Degraded Character Image Restoration

John D. Hobby Henry S. Baird

Abstract

The design and analysis of an algorithm for the restoration of degraded images of machine-printed characters is presented. The input is a set of degraded bilevel images of a single unknown character; the output is an approximation to the character's ideal artwork. The algorithm seeks to minimize the discrepancy between the approximation and the ideal, measured as the worst-case Euclidean distance between their boundaries. We investigate a family of algorithms which superimpose the input images, add up the intensities at each point, and threshold the result. We show that, under degradations due to random spatial sampling error, significant asymptotic improvements can be achieved by suitably preprocessing each input image and postprocessing the final result. Experimental trials on special test shapes and Latin characters are discussed.

1 Introduction

In the last few years, a variety of document-image degradation models have been proposed, and their applications investigated [3]. Models of this sort are often instantiated as a software *image generator* which reads a single ideal *prototype* image (of, say, a machine-printed symbol), and, as directed by the model, writes an arbitrarily large number of pseudo-randomly degraded images. Here, we explore some implications of inverting this procedure by designing an *image restorer* which reads a set of degraded images and attempts to recover, or closely approximate, the ideal artwork from which they were derived.

This procedure is a special case of classical image restoration [12], of which image deconvolution is another, well-studied special case. The problem is similar in some respects to super-resolution surface reconstruction from multiple images [6] and sub-pixel edge location in grey-level images [1]. Our problem domain is distinguished from many that are treated in this literature by several factors: (1) we read many input images (not merely one); (2) the input images are bilevel (not grey or color); (3) the ideal image to be recovered is also bilevel and at a much higher (*e.g.* $\times$ 10) spatial sampling rate than the input; (4) we do not, in general, know the parameters of the image degradation model; and, (5) we may know some constraints on the class of images, such as the frequent occurrence along their boundaries of straight lines, sharp corners, and curves of large radius. For these reasons we foresee advantages in algorithms specially adapted to the problem domain.

We anticipate several applications of such an algorithm. One is speeding up the adaptation of an OCR system to a given document, by a procedure of the following sort: for each of the most commonly confused symbols, a few images are lifted and their ideal prototype is inferred and input to the generator, which can write a much larger training set than can be collected from the document. Another application is calibration of a degradation model (estimation of its parameters) to a document for which ideal prototypes are unknown.

We have investigated a family of algorithms which superimpose the input images, add up the intensities at each point, and threshold the result. Methods of this sort have been occasionally reported in the OCR literature [11], but we are not aware of any attempt to analyze their asymptotic performance or improve them by special pre- and post-processing.

There are several generally related papers in the literature on document image analysis which are worth mentioning although they neither attack the same problem nor use the same method. Billawa, Hart, and Peairs [5] studied the restoration of repeated defects at known locations in an image (e.g. scratches on a copier platen). Shin *et al* [13] explored not dissimilar methods for the purpose of contrast enhancement.

2 The Algorithm

The basic idea of the Image Averaging algorithm is to superimpose the input images, add up the intensities at each point, and threshold the result to obtain a new binary image. A broad class of algorithms can be expressed in this form by suitably preprocessing each input image and postprocessing the final output as suggested by Figure 1.

After the input images have been preprocessed, they can be superimposed by simply finding the centroid of each input image to subpixel accuracy and shifting the images so that the centroids coincide. It is also possible to use a feedback process to try to readjust the relative positions once a tentative consensus image is available as Hastie et. al. do [7], but this involves considerable complexity and does not avoid the theoretical limitations discussed in Section 4.

The superimposed inputs can be converted to a binary image by simple thresholding or by an edge detection algorithm such as Avrahami and Pratt [1].

As the picture suggests, the most important function of the postprocessing step is to smooth out the high-frequency "wobbles" in the outlines produced by the thresholding process. Adding up input images and thresholding them reduces the magnitude of this noise so that the wobbles are more readily distinguished from desired features such as the serifs on the letter "A" in Figure 1. The smoothing algorithm needs to operate on polygonal outlines and produce output that fits the input as closely as possible while smoothing out wobbles up to some specified magnitude. There should be no attempt to minimize the number of vertices in the output at the expense of quality of fit.

A good choice of smoothing algorithm is Hobby's algorithm [8]. This algorithm minimizes the number of inflections in the resulting polygonal outlines subject to a bound on the deviation. It produces a description of a class of outlines that obey these criteria so that the output can be chosen to fit the input as well as possible. The post-processing step consists of choosing the maximum deviation parameter as a yet-to-be determined function of the number of input images and running Hobby's algorithm.

The other essential component of the Image Averaging algorithm is the preprocessing step. One option is to omit the preprocessing and just treat the inputs as binary images where the pixels are unit squares. This gives the *Naive Averaging* algorithm.

Alternatively, Hobby's polygonal smoothing algorithm can be applied to the input images as part of the preprocessing step, but this must be done carefully to avoid destroying significant features. We con-

