Word Spotting in the Wild

Kai Wang and Serge Belongie

Department of Computer Science and Engineering
University of California, San Diego
{kaw006,sjb}@cs.ucsd.edu

Abstract. We present a method for spotting words *in the wild*, i.e., in real images taken in unconstrained environments. Text found in the wild has a surprising range of difficulty. At one end of the spectrum, Optical Character Recognition (OCR) applied to scanned pages of well formatted printed text is one of the most successful applications of computer vision to date. At the other extreme lie visual CAPTCHAs – text that is constructed explicitly to fool computer vision algorithms. Both tasks involve recognizing text, yet one is nearly solved while the other remains extremely challenging. In this work, we argue that the appearance of words in the wild spans this range of difficulties and propose a new word recognition approach based on state-of-the-art methods from generic object recognition, in which we consider object categories to be the words themselves. We compare performance of leading OCR engines – one open source and one proprietary – with our new approach on the ICDAR Robust Reading data set and a new word spotting data set we introduce in this paper: the Street View Text data set. We show improvements of up to 16% on the data sets, demonstrating the feasibility of a new approach to a seemingly old problem.

1 Introduction

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. 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 for systems that prevent automated software bots from abusing internet resources, which are known as CAPTCHAs [1]. 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.

Our use of the phrase *in the wild* is analogous to Labeled Faces in the Wild (LFW) [2]: a data set constructed to study face recognition in unconstrained

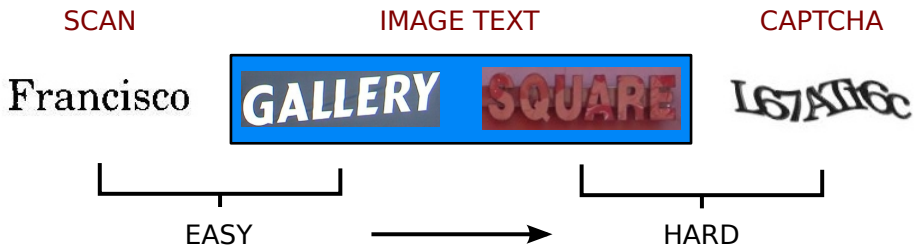
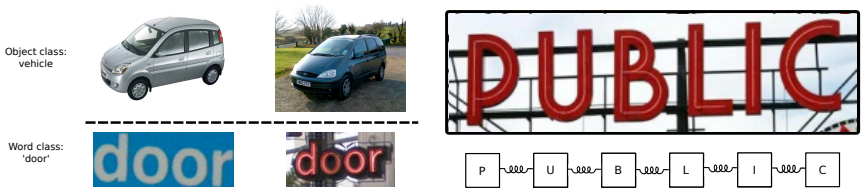


Fig. 1. This figure shows examples of images of words ordered by difficulty. In the extreme cases, the behavior of OCR engines is well understood: it is highly accurate when reading scanned text (far left) and is inaccurate when reading a CAPTCHA[1] (far right). In between these two extremes sits text found in the wild. Due to its unconstrained nature, in some cases the image text is similar to scanned text and can be read, while in others it cannot.

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 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 2(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 [3]. (3) We present a new word spotting approach that imports techniques from generic object recognition and significantly outperforms conventional OCR based methods.



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

Fig. 2. 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.

2 Motivating Applications

Accurate word spotting plays an important role in systems for image retrieval and navigation. Research in Content Based Image Retrieval (CBIR) [4] 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 [5] 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 Dataset

We introduce the Street View Text (SVT) data set harvested from Google Street View[1]. Image text in this data exhibits high variability and often has low resolution. Figure 3 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.

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

Fig. 3. 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.

Data Collection. 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 a unique city 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 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.

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

4 Related Work

4.1 Scanned document OCR

The topic of OCR has been well studied [6] [7] and existing commercial products are in widespread use. One example is Google Book Search[4], which has scanned more than 10 million volumes[5], making them accessible for full text searches. Another example is the Kurzweil National Federation of the Blind (KNFB) reader.[6] 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 [8] 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.[7] The task of separating out individual characters was also identified in [9] 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.

4.2 Image text OCR

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 [10]. In that work, detected text regions 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 [11]. In the work of [12] 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 [13] and [5]. In [13], 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 [5], the authors performed word spotting in scanned handwritten historical documents. To perform word spotting, they clustered words together by appearance,

[4] http://books.google.com/
[5] http://googleblog.blogspot.com/2009/10/tale-of-10000000-books.html
[6] http://www.knfbreader.com/
[7] http://yann.lecun.com/exdb/mnist/index.html

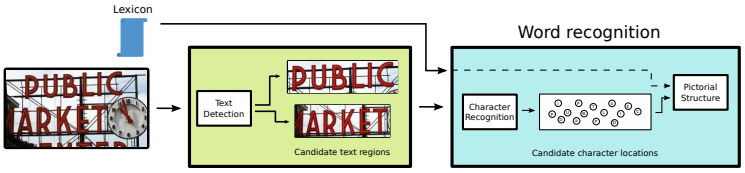


Fig. 4. Word spotting overview. This is an illustration of a word spotting system with two steps: text detection [10] and word recognition. In this work, we focus on the latter problem where the input is an image region and a lexicon of words. In our Street View Text data set, the lexicon was created 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.

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 [14] [15]. We also build on the work of [16], who studied the use of various features and classification methods to classify individually cropped characters.

5 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 4 shows a diagram of this pipeline.

