the blue channel, we also observe 56.3% and 65.8% reduction in the number of vertices and edges. Obviously, the multi-channel directed graph complexity can be significantly reduced by grouping the unambiguous text path segments into tracklets in the directed graph transformation.

TABLE 6.1: Vertex and edge comparison between multi-channel directed graph and integrated directed graph on ICDAR2011 database.

Reduced ratio	$Vertex$	$Edge$
$Red_{channel}$	56.7%	66.0%
$Green_{channel}$	62.8%	72.8%
$Blue_{channel}$	56.3%	65.8%

TABLE 6.2: Experimental results of proposed text line extraction method on ICDAR2011 database.

Method	recall	precision	f-measure
[13]	0.581	0.672	0.623
[60]	0.647	0.731	0.687
[56]	0.631	0.833	0.718
[99]	0.750	0.820	0.730
[50]	0.683	0.863	0.762
Baseline	0.716	0.902	0.798
[43]	0.760	0.860	0.800
[51]	0.762	0.862	0.809
Integrated k-shortest paths optimization	0.788	0.866	0.825

In Table 6.2, we illustrate the performance of the integrated k-shortest paths optimization method, which achieves the recall, precision, and f-measure performance of 0.788, 0.866, and 0.825, respectively. Compared with the baseline, the proposed integrated k-shortest paths optimization method can significantly improve the text line extraction performance by incorporating different channels into one framework. Compared with the multi-channel text line extraction, the complexity of directed graph is decreased and the processing time of k-shortest paths optimization are accelerated with no performance loss. We also achieve competitive text line extraction performance in comparison with state-of-the-art methods in terms of recall and f-measure.

Figure 6.6 presents the successful and failure examples of the extracted text lines in the same images of Figure 4.10. The green, blue, and red text line boxes denote the

(a) (b) (c)

(d) (e) (f)

(g) (h) (i)

FIGURE 6.6: Examples of text line extraction results in the integrated directed graph by the k-shortest paths global optimization.

correct, false positive, and failure of text line extraction results, respectively. We can achieve the same results as multi-channel text line extraction by k-shortest path global optimization. Consequently, the directed graph transformation can reduce the graph complexity without scarifying the performance.

6.4 Conclusion

The objective of this chapter is to reduce the complexity of the multi-channel text line extraction method by reducing the multi-channel directed graph. The reduced directed graph can further reduce the complexity of k-shortest paths global optimization. The basic idea is to transform the duplicate text line or segment with numerous vertices and edges into two vertices and one directed edge based on the extracted text line in the gray channel. In our experiments, the comparative results in terms of vertices and edges on various channels demonstrate the significant complexity reduction. We can also expect that the directed graph transformation can successfully reduce the complexity of the k-shortest paths optimization for text line extraction.

Chapter 7

Conclusion

In this thesis, a text line extraction with user-intention and complete text line extraction methods are studied in natural scene images. In the case of text line extraction with user-intention, the character size, skew ratio of text line, and reduction ratio are estimated on the sub-region of the image with the tap and swipe gesture, and then the character components of user-interested text line are accumulated with seed characters. In the case of complete text line extraction, we formulate the complete text line extraction problem as a text line paths global optimization problem. The text line paths global optimization takes advantage of the particular structure of the constructed directed graph upon the extracted MSER text components.

The conventional text line extraction methods have many drawbacks, which limit the application of content-based image understanding. These drawbacks include non-structure construction of the isolated text candidates, lack of robustness with knowledge-based heuristic rules, and exhaustive searching. Consequently, the text line extraction with

user-intention and path optimization are supposed to overcome the aforementioned limitations. The experimental results demonstrate that the text line extraction with user-intention and text line paths optimization method are good solutions for the task of text line extraction.

In Chapter 2, two types of user-intention gestures named swipe and tap are investigated for text line extraction. We make full use of the user interaction information to estimate the character size, skew ratio of text line, and reduction ratio. From the experimental results, we achieve precision performance of 0.81 and 0.84 for text line extraction by the tap and swipe gesture, respectively. Higher precision performance achievement by swipe gesture than tap gesture is reasonable due to more information captured from the user. Obviously, we can successfully extract the user interested text lines through the interaction with the user.