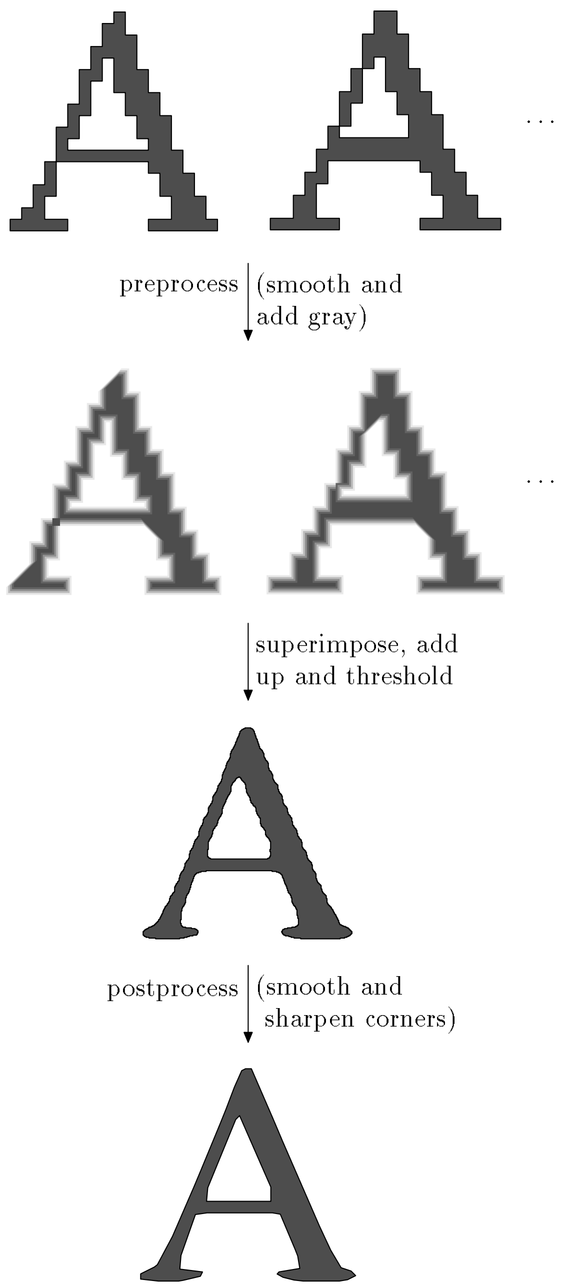


Figure 1: An example of how the algorithm performs with simulated input images as might be obtained from a 7 point font at approximately 300 dots/inch. Only the first two of 100 input images are shown here.

sidered various preprocessing options some involving the smoothing algorithm and some not. The options were evaluated both empirically and according to their theoretical performance as the number of input images approaches infinity. The details appear in later sections; we first concentrate on presenting the most successful preprocessing strategies.

2.1 Preprocessing by Smoothing the Outlines

One approach to preprocessing is to take each rasterized input image and try to guess the underlying shape that generated it. This can be thought of as "inverse rasterization," and the algorithm of [8] can do it as shown in Figure 2.

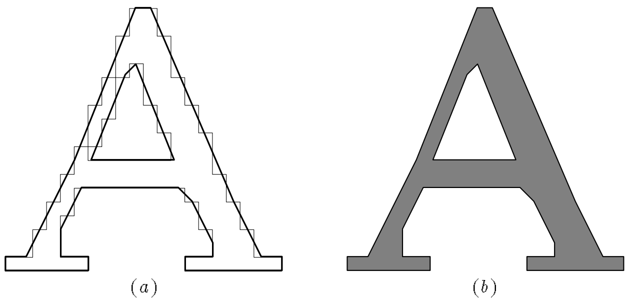


Figure 2: (a) A comparison of the original outlines (thin lines) with the smoothed versions (thick lines) for one of input images from Figure 1; (b) the smoothed outlines filled in.

The smoothing algorithm has an error tolerance parameter ϵ that must be set to at most $\frac{1}{2}$ pixel in order to guarantee that a simple rasterization process could regenerate the original input outlines. In fact, it may be better to choose a value like $\epsilon = \frac{1}{3}$ in order to avoid smoothing out features that may be significant. Using this preprocessing strategy for Image Averaging yields the *Presmoothing* algorithm.

2.2 Adding up the Inputs and Thresholding

Once we have polygonal outlines for each of the input images, how do we add them up? One way would be to build a data structure that divides the plane into regions according to the number of input images that overlap at each point. This would require maintaining a potentially very large number of polygonal regions and figuring out how to update them given the outlines that describe a new input image.

A more practical approach is first to rasterize each polygonal outline using any reasonable scan-conversion algorithm, and then add up the rasterized images. This rasterization should be done at a resolution substantially higher than that of the input images since the resolution limits the precision of the final output. It is best to use a run-length representation so that the run time and space requirements will not be quadratic in the resolution.

Scale each input by some factor σ, and assume that the scan-conversion algorithm finds all triplets of integers (x, y, d) such that the scaled outlines cross the segment $(x, y)\,(x + 1, y)$ and $d = \pm 1$ depending on whether the crossing is in the downward or upward direction. After all the inputs have been rasterized in this fashion, we can collect all the triplets with a given y-value and sort them according to x. Enforcing a threshold t involves scanning the sorted list, maintaining the cumulative total of the d values and saving only those triplets that raise the total from $t-1$ to t or lower it from t to $t - 1$. Knuth has published detailed implementations of all these algorithms [9, Parts 19,20,22].

2.3 Preprocessing via Smooth-Shaded Outlines

Rather than trying to guess the underlying shape from looking at an input image, a better preprocessing strategy might be to try to express the range of possibilities. In other words, the preprocessed image should have fuzzy edges that simulate the result of averaging all the possible underlying images. This can be done by taking a closer look at the output of Hobby's smoothing algorithm [8]. In addition to the polygonal approximation indicated by the dashed line in Figure 3, there is a sequence of trapezoids that the dashed line passes through. Instead of using the dashed line as a black-white boundary, each trapezoid can have smoothly varying gray levels such that the dashed line is 50% dark and the darkness reaches 0% and 100% at the parallel segments of the trapezoid boundary (thick lines in the figure). Image Averaging using this preprocessing strategy with tolerance $\epsilon = \frac{1}{3}$ gives the *Smooth-Shading* algorithm.

In order to make use of images with smooth-shaded edges as described above, we need to generalize the idea of rasterizing the images at a higher resolution and adding up the rasterizations. Consider a horizontal scan line passing through such an image. The darkness is a piecewise-linear function of the x coordinate along the line and the slope discontinuities occur at points where the scan line crosses trapezoid boundaries as shown in Figure 4.

We can rasterize the trapezoids to get (x, y, d) triples as before, but the d values should represent changes in the slope of the darkness function, not changes in the darkness itself. After rasterizing the trapezoids from all the input images, it is a simple matter to recreate the total darkness function from the x-sorted (x, y, d) triples on a particular scan line. The main difficulties are computing the correct d values and preventing accumulated rounding errors from getting too large. This gives Algorithm 1. (See also Figure 5).

