STUDIA UNIV. "BABEŞ–BOLYAI", INFORMATICA, Volume I, Number 2, October 1996

# ANALYSING THE NOISE SENSITIVITY OF SKELETONIZATION ALGORITHMS

ATTILA FAZEKAS AND ANDRÁS HAJDU

**Abstract.** Many skeletonization algorithms have been analysed from several points of view in the past ten years to compare the results they produce. Our attempt here is to add a new comparative analysis to this research area which investigates the noise sensitivity of these algorithms. We examined the performances of five algorithms (they are based on different thinning models) on a huge picture set, and sorted the algorithms according to the results they produced. This analysis can be useful for those who would like to apply the most efficient algorithm for a special kind of noisy image.

## 1. INTRODUCTION

The necessity of designing skeletonization algorithms dates back to the early years of computer technology, to the 1950s. It was realised that in some applications (the first problem was the character recognition), it is enough to take only a reduced amount of information into account instead of the whole image, which is usually a line-drawing. The basic idea was to "peel" the original picture by iteratively removing certain contour points. This procedure is the so–called skeletonization, which creates a line-like shape (the skeleton), so the further analysis becomes easier to execute. The skeleton has the following advantages: there is less information to process, and the shape analysis can be made more easily. Since then many new challenges have occurred from several parts of life, and now skeletonization is applied in a very wide range, e.g., in the analysis of blood cells or chromosome shapes in medical science, or in identifying signatures and fingerprints. Many papers have been published to take a survey of the skeletonization processes without going into details, see [1,4,7]. These articles make the reader familiar with the most important concepts of skeletonization.

A lot of algorithms have been developed and implemented during the past ten years to find the skeletons of different images. It is very difficult to measure the "goodness" of such a method quantitatively. The analytical comparison of the methods is very sophisticated, since they are based on different models, cf. [8,

1991 *Mathematics Subject Classification.* 68U10 Image Processing.

p. 263]. That is the reason why the skeletonizations are compared according to the results they produce in the practice. There are papers about the technical parameters of these algorithms (like computation speed, memory requirement, etc.), and there are observations based on the result skeletons the algorithms produced. A possible way to classify the algorithms is to examine if the result skeletons meet the following (natural) conditions:

- The skeleton should accurately reflect the shape of the original image
- The topological properties (homotopy) of the object and the background should be preserved
- The thickness of the skeleton should be one pixel
- The skeletonization should preserve symmetry
- The skeletonization process should be immune to noise

A more detailed description about the requested properties of skeletons can be found in [8, p. 239].

Usually a reference skeleton is composed, and then the distance of the result and the reference skeleton is calculated by using a suitable distance function. The reference skeleton can be obtained for example, by asking humans to select the skeleton of the object and then averaging the selected skeletons. An interesting way of selecting the reference skeletons can be found in [8, p. 283], where a lot of humans were involved in the creation of the reference skeletons. The most frequently used distance functions are introduced in Section 2. Other functions which are useful to investigate the similarity of the result and the reference skeletons can be found in [8, p. 283]. In the case of using several distance functions, strong correlation can be measured between the distances. Few methods divide the skeletons into fragments, and use polygons to match the result skeletons to the reference instead of using these distance functions (see [8, p. 307]).

The aim of this paper is to analyse the "goodness" of skeletonizations from a special point of view. We found that the skeletonizations were not examined statistically (with a large number of experiments) to test how the noise corruption affects the extraction of the skeleton, and how the skeletonization processes can cope with noisy images. Unfortunately the input pictures are rarely ideal, but are corrupted with some kind of noise. For example, the contours in the image of a printed circuit board are often corrupted by a contour noise, which makes the contours disconnected, thicker, etc. Our purpose was to decide which is the most efficient algorithm for noisy images, among the investigated ones.

## 2. BASIC CONCEPTS AND NOTATIONS

In the following we need some concepts, so we give the most important definitions that are used.

A *binary picture* can be represented by a 2D-array of points which have the value either 0 or 1 (either white or black). The *background* is the set of the white points and the *object* is the set of the black points.

A *contour (edge) point* is an object point with at least one background neighbour. The *contour (edge)* consists of all the contour points of the object.

A *thinning algorithm* operates on the contours of the objects and eliminates contour points chosen by some kind of neighboring considerations. The common method is to remove iteratively all the contour points of the object except those points which belong to the skeleton. This type of thinning is the contour sequential algorithm, which is based on contour tracing.

Another possibility is to use a template of a given size to scan the picture, and remove the points that belong to an edge but not to the medial axis of the object. The procedure stops when no more changes are made.

To test the similarity (i.e., to calculate the distance) of the result and the reference skeletons one can use the Hamming-distance, which has the following form:

$$X(T,R) = \sum_{i=1}^{m}\sum_{j=1}^{n}(T_{ij}\,\mathrm{XOR}\,R_{ij}), \qquad (1)$$

where pictures $T$ and $R$ of size $m \times n$, contain the test and the reference skeleton, $i, j$ are pixel coordinates ($ij$ means the position of the pixel in the $i$th row and $j$th column) and XOR means the logic operator exclusive OR. This function computes the value of the XOR operator according to the result and the reference skeleton.