In Chapter 3, the directed graph construction method was introduced and studied based on the MSER-based text components. We filtered out the non-character noises by the well-designed CNN. The text directed graph was constructed according to the spatial arrangement in the text lines, which can build the relationship and eliminate the disorder of isolated MSER text components. The results showed that the directed graph construction was reasonable and successful due to the observation of human reading sense in the assigned direction from left to right. Obviously, the target text line paths were included in the directed graph through the directed graph construction upon text components.

In Chapter 4, the k-shortest paths global optimization was analyzed and evaluated on the ICDAR2011 and ICDAR2013 databases for the text line extraction. The experimental

results presented that the k-shortest paths global optimization method was comparable with state-of-the-art methods. On the ICDAR2011, the proposed k-shortest paths global optimization method achieved precision of 0.716, recall of 0.902, and f-measure of 0.798. Our experimental results approved that we can successfully extract the text line iteratively by applying the k-shortest paths global optimization method. Moreover, the global optimization can determine the number of text lines automatically to avoid exhaustive searching.

In Chapter 5, the k-shortest paths global optimization method was extended to the multi-channel combination for text line extraction. According to the experimental results, the path global optimization of the multi-channel combination achieved precision of 0.781, recall of 0.863, and f-measure of 0.820, respectively. We observed that the recall of 7.2% and f-measure of 2.2% were improved with the complementary usage of red, green, and blue channels in the directed graph and k-shortest paths global optimization. This approved that the multi-channel text line extraction can achieve promising performance, and the performance of text line extraction can be improved significantly by combining the red, green, and blue channels by taking advantage of the structure of the multi-channel directed graph.

In Chapter 6, the efficiency of the multi-channel k-shortest paths global optimization was improved based on the multi-channel directed graph transformation. The experimental results showed that the complexity of the multi-channel directed graph and path optimization can be reduced significantly by grouping the duplicate text path segments into tracklets. The number of green channel vertices and edges were reduced 2.69 and 3.67 times, respectively. Similarly, we also observed the reduction of vertex and edge complexity in the red and blue channel. Furthermore, the processing time of the k-shortest

paths global optimization was accelerated by 1.5 times, which proved the performance of complexity reduction.

As the k-shortest paths global optimization has achieved promising performance, the following improvements and extensions can be made in the future.

- Firstly, it is worth to use different structure of CNN classifier instead of the simple CNN with 3-convolutional layers, which is used for the text character feature extraction and classification of non-character noise removal. Many options are available, such as VGGNet, GoogleNet, and ResNet. These deep neural networks can improve the performance of feature extraction and classification for non-character noises removal in text line extraction. Besides, the CNN can incorporate with the gray-scale or color-scale feature extraction and classification to improve the performance of text line extraction.

- Secondly, we can try using a more effective text component detector to improve the performance of k-shortest paths global optimization. In the thesis, the MSER-based candidate text character extraction method was utilized for directed graph construction. The main bottleneck of the proposed method is the limited accuracy performance of text character extraction. Consequently, we need to find a more accurate text character detector for k-shortest paths global optimization. With the development of deep learning technology, We can apply deep learning based text character detection, such as fast region-based convolutional neural network, single shot multi-box detector, region-based fully convolutional neural networks. As expected, the recall performance of text line extraction can be improved a lot considerably.

- Thirdly, in the future, we can apply the deep learning method in the false text line removal to improve the precision. In the thesis, we extract the text lines by the k-shortest paths global optimization upon the text characters in the directed graph. The single false positive sample may be similar to the true text character, and the false text line may be generated by the noise candidates with the same similarity of text line. However, the texture of noise text line can make a significant to the ground-truth text lines. Thus, we can discriminate the noise text lines from the truth text lines by applying the deep learning technology. As expected, the precision performance of text line extraction can be improved significantly.

