G. Nagy, Preprocessing document images by resampling is error prone and unnecessary, Proc. SPIE Document Recognition and Retrieval, DRR XX, SPIE VOl. 8658-0U, San Francisco, February 2013

Preprocessing document images by resampling is error prone and unnecessary

George Nagy

DocLab†, Rensselaer Polytechnic Institute
110 Eighth Street, Troy, NY, 12180 USA

ABSTRACT

Integrity tests are proposed for image processing algorithms that should yield essentially the same output under 90 degree rotations, edge-padding and monotonic gray-scale transformations of scanned documents. The tests are demonstrated on built-in functions of the Matlab Image Processing Toolbox. Only the routine that reports the area of the convex hull of foreground components fails the rotation test. Ensuring error-free preprocessing operations like size and skew normalization that are based on resampling an image requires more radical treatment. Even if faultlessly implemented, resampling is generally irreversible and may introduce artifacts. Fortunately, advances in storage and processor technology have all but eliminated any advantage of preprocessing or compressing document images by resampling them. Using floating point coordinate transformations instead of resampling images yields accurate run-length, moment, slope, and other geometric features.

Keywords: Normalization, image rotation, convex hull, resampling, morphology

1. INTRODUCTION

Preprocessing that requires image resampling is error prone and often gratuitous. In the first part of this communication, some tests are proposed for detecting inaccurate preprocessing. In the second part, means are presented for avoiding resampling altogether. None of the material here is technically novel. As one of the reviewers pointed out, the effects of undetected inaccuracies are often too small to be of practical significance. Therefore this contribution is entirely polemical.

Inconsistencies in image processing results can often be traced to one of two causes. The first is rooted in the fact that 2-D co-ordinates can only be partially ordered. Therefore any sequence of processing pixels has an inherent bias that must be taken into account to avoid different results from row-by-row vs. column-by-column processing. The second culprit is edge anomalies in window-based algorithms. Such algorithms require special provisions when the window overlaps the border of the image. These provisions often require a great deal of code for each border. The anomalies resulting from faulty orientation or edge coding can often be spotted using the simple integrity tests that we describe below. Failing the proposed tests can reveal avoidable errors due to programming mistakes.

A more pernicious kind of error is caused by rounding off coordinates that were transformed for resampling an image. The resulting distortion can be reduced, but not eliminated, by using higher-order interpolation. Resampling is demonstrably undesirable and unnecessary for normalizing geometric features. A better approach is computation of features from exact (i.e., floating-point) transformation of the original pixel coordinates.

Although these problems also arise in other domains, the focus here is on document image analysis (DIA) where 90° rotations are seldom tested because of the predominant portrait orientation of documents, and where batch scanning occasionally results in unintended 180° rotations. Furthermore, the degradation due to resampling is most severe in bi-level images that are traditionally used for high-contrast documents because of lower storage and processing cost and high compressibility. Errors and inconsistencies in the preprocessing phase give rise to unexpected and sometimes inexplicable effects in feature extraction for document classification, duplicate detection, layout analysis, table processing, and OCR.

Integrity tests for image processing are discussed in Section 2, which also reports the results of some tests on Matlab image processing routines. Section 3 demonstrates the benefits of transforming pixel coordinates instead of resampling images.

2. INTEGRITY TESTS

Some integrity tests should be applied to every document image processing program. Failing a sanity check does not necessarily indicate that the software is unfit for the proposed use, but suggests caution. Small differences may matter: sometimes only a few pixels separate two pattern classes, like "1" and "l'. The following checks are widely applicable.

2.1 Invariance under 90° and 180° Rotation

Most image analysis routines with a scalar output should yield output on an image rotated by multiples of 90° that is identical or equivalent to their output on the original image. (An example of "equivalence" is counting the number of horizontal and vertical edges or line segments, where the counts should switch under a 90° rotation.) Although the illustrations below are based on bi-level images, the invariance requirement applies to gray-level images as well.