5.1 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) [17] 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.

5.2 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 [14] [15]. 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 2(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:

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

The parameter θ controls the balance between part match cost and deformation 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 [15]. 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.

6 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 [3], 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[8], referred to as ABBYY and TESS. In using the OCR engines, we experimented with pre-thresholding the images using the technique from [10], 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.

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

6.1 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 1.

In Table 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) [18] and Shape Context (SC) [19], and classified using Nearest-Neighbor (NN) as reported in [16]. The methods listed were the ones that performed best from [16].

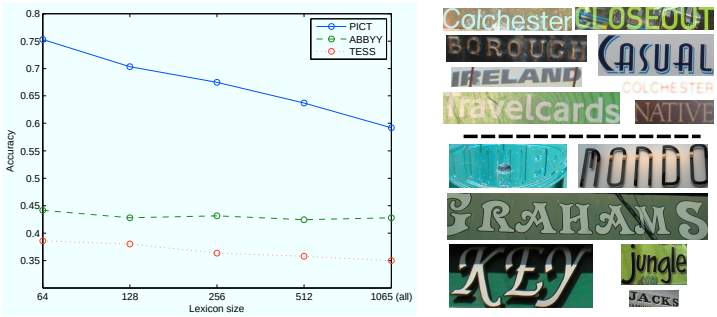
Feature	Chars74K-5	Chars74K-15	ICDAR03-CH
HOG+NN	**45.33 ± .99**	**57.5**	**51.5**
MKL	-	55.26	-
GB+NN	36.9 ± 1.0	47.09	27.81
SC+NN	26.1 ± 1.6	34.41	18.32
ABBYY	18.7	18.7	21.2
TESS	17.3	17.3	17.4

Table 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.

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.

6.2 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.



(a) (b)

Fig. 5. 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.

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 2.

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

Table 2. Number of trials for each lexicon size.

For each size k, we took all our testing data, randomized the order, and 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 5 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.

7 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 [20] 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 7 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 shows the breakdown of results after separating the data.

	ICDAR (1065)			SVT		
METHOD	ALL	EASY (40%)	HARD (60%)	ALL	EASY (33%)	HARD (67%)
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. 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.

Fig. 6. 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.

	ICDAR (1065)			SVT		
METHOD	ALL	EASY (40%)	HARD (60%)	ALL	EASY (33%)	HARD (67%)
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

Table 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.

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 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 7 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 misdetected 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 4 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.

Fig. 7. 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.

8 Conclusion

In this paper 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.

9 Acknowledgements

We thank Boris Babenko and Steve Branson for helpful conversations, and Grant Van Horn for assistance with data collection. This material is based upon work supported by NSF CAREER Grant No. 0448615, an NSF Graduate Research Fellowship, a Google Research Award, and the Amazon AWS in Education Program.

References

1. von Ahn, L., Blum, M., Hopper, N.J., Langford, J.: Captcha: Using hard ai problems for security. In: Eurocrypt. (2003)
2. Huang, G.B., Ramesh, M., Berg, T., Learned-Miller., E.: Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst (2007)
3. Lucas, S.M., Panaretos, A., Sosa, L., Tang, A., Wong, S., Young, R.: ICDAR 2003 robust reading competitions. ICDAR (2003)

4. Smeulders, A.W.M., Worring, M., Santini, S., Gupta, A., Jain, R.: Content-based image retrieval at the end of the early years. IEEE Trans. PAMI **22** (2000) 1349–1380
5. Rath, T.M., Manmatha, R.: Word image matching using dynamic time warping. CVPR (2003)
6. Nagy, G.: At the frontiers of OCR. In: Proceedings of IEEE. Volume 80. (1992) 1093–1100
7. Mori, S., Suen, C.Y., Yamamoto, K.: Historical review of OCR research and development. Document Image Analysis (1995) 244–273
8. Casey, R.G., Lecolinet, E.: A survey of methods and strategies in character segmentation. IEEE Trans. PAMI **18** (1996) 690–706
9. Chellapilla, K., Larson, K., Simard, P.Y., Czerwinski, M.: Designing human friendly human interaction proofs (hips). In: CHI. (2005)
10. Chen, X., Yuille, A.L.: Detecting and reading text in natural scenes. In: CVPR. (2004)
11. Vanhoucke, V., Gokturk, S.B.: Reading text in consumer digital photographs. In: SPIE. (2007)
12. Weinman, J.J., Learned-Miller, E., Hanson, A.R.: Scene text recognition using similarity and a lexicon with sparse belief propagation. IEEE Trans. PAMI **31** (2009) 1733–1746
13. Mori, G., Malik, J.: Recognizing objects in adversarial clutter: Breaking a visual CAPTCHA. In: CVPR. (2003)
14. Fischler, M., Elschlager, R.: The representation and matching of pictorial structures. IEEE Trans. on Computers **22** (1973) 67–92
15. Felzenszwalb, P.F., Huttenlocher, D.P.: Pictorial structures for object recognition. IJCV **61** (2005) 55–79
16. de Campos, T., Babu, B., Varma, M.: Character recognition in natural images. In: VISAPP. (2009)
17. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR. (2005)
18. Berg, A.C., Berg, T.L., Malik, J.: Shape matching and object recognition using low distortion correspondence. In: CVPR. (2005)
19. Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. IEEE Trans. PAMI **24** (2002) 509–522
20. Canny, J.: A computational approach to edge detection. IEEE Trans. PAMI **8** (1986) 679–698