Fast and Low-power OCR for the Blind

Frank Liu

CS229 - Machine Learning
Stanford University

1. Introduction

Optical character recognition, or OCR, is the process of extracting text from scanned documents. Although OCR is a well-studied topic, text embedded in natural scenes also carries extremely useful information. As such, it is often necessary for computers or mobile devices to automatically recognize them.

Among the various subtopics of scene text classification, character classification (i.e. the process of recognizing various letters and digits) is perhaps the most important. However, this task is rather difficult, since recognizable characters often appear in a variety of different scenes with a variety of different designs. Additionally, they are difficult to accurately classify due to the variant properties of scene text.

A common way to learn images of characters is to apply a scale and viewpoint-invariant feature extraction algorithm with a classification algorithm. For the English alphabet, this classification algorithm would categorize feature sets from each image into 36 categories (26 letters and 10 numbers, excluding symbols such as ”!” and ”-”). Many classification algorithms are viable, but each produce different results when paired with a different set of features.

While OCR has many applications, blind men and women have perhaps the greatest use for letter and word recognition. The vOICe project [10] is an excellent example of this. vOICe kits attempt to help the blind recognize everyday objects, such as a pedestrian walkway or a parking lot. The package also comes with an OCR algorithm, which allows users to listen to, for example, the time displayed on a clock or text written in a book. However, one of the downsides of the vOICe project is that its OCR algorithm can take a couple of seconds to process text, and the accuracy is somewhat questionable given the low-resolution camera input.

Despite the potential for OCR as seen in the vOICe project, the vast majority of recent OCR research is focused towards improving classification accuracy. Batuwita and Bandara, for example, present a constrained OCR algorithm for fuzzy, low-resolution scanned letters and numbers. While their work presents a high classification accuracy (98%), it takes over half a second to classify each character. [2] Similarly, Si *et al.* [7] and Namane *et al.* [1] both achieve a high-accuracy (approximately 96%) classification method for scanned text, but explicitly mentions the immense computational complexity of their algorithms. For many recent OCR-related algorithms, runtime will often double or triple for a just few percent increase in accuracy. Although there are certain advantages to having extremely accurate algorithms, OCRs usefulness to the blind only manifests itself when letters and characters can be quickly recognized and output to the user through audio or some other form of sensory communication.

This paper will detail the design and implement a novel method for fast, low-power word recognition, creating an algorithm designed to use small amounts of computing power while still maintaining a reasonably high level of accuracy on everyday recognizable characters. Section 2 explores several feature extraction and classification algorithm combinations. Some of these algorithms are tweaked some portions of the algorithms to obtain better classification accuracy or a lower runtime. Section 2 also presents a qualitative formula that determines the quality of a particular combination of feature extraction + classification algorithms. Finally, Section 3 presents optimized results compiled using the best selected combination of feature extraction + classification algorithms. Testing will occur on the OCR dataset collected by Rob Kassel at the MIT Spoken Language Systems group (found here: www.seas.upenn.edu/ taskar/ocr/letter.data.gz).

2. Methodology

A power-efficient OCR algorithm attempts to minimize the computational steps used in the OCR process while still achieving a high classification accuracy. For mobile and low-power applications, an extremely accurate but computationally expensive algorithm is undesirable; on the other hand, a fast and power-efficient algorithm with a relatively low classification accuracy is also inadequate.

To better quantitatively measure this balance, the OCR method employs a weighted sum over the runtime and test set classification accuracy to determine which combination is the best. This equation is based on user preferences generated by the vOICe project:

$$B(r, a) = \frac{r}{n} + a \tag{1}$$

where r is the overall runtime over the test set, a is the classification accuracy, and n is the number of points in the character dataset. This formula reflects the desire to have an algorithm as close to 100% as possible, while still maintaining a strong relative runtime.

