Application of a Growing Self-Organizing Map to Thinning of Binary Characters with Noise

Igor Farkaš & Lucius Chudý
Institute of Measurement Science
Slovak Academy of Sciences
Dúbravská cesta 9, 842 19 Bratislava, Slovakia
e-mail:{farkas,chudy}@neuro.savba.sk

Abstract

We present an improved version of our thinning algorithm, based on growing SOM-like approach, specifically Dynamic Cell Structures (DCS). The algorithm creates an output representation of the pattern in the form of undirected graph possessing the desirable features of a skeleton. The line-like graph topology, would otherwise be violated by original DCS, is forced by modifications in unit connections' update and winners' search. The algorithm was tested on binary characters and is shown to be robust with respect to boundary noise.

1 Introduction

Thinning is a fundamental early processing step in pattern analysis. Its purpose is to transform the original pattern onto a line-like structure, called the *skeleton*, while preserving the topological properties of the original pattern. The advantages of skeletons are the reduction in the required memory space for storing essential structural information of the pattern, the simplification of data structures required in processing the patterns, and the reduction in the required processing time. Thinning algorithms have thus been very useful in various applications, e.g. character recognition, fingerprint recognition, biomedical dignosis etc. (see [8] for references therein).

Thinning algorithms are commonly classified according to two viewpoints. Firstly, they can either be *iterative* or *non-iterative*. In iterative techniques, the final skeleton results from repetitive operations performed on the original pattern. Non-iterative techniques yield the skeleton after a single pass over the pattern. Secondly, the thinning algorithms are divided into *direct* or *indirect*. Direct techniques change the original pattern, e.g. by iterative deletion of individual pixels by some rule, until the skeleton remains. On the contrary, indirect techniques do not alter the original pattern, but use their own resources for representing a skeleton.

Most of the techniques developed so far belong to direct iterative ones. They are typically based on various decision rules employed for deletion/retention of examined individual pixel and are further subdivided into sequential and parallel algorithms (see [8] for survey).

As a matter of fact, it has been shown that indirect techniques can perform better than some widely recommended direct techniques. The example may be the CBSA algorithm [9], which is based on fuzzy clustering of the pattern. This method produces cluster centers, which are then connected using the adjacency matrix to produce a skeleton. CBSA was shown to be superior mainly in being insensitive to the boundary noise.

The only neural-network based method we found in literature – the self-organizing graph (SOG) [1], also performs clustering of the original pattern. It is similar to a one-dimensional self-organizing map (SOM) [7], but does not exploit its topology. SOG uses time-varying neighbourhood, but does not produce the skeleton itself; the resulting centers need further to be linked (which is not a trivial task). In addition, both CBSA and SOG presume a preset, constant number of units, which requires some a priori knowledge about the pattern. [1]

In this paper, we present an improved version of our thinning algorithm, which can be viewed as a growing SOM. In recent years, SOM-like algorithms employing time-varying topology have been designed as a powerful method for topological approximation of various data structures [5], [3], [2], [10]. All of these algorithms yield output representation in the form of undirected graph. Our algorithm has been derived from Dynamic Cell Structures (DCS) [3], which are not constrained to fixed graph dimension (as e.g., in the case GCS [5]). Therefore, DCS appear to be suitable to thinning as well. Unlike CBSA or SOG, DCS do not use a fixed number of units, but these are inserted according to the need. However, the direct application of DCS to thinning of characters is not feasible, due to their varying shape and thickness, which would often cause the violation of line-like graph topology (i.e., each unit, except for "junctions", has two connections). We tested our algorithm on noisy patterns and demonstrate its robustness with respect to boundary noise.

2 Thinning algorithm

The DCS network [3] can be viewed as a graph $\mathcal{G} = (W, C)$, where $W = \{\mathbf{w}_1, \mathbf{w}_2, ..., \mathbf{w}_n\}$ is the set of n knots (units) given by their coordinates, $\mathbf{w}_i \in \mathcal{R}^2$, and $C[n \times n]$ is the (symmetric) connection matrix defining edges in $\mathcal{G}$. $C_{ij} = C_{ji} \in \langle 0, 1\rangle$ defines the strength of the connection between units i and j in $\mathcal{G}$ (if zero, the connection does not exist).

