Computer Vision and Machine Learning

(Image features)

Bhabatosh Chanda

bchanda57@gmail.com

Introduction

- Feature of an object or a scene is a representative attribute or property of the same.
- A set of features is some kind of description of the object or scene.
- Features should be **descriptive** as well as **distinctive**.
- Features should be **non-redundant** and **non-derogatory.**
- Most common visual features are:
  - Shape
  - Texture
  - Colour

Introduction

Features are also termed as *attributes/properties*

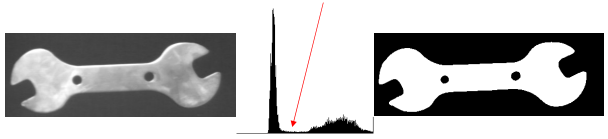
- Shape feature
  - Structural features
  - Moment features
  - Geometrical
- Topological features
- Textural features
- Colour features

Region extraction

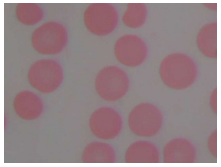
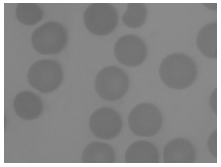
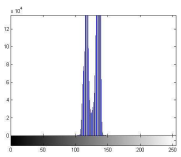
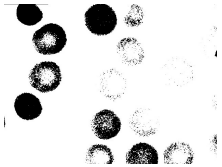
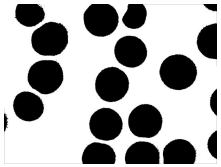
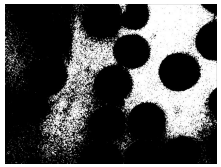
- Pixel classification approach using local properties
- Graylevel thresholding:

$$b(r,c) = \begin{cases} 1 & \text{if } g(r,c) > th \\ 0 & \text{otherwise} \end{cases}$$

th is the threshold value

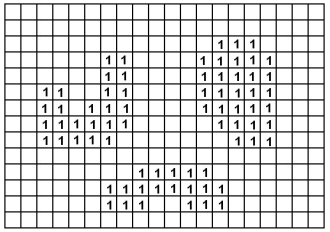


Graylevel thresholding: Example

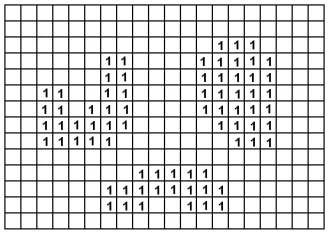
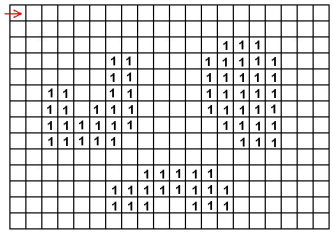
Th = 115 Th = 125 Th = 135

Component labelling

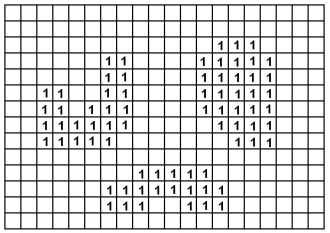
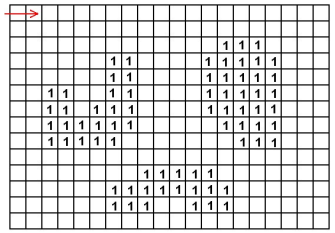


- Scan the image in raster order till a 1-pixel found
- Recursively assign a new label to this pixel and each pixel adjacent to it.
- If bottom-right pixel is not reached, go to first step.

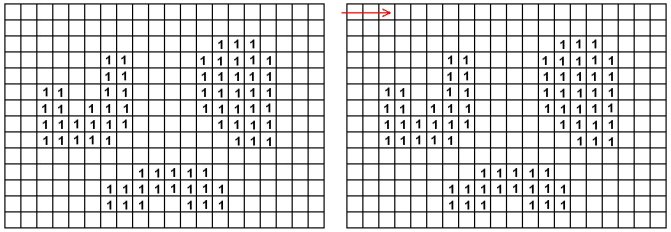
Component labeling: Example

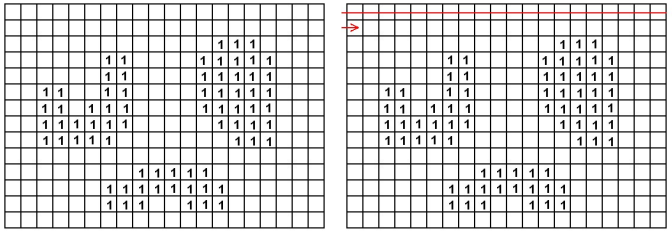
Component labeling: Example

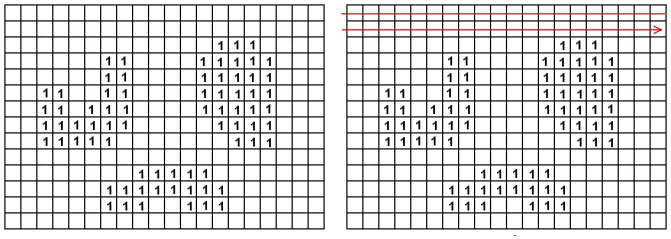
Component labeling: Example



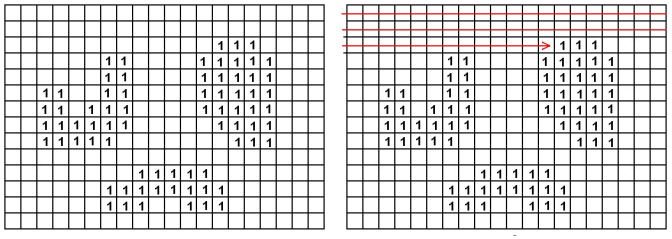
Component labeling: Example



