CSE 554

Lecture 1: Binary Pictures

Fall 2018

Geometric Forms

- Continuous forms
  - Defined by mathematical functions
  - E.g.: parabolas, splines, subdivision surfaces
- Discrete forms
  - Disjoint elements with connectivity relations
  - E.g.: polylines, triangle surfaces, pixels and voxels

Curves	Surfaces

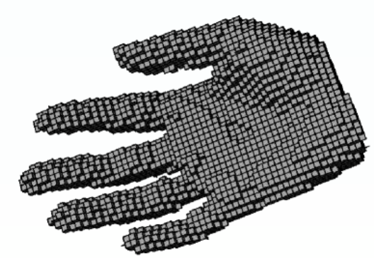
$y = x^2$

$z = Sin[x]Sin[y]$

Polyline

Triangle surfaces

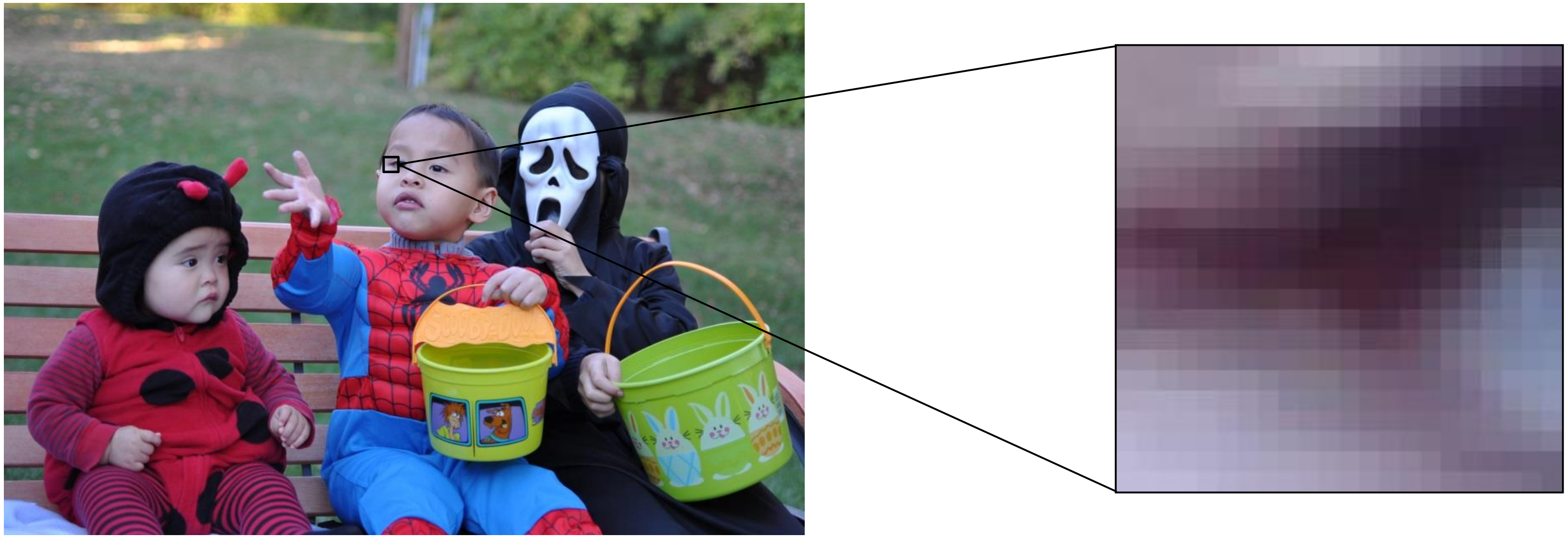
Pixels



Voxels

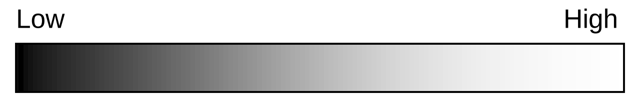
Digital Pictures

- Made up of discrete points associated with colors
  - Image: 2D array of pixels

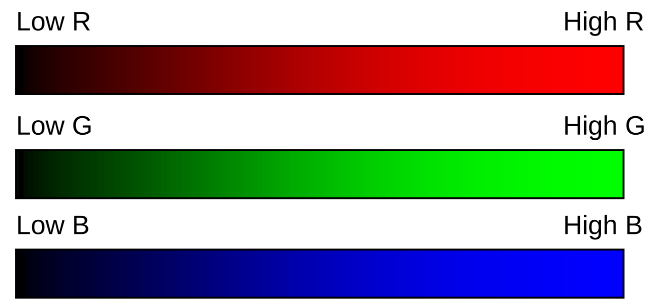


Digital Pictures

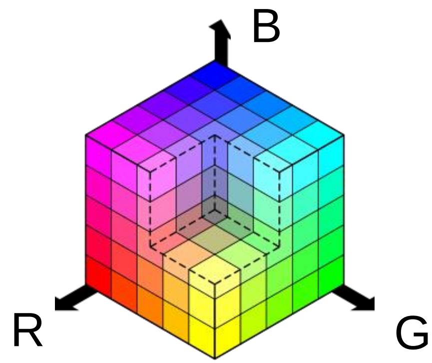
- Color representations
  - Grayscale: 1 value representing grays from black (lowest value) to white (highest value)
    - 8-bit (0-255), 16-bit, etc.
  - RGB: 3 values each representing colors from black (lowest value) to pure red, green, or blue (highest value).
    - 24-bit (0-255 in each color)
  - XYZ, HSL/HSV, CMYK, etc.



