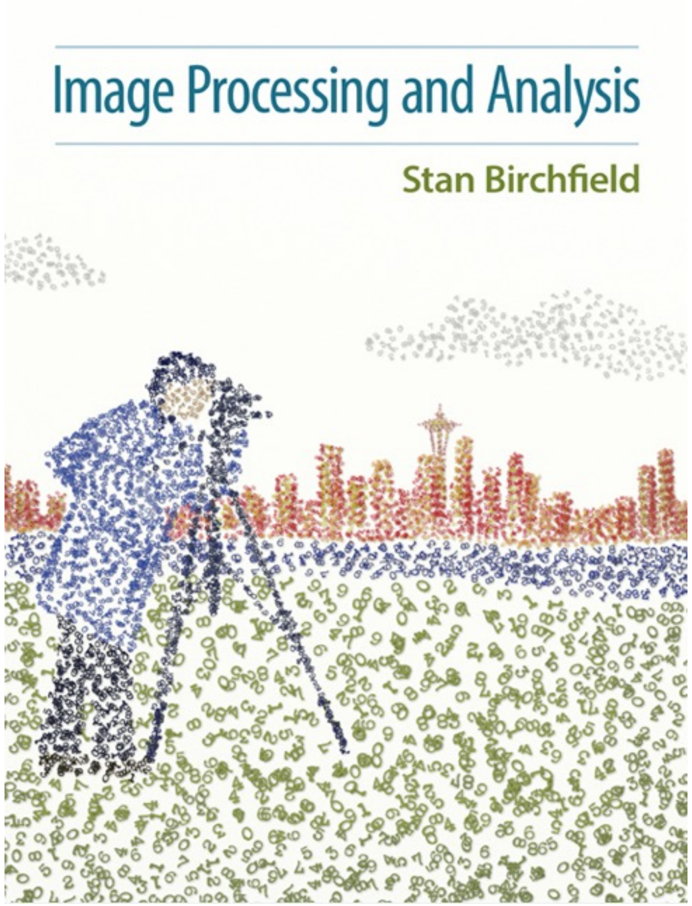
Prof. Kjersti Engan

ELE510 Image processing and computer vision

Binary Image Processing

University of Stavanger

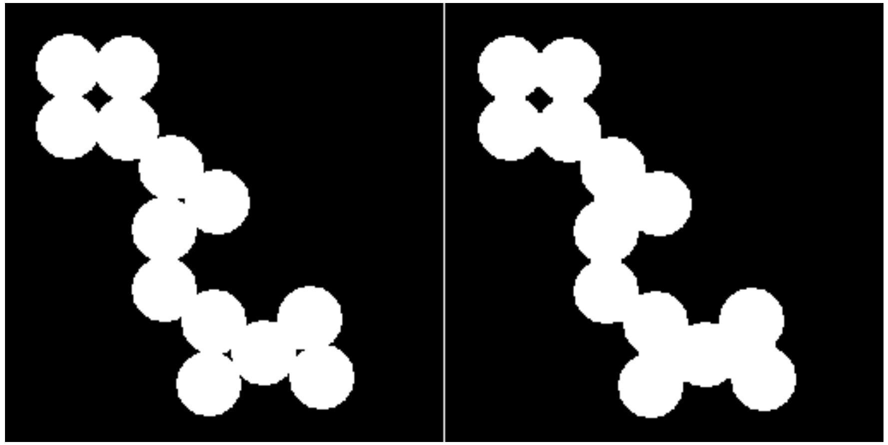


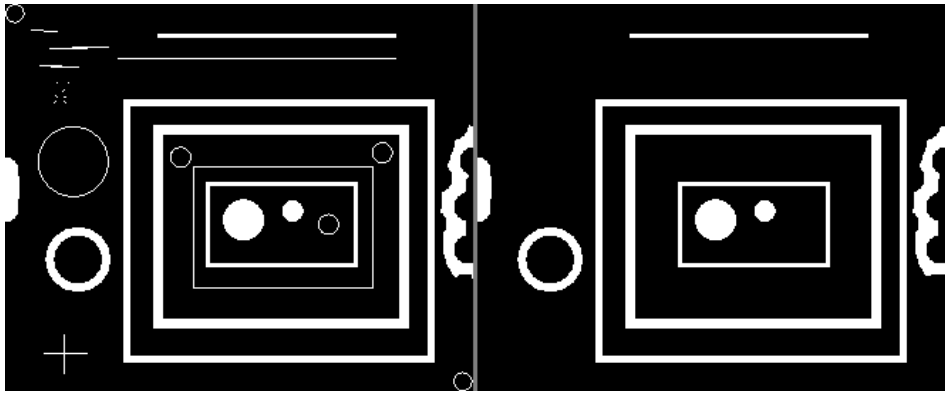
Parts in course presentations is material from Cengage learning. It can be used for teaching, and it can be share it with students on access controlled web-sites (Canvas) for use in THIS course. **But it should not be copied or distributed further in any way** (by you or anybody).

