

Ink Normalization and Beautification

Patrice Y. Simard
Microsoft Research
One Microsoft Way
Redmond, WA 98052
patrice@microsoft.com

Dave Steinkraus
Microsoft Research
One Microsoft Way
Redmond, WA 98052
steinkraus@hotmail.com

Maneesh Agrawala
Microsoft Research
One Microsoft Way
Redmond, WA 98052
maneesh@microsoft.com

Abstract

Handwriting recognition is difficult because of the high variability of handwriting and because of segmentation errors. We propose an approach that reduces this variability without requiring letter segmentation. We build an ink extrema classifier which labels local minima of ink as {bottom, baseline, other} and maxima as {midline, top, other}. Despite the high variability of ink, the classifier is 86% accurate (with 0% rejection). We use the classifier information to normalize the ink. This is done by applying a “rubber sheet” warping followed by a “rubber rod” warping. Both warpings are computed using conjugate gradient methods. We display the normalization results on a few examples. This paper illustrates the pitfalls of ink normalization and “beautification”, when solved independently of letter recognition.

1. Introduction

Almost every handwriting recognition system includes a preprocessing step for the purpose of removing slant and slope. For off-line recognition, the normalization typically relies on histogram projection [4, 8], entropy [3], or more complex heuristics [2]. Wienecke et al. [9] track the baseline and midline with cubic splines. For on-line recognition, tracking baseline and midline can also be useful, for computing off-line features. For instance, Bengio et al. [1] fit parallel quadratic curves to track bottom, baseline, midline and top through each word, in order to extract Amaps (angle maps).

None of these methods tracks baseline or midline accurately, in the sense that they do not identify extrema of ink (off-line or on-line) as belonging or not belonging to these lines with accurate probabilities. In this paper, we cast baseline and midline tracking as a classification problem, and use state-of-the-art machine learning techniques to solve it. Our aim is threefold:

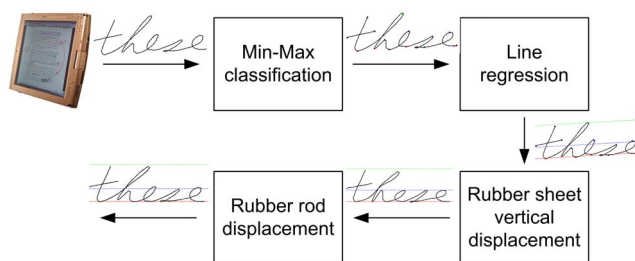


Figure 1. System overview

1. Achieve unprecedented accuracy in identifying bottom line, baseline, midline and top line. For this to be possible, we have created a large training database containing 10,000 words with each extremum labeled. Results on this database are presented in section 3.
2. Show that it is possible to extract useful information from ink without segmentation. Segmentation is the Achilles heel of machine learning, and arguably the limiting factor in handwriting recognition. By classifying extrema, which are easily located, we enable preprocessing or provide new features for more complex tasks such as letter segmentation and classification.
3. Develop algorithms for normalization and ink beautification, without relying on accurate segmentation and letter recognition. To this end, we have implemented two forms of warping, described in sections 5.1 and 5.2. The results on sample ink are shown in section 6.

2 System overview

Our system is essentially composed of 4 parts and is depicted in Figure. 1. First, a classifier labels the minima and maxima in the ink. Multi-line regression is then used to find the relative offsets between bottom, base, mid, and top lines. This second step also yields the new desired posi-

tion for each minimum and maximum. The third step interpolates these “point” displacement to a 2D grid by enforcing rubber-sheet or “thick-plate” constraints. Finally, kinks and curvature changes in the ink are removed by enforcing curvature and compression constraints between the original and displaced ink.

3 Min-Max classification

Local vertical minima and maxima can easily be located on the time trajectories collected from the pen-enabled capture devices. To proceed to displacing the ink for better alignment, we need to classify each of these extrema to a label corresponding to its target position. We have chosen 3 categories for minima: bottom, baseline, and other. The last category is used for every minimum other than the first two, and is not subject to any alignment constraint. Similarly, we have 3 categories for maxima: midline, top, and other.

