Path

A path from $p(x,y)$ to $q(s,t)$ is a sequence of distinct pixels.

$(x_0,y_0), (x_1,y_1)\ldots\ldots\ldots(x_n,y_n)$

Where

$(x_0,y_0) = (x,y)$, $(x_n,y_n) = (s,t)$
(x_i,y_i) is adjacent to (x_{i-1},y_{i-1})
for $1 \le i \le n$

$n \Rightarrow$ length of the path.

Connected component

Let

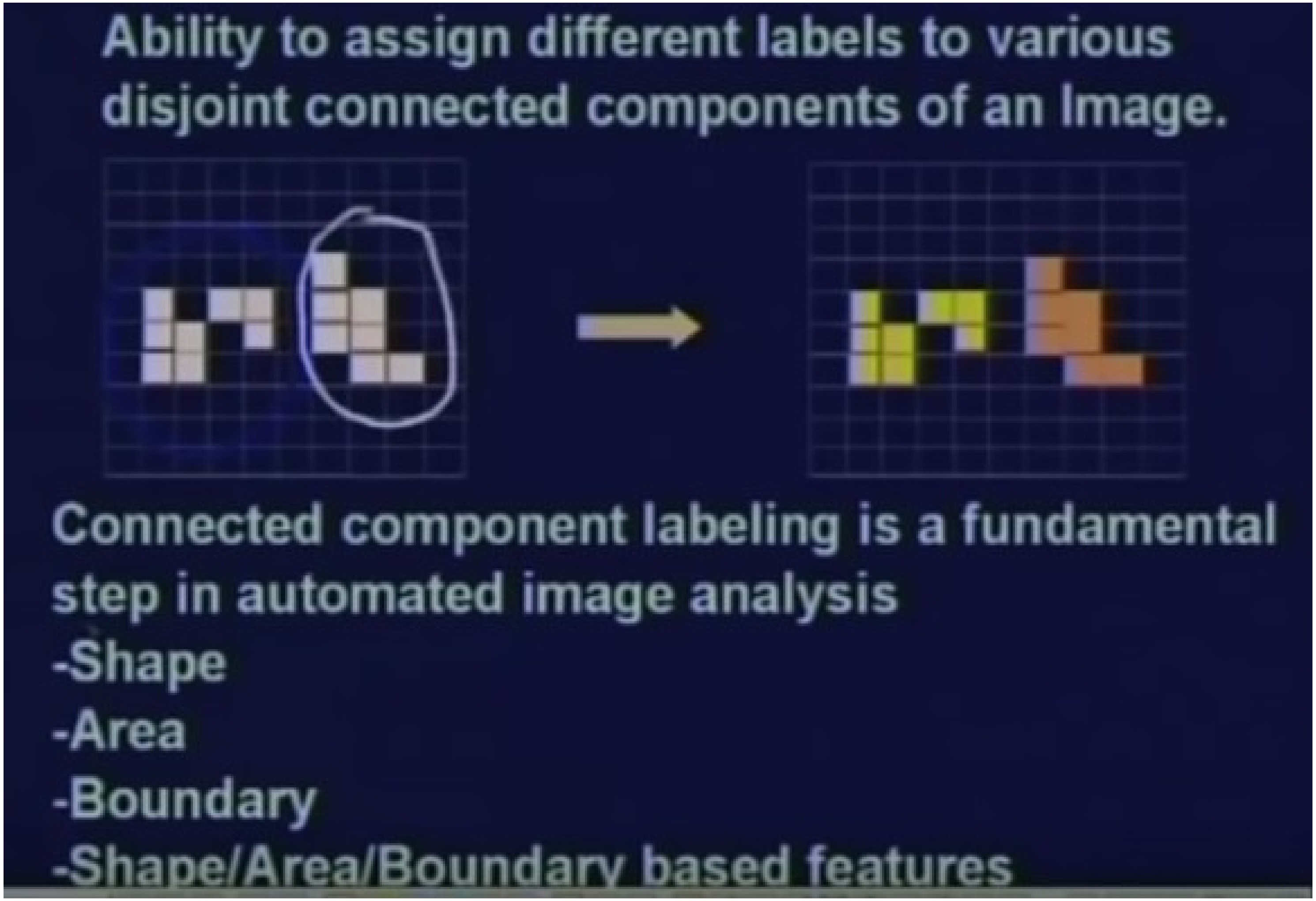
$S \subseteq I$ and $p, q \in S$

Then p is connected to q in S if there is a path
From p to q consisting entirely of pixels in S