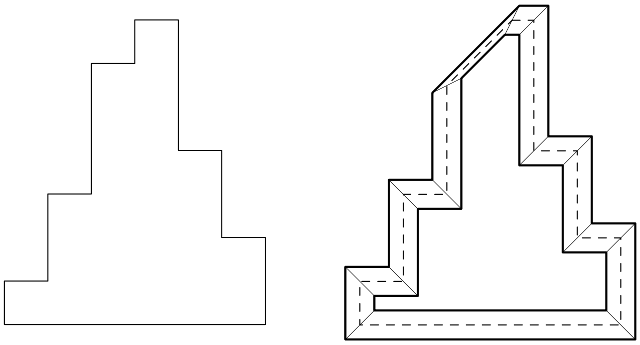


Figure 3: (a) An outline extracted from an input image; (b) the results of Hobby's smoothing algorithm with tolerance $\epsilon = \frac{1}{3}$. The smoothed polygonal outline is the dashed line and the other lines delimit trapezoids that define a range of alternative outlines.

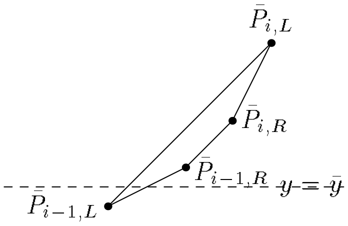


Figure 5: A typical trapezoid for Algorithm 1 and one of the scan lines $y = \bar{y}$ passing through it. The $y = \bar{y}$ line is shown dashed.

In order to avoid accumulating errors in ΔD in Step 8 of the algorithm, the third component of each triple in (1) should be stored as a fixed-point number. There can still be accumulated error in the total darkness D, but this is partially alleviated by properties of the trapezoids from [8]: trapezoids with a large horizontal extent tend to have $Y_{iL} = Y_{i-1,L}$ and $Y_{iR} = Y_{i-1,R}$ so that $d' = d''$ in (1). (The bottommost trapezoid in Figure 3b is an example of this.)

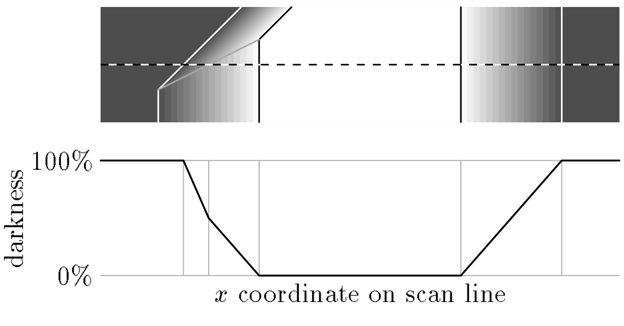


Figure 4: A section of a smooth-shaded image and the trapezoids that defined it. The graph shows the relative darkness along the dashed scan line.

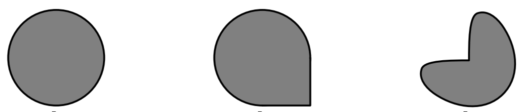
3 Experimental Results

We have experimented with the three special test shapes shown in Figure 6 and two Latin characters: the Helvetica capital 'R' and Times Roman capital 'R'. All trials were run on input images pseudorandomly generated by a program implementing the parameterized image defect model described in [2]. The degraded images were at a nominal text size of 8 point, and imaged at a spatial sampling rate of 300 pixel/inch. The discrepancies between approximated and ideal boundaries are expressed in units of input pixel-width. In order to estimate the algorithm's asymptotic performance, randomized trials on 250 input images were repeated 25 times and their means and standard errors computed.

Table 1 reports the results of experiments on images degraded by uniformly randomized spatial sampling error. It shows how the three algorithms performed on the test shapes and on the Helvetica and Times-Roman 'R's. The smooth-shading algorithms

Algorithm 1 How to do the "add up and threshold" step given a set of input outlines, a scale factor σ and a threshold T.

1. Initialize $j \leftarrow 1$.
2. Apply the Hobby's smoothing algorithm [8] to the j-th outline, obtaining $P_{iL} = (X_{iL}, Y_{iL})$ and $P_{iR} = (X_{iR}, Y_{iR})$ for $i = 0, 1, 2, \ldots, n_j$.
3. Let $\bar{P}_{iL} = \sigma P_{iL}$ and $\bar{P}_{iR} = \sigma P_{iR}$ for each i. Then initialize $i \leftarrow 1$.
4. Find all integers $\bar{y}$ such that the trapezoid $\bar{P}_{i-1,R}\bar{P}_{iR}\bar{P}_{iL}\bar{P}_{i-1,L}$ intersects the line $y = \bar{y}$ and execute Step 5 for each pair of intersection points $(x', \bar{y})$ and $(x'', \bar{y})$.
5. Compute darkness values d' and d'' for $(x', \bar{y})$ and $(x'', \bar{y})$ by letting the darkness be 1 at $\bar{P}_{i-1,L}$ and $\bar{P}_{iL}$, 0 at $\bar{P}_{i-1,R}$ and $\bar{P}_{iR}$, and varying linearly in between. Then store triples
$$\left(x', \bar{y}, \frac{d'' - d'}{|x'' - x'|}\right) \quad \text{and} \quad \left(x'', \bar{y}, \frac{d' - d''}{|x'' - x'|}\right). \tag{1}$$
6. If $i < n_j$, increment i and go back to Step 4.
7. If there are more outlines, increment j and go back to Step 2.
8. For each $\bar{y}$ such that there are triples with $y = \bar{y}$, sort the triples by x and scan them in order, maintaining total darkness D and rate of change of darkness ΔD. Each time the total darkness crosses the threshold T, output the point $(x, \bar{y})$ where this happens.



Test shape 1 Test shape 2 Test shape 3

Figure 6: Three test shapes that were used in the experiments.

did significantly better than naive averaging and pre-smoothing, and the test shapes were significantly easier than the 'R's.

input shape	smooth shading	naive averaging	pre-smoothing
Test 1	0.08 ± 0.01	0.11 ± 0.02	0.16 ± 0.03
Test 2	0.12 ± 0.02	0.18 ± 0.06	0.18 ± 0.01
Test 3	0.12 ± 0.01	0.16 ± 0.03	0.20 ± 0.04
Times R	0.47 ± 0.05	0.53 ± 0.05	0.51 ± 0.10
Helv. R	0.63 ± 0.04	0.68 ± 0.12	0.66 ± 0.11

Table 1: Error in recovering the ideal shape for various algorithms and input shapes under uniform spatial sampling error. Entries of the form $\mu \pm \sigma$ mean that the mean is μ and the standard error is σ.

