FUNDAMENTALS OF IMAGE AND VIDEO PROCESSING

Thinning

What we'll see in this section

- Hit-or-miss trasform
- Thinning
- Zhang-Suen Thinning Algorithm
- Safe Point Thinning Algorithm
- Summary

Hit-or-miss trasform

- The **hit-or-miss trasform** [1.] is a general binary morphological operation that can be used to look for a particular patterns of foreground and background pixels in an image (1s and 0s respectively).
- It is the basic operation of binary morphology.

	1	
0	1	1
0	0	

Figure 1: Example of kernel

- It is applied to **binary images** and produces another binary image as output.
- It uses a structuring element or kernel to look for those patterns.
- It uses a 3 × 3 kernel. It can contain 1s, 0s and "I don’t care" value.

Hit-or-miss trasform

- It translates the origin of the structuring element to all points in the image, and then comparing the structuring element with the underlying image pixels.
- It is performed simply iterating over all pixels of the image and comparing the kernel with the underlying image pixels. If they exactly match the pixel under the pixel is set to 1 otherwise is set to 0.

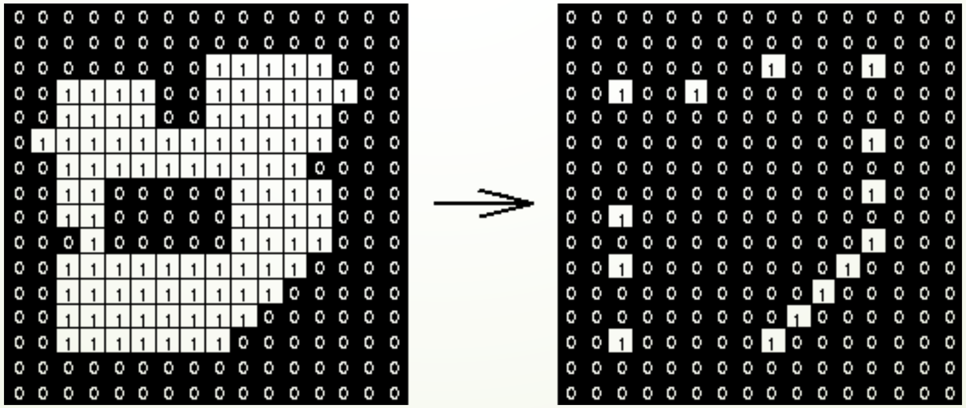


Figure 2: Example of Hit-or-miss trasform combining all the images

Thinning

- The hit-or-miss transform has many applications in more complex morphological operations. It is being used to construct the thinning and thickening operators.
- **Thinning** [2.] is a morphological operation that is used to remove selected foreground pixels from binary images.
- It is commonly used to tidy up the output of edge detectors by reducing all lines to single pixel thickness.
- It is applied to binary images and produces another binary image as output.

Thinning

- Given a binary image *I* and a structuring element *J*:
  thin(I,J)= 1 − hit-or-miss(I,J), where the subtraction is the logical subtraction X − Y = X ∩ ¬Y.
- A pixel is deleted (i.e. set to 0) if kernel and sub-image do not exactly match, otherwise is left unchanged.
- This is the effects of a single pass of a thinning operation over the image. In fact, the operator is normally applied repeatedly until it causes no further changes to the image (*i.e.* until *convergence*).
- The choice of the kernel determines which pixels are deleted from the region.

