Skeletonization (part 1)

Jason Rupard

CAP6400

Skeletonization Topic

- Introduction
- Medial-Axis Transform (MAT)
- Thinning
- Other Concepts

Introduction

- What is a skeleton?
  - Opinion

- Do we know it?

- Skeleton
  - Represents the shape of an object in a small amount of pixels

Introduction

- Can convey all information found in the original object
  - Position
  - Size
  - Orientation

- These all help to characterize the object

Introduction

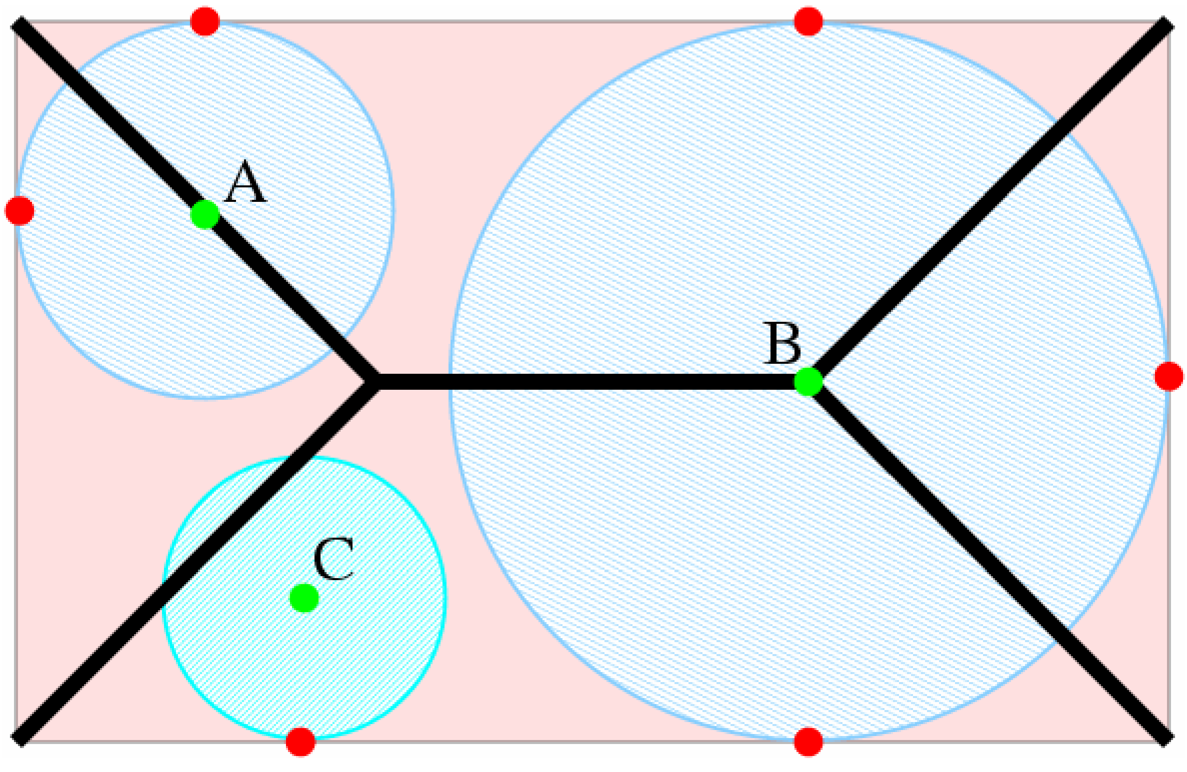
- Most techniques to extract a skeleton are based on Thinning
  - Most of these only differ performance wise
  - Rarely do they differ on how the skeleton is found or it quality

Medial-Axis Transform

- One of the first definitions for the skeleton
  - BLUM 1967
- Originally based on the concept of wave fronts
- Different ways to accomplish
  - BLUM MAT
  - Distance Maps

BLUM MAT

- For any point P in the object
  - Locate the closest boundary point, if there is more then one at this minimum distance – It's a skeletal point



