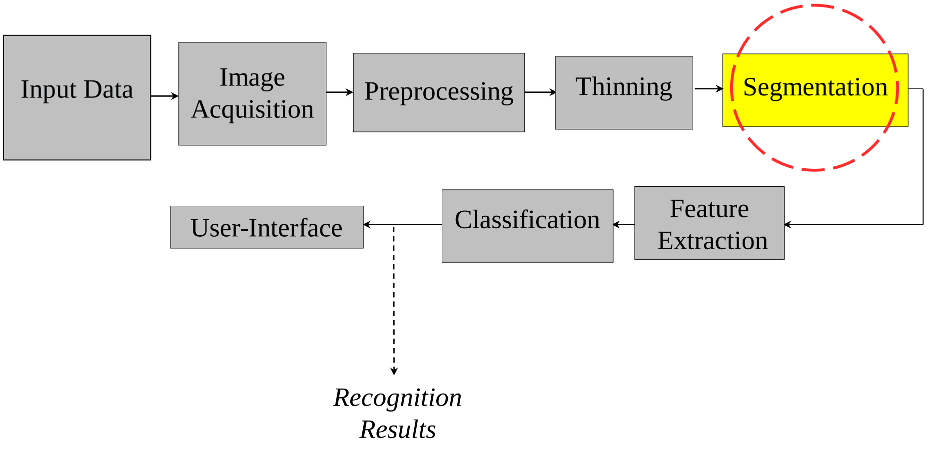
S E G M E N T A T I O N

An important step in Image Analysis for Object Recognition

Image Analysis
for Object Recognition



SPINAKER
Intensywne Międzynarodowe Programy Kształcenia (nabór 2021) - BPI/SPI/2021/1/00066/U/00001

What is our object?
Is it the white ***stanchions or the men****?*

Then accordingly, what is the ***background****?*

Now, John's example of *OPTICAL Illusion*

Hence, segmentation is:

- An essential stage to make the problem of image recognition easier to solve.

- And it involves
dividing object Image into separate regions, in which the resulting parts has simple characteristic points or features (e.g., brightness, color, texture).

For example, separating the **GREEN WOODS** from the **SKY** and the **LAND** in an image.

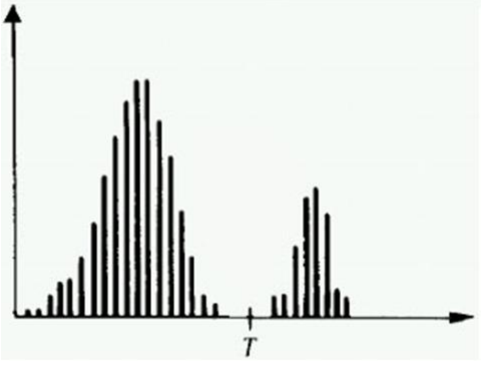
Basic Methods of Image Segmentation

1. Brightness Thresholding

The function that defines the threshold value in this method is:

$$g(x,y) = \begin{cases} 1 & for\ f(x,y) \geq T \rightarrow Image\ pixels \\ 0 & otherwise\ (f(x,y) < T), \rightarrow background \end{cases}$$

The threshold values are usually taken from the image histogramu



Brightness Histogram of the tested object image

Two clear maxima's in the histogram defining the brightness distribution of both the object and the background.

When the threshold value is determined from the WHOLE IMAGE, then we are dealing with *GLOBAL* THRESHOLD.

However, when the threshold is dependent on the *x,y* coordinates of the image, then we have *DYNAMIC* THRESHOLD.

Moreover, when the threshold depends on both image content (i.e., the value of $f(x,y)$ *)* and a certain image feature, $p(x,y)$ say (for example, the image average brightness in some environment), then the threshold is called LOCAL.

We also have multi-level thresholding, in which the level values are

$$T=[T_1, T_2, \ldots T_N].$$

As a result, the image is segmented into ***N+1 different brightness levels.***

As an example, the two-level threshold segmentaion, we have the following regions:

R_1: gdy $f(x,y) \leq T_1$,
R_2: gdy $T_1 < f(x,y) \leq T_2$,
R_3: gdy $f(x,y) \geq T_2$.

Thresholding of brightness can be extended into multi-dimensional thresholding, and hence can be applied to the color images that are encoded in the color representing systems:

RGB (*Red-Green-Blue*)
Or,
HSI (*Hue*, *Saturation*, *Intensity*).

The individual threshold values can be described on the basis of a three-dimensional histogram – constructed from three color components.

Light Distribution Correction

Light distribution correction of some parts $f(x,y)$ can be done by the aid of optical or electronical methods.

For example, to correct the brightness of $f(x,y)$, then knowing the light distribution $g(x,y)$ we can make a brightness correction to:

$$h(x,y)= k*f(x,y)/g(x,y).$$

This of course is a good method when the light distribution does not vary with time.

2. Region Segmentation Approach

Good for images containing some common features like brightness, color, texture.

Here, there are two main methods:

- **Region Growing,**
- **Region splitting and merging.**

2. Region Segmentation Approach

a. Region growing

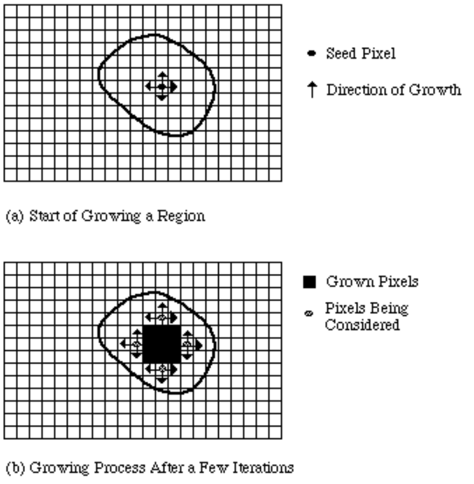
In this method, a primary pixel is considered as a SEED.