Various Algorithms in Thinning

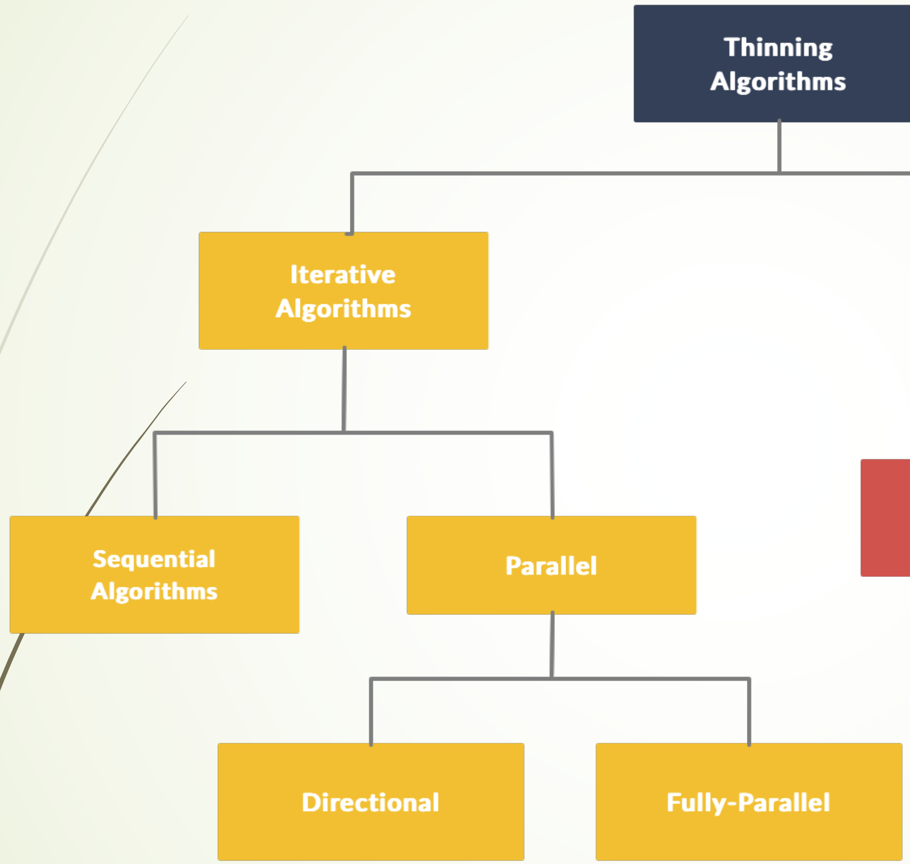


Figure 3: Types of Thinning

Iterative Algorithm

- It works on the pixel by pixel based thinning.
- It examine the pixels until the result is obtained.
- It mainly divides into two parts Parallel and sequential.
- Sequential thinning takes place in predetermined order in which processing takes place in fixed sequence.
- In parallel thinning only the result that remains after the previous iteration is taken in consideration.

Non-Iterative Algorithm

- It is just opposite to iterative as it does not works on the pixel to pixel examine.
- There are some popular methods on which implementation is based like medical transforms, distance transforms and other methods.
- The results of medical transforms contain no noise in the data because he works on the grey-level imaging technique where intensity is represented in terms of distance from the boundary.

Zhang-Suen Thinning Algorithm

- The **Zang-Suen Thinning Algorithm** [3.] is a valid and well-known algorithm that was proposed by Zhang and Suen in 1984.
- It works on an iterative parallel thinning algorithm that works on 3*3 neighbourhood.
- It takes a binary 2D image and removes pixels from the object's border by making successive iterations until convergence.

P9	P2	P3
P8	P1	P4
P7	P6	P5

Figure 4: ZS 3*3 Neighbourhood

Zhang-Suen Thinning Algorithm

- How we can see in **Figure 2**, a binary 2D image is a matrix M where each pixel M[i][j] is either 1 or 0.
- A region in an image is a connected set of pixels with 1 value.
- The algorithm iteratively removes all contour points, changing the value from 1 to 0, that satisfy certain conditions about their 8 neighbours.
- "To work in parallel" for the pixels means that 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.

Zhang-Suen Thinning Algorithm

- It is called a 2-pass algorithm, which means that for each iteration it performs two sets of checks to remove pixels from the image.
- The first two conditions are the same for both:
  - ***2 <= N(P1) <= 6***
  - ***S(P1) = 1***
- The other two differ in: ***P2 * P4 * P6 == 0*** *and* ***P4 * P6 * P8 == 0*** for the first sub-iteration, ***P2 * P4 * P8 == 0*** *and* ***P2 * P6 * P8 == 0***.
- N(P1) is the number of neighbours of P1 with value 1.
- S(P1) is the number of 0-1 patterns in the sorted sequence of neighbours P2, P3, . . . , P9, P2.

Zhang-Suen Thinning Algorithm

- So, it rotates on an image and processes the whole image to give as output image.

Figure 5: Example of application of the Zhang-Suen Thinning Algorithm

Zhang-Suen Thinning Algorithm

```
function Zhang_Suen_Thinning_Algorithm(image):
    image_Thinned <- image.copy()
    c1, c2 <- 1
    while c1 OR c2 do
        c1 <- []
        rows, columns <- image_Thinned.shape
        for x <- 1 to (rows - 1) do
            for y <- 1 to (columns - 1) do
                if (image_Thinned[x][y] == 1 AND
                    2 <= N(P1) <= 6 AND
                    SP(1) == 1 AND
                    P2 * P4 * P6 == 0 AND
                    P4 * P6 * P8 == 0):
                    c1.append((x,y))
        for x, y in c1 do
            image_Thinned[x][y] <- 0
        c2 <- []
        for x <- 1 to (rows - 1) do
        for y <- 1 to (columns - 1) do
                if (image_Thinned[x][y] == 1 AND
                    2 <= N(P1) <= 6 AND
                    SP(1) == 1 AND
                    P2 * P4 * P8 EQUALS 0 AND
                    P2 * P6 * P8 EQUALS 0):
                    c2.append((x,y))
        for x, y in c2 do
            image_Thinned[x][y] <- 0
    return image_Thinned
```

Figure 6: Pseudo-code of the algorithm

SPTA (Safe-Point Thinning Algorithm)

- It is a fast skeletonization algorithm, that prevents excessive erosion, and gives good skeletons (skeletons that retain the shape information of the original pattern);
- With **skeletonization** we mean an image processing technique that reduces a binary object, or a region, to a 1-pixel wide representation called skeleton.
- A simple approach to skeletonization is provided by thinning, that is what we have described so far.