To achieve maximum balance between computing efficiency and recognition accuracy, several potential algorithms learning algorithms were analyzed to see which produced the best results:

(1) K-nearest neighbor, $K = 5$

(2) K-nearest neighbor, $K = 20$

(3) Naive Bayes, $n = 0.01 * \frac{N}{62}$

(4) Support Vector Machines (SVM)

For kNN, K refers to the number of closest neighbors analyzed. For Naive Bayes, the parameter n refers to the Laplace smoothing factor, conveniently defined as 1% of the test set's element count divided by 62 (the total number of lowercase and uppercase alphanumeric characters in the English alphabet).

To allow the OCR method to learn the characters, each learning algorithm was paired with a feature extraction algorithm, designed to take keypoint vectors at each pixel in the image. These features consisted of some variable number f of feature vectors, each of which was 64-bits wide. Feature vectors which had less than 64 bits were simply zero-padded until the vector had length of 64. The following image shows a *selected* set of SIFT interest keypoints extracted from an image of Huang Engineering Center:

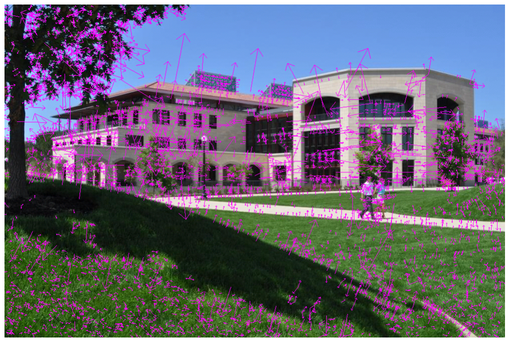

Figure 1: *Selected SIFT features taken from an image of Huang Engineering Center.*

Below is a list of all of the feature extraction algorithms tested for this project:

(1) Harris Corners [5]

(2) Histogram of Gradients

(3) Scale-Invariant Feature Transform (SIFT) [8]

(4) Speeded-up Robust Features (SURF) [4]

To keep runtime calculations consistent, only MATLAB libraries and scripts were used to code the feature recognition and classification algorithms.

As per Equation 1, each combination of algorithms was analyzed for both runtime and test-set accuracy. The table below shows a summary of the results. The maximum output value from Equation 1 was normalized to 100.

In the experiments, the Gaussian kernel was used to train the SVM classifier. Surprisingly, SIFT performed considerably better than HoG, while SURF

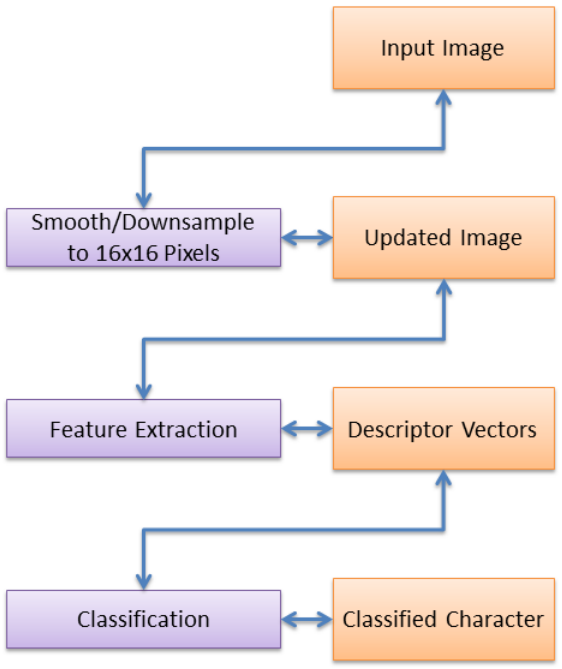

Figure 2: *Flow chart for the algorithm.*

	kNN5	kNN20	NB	SVM
HC	61.03	65.72	26.40	48.95
HoG	63.28	73.80	50.08	83.62
SURF	71.54	81.44	86.62	100
SIFT	65.93	69.52	75.69	88.16

Table 1: *Normalized values (100 = best) for B(r, a) for different algorithm combinations.*

outperformed SIFT by a relatively small amount despite the reported 4x speedup over SIFT.

