Recognition of Relatively Small Handwritten Characters, *or* "Size Matters"

Vadim Mazalov and Stephen M. Watt
Department of Computer Science
University of Western Ontario
London, Canada
{*vmazalov, Stephen.Watt*}@*uwo.ca*

Abstract

Shape-based online handwriting recognition suffers on small characters, in which the distortions and variations are often commensurate in size with the characters themselves. This problem is emphasized in settings where characters may have widely different sizes and there is no absolute scale. We propose methods that use size information to adjust shape-based classification to take this phenomenon appropriately into account. These methods may be thought of as a pre-classification in a size-based feature space and are general in nature, avoiding hand-tuned heuristics based on particular characters.

1 Introduction

Size normalization is usually one of the early steps in the recognition of both handwritten and typeset characters, but can also be the source of errors. Characters can have different sizes for two reasons: First, the same symbol may appear in different sizes. An obvious example of this would be footnotes and titles having different sizes from normal text. Other examples would include: place names in map labels having greatly varying size, and the symbols of mathematics, which are smaller when written as superscripts or subscripts or larger when written as n-ary operators. Secondly, different symbols within the same symbol set may have different size relative to each other. For example, a period will be smaller than a lower case "o", which will in turn be smaller than a capital "M". When these two situations are combined, size normalization is a double-edged sword—it is required, but it can also lead to increased ambiguity.

We are motivated by the application of online mathematical handwriting recognition, where digital ink traces are available for symbols that are typically well-separated. Many alphabets are in use simultaneously and there is no dictionary of valid words. Characters will be of greatly varying size and size can vary on a character-by-character basis, rather than word-by-word or sentence-by-sentence. In this setting, we have found it effective to use shape-based classification with orthogonal series representation of the curves traced [3]. It was observed, however, that for very small traces the shape of the curve, when scaled, may be quite arbitrary. In these cases, the original size of a symbol is of high importance.

Recognition systems may adopt *ad hoc* rules to identify characters of unusual size, e.g. commas, long lines, arrows, *etc*. What is lacking in this approach are general principles by which such symbols requiring special treatment may be determined without any *a priori* knowledge of the symbol set, and how special rules to recognize them may be generated.

We propose a two-step processing method with samples being first pre-classified by size, and then recognized by shape. We take advantage of the usual cluster analysis techniques on a space of feature vectors computed from size measures. This may be used in two ways: first to do absolute classification based on size, and second, to do a blended classification, weighting unusually sized samples differently than samples whose size tends to the mean. These ideas can further be extended to literally any symbol set to identify those classes that are more easily separated by size measures than shape measures, e.g. lines, dots, etc.

We present three approaches to classification of small characters based on the relative size of the samples with respect to other symbols in the collection. The size of all samples is expressed in a metric unit, derived from the dataset. In the first method, that can be regarded as a 1-dimensional classifier, a feature is computed from a letter based on its width and height, regulated by a parameter. Given that the parameter is optimized, the method is shown to yield good results for our purposes. This method can be further extended to linear characters, such as "–", "|" with appropriate size measure. The second method is a generalized version of the first technique and it suggests to compute several parameters not only from the size of a letter, but also from its shape, e.g. the area of the convex hull of trace points of the character. Then, one-vs-one support vector machine (SVM) classification becomes a natural way to differentiate classes, if the number of classes is small. However, there are some dictionaries with large set of characters that have identical shape and can only be distinguished by its size. Examples include some capital and low-case characters from the Latin and Greek alphabets, e.g. Kk, Oo, most of the symbols from the Russian alphabet, e.g. Вв, Гг, Дд, Ии, musical notation, and Benesh notation. The third approach is the most robust and suitable for collections with large number of small classes. The distance to the convex hull of coefficients of approximation of coordinate functions [4] is adjusted based on the size of the test sample and the average size of samples in the candidate class. All of the methods are shown to improve significantly the current state of our algorithm with respect to small characters.