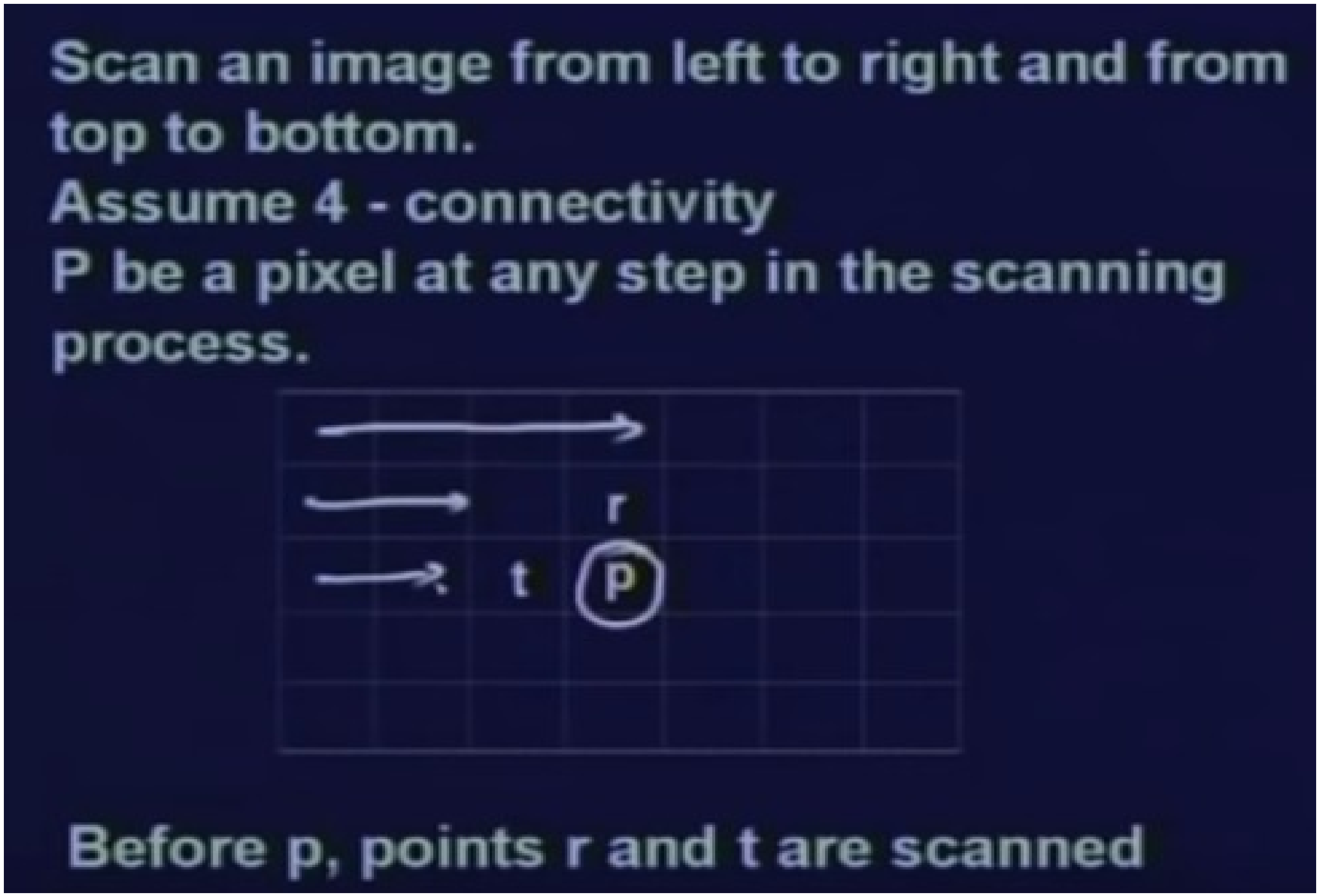
For any $p \in S$, the set of pixels in S that are
Connected to p is call a connected component
of S.