Another useful function is the distance

$$D_1(T,R) = \frac{1}{N_T}\sum_{i=1}^{m}\sum_{j=1}^{n}\sqrt{d(T_{ij},R)} + \frac{1}{N_R}\sum_{i=1}^{m}\sum_{j=1}^{n}\sqrt{d(T,R_{ij})},$$

with

$$d(T_{ij},R) = \begin{cases} 0 & \text{if } T_{ij} = 0; \\ \min_{\{u,v\,|\,R_{uv}\neq 0\}}\{(u-i)^2+(v-j)^2\} & \text{if } T_{ij} \neq 0 \end{cases}$$

where $1 \leq u \leq m$, $1 \leq v \leq n$, and $N_T$ and $N_R$ are the numbers of the object points in the test and the reference pictures, respectively. Thus this function computes the mean Euclidean distance of the pixels of the test and the reference skeletons.

We corrupt an picture with *additive noise* if we change background points to object points (the number of object points increases), and we generate *subtractive noise* when object points become background points (the number of the object points decreases).

The noise is *global* if the whole picture is involved, and we talk about *contour noise* when only the object contours become "noisy". A contour noise can be generated by eliminating contour points (subtractive noise) or by adding contour points from the background (additive noise). Additive and subtractive noise can be generated for a picture independently.

The *level* of the noise indicates the percentage of the picture points that can be changed during the noise generation.

For example, producing an additive global noise at the level of 50% means that we randomly choose the half of the picture points, and change them to object points. If a chosen point is an object point then it remains unchanged.

We used bold characters in the tables to highlight the minimum values.

## 3. DESCRIPTION OF THE EXPERIMENT

**3.1. Skeletonizations.** To test the noise sensitivity of the skeletonizations five algorithms were examined. The analysed algorithms are listed below with a brief description about the way they work:

**1.:** E.S. Deutch's algorithm (DE), cf. [2]
: This classical thinning algorithm has two subcycles and uses $3 \times 3$ templates.

**2.:** T. Pavlidis' algorithm (PA), cf. [6]
: This is a Contour Sequential Algorithm, so only the contour points are processed.

**3.:** V.K. Govindan's algorithm (GO), see [3]
: This Pattern Adaptive Thinning Algorithm examines the object points along the contour, while an adaptive algorithm adjusts the thinning process to picture shape.

**4.:** N.J. Naccache and colleague's algorithm (NA), see [5]
: This is a Safe Point Thinning Algorithm which uses $3 \times 3$ templates, but the input pictures have to be smoothed first, because the algorithm is very sensitive to the "salt and pepper" (global) noise.

**5.:** R.-Y. Wu's algorithm (WU), cf. [9]
: This is a One-pass Parallel Thinning which uses $3 \times 4$ and $4 \times 3$ templates instead of $3 \times 3$ ones to avoid excessive erosion during the deletion of edge points. The authors claim that this algorithm produces perfectly 8-connected and *noise insensitive* results. (They are right as we shall see.)

Some of these algorithms are known from the literature as very efficient ("good") ones (e.g., Pavlidis' algorithm). We tried to select algorithms which are based on different models, and try to find out whether a relationship exists between the type of algorithm and the noise sensitivity. In the following we refer to these algorithms by two capital letters, as algorithms DE, PA, GO, NA, WU.

**3.2. Original pictures.** To perform the analysis we used 10 binary pictures; their properties are summarized in Table 3.2:

The images can be grouped as printed circuit boards (#1,#3), pictographs (#6,#7,#8), text information (#4,#5,#9), and sophisticated structured line-drawings(#2,#10), respectively. We found that these kinds of images often occur in practice, that is

TABLE 1. Properties of test pictures involved in the analysis

| # | Picture size | Number of object points (Area) | Number of contour points (Perimeter) |
|---|---|---|---|
| 1 | 100 × 100 | 4014 | 769 |
| 2 | 200 × 200 | 8343 | 5204 |
| 3 | 100 × 100 | 4326 | 961 |
| 4 | 320 × 200 | 14068 | 4018 |
| 5 | 100 × 100 | 2366 | 970 |
| 6 | 320 × 200 | 3468 | 2179 |
| 7 | 160 × 160 | 15007 | 3132 |
| 8 | 200 × 160 | 5016 | 1671 |
| 9 | 200 × 160 | 7901 | 2908 |
| 10 | 320 × 200 | 15833 | 6477 |