Grayscale

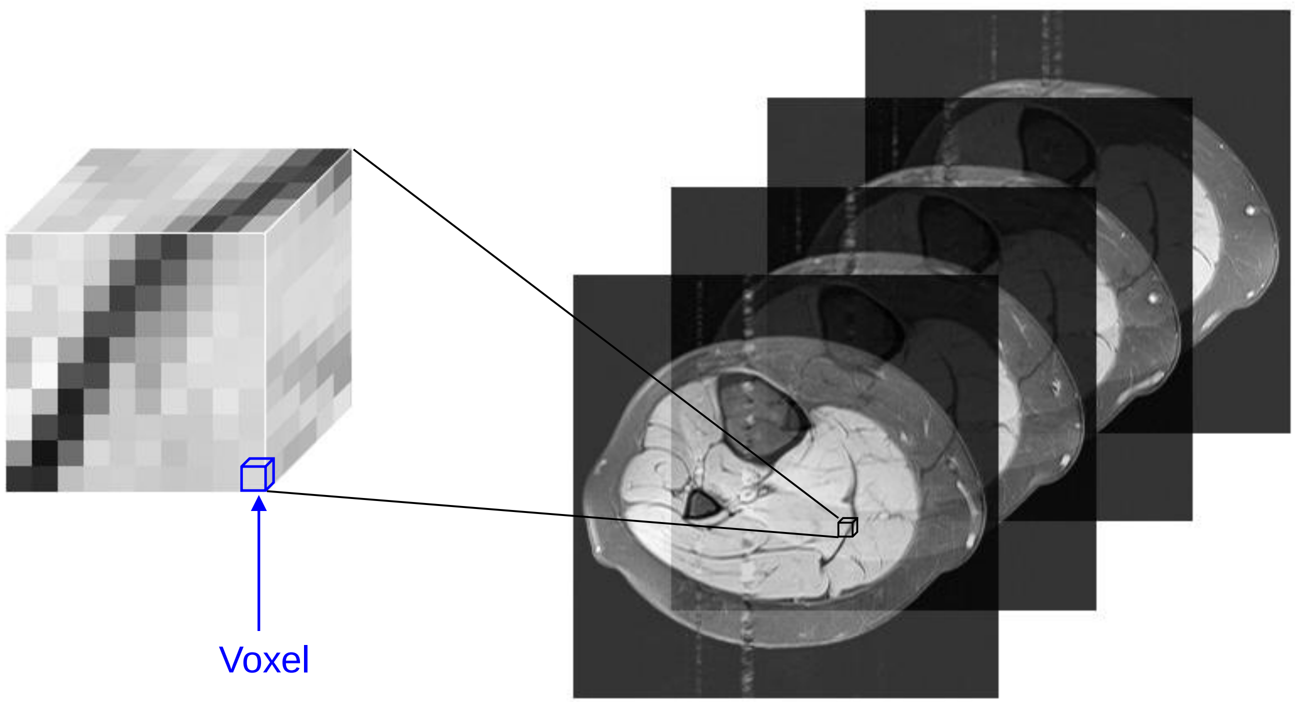


RGB



Digital Pictures

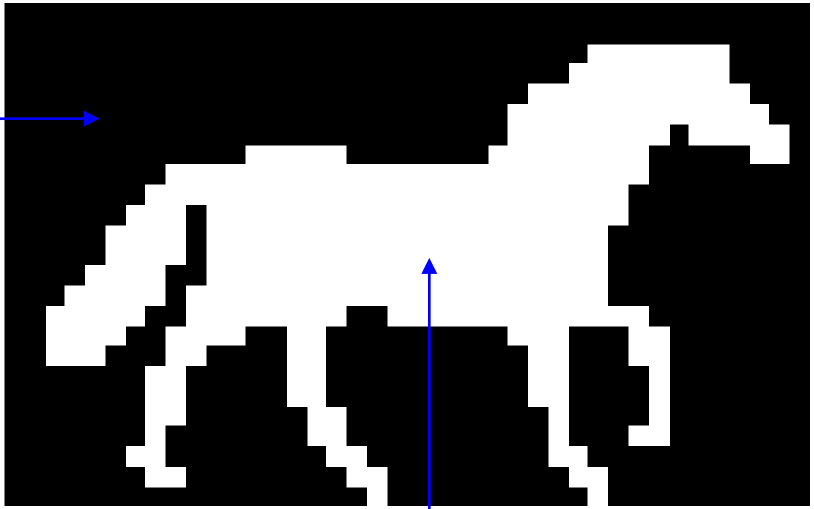
- Made up of discrete points associated with colors
  - Volume: 3D array of voxels



Binary Pictures

- A grayscale picture with 2 colors: black (0) and white (1)
  - The set of 1 or 0 pixels (voxels) is called object or background
  - A "blocky" geometry

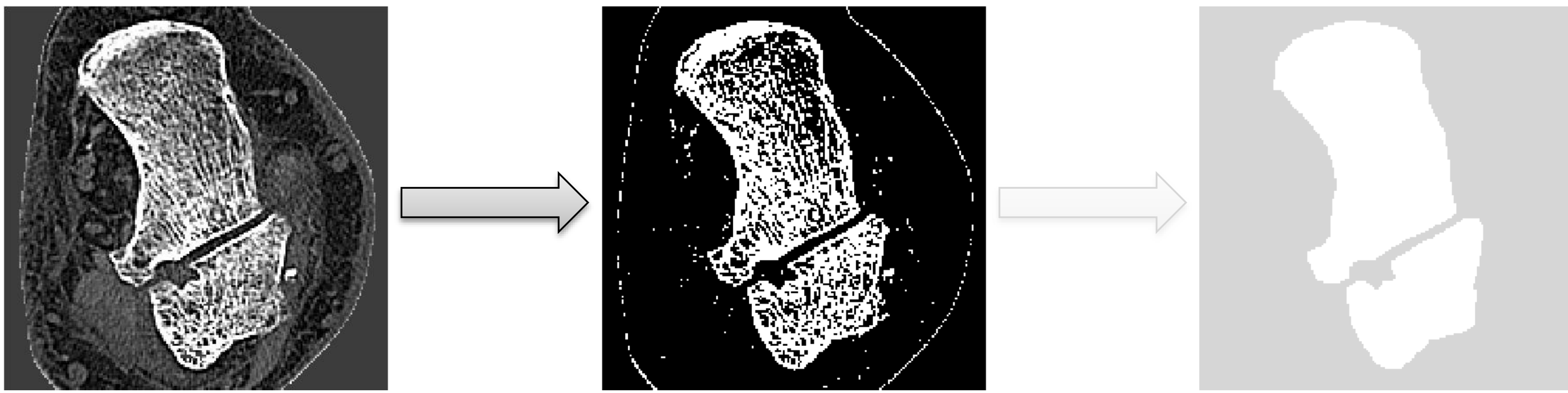
Background



Object

Analogy: Lego, Minecraft

Binary Pictures

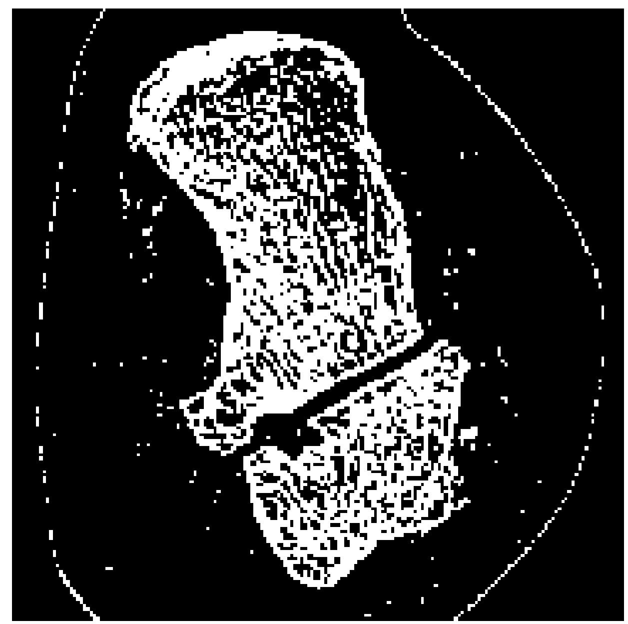


Creation Processing

Segmentation

- Separating object from background in a grayscale picture
  - A simple method: thresholding by pixel (voxel) color
    - All pixels (voxels) with color above a threshold is set to 1

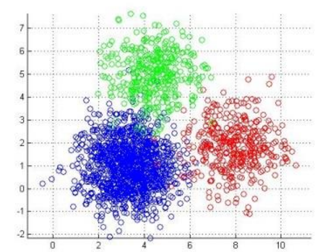
Grayscale picture

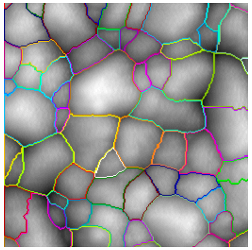


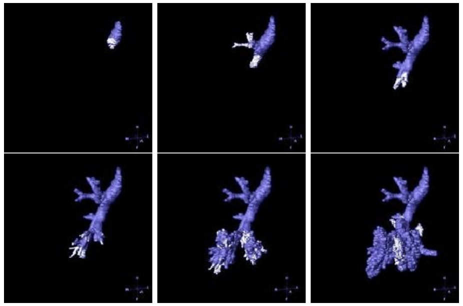
Thresholded binary picture

Segmentation

- Separating object from background in a grayscale picture
  - A simple method: thresholding by pixel (voxel) color
  - Other methods:
    - K-means clustering
    - Watershed
    - Region growing
    - Snakes and Level set
    - Graph cut
    - …