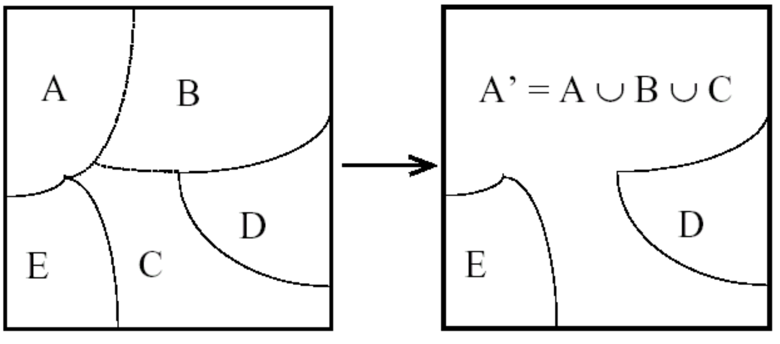
The region is growing by adding the neighbours of similar character (color, texture or brightness).
This is done under a condition to be satisfied, for example:
The condition can be verified, for example statistically (probability of exising – scalar product vector … or by kNN classifiers)

And this is the most difficult task, namely:

how to select the CRITERION of similarity between seeds.

Example of region growing



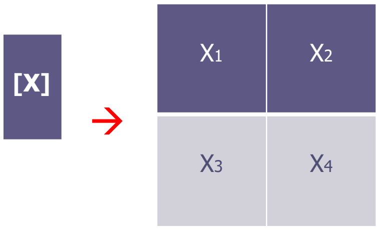


Segmentation by RG

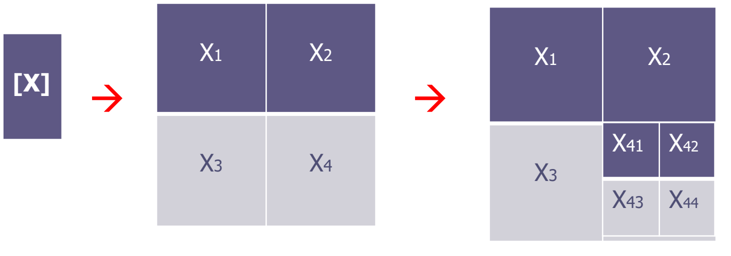
b. Region splitting and merging

Example of region splitting

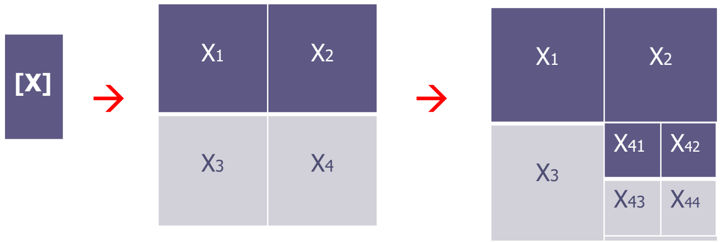
Example of region splitting



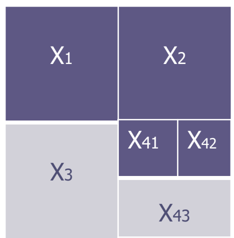
Example of region splitting

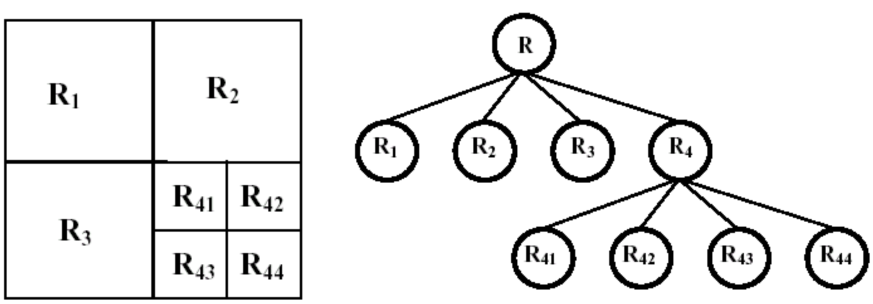


Example of region splitting



and merging





3. Segmentation by Template Matching

Before Thinning
Just RECALL
Segmentation by Binarization

Analysis and Processing of Biometric Images

Thinning

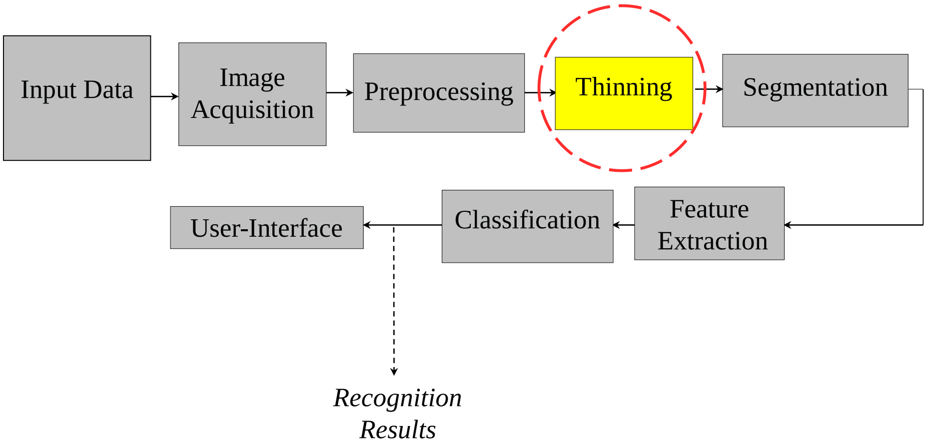
Lecture 04

