Identifying Features via Homotopy on Handwritten Mathematical Symbols

Rui Hu and Stephen M. Watt
Computer Science Department
University of Western Ontario
London, Canada N6A 5B7
rhu8@uwo.ca, Stephen.Watt@uwo.ca

***Abstract*—In handwritten mathematics, it is common to have characters in various sizes and for writing not to follow simple baselines. For example, subscripts and superscripts appear relatively smaller than normal text and are written slightly below or above it. Rather than use the location, features and size to identify the character, it may be more effective to do the reverse — to use knowledge about specific characters to determine baseline, size, etc. In this approach, it is necessary to find the location of certain expected features that are determined by particular points. In earlier work, we have presented a method to derive the determining points for a new instance of a symbol from those on an average model for each symbol type. For those characters that are significantly different from the average instance, one can use a numerical homotopy between the average instance and the target character, and apply the determining point algorithm at each step. The present article studies the factors to be taken into account in performing such homotopies. We examine two strategies for possible starting points for the homotopy, and we examine the relation between the distance and the number of steps required. The first starting point strategy performs a homotopy from the average of samples of the same type. The second strategy uses a homotopy from the nearest neighbour with known determining points. Our experimental results show a useful relation between the homotopy distance and the number of steps usually required and improved strategies to find determining points for poorly written characters.**

I. Introduction

Mathematical handwriting recognition differs from natural language handwriting recognition in many ways [1], [2]. Symbols are taken from many alphabets, for example, and are written in a two dimensional layout in various sizes with layout and size carrying meaning. The vocabulary of different symbols is larger than in western alphabets, and there are more distinct types of strokes than in East Asian ideographs. In addition, there is no fixed dictionary of words to help with disambiguation [3]. On the other hand, characters tend to be well separated. One of the problems arising from these differences is that baseline estimation is more difficult and may not be used reliably to disambiguate characters. Subscripts and superscripts appear relatively smaller than normal text and are written slightly below or above it. Moreover, these subscripts and superscripts may themselves have subscripts or superscripts. Such notation makes the analysis of the spatial relationships between symbols challenging as it introduces various ambiguities. For example, whether a particular symbol is a lower case “p” or an upper case “P” makes the difference between a subscripted p_q or the juxtaposed Pq.

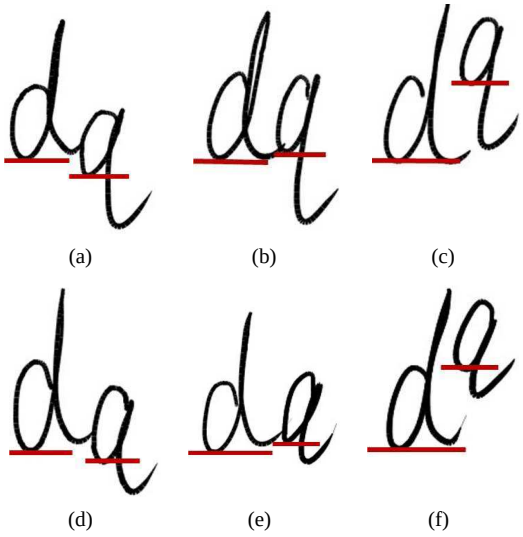
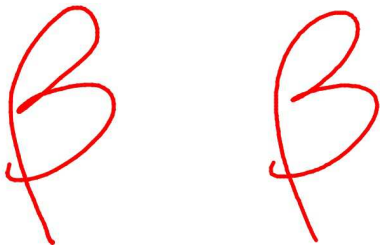


Fig. 1: Baselines: (a) d_q (b) dq (c) d^q (d) d_q (e) dq (f) d^q

In earlier work [4] we have explored the idea that it may be more desirable to identify the possible local baselines in a formula, as well as other metric lines and sizes, from features on individual symbols, rather than the other way around. The features that can be used to do this are typically determined by points appearing at symbol-dependent locations. For example, if a symbol is a lower case “p”, then the baseline is determined by the lowest point in the bowl (loop), but if it is an upper case “P”, then the baseline is determined by the lowest point of the stem. In this article, we refer to such a point, one that determines the height of a metric line, as a *determining point*. Knowing the locations of the determining points can help us identify the size and spatial relationships of symbols and consequently use them in formula recognition. For example, one can use the determining points to help resolve the juxtaposition ambiguity problem which commonly exists in mathematical handwriting recognition. This problem arises when symbols that are next to each other are written in different sizes and at different height. Figure 1 shows an example. Note that to determine the relative position, it is definitely not sufficient to compare the baselines of the symbols’ bounding boxes. This is clearly seen Figure 1(c). Similarly, having an imputed baseline determined by symbols (such as at 50% height for “q”) would mis-treat either Figure 1(b) or 1(e). We thus find it is important to locate and use the

(a) (b)

Fig. 2: Symbol ambiguity: (a) Pq or p_q (b) pq or $P9$



(a) (b)

Fig. 3: Approximation using Legendre-Sobolev series. (a) Original. (b) Degree 12 approximation with $\mu = 1/8$.

symbol's determining points. Figure 2 shows another example of the juxtaposition problem caused by variance of baselines.

