Reading Text in Consumer Digital Photographs

Vincent Vanhoucke and S. Burak Gokturk

http://www.riya.com

Riya, 3 Waters Park Drive, Suite 210, San Mateo, CA, USA

ABSTRACT

We present a distributed system to extract text contained in natural scenes within consumer photographs. The objective is to automatically annotate pictures in order to make consumer photo sets searchable based on the image content. The system is designed to process a large volume of photos, by quickly isolating candidate text regions, and successively cascading them through a series of text recognition engines which jointly make a decision on whether or not the region contains text that is readable by OCR. In addition, a dedicated rejection engine is built on top of each text recognizer to adapt its confidence measure to the specifics of the task. The resulting system achieves very high text retrieval rate and data throughput with very small false detection rates.

Keywords: text detection, text recognition, model combination, digital photographs

1. INTRODUCTION

A lot of progress has recently been made in detecting text from natural images, with engines capable of extracting typically up to 70% of the text in natural scenes.[1] However, many text-like features exist in natural scenes, which low-level image feature analysis is very hard-pressed to distinguish from actual text. Figure 1 illustrates a few typical man-made patterns which commonly result in false positives during text extraction. Morphological analysis of the text recognized would easily eliminate (a) as a false positive, but examples like (b) where letters can be identified ('G O CCC'), or (c) where text-like features are intertwined with actual words are a lot more difficult to deal with. Given these difficulties, it is mostly due the lack of robustness of text recognition systems that current technology fails to deliver a completely reliable solution to extracting textual content from digital images.

Multiple issues are a challenge to traditional OCR technology in this context: first, the resolution of the text can be very small and inconsistent, particularly when the text appears slanted to the camera; second, extracting the text and separating it from the environment is difficult to perform with high accuracy, which causes some significant amount of irrelevant non-text components to be presented to the OCR; third, text found in natural scenes often use very artistic fonts with inconsistent spacing and alignment, which most OCR engines are not well equipped to parse.

Commercial OCR companies have typically focused their efforts on perfecting the accuracy of scanned high-resolution fixed-DPI (dot per inch) documents, at the expense of robustness to low-resolution, possibly distorted text that might not be coming from a clean black and white document source. Recently, the industry appears to be moving towards improving their robustness to such inputs. In this context, the use of voting to improve the robustness of OCR engines has a long history:[2–5] by leveraging and combining the hypotheses of multiple text recognition engines trained independently, the overall accuracy can be significantly enhanced at the cost of additional processing time.

Figure 1. Typical false positives resulting from text detection: (a) a row of windows, (b) buttons on a dashboard, (c) natural clutter around text.

This paper describes a text extraction method which mitigates the limitations of OCR technology through improved confidence measures and the combination of text recognition engines using bayesian model combination (BAYCOM[6]). BAYCOM was initially developed for automatic speech recognition to deal with situations where one requires a system to accept unreliable speech input, yet operate at a very low false accept rate. It builds on ROVER[7] which was meant as a method to enhance the reliability of speech recognition systems, and introduces a principled way to combine confidence scores to determine the most likely output in bayesian fashion.

The complete system builds on top of the text detector described in.[8] That detector achieved the second best overall detection performance in the ICDAR 2005 text locating competition[1] while being 40 times faster than the best contender. The efficiency of this detector makes it very suitable to quickly process large amounts of data. Significant improvements to some of the components of the detector are described in section 2.

Section 3.2 describes a generic confidence engine model which leverages features specific to text extracted from digital pictures to improve on the character and word-level confidence measures that commercial OCR engines typically provide. Section 3.3 describes how BAYCOM is applied to the text recognition problem, and Section 3.4 describes an efficient pipelined implementation of the combination method.

2. TEXT DETECTION

The text detection front-end is a typical AdaBoost[9] cascade. The algorithm used for detection and text normalization is described in detail in[8] : the input image is downsampled multiple times to produce a set of images at various resolutions. Each of these images is passed through a feature extraction module which performs an analysis of the intensities and edges in the image. The downsampled image is then scanned using a rectangular mask, and an AdaBoost cascade classifier is used to evaluate the probability of each rectangular patch to contain a text segment at that resolution. The rectangles are then clustered into likely text regions, which are binarized based on the local statistics of the image. Finally, connected components are extracted from each region and analyzed using a set of heuristics to separate lines of text from the background components. In the following sections, we will focus on a few important enhancements we made to the original system.

