IMPROVING TEXT RECOGNITION IN IMAGES OF NATURAL SCENES

A Dissertation Presented

by

JACQUELINE L. FEILD

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

February 2014

School of Computer Science

IMPROVING TEXT RECOGNITION IN IMAGES OF NATURAL SCENES

A Dissertation Presented

by

JACQUELINE L. FEILD

Approved as to style and content by:

Erik G. Learned-Miller, Chair

Allen Hanson, Member

Rui Wang, Member

Adrian Staub, Member

Lori A. Clarke, Chair
School of Computer Science

To Henry, for going on this adventure with me.

ACKNOWLEDGEMENTS

There are so many people I would like to thank for helping me throughout my PhD journey. First, I would like to thank my husband, Henry, for his continuous love and support. Going through this entire process together made success possible, and I am thankful for his understanding. I am also thankful for the incredible amount of support and encouragement I received from my family and friends, including my parents, brother, extended-family, in-laws and ski-family. I am very lucky to have them all in my life.

I would also like to thank my advisor, Erik Learned-Miller, for guiding my research progress. I grew a lot as a researcher during my time in the Vision Lab, and I appreciate his support as I worked towards this goal. I would also like to thank my other committee members, Allen Hanson, Rui Wang and Adrian Staub for their help with my thesis and their unique perspectives.

My time at UMass would have been much more difficult without Leeanne Leclerc, who somehow managed to know the answer to every question I ever asked her. I am grateful for her immense knowledge of how things work in the department and on campus, that she always made sure I got paid while on fellowship, and of course for her incredible support from day one. I also want to thank Laurie Downey and Priscilla Scott for making sure that everything went smoothly over the years.

Lastly, I would like to thank all of my fellow graduate students who helped me navigate through this journey. I am thankful for everyone who shared their experiences, advice, encouragement and friendship with me. I am especially thankful for everyone in the Vision Lab who helped me work through my research ideas, checked on my progress, and supported me to the end.

ABSTRACT

IMPROVING TEXT RECOGNITION IN IMAGES OF NATURAL SCENES

FEBRUARY 2014

JACQUELINE L. FEILD

B.Sc., LOYOLA UNIVERSITY MARYLAND

M.Sc., UNIVERSITY OF MASSACHUSETTS AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor Erik G. Learned-Miller

The area of scene text recognition focuses on the problem of recognizing arbitrary text in images of natural scenes. Examples of scene text include street signs, business signs, grocery item labels, and license plates. With the increased use of smartphones and digital cameras, the ability to accurately recognize text in images is becoming increasingly useful and many people will benefit from advances in this area.

The goal of this thesis is to develop methods for improving scene text recognition. We do this by incorporating new types of information into models and by exploring how to compose simple components into highly effective systems. We focus on three areas of scene text recognition, each with a decreasing number of prior assumptions. First, we introduce two techniques for character recognition, where word and character bounding boxes are assumed. We describe a character recognition system that incorporates similarity information in a novel way and a new language model that models syllables in a word to produce word labels that can be pronounced in English. Next we look at word recognition, where only word bounding boxes are assumed. We develop a new technique for segmenting text for these images called bilateral regression segmentation, and we introduce an open-vocabulary word recognition system that uses a very large web-based lexicon to achieve state of the art recognition performance. Lastly, we remove the assumption that words have been located and describe an end-to-end system that detects and recognizes text in any natural scene image.

TABLE OF CONTENTS

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1

INTRODUCTION

The area of scene text recognition focuses on the problem of recognizing arbitrary text in images of natural scenes. Examples of scene text include street signs, business signs, grocery item labels, and license plates. Potential applications of scene text recognition include improving navigation for people with low vision, recognizing and translating text into other languages, improving image retrieval and aiding autonomous navigation for cars and robots.

This problem is similar to the well studied area of optical character recognition (OCR) for documents. However, unlike images of documents that usually have standard fonts, structured text on a plain background and are usually captured in a controlled setting, images of natural scenes have many characteristics that make them difficult to analyze. They often contain more extreme lighting variation, may include unusual or highly stylized fonts, often vary widely in color and texture and may be captured from a wide range of viewing angles. In addition, scene text images usually contain only a few words, so it is more difficult to benefit from linguistic constraints or to learn repeated patterns of appearance.

Because of these additional challenges, existing solutions for recognizing text in documents do not perform well when applied directly to images of scene text and new techniques that can address these difficulties are needed. The goal of this thesis is to develop improved methods by incorporating new information sources into models and by exploring how to compose simple components into highly effective systems.

This thesis develops techniques for three different areas of scene text recognition, each with a different number of prior assumptions. Character recognition is explored first, which is the problem of producing an output label for each character in an image, assuming that individual character bounding boxes are given as input. Next the focus is on word recognition, which is the similar problem of producing an output label for each word in an image, assuming that word bounding boxes are given as input. This is sometimes also referred to as cropped word recognition. Finally, the prior assumption that text has already been located is removed and the problem of end-to-end recognition is explored, where all text in an image must be located and labelled.

This thesis begins by focusing on the area of character recognition, and in Chapter 4 a character recognition system that incorporates similarity information in a novel way is introduced. This is motivated by the idea that characters in the same word that have similar appearances should be given the same label, and characters in the same word that have different appearances should be given different labels. While trying to recognize the correct label for a character image is a difficult task, verifying whether two character images are nearly identical is relatively easy. Although characters may vary widely in appearance across fonts, characters appearing in the same scene text word will almost always be in the same font. Within a font, we expect consistency of character appearance that can be verified with a simple classifier. This allows us to take advantage of similarity information to constrain the space of possible recognized words and produce character labels that are consistent with appearances within a scene text word.

In Chapter 5 a new language model that models syllables in a word to produce word labels that can be pronounced in English is described. While many appearance models have been shown to perform very well for recognizing individual characters, language information is also important for improving recognition results. Many recent approaches incorporate bigram information, which describes how likely a pair of characters is to occur next to each other, into their models. Bigram information is very informative, but it is a highly local source of information. Some bigrams are very uncommon, but will still occur next to each other in a word, often across a syllable boundary. As an example, consider the word ‘Amherst’. The combination of ‘m’ followed by ‘h’ is very rare in English, and as a result the correct spelling has a low probability under a bigram model. To overcome this problem, we describe a new probabilistic syllable model that encapsulates information about syllables in English, and forces output labelings to be consistent with a grammar. This better models language information across syllable boundaries and each labeling is pronounceable in English. This is important for the domain of scene text recognition where many of the words are proper nouns that are not likely to be in a standard dictionary.

While these techniques improve performance for character recognition, removing the assumption that characters have been located a priori, and focusing on the problem of word recognition, will make these techniques more useful in the real-world. In Chapter 6 a technique for segmenting text pixels from the background in natural scene images of cropped words is developed, which will allow characters to be found without needing to know their location a priori. This technique is motivated by two observations. The first is that many scene text images have a consistent foreground color, but may have complex backgrounds. The second is that often images contain smooth color changes caused by lighting, causing foreground text to vary widely from one part of the scene to another. These images are not handled well by existing segmentation techniques that rely on clustering colors. Our approach is to use a regression model that allows us to model smooth color changes. We only model the foreground of each image, since the background is often complex and difficult to model well. Our segmentation method provides a way to model the foreground pixels closely while ignoring pixels that belong to the background, even though they are spatially adjacent. We also describe an effective word recognition system that combines segmentation with simple, yet effective, components for recognition. We evaluate this system on the problem of word spotting, where recognized words come from a small, pre-specified lexicon of valid labels.

While many existing approaches also require that recognized words be drawn from a lexicon, this assumption is restrictive for scene text, since many words are proper nouns that are not in a standard lexicon. To remove this assumption, in Chapter 7 an open-vocabulary word recognition system that does not require recognized words to be drawn from a lexicon is introduced. The web is a rich source of language information and contains a collection of dictionary words and proper nouns that is constantly being updated. Existing methods obtain web-based language information by sending queries to a search engine to collect document frequency information for each query. This process can be slow and expensive for researchers, due to the currently available APIs. To overcome these limitations, we demonstrate the use of a static N-Gram Data Set released by Google. It includes approximately 13.5 million words that occur in a crawl of the web and their term frequencies, which we use as a probabilistic lexicon. We incorporate this language information into a full system, including segmentation and a simple, efficient method for text recognition.

Finally, in Chapter 8 the assumption that text has already been located is removed and an end-to-end scene text recognition system for automatically finding and labeling text in images of natural scenes is described. We begin by combining the recognition method from Chapter 7 with a state-of-the-art text detection technique. Next, we take advantage of the fact that we are performing text detection and recognition together, and we use recognition information to perform image-specific parameter adaptation in the text detection step to significantly improve performance. We also incorporate this technique into a hybrid system that combines our recognition system with an open-source recognition system to further improve performance. We evaluate this system on the task of end-to-end text recognition and show promising results.

1.1 Contributions

This thesis includes the following contributions:

- Chapter 4
  - A description of a novel technique for incorporating similarity information
  - A description of an effective character recognition system
  - An evaluation of this system compared to state-of-the-art systems on a public data set
- Chapter 5
  - A novel language model that produces word labels that can be pronounced in English
  - An analysis of using this model for character recognition, compared to a bigram model and a dictionary model
- Chapter 6
  - A new model for segmentation in scene text images called bilateral regression segmentation
  - A description of a complete system for word spotting with a pre-specified lexicon
  - An evaluation of bilateral regression segmentation compared to existing methods
  - An evaluation of the complete system compared to state-of-the-art methods on standard data sets

- Chapter 7
  - A demonstration of a new approach for incorporating web-based language information
  - A description of an efficient system for open vocabulary word recognition using a very large lexicon of over 13.5 million words.
  - An evaluation of our system against state-of-the-art methods on standard data sets
- Chapter 8
  - A demonstration of a novel framework for automatic image-specific parameter adaptation
  - A description and evaluation of an end-to-end system with state-of-the-art performance on standard data sets

CHAPTER 2

RELATED WORK

There is existing work on many different subproblems of scene text recognition. In this chapter we will define each problem, discuss the assumptions that are made, and describe existing work in that area. We will also explain how our work is different from existing methods.

2.1 Character Recognition

Character recognition is the problem of predicting a label for each character in a scene text image. It assumes that character locations described by bounding boxes are given, so the tasks of text detection and text segmentation are assumed to be complete. This makes the problem significantly easier, as neither detection or segmentation is a solved problem, but allows appearance and language models to be evaluated in an ideal setting.

De Campos et al. present an evaluation of six types of local feature descriptors with a bag of words approach [13]. Similarly, Yi et al. present a performance evaluation of five existing feature descriptors, and look at the effect of two sampling methods, five dictionary sizes, four types of coding schemes and two different SVM kernels on character recognition performance [72]. Tian et al. also focus on feature descriptors and extend the histogram of oriented gradients descriptor in co-occurance HOG descriptors, which capture the gradient orientation of neighboring pixel pairs to improve recognition [56]. Weinman et al. incorporate many different sources of information into their recognition process, such as character appearance, bigram frequencies, similarity and lexicons [66]. Other approaches include convolutional neural networks that require no preprocessing [50] and a technique that uses image binarization followed by GAT correlation [75]. Donoser et al. also show that character recognition results can be improved using information from a web search engine [15]. The current state-of-the-art character recognition results on the ICDAR 2003 data set are presented by Coates et al. [12]. They take an unsupervised approach to learning features from unlabeled data.

We present a contribution to character recognition that is inspired by Wienman et al. and Donoser et al. [66, 15]. Weinman et al. showed that similarity information can be used to improve recognition performance, but their similarity features were not designed specifically for scene text data. We have designed a similarity classifier that is trained to predict equivalence between two scene text characters and we show that we can improve character recognition performance over state-of-the-art using this information. Following the technique of Donoser et al., we also incorporate global language information from a search engine.

2.2 Text Segmentation

The problem of text segmentation is to decide if each pixel in a cropped word image is part of the text or the background. The output of a text segmentation algorithm is a binary image that labels each pixel with the class it belongs to. Those pixels that are part of the text can then be grouped into connected components and treated as characters, allowing for text recognition without knowing character bounding box locations a priori.

Many of the methods developed to solve this problem cluster colors in the image to produce several possible segmentations, then choose the one that is most likely to be correct [55, 60, 26]. Similarly, Wang et al. [64] extract color information from confident text regions and use it to create segmentations. Mishra et al. [38] also extract foreground and background colors, and use an MRF model in an iterative graph cut framework.

Another recent method by Zhou et al. [77] takes a different approach and estimates rendering parameters and illumination effects to improve segmentation accuracy. They use iterative optimization to find the best parameters for the light source, material properties and blur kernel size and use that information to inform segmentation decisions for each pixel. In addition, since the parameters are learned, new text images can be synthesized that mimic the appearance of an existing image.

The segmentation approach we present in this thesis is similar to the clustering methods, but we use color clustering as a starting point to fit a regression model for each image. When colors change across an image, as often occurs in scene text images due to lighting, pixels that belong to the foreground text are not well modeled by the unimodal localized distributions (like the Gaussians) usually used in clustering. Using a regression model allows us to segment a larger class of images, since we can model the smooth color changes.

2.3 Word Spotting

Word Spotting is the problem of selecting a word label for a scene text word image from a pre-specified lexicon. This assumes that we are given a cropped image of a word, so text detection is complete. This also assumes that we have a pre-specified, and usually small (50 or 1000 words), lexicon that contains the true word label. This is a simplified version of word recognition, where a system can 'spot' words in the images. An example application is identifying specific keywords to direct a low-vision user during a navigational task, like the word 'RESTAURANT'.

This problem was introduced for the problem of scene text by Wang et al. at ECCV 2010 [62, 61]. They presented an end-to-end word spotting system using random ferns and pictorial structures, along with a new data set for word spotting called Street View

Text. Others have also approached this problem by combining bottom-up and top down cues [40] and using unsupervised feature learning combined with a convolutional neural network in an end-to-end system [63]. Novikova et al. present a system for word spotting with a large lexicon of around 100,000 words. They model visual and lexicon information in one model using weighted finite-state transducers [45]. Goel et al. take a different approach and instead of detecting character locations they present a system that matches input images to synthetic images generated from the lexicon words [18]. They use gradient-based features with a novel weighted dynamic time warping approach.

We present a word spotting system that uses a novel segmentation method designed for scene text images combined with standard, but effective, recognition methods. One of the main differences between the work we present and these existing solutions is the technique used to detect character locations. These methods use a sliding window approach to evaluate all possible locations and sizes to find possible characters. They avoid relying on an initial hard segmentation step, but evaluating all sub-windows is expensive, and there is great potential for confusion when non-text areas exhibit character-like features. In contrast, a text segmentation based method like ours can take advantage of coherence across an image. For example, the color characteristics of easier characters can help identify more difficult characters. We demonstrate that a segmentation-based approach can outperform sliding-window based approaches for the task of word spotting.

2.4 Open-Vocabulary Word Recognition

Word recognition with an open vocabulary is the problem of predicting a label for a word in a scene text word image. The word label does not have to occur in any lexicon. The only assumption of this problem is that we are given a cropped word image, so text detection is complete. This problem is significantly more difficult than word spotting.

To solve this problem, Weinman et al. integrate both character segmentation and recognition using a semi-Markov model and character width information [65]. Mishra et al. use higher order language priors to improve open-vocabulary word recognition accuracy [39]. In addition, Neumann et al. have demonstrated a real-time solution using extremal regions [41, 42]. Kumar et al. have also shown increased performance on this task by developing specialized segmentation techniques and using commercial OCR systems for recognition. These methods include segmenting the middle rows of each image first, and propagating labels to the rest of the pixels [27] and using a non-linear enhancement with image plane selection [28]. Very recently, researchers from Google have demonstrated significantly improved state-of-the-art performance on this problem using a deep-learning approach for character classification and large amounts of training data and computational resources [4].

We present an open-vocabulary word recognition system that uses a large web-based lexicon, since many scene text words are proper nouns that are not likely to occur in a standard lexicon. Web based language information was first used by Donoser et al. [15] and was used to create dynamic dictionaries for handwriting recognition by Oprean et al. [46]. Instead of querying a search engine to obtain language information as they did, which is slow and expensive, we demonstrate the use of an n-gram data set created from a crawl of the web. This gives us a similar type of information but is fast to use. We show that we can combine this rich source of global language information with a standard recognition technique to improve performance on this problem.

2.5 Text Detection

Text detection is the problem of locating text in an image. Given any image, a text detector must return the bounding boxes of all text in the image. There are no assumptions made about size, orientation or location of text. Text detection in natural scene images is difficult for many of the same reasons that recognition is difficult including complex backgrounds, variation of text due to lighting and perspective distortion and low quality images. Text detection is a very active research area and this section does not include a comprehensive list of existing methods. Instead, we discuss the main approaches and include descriptions of related techniques as examples.

There are two main approaches to the problem of text detection. Most current methods use a sliding-window based approach or a connected-component based approach. The sliding window based approaches scan the entire image with a sliding window, classifying each patch as text or non-text using local features. They usually do this at multiple scales. Chen et al. use an Adaboost classifier with gradient-based features [10]. Lee et al. also use a variant of Adaboost, only with local gabor filters and texture features [32]. Wang et al. use a random fern classifier and histogram of oriented gradients features [61]. Wang et al. learn a set of low-level features and use a convolutional neural network for classification [63].

Connected-component based approaches identify character candidates using a variety of local features, then group characters together into text lines. The elimination of non-text regions is usually rule-based. Chen et al. detect characters using multi-resolution edge detection and color analysis along with geometric analysis and affine rectification [9]. Phan et al. and Bai et al. use gradient-based features for character detection [48, 2]. Epshtein et al. introduce the stroke width transform and detect characters by looking for connected-components with similar stroke width values [17]. Yi et al. use stroke with and color uniformity to detect character candidates [71]. Yao et al. also use stroke width information and filter non-text components using a random forest classifier to detect text at arbitrary orientations [69]. Neumann et al. show that maximally stable extremal regions (MSERs) are effective for character candidate detection and provide a method for efficiently pruned exhaustive search for text. Recently, MSERs have been shown to perform well in several other systems as well. Yin et al. use MSERs with an adaboost classifier [74] and Ye et al. use MSERs with a support vector machine (SVM) [70]. Shi et al. also use MSERs in a graph-based approach [53].

Both approaches have led to good performance on the problem of text detection. The sliding window based approaches often have simpler architectures and can be more robust to noise, but they are computationally expensive, since a large number of image rectangles must be evaluated at different sizes and aspect ratios. In addition, non-horizontal text detection is difficult using this approach. Alternatively, connected-component based approaches are more efficient and run in real-time. However, many parameters are often required and finding a robust method for filtering out non-text from the large number of initial character candidates is very difficult.

2.6 End-To-End Scene Text Recognition

The full problem of scene text recognition is to predict a word label for all text found in an image. There are no assumptions about what the image may contain, for example it may have multiple lines of text that need to be detected and recognized. There are also no assumptions about the predicted word labels.

Chen et al. present a system to detect and recognize text in city scenes [10]. They use AdaBoost to classify text regions that are then binarized and processed by commercial optical character recognition (OCR) software. Chen et al. present a system to detect, recognize and translate text from Chinese signs [9]. Neumann et al. present a real time text detection and recognition system based on maximally stable extremal regions (MSERs) [41]. They also extend this method for real-time performance [42] and improve recognition performance further by choosing character segmentations later in the pipeline when more information is known about which is the best [43]. Weinman et al. also present an end-to-end system by combining their recognition system with an existing text detection system by Yi et al. [67, 71]. Recently, Milyaev et al. showed that binarizing an image and passing it to a commercial optical character recognition (OCR) system leads to state-of-the-art recognition performance on the ICDAR data sets [37].

We improve end-to-end text recognition by using recognition information to perform image-specific parameter adaptation. Most existing techniques combine text detection and recognition in a feed-forward pipeline, performing both tasks in isolation. Instead, we take advantage of performing both tasks together, and use information from the recognition phase to adapt a key text detector parameter to each image to improve performance.

2.7 ICDAR Competitions

Over the last ten years there have been several Robust Reading competitions held at the International Conference on Document Analysis and Recognition (ICDAR). The first competition in 2003 released standard data sets for the problems of text detection, character recognition and word recognition. Ground truth text detection results were provided for the recognition competitions, so teams could enter either competition separately. There were five entries for the text detection competition and zero entries for the character and word recognition competitions. A description of the data sets and the evaluation of the text detection methods are described in a final report by Lucas et al. [35].

In 2005 the Robust Reading competition included only the task of text detection and received five new entries. Several of these methods are described in the final report by Lucas [34] .

For the next competition in 2011, the original data set was updated, fixed and expanded based on feedback from the community. In addition, a new evaluation method was chosen for the text detection competition. They chose to use the DetEval software which is publicly available from Wolf et al. [68]. The word recognition task was reintroduced and there were nine entries for text detection and three entires for word recognition. Summaries of the methods and results are summarized in the report by Shahab et al. [52].

The most recent competition was in 2013. The data set from 2011 was revised again to fix ground truth errors and remove images duplicated over the training and test sets. Additionally, pixel-level ground truth labels were provided for the first time for text segments. There were competitions for text detection, text segmentation (new), word recognition and reading text in videos (new). There were nine entries to the text detection competition, seven entries for text segmentation, eight entires for word recognition and one entry for reading text in videos. Descriptions of some of these methods and a summary of results are provided in the report by Karatzas et al. [25].

CHAPTER 3

DATA SETS AND EVALUATION PROCEDURES

In this chapter we describe the scene text data sets used in our experimental evaluations in more detail. We also describe how to compute some of the evaluation metrics used in our experiments.

3.1 VIDI Data Set

The VIDI data set was created by Weinman et al. [66] from images of text on signs from around a city. It consists of 95 grayscale sign images with ground truth labels and ground truth character bounding boxes. There are a total of 215 words and 1209 characters in the data set, including digits, lowercase letters, and uppercase letters. The average number of words per sign is 2.26 and the average number of letters per word is 5.62. Sample images from this data set are shown in Figure 3.1.

Published with the VIDI data set is a training set of synthetic character images from different fonts. This includes that characters A-Za-z0-9 in 1866 different fonts. The images contain one character each, with black text on a white background and are all generated in a 128x128 pixel window with the same baseline. Sample images from this data set are shown in Figure 3.2.

