Skeletonization of Binary Images

Filippo Momesso

January 27, 2022

University of Trento
filippo.momesso@studenti.unitn.it

Overview

Basic definitions

Zhang-Suen Parallalel Thinning

Morphological Thinning

Lee's 3D Skeletonization

Wrap-up

Basic definitions

Skeletonization

An image processing technique which reduces a binary object (or region) to a 1 pixel wide representation called **skeleton**.

Useful in many application fields such as *shape recognition and analysis*, *animation*, *motion tracking* or *medical imaging* [3].

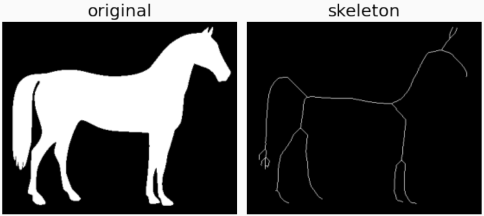


Figure 1: Example of a skeletonized image.

What is Skeletonization?

Skeleton

The ideal skeleton should:

- be a **connected subset** of points from the original region,
- represent the **geometric** characteristics of the region (e.g. area, curvature...),
- preserve the **topological** characteristics of the region (e.g. connectivity, holes, cavities...)

Three major skeletonization techniques:

- Medial-axis distance transform
- Non-pixel-based methods (computes analytically the skeleton)
- Thinning methods

We will focus on **thinning methods** since they are quite efficient and commonly used in state-of-the-art applications.

Zhang-Suen Parallalel Thinning

Zhang-Suen algorithm [4] is a fast parallel algorithm which takes a binary 2D image and removes pixels from the object's border by making successive iterations until convergence.

2D Binary image

A matrix M where each pixel $M[i][j]$ is either 1 or 0. A **region** in an image is a connected set of 1-valued pixels.

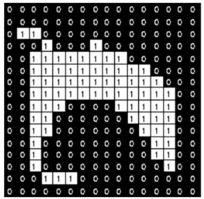


Figure 2: In white a region.

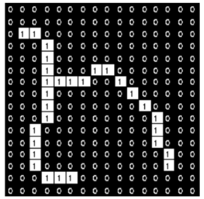


Figure 3: The skeleton of the region on the left.

Zhang-Suen Parallalel Thinning

- Given an element $P_1 = M[i][j]$, its neighbours are:

$P_9 = M[i-1][j-1]$	$P_2 = M[i-1][j]$	$P_3 = M[i-1][j+1]$
$P_8 = M[i][j-1]$	$P_1 = M[i][j]$	$P_4 = M[i][j+1]$
$P_7 = M[i+1][j-1]$	$P_6 = M[i+1][j]$	$P_5 = M[i+1][j+1]$

- The algorithm iteratively removes all countour points (change the value from 1 to 0) which satisfy some conditions on their 8 neighbours.

- The new value of a pixel at the n-th iteration is based on the values of its neighbours at the $n-1$-th iteration. This allows all pixel to be processed in **parallel** at each iteration.

Pixel Deletion Conditions

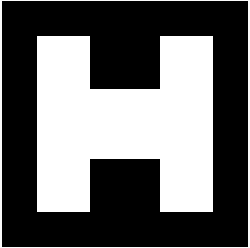
- An iteration (full pass over all the pixels) is divided in two sub-iterations.

- **Sub-iteration 1**: P_1 is deleted if
  1. $2 \leq B(P_1) \leq 6$
  2. $A(P_1) = 1$
  3. $P_2 * P_4 * P_6 = 0$
  4. $P_4 * P_6 * P_8 = 0$

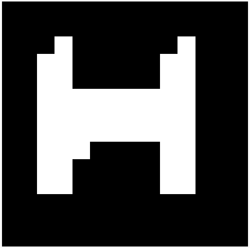
- **Sub-iteration 2**: P_1 is deleted if
  1. Conditions 1 and 2 are true
  2. $P_2 * P_4 * P_8 = 0$
  3. $P_2 * P_6 * P_8 = 0$

- $B(P_1)$ is the number of **1-value neighbours** of P_1. Condition 1 is needed to **preserve an endpoint** of the skeleton.
- $A(P_1)$ is the number of **0-1 patterns** in the ordered sequence of neighbours $P_2, P_3, \ldots, P_9, P_2$. Condition 2 is needed to preserve **connectivity**, i.e. not split the skeleton in two.

