X_{2m}, the conditional probability density (32) does not depend on a_2. Hence a_{2m} in ML method, i.e., the value of a_2 minimizing (31), is obtained by the maximization of

$$f(A_{1nm}, B_{nm}; a_2, \overline{X}_{1nm}, X_{2m})$$

with respect to a_2. Using (30), (13) becomes

$$f_W(\overline{W}_{nm}; a_2, \overline{X}_{1nm}, X_{2m}) = f[H(\overline{W}_{nm}, \overline{X}_{1nm}); a_2, \overline{X}_{1nm}, X_{2m}]|J|, \quad (33)$$

where the Jacobian J does not depend on a_2. Then a_{2m} is equal to a^*_{2m} maximizing (13). This completes the proof for ML method.

For MPP method a_{2m} maximizing (19) is obtained by the maximization of

$$f_2(a_2)f(A_{1nm}, B_{nm}; a_2, \overline{X}_{1nm}, X_{2m})$$

with respect to a_2 and because of (33), a_{2m} is equal to a^*_{2m} maximizing (20). Q.E.D.

VI. Conclusion

A concept of two-stage parameter estimation in the identification of static system has been presented. It has been shown how the maximum likelihood method and the Bayesian approach can be applied to the problem under consideration. The estimation algorithms for simple special cases have been included to clarify the details. The comparison of the direct and two-stage approaches to the second-stage parameter estimation has been discussed. It has been shown that the sufficiency of the first-stage estimator is the sufficient condition of the equivalence of the two approaches for the methods described here. The necessity of this condition and the existence of the cases for which direct and two-stage estimations give different results are now the open questions. The concept of two-stage estimation can be easily extended to handle dynamic systems. Further studies are also required to clarify the comparison problem.

References

[1] Z. Bubnicki, *Identification of Control Plants*. Amsterdam-Oxford-New York: PWN-Warszawa: Elsevier Scientific, 1980.

[2] ——, "Problems of complex systems identification," in *Proc. of International Conference on Systems Engineering*, Lanchester Polytechnic, Coventry, England, 1980.

[3] ——, "On the multistage identification," *Syst. Sci.*, vol. 3, no. 2, pp. 207–210, 1977.

[4] H. Cramer, *Mathematical Methods of Statistics*. Princeton, NJ: Princeton Univ., 1946.

Some New Heuristics for Thinning Binary Handprinted Characters for OCR

F. W. M. STENTIFORD AND R. G. MORTIMER

***Abstract*—A standard thinning algorithm which consists of a matrix matching scheme [1] used in conjunction with a connectivity measure [2] is briefly described. Several new heuristics are introduced to obviate some of the common shortcomings of thinning algorithms when applied to handprinted data.**

Manuscript received July 10, 1981; revised June 21, 1982.

F. W. M. Stentiford is with British Telecom Research Laboratories, Department R14-1-1, Martlesham Heath, Ipswich, Suffolk, England IP5 7RE.

R. G. Mortimer is with Hatfield Polytechnic, College Lane, Hatfield, Herts, England.

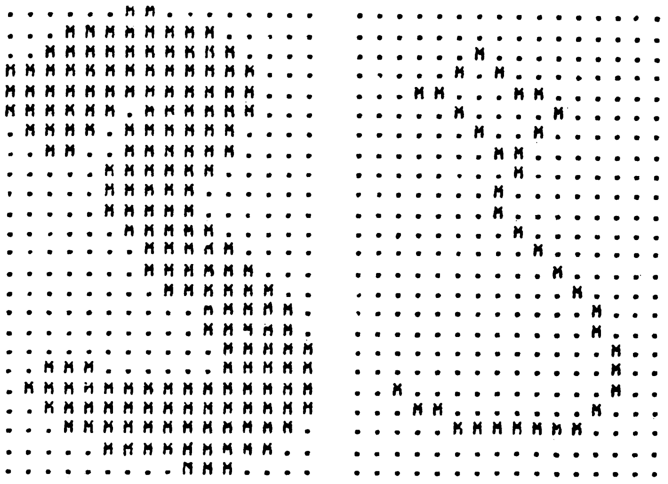

Fig. 1. Effect of noise hole.