Distance Maps

- Concept
  - Make an image in which its object's pixels are colored based on their distance from the object's boundary
  - Dark pixels are close to the object's boundary
  - Light pixels are further away from the boundary

Distance Maps

- 3 ways to compute the distance
  - Euclidean Distance
    - Coordinates
  - City Block Distance
    - Up – Down : Left – Right
  - Chessboard Distance
    - Any Direction (King Move)

$$D = \sqrt{(x_2 - x_1)^2 - (y_2 - y_1)^2}$$

$$D = | x_2 - x_1 | + | y_2 - y_1 |$$

$$D = \max(| x2 - x1 |, | y2 - y1 |)$$

Distance Maps

- Another way to compute the distance is to use morphology
  - Erosion

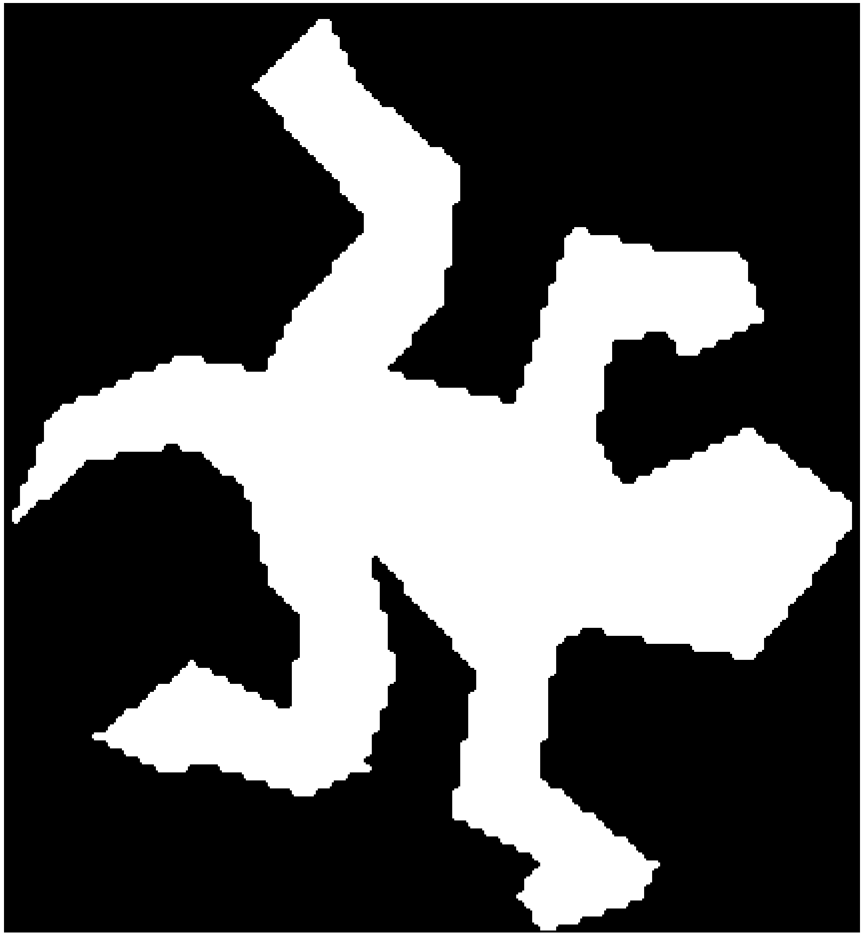
1	1	1
1	1	1
1	1	1

  - If foreground pixels match mask, origin stays
  - Else, origin is deleted
    - The distance map for that point gets the number of erosions before it was deleted

