# EXTRACTION OF TEXT

# FROM IMAGES AND VIDEOS

**PHAN QUY TRUNG**

(B. Comp. (Hons.), National University of Singapore)

**A THESIS SUBMITTED**

**FOR THE DEGREE OF DOCTOR OF PHILOSOPHY**

**DEPARTMENT OF COMPUTER SCIENCE**

**NATIONAL UNIVERSITY OF SINGAPORE**

**2014**

# Declaration

I hereby declare that this thesis is my original work and it has been written by me in its entirety. I have duly acknowledged all the sources of information which have been used in the thesis.

This thesis has also not been submitted for any degree in any university previously.

Phan Quy Trung

10 April 2014

*To my parents and my sister*

# Acknowledgements

I would like to express my sincere gratitude to my advisor Prof. Tan Chew Lim for his guidance and support throughout my candidature. With his vast knowledge and experience in research, he has given me advice on a wide range of issues, including the directions of my thesis and the best practices for conference and journal submissions. Most importantly, Prof. Tan believed in me, even when I was unsure of myself. His constant motivation and encouragement have helped me to overcome the difficulties during my candidature.

I would also like to thank my colleague and co-author Dr. Palaiahnakote Shivakumara for the many discussions and constructive comments on the works in this thesis.

I thank my labmates in CHIME lab for their friendship and help in both academic and non-academic aspects: Su Bolan, Tian Shangxuan, Sun Jun, Mitra Mohtarami, Chen Qi, Zhang Xi and Tran Thanh Phu. I am particularly thankful to Bolan and Shangxuan for their collaboration on some of the works in this thesis.

My thanks also go to my friends for their academic and moral support: Le Quang Loc, Hoang Huu Hung, Le Thuy Ngoc, Nguyen Bao Minh, Hoang Trong Nghia, Le Duy Khanh, Le Ton Chanh and Huynh Chau Trung. Loc and Hung have, in particular, helped me to proofread several of the works in this thesis.

Lastly, I thank my parents and my sister for their love and constant support in all my pursuits.

# Table of Contents

# Summary

With the rapid growth of the Internet, the amount of image and video data is increasing exponentially. In some image categories (e.g., natural scenes) and video categories (e.g., news, documentaries, commercials and movies), there is often text information. This information can be used as a semantic feature, in addition to visual features such as colors and shapes, to improve the retrieval of the relevant images and videos.

This thesis addresses the problem of text extraction in natural scene images and in videos, which typically consists of text localization, tracking, enhancement, binarization and recognition.

Text localization, i.e., identifying the positions of the text lines in an image or video, is the first and one of the most important components in a text extraction system. We have developed two works, one for text in natural scene images and the other for text in videos. The first work introduces novel gap features to localize difficult cases of scene text. The use of gap features is new because most existing methods extract features from only the characters, and not from the gaps between them. The second work employs skeletonization to localize multi-oriented video text. This is an improvement over previous methods which typically localize only horizontal text.

After the text lines have been localized, they need to be enhanced in terms of contrast so that they can be recognized by an Optical Character Recognition (OCR) engine. We have proposed two works, one for single-frame enhancement and the other for multiple-frame enhancement. The main idea of the first work is to segment a text line into individual characters and

binarize each of them individually to better adapt to the local background. Our character segmentation technique based on Gradient Vector Flow is capable of producing curved segmentation paths. In contrast, many previous techniques allow only vertical cuts. In the second work, we exploit the temporal redundancy of video text to improve the recognition accuracy. We develop a tracking technique to identify the framespan of a text object, and for all the text instances within the framespan, we devise a scheme to integrate them into a text probability map.

The two text enhancement works above use an OCR engine for recognition. To obtain better recognition accuracy, we have also explored another approach in which we build our own algorithms for character recognition and word recognition, recognition i.e., without OCR. In addition, we focus on perspective scene text recognition, which is an issue of practical importance but has been neglected by most previous methods. By using features which are robust to rotation and viewpoint change, our work requires only frontal character samples for training, thereby avoiding the labor-intensive process of collecting perspective character samples.

Overall, this thesis describes novel methods for text localization, text enhancement and text recognition in natural scene images and videos. Experimental results show that the proposed methods compare favourably to the state-of-the-art on several public datasets.

# List of Tables

# List of Figures

# List of Abbreviations

## Chapter 1

# Introduction

With the rapid growth of the Internet, more image and video databases are available online. In such databases, there is a need for search and retrieval of images and videos. As most search engines are still text-based, manual keyword annotations have traditionally been used. However, this process is laborious and inconsistent, i.e., two users may choose different keywords for the same image or video. An alternative approach is to generate the keywords from the text that appears in an image (e.g., road signs and bill boards) or a video (e.g., captions). These keywords can then be used as semantic features (in addition to visual features such as colors and shapes) to improve the retrieval of the relevant images and videos. Other general applications include sign translation, intelligent driving assistance, navigation aid for the visually-impaired and robots, video summarization, and video skimming. Domain-specific applications are also possible, e.g., aligning segments of lecture videos with the corresponding external slides. Therefore, there is an increasing demand for text extraction in images and videos.

Although many methods have been proposed over the past years, text extraction is still a challenging problem because of the almost unconstrained text appearances, i.e., texts can vary drastically in fonts, colors, sizes and alignments. Moreover, videos are typically of low resolutions, while natural scene images are often affected by deformations such as perspective distortion, blurring and uneven illumination.

In this thesis, we address the problem of text extraction in images and videos. We formally define the problem and the scope of study in the next section.

## 1.1 Problem Description and Scope of Study

Given an image or a video, the goal of text extraction is to locate the text regions in the image or video and recognize them into text strings (so that they can be used for e.g., indexing). Furthermore, if the input is a video, each text string is annotated with the time stamps (or frame numbers) that mark its appearance/disappearance in the video. Its position in each frame is also recorded because a text line may move between the frames.

The scope of this thesis is text extraction in natural scene images (Figure 1.1a) and in videos (e.g., news, documentaries, commercials and movies) (Figure 1.1b).


(a) Natural scene image (b) Video frame

**Figure 1.1. A scene image and a video frame.**

## 1.2 Contributions

This thesis makes the following contributions:

- We present two text localization works, one for scene text and the other for video text (**Chapter 3**). The former proposes using gap features for text localization, which is a novel approach because most existing methods utilize only character features. The latter addresses the problem of multi-oriented text localization, which has been neglected by most previous methods.
- After the text lines are localized, they need to be enhanced prior to recognition. Thus, we propose two text enhancement works, one for single-frame enhancement and the other for multiple-frame enhancement (**Chapter 4**). The first work illustrates the importance of binarizing each character in a text line individually instead of binarizing the whole line. The second work shows that integrating the multiple instances of the same video text leads to significantly better recognition accuracy.
- In addition to using OCR engines for text recognition (in the above two works), we also explore a different approach: recognition without OCR. In particular, we propose a technique for recognizing perspective scene text (**Chapter 5**). This problem is of great practical importance, but has been neglected by most previous methods (which only handle frontal texts). Thus, with this work, we address an important research gap.

# Chapter 2

# Background & Related Work

This chapter provides a brief overview of the challenges of the different types of texts considered in this thesis. We also review existing text extraction methods and identify some of the research gaps that need to be addressed.

## 2.1 Challenges of Different Types of Text

The extraction of text in images has been well-studied by document analysis techniques such as Optical Character Recognition (OCR). However, these techniques are limited to scanned documents. It is evident from Figure 2.1, Figure 1.1 and Figure 2.2 that natural scene images and videos are much more complex and challenging than document images. Hence, traditional document analysis techniques generally do not work well for text in natural scene images and videos. As an illustrative example, if OCR engines are used to recognize text in videos directly, the recognition rate would typically be in the range 0% to 45% (Chen & Odobez 2005). For comparison, the typical OCR accuracy for document images is over 90%.

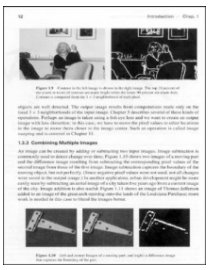
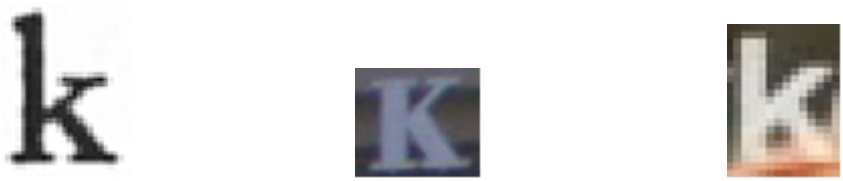


**Figure 2.1. A document image.**



(a) Document character (b) Natural scene character (c) Video character

**Figure 2.2. A document character, a scene character and a video character.**

The major challenges of scene text and video text are listed in Table 2.1. While the majority of the challenges are common to both scene text and video text, some of them are applicable to only one type of text. For example, low resolution is specific to video text, while perspective distortion mainly affects scene text.

Note that Table 2.1 shows the typical challenges for each type of text. In practice, there are exceptions. For example, a video text line with special 3D effects may also be considered as having perspective "distortion".

**Table 2.1. Challenges of text in natural scenes and text in videos.**

| | Text in Natural Scene Images | Text in Videos |
|---|---|---|
| Low resolution | | ✓ |
| Compression artifacts | | ✓ |
| Unconstrained appearances | ✓ | ✓ |
| Complex backgrounds | ✓ | ✓ |
| Varying contrast | ✓ | ✓ |
| Perspective distortion | ✓ | |
| Lighting | ✓ | |
| Domain-independence and multilingualism | ✓ | ✓ |

We will now describe each of the challenges in detail:

- **Low resolution**: For fast streaming on the Internet, videos are often compressed and resized to low resolutions. For comparison, the resolutions of video frames can be as small as 50 dpi (dots per inch) while that of scanned documents is typically much larger,

e.g., from 150 to 400 dpi (Liang et al. 2005). This translates to a typical character height of 10 pixels for the former and 50 pixels for the latter (Li & Doermann 1999). Therefore, traditional OCR engines, which are tuned for scanned documents, do not work well for videos.

- **Compression artifacts**: Since most compression algorithms are designed for general images, i.e., not optimized for text images, they may introduce noise and compression artifacts, and cause blurring and color bleeding in text areas (Liang et al. 2005).
- **Unconstrained appearances**: Texts in different images and videos have drastically different appearances, in terms of fonts, font sizes, colors, positions within the frames, alignments of the characters and so on. The variation comes from not only the text styles but also the contents, i.e., the specific combination of characters that appear in a text line. According to (Chen & Yuille 2004), text has much more variation than other objects, e.g., face. By performing Principle Component Analysis, the authors noticed that text required more than 100 eigenvalues to capture 90% of the variance while face only required around 15 eigenvalues.
- **Complex backgrounds**: While scanned documents contain simple black texts on white backgrounds, natural scenes and videos have much more complex backgrounds, e.g., a street scene or a stadium in a sports news video. Hence, without pre-processing steps such as contrast enhancement and binarization, OCR engines are not able to recognize the characters directly.

- **Varying contrast**: Some text lines may have very low contrast against their local backgrounds (partly due to the compression artifacts and the complex backgrounds mentioned above). It is difficult to detect both high contrast text and low contrast text (sometimes in the same image or video frame), and at the same time, keep the false positive rate low.
- **Perspective distortion**: Because a natural scene image often contains a wide variety of objects, e.g., buildings, cars, trees and people, text may not be the main object in the image. Hence, the text in a natural scene image may not always be frontal and parallel to the image plane. In other words, scene text may be affected by perspective distortion (Jung et al. 2004; Liang et al. 2005; Zhang & Kasturi 2008). Since OCR engines are designed for frontal scanned documents, they cannot handle perspective characters.
- **Lighting**: Natural scene images are captured under varying lighting conditions. Some characters may not receive enough lighting. They appear dark and do not have sufficient contrast to the local background. On the other hand, some characters may be affected by the camera flash. They appear too bright and some of the edges are not visible. These problems make it much more difficult to correctly recognize the characters.
- **Domain-independence and multilingualism**: Although there are some domain-specific text extraction systems (e.g., for sports videos), the majority of the methods in the literature are designed

for general videos, which means that there is no prior information about the text position and appearance. Moreover, the characters of different languages such as English, Chinese and Arabic have different properties. Certain textual features, e.g., contrast with the local background, are observed across different languages while other features, e.g., text stroke statistics, are highly language-dependent (Lyu et al. 2005).

It is worth noting that video text can be further classified into two types: video graphics text and video scene text. The former is artificially added to the video during the editing process, e.g., captions, while the latter appears in the scene captured by the camera (similar to text in natural scene images) (Figure 2.3). In the literature, the term "scene text" is used for both scene text in videos and scene text in still images. To avoid confusion, in this thesis, we will use the various terms with the following meanings:

- *Scene text* refers to text that appears in a still image of a natural scene.
- *Video text* refers to text that appears in a video in general.
- *Video graphics text* refers to text that is artificially added to a video.
- *Video scene text* refers to text that appears as part of a scene in a video.

In general, video scene text is much more challenging (e.g., having lower contrast, more blurring and more complex background) than video graphics text because the former is captured in an uncontrolled environment while the latter is intentionally added for the viewers and thus have better

readability. Moreover, similar to scene text in natural images, video scene text might be affected by perspective distortion and lighting.

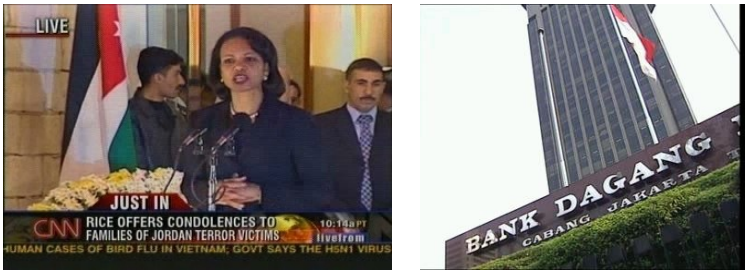



**Figure 2.3. Video graphics text (left) and video scene text (right).**

This section has summarized the challenges of the different types of texts. In the following sections, we review existing text extraction methods for both natural scene images and videos. For the sake of completeness, we will also mention relevant methods for document images.

## 2.2 Text Extraction Pipeline

A text extraction system typically consists of five steps: (1) Localization, (2) Tracking, (3) Enhancement, (4) Binarization and (5) Recognition (Figure 2.4). The first step (Localization) aims to detect and accurately locate all the text lines in an image or a video frame. The second step (Tracking) helps to track the movement of the text lines over multiple frames, e.g., a text line moving from bottom to top in a movie credits scene. In the third step (Enhancement), the localized and tracked text lines are enhanced in terms of contrast and resolution to improve their readability. The fourth step (Binarization) converts the text lines into black and white images so that they can be used in the last step (Recognition), which recognizes the characters by

using either an existing OCR engine or a custom-built OCR engine with its own feature extraction scheme.

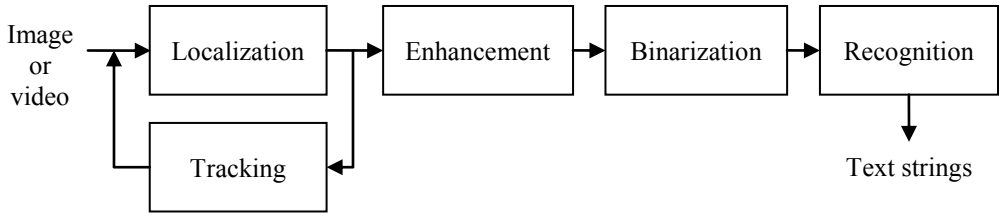



**Figure 2.4. The typical steps of a text extraction system. (Figure adapted from (Jung et al. 2004).)**

Some text extraction systems may slightly change the order of the steps or omit certain steps. For example, Binarization is not needed if the Recognition step can work on grayscale or color images directly. As another example, because temporal information is not available in natural scene images, the Tracking step is omitted for these images.

The next section discusses Localization, the first step in the pipeline.

## 2.3 Text Localization

The goal of text localization is to locate all the text lines in an input image or a video frame. A text line's position is usually represented by a rectangular bounding box (Figure 2.5). Some methods may provide additional information about a localized text line, e.g., a "text mask", which indicates whether a particular pixel in the bounding box is a text pixel or a background pixel. Depending on the application, localization can also be performed at the word level, instead of at the text line level.


**Figure 2.5. The (white) bounding boxes of the localized text lines.**

Text in images often has the following characteristics, which makes it distinguishable from the background:

- Text has sufficient contrast to the local background (to be readable).
- The strokes of a character are in four main directions: horizontal, vertical, left diagonal and right diagonal.
- The pixels of a single character have almost uniform intensity values or colors.
- Characters of the same text line are aligned on a straight line.
- Characters of the same text line have similar widths and heights.
- Characters of the same text line are spaced regularly.

Different methods make use of different properties to localize the text lines. They can be classified into three main approaches: gradient-based, intensity/color-based and texture-based. As its name suggests, the first approach relies on the first two properties of text and often performs edge detection to identify regions in the input image with those properties. Similarly, the second approach analyzes regions in which the pixels have similar intensity values or colors (the third text property). Different from the

previous two approaches, the last approach considers text as a special texture and applies techniques such as Discrete Cosine Transform and wavelet decomposition for feature extraction. For text/non-text classification, this approach typically employs machine learning techniques such as neural networks and Support Vector Machines (SVM).

It is worth mentioning that unlike the first three properties, the last three properties of text are usually used at a later stage in a localization method (rather than as the main feature). For example, these properties can be used to remove false positives.

### 2.3.1 Gradient-based Localization

Gradient-based methods assume that in order for text to be readable, it needs to have enough contrast with the local background. Therefore, these methods look for regions with high intensity variation and/or dense edges. In addition, while most methods make use of "unstructured" edges (e.g., in the form of edge energy or edge density), a few recent works focus on "structured" edges such as strokes (parallel edges) and corners (intersected edges).

(Cai et al. 2002; Lyu et al. 2005) used both global thresholding and adaptive local thresholding of the edge map to suppress edges in complex backgrounds. In addition, two operators were proposed to enhance the remaining edges. The disadvantage of this method is that if the global and local thresholding processes fail to suppress all the non-text edges, the two proposed operators will enhance edges in not only the text regions but also in

the background regions. This will increase the number of false positives in the localization result.

Different from the previous methods which do not use the edge orientation information, (Liu et al. 2005) computed 4 Sobel edge maps for the 4 main directions of text strokes: horizontal, vertical, left diagonal and right diagonal. For each edge map, a sliding window was used to extract 6 statistical features: mean, standard deviation, energy, entropy, inertia, local homogeneity and correlation. K-means was employed to classify pixels into two clusters: text and non-text. This method is good at localizing reasonably high contrast text but may miss low contrast text because the Sobel edge operator mainly detects the strong edges.

Other than edge-related features, the property of high intensity variation in text regions has also been explored for text localization. (Kim & Kim 2009) made an interesting observation that due to color bleeding, there were often "transient" pixels between text and background. These pixels were identified as groups of 3 consecutive pixels that followed an exponential increase/decrease in intensity values (depending on whether text was brighter/darker than the background). Region growing were performed to extend the transient pixels into candidate text regions. This method offers a new perspective into the problem of text localization and handles video graphics text well. However, it can only localize horizontal text and fails to pick up scene text, as shown in the sample results in the paper.

(Wong & Chen 2003) exploited the intensity variation in a different way. The method first computed the horizontal gradients by using the $[-1,\ 1]$ mask. For each $1 \times 21$ region, the maximum gradient difference value was

computed as the difference between the largest and the smallest gradient values. Candidate line segments were found by thresholding the difference map, and were then filtered by using heuristic rules based on the number of transitions between text and background, and the mean and variance of the distances between these transitions. Because this method makes extensive use of heuristic rules and threshold values for analyzing the candidate line segments, it may not generalize well to other datasets. In addition, the simple $[-1,\ 1]$ mask may miss non-horizontal text because it only detects vertical edges.

As mentioned at the beginning of this section, a few recent methods extract features from structured edges, e.g., strokes and corners, instead of from unstructured edges. The former is more robust than the latter due to the additional constraints on the edges. For example, to form a stroke, two edges have to be almost parallel to each other.

(Epshtein et al. 2010) observed that characters in the same word or text line had almost constant stroke widths. The proposed Stroke Width Transform assigned a stroke width value to each pixel in the input image, based on the width of the stroke that it most likely belonged to. For each Canny edge pixel $p$, the method searched for another edge pixel $q$ along the gradient direction at $p$. Ideally, if $p$ and $q$ belonged to the same stroke, the gradient directions at $p$ and $q$ should be exactly opposite of each other. However, to allow for some tolerance, as long as $q$'s gradient direction was roughly opposite that of $p$ (within $\pi$ / 6), all the pixels along the traversed ray were declared to have a stroke width of $\|\overrightarrow{pq}\|$ (Figure 2.6). Pixels with similar strokes widths were merged into candidate text regions.

(Yao et al. 2012) also used Stroke Width Transform to identify the character candidates. However, instead of using heuristic rules for false positive elimination and character linking, the authors designed several character-level and chain-level features and used Random Forest (Breiman 2001) as classifiers.

Stroke Width Transform-based methods are fast and are able to handle multi-oriented text (as long as the characters are aligned on a straight line). However, the accuracy of the Stroke Width Transform is highly dependent on whether the inner and outer contours of a character are almost parallel to each other. For stroke intersections, this condition does not hold, which leads to connected components (CCs) that contain holes or do not preserve the complete shape of a character (Chen et al. 2011). These CCs may be wrongly classified as non-text.

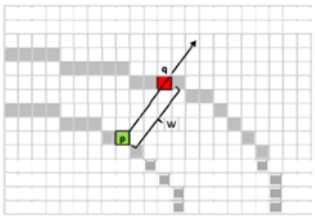



**Figure 2.6. Stroke Width Transform. (Figure adapted from (Epshtein et al. 2010).)**

Another feature that can be derived from text strokes is corner point, i.e., the intersection of two stokes in different directions. This feature can be extracted using operators such as Harris corner detector (Harris & Stephens 1988), Susan corner detector (Smith & Brady 1997) and Shi-Tomasi corner detector (Shi & Tomasi 1994).

(Liu et al. 2010; Liu & Wang 2010) used the Shi-Tomasi detector to look for regions with dense corner points. The input video frame was divided into 64 non-overlapping blocks. Each block was considered as a candidate text block if it contained more than a certain number of corner points. In a similar approach, (Zhao et al. 2011) dilated Harris corner points to form candidate text regions. For text/non-text classification, this method used heuristic rules based on a region's corner point density and shape.

The above three methods were designed for video captions (i.e., graphics text), which have reasonably high contrast with the local background, and thus the corner point feature works well. However, for texts with lower contrast like scene texts, a detector may fail to detect sufficient corner points to classify a region as text region. In addition, the method by (Zhao et al. 2011) does not work for multi-oriented text due to the constraints used for false positive elimination. For example, it was assumed that for a true text region, its width was always greater than its height. Although this assumption is true for horizontal text, it does not hold for multi-oriented text.

In summary, gradient-based methods make the assumption that text has sufficient contrast with the local background and thus find potential text lines in regions with high contrast, high intensity variation and dense edges (either structured or unstructured). These methods are generally fast but can be sensitive to the threshold values used for edge detection. High values may cause low contrast text to be missed, while low values may increase the number of false positives, especially in complex backgrounds.

### 2.3.2 Texture-based Localization

To overcome the problem of complex background of gradient-based methods, the texture-based approach considers text as a special texture. These methods apply techniques such as Discrete Cosine Transform and wavelet decomposition for feature extraction. For text/non-text classification, they often employ machine learning techniques such as neural networks and SVM.

(Kim et al. 2001; Kim et al. 2003; Jung & Han 2004) extracted raw intensity values and used SVM to classify every pixel as text/non-text. For each M × M window, the feature vector of the center pixel was defined as the intensity values of the neighboring pixels according a mask which captured the four main directions of text strokes (Figure 2.7).

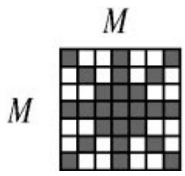



**Figure 2.7. In each window, only the pixels at the positions marked by gray are fed into SVM. (Figure adapted from (Kim et al. 2003).)**

In general, intensity values are not robust against the different text appearances in different input images. Therefore, a number of methods have used gradient information instead. (Lienhart & Wernicke 2002) extracted features from the edge orientation image, which was computed based on the gradient information in all the RGB channels. A neural network classified each 20 × 10 window as text/non-text. The authors noticed that the localization rate decreased significantly for texts of small font sizes, e.g., less than 10 pixels in height.

The gradient information was also employed by (Chen et al. 2001a; Chen et al. 2004a; Chen et al. 2004b). A 16 × 16 sliding window was used to extract the following features from each text candidate region: the x and y derivatives of the intensity values, the distance map (to strong edge points) and the normalized gradient values (such that the local mean became zero and the local variance matched the global variance) (Figure 2.8). In the experiments, the normalized gradient value was found to be better than the other features, because it achieved a degree of invariance to texts of different intensity values and backgrounds.


**Figure 2.8. The various features tested in (Chen et al. 2004b). From top to bottom: candidate text region, x-derivative, y-derivative, distance map and normalized gradient values. (Figure adapted from (Chen et al. 2004b).)**

(Chen & Yuille 2004) also employed many intensity and gradient features. The main difference between this method and the previous methods lies in the use of AdaBoost (Freund & Schapire 1996), which is capable of building a strong classifier out of a set of weak classifiers. To extract the intensity and gradient features, the authors designed several block patterns (Figure 2.9). The aim of these patterns was to average out the variances for regions with large variances, and thus achieve a low entropy, i.e., similar responses for different text appearances in different images. Based on these

block patterns, the means and standard deviations of the intensity values and the x and y intensity derivatives were extracted. One of the contributions of this work is the convincing explanation of the motivation for designing the block patterns. However, these patterns are mainly for horizontal text and thus this method will have difficulties localizing multi-oriented text.

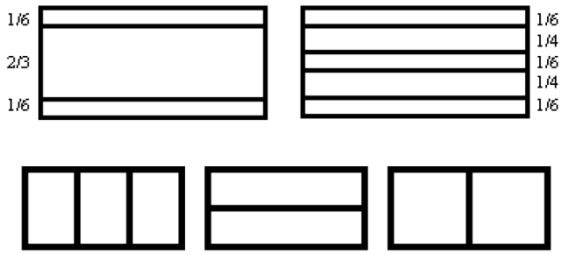



**Figure 2.9. Block patterns. (Figure taken from (Chen & Yuille 2004).)**

Similar to the previous method, (Pan et al. 2008; Pan et al. 2009; Pan et al. 2011) and (Wang et al. 2011) employed a sliding window scheme. However, these methods put more emphasis on the gradient orientations and used Histogram of Oriented Gradients (Dalal & Triggs 2005) as the main feature. The classifiers used were WaldBoost (Sochman & Matas 2005) and Random Ferns (Ozuysal et al. 2007), respectively. Due to the use of sliding window at multiple scales, these methods are computationally expensive.

In addition to gradient features, another way to analyze high contrast pixels and edges is through wavelet decomposition. (Li et al. 2000) extracted Haar wavelet features using an image pyramid. Text regions were expected to have high responses in the high frequency subbands (HL, LH and HH). Therefore, the following features were extracted from each 16 × 16 window in each of the subbands: mean, second-order and third-order central moments.

(Ye et al. 2005) also employed wavelet features. Other than wavelet moments (inspired by the previous method), this work also extracted wavelet energy histogram, wavelet direction histogram, wavelet co-occurrences and crossing count histogram (which captured the periodicity of the peaks in the vertical projection profile).

Compared to other texture analysis approaches, the unique advantage of Discrete Cosine Transform (DCT) is that it is available in the compressed domain, e.g., in JPEG and MPEG formats. Hence, little or no decoding is required. (Zhong et al. 2000) observed that text regions usually had high intensity variation in both the horizontal direction (due to the characters and the spaces between them) and the vertical direction (due to the spaces between the text lines). For each DCT block in an MPEG I-frame, the horizontal energy was calculated by summing the absolute values of the DCT coefficients with zero horizontal frequency (i.e., summing across different vertical frequencies). Candidate text blocks were found by adaptively thresholding the energy map. The main advantage of this method is the computational time. However, the authors mentioned that it has difficulties with texts of large font sizes because the $8 \times 8$ DCT blocks fail to capture the local variations of such large text strokes.

In summary, texture-based methods aim to extract the distinctive features of text from various sources of information: intensity values, gradient magnitudes and orientations, wavelet responses, DCT coefficients and so on. Machine learning techniques are used for text/non-text classification. Texture-based methods are more robust than gradient-based methods against complex backgrounds. They can also be re-trained for different datasets. However, they

have two drawbacks. First, classifiers such as neural networks and SVM require a large training set, sometimes in thousands, of text and non-text samples. Moreover, it is especially hard to ensure that the non-text samples are representative (Kim et al. 2003). Second, most texture-based methods are computationally expensive.

### 2.3.3 Intensity-based and Color-based Localization

The main assumption of intensity-based and color-based methods is that characters in the same "group" have similar intensity values or colors. Different methods make this assumption at different levels: the text line level, the word level or the character level.

(Neumann & Matas 2010; Neumann & Matas 2011; Neumann & Matas 2012) used Maximally Stable Extremal Regions (MSER) (Matas et al. 2002) to extract character candidates. The main idea of MSER is to identify regions which remain stable over a range of thresholds on the intensity values. Many natural scene characters have almost uniform intensity values and thus, they can be extracted as MSERs. MSER-based methods are fast because there are efficient algorithms for MSER extraction. However, the main drawback of these methods is that for images with blurring and uneven illumination, the assumption that the pixels of a scene character have almost uniform intensity values no longer holds. Thus, a single character may be split into several MSERs. In addition, touching characters may also be detected as a single MSER. Both of these problems affect the text localization result.

In general, colors provide more information than intensity values. Designed for high resolution images such as book and journal covers, early

color-based methods, e.g., (Zhong et al. 1995; Jain & Yu 1998; Sobottka et al. 1999), relied purely on color features to localize the text lines. These methods employed color quantization and region growing (or splitting) to group neighboring pixels of similar colors into CCs.

(Mariano & Kasturi 2000) proposed a technique to capture the periodicity of patterns in text regions. Hierarchical color clustering was performed in the L*a*b* color space for every third row in the input image. Each cluster was checked using empirical rules to determine whether they formed the color streaks of a text line. The method then found the text box boundaries for each set of streaks. This method is good at localizing low contrast text. However, the false positive rate reported in the paper was very high (39%).

The drawback of methods that use only color features (such as the above methods) is that the CCs obtained by color similarity may not preserve the complete shapes of the characters due to noise and color bleeding. Therefore, more recent methods often combine colors with other features.

Gradient features and color features were combined in (Chen et al. 2004c). The edges in the input image were obtained by using the Laplacian of Gaussian. CCs were generated by grouping edges based on the similarity in size and intensity values. To model the color distribution of each individual character and its surrounding background, Gaussian Mixture Model was employed to identify two peaks, one corresponding to the foreground and the other corresponding to the background. Line fitting (based on Hough transform) was used to group CCs into words and lines.

(Yi & Tian 2011) proposed and compared two different features for text localization, one based on color and the other based on gradient. For the color feature, the method performed K-means clustering in the RGB space to identify the dominant colors in the input image (Figure 2.10). For the gradient feature, the method identified "pixel couples", pairs of pixels that had similar gradient magnitudes and almost opposite directions. Using the above two features, a set of candidate text CCs were obtained. In the experiments, the color feature outperformed the gradient feature. However, it also required more computational time because each color cluster had to be handled separately.


**Figure 2.10. The left most column shows the input image while the remaining columns show the color clusters identified by K-means. (Figure taken from (Yi & Tian 2011).)**

In summary, intensity-based and color-based methods assume that the characters in the same word/line have similar intensity values or colors. Recent methods tend to use intensity values/colors in hybrid approaches, i.e., together with either gradient features or texture features. This is because in challenging situations, e.g., stylized graphics text and scene text with uneven

illumination, the intensity/color homogeneity assumption only holds at the finest level (the character level). Hence, using intensity/color features alone may not be sufficient for text localization.

### 2.3.4 Summary

In this section, we have reviewed various text localization methods. They can be classified into three approaches: gradient-based, texture-based and intensity/color-based. The first approach locates potential text lines by identifying regions with high contrast variation or dense edges. This approach is fast but may produce many false positives for images with complex backgrounds. To deal with this problem, the second approach extracts textual features using techniques such as wavelet decomposition and Discrete Cosine Transform. Neural networks and SVM are used for text/non-text classification. Although it can be re-trained for different datasets, this approach requires a lot of positive and negative samples for training and is computationally expensive. The last approach is based on the assumption that characters of the same word/line often have similar intensity values or colors. Recent works often combine them with either gradient features or texture features to improve the performance.

Although many methods have been proposed, there are still several research issues that need to be addressed. First, most methods focus solely on features which are extracted from the characters. However, due to the challenges of scene text and video text (e.g., blurring, distortion, low resolution, etc.), the edges and the shapes of the characters may not be reliable (e.g., some edges may be broken into multiple parts or even missing).

Therefore, additional features should be explored to improve the robustness of text localization. In section 3.1, we propose novel inter-character features, which are extracted from the gaps between consecutive characters, and show that these features help to improve the text localization performance.

Second, many methods are designed for only horizontal texts, and are not able to pick up multi-oriented texts. However, in practice, text can appear with any orientation. Hence, to address this issue, our work in section 3.2 employs skeletonization to handle multi-oriented text.

## 2.4 Text Tracking

One of the key differences between videos and the related types of text images is the availability of temporal information. In order to be readable, a text line usually appears on the screen for at least 2 seconds (Wang et al. 2004; Miao et al. 2007). Hence, the temporal redundancy can be used to improve the performance of a video text extraction system.

