25.09.2020

Digital Image Processing (CSE/ECE 478)
Lecture-13: Morphological Operations, Intro to Geometric Operations

Ravi Kiran

Announcements

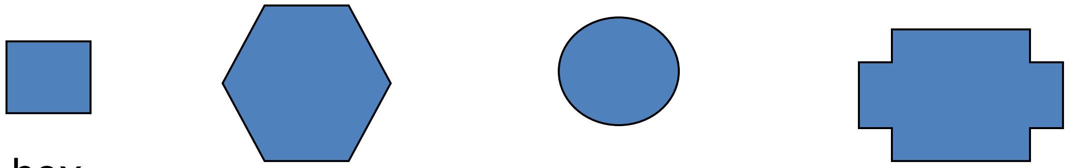
- Projects
  - Form group(s) of >= 1 and <= 3
  - List will be announced at 9pm on 16th September
  - Read guidelines re: project preferences carefully

Binary Images

Structuring Elements

A **structuring element** **= a shape mask**

They can be any shape and size that is digitally representable, and each has an **origin.**



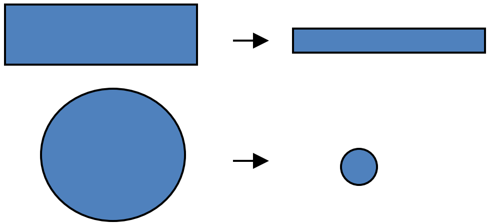
box hexagon disk something

box(length,width) disk(diameter)

Erosion

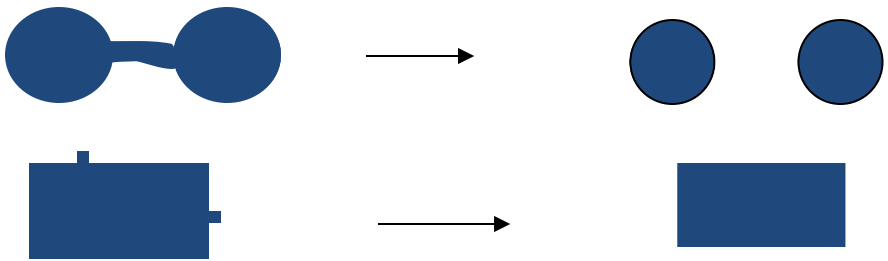
Erosion shrinks the connected sets of 1s of a binary image.

It can be used for



1. shrinking features

2. Removing bridges, branches and small protrusions



Erosion : Operation (min filter)

1	1	1
1	1	1
1	1	1

Set of coordinate points =

{ (-1, -1), (0, -1), (1, -1),

(-1, 0), (0, 0), (1, 0),

(-1, 1), (0, 1), (1, 1) }

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	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	1	1	1	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

→

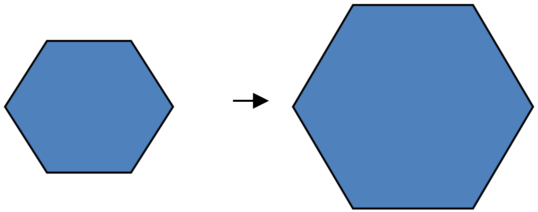
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	1	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	0	0	0	0	0	0	0	0	0	0	0	0	0

Dilation

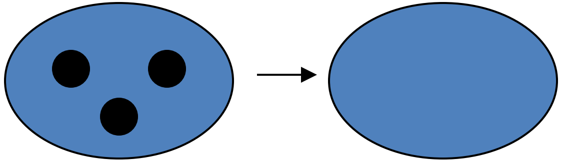
Dilation expands the connected sets of 1s of a binary image.

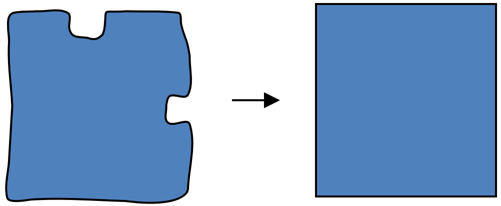
It can be used for

1. growing features



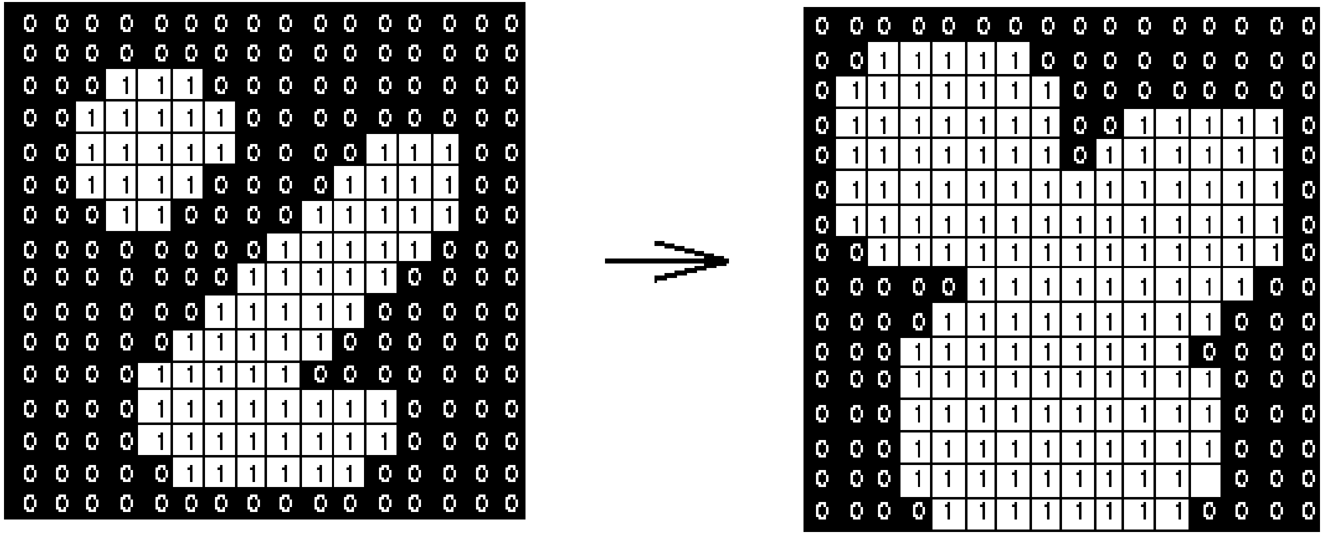
2. filling holes and gaps





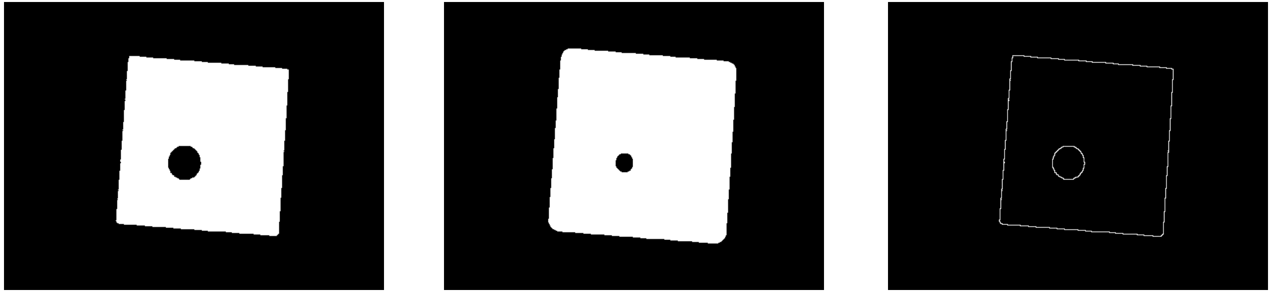
Dilation (max filter)

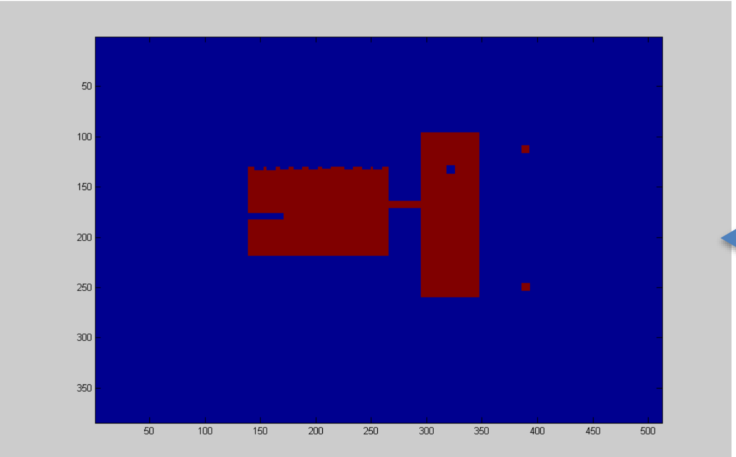
1	1	1
1	1	1
1	1	1



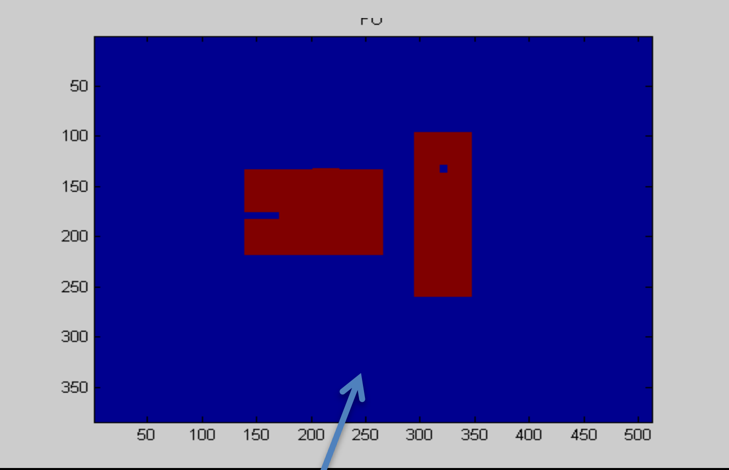
Boundary Detection

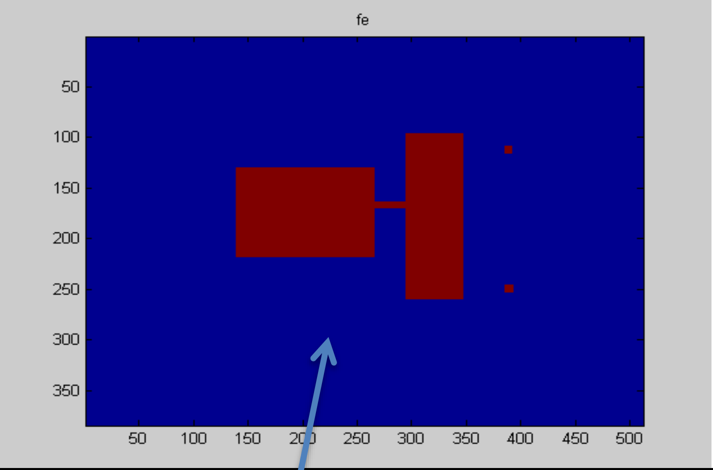
1. Dilate input image
2. Subtract input image from dilated image
3. Boundaries remain!





ORIGINAL





OPENING

CLOSING

Erosion

- Simple application of **pattern matching**
  - **Fixed template**

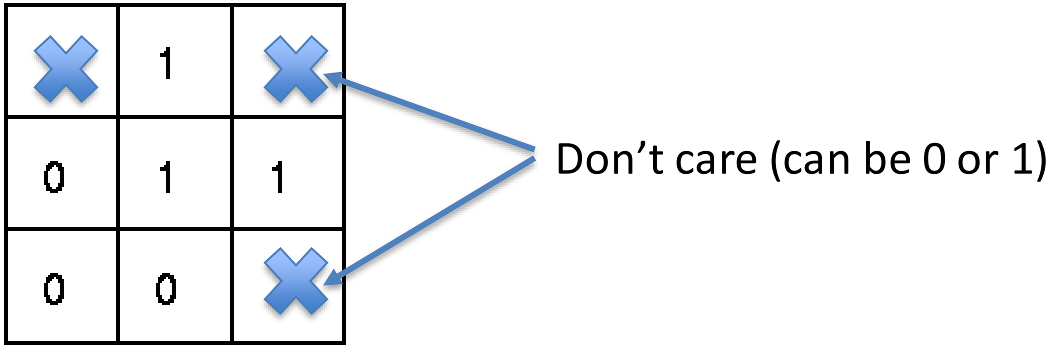
1	1	1
1	1	1
1	1	1

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	0	0	0	0	0	0	0
0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0
0	0	1	1	1	1	1	0	0	0	0	1	1	1	0	0
0	0	1	1	1	1	0	0	0	0	1	1	1	1	0	0
0	0	0	1	1	0	0	0	0	1	1	1	1	1	0	0
0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
0	0	0	0	0	0	0	1	1	1	1	1	0	0	0	0
0	0	0	0	0	0	1	1	1	1	1	0	0	0	0	0
0	0	0	0	0	1	1	1	1	1	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	0	0	0	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	1	1	1	1	1	1	1	1	0	0	0	0
0	0	0	0	0	1	1	1	1	1	1	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	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	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	1	1	1	1	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	0	0	0	0	0	0	0	0	0	0	0	0	0

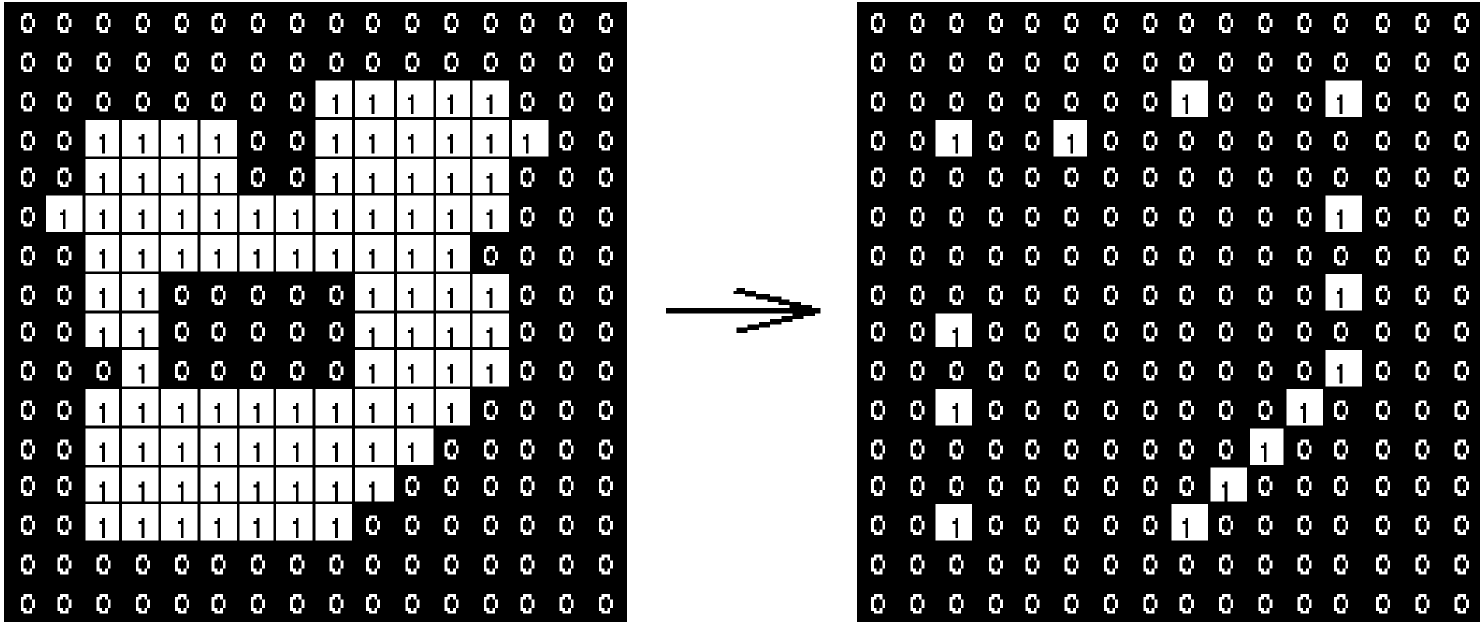
Hit-and-miss Transform

- Look for particular patterns of foreground and background pixels.



