Part II

Allograph-Level Approach
Chapter 4

Grapheme Clustering for Writer Identification and Verification

Perfect clarity would profit the intellect, but damage the will.

We arrive at truth, not by reason only, but also by the heart.

Blaise Pascal

Abstract

This chapter introduces our allograph-level method for writer identification and verification. The fundamental underpinning of this effective method is the idea of assuming that each writer acts as a stochastic generator of ink-trace fragments, or graphemes. The probability distribution of these simple shapes in a given handwriting sample is characteristic for the writer and is computed using a common codebook of graphemes obtained by clustering. Originally proposed in (Schomaker and Bulacu 2004), the theoretical model that supports this approach is also provided here in its essential aspects. While in other studies (Schomaker and Bulacu 2004, Schomaker et al. 2004), contours were employed to encode the graphemes, the current work explores a complementary shape representation using normalized bitmaps. The most important aim of the present study is to compare three different clustering methods for generating the grapheme codebook: k-means, Kohonen SOM 1D and 2D. Large scale computational experiments show that the proposed allograph-level writer identification method is robust to the underlying shape representation used (whether contours or normalized bitmaps), to the size of codebook used (stable performance for sizes from $10^2$ to $2.5 \times 10^3$) and to the clustering method used to generate the codebook (essentially the same performance was obtained for all three clustering methods).

4.1 Introduction

Research in writer identification and verification has received significant interest in recent years due to its forensic applicability (Zois and Anastassopoulos 2000, Said et al. 2000, Srihari et al. 2002, Schomaker and Bulacu 2004, Schlapbach and Bunke 2004, Bensefia et al. 2005b). In writer identification, for a query sample of unknown author, a one-to-many search is performed over a large database with handwritten samples of
known authorship. The system retrieves a reduced list of candidates containing the samples most similar to the query in terms of individual handwriting style. This reduced list will be further scrutinized by the forensic expert in order to take the final decision regarding the identity of the author for the questioned sample. Writer identification is therefore possible only if there exist previous samples of handwriting by that person enrolled in the forensic database. In **writer verification**, a one-to-one comparison is performed with an automatic decision whether or not the two compared samples are written by the same person. The decidability of this problem reflects the nature of handwriting individuality (Srihari et al. 2002) and also the discrimination power of the features used for the writer verification task.

While **texture-level** methods that use directional PDFs (capturing slant, size, curvature, regularity) prove to be very efficient as seen in the previous chapters, they must be complemented by **allograph-level**, i.e. character-shape based approaches in order to obtain adequate and robust results. The discriminatory power of singular characters is analyzed in (Srihari et al. 2003) and (Zhang et al. 2003), under the assumption that the individual characters are perfectly separated and labeled using some form of human intervention. Other new results also show that writer-specialized handwriting recognizers can be used for writer identification and verification (Schlapbach and Bunke 2004).

In recent work, Schomaker has proposed an effective writer identification method in which the writer is assumed to act as a stochastic generator of ink-blob shapes, or graphemes (Schomaker and Bulacu 2004). The probability distribution of grapheme usage is characteristic of each writer and is computed using a common codebook obtained by clustering. A brief account of the underlying theoretical model of this approach will be given in the next section of this chapter. This theoretically founded approach was initially applied to isolated uppercase handwriting (Schomaker and Bulacu 2004) and later it was extended to lowercase cursive handwriting by using a segmentation method (Schomaker et al. 2004).

In these previous studies, we have used contours for shape representation and a 2D Kohonen self-organizing map (KSOM) for generating the grapheme codebook. While contours posses definite advantages for shape matching, they are nevertheless susceptible to problems regarding the starting point, open/closed loops or the presence of multiple inner contours. On the other hand, pixel-based representations can be more robustly extracted from the handwriting images, but the matching process becomes more vulnerable in this case, e.g., due to quantization in rescaling. The first purpose of this work is to explore the use of normalized bitmaps as the underlying shape representation. In this respect, our study comes closest to the work reported in (Bensefia et al. 2002, Bensefia et al. 2003, Bensefia et al. 2004) where an information-retrieval framework is used for writer identification. In contrast, our approach uses explicit probability distributions constructed on the basis of the shape codebooks to characterize writer individuality.
4.2 Theoretical model

The process of handwriting consists of a concatenation of ballistic strokes bounded by points of high curvature in the pen-tip trajectory (Schomaker 1991). Curved shapes are realized by differential timing in the movements of the wrist and finger subsystems (Schomaker et al. 1989). Handwriting is not a feedback process governed by peripheral environment factors. As a consequence of neural and neuro-muscular propagation delays, handwriting would be too slow if based upon continuous feedback (Schomaker 1991). Rather, the brain is planning series of ballistic strokes ahead in time in a feed-forward manner (Plamondon and Maarse 1989, Plamondon and Guerfali 1998). A character is assumed to be produced by a "motor program" (Schmidt 1975) that can be triggered to produce a pen-tip movement yielding the character shape on paper (Schomaker et al. 1989).