Text tracking helps to track the movement of a text object over multiple frames. It can be classified into intensity-based tracking, signature-based tracking, probabilistic tracking and tracking in compressed domain. Given a text instance in the current frame, the first two approaches look for potential text instances in the next frame that minimize a similarity score. The third approach employs a probabilistic framework to handle partial occlusion while the fourth and final approach makes use of the information readily available in the compressed domain to reduce the computational time.

There are also methods that do not really track text but rather perform text localization for every $N^{th}$ frame. Methods in this category often employ

similarity measures to determine whether different text instances belong to the same text object.

Another aspect of text tracking is whether a method tracks general text or is designed for specific types of text motions, e.g., text moving from bottom to top in movie credits and text scrolling from right to left in news programmes. Most of the methods reviewed in this section are for general text.

In the following sections, the term *text object* refers to a single text line that appears in multiple frames. Each appearance in a particular frame is called a *text instance*.

### 2.4.1 Localization-based Tracking

As aforementioned, this approach does not really track text but instead focuses on matching different instances of the same text object. Several criteria have been explored: the degree of overlapping of the bounding boxes (Wolf et al. 2002), the sum of absolute differences (Lee et al. 2003), the sum of squared differences (Zhao et al. 2011) and the similarity of cumulative histograms of intensities (Shiratori et al. 2006; Tanaka & Goto 2008; Goto & Tanaka 2009). (Yi et al. 2009) combined these features and also introduced a new feature based on the similarity of edge pixel distributions. (Wolf 2003) matched text instances by comparing the horizontal and vertical projection profiles of Sobel gradient values. (Liu et al. 2010) went one step further and performed matching based on not only the magnitudes but also the orientations of the gradients.

The major drawback of this approach is that the tracking result is at a very coarse level (because text localization only is performed once every $N^{th}$

frame). The remaining sections cover methods which are capable of tracking text at the frame level.

### 2.4.2 Intensity-based Tracking

(Li & Doermann 1999; Li et al. 2000) performed text tracking using the sum of squared differences (SSD) under a pure 2D translational model between consecutive frames. This method is not able to track texts with complex motions due to the use of a pure translation model and the assumption that texts move at a constant velocity. Another drawback of this method, as mentioned in (Crandall et al. 2003), is that the SSD takes into account both the text pixels and the background pixels and thus the method may track both types of pixels, instead of just tracking the text pixels (Figure 2.11).

### 2.4.3 Signature-based Tracking

Instead of using the SSD of intensity values, (Wernicke & Lienhart 2000; Lienhart & Wernicke 2002) computed a signature (or descriptor) of each text object. Text localization was performed for every $30^{th}$ frame. If a text object was detected in a particular frame, it would be tracked both forward and backward. The signature of a text object was defined as the horizontal and vertical projection profiles of gradient magnitudes (Figure 2.12). Although the gradient-based signature is more robust than the SSD of intensity values, the authors mentioned that this method is still not able to track cases of text fading in/out or zooming in/out.

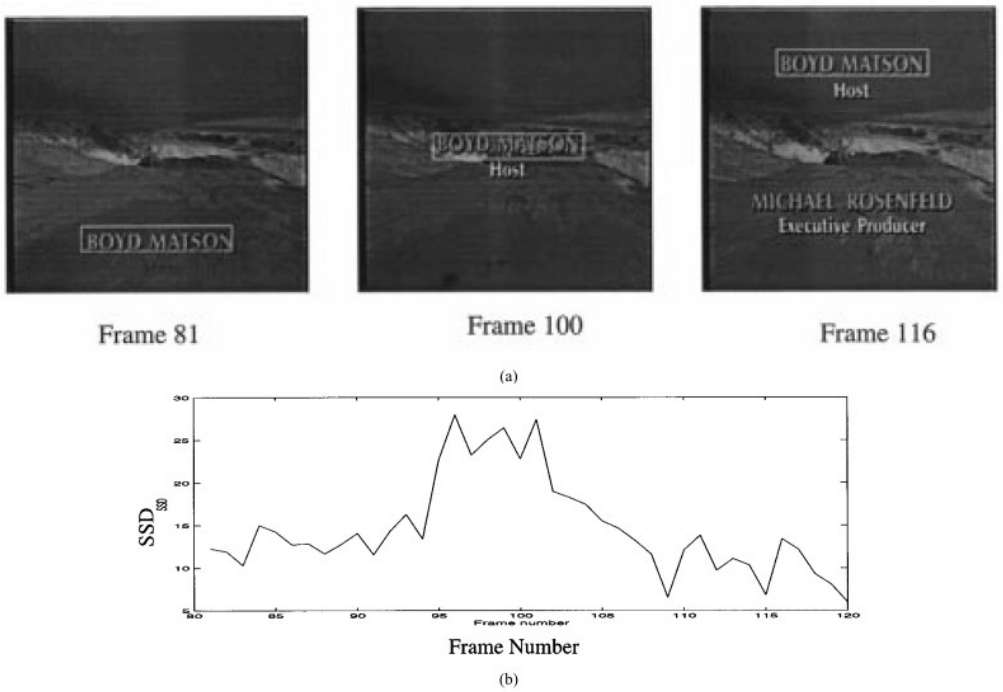



**Figure 2.11. SSD-based text tracking. Top row: different instances of the same text object. Bottom row: plot of SSD values. The SSD values increase significantly when the text object moves over a complex background (frame 100). (Figure taken from (Li et al. 2000).)**

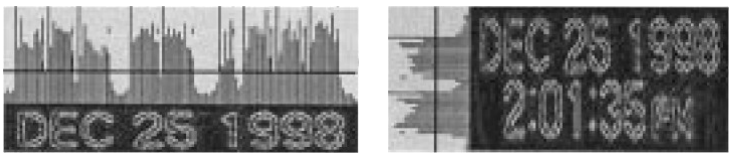



**Figure 2.12. Projection profiles of gradient magnitudes. (Figure adapted from (Lienhart & Wernicke 2002).)**

Projection profile-based signature was also used in (Qian et al. 2007). However, it was computed from the intensity values instead of the gradient magnitudes. This method faces the same problem as (Li et al. 2000) because intensity-based comparisons are not robust enough when text objects move through regions with complex backgrounds.

### 2.4.4 Probabilistic Tracking

Probabilistic tracking has been used for many computer vision tasks. Its main advantage is the ability to handle partial/total occlusion of the target (for a short period of time) by maintaining multiple hypotheses about the target's possible locations.

(Merino & Mirmehdi 2007) adopted particle filtering (Isard & Blake 1996; Isard & Blake 1998) for text tracking. Each text line was tracked by an independent tracker. The state representation was simply the 2D translation and rotation of the centroid of a text line. To handle the complex movements of text in natural scenes, a new state was generated based on the current state with added Gaussian noise. The observation likelihood was computed based on the Scale-invariant Feature Transform (SIFT) descriptors (Lowe 2004) of individual character CCs. Particle filtering was also used in (Minetto et al. 2011) for tracking text in outdoor videos. However, the observation likelihood was computed based on Histogram of Oriented Gradients features instead.

The advantage of these methods is that they are able to deal with partial occlusion of text objects (Figure 2.13). However, there is still room for improvement. For (Merino & Mirmehdi 2007), raw intensity values were used for CC generation. More robust features, as explored by text localization methods surveyed in section 2.3, can be used. In addition, the paper did not explain the handling of the interaction between the independent trackers. Switching of targets may occur if nearby text objects happen to contain similar characters. Similarly, the performance measures used in (Minetto et al. 2011) did not penalize target switch. Hence, it is not clear how well these two methods perform in this aspect. Avoiding target switch is important because

multiple-frame integration methods (section 2.5.2) assume that the different text instances to be integrated belong to the same text object (in other words, they must have exactly the same text content).


**Figure 2.13. By using a probabilistic framework, (Merino & Mirmehdi 2007) is able handle partial occlusion. However, the tracking result is at a very coarse level (the whole sign instead of individual text lines). (Figure taken from (Merino & Mirmehdi 2007).)**

### 2.4.5 Tracking in Compressed Domain

Similar to text localization, text tracking can be done in the compressed domain with the main advantage of reducing the computational time. (Gargi et al. 1999) and later (Crandall et al. 2003) explored using motion vectors in the P-frames of the MPEG format for text tracking (Figure 2.14). This section only covers the latter because it is an improved work of the former. In (Crandall et al. 2003), two tracking algorithms were presented, one for rigid text and the other for text that could rotate between different frames. Given a text instance in the current frame, the first algorithm searched for macroblocks in the next frame which pointed back to the text instance. A clustering algorithm was then performed on the motion vectors of the macroblocks. The mean of the largest cluster was chosen as the representative motion vector and was used to compute the position of the text instance. The authors mentioned

that the performance of this algorithm is dependent on the quality of the motion vectors, which is in turn affected by the encoding quality of MPEG videos.

The second tracking algorithm allowed text to change size and rotate. Since the motions were more complex, MPEG motion vectors were no longer used. Instead, the localization method proposed in the same paper was applied on every frame (similar to section 2.4.1 on localization-based tracking methods). A text object's signature was defined based on the character contours as it was assumed that the shapes of the characters stayed almost the same under growing, shrinking and rotation. A binarization method was applied to obtain individual character CCs from the localization result. Feature points (points with maximum curvature) were extracted from the CC contours and their coordinates were normalized to $[0, 1]$ to achieve scale invariance. The set of normalized coordinates then became the signature of a text object. Although it is still computationally expensive (because localization is performed on every frame), this method is one of the first attempts to track non-rigid text in complex video scenes, e.g., commercials. A drawback of this method is that the matching criterion is dependent on the quality of binarization. For example, suppose we have two instances of the same text object, one on clean background and the other on complex background. For the former, it is possible to obtain a good binarization result while for the latter, the binarization result may contain a lot of noise. Due to this difference in the binarization results, the two instances may be wrongly classified as belonging to two different text objects.

A closely related work to the above method is (Gllavata et al. 2004). The difference is that in the latter, instead of using all the macroblocks, a selection process was performed to increase the reliability of the motion vectors. Given the bounding box of a text instance in the previous frame, the intersection area between a macroblock and this box was required to be more than 30% of the macroblock's own area. This method shares the same advantages and disadvantages as the previous method. In particular, when the motion vectors are not available, e.g., in I-frames, both methods attempt to estimate this information from other P-frames, e.g., by assuming constant velocity. However, this may introduce errors.


**Figure 2.14. Motion vectors in a P-frame. (Figure taken from (Gllavata et al. 2004).)**

### 2.4.6 Summary

Text tracking plays an important role in a video text extraction system because it identifies the different instances of a text object, which is required for multiple-frame enhancement (discussed in the next section). Text tracking is, however, less researched than text localization (in quantity and to a certain extent, quality) (Zhang & Kasturi 2008). Many methods only handle simple translational motions, although there have been attempts to track non-rigid text

(Crandall et al. 2003) and partially occluded text (Merino & Mirmehdi 2007; Minetto et al. 2011). Tracking in compressed domain has also been explored by exploiting the motion vectors in the MPEG format.

An area that can be further explored is to determine which features to track. Some methods, e.g., (Lienhart & Wernicke 2002; Gllavata et al. 2004; Minetto et al. 2011), simply extract features from all the pixels in text regions (including the background pixels). Thus, instead of tracking only the texts, these methods may also (wrongly) track parts of the backgrounds. To deal with this problem, we present a technique for identifying the text pixels prior to tracking in section 4.2.

## 2.5 Text Enhancement

Although OCR engines work well for scanned documents, they do not produce satisfactory results out-of-the-box for natural scene images and video frames due to the reasons mentioned in section 2.1. The goal of text enhancement (and subsequently text binarization) is to pre-process a localized text image so that it can be recognized by an OCR engine.

For still images such as document images and video key frames, single-frame enhancement methods can be used to improve the contrast and/or resolution of a text image. However, using only a single frame, it is very difficult to achieve significant enhancement (Mancas-Thillou 2006). Multiple-frame enhancement methods, which exploit the redundancy in multiple video frames, generally have more success than single-frame enhancement methods. Two common types of multiple-frame enhancements are temporal integration and super resolution. The former enables better binarization results while the

latter helps to enlarge small texts and low-contrast texts so that they can be recognized by an OCR engine.

### 2.5.1 Single-frame Enhancement

As mentioned above, it is difficult to achieve significant enhancement using only a single frame. Therefore, existing methods often use simple techniques such as contrast stretching/histogram equalization (Kuo & Ranganath 1995; Lyu et al. 2005), bilinear interpolation (Sato et al. 1998; Sato et al. 1999), bicubic interpolation (Lienhart & Wernicke 2002; Pilu & Pollard 2002) and edge sharpening (Chen et al. 2001b; Mancas-Thillou 2006).

Another challenge of performing single-frame enhancement for natural scene images and video frames is that the text lines are typically much shorter than those in document images. Thus, it may not be feasible to utilize the information redundancy, as done in e.g., (Luong & Philips 2008) which exploited the multiple occurrences of the same characters in a single document image.

### 2.5.2 Multiple-frame Integration

One of the key differences between videos and images is the temporal redundancy, i.e., a text line may appear on the screen for several frames. A popular multiple-frame enhancement technique to simplify the background is to apply the max (min) operator for dark (bright) text on bright (dark) background (Figure 2.15), as done in e.g., (Sato et al. 1998; Sato et al. 1999; Lienhart & Wernicke 2002; Teo et al. 2004; Wang et al. 2004; Zhou et al.

2007). The rationale for this technique is that the text pixels of the same text object (i.e., with the same content) often have the same intensity values throughout the multiple frames while the background pixels keep changing. Therefore, applying the max/min operator on each individual pixel across the multiple frames does not affect the text pixels while simplifying the background considerably. This method does not work well if both the text and the background are moving, unless registration is performed to align the multiple text instances. Another drawback of the max/min operator is that it can be affected by a single "outlier" text instance, e.g., an instance that is much brighter or darker than the rest.

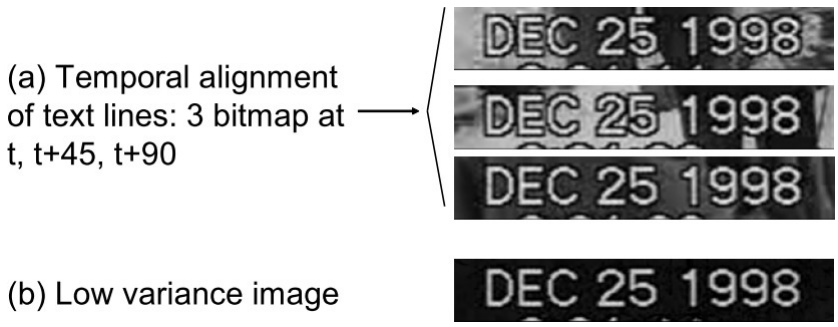



**Figure 2.15. Result of the max/min operator (b) on text instances (a). In this case, the min operator is used because text is brighter than the background. (Figure adapted from (Lienhart 2003).)**

An alternative technique is to take the average of the text instances (Figure 2.16), as done in e.g., (Li & Doermann 1999; Hua et al. 2002; Guo et al. 2007).

Furthermore, instead of using all the text instances, several methods have been proposed to select only the good frames. By assuming that text was monochrome and brighter than the background, (Hua et al. 2002) considered only frames of high contrast, which was measured by the percentage of dark pixels (Figure 2.17). The same idea could also be applied at the block level

(Figure 2.18). This method has two drawbacks. First, the selection criteria are rather simple and thus it does not work well for frames with complex backgrounds. Second, it is limited to bright text on dark background and does not deal with the other text polarity (dark text on bright background).

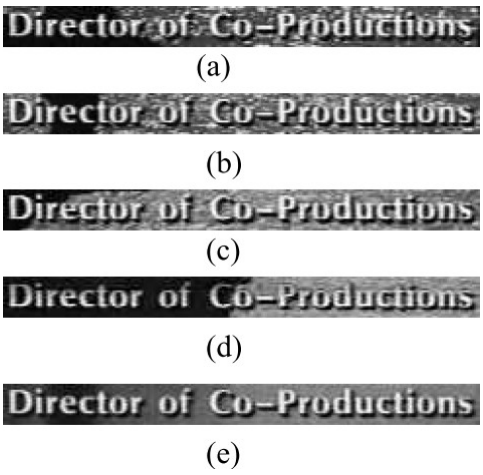



**Figure 2.16. Taking the average of text instances (a)-(d) helps to simplify the background (e). (Figure adapted from (Li & Doermann 1999).)**

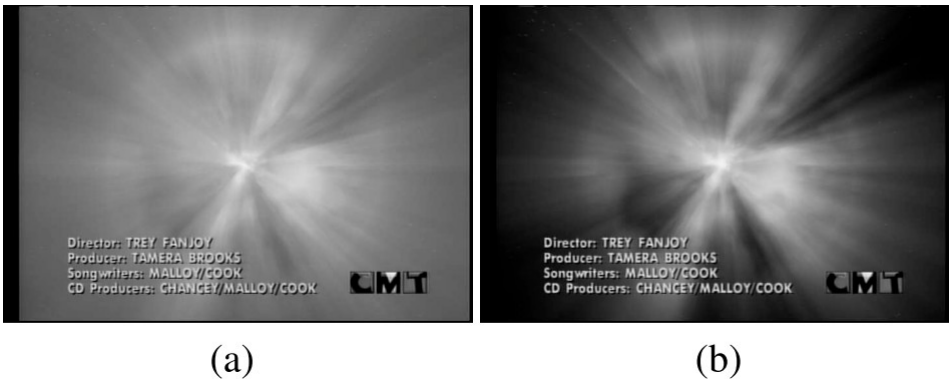



**Figure 2.17. The results of averaging all text frames (a) and averaging only the selected frames (b). The contrast between text and background in the latter is improved. (Figure taken from (Hua et al. 2002).)**


**Figure 2.18. Averaging at the frame level (left) and at the block level (right). The latter gives better contrast around the individual words. (Figure adapted from (Hua et al. 2002).)**

(Goto & Tanaka 2009) tested other criteria for frame selection based on the edge information and Otsu's binarization result. The best performing criteria were the number of edges and the sum of absolute values of edge intensities. (Yi et al. 2009) designed four filters to extract text strokes in four directions. Text instances were sorted based on their responses to the four filters. Only a few clearest instances were selected for enhancement. Another improvement of this work over the previous methods was the use of both the max/min operator and averaging. Based on Otsu's binarization result, averaging was applied for text pixels while the max/min operator was applied for background pixels. Like (Hua et al. 2002), this work is limited to bright text on dark background.

### 2.5.3 Multiple-frame Super Resolution

Other than multiple-frame integration, another approach to utilize the temporal redundancy in videos is multiple-frame super resolution. According to (Mancas-Thillou 2006), although there are many super resolution methods, most of them are for general images and do not exploit the characteristics of text. The two methods reviewed in this section have been applied for text in video/image sequences. They both formulate super resolution as an estimation of a high resolution image given a set of low resolution observations. In addition, both methods require the low resolution images to be registered beforehand.

(Capel & Zisserman 2000) compared four different estimators: an ML (maximum likelihood) estimator, an ML estimator with error back-projection, an MAP (maximum a posteriori) estimator and a Total Variation estimator.

The likelihood of an observed low resolution image given a high resolution estimate was:

$$P(m_n|\hat{s}) = \prod_{x,y} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{\widehat{\mathrm{m}}_{\mathrm{n}}(\mathrm{x},\mathrm{y}) - \mathrm{m}_{\mathrm{n}}(\mathrm{x},\mathrm{y})}{2\sigma^2}\right) \quad \textbf{(2.1)}$$

where $m_n$, $\hat{s}$ and $\widehat{m}_n$ were the observed low resolution image, the high resolution estimate and the estimate's projection to low resolution using the assumed image acquisition model, respectively. $\sigma$ was the variance of the Gaussian image noise (whose mean was assumed to be zero). With this formulation, the ML estimator did not work well and introduced a lot of noise in the high resolution estimate. This is because the problem was ill-posed due to the limited number of low resolution observations and the unknown blurring process (Mancas-Thillou 2006).

The second estimator improved the result of the ML estimator by incorporating an error back-projection method (Irani & Peleg 1991). Given a high resolution estimate, this method computed the errors between the observed low resolution images and the estimate's projection to low resolution (using the assumed image acquisition model). These errors were then mapped back to high resolution and used to update the estimate iteratively.

MAP, the third estimator, added a regularization term based on the Huber gradient penalty function and thus encouraged smooth and piecewise constant results. The last estimator achieved a similar effect by incorporating a Total Variation regularization term. Experimental results showed that the last two estimators produced visually sharper results than the first two estimators. It was also noticed that the Total Variation estimator required fewer

parameters than the MAP estimator. Unfortunately, the improvement in recognition rate was not reported.

In another approach, (Donaldson & Myers 2005) used a similar likelihood as the above method and compared two different priors for MAP estimation: a smoothness prior and a bimodality prior. While the former had often been used in the literature for general images, the latter was specific to text and had only been explored in very few works. The bimodality prior, which was defined as an exponential of a fourth-order polynomial, worked based on the assumption that the intensity histogram of a text image consisted of two peaks, one for the foreground and the other for the background (Figure 2.19). The experiments were conducted using three different priors: only the smoothness prior, only the bimodality prior and a combined smoothness-bimodality prior. The bimodality prior outperformed the smoothness prior. Moreover, adding the smoothness constraint to the bimodality prior did not improve the performance noticeably (Figure 2.20). This work could be further improved in the following two areas. First, the authors mentioned that the form of the bimodality prior was rather simplistic because it assumed that the foreground peak and the background peak had the same variance. Therefore, the bimodality model could be made more accurate (at the cost of a more complex optimization process). Second, the method is not fully automatic because parameters such as the blur diameters and the variances of the foreground and background peaks had to be manually tuned.

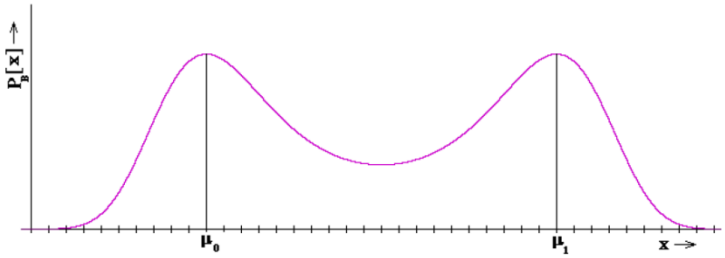



**Figure 2.19. The bimodality model used in (Donaldson & Myers 2005). $\mu_0$ and $\mu_1$ are the two intensity peaks. (Figure taken from (Donaldson & Myers 2005).)**

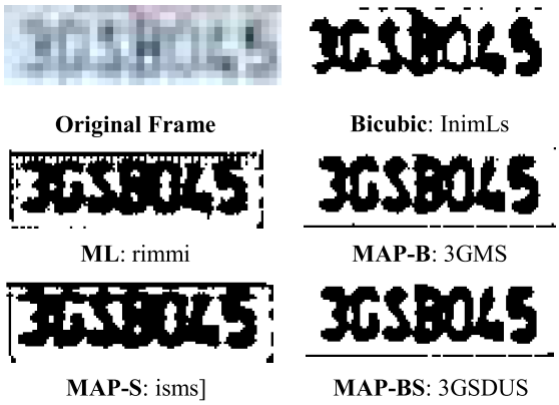



**Figure 2.20. Super resolution of text on license plates using 16 images. From left to right, top to bottom: one of the low resolution images, bicubic interpolation, ML estimation, MAP estimation with bimodality prior, MAP estimation with smoothness prior and MAP estimation with combined bimodality-smoothness prior. The text strings are the recognition results. (Figure taken from (Donaldson & Myers 2005).)**

### 2.5.4 Summary

In summary, text enhancement methods can be classified into single-frame enhancement and multiple-frame enhancement. Due to the limited information available, the former often employs simple techniques such as contrast stretching, bilinear/bicubic interpolation and edge sharpening. In section 4.1, we present a method for single-frame enhancement. We go beyond the simple techniques and propose a combination of character

segmentation and binarization[1]. In this way, our method can better adapt to the local background of each character and produce a superior enhancement result.

For multiple-frame enhancement, the two common techniques are temporal integration and super resolution. Given multiple instances of a text object, temporal integration helps to improve the contrast and simplify the background by using averaging or the max/min operator. A drawback of these operations is that they may also accidentally enhance the background regions. Hence, to overcome this drawback, we describe a method that focuses on only the text pixels in section 4.2. Our experiments show that the proposed method outperforms common operations such as average and max/min.

Another approach that utilizes multiple frames is super resolution, which aims to produce a high resolution estimate from low resolution observations. Both of the methods reviewed in this section use an MAP formulation and introduce text-specific priors such as smoothness and bimodality.

## 2.6 Text Binarization

Given a localized and enhanced text region, the binarization step helps to produce a black and white image which can be recognized by an OCR engine. Text pixels should be preserved and set to black while background pixels should be suppressed and set to white. Although binarization is usually performed at the text line level (i.e., the whole localized text region),

[1] Our work can be classified as both an enhancement method (because it helps to enhance the contrast of an image) and a binarization method (because it produces a black and white image as the final output). In section 4.1, we choose to present it as an enhancement method.

sometimes it is also done at the word level (Hua et al. 2002) and at the character level (Tang et al. 2002; Huang et al. 2009). The advantage of using a finer level is that the local background of each word/character may be less complex than the background of the whole line. Hence, it may be easier to perform binarization at these levels. In addition, different words/characters could be binarized using different parameter values.

A sub-problem of binarization is to determine text polarity, i.e., whether a text region contains normal text (bright text on dark background) or inverse text (dark text on bright background). This is required because some OCR engines only work for inverse text (Li et al. 1998). Some methods attempt to classify text polarity (Crandall et al. 2003; Lyu et al. 2005; Liu et al. 2006) while others simply assume a fixed polarity (Ngo & Chan 2005; Liu & Wang 2010).

Binarization is not a new problem and many methods have been proposed for document images. This section focuses on binarization methods that have been applied for scene images and video frames. These methods can be classified into three main approaches: intensity-based, color-based and stroke-based. The first approach analyzes the intensity histogram of a text region to find suitable binarization thresholds. Methods in this approach can be further classified as global and local, depending on whether they use a single threshold for the whole text region or different thresholds for different local regions. The second approach assumes that the characters in a text region have similar colors and thus often performs color clustering to identify the dominant peaks in the color space. The third and final approach focuses on extracting text strokes. A region growing process may also be used to include

neighboring pixels of the stroke pixels to obtain more complete shapes of the characters.

### 2.6.1 Intensity-based Binarization

Many traditional binarization algorithms for document images, e.g., (Otsu 1979; Niblack 1986; Sauvola & Pietikäinen 2000), belong to this approach. Although some papers for scene images and video frames use a global thresholding algorithm, e.g., Otsu's method (Chen et al. 2001b), most papers utilize an adaptive thresholding algorithm, e.g., Niblack's method (Li & Doermann 1999; Newman et al. 1999; Chen & Yuille 2004; Pan et al. 2011) and Sauvola's method (Wolf et al. 2002), due to their abilities to deal with intensity variations and complex backgrounds.

(Lyu et al. 2005) proposed an adaptive version of Otsu's method, together with an "inward filling" algorithm to clean up the binarization result. The method first moved a window horizontally across a text region and binarized the window's content using Otsu's algorithm. After that, another pass was done in the vertical direction in a similar manner. Given the binarization result, the method applied an "inward filling" process, a variant of the seed filling process proposed in (Lienhart & Wernicke 2002). The aim was to clean up non-text pixels by performing flood filling from the boundary. Figure 2.21 compares the binarization results of this method, Otsu's method (Otsu 1979) and Sato's method (Sato et al. 1998) (which will be described in section 2.6.3 on stroke-based binarization methods).


**Figure 2.21. From top to bottom: a text region, the binarization results by (Lyu et al. 2005), by (Otsu 1979) and by (Sato et al. 1998), and the ground truth. (Figure adapted from (Lyu et al. 2005).)**

The drawback of the above methods is that they do not model the spatial relationships between adjacent pixels. To overcome this problem, (Wolf & Doermann 2002) posed the binarization problem as an MAP estimation problem, in which Markov Random Field (MRF) was employed to capture the spatial relationships. The likelihood (or conditional density) was modelled as Gaussian noise with its mean being the shift amount of Niblack's adaptive threshold values from 127.5 (the average of 0 and 255) and its variance estimated using Otsu's method. The prior distribution was modelled by MRF and was learned from training data of binary characters for 4 × 4 pixel cliques. By capturing the spatial relationships between the pixels, this method was able to recover some of the missing pixels (Figure 2.22). However, its improvement in terms of the recognition rate was not significant. It only slightly outperformed Sauvola's method, a variant of Niblack's method. The authors mentioned that the method's performance is sensitive to parameter values, e.g., the variance of Gaussian noise.

Since each binarization method has its own strengths and weaknesses, (Chen et al. 2002; Chen et al. 2004a) proposed a multi-hypothesis approach in which pixels within a text region were clustered into K classes, with K varying from 2 to 4. Different clustering and labelling algorithms were tested: EM


(a) (b)

**Figure 2.22. Binarization results of Sauvola's method (a) and the MAP-MRF method in (Wolf & Doermann 2002) (b). By capturing the spatial relationships, the latter is able to recover some of the missing pixels. (Figure taken from (Wolf & Doermann 2002).)**

(Expectation Maximization), K-means and MRF. For each hypothesis (i.e., each of the K classes), CC filtering was used to remove non-text CCs based on geometric constraints. The experiments showed that K = 2 (i.e., bimodal) combined with either K-means or MRF gave the best results. The disadvantage of using multiple hypotheses is the high computational time. The MRF results of this work also confirmed those of (Wolf & Doermann 2002): modelling the spatial relationships of the pixels does not improve the recognition rate significantly.

### 2.6.2 Color-based Binarization

While the previous approach only uses intensity values, this approach also incorporates color information to obtain better binarization results. Similar to color-based text localization, color-based binarization mainly relies on the assumption that the characters in the same word/line have similar colors. In (Liu et al. 2010; Liu & Wang 2010), the text color was determined as the dominant peak in a 512-bin RGB color histogram. All the pixels in the corresponding histogram bin were marked as text pixels. As a verification step, the brightness values of the text pixels were also required to be greater than a pre-defined threshold. Due to the last step, these methods only handle bright text on dark background.

To deal with the problem of text polarity, (Wernicke & Lienhart 2000; Lienhart & Wernicke 2002) compared two color histograms, one covering the middle rows of a text region and the other covering a few rows immediately above and below the text region. The maximum and minimum values of the difference histogram were considered as the dominant text and background colors, respectively. Text polarity was then determined by simply comparing these two colors. For binarization, a seed filling algorithm was used to remove background pixels. The bounding box of a text region was first extended horizontally by 20% and vertically by 40% to ensure that no text pixels touched the boundary. After that, for every pixel on the boundary, a 4-neighborhood flood filling algorithm was applied where connectivity was defined based on the Euclidean distance of RGB colors. Finally, CC filtering based on geometric constraints was used to clean up the binarization result. As discussed in (Lyu et al. 2005), the drawback of this method is that the seed filling algorithm sometimes remove true text pixels which happen to touch the boundary (even after the bounding box has been extended).

Like the previous approach, this approach also has multi-hypothesis methods. In (Mancas-Thillou & Gosselin 2007), the foreground hypotheses of a text region were obtained by performing K-means clustering with two different measures: Euclidean distance and cosine-based similarity. Since these two measures were complementary, each was useful for different cases of text regions (Figure 2.23). The best hypothesis was chosen based on the average response to Log-Gabor filters, which were expected to produce large responses for text strokes. This method works well for natural scene images. However, the authors mentioned that using color information alone may not be

sufficient for challenging cases such as “embossed” texts (which have similar colors as the backgrounds).


**Figure 2.23. Different measures work well for different inputs: the input text regions (left) and the two foreground hypotheses, one based on Euclidean distance (middle) and the other one based on cosine similarity (right). (Figure taken from (Mancas-Thillou & Gosselin 2007).)**

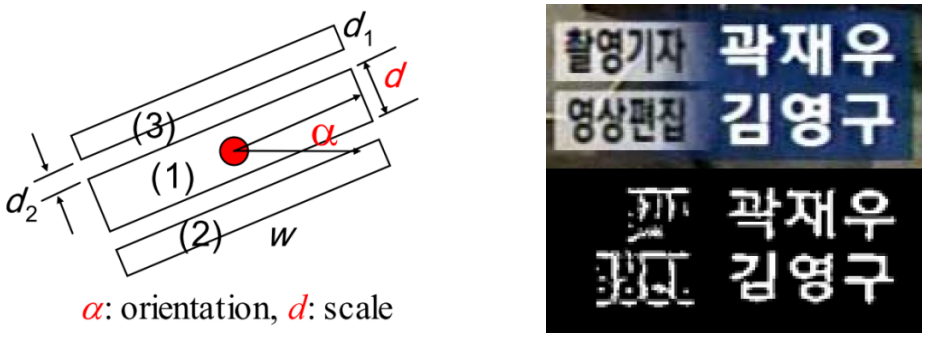
### 2.6.3 Stroke-based Binarization

Since a character’s shape can be approximated by its strokes, methods in this approach extracts text strokes from an input text region and use them as a “base” for the final binarization result. A region growing process may be used to include more pixels and obtain more complete shapes of the characters.

(Sato et al. 1998; Sato et al. 1999) designed four filters for extracting text strokes in four directions: horizontal, vertical, left diagonal and right diagonal. The filter responses were combined in a unified stroke map. Global thresholding with a fixed threshold was used to obtain the final binarization result. A drawback of this method, as discussed in (Lyu et al. 2005), is that it generates relatively weaker responses for stroke intersections (which appear more in Chinese texts than in English texts) than individual strokes.