We have previously addressed the problem of how to find determining points automatically [4]. Our approach was to represent symbols as approximating polynomial curves, $(x(s), y(s))$, parameterized by arc length and to specify determining points by their type and arc length location on a model character. The determining points on new samples are then identified by finding local minimization or maximization of $y(s)$ using the model points' location as an initial estimate. The previous article showed that, for characters that are significantly different from the average instance, one can use a numerical homotopy between the model instance and the target character, and apply the local extremum-finding algorithm at each step.

The present article extends the earlier work by addressing several questions related to the choice of homotopy method and the efficiency implications of those choices. We concentrate on two strategies: The first performs a homotopy from the computed and hand-annotated average symbol the class. The second uses a homotopy from the nearest neighbour in the class with known determining points, the determining points of the nearest neighbour having been derived automatically from other labelled points of the same class.

The questions we ask are the following:

- To find the determining points of a new sample, what are the differences between starting the homotopy from the average symbol of the class versus starting from the nearest labelled neighbour?
- What is the relationship between the distance from the new sample to the model point and the number of homotopy steps required?
- How can we best use the results to find the determining points in new samples?

The remainder of this article is organized as follows. Section II describes related work. Section III recalls how to approximate digital ink traces using functional approximation and introduces a few types of determining points of interest. Section IV presents the algorithm to identify determining points using homotopy strategies. Section V presents the experimental results and suggests improved strategies to find determining points in poorly written characters. In Section VI we conclude the article.

II. Related Work

Several related problems have been studied in the past. One of the early attempts is the projection method [5], [6], which accepted binary images as input and counted the black pixels line by line. The baseline was then identified by finding the line with the maximum number of black pixels. The method is not entirely reliable. There are cases where the method fails to estimate the baseline locations. This is a concern for our application where there is often only one or a few characters with the same local baseline.

A more advanced method was presented by Pechwitz and Märgner [7] in 2002, based on polygonally approximated symbol skeleton. The method was able to extract certain features from skeletons and then estimate the location of baselines. However, this method is limited to a specific language and cannot be used to detect other metric lines.

In 2010, Infante Velázquez [8] developed a tool to locate determining points in handwritten characters represented in InkML [9] through manual annotation. This tool allowed recording each determining point with absolute coordinates. As the sampling rate and resolution vary between different vendors and over generations of technology, such representation is, however, not device-independent. Similar problems exist in [10].

Zanibbi *et al.* [11] proposed a technique that can gradually translate and scale individual symbols to closely approximate their relative positions and sizes in a corresponding typeset formula. As this technique used bounding boxes to detect baseline locations, it may lead to troubles with vertical placement and scale. For example, it fails to distinguish between "$x2$" and "x^2".

Hu and Watt [12] later described an algorithm that can find those special points that determine the shape of a character. But that approach was unable to capture the geometric meaning of each determining point and therefore did not provide sufficient information to calculate desired symbol metrics, such as the baseline location.

Harouni *et al.* [13] presented a method to find determining points in handwritten Arabic characters. It first divided each ink stroke into pieces, in each of which the local extremum was computed. The points that achieved the local extremum were later defined as the determining points. However, this method is not suitable for mathematics in that it requires extra effort to split strokes and may generate undesired determining points that lack geometric meaning.

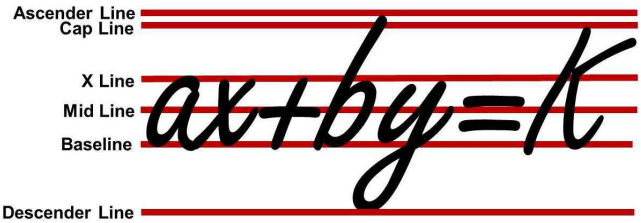


Fig. 4: An example to illustrate the concepts of metric lines.

(a) (b)

Fig. 5: (a) Multiple samples. (b) Average symbol.

III. Preliminaries

A. Orthogonal Series Approximation

Digital ink curves are typically represented as series of points over time, each of which contains x and y values in a rectangular coordinate system. The time interval and resolution vary between hardware vendors and devices from different hardware will usually produce different series of points for the same characters. As technology evolves, this means that resolution-dependent algorithms cannot directly use archived data. In this case, datasets must be "re-sampled" (i.e. interpolated), which introduces its own problems.

