ECE 172A QUIZ 1 REVIEW

KRITIKA
MURALIDHARAN
02/01/2012

I. EDGE DETECTION

Edge: A place of local transition from one object to another. (Need not be complete borders, just locally identifiable probable transitions)

The following are methods for detecting edges LOCALLY.

FIRST DERIVATIVE METHODS

Most edge detectors are based in some way on measuring the intensity gradient at a point in the image.

Gradient of a function f is

$$\nabla f = \begin{bmatrix} \frac{\partial}{\partial x} f \\ \frac{\partial}{\partial y} f \end{bmatrix}$$

$\|\nabla f\|$ - gradient magnitude
- gives strength of the edge

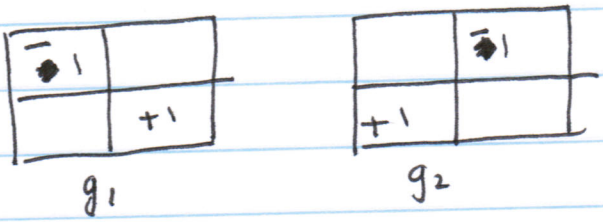
$\phi(\nabla f)$ - direction of greatest intensity change
- edge normal.

- We use discrete approximations of gradients
- Since derivatives are linear and shift invariant, gradient calculations are most often done using convolution

(i) ROBERTS KERNEL:

- Computes forward differences: $\frac{\partial I}{\partial x} \approx I(x+1, y) - I($

Implements this using the following kernels.



g_1 g_2

- These aren't derivatives w.r.t x & y
- They are derivatives w.r.t the diagonal direction
- Can be thought of as the components of a gradient in such a co-ordinate system.

$$\Rightarrow \quad g = \sqrt{(g_1 * f)^2 + (g_2 * f)^2}$$

Disadvantages

- 2X2 $\Rightarrow$ no clear center
- too small to reliably find edges in noisy images

KIRSCH COMPASS KERNELS:

NOTE ABOUT CENTRAL DIFFERENCE

- Sobel & Prewitt Kernels approximate derivatives using central differences.

$$\frac{\partial f}{\partial x} = \frac{f(x+1) - f(x-1)}{2}$$

- This corresponds to the convolution kernel of the form

-1	0	+1

- Rotating this by 90° gives the $\frac{\partial f}{\partial y}$ kernel.
- Kernels like thise are sensitive to noise.
- We can reduce some of the effects of noise by averaging.
- Average along 'y' direction while computing $\frac{\partial}{\partial x}$ and along 'x' direction while computing $\frac{\partial}{\partial y}$.

(3)

SOBEL KERNEL:

- Computes 'central differences' $\frac{\partial I}{\partial x} \approx \frac{I(x+1) - I(x-1)}{2}$
- gives higher weight to central pixels when averaging

-1	0	1
-2	0	2
-1	0	1

$\frac{\partial}{\partial x}$

-1	-2	-1
0	0	0
1	2	1

$\frac{\partial}{\partial y}$

- Larger weight to central pixels ⇒ better noise suppression characteristics

PREWITT KERNEL

- Also computes central differences.

-1	-1	-1
0	0	0
1	1	1

-1	0	+1
-1	0	+1
-1	0	+1

Rotated versions for diagonal edges

Sobel

0	1	2
-1	0	1
-2	-1	0

-2	-1	0
-1	0	1
0	1	2

Prewitt

0	1	1
-1	0	1
-1	-1	0

-1	-1	0
-1	0	1
0	1	1

DIRECTIONAL DERIVATIVES

- Apply rotated versions of masks in a neighborhood and "search" for edges of different orientations
- they do not explicitly compute gradient. Instead, they compute first derivatives in specific directions.
- We approximate the gradient magnitude by the result that produces the maximum first derivative.
- Orientation is limited to these specific directions

Eg: Prewitt compass: (scale fac = $\frac{1}{1+1+1+1+1} = \frac{1}{5}$)

1	1	-1
1	-2	-1
1	1	-1

E

1	-1	-1
1	-2	-1
1	1	1

NE

-1	-1	-1
1	-2	1
1	1	1

N

etc

(5)

KIRSCH (scale factor = $\frac{1}{5+5+5} = \frac{1}{15}$)

-3	-3	-3
-3	0	-3
5	5	5

N

-3	-3	-3
-3	0	5
-3	5	5

NW

-3	-3	5
-3	0	5
-3	-3	5

W

etc.