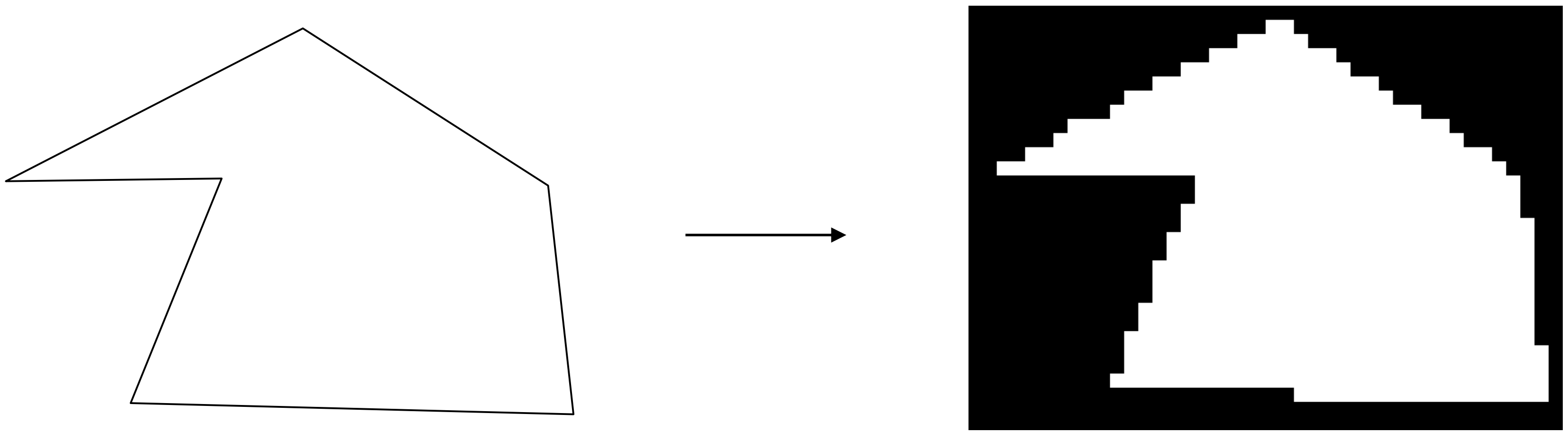




  - More details covered in *Computer Vision* course

Rasterization

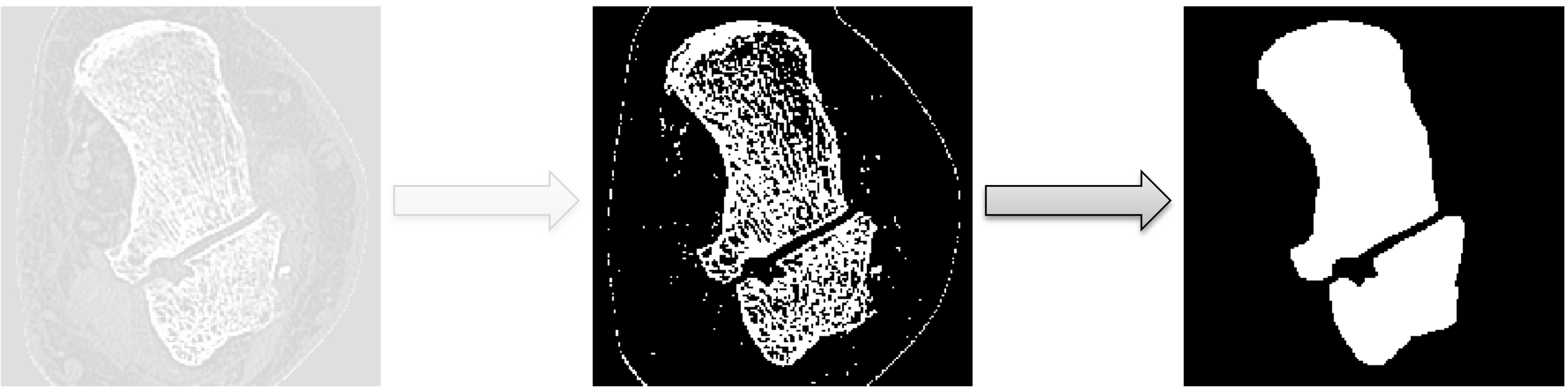
- Filling the interior of a shape by pixels or voxels
  - Known as “scan-conversion”, or “pixelization / voxelization”
  - More details covered in *Computer Graphics* course



2D Polygon

Binary Picture

Binary Pictures



Creation Processing

Binary Pictures

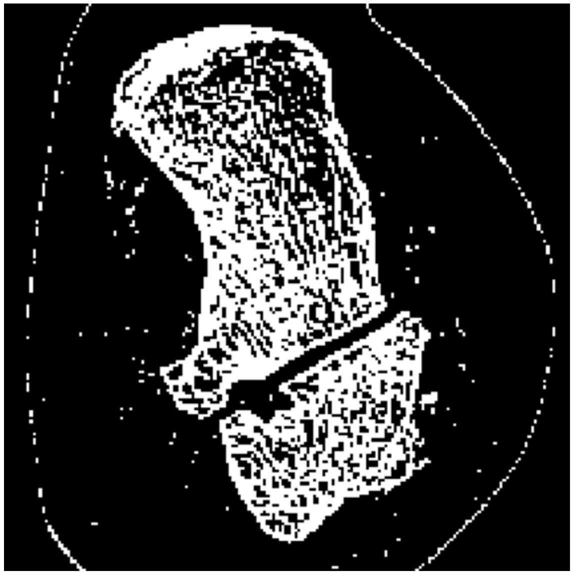
Washington University in St. Louis
School of Engineering & Applied Science

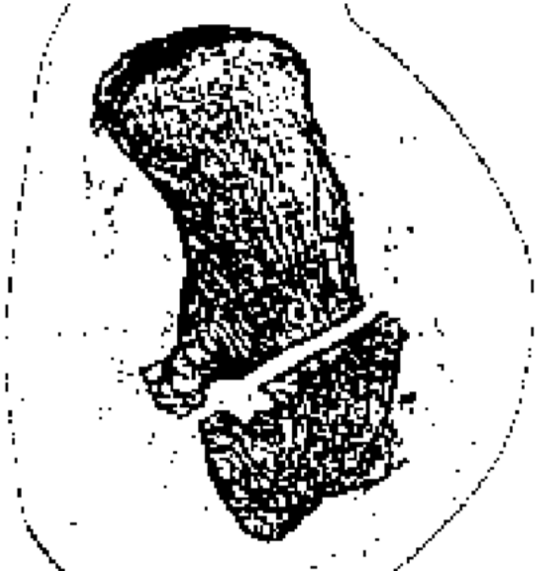
Removing islands and filling holes

Smoothing boundaries

Islands & Holes

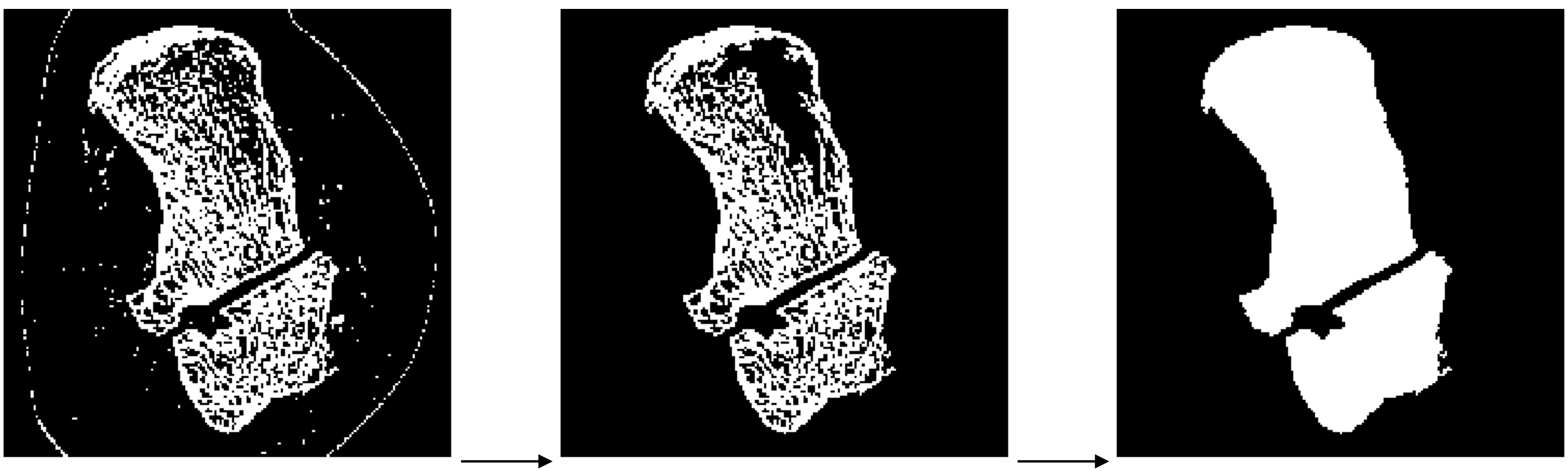
- Observations:
  - Islands (holes) are not as “big” as the object (background).
  - Islands (holes) of the object are holes (islands) of the background





Islands & Holes

- Observations:
  - Islands (holes) are not as "big" as the object (background).
  - Islands (holes) of the object are holes (islands) of the background



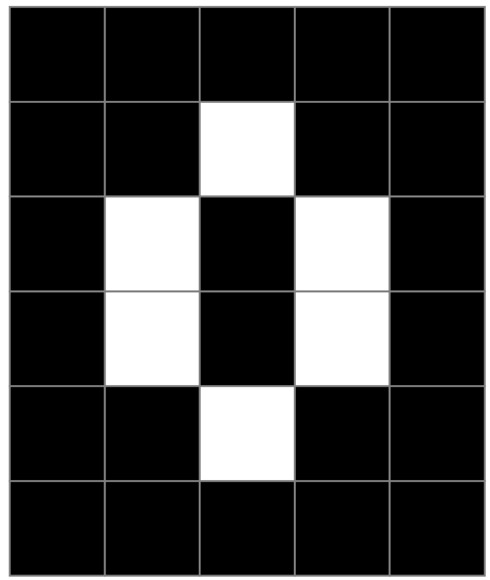
Take the largest 2 connected components of the object

Invert the image, take the largest connected component of the object, invert again

Connected Components

- Definition
  - A maximum set of pixels (voxels) in the object or background, such that any two pixels (voxels) in the set are connected by a path of connected pixels (voxels)