To make the representation robust against changes in hardware, we represent handwritten symbols in the space of coefficients of a functional approximation. This approach has been used in earlier work [14]–[17]. Following this approach, we consider an ink trace as a segment of a plane curve $(x(s), y(s))$, parameterized by Euclidean arc length

$$\mathrm{d}s^2 = \mathrm{d}x^2 + \mathrm{d}y^2.$$

The arc length has been found to be the most robust parameterization in most cases. It also makes intuitive sense since it gives curves that look the same the same parameterization [14]. Given a digital ink trace, we represent it using an approximation in a finite dimensional function space

$$x(s) \approx \sum_{i=0}^{d} x_i B_i(s) \qquad y(s) \approx \sum_{i=0}^{d} y_i B_i(s),$$

where $B_i(s)$ are orthogonal basis polynomials, e.g. Chebyshev, Legendre or Legendre-Sobolev polynomials. By choosing appropriate degree d and basis polynomials B_i, $i = 0, ..., d$, the approximating curve can, under some modest assumptions, be made as close as desired to the original trace. We find Legendre-Sobolev polynomials, orthogonal with respect to the inner product

$$\langle f, g \rangle = \int f(s)g(s)\mathrm{d}s + \mu \int f'(s)g'(s)\mathrm{d}s,$$

to be well suited since they also keep the derivatives close. An example of using these polynomials in approximation is shown in Figure 3. If a symbol has multiple strokes, we join consecutive strokes by concatenating the point series. After arc-length normalization, we may represent the digital ink trace, or symbol, as the vector of coefficients of the approximation $(x_0, ..., x_d, y_0, ..., y_d)$. We may standardize the location and size of the character by setting x_0, y_0 to 0 and the norm of the vector to 1.

B. Determining Points

We borrow the idea of character metrics from typography, where a number of determining points are identified to measure the metrics of different font families. These determining points have locations that vary from symbol to symbol, but typically occur where parts of the symbols touch certain invisible horizontal lines. In this article, we focus on symbols in European alphabets and consider six types of metric lines. These are the *baseline*, *x line*, *mid line*, *ascender line*, *cap line* and *descender line*, as shown in Figure 4.

IV. Algorithm

In Section III-A, we have explained how to represent handwritten samples using coefficients of a functional approximation. This representation is device-independent, allowing us to focus on finding determining points without worrying about device-dependency. It also enables various symbolic-numeric polynomial algorithms to conduct useful analysis on the handwritten mathematical samples [14].

Our algorithm to find determining points on a sample is based on the observation that samples of the same class typically have the same features. We can specify the location of a determining point by the value of the normalized arc length parameter at which it occurs on a model symbol. Although the precise locations of the determining points on a new sample will be different, we expect them to occur near these parameter values. To find precise locations of the determining points on the new sample, we can follow the ink trace, starting from the approximate locations, until local vertical extrema of the right type (minimum or maximum) is found.

More specifically, to detect the determining points of samples, we first choose a reference symbol for each class. We then annotate the reference symbol, identifying the locations of determining points of interest. As the computation is based on curves $(x(s)), y(s))$ parameterized by normalized arc length s, the locations of the determining points are recorded by their s values in the interval $[0, 1]$. We can then compute determining points in target samples, starting from the location of the corresponding determining points in the reference symbol. Each determining point of the sample can then be identified by finding the extremum of the polynomial $y(s)$ near the starting point. This can be achieved using any one of a number of numerical methods. In our implementation, we applied Newtons method to solve $y'(s) = 0$. These steps are shown in Algorithm 1, which is similar to that of [4]. The difference is that article always uses the average instance of each class as the reference symbol, but here we allow other suitable reference symbols. This allows us to investigate the effect of choosing different starting points.

Algorithm 1: LocateDeterminingPoints

Input : A, the coefficient vector of a reference symbol.

$D_A = [(s_1, T_1, K_1), \ldots (s_n, T_n, K_n)]$, a vector of determining points. For each, the position is given as arc length s_i on the curve of A, the value T_i states which type of metric line is being defined, and the value K_i states whether the metric line is given by a local minimum or local maximum at $y_A(s_i)$.

S, the coefficient vector for the input sample whose determining points are to be found.

Output: $D_S = [(\ell_1, T_1, K_1), \ldots, (\ell_n, T_n, K_n)]$, giving the locations, ℓ_i, and types of the determining points of S.