SPTA (Safe-Point Thinning Algorithm)

n_3	n_2	n_1
n_4	P	n_0
n_5	n_6	n_7

Figure 7: a point p and its neighbourhood

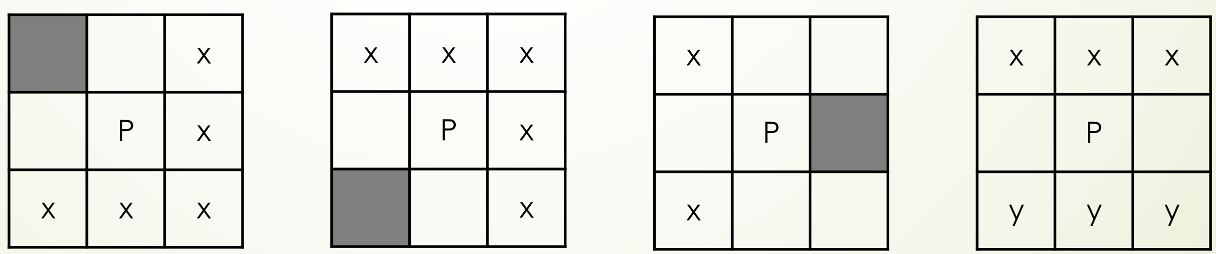
- The **Safe-Point Thinning Algorithm** [4.] is suitable for binary patterns.
- Reconstructability: the algorithm should be such that it preserves enough information in the skeleton to reconstruct as much as possible a model similar to the original one.
- Each element in a binary pattern can be either a dark point or a white point;
- The **eight-neighbours** of point P are defined to be the eight points adjacent to P (points n_0 to n_7 in **Figure 7**);
- Points n_0, n_2, n_4 and n_6 are also referred to as the **four-neighbours** of P;
- A skeleton is said to be *j*-connected (*j* is equal to four or eight) if between any two dark points p_0 and p_n there exists a path $p_0 p_1 \ldots p_{i-1} p_i \ldots p_n$ such that p_{i-1} is a *j*-neighbours of p_i for $1 \leq i \leq n$;

SPTA (Safe-Point Thinning Algorithm)

- In the proposed algorithm we have:
  - An **edge-point:** a dark point that has at least one white four-neighbour;
  - An **end-point:** a dark point that has at most one dark eight-neighbour;
  - A **break-point:** a dark point, the deletion of which would break the connectedness of the original pattern;
- SPTA consists of executing many passes over the pattern, where in each pass a few dark points are **flagged**.
- A flagged point must be such that it is an edge point, but not an end point, nor a breakpoint, and its eventual deletion must not cause excessive erosion in the model. At the end of the pass, all flagged points are deleted.
- If no points are deleted at the end of a pass, then the skeletonization procedure stops.

SPTA (Safe-Point Thinning Algorithm)

- The SPTA identifies an edge-point as one or more of the following four types:
  - a left edge-point, which has its left neighbour n_4 white (see **Figure 7**);
  - a right edge-point, which has n_0 white;
  - a top edge-point, which has n_2 white;
  - a bottom edge-point, which has n_6 white;
- To identify such a left edge-point p, SPTA needs to compare the neighbourhood of p with the four windows:



(a) **(b)** **(c)** **(d)**

Figure 8: if the neighbourhood of *p* matches anyone of the four windows, then *p* is not flagged; *x*'s and *y*'s are *don't care*.

SPTA (Safe-Point Thinning Algorithm)

- If the neighbourhood of P corresponds to any of the windows **(a)**, **(b)** or **(c)** in **Figure 8**, then two situations may occur:
  - if all x's are white, then P is an end-point;
  - if at least one of the x's is a dark point, then P is a break-point. Thus, in either of these cases, P should not be flagged;
- In **(d)** if at least one x and at least one y are dark, then P becomes a break-point and thus it should not be flagged;
- This four windows are the only ones we require to conduct end-point, break-point and excessive erosion tests on a dark point P;
- A left edge-point which matches anyone of the four windows of **Figure 8** is called a left **safe-point**.