The allographic shape variations reflecting the character forms engrained in the motor memory of the writer allows for very effective writer identification and verification. Schomaker proposed a theoretical model and provided an experimental evaluation for this allograph-level approach to writer identification (Schomaker and Bulacu 2004). Here we will present the main aspects of this model; more details can be found in (Schomaker and Bulacu 2004).

Assume there exists a finite list \( S \) of allographs for a given alphabet \( L \). Each allograph \( s_{il} \) is considered to be the \( i \)th allowable shape (style) variation of a letter \( l \in L \) which should in principle be legible at the receiving end of the writer-reader communication line (Kondo and Attachoo 1986). The source of allographic variation may be located in teaching methods and individual preferences. The human writer is thus considered to be a pattern generator, stochastically selecting each allograph shape \( s_{il} \) when a letter \( l \) is about to be written (Shannon 1948). It is assumed that the probability density function \( p_w(S) \), i.e., the probability of allographs being emitted by writer \( w \), will be informative in the identification of writer \( w \) if it holds that

\[
  w \neq v \implies p_w(S) \neq p_v(S) \tag{4.1}
\]

where \( w \) and \( v \) denote writers, \( S \) is a common allograph codebook and \( p(.) \) represents
the discrete PDF for allograph emission. This (i.e. eq. 4.1) will be realizable if for hand-written samples \( u \) emitted by \( w \) and characterized by

\[ \bar{x}_{wu} = p_{wu}(S) \]  

and assuming that the sample \( u \) is representative

\[ \bar{x}_{wu} \approx p_w(S) \]  

it holds that

\[ \forall a, b, c, w, v \neq w : \Delta(\bar{x}_{wa}, \bar{x}_{wb}) < \Delta(\bar{x}_{wa}, \bar{x}_{vc}) \]  

where \( \Delta \) is an appropriate distance function on PDFs \( \bar{x} \), \( v \) and \( w \) denote writers, as before, and \( a, b, c \) are handwriting-sample identifiers. Equation 4.4 states that, in feature space, the distance between any two samples of the same writer is smaller than the distance between any two samples by different writers. In ideal circumstances, this relation would always hold, leading to perfect writer identification. Note that in this model (eq. 4.1), the implication is unidirectional: in case of forged handwriting, \( p_w(S) \) does not equal \( p_v(S) \) but writer \( w \) imposes as \( v \) (\( w = v \)).

A problem at this point is that an exhaustive list \( S \) of allographs for a particular script and alphabet is difficult to obtain in order to implement this stochastic allograph-emission model. Clustering of character shapes with a known letter label is possible and has been realized (Vuurpijl and Schomaker 1997). However, the amount of handwritten image data for which no character ground truth exists vastly exceeds the size of commercial and academic training sets which are labeled at the level of individual characters. At this point in time, a commonly accepted list of handwritten allographs (and their commonly accepted names, e.g., in Latin, such as in the classification of species in the field of biology) does not exist, as yet. In this respect, it is noteworthy that for machine-print fonts, with their minute shape differences in comparison to handwriting variation, named font categories exist (e.g., Times-Roman, Helvetica, etc.), whereas we do not use generally agreed names for handwritten character families.

Therefore, it would be conducive to use an approach which avoids expensive character labeling at both training and operational stages. Unfortunately, automatic character segmentation in handwriting cannot be performed reliably. As a consequence, we will use a generic and imperfect segmentation method that generates ink fragments (graphemes) that do not overlap complete characters. These glyphs are usually sub-allographic parts of characters and have the advantage that they can be extracted reliably and in a non-parametric manner from a handwritten sample.

Despite not being complete characters, the graphemes generated by the heuristic over-segmentation method are nevertheless informative of the allographic character
variant of which they are a component. We will show that such sub-allographic script fragments are usable for writer identification. We will use the empirical distribution of grapheme occurrence as an approximation for the writer-specific allograph-emission probability.

We will assume a finite set or codebook $C$ of sub- or supra-allographic shapes and we will estimate and use $p_w(C)$ as the writer descriptor in our identification and verification tests. In the next sections, we will describe the construction of the grapheme codebook $C$, the computation of an estimate of the writer-specific pattern-emission PDF $p_w(C)$, and an appropriate distance function $\Delta$ for these PDFs.