Introduction

Character thinning is used as a preprocessing stage in OCR to simplify the subsequent recognition problem. However, specific defects in the data can cause thinning algorithms to destroy information and lead to misrecognition. The number and importance of such defects are particular to each problem and its application, and their effect on a thinning algorithm can only be estimated by experiment. Three such common data defects are 1) the presence of several sorts of small holes which lead to spurious loops in the skeleton, 2) single element edge irregularities which lead to spurious tails, and 3) acute angles between limbs which lead to distortions and results also having spurious tails.

Most thinning algorithms discussed in the literature [1]–[8] rely on the steady erosion of character boundaries while maintaining the connectivity of the shape. All are sensitive to one or more of the above data defects. Connectivity rules tend to overemphasize the importance of small holes and produce topologically incorrect skeletons with spurious loops (Fig. 1). Tenuously connected irregularities residing on limb edges (Fig. 2) gain exaggerated importance by being labeled as "limb ends" very early in the thinning process. This leads to the generation of spurious tails.

It is a characteristic of boundary eroding algorithms that more material is removed from one side of a character limb than the other if the perimeter is longer. This means that limbs will tend to be eroded much more rapidly from the outside of an acute angle junction than from the inside. This leads to the production of an extra tail (Fig. 3) or to a "necking" effect at cross overs (Fig. 4).

In this correspondence the problems are considerably reduced by the introduction of some preprocessing stages before a standard thinning algorithm is applied. These preprocessing heuristics are specifically aimed at reducing the failure rate due to the above defects in the data.

Preprocessing

A. Hole Removal

In this work characters are restricted to black and white elements arranged in a 16 × 24 matrix. Preprocessing begins with the detection of six types of hole by raster scanning the character with six patterns of bits. Three of these (H_i) will fit all holes containing one or two elements (Fig. 5) and the remaining three (I_i) will fit the same size of hole (Fig. 6) but only if the hole is embedded at least two elements deep in a limb.

Embedded holes identified by the I_i are likely to represent a real loop in the character and are retained. Other holes identified

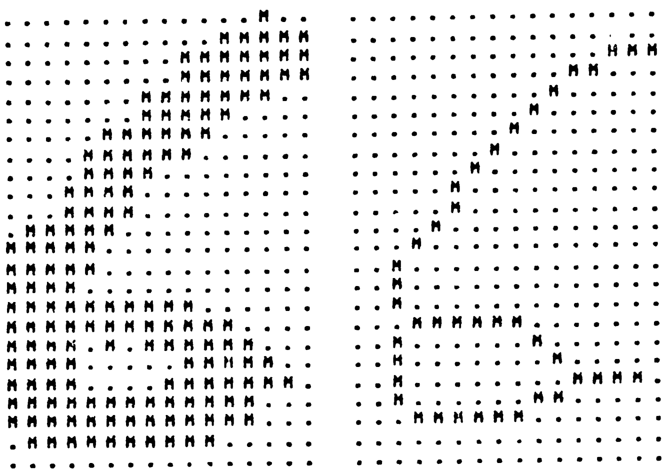
Fig. 2. Effect of spurious projection.

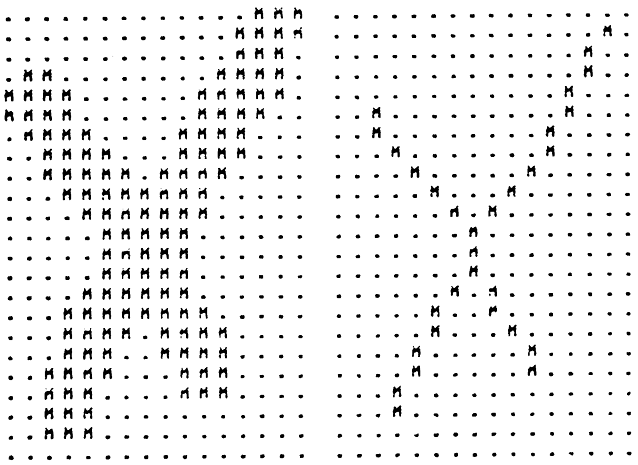
Fig. 4. "Necking" effect.

