Skeletonization of Sparse Shapes using Dynamic Competitive Neural Networks

Waldo Hasperué [1], Leonardo Corbalán [2], Laura Lanzarini [3], Oscar Bría [4]

III-LIDI (Institute of Research in Computer Sciences LIDI)
Faculty of Computer Sciences. National University of La Plata.
La Plata, Argentina, 1900
{whasperue, corbalan, laural, onb}@lidi.info.unlp.edu.ar

Abstract

The detection of regions and objects in digital images is a topic of utmost importance for solving several problems related to the area of pattern recognition. In this direction, skeletonization algorithms are a widely used tool since they allow us to reduce the quantity of available data, easing the detection of characteristics for their recognition and classification. In addition, this transformation of the original data in its essential characteristics eases the elimination of local noise which is present in the data input.

This paper proposes a new skeletonization strategy applicable to sparse images from a competitive, dynamic neural network trained with the AVGSOM method. The strategy developed in this paper determines the arc making up the skeleton combining AVGSOM non-supervised learning with a minimum spanning tree.

The proposed method has been applied in images with different spanning shape and degree. In particular, the results obtained have been compared to existing solutions, showing successful results.

Finally, some conclusions, together with some future lines of work, are presented.

Palabras clave: Skeletonization, Dynamic Self-Organizing Maps, Neural Networks, Digital Image Processing.

1. Introduction

Shape recognition and analysis represent one of the main problems of the areas such as pattern recognition, image processing, and computer vision. One of the major steps that should be taken in order to achieve good results is to work with a proper characterization of input data.

In this direction, skeletonization algorithms are a widely used tool, which allows reducing the quantity of the available data, thus easing the extraction of characteristics for their later recognition and classification [4] [14] [15]. In addition, this transformation of the original data in its essential characteristics eases the elimination of local noise which is present in the data input.

[1] Waldo Hasperué has a Licenciate Degree in Computer Science. He currently has a CIC Scholarship.
[2] Leonardo Corbalan has a Licenciate Degree in Computer Science.
[3] Laura Lanzarini has a Licenciate Degree in Computer Science. She is currently Full Time Professor at the School of Computer Science.
[4] Oscar Bría has a Master Degree in Computer Science. He is currently co-chair Professor at the School of Computer Science.

Specially, within the area of digital image processing, there exist several skeletonization algorithms, though most of them present problems when we deal with images with low pixel connectivity. For instance, under-illumination or errors in the capture may easily generate images with these characteristics. A similar problem is present in the analysis of old documents' images or those printed with low quality. In these cases, conventional skeletonization techniques offer really poor results.

The existing methods may be grouped in three categories. The first is made up by skeletonization methods based on distance transforms [2] [8]. This type of skeletons allows reconstructing the original image. The second category is composed by successive object thinning methods, which attempt to reduce it to its central axes. Finally, the third category is represented by non-iterative methods which generate skeletons with arc-connected shapes [3] [10] [11] [13]. Generally, this type of method puts certain critical points in the first place so as to then compute a path to join them through the object. This paper introduces a new method which presents characteristics of the elements of the second and third category.

2. Objective

The proposal of the paper is to present a new skeletonization method, capable of approaching the main shape curve that is present in a sparse image, basing on a dynamic, competitive neural network. The network is trained with AVGSOM [7] and receives as input the coordinates of each of the image pixels; in this way, the elements making up the network architecture turn into vectors (points or pixels) typical of the most representative areas.

From this initial distribution, a tree is built up and several local adjustments are made over it so as to avoid having too distant nodes. In this phase, small AVGSOM networks are trained, independently of the original network, with the objective of making partial corrections to the tree, thus allowing us to achieve a proper skeleton in an acceptable response time. Figure 1 exemplifies the results of the proposed algorithm for three different shapes.

The following sections are organized as follows: In the first place, section 2 will make a brief introduction of dynamic self-organizing maps, which will allow us to understand the AVGSOM method training algorithm described in section 3. See [7] for a more detailed description of this method. Section 4 describes the proposed skeletonization algorithm. Finally, the results obtained are analyzed and some future lines of work are presented.

