Thinning Algorithms

Diego Enciso[1], Juan Galvez[2], Luis Jáuregui[3], Macarena Oyague[4] and Miguel Yurivilca[5]

***Abstract*—In this paper we'll be making comparisons between two Thinning Algorithms: Zhang-Suen Algorithm and Guo-Hall Algorithm. Throughout this paper, the algorithms will be analyzed empirically and theoretically. Finally the results obtained in each one will be run in two different CPUs with different architectures and operating systems to evaluate how they behave.**

I. Introduction

Currently, the images can be processed in such a way that a transformation is made in order to obtain an expected result, and thus meet certain requirements that the user wants or needs. This is done by means of specific algorithms for each case. Different examples of image transformation can be elongation, color change, image compression, eliminating pixels, smoothing effect, among others. This paper focus on the thinning method of skeletonization.

Skeletonization provides an effective and compact representation of an image object by reducing its dimensionality to a "medial axis" or "skeleton" while preserving the topologic and geometric properties of the object [1]. The process can be viewed as a procedure to transform the width of a binary pattern into just one single pixel. Essentially, such transformation can be achieved by successively removing points or layers of outline from a binary pattern until all the lines or curves are of unit width. The resulting set of lines or curves is called the skeleton of the pattern [2]. However, one of the problems that is frequently encountered is the presence of surfaces or small barbules (especially of clusters of voxels at the level of junctions) in a skeleton that we would prefer to be thin [3].

In the next sections, two 2d skeletonization thinning algorithms will be analysed: Zhang-Suen Algorithm and Guo-Hall Algorithm.

Thinning Algorithm

The thinning method transforms an input binary image into a skeleton by reducing the original image which contains different thicknesses to a thin representation (a set of curves and lines). It is based on a digital simulation of the fire front propagation: the border points of a binary object that satisfy certain topological and geometric constraints are deleted in iteration steps. The entire process is then repeated until only the skeleton is left [4][5].

This image transformation algorithm is commonly known for having applications that helps different medicine and research areas, but for the 3-dimensional domain. Below are some examples:

1, 2, 3, 4 *and* 5 are students of Computer Science at the University of Engineering and Technology, Lima - Perú. This is a Paper for the subject of Analysis and Design of Algorithms.

Medicine

This algorithm is widely used in medical imaging applications including volumetric, structural, and topological analysis, object representation, stenosis detection, pathfinding, etc [6]. A common objective of medical imaging is to extract information of internal human organ or tissue through in vivo or ex vivo imaging. Often, these images are processed through complex cascades of processing and analysis steps. Skeletonization is a transformation process that reduces a volumetric object into a significantly reduced, simplified and compact representation, referred to as "medial axis" or "skeleton" [6]. For example, a clinical application in neurosurgery is the symbolic description of vascular trees derived from multimodal three dimensional images. It is made by a 3D skeletonization method, which is adapted to tubular forms and is advisable for that symbolic description [7].

Biology

A simplified skeleton of a plant is helpful for the structural analysis and understanding of plants of interest and the measurements of their traits such as the areas, perimeters of leaves, curvatures, and the lengths between different nodes. That is the reason why simplify the complicated 3D structure of the plant point cloud data into 1D curved skeleton is crucial for the plant phenotyping. With skeletonization, the extraction of the skeleton voxels, and find the nearest neighbors to connect the skeleton points as a connected representation can be made [8].

II. Related Works

Skeletonization by thinning is the only technique to be analyzed in the paper. Skeletonization by general fields, by geometric approaches and by distance are other methods that have the same goal as the method of the paper. However, there are more techniques that transform images, including Distance transform, Poisson Disk Sampling and Convex Hull. These will be explained briefly below.

A. Skeletonization by general fields

In general potential field function, the potential at some point interior to the object is determined as a sum of potentials generated by the boundary of the object. In the discrete case, the boundary voxels are considered point charges generating the potential field. The curve skeleton is extracted by detecting the local extremes of the field and connecting them [9].

B. Skeletonization by geometric approaches

Geometric methods are generally applied to polygons and polyhedra. They are based on the Voronoi diagram, which represents a subdivision of the space into regions that are

closer to a generator element (a mesh vertex in the case of a 3D model) than to any other such element. The internal edges and faces of the Voronoi diagram can be used to extract the skeleton of the shape [9][10].