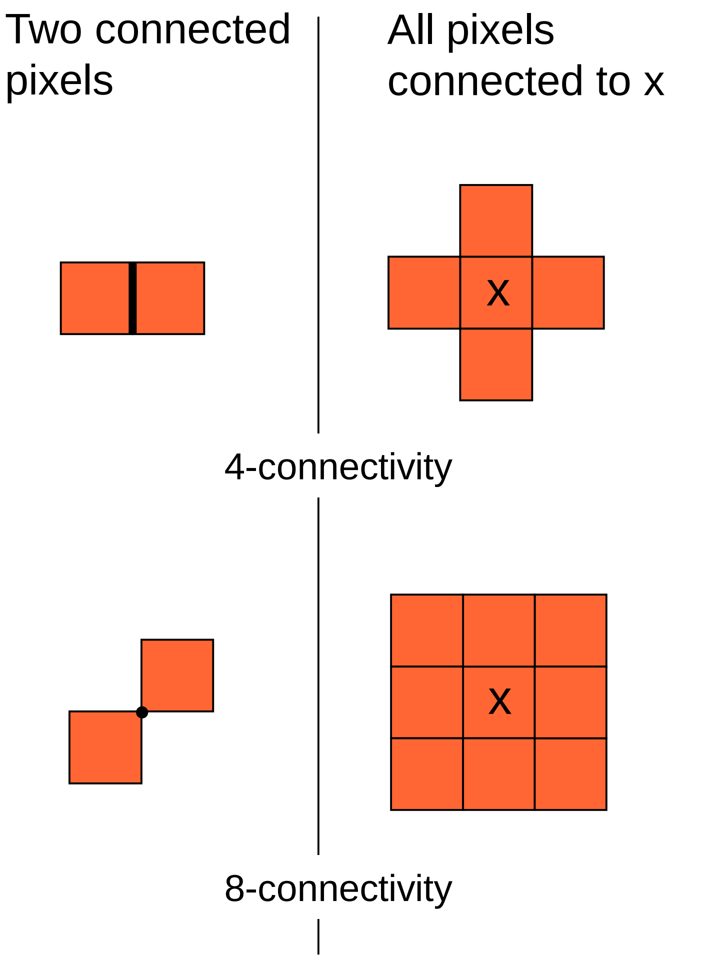
Connected Components



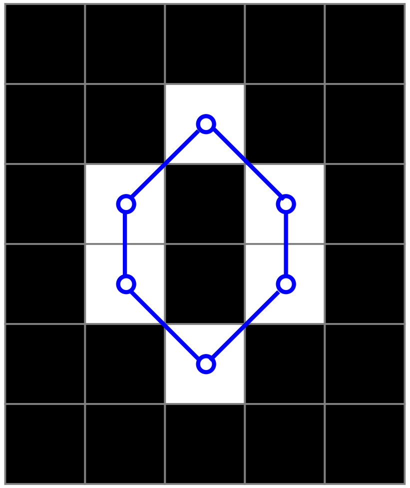
How many connected components are there in the object? What about background?

Connectivity (2D)

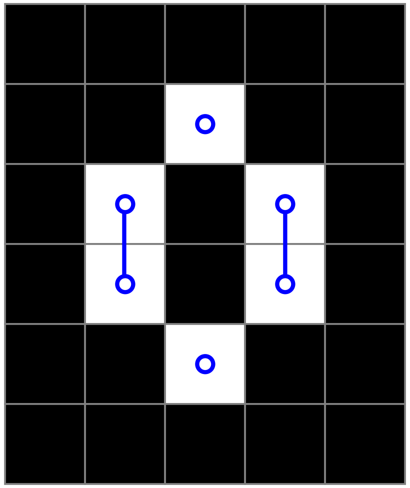
- Two pixels are connected if their squares share:
  - A common edge
    - 4-connectivity
  - A common vertex
    - 8-connectivity



Connectivity (2D)



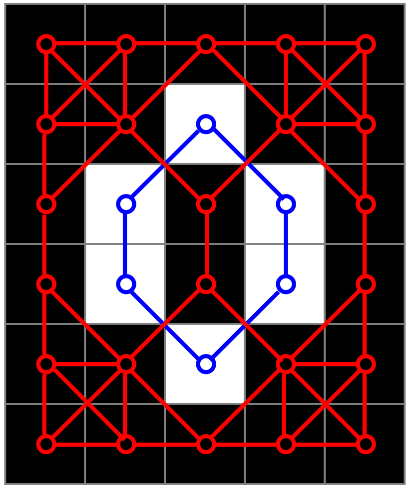
Object: 8-connectivity (1 comp)



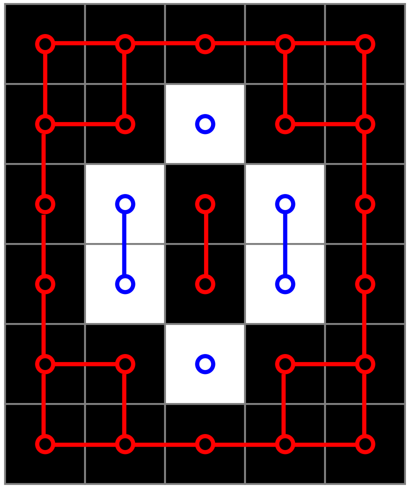
Object: 4-connectivity (4 comp)

Connectivity (2D)

- *What connectivity should be used for the background?*



Object: 8-connectivity (1 comp)
Background: 8-connectivity (1 comp)

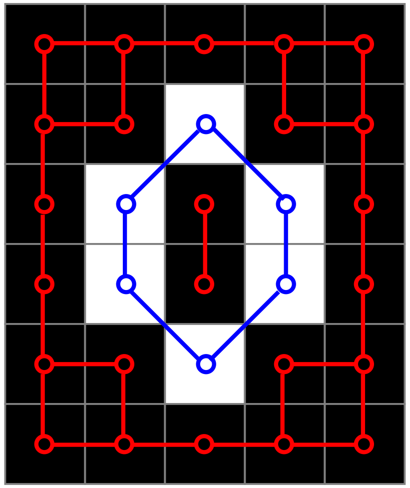


Object: 4-connectivity (4 comp)
Background: 4-connectivity (2 comp)

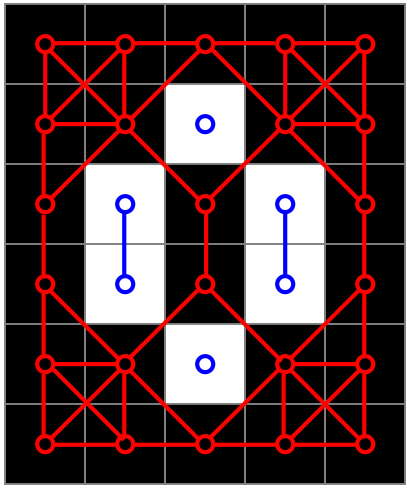
Paradox: a closed curve does not disconnect the background, while an open curve does.

Connectivity (2D)

- Different connectivity for object (O) and background (B)



Object: 8-connectivity (1 comp)
Background: 4-connectivity (2 comp)

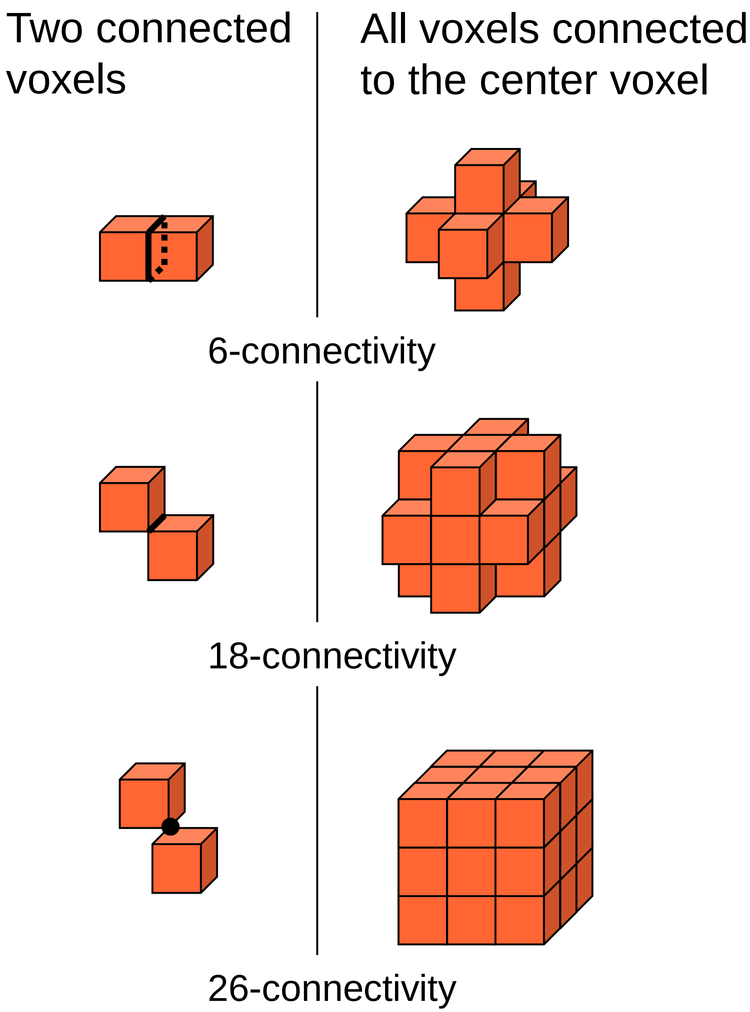


Object: 4-connectivity (4 comp)
Background: 8-connectivity (1 comp)