Regarding variability of characters, the main problem with DCS for thinning of patterns is to prevent the final graph from violating *line-like* topology. The most frequent cause of violation is the presence of m-polygons (the simplest, 3-polygon i-j-k exists, if $C_{ij}, C_{jk}, C_{ik} > 0$). [2] Polygons naturally emerge during the course of algorithm and would remain due to its attempt to approximate the data distribution (i.e., they mainly appear in thicker arcs of a character or at junctions). Therefore, it is necessary to impose some restrictions upon making connections between units so that all arcs in the character, independently of their thickness, were approximated by connected lines only, and more than two neighbors for a knot were allowed only at junctions in the character.

Our approach – skeletonizing DCS (S-DCS), includes 3 modification steps and has the following form:

```
• Initialization: set $\theta, \lambda, \epsilon_B, \epsilon_N$; compute $\alpha$, determine the stopping criterion;
    - choose randomly $\mathbf{v}_1, \mathbf{v}_2$ and set $\mathbf{w}_1 = \mathbf{v}_1, \mathbf{w}_2 = \mathbf{v}_2, C_{12} = C_{21} = 1$.
• Phase 1: /*DCS*/
          do {
             for ($\lambda$ times) {                          % repeat inner cycle (IC)
                getNextExample($\mathbf{v}$);
                calculateTwoClosest($\mathbf{v}, \mathbf{w}_B, \mathbf{w}_N$);
                updateConnectionStrengths($\mathbf{w}_B, \mathbf{w}_N, C, \alpha, \theta$);
                kohonen($\mathbf{w}_B, \mathbf{v}, \epsilon_B, \epsilon_N$);
                updateWinnerResource($\beta_B$);             % i.e. $\Delta\beta_B += ||\mathbf{w}_B - \mathbf{v}||^2$)
             }
             addNewNeuron();      resetResources();
          } while (!stoppingCriterion());
• Phase 2: /*connections' update*/
          - repeat IC ($2\lambda$ times) with breadthSearch($\mathbf{v}, \mathbf{w}_B, \mathbf{w}_N$) and modified $C_{ij}$ update (Eq. 1)
• Phase 3: /*fine tuning*/
          - repeat IC (without $C_{ij}$ update) with learning rate decay
```

[1] Though, it must be noted that this also has an advantage: e.g., in using neural networks for character recognition, the data input length should be fixed. This is satisfied if skeletons of all characters have the same number of centers.

[2] Of course, it is necessary to differentiate between undesirable m-polygons and "natural" m-polygons, which occur in case of "looped" characters (e.g. α, ϕ). Fortunately, the latter typically have significantly higher m.

The purpose of all three modification steps is briefly explained below.

(1) Adding a new unit is associated with defining the stopping criterion. Rather than quantization error E used in original DCS, we employ average Euclidean distance between two connected units, $D = E\{||\mathbf{w}_i - \mathbf{w}_j||\}$, for $(i, j) \mid C_{ij} > 0$. The reason is that we want a line-like approximation of all arcs in the character (which can have varying thickness). Consequently, we thus allow local E's to vary from unit to unit, but tend to have equidistant placement of connected units. Therefore, the new unit is inserted between the two most distant connected units. Placing it according to the ratio of their resource values β_i, β_j, as in original DCS, appeared to be superior to mid-placement, so we adopted this strategy.

(2) The purpose of bradth-search (BS) procedure, known from the graph theory, is to get rid of undesirable cross-border connections, [3] which otherwise may remain, if we look for winners in terms of Euclidean distance. BS can be interpreted as spreading a wave (in all directions) from the current input over the pattern (black pixels in a 2D grid), until two closest units (B, N) are found. For the purpose of thinning of noisy patterns, BS must be allowed to visit white pixels as well (however, only those having direct black neighbors in a grid), in order not to by-pass winner(s) currently positioned in "white" position(s). According to simulations, BS is more robust, though not significantly faster than the former calculation of constrained distance [4].

(3) Modification in updating C matrix in phase 2 is aimed at removal redundant within-border connections (i.e. those making 3-polygons in the graph). [4] Hence, having found B and N, the connection weights of C are updated as follows:

$$C_{kl} \leftarrow \begin{cases} 1 & : \; k = B \wedge l = N \wedge (\,! \exists m : C_{km} > 0 \wedge C_{lm} > 0) \\ C_{kl} & : \; k = B \wedge l = N \wedge (\exists m : C_{km} > 0 \wedge C_{lm} > 0) \\ 0 & : \; C_{kl} < \theta \\ \alpha . C_{kl} & : \; \text{otherwise} \end{cases} \tag{1}$$

From Eq. 1 it is obvious that setting the winner couple connection $C_{BN} = 1$ is prevented in the case when there exists a 3-polygon B-N-m. As a consequence, in such a polygon, the connection with the lowest strength will most probably die off. This strategy, works quite well, but the died connection may not always be the one, that "we would like" to be removed. As a consequence, the resulted skeleton can be perturbed in shape, though being topologically correct (see e.g. junctions at skeletons of χ, ψ, ω in Fig. 2).

