

Mathematical Morphology

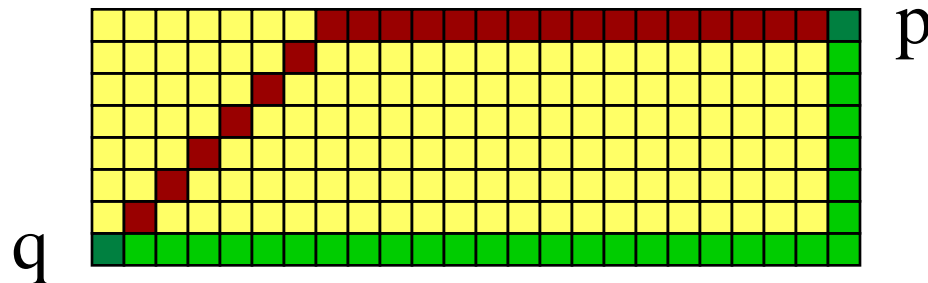
Mathematical Morphology

- ★ Mathematical Morphology was developed in the '60 in France (G. Matheron e J. Serra, Ecole des Mines) and in different form with the name Image Algebra in USA (S. R. Sternberg, Michigan University).

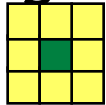
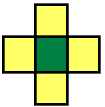
Preliminary statements

- ★ Let $F=\{1\}$ et $F^*=\{0\}$ be the two sets constituting a binary image on a square plane tessellation. Let call them image and background respectively
- ★ Given two pixels p and q , having coordinates (i,j) and (h,k) , the two distance functions d_4 (called city block) et d_8 (chessboard) are defined as:

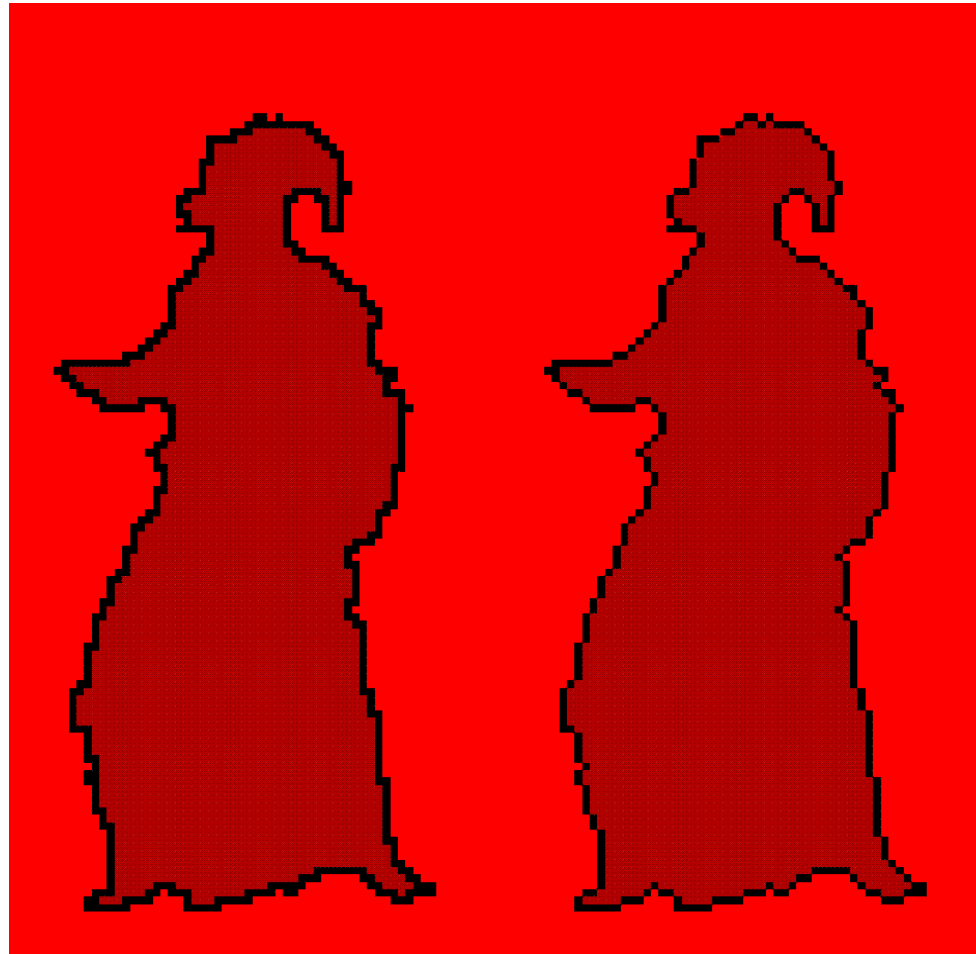
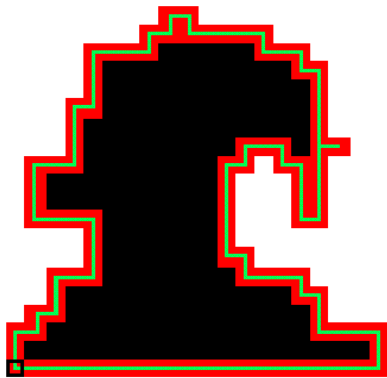
$$d_4(p,q) = |i-h| + |j-k| \quad d_8(p,q) = \max \{|i-h|, |j-k|\}$$



Preliminary statements

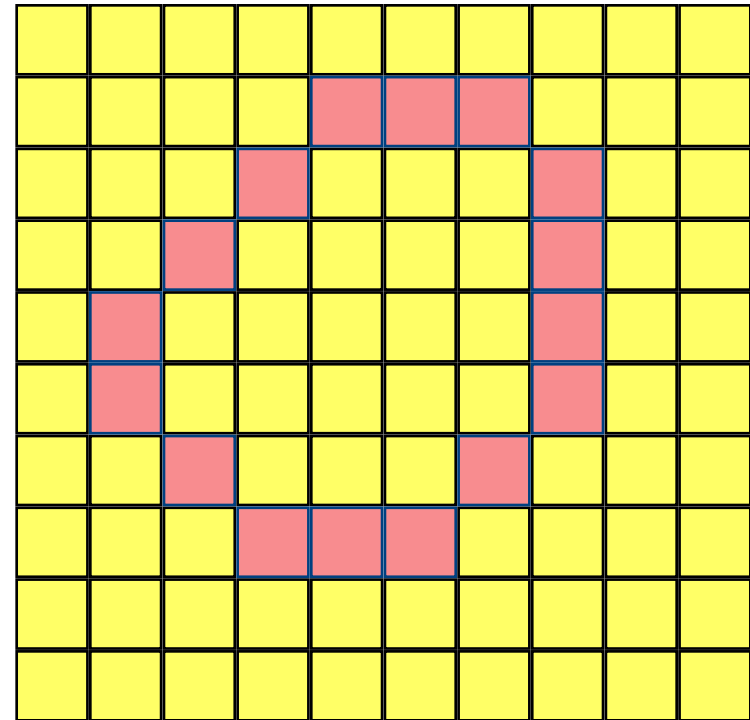
- ★ The neighbors of pixel p are all pixels that have unitary distance from p
- ★ The set of neighbors constitutes the neighborhood of p . Depending on the adopted metric we consider the $N_4(p)$ or $N_8(p)$ neighborhoods.  
- ★ A **path** of length n from p to q is a sequence of pixels $p=p_0, p_1, p_2, \dots, p_n=q$ that in the adopted metric has p_i neighbor of p_{i+1} , $1 \leq i \leq n$
- ★ A subset of F (or of F^*) is **connected** if for each couple of pixels in the subset exist a path between them entirely belonging to the subset.
- ★ The **contour** C of an image F is the subset of F having unitary distance from F^*

4 and 8 connectivity



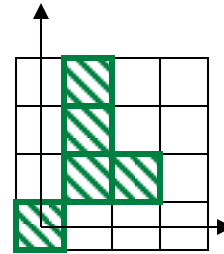
Topological Paradox

- ★ Contour in 8 connectivity
- ★ F and F* are separated only in 4 connectivity



Preliminary Statements

★ $A \subseteq E^n, t \in E^n$



A

★ Translation of A by a vector t

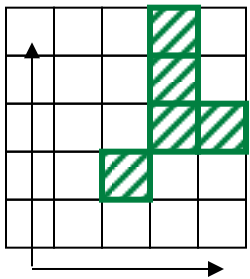
$$A_t = \{ c \in E^n \mid c = a + t, a \in A \}$$

★ Reflection of A

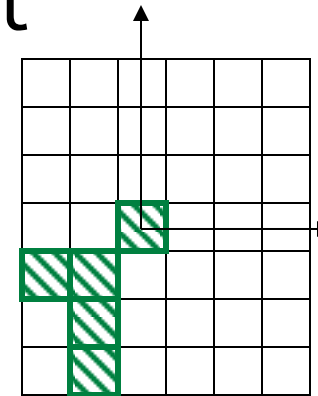
$$A_r = \{ c \mid c = -a, a \in A \}$$

★ Complement of A

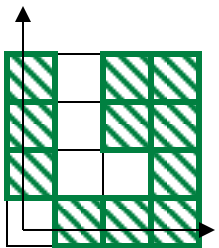
$$A_c = E^n - A$$



$A_{(2,1)}$



A^r



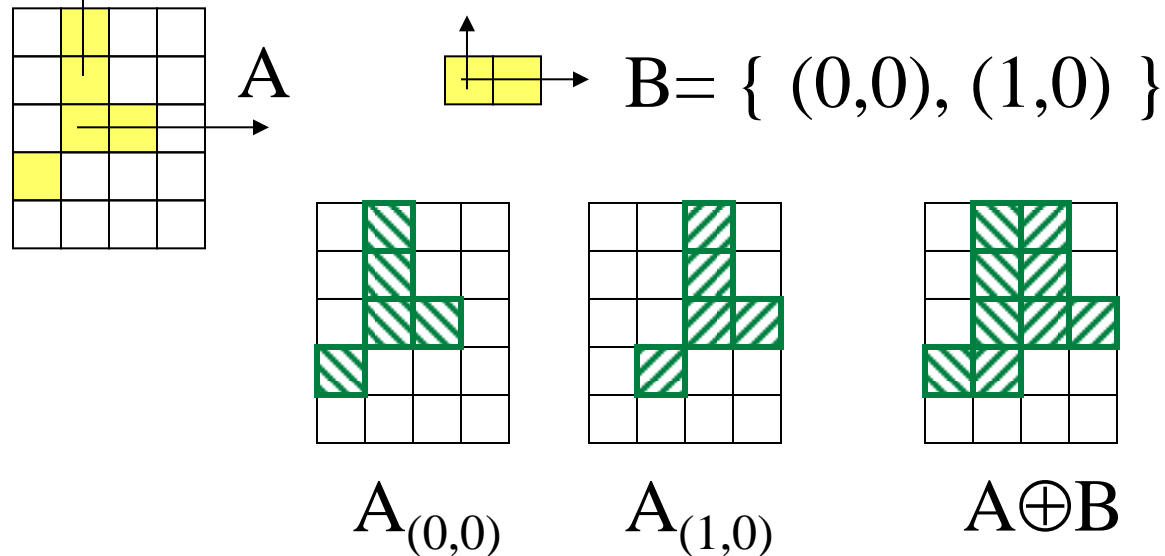
A^c

Minkowski sum (Dilation)

★ $A \oplus B = \{ c \in E^n \mid c = a + b, a \in A, b \in B \}$

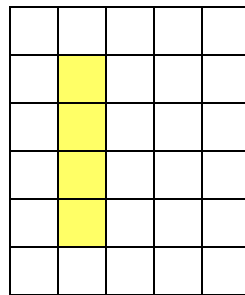
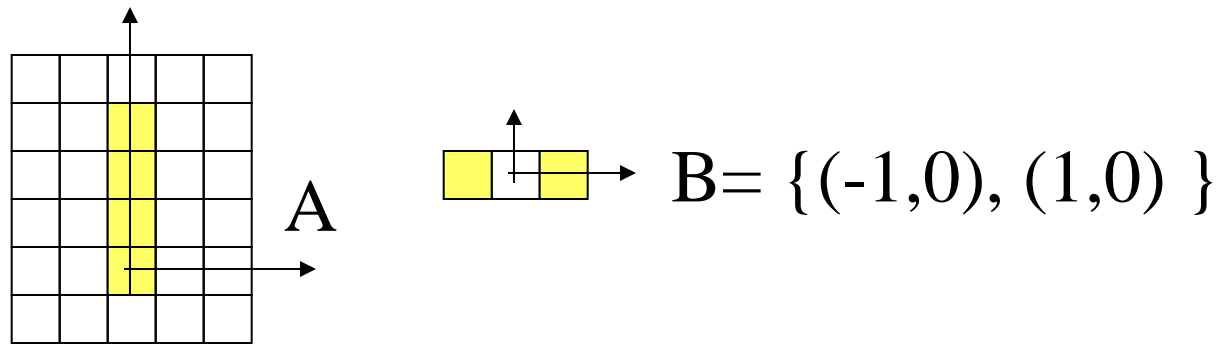
★ $A \oplus B = \cup_{b \in B} A_b$, $b \in B$

★ It can be easily shown that: $A \oplus B = B \oplus A$

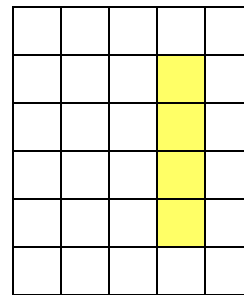


Dilation

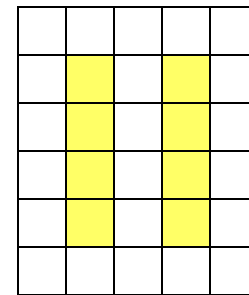
- ★ B is usually called *structural element*