- If matched, set pixel = 1

Example: Find right-angled convex corners



Example: Find right-angled convex corners

	1	
0	1	1
0	0	

	1	
1	1	0
	0	0

	0	0
1	1	0
	1	

0	0	
0	1	1
	1	

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	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0
0	0	1	1	1	1	0	0	1	1	1	1	1	1	0	0
0	0	1	1	1	1	0	0	1	1	1	1	1	0	0	0
0	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	0	0	0	0	0	1	1	1	1	0	0	0
0	0	1	1	0	0	0	0	0	1	1	1	1	0	0	0
0	0	0	1	0	0	0	0	0	1	1	1	1	0	0	0
0	0	1	1	1	1	1	1	1	1	1	1	0	0	0	0
0	0	1	1	1	1	1	1	1	1	1	0	0	0	0	0
0	0	1	1	1	1	1	1	1	1	0	0	0	0	0	0
0	0	1	1	1	1	1	1	1	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	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	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	0	0	0	0	1	0	0	0	1	0	0	0
0	0	1	0	0	1	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	0
0	0	0	0	0	0	0	0	0	0	0	0	1	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	0	0	0	0	0	0	1	0	0	0
0	0	1	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	1	0	0	0
0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	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	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Sample HAM transforms

1)

0	0	0
0	1	0
0	0	0

Locate isolated points

Sample HAM transforms

1)

0	0	0
0	1	0
0	0	0

Locate isolated points

2)

0	1	0
0	0	0

Locate end points on thin lines

Sample HAM transforms

1)

0	0	0
0	1	0
0	0	0

Locate isolated points

2)

0	1	0
0	0	0

Locate end points on thin/tapering structures

3a)

	1	
	1	
1		1

3b)

1		
	1	
1		1

3c)

*	0	1
1	1	0
*	1	*

Locate triple junctions

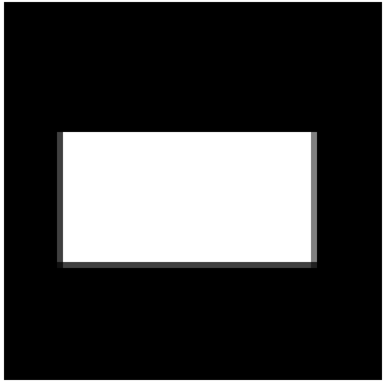
Distance Transform
(with chessboard distance metric)

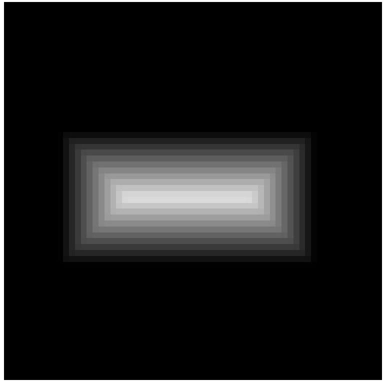
0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0

Binary Image

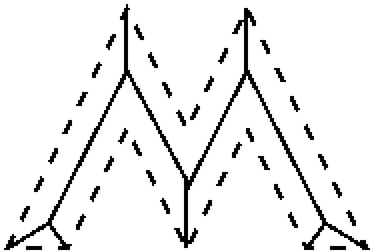
0	0	0	0	0	0	0
0	1	1	1	1	1	0
0	1	2	2	2	1	0
0	1	2	3	2	1	0
0	1	2	2	2	1	0
0	1	1	1	1	1	0
0	0	0	0	0	0	0

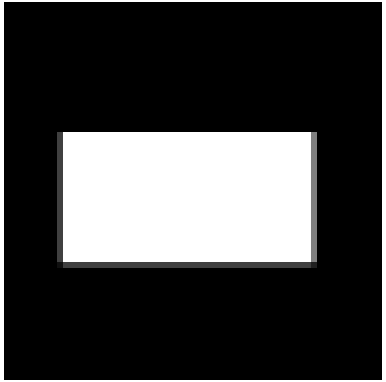
Distance transformation

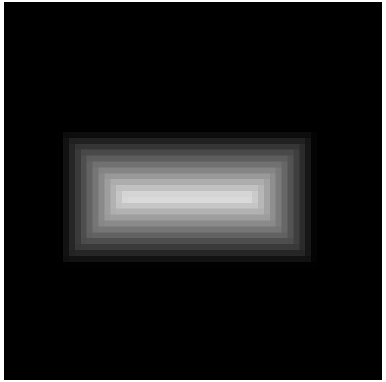


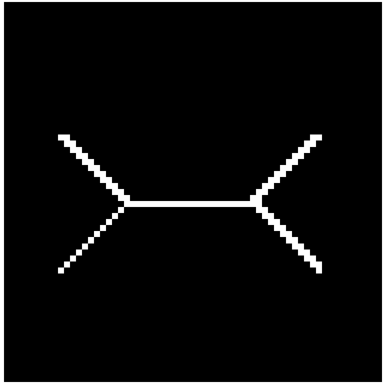


Skeletonization

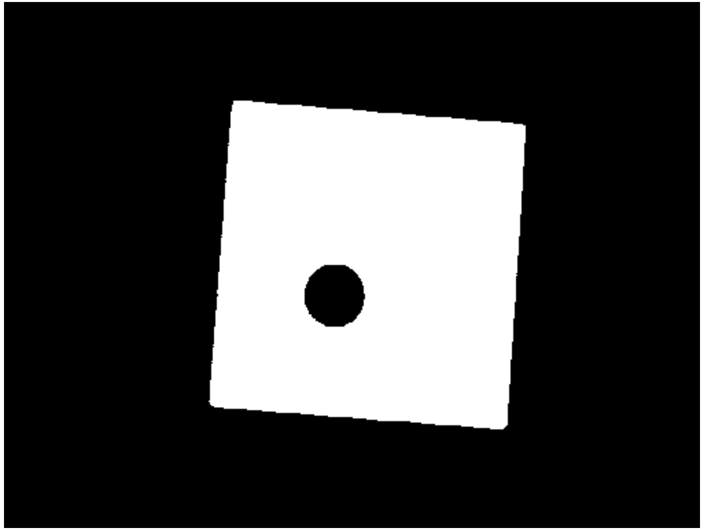


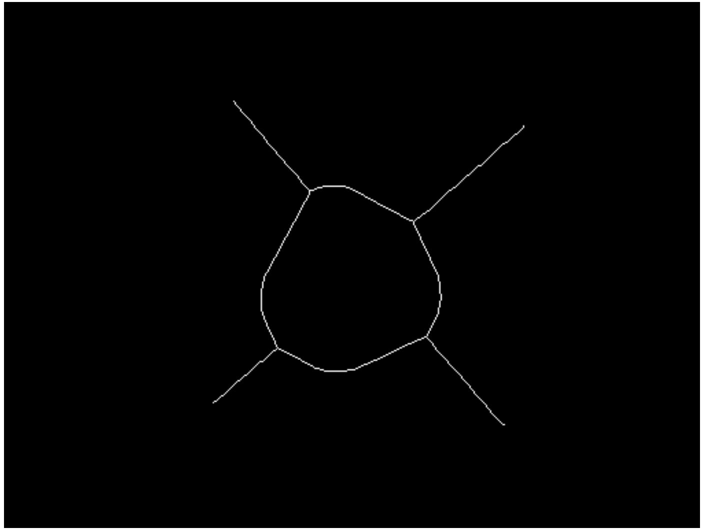




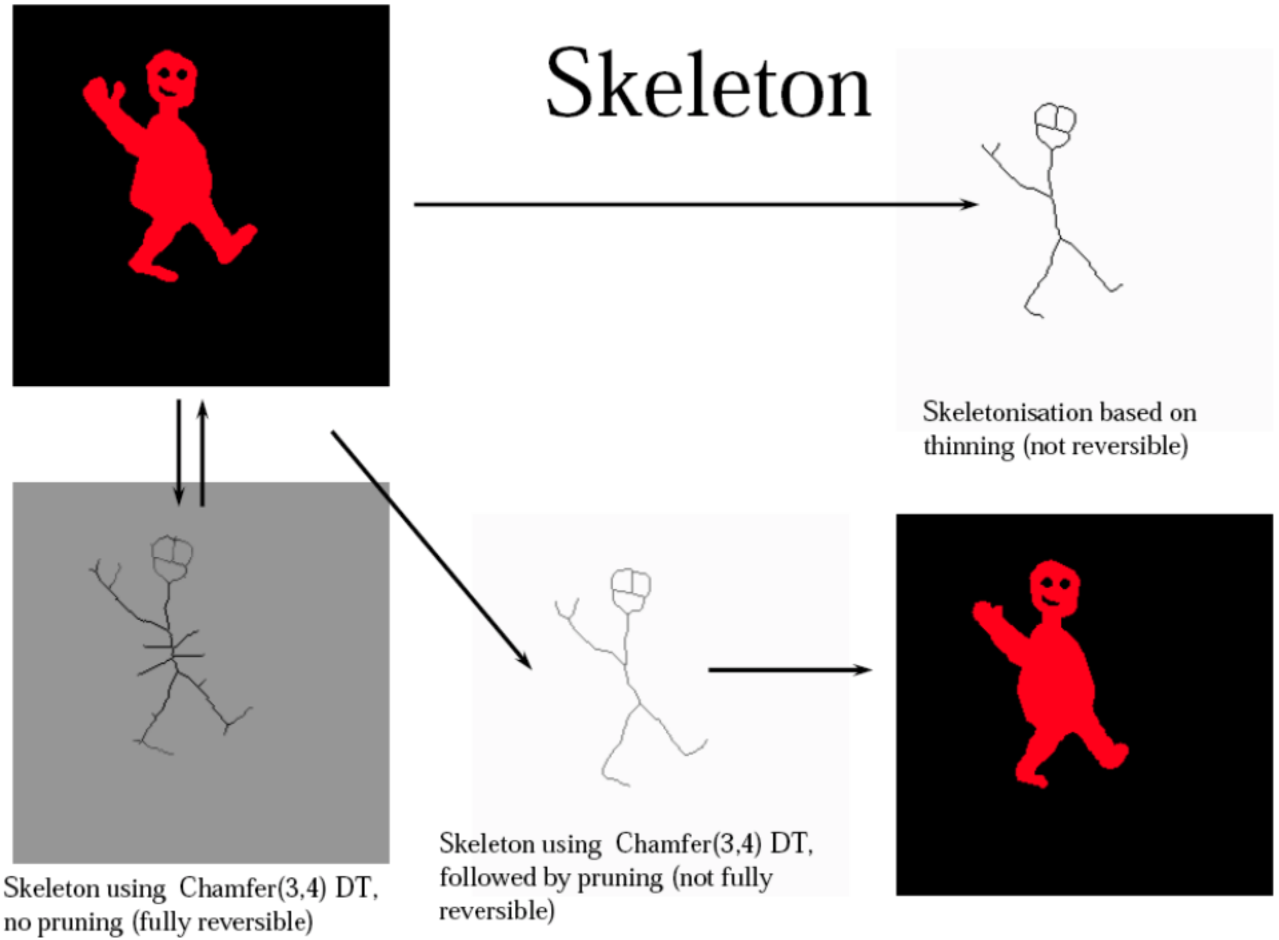


Skeletonization - Example

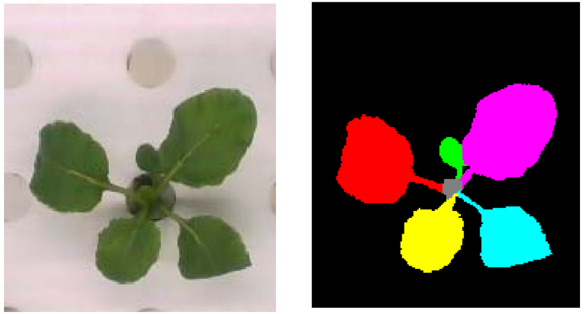


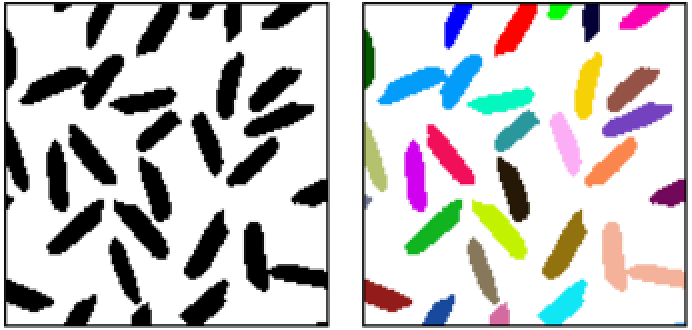


Skeleton

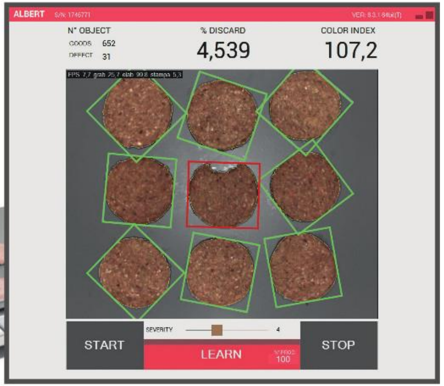

Skeleton
Skeletonisation based on thinning (not reversible)
Skeleton using Chamfer(3,4) DT, no pruning (fully reversible)
Skeleton using Chamfer(3,4) DT, followed by pruning (not fully reversible)