Binary image processing – morphological operations (3.1)

Three points from the topic:

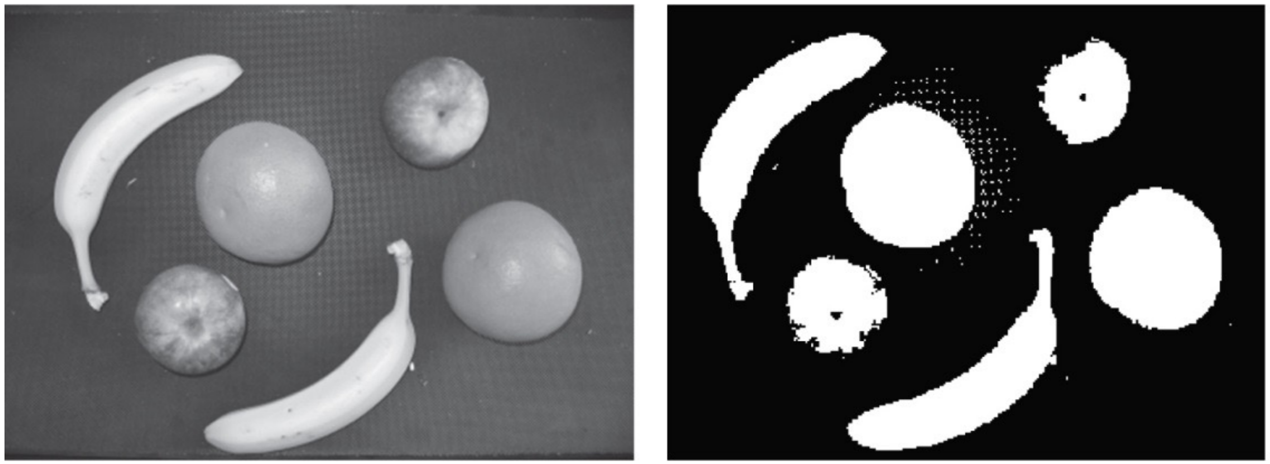
1. Binary images as a set
2. Opening and closing of a binary image
3. Hit and miss operator





Mathematical morphology

- **Mathematical morphology:** a branch of mathematics developed to process images by considering the shape of the pixel regions.
- A **binary image** is an array of values such that *I*(*x, y*) has 1 or 0 for each pixel location (*x, y*).



Stan Birchfield

Binary images as a set

- fundamental set operators:

Z is a point in the plane, i.e. (x,y) coordinates
A and B are sets of points in the plane

$$\mathcal{A} \bigcup \mathcal{B} \equiv \{\mathbf{z} : \mathbf{z} \in \mathcal{A} \text{ or } \mathbf{z} \in \mathcal{B}\} \qquad \text{(union)}$$

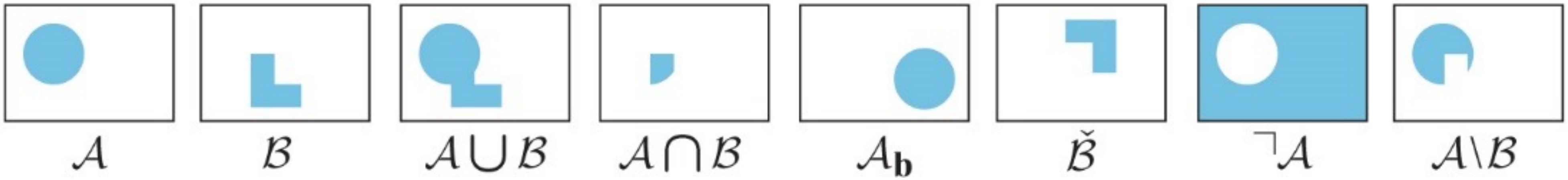
$$\mathcal{A} \bigcap \mathcal{B} \equiv \{\mathbf{z} : \mathbf{z} \in \mathcal{A} \text{ and } \mathbf{z} \in \mathcal{B}\} \qquad \text{(intersection)}$$

$$\mathcal{A}_{\mathbf{b}} \equiv \{\mathbf{z} : \mathbf{z} = \mathbf{a} + \mathbf{b}, \mathbf{a} \in \mathcal{A}\} \qquad \text{(translation)}$$

$$\check{\mathcal{B}} \equiv \{\mathbf{z} : \mathbf{z} = -\mathbf{b}, \mathbf{b} \in \mathcal{B}\} \qquad \text{(reflection)}$$

$$\neg\mathcal{A} \equiv \{\mathbf{z} : \mathbf{z} \notin \mathcal{A}\} \qquad \text{(complement)}$$

$$\mathcal{A} \setminus \mathcal{B} \equiv \{\mathbf{z} : \mathbf{z} \in \mathcal{A}, \mathbf{z} \notin \mathcal{B}\} = \mathbf{A} \bigcap \neg\mathcal{B} \qquad \text{(difference)}$$