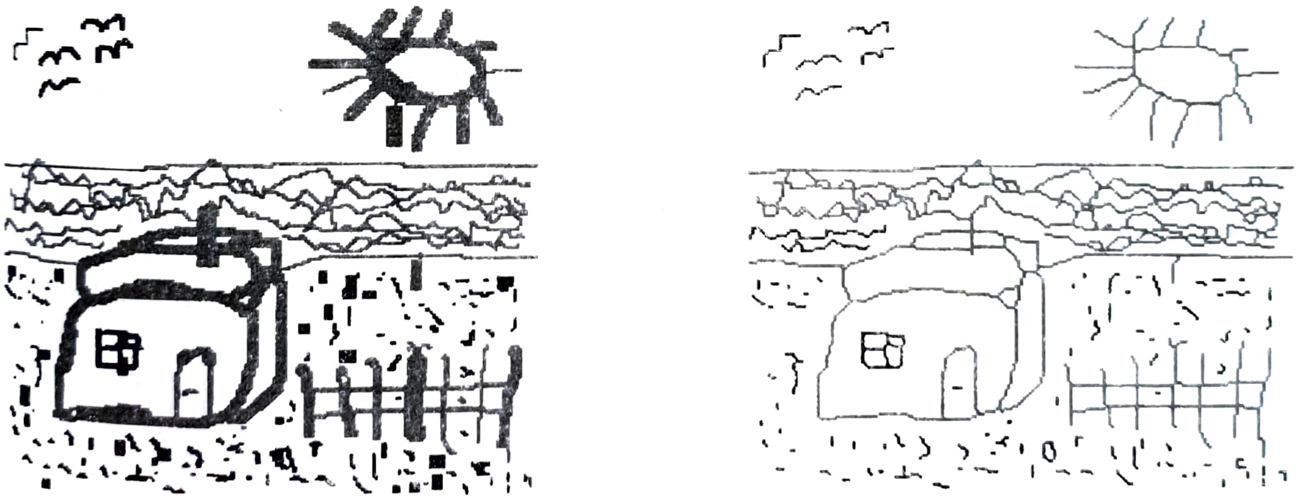

FIGURE 1. Picture #2 (a), and its skeleton (b) extracted by the algorithm DE

the reason why we chose them for this analysis. Figure 1(a) shows the test picture #2, and the result skeleton (extracted by DE) is shown in Figure 1(b).

TABLE 2. Noise corruptions applied to the original pictures

| Noise | A | B | C | D | E | F |
|---|---|---|---|---|---|---|
| Level | +2% | +2%/−2% | +10% | −10% | +5%/−10% | +15%/−15% |
| Type | global | global | contour | contour | contour | contour |

3.3. **Generating noise.** We corrupted our test pictures with uniformly distributed noises. Additive (background points become object points) and subtractive (objects points become background points) noises were used at different corruption levels for the whole pictures (global noise) and for only the object contours (contour noise). The levels of corruption are given in percents, a positive percentage values mean additive, negative values mean subtractive noise.

Table 3.3 shows the noise levels and types we generated in the test pictures. The noise corruptions used are denoted by A, B, C, D, E, and F, respectively.

Picture #2 is shown in Figure 2(a) after corrupting with a $+15\%/-15\%$ global noise, and the result of the skeletonization DE is on Figure 2(b). It is worth examining how the noise corruption affected the result of the skeletonization. For example, little circles can be seen on the sun in Figure 2(b), which are missing from Figure 1(b). The reason is that the subtractive noise ($-15\%$) eliminated some points from the main circle of the sun, thus holes appeared there, and the skeletonization preserved the topological features. Little line segments, and points in the sky can be seen as the consequence of the additive noise corruption.

3.4. **Reference skeleton.** There are several ways to find a good reference skeleton for further analysis; one was mentioned in the Introduction. To obtain the reference skeletons for the test pictures we used another method which was simpler and more suitable for our investigations. Our test pictures are artificial (ideal) images, which means that they are not corrupted by noise, so we can consider the result skeleton of a skeletonization in the test picture as the reference skeleton of the given skeletonization. This is a simple way to obtain a reference skeleton, and, on the other hand, we investigate only the effects of noise corruption, so this method does not mean a restriction. For example, we use Figure 1(b) as the reference skeleton of the algorithm DE.

3.5. **Calculating the distance of the result and the reference skeletons.** We obtained 50 skeletons after performing the 5 algorithms for the 10 original pictures and these skeletons were used as references. From the 10 pictures we produced 70 pictures for every kind of noise corruption, thus we had 4200 pictures corrupted by noise.

The 5 skeletonizations were performed for all the 4200 pictures, thus altogether we had 21000 result skeletons in the end.