The test runs used Algorithm 2 from Section 4 to deal with the sharp corners in the test shapes. This performed as well as the analysis leads us to expect on the three test shapes, but less well on the 'R's.

We also experimented with more complex document-image degradations, which have not yet yielded to analysis. Table 2 shows the effect of combining these degradations with uniform spatial sampling error. The degradation labeled S in the table is skew (rotation) varying normally with mean 0.0 and standard error 4.0 (degrees). For each image, this skew angle was passed to the image averaging algorithm, which attempted to correct for it; our motivation for this policy is that skew can often be estimated accurately from the complete page image.

input shape	Defect model		
	U	$U + S$	$U + B$
Test 1	0.08 ± 0.01	0.16 ± 0.01	0.19 ± 0.02
Test 2	0.12 ± 0.02	0.32 ± 0.04	0.26 ± 0.03
Test 3	0.12 ± 0.01	0.23 ± 0.05	0.31 ± 0.05
Times R	0.47 ± 0.05	0.66 ± 0.04	0.70 ± 0.02
Helv. R	0.63 ± 0.04	0.67 ± 0.05	1.44 ± 0.06

Table 2: Error in recovering the ideal shape for the smooth-shading algorithm with various input shapes and image defect models. In the column labels, U refers to uniform spatial sampling error, S refers to known random skew, and B refers to randomized blurring and thresholding. Entries of the form $\mu \pm \sigma$ mean that the mean is μ and the standard error is σ.

The degradation labeled B in Table 2 is blurring and thresholding: the blurring is a circularly symmetric Gaussian kernel whose standard error varies normally with mean 0.5 and standard error 0.5 (units of input pixels); and the threshold varies normally with mean 0.5 and standard error 0.125 (intensity). This did not increase the errors much relative to the results for the variable-skew trials, with one striking exception: on the Times-Roman 'R', the error was much larger (mean 1.50, σ 0.06); on inspection, this was clearly due to blunting of the tips of sharp serifs.

4 The Sharp-Corner Problem

The algorithms tested in Section 3 tend to produce poor results for shapes that have sharp corners unless the thresholding process is modified or a special post-processing step is used. Before explaining the remedies to this problem, we need to know what causes this effect and how its magnitude of depends on the preprocessing strategy.

Figure 7 shows what happens near a 90° corner with the Naive Averaging algorithm. Each input image is generated by sampling the test shape on a randomly shifted grid. Adding these images and thresholding them produces a rounded corner as shown by the dark shaded region in the figure. (Like all the figures in this section, Figure 7 is based on a coordinate system with positive y oriented upward.)

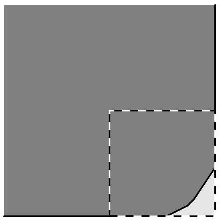


Figure 7: A magnified portion of the lower-right corner on Test shape 2. The dark-shaded region is the result of adding-up and thresholding with no preprocessing. It is superimposed on the original test shape (lightly shaded). The dashed square shows the size of one input pixel.

The Image Averaging algorithm attempts to align the resulting input images by shifting them so that their centroids coincide. If we make the optimistic assumption that this process correctly compensates for the random shift in the sample grid, we have the *perfect positioning assumption*. Under this assumption, the process of generating input images followed by Naive Averaging is equivalent to the following:

1. Select n random points $P_1, P_2, \ldots, P_n$ in the unit square and let $P_i + \mathbb{Z}^2$ be the set of grid points obtained by adding integers to the x and y components of P_i.

2. For each $(\bar{x}, \bar{y})$ and each $i \le n$, the square

$$\bar{x} - \frac{1}{2} \le x < \bar{x} + \frac{1}{2}, \qquad \bar{y} - \frac{1}{2} \le y < \bar{y} + \frac{1}{2} \tag{2}$$

contains a unique point from $P_i + \mathbb{Z}^2$. There are n such points for each $(\bar{x}, \bar{y})$. Color $(\bar{x}, \bar{y})$ black if at least $n/2$ of these are in the test shape.

This rule for deciding the color of $(\bar{x}, \bar{y})$ amounts to looking at a crude estimate of the area of the intersection of the square (2) and the test shape. Since the expected error in this estimate is $O(1/\sqrt{n})$, the limiting behavior for large n is to include those points $(\bar{x}, \bar{y})$ for which at least half of the square (2) is in the test shape. This defines a function F_{NA} that maps a test shape into the expected result of Naive Averaging.

Since any line through the center of a square divides it into two equal pieces, F_{NA} maps any half plane into itself. Now consider the 90° wedge W defined by

$$x \le 0, \qquad y \ge 0 \tag{3}$$

as shown in Figure 8a. If $\bar{x} \le -\frac{1}{2}$, the intersection of (2) and W has area $\ge \frac{1}{2}$ when $y \ge 0$. If $\bar{y} \ge \frac{1}{2}$, the intersection is at least 50% covered when $x \le 0$. If $\bar{x} > \frac{1}{2}$ or $\bar{y} < -\frac{1}{2}$, the intersection area is always 0. Hence $F_{NA}(W)$ must equal W except possibly for points $(\bar{x}, \bar{y})$ in a unit square centered on the origin. (This is the dashed square in Figure 8a.)

If $(\bar{x}, \bar{y})$ is in the dashed square, the intersection of (2) with W has area

$$\left(\frac{1}{2} - \bar{x}\right)\left(\frac{1}{2} + \bar{y}\right). \tag{4}$$

Setting this equal to $\frac{1}{2}$ gives a hyperbola that passes though $(-\frac{1}{2}, 0)$ and $(0, \frac{1}{2})$. This is the curved boundary of the dark shaded region in Figure 8b. Note how the experimental result in Figure 7 agrees with computed shape shown in Figure 8b.

The point is that the expected asymptotic result $F_{NA}(W)$ deviates from W by a certain fraction of an input pixel and this distance depends only on the shape of W. The following theorem formalizes the idea that the result of Naive Averaging on W approaches $F_{NA}(W)$ as the number of input images approaches infinity.