Labeling extrema automatically is a difficult task because of the large variations between characters and handwriting styles. A crude labeling can be achieved by projecting the ink horizontally and looking at the histogram to detect the baseline and midline. This method is often used to normalize the ink before classification [4]. We have used this method to pre-label our data. To achieve higher accuracy, we have built a convolution network classifier, similar to those used for character classification [5]. The classifier architecture is depicted in Figure 2. The architecture is

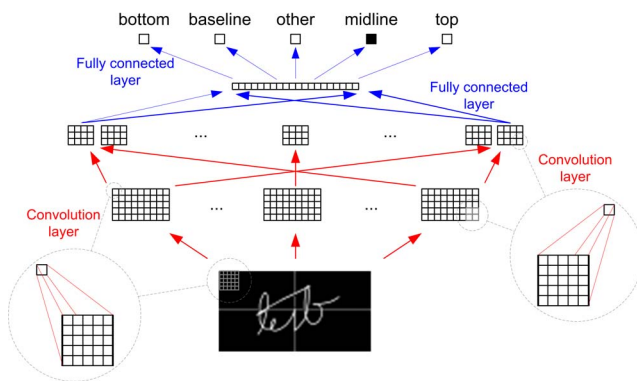


Figure 2. Convolution classifier: First 2 layers of weights are 5x5 subsampled convolutional kernels (red). Last two layers are fully connected layers of weights (blue). The sight on the input image (showing ‘test’) is not part of the input but is displayed here to show which extremum is being classified (top of ‘s’, a midline).

identical to [5]. The input is an 85 by 45 pixel image centered on the extremum to be classified. The first and second convolutional layers have respectively 5 features and 50 features. The 3rd layer is fully connected and has 100

Labels	Examples	NN	NN (harm.)
other	3466	19%	19%
ascender	843	29%	14%
midline	2276	17%	4%
baseline	3198	7%	0%
descender	175	15%	10%
Total:	9958	16%	9%

Table 1. MinMax error rates for Convolution neural network and histogram projection. On each line, the error percentage represents the number of errors for that category. For instance, the convolutional neural net misclassified 29% of the 843 ascenders it was presented from the test set. The 4th column ‘NN (harm.)’ contains the most harmful errors. Misclassifying something as ‘other’ is deemed less harmful because ‘other’ will not be displaced by the beautification algorithm (misclassifying ‘other’ as something else, however, is always harmful).

units. The output has 5 units, one for each label, and is trained with cross-entropy. Microsoft Tablet PC provided a large database of handwritten words. There are on average 4 extrema per letter and 5 letters per word. Approximately 10,000 words were labeled. The labeling process is error-prone, and many extrema are so ambiguous that labelers disagree on about 5% of the data (estimate). For instance, the bottom of an ‘f’ can be a baseline, a descender, or halfway in between, depending on the handwriting; the extremities of a script ‘e’ can arguably be either baseline or ‘other’. We trained a neural network on 10,000 labeled words and tested it on a different subset of 519 words. The results are shown in Table 1.

4 Fitting lines

In this section we fit parallel lines through text. Note that our goal is to get an estimate of where the final lines will be rendered. This is different from assuming that the baseline is straight. Clearly the points returned by our classifier will not be aligned. In this respect, we make fewer assumptions than [1] who assume that the baselines are quadratic. Fitting lines through the points, however, will estimate the vertical position at which the line will be rendered after warping. It will also estimate the relative distance between baseline, midline, top line and bottom line. This method will fail in extreme cases, for instance if one wrote a line or a word along a circle.

