Universidade de Aveiro
Departamento de Electrónica,
Telecomunicações e Informática

Image Processing

Edge Detection, Segmentation and Mathematical Morphology

Beatriz Sousa Santos, J. Madeira

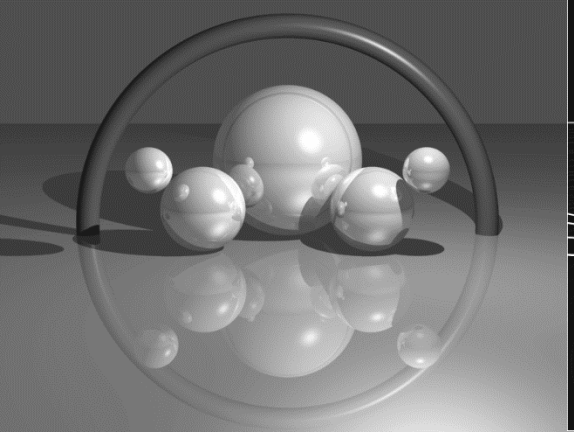
Overview

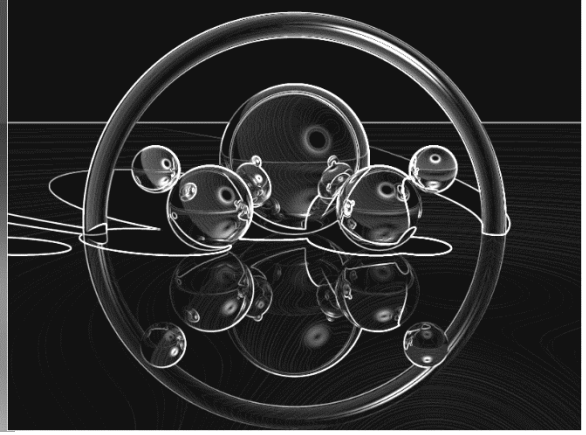
- Edge Detection
- Operators for Edge Detection
- Image Segmentation
- Region Segmentation Algorithms
- Mathematical Morphology

EDGE DETECTION

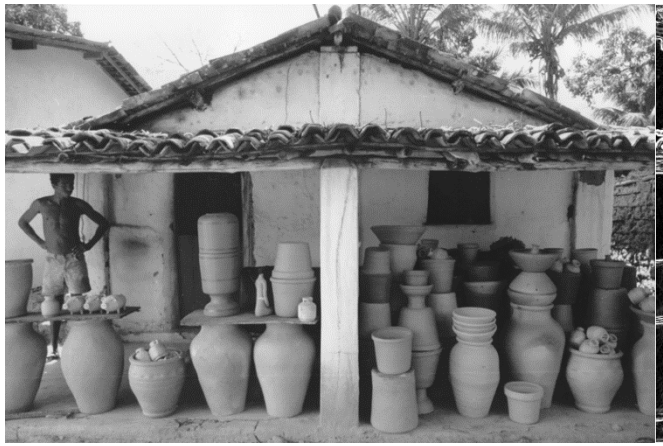
I - Edge Detection

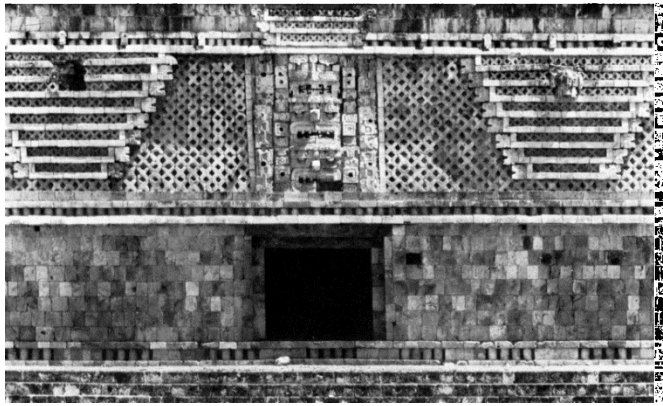
- Edges
- Edge Detection
- Global processing and Hough Transform – Not addressed in our course…
- Edges and Segmentation – Explicit edge detection or countour following

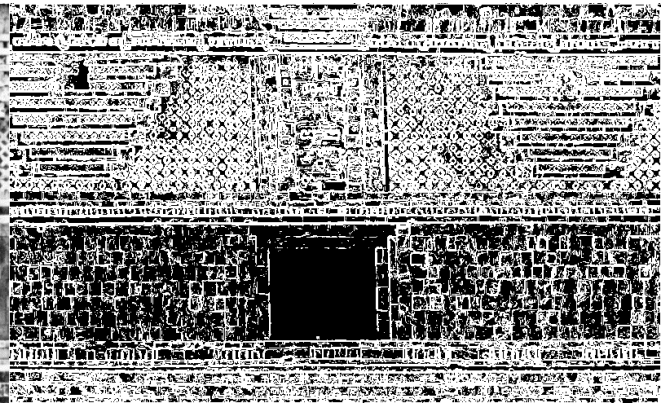




A simple edge detection method applied to images of different complexity

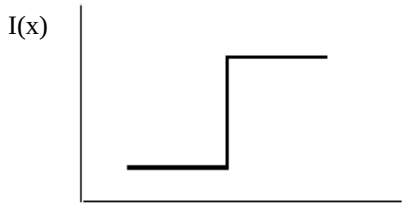






I - 1 - Edges

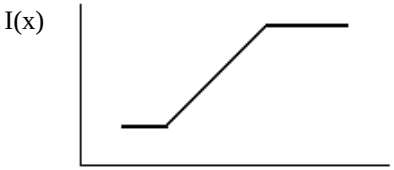
Ideal edge:



- Edges help reveal / understand the contents of an image

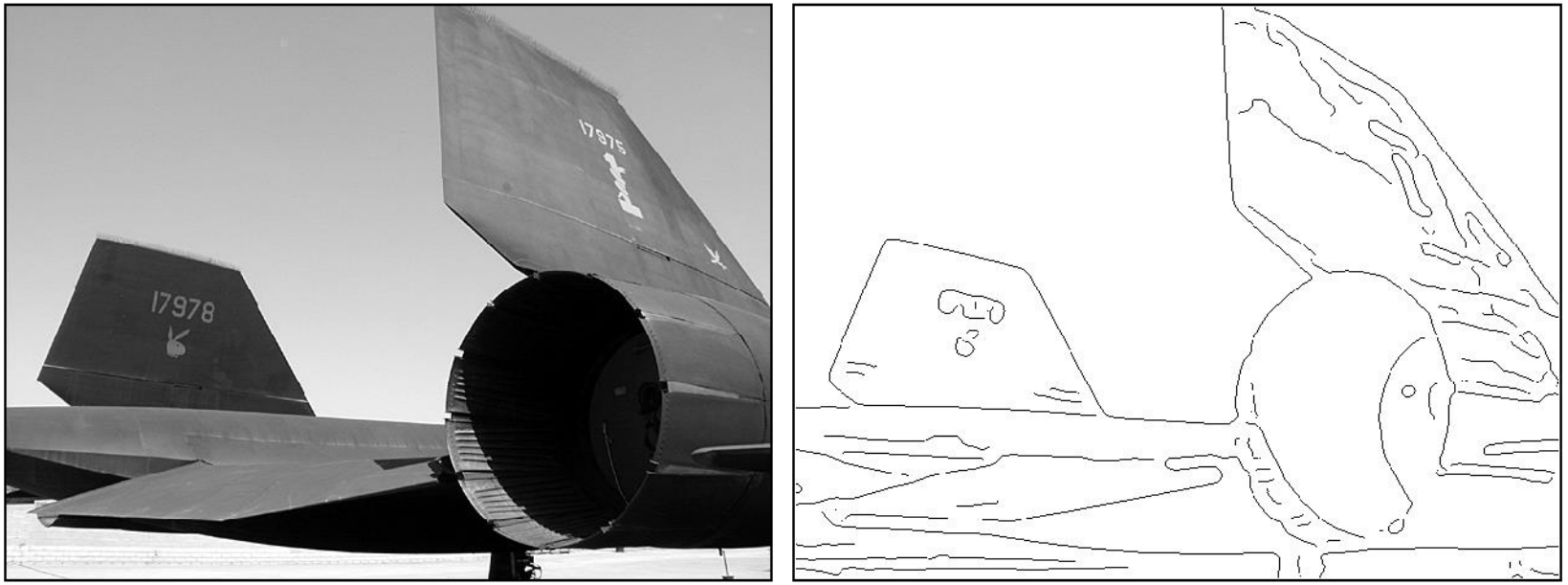
- Edge detection is affected by:
  - noise
  - non-ideal edges

Non-ideal edge:



- Possible origin:
  - Outlines of objects
  - Surface normal vectors discontinuities
  - Illumination discontinuities
  - Reflectance discontinuities

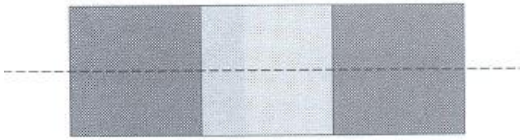
Example



(a) (b)

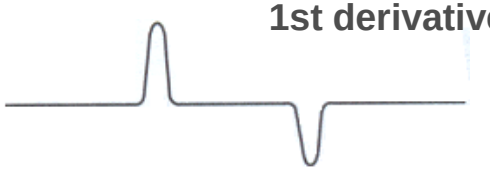
[Burger & Burge]

I - Edge Detection



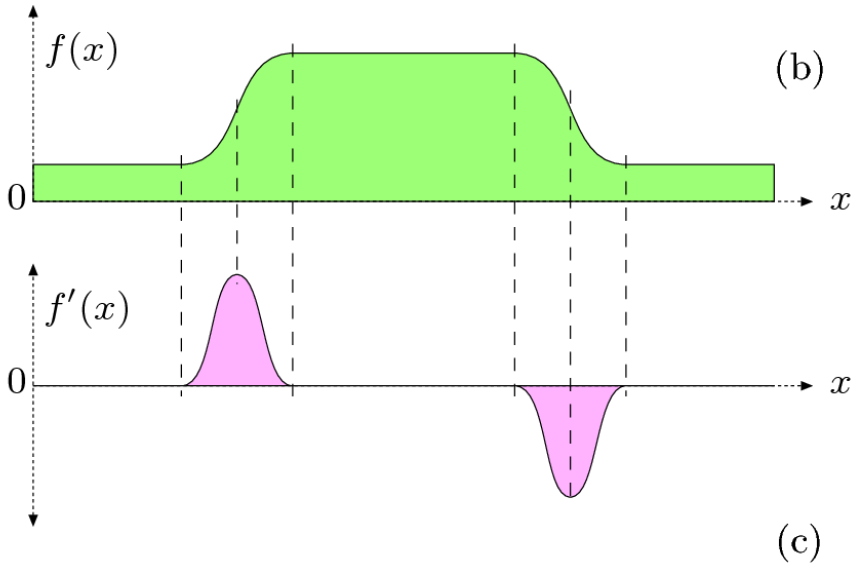
- Edges are usually detected in two steps:
  - Applying a kernel (derivative approximation)
  - Aggregating detected pixels (edgels) into edgels

- Using derivatives along an image row – Main ideas:
  - The 1st derivative is:
    - > 0 or < 0, depending on the variation of *I(x,y)*
    - = 0, in areas of constant *I(x,y)*
  - The 2nd derivative goes through 0 at "positive" and "negative" edges



1st Derivative

(a)

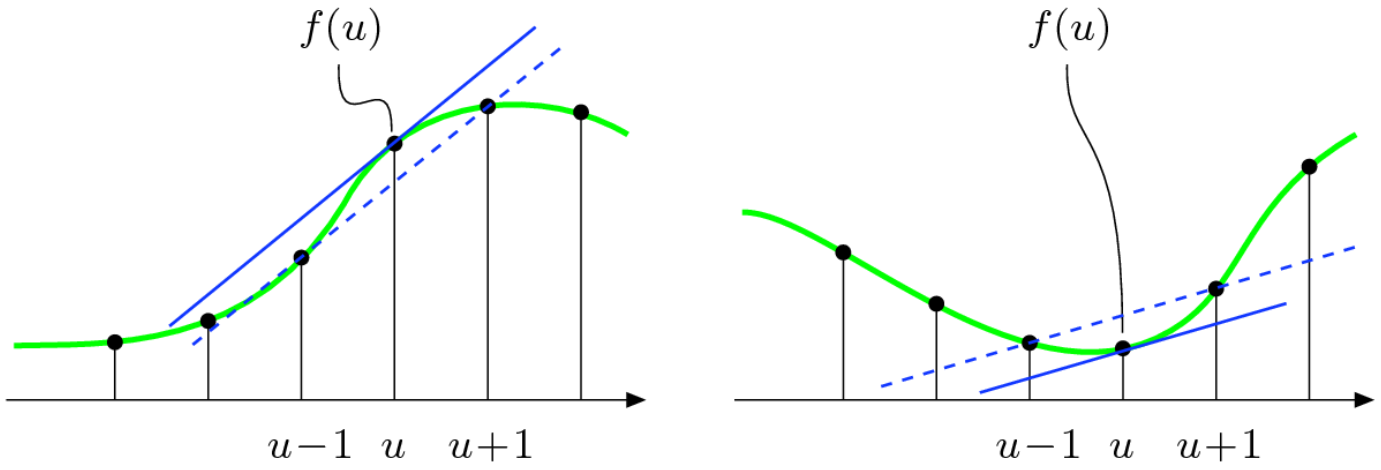


$$f'(x) = \frac{df}{dx}(x)$$

$$\frac{df}{du}(u) \approx \frac{f(u+1) - f(u-1)}{2} = 0.5 \cdot \big(f(u+1) - f(u-1)\big)$$

[Burger & Burge]

1st Derivative – Simple estimation ?



$$\frac{df}{du}(u) \approx \frac{f(u+1) - f(u-1)}{2} = 0.5 \cdot \big(f(u+1) - f(u-1)\big)$$

[Burger & Burge]

Partial Derivatives and Gradient

$$H_x^D = \begin{bmatrix} -0.5 & \mathbf{0} & 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 & \mathbf{0} & 1 \end{bmatrix}$$

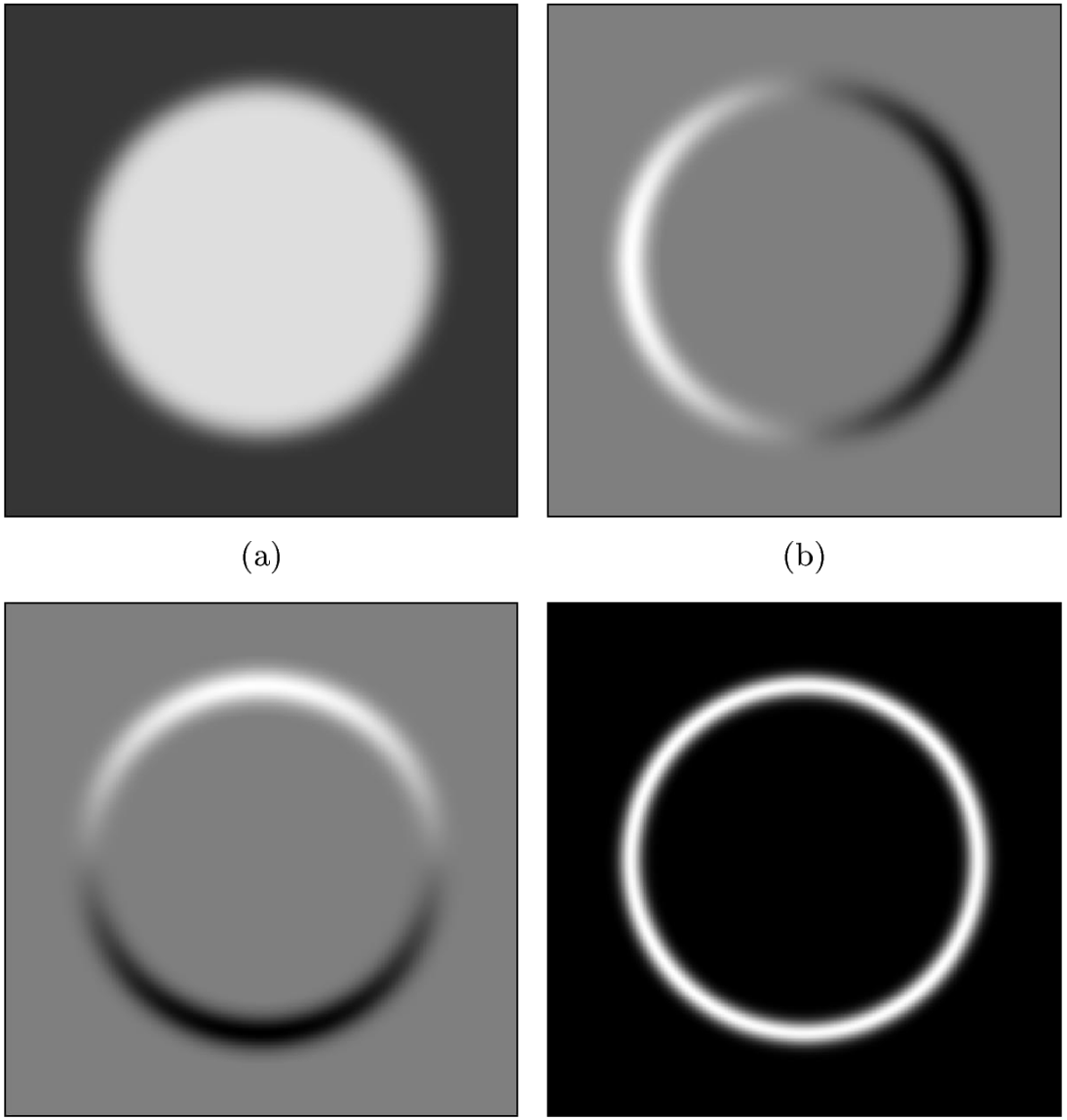
$$H_y^D = \begin{bmatrix} -0.5 \\ \mathbf{0} \\ 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 \\ \mathbf{0} \\ 1 \end{bmatrix}$$

$$\nabla I(u,v) = \begin{bmatrix} \frac{\partial I}{\partial u}(u,v) \\ \frac{\partial I}{\partial v}(u,v) \end{bmatrix}$$

$$|\nabla I|(u,v) = \sqrt{\left(\frac{\partial I}{\partial u}(u,v)\right)^2 + \left(\frac{\partial I}{\partial v}(u,v)\right)^2}$$

[Burger & Burge]

Example

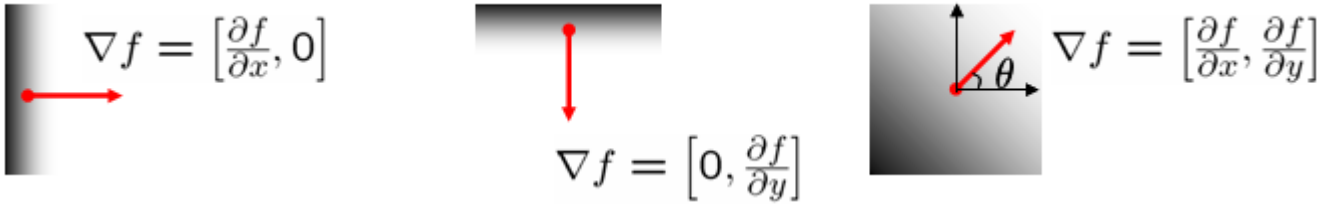


[Burger & Burge]

Edge detection with derivative operators

- The gradient vector of image *f(x,y)* points towards the direction of fastest intensity change

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$



Gradient direction

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

Gradient magnitude

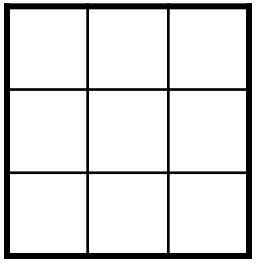
$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

- How to compute derivatives of a discrete image *f(x,y)*?

- One option is to compute finite differences:

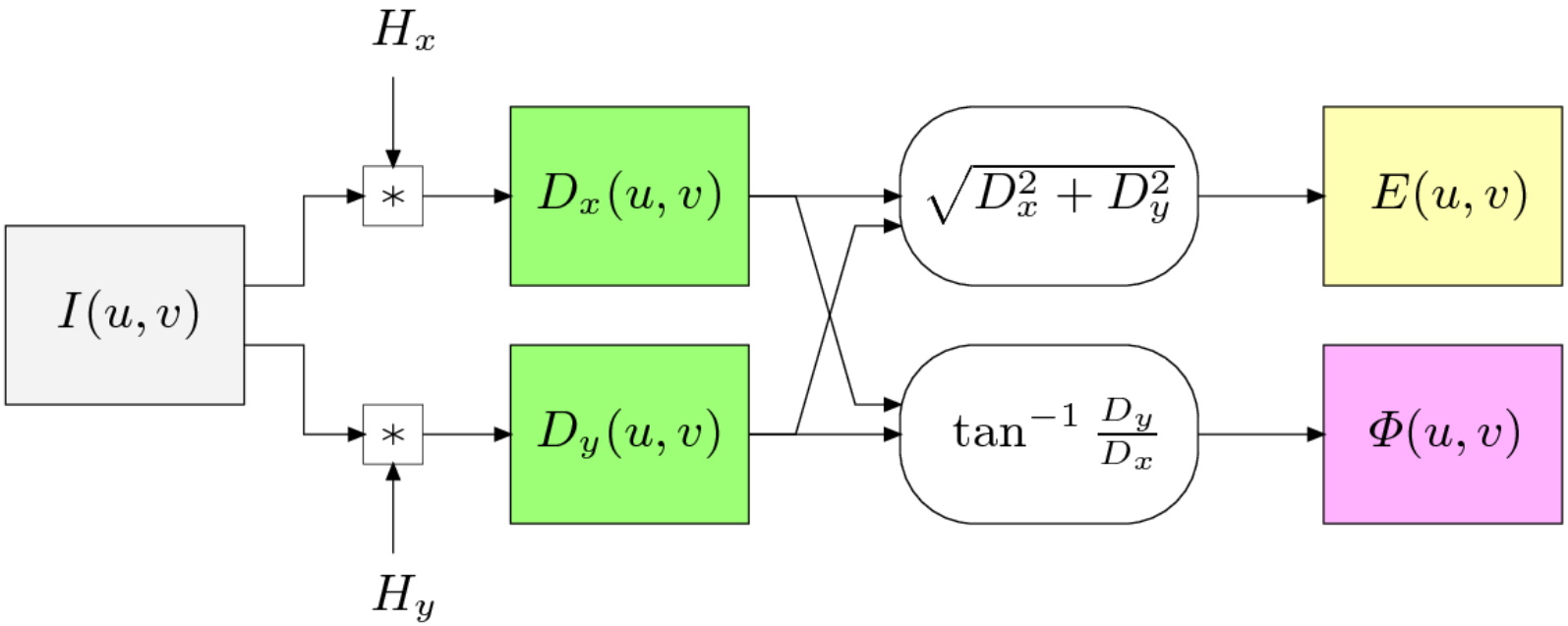
$$\frac{\partial f}{\partial x}[x, y] \approx f[x + 1, y] - f[x, y]$$

and apply the corresponding kernel



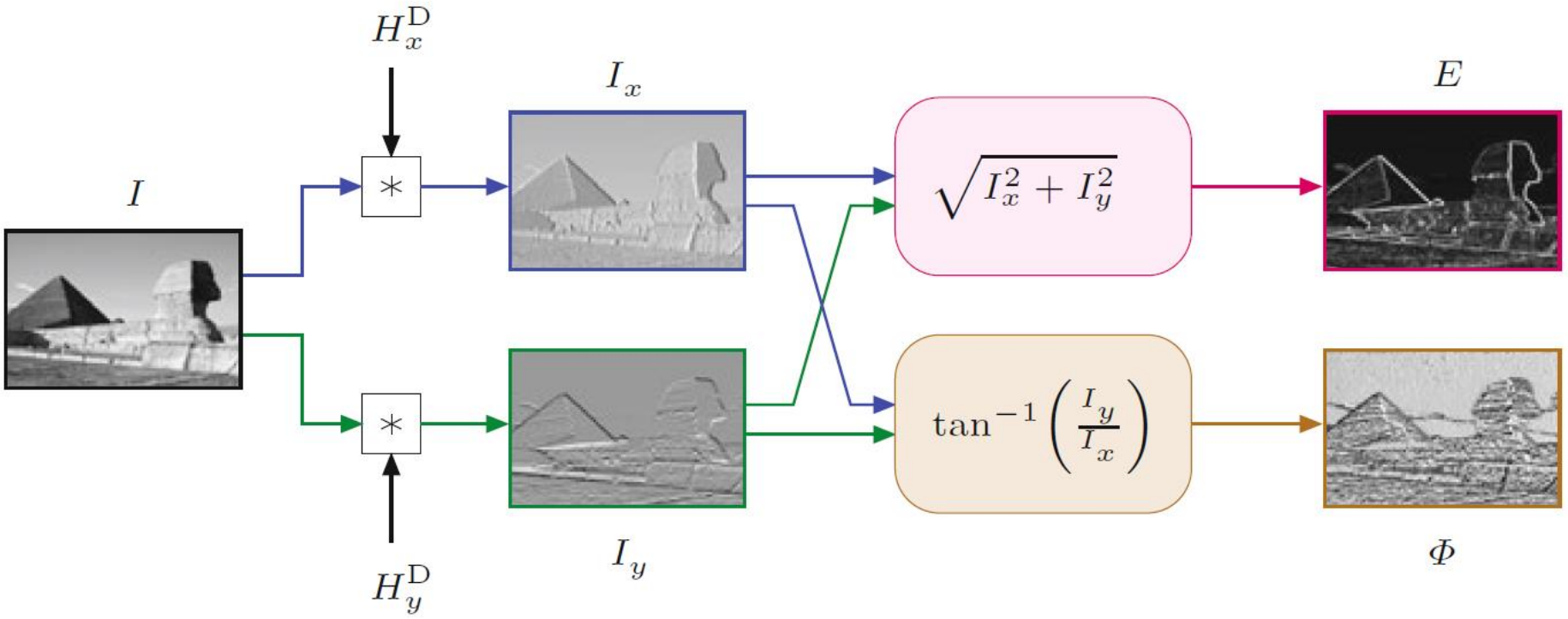
- Several operators that approximate the gradient can be used to detect edges:
  - Roberts
  - Prewitt
  - Sobel

How to apply ?



[Burger & Burge]

How to apply ?