KMM

Image Thinning Algorithm

SPINAKER
Intensywne Międzynarodowe Programy Kształcenia (nabór 2021) - BPI/SPI/2021/1/00066/U/00001

Image Analysis for Object Recognition

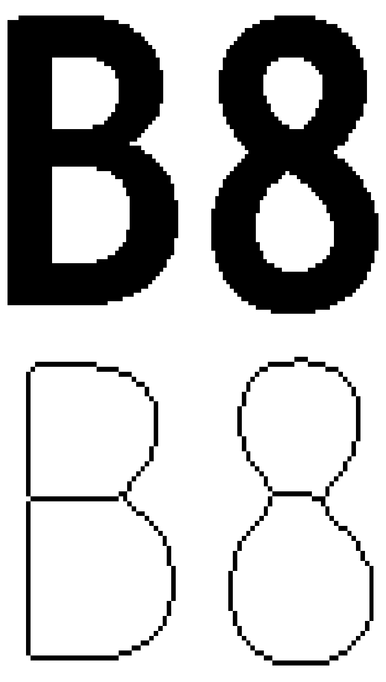


Why?

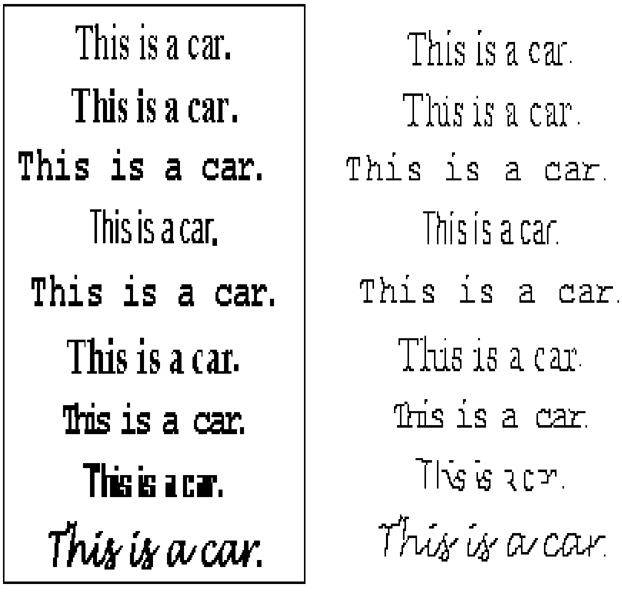
Why thinning

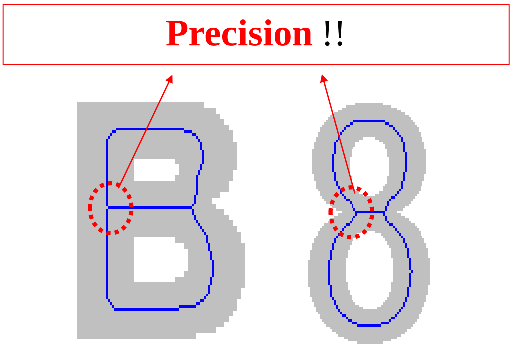
SPINAKER
Intensywne Międzynarodowe Programy Kształcenia (nabór 2021) - BPI/SPI/2021/1/00066/U/00001

One of the most important reason for thinning is *image size reduction*, as can be seen from the following examples:



Examples of thinning whole sentences





Clear difference exists
between the letter B and the digit 8

THINNED IMAGE FEATURES

The output of a GOOD thinning algorithm should be a LINE with:

- one-pixel width,
- noninterrupted,
- and centred.

- Thousands of thinning algorithms …
- In 1967 Blum gave the definition of a ‚skeleton’
- The main aim was to reduce the dimension of images.

Examples:

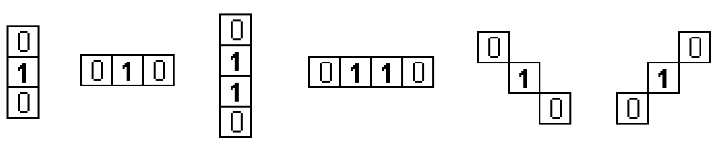
- Rosenfeld’s
- Pavlidis’s
- Zhang’s
- Guo-Hall’s
- Ahmad’s
- Saeed’s
- KMM
- ….