3. Dynamic Self-Organizing Maps

As it can be seen in figure 1, the skeleton we are looking for is based on the connection, through arcs, of representative positions of the shape contained in the sparse image. In order to establish such position, it is necessary to group the image pixels by some similarity criterion, giving place to clusters formation. The searched-for positions will be the representative of each cluster and they generally correspond to the cluster's centroid. This type of problems is successfully solved by using a competitive neural network.

Competitive neural networks are made up by artificial neurons which, as its name suggests, "compete" among each other for the representation of input patterns. For this, a weight vector is associated to each neuron, against which each input data is compared by using a similarity measure depending on the problem. These weight vectors are learnt by the network through a non-supervised learning process.

One of the most famous competitive neural networks is the Self Organizing Map (SOM) defined by Kohonen [9]. This network, even though have given successful results in several situations, presents a disadvantage: the use of a static architecture, because the quantity of neurons making up the architecture as well as its connection method should be defined a priori, before starting the training, all of which thus conditions the network's efficiency and efficacy.

To solve this, alternative solutions have been proposed, the so-called Dynamic Self-Organizing Maps, which keep the capacity of properly maintaining the data topology, thus allowing the incorporation or elimination of elements during the training. In this way, it is not necessary to indicate a priori the quantity of neurons to be used since the architecture is of variable dimension.

In general, most of the existing strategies for defining Dynamic Self-Organizing Maps present the following characteristics [6]:

- The network structure is a graph made up by interconnected competitive neurons. The connection is regular and plays an important role at the time of visualizing the reduction of input space.
- Each network neuron corresponds to a prototype vector, which aims at representing a set of similar input data set. The similarity measure to be used depends on the problem.
- Training is carried out through a competitive process in which neurons aim at representing input data. For each datum, its resemblance with the prototype vector of each neuron is evaluated, considering the winner as the most alike. Adaptation is mainly applied to the wining neuron and, to a lesser extent, to its closest surrounding. This is what allows us to gradually correct the structure so as to preserve the topology.
- In each step of the adaptation, local error data is stored in the winner neuron. This aims at avoiding that a same element of the network stores the representation of most of the input patterns. The error calculation depends on the application.
- The stored error information is used to determine where new units should be inserted in the network. When an addition is carried out, the error information is locally redistributed thus avoiding new additions in a same place.

We can see in the previous list that the first three characteristics correspond to the SOM defined by Kohonen [9], while the last two are related to the need of identifying the place in which new elements are to be added in the architecture.

There exist several solutions for determining the architecture of a dynamic competitive neural network [1] [5] [7], whose main differences are rooted in the way neighbor neurons are connected and the strategy used for inserting new elements. In this paper, we have used AVGSOM [7], whose performance is detailed in the next section.

4. AVGSOM

This method makes use of a rectangular grid in which each neuron has a maximum of four neighbors. Training begins with a minimum structure of four neurons, in which each one has two neighbors, making up a matrix of 2x2 and the corresponding prototype vectors are initialized at random. At each iteration step, the prototype vectors are adapted, and the error is stored in the winning neuron, as usual.

The fact that a neuron overpasses, during the training, the stored error threshold established a priori, points out the need of adapting the structure in order to avoid the excessive accumulation of patters around the winner. Any information on the structure should respect the initially proposed rectangular topology, since this eases the visualization of input data.

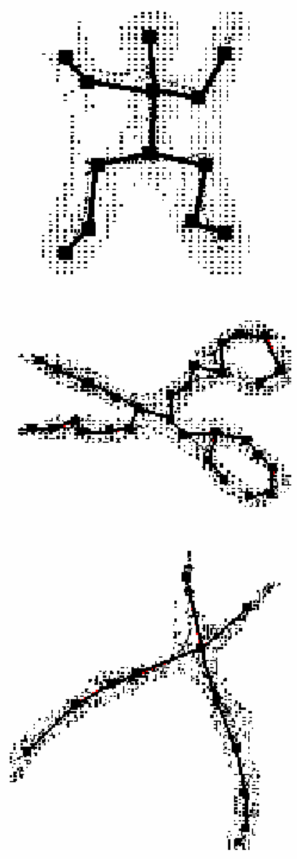


Figure 1. Different shapes and its corresponding skeletons obtained with the proposed method.

For such reason, if the winner has four neighbors, the difference between the winner and its closest and direct neighbors should be reduced, thus allowing other elements of the network to win the contest during the successive adaptive steps.