$A_{(-1,0)}$

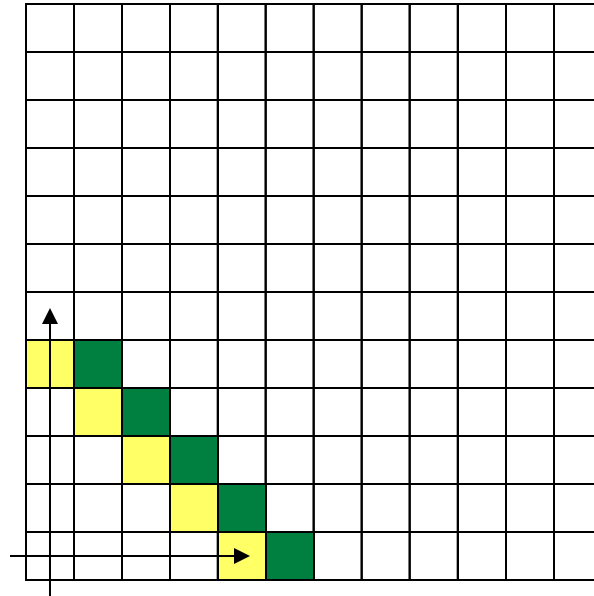


$A_{(1,0)}$

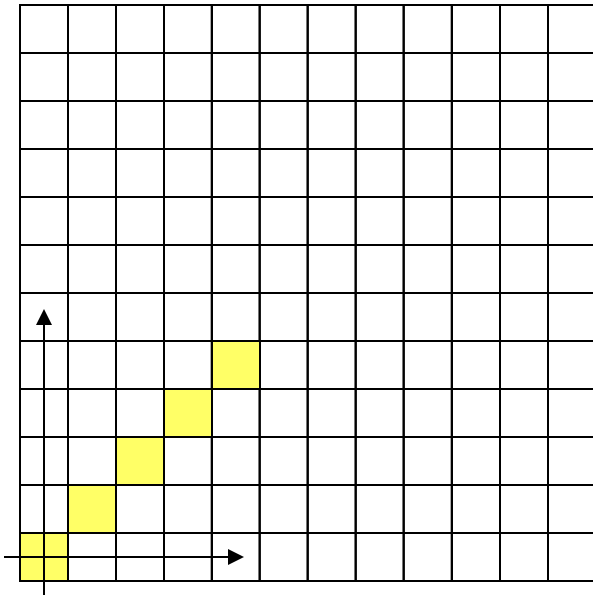


$A \oplus B$

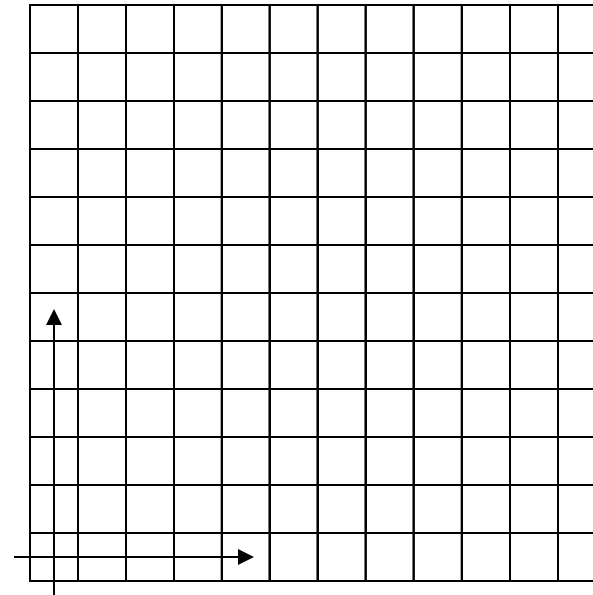
Dilation



B

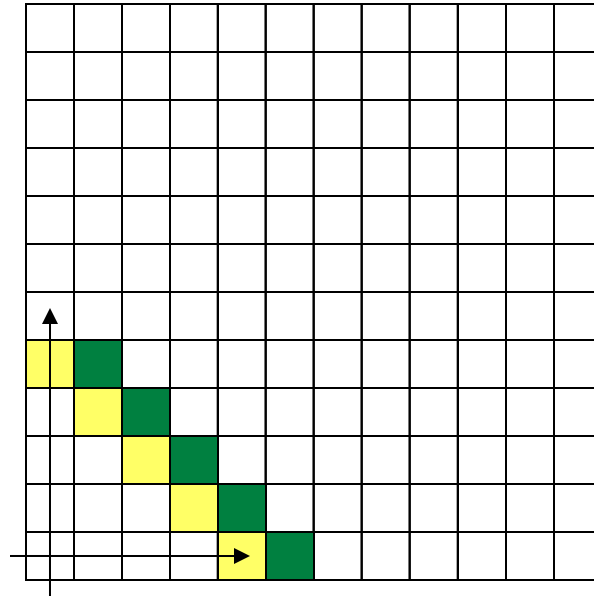


A

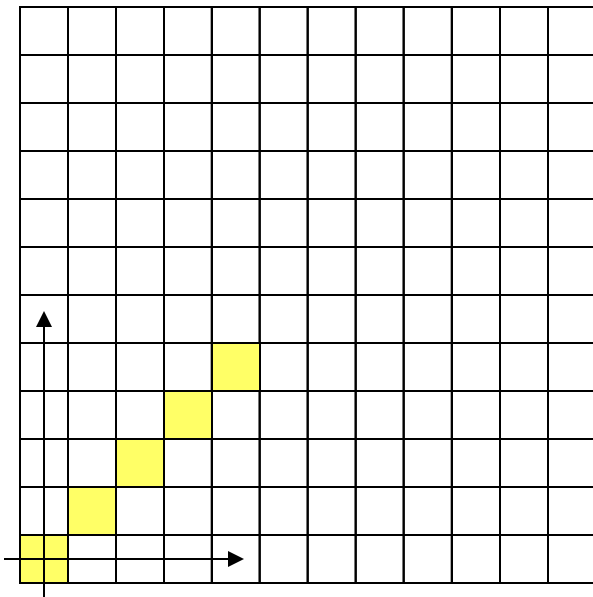


$A \oplus B$

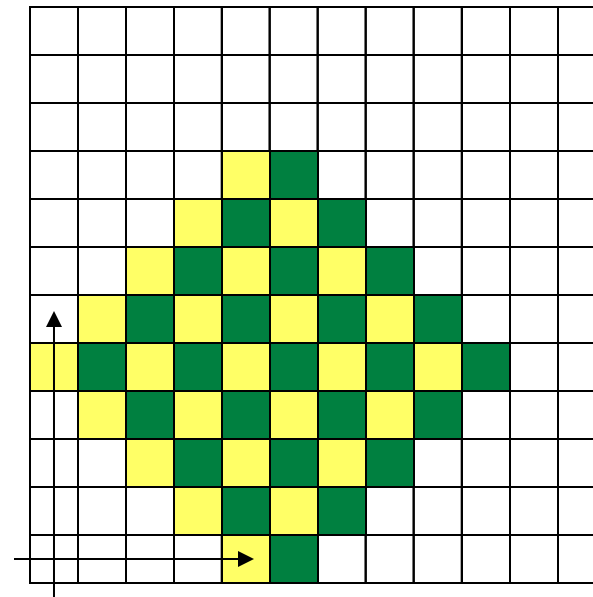
Dilation



B

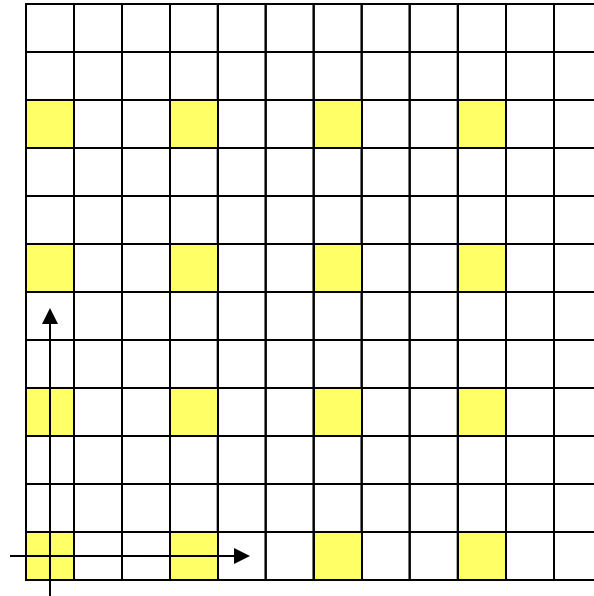


A

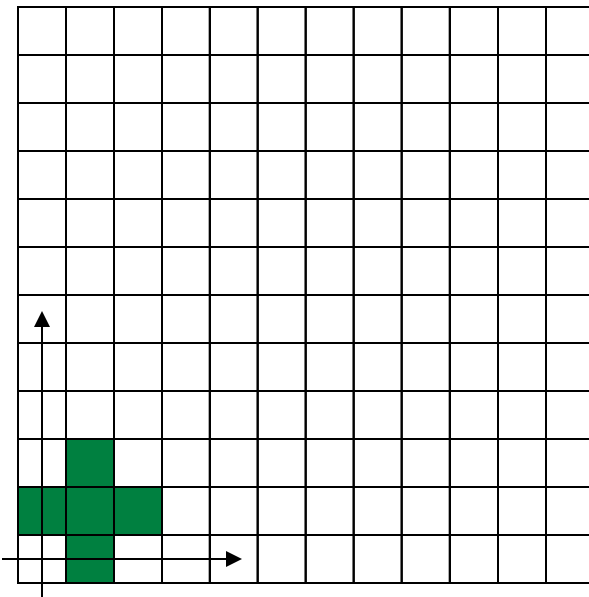


$A \oplus B$

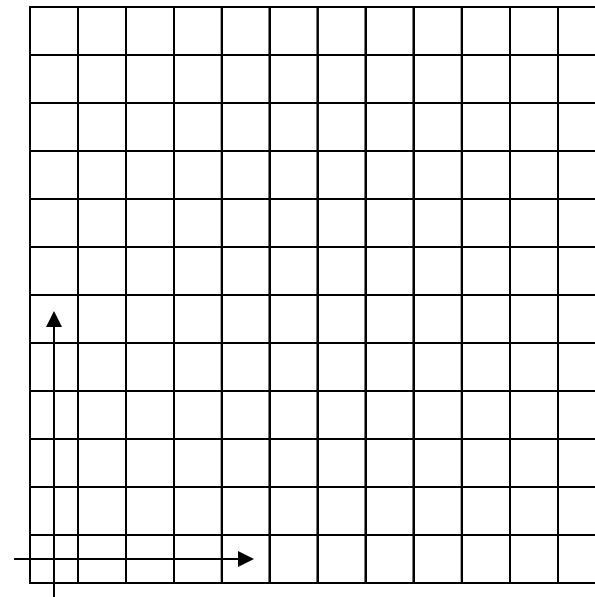
Dilation



A

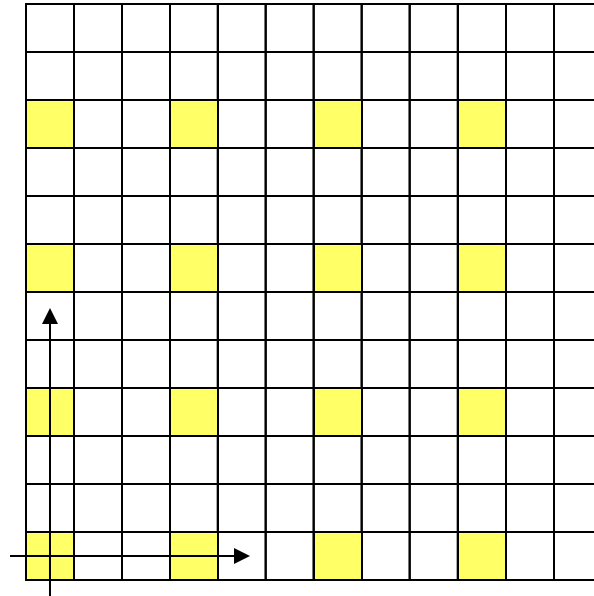


B

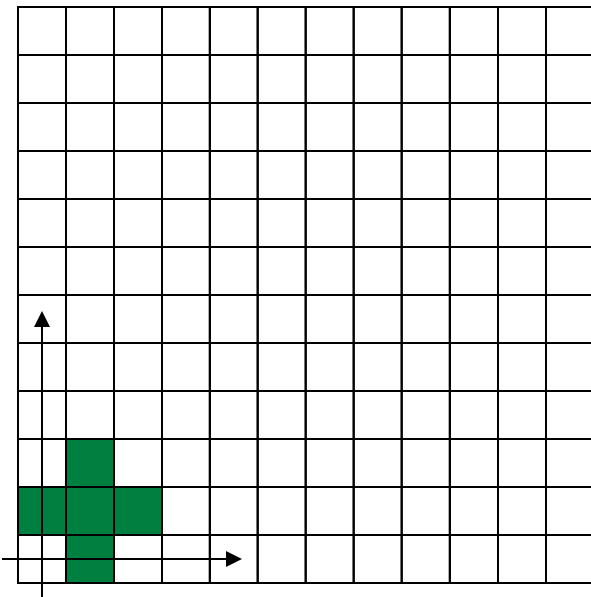


$A \oplus B$

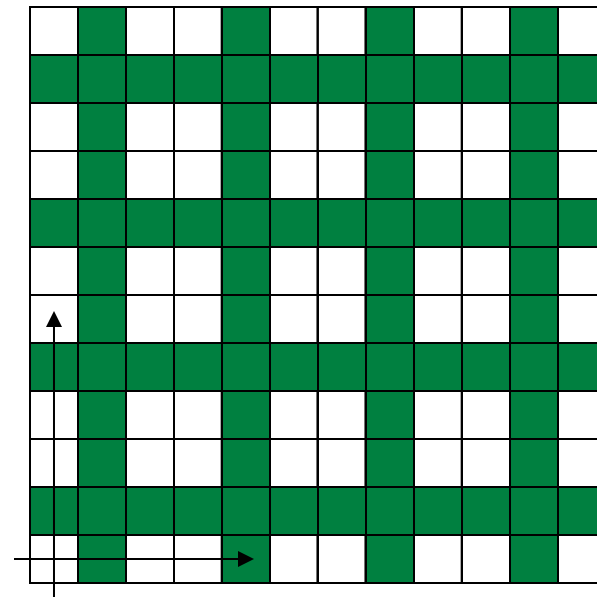
Dilation



A

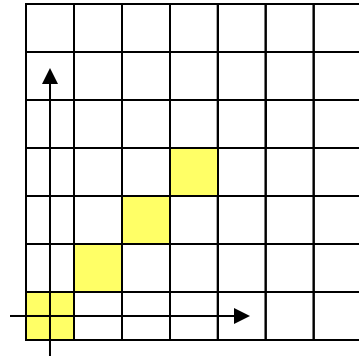


B

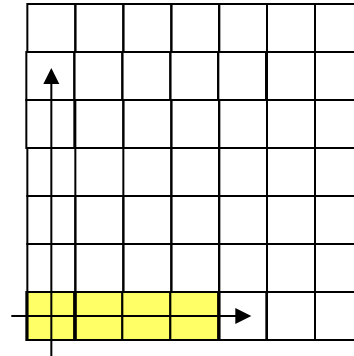


$A \oplus B$

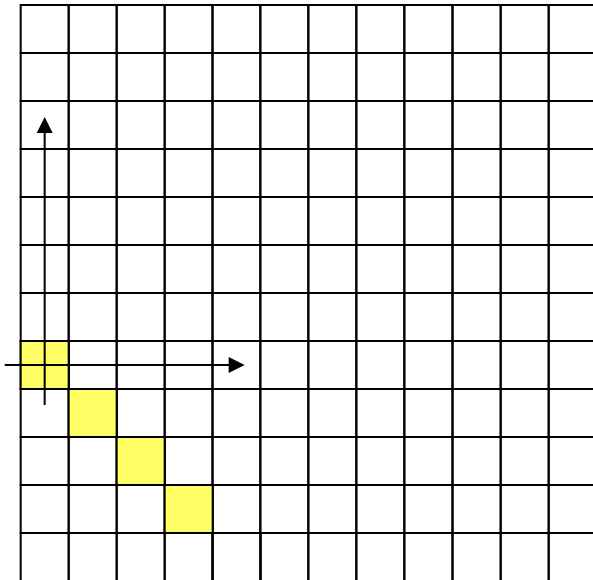
Dilation



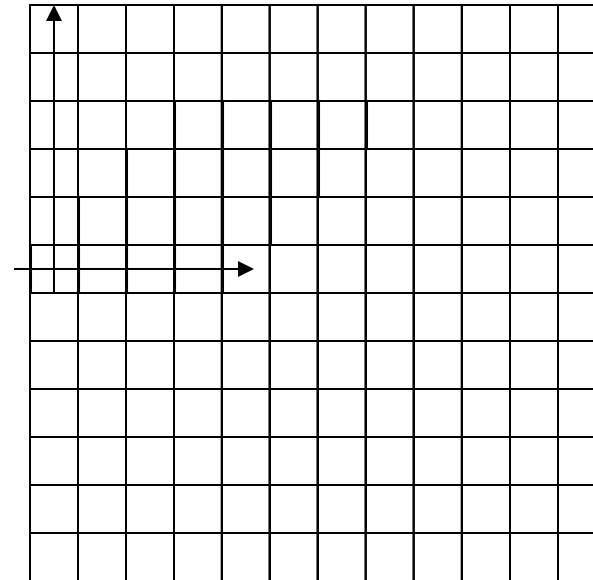
A



B

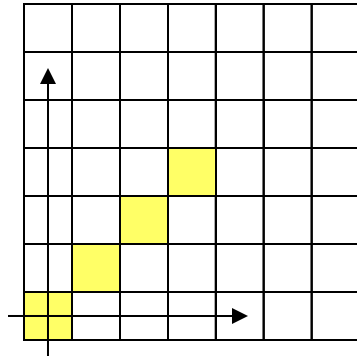


C

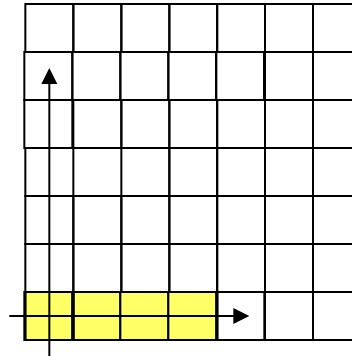


$A \oplus B \oplus C$

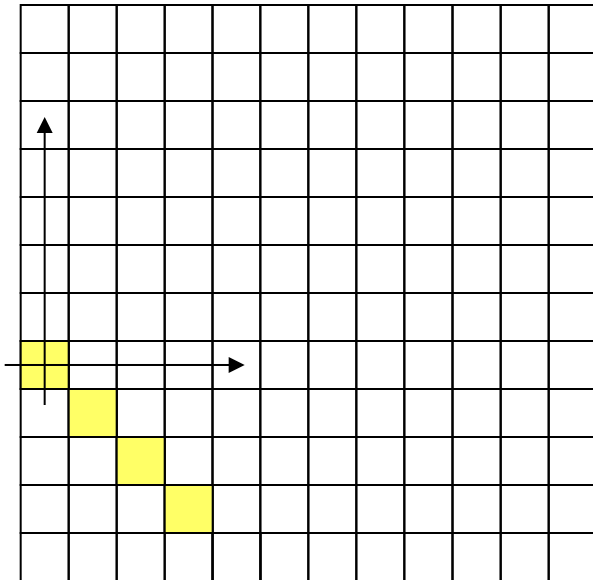
Dilation



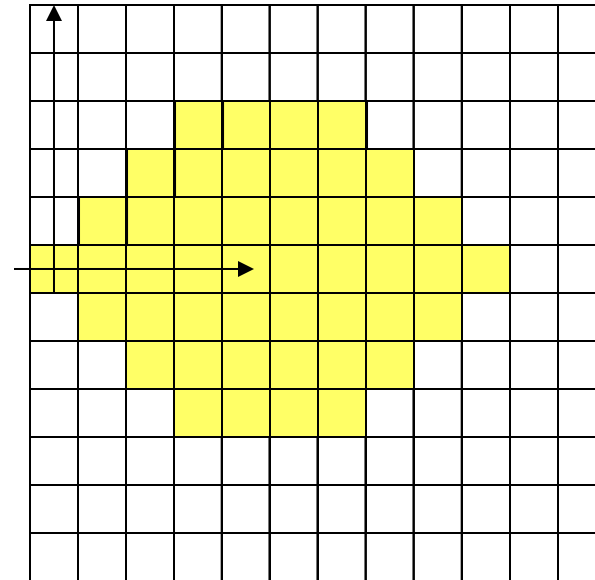
A



B



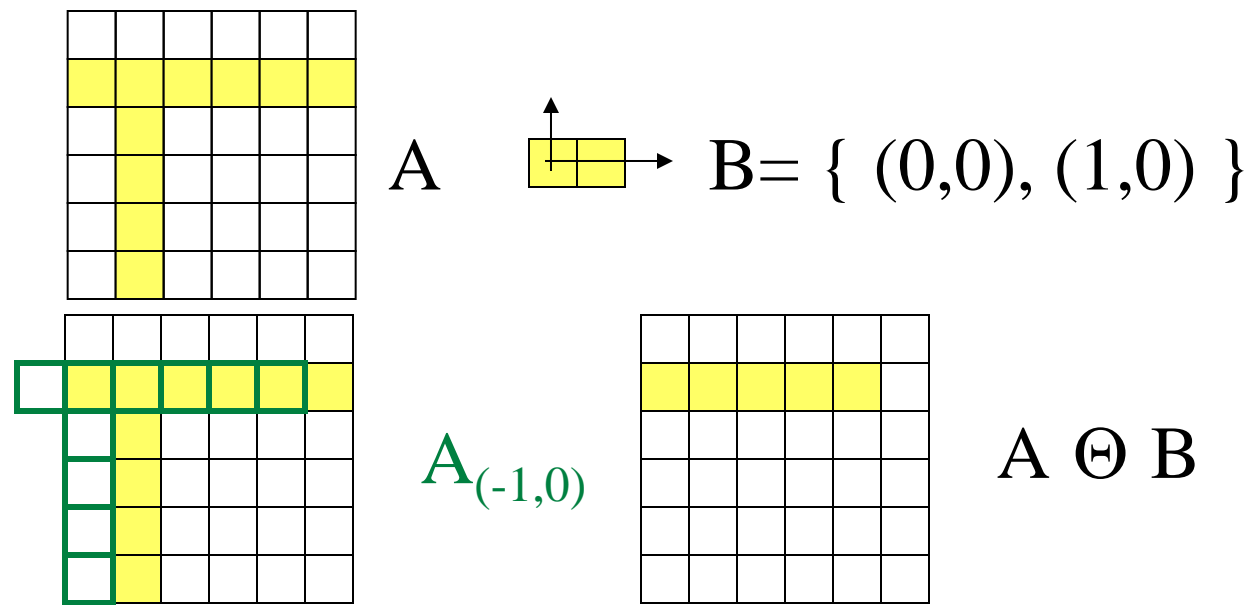
C



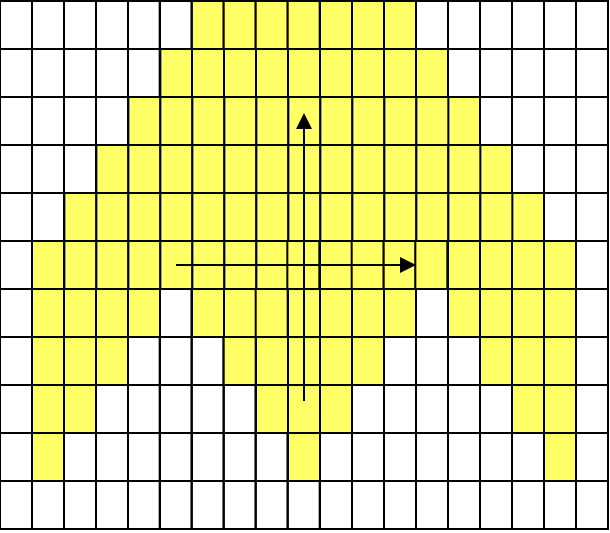
$A \oplus B \oplus C$

Minkowski difference (*Erosion*)

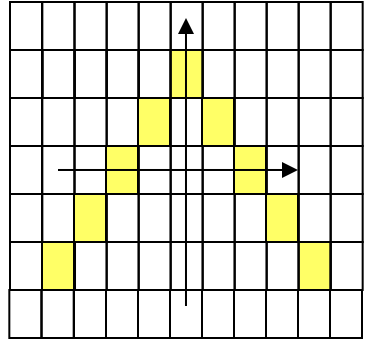
- ★ $A \ominus B = \{ c \in E^n \mid c + b \in A, \text{ per ogni } b \in B \}$
- ★ $A \ominus B = \bigcap_{b \in B} A_{-b}$
- ★ $A \ominus B = \{ c \in E^n \mid B_c \subset A \}$



