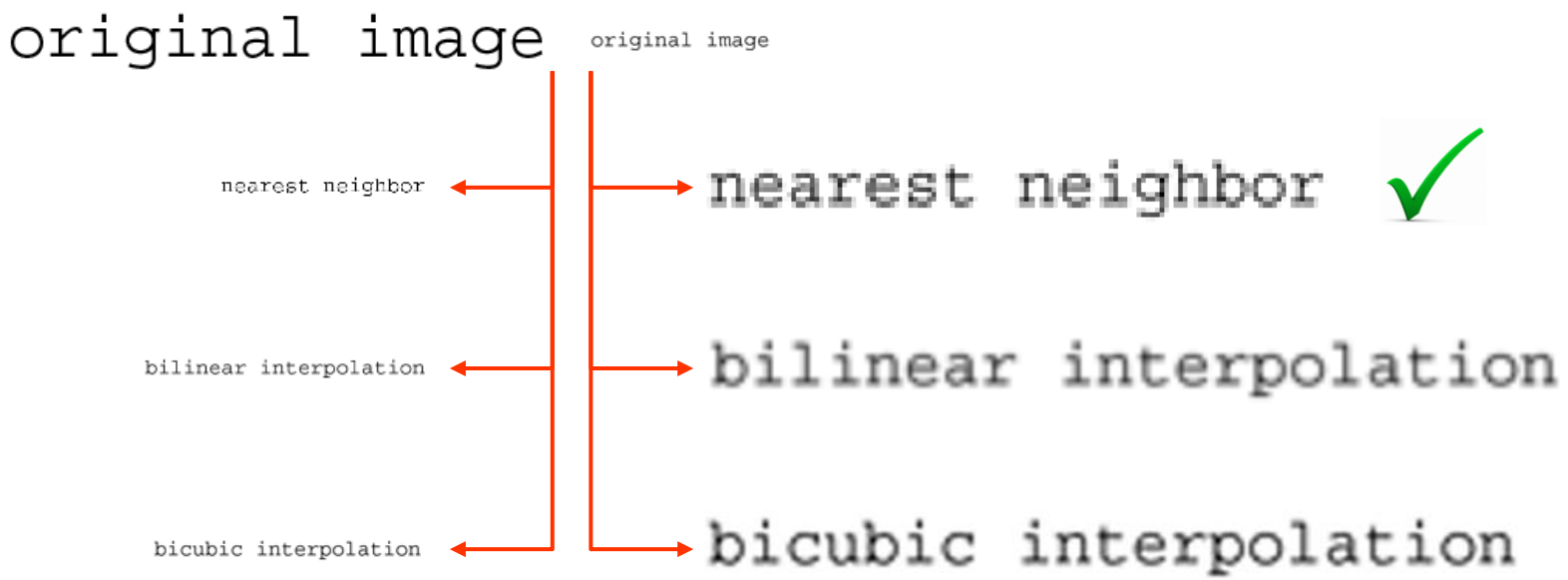
# Digital Image Processing

**Lecture # 2**
**Fundamentals & Transformations**

# Image Interpolation

- Image resizing
- Three methods



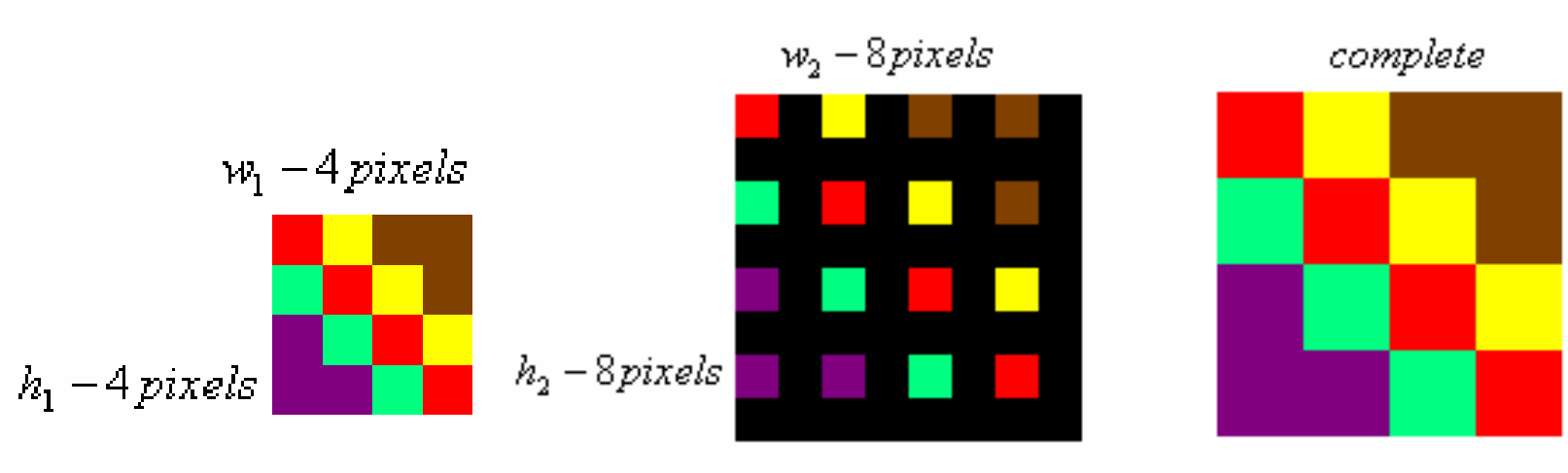
# Enlarging an Image

- Pixel replication

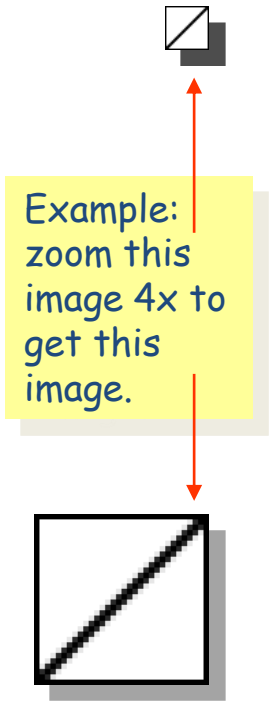
[1 2 3 4 5]

[1 1 2 2 3 3 4 4 5 5] (One step)

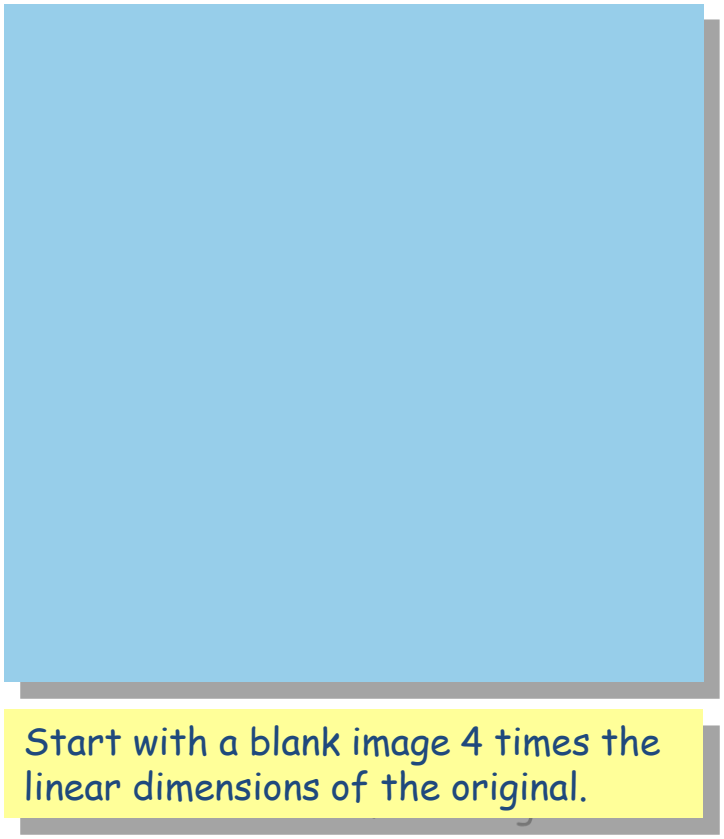
[1 1 1 2 2 2 3 3 3 4 4 4 5 5 5] (Two step)



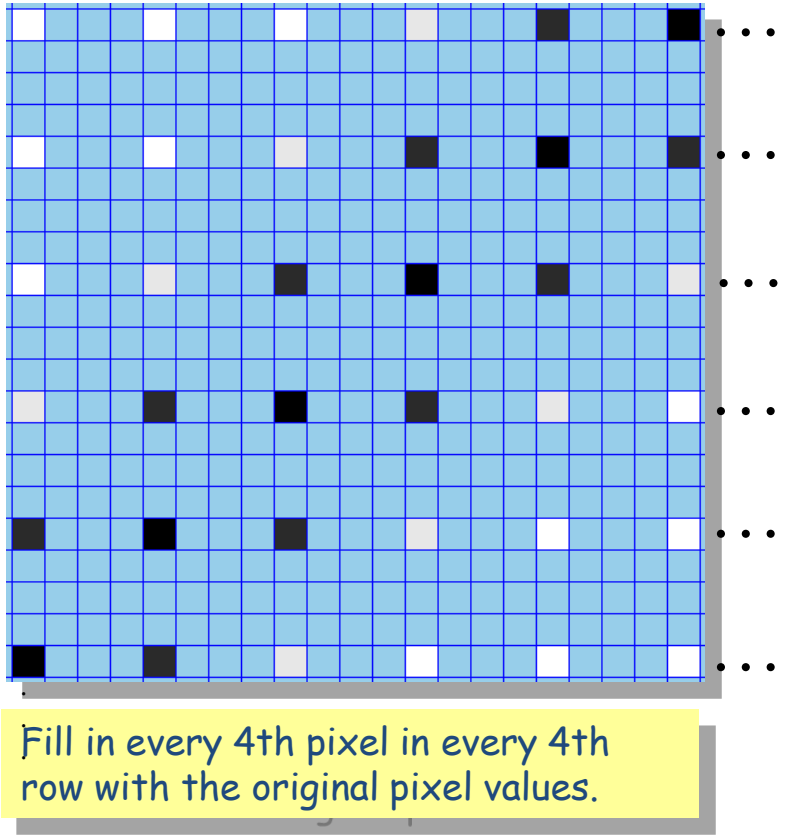
# Enlarging an Image



Example: zoom this image 4x to get this image.

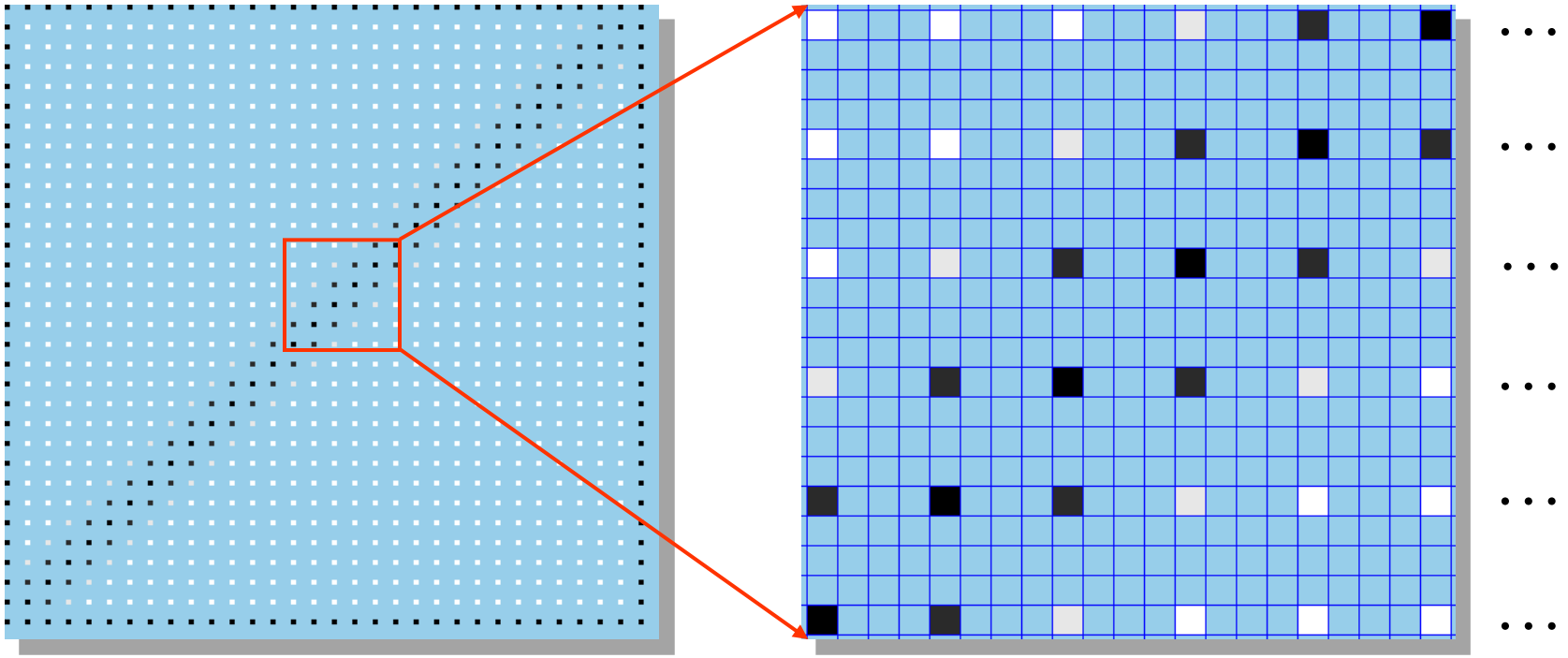


Start with a blank image 4 times the linear dimensions of the original.



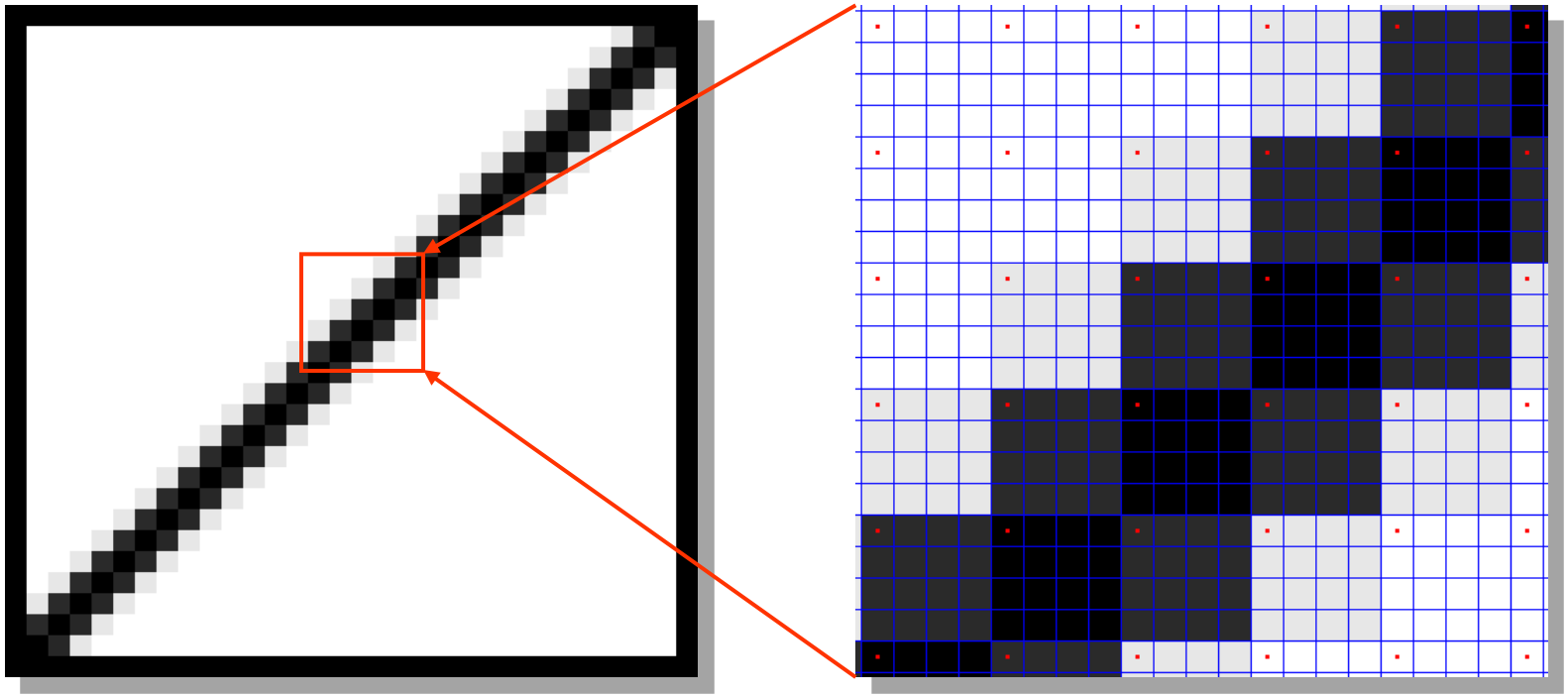
Fill in every 4th pixel in every 4th row with the original pixel values.

# Enlarging an Image



Detail showing every 4th pixel in every 4th row with the original pixel values.

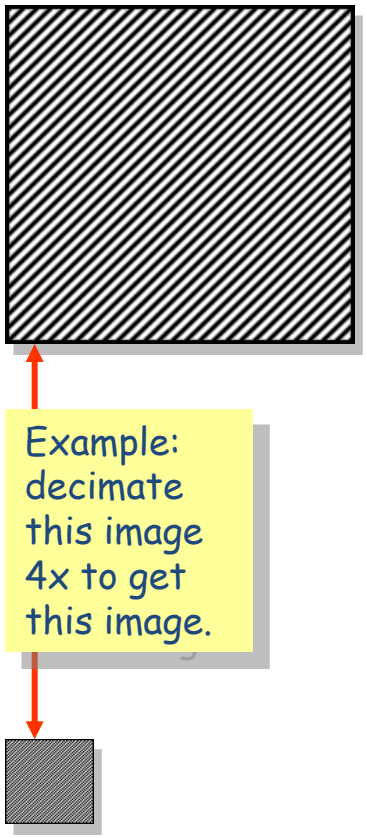
# Enlarging an Image



Replicate the values

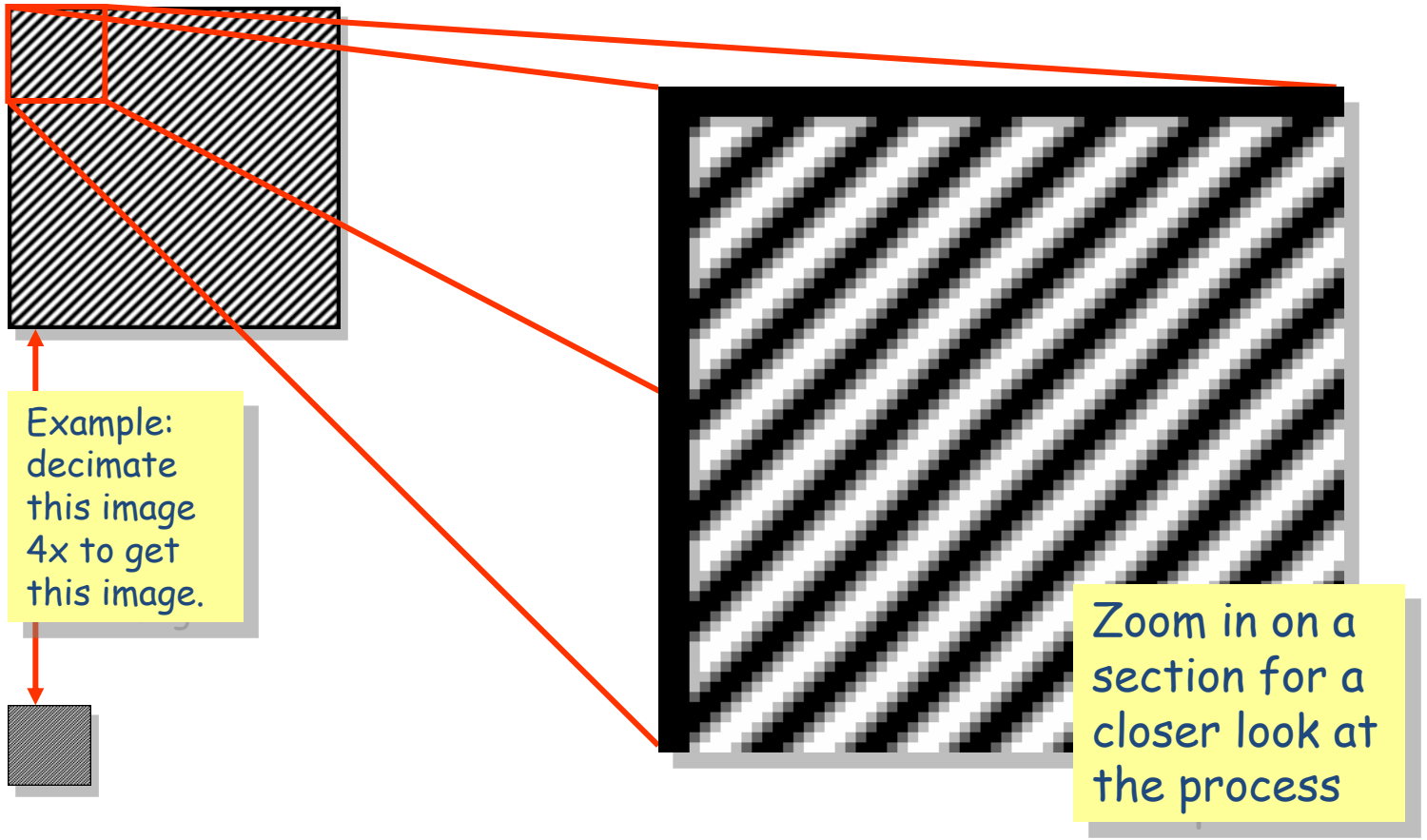
# Reducing an Image

- Pixel Decimation

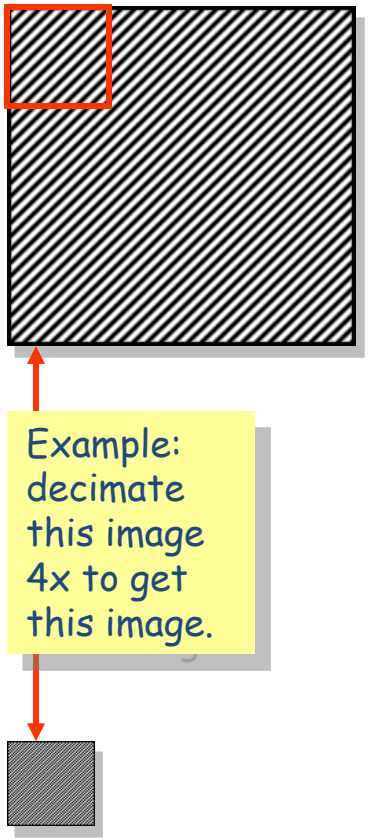



Decimation by a factor of *n*: take every *n*th pixel in every *n*th row

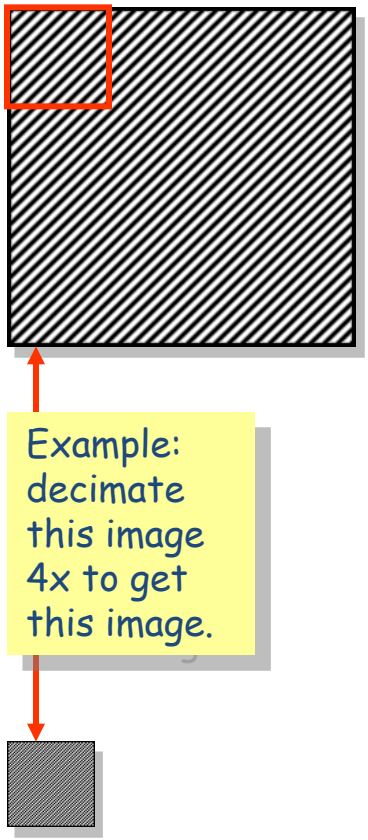
# Reducing an Image

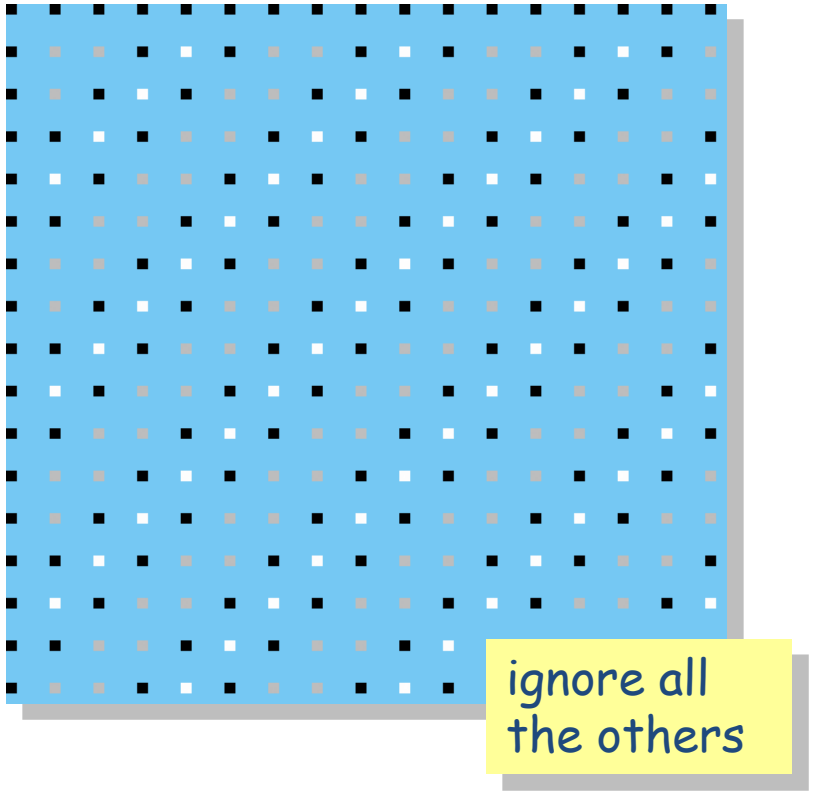


# Reducing an Image

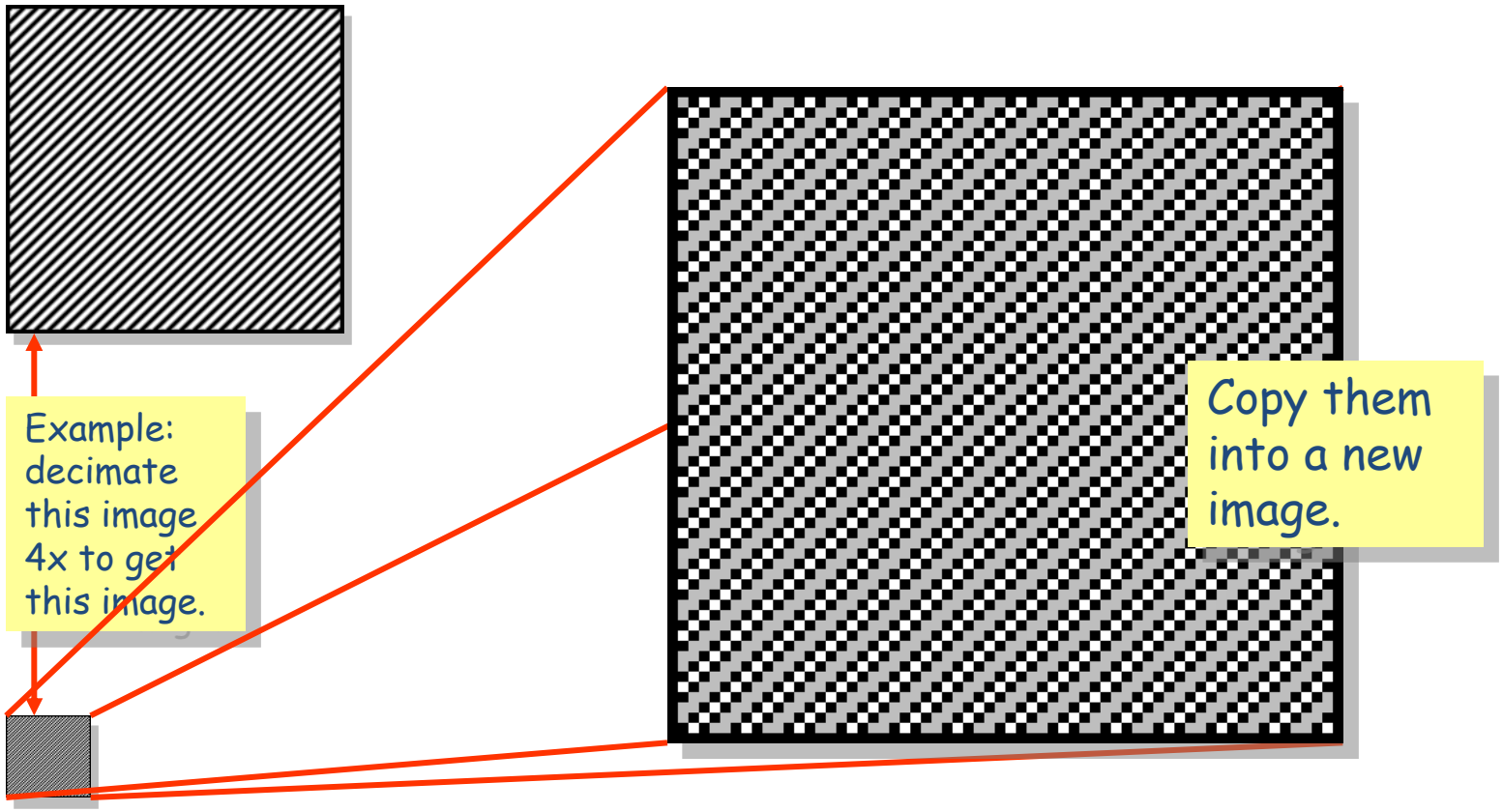



# Reducing an Image



# Reducing an Image



# Nearest Neighbor Interpolation

The "Nearest Neighbor" algorithm is a generalization of pixel replication and decimation.

It also includes fractional resizing, *i.e.* resizing an image so that it has $p/q$ of the pixels per row and $p/q$ of the rows in the original. ($p$ and $q$ are both integers.)

# Image Interpolation

- Nearest neighbour interpolation
  - Simple but produces undesired artefacts
- Bilinear Interpolation
  - Contribution from 4 neighbors
- Bicubic Interpolation
  - Contribution from 16 neighbors

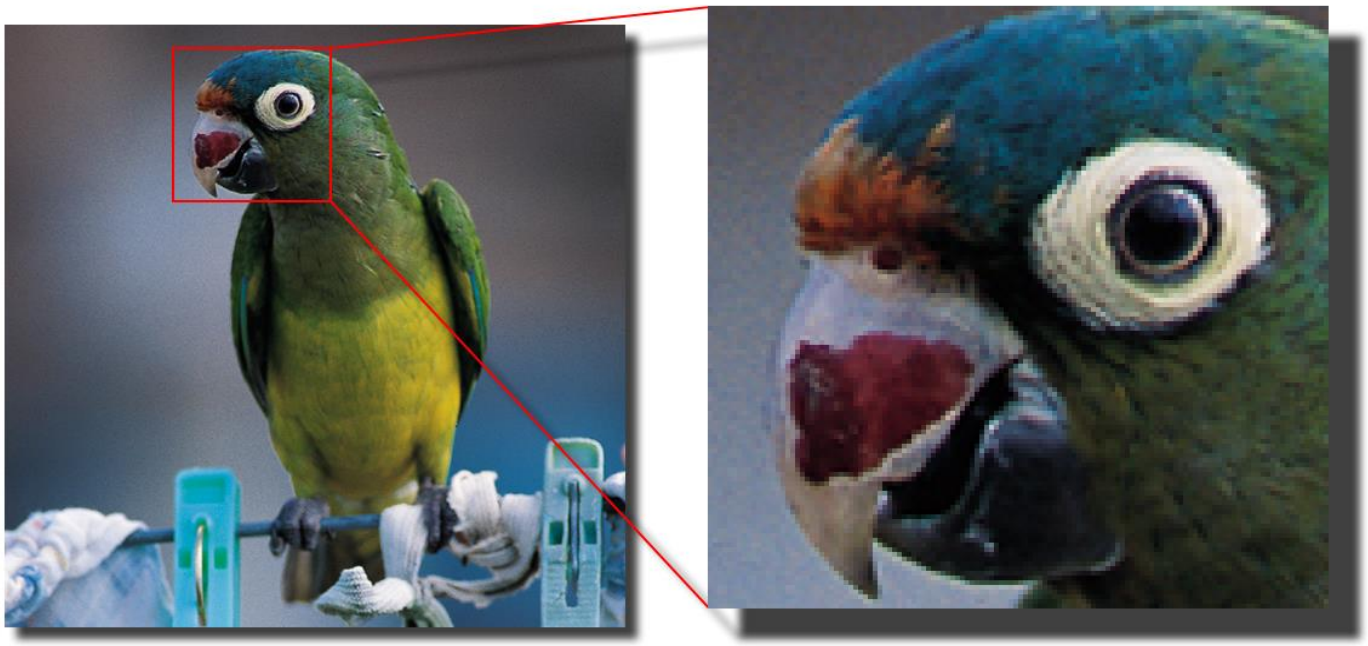
# Interpolation: Comparison

We'll enlarge this image by a factor of 4 …

… via bilinear interpolation and compare it to a nearest neighbor enlargement.

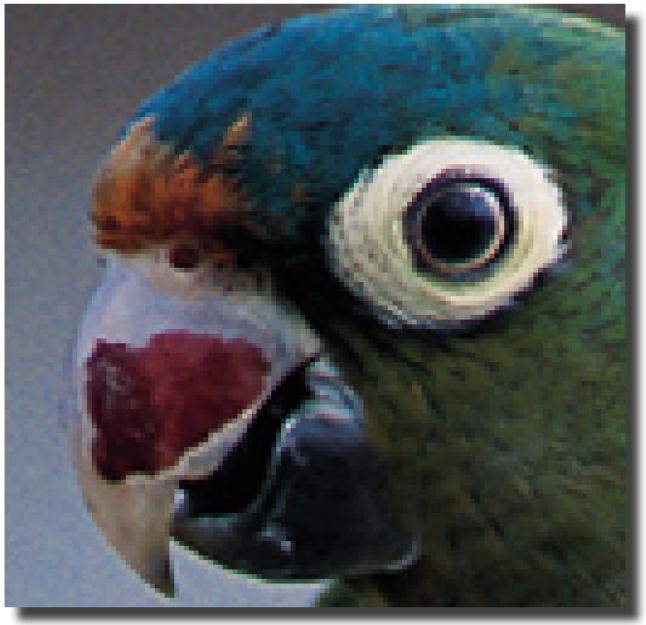
# Interpolation: Comparison

Original Image



To better see what happens, we'll look at the parrot's eye.

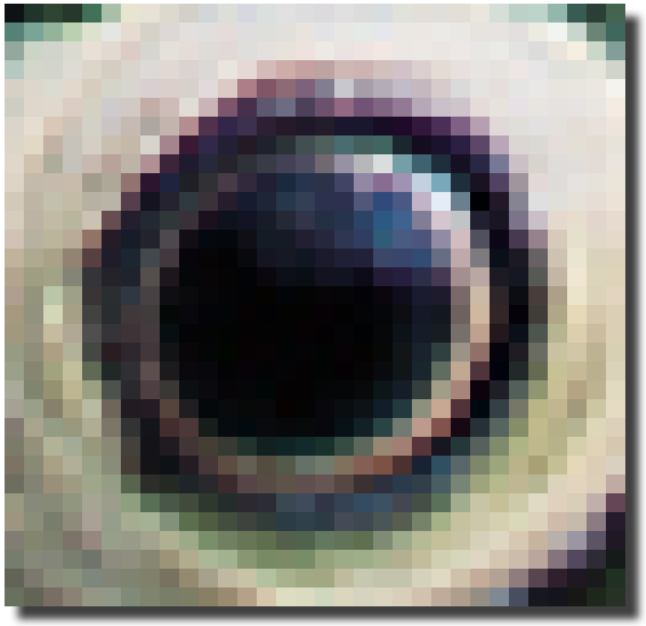
# Interpolation: Comparison



Pixel replication

Bilinear interpolation

# Interpolation: Comparison

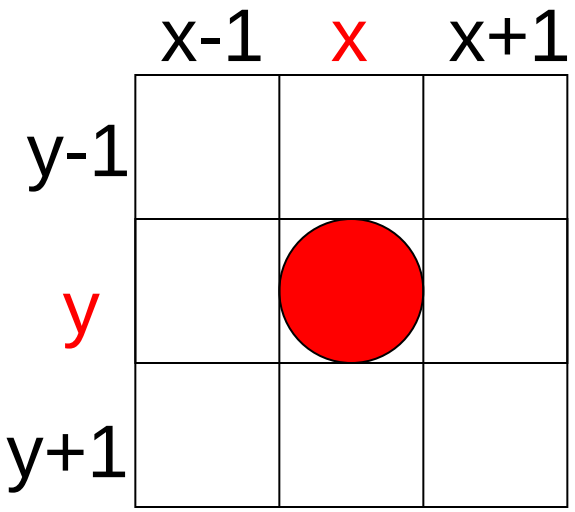


Pixel replication

Bilinear interpolation

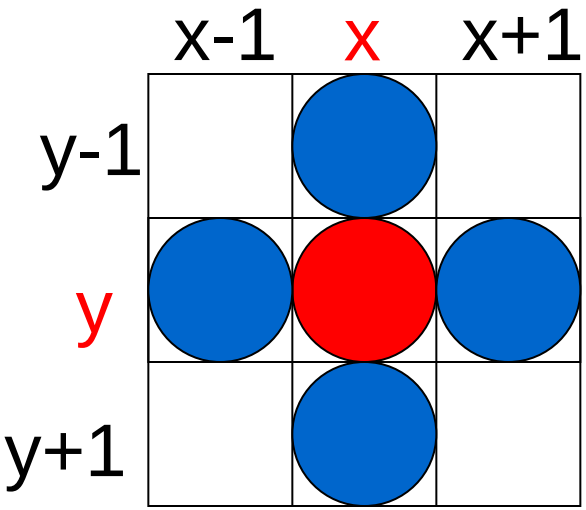
# Relationships between pixels

- Neighbors of pixel are the pixels that are adjacent pixels of an identified pixel



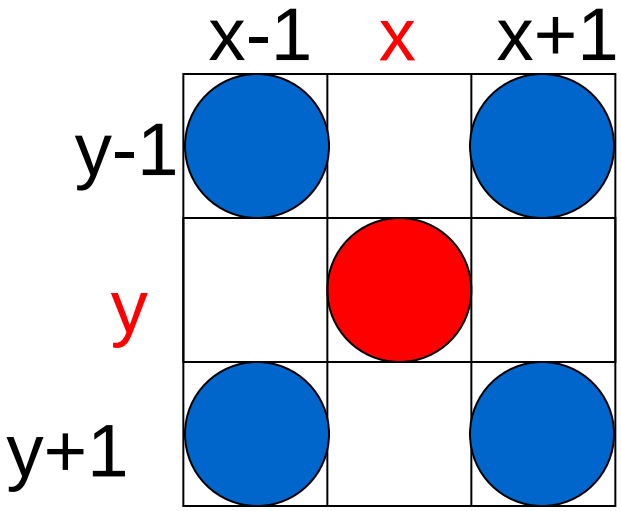
# 4- Neighbors of a Pixel –$N_4(p)$

What are the coordinates of each of the blue pixels



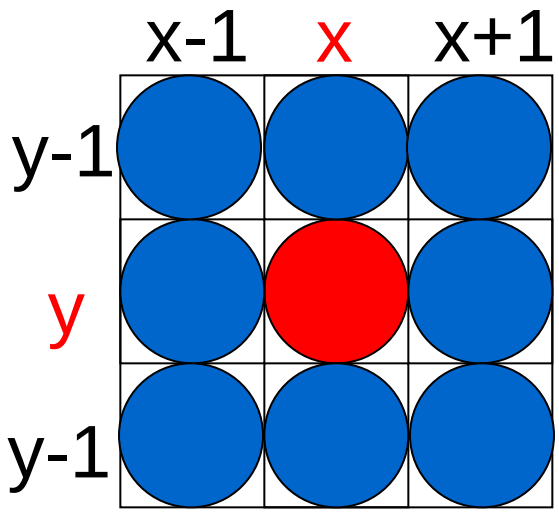

(x-1,y), (x+1,y), (x, y-1), (x, y+1)

# Diagonal Neighbors of a Pixel –$N_D(p)$




(x-1,y-1), (x+1,y-1), (x-1, y+1), (x+1, y+1)

# 8- Neighbors of a Pixel –$N_8(p)$

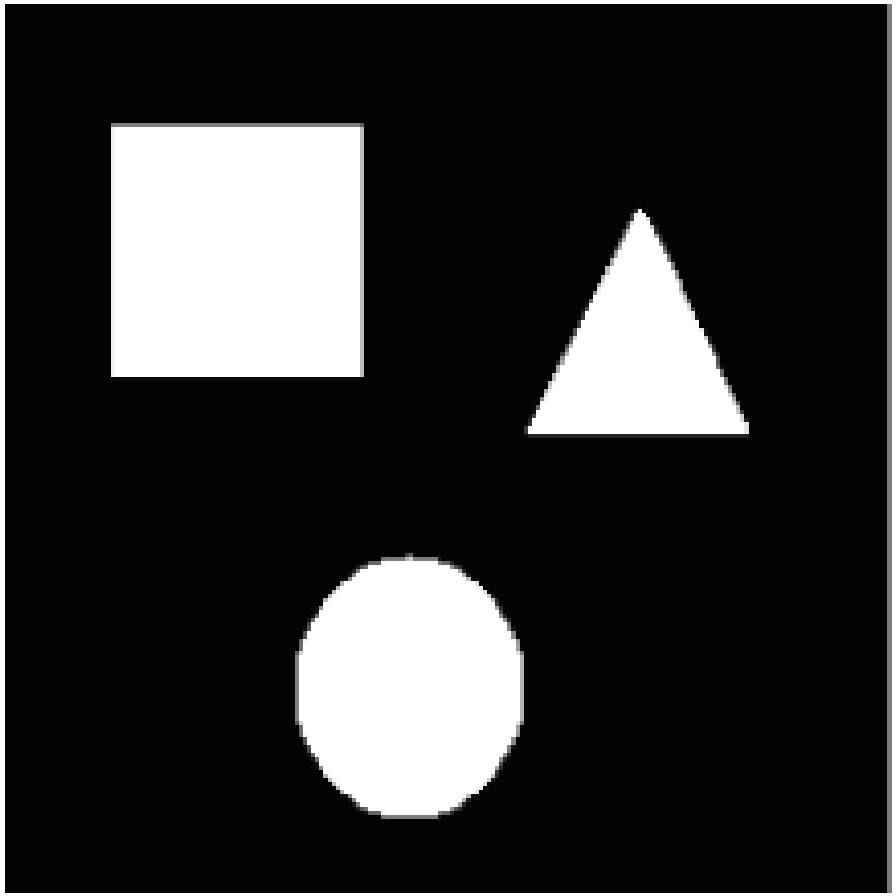



$$N_8(p) = N_4(p) \cup N_D(p)$$

(x-1,y), (x+1,y), (x, y-1), (x, y+1)

(x-1,y-1), (x+1,y-1), (x-1, y+1), (x+1, y+1)

# Determine different regions in the image



# Connectivity

- Establishing boundaries of objects and components in an image
- Group the same region by assumption that the pixels being the same color or equal intensity
- Two pixels p & q are connected if
  - *They are adjacent in some sense*
  - *If their gray levels satisfy a specified criterion of similarity*

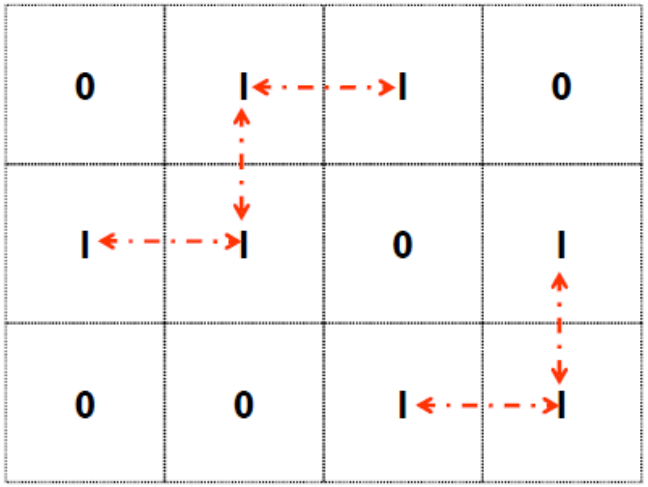
# Connectivity

**V:** Set of gray levels used to define the criterion of similarity

**4-connectivity**

If gray level $(p,q) \in V, and\ q \in N_4(p)$

Set of gray levels V = {1}



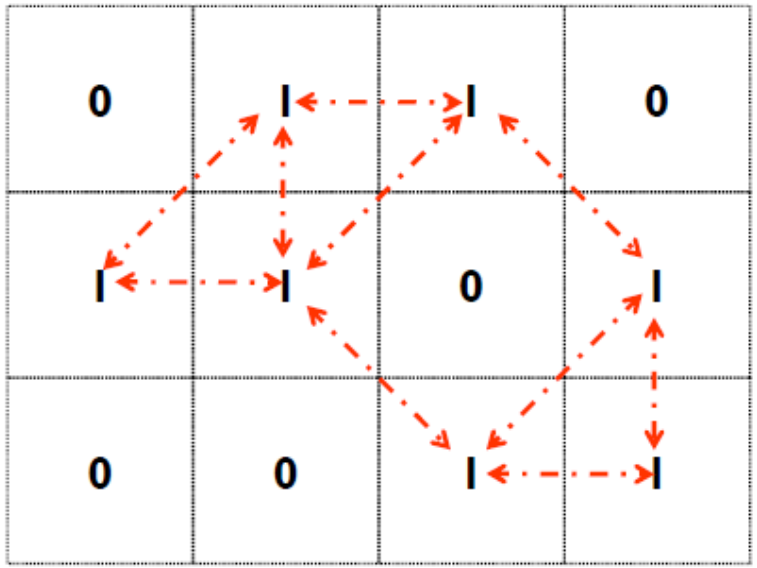
# Connectivity

**V:** Set of gray levels used to define the criterion of similarity

**8-connectivity**

If gray level $(p,q) \in V, and\ q \in N_8(p)$

Set of gray levels V = {1}



# Connectivity

**V:** Set of gray levels used to define the criterion of similarity

**m-connectivity (Mixed Connectivity)**

If gray level

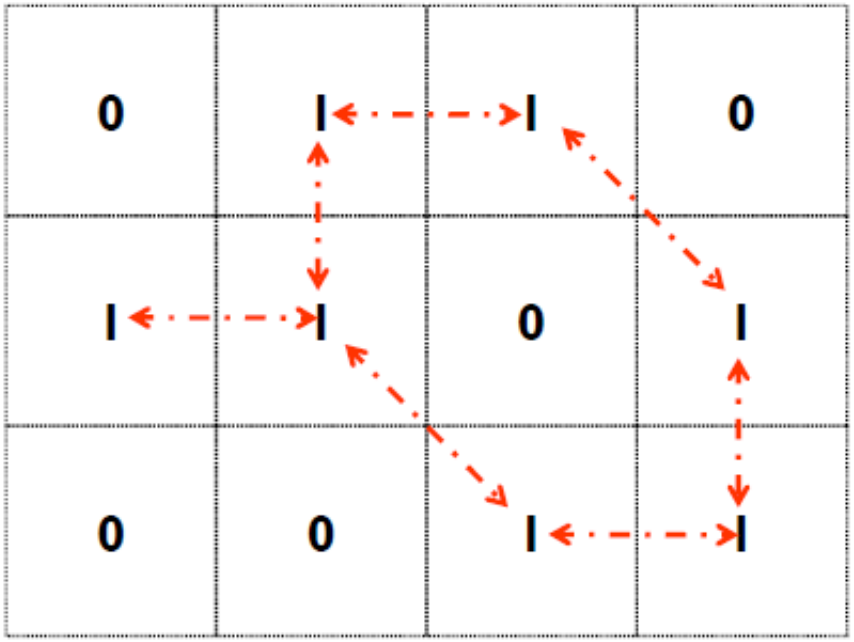
$(p,q) \in V$, *and* $q$ satisfies one of the following:

a. $q \in N_4(p)$ *or*

b. $q \in N_D(p)$ *And* $N_4(p) \cap N_4(q)$ has no pixels whose values are from V

# Example: m – Connectivity

- Set of gray levels V = {1}



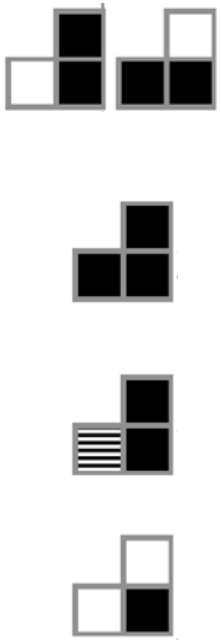

Note: Mixed connectivity can eliminate the multiple path connections that often occurs in 8-connectivity

# Paths

- Path: Let coordinates of pixel *p: (x, y)*, and of pixel *q: (s, t)*
- A *path* from *p* to *q* is a sequence of distinct pixels with coordinates: $(x_0, y_0), (x_1, y_1), \ldots\ldots, (x_n, y_n)$

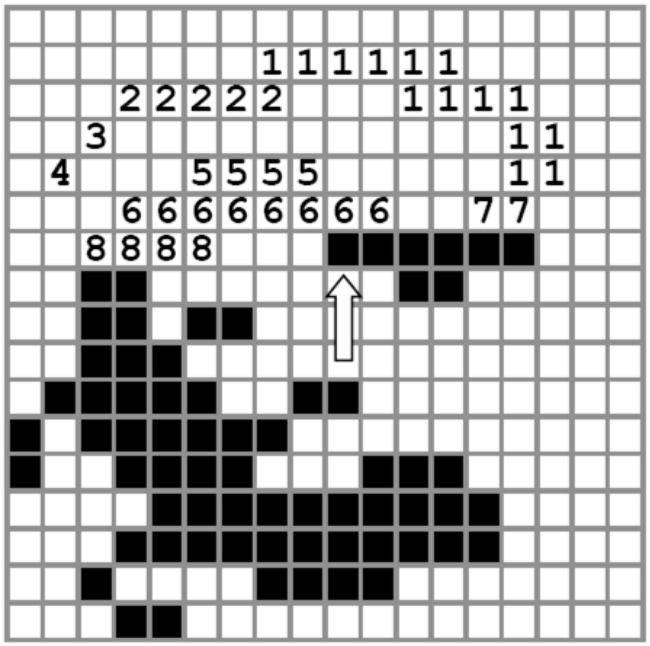
  *where* $(x_0, y_0) = (x, y)$ & $(x_n, y_n) = (s, t)$, *and* $(x_i, y_i)$ *is adjacent to* $(x_{i-1}, y_{i-1})$ $1 \leq i \leq n$

# CC labeling – 4 Connectivity



- Process the image from left to right, top to bottom:
  1.) If the next pixel to process is 1
    i.) If only one of its neighbors (top or left) is 1, copy its label.
    ii.) If both are 1 and have the same label, copy it.
    iii.) If they have different labels
      – Copy the label from the left.
      – Update the equivalence table.
    iv.) Otherwise, assign a new label.

Pass 1

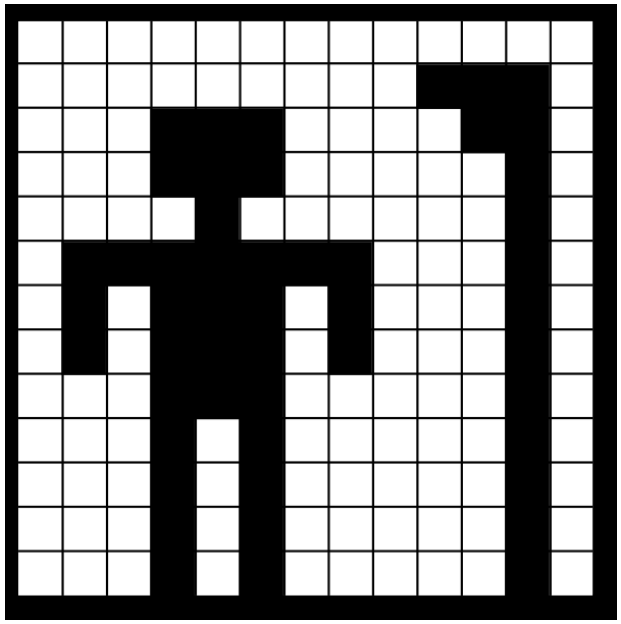


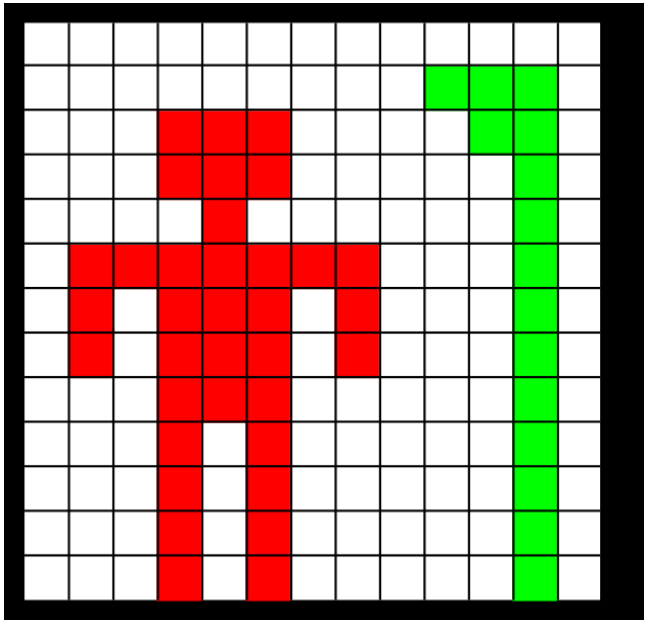
- Re-label with the smallest of equivalent labels

Pass 2

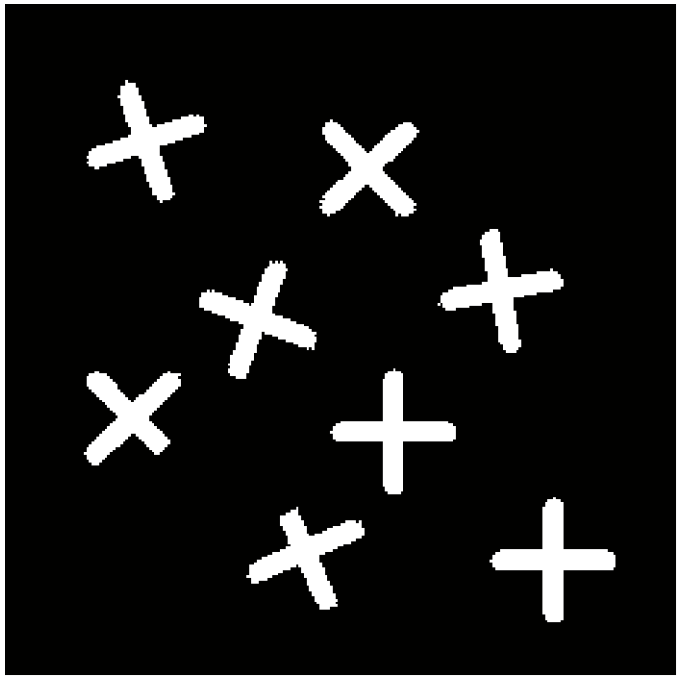
{1, 2, 7}
{3}
{4}
{5, 6, 8}

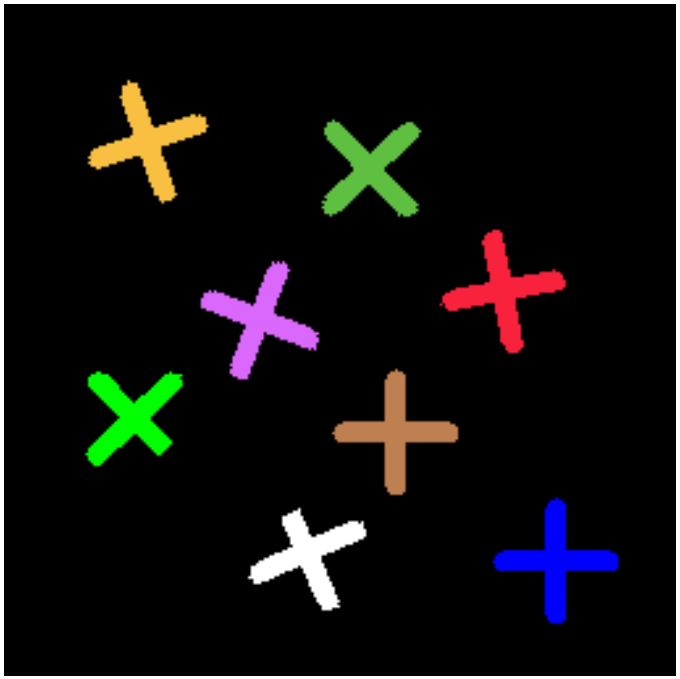
# CC labeling – 4 Connectivity





# CC labeling – 4 Connectivity

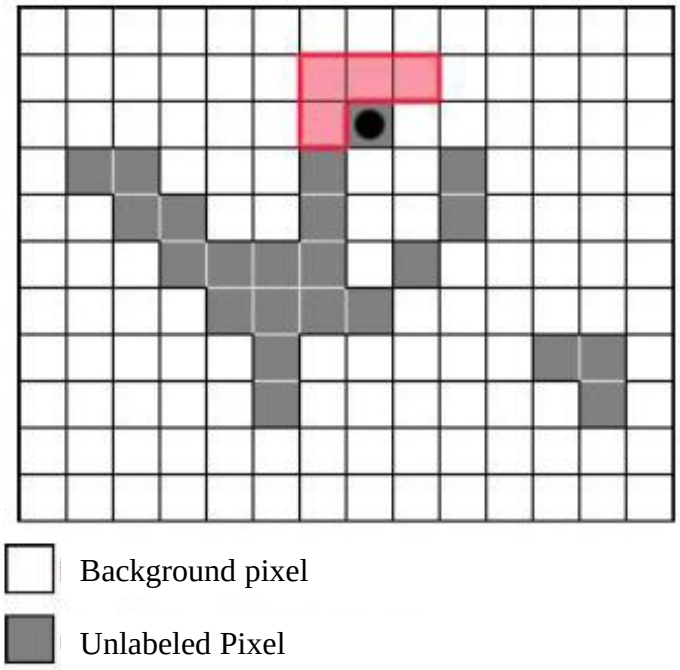


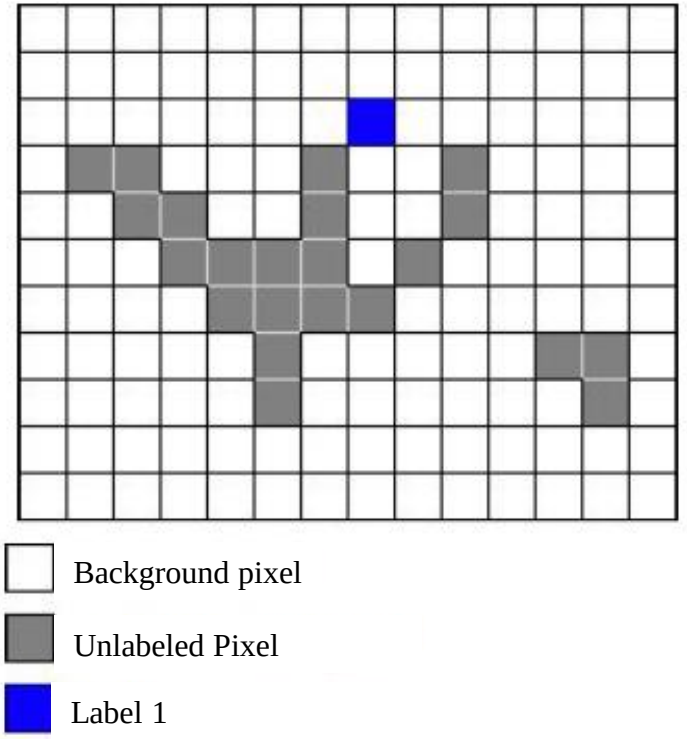


# CC labeling – 8 Connectivity

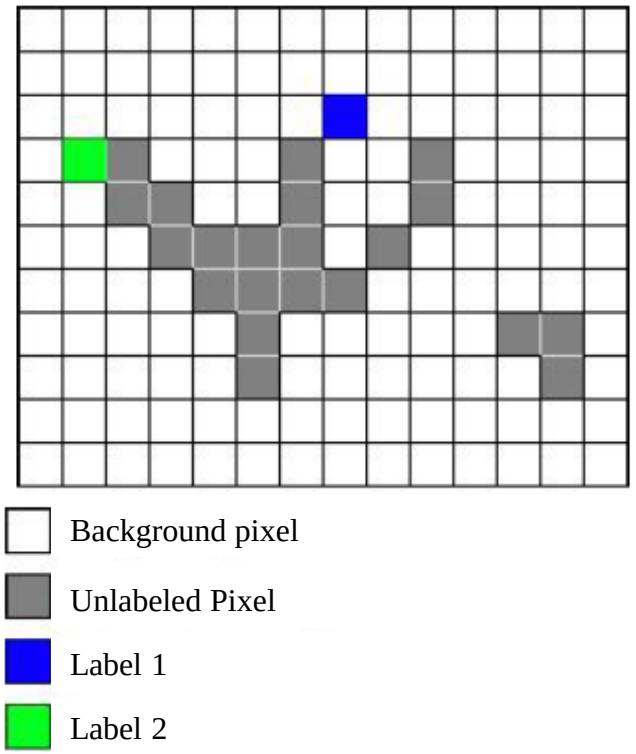
Same algorithm but examine also the upper diagonal neighbors of *p*

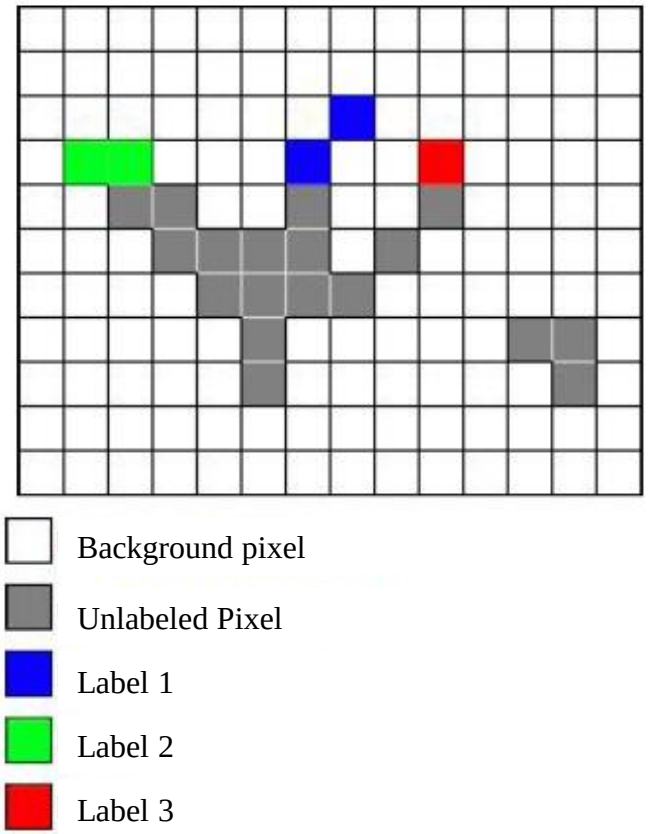
# CC labeling – 8 Connectivity



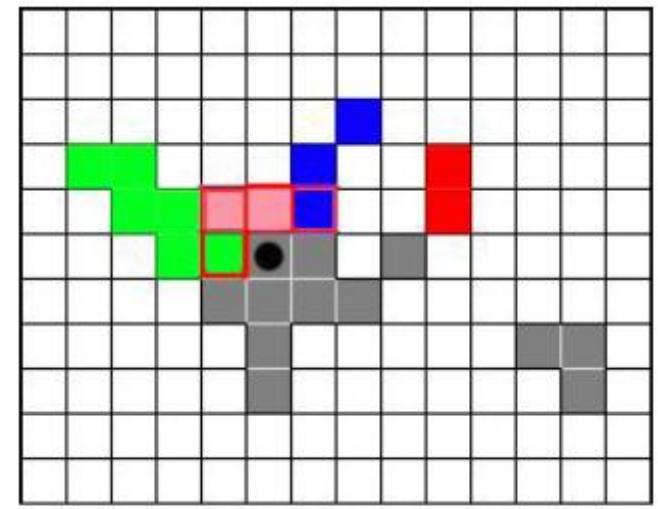



# CC labeling – 8 Connectivity



# CC labeling – 8 Connectivity



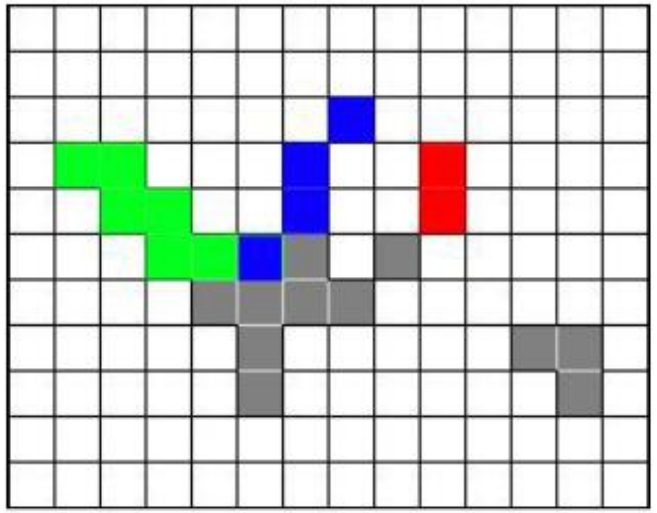
Background pixel

Unlabeled Pixel

Label 1

Label 2

Label 3



Background pixel

Unlabeled Pixel

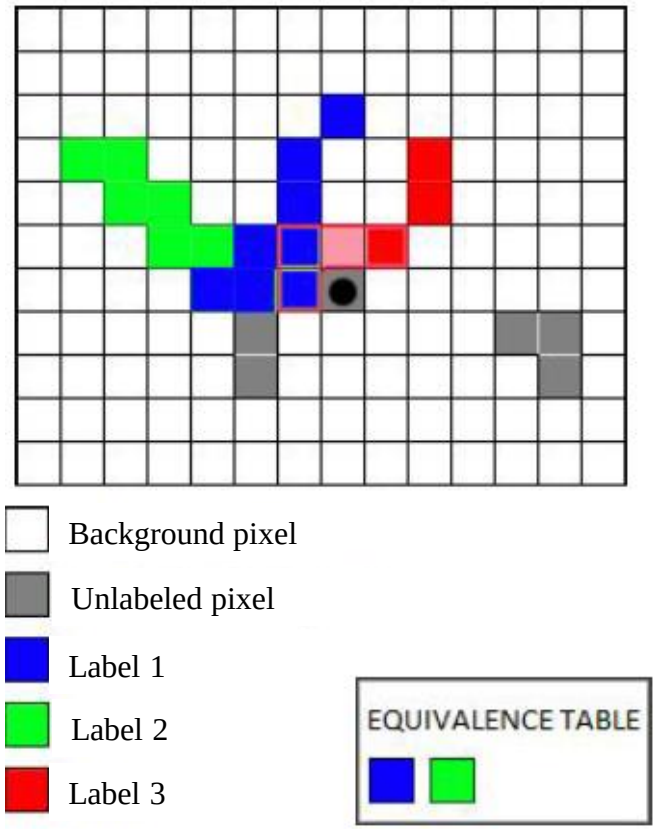
Label 1

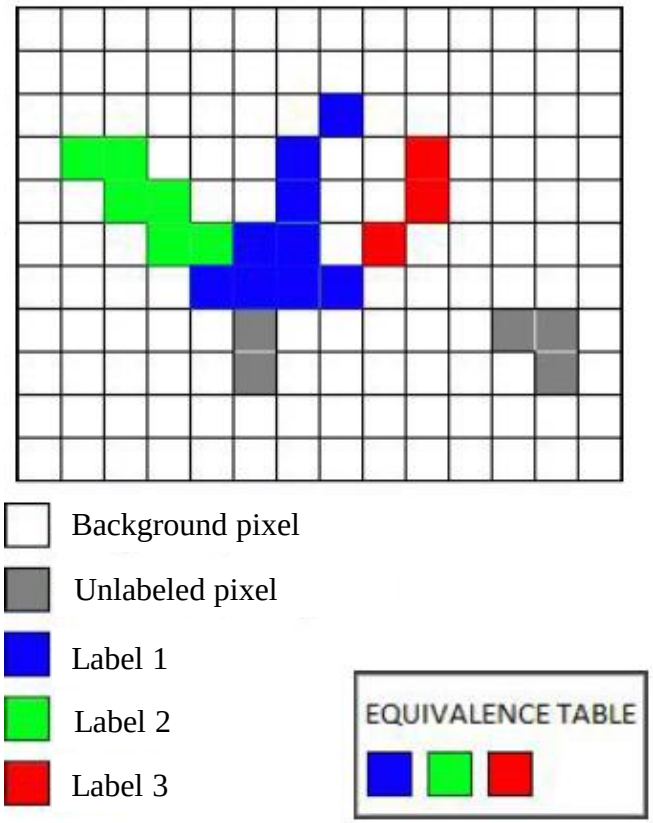
Label 2

Label 3

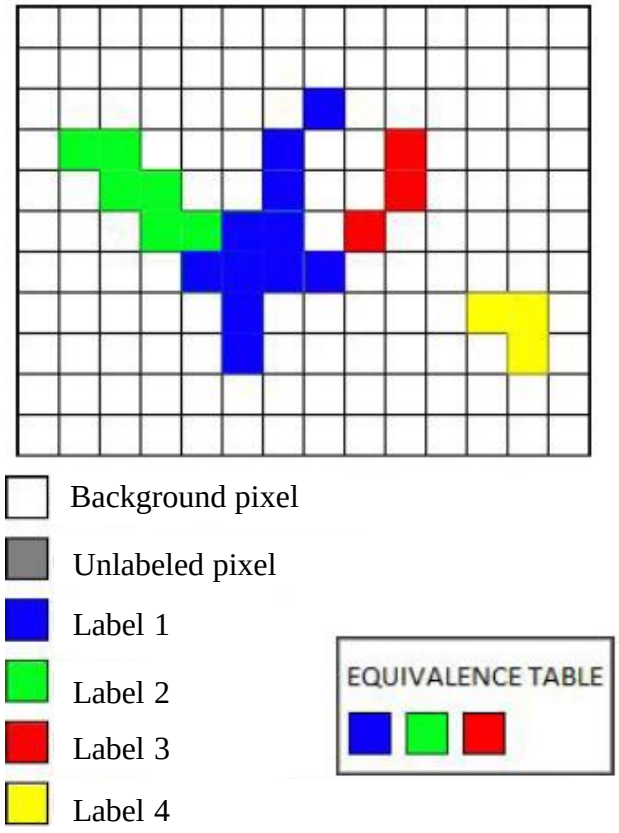
EQUIVALENCE TABLE

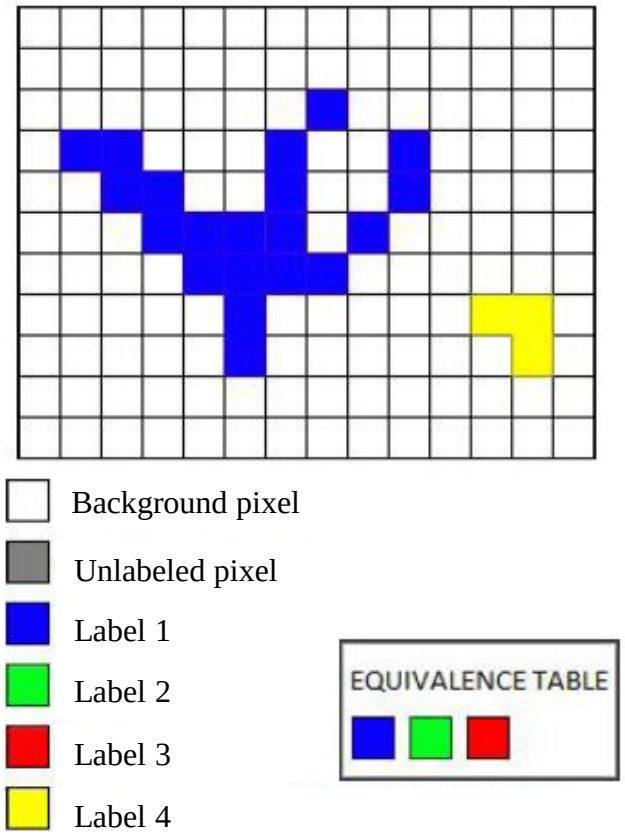
# CC labeling – 8 Connectivity



# CC labeling – 8 Connectivity



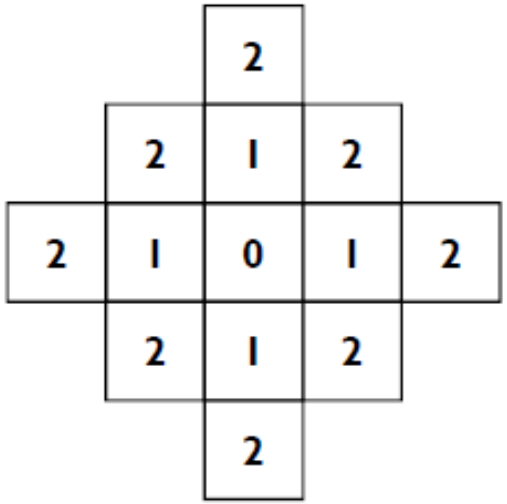
# Distance Metrics

- Let pixels *p*, *q* and *z* have coordinates *(x,y), (s,t)* and *(u,v)* respectively.

- D is a distance function or metric if
  - *D(p,q) ≥ 0 and*
  - *D(p,q) = 0 iff p = q and*
  - *D(p,q) = D(q,p) and*
  - *D(p,z) ≤ D(p,q) + D(q,z)*

# City block distance ($D_4$ distance)

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




- Diamond with center at *(x,y)*
- $D_4$ = 1 are the 4 neighbors of pixel *p(x,y)*

# Chessboard distance ($D_8$ distance)

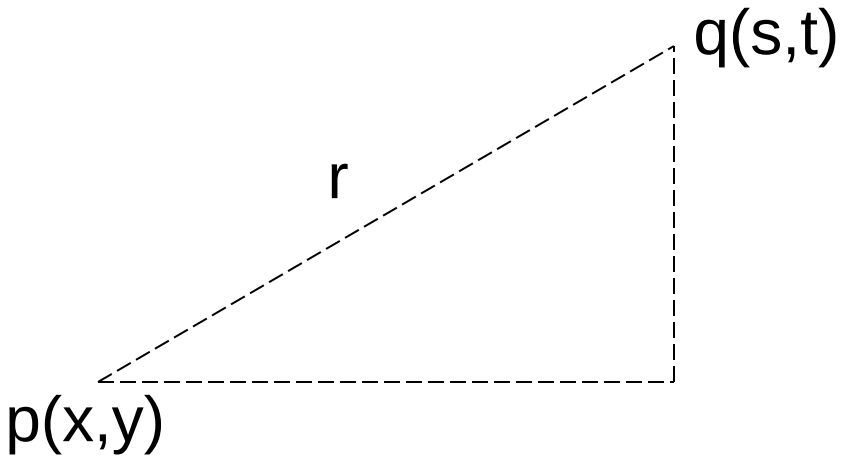
$$D_8(p,q) = \max(|x-s|, |y-t|)$$

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

- Square centered at *p(x,y)*
- $D_8$ = 1 are the 8 neighbors of pixel *p(x,y)*

# Euclidean Distance

$$D_e(p,q) = \sqrt{(x-s)^2 + (y-t)^2}$$




A circle with radius *r* centered at *(x,y)*

# Arithmetic Operations

- Carried out between corresponding pixel pairs

$$s(x,y) = f(x,y) + g(x,y)$$

$$d(x,y) = f(x,y) - g(x,y)$$

$$p(x,y) = f(x,y) \times g(x,y)$$

$$d(x,y) = f(x,y) \div g(x,y)$$

# Arithmetic Operations

- Conversion to range 0 – 255
- Difference of two 8-bit images: -255 to 255
- Sum of two 8-bit images: 0 to 510
- Solution?

Set all values < 0 to 0

Set all values > 255 to 255

Full range of arithmetic operation not captured

# Arithmetic Operations

- First perform the operation

$$f_m = f - \min(f)$$

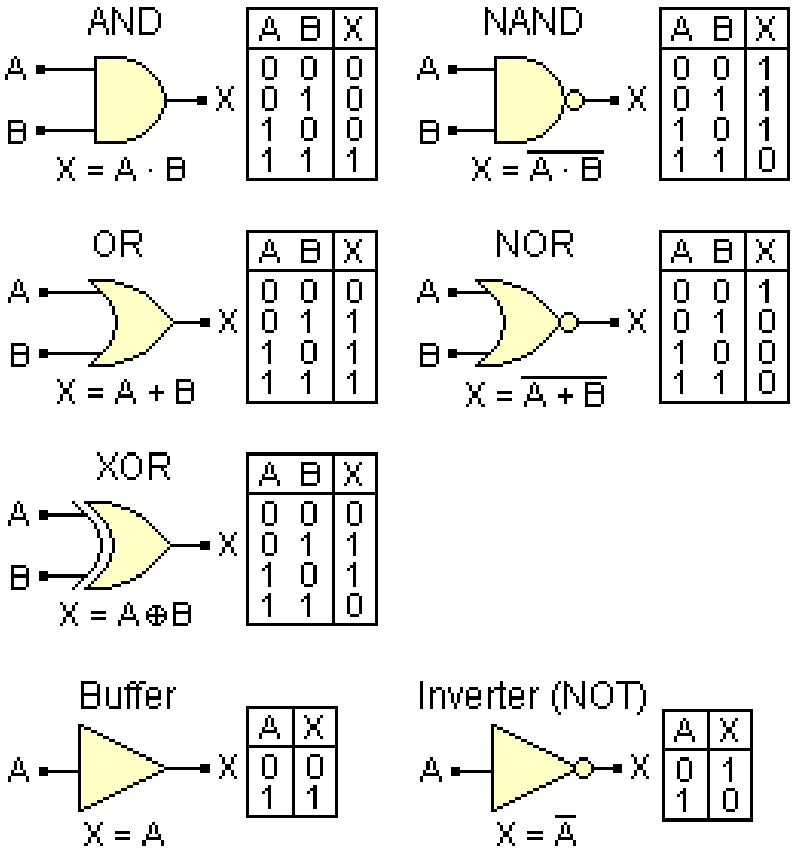
Creates an image whose minimum value is 0

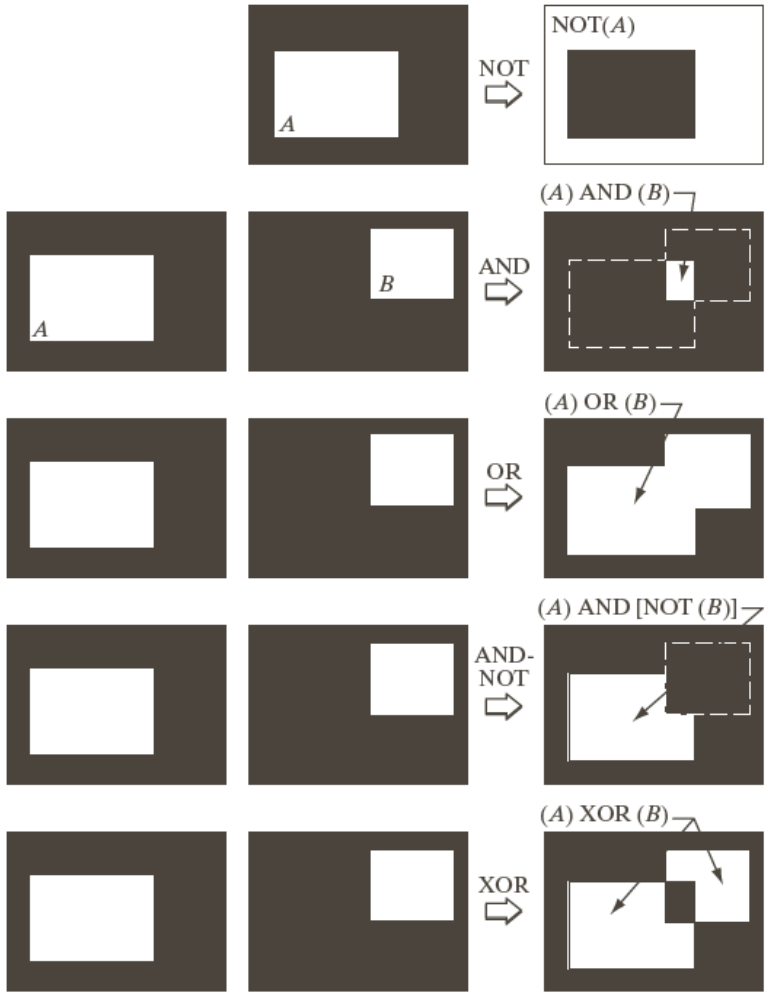
- Then perform

$$f_s = K\left[ f_m / \max(f_m) \right]$$

Creates a scaled image $f_s$ with values in the range [0 K]

# Logical Operations (Binary Images)



# Reading Assignment

- 2.6.1 Array versus Matrix Operations
- 2.6.2 Linear versus Non-Linear Operations

# Readings from Book (3rd Edn.)

- 2.4 Image Sampling & Quantization
- 2.5 Basic Relationships between Pixels
- 3.1 Background
- 3.2 Basic intensity transformations

# Acknowledgements


- Digital Image Processing", Rafael C. Gonzalez & Richard E. Woods, Addison-Wesley, 2002
- Peters, Richard Alan, II, Lectures on Image Processing, Vanderbilt University, Nashville, TN, April 2008
- Brian Mac Namee, Digitial Image Processing, School of Computing, Dublin Institute of Technology
- Computer Vision for Computer Graphics, Mark Borg

Material in these slides has been taken from, the following resources