Bibliography

[1] CP Sumathi, T Santhanam, and G Gayathri Devi. A survey on various approaches of text extraction in images. *Computer Science and Engineering Survey*, 3(4):27–42, 2012.

[2] Kumary R Soumya, Ancy.T.V, and Athulya Chacko. Text extraction from images: a survey. *Advances in Computer Science and Technology*, 3(2):100–104, 2014.

[3] Jiang Gao and Jie Yang. An adaptive algorithm for text detection from natural scenes. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2, pages 84–89, 2001.

[4] Jian Fan, Xiaofan Lin, and Steven Simske. A comprehensive image processing suite for book re-mastering. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 447–451, 2005.

[5] Keechul Jung, Kwang In Kim, and Anil K. Jain. Text information extraction in images and video: a survey. *Pattern Recognition*, 37(5):977–997, 2004.

[6] Line Eikvil. Optical character recognition. *http://citeseer.ist.psu.edu/142042.html*, 1993.

[7] Noman Islam, Zeeshan Islam, and Nazia Noor. A survey on optical character recognition system. *arXiv preprint arXiv:1710.05703*, 2017.

[8] Nobuo Ezaki, Marius Bulacu, and Lambert Schomaker. Text detection from natural scene images: towards a system for visually impaired persons. In *Proceedings of the International Conference on Pattern Recognition*, volume 2, pages 683–686, 2004.

[9] Yangxing Liu, Satoshi Goto, and Takeshi Ikenaga. A contour-based robust algorithm for text detection in color images. *IEICE Transactions on Information and Systems*, 89(3):1221–1230, 2006.

[10] Xiangrong Chen and Alan L Yuille. Detecting and reading text in natural scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 366–373, 2004.

[11] Jing Zhang and Rangachar Kasturi. Text detection using edge gradient and graph spectrum. In *Proceedings of the International Conference on Pattern Recognition*, pages 3979–3982, 2010.

[12] Huizhong Chen, Sam S Tsai, Georg Schroth, David M Chen, Radek Grzeszczuk, and Bernd Girod. Robust text detection in natural images with edge-enhanced maximally stable extremal regions. In *Proceedings of the International Conference on Image Processing*, pages 2609–2612, 2011.

[13] Boris Epshtein, Eyal Ofek, and Yonatan Wexler. Detecting text in natural scenes with stroke width transform. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2963–2970, 2010.

[14] Victor Wu, Raghavan Manmatha, and Edward M. Riseman. Textfinder: An automatic system to detect and recognize text in images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(11):1224–1229, 1999.

[15] Yao Li and Huchuan Lu. Scene text detection via stroke width. In *Proceedings of the International Conference on Pattern Recognition*, pages 681–684, 2012.

[16] Kwang In Kim, Keechul Jung, and Jin Hyung Kim. Texture-based approach for text detection in images using support vector machines and continuously adaptive mean shift algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(12):1631–1639, 2003.

[17] Michael R Lyu, Jiqiang Song, and Min Cai. A comprehensive method for multilingual video text detection, localization, and extraction. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(2):243–255, 2005.

[18] CS Shin, KI Kim, MH Park, and Hang Joon Kim. Support vector machine-based text detection in digital video. In *Proceedings of the IEEE Signal Processing Society Workshop on Neural networks for signal processing*, volume 2, pages 634–641, 2000.

[19] Jiri Matas, Ondrej Chum, Martin Urban, and Tomás Pajdla. Robust wide-baseline stereo from maximally stable extremal regions. *Image and vision computing*, 22(10):761–767, 2004.

[20] Jaakko Sauvola and Matti Pietikäinen. Adaptive document image binarization. *Pattern Recognition*, 33(2):225–236, 2000.

[21] Max Jaderberg, Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Reading text in the wild with convolutional neural networks. *Computer Vision*, 116(1):1–20, 2016.

[22] Jing Zhang and Rangachar Kasturi. A novel text detection system based on character and link energies. *IEEE Transactions on Image Processing*, 23(9):4187–4198, 2014.