C. Skeletonization by distance

Skeletonization by distance assigns a value to each pixel within the shape of a figure equivalent to the distance of that pixel to the nearest point on the border. The symmetric axis of the figure corresponds to the 'local maxima' of the distance transform [11].

D. Distance transform

Distance transforms are an important tool in computer vision, image processing and pattern recognition. A distance transform of a binary image specifies the distance from each pixel to the nearest non-zero pixel. This technique plays a central role in the comparison of binary images, particularly for images resulting from local feature detection techniques such as edge or corner detection [12].

E. Poisson Disk Sampling

Poisson Disk Sampling is a random process for selecting points from a subdomain of a metric space. A selected point must be disk-free, at least a minimum distance, 'r', from any previously selected point. Thus each point has an associated disk of radious 'r' that precludes the bution. With this method, a bias-free dart is thrown and accepted if it does not violates the minimum distance bound with previous successful darts, other wise it is rejected. As more darts make it to the domain, the remaining parts of the domain valid for the sampling process ("voids") get smaller, decreasing the probability of being hit [13].

F. Convex Hull

Convex hull for a determined set (use set 'S' as an example) can also be defined as the set of points that can be expressed as convex combinations of the points in that set 'S'. It is different for each type of objects because it depends upon the feature point of every object. If the convex hull cannot be constructed from the feature points specified,the value will decrease by 1 from the total number of points. This process (decrease by 1) will continue to happen until the appropriate convex hull can be made. Finally it can be defined for object of any kind with any number of dimensions [14].

III. Algorithms Description

Zhang-Suen and Guo-Hall thinning algorithms will be theoretically analyzed.

Zhang-Suen thinning algorithm

Consists of two sub iterations:

- The first sub iteration aims to delete the south-east boundary points and the north-west corner points.
- The second sub iteration aims to delete the north-west boundary points and the south-east corner points.

End points and pixel connectivity are preserved. Each pattern is thinned down to a "skeleton" of unitary thickness. Experimental results show that this method is very effective [15].

In the first sub iteration, the pixel P_1 is deleted if it satisfies the following conditions:

(a) $2 \leq B(P_1) \leq 6$
(b) $A(P_1) = 1$
(c) $P_2 * P_4 * P_6 = 0$
(d) $P_4 * P_6 * P_8 = 0$

where $A(P_1)$ is the number of 01 patterns in the sequence P_2, P_3, $\ldots$, P_8, P_9, P_2 that are the neighbours of P_1 (Fig. 1) and $B(P_1)$ is the number of nonzero neighbours of P_1.

P_9	P_2	P_3
P_8	P_1	P_4
P_7	P_6	P_5

Fig. 1. 3×3 window of 9 pixels

In the second sub iteration, only conditions (c) and (d) are changed as follows:

(c') $P_2 * P_4 * P_8 = 0$
(d') $P_2 * P_6 * P_8 = 0$

Guo-Hall thinning algorithm

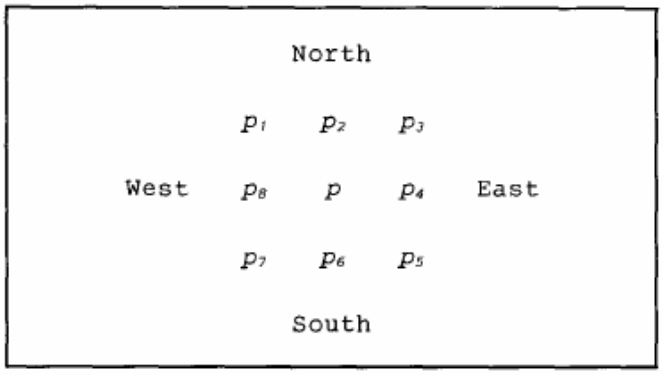


Fig. 2. Neighborhood definitions for pixel p [16]

This algorithm uses operators with a 3x3 support as defined in Fig 2. p2, p4, p6 and p8 as p's side neighbors and p1, p3, p5 and p7 as p's diagonal neighbors. There are several defined variables to be used in this algorithm. C(p) is the number of distinct 8-connected components of ones in p's 8-neighborhood. C(p) = 1 implies p is 8-simple when p is a boundary pixel [16]. B(p) is the number of ones in p's 8-neighborhood. N(p) is a useful variable for endpoint detection, but which can also help to achieve thinner results.

```
C(p1)  = !p2 & (p3 | p4) + !p4 & (p5 | p6) +
          !p6 & (p7 | p8) + !p8 & (p1 | p2)

N1(p1) = (p9 | p2) + (p3 | p4) +
         (p5 | p6) + (p7 | p8)
```

```
N2(p1) = (p2 | p3) + (p4 | p5) +
         (p6 | p7) + (p8 | p9)

N(p1)  = MIN[N1(p1), N2(p1)]
```