As an example, rotations of 90°, 180°, and 270° were applied to an image to test some built-in routines of the Matlab (R2010a) Image Processing Tool Box. The `rot(A, 90)` routine works perfectly. On the test bitmap of Fig. 1, the regionprops properties `Area, MajorAxisLength, MinorAxisLength, Eccentricity, Perimeter,` and `EulerNumber` all gave almost the same output (within 0.0000000001%) for every orientation. The only functions that failed were `ConvexArea` and `Solidity` (which uses the output of `ConvexArea`). The error is presumably due to inconsistent inclusion or exclusion of pixels centered on segments of the convex boundary with different orientations. Fig. 1 shows a connected component where the original and rotated filled-in convex hulls differ by 4 pixels. Matlab's convex hull computation was also inconsistent for every connected component of several Chinese characters, including the one in Fig. 2. Inconsistent values of the number of pixels enclosed by the convex hall (`Solidity)` are reported in Table 1. Although the differences are only a few pixels, there is no excuse for them.

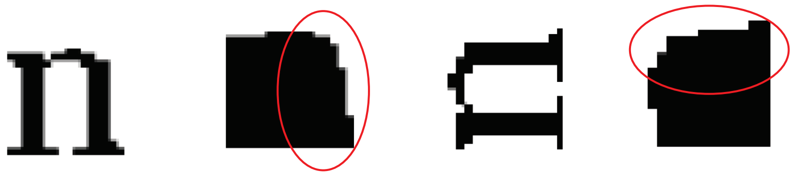

Fig. 1. Left to right: original n and its convex hull, rotated n and its convex hull (anomalies circled).

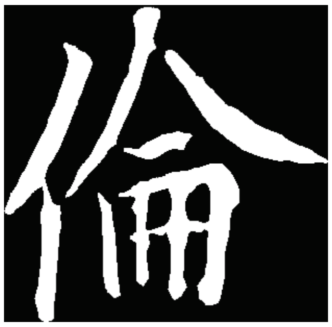

Fig. 2. A Chinese character with six connected components. The largest CC is the one on the bottom right.

Table 1. The table shows the convex area of the six components of a Chinese character under rotations. The circled values indicate a 27-pixel (~0.5%) difference in one component when the character is rotated 180°.

CC	0°	90°	180°	270°
1	1129	1132	1132	1129
2	1500	1493	1489	1495
3	5485	5480	5474	5479
4	5678	5665	5651	5664
5	12411	12401	12404	12414
6	12832	12835	12834	12831

Another common error arises when using morphological operators (which should commute with rotation if the kernel has the appropriate rotational symmetry). For example, consider *closing* the image of Fig. 2 with a square structuring element of size 50×50. Call the entire image A, and the closed image A _closed (Fig 3a). Now rotate A by 180° (A_rot), then *close* A_rot to obtain (A_rot_closed), and finally rotate A_rot_closed by 180°, resulting in A _rot_closed_rot (Fig. 3b). We would expect that A_closed ≡ A_rot_closed_rot, i.e. close(A) ≡ rotate(close(rotate(A))).. However, the two images differ by 50 pixels, as shown by their absolute difference in Fig. 3c.