ROBINSON 3-LEVEL (scale factor = $\frac{1}{1+1+1} = \frac{1}{3}$)

-1	0	1
-1	0	1
-1	0	1

W

0	1	1
-1	0	1
-1	-1	0

SW

1	1	1
0	0	0
-1	-1	-1

S

etc.

ROBINSON 5-LEVEL (scale fac: $\frac{1}{1+2+1} = \frac{1}{4}$)

1	2	1
0	0	0
-1	-2	-1

S

2	1	0
1	0	-1
0	-1	-2

SE

1	0	-1
2	0	-2
1	0	-1

E

etc.

II. NOISE REDUCTION

① SPATIAL AVERAGE FILTERING.

2-D spatial filtering:

I → image

w → filter / convolution kernel

O → filtered output

$$O(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)\, I(x+s, y+t)$$

where, if w is MxN, $a = \frac{M-1}{2}$ & $b = \frac{N-1}{2}$

(ie) at each point, compute output by multiplying each of the pixels in its neighborhood by a weight and add them all up.

Averaging (instead of just summation is)

$$O(x,y) = \frac{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)\, I(x+s, y+t)}{\sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t)}$$

- Advantage: Less noise
- Disadvantage: Affects sharp transitions (ie) blurs edges
- Reducing sharp transitions is low-pass filtering

- 3x3 averaging filter output can be distinguished from larger (say 7x7) filter output by the extent of blurring
  - larger filter ⇒ averaging over larger neighborhood
    ⇒ more 'blurry' output.

Non-linear Smoothing:

One set of non-linear smoothing operators are order-statistic filters

given $\{x_1, \cdots, x_n\}$

order statistics $\{x_{(1)}, \cdots, x_{(n)}\}$ are a re-ordering such that

$$x_{(1)} \leq x_{(2)} \leq \cdots \leq x_{(n)}$$

- Unlike averaging, this is a non-linear operation
- for filters with an odd number of elements, gives a value from the original image.

- very good for salt and pepper noise (significantly less blurring than mean filters)

- Isolated noise clusters with size $< \frac{N^2}{2}$ pixels

  are eliminated by an NxN median filter

Other non-linear smoothing operators.

k- TRIMMED MEAN

- Throw out 'k' highest & lowest values, average the rest.

- output less likely to be affected by outlying neighborhood values.

Example: (salt and pepper noise)

0	0	0	0	0
0	0	0	10	0
0	-5	0	0	0
0	0	0	0	0
0	0	0	0	0

3×3 mean filter →

0	0	10/9	10/9	10/9
-5/9	-5/9	5/9	10/9	10/9
-5/9	-5/9	5/9	10/9	10/9
-5/9	-5/9	-5/9	0	0
0	0	0	0	0

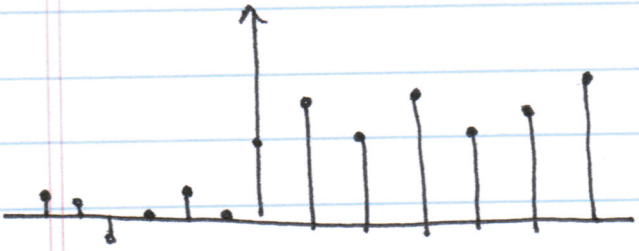
3×3 1-trimed mean filter →

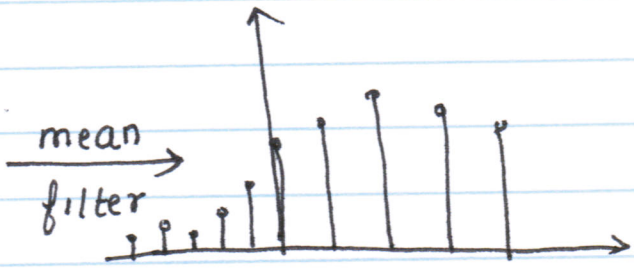
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

3×3 median filter →

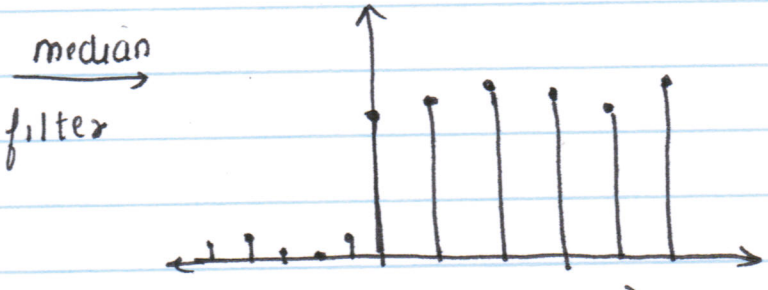
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Example (edge blurring)