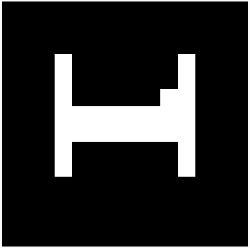
Pixel Deletion Conditions



(a) Original image.



(b) After sub-iteration 1.



(c) After sub-iteration 2.

Figure 4: Effects of one single iteration on an example image.

Thanks to conditions 3 and 4:

- Sub-iteration 1 removes East and South boundary pixels and North-West corner pixels.
- Sub-iteration 2 removes West and North boundary pixels and South-East corner pixels.

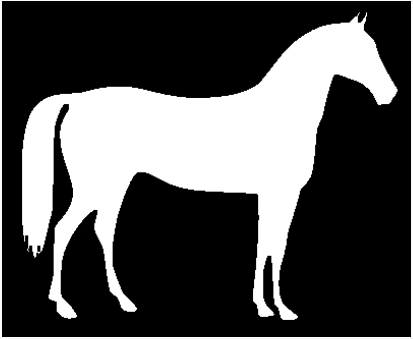


Figure 5: Animation of Zhang-Suen algorithm removing contour pixels until only the skeleton remains.

Zhang-Suen Pseudocode

Algorithm 1 Zhang-Suen Thinning Algorithm

function *zhang-skeletonization*(*image*) **return** *skeletonized image*
    skeleton $\leftarrow$ *image* with a padding of zeros ▷ state at the $(n-1)$-iteration
    cleanedSkeleton $\leftarrow$ *image* with a padding of zeros ▷ state at the (n)-iteration
    pixelRemoved $\leftarrow$ *True*
    while *pixelRemoved* **do** ▷ iterate until no more pixels are removed
        pixelRemoved $\leftarrow$ *False*
        for *iter* $\leftarrow 1$ **to** 2 **do** ▷ sub-iterations
            for $i \leftarrow 1$ **to** $rows - 1$ **do**
                for $j \leftarrow 1$ **to** $cols - 1$ **do** ▷ iterate over each pixel
                    if $skeleton[i][j] = 1$ **then**
                        if $2 \leq nonZeroNeighbours \leq 6$ AND ▷ condition 1
                            $zeroOnePatterns = 1$ **then** ▷ condition 2
                            if $iter = 1$ AND
                                $P_2 * P_4 * P_6 = 0$ AND $P_4 * P_6 * P_8 = 0$ **then** ▷ condition 3, 4
                                $cleanedSkeleton[i][j] \leftarrow 0$
                                pixelRemoved $\leftarrow$ *True*
                            else if $iter = 2$ AND
                                $P_2 * P_4 * P_8 = 0$ AND $P_2 * P_6 * P_8 = 0$ **then** ▷ condition 3, 4
                                $cleanedSkeleton[i][j] \leftarrow 0$
                                pixelRemoved $\leftarrow$ *True*
            skeleton $\leftarrow$ *cleanedSkeleton*
    return *skeleton* without padding of zeros

Morphological Thinning

Morphological Thinning

Morphological Thinning [1] is a morphological operation that removes contour pixels from a region of a binary image. It relies on the **Hit-or-Miss transform** operation.

Hit-or-Miss transform

General binary morphological operation used to look for the presence of specific patterns of *foreground* and *background pixels* (1s or 0s respectively).

- Hit-or-Miss transform uses a *structuring element* or *kernel* to look for those patterns.
- Tipycally it is used a 3×3 kernel. It can contain both 1s and 0s and a special "I don't care" value.
- Iteration over all pixels, comparing the kernel with the underlying sub-image.
- $\mathrm{M[i][\,j\,]} = \begin{cases} 1 & \text{if } \mathit{kernel} = \mathrm{M[i-1{:}i+1][j-1{:}j+1]} \\ 0 & \mathit{otherwise} \end{cases}$

	0	0
1	1	0
	1	

Figure 6: Example of kernel. Cells in blank are "I don't care" valued.