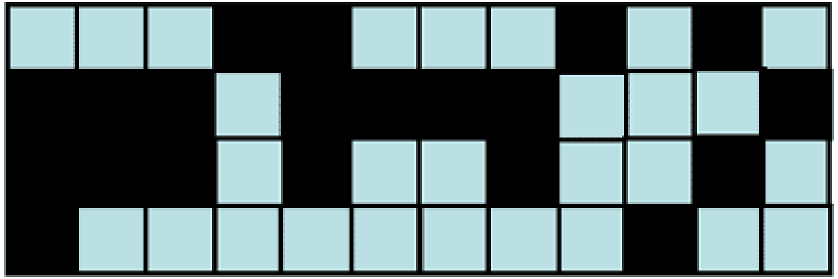
Components



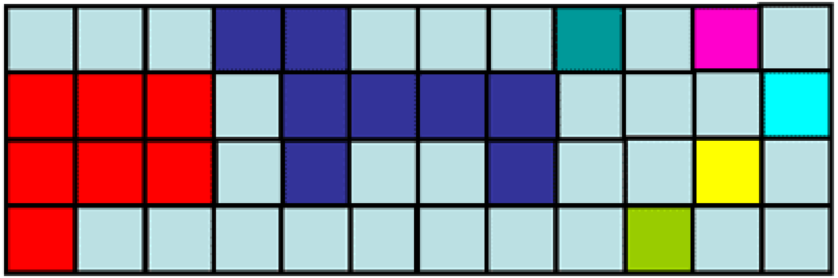



ALBERT
OPTO ENGINEERING
SENSURE

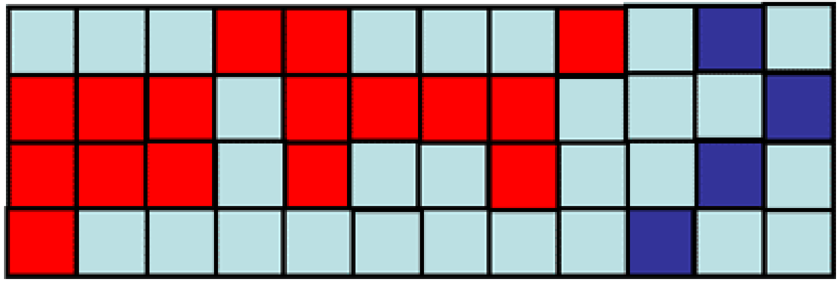

ALBERT
N° OBJECT
GOODS 652
DEFECT 31
% DISCARD
4,539
COLOR INDEX
107,2
START
SEVERITY
LEARN
STOP



Binary image:
0 - objects;
1 -background

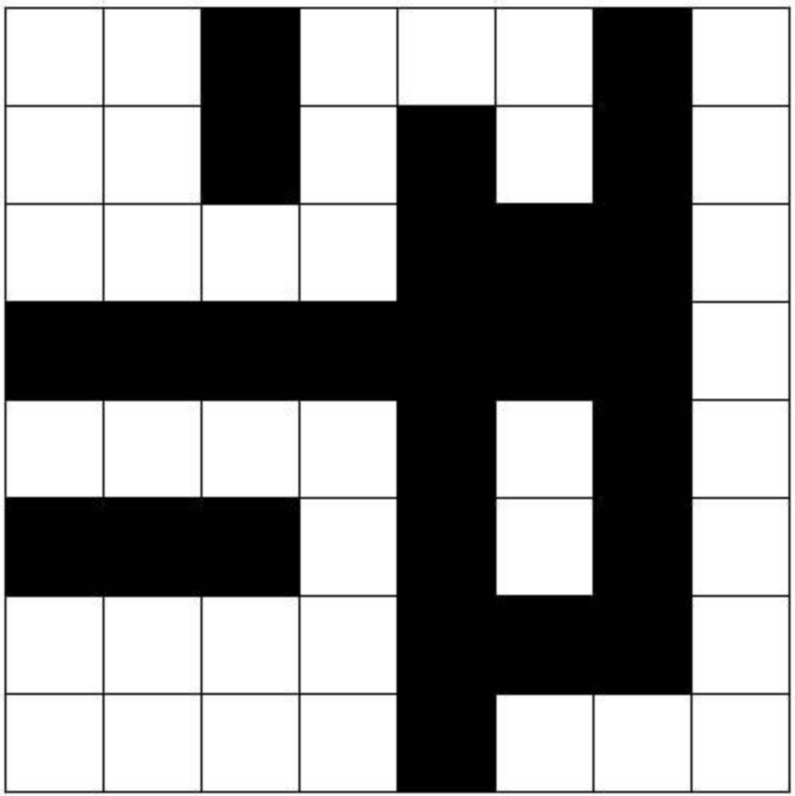


4-connected
objects +
8-connected
background

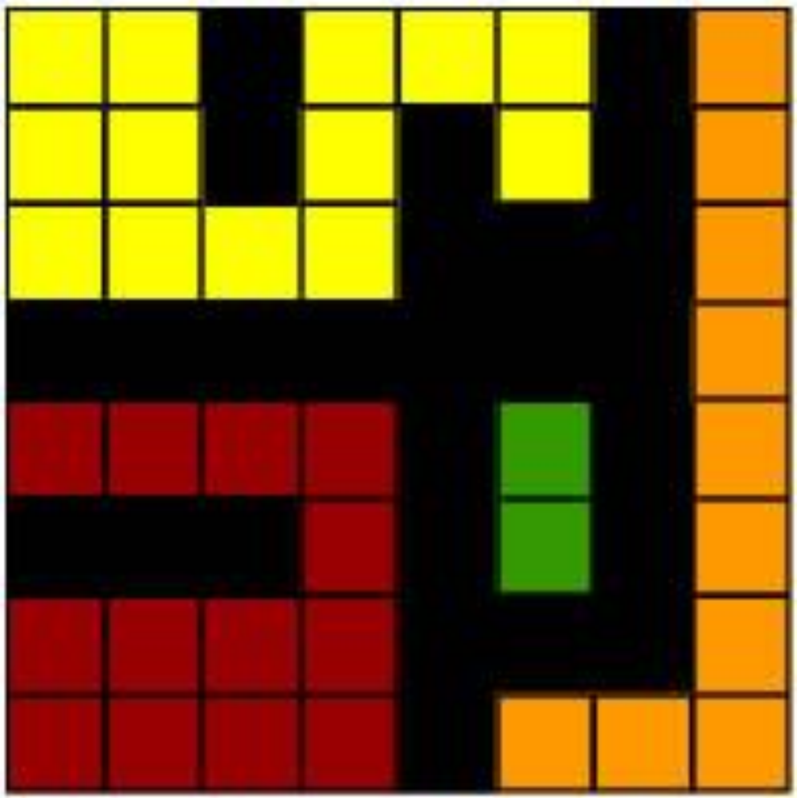


8-connected
objects +
8-connected
background

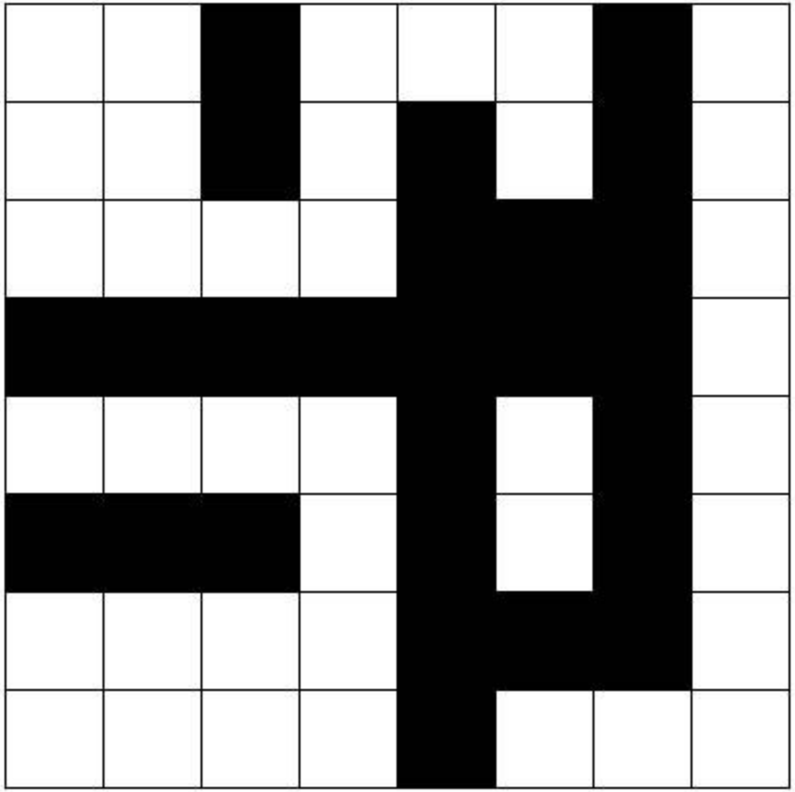
Two-Pass Algorithm for Connected Component Labelling



Two-Pass Algorithm for CCL

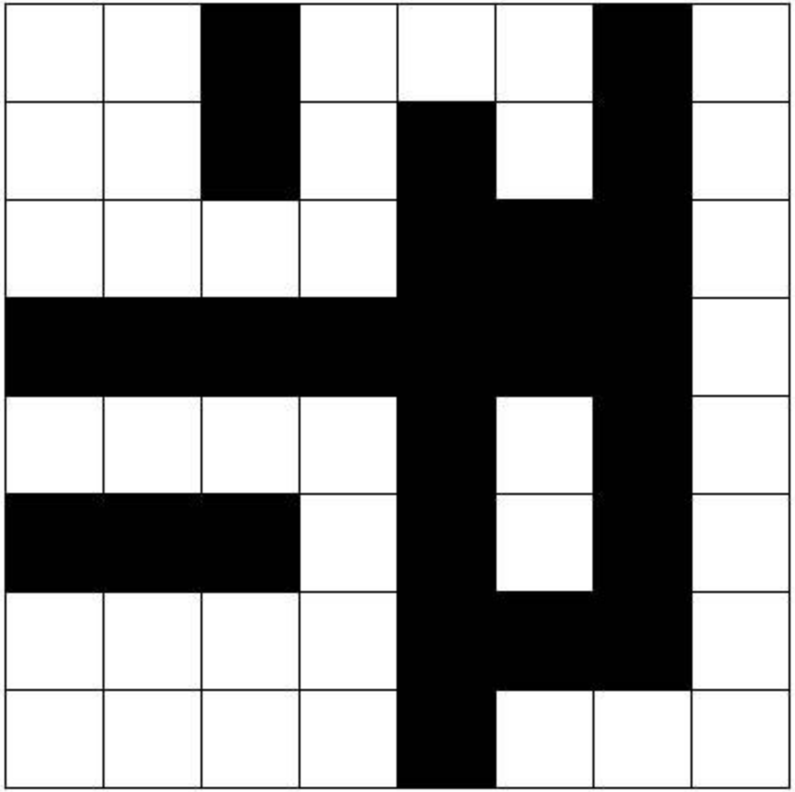


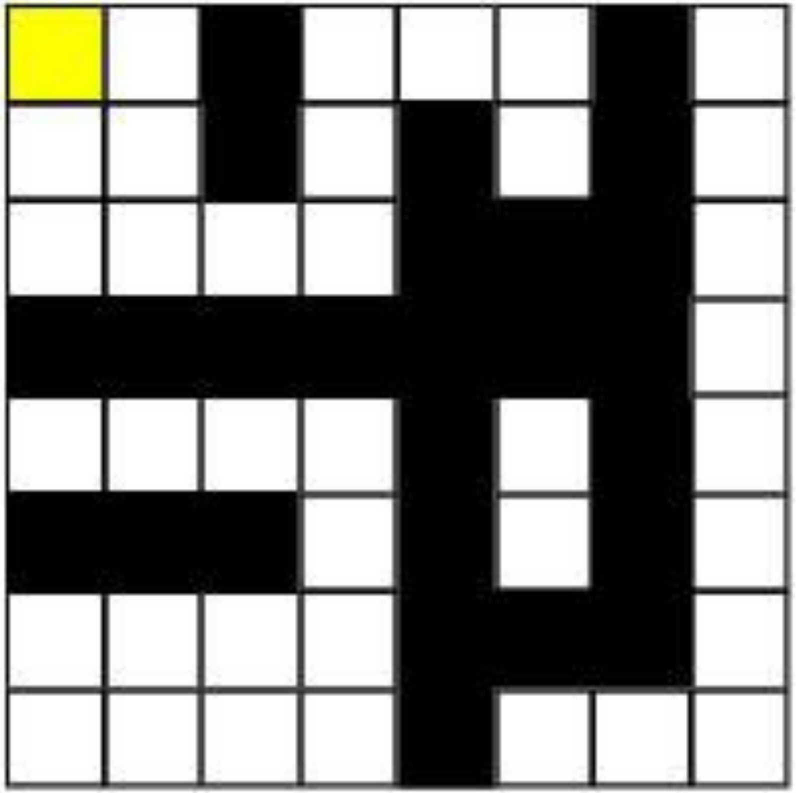
Two-Pass Algorithm for Connected Component Labelling



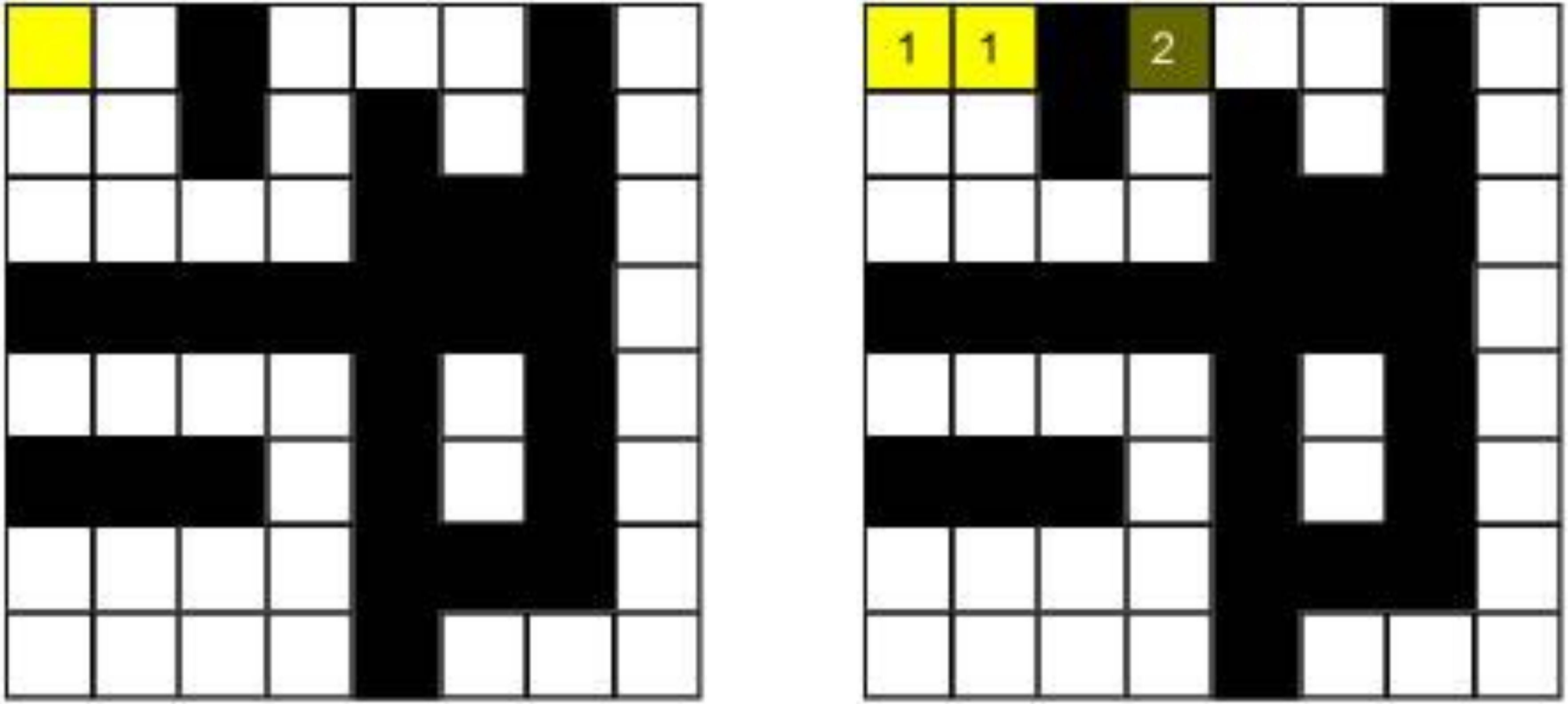
1	1	0	1	1	1	0	1
1	1	0	1	0	1	0	1
1	1	1	1	0	0	0	1
0	0	0	0	0	0	0	1
1	1	1	1	0	1	0	1
0	0	0	1	0	1	0	1
1	1	1	1	0	0	0	1
1	1	1	1	0	1	1	1