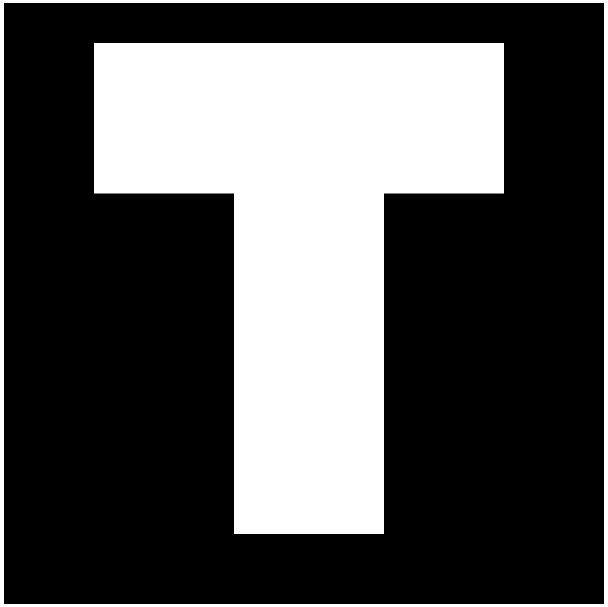
Distance Maps

- Once the map is obtained, we could apply two methods of finding the skeleton from it
  - Apply Laplacian Operator
  - Threshold to extract max values