If the winner has less than four direct neighbors, a new element will be inserted in the structure, whose initial weight vector will be close to that of their neighbors so as to not distort the network topology during the learning.

The method used for determining the weight vector of a new neuron in AVGSOM is the average of the prototype vectors of those that will be neighbors of the new neuron. In this way, not only the elements of the network relate to each other within a graph but also the weight vector of the new neuron lies "in the middle" of its neighbor weight vectors in the input space, preserving the data topology almost totally.

```
Begin with a neural network made up by four neurons in which each has two
neighbors.
Initialize, with random values, the prototype vectors corresponding to each
Network neuron.

Repeat
   For each input pattern
     ▪ Enter the pattern to the neural network
     ▪ Identify the winner neuron and adapt its weight vector and those of
       its neighbors, as SOM usually does.
     ▪ Store in the winner neuron the magnitude of the corresponding
       error. Such value corresponds to the similarity measure evaluated
       in the previous point.
   End For
   For each network neuron
     ▪ If the neuron error surpasses the GT threshold then
          ▪ If the neuron has four direct neighbors then
                ▪ Distribute the error stored by the winner neuron among
                  its direct neighbors. With this, in the next iterations,
                  the neighbors will have more opportunities for saturating
                  themselves, and in this way “push” the error towards the
                  limits of the network, thus achieving its expansion.
              Otherwise
                ▪ Select at random one of the free spaces to generate the
                  new neighbor neuron.
                ▪ The prototype vector corresponding to this new neuron is
                  computed as the average of the prototype vectors of those
                  which will be its neighbors.
                ▪ Assign zero as stored error for this new neuron.
                ▪ Assign the zero value to the error stored by this neuron.
     ▪ If this neuron never wins then
          ▪ Increase by 1 its failure counter.
     ▪ If the failure counter of this neuron reaches a pre-established
       threshold then it should be eliminated.
   End For
Until no new neurons are created or the growing rate is minimum.
```

Figure 2. Different shapes and its corresponding skeletons obtained with the proposed method.

```
Establish threshold τ
Train an AVGSOM network
For each point in the image determine the neuron which represents
it
Build a MST with the network's neurons
While there exists an arc with a weight greater than τ
    ▪ Select arc a with the greatest weight
    ▪ Let $n_1$ and $n_2$ be the nodes connected by arc a
    ▪ Create a new neuron $n_3$ whose weights are the average of
      weights $n_1$ and $n_2$
    ▪ Initialize an AVGSOM network with the three neurons $n_1$, $n_2$
      and $n_3$
    ▪ Train this network with the points represented by $n_1$ and $n_2$
    ▪ Add the neurons of the AVGSOM network trained in the
      previous step to the set of neurons.
    ▪ Update MST.
End While
```

Figure 3. Algorithm to obtain a shape's skeleton using AVGSOM.

Figure 2 details the training algorithm used by AVGSOM.

GT threshold is computed as GT = -D * ln(SF), SF being a value between 0 and 1 corresponding to the dispersion factor, indicated as parameter [1].

A complete description of the method can be found in [7].

5. Skeletonization Algorithm from AVGSOM

The algorithm aims at obtaining the skeleton of an image. Even though the examples shown in this paper make use of two-dimension binary images, they can be directly applied over color images or data inputs of more than three-dimensions.

The first step consists in placing the artificial neurons in the most representative areas of the image. This is achieved by training an AVGSOM network using the color of each pixel as input. As explained in the previous section, the network starts with a minimum structure and makes use of a really low spanning factor (10^{-4}), so as to obtain a structure with few elements. As a result of this stage, we obtain a set of input pattern representative neurons. Interpreting each of them as the nodes of a completely connected graph - where the weight of each arc represents the distances between the neurons it connects - the minimun spanning tree (MST) is searched.

A fundamental parameter of this algorithm is the threshold value **τ** which specifies the maximum distance that may exist between two nodes (neurons) of the tree. For such reason, an iterative process is used to segment the arcs whose values overpass such threshold. The process ends when all the tree arcs are below or equal to τ.

Figure 3 represents the proposed algorithm pseudocode.