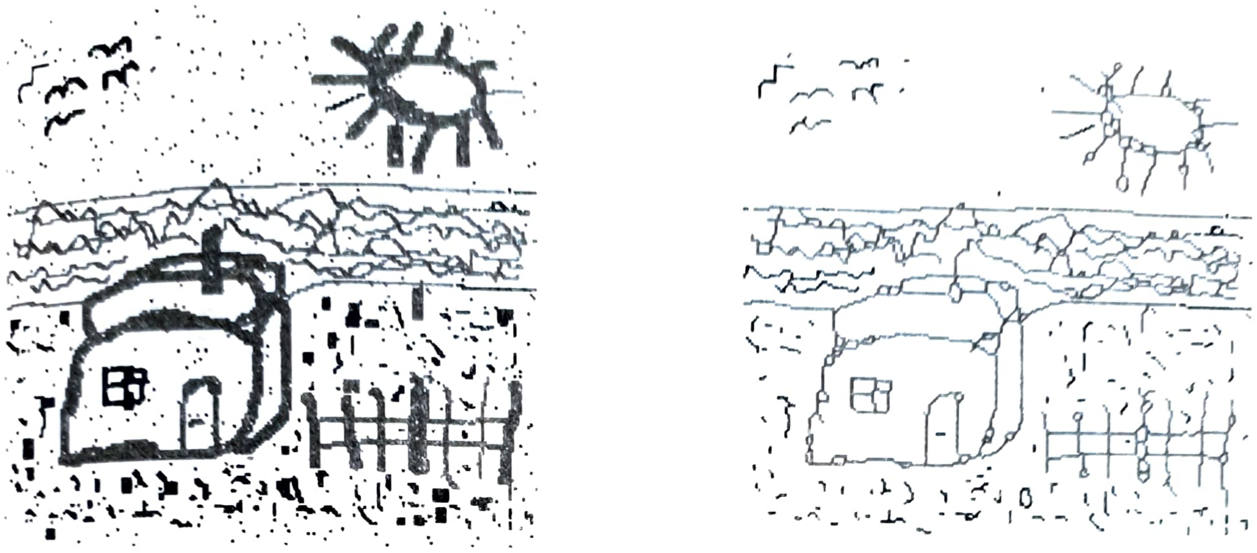

FIGURE 2. Picture #2 after corrupting with a +15%/ − 15% global noise (a), and its skeleton (b) extracted by the algorithm DE

As the next step of our analysis, we compared the result skeletons with the corresponding reference skeletons. To perform this comparison, first we translated the result skeleton both horizontally and vertically to find the best matching to the reference (i.e., to make the result skeleton cover the reference one). We chose the position for which the distance of the result and the reference skeleton was minimal. This technical procedure has to be executed as a preprocession, since the translation of the object implies the translation of the skeleton. For example, an additive/subtractive contour noise corruption translates a solid rectangle by one pixel if the additive part of the noise corrupts all the points of one side of the rectangle and the subtractive part corrupts the points of the opposite side. The maximal translation allowed was 2 pixels in any of the four directions.

After translating the result skeleton, we calculated its Hamming-distance (1) from the reference skeleton. Thus at the end of the calculation we had a database of 21000 experimental distance values, which we could use for further statistical investigations. To evaluate this data set we used the statistical program package SPSS[1].

[1]Copyright © SPSS Inc.

TABLE 3. The mean values of computation times (the values are given in seconds)

| # | DE | PA | GO | NA | WU |
|---|---|---|---|---|---|
| 1 | 0.428 | 1.005 | 0.657 | 0.513 | **0.358** |
| 2 | 0.779 | 2.570 | 1.901 | 1.276 | **0.752** |
| 3 | 0.421 | 1.174 | 0.711 | 0.570 | **0.399** |
| 4 | 3.092 | 7.267 | 4.667 | 5.728 | **2.699** |
| 5 | 0.178 | 0.554 | 0.392 | 0.265 | **0.168** |
| 6 | **0.311** | 0.983 | 0.772 | 0.604 | 0.312 |
| 7 | 2.338 | 5.254 | 3.270 | 3.108 | **2.114** |
| 8 | 1.111 | 2.084 | 1.419 | 1.738 | **0.829** |
| 9 | 0.741 | 1.973 | 1.435 | 1.166 | **0.623** |
| 10 | 2.757 | 6.384 | 5.125 | 3.866 | **2.238** |

## 4. STATISTICAL EVALUATION OF THE RESULT DATA

4.1. **Computation time.** This paper does not focus on the technical parameters of the skeletonizations, but sometimes it can be useful to know how much it takes an algorithm to process a picture corrupted with a special kind of noise. Table 4.1 contains the mean values of the algorithms' computation times for each of the pictures. The computation time certainly depends on the size and on the difficulty level of the picture. From the skeletonizations examined we found the one-pass algorithm WU to be the fastest, according to this table. A table containing computation speed for lots of algorithms can be found in [8, p. 239].

4.2. **Picture – Distance, and Noise type – Distance relations.** We examined the performance of the algorithms with respect to the pictures and to the noise types. Figure 4.2 contains the mean distance values of the test and the reference skeletons for every picture, and the result of the skeletonizations are shown by polygons. Smaller distance values mean better matching to the reference skeletons, so that skeletonization produces the best result which has the smallest distance values (i.e., the lowest polygon).

Figure 4.2 contains information about a similar analysis, but here the noise types were examined instead of the pictures. The same conclusions can be drawn, i.e., that algorithm produces the best result which has the smallest values.

Detailed tables of the distance values can be found in the Appendix. Appendix I contains a table about the expected values (means) of the distance values, while Appendix II shows their standard deviations.

4.3. **Performance indices for skeletonizations.** We assigned a rank value to the algorithms according to the distance values they produced in the given test.