2.1. Font Size Segmentation

The text detection cascade produces a collection of possibly overlapping patches which are candidate text segments. Each patch i has dimension $H_i \times 2H_i$, where H_i is a good initial approximation of the height of the detected text over that segment. Patches are clustered based on their overlap to produce candidate text regions. This initial estimate $H = E(H_i)$ of the text scale is then used to determine at which scale the text binarization should proceed for optimal results. A shortcoming of this approach is illustrated in Figure 2. In this instance, small text is present in the immediate vicinity of very large text, which means that the average height H of text in this detection region will match neither actual font sizes.

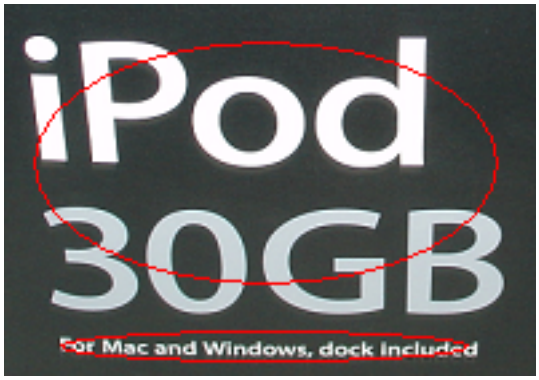

Figure 2. Splitting of text regions based on font scale using Gaussian splitting.

To address this issue, we perform a coarse segmentation of the text by clustering the text patches based on their coordinates (X_i, Y_i) and their height H_i. We use a tree-structured EM algorithm[10] to fit full-covariance

Gaussians in the (X_i, Y_i, H_i) space. The benefit of Gaussian Mixture clustering is that it allows clusters to have different covariance structures, which means that no assumptions are made about the shape of each text cluster independently. By initializing the EM algorithm along the H direction and measuring whether the resulting clusters overlap or not in (X_i, Y_i) space, we are able to robustly determine whether or not the text region under consideration consists of separable clusters with distinct average scales.

2.2. Discriminative Color Model

The detected text regions are binarized using the local statistics of the image intensity. The text is then extracted using connected component analysis based on the type of relationships that are expected between letters: consistent spacing, arrangement along a line and consistent size. Another useful constraint is the consistency of the letter color within a given word or word segment.

Introducing color consistency as a factor in the text decoding is surprisingly difficult. First and foremost, the amount of data we have to estimate a color model for each connected component can be very small. Consider for example building a color model for a period ('.') when the font is the size of the font used in this line of text: the amount of data is very small. Secondly, because we are looking at text extracted from signs and objects in a natural setting, illumination changes and shadows can very significantly alter the perceived hue of any particular letter compared to its neighbors. Aliasing is also a very dominant effect: when letters are made up of connected components of different sizes, aliasing might cause the smaller components to appear having very different tone due to leakage of the background color.

As a consequence, a generative color model for individual connected components is not appropriate, and we have to resort to a discriminative model. The intuition for this model is very simple: for each connected component C, we build both a foreground Gaussian $\mathcal{G}_f(C)$ and a background Gaussian $\mathcal{G}_b(C)$ in YCrCb space. The foreground Gaussian is estimated based on the connected component pixels, while the background Gaussian is estimated from pixels around the component which do not belong to any other foreground component. This is achieved by applying dilation to the binary image, subtracting the original binary image from the dilated one, and considering the remaining foreground pixels in the neighborhood of the component under consideration. Figure 3 illustrates the background extraction method.

Figure 3. Background model for connected components: in green (filled letters), the connected components; in red (around each letter), the background pixels obtained through dilation and subtraction; in blue (rectangular box), the boundary of the local window around the connected component corresponding to the letter 'T'. All the red pixels within the blue box will be used as a background model for 'T'.

A discriminative color distance measure can then be constructed by asking whether a given component C is closer to the foreground or background model of another component C' using a ratio of their Mahalanobis distance $\mathcal{M}$:

$$d(C, C') = \frac{\mathcal{M}(\mathcal{G}_f(C), \mathcal{G}_f(C'))}{\mathcal{M}(\mathcal{G}_f(C), \mathcal{G}_b(C'))} \tag{1}$$

This distance, combined with the other structural constraints between letters, enhances the separation between foreground letter components and background components.

Figure 4. Example of a font for which the 1 connected component / letter assumption breaks down. In this instance, the red components were successfully re-connected with their corresponding letter. The green dot on the 'i' was not due to the vertical gap between the letter body and the dot.