[Burger & Burge]

OPERATORS FOR EDGE DETECTION

Prewitt and Sobel Operators

$$H_x^P = \begin{bmatrix} -1 & 0 & 1 \\ -1 & \mathbf{0} & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^P = \begin{bmatrix} -1 & -1 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\nabla I(u,v) \approx \frac{1}{6} \cdot \begin{bmatrix} \big(I * H_x^P\big)(u,v) \\ \big(I * H_y^P\big)(u,v) \end{bmatrix}$$

$$H_x^S = \begin{bmatrix} -1 & 0 & 1 \\ -2 & \mathbf{0} & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad \text{and} \quad H_y^S = \begin{bmatrix} -1 & -2 & -1 \\ 0 & \mathbf{0} & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

$$\nabla I(u,v) \approx \frac{1}{8} \cdot \begin{bmatrix} \big(I * H_x^S\big)(u,v) \\ \big(I * H_y^S\big)(u,v) \end{bmatrix}$$

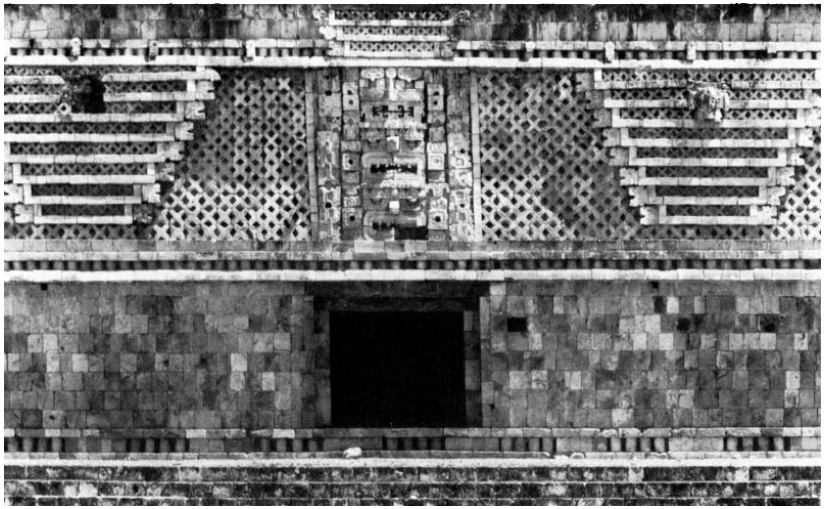
[Burger & Burge]

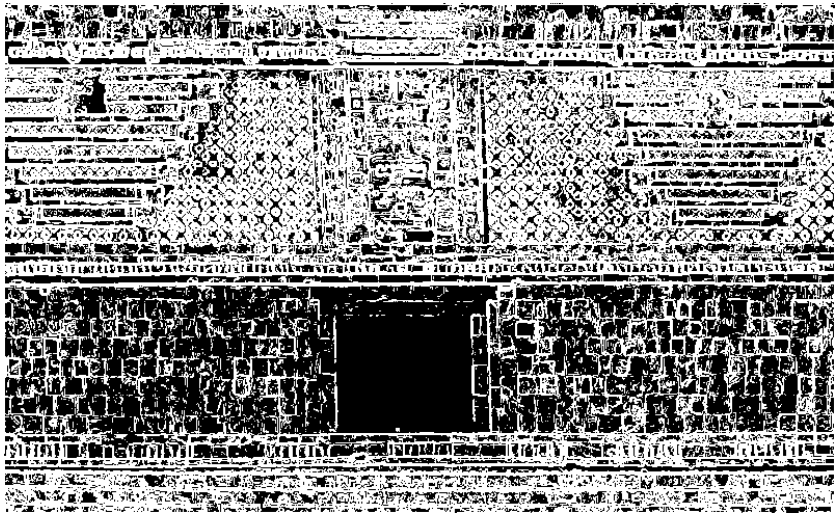
Detecting edges with a pre-defined orientation

Prewitt Operator

1	1	1
0	0	0
-1	-1	-1

-1	0	1
-1	0	1
-1	0	1

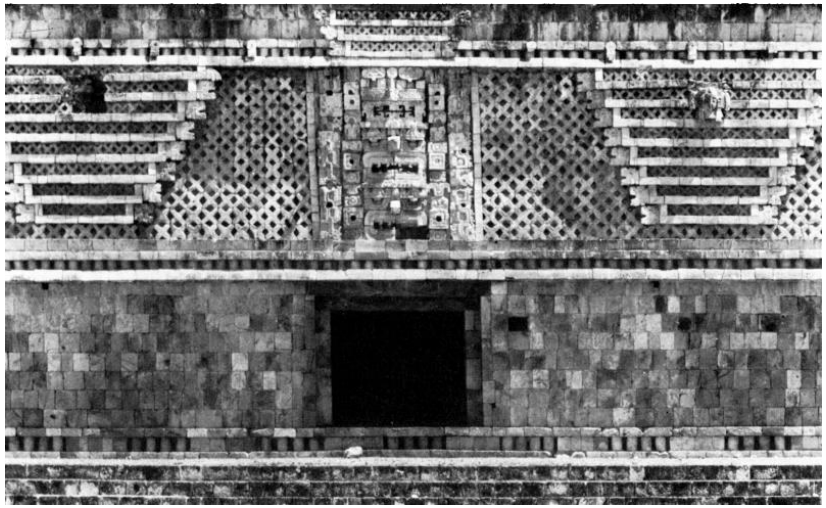


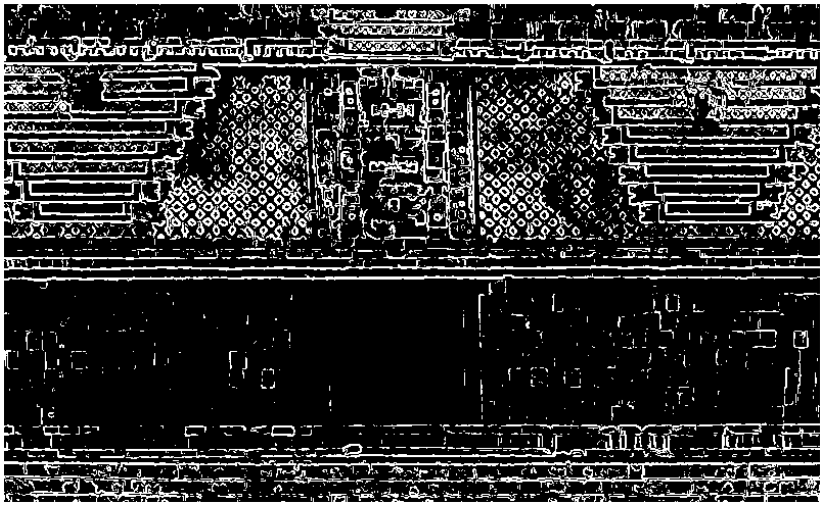


Sobel Operator

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

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

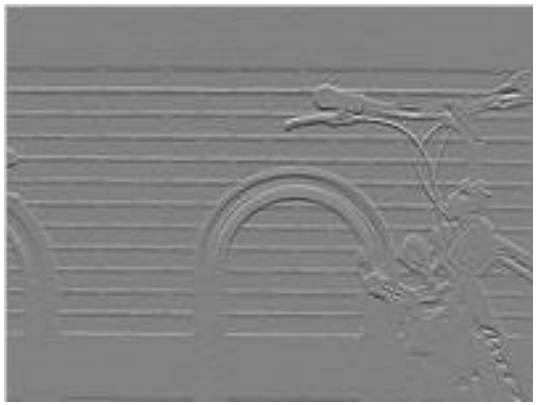




Sobel Operator

Original image

Resulting image



Images resulting from approximating the gradient in the x and y directions

Roberts Operator

$D_1 = I * H_1^R$

$D_2 = I * H_2^R$

$$H_1^R = \begin{bmatrix} 0 & \mathbf{1} \\ -1 & 0 \end{bmatrix} \quad \text{and} \quad H_2^R = \begin{bmatrix} -1 & 0 \\ 0 & \mathbf{1} \end{bmatrix}$$

[Burger & Burge]

Extended Sobel Operator

$$H_0^{\mathrm{ES}} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \qquad H_1^{\mathrm{ES}} = \begin{bmatrix} -2 & -1 & 0 \\ -1 & 0 & 1 \\ 0 & 1 & 2 \end{bmatrix},$$

$$H_2^{\mathrm{ES}} = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}, \qquad H_3^{\mathrm{ES}} = \begin{bmatrix} 0 & -1 & -2 \\ 1 & 0 & -1 \\ 2 & 1 & 0 \end{bmatrix},$$

$$H_4^{\mathrm{ES}} = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, \qquad H_5^{\mathrm{ES}} = \begin{bmatrix} 2 & 1 & 0 \\ 1 & 0 & -1 \\ 0 & -1 & -2 \end{bmatrix},$$

$$H_6^{\mathrm{ES}} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}, \qquad H_7^{\mathrm{ES}} = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 0 & 1 \\ -2 & -1 & 0 \end{bmatrix}.$$

[Burger & Burge]

Kirsch Operator

$$H_0^{\mathrm{K}} = \begin{bmatrix} -5 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & 3 & 3 \end{bmatrix}, \qquad H_4^{\mathrm{K}} = \begin{bmatrix} 3 & 3 & -5 \\ 3 & 0 & -5 \\ 3 & 3 & -5 \end{bmatrix},$$

$$H_1^{\mathrm{K}} = \begin{bmatrix} -5 & -5 & 3 \\ -5 & 0 & 3 \\ 3 & 3 & 3 \end{bmatrix}, \qquad H_5^{\mathrm{K}} = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & -5 \\ 3 & -5 & -5 \end{bmatrix},$$

$$H_2^{\mathrm{K}} = \begin{bmatrix} -5 & -5 & -5 \\ 3 & 0 & 3 \\ 3 & 3 & 3 \end{bmatrix}, \qquad H_6^{\mathrm{K}} = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 0 & 3 \\ -5 & -5 & -5 \end{bmatrix},$$

$$H_3^{\mathrm{K}} = \begin{bmatrix} 3 & -5 & -5 \\ 3 & 0 & -5 \\ 3 & 3 & 3 \end{bmatrix}, \qquad H_7^{\mathrm{K}} = \begin{bmatrix} 3 & 3 & 3 \\ -5 & 0 & 3 \\ -5 & -5 & 3 \end{bmatrix}.$$

[Burger & Burge]

Edge detectors with “combined” features

- Some detectors combine different features in a single kernel to deal with several issues:
  - Edge orientation and contrast
  - Gradient orientation and magnitude
  - Noise
  - Scale

- Two possible approaches:
  - Marr-Hildreth
  - Canny

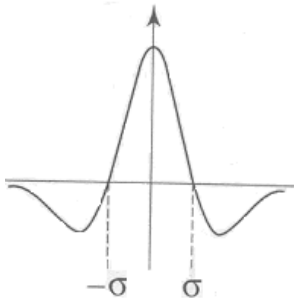
Marr-Hildreth detector (1980)

The kernel combines:

- a smoothing Gaussian function

- a Laplacian derivative operator (2nd order)

Section of the continuous function
(with circular symmetry)



0	0	0	−1	−1	−2	−1	−1	0	0	0
0	0	−2	−4	−8	−9	−8	−4	−2	0	0
0	−2	−7	−15	−22	−23	−22	−15	−7	−2	0
−1	−4	−15	−24	−14	−1	−14	−24	−15	−4	−1
−1	−8	−22	−14	52	103	52	−14	−22	−8	−1
−2	−9	−23	−1	103	178	103	−1	−23	−9	−2
−1	−8	−22	−14	52	103	52	−14	−22	−8	−1
−1	−4	−15	−24	−14	−1	−14	−24	−15	−4	−1
0	−2	−7	−15	−22	−23	−22	−15	−7	−2	0
0	0	−2	−4	−8	−9	−8	−4	−2	0	0
0	0	0	−1	−1	−2	−1	−1	0	0	0

Digital implementation of the
Marr-Hildreth detector

- The kernel is larger than the previous ones (precision)

- Standard deviation σ of Gaussian controls its width

- The larger σ, the larger is the smoothing

- After smoothing, zero-crossings are detected

- The value of σ can be used to extract edge details at different scales

After filtering with:

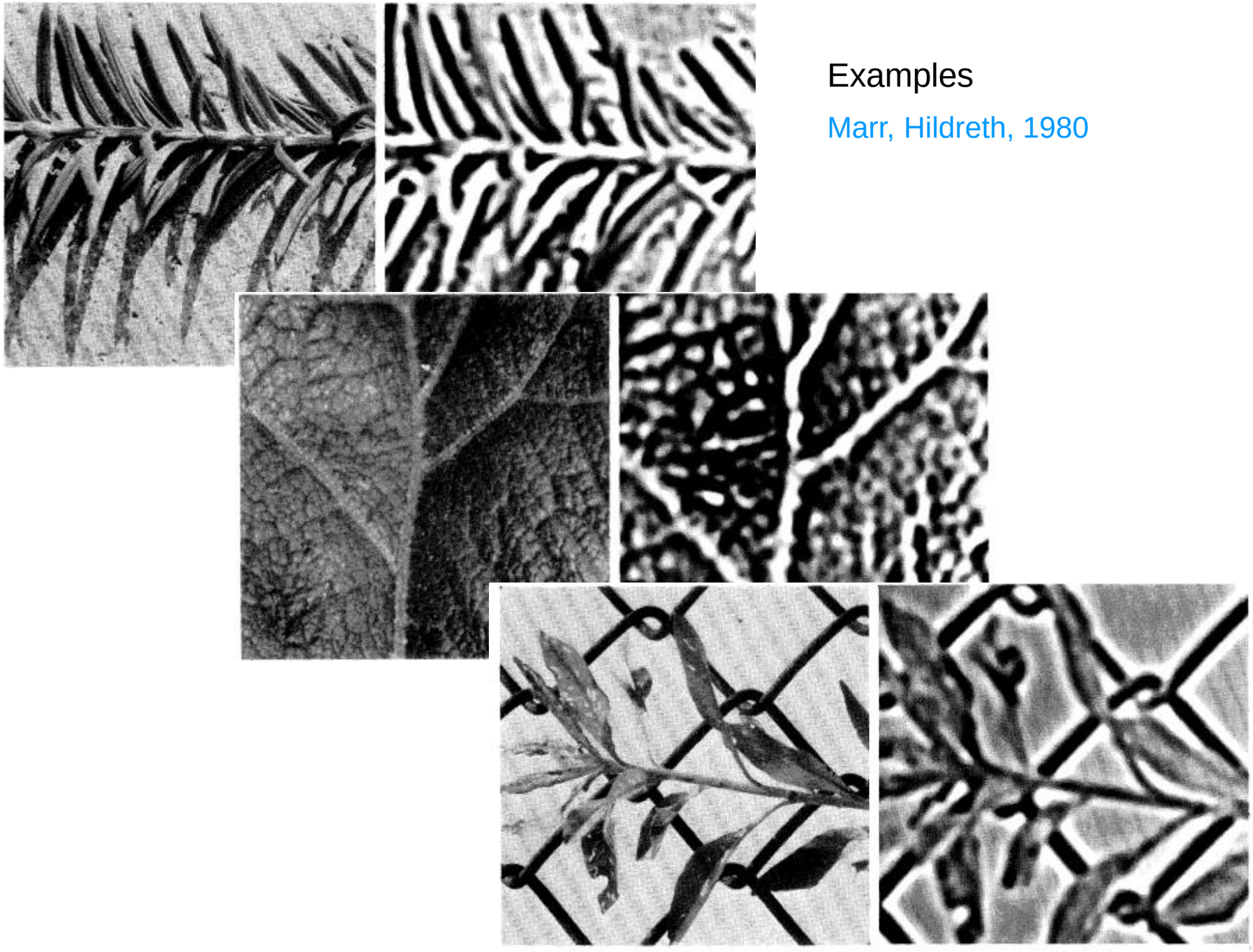
σ=1,5

σ=5

Zero-crossings of the filtered images

Zero-crossings common to both images

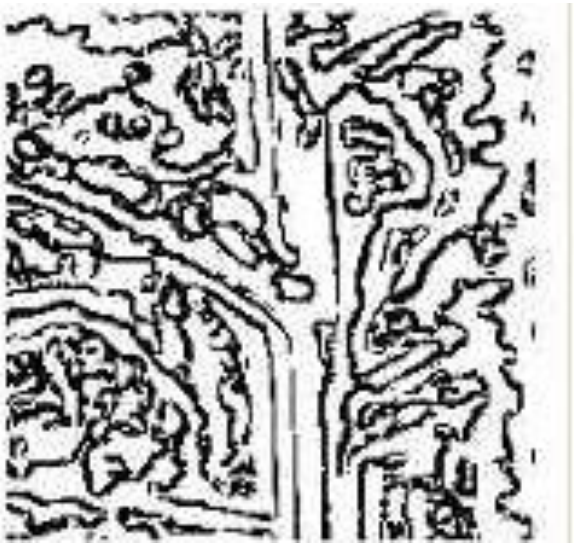
$$M_H = \left(\frac{r^2 - \sigma^2}{\sigma^4}\right) exp\left(\frac{-r^2}{2\sigma^2}\right)$$



Examples

Marr, Hildreth, 1980

Marr- Hildreth – Application to a color image



Detector de Canny (1986)

- Multi-step method, that allows detecting edges and attenuating noise

- Sometimes characterized as the *optimal edge detector* (optimal for well-defined edges with white noise)

- As the Marr-Hildreth detector, it also combines a *smoothing Gaussian* with a *Laplacian*, but it uses directional information from the gradient

- It detects edges through the zero-crossings of the image's 2nd derivative, smoothed with a 1D Gaussian filter in the direction of the gradient

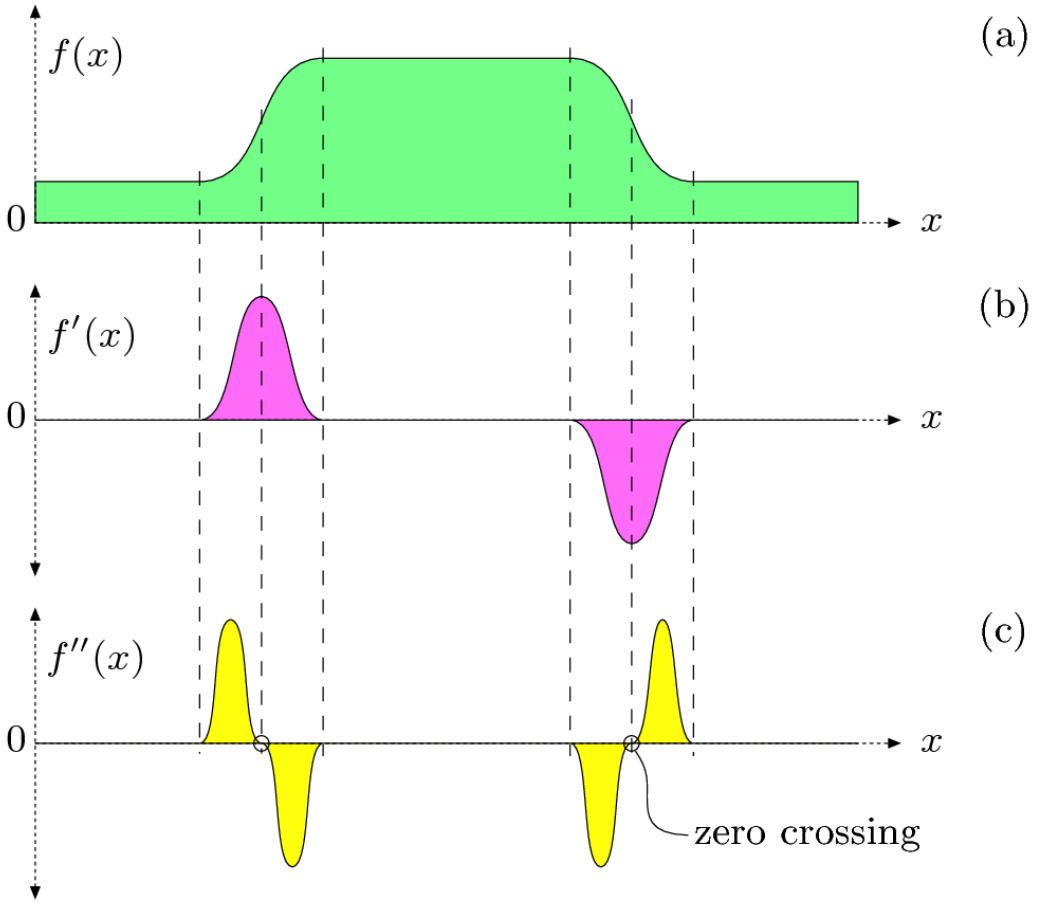
http://www.pages.drexel.edu/~weg22/can_tut.html

http://www.icaen.uiowa.edu/~dip/LECTURE/PreProcessing3.html#canny

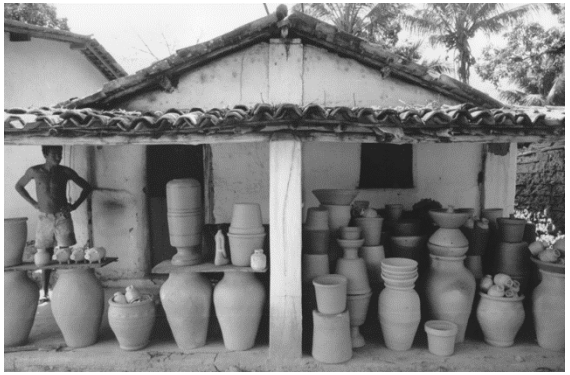
Canny Edge Detector

Goals:

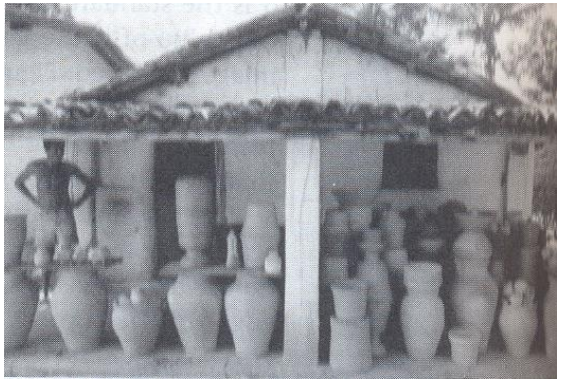
- Minimize “false” edges
- Better edge localization



[Burger & Burge]



Original image

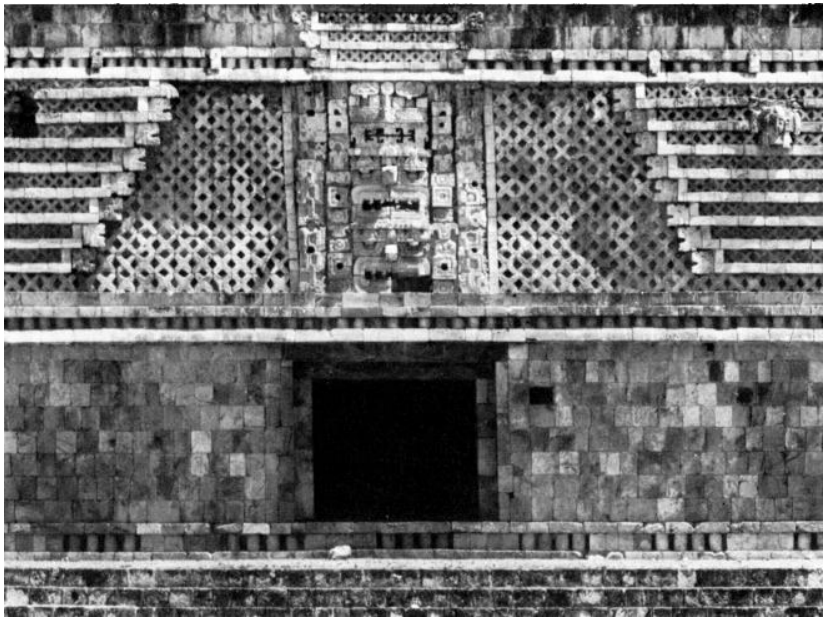


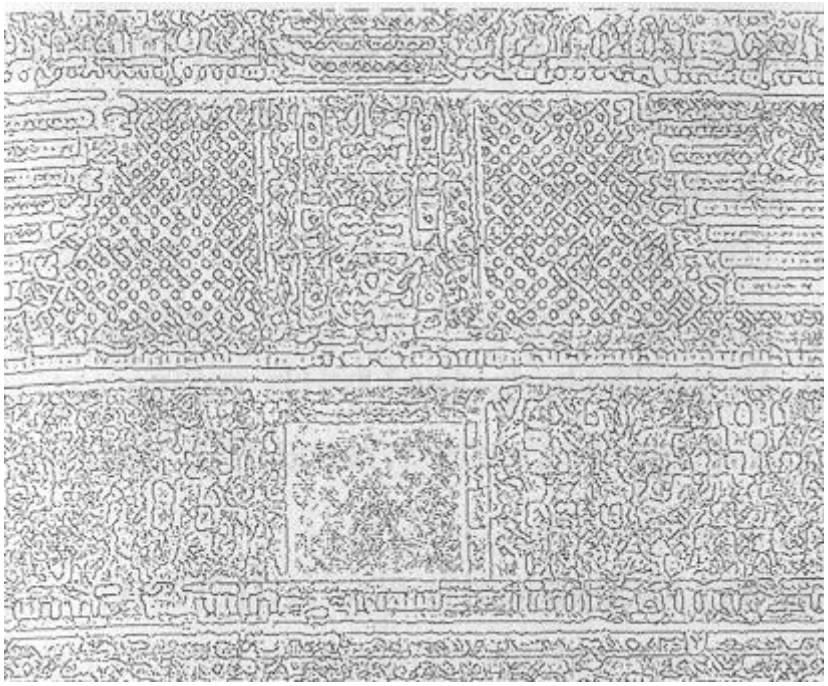
x and y components of the 1D smoothing with a 1D Gaussian σ=6

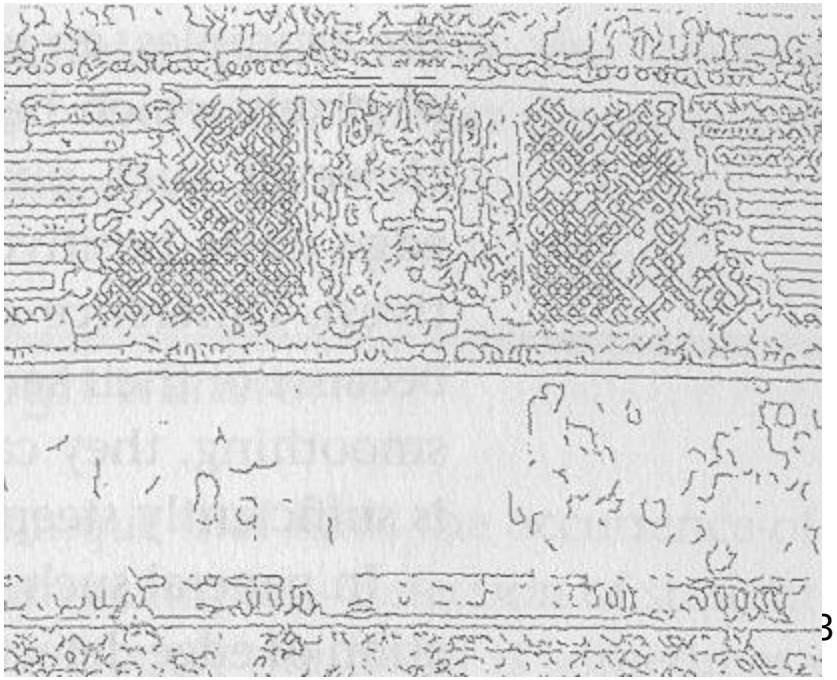
x and y components of the derivative

Final image

Example: results of the Canny edge detector, σ= 1,5 and σ=5

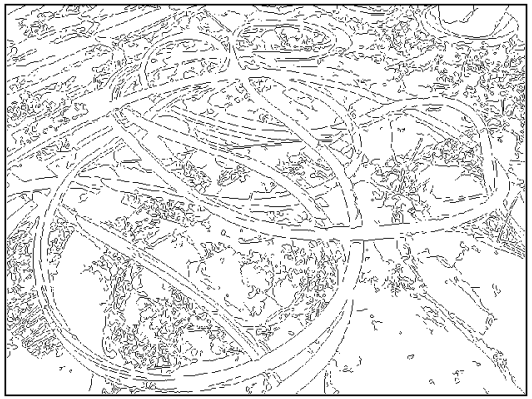




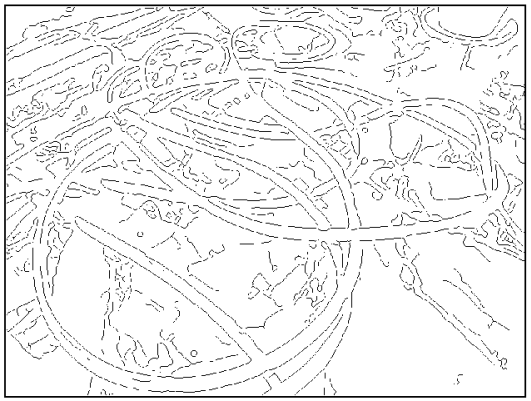


Another example

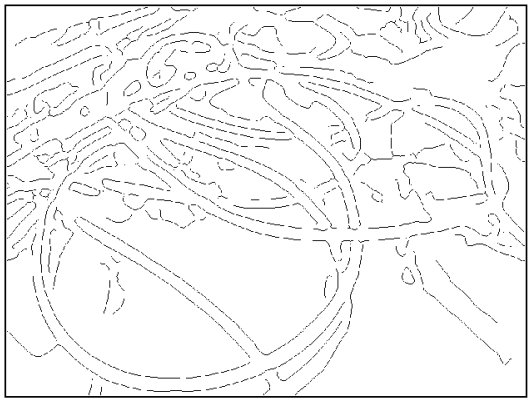
Original



$\sigma = 1.0$

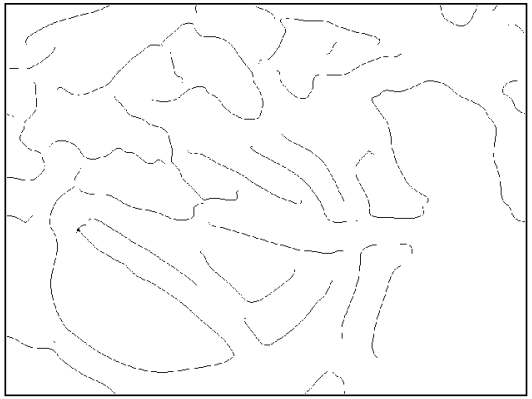


$\sigma = 2.0$



$\sigma = 4.0$

$\sigma = 8.0$



$\sigma = 16.0$

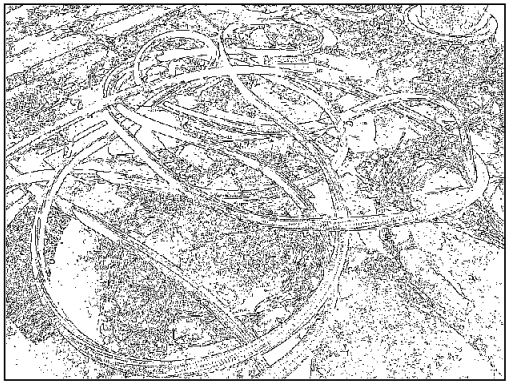
[Burger & Burge]

Comparison

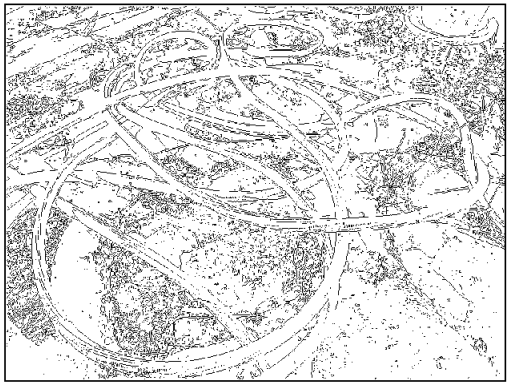
Criteria:

- Amount of “false” edges / segments
- Connectedness

Original



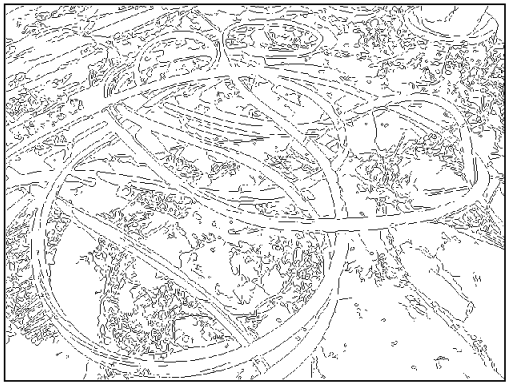
Roberts



Prewitt

Sobel

Laplacian of Gaussian



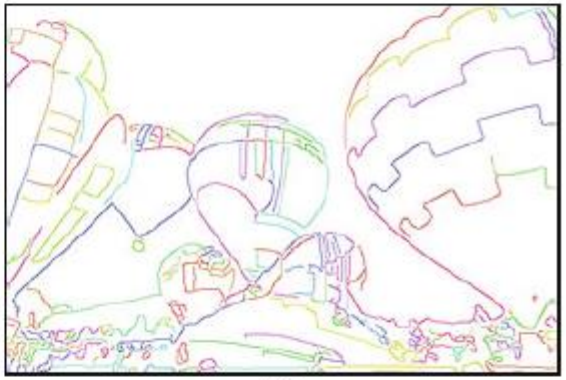
Canny ($\sigma = 1.0$)