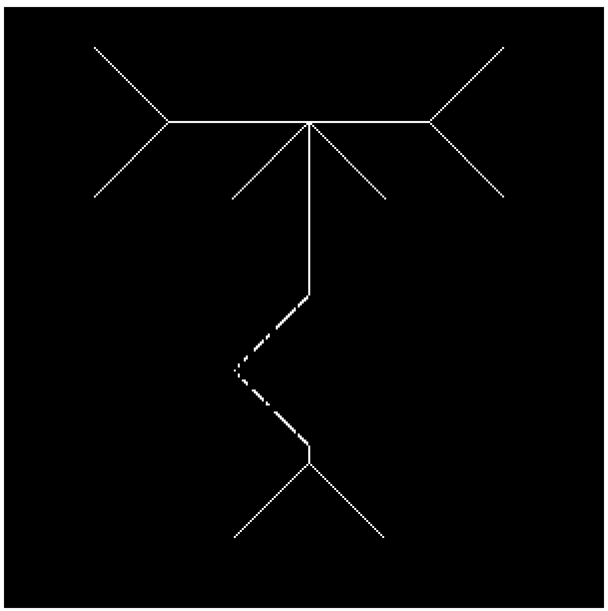
Distance Maps



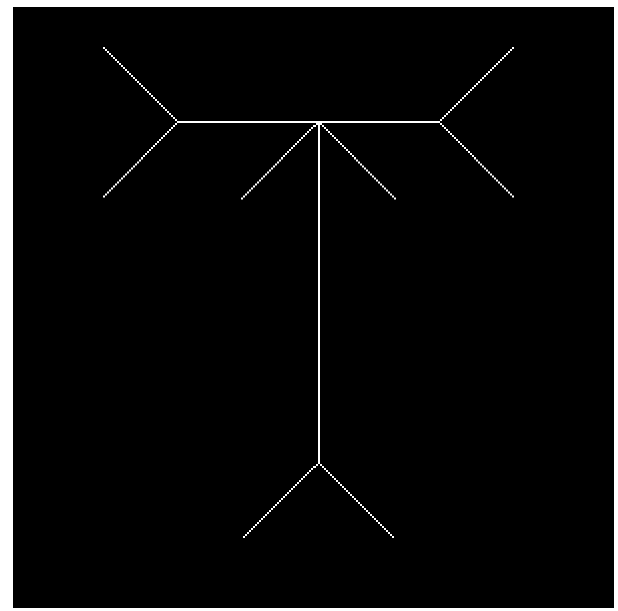
Distance Maps



Final Output
Using Laplacian



Missing ONE pixel on its side



A “perfect” T output

Thinning

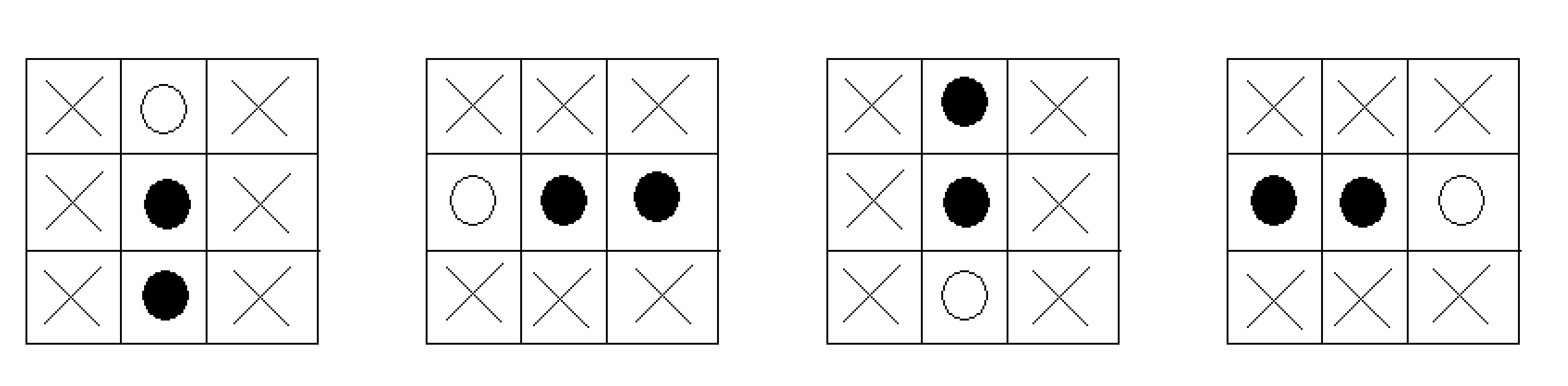
- A form of morphology
- Identifying those pixels belonging to a object that are essential for representing object’s shape
- Stripping away outer pixels

Thinning

- Three things
  - Not all objects should be thinned
  - The same skeleton will not work for all situations
    - It depends on what you need in further processing
  - Thinning is not defined by the algorithm used
    - It identifies the skeleton

Thinning: Iterative Morphological Methods

- Stentiford 1983
- Use 3x3 templates
  - A match of the template means to delete the origin pixel



Thinning: Iterative Morphological Methods

1. Find location (i,j) in image who matches template 1
2. If origin is not a endpoint and connectivity = 1, mark for delete
3. Repeat 1 and 2 for all pixel locations matching template 1
4. Repeat 1-3 for templates 2, 3, and 4
5. If any pixels marked for deletion, delete them
6. If any pixels are deleted, repeat process from step 1, otherwise stop

Thinning: Iterative Morphological Methods

- A pixel is an endpoint if it has only one neighbor

- Checking connectivity because we don’t want to split an object into two

Connectivity

$$C_n = \sum_{k \in S} N_k - (N_k \bullet N_{k+1} \bullet N_{k+2})$$

- S={1,3,5,7}
- Origin = N_0
- N_k = 1, if pixel is background
- N_k = 0, if pixel is part of object
- N_1 is to the right of the origin, ordered counterclockwise

N4	N3	N2
N5	N0	N1
N6	N7	N8

Part 2

- More Connectivity
- Stentiford: Minimize Artifacts
- Zhang-Suen Thinning
- Holt Thinning
  - Holt staircase removal

Connectivity

- More Connectivity
  - Visit P_2, P_3, P_4, …, P_8, P_9, P_2
  - Number of black-white(01) changes = Connectivity
  - 1 = object pixel, 0 = background

P_9 $(i-1, j-1)$	P_2 $(i-1, j)$	P_3 $(i-1, j+1)$
P_8 $(i, j-1)$	P_1 (i, j)	P_4 $(i, j+1)$
P_7 $(i+1, j-1)$	P_6 $(i+1, j)$	P_5 $(i+1, j+1)$