2.3. Letter Fragment Recovery

A strong assumption of the word decoding process is the identification of connected components with letters. This is a very good heuristic in practice for latin characters in most of the fonts typically used on signs and objects. Such fonts are typically much more readable and are hence favored on most signs. Dots on i and j letters are not very salient and can be discarded at little cost in the OCR performance.

In many cases, this assumption breaks down, at great cost to the recognition accuracy, as illustrated in Figure 4. To deal with such cases, connected components which are discarded during the reading process are collected and are evaluated in turn as candidate letter fragments. The evaluation is based on a distance metric which incorporates the relative size and position between fragments, as well as their color distance.

This simple approach solves most of the issues resulting from the connected component assumptions when letters contain dots or thin edges which disappear at low resolution. It does not address the case of connected letters forming a single connected component, which in general is a much more difficult case for OCR engines to deal with.

3. TEXT RECOGNITION

3.1. Experimental Setup

Experiments throughout this paper were run on an Intel Pentium 4 3GHz, using a consumer photo album containing 5316 pictures with an average resolution of 1.5 megapixel. This amounts to 2.7Gb of JPEG encoded data. Each picture was individually labeled with all the text contained in the picture, including partially occluded words and words written using artistic fonts. 1591 photos contained any text at all (15% of the pictures), with a total number of words of 10552. The text detection process takes on average 880 milliseconds per photograph. This paper is not intended as a performance comparison of commercial OCR engines, hence the engines we used will only be referred throughout as A, B and C.

3.2. Improved Confidence Measures

OCR engines typically use some form of layout analysis to segment text from images in a document. As a result, the recognition engine itself is not primarily designed to perform a text vs. non-text decision. The confidence measure produced by the engine is a good assessment of whether a recognized character has a chance of having been substituted with another, but makes a poor rejection measure on input which might not be text at all. To complement this confidence measure, we extract a variety of salient features from the recognizer output and train a text / non-text classifier using a combination of those features. The features used as input to the classifier depend on the type of information provided by the recognition engine, and typically include:

- the text detection confidence, based on the cumulative posterior computed by the classifier,
- the average character confidence,
- the language model probability,
- the difference between the font size as estimated by the text normalization, and the font size estimated by the OCR: any discrepancy indicates a problem during reading,

- the percentage of letters recognized as bold: bold letters tend to be better segmented,
- the percentage of letters recognized as sans-serif, which are more common on signs than in typical text as expected by an OCR engine,
- the percentage of letters that are digits or lower case: upper case letters are more common on signs,
- the number of letters in the word: false detections tend to result in very short letter sequences,
- the portion of the detection window covered by the letters: any discrepancy indicates that a word might be missing from the output,
- the presence of repetitive sequences of letters: it is common for man-made patterns such as window rows and fences to be detected as 'oooooo' or 'IIIIIII', as depicted in Figure 1,
- the case consistency within a word: except for an initial capital letter, a lower-case word should be lower case throughout, and similarly upper-case words should be consistent.

We used Linear Discriminant Analysis (LDA, see e.g. [11, Chapter 4]) to combine these features. LDA is in general very simple and robust, but makes some strong assumptions about the class-conditional covariances, which in practice are not met when the training data contains many outliers. The effect of outliers can be eliminated by performing a resampling of the training data: first we train the two-class LDA based on all the available data, then rank the training data based on its LDA score. Remove the top 60% of the high-score class and the lowest 60% of the low-score class from the training data, and retrain the classifier. This ensures that outliers away from the decision boundary don't skew the class-conditional covariance and generally leads to better class separation.

Figure 5 shows the impact of the confidence engines on three different commercial engines we experimented with. This simple approach significantly reduces the false positive rate of the engines, allowing them to be used in this challenging environment without retraining.

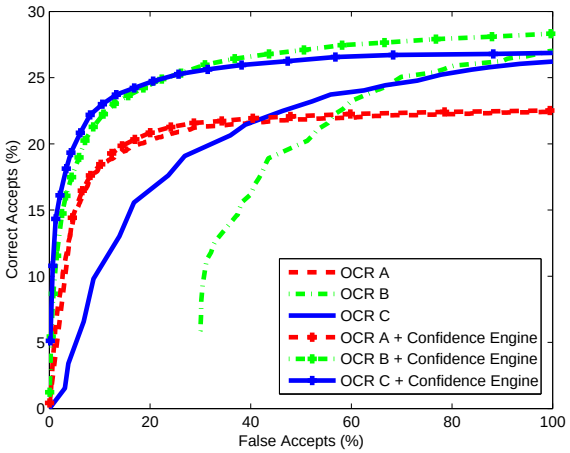

Figure 5. Confidence engine performance.