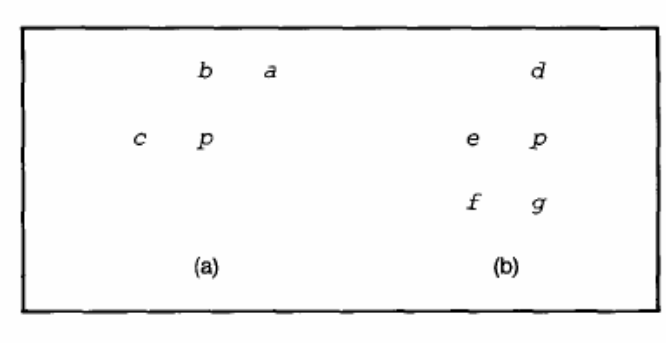


Fig. 3. Objects illustrating the rationale for conditions used [16]

A pixel p1 is deleted if all of the conditions are satisfied. C(p1) condition is necessary for preserving local connectivity when p1 is deleted and avoids deletion of pixels in the middle of curves. C(p1) takes a value over the range 0 to 4 as it counts the total of occurrences of a side zero with a one in at least one of the two adjacent pixels in the clockwise direction around p1's 8-neighborhood [16]. The variable N(p1) provides an endpoint check replacing B(p1). When B(p1) = 1, p1 is an obvious endpoint and N(p1) = 1. But when B(p1) = 2, p1 may or may not be and endpoint. In fig 3(a) a count of neighboring ones will not discriminate between the redundant midpoint pixel p1 and the endpoints a or c. Thus, the redundant pixel p1 is deleted in odd iterations while endpoints a and c are preserved. In general N(p1) allows endpoints to be preserved while deleting many redundant pixels in the middle of curves [16].

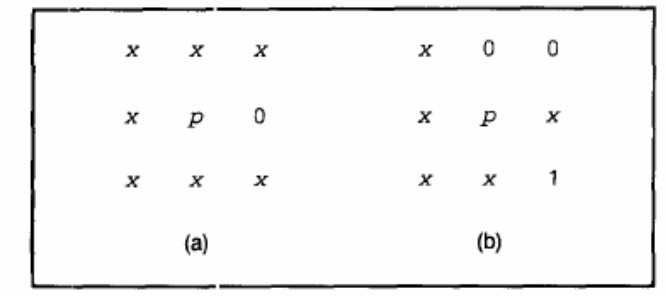


Fig. 4. Neighborhood conditions required for deletion of p1 in odd iterations (x represents a pixel which may be either 0 or 1) [16]

Condition c(1): (p2 or p3 or !p5) and p4 = 0, used for odd iterations, is satisfied when p1's neighborhood takes either of the forms shown in Fig 4, where x refers to a "don't care" pixel (either zero or one is acceptable). Condition c(2): (p6 or p7 or !p9) and p8 = 0 is satisfied for 180° rotations of either of the two conditions shown in Fig 4. Thus, c(1) tends to identify pixels at the north and east boundary of objects and c(2) identifies pixels at the south and west boundary of objects. Thinning of an object proceeds by successively removing in distinct iterations north and east, and then south and west boundary pixels. The connectivity properties of the image are preserved [16].

IV. Theoretical Analysis

Zhang-Suen thinning algorithm

```
Verify conditions (A)
While points are deleted do
    For all pixels p(i,j) = P1 do
        if 2 <= B(P1) <= 6 and A(P1) = 1
                P2 x P4 x P6 = 0...(N1.1)
                P4 x P6 x P8 = 0...(N1.2)
        then
            delete pixel p(i,j)
    For all pixels p(i,j) = P1 do
        if 2 <= B(Pi) <= 6 and A(p1) = 1
                P2 x P4 x P8 = 0...(N2.1)
                P2 x P6 x P8 = 0...(N2.2)
        then
            delete pixel p(i,j)
```

A(P1) is the number of 01 patterns in the ordered set P2, P3, P4, ... P8, P9 that are the eight neighbors of P1 (Figure 1), and B(Pi) is the number of nonzero neighbors of P1, that is,

Each sub-iteration traverse the image by its columns and rows, so the complexity in time scales to $\theta(rows * cols)$. The conjunction of 2 sub-iterations is called while pixels are deleted, so we need to calculate the probability of the conditions inside each sub-iteration are true. To do so, we analyze each one separately to then evaluate the last truth operation.