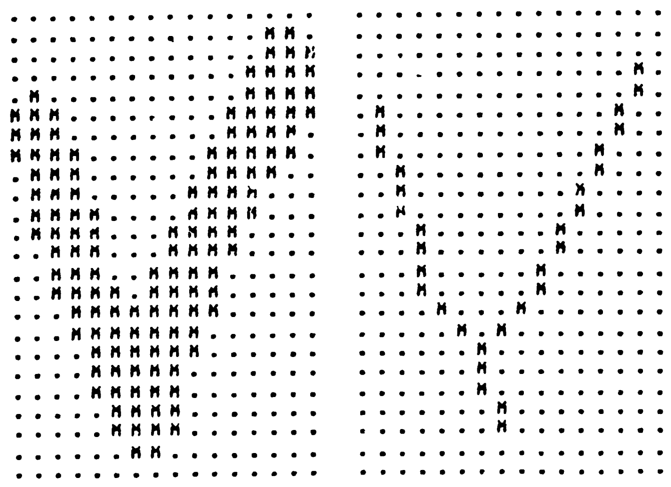
Fig. 3. Tail generation at acute limb angle.

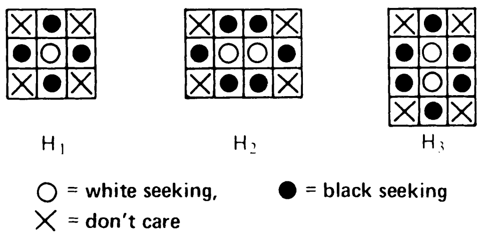

Fig. 5. Hole detectors.

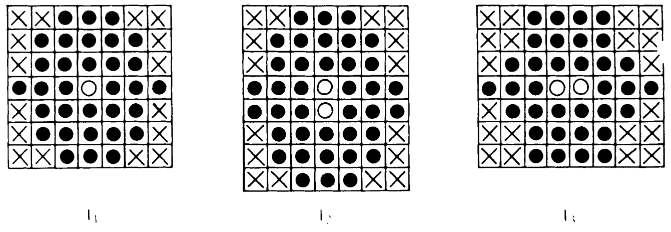

Fig. 6. Isolated hole detectors.

by the H_i are located fairly close to a limb edge and can be attributed to the effects of noise. These holes are removed by 1) merging the hole with adjacent white areas by the removal of black elements, or 2) filling it with additional black elements.

B. Smoothing

The second preprocessing stage is the removal of all black elements having less than three black neighbors and having connectivity 1 (Appendix). This has the effect of removing single element projections and all isolated spots having one or two elements.

C. Acute Angle Emphasis

The final preprocessing stage involves the detection of upward and downward acute angles between limbs by again scanning the character with patterns of bits. Five of these (D_i) will fit the sharpest forms of downward pointing acute angles (Fig. 7) and a reflected set (U_i) will fit upward pointing acute angles. After a fit is found the central black element is deleted and the process is repeated twice more. The second and third iterations are only carried out if the preceding iteration effected a deletion. Only D_i, U_i, $i = 1,2,3$ are scanned during the second iteration and only D_1, U_1 on the third, since fits are not possible with the other patterns. This heuristic compensates for much of the uneven thinning by increasing the length of the boundary on the inside of angled intersections. The result of such preprocessing on the character shown in Fig. 3 is shown in Fig. 8.

D. Summary

The preprocessing algorithms described in this section can be described more formally as follows:

Step 1	Identify character hole which matches an H_i.
Step 2	Given a match in Step 1, if one of the picture elements at the top left, top right, bottom right, or bottom left corner of H_i is white then take first such and mark its two black neighbors for deletion; else if none of I_m, $m = 1,2,3$ fits the hole then mark hole for filling.
Step 3	Return to Step 1 until all such holes have been identified.
Step 4	Delete all marked elements.
Step 5	Fill all marked holes.
Step 6	Remove elements having two, one, or no black neighbors and having connectivity 1 or 0.
Step 7	Identify upward or downward acute angle fitting U_i or D_i ($i = 1,2,\cdots,5$).
Step 8	For each fit the central element is marked for deletion.
Step 9	Loop to Step 7 until all such fits have been detected.
Step 10	Delete all marked elements.
Step 11	If elements have been deleted in Step 10 repeat Steps 7–10 but using only U_i, D_i ($i = 1,2,3$); otherwise Exit.