Another stroke filter, with two parameters for scale and orientation, was designed by (Liu et al. 2006; Jung et al. 2008) (Figure 2.24a). The mean intensity values in the three local rectangular regions were used to calculate the filter response in two different ways, called the “bright” and “dark”

responses. Pixels of the correct polarity and closer to the filter center would generate greater responses. For binarization, the filter was applied on a text region with three different scales and four different orientations. From all the combinations of these two parameters, two overall response maps (one for bright and the other for dark) contained the largest possible response at each pixel. Given these two maps, SVM was used for text polarity classification based on two features: the sum of the response values and the number of edge points. The map chosen by SVM was then binarized by a simple adaptive thresholding algorithm. The authors mentioned that this method has difficulties with text lines which contain more than one polarity (Figure 2.24b).




(a) Stroke filter

(b) Sample result for text with two polarities

**Figure 2.24. (a) The stroke filter used in (Liu et al. 2006). (b) This method does not handle text with two different polarities well. (Figures adapted from (Liu et al. 2006).)**

### 2.6.4 Summary

In summary, text binarization converts an input text region to a black and white image which can be recognized by an OCR engine. It can be classified into three approaches: intensity-based, color-based and stroke-based. In addition, several methods have shown that different binarization algorithms

perform well for different cases of input text regions. Hence, a multi-hypothesis approach can be used to achieve better recognition accuracy at the cost of increasing the processing time.

An area that can be further explored is the use of binarization at a finer level (i.e., either at the word level or at the character level instead of at the text line level). Its advantage is that the local background of each character can be much simpler than the overall background of the whole text line. However, only a few works have pursued this direction (Hua et al. 2002; Tang et al. 2002; Huang et al. 2009). Our work in section 4.1 also follows this direction. Our experiments show that performing binarization at the character level helps to improve the recognition accuracy.

## 2.7 Text Recognition

Text recognition aims to produce the final output of a text extraction system in the form of text strings, which can be used for indexing purposes. There are two main approaches: recognition using OCR and recognition without OCR. The first approach relies on text enhancement (section 2.5) and text binarization (section 2.6) to produce a black and white image from a localized text region. This binary image is then fed into an OCR engine, e.g., ABBY FineReader[2] and Tesseract[3], to get the recognized text string. However, because OCR engines are designed for scanned documents of high resolution and with little distortion, they may not be able to recognize video texts of very low resolutions or scene texts with heavy distortions, e.g., perspective

[2] http://www.abbyy.com/
[3] http://code.google.com/p/tesseract-ocr/

distortions. To overcome this problem, in the second approach, researchers use their own feature extraction schemes and classifiers to achieve better recognition accuracy.

### 2.7.1 Recognition using OCR

One of the main advantages of using an OCR engine for recognition is that the engine typically makes extensive use of a language model (at the character level and at the word level) to improve the recognition accuracy. Thus, many methods simply use an OCR engine as a black box to recognize the binarized text regions (Lienhart & Wernicke 2002; Chen & Odobez 2005; Lyu et al. 2005; Wang et al. 2008; Huang et al. 2009; Liu et al. 2009). This is a sensible approach for texts which have sufficient contrast against the local backgrounds and can be well binarized by a text binarization algorithm. An example is video graphics text.

Various post-processing techniques can be used to refine the OCR output. For domain-specific applications, the recognition rate can be improved by introducing a lexicon. For example, a lexicon of team names and player names can be used for sports videos (Zhang et al. 2002; Ballan et al. 2010). A recognized text string can then be compared with the lexicon words. If the edit distance is below a pre-defined threshold, it can be corrected to the nearest word in the lexicon. Otherwise, if the distance is greater than the threshold, the string can be rejected.

For general post-processing, a popular technique is to use multiple hypotheses. (Chen & Odobez 2005) coupled binarization and recognition in a particle filtering algorithm and used a language model to combine the OCR

outputs of multiple hypotheses into a final text string. The state representation of the particle filtering algorithm was a pair of upper and lower binarization thresholds (assuming a 3-class model for the intensity histogram of a text region). The state transition probability within a small range of the current state followed a uniform distribution, with some Gaussian noise added at the boundary of the uniform range. The observation likelihood of a state was obtained by first binarizing the text region according to the state's pair of thresholds, getting the OCR output and finally computing its probability using a language model. The best K hypotheses from the above procedure went through a voting process which combined their OCR outputs into a single text string. Dynamic programming was used to align the multiple OCR outputs. Each character's confidence value in a column was defined based on the number of its occurrences in that column (Figure 2.25).

| | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | A | S | I | O | C | I | A | T | I | | P | R | O | D | U | C | r | R | s |
| | A | S | S | O | C | I | A | T | E | | P | R | O | D | U | C | E | i | s |
| | A | S | S | O | C | | A | 1 | [ | | P | R | O | D | U | C | E | R | S |
| | A | | | | | I | A | T | E | | P | R | O | D | U | C | E | t | B |
| | A | S | S | O | C | I | A | T | E | | P | R | O | D | U | C | [ | R | 5 |
| | A | | S | | | I | A | T | I | | f | R | O | D | U | C | | | |
| | A | S | S | O | C | I | A | I | I | | | | | D | U | C | E | R | S |
| | A | | | | | | o | | | | P | R | O | D | U | | | | |
| | | | | | o | | A | T | [ | | P | R | O | D | U | C | E | R | S |
| | | | | | o | | A | T | E | | P | : | ) | D | U | C | E | R | S |
| | | | | | | I | A | T | [ | | | | | D | U | C | E | R | S |
| | | | | | C | I | A | T | r | | P | R | O | D | U | C | E | R | S |
| | A | | g | O | C | I | A | T | E | | P | R | O | D | U | C | E | R | S |
| = | A | S | S | O | C | I | A | T | E | | P | R | O | D | U | C | E | R | S |

**Figure 2.25. The voting process used in (Chen & Odobez 2005) to combine the OCR outputs of different binarization hypotheses (all rows except the last one) into a single text string (the last row). (Figure adapted from (Chen & Odobez 2005).)**

Similar voting-based post-processing methods were also used in (Mita & Hori 2001) and (Liu et al. 2009). The difference between these two methods

and the previous method is that the multiple OCR outputs came from recognizing different text instances of the same text object in different video frames, rather than from recognizing different binarization hypotheses of the same text instance.

Another technique that has been used for post-processing is document-specific modelling (Kae et al. 2010). This method first performed a consistency check on the OCR output to identify a set of high confidence characters, i.e., characters whose different instances had consistently been given the same labels by an OCR engine. From this set of characters, it was possible to build a document-specific font model (instead of using the OCR engine's pre-trained font models). A multiclass SVM based on SIFT descriptors (Lowe 2004) was used for recognition. This method has offered a new perspective into the recognition problem. However, its success depends on the amount of information redundancy. Although this is inherent in document images, it may not be so for scene texts and video texts, which tend to be much briefer. For the case of video texts, characters from different frames can be used at the risk of violating the assumption made by this method: the characters should be of the same or very similar fonts.

In summary, recognition using OCR is a sensible approach for texts that can be well binarized. However, this approach has two drawbacks. First, for text regions with complex backgrounds, e.g., scene texts, text binarization may lead to information loss. Second, as mentioned before, because OCR engines are designed for the controlled settings of scanned documents, they are not able to handle texts with low resolutions, heavy distortions or fancy fonts.

### 2.7.2 Recognition without OCR

This approach overcomes the drawbacks of the previous one by using custom-built features and classifiers. Many methods in this approach also recognize grayscale/color images directly, thereby avoiding information loss.

According to (Casey & Lecolinet 1996), text recognition consists of four steps: format analysis, character segmentation, feature extraction and classification (Figure 2.26). The text localization step discussed in section 2.3 has already played the role of format analysis, i.e., locating the text lines. The remainder of this section covers the last three steps of text recognition: character segmentation, feature extraction and classification. It also surveys various methods for combining individual character recognition hypotheses into final word recognition results.

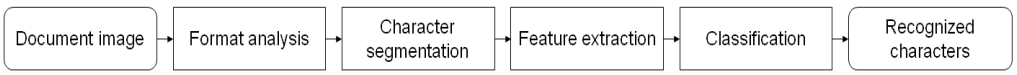



**Figure 2.26. The four main steps of text recognition. (Figure adapted from (Casey & Lecolinet 1996).)**

#### 2.7.2.1 Character Segmentation

Character segmentation, the splitting of a text line into individual character images, is a well-known problem in document analysis, especially for handling touching handwritten characters. (Casey & Lecolinet 1996) provided a comprehensive survey of character segmentation methods for document images. There are three main approaches mentioned in the paper: *dissection*, the decomposition of a text line image into individual character images, *recognition-based*, the use of recognition results to provide feedback

for segmentation, and *holistic*, the direct recognition of words (without segmentation) through matching features such as ascenders and descenders.

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

A common video character segmentation method is projection profile analysis (Lienhart & Wernicke 2002). The edge information (or other kinds of "energy") in each column is analyzed to distinguish between columns that contain text and gap columns. The former is often assumed to have higher energy than the latter. Heuristic rules have also been proposed to further split and merge the segmented regions based on assumptions about the characters' widths and heights (Miao et al. 2007; Huang et al. 2009). Although these methods are simple and fast, it is difficult to determine a good threshold that works for images of different contrast (Figure 2.27). In addition, because they work based on columns, they can only produce vertical cuts, which are not sufficient for difficult cases such as touching characters. (In these cases, more flexible cuts, e.g., curved cuts, would be needed to separate the characters from each other.)

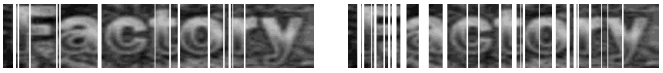


**Figure 2.27. The results of projection profile analysis are sensitive to threshold values. With a high threshold, true cuts are missed (left), while with a low threshold, many false cuts are detected (right).**

To overcome this problem, a number of papers, inspired by works on touching handwritten characters, modelled the segmentation problem as a minimum cost path finding problem. (Kopf et al. 2005) used Dijkstra's algorithm to perform path finding from the top row to the bottom row of the input image. A path's cost was defined as the cumulative absolute difference in grayscale intensities between consecutive pixels, based on the assumption that the background region had little variation in intensity. This method may not work well for images with complex backgrounds. In a similar approach, (Tse et al. 2007) applied path finding recursively until the segmented regions met the stopping criteria, e.g., their widths were below a threshold. The major drawback of this method is that it requires binarization to get CCs, which is extremely difficult to do reliably for scene characters and video characters, as aforementioned.

Different from the previous methods, (Saidane & Garcia 2008) used a machine learning technique, convolutional neural networks, for segmentation. The input was the three images corresponding to three color channels of the text line image. The output was a vector which classified whether each column was a gap column between consecutive characters. The drawback of this method is that it allows only vertical cuts.

#### 2.7.2.2 Feature Extraction and Character Classification

For a comprehensive survey of feature extraction methods for character recognition in document images, the reader is referred to (Due Trier et al. 1996). Some of the best performing features mentioned in that survey have been applied for scene text and video text. For example, (Zhang et al. 2002) used Zernike moments (Khotanzad & Hong 1990) to recognize text in basketball videos.

Gabor features, although not mentioned in the above survey, are also quite popular for character recognition. The advantage of Gabor features is that they can be extracted directly from grayscale images. (In contrast, features such as Zernike moments require binary inputs.) This is important because the binarization process may lose some valuable information available in the original grayscale image. In (Chen et al. 2004c), each character was normalized in size and divided into local regions using a 7 × 7 grid. Local Gabor wavelet features were then extracted from each region. An earlier work by (Yoshimura et al. 2000) also used Gabor filters but in a different way. Instead of pure Gabor jets (the responses to Gabor filters), this work used the accumulated values in four directions (Figure 2.28). The latter was found to be more robust to the different appearances of the characters.

Recently, Histogram of Oriented Gradients, a feature that has been shown to work well for many object recognition problems, has been adopted for recognizing natural scene characters in (Wang & Belongie 2010; Wang et al. 2011; Mishra et al. 2012a; Mishra et al. 2012b). Unsupervised feature learning has also been explored for the same problem in (Coates et al. 2011; Wang et al. 2012). Although these features have shown promising results on

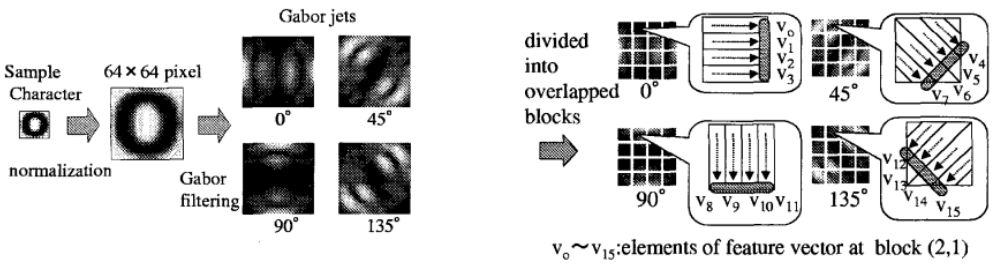



**Figure 2.28. Gabor jets (left) and the corresponding accumulated values in four directions (right). (Figures taken from (Yoshimura et al. 2000).)**

datasets for frontal scene characters, they are not robust to rotation and viewpoint change. Thus, they may not work well for scene characters which are affected by perspective distortions.

#### 2.7.2.3 Word Recognition

The word recognition result may not simply be the combination of the characters with the highest estimated probabilities. The rationale is that there may be errors in the estimated probabilities. Furthermore, in a given language, different character combinations may have drastically different frequencies. Therefore, most methods use additional information to find the most likely word from the set of character probabilities.

(Zhang & Chang 2003) proposed a framework which allowed for integration of multiple language models from different sources, e.g., combining a general linguistic corpus and the (specific) surrounding text in the web page that contained a video. The problem of word recognition was posed as an MAP estimation problem. The conditional density function was computed based on the features extracted, e.g., Zernike moments, and the prior was a linear combination of the different language models. The combination weights were learned from training data.

(Weinman & Learned-Miller 2006; Weinman et al. 2009; Smith et al. 2011) proposed a novel similarity constraint to force characters which were visually similar to take the same label. The appearance features of each character were extracted using Gabor filters at 3 different scales and 6 different orientations. The similarity between two characters was then computed using the vector angle distance. The final probabilistic framework classified a character using three different sources of information: the individual character appearance, the language model and the similarity between all pairs of characters in the input text region.

(Wang & Belongie 2010; Wang et al. 2011) adopted Pictorial Structures (Felzenszwalb & Huttenlocher 2005), an object recognition framework, for word recognition. In this framework, each word was considered as an object. The characters of the word then became the object parts. The score of a word took into account both the individual character probabilities and the regularity of the characters (in terms of sizes, distances and so on). Dynamic programming was used to return the top-scoring words. These works have shown that object recognition frameworks can be adopted for text recognition. However, their drawback is that they require all characters of a word to be correctly recognized. In other words, they cannot handle cases where one or more characters are occluded.

To overcome this drawback, recent works formulate word recognition as an optimization problem. (Wang et al. 2012) estimated the probabilities of the character candidates and used a variant of the Viterbi algorithm (Sarawagi & Cohen 2004) to find the optimal alignment between the character candidates and the words in a lexicon. (Mishra et al. 2012a; Mishra et al. 2012b) went one

step further and incorporated the language model, i.e., character n-grams, into the optimization process. The character candidates were used to build a Conditional Random Field (CRF) (Lafferty et al. 2001) with unary and pairwise terms. The unary terms captured the character probabilities while the pairwise terms captured the character bigram statistics. Finally, the CRF energy was minimized to find the optimal word. Similarly, (Novikova et al. 2012) used weighted finite-state transducers (Mohri et al. 2002; Povey et al. 2012), which was capable of capturing both the character probabilities and the language model, for optimization. The drawback of these works is that they were only tested on scene texts that were frontal parallel to the image plane.

In practice, scene texts can appear with perspective distortion. One approach is to rectify perspective texts prior to recognition, e.g., (Dance 2001; Myers et al. 2005; Neumann & Matas 2010). However, these methods rely heavily on the quality of the binarized character shapes. Thus, although they work for texts on plain backgrounds, it is unclear whether they can handle texts with cluttered backgrounds. In a recent work, (Li & Tan 2010) recognized perspective characters without rectification. However, this work only focused on character recognition, and did not address word recognition. The dataset was also limited to simple sign images. Therefore, despite its importance, the issue of handling perspective texts has not been adequately addressed by existing works.

### 2.7.3 Summary

In summary, there are two approaches to text recognition: recognition using OCR and recognition without OCR. Methods in the first approach

simply use existing OCR engines as a black box and instead focus on pre-processing, e.g., enhancement and binarization, and post-processing, e.g., using lexicon, language model, and voting based on the OCR outputs of multiple hypotheses.

In the second approach, researchers propose their own character segmentation and recognition schemes to improve the recognition accuracy. For character segmentation, many methods only allow vertical cuts and thus may fail to separate low contrast and touching scene characters and video characters. To overcome this drawback, we propose a character segmentation technique that is able to produce curved segmentation paths in section 4.1. For character recognition, various schemes have been explored, from traditional features such Zernike moments and Gabor filters to more recent techniques such as Histogram of Oriented Gradients and unsupervised feature learning. Finally, for word recognition, most methods use not only the character probabilities but also additional information like the language model to find the most likely word. To incorporate multiple sources of information, different optimization frameworks have been used such as CRF and weighted finite-state transducers.

It is worth mentioning that most recognition methods are still limited to texts that are frontal parallel to the image plane. However, in many real-world scenarios, texts (especially those in natural scene images) suffer from various deformations such as perspective distortions. Therefore, to address this issue, we present a method for recognizing perspective scene texts in Chapter 5.

This section concludes our review of existing text extraction techniques. Based on the review, we have identified several research gaps. The following chapters present our works to address them.

# Chapter 3

# Text Localization in Natural Scene Images and Video Key Frames

This chapter describes our work on text localization. We have proposed two methods, one for natural scene images and the other for video key frames. The first method introduces novel inter-character features, which are extracted from the spaces between consecutive characters, to localize difficult cases of scene text. The second method employs skeletonization to localize multi-oriented video text. This is an improvement over existing works, which typically pick up only horizontal text.

## 3.1 Text Localization in Natural Scene Images

### 3.1.1 Motivation

From our survey in section 2.3, we have noticed that most existing methods focus solely on character features (e.g., character intensity values (Neumann & Matas 2012) and character stroke widths (Epshtein et al. 2010)). There has been little work on utilizing the gap regions between consecutive characters for the purpose of text localization. This information is useful for cases in which the character edges are broken and not reliable for extracting features, but the regular gaps between them are still visible. Therefore, we propose a novel method that combines both the character features and the gap features.

Our contributions are two-fold. (1) To exploit both text and gap features, we employ Gradient Vector Flow (GVF) (Xu & Prince 1998) for symmetry detection. To the best of our knowledge, this is the first attempt to use GVF for text localization. (2) Our method is shown to work well on two public datasets.

### 3.1.2 Proposed Method

We first use GVF to detect local symmetries and identify character candidates. The second step then groups these characters into text lines based on the similarity in size, GVF distance and color. The final step performs texture analysis to remove false positives.

#### 3.1.2.1 Text Candidate Identification

Characters in a line exhibit many local symmetries. In this work, we focus on two types of symmetries: *intra-character symmetry* (or self-symmetry) and *inter-character symmetry* (or symmetry between consecutive characters). The former arises because of the symmetry between the inner and outer contours of the same character. The latter is due to the correspondence between the outer contours of two consecutive characters.

Therefore, to locate text regions, we adopt GVF (Xu & Prince 1998) to extract both types of symmetries. GVF is traditionally used together with active contour for non-rigid registration and motion tracking. Unlike the normal gradient which gives little information in homogenous regions, GVF propagates the gradient information, i.e., the magnitude and direction, from

nearby regions into these regions. Hence, it helps to increase the capture range of the edges and attract the active contour into concave regions. The GVF field is computed by minimizing the below expression (Xu & Prince 1998):

$$\mathcal{E} = \iint \mu\left(u_x^2 + u_y^2 + v_x^2 + v_y^2\right) + |\nabla f|^2 |g - \nabla f^2| dxdy \tag{3.1}$$

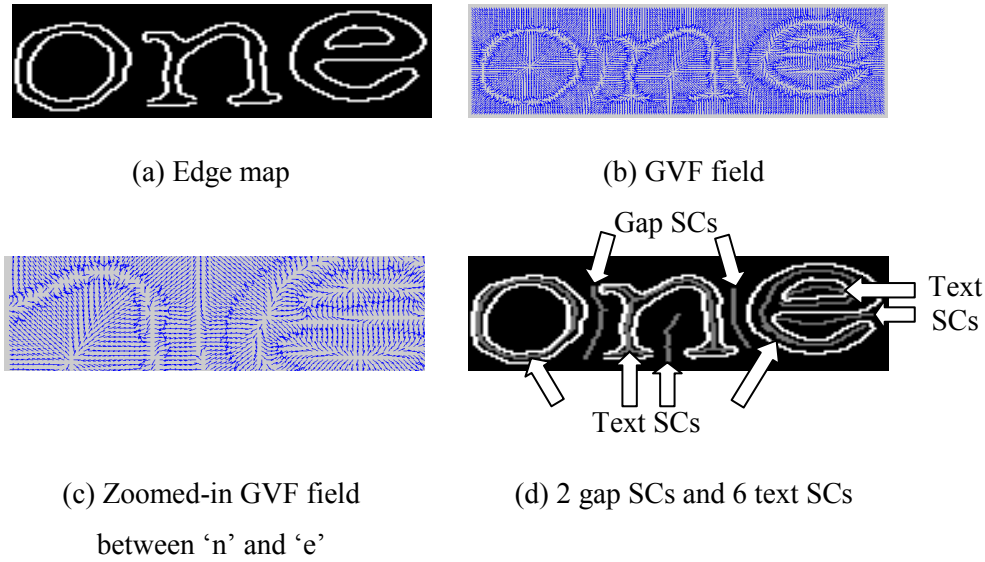
where $g(x,y) = \left(u(x,y), v(x,y)\right)$ is the GVF vector field. $u(x,y)$ and $v(x,y)$ represent the horizontal component and the vertical component of a GVF vector, respectively. $f(x,y)$ is the edge map of the input image (Figure 3.1a and b).

A property of the GVF field is that starting from any point, if we follow the GVF directions, we will reach a nearby edge. Thus, local symmetry points are identified as the locations where two neighboring GVF arrows are opposite of each other, because this indicates that the region is at the center of two edges (Figure 3.1c).

Concretely, $(x,y)$ is a *vertical* symmetry point if and only if:

$$\begin{cases} u(x,y) < 0 \\ u(x+1,y) > 0 \\ angle\left(g(x,y), g(x+1,y)\right) > \theta_{min} \end{cases} \tag{3.2}$$

Intuitively, the above three conditions require that: the GVF vector at $(x,y)$ points to the left, the GVF vector at $(x+1,y)$ points to the right, and there is an adequately large angle (e.g., greater than $\pi/6$) between them. The last condition ensures that there are enough attraction forces from the edges on the two sides.




(a) Edge map (b) GVF field

(c) Zoomed-in GVF field between 'n' and 'e' (d) 2 gap SCs and 6 text SCs

**Figure 3.1. GVF helps to detect local text symmetries. In (d), the 2 gap SCs and the 6 text SCs are shown in gray. The two gap SCs are between 'o' and 'n', and between 'n' and 'e'. The remaining SCs are all text SCs.**

In addition to the vertical direction, we derive similar constraints for symmetry points in three other directions: horizontal, left-diagonal and right-diagonal. Together, they represent the four main orientations of text strokes (Liu et al. 2005), and thus help to detect most of the local symmetries in text regions.

To illustrate the effectiveness of the proposed symmetry detection, we first apply it to a single text line (Figure 3.1d). The intra-character and inter-character symmetries are shown in gray. Due to the structure of text, symmetry points of the same type often form "clusters". Hereafter, we refer to these clusters as *symmetry components* (SC).

For the purpose of scene text localization, we use the same process on full images. In Figure 3.2b and c, $f(x, y)$ in Equation (3.1) is set to the Sobel edge map and the Canny edge map (Canny 1986), respectively. Similar to the previous example, most local symmetries in text regions are picked up.

Although SCs are also detected for symmetrical structures in the background, it is possible to distinguish between the two cases.

The key difference between text SCs and non-text SCs is that when GVF is run on two different edge maps, the former remains relatively stable while the latter is highly inconsistent. The rationale is that the character edges in the two edge maps resemble each other, while there are drastic changes in the background, i.e., a lot more edges are picked up by the Canny edge detector but not by the Sobel edge detector.

We have also observed that using Sobel-GVF gives better precision (i.e., a higher percentage of SCs are true text SCs). However, Canny-GVF helps in recall, especially for small text.

Based on the above two observations, we propose a relaxed intersection to filter out non-text SCs and combine the advantages of both Sobel-GVF and Canny-GVF. Let $C = \{c_i\}$ and $S = \{s_j\}$ be the sets of Canny and Sobel SCs, respectively. $c_i$ is retained if:

$$\exists s_j : \frac{|c_i \cap s_j|}{|c_i|} \geq T_1 \tag{3.3}$$

Otherwise, it is removed. Thus, this intersection retains only Canny SCs which have sufficient overlap (measured by the number of pixels) with a Sobel SC. Based on the training data, $T_1$ is set to 0.4. (In section 3.1.3.5, we analyze how the performance of the proposed method changes with respect to this parameter.) Figure 3.2d shows that the above process helps to suppress most of the non-text SCs.

Note that the use of two different edge maps as above can also be interpreted in the spirit of hysteresis thresholding, a physics-inspired technique that uses two thresholds instead of one for better result (Canny 1986). Sobel-GVF and Canny-GVF correspond to the high and low "thresholds", respectively. The former contains high confidence symmetry points but may suffer in recall. Thus, the latter is used to recover points that are connected to the high confidence ones.


(a) Input image (b) Sobel SCs

(c) Canny SCs (d) Text candidates

**Figure 3.2. Text candidate identification.**

#### 3.1.2.2 Text Grouping

The remaining SCs (after filtering) in Figure 3.2d are considered as text candidates. The purpose of this step is to group horizontally-aligned text candidates which have consistent properties into text lines. In each iteration of the grouping process, we create a new group (i.e., text line) which initially contains the first unassigned SC in the input image (in top left to bottom right order). In the immediate left and right regions of the current group, if there are

any unassigned SCs which satisfy the similarity constraints (to be defined below), we add them into the group and re-examine the expanded neighborhood. Otherwise, we go to the next iteration and repeat the process, until there are no more unassigned SCs.

The similarity constraints are based on the following observations:

- In a line, characters have comparable **heights**.
- The character **stroke thickness** is consistent. Similarly, the spaces between characters (**gap "thickness"**) are regular.
- Many scene texts are of uniform **colors** (so that they are easy to read from distance).

More formally, let $g$ be the current group and $c$ be an unassigned SC in $g$'s neighborhood. $c$ is added to $g$ if:

$$\begin{cases} T_2 \leq \dfrac{Height(c)}{MedianHeight(g)} \leq 1/T_2 \\ |GVFDistance(c) - MedianGVFDistance(g)| \leq T_3 \\ RGBColorDistance(c, g) \leq T_4 \end{cases} \quad \textbf{(3.4)}$$

where $GVFDistance(.)$ returns the distance from a SC to either of the two edges that give rise to it. (This can easily be computed by following the GVF directions until we reach the edges.) If $c$ is an intra-character SC, the distance corresponds to half of the stroke thickness. Otherwise, if $c$ is an inter-character SC, it corresponds to half of the gap thickness. Therefore, this constraint fully captures the second observation mentioned above.

Although stroke thickness has been explored in a previous work (Epshtein et al. 2010), gap thickness is new and is a by-product of the GVF

formulation in the previous section. Thus, GVF not only helps to detect local symmetries but also plays an important role in the grouping process.

The parameter values are determined empirically based on the training data: $T_2 = 0.4$, $T_3 = 6$ and $T_4 = 70$. Figure 3.3a shows some of the groups formed by the text candidates in Figure 3.2d. Note that the first two groups actually cover the whole length of the first text line. However, for illustration purpose, only the word "Centre" is shown. Furthermore, the second group corresponds to the gaps between the characters (rather than the characters themselves). This is the advantage of the proposed method. Groups formed by gap SCs are especially useful for difficult cases of scene text where the character edges are broken, but the gaps are still visible.

To obtain the final grouping output in Figure 3.3b, we impose one more constraint: most text lines have at least three characters (Epshtein et al. 2010; Chen et al. 2011). Hence, only groups with three or more SCs are retained. For cases of double detection, i.e., two groups are detected for the same text line, we take the union of the two bounding boxes.


(a) Text groups (b) Output for Figure 3.2a

**Figure 3.3. Text grouping. In (a), the SCs are shown in white. For the second group, the characters are shown in gray to illustrate why the gap SCs are detected in the first place.**

#### 3.1.2.3 Text Verification

At the end of the previous step, we obtain a set of candidate text lines. However, some of them may correspond to text-like patterns, e.g., symmetrical structures with regular spaces, in the background. Hence, we perform local texture analysis for verification purpose.

To learn the texture of text, we use Histogram of Oriented Gradients (HOG) (Dalal & Triggs 2005), a popular descriptor that has successfully been employed for many object detection problems. Using the training data, we collect 11,600 positive samples and 14,100 negative samples. The patch size is fixed to 48 × 48.

Furthermore, as suggested in (Chen & Yuille 2004), we divide a patch into three partitions (Figure 3.4a). HOG features are extracted for each partition and then concatenated to form the feature vector. The rationale for such a division is that the top 1/6 and bottom 1/6 correspond to the ascender and descender of text, which typically have different gradient orientation distributions than the middle partition. Experimentally, we have found that this approach is more effective for rejecting false positives than applying HOG on the whole patch (i.e., without division).

SVM is used to classify whether each candidate region is text or non-text. We normalize each region to 48-pixel height. A window is then slid across, and at each position, SVM is used to estimate the confidence score that the window contains text. The overall score of the region is computed by a weighted average where the weights follow a Gaussian distribution (Chen et al. 2004a). If the score is non-negative, the region is retained; otherwise, it is discarded.

Figure 3.4b shows some of the false positives that have successfully been removed by HOG-SVM. It is evident that the texture features supplement the symmetry features used in the previous steps.

(a) (b)

**Figure 3.4. Block pattern (a) and sample false positives that are successfully removed by using HOG-SVM (b).**

### 3.1.3 Experimental Results

#### 3.1.3.1 Datasets

We performed experiments on two public datasets, ICDAR 2003 (Lucas et al. 2003) and Microsoft Text Detection dataset (MS) (Epshtein et al. 2010). The first dataset consists of the training set (250 images) and the test set (249 images). It has a wide range of images, e.g., book covers, bill boards and outdoor scenes. The resolutions vary from 307 × 93 to 2048 × 1536. The second dataset contains 307 street images, with resolutions from 1024 × 768 to 1280 × 960.

Because the MS dataset does not contain separate training data, our method was trained on the ICDAR training set for both experiments. The parameter values, as mentioned in the previous sections, also remained the same for both experiments.

#### 3.1.3.2 Performance Measures and Methods for Comparison

For quantitative evaluation, we used the performance measures of the ICDAR 2005 competition (Lucas 2005): precision (P), recall (R) and f-measure (F). These measures are briefly summarized below.

Let $E$ be the set of bounding boxes returned by a text localization method and $T$ be the set of groundtruth bounding boxes. The best matched area of a bounding box $r$ with respect to a set $R$ is defined as:

$$m(r, R) = \max m_p(r, r') \mid r' \in R \tag{3.5}$$

where $m_p(r, r') \in [0,1]$ is the intersection area of the two bounding boxes divided by the area of the minimum box that encloses both boxes. $m_p = 1$ if the two boxes are identical and $m_p = 0$ if there is no overlap.

Precision, recall and f-measure are then computed as:

$$precision = \frac{\sum_{r_e \in E} m(r_e, T)}{|E|} \tag{3.6}$$

$$recall = \frac{\sum_{r_t \in T} m(r_T, E)}{|T|} \tag{3.7}$$

$$f = \frac{1}{\frac{\alpha}{precision} + \frac{1-\alpha}{recall}} \tag{3.8}$$

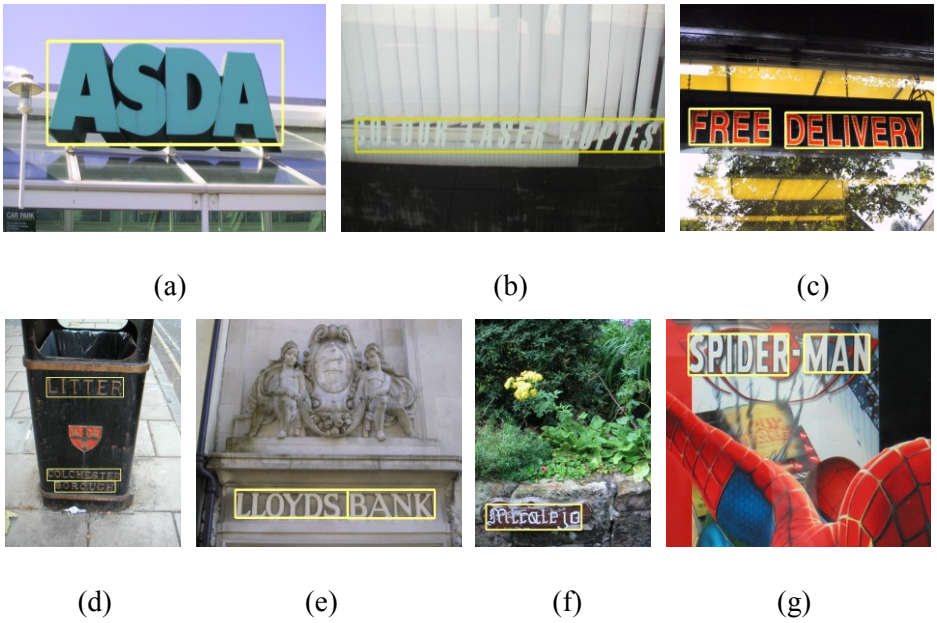
where $\alpha = 0.5$ to balance recall and precision.