Theorem 4.1 *For any region W and any probability ρ, there exist families of regions W_n^- and W_n^+ that satisfy $W_n^- \subseteq F_{NA}(W) \subseteq W_n^+$, approach $F_{NA}(W)$ as n approaches ∞, and have the following properties: for any point $P^+ \notin W_n^+$, generating n input images by sampling W with randomly shifted grids*

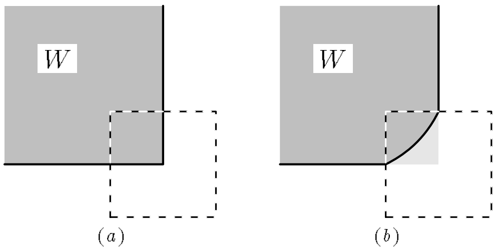


Figure 8: (a) A 90° wedge W and a region (dashed) where $F_{NA}(W)$ might disagree with W; (b) The expected result $F_{NA}(W)$ of Naive Averaging (dark shaded) and the difference $W \setminus F_{NA}(W)$ (lightly shaded). Wedge W extends to infinity leftward and upward.

and doing Naive Averaging under the perfect positioning assumption produces a result W_{NA} where the probability that $P^+ \in W_{NA}$ is at most ρ; similarly, any $P^- \in W_n^-$ is outside of W_{NA} with probability at most ρ.

Proof. We define W_n^- and W_n^+ in terms of parameters yet-to-be-chosen parameters α_n^- and α_n^+. Let W_n^- be the set of all $(\bar{x}, \bar{y})$ for which the intersection of W with the square (2) has area at least α_n^-; region W_n^+ is similar except the area bound is α_n^+. This gives $W_n^- \subseteq F_{NA}(W) \subseteq W_n^+$ whenever $\alpha_n^- \geq \frac{1}{2} \geq \alpha_n^+$. Furthermore, W_n^- and W_n^+ approach $F_{NA}(W)$ if α_n^- and α_n^+ approach $\frac{1}{2}$ as n approaches ∞.

It remains to show that α_n^+ and α_n^- can be chosen so that $P^+ \in W_{NA}$ and $P^- \notin W_{NA}$ with arbitrarily small probability. At $(\bar{x}, \bar{y}) = P^+$, the fraction α of square (2) within W is at most α^+, and is $< \frac{1}{2}$. The probability that W contains at least $n/2$ of the n sample points in square (2) is

$$\sum_{k \geq n/2} \binom{n}{k} \alpha^k (1-\alpha)^{n-k} < 2^n \sum_{k \geq n/2} \alpha^k (1-\alpha)^{n-k}$$

$$= \frac{2^n \alpha^{\lceil n/2 \rceil} (1-\alpha)^{\lfloor n/2 \rfloor}}{1 - \frac{\alpha}{1-\alpha}} \qquad (5)$$

$$\leq \frac{2^n \alpha^{n/2} (1-\alpha)^{n/2}}{\frac{1-2\alpha}{1-\alpha}} \leq \frac{(4\alpha - 4\alpha^2)^{n/2}}{1-2\alpha}. \qquad (6)$$

For any fixed $\alpha < \frac{1}{2}$, this approaches zero as n approaches ∞. Hence solving

$$\frac{(4\alpha_n^+ - 4(\alpha_n^+)^2)^{n/2}}{1 - 2\alpha_n^+} = \rho$$

for α_n^+ yields a function that approaches $\frac{1}{2}$ as n approaches ∞. Since (6) bounds the probability that $P^+ \in W_{NA}$, this α_n^+ satisfies the theorem.

The probability that $P^- \notin W_{NA}$ is the probability that more than $n/2$ of the sample points in square (2) are outside of W when $(\bar{x}, \bar{y})$ is such that at most $1 - \alpha_n^-$ of the square is outside W. Choosing $\alpha_n^- = 1 - \alpha_n^+$ makes this probability at most ρ as required. □

4.1 The Asymptotic Behavior of Presmoothing and Smooth Shading

We have analyzed Naive Averaging under the perfect positioning assumption with only spatial sampling error. Even under these favorable conditions, it approaches something different from the desired shape as the number of input images approaches infinity. Do the Presmoothing and Smooth Shading algorithms suffer from similar defects?

First, consider the presmoothing algorithm. It is harder to analyze than Naive Averaging because the polygonal smoothing algorithm expands the area affected by each sample point. In the simple case of the 90° wedge (3), the analysis is easy because the smoothing algorithm leaves such 90° corners unchanged. Hence, the previous analysis holds and the expected result as the number of input images n approaches infinity deviates from the 90° wedge as shown in Figure 8.

Now consider the Smooth Shading algorithm for the 90° wedge W given by (3) under the perfect positioning assumption. The preprocessed image differs from that of the Naive Algorithm in that the sharp edges shown in Figure 9a are replaced by gradual shading from black to white over some distance 2ϵ as shown in Figure 9b. (The recommended value for ϵ in Section 2.3 is $\frac{1}{3}$.)

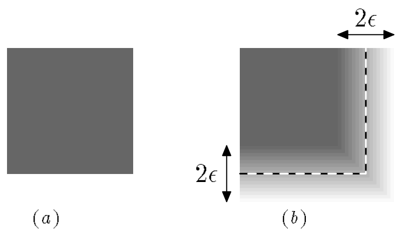


Figure 9: (a) A 90° wedge; (b) the effect the preprocessing used by the Smooth Shading algorithm. The dashed line shows the corresponding position of the edge of the 90° wedge.

For Naive Averaging, the expected darkness at a point $(\bar{x}, \bar{y})$ before the thresholding step was the area of the 90° wedge W inside the $(\bar{x}, \bar{y})$ centered unit square (2). For Smooth Shading, the corresponding rule is similar except that the 90° wedge is replaced by the smooth-shaded version in Figure 9b and the area becomes the integral of the shading density over the square.