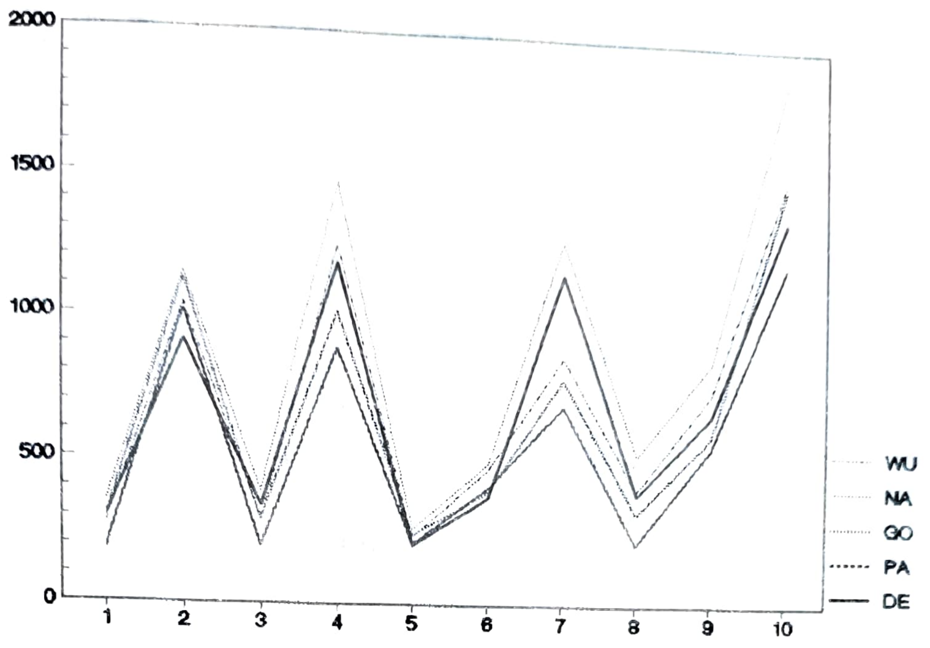

FIGURE 3. Picture – Distance relation

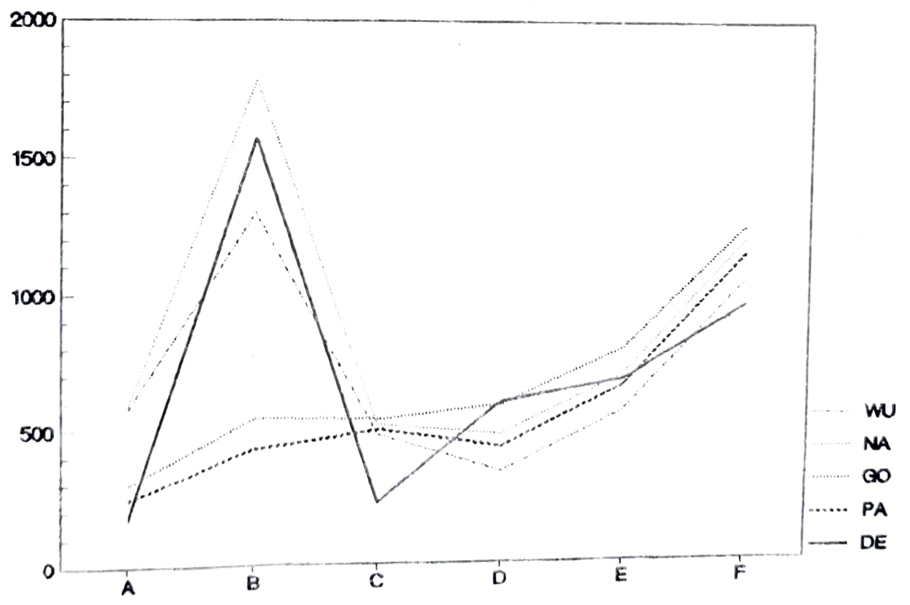

FIGURE 4. Noise type – Distance relation

TABLE 4. Performance indices for the pictures

| # | DE | PA | GO | NA | WU |
|---|---|---|---|---|---|
| 1 | 2.50 | **2.17** | 4.17 | 3.50 | 2.67 |
| 2 | **2.00** | 3.17 | 4.00 | 3.50 | 2.33 |
| 3 | 2.67 | **2.00** | 4.00 | 4.00 | 2.33 |
| 4 | 2.67 | **2.00** | 3.50 | 3.50 | 3.17 |
| 5 | **2.17** | 2.83 | 4.17 | 3.33 | 2.50 |
| 6 | **2.33** | 3.50 | 3.17 | 3.17 | 2.83 |
| 7 | 2.67 | **2.33** | 3.33 | 4.33 | **2.33** |
| 8 | 3.00 | **1.83** | 3.83 | 4.17 | 2.17 |
| 9 | **2.50** | **2.50** | 3.50 | 3.67 | 2.83 |
| 10 | **2.17** | 2.33 | 4.17 | 4.17 | **2.17** |

TABLE 5. Performance indices for the noise types

| | DE | PA | GO | NA | WU |
|---|---|---|---|---|---|
| A | **1.00** | 2.10 | 2.90 | 4.50 | 4.50 |
| B | 3.80 | **1.00** | 2.00 | 4.70 | 3.50 |
| C | **1.00** | 3.30 | 4.00 | 3.40 | 3.30 |
| D | 4.60 | 2.30 | 4.20 | 2.90 | **1.00** |
| E | 3.20 | 2.70 | 4.70 | 3.40 | **1.00** |
| F | **1.20** | 3.40 | 4.90 | 3.50 | 1.90 |

TABLE 6. Overall performance indices for the algorithms

| DE | PA | GO | NA | WU |
|---|---|---|---|---|
| **2.46** | **2.46** | 3.78 | 3.73 | 2.53 |

The algorithm with the smallest distance value got rank 1, and the largest value got rank 5. Identical distance values got the same rank, and in this case the next rank value was skipped (e.g., 1, 1, 3, 3, 5). By averaging these rank values we were able to assign a performance index to each of the skeletonizations. Table 4.3 shows the performance indices for the pictures, while Table 4.3 shows the same for noise types.