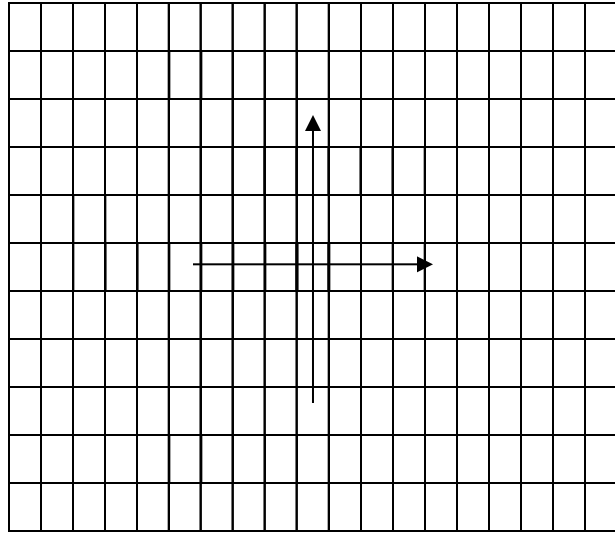
Erosion



A

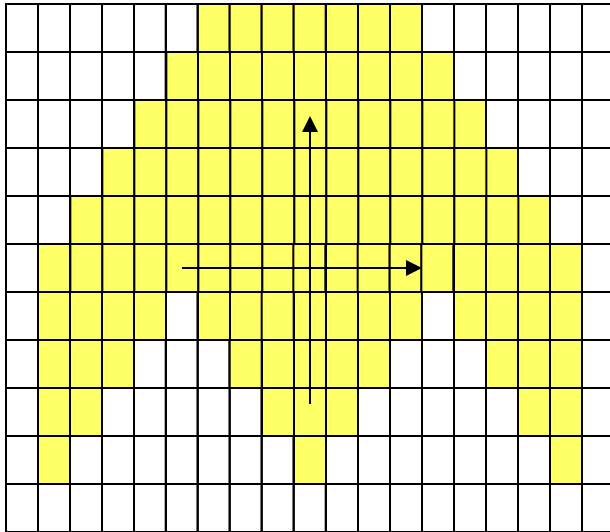


B

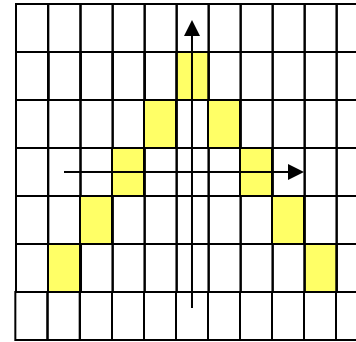


$A \ominus B$

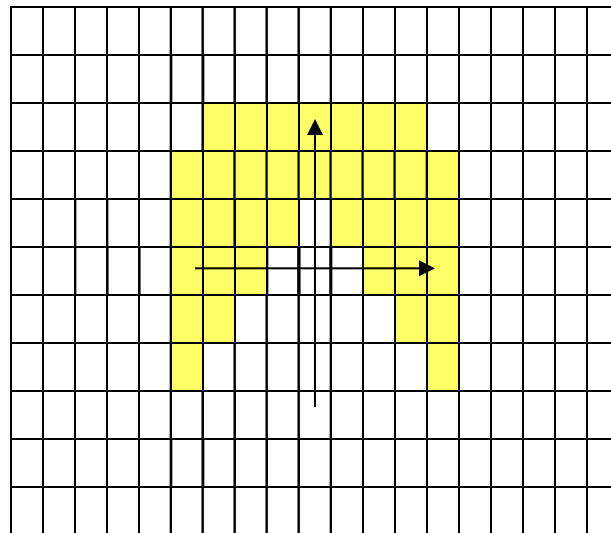
Erosion



A

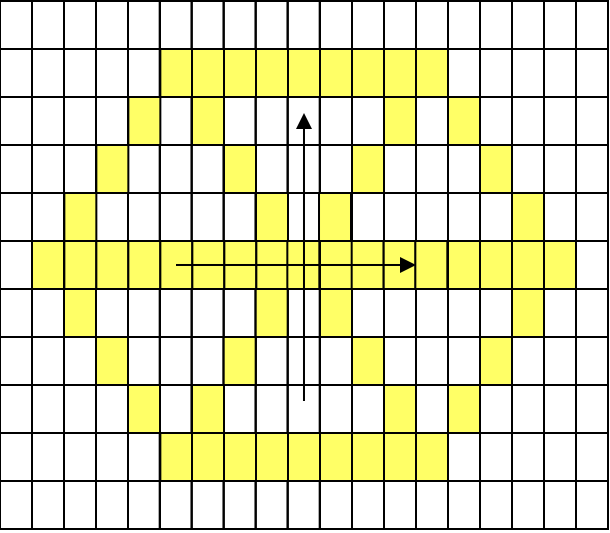


B

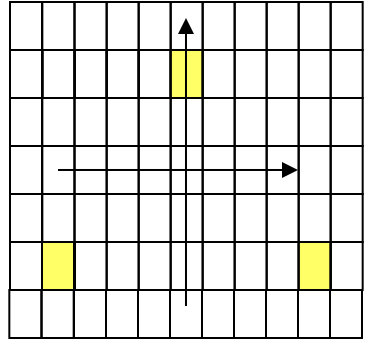


$A \ominus B$

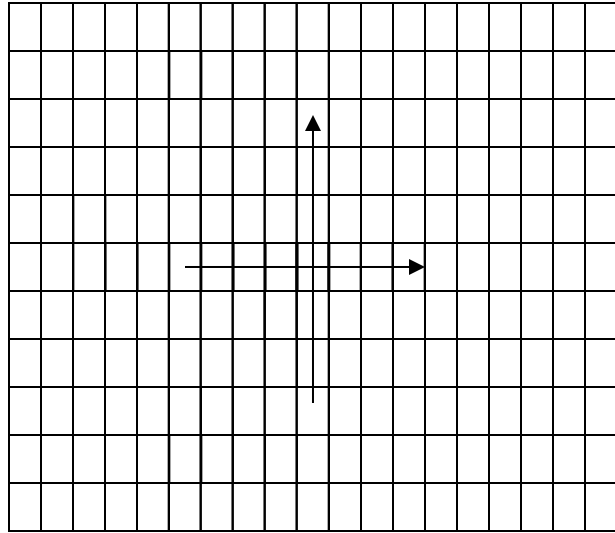
Erosion



A

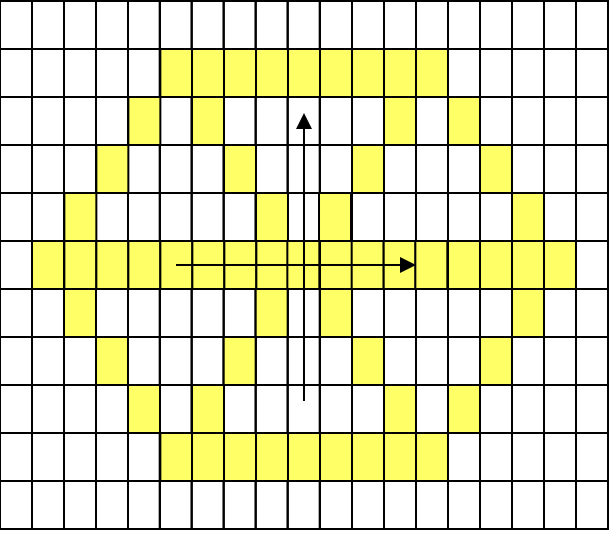


B

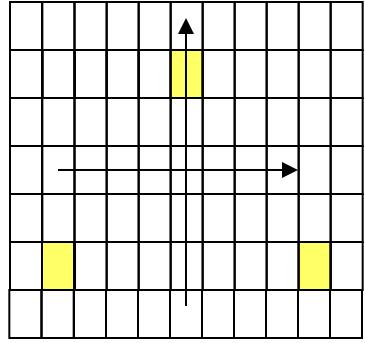


$A \ominus B$

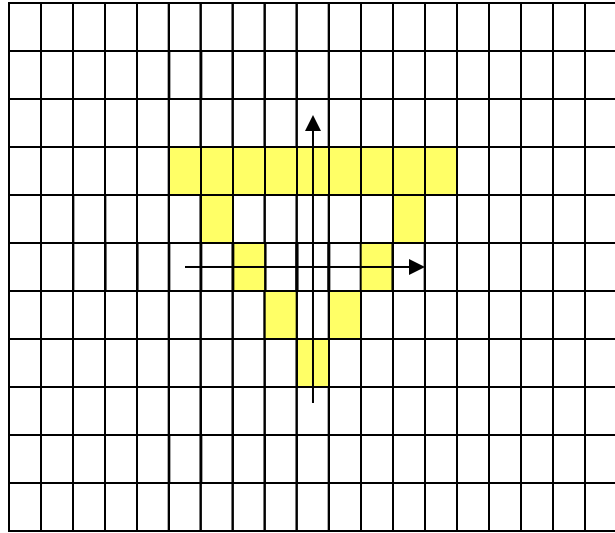
Erosion



A

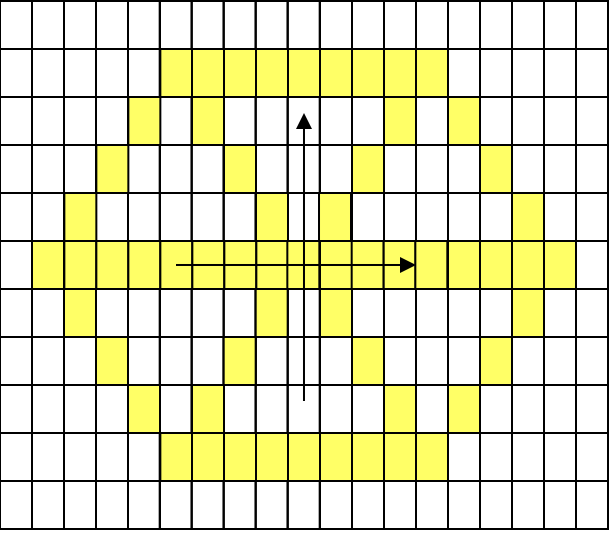


B

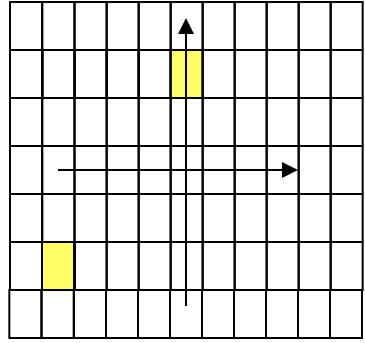


$A \ominus B$

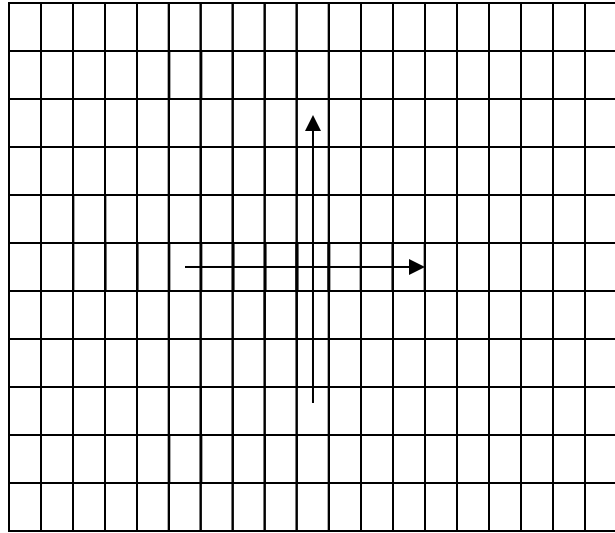
Erosion



A

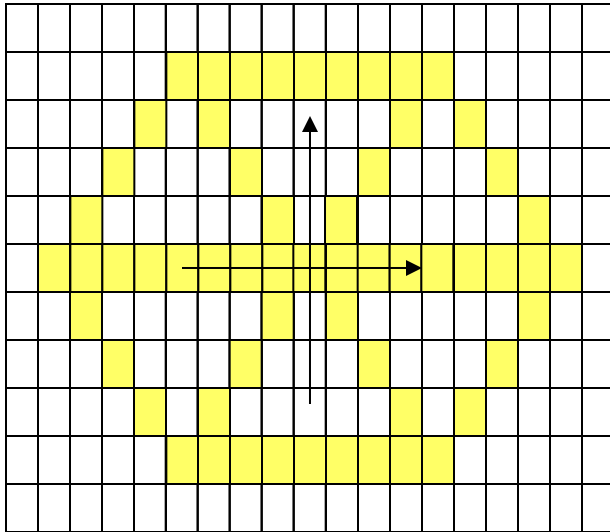


B

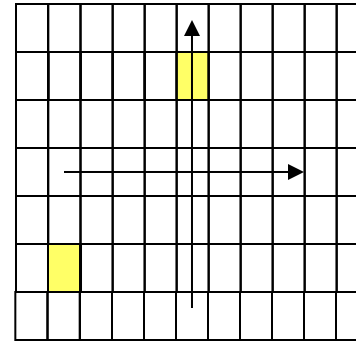


$A \ominus B$

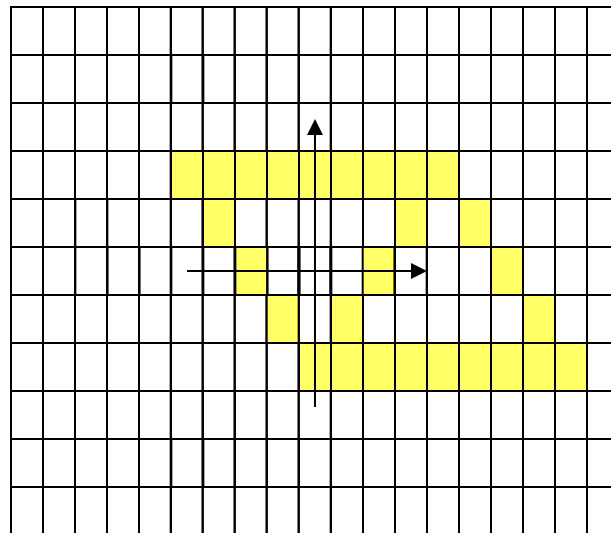
Erosion



A



B



$A \ominus B$

Dilation (+) and Erosion (-) properties

$$A + \{\emptyset\} = A - \{\emptyset\} = A$$

$$A + \{a\} = A - \{a\}^r = A_a, \text{ translation}$$

$$A + B = (A^c - B^r)^c$$

$$A - B = (A^c + B)^c$$

$$(A+B)^c = A^c - B^r$$

$$A+B_t = (A+B)_t$$

$$A-B_t = (A-B)_{-t}$$

Decomposition: $B=B_1+B_2+B_3+\dots+B_n$

$$A+B = (\dots(((A+ B_1)+B_2)+B_3) +\dots)+B_n$$

$$A-B = (\dots(((A- B_1)-B_2)-B_3) -\dots)-B_n$$

Dilation (+) and Erosion (-) properties

$$(A+B)+C=A+(B+C)$$

$$(A-B)-C=A-(B+C)$$

$$(A\cup B)+C=(A+C)\cup(B+C)$$

$$(A\cap B)-C=(A-C)\cap(B-C)$$

$$A+(B\cup C)=(A+B)\cup(A+C)$$

$$A-(B\cup C)=(A-C)\cap(B-C)$$

$$A\subseteq B\Rightarrow(A+C)\subseteq(B+C)$$

$$A\subseteq B\Rightarrow(A-C)\subseteq(B-C)$$

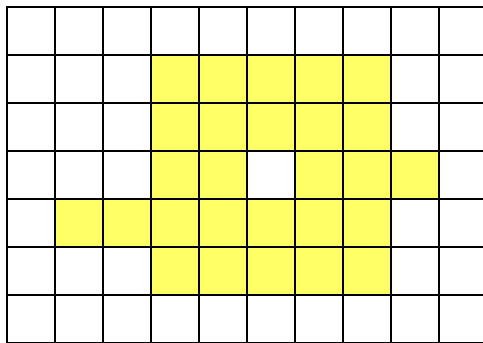
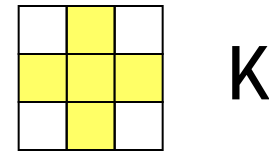
$$B\subseteq C\Rightarrow(A-B)\supseteq(A-C)$$

$$(A\cap B)+C\subseteq(A+C)\cap(B+C) \quad (A\cup B)-C\supseteq(A-C)\cup(B-C)$$

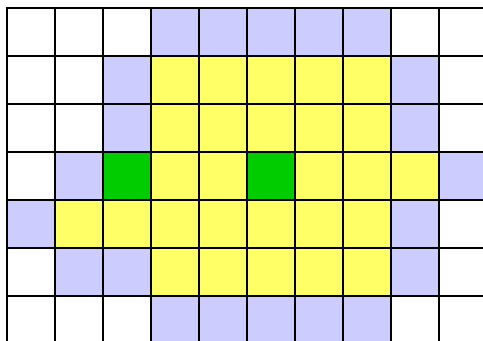
$$A-(B\cap C)\supseteq(A-C)\cup(B-C)$$

Closing operator

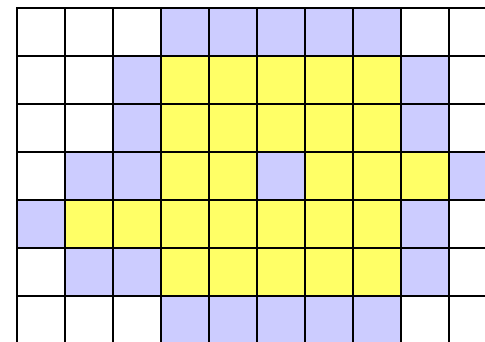
- ★ $C(A, K) = (A+K)-K$
- ★ $A \subseteq C(A, K) = C(C(A, K), K)$



A



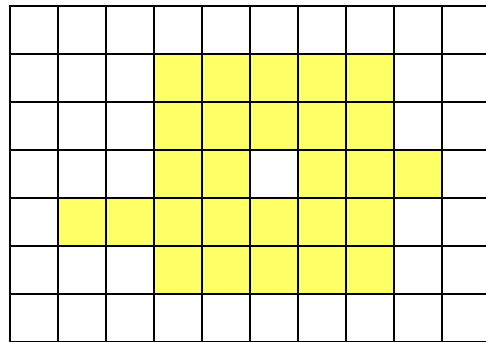
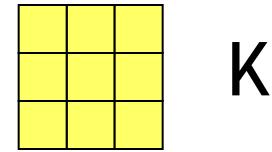
$(A+K)-K$