Hit-or-Miss Transform

Thinning operation

Given a binary image I and a kernel K:

$$thin(I, K) = I - \textit{hit-or-miss}(I, K) \tag{1}$$

where the subtraction is the logical subtraction $X - Y = X \cap \neg Y$

- A pixel (i, j) is deleted (i.e. set to 0) if kernel and sub-image **do not** exacly match, otherwise is left unchanged.
- To obtain a skeleton of the image, $thin(I, K)$ should be repeated until no change occurs.
- The choice of the kernel determines which pixels are deleted from the region.

Skeletonization Kernels

- Recalling Slide 4 a skeleton should preserve topological characteristics of the region such as connectivity, holes, cavities etc.
- $thin(I, K)$ deletes pixels based on the kernel K.
- The two kernels below (and all their 90° rotations) allow to delete only pixels whose deletion preserves the above mentioned characteristics.
- In each iteration $thin(I, K)$ is executed for each of the 8 resulting kernels.

0	0	0
1	1	1
	1	

(a) Detects deletable border pixels.

	0	0
1	1	0
	1	

(b) Detects deletable corner pixels.

Figure 7: Morphological thinning kernels (and all their 90° rotations)

Spurs Problem

Spurs

Skeletons obtained by $thin(I, K)$ with kernels presented in the previous slide can present **short spurs** produced by irregularities in the boundary of the region as shown below.

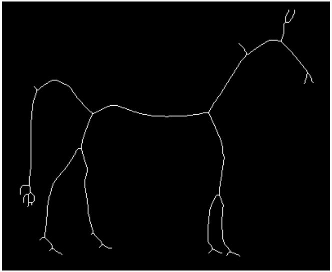


Figure 8: Morphological thinning skeleton with spurs.

- Spurs can be removed by a process called **pruning**.
- Pruning is performed by applying $thin(I, K)$ with ad-hoc kernels for a **fixed amount** of iterations.
- Pruning until convergence would actually remove all pixels exept those which form closed loops.

Kernels for Spur Removal

By using the kernels (and all their 90° rotations) shown below it is possible to remove spurs from skeleton.

0	0	0
0	1	0
		0

0	0	0
0	1	0
0		

Figure 9: Spur removal kernels.

After removing spurs for 10 iterations we get the following skeleton.

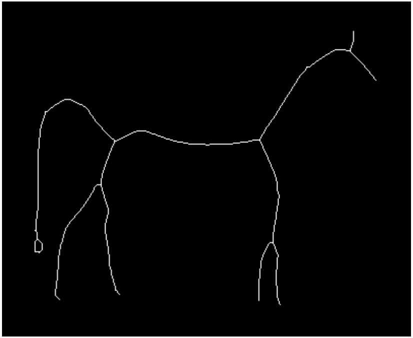


Figure 10: Skeleton after spur removal.

Morphological Thinning Pseudocode

Algorithm 2 Morphological Thinning Algorithm

```
function morphological-thinning(image) return skeletonized image
    skeleton ← image with a padding of zeros                    ▷ state at the (n − 1)-iteration
    cleanedSkeleton ← image with a padding of zeros             ▷ state at the (n)-iteration
    thinningKernel ← array of kernels for thinning operation
    spurKernels ← array of kernels for spur removal operation
    pixelRemoved ← True
    while pixelRemoved do                        ▷ iterate until no more pixels are removed
        pixelRemoved ← False
        cleanedSkeleton ← thin(cleanedSkeleton, thinningKernels)
        if cleanedSkeleton ≠ skeleton then
            skeleton ← cleanedSkeleton, pixelRemoved ← True
    for i ← 1 to pruningSteps do                                             ▷ pruning loop
        cleanedSkeleton ← thin(cleanedSkeleton, spurKernels)
        if cleanedSkeleton = skeleton then break
        skeleton ← cleanedSkeleton
    return skeleton without padding of zeros

function thin(image, kernels)
    for kernel in kernels do
        out ← hit-or-miss(image, kernel)
        image ← image − out
    return image
```

Lee's 3D Skeletonization

3D Skeleton

In 3D Euclidean space the skeleton of an object is the **locus of the centers** of all inscribed maximal spheres, where the spheres touch the boundary **at more than one point**.

A skeleton can be defined by *medial axes* or *medial surfaces*. We will focus on medial axes.

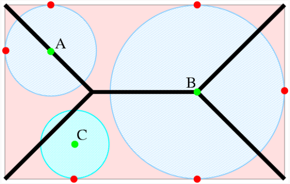


Figure 11: 2D representation of the above definition.