Exemplary Saeed’s Thinning Algorithm

000000000001111	000000000003333	000000000003333	000000000000000	000000000000000
000000000011111	000000000031113	000000000021113	000000000011110	000000000011110
010000000111111	030000000333333	020000000223333	010000000110000	010000000100000
100000001000000	300000003000000	200000002000000	100000001000000	100000001000000
100000001100000	300000003300000	200000002200000	100000001100000	100000000100000
100000001111100	300000003133300	200000003122300	100000000111000	100000000011000
100000000111100	300000000333300	200000000322200	100000000011100	100000000000100
110000000000100	330000000000300	220000000000200	110000000000100	010000000000100
111000000011000	313000000033000	313000000022000	010000000011000	010000000011000
111111111111000	311333333333000	311333332222000	011000001111000	001000001100000
011111111100000	031111113300000	031111112200000	001111111100000	000111110000000
001111110000000	003333330000000	003333330000000	000000000000000	000000000000000

08:15:22

Remove the pixels in such a way that will be left only those which have te following layout:

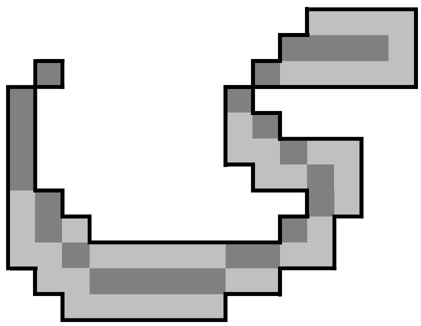


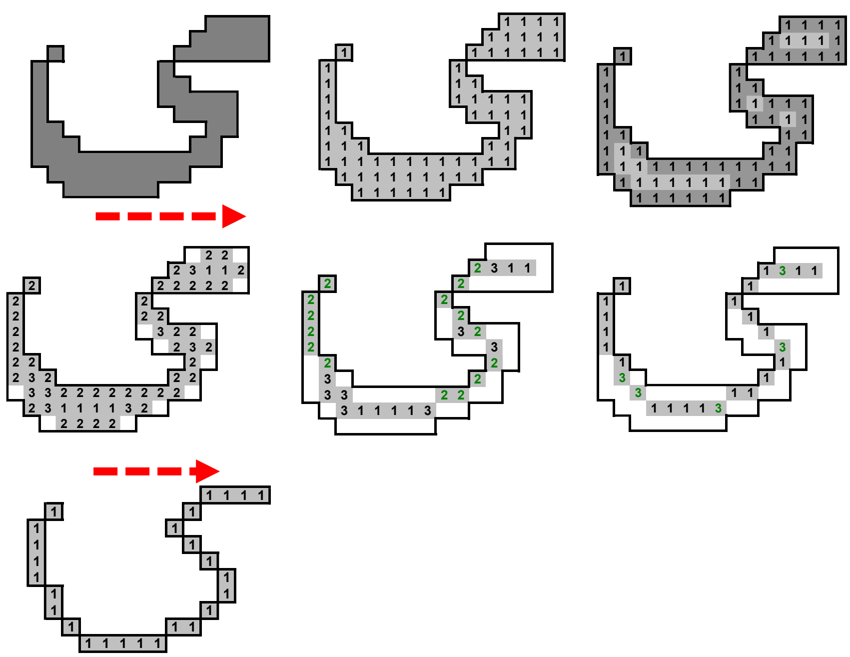
2-pixel line into 1-pixel one:

1000000	1000000	1000000
1100001	x100001	0100001
0111110	0x11110	0011110
0001000	000x000	0000000
0000000	0000000	0000000

Saeed’s modified Algorithm

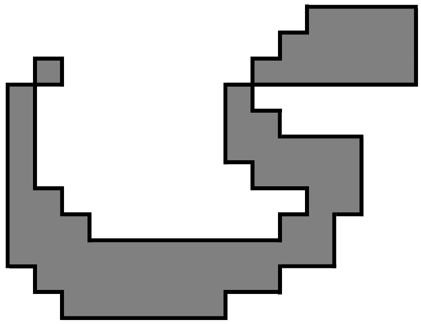
KMM Algorithm





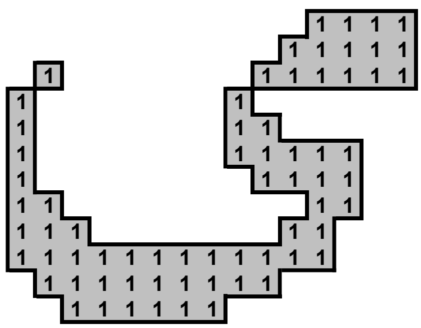
Stages of KMM thinning algorithm

Stage 1



Stage 1

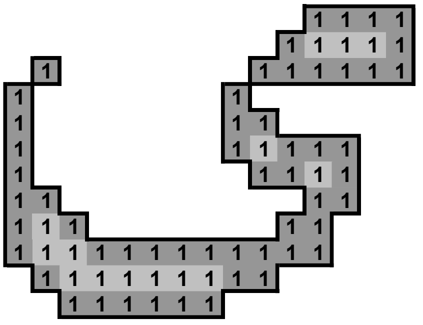
... Image pixels
(the black pixels)
are marked ONES



SPINAKER
Intensywne Międzynarodowe Programy Kształcenia (nabór 2021) - BPI/SPI/2021/1/00066/U/00001

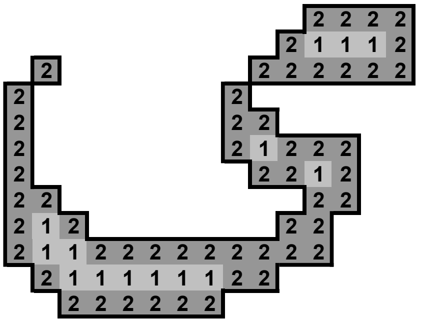
Stage 2

The ONES of direct contact with the background (with the zeros) simultaneously, will form the image contour