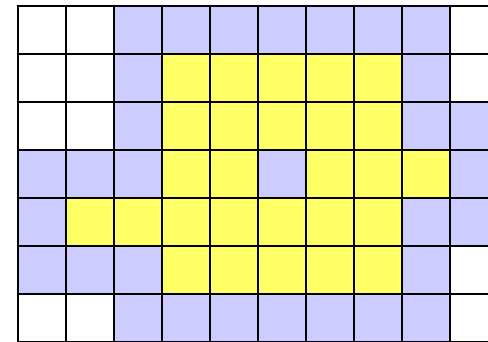
A+K

Closing operator

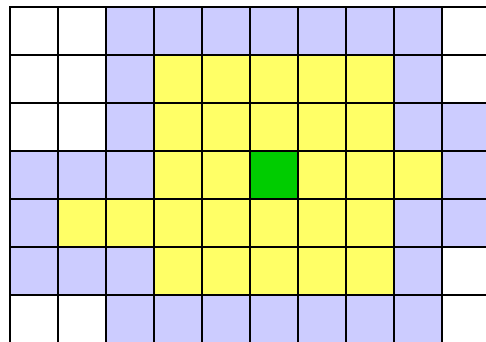
★ $C(A, K) = (A+K)-K$



A



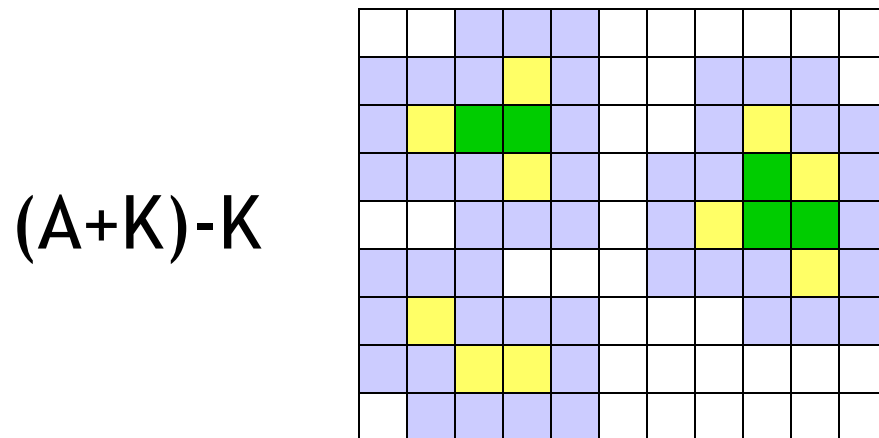
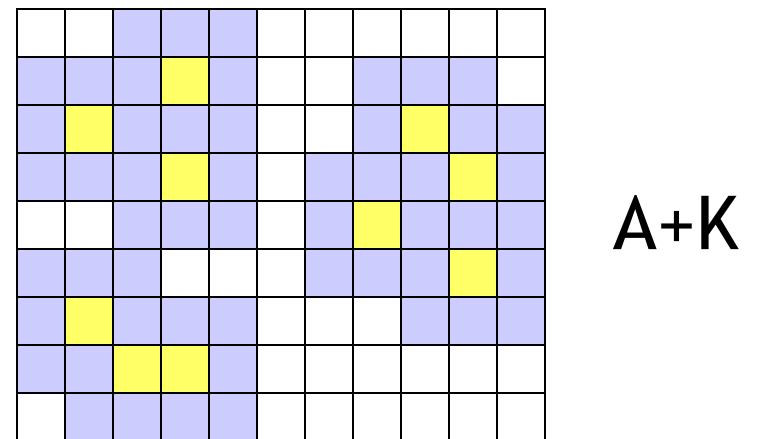
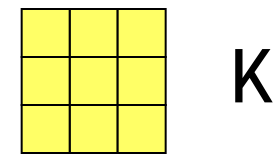
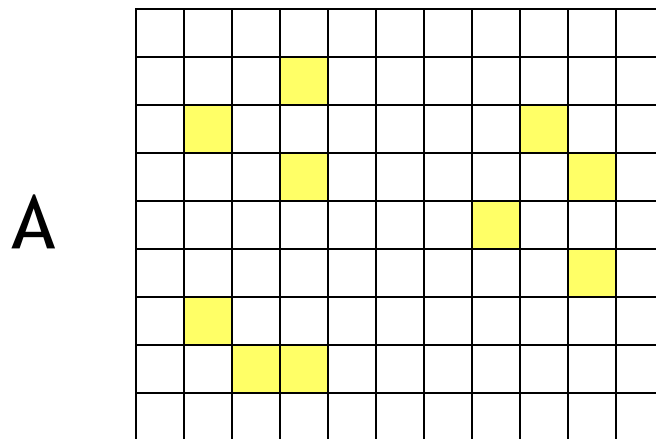
A+K



(A+K)-K

Closing operator

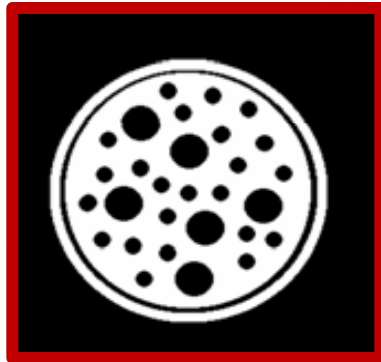
★ $C(A, K) = (A+K)-K$



Closing

- ✦ Dilate, then erode
- ✦ Fill holes, but keep original shape

Structural
element: •



Before closing

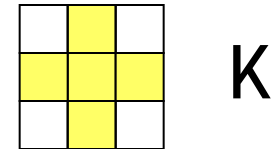
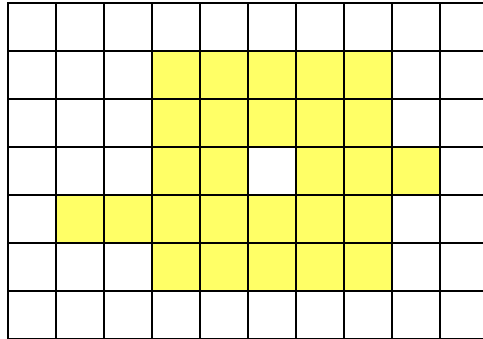
After closing

Opening operator

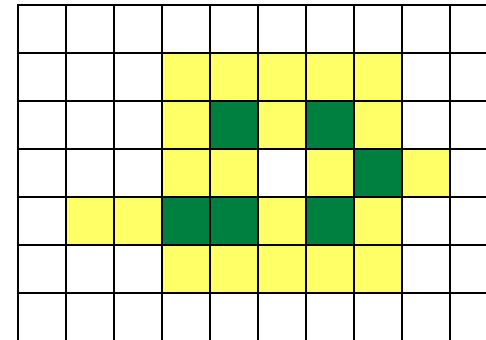
★ $O(A, K) = (A-K)+K$

★ $O(O(A,K),K)=O(A,K)\subseteq A$

A

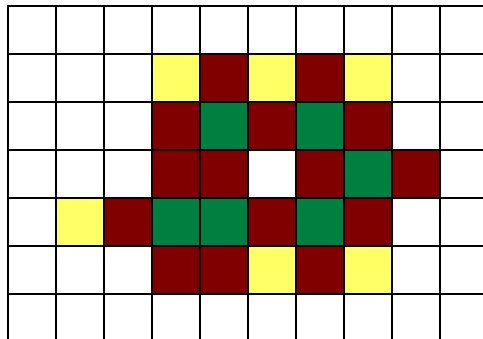


K



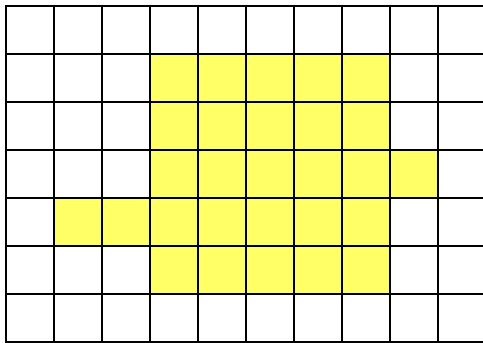
A-K

$(A-K)+K$

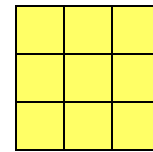


Opening operator

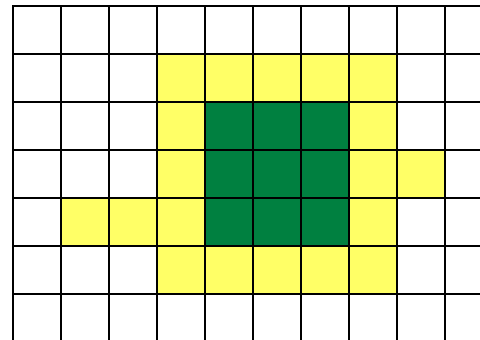
★ $O(A, K) = (A-K)+K$



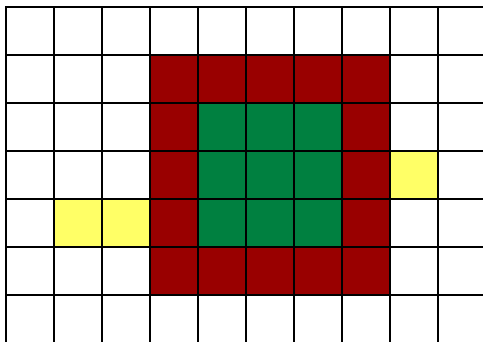
A



K



A-K



$(A-K)+K$

Opening

- ✦ Erode, then dilate
- ✦ Remove small objects, keep original shape

Structural
element: •



Before opening



After opening

Opening and Closing properties

- ★ F and F* can be changed changing the two operators:

$$(A \circ B)^c = A^c \bullet B$$

$$(A \bullet B)^c = A^c \circ B$$

- ★ Idempotency:

$$(A \circ B) \circ B = A \circ B$$

$$(A \bullet B) \bullet B = A \bullet B$$

- ★ Monotonicity:

$$A \subseteq B \Rightarrow A \circ C \subseteq B \circ C$$

$$A \subseteq B \Rightarrow A \bullet C \subseteq B \bullet C$$

Examples

★ Propagation

$$C = \{ X \};$$

$$\text{do } D = C$$

$$C = (C \oplus K) \cap F$$

$$\text{while}(D \neq C)$$

C = evolving image

F = original image

★ Contour

★ Internal: $A - (A \ominus K)$

★ External: $(A \oplus K) \cap \bar{A}$ or $(A \oplus K) - A$

★ Double: $(A \oplus K) \cap \overline{(A \ominus K)} = (A \oplus K) - (A \ominus K)$

Hit or Miss operator

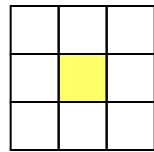
★ $A \otimes (J, K) = (A - J) \cap (A^c - K)$
con il vincolo $J \cap K = \emptyset$

★ Suitable for 'template' matching

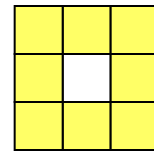
Hit or Miss

★ Search of isolated points(8-connection)

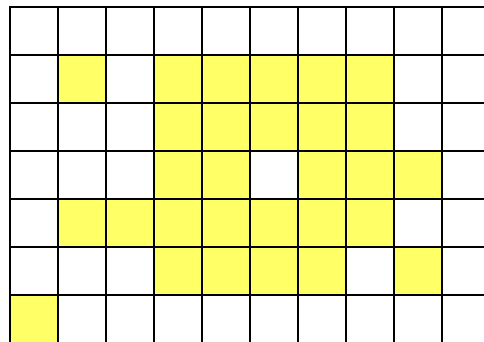
★ $A - J = A$



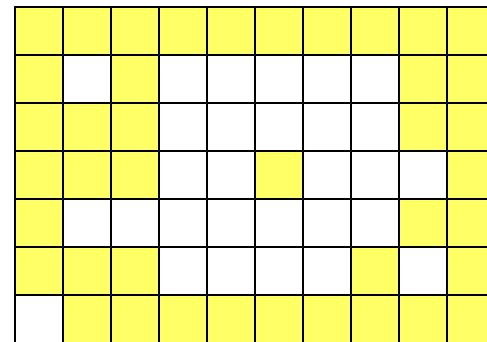
J



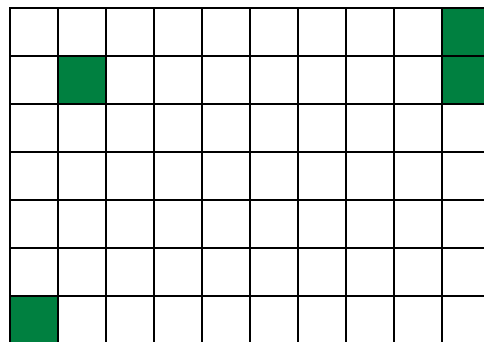
K



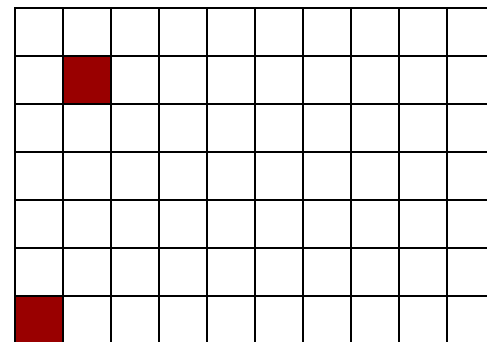
A



A^c



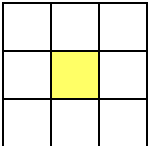
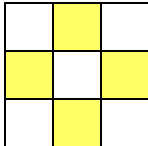
$A^c - K$

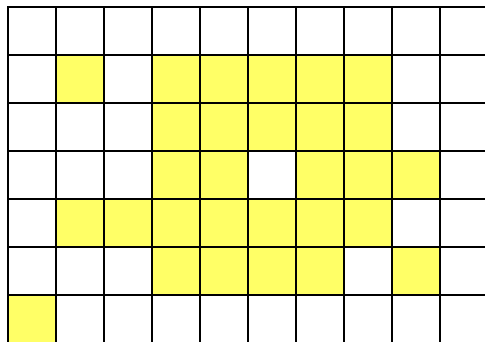


Final
Result

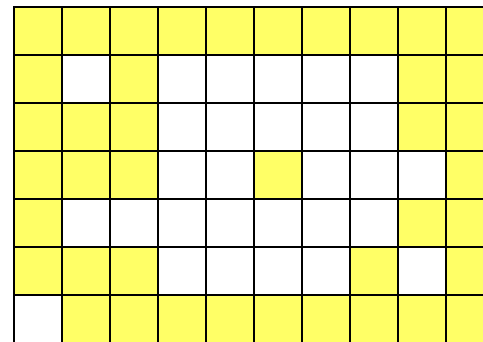
Hit or Miss

★ Search of isolated points(4-connection)

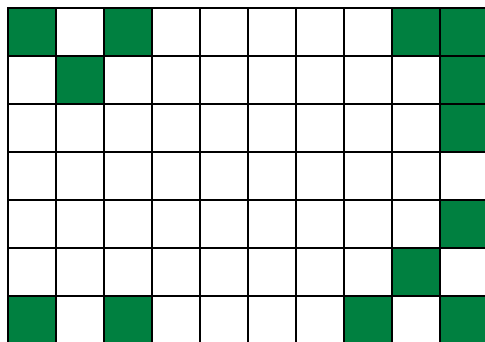
★ A-J=A  J  K



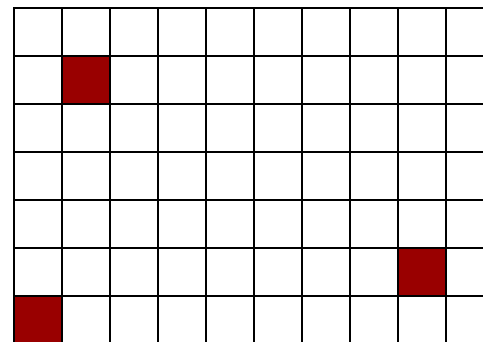
A



A^c

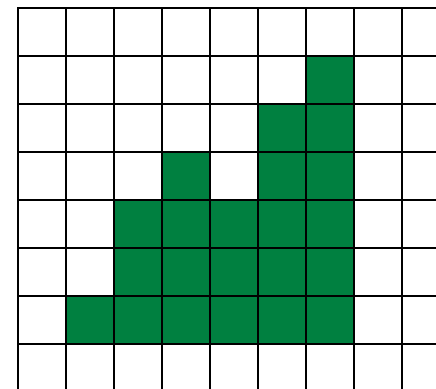
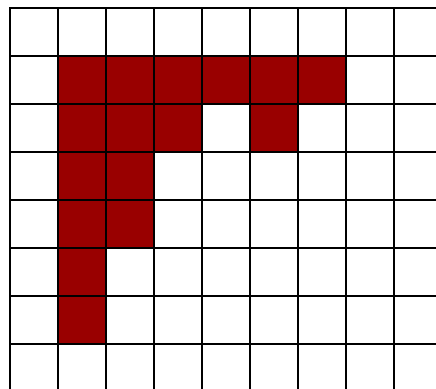
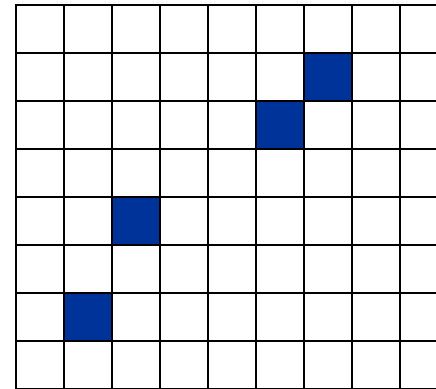
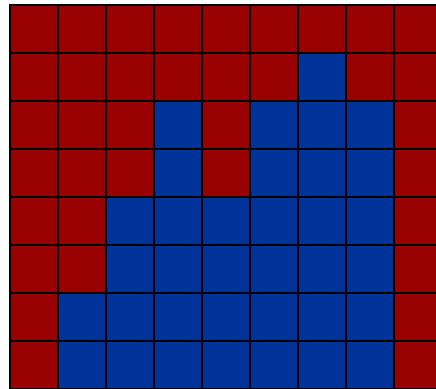
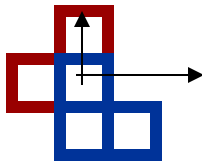


A^c-K



Final
Result

Hit or Miss

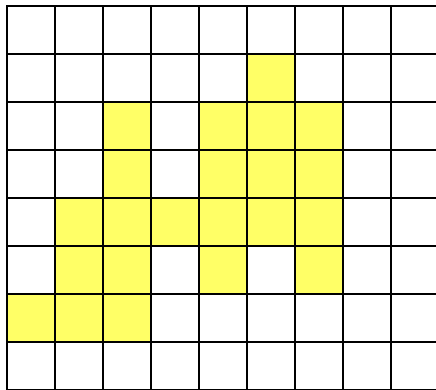


Pixels satisfying the background constraints

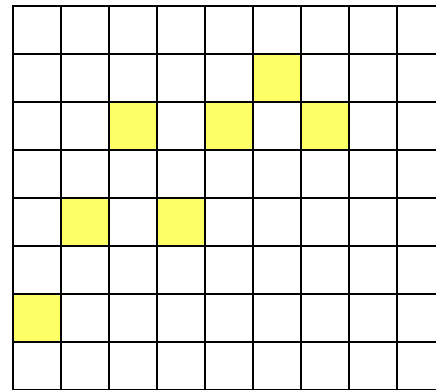
Pixels satisfying the foreground constraints

Umbral

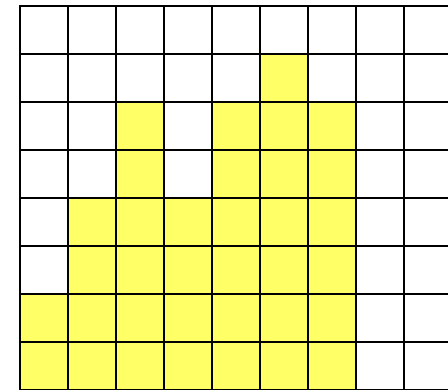
- ★ This is an extension to multidimensional ‘images’ in particular to grey level and color images
- ★ $A \subseteq E^n$, $F \subseteq E^{n-1}$, $x \in F$, $y \in E$
- ★ Top of a set A:
$$T[A](x) = \max \{ y \mid (x, y) \in A \}$$
- ★ Umbral of f:
$$U[f] = \{ (x, y) \in F \cup E \mid y \leq f(x) \}$$