=> Any two pixels of a connected component
are connected to each other
Distinct connected components are disjoint

Connected component labelling



Algorithm


Scan an image from left to right and from top to bottom.
Assume 4 - connectivity
P be a pixel at any step in the scanning process.
r
t
p
Before p, points r and t are scanned

STEPS

I(p) => Pixel value at position p.

L(p) => Label assigned to pixel location p.

If I(p) = 0, move to next scanning position.

If I(p) = 1 and I(r) = I(t) = 0

Then assign a new label to position p

If I(p) = 1 and only one of the two neighbor is 1

Then assign its label to p.

If I(p) = 1 and both r and t are 1's, then

- If L(r) = L(t) than L(p)=L(r)
- If L(r) $\neq$ L(t) then assign on of the labels to p and make a note that the two labels are equivalent

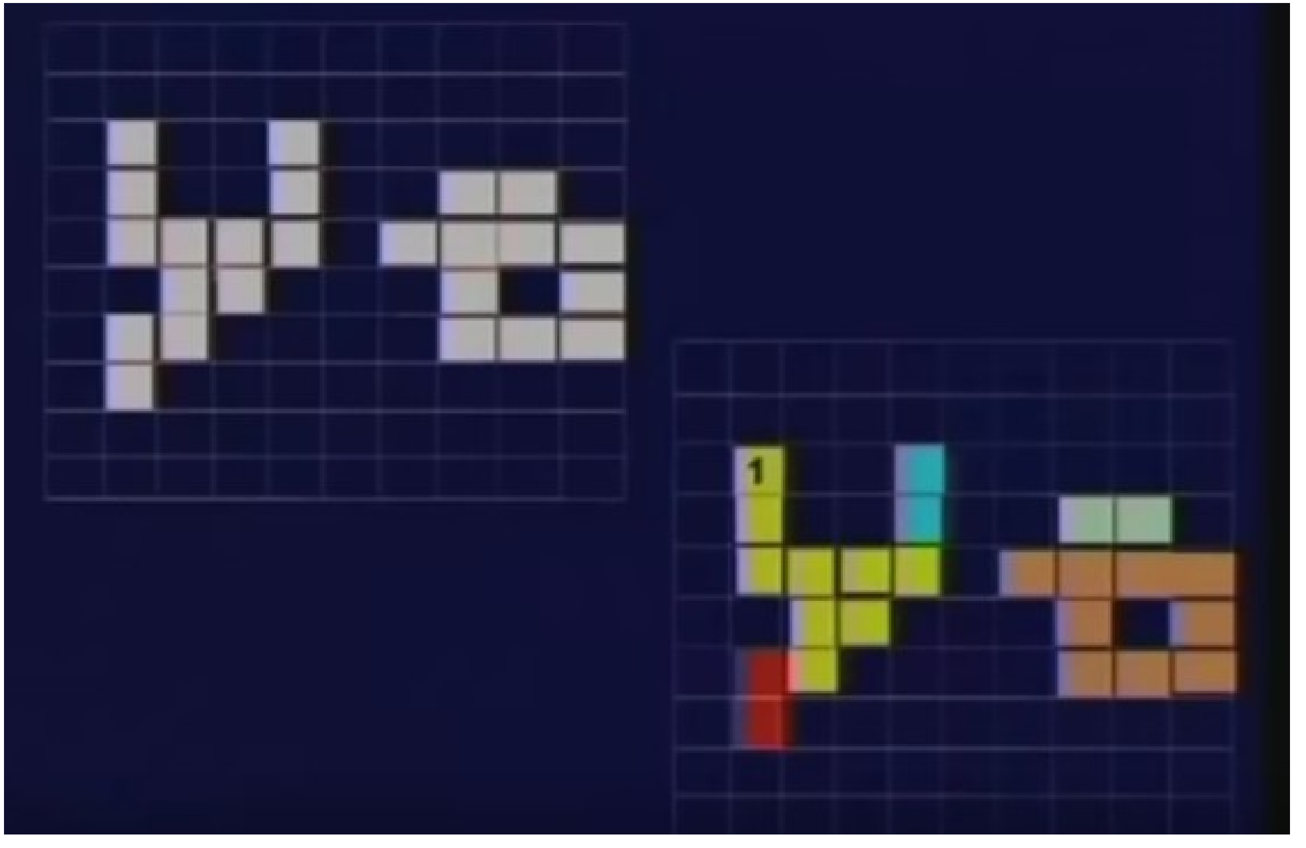
At end of the scan all pixels with value 1 are labeled.