The rest of the paper is organized as follows. Some of the preliminaries are given in Section 2. Description of the size-sensitive classification schemes is given in Section 3, including the details of the measurement unit, the 1-dimensional and 3-dimensional classification algorithms, as well as the weight-based method. Experimental setting and results are reported in Section 4. Section 5 concludes the paper.

2 Previous Work

Partially related problems have been studied in the past. A conventional approach to identification of small samples is by comparison with a fixed threshold, expressed in pixels. The adaptive normalization method developed in [7] adjusts the size of a character based on its aspect ratio. In [2] it is proposed to estimate the principal line, and correspondingly the size of symbols, using the pixel count histogram when projected on the vertical axis. Recognition rate of handwritten numerals depending on the size was investigated in [5]. In [10] it is proposed to perform size normalization with Hough transform.

These methods are designed for either processing characters independently or for extraction of information from a set of characters. In contrast, we propose to apply special classification rules to *relatively* small symbols.

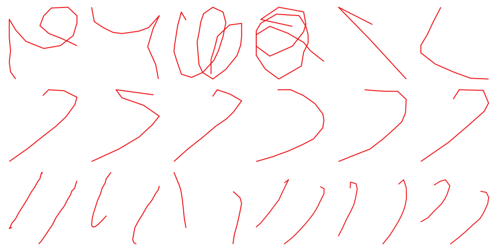

Figure 1. Examples of scaled small characters from the top row to the bottom: period, comma, quotes.

An efficient and accurate technique for online classification of characters has been developed in [3, 4]. In the approach, the feature vector of a character is constructed from coefficients of approximation of the strokes with orthogonal polynomials. Classification is based on the minimal distance to convex hulls of training classes, where each character is represented as a point with coordinates being the coefficients of approximation. Normalization of a sample with respect to size is achieved by normalizing the coefficients vector so that its norm is equal to one. The method was tested on a collection of handwritten samples, and can be as well used as a general purpose algorithm for recognition of two-dimensional patterns. The technique is overall robust, but has a drawback, related to size-normalization – it does not take into account the initial size of a sample. As a result, small samples are scaled to the size of a regular character that leads to incorrect classification. Examples of small samples are shown in Figure 1, where it is easy to observe that, for instance, normalized period can be mis-classified as many other symbols, comma resembles a closing bracket, while quotes are hard to distinguish from "11". Thus, the algorithm requires a robust adaptive size normalization approach.

3 Size-Sensitive Classification Schemes

3.1 The Unit of Measurement

To treat small samples efficiently, one has to identify what the small character is. The size of a small symbol should not be dependent on the device, nor identified as a constant amount of pixels. Instead, the size should be expressed in terms of some properties of the dataset. Similar to the notion of *Ex-typography*, we choose to take the average height of lower-case x as the unit measure, and denote this value as ex, analogous to the *ex* measure in CSS [1]. In other words, ex can be understood as a metric unit for all characters in a database. In this setting, we can separate small classes from other classes based on dynamic size measures.

3.2 1-Dimensional Classification

The algorithm described in this section is the simplest form of a classifier, since only one feature is analyzed – the size of the sample. Despite its simplicity, in the experimental section we show that this technique has very low error in recognition of certain classes due to the dynamic nature of the size measure.

The Size Measure If the size of a character c is analyzed by its bounding box, there are essentially two types of size measures: perimeter-based and area-based. The perimeter-based measure is studied in this section $s(c) = \alpha w(c) + h(c)$, where α is a parameter, $w(c)$ and $h(c)$ are width and height of the bounding box of the character. We empirically find the α that gives the lowest classification error. The area-based feature is considered in Section 3.3.