Set A

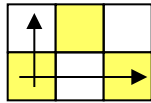


Top of A

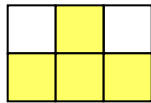


Umbral of A

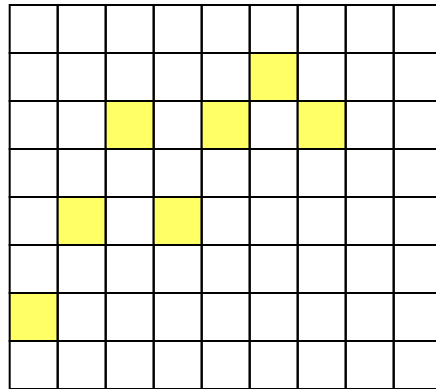
Dilation - Example



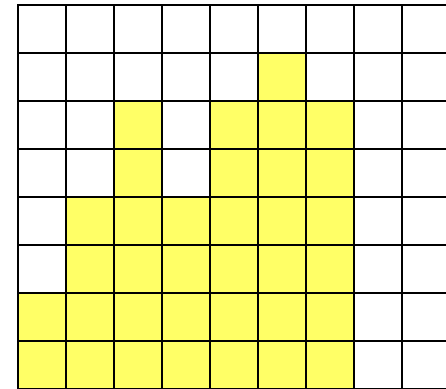
k



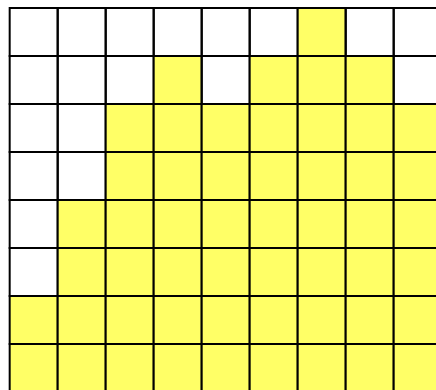
U[k]



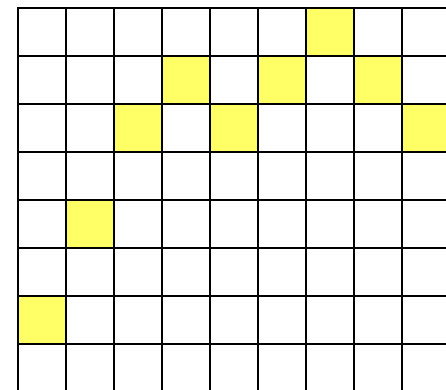
f



U[f]

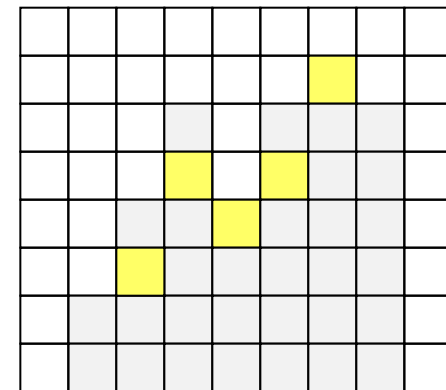
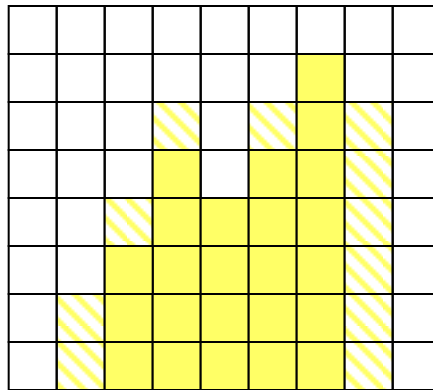
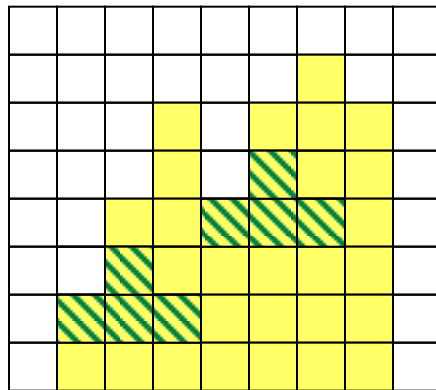
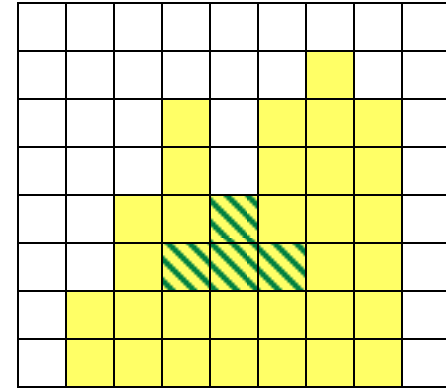
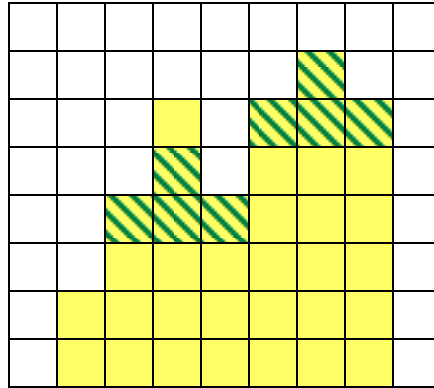
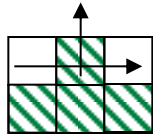


U[f] ⊕ U[k]



$f \oplus k = T[U[f] \oplus U[k]]$

Erosion - Example



$$U[f] \ominus U[k]$$

$$T\{U[f] \ominus U[k]\}$$

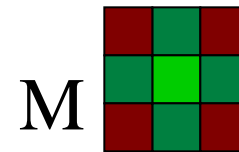
Hit or Miss

✦ J and K can be seen as a single template with three values:

✦ Image points

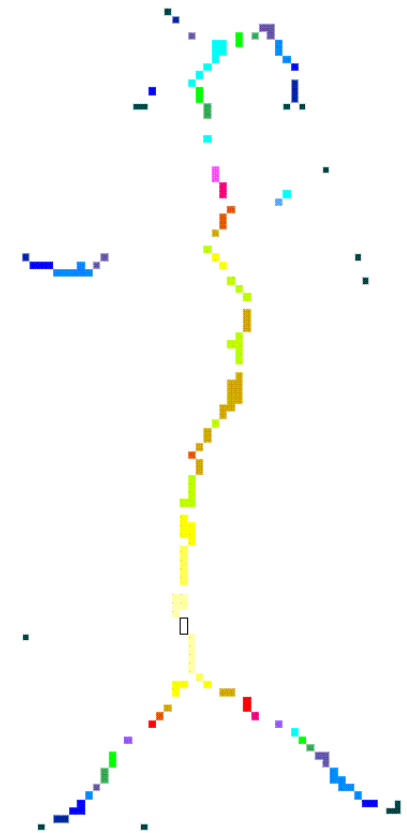
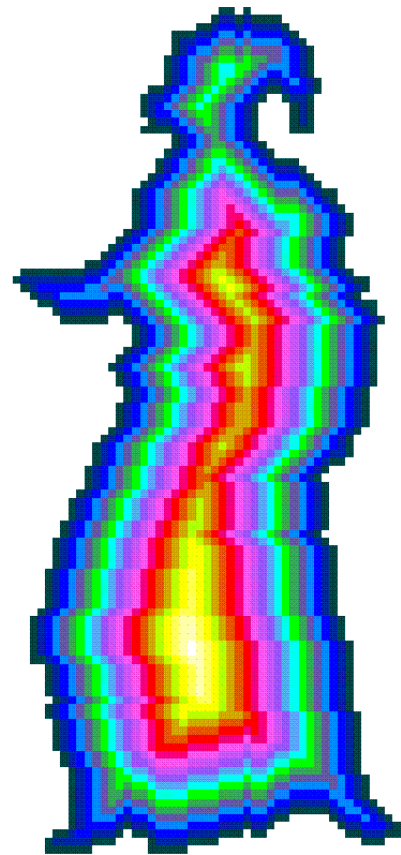
✦ Background points

✦ Do not care points



Distance transform and MAT

- ✦ The **Distance Transform (DT)** is obtained by labeling all the pixels inside a binary object with their distance to the background
- ✦ Applying twenty iterations of the erosion operator (structural element: unit disk) twenty successive colored layers showing equi-distant contours from the background for a Manhattan distance metric are obtained
- ✦ Every pixel has a color corresponding to its distance label which increases going inwards. In practice, this value represents the side of the greatest digital disk having its centre on this pixel, which is completely contained in the binary object.
- ✦ Any pattern can be interpreted as the union of all *its maximal digital disks (local maximum in DT)*. A maximal disk is a disk contained in the object that is not completely overlapped by any other disk.
- ✦ The set of the centers of the maximal disks with their labels, constitutes the **MAT**



Distance transform

- ★ DT implementation using dilation and addition operators:

$R = \emptyset$

while($A \neq \emptyset$) do

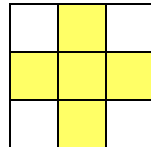
$R = R + A$

$A = A \ominus K$

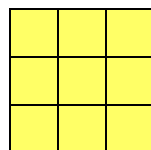
done

$R =$ evolving image
at the end DT

Distance transform



			1	1	1	1	1		
		1	2	2	2	2	2	1	
		1	2	3	3	3	2	1	
	1	2	3	4	4	4	3	2	1
1	2	2	3	3	3	3	2	1	
	1	1	2	2	2	2	2	1	
			1	1	1	1	1		

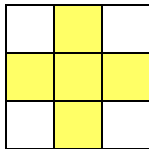


			1	1	1	1	1		
		1	1	2	2	2	1	1	
		1	2	2	3	2	2	1	
	1	1	2	3	3	3	2	1	1
1	1	2	2	2	3	2	2	1	
	1	1	1	2	2	2	1	1	
			1	1	1	1	1		

DT - sequential algorithm

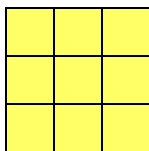
- ★ Two image scan:
 - ★ First top-down and left-right
 - ★ Foreground pixels are assigned to the minimum value among the **neighbors** already assigned +1 (background pixels are pre-set to 0)
 - ★ Second bottom-up and right-left
 - ★ Foreground pixels are assigned to the minimum value among the **neighbors** already assigned + 1 and the value of the pixel itself

DT - sequential algorithm



			1	1	1	1	1		
		1	2	2	2	2	2	1	
		1	2	3	3	3	3	2	
	1	2	3	4	4	4	4	2	1
1	2	3	4	5	5	5	3	2	
	1	2	3	4	5	6	4	3	
			1	2	3	4	5		

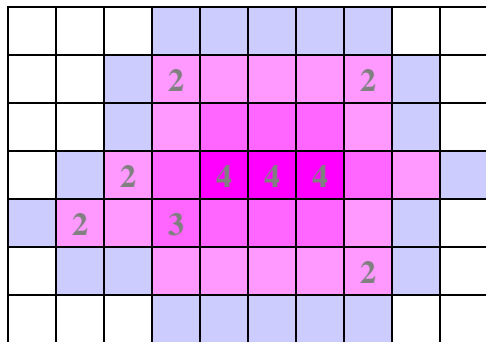
			1	1	1	1	1		
		1	2	2	2	2	2	1	
		1	2	3	3	3	2	1	
	1	2	3	4	4	4	3	2	1
1	2	2	3	3	3	3	2	1	
	1	1	2	2	2	2	2	1	
			1	1	1	1	1		



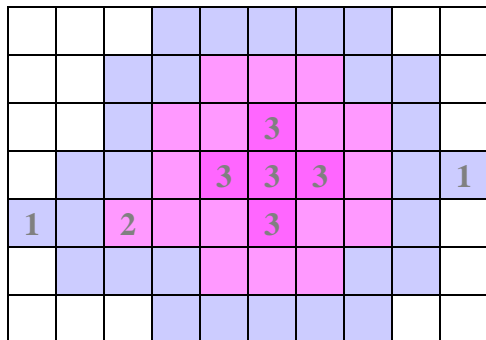
			1	1	1	1	1		
		1	1	2	2	2	1	1	
		1	2	2	3	2	2	1	
	1	1	2	3	3	3	2	1	1
1	1	2	2	3	4	3	2	2	
	1	2	3	3	4	3	3	1	
			1	2	3	4	4		

			1	1	1	1	1		
		1	1	2	2	2	1	1	
		1	2	2	3	2	2	1	
	1	1	2	3	3	3	2	1	1
1	1	2	2	2	3	2	2	1	
	1	1	1	2	2	2	1	1	
			1	1	1	1	1		

DT - local maxima

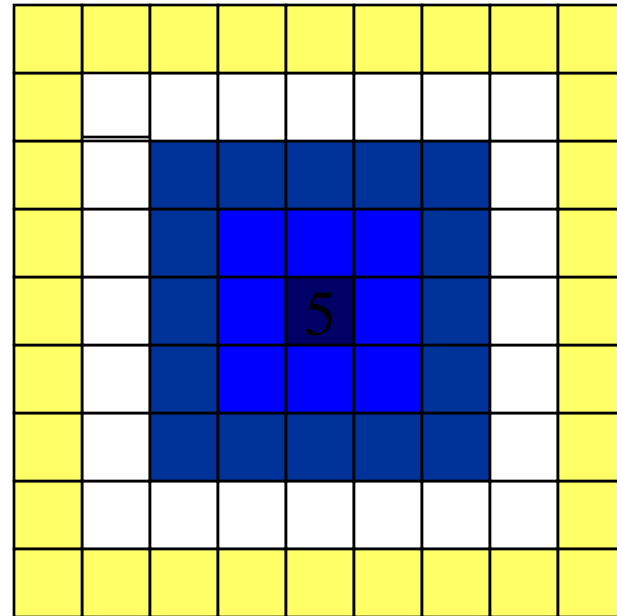
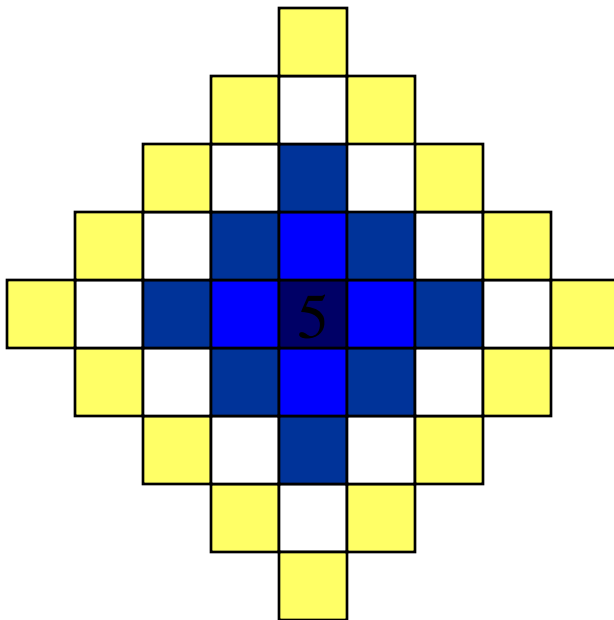


- ★ The local maxima set is a compact object representation

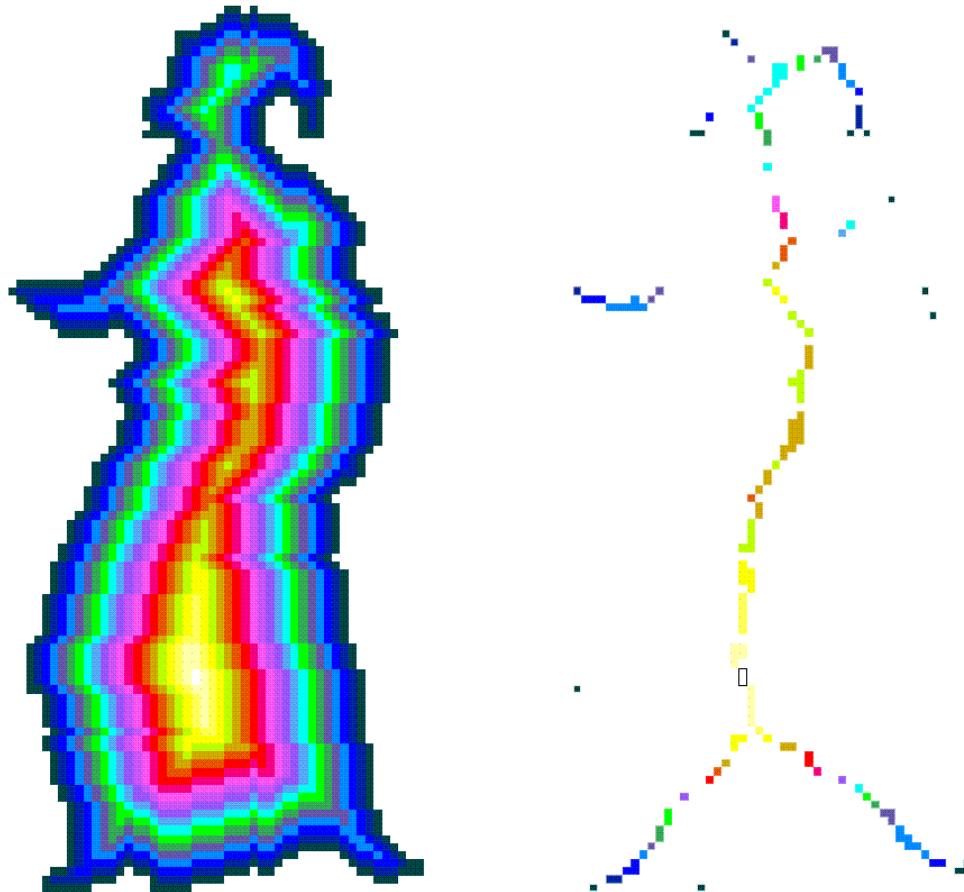


- ★ The object can be rebuilt as union of the *maximal digital disks*

Disks in 4 and 8 connectivity

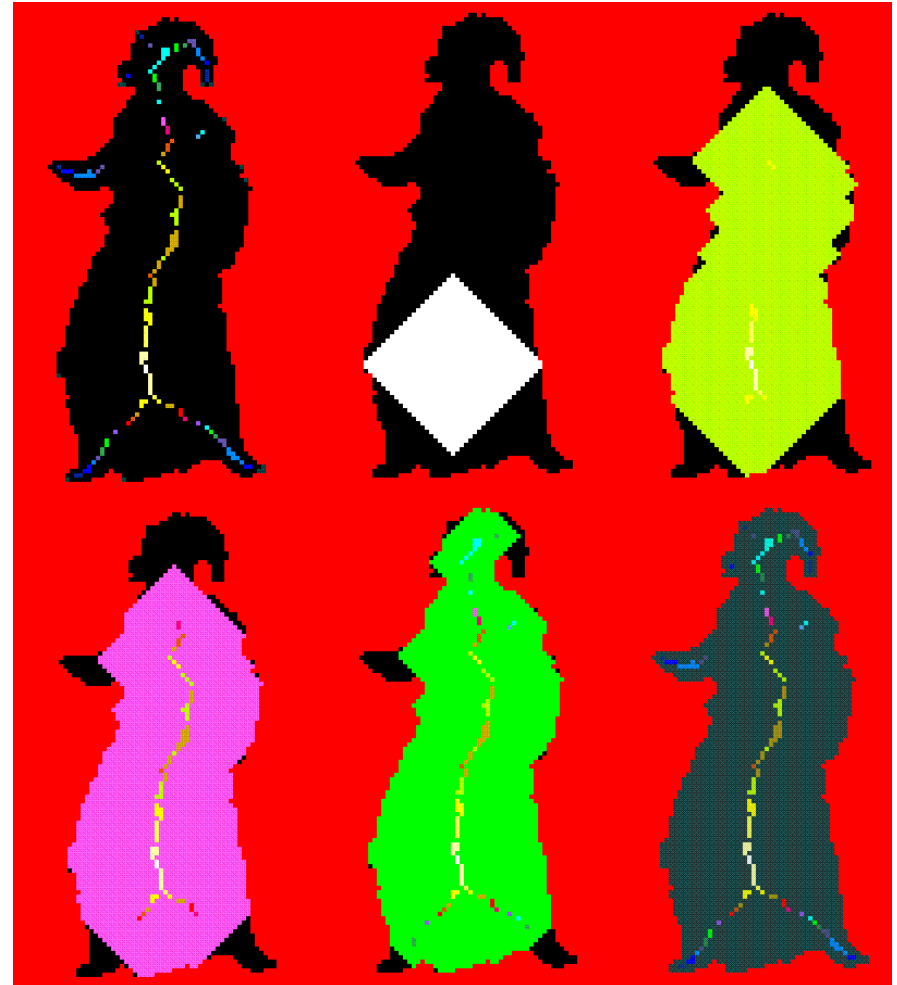


Distance transform and MAT



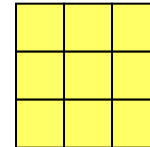
Reverting progressively MAT

- ✦ A procedure to derive the MAT from the DT is based on the comparison of neighboring labels to establish whether a local maximum exists
- ✦ This transform is complete in the sense that it is possible to revert it, so obtaining the original object back
- ✦ This recovery process can be implemented by expanding every pixel belonging to the MAT, using the corresponding maximal disc whose size is given by the pixel label. The logical union of all such discs reconstructs the original object
- ✦ This figure shows the progressive reconstruction, starting from the set of disks corresponding to the highest level (two white disks) until the sixth and last monk's profile, where discs, reduced to just one pixel, have been included
- ✦ This transform is compact since the full object may be described only by its labeled disk centers