- **B(P1)**: Analyzes the amount of neighbours that are a black pixel. To calculate this, we apply the binomial distribution with n = 8 and p = 0.5, since there are 8 neighbours and the chance of each one being a black pixel is $\frac{1}{2}$. Having x between 2 and 6 results in 0.856 of chance.
- **A(P1)**: Verifies that the amount of transitions from 0 to 1 between contiguous neighbours is at most 1. To do this, we apply a permutation in the order of $\frac{84}{256} = 0.328$.
- **N1.1 and N1.2 when odd, N2.1 and N2.2 when even**: these conditions both are calculated with binomial distribution, with n = 3 and p = 0.5. Having x with at least one 0 results in 0.965 of chance.

Finally, we evaluate the condition to erase a pixel multiplying these 3 previous operands, $0.856 * 0.328 * 0.965 * 0.965 = 0.261$.

V. Time Complexity

Theoretically, in both algorithms, in the worst case scenario we erase 1 pixel in each iteration and doing, in fact, rows*cols iterations; each one calling the conjunction of the sub-iterations once, ending up being $\mathcal{O}(rows^2 * cols^2) \approx \mathcal{O}(n^4)$. Theoretically as well, in the best case scenario, the loop does 1 iteration, calling our sub-iterations just once with a time

complexity of $\Omega(rows * cols) \approx \Omega(n^2)$. (Approximations are made considering we have a quadratic n*n picture).

Guo-Hall thinning algorithm

```
Thinning_Iteration
While points are deleted do
    For all pixels p(i,j) do
        if  C(p1) = 1 and  2 <= N(p1) <= 3
                (p2 | p3 | !p5) & p4 = 0
                in odd iterations
                (p6 | p7 | !p9) & p8 = 0
                in even iterations
        then
            delete pixel p(i,j)
        end if
    end for
end while
```

```
Where:
C(p1)   = !p2 & (p3 | p4) + !p4 & (p5 | p6) +
          !p6 & (p7 | p8) + !p8 & (p1 | p2)

N1(p1) = (p9 | p2) + (p3 | p4) +
         (p5 | p6) + (p7 | p8)

N2(p1) = (p2 | p3) + (p4 | p5) +
         (p6 | p7) + (p8 | p9)

N(p1)   = MIN[N1(p1), N2(p1)]
```

VI. Experiments and Results

The algorithms were tested using the same images in both cases, cow and duck images. The results are in the chart below:

CPU 1	Cow	Duck
Zhang-Suen	5.05005 sec	24.4098 sec
Gou-Hall	2.40053 sec	16.4013 sec

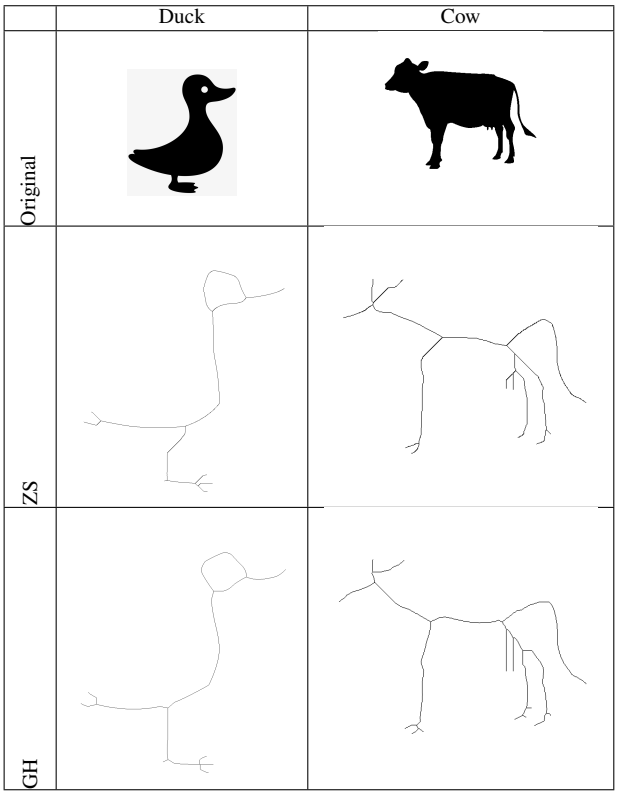
CPU 2	Cow	Duck
Zhang-Suen	12.3745 sec	54.2858 sec
Gou-Hall	2.5839 sec	15.8636 sec