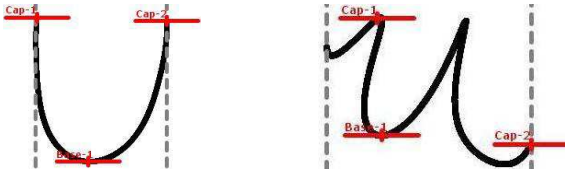
1. Let $x_A(s)$, $y_A(s)$, $x_S(s)$, $y_S(s)$ be the coordinate functions of the symbols given by A and S.
2. **for** $i \in 1..n$ **do**
   - **if** $K_i =$ ***max*** **then**
     - $f \longleftarrow -y_S$
   - **if** $K_i =$ ***min*** **then**
     - $f \longleftarrow +y_S$
   - $\ell_i \longleftarrow s$ such that $f(s)$ is minimized near s_i.
   - *Note this is the local minimum of a real univariate polynomial and any standard method may be used. For example, we use Newton's method to solve $f'(s) = 0$ with initial point $s = s_i$.*
3. Return$[(\ell_1, T_1, K_1), \ldots, (\ell_n, T_n, K_n)]$

A. The Reference Symbol

We examine two types of reference symbols as the choice of starting point for the homotopy: the *average symbol* and the *nearest neighbour*.

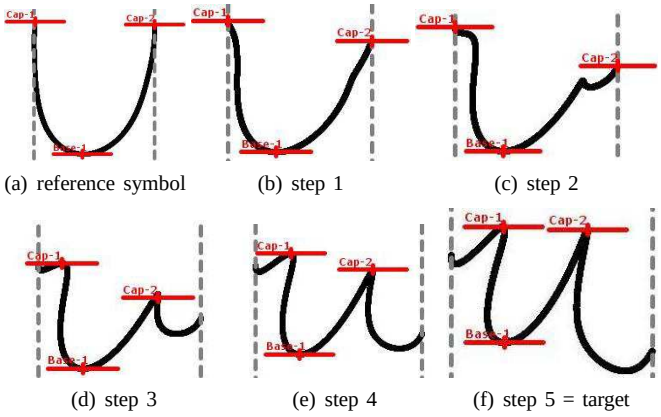
Average Symbol: The functional representation of digital ink traces has the advantage that the curves become points in a linear space. It therefore makes sense to talk about the average of several points. Our first choice for reference symbol is the average symbol of a class, computed as $C_{avg} = \sum_{i=1}^{n} C_i/n$, where n is the number of the samples and C_i is the coefficient vector for the i^{th} sample. Note that this computed average is in general not actually one of the sample points. Figure 5(a) shows a small set of samples provided by different writers and Figure 5(b) shows the average symbol.

Choosing the average symbol as the reference symbol for a class is an intuitive choice because presumably it has little deformation compared to other samples in the class. Moreover, it will lie within the convex hull of the set of points being averaged. Earlier work has shown [18] that, with this representation at relatively low approximation degree, the symbol classes are almost completely pairwise separable by single hyperplanes. We may therefore take the volume enclosed by the convex hull of points as being properly included within the class. Finally, the choice of average symbol is attractive in that only one symbol per class need be annotated.



(a) reference symbol (b) target with misplaced point

Fig. 6: Failure example.



(a) reference symbol (b) step 1 (c) step 2

(d) step 3 (e) step 4 (f) step 5 = target

Fig. 7: Success in 5 steps.

Nearest Neighbour: Another choice of starting point for the homotopy is the nearest neighbour in the class of the new symbol. This has the advantage that it should resemble the new symbol more closely than the average symbol. We would therefore suspect this starting point might require fewer homotopy steps. This choice of starting point is not as simple as the average symbol, however, because all of the symbols in the database must be labelled with their determining points. For a database of any size, this is not something that can be done by hand.

B. Homotopy

For those samples that are significantly different from the reference symbol, we use a numerical homotopy between the reference symbol and the target sample and trace the movement of the determining points. Because the classes are linearly separable in the function space a line from the reference symbol to the target sample lies entirely within the class. Dividing this line into several equal steps, we may apply Algorithm 1 at each step to follow the determining points through the homotopy. If C_{ref} is the reference symbol for the class and C_{targ} is the new sample target, then the line joining the two points in the function space is given by $C(t) = (1-t)C_{ref} + tC_{targ}$, with t ranging from 0 to 1. The determining points should move smoothly as the character is deformed continuously by the homotopy. By taking discrete steps between the source and the target, we can choose a sequence of intermediary points with each consecutive pair close enough for Algorithm 1 to apply. Figure 6 shows an example where Algorithm 1 failed to identify one of the determining points when applied naively. However, when applied in a 5-step homotopy, it succeeds, as shown in Figure 7.