[23] Palaiahnakote Shivakumara, Trung Quy Phan, Shijian Lu, and Chew Lim Tan. Gradient vector flow and grouping-based method for arbitrarily oriented scene text detection in video images. *IEEE Transactions on Circuits and Systems for Video Technology*, 23(10):1729–1739, 2013.

[24] Yi-Feng Pan, Xinwen Hou, and Cheng-Lin Liu. Text localization in natural scene images based on conditional random field. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 6–10, 2009.

[25] Huiying Shen and James Coughlan. Grouping using factor graphs: an approach for finding text with a camera phone. In *Proceedings of the International Workshop on Graph-Based Representations in Pattern Recognition*, pages 394–403, 2007.

[26] Xuwang Yin, Xu-Cheng Yin, Hong-Wei Hao, and Khalid Iqbal. Effective text localization in natural scene images with mser, geometry-based grouping and adaboost. In *Proceedings of the International Conference on Pattern Recognition*, pages 725–728, 2012.

[27] Q. Ye and D. Doermann. Text detection and recognition in imagery: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(7):1480–1500, 2015.

[28] Qixiang Ye, Jianbin Jiao, Jun Huang, and Hua Yu. Text detection and restoration in natural scene images. *Visual Communication and Image Representation*, 18(6):504–513, 2007.

[29] Shehzad Muhammad Hanif and Lionel Prevost. Text detection and localization in complex scene images using constrained adaboost algorithm. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 1–5, 2009.

[30] Rodrigo Minetto, Nicolas Thome, Matthieu Cord, Jonathan Fabrizio, and Beatriz Marcotegui. Snoopertext: A multiresolution system for text detection in complex visual scenes. In *Proceedings of the International Conference on Image Processing*, pages 3861–3864, 2010.

[31] Huiping Li, David Doermann, and Omid Kia. Automatic text detection and tracking in digital video. *IEEE Transactions on Image Processing*, 9(1):147–156, 2000.

[32] Yu Zhong, Hongjiang Zhang, and Anil K Jain. Automatic caption localization in compressed video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(4):385–392, 2000.

[33] Julinda Gllavata, Ralph Ewerth, and Bernd Freisleben. Text detection in images based on unsupervised classification of high-frequency wavelet coefficients. In *Proceedings of the International Conference on Pattern Recognition*, volume 1, pages 425–428, 2004.

[34] Qixiang Ye, Qingming Huang, Wen Gao, and Debin Zhao. Fast and robust text detection in images and video frames. *Image and Vision Computing*, 23(6):565–576, 2005.

[35] Yi-Feng Pan, Cheng-Lin Liu, and Xinwen Hou. Fast scene text localization by learning-based filtering and verification. In *Proceedings of the International Conference on Image Processing*, pages 2269–2272, 2010.

[36] Palaiahnakote Shivakumara, Trung Quy Phan, and Chew Lim Tan. New fourier-statistical features in rgb space for video text detection. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(11):1520–1532, 2010.

[37] Wonjun Kim and Changick Kim. A new approach for overlay text detection and extraction from complex video scene. *IEEE transactions on Image Processing*, 18(2):401–411, 2009.

[38] Hideaki Goto and Makoto Tanaka. Text-tracking wearable camera system for the blind. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 141–145, 2009.

[39] Qixiang Ye and David Doermann. Scene text detection via integrated discrimination of component appearance and consensus. In *Proceedings of the International Workshop on Camera-Based Document Analysis and Recognition*, pages 47–59, 2013.

[40] Lluis Gomez and Dimosthenis Karatzas. Multi-script text extraction from natural scenes. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 467–471, 2013.

[41] Myung-Chul Sung, Bongjin Jun, Hojin Cho, and Daijin Kim. Scene text detection with robust character candidate extraction method. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 426–430, 2015.

[42] Anhar Risnumawan, Palaiahankote Shivakumara, Chee Seng Chan, and Chew Lim Tan. A robust arbitrary text detection system for natural scene images. *Expert Systems with Applications*, 41(18):8027–8048, 2014.