Classification Consider a dataset with only two classes $\{\circ, \times\}$ that are to be classified with respect to size, and the average size of $\circ$ is less than the average size of $\times$. Let $s_{\{\circ,\times\}}$ be the size threshold that separates the classes. Then a sample from the class $\circ$ ($\times$) is considered to be classified incorrectly, if its size is greater (smaller) than $s_{\{\circ,\times\}}$. We denote with $I_\circ$ ($I_\times$) the set of incorrectly classified samples of $\circ$ ($\times$). Then, the overlap of the classes is computed as

$$D_{\{\circ,\times,s_{\{\circ,\times\}}\}} = \sum_{i \in I_\circ} (s(i) - s_{\{\circ,\times\}}) + \sum_{i \in I_\times} (s_{\{\circ,\times\}} - s(i))$$

The threshold $s_{\{\circ,\times\}}$ that minimizes the overlap can be found in $O(n)$ given that sizes have been computed and stored in a sorted array, where n is the total number of samples in $\circ$ and $\times$, see Algorithm 1 for details. The algorithm can be easily extended to an arbitrary amount of classes.

The classification error is measured as described in Algorithm 2.

3.3 3-Dimensional Classification

In this scheme three features are extracted from characters: the height and the width of the bounding box, and the area of the convex hull of points of the sample, see Figure 2. We test whether these indicators are sufficiently discriminative with an SVM classifier.

Algorithm 1 Find Separating Threshold($S_\circ$, $S_\times$, s)

```
Input: $S_\circ$ – the set of samples of the class $\circ$, $S_\times$ – the set of samples of the class $\times$, $s$ – the array of sizes of samples from both classes, sorted in ascending order.
Output: $s_{\{\circ,\times\}}$.
  Compute differences between consecutive elements of $S$ as $\Delta_i = s[i] - s[i-1], i = 1, .., n$.
  $D_{\{\circ,s[n]\}} \leftarrow 0$
  for all $i = n-1$ to $0$ do
    Compute the overlap for samples of the class $\circ$, if $s[i]$ is the threshold
        $D_{\{\circ,s[i]\}} \leftarrow k_{\{\circ,s[i]\}}\Delta_{i+1} + D_{\{\circ,s[i+1]\}}$
    where $k_{\{\circ,s[i]\}}$ is the number of incorrectly discriminated samples of $\circ$ for the threshold $s[i]$.
  end for
  $D_{\{\times,s[0]\}} \leftarrow 0$
  for all $i = 1$ to $n$ do
    Compute the overlap for samples of the class $\times$, if $s[i]$ is the threshold
        $D_{\{\times,s[i]\}} \leftarrow k_{\{\times,s[i]\}}\Delta_i + D_{\{\times,s[i-1]\}}$
    where $k_{\{\times,s[i]\}}$ is the number of incorrectly discriminated samples of $\times$ for the threshold $s[i]$.
  end for
  for all $i = 0$ to $n$ do
        $D_{\{\circ,\times,s[i]\}} \leftarrow D_{\{\circ,s[i]\}} + D_{\{\times,s[i]\}}$
  end for
  return $\{s[m] \mid D_{\{\circ,\times,s[m]\}} = \min_{i=0..n} D_{\{\circ,\times,s[i]\}}\}$
```

Algorithm 2 classificationError(α)

```
Input: $\alpha$ - the parameter in the size measure.
Output: Classification error.
  For the given $\alpha$: Compute sizes of samples.
  {In 10-fold cross-validation over the dataset}
  for $i = 1$ to $10$ do
    Take the $i$-th training set and find $s_{\{\circ,\times\}}$ with Algorithm 1.
    Test $s_{\{\circ,\times\}}$ with the $i$-th test set. The classification error is reported as the ratio of incorrectly discriminated samples to the total number of samples in the test set.
  end for
  return The average discrimination error over the 10 runs.
```

3.4 Weight-based classification

The letter "." can usually be classified based on its size in ex units. By analyzing sizes of characters in a

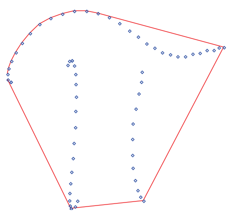

Figure 2. Convex hull of a sample

Algorithm 3 WeightedClassification(x)