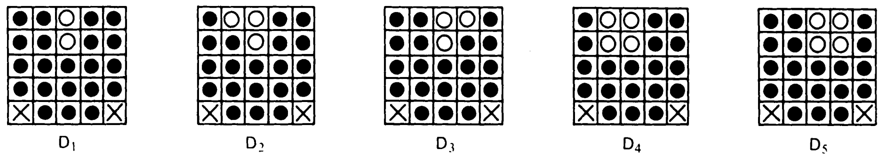

Fig. 7. Downward angle detectors.

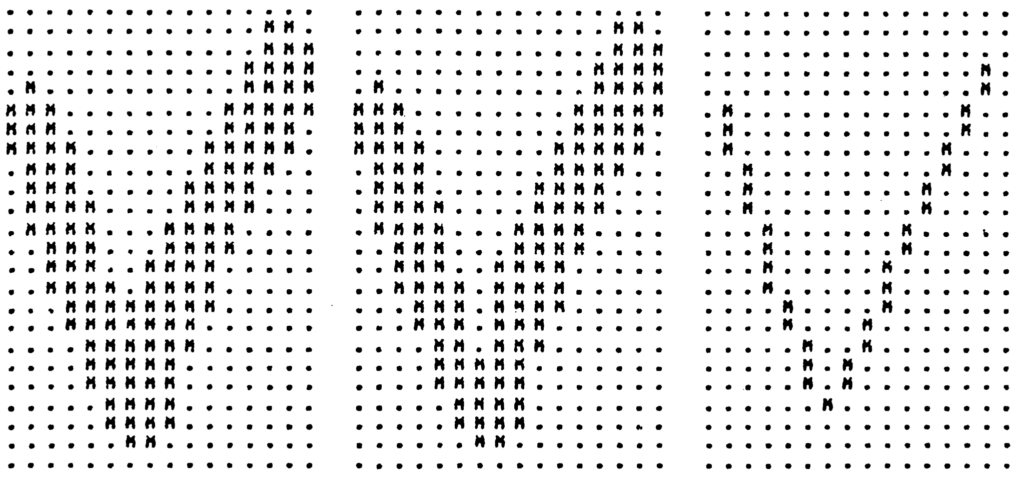

Fig. 8. Application of acute angle emphasis heuristic.

Step 12 If elements have been deleted in Step 11 repeat Steps 7–10 using U_1 and D_1 only.
Step 13 Exit.

Thinning Algorithm

After preprocessing, a variation of a standard thinning algorithm (1) is applied. Four matrices (M_i) in Fig. 9 are scanned over the character, and wherever a matrix fits the central black element is marked for deletion. Elements are not so marked if they are limb endpoints or if the connectivity measure for that point (Appendix) is greater than one. An endpoint is defined as a black element which is 8-connected to only one other black element. Elements already marked are considered to be white for the purpose of subsequent end point or connectivity calculations. When all four matrices have been scanned in this way, all marked elements are deleted and the process repeated until no more erosion can take place.

The results of such a thinning process are sensitive both to the order of application of the M_i and also the direction of scan. An ordering which minimizes spurious tail production is given in Table I. For example M_3, the south edge eroding matrix, is scanned from right to left moving upwards across the character.

This algorithm can be summarized as follows.

Step 1 Scan matrix M_1 across character according to Table I and identify next fit position.
Step 2 If the central element at a fit is not an endpoint and has connectivity value one, then mark it for deletion.
Step 3 Repeat Steps 1 and 2 for all fit positions.
Step 4 Repeat Steps 1–3 for each of M_2, M_3, M_4.
Step 5 Delete all marked elements.
Step 6 If one or more elements are deleted in Step 4 then return to Step 1.
Step 7 Exit.

Results and Discussion

Results on an unseen test set of 680 unconstrained handprinted alphanumeric characters indicated that less than 1 percent of thinning failures could be attributed to the algorithm. A significant reduction in confusions between the class pairs (K, X), (H, X), (V, Y), and (H, M) were recorded after thinning using the preprocessing heuristics.

Improvements beyond the 1 percent failure rate would follow the identification of more complex configurations of character elements which lead to the remaining commonest types of thinning failures.