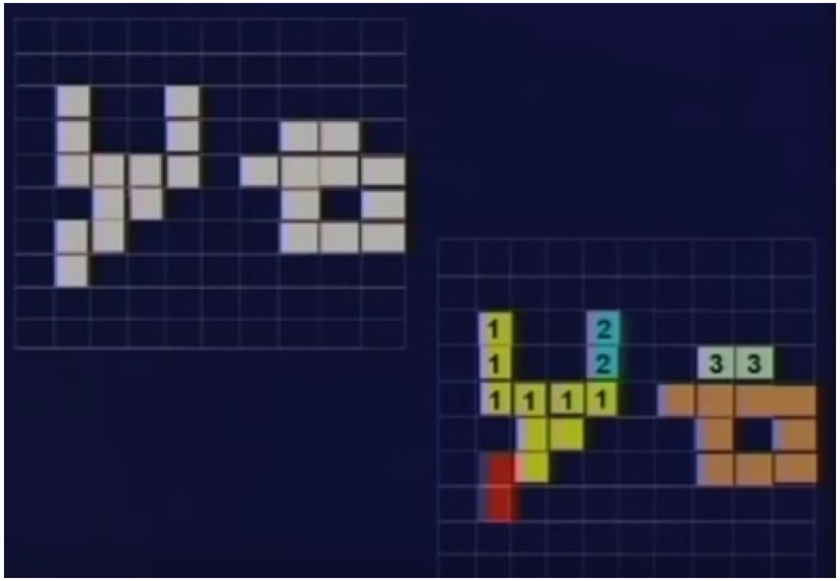
Some labels are equivalent.

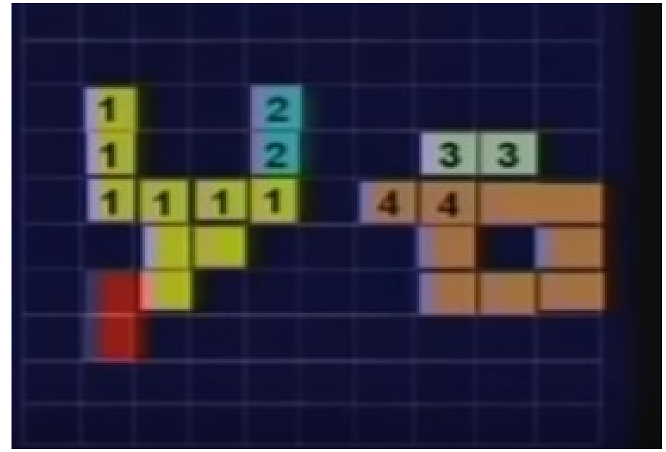
During second pass process equivalent pairs to from equivalence classes.

Assign a different label to each class.
In the second pass through the image replace each label by the label assigned to its equivalence class.

Algorithm demonstration

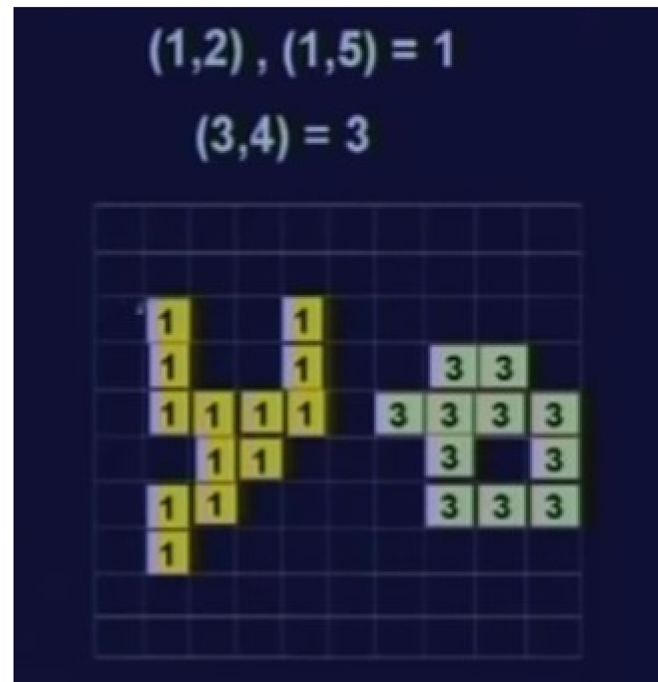







(1,2) , (3,4) , (1,5)


