UNIVERSITY OF CALIFORNIA, SAN DIEGO

Word spotting in the wild

A dissertation submitted in partial satisfaction of the
requirements for the degree
Doctor of Philosophy

in

Computer Science

by

Kai Wang

Committee in charge:

Professor Serge Belongie, Chair
Professor Virginia de Sa
Professor Charles Elkan
Professor David Kriegman
Professor Mohan Trivedi

2013

The dissertation of Kai Wang is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

_____

_____

Chair

University of California, San Diego

2013

DEDICATION

To my parents.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

First, I want to thank my family for all the love and support they have provided me to reach this point. I thank my parents for trusting in me when I told them I wanted to spend another five (or more) years in school to train myself as a researcher. Despite not having a clue of what that road entailed, they have been infinitely supportive in me reaching my goals – without their support, I hesitate to think where I would be at this point.

After my family, I want to give the earliest thanks to the people responsible for jump starting my research career: Donald Patterson and Matthai Philipose. Don was a graduate student at the University of Washington when he allowed me to assist on my first research project during my undergraduate studies. This experience gave me a glimpse into the previously unknown world of academic research and completely changed my view of what I wanted to pursue after college. I am in deep gratitude to him for taking a chance on a college sophomore during a critical time in the completion of his own thesis. I am also thankful that he introduced me to Matthai, my research mentor for the latter two years of college. I thank Matthai for all the time he invested advising and preparing me for my own research during graduate school.

I want thank my Ph.D. adviser, Serge Belongie, for his years of constant support. His guidance, patience, and open-mindedness have not only allowed me to investigate topics relevant to Computer Vision but also beyond, to those that cross over to different research areas. This level of freedom and support from an adviser is unique, and I'm incredibly grateful to have had him as mine.

I have many colleagues to thank from my time at UC San Diego. Two people critical to the completion of this dissertation are Boris Babenko and Piotr Dollar. Boris has been an excellent mentor, collaborator, and friend during my time as a graduate student. I thank Piotr for his key support and feedback during the critical early stages of this work.

I want to thank the students in the generation above me – Vincent Rabaud, Carolina Galleguillos, Andrew Rabinovich, and those preceeding them – for setting examples of the high quality research contributions that I would later strive for. I

also want to thank the other students in the UCSD Computer Vision Lab for their countless hours of discussion, collaboration, technical, and moral support: Steve Branson, Catherine Wah, Oscar Beijbom, Grant Van Horn, Sam Kwak, Tsung-Yi Lin, Phuc Nguyen, Hani Altwaijry, Mohammad Moghimi, Eric Christensen, Arturo Flores, and others. I thank Prof. David Kriegman, Prof. Charles Elkan, Prof. Virginia de Sa, and Prof. Mohan Trivedi for serving on my thesis committee.

In addition to those whose presence contributed to my proper dissertation work, I also want to thank those with whom I've worked with outside of Computer Vision. I thank Stefan Savage for being the first person to draw me into collaborations with Computer Security researchers. I'm grateful for his mentorship and the influence he's had in broadening my concept of research impact. I also thank Hovav Schacham and David Wagner for their fruitful collaborations in elections research. I consider my time spent outside of my dissertation work to have been extremely beneficial to my development as a well-rounded researcher.

Finally, I want to thank some friends I've met during my graduate years that have made this time the most enjoyable of my life: Andrea Lu, Hoang Nhan, Michael Lee, Chris Kanich, Marti Motoyama, Nakul Verma, Ryan Braud, Daniel Yee, Kaisen Lin, Alan Leung, Karyn Benson, Ming Wang, and Zach Tatlock.

Portions of this dissertation are either based on, or reprints of, papers that I have written with others in the past. Listed below are my contributions to each paper used in this document.

- Chapter 2 and Chapter 3 are based on "Word spotting in the wild" by K. Wang and S. Belongie [77]. The dissertation author was the primary investigator and author of this paper.

- Chapter 4 is based on "End-to-end Scene Text Recognition", K. Wang, B. Babenko, and S. Belongie [76]. The dissertation author was the primary investigator, contributed to algorithm development, implementation, and the writing of the paper.

- Chapter 5, in full, is a reprint of material that has been submitted for publication: "Scaling Up Scene Text Recognition via Data-driven Synthesis",

K. Wang, P. Nguyen, A. Bissacco, and S. Belongie. The dissertation author was the primary investigator, contributed to algorithm development, implementation, and was author of the paper.

VITA

2006	B. S. with College Honors in Computer Science, University of Washington, Seattle
2010	M. S. in Computer Science, University of California, San Diego
2013	Ph. D. in Computer Science, University of California, San Diego

PUBLICATIONS

D. Patterson, L. Liao, K. Gajos, M. Collier, N. Livik, K. Olson, S. Wang, D. Fox, and H. Kautz, "Opportunity Knocks: a system to provide cognitive assistance with transportation services", In *International Conference in Ubiquitous Computing (UBICOMP)*, 2004.

W. Pentney, H. Kautz, M. Philipose, A.-M. Popescu, and S. Wang, "Sensor-based understanding of daily life via large-scale use of common sense", In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2006.

S. Wang, W. Pentney, A. -M. Popescu, T. Choudhury, and M. Philipose, "Commonsense-based joint training of human activity recognizers", In *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007.

B. Laxton, K. Wang, and S. Savage, "Reconsidering physical key secrecy: teleduplication via optical decoding", In *ACM Computer and Communications Security (CCS)*, 2008.

P. Faymonville, K. Wang, J. Miller, and S. Belongie, "CAPTCHA-based image labeling on the soylent grid", In *Human Computation Workshop (HCOMP)*, 2010.

K. Wang and S. Belongie, "Word spotting in the wild", In *European Conference on Computer Vision (ECCV)*, 2010.

K. Wang, E. Rescorla, H. Shacham, and S. Belongie, "OpenScan: A fully transparent optical scan voting system", In *USENIX Electronic Voting Workshop (EVT/WOTE)*, 2010.

K. Wang, B. Babenko, and S. Belongie, "End-to-end Scene Text Recognition", In *International Conference on Computer Vision (ICCV)*, 2011.

K. Wang, E. Kim, N. Carlini, I. Motyashov, D. Nguyen, and D. Wagner, "Operator-assisted tabulation of optical scan ballots", In *USENIX Electronic Voting Workshop (EVT/WOTE)*, 2012.

E. Kim, N. Carlini, A. Chang, G. Yiu, K. Wang, and D. Wagner, "Improved Support for Machine-Assisted Ballot-Level Audits", In *USENIX Electronic Voting Workshop (EVT/WOTE)*, 2013.

K. Wang, P. Nguyen, A. Bissacco, and S. Belongie, "Scaling Up Scene Text Recognition via Data-driven Synthesis", Submitted to *International Conference on Computer Vision (ICCV)*, 2013.

P. Nguyen, K. Wang, S. Han, and S. Belongie, "Benchmarking Text Detection in Video", Submitted to *British Machine Vision Conference (BMVC)*, 2013.

ABSTRACT OF THE DISSERTATION

Word spotting in the wild

by

Kai Wang

Doctor of Philosophy in Computer Science

University of California, San Diego, 2013

Professor Serge Belongie, Chair

Text is a fundamental medium for visual communication. Methods to automatically read text, known as Optical Character Recognition (OCR), have a long history and by the 1960's found significant commercial application. Though in the past the focus of OCR technology has been in the domain of scanned books and documents, there are significant new sources of images resulting from the widespread use of digital cameras, mobile phones, vehicle-mounted cameras, wearable cameras, and more. This new flood of images continues to grow at a rapid pace and presents the OCR problem with new challenges and opportunities.

In contrast to the well-controlled scanned document environments of the past, these new data sources present a fundamental technical challenge to existing OCR engines due to the nature of being acquired in unconstrained environments.

These factors present us with a unique situation: OCR is historically a desirable technology, more images are being collected now than ever before, and due to the nature of this new wave of data, conventional techniques to solve the problem are unfit.

In a parallel literature, generic object recognition research has emerged to deal with the challenge of visual recognition in complex real world environments. While those advances pushed the field's ability to recognize a wide range of objects, very little had been done to translate this paradigm to recognition of one of the most important visual objects: text. In this dissertation, we aim to bridge the gap between the new learning-based techniques of object recognition to the problem of recognizing text in the wild. While the expectation of non-practitioners may have been that text recognition was a solved problem (due to its past success), we both break that perception and present a path for future progress in this challenging problem of considerable practical interest.

Chapter 1

Introduction

Text is a fundamental medium for visual communication. The ability to read and write is taught to children at a young age and is refined throughout adulthood. Text is a versatile channel: it is used to convey detailed ideas in long-form, e.g., through books, and simpler ideas in brief, e.g., signs. To say that text is ubiquitous in the man-made world is an understatement; reading text is how humans are able to navigate the man-made world.

Given the prominent position of text in everyday life, it is not surprising that the technology of automatic text reading, known as Optical Character Recognition (OCR), has a long history [63, 52, 29]. Patents for OCR-like systems to aide the blind were granted as early as the 1800's. Figure 1.1 shows an early mechanical template-matching OCR system. By the 1960's, OCR technology had found application for automated data processing in numerous industries including banking, mail, and government. Today, OCR is now regarded as a mature technology. There exist many commercial products (ABBYY[1], OmniPage[2], Tesseract[3], etc.) and there have been demonstrated successes in large scale book scanning and digitization (Internet Archive - over 3 Million since 2011[4], Google book project - over 20 Million since 2012[5]). OCR for Scanned documents has been a prominent

[1]http://finereader.abbyy.com/
[2]http://www.nuance.com/
[3]http://code.google.com/p/tesseract-ocr/
[4]http://blog.archive.org/2011/09/17/3-million-texts-for-free/
[5]http://chronicle.com/article/Google-Begins-to-Scale-Back/131109/

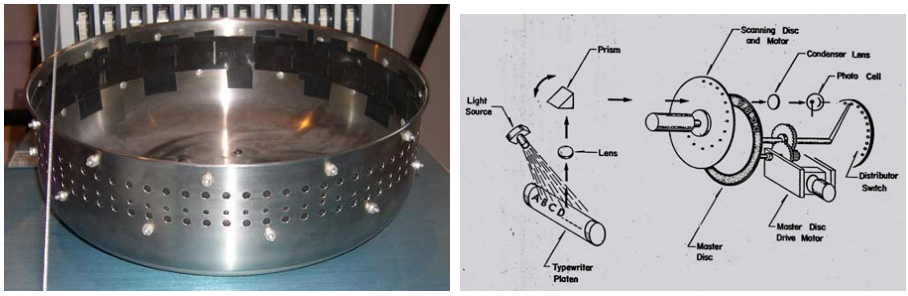


Figure 1.1: The left image a mechanical OCR system created by Jacob Rabinow[8]. The right image is a diagram of another system by NBS Ordnance Laboratory[9]. Both are early mechanical template matching-based systems.

success story of applied machine perception.

Though in the past the focus of OCR technology has been on scanned books and documents, there are significant new sources of images resulting from the widespread use of digital cameras, mobile phones, vehicle-mounted cameras, wearable cameras, and more. This new flood of images continues to grow at a rapid pace and the combined amount of visible text has the potential to surpass all previous scanned book sources; the rate at which new books are being produced and scanned will pale in comparison to the rate of images produced through these new means. The rise of ubiquitous image acquisition and storage has deeply impacted Computer Vision as a field and is one modality of the phenomenon commonly known as *Big Data*. In contrast to the well-controlled scanned document environments of the past, these new data sources present a fundamental technical challenge to existing OCR engines due to their nature of being acquired in unconstrained environments, or *in the wild*.

As others have observed, text acquired in unconstrained environments (often referred to as *scene text*) present many new challenges relative to scanned documents [12, 45, 81]. Figure 1 shows examples of text on a spectrum of difficulty levels. When we consider the extreme cases, the performance of OCR engines is known to be excellent when given scanned text and very poor on text that is highly obscured. Indeed, the fact that OCR has difficulty reading such text is the basis

[8]http://museum.nist.gov/panels/reading/figure1.htm
[9]http://museum.nist.gov/exhibits/rabinow/exhibits/ocr.html

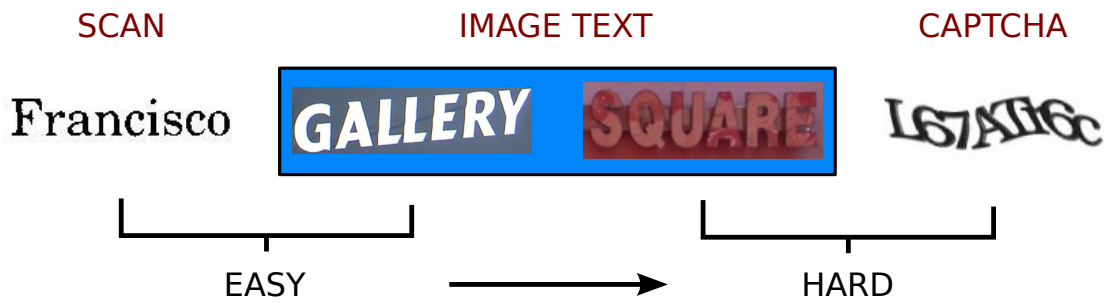


Figure 1.2: This figure shows text at a range of difficulties. In the extreme cases, the behavior of OCR is well established: there is an expectation that OCR will perform well on scanned text (far left) and perform very poorly on purposely obscured text (far right). In between these two extremes sits text found in the wild. Due to its unconstrained nature text can appear anywhere in the spectrum.

for systems that prevent automated software bots from abusing internet resources, which are known as CAPTCHAs [75]. Depending on the particular instance, text found in the wild can appear similar to a scanned page, similar to a CAPTCHA, or somewhere in-between. The challenges of scene text include increased variation in the appearance of the text itself, background clutter, variable image quality, diversity of font, shape, and texture, and general sparseness of text in images. The sum of this is that the accuracy of conventional OCR methods on scene text is far worse compared to the accuracy on scanned documents. These factors present us with a unique situation: OCR is historically a desirable technology, more images are being collected than ever before, and due to the unconstrained nature of this new wave of data the existing techniques are unfit.

In a parallel literature, generic object recognition research has emerged to meet the challenges of visual recognition in complex real world environments. This branch of computer vision has become one of the primary areas of focus in the field and has shown many impressive advances. A very small sample of these include face detection [61, 74] and recognition [72, 2], pedestrian detection [15, 24, 19], near-duplicate image matching [44, 1], and generic object recognition [22, 17, 31]. These advances demonstrate the great potential in applying ideas from statistical machine learning [28, 64] along with feature representations that have been well-engineered [3, 44, 15] or learned [14, 38, 40]. While these advances have pushed

our ability to recognize a wide range of objects, surprisingly little had been done to translate this paradigm to recognition of one of our most common visual objects, text. In this disseration, we aim to connect the new learning-based techniques of object recognition to the problem of scene text recognition.