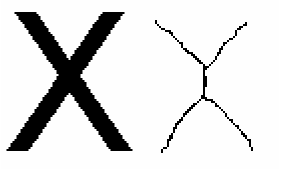
1

0	0	1
1	P_1	0
1	0	0

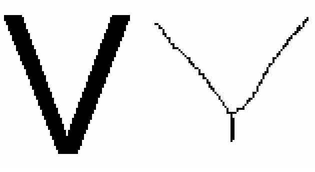
2

Artifacts

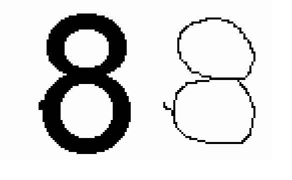
- Necking
  - Intersection of two lines



- Tailing
  - Lines meeting at an acute angle



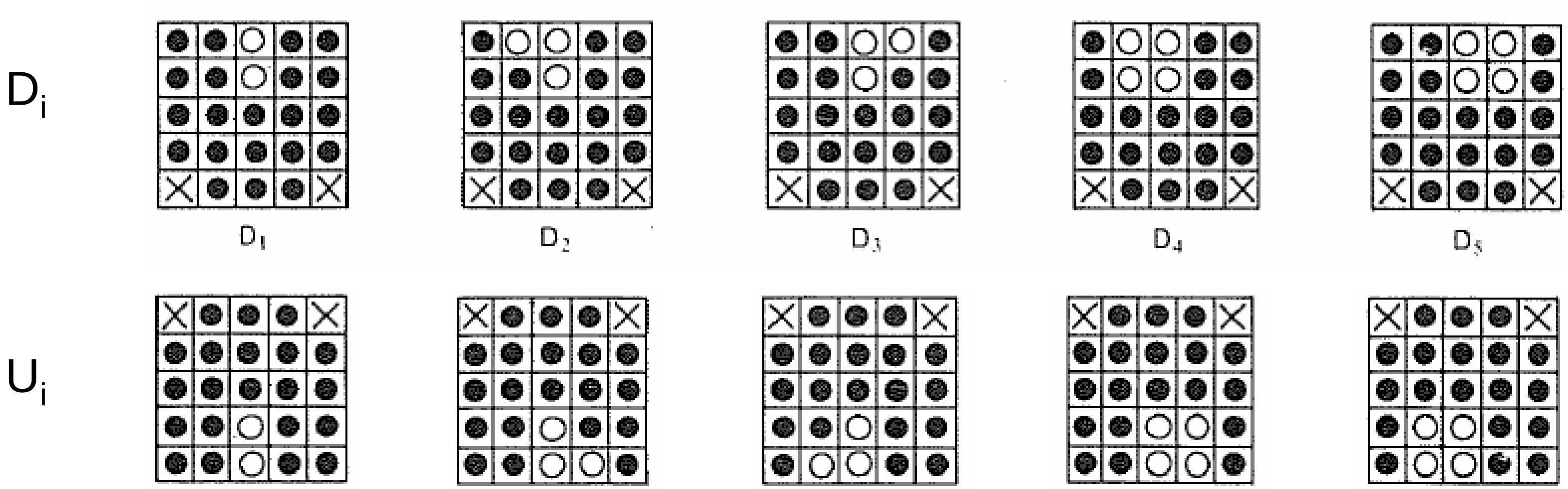
- Projections
  - Rough images



Stentiford: To Minimize Artifacts

- A preprocess stage to minimize artifacts
- Smoothing (projections)
  - Pass over all pixels
    - Delete current pixel if it has less than 3 object neighbors and Connectivity < 2
- Acute Angle Emphasis (necking)
  - Pixels near the joint between two lines are set to the background if they “plug up” an acute angle

Acute Angle Emphasis



- Run each mask through once
  - Mark origin for delete if mask matches
- Delete all marked pixels
- If any deleted
  - Run through again but with masks D(1,2,3) and U(1,2,3) only, mark matches
  - Else exit
- Delete all marked pixels
- If any deleted
  - Run Through again but with masks D1 and U1 only, mark matches
  - Else exit
- Delete Marked and exit

Zhang-Suen Thinning Algorithm

- Fast and Simple
  - Can be parallelized
  - 2 subiterations
- First subiteration: pixel I(i,j) is marked for deletion if ALL of the following 4 conditions are true
  1. Its connectivity = 1
  2. Has at least 2 object neighbors and not more than 6
  3. At least one of P_2, P_4, P_6 are background
  4. At least one of P_4, P_6, P_8 are background
- Delete Marked