Stage 2

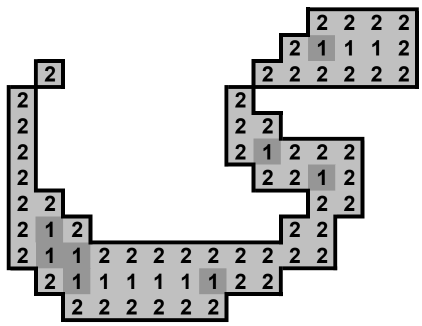
Contour ONES are changed into (marked) ‘2’



Stage 3

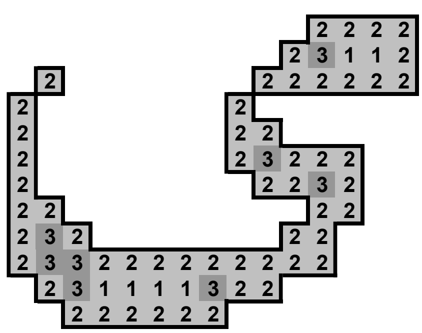
Define the ’elbow points’ –

‚Corner points’



Stage 3

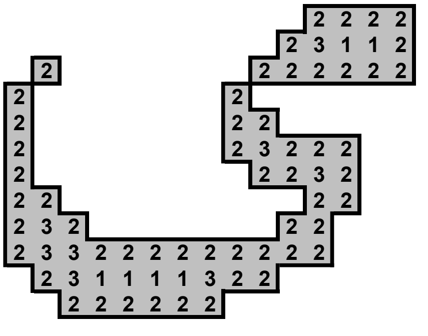
..., and change them into ‘3’



Remark

- This phase may finish the algorithm if the goal is to find the contour.

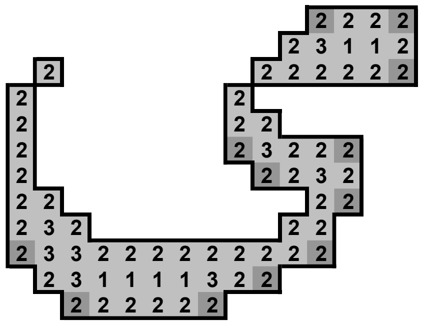
Stage 3 - (contin.)



Extended Version of the Algorithm

- **Stage 4**

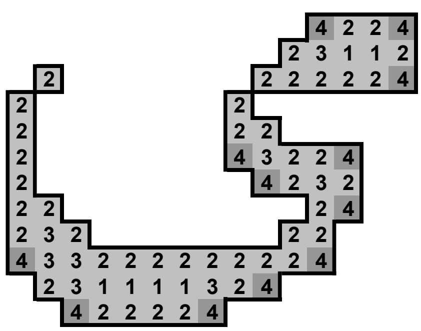
  Contour points (2's forming the outline of the tested image) with 2, 3 or 4 neighbors are …..



Extended Version of the Algorithm

- **Stage 4**

  Contour points (2's forming the outline of the tested image) with 2, 3 or 4 neighbors are changed into 4's



Extended Version of the Algorithm

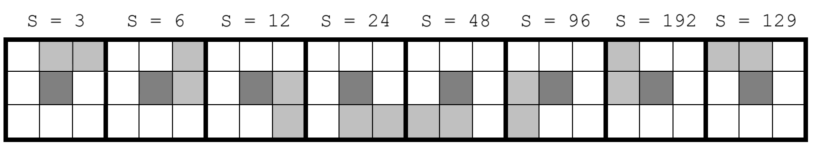
- Notice that there are EIGHT possible configurations for each of the THREE CASES, depending on the distribution of the pixels surrounding the tested pixel ***x***.
- The weight of each neighbouring pixel is taken from the following pixel:

128	1	2
64	*x*	4
32	16	8

Extended Version of the Algorithm

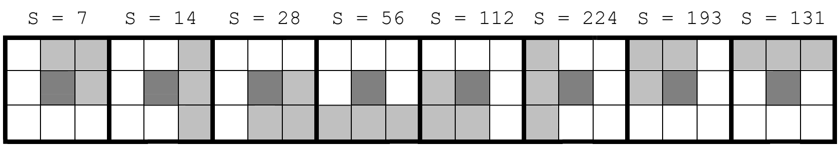
Hence, we have 24 different possibilities:

... with two



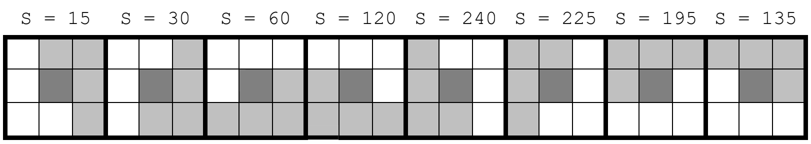
Extended Version of the Algorithm

... with three



Extended Version of the Algorithm

... or with four neighbours

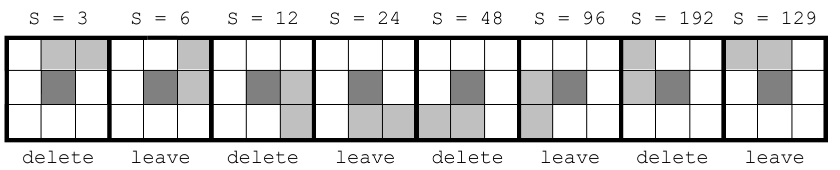


- The sum ***s*** of the neighbouring pixels within the range 0 to 255, will decide whether to leave or remove (delete/discard) the tested pixel.