In Chapter 2 we introduce the Street View Text (SVT) dataset, a scene text dataset collected from Google Streetview. We use this dataset throughout this work and have made it publicly available to the research community.

In Chapter 3 we examine the state of scene text recognition (circa 2007) and identify what we observe as the performance bottleneck. Existing work on scene text had framed the problem as pre-processing: given an image, the challenge is to detect the text-like image regions so they can be sent to a black-box OCR engine to be read. This paradigm assumes that the final step of text reading is easily handled by conventional OCR methods. In our work, we show that the performance bottleneck to this paradigm is the OCR engine itself and propose a new approach to text reading that replaces the black box engine using ideas from the object detection literature.

In Chapter 4 we investigate the challenges of constructing an end-to-end system for scene text. Previous work had focused on advances to isolated sub-components. In our work we do two things: (1) construct an end-to-end pipeline out of state-of-the-art components and (2) construct a much simpler end-to-end system by extending our work from Chapter 3. We show that we can achieve comparable accuracy using the latter, simpler method.

In Chapter 5, we take a step back and highlight the looming challenge of *scaling up* statistical machine learning-based systems to novel languages. While recent proposals for scene text recognition have shown promise, they face significant challenges for data annotation when considering one of OCR's killer applications: text recognition for translation. We propose to address this issue through a novel data-driven synthesis approach that leverages techniques from the Computer Graphics literature.

Finally, in Chapter 6, we end with a discussion about future directions for work in this area. While the expectation of non-practitioners may have been that

text recognition was a solved problem (due to its past commercial success), we hope to both break that perception and outline a path for continued progress in this problem of great practical interest.

Chapter 2

Street view text dataset (SVT)

Throughout this thesis we perform experiments on a new dataset that we collected and publicly released: the Street View Text (SVT) dataset. This data was harvested from Google Street View[1]. Image text in this data exhibits high variability and often has low resolution. Figure 2.1 shows examples from the SVT set and a histogram of word heights. In dealing with outdoor street level imagery, we note two characteristics. (1) Image text often comes from business signage and (2) business names are easily available through geographic business searches. These factors make the SVT set uniquely suited for word spotting in the wild: given a street view image, the goal is to identify words from nearby businesses. We used Amazon's Mechanical Turk[2] to harvest and label the images from Google Street View. To build the data set, we created several Human Intelligence Tasks (HITs) to be completed on Mechanical Turk. We refer to those that work on these HITs as *workers*.

Harvest images. Workers are assigned different cities and are requested to acquire 20 images that contain text from Google Street view. They were instructed to: (1) perform a *Search Nearby:** on their city, (2) examine the businesses in the search results, and (3) look at the associated street view for images containing text from the business name. If words are found, they compose the scene to minimize

[1] http://maps.google.com
[2] http://mturk.com

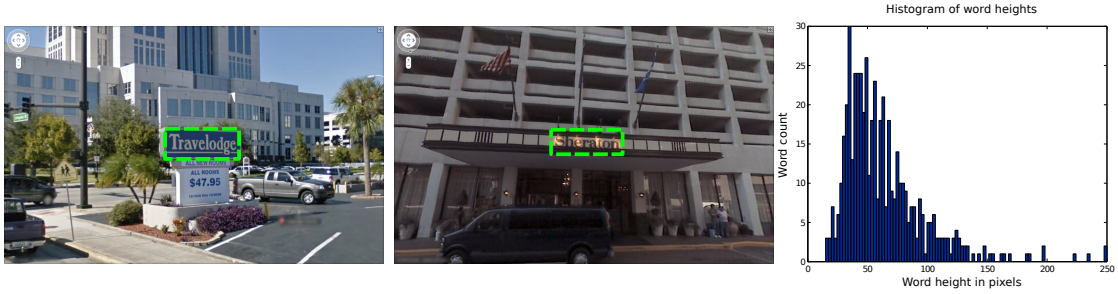


Figure 2.1: Examples from our Street View Text (SVT) data set and a histogram of word heights. The words appearing in this data set have high variability in appearance, suffer effects of cast shadows, and often have low resolution. The median height is 55 pixels.

skew, save a screen shot, and record the business name and address.

Image annotation. Workers are presented with an image and a list of candidate words to label with bounding boxes. This contrasts with the ICDAR Robust Reading data set in that we only label words associated with businesses. We used Alex Sorokin's Annotation Toolkit[3] to support bounding box image annotation. All images were labeled by three workers, and bounding boxes were accepted when at least two workers agreed with sufficient overlap.

For each image, we obtained a list of local business names using the *Search Nearby:** in Google Maps at the image's address. We stored the top 20 business results for each image, typically resulting in 50 unique words. To summarize, the SVT data set consists of images collected from Google Street View, where each image is annotated with bounding boxes around words from businesses around where the image was taken. The data set contains 350 total images (from 20 different cities) and 725 total labeled words. We split the data into a training set of 100 images and test set of 250 images, resulting in 211 and 514 words in the train and test sets. In correspondence with ICDAR, we divide our benchmark into SVT-SPOT (word locating), SVT-WORD (word recognition), and SVT-CHAR (character recognition). In this work, we address SVT-WORD. In total, the cost of acquiring the data from Mechanical Turk was under $500 USD. The data was

[3]http://vision.cs.uiuc.edu/annotation/

Figure 2.2: Selected images from our publicly available Street View Text dataset. This benchmark represents a practical word spotting application scenario.

acquired in the first quarter of 2010. The dataset is publicly available[4] and has been used by others in the literature. More examples from the dataset are shown in Figure 2.2.

Chapter 2 is based on "Word spotting in the wild" by K. Wang and S. Belongie [77]. The dissertation author was the primary investigator and author of this paper.

[4]http://vision.ucsd.edu/content/street-view-text

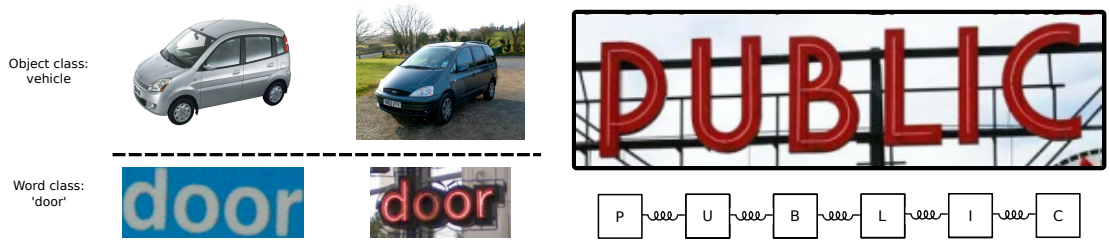
Chapter 3

Word-spotting in the wild

Finding words in images is a fundamental computer vision problem, and is especially challenging when dealing with images acquired in the wild. The field of Optical Character Recognition (OCR) has a long history and has emerged as one of the most successful practical applications of computer vision. However, text found in the wild can take on a great variety of appearances, and in many cases can prove difficult for conventional OCR techniques.

Our use of the phrase *in the wild* is analogous to Labeled Faces in the Wild (LFW) [33]: a data set constructed to study face recognition in unconstrained settings. Similar to text reading, face recognition under controlled settings is a well understood problem with numerous effective algorithms. However, as LFW shows, the variation in lighting, pose, imaging device, etc., introduce challenges for recognition systems. Much as that dataset acted as a catalyst for renewing progress in face recognition, an important goal of this work is to spur interest in the problem of spotting words in the wild.

The word spotting problem contrasts with general text reading in that the goal is to identify specific words. Ideally, there would be no distinction between the standard text reading and word spotting; spotting words would simply amount to filtering the output from OCR engines to catch the words of interest. However, due to the challenges presented by text found in the wild, we approach the word spotting problem directly, where we are presented with an image and a lexicon of words to spot. We evaluate the performance of conventional OCR engines and



(a) Word recognition. (b) Pictorial structure for words.

Figure 3.1: The left figure (a) shows our analogy to the generic object classification problem. In both cases, individual instances of the same class can take on vastly different appearances. The right figure (b) is an illustration of modeling the word 'PUBLIC' using a pictorial structure.

also present a new method rooted in ideas from object recognition. In our new approach, we treat each word in a lexicon as an object category and perform word category recognition. Figure 3.1(a) shows an analogy to generic object recognition: just as instances of the object category *vehicle* can look vastly different from image to image, the word 'door' can also take on a variety of appearances depending on the font, lighting, and pose in a scene. In this formulation, we can leverage techniques that have been designed to be robust for recognizing generic categories and apply them to word recognition.

Our contributions are the following. (1) We introduce the Street View Text data set: an outdoor image text data set annotated with a list of local business names per image. (2) We benchmark conventional OCR engines on our new data set and the existing ICDAR Robust Reading image text database [45]. (3) We present a new word spotting approach that imports techniques from generic object recognition and significantly outperforms conventional OCR based methods.

3.1 Related work

3.1.1 Scanned document OCR

The topic of OCR has been well studied [53, 52] and existing commercial products are in widespread use. One example is Google Book Search[1], which has scanned more than 20 million volumes[2] making them accessible for full text searches. Another example is the Kurzweil National Federation of the Blind (KNFB) reader[3]. The KNFB reader is an OCR engine that runs on a mobile phone and allows a person who is visually impaired to read printed text from an image taken by the camera. The key to high performance for the KNFB reader is having a high quality camera built into the mobile phone and a feedback loop to assist the user in taking pictures in an ideal setting, thereby minimizing the effects of motion blur, lighting, and skew.

A critical step for OCR accuracy is image binarization for character segmentation. The survey of [10] identifies incorrect segmentation as one of the major contributors to errors in using conventional OCR on scanned documents. Previous work on classifying hand written digits from the MNIST data set has shown that when the correct segmentation is provided, it is possible to achieve recognition rates nearing that of humans[4]. The task of separating out individual characters was also identified in [11] as one of the distinguishing features of CAPTCHAs being difficult for OCR while remaining manageable for humans. Character segmentation is a significant challenge that conventional OCR engines face when dealing with words in the wild.

3.1.2 Object recognition

Existing work on image text typically breaks the process into two subtasks: text detection and word recognition. Advances have been made in detecting image text using an AdaBoost-based approach [12]. In that work, detected text regions

[1] http://books.google.com/
[2] http://chronicle.com/article/Google-Begins-to-Scale-Back/131109/
[3] http://www.knfbreader.com/
[4] http://yann.lecun.com/exdb/mnist/index.html

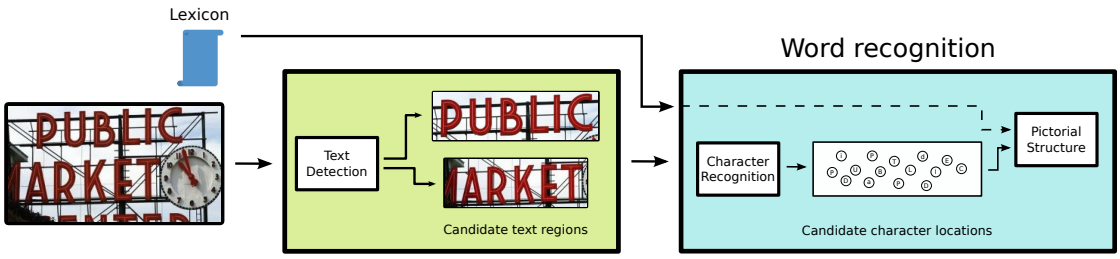


Figure 3.2: **Word spotting overview.** This is an illustration of a word spotting system with two steps: text detection [12] and word recognition. In this work, we focus on the latter problem where the input is an image region and a lexicon of words. In our Street View Text data set, the lexicon was created out of local business searches around where the image was acquired. We run character detectors to discover possible character locations and then score words in our lexicon by modeling them as pictorial structures.

are sent to a conventional OCR engine to be decoded. Others have explored the problem of improving recognition rates by combining outputs of several different OCR engines to get a more robust reading [73]. In the work of [81] the authors assumed character bounding boxes were provided, and proposed a model that incorporated character appearance similarity and dissimilarity within a word.

The works that are most similar to ours are that of [51] and [60]. In [51], the authors investigated methods of breaking visual CAPTCHAs. In their CAPTCHA experiments, the problem was also one of word spotting: categorize the image of a word as one of a list of possible keywords. Our new approach highlights the similarities between words in the wild and with visual CAPTCHAs. In [60], the authors performed word spotting in scanned handwritten historical documents. To perform word spotting, they clustered words together by appearance, manually provided labels to clusters, and propagated the labels to the cluster members, allowing them to create a word index to browse a large corpus.

In our methods, we draw on work done using part-based methods for object recognition; in particular, the modeling of objects using pictorial structures [27, 26]. We also build on the work of [16], who studied the use of various features and classification methods to classify individually cropped characters.

3.2 Motivating applications

Accurate word-spotting plays an important role in systems for image retrieval and navigation. Research in Content Based Image Retrieval (CBIR) [68] has explored different forms of querying large image collections, including queries by keyword and image example. Integrating a word spotting component enables queries by word occurrence, returning images in which the specified words appear. The work of [60] describes a system that allows for retrieval of historical documents based on handwritten word spotting.

Word spotting is an essential component of a vision based navigation system. In our case, this arises in the form of developing assistive technologies for the blind. Two broad goals of the project are to develop a computer vision system that can benefit the blind and visually impaired communities, and to study the challenges of performing vision-based navigation in real world environments. For navigation, it is important to be able to spot specific keywords in order to guide a blind user. Detecting keywords on signage can be used, for example, to direct a user to the correct aisle in a supermarket while detecting words from a shopping list can be used to locate specific products.

3.3 Word recognition

In our approach, we first perform character detection for every letter in an alphabet and evaluate the configuration scores for the words in our lexicon to find the most suitable one. Our method is designed to be used in conjunction with a text detector. In our description, we use the term 'input image' to mean the cropped out image region around a word provided by a text detector. Figure 3.2 shows a diagram of this pipeline.

3.3.1 Character recognition

Character recognition in images was recently studied in [16]. In their work, they benchmarked different features and classification algorithms for recognizing

cropped characters. In our experiments, we test our character detector using the same data and methodology, and list accuracies next to those from their work. For our character detector, we use Histograms of Oriented Gradient (HOG) [15] features with a nearest neighbor classifier.

Character classification: To compare two images of cropped characters, we first resize them to take on the same height and aspect ratio, then densely calculate their HOG features. Each character is now represented as an array of dimension $m \times n \times d$ where m and n are the number of rows and columns after spatial binning, and d is the number of dimensions in each histogram. We measure the similarity between characters by performing Normalized Cross Correlation (NCC) between each dimension and averaging the scores. Since the characters were resized to be the same dimension, the result is a single number. This is the value we use for nearest neighbor classification.