For example if $(\bar{x}, \bar{y}) = (-\frac{1}{13}, \frac{1}{3})$ and $\epsilon = \frac{1}{3}$, the shading density functions for (2) are as in Figure 10. There are three subregions where the shading density is nonzero, and a different shading density function applies within each subregion as indicated in the figure. For the subregion labeled $\frac{1+3y}{2}$, the integral is

$$\begin{aligned}
&\int_{-1/6}^{1/3} \left(\frac{15}{26} - y\right)\left(\frac{1+3y}{2}\right) dy \\
&\quad = \int_{-1/6}^{1/3} \frac{15}{52} + \frac{19y}{52} - \frac{3y^2}{2}\, dy \\
&\quad = \frac{15}{104} + \frac{19}{1248} - \frac{1}{48} = \frac{173}{1248}
\end{aligned}$$

Similar computations show that the integral of $\frac{1-3x}{2}$ over the indicated region is $\frac{23}{96}$ and the integral of 1 over the indicated rectangle is $\frac{19}{156}$ This gives a total of $\frac{173}{1248} + \frac{23}{96} + \frac{19}{156} = \frac{1}{2}$, indicating that thresholding at $\frac{1}{2}$ places $(\bar{x}, \bar{y}) = (-\frac{1}{13}, \frac{1}{3})$ is on the boundary.

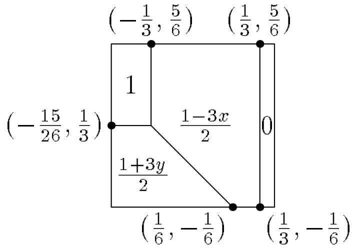


Figure 10: Shading density functions for the square (2) with $(\bar{x}, \bar{y}) = (-\frac{1}{13}, \frac{1}{3})$ and $\epsilon = \frac{1}{3}$.

Since $(-\frac{1}{13}, \frac{1}{3})$ is not on the boundary of W, the expected result of Smooth Shading on the 90° wedge W differs from W by a constant amount, even as the number of inputs approaches infinity. It does not seem worthwhile to try to derive expressions that define the boundary of the expected result W' of Smooth Shading on W under the perfect positioning assumption with only spatial sampling error, but the authors' numerical experiments indicate that the closest approach to $(0, 0)$ is approximately $(-0.181, 0.181)$ and that the total area of $W \setminus W'$ is approximately 0.091.

4.2 Restoring Sharp Corners

The function $F_{NA}(W)$ that models the effects of spatial sampling error followed by Naive Averaging is equivalent to blurring and then thresholding. Specifically, the blurring function is to convolve with a unit square of uniform density. This suggests replacing the thresholding step with deblurring followed by thresholding. The motion deblurring algorithm of Lee and Vardi is appropriate for this application [10, 14].

Alternatively, we can try to invert the function F_{NA} or whatever function is appropriate for the form of Image Averaging that is being used. This pseudo-inverse can be applied after smoothing the outlines extracted from the thresholded image as a final post-processing step. This strategy was adopted for the following reasons:

1. Smooth Shading and Presmoothing are hard to model with blurring functions, and these strategies performed better than Naive Averaging in our tests.

2. Dropping the perfect positioning assumption and allowing error sources other than spatial sampling error makes it hard to know what blurring kernel to use.

3. Post-processing the final outlines requires significantly less space and run time than deblurring would.

As the above points imply, correcting for the sharp corner problem is necessarily a somewhat heuristic procedure. We just need to start with something that has parameters to choose and does roughly the opposite of what F_{NA} does. In order to get a better understanding of F_{NA}, consider the region W that satisfies

$$y > m_1 x, \quad y > m_2 x, \quad \text{where} \quad -1 < m_1 < m_2 < 1$$

as shown in Figure 11. We need an expression for the area of the square (2) within W when $|\bar{x}| < 1$ and $(\bar{x}, \bar{y})$ is such that W contains $(\bar{x} \pm \frac{1}{2}, \bar{y} + \frac{1}{2})$ but neither of the points $(\bar{x} \pm \frac{1}{2}, \bar{y} - \frac{1}{2})$.

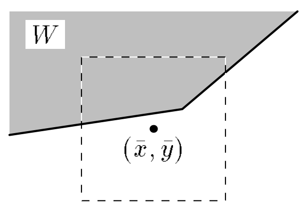


Figure 11: The region W for the derivation of F_{NA} with the square (2) dashed. In this case $m_1 = 0.15$ and $m_2 = 0.85$. Note that W extends to infinity leftward and upward.

This area is

$$\int_{\bar{x}-1/2}^{0} \bar{y}+\frac{1}{2}-m_1 x\,dx+\int_{0}^{\bar{x}+1/2} \bar{y}+\frac{1}{2}-m_2 x\,dx$$
$$=\bar{y}+\frac{1}{2}+\frac{m_1(\bar{x}-\frac{1}{2})^2}{2}-\frac{m_2(\bar{x}+\frac{1}{2})^2}{2}$$

Setting this equal to $\frac{1}{2}$ gives the boundary of $F_{NA}(W)$ as

$$\begin{aligned}\bar{y} &= \frac{m_2(\bar{x}+\frac{1}{2})^2}{2}-\frac{m_1(\bar{x}-\frac{1}{2})^2}{2}\\ &= \frac{m_2-m_1}{2}\bar{x}^2+\frac{m_1+m_2}{2}\bar{x}+\frac{m_2-m_1}{8}.\end{aligned} \tag{7}$$

This hits the boundary of W at $(-\frac{1}{2},-\frac{1}{2}m_1)$ and $(\frac{1}{2},\frac{1}{2}m_2)$ and passes through $(0,\frac{m_2-m_1}{8})$.

Now consider the total area inside W but below $F_{NA}(W)$. This is the integral of (7) on $-\frac{1}{2}\le\bar{x}\le\frac{1}{2}$ minus the integral of $\max(m_1x,m_2x)$ over the same interval. Integrating (7) gives

$$\frac{m_2-m_1}{24}+0+\frac{m_2-m_1}{8}$$

and

$$\int_{-1/2}^{1/2}\max(m_1x,m_2x)\,dx$$
$$=\int_{-1/2}^{0} m_1x\,dx+\int_{0}^{1/2} m_2x\,dx=-\frac{m_1}{8}+\frac{m_2}{8}.$$

Thus the area of $W\setminus F_{NA}(W)$ is $\frac{m_2-m_1}{24}$.