3.3. Bayesian Model Combination

Bayesian model combination treats model combination as a decision theoretic problem. It was shown in[6] that under the assumption that the statistical models to be combined are independent of each other, the posterior probability of any output can be computed very easily. This formulation generalizes confidence-based voting by taking directly into account the relative strength of the different classifiers.

We will describe here a simplified version of BAYCOM which illustrates how the technique can be implemented in practice. Assume that we have N classifiers, which each output a hypothesis H_i with confidence score s_i. No assumption is made about how this score correlates with error rate.

A simple derivation shows that the log-posterior $\mathcal{L}$ of hypothesis H to be correct based on the N classifiers is:

$$\mathcal{L}(H) = \log P(H) + \sum_{i:H_i=H} \alpha_i + \log \frac{p_i(s_i|C)}{p_i(s_i|E)} \tag{2}$$

$P(H)$ denotes the prior of hypothesis H, and can be omitted if there is no prior information independent of the classifiers. $p_i(s_i|C)$ is the probability that classifier i outputs score s_i assuming that the hypothesis is correct. $p_i(s_i|E)$ is the probability that classifier i outputs score s_i assuming that the hypothesis is incorrect. α_i is an offset which has a closed form expression based on the absolute error rate of classifier i. In practice, α_i can be determined experimentally on held out data.

Note that the log-posterior is computed as a sum over all hypotheses H_i which are equal to H, which means that each classifier contributes additively to the total score when they each select the same hypothesis, exactly like they would do in a traditional voting scheme. $p_i(s_i|C)$ and $p_i(s_i|E)$ can be estimated by accumulating the score histograms for correct and incorrect held out data, and taking the log-difference of the two histograms.

Figure 6 demonstrates the efficacy of the method. At a 4% false accept rate, the overall recall is improved by 25% compared to the output of the best engine.

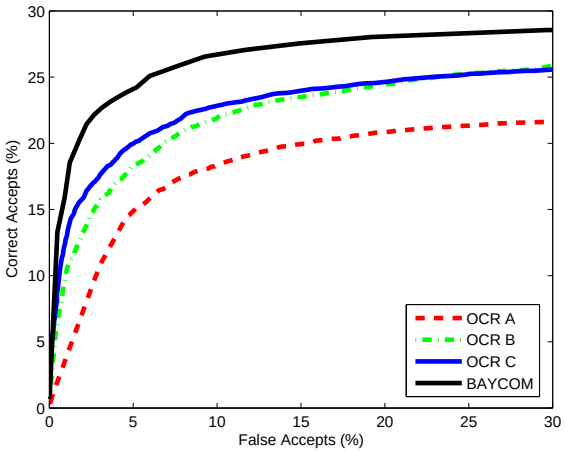

Figure 6. Model combination performance.

3.4. Cascaded Architecture

The main drawback of model combination is that the CPU time required to process the images is now multiplied by the number of OCR engines used in the combination. In addition, because of the large number of false positives returned by the text detection, there is a large overhead associated with processing false positives. To alleviate both these problems, we arranged the model combination into the cascade depicted in Figure 7.

The detected text is first passed through engine A, which is the fastest and least accurate. Based on the confidence of the text read by this engine, a large proportion of the false positives get immediately discarded. The remaining text is passed successively through the other engines in decreasing order of speed. After each recognition, the candidate hypotheses computed so far are evaluated through model combination and further rejection is applied based on their confidence measures. This pipeline architecture makes sure that the least amount of CPU is spent on candidate text regions that are likely to be false positives, while focusing the

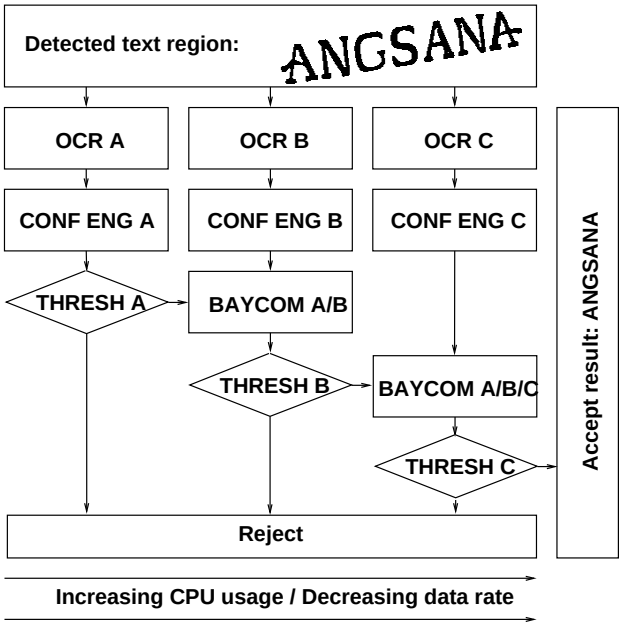

Figure 7. Cascaded text recognition architecture.