Character detection: To perform character detection over an input image we take all the training examples for a particular character class, resize them to the height of the input image (while maintaining aspect ratio), and compare the character's HOG features to those of the input. Between each training example and the input, we again calculate the NCC between each HOG dimension and combine them again by averaging. The result will be a list of scores measuring the similarity of a template to each location in the input image. This is done for all the training examples of a class, and the results are combined together per class by taking the max at each location. We perform non-maximum suppression to discover peaks and consider those as candidate character locations.

This is done for every character class to create a list of character locations with discrete spatial positions. Next, we use this list of detections to evaluate the configuration of strings in our lexicon to the input image.

3.3.2 Word configuration

After performing character detection, we consider each word in our lexicon and measure its character configuration within the input image. We represent a word using a pictorial structure [27, 26]. A pictorial structure is a mass-spring model that takes into account costs of matching individual parts to image locations and their relative placement. A word is naturally broken down into character 'parts' and takes on a simple chain structure. Figure 3.1(b) shows an example of a string as a pictorial structure.

We formulate the problem of optimal character placement in an image of text in the following way. Let $G = (V, E)$ be an undirected graph representing a string S. The vertices $V = \{v_1, ..., v_n\}$ correspond to characters in S where n is the length of S. Edges $(v_i, v_j) \in E$ connect letters that are adjacent in S. This creates a conceptual spring between pairs of letters. We use the terms parent and child to refer to the left and right nodes in a pair of adjacent characters. Let $L = (l_1, ..., l_n)$ represent a particular configuration of characters in an image where l_i is the spatial $[x, y]^\top$ coordinate placement of character v_i.

We measure cost $m_i(l_i)$ as one minus the similarity score of a character detection calculated in the previous step. To calculate the deformation cost $d_{i,j}(l_i, l_j)$, we use our domain knowledge of character layout. We expect a child character to appear one character width away from its parent. Let the expressions $w(l_i)$ and $h(l_i)$ represent the width and height of a character detection at location l_i. Let $l_i^* = l_i + [w(l_i), 0]^\top$ represent the expected position of a child of l_i. We specify a covariance matrix that normalizes the deformation cost to the dimensions of the parent character: $\Sigma = \begin{bmatrix} w(l_i) & 0 \\ 0 & h(l_i) \end{bmatrix}$. Our deformation cost is calculated as: $d_{i,j}(l_i, l_j) = \sqrt{(l_i^* - l_j)^\top \Sigma^{-1} (l_i^* - l_j)}$. The objective function for our optimal character configuration for a string S is computed as:

$$L^* = \underset{L}{\operatorname{argmin}} \left(\theta \sum_{i=1}^{n} m_i(l_i) \; + (1 - \theta) \sum_{(v_i, v_j) \in E} d_{ij}(l_i, l_j) \right) \tag{3.1}$$

The parameter θ controls the balance between part match cost and defor-

mation cost. The result is a configuration L^* that represents the optimal character placement for reading S in an image. Solving for L^* can be done efficiently using dynamic programming as described in [26]. We refer to this configuration cost as $D_c(L)$.

The score generated by L^* can take into account a local measure of coherence between a string and an image, but is uninformed of higher order and global configuration costs. To supplement the score configuration score, we also incorporate other domain knowledge-influenced measures into our final match score.

- **Horizontal span**: Given our input is an image of a cropped word from a character detector, we assume that a suitable string is one whose characters span most of the input image. We calculate this as the horizontal range of the character configurations divided by the width of the input image and call it $D_s(L)$.

- **Character distribution**: Character spacing within a single string should be consistent, and we factor this into the final score by measuring the standard deviation of the spacing between every pair of adjacent characters in the string, which we refer to as $D_d(L)$.

The final cost D is a weighted sum of these terms: $D(L) = \alpha_1 D_c(L) + \alpha_2 D_s(L) + \alpha_3 D_d(L)$ where $\alpha_1 + \alpha_2 + \alpha_3 = 1$. Through validation on our training data, we determined reasonable parameters to be $\theta = .9$, $\alpha_1 = .5$, $\alpha_2 = .4$, and $\alpha_3 = .1$. These parameters were used in both the ICDAR and SVT benchmarks.

3.4 Experiments

We evaluate the performance of our character recognizer in isolation and our word recognition system as a whole on existing public image text data sets. The data sets we use are from the ICDAR 2003 Robust Reading challenge [45], Chars74K [16], and our SVT data set. In our experiments, we compare to results attained using conventional OCR systems ABBYY FineReader 9.0 and Tesseract

OCR[5], referred to as ABBYY and TESS. In using the OCR engines, we experimented with pre-thresholding the images using the technique from [12], where they performed locally adaptive thresholding with a heuristic for a parameter sweep at each pixel. However, we found that deferring the thresholding task to the individual OCR engines resulted in better accuracy, and so we only report those results. In all our experiments, we resized images to take on a height of 50 pixels and used 4×4 pixel cells with 10 orientation bins for the HOG features.

3.4.1 Character classification results

We benchmarked our character classifier on the Chars74K-5, Chars74K-15, and ICDAR03-CH data sets. The Chars74K-5 and Chars74K-15 contained 5 and 15 training instances per class, respectively, while the test sets included the same 15 instances of each character class. The ICDAR03-CH data set is the character classification subproblem from the ICDAR Robust Reading data set. In all data sets, the characters included upper and lowercase letters, and digits 0 through 9; in total 62 symbols. Our evaluation methodology mirrored that of [16] and our results are reported next to theirs in Table 3.4.1.

In Table 3.4.1, our classifier is labeled as HOG+NN and is displayed in bold in the first row. The next three rows are reproduced from [16]. The first is Multiple Kernel Learning (MKL), which is a combination of a number of features described in [16]. In that work, results for MKL were only reported on the Chars74K-15, accounting for the dashes in the other two columns. The next two rows show performance using features from Geometric Blur (GB) [5] and Shape Context (SC) [3], and classified using Nearest-Neighbor (NN) as reported in [16]. The methods listed were the ones that performed best from [16].

Our HOG+NN classifier outperforms those tested in [16] in all three benchmarks, and more significantly on the Chars74K-5 and ICDAR03-CH. However, we note that any suitable classification technique that can produce a list of discrete character detections can be substituted into the word recognition pipeline.

[5]http://code.google.com/p/tesseract-ocr/

Table 3.1: Results for character classification. Our HOG+NN approach performs best on the three benchmarks, demonstrating the benefit of using HOG features for character classification.

Feature	Chars74K-5	Chars74K-15	ICDAR03-CH
HOG+NN	**45.33 ± .99**	**57.5**	**51.5**
MKL	-	55.26	-
GB+NN	36.9 ± 1.0	47.09	27.81
SC+NN	26.1 ± 1.6	34.41	18.32
ABBYY	18.7	18.7	21.2
TESS	17.3	17.3	17.4

3.4.2 Word recognition results

We ran experiments on the ICDAR03-WORD and SVT-WORD data sets: the word recognition benchmarks of both data sets. Unlike SVT-WORD, ICDAR03-WORD is not explicitly structured for word spotting. Therefore, in our experiments, we construct lexicons synthetically using the ground truth. In both benchmarks, we use the exact same parameter settings and character training data, from ICDAR. In our comparisons to ABBYY and TESS, we provided the lexicons in the form of custom dictionaries and corrected OCR output to be the word with the smallest edit-distance in the lexicon.

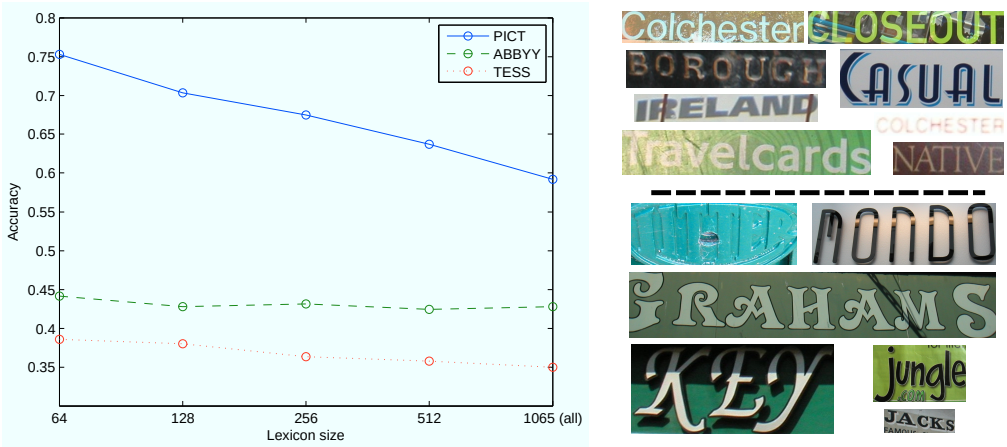
ICDAR Robust Reading: Word Recognition.

In this experiment, we compare our approach, labeled as PICT, to the OCR engines ABBYY and TESS on ICDAR03-WORD. For simplicity, we filtered out words containing symbols other than letters and numbers, leaving 1,065 testing images. To formulate this problem as word spotting, we constructed tests of various sizes where we built synthetic lexicons out of the ground truth words for a particular test run. We divided the test set according to Table 3.4.2.

Table 3.2: Number of trials for each lexicon size.

Lexicon size	64	128	256	512	1065
Trials	16	8	4	2	1

For each size k, we took all our testing data, randomized the order, and



(a) (b)

Figure 3.3: Subfigure (a) shows the performance of our method PICT, and OCR engines Abbyy FineReader 9.0 (ABBYY) and Tesseract OCR (TESS) on the ICDAR word benchmark. In this experiment, synthetic lexicons were created out of the ground truth in each run. We provided custom dictionaries to ABBYY and TESS and corrected their output to the nearest lexicon word by edit-distance. The y-axis marks word recognition accuracy and the x-axis marks the lexicon size. The full test size is 1,065 word images. In subfigure (b), the examples above the line are those that PICT only recognizes correctly, and the examples below are when all methods fail.

tested on contiguous chunks of size k until all of the test data was used. For example, when $k = 64$, we randomized the order of the test data and sampled sections of 64 images at a time (16 sections). We evaluated the three systems on each group of images where the lexicon consisted of words only from that set.

Figure 3.3 shows the word recognition results. The results are averaged over all the trials for each lexicon size. In our results, we see that at a lexicon size of 1,065, PICT significantly outperforms both OCR engines by over 15% and has more than 30% improvement when limiting the lexicon size to 64.

Street View Text: Word Recognition.

In this benchmark, we tested ABBYY, TESS, and PICT on our Street View Text benchmark. On the SVT benchmark, PICT used the exact same training data

and parameters as used in ICDAR03-WORD. No character training data from SVT was used. The test size was 514 word images and each image had an associated list of businesses to categorize from. The accuracies for TESS, ABBYY, and PICT were 31.5%, 47.7%, and 59.0% respectively. Our PICT approach shows significant improvement over the OCR engines.

Implementation Details: The system was implemented in C++ using the OpenCV framework. Average processing time to run PICT was under six seconds on an Intel Core 2 processor.

3.5 Error analysis

In an attempt to better understand the complexity of image text as it relates to the performance of conventional OCR, we introduce a simple diagnostic to gauge image difficulty. In both ICDAR and SVT data sets, there are examples of words that span the difficulty spectrum: some are well-suited for OCR while others present a challenge approaching that of a CAPTCHA. In our analysis, we separate the data into two groups, 'EASY' and 'HARD', based on a simple heuristic that is independent of either OCR engine. The intuition behind our heuristic is that easy examples are likely to have continuous edges around each character and few spurious edges from the background. We ran a Canny edge detector [8] on the the data and separated the images by calculating the number of continuous edges divided by the image's aspect ratio. This value represents approximately the number of line segments in a space typically occupied by one to two characters. We placed images with values between 1 and 3.5 into the EASY category, and all others into the HARD category; see Figure 3.5 for examples of each category. In the EASY category, we can see that the edges around characters are often reliably traced, whereas in the HARD category, many edges are picked up from the background and shadows. Table 3.5 shows the breakdown of results after separating the data.

While this is not meant to be a definitive method for categorizing the data – indeed, there could be a more sophisticated heuristic to accurately identify text

Figure 3.4: In our analysis, we use a simple and intuitive heuristic based on edge detection to group images into EASY and HARD. The EASY examples are typically those whose characters are well outlined, and the HARD ones typically contain more broken characters and edges from the background and shadows. This is a coarse estimate of those images that are more CAPTCHA-like.

Table 3.3: This table shows the breakdown of results after applying our image diagnostic to categorize images as EASY and HARD. The proportion of the easy data for ICDAR and SVT data sets were 40% and 33% respectively.

	ICDAR (1065)			SVT		
METHOD	ALL	EASY	HARD	ALL	EASY	HARD
TESS	35.0	41.7	30.5	31.5	43.2	25.8
ABBYY	42.8	56.9	33.4	47.7	62.7	40.3
PICT	**59.2**	**65.0**	**55.3**	**59.0**	**63.9**	**56.8**

Table 3.4: This table shows the breakdown of how often the two OCR engines determine the that image *does not contain readable text*. This situation constitutes a large portion of the overall errors in each engine.

	ICDAR (1065)			SVT		
METHOD	ALL	EASY	HARD	ALL	EASY	HARD
TESS	33.8	32.6	34.6	46.5	42.0	48.4
ABBYY	45.2	34.5	52.4	44.6	29.6	51.9

that can be read at scanned document levels – it is a simple and intuitive measure of image text complexity and provides a coarse estimate of how difficult an image of text is to segment. We can see all the methods perform significantly better on the EASY subset and the OCR methods suffer greater reductions on the HARD subset.

One reason for the significant performance drop of the OCR methods is

that proper character segmentation is likely more challenging on the HARD set. The improvement in performance of the PICT model can be attributed to the fact that it avoids character segmentation, instead relying on character detection in a sliding window fashion. These detections are collected using a part based word model designed that is robust to small errors. Figure 3.5 shows examples of these situations. In the images for 'MARLBORO' and 'STUFF', they are complex in appearance and suffer from cast shadows; as a result, accurate segmentation is extremely challenging. However, the detection approach focuses on finding local maxima in the response from the character classifier rather than segmentation. In the 'Marriott' example, a single mis-detected part, the letter 'r', still results in word configuration score that allows it to be categorized correctly. While it is the case that minor errors in character classification are corrected using edit-distance for the OCR engines, we see from Table 3.5 that a common failure case is when the OCR engine returns no reading at all, suggesting that significant errors in segmentation can result in irrecoverable errors for OCR. The performance of PICT on the HARD subsets is what sets it apart from the OCR methods.

3.6 Discussion

In this chapter we explored the problem of word spotting and evaluated different methods to solve the problem. We have shown that approaching word spotting as a form of object recognition has the benefits of avoiding character segmentation – a common source of OCR errors – and is robust to small errors in character detection. When dealing with words in the wild, it is often the case that accurate segmentation is unattainable, and especially in these cases, our detection based approach shows significant improvement. Clearly, there is still room for improvement in performance, but we have shown that framing the word spotting problem as generic object recognition is a promising new direction.

