FONT FINDER: VISUAL RECOGNITION OF TYPEFACE IN PRINTED DOCUMENTS

Tu Bui and John Collomosse

Centre for Vision Speech and Signal Processing,
University of Surrey,
Guildford, United Kingdom.
{tb00083 | j.collomosse}@surrey.ac.uk

ABSTRACT

We describe a novel algorithm for visually identifying the font used in a scanned printed document. Our algorithm requires no pre-recognition of characters in the string (i. e. optical character recognition). Gradient orientation features are collected local the character boundaries, and quantized into a hierarchical Bag of Visual Words representation. Following stop-word analysis, classification via logistic regression (LR) of the codebooked features yields per-character probabilities which are combined across the string to decide the posterior for each font. We achieve 93.4% accuracy over a 1000 font database of scanned printed text comprising Latin characters.

***Index Terms*—** Optical Font Recognition (OFR), Typeface Classification, Logistic Regression.

1. INTRODUCTION

This paper contributes a novel algorithm for recognizing the font used to generate a passage of printed text. Font recognition is the specific task of detecting the style of glyphs; i. e. the basic alphabet of characters used to synthesize text, as distinct from other typeface attributes such as weight, style (e. g. italicized), or size. Although optical character recognition (OCR) has received considerable attention in recent decades, automated methods for recognizing typeface in printed text have been sparsely researched. Nevertheless with so many fonts now in open circulation on digital font exchanges, it is desirable to be able to identify the particular font used within an existing digital image or document.

Our proposed algorithm accepts a digital image as input, and after pre-processing to segment characters from the passage of text on the page, extracts gradient orientation information local to character boundaries. A supervised classification strategy using logistic regression (LR) determines the probabilities of each individual character being one of a set of 1000 pre-trained fonts. These probabilities are combined to yield a posterior probability for each font. During training a discriminative codebook is learned from extracted boundary features, which is used to quantize the feature-space used for LR classification at test-time. We adopt a Bag of Visual Words (BoVW) framework with stop-word analysis to form this codebook — a technique commonly used for object and activity recognition — here applying this approach for the first time the problem of optical font recognition (OFR).

2. RELATED WORK

OFR techniques typically operate over bounding-boxes, positioned by a segmentation pre-process to encapsulate individual characters on the page. Subsequent processing can be broadly categorized as adopting either of two strategies: 1) computing a global descriptor from texture within the box; 2) analyzing local typographic features within box sub-regions.

The first category of method typically adopts spectral or statistical texture descriptors as a representation. Multi-dimensional Gabor filters were used by Zhu et al. [1], coupled with a k-NN classification strategy using the L^2 norm to classify Latin characters, and later by Ha and Tian [2] for Chinese characters. Typically accuracies for these systems were $\sim 96\%$ for a 10 font dataset. Ma and Doermann [3] later explored a grating cell operator and neural network classification scheme as alternative to Gabor as a spectral approach, yielding improvements of a couple of percent for a 5 font dataset evaluated over Latin, Greek and Cyrillic characters. A global shape descriptor using a quad-tree decomposition was shown by Sexton et al. [4] to perform with $\sim 95\%$ accuracy over 50 fonts. One-dimensional transforms of signals generated by projecting character profiles to the horizontal/vertical axis were explored by Zramdini and Ingold [5] and combined with naïve Bayes classification for typeface (i. e. font, size and style) recognition, and could discriminate between 7 fonts with $\sim 92\%$ accuracy. Commercial websites exist for OFR, however these are semi-automated requiring user-supplied knowledge of each character's ASCII classification i. e. OCR. It appears that at some of these websites use template cross-correlation to match against a database of sample font images collected for each ASCII character independently. This contrasts with approaches in the academic literature that do not require an OCR pre-process.