3.2 ICDAR 2003 Data Set

The ICDAR 2003 data set was created for the Robust Reading competition [35] and includes images of text in the environment, but not necessarily from a sign. This data set is suitable for the tasks of end-to-end recognition, text localization, word

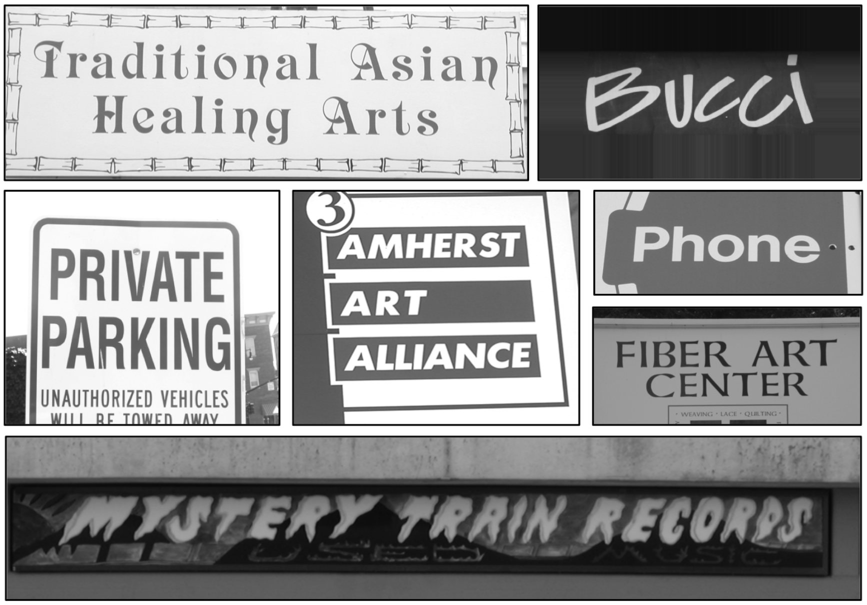


Figure 3.1: Sample images from the VIDI data set.

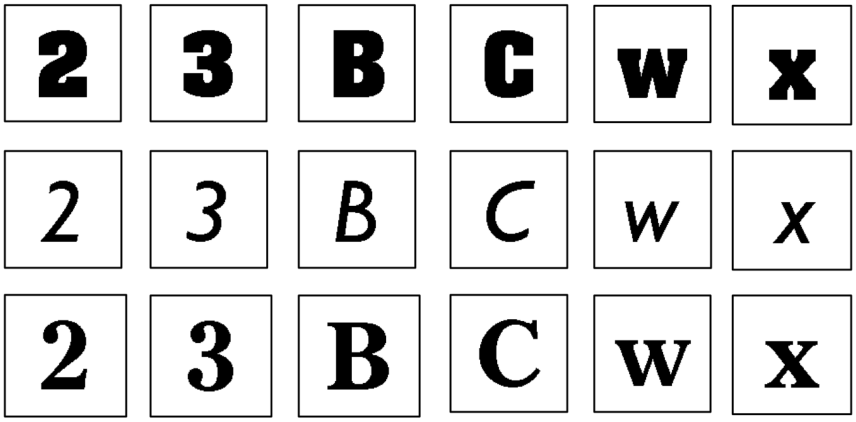


Figure 3.2: Sample images from the VIDI training set of synthetic characters.

recognition and character recognition. There are 251 color images containing 1110 words and 5430 characters. There is also a training set of comparable size. Ground truth information is provided for word bounding box locations and labels as well as character locations and labels for both the training and test sets. Note that unlike the VIDI data set, this data set includes punctuation in ground truth labels. Figure 3.3 shows sample images from this data set.

There is also a version of this data set called the ICDAR 2003 Scene data set which is identical except that it only contains 1107 words. We acknowledge it here because several researchers in this area use this data set instead of the original version.

The task of word spotting requires a lexicon for each word in the data set, but this data set was not created with ground truth lexicons. We follow the experiments of Wang et al. [61] and use two different approaches for creating lexicons to use for this task. The first approach is to use the same lexicon for all words, consisting of the ground truth words for all images in the data set. This is referred to as the ICDAR03(FULL) lexicon. The second approach is to create a lexicon for each word using the ground truth label for that word plus 50 other random words from the data set. This is called the ICDAR03(50) lexicon.

3.3 ICDAR 2011 Data Set

There was also a data set created for the ICDAR 2011 Robust Reading competition [52] which updated, fixed and expanded the ICDAR 2003 data set. This also contain images different types of text in the environment. It contains 255 color images with 1189 words. There is also a training set of a similar size. Ground truth information is provided for word bounding box locations and labels, which makes this suitable for the tasks of end-to-end recognition, text localization and word recognition. Unlike the ICDAR 2003 data set, this does not contain ground truth character location and label information. Sample images from the ICDAR 2011 data set are shown in

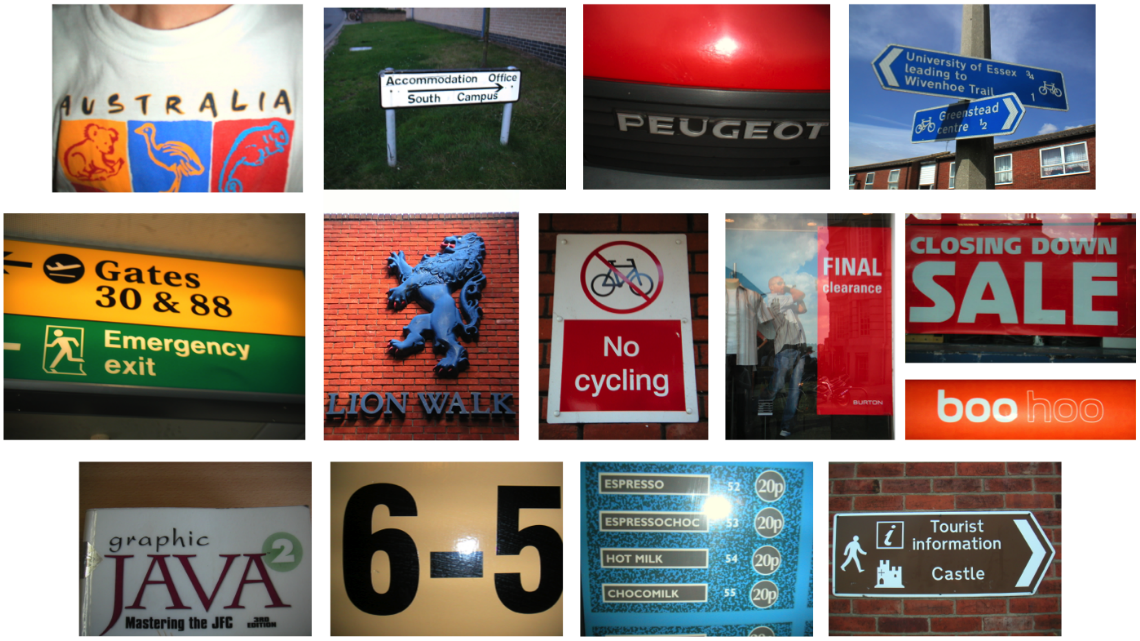


Figure 3.3: Sample images from the ICDAR 2003, 2011 and 2013 data sets.

Figure 3.3. Note that for the task of word spotting, we create the ICDAR11(full) lexicon and the ICDAR11(50) lexicon in the same way we did for the ICDAR 2003 data set.

3.4 ICDAR 2013 Data Set

The latest version of the ICDAR data set is ICDAR 2013 [25]. This is also based on the previous data sets, but ground truth errors are fixed and image duplicates are removed. It contains 233 color images with 1095 words. The training set is of a similar size. Ground truth information is provided for word bounding box locations and labels, which makes this suitable for the tasks of end-to-end recognition, text localization and word recognition. In addition, pixel-level ground truth labels are provided for the first time for text segments, enabling the task of text segmentation. Figure 3.3 shows sample images from this data set.

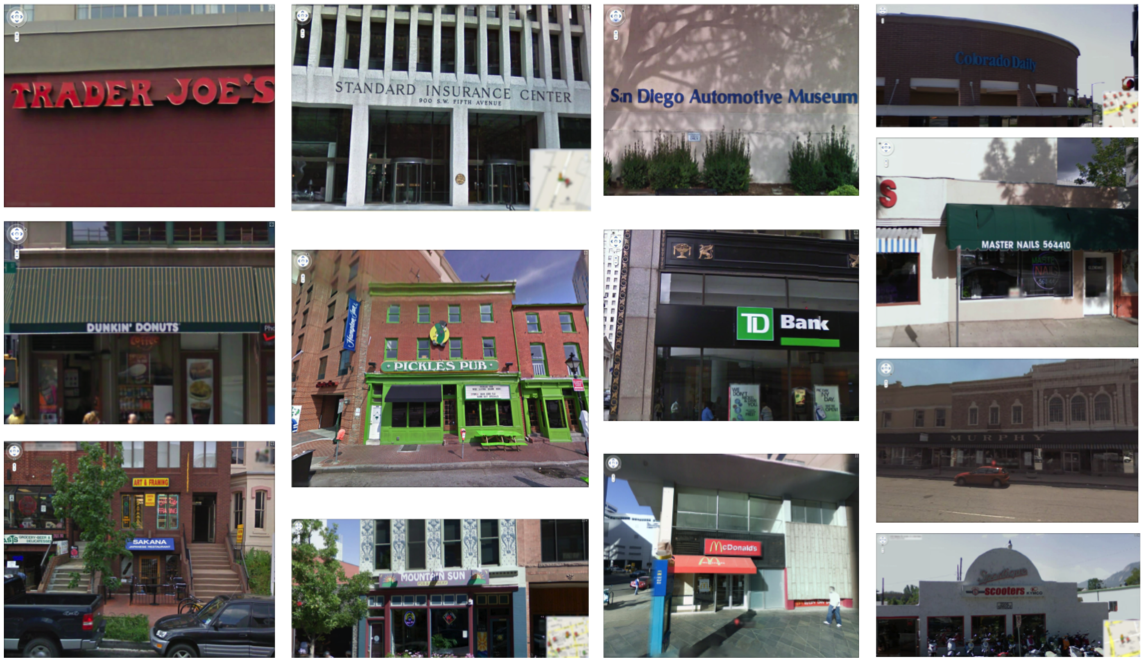


Figure 3.4: Sample images from the SVT data set.

3.5 SVT Data Set

The Street View Text (SVT) data set [62] was designed specifically for the word spotting problem. It consists of 647 words from 250 images. Each image is taken from Google Street View and word bounding box locations and ground truth labels are provided. In addition, there is a lexicon given for every word, which contains the ground truth label plus other local business names obtained from using the 'Search Nearby' feature in Google Maps. The lexicons consist of around 50 unique words each. Sample images from the SVT data set are shown in Figure 3.4.

3.6 End-to-End Evaluation

In our experiments we most often look at the character recognition accuracy, word recognition accuracy and end-to-end precision/recall. The character accuracy is calculated by dividing the number of correctly recognized characters by the total number of characters. The word accuracy is calculated the same way, except with words instead of characters.

For end-to-end recognition accuracy, we follow the procedure used by the first ICDAR 2003 Robust Reading competition. It is described in the competition report [35].

For each detected bounding box, we compute a match score. This is found by calculating how well each detection matches the best ground truth bounding box. A score is computed for each ground truth bounding box by dividing the area of intersection between the two rectangles by the minimum bounding box containing both rectangles. This score has a value of zero when the bounding boxes do not intersect, and a value of one when they are identical. The maximum score over all of the ground truth rectangles is the match score for a detection.

When computing end-to-end recognition results, a detected bounding box is considered to be correct if it matches a ground truth bounding box with a match score greater than .5 and the detected text matches the ground truth text exactly.

CHAPTER 4

USING CHARACTER SIMILARITY INFORMATION TO IMPROVE CHARACTER RECOGNITION

In this chapter we describe a new way to incorporate character similarity information for scene text recognition. We train a similarity expert that learns to classify each pair of character images as equivalent or not. Next we describe a character recognition system that combines this similarity information with appearance and language information. We formulate the search for the maximum likelihood interpretation of a word as an integer program. We incorporate the equivalence information as constraints in the integer program and build an optimization criterion out of appearance features and character bigrams. Finally, we take the optimal solution from the integer program, and compare all nearby solutions using a probability model for strings derived from search engine queries. We evaluate the performance of this system compared to the current state-of-the-art.

4.1 Similarity Information

The use of character similarity information is motivated by the idea that characters in the same word that have similar appearances should be given the same label, and characters in the same word that have different appearances should be given different labels. By identifying these constraints, the space of possible word labels can be reduced. While trying to recognize the correct label for a character image is a difficult task, verifying whether two character images are nearly identical is relatively easy. Although characters may vary widely in appearance across fonts, characters

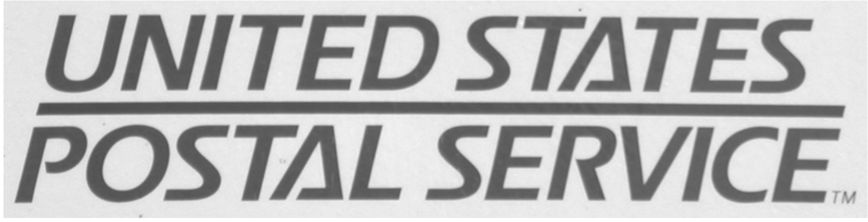


Figure 4.1: An example sign image. Due to the specialized font, the character 'A' is particularly difficult to recognize.

appearing in the same scene text word will almost always be in the same font, where we expect consistency of character appearance. Figure 4.1 shows a sample image where similarity information is useful. The character 'A' is particularly difficult to recognize because it looks like a triangle. Even if we don't know what the label should be, we can use the fact that this character appears in multiple places to constrain the results.

Recently, Weinman et al. showed that using information about similarity among characters can improve scene text recognition [66]. They compute a raw similarity score for each pair of characters A and B by computing

$$1 - f_A \cdot f_B, \tag{4.1}$$

where f_A and f_B are unit feature vectors for the characters A and B. This yields a number between 0 (when the vectors are identical) and 2 (when the vectors point in opposite directions) which is then put through a learned monotonic non-linearity. This similarity score was then used to define factors between each pair of characters in a factor graph, and integrated into a general belief propagation framework using other types of appearance and language information.

This similarity score encodes important information about similarity, but our hypothesis is that we can develop a better score using a classifier that is designed specifically to evaluate the similarity of two character images. We will investigate whether this score will lead to more accurate character recognition. This idea is motivated by recent work on the face verification problem [19, 30] that has shown relatively high accuracy rates in determining whether two faces are of the same person or not. That problem is more difficult than determining whether two characters represent the same letter in the same font under natural viewing conditions.

The goal is to create a classifier that takes two character images as input and predicts whether the characters should have the same label. We use a support vector machine (SVM) [7, 8, 20] trained on a set of feature vectors extracted from pairs of character images. We extract one SIFT descriptor [33, 58] from each image by placing it in the center of the image, scaled to cover the entire image. We also create an alternate version of this classifier by extracting a two by two non-overlapping grid of four SIFT descriptors from each image rather than a single SIFT descriptor. In our experiments, we investigate if this provides a performance increase over the single descriptor version.

We form two feature vectors for each pair of images. One is created by subtracting the SIFT descriptor of the first image from the second and the other is created by subtracting the SIFT descriptor of the second image from the first. We append the ratios of the original image widths and the original image heights to both difference descriptors. We add these additional features because they are good predictors of dissimilarity. If two images vary significantly in their original size, then they are more likely to have different labels than two images with similar sizes. For each pair of training images A and B, we then end up with two training sample vectors: $f(A, B)$ and $f(B, A)$, where f is the augmented SIFT vector described above. While we use both feature vectors in training to generate more training data, at test time we use only the first to represent a pair of images.

4.2 A Complete Character Recognition System

We combine this new similarity information with appearance and language information into a complete character recognition system. We use appearance features developed by Weinman [22] that are given to us pre-computed. They were computed by training class conditional weights over a vector of edge-like features to maximize the classification performance on a set of synthetic fonts. We use language features that combine bigram and case transition statistics. The bigram statistics were trained on a selection of books from Project Gutenberg[1]. Inter-word case change statistics (i.e. changing from upper to lower or lower to upper) were trained on the press-related sections of the Brown Corpus of American English.[2] For the first transition, we assume a uniform probability of transitioning from upper case to lower case and upper case to upper case.

The goal of this system is to find the word labels for each image that maximize the conditional probability of the labels given the observations. Given a set of N character image observations $\mathbf{x} = \{x_1, x_2, \ldots, x_N\}$, the recognition task is to assign the best set of labels $\mathbf{y} = \{y_1, y_2, \ldots, y_N\}$ for these characters subject to a set of consistent equivalence and difference constraints $\mathcal{C}$. That is, we want to compute

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmax}}\, p(\mathbf{y}|\mathbf{x}) = \underset{\mathbf{y}}{\operatorname{argmax}}\, p(\mathbf{y}, \mathbf{x}) \tag{4.2}$$

subject to $\mathcal{C}$. We assume a Markov model over the labels, leading us to express $p(\mathbf{y}, \mathbf{x})$ as

$$p(\mathbf{y}, \mathbf{x}) = \prod_{i=1}^{N} p(x_i|y_i) \prod_{i=1}^{N} p(y_i|y_{i-1}). \tag{4.3}$$

[1] http://www.gutenberg.org

[2] http://icame.uib.no/brown/bcm.html

Rather than maximizing Eq. 4.3, we can equivalently minimize its negative log. Thus,

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmin}} \; -\sum_{i=1}^{N} \log p(x_i|y_i) - \sum_{i=1}^{N} \log p(y_i|y_{i-1}). \tag{4.4}$$

Let $\mathcal{A}$ denote our alphabet. For simplicity, let $\phi_{i:j} = -\log p(x_i|y_i = \mathcal{A}_j)$, the negative log probability that character i takes on the label $\mathcal{A}_j$. Similarly we will let $\phi_{i(i+1):jk} = -\log p(y_{i+1} = \mathcal{A}_k|y_i = \mathcal{A}_j)$, the negative log probability that character i takes on the label $\mathcal{A}_j$ and character $i+1$ takes on the label $\mathcal{A}_k$. Using this notation, we have

$$\mathbf{y}^* = \underset{\mathbf{y}}{\operatorname{argmin}} \; \sum_{i=1}^{N} \phi_{i:j} + \sum_{i=1}^{N-1} \phi_{i(i+1):jk}. \tag{4.5}$$

We use an integer program to find the initial word labeling $\mathbf{y}^*$. We follow that with an error correction process to incorporate global language information. Both methods are described next.

4.2.1 Integer Program Formalization

An integer program (IP) is an optimization problem of a linear objective function over integer-valued variables $\mathbf{y}$, where the space of solutions is bounded by a set of linear constraints. The goal is to find the assignment to these variables that minimizes the objective function. An IP in standard form [3] is written

$$\begin{aligned} \text{minimize } & \mathbf{c}^T\mathbf{y} \\ \text{subject to } & \mathbf{A}\mathbf{y} = \mathbf{b} \\ & \mathbf{y} \geq \mathbf{0} \\ & \mathbf{y} \in \mathbb{Z}^n, \end{aligned}$$

where $\mathbf{c} \in \mathbb{R}^n, \mathbf{b} \in \mathbb{R}^m$, $\mathbf{A} \in \mathbb{R}^m \times \mathbb{R}^n$, and $\mathbb{Z}$ is the integers. Here we are now using $\mathbf{y}$ to denote the set of variables in the optimization criterion, rather than the set of labels for our characters. The connection between these uses will become clear below. We will solve our optimization problem by posing it as an IP over binary valued variables.

Using the notation defined for Eq. (4.5), let $y_{i:j} = 1$ if variable $y_i = \mathcal{A}_j$ and 0 otherwise. Let $y_{i(i+1):jk} = 1$ if variables $y_i = \mathcal{A}_j$ and $y_{i+1} = \mathcal{A}_k$ and $y_{i(i+1):jk} = 0$ otherwise. Our optimization problem from Eq. (4.5) (before integrating the equivalence and non-equivalence constraints $\mathcal{C}$) can then be written

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{N}\sum_{j=1}^{|\mathcal{A}|} \phi_{i:j} y_{i:j} + \sum_{i=1}^{N-1}\sum_{j=1}^{|\mathcal{A}|}\sum_{k=1}^{|\mathcal{A}|} \phi_{i(i+1):jk} y_{i(i+1):jk} && (4.6)\\
\text{subject to} \quad & \sum_{j=1}^{|\mathcal{A}|} y_{i:j} = 1 && (4.7)\\
& \sum_{k=1}^{|\mathcal{A}|} y_{i(i+1):jk} = y_{i:j} && (4.8)\\
& \sum_{j=1}^{|\mathcal{A}|} y_{i(i+1):jk} = y_{(i+1):k} && (4.9)\\
& y_{i:j}, y_{i(i+1):jk} \geq 0 && (4.10)\\
& y_{i:j}, y_{i(i+1):jk} \in \mathbb{Z}. && (4.11)
\end{aligned}$$

Eq. (4.7) ensures that we choose exactly one label for each character. Eqs. (4.8) and (4.9) ensure that we choose exactly one pairwise factor for each pair of characters and enforce consistency between assignments. Lastly Ineq. (4.10) and Eq. (4.11) ensure that our variables are restricted to non-negative integers.

We will enforce the equivalence and non-equivalence constraints $\mathcal{C}$ as follows. Let $\mathcal{C} = \{\mathcal{C}_s, \mathcal{C}_d\}$, where $\mathcal{C}_s$ is the set of equivalence constraints and $\mathcal{C}_d$ is the set of non-equivalence constraints. In order to enforce these constraints, we add the following to our IP constraint set:

$$y_{i:j} - y_{i':j} = 0, \forall (i, i') \in \mathcal{C}_s \tag{4.12}$$

$$y_{i:j} + y_{i':j} \leq 1, \forall (i, i') \in \mathcal{C}_d \tag{4.13}$$

The equivalence constraints expressed in Eq. (4.12) enforce that whenever either $y_{i:j}$ or $y_{i':j}$ is set to 1, the other must be set to 1 as well. The non-equivalence constraints expressed in Ineq. (4.13) enforce that both $y_{i:j}$ and $y_{i':j}$ cannot be set to 1 at the same time. Note that non-equivalence constraints such as (4.13) can be incorporated into an IP in standard form by including both the constraint and its negation.