Chapter 3 is based on "Word spotting in the wild" by K. Wang and S. Belongie [77]. The disseratation author was the primary investigator and author of this paper.

Figure 3.5: This figure shows some advantages of using part based object detection. In the images of 'MARLBORO' and 'STUFF', character segmentation is extremely challenging because of the cast shadows and letter designs. Using the character detection approach allows us to avoid explicit segmentation and instead relies on local peaks from our character detector. The configuration of the word 'Marriott' shows how a pictorial structure model is tolerant of minor errors in the part detections. We can see that even though the first 'r' is not in the correct position, the total configuration cost for the word is better than that of the others associated with that image.

Chapter 4

End-to-end Scene Text Recognition

The ICDAR Robust Reading challenge [45] was the first public dataset collected to highlight the problem of detecting and recognizing scene text. In this benchmark, the organizers identified four subproblems: (1) cropped character classification, (2) full image text detection, (3) cropped word recognition, and (4) full image word recognition. The work of [16] addressed the cropped character classification problem (1) and showed the relative effectiveness of using generic object recognition methods versus off-the-shelf OCR. The works of [12, 21] introduced methods for text detection (2). The cropped word recognition problem (3) has also recently received attention by [81] and in our previous work [77]. While progress has been made on the isolated components, there has been very little work on the full image word recognition problem (4); the only other work we are aware of that addresses the problem is [55].

In this chapter, our contributions are two-fold: (1) We evaluate the word detection and recognition performance of the two-step approach consisting of a state-of-the-art text detector and a leading OCR engine. (2) We construct a system rooted in modern object recognition techniques by extending our work from Chapter 3. We show that our object recognition-based pipelines perform significantly better than one using conventional OCR. We also show that, surprisingly, an object recognition-based pipeline achieves competitive performance *without* the

Figure 4.1: The problem we address in this chapter is that of word detection and recognition. Input consists of an image and a list of words (e.g., in the above example the list contains around 50 total words, and include `'TRIPLE'` and `'DOOR'`). The output is a set of bounding boxes labeled with words.

need for an explicit text detection step. This result provides a significant simplification of the end-to-end pipeline and blurs the line between word recognition and the more common object recognition problems studied in computer vision.

4.1 Overview of Full Image Word Recognition

We discuss each step in detail. Figure 4.2 shows an overview of our approach.

4.1.1 Character detection

The first step in our pipeline is to detect potential locations of characters in an image. We perform multi-scale character detection via sliding window classification; this approach has been extremely successful in face [74] and pedestrian [15]

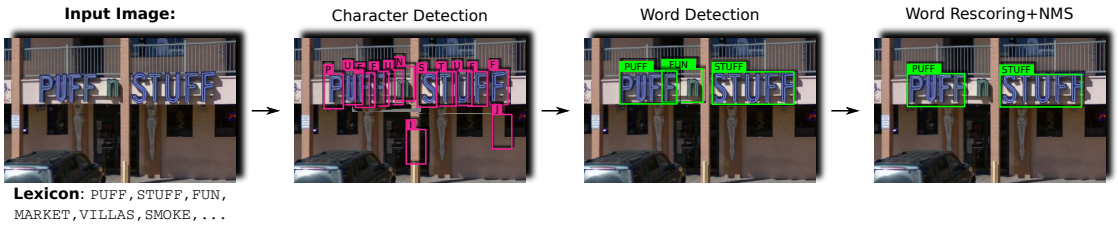


Figure 4.2: An overview of our word detection and recognition pipeline. Starting with an input image and a lexicon, we perform multi-scale character detection. The words `'PUFF'` and `'STUFF'` appear in the image while the other words in the lexicon can be thought of as "distractors". Next we perform word detection using a Pictorial Structures framework, treating the characters as "parts" of a word. Finally, we re-score detected words using features based on their global layout, and perform non-maximal suppression (NMS) over words.

detection. However, since our problem requires detection of a large number of categories (62 characters), we must be mindful in our choice of classifier. In this respect Random Ferns [58, 7, 67] are an appealing choice as they are naturally multi-class and efficient both to train and test. In the following sections we will review the basics of Random Ferns and how we use them for detection, and discuss the details of our training data.

Character Detection with Random Ferns For each location ℓ in an image we will extract some feature vector x, and compute a score, $u(\ell, c)$, that tells us the likelihood of character c being in this location, as opposed to the background c_{bg}:

$$\begin{aligned} u(\ell, c) &= \log\left(\frac{\mathbf{p}(c|x)}{\mathbf{p}(c_{bg}|x)}\right) \\ &= \log\Big(\mathbf{p}(x|c)\Big) - \log\Big(\mathbf{p}(x|c_{bg})\Big) + \log\left(\frac{\mathbf{p}(c)}{\mathbf{p}(c_{bg})}\right). \end{aligned} \tag{4.1}$$

We will assume a uniform prior over categories which means the last term in the second line becomes a constant and can be ignored for our purposes. For the simplicity of the model we will assume that our feature space consists of N binary features (i.e., $x \in \{0,1\}^N$). Notice that storing a representation of the joint probability $\mathbf{p}(x|c)$ would require a table of size 2^N. A common simplification of

this model is to assume that all features are conditionally independent (i.e., the Naive Bayes model [6]):

$$\mathbf{p}(x|c) = \prod_{i=1}^{N} \mathbf{p}(x[i]|c).$$

Random Ferns, introduced in [58], can be interpreted as a compromise between the above oversimplification and a fully joint probability table: the features are partitioned into M groups, $x_1, \ldots, x_M$, of size $S = N/M$, and an independence assumption is made for these groups rather than individual features. This results in the following formula for the conditional probability:

$$\mathbf{p}(x|c) = \prod_{i=1}^{M} \mathbf{p}(x_i|c).$$

Notice that the conditional probability for each group, or *Fern*, x_i can be computed using a table of size $2^S \times M$ per category. At run time we must simply compute our binary features, look up the corresponding fern probabilities in stored tables, and multiply the results (or take a log and add). In our present implementation the features consist of applying randomly chosen thresholds on randomly chosen entries in a HOG descriptor [15] computed at the window location. This framework scales well with the number of categories, and has been incorporated in real-time systems for keypoint matching [58] and object recognition [67].

The final step of character detection is to perform non-maximal suppression (NMS). We do this separately for each character using a simple greedy heuristic (similar to what is described in [25]): we iterate over all windows in the image in descending order of their score, and if the location has not yet been suppressed, we suppress all of its neighbors (i.e., windows that have an overlap over some threshold).

The character detection step can be applied directly to the image or after a generic text detector has identified regions of interest.

Equipped with this simple but robust classification module we must now face the task of collecting enough training data to achieve good detection performance.

Figure 4.3: Top: synthetic data generated by placing a small random character (with 1 of 40 different fonts) in the center of a 48×48 pixel patch and two neighboring characters, adding Gaussian noise and a random affine deformation. Bottom: "real" characters from the ICDAR dataset. To train our character detector we generated 1000 images for each character.

Synthetic Training Data Collecting a sufficiently large dataset is a typical burden of using a supervised learning method. However, some domains have enjoyed success by training and/or evaluating on synthetically generated images: fingerprints [9], fluorescent microscopy images [42], keypoint deformations [58], and even pedestrians [30, 47]. Beyond the obvious advantage of having limitless amounts of data, synthesizing training images allows for precise control over alignment of bounding boxes – an important property that is often critical to learning a good classifier.

We synthesized about 1000 images per character using 40 fonts. For each image we add some amount of Gaussian noise, and apply a random affine deformation. Examples of our synthesized examples are shown in Figure 4.3, along with examples of "real" characters from the ICDAR dataset.

4.1.2 Pictorial Structures

To detect words in the image, we use the Pictorial Structures (PS) [26] formulation that takes the locations and scores of detected characters as input and finds an optimal configuration of a particular word. More formally, let $w = (c_1, c_2, ..., c_n)$ be some word with n characters from our lexicon, $Ł_i$ be the set of detected locations for the i^{th} character in w, and $u(\ell_i, c_i)$ be the score of a particular detection at $\ell_i \in Ł_i$, computed with Eqn. (4.1). We seek to find a configuration $L^* = (\ell_1^*, \ldots, \ell_n^*)$ by optimizing the following objective function:

$$L^* = \underset{\forall i, \ell_i \in \mathcal{L}_i}{\operatorname{argmin}} \left(\sum_{i=1}^{n} -u(\ell_i, c_i) + \sum_{i=1}^{n-1} d(\ell_i, \ell_{i+1}) \right), \tag{4.2}$$

where $d(l_i, l_j)$ is a pairwise cost that incorporates spatial layout and scale similarity between two neighboring characters[1]. In practice, a tradeoff parameter is used to balance the contributions of the two terms.

The above objective can be optimized efficiently using dynamic programming as follows. Let $D(l_i)$ be the cost of the optimal placement of characters $i+1$ to n with the location of the i^{th} character fixed at ℓ_i:

$$D(l_i) = -u(l_i, c_i) + \min_{l_{i+1} \in \mathcal{L}_{i+1}} d(l_i, l_{i+1}) + D\left(l_{i+1}\right). \tag{4.3}$$

Notice that total cost of the optimal configuration L^* is $\min_{\ell_1 \in Ł_1} D(\ell_1)$. Due to the recursive nature of $D(\cdot)$ we can find the optimal configuration by first precomputing $D(\ell_n) = -u(\ell_n, c_n)$ for each $\ell_n \in Ł_n$ and then working backwards toward the first letter of the word. For improved efficiency we also include a pruning rule when performing the minimization in Eqn. (4.3) by only considering locations of ℓ_{i+1} that are sufficiently spatially close to ℓ_i.

Pictorial Structures with a Lexicon. The dynamic programming procedure for configuring a single word can be extended to finding configurations of multiple words. Consider for example the scenario where the lexicon contains the two words {`ICCV`,`ECCV`}. The value of $D(\ell_2)$ is the same for both words because they

[1]The deformation cost measures the deviation of a child character to the expected location relative to its parent, which is specified as one character-width away, as in [77].

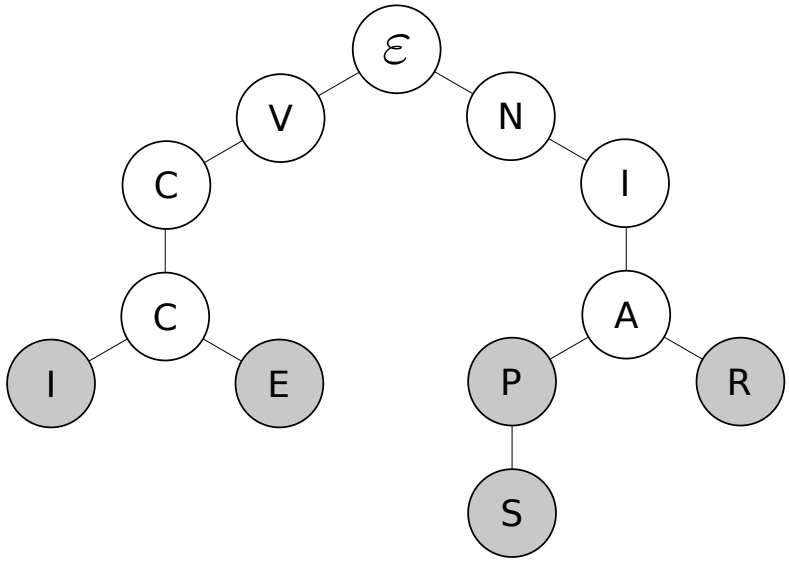


Figure 4.4: An example of a trie data structure built for a lexicon containing the words {‘ICCV’,‘ECCV’, ‘SPAIN’,‘PAIN’,‘RAIN’}. Every node in the trie that is the beginning of a word is shaded in gray. To efficiently perform Pictorial Structures for all words in the lexicon, we traverse the trie, storing intermediate configuration solutions at every node. When a shaded node is reached, we return the optimal configurations for the corresponding word.

share the suffix ‘CCV’, and can therefore be computed once and used for configuring both words. We leverage this by building a trie structure out of the lexicon, with all the words reversed. Figure 4.4 shows an example of a trie for five words, with the shaded nodes marking the beginning of words from the lexicon (the rest of string is formed by tracing back to the root of the tree). To find configurations of the lexicon words in the image, we traverse the trie and store intermediate solutions at every node. When we reach nodes labeled as words (the grayed out nodes), we return the optimal configurations as word candidates. In practice, since an image may contain more than one instance of each word, we return a few of the top configurations for each word. In the worst case, when no two words in the lexicon share a common suffix, this method is equivalent to performing the optimization for each word separately. In practice, however, performing the optimization jointly is typically more efficient. In the remainder of the chapter we will refer to the above procedure as “PLEX”.

4.1.3 Word Re-scoring and NMS

The final step in our pipeline is to perform non-maximal suppression over all detected words. Unfortunately, there are a couple problems with the scores returned by PLEX. First, these scores are not comparable for words of different lengths. The more important issue, however, is that the Pictorial Structures objective function captures only pairwise relationships and ignores global features of the configuration. While this allows for an efficient dynamic programming solution to finding good configurations, we would like to capture some global information in our final step. We therefore re-score each word returned by PLEX in the following manner. We compute a number of features given a word and its configuration:

- The configuration score from PLEX (i.e., cost of L^*)
- Mean, median, minimum and standard deviation of character scores (i.e., $u(\ell_i)$)
- Standard deviation of horizontal and vertical gaps between consecutive characters
- Number of characters in a word.

These features are fed into an SVM classifier, the output of which becomes the new score for the word. To train the classifier we simply run our system on the entire training dataset, label each returned word positive if it matches the ground truth and negative otherwise, and feed these labels and computed features into a standard SVM package[2]. Parameters of the SVM are set using cross validation on the training data. Once the words receive their new scores, we perform non-maximal suppression in the same manner as we described for character detection in Section 4.1.1.

The full system, implemented in Matlab, takes roughly 15 seconds on average to run on a 800×1200 resolution image with lexicon size of around 50 words. We expect the runtime to be much lower with more careful engineering (e.g., [67] showed real-time performance for Ferns).

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

4.2 Experiments

In this section we present a detailed evaluation of our PLEX pipeline, as well as a two-step pipeline consisting of Stroke Width Transform [21] (a state-of-the-art text detector) and ABBYY FineReader[3] (a leading commercial OCR engine). We used data from the Chars74K[4] dataset, introduced in [16] for cropped character classification; the ICDAR Robust Reading Competition dataset [45], discussed in Section 4; and Street View Text (SVT), a full image lexicon-driven scene text dataset introduced in [77][5].

4.2.1 Character Classification and Detection

We begin with an evaluation of character classification on the Chars74K-15 (where there are 15 training examples per character class) and the ICDAR-CH (character classification sub-benchmark). We measure performance of Ferns trained on synthetic data and Ferns trained on the real images from the respective datasets (labeled 'NATIVE'). We also compare to previously published results of HOG+NN and ABBYY [77], as well as MKL [16].