[Burger & Burge]

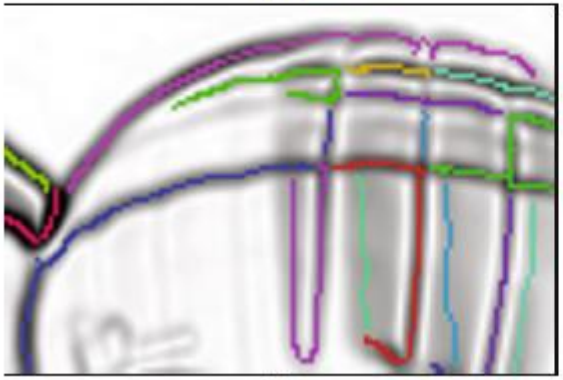
Another example

(a)

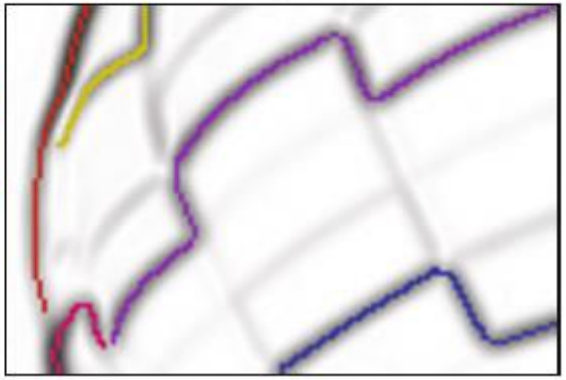
(b)



(c)



(d)



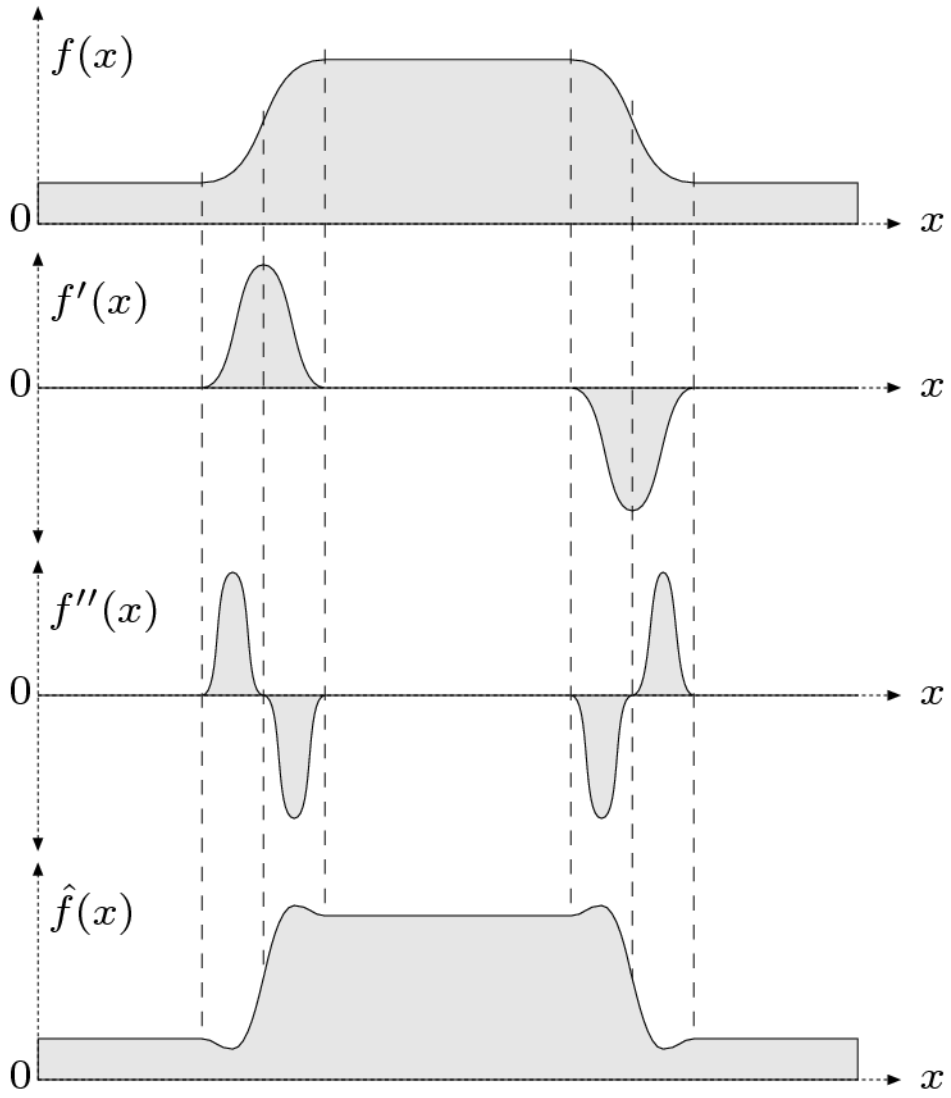
(e)

[Burger & Burge]

- The previous operators are known as edge detectors
- The derivative operators are noise sensitive, generating false positives
- They transform an image into a binary domain:
  - a set of edge pixels – *edgels*
  - and false positives and false negatives
- A better characterization: *edge emphasis*

Edge sharpening

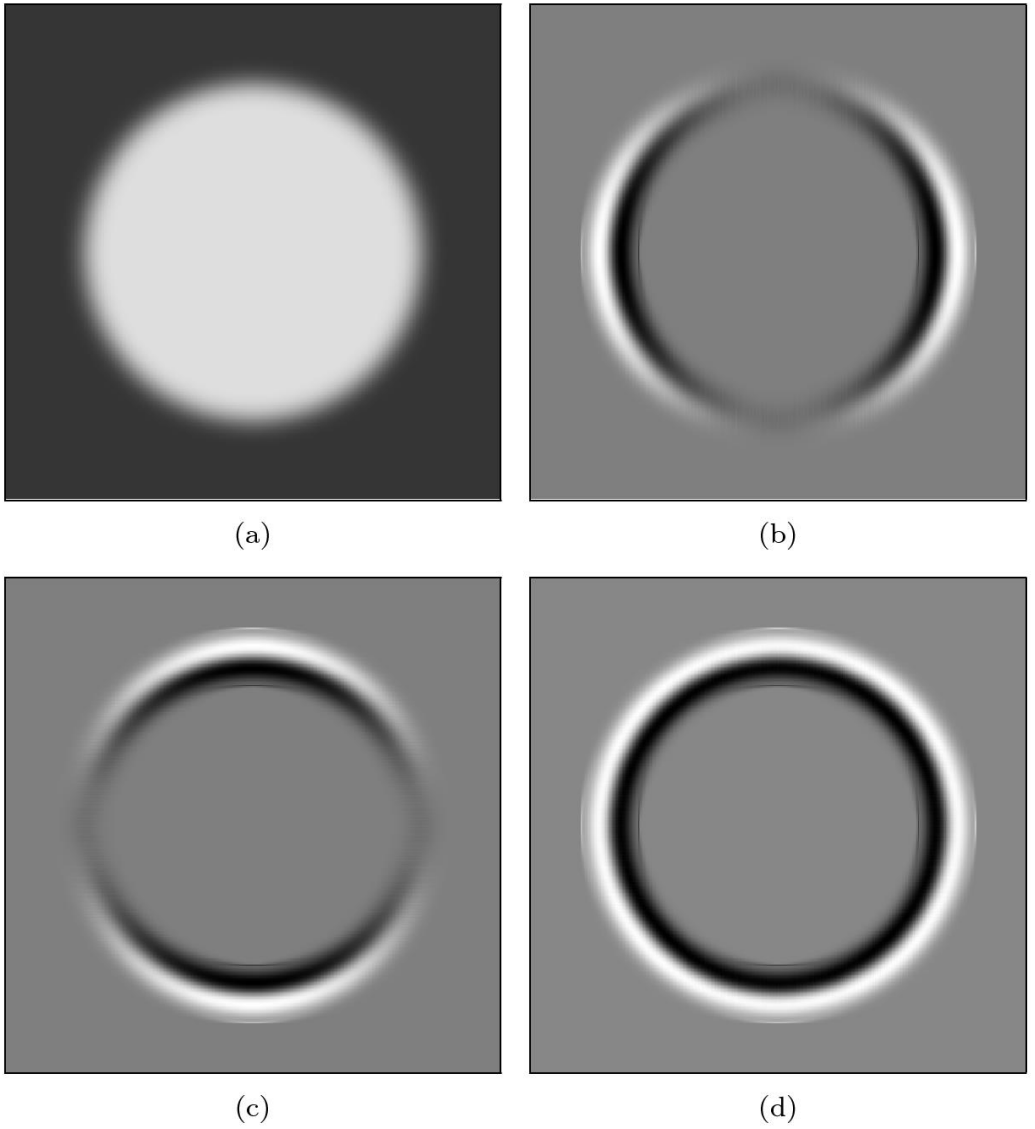
$$\hat{f}(x) = f(x) - w \cdot f''(x)$$



[Burger & Burge]

Edge sharpening – Laplace operator

$$H^L = H_x^L + H_y^L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$



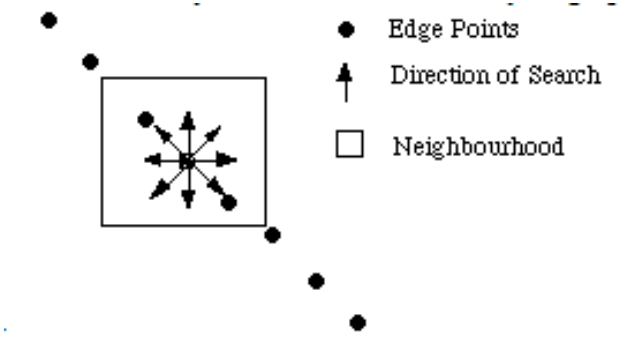
(a) (b) (c) (d)

[Burger & Burge]

Edge linking

- Edgel aggregation can be done sequentially
- A contour is followed: at each pixel a decision is taken on how to pursue
- An alternative is to perform aggregation after edge detection, attempting to:
  - Reduce the effect of spurious pixels (artefacts)
  - Compose continuous edges from fragments

Local contour following



- Expand the neighborhood to detect an edge
- Neighborhood must be:
  - Large enough to close false *gaps*
  - Small enough not to link independent fragments

- Example: on a simple approach over the gradient image, two pixels (x,y) and (x',y') belong to the same edge if:

$$|\nabla I(x,y) - \nabla I(x',y')| < t_1$$

and

$$|\theta(I(x,y)) - \theta(I(x',y'))| < t_2$$

θ - gradient direction
t_1 and t_2 - *thresholds*

IMAGE SEGMENTATION

Image Segmentation

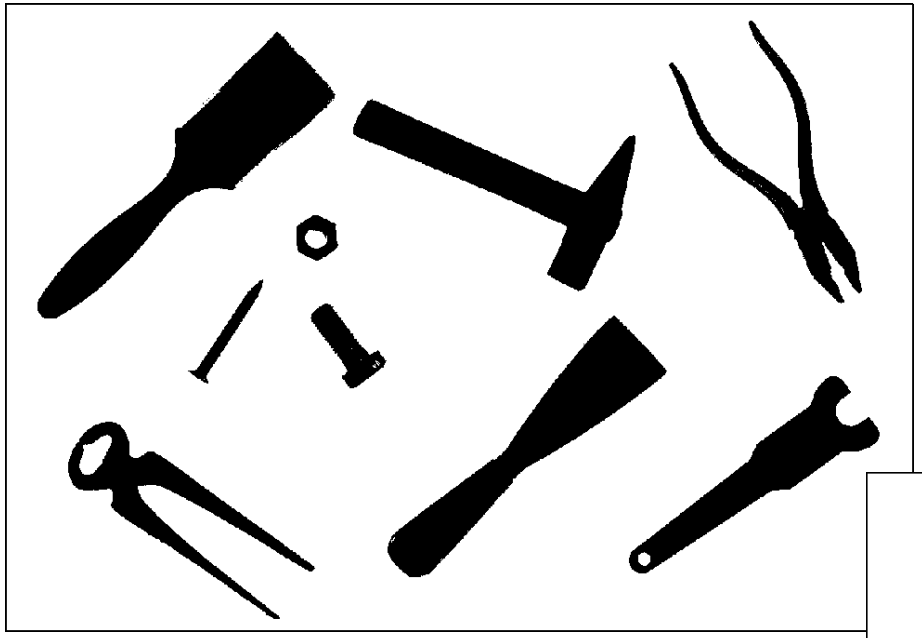
- Subdivide an image into regions that are meaningful in the application context
- In general, it is a previous step to image analysis
- It can be regarded as an image analysis step

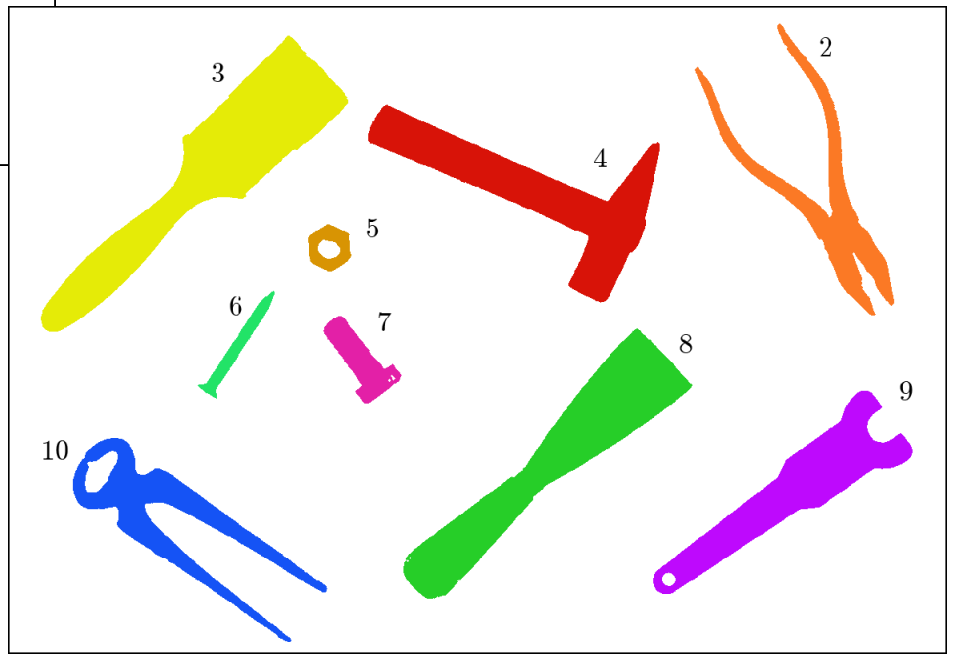
Image Processing: *image in; image out*

Image Analysis: data reduction; knowledge extraction

- Group pixels with similar features
  - In principle, are associated with the same entity

Example

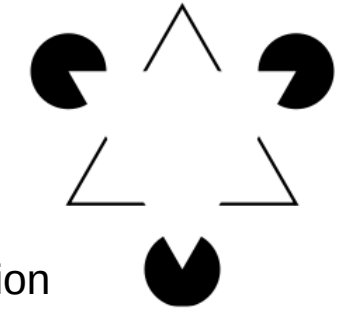




[Burger & Burge]

- Humans can easily segment an image: parallel processing + a priori scene knowledge

- They even segment non-existing objects !



Kanizsa illusion

- Simplest computational approaches process one pixel at a time, and do not use neighborhood information

- Which is the usual strategy:

  subdivide a problem into simpler problems,
  for which we know resolution methods

- For image segmentation, it is reasonable to start with *low-level* pixel by pixel operations

Note:

- No segmentation method can be successfully applied to every case
- There is no perfect segmentation method

Common segmentation approaches:

Based on:

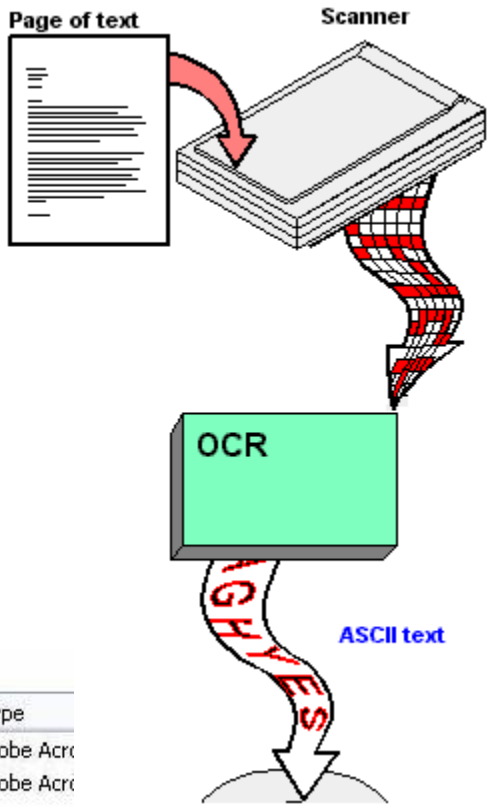
- Global knowledge on pixel intensities
- Regions
- Edges / contours

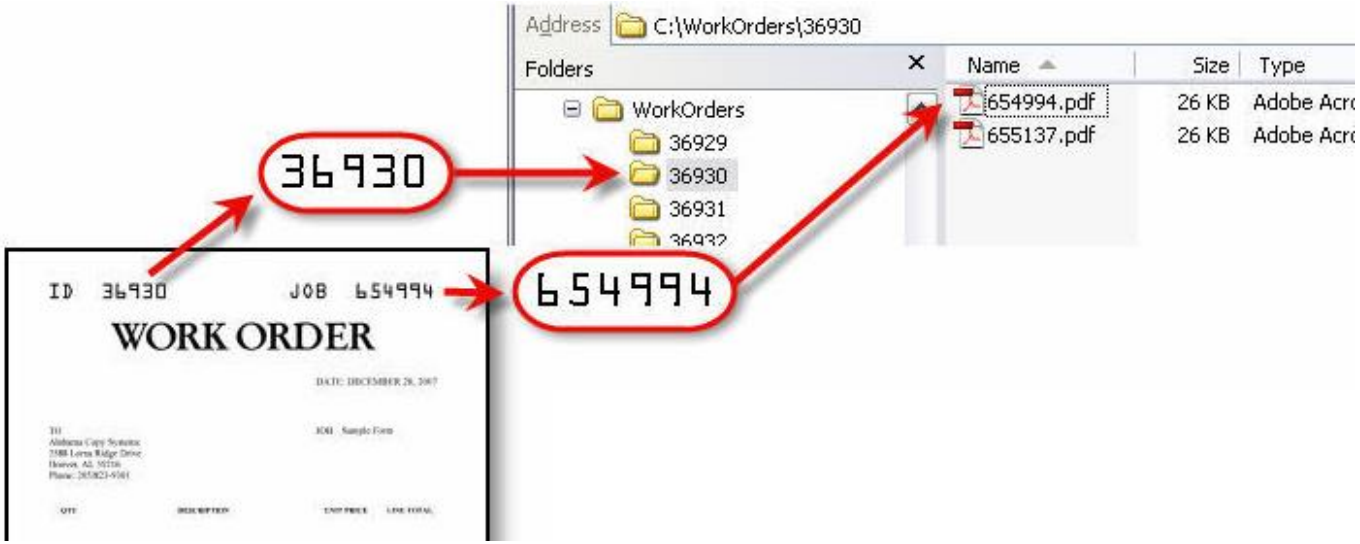
SEGMENTATION BY THRESHOLDING

Gray scale thresholding

- Oldest segmentation approach
- Appropriate whenever the intensity of the objects of interest is homogeneous and they are different from the background

  Example: OCR (*Optical Character Recognition*)

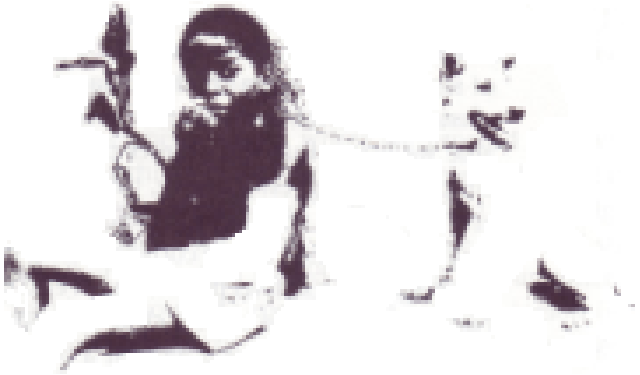


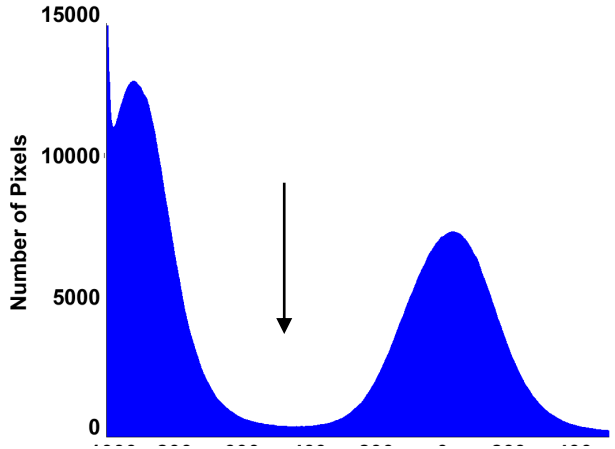


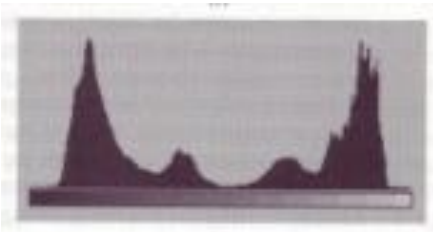
Gray scale thresholding

- If the threshold value is unknown, analyze the histogram to choose an adequate threshold value

- For a bimodal histogram, the threshold value corresponds to the valley between the peaks

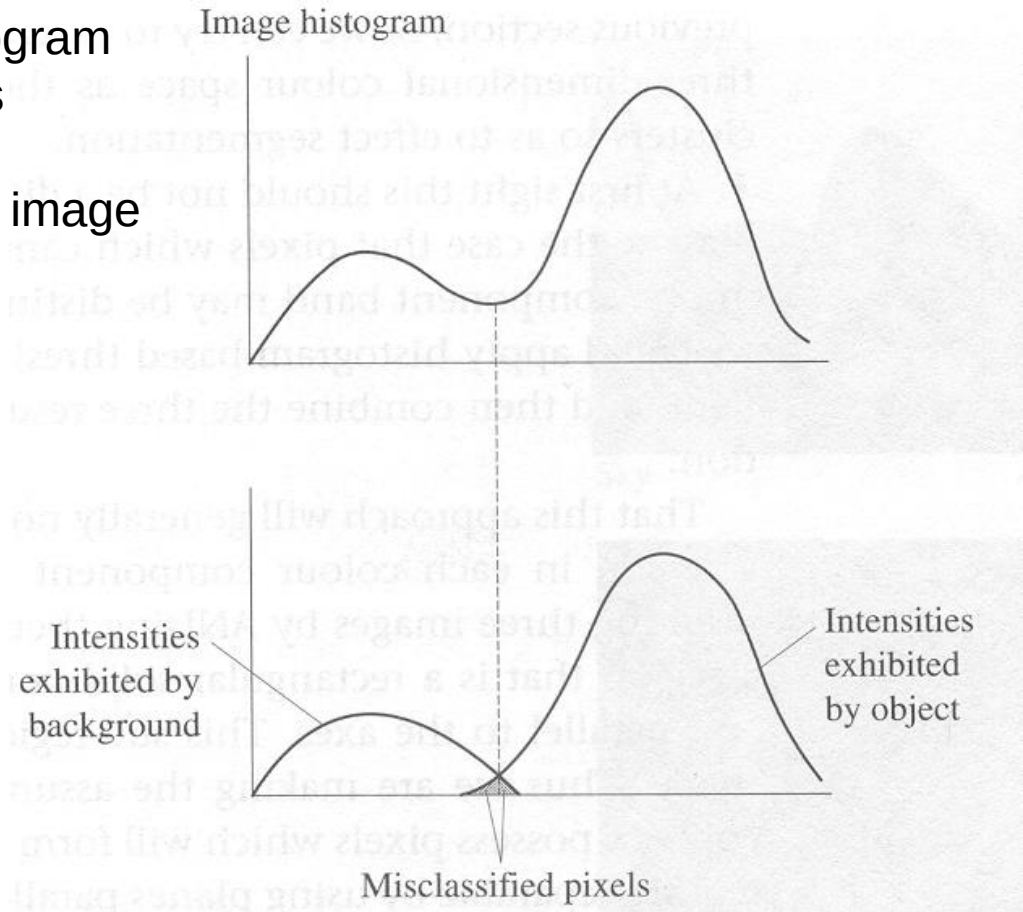






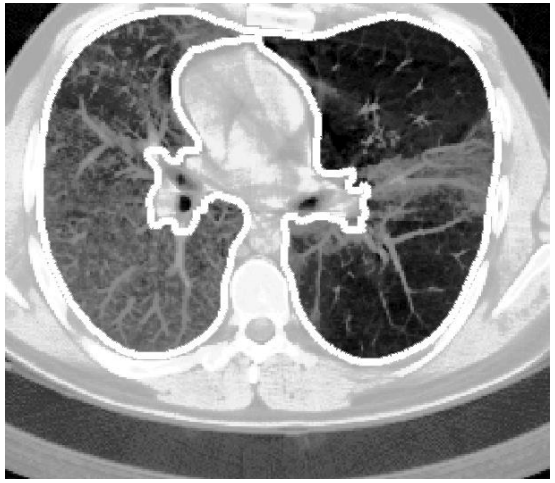
- This approach can produce "classification errors", depending on the image histogram and the intensity values of the objects

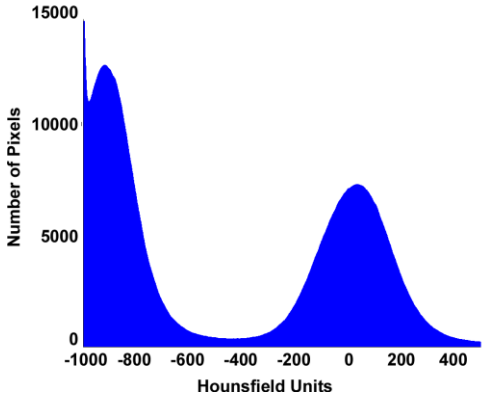
- The *thresholding* can be applied to image sub-regions



Example – Processing chest CT images and segmentation

- Detecting contours of the lungs and of lung air bubbles on chest CT images
- Different gray levels correspond to different tissues and air in each CT image
- Determine the histogram of each image and the intensity values corresponding to the transition between the lungs and other tissues
- Identify the lung contour pixels in each image
- "Enhance" the contours

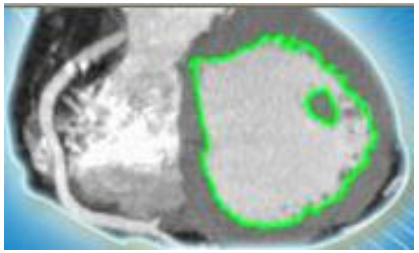




CT image histogram

SEGMENTATION BY REGION GROWING

Segmentation by Region-Growing



- The simplest method for region growing is pixel aggregation
- Need for “seed-pixels”
- And for aggregation rules based on :
  - connectivity
  - similarity of common features
- All pixels that are neighbors to the seed pixel and have similar features, are labeled as belonging to the same region
- The placement of the initial seed pixels is an important issue
- For instance, that can be done by the user or as a result of histogram analysis

Labels – Neighborhoods

$$I(u,v) = \begin{cases} 0 & \text{Hintergrundpixel } (\textit{background}) \\ 1 & \text{Vordergrundpixel } (\textit{foreground}) \\ 2, 3, \ldots & \text{Regionenmarkierung } (\textit{label}) \end{cases}$$

$$\mathcal{N}_4(u,v) = \begin{array}{|c|c|c|} \hline & N_2 & \\ \hline N_1 & \times & \\ \hline & & \\ \hline \end{array} \qquad \mathcal{N}_8(u,v) = \begin{array}{|c|c|c|} \hline N_2 & N_3 & N_4 \\ \hline N_1 & \times & \\ \hline & & \\ \hline \end{array}$$

[Burger & Burge]

Flood-Filling

```
1:  RegionLabeling(I)
        I: binary image (0 = background, 1 = foreground)
        The image I is labeled (destructively modified) and returned.
2:      Initialize m ← 2 (the value of the next label to be assigned).
3:      Iterate over all image coordinates (u, v).
4:          if I(u, v) = 1 then
5:              FloodFill(I, u, v, m)        ▷ use any of the 3 versions below
6:              m ← m + 1.
7:      return the labeled image I.
```

```
8:  FloodFill(I, u, v, label)                ▷ Recursive Version
9:      if coordinate (u, v) is within image boundaries and I(u, v) = 1 then
10:         Set I(u, v) ← label
11:         FloodFill(I, u+1, v, label)
12:         FloodFill(I, u, v+1, label)
13:         FloodFill(I, u, v−1, label)
14:         FloodFill(I, u−1, v, label)
15:     return.
```

[Burger & Burge]

Flood-Filling

```
16: FloodFill(I, u, v, label)                    ▷ Depth-First Version
17:     Create an empty stack S
18:     Put the seed coordinate ⟨u, v⟩ onto the stack: Push(S, ⟨u, v⟩)
19:     while S is not empty do
20:         Get the next coordinate from the top of the stack:
                ⟨x, y⟩ ← Pop(S)
21:         if coordinate (x, y) is within image boundaries and I(x, y) = 1
                then
22:             Set I(x, y) ← label
23:             Push(S, ⟨x+1, y⟩)
24:             Push(S, ⟨x, y+1⟩)
25:             Push(S, ⟨x, y−1⟩)
26:             Push(S, ⟨x−1, y⟩)
27:     return.
```