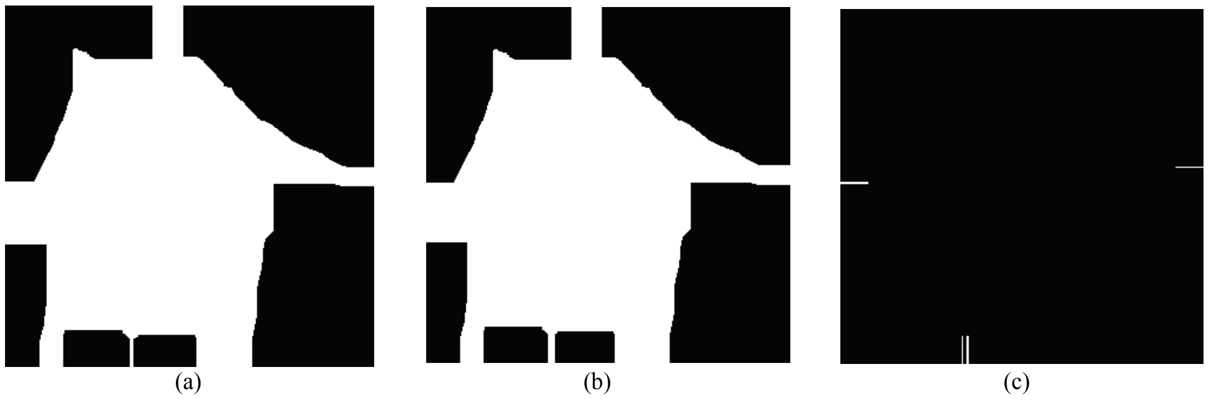

Fig. 3. Anomalies resulting from closing the image of Fig. 2 with a 50×50 structuring element. Image (a) should be the same as image(b). The absolute difference between them is displayed in (c)

The same phenomenon arises, as might be expected, with other operations like *erosion* and *dilation* and with combinations of such operations. In practice these anomalies may arise in *smearing*[1] an upside-down page image with a sequence of operators for layout analysis. (N.B. Casey, Wong and Wahl did *not* make this mistake.)

This effect is unavoidable with any structuring element whose origin does not fall on integer pixel coordinates. It is greatly aggravated when the origin is far from the structuring element's centroid. The solution is simple: use a 49×49 or a 51×51 pixel element, and make sure that the origin is at its centroid (Matlab automatically places it there—if it can.)

The best known examples of undesirable asymmetries are those connected with the *skeleton* of an image, the digital version of the medial axis. Ideally, a thinned image should be only a single pixel wide, and every foreground pixel should be equidistant from the nearest background pixels. It is clear that there are no such pixels in a bar that has a width of an even number of pixels. Most thinning/skeletonizing algorithm (several hundred have been published) make an arbitrary choice between left and right, or top and bottom. Therefore the resulting image will fail the proposed rotation test. Related problems arise with connectivity (see, however, Echardt and. Maderlechner [2] on reversible skeletons and their morphological interpretation). The skeleton of the image of Fig. 2 is shown in Fig. 4.

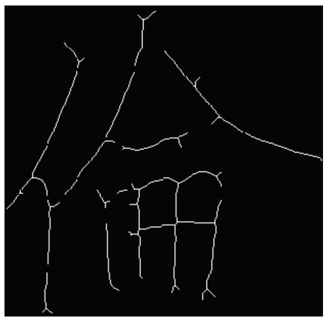

Fig. 4. Skeleton of the Chinese character image of Fig. 2.
The skeleton of a digital image is a useful but intrinsically asymmetric representation.

2.2 Invariance under Monotonic Remapping of the Reflectance to Grey-Value

Some scanners map *optical reflectance* to gray-value (typically 0-255), some map *reflective density* (the logarithm of reflectance).[3,4] The mapping can be modified by adjusting the scanner's brightness and contrast settings. These settings are seldom preserved in the output file. Even if calibration charts are scanned with each batch of documents (as they should be!), often the resulting information is eventually separated from the document stream. Then there is no way of knowing the difference in reflectance between a gray value of 110 and a gray-value of 150.

Scanning does not preserve contrast, as shown in Fig. 5. Furthermore, although the tails of the `e` and of the `c` appear to be the same in the original, they are different in the digitized version. This can happen even with a perfect ideal scanner because of the random placement of the glyphs with respect to the underlying sampling grid.[5]