Table 4.2.1 lists the character classification results on the two datasets. We see that NATIVE+FERNS outperforms other methods on the ICDAR-CH dataset. However, its performance on the Chars74K-15 benchmark is below that of previous results using HOG+NN. Upon further inspection, we noticed significant similarity between the images in the training and testing sets from Chars74K (in some cases near duplicates) which work to the advantage of a Nearest Neighbor classifier. In contrast, the training and testing split in ICDAR-CH was done on a per image basis, making it highly unlikely to have near duplicates across the split – this helps account for the drop in performance of HOG+NN on ICDAR-CH. Finally, we see that training on purely synthetic data shows competitive performance to training on the native data.

While the character classification accuracy of SYNTH+FERNS appears

[3]http://finereader.abbyy.com
[4]http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/
[5]The dataset has undergone revision since originally benchmarked in Chapter 3.

Table 4.1: Character classification accuracy of Ferns versus previously published results on the Chars74K and ICDAR benchmarks. The SYNTH+FERNS method was trained on synthetic data while the NATIVE+FERNS was trained on data from their respective datasets.

Method	Chars74K-15	ICDAR-CH
SYNTH+FERNS	.47	.52
NATIVE+FERNS	.54	.64
HOG+NN [77]	.58	.52
MKL [16]	.55	-
ABBYY [77]	.19	.21

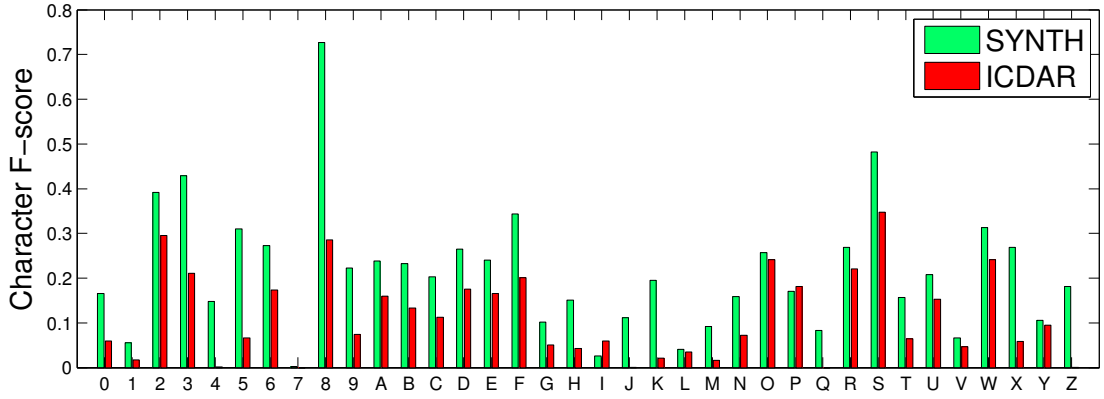


Figure 4.5: Character detection performance (F-score) comparing Fern classifiers trained on synthetic data versus data from ICDAR.

lower than NATIVE+FERNS, our end goal is to use this classifier in a sliding window fashion for character detection. We therefore evaluated the character detection performance of SYNTH+FERNS and ICDAR+FERNS (trained using real characters from the ICDAR data) on the full images from ICDAR. Figure 4.5 shows the F-score, defined as $(\frac{1}{0.5\times precision} + \frac{1}{0.5\times recall})^{-1}$, for each character. From this plot we see that although ICDAR+FERNS performed better on cropped character classification, SYNTH+FERNS is more effective when used for sliding window character detection. A possible explanation for this is that training on synthetic data benefits both from a larger volume of training examples, and from more consistent alignment of the data.

Table 4.2: Accuracy of cropped word recognition comparing Pictorial Structures-based methods (trained on synthetic data and data from ICDAR) to ABBYY FineReader.

Method	ICDAR(FULL)	ICDAR(50)	SVT
SYNTH+PLEX	.62	.76	.57
ICDAR+PLEX	.57	.72	.56
ABBYY	.55	.56	.35

4.2.2 Cropped Word Recognition

Next, we evaluate cropped word recognition on the ICDAR-WD and SVT-WD (the cropped word benchmarks of the respective datasets). This is akin to measuring recall of a system that has a "perfect" text detector. In the SVT-WD case, a lexicon of about 50 words is provided with each image as part of the dataset. For the ICDAR dataset, we measure performance using a lexicon created from all the words that appear in the test set (we call this ICDAR-WD(FULL)), and with lexicons consisting of the ground truth words for that image plus 50 random "distractor" words added from the test set (we call this ICDAR-WD(50)). The latter benchmark allows for direct comparison to SVT-WD. For simplification, we ignore all words that contain non-alphanumeric characters, as well as words with 2 or fewer characters.

Table 4.2.2 shows our results for word recognition on cropped images for three methods: 1) PLEX with Ferns trained on ICDAR-CH data, 2) PLEX with Ferns trained on synthetic data, and 3) ABBYY. The latter is a generic OCR system and does not take a lexicon as input. To simulate lexicon driven OCR, we return the word in the lexicon that has the smallest edit distance to the raw output of ABBYY (i.e., a type of spell checking). It is important to note that evaluating the raw output from ABBYY results in poor performance and some form of post-processing is essential – without spell checking, the accuracy of ABBYY is .21 on the ICDAR-WD(FULL).

Comparing the results in Table 4.2.2 to our previous results from [77], we notice that ABBYY performs considerably better on ICDAR-WD(FULL) than before. This difference was observed after expanding word boxes by 25% in both

dimensions.

These results show that training our system with synthetic data indeed leads to better performance. They also suggest that the SVT dataset is significantly more challenging than ICDAR.

4.2.3 Word Detection and Recognition

Our main experiment consists of evaluating end-to-end word detection and recognition on the ICDAR and SVT datasets. We follow the evaluation guidelines outlined in [45], which are essentially the same as the evaluation guidelines of other object recognition competitions, like PASCAL VOC [22]. A bounding box is counted as a match if it overlaps a ground truth bounding box by more than 50% and the words match (ignoring case).

ICDAR Evaluation We compare performance of several end-to-end pipelines on the ICDAR dataset. Our first pipeline is a combination of a Stroke Width Transform (SWT) and ABBYY (naming this pipeline SWT+ABBYY). We acquired a set of bounding boxes returned by SWT from the authors of [21]; these regions are then fed into ABBYY. As we did before, we correct results from ABBYY by converting its output to the word in the lexicon with the smallest edit distance. In this case, we throw out all bounding boxes for which ABBYY returns an empty string, or for which the smallest edit distance to a lexicon word is above some threshold – this helps reduce the number of false positives for this system.

Next, we apply PLEX to full images without a text detection step (named PLEX). Finally, we combine SWT with PLEX as the reading engine (named SWT+PLEX). This hybrid pipeline serves as a sanity check to see if text detection improves results of PLEX. To show the effect of the re-scoring technique presented in Section 4.1.3, we evaluate the latter two pipelines with and without this step (adding '+R' to the name when re-scoring is used). Motivated by our earlier experiments, we all PLEX-based systems were trained on synthetic data.

We construct a lexicon for each image by taking the ground truth words that appear in that image and adding K extra distractor words chosen at random

from the test set, as well as filtering short words, as in the previous experiment.

Figure 4.8 shows select examples of output; Figure 4.6 shows precision and recall plots for different values of K as we sweep over a threshold on scores (or maximum edit distance for ABBYY, as described above). From these results, we make the following observations. (1) Re-scoring significantly improves performance of PLEX, especially for larger lexicons. (2) The performance of PLEX-based pipelines is significantly better than SWT+ABBYY. While the gap in F-scores of these methods shrinks as the lexicon increases, the PLEX based systems obtain a considerably higher recall at high precision points in all cases. (3) PLEX+R, a system that *does not rely on explicit text detection*, is not only comparable to SWT+PLEX+R, but actually outperforms it for smaller length lexicons.

While an explicit text detection step could in principle improve the precision of a system, the recall is also limited by that of the text detector. Improving the recall of such a two stage pipeline would therefore necessitate improving the recall of text detection. Upon further examination of our results, we found a strong positive correlation between the words that ABBYY was able to read and the words that were detected by the SWT detector. Recall that in the cropped word experiment, ABBYY achieved .56 accuracy on the ICDAR-WD(50) benchmark (correctly reading 482 words). In the end-to-end benchmark of ICDAR with $K = 50$, SWT+ABBYY correctly read 438 words (very close to its performance on cropped words, which simulates a "perfect" text detector). This shows that improving the recall of SWT would not have a big impact on the performance of SWT+ABBYY, unless ABBYY was improved as well.

While ABBYY is a black box, the PLEX pipeline is constructed using computer vision techniques that are well understood and constantly improved by the community. We believe this paves a clearer path towards improving reading accuracy.

The work of [55] reported word recognition results on the ICDAR dataset of 0.42/0.39/0.40, for precision, recall and F-score. In our experiments, we created word lists for every image, however word lists were not provided in the experiments in [55], making the results not directly comparable. The closest comparison in our

framework is to provide the entire ground truth set (> 500) as a word list to each test image. In that case, our PLEX+R pipeline achieves 0.45/0.54/0.51.

SVT Evaluation For the SVT dataset, we evaluated only PLEX and PLEX+R because we were unable to obtain SWT output for this data (and the original implementation is not publicly available). Recall that this dataset comes with a lexicon for each image (generated from local business searches in Google Maps). Figure 4.9 shows examples of output and Figure 4.7 shows precision and recall plots for this experiment. Again we see that re-scoring makes a dramatic improvement in the results. As with the cropped word recognition results, comparing the performance on the ICDAR(50) to the performance on SVT exposes the relative difficulty of SVT. One difference between ICDAR and SVT that may contribute to this difficulty is that for each ICDAR image *all* of the words in that image are contained in the lexicon. On the other hand, in SVT, many of the images contain irrelevant text that leads to a higher number of false positives for our system. Notice, however, that this problem would not be alleviated by the use of a text detector – the burden still lies with the reading module.

4.3 Discussion

These results establish a baseline for using generic computer vision methods on end-to-end word recognition in the wild. We show that we can outperform conventional OCR engines and do so *without* the explicit use of a text detector. The latter is a promising new direction, significantly simplifying the recognition pipeline.

Chapter 4 is based on "End-to-end Scene Text Recognition", K. Wang, B. Babenko, and S. Belongie [76]. The dissertation author was the primary investigator, contributed to algorithm development, implementation, and the writing of the paper.

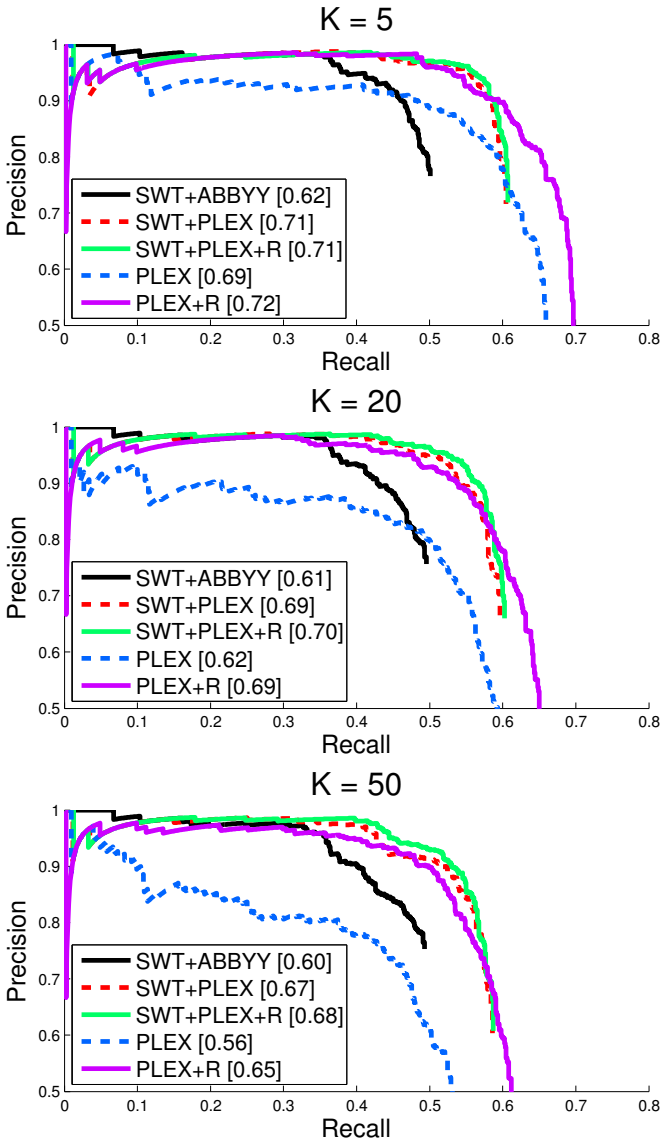


Figure 4.6: Precision and recall of end-to-end word detection and recognition methods on the ICDAR dataset. Results are shown with lexicons created with 5, 20, and 50 distractor words. F-scores are shown in brackets next to pipeline name.

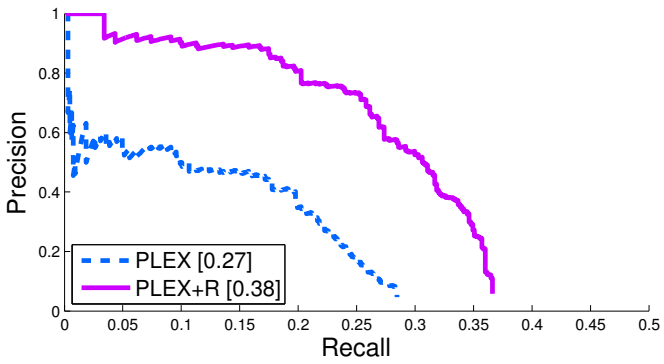


Figure 4.7: Precision and recall of end-to-end word detection and recognition methods on the Street View Text dataset. F-scores are shown in brackets next to pipeline name.

Figure 4.8: Selected results of end-to-end methods on the ICDAR dataset (for a lexicon with $K = 20$ distractors). Results from PLEX+R are shown in green and results from SWT+ABBYY are shown in blue. In the first two images ABBYY has trouble reading text with noisy image conditions and unusual fonts; the last image is more well suited for ABBYY as it is more similar to a scanned document.

Figure 4.9: Selected results on the Street View Text dataset. PLEX+R results are shown in green and words from the corresponding lexicons are shown in dashed pink (recall that these images can contain other irrelevant text).