This suggests that a pseudo inverse of $F_{NA}(W)$ should add area near each convex corner of W and subtract area near each concave corner. The amount of area should be roughly proportional to the angle at the corner. Algorithm 2 shows one way to perform this operation on a polygonal outline. It adds γ units of area per radian, where γ is a parameter to be determined empirically. For small angles, the above argument suggests additional area $\frac{m_2-m_1}{24}$ for an angle of approximately $\frac{m_2-m_1}{1+m_1m_2}$ so that $\gamma\approx\frac{1+m_1m_2}{24}$. This ranges from $\frac{1}{24}$ to $\frac{1}{12}$ depending on m_1 and m_2. We also found an area difference of 0.091 for Smooth Shading with an angle of $\frac{\pi}{2}$. This suggest $\gamma\approx\frac{1}{17}$.

5 An Application

We applied an image averaging method (smooth shading) to a challenging OCR problem, a selection from which is shown in Figure 12. This consists of ten pages of text which appears to have been typewritten then photo-offset: the image quality is variable

---

Algorithm 2 An algorithm for postprocessing the smoothed outlines from the thresholded image to correct for the sharp corner problem by adding γ units of area per radian.

---

1. Let $\gamma'=\gamma/20$ and repeat Steps 2–6 20 times.
2. For each polygon edge, compute a shift amount $s_i=\gamma'(\theta_i+\phi_i)/(2l_i)$ where l_i is the length of the edge and θ_i and ϕ_i are the angles at the surrounding vertices.
3. For each edge, find the line ℓ_i obtained by shifting the edge perpendicularly s_i units to its right.
4. For each vertex v_i, find the point v_i' where the ℓ_i and ℓ_j for the edges incident on v_i intersect. This forms a new polygon whose edges are parallel to those of the old polygon.
5. For each edge of the new polygon whose orientation is opposite that of the corresponding old polygon edge, collapse the edge to a point where the ℓ_i and ℓ_j for the surrounding edges intersect.
6. Repeat Step 5 until all the edge orientations agree. Then update the old polygon to equal the new polygon.

---

and often low. The text is a collection of 50 puzzle cryptograms for hobbyists; as a result, contextual constraints are unusually weak. The "body text"—setting aside the headers—contains 23684 characters including word-spaces. The single fixed-pitch typewriter face remains unidentified (we have not attempted to match it with the faces in our collection of typographer's artwork).

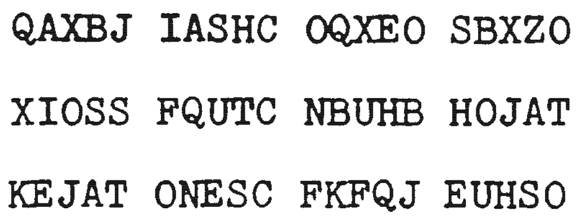


Figure 12: A magnified portion of the body text of a collection of puzzle cryptograms for hobbyists. The pages were originally typewritten, and then photo-offset.

We processed these images in two phases. In the first phase, we ran our experimental page reader using a classifier trained on twenty fixed-pitch faces. This involved page layout, fixed-pitch processing, character classification, shape-directed resegmentation, and crude contextual filtering (*i.e.* each space-delimited word was expected to be either all upper case alphabetic or all numeric). We measured the "classification error" of this phase as follows. First, we semi-manually identified every pair of strings of characters $\langle T, E\rangle$ where T is from the ground truth and E is the string erroneously substituted for T by the page reader. T or E may include spaces and may be empty. Each $\langle T, E\rangle$ pair is locally minimal in that T and E share no character. We adopted the policy that each pair contributes to the error count a score equal to the maximum of the lengths of T and E, unless that number exceeds 10, in which case it is ignored (forced to zero), on the somewhat arbitrary assumption that the pair results from a layout, rather than a classification, mistake. Under this policy we counted 1706 classification errors, for an nominal error rate of 7.2% of the characters.

In the second phase, we attempted to *improve the error rate completely automatically* by applying the image averaging algorithm, as follows. We sorted the character images from the first phase by their top-choice character labels (a subset of the images labeled "I" is shown in Figure 13. No attempt, automatic or manual, was made to set aside erroneously labeled images.

Then, for each character label, all of its images were given as input to the image averaging algorithm.

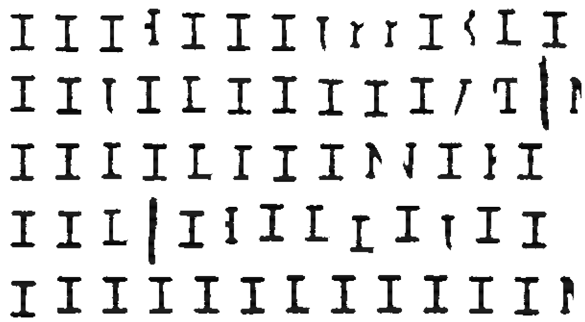


Figure 13: These are 67 magnified images selected from the total of 1009 labeled "I". Many, but not most, are mislabeled. Some are fragments of characters due to missegmentation, and some are pencil annotations.

Each output of the image averaging algorithm—the entire alphabet is shown in Figure 14.—was then used as an "ideal artwork" seed for input to a pseudorandom generator that wrote 150 degraded images (Figure 15 shows examples for "I"). These were used to train a classifier which was used to read the pages a second time. The output was then scored, as described above. The result was a *20% reduction in error*, to 1376 errors, or nominal error rate of 5.8%.

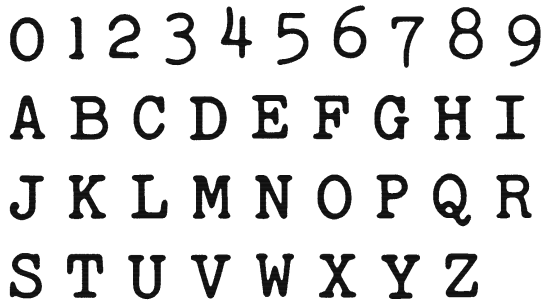


Figure 14: The body text consisted only of upper case alphabetics and numerals. The "I" shown here should be compared with the images in Figure 13, a subset of those from which it was automatically inferred.