Zramdini and Ingold [6] also explored local feature analysis, proposing a set of eight local typographical features of discriminative value for OFR extracted from independent spatial regions of the character. The approach required precise alignment of characters to extract the features, and adopted an empirically calibrated thresholding approach on their values to discriminative between a set of 10 fonts with $\sim 97\%$ accuracy. These typographical features were later used in a conditional random field (CRF) [7] to discriminate up to 11 fonts using a variety of classification approaches, however the most promising (Eigenmodel/PCA) yielded only $\sim 70\%$ ac-

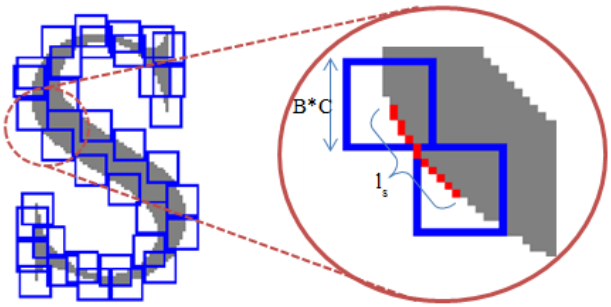

Fig. 1. Feature Extraction. HOG features (blue) are computed local to points spaced equidistantly ($l_s = B \times C$) around edges of the character, isolated within an area-normalized bounding box. B and C indicate the *block* and *cell* size of HOG windows in pixels.

curacy however unlike [6] other typeface attributes e. g. size and style could also be identified.

Our algorithm is based upon local features and so also falls within this second category of technique. However rather than prescribing a set of typographical features and corresponding spatial locations [6, 7], uniquely we *learn* a set of discriminative features using a gradient domain operator and codebooking approach. This enables us to generalize recognition to a dataset of 1000 fonts; an order of magnitude greater than prior approaches, whilst retaining a competitive performance of $\sim 93\%$ accuracy (see Sec.4).

3. OPTICAL FONT RECOGNITION

As with prior approaches, we assume a pre-processing step that identifies a bounding box around each character within the printed passage. This is achieved in our system via a series of morphological filtering and thresholding operations, however any approach could be substituted (see [8] for a brief review). Our algorithm accepts as input a set of characters segmented in this form, processing each independently to determine a font classifications which are then fused to produce a definitive classification for the text passage.

We now describe the process by which the system is trained (Sec. 3.2), followed by the classification process (Sec. 3.3). Both operations depend upon a process for extracting visual features from a bounding box containing a single character, which we now explain (Sec. 3.1).

3.1. Feature Extraction

Given a printed character, we begin by uniformly scaling the bounded region to a constant area of $\sim$ 25k pixels. Typically this yields characters of approximately 150×200 pixels. A set of chain codes are constructed from the edges of the character, providing a means to iterate around its external (and any internal) boundaries. We extract a set of Histogram of Oriented Gradient (HOG) descriptors [9] local to points at equi-spaced intervals (l_s pixels) around the boundaries.

In our configuration, each HOG descriptor is computed using a block and cell size [9] of 3×3, yielding a regular grid of side length$l_s = 9$ pixels, centered upon the point being sampled (Fig. 1). Frequency histograms of gradient orientation are computed within each cell using a quantization level of 8 bins. The frequency histograms are normalized and then concatenated to form the $3 \times 3 \times 8 = 72$ dimensional HOG descriptor for that point. Typically a set of around 30-40 descriptors are extracted in this manner per character.

3.2. Codebook generation

Given a training dataset comprising 5 examples of each letter (A-Z) from 1000 fonts, we collect ~ 4 million HOG descriptors via the above method. Hierarchical k-means clustering is applied to a random sub-sample of these data comprising 10% of the descriptors, yielding k cluster centers quantizing the HOG descriptor space – suitable values for k are explored in Sec. 4.3. A hard-assignment Bag of Visual Words (BoVW) strategy is followed, assigning all training descriptors to the nearest cluster center via k-NN so mapping each descriptor to one of k codewords (cluster center). A frequency histogram is formed by counting the codewords for each training character.

Stop-words are identified by summing the frequency histograms for all training characters and identifying the S most frequent codewords. Suitable values for S are explored in Sec. 4.3. These typically correspond to straight sections of glyphs that are by their commonalities are non-discriminative within the training set. We therefore apply stop-word removal by expunging the corresponding bins of the frequency histograms computed for the training characters. The frequency histograms are finally normalized and used as the visual descriptor for the training character in subsequent training of the classifier.

3.3. Font Classification

We divide the training visual descriptors into classes corresponding to their respective fonts $\mathcal{F} = \{F_1, ..., F_{1000}\}$, and train a multi-class logistic regression (LR) classifier using a one-vs-all combination strategy. Meta-parameter search is performed to optimize the auxiliary cost parameter during training via 5-fold cross-validation. As a global codebook was generated over all letters and fonts, the elements of the descriptors correspond to visual features that mutually distinguish different fonts and letters. In this $k - S$ dimensional representation, LR learns linear decision boundaries to separate fonts irrespective of the letter used.