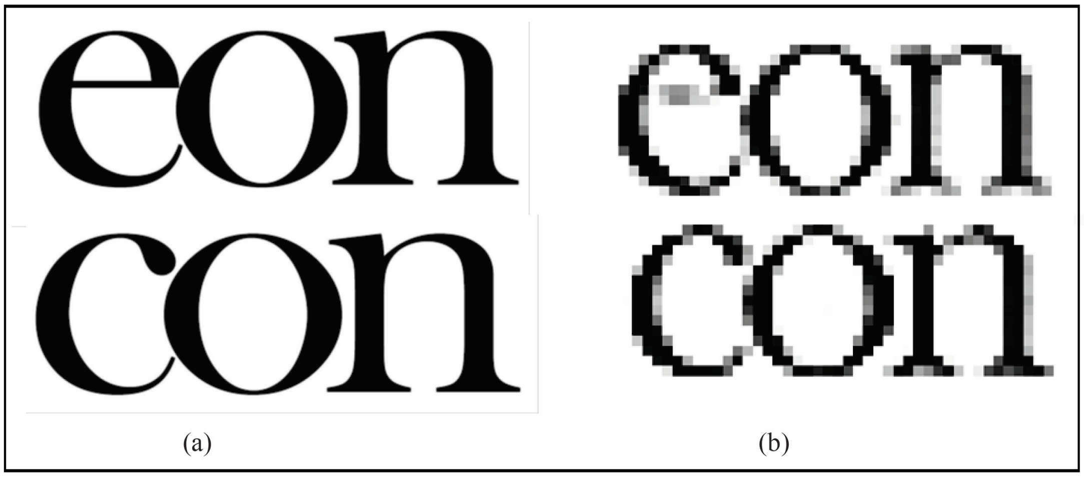

Fig. 5. "eon" and "con". (a) Original images (b) Gray-scale representation of scanned images. The broad range of gray-level values is due to the finite size of the point spread function. Pixel values at edges are the average of the foreground and background areas.[6]

The gray levels of the high-contrast test image of Fig. 5a were remapped with each of the following monotonic transformations:

(a) $y' = 40\ln(y+1)$ (b) $y' = 16\sqrt{y}$

(c) $y' = y^2/256$ (d) $y' = 7y/8 + 20$

Matlab's built in binarization function, `graythresh`, gave the same output as on the original scan only with the linear transformation (d). This function is based on Otsu's global binarization algorithm.

Fig. 6 shows an example where even local binarization may be unable to find a threshold that preserves the horizontal bar of the `e` in `eon`, yet leaves a gap between **`c`** and **`o`** in `con`.

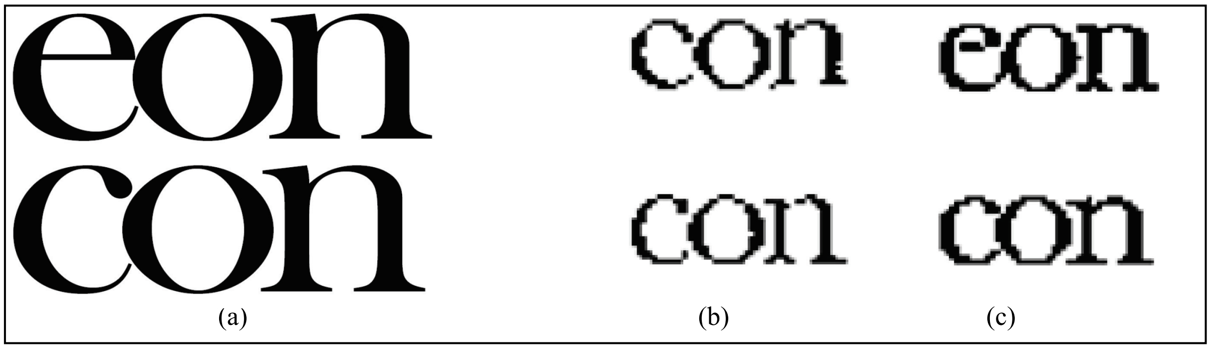

Fig. 6. "eon" and "con". (a) Original patterns, highly magnified, on the same page. The page was scanned at 100 dpi with 8-bit depth. (b) The chosen binarization threshold preserves the gaps between **`e`** and **`o`**, and between **`c`** and **`o`**, but erases the bar of the **`e`**.
(c) A lower threshold produces the opposite effect: it preserves the bar of the **`e`**, but makes **`e`** and **`o`**, and **`c`** and **`o`**, touch.