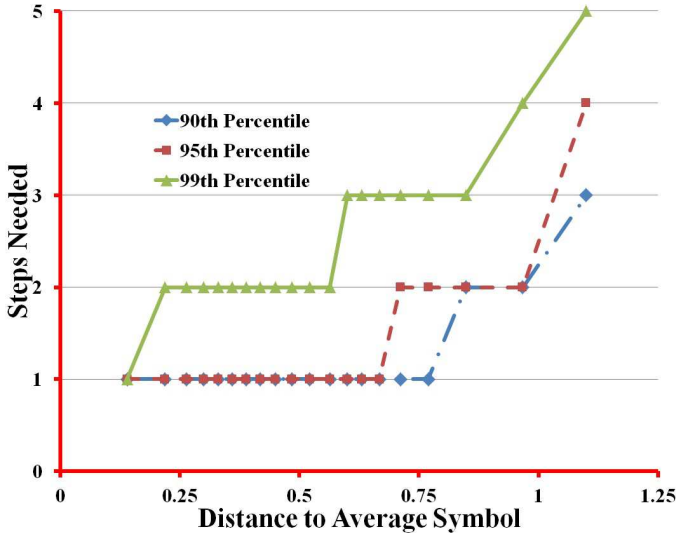


Fig. 8: Distance to **average symbol** *vs* the number of homotopy steps required. $\Delta s = 0.02$, $\Delta y = 1\%$. The overall success rate was 99.57% (45,440 out of 45,637). Each distance interval in the dense area ($r \leq 0.59$) contains 2800 samples while each interval in the sparse area ($r > 0.59$) contains 1500 samples, except the last one, which contains 1340 samples.

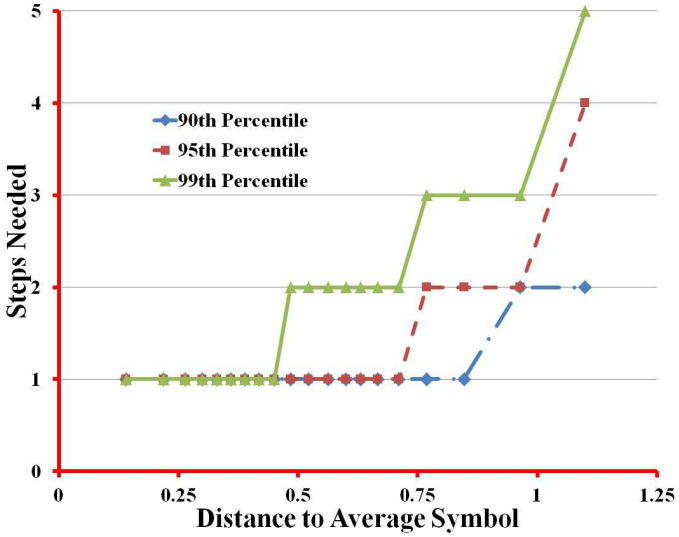


Fig. 9: Distance to **average symbol** *vs* the number of homotopy steps required. $\Delta s = 0.02$, $\Delta y = 3\%$. The overall success rate was 99.63% (45,470 out of 45,637). Each distance interval in the dense area ($r \leq 0.59$) contains 2800 samples while each interval in the sparse area ($r > 0.59$) contains 1500 samples, except the last one, which contains 1370 samples.

V. Experiments and Results

To evaluate the characteristics of the homotopy methods, we have run tests with a moderately large dataset of handwritten mathematical characters. The dataset consisted of altogether 45,637 samples from 240 different symbol types. These samples included Latin and Greek letters, digits, operators, and other mathematical characters, collected from various writers. All of these samples had been classified in advance. As some equivalent symbols were written in different styles, such as completely different forms, different numbers of strokes, or strokes in different orders, the symbols were grouped in a total of 388 classes.

We started the experiments by computing the average symbol of each class, and manually annotating this average symbol with its determining points. We then used this information to try to find the corresponding determining points on all samples in the dataset. We applied a 4-step homotopy strategy as this had achieved a high success rate in earlier work [4] and required a reasonable amount of time to compute. Then we visually inspected the determining points in each of the samples and manually adjusted the few incorrect ones. This collection of samples with corrected determining points then served as the ground truth.