[Burger & Burge]

Flood-Filling

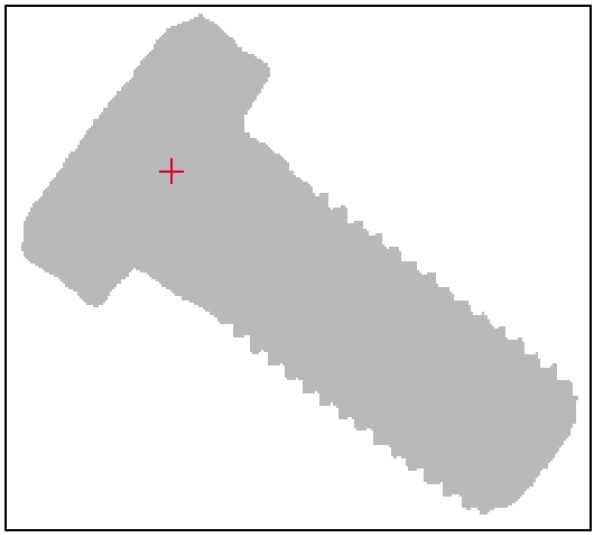
```
28:  FloodFill(I, u, v, label)                    ▷ Breadth-First Version
29:      Create an empty queue Q
30:      Insert the seed coordinate ⟨u, v⟩ into the queue: Enqueue(Q, ⟨u, v⟩)
31:      while Q is not empty do
32:          Get the next coordinate from the front of the queue:
                 ⟨x, y⟩ ← Dequeue(Q)
33:          if coordinate ⟨x, y⟩ is within image boundaries and I(x, y) = 1
                 then
34:              Set I(x, y) ← label
35:              Enqueue(Q, ⟨x+1, y⟩)
36:              Enqueue(Q, ⟨x, y+1⟩)
37:              Enqueue(Q, ⟨x, y−1⟩)
38:              Enqueue(Q, ⟨x−1, y⟩)
39:      return.
```

[Burger & Burge]

Flood-Filling

(a)

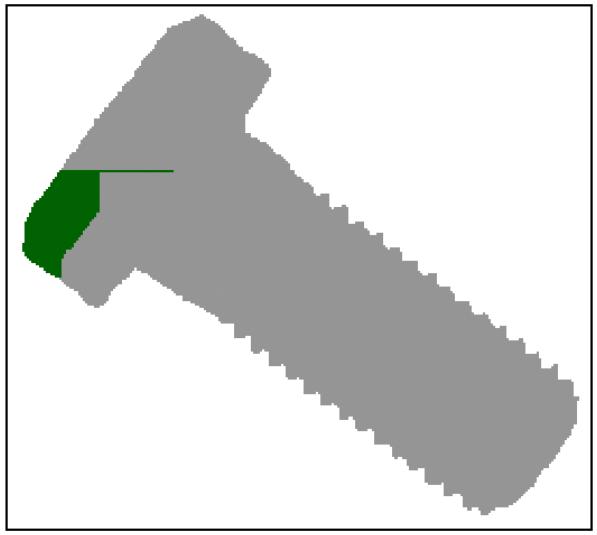
Original

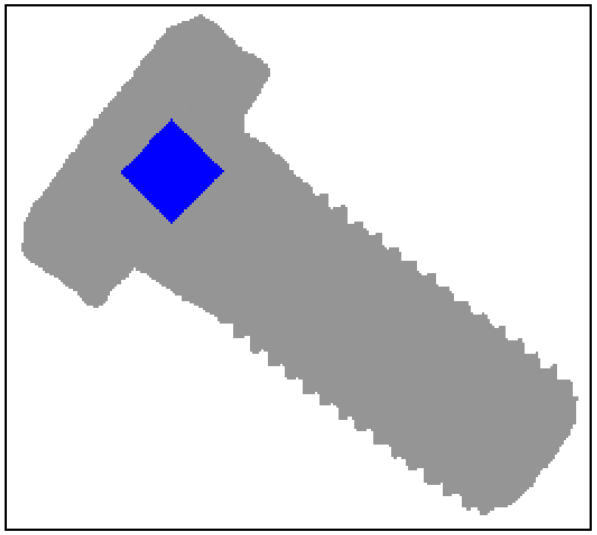


depth-first **breadth-first**

(b)

$K = 1.000$



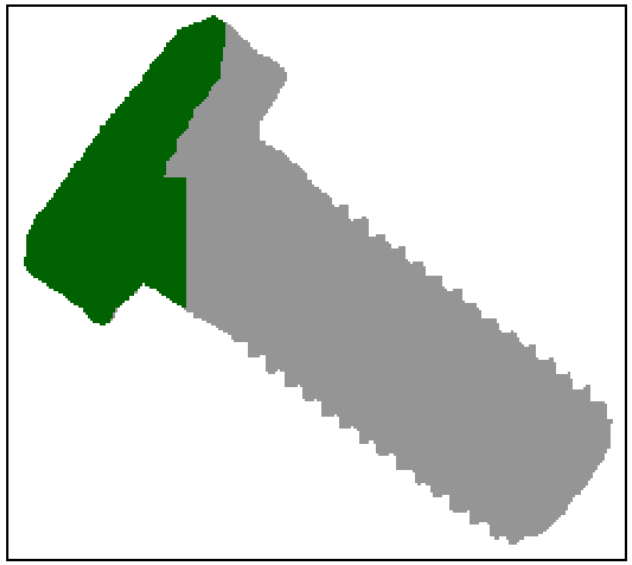


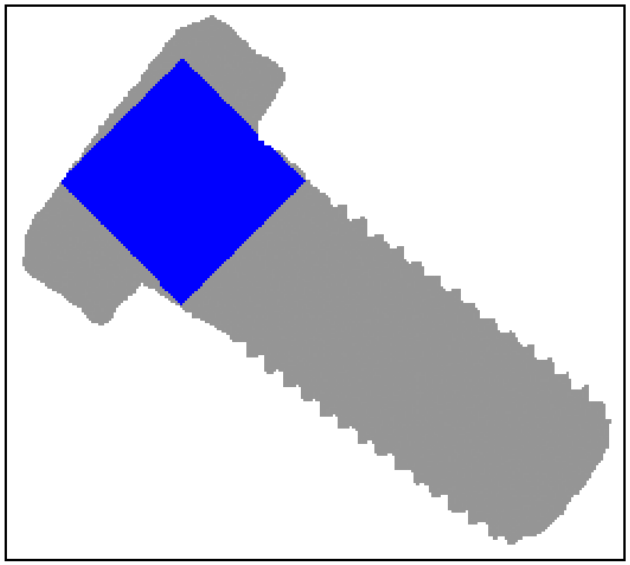
[Burger & Burge]

Flood-Filling

(c)

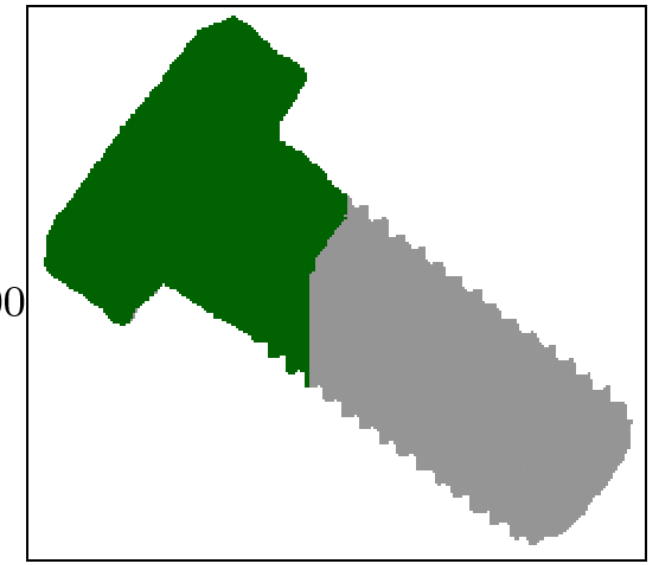
$K = 5.000$

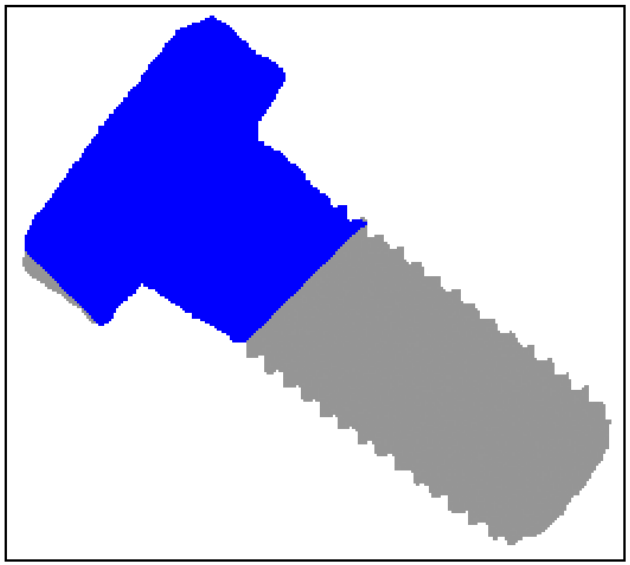




(d)

$K = 10.000$





[Burger & Burge]

Example: similarity based on a *threshold* value and the gradient

Threshold is too high

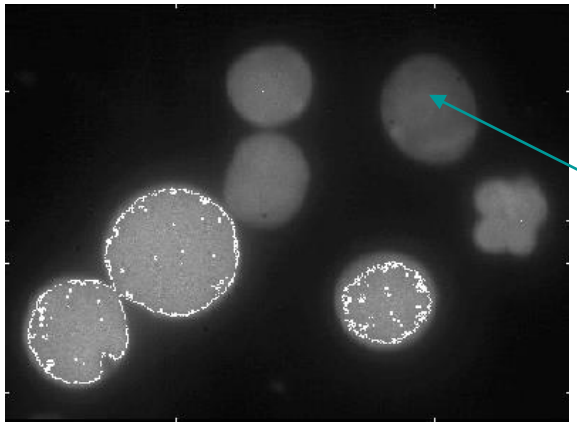
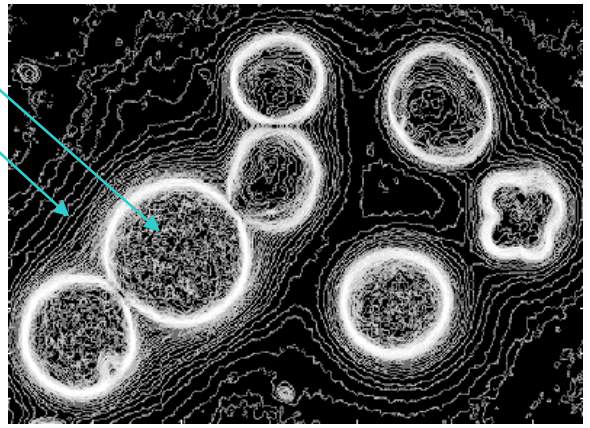


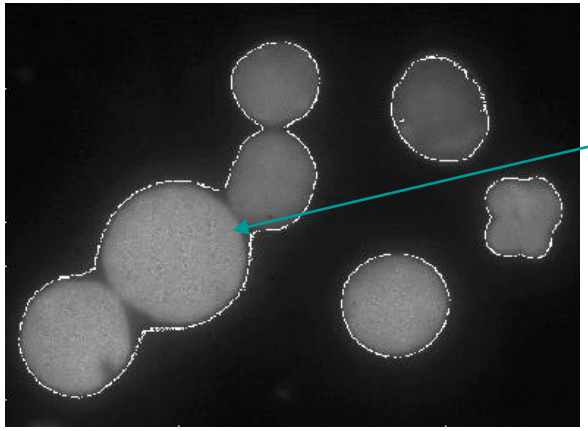
Image has noise and texture

Objects that are not detected

Image gradient

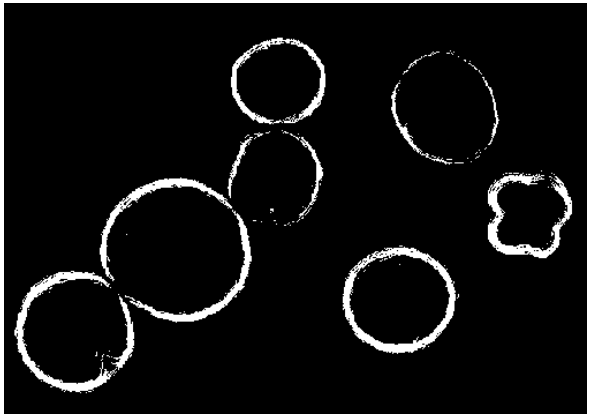


Threshold is too low



Confusion between objects

Gradient of the smoothed image



http://www.owlnet.rice.edu/~elec539/Projects97/WDEKnow/criteria.html

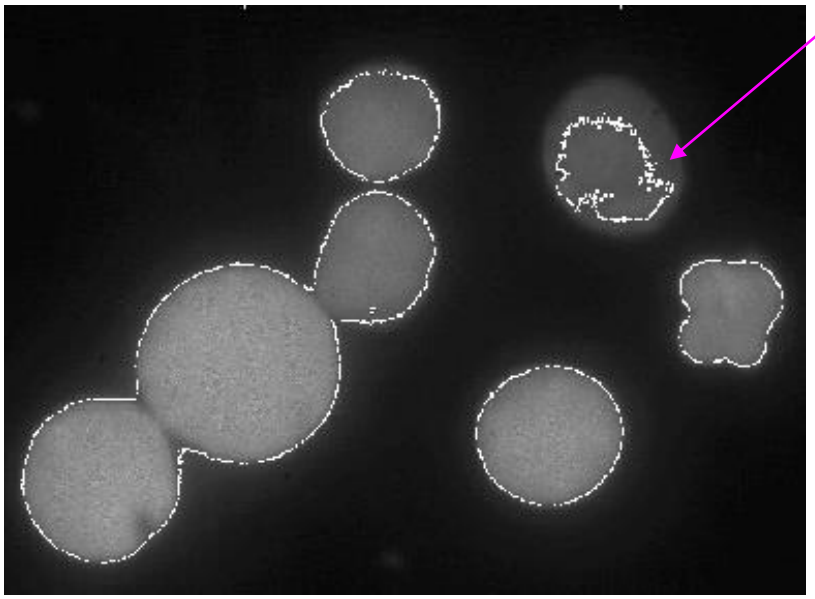
Criteria used:

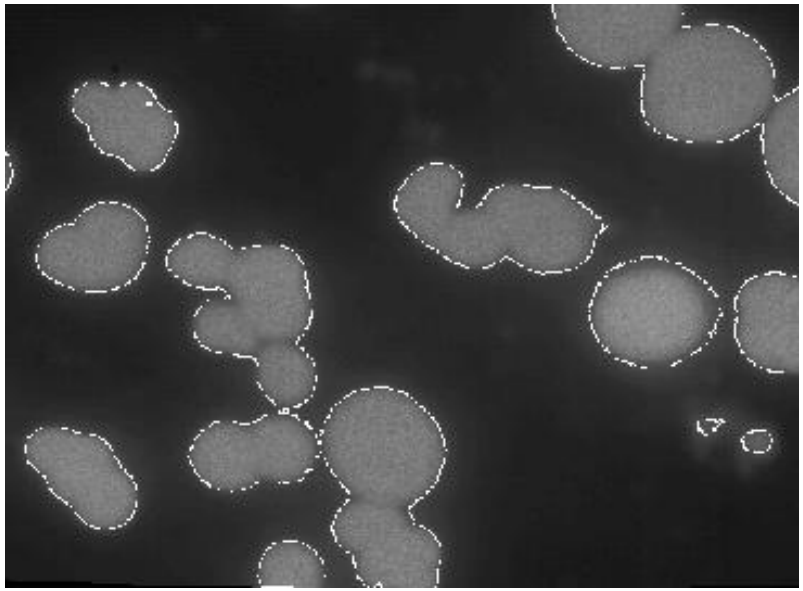
A pixel is added to a region if it satisfies one of the following criteria:

1 - low gradient
and
gray level > *threshold*

2 - high gradient
and
gray level < region average +- region standard deviation

Similarity criteria have to be improved





SEGMENTATION BY REGION LABELING

Sequential Region Labeling

SequentialLabeling(I)

I: binary image (0 = *background*, 1 = *foreground*)
The image I is labeled (destructively modified) and returned.

Pass 1—Assign Initial Labels:

Initialize $m \leftarrow 2$ (the value of the next label to be assigned).
Create an empty set $\mathcal{C}$ to hold the collisions: $\mathcal{C} \leftarrow \{\}$.
for $v \leftarrow 0 \ldots H-1$ **do** ▷ H = height of image I
  for $u \leftarrow 0 \ldots W-1$ **do** ▷ W = width of image I
    if $I(u,v) = 1$ **then** do one of:
      if all neighbors of (u,v) are background pixels (all $n_i = 0$) **then**
        $I(u,v) \leftarrow m$.
        $m \leftarrow m + 1$.
      else if exactly *one* of the neighbors has a label value $n_k > 1$ **then**
        set $I(u,v) \leftarrow n_k$
      else if *several* neighbors of (u,v) have label values $n_j > 1$ **then**
        Select one of them as the new label:
          $I(u,v) \leftarrow k \in \{n_j\}$.
        for all other neighbors of u,v) with label values $n_i > 1$ and $n_i \neq k$ **do**
          Create a new label collision $\boldsymbol{c}_i = \langle n_i, k \rangle$.
          Record the collision: $\mathcal{C} \leftarrow \mathcal{C} \cup \{\boldsymbol{c}_i\}$.

Remark: The image I now contains label values $0, 2, \ldots m-1$.

[Burger & Burge]

Sequential Region Labeling

PASS 2—RESOLVE LABEL COLLISIONS:

17: Let $\mathcal{L} = \{2, 3, \ldots m-1\}$ be the set of preliminary region labels.

18: Create a partitioning of $\mathcal{L}$ as a *vector of sets*, one set for each label value: $\mathcal{R} \leftarrow [\mathcal{R}_2, \mathcal{R}_3, \ldots, \mathcal{R}_{m-1}] = [\{2\}, \{3\}, \{4\}, \ldots, \{m-1\}]$, so $\mathcal{R}_i = \{i\}$ for all $i \in \mathcal{L}$.

19: **for all** collisions $\langle a, b \rangle \in \mathcal{C}$ **do**

20: Find in $\mathcal{R}$ the sets $\mathcal{R}_a$, $\mathcal{R}_b$ containing the labels a, b, resp.:
$\mathcal{R}_a \leftarrow$ the set that currently contains label a
$\mathcal{R}_b \leftarrow$ the set that currently contains label b

21: **if** $\mathcal{R}_a \neq \mathcal{R}_b$ (a and b are contained in different sets) **then**

22: Merge sets $\mathcal{R}_a$ and $\mathcal{R}_b$ by moving all elements of $\mathcal{R}_b$ to $\mathcal{R}_a$:
$\mathcal{R}_a \leftarrow \mathcal{R}_a \cup \mathcal{R}_b$
$\mathcal{R}_b \leftarrow \{\}$

Remark: All *equivalent* label values (i. e., all labels of pixels in the same region) are now contained in the same set $\mathcal{R}_i$ within $\mathcal{R}$.

PASS 3—RELABEL THE IMAGE:

23: Iterate through all image pixels (u, v):

24: **if** $I(u, v) > 1$ **then**

25: Find the set $\mathcal{R}_i$ in $\mathcal{R}$ that contains label $I(u, v)$.

26: Choose one unique representative element k from the set $\mathcal{R}_i$ (e. g., the minimum value, $k = \min(\mathcal{R}_i)$).

27: Replace the image label: $I(u, v) \leftarrow k$.

28: **return** the labeled image I.

[Burger & Burge]

Sequential Region Labeling

(a)

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

0 Background

1 Foreground

(b) only background neighbors

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

new label (**2**)

0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	2	1	0	0	1	1	0	1	0
0	1	1	1	1	1	1	0	0	1	0	0	1	0
0	0	0	0	1	0	1	0	0	0	0	0	1	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	0
0	1	1	0	0	0	1	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

[Burger & Burge]

Sequential Region Labeling

(c) exactly one neighbor label

0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	2	1	0	0	1	1	0	1	0
0	1	1	1	1	1	1	0	0	1	0	0	1	0
0	0	0	0	1	0	1	0	0	0	0	0	1	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	0
0	1	1	0	0	0	1	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

neighbor label is propagated

0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	2	2	0	0	1	1	0	1	0
0	1	1	1	1	1	1	0	0	1	0	0	1	0
0	0	0	0	1	0	1	0	0	0	0	0	1	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	0
0	1	1	0	0	0	1	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

(d) two different neighbor labels

0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	2	2	0	0	3	3	0	4	0
0	5	5	5	1	1	1	0	0	1	0	0	1	0
0	0	0	0	1	0	1	0	0	0	0	0	1	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	0
0	1	1	0	0	0	1	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

one of the labels (**2**) is propagated

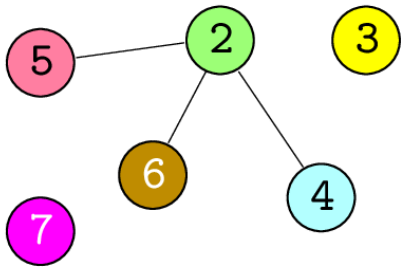
0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	2	2	0	0	3	3	0	4	0
0	5	5	5	2	1	1	0	0	1	0	0	1	0
0	0	0	0	1	0	1	0	0	0	0	0	1	0
0	1	1	1	1	1	1	1	1	1	1	1	1	0
0	0	0	0	1	1	1	1	1	1	1	1	1	0
0	1	1	0	0	0	1	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

[Burger & Burge]

Sequential Region Labeling

0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	2	2	0	0	3	3	0	4	0
0	5	5	5	2	2	2	0	0	3	0	0	4	0
0	0	0	0	2	0	2	0	0	0	0	0	4	0
0	6	6	2	2	2	2	2	2	2	2	2	2	0
0	0	0	0	2	2	2	2	2	2	2	2	2	0
0	7	7	0	0	0	2	0	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

(a)

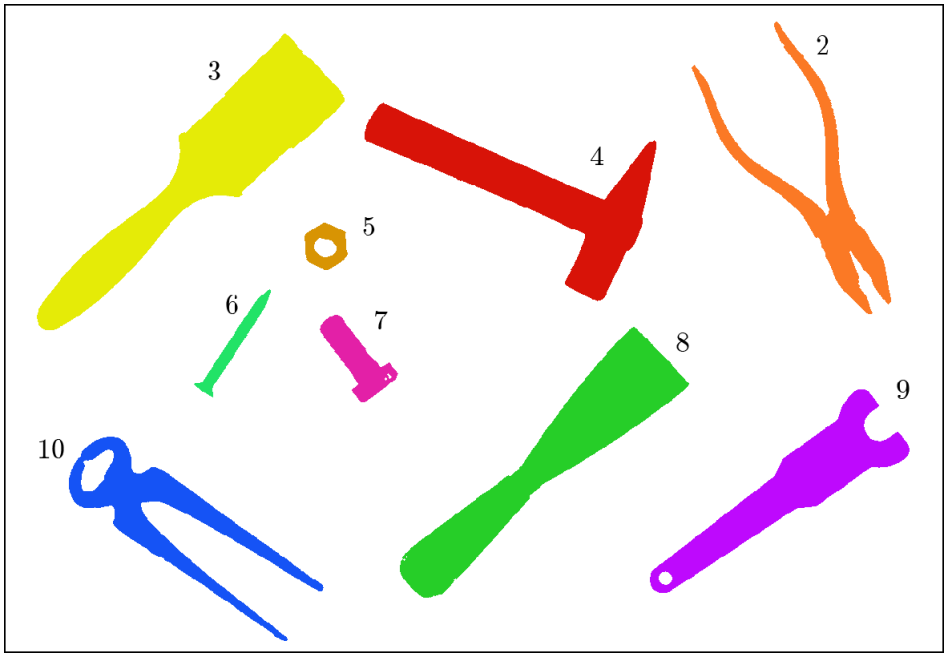


(b)

0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	2	2	0	0	3	3	0	2	0
0	2	2	2	2	2	2	0	0	3	0	0	2	0
0	0	0	0	2	0	2	0	0	0	0	0	2	0
0	2	2	2	2	2	2	2	2	2	2	2	2	0
0	0	0	0	2	2	2	2	2	2	2	2	2	0
0	7	7	0	0	0	2	0	2	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0

[Burger & Burge]

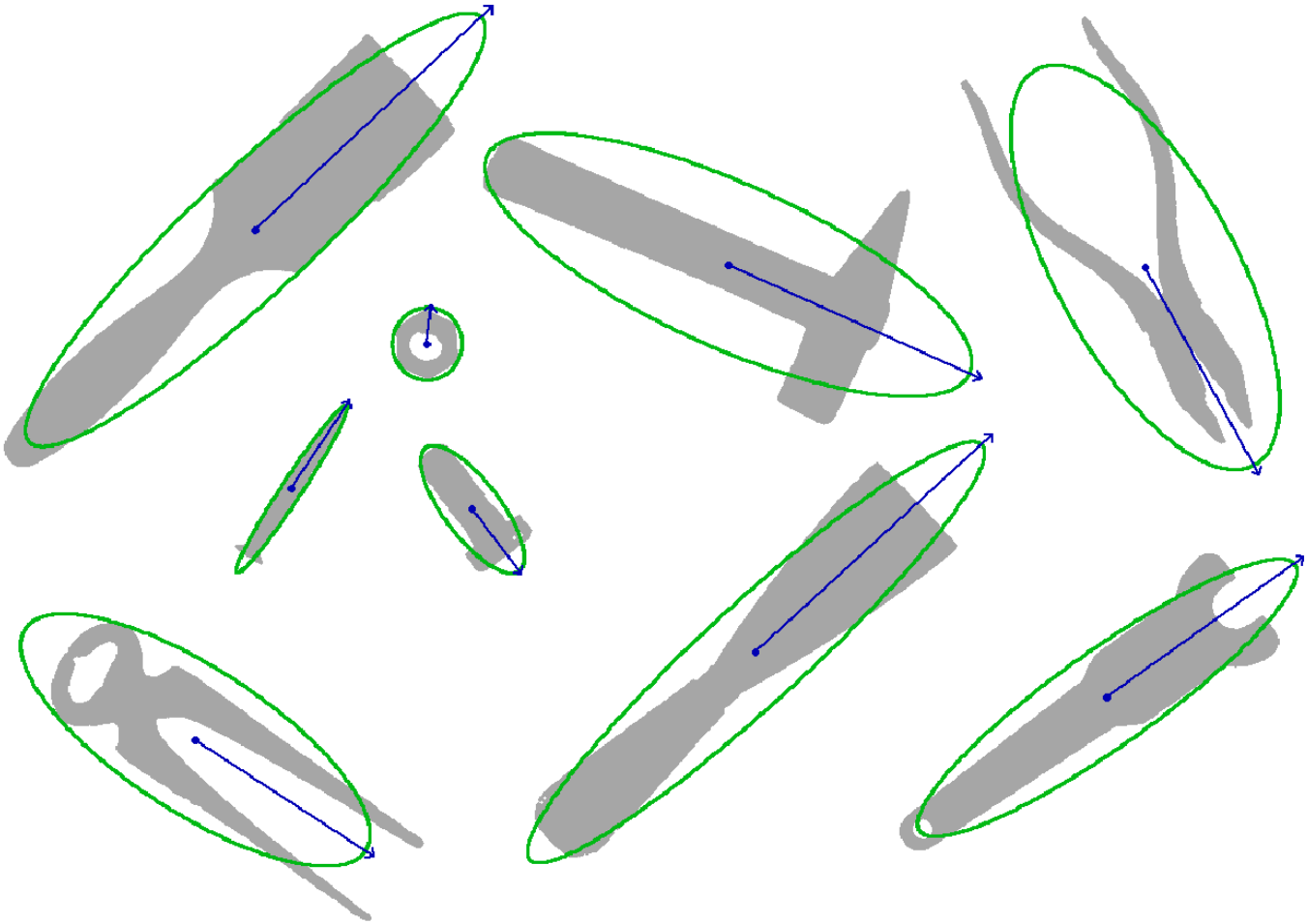
Sequential Region Labeling



Label	Area (*pixels*)	Bounding Box (*left*, *top*, *right*, *bottom*)	Center (x_c, y_c)
2	14978	(887, 21, 1144, 399)	(1049.7, 242.8)
3	36156	(40, 37, 438, 419)	(261.9, 209.5)
4	25904	(464, 126, 841, 382)	(680.6, 240.6)
5	2024	(387, 281, 442, 341)	(414.2, 310.6)
6	2293	(244, 367, 342, 506)	(294.4, 439.0)
7	4394	(406, 400, 507, 512)	(454.1, 457.3)
8	29777	(510, 416, 883, 765)	(704.9, 583.9)
9	20724	(833, 497, 1168, 759)	(1016.0, 624.1)
10	16566	(82, 558, 411, 821)	(208.7, 661.6)

[Burger & Burge]