3. Optimizations

As shown in Section 2, SURF with SVM has shown better results than all of other algorithm combinations. To further improve the OCR method, several techniques were used to improve both the classification accuracy and runtime of the algorithm.

3.1. Code Conversion to C

To improve runtime performance and decrease library and platform-related overhead, the original SURF + SVM code was ported from MATLAB to C using libraries provided by Rob Hess from Oregon State University. [6] This provided an almost three-fold improvement in runtime, with no change to the classification accuracy.

3.2. Image Downsampling and Smoothing

In order to improve the efficiency of the algorithm, the original image is downsampled to a 16×16 square image via bicubic interpolation. Furthermore, to compensate for spurious high frequency information in the downsampled image, the method employs a Gaussian lowpass filter to smooth out any potential sharp corners. This increases the reliability of the SURF descriptor.

Using LIBSVM [3] to fine tune the classification parameters, two distinct patters were created for feature extraction and training, both designed to compensate (in one way or another) for the system's lack of computing power.

(1) 12×12 window size, 128 descriptor elements per feature. Although the descriptor for each pixel contains quite a bit of information, the number of pixels we consider is rather small.

(2) 6×6 window size, 72 descriptor elements per feature. This pattern reduces the number of elements per descriptor, but increases the number of pixels in the interest region.

For pattern 1 (see Figure 3), a search range of $\log_2(C) = [3, 5, ..., 13, 15]$ and $\log_2(\gamma) = [-15, -13, ..., -5, -3]$. A region was then identified around $(C, \gamma) = (8, 0.03125)$ which has a high cross-validation accuracy of 75.2%. Immediately following this, the grid search was rerun using $\log_2(C) = [1.5, 2.5, 3.5]$ and $\log_2(\gamma) = [-6.5, -5.5, -4.5]$. This produced a cross-validation accuracy of 75.4% and an optimized pair of parameters - $(C, \gamma) = (5.6569, 0.0221)$ Using these parameters, a test set accuracy of $3545/5198 = 68.1993\%$ was achieved.

For pattern 2 (see Figure 4), a search range of $\log_2(C) = [1, 3, ..., 11, 13]$ and $\log_2(\gamma) = [-15, -13, ..., -5, -3]$ was used before determining the ranges for fine grid search to be $\log_2(C) = [2.5, 3, 3.5]$ and $\log_2(\gamma) = [-7.5, -7, -6.5]$. The best parameter

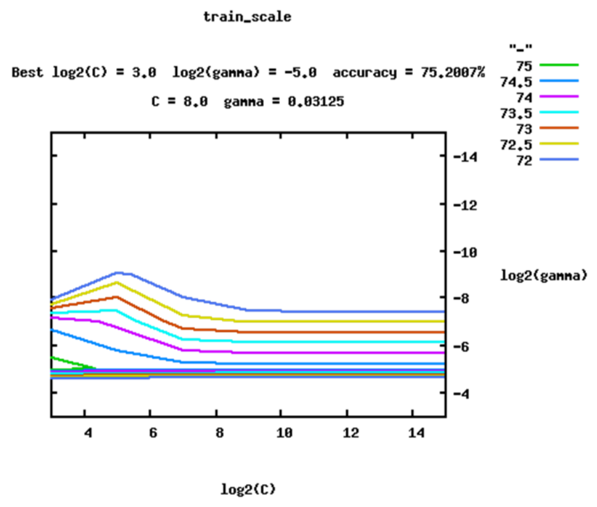

Figure 3: *Grid search for pattern 1. Top graph corresponds to a coarse search, whereas the bottom graph corresponds to a fine search.*