4.2.2 Optimization Considerations

We use the Mosek[3] optimization toolbox, which uses a variant of the branch-and-cut method, to efficiently solve our integer programs. Branch-and-cut works by first relaxing the integer program to a linear program. The optimization proceeds, eliminating non-integral solutions by adding constraints that remove these solutions from consideration. Once no more constraints can be added, the optimization uses the branch and bound strategy, which incrementally adds integer constraints on the variables. A branch in the optimization tree corresponds to choosing 0 or 1 for a specific variable. A lower bound on the optimization criterion is maintained by checking conditions on LP relaxations solved throughout the algorithm. An upper bound is maintained by noting the cases where the solution to an LP relaxation has binary values. These bounds allow the algorithm to prune subtrees of the optimization. For more information on linear optimization, see [3].

The Mosek solver uses finite error tolerances on both the integer feasibility and the optimization criterion in order to improve performance. Therefore, we do not have a formal guarantee of optimality. In our experiments, the solver's optimum was

[3] http://mosek.com/

always the true optimum. Here the *true optimum* refers to the best solution with respect to the optimization criterion, not necessarily to the *correct* solution.

4.2.3 Handling Inconsistent Constraints

If we assume ground-truth equivalence and non-equivalence constraints, our optimization space is guaranteed to be feasible. When we estimate equivalence and non-equivalence, we need to ensure that our constraints are consistent. For example, suppose we estimate that characters i and j are equivalent, characters j and k are equivalent, and characters i and k are different. These constraints are contradictory and hence there is no solution that satisfies them. We resolve this by removing constraints that violate consistency. An example of this scenario is shown in Figure 4.2.

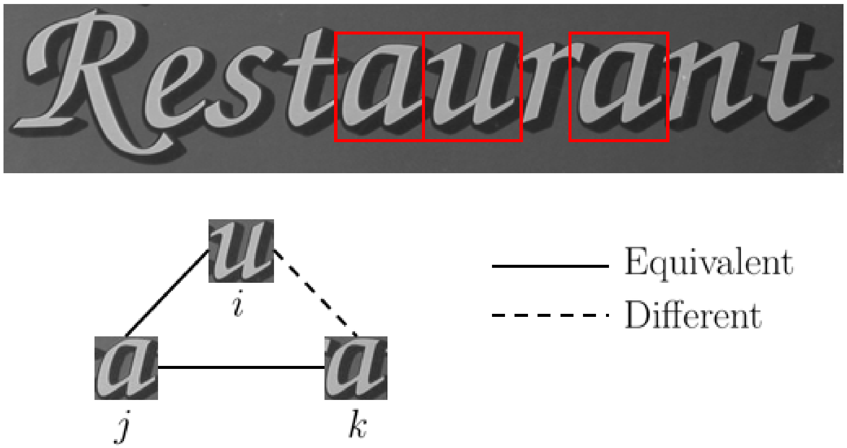


Figure 4.2: An example of inconsistent constraints. The similarity classifier labels i and j as equivalent, j and k as equivalent, but i and k as different. Through transitivity we know that i should be equivalent to k, which is inconsistent with the classifier output. In order to make our constraints consistent, we remove the constraints associated with this clique.

The root of this problem is that under certain conflicting constraints, we can determine that two characters should be equivalent through transitivity (i.e. $i = j = k$), and this equivalence conflicts with a non-equivalence constraint. To detect such a conflict, we compute a graph over characters, where the graph contains an edge between characters i and j if i and j are equivalent under transitivity. All connected components of this graph will be fully connected. If we find that we have included an edge between two nodes that we have estimated to be different, we have a conflict. We can remove this conflict by removing all constraints in the relevant clique.

4.2.4 Error Correction

The final stage of recognition involves using information from a search engine to incorporate language statistics that are more global than our bigram model. We use search engine results to model the distribution of common strings, as in described by Donoser et al. [14].

Given a labeling h_{init} from our IP optimization, we create a set of 1-character substitutions of h_{init} over all characters in our alphabet $\mathcal{A}$. We add to this set any suggestions made by the search engine when these candidate strings are submitted to the search engine. This results in a set of hypotheses $\mathcal{H}$ for the true string. For each $h \in \mathcal{H}$, we record the number of hits from the search engine. We induce a probability distribution $p_{\mathcal{H}}(h)$ over $\mathcal{H}$ by normalizing the search hit counts with add-1 smoothing.

Rather than relying solely on search hit counts to correct errors in h_{init}, we wanted to combine this information with appearance information $p_{\mathbf{x}}(h)$ to produce a final probability for each h. If $p_{\mathbf{x}}$ and $p_{\mathcal{H}}$ are both probabilities, we could assume independence and multiply them together. However, as occurs frequently when combining language models with appearance models, these two distributions were "imbalanced" in the sense that the appearance term dominated the product, rendering $p_{\mathcal{H}}$ useless. To address this, we introduced a correction factor α to balance the terms:

$$f(h) = p_{\mathbf{x}}(h)^{\alpha} \cdot p_{\mathcal{H}}(h)^{(1-\alpha)}, \tag{4.14}$$

for $0 \leq \alpha \leq 1$. We compute $p_{\mathbf{x}}$ by evaluating each hypothesis $h \in \mathcal{H}$ according to a linear function in the form of Eq. (4.6), with bigram factors removed, and then normalizing. We remove the bigram factors to rely solely on language information from the search engine. After setting α on held out data, the best hypothesis $h^* \in \mathcal{H}$ is simply the one maximizing Eq. (4.14).

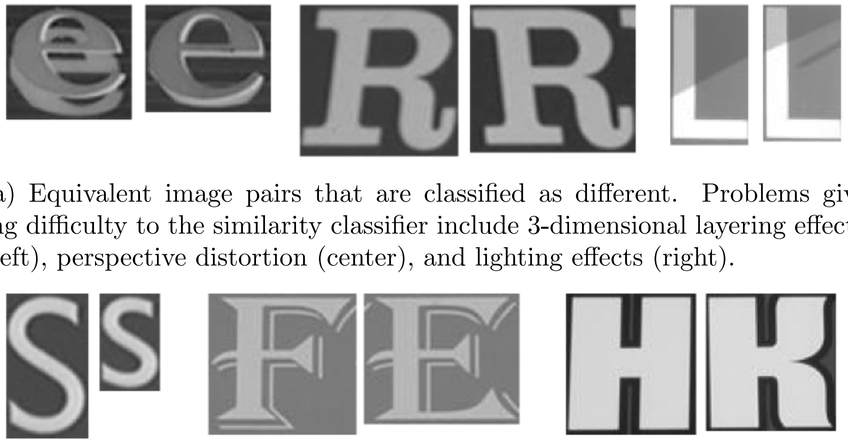
4.3 Experiments

For these experiments, we chose to use the VIDI data set, which is described in more detail in Chapter 3. We chose this data set to evaluate our algorithms because a consistent body of work has been evaluated on this data set. We also used the ICDAR 2003 data set as a source of exploratory data for our initial experiments on equivalence classification.

While scene text recognition requires finding text in an image, possibly segmenting it, and finally recognizing, we adopted a common simplification by starting with hand segmentations of each character in the form of a rectangular bounding box. *This is a substantial simplification of the full scene text recognition problem,* and the difficulty of solving the initial stages of detection and segmentation should not be underestimated. Nevertheless, we felt we could better assess our contributions by deferring the solution of these initial stages. We compare results to others that have made the same assumptions.

4.3.1 Similarity Classifier Experiments

We use five-fold cross validation to evaluate the accuracy of the similarity classifier on the test data. We extract all pairs of same and different images originating from the same sign from the data set and divide them into five groups, making sure that all pairs from the same sign are in the same group. We train an SVM with a quadratic kernel using four of the folds as the training set and the remaining fold as the test set.



(a) Equivalent image pairs that are classified as different. Problems giving difficulty to the similarity classifier include 3-dimensional layering effects (left), perspective distortion (center), and lighting effects (right).

(b) Different image pairs that are classified as equivalent. The leftmost pair represents a capital S and a lowercase s, which are considered to be different according to our evaluation criterion.

Figure 4.3: Examples of classification errors made by the similarity classifier using four SIFT descriptors.

The result is a classification for each of the 10,290 pairs of similar and dissimilar characters in the test set. Using the classifier with one SIFT descriptor we correctly classify 10,215 pairs for an accuracy of 99.27%. Using the classifier with four SIFT descriptors, we correctly classify 10,230 pairs for an accuracy of 99.42%. Examples of classification errors made by the classifier using four SIFT descriptors are shown in Figure 4.3.

While we achieve over 99% accuracy for our equivalence classification for both types of features, this result is not quite as good as it sounds. In particular, for each string of words with k characters in our sign database, there are $\mathcal{O}(k^2)$ similarity comparisons. Since we are making hard decisions about equivalence, an error in any one of these $\mathcal{O}(k^2)$ equivalence determinations would result in at least one incorrect word coming out of our integer program. While we mitigate this problem to some extent by eliminating equivalence constraints which are inconsistent with each other, our IP accuracy is still highly sensitive to errors in equivalence determination. Thus, the error rate needs to be extremely low for this source of information to be helpful.

As the next section shows, the similarity information, even with some errors, does indeed improve the accuracy of our integer program (see Table 4.1). With our second processing step using search engine correction, it proves even more helpful.

4.3.2 Character Recognition System Experiments

In this section, we first describe our cross-validation scheme which allows us to use the same data for training and testing by using different folds. We report results for a variety of experiments that compare accuracies of sign recognition with no similarity, with estimated similarity, and with ground truth similarity as given by an oracle. We report accuracies with and without post-processing using the search engine-based language model.

4.3.2.1 Cross Validation

While we developed the general form of our similarity expert using the ICDAR data, after we settled on the form of our model, we wanted to adapt the parameters (for the similarity SVM and the α parameter for balancing appearance and language information in the IP) to the properties of the VIDI data set. To do this, we split the VIDI data set into five folds of approximately the same size. No characters from a single sign appeared in more than one fold. The reason for avoiding having some characters from a sign go into one fold and some go into another is that this would make the similarity classification artificially easy, since the training data and test data might have pairs of characters that were virtually equivalent.

After splitting the VIDI data into five folds, we used four folds for training the similarity SVM and used this SVM to rate all of the pairs in the other fold as equivalent or different. This resulted in five independent sets of estimated equivalences and differences. Again using four folds at a time for training, we estimated the parameter α from Eq. (4.14). We then solved the IP for the test fold and applied the error

		No error correction	Error Correction
No Similarity	Word Accuracy	75.35	88.37
	Char. Accuracy	91.81	94.21
Similarity Classifier	Word Accuracy	78.60	92.56
(1 SIFT feature)	Char. Accuracy	93.05	96.20
Similarity Classifier	Word Accuracy	**78.60**	**92.56**
(4 SIFT features)	Char. Accuracy	**93.30**	**96.28**
Ground Truth	Word Accuracy	83.72	93.02
Similarity	Char. Accuracy	94.46	96.44

Table 4.1: A table of word and character accuracies for each experiment. Results are shown with and without error correction.

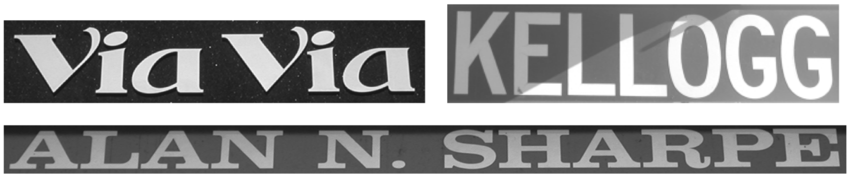
correction procedure of Section 4.2.4, often resulting in dramatic increases in word accuracy.

Table 4.1 shows a variety of results, compiled across folds, for word accuracy and character accuracy. Word accuracy is simply the percentage of words that are completely correct, including the proper case. A single character error, even if just a case disagreement, renders a word incorrect.

We show an improvement in word recognition accuracy due to similarity. With and without error correction we attain larger than 3% improvement in word accuracy over the equivalent method with similarity removed. Our best result of 92.56% achieves close to the same accuracy as our technique using ground-truth similarity. Furthermore, this result is higher than the state of the art result of 86.05% reported in [22]. See Figure 4.4 for examples of cases where the use of equivalence information improved performance.

4.4 Discussion

Perhaps the most immediate question about our results is what caused the improvement? It is tempting to conclude that our large gain in performance was due only to the search engine-based correction. However, a closer examination suggests



IP Solution without Equivalence Information	IP Solution with Equivalence Information
Via Wia	Via Via
KELLOGE	KELLOGG
ALAN N SHAREE	ALAN N SHARPE

Figure 4.4: Sample signs from the data set where equivalence or difference information improves recognition performance. The first two examples show how equivalence information can improve recognition while the third example shows how difference information can improve performance.

that we are squeezing more information out of similarity than was demonstrated in previous work.

In particular, in previous work [66], it is shown that similarity can be beneficial when there is a poor language model, but that when language information is added in the form of a lexicon, the similarity information, as implemented, is of little additional benefit. Specifically, without a dictionary, similarity information increases word accuracy from 75.35% to 78.60%, for a gain of about 3.25%. But when a lexicon is added, it seems to reduce the benefits of adding similarity. With a lexicon, similarity raises the accuracy only about 0.50%, from 85.58% to 86.05%.

However, in our work, even with the sophisticated language model implicitly defined by the search engine queries, we still see a 4% gain in word accuracy from adding similarity: from 88.37% to 92.56%. It is interesting to note that this occurs despite the significantly smaller gain in character accuracy of about 2%. We hypothesize one reason this may occur. If a word has exactly one error, and it violates an equivalence constraint, then this constraint effectively forces the algorithm to choose a single label for the equivalent characters. If the algorithm is correct in this guess 50% of the time, then the character accuracy would not change, but the word accuracy would be increased, since some of the single error words would be converted to zero errors, and others would be converted to two errors. Despite this analysis, it is likely that the system of Weinman et al. [66] would benefit significantly from post-processing using the search engine technique. Hence, it is difficult to conclude from our current experiments which combination of components would lead to the best overall system.

Given these observations, one direction for future work is to systematically vary factors and study trade-offs between belief propagation with soft equivalence constraints and integer programming with hard equivalence constraints. In addition, our similarity results can most likely be further improved by incorporating better alignment algorithms before using our similarity expert.

4.5 Conclusion

In this chapter we described a novel approach to incorporating similarity information that improves scene text recognition performance. We trained a similarity expert that learned to classify each pair of characters in a sign image as equivalent or not, and we formulated the search for the maximum likelihood interpretation of a sign as an integer program. We incorporated the equivalence information as constraints in the integer program and built an optimization criterion out of appearance features and character bigrams. Finally, we took the optimal solution from the integer program, and compared all nearby solutions using a probability model for strings derived from search engine queries. We demonstrated word error reductions of more than 30% relative to previous methods on the same data set with a word accuracy rate of 92.56%.

CHAPTER 5

A PROBABILISTIC SYLLABLE MODEL FOR CHARACTER RECOGNITION

While many appearance models have been shown to perform very well for recognizing individual characters, language information is also important for improving recognition results. Many existing techniques, including the one in the previous chapter, incorporate n-gram information as an additional source of information. One problem is that some n-grams are very uncommon, but will still appear in a word across a syllable boundary and these words are given a low probability under an n-gram model. To overcome this problem, we introduce a probabilistic syllable language model that uses a probabilistic context-free grammar to generate recognized word labels that are consistent with English syllables. We evaluate this language model for the problem of character recognition compared to a bigram model and a dictionary model.

5.1 Probabilistic Syllable Model

We introduce a new probabilistic syllable language model that incorporates additional information about syllables into the model. While many appearance models have been shown to perform very well for recognizing individual characters [13, 41, 12], language information is also important for improving recognition results, especially in difficult images such as those shown in Figure 5.1. Many existing techniques incorporate n-gram information into their models, which describes how likely groups of characters are to occur next to each other [41, 54, 66, 39]. This information is very informative, but it is a highly local source of information so it can lead to word

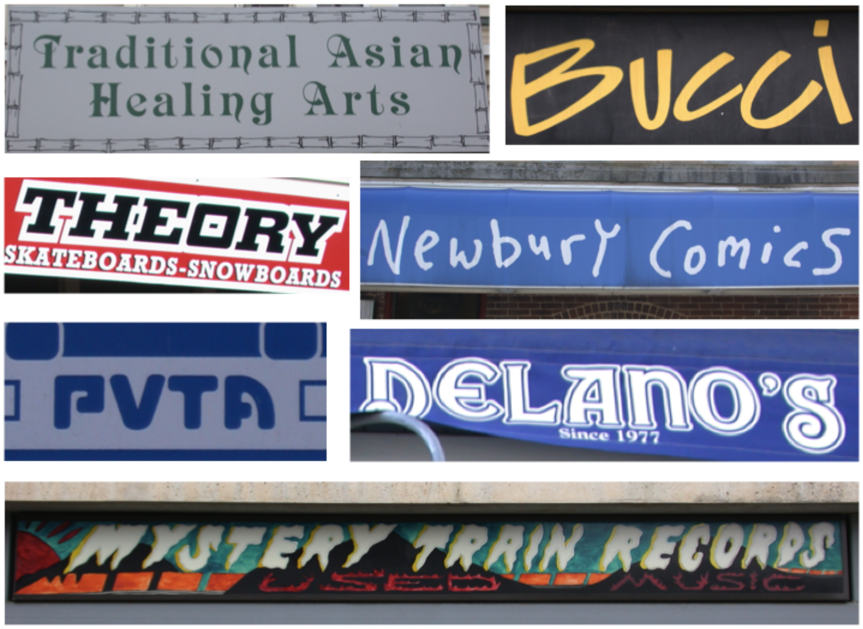


Figure 5.1: Sample scene text images with fonts that are difficult to recognize. Performance can be improved by combining appearance information with language information.

labeling errors. For example, bigram models allow a word to have a high probability as long as neighboring character labels have a high probability of occurring together. This means that a word may have a sequence of three unlikely consonants, but the probability will be high as long as each pair is likely to occur next to each other. Additionally, pairs of neighboring characters that occur across a syllable boundary may have a very low probability of occurring together, giving the entire word a low probability. As an example, consider the word ‘Amherst’. The combination of ‘m’ followed by ‘h’ is very rare in English, and as a result the word has a low probability under a bigram model.

We introduce a new probabilistic syllable language model that overcomes this problem by incorporating additional information about syllables into the model. We demonstrate the use of a probabilistic context-free grammar (PCFG), which encapsulates information about syllables, consonant groups and vowel groups in English and forces word labels to be consistent with a grammar. When humans encounter a new word, we often parse the word into syllables first and then look at the vowel and consonant sequences. This model produces word labels that can be parsed in the same way, because each will be made up of syllables. As a result, each recognized word generated under this language model is pronounceable. This type of syllable-based language model is particularly useful for the domain of scene text recognition where many of the words are proper nouns. These words are not likely to be in a standard dictionary, but we can take advantage of the fact that they should all be pronounceable.

This work is related to literature on probabilistic context-free grammars (PCFG). In this work, we use a PCFG as a language model for a text recognition task. This was done previously for mathematical equation recognition [36]. In addition, probabilistic context-free grammars have been used as language models for speech recognition tasks [21, 31]. They have also been used for syllabification tasks [23, 24].

5.1.1 Probabilistic Context-Free Grammar Definition

We model syllables in words with a probabilistic context-free grammar (PCFG). A context-free grammar G is formally defined as a four tuple $G =< V, \Sigma, R, S >$, where V is a set of non-terminal characters, Σ is a set of terminal characters, R is a set of production rules and S is the start symbol. A probabilistic context-free grammar associates a probability with each production rule. The probability of a particular parse under a grammar can be found by multiplying the probabilities of each rule in the parse.

Using a PCFG for our language model will incorporate a broader range of information. Instead of producing results which are consistent at the level of pairs of characters, results under this model will be consistent at the syllable level. This syllable model will also alleviate the problem of penalties on neighboring labels that cross a syllable boundary. Consider the example of the word 'Amherst' which was mentioned previously. A syllable model can produce the syllables 'am' and 'herst', which are both likely under a standard English syllable model, giving 'Amherst' a high probability. Next we define the probabilistic context-free grammar and explain our training method.

We define a PCFG G that has the following set of terminal characters,
Σ = { A,B,C,D,E,F,G,H,I,J,K,L,M,N,O,P,Q,R,S,T,U,V,W,X,Y,Z,
a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u,v,w,x,y,z }.
The set of non-terminals is,
V = {W,S,S_1,S_2,S_3,S_4,S_5,S_6,S_7,S_8,V,C,V_1,..,V_3, C_1,..,C_5}
with the start symbol of W.

The start symbol W represents a word. The non-terminal S represents a syllable and S_1-S_8 represent the eight types of syllables in this grammar. Each syllable type is made up of some combination of vowel and consonant sequences, represented by the non-terminals V and C. Vowel sequences can be one to three vowels long and consonant sequences can be one to five consonants long. Within each sequence, this grammar models the character at each position explicitly from training data, represented by the non-terminals $V_1 - V_3$ and $C_1 - C5$. The rules R are listed in Table 5.1.

Table 5.2 contains examples of words of varying lengths that are randomly generated from this grammar. These examples show that this grammar generates words that are pronounceable. Note that they are not necessarily words in English, since this grammar is only a basic approximation of English grammatical rules. The case of each character is not taken into account by this grammar, so we converted these examples to lowercase for readability, since letters can swap between uppercase and lowercase within a word.

$$
\begin{array}{lll}
\mathrm{W} & \rightarrow & \mathrm{S} \\
\mathrm{W} & \rightarrow & \mathrm{SW} \\
\mathrm{S} & \rightarrow & S_1|S_2|S_3|S_4|S_5|S_6S_7|S_8 \\
S_1 & \rightarrow & \mathrm{V} \\
S_2 & \rightarrow & \mathrm{CV} \\
S_3 & \rightarrow & \mathrm{VC} \\
S_4 & \rightarrow & \mathrm{CVC} \\
S_5 & \rightarrow & \mathrm{VCe} \\
S_6 & \rightarrow & \mathrm{CVCe} \\
S_7 & \rightarrow & \mathrm{CVCeC} \\
S_8 & \rightarrow & \mathrm{VCeC} \\
\mathrm{V} & \rightarrow & V_1|V_2|V_3 \\
\mathrm{C} & \rightarrow & C_1|C_2|C_3|C_4|C_5 \\
V_1 & \rightarrow & a|e|i|o|u|y \\
V_2 & \rightarrow & aa|ae|ai|ao|au|ay|...|yy \\
V_3 & \rightarrow & aaa|aae|aai|aao|aau|aay|...|yyy \\
C_1 & \rightarrow & b|...|z \\
C_2 & \rightarrow & bb|...|zz \\
C_3 & \rightarrow & bbb|...|zzz \\
C_4 & \rightarrow & bbbb|...|zzzz \\
C_5 & \rightarrow & bbbbb|...|zzzzz
\end{array}
$$

Table 5.1: The set of rules for our probabilistic context-free grammar.