[43] Zheng Zhang, Wei Shen, Cong Yao, and Xiang Bai. Symmetry-based text line detection in natural scenes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2558–2567, 2015.

[44] Weilin Huang, Yu Qiao, and Xiaoou Tang. Robust scene text detection with convolution neural network induced mser trees. In *Proceedings of the European Conference on Computer Vision*, pages 497–511, 2014.

[45] Jung-Jin Lee, Pyoung-Hean Lee, Seong-Whan Lee, Alan Yuille, and Christof Koch. Adaboost for text detection in natural scene. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 429–434, 2011.

[46] Kai Wang, Boris Babenko, and Serge Belongie. End-to-end scene text recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1457–1464, 2011.

[47] Adam Coates, Blake Carpenter, Carl Case, Sanjeev Satheesh, Bipin Suresh, Tao Wang, David J Wu, and Andrew Y Ng. Text detection and character recognition in scene images with unsupervised feature learning. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 440–445, 2011.

[48] Tao Wang, David J Wu, Adam Coates, and Andrew Y Ng. End-to-end text recognition with convolutional neural networks. In *Proceedings of the International Conference on Pattern Recognition*, pages 3304–3308, 2012.

[49] Liuan Wang, Wei Fan, Jun Sun, Satshi Naoi, and Tanaka Hiroshi. Text line extraction in document images. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 191–195, 2015.

[50] Xu-Cheng Yin, Xuwang Yin, Kaizhu Huang, and Hong-Wei Hao. Robust text detection in natural scene images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(5):970–983, 2014.

[51] Shangxuan Tian, Yifeng Pan, Chang Huang, Shijian Lu, Kai Yu, and Chew Lim Tan. Text flow: A unified text detection system in natural scene images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4651–4659, 2015.

[52] Xiaobing Wang, Yonghong Song, and Yuanlin Zhang. Natural scene text detection with multi-channel connected component segmentation. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 1375–1379, 2013.

[53] Nikos Liolios, Nikos Fakotakis, and G Kokkinakis. Improved document skew detection based on text line connected-component clustering. In *Proceedings of the International Conference on Image Processing*, volume 1, pages 1098–1101, 2001.

[54] Lloyd A. Fletcher and Rangachar Kasturi. A robust algorithm for text string separation from mixed text/graphics images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(6):910–918, 1988.

[55] Shang-Hsuan Tsai, Vasudev Parameswaran, and Radek Grzeszczuk. Text detection using multi-layer connected components with histograms, December 17 2013. US Patent 8,611,662.

[56] Cunzhao Shi, Chunheng Wang, Baihua Xiao, Yang Zhang, and Song Gao. Scene text detection using graph model built upon maximally stable extremal regions. *Pattern Recognition Letters*, 34(2):107–116, 2013.

[57] SeongHun Lee, Min Su Cho, Kyomin Jung, and Jin Hyung Kim. Scene text extraction with edge constraint and text collinearity. In *Proceedings of the International Conference on Pattern Recognition*, pages 3983–3986, 2010.

[58] Yi-Feng Pan, Xinwen Hou, and Cheng-Lin Liu. A hybrid approach to detect and localize texts in natural scene images. *IEEE Transactions on Image Processing*, 20(3):800–813, 2011.

[59] Hyung Il Koo and Duck Hoon Kim. Scene text detection via connected component clustering and nontext filtering. *IEEE Transactions on Image Processing*, 22(6):2296–2305, 2013.

[60] Lukáš Neumann and Jiří Matas. Real-time scene text localization and recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3538–3545, 2012.

[61] Yao Li, Chunhua Shen, Wenjing Jia, and Anton Van Den Hengel. Leveraging surrounding context for scene text detection. In *Proceedings of the International Conference on Image Processing*, pages 2264–2268, 2013.

[62] Jun Du, Qiang Huo, Lei Sun, and Jian Sun. Snap and translate using windows phone. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 809–813, 2011.

[63] Liuan Wang, Yutaka Katsuyama, Wei Fan, Yuan He, Jun Sun, and Yoshinobu Hotta. Text detection in natural scene images with user-intention. In *Proceedings of the International Conference on Image Processing*, pages 2256–2259, 2013.