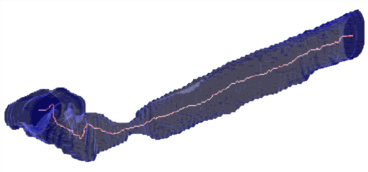


Figure 12: In white the medial axes skeleton of a 3D tubular object.

Topological properties

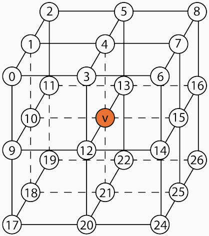
- A 3D binary image is a 3D binary matrix of size $k_{max} \times j_{max} \times i_{max}$. Every pixel v is represented by its coordinates (k, j, i) and has the value 1 or 0.
- When we talk about 3D geometries we can describe them in terms of their topological properties such as the number of *connected objects*, *cavities* and *holes*.
- The **Euler Characteristic** χ is a compact way combine those characteristics in a number that describes the geometry.

$$\chi(S) = O(S) - H(S) + C(S) \tag{2}$$

  where $O(S)$, $H(S)$ and $C(S)$ are the numbers of connected objects, holes and cavities of S, the set of all the points with the value 1.
- By using a local formula $G(S)$ for the Euler Characteristic, the complexity can be reduced by calculating $\chi(S)$ considering pixel by pixel neighbourhoods.

Neighborhoods

- Considering a pixel v, its 26-neighbours $N(v)$ are defined based on the cube below.



n	$8\,\delta G_6$	$8\,\delta G_{26}$	n	$8\,\delta G_6$	$8\,\delta G_{26}$	n	$8\,\delta G_6$	$8\,\delta G_{26}$	n	$8\,\delta G_6$	$8\,\delta G_{26}$	n	$8\,\delta G_6$	$8\,\delta G_{26}$
1	−1	1	53	3	1	105	−1	5	157	3	1	209	1	−1
3	1	−1	55	1	−1	107	1	3	159	1	−1	211	3	1
5	1	−1	57	1	3	109	1	3	161	−1	−3	213	−1	1
7	3	1	59	−1	1	111	−1	1	163	1	−1	215	1	−1
9	−1	−3	61	3	1	113	1	−1	165	1	3	217	1	3
11	1	−1	63	−3	−1	115	−1	1	167	3	1	219	3	1
13	1	−1	65	−1	−3	117	−1	1	169	−1	1	221	−1	1
15	−1	1	67	1	3	119	−3	−1	171	1	−1	223	−3	−1
17	1	−1	69	1	−1	121	1	3	173	1	3	225	−1	1
19	3	1	71	3	1	123	−1	1	175	−1	1	227	1	3
21	3	1	73	−1	1	125	−1	1	177	1	−1	229	1	3
23	5	−1	75	1	3	127	−7	−1	179	−1	1	231	3	1
25	1	3	77	1	−1	129	−1	−7	181	3	1	233	−1	5
27	3	1	79	−1	1	131	1	−1	183	1	−1	235	1	3
29	3	1	81	1	−1	133	1	−1	185	1	3	237	1	3
31	1	−1	83	3	1	135	3	1	187	−1	1	239	−1	1
33	−1	−3	85	−1	1	137	−1	−3	189	3	1	241	1	−1
35	1	−1	87	1	−1	139	1	−1	191	−3	−1	243	−1	1
37	1	3	89	1	3	141	1	−1	193	−1	−3	245	−1	1
39	3	1	91	3	1	143	−1	1	195	1	3	247	−3	−1
41	−1	1	93	−1	1	145	1	−1	197	1	−1	249	1	3
43	1	−1	95	−3	−1	147	3	1	199	3	1	251	−1	1
45	1	3	97	−1	1	149	3	1	201	−1	1	253	−1	1
47	−1	1	99	1	3	151	5	−1	203	1	3	255	1	−1
49	1	−1	101	1	3	153	1	3	205	1	−1			
51	−1	1	103	3	1	155	3	1	207	−1	1			

- This cube can be divided into eight overlapping 2×2 octants $N^2(v)$.
- Each octant have an Euler Characteristic value $G(N^2(v))$.
- Summing up each octant's $G(N^2(v))$ we obtain the **local Euler Characteristic** value for the pixel v.
- Since there are only 256 possible octant pixel configurations, we can store a **look-up table** to speed up computation.