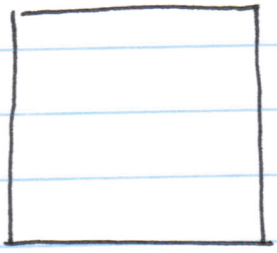
(blurs out edge)



(preserves edge)

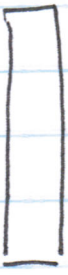
NOTE: - 2-D median filter on a 2D step function $u(n_1, n_2)$ does not preserve the discontinuities (1-D filter preserves edge on 1-D step)

- discontinuities can be preserved by separable median filtering
  - filter using 1-D horizontal median filter followed by 1-D vertical median filter

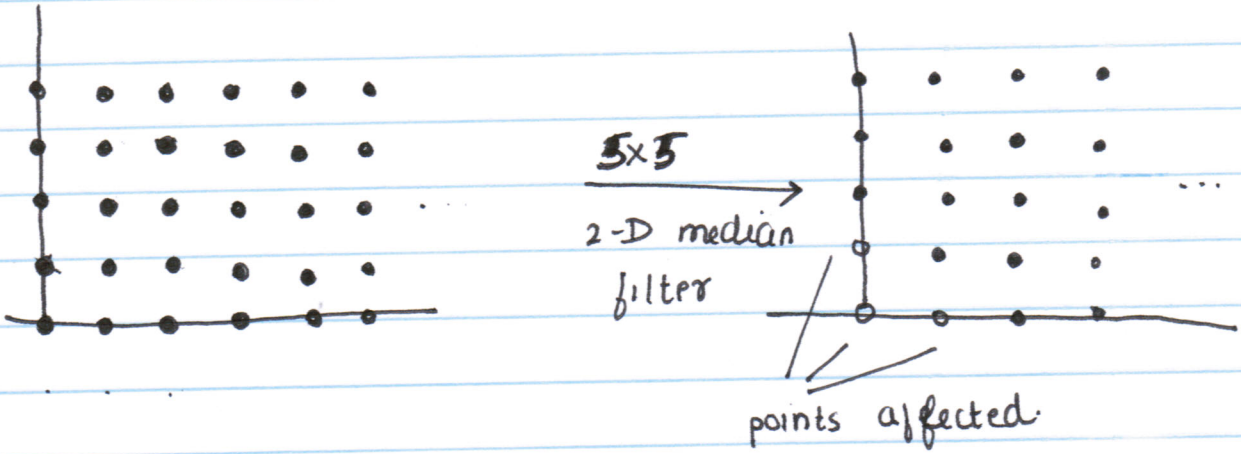


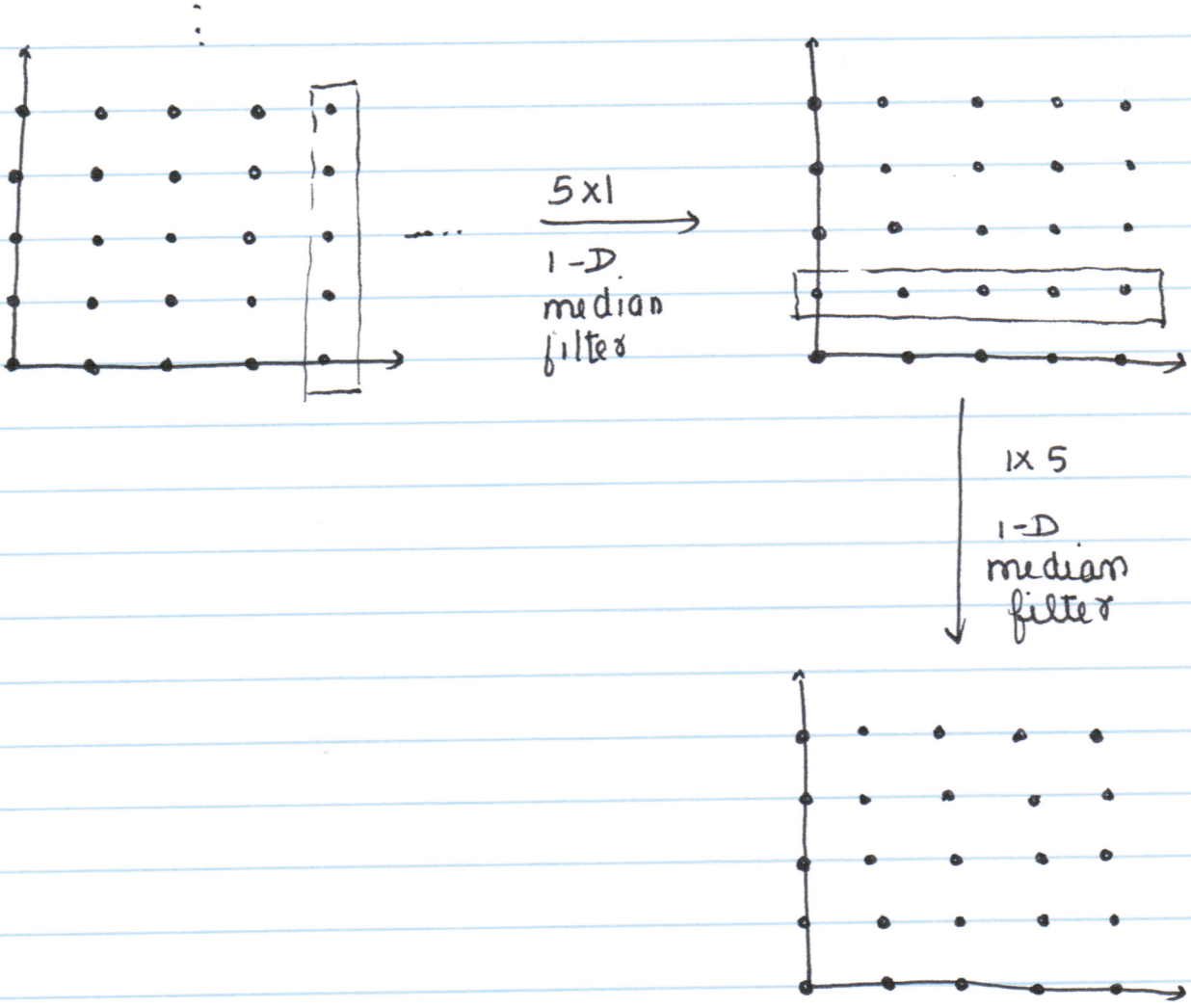
nxn 2D median filter

1xn 1-D horizontal median filter



nx1 1-D vertical median filter





NOTE:

- Repeated application of a median filter on an image adversely affects structures (blobs) smaller than the filter mask but has less of an effect on relatively large blobs

- Smallest surviving blob in the output image can be used to guess the size of the largest filter that has been applied to the image.

Quick Review