In the experiments, we compared the proposed method against two recent localization methods for scene text: (Neumann & Matas 2011) and (Epshtein et al. 2010).

#### 3.1.3.3 Experiment on Natural Scenes

In the ICDAR 2003 dataset, the ground truth is provided at the word level. We used projection profile analysis to split the localized text lines into words. Figure 3.5 shows sample localization results of our method. In Figure 3.5a, the text line contains touching characters. It is mentioned in (Neumann & Matas 2011) that this method cannot handle such cases because it requires each Maximally Stable Extremal Region to be an isolated character. Similarly, the method by (Epshtein et al. 2010) expects a text line to have at least three characters, which is not satisfied because the whole line is extracted as one single region (with consistent stroke width). On the other hand, our method detects skeleton-like SCs, which are disconnected from each other and thus form a valid group. Figure 3.5b shows another challenging case where text has the same color as the background. In the edge map, the top portions of the character edges are lost. However, the bottom and especially the side contours are still visible. Thus, our method is able to detect and group SCs into a text line, while the methods by (Neumann & Matas 2011) and (Epshtein et al. 2010) will have difficulties because the characters become connected to each other (through the background at the top).

Figure 3.6 illustrates that a variety of texts in the dataset are successfully picked up despite the stylish fonts, blurring, partial occlusion and complex backgrounds.




(a) (b) (c)

(d) (e) (f) (g)

**Figure 3.5. Sample text localization results on the ICDAR 2003 dataset.**


**Figure 3.6. Sample localized text lines on the ICDAR 2003 dataset.**

On the ICDAR test set, the proposed method outperformed the top entries of the ICDAR 2005 competition, as well as recent methods (Epshtein et al. 2010; Neumann & Matas 2011), in terms of f-measure (Table 3.1). (Note that the ICDAR 2005 competition reused the dataset from the ICDAR 2003 competition).

Our method also achieved the highest recall, which shows the advantage of using GVF SCs to exploit both text and gap features to pick up more text lines. On the other hand, existing methods typically ignore the latter. Both methods by (Epshtein et al. 2010) and (Neumann & Matas 2011) only extract character features, through Stroke Width Transform and Maximally Stable Extremal Regions, respectively.

The proposed method had a slightly worse precision than the methods by (Epshtein et al. 2010) and (Neumann & Matas 2011) because the background may contain structures that happen to satisfy the symmetry constraints, e.g., regular vertical stripes. This problem will be explored in the future. For example, an OCR engine can be employed to recognize these patterns and reject highly unlikely strings, e.g., '11111'.

**Table 3.1. Results on the ICDAR 2003 dataset.**

| Method | Precision | Recall | F-measure |
|---|---|---|---|
| 1st ICDAR 2005 | 0.62 | 0.67 | 0.62 |
| 2nd ICDAR 2005 | 0.60 | 0.60 | 0.58 |
| (Neumann & Matas 2011) | 0.72 | 0.62 | 0.67 |
| (Epshtein et al. 2010) | **0.73** | 0.60 | 0.66 |
| Our method | 0.70 | **0.69** | **0.69** |
| Our method without HOG | 0.63 | **0.69** | 0.66 |

#### 3.1.3.4 Experiment on Street Scenes

Figure 3.7 shows sample localization results of our method on street images. Figure 3.8 shows that our method is able to pick up texts of a variety of appearances.

Our method achieved a significantly higher recall and a better overall f-measure than the method by (Epshtein et al. 2010) (Table 3.2). (The results of the other methods on this dataset are not available.). Both the proposed method and the method by (Epshtein et al. 2010) degraded in performance on this dataset because it is more challenging than the ICDAR dataset. The images have a wider view and contain more objects, e.g., buildings, pedestrians and cars. Moreover, 45% of the text lines are less than 20 pixels in height. (That in the ICDAR dataset is only 9%.)


**Figure 3.7. Sample text localization results on the Microsoft dataset.**


**Figure 3.8. Sample localized text lines on the Microsoft dataset.**

**Table 3.2. Results on the Microsoft dataset.**

| Method | Precision | Recall | F-measure |
|---|---|---|---|
| (Epshtein et al. 2010) | **0.54** | 0.42 | 0.47 |
| Our method | 0.50 | **0.51** | **0.51** |
| Our method without HOG | 0.44 | **0.52** | 0.48 |

#### 3.1.3.5 Additional Experiments

To show the contribution of the first two steps alone (i.e., local symmetry detection and text grouping), we turned off the HOG-based text verification step (denoted as *Our method without HOG* in Table 3.1 and Table 3.2). The last rows in these two tables indicate the effectiveness of the proposed symmetry detection, as the recalls on both datasets were much higher than those of (Epshtein et al. 2010) and (Neumann & Matas 2011). In addition, it shows that the texture feature supplements the symmetry features, and helps to improve the precision.

We also examined how the performance of our method changes with respect to $T_1$ in Equation (3.3). Figure 3.9 demonstrates that the overall f-measure is not too sensitive to $T_1$, as long as $T_1 \in [0.3,0.5]$.

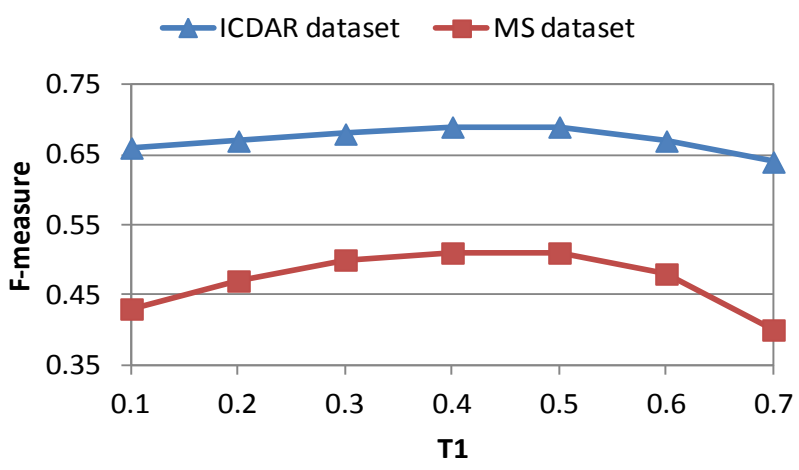



**Figure 3.9. F-measures for different values of $T_1$.**

One of the main contributions of this work is the local symmetry detection technique based on GVF. It allows our method to exploit both the text features and the gap features to localize text regions. The latter is new and has not been explored by existing methods.

In addition, the proposed GVF-based symmetry detection technique will be further used in section 4.1 for a different purpose: single-frame text enhancement. In that section, due to the nature of the task, we only focus on the inter-character symmetry (i.e., the gap SCs) and ignore the intra-character symmetry (i.e., the text SCs).

## 3.2 Text Localization in Video Key Frames

Our scene text localization method in the previous section follows a bottom-up approach. However, video frames are typically of much lower resolutions than natural scene images. Due to this challenge, it is difficult to reliably extract the video characters and the gaps between them. Hence, a bottom-up approach is not suitable for video frames. This section presents a top-down approach to text localization in video frames.

This work does not utilize the temporal information (yet) so its input is a video key frame. The use of temporal information will be discussed in a later work in section 4.2.

### 3.2.1 Motivation

It is evident from our survey in section 2.3 that most methods for videos address the localization of horizontal text but not multi-oriented text. This is because most of the non-horizontal text lines are video scene text, which is much more difficult to localize due to varying lighting and complex transformations (Jung et al. 2004; Zhang & Kasturi 2008). For some existing methods, extension to multi-oriented text is no trivial matter. For example, the uniform-colored method (Mariano & Kasturi 2000) performs color clustering

on each row, while the gradient-based method (Wong & Chen 2003) identifies candidate text segments row-wise. The edge-based method (Cai et al. 2002) analyzes the horizontal and vertical projection profiles of the edge map. Thus many existing methods rely heavily on the horizontal text assumption and break down on multi-oriented text.

Only a few papers consider text of arbitrary orientation in video, e.g., (Kim et al. 2003; Wang et al. 2008), under the assumption that text is of large font size and of high contrast. (Crandall et al. 2003) proposed a method for extracting multi-oriented special effects text; however, this method is limited to graphics text of fixed directions (0, 15, 30 degrees and so on).

A few methods for text in natural scene images, e.g. (Chen et al. 2004c; Epshtein et al. 2010; Yi & Tian 2011), can handle multi-oriented text. However, they require each single character to be extracted as a complete CC. For example, to allow for non-horizontal text, (Chen et al. 2004c; Yi & Tian 2011) performed line fitting using Hough transform on individual character CC centroids. Although this requirement is reasonable for scene characters, it is **not** guaranteed for video characters. Due to the poor resolution and the low contrast of the video frames, a CC may only contain the partial shape of a character and thus the performance of these CC-based methods will degrade.

Multi-oriented text localization in video key frames without any constraints on background, contrast and orientation, and with high precision and recall is still a challenging problem (Lienhart & Wernicke 2002; Crandall et al. 2003; Lyu et al. 2005). Therefore, we propose a method which is able to handle video graphics text and video scene text of arbitrary orientation under the assumption that the characters are aligned on a straight line.

### 3.2.2 Proposed Method

The proposed method consists of four steps: text localization, connected component classification, connected component segmentation and false positive elimination. In the first step, we identify candidate text regions by using the Laplacian operator. The second step uses skeletonization to analyze each CC in the text regions. Simple CCs are retained while complex CCs are segmented in the third step. False positives are removed in the last step. Figure 3.10 shows the flowchart of the proposed method.

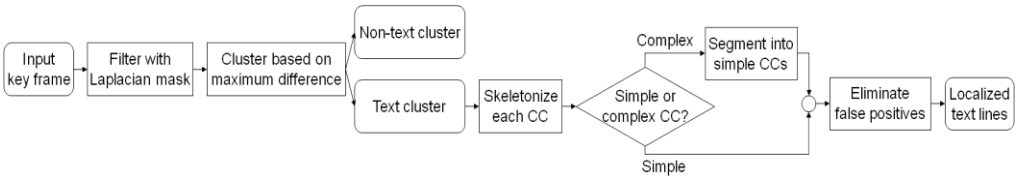



**Figure 3.10. Flowchart of the proposed method.**

#### 3.2.2.1 Text Localization

Because video text can have a very low contrast against complex local backgrounds, it is important to pre-process the input image to highlight the difference between text and non-text regions. Text regions typically have a large number of discontinuities, e.g., the transitions between text and background. Therefore, the input video key frame is converted to grayscale and filtered by a 3 × 3 Laplacian mask to detect the discontinuities in four directions: horizontal, vertical, left diagonal and right diagonal (Figure 3.11).

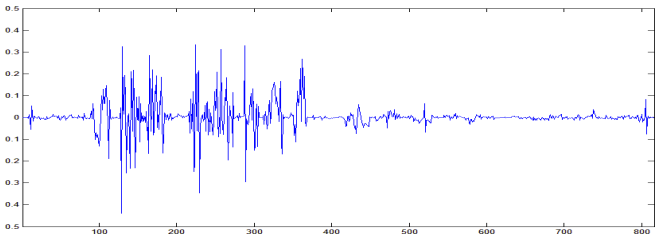
| | | |
|---|---|---|
| 1 | 1 | 1 |
| 1 | -8 | 1 |
| 1 | 1 | 1 |

**Figure 3.11. The 3 × 3 Laplacian mask.**

Since the Laplacian mask produces two values for every edge, text regions have many positive and negative peaks of large magnitudes, and the reverse is true for non-text regions. It is observed that the zero crossings correspond to the transitions between text and background (Figure 3.12).


(a) Input


(b) Laplacian filtered




(c) Profile of the middle row of (b)

**Figure 3.12. Profiles of text and non-text regions. In (c), the x-axis shows the column numbers while the y-axis shows the pixel values.**

Ideally, there should be the same number of text-to-background and background-to-text transitions. This condition, however, does not hold for low contrast text on complex background so we use a weaker condition to ensure that the low contrast text is not missed. Maximum difference (MD) (Wong & Chen 2003), defined as the difference between the maximum value and the minimum value within a local $1 \times N$ window, is computed from the Laplacian-filtered image $g$:

$$
\begin{aligned}
Max(x,y) &= \max_{t \in [-N/2, N/2]} g(x, y-t) \\
Min(x,y) &= \min_{t \in [-N/2, N/2]} g(x, y-t) \\
MD(x,y) &= Max(x,y) - Min(x,y)
\end{aligned}
\tag{3.9}
$$

The MD map is obtained by moving the window over the image (Figure 3.13c). $N$ is empirically determined to be 21.

Text regions typically have larger MD values than non-text regions due to the larger magnitudes of the positive and negative peaks. Therefore, we use K-means to classify all pixels into two clusters, text and non-text, based on the Euclidean distance of MD values. The morphological operation *opening* is used to remove small artifacts (Figure 3.13d).


(a) Input (b) Laplacian filtered (c) MD map (d) Text cluster

**Figure 3.13. The intermediate results of text localization.**

#### 3.2.2.2 Connected Component Classification

Traditionally, bounding boxes are used for displaying the localized text blocks. This is sufficient for horizontal text lines; however, for skewed text lines, rectangular boxes will enclose many unnecessary background pixels. Neighboring skewed text lines will also lead to overlapping bounding boxes. Hence we propose to use CCs for displaying text lines. We further propose using *skeletonization* to segment CCs into separate text lines.

There are two types of CCs: *simple* and *complex*. A simple CC is either a single text string or a false positive. For example, the CCs at the bottom of Figure 3.13d are simple CCs. On the other hand, a complex CC contains multiple text strings which are connected to each other and to false positives in the background. For example, the CC in the middle of Figure 3.13d contains

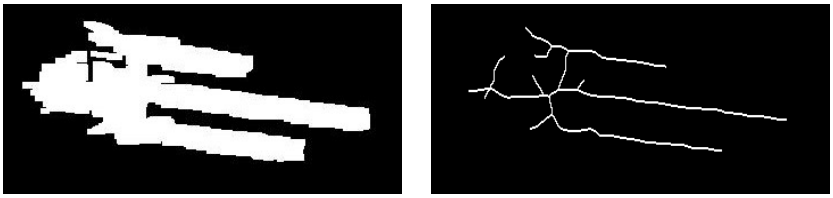
three text strings and a false positive (the logo). High contrast text often appears as simple CCs while low contrast text often appears as complex CCs.

In the first case (simple CCs), the whole component is displayed in the result (if it is a text CC) while in the second case (complex CCs), we want to output only the text part and suppress the non-text part of the CC. In order to do so, we need to segment a complex CC into multiple simple CCs and retain only the text CCs.

The segmentation step will be described in detail later. For now, we discuss how to classify every CC as either simple or complex. Skeleton is a well-defined concept in digital image processing to represent the structure of a region (Figure 3.14). The intersection points (or junction points) of a skeleton show the locations where the sub-components of different orientation are connected to each other. Hence, the rule for CC classification is defined based on the number of intersection points:

$$N\_Intersection(c_i) = \left|Intersection\big(Skeleton(c_i)\big)\right|$$
$$Type(c_i) = \begin{cases} Simple, & N\_Intersection(c_i) = 0 \\ Complex, & otherwise \end{cases} \quad \textbf{(3.10)}$$

$\{c_i\}$ is the set of CCs in the text cluster obtained in the previous step. $Skeleton(.)$ returns the result of skeletonization. $Intersection(.)$ returns the set of intersection points. At the end of this step, simple CCs are retained while complex CCs are sent for segmentation in the next step.



(a) Connected component (b) Skeleton

**Figure 3.14. Skeleton of a connected component from Figure 3.13d.**

#### 3.2.2.3 Connected Component Segmentation

In order to output only the text part of a complex CC, we need to segment, or split, it into multiple simple CCs based on the intersection points. In Figure 3.15, point A shows the location where the first text line of Figure 3.13a connects to the logo (a false positive). By segmenting the complex CC from A to B, we are able to get back the first text line.

AB is called a skeleton segment, which is defined as a continuous path from an intersection point to either an end point or another intersection point. In addition, the path should not include any other point in the middle. For each skeleton segment, we extract the corresponding sub-component from the complex CC. In Figure 3.16, sub-components 1, 2 and 3 correspond to the first three text lines in Figure 3.13a while sub-components 4 and 5 correspond to non-text regions (they are part of the logo in Figure 3.13a).

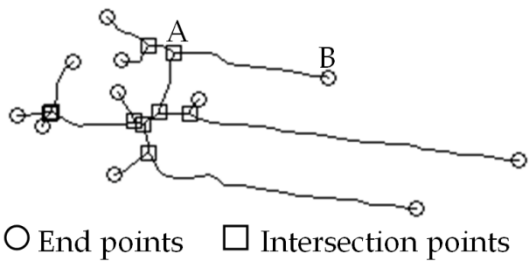



**Figure 3.15. End points and intersection points of Figure 3.14b.**

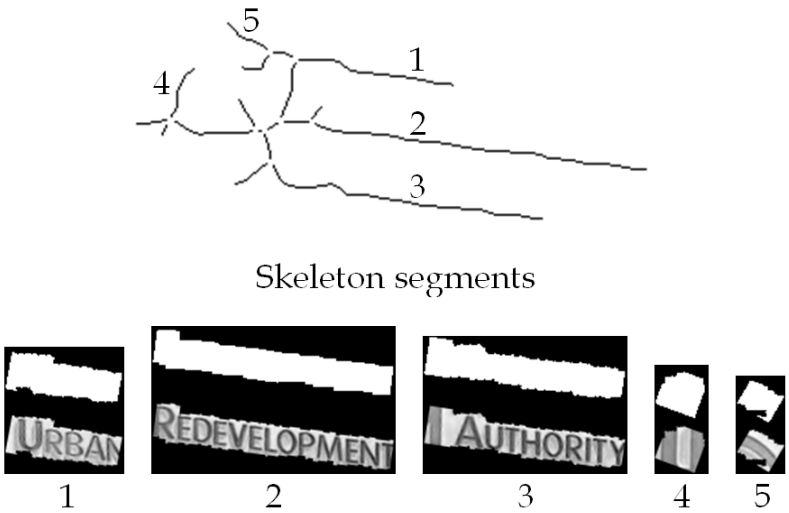



**Figure 3.16. Skeleton segments of Figure 3.14b and their corresponding sub-components. (Only 5 sample sub-components are shown here.)**

To remove false positives (such as sub-components 4 and 5), we propose using text-specific features, which are described in the next section.

#### 3.2.2.4 False Positive Elimination

After the previous step, we have a set of simple CCs, $\{b_i\}$, each of which is either an original simple CC or a new simple CC segmented from a complex CC. $b_i$ is a true text block if:

$$Straightness(b_i) \leq T_1 \wedge Edge\_Density(b_i) \geq T_2 \quad \textbf{(3.11)}$$

The first feature, straightness, comes from the observation that text strings appear on a straight line (our assumption) while false positives can have irregular shapes. It is defined as:

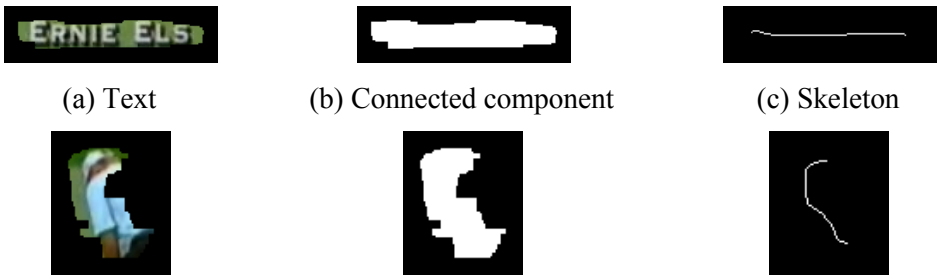
$$s_i = Skeleton(b_i)$$
$$Straightness(b_i) = \frac{Length(s_i)}{End\_Distance(s_i)} \quad \textbf{(3.12)}$$

Note that all $b_i$'s are simple CCs and thus all $s_i$'s have exactly two end points and zero intersection points. For text, $Length(.)$, the length of the skeleton, is close to $End\_Distance(.)$, the straight line distance between the two end points while for non-text, $Length(.)$ is much larger than $End\_Distance(.)$ (Figure 3.17).

The second feature, edge density, is defined as:

$$e_i = Sobel(b_i)$$
$$Edge\_Density(b_i) = \frac{Edge\_Length(e_i)}{CC\_Area(b_i)} \quad \textbf{(3.13)}$$

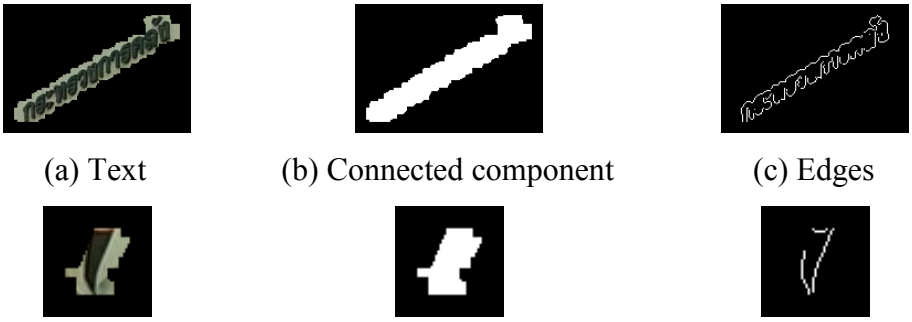
$Sobel(.)$ returns the binary Sobel edge map (for only the white pixels of $b_i$). $Edge\_Length(.)$ is the total length of all the edges in the edge map. $CC\_Area(.)$ is the area of the CC. This feature assumes that edges are denser in text regions than in non-text regions because the former typically contains many text strokes (Figure 3.18). The parameters are empirically determined: $T_1 = 1.2$ and $T_2 = 0.1$.




(a) Text (b) Connected component (c) Skeleton

(d) Non-text (e) Connected component (f) Skeleton

**Figure 3.17. False positive elimination based on skeleton straightness.**



(a) Text (b) Connected component (c) Edges

(d) Non-text (e) Connected component (f) Edges

**Figure 3.18. False positive elimination based on edge density.**

### 3.2.3 Experimental Results

#### 3.2.3.1 Datasets

As there is no standard benchmarking dataset for video text, we selected a variety of video key frames, extracted from news programmes, sports videos and movie clips to form two datasets:

- The horizontal text dataset contained 960 video key frames. The English sub-dataset contained 800 images (652 images for video graphics text and 148 images for video scene text) while the Chinese sub-dataset contained 160 images (153 for video graphics text and 7 for video scene text).
- The non-horizontal text dataset contained 241 video key frames. The English sub-dataset contained 220 images (44 for video graphics text and 176 for video scene text) while the Chinese sub-dataset contained 21 images (4 for video graphics text and 17 for video scene text).

#### 3.2.3.2 Methods for Comparison

We implemented four existing methods. (Liu et al. 2005), denoted as *edge-based method 1*, extracts six statistical features from four Sobel edge maps. (Cai et al. 2002), denoted as *edge-based method 2*, performs Sobel edge detection in the YUV color space and applies two text area enhancement filters. (Wong & Chen 2003), denoted as *gradient-based method*, computes the maximum gradient difference to identify candidate text regions. (Mariano & Kasturi 2000), denoted as *uniform-colored method*, performs hierarchical clustering in the L*a*b* color space to locate uniform-colored text strings.

We chose these four methods because they make use of different features for text localization: edge features (Cai et al. 2002; Liu et al. 2005), gradient features (Wong & Chen 2003) and color features (Mariano & Kasturi 2000). Another reason was that they are all unsupervised methods and thus, no training data were required. On the other hand, supervised methods often require a large number of positive and negative samples. It is especially hard to ensure that the negative samples are representative (Kim et al. 2003).

The parameters of the existing methods were set according to the respective papers. The same parameter values were used for all the experiments.

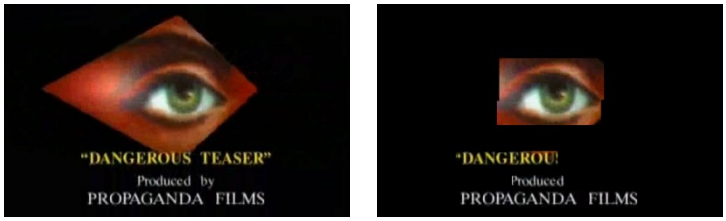
#### 3.2.3.3 Performance Measures

We evaluated the performance at the text line level, which is a common granularity level in the literature (Mariano & Kasturi 2000; Cai et al. 2002; Wong & Chen 2003; Chen et al. 2004b; Ye et al. 2005). For each video key frame in the dataset, we manually counted the number of *Actual Text Blocks*

(*ATB*). The following categories were defined for each localized block by a method.

- *Falsely Localized Block* (*FLB*): A block that does not contain text.
- *Truly Localized Block* (*TLB*): A block that contains at least one true character.
- *Partially Localized Block* (*PLB*): A TLB that misses some characters of a text line. In other words, PLB is the subset of TLB that only enclose the text lines partially.

For example, the numbers of the different types of blocks in Figure 3.19 are 3 ATBs (3 text lines), 3 TLBs (all lines are localized), 2 PLBs (the first two lines are only partially localized) and 1 FLB (the eye).




(a) Input (b) Localized text blocks

**Figure 3.19. Sample ATBs, TLBs, FLBs and PLBs.**

The performance measures were defined as follows:

- *Recall (R)* = TLB / ATB
- *Precision (P)* = TLB / (TLB + FLB)
- *F-measure (F)* = 2 × P × R / (P + R)
- Partial Localization Rate (PLR) = PLB / TLB

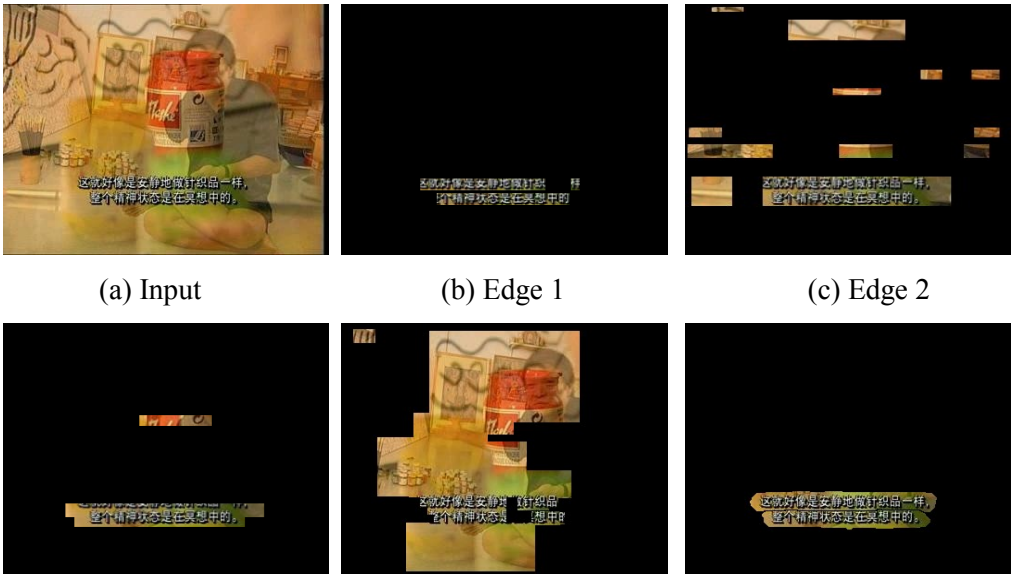
Our definition of Recall is more forgiving than the traditional definition because it considers both fully and partially localized text lines. Due to the challenges of video scene text and the arbitrary orientation of the text lines, it

is difficult for a method to always enclose a full text line in a block. Sometimes it misses some characters of very low contrast and localizes only parts of a line. Since the goal of this work is text localization (how well a method locates potential text blocks), partial detection is still acceptable because it shows that a method is able to detect the presence of text (albeit partially). Having said that, we also included PLR as a performance measure and provide discussion on partial localization in all experiments to ensure a fair comparative study.

#### 3.2.3.4 Experiment on Horizontal Text

In this experiment, we used the horizontal text dataset described earlier. Figure 3.20 shows a sample image with two horizontal Chinese text lines on a complex background. The edge-based method 1 misses some characters of the first line. The edge-based method 2 and the uniform-colored method produce many false positives, while the gradient-based method fails to localize the first line. The proposed method is the only one that fully localizes and separates the text lines from each other, without any false positives.

Table 3.3 shows the performance of the four existing methods and the proposed method on the horizontal text dataset. The proposed method had the highest recall, the second highest precision (almost the same as that of the gradient-based method) and the highest F-measure. This shows the advantage of the proposed method because it achieved good results (including high precision) while making fewer assumptions about text. By assuming that text has horizontal orientation, the existing methods can remove false positives more easily, e.g., by using projection profile analysis.




(a) Input (b) Edge 1 (c) Edge 2

(d) Gradient (e) Color (f) Proposed

**Figure 3.20. The localized blocks of the four existing methods and the proposed method for a horizontal text image.**

The drawback of the proposed method is PLR, which was not as good as those of the gradient-based method and the edge-based method 2. This drawback will be discussed in detail in the next section.

Table 3.3 also shows that the proposed method works slightly better for English text than for Chinese text. The latter has more complicated strokes and thus, for text lines of small font sizes, it is difficult to distinguish the strokes from the complex backgrounds. In addition, the spaces between Chinese characters can be larger than those of English characters, which leads to more partial localization.

**Table 3.3. Experimental results on horizontal text.**

| Method | English | | | | Chinese | | | |
|---|---|---|---|---|---|---|---|---|
| | R | P | F | PLR | R | P | F | PLR |
| Edge 1 | 0.58 | 0.68 | 0.63 | 0.22 | **0.79** | 0.63 | 0.70 | 0.43 |
| Edge 2 | 0.58 | 0.39 | 0.47 | 0.12 | 0.61 | 0.36 | 0.45 | 0.18 |
| Gradient | 0.66 | **0.83** | 0.74 | **0.03** | 0.69 | **0.76** | 0.72 | **0.10** |
| Color | 0.55 | 0.45 | 0.50 | 0.35 | 0.69 | 0.51 | 0.59 | 0.56 |
| Proposed | **0.86** | 0.82 | **0.84** | 0.13 | **0.79** | 0.75 | **0.77** | 0.23 |

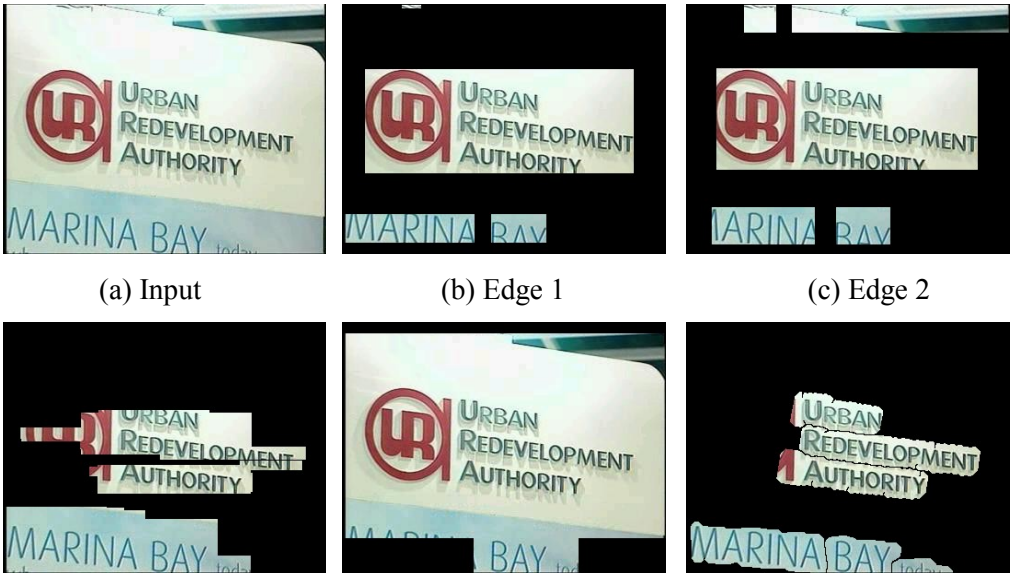
#### 3.2.3.5 Experiment on Non-horizontal Text

The non-horizontal text dataset (described in Section 3.2.3.1) was used for this experiment. Figure 3.21 shows a sample image which has four non-horizontal text strings. Although the existing methods are able to localize the three text strings in the middle, they all fail to separate them because the (non-rotated) rectangular bounding boxes of the skewed text lines overlap with each other. On the other hand, the proposed method localizes all the text strings correctly without any false positives.

Figure 3.22 shows more sample results where the proposed method works for multi-oriented texts of different font sizes.

It is clear from Figure 3.21 that the existing methods are not designed for non-horizontal text. The gradient-based method and the uniform-colored method work on a row by row basis. Similarly, the edge-based method 2 employs projection profile analysis in the horizontal and vertical direction to localize the text blocks. The only method that is easy to extend to multi-oriented text is the edge-based method 1. This method works based on the text cluster (similar to Figure 3.13d) and thus, all the subsequent steps of the proposed method can be applied for this method. Even then, the quality of the text cluster, e.g., whether low contrast text lines are included, will make a difference. According to the experimental results in the previous section, the proposed method outperformed the edge-based method 1, which implies that the text cluster of the former is better than that of the latter.