Chapter 5

Scaling Up Scene Text Recognition via Data-driven Synthesis

Scene text recognition is a challenging problem area that has received increasing attention from the computer vision community. Much work has been done on various sub-problems, including: text detection [12, 21], character classification [16, 13], word recognition [77, 50, 49, 57], and end-to-end systems [76, 56, 78]. As a result, the domain has enjoyed significant advances on an increasing number of public scene text benchmarks [45, 80, 16, 77, 49, 41].

A key to many recent advances in scene text has been in the adoption of supervised learning. For text detection, this problem is posed as binary classification between text and non-text (background) [12]. In character classification, this is a multi-class categorization problem between different symbol classes [16]. With the application of supervised learning, problems in scene text resemble more specialized versions of the familiar general object recognition problems broadly studied in the field.

For supervised scene text recognition systems, there is a need for vast amounts of annotated data. The work of [54] showed that performance of classifying digits taken from house numbers (digits 0-9) continued to improve with 100,000 labeled examples (or about 10,000 per symbol). The need for labeled data is even

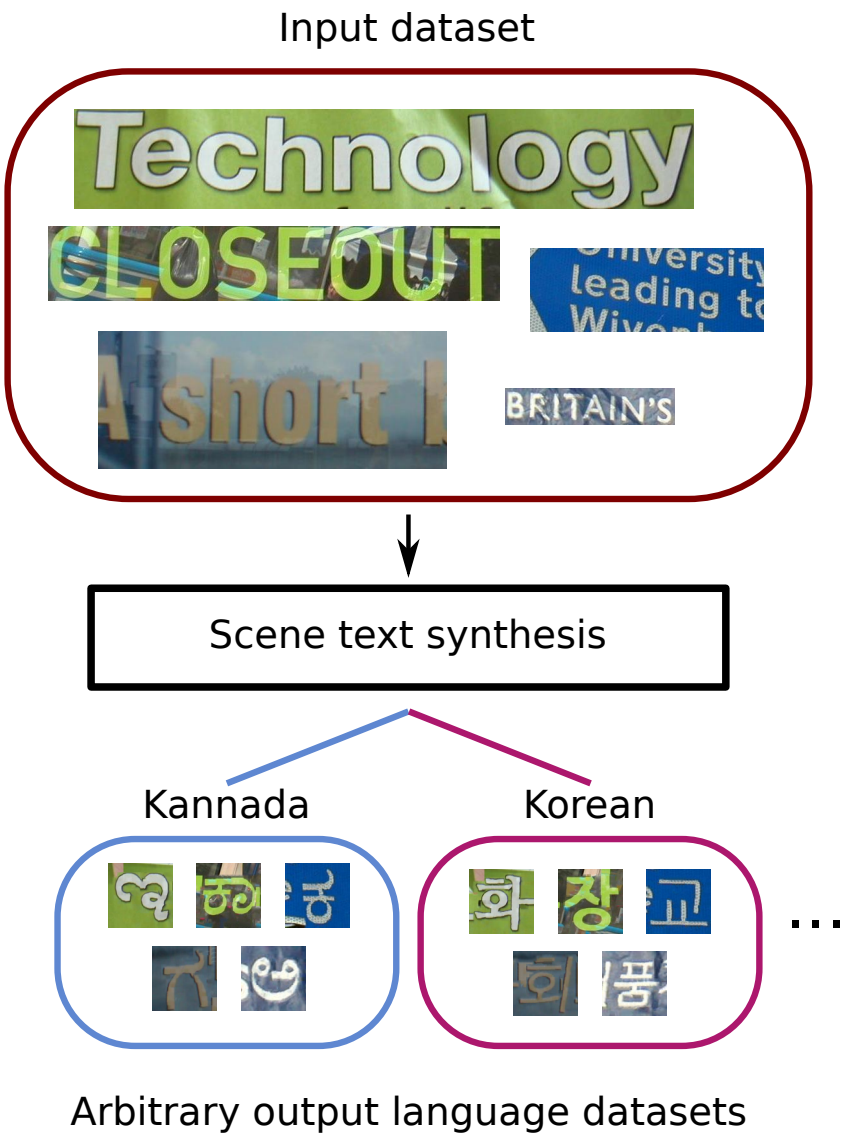


Figure 5.1: We address the looming challenge of scaling up text recognition systems to new languages. We propose to address this through scene text synthesis. In our approach we take a labeled training set of scene text in any language and automatically generate new training for arbitrary languages that retain the visual characteristics of the original scene text. The new data can then be directly used to train existing text recognition pipelines. Our proposal represents a painless way to extend supervised scene text OCR systems to new languages.

Figure 5.2: This figure highlights the redundancy between symbols from different character sets. The shared structure is evident. Our motivation for a data-driven synthesis approach is to re-purpose existing scene text of a particular language to hallucinate characters of another by rearranging the sub-structures. Symbols left to right: Latin 'E', Tifinagh 'yadd', Greek capital 'xi', Russian capital 'ie', and Vai syllable 'dho', Cherokee letter 'gv', Armenian capital 'eh', Euler constant, Canadian syllabics carrier 'sa', and Tai Le letter 'tone-6'. Source: http://shapecatcher.com/ .

more pronounced when confronting one of this domain's 'killer applications': text recognition of arbitrary languages for purposes of translation.

To achieve the goal of general language recognition, supervised systems need a way to *scale up* their recognition capability to space of all visual symbols. Though the scene text domain can be viewed as a specialized form of general object recognition, there are two significant ways in which text differs from other visual objects that make scaling up a challenge: data acquisition and data annotation.

In many object recognition domains, data acquisition and annotation can be managed in relatively natural ways. For many popular datasets (Caltech 256 [31], CUB Birds [82], and ImageNet [17]) images were acquired by crawling the internet. The data was then cleaned and labeled either through paid human annotation (e.g., Amazon Mechanical Turk) or through the help of volunteer citizen scientists (e.g., Cornell Lab of Ornithology[1] and Zooniverse[2]).

Unlike with physical objects, large scale scene text acquisition (in stark contrast to scanned text) is not easily found by crawling the web. It is rare to be able to perform a web search and obtain even a weakly (or noisy) labeled image for a scene text of a particular symbol (in contrast to objects like cars or landmarks like the Eiffel Tower).

Relatively speaking, multi-language scene text annotation is not naturally amenable to paid human annotation. The ability to interpret visual symbols is

[1]http://allaboutbirds.org/labs
[2]http://zooniverse.org

learned and culture-specific, rather than learned and universal (like visual objects in ImageNet). One must find annotators who are literate in a given language. Additionally, visual symbols are not generally regarded as 'charismatic' (like Birds) making it challenging to appeal to volunteer citizen scientists for free annotation.

In this paper, we focus on the problem of character classification. Most supervised character classification systems require bounding box annotations around individual characters [16, 77, 13]. To scale this paradigm up to the extreme, consider the Unicode Standard[3]. This standard supports over 100,000 characters in over 100 languages. If we require mere 100 bounding box annotations for each character class then we still need over 10 million bounding boxes to cover this space of visual symbols. Requiring so many carefully annotated examples for so many different languages may not be feasible via human annotation alone.

Figure 5.2 highlights the intuition behind our work. Though the space of visual symbols is vast, we know there is significant redundancy between symbols: characters in different languages often look similar. There is also redundancy within symbols: characters are typically formed as a collection of lines, corners, t-junctions, and other strokes. We approach the fundamental challenge of scaling up supervised scene text through scene text synthesis: synthetically producing text that appears to have been taken *in the wild* for an arbitrary symbol for the end goal of classification.

Contributions. We present a data-driven technique for scene text synthesis that produces scene text for an arbitrary, new symbol set. Figure 5 illustrates our goal. Our technique takes images of scene text from one language (e.g., English) with standard character bounding box annotations and automatically produces new images of scene text into any other (Korean, Kannada, etc.) language. We experimentally show that the synthesized data yields superior classification performance (using standard features and learning algorithms) on two public datasets, Chars74K (Kannada) and KAIST Scene Text (Korean). Our use of synthesis also enables us to significantly outperform previously published results on the 657-way

[3]http://www.unicode.org/

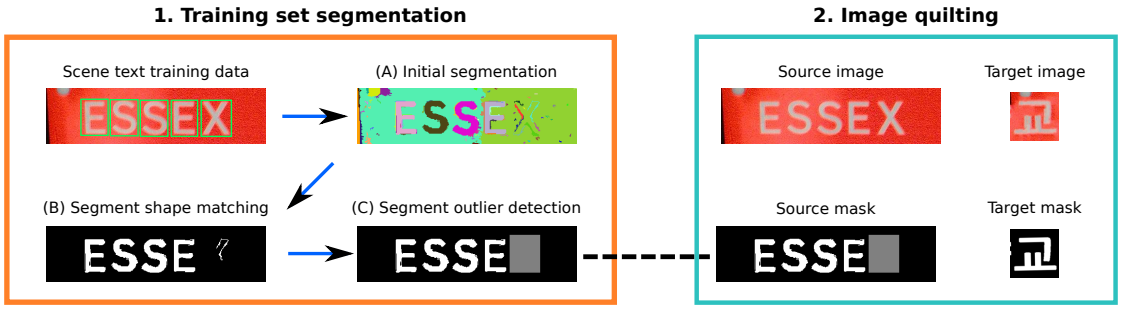


Figure 5.3: This figure shows the two main steps for scene text synthesis. **1. Training set segmentation.** The first step is to infer the foreground/background segmentation in our training data (with character-level bounding boxes). This consists of producing an initial segmentation of the image (A). After that we use shape matching to select the best segment within each character bounding box (B). Then we compute foreground color similarities between each pair of remaining segments, and perform outlier detection to reject spurious segments (C). Character bounding boxes with rejected segments have their entire region ignored (colored in gray). **2. Image quilting.** After masks are produced for all the training data, we apply the Image Quilting [20] to synthesize scene text for an arbitrary font.

classification problem for Kannada. Scene text synthesis enables us to scale up scene text recognition while still benefitting from the performance gains of supervised statistical machine learning.

5.1 Related work

Our work represents a convergence of ideas from diverse sources, combined together to address a practical problem.

Scene text. The creation of the public ICDAR Robust Reading data set by [45] was a catalyst for much of the emerging scene text research to follow. Though Optical Character Recognition (OCR) previously diverged into a separate subcommunity, the problem of OCR *in the wild* has recently seen significant attention from the object recognition community [45, 12, 80, 81, 16, 77, 21, 55, 76, 69, 56, 50, 57, 49, 83, 4].

In [12], the authors demonstrated the promise of using a sliding window-based classifier for text detection. This is a typical first step in a scene text pipeline. The work of [16] demonstrated that simple object categorization methods can significantly outperform conventional OCR engines on character recognition. In [77], they demonstrated the feasibility of approaching OCR as a word categorization problem. This has been extended by [50, 49, 57] who demonstrate the feasibility of *large-lexicon* word recognition. The works of [76, 55, 56] demonstrate full image, end-to-end word recognition systems.

Synthesis for recognition. In certain domains, the use of synthetically generated data has proven itself an effective and efficient way to train recognition systems. Notable instances include 3-D pose recognition using depth cameras [66], pedestrian detection [47, 59], and material texture recognition [43]. We share in spirit with [43] the goal of 'scaling up' the number of recognizable material categories through synthesis.

Our work is not the first to leverage synthetic data or data augmentation for scene text systems ([76] and [13] are recent examples); ours is the first that proposes a data-driven approach for synthesizing training data for recognition of arbitrary languages. This point is critical for 'scaling up' scene text OCR to the vast number of visual symbols in the world.

Texture synthesis and style. In our work we draw from the influential works of [79, 32, 20] in texture synthesis. A source of inspiration for our work comes from [70] who investigate learning to separate style from content. In our problem, we allow font rendering engines to give us access to content – the symbol category – while transferring style through texture synthesis. Also related in this vein is the work of [39].

Domain adaptation. The goals of scene text synthesis are similar to those of recent works in domain adaptation in object recognition [62, 35]. In our work, we explicitly adapt data that we can easily generate – binary masks of text – to ones we cannot – scene text – using techniques from graphics [20].

5.2 Data-driven scene text synthesis

Figure 5.3 shows our full pipeline. The core of our pipeline adopts methods from texture synthesis, specifically, the Image Quilting approach from [20]. The right side of Figure 5.3 shows a result of using Image Quilting. Given a source image and mask (left), one can specify an arbitrary target mask and synthesize a target image (right) that appears visually similar to the source image. In this example, the target mask is a Korean symbol and the resulting target image is that same symbol after adopting the foreground and background textures of the source. The gray region in the source mask is treated as an 'ignore' region, which we further discuss in Section 5.2.1.

In our application, the images and masks have practical meanings. (1) The source images are examples of real scene text and are what we want our synthesized text to look like. In practice, many scene text datasets now exist [45, 77, 49] making source images easy to collect (for a very small set of languages). (2) The source mask specifies which parts of the image are foreground and background. In our synthesized images, we want them to have the same foreground and background as our real scene text. For typical scene text datasets, the masks are not readily available, but we address this in Section 5.2.1. (3) The target mask is how we direct the synthesis process to create new data for an arbitrary symbol. Target masks are easy to produce using standard font rendering engines. Given items 1-3, a target image can be produced through Image Quilting. (4) The resulting target images can be used to train character classifiers for the symbol class represented by the target mask. Target images are exactly the data that is most difficult to collect and annotate for arbitrary languages and are necessary when using supervised learning methods. Producing this data is what our work addresses.

5.2.1 Training set segmentation

Prior to applying Image Quilting we must obtain segmentations for our source images. The left side of Figure 5.3 illustrates our approach. We assume we are given a training dataset of cropped words with character-level bounding box

annotations (e.g., ICDAR [45]). Typically, character segmentation of text in the wild is a challenging problem and is a leading reason why scene text is so much more difficult than scanned document OCR [48, 10]. However, our requirements from segmentation are far simpler than general foreground/background separation due to our end-goal being synthesis. We observe the following simplifications.

- **Segments have known shape.** We have character bounding boxes for the images we are trying to segment, therefore we know approximately the shape of a correct segment. The upper left image in Figure 5.3 shows a training image with its ground truth bounding boxes overlaid. A correct segment is expected to be located within the bounding box with the shape of the ground truth character.