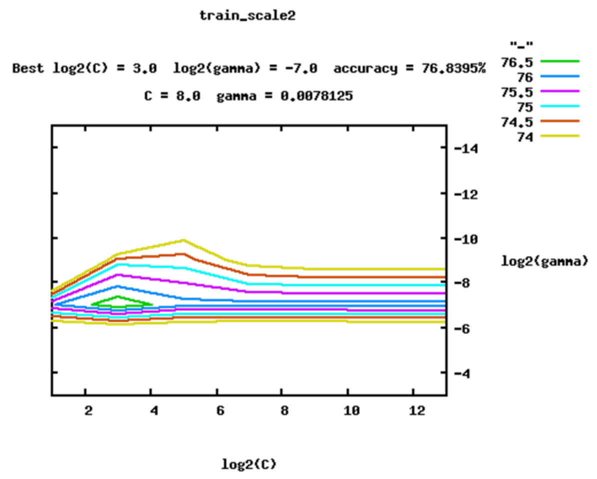

Figure 4: *Grid search for pattern 2. Top graph corresponds to a coarse search, whereas the bottom graph corresponds to a fine search.*

pair (C, γ) turned out to be $(5.6569, 0.0078)$, turning in a test set accuracy of $3607/5198 = 69.3921\%$.

Given the different possible current OCR algorithms, 70% likely is very close to the maximum possible accuracy achievable while still maintaining a low runtime. Both patterns were able to achieve approximately 0.4ms classification runtime per character.

4. Results

Preliminary ”field tests” were generated using an field-programmable gate array (FPGA) running a MIPS-like processor at a clock frequency of 150MHz. Due to time constraints, the updated x86-based OCR method was not ported directly to the MIPS assembly. Instead, Valgrind [9] was used to determine the number of instances of each assembly instruction that occurred when running the OCR method over the test set. These instructions and their corresponding number of occupancies N in the code were then copied to the FPGA, which then ran each instruction N times.

Over the entire test set, the FPGA-based system observed a 5.07ms classification runtime per character. This means that the system can correctly classify approximately 70 out of 100 characters every half second on a 150MHz clock speed.

5. Further Work

Low-power OCR has huge potential as a blindness aid, especially as processing power becomes cheaper and cheaper. Although the OCR system shown here has shown a good set of preliminary results, there is still plenty of room for future work.

(1) Comprehensive field tests. In this project, no real field tests (with an actual live webcam feed and user) were performed. Doing so would be a huge step forward in the project, and could allow for more optimizations tailored towards actual user-reported problems.

(2) Addition of text-to-speech capabilities. At the moment, there is no way for a blind user to assess the output of the OCR method. Adding text-to-speech software and recompiling runtime results would allow for a better understanding of the viability of the system in a real-world setting.

(3) Improved accuracy for OCR algorithm. While a low runtime is critical for the application to perform well, a reasonably high accuracy just as important; 70% classification accuracy may be a bit too inaccurate for a user to perform tasks on a daily basis. To increase classifications accuracy or decrease runtime, more learning and feature extraction algorithms could be analyzed. Image

modification for more robust feature extraction is also another possibility to improve the OCR method.

References

[1] E. H. Soubari P. Meyrueis A. Namane, M. Maamoun. Csm-based feature extraction for degraded machine printed character recognition. In *proceedings of ICMLC'08*, 2010. 1

[2] K. Batuwita and G. Bandara. Fuzzy recognition of offline handwritten numeric characters. In *proceedings of CIS'10*, 2010. 1

[3] C. Chang and C. Lin. http://www.csie.ntu.edu.tw/ cjlin/libsvm/. 3

[4] T. Tuytelaars H. Bay and L. V. Gool. Surf: Speeded-up robust features. In *proceedings of ECCV'06*, 2006. 2

[5] C. Harris and M. Stephens. A combined corner and edge detector. In *proceedings of AVC'88*, 1988. 2

[6] Rob Hess. http://eecs.oregonstate.edu/ hess. 3

[7] X. Tian J. Si, F. Yang. A new algorithm of mixed chinese-english character segmentation based on irregularty degree. In *proceedings of ICMLC'08*, 2010. 1

[8] D. G. Lowe. Distinctive image features from scale-invariant keypoints. In *IJCV'04*, 2004. 2

[9] Valgrind. http://valgrind.org/. 4

[10] The vOICe project. http://www.seeingwithsound.com/ocr.htm. 1