4.3 Datasets

The writer identification and verification study reported here was performed using two datasets: Firemaker and ImUnipen.

For the tests carried out in this chapter, we use pages 1, 2 and 4 from the Firemaker set (Schomaker and Vuurpijl 2000) comprising handwriting collected from 250 Dutch subjects, predominantly students. Page 1 contains 5 paragraphs of copied text in lower-
The ImUnipen set contains handwriting from 215 subjects, 2 samples per writer. The images were derived from the Unipen database (Guyon et al. 1994) of on-line handwriting. The time sequences of coordinates were transformed to simulated 300 dpi images using a Bresenham line generator and an appropriate brushing function. The samples contain lowercase handwriting with varying text content and amount of ink. The dataset was divided in two parts: 65 writers (130 samples) were used for training the grapheme codebook and the rest of 150 writers (300 samples) were used for testing.
4.4 Segmentation method

In free-style cursive handwriting, connected-components may encompass several characters or syllables. A segmentation method that isolates individual characters remains an elusive goal for handwriting research. Nevertheless, several heuristics can be applied, yielding graphemes (sub- or supra-allographic fragments) that may or may not overlap a complete character. While this represents a fundamental problem for handwriting recognition, the fraglets generated by the segmentation procedure can still be effectively used for writer identification. The essential idea is that the ensemble of these simple graphemes still manages to capture the shape details of the allographs emitted by the writer.

We segment handwriting at the minima in the lower contour with the added condition that the distance to the upper contour is in the order of the ink-trace width.
4. Grapheme Clustering for Writer Identification and Verification

Figure 4.4: Segmentation at the minima in the lower contour that are proximal to the upper contour.

(see Fig. 4.4). For contour extraction we use Moore’s algorithm. After segmentation, graphemes are extracted as connected components, followed by a size normalization to 30x30 pixel bitmaps, preserving the aspect ratio of the original pattern.

4.5 Grapheme codebook generation

A number of 130 samples corresponding to 65 writers have been taken from the ImU-nipen dataset. The graphemes have been extracted from these samples using the described procedure yielding a training set containing a total of 41k patterns (normalized bitmaps).

Three clustering methods will be used to generate the grapheme codebook: k-means, Kohonen SOM 1D and 2D. We use standard implementations of these methods. Complete and clear descriptions of the algorithms can be found in references (Kohonen 1988, Duda et al. 2001).

The size of the codebook (the number of clusters used) yielding optimal performance is an important parameter in our method. In the experiments, we will explore a large range of codebook sizes. This will allow a thorough comparison of the considered clustering algorithms.

Figures 4.1, 4.2 and 4.3 show examples of shape codebooks that have been obtained by training using each of the three clustering methods. The two grapheme codebooks obtained using Kohonen training show spatial order, while the one obtained using k-means is “disorderly”. The ksom1D codebook must be understood as a long linear string of shapes and gradual transitions can be observed if the map is “read” in left-to-right top-to-bottom order. The ksom2D codebook shows a clear bidimensional organization.
4.6 Computing writer-specific grapheme-emission PDFs

The writer is considered to be characterized by a stochastic pattern generator, producing a family of basic shapes (Schomaker and Bulacu 2004). The individual shape emission probability is computed by building a histogram in which one bin is allocated to every grapheme in the codebook.

For every sample \( i \) of handwriting, the graphemes are extracted using the segmentation / connected-component-detection / size-normalization procedure described before. For every grapheme \( g \) in the sample, the nearest codebook prototype \( j \) (the winner) is found using the Euclidean distance and this occurrence is counted into the corresponding histogram bin:

\[
j = \text{argmin}_n [ \text{dist}(g, C_n) ], \quad h_{ij} \leftarrow h_{ij} + 1
\]

where \( n \) is an index that runs over the shapes in the codebook \( C \). In the end, the his-

Figure 4.5: Density plots of grapheme emission PDFs computed using a ksom2D codebook. Two samples A and B from the same writer (left panel) yield a much larger common density than samples from different writers (right panel). The common density resulting from the overlap between the two sample PDFs is depicted in the third column ('Common').
4. Grapheme Clustering for Writer Identification and Verification

togram $h_i$ is normalized to a probability distribution function $p_i$ that sums up to 1. This PDF (see Fig. 4.5) is the writer descriptor used for identification and verification.