Thinning

Removing pixels

A pixel can be removed from the image (set to 0) if:

- is **not an endpoint** (endpoint if it has exactly one 1-valued neighbour in the 26-neighborhood).
- is **Euler-Invariant** (if removed the Euler Characteristic does not change).
- its removal **does not disconnect** the object.

A pixel with such characteristics is called **simple point**.

Simple point detection is a crucial step in Lee's 3D Skeletonization [2] algorithm.

- Euler-invariance can be checked efficiently using the Euler Table mentioned in Slide 20 on the preceding page.
- Connectivity can be determined using an **octree** data structure for representing the neighborhood $N(v)$.

Connectivity checking

We can use a recursive procedure N(v)_labeling to determine the number of connected objects in the $N(v)$ neighborhood if pixel v is removed.

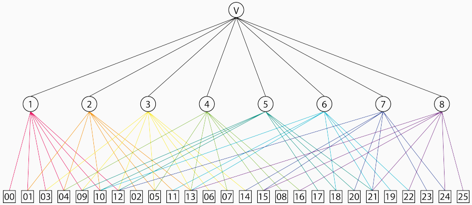


Figure 13: Octree data structure for pixel v and its 26-neighbohood.

- The procedure recursively assigns labels to each pixel in the 26-neighborhood.
- All connected pixels have the same label, i.e. each label represent a connected object.
- If more than one different label is assigned, there is more than one connected object.

Lee's 3D Skeletonization Algorithm

3D Skeletonization Algorithm

A thinning iteration is composed of the following phases:

1. For each one of the six directions (N, S, W, E, U, B):
   1.1 Find a list of simple points candidates satisfying:
      - Belongs to a border in the direction we're iterating on
      - Not an end point
      - Euler-Invariant
      - Preserve connectivity (through N(v)_labeling)

   1.2 For each simple point candidate:
      1.2.1 Re-check if it preserve connectivity
      1.2.2 if so delete it

To apply Lee's algorithm on 2D images:

- add the third dimension by applying a padding of 0s;
- instead of iterating on the six directions, check only the four N, S, W, E.

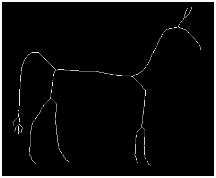


Figure 14: Lee's skeleton of our horse.

Wrap-up

What we've learned in this lecture

- Skeletons are useful for many image-based applications.
- There exist many different methods for extract a skeleton for both 2D and 3D binary images.
- Zhang-Suen is a fast and simple parallel thinning method which **iteratively removes pixels** from the boundaries of a region satisfying certain **conditions**.
- Morphological thinning iteratively remove pixels exploiting the **hit-or-miss transform** and some **ad-hoc kernels**.
- In morphological thinning boundary irregularities can produce a skeleton with short **spurs** that can be removed through **pruning**.
- Lee's 3D skeletonization method is a complex thinning algorithm which exploits **topological properties** of 3D objects.

References

[1] R. Gonzalez and R. Woods. "Digital Image Processing". In: *Addison-Wesley Publishing Company* (1992), pp. 518–548.

[2] T.C. Lee, R.L. Kashyap, and C.N. Chu. "Building Skeleton Models via 3-D Medial Surface Axis Thinning Algorithms". In: *CVGIP: Graphical Models and Image Processing* 56.6 (1994), pp. 462–478. issn: 1049-9652. doi: https://doi.org/10.1006/cgip.1994.1042. url: https://www.sciencedirect.com/science/article/pii/S104996528471042X.

[3] Punam K. Saha, Gunilla Borgefors, and Gabriella Sanniti di Baja. "A survey on skeletonization algorithms and their applications". In: *Pattern Recognition Letters* 76 (2016). Special Issue on Skeletonization and its Application, pp. 3–12. issn: 0167-8655. doi: https://doi.org/10.1016/j.patrec.2015.04.006. url: https://www.sciencedirect.com/science/article/pii/S0167865515001233.

[4] T. Y. Zhang and C. Y. Suen. "A Fast Parallel Algorithm for Thinning Digital Patterns". In: *Commun. ACM* 27.3 (Mar. 1984), pp. 236–239. issn: 0001-0782. doi: 10.1145/357994.358023. url: https://doi.org/10.1145/357994.358023.

Thank you for your attention.