We do not yet know how to characterize the circumstances under which this strategy, of inferring prototypes from sets of possibly imperfectly labeled images, will be safe. All we can say at present is that, in this particular application, faced with an average error rate of about 7%, it worked well. If a user were willing to oversee the method, a quick glance at the inferred prototypes should be sufficient to guard against disaster.

Another strategy for automatically retraining a classifier on imperfectly labeled data has been de-

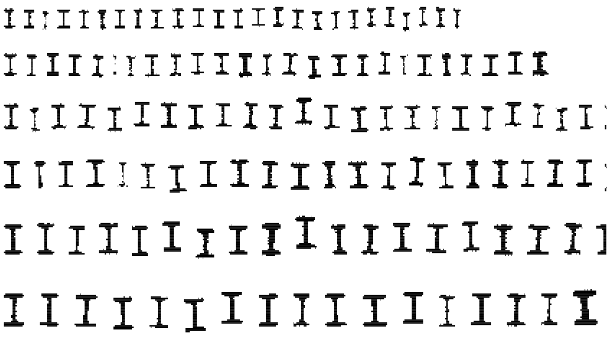


Figure 15: For each letter of the alphabet, 150 images were generated using the prototype artwork inferred by the image averaging algorithm.

scribed in [4]. The present work departs from this in (a) not using the real images directly; and (b) relying heavily on synthetic images from an image degradation model. In future work, we hope to test whether these differences are relatively advantageous and disadvantageous. It should be straightforward to pursue both of these strategies simultaneously on the same document—potentially combining their advantages—simply by mixing the real and synthetic data before retraining.

Perhaps, in the future, we can find ways to infer automatically, not merely faithful prototypes for character artwork, but the parameters of the document's image degradation as well: this may allow even more effective automatic bootstrapping.

6 Summary

We have described an algorithm for image restoration which is specially adapted to applications in document image analysis. It accepts as input a set of bilevel images, and attempts to infer from them a single bilevel image at a higher spatial sampling rate. This output image should be similar to the ideal prototype of which the input images are degraded derivatives. Our basic approach follows prior art by "averaging" the images: the input images are superimposed, intensities are added up, and the result is thresholded. We show that a naive implementation of this is inferior to an algorithm which specially preprocesses the input images and postprocesses the result. The analysis concentrates on degradations due to uniform spatial sampling error. Experimental trials on a variety of degradations and symbol shapes are described. Results on artificial test shapes illustrate the advantages of the new algorithm, particularly in ameliorating rounding effects at sharp angles along the boundary.

Future work should include analysis and experiments on a wider variety of symbol shapes and degradations, including images lifted from actual documents. There also needs to be more work on the sharp corner problem, particularly the alternative mentioned in Section 4.2 involving deconvolution. Algorithm 2 is ad hoc and performed poorly on features such as the serifs on the Times Roman '**R**'.

7 Acknowledgement

We are grateful to Jim Reeds for bringing the application to our attention, providing the ground truth and scoring, and commenting on a draft of this paper. Jonathan Hull and David Lee kindly pointed out several references. Theo Pavlidis and George Nagy assisted our search for other references.

References

[1] Gideon Avrahami and Vaughan Pratt. Subpixel edge detection in character digitization. In Robert A. Morris and Jacques André, editors, *Raster Imaging and Digital Typography II*, pages 54–64. Cambridge University Press, 1991.

[2] H. Baird. Document image defect models. In *Proc. of IAPR Workshop on Syntactic and Structural Pattern Recognition*, pages 38–46, Murray Hill, NJ, June 1990.

[3] H. S. Baird. Document image defect models and their uses. *Proc., IAPR 2nd Int'l Conf. on Document Analysis and Recognition*, October 1993.

[4] H.S. Baird and G. Nagy. A self-correcting 100-font classifier. In *Proc., IS&T/SPIE Symposium on Electronic Imaging: Science and Technology*, San Jose, CA, February 1994.

[5] N. Billawala, P. E. Hart, and M. Peairs. Image continuation. In *Proc., 2nd International Conference on Document Analysis and Recognition*, pages 53–57, Tsukuba Science City, Japan, October 1993.

[6] P. Cheeseman, R. Kanefsky, R. Hanson, and J. Stutz. Super resolved surface reconstruction from multiple images. Technical Report FIA–93–02, NASA Ames Research Center, Moffett Field, CA 94035–1000, February 1993.

[7] Trevor Hastie, Patrice Simard, and Eduard Säckinger. Learning prototype models for tangent distance. In Gerald Tesauro, David S. Touretzky, and Todd K. Leen, editors, *Neural Information Processing Systems*, volume 7, pages 999–1006. Morgan Kaufmann Publishers, San Mateo, CA, 1995.

[8] John D. Hobby. Polygonal approximations that minimize the number of inflections. In *Proceedings of the Fourth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 93–102, January 1993.

[9] D. E. Knuth. *METAFONT the Program*. Addison Wesley, Reading, Massachusetts, 1986. Volume D of *Computers and Typesetting*.

[10] D. Lee and Y. Vardi. Experiments with maximum likelihood method for image motion deblurring. In K. V. Mardia, editor, *Advances in Applied Statistics*, volume II, pages 355–383. Abington, Carfax Pub. Co., 1994.

[11] George Nagy. R.P.I., Troy, N.Y., August 1995. personal communication.

[12] Timothy J. Schulz and Donald L. Snyder, editors. *Image reconstruction and restoration*. SPIE–the International Society for Optical Engineering, Bellingham, Wash., USA, 1994.

[13] Yong-Shul Shin, R. Sridhar, V. Demjanenko, P.W. Palumbo, and J.J. Hull. Contrast enhancement of mail piece images. In *Proc., 1992 SPIE/IS&T Symposium on Electronic Imaging: Science and Technology*, pages 27–37, San Jose, CA, February 1992.

[14] Y. Vardi and D. Lee. From image deblurring to optimal investments: Maximum likelihood solutions for positive linear inverse problems. *Journal or the Royal Statistical Society, Series B*, 55(3):569–612, 1993.