During each iteration step, the arc with greatest weight, called a, is segmented. For it, and after it is identified, a new AVGSOM network is built up, and which will be initially composed by three neurons: those connected by arc a, called n1 and n2 , and a new neuron, called n3, whose initial weights are the average of the weights of the two previous. This network will be trained only with the pixels which, up to now, are represented by n1 and n2. At the end of the training, arc a is eliminated from the tree and, in its place, the neurons of the small AVSGOM network (recently trained) are added respecting the connection conditions of the minimum spanning tree. This process goes on until the weights of all the tree arcs are below threshold τ. Figure 4 shows this process.

Figure 4.a) shows the neurons obtained by the first training of the neural network using all the image pixels. The quantity of neurons is determined by the AVGSOM method. Then, they are connected by using a MST, giving as result the skeleton of figure 4.b). From this point an iterative process begins, which searches the arc with higher length in the tree and replaces it with a sub-tree, provided that its weight is greater than the pre-established threshold τ. Figure 4.c) shows the result of this first modification. It is important to notice that this second training is not carried out over the entire figure, but only in the area referring to the arc to be replaced. Notice the modification of the positions of the two original neurons. This process keeps on repeating itself (Figure 4.c and 4.d) until no tree arc is left with a weight greater than τ. Take into account that, in each step, more than one neuron can be added. Figure 4.e) shows the final skeleton.

6. Results obtained

Various tests have been carried out with sparse images using different spanning degrees and threshold values. In the first experiment, the algorithm was applied to a collection of objects. The original images of these objects were color photos to which a thresholding computed by the Otsu algorithm were applied. The images' low quality is due to the fact that no processing type –in relation to illumination and brightness - was carried out. In addition, in some cases, the threshold was slightly modified so as to increase noise. The results of these tests can be seen in figure 1 and 5.

In a second experiment, the skeleton of an object pair was computed with different noise levels. In each case, the shape pixel number varied from 0% to 75%. In both cases, the sparse factor (SF) used for training the first neural network was 10-4. The responses obtained can be seen in figure 6 and in table 1.

Table 1 allows noting that, even though the tree undergoes several modifications, the quantity of patterns involved in each of them is a reduced number. For instance, the six-digit image without noise has 1274 pixels and begins with a tree made up by 12 neurons. Obtaining the final skeleton required 5 modifications and each of them implied, in average, 17.19 % of patterns.

In addition, as the number of iterations increases the quantity of patterns involved decreases. This is due to the fact that the competitive neurons adjust, every time in a better way, the input pattern space. In those places in which arcs are longer, the proposed method adds new representatives reducing the quantity of patterns assigned to each of them. Figure 7 represents the quantity of patterns involved in each of the iterations carried out to obtain the six-digit image tree with different noise levels. In it, we can notice how the quantity of used patterns decreases.

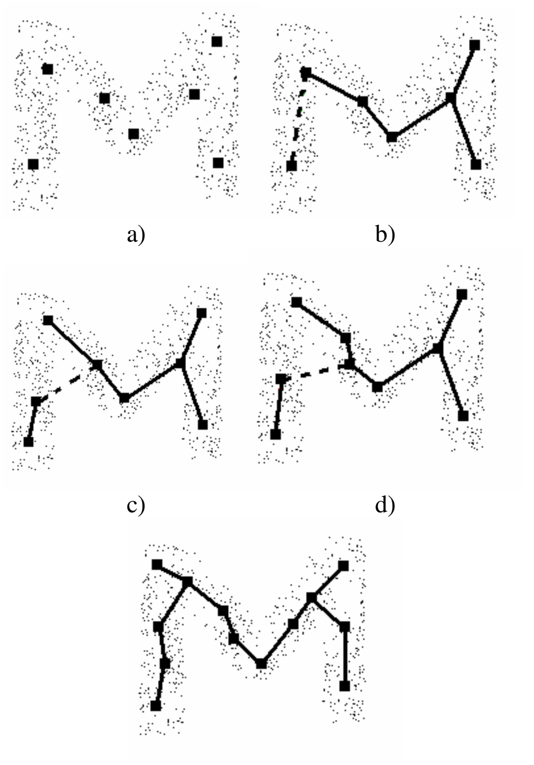


Figure 4: Description of the proponed skeletonization method. a) Neurons of the original AVGSOM network, b) Connection of such neurons with a MST. The broken arc overpasses the threshold, c) Result of the first replacement, d) Skeleton after the second replacement, e) Skeleton after the fifth replacement.