Distance between two points

★ Distance between $X, Y \in Z$:



$A = \{ X \}; D = \{ Z \}$ $A =$ evolving binary image

while ($Z \neq A$) do $F =$ original image

$Z = A$

$A = (A \oplus K) \cap F$ $Z =$ connected component

$D = D + A$

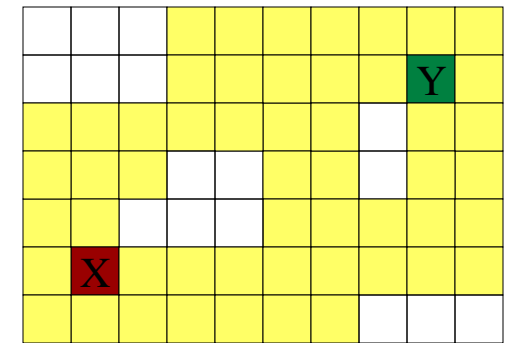
Done

★ If $A \equiv (A \oplus K) \cap F$ and Y has not been already reached: Z is not connected and Y is not reachable from X

★ Following a path of max gradient we can find one of the minimum paths between X, Y

K

F



D

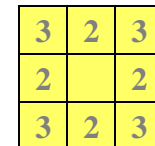
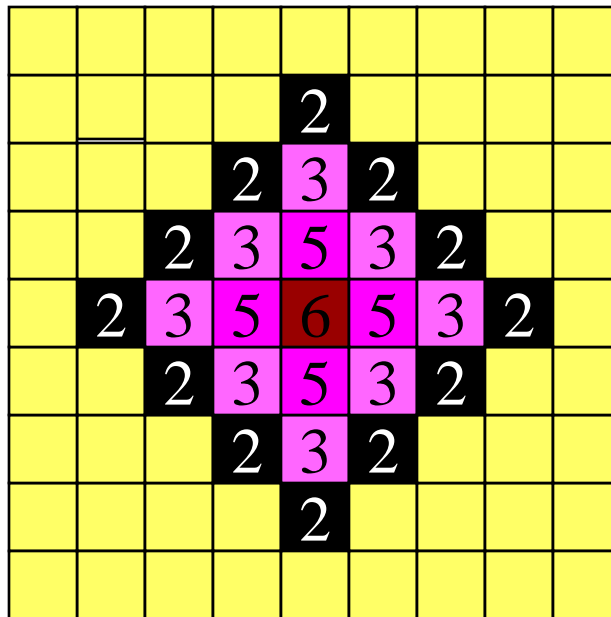
			4	4	4	3	2	1	
			5	5	4	3	2	1	
6	6	6	6	5	4	3		1	1
7	7	7			4	4		2	1
8	8				5	4	3	2	1
8	9	8	7	6	5	4	3	2	1
8	8	8	7	6	5	4			

Weighted DT

- ★ In this case all neighbors are not considered at the same distance (e.g. 8-connectivity)
- ★ Sequential algorithm, two scannings
 - ★ Direct scanning
 - ★ $\text{val} = \min_i \{p_i + w_i\}$ (i *previous neighbors*)
 - ★ Inverse scanning
 - ★ $\text{new-val} = \min_k \{p_k + w_k\}$ (k vicini *successivi*)
- ★ Example: a good approximation to the Euclidean distance (the result is about doubled) is given by:

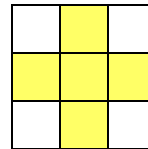
$$W = \begin{array}{|c|c|c|} \hline 3 & 2 & 3 \\ \hline 2 & & 2 \\ \hline 3 & 2 & 3 \\ \hline \end{array}$$

Weighted DT



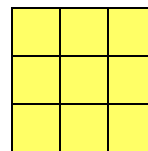
Topological paradoxes

- ★ The contour computed by the 4-connected disk is 8-connected



			1	1	1	1	1		
		1	2	2	2	2	2	1	
		1	2	3	3	3	2	1	
	1	2	3	4	4	4	3	2	1
1	2	2	3	3	3	3	2	1	
	1	1	2	2	2	2	2	1	
			1	1	1	1	1		

- ★ The contour computed by the 8-connected disk is 4-connected



			1	1	1	1	1		
		1	1	2	2	2	1	1	
		1	2	2	3	2	2	1	
	1	1	2	3	3	3	2	1	1
1	1	2	2	2	3	2	2	1	
	1	1	1	2	2	2	1	1	
			1	1	1	1	1		

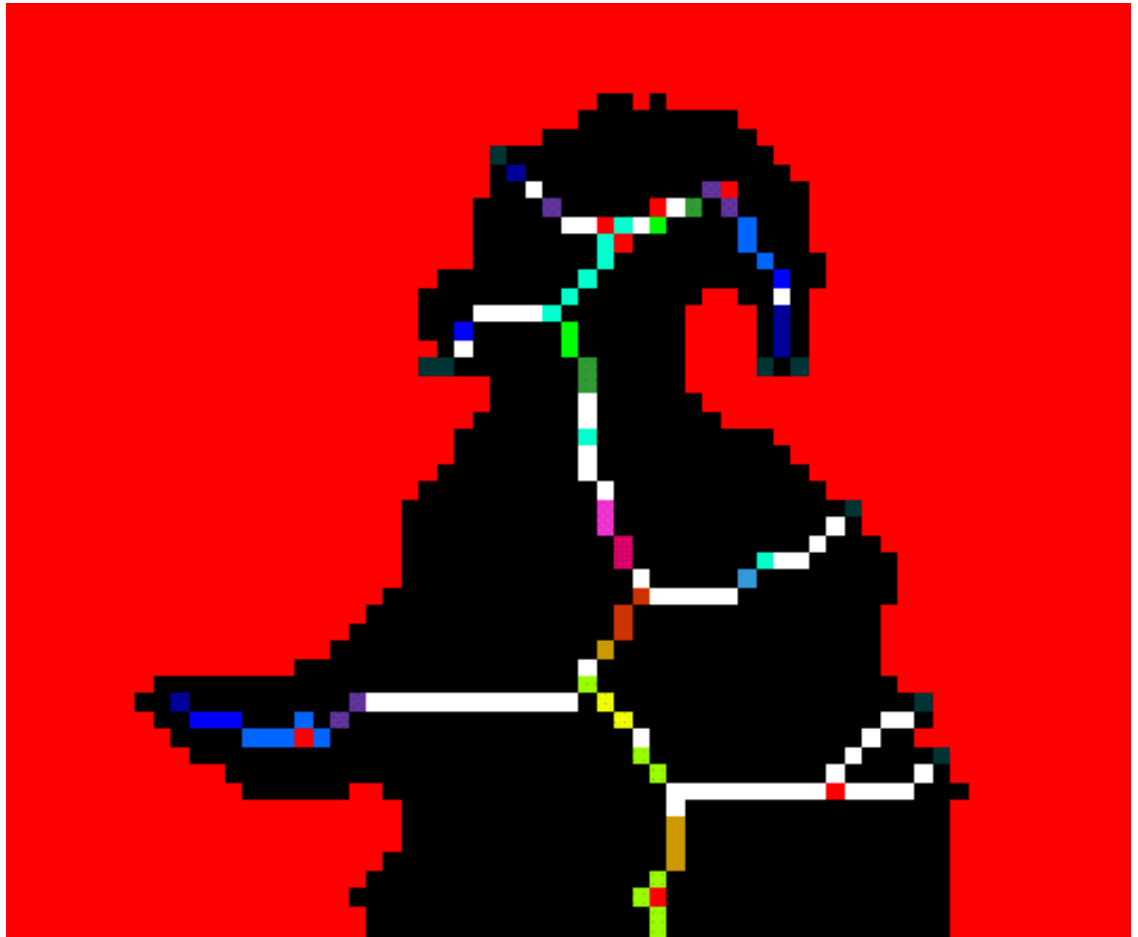
- ★ Hexagonal tessellation avoid paradoxes

Skeleton

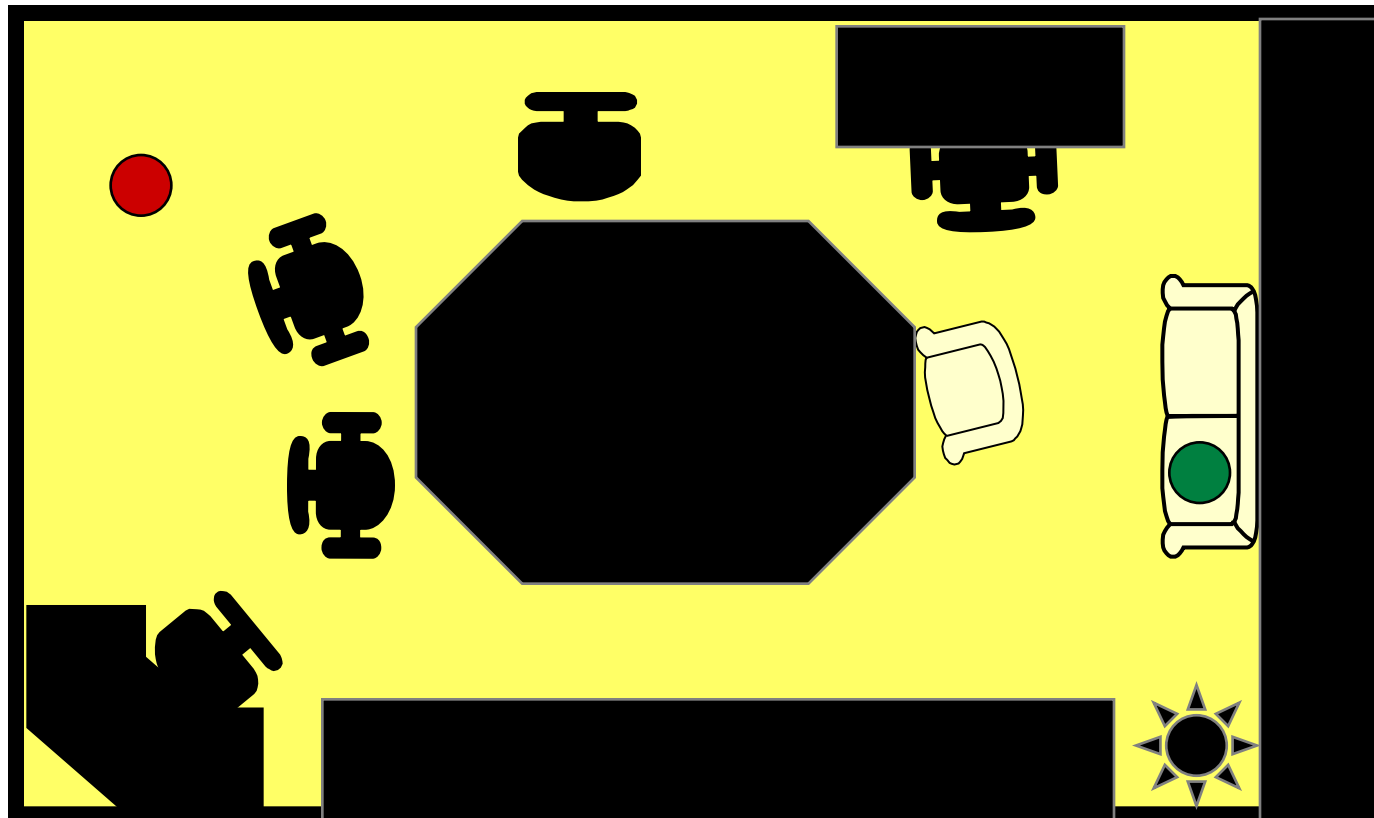
- ★ The skeleton F is a subset of F having the following property:
 - ★ It is connected if F is connected and maintains the same multiplicity order of F
 - ★ Its thickness is everywhere unitary
 - ★ It is centred in F
 - ★ All local maxima of F (MAT) belong to it

Skeleton from MAT

- ✦ MAT does not ensure connectivity for a connected object
- ✦ Many different algorithms were designed for generating skeletons. *A simple one proceeds in two steps:*
 - ✦ *find gaps between different branches and bridge them by joining the extremes along paths with directions dependent on context (white pixels);*
 - ✦ *thin the obtained branches so as to produce a one-pixel wide representation (red pixels)*
- ✦ One possible application of the skeleton is contour smoothing, where pruning branches with relatively short lengths and then inverting the skeleton transformation, a more regular contour is obtained
- ✦ The skeleton transformation is not reversible in general