(i) Dilation: $A \oplus B = \{ x \mid (\hat{B})_x \cap A \neq \emptyset \}$

- set of all points x such that $(\hat{B})_x$ and A overlap by atleast one pixel
- 'Expands' the boundary of A

(ii) Erosion: $A \ominus B = \{ x \mid (B)_x \subseteq A \}$

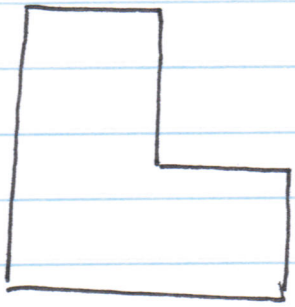
- set of all points x such that $(B)_x$ is completely contained in A
- 'contracts' the boundary of A.

(iii) Opening: $A \circ B = (A \ominus B) \oplus B$

- Erosion first finds all points where the structuring element fits inside the image but only marks the positions at the origin of the element
- with dilation, we 'fill back in' the full structuring element wherever it originally fit inside the object

(iv) Closing: $A \bullet B = (A \oplus B) \ominus B$

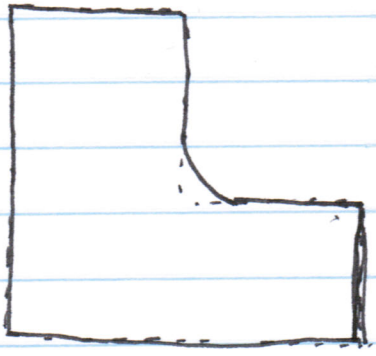
- We can remove small holes in objects by dilating
- However, this also distorts object. This effect is minimized by erosion.