2PA: No top, left pixels → Create new label

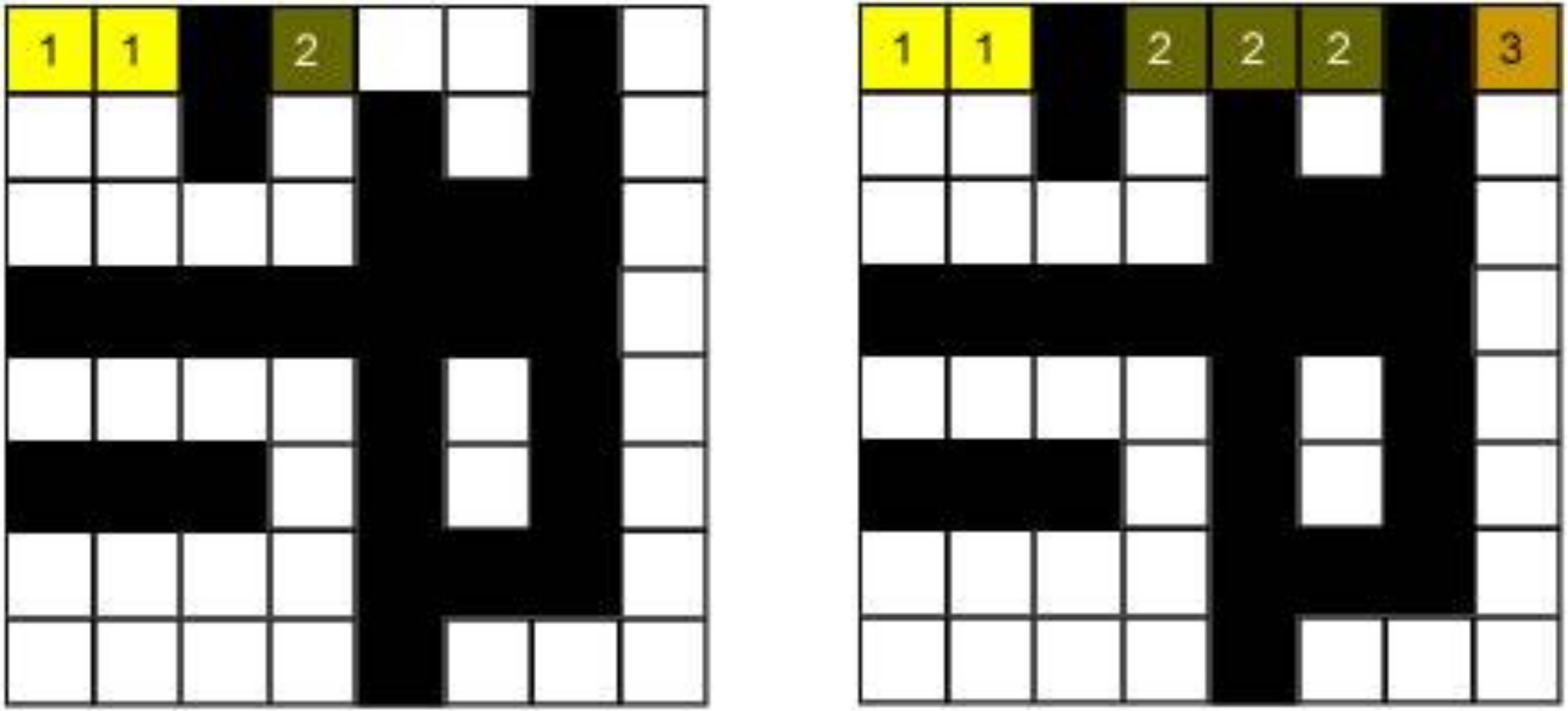




2PA: left pixel labeled→ Copy label ; left pixel BG → new label

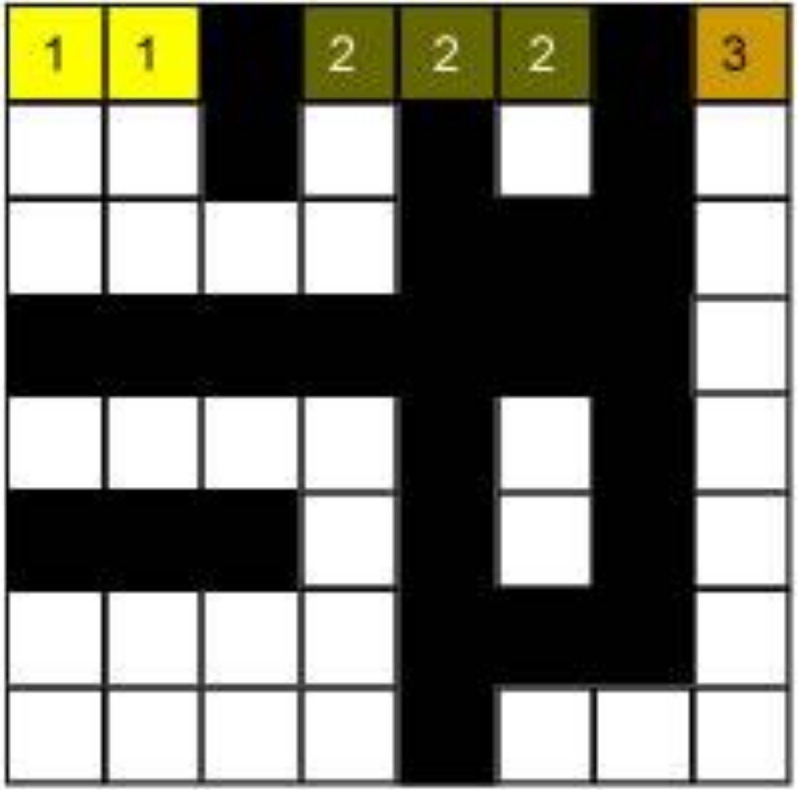


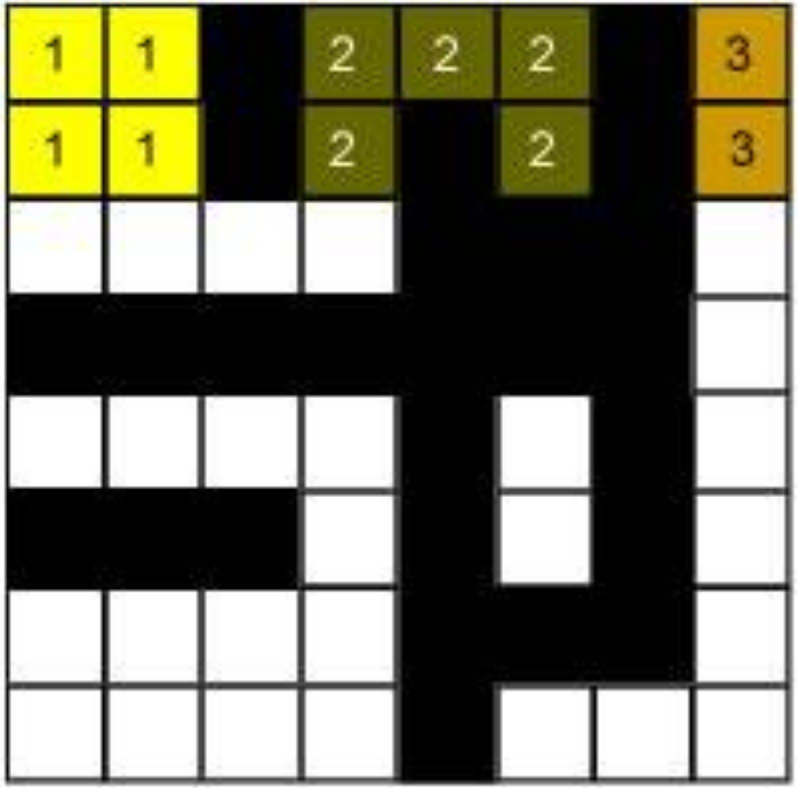
2PA: After row 1 is done



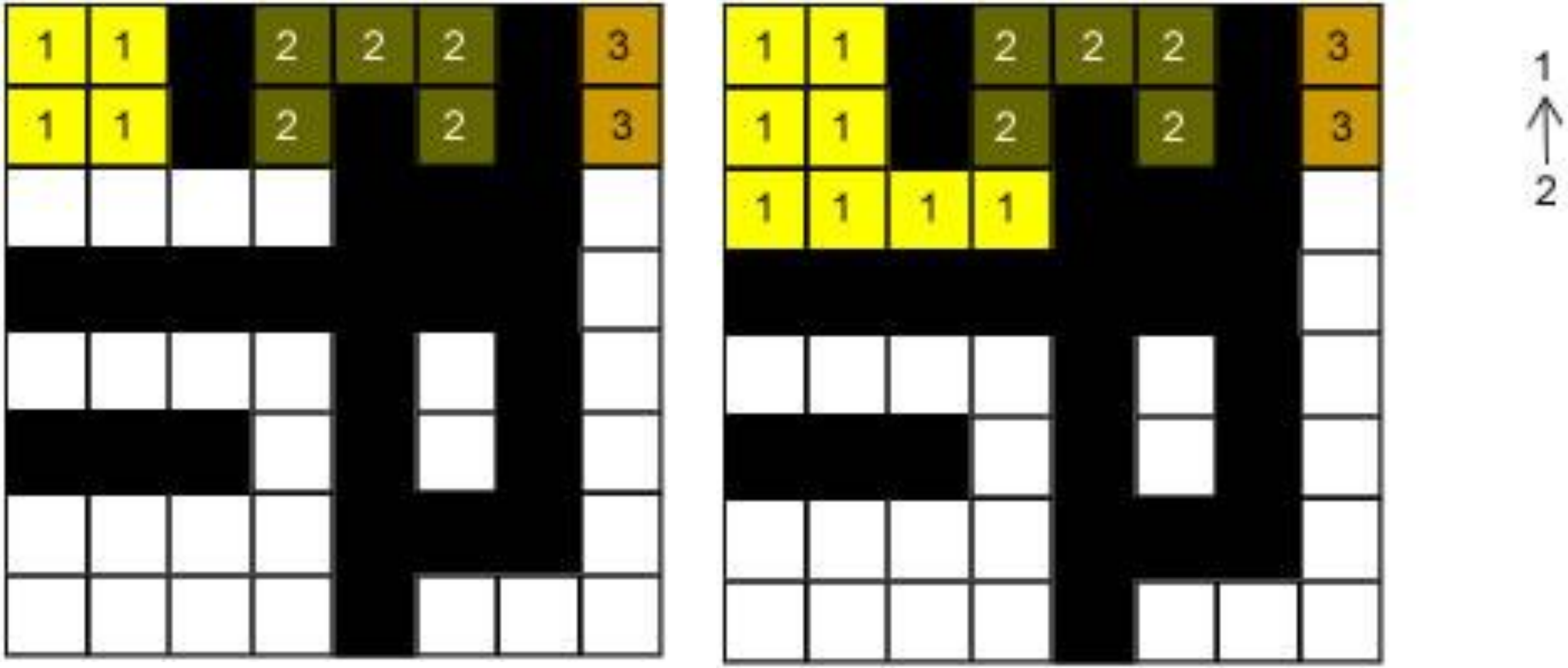
2PA: left/top pixel labeled→ Copy label