3 Results

We tested S-DCS on printed Greek characters (MS Windows font). The feasibility of our algorithm (specifically, its former version) in the case of noise-less patterns was demonstrated in [4]. Now we aimed at experimenting with noisy patterns. By noise we understand perturbations in the outline of a pattern, which usually cause deformation or offshoots (spurious tails) in the final skeleton (mainly in direct techniques based on contour following).

The noise level was quantified by the percentage of inverted pixels of all outline pixels (reaching the maximum at 50%, see Fig. 1). The signal-to-boundary noise ratio is defined as [6]

$$SBNR = \frac{\text{Area}[\partial \mathcal{I}]}{\text{Area}[\mathcal{I}'/\mathcal{I}] + \text{Area}[\mathcal{I}/\mathcal{I}']}$$

where $\mathcal{I}$ and $\mathcal{I}'$ are respectively noise-free and noisy images, $\partial \mathcal{I}$ is the boundary of $\mathcal{I}$, and "/" denotes the set difference. Thus, the error created by boundary noise at a particular *SBNR* can

[3] A cross-border connection in the graph is the one connecting two directly unconnected parts of the image.

[4] A within-border connection is the one connecting two directly connected parts of the image.

be measured by the normalized quantity

$$m_e(SBNR) = \min\left\{1, \frac{\text{Area}[\mathcal{S}'/\mathcal{S}] + \text{Area}[\mathcal{S}/\mathcal{S}']}{2 \times \text{Area}[\mathcal{S}]}\right\},$$

where $\mathcal{S}$ and $\mathcal{S}'$ are the resulting skeletons of $\mathcal{I}$ and $\mathcal{I}'$, respectively. A highly noise-sensitive algorithm will yield an m_e close to 1.

We performed multiple runs on Greek characters with various levels of noise. The parameters of S-DCS were set as folows: connection deletion threshold $\theta = 0.001$, forgetting constant $\alpha = \sqrt[e]{|\mathcal{I}|}$, where $|\mathcal{I}|$ denotes the size of the training set (number of black pixels forming the character), $\lambda = |\mathcal{I}|/2$, learning constants $\epsilon_B = 0.2, \epsilon_N = 0.01$. The stopping criterion had to be set manually, $D = 6$ (for noise-free characters, however, it can be derived from average arc thickness estimate [4]). Producing a skeleton took below 10000 iterations (depending on character), which corresponded to a few seconds of CPU time on 486-based PC. Average results are shown in Table 1. Average m_e remains relatively well below 1, and does not significantly grow with decreasing *SNBR*. We interpret this fact as insensitiveness of S-DCS to the noise level. This may not be surprising, as the knot placement is based on input statistics, and not on precise positions of individual pixels. A few final skeletons are displayed in Fig. 2.

Noise level (in %)	*SBNR*	$m_e(SBNR)$
10	5.47 ± 0.22	0.37 ± 0.10
20	2.68 ± 0.12	0.40 ± 0.10
30	1.78 ± 0.06	0.47 ± 0.09
40	1.33 ± 0.04	0.46 ± 0.06
50	1.07 ± 0.02	0.50 ± 0.10

Table 1: Experimental results of approximating skeletons of noisy patterns with S-DCS algorithm. We ran 10 simulations for each pattern and each noise level. Corresponding average $E = 2.0 \pm 0.1$, number of inserted units per character varied between 15 and 22.

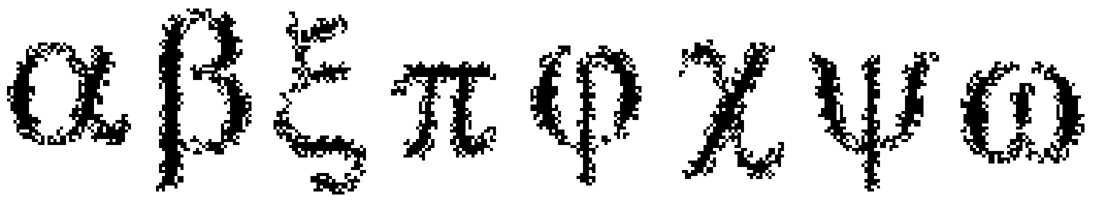

Figure 1: Examples of patterns with 50% boundary noise used in experiments.