Zhang-Suen Thinning Algorithm

- Second subiteration: same as first except rules 3 and 4 are changed

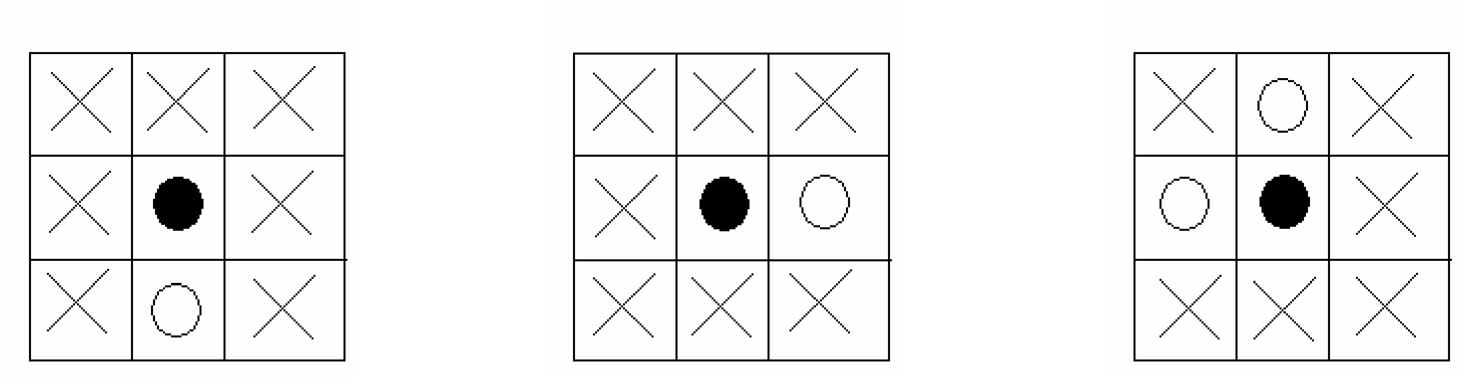
3. At least one of P_2, P_4, P_8 are background
4. At least one of P_2, P_6, P_8 are background

- Delete Marked

- If at the end of any subiteration there are no pixels to be deleted, the skeleton is complete

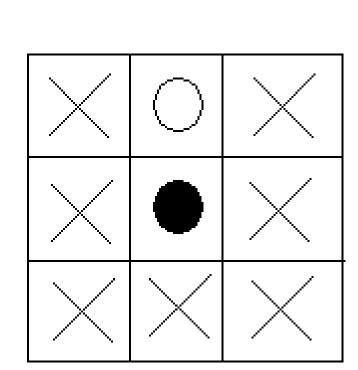
Zhang-Suen Thinning Algorithm

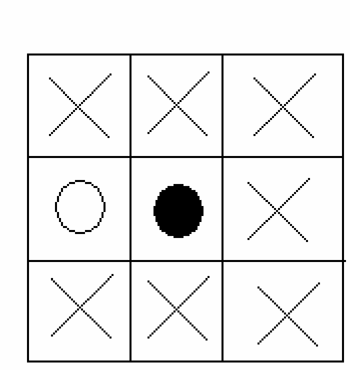
- First subiteration removes south or east boundary pixels or north-west corner pixels

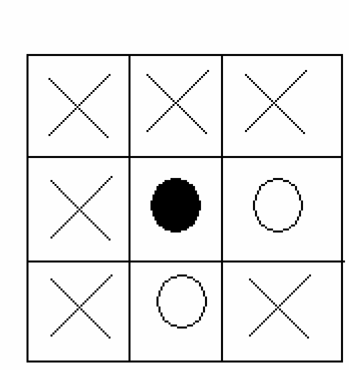


Zhang-Suen Thinning Algorithm

- Second subiteration removes north or west boundary pixels or south-east corner pixels