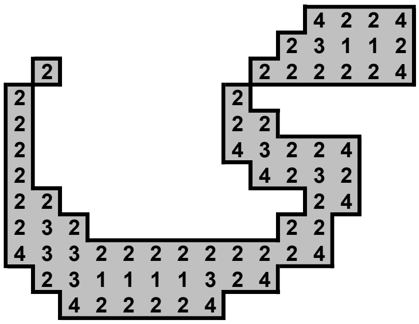
  This table is called ***Deleting Table***. It presents the SUMS for which the tested pixel should be removed (deleted).

3,	5,	7,	12,	13,	14,	15,	20,
21,	22,	23,	28,	29,	30,	31,	48,
52,	53,	54,	55,	56,	60,	61,	62,
63,	65,	67,	69,	71,	77,	79,	80,
81,	83,	84,	85,	86,	87,	88,	89,
91,	92,	93,	94,	95,	97,	99,	101,
103,	109,	111,	112,	113,	115,	116,	117,
118,	119,	120,	121,	123,	124,	125,	126,
127,	131,	133,	135,	141,	143,	149,	151,
157,	159,	181,	183,	189,	191,	192,	193,
195,	197,	199,	205,	207,	208,	209,	211,
212,	213,	214,	215,	216,	217,	219,	220,
221,	222,	223,	224,	225,	227,	229,	231,
237,	239,	240,	241,	243,	244,	245,	246,
247,	248,	249,	251,	252,	253,	254,	255.

For example, for the pixel with two neighbours:



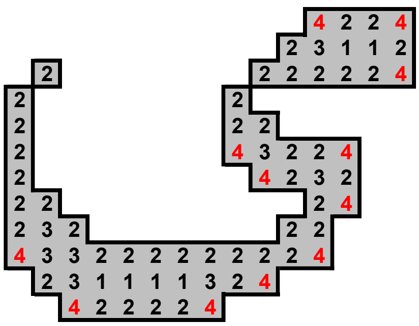
Extended Version of the Algorithm



Extended Version of the Algorithm

Stage 5

Discard the pixels tabbed as '4'

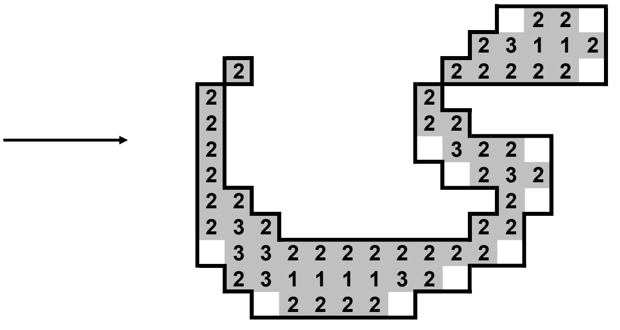


Extended Version of the Algorithm

Stage 5

Discard the pixels tabbed as ‘4’ ….

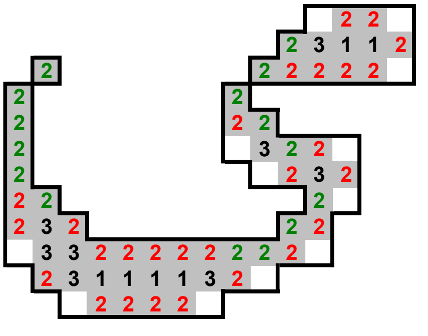
To get → →



Extended Version of the Algorithm

Stage 6

Check all the ‘2’ for the possibility of their removal without losing information (without causing any interruption)

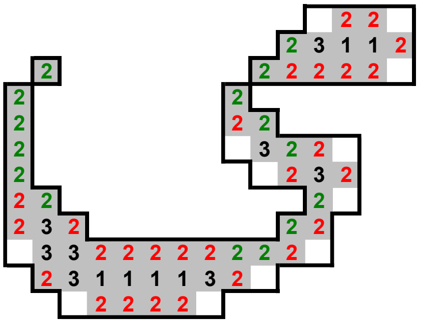


Extended Version of the Algorithm

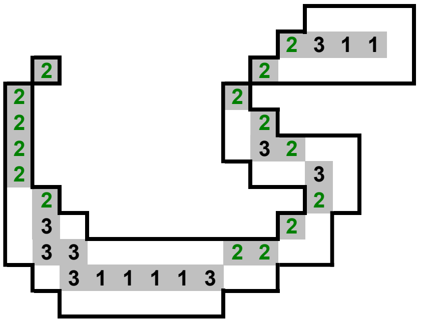
Stage 6

If possible, remove them.