Given sets of N_j ($0 \leq j \leq M-1$) points belonging to M labeled lines (e.g. baseline or midline), we need to know the relative offsets b_j of each of these lines in order to compute the optimal displacement that will align each point to its

respective line. The lines (e.g. baseline and midline) are constrained to be parallel. This is a fairly straightforward multiline linear regression. If each point (x, y) belonging to line j follows the equation,

$$y = ax + b_j \quad (1)$$

finding a and b_j given M collection of N_j point (x_i^j, y_i^j) can be done by minimizing:

$$E(a, b_0, \dots, b_{M-1}) = \sum_{j=0}^{M-1} \sum_{i=0}^{N_j-1} \frac{1}{2} (y_i^j - (ax_i^j + b_j))^2 \quad (2)$$

Setting the derivatives of $E(a, b_0, \dots, b_{M-1})$ with respect to a, b_0, \dots, b_{M-1} to 0 yields the following linear system of $M + 1$ equations and $M + 1$ variables:

$$T_{xx}a + \sum_{j=0}^{M-1} T_x^j b_j - T_{xy} = 0 \quad (3)$$

$$T_x^0 a + N_0 b_0 - T_y^0 = 0 \quad (4)$$

$$\vdots \quad \vdots \quad \vdots$$

$$T_x^{M-1} a + N_{M-1} b_{M-1} - T_y^{M-1} = 0 \quad (5)$$

Where $T_{xx} = \sum_{j=0}^{M-1} \sum_{i=0}^{N_j-1} x_i^j x_i^j$, $T_x = \sum_{j=0}^{M-1} \sum_{i=0}^{N_j-1} x_i^j$, $T_{xy} = \sum_{j=0}^{M-1} \sum_{i=0}^{N_j-1} x_i^j y_i^j$, $T_x^j = \sum_{i=0}^{N_j-1} x_i^j$, and $T_y^j = \sum_{i=0}^{N_j-1} y_i^j$. This yields:

$$a = \frac{T_{xy} - \sum_{j=0}^{M-1} \frac{1}{N_j} T_x^j T_y^j}{T_{xx} - \sum_{j=0}^{M-1} \frac{1}{N_j} T_x^j T_x^j} \quad (6)$$

$$b_j = \frac{1}{N_j} (T_y^j - T_x^j a) \quad (7)$$

Note that the number of lines M varies (some writing samples may not have any descenders or ascenders). It is also possible to do the regression for a full paragraph of text at once, with multiple baselines, midlines, etc. If none of the M lines have more than one point, the system is underconstrained, in which case we assume $a = 0$. Examples of line regressions are given in Figure 3 and 4.

5 Ink warping

Isometric transformations of handwritten text, such as translation and rotation, can help restore alignment between words. Unfortunately, such transformations do not restore alignment between letters within a word. To restore alignment within a word, one must move letters with respect to each other, which results in an direct alteration of the ink's appearance. To be visually appealing, such alteration should satisfy simultaneously the following constraints:

1. Alignment: the displacement should move the extrema to the line they belong to. This constraint affects only a few points (at extrema locations), and only restricts the vertical component of the displacement.
2. Spatial feature preservation: Intersections and distance between pieces of ink should be preserved. For instance, the top of a 'u' should not be closed and thus transformed into an 'o' or an 'a'. All the strokes of a multi-stroke character should move together ('t', accented 'e', script letters, etc). These are 2-dimensional *image* constraints between pieces of ink that have been drawn at different times and which may or may not be connected.
3. Local ink feature preservation: Curvature, aspect ratios, angle. The displacement should attempt to preserve curvature, aspect ratio, and angle within a letter. Of these 3, the first is the most important. An inversion of curvature or the introduction of a kink is both unnatural and immediately detectable. The second is a compressibility constraint. A 'd' should not be transformed into a 'a' by vertical compression of the upper part. The last constraint, angle, is used to preserve the slant and angle of separated strokes, such as 't' crossings and accents. This constraint can be gently enforced locally to affect the global angle/slant of an accent or a letter. Each of these local constraints can be viewed as a set of springs between the transformed ink and the original ink.