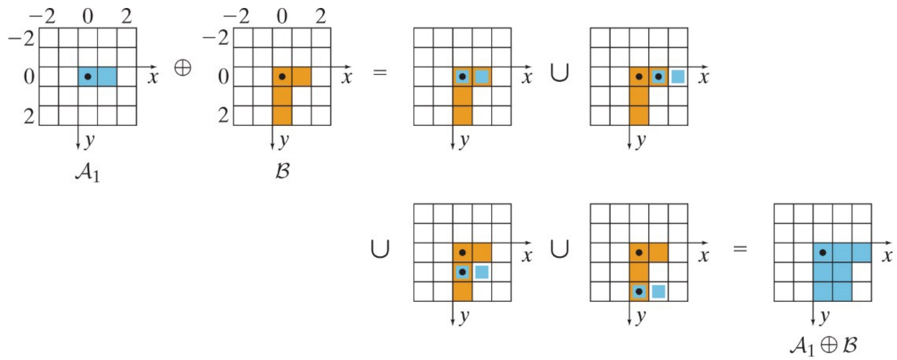


Minkowski addition and subtraction

- The **Minkowski addition** of two sets *A* and *B* is defined as the set of points resulting from all possible vector additions of elements of the two sets:

$$\mathcal{A} \oplus \mathcal{B} \equiv \{\mathbf{z} : \mathbf{z} = \mathbf{a} + \mathbf{b}, \mathbf{a} \in \mathcal{A}, \mathbf{b} \in \mathcal{B}\}$$

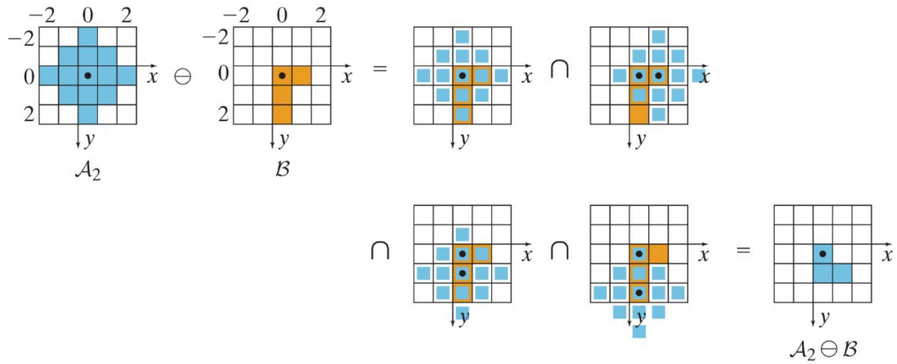
$$= \bigcup_{\mathbf{b} \in \mathcal{B}} \{\mathbf{a} + \mathbf{b} : \mathbf{a} \in \mathcal{A}\} = \bigcup_{\mathbf{b} \in \mathcal{B}} \mathcal{A}_{\mathbf{b}}$$



- **Minkowski subtraction** of two sets:

$$\mathcal{A} \ominus \mathcal{B} \equiv \{\mathbf{z} : \mathbf{z} - \mathbf{b} \in \mathcal{A}, \forall \mathbf{b} \in \mathcal{B}\}$$

$$= \bigcap_{\mathbf{b} \in \mathcal{B}} \{\mathbf{a} + \mathbf{b} : \mathbf{a} \in \mathcal{A}\} = \bigcap_{\mathbf{b} \in \mathcal{B}} \mathcal{A}_{\mathbf{b}}$$