Since the existing methods did not produce satisfactory results for multi-oriented text, we considered only the proposed method in this experiment. Even though this dataset was more difficult than the previous one because it




(a) Input (b) Edge 1 (c) Edge 2

(d) Gradient (e) Color (f) Proposed

**Figure 3.21. The localized blocks of the four existing methods and the proposed method for a non-horizontal text image.**


(a) Input key frames

(b) Localized blocks

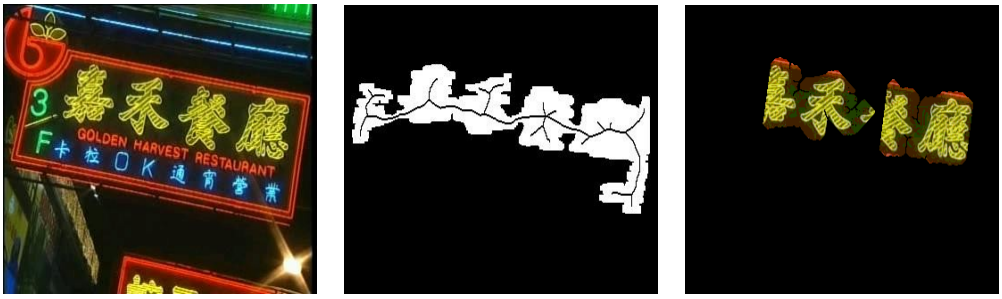
**Figure 3.22. Results of the proposed method for non-horizontal text.**

contained more video scene texts, the proposed method was still able to achieve similar F-measures on both the English and Chinese sub-datasets (Table 3.4). These results show that the proposed method can handle multi-oriented video scene text well.

Similar to previous experiment, the drawback of the proposed method is the high PLR due to the CC segmentation step. The intention of this step is to segment a whole text string into a simple CC. Nevertheless, sometimes a string is split into multiple CCs. If one of them does not satisfy the false positive elimination rule, only part of the string is extracted (Figure 3.23). This happens more often for Chinese text than English text because the former has larger spaces between the characters.

**Table 3.4. Experimental results on non-horizontal text.**

| Method | English | | | | Chinese | | | |
|---|---|---|---|---|---|---|---|---|
| | R | P | F | PLR | R | P | F | PLR |
| Proposed | 0.85 | 0.77 | 0.81 | 0.14 | 0.81 | 0.74 | 0.77 | 0.35 |




(a) Input (b) A CC and its skeleton (c) Extracted pixels

**Figure 3.23. The CC segmentation step may split a text line into multiple parts. For clarity, (b) and (c) only show the corresponding results of the largest Chinese text line, although the English text line is also localized.**

### 3.2.3.6 Experiment on Processing Time

Table 3.5 shows the average processing time of the proposed method and the existing methods for 256 × 256 video key frames on a Core 2 Duo 2.0 GHz machine. For the existing methods, the processing time is reported only for horizontal text because they were not included in the experiment on non-horizontal text. The proposed method was slower than the gradient-based method, slightly slower than the edge-based method 2 but much faster than the

edge-based method 1 and the uniform-colored method. The proposed method also took longer to localize non-horizontal text because more time was required to segment complex CCs into simple CCs.

**Table 3.5. Average processing time (in seconds).**

| Method | Horizontal Text | Non-horizontal Text |
|---|---|---|
| Edge 1 | 22.1 | N.A |
| Edge 2 | 6.1 | N.A |
| Gradient | **1.1** | N.A |
| Color | 13.9 | N.A |
| Proposed | 7.8 | 10.3 |

One of the main contributions of this work is the use of skeletonization to segment a complex CC into constituent parts and separate the connected text lines from each other. It allows our method to localize multi-oriented text and thus improves over exiting methods which are typically limited to only horizontal text. In the future, we will study the problem of partial localization, especially for Chinese text. For example, the edge map could be used to verify the intersection points found by skeletonization.

## 3.3 Summary

In this chapter, we have presented two methods for text localization, one for natural scenes and the other for video key frames. The main contribution of the first work is the use of gap, i.e., inter-character, features for text localization. This direction has not been explored by previous methods. Experimental results on two public datasets demonstrate the effectiveness of the proposed gap features.

The second work uses skeletonization to localize multi-oriented video text. Thus, it has relaxed the horizontal text assumption of many existing methods. Experimentally, the proposed method performs well on both English and Chinese texts.

# Chapter 4

# Single-frame and Multiple-frame Text Enhancement

After text lines have been localized using our methods in the previous chapter, they need to be enhanced prior to recognition. This chapter presents two methods for text enhancement, one for single-frame text and the other for multiple-frame text. In the first method, instead of binarizing a whole text line, we first segment it into individual characters and then binarize each of them individually. In this way, the parameters of the binarization algorithm can be set adaptively according to the local background of each character to produce a better binarized text image. In the second method, given a localized word in a video key frame, we track it both forward and backward in time to identify its first frame and last frame of occurrence. After that, all the text instances within the word's framespan are integrated to derive the final binarized text image.

## 4.1 Single-frame Enhancement

As mentioned above, this work aims to binarize each character in a text line individually to achieve a better binarization. Our focus is on character segmentation, i.e., separating the characters in the same text line from each other. For the binarization step, we use an existing method (Su et al. 2010). Hence, the next sections describe the details of our character segmentation

technique. The combination of character segmentation and binarization will be discussed in section 4.1.3.5.

### 4.1.1 Motivation

From our survey in section 2.7.2.1, we have noticed that existing character segmentation methods often have two limitations. First, methods which are originally designed for text in scanned documents typically require a binary image as input. Thus, they are not suitable for scene text and video text. Second, many methods allow only vertical segmentation paths. However, in difficult cases such as touching characters, more flexible paths would be needed to separate the characters from each other.

Hence, to overcome these limitations, we extend the GVF-based symmetry detection technique in section 3.1.2.1 to segment an input text line into individual characters. Our method works directly on grayscale images and is able to produce curved segmentation paths. Therefore, it overcomes both of the limitations of existing methods.

### 4.1.2 Proposed Method

An overview of our approach is shown in Figure 4.1. The input is a cropped text line image (which can be obtained using the text localization methods in sections 3.1 and 3.2). A simple pre-processing step is used: the text line is rotated back to horizontal orientation (if it is non-horizontal) and normalized to a fixed height of 128 pixels (because some text lines are too small to be readable at their original sizes). After that, there are three main

steps: cut candidate identification, minimum cost path finding and false positive elimination. The first step identifies pixels that are potentially part of non-vertical cuts. In the second step, we find multiple least cost paths from the top row to the bottom row of an image. The third step helps to remove false cuts that go through the middle of the characters. The final outputs are the character segmentation paths.

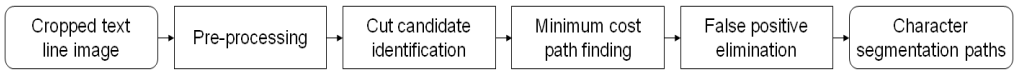



**Figure 4.1. The flowchart of the proposed method.**

#### 4.1.2.1 Cut Candidate Identification

We modify the GVF-based symmetry detection technique in section 3.1.2.1 for the purpose of character segmentation. Within a gap between two consecutive characters, there is more than one segmentation path that can separate the two characters. One way to define a good path is that it should stay as far as possible from the two character edges to allow room for errors in case the edge information is not accurate or the character contours are partly broken due to low contrast.

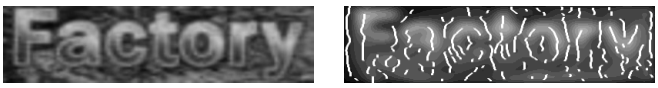
The symmetry detection technique in section 3.1.2.1 identifies points that are equally far from the two character edges. Hence, these points satisfy the criterion of a good path as mentioned above. We define a *candidate cut pixel* $(x, y)$ as a pixel that satisfies Equation (3.2) in section 3.1.2.1. For the reader's convenience, this equation is reproduced below:

$$\begin{cases} u(x,y) < 0 \\ u(x+1,y) > 0 \\ angle\big(g(x,y), g(x+1,y)\big) > \theta_{min} \end{cases} \tag{4.1}$$

where $g(x,y) = \big(u(x,y), v(x,y)\big)$ is the GVF field and $angle(.)$ returns the angle between two vectors.

Figure 4.2 shows the detected candidate cut pixels of a text line with complex background. GVF is able to detect pixels in the gaps between consecutive characters. Although these pixels do not form complete cuts yet, they play an important role in the path finding process, which is described in the next step, where the segmentation paths are encouraged to go through these pixels instead of other pixels in the same gap.

A side effect of Equation (4.1) is that it also captures "medial" (or intra-character) pixels, i.e., those that are in the middle of the character strokes (Figure 4.2). However, it is still possible to distinguish between candidate cut pixels and medial pixels. Since medial pixels are part of a character, if a segmentation path wants to go through these pixels, it has to make several background-to-character and character-to-background transitions. This is not the case for candidate cut pixels because the path would only stay in the background.




(a) Input (b) Candidate cut pixels

**Figure 4.2. Candidate cut pixels of a sample image. In (b), the image is blurred to make the (white) cut pixels more visible.**

We would like to mention that the use of GVF in this work is different from that in section 3.1.2.1 in the following ways:

- Section 3.1.2.1 uses the intra-character and inter-character symmetries in four directions (vertical, horizontal, left-diagonal and right-diagonal) to localize the text lines. However, in this work, due to the nature of the segmentation task, we only focus on **the inter-character symmetry**, i.e., the gaps between consecutive characters, **in the vertical direction**. The rationale for using only the inter-character symmetry and only the vertical direction is that we are mainly interested in symmetry components that can separate two consecutive characters from each other[4].
- Section 3.1.2.1 only deals with scene text (which typically has sufficient resolution) and thus it is reasonable to assume that the intra-character/inter-character symmetry of the same character/gap often form connected components. On the other hand, this work also handles video text. Due to the low resolution of video text, the inter-character symmetry component within a single gap may no longer be a single connected component. Instead, it may be broken into multiple disconnected components (Figure 4.2b). Hence, to form a complete cut from top to bottom, we propose a minimum cost path finding algorithm (which is described in the next section).

[4] And it is assumed that multi-oriented text lines have already been rotated back to the horizontal orientation after the pre-processing step.

#### 4.1.2.2 Minimum Cost Path Finding

Inspired by a method for segmenting merged characters in document images (Wang & Jean 1993), we formulate the character segmentation problem as a minimum cost path finding problem where from the top row, it costs less to go through a gap and reach the bottom row than cutting through the middle of a character.

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

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

Initialization:

$$d(p) = \begin{cases} 0, & if\ p = p_0 \\ +\infty, & otherwise \end{cases} \tag{4.2}$$

Update rule:

$$d(p) = min \begin{cases} d(p_{left-up}) + c(p_{left-up}, p) \\ d(p_{up}) + c(p_{up}, p) \\ d(p_{right-up}) + c(p_{right-up}, p) \end{cases} \tag{4.3}$$

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

$$c(p_1, p_2) = \begin{cases} 0 & if\ candidate(p_2) \\ |I(p_1) - I(p_2)|^2 & if\ p_1.x = p_2.x \\ k \times |I(p_1) - I(p_2)|^2 & otherwise \end{cases} \quad \textbf{(4.4)}$$

where $candidate(p)$ returns true if $p$ is a candidate cut pixel and $k$ is the diagonal move penalty (to be explained later). (Recall that candidate cut pixels are defined in Equation (4.1).)

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

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

Note that the above algorithm finds the best path for only one starting point on the top row. To segment all the characters, we run it multiple times with different starting points. Ideally, we only need to put a starting point

every $w$ pixels where $w$ is the estimated character width (based on the height of the input image). However, because the characters have variable widths, e.g., 'i' versus 'm', and furthermore, the gaps between the words may not be a multiple of $w$, more frequent starting points are required. In our implementation, a starting point is placed every $w/4$ pixels.

#### 4.1.2.3 False Positive Elimination

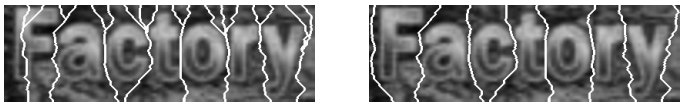
In the previous step, the cost function is carefully designed to discourage segmentation paths that cut through the characters. However, these false cuts may still occur for various reasons, e.g., low contrast which leads to a small difference in intensity values of consecutive pixels on the path. In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

It is interesting to observe that if there are more starting points than required in a gap, the minimum cost paths usually converge to the same end point (Figure 4.3a). This suggests that end points are more reliable than the starting points, especially because the latter are placed according to a heuristic rule based on the estimated character width.

In order to verify whether a segmentation path is a true cut or a false cut (going through a character), we perform backward path finding from the end points to the top row (similar to forward path finding, except that the directions of the edges are reversed). For true cuts, it is likely that the forward path and the backward path are close to each other because they both aim to pass through the candidate cut pixels in the background. However, for false cuts, instead of going the same route as the forward path, the backward path may switch to either side of the character because the cost would be lower

since there are no transitions between the character and the background (Figure 4.3b).

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




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

**Figure 4.3. Two-pass path finding algorithm. In (a), different starting points converge to the same end points. In (b), the false cuts going 'F' have been removed while the true cuts are retained.**

### 4.1.3 Experimental Results

#### 4.1.3.1 Datasets

We used our text localization method in section 3.2 to extract a variety of text lines from TRECVID videos[5], including news programmes, commercials and movie clips. The text lines were divided into 4 datasets: English horizontal (200 images), English non-horizontal (100 images), Chinese horizontal (200 images) and Chinese non-horizontal (100 images). The horizontal datasets mostly contained video graphics text while the non-horizontal datasets mostly contained video scene text.

[5] http://trecvid.nist.gov/

#### 4.1.3.2 Methods for Comparison

For comparison purpose, we implemented two existing methods: (Huang et al. 2009), denoted as *Huang's method*, and (Kopf et al. 2005), denoted as *Kopf's method*. Huang's method binarizes the gradient map of a text image and performs projection profile analysis to locate potential vertical cuts. Heuristic rules are used to merge the segmented regions based on assumptions about a character's height and width. Kopf's method performs minimum cost path finding and uses a similar graph structure as our method but with a different cost function. It makes a simple modification from document analysis methods by using the absolute difference in grayscale intensities between consecutive pixels on a path. On the other hand, our method defines the cost function based on GVF and also employs backward path verification.

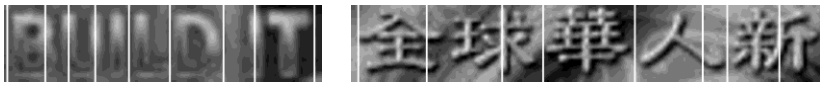
#### 4.1.3.3 Sample Segmentation Results

Figure 4.4 shows sample segmentation results of the existing methods and the proposed method. The image on the left hand side contains English characters of very low contrast. Huang's method misses 3 cuts (between 'B' and 'U', between 'U' and 'I' and between 'I' and 'T') and produces 4 false cuts. Kopf's method has the same number of false cuts but reduces the number of missing true cuts to 2 (between 'U' and 'I', and between 'I' and 'T'). The proposed method is the only one that identifies all the cuts correctly, without any false cuts.

In the image on the right hand side, the Chinese characters are also of low contrast. The proposed method detects all the cuts correctly while both

existing methods miss one cut (Huang's method misses the cut between the first and the second characters and Kopf's method misses the cut between the second and the third characters). Similar to the previous image, both existing methods produce many more false cuts than the proposed method (4 (Huang's method) and 6 (Kopf's method) versus 2 (our method)).

Figure 4.5 shows more results of the proposed method.




(a) Segmentation results by Huang's method

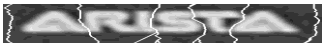

(b) Segmentation results by Kopf's method


(c) Segmentation results by our method

**Figure 4.4. Results of the existing methods and the proposed method.**


(a) Frame with video scene text


(b) Result for a line in (a)




(c) Result for logo text

**Figure 4.5. Results of the proposed method for non-horizontal text (b) and logo text with touching characters (c). In (c), the gap between 'R' and 'I' is missed because the touching part is quite thick.**

#### 4.1.3.4 Segmentation Accuracy

We used Recall (R), Precision (P) and F-measure (F) as the performance measures, and made the following definitions:

- *Actual Cuts* (AC): Ground truth cuts, which are counted manually.

- *True Cuts* (TC): Detected cuts that only pass through the background region.
- *False Cuts* (FC): Detected cuts that go through the characters.

The performance measures were calculated as follows:

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

Table 4.1 shows the performance of the existing methods and our method on English horizontal and non-horizontal text lines. Huang's method did not perform as well as the other two methods because the threshold values used in projection profile analysis do not generalize well to images of different contrast. Although the remaining two methods had similar recall, the proposed method had significantly higher precision and F-measure. Kopf's method produces many false cuts for images with complex background. On the other hand, by using GVF and backward path verification, the proposed method is able to stay as far as possible from the character edges (to allow room for errors) and remove the majority of the false cuts.

Similarly, for Chinese horizontal and non-horizontal text lines, our method achieved higher precision and F-measure than Kopf's method, although the latter had a slightly higher recall for Chinese horizontal text lines (Table 4.2). Huang's method still did not perform well for these two datasets for reasons mentioned above.

The recall of both Kopf's method and the proposed method increased for Chinese text, compared to English text. The English datasets are more challenging than the Chinese datasets because they have more variety of text

lines, including stylized text used in commercials. Another reason is that Chinese characters have more regular widths than English characters and thus it is easier to detect the gaps.

In terms of precision, all three methods degraded in performance. A Chinese character typically consists of multiple sub-components and furthermore, there are gaps between these components. Therefore, more false cuts were detected.

Similarly, all three methods had a lower precision for non-horizontal text, compared to horizontal text. Multi-oriented text is often stylized video graphics text or video scene text. In both cases, the background is complex; and the contrast is also low in the second case. Hence, the methods are more likely to make mistakes. The degradation in F-measure of Huang's method and the proposed method were, however, less than that of Kopf's method.

**Table 4.1. Segmentation results on English text.**

| Method | English Horizontal | | | English Non-horizontal | | |
|---|---|---|---|---|---|---|
| | Recall | Precision | F | Recall | Precision | F |
| Huang's method | 0.66 | 0.55 | 0.60 | 0.64 | 0.49 | 0.55 |
| Kopf's method | 0.89 | 0.76 | 0.82 | 0.88 | 0.62 | 0.73 |
| Our method | 0.89 | 0.91 | **0.90** | 0.91 | 0.85 | **0.88** |

**Table 4.2. Segmentation results on Chinese text.**

| Method | Chinese Horizontal | | | Chinese Non-horizontal | | |
|---|---|---|---|---|---|---|
| | Recall | Precision | F | Recall | Precision | F |
| Huang's method | 0.64 | 0.47 | 0.54 | 0.63 | 0.46 | 0.53 |
| Kopf's method | 0.96 | 0.60 | 0.74 | 0.95 | 0.57 | 0.71 |
| Our method | 0.95 | 0.81 | **0.87** | 0.96 | 0.74 | **0.84** |

#### 4.1.3.5 Recognition Accuracy

To show that character segmentation helps to improve the recognition rate, we used a recent binarization method (Su et al. 2010), denoted as *Su's method*, at two different levels: the text line level and the character level (i.e., the individual characters segmented by the proposed method). This method outperforms traditional methods such as Otsu's method and Niblack's method on the dataset of the Document Image Binarization Contest 2009 (Gatos et al. 2009).

In this experiment, we considered only English text lines. The performance measure was the character recognition rate (CRR) using Tesseract, Google's open source OCR engine. Moreover, to ensure a fair comparison, for the character level, we put the binarized results together into a line so that the OCR engine could utilize its language model to better recognize the characters.

Figure 4.6 shows the binarization results of Su's method for a challenging image with a complex and uneven background. Without segmentation, the last four characters are not binarized well because the method is not able to handle both characters with clean background and characters with complex background in the same text line. Hence, only the first two characters are recognized correctly. On the other hand, with segmentation, the binarization result is significantly improved because the method is free to choose the appropriate parameter values for each individual character (instead of for the whole text line). As a result, six characters are recognized correctly.

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

**Figure 4.6. Binarization results using Su's method without segmentation (b) and with segmentation (c), together with the recognition results. In (c), both the binarization and recognition results are improved.**

Table 4.3 shows that it is better to perform binarization at the character level than at the text line level. Part-by-part binarization helps to reduce the problem of complex and uneven background by using local information. Hence, the CRR was greatly improved.

In addition, because Otsu's method and Niblack's method are widely used in the literature, their performance on the same dataset is also reported. The combination of the proposed character segmentation method and Su's binarization method gave a higher CRR than both of these methods.

It is also observed that Su's method (without segmentation) was only slightly better than Otsu's method and Niblack's method in terms of CRR, although it outperforms them on a document image dataset, as mentioned above. This can be explained by the fact that the performance on that dataset was measured based on several visual metrics (i.e., not goal-oriented) such as ground truth binary pixels (Su et al. 2010). As discussed in (Wolf & Doermann 2002), visually better binarization results may not necessarily lead to better CRR.

**Table 4.3. Recognition rates on English text.**

| Method | CRR |
|---|---|
| Su's method (line level) | 59.1% |
| Segmentation + Su's method (character level) | **66.6%** |
| Otsu's method | 54.0% |
| Niblack's method | 58.1% |

The main contribution of this work is that GVF is used in a novel way to allow our method to produce curved segmentation paths. Binarizing each character individually leads to better binarization, which in turn improves the recognition accuracy. As mentioned in the experiments, the proposed method has lower precision on Chinese text than on English text because it sometimes produces false gaps between the sub-components of a single Chinese character. This problem will be studied in the future.

## 4.2 Multiple-frame Integration

The single-frame enhancement method in the previous section has not utilized the temporal information. When the input to a text extraction system is a video, it is crucial to exploit the temporal redundancy to improve the recognition accuracy. This section presents a method for multiple-frame integration of video text.

### 4.2.1 Motivation

While there are existing methods for multiple-frame integration (section 2.5.2), they have two drawbacks. First, the end result of many multiple-frame integration methods such as (Li & Doermann 1999; Hua et al. 2002; Lienhart & Wernicke 2002; Zhou et al. 2007; Yi et al. 2009) is an enhanced grayscale image. In other words, these methods need to rely on another binarization method to binarize the enhanced image before sending it to an OCR engine. Second, some methods such as (Hua et al. 2002; Yi et al. 2009) can handle only a fixed text polarity, e.g., bright text on dark background.

Therefore, we propose a multiple-frame integration method to overcome these two drawbacks. The end result of our method is a binarized image, which can be readily fed to an OCR engine. In addition, we have a text polarity detection step, which allows our method to handle both bright text on dark background and dark text on bright background.

### 4.2.2 Proposed Method

We use our text localization method in section 3.2 to extract the text lines from a video key frame. Non-horizontal texts are rotated back to horizontal orientation. We then split the text lines into individual words using the method proposed by (Shivakumara et al. 2011a). The motivation for using words (instead of text lines) for multiple-frame integration is that it is useful to have different framespans for different words. For example, if one word is occluded for a few frames, we can use a shorter framespan for it, while still having longer framespans for the remaining words. In contrast, using text lines means that all the words in the same line need to have the same framespan. Thus, lines are less flexible than words.

Our method requires two inputs: (1) the word bounding box in a reference frame and (2) the frame ID of the reference frame. Most video texts are either static or have linear motion (e.g., right-to-left) (Lienhart 2003). Hence, the scope of this work is limited to these two types of motion. Note that static texts also benefit from multiple-frame integration if their local backgrounds change between the frames. Our method has three main steps: text instance identification, text probability estimation and character shape refinement.

#### 4.2.2.1 Identification of Text Instances

We use the term *word instance* to refer to the appearance of a word in a particular frame. Let $t_{ref}$ be the frame ID of the reference word instance. This step aims to identify the first frame ($t_{start}$) and the last frame ($t_{end}$) of occurrence of the word. Furthermore, because the word may move between the frames, we need to identify its bounding box in each frame.

This step is important in two aspects. First, we need to ensure that the extracted instances contain the same word as the reference one. Otherwise, the irrelevant instances would negatively affect the integration, e.g., blurring the character edges. Second, we need to extract as many relevant instances as possible because with more information, we can achieve a better integration. We propose to combine SIFT (Lowe 2004) and Stroke Width Transform (SWT) (Epshtein et al. 2010).

##### *4.2.2.1.1 Text Descriptor Extraction*

Due to the low-resolution nature of video words, applying SIFT on them directly will give only a few keypoints, which are not sufficient for robust tracking. Hence, we first use SWT to identify the text pixels, from which we can extract more keypoints and descriptors.

SWT has been described in section 2.3.1 (survey on gradient-based text localization methods). Its main idea is to find pairs of corresponding text pixels. It first computes the Canny edge map. For each edge pixel, it follows that pixel's gradient direction. If it reaches another edge pixel with opposite gradient direction, the two pixels are considered to be a pair of corresponding

text pixels. To handle both bright text on dark background and dark text on bright background, we propose the following rule. We apply SWT on the reference word instance twice, once by following the gradient directions of edge pixels and once by following the inverse gradient directions. Each time we count the number of corresponding pixels. The polarity with more corresponding pixels is selected as the correct polarity. The rationale is that when the polarity is wrong, tracing the ray from an edge pixel will lead us to the background, without finding any matching edge pixel. Thus there will be much fewer corresponding pixels.

After identifying the text polarity, for each pixel in a pair of corresponding pixels, we extract its SIFT descriptor at a fixed scale. (The orientation at the specified location and scale is left to SIFT to determine.) We refer to this scheme as *SWT-SIFT*.

*4.2.2.1.2 Text Tracking*

To track linearly moving texts, we first extract the descriptors of the reference word instance using SWT-SIFT. To identify the end frame, we go from $t_{ref} + 1$ to $NumFrames$ (the number of frames of the video). For each frame under consideration, we slightly extend the word bounding box in the previous frame to form the search area in the current frame. Within this area, we again extract the descriptors using SWT-SIFT. We use the nearest neighbor algorithm (Lowe 2004) to match the two sets of SIFT descriptors, one set from the reference word instance and the other set from the search area in the current frame. To obtain the homography between the two sets of descriptors, we use the RANSAC algorithm (Fischler & Bolles 1981).

Figure 4.7 illustrates the tracking process. (a) and (b) show the keypoints extracted with normal SIFT and with SWT-SIFT. As mentioned before, the latter gives more keypoints. In (c) and (d), the left hand side shows a reference word instance, and the right hand side shows the search area in a few frames later. The colored lines show the matched SIFT descriptors. With normal SIFT, there are no matches, while with SWT-SIFT, there are many matches. This illustrates the advantage of the latter. In (e), we use the estimated homography to project the bounding box coordinates of the reference word instance onto the search area, and obtain the new bounding box position.


(a) Keypoints extracted with normal SIFT

(b) Keypoints extracted with SWT-SIFT


(c) Descriptor matching for normal SIFT keypoints

(d) Descriptor matching for SWT-SIFT keypoints


(e) The projection of the reference bounding box

**Figure 4.7. Text tracking using SIFT. In (c), all keypoints are shown. In (d), for clarity, only matched keypoints are shown.**

The criterion for declaring a frame to be the end frame is based on the number of SIFT descriptor matches that conform to the overall homography. This number is typically much smaller than the number of SIFT descriptors in

the reference instance due to two reasons. First, when the word moves, its local background changes, and we do not expect all the reference keypoints to be matched in the search area of a new frame. Second, RANSAC performs geometric verification to filter out false matches that do not conform to the homography. Hence, the number of matched SIFT descriptors is further reduced. Due to these reasons, we use a conservative threshold of 0.1. That is, if the number of matched SIFT descriptors according to RANSAC is less than 10% of the number of reference SIFT descriptors, the frame before the current frame is declared as the end frame (Algorithm 4.1). We use a similar procedure to identify $t_{start}$, by going backward in time.

**Algorithm 4.1. End frame estimation.**

---

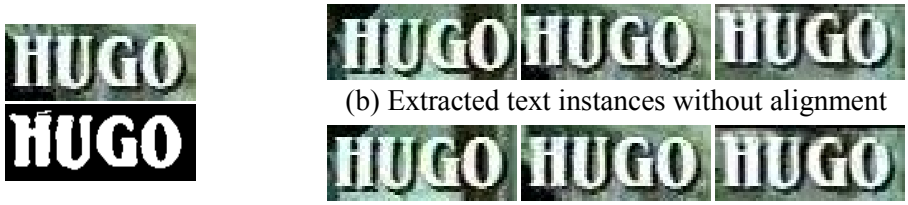
**for** $t = t_{ref} + 1$ to $NumFrames$
  initialize the search area based on the bounding box in frame $t - 1$
  extract text descriptors from the search area using SWT-SIFT
  **if** $matchRatio\left(descriptors_{t_{ref}}, descriptors_t\right) < 0.1$
    $t_{end} = t - 1$
    break
  **else**
    use the homography to estimate the bounding box in the current frame
  **end if**
**end for**
**if** $t_{end}$ is still not set
  $t_{end} = NumFrames$
**end if**

---

#### *4.2.2.1.3 Text Instance Alignment*

The bounding box coordinates returned by the tracking step may be off by a few pixels due to the error in the estimated homography. For temporal integration, we need the text instances to be aligned at pixel level. Thus, we use SWT to estimate the *text mask* as follows. After identifying the pairs of corresponding pixels, we set all the pixels that lie on a ray connecting a pair of pixels to 1, and all other pixels to 0. With this process, we get the text mask of the reference word instance. For each word bounding box estimated by the tracking step, we slightly extend it and also obtain its text mask. Then, we slide the reference text mask over the text mask of the extended bounding box. The position that gives the most number of intersected pixels (counting only pixels with values of 1) is selected as the correct alignment between the current instance and the reference one.

Figure 4.8a shows a reference word instance and its text mask. In (b), we show a few instances of the same word that are tracked in other frames, without alignment. It is observed that the extracted instances are off by a few pixels compared to the reference one. The first instance is shifted in the down-right direction, the second instance is shifted to the left, and the last instance is shifted in the up-left direction. In (c), we show the corresponding instances, but with pixel-level alignment using the text mask in (a). It is evident that alignment helps to "stabilize" the text instances and ensure that the text pixels in the different instances have the same positions. This is crucial for multiple-frame integration.




(a) Reference instance and its text mask

(b) Extracted text instances without alignment

(c) Extracted text instances with alignment

**Figure 4.8. Sample extracted text instances.**

The advantage of our tracking technique over existing methods is that it tracks only text pixels. Thus, it is more robust to background changes than methods that do not attempt to identify the text pixels. For example, (Lienhart & Wernicke 2002; Gllavata et al. 2004; Minetto et al. 2011) perform tracking by extracting features from all pixels in a bounding box, including background pixels. Moreover, SWT is rotation-invariant, and SIFT is robust to rotation, scale change and viewpoint change. Hence, our technique can be extended to track complex text movements.

#### 4.2.2.2 Text Probability Estimation

At the end of the previous step, we have identified all the word instances between $t_{start}$ and $t_{end}$. We also have their text masks. The text probability map is integrated from these masks:

$$TextProb = \frac{1}{N}\sum_{t=t_{start}}^{t_{end}} mask_t \tag{4.5}$$

where $N = t_{end} - t_{start} + 1$ is the number of frames between $t_{start}$ and $t_{end}$. Figure 4.9b shows a sample probability map. It is observed that true text pixels

have higher values (represented by brighter colors) while background pixels have lower values (represented by darker colors). The rationale is that after the text instances have been aligned, the text pixels will be "stable", i.e., they stay at the same position across the different instances. Thus, when we sum up the text masks, these pixels will accumulate high values. On the other hand, the background changes from one word instance to another, and thus will not accumulate high values. Another advantage of the proposed integration is that even if a text part is missed in one or few instances (e.g., due to occlusion or blurring), it will still accumulate high values in the text probability map because it can clearly be seen in the remaining instances.


(a) Word instances and their corresponding SWT masks (b) Text probability map (c) Initial binarization

**Figure 4.9. Text probability estimation.**

We then apply a simple thresholding on the text probability map to obtain an initial binarization of the word:

$$TextBin(x,y) = \begin{cases} 1, & TextProb(x,y) \geq K \\ 0, & otherwise \end{cases} \quad \textbf{(4.6)}$$

$K$ is determined empirically to be 0.7. In other words, it is expected that true text pixels will take a value of 1 (i.e., white pixel) in at least 70% of the text masks. (Section 4.2.3.4 analyzes how the recognition accuracies vary with respect to $K$.)

Figure 4.9 shows a few instances from a word's framespan. Because this is a difficult case, the text masks contain erroneous background information. However, as explained above, these background parts are not stable enough to accumulate high values. In addition, in the last two instances, a large part of 'T' is missing due to the low contrast and the similar background colors. Despite these problems, the text probability map in Figure 4.9b shows that the character shapes are recovered by using information from the other instances. The initial binarization in Figure 4.9c successfully separates text from the background, despite the imperfect individual text masks in Figure 4.9a.

#### 4.2.2.3 Character Shape Refinement

The initial binarization in the previous step may not reflect the true character shapes because SWT tends to produce rounded strokes. In addition, there may be disconnections in the strokes due to complex backgrounds. Thus, in this final step, we refine the character shapes to improve the recognition accuracy.

We slide a window over $I_{mean}$, the averaged intensity image of all the word instances between $t_{start}$ and $t_{end}$. For a window centered at $(x, y)$, we refine the character shapes as follows:

$$TextRefined(x,y) = \begin{cases} 1, & |I_{mean}(x,y) - fg_{mean}| < |I_{mean}(x,y) - bg_{mean}| \\ 0, & otherwise \end{cases} \quad (4.7)$$

$fg_{mean}$ and $bg_{mean}$ are the average intensity values of the foreground and background pixels in the local window. The classification of pixels into

foreground and background is based on the corresponding window in $TextBin$. In other words, the white pixels in the corresponding window in $TextBin$ are assumed to be foreground and their intensity values in $I_{mean}$ are used to estimate $fg_{mean}$. Similarly, for the black pixels in the corresponding window in $TextBin$, their intensity values in $I_{mean}$ are used to estimate $bg_{mean}$. Intuitively, Equation (4.7) means that in each window, if the center pixel is closer to the estimated foreground intensity, it is set to foreground. Otherwise, it is set to background.