5.1 Rubber sheet optimization

The first two constraints can be optimized simultaneously by considering a rubber sheet displacement. The displacement $(u_{i,j}, v_{i,j})$, which is composed of an u and a v component at each point i, j is constrained to fixed values at the ink extrema locations in J :

$$\forall (i, j) \in J, \quad v_{i,j} = t_{i,j} \quad (8)$$

A soft constraint version can be written as:

$$E_t(v) = \frac{1}{2} \sum_{(i,j) \in J} (v_{i,j} - t_{i,j})^2 \quad (9)$$

The horizontal component u is unconstrained ($E_t(u) = 0$). The second constraint, preservation of text features, is enforced by requiring the displacement to be smooth. This can be done by minimizing the first and second derivatives of the displacement field. Minimizing the first derivative can be written as:

$$E_m(u, v) = \frac{1}{2} \sum_{(i,j)} [(u_{i+1,j} - u_{i,j})^2 + (u_{i,j+1} - u_{i,j})^2 + (v_{i+1,j} - v_{i,j})^2 + (v_{i,j+1} - v_{i,j})^2] \quad (10)$$

Minimizing the second derivative can be written as:

$$E_p(u) = \frac{1}{2} \sum_{(i,j)} [(u_{i+1,j} - 2u_{i,j} + u_{i,j+1})^2 + 2(u_{i+1,j+1} - u_{i,j+1} - u_{i+1,j} + u_{i,j})^2 + (u_{i,j+1} - 2u_{i,j} + u_{i,j-1})^2] \quad (11)$$

The second-derivative constraint is applied to both the u and v components of the displacement field, $E_p(u, v) = E_p(u) + E_p(v)$. The model that minimizes $E_t(u, v)$ and $E_m(u, v)$ is called the membrane model, while the model that minimizes $E_t(u, v)$ and $E_p(u, v)$ is called the thin plate model [7]. There is abundant literature on algorithms to solve either problem efficiently using conjugate gradient and multiresolution [6]. We should note that the constraints $E_t(u)$, $E_m(u)$ and $E_p(u)$, and the constraints $E_t(v)$, $E_m(v)$ and $E_p(v)$ are independent. The solution $u = 0$ clearly minimizes $E_t(u)$, $E_m(u)$ and $E_p(u)$. Our problem is therefore to minimize:

$$E(v) = \alpha_t E_t(v) + \alpha_m E_m(v) + \alpha_p E_p(v) \quad (12)$$

where α_t , α_m , and α_p are weighting factors on the errors we are trying to minimize. If hard constraints are used for the target displacements, we then have $E_t(v) = 0$ and we use the notation $\alpha_t = \infty$.

Equation 12 is linear in v and can be solved with conjugate gradient descent. We used a multiresolution approach based on Szeliski's paper [6]. The results are illustrated in Figure 3. The threshold parameters have been adjusted manually to fit a variety of handwriting styles as well as possible. For this example, we observe (second line) that the 2D rubber sheet displacement is effective at satisfying the target constraints, and preserves the relationships between various pieces of ink (e.g. 't' crossing). The result still leaves much to be desired. The first 'c' has been flattened and appears somewhat unnatural. The last 's' has been kinked, and the 't' crossings have changed orientation. The third line shows a 1D rubber rod displacement applied to the ink. This displacement will be explained in detail in the next section. The limitation of this displacement is that it does not preserve the 2D spatial relationships along the ink. For instance, the first 'a' crosses itself, unlike the original. The 't' crossings are left behind. The last line shows a combination of the two methods, which will be explained in more detail in the next section. The drawbacks of both method have been somewhat mitigated.