A

B

$A \circ B$



$A \bullet B$

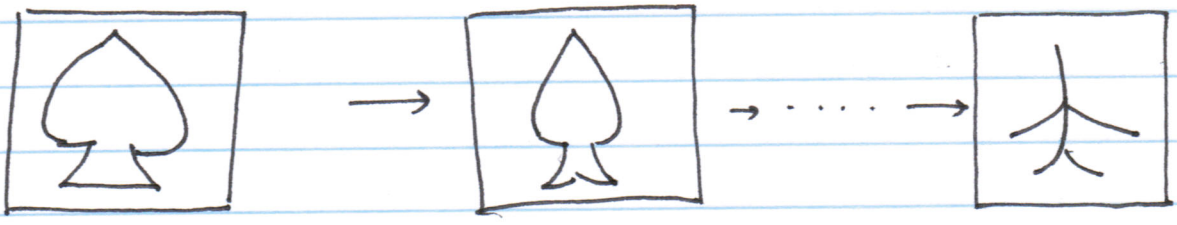
(v) Thinning: $A \otimes B = A - (A \circledast B)$

$= A \cap (A \circledast B)^c$

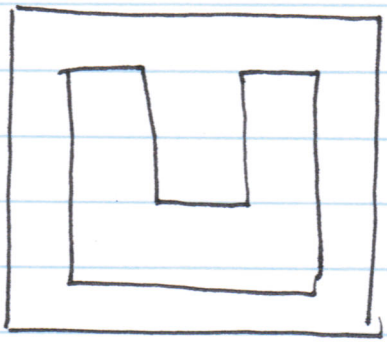
↳ hit-or-miss

Sequential thinning:

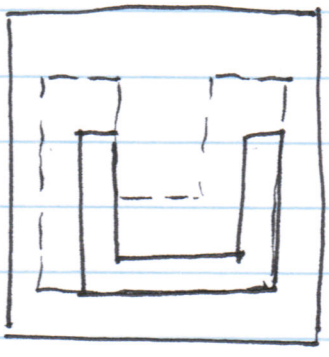
$A \otimes \{B\} = (\cdots((A \otimes B^1) \otimes B^2) \cdots) \otimes B^n$



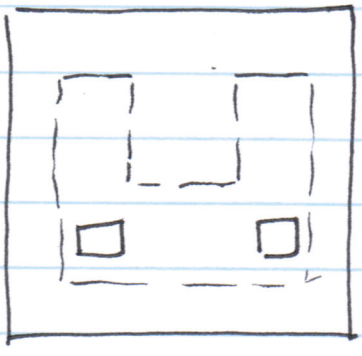
Textbook example:



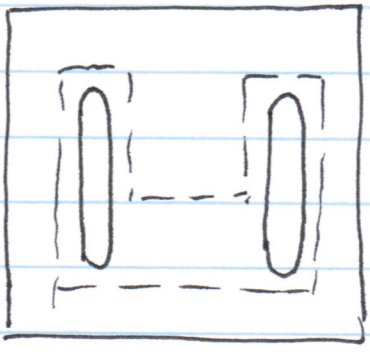
image



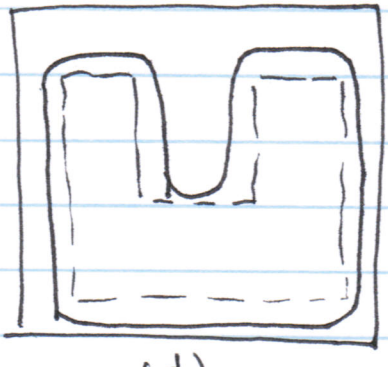
(a)



(b)



(c)



(d)

find the structuring element and morphological operation(s) that resulted in (a), (b), (c) and (d)