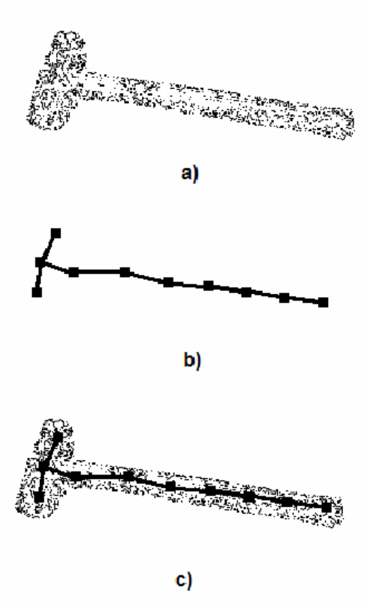


Figure 5. a) Original Figure. b) Tree found by the proposed method. c) The tree over the original figure.

The third experiment consisted in applying the method so as to find the skeleton of a text. Figure 8 shows in a) a text obtained from a highly damaged document, and in b) the skeleton resulting from the application of the proposed method to each of the letters [12]. Figure 9 shows the same process applied to a handwritten text in a ceramic teapot.

7. Conclusions

A new skeletonization method applicable to sparse images not requiring connectivity among pixels that determine the shape has been presented. Its performance is based on identifying the most representative areas of the image through competitive neurons and connecting them in a tree form. The skeleton is built up by an iterative process which corrects both the quantity and location of neurons in the input points space, keeping always the connection style.

Table 1. For the images of Figure 6, the table shows the quantity of neurons obtained from the first AVSGOM training, the quantity of modifications made to obtained the final skeleton, the average percentage of patterns used at each modification in relation to the total patterns and the quantity of neurons (tree nodes) at the time the algorithm is finished. The quantity minus one is the quantity of arcs of the obtained skeleton.

Drawing	Initial Neurons SF = 10^{-4}	Quantity of modifications to the tree	Patterns per iteration	Quantity of final neurons
Six digit				
Without noise	12	5	17.19%	17
25% noise	11	5	18.36%	16
50% noise	10	7	18.06%	17
75% noise	5	12	24.01%	17
Hammer				
Without noise	7	0	-	7
25% noise	8	0	-	8
50% noise	4	3	42.83%	7
75% noise	4	3	45.69%	7

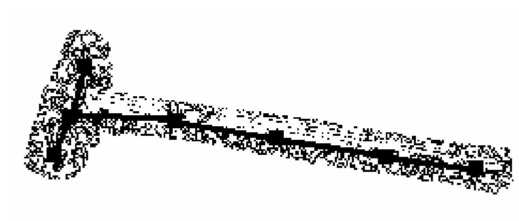
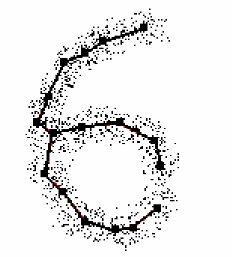
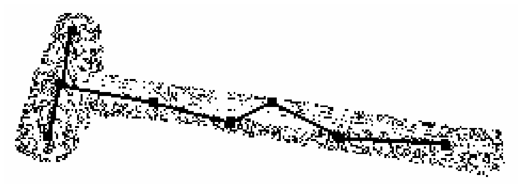
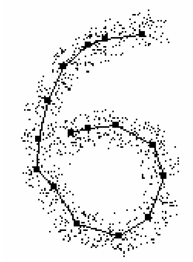
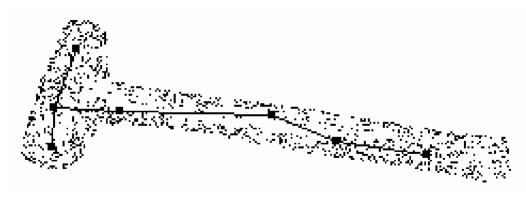
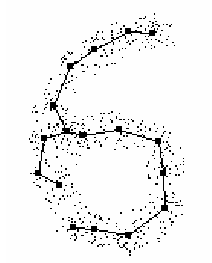
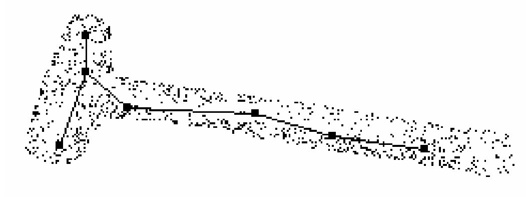
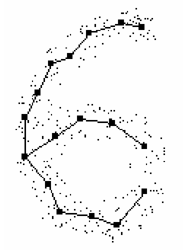
Without noise		
25% of noise		
50% of noise		
75% of noise		