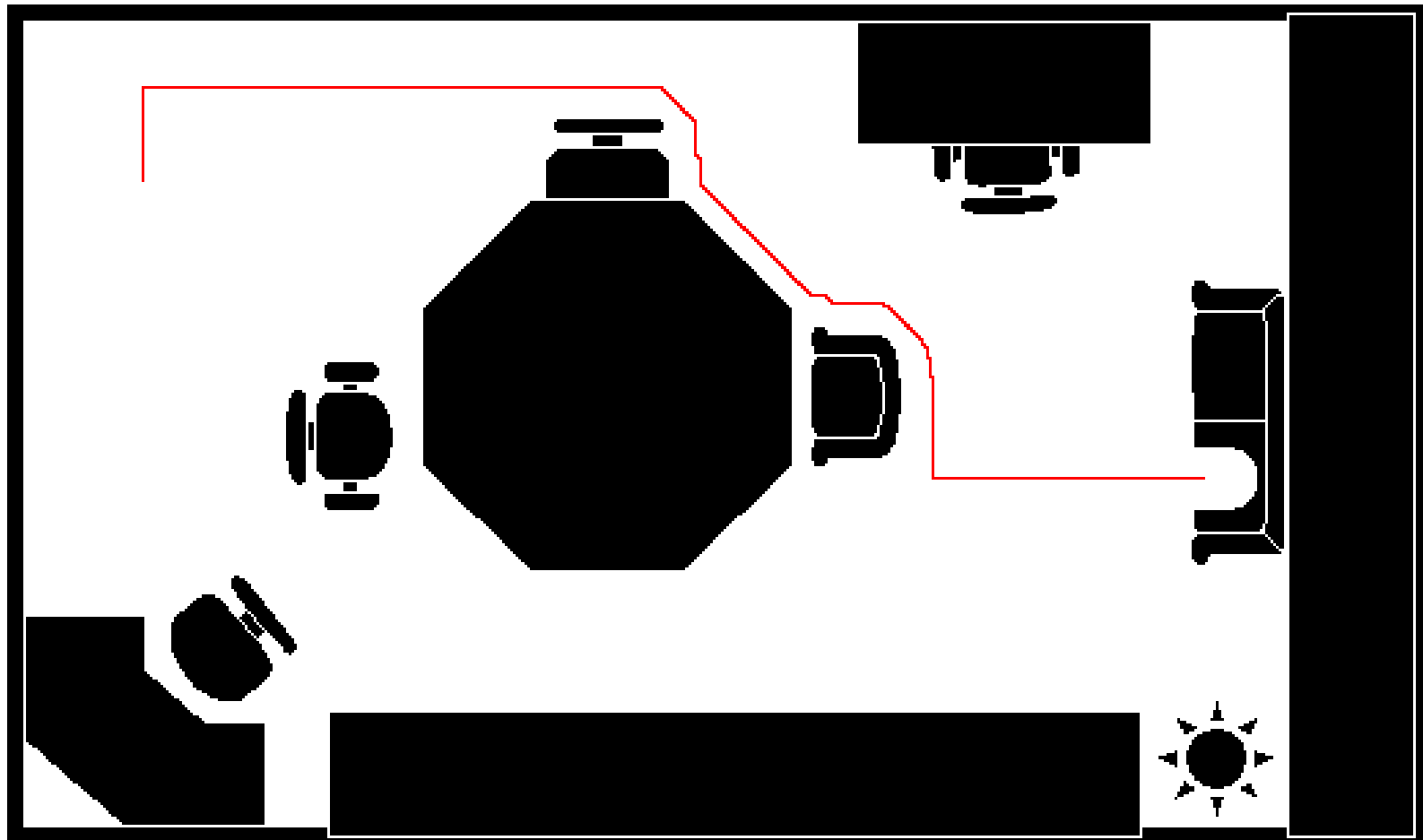


Example



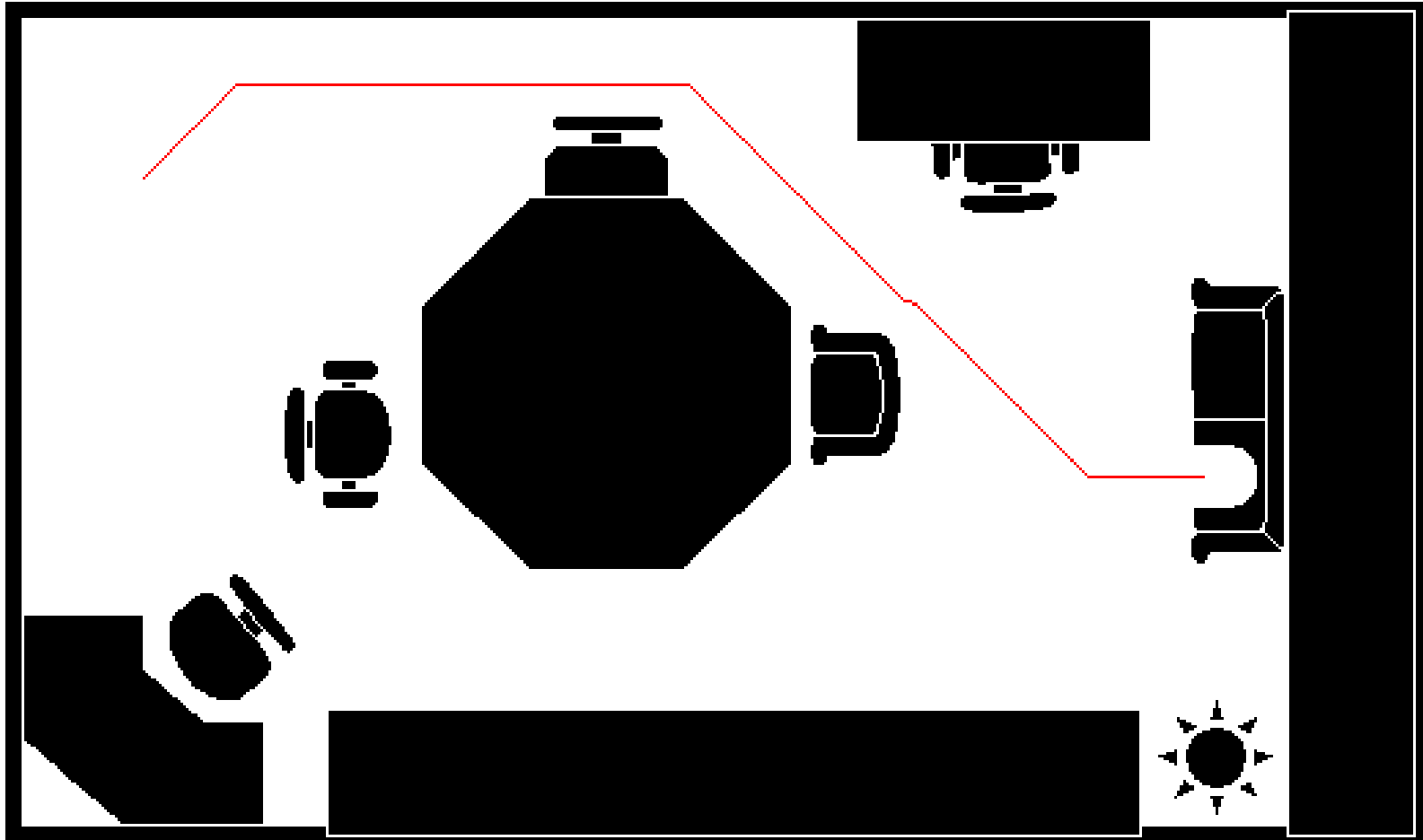


Minimum path
4-conn

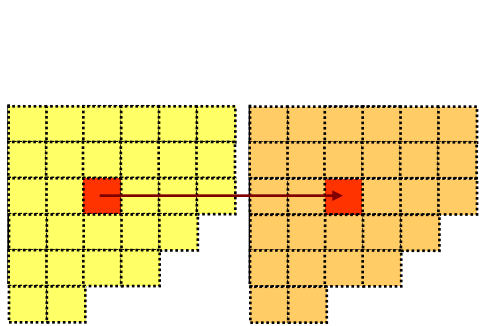




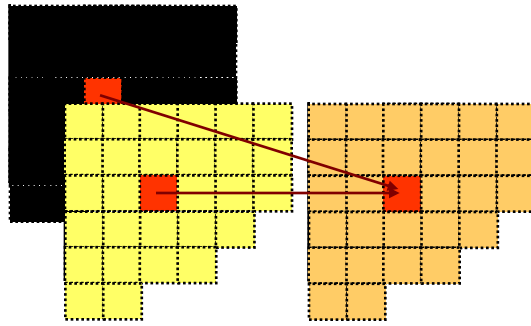
Minimum path
8-conn



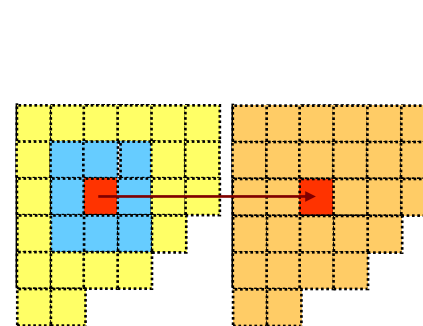
Low level IP operations



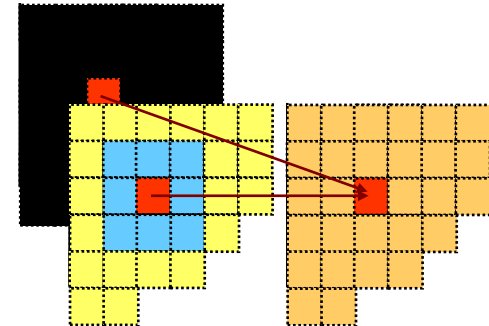
Monadic Point Operation



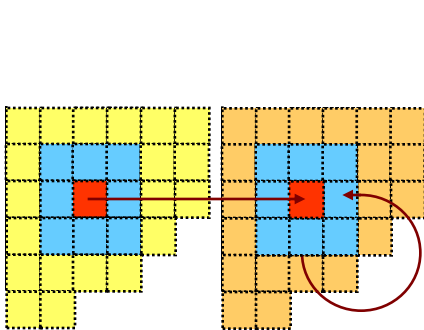
Dyadic Point Operation



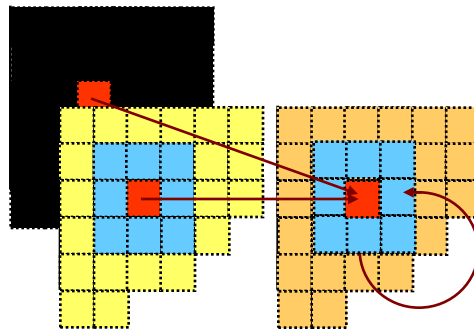
Monadic Local NN Operation



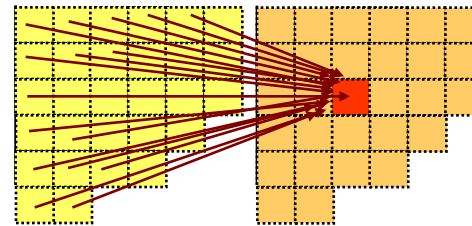
Dyadic Local NN Operation



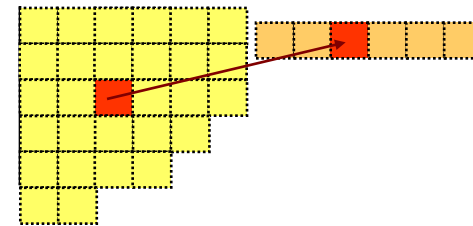
Monadic Recursive NN Operation



Dyadic Recursive NN Operation

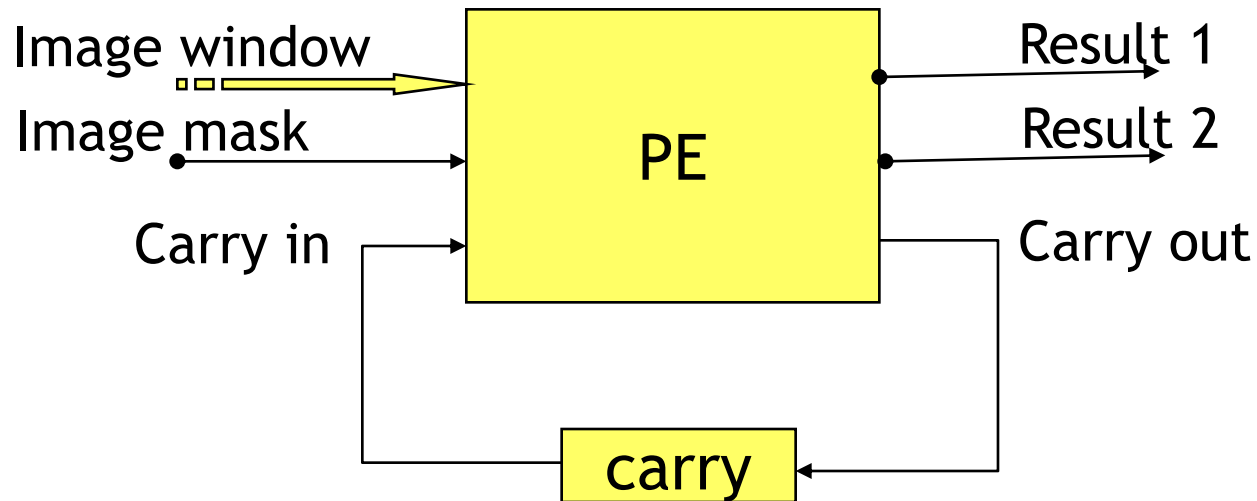


Global Operation



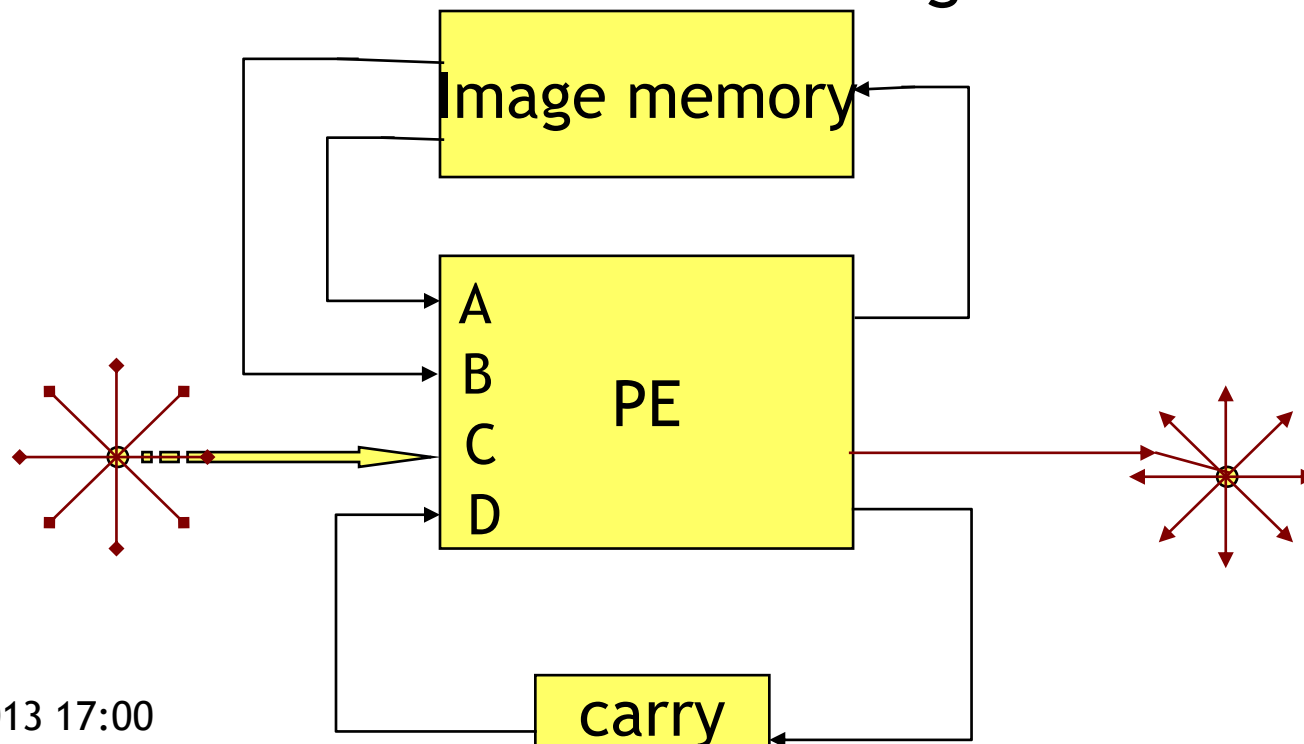
Statistical Operation

Neighbours parallelism cellular logic PE



SIMD Processing Element

- ◆ Bit parallel or bit serial neighbourhood connections
- ◆ Memory size: 32 - 64 Kbit
- ◆ Registers for accumulation of inputs/results
- ◆ Known pixel address in the image



Window mapping in PA

- ★ Fast processing of windows
- ★ Neighbourhood connectivity preserved

Scanning
↓

IMAGE

0	1	2	3	0	1	2	3
4	5	6	7	4	5	6	7
8	9	A	B	8	9	A	B
C	D	E	F	C	D	E	F
0	1	2	3	0	1	2	3
4	5	6	7	4	5	6	7
8	9	A	B	8	9	A	B
C	D	E	F	C	D	E	F

- ★ Processing speed reduced by:
 - ★ Overhead: (image size / array size)
- ★ Overhead in processing window edge pixels

Processor Array

0	1	2	3
4	5	6	7
8	9	A	B
C	D	E	F

Crinkle mapping in PA

- ★ Fast access of distant pixels
- ★ Negligible edge problem
- ★ No use of neighborhood connections

IMAGE

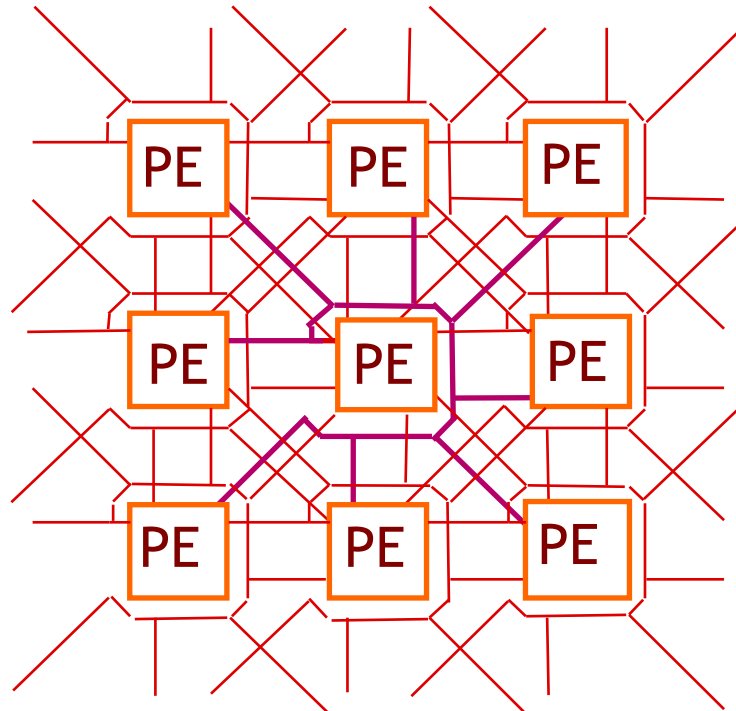
0	0	1	1	2	2	3	3
0	0	1	1	2	2	3	3
4	4	5	5	6	6	7	7
4	4	5	5	6	6	7	7
8	8	9	9	A	A	B	B
8	8	9	9	A	A	B	B
C	C	D	D	E	E	F	F
C	C	D	D	E	E	F	F

Processor Array

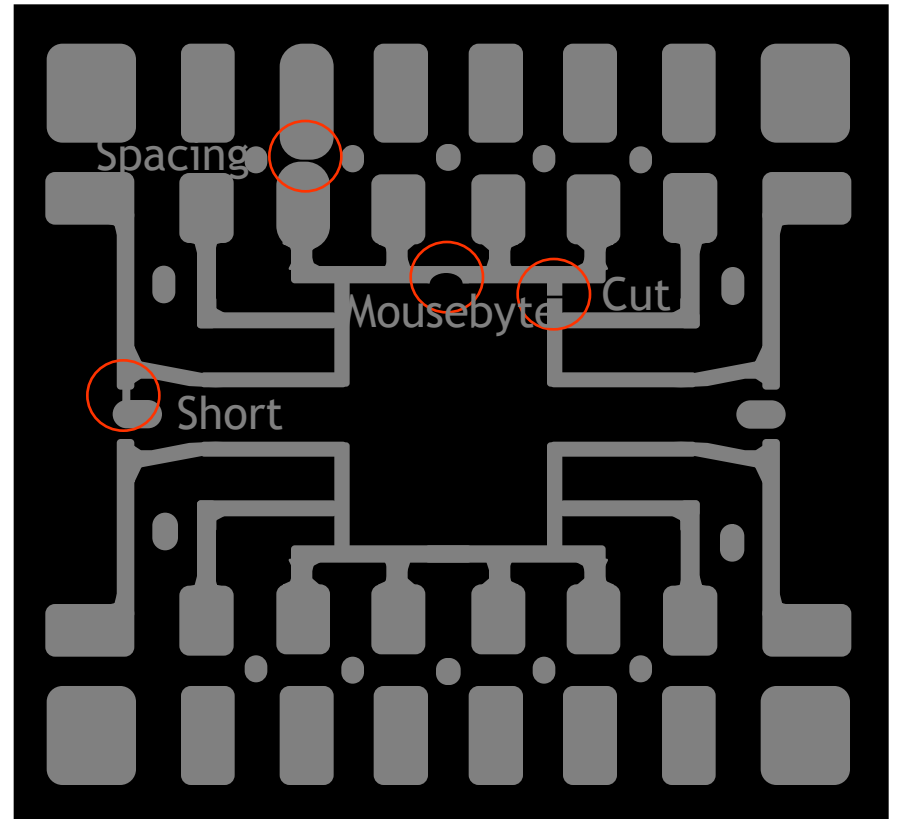
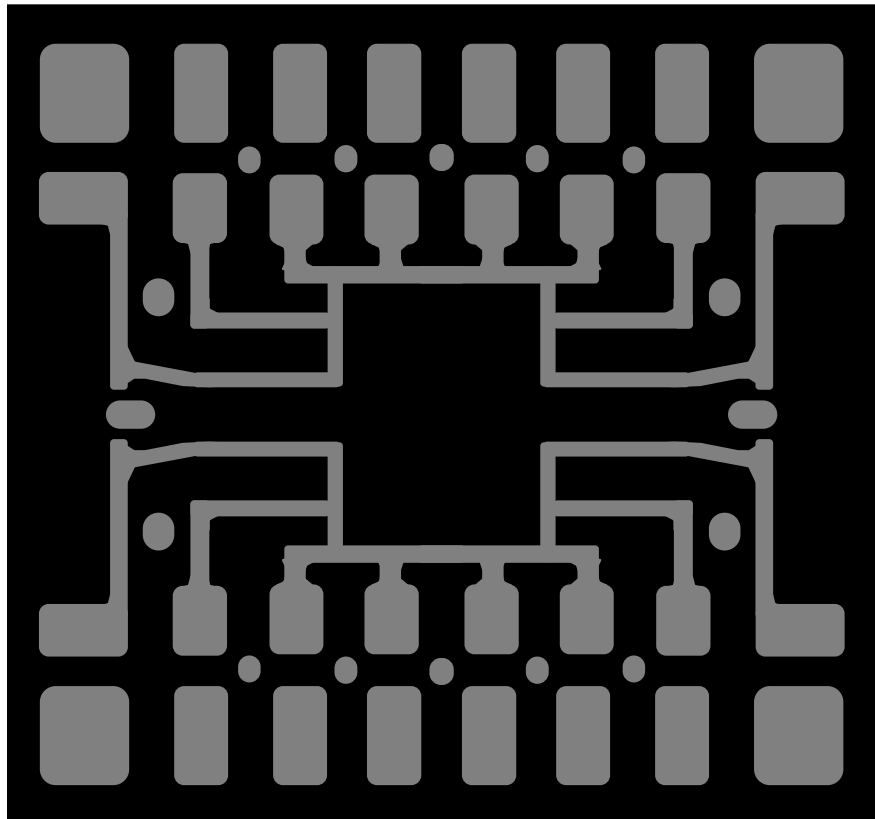
0	1	2	3
4	5	6	7
8	9	A	B
C	D	E	F