Orientation + Eccentricity

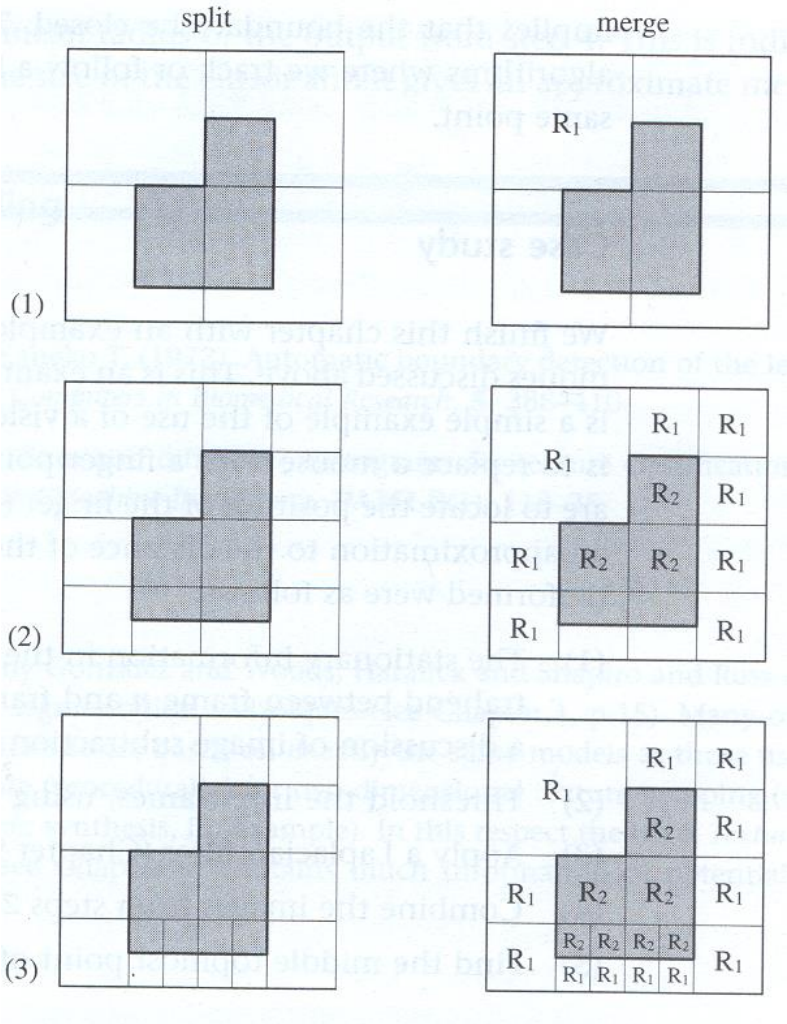


[Burger & Burge]

SEGMENTATION BY RECURSIVE SUBDIVISION

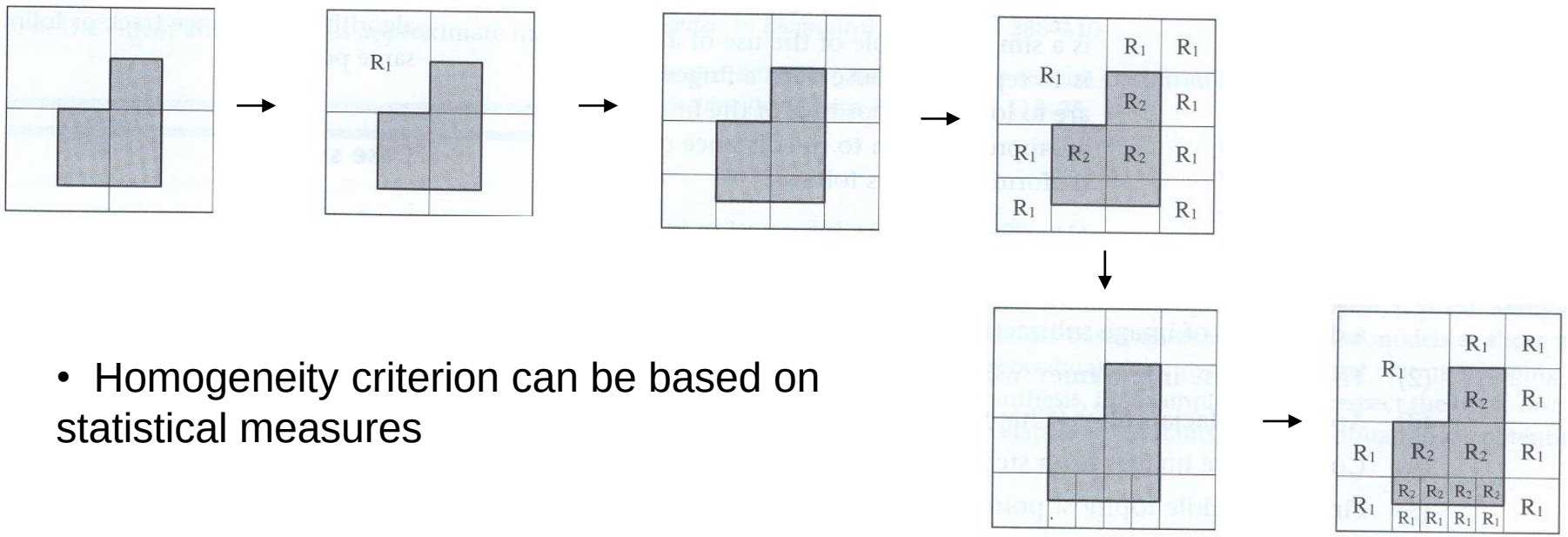
II 3 - Region split-and-merging segmentation

- “Divide-and-Conquer” strategy
- Image subdivision until we get homogeneous regions
- No need for “seed pixels”
- But, “*blocky*” regions



- Simple solution:

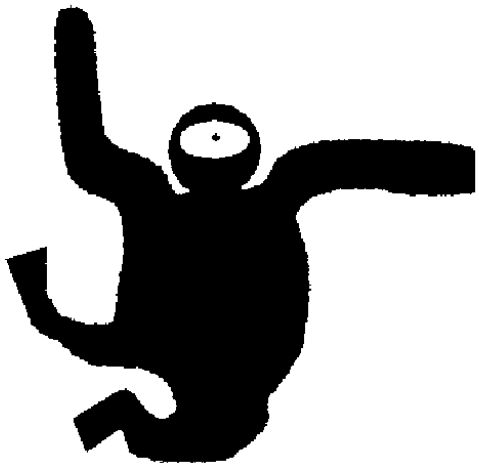
```
If current region homogeneity test is FALSE
    then split into four quadrants
         attempt to merge those quadrants
         recursively call the procedure
for each subdivision
   find any remaining merges
```

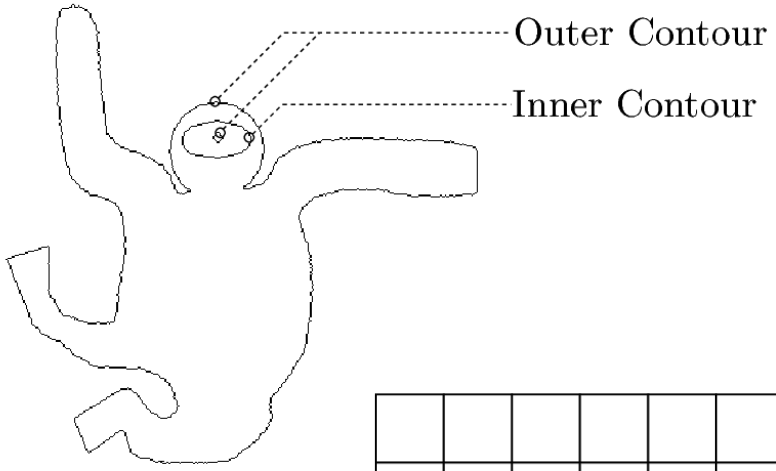


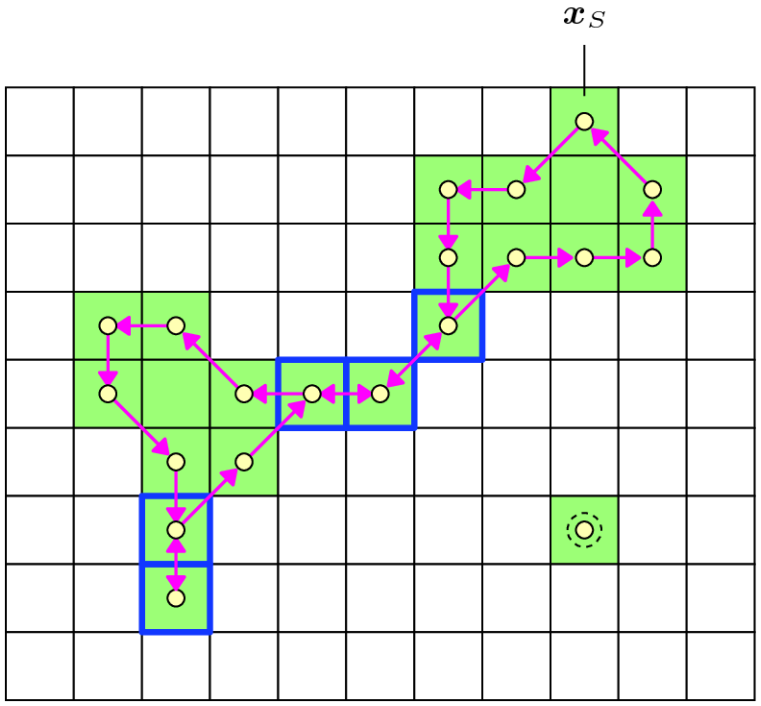
- Homogeneity criterion can be based on statistical measures

REPRESENTING REGION CONTOURS

Region Contours

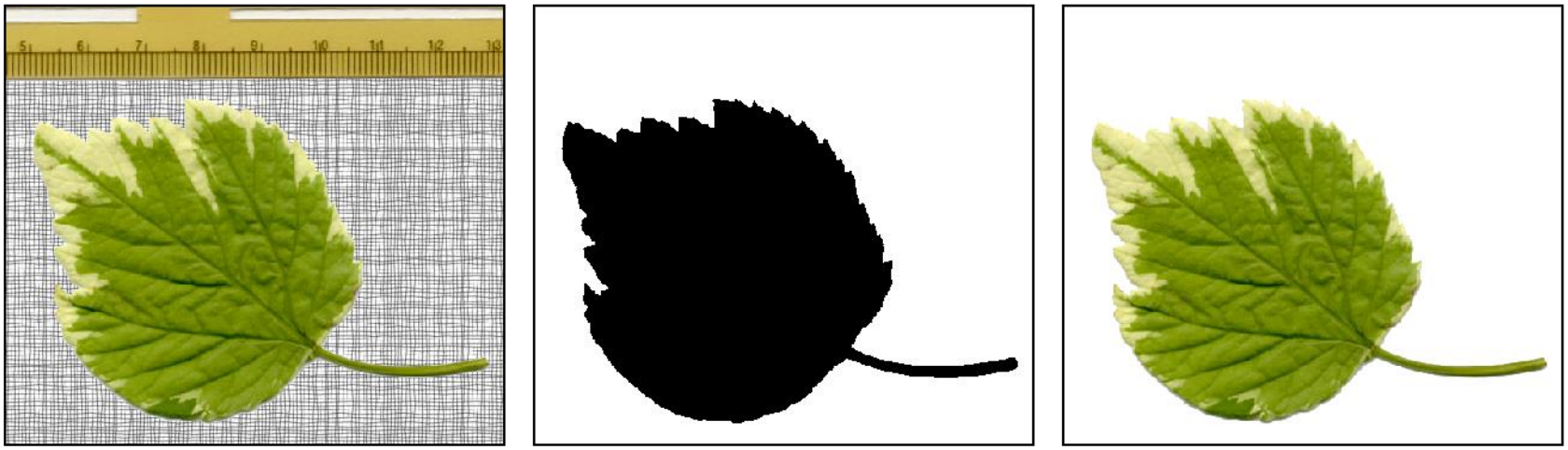






[Burger & Burge]

Representation – Binary Mask



(a) (b) (c)

[Burger & Burge]

Representation – Run-Length Encoding

Bitmap

	0	1	2	3	4	5	6	7	8
0									
1			×	×	×	×	×	×	
2									
3					×	×	×	×	
4		×	×	×		×	×	×	
5	×	×	×	×	×	×	×	×	×
6									

$\longrightarrow$

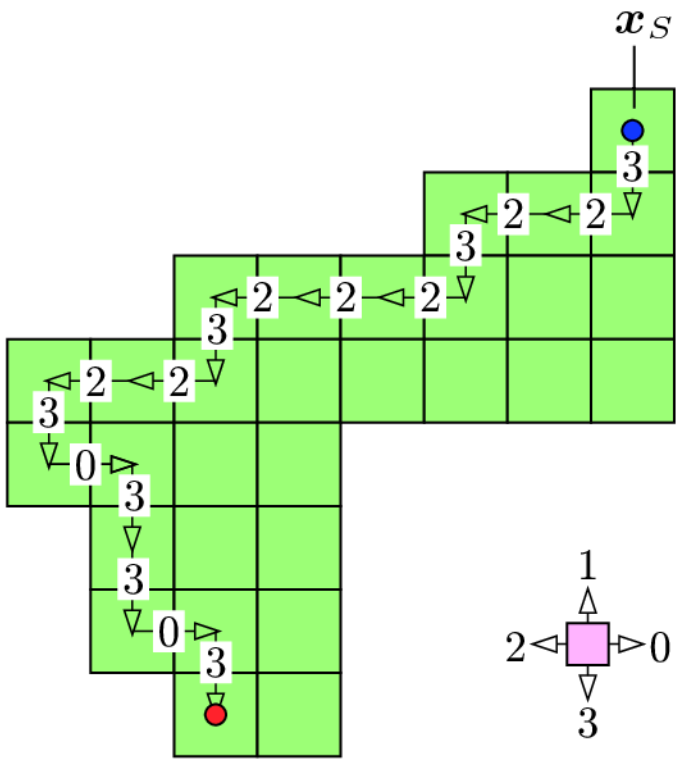
RLE

$\langle \text{row}, \text{column}, \text{length} \rangle$

$\langle 1, 2, 6 \rangle$
$\langle 3, 4, 4 \rangle$
$\langle 4, 1, 3 \rangle$
$\langle 4, 5, 3 \rangle$
$\langle 5, 0, 9 \rangle$

[Burger & Burge]

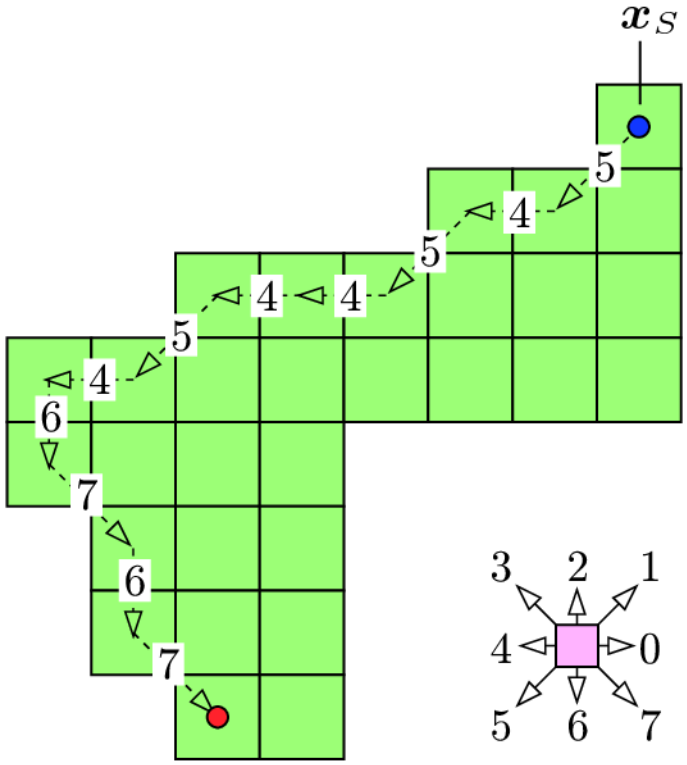
Representation – Chain Codes



4-Chain Code

3223222322303303...111

$length = 28$



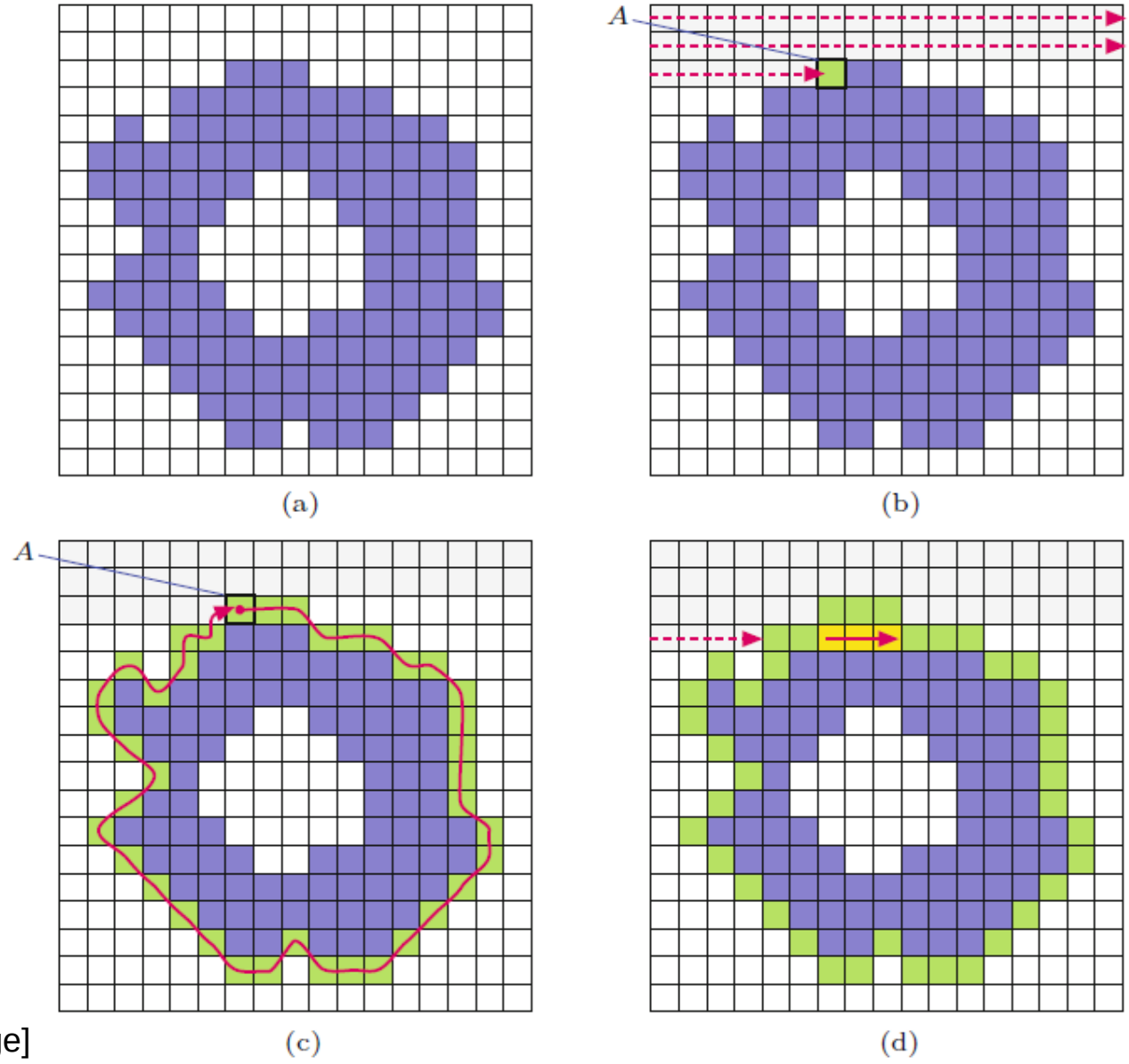
8-Chain Code

54544546767...222

$length = 18 + 5\sqrt{2} \approx 25$

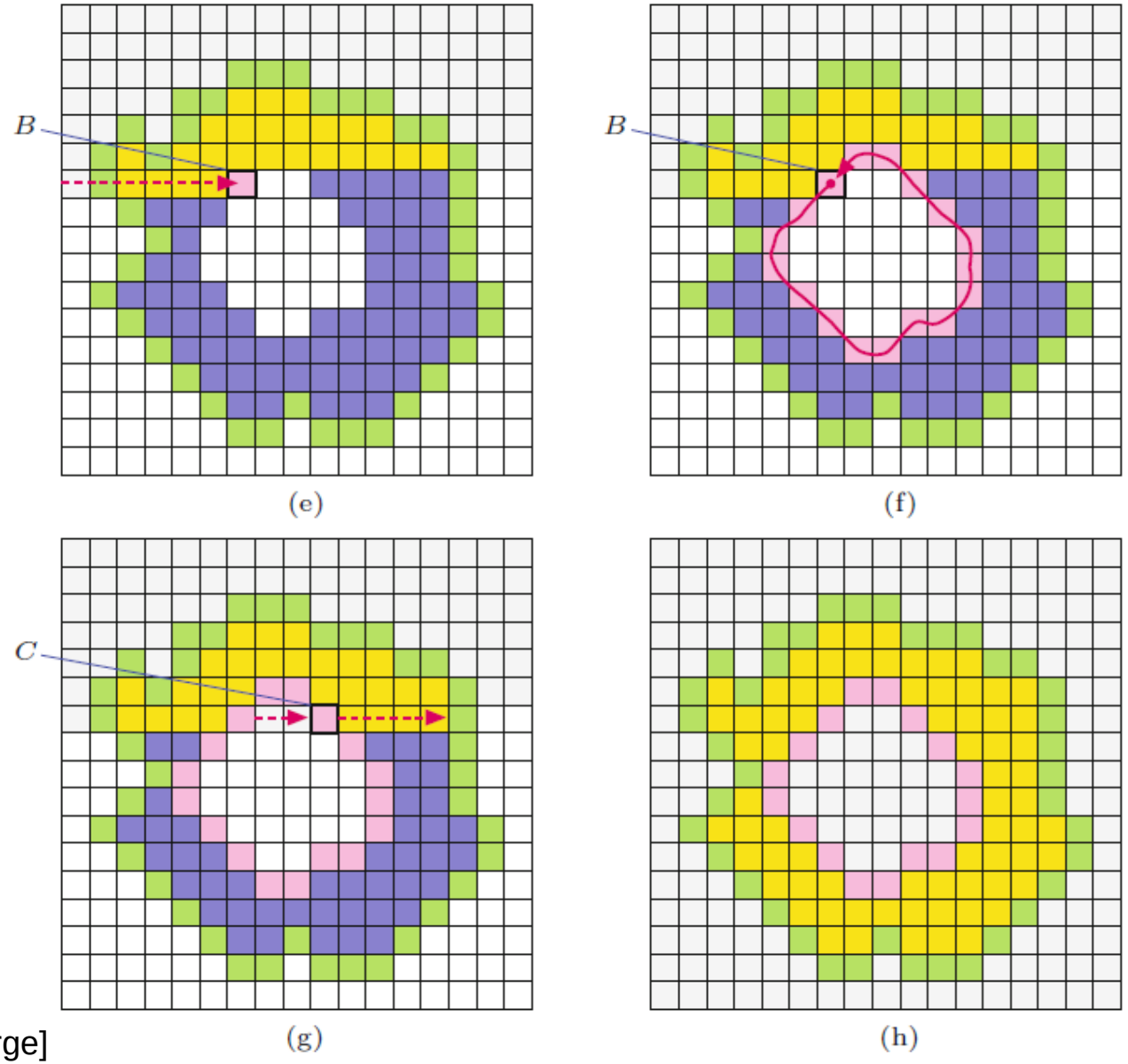
[Burger & Burge]

Combining: Region Labeling + Contour Following



[Burger & Burge]

Combining: Region Labeling + Contour Following



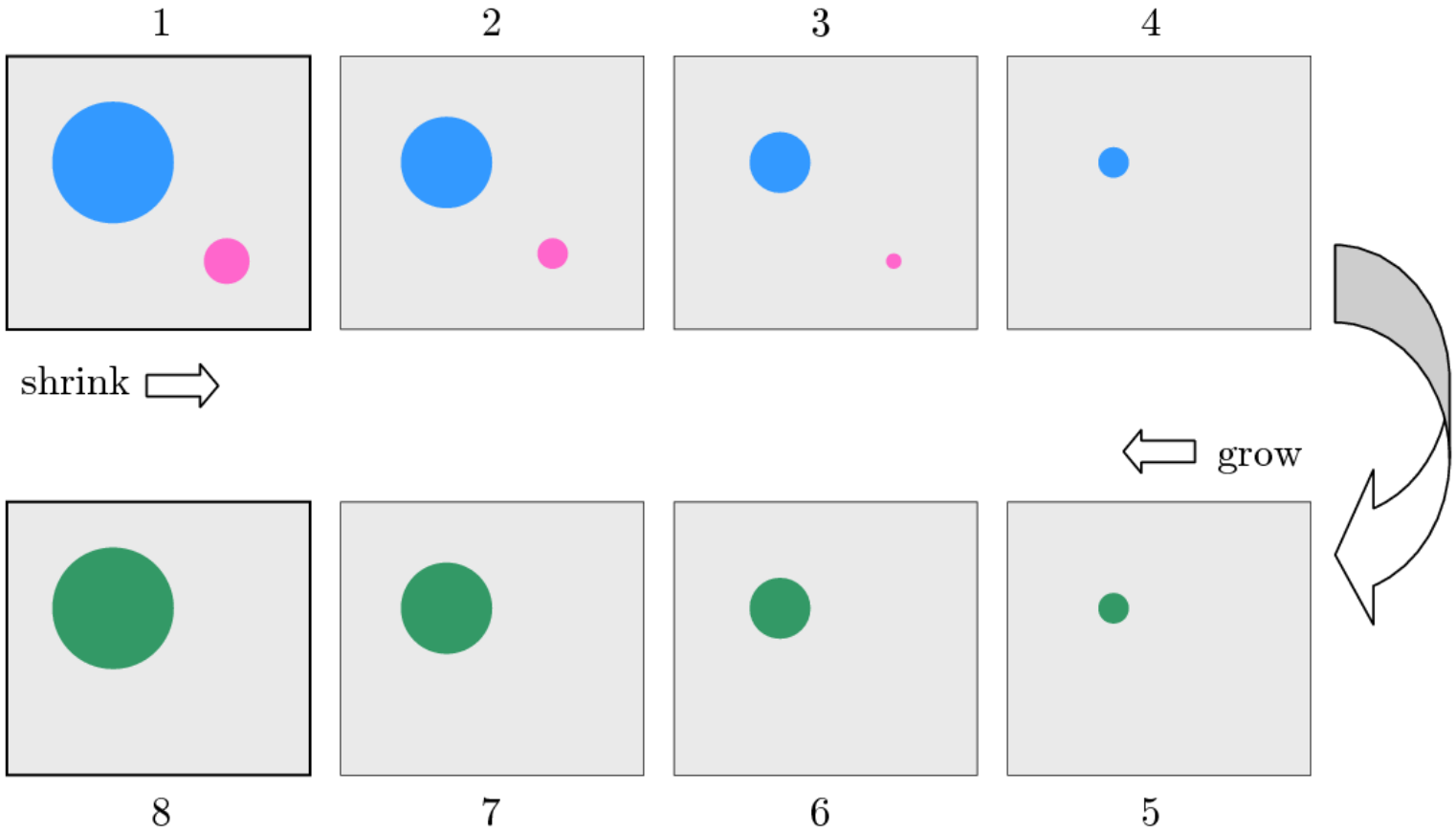
[Burger & Burge]

MATHEMATICAL MORPHOLOGY

III. Mathematical Morphology Operations

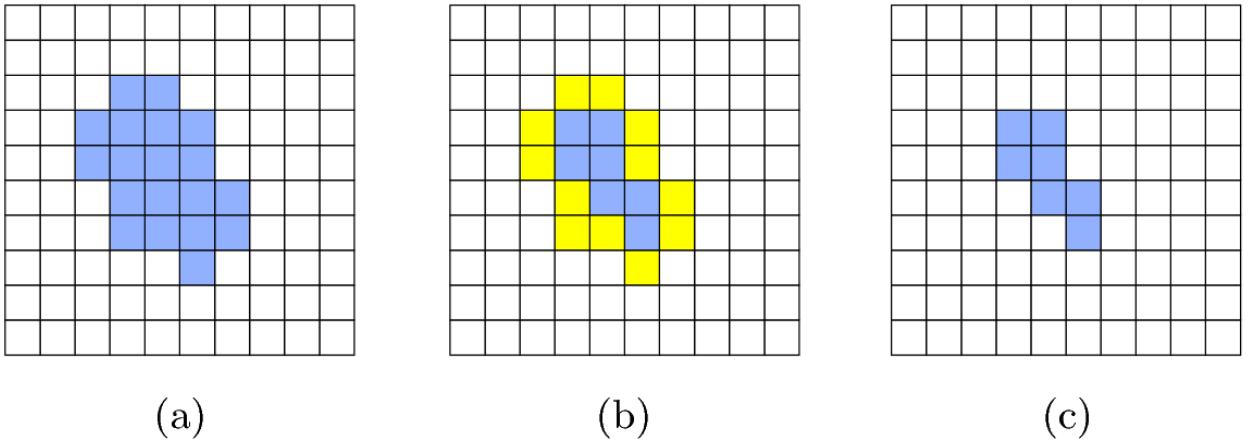
- Mathematical morphology operations treat images as sets of points
- They modify in a controlled way the structure / morphology of an image
- They can be easily applied to binary images
- But can also be applied to grayscale or color images
- Encompass a set of methods used in image analysis for:
  - Filtering
  - Segmentation
  - …

Fundamental ideas



[Burger & Burge]

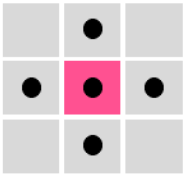
Fundamental ideas



[Burger & Burge]

- Structuring element

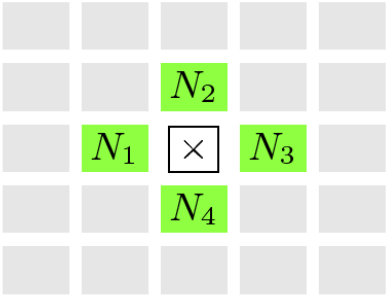
$$H(i,j) \in \{0,1\}$$

$H =$ 

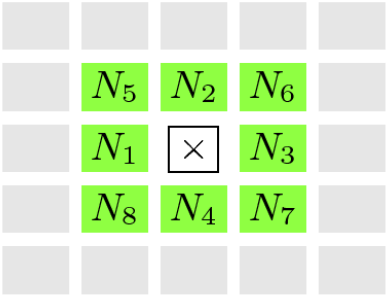
origin (hot spot)