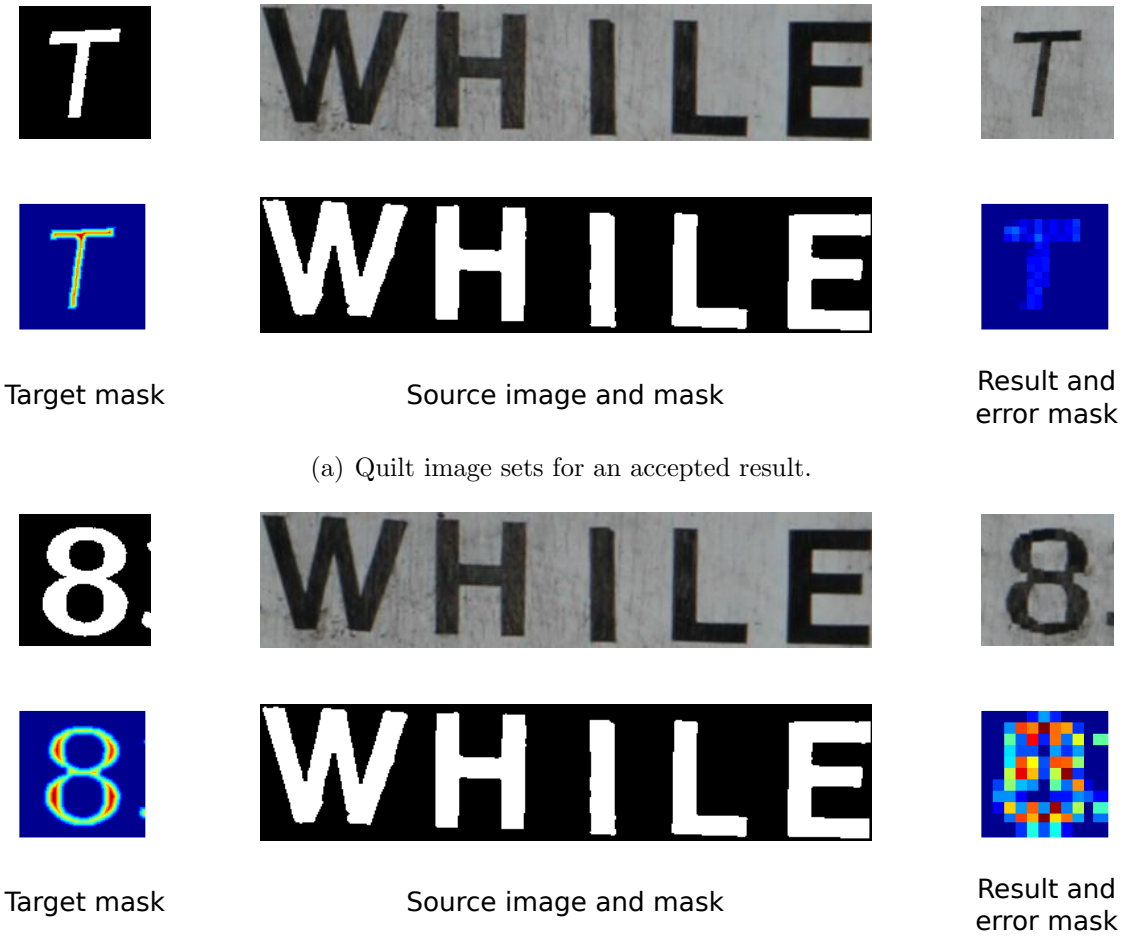
- **Properly segmenting all characters is not required.** Our end goal of training set segmentation is to segment enough foreground and background to be later used for synthesis. We have the flexibility to detect and exclude problematic segments (or entire images) in the synthesis stage.

- **Within a word, segments have consistent texture.** Characters within the same word ought to have similar foreground appearances. This provides a powerful cue for automatically detecting if a group of segments all represent the same foreground.

Given those characteristics of the problem, we describe our segmentation steps, highlighted in Figure 5.3.

A. Produce initial segmentation. The first step is to produce an initial segmentation of the training image. We use the graph-based segmentation method of [23]. Other methods such as [55] can also be used to produce segmentation candidates. Figure 5.3(A) shows an example of the initial segmentation. While the initial segmentation is far from perfect, our next steps mitigate the errors.

B. Use shape matching to select most promising segments. Next, we leverage the available ground truth for each bounding box by selecting the segment that has the most similar shape to its respective ground truth character. For



(a) Quilt image sets for an accepted result.

(b) Quilt image sets for a rejected result.

Figure 5.4: This figure shows the quilt (**Q**) approach. First, a target mask is produced using a standard font rendering engine. The stroke width of this mask is estimated by computing a distance transform on its edge contour map and selecting the maximum value. We measure the compatibility of the source mask and target mask for possible rejection. Figure (a) shows a successfully produced image: the linear substructures are shared between the source and target masks. Figure (b) shows a quilt result that is rejected: the curved structures of the target masks are not well-matched in the source and the result are jagged edges in the place of curves.

each bounding box, we render the ground truth character in `Arial` font and compute the Shape Context [3] score between each segment and the rendered character. Figure 5.3(B) shows the result of selecting the best segmentation component. A single segment remains for each character bounding box. While relying on a single font for shape matching is not optimal, we found it sufficient in our experiments. It's often the case that no good segmentation exists for a ground truth character. Our final step attempts to identify and ignore those segments.

C. Filter bad segments as outliers. To filter spurious segments, we leverage the fact that correct segments within the same word ought to have consistent texture. For each pair of components we compute the χ^2 test statistic of their color histograms. Using these pairwise distances we apply the distance-based outlier method, $DB(p, D)$ [34], to reject components with dissimilar foreground color distributions. A segment is a $DB(p, D)$-outlier if at least a fraction p other segments within the word are greater than distance D. This technique does not make any distribution assumptions about the distances, however it requires choosing the p and D parameters. In our experiments, we set p to be 0.5 and D to be 1/3. This means a segment is rejected if at least half of all the other segments are greater than distance D away. Figure 5.3(C) shows the result after segment rejection. The remaining segments are treated as foreground with the rest of the pixels as background. Character bounding boxes with rejected segments have their entire bounding boxes marked as ignore for the synthesis stage.

5.2.2 Texture transfer

After the training set segmentation is complete, we now have the foreground/background (and ignore) masks to perform synthesis.

1. Generate a target mask. First, we generate a mask of the character (or word) we want to render using standard font rendering engines. We refer to this as the target mask. In the parlance of [70] this mask defines the *content* of our new object: the symbol class it represents. Next we estimate the stroke width of this mask. We do this by applying a Canny edge detector to the mask, computing a distance transform on the edge map, and returning the highest value. We use

the stroke width to define the window size during synthesis.

2. Apply image quilting. Next, we want to transform our target mask to take on the *style* [70] of the source image. Recall that we have two binary masks: one from our source image (real scene text) and a binary mask for the character we wish to produce. Given a source image, its mask, and a target mask, Image Quilting produces a new target image. The method works by copying image patches of a fixed size (stroke width of the target mask) in raster scan order from the source to the target. The regions marked ignore are never matched in this stage. The quilting method transfers patches that balance correspondences between the masks and consistency in the image. We refer the reader to the more detailed description of the algorithm from the original work: [20]. In our experiments, we set the balance between the two to be 0.5.

3. Test for rejection. After a target image is rendered, we attempt to detect and reject those with poor mask correspondences. The result of the quilt is dependent on the suitability of a source image is as a target mask. In Figure 5.4(b), the target mask has many local patches that are curved, while the source is predominantly made up of patches over straight lines. The quilt method produces a jaggy, rather than curved, result. In contrast, in Figure 5.4(a), both target and source are made up of straight line patches, producing a result truer to the target mask. To test for rejection, we compute the mean error of the sum squared difference between mask patches. Those with mean errors higher than a threshold we reject. Because our end goal is to use this data to train a classifier, we can afford to be selective with results from the generation process.

Scene text generation by data amplification. The power of this approach is that it can hallucinate the style of any pixel-segmented text into the shape of any new text. We can painlessly 'scale up' scene text recognition systems to new languages by leveraging the data we already have. Rather than hand tuning a synthesis engine with some set of parameters, this method creates data that inherits the complex photometric characteristics of images of real text. The only parameter of this method is in the selection of a rejection threshold.

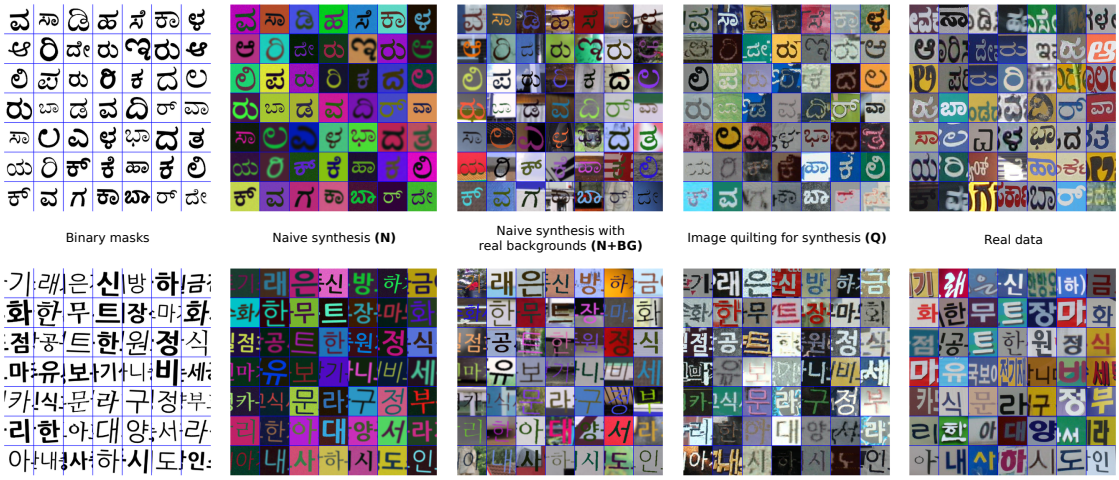


Figure 5.5: This figure shows a montage of the data used in our experiments. The first row are characters from the Kannada experiments and the second are characters from Korean. From left to right: Binary masks , Naive (N), Naive + background (N+BG), Quilt (Q), and real data from the native dataset. Images in Columns 2-4 are created by transforming the same target masks from column 1.

5.3 Experimental setup

The starting point for our synthesis is the target binary mask that specifies the symbol class to be produced. In our experiments, we compare our data-driven synthesis approach to two baseline synthesis methods that also transform a binary mask. In the experiment section, we refer to our method as Quilt (Q). Figure 5.5 shows examples of all methods transforming the same set of binary masks.

Naive (N). The most obvious synthesis method is to begin with the mask and apply simple photometric enhancements. We refer to this approach as *naive*. An instance of naive is generated by selecting a random color for the background, a random color for the foreground, a random amount of Gaussian noise, and random amount of image blur. These are comparable to the training examples synthesized in [76] and we used the same parameters as the authors in that work[4]. The major drawback of this approach is in its requirement of hand-tuning: one must engineer prior knowledge of the target appearance into the mask. The more complicated

[4]Source code from: http://vision.ucsd.edu/~kai/grocr/

the appearance model for the characters (as may be necessary to produce realistic data), the more parameters one must tune.

Naive + background (N+BG). Our other baseline is to use the same naive method to transform the mask foreground but use a different method for the background. We use a separate dataset of urban images to define the backgrounds of the text. To produce an example, we sample a random non-text patch from our dataset, generate an instance of mask, and superimpose it on top of the patch. This data is produced in a similar manner as those from [13]. The downside to this approach is that the appearance of the background patch may not be well-aligned with the distribution of text in the wild.

We evaluate the different synthesis methods in their ability to train classifiers for new languages. We use two public datasets: the Chars74K-Kannada [16] and KAIST Scene Text [41]. Our experiments compare performance across our synthetic methods: naive (N), naive + background (N+BG), and quilt (Q) – while comparing to performance using real data as a reference point. For each language we used 6 different fonts (available on a standard PC) for producing target binary masks. The same binary masks were given to each method, allowing us to examine the impact of texture on synthetic data while holding shape constant. For both language experiments we use the same ICDAR Robust Reading [45] as the training dataset for the quilt method.

Features and classifier. In all our experiments we scaled all characters to 100×100 pixels and used HOG [15] features with spatial bins of size 10, and 8 orientation bins[5]. We used a one-versus-all linear SVM as our classifier[6]. During testing, we apply our classifier over a range of scales on the character – scaling the bounding box from .8× to 1.5× the ground truth size – and keep the maximum response.

[5]http://vision.ucsd.edu/~pdollar/toolbox/doc/
[6]http://www.csie.ntu.edu.tw/~cjlin/liblinear/

Table 5.1: This table lists the performance different training sets on our four experiments. Section 5.3.1 has the details of our setup. Despite using the same number of training examples and using the same set of binary masks to begin with, we see that the quilt method performs the best among the synthesis methods. This shows the relative importance of realistic synthetic data on character classification.

Experiment:	N	N+BG	Q	Real training data
Kannada-17	51.5%	53.5%	55.6%	72.4%
Kannada-49	45.7%	46.7%	52.2%	56.1%
Korean-38	57.9%	55.9%	58.5%	73.7%
Korean-84	47.8%	48.7%	51.8%	58.4%

5.3.1 Data

Chars74K-Kannada. Chars74K is a scene text dataset for the English and Kannada languages [16]. These images were taken using various hand held devices. Many previous works have experimented on the English portion of the data, but due to the high number of character classes (657) and severe imbalance of data across classes, fewer have worked on Kannada. These barriers are also some of the motivating factors for our investigation into scene text synthesis. The Kannada dataset comes with cropped characters of Kannada as scene text (Img) and black/white handwritten text (Hnd). For Chars74K-Kannada, we perform two experiments: full and balanced.

Full. In this experiment we use the entire Chars74K-Kannada for evaluation. We measure the accuracy of our different synthesis methods for 657-way categorization. For all synthesis methods we produced 20 examples per character class. This dataset contains over 4000 characters distributed unevenly across the categories. Note that for this experiment, there is no real image training baseline because all real characters are in the evaluation set. This evaluation allows us to compare to previous results that performed recognition using the handwritten portion of the dataset [16, 65, 36].

Kannada-49 and Kannada-17. In order to control for complications from category imbalance, we subsampled data from the full Chars74K-Kannada dataset to create two balanced experiments. For Kannada-49, we have 49 symbol categories where each category contains exactly 10 examples for training and 10

Table 5.2: Performance on the 657 category Chars74K-Kannada full Img dataset. We can observe a significant difference when training on the data from quilt compared to other synthetic methods. Through using synthetic data, we are also able to significantly outperform previously published results on this task.

Method	**Training**	**Accuracy on Img**
HOG+ESVM [65]	Hnd	1.76%
Shape Context [16]	Hnd	3.49%
Global DCT [36]	Hnd	11.4%
HOG+SVM	N	4.52%
HOG+SVM	N+BG	9.99%
HOG+SVM	Q	16.63 %

examples for testing. For Kannada-17, we have 17 symbol categories and each category contains exactly 20 examples training and 20 examples for testing. The number of categories used in these experiments were the maximum afforded by the full dataset. We compare the accuracy between using the various synthetic methods and use the accuracy of training on the real data as a reference. For all synthesis methods we produced 300 examples per class.

KAIST Scene Text Database. The KAIST Scene Text Database is a scene text dataset containing both English and Korean [41] text. Previous work using this dataset has reported accuracy of text detection while ours is the first to use it for evaluating character classification. In the same way as for Kannada, we created two balanced benchmarks: Korean-84 and Korean-38. For Korean-84 we have 84 Korean symbol categories and 15 examples each for training and testing. For Korean-38 we have 38 Korean symbol categories and 30 examples each for training and testing. For all synthesis methods we produced 300 examples per class.