Connectivity (3D)

Washington University in St. Louis
School of Engineering & Applied Science

- Two voxels are connected if their cubes share:
  - A common face
    - 6-connectivity
  - A common edge
    - 18-connectivity
  - A common vertex
    - 26-connectivity
- Use 6- and 26-connectivity respectively for O and B (or B and O)



Finding Connected Components

- The "flooding" algorithm
  - Start from a seed pixel/voxel, expand the connected component
  - Either do depth-first or breadth-first search (a LIFO stack or FIFO queue)

```
// Finding the connected component containing an object pixel p
1. Initialize
     1. Create a result set S that contains only p
     2. Create a Visited flag at each pixel, and set it to be
        False except for p
     3. Initialize a queue (or stack) Q that contains only p.
2. Repeat until Q is empty:
     1. Pop a pixel x from Q.
     2. For each unvisited object pixel y connected to x, add y
        to S, set its flag to be visited, and push y to Q.
3. Output S
```

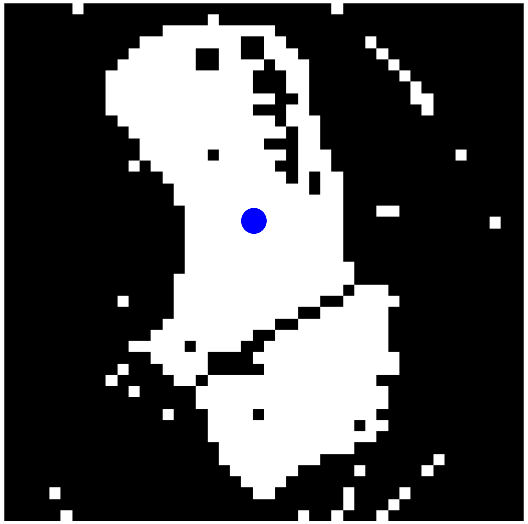
Finding Connected Components

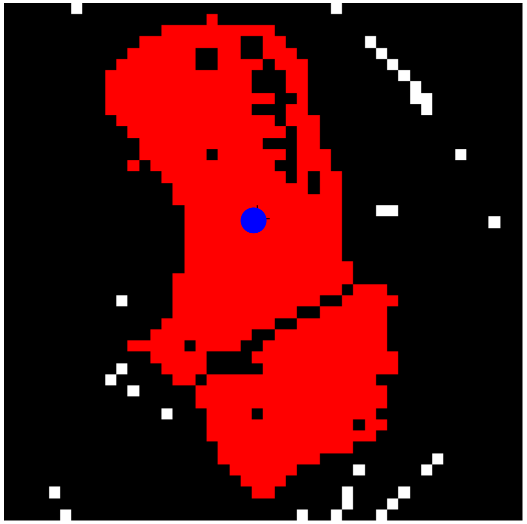
- Why using a "visited" flag?
  - Otherwise, the program will not terminate
- Why not checking to see if y is in S?
  - Checking the visited flag is much faster (O(1) vs. O(log n))

```
1.  …
2. Repeat until Q is empty:
    1. Pop a pixel x from Q.
    2. For each unvisited object pixel y connected to x, add y
       to S, set its flag to be visited, and push y to Q.
3. Output S
```

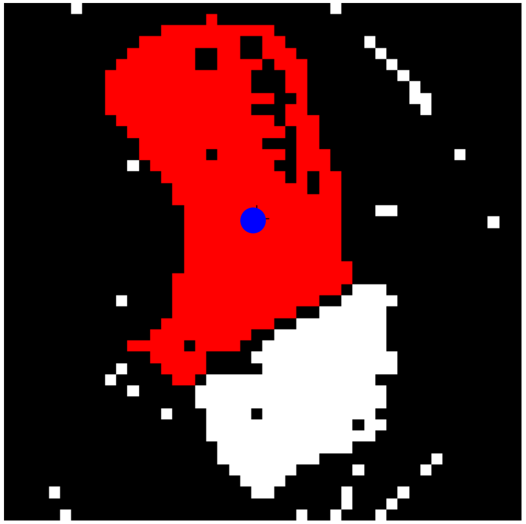
Connectivity (2D)

- Connected components containing the blue pixel:





8-connectivity



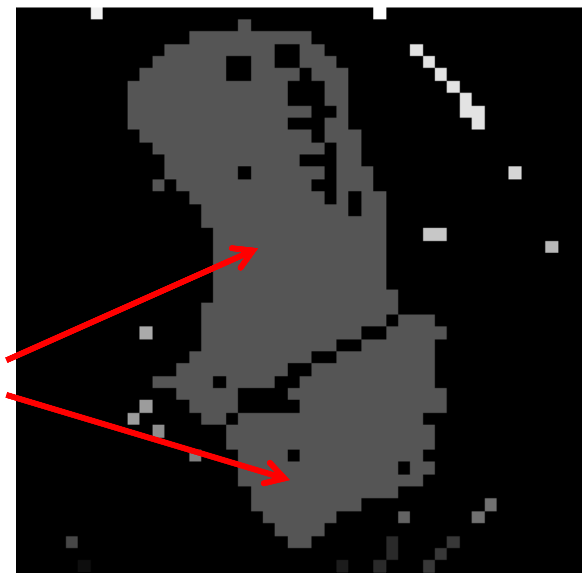
4-connectivity

Finding Connected Components

Washington University in St. Louis
School of Engineering & Applied Science

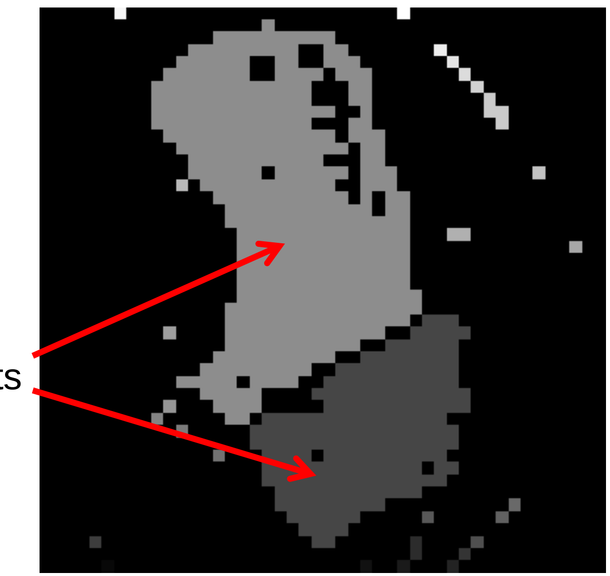
- Labeling all components in an image:
  - Loop through each pixel (voxel). If it is not labeled, use it as a seed to find a connected component, then label all pixels (voxels) in the component.