Dilation and Erosion

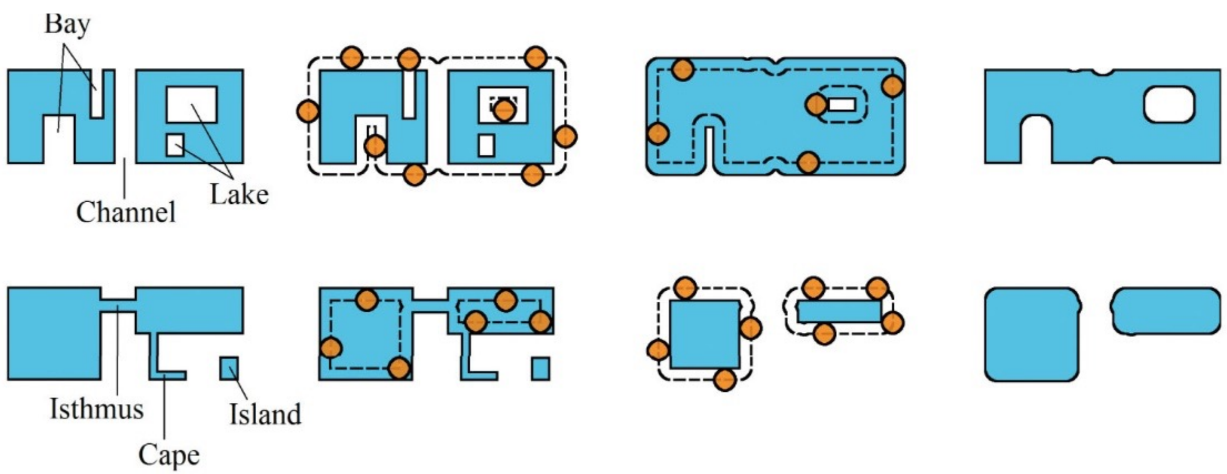
- Based on Minkowski addition and subtraction, we define 2 fundamnetal morphological operators:
- Dilation: identical to Minkowski addition
- Erosion: the Minkowski subtraction after reflecting the second operand -> keep a set of locations where the original set fits inside the other set.

$$\mathcal{A} \oplus \mathcal{B} \equiv A \oplus \mathcal{B} = \{\mathbf{z} : \mathbf{z} = \mathbf{a} + \mathbf{b}, \mathbf{a} \in \mathcal{A}, \mathbf{b} \in \mathcal{B}\} \quad \text{(dilation)}$$

$$\mathcal{A} \check{\ominus} \mathcal{B} \equiv A \ominus \check{\mathcal{B}} = \{\mathbf{z} : \mathbf{z} + \mathbf{b} \in \mathcal{A}, \forall \mathbf{b} \in \mathcal{B}\} \quad \text{(erosion)}$$

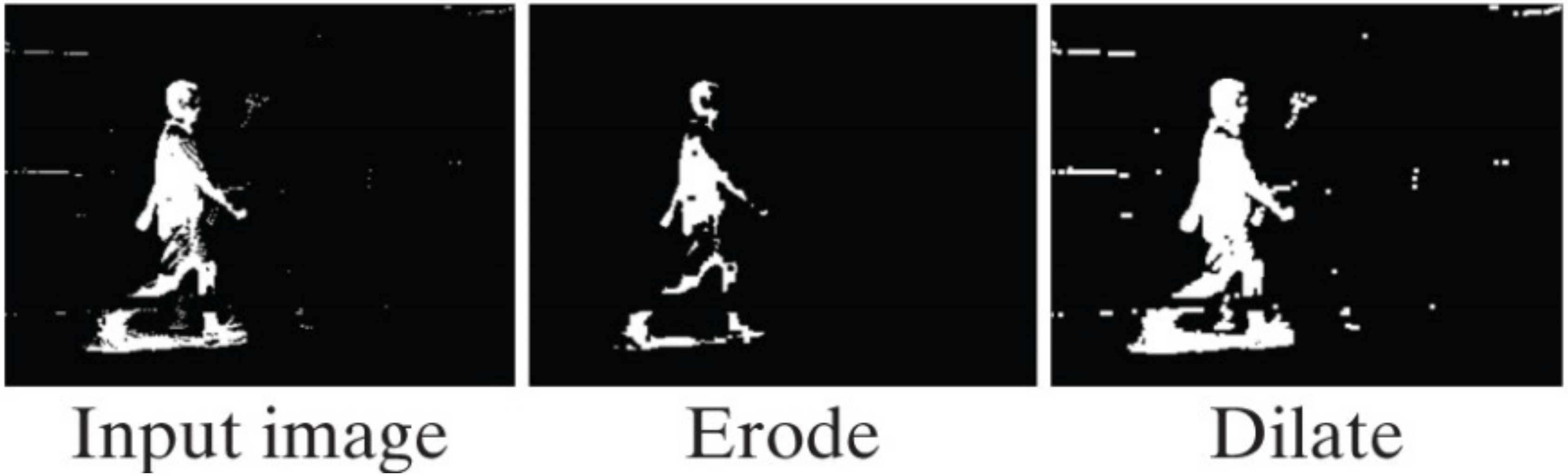
Opening and Closing

- Usually the set B is a structuring element (SE), much smaller than the image A. Can formulate dilation and erosion as translating B across the image performing test (center out).
- Closing is defined as dilation folowed by erosion $A \bullet B$
- Opening is defined as erosion followed by dilation $A \circ B$