Equivalent pairs:
(1,2) , (3,4) , (1,5)


(1,2) , (1,5) = 1
(3,4) = 3

Paths & Path lengths

- A *path* from pixel p with coordinates (x, y) to pixel q with coordinates (s, t) is a sequence of distinct pixels with coordinates:
  $(x_0, y_0), (x_1, y_1), (x_2, y_2) \ldots (x_n, y_n)$,
  where $(x_0, y_0)=(x, y)$ and $(x_n, y_n)=(s, t)$;
  (x_i, y_i) is adjacent to (x_{i-1}, y_{i-1}) $1 \leq i \leq n$
- Here n is the *length* of the path.
- We can define 4-, 8-, and m-paths based on type of adjacency used.

Paths

<u>Example # 1</u>: Consider the image segment shown in figure. Compute length of the ***shortest-4, shortest-8 & shortest-m paths*** between pixels p & q where,
V = {1, 2}.

	4	2	3	2	q
	3	3	1	3	
	2	3	2	2	
p	2	1	2	3	

Paths

Example # 1:

Shortest-4 path:

V = {1, 2}.

	4	2	3	2	q
	3	3	1	3	
	2	3	2	2	
p	2 →	1	2	3	

Paths

Example # 1:

Shortest-4 path:

V = {1, 2}.

	4		2		3	2	q
	3		3		1	3	
	2		3		2	2	
p	2	→	1	→	2	3	

Paths

<u>Example # 1</u>:

<u>Shortest-4 path:</u>

V = {1, 2}.

	4	2	3	2	q
	3	3	1	3	
	2	3	2	2	
p	2 →	1 →	2	3	

Paths

Example # 1:

Shortest-4 path:

V = {1, 2}.

	4	2	3	2	q
	3	3	1	3	
	2	3	2	2	
p	2	1	2	3	

Paths

<u>Example # 1</u>:

<u>Shortest-4 path:</u>

V = {1, 2}.

	4	2	3	2	q
	3	3	1	3	
	2	3	2	2	
p	2	1	2	3	

So, Path does not exist.

Paths

Example # 1:

Shortest-8 path:

$V = \{1, 2\}$.

4	2	3	2	q
3	3	1	3	
2	3	2	2	
p 2	1	2	3	

Paths

Example # 1:

Shortest-8 path:

V = {1, 2}.

	4	2	3	2	q
	3	3	1	3	
	2	3	2	2	
p	2 →	1	2	3	

Paths

Example # 1:

Shortest-8 path:

V = {1, 2}.

	4	2	3	2	q
	3	3	1	3	
	2	3	2	2	
p	2	1	2	3	

Paths

Example # 1:

Shortest-8 path:

V = {1, 2}.

	4	2	3	2	q
	3	3	1	3	
	2	3	2	2	
p	2	1	2	3	

Paths

Example # 1:

Shortest-8 path:

V = {1, 2}.

	4	2	3	2	q
	3	3	1	3	
	2	3	2	2	
p	2	1	2	3	

Paths

Example # 1:

Shortest-8 path:

V = {1, 2}.

	4	2	3	2	q
	3	3	1	3	
	2	3	2	2	
p	2	1	2	3	

So, shortest-8 path = 4

Paths

Example # 1:

Shortest-m path:

V = {1, 2}.

	4	2	3	2	q
	3	3	1	3	
	2	3	2	2	
p	2	1	2	3	

Paths

<u>Example # 1</u>:

<u>Shortest-m path:</u>

V = {1, 2}.

	4	2	3	2	q
	3	3	1	3	
	2	3	2	2	
p	2 →	1	2	3	

Paths

<u>Example # 1</u>:

<u>Shortest-m path:</u>

V = {1, 2}.

	4	2	3	2	q
	3	3	1	3	
	2	3	2	2	
p	2 →	1 →	2	3	

Paths

Example # 1:

Shortest-m path:

V = {1, 2}.