Figure 6. Two sparse shapes to which three different noise levels were applied. Each case shows the skeleton obtained from the method based on AVGSOM.

The results have been successful, allowing us to obtain a proper skeleton within a reasonable time.

The election of the AVGSOM method for training the neural network has allowed us to solve to great problems. In the first place, it is not necessary to indicate the quantity of initial neurons of the network and, in the second place, its architecture eases obtaining the minimum spanning tree since the nearest neurons are connected as direct neighbors.

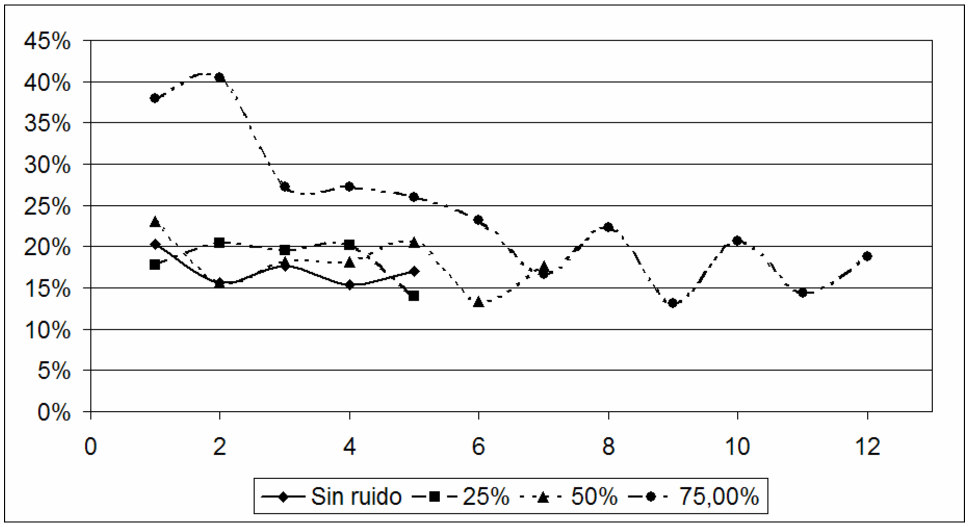


Figure 7. The graphic shows, for the six digit shape and its three variants with noise, the reduction in the percentage of patterns used at each tree modification.

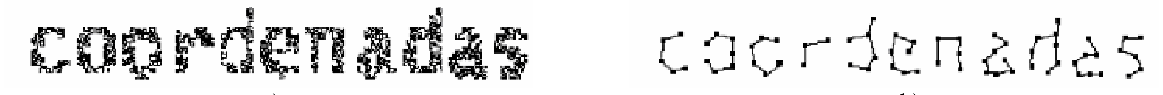


Figure 8. a) Original Image - b) Skeleton found with the proposed algorithm.