Engine A	160 ms
Engine B	173 ms
Engine C	524 ms
Combination A+B+C	850 ms
Cascade A+B+C	420 ms

Table 1. Text recognition CPU usage per photograph.

resources of the system on the relevant text. The confidence thresholds at each stage are experimentally tuned so that no loss of accuracy is incurred. Alternatively, a principled method for selecting those thresholds has recently been proposed:[12] by formulating the problem as a constrained optimization over a known speed vs. accuracy landscape, a global search with efficient heuristics can be performed to yield the desired performance characteristics.

The CPU consumption of the recognition system is depicted in Table 1. The cascading of the recognizers saves about 50% of the CPU time. Using this cascade, the performance degradation at no rejection is about 1% absolute. However, the performance characteristics between the 0% and 5% false accept points are entirely unaffected by the speedup.

4. CONCLUSION

The system described in this paper uses a cascaded approach to text detection, normalization, recognition and confidence estimation in order to quickly extract the text contained in natural scenes with high recall, while tightly controlling the rate of false accepts. Improvements to the detection include an improved character analysis of the detected text using better scale estimation, a discriminative color model and letter fragment analysis. Improvements to the recognition include a bayesian approach to model combination, and faster processing using a cascade of recognizers.

Used in the context of text-based image retrieval, the low false positive rate that the text extraction is able to achieve limits the amount of pollution incurred by the the search engine index. It also enables the system to display the text on the user interface with a very high degree of perceived accuracy (Figure 8).

A variant of the engine described is available on the web at `http://www.riya.com`.

Figure 8. The high accuracy and low false positive rate of the system makes it possible to display the recognized text on the user interface without adding clutter, and to enable users to manually edit and correct the results.

ACKNOWLEDGMENTS

The authors would like to thank Xiangrong Chen, Stellan Lagerstrom, Diem Vu and Neelesh Vaikhary for their contributions to this system.

REFERENCES

1. S. M. Lucas, "ICDAR 2005 text locating competition results," in *Proc. ICDAR*, 2005.
2. J. Handley and T. Hickey, "Merging optical character recognition outputs for improved accuracy," in *Proc. RIAO 91 Conference*, pp. 160–174, 1991.
3. S. V. Rice, J. Kanai, and T. A. Nartker, "A report on the accuracy of OCR devices," Tech. Rep. TR–92-02, ISRI, Univ. Nevada Las Vegas, Las Vegas, Nevada, 1992.
4. X. Lin, "Reliable OCR solution for digital content re-mastering," Tech. Rep. HPL-2001-232, HP Laboratories Palo Alto, 2001.
5. T. Ho, "Multiple classifier combination: Lessons and next steps," in *Hybrid Methods in Pattern Recognition*, A. Kandel and H. Bunke, eds., pp. 171–198, World Scientific, 2002.
6. A. Sankar, "Bayesian model combination (BAYCOM) for improved recognition," in *Proc. ICASSP*, **1**, pp. 845–848, 2005.
7. J. G. Fiscus, "A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER)," in *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding*, pp. 347–354, December 1997.
8. X. Chen and A. L. Yuille, "Detecting and reading text in natural scenes," in *Proc. CVPR*, **2**, pp. 366–373, 2004.
9. Y. Freund and R. E. Schapire, "A decision-theoretic generalization of on-line learning and an application to boosting," in *European Conference on Computational Learning Theory*, pp. 23–37, 1995.
10. J. Bilmes, "A gentle tutorial of the EM algorithm and its applications to parameter estimation for gaussian mixture and hidden Markov models," Tech. Rep. TR-97-021, International Computer Science Institute, Berkeley, California, 1998.
11. A. Webb, *Statistical Pattern Recognition*, Arnold, 1999.
12. K. Chellapilla, M. Shilman, and P. Simard, "Optimally combining a cascade of classifiers," in *Proc. of Document Recognition and Retrieval XIII, SPIE-IS&T Electronic Imaging*, **6067**, pp. 60670Q–1, January 2006.