	4	2	3	2	q
	3	3	1	3	
	2	3	2	2	
p	2 →	1 →	2	3	

Paths

<u>Example # 1</u>:

<u>Shortest-m path:</u>

V = {1, 2}.

	4	2	3	2	q
	3	3	1	3	
	2	3	2	2	
p	2 →	1 →	2	3	

Paths

Example # 1:

Shortest-m path:

V = {1, 2}.

	4	2	3	2	q
	3	3	1	3	
	2	3	2	2	
p	2	1	2	3	

Paths

Example # 1:

Shortest-m path:

V = {1, 2}.

	4	2	3	2	q
	3	3	1	3	
	2	3	2	2	
p	2	1	2	3	

So, shortest-m path = 5

Paths

Example # 1:

Shortest-m path:

V = {1, 2}.

	4	2	3	2	q
	3	3	1	3	
	2	3	2	2	
p	2	1	2	3	

So, shortest-m path = 5

Relationships between Pixels

- On completion the students will be able to

1. Learn different distance measures
2. Application of Distance measure
3. Arithmetic/ Logical operations on images
4. Neighborhood operations on images

Take three pixels

$P \approx (x,y)$ $q \approx (s,t)$ $z \approx (u,v)$

D is a distance function or metric if

$D(p,q) \geq 0$; $D(p,q) = 0$ iff $p = q$

$D(p,q) = D(q,p)$

$D(p,z) \leq D(p,q) + D(q,z)$

Euclidian Distance

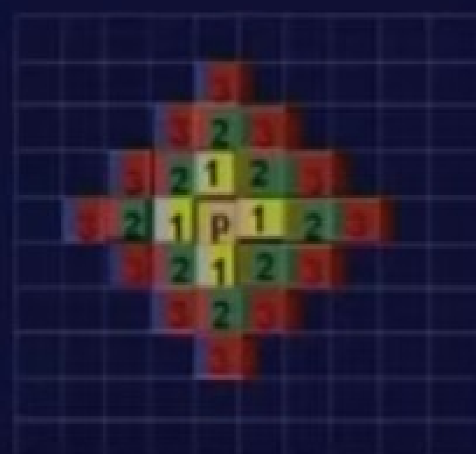
$$D_e(p,q) = [\,(x-s)^2 + (y-t)^2\,]^{1/2}$$

Set of points $S = \{\, q \mid D(p,q) \le r \,\}$ are the points contained in a disk of radius r centered at p.

D_4 distance or City-Block (Manhattan) Distance.

$$D_4(p,q) = | x-s | + | y-t |$$

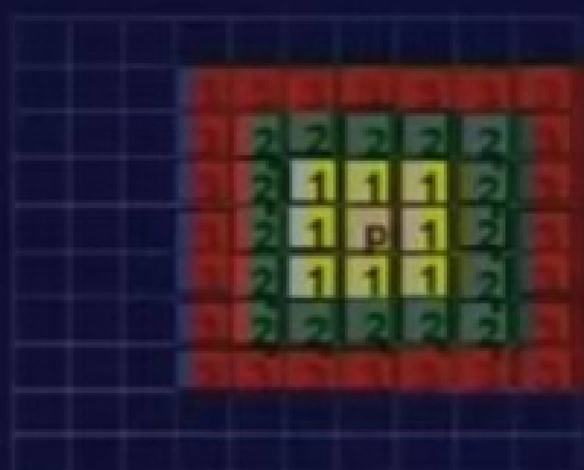
Points having city block distance from p less than or equal to r from diamond centered at p.



D_8 distance or chess board distance is defined as

$$D_8(p,q) = \max(|x-s|, |y-t|)$$

$S = \{ q \mid D_8(p,q) \leq r \}$ forms a square centered at p.