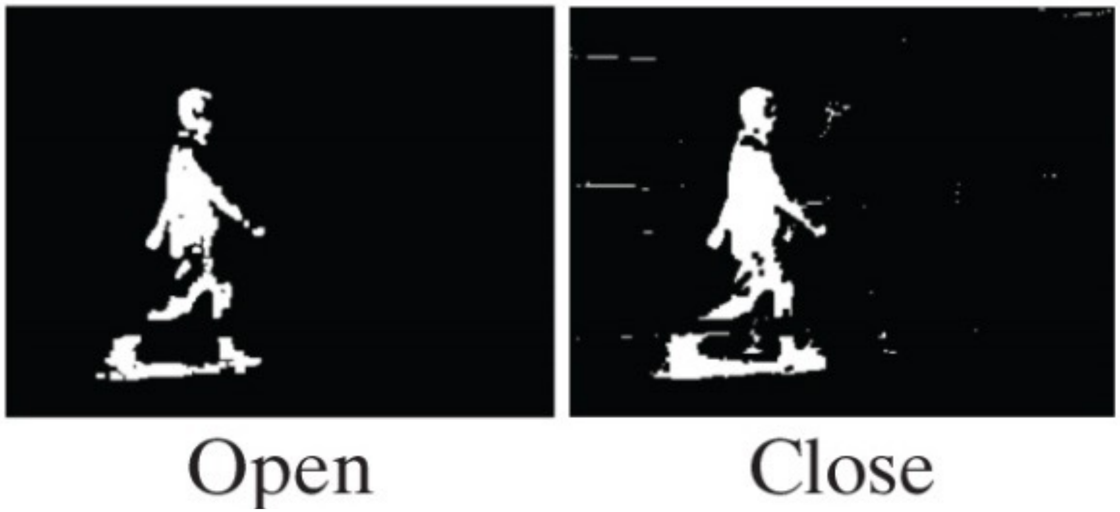
$A \bullet B$

$A \circ B$

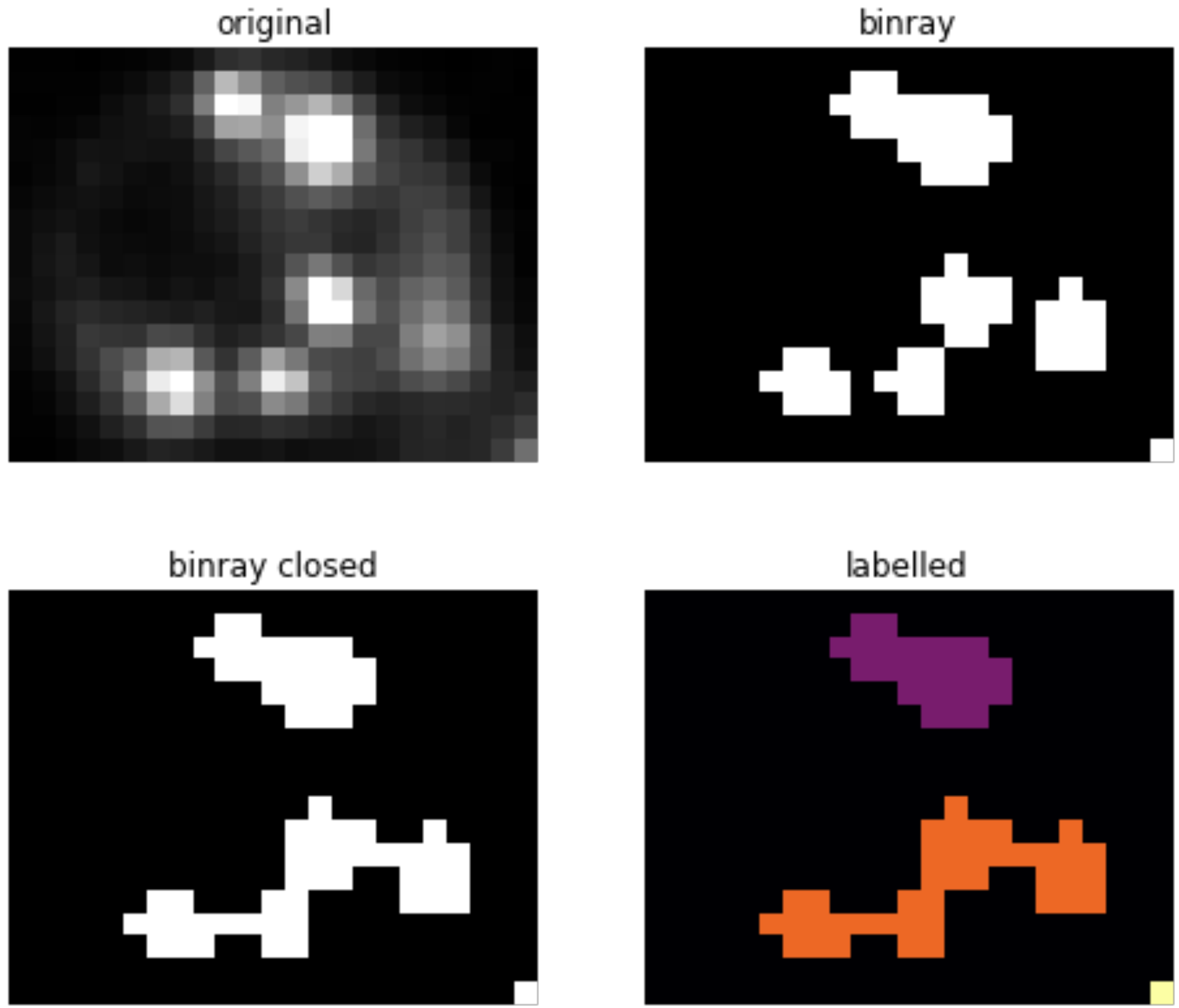


Erosion removes salt noise,
but shrinks foreground.

Dilate fills pepper noise but
expands foreground.



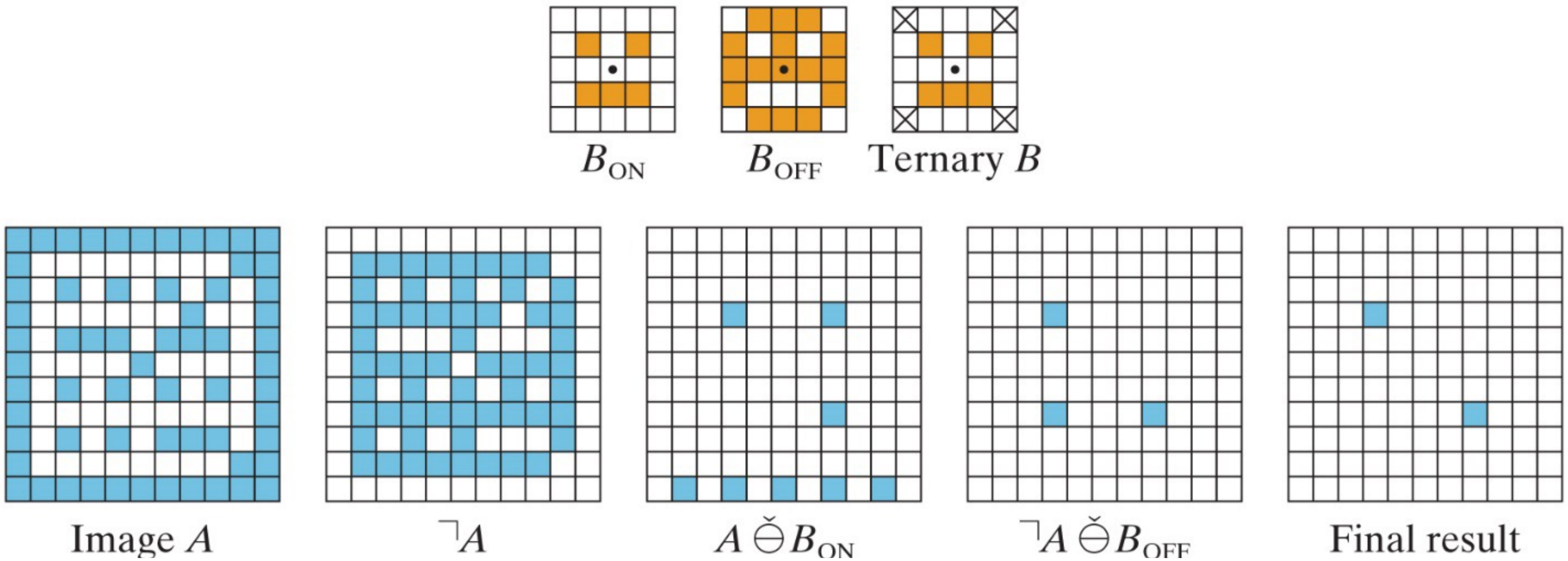
Stan Birchfield


original
binray
binray closed
labelled

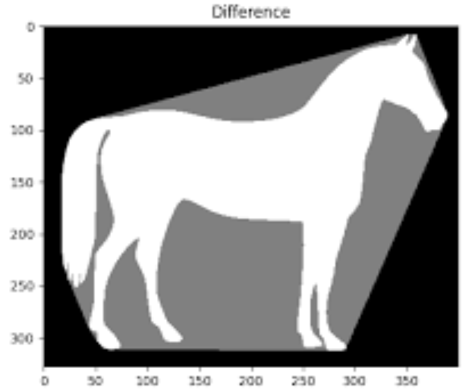
Hit and miss operator

- To detect the shape in the image, the **hit-miss operator** uses erosion to find all the places in the image where B_{ON} matches the foreground and B_{OFF} matches the background:

$$A \circledast (B_{\mathrm{ON}}, B_{\mathrm{OFF}}) \equiv (A \check{\ominus} B_{\mathrm{ON}}) \cap (\neg A \check{\ominus} B_{\mathrm{OFF}}) \qquad \text{(hit-miss operator)}$$