Considering CPU1: Intel i9 series, MacOS. CPU2: Ryzen 7, Arch Linux OS.
As seen in the chart above, in both cases Gou-Hall is faster than Zhang-Suen.

VII. Conclusions

In this project two thinning algorithms were analyzed empirically and theoretically. It must be taken into consideration that both algorithms are quite similar (so there shouldn't be that much difference in the time complexity) because they iterate through an image (matrix of pixels) looking for some certain conditions to be true; if those conditions are true then the algorithm proceeds to "delete" the pixel. In the theoretical analysis the worst case scenario was analyzed, the result is that both have the same performance (time complexity). In the empirical analysis the tests were carried out on two different machines to reinforce or depreciate the results. Those results weren't as expected because in all the tests Gou-Hall had better performance than Zhang-Suen approach.

Despite the fact that, theoretically, both had the same time complexity, in the empirical analysis Gou-Hall had lower time



complexity so it can be concluded that for those tests Gou-Hall is better than Zhang-Suen.

References

[1] Saha, P., Borgefores, G., Sanniti, G. (2016). A survey on skeletonization algorithms and their applications. *Pattern Recognition Letters, 76,* pp, 3-12.

[2] TQ. Yan, CX. Zhou, A Continuous Skeletonization Method Based on Distance Transform. In: Huang DS., P. Gupta, X. Zhang, P. Premaratne (eds) Emerging Intelligent Computing Technology and Applications. ICIC 2012. Communications in Computer and Information Science, vol 304. Springer, Berlin, Heidelberg. 2012. https://doi.org/10.1007/978-3-642-31837-5_37

[3] M. Näf, G. Székely, R. Kikinis, M. Shenton, O. Kübler, 3D Voronoi Skeletons and Their Usage for the Characterization and Recognition of 3D Organ Shape. *Computer Vision, Graphics and Image Processing, 66*(2), 1997, pp 147-161.

[4] A. Sider, L. Ben Boudaoud and A. A Kamel Tari, A new thinning algorithm for binary images. 2015.

[5] K. Palágyi, A 3D fully parallel surface-thinning algorithm. *Theoretical Computer Science, 406*(1-2), 2008, pp 119-135.

[6] D. Jin, P.K. Sara, A New Fuzzy Skeletonization Algorithm and Its Applications to Medical Imaging. In: A. Petrosino (eds) Image Analysis and Processing – ICIAP 2013. ICIAP 2013. Lecture Notes in Computer Science, vol 8156. Springer, Berlin, Heidelberg. 2013. https://doi.org/10.1007/978-3-642-41181-6_67.

[7] L. Verscheure et al., Three-dimensional skeletonization and symbolic description in vascular imaging: preliminary results. 2015.

[8] B. Ramamurthy, J. Doonan, J. Zhou, J. Han and Y. Liu, Skeletonization of 3D plant point cloud using a voxel base thinning algorithm. 2015.

[9] N. Cornea, D. Silver and P. Min, Curve-Skeleton Properties, Applications and Algorithms. 2005.

[10] D. Ayala, M. Vigo, J. Martinez and N. Pla-Garcia, Skeleton Computation of an Image Using a Geometric Approach. 2010.

[11] M. Wright, R. Cipolla, P.J. Giblin, Skeletonisation using an extended Euclidean distance transform. BMVA Press, 1994.

[12] P. Felzenszwalb and D. Huttenlocher, Distance Transforms of Sampled Functions.

[13] Maximal Poisson-Disk Sampling with Finite Precision and Linear Complexity in Fixed Dimensions.
[14] M. Sharif, A new Approach to Compute Convex Hull. 2011.
[15] T. Y. Zhang and C. Y. Suen, A fast parallel algorithm for thinning digital patterns. Commun. ACM 1984. 236–239.
[16] Z. Guo, R. Hall. Parallel Thinning with two subiteration algorithm. Communications of the ACM 1989.