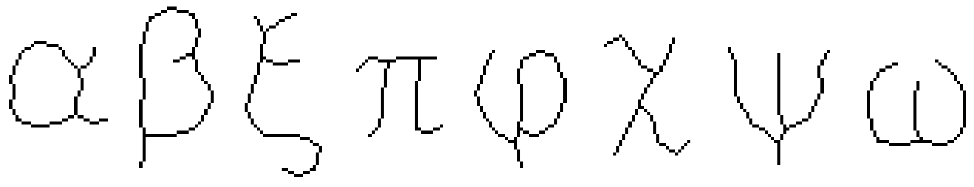

Figure 2: Examples of obtained skeletons with S-DCS, trained on patterns from Fig. 1. Graph-to-skeleton transformation procedure was realized by interpolating and integer-rounding to fit the values into the grid.

4 Discussion

Though not compared to other thinning algorithms (which would require the use of the same training set), we can make a few statements concerning the properties of S-DCS. According to [11], a good thinning algorithm should satisfy at least these four objective criteria – connectivity, thinness, sensitivity to boundary noise and CPU running time. Regarding these, we can say that:

(a) S-DCS produce almost surely connected skeletons. By almost surely we mean that there occur cases when a connection happens to be either missing or redundant. The reason may be e.g. that the correct winner is not found with BS procedure, because it currently lies away from the data area, or "unwanted" connection (of the three in a polygon) dies off due to Eq. 1 applied in phase 2). (b) The produced skeletons have intrinsically unitary thinness as implies the nature of the algorithm, together with the graph-to-skeleton transformation procedure. (c) S-DCS is quite robust with respect to boundary noise, up to its maximum level, because error m_e grows slightly. (d) In comparison to direct techniques [11], the speed of S-DCS appears to be a weak point; hence, it is suitable for off-line thinning tasks only.

It may not be surprising that S-DCS share the same virtues and vice as above mentioned CBSA. Their advantage is noise robustness, as can be expected from their stochastic nature. On the other hand, they are relatively slow in comparison to deterministic direct, especially one-pass techniques.

Acknowledgement

This work was supported by the grants from the Slovak Grant Agency for Science (No. 2/2040/95 and 95/5305/468).

References

[1] P. Ahmed. A neural network based dedicated thinning method. *Pattern Recognition*, 16:585–590, 1995.

[2] J. Blackmore and R. Miikkulainen. Incremental grid growing: encoding high-dimensional structure into a two-dimensional feature map. In *Proc. of ICNN'93, San Francisco, CA*, pages I–450, 1993.

[3] J. Bruske and G. Sommer. Dynamic cell structure learns perfectly topology preserving map. *Neural Computation*, 7:845–865, 1995.

[4] I. Farkaš and L. Chudý. Modified dynamic cell structures as a thinning algorithm. In P. Sinčák, editor, *Proc. of 1-st Slovak Neural Network Symposium, Herľany*, pages 71–80, November 1996.

[5] B. Fritzke. Growing cell structures - a self-organizing network for unsupervised and supervised learnig. *Neural Networks*, 7(9):1441–1460, 1994.

[6] B.K. Jang and R.T. Chin. One-pass parallel thinning: analysis, properties, and quantitative evaluation. *IEEE Trans. Patt. Anal. Machine Intell.*, 14(11):1129–1140, 1992.

[7] T. Kohonen. *Self-Organizing Maps*. Springer Verlag, 1995.

[8] L. Lam, S. Lee, and C.Y. Suen. Thinning methodologies – a comprehensive survey. *IEEE Trans. Patt. Anal. Machine Intell.*, 14(9):869–885, 1992.

[9] S. Mahmood, I.S. Abuhaiba, and R.G. Green. Skeletonization of arabic characters using clustering based skeletonization algorithm (CBSA). *Pattern Recognition*, 24(5):453–464, 1991.

[10] C. Szepesvári and A. Lőrincz. Approximate geometry representations and sensory fusion. *Neurocomputing*, 12(2-3):267–287, 1996.

[11] R.W. Zhou, C. Quek, and G.S. Ng. A novel single-pass thinning algorithm and an effective set of performance criteria. *Pattern Recognition Letters*, 16:1267–1275, 1995.