Morphological image processing



- Removing noise (salt and pepper noise)
- Thinning
- Thickening
- Labeling regions
- Region properties – area, perimeter, convex hull, eccentricity
- Boundary tracing – boundary Representations - signatures
- Hole filling
- Computing distances
- Skeletonization

Many intersting examples in book

Distance transform

Distance between two points: $(x_1, y_1), (x_2, y_2)$

Euclidean distance: $D_e = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$

Cityblock distance: $D_{cb} = |x_1 - x_2| + |y_1 - y_2|$

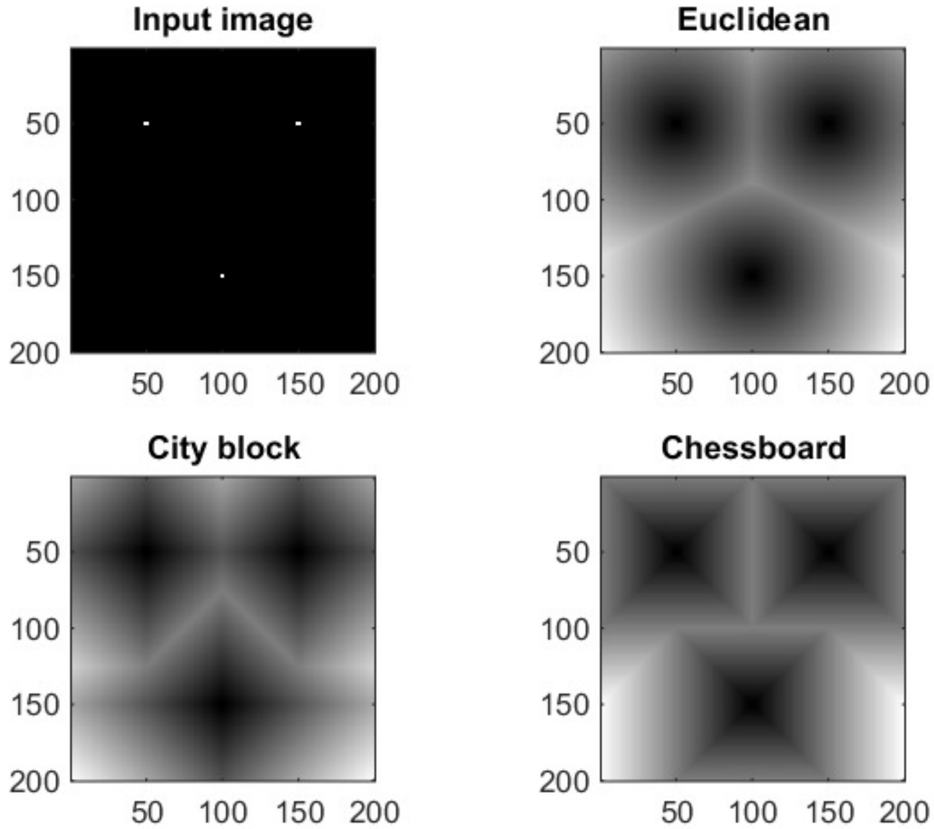
Chessboard distance: $D_{chess} = \max(|x_1 - x_2|, |y_1 - y_2|)$

The distance transform can be found as the distance from each pixel to any given feature.

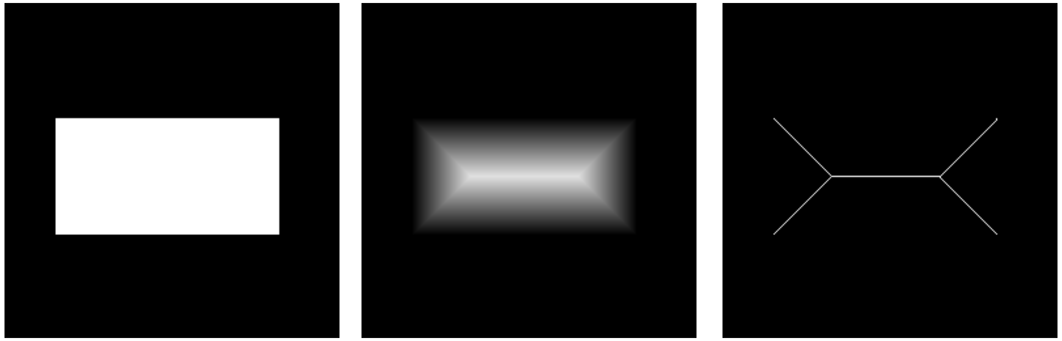
Distance transform

The distance transform on a binary image. For each pixel in the binary image it finds the distance to the nearest non-zero pixel.