The rationale is that within a small neighborhood, text pixels often have similar intensity values. Thus, we can recover true text pixels which may be missed by SWT. In addition, we can suppress background pixels (e.g., those inside the hole of a character) which may have been wrongly picked up by SWT. Thus, this step helps to preserve the distinctive character features (e.g., sharp edges and holes), which are crucial for correct recognition.

In Figure 4.10, each pair of images consists of the results before refinement ($TextBin$) and after refinement ($TextRefined$). The strings below the images are the recognition result by an OCR engine (ABBYY FineReader 9.0). For "10PM", refinement helps to separate touching characters. For "to", before refinement, the left part of the horizontal stroke of 't' is missing. After refinement, it is recovered. And for "Call", before refinement, the hole of 'a' is not clear, which becomes much better after refinement. In all of these cases, the recognition results are wrong before refinement, but become correct after refinement. Note that in Figure 4.8 and Figure 4.9, the text pixels are shown in white and the background is shown in black. However, in Figure 4.10 and the subsequent figures, before sending a binarized word for recognition, we

change the polarity to black text on white background (which is the polarity that the OCR engine expects).

10?M 10PM CO to Coll Call

**Figure 4.10. Character shape refinement.**

### 4.2.3 Experimental Results

#### 4.2.3.1 Datasets

Since there is no standard dataset for video text, we used our text localization method in section 3.2 to extract a variety of video words. The moving text dataset contained words extracted from English videos (from TRECVID 2005 and 2006[6]) and German videos[7]. The static text dataset contained English words extracted from TRECVID 2005 and 2006. Table 4.4 provides the statistics of the datasets.

**Table 4.4. Statistics of the moving text dataset and the static text dataset.**

| | Moving text | Static text |
|---|---|---|
| Frame rate (frames/second) | 30 | 30 |
| Frame resolutions | 352 × 240 to 384 × 288 | 352 × 240 |
| Text motion types | Bottom to top, right to left and left to right | Static |
| Number of words | 250 | 212 |
| Number of characters | 1545 | 1389 |

[6] http://trecvid.nist.gov/
[7] The MoCA Project. http://pi4.informatik.uni-mannheim.de/pi4.data/content/projects/moca/Project-textSegmentationAndRecognition.html

#### 4.2.3.2 Methods for Comparison

For comparison, we implemented three methods:

- Niblack binarization on only the reference instance.
- Min/max operator (used in (Lienhart & Wernicke 2002; Zhou et al. 2007)) on multiple instances, followed by Niblack binarization.
- Combination of average and min/max operators (used in (Yi et al. 2009)) on multiple instances (denoted *Average-Min/max*), followed by Niblack binarization.

The first method is included to show the difference between using only a single instance and using temporal integration. For the other two methods, we need to detect the text polarity to decide whether to use the min operator or the max operator. (Lienhart & Wernicke 2002; Zhou et al. 2007) used heuristic rules while (Yi et al. 2009) did not address this issue. Since we are more interested in the different integration techniques, we used the same polarity detection technique (in section 4.2.2.1) for both methods. Similarly, whenever multiple instances were required, they were identified using the same technique in section 4.2.2.1.

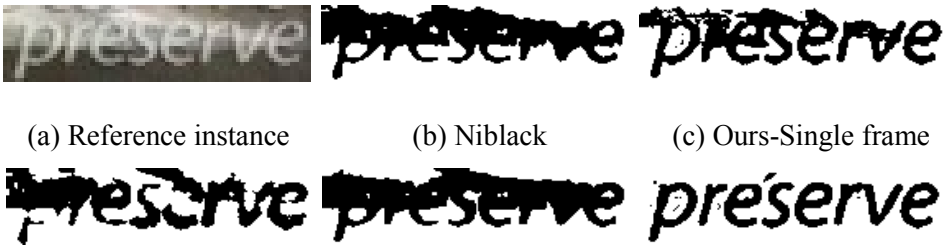
For our method, we ran it with two different settings:

- On only the reference instance (*Ours-Reference instance*). In this setting, no temporal integration was done, i.e., $TextBin$ is equal to the text mask of the reference instance. We still performed character shape refinement.
- On multiple instances (*Ours-Multiple instances*).

#### 4.2.3.3 Sample Results

Figure 4.11 shows sample results on a word affected by lighting (special effect). The proposed method on multiple instances is the only one that produces a good binarization. The recognition result by the OCR engine is also correct ("preserve"). For all the other methods, the OCR engine returns an empty string because the binarized results contain a lot of background information.

Figure 4.12 shows more results of our method. Note the fact that the last image has a different text polarity than the rest. This shows that the polarity detection technique in section 4.2.2.1 allows our method to handle different text polarities.



(a) Reference instance (b) Niblack (c) Ours-Single frame



(d) Min/max (e) Average-Min/max (f) Ours-Multiple frames

**Figure 4.11. Sample results of the existing methods and our method. For Min/max and Average-Min/max, only the final binarized images are shown.**

stryker INFLUENTIALS

only ET GLOBAL

**Figure 4.12. Sample results of our method. The left image in each pair is the reference instance. The strings below the images are the OCR results.**

#### 4.2.3.4 Recognition Accuracy

To examine whether temporal integration helps to improve the recognition rate, we used ABBYY FineReader 9.0 as the OCR engine. The performance measures were the **case-sensitive** character recognition rate (CRR) and word recognition rate (WRR). Following (Ntirogiannis et al. 2011), CRR and WRR were computed by comparing the recognized strings and the ground truth (GT) strings. (Characters such as punctuation marks are ignored.)

$$CRR = \frac{\#\ correct\ characters}{\#\ GT\ characters} \tag{4.8}$$

$$WRR = \frac{\#\ correct\ words}{\#\ GT\ words} \tag{4.9}$$

where $\#\ correct\ characters = \#\ GT\ characters - (\#\ insertions + \#\ substitutions + \#\ deletions)$. Insertions, substitutions and deletions are between the recognized strings and the GT strings.

Table 4.5 shows the recognition rates on the moving and static text datasets, respectively. In each table, the first two methods only use the reference instance while the remaining ones use multiple instances. The first observation is that using multiple instances leads to significantly better CRRs and WRRs. This demonstrates the importance of temporal integration.

Between the two single-instance methods, our method achieved better CRR and WRR than Niblack, especially on the static text dataset. The advantage of using SWT to estimate text masks is that it searches for a text-specific feature, namely pairs of edges with constant stroke widths. In contrast,

Niblack does not analyze the edge structures and only works on the intensity values. Thus, it may produce noise for cluttered backgrounds.

Among the multiple-instance methods, our method achieved the best CRR and WRR on both datasets. In terms of WRR, our method was 14.8% and 13.2% better than Average-Min/max on the moving text and static text datasets, respectively. This demonstrates the advantage of our method.

Between the two existing multiple-instance methods, Average-Min/max was better than Min/max. The latter had a significant drop in accuracies for moving texts. The reason is that for moving texts, the alignment of the text instances may not be perfect. If an instance is misaligned, it will introduce "outlier" values (intensity values that are significantly higher or lower than the corresponding values in other instances). Min/max is sensitive to such outlier values. In contrast, Average-Min/max only uses the min/max operator on background pixels, and thus is less affected by outlier values. However, its drawback is the use of Otsu's binarization for estimating text and background pixels. This binarization method is not able to handle video words with complex backgrounds.

To examine the effectiveness of character shape refinement, we turned it off in the last row of Table 4.5. The results show that refinement helps to improve the WRRs on the moving text and the static text datasets by 3.6% and 6.6%, respectively.

We also examined how the recognition rates of our method change with respect to parameter $K$ (in Equation (4.6)). Figure 4.13 shows that the WRRs were not too sensitive to $K$ as long as $K \in [0.6, 0.8]$.

**Table 4.5. Recognition rates on the moving text dataset and the static text dataset (in %).**

| Method | Moving text | | Static text | |
|---|---|---|---|---|
| | CRR | WRR | CRR | WRR |
| Niblack | 68.2 | 51.2 | 54.6 | 32.1 |
| Ours-Reference instance | 70.7 | 52.8 | 65.4 | 47.6 |
| Min/max | 49.9 | 26.8 | 63.2 | 43.9 |
| Average-Min/max | 74.0 | 55.6 | 73.0 | 58.0 |
| Ours-Multiple instances | **82.8** | **70.4** | **80.9** | **71.2** |
| Ours-Multiple instances, without shape refinement | 81.9 | 66.8 | 78.6 | 64.6 |

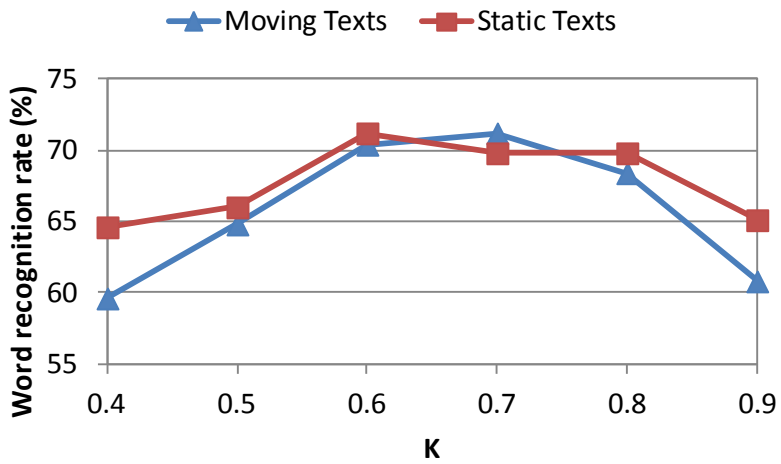



**Figure 4.13. Word recognition rates of our method for different values of *K*.**

This work has shown the importance of text pixel identification prior to enhancement, in both the tracking step and the integration step. It allows our method to track only the text pixels and enhance only the text regions. In contrast, many existing methods use all pixels for enhancement, which may result in accidental enhancement of the background regions. Currently, the proposed method works for static text and moving text with linear motion (e.g., right-to-left). In the future, we will study more complex text movements.

## 4.3 Summary

This chapter has described two methods for text enhancement, one for single-frame enhancement and the other for multiple-frame enhancement. The

main contribution of the first work is that GVF is used in a novel way to produce curved segmentation paths. This in turn allows our method to binarize each character individually (instead of binarizing a whole text line) and leads to improved recognition accuracy.

The second work has shown the importance of utilizing the temporal redundancy to achieve significantly better recognition accuracy. Our work has also demonstrated the advantages of text pixel identification prior to enhancement. It allows our method to track and enhance only the text regions. In contrast, many previous methods utilize all pixels for enhancement, which may lead to accidental enhancement of the background regions.

# Chapter 5

# Recognition of Scene Text with Perspective Distortion

The previous chapter has shown that text enhancement methods can be used together with an OCR engine (as a black box) to improve the recognition accuracy. While it is a reasonable choice for texts which are frontal parallel to the image plane, this approach is not suitable for scene texts with perspective distortions because OCR engines are not designed to handle these distortions. Hence, we have explored a different approach in which we propose our own algorithms for character and word recognition.

This chapter presents our work on scene text recognition. We focus on texts with perspective distortions, which is an improvement over many existing methods which handle only frontal texts.

## 5.1 Motivation

As mentioned in section 2.7, although there are existing works to recognize text in natural scene images, e.g., (Smith et al. 2011; Wang et al. 2011; Novikova et al. 2012; Mishra et al. 2012b), their scopes are limited to horizontal texts which are frontal parallel to the image plane. However, in practice, scene texts can appear in any orientation, and with perspective distortion. Thus, the important issue of handling perspective texts has been neglected by previous works.

This work attempts to address the recognition of perspective texts in street images, which facilitates the application of business name search on online maps (Wang et al. 2011). This application is motivated by the availability of ground-level, 360° views of various locations on Google Maps and Microsoft Bing Maps. These geo-tagged images contain useful text information such as business names, addresses, operating hours and so on. The large-scale nature of street image data provides an exciting opportunity to benefit millions of users.

Using a traditional visual feature such as Histogram of Oriented Gradients (HOG) (as employed in (Wang et al. 2011; Mishra et al. 2012b)) would lead to a low accuracy on perspective texts. The reason is that the feature is not able to handle the different poses of the characters. To deal with this problem, one approach is to train a classifier on discretized poses of individual characters. However, the major drawback of this approach is that it is labor-intensive and time-consuming to collect enough training samples for a large number of character classes (62 classes for English characters and digits), each with, say, 10 discrete poses. In addition, when collecting character samples from natural scenes, it is very difficult to control the poses of the characters accurately.

Hence, we take a different approach and use SIFT in a bag-of-keypoints approach. Because SIFT is robust to both rotation and viewpoint change, our system is trained on **only frontal characters** (from commonly used datasets such as ICDAR 2003 (Lucas et al. 2003)). Our extensive experiments show that this approach achieves good accuracies, while avoiding the high cost of collecting samples of perspective characters.

Following recent works (Wang et al. 2011; Mishra et al. 2012b), the scope of this work is limited to cropped word recognition with a lexicon, i.e., a list of words of interest. The lexicon serves as a form of context information, and is especially relevant for the application of business name search. Given a street image and its address, the lexicon can be built by collecting the shop names around the address via a search engine (Wang et al. 2011). There are also other applications where such a lexicon is available. (Ballan et al. 2010) used a list of soccer players' names for text recognition in sports videos. (Graves et al. 2009) constructed a list of the most common English words for handwriting recognition. Another example is the list of products in a supermarket, which can be used for the application of aiding the visually-impaired. Figure 5.1 illustrates the problem setting.

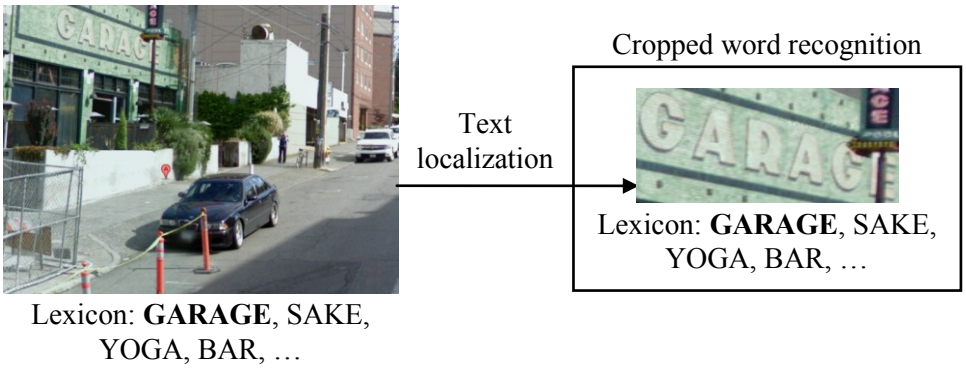



**Figure 5.1. The problem of cropped word recognition. A "cropped word" refers to the region cropped from the original image based on the word bounding box returned by a text localization method. Given a cropped word image, the task is to recognize the word using the provided lexicon.**

Our contributions are as follows. (1) We present an approach to recognize perspective scene texts. This issue is of great practical importance, but has been neglected by most previous works. (2) Our system is trained on only frontal characters, which drastically reduces the cost of collecting

training data. (3) For performance evaluation, we introduce a new dataset called StreetViewText-Perspective, which contains texts in street images with a variety of viewpoints. On this dataset, our method compares favorably to the state-of-the-art.

## 5.2 Proposed Method

An overview of our approach to perspective text recognition is shown in Figure 5.2. We describe the detection and recognition of characters below. The optimized alignment of the recognized characters with the lexicon will be discussed in section 5.2.2.

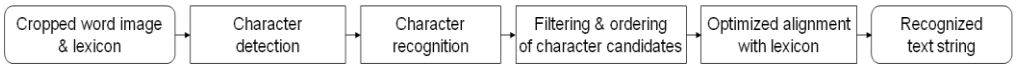



**Figure 5.2. The flowchart of the proposed method.**

### 5.2.1 Character Detection and Recognition

#### 5.2.1.1 Detection of Character Candidates

In the first step, we use Maximally Stable Extremal Regions (MSERs) (Matas et al. 2002) to detect the potential character locations in a cropped word image (hereafter referred to as *character candidates*). The main idea of MSER is to identify regions which remain stable over a range of thresholds on the intensity values. It has been shown that scene characters can be extracted as MSERs (Neumann & Matas 2010; Neumann & Matas 2012). MSERs are also robust to viewpoint change (Mikolajczyk et al. 2005). Hence, they are suitable for perspective characters.

However, not all the extracted MSERs from a cropped word correspond to characters. Thus, we classify them into text MSERs and non-text MSERs using four features: relative height, aspect ratio, number of holes and number of horizontal crossings (Neumann & Matas 2010; Neumann & Matas 2012). The text MSERs are retained while the non-text MSERs are discarded.

In (Neumann & Matas 2010; Neumann & Matas 2012), the text MSERs were directly used for text localization. However, in this work, we use the bounding boxes of the MSERs instead. The reason is that although MSERs provide a useful initial segmentation of the characters, they do not always correspond to the whole characters. Figure 5.3 shows an example where the MSERs corresponding to 'E' and 'S' have incomplete shapes. Therefore, using the MSER bounding boxes as character candidates helps to recover some of the missing parts (if any) of the characters.


(a) Cropped word image (b) MSERs

(c) Character candidates based on MSER bounding boxes

**Figure 5.3. Character detection based on MSERs. For better illustration, only the non-overlapping MSERs are shown in (b). The handling of overlapping MSERs will be discussed later.**

#### 5.2.1.2 Estimation of Character Probabilities

For each character candidate detected in the previous section, we need to estimate the probability that it takes character label $l$. In this work, we focus on English characters: $l \in L = \{A, \dots, Z, a, \dots, z, 0, \dots, 9\}$. Formally, we would

like to estimate $P(l|c_i)$, the probability that $c_i$, the $i^{\text{th}}$ character candidate, takes label $l$.

As mentioned before, our goal is to train the system on only frontal characters (to reduce the cost of collecting training data). This requires the features extracted from the character candidates to be robust to rotation and viewpoint change. Thus, we propose to use SIFT. SIFT has been explored for text recognition in (Zheng et al. 2010; Iwamura et al. 2011) and for word spotting in (Rusinol et al. 2011; Yalniz & Manmatha 2012). The first two works extracted the descriptors only at sparse interest points, which is not sufficient for perspective characters (to be explained later). The last two works were only tested on frontal scanned document images. In contrast, we adopt dense SIFT (which was used for scene classification in (Bosch et al. 2006)) for perspective character recognition.

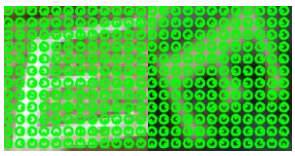
More specifically, the region inside a character candidate[8] is normalized to a fixed size of 48 × 48. We use a grid with spacing of 2 pixels. At each grid point, we extract SIFT descriptors at multiple scales. Note that only the locations and the scales are fixed. The dominant gradient directions of the descriptors may vary across different grid points, as well as across different scales at the same grid point. (In the literature, the term "dense SIFT" sometimes refers to an extraction scheme where the orientations of the dense interest points are fixed. However, we allow them to vary to ensure the rotation-invariance of the descriptors.)

The rationale for using dense SIFT is that it provides more information to discriminate among a large number of classes (62 character classes). With

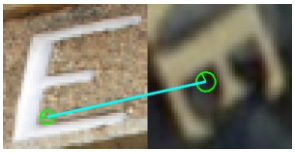
[8] Recall that *character candidate* refers to the bounding box of a character detected by MSER.

the original SIFT, the descriptors are only extracted at interest points. However, scene characters typically suffer from deformations, e.g., blurring and uneven illumination, which reduce the number of detected interest points. More importantly, instances of the same class often suffer from different types of deformations. Thus, the sets of detected interest points are not consistent. This negatively affects the matching of the descriptors. Figure 5.4 shows that using dense SIFT helps to overcome the above problems.

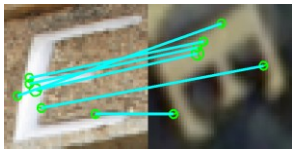
(a) Interest points using normal SIFT



(b) Interest points using dense SIFT



(c) Descriptor matching using normal SIFT



(d) Descriptor matching using dense SIFT

**Figure 5.4. Using normal SIFT leads to few descriptor matches. In contrast, dense SIFT provides more information for character recognition. The left image in each pair is from the training set while the right one is from the test set. Note the fact that the right one is a rotated character. For better illustration, in (b), we only show one scale at each point.**

Furthermore, instead of matching the descriptors directly, we follow a bag-of-keypoints approach (Csurka et al. 2004). By ignoring the spatial information of the keypoints, this approach allows for more distortion between the training and the testing samples. K-means clustering is used to build a vocabulary of 3,000 visual words from a random subset of (dense) SIFT descriptors extracted from training samples. (Section 5.4.1.4 shows how the recognition accuracy varies with respect to the vocabulary size.) With this vocabulary, the descriptors of a character candidate are assigned to the nearest

clusters. The feature representation then becomes a histogram which counts the number of descriptors belonging to each of the visual clusters. We use a standard SVM package[9] with Histogram Intersection Kernel (Maji et al. 2013) to estimate $P(l|c_i) \in [0,1]$. (The training and the testing data are described in section 5.4.)

The fact that our method recognizes perspective characters directly is an advantage over methods which rely on rectification such as (Neumann & Matas 2010; Neumann & Matas 2011). The rectification process is error-prone due to the challenges of scene characters, including blurring and cluttered backgrounds.

#### 5.2.1.3 Non-maximal Suppression

Since multiple MSERs may be detected for the same character (Neumann & Matas 2012), we perform non-maximal suppression (Felzenszwalb et al. 2010) on the set of character candidates. A character candidate is suppressed if it has a significant overlap with another character candidate and the latter has a higher confidence. The overlap ratio is calculated based on the two MSER areas. The confidence of a character candidate is defined as the maximum character probability:

$$Confidence(c_i) = \max_{l \in L} P(l|c_i) \tag{5.1}$$

[9] LIBSVM. http://www.csie.ntu.edu.tw/~cjlin/libsvm/

After suppression, the remaining character candidates are fed into the next step for word recognition.

MSER and SIFT have been used separately for character detection and character recognition in previous works. However, to the best of our knowledge, this work is the first attempt to combine them in a coherent way to recognize perspective characters while using only frontal training data.

### 5.2.2 Recognition at the Word Level

The recognition of perspective texts is much more difficult than that of frontal, horizontal texts due to additional challenges. With arbitrary orientation, it is difficult to distinguish characters such as '6' and '9', and 'u' and 'n', unless there is context information. Furthermore, some characters may be hard to read (due to severe distortions) or even occluded. To deal with these problems, we use a lexicon as the context information. We formulate word recognition as finding the optimal alignment between the character candidates and the lexicon words.

Let $W$ denote the lexicon of an image. Let $C = \{c_1, \dots, c_n\}$ be the set of character candidates. Each character candidate can take a label from $L \cup \{\varepsilon\}$, where $L = \{A, \dots, Z, a, \dots, z, 0, \dots, 9\}$ and $\varepsilon$ is the empty label. Let $a_w$ denote an alignment vector of $C$ to a word label $w \in W$. $a_w(i) = k$ $(k > 0)$ represents that $c_i$ is aligned with $w(k)$, the $k^{\text{th}}$ character of string $w$. $a_w(i) = 0$ indicates that $c_i$ takes the empty label.

For example, in Figure 5.5, $a_{PIONEER}(6) = 7$ indicates that $c_6$ is aligned with the $7^{\text{th}}$ character ('R'). $a_{PIONEER}(1) = 0$ and $a_{PIONEER}(3) = 0$ because $c_1$ and $c_3$ take the empty label. Note that for this image, some of the characters

are missed ('N') or partially detected ('P' and 'O'). However, the alignment vector still allows for a flexible matching.

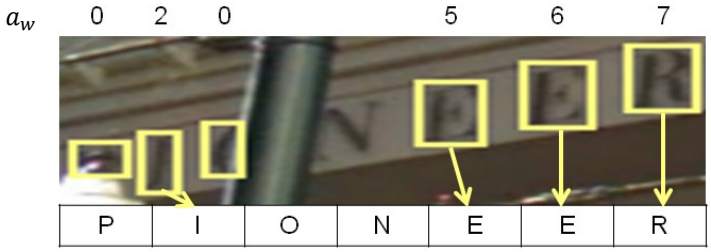



**Figure 5.5. A sample alignment between a set of 6 character candidates (shown in yellow) and the word "PIONEER". The top row shows the value of the alignment vector (of length 6).**

We define $AlignScore(a_w)$ to be the alignment score, which measures how well the labels of the character candidates match the word label $w$ (to be explained more later). The optimal word label $w^*$ can then be found as follows:

$$w^* = \underset{w \in W}{argmax} \quad MaxWordScore(w) \tag{5.2}$$

where

$$MaxWordScore(w) = \underset{a_w \in A_w}{max} \quad AlignScore(a_w) \tag{5.3}$$

$A_w$ denotes the set of all the possible alignments between the character candidates and word label $w$.

Intuitively, Equations (5.2) and (5.3) mean that for each word label in the lexicon, we compute its maximum alignment score. Then, among all the

lexicon words, the one with the highest maximum alignment score is returned as the optimal word label.

#### 5.2.2.1 Ordering of Character Candidates

Our optimized alignment algorithm requires the character candidates to be ordered into a sequence. For simplicity, we assume that text is written from left to right or from top to bottom. If a word image is nearer to the horizontal orientation, the character candidates are ordered by the x-coordinates. Otherwise, they are ordered by the y-coordinates. A word image is classified as either nearer to the horizontal orientation or to the vertical orientation based on the angle of the major axis of its bounding quadrilateral. (For perspective word images, we use quadrilaterals to mark the word locations (Section 5.3).)

#### 5.2.2.2 Alignment Score

As mentioned before, the alignment score measures how well the labels of the character candidates match a word label in the lexicon. It is computed based on the individual scores of the character candidates. Let $Score(c_i, l)$ be the score of assigning label $l$ to character candidate $c_i$:

$$Score(c_i, l) = \begin{cases} P(l|c_i), & if\ l \neq \varepsilon \\ 1 - Confidence(c_i), & if\ l = \varepsilon \end{cases} \quad \textbf{(5.4)}$$

If a character candidate takes a non-empty label, we directly use the corresponding SVM probability. Otherwise, if it takes the empty label, we use a penalty score (inspired by (Mishra et al. 2012b)). The purpose of the penalty

score is to discourage character candidates with high confidence from taking the empty label.

The alignment score of the whole word is the average of the individual scores of the character candidates:

$$AlignScore(a_w) = \frac{1}{n}\sum_{i=1}^{n} Score\left(c_i, w\left(a_w(i)\right)\right) \quad \textbf{(5.5)}$$

Recall that $a_w(i)$ is the index of string $w$ that $c_i$ is aligned to. Thus, $w\left(a_w(i)\right)$ is the label assigned to $c_i$.

#### 5.2.2.3 Optimized Alignment Algorithm

Equation (5.2) is implemented by looping through the word labels in the lexicon. For each word label $w$, we need to compute $MaxWordScore(w)$ (Equation (5.3)). The rest of this section describes our optimized alignment algorithm for doing this. Since $w$ is fixed in Equation (5.3), we drop $w$ in some of the below notations for clarity.

Let $F(c_i, p)$ be the optimal alignment score of character candidates $c_i, c_{i+1}, \dots, c_n$, with $c_i$ aligned at index $p$ of $w$. $F$ can be computed using dynamic programming.

The initialization is described in Algorithm 5.1. Intuitively, Equation (5.6) means that only $c_i$ is assigned a non-empty label while $c_{i+1}, \dots, c_n$ are assigned the empty label. Hence, we use the score of $c_i$ and add the penalty scores of $c_{i+1}, \dots, c_n$.

**Algorithm 5.1. Initializing F's.**

**for** $i = n$ down to 1

**for** $p = len(w)$ down to 1

$$F(c_i, p) = Score\big(c_i, w(p)\big) + \sum_{k=i+1}^{n} Score(c_k, \varepsilon) \tag{5.6}$$

**end for**

**end for**

After that, we update $F$ backwards using Algorithm 5.2. In this algorithm, $j \in [i+1, n]$ and $q \in [p+1, len(w)]$. $c_j$ can be thought of as the first character candidate with a non-empty label after $c_i$. Intuitively, the right hand side (RHS) of Equation (5.7) means that we loop through the combinations of $j$ and $q$. For each combination of $j$ and $q$:

- We use $F\big(c_j, q\big)$, the optimal alignment score for $c_j, \dots, c_n$, as the starting point. (Note that because we compute $F$ backwards, $F\big(c_j, q\big)$ has already been computed, and thus we can use its value.)
- We then add the penalty scores of assigning the empty label to $c_{i+1}, \dots, c_{j-1}$.
- Finally, we add the score of $c_i$.
- After we have looped through all the combinations of $j$ and $q$, if the RHS of Equation (5.7) is greater than the initialized value of $F(c_i, p)$ (in Equation (5.6)), we update the value of the latter to the former.

In our implementation, to reduce the computational time, we restrict the range of $p$ based on $c_i$'s relative position in the image. For example, if $c_i$ is near the left boundary of the image, $p$'s range can be restricted to only the first few indices of $w$. $q$'s range can also be restricted in a similar way.

**Algorithm 5.2. Updating F's.**

**for** $i = n$ down to 1

**for** $p = len(w)$ down to 1

**if** $F(c_i, p) < \max_{j,q} \left(F(c_j, q) + \left(\sum_{k=i+1}^{j-1} Score(c_k, \varepsilon)\right) + Score(c_i, w(p))\right)$ **(5.7)**

update $F(c_i, p)$ to $\max_{j,q} (.)$

**end if**

**end for**

**end for**

When all $F$'s have been computed, $MaxWordScore(w)$ in Equation (5.3) is obtained by:

$$MaxWordScore(w) = \max_{i,p} \left(F(c_i, p) + \sum_{k=1}^{i-1} Score(c_k, \varepsilon)\right) \quad \textbf{(5.8)}$$

The intuition of Equation (5.8) is that we loop through the combinations of $i$ and $p$. For each combination of $i$ and $p$, $c_i$ acts as the first character candidate with a non-empty label (among all the detected character candidates). Hence, we use the score of $c_i$ and add the penalty scores of $c_1, \ldots, c_{i-1}$.

Our optimized alignment algorithm has a few advantages over existing works. First, it explicitly allows the empty label, and thus is able to handle cases where one or more characters are missed or occluded. This is an advantage over (Wang et al. 2011), which does not allow skipping characters when matching with a word label in the lexicon. Second, many methods, e.g., (Wang et al. 2011; Novikova et al. 2012), require normalization for word

length to avoid bias towards shorter words. In contrast, because the magnitude of our alignment score depends on the number of character candidates (and not on the lexicon word length), no normalization is required.

### 5.2.3 Recognition at the Text Line Level

Most recent methods, e.g., (Wang et al. 2011; Mishra et al. 2012a; Novikova et al. 2012; Wang et al. 2012; Mishra et al. 2012b; Neumann & Matas 2013; Shi et al. 2013b), perform recognition at the word level. One of the reasons is that most common datasets, e.g. ICDAR 2003 (Lucas et al. 2003) and Street View Text (Wang et al. 2011), are annotated at the word level. However, at this level, each word is recognized independently, and the labels of the neighboring words are not taken into account.

Hence, in this work, we also explore recognition at the text line level. In particular, we utilize the language context information at the text line level to improve the recognition accuracy. For example, suppose that a text line contains two words: the first one is easy to recognize while the second one is hard to read. Further suppose that for the second word, we have two hypotheses with similar scores: “FRANCISCA” and “FRANCISCO”. Without any context information, the hypothesis with the higher score will be returned as the recognition result. However, if the first word is recognized as “SAN” with high confidence, it is reasonable to favor “FRANCISCO” because “SAN” and “FRANCISCO” often appear together.

Thus, we will describe a recognition scheme in which the scores of all the words in a text line are taken into account. More specifically, we model a text line using a linear-chain Conditional Random Field (CRF) (Lafferty et al.

2001). For each word in the text line, we add a node in the CRF graph (according to the order that the words appear in the text line). We also add edges so that a node is connected to the previous node (corresponding to the previous word) and to the next node (corresponding to the next word).

Let $x_i$ denote the node corresponding to the $i^{\text{th}}$ word image on a text line. Let $y_i$ denote the label assigned to $x_i$. To find the optimal set of word labels for a text line, we need to minimize the following energy function:

$$\sum_i E(y_i, x_i) + \sum_{(i,j)\in\mathcal{N}} E(y_i, y_j, x_i, x_j) \tag{5.9}$$

$E(y_i, x_i)$ is the unary term that represents the cost of assigning label $y_i$ to $x_i$. $E(y_i, y_j, x_i, x_j)$ is the pairwise term that captures the relationship between two neighboring word labels. $\mathcal{N}$ is the set of neighboring pairs.

To construct the CRF graph for a text line, we need to know which words belong to that text line, as well as their ordering in the text line. As aforementioned, most common datasets do not provide these information. Hence, we manually annotate these information for the datasets that we use for experiments (which are described in Sections 5.3 and 5.4). Our annotations consist of two fields:

- $LineNumber$: This field specifies which text line a cropped word image belongs to.
- $WordNumber$: This field specifies the order of a cropped word image on a text line.