Given a query string of characters $Q = \{q_1, ..., q_{|}Q|\}$ we can decide the classification of any given character q_i by extracting its visual descriptor via the process of Secs. 3.2-3.2 and testing against each of the $f \in |\mathcal{F}|$ component SVMs to evaluate $p(F_f|q_i)$, so determining the font used F_{q_i}:

$$F_{q_i} = \underset{f}{\operatorname{argmax}}\, p(F_f|q_i). \quad (1)$$

Individual classification of q_i will fail when the glyph is non-discriminative, for example letters 'O','A','K' etc. are

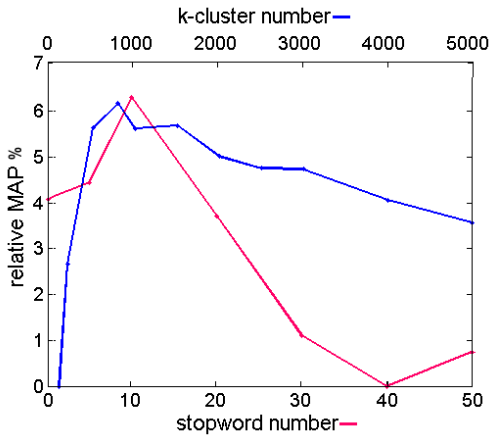

Fig. 2. Analysis of dictionary parameters; codebook size (k) and percentage of stop-words discarded (S). Plotting relative performance (MAP) vs. a baseline using $k = 100, S = 40$. Performance is shown to peak at $k = 800, S = 10$.

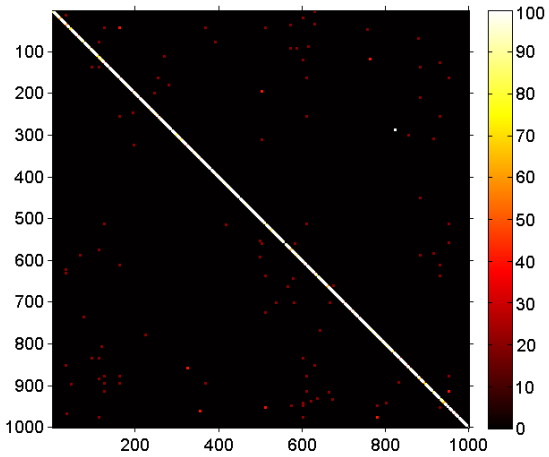

Fig. 3. Confusion matrix characterising performance (MAP %) over 1000 fonts using a corpus of English text passages of $\sim$ 100 characters. Overall accuracy 93.4% MAP at $k = 800, S = 10$ for 600dpi scan.

frequently similar between typefaces. Thus a joint probability over all Q is preferred:

$$F_Q = \underset{f}{\operatorname{argmax}} \prod_{i=1}^{|Q|} p(F_f|q_i). \quad (2)$$

A threshold on $p(F_Q|Q) < 0.5$ may be used to determine the null category i.e. no matching font found.

4. RESULTS AND DISCUSSION

We evaluate over a comprehensive dataset of 1000 TrueType fonts selected to represent a wide gamut of styles including examples of cursive, mono, gothic, serif, sans-serif and decorative fonts; 865 sourced from the Internet and 135 from fonts installed as default by the Microsoft Windows 7 operating system. Without loss of generality, only samples of the capital letters A-Z were considered in the evaluation of our work. The dataset comprised 10 strings each of 100 characters length for each font (i. e. 1m printed characters). The resulting test document ($\sim$ 600 pages) was printed on a Ricoh Aficio MP C4000 printer/copier, and scanned back in using a sheet-feeder at resolutions of 200-600dpi (in 100 dpi increments) to create a multi-resolution dataset. The dataset was scanned three times to triple-sample the printed content (each scan resulted in slightly different digital output due to imaging noise).

4.1. Predictive accuracy over text passages

We first predict the performance of our system over passages of English language text.

$$P(F_n|Q) = \max_f \prod_{i=1}^{26} p\left(F_n|q_i\right)^{\rho(q_i)}. \quad (3)$$

where in this example the printed string Q='$ABC...Z'$ in a particular font F_m to be tested, and $\rho(.)$ is a prior probability proportional to the frequency of occurrence of the character q_i in text. By marginalizing this weighted score over all characters we obtain an estimate for the probability of font F_n being correctly detected given a typical piece of English text.