Length	2	3	4	6	8
Words	co	nag	tear	tanluw	ancenner
	el	sel	pene	enples	opintest
	ta	bal	whin	esshep	ritfurci
	ni	ner	bini	tyfmyc	itentlec
	am	dow	thaw	enodan	iinefoth

Table 5.2: Words of varying lengths that are randomly generated by our grammar.

5.1.2 Model Training

We estimated the probabilities for this context free grammar on a combination of two types of documents. First, we used a syllabified version of Webster's dictionary to count and normalize the information needed. Since a dictionary does not contain a proportional amount of syllables (i.e. there are many words in a dictionary that start with zy, but these do not occur nearly as often in real documents), we augmented this training data with the same information from the top ten books from Project Gutenberg[1]. We tested the three methods of just dictionary information, just book information and both types of information together and found that all three performed similarly. For the experiments in this thesis we use the combination method.

5.2 Experiments

In this section we compare the performance of a probabilistic syllable model to three different models for text recognition. These include an appearance model, a model that combines appearance and bigram information, and a model that combines appearance and dictionary information.

5.2.1 Data Sets

We use two publicly available data sets in our experiments, VIDI and ICDAR 2011. They are both described in detail in Chapter 3. For these experiments, we use the ground truth character location information provided by the VIDI data set. Since ICDAR 2011 does not include character bounding boxes, we use the text segmentation method in Chapter 6 to identify character locations.

Since the probabilistic syllable model produces labels from the 52 character classes A...Za...z, we use subsets of both of these data sets created by removing words that

[1] http://www.gutenberg.org

include punctuation and numbers. The ICDAR 2011 subset includes 1008 words and the VIDI subset includes 209 words.

5.2.2 Appearance Model

Since the focus of this chapter is on demonstrating the benefit of using a probabilistic syllable model, we use a very simple appearance model in our experiments. We choose to use a logistic regression classifier because it is easy to train and produces a conditional probability for each character class, given an input feature descriptor. Note that this will not produce state-of-the-art character recognition results, but is sufficient for showing the benefits of using our new language model over a bigram model and a dictionary model.

We choose to use a histogram of oriented gradients (HOG) descriptor to model the appearance of characters. This descriptor has been shown to work well for scene text images [62, 61, 40, 39]. We resize each character to 60 by 60 pixels, and we extract one HOG descriptor, centered over the image.

We use these descriptors to train a 52 class (A-Za-z) logistic regression classifier. We use an implementation by Mark Schmidt [51]. This classifier is trained with synthetic font images provided by Weinman et al. [66]. These are binary character images for each character class in 1866 different fonts. We used 1866 positive example images and 200 negative example images for each class.

Once trained, this classifier takes a feature descriptor from an image and produces the conditional probability of each character class. To compute a word label for a new word image using only appearance information, we extract a HOG descriptor from each character image and find the maximum probability label for each using the classifier.

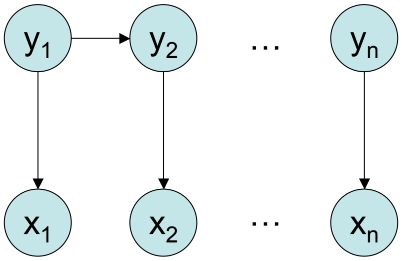


Figure 5.2: Hidden Markov model used to combine appearance information with bigram probabilities.

5.2.3 Bigram Language Model

We also show the results of using appearance information with bigram language information. These two sources of information can be combined using a standard hidden Markov model (HMM). This is represented by the graphical model in Figure 5.2. Each output label y_i takes into account the appearance of that character x_i and the previous label y_{i-1} Given this model, we know that,

$$p(\mathbf{x}, \mathbf{y}) = p(y_1) * \prod_{i=1}^{N-1} p(y_{i+1}|y_i) * \prod_{i=1}^{N} p(x_i|y_i)$$

Our goal is to find the word labels $\mathbf{y}$ that maximize that probability. We do this using the Viterbi algorithm, which uses dynamic programming to efficiently compute the most probable character labels, given appearance and bigram probabilities [59].

To compute a word label for a new word image we extract appearance information using the process described in the previous section and estimate bigram probabilities from a collection of books from Project Gutenberg.[2] We then use the Viterbi algorithm to compute the most probable word label given the appearance and bigram information.

[2] http://www.gutenberg.org/

5.2.4 Probabilistic Syllable Language Model

In comparison, we show the result of using appearance information with our probabilistic syllable model (PSM). One of the benefits of using a probabilistic context-free grammar is that a dynamic programming algorithm exists to efficiently search for the most probable parse of a sequence of characters under a grammar. This algorithm is called CYK [76]. So for a new word image, we extract HOG descriptors for each character, and calculate the conditional probability for each class using the logistic regression classifier described above. We alter CYK slightly to include these appearance probabilities. So for each character, we give CYK a different distribution over the terminal characters, based on the appearance model probabilities for that character. Then, we run the standard CYK algorithm to find the most probable output labels using our probabilistic syllable language model.

5.2.5 Dictionary Language Model

We also compare the performance of a probabilistic syllable model to the performance of a dictionary model. To label a new word image using a dictionary, we evaluate the probability of each word in the dictionary by multiplying the appearance probabilities of each character in the word. Then, we choose the dictionary word with the highest probability as the label. Since a dictionary does not include case information, we evaluate three versions of each dictionary word, one in all uppercase letters, one in all lowercase letters and one in title case with the first letter in uppercase and the rest in lowercase. In order to make a fair comparison to the probabilistic syllable model, we modify the labeling process to include case as well. We generate label versions using CYK, restricted to choose only uppercase letters, only lowercase letters, or an uppercase letter followed by all lowercase letters.

	VIDI	ICDAR11
Appearance	29.19	14.09
Appearance + Bigrams	31.10	15.38
Appearance + PSM	33.49	16.37

Table 5.3: Word accuracy results comparing a probabilistic syllable model to a bigram model on the VIDI and ICDAR11 data sets.

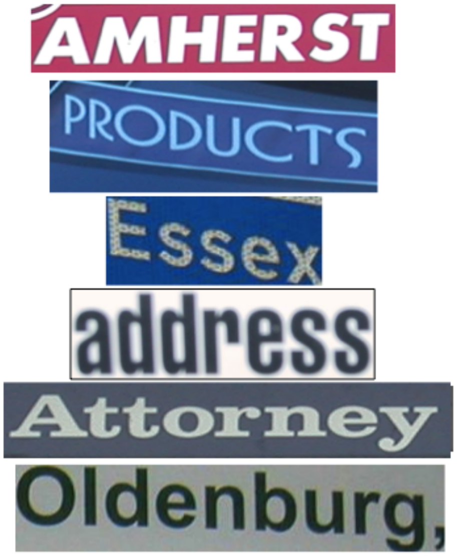
	VIDI	ICDAR11
Appearance + PSM$_{case}$	59.33	27.38
Appearance + Dictionary$_{case}$	57.42	30.46

Table 5.4: Word accuracy results comparing a probabilistic syllable model to a dictionary model on the VIDI and ICDAR11 data sets.

5.2.6 Results

We computed word labels for images in both data sets using appearance information, appearance and bigram language information, and appearance information combined with our probabilistic syllable model. The word accuracy results are shown in Table 5.3. This experiment shows that on the VIDI data set, the word accuracy increased by around 2% when bigram language information is added and by another 2% when the probabilistic syllable model is used, compared to the bigram model. For the ICDAR 2011 data set, the word accuracy increased by 1% each time. This demonstrates the benefits of using a more sophisticated model, that can capture correct language information across syllable boundaries.

Table 5.5 shows the output of the HMM model and the PCFG model for some sample scene text images. Each of these examples shows the benefit of using a probabilistic context-free grammar as a language model instead of a bigram model. As mentioned previously, one of the downfalls of a bigram model is that it gives high probabilities to entire words as long as each pair of neighboring characters is likely to occur together. In the first example, 'lm' and 'mb' are common bigrams, but



(a)

	Word	**HMM Output**	**PCFG Output**
1	AMHERST	LMBERst	AMHERst
2	PRODUCTS	pPOoUCTS	pRODUCTS
3	Essex	SssEx	EssEx
4	address	Rdiness	address
5	Attorney	Nttorney	Attorney
6	Oldenburg	Cldenburg	oldenburg

(b)

Table 5.5: Output of the HMM model vs. the PCFG model for sample scene text images.

put together in a sequence they become highly unlikely. The PCFG constructs results by syllables instead, so the output in each example, even if it is incorrect, is pronounceable.

We also computed word labels for images in both data sets using appearance information and a dictionary, compared to appearance information and a probabilistic syllable model. The word accuracy results are shown in Table 5.4. On the VIDI data set, the syllable model performs better than the dictionary model with a word accuracy of 59.33% compared to 57.42% using the dictionary model. On the ICDAR 2011 data set, the dictionary model performs better with a word accuracy of 30.46 % compared to 27.38% using the probabilistic syllable model. The strength of the dictionary model is that it maps each word image to the best dictionary word. The downfall is that it cannot produce labels that do not occur in the dictionary. In contrast, the probabilistic syllable model labelled 16.67% of the ICDAR 2011 non-dictionary words correctly, and 33% of the VIDI non-dictionary words correctly. This makes the probabilistic syllable model a better choice for data sets that include a large fraction of non-dictionary words.

5.3 Discussion

This model suggests several directions for future work. The first is to explore changes to the grammar definition. In this chapter, we defined a grammar that models each syllable as a sequence of consonant and vowel groups, and models the probabilities of each combination of consonants or vowels within those groups. This grammar does not use any information about how often syllable types occur next to each other, which can be a problem, for example, when words are generated with two vowel groups next to each other. We could alter the grammar to include information about what types of syllables occur near each other. As another extension, we could also learn how consonant and vowel groups relate to one another, i.e a particular vowel group follows a particular consonant group with high probability.

The experimental results in Table 5.3 and Table 5.4 also show the motivation for incorporating case information into the model. We see a large increase in accuracy on both data sets when case information is added to the probabilistic syllable model. Without this information, the case can swap between lowercase and uppercase in the middle of a word. One special case we discovered is when words have an uppercase letter in the middle of the word. This can occur in business names, i.e. PeoplesBank. The uppercase letter is likely to occur at the beginning of a syllable, so a syllable-based language model like this is a natural choice to handle this special case.

It should also be noted that the particular grammar presented in this chapter could also be represented as a regular grammar using a finite state machine, or a composition of finite state machines. We chose to represent it as a context-free grammar here for several reasons. First, as described in Section 5.2.4, the standard CYK algorithm generates the most probable parse under a grammar in O(n^3) time, which we found to be practical in our experiments. In contrast, representing the grammar as a set of finite state machines was cumbersome, and the machines were very large due to having 52 character classes. In practice we found the amount of memory required for our implementation to be prohibitively large. In addition, there is no standard algorithm, like CYK, for finding the most probable string under a finite state machine. In addition to these difficulties, as discussed above, many other grammar definitions may be explored in future work. These extensions may not be regular grammars, so having a more general framework for generating the most probable parse is beneficial.

5.4 Conclusion

In this chapter, we presented a new language model for scene text recognition. It incorporates more sophisticated language information by modeling syllables with a probabilistic context-free grammar. This approach is a better model of language information across syllable boundaries, so words with unlikely bigrams that cross syllable boundaries are not penalized. In addition, words are made up of syllable components, so word labels produced are pronounceable. In our experiments, we showed an increase in recognition performance when using this language model, compared to a bigram model and show the benefits of using it compared to a dictionary model.

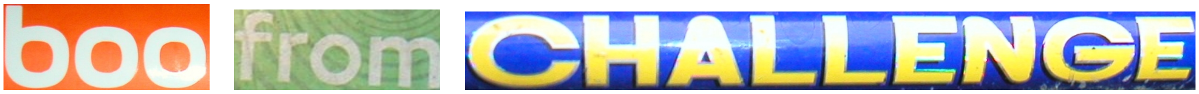
CHAPTER 6

BILATERAL REGRESSION SEGMENTATION

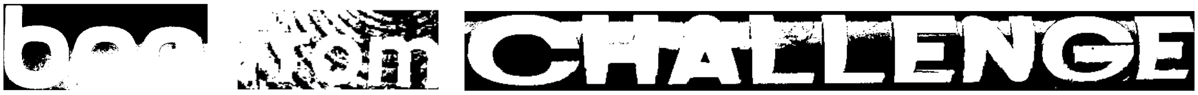
We present a new model for segmenting text in natural scene images called bilateral regression segmentation. This technique allows us to remove the assumption made in previous chapters that character bounding boxes are given, but still assumes that word bounding boxes are given. The model is motivated by the observation that many scene text images contain smooth color changes due to lighting conditions and these images are not handled well by existing segmentation techniques based on color clustering. Instead, we use a regression to model these changes closely and show that we can segment images that are often missed by other techniques. We also describe a cropped word recognition system that combines bilateral regression segmentation with simple, yet effective, components for recognition. We evaluate this system on the problem of word spotting, where recognized words come from a small, pre-specified lexicon of valid labels.

6.1 Text Segmentation

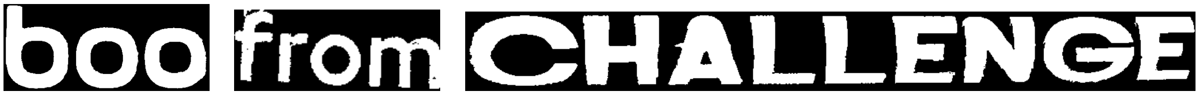
During the segmentation stage, our goal is to separate pixels in an image into two groups. The foreground should contain pixels that represent text, and all other pixels should be assigned to the background. Some existing object segmentation techniques divide images into coherent regions. However, in a scene text image, disjoint letters may be segmented as different regions, with no way to associate them as all belonging to the foreground. Other segmentation techniques use color information to group regions that are similar [55, 60, 26, 64, 38]. These work well when images have two



(a) Original Images



(b) Segmentation by Otsu's method



(c) Bilateral Regression Segmentation

Figure 6.1: Sample images where the color changes across the image. We model these changes using a regression-based segmentation method. This figure is best viewed in color.

distinct colors, but as colors change across images and image backgrounds become more complex, it becomes harder to find the correct distinction between background and foreground pixels.

We observe that in scene text images, the foreground pixels are very often a single constant or smoothly varying color and the background may be very complex. Figure 6.1 shows examples of images with colors that change across the image. Figure 6.2 shows examples where there are more than two prominent colors in an image and backgrounds are complex. To address these characteristics of scene text images and problems with existing techniques, we present a regression-based segmentation technique.

Regression allows us to model the smoothly varying color changes that often occur due to lighting. One possible approach for modeling an image with regression is to use a mixture of regressions. This is also known as a mixture of experts [49]. To optimize such a model, an expectation-maximization procedure can be used to alternate between assigning pixels to different regressions and re-estimating the regressions based on the assignments. These can be hard or soft assignments. This

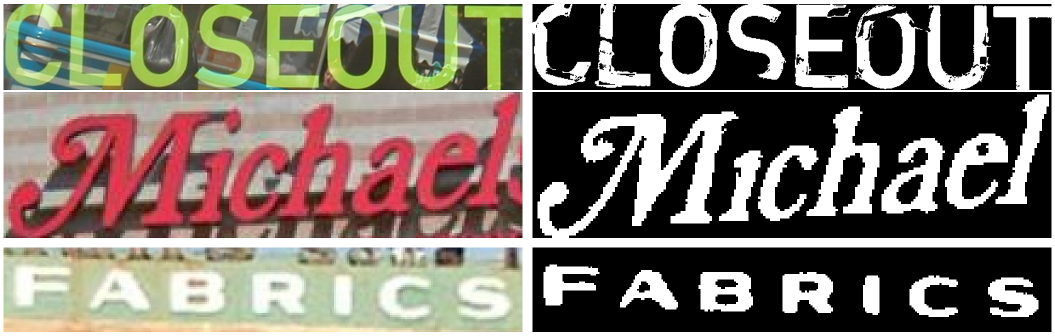


Figure 6.2: Sample images with complex backgrounds and their segmentations using bilateral regression.

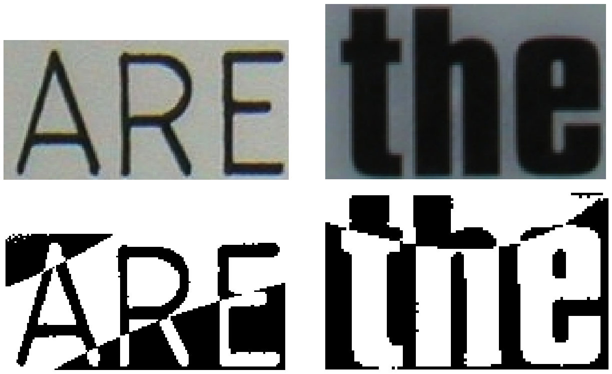


Figure 6.3: Sample segmentations that result from poor initialization using a mixture of two regressions.

type of method poses several difficulties. First, it can be difficult to initialize these models. Figure 6.3 shows examples of the type of segmentations that can result from poor initialization. Also, the complex backgrounds often found in scene text images are not well modeled by simple mixture models.

Instead of modeling every pixel in an image with a regression as the mixture of regressions framework does, we propose a technique that only models a subset of the pixels. We present a method to extract and model just the subset of pixels that belong to a coherent region that we are interested in modeling (like the foreground). This gives us a simple way to model pixel colors without the results being affected by nearby, unrelated pixels. We use this technique to model foreground hypotheses and present a selection procedure that chooses the best foreground segmentation from this set. Since this allows us to ignore background pixels, this technique is robust for images with complex backgrounds.

6.1.1 Bilateral Regression Segmentation

We now introduce our regression based segmentation technique that models only the foreground of each image. We call this method bilateral regression, because it borrows ideas from bilateral filtering [1, 57].

Polynomial regression models can be used to model the relationship between two variables x and y as a polynomial curve. The order of the polynomial changes based on the relationship between x and y. A regression model of order one is the line that best models y as a function of x,

$$y = ax + b$$

Similarly, a regression model of order two is the quadratic curve that best models y as a function of x,

$$y = ax^2 + bx + c$$

This can be easily extended to two dimensions, where the regression models the relationship between three variables, x, y and z,

$$z = ax^2 + by^2 + cxy + dx + ey + f$$

This model represents the quadratic surface that best models z as a function of x and y. In this form, a regression model can be used to model smooth brightness changes in an image. We use color images in this work, which can be modeled using a separate quadratic surface for each color plane.

Each of the above equations can be re-written in matrix form. In the one dimensional linear case,

$$\begin{aligned} y &= ax + b \\ &= \begin{bmatrix} x_1 & 1 \\ x_2 & 1 \\ . & . \\ . & . \\ . & . \\ x_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \\ &= XC \end{aligned}$$

And similarly, for a two dimensional quadratic regression,

$$\begin{aligned} z &= ax^2 + by^2 + cxy + dx + ey + f \\ &= \begin{bmatrix} x_1^2 & y_1^2 & x_1y_1 & x_1 & y_1 & 1 \\ x_2^2 & y_2^2 & x_2y_2 & x_2 & y_2 & 1 \\ . & . & & & & \\ . & . & & & & \\ . & . & & & & \\ x_n^2 & y_n^2 & x_ny_n & x_n & y_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \\ e \\ f \end{bmatrix} \\ &= XC \end{aligned}$$

The vector of coefficients defining the model can be found using a least squares approach. This leads to a solution that minimizes the sum of squared differences between the data and their values predicted by the model. Given the matrix notation described above, we can find the following solution for the regression coefficients C,

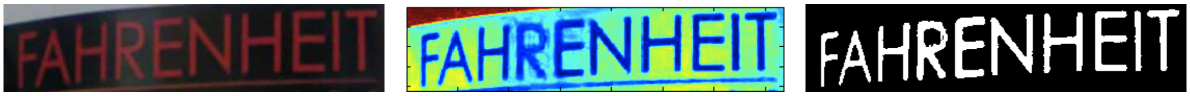
$$
\begin{aligned}
y &= XC \\
X^\top y &= X^\top XC \\
(X^\top X)^{-1}X^\top y &= (X^\top X)^{-1}X^\top XC \\
C &= (X^\top X)^{-1}X^\top y
\end{aligned}
$$

It is also possible to find a weighted least squares solution, where there is a weight associated with each data point.

Our goal is to model only the foreground of an image, so our approach is to use a weighted regression, where each pixel is weighted according to how close it is to the foreground in feature space. This allows the regression to select out pixels we are interested in modeling (those that are part of the foreground text) and to ignore pixels that are a poor fit (those that are part of the background scene). Since we do not know the color of the foreground text a priori, we model the top n most prominent colors in each image separately and then automatically select the best segmentation.

For each foreground color, we calculate pixel weights as in bilateral filtering to select the subset of similar pixels automatically. Each pixel is weighted according to its spatial distance from a representative seed pixel, combined with its distance in color space. To calculate these distances, we use two Gaussian distributions generated from the seed pixel p from image I. We define $p = I(x, y)$ to have color $c_p = (r_p,g_p,b_p)$. The first distribution G_s is a two dimensional Gaussian distribution based on the spatial location of pixel p. It has $\mu = (x, y)$ and $\sigma = \sigma_s$ in both dimensions. The second distribution G_c is a three dimensional Gaussian distribution based on the color of pixel p with $\mu = (r_p, g_p, b_p)$ and $\sigma = \sigma_c$ in all dimensions. The weight of each pixel q with color c_q is then

$$w_q = G_s(||p - q||) * G_c(||c_p - c_q||).$$



(a) Original Image (b) Error Image (c) Segmentation

Figure 6.4: A sample image, the corresponding regression error image (blue represents low error and red represents high error) and the resulting segmentation image. This figure is best viewed in color.

These weights allow the regression model to ignore pixels that are a poor fit, so the regression represents a close fit to the foreground pixels. Additionally, the model can ignore an arbitrary amount of data that is too far away in feature space. This can be thought of as a type of image-adaptive robust regression, just the way the bilateral filter can be thought of as a image-adaptive, robust way of estimating the local mean of an image. This idea is similar in spirit to several extensions to the bilateral filter that include linear components [6, 11, 16]. However, our goal is not to smooth images, but to use the weights to select a subset of pixels to build a local model that fits the data well.

We can create a segmentation from this model by calculating the error between each pixel and the model. We threshold the error image using Otsu's method to obtain a segmentation. Figure 6.4 shows a sample image, the regression error image and the resulting segmentation.