We then went back and tested the data by forgetting the determining point annotations and seeing how many steps it took to recover them. We adopted the average symbol and the nearest neighbour as the reference symbols in the experiments and evaluated the number of homotopy steps required, respectively. As the locations of determining points were recorded as locations given by the arc length parameter, we compared the values with the ground truth. If the difference was within a particular threshold Δs, we considered it to be correct. Note that this threshold may not work for all the samples as different parameter values may result in the same height of metric lines. For example, given an upper case "T", any point located on the top bar can be used to determine the height of the cap line, but the difference in their parameter values may not fall into the threshold Δs. In such cases, we compared the normalized heights of the metric lines with the ground truth. We considered a result as correct if and only if the difference was within a particular threshold Δy.

We have investigated the relation between the distance to the reference point and the number of steps required under two homotopy strategies. The first strategy performs a homotopy from the average symbol. The second strategy uses a homotopy from the nearest neighbour with known determining points.

Figures 8 to 10 show, for different thresholds, the number of steps required to correctly identify the determining points using a homotopy from the average symbol. The figures show contours for the number of steps required to obtain certain success rates (90%, 95%, 99%) at varying distances to the reference symbol. Figures 11 and 12 show the results of similar experiments, but where the reference symbol was the nearest neighbour in the class.

This information may be used to tell how many steps are required to find the determining points in a new sample. After calculating the distance from the new sample to the average symbol, we can read off how many homotopy steps are required for the desired success rate.

Figure 8 shows the relation when the tolerances are $\Delta s = 0.02$ and $\Delta y = 1\%$. The overall success rate with no limit on the number of steps was 99.57% (45,440 out of 45,637 samples). We divided the set of samples into groups, based on distance intervals to the average symbol. The size of each interval depended on the density of samples at that distance.

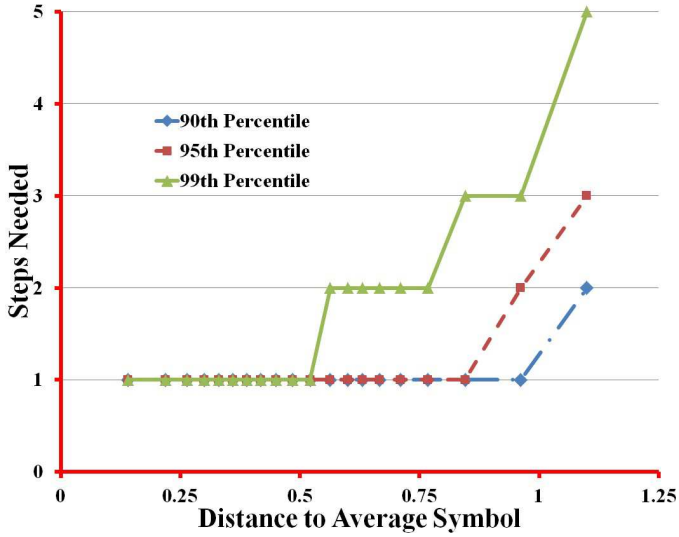


Fig. 10: Distance to **average symbol** *vs* the number of homotopy steps required. $\Delta s = 0.02$, $\Delta y = 7\%$. The overall success rate is 99.69% (45,496 out of 45,637). Each distance interval in the dense area ($r \leq 0.59$) contains 2800 samples while each interval in the sparse area ($r > 0.59$) contains 1500 samples, except the last one, which contains 1396 samples.

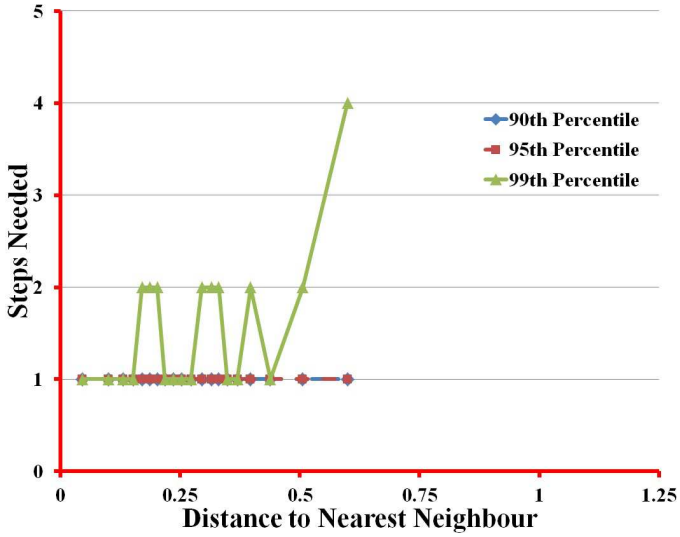


Fig. 11: Distance to **nearest neighbour** *vs* the number of homotopy steps required. $\Delta s = 0.02$, $\Delta y = 1\%$. The overall success rate is 98.86% (45,116 out of 45,637). Each distance interval in the dense area ($r \leq 0.32$) contains 2800 samples while each interval in the sparse area ($r > 0.32$) contains 1500 samples, except the last one, which contains 1016 samples.