5.2 Rubber rod optimization

The third kind of constraint for ink warping – maintaining curvature, aspect ratio and angle – can be enforced by putting local constraints between the new ink trajectory and

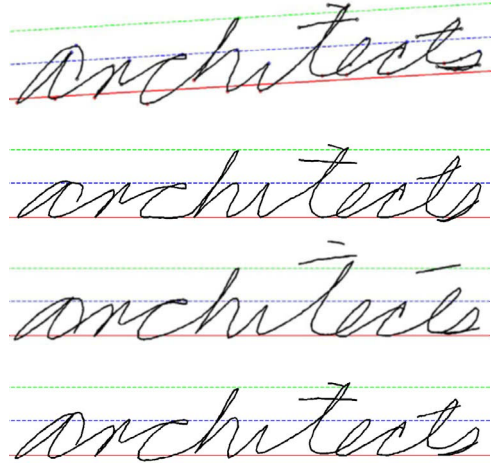


Figure 3. Top: Original image, with regression lines and extrema labels shown. The next 3 pictures use rubber sheet, rubber rod, and both deformations to bring extrema to their rightful lines. Second from the top: rubber sheet, $\alpha_t = \infty$, $\alpha_m = 0.01$, $\alpha_p = 1$. Third from the top: rubber rod, $\beta_t = 0.001$, $\beta_l = 0.01$, $\beta_c = 1$, $\beta_a = 0.02$. Bottom: rubber sheet displacement followed by a rubber rod displacement (same parameters as above).

the original ink. Let $x(t)$ and $y(t)$ be the coordinates of points along the ink trajectory of the beautified ink. If $\chi(t)$ and $\psi(t)$ are target points that we would like the ink to go through for a subset of points J , the target constraint can be written as:

$$E_t(x, y) = \frac{1}{2} \sum_{(t) \in J} (x_t - \chi_t)^2 (y_t - \psi_t)^2 \quad (13)$$

The most disturbing aspect of the rubber sheet deformation is that it can introduce kinks and changes of curvature in the displaced ink, as happens to the second 'a' in Figure 4, second line. To prevent this, we introduce a constraint that ties the curvature of the displaced curve $x(t), y(t)$ to the curvature of the original curve $X(t), Y(t)$. Since this constraint is optimized by gradient descent, special care was taken to choose a constraint that does not generate arbitrarily large gradients (which would cause large eigenvalues in the parameter space of optimization). For instance, the traditional definition of curvature of $x(t), y(t)$ defined by:

$$\kappa(t) = \frac{x'(t)y''(t) - y'(t)x''(t)}{(x'(t)^2 + y'(t)^2)^{3/2}} \quad (14)$$

varies from 0 for a straight line, to arbitrarily large values for sharp reversal in direction. It is not stable when optimized using gradient descent. Instead, we optimize the

following constraint:

$$E_c(x, y) = \frac{1}{2} \sum_t (\cos(\theta(t)) - \cos(\Theta(t)))^2 + (\sin(\theta(t)) - \sin(\Theta(t)))^2 \quad (15)$$

where $\theta(t)$ (resp. $\Theta(t)$) is defined to be the angle between 3 consecutive points on the curves $x(t), y(t)$ (resp. $X(t), Y(t)$). We do not allow consecutive duplicate points. The derivative of $E_c(x, y)$ with respect to $x(t)$ and $y(t)$ is bounded everywhere and is therefore better behaved for optimization purposes.

Optimizing curvature alone could result in large changes in length between consecutive points. To circumvent this problem, we add a constraint to prevent compression along the ink trajectory:

$$E_l(x, y) = \frac{1}{2} \sum_t (l(t) - L(t))^2 \quad (16)$$

where $l(t)$ and $L(t)$ are defined to be the distance between two consecutive points on the curves $x(t), y(t)$ and $X(t), Y(t)$ respectively.

The above two constraints, when fully satisfied, still allow individual pieces of ink to rotate. For completeness, we add one more constraint to act on the absolute angle of a curve. This constraint is meant to be weak, but to have an effect on otherwise unconstrained pieces of ink (e.g. accents, 't' crossings, etc).

$$E_a(x, y) = \frac{1}{2} \sum_t (\cos(\bar{\theta}(t)) - \cos(\bar{\Theta}(t)))^2 + (\sin(\bar{\theta}(t)) - \sin(\bar{\Theta}(t)))^2 \quad (17)$$

where $\bar{\theta}(t)$ (resp. $\bar{\Theta}(t)$) is defined to be the angle between 2 consecutive points on the curves $x(t), y(t)$ (resp. $X(t), Y(t)$) and the horizontal.