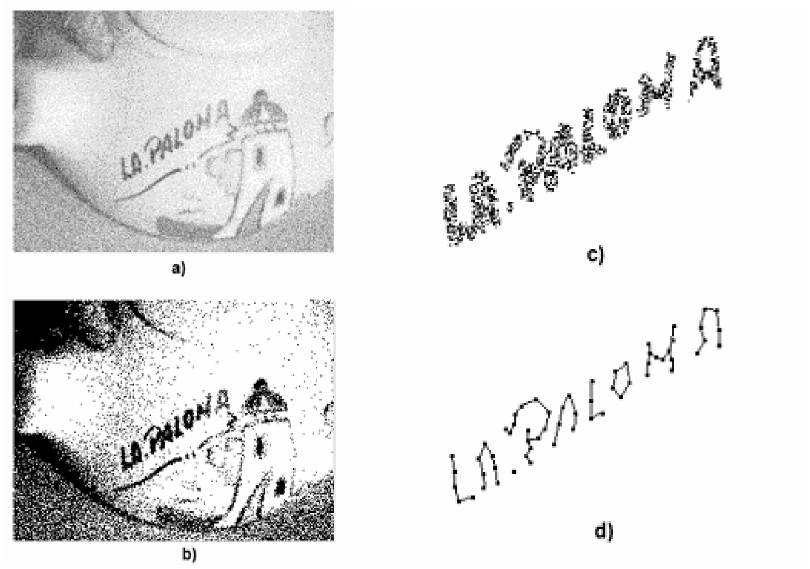


Figure 9. a) Original Image, b) Image alter a poor pre-processing, c) Extension of the area of interest, d) Skeleton found with the proposed method.

References

[1] D. Alahakoon, S.K. Halgamuge and B. Srinivasan. Dynamic Self-Organizing Maps with Controlled Growth for Knowledge Discovery. *IEEE Transactions On Neural Networks,* 11(3):601-614, 2000.

[2] G. Borgefors and S. Svensson. Fuzzy border distance transforms and their use in 2D skeletonization. *Proceedings. 16th International Conference on Pattern Recognition,* 1:11-15, pages 180-183, Aug. 2002.

[3] N.G. Bourbakis, D. Goldman, R. Fematt, I. Vlachavas and L.H. Tsoukalas. Path planning in a 2-D known space using neural networks and skeletonization. *IEEE International Conference on Systems, Man, and Cybernetics, 'Computational Cybernetics and Simulation',* 3:12-15, pages 2001-2005, Oct. 1997.

[4] Q. Dawound and E.R. Kamel. New approach for the skeletonization of handwritten characters in gray-level images. *Proceedings of the 17th International Conference on Pattern Recognition,* 2:23-26, pages 839-842, Aug. 2004.

[5] B. Fritzke. A growing neural Gas Network Learns Topologies In Gesauro, T., Touretzky, D. S. and Leen, T. K. editors, *Advances in Neural Information Processing Systems 7*, pages 625-632. MIT Press, Cambridge MA, 1995.

[6]B. Fritzke. Growing Self-organizing Networks – Why?. In: Verleysen, M. (ed.), *ESANN'96: European Symposium on Artificial Neural Networks*, pages 61-72. D-Facto Publishers, Brussels, 1996

[7] W. Hasperué and L. Lanzarini. Dynamic Self-Organizing Maps. A new strategy to upgrade topology preservation. *XXXI Congreso Latinoamericano de Informática CLEI 2005. Cali. Colombia*, pages 1081-1087. Oct. 2005.

[8] J.-H. Jang and K.-S. Hong. A pseudo-distance for the segmentation-free skeletonization of gray-scale images. *Proceedings. Eighth IEEE International Conference on Computer Vision*, 2:18-23. Jul. 2001.

[9] T. Kohonen. Self-Organizing Maps. 2nd Edition. Springer. ISSN 0720-678X. 1997.

[10] R.M. Palenichka and M.B. Zaremba. Multi-scale model-based skeletonization of object shapes using self-organizing maps. *Proceedings. 16th International Conference on Pattern Recognition*, 1:143146, Aug. 2002.

[11] H. Sasamura and T. Saito. A simple learning algorithm for growing self-organizing maps and its application to the skeletonization. *Proceedings of the International Joint Conference on Neural Networks*, 1:787-790, Jul. 2003.

[12] R. Singh, M.C. Wade and N.P. Papanikolopoulos. Letter-level shape description by skeletonization in faded documents. *In Proc. 4th IEEE Workshop Applicat. Comput. Vision*, pages 531-536, 1998.

[13] R. Singh, V. Cherkassky and N.P. Papanikolopoulos. Self-organizing maps for the skeletonization of sparse shapes. *IEEE Transactions on Neural Networks*, 11(1):241-248, 2000.

[14] Q.-Z. Ye and P.E. Danielsson. Inspection of printed circuit boards by connectivity preserving shrinking. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):737-742, 1988.

[15] S.C. Zhu and A.L. Yuille. FORMS: a flexible object recognition and modelling system. *Int. J. Comput. Vision*, 20(3):187-212, 1996.