Once we have segmentations for the n most prominent color regions, we want to automatically select the segmentation representing the true foreground. To do this, we choose the segmentation with the components that can best be recognized as characters. We represent each cropped connected-component image with a histogram of oriented gradients (HOG) descriptor and calculate the l_1 distance to each image in a reference set of synthetic character images from 200 different fonts for 62 character classes. These include twenty-six uppercase letters, twenty-six lowercase letters and

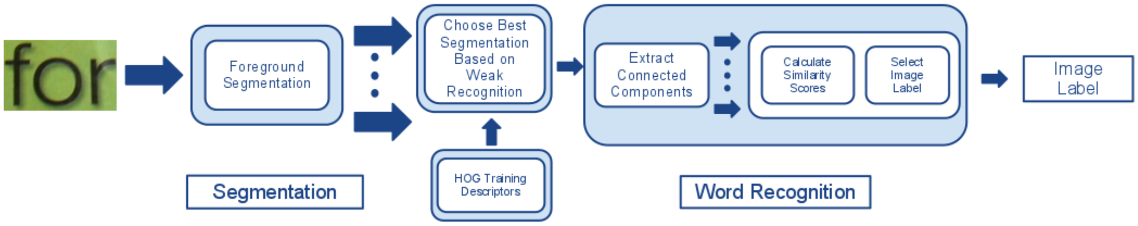


Figure 6.5: System Overview.

ten digits. The images are provided by Weinman [66]. The score of a segmentation image is the average of minimum distance for each component.

Before scoring connected components, we filter out noisy components that are not likely to be text. We remove components that have a height of less than one third of the image height and those that are more than 2.5 times as wide as they are tall. We also remove components that span the entire width or height of the image, since we know that the input images have a at least a small border around each word. In addition, we filter out images that contain a large amount of overlapping connected components, since the characters in a good segmentation should not be overlapping.

We want to choose a segmentation with foreground components that cover the image area as much as possible, so we choose the segmentation with the best score from the those that are within 10 percent as covered as the most covered of the choices.

6.2 A Complete Word Spotting System

We combine bilateral regression segmentation with simple components for recognition to create a complete word recognition system. An overview of this system is shown in Figure 6.5. Specifically, we address the problem of word spotting, where word labels are chosen from a small, pre-specified lexicon. This problem was introduced by Wang et al. [62, 61].

For each connected component in a segmentation image, we compute similarity scores to each of the 62 possible character classes. As above, we represent each cropped connected-component image with a HOG descriptor and calculate the l_1 distance to each image in the reference set described in the previous section. We use a nearest-neighbor approach where the similarity score for each character class is the distance to the nearest neighbor in that class. So for each connected component, we compute a vector of 62 similarity scores.

Given these similarity scores, we want to choose the most likely lexicon word label for an image. For each character, we form an equivalence class containing the three character classes with the highest similarity. Then we calculate the string edit distance to each lexicon word, where the substitution of a character for a member of its equivalence class has zero cost. The string edit distance returns the minimum number of insertions, deletions and non-equivalence class substitutions required to transform one string into the other. We label the image with the lexicon word that has the smallest edit distance. If there is more than one lexicon word with the smallest edit distance, we repeat the process, only we form larger equivalence classes from the top ten choices for each character. We do this because we want to favor words that include characters that were found to be similar in appearance. We calculate the edit distance to the remaining tied words and choose the lexicon word with the smallest value. If tied words remain, we choose a random word from this final set of ties.

6.3 Experiments

In the following sections, we evaluate both our proposed segmentation method and our complete word recognition system.

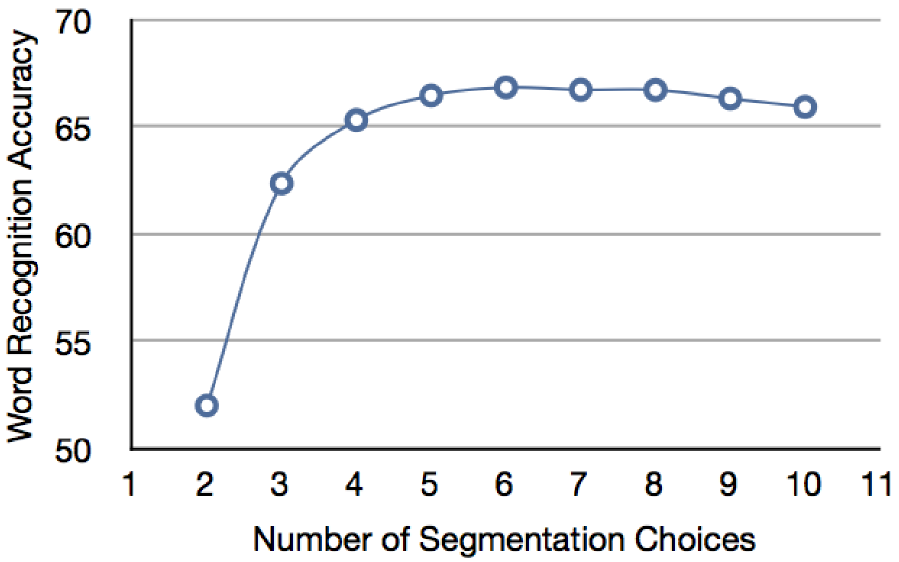


Figure 6.6: Word recognition accuracy results for different numbers of segmentation choices

6.3.1 Parameter Selection

For our experiments, we chose the value of n segmentation choices by looking at the performance of a range of values on a training set. We use the ICDAR 2003 training set, which is described in Chapter 3. For each number of segmentation choices from two to ten, we calculated the word recognition accuracy of our system. Figure 6.6 contains results of this experiment. This shows that the accuracy of our system is not very sensitive to the choice of this parameter. We use a value of $n = 6$ for all experiments described in this paper since it performs the best.

We set values for σ_s and σ_c for the regression weights experimentally. For all experiments on all data sets, σ_s is one third of the image width and $\sigma_c = 10$.

6.3.2 Bilateral Regression Segmentation Evaluation

To evaluate the new segmentation technique we propose in this paper, we need to compare the segmentations we produce to those produced by existing segmentation methods for scene text. One way to do that is to compare the segmentations to foreground/background ground truth information for a data set. As far as we know, complete ground truth information does not exist for any scene text data set, including the widely used ICDAR scene text data sets. In addition, this analysis does not capture exactly what we are interested in evaluating. Since the segmentation of scene text is done as an initial step for a recognition process, we want to compare whether our segmentations allow us to recognize words better than another segmentation method. We do this by varying the segmentation method used by our complete recognition system.

6.3.2.1 Data set

We use the ICDAR 2003 data set for this evaluation. We use the scene test set instead of the word test set because we were provided segmentations from the state-of-the-art segmentation method published by Mishra et al. [38] for direct comparison. We use the ICDAR03(50) and ICDAR03(FULL) lexicons that are described in Chapter 3.

6.3.2.2 Results

Table 6.1 shows the word recognition accuracy for both lexicon versions for the ICDAR 2003 data set. These results are evaluated in a case-insensitive way. This means that the label ‘The’ for an image with ground truth ‘THE’ is considered correct. Since our algorithm may contain a random choice during the labeling process, the accuracies we report are the average over 50 trials. We compare our technique to two existing segmentation methods. The first is Otsu’s method [47] and the other is by Mishra et al. [38].

These results show that our segmentation method provides more accurate recognition than existing methods. Our method is also more than an order of magnitude

Segmentation Method	ICDAR03(FULL)	ICDAR03(50)
Otsu	58.81	66.40
Mishra et al.	66.33	74.76
Bilateral Reg.	67.76	76.53

Table 6.1: Word accuracy for word spotting on the ICDAR 2003 scene data set of 1107 words.

faster than the method by Mishra et al. Their method takes an average of 32 seconds per image while our method takes an average of 3 seconds per image.

6.3.2.3 Segmentation Selection Evaluation

Since our method produces n segmentations and chooses the best automatically, we want to analyze this selection process. We performed the following experiments using the ICDAR 2003 data set. We compare our selection process to two baseline selection techniques and an oracle. The first baseline process is to always choose the segmentation created by the most prominent color in the image (assuming it is the foreground). The second baseline process is to always choose the segmentation created by the second most prominent color in the image (assuming that the most prominent is the background). The oracle chooses the segmentation that results in the best labeling of an image. That is, if a segmentation results in the correct labeling it is chosen. The word accuracies for the first and second baseline processes are 13.24% and 44.23% respectively and the word accuracy of the oracle is 71.80%. The word accuracy for our selection process is 66.94%, which is just a few percent less than the oracle. This shows that the high level recognition information we use in our selection process plays an important role in improving segmentation selection.

6.3.3 Complete Word Recognition System Evaluation

We evaluate our complete word recognition system by comparing it to the existing state-of-the-art system for the problem of word spotting.

	ICDAR03(FULL)	ICDAR03(50)	SVT
Wang et al.	62.00	76.00	57.00
Otsu + Word Rec.	67.21	72.13	43.16
Bilateral Regression + Word Rec.	73.43	79.47	54.20

Table 6.2: Word accuracy for word spotting on the ICDAR 2003 and SVT data sets. The ICDAR 2003 data set used is a subset of the original, to allow for a fair comparison to existing work.

	ICDAR03(FULL)	ICDAR03(50)	ICDAR11(FULL)	ICDAR11(50)
Bilateral Regression	66.78	76.03	62.28	72.69

Table 6.3: Word accuracy for word spotting on the complete ICDAR 2003 and ICDAR 2011 data sets.

6.3.3.1 Data sets

We evaluate our method on three data sets, ICDAR 2003, ICDAR 2011 and SVT. We use the ICDAR(50) and ICDAR(FULL) lexicons. These are all described in Chapter 3.

For ICDAR 2003, we follow the experiments of Wang et al. [61] and present results on a subset. We remove all words that contain non-alphanumeric characters and those with a length of two or less, for a total of 862 words. We also present results on the complete test set to allow for future comparisons with our method. We do not know of any existing word spotting results for the ICDAR 2011 data set, but provide ours for future comparison and completeness.

6.3.3.2 Results

Table 6.2 shows the word recognition accuracy for both lexicon versions for the ICDAR 2003 data set and the SVT data set. As in the previous evaluation, these results are also evaluated in a case-insensitive way. Additionally, the accuracies we report are the average over 50 trials. We compare our method to the current state

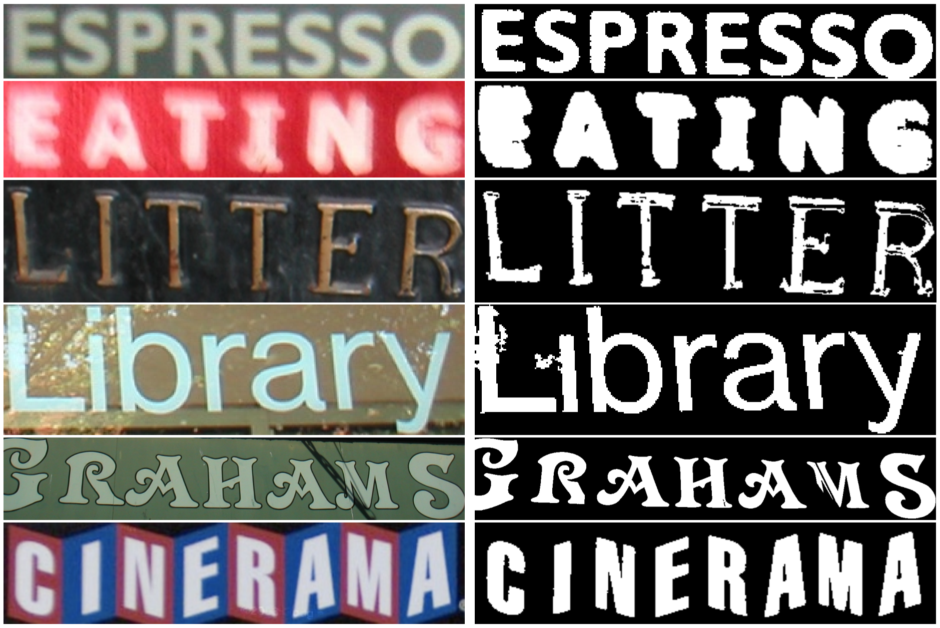


Figure 6.7: Examples of words that we identify correctly and their foreground segmentations.

of the art system by Wang et al. [61]. We also compare to a version of our word recognition system that uses Otsu's method for segmentation.

Using our method, there is a large increase in word accuracy using the ICDAR03(FULL) lexicon from 62% to 73.43%. This is a 30% reduction in error over the current state of the art. There is also a smaller increase from 76% to 79.47% using the ICDAR03(50) lexicon and a decrease from 57% to 54.2% for the SVT data set. Figure 6.7 shows examples of sign images that we label correctly and their segmentations. Figure 6.8 shows examples of sign images that we label incorrectly. The difficulties include low resolution, low contrast, abrupt lighting changes and connected text.

Table 6.3 shows the word recognition accuracy for word spotting on the complete ICDAR 2003 and ICDAR 2011 data sets. These are provided for future comparison and completeness.

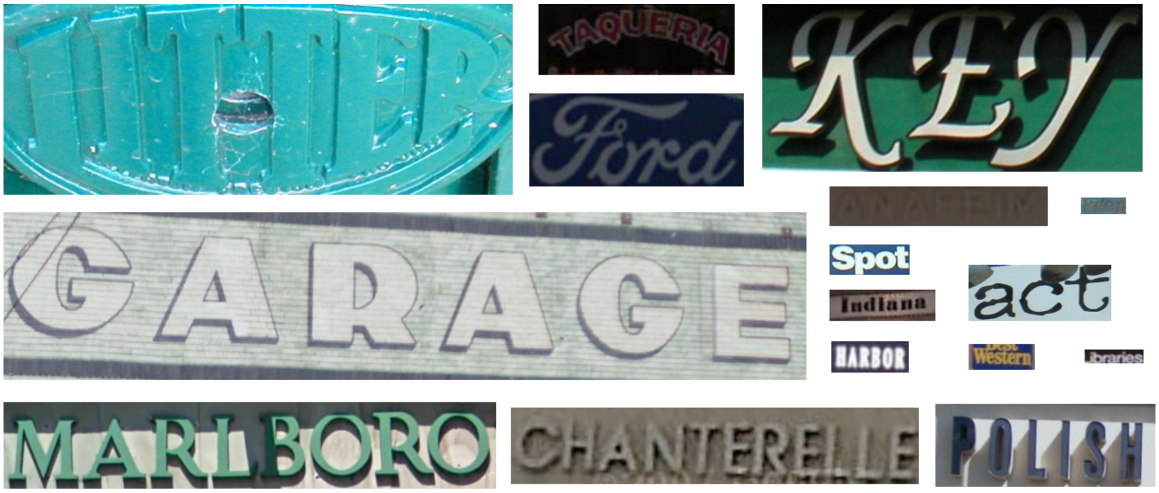


Figure 6.8: Sample images that we identify incorrectly. Characteristics that make these images difficult are low resolution, abrupt lighting changes, connected text, and low contrast.

6.4 Discussion

In our experiments, we see a large increase in word accuracy on the ICDAR 2003 data set, but a modest decrease on the SVT data set. We believe this is because the images in the SVT data set are much more difficult to segment. Overall, they have a lower resolution than the images in the ICDAR03 data set and they exhibit more artifacts due to blur. This may be because they were collected from Google Street View and the images are taken from a moving vehicle. In this setting, approaches that do not rely on segmentation seem to perform better. However, when images have sufficient resolution and less blur, such as in the ICDAR 2003 data set, our approach based on segmentation performs better.

We chose a segmentation-based approach because we observed that by segmenting images into foreground and background components, we can eliminate many areas of the image that might exhibit features of text. We believe that this contributes to the method's success in many instances. The disadvantage to this approach is that if text is connected, or if an image is blurry or low resolution and the boundary between characters becomes less clear, it is difficult to find the correct segmentation. This is because our technique relies on recognizing distinct connected components to select the best segmentation.

There are many directions for future work that may improve this method. As discussed above, the biggest weakness of this segmentation method is that it relies on characters being distinct connected components in order to choose the best of n segmentations, so it does not work well on connected or blurred text. It would likely improve performance to adapt the segmentation selection procedure to not rely on text being separated. It should also be advantageous to use a better recognition model in the segmentation selection procedure. Right now we use a nearest neighbor approach, but this could be replaced with a logistic-regression classifier, like the one described in the previous chapter, or a CRF-based recognition model like the one described in the next chapter. In this work, we use the average character distance over a word as the score of a segmentation, but changes to this score function could also be explored.

Another weakness of this work is that it is evaluated using the task of word spotting. While this captures the fact that we want to evaluate our segmentation technique by how well it allows us to recognize words, it does not provide a full picture of how changes to the segmentation technique change performance. This is because word spotting requires that we choose a label from a pre-specified lexicon. Consider the case where there maximum probability character labels predict ‘SGHOOL’, and this maps to the word spotting label ‘SCHOOL’. A change to the segmentation technique may produce the character labels ‘SCHOOL’, and we should prefer this version of the method. Unfortunately, this also maps to ‘SCHOOL’, so we won’t see any change in the word spotting performance. A better evaluation would be on the task of open-vocabulary word recognition, where words are not chosen from a pre-specified lexicon.

6.5 Conclusion

In this chapter, we presented a new model for segmenting text in natural scene images of cropped words called bilateral regression segmentation. We used a regression-based technique to model smooth color changes in just the subset of pixels that belong to the foreground text, while ignoring the background pixels altogether. We showed that it is suitable for segmenting images with color changes like those caused by lighting and complex backgrounds. We evaluated this method compared to the current state-of-the-art on the task of word-spotting and showed that our method leads to better recognition accuracy.

CHAPTER 7

A WEB-BASED LEXICON FOR OPEN-VOCABULARY WORD RECOGNITION

In the previous chapter, we relied on a pre-specified, small lexicon for possible word labels. This utility of this system is limited, since text in the environment is likely to contain proper nouns and other words that will not appear in a general lexicon and specialized lexicons have to be built by hand. Instead, in this chapter we present a system for open-vocabulary cropped word recognition. We present a new approach for incorporating web-based language information that improves recognition performance. We also describe a complete system for open-vocabulary word recognition that combines the segmentation method from the previous chapter with a standard method for recognition and an error correction step that relies on the web-based language information. We evaluate this system compared to existing state of the art approaches.

7.1 A Web-Based Lexicon

Lexicon information has been shown to improve performance in scene text recognition. However, general lexicons are not likely to contain the proper nouns and other words that appear in scene text images. To address this problem, Donoser et al. introduced the idea of using document frequency counts for a query from a search engine as a source of global language information in word recognition [15]. This information source is always changing and reflects new words, like business and street names. Unfortunately, this process can be slow for researchers, since most search engines limit query submissions to one query per second and only allow a small number of queries per month through the API without paying fees.

To overcome these limitations, we introduce a new approach to incorporating web-based language information. We use a static word n-gram data set called Web 1T 5-gram released by Google [5]. It includes words that occur in a crawl of the web and their term frequencies. Since this information does not require querying a search engine, we can incorporate language information efficiently.

We construct a lexicon containing the word unigrams in the Web 1T 5-gram data set and the frequency count associated with each. The frequency count is the number of times a word unigram occurs on web pages. This lexicon contains around 13.5 million words. Since it is created from word unigrams that are found on the web, many entries are misspelled words or contain symbols within a word. The word unigrams are not processed at all to remove these errors, the data set contains all word unigrams found on the web crawl. Our method is robust to these included entries because we use the frequency count information to determine how common each entry is. We describe how we use this lexicon for error correction in the next section.

7.2 An Open-Vocabulary Word Recognition System

The word recognition system we describe in this section is shown in Figure 8.1. First, we segment each cropped word image into foreground and background components using the text segmentation method presented in the previous chapter. Given the characters from this segmentation, our goal is to find the best word label given appearance and character bigram probabilities for the characters and global language information from a web-based lexicon. We could evaluate the probability of every lexicon word based on this information and choose the word with the maximum probability, but since it contains over 13.5 million words this approach is too ex-

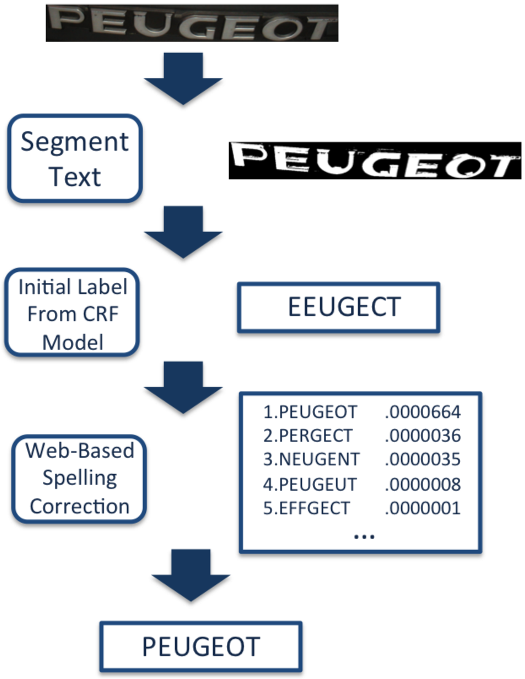


Figure 7.1: This describes a step-by-step example of our system. First, an image is segmented into foreground text and background. Next, a conditional random field (CRF) model is used to find the most likely text string, given the connected components in the segmentation. Finally, web-based error correction is performed, where global language and appearance information are combined. The most likely hypothesis is chosen as the final text label.

pensive. Instead, we describe a fast approximation to this approach. We use the Viterbi algorithm to find an initial word label based on just appearance and bigram probabilities, and then we correct any errors in the initial label by evaluating the probability of lexicon words that are within 2-characters of this label given global language information.

Below, we explain the process for finding an initial word label and describe the fast web-based error correction step in more detail.

7.2.1 Initial Word Recognition

Given the binary foreground/background image output by bilateral regression segmentation, we use a CRF model to produce an initial text label for each image.

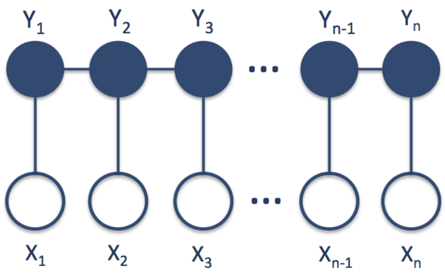


Figure 7.2: We use a linear-chain conditional random field (CRF) model.