Input: x - a test sample.
Output: The result of classification.

```
$s_x \leftarrow$ width($x$) + height($x$)
{Select $k$ nearest neighbours of candidate classes $C_1, ..., C_N$, as described in [4]}
for $i = 1$ to $N$ do
    $d_i \leftarrow D(x, \text{CHNN}_k^i)$
    if $C_i$ is a class of small symbols then
        $d_i \leftarrow (\omega(s_x) + \beta|\omega(\bar{s}_i) - \omega(s_x)|) \cdot d_i$
    end if
end for
return $C_j | d_j = \min_{i=1..N} d_i$
```

dataset, one can obtain the minimal size threshold of samples, other than ".". If the size of a test sample is smaller than the threshold, then it is automatically classified as ".". If the size is greater, the character still can be ".". Therefore, the class of "." is considered in computation of distances, described below.

Unlike ".", other small symbols, such as ",", preserve its initial shape after normalization, even though the letter maybe scaled significantly and appear as a different character. Thus, the shape and size should both be considered in classification. The distance to the small classes is *adjusted* based on the average relative size of samples in the class and the relative size of the test sample

$$D_{adj} = (\omega(s_x) + \beta|\omega(\bar{s}_i) - \omega(s_x)|) \cdot D(x, \text{CHNN}_k^i)$$

where s_x is the relative size of the test sample x (the sum of its width and height), $\bar{s}_i$ is the average relative size of samples in the test class i, β is a parameter, $D/D_{adj}(x, \text{CHNN}_k^i)$ is the distance/adjusted distance from the test sample to the convex hull of k nearest neighbours of the class i [4], where i is one of the small classes. The distance to regular-size classes is computed without the weight adjustment. We take the function $\omega(s)$ to have the form s^γ where γ is a numeric parameter to be evaluated. See Figure 3 for examples of $\omega(s)$. This method is illustrated in Algorithm 3.

Besides their size, small characters can usually be differentiated by positioning, relative to the baseline and mean line. However, we leave that analysis to another recognition layer, responsible for the spatial segmentation of formulas.

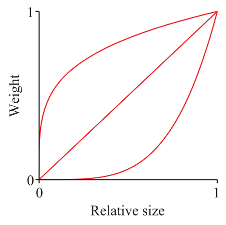

Figure 3. Examples of the weight function depending on the relative size: $\omega(s) = s^{1/4}$, $\omega(s) = s$, and $\omega(s) = s^4$

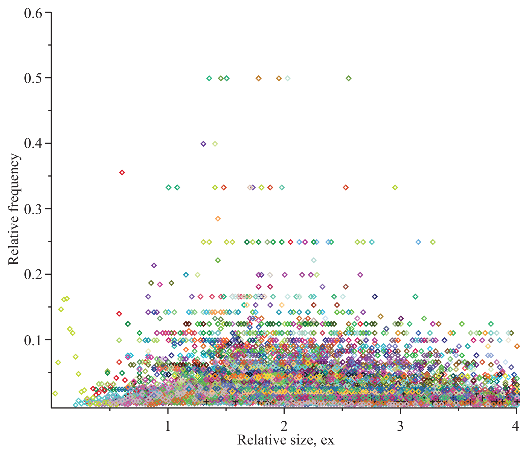

Figure 4. Relative frequency vs relative size for the different classes in the ORCCA dataset

4 Experiments

4.1 Experimental Setting

The experimental dataset is based on the database of handwritten characters, collected at the Ontario Research Centre for Computer Algebra, a subset of the dataset described in [4]. Since the dataset does not contain classes with small characters, we obtained samples "." and ","/"′" by decomposing the following symbols: ":", "$\ddot{a}$", "$\div$", "$\dot{a}$", "$\doteq$","!","$\ldots$","i","j","$\odot$","?",";". Visual examination of the small characters written within the context of another character and the small letters written independently did not reveal significant differences. Therefore, we find this setting adequate.