The mean average precision (MAP) over all fonts, i. e. mean of $P(F_n|Q)$ over all 1000 fonts is 92.2%. This compares favorably to existing techniques surveyed in Sec. 2 that achieve accuracies within a few percent [5, 6, 4, 3, 7] but over a font set containing tens rather than 1000 of fonts (i. e. two orders of magnitude smaller than us). In this experiment the codeword dictionary size was $k = 800$, and stop-word count $S = 10$.

4.2. Blind evaluation over text passages

Next, we evaluate the performance of our system over printed strings $\mathcal{Q} = \{Q_1, Q_2, ..., Q_{25}\}$ of $\sim$ 100 characters, and composed of English words. As in Sub-sec. 4.1, each string was rendered in a font F_m and the likelihood of F_n evaluated for $n = [1, 1000]$ as follows (note the lack of prior as here we do not depend on knowledge of the letter corresponding to each character):

$$p(F_n|Q_j) = \prod_{j=1}^{|Q_j|} p(F_n|q_i). \quad (4)$$

The MAP for each font was computed via:

$$p(F_n|\mathcal{Q}) = \frac{1}{\mathcal{Q}} \sum_{j=1} |\mathcal{Q}| \prod_{j=1}^{|Q_j|} p(F_n|q_i). \quad (5)$$

the results of which are presented in the confusion matrix of Fig. 3. Overall MAP performance was 93.4% which is com-

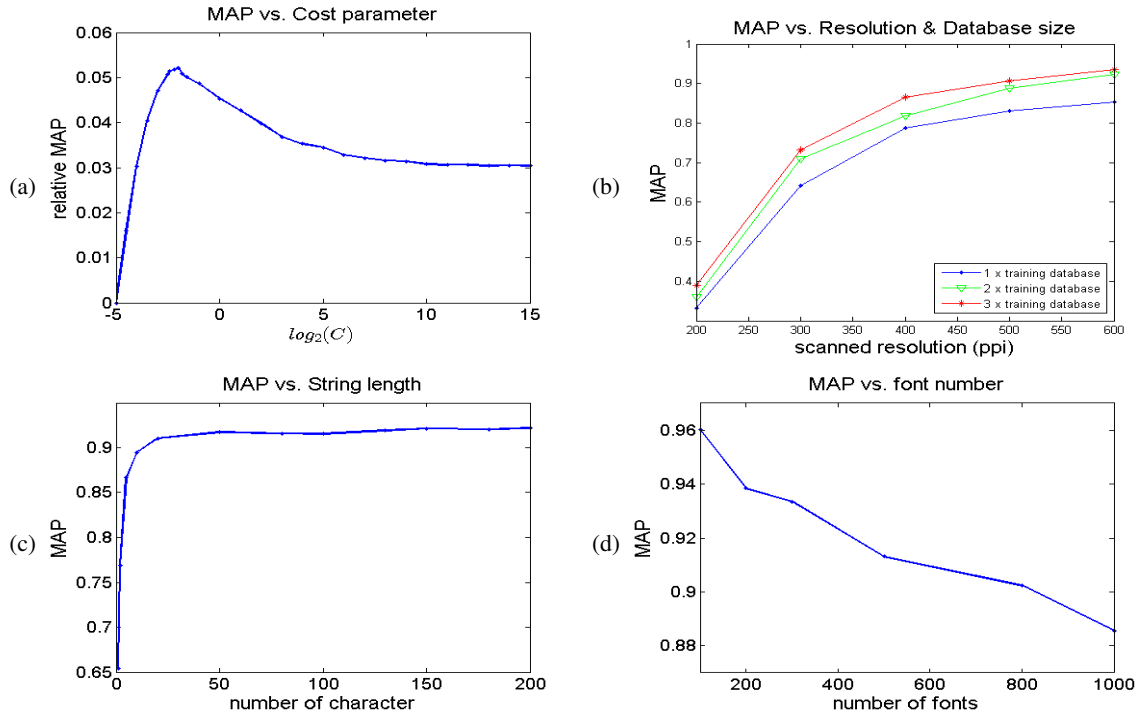

Fig. 4. (a) Importance of tuning the LR meta-parameter (cost) which boosts performance by $\sim 5\%$ (achieved by independent 5-fold cross-validation) versus an untuned baseline. (b) Effect of scan resolution on accuracy (MAP of $\sim 73\%$ at 300dpi vs. $\sim 93\%$ at 600dpi. Triple-scanning the dataset to multiply up training data yields up to 10% performance boost enabling the system to generalise better from noisy scan data. (c) Analysis of performance vs. length of passage tested ($|Q|$). Performance increases significantly as more characters are available to the classifier. (d) Analysing the scalability of the proposed approach with font dataset size (performance drops only 6% over two orders of magnitude increase (single training set).

parable to the prediction of Sub-sec. 4.1 for general English text. System parameters k and S were as before.