[64] Artur Ferreira. Survey on boosting algorithms for supervised and semi-supervised learning. *Institute of telecommunications*, 2007.

[65] Simon M Lucas, Alex Panaretos, Luis Sosa, Anthony Tang, Shirley Wong, Robert Young, Kazuki Ashida, Hiroki Nagai, Masayuki Okamoto, Hiroaki Yamamoto, et al. Icdar 2003 robust reading competitions: entries, results, and future directions. *International Journal of Document Analysis and Recognition*, 7(2-3):105–122, 2005.

[66] Georgios Louloudis, Basilios Gatos, Ioannis Pratikakis, and Constantin Halatsis. Text line and word segmentation of handwritten documents. *Pattern Recognition*, 42(12):3169–3183, 2009.

[67] Lei Sun and Qiang Huo. A component-tree based method for user-intention guided text extraction. In *Proceedings of the International Conference on Pattern Recognition*, pages 633–636, 2012.

[68] Krystian Mikolajczyk, Tinne Tuytelaars, Cordelia Schmid, Andrew Zisserman, Jiri Matas, Frederik Schaffalitzky, Timor Kadir, and Luc J. Van Gool. A comparison of affine region detectors. *Computer Vision*, 65(1-2):43–72, 2005.

[69] Weilin Huang, Yu Qiao, and Xiaoou Tang. Robust scene text detection with convolution neural network induced mser trees. In *Proceedings of the European Conference on Computer Vision*, pages 497–511, 2014.

[70] Llífs Gómez and Dimosthenis Karatzas. Mser-based real-time text detection and tracking. In *Proceedings of the International Conference on Pattern Recognition*, pages 3110–3115, 2014.

[71] Asif Shahab, Faisal Shafait, and Andreas Dengel. Icdar 2011 robust reading competition challenge 2: Reading text in scene images. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 1491–1496, 2011.

[72] Dimosthenis Karatzas, Faisal Shafait, Seiichi Uchida, Masakazu Iwamura, L-luis Gomez i Bigorda, Sergi Robles Mestre, Joan Mas, David Fernandez Mota, Jon Almazan Almazan, and Lluis Pere de las Heras. Icdar 2013 robust reading competition. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 1484–1493, 2013.

[73] David Nistér and Henrik Stewénius. Linear time maximally stable extremal region-s. In *Proceedings of the European Conference on Computer Vision*, pages 183–196, 2008.

[74] Yann LeCun, Koray Kavukcuoglu, Clément Farabet, et al. Convolutional networks and applications in vision. In *Proceedings of the International Symposium on Circuits and Systems*, pages 253–256, 2010.

[75] Li Chen, Song Wang, Wei Fan, Jun Sun, and Satoshi Naoi. Beyond human recognition: A cnn-based framework for handwritten character recognition. In *Proceedings of the Asian Conference on Pattern Recognition*, pages 695–699, 2015.

[76] Xuefeng Xiao, Yafeng Yang, Tasweer Ahmad, Lianwen Jin, and Tianhai Chang. Design of a very compact cnn classifier for online handwritten chinese character cecognition using dropweight and global pooling. *arXiv preprint arXiv:1705.05207*, 2017.

[77] Meijun He, Shuye Zhang, Huiyun Mao, and Lianwen Jin. Recognition confidence analysis of handwritten chinese character with cnn. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 61–65, 2015.

[78] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, et al. Deep face recognition. In *Proceedings of the British Machine Vision Conference*, volume 1, page 6, 2015.

[79] Shreyas Saxena and Jakob Verbeek. Heterogeneous face recognition with cnns. In *Proceedings of the European Conference on Computer Vision*, pages 483–491, 2016.

[80] Wael AbdAlmageed, Yue Wu, Stephen Rawls, Shai Harel, Tal Hassner, Iacopo Masi, Jongmoo Choi, Jatuporn Lekust, Jungyeon Kim, Prem Natarajan, et al. Face recognition using deep multi-pose representations. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, pages 1–9, 2016.