We consider each connected component in the binary image as a character and we use a linear-chain CRF to represent the sequence of those characters in a word. A graphical representation of the model is shown in Figure 7.2. The open circle nodes are observed and the shaded nodes are variables that are predicted by the model. The variables $Y_1, Y_2, ..., Y_n$ are character labels, and can take one of the 62 different labels from the set A-Z, a-z or 0-9. The variables $X_1, X_2, ..., X_n$ represent appearance features of each individual character.

We create appearance features by extracting one HOG descriptor from each character, centered and covering the entire character image. These are the same appearance features used in the segmentation step above. We also add a weak case feature to represent the height of each character. This feature value is the height of a character divided by the height of the tallest character in the same word. We concatenate the HOG descriptor with the case feature value into one feature vector.

We estimate the CRF model parameters with maximum likelihood training by minimizing the negative log-likelihood of the objective function. We use both the ICDAR 2003 training set and the ICDAR 2011 training set as training data. We found that this was not enough data to learn a good model, so we also generated synthetic training data. This was straightforward because we are using binary foreground/background images. We generated our own using the set of synthetic fonts introduced by Weinman et al. [66], described in Chapter 3. We selected a random word from a dictionary and a random font, and generated each word as white text on a black background. We included words in lowercase, uppercase and title case.

Next, we use the Viterbi decoding algorithm to find an initial word label, given the CRF model [59]. This is a fast, dynamic-programming solution for finding the joint configuration of labels $Y_1, Y_2, ..., Y_n$ that has the highest probability. We also compute three other word labels to encourage case consistency. We know that text is usually written either in all uppercase letters, all lowercase letters, or an uppercase letter followed by all lowercase letters (title case). We compute a word label for each version by restricting the Viterbi algorithm to use only these subsets of characters. Since our model only includes a weak case feature, this method helps to produce labels that follow the case patterns that we expect to see most often. We use the restricted version of the Viterbi model to produce these word labels instead of just transforming the initial word label to have the case patterns since many characters look different in lowercase and uppercase.

7.2.2 Web-based Error Correction

We use a web-based error correction step to fix any errors in the initial text labels, which uses the web-based lexicon described in Section 7.1.

To correct errors, we build a list of hypotheses for possible word labels, evaluate each hypothesis based on the appearance and the global language information obtained from the lexicon, and choose the most likely hypothesis. We begin with the four initial word labels from the previous step, and add hypotheses to this set for all two-character edits of these strings. This means that each hypothesis added must have the same length as the original word labels, but can have up to two characters that are different. We add all two character edits because it allows us to correct a large amount of errors while maintaining a reasonable running time.

Next we calculate the language probability, p_l, for each hypothesis, which is the term frequency count normalized by the sum of all frequency counts in the hypothesis list. To get the final probability of a hypothesis, we multiply this by the appearance probability, p_a, of each character in the word. This value comes from the node marginals from the CRF model trained in the previous step. To summarize, the probability of a hypothesis h with characters $c_1 ... c_n$ in the error correction step is

$$p(h) = p_l(h) * \prod_{i=1}^{n} p_a(c_i).$$

We choose the hypothesis with the highest probability as the final word label for the error correction step. If none of the hypotheses can be found in the lexicon, we back off to the initial word label from the previous step. This allows us to label images with words that are not found in the lexicon.

This error correction step is important because prior to incorporating this global language information, the CRF model used only character bigram information. While bigrams are useful for improving labels, they contain local information. In practice, many words contain bigrams that are highly unlikely if looked at alone. For example, the word 'Amherst', contains the characters 'mh', which have a low bigram probability. However, as a word, Amherst is a common town name. To recognize words like this correctly, global language information is required.

7.3 Experiments

7.3.1 Implementation Details

We use a software package for graphical models by Mark Schmidt to implement the CRF model in this paper [51]. This package includes standard methods for parameter estimation, inference and decoding. Using this implementation, our method for finding initial word labels is efficient. It took an average of .09 seconds per image to find the four initial word labels.

	ICDAR 03 (S)	ICDAR 11
Without Error Correction	52.90	41.04
With Error Correction	62.76	48.86

Table 7.1: Word accuracy results with and without web-based error correction.

Our techniques for text segmentation and error correction are also efficient. We implemented bilateral regression segmentation and error correction in Matlab and the average running time for our unoptimized segmentation code on a standard desktop is around 3 seconds over the ICDAR 2003 test set. The smallest image in this set is 17 x 12 pixels and the largest is 630 x 1204 pixels. The average size is around 70 x 200 pixels. The average running time for our unoptimized error correction code is also 3 seconds per image.

7.3.2 Complete System Evaluation

We evaluate our complete system on the task of open-vocabulary word recognition using the ICDAR 2003 and ICDAR 2011 data sets. These are described in more detail in Chapter 3.

In order to compare against existing work, we follow the experiments of Mishra et al. [39] and present results on a subset of the ICDAR 2003 data set. It is created by removing all words with non-alphanumeric characters and all words with less than three characters. The evaluations on this subset are done in a case-insensitive way. For the ICDAR 2011 data set, we present results on the complete data set and, following previous work, evaluate results in a case-sensitive way.

Table 7.1 shows the word accuracy of our system with and without error correction. Performance increases by almost 10% on ICDAR 2003 and over 7.5% on ICDAR 2011. This shows the importance of using web-based error correction. Table 7.2 shows our results compared to existing methods. On the ICDAR 2003 data set our method increases word accuracy by over 4.5% over the existing state-of-the-art. For

	ICDAR 03 (S)	ICDAR 11
Neumann's Method [52]	-	33.11
KAIST AIPR System [52]	-	35.60
TH-OCR System [52]	-	41.2
Mishra et al. [39]	57.92	-
Our Method	**62.76**	**48.86**

Table 7.2: Open-vocabulary word accuracy results for word recognition on the ICDAR 2003 and ICDAR 2011 data sets. The first column is a subset (S) of the data set with all words with non alpha-numeric characters or less than 3 characters removed. The reduced set is evaluated in a case insensitive way. The second column includes the complete data set and is evaluated case sensitive.

	Total Edit Distance	Word Accuracy
PhotoOCR [4]	122.7	82.83
PicRead [45]	332.4	57.99
NESP [27]	360.1	64.20
PLT [29]	392.1	62.37
MAPS [28]	421.8	62.74
Feild's Method	422.1	47.95
PIONEER [67]	479.8	53.70
Baseline: ABBYY OCR	539.0	45.30
TextSpotter [43]	606.3	26.85

Table 7.3: ICDAR 2013 Robust Reading Competition results.

the ICDAR 2011 data set, our method increases word accuracy by over 7.5% over the best method submitted to the Robust Reading competition. These results show that our technique out-performs state-of-the-art methods for open-vocabulary word recognition. Figure 7.3 shows examples of words that were recognized correctly with this system and Figure 7.4 shows several failure cases.

7.3.3 ICDAR 2013 Robust Reading Competition Results

We also evaluated our system on the ICDAR 2013 data set by submitting to the ICDAR 2013 Robust Reading competition on the task of cropped word recognition.

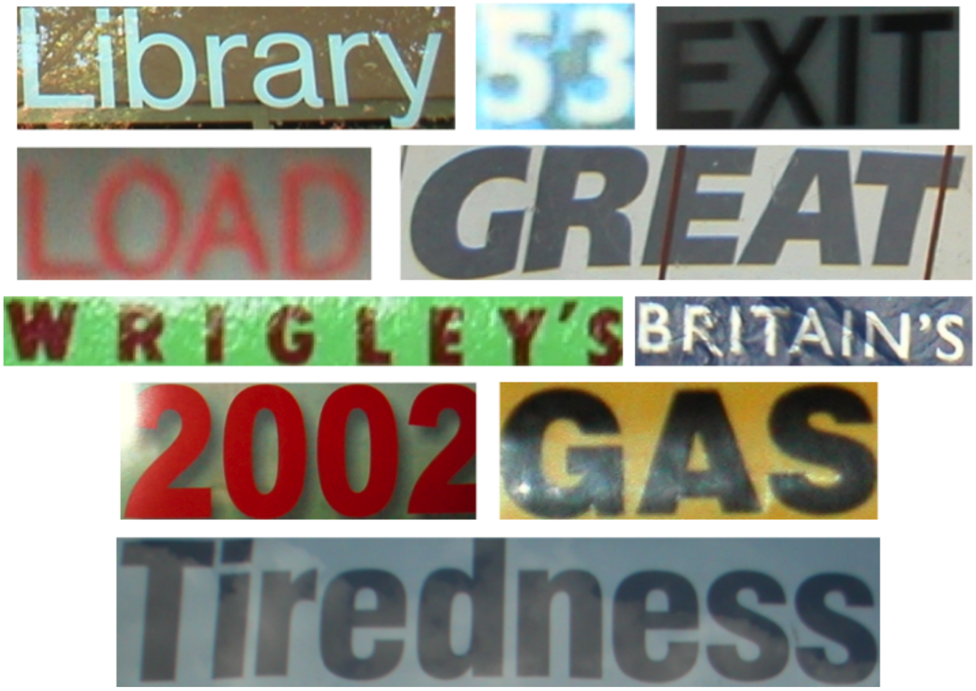


Figure 7.3: Sample images that we recognize correctly. This image is best viewed in color.

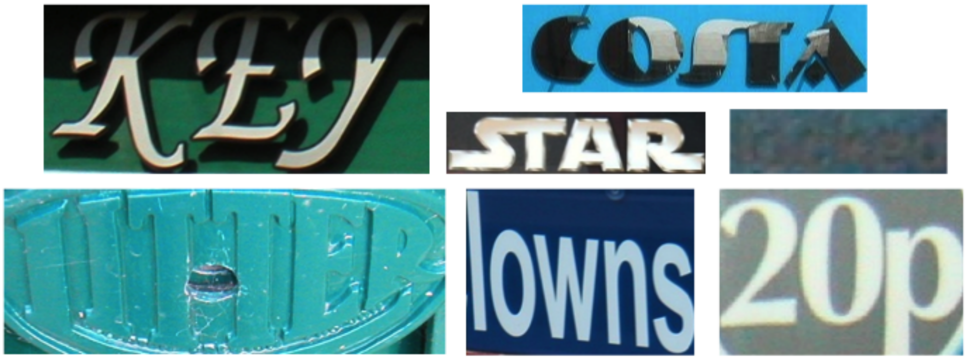


Figure 7.4: Sample images that we recognize incorrectly. Characteristics that make these images difficult include low resolution, abrupt lighting changes and low contrast. In addition, words that do not appear in the web-based lexicon, but look similar to something that does can be confused. Here 'lowns' is recognized as 'Towns' and '20p' is recognized as '200'. This image is best viewed in color.

The results of the competition are shown in Table 7.3. The evaluation metrics are word accuracy and total edit distance, where edit distance is defined as the number of insertions, deletions and substitutions required to transform a word label into the ground truth label. The baseline system is the commercial OCR system by ABBYY.

Our system placed 6th in this competition with a total edit distance of 422.1 and a word accuracy of 47.95. It is interesting to compare our system to the 5th place system. The edit distances of the two systems are just .3 apart, but the word accuracies differ by 14.8%. This shows that there must be many instances where we label almost all characters correctly, but do not label the entire word correctly.

7.4 Discussion

One of the main differences between the work we present here and other existing solutions is the technique used to detect character locations. Many recent techniques use a sliding window approach to evaluate all possible locations and sizes to find possible characters [45, 39, 65]. These approaches avoid relying on an initial hard segmentation step, but evaluating all sub-windows is expensive, and there is great potential for confusion when non-text areas exhibit character-like features. In contrast, a text segmentation based method can take advantage of coherence across an image. For example, the color characteristics of easier characters can help identify more difficult characters. In this chapter, we demonstrate that a segmentation-based approach can outperform sliding-window based approaches for the task of word recognition.

There are several possible directions for future work that would improve this word recognition technique. One of the main weaknesses is that it cannot recognize connected text. This stems from the choice we made to use a segmentation-based method to identify characters, so we can only recognize characters that are individual connected components. One way to address this might be to use a sliding window approach only when a connected component cannot be recognized well, since it is very computationally expensive. Similarly, a text splitting technique could be used instead and we could try to identify the parts of the connected text. A different approach would also be to train a character classifier that identifies two or three character groups in addition to single characters.

The other main weakness to be addressed is that our technique cannot handle missing characters. The final word label must be the same length as the number of characters detected. This is problematic when some number of characters are not detected, but the word is still recognizable with a high probability. An important extension would be to develop a method for evaluating if a segmented image may be missing a letter or group of letters, and then extend the error correction step to allow word labels of different lengths than the number of detected characters.

A last direction for future work is to incorporate better case features into the model. Right now we use case information in the CRF feature vectors, but they do not provide perfect case labels. In our experiments a case-insensitive evaluation of our system leads to better recognition performance, which means that there are words where we label the characters correctly, but in the wrong case.

7.5 Conclusion

In this chapter, we presented an efficient system for the task of open-vocabulary word recognition. We demonstrated a new approach to incorporating web-based language information that allows us to take advantage of a lexicon of over 13.5 million words that appear on the web for error correction. In our experiments, we presented state-of-the-art experimental results for open vocabulary word recognition using this system on two standard data sets, ICDAR 2003 and ICDAR 2011.

CHAPTER 8

END-TO-END SCENE TEXT RECOGNITION

In this chapter, we remove the assumption that text has already been located and describe an end-to-end scene text recognition system for automatically finding and labeling text in images of natural scenes. Recently, many new techniques for text detection and text recognition in natural images have been proposed, but the majority of these methods look at these two problems in isolation [17, 71, 74, 39, 65]. This has led to increased state of the art performance on each individual problem, but does not enable useful real-world applications which require an end-to-end solution. Towards this end, there have been several end-to-end solutions proposed recently as well. One approach is to combine detection and recognition in a feed-forward pipeline [42, 67, 37]. While this type of system can be used in real-world applications, it does not take advantage of the fact that information from detection and recognition can be shared to improve results. To address this, we show that performance can be improved by using recognition information to inform parameter choices in the detection phase. We use a state of the art text detection technique and perform image specific parameter adaptation. We also introduce a hybrid recognition component that uses an open-source OCR system when it is confident in the labeling, and passes more difficult images to our specialized recognition module. We evaluate this system compared to current state-of-the-art approaches on publicly available data sets.

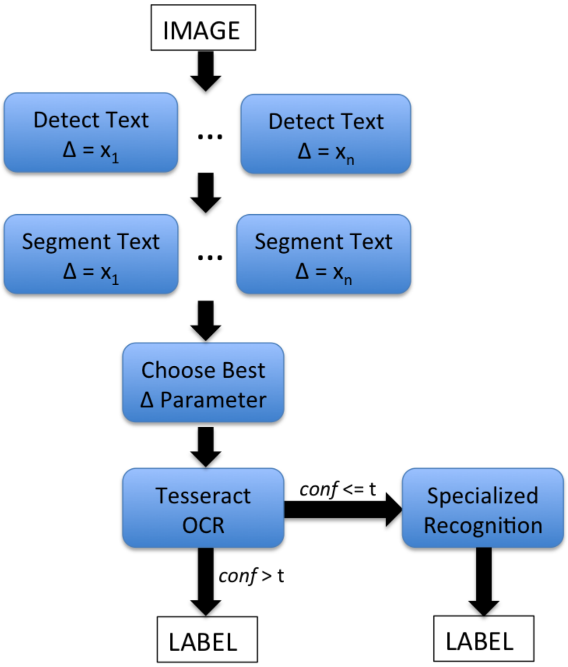


Figure 8.1: A step-by-step example of our final end-to-end system. First, text is detected and segmented using a state-of-the-art method over a range of settings for the delta parameter. Next, the delta parameter is optimized based on recognition information. Finally, each detection is recognized by either Tesseract, if it is confident of its label choice, or our own specialized recognition method.

8.1 End-to-End Scene Text Recognition

Here we combine the text recognition method from the previous chapter with a state-of-the-art text detection technique to create an end-to-end system. In this chapter we will refer to our recognition method as STR. In this section we will describe the detection method in more detail and explain how we can improve detection and recognition performance by using image-specific parameter adaptation and by creating a hybrid recognition component that uses an open-source optical character recognition (OCR) method. Our complete system is shown in Figure 8.1.

8.1.1 Text Detection and Segmentation

For text detection, we use the existing state-of-the-art method USTB_TexStar by Yin et al. [73]. This method first identifies maximally stable extremal regions

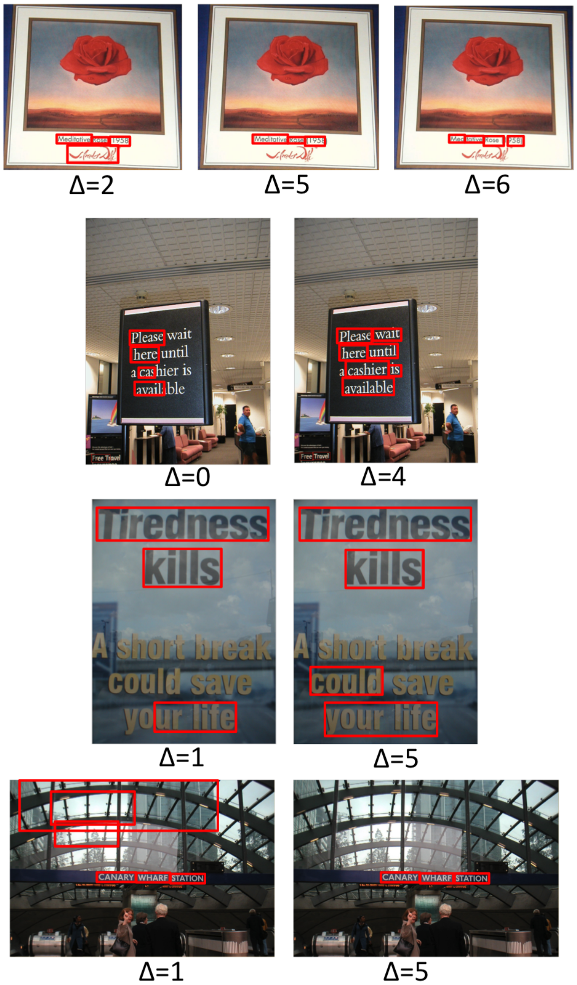
(MSERs) and then prunes them using the strategy of minimizing regularized variations of extremal regions (ERs) to extract character candidates. These character candidates are grouped into text candidates by adaptive single-link clustering in which similarity weights and a clustering threshold are learned by a self-training distance metric learning algorithm. Text candidates are evaluated using a character classifier and non-text regions are removed. Finally, each text region is divided into word regions by a word partition step [74]. One advantage of this method is that text segmentation is completed as part of the detection process, and we can use this output instead of performing segmentation as an additional step, using a method like the one described in Chapter 5. The segmentation method used in this detection process is called USTB_FuStar.

8.1.2 Image Specific Parameter Adaptation Using Recognition

The USTB_TexStar method is based on using MSERs to find candidate characters. The MSER algorithm identifies candidate connected components as those with a size and shape that stay relatively constant over a range of threshold values. This range is controlled by a parameter delta, and is a measure of the stability of a component.

We hypothesized that the best value of the delta parameter should depend on individual image characteristics, since images with sharper edges contain components that are stable over a larger range of threshold values. We confirmed experimentally that text detection output can change significantly based on the value of the delta parameter that is used. To illustrate this, Figure 8.2 shows sample images and their text detection output for different values of the delta parameter. This shows that selecting the value for the delta parameter is an important choice for accurate text detection.

For the text detection task, this parameter is found by learning the best value over a set of training images, and using that value for all test images. This makes


Δ=2 Δ=5 Δ=6

Δ=0 Δ=4

Δ=1 Δ=5

Δ=1 Δ=5

Figure 8.2: Sample images and their text detection output with different values for the delta parameter. This figure is best viewed in color.

sense when text detection is performed in isolation, but this globally chosen delta value can not be optimal for all images. Our goal is improve detection performance by automatically adapting the delta value to each image, instead of to each data set. We show how to use recognition information to determine the best parameter value for each image, which we can take advantage of since we are performing both tasks together.

At a high level, the idea is to perform text detection using a range of delta parameter values and to choose the delta value that results in detections that look the most like text. The output of the text detection step for each image is a set of bounding boxes identifying regions that contain text. We run text recognition on each bounding box, which gives us the appearance probability of the maximum probability character label for each character candidate. Using this information, we use the following score function to evaluate the image,

$$S_{\Delta} = \frac{1}{n} * \sum_{i=1}^{n} log(A_i).$$

In this equation, $A_1, A_2, ..., A_n$ represent the appearance probabilities of the characters in the image. This score function gives higher scores to images with characters that are recognized with a higher probability. Since we are comparing detections over the same image with different parameter values, this allows us to choose the parameter value that results in detections that look the most like text as possible.

This score function is simple, but the experiments in Section 8.2.1 show that it leads to a large increase in performance. We experimented with incorporating language information as well, but it did not improve performance. We also experimented with using a classifier to predict if the detection output from one parameter setting is better than the output from another setting using the appearance information, language information, the number of bounding boxes and the number of characters as features. This also lead to comparable performance, so we use the simple score function described above instead.

8.1.3 A Hybrid System for Text Recognition

For many years, commercially developed optical character recognition (OCR) systems were used as a baseline for performance for scene text recognition tasks. These systems were developed for use on documents and did not work well given the additional challenges of scene text images. Recently though, Milyaev et al. showed state-

Groundtruth	Tesseract Label	STR Label
PROPER	PROPEP	PROPER
FOOD	FUDE	FOOD
PRONTO	PRONTC	PRONTO
Professional	Profesgignal	Professional
Mining	Jmnlne	Mining

Groundtruth	Tesseract Label	STR Label
SUPERKINGS	SUPERKINGS	SUPERBEINGS
Counselling	Counselling	counselling
University	University	Universrwot
Estates	Estates	Eshter
Rettungsweg	Rettungsweg	RenungsWE

Table 8.1: The top of the table shows sample text where the STR label is correct but the Tesseract label is not. The bottom of the table shows sample text where the Tesseract label is correct but the STR label is not.

of-the-art recognition performance by combining a specialized binarization technique with Omnipage OCR [37].

Inspired by this increase in performance, we experimented with the open-source commercial OCR system Tesseract[1] developed by Google. We hypothesized that the set of images that could be recognized correctly by Tesseract and STR would be different, since both recognition systems have different strengths and weaknesses. We confirmed with initial experiments that there were some text images that Tesseract could label correctly but STR could not and that there were other text images that Tesseract could label correctly while STR could not. Table 8.1 shows some example words in both categories from the ICDAR 2011 data set. We describe a way to combine Tesseract and STR into a hybrid recognition system to improve overall performance.