One component



8-connected object

Two components

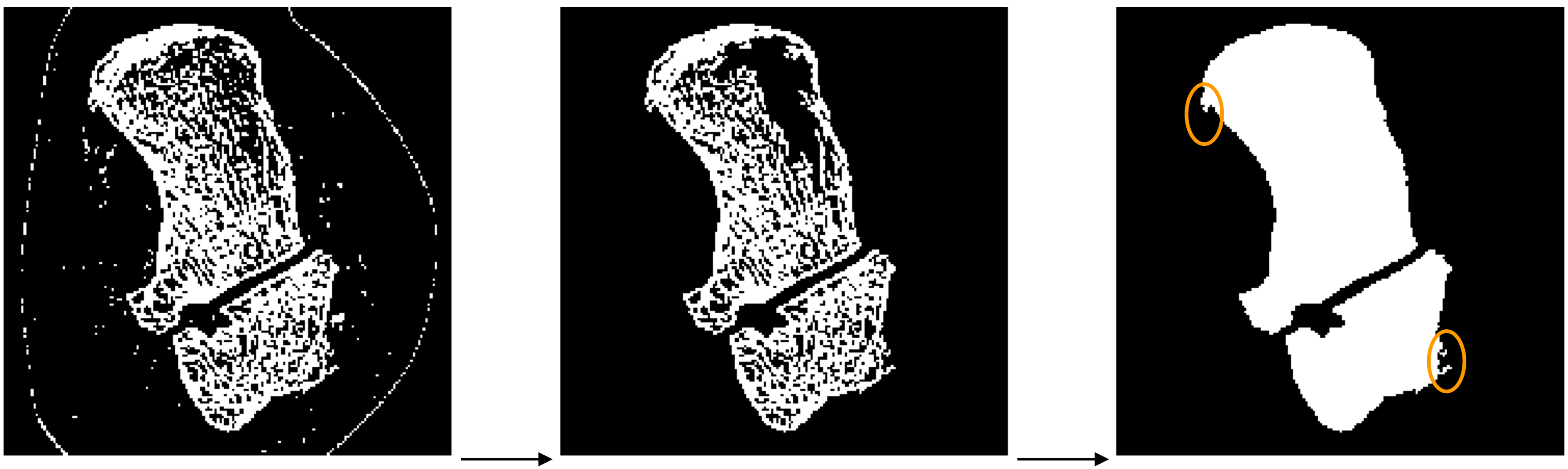


4-connected object

Using Connected Components

Washington University in St. Louis
School of Engineering & Applied Science

- Pruning isolated islands from the main object
- Filling interior holes of the object

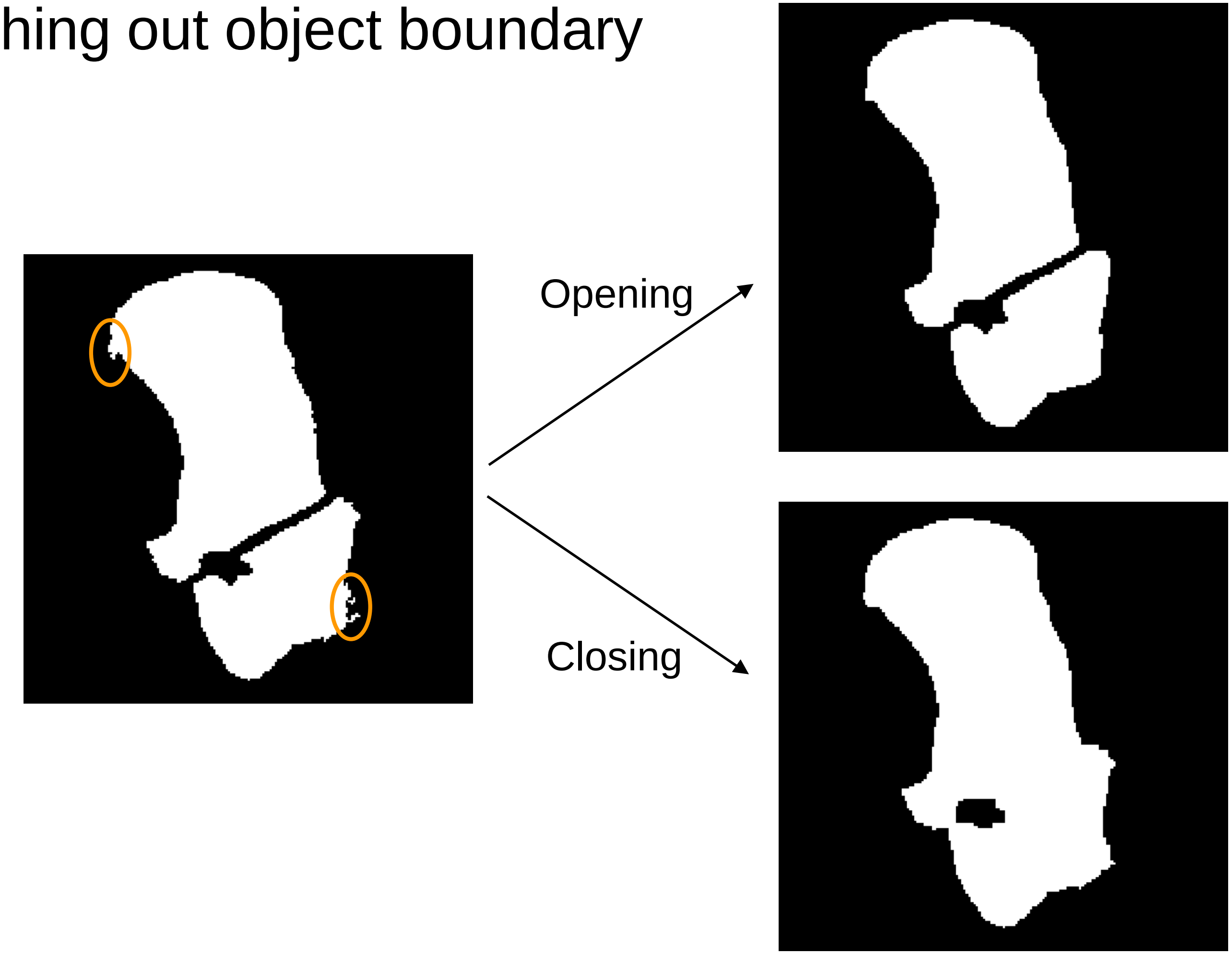


Take the largest components of the object

Invert the largest component of the background

Morphological Operators

- Smoothing out object boundary

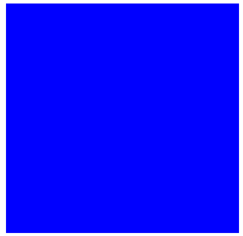


Morphological Operators

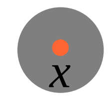
- Operations to change shapes
  - Erosion
  - Dilation
  - Opening: first erode, then dilate.
  - Closing: first dilate, then erode.

Mathematical Morphology

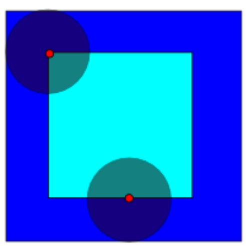
Input:



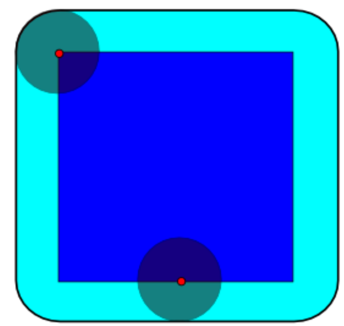
Object (A)



Structure element at x (B_x)



Erosion



Dilation

$$\boldsymbol{A \ominus B = \{x \in A | B_x \subseteq A\}}$$

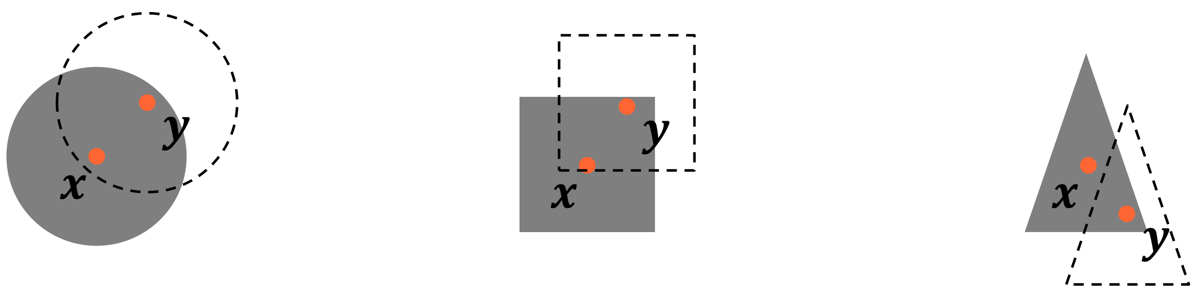
$$\boldsymbol{A \oplus B = \bigcup_{x \in A} B_x}$$

Mathematical Morphology

- Structure element B is symmetric if:

$$\boldsymbol{x} \in \boldsymbol{B_y} \iff \boldsymbol{y} \in \boldsymbol{B_x}$$

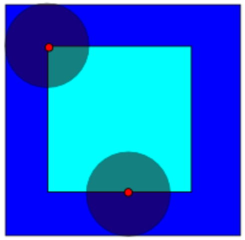
- Examples:



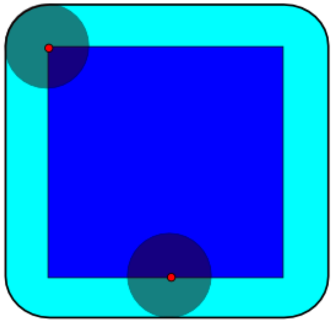
Circle ✓ | Square ✓ | Triangle ✗

Mathematical Morphology

- Duality (for symmetric structuring elements)
  - Erosion (dilation) is equivalent to dilation (erosion) of the background



Erosion



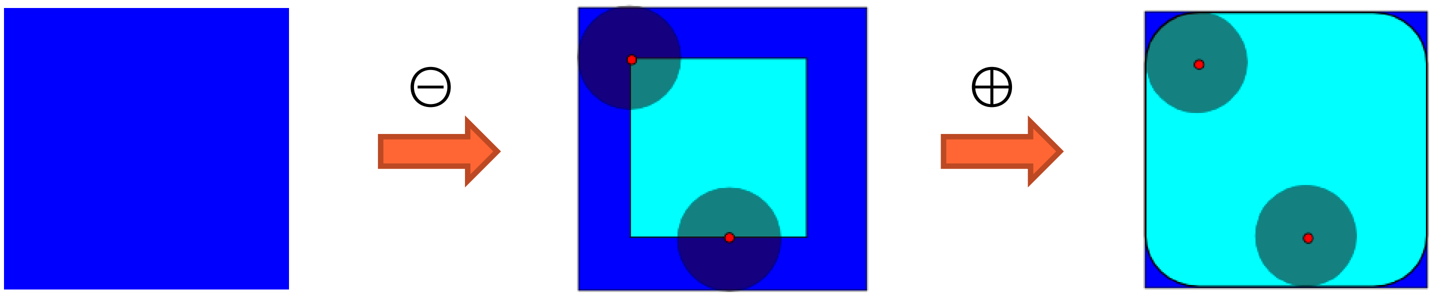
Dilation

$$\boldsymbol{A \ominus B = \overline{\bar{A} \oplus B}}$$

$$\boldsymbol{A \oplus B = \overline{\bar{A} \ominus B}}$$

Mathematical Morphology

- Opening (erode, then dilate)

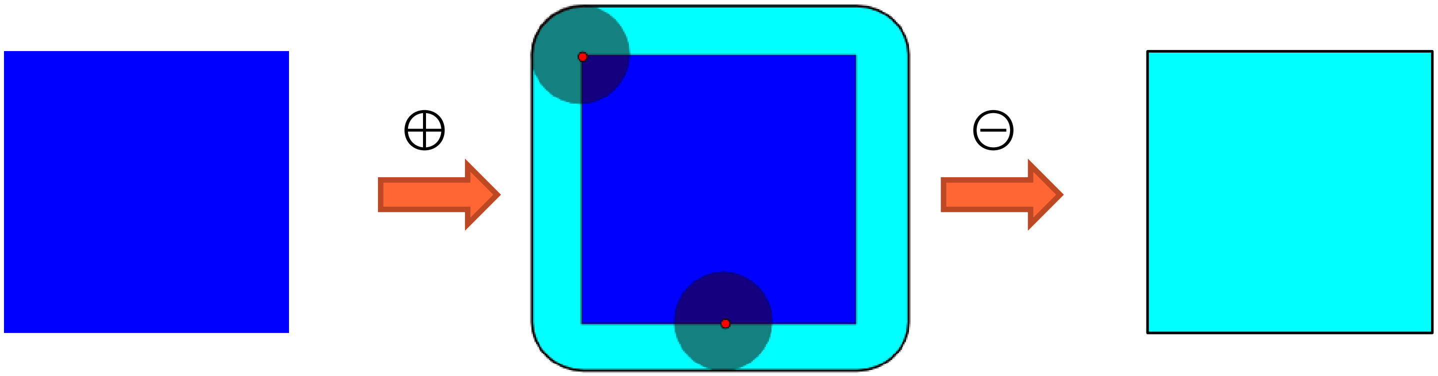


$$\boldsymbol{A} \circ \boldsymbol{B} = (\boldsymbol{A} \ominus \boldsymbol{B}) \oplus \boldsymbol{B}$$

- Union of all structure elements B that can fit inside A
  - Shaves off convex corners and thin spikes

Mathematical Morphology

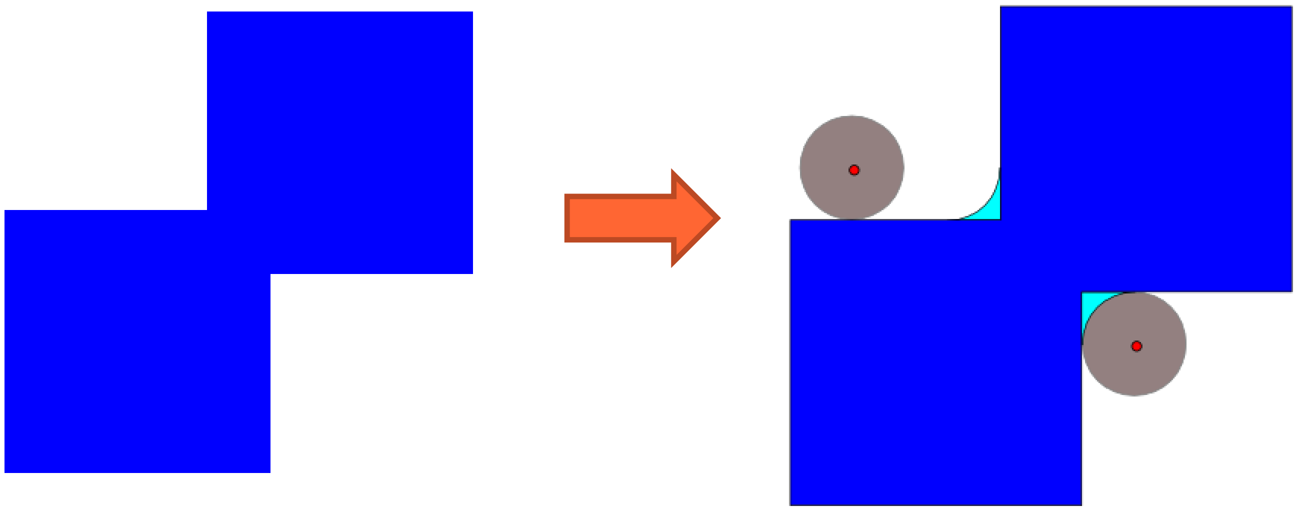
- Closing (dilate, then erode)



$$\boldsymbol{A} \cdot \boldsymbol{B} = (\boldsymbol{A} \oplus \boldsymbol{B}) \ominus \boldsymbol{B}$$

Mathematical Morphology

- Closing (dilate, then erode)

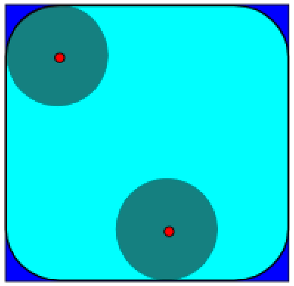


$$A \cdot B = (A \oplus B) \ominus B$$

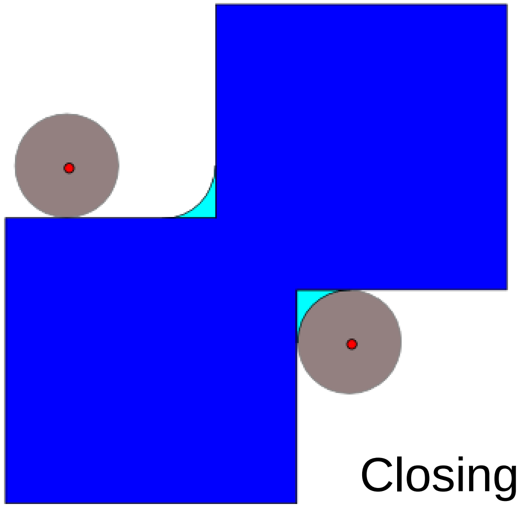
  - Complement of union of all B that can fit in the complement of A
    - Fills concave corners and thin tunnels

Mathematical Morphology

- Duality, again! (for symmetric structuring elements)
  - Opening (closing) object is equivalent to closing (opening) background