Hardware and processing requirements go up as the size and number of matching matrices are increased. However the basic binary matching process is common to the thinning, the hole detection, and the acute angle detection. This means that existing image processing hardware designs working at video rates [9] are able to carry out the required tasks quite economically. Smoothing, end point detection, and the calculation of connectivity values are carried out rapidly using a "table-lookup" scheme for all possible 3 × 3 configurations.

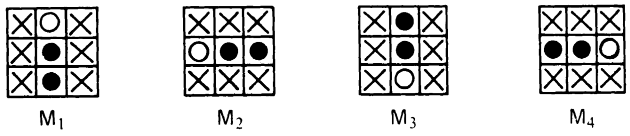

Fig. 9. Thinning matrices.

TABLE I
MATRIX ORDERING AND SCAN DIRECTIONS

ORDER OF APPLICATION	DIRECTION OF SINGLE SCAN LINE	DIRECTION OF SUCCESSIVE SCAN LINES
M_1	Left to Right	Downwards
M_2	Upwards	Left to Right
M_3	Right to Left	Upwards
M_4	Downwards	Right to Left

Appendix

The connectivity measure N used is one defined by Yokoi *et al.* [2] as follows:

$$N(\gamma_0) = \sum_{K \in S} (\bar{\gamma}_K - \bar{\gamma}_K \cdot \bar{\gamma}_{K+1} \cdot \bar{\gamma}_{K+2})$$

where

$$S = \{1,3,5,7\}$$
$$\bar{\gamma}_K = 1 - \gamma_K$$
$$\gamma_j = \gamma_{j-8} \quad \text{for } j > 8$$

and the eight neighboring pixels of γ_0 are labeled:

$$\begin{array}{lll} \gamma_4\gamma_3\gamma_2 & & \\ \gamma_5\gamma_0\gamma_1 & \text{with } \gamma_i = 0 & \text{if white} \\ \gamma_6\gamma_7\gamma_8 & \phantom{\text{with } \gamma_i} = 1 & \text{if black} \end{array}$$

Acknowledgment

The authors wish to thank their colleagues for many useful discussions relevant to this work, and acknowledgement is made to the Director of Research of British Telecom for permission to make use of the information contained in this correspondence.

References

[1] E. R. Davies and A. P. N. Plummer, "Thinning algorithms and their role in image processing," presented at the British Pattern Recognition Association Conference on Pattern Recognition, Jan. 9-11, 1980.
[2] S. Yokoi, J. Toriwaki, and T. Fukumura, "Topological properties in digitized binary pictures," *Systems Computers Controls*, vol. 4, pp. 32-39, 1973.
[3] R. Stefanelli and A. Rosenfeld, "Some parallel thinning algorithms for digital pictures," *J. Ass. Comput. Mach.*, vol. 18, no. 2, pp. 255-264, Apr. 1971.
[4] E. E. Triendl, "Skeletonization of noisy handdrawn symbols using parallel operations," *Pattern Recognition*, vol. 2, pp. 215-226, 1970.
[5] C. Arcelli, L. P. Cordella, and S. Levialdi, "More about a thinning algorithm," *Electron. Lett.*, vol. 16, no. 2, pp. 51-53, Jan. 17, 1980.
[6] C. J. Hilditch, "Linear skeletons from square cupboards," in *Machine Intelligence 4*, B. Meltzer and D. Michie, Eds. Edinburgh: Edinburgh Univ., 1969, pp. 403-422.
[7] M. Beun, "A flexible method for automatic reading of handwritten numerals II," *Philips Tech. Rev.*, vol. 33, no. 5, pp. 130-137, 1973.
[8] C. Arcelli and G. S. Dibaja, "A thinning algorithm based on prominence detection," *Pattern Recognition*, vol. 13, pp. 225-235, 1981.
[9] J. R. Ullmann, "Video-rate digital image analysis equipment," *Pattern Recognition*, vol. 14, pp. 305-318, 1981.

Pyramid Linking is a Special Case of ISODATA

SIMON KASIF AND AZRIEL ROSENFELD, FELLOW, IEEE

***Abstract*—It is shown that the "pyramid linking" method of image segmentation can be regarded as a special case of the ISODATA clustering algorithm and hence is guaranteed to converge.**

Manuscript received November 20, 1981; revised July 19, 1982. This work was supported by the National Science Foundation under Grant MCS-79-23422.