Our method is simple but very effective for improving performance. The detection output from Tesseract includes a confidence score associated with each recognized

[1] https://code.google.com/p/tesseract-ocr/

	Precision	Recall	f
Δ = random choice	50.5	36.2	42.1
Δ = 1 (default)	49.5	37.9	42.9
Δ = 5 (learned)	55.3	38.4	45.4
Δ = optimized	58.6	39.9	47.5

Table 8.2: End-to-end recognition results on the ICDAR 2011 data set using STR and different methods for choosing the delta parameter.

word. We run Tesseract on each region from our text detector and examine the confidence score. If that confidence is above a threshold, meaning that Tesseract is confident in its label, we use the label. If it is below a threshold, we pass the the region to STR and use that label instead.

Here we also experimented with using a classifier to choose which recognition system to choose for each image. We used the confidence scores from each system as features in an SVM classifier. This lead to comparable performance, so we use the simpler threshold method instead.

8.2 Experiments

In this section we evaluate our contributions on the task of end-to-end text recognition using the ICDAR 2011 and ICDAR 2013 data sets. These are described in more detail in Chapter 3. We chose the ICDAR 2011 data set because it is used by all previous authors for this task and will allow a direct comparison to the current state-of-the-art. Following previous work, we evaluate label results in a case-sensitive way. There is no previous work using the ICDAR 2013 data set for the end-to-end task, so we provide our results here for future comparison.

8.2.1 Evaluation of Parameter Optimization

In this section we show that choosing the delta parameter using our automatic optimization method leads to better recognition performance than several other meth-

	Precision	Recall	f
Δ = random choice	84.4	64.9	73.4
Δ = 1 (default)	83.8	68.2	75.2
Δ = 5 (learned)	89.1	64.8	75.1
Δ = optimized	90.4	64.6	75.3

Table 8.3: End-to-end text detection results on the ICDAR 2011 data set using different methods for choosing the delta parameter.

ods. Table 8.2 shows the end-to-end recognition precision, recall and f measure for these methods. One choice is to randomly choose a delta parameter from a set of reasonable choices for each image. Here we choose delta from the discrete set zero to six. This leads to a precision value of 50.5, recall of 36.2 and an f measure of 42.1. Another option is to use the default setting of the text detector for all images, which in this case was the value one. This leads to a decrease in precision to 49.5, but a larger increase in recall to 37.9 and an increased f measure of 42.9. In addition to these, we also tried learning the best delta parameter on a set of training images and using it for all images. We selected the value that gave the best end-to-end recognition performance, which was five. This lead to a large increase in precision to 55.3 and an increase in recall to 38.4 for an f measure of 45.4. We can significantly improve both precision and recall by optimizing the delta parameter using the score function described in section 8.1.2. The precision is 58.6 and the recall is 39.9 for an f measure of 47.5.

We also include results for text detection precision, recall and f measure when using different values of the delta parameter. These are computed using the evaluation method developed by Wolf et al. that was adopted by the ICDAR competitions [68]. These results are shown in Table 8.3. These results show that there is not a large change in the f measure of text detection performance when the delta parameter changes. The performance using the optimized delta values is slightly higher than

	Precision	Recall	f
Tesseract	38.8	26.4	31.4
STR	58.6	39.9	47.5
Hybrid	61.7	42.0	50.0
Hybrid (oracle)	65.3	44.1	52.9

Table 8.4: End-to-end recognition results on the ICDAR 2011 data set using different recognition systems.

	Precision	Recall	f
Neumann et al. [42]	37.1	37.2	37.2
Neumann et al. [43]	39.4	37.8	38.6
Weinman et al. [67]	41.1	36.5	38.6
Neumann et al. [44]	44.8	45.4	45.2
Tesseract/STR Hybrid	61.7	42.0	50.0
Tesseract/STR Hybrid (case in.)	64.4	43.8	52.2

Table 8.5: A comparison of end-to-end recognition results for current methods on the ICDAR 2011 data set.

the learned delta values, but it does not suggest the large increase in end-to-end recognition performance that we see.

8.2.2 Evaluation of Hybrid System

In this section we show that combining recognition systems into a hybrid system using a confidence threshold leads to better end-to-end recognition performance than using either system alone. These results are shown in Table 8.4.

Tesseract has a precision of 38.8, recall of 26.4 and an f measure of 31.4. STR performs significantly better on this problem, with a precision of 58.6, a recall of 39.9 and an f-measure of 47.5. We learn the threshold of $t = 75$ to combine these systems by choosing the threshold that leads to the best end-to-end performance on the ICDAR 2011 training set. This hybrid system leads to an increase of precision by

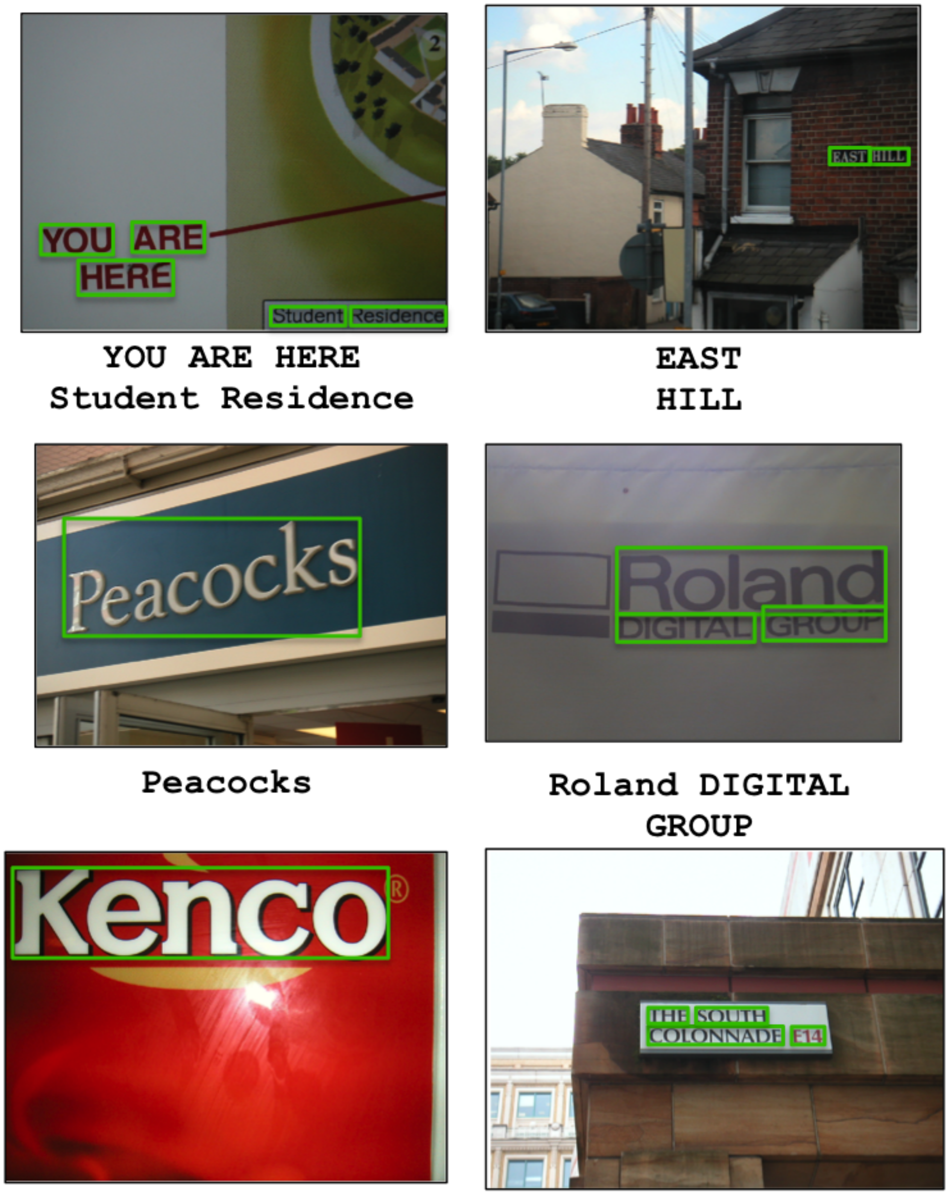
Data Subset	Tesseract Label Correct	Tesseract Label Used	STR Label Correct	# of Images
A	✓	✓	✓	225
B	✓	✓		45
C	✓		✓	35
D	✓			9
E		✓	✓	20
F		✓		54
G			✓	194
H				227

Table 8.6: A description of the eight data subsets of the ICDAR 2011 data set and the number of images from the data set in each category.

3.1% to 61.7, an increase in recall of 2.1% to 42.0 and an increase to the f measure of 2.5% to 50.0.

For comparison, we also present results of our hybrid system with an oracle to select the confidence value for Tesseract. If Tesseract had perfect confidence values that were above the threshold when the label was correct and below the threshold when the label was incorrect, then the precision would be 65.3, recall would be 44.1 and the f measure would be 52.9. This shows that performance could increase even more if we could improve the confidence scores produced by Tesseract.

In Table 8.5 we show a comparison of our hybrid system to other published methods for this task. Our precision and f-measure are higher than all of the other systems. Our precision is 16.9% higher than the current best at 61.7 and our f measure is 4.8% higher at 50.0. Our recall is 2.8% lower than the current best at 42.0. This increase in the f measure is significant increase, given the small improvements shown on this task in recent years. Figure 8.3 shows sample images where our system detects and recognizes all of the existing text. Figure 8.4 shows sample images where there are incorrect results due to missed detections or incorrect labels. Figure 8.5 shows difficult images from the test set where we do not recognize any text.



YOU ARE HERE
Student Residence

EAST
HILL

Peacocks

Roland DIGITAL
GROUP

Kenco

THE SOUTH
COLONNADE E14

Figure 8.3: Sample images with text detection and recognition output. We correctly detect and recognize all of the text in these images.

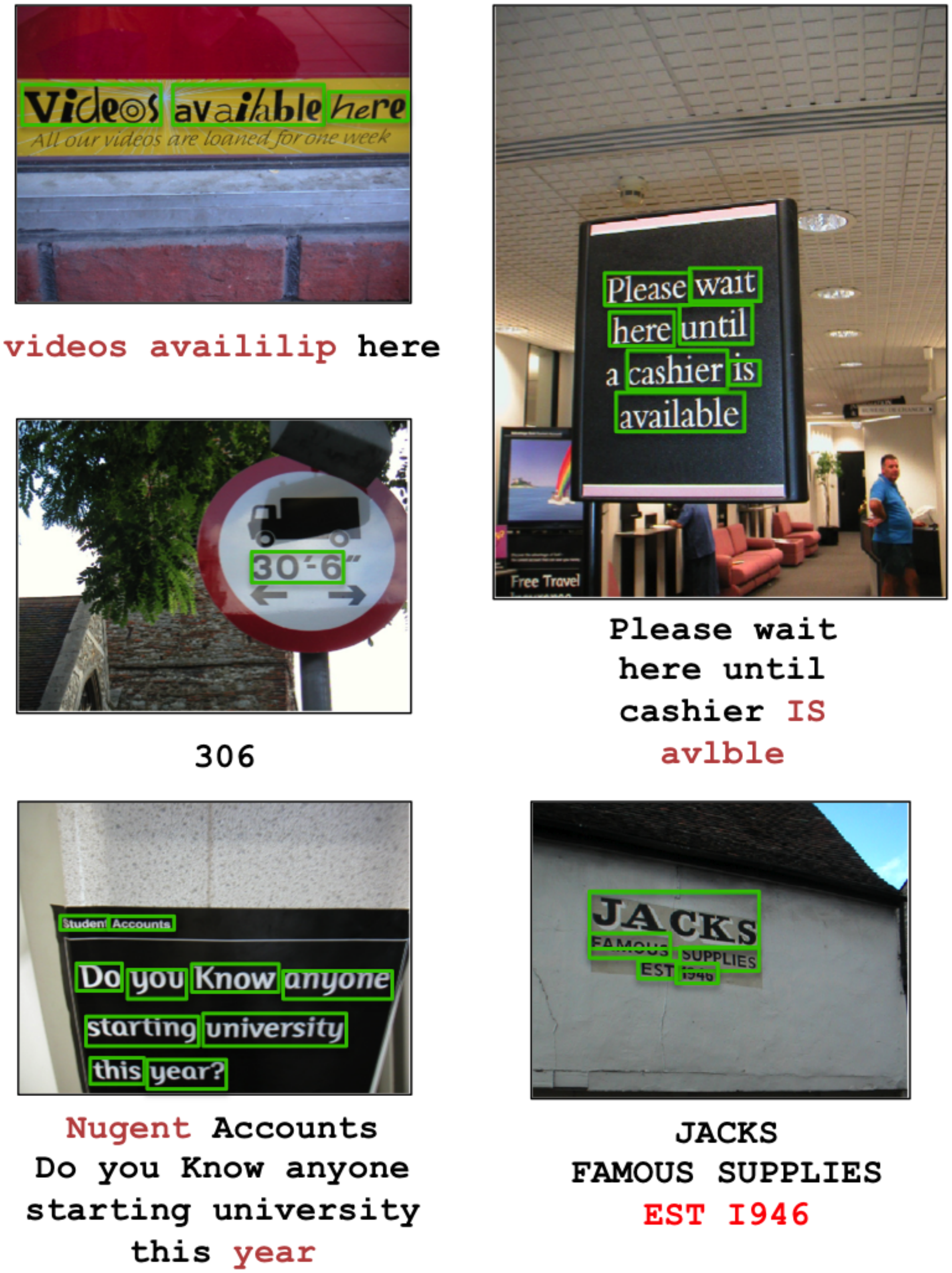


Figure 8.4: Sample images that have incorrect detection and/or recognition results. Errors include missing detections, incorrect word labels, capitalization errors and missing punctuation.

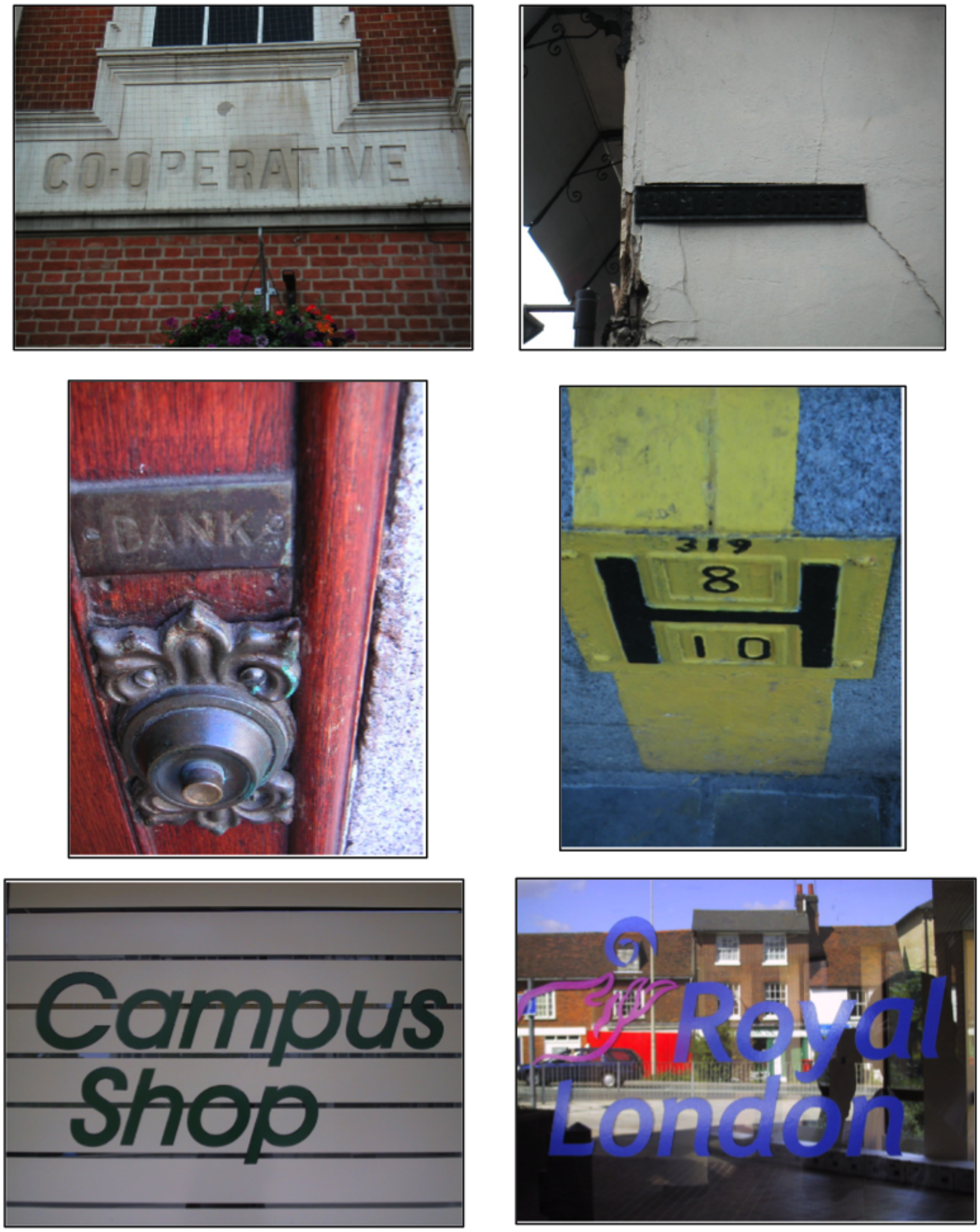


Figure 8.5: Sample images where we do not detect any text. Difficult characteristics include low contrast between text and background, unusual text layout and complex background scenes.

Data Subset	Groundtruth	Tesseract	Conf.	ST Rec.
A	the	the	87	the
	parcel	parcel	91	parcel
	Bookshop	Bookshop	88	Bookshop
B	tickets	tickets	87	tiger
	FAMOUS	FAMOUS	86	Famous
	Centre	Centre	88	centre
C	Factory	Factory	59	Factory
	access	access	71	access
	ARE	ARE	74	ARE
D	LITTER	LITTER	53	LETTER
	Centre	Centre	69	cente
	Education	Education	74	Educaton
E	COLCHESTER	COLCH	89	COLCHESTER
	Cycles	Cycies	79	Cycles
	EXIT	XIT	94	EXIT
F	HOT MILK	HOT	93	HOT
	area!	area	93	area
	Virtual	irt	89	virtual
G	PROPER	PROPEP	74	PROPER
	Psychology	Psycholo9V	72	Psychology
	Professional	Profesgignal	58	Professional
H	SHINING	BHHWIE	46	SHINE
	priory	Dkiony	63	PRIORI
	sure	SUTE	72	Sure

Table 8.7: Sample output from Tesseract and STR for each of the eight data subsets in Figure 8.6.

There is also a method for this task by Milyaev et al. [37] that uses a commercial OCR system. We put this method in a different category because the details of the recognition component are not described or published, so it cannot be implemented and extended. This method has shown the highest precision, recall and f measure for this task with a precision of 66, recall of 46 and an f measure of 54. It is interesting to note that the oracle version of our hybrid system has a performance almost comparable to this method.

We also describe the eight possible data subsets created by the correctness of Tesseract's label, if Tesseract's label is used and the correctness of STR's label in

	Precision	Recall	f
BR + Tesseract	42.8	28.3	34.1
USTB_FuStar + Tesseract	38.8	26.4	31.4
BR + STR	54.3	35.9	43.2
USTB_FuStar + STR	58.6	39.9	47.5
BR + Hybrid	58.8	38.9	46.9
USTB_FuStar + Hybrid	61.7	42.0	50.0
BR + Hybrid (oracle)	61.5	40.7	49.0
USTB_FuStar + Hybrid (oracle)	65.3	44.1	52.9

Table 8.8: End-to-end recognition results on the ICDAR 2011 data set using different segmentation and recognition method combinations.

Table 8.6. This table includes the number of images in each category. This is useful for understanding the output of each system and the hybrid system. It shows that there are 225 words that both systems label correctly, 45 images that Tesseract labels correctly but STR does not and 194 images that STR labels correctly but Tesseract does not. We can also see that data subsets A, B, C and G are labelled correctly by the hybrid system. Data subsets D and E are errors that could be corrected if the confidence values from Tesseract were more accurate. Subsets F and H represent images that neither system labelled correctly. Table 8.7 shows sample words from each of the data subsets in Table 8.6 and the Tesseract label and confidence score as well as the STR label.

8.2.3 Evaluation of Segmentation Method

For completeness, we also evaluate our end-to-end system using the segmentation method described in Chapter 6. Table 8.8 shows the precision, recall and *f* measure of combining bilateral regression segmentation with Tesseract, STR and the hybrid recognition system compared to using USTB_FuStar for segmentation. The precision, recall and *f* measure is much higher when using bilateral regression segmentation with Tesseract. However, when combined with STR or the hybrid system, using USTB_FuStar leads to significantly better performance.

8.2.4 Evaluation on ICDAR 2013 Data Set

We also present results on the ICDAR 2013 Robust Reading data set [25]. Table 8.9 includes results comparing the components we use for each task to other submissions to the 2013 Robust Reading competition. This includes text detection, text segmentation and cropped word recognition. We also include results for end-to-end text recognition. To our best knowledge, there are no previously published results for end-to-end text recognition on this data set.

As shown in Table 8.9, the text detection and segmentation methods we use perform well. They were first place and 3rd place respectively in the competition. Our cropped word recognition approach is also fairly competitive. Note that in our end-to-end system, STR is only the basic recognition approach; actually, our hybrid recognition framework described in this paper has a higher recognition accuracy. Moreover, our method also achieves better performance with the addition of automatic parameter optimization. Our end-to-end text recognition system performs well with f measures for case-sensitive and case-insensitive on this dataset of 50.6% and 53.2% respectively.