We can now optimize:

$$E_R(x, y) = \beta_t E_t(x, y) + \beta_c E_c(x, y) + \beta_l E_l(x, y) + \beta_a E_a(x, y) \quad (18)$$

Unlike the 2D rubber sheet optimization, E_R is highly non-linear in x and y . It is likely to have multiple local minima. To optimize it, we use the Polak-Ribiere conjugate gradient technique. The rubber rod optimization is used to displace the extrema of the ink, along a pre-shaped rubber rod, to their desired positions along the corresponding lines. The results are displayed on the third lines of Figures 3 and 4. As mentioned previously, the rubber rod alone is not effective at capturing 2D relationships such as accents, 't'-crossings, intersections, or loop-closing. On the other hand, the 2D rubber sheet constraint ignores the path followed by the ink. This is illustrated in Figure 4. In this figure, two points on and near the second 'a' of 'areas' end up being compressed on the baseline, with a catastrophic effect on that letter (second line on the figure). To prevent this from occurring, we

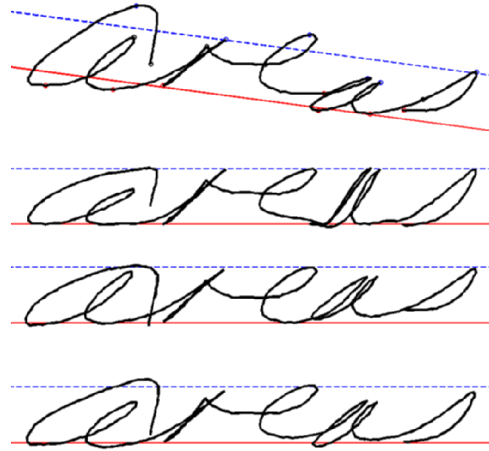


Figure 4. Top: Original image, with regression lines and extrema labels shown. The next 3 pictures use rubber sheet, rubber rod, and both deformations to bring extrema to their rightful lines. Second from the top: rubber sheet, $\alpha_t = \infty, \alpha_m = 0.01, \alpha_p = 1$. Third from the top: rubber rod, $\beta_t = 0.001, \beta_l = 0.01, \beta_c = 1, \beta_a = 0.02$. Bottom: rubber sheet displacement followed by a rubber rod displacement (same parameters as above).

enforce ink 'rigidity' using the rubber rod constraint. We first compute the rubber sheet deformation and apply the corresponding displacement to compute a new $X(t), Y(t)$ target trajectory. This will be the target of $E_t(x, y)$ of the rubber rod constraints (J contains every point). We then optimize $E_R(x, y)$. The results are shown on the last line of Figures 3 and Figure 4. Clearly, the curvature and compression improved the looks of the first 'c' and 's' in 'architects' and second 'a' in 'areas'. The angle constraint helps the 't' crossing in 'architects'.

Ideally, rubber sheet and rubber rod could be optimized together. However, the 2D approach is computationally expensive because of the number of pixels, while the 1D approach is computationally expensive because of the large number of iterations required to solve a nonlinear problem. Simultaneously optimizing both constraints while retaining practical speed is non-trivial. The present system processes a word per second on average on a 3GHz P4, without SSE or MMX optimization.