For example, red **2's** are to be deleted →



Extended Version of the Algorithm

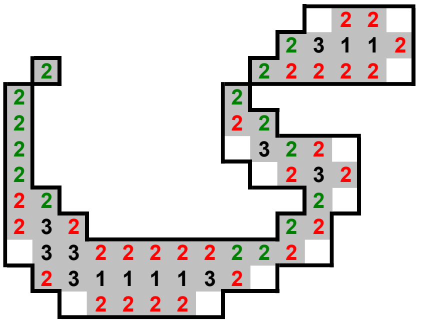


Extended Version of the Algorithm

Stage 6

However, if not

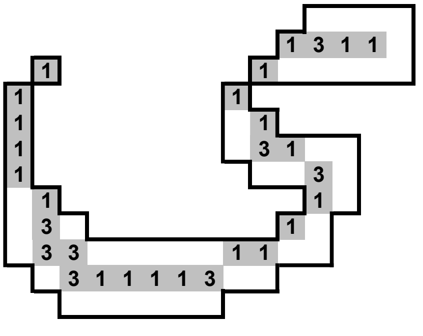
(as the case with the green **2's**, …) →



Extended Version of the Algorithm

Stage 6

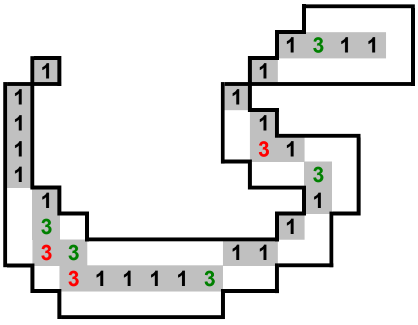
then change them into 1's



Extended Version of the Algorithm

Stage 7

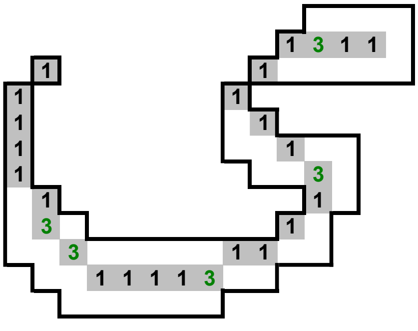
Similar test is repeated, but for the pixels tabbed ’3’.



Extended Version of the Algorithm

Stage 7

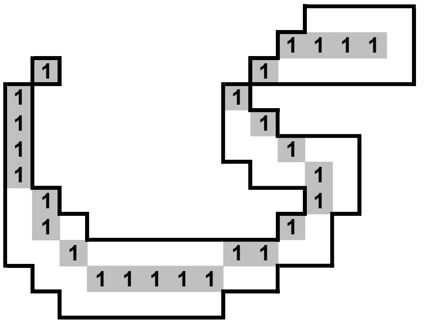
... Delete the pixels according to the Deletion Table.



Extended Version of the Algorithm

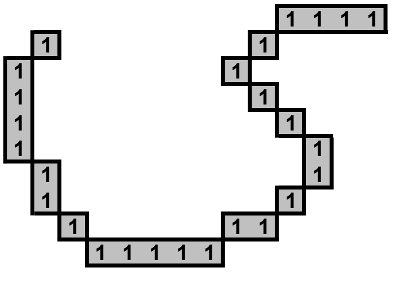
Stage 7

If should left, then
change them into ‘1’



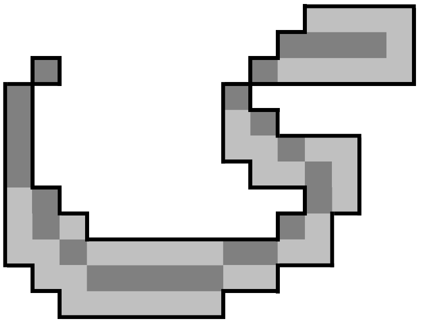
Extended Version of the Algorithm

Repeat stages 2 – 7 as long as there are pixels to remove from the skeleton

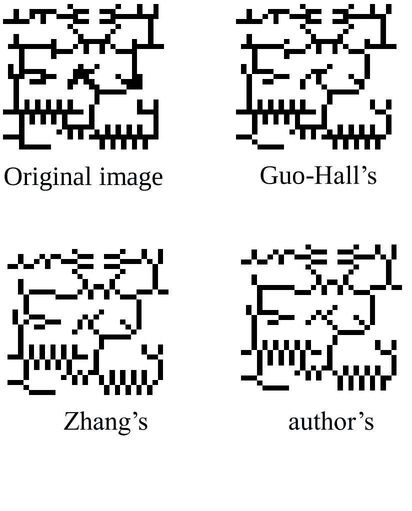


Extended Version of the Algorithm

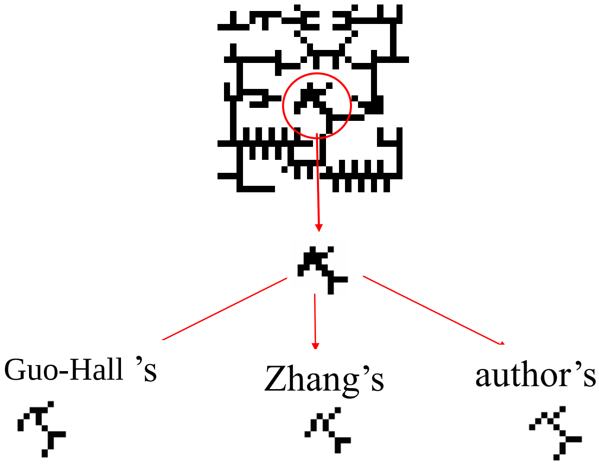
... until obtaining the final possible image shape:

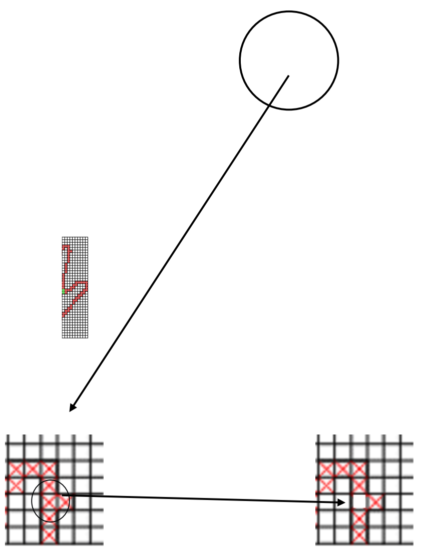


Comparison with other algorithms

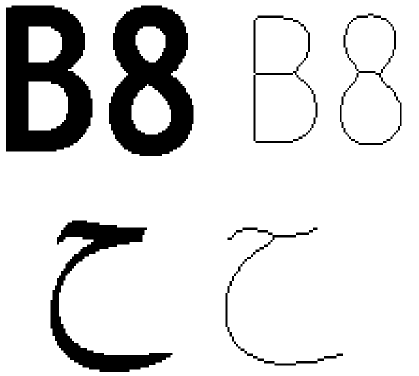


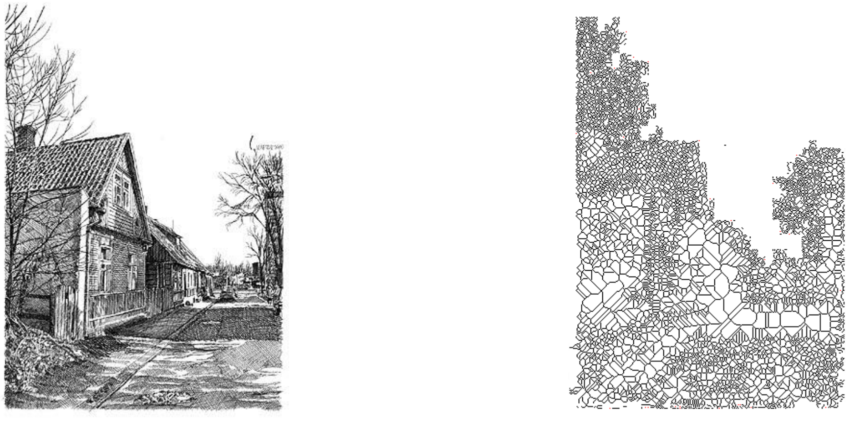
Special considerations





EXAMPLES





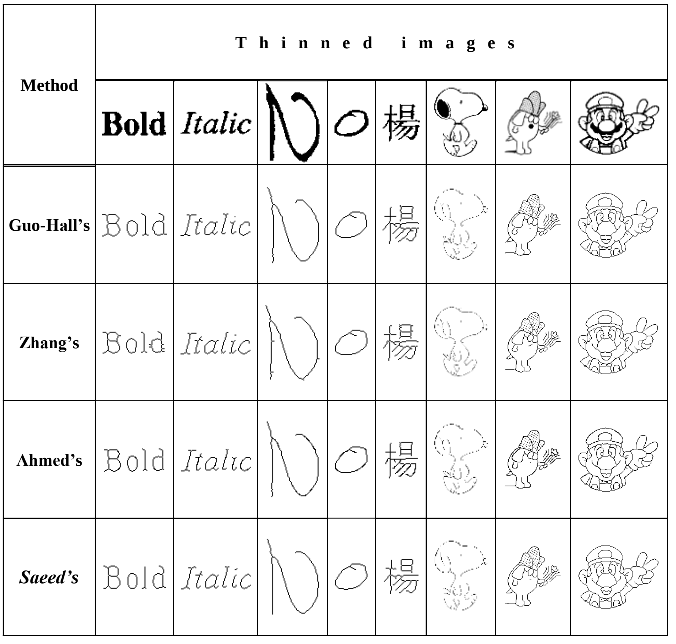
Bitmap – more than 2 colours

SPINAKER
Intensywne Międzynarodowe Programy Kształcenia (nabór 2021) - BPI/SPI/2021/1/00066/U/00001

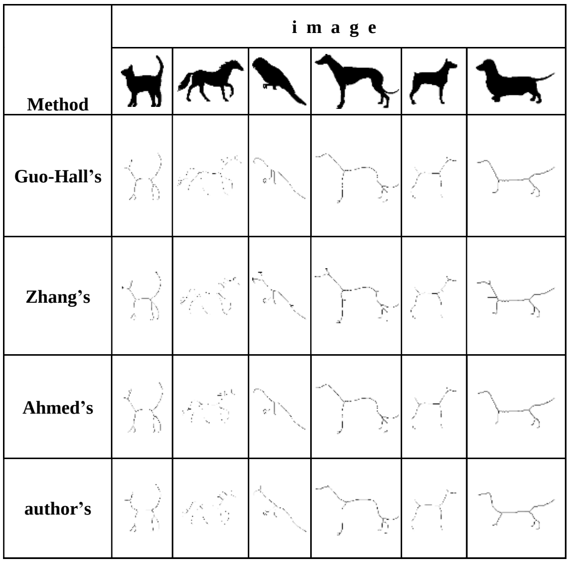
Bitmap – monochromatic

… by another algorithm, for comparison

SPINAKER
Intensywne Międzynarodowe Programy Kształcenia (nabór 2021) - BPI/SPI/2021/1/00066/U/00001


Method
Thinned images
Bold
Italic
楊
Guo-Hall's
Zhang's
Ahmed's
Saeed's

Skeletons


Method
image
Guo-Hall's
Zhang's
Ahmed's
author's

Thank you

SPINAKER
Intensywne Międzynarodowe Programy Kształcenia (nabór 2021) - BPI/SPI/2021/1/00066/U/00001