(overrides) left pixel BG → new label

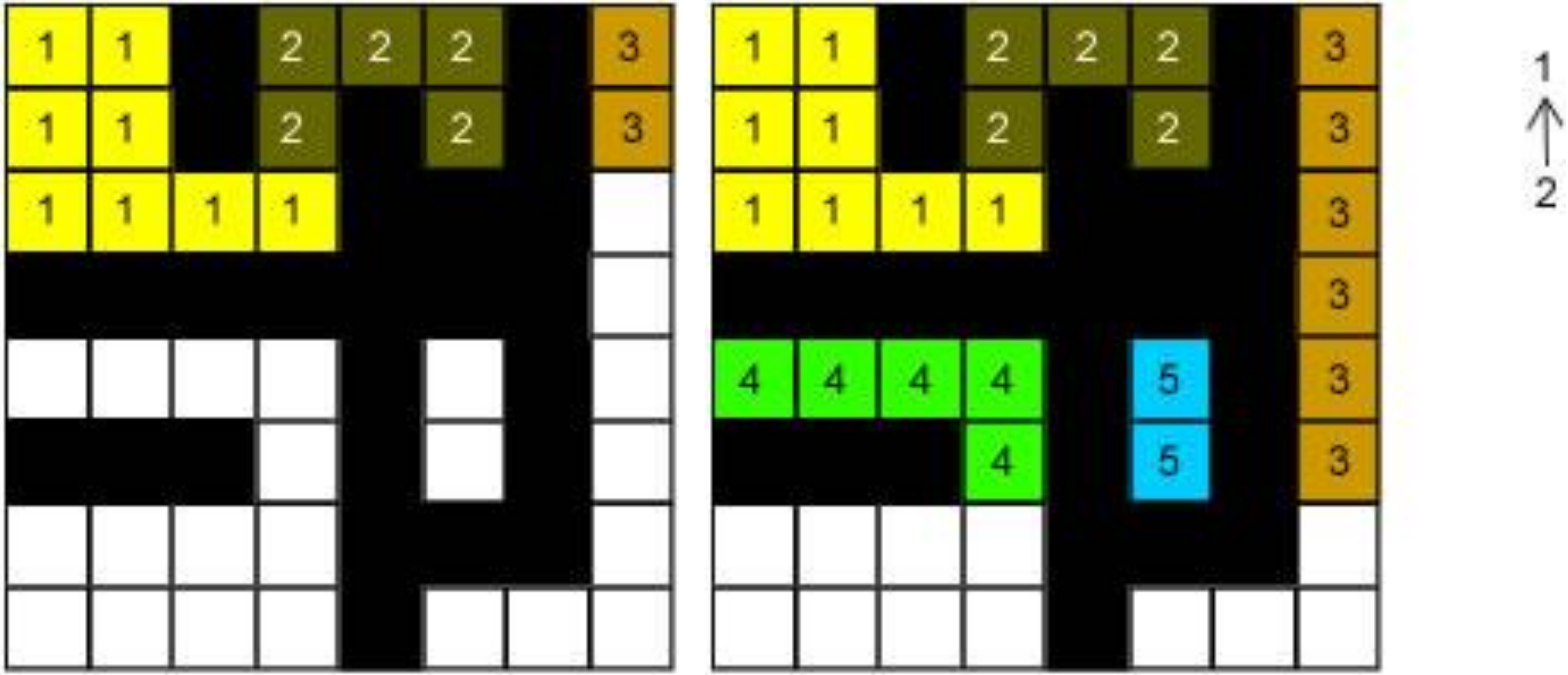




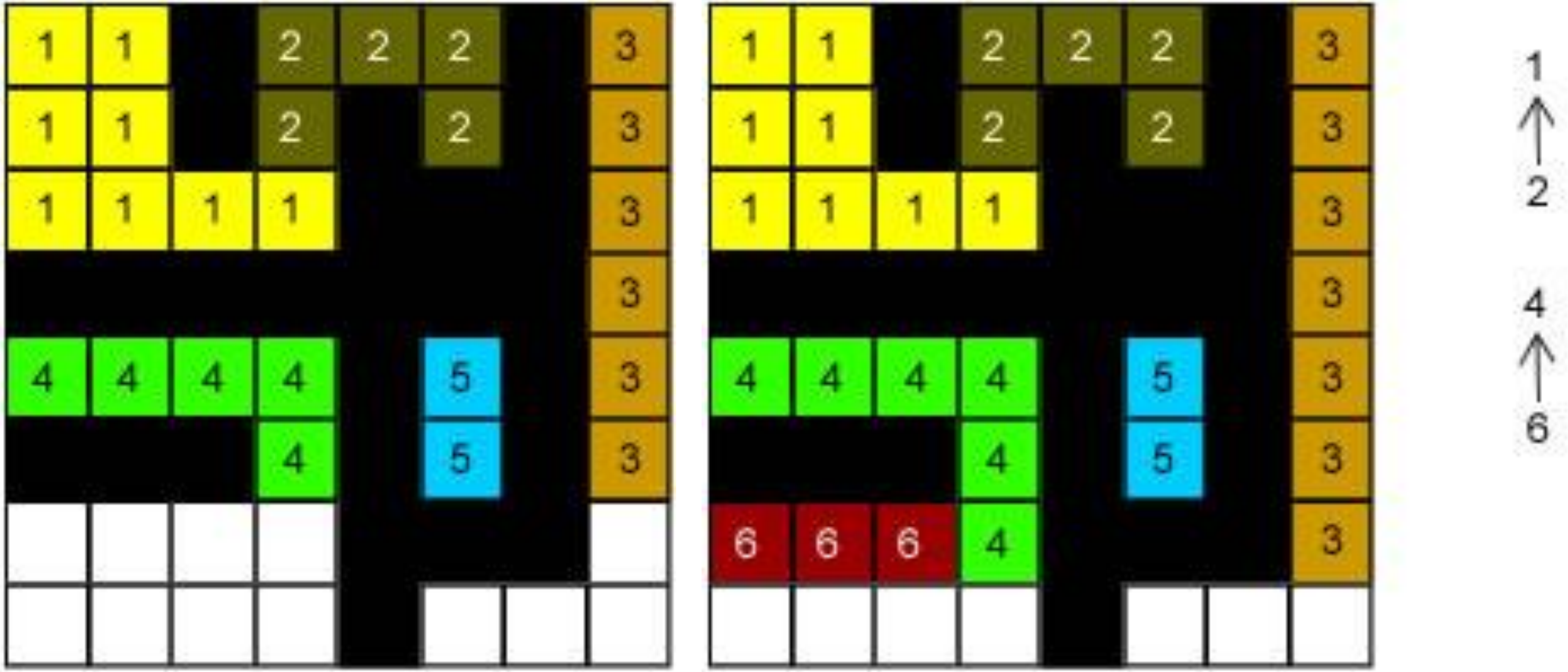
2PA: left/top pixel labeled, different labels →Copy smaller id label, record the association



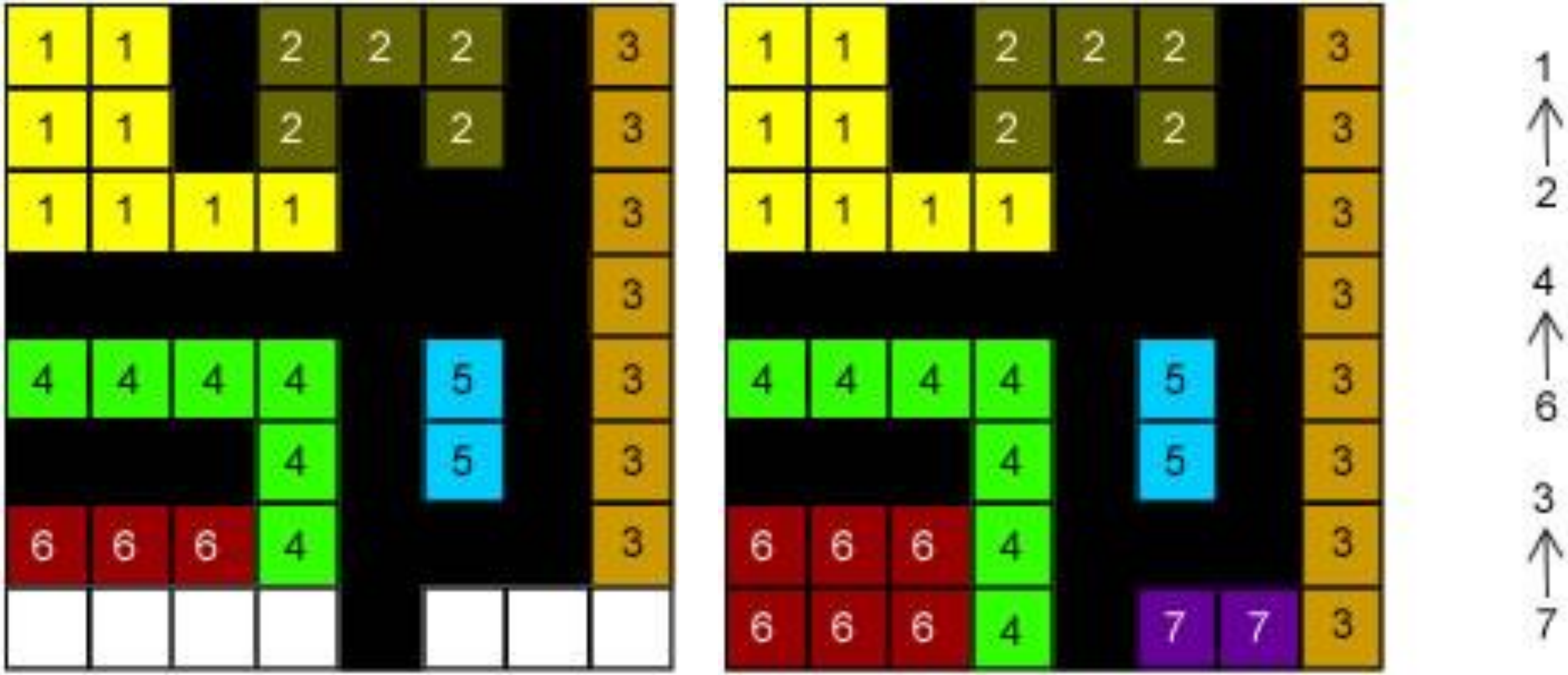
2PA: left/top pixel labeled, different labels →Copy smaller id label, record the association