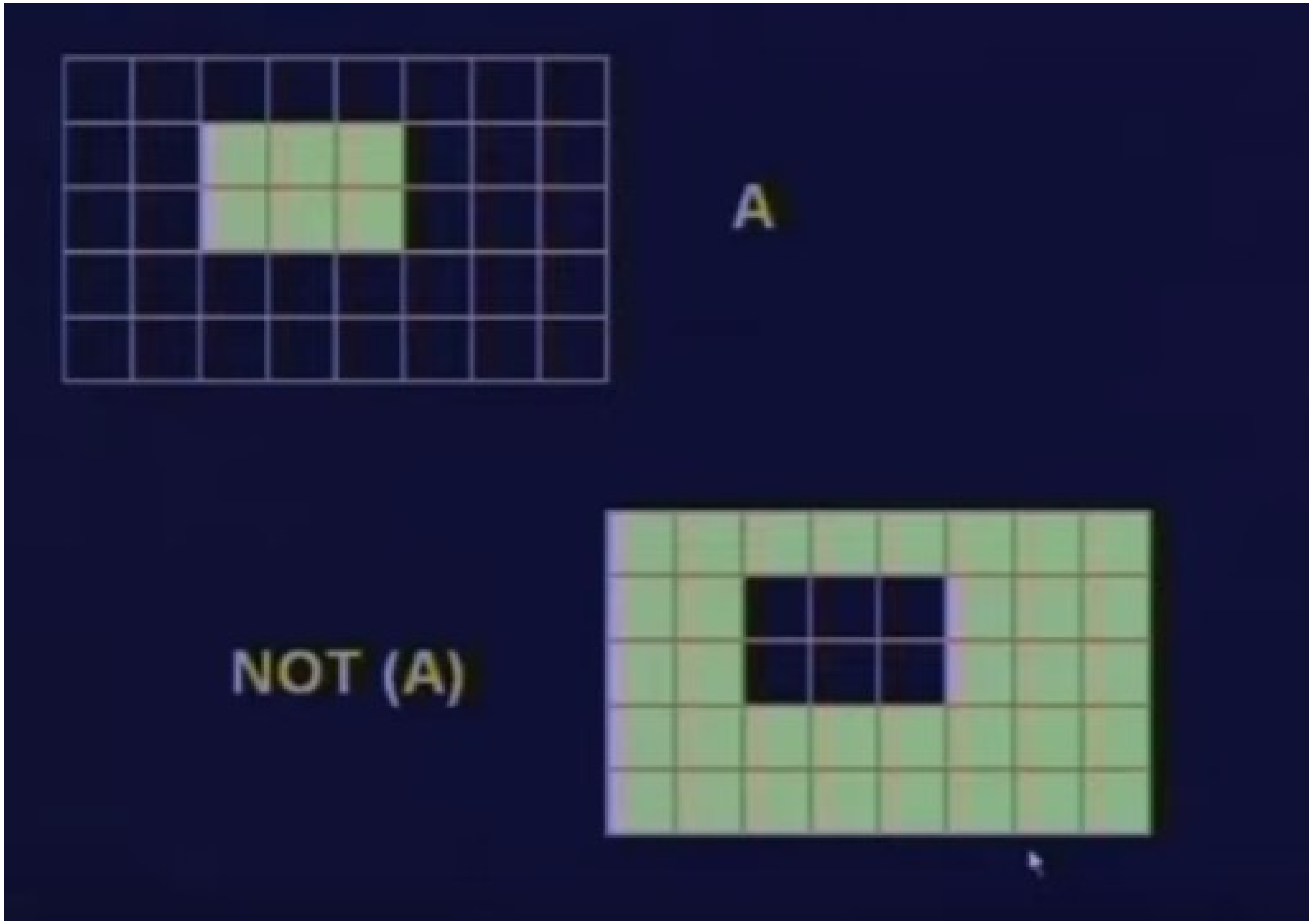
Points with $D_8 = 1$ are 8 neighbors of p