Holt Thinning

- Pixel Survival: If condition true, pixel stays

- First subiteration:

$$v(C) \wedge (\neg edge(C) \vee (v(E) \wedge v(S) \wedge (v(N) \vee v(W))))$$

- Second subiteration

$$v(C) \wedge (\neg edge(C) \vee (v(W) \wedge v(N) \wedge (v(S) \vee v(E))))$$

Holt Thinning

- Holt comes up with one subiteration if edge information about neighbors is made available

$$
\begin{aligned}
&v(C) \wedge (\neg edge(C) \vee \\
&(edge(E) \wedge v(N) \wedge v(S)) \vee \\
&(edge(S) \wedge v(W) \wedge v(E)) \vee \\
&(edge(E) \wedge edge(SE) \wedge edge(S)))
\end{aligned}
$$

Holt Thinning

- North-West Bias
- Pixel on west edge is kept if it is not on a corner and its east neighbor is on a edge (vertical 2-stroke)

$$edge(E) \wedge v(N) \wedge v(S)$$

- Pixel on north edge is kept if it is not on a corner and its south neighbor is on a edge (horizontal 2-stroke)

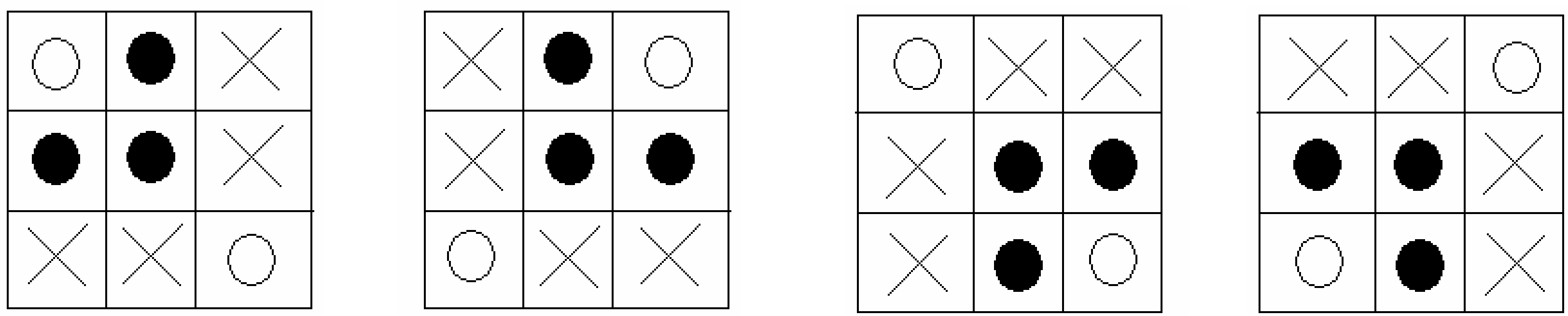
$$edge(S) \wedge v(W) \wedge v(E)$$

- Pixel is kept if it is part of a 2x2 square

$$edge(E) \wedge edge(SE) \wedge edge(S)$$

Holt Staircase Removal

0	1	0	0
0	1	1	0
0	0	1	0



- To make sure a hole is not create
  - One of the unknown side neighbors must be 0
    - Side Neighbors: N, E, S, and W

Side Notes

- Holt is faster then Zhang-Suen
- Zhang-Suen gives a better skeleton
- Parker:
  - Stentiford’s Preprocessing
  - Zhang-Suen’s Thinning Algorithm
  - Holt’s staircase removal Postprocessing

Reference

- Algorithms for Image Processing and Computer Vision – J. R. Parker

- Christopher M. Holt , Alan Stewart , Maurice Clint , Ronald H. Perrott, An improved parallel thinning algorithm, Communications of the ACM, v.30 n.2, p.156-160, Feb. 1987

- T. Y. Zhang , C. Y. Suen, A fast parallel algorithm for thinning digital patterns, Communications of the ACM, v.27 n.3, p.236-239, March 1984

- http://www.ee.ucl.ac.uk/~fstentif/Thinning.pdf