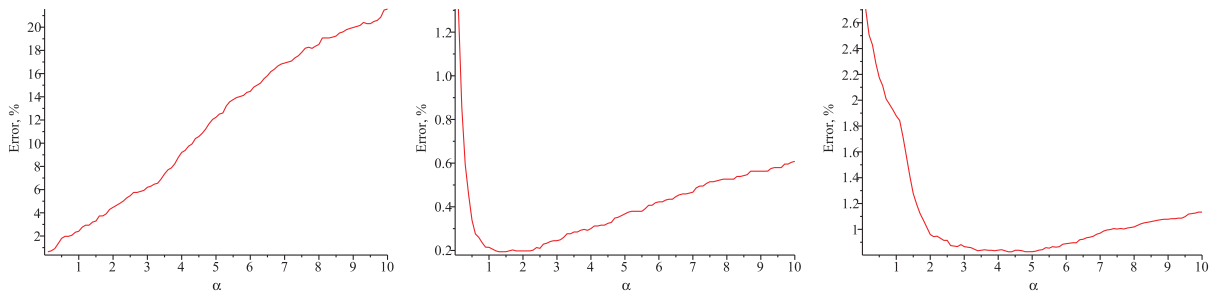

Figure 5. The classification rate depending on α for: "." and "," (left), "." and the rest of the classes (centre), "," and the rest of the classes (right)

Overall, we have collected 803 samples of "." and 315 samples of ","/"'".

The physical size of an ex unit is 823. The relative frequency of sizes of samples, shown in Figure 4, was computed as follows:

1. Split the range of sizes in k intervals: $(s_0, s_1), (s_1, s_2), ..., (s_{k-1}, s_k)$. In the experiments, $k = 40$.
2. The relative frequency on an interval m is found as the ratio of the number n_m of samples in the interval to the total number of samples in the class: $n_m / \sum_{i=1}^{i=k} n_i$.
3. Sizes are computed as the sum of width and height with $\alpha = 1$.

Note, that the most frequent size of "." is $\approx 0.02ex$. Therefore, the value of $0.01ex$ may be interpreted as *thickness* of digital ink and can be used in calligraphy of recognized characters or for beautification of scripts. Another interesting observation is that the frequencies seem to be centered approximately at $ex = 2$, which proves ex being the appropriate unit of measure for this type of analysis.

The recognition experiments were performed in 10-fold cross-validation: each collection has been split randomly in 10 approximately equal parts and the classification rate has been measured 10 times.

4.2 Performance before the Improvement

To estimate the performance of the methods developed in this paper, we first measure recognition of small characters with the algorithm described in [4] and optimized in [8], where 97.6% classification rate was achieved. The recognizer is trained with all samples from our dataset (small and regular) and tested with small samples. The obtained classification error of the small samples is $\approx 17.5\%$, which is significantly higher than the classification error of regular sized characters reported in [8].

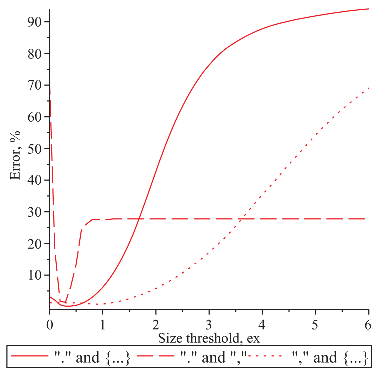

Figure 6. The recognition error depending on the size threshold for $s_{\{".",\{...\}\}}$, $s_{\{".",","\}}$, and $s_{\{",",\{...\}\}}$

4.3 1-Dimensional Classification

In this experiment, all characters are divided in three parts: ".", ",", and the rest of the regular size classes in the dataset, denoted as $\{...\}$. The objective is to find optimal values of α that allow correct pair-wise discrimination between the parts. The recognition error as a function of α is shown in Figure 5. The values of α that yield the lowest classification error between "." and "," (0.6%), "." and $\{...\}$ (0.2%), "," and $\{...\}$ (0.8%) are respectively 0.1, 1.3, 4.4, and the values of the size threshold $s_{\{\circ,\times\}}$ are respectively $0.26ex$, $0.34ex$ and $0.95ex$. The stability of the recognition error depending on the threshold is shown in Figure 6.