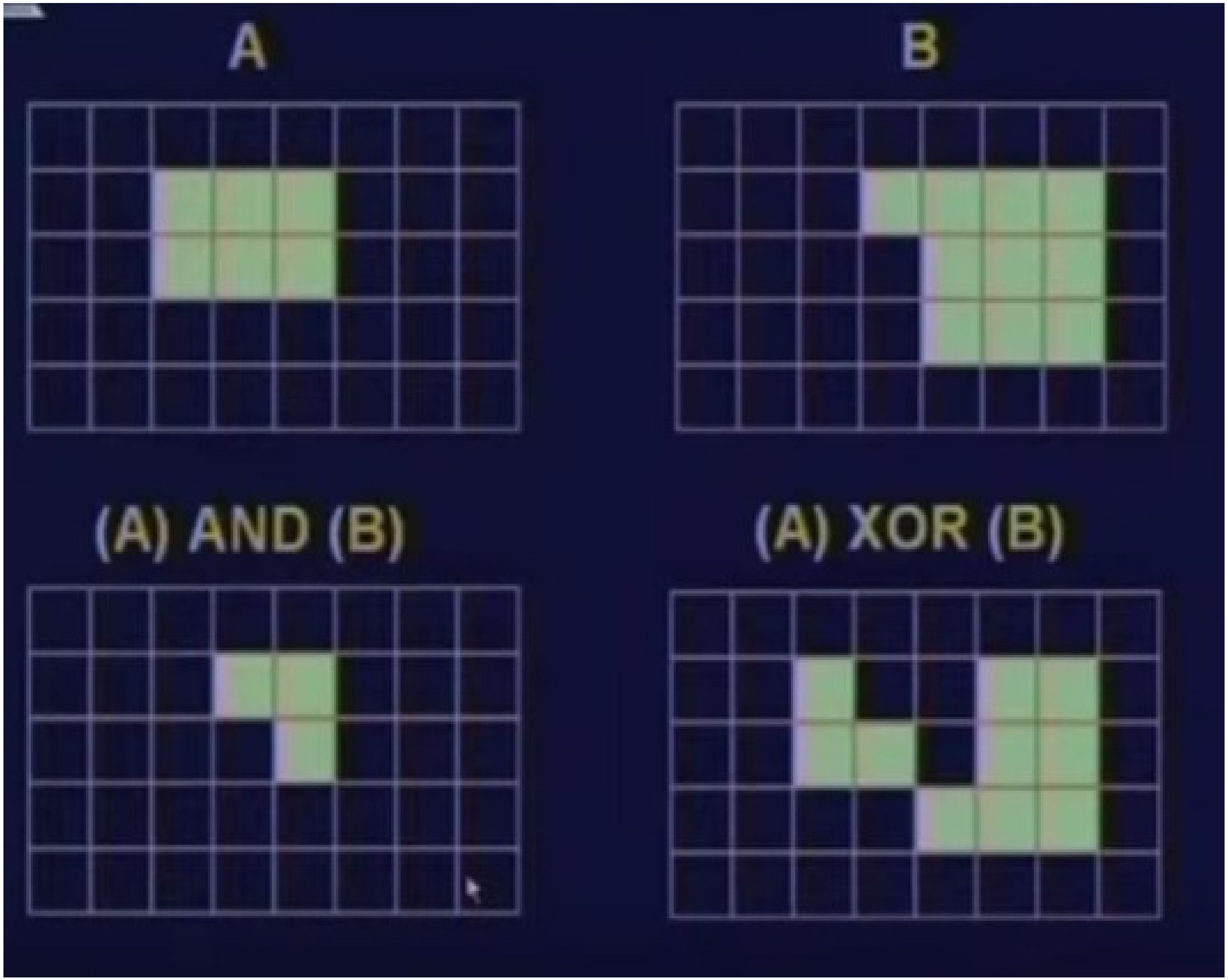
Arithmetic / Logical Operation

Following Arithmetic/Logical operations between two pixels p and q are used extensively

Arithmetic	Logical
p+q	p.q
p-q	p+q
p*q	p'
p%q	

Logical operations apply to binary images Only => Usually pixel by


A
NOT (A)


A
B
(A) AND (B)
(A) XOR (B)

Neighborhood Operations

The value assigned to a pixel is a function of its gray label and the gray labels of its neighbors.

Z_1	Z_2	Z_3
Z_4	Z_5	Z_6
Z_7	Z_8	Z_9

$$Z = 1/9\,(Z_1 + Z_2 + Z_3 + \ldots\ldots + Z_9) = \text{Average}$$

Template

More general form

Z_1	Z_2	Z_3
Z_4	Z_5	Z_6
Z_7	Z_8	Z_9

W_1	W_2	W_3
W_4	W_5	W_6
W_7	W_8	W_9

$$Z = W_1Z_1 + W_2Z_2 + \dots\dots\dots + W_9Z_9$$

$$= \sum_{i=1}^{9} W_iZ_i$$

Same as averaging if $W_i = 1/9$

Neighborhood Operations

Various important operations can be Implemented by proper selection of Coefficients W_i

----- Noise filtering

----- Thinning

----- Edge detection

etc…