8.3 Discussion

It is interesting to note that both the parameter adaptation score function and method for creating a hybrid system are both simple but effective components. We experimented with score functions that also incorporated language information, but the performance was slightly worse. We also tried building a logistic regression classifier to predict whether one set of text detection output was better than another. We tried combinations of character appearance, language, number of bounding boxes and

(a)	**Text detection**			
Methods	Recall	Precision	f	***Results***
USTB_TexStar	66.45	88.47	75.89	1st place
TextSpotter	64.84	87.51	74.49	2nd place
CASIA_NLPR	68.24	78.89	73.18	3rd place
(b)	**Text segmentation** (atom based results)			
Methods	Recall	Precision	f	***Results***
I2R_NUS_FAR	68.64	80.59	74.14	1st place
NSTextractor	63.38	83.57	72.09	2nd place
USTB_FuStar	68.03	72.46	70.18	3rd place
(c)	**Cropped word recognition**			
Methods	T.E.D.	C.R.W.	C.R.W.(u)	***Results***
PhotoOCR	122.7	82.83	85.30	1st place
PicRead	332.4	57.99	61.92	2nd place
NESP	360.1	64.20	64.84	3rd place
STR	422.1	47.95	52.33	6th place
(d)	**End-to-end text recognition**			
Our method	Recall	Precision	f	Remarks
	44.1	67.0	53.2	Case-insensitive
	41.9	63.7	50.6	Case-sensitive

Table 8.9: Results on ICDAR 2013 data set, where "USTB_TexStar", "USTB_FuStar" and "STR" are the text detection, text segmentation, and (basic) word recognition methods used in our system. "T.E.D.", "C.R.W." and "u" represent "Total Edit Distance", "Correctly Recognized Words"(%) and "upper" respectively. Competition results come from [25].

number of characters as features. The performance of this classifier was comparable to the simple score function, so we chose to use the simple component. We saw the same trend for the hybrid system component. We used an SVM classifier with an RBF kernel to predict which recognition system should be used for each image. Our features were the confidence scores from both recognition systems. Using this classifier in place of the threshold led to a .3% increase in precision, recall and f measure. These results are comparable, so we chose the more simple threshold method.

In this chapter we showed that the idea of adapting parameters to each image increased performance considerably. One direction of future work is to extend this idea to include adapting multiple parameters simultaneously. For the task of text detection using the USTB_TexStar method, one other parameter to adapt might include the parameter that controls where to split text lines into words. There are many examples in the output where text lines are detected correctly, but they are not broken into words correctly. The way each task is structured, even if we identify all of the characters correctly, each bounding box can only be mapped to one ground truth word, so it will be counted as incorrect. Other parameters to investigate are the parameters that control the minimum and maximum size component to accept. These parameter help filter false positives, but also filter good detections in some cases.

8.4 Conclusion

We have presented a system for end-to-end text recognition with state-of-the-art performance compared to existing published methods. We introduced the idea of image specific parameter adaptation using recognition information and showed that this increases performance significantly. We also compared two recognition systems, Tesseract and STR, and showed that combining these into a hybrid system leads to better performance than using either system individually. Finally, we compared our system to the current state of the art on the publicly available ICDAR 2011 data set and presented the first end-to-end results on the ICDAR 2013 data set.

CHAPTER 9
CONCLUSIONS

In this thesis we focused on the problem of recognizing text in images of natural scenes. This problem has important real-world applications, and existing solutions for recognizing text in documents do not work well due to the additional challenging characteristics of scene text images. To improve recognition performance, we presented methods that incorporated new information sources into models and we composed simple components into highly effective systems. We focused on three different scene text tasks, each with a different number of prior assumptions: character recognition, cropped word recognition and end-to-end text recognition. Throughout the thesis we aimed to develop methods that allow us to reduce necessary assumptions.

First, we described a novel approach to incorporating similarity information. We found that the problem of deciding whether two character images are equivalent is much easier than deciding the character label of an image, and we took advantage of this information to constrain our label search space. We did this by training a similarity expert that learned to classify each pair of characters in a sign image as equivalent or not. We formulated the search for the maximum likelihood interpretation of a sign as an integer program and incorporated the equivalence information as constraints. Since the labels produced by the integer program only took into account local language information via bigram probabilities, we also presented an error correction step based on global language information from a search engine. We found that adding similarity information increased overall performance, and we demonstrated word error reductions of more than 30% relative to previous methods on the same data set. Our system had a word accuracy rate of 92.56%.

Next we presented a new language model for scene text recognition. We modeled words with a probabilistic context free grammar, which captured information about syllables. We found that this was a better model of language information across syllable boundaries, so words with unlikely bigrams that cross syllable boundaries were not penalized. Also, since words are made up of syllables, labels produced by this model are pronounceable. This eliminates all incorrect labels that a human would know are wrong since they cannot even be pronounced. We compared this model to the commonly used bigram model and showed that using more sophisticated language information improves character recognition performance. We also showed benefits of this model compared to a dictionary model.

We also presented a new model for segmenting text in natural scene images called bilateral regression segmentation. This method allows us to remove the assumption that character detection is complete. We can use this technique to divide each image into foreground text and background, and consider each foreground connected component as a character. We used a regression-based technique to model smooth color changes in just the subset of pixels that belong to the foreground text, while ignoring the background pixels altogether. We showed that it is suitable for segmenting images that are not handled well by existing methods, those with color changes caused by lighting and complex backgrounds. We evaluated this method compared to the current state-of-the-art segmentation method and showed performance improvement of up to almost 2%. We also showed that our cropped word spotting system led to increased performance over current methods on some data sets by 11%.

Next we removed the assumption that word labels must be drawn from a pre-specified lexicon and described an efficient system for the task of open-vocabulary word recognition. We demonstrated a new approach to incorporating web-based language information that allowed us to take advantage of a lexicon of over 13.5 million words that appear on the web for error correction. In our experiments, we presented state-of-the-art experimental results for open vocabulary word recognition using this system on two standard data sets, ICDAR 2003 and ICDAR 2011. The word recognition accuracies were 62.76% and 48.86% respectively.

Finally, in the last chapter of this thesis we presented a system for end-to-end text recognition that does not require any assumptions. We combined our recognition system from Chapter 7 with a state-of-the-art text detector and improved performance by introducing the idea of image specific parameter adaptation using recognition information. We showed that adapting text detection parameters to each image significantly improves recognition performance. We also compared two recognition systems, Tesseract and STR, and showed that combining these into a hybrid system leads to better performance than using either system individually. We compared our system to the current state of the art methods on the ICDAR 2011 data set and presented the first end-to-end results on the ICDAR 2013 data set.

9.1 Future Work

The evaluation and analysis of the methods developed in this thesis suggest several interesting directions for future work. One extension is to integrate bilateral regression segmentation from Chapter 6 with our word recognition method from Chapter 7. Right now, our segmentation method uses recognition information to automatically select the best segmentation choice, but it uses a nearest neighbor classifier to recognize characters, while our recognition method from uses a CRF classifier. We expect segmentation performance to increase with the integration of a better character classifier. Related to this, another avenue for future work is to explore the use of different character classifiers. Our systems are designed so that it is easy to replace the classifiers we use with other methods. Recently, state-of-the-art character classification was demonstrated using a deep-learning approach [12], and future work should explore whether this technique can be used to improve our system.

One of the biggest challenges for our recognition approach is connected text, either due to the font type or image blur. This is a challenge because our method assumes that each connected component is one distinct character. Future research should explore how to adapt this method so it does not rely on having separated characters. One possible way to address this is to use a sliding window character classification approach. Since we originally chose to use a segmentation-based approach due to computational complexity, the sliding window could be used only when a connected component could not be recognized well. Another possible approach is to explore using a text splitting technique to try to identify the parts of the connected text. Alternatively, a classifier could be trained that identifies two and three character groups in addition to single characters.

Another direction for future work is to extend our recognition technique to handle missing characters. As described, the final word label must be the same length as the number of characters detected. This is problematic when some number of characters are not detected but the word is still recognizable with a high probability. Future work should explore how to evaluate if a segmented image is missing a letter or group of letters and extend the error correction step to allow word labels with lengths different from the number of detected characters.

In addition to future work that improves the components we present in this thesis, there are many interesting research directions for the scene text community in general. First, it is important to expand the type of data sets used to evaluate text recognition systems. Most of the scene text community evaluates new techniques using the ICDAR competition data, which makes assumptions about the type of text to be recognized. Most of the text in the ICDAR data sets is oriented horizontally and captured from the front. In the future, scene text recognition techniques also need to focus on text captured at a wider variety of viewing angles, which is a much more difficult task. The Street View Text data set took one step in this direction, as this data set is significantly harder to recognize than the ICDAR data set due to things like low resolution and blur.

Very recently, Google has demonstrated incredible recognition performance using the latest techniques for character classification and language modeling [4]. They train their models with extremely large amounts of data and use vast computational resources to significantly improve performance over existing techniques. This demonstration leads to the research question of how to adapt this technique to the case where computational resources are limited. Also since they have shown that having more training data improves performance, what are some methods that non-industrial researchers can use to generate larger quantities of training data.

Related to this, another important area for future research is how to implement a scene text system in real time on a restricted architecture, like a smartphone. Some of the important applications of scene text recognition include being able to translate text into another language and aiding navigational tasks for people with low vision and in these cases, it may not always be feasible for the user to have internet connectivity. It would be useful for this type of system to be able to run in real-time without a lot of memory or computational power.

BIBLIOGRAPHY

[1] Aurich, Volker, and Weule, Jorg. Non-linear gaussian filters performing edge preserving diffusion. In *DAGM Symposium* (1995).

[2] Bai, Bo, Yin, Fei, and Liu, Cheng Lin. Scene text localization using gradient local correlation. In *International Conference on Document Analysis and Recognition* (2013).

[3] Bertsimas, D., and Tsitsiklis, J. N. *Introduction to Linear Optimization.* Athena Scientific, 1997.

[4] Bissacco, Alessandro, Cummins, Mark, Netzer, Yuval, and Neven, Hartmut. Photoocr: Reading text in uncontrolled conditions. In *International Conference on Computer Vision* (2013).

[5] Brants, Thorsten, and Franz, Alex. Web 1t 5-gram version 1. Linguistic Data Consortium, Philadelphia, 2006.

[6] Buades, A., Coll, B., and Morel, J.M. The staircasing effect in neighborhood filters and its solution. *IEEE Transactions on Image Processing 15* (2006).

[7] Burges, C. J. C. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery 2*, 2 (1998), 121–167.

[8] Chang, C., and Lin, C. LIBSVM: a library for support vector machines. Software available at http://www.csie.ntu.edu.tw/ cjlin/libsvm, 2001.

[9] Chen, X., Yang, J., Zhang, J., and Waibel, A. Automatic detection and recognition of signs from natural scenes. *IEEE Transactions on Image Processing 13*, 1 (2004).

[10] Chen, X., and Yuille, A.L. Detecting and reading text in natural scenes. In *International Conference on Computer Vision and Pattern Recognition* (2004).

[11] Choudhury, P., and Tumblin, J. The trilateral filter for high contrast images and meshes. In *Eurographics Symposium on Rendering* (2003).

[12] Coates, A., Carpenter, B., Case, C., Satheesh, S., Suresh, B., Wang, T., Wu, D.J., and Ng, A.Y. Text detection and character recognition in scene images with unsupervised feature learning. In *International Conference on Document Analysis and Recognition* (2011).

[13] De Campos, T., Babu, B., and Varma, M. Character recognition in natural images. In *International Conference on Computer Vision Theory and Applications* (2009).

[14] Donoser, M., Bischof, H., and Wagner, S. Using web search engines to improve text recognition. In *International Conference on Pattern Recognition* (2008).

[15] Donoser, M., Bischof, H., and Wagner, S. Using web search engines to improve text recognition. In *International Conference on Pattern Recognition* (2009).

[16] Elad, M. On the origin of the bilateral filter and ways to improve it. *IEEE Transactions on Image Processing 11*, 10 (2002), 1141–1151.

[17] Epshtein, B., Ofek, E., and Wexler, Y. Detecting text in natural scenes with stroke width transform. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on* (2010), IEEE, pp. 2963–2970.

[18] Goel, Vibhor, Mishra, Anand, Alahari, Karteek, and Jawahar, CV. Whole is greater than sum of parts: Recognizing scene text words. In *International Conference on Document Analysis and Recognition* (2013).

[19] Guillaumin, M., Verbeek, J., and Schmid, C. Is that you? Metric learning approaches for face identification. In *International Conference on Computer Vision* (2009).

[20] Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., and Witten, I. H. The WEKA data mining software: An update. In *SIGKDD Explorations, Volume 11, Issue 1* (2009).

[21] Jurafsky, D., Wooters, C., Segal, J., Stolcke, A., Fosler, E., Tajchman, G., and Morgan, N. Using a stochastic context-free grammar as a language model for speech recognition. In *IEEE International Conference on Acoustics Speech and Signal Processing* (1995), vol. 1, pp. 189–189.

[22] J.Weinman, and Learned-Miller, E. G. Improving recognition of novel input with similarity. In *International Conference on Computer Vision and Pattern Recognition* (2006), vol. 1, pp. 308–315.

[23] K. Müller. Probabilistic Context-Free Grammars for Syllabification and Grapheme-to-Phoneme Conversion. In *Proceedings of the Conference on Empirical Methods on Natural Language Processing* (2001).

[24] K. Muller. Probabilistic context-free grammars for phonology. In *Proceedings of the ACL Workshop on Morphological and Phonological Learning* (2002), Association for Computational Linguistics.

[25] Karatzas, Dimosthenis, Shafait, Faisal, Uchida, Seiichi, Iwamura, Masakazu, i Bigorda, Lluis Gomez, Mestre, Sergi Robles, Mas, Joan, Mota, David Fernandez, Almazan, Jon Almazan, and de las Heras, Lluis Pere. Icdar 2013 robust readubg competition. In *International Conference on Document Analysis and Recognition* (2013).

[26] Kita, K., and Wakahara, T. Binarization of color characters in scene images using k-means clustering and support vector machines. In *International Conference on Pattern Recognition* (2010).

[27] Kumar, Deepak, Prasad, MN, and Ramakrishnan, AG. Maps: Midline analysis and propagation of segmentation. In *Proceedings of the Eighth Indian Conference on Computer Vision, Graphics and Image Processing* (2012), ACM.

[28] Kumar, Deepak, Prasad, MN Anil, and Ramakrishnan, AG. Nesp: Nonlinear enhancement and selection of plane for optimal segmentation and recognition of scene word images. In *IS&T/SPIE Electronic Imaging* (2013), International Society for Optics and Photonics.

[29] Kumar, Deepak, and Ramakrishnan, AG. Power-law transformation for enhanced recognition of born-digital word images. In *Signal Processing and Communications (SPCOM), 2012 International Conference on* (2012), IEEE.

[30] Kumar, N., Berg, A. C., Belhumeur, P. N., and Nayar, S. K. Attribute and simile classifiers for face verification. In *International Conference on Computer Vision* (2009).

[31] Lari, K., and Young, SJ. Applications of stochastic context-free grammars using the inside-outside algorithm. *Computer speech & language 5*, 3 (1991), 237–257.

[32] Lee, Jung-Jin, Lee, Pyoung-Hean, Lee, Seong-Whan, Yuille, Alan, and Koch, Christof. Adaboost for text detection in natural scene. In *Document Analysis and Recognition (ICDAR), 2011 International Conference on* (2011), pp. 429–434.

[33] Lowe, D. Distinctive image features from scale-invariant keypoints. *Int. Journal of Computer Vision* (2003).

[34] Lucas, Simon M. Icdar 2005 text locating competition results. In *Document Analysis and Recognition, 2005. Proceedings. Eighth International Conference on* (2005), pp. 80–84.

[35] Lucas, S.M., Panaretos, A., Sosa, L., Tang, A., Wong, S., and Young, R. Icdar 2003 robust reading competitions. In *International Conference on Document Analysis and Recognition* (2003).

[36] Miller, E.G., and Viola, P.A. Ambiguity and constraint in mathematical expression recognition. In *Proceedings of the National Conference of Artificial Intelligence* (1998), pp. 784–791.

[37] Milyaev, Sergey, Barinova, Olga, Novikova, Tatiana, Lempitsky, Victor, and Kohli, Pushmeet. Image binarization for end-to-end text understanding in natural images. In *International Conference on Document Analysis and Recognition* (2013).

[38] Mishra, A., Alahari, K., and Jawahar, CV. An mrf model for binarization of natural scene text. In *International Conference on Document Analysis and Recognition* (2011).

[39] Mishra, A., Alahari, K., and Jawahar, CV. Scene text recognition using higher order language priors. In *British Machine Vision Conference* (2012).

[40] Mishra, A., Alahari, K., and Jawahar, CV. Top-down and bottom-up cues for scene text recognition. In *Computer Vision and Pattern Recognition* (2012).

[41] Neumann, L., and Matas, J. A method for text localization and recognition in real-world images. In *Asian Conference on Computer Vision* (2010).

[42] Neumann, Lukáš, and Matas, Jirı. Real time scene text localization and recognition. In *Computer Vision and Pattern Recognition* (2012).

[43] Neumann, Lukáš, and Matas, Jirı. On combining multiple segmentations in scene text recognition. In *International Conference on Document Analysis and Recognition* (2013).

[44] Neumann, Lukáš, and Matas, Jirı. Scene text localization and recognition with oriented stroke detection. In *International Conference on Computer Vision* (2013).

[45] Novikova, T., Barinova, O., Kohli, P., and Lempitsky, V. Large-lexicon attribute-consistent text recognition in natural images. In *European Conference on Computer Vision* (2012).

[46] Oprean, Cristina, Likforman-Sulem, Laurence, Popescu, Adrian, and Mokbel, Chafic. Using the web to create dynamic dictionaries in handwritten out-of-vocabulary word recognition. In *International Conference on Document Analysis and Recognition* (2013).

[47] Otsu, N. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics 9* (1979).

[48] Phan, Trung Quy, Shivakumara, Palaiahnakote, and Tan, Chew Lim. Text detection in natural scenes using gradient vector flow-guided symmetry. In *International Conference on Pattern Recognition* (2012), pp. 3296–3299.

[49] Quandt, R.E., and Ramsey, J.B. Estimating mixtures of normal distributions and switching regressions. *Journal of the American Statistical Association 73* (1978), 730–738.

[50] Saidane, Z., and Garcia, C. Automatic scene text recognition using a convolutional neural network. In *International Workshop on Camera-Based Document Analysis and Recognition* (2007).

[51] Schmidt, Mark. Ugm: Matlab code for undirected graphical models. http://www.di.ens.fr/ mschmidt/Software/UGM.html, 2013.

[52] Shahab, A., Shafait, F., and Dengel, A. Icdar 2011 robust reading competition challenge 2: Reading text in scene images. In *International Conference on Document Analysis and Recognition* (2011).

[53] Shi, Cunzhao, Wang, Chunheng, Xiao, Baihua, Zhang, Yang, and Gao, Song. Scene text detection using graph model built upon maximally stable extremal regions. *Pattern Recognition Letters* (2012).

[54] Smith, D., Feild, J., and Learned-Miller, E. Enforcing similarity constraints with integer programming for better scene text recognition. In *International Conference on Computer Vision and Pattern Recognition* (2011).

[55] Thillou, C., and Gosselin, B. Color binarization for complex camera-based images. In *Proc. Electronic Imaging Conference of the International Society for Optical Imaging* (2005).

[56] Tian, Shangxuan, Lu, Shijian, Su, Bolan, and Tan, Chew Lim. Scene text recognition using co-occurrence of histogram of oriented gradients. In *International Conference on Document Analysis and Recognition* (2013).

[57] Tomasi, C., and Manduchi, R. Bilateral filtering for gray and color images. In *International Conference on Computer Vision* (1998).

[58] Vedaldi, A., and Fulkerson, B. VLFeat: An open and portable library of computer vision algorithms. http://www.vlfeat.org/, 2008.

[59] Viterbi, A. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory* (1967).

[60] Wang, B., Li, X.F., Liu, F., and Hu, F.Q. Color text image binarization based on binary texture analysis. *Pattern Recognition Letters 26* (2005).

[61] Wang, K., Babenko, B., and Belongie, S. End-to-end scene text recognition. In *International Conference on Computer vision* (2011).

[62] Wang, K., and Belongie, S. Word spotting in the wild. In *European Conference on Computer vision* (2010).

[63] Wang, T., Wu, D.J., Coates, A., and Ng, A.Y. End-to-end text recognition with convolutional neural networks. In *International Conference on Pattern Recognition* (2012).

[64] Wang, X., Huang, L., and Liu, C. A novel method for embedded text segmentation based on stroke and color. In *International Conference on Document Analysis and Recognition* (2011).

[65] Weinman, J. Typographical features for scene text recognition. In *International Conference on Pattern Recognition* (2010).

[66] Weinman, J., Learned-Miller, E., and Hanson, A. Scene text recognition using similarity and a lexicon with sparse belief propagation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2009).

[67] Weinman, Jerod, Butler, Zachary, Knoll, Dugan, and Feild, Jacqueline. Toward integrated scene text reading. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2013).

[68] Wolf, Christian, and Jolion, Jean-Michel. Object count/area graphs for the evaluation of object detection and segmentation algorithms. *International Journal of Document Analysis and Recognition (IJDAR) 8*, 4 (2006), 280–296.

[69] Yao, Cong, Bai, Xiang, Liu, Wenyu, Ma, Yi, and Tu, Zhuowen. Detecting texts of arbitrary orientations in natural images. In *International Conference on Computer Vision and Pattern Recognition* (2012), pp. 1083–1090.

[70] Ye, Qixiang, and Doermann, David. Scene text detection via integrated discrimination of component appearance and consensus. In *International Conference on Document Analysis and Recognition* (2013).

[71] Yi, Chucai, and Tian, YingLi. Text string detection from natural scenes by structure-based partition and grouping. *Image Processing, IEEE Transactions on 20*, 9 (2011), 2594–2605.

[72] Yi, Chucai, Yang, Xiaodong, and Tian, Yingli. Feature representations for scene text character recognition: A comparative study. In *International Conference on Document Analysis and Recognition* (2013).

[73] Yin, X.-C., Yin, X., Huang, K., and Hao, H.-W. Robust text detection in natural scene images. *IEEE Transactions on Pattern Analysis and Machine Intelligence (preprint)* (2013).

[74] Yin, Xuwang, Yin, Xu-Cheng, Hao, Hong-Wei, and Iqbal, Khalid. Effective text localization in natural scene images with mser, geometry-based grouping and adaboost. In *Pattern Recognition (ICPR), 2012 21st International Conference on* (2012), pp. 725–728.

[75] Yokobayashi, M., and Wakahara, T. Segmentation and recognition of characters in scene images using selective binarization in color space and gat correlation. In *International Conference on Document Analysis and Recognition* (2005).

[76] Younger, Daniel H. Recognition and parsing of context-free languages in time n^3. *Information and Control 10*, 2 (1967), 189–208.

[77] Zhou, Yahan, Feild, Jacqueline, Wang, Rui, and Learned-Miller, Erik. Scene text segmentation via inverse rendering. In *International Conference on Document Analysis and Recognition* (2005).