- Neighborhoods

$\mathcal{N}_4$



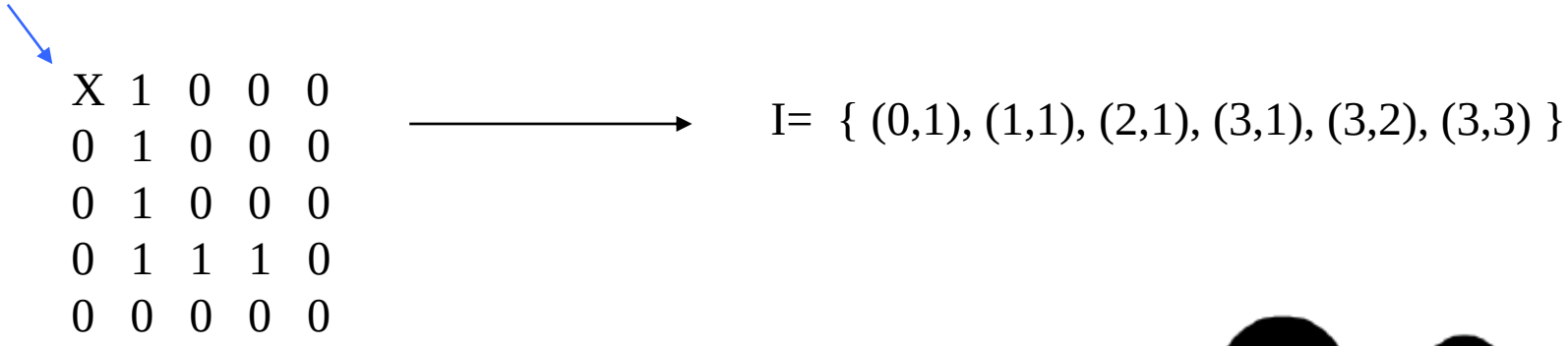
$\mathcal{N}_8$



[Burger & Burge]

- Mathematical morphology operations treat images as sets of points
- They can be easily applied to binary images
- A binary image can be represented as a set of 2D points by listing, row by row, the coordinates of the foreground (black) pixels:

origin



- Represent a binary object as a set of point in 2D space

Operations on sets of 2D points

- Binary image

$$I(u,v) \in \{0,1\} \qquad \boldsymbol{p} = (u,v)$$

- Foreground

$$\mathcal{Q}_I = \{\boldsymbol{p} \mid I(\boldsymbol{p}) = 1\}$$

- Background

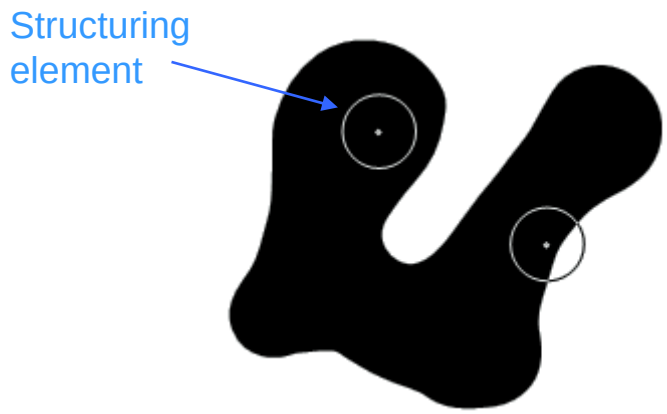
$$\mathcal{Q}_{\bar{I}} = \bar{\mathcal{Q}}_I = \{\boldsymbol{p} \in \mathbb{Z}^2 \mid \boldsymbol{p} \notin \mathcal{Q}_I\}$$

- Union

$$\mathcal{Q}_{I_1 \vee I_2} = \mathcal{Q}_{I_1} \cup \mathcal{Q}_{I_2}$$

[Burger & Burge]

- Some morphological operations can be regarded as "*shape filtering*"
- The "shape" has to be known *a priori*
- Scan an image with a given structuring element (SE)
- There are locations where the SE is:
  - fully inside the object
  - partially inside the object



- Let's consider an algorithmic approach to Mathematical Morphology

Main morphological operations

- Dilation
- Erosion

Basic operations

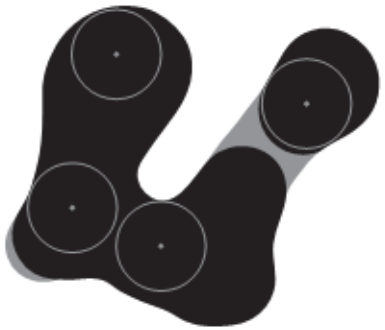
- Opening
- Closing

Composite operations

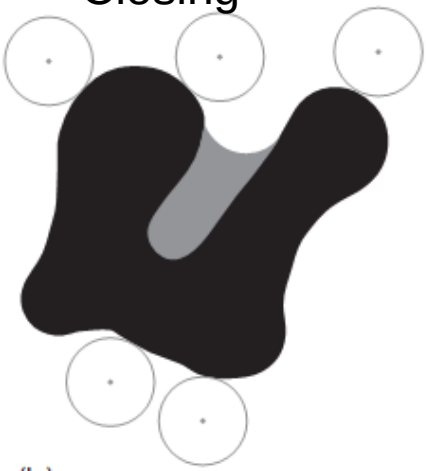
Dilation

Erosion

Opening

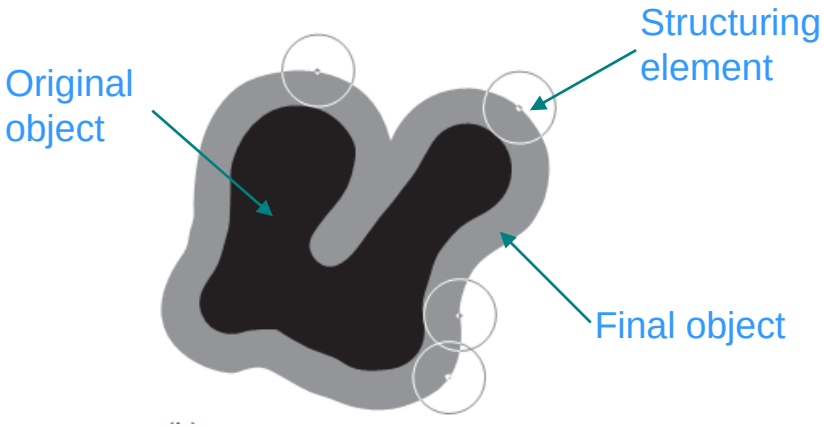


Closing



DILATION & EROSION

Dilation



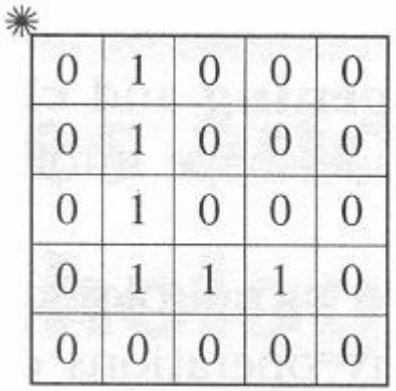
$$I \oplus X = \{j | j = i + x,\ i \in I,\ x \in X\}$$

- Replicate the SE at each original image foreground pixel

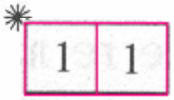
- In general, it originates an expanded version of the foreground

- Small holes and intrusions are filled

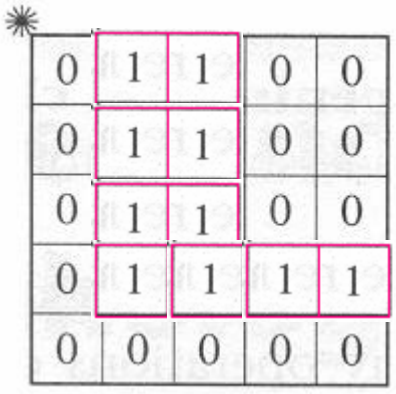
Place the origin of the SE at each foreground (1) pixel of image I and copy all SE 1 pixels to the corresponding pixels in the result image

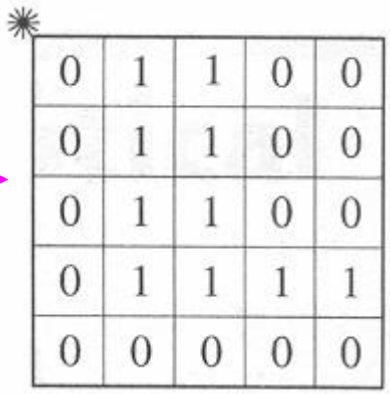


Original image



Structuring element





Final image

Example

$$I \oplus X = \{j | j = i + x,\ i \in I,\ x \in X\}$$

$$I \oplus X = (I + \{(0,0)\}) \cup (I + \{(0,1)\})$$

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

Image I

1	1

Structuring element X

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

Dilation $I \oplus X$

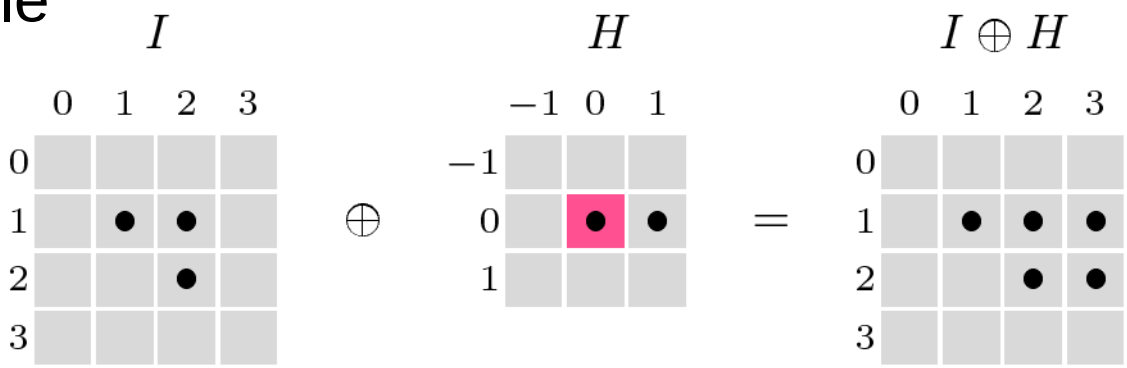
$$I = \{(0,1),(1,1),(2,1),(3,1),(3,2),(3,3)\}$$

$$I + \{(0,0)\} = \{(0,1),(1,1),(2,1),(3,1),(3,2),(3,3)\}$$

$$I + \{(0,1)\} = \{(0,2),(1,2),(2,2),(3,2),(3,3),(3,4)\}$$

$$I \oplus X = \{(0,1)\ (0,2),(1,1),(1,2),(2,1),(2,2),(3,1),(3,2),(3,3),(3,4)\}$$

Another example



$$I \equiv \{(1,1),(2,1),(2,2)\},\ H \equiv \{(\mathbf{0},\mathbf{0}),(\mathbf{1},\mathbf{0})\}$$

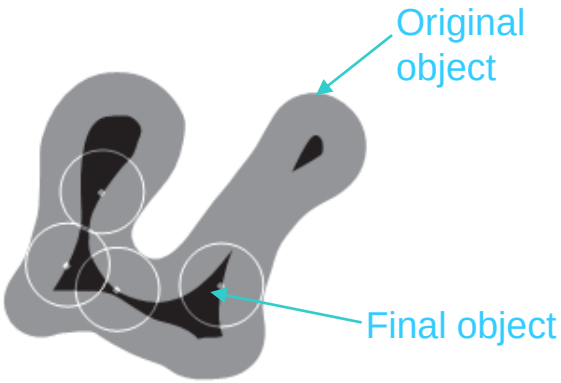
$$\begin{aligned} I \oplus H \equiv \{ &(1,1)+(\mathbf{0},\mathbf{0}),\ (1,1)+(\mathbf{1},\mathbf{0}),\\ &(2,1)+(\mathbf{0},\mathbf{0}),\ (2,1)+(\mathbf{1},\mathbf{0}),\\ &(2,2)+(\mathbf{0},\mathbf{0}),\ (2,2)+(\mathbf{1},\mathbf{0})\ \} \end{aligned}$$

$$I \oplus H \equiv \left\{ (\boldsymbol{p}+\boldsymbol{q}) \mid \text{for some } \boldsymbol{p} \in I \text{ and } \boldsymbol{q} \in H \right\}$$

$$I \oplus H \equiv \bigcup_{\boldsymbol{p} \in I} H_{\boldsymbol{p}} = \bigcup_{\boldsymbol{q} \in H} I_{\boldsymbol{q}}$$

[Burger & Burge]

Erosion



$$I \ominus X = \{j | (X)\, i \subseteq I\}$$

- The dual operation of dilation (eroding an object using a symmetrical SE is equivalent to dilating the background)

- In general, the eroded object is shrunk

- Holes are enlarged and small extrusions are removed

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

Original image

1	1

Structuring element

Place the origin of the SE at each foreground (1) pixel of image I and set to 1 the corresponding pixel in the result image, whenever the SE pattern exists in the original image

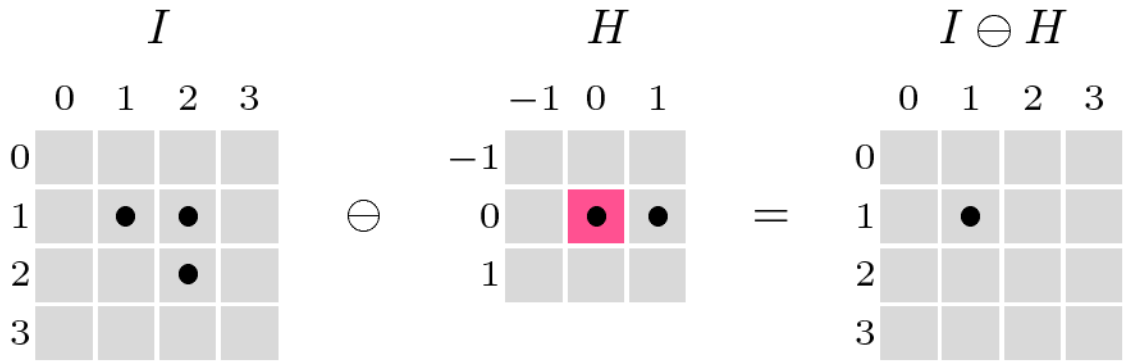
0	1	0	0	0
0	1	0	0	0
0	1	0	0	0
0	1	1	1	0
0	0	0	0	0

→

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

Final image

Example



$$I \equiv \{(1,1),(2,1),(2,2)\},\ H \equiv \{(\mathbf{0},\mathbf{0}),(\mathbf{1},\mathbf{0})\}$$

$$I \ominus H \equiv \{\,(1,1)\,\} \text{ because}$$
$$(1,1)+(\mathbf{0},\mathbf{0}) = (1,1) \in I \quad \textbf{and} \quad (1,1)+(\mathbf{1},\mathbf{0}) = (2,1) \in I$$

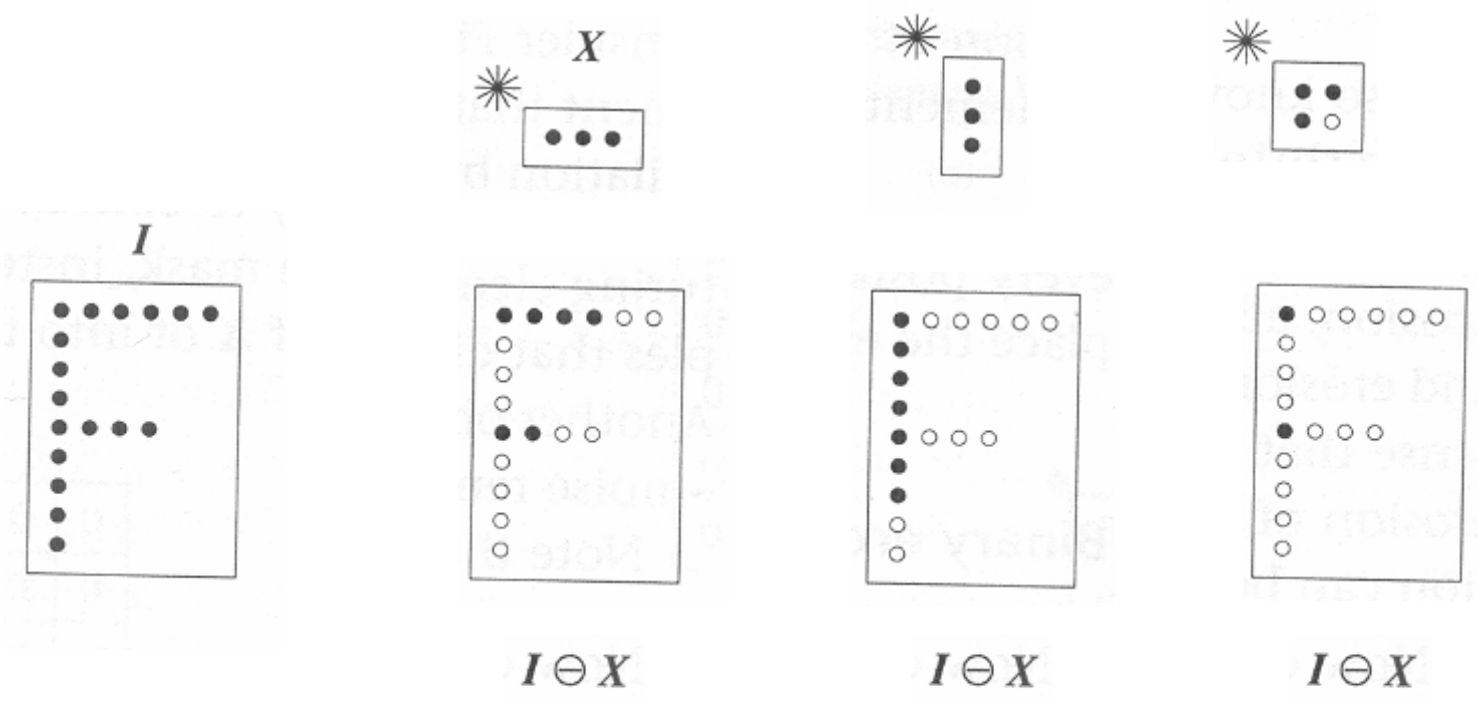
$$I \ominus H \equiv \{\boldsymbol{p} \in \mathbb{Z}^2 \mid (\boldsymbol{p}+\boldsymbol{q}) \in I,\ \text{for every } \boldsymbol{q} \in H\}$$

$$I \ominus H \equiv \{\boldsymbol{p} \in \mathbb{Z}^2 \mid H_{\boldsymbol{p}} \subseteq I\}$$

[Burger & Burge]

Exercise

Determine the erosion result, for each one of the structuring elements:



Properties of Dilation and Erosion

- Dual

$$I \oplus H \equiv \overline{(\bar{I} \ominus H^*)} \qquad I \ominus H \equiv \overline{(\bar{I} \oplus H^*)} \qquad H^* \equiv \{-\boldsymbol{p} \mid \boldsymbol{p} \in H\}$$

- Commutative

$$I \oplus H = H \oplus I \qquad I \ominus H \neq H \ominus I$$

- Associative

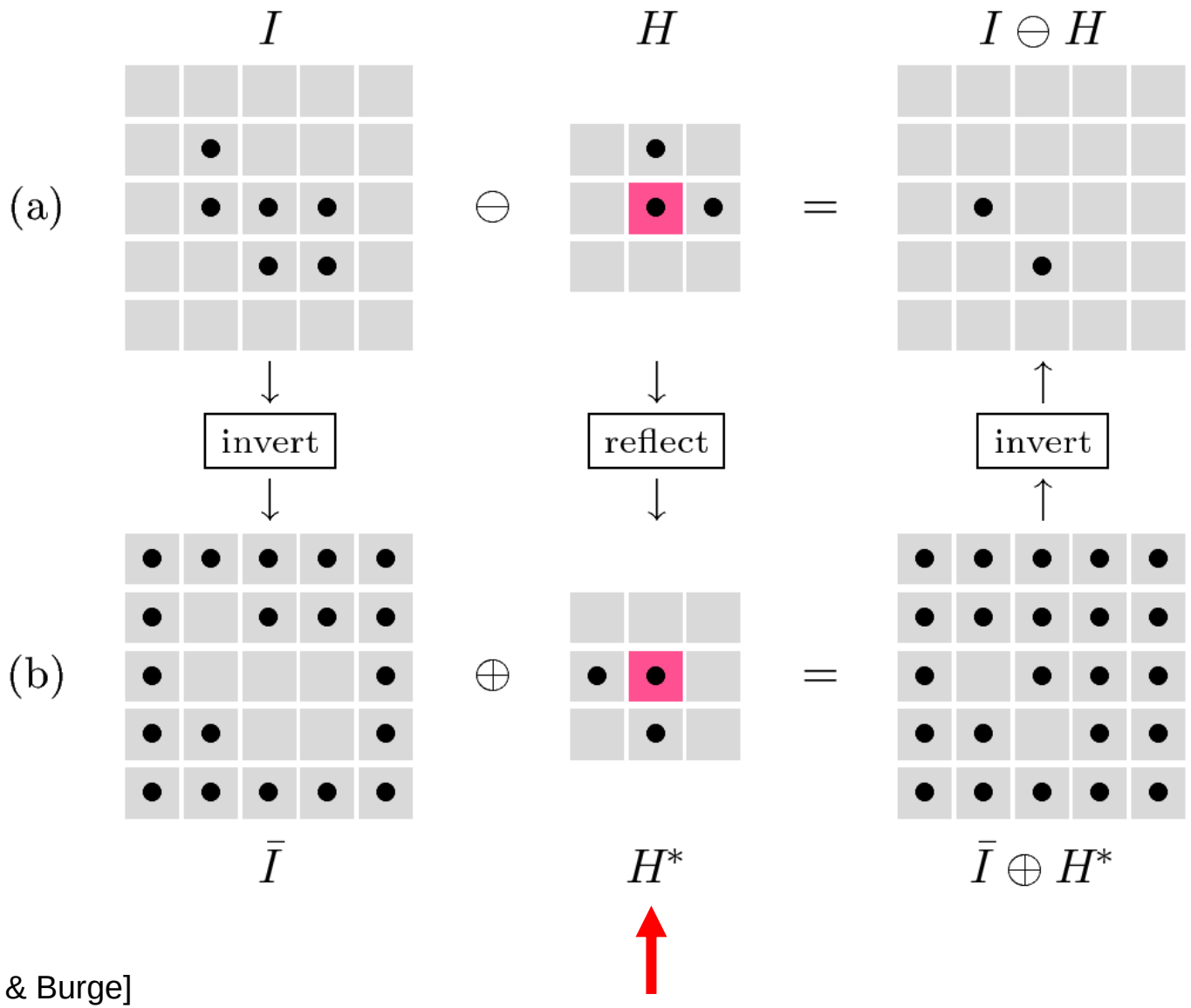
$$(I_1 \oplus I_2) \oplus I_3 = I_1 \oplus (I_2 \oplus I_3) \qquad (I_1 \ominus I_2) \ominus I_3 = I_1 \ominus (I_2 \oplus I_3)$$

$$H_{\text{big}} = H_1 \oplus H_2 \oplus \ldots \oplus H_K$$

$$I \oplus H_{\text{big}} = (\ldots((I \oplus H_1) \oplus H_2) \oplus \ldots \oplus H_K)$$

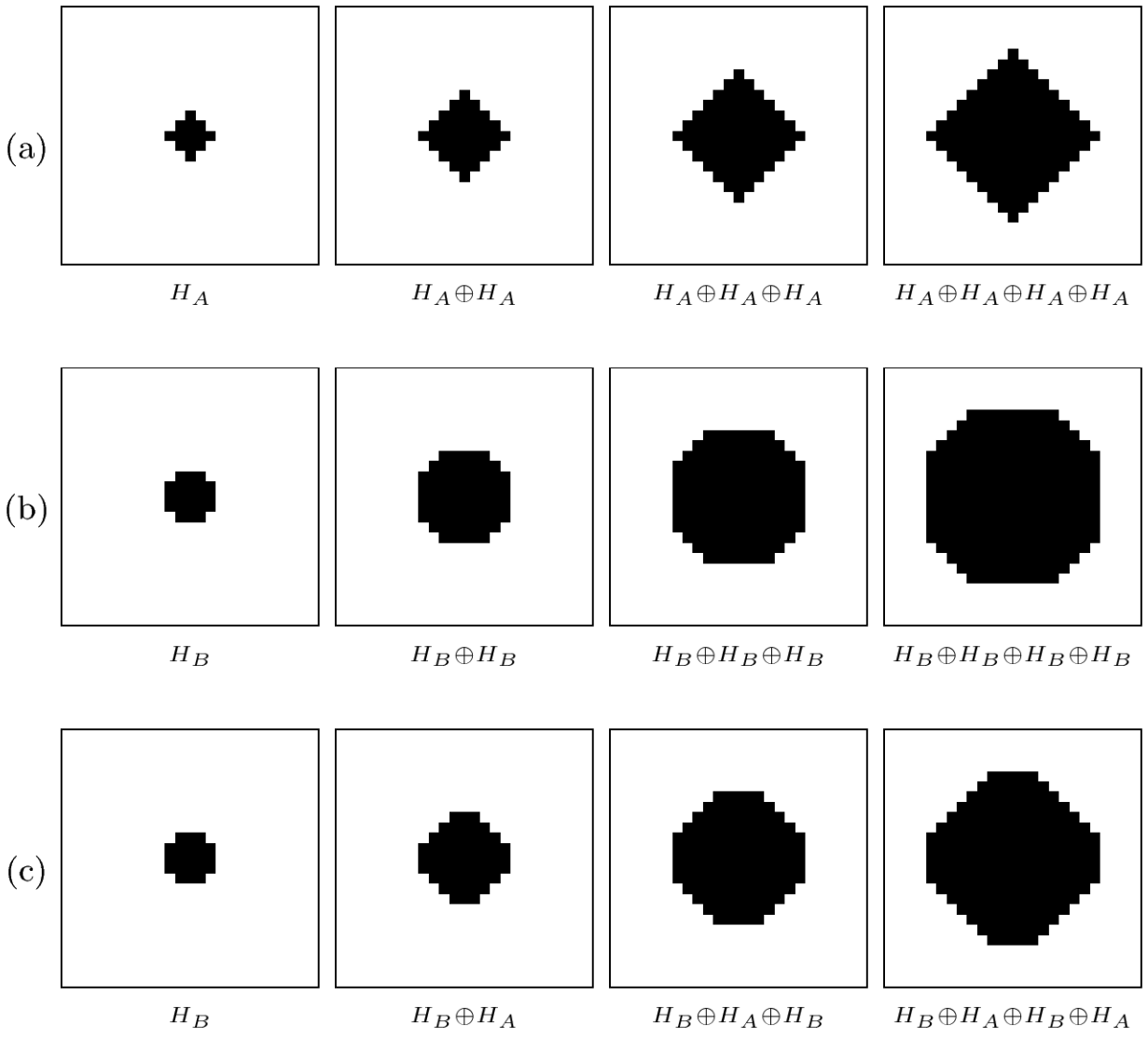
[Burger & Burge]

Example – Dual operations



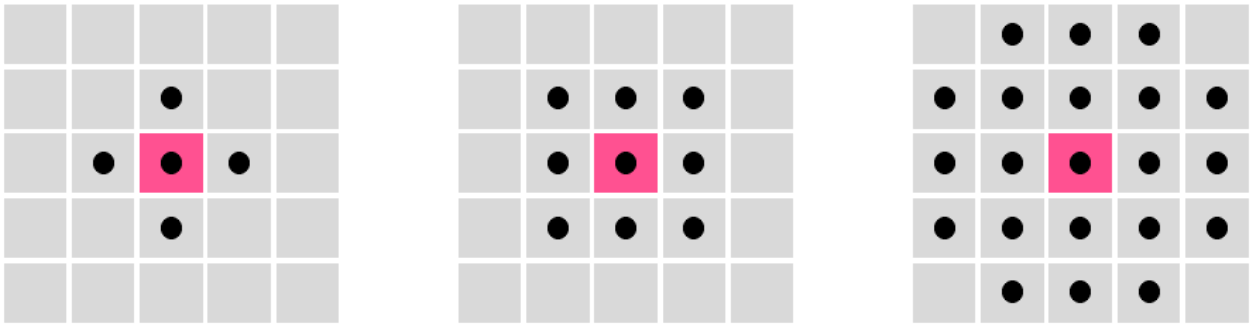
[Burger & Burge]

Example – Repeated application



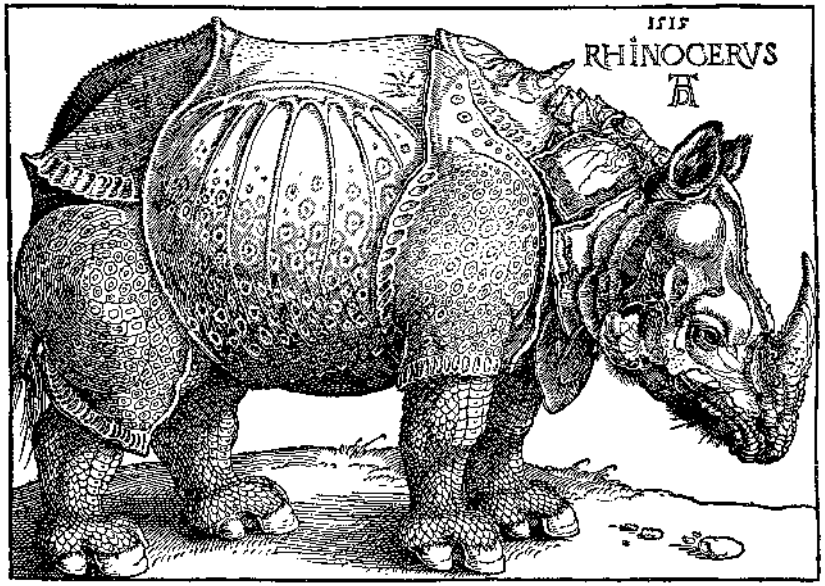
[Burger & Burge]