4.7 Results

We performed large scale computational experiments to compare the three clustering methods over a large range of codebook sizes. The number of clusters used was varied from 9 (3x3) to 2500 (50x50). A number of 200 epochs have been used for training the Kohonen SOMs. Computations have been performed on a Beowulf high-performance Linux cluster with 1.7GHz/0.5GB nodes. Training times for codebooks of size 400: k-means - 1 hrs, ksom1D - 10 hrs, ksom2D - 17 hrs. Computation times for the grapheme emission PDF on codebooks of size 400: k-means - 0.5 s / sample, ksom1D - 1.5 s / sample, ksom2D - 3.1 s / sample. These computation times were obtained using the ‘gcc’ compiler with optimization for single-precision floating-point calculations. The total computation time used in the experiments amounts to approx. 800 CPU hrs.

4.7.1 Writer identification

Writer identification results are computed using nearest-neighbor classification (Cover and Hart 1967) in a leave-one-out strategy. For a query sample $q$, the distances to all the other samples $i \neq q$ are computed. Then all the samples $i$ are ordered in a sorted hit list with increasing distance to the query $q$ (Press et al. 1992). Ideally, the first ranked sample (Top 1) should be the pair sample produced by the same writer (in all our experiments there are 2 samples per writer).

An appropriate dissimilarity measure between the grapheme PDFs is the $\chi^2$ distance (Press et al. 1992):

$$\chi^2_{q_i} = \sum_{n=1}^{k} \frac{(p_{qn} - p_{in})^2}{p_{qn} + p_{in}}$$  \hspace{1cm} (4.6)

where $p$ are entries in the PDF, $n$ is the bin index and $k$ is the number of bins in the PDF (equal to the size of the grapheme codebook). In our experiments, $\chi^2$ outperformed other distance measures: Hamming, Euclid, Minkowski order 3, Bhattacharya.

We point out that our writer identification results are realistic and rather conservative because we do not make a separation between a training set and a test set. Keeping all the data in one batch makes the testing conditions actually more difficult and realistic, with more distractors: not 1, but 2 per false writer and only one correct hit.

Figures 4.6, 4.7 and 4.8 show our results obtained on the experimental datasets. Writer identification performance (Top-1 and Top-10) reaches a plateau for codebook
sizes larger than about 100 (10x10) shapes. More remarkable is the fact that the same performance is achieved by all three clustering methods. Table 4.1 gives numerical results for codebooks of size 400 which was chosen as an anchor point.

The writer identification performance is stable over a very large range of codebook sizes, from 100 to 2500. Therefore the codebook size does not represent a critical parameter for our allograph-level writer identification approach. Additionally, the three clustering algorithms used to generate the shape codebook yielded the same level of performance. We are thus confident that our results are robust and reproducible.

The lower performance obtained on the Firemaker uppercase dataset can be explained by two factors: the amount of handwriting in these samples is very reduced (only one paragraph of 100-150 characters) and the codebooks have been generated based on samples that contain almost exclusively lowercase (cursive) handwriting. Nevertheless, the overall performance levels achieved on lowercase and uppercase are quite comparable. In the previous chapter, using edge-based directional features under the condition that approximately the same amount of ink is present in all samples, the performance level achieved on lowercase and uppercase was roughly the same (Bulacu and Schomaker 2003). Here again, the empirical results contradict the intuition that writer identification is more effective on lowercase rather than uppercase handwriting.
The slightly higher performance obtained on ImUnipen is due to the smaller number of writers contained in the dataset.

The writer identification results presented here are in the same ballpark as the ones we reported in a previous study using contours for shape representation and Kohonen 2D for codebook training (Schomaker et al. 2004). This constitutes additional evidence regarding the robustness of the proposed method of using grapheme emission PDFs for writer identification.

4.7.2 Writer verification

In the writer verification task, the distance $\xi$ between two given handwriting samples is computed using the grapheme PDFs. Distances up to a predefined decision threshold $T$ are deemed sufficiently low for considering that the two samples have been written by the same person. Beyond $T$, the samples are considered to have been written by different persons. Two types of error are possible: falsely accepting (FA) that two samples are written by the same person when in fact this is not true or falsely rejecting (FR) that two samples are written by the same person when in fact this is the case. The associated error rates are FAR and FRR. In a scenario in which a suspect must be found in a stream
of documents, FAR becomes false alarm rate, while FRR becomes miss rate. These error rates can be empirically computed by integrating up-to/from the decision threshold $T$ the probability distribution of distances between samples written by the same person $P_S(\xi)$ and the probability distribution of distances between samples written by different persons $P_D(\xi)$:

$$\text{FAR} = \int_0^T P_D(\xi) \, d\xi$$  \hspace{1cm} (4.7)

$$\text{FRR} = \int_T^\infty P_S(\xi) \, d\xi.$$  \hspace{1cm} (4.8)

By varying the threshold $T$ a Receiver Operating Characteristic (ROC) curve is obtained that illustrates the inevitable trade-off between the two error rates. The Equal Error Rate (EER) corresponds to the point on the ROC curve where FAR = FRR and it quantifies in a single number the writer verification performance.