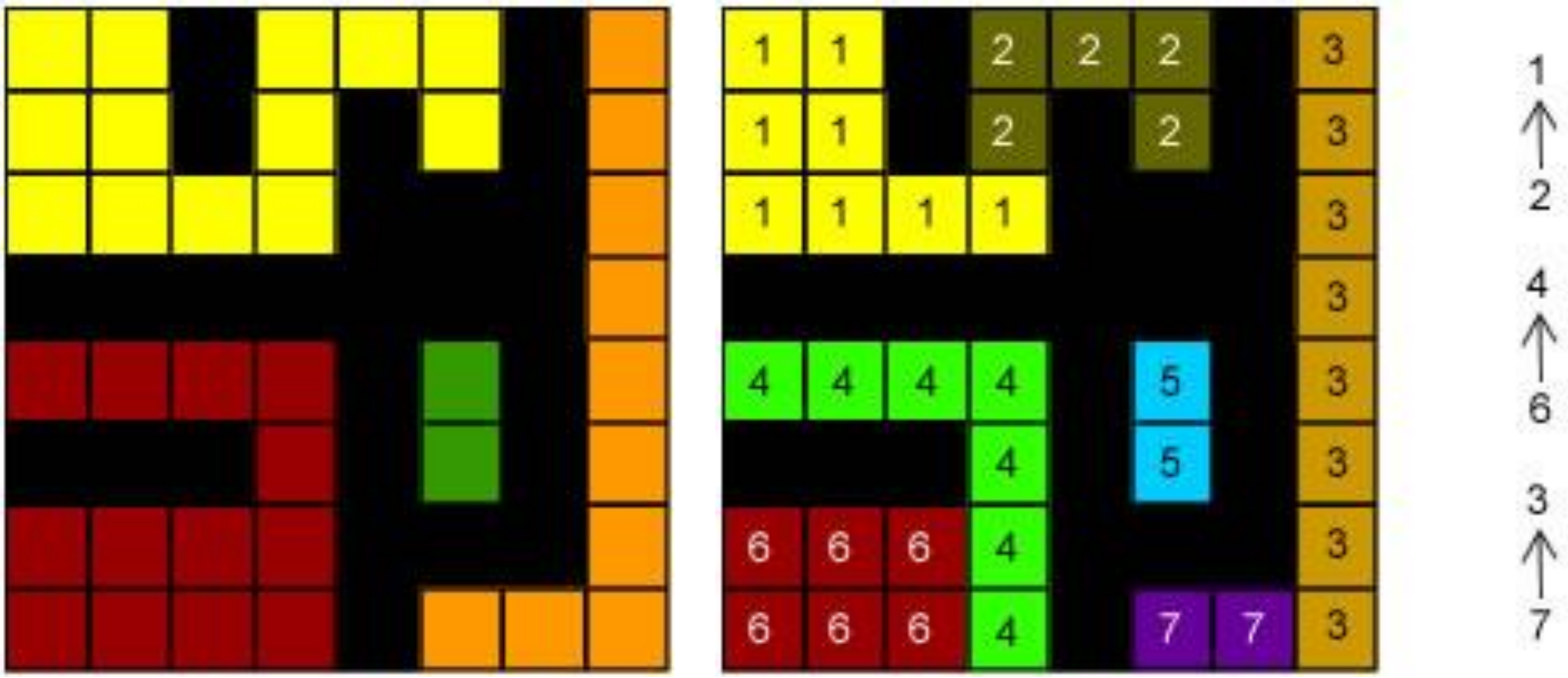
2PA



2PA: After first pass is complete

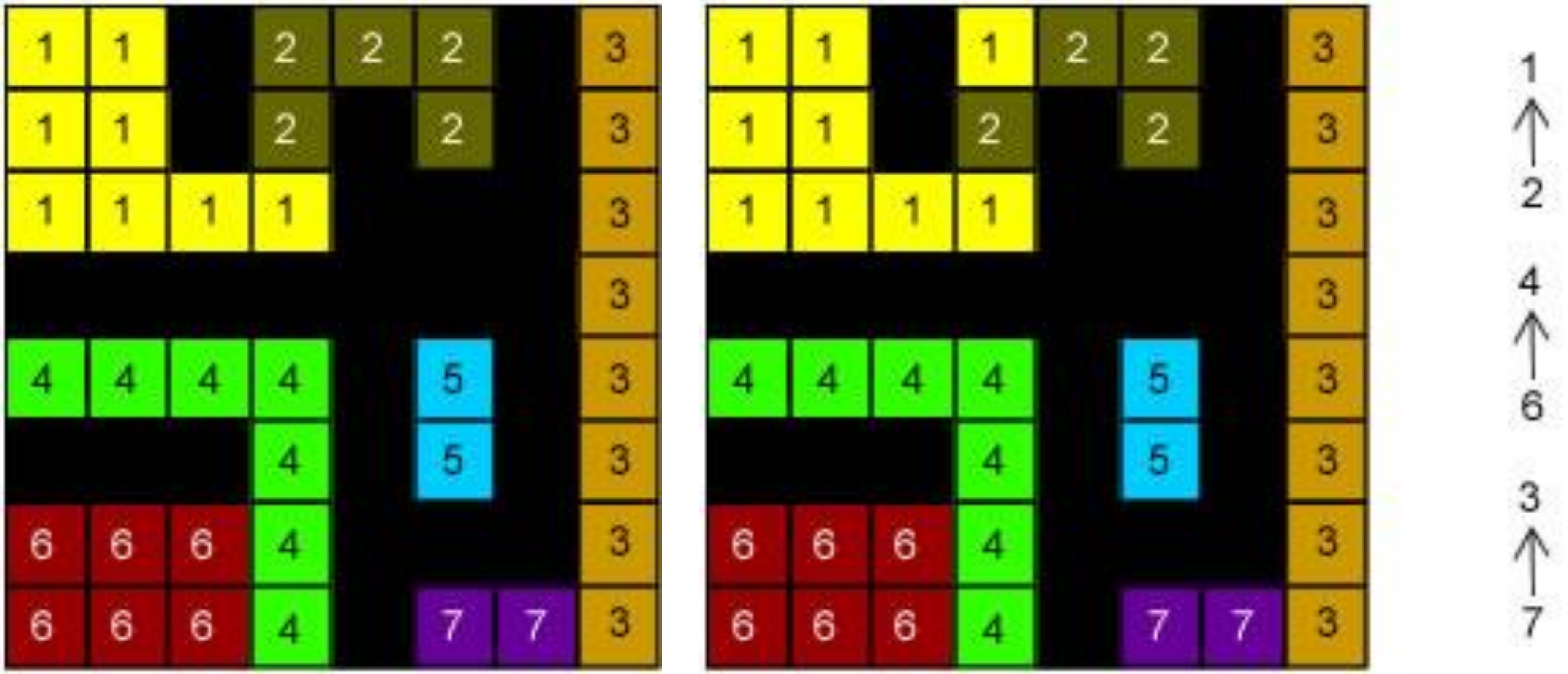


2PA: After first pass is complete



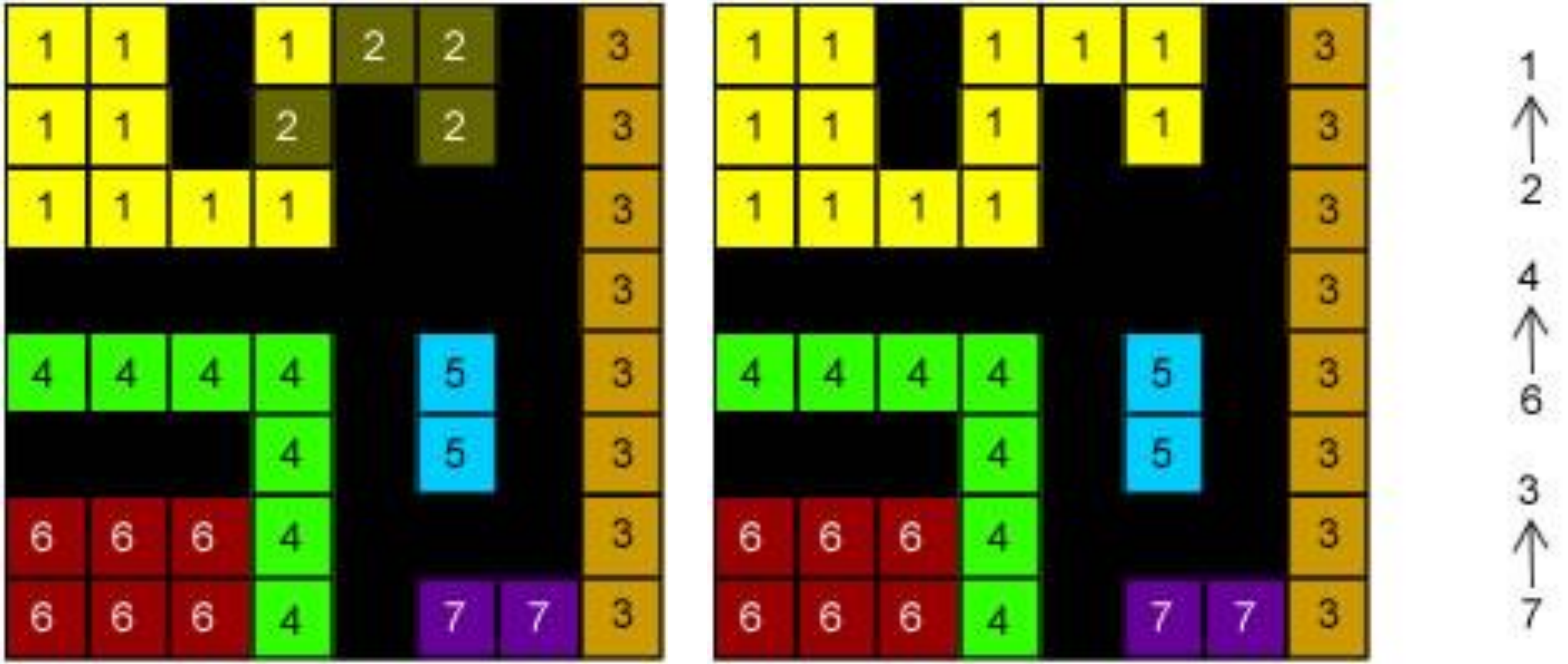
2PA: Second pass: Replace child label with root label.

Union-Find data structure ensures ‘find’-ing root is O(1).



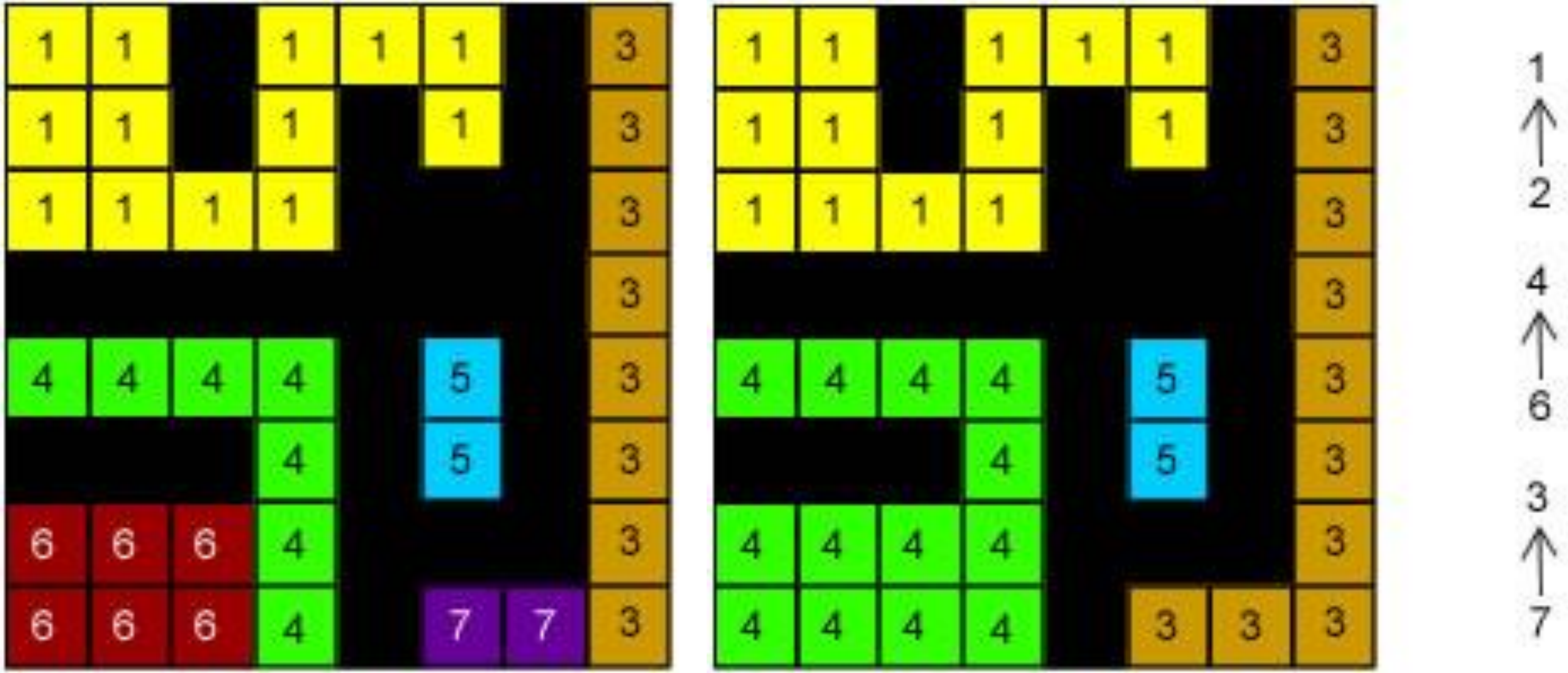
2PA: Second pass: Replace child label with root label.

Union-Find data structure ensures 'find'-ing root is O(1).

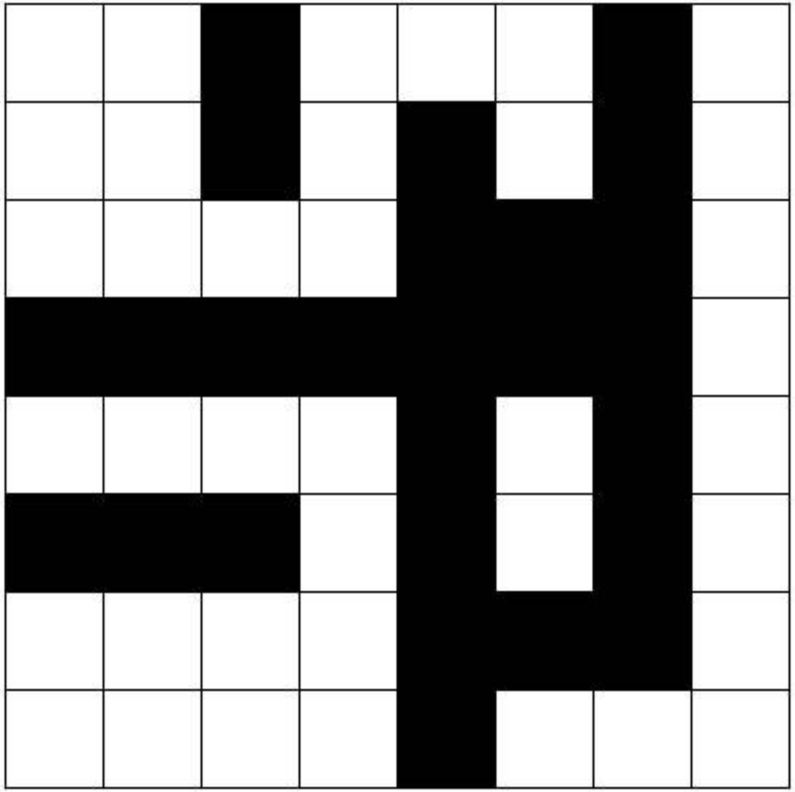


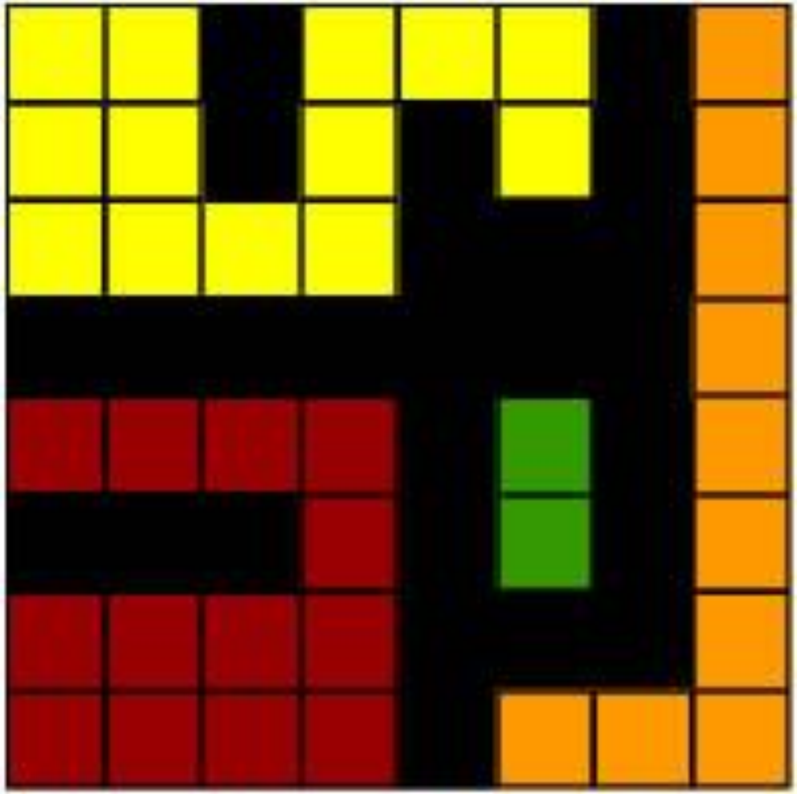
2PA: Second pass: Replace child label with root label.

Union-Find data structure ensures ‘find’-ing root is O(1).



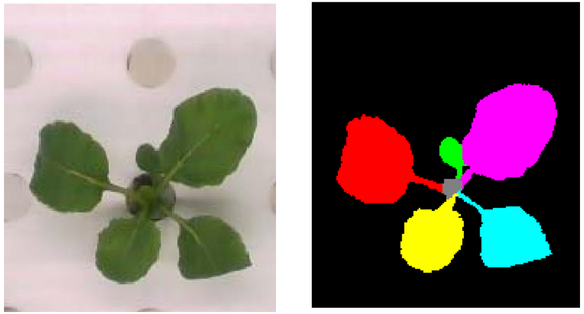
2PA-CCL (Rosenfeld&PFaltz 1968): Requires only two rows of image at a time