By averaging these performance indices we could calculate an overall performance index for every skeletonization. The Table 4.3 contains these indices:

According to this table we can conclude that algorithms DE, PA, and WU produce better skeleton matching than GO or NA. This result explains why

TABLE 7. Performance indices for pictures (only for contour noise)

| # | DE | PA | GO | NA | WU |
|---|---|---|---|---|---|
| 1 | 2.25 | 2.50 | 5.00 | 3.25 | **2.00** |
| 2 | 2.00 | 4.00 | 4.75 | 2.75 | **1.50** |
| 3 | 2.50 | 2.25 | 4.75 | 3.75 | **1.75** |
| 4 | 3.00 | 2.25 | 4.00 | 3.25 | **2.25** |
| 5 | 2.00 | 3.50 | 5.00 | 2.75 | **1.75** |
| 6 | 2.50 | 4.25 | 3.75 | 2.50 | **2.00** |
| 7 | 3.00 | 2.75 | 3.75 | 4.00 | **1.50** |
| 8 | 3.25 | 2.00 | 4.50 | 3.75 | **1.50** |
| 9 | 2.50 | 3.00 | 4.00 | 3.25 | **2.25** |
| 10 | 2.00 | 2.75 | 5.00 | 3.75 | **1.50** |

TABLE 8. Overall performance indices for the algorithms (only for contour noise)

| DE | PA | GO | NA | WU |
|---|---|---|---|---|
| 2.50 | 2.92 | 4.45 | 3.30 | **1.80** |

the authors of algorithm NA warn for smoothing before skeletonizing, as algorithm NA has the worst results in the case of global noise which has a "salt and pepper" effect.

4.4. **Performance indices only for contour noise corruption.** During our investigations we found that it was a bit unfair to involve global noise corruption into the analyses, as mainly contour noise occurs in practice, and these algorithms are quite sensitive to global noise. That is why we calculated performance indices of the algorithms for only contour noise corruption. The Table 4.4 is similar to Table 4.3, but global noises were excluded from this analysis.

According to the above discussed method, we also calculated the overall performing indices of the algorithms for the contour noise corruptions (Table 4.4).

This table indicates larger difference between the "goodness" of the algorithms. Algorithm WU seems to be the most reliable one (so the authors are right about noise immunity), but DE is better in coping with higher noise level. Algorithm GO produced the worst performance with respect to noise sensitivity.

4.5. **Variance analysis.** In our analysis we grouped the distance values on the basis of the algorithm name as a factor. It turned out that this factor is significant for the model, and the following groups were obtained:

TABLE 9. Pictures grouped according to the skeletonizations

| # | 1 | 3 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|
| 1 | | DE, PA, GO | DE, GO, NA, WU | | | DE, GO | |
| 3 | DE, PA, GO | | DE, GO, NA, WU | | | DE, GO, WU | |
| 5 | DE, GO, NA, WU | DE, GO, NA, WU | | | | | PA, GO |
| 6 | | | | | PA | | |
| 7 | | | | PA | | | PA, GO |
| 8 | DE, GO | DE, GO, WU | PA, GO | | | | |
| 9 | | | | | PA, GO | | |

- DE
- PA
- GO, NA
- WU

This result shows that algorithms GO and NA produce similar results for any kind of noise and picture type.

We examined every algorithm if it worked similarly when only contour noise corruption was generated in the pictures. In this analysis the distance values were grouped on the basis of the picture number as a factor for all the skeletonizations. The following table shows the cases when at least two pictures belong to the same group. A picture can belong to more than one group (it is known from factor analysis), which means that there are similarities in some details, but this relation is not transitive. Two pictures belong to the same group if the algorithm produces similar results. For example, Pictures #1 and #3 usually belong to the same group (because they have the same type, as both of them are pictures of printed circuit boards). Further similar conclusions can be derived, e.g., Pictures #2 and #10 always form separate groups, as they have sophisticated structure.

4.6. **Regression analysis.** Our statistical investigations discovered strong relationship between the level of the noise corruption, and the distance values. They are correlated at the level of $r = 0.7584$, and the hypothesis that these variables are uncorrelated should be rejected at every normally used significance level (95%, 99%). Moreover, a linear relationship was conjectured and a regression analysis proved this hypothesis. From $R$ statistics we obtained that the linear model is acceptable, and the hypothesis that the linear model is not suitable should be rejected at every normally used significance level (95%, 99%).

## 5. CONCLUSIONS

This paper presents statistical results about the tolerance of five skeletonization algorithms with respect to noisy images. A large database of 21000 skeletons was used to obtain performance indices for the algorithms. Linear correlation was detected between the level of the noise and the distance of the reference and the test skeletons. The algorithms could be grouped according to their tolerance with respect to different types of noises and images. The calculated rank values of the algorithms may help one to choose an algorithm which produces the most reliable result for a given type of image which is corrupted with a given type of noise. It seems to be interesting to go on with analysing other skeletonizations which are based on other models, or to make investigations in a higher dimension (3D).