Process of SPTA

- A pass in the SPTA consists of two scans, where one scan examines each point in the model (The scan sequence can be in either row or column direction, as chosen by the user):
  - In the **first scan**, all left-edge points and all right-edge points that are not left-safe points and right-safe points, respectively, are flagged (scan 1 flags some left and right edge-points).
  - in the **second scan**, the corresponding top and bottom edge points are flagged (scan 2 flags some top and bottom edge-points).
- If there are no marked points, the procedure stops, otherwise the next step starts.
- Initially in the pattern all dark points are labelled zero and all white points are labelled minus MAXINT;

Labelling of Pattern Points and Reconstruction

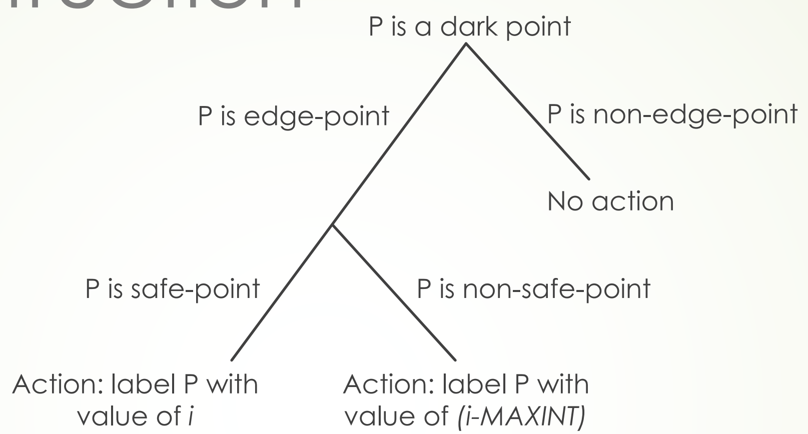


Figure 9: the decision tree required to label a dark point

- During any pass of the skeletonization, the value of the non-edge-points does not change;
- For the edge-points two situations occur:
  - if it is flagged it takes on the value of (*i* – MAXINT), where MAXINT is the largest integer storable in a computer and the value of i represents the number of iterations of the skeletonization process over a pattern;
  - if it is identified as a safe-point, then it is labelled by the value of *i*.

Example of SPTA

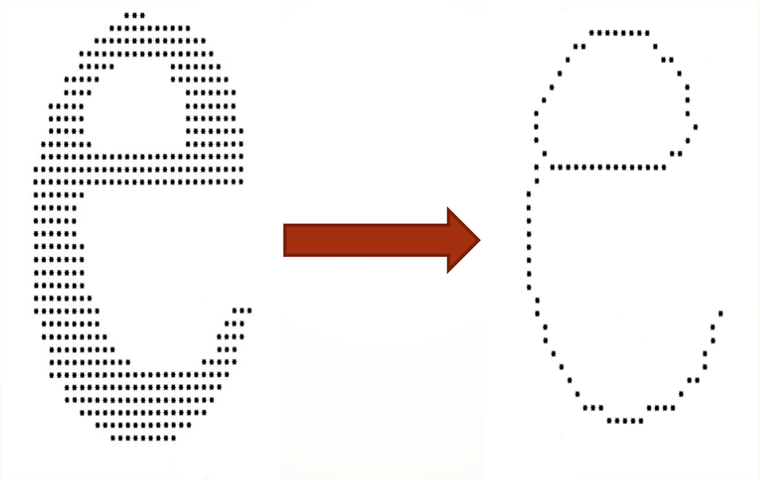


Figure 10: a sample pattern and its skeleton produced by SPTA

Summary

- The thinning method iteratively removes pixels using the **hit-or-miss transform** and some kernels.
- **Thinning** and **skeletonization** are forms of morphological processing and are used when only the fundamental shape of an object on the image is of interest.
- The **Zhang-Suen Thinning Algorithm** is a method that iteratively removes pixels from the boundaries of a region that satisfies certain conditions.
- The **Safe-Point Thinning Algorithm** is an optimal, fast method that produces good skeletons and it has robust reconstructability.

Bibliography

1. Robert Fisher, Simon Perkins, Ashley Walker and Erik Wolfart, '**Image Processing Learning Resources**', 2004.
2. Louissa Lam, Seong-Whan Lee, Member, IEEE, and Ching Y.Suen, Fellow, IEEE, '***Thinning Methodologies – A Comprehensive Survey***', IEEE transactions on pattern analysis and machine intelligence, vol. 14, No.9, September 1992.
3. T. Y. Zhang and C. Y. Suen, '***A Fast Parallel Algorithm for Thinning Digital Patterns***', Commun. ACM 27.3, March 1984.
4. Nabil Jean Naccacche and Rajjan Shinghal, '***SPTA: A Proposed Algorithm for Thinning Binary Patterns***', IEEE transactions on systems, man, and cybernetics, vol. SMc-14, No.3, May/June 1984.