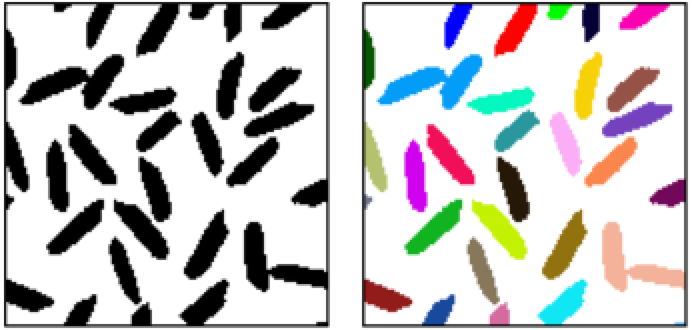


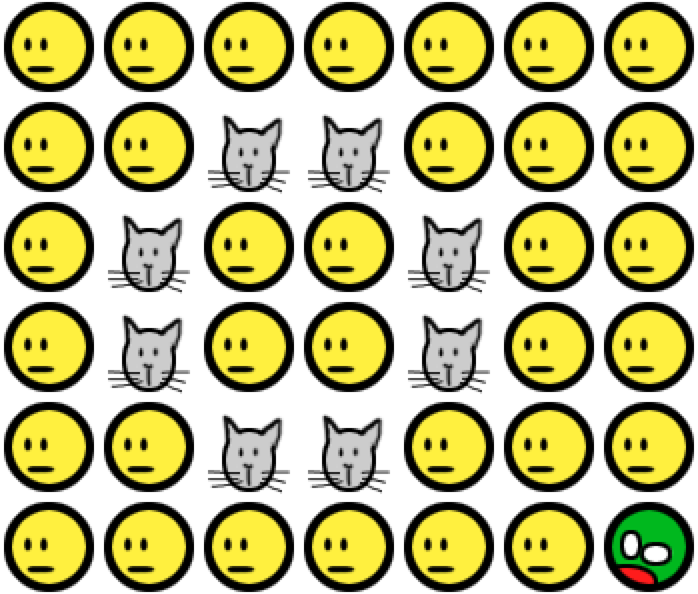


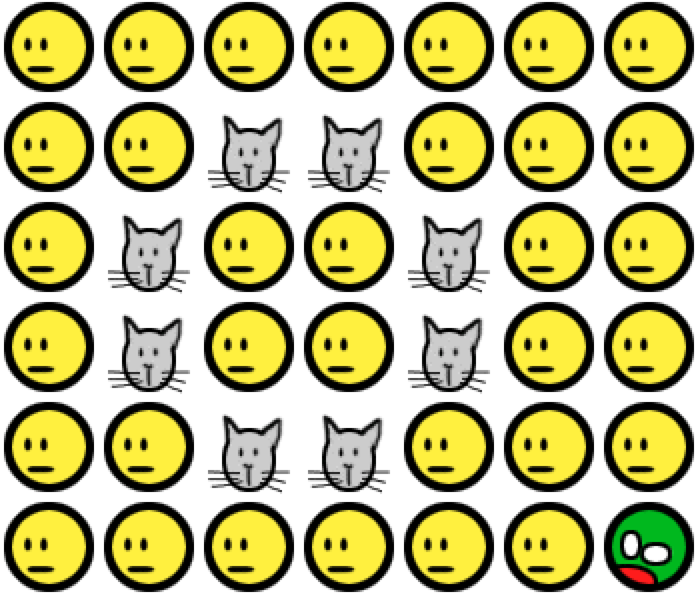
Connected Component Labeling

bwlabel label2rgb



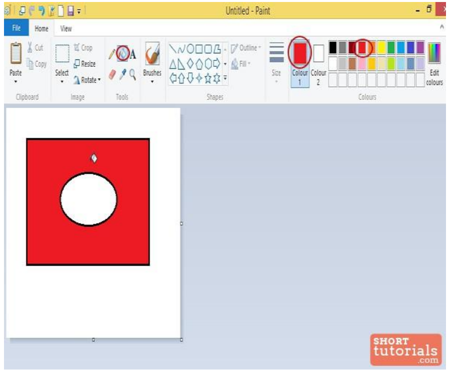


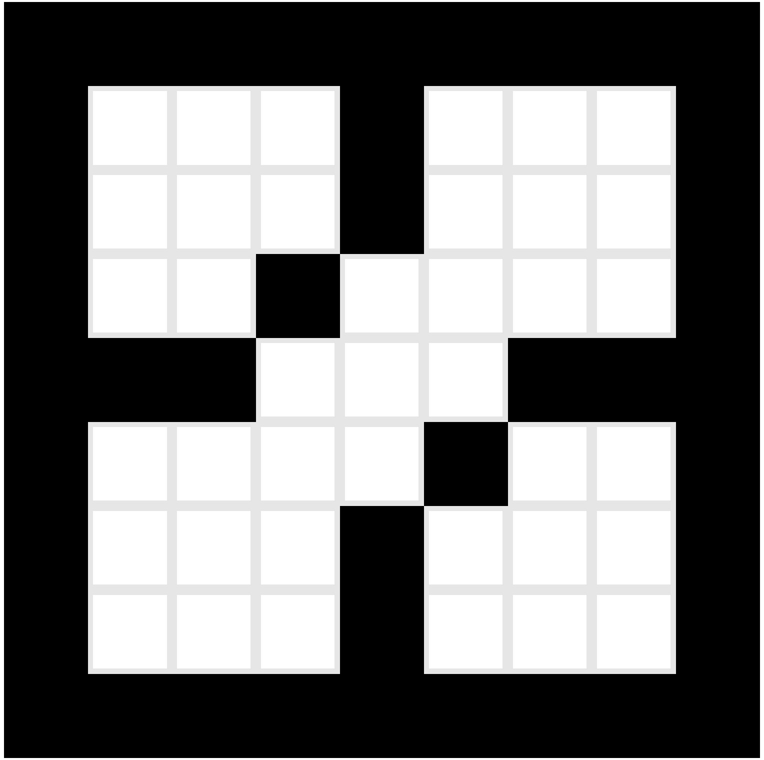




http://inventwithpython.com/blogstatic/floodfill/zombie2.gif

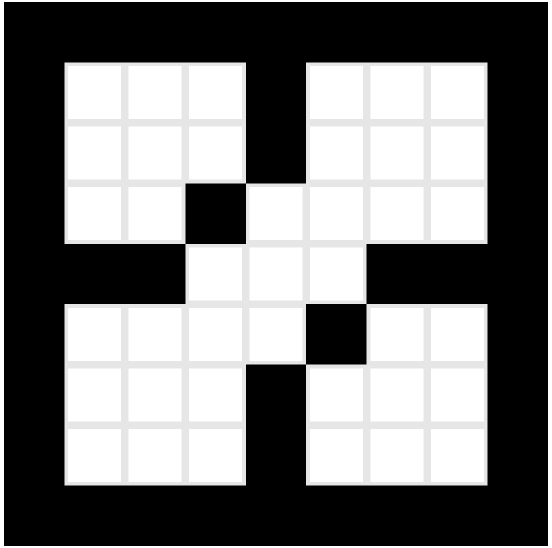
Flood-Fill





https://upload.wikimedia.org/wikipedia/commons/7/7e/Recursive_Flood_Fill_4_%28aka%29.gif

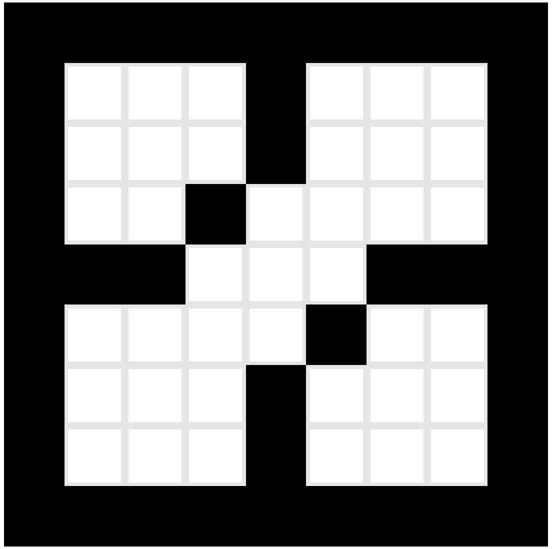
Flood-Fill Algorithm (4-conn)

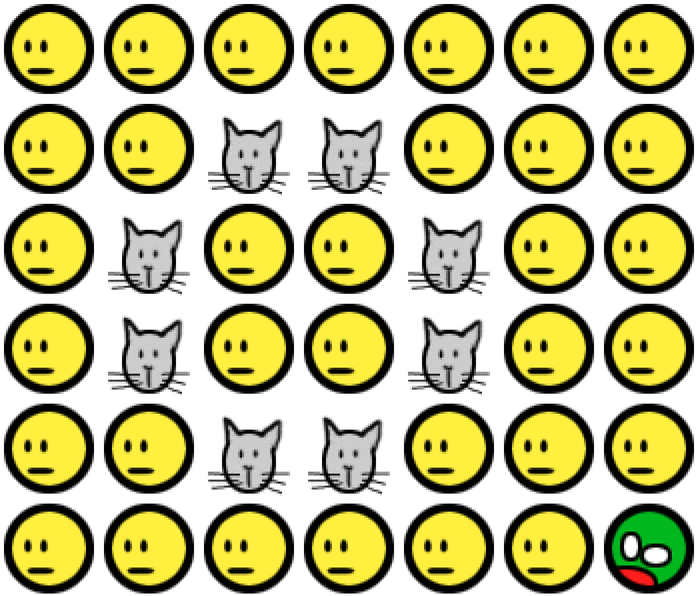


```
void floodFill(int x, int y, int fill, int old)
{
    if ((x < 0) || (x >= width)) return;
    if ((y < 0) || (y >= height)) return;
    if (getPixel(x, y) == old) {
            setPixel(fill, x, y);
            floodFill(x+1, y, fill, old);
            floodFill(x, y+1, fill, old);
            floodFill(x-1, y, fill, old);
            floodFill(x, y-1, fill, old);
    }
}
```

Many/More efficient versions exist !

If 8-conn, all white pixels filled
(all humans eventually become zombie !)





Summary of Morphological Filtering

Operation	Equation	Comments (The Roman numerals refer to the structuring elements shown in Fig. 9.26).
Translation	$(A)_z = \{w \mid w = a + z, \quad \text{for } a \in A\}$	Translates the origin of A to point z.
Reflection	$\hat{B} = \{w \mid w = -b, \quad \text{for } b \in B\}$	Reflects all elements of B about the origin of this set.
Complement	$A^c = \{w \mid w \notin A\}$	Set of points not in A.
Difference	$A - B = \{w \mid w \in A, w \notin B\} = A \cap B^c$	Set of points that belong to A but not to B.
Dilation	$A \oplus B = \{z \mid (\hat{B})_z \cap A \neq \varnothing\}$	"Expands" the boundary of A. (I)
Erosion	$A \ominus B = \{z \mid (B)_z \subseteq A\}$	"Contracts" the boundary of A. (I)
Opening	$A \circ B = (A \ominus B) \oplus B$	Smoothes contours, breaks narrow isthmuses, and eliminates small islands and sharp peaks. (I)
Closing	$A \bullet B = (A \oplus B) \ominus B$	Smoothes contours, fuses narrow breaks and long thin gulfs, and eliminates small holes. (I)

MATLAB codes

circshift(A,z)

fliplr(flipud(B))

~A or 1-A

A &~B

imdilate(A,B)

imerode(A,B)

imopen(A,B)

imclose(A,B)

Slide courtesy: EE465: Introduction to Digital Image Processing

Summary (Con’d)

Hit-or-miss transform	$A \circledast B = (A \ominus B_1) \cap (A^c \ominus B_2)$ $= (A \ominus B_1) - (A \oplus \hat{B}_2)$	The set of points (coordinates) at which, simultaneously, B_1 found a match (“hit”) in A and B_2 found a match in A^c.	bwhitmiss(A,B)
Boundary extraction	$\beta(A) = A - (A \ominus B)$	Set of points on the boundary of set A. (I)	A&~(imerode(A,B))
Region filling	$X_k = (X_{k-1} \oplus B) \cap A^c$; $X_0 = p$ and $k = 1, 2, 3, \ldots$	Fills a region in A, given a point p in the region. (II)	region_fill.m
Thinning	$A \otimes B = A - (A \circledast B)$ $= A \cap (A \circledast B)^c$ $A \otimes \{B\} =$ $((\ldots((A \otimes B^1) \otimes B^2) \ldots) \otimes B^n)$ $\{B\} = \{B^1, B^2, B^3, \ldots, B^n\}$	Thins set A. The first two equations give the basic definition of thinning. The last two equations denote thinning by a sequence of structuring elements. This method is normally used in practice. (IV)	bwmorph(A,’thin’);

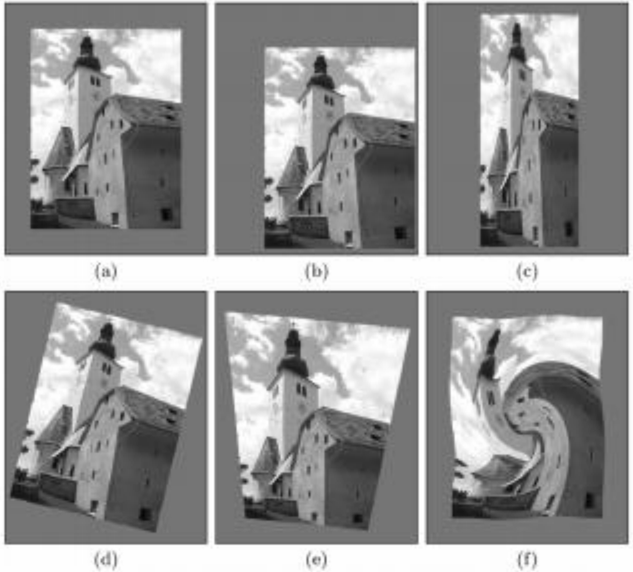
Morphological Filtering using MATLAB

- https://in.mathworks.com/help/images/morphological-filtering.html

GEOMETRIC OPERATIONS

Geometric Operations

- Filters, point operations change intensity
- Pixel position (and geometry) unchanged
- Geometric operations: change image geometry
- **Examples:** translating, rotating, scaling an image



Examples of Geometric operations

Geometric Operations

- Example applications of geometric operations:
  - Zooming images, windows to arbitrary size
  - Computer graphics: deform textures and map to arbitrary surfaces
- **Definition:** Geometric operation transforms image *I* to new image *I'* by modifying **coordinates of image pixels**

$$I(x, y) \rightarrow I'(x', y')$$

- Intensity value originally at (x,y) moved to new position (x',y')

(x,y)

$(x + d_x, y + d_y)$

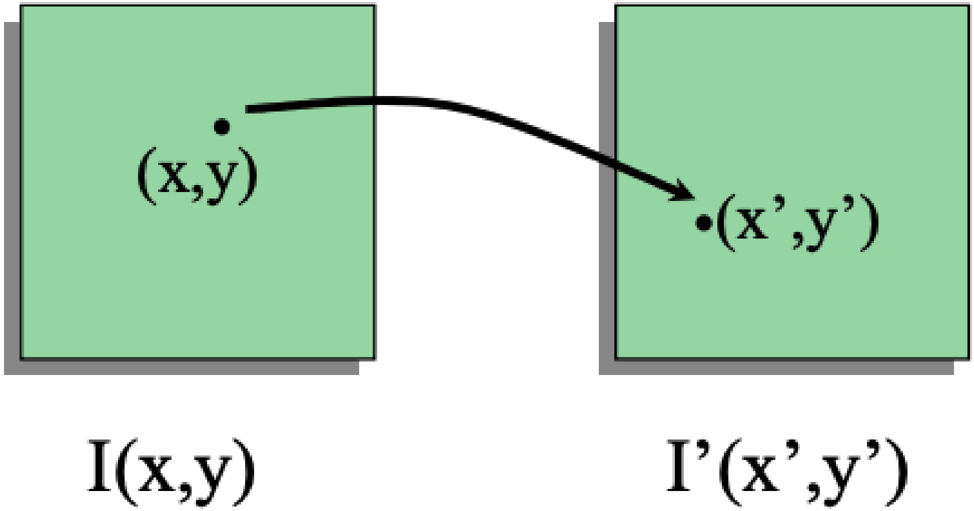
Example: Translation geometric operation moves value at (x,y) to $(x + d_x, y + d_y)$

https://web.cs.wpi.edu/~emmanuel/courses/cs545/S14/slides/lecture11.pdf

$$x \to f_x(x, y) = x'$$

$$y \to f_y(x, y) = y'$$

$$I(x, y) = I'\left(f_x(x, y), f_y(x, y)\right)$$

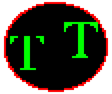


Common Geometric Operations

- **Scale** - change image content size

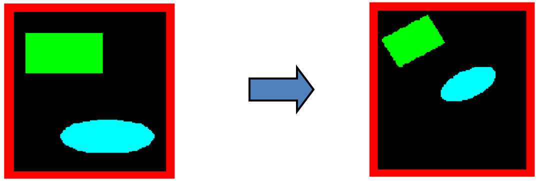
- **Rotate** - change image content orientation

- **Reflect** - flip over image contents



- **Translate** - change image content position

- **Affine Transformation**
  - general image content linear geometric transformation



Simple Mappings

- **Translation:** (shift) by a vector (d_x, d_y)

$$T_x : x' = x + d_x \qquad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} d_x \\ d_y \end{pmatrix}$$
$$T_y : y' = y + d_y$$

or

https://web.cs.wpi.edu/~emmanuel/courses/cs545/S14/slides/lecture11.pdf

Simple Mappings

- **Translation:** (shift) by a vector (d_x, d_y)