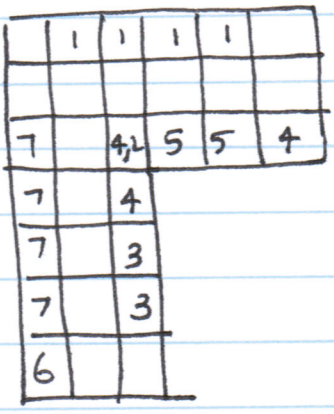
(a) erosion by 

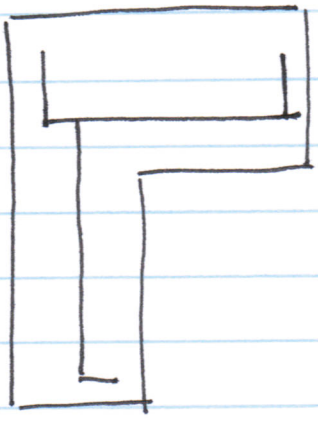
(b) erosion by 

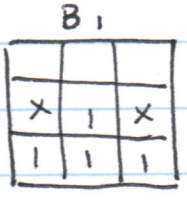
(c) erosion by  and dilation by 

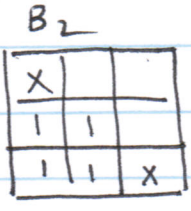
(d) dilation by  and erosion by 

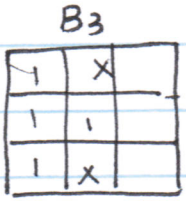
Thinning Example:

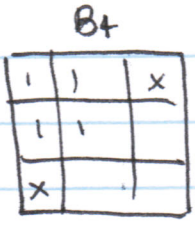


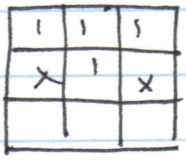


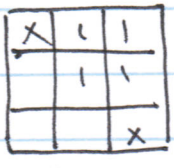


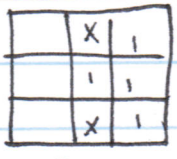


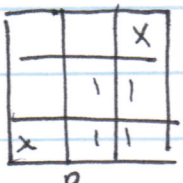










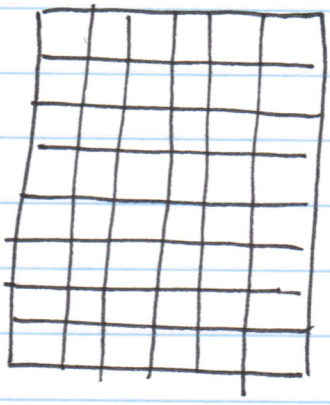


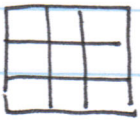
Morphological Skeletonization

$$S(A) = \bigcup_{k=0}^{K} S_k(A)$$

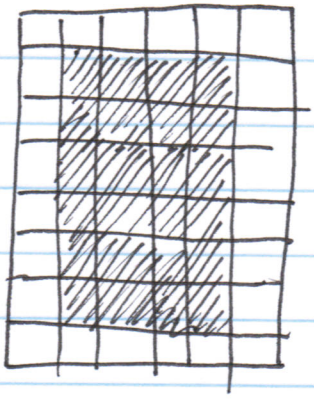
$$S_k(A) = (A \ominus_k B) - (A \ominus_k B) \circ B$$

K = one step before empty set
= max $\{ k \mid (A \ominus_k B) \neq \emptyset \}$.



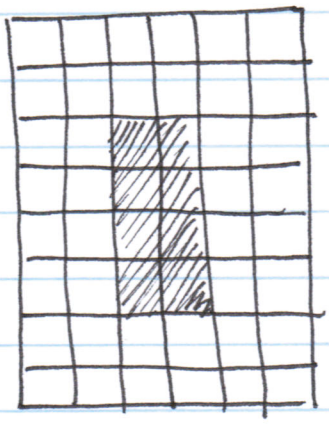


$A \ominus B$ =



$(A \ominus B) \circ B$ = same. $\Rightarrow$ $S_1(A) = \emptyset$

$(A \ominus_2 B) =$



$(A \ominus_2 B) \circ B = \phi \Rightarrow S_2(A) =$

$(A \ominus_3 B) = \phi$

$\therefore S(A) = S_2(A)$.

PRUNING:

- Essential component during skeletonization and thinning
- Used to remove parasitic components (spurs) that need to be cleaned up.

Assumption: parasitic component is < threshold (k)

∴ We can thin the skeleton with a sequence of structuring elements designed to detect end-points.

$$X_1 = A \circledast_k \{B\}$$

↳ 'k' times.

Then, 'restore' skeleton to original form without spurs.

(i) $X_2 = \bigcup_{i=1}^{8} (X_1 \circledast B^i)$ (end point detection)

(ii) Dilate valid endpoints 'k' times (regain what you lost) and don't add any new points

$$X_3 = (X_2 \oplus H) \cap A \qquad H = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$X_4 = X_1 \cup X_3 =$ result