Distance can be defined in different ways. Eucledian distance is often used.

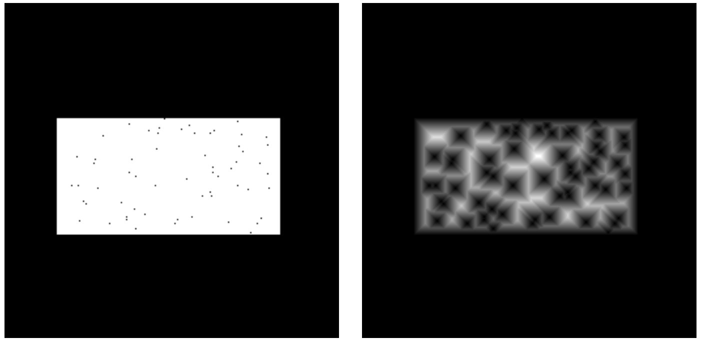


Skeletonization – by distance transform



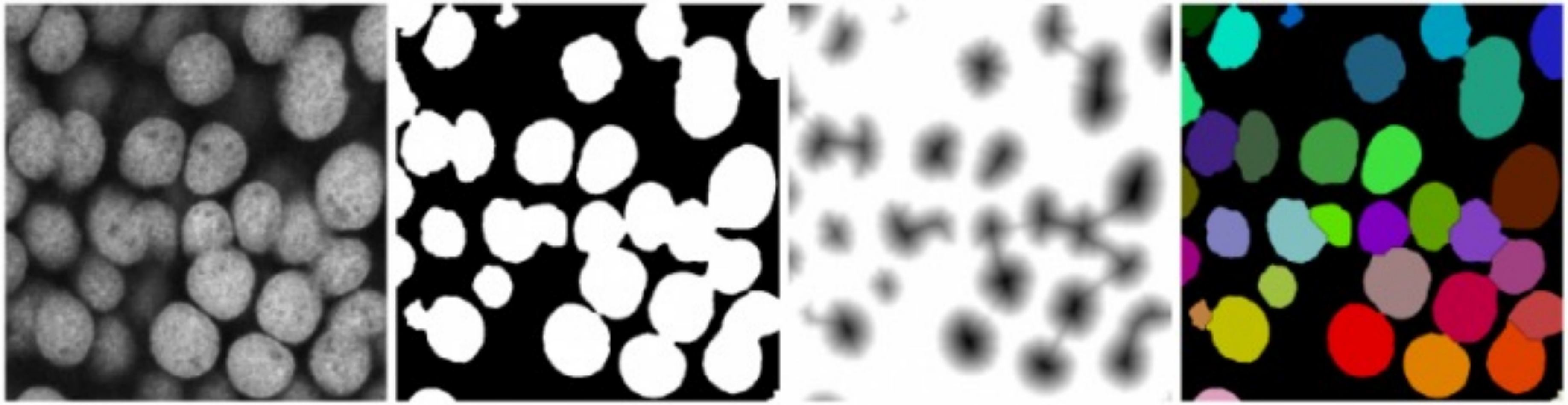
For skeletonization, find distance from each foreground pixel to the nearest background pixel.

Alternatively – skeletonization by *morphological thinning* which is based on a series of hit and miss transforms

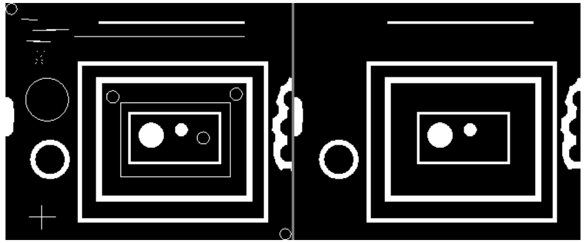


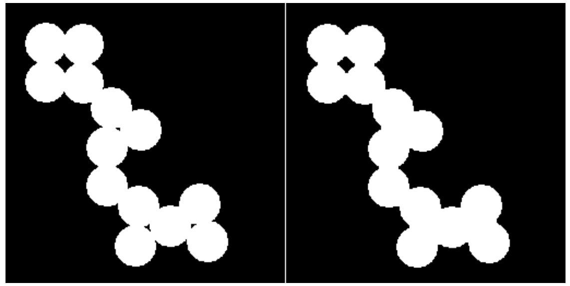
Distance transform is noise sensitive

Distance transform used as part of a segmentation algorithm



Binary image processing – morphological operations (3.1)





Three points from the topic:

1. Binary images as a set
   - ✓ Can use set operators to define operations
2. Opening and closing of a binary image
   - ✓ Open: remove smaller objects etc. Closing: close smaller holes etc.
3. Hit and miss operator
   - ✓ When we want to find an exact pixel structure/object.