5.4 Results

5.4.1 Kannada-Full

Table 5.3.1 shows our results on the full Kannada dataset. The first three rows list previously published results on the 657-way classification problem: one method based on Shape Context [16, 3], one based on Exemplar SVM [65, 46], and one using a DCT feature representation [36]. The last three rows show results from training on different synthetic data methods: naive (N), naive + background (N+BG), and quilt (Q). Despite using the same set of binary masks and the same amount of training data, the quilt method performs significantly better than the synthetic baselines, and previously published methods. As suggested in their works [16, 65], the low performance of existing methods is not surprising given they were trained using the Hnd data, which has little resemblance to the real data. The lack of access to real training data is one of the motivators of our work.

5.4.2 Kannada-49, Kannada-17, Korean-84, and Korean-38

Table 5.3 shows the results for our balanced experiments. In all experiments we observe the quilt method performing better than the baseline synthesis methods. In both larger experiments, Kannada-49 and Korean-84, we see an increasing performance gap between the quilt method and the second best performing method.

5.5 Discussion

We have presented a data-driven approach for producing synthetic scene text of arbitrary languages. Our approach takes scene text with standard bounding box annotations from any language (which is readily available) and automatically produces data for any other language (which is hard to collect). We have shown

experimentally that transferring texture to binary masks has a measurable impact on classification performance. Our approach is a step toward maintaining the performance benefits enjoyed through statistical machine learning while mitigating practical issues of 'scaling up' OCR.

Chapter 5, in full, is a reprint of material that has been submitted for publication: "Scaling Up Scene Text Recognition via Data-driven Synthesis", K. Wang, P. Nguyen, A. Bissacco, and S. Belongie. The dissertation author was the primary investigator, contributed to algorithm development, implementation, and was author of the paper.

Chapter 6

Conclusion and future directions

In this dissertation, we have proposed a new direction for performing text recognition in the wild. We have approached the problem not in the traditional OCR sense, but as word spotting: lexicon-constrained word detection. Framing the problem as word spotting achieves the following goals.

- It creates a bridge to engage the broader Computer Vision research community. Placing scene text on the object recognition agenda is crucial for solving the problem because, we argue, that scene text is a form of object recognition, rather than a separate entity.

- It simplifies the scene text domain enough to stay practically relevant, while isolating the contribution of Computer Vision component to the system. This enables meaningful comparisons between different methods. In contrast, building a general OCR system may require heavy reliance on language models and other non-vision processing which can obscure the performance impact of the vision methods themselves.

- It demonstrates considerably higher accuracy in domain-constrained OCR than conventional OCR engines. The work in this thesis and those following [50, 49, 57] have repeatedly shown this to be the case.

In addition to our work on promoting word spotting, we have taken a broad look at the implications of supervised learning-based methods on universal language

recognition. We have argued that data annotation is a looming problem in this domain and proposed a computer graphics-inspired approach to meet the challenge.

To fully realize the potential of an object-centric approach to scene text recognition requires further advances. Some areas of further investigation are the following.

- **Efficient sliding-window engineering.** Due to their advantage in speed, component based methods [55, 21] are often used for text detection as a first stage in a pipeline. However, the tradeoff of this fast first stage is that low contrast and blurred characters are often missed. On the other hand, texture-based methods have shown excellent performance on recognizing even the most challenging characters, at a significant cost in speed. In Chapter 4, we presented an end-to-end method that was fully texture-based though performed far from real-time (currently 15 seconds to process an 800×1200 resolution image). Investigation into cascade classifiers, similar to [18], to develop efficient texture-based pipelines can lead to a significant advance.

- **Integration of synthesis to recognition.** In Chapter 5 we presented a method to synthesize scene text for purposes of character classification. An unexplored direction is to incorporate synthesis into the word recognition stage as well. Synthetic scene text at the word level can be potentially used in a generative model, similar to [71], or in a bag-of-features word-level recognition approach similar to [37].

Bibliography

[1] Bay, H., Tuytelaars, T., and Gool, L. V. Surf: Speeded up robust features. In *ECCV* (2006).

[2] Belhumeur, P., Hespanha, J., and Kriegman, D. Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *TPAMI* (1997).

[3] Belongie, S., Malik, J., and Puzicha, J. Shape matching and object recognition using shape contexts. *TPAMI* (2002).

[4] Ben-Ami, I., Basha, T., and Avidan, S. Racing bib number recognition. In *BMVC* (2012).

[5] Berg, A. C., Berg, T. L., and Malik, J. Shape matching and object recognition using low distortion correspondence. In *CVPR* (2005).

[6] Bishop, C. *Pattern recognition and machine learning*. Springer, 2006.

[7] Bosch, A., Zisserman, A., and Munoz, X. Image classification using random forests and ferns. In *ICCV* (2007).

[8] Canny, J. A computational approach to edge detection. *TPAMI* (1986).

[9] Cappelli, R., Maio, D., Maltoni, D., and Erol, A. Synthetic fingerprint-image generation. In *ICPR* (2000).

[10] Casey, R. G., and Lecolinet, E. A survey of methods and strategies in character segmentation. *TPAMI* (1996).

[11] Chellapilla, K., Larson, K., Simard, P. Y., and Czerwinski, M. Designing human friendly human interaction proofs (hips). In *CHI* (2005).

[12] Chen, X., and Yuille, A. L. Detecting and reading text in natural scenes. In *CVPR* (2004).

[13] Coates, A., Carpenter, B., Case, C., Satheesh, S., Suresh, B., Wang, T., Wu, D., and Ng, A. Text detection and character recognition in scene images with unsupervised feature learning. In *ICDAR* (2011).

[14] Coates, A., and Ng, A. Selecting receptive fields in deep networks. In *NIPS* (2011).

[15] Dalal, N., and Triggs, B. Histograms of oriented gradients for human detection. In *ICCV* (2005).

[16] de Campos, T., Babu, B., and Varma, M. Character recognition in natural images. In *VISAPP* (2009).

[17] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR* (2009).

[18] Dollar, P., Appel, R., and Kienzle, W. Crosstalk Cascades for Frame-Rate Pedestrian Detection. In *ECCV* (2012).

[19] Dollar, P., Wojek, C., Schiele, B., and Perona, P. Pedestrian detection: A benchmark. In *CVPR* (2009).

[20] Efros, A., and Freeman, W. Image quilting for texture synthesis and transfer. In *SIGGRAPH* (2001).

[21] Epshtein, B., Ofek, E., and Wexler, Y. Detecting text in natural scenes with stroke width transform. In *CVPR* (2010).

[22] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. The pascal visual object classes (voc) challenge. *IJCV* (2010).

[23] Felzenszwalb, P., and Huttenlocher, D. Efficient graph-based image segmentation. *IJCV* (2003).

[24] Felzenszwalb, P., McAllester, D., and Ramanan, D. A discriminatively trained, multiscale, deformable part model. In *CVPR* (2008).

[25] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. Object detection with discriminatively trained part based models. *TPAMI* (2009).

[26] Felzenszwalb, P. F., and Huttenlocher, D. P. Pictorial structures for object recognition. *IJCV 61* (2005), 55–79.

[27] Fischler, M., and Elschlager, R. The representation and matching of pictorial structures. *IEEE Trans. on Computers* (1973).

[28] Freund, Y., and Schapire, R. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT* (1995).

[29] Fujisawa, H. Forty years of research in character and document recognition – an industrial perspective. *Pattern Recognition* (2008).

[30] Grauman, K., Shakhnarovich, G., and Darrell, T. Inferring 3d structure with a statistical image-based shape model. In *CVPR* (2008).

[31] Griffin, G., Holub, A., and Perona, P. Caltech-256 object category dataset. Tech. Rep. 7694, California Institute of Technology, 2007.

[32] Hertzmann, A., Jacobs, C., Oliver, N., Curless, B., and Salesin, D. Image analogies. In *SIGGRAPH* (2001).

[33] Huang, G. B., Ramesh, M., Berg, T., and Learned-Miller., E. Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Tech. Rep. 07-49, University of Massachusetts, Amherst, 2007.

[34] Knorr, E., and Ng., R. Algorithms for mining distance-based outliers in large datasets. In *VLDB* (1998).

[35] Kulis, B., Saenko, K., and Darrell, T. What you saw is not what you get: Domain adaptation using asymmetric kernel transforms. In *CVPR* (2011).

[36] Kumar, D., and Ramakrishnan, A. Recognition of kannada characters extracted from scenes. In *DAR* (2012).

[37] Lazebnik, S., Schmid, C., and Ponce, J. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *CVPR* (2006).

[38] Le, Q., Monga, R., Devin, M., Corrado, G., Chen, K., Ranzato, M., Dean, J., and Ng, A. Building high-level features using large scale unsupervised learning. In *ICML* (2012).

[39] Learned-Miller, E. Data driven image models through continuous joint alignment. *TPAMI* (2006).

[40] Lecun, Y. Learning invariant feature hierarchies. In *CVPR-BCVI* (2012).

[41] Lee, S., Cho, M. S., Jung, K., and Kim, J. H. Scene text extraction with edge constraint and text collinearity link. In *ICPR* (2010).

[42] Lehmussola, A., Ruusuvuori, P., Selinummi, J., Huttunen, H., and Yli-Harja, O. Computational framework for simulating fluorescence microscope images with cell populations. *IEEE Trans. Med. Imaging* (2007).

[43] Li, W., and Fritz, M. Recognizing materials from virtual examples. In *ECCV* (2012).

[44] Lowe, D. Distinctive image features from scale-invariant keypoints. *IJCV* (2004).

[45] Lucas, S. M., Panaretos, A., Sosa, L., Tang, A., Wong, S., and Young, R. ICDAR 2003 robust reading competitions. In *ICDAR* (2003).

[46] Malisiewicz, T., Gupta, A., and Efros, A. A. Ensemble of exemplar-svms for object detection and beyond. In *ICCV* (2011).

[47] Marn, J., Vazquez, D., Geronimo, D., and Lopez, A. M. Learning appearance in virtual scenarios for pedestrian detection. In *CVPR* (2010).

[48] Mishra, A., Alahari, K., and Jawahar, C. V. An mrf model for binarization of natural scene text. In *ICDAR* (2011).

[49] Mishra, A., Alahari, K., and Jawahar, C. V. Scene text recognition using higher order language priors. In *BMVC* (2012).

[50] Mishra, A., Alahari, K., and Jawahar, C. V. Top-down and bottom-up cues for scene text recognition. In *CVPR* (2012).

[51] Mori, G., and Malik, J. Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In *CVPR* (2003).

[52] Mori, S., Suen, C. Y., and Yamamoto, K. Historical review of OCR research and development. *Proceedings of the IEEE* (1992).

[53] Nagy, G. At the frontiers of OCR. *Proceedings of IEEE* (1992).

[54] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning* (2011).

[55] Neumann, L., and Matas, J. A method for text localization and recognition in real-world images. In *ACCV* (2010).

[56] Neumann, L., and Matas, J. Real-time scene text localization and recognition. In *CVPR* (2012).

[57] Novikova, T., Barinova, O., Kohli, P., and Lempitsky, V. Large-lexicon attribute-consistent text recognition in natural images. In *ECCV* (2012).

[58] Ozuysal, M., Fua, P., and Lepetit, V. Fast keypoint recognition in ten lines of code. In *CVPR* (2007).

[59] Pishchulin, L., Jain, A., Wojek, C., Andriluka, M., Thormaehlen, T., and Schiele, B. Learning people detection models from few training samples. In *CVPR* (2011).

[60] Rath, T. M., and Manmatha, R. Word image matching using dynamic time warping. In *CVPR* (2003).

[61] Rowley, H., Baluja, S., and Kanade, T. Neural network-based face detection. *TPAMI* (1998).

[62] Saenko, K., Kulis, B., Fritz, M., and Darrell, T. Adapting visual category models to new domains. In *ECCV* (2010).

[63] Schantz, H. *The history of OCR: Optical Character Recognition*. Recognition Technologies Users Association, 1982.

[64] Scholkopf, B., and Smola, A. *Learning with Kernels: Support Vector Machines, Regularization, Optimization and Beyond*. MIT Press, 2002.

[65] Sheshadri, K., and Divvala, S. Exemplar driven character recognition in the wild. In *BMVC* (2012).

[66] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. Real-time human pose recognition in parts from a single depth image. In *CVPR* (2011).

[67] Shotton, J., Johnson, M., and Cipolla, R. Semantic texton forests for image categorization and segmentation. In *CVPR* (2008).

[68] Smeulders, A. W. M., Worring, M., Santini, S., Gupta, A., and Jain, R. Content-based image retrieval at the end of the early years. *TPAMI* (2000).

[69] Smith, D., Feild, J., and Learned-Miller, E. Enforcing similarity constraints with integer programming for better scene text recognition. In *CVPR* (2011).

[70] Tenenbaum, J., and Freeman, W. Separating style and content with bilinear models. *Neural Computation* (2000).

[71] Tu, Z., Chen, X., Yuille, A., and Zhu, S. C. Image parsing: Unifying segmentation, detection, and recognition. *IJCV* (2005).

[72] Turk, M. A., and Pentland, A. P. Face recognition using eigenfaces. In *CVPR* (1991).

[73] Vanhoucke, V., and Gokturk, S. B. Reading text in consumer digital photographs. In *SPIE* (2007).

[74] Viola, P., and Jones, M. Rapid object detection using a boosted cascade of simple features. In *CVPR* (2001).

[75] von Ahn, L., Blum, M., Hopper, N. J., and Langford, J. Captcha: Using hard ai problems for security. In *Eurocrypt* (2003).

[76] Wang, K., Babenko, B., and Belongie, S. End-to-end scene text recognition. In *ICCV* (2011).

[77] Wang, K., and Belongie, S. Word spotting in the wild. In *ECCV* (2010).

[78] Wang, T., Wu, D., Coates, A., and Ng, A. End-to-end text recognition with convolutional neural networks. In *ICPR* (2012).

[79] Wei, L., and Levoy, M. Fast texture synthesis using tree-structure vector quantization. In *SIGGRAPH* (2000).

[80] Weinman, J., and Learned-Miller, E. Improving recognition of novel input with similarity. In *CVPR* (2006).

[81] Weinman, J. J., Learned-Miller, E., and Hanson, A. R. Scene text recognition using similarity and a lexicon with sparse belief propagation. *TPAMI* (2009).

[82] Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., and Perona, P. Caltech-ucsd birds 200. Tech. Rep. CNS-TR-2010-001, California Institute of Technology, 2007.

[83] Yao, C., Bai, X., Liu, W., Ma, Y., and Tu, Z. Detecting texts of arbitrary orientations in natural images. In *CVPR* (2012).