Pixel Parallelism: Processor arrays

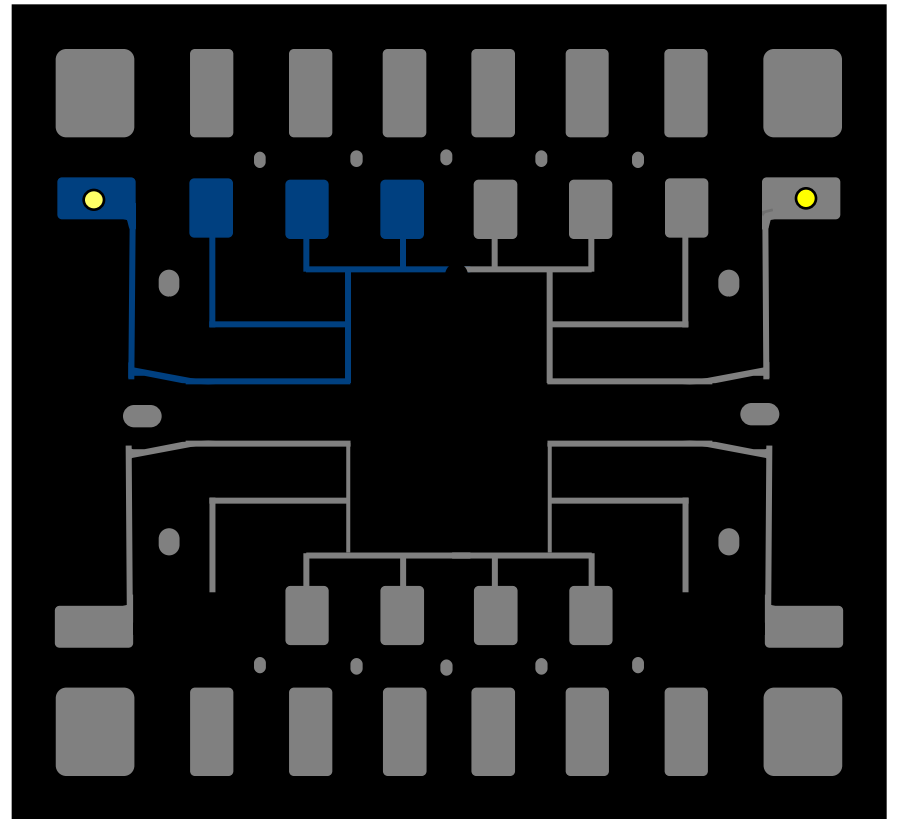
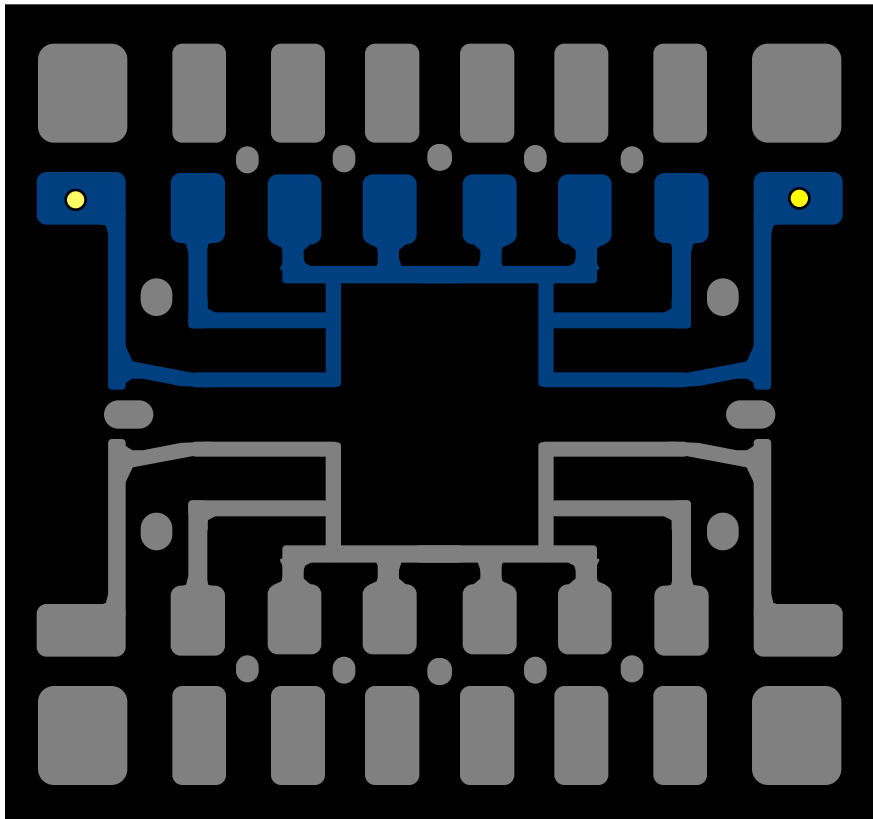
- ★ Processor Element (PE) includes local memory
- ★ Image distributed over all PE
- ★ All PE run the same program (SIMD)



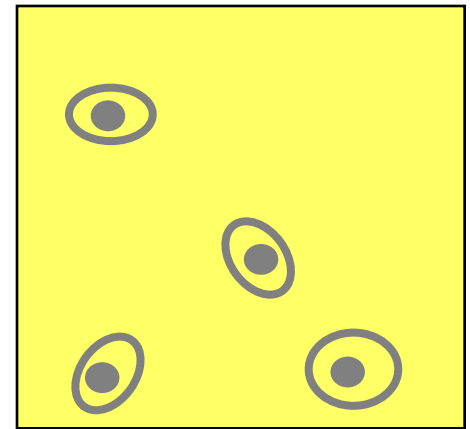
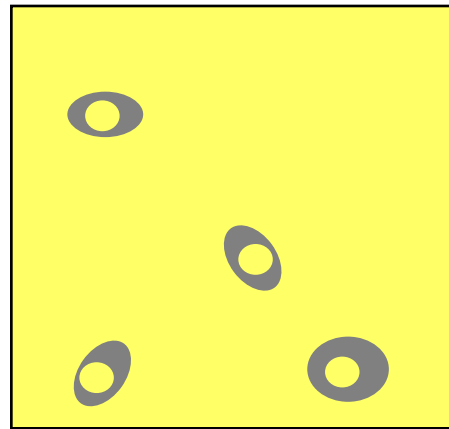
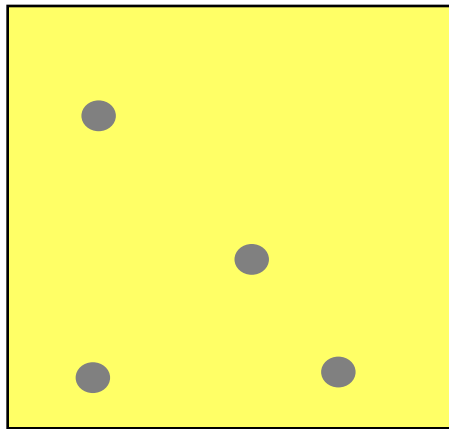
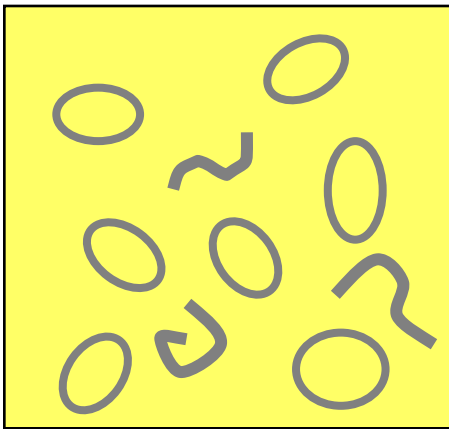
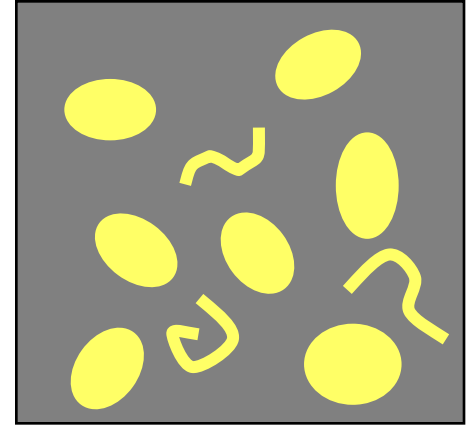
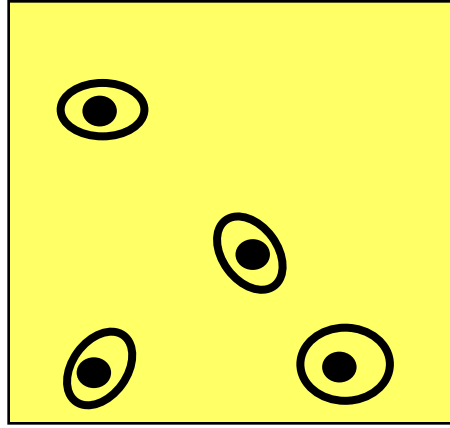
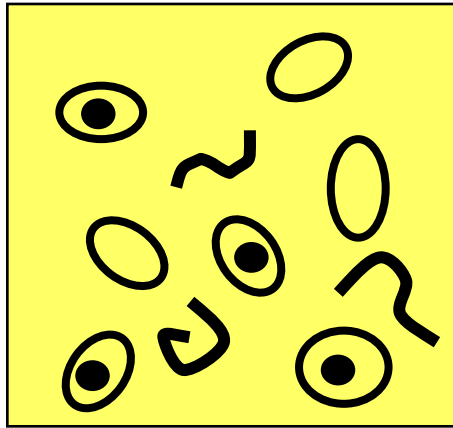
Propagation: examples



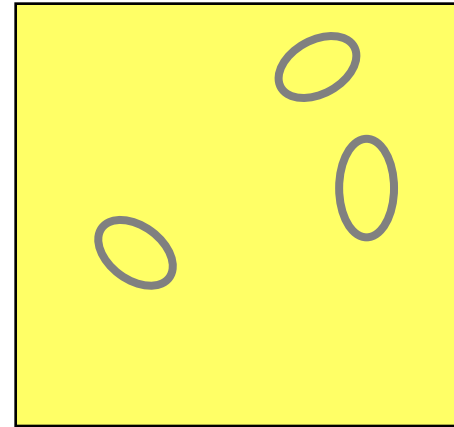
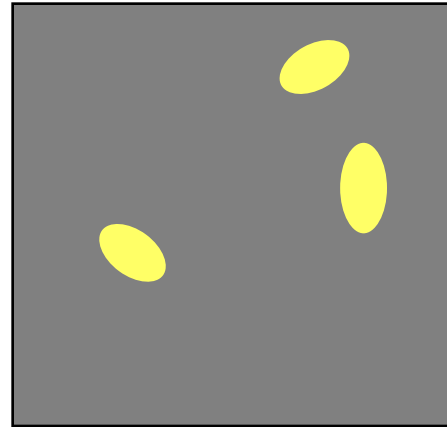
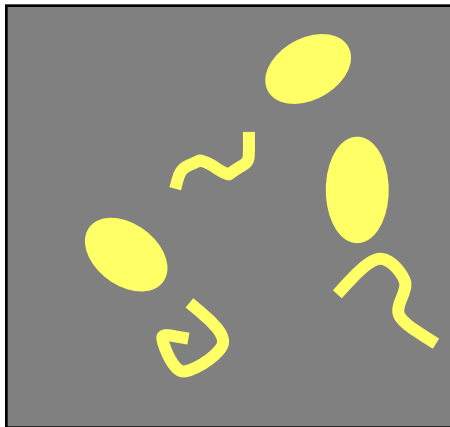
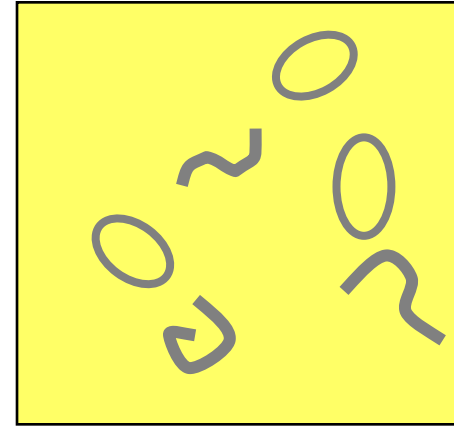
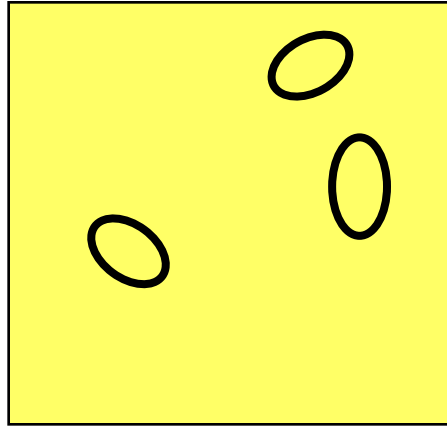
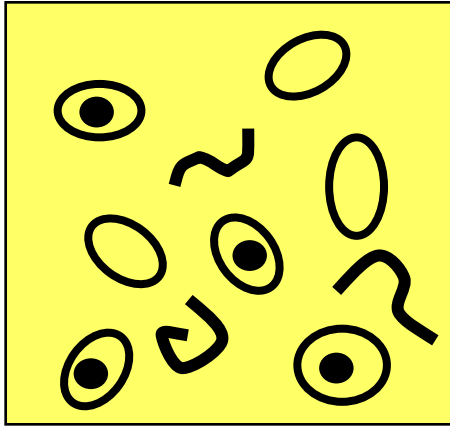
Mousebyte



Global OR (Or-sum-tree)

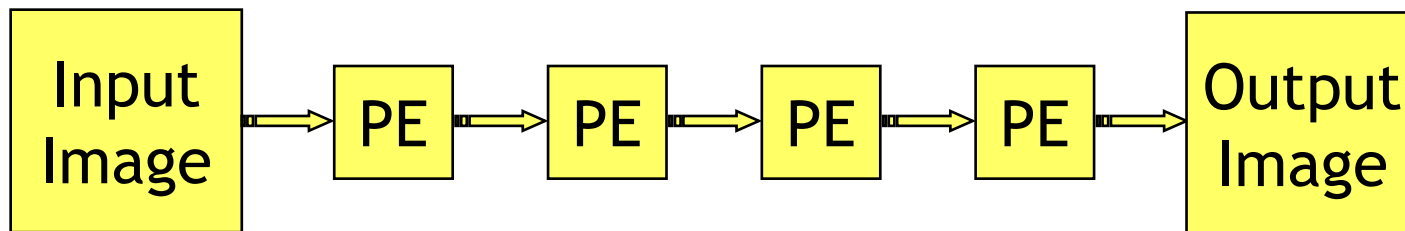


Global OR (Or-sum-tree)

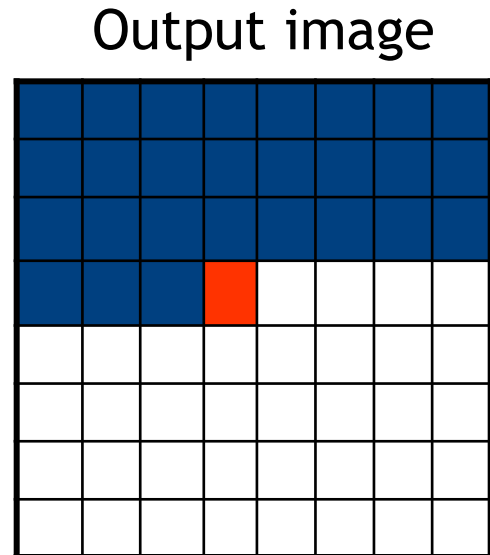
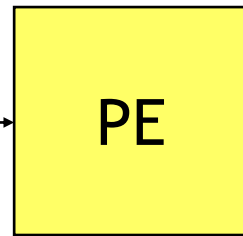
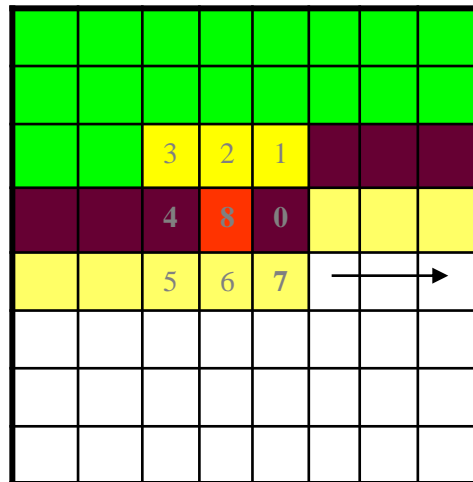
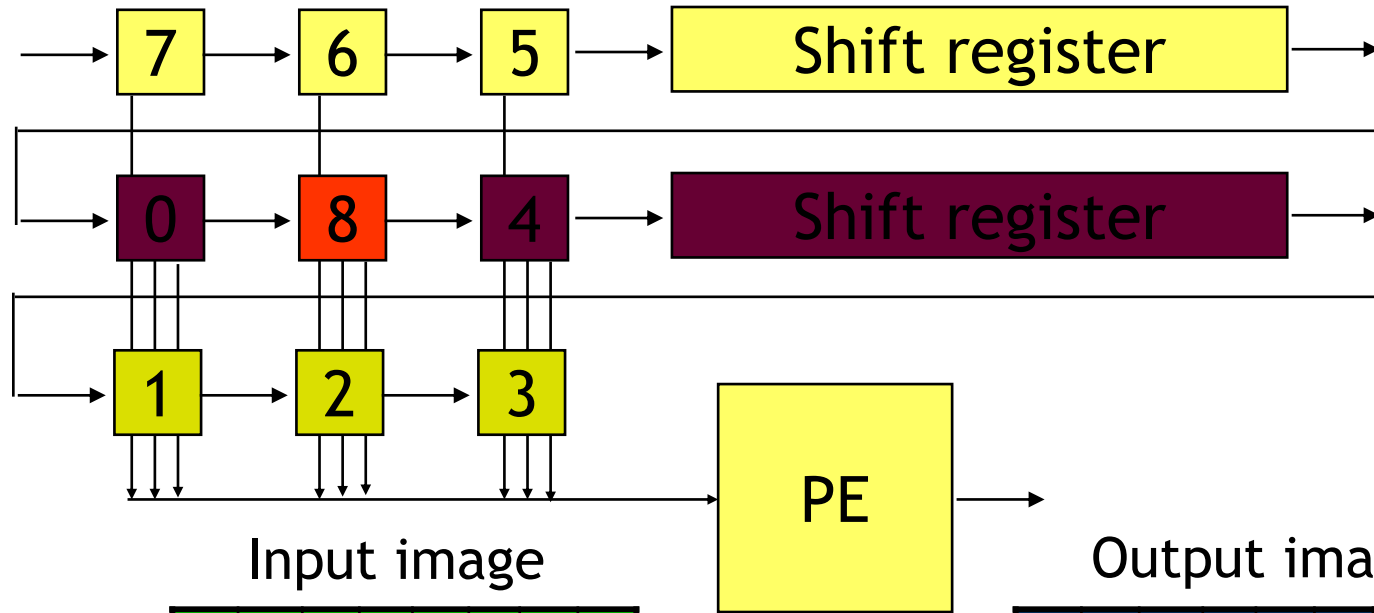


Operation Parallelism: Pipeline

- ★ Processing is done combined with transport
- ★ PE may be identical (Systolic array)
- ★ PE may be programmable (General purpose pipeline)
- ★ PE may be hard wired (Special purpose pipeline)



Cellular logic pipeline



Pixel 8 under processing

Processed pixels

Pipeline characteristics

- ★ Each PE processes another operation over all pixels
- ★ Output speed is as fast as processing a single instruction
- ★ Algorithm length is limited by the number of PEs. For larger algorithms recycling is necessary.
- ★ Complicated (e.g. grey value) operations can be realized by more complicated hardware
- ★ Pipelines are either inflexible or difficult to program

Comparison PA versus PL

✦ Processor arrays:

- ✦ Input/output problem
 - ✦ Lowest level: stream of instructions
 - ✦ Relatively easy to program
 - ✦ Higher degree of parallelism possible (images can be large)
 - ✦ Less powerful PE's
- ✦ 1. load image (part)
 - ✦ 2. load instruction
 - ✦ 3. process instruction
 - ✦ 4. repeat

✦ Pipelines:

- ✦ Instruction load problem
 - ✦ Lowest level: stream of pixels
 - ✦ Relatively difficult to program
 - ✦ Difficult to realize a very high degree of parallelism (algorithms are not long enough)
 - ✦ More powerful PEs
- ✦ 1. load instructions (part)
 - ✦ 2. process
 - ✦ 3. shift
 - ✦ 4. repeat