## APPENDIX I

The Table 5 contains the expected value (mean) of the distance values produced by the algorithms. The column headings represent the generated noise (heading * indicates the case when every picture corrupted with every kind of noise was involved in the analysis). Row numbers represent the test pictures. The smallest distance values are highlighted with bold numbers for every picture–noise pair.

## APPENDIX II

The Table 5 contains the standard deviation of the distance values produced by the algorithms. The column headings represent the generated noise (heading * indicates the case when every picture corrupted with every kind of noise was involved into the analysis). Row numbers represent the test pictures. The smallest values are highlighted with bold numbers for every picture–noise pair.

TABLE 10.

| # | * | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|
| 1 | 296<br>**185**<br>295<br>345<br>269 | **56**<br>84<br>126<br>159<br>162 | 988<br>**170**<br>250<br>983<br>633 | **54**<br>151<br>238<br>149<br>160 | 181<br>134<br>283<br>159<br>**109** | 207<br>206<br>357<br>229<br>**189** | **290**<br>367<br>518<br>389<br>358 |
| 2 | **914**<br>1018<br>1129<br>1152<br>1042 | **246**<br>378<br>479<br>730<br>725 | 1089<br>**596**<br>775<br>1430<br>1169 | **409**<br>915<br>884<br>788<br>772 | 959<br>881<br>1130<br>821<br>**686** | 1113<br>1246<br>1404<br>1159<br>**1041** | **1670**<br>2093<br>2102<br>1980<br>1860 |
| 3 | 343<br>**201**<br>295<br>394<br>291 | **52**<br>72<br>104<br>158<br>151 | 1048<br>**177**<br>262<br>1017<br>643 | **64**<br>159<br>232<br>211<br>188 | 252<br>141<br>250<br>196<br>**128** | 275<br>232<br>358<br>291<br>**223** | **367**<br>426<br>565<br>488<br>414 |
| 4 | 1192<br>**896**<br>1021<br>1474<br>1249 | **306**<br>439<br>498<br>1175<br>1181 | 2925<br>**749**<br>905<br>3290<br>3530 | **344**<br>785<br>819<br>828<br>852 | 971<br>598<br>829<br>680<br>**431** | 1099<br>991<br>1183<br>1054<br>**847** | **1505**<br>1815<br>1893<br>1815<br>1652 |
| 5 | **211**<br>217<br>247<br>271<br>243 | **64**<br>94<br>117<br>182<br>192 | 359<br>**147**<br>180<br>442<br>348 | **88**<br>190<br>207<br>162<br>189 | 197<br>179<br>231<br>181<br>**134** | 226<br>261<br>301<br>245<br>**214** | **334**<br>428<br>449<br>411<br>384 |
| 6 | **380**<br>416<br>403<br>501<br>478 | **126**<br>184<br>176<br>483<br>487 | 406<br>**268**<br>280<br>693<br>643 | **185**<br>370<br>293<br>318<br>335 | 400<br>347<br>399<br>304<br>**265** | 469<br>503<br>507<br>449<br>**413** | **692**<br>825<br>765<br>760<br>727 |
| 7 | 1172<br>**706**<br>801<br>1288<br>878 | **123**<br>269<br>324<br>484<br>403 | 1633<br>**661**<br>830<br>3488<br>2235 | **302**<br>727<br>810<br>808<br>538 | 818<br>410<br>392<br>493<br>**290** | 934<br>749<br>844<br>857<br>**570** | **1223**<br>1421<br>1605<br>1598<br>1234 |
| 8 | 392<br>**219**<br>326<br>534<br>409 | **118**<br>133<br>192<br>580<br>570 | 993<br>**221**<br>310<br>1279<br>984 | **60**<br>125<br>202<br>225<br>166 | 337<br>148<br>288<br>224<br>**110** | 374<br>238<br>367<br>325<br>**206** | 470<br>447<br>594<br>570<br>**417** |
| 9 | 681<br>**564**<br>605<br>871<br>759 | **205**<br>280<br>314<br>730<br>748 | 1475<br>**407**<br>470<br>1747<br>1338 | **237**<br>516<br>478<br>490<br>531 | 566<br>394<br>503<br>448<br>**309** | 660<br>641<br>707<br>663<br>**555** | **944**<br>1148<br>1156<br>1146<br>1075 |
| 10 | 1382<br>**1219**<br>1506<br>1880<br>1528 | **380**<br>530<br>654<br>1156<br>1142 | 2788<br>**878**<br>1131<br>3417<br>2427 | **485**<br>977<br>1119<br>1090<br>991 | 1197<br>1010<br>1482<br>1171<br>**875** | 1389<br>1440<br>1840<br>1656<br>**1336** | **2051**<br>2481<br>2812<br>2787<br>2395 |

## References

[1] Y.-S. Chen, W.-H. Hsu, *A comparison of some one-pass parallel thinnings*, Pattern Recognition Letters, 11 (1990), pp. 35–41.

[2] E. S. Deutsch, *Thinning algorithms on rectangular, hexagonal and triangular arrays*, Communications of the ACM, 15 (1972), pp. 827–836.

[3] V. K. Govindan, A. P. Shivaprasad, *A pattern adaptive thinning algorithm*, Pattern Recognition, 20 (1987), pp. 623–637.