[81] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on Audio, Speech, and Language processing*, 22(10):1533–1545, 2014.

[82] Dimitri Palaz, Ronan Collobert, et al. Analysis of cnn-based speech recognition system using raw speech as input. In *Proceedings of the INTERSPEECH*, number EPFL-CONF-210029, 2015.

[83] Dimitri Palaz, Mathew Magimai Doss, and Ronan Collobert. Convolutional neural networks-based continuous speech recognition using raw speech signal. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 4295–4299, 2015.

[84] Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. Comparative study of cnn and rnn for natural language processing. *arXiv preprint arXiv:1702.01923*, 2017.

[85] Kaisheng Yao, Baolin Peng, Yu Zhang, Dong Yu, Geoffrey Zweig, and Yangyang Shi. Spoken language understanding using long short-term memory neural networks. In *Proceedings of the Workshop on Spoken Language Technology*, pages 189–194, 2014.

[86] Dinghan Shen, Martin Renqiang Min, Yitong Li, and Lawrence Carin. Adaptive convolutional filter generation for natural language understanding. *arXiv preprint arXiv:1709.08294*, 2017.

[87] T. E. de Campos, B. R. Babu, M. Varma, and Manik Varma. Character recognition in natural images. In *Proceedings of the International Conference on Computer Vision Theory and Applications*, 2009.

[88] Boran Yu and Hongjie Wan. Chinese txt detection and recognition in natural scene using hog and svm. *DEStech Transactions on Computer Science and Engineering*, 2016.

[89] Kai Wang and Serge Belongie. Word spotting in the wild. In *Proceedings of the European Conference on Computer Vision*, pages 591–604, 2010.

[90] Qiu-Feng Wang, Erik Cambria, Cheng-Lin Liu, and Amir Hussain. Common sense knowledge for handwritten chinese text recognition. *Cognitive Computation*, 5(2):234–242, 2013.

[91] Yeganeh M Marghi, Farzad Towhidkhah, and Shahriar Gharibzadeh. Human brain function in path planning: a task study. *Cognitive Computation*, pages 1–14, 2016.

[92] Liang Yang, Hongfei Lin, Yuan Lin, and Shengbo Liu. Detection and extraction of hot topics on chinese microblogs. *Cognitive Computation*, 8(4):577–586, 2016.

[93] Hao Jiang, Sidney Fels, and James Little. A linear programming approach for multiple object tracking. In *Proceedings of the International Conference on Computer and Information Science*, pages 744–750, 2007.

[94] Lester Randolph Ford Jr and Delbert Ray Fulkerson. *Flows in networks*. 2015.

[95] Richard Bellman. On a routing problem. *Quarterly of applied mathematics*, pages 87–90, 1958.

[96] Jerome Berclaz, Francois Fleuret, Engin Turetken, and Pascal Fua. Multiple object tracking using k-shortest paths optimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(9):1806–1819, 2011.

[97] Simon M Lucas. Icdar 2005 text locating competition results. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 80–84, 2005.

[98] Chucai Yi and Yingli Tian. Text detection in natural scene images by stroke gabor words. In *Proceedings of the International Conference on Document Analysis and Recognition*, pages 177–181, 2011.

[99] Weilin Huang, Zhe Lin, Jianchao Yang, and Jue Wang. Text localization in natural images using stroke feature transform and text covariance descriptors. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1241–1248, 2013.

[100] Xu-Cheng Yin, Wei-Yi Pei, Jun Zhang, and Hong-Wei Hao. Multi-orientation scene text detection with adaptive clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1930–1937, 2015.

[101] Juhua Liu, Hai Su, Yaohua Yi, and Wenbin Hu. Robust text detection via multi-degree of sharpening and blurring. *Signal Processing*, 124:259–265, 2016.

[102] Anna Zhu, Renwu Gao, and Seiichi Uchida. Could scene context be beneficial for scene text detection? *Pattern Recognition*, 58:204–215, 2016.