2.3 Invariance under Padding

In window-based image processing, careful coding is required to avoid anomalies at the edges of the image. A common expedient is to pad the image (e.g., with Matlab `padarray`). Binarization with `graythresh` did not change when the image was extended with white pixels, but padding with black pixels raised the binarization threshold slightly. Background padding did not alter the computed areas, the parameters of the fitted ellipses, the perimeters, the Euler Numbers, or the convex areas.

2.4 Invariance under Moderate Changes in Spatial sSampling Rate.

Document categorization, layout analysis, component detection, and OCR should give essentially the same output whether the image is scanned at 290 dpi or 310 dpi. In some large-scale document digitization operations, the resolution is set to give adequate representation to the thinnest stroke on the page.[7] In addition to invariance under changes in dpi, the results should be invariant to a change in page size from A4 to US Letter Size, or in type size from 10pt to 12pt.

Even if the scanner software inserts the original dpi setting into the scanned image file by the scanner, it often disappears during file-format transformations (e.g., TIF to PDF or PNG), which makes it impossible to calculate the physical size of the scanned page, type size, or line spacing. Good digitization practice requires associating every transformation applied to the original hardcopy with the document image file.

3. NORMALIZING FEATURES VS. NORMALIZING IMAGES

Preprocessing document images by resampling is widely recommended for size, rotation or skew normalization.[8]. For example, a recent guide for practitioners[9] states that: "After the skew angle of the page has been detected, the page must be rotated in order to correct this skew". Resampling is usually accomplished by interpolating the pixel values in some neighborhood of the source coordinates of the transformed pixel, and then rounding or truncating the result. Matlab offers nearest-neighbor, bilinear and bicubic interpolants.

Thorough analyses of resampling are available in a general picture processing context[10,11,12], but not specifically for document image analysis. Resampling bi-level images is especially prone to anomalies. Size and skew normalization by resampling may make it more difficult to segment an image because it can change the number of connected components.

It is generally agreed that features invariant to scale, gray scale and skew are more reliable than image normalization during a preprocessing step. Many geometric features invariant to translation, scale and rotation have been proposed and used, including Fourier-Mellon transforms, Hu, Zernike and Krawtchouk moments[13,14,15,16,17], functions of the number of stroke crossings along transects[18], and, by definition, all "topological "features[19]. Nevertheless, resampling remains a common approach to normalization because it simplifies subsequent extraction of features lacking an invariant formulation.[20,21] Examples of non-invariant features include template matching and N-tuples. Furthermore, most of the features listed are not invariant to rubber-sheet transformations.

Inaccuracies in feature extraction from resampled images are the intrinsic consequence of the irreversible mapping of real numbers into integers. The mapping can be avoided by retaining the floating point values of the transformed coordinates. Normalizing the features can then take into account the size, skew and gray scale of the original image without requiring the truncation of real numbers. Even though often the difference between approximation and exact calculation is small, feature normalization does not need experimental justification because it is a dominant alternative (i.e., it is always superior or equal) to resampling.

When invariant features cannot be used, feature normalization should be accomplished by computing the features from the exact values of the transformed coordinates. We illustrate this with simple examples.

Consider the 1-D "image"	A =0110011110
	x = 0123456789
Scaling by 0.8 yields	B = 01001110
	x' = 01234567 via co-ordinate transformation: $x = \lfloor 1.25x' + ½ \rfloor$

Scaling the coordinates exactly yields

x'' = 0.0, 0.8, 1.6, 2.4, 3.2, 4.0, 4.8, 5.6, 6.4, 7.2.

The length (2) of the second zero-run in B is twice that of the preceding one-run (1). In contrast, computing the ratio of run lengths from x", (4.0–2.4)/(2.4–0.8), results in the correct ratio of 1.0. The second geometric moment $\Sigma A(x)x^2$ of A is *179*, and of B ($\Sigma B(x')(x')^2$) it is *78*. If, however, we compute the geometric moment from $\Sigma A(x'')(0.8x'')^2$, the result is *114.56*. This is an exact result: $\sqrt{(114.56/179)} = 0.80$, Computing the moment feature from the resampled bitmap was off by more than 30%! (The second moment feature could of course be normalized by multiplying the value obtained on the original image by the square of the scaling factor.)