For the Firemaker dataset, $P_S(\xi)$ has been constructed using the 250 same-writer distances, while $P_D(\xi)$ has been constructed using all the $C_{250}^2 - 250 = 124500$ different-writer distances arising in the dataset. Similarly for ImUnipen. In figures 4.6, 4.7 and
4. Grapheme Clustering for Writer Identification and Verification

Table 4.1: Writer identification and verification accuracies (percentages) for codebooks of size 400 (20x20). The same writer identification and verification performance is achieved by all three clustering methods. The performance levels are consistent across the three datasets.

<table>
<thead>
<tr>
<th>Dataset / Method</th>
<th>Top 1</th>
<th>kmeans</th>
<th>ksom1D</th>
<th>ksom2D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Firemaker lowercase</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(250 writers)</td>
<td>Top 1</td>
<td>75.3</td>
<td>75.3</td>
<td>78.1</td>
</tr>
<tr>
<td></td>
<td>Top 10</td>
<td>91.8</td>
<td>92.2</td>
<td>92.6</td>
</tr>
<tr>
<td></td>
<td>EER</td>
<td>5.7</td>
<td>5.4</td>
<td>5.3</td>
</tr>
<tr>
<td>Firemaker uppercase</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(250 writers)</td>
<td>Top 1</td>
<td>64.7</td>
<td>63.6</td>
<td>64.9</td>
</tr>
<tr>
<td></td>
<td>Top 10</td>
<td>91.6</td>
<td>90.6</td>
<td>93.2</td>
</tr>
<tr>
<td></td>
<td>EER</td>
<td>8.0</td>
<td>8.2</td>
<td>9.2</td>
</tr>
<tr>
<td>ImUnipen</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>(150 writers)</td>
<td>Top 1</td>
<td>77.7</td>
<td>79.0</td>
<td>76.3</td>
</tr>
<tr>
<td></td>
<td>Top 10</td>
<td>92.7</td>
<td>89.3</td>
<td>91.3</td>
</tr>
<tr>
<td></td>
<td>EER</td>
<td>14.7</td>
<td>15.0</td>
<td>14.7</td>
</tr>
</tbody>
</table>

For Firemaker uppercase, the EER hovers around 8%. For Firemaker lowercase, the EER reaches a minimum of about 3% for a codebook size of 100 and increases to about 7% for larger codebooks. A similar increase in the EER for larger codebooks can be seen also for the ImUnipen set, from 8% (codebook with 9 shapes) to 14% (codebooks with $10^3$ shapes). This effect can be explained considering that, as the codebook size increases, the grapheme emission PDFs reside in increasingly higher dimensional spaces that progressively become less and less populated. The distances between the individual handwriting samples increase in relative terms. As a result it becomes gradually more difficult to find a unique threshold distance that separates the sample pairs written by the same person from those written by different persons. Clearly, an individualized threshold is needed that depends on the variability in feature space of the handwriting belonging to that particular person. However estimating this within-writer variability using a limited amount of handwritten material is a difficult problem that requires further research. The described dimensionality problem does not significantly affect the distance rankings with respect to a chosen sample and consequently writer identification performance remains essentially stable over a large range of codebook sizes. A slight decrease in the writer identification performance with increasing codebook size can however be noticed in Fig. 4.6.

We must point out that the essence of the proposed method does not consist in an exhaustive enumeration of all possible allographic part shapes. Rather, the grapheme codebook spans up a shape space by providing a set of nearest-neighbor attractors for
the ink fraglets extracted from a given handwritten sample. The three clustering methods considered in this chapter seem to perform this task equally well.

4.8 Conclusions

The use of grapheme emission PDFs in writer identification and verification yields valuable results. Ultimately, writing style is determined by allographic shape variations and small style elements which are present within a character are the result of the writer’s physiological make up as well as education and personal preference. The proposed method proves to be robust to the underlying shape representation used (whether contours or normalized bitmaps), to the size of codebook used (stable performance for sizes from $10^2$ to $2.5 \times 10^3$) and to the clustering method used to generate the codebook (essentially the same performance was obtained for k-means, ksom1D and ksom2D).

In the next chapter of the thesis, we will combine the texture-level and allograph-level approaches to improve the performance and robustness of our writer identification and verification system. We will also extend our experiments to bigger datasets containing more writers.