Usual structuring elements



[Burger & Burge]

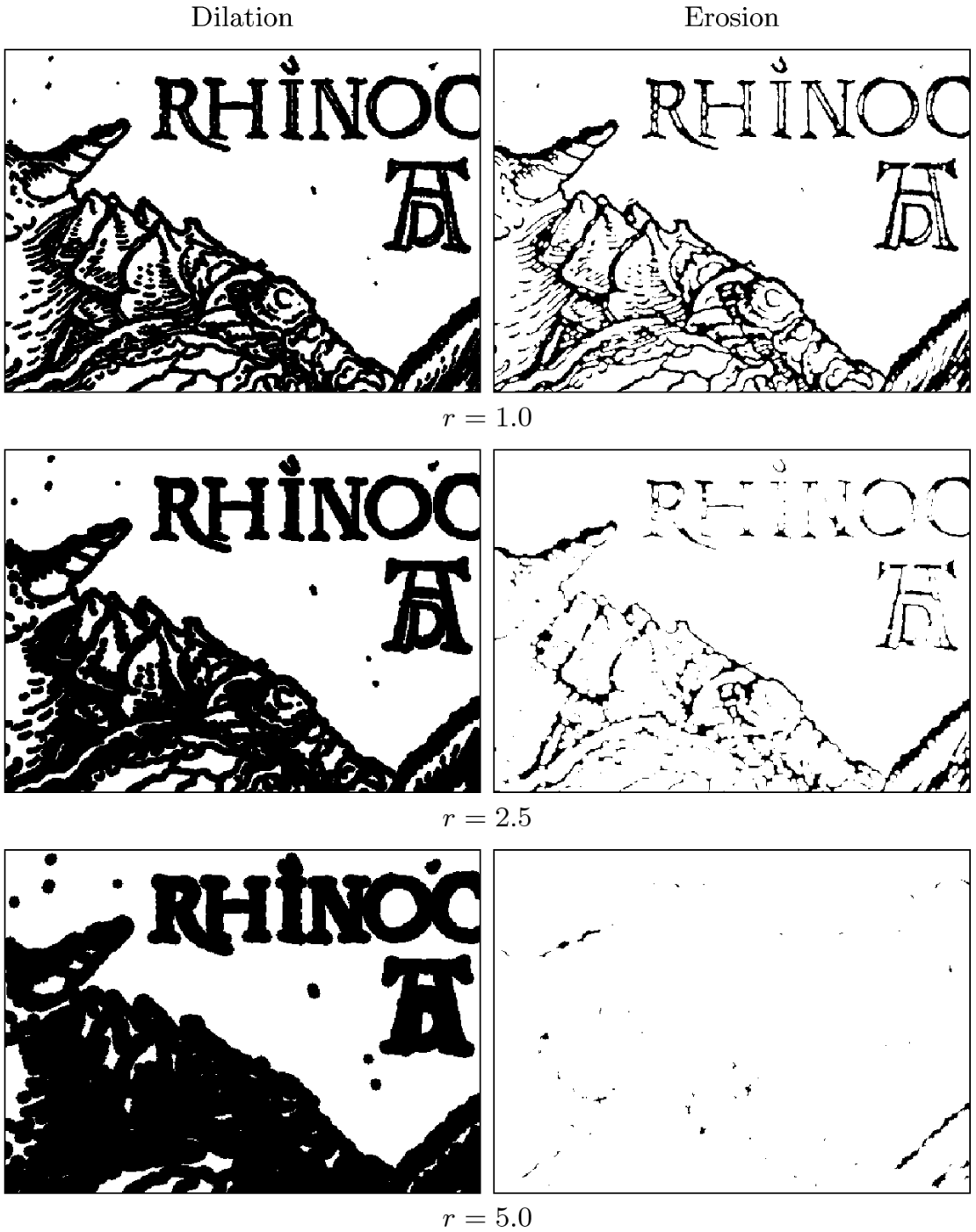
Example – Original image



[Burger & Burge]

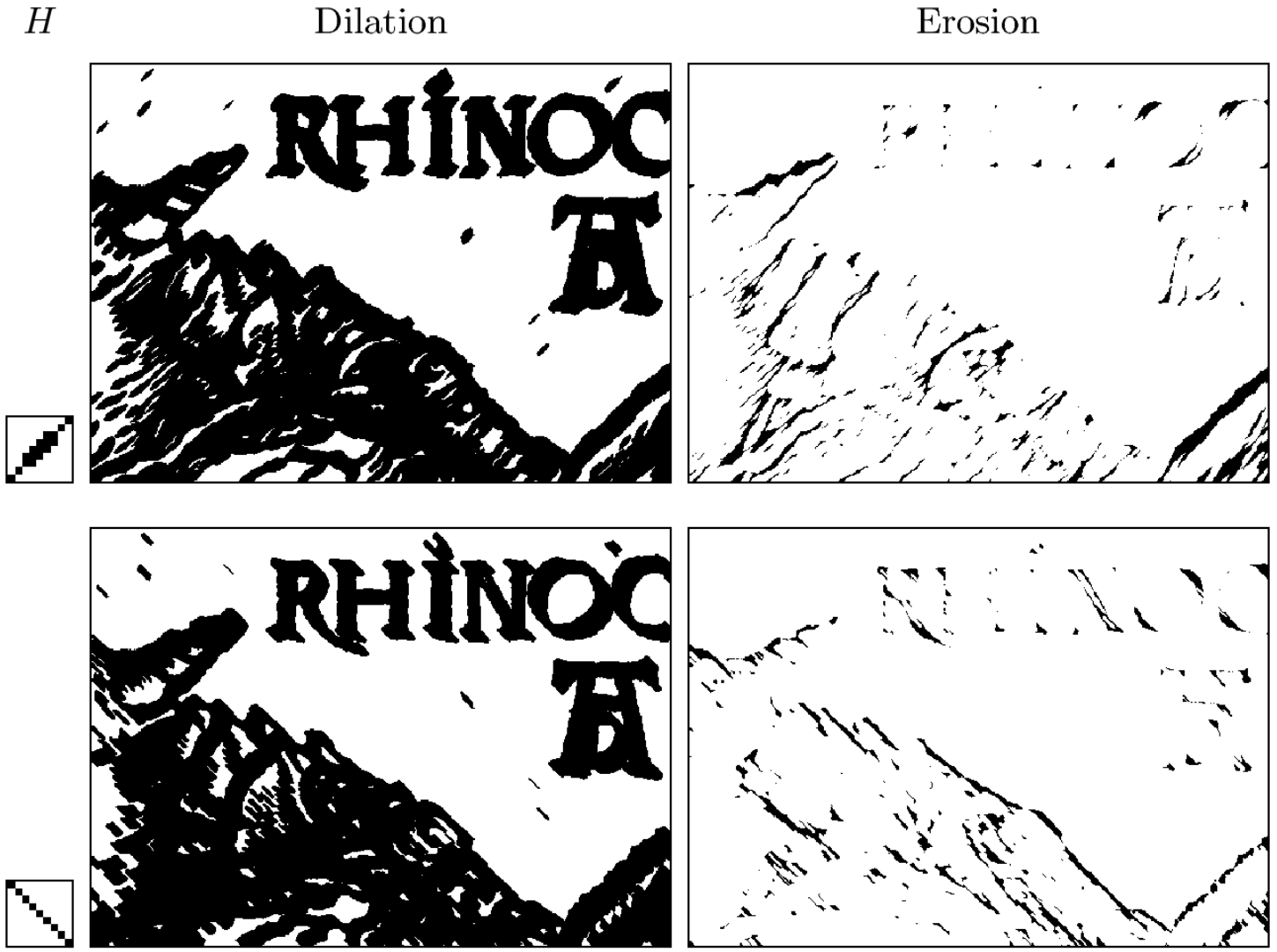
Example

Small Disk



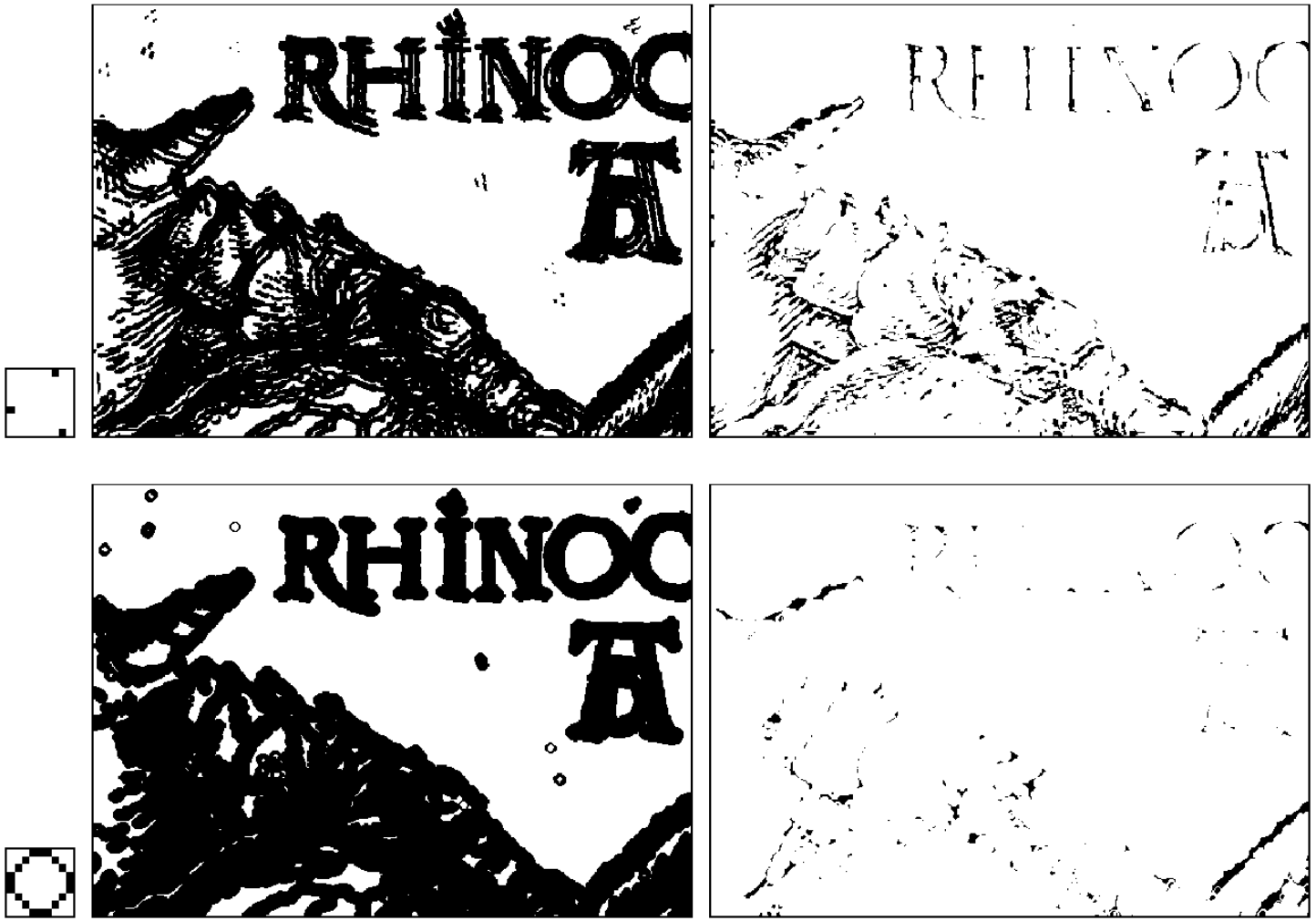
[Burger & Burge]

Example



[Burger & Burge]

Example



[Burger & Burge]

Contour extraction – *Outlining*

- Since the erosion with an appropriate SE results in a isotropic shrinking, foreground contours can be obtained with the composed operation:

$$I - (I \ominus X)$$

- Erode the foreground, then subtract it from the original image

- Isotropic erosion result from symmetrical SEs of size 3x3 or 5x5, …

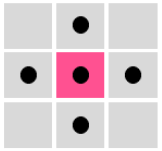
- The size of the SE determines the thickness of the resulting contours

Contour obtained with a 3x3 SE

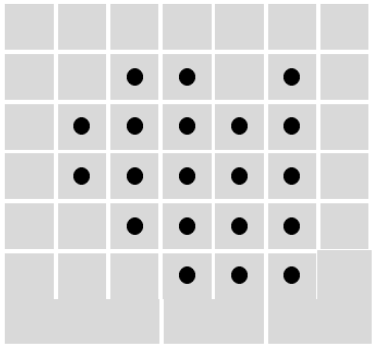
(Amer, 2002)

Example

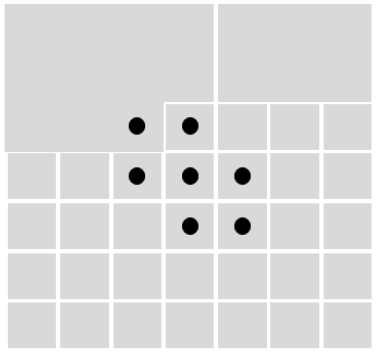
$H_n =$



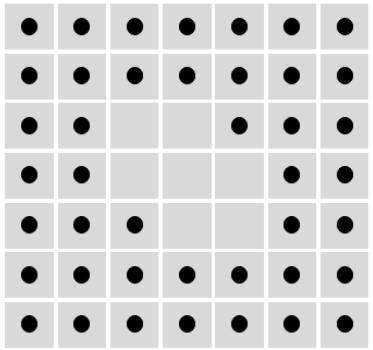
I



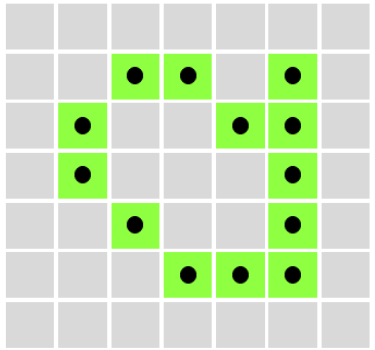
$I \ominus H_n$



$\overline{I \ominus H_n}$



$B = I \cap \overline{I \ominus H_n}$



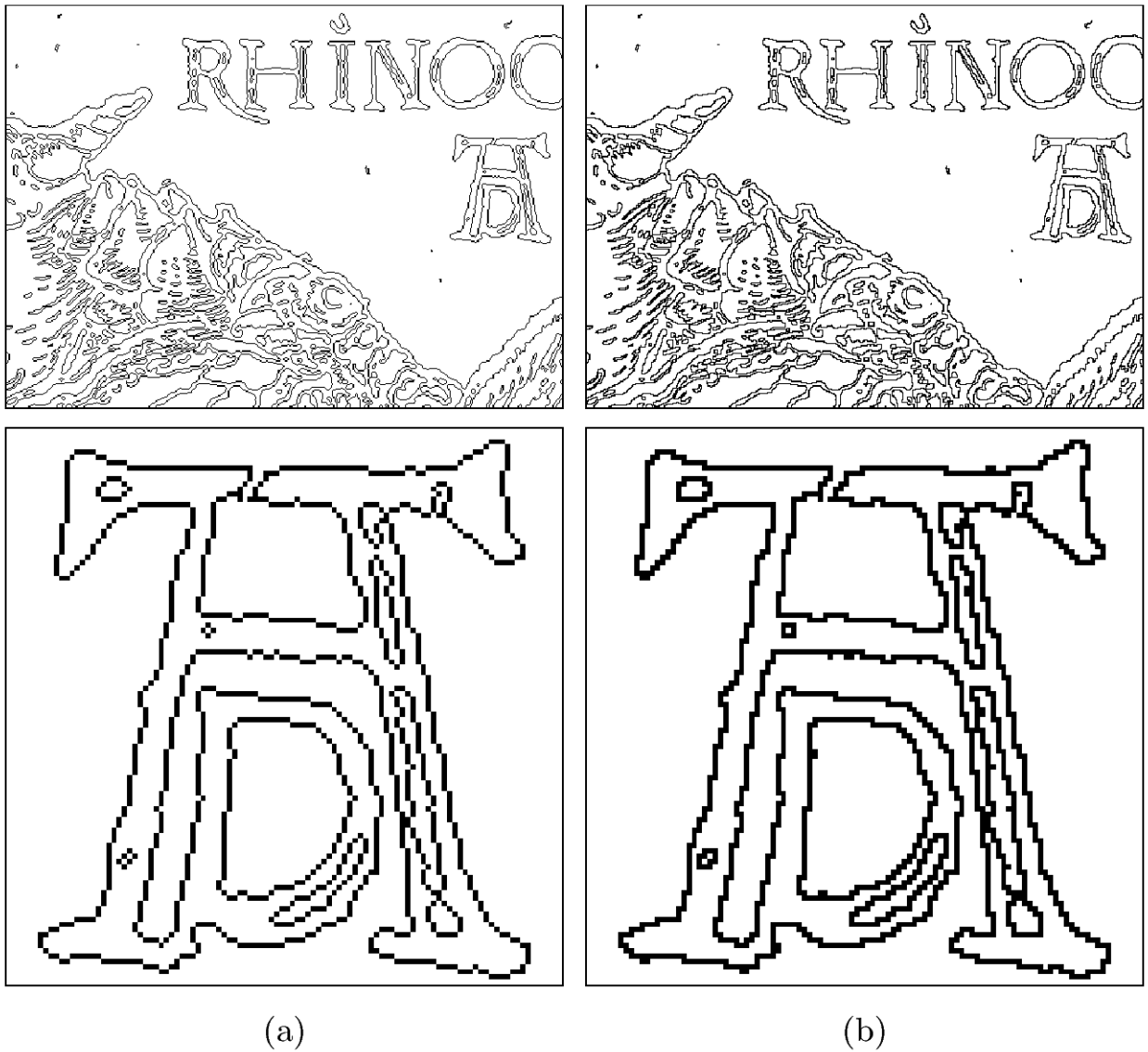
$$I' = I \ominus H_n$$

$$B = I \cap \overline{I'} = I \cap \overline{(I \ominus H_n)}$$

$$B(u,v) = \mathrm{XOR}\big(I(u,v), I'(u,v)\big) \quad \text{for all } (u,v)$$

[Burger & Burge]

Example



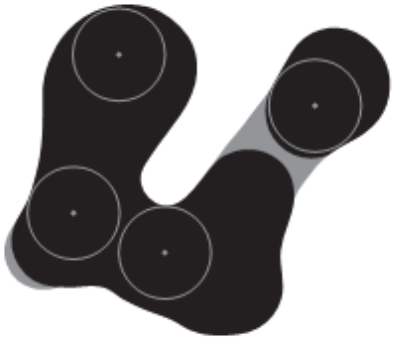
(a) (b)

Contour connectivity vs SE connectivity

[Burger & Burge]

OPENING & CLOSING

Opening



$$I \circ X = (I \ominus X) \oplus X$$

- Erosion followed by a dilation
- Idempotent
- Union of alll SEs that fit inside the object
- Opening with a circular SE tends to:
  - smooth contours
  - break narrow bridges
  - remove small extrusions

The SE is applied to the whole object, but no 1 pixel of the SE can appear outside the object

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

Imagem original

1	1

Elemento estruturante

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

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

$I \circ X$

Example

$$I \circ X = (I \ominus X) \oplus X$$

Erosion followed by a dilation

Erosion:
Place the origin of the SE over each 1 pixel; whenever the SE Is fully inside the object, set to 1 the corresponding pixel, in tthe resulting image

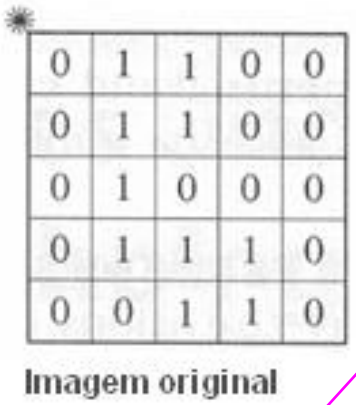
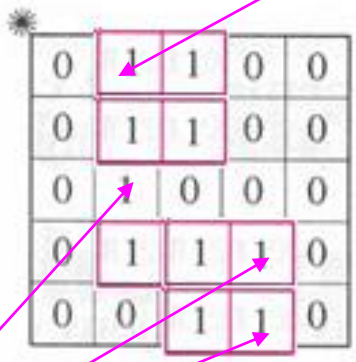
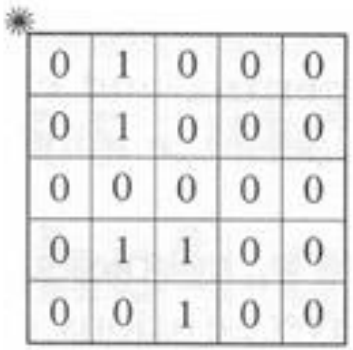
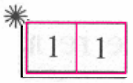


Imagem original





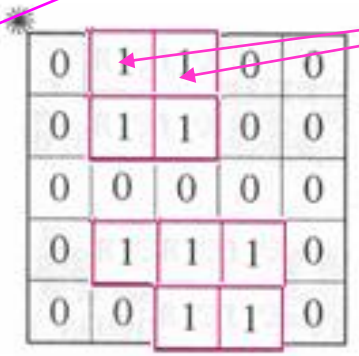
$I \ominus X$



Structuring element

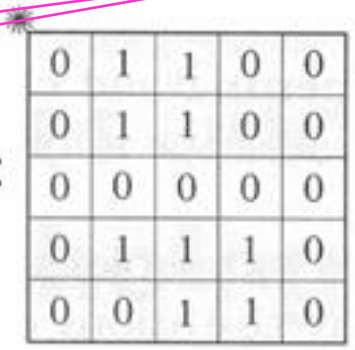
Dilation:
Place the origin of the SE over each 1 pixel; and set to 1 in the final image the pixels corresponding to the SE

These pixels are set to 0 in the result image, since the SE does not fit inside the object, whenever its origin is placed over them



$(I \ominus X) \oplus X$

=

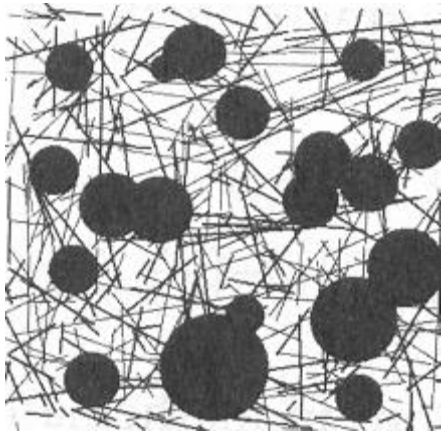


$I \circ X$

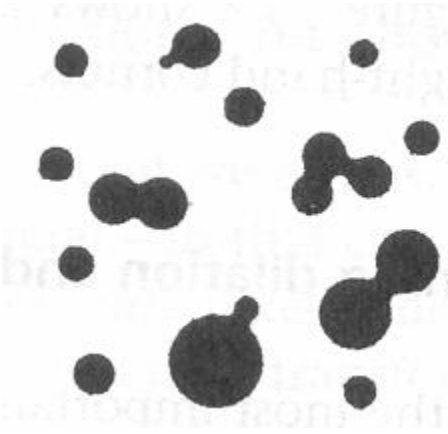
Example

Opening with a circular structuring element

Original image



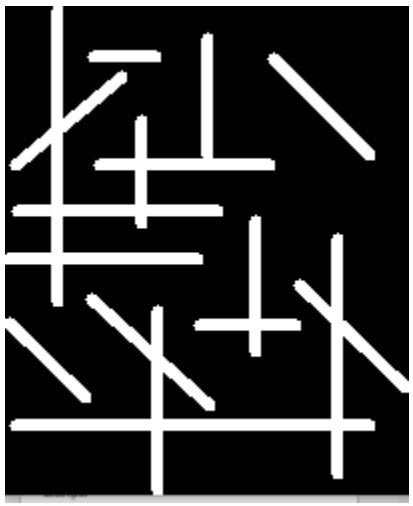
After the erosion

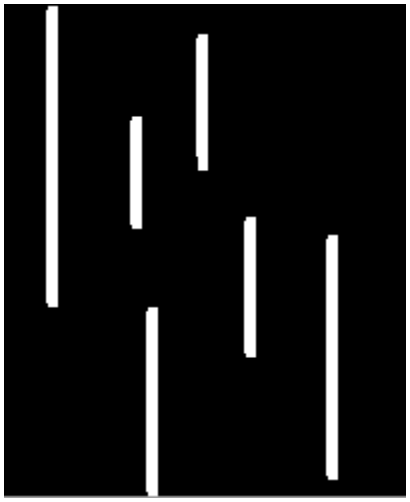


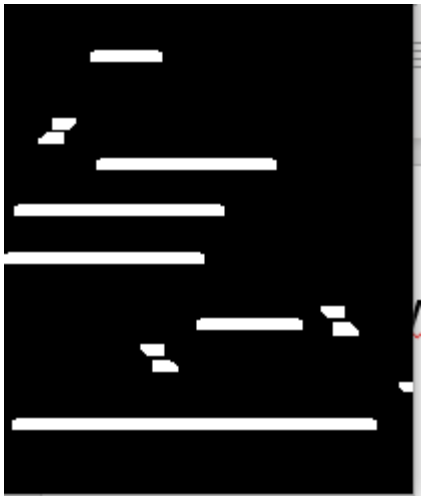
After the dilation

The radius of the SE must be larger than the objects that are to be eliminated

Opening with different structuring elements

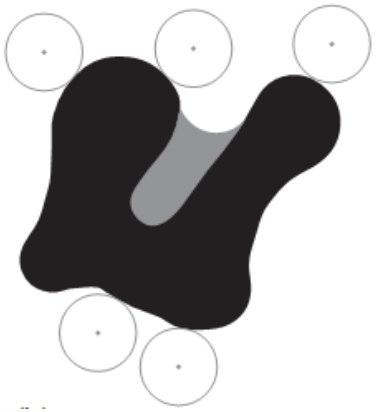






Closing

$$I \bullet X = (I \oplus X) \ominus X$$



- Dilation followed by an erosion
- Dual of *opening*
- Idempotent

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

Original image

1	1

Structuring element

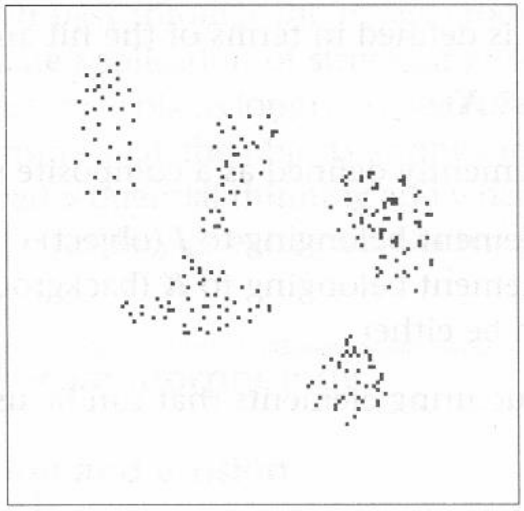
0	1	1	1	0
0	1	1	1	0
0	1	1	0	0
0	1	1	1	1
0	0	1	1	1

?