4.4 3-Dimensional Classification

These experiments were performed with the SVM-Java [6], a Java implementation of SMO [9] technique

Table 1. Classification error, depending on β and γ

β	0.3	0.6	0.3	0.6	0.9	0.3	0.6	0.9
γ	2.4	2.4	2.7	2.7	2.7	3.0	3.0	3.0
Er.,%	2.75	3.48	2.76	2.94	3.21	2.06	2.23	2.60

for training an SVM. A subset of the collection of regular classes has been considered in this experiment: we randomly selected 1000 samples. The classes of "." and "," remained unchanged. The following respective error rates have been obtained for one-versus-one classification with the linear kernel for the classes "." and ",", "." and {...}, "," and {...}: 2.38%, 1.44%, 4.92%. These results can be further improved by considering alternative kernels.

4.5 Weight-based classification

With optimization of the parameters β and γ, we obtained the classification error, as reported in Table 1. With the best result of 2.06% error, one can observe significant improvement over the original error of 17.5% of the algorithm on small samples.

5 Conclusions and Future Work

We have presented methods to address the large shape variations that can occur in small characters in handwritten samples. When there are only one or two classes which have much smaller characters than the rest, we have found that simple discrimination based on an optimized linear combination of width and height to be very effective. We have shown this can be combined effectively with shape-based methods by weighting shape and size depending on size of typical characters in the classes. We have found that using the area of the convex hull of characters, rather surprisingly, does not improve the accuracy over using a linear combination of width and height.

The presented work does not address differentiation between disconnected segments of a symbol and independent small characters. This is the question of recognition of groups of strokes that can be solved by construction of classification theories and computation of the confidence of each theory. In this paper we have focused on devising general methods for very small characters. In the future, we wish to examine how these ideas can be applied to the automatic identification and pre-classification of very large characters.

References

[1] Cascading style sheets (css) snapshot 2010, May 2011. W3C Working Group.

[2] H. S. Beigi, K. Nathan, G. J. Clary, and J. Subrahmonia. Size normalization in on-line unconstrained handwriting recognition. In *Proc. IEEE Int'l Conf. Acoustics, Speech and Signal Processing*, pages 169–172, 1994.

[3] B. W. Char and S. M. Watt. Representing and characterizing handwritten mathematical symbols through succinct functional approximation. In *Proc. ICDAR*, pages 1198–1202. IEEE Computer Society, 2007.

[4] O. Golubitsky and S. M. Watt. Distance-based classification of handwritten symbols. *International Journal on Document Analysis and Recognition*, 13(2):113–146, 2010.

[5] C. L. He, P. Zhang, J. Dong, C. Y. Suen, and T. D. Bui. The role of size normalization on the recognition rate of handwritten numerals. *The 1st IAPR TC3 NNLPAR*, 1:1–5, 2001.

[6] X. Jiang and H. Yu. SVM-JAVA: A java implementation of the SMO (sequential minimal optimization) for training SVM, 2008.

[7] C.-L. Liu, M. Koga, H. Sako, and H. Fujisawa. Aspect ratio adaptive normalization for handwritten character recognition. In *Proc. of the Third International Conference on Advances in Multimodal Interfaces*, ICMI '00, pages 418–425, London, UK, 2000. Springer-Verlag.

[8] V. Mazalov and S. M. Watt. Improving isolated and in-context classification of handwritten characters. In *Proc. Document Recognition and Retrieval XIX, (DRR XIX)*, San Francisco, California, January 2012.

[9] J. Platt. Sequential minimal optimization: A fast algorithm for training support vector machines. *Advances in Kernel Methods-Support Vector Learning*, 208:98–112, 1999.

[10] A. Rosenthal, J. Hu, and M. Brown. Size and orientation normalization of on-line handwriting using hough transform. In *Proc. of the 1997 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '97) -Volume 4 - Volume 4*, ICASSP '97, pages 3077–, Washington, DC, USA, 1997. IEEE Computer Society.