Computing moments from resampled bitmaps that correct skew (rotation or shear) is also error prone. A worst-case situation is the 45° rotation of a 2×2 digital square (Fig. 7), where the error in the second central moment $M^{2,0}$ is 100% (2×1^2 instead of $4\times(½)^2$). The geometric moments can, however, be computed exactly from the rotated coordinates. Direct methods have also been derived for geometric moments under rotation (analogous to the one shown for 1-D scaling) and other linear transformations, but not for perspective or rubber-sheet transformations.[22,23]

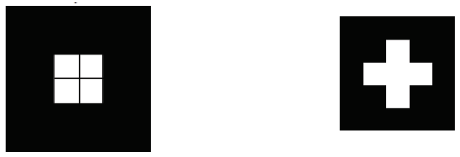

Fig.7. Digital square and rotated digital square. The centers of the white pixels of the square are $1/\sqrt{2}$ pixels from the origin. When the square is rotated 45°, the centers are 1 pixel from the origin. Therefore rotation introduces a significant difference in the moments. Small patterns and large rotations are common in camera-based OCR.

Non-linear shape normalization for Chinese handwritten text recognition[24,25,26], mapping pen coordinates for extracting features like turning angles[27], and document defect models[28], are all usually implemented via resampling. The error in computing features from the normalized bitmaps is reduced, but not eliminated, by higher spatial sampling rate and by scaling or rotating about the centroid of the image rather than the origin.

All normalized features based on geometric distance (*convex areas, perimeter lengths, stroke widths, moments, gradients, Hessians, directional derivatives* and *projections*), can be computed exactly using either invariant formulations or transformed coordinates instead of resampled bitmaps. If each coordinate is to be referenced several times, then the transformed coordinates of all the pixels on a whole page can be pre-computed and stored as floating-point numbers. Matlab takes less than 0.1 sec to compute 2000×3000 exact *x* and *y* rotation coordinates on a 2.5 GHz laptop. In contrast, rotating this bi-level image by resampling with nearest-neighbor interpolation takes Matlab 1.7 sec!

4. SUMMARY

Some simple tests were proposed to spot inconspicuous, generally easily fixed, and in the author's experience not uncommon, anomalies in document image processing programs. Such errors may occur even in well-established and widely-used programs such as Matlab routines. Even if the programs are perfectly correct, unexpected effects arise from the very nature of the digital grid.

Examples of inaccurate feature computation based on resampling images were presented. To prevent such inaccuracies, normalizing image features using exactly, and therefore reversibly, transformed coordinates was advocated in lieu of extracting the features from resampled bitmaps.

ACKNOWLEDMENT

Professors Henry S. Baird (Lehigh U.) and Elisa H. Barney Smith (BSU) gave helpful suggestions on early drafts. I am also grateful to the referees, whose comments suggested that the point of the submitted version needed additional clarification and emphasis.

REFERENCES

[1] Wong, K.Y., Casey, R.G. and Wahl, F.M. “Document analysis system”, IBM J. Res. Develop. 26, 647-656 (1982).

[2] Echardt, U. and Maderlechner, G., “Thinning Binary Pictures by a Labeling Procedure,” Proc. ICPR 11, 582-585 (1992).

[3 Nagy, G., "Optical Scanning Digitizers," *IEEE Computer* 16, #5, 13-25 (1983).

[4] AGFA, An introduction to digital scanning, Agfa-Gavaert N.V. Belgium (1994).

[5 Sarkar, P., Lopresti, D., Zhou, j., and Nagy, G., "Spatial Sampling of Printed Patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20, #3, 344-351 (1998).

[6] Barney Smith, E. H., “Characterization of Image Degradation Caused by Scanning,” *Pattern Recognition Letters* 19, Number 13, 1191-1197 (1998).