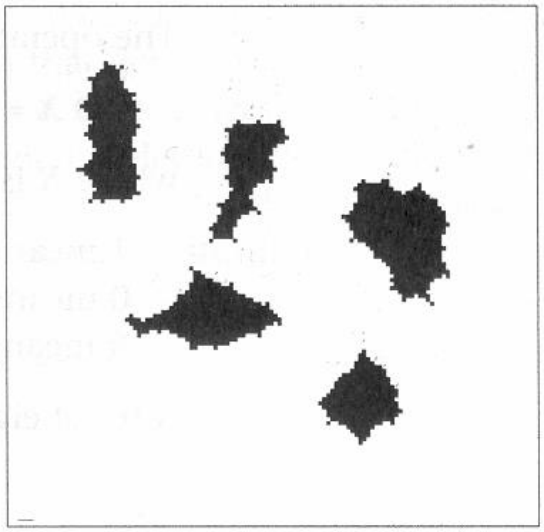
Example

Closing with a circular structuring element

Original image



Final image



Properties of Opening and Closing

$$I \circ H = (I \ominus H) \oplus H \qquad I \bullet H = (I \oplus H) \ominus H$$

- Idempotent

$$I \circ H = (I \circ H) \circ H = ((I \circ H) \circ H) \circ H = \ldots$$
$$I \bullet H = (I \bullet H) \bullet H = ((I \bullet H) \bullet H) \bullet H = \ldots$$

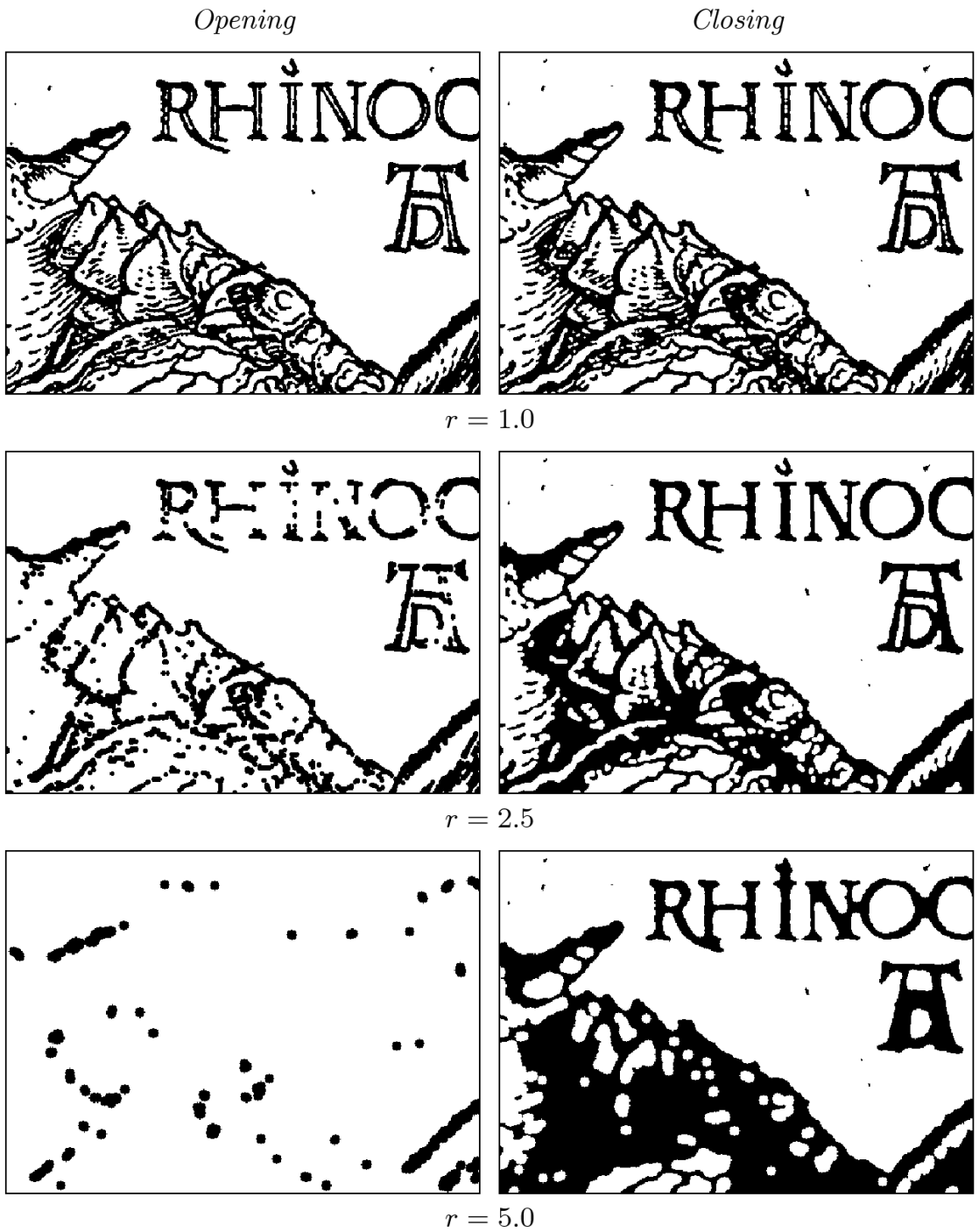
- Dual

$$I \circ H = \overline{(\bar{I} \bullet H)} \quad \text{and} \quad I \bullet H = \overline{(\bar{I} \circ H)}$$

[Burger & Burge]

Example

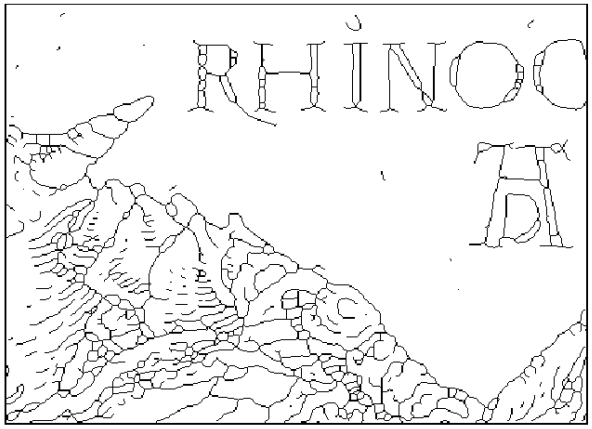
Small Disk

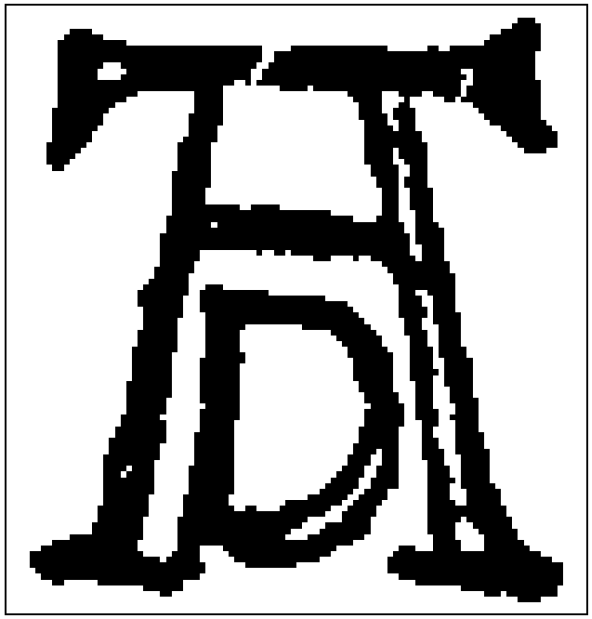


[Burger & Burge]

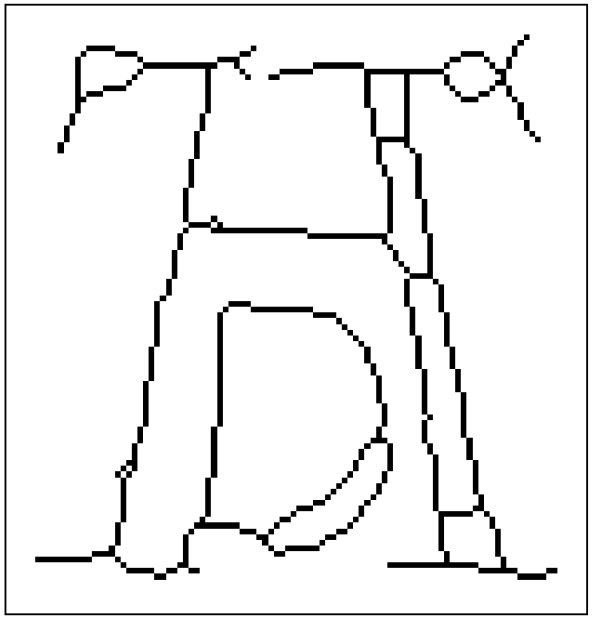
Application – *Skeletonizing / Thinning*

Successive erosions !!





(a)



(b)

[Burger & Burge]

GRAY-SCALE MATHEMATICAL MORPHOLOGY

Grayscale Morpological Operations

- The same operations
- The structuring element represents a real-valued 2D function

$$H(i,j) \in \mathbb{R}\,, \qquad \text{for} \;\; (i,j) \in \mathbb{Z}^2$$

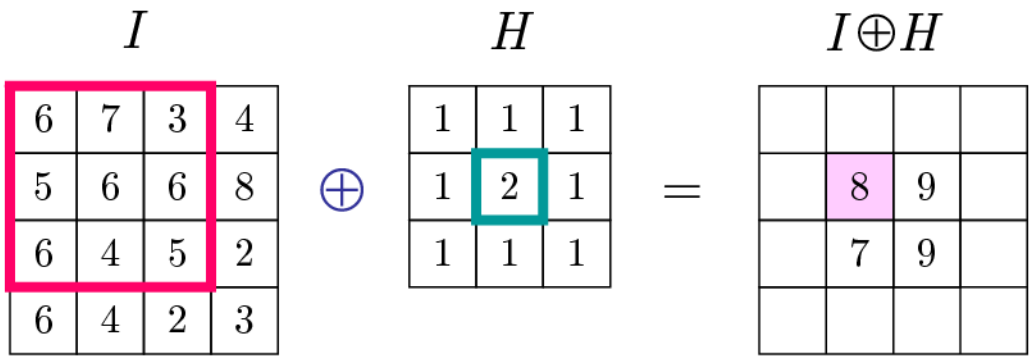
  - Positive, negative or zero values
  - *"Don't care"*

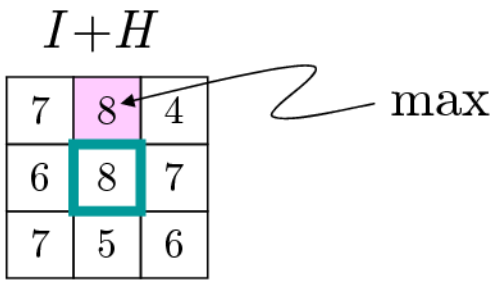
$$\begin{array}{|c|c|c|} \hline 0 & 1 & 0 \\ \hline 1 & 2 & 1 \\ \hline 0 & 1 & 0 \\ \hline \end{array} \neq \begin{array}{|c|c|c|} \hline & 1 & \\ \hline 1 & 2 & 1 \\ \hline & 1 & \\ \hline \end{array}$$

[Burger & Burge]

Dilation

$$(I \oplus H)(u,v) = \max_{(i,j)\in H} \{I(u+i, v+j) + H(i,j)\}$$

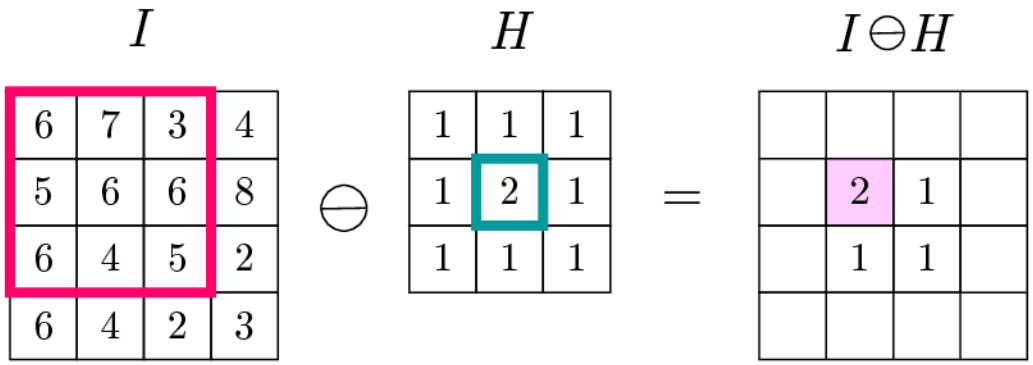


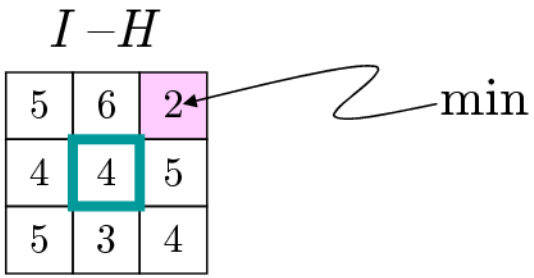


[Burger & Burge]

Erosion

$$(I \ominus H)(u,v) = \min_{(i,j)\in H} \{I(u+i, v+j) - H(i,j)\}$$





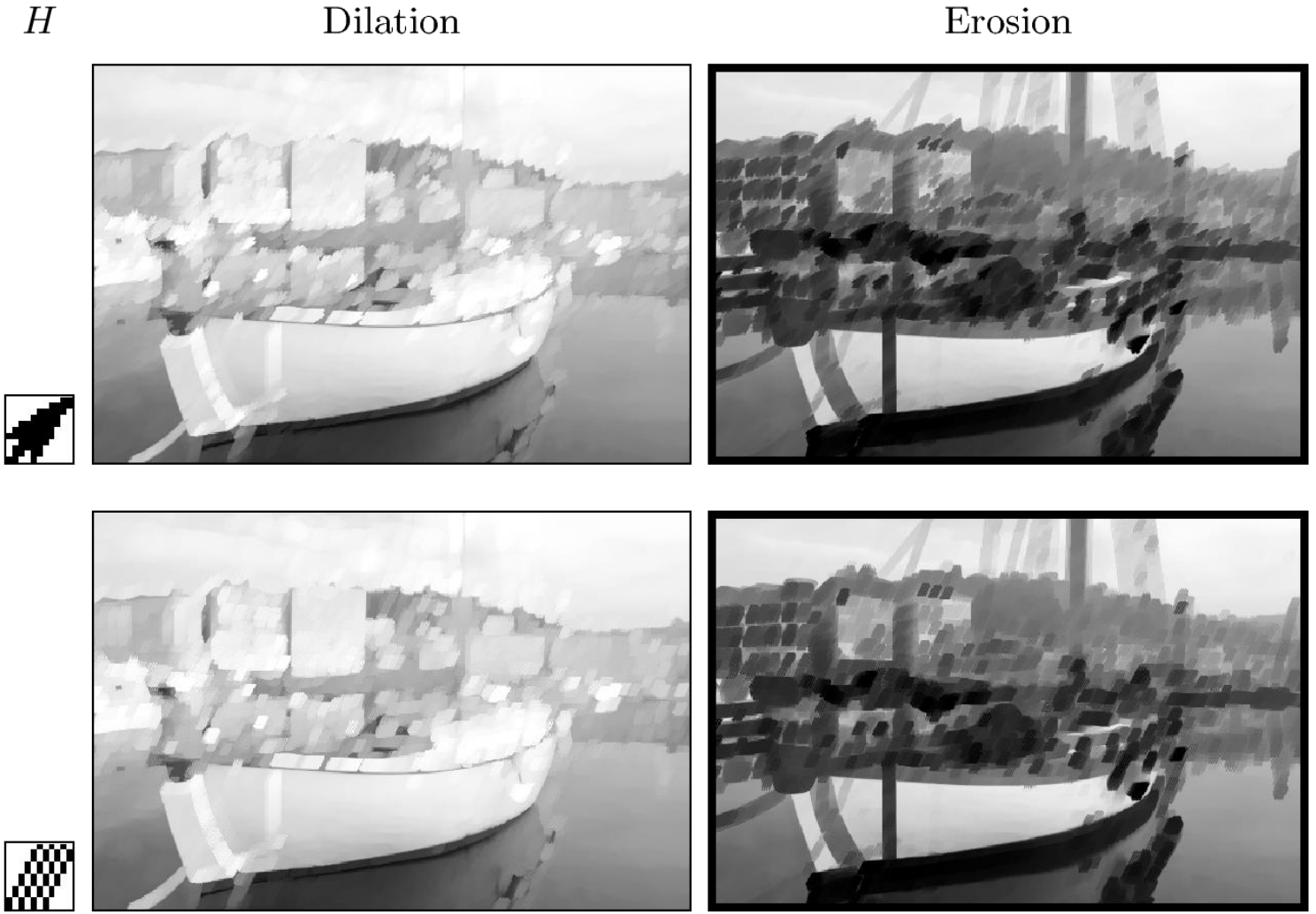
[Burger & Burge]

Example

Small Disk

[Burger & Burge]

Example



[Burger & Burge]

Example

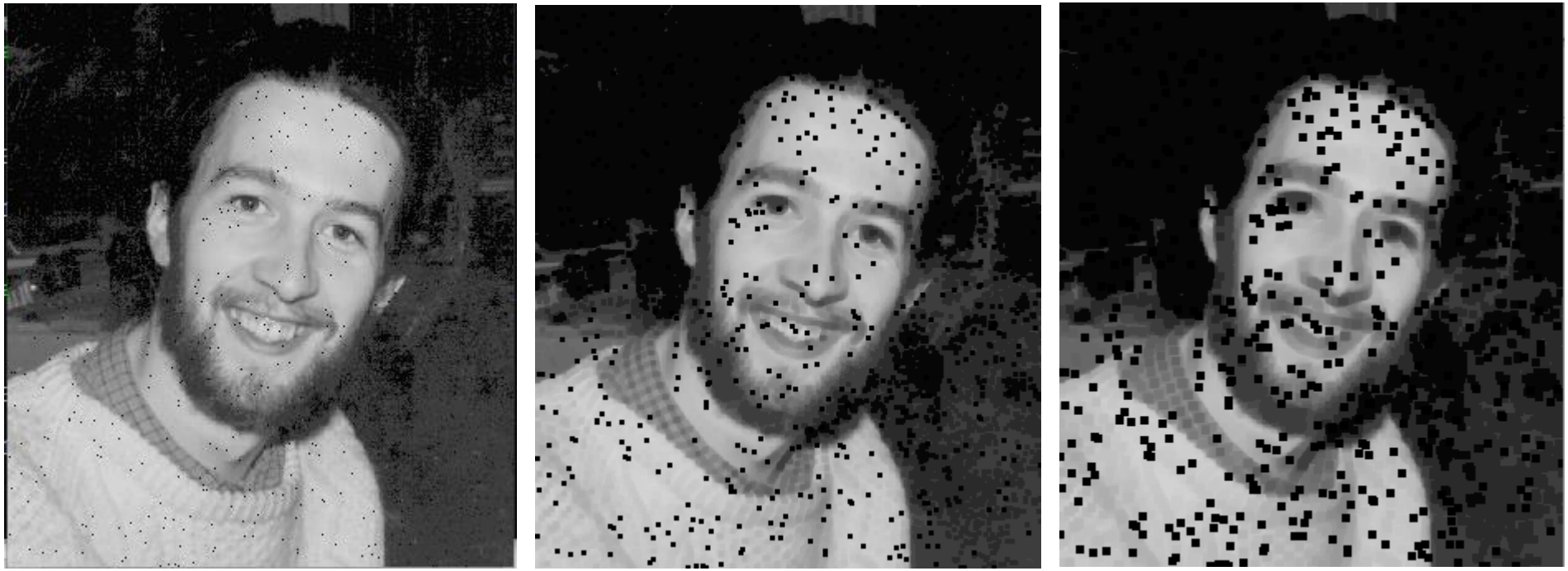
[Burger & Burge]

Example – “Pepper-Noise”



Example – “Salt-Noise”

Example – “Pepper-Noise”

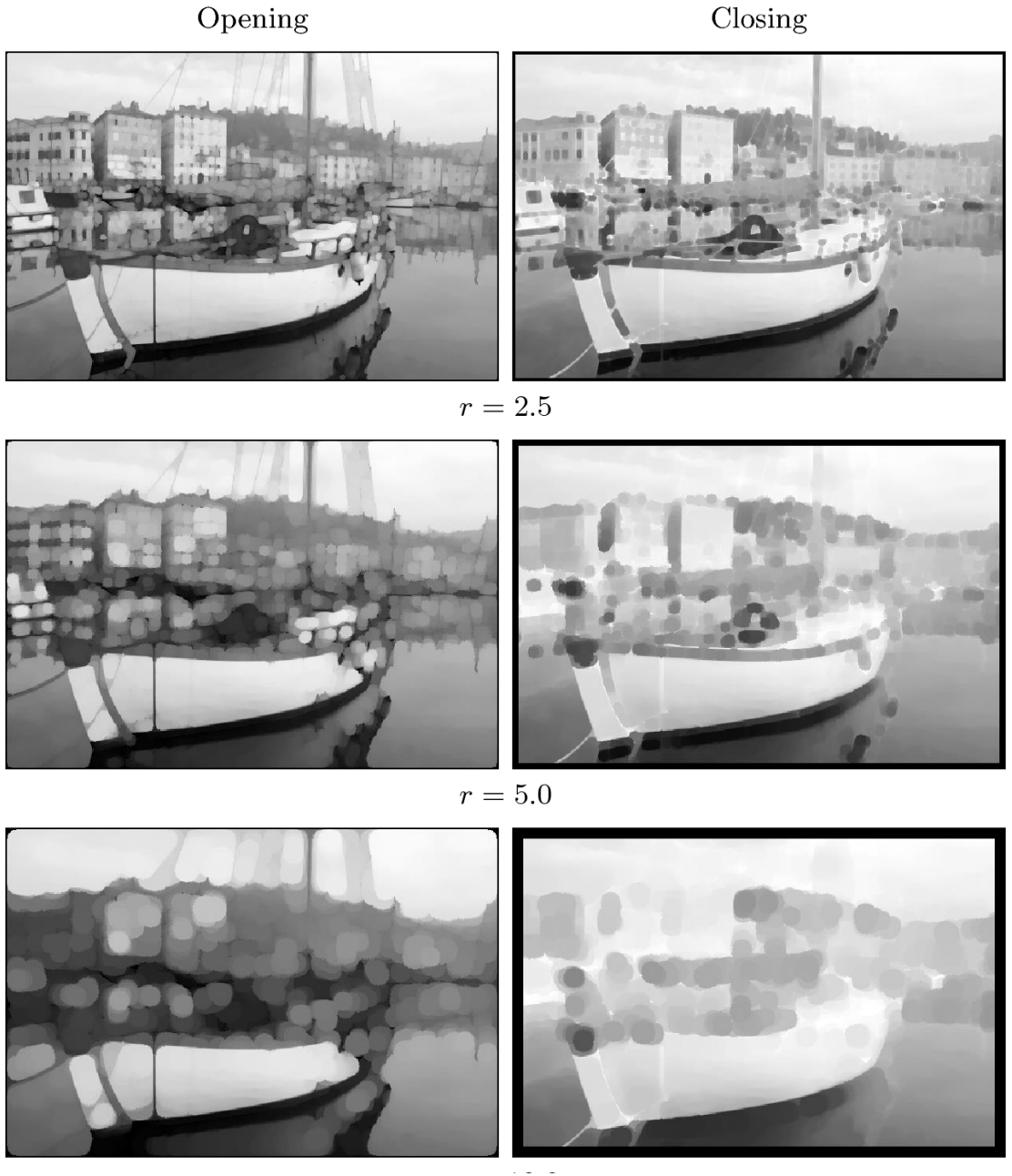


Example – “Salt-Noise”



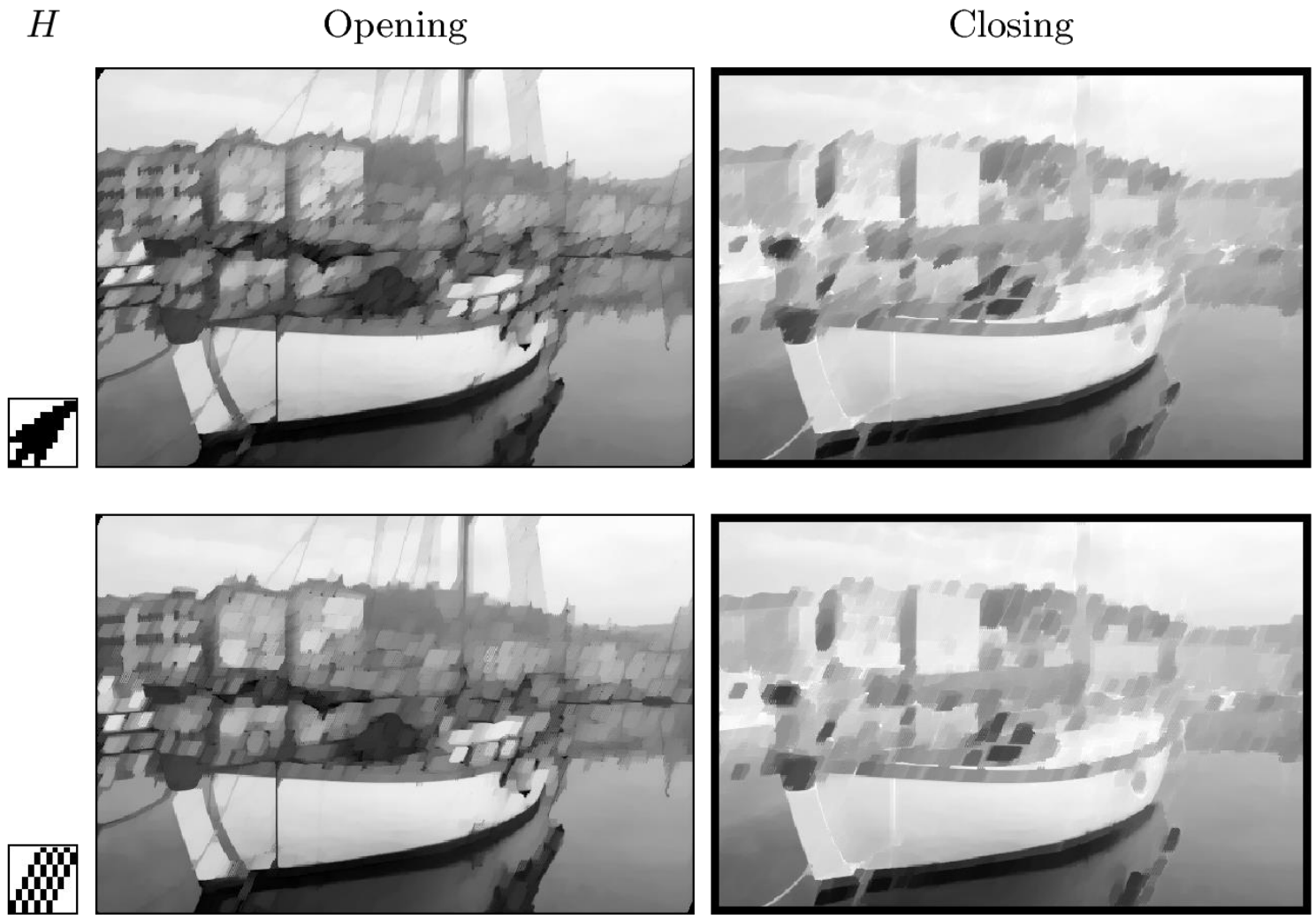
Example

Small Disk



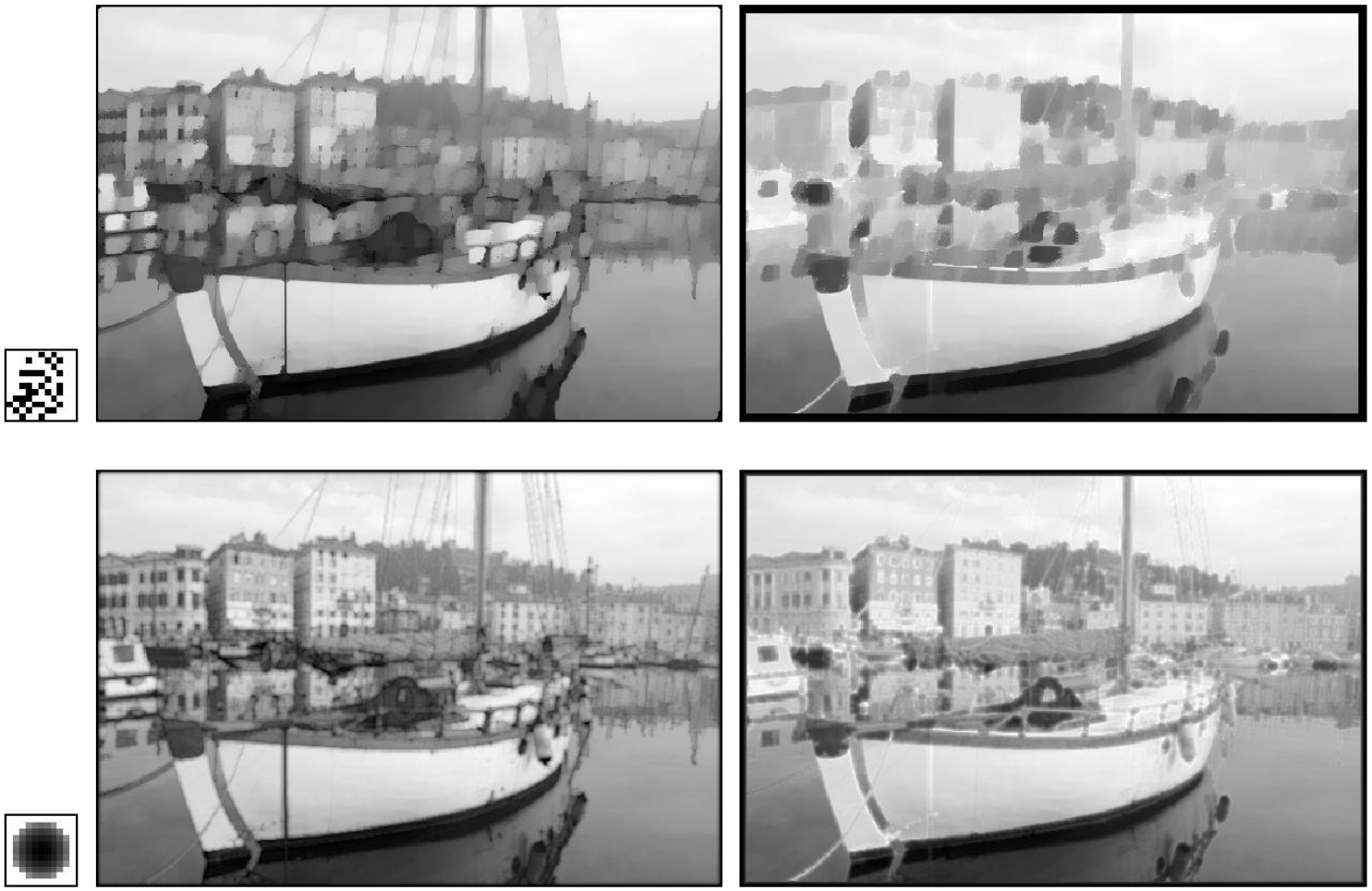
[Burger & Burge]

Example



[Burger & Burge]

Example



[Burger & Burge]

Example – “Salt-and-Pepper-Noise”

REFERENCES

References

- W. Burger, M. J. Burge, Principles of Digital Image Processing, Vol.1 and Vol. 2, Springer, 2009
- A. Watt, F. Policarpo, The Computer Image, Addison Wesley, 1998
- R. Gonzalez, R. Woods, Digital Image Processing, 2nd Ed., Addison Wesley, 2002

Acknowledgments

To Professors Augusto Silva and Paulo Dias, and to PhD students.

Example of the first slide: http://spie.org/x8899.xml?ArticleID=x8899

Junqing Chen and Thrasyvoulos Pappas, “Adaptive perceptual color-texture image segmentation” SPIE - Electronic Imaging & Signal Processing, 2006 DOI: 10.1117/2.1200602.0016