6 Results on paragraphs

The results on a 2-line paragraph are shown on Figure 5. The first two lines are the original writing on a Tablet PC. Line 3 and 4 are the result of the end-to-end system. Base-

whose woods these were I think I know
 his house is in the millage bough
 whose woods these were I think I know
 his house is in the millage bough
 whose woods these were I think I know
 his house is in the millage bough
 whose woods these were I think I know
 his house is in the millage bough

Figure 5. Lines 1-2: original ink. Line 3-4: end-to-end system with warping of baseline and midline. Line 5-6: affine transform of the ink (no warping) with manual labeling. Line 7-8: warping with manual labeling.

line and midline were labeled automatically with our extrema classifier. The strange 'dot' on the 'i' is the result of warping the dot to the midline because of a classification error (midline instead of other). For the next four lines, we wanted to illustrate the difference between affine transformation – which leaves the ink unchanged except for rotation and translation – and warping. To make the experiment more conclusive, we hand-labeled every extremum to produce the maximum deformations. Without hand-labeling, many extrema would be classified as 'other', and would not force the ink to be warped. Lines 5 and 6 show the ink resulting from affine transformation. Lines 7 and 8 show the ink resulting from warping.

The first conclusion from this experiment is that the beautification effects of ink warping are subtle (if not disappointing). When we asked people to rank the quality of the ink on this picture, the consensus was (from best to worst): 7-8 (full warping), 3-4 (our end-to-end system), 5-6 (affine), and 1-2 (original). Our classifier did very well for some writers, but gave idiotic answers for others. We attribute this to poor generalization due to training on a database limited to only 10,000 words (which is very small for handwriting recognition tasks).

7 Conclusion

While many publications focus on handwriting recognition, we have concentrated our efforts on the first preprocessing step, ink normalization. We have cast this problem as a full-fledged classification problem, and used machine learning to solve it. With a 10,000-word database, our

ink extrema classification accuracy is encouraging (86%), although it is clear that a larger database will further decrease the error rate. Extrema classification is easier than letter recognition because it does not require segmentation. We expect that such features, whether used directly or in a preprocessing step, have great potential to improve handwriting recognition.

Our classification results were obtained with on-line ink. However, if extrema are computed on images (which is not difficult), our min-max classifier can be used "as is".

We have implemented a warping algorithm that straightens the bottom, baseline, midline, and top line. We have tested warping these lines for ink beautification. The results are more subtle than we had expected. Some subjects liked it while others were skeptical. A formal user study is needed to reach a conclusion on the validity of our beautification approach. The warping algorithm, however, shows great promise as a preprocessing step for handwriting processing.

References

- [1] Y. Bengio, Y. LeCun, C. Nohl, and C. Burges. Lerec: A nn/hmm hybrid for on-line handwriting recognition. *Neural Computation*, 7(6):1289–1303, November 1995.
- [2] R. M. Bozinovic and S. N. Srihari. Off-line cursive script word recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 11(1):68–83, 1989.
- [3] M. Côté, E. Lecolinet, M. Cheriet, and C. Y. Suen. Automatic reading of cursive scripts using a reading model and perceptual concepts the perpecto system. *IJDAR*, 1(1):3–17, 1998.
- [4] G. Nicchiotti and C. Scagliola. Generalised projections: A tool for cursive handwriting normalisation. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 729–732, 1999.
- [5] P. Simard, D. Steinkraus, and J. Platt. Best practice for convolutional neural networks applied to visual document analysis. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 958–962, 2003.
- [6] R. Szeliski. Fast surface interpolation using hierarchical basis functions. In *IEEE Trans. on Pattern Analysis and Machine Intelligence*, volume 12, pages 513–528, 1990.
- [7] D. Terzopoulos. Multilevel computational processes for visual surface reconstruction. In *Comput. Vision, Graphics, Image Processing*, volume 24, pages 52–96, 1983.
- [8] A. Vinciarelli and J. Luettin. A new normalization technique for cursive handwritten words. *Pattern Recognition Letters*, 22(9):1043–1050, 2001.
- [9] M. Wienecke, G. A. Fink, and G. Sagerer. Towards automatic video-based whiteboard reading. In *International Conference on Document Analysis and Recognition (ICDAR)*, pages 87–91, 2003.