Figure 5.6 show example annotations for a street image. With these annotations, we would be able to construct the CRF graphs (two graphs in this

case, one for the “liquid” text line, and the other one for the “LIQUID AGENCY” text line).


(a) Original full image of a street scene

(b) Our annotations for the cropped word images

**Figure 5.6. Example *LineNumber* and *WordNumber* annotations.**

Having explained the CRF graph construction, we will now describe the cost functions. The cost (or penalty) function for the unary term in Equation (5.9) is defined as follows:

$$E(y_i, x_i) = 1 - MaxWordScore(y_i, x_i) \tag{5.10}$$

where $MaxWordScore(y_i, x_i)$ is the score of assigning label $y_i$ to $x_i$, i.e., to the $i^{\text{th}}$ cropped word image. It has been defined earlier in Equation (5.3).

For the pairwise term in Equation (5.9), one way to capture the relationship between the labels of two neighboring words is to use the word $n$-gram model:

$$E(y_i, y_j, x_i, x_j) = 1 - P(y_j | y_i) \tag{5.11}$$

where $P(y_j|y_i)$ is the word bigram probability. This cost function helps to give preferences to sequences of words that occur frequently (in text corpuses).

To estimate the word bigram probabilities, we use Google's Web 1T 5-gram dataset[10], which provides the word $n$-gram[11] counts obtained from a trillion word tokens crawled from web pages on the Internet. This dataset is a huge corpus with 13.6 million distinct words. More importantly, one of its key advantages over traditional corpuses is that since it uses texts from web pages, it contains more brand names, shop names, street names, etc. Thus, it is suitable for our problem because these kinds of words often appear in natural scene images, especially street images.

This dataset was first used for scene text recognition in a recent publication (Feild & Learned-Miller 2013). However, the authors only used the word unigram probabilities to verify the recognition results at the word level. In contrast, we use the word bigram probabilities to perform recognition at the text line level.

Finally, to minimize the energy function in Equation (5.9), we use the well-known Viterbi algorithm (Viterbi 1967). In addition, to reduce the computational time and to avoid over-correction by the word bigram model, for each node in the CRF graph, we only keep its top $K$ hypotheses. In other words, $E(y_i, x_i)$ in Equation (5.10) is modified as follows:

$$E(y_i, x_i) = \begin{cases} 1 - MaxWordScore(y_i, x_i), & if\ y_i\ is\ in\ top\ K\ of\ x_i \\ 1, & otherwise \end{cases} \quad \textbf{(5.12)}$$

[10] http://catalog.ldc.upenn.edu/LDC2006T13
[11] Up to 5-gram

Note that because each $x_i$ corresponds to a different cropped word image, it will have a different set of top $K$ hypotheses. We empirically set $K$ to 5 in our implementation.

In Section 5.4, we will present our experiments for recognition at both the word level and the text line level. Before that, we describe the dataset that we specifically propose for evaluating perspective text recognition in the next section.

## 5.3 StreetViewText-Perspective Dataset

Most of the standard datasets for scene text recognition, e.g., (Lucas et al. 2003; de Campos et al. 2009; Weinman et al. 2009; Wang et al. 2011; Mishra et al. 2012a), are limited to frontal texts. For example, the annotators of the Street View Text (SVT) dataset were instructed to "minimize skew" when choosing the angles of texts (Wang et al. 2011). Recently, there are more challenging datasets: NEOCR (Nagy et al. 2011) and MSRA-TD500 (Yao et al. 2012), which include multi-oriented texts and perspective texts. However, because they are not specifically designed for perspective texts, many of the words in these datasets are still frontal.

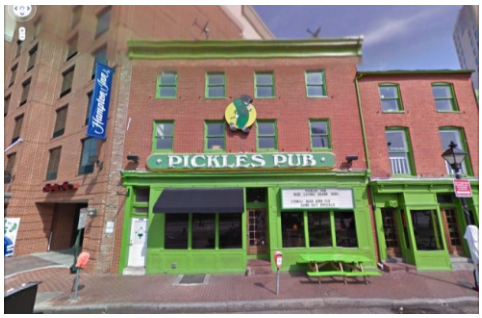
Hence, although we do include MSRA-TD500 in our experiments, we also introduce a new dataset called StreetViewText-Perspective (*SVT-Perspective*)[12], which is specifically designed for evaluating perspective text recognition. Our dataset is built based on the original SVT dataset (Wang et al.

[12] Available at http://www.comp.nus.edu.sg/~phanquyt/

2011) for two reasons. (SVT is a public dataset that contains images taken from Google Street View with frontal texts of shop names, street names, etc.)

First, we would like to reuse the lexicons in SVT, which were collected by Amazon Mechanical Turk workers. As a consequence, our dataset contains images taken at the same addresses on Google Street View. However, instead of choosing the frontal texts, we intentionally picked side-view angles such that texts are still readable to humans. The lexicon of each image was taken to be the same as that of the corresponding SVT image. Second, as our images were taken at the same locations, they allow for a meaningful analysis of the degradation in performance between frontal and perspective texts.

For each image in our dataset, the words were manually annotated using quadrilaterals. Similar to SVT, we only annotated the words that were present in the image-specific lexicons. Figure 5.7 shows a comparison of an image from SVT and an image from SVT-Perspective.




Lexicon: **PICKLES**, **PUB**,

HOTEL, INN, …

(a) SVT image


Lexicon: **PICKLES**, **PUB**,

HOTEL, INN, …

(b) SVT-Perspective image

**Figure 5.7. An image from SVT and the corresponding image from SVT-Perspective. Both images are taken at the same address, and thus have the same lexicon. In (b), the bounding quadrilaterals are shown in black for "PICKLES" and "PUB".**

Our dataset contains 238 images[13], which correspond to the images in the SVT test set. The number of cropped words is 639. The words are of a variety of viewpoints and orientations. Their heights vary from 9 to 330 pixels.

## 5.4 Experimental Results

We performed experiments on perspective texts, multi-oriented texts and frontal texts, at both the word level and the text line level. For the first class of texts, we used our own dataset for reasons explained in the previous section. For the second class of texts, we picked MSRA-TD500 (Yao et al. 2012) because it is a very recent dataset that is specifically designed for multi-oriented texts. (Note that in terms of size, NEOCR (Nagy et al. 2011) is larger than MSRA- TD500. However, it also contains more languages. The English subsets of these two datasets, which are our focus in this work, are comparable in size (Nagy et al. 2011; Yao et al. 2012).) For the third and final class of texts, among the various datasets that have been used in the literature (de Campos et al. 2009; Weinman et al. 2009; Mishra et al. 2012a), we chose ICDAR 2003 (Lucas et al. 2003) and SVT (Wang et al. 2011) because they are the most widely used datasets with many reported results.

MSRA-TD500 only contains annotations at the text line level. Thus, to evaluate word recognition, we manually added word-level annotations[14] for the English words in this dataset (denoted as *MSRA-TD500-Word*).

---

[13] The test set of the SVT dataset contains 249 images. We excluded a small number of images (11 images) because we were not able to find the same shops or buildings using the addresses provided in SVT. The reason is that these shops or buildings have closed or moved. Thus, when Google updated the original addresses with more recent street images, they could no longer be found.

[14] Available at http://www.comp.nus.edu.sg/~phanquyt/

For ICDAR 2003, we used the benchmarks for character recognition (*ICDAR-Char*) and word recognition (*ICDAR-Word*).

For SVT, we used both the original word-level annotations (*SVT-Word*) and the character-level annotations provided in (Mishra et al. 2012b) (*SVT-Char*).

Following recent works, e.g., (Wang et al. 2011; Mishra et al. 2012b), we used the **case-insensitive** word recognition accuracy as the performance measure. This is reasonable considering the application of business name search in street images, where, for indexing purpose, case does not matter.

Moreover, although it is mentioned in the previous section that each word was manually annotated using a quadrilateral, we have experimentally found that there is negligible difference (in word recognition accuracy) between using the quadrilaterals and using their minimum bounding rectangles (Figure 5.8). Thus, all the experiments in this section used the latter. This helps to ensure that the same input images are used for all the methods in the experiments, because the existing methods that we used for comparison assume rectangular cropped words.


(a) Cropped word using quadrilateral

(b) Cropped word using the minimum bounding rectangle of the quadrilateral

**Figure 5.8. All the experiments in this section used rectangular cropped words (b).**

Due to the large number of visual words used (Section 5.2.1.2), we need to collect enough data to train the character classifier. We used samples from

ICDAR-Char (only the training subset) and two other public datasets for frontal texts: Weinman's dataset (Weinman et al. 2009) and Chars74k (de Campos et al. 2009) (only the English subset). In total, we had 19,800 training samples. This training set was used for all the experiments in this section.

We will now describe the experiments for recognition at the word level. The experiments for the text line level will be discussed in Section 5.4.2.

### 5.4.1 Recognition at the Word Level

#### 5.4.1.1 Experiment on Perspective Texts

In this experiment, we used our SVT-Perspective dataset for evaluation. We obtained the source codes of (Wang et al. 2011) and (Wang et al. 2012) from the authors' websites. We also re-implemented (Mishra et al. 2012b) following the descriptions in the paper, and included ABBYY FineReader 9.0, a commercial OCR engine, in the comparative study. We used the same experimental settings as (Wang et al. 2011; Mishra et al. 2012b). In particular, words with less than 3 characters or containing non-alphanumeric characters were ignored.

The second column of Table 5.1 shows that our method significantly outperformed the other methods. The increase in accuracy from 45.7% (of (Mishra et al. 2012b)) to 62.3% (of our method) represents a relative improvement of 36%. In Figure 5.9, our method recognized the words correctly despite the blurring, occlusion and large variation in text appearance.

These results show the advantage of using dense SIFT to recognize perspective texts. Our alignment algorithm also contributes to the handling of

**Table 5.1. Recognition accuracy on perspective words (in %).**

| Method | SVT-Perspective-Word | SVT-Perspective-Word (Full) |
|---|---|---|
| ABBYY FineReader 9.0 | 16.9 | 9.7 |
| (Wang et al. 2011) | 40.5 | 26.1 |
| (Mishra et al. 2012b) | 45.7 | 24.7 |
| (Wang et al. 2012) | 40.2 | 32.4 |
| Our method | **62.3** | **42.2** |


| | | | |
|---|---|---|---|
| (Wang et al. 2011) | COFFEES | LIGHT | ADLER |
| (Mishra et al. 2012b) | SQUARE | FOR | SAN |
| (Wang et al. 2012) | INC | THE | SAN |
| Our method | **MURPHY** | **JONES** | **LIGHTS** |


| | | | | |
|---|---|---|---|---|
| (Wang et al. 2011) | FIRST | ALLEY | ICON | LION |
| (Mishra et al. 2012b) | SAKE | CENTER | AND | AMC |
| (Wang et al. 2012) | FRY | AMC | SPA-GHETTI | AMC |
| Our method | **GARAGE** | **CINERAMA** | **WARE-HOUSE** | **CINE-RAMA** |

**Figure 5.9. Sample recognition results for multi-oriented texts and perspective texts.**

characters that are occluded or hard to read. Moreover, we have shown that using **only frontal characters** for training is a sensible and realistic approach because it avoids the cost of collecting perspective character samples.

We would like to emphasize that our training data did **not** contain any samples from SVT. The training data came from other datasets, as aforementioned. Thus, the successful recognition of the perspective words is purely due to the generalization power of dense SIFT and SVM (and **not** because of the similarity between SVT and SVT-Perspective).

We also analyzed how the recognition accuracy varied with the lexicon size. Intuitively, a larger lexicon makes it more difficult to recognize a word, especially if there are several similar words in the lexicon. In addition to the original lexicon size (of around 50 words per image), we used another lexicon size denoted as *Full*. This lexicon contained 377 words (an increase of 7.5 times in size) and was constructed by putting all the ground truth words in the test set into a list (following the procedure in (Wang et al. 2011)). The third column of Table 5.1 shows that even when a larger lexicon was used, our method still achieved the best accuracy.

#### 5.4.1.2 Experiment on Multi-oriented Texts

In this experiment, we ran the same set of methods on MSRA-TD500-Word. Since this dataset does not have lexicons, we constructed a *Full* lexicon of 395 words in a similar way as in the previous section. Table 5.2 shows that our method also significantly outperformed the other methods on this dataset. The increase in accuracy from 44.5% (of (Wang et al. 2011)) to 58.4% (of our

method) represents a relative improvement of 31%. Figure 5.10 shows sample results of our method.

We are the first to report the recognition accuracy on MSRA-TD500, a very recent public dataset for multi-oriented texts. The fact that our method performed well on both SVT-Perspective and MSRA-TD500 demonstrates its advantage over existing methods.

**Table 5.2. Accuracy on multi-oriented words (in %).**

| Method | MSRA-TD500-Word (Full) |
|---|---|
| ABBYY FineReader 9.0 | 23.2 |
| (Wang et al. 2011) | 44.5 |
| (Mishra et al. 2012b) | 27.8 |
| (Wang et al. 2012) | 20.8 |
| Our method | **58.4** |


AZONA EXPO HYDRANT TARGET

**Figure 5.10. Sample recognition results of our method for multi-oriented words.**

#### 5.4.1.3 Experiment on Frontal Texts

In the previous sections, we do not evaluate cropped character recognition because SVT-Perspective and MSRA-TD500 do not have character-level annotations. In contrast, ICDAR and SVT do have annotations at the character level. Thus, we compared the character recognition accuracy (using 62 classes) on cropped character images. We have also included the results reported by other recent works.

Table 5.3 shows that on SVT-Char, our method achieved the state-of-the-art accuracy of 67.0%. The previous best known result on this dataset was 61.9% (Mishra et al. 2012b).

On ICDAR-Char, our method outperformed (Wang et al. 2011). Although its accuracy was lower than those of (Coates et al. 2011; Wang et al. 2012), these results should be interpreted with the following consideration: when rotation-invariant features are used, it is difficult to distinguish pairs of characters such as 'u' and 'n', and '6' and '9', especially because no context information is available at the character level. Therefore, because our method uses rotation-invariant features, it is at a slight disadvantage compared to the other methods, including (Coates et al. 2011; Wang et al. 2012). Figure 5.11 shows sample results of our method.

**Table 5.3. Cropped character recognition accuracy (in %).**

| Method | ICDAR-Char | SVT-Char |
|---|---|---|
| ABBYY FineReader 9.0 | 21.0 | 11.7 |
| (Wang et al. 2011) | 64.0 | N.A |
| (Mishra et al. 2012b) | N.A | 61.9 |
| (Coates et al. 2011) | 81.7 | N.A |
| (Wang et al. 2012) | **83.9** | N.A |
| Our method | 75.6 | **67.0** |

(a) Success cases (b) Failure cases

**Figure 5.11. Sample character recognition results of our method. In (a), the characters were correctly recognized despite the strong highlight, small occlusion, similar text and background colors, and rotation. In (b), the characters were wrongly recognized due to low resolution, strong shadow and rotation invariance. The last character was recognized as '6'.**

The next experiment is on word recognition. Each image in SVT-Word comes with a lexicon (of around 50 words). On the other hand, ICDAR-Word

does not have lexicons. For fair comparison, we used the lexicons provided in (Wang et al. 2011). Table 5.4 shows that on SVT-Word, our method achieved the best recognition accuracy. Our accuracy of 73.7% is slightly higher than the previous best known result of 73.6% (Mishra et al. 2012a). On ICDAR-Word, our method achieved the third best accuracy. Sample results of our method are shown in Figure 5.12.

Furthermore, the fact that the images in SVT and SVT-Perspective were taken at the same addresses on Google Street View allows for an analysis of the degradation in performance between frontal and perspective texts. The drop in accuracy of our method (-15.5%) was significantly lower than those of the other methods (Table 5.5). This shows that our method is more robust

**Table 5.4. Recognition accuracy on frontal words (in %).**

| Method | ICDAR-Word | SVT-Word |
|---|---|---|
| ABBYY FineReader 9.0 | 56.0 | 35.0 |
| (Wang et al. 2011) | 76.0 | 57.0 |
| (Wang et al. 2012) | **90.0** | 70.0 |
| (Novikova et al. 2012) | 82.8[15] | 72.9 |
| (Mishra et al. 2012b) | 81.8 | 73.3 |
| (Mishra et al. 2012a) | 80.3 | 73.6 |
| Our method | 82.2 | **73.7** |


**Figure 5.12. Sample results of our method for frontal words. It was able to recognize the words under challenging scenarios: transparent text, occlusion, fancy font, similar text and background colors and strong highlight.**

[15] Achieved using a slightly larger lexicon for ICDAR-Word.

**Table 5.5. Degradation in performance between frontal and perspective texts (in %).**

| Method | SVT-Word | SVT-Perspective-Word | % change |
|---|---|---|---|
| ABBYY FineReader 9.0 | 35.0 | 16.9 | -51.7 |
| (Wang et al. 2011) | 57.0 | 40.5 | -28.9 |
| (Mishra et al. 2012b) | 73.3[16] | 45.7[17] | -37.7 |
| (Wang et al. 2012) | 70.0 | 40.2 | -42.6 |
| Our method | 73.7 | 62.3 | **-15.5** |

against rotation and perspective distortion, which is highly important for practical applications.

#### 5.4.1.4 Experiment on Number of Visual Words

We analyzed how the recognition accuracy of our method changed with respect to the number of visual words, which is used for K-means clustering in section 5.2.1.2. As illustrated in Figure 5.13, there was a trend on all four datasets: the accuracies increased with the number of visual words used. However, the accuracies slightly dropped at 4,000 visual words for ICDAR-Word and SVT-Word (for SVT-Perspective-Word and MSRA-TD500-Word, there were very small increases in accuracy). Thus, with the amount of training data that we have, the typical value to use for the vocabulary size is 2,000–4,000 visual words.

### 5.4.2 Recognition at the Text Line Level

In addition to the word level, we performed an experiment at the text line level. As mentioned in Section 5.2.3, we manually annotated two

[16] Taken from (Mishra et al. 2012b).
[17] Obtained from our re-implementation of (Mishra et al. 2012b), which follows the paper closely. Its accuracy on SVT-Word was 69.1%, which is close to the 73.3% accuracy reported in (Mishra et al. 2012b) for the same dataset.

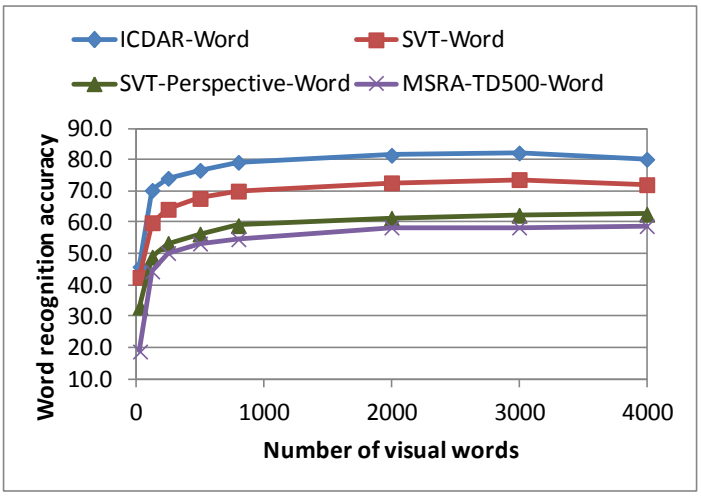
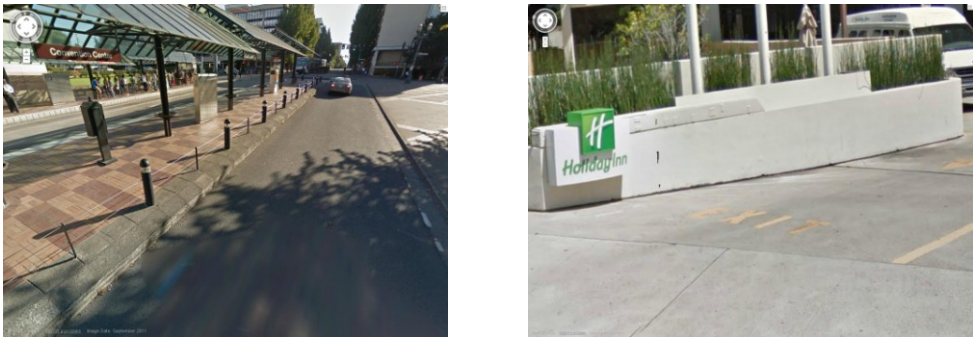



**Figure 5.13. Recognition accuracies of our method for different vocabulary sizes.**

additional fields ($LineNumber$ and $WordNumber$) for SVT-Perspective, MSRA, SVT and ICDAR.

Figure 5.14 shows sample results of performing recognition at the text line level. In both cases, the language context information at the text line level helps to correct the word-level results. In particular, for the image on the right hand side, the correct label of the first word ("HOLIDAY") is only the 4$^{th}$ best hypothesis (based on its $MaxWordScore$ at the word level). However, because "HOLIDAY INN" is a well-known brand name, this combination has a higher word bigram count in the corpus than the other combinations such as "HARBOR INN" and "HOTEL INN". Thus, our CRF model utilizes this information to "flip" the label of the first word from "HARBOR" to "HOTEL" and return the correct recognition result.

In this experiment, we did not include any existing methods because all the methods previously used for comparison in Section 5.4.1 do not utilize the text line-level context information.



(a) Input images

| | | | |
|---|---|---|---|
| CONVENTION | CONVENTION | HARBOR | INN |
| RED | CENTER | HOTEL | INC |
| CONDOMINIUMS | RED | SAN | SAN |
| NETWORK | LION | HOLIDAY | LAW |
| CAFE | DENNY | WESTERN | LLC |

(b) The top 5 hypotheses of each word

| | |
|---|---|
| Result at the word level:<br>CONVENTION CONVENTION | Result at the word level:<br>HARBOR INN |
| Result at the text line level:<br>CONVENTION **CENTER** | Result at the text line level:<br>**HOLIDAY** INN |

(c) The recognition results at different levels

**Figure 5.14. Sample results of recognition at the text line level. In (a), the image on the left contains a single text line ("CONVENTION CENTER") and the image on the right also contains a single text line ("HOLIDAY INN"). In (c), the words that are changed due to the use of the language context information at the text line level are bolded and underlined.**

Table 5.6 shows the accuracies of our method on various datasets when performing recognition at the word level and at the text line level. Note that the accuracies still refer the word recognition accuracies, i.e., the percentage of words correctly recognized. The difference is that when recognition is performed at the text line level, some word labels may be "flipped" due to the availability of context information (as illustrated in Figure 5.14 above).

It is observed that the recognition accuracies at the text line level are greater than or equal to those at the word level. On SVT-Perspective-Word,

the recognition accuracy increases by 1.4%, thanks to use of the language context information. The gains on the other three frontal text datasets are smaller, ranging from 0% (on MSRA-TD500-Word (Full)) to 0.3% (on SVT-Word).

A reason is that perspective texts are much harder to recognize, and the scores of the word label hypotheses may be less accurate, which leaves more room for improvement when incorporating the word bigram probabilities. On the other hand, frontal texts are easier to recognize and thus, the increases in accuracies are smaller.

Nevertheless, the results demonstrate that the language context information is useful for perspective text recognition. In the future, we will explore other forms of context information to further improve the recognition accuracy.

**Table 5.6. Accuracies of our method when performing recognition at the word level and at the text line level (in %).**

| Dataset | Recognition at the word level | Recognition at the text line level |
|---|---|---|
| SVT-Perspective-Word | 62.3 | 63.7 |
| MSRA-TD500-Word (Full) | 58.4 | 58.4 |
| SVT-Word | 73.7 | 74.0 |
| ICDAR-Word | 82.2 | 82.3 |

### 5.4.3 Experiment on Processing Time

On SVT-Perspective (with the original lexicons of around 50 words per image), the average processing time of our unoptimized Matlab code was 59 seconds, which include 38.6 seconds to recognize a cropped word image (i.e., at the word level) and 20.4 seconds to incorporate the language context

information (i.e., at the text line level). This was measured on a machine with Intel Core i5 processor (quad-core, 3.2 GHz) and 4 GB RAM.

In the future, we will explore optimization techniques to reduce the computational time. For example, at the word level, a trie structure can be used for the lexicon to avoid redundant computation (Wang et al. 2011). At the text line level, more efficient data structures can be implemented to speed up the querying of word bigram counts from the large corpus.

## 5.5 Summary

Our work serves as a step towards practical applications (of scene text extraction) in two aspects. First, most existing works make the simplistic assumption that text is horizontal and frontal parallel to the image plane. However, in many real-world scenarios, this assumption does not hold. Thus, by handling perspective texts, this work has attempted to address an important research gap. Second, an attractive feature of our method is that it is trained on only frontal character samples, and thus does not require collecting samples of perspective characters. This drastically reduces the cost of data collection.

The second aspect is achieved by the use of dense SIFT in a bag-of-keypoints framework, which is robust to rotation and viewpoint change. Our optimized alignment algorithm is also designed to handle the challenges of perspective texts, e.g., one or more characters may be hard to read or occluded.

Another contribution is the SVT-Perspective dataset, which we propose to evaluate perspective text recognition. On this dataset, our method compares

favorably to the state-of-the-art. Therefore, our results and dataset serve as a baseline for future studies on perspective texts.

Currently, our method is limited to cases where lexicons are available. Although this assumption is common in the literature (Wang et al. 2011; Mishra et al. 2012b), it restricts the applicability of our method. Relaxing or even removing this assumption will be left for future research.

# Chapter 6

# Conclusions and Future Work

## 6.1 Summary of Contributions

In this thesis, we address the problem of text extraction in images and videos, which can be used for many applications such as content-based image/video retrieval, sign translation and navigation aid for the visually-impaired and robots. From our literature review, we have identified a number of research gaps and proposed novel works to address them. Our works contribute to the progress of the research field in the following areas:

- **Text Localization**

  We have proposed two text localization works, one for text in natural scene images (Phan et al. 2012) and the other for text in video key frames (Phan et al. 2009; Shivakumara et al. 2011b).

  - The first work introduces novel gap, i.e., inter-character, features to localize difficult cases of scene text. While previous methods for scene text focus on only the character features, this work has shown that the gap features can also play an important role in text localization. Our work achieves better localization performance than existing ones on two public datasets (Lucas et al. 2003; Epshtein et al. 2010).
  - The second work uses skeletonization to localize multi-oriented video text. This is an improvement over previous methods for

video text, which typically localize only horizontal text. Our work outperforms existing ones on video frames with multi-oriented English and Chinese texts. It has been cited by recent papers, e.g., (Bouman et al. 2011; Chen et al. 2011; Du et al. 2011; Mosleh et al. 2012; Shi et al. 2012; Sun & Lu 2012; Wen et al. 2012; Shi et al. 2013a), including those which further pursue the direction of multi-oriented text localization, e.g., (Sharma et al. 2012; Yao et al. 2012; Zhang & Lai 2012).

- **Text Enhancement**

  We have presented two works, one for single-frame text enhancement (Phan et al. 2011) and the other for multiple-frame enhancement (Phan et al. 2013a).

  - With the first work, we have shown that binarizing each character in a text line individually (instead of binarizing the whole text line) helps to improve the recognition accuracy. This is achieved through our character segmentation technique which is capable of producing curved segmentation paths to closely match the characters' shapes. Hence, it improves over many existing techniques which allow only vertical cuts. This work has been cited by recent papers, e.g., (Elagouni et al. 2013; Goel et al. 2013; Sharma et al. 2013).
  - With the second work, we have demonstrated that exploiting the temporal redundancy in videos leads to significantly better recognition accuracy. Our work also emphasizes the importance of identifying the text pixels prior to enhancement so that the

text region is enhanced while the background region is suppressed. In contrast, many previous methods utilize all pixels (including background pixels) for enhancement, which may result in accidental enhancement of the background region.

- **Text Recognition**

  We have proposed a work (Phan et al. 2013b) to address an issue which has been neglected by most previous methods for scene text: the recognition of perspective texts. By using features which are robust to rotation and viewpoint change, our work achieves the state-of-the-art recognition accuracy on several public datasets (Wang et al. 2011; Yao et al. 2012; Mishra et al. 2012b). Moreover, our work requires only frontal character samples for training, thereby avoiding the labor-intensive and time-consuming process of collecting perspective character samples. We also propose a dataset for evaluating perspective text recognition. Hence, our results and dataset serve as a baseline for future studies in this direction.

## 6.2 Future Research Directions

There are several directions for future research. One direction is to utilize more sophisticated context information. Our work in Chapter 5 explores the language model at the text line level by using word bigram probabilities. It may be worthwhile to explore not only the co-occurrence statistics but also the semantic coherence of the words that appear in a same image. For example, suppose that a word is equally likely to be “food” or

“foot” (because the last character is hard to recognize). If we know that the word “restaurant” is also present in the same image, it makes sense to give preference to the former due to the semantic coherence.

Moreover, our work in Chapter 5 assumes the availability of image-specific lexicons. In practice, there are cases where such lexicons cannot be obtained. Thus, if this assumption can be relaxed or even removed (i.e., performing lexicon-free recognition), it will make the proposed method more general and applicable to more real-world scenarios.

Another direction, which is specific to the case of video text, is to investigate techniques that can track texts with complex motions. The proposed multiple-frame text enhancement method in section 4.2 currently handles only static text and linearly moving text. If it can be extended to track more complex movements (e.g., rotation and zooming in/out), it can be used for a wider range of video texts.

# Publications during Candidature

1. **T. Q. Phan**, P. Shivakumara, S. Tian and C. L. Tan. Recognizing Text with Perspective Distortion in Natural Scenes. International Conference on Computer Vision 2013.

2. **T. Q. Phan**, P. Shivakumara and C. L. Tan. Recognition of Video Text Through Temporal Integration. International Conference on Document Analysis and Recognition 2013.

3. **T. Q. Phan**, P. Shivakumara and C. L. Tan. Detecting Text in the Real World. ACM International Conference on Multimedia 2012.

4. **T. Q. Phan**, P. Shivakumara and C. L. Tan. Text Detection in Natural Scenes Using Gradient Vector Flow-Guided Symmetry. International Conference on Pattern Recognition 2012.

5. **T. Q. Phan**, P. Shivakumara and C. L. Tan. A Gradient Vector Flow-Based Method for Video Character Segmentation. International Conference on Document Analysis and Recognition 2011.

6. P. Shivakumara, **T. Q. Phan** and C. L. Tan. A Laplacian Approach to Multi-Oriented Text Detection in Video. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(2), 2011, pp. 412–419.

7. **T. Q. Phan**, P. Shivakumara and C. L. Tan. A Skeleton-Based Method for Multi-Oriented Video Text Detection. International Workshop on Document Analysis Systems 2010.

8. T. Q. Phan, P. Shivakumara, S. Bhowmick, S. Li, C. L. Tan and U. Pal. Semi-Automatic Ground Truth Generation for Text Detection and Recognition in Video Images. IEEE Transactions on Circuits and Systems for Video Technology, 2014 (to appear).

9. P. Shivakumara, **T. Q. Phan**, S. Lu and C. L. 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), 2013, pp. 1729–1739.

10. P. Shivakumara, **T. Q. Phan**, S. Bhowmick, C. L. Tan and U. Pal. A Novel Ring Radius Transform for Video Character Reconstruction. Pattern Recognition, 46(1), 2013, pp. 131–140.

11. S. Tian, P. Shivakumara, **T. Q. Phan** and C. L. Tan. Scene Character Reconstruction through Medial Axis. International Conference on Document Analysis and Recognition 2013.

12. P. Shivakumara, R. P. Sreedhar, **T. Q. Phan**, S. Lu and C. L. Tan. Multi-Oriented Video Scene Text Detection through Bayesian Classification and Boundary Growing. IEEE Transactions on Circuits and Systems for Video Technology, 22(8), 2012, pp. 1227–1235.

13. B. Su, S. Lu, **T. Q. Phan** and C. L. Tan. Character Extraction in Web Images for Text Recognition. International Conference on Pattern Recognition 2012.

14. **T. Q. Phan**, P. Shivakumara and C. L. Tan. Video Script Identification based on Text Lines. International Conference on Document Analysis and Recognition 2011.

15. P. Shivakumara, **T. Q. Phan** and C. L. Tan. Video Character Recognition through Hierarchical Classification. International Conference on Document Analysis and Recognition 2011.

16. P. Shivakumara, A. Dutta, **T. Q. Phan**, C. L. Tan and U. Pal. A Novel Mutual Nearest Neighbor based Symmetry for Text Frame Classification in Video. Pattern Recognition, 44(8), 2011, pp. 1671–1683.

17. P. Shivakumara, **T. Q. Phan** and C. L. Tan. New Fourier-Statistical Features in RGB Space for Video Text. IEEE Transactions on Circuits and Systems for Video Technology, 20(11), 2010, pp. 1520–1532.

18. P. Shivakumara, **T. Q. Phan** and C. L. Tan. New Wavelet and Color Features for Text Detection in Video. International Conference on Pattern Recognition 2010.

19. P. Shivakumara, W. Hua, **T. Q. Phan** and C. L. Tan. Accurate Video Text Detection through Classification of Low and High Contrast Images. Pattern Recognition, 43(6), 2010, pp. 2165–2185.

# Bibliography


Ballan, L., Bertini, M., Del Bimbo, A. & Serra, G. (2010). Semantic Annotation of Soccer Videos by Visual Instance Clustering and Spatial/temporal Reasoning in Ontologies. *Multimedia Tools and Applications*, 48(2), pp. 313–337.

Bosch, A., Zisserman, A. & Munoz, X. (2006). Scene Classification Via pLSA. In *Proceedings of the 2006 European Conference on Computer Vision*, pp. 517–530.

Bouman, K.L., Abdollahian, G., Boutin, M. & Delp, E.J. (2011). A Low Complexity Sign Detection and Text Localization Method for Mobile Applications. *IEEE Transactions on Multimedia*, 13(5), pp. 922–934.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1), pp. 5–32.