The authors are with the Computer Vision Laboratory, Computer Science Center, University of Maryland, College Park, MD 20742.

An important class of image segmentation techniques involve segmenting the image into homogeneous regions by a repeated region merging process; see [1, ch. 5] for a general treatment of such methods. The purpose of this correspondence is to show that one such method, "pyramid linking," which was introduced in [2], can be regarded as a special case of the ISODATA clustering algorithm (e.g., [3]), and is therefore guaranteed to converge.[1]

The pyramid linking process can be briefly summarized as follows [2]. Given a $2^n \times 2^n$ image, we build an exponentially tapering "pyramid" of images of sizes $2^{n-1} \times 2^{n-1}, 2^{n-2} \times 2^{n-2}, \cdots$ by repeatedly taking averages of 4×4 blocks that overlap 50 percent in the horizontal and vertical directions. In this pyramid each pixel at any level above the base has 16 "sons" on the level below that contribute to its average, and each pixel at any level below the apex has four "fathers" on the level above to whose average it contributes. We next "link" each pixel to the father whose value is closest to its own. We then recompute the averages, using only those sons that are linked to a given pixel in computing its new value. We now redefine the links if necessary, recompute the averages again, and repeat the process until there is no further change.

This iterative linking process, at each level of the pyramid, is a special case of the following general procedure, which is a simplified version of ISODATA. Let $S = \{z_1, \cdots, z_n\}$ be a set of real numbers; let $S_1^{(0)}, \cdots, S_m^{(0)}$ be an arbitrary initial partition of S; and let $\mu_i^{(0)}$ be the mean of the z's in $S_i^{(0)}$, $1 \leqslant i \leqslant m$. We modify the partition by shifting z's from one subset to another according to some rule, but subject to the restriction that z can shift from $S_i^{(0)}$ to $S_j^{(0)}$ only if it is closer to $\mu_j^{(0)}$ than to $\mu_i^{(0)}$. This yields a new partition $S_1^{(1)}, \cdots, S_m^{(1)}$ with new means $\mu_1^{(1)}, \cdots, \mu_m^{(1)}$, where some of the subsets may be empty (and have unidentified means). The process can then be repeated. We now show that the resulting sequence of partitions must stabilize after a finite number of steps. At this point every z is at least as close to the mean of its subset as it is to the mean of any other subset, so that no further change is possible.

To prove this convergence result, we introduce a cost function F, namely the sum of the squared differences between the z's and the means of their subsets:

$$F \equiv \sum_{i=1}^{m} \sum_{z \in S_i} (z - \mu_i)^2.$$

At a given iteration let T_{ij} be the set of z's that shift from S_i to S_j, $1 \leqslant i \neq j \leqslant m$. Then just before the kth iteration we have

$$F^{(k-1)} = \sum_{i=1}^{m} \left[\sum_{z \in S_i - \bigcup_{\substack{j=1 \\ j \neq i}}^{m} T_{ij}} (z - \mu_i)^2 + \sum_{\substack{j=1 \\ j \neq i}}^{m} \sum_{z \in T_{ij}} (z - \mu_i)^2 \right] \tag{1}$$

where for compactness we have omitted the superscript $(k-1)$ from the S's and T's, and where a sum over an empty set is defined to be zero. Now for all $z \in T_{ij}$ we have $(z - \mu_j)^2 < (z - \mu_i)^2$, since such a z must be closer to the mean of S_j than to the mean of S_i. Hence (1) implies

$$F^{(k-1)} > \sum_{i=1}^{m} \left[\sum_{z \in S_i - \bigcup_{\substack{j=1 \\ j \neq i}}^{m} T_{ij}} (z - \mu_i)^2 + \sum_{\substack{j=1 \\ j \neq i}}^{m} \sum_{z \in T_{ij}} (z - \mu_j)^2 \right]. \tag{2}$$

[1] We make no claims about the global optimality of the limit to which the algorithm converges, but only that it reaches a stable state after a finite number of steps. On the convergence of algorithms related to ISODATA see [4].