Opening



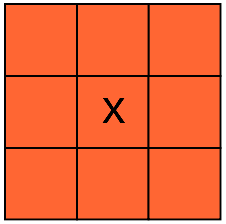
Closing

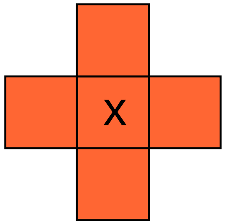
$$A \circ B = \overline{\bar{A} \cdot B}$$

$$A \cdot B = \overline{\bar{A} \circ B}$$

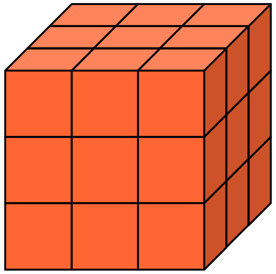
Digital Morphology

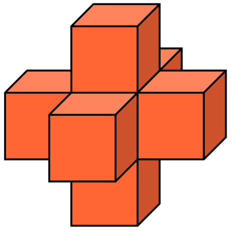
- Structuring elements (symmetric)
  - 2D pixels: square or cross





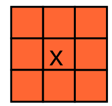
  - 3D voxels: cube or cross



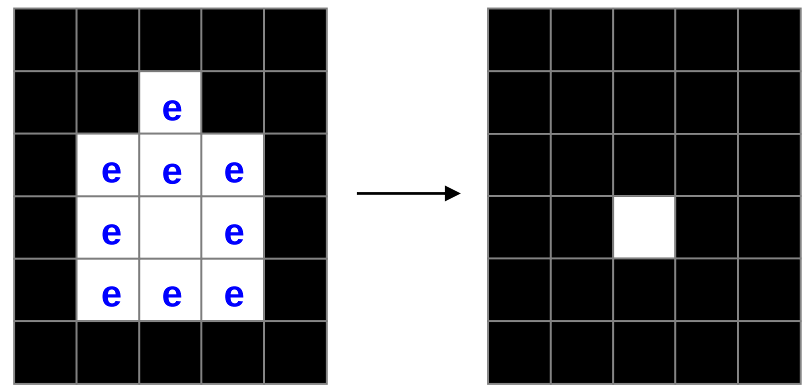


Digital Morphology

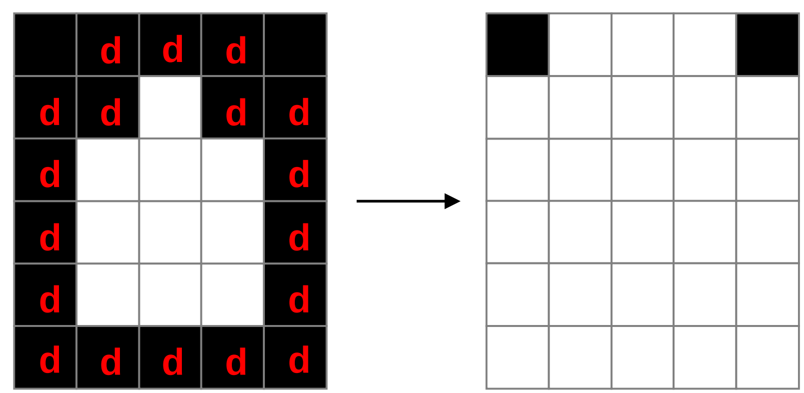
Washington University in St. Louis
School of Engineering & Applied Science

- Structuring element: 

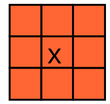
  - **Erosion**
    - **e**: an object pixel with some background pixel in its square neighborhood



  - **Dilation**
    - **d**: a background pixel with some object pixel in its square neighborhood

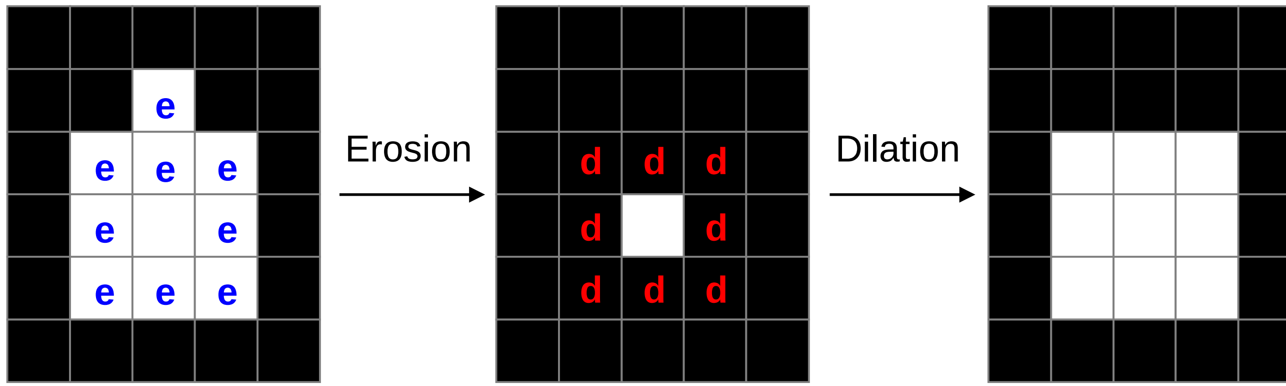


Digital Morphology

- Structuring element: 

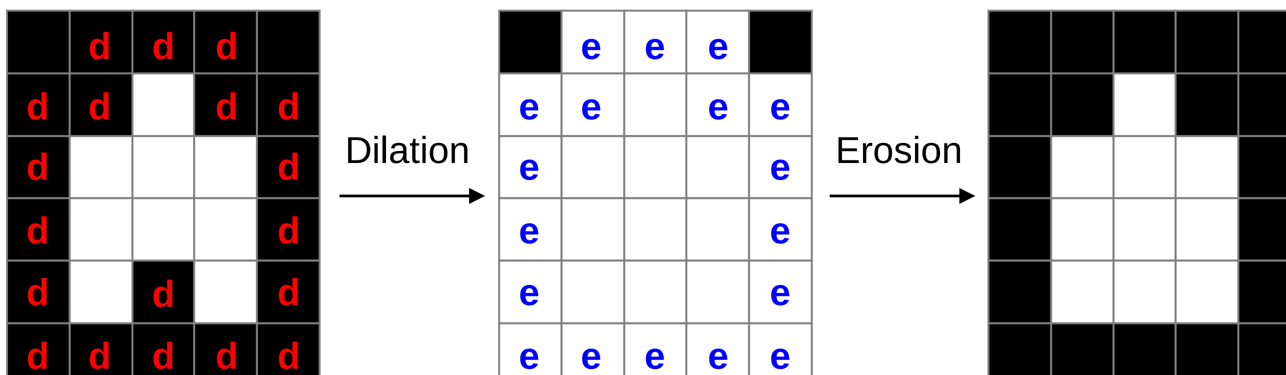
  – **Opening**

  Union of 3x3 white squares that fit inside object



  – **Closing**

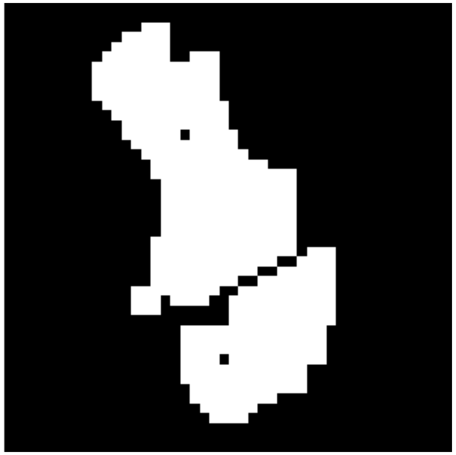
  Union of 3x3 black squares that fit outside object



Digital Morphology

- Increasing the size of the structuring element
  - Leads to more growing/shrinking and more significant smoothing

Original



Opening by 3x3 square

Opening by 5x5 square

  - Equivalent to repeated applications with a small structuring element
    - E.g.: k erosions (dilations) followed by k dilation (erosions) with a 3x3 square is equivalent to opening (closing) with a (2k+1)x(2k+1) square.

Digital Morphology

- Implementation tips
  - Using duality of erosion and dilation, you only need to implement one function to do both morphological operations (for symmetric structure elements).
    - Dilation is same as erosion of the background
  - When performing multiple-round opening, make sure you first do k times erosion then k times dilation
    - What happens if you alternate erosion and dilation for k times?
  - Handle image boundary in a graceful way (not crashing the program…)
    - For example, treat outside of the image as background

Lab Module 1

- A simple 2D segmentation routine
  - Initial segmentation using thresholding (using your code from Lab 0)
  - Using connected components and opening/closing to “clean up” the segmentation.