Cai, M., Song, J. & Lyu, M.R. (2002). A New Approach for Video Text Detection. In *Proceedings of the 2002 International Conference on Image Processing*, pp. 117–120.

De Campos, T., Babu, B.R. & Varma, M. (2009). Character Recognition in Natural Images. In *Proceedings of the 2009 International Conference on Computer Vision Theory and Applications*, pp. 273–280.

Canny, J. (1986). A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), pp. 679–698.

Capel, D. & Zisserman, A. (2000). Super-Resolution Enhancement of Text Image Sequences. In *Proceedings of the 2000 International Conference on Pattern Recognition*, pp. 600–605.

Casey, R.G. & Lecolinet, E. (1996). A Survey of Methods and Strategies in Character Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7), pp. 690–706.

Chen, D., Bourlard, H. & Thiran, J.-P. (2001a). Text Identification in Complex Background Using SVM. In *Proceedings of the 2001 Conference on Computer Vision and Pattern Recognition*, pp. 621–626.

Chen, D. & Odobez, J.-M. (2005). Video text recognition using sequential Monte Carlo and error voting methods. *Pattern Recognition Letters*, 26(9), pp. 1386–1403.

Chen, D., Odobez, J.-M. & Bourlard, H. (2004a). Text detection and recognition in images and video frames. *Pattern Recognition*, 37(3), pp. 595–608.

Chen, D., Odobez, J.-M. & Thiran, J.-P. (2004b). A localization/verification scheme for finding text in images and video frames based on contrast independent features and machine learning methods. *Signal Processing: Image Communication*, 19(3), pp. 205–217.

Chen, D., Olobez, J.-M. & Bourlard, H. (2002). Text segmentation and recognition in complex background based on Markov random field. In *Proceedings of the 2002 International Conference on Pattern Recognition*, pp. 227–230.

Chen, D., Shearer, K. & Bourlard, H. (2001b). Text Enhancement with Asymmetric Filter for Video OCR. In *Proceedings of the 2001 International Conference on Image Analysis and Processing*, pp. 192–197.

Chen, H., Tsai, S.S., Schroth, G., Chen, D.M., Grzeszczuk, R. & Girod, B. (2011). Robust Text Detection in Natural Images with Edge-enhanced Maximally Stable Extremal Regions. In *Proceedings of the 2011 International Conference on Image Processing*, pp. 2609–2612.

Chen, X., Yang, J., Zhang, J. & Waibel, A. (2004c). Automatic Detection and Recognition of Signs From Natural Scenes. *IEEE Transactions on Image Processing*, 13(1), pp. 87–99.

Chen, X. & Yuille, A.L. (2004). Detecting and reading text in natural scenes. In *Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition*, pp. 366–373.

Coates, A., Carpenter, B., Case, C., Satheesh, S., Suresh, B., Wang, T., Wu, D.J. & Ng, A.Y. (2011). Text Detection and Character Recognition in Scene Images with Unsupervised Feature Learning. In *Proceedings of the 2011 International Conference on Document Analysis and Recognition*, pp. 440–445.

Crandall, D., Antani, S. & Kasturi, R. (2003). Extraction of special effects caption text events from digital video. *International Journal on Document Analysis and Recognition*, 5(2-3), pp. 138–157.

Csurka, G., Dance, C.R., Fan, L., Willamowski, J. & Bray, C. (2004). Visual Categorization with Bags of Keypoints. In *Proceedings of the 2004 European Conference on Computer Vision*, pp. 1–22.

Dalal, N. & Triggs, B. (2005). Histograms of oriented gradients for human detection. In *Proceedings of the 2005 Conference on Computer Vision and Pattern Recognition*, pp. 886–893.

Dance, C.R. (2001). Perspective Estimation for Document Images. In *Proceedings of the 2001 SPIE Conference on Document Recognition and Retrieval*, pp. 244–254.

Donaldson, K. & Myers, G.K. (2005). Bayesian Super-Resolution of Text in Video with a Text-Specific Bimodal Prior. In *Proceedings of the 2005 Conference on Computer Vision and Pattern Recognition*, pp. 1188–1195.

Du, Y., Ai, H. & Lao, S. (2011). Dot Text Detection Based on FAST Points. In *Proceedings of the 2011 International Conference on Document Analysis and Recognition*, pp. 435–439.

Elagouni, K., Garcia, C., Mamalet, F. & Sébillot, P. (2013). Text recognition in multimedia documents: a study of two neural-based OCRs using and avoiding character segmentation. *International Journal on Document Analysis and Recognition*, 17(1), pp. 1–13.

Epshtein, B., Ofek, E. & Wexler, Y. (2010). Detecting Text in Natural Scenes with Stroke Width Transform. In *Proceedings of the 2010 Conference on Computer Vision and Pattern Recognition*, pp. 2963–2970.

Feild, J.L. & Learned-Miller, E.G. (2013). Improving Open-Vocabulary Scene Text Recognition. In *Proceedings of the 2013 International Conference on Document Analysis and Recognition*, pp. 604–608.

Felzenszwalb, P.F., Girshick, R.B., McAllester, D.A. & Ramanan, D. (2010). Object Detection with Discriminatively Trained Part-Based Models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9), pp. 1627–1645.

Felzenszwalb, P.F. & Huttenlocher, D.P. (2005). Pictorial structures for object recognition. *International Journal of Computer Vision*, 61(1), pp. 55–79.

Fischler, M.A. & Bolles, R.C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), pp. 381–395.

Freund, Y. & Schapire, R.E. (1996). Experiments with a New Boosting Algorithm. In *Proceedings of the 1996 International Conference on Machine Learning*, pp. 148–156.

Gargi, U., Crandall, D., Antani, S., Gandhi, T., Keener, R. & Kasturi, R. (1999). A system for automatic text detection in video. In *Proceedings of the 1999 International Conference on Document Analysis and Recognition*, pp. 29–32.

Gatos, B., Ntirogiannis, K. & Pratikakis, I. (2009). ICDAR 2009 Document Image Binarization Contest (DIBCO 2009). In *Proceedings of the 2009*

*International Conference on Document Analysis and Recognition*, pp. 1375–1382.

Gllavata, J., Ewerth, R. & Freisleben, B. (2004). Tracking text in MPEG videos. In *Proceedings of the 2004 ACM International Conference on Multimedia*, pp. 240–243.

Goel, V., Mishra, A., Alahari, K. & Jawahar, C.V. (2013). Whole is Greater than Sum of Parts: Recognizing Scene Text Words. In *Proceedings of the 2013 International Conference on Document Analysis and Recognition*, pp. 398–402.

Goto, H. & Tanaka, M. (2009). Text-Tracking Wearable Camera System for the Blind. In *Proceedings of the 2009 International Conference on Document Analysis and Recognition*, pp. 141–145.

Graves, A., Liwicki, M., Fernandez, S., Bertolami, R., Bunke, H. & Schmidhuber, J. (2009). A Novel Connectionist System for Unconstrained Handwriting Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5), pp. 855 –868.

Guo, G., Jin Jin, Xijian Ping & Tao Zhang (2007). Automatic Video Text Localization and Recognition. In *Proceedings of the 2007 International Conference on Image and Graphics*, pp. 484–489.

Harris, C. & Stephens, M. (1988). A combined corner and edge detector. In *Proceedings of the 1988 Alvey Vision Conference*, pp. 147–151.

Hua, X.-S., Yin, P. & Zhang, H.-J. (2002). Efficient Video Text Recognition Using Multiple Frame Integration. In *Proceedings of the 2002 International Conference on Image Processing*, pp. 22–25.

Huang, X., Ma, H. & Zhang, H. (2009). A new video text extraction approach. In *Proceedings of the 2009 International Conference on Multimedia and Expo*, pp. 650–653.

Irani, M. & Peleg, S. (1991). Improving resolution by image registration. *Computer Vision, Graphics, and Image Processing*, 53(3), pp. 231–239.

Isard, M. & Blake, A. (1998). CONDENSATION - conditional density propagation for visual tracking. *International Journal of Computer Vision*, 29(1), pp. 5–28.

Isard, M. & Blake, A. (1996). Contour tracking by stochastic propagation of conditional density. In *Proceedings of the 1996 European Conference on Computer Vision*, pp. 343–356.

Iwamura, M., Kobayashi, T. & Kise, K. (2011). Recognition of Multiple Characters in a Scene Image Using Arrangement of Local Features. In *Proceedings of the 2011 International Conference on Document Analysis and Recognition*, pp. 1409 –1413.

Jain, A.K. & Yu, B. (1998). Automatic text location in images and video frames. In *Proceedings of the 1998 International Conference on Pattern Recognition*, pp. 1497–1499.

Jung, C., Liu, Q. & Kim, J. (2008). A new approach for text segmentation using a stroke filter. *Signal Processing*, 88(7), pp. 1907–1916.

Jung, K. & Han, J. (2004). Hybrid approach to efficient text extraction in complex color images. *Pattern Recognition Letters*, 25(6), pp. 679–699.

Jung, K., In Kim, K. & K. Jain, A. (2004). Text information extraction in images and video: a survey. *Pattern Recognition*, 37(5), pp. 977–997.

Kae, A., Huang, G., Doersch, C. & Learned-Miller, E. (2010). Improving state-of-the-art OCR through high-precision document-specific modeling. In *Proceedings of the 2010 Conference on Computer Vision and Pattern Recognition*, pp. 1935–1942.

Khotanzad, A. & Hong, Y.H. (1990). Invariant Image Recognition by Zernike Moments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(5), pp. 489–497.

Kim, K., Jung, K., Park, S. & Kim, H. (2001). Support vector machine-based text detection in digital video. *Pattern Recognition*, 34(2), pp. 527–529.

Kim, K.I., Jung, K. & Kim, J.H. (2003). 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), pp. 1631–1639.

Kim, W. & Kim, C. (2009). A new approach for overlay text detection and extraction from complex video scene. *IEEE Transactions on Image Processing*, 18(2), pp. 401–411.

Kopf, S., Haenselmann, T., Effelsberg, W. & Kopf, S.; H. (2005). *Robust Character Recognition in Low-Resolution Images and Videos*. Technical report. Department for Mathematics and Computer Science, University of Mannheim.

Kuo, S.-S. & Ranganath, M.V. (1995). Real time image enhancement for both text and color photo images. In *Proceedings of the 1995 International Conference on Image Processing*, pp. 159–162.

Lafferty, J.D., McCallum, A. & Pereira, F.C.N. (2001). Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In *Proceedings of the 2001 International Conference on Machine Learning*, pp. 282–289.

Lee, C.W., Jung, K. & Kim, H.J. (2003). Automatic text detection and removal in video sequences. *Pattern Recognition Letters*, 24(15), pp. 2607–2623.

Li, H. & Doermann, D. (1999). Text enhancement in digital video using multiple frame integration. In *Proceedings of the 1999 ACM International Conference on Multimedia*, pp. 19–22.

Li, H., Doermann, D. & Kia, O. (2000). Automatic text detection and tracking in digital video. *IEEE Transactions on Image Processing*, 9(1), pp. 147–156.

Li, H., Doermann, D.S. & Kia, O.E. (1998). Text Extraction, Enhancement and OCR in Digital Video. In *Proceedings of the 1998 International Workshop on Document Analysis Systems*, pp. 363–377.

Li, L. & Tan, C.L. (2010). Recognizing Planar Symbols with Severe Perspective Deformation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(4), pp. 755–762.

Liang, J., Doermann, D. & Li, H. (2005). Camera-based Analysis of Text and Documents: A Survey. *International Journal on Document Analysis and Recognition*, 7(2), pp. 84–104.

Lienhart, R. (2003). Video OCR: A Survey and Practitioner's Guide. In *Video Mining*, pp. 155–184.

Lienhart, R. & Wernicke, A. (2002). Localizing and Segmenting Text in Images and Videos. *IEEE Transactions on Circuits and Systems for Video Technology*, 12(4), pp. 256–268.

Liu, A., Jinghao Fei, Jianping Fan, Lin Pang, Yongdong Zhang & Jintao Li (2009). Confusion network based Video OCR post-processing approach. In *Proceedings of the 2009 International Conference on Multimedia and Expo*, pp. 137–140.

Liu, C., Wang, C. & Dai, R. (2005). Text Detection in Images Based on Unsupervised Classification of Edge-based Features. In *Proceedings of the 2005 International Conference on Document Analysis and Recognition*, pp. 610–614.

Liu, Q., Jung, C. & Moon, Y. (2006). Text segmentation based on stroke filter. In *Proceedings of the 2006 ACM International Conference on Multimedia*, pp. 129–132.

Liu, X. & Wang, W. (2010). Extracting captions from videos using temporal feature. In *Proceedings of the 2010 ACM International Conference on Multimedia*, pp. 843–846.

Liu, X., Wang, W. & Zhu, T. (2010). Extracting Captions in Complex Background from Videos. In *Proceedings of the 2010 International Conference on Pattern Recognition*, pp. 3232–3235.

Lowe, D.G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal on Computer Vision*, 60(2), pp. 91–110.

Lucas, S.M. (2005). ICDAR 2005 text locating competition results. In *Proceedings of the 2005 International Conference on Document Analysis and Recognition*, pp. 80–84.

Lucas, S.M., Panaretos, A., Sosa, L., Tang, A., Wong, S. & Young, R. (2003). ICDAR 2003 Robust Reading Competitions. In *Proceedings of the 2003 International Conference on Document Analysis and Recognition*, pp. 682–687.

Luong, H. & Philips, W. (2008). Robust reconstruction of low-resolution document images by exploiting repetitive character behaviour. *International Journal on Document Analysis and Recognition*, 11(1), pp. 39–51.

Lyu, M.R., Song, J. & Cai, M. (2005). A Comprehensive Method for Multilingual Video Text Detection, Localization, and Extraction. *IEEE Transactions on Circuits and Systems for Video Technology*, 15(2), pp. 243–255.

Maji, S., Berg, A.C. & Malik, J. (2013). Efficient Classification for Additive Kernel SVMs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(1), pp. 66–77.

Mancas-Thillou, C. (2006). *Natural Scene Text Understanding*. PhD Thesis. Belgium: Faculte Polytechnique de Mons.

Mancas-Thillou, C. & Gosselin, B. (2007). Color text extraction with selective metric-based clustering. *Computer Vision and Image Understanding*, 107(1-2), pp. 97–107.

Mariano, V.Y. & Kasturi, R. (2000). Locating Uniform-Colored Text in Video Frames. In *Proceedings of the 2000 International Conference on Pattern Recognition*, pp. 539–542.

Matas, J., Chum, O., Urban, M. & Pajdla, T. (2002). Robust Wide Baseline Stereo from Maximally Stable Extremal Regions. In *Proceedings of the 2002 British Machine Vision Conference*, pp. 384–393.

Merino, C. & Mirmehdi, M. (2007). A Framework Towards Realtime Detection and Tracking of Text. In *Proceedings of the 2007 International Workshop on Camera-Based Document Analysis and Recognition*, pp. 10–17.

Miao, G., Zhu, G., Jiang, S., Huang, Q., Xu, C. & Gao, W. (2007). A Real-Time Score Detection and Recognition Approach for Broadcast Basketball Video. In *Proceedings of the 2007 International Conference on Multimedia and Expo*, pp. 1691–1694.

Mikolajczyk, K., Tuytelaars, T., Schmid, C., Zisserman, A., Matas, J., Schaffalitzky, F., Kadir, T. & Van Gool, L. (2005). A Comparison of Affine Region Detectors. *International Journal on Computer Vision*, 65(1-2), pp. 43–72.

Minetto, R., Thome, N., Cord, M., Leite, N.J. & Stolfi, J. (2011). SnooperTrack: Text Detection and Tracking for Outdoor Videos. In *Proceedings of the 2011 International Conference on Image Processing*, pp. 505–508.

Mishra, A., Alahari, K. & Jawahar, C.V. (2012a). Scene Text Recognition using Higher Order Language Priors. In *Proceedings of the 2012 British Machine Vision Conference*, pp. 1–11.

Mishra, A., Alahari, K. & Jawahar, C.V. (2012b). Top-Down and Bottom-up Cues for Scene Text Recognition. In *Proceedings of the 2012 Conference on Computer Vision and Pattern Recognition*, pp. 2687–2694.

Mita, T. & Hori, O. (2001). Improvement of Video Text Recognition by Character Selection. In *Proceedings of the 2001 International Conference on Document Analysis and Recognition*, pp. 1089–1093.

Mohri, M., Pereira, F. & Riley, M. (2002). Weighted finite-state transducers in speech recognition. *Computer Speech & Language*, 16(1), pp. 69–88.

Mosleh, A., Bouguila, N. & Hamza, A.B. (2012). Image Text Detection Using a Bandlet-Based Edge Detector and Stroke Width Transform. In *Proceedings of the 2012 British Machine Vision Conference*, pp. 1–12.

Myers, G.K., Bolles, R.C., Luong, Q.-T., Herson, J.A. & Aradhye, H.B. (2005). Rectification and Recognition of Text in 3-D Scenes. *International Journal of Document Analysis and Recognition*, 7(2-3), pp. 147–158.

Nagy, R., Dicker, A. & Meyer-Wegener, K. (2011). NEOCR: A Configurable Dataset for Natural Image Text Recognition. In *Proceedings of the 2011 International Workshop on Camera-Based Document Analysis and Recognition*, pp. 150–163.

Neumann, L. & Matas, J. (2010). A Method for Text Localization and Recognition in Real-world Images. In *Proceedings of the 2010 Asian Conference on Computer Vision*, pp. 770–783.

Neumann, L. & Matas, J. (2012). Real-Time Scene Text Localization and Recognition. In *Proceedings of the 2012 Conference on Computer Vision and Pattern Recognition*, pp. 3538–3545.

Neumann, L. & Matas, J. (2013). Scene Text Localization and Recognition with Oriented Stroke Detection. In *Proceedings of the 2013 International Conference on Computer Vision*, pp. 97–104.

Neumann, L. & Matas, J. (2011). Text Localization in Real-world Images using Efficiently Pruned Exhaustive Search. In *Proceedings of the 2011 International Conference on Document Analysis and Recognition*, pp. 687–691.

Newman, W., Dance, C., Taylor, A., Taylor, S., Taylor, M. & Aldhous, T. (1999). CamWorks: A Video-Based Tool for Efficient Capture from Paper Source Documents. In *Proceedings of the 1999 International Conference on Multimedia Computing and Systems*, pp. 647–653.

Ngo, C.W. & Chan, C.K. (2005). Video text detection and segmentation for optical character recognition. *Multimedia Systems*, 10(3), pp. 261–272.

Niblack, W. (1986). *An Introduction to Digital Image Processing*, New Jersey: Prentice Hall.

Novikova, T., Barinova, O., Kohli, P. & Lempitsky, V. (2012). Large-Lexicon Attribute-Consistent Text Recognition in Natural Images. In *Proceedings of the 2012 European Conference on Computer Vision*, pp. 752–765.

Ntirogiannis, K., Gatos, B. & Pratikakis, I. (2011). Binarization of Textual Content in Video Frames. In *Proceedings of the 2011 International Conference on Document Analysis and Recognition*, pp. 673–677.

Otsu, N. (1979). A Threshold Selection Method from Gray-Level Histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1), pp. 62–66.

Ozuysal, M., Fua, P. & Lepetit, V. (2007). Fast Keypoint Recognition in Ten Lines of Code. In *Proceedings of the 2007 Conference on Computer Vision and Pattern Recognition*, pp. 1–8.

Pan, Y.-F., Hou, X. & Liu, C.-L. (2011). A Hybrid Approach to Detect and Localize Texts in Natural Scene Images. *IEEE Transactions on Image Processing*, 20(3), pp. 800–813.

Pan, Y.-F., Hou, X. & Liu, C.-L. (2008). A Robust System to Detect and Localize Texts in Natural Scene Images. In *Proceedings of the 2008 International Workshop on Document Analysis Systems*, pp. 35–42.

Pan, Y.-F., Hou, X. & Liu, C.-L. (2009). Text Localization in Natural Scene Images Based on Conditional Random Field. In *Proceedings of the 2009 International Conference on Document Analysis and Recognition*, pp. 6–10.

Phan, T.Q., Shivakumara, P., Lu, T. & Tan, C.L. (2013a). Recognition of Video Text Through Temporal Integration. In *Proceedings of the 2013 International Conference on Document Analysis and Recognition*, pp. 589–593.

Phan, T.Q., Shivakumara, P., Su, B. & Tan, C.L. (2011). A Gradient Vector Flow-Based Method for Video Character Segmentation. In *Proceedings of the 2011 International Conference on Document Analysis and Recognition*, pp. 1024–1028.

Phan, T.Q., Shivakumara, P. & Tan, C.L. (2009). A Laplacian Method for Video Text Detection. In *Proceedings of the 2009 International Conference on Document Analysis and Recognition*, pp. 66–70.

Phan, T.Q., Shivakumara, P. & Tan, C.L. (2012). Detecting Text in the Real World. In *Proceedings of the 2012 ACM International Conference on Multimedia*, pp. 765–768.

Phan, T.Q., Shivakumara, P., Tian, S. & Tan, C.L. (2013b). Recognizing Text with Perspective Distortion in Natural Scenes. In *Proceedings of the 2013 International Conference on Computer Vision*, pp. 569–576.

Pilu, M. & Pollard, S. (2002). A light-weight text image processing method for handheld embedded cameras. In *Proceedings of the 2002 British Machine Vision Conference*, pp. 1–10.

Povey, D., Hannemann, M., Boulianne, G., Burget, L., Ghoshal, A., Janda, M., Karafiát, M., Kombrink, S., Motlícek, P., Qian, Y., Riedhammer, K., Veselý, K. & Vu, N.T. (2012). Generating exact lattices in the WFST framework. In *Proceedings of the 2012 International Conference on Acoustics, Speech and Signal Processing*, pp. 4213–4216.

Qian, X., Liu, G., Wang, H. & Su, R. (2007). Text detection, localization, and tracking in compressed video. *Image Communication*, 22(9), pp. 752–768.

Rusinol, M., Aldavert, D., Toledo, R. & Llados, J. (2011). Browsing Heterogeneous Document Collections by a Segmentation-Free Word

Spotting Method. In *Proceedings of the 2011 International Conference on Document Analysis and Recognition*, pp. 63–67.

Saidane, Z. & Garcia, C. (2008). An Automatic Method for Video Character Segmentation. In *Proceedings of the 2008 International Conference on Image Analysis and Recognition*, pp. 557–566.

Sarawagi, S. & Cohen, W.W. (2004). Semi-Markov conditional random fields for information extraction. In *Proceedings of the 2004 Conference on Neural Information Processing Systems*, pp. 1185–1192.

Sato, T., Kanade, T., Hughes, E.K. & Smith, M.A. (1998). Video OCR for Digital News Archive. In *Proceedings of the 1998 International Workshop on Content-Based Access of Image and Video Databases*, pp. 52–60.

Sato, T., Kanade, T., Hughes, E.K., Smith, M.A. & Satoh, S. (1999). Video OCR: indexing digital news libraries by recognition of superimposed captions. *Multimedia Systems*, 7(5), pp. 385–395.

Sauvola, J. & Pietikäinen, M. (2000). Adaptive document image binarization. *Pattern Recognition*, 33(2), pp. 225–236.

Sharma, N., Shivakumara, P., Pal, U., Blumenstein, M. & Tan, C.L. (2012). A New Method for Arbitrarily-Oriented Text Detection in Video. In *Proceedings of the 2012 International Workshop on Document Analysis Systems*, pp. 74–78.

Sharma, N., Shivakumara, P., Pal, U., Blumenstein, M. & Tan, C.L. (2013). A New Method for Character Segmentation from Multi-oriented Video Words. In *Proceedings of the 2013 International Conference on Document Analysis and Recognition*, pp. 413–417.

Shi, C., Wang, C., Xiao, B., Zhang, Y. & Gao, S. (2013a). Scene text detection using graph model built upon maximally stable extremal regions. *Pattern Recognition Letters*, 34(2), pp. 107–116.

Shi, C., Wang, C., Xiao, B., Zhang, Y., Gao, S. & Zhang, Z. (2013b). Scene Text Recognition Using Part-Based Tree-Structured Character Detection. In *Proceedings of the 2013 Conference on Computer Vision and Pattern Recognition*, pp. 2961–2968.

Shi, C., Xiao, B., Wang, C. & Zhang, Y. (2012). Graph-Based Background Suppression for Scene Text Detection. In *Proceedings of the 2012 International Workshop on Document Analysis Systems*, pp. 210–214.

Shi, J. & Tomasi, C. (1994). Good features to track. In *Proceedings of the 1994 Conference on Computer Vision and Pattern Recognition*, pp. 593–600.

Shiratori, H., Goto, H. & Kobayashi, H. (2006). An Efficient Text Capture Method for Moving Robots Using DCT Feature and Text Tracking. In *Proceedings of the 2006 International Conference on Pattern Recognition*, pp. 1050–1053.

Shivakumara, P., Bhowmick, S., Su, B., Tan, C.L. & Pal, U. (2011a). A New Gradient Based Character Segmentation Method for Video Text

Recognition. In *Proceedings of the 2011 International Conference on Document Analysis and Recognition*, pp. 126–130.

Shivakumara, P., Phan, T.Q. & Tan, C.L. (2011b). A Laplacian Approach to Multi-Oriented Text Detection in Video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 33(2), pp. 412–419.

Smith, D.L., Field, J. & Learned-Miller, E. (2011). Enforcing Similarity Constraints with Integer Programming for Better Scene Text Recognition. In *Proceedings of the 2011 Conference on Computer Vision and Pattern Recognition*, pp. 73–80.

Smith, S.M. & Brady, J.M. (1997). SUSAN—A New Approach to Low Level Image Processing. *International Journal of Computer Vision*, 23(1), pp. 45–78.

Sobottka, K., Bunke, H. & Kronenberg, H. (1999). Identification of Text on Colored Book and Journal Covers. In *Proceedings of the 1999 International Conference on Document Analysis and Recognition*, pp. 57–63.

Sochman, J. & Matas, J. (2005). WaldBoost - learning for time constrained sequential detection. In *Proceedings of the 2005 Conference on Computer Vision and Pattern Recognition*, pp. 150–156.

Su, B., Lu, S. & Tan, C.L. (2010). Binarization of historical document images using the local maximum and minimum. In *Proceedings of the 2010 International Workshop on Document Analysis Systems*, pp. 159–166.

Sun, Q. & Lu, Y. (2012). Text Location for Scene Image with Inherent Features. In *Proceedings of the 2012 Chinese Conference on Pattern Recognition*, pp. 522–529.

Tanaka, M. & Goto, H. (2008). Text-tracking wearable camera system for visually-impaired people. In *Proceedings of the 2008 International Conference on Pattern Recognition*, pp. 1–4.

Tang, X., Gao, X., Liu, J. & Zhang, H. (2002). A spatial-temporal approach for video caption detection and recognition. *IEEE Transactions on Neural Networks*, 13(4), pp. 961–971.

Teo, B.C., Ghosh, D. & Ranganath, S. (2004). Video-text extraction and recognition. In *Proceedings of the 2004 IEEE Region 10 Conference*, pp. 319–322.

Due Trier, Ø., Jain, A.K. & Taxt, T. (1996). Feature extraction methods for character recognition-A survey. *Pattern Recognition*, 29(4), pp. 641–662.

Tse, J., Jones, C., Curtis, D. & Yfantis, E. (2007). An OCR-Independent Character Segmentation Using Shortest-Path in Grayscale Document Images. In *Proceedings of the 2007 International Conference on Machine Learning and Applications*, pp. 142–147.

Viterbi, A.J. (1967). Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory*, 13(2), pp. 260–269.

Wang, F., Ngo, C.-W. & Pong, T.-C. (2008). Structuring low-quality videotaped lectures for cross-reference browsing by video text analysis. *Pattern Recognition*, 41(10), pp. 3257–3269.

Wang, J. & Jean, J. (1993). Segmentation of merged characters by neural networks and shortest-path. In *Proceedings of the 1993 ACM/SIGAPP Symposium on Applied Computing: States of the Art and Practice*, pp. 762–769.

Wang, K., Babenko, B. & Belongie, S. (2011). End-to-End Scene Text Recognition. In *Proceedings of the 2011 International Conference on Computer Vision*, pp. 1457–1464.

Wang, K. & Belongie, S. (2010). Word Spotting in the Wild. In *Proceedings of the 2010 European Conference on Computer Vision*, pp. 591–604.

Wang, R., Jin, W. & Wu, L. (2004). A Novel Video Caption Detection Approach Using Multi-Frame Integration. In *Proceedings of the 2004 International Conference on Pattern Recognition*, pp. 449–452.

Wang, T., Wu, D.J., Coates, A. & Ng, A.Y. (2012). End-to-End Text Recognition with Convolutional Neural Networks. In *Proceedings of the 2012 International Conference on Pattern Recognition*, pp. 3304–3308.

Weinman, J.J. & Learned-Miller, E. (2006). Improving Recognition of Novel Input with Similarity. In *Proceedings of the 2006 Conference on Computer Vision and Pattern Recognition*, pp. 308–315.

Weinman, J.J., Learned-Miller, E. & Hanson, A.R. (2009). Scene Text Recognition Using Similarity and a Lexicon with Sparse Belief Propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(10), pp. 1733–1746.

Wen, S., Song, Y., Zhang, Y. & Yu, Y. (2012). A Phase-Based Approach for Caption Detection in Videos. In *Proceedings of the 2012 Asian Conference on Computer Vision*, pp. 408–419.

Wernicke, A. & Lienhart, R. (2000). On the segmentation of text in videos. In *Proceedings of the 2000 International Conference on Multimedia and Expo*, pp. 1511–1514.

Wolf, C. (2003). *Text detection in images taken from video sequences for semantic indexing*. PhD Thesis. INSA de Lyon.

Wolf, C. & Doermann, D. (2002). Binarization of Low Quality Text using a Markov Random Field Model. In *Proceedings of the 2002 International Conference on Pattern Recognition*, pp. 160–163.

Wolf, C., Jolion, J.-M. & Chassaing, F. (2002). Text localization, enhancement and binarization in multimedia documents. In *Proceedings of the 2002 International Conference on Pattern Recognition*, pp. 1037–1040.

Wong, E.K. & Chen, M. (2003). A new robust algorithm for video text extraction. *Pattern Recognition*, 36(6), pp. 1397–1406.

Xu, C. & Prince, J.L. (1998). Snakes, Shapes, and Gradient Vector Flow. *IEEE Transactions on Image Processing*, 7(3), pp. 359–369.

Yalniz, I.Z. & Manmatha, R. (2012). An Efficient Framework for Searching Text in Noisy Document Images. In *Proceedings of the 2012 International Workshop on Document Analysis Systems*, pp. 48–52.

Yao, C., Bai, X., Liu, W., Ma, Y. & Tu, Z. (2012). Detecting Texts of Arbitrary Orientations in Natural Images. In *Proceedings of the 2012 Conference on Computer Vision and Pattern Recognition*, pp. 1083–1090.

Ye, Q., Huang, Q., Gao, W. & Zhao, D. (2005). Fast and robust text detection in images and video frames. *Image and Vision Computing*, 23(6), pp. 565–576.

Yi, C. & Tian, Y. (2011). Text String Detection from Natural Scenes by Structure-based Partition and Grouping. *IEEE Transactions on Image Processing*, 20(9), pp. 2594–2605.

Yi, J., Peng, Y. & Xiao, J. (2009). Using Multiple Frame Integration for the Text Recognition of Video. In *Proceedings of the 2009 International Conference on Document Analysis and Recognition*, pp. 71–75.

Yoshimura, H., Etoh, M., Kondo, K. & Yokoya, N. (2000). Gray-scale character recognition by Gabor jets projection. In *Proceedings of the 2000 International Conference on Pattern Recognition*, pp. 335–338.

Zhang, D. & Chang, S.-F. (2003). A Bayesian framework for fusing multiple word knowledge models in videotext recognition. In *Proceedings of the 2003 Conference on Computer Vision and Pattern Recognition*, pp. 528–533.

Zhang, D., Rajendran, R.K. & Chang, S.-F. (2002). General and domain-specific techniques for detecting and recognizing superimposed text in video. In *Proceedings of the 2002 International Conference on Image Processing*, pp. 22–25.

Zhang, J. & Kasturi, R. (2008). Extraction of Text Objects in Video Documents: Recent Progress. In *Proceedings of the 2008 International Workshop on Document Analysis Systems*, pp. 5–17.

Zhang, Y. & Lai, J. (2012). Arbitrarily oriented text detection using geodesic distances between corners and skeletons. In *Proceedings of the 2012 International Conference on Pattern Recognition*, pp. 1896–1899.

Zhao, X., Lin, K.-H., Fu, Y., Hu, Y., Liu, Y. & Huang, T.S. (2011). Text From Corners: A Novel Approach to Detect Text and Caption in Videos. *IEEE Transactions on Image Processing*, 20(3), pp. 790–799.

Zheng, Q., Chen, K., Zhou, Y., Gu, C. & Guan, H. (2010). Text Localization and Recognition in Complex Scenes Using Local Features. In *Proceedings of the 2010 Asian Conference on Computer Vision*, pp. 121–132.

Zhong, Y., Karu, K. & Jain, A.K. (1995). Locating text in complex color images. In *Proceedings of the 1995 International Conference on Document Analysis and Recognition*, pp. 146–149.

Zhong, Y., Zhang, H. & Jain, A.K. (2000). Automatic Caption Localization in Compressed Video. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(4), pp. 385–392.

Zhou, J., Lei Xu, Baihua Xiao, Ruwei Dai & Si si (2007). A robust system for text extraction in video. In *Proceedings of the 2007 International Conference on Machine Vision*, pp. 119–124.