There were more samples near the average symbol. In this dense region ($r \leq 0.59$) the intervals were chosen to contain 2800 samples each. In the sparse region, further from the average symbol ($r > 0.59$), each interval contained 1500 samples, except the last one contained 1340 samples. For each interval we determined the 90^{th}, 95^{th}, and 99^{th} percentiles of the number of homotopy steps required to correctly identify the determining points. Given the number of samples in each interval, the margin of error is 2% at a confidence level of 95%. As shown in the graph, as the distance to the average symbol increases, the number of required steps to achieve the 90^{th}, 95^{th}, and 99^{th} percentiles increases. Figures 9 and 10 show the relation between the distance to the average symbol and the number of homotopy steps needed with larger Δy tolerances.

Figure 11 shows the relation between the distance to the nearest neighbour and the number of homotopy steps needed where $\Delta s = 0.02$ and $\Delta y = 1\%$. The overall success rate was 98.86% (45,116 out of 45,637 samples). We divided the distances into a number of intervals, as before. In the dense area ($r \leq 0.32$) the intervals were chosen to contain 2800 samples each. In the sparse area ($r > 0.32$), each interval contained 1500 samples, except the last one, which contained 1016 samples. For each interval we determined the 90^{th}, 95^{th}, and 99^{th} percentiles of the number of homotopy steps required. The margin of error is 2% at confidence level of 95%. As before, the number of required steps generally increased with the distance. Figure 12 shows the results when $\Delta s = 0.02$ and $\Delta y = 3\%$. When $\Delta s = 0.02$ and $\Delta y = 7\%$, the graph looks identical to Figure 12 except the overall success rate is 99.68% (45,490 out of 45,637). In that experiment, the last interval contained 1390 samples.

We have seen that with both types of reference symbol—average and nearest neighbour—the number of required homotopy steps increases with the distance. Comparing the results, we also see that for a given distance the number of required steps is about the same for both strategies. As is completely expected, the distance to the reference symbol is usually smaller when the nearest neighbour strategy is used. Therefore with nearest neighbour we have on average a smaller number of required steps. The tradeoff is that we must retain annotations for all symbols in the classifier, rather than just for the average symbol.

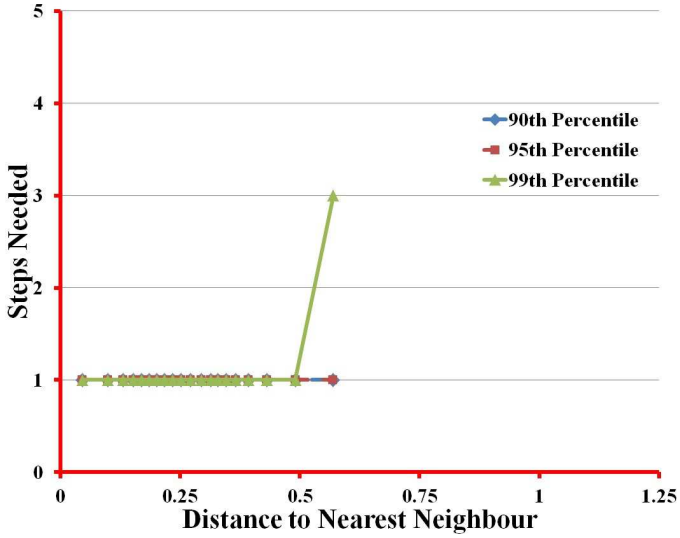


Fig. 12: Distance to **nearest neighbour** *vs* the number of homotopy steps required. $\Delta s = 0.02$, $\Delta y = 3\%$. The overall success rate is 99.45% (45,384 out of 45,637). Each distance interval in the dense area ($r \leq 0.32$) contains 2800 samples while each interval in the sparse area ($r > 0.32$) contains 1500 samples, except the last one, which contains 1284 samples.

VI. Conclusion

We have evaluated different strategies to locate special points on handwriting samples by deforming a known instance using linear homotopy on polynomial models.

We have evaluated two strategies for choice of starting point for the homotopy: to use the average symbol of the class and to use the nearest neighbour within the class. We have found that the number of homotopy steps required as a function of distance is about the same with the two strategies. The variance in the number of steps is also similar, but, is perhaps somewhat lower with the nearest neighbour strategy, suggesting a slightly better predictability. At any given tolerance level, the overall success rate was slightly higher when starting with the average symbol than when starting with the nearest neighbour. This does not appear to be significant.