[4] L. Lam, S.-W. Lee, C. Y. Suen, *Thinning methodologies – A comprehensive survey*, IEEE Transactions on Pattern Analysis and Machine Intelligence, 14 (1992), pp. 869–885.

TABLE 11.

| # | * | A | B | C | D | E | F |
|---|---|---|---|---|---|---|---|
| 1 | 323<br>**97**<br>130<br>300<br>184 | **17**<br>24<br>38<br>18<br>19 | 71<br>**50**<br>69<br>64<br>63 | **17**<br>36<br>43<br>28<br>26 | 29<br>32<br>48<br>29<br>**23** | 35<br>38<br>47<br>32<br>**29** | **32**<br>45<br>41<br>40<br>37 |
| 2 | 477<br>556<br>530<br>448<br>**409** | **28**<br>45<br>87<br>34<br>38 | 85<br>63<br>98<br>68<br>**60** | **29**<br>69<br>92<br>43<br>46 | 55<br>65<br>101<br>46<br>**42** | **56**<br>74<br>96<br>58<br>**56** | 62<br>84<br>69<br>59<br>**54** |
| 3 | 337<br>**118**<br>152<br>302<br>186 | **16**<br>22<br>33<br>19<br>18 | 68<br>56<br>78<br>62<br>**53** | **19**<br>34<br>44<br>27<br>26 | 29<br>28<br>51<br>27<br>**26** | 39<br>37<br>45<br>36<br>**33** | **35**<br>49<br>61<br>42<br>42 |
| 4 | 885<br>453<br>**447**<br>892<br>687 | **34**<br>57<br>74<br>50<br>48 | 116<br>**103**<br>113<br>111<br>105 | **48**<br>80<br>91<br>64<br>75 | 71<br>66<br>84<br>58<br>**48** | 70<br>101<br>80<br>**70**<br>**70** | 73<br>104<br>93<br>**72**<br>83 |
| 5 | 115<br>111<br>110<br>117<br>**95** | 17<br>23<br>29<br>**16**<br>19 | 48<br>**28**<br>32<br>39<br>30 | **18**<br>27<br>32<br>22<br>27 | 32<br>29<br>30<br>22<br>**20** | 30<br>33<br>37<br>**25**<br>29 | 31<br>29<br>32<br>**27**<br>33 |
| 6 | 190<br>212<br>197<br>176<br>**167** | **18**<br>37<br>34<br>25<br>25 | 46<br>46<br>52<br>**41**<br>**41** | **21**<br>39<br>44<br>26<br>36 | 38<br>43<br>49<br>**27**<br>32 | 43<br>39<br>44<br>38<br>**36** | 43<br>51<br>49<br>**32**<br>40 |
| 7 | 1165<br>**386**<br>442<br>1055<br>681 | **27**<br>122<br>148<br>38<br>29 | **98**<br>186<br>221<br>**98**<br>102 | **45**<br>123<br>130<br>65<br>69 | 70<br>110<br>75<br>49<br>**38** | 76<br>115<br>149<br>75<br>**59** | 84<br>101<br>105<br>**82**<br>84 |
| 8 | 308<br>**117**<br>142<br>366<br>305 | **22**<br>26<br>41<br>32<br>34 | 81<br>**54**<br>60<br>74<br>65 | **17**<br>32<br>45<br>31<br>33 | 42<br>27<br>31<br>23<br>**19** | 38<br>42<br>45<br>30<br>**29** | 49<br>41<br>51<br>40<br>**35** |
| 9 | 440<br>291<br>**278**<br>456<br>353 | **29**<br>51<br>52<br>35<br>35 | 99<br>**58**<br>66<br>88<br>62 | **37**<br>71<br>54<br>47<br>51 | 48<br>61<br>56<br>**38**<br>39 | 49<br>58<br>54<br>**44**<br>45 | 57<br>65<br>60<br>**45**<br>50 |
| 10 | 847<br>**629**<br>695<br>906<br>645 | **43**<br>54<br>70<br>44<br>50 | 101<br>**75**<br>99<br>96<br>83 | **40**<br>66<br>84<br>57<br>63 | 66<br>73<br>94<br>63<br>**56** | 67<br>76<br>97<br>67<br>**62** | **78**<br>99<br>112<br>**78**<br>101 |

[5] N. J. Naccache, R. Shinghal, *SPTA: A proposed algorithm for thinning binary patterns*, IEEE Transactions on Systems, Man, and Cybernetics, 14 (1984), pp. 409–418.

[6] T. Pavlidis, *A thinning algorithm for discrete binary images*, Computer Graphics and Image Processing, 13 (1980), pp. 142–157.

[7] C. Ronse, *A topological characterization of thinning*, Theoretical Computer Science, 43 (1986), pp. 31–41.

[8] C. Y. Suen, P. S. P. Wang (ed), *Thinning methodologies for pattern recognition*, World Scientific Publishing Co., Singapore, 1994.

[9] R.-Y. Wu, W.-H. Tsai, *A new one-pass parallel thinning algorithm for binary images*, Pattern Recognition Letters, 13 (1992), pp. 715–723.

ATTILA FAZEKAS AND ANDRÁS HAJDU

Lajos Kossuth University, 4010, Debrecen PO Box 12, Hungary
*E-mail address*: {fattila,hajdua}@math.klte.hu