A further experiment reported in Fig. 4(c) illustrates that the length of passage tested is positively correlated with accuracy, justifying our joint approach to combining characters for classification (eq. 2). In practice combining just a few characters raises MAP performance to over 90% from 62% for individual characters.

4.3. Parameter selection

In Fig. 2 we explore the parameter space for k (codeword dictionary size) and stopword count S. Several values of k across two orders of magnitude were evaluated, and a peak at $k = 800$ observed revealing an optimal level of quantization for the HOG descriptor space. Similarly, increasingly aggressive culling of stopwords is explored for $k = 800$, revealing an increase of around 2% for removal of the $S = 10$ most common codewords prior to performance reduction due to discarded information. The experimental setup used to tune these parameters was as per Sub-sec.4.1.

The importance of cross-validating data (5-fold) for meta-parameter tuning in LR is emphasised in Fig. 4(a) where gains of $\sim 10\%$ are demonstrated by doing so. Training data has a significant impact on performance. Simply scanning training samples of fonts three times at the chosen resolution, and combining the imagery for training, can yield a further 10% MAP improvement (Fig. 4b). The resolution at which text is scanned for training and test is proportional to the final system accuracy (Fig.4b). A key benefit of our approach is scalability (Fig.4d) we drop only 8% accuracy moving from a dataset of 10 to 1000 fonts. At 10 fonts our performance of 96% (even without tricks such as duplicating training data to boost performance) exceeds of matches the state of the art. However only our approach can do so whilst scaling to a font dataset two orders of magnitude larger.

5. CONCLUSION

We have presented a novel OFR algorithm for font classification from a printed passage comprising one or more characters. We reported 93.4% accuracy in classifying 1000 fonts using HOG features sampled from the glyph perimeter and a hard-assignment BoVW framework with logistic regression (LR) classifier. Our accuracy matches or exceeds that quoted in the state of the art approaches, but over on a diverse font set two orders of magnitude larger than previously attempted (1000 vs. 10s of fonts).

Future work will explore the impact of further scaling font set size and potentially further use of hierarchical methods to both represent and classify this data.

6. REFERENCES

[1] Y. Zhu, T. Tan, and Y. Wang, "Font recognition based on global texture analysis," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 10, no. 23, pp. 1192–1200, 2001.

[2] M-H. Ha and X-D. Tian, "Optical font recognition of chinese characters based on texture features," *Lecture Notes in AI (Proc. of ICMLC)*, vol. 3930, pp. 1005–1013, 2006.

[3] H. Ma and D. Doermann, "Font identification using the grating cell texture operator," in *Proc. SPIE Symp. on Electronic Imaging. Track: Document Recognition and Retrieval XII*. 2005, pp. 119–122, ACM.

[4] A. Sexton, A. Todman, and K. Woodward, "Font recognition using shape-based quad-tree and kd-tree decomposition," in *Proc. Intl. Conf. on Computer Vision and Pattern Recognition and Image Processing*, 2000, pp. 212–215.

[5] A. Zramdini and R. Ingold, "Optical font recognition from projection profiles," *Electronic Publishing*, vol. 6, no. 3, pp. 249–260, 1993.

[6] A. Zramdini and R. Ingold, "Optical font recognition using typographical features," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, no. 20, pp. 877–882, 1998.

[7] A. Satkhozhina, I. Ahmadullin, and J. Allebach, "Optical font recognition using conditional random field," in *Proc. of ACM Symp. on Document Engineering*. 2013, pp. 119–122, ACM.

[8] R. Kharate, S. Jagade, and S. Holambe, "A brief review and survey of segmentation for character recognition," *Intl. Journal of Engineering Sciences*, vol. 1, no. 2, pp. 14–17, 2013.

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proc. Intl. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 886–893.