We have also evaluated the number of homotopy steps required as a function of the distance between the new symbol and the starting point in the function space. For the regions where we have sufficient data, we have found a clear relation between the distance in the function space and the required number of steps with both starting point strategies.

In previous work [4], we had used a highly conservative number of homotopy steps (10 to 20) in order to obtain an error rate of 0.25%. The present results allow the choice of an appropriate and much reduced number of steps depending on the homotopy distance (to 1 to 5 steps). In all cases the success rate at correctly locating the determining points on test symbols was greater than 98.8%, so these explorations improve the efficiency of a useful algorithm.

References

[1] M. Koschinski, H.-J. Winkler, and M. Lang, "Segmentation and Recognition of Symbols within Handwritten Mathematical Expressions," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing*, vol. 4, Detroit, USA, 1995, pp. 2439–2442.

[2] E. Smirnova and S. M. Watt, "Aspects of Mathematical Expression Analysis in Arabic Handwriting," in *Proceedings of the International Conference on Document Analysis and Recognition*, vol. 2, Curitiba, Brazil, 2007, pp. 1183–1187.

[3] E. Smirnova and S. M. Watt, "A Context for Pen-Based Mathematical Computing," in *Proceedings of the 2005 Maple Summer Workshop*, Waterloo, Canada, 2005, pp. 409–422.

[4] R. Hu and S. M. Watt, "Determining Points on Handwritten Mathematical Symbols," in *Proceedings of the 2013 Conferences on Intelligent Computer Mathematics*, Bath, UK, 2013, pp. 168–183.

[5] B. Al-Badr and S. A. Mahmoud, "Survey and Bibliography of Arabic Optical Text Recognition," *Signal Processing*, vol. 41, no. 1, pp. 49–77, 1995.

[6] A. Amin, "Off-Line Arabic Character Recognition: the State of the Art," *Pattern Recognition*, vol. 31, no. 5, pp. 517–530, 1998.

[7] M. Pechwitz and V. Märgner, "Baseline Estimation for Arabic Handwritten Words," in *Proceedings of the 8th International Workshop on Frontiers in Handwriting Recognition*, 2002, pp. 479–484.

[8] M. T. Infante Velázquez, "Metrics and Neatening of Handwritten Characters," Master's thesis, The University of Western Ontario, London, Canada, 2010.

[9] S. M. Watt and T. Underhill (Editors), "Ink Markup Language (InkML) W3C Recommendation," http://www.w3.org/TR/InkML/, September 2011.

[10] S. D. Connell and A. K. Jain, "Template-Based Online Character Recognition," *Pattern Recognition*, vol. 34, pp. 1–14, 1999.

[11] R. Zanibbi, K. Novins, J. Arvo, and K. Zanibbi, "Aiding Manipulation of Handwritten Mathematical Expressions through Style-Preserving Morphs," in *Proceedings of Graphics Interface*, 2001, pp. 127–134.

[12] R. Hu and S. M. Watt, "Optimization of Point Selection on Digital Ink Curves," in *Proceedings of the 2012 International Conference on Frontiers in Handwriting Recognition*, September 2012, pp. 527–532.

[13] M. Harouni, D. Mohamad, and A. Rasouli, "Deductive Method for Recognition of On-Line Handwritten Persian/Arabic Characters," in *Proceedings of the 2nd International Conference on Computer and Automation Engineering*, vol. 5, February 2010, pp. 791–795.

[14] S. M. Watt, "Polynomial Approximation in Handwriting Recognition," in *Proceedings of the 2011 International Workshop on Symbolic-Numeric Computation*. ACM, 2011, pp. 3–7.

[15] O. Golubitsky and S. M. Watt, "Online Stroke Modeling for Handwriting Recognition," in *Proceedings of the 2008 Conference of the Center for Advanced Studies on Collaborative Research: Meeting of Minds*. ACM, 2008, pp. 72–80.

[16] B. W. Char and S. M. Watt, "Representing and Characterizing Handwritten Mathematical Symbols through Succinct Functional Approximation," in *Proceedings of the Ninth International Conference on Document Analysis and Recognition*. IEEE Computer Society, 2007, pp. 1198–1202.

[17] O. Golubitsky and S. M. Watt, "Distance-Based Classification of Handwritten Symbols," *International Journal on Document Analysis and Recognition*, vol. 13, no. 2, pp. 133–146, June 2010.

[18] O. Golubitsky and S. M. Watt, "Online Recognition of Multi-Stroke Symbols with Orthogonal Series," in *Proceedings of the 10th International Conference on Document Analysis and Recognition*, Barcelona, Spain, 2009, pp. 1265–1269.