[7] Family & Church History Department, Digital Image Specification, Version 19a, Revised 30 April (2007).

[8] Ha, T .M. and Bunke, H., “Image processing methods for document image analysis”, in Handbook of *Character Recognition and Document Image Analysis* (H. Bunke and P.S.P. Wang, eds.), World Scientific (1997).

[9] Cheriet, M. , Kharma, N., Liu, C-L, Suen, C.Y, [Character Recognition Systems], Wiley-Interscience (2007).

10] Abdou, I.E. and Wong, K.Y. ,”Analysis of linear interpolation schemes for bi-level image applications,” *IBM J. of R&D* 26, 6, 667-680 (1982).

[11] Parker, J. A., Kenyon, R.V., Troxel, D.E., “Comparison of interpolating methods for image resampling,” *IEEE Trans. Medical Imaging* 2, 1, 31-39 (1983).

[12] Gonzalez, R.C. and Woods, R.E., [Digital Image Processing], Pearson (2008).

[13] Hu, M.K., “Visual pattern recognition by moment invariants,” *IRE Trans. Information Theory* 8, 2, 179-187 (1962).

[14] Teague, M.R., “Image analysis via the general theory of moments,” *JOSA* 70, Issue 8, 920-930 (1980)..

[15] Khotanzad, A. and Hong, Y.H.,“Invariant image recognition by Zernike moments,” *IEEE Trans. PAMI* 12, 5,489-497 (1990).

[16] Yap, P.T., Paramesran, R., and Seng-Huat, O., “Image analysis by Krawtchouk moments,” *IEEE Trans. Image Processing* 12, 11, 1367-1377 (2003).

[17] X. Wang, B. Yiao, J-F Ma, “Scaling and rotation invariant analysis approach to object recognition based on Radon and Fourier–Mellin transforms,” *Pattern Recognition* 40, 12 3503-3508 (2007).

[18] Ding, X., “Machine printed Chinese character recognition,” in *Handbook of Character Recognition and Document Image Analysis* (G. Bunke and P.S.P. Wang, eds.), World Scientific (1997).

[19] Nadler, M., and Smith,E.P., [Pattern Recognition Engineering], Wiley (1993).

[20] Kmiec, M., “New optimal character recognition method based on Hu invariant moments and weighted voting,” *Journal of Applied Computer Science* 19, 1, 33-5 (2011).

[21] Tatele, S., and Khare, A., “Character recognition and transmission of characters using network security,” Int’l J. *Advances in Engineering and Technology*, 11 (2011).

[22] Sivaramakrishna, R. and Shashidharf, N.S., “Hu’s moment invariants: how invariant are they under skew and perspective transformations?” *Proc. WESCANEX*, Winnipeg, 292-295 (1997).

[23] Pan, P., Zhu, Y., Sun, J.,and Naoi, S., “Recognizing characters with severe perspective distortion using hash tables and perspective invariants” Proc.. *ICDAR 2011*: 548-552 (2011).

[24] Yamada, H., Yamamoto, K., Saito, T., “A nonlinear normalization method for Kanji character recognition—line density equalization,” *Pattern Recognition* 23(9): 1023-1029 (1990).

[25] Liu, C-L, Sako, H., Fujisawa, H. “Handwritten Chinese Character Recognition: Alternatives to Nonlinear Normalization.” *ICDAR 2003*: 524-528 (2003).

[26] Uchida, S. and Sakoe, H. “A Survey of Elastic Matching Techniques for Handwritten Character Recognition,” *IEICE Transactions* 88-D(8): 1781-1790 (2005).

[27] Watt, S. and Dragan, L., “Recognition for Large Sets of Handwritten Mathematical Symbols,” Proc. ICDAR, Seoul (2005).

[28] Baird, H.S., “Document Image Defect Models” in H. S. Baird, H. Bunke, & K. Yamamoto (Eds.), *Structured Document Image Analysis*, Springer-Verlag 1992: Reprinted in L. O'Gorman & R. Kasturi (Eds.), *Document Image Analysis*, IEEE Computer Society Press (1995).