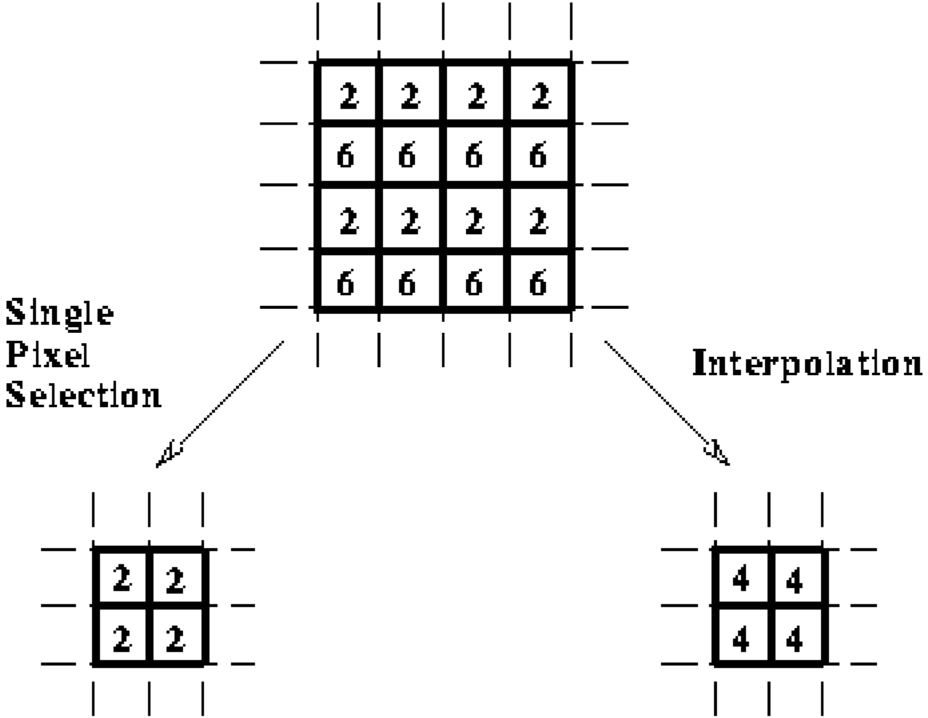
$$\begin{aligned} T_x &: x' = x + d_x \\ T_y &: y' = y + d_y \end{aligned} \quad \text{or} \quad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} d_x \\ d_y \end{pmatrix}$$

- **Scaling:** (contracting or stretching) along x or y axis by a factor s_x or s_y

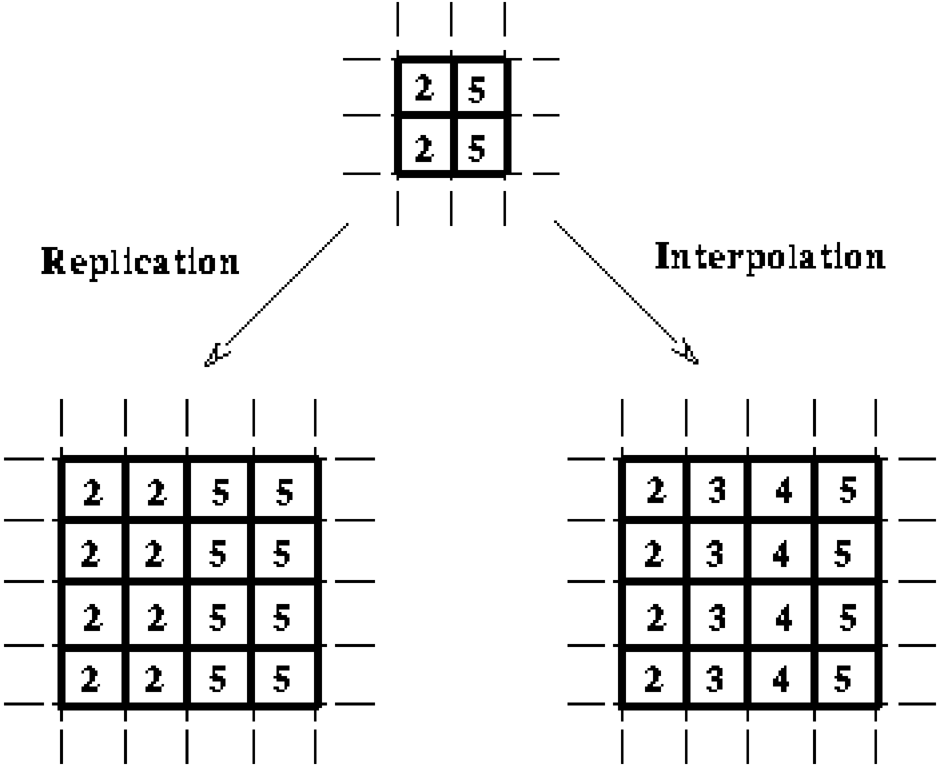
$$\begin{aligned} T_x &: x' = s_x \cdot x \\ T_y &: y' = s_y \cdot y \end{aligned} \quad \text{or} \quad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} s_x & 0 \\ 0 & s_y \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

https://web.cs.wpi.edu/~emmanuel/courses/cs545/S14/slides/lecture11.pdf

Scaling (Shrink)



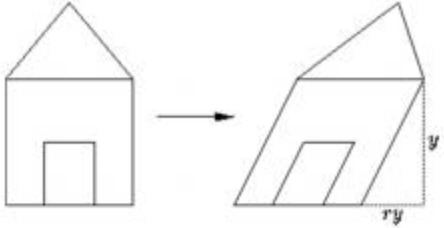
Scaling (Stretch)



Simple Mappings

- **Shearing:** along x and y axis by factor b_x and b_y

$$T_x : x' = x + b_x \cdot y \qquad \text{or} \qquad \begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} 1 & b_x \\ b_y & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$
$$T_y : y' = y + b_y \cdot x$$



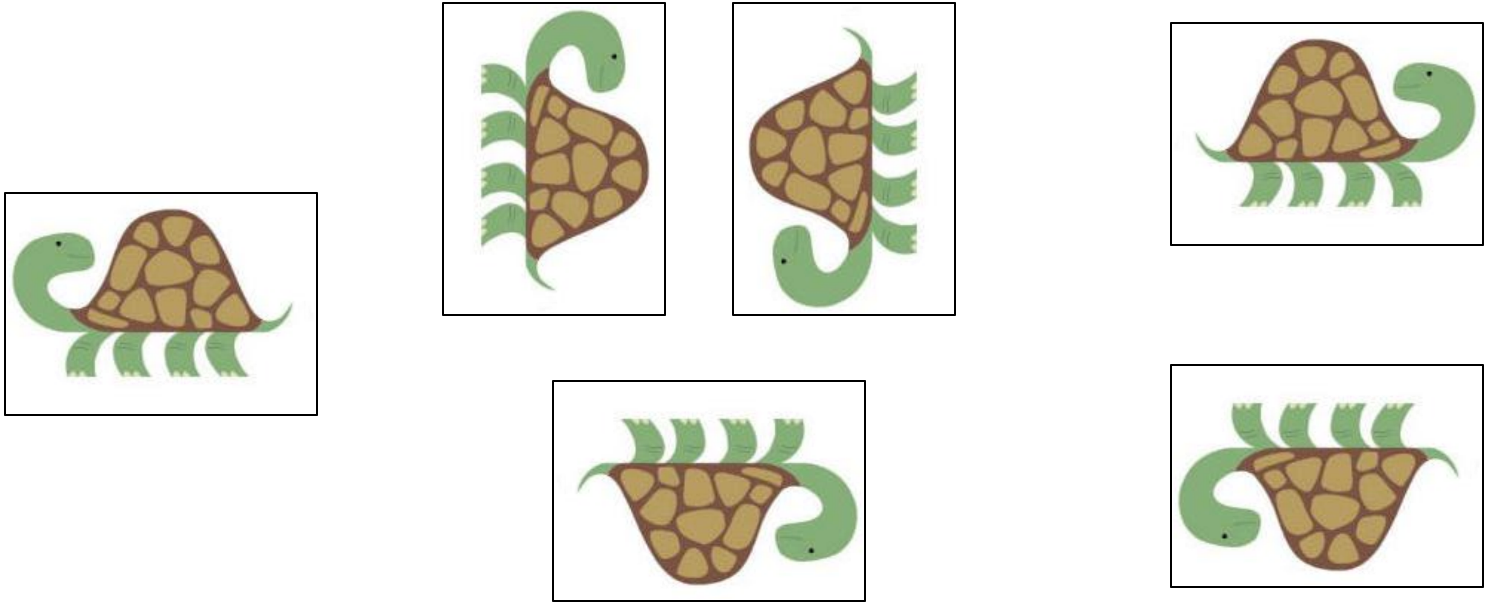
- **Rotation:** the image by an angle α

$$T_x : x' = x \cdot \cos\alpha - y \cdot \sin\alpha$$
$$T_y : y' = x \cdot \sin\alpha + y \cdot \cos\alpha$$

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix}$$

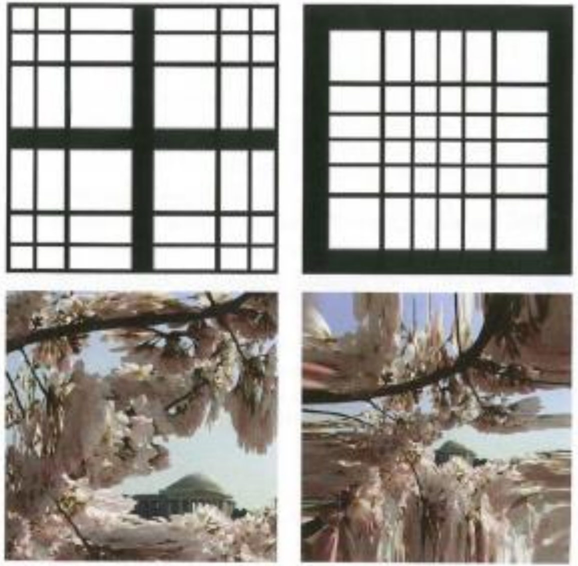
https://web.cs.wpi.edu/~emmanuel/courses/cs545/S14/slides/lecture11.pdf

90, 180 rotations , Flipping



- **Image warping:** we can use a function to select which pixel somewhere else in the image to look up
- For example: apply function on both texel coordinates (x, y)

$$x' = x + y * \sin(\pi * x)$$



https://web.cs.wpi.edu/~emmanuel/courses/cs545/S14/slides/lecture11.pdf

Homogeneous Coordinates

- Notation useful for converting scaling, translation, rotating into point-matrix multiplication
- To convert ordinary coordinates into homogeneous coordinates

$$\boldsymbol{x} = \begin{pmatrix} x \\ y \end{pmatrix} \quad \text{converts to} \quad \hat{\boldsymbol{x}} = \begin{pmatrix} \hat{x} \\ \hat{y} \\ h \end{pmatrix} = \begin{pmatrix} h\,x \\ h\,y \\ h \end{pmatrix}$$

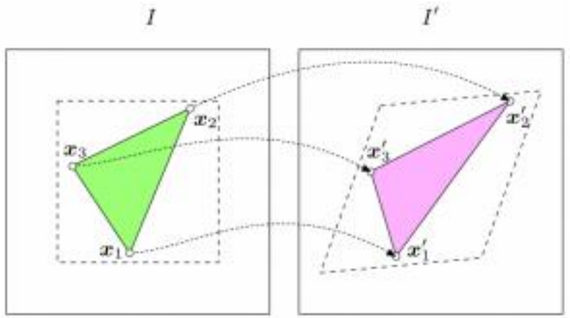
https://web.cs.wpi.edu/~emmanuel/courses/cs545/S14/slides/lecture11.pdf

Affine (3-Point) Mapping

- Can use homogeneous coordinates to rewrite translation, rotation, scaling, etc as vector-matrix multiplication

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

- **Affine mapping:** Can then derive values of matrix that achieve desired transformation (or combination of transformations)

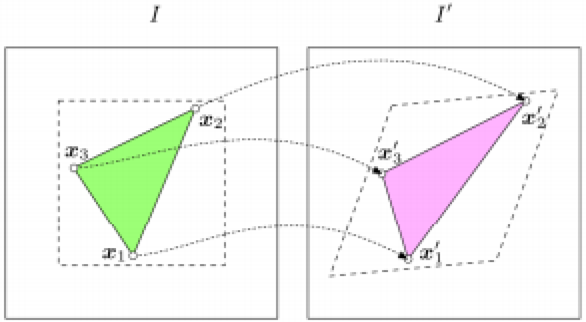


- Inverse of transform matrix is **inverse mapping**

https://web.cs.wpi.edu/~emmanuel/courses/cs545/S14/slides/lecture11.pdf

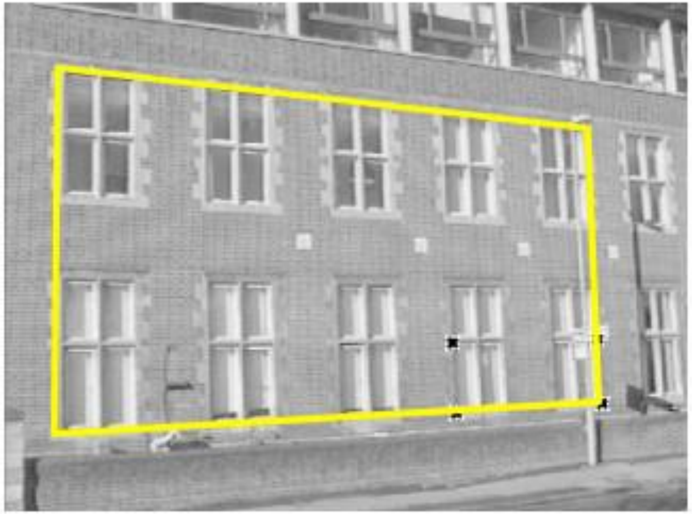
Affine (3-Point) Mapping

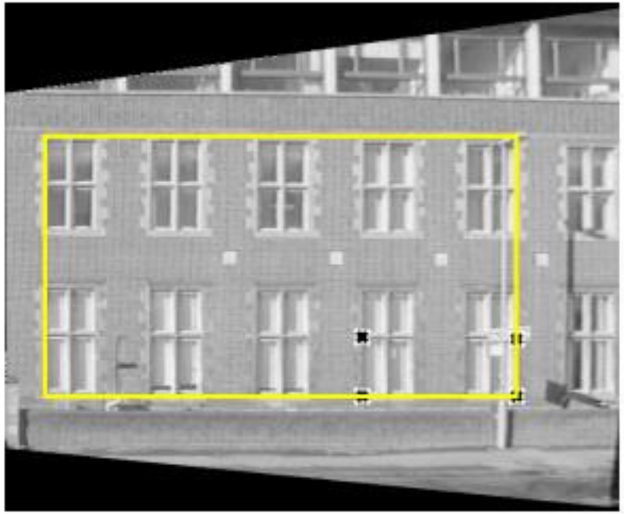
- What's so special about affine mapping?



- Maps
  - straight lines -> straight lines,
  - triangles -> triangles
  - rectangles -> parallelograms
  - Parallel lines -> parallel lines
- Distance ratio on lines do not change

Homography





from Hartley & Zisserman

$$\begin{pmatrix} x_1' \\ x_2' \\ x_3' \end{pmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \times \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

References

- G&W, 3rd Ed., 9.1-9.3, 9.6
- Grayscale Morphology:
  http://people.ciirc.cvut.cz/~hlavac/TeachPresEn/11ImageProc/71-06MatMorfolGrayEn.pdf

Scribe List

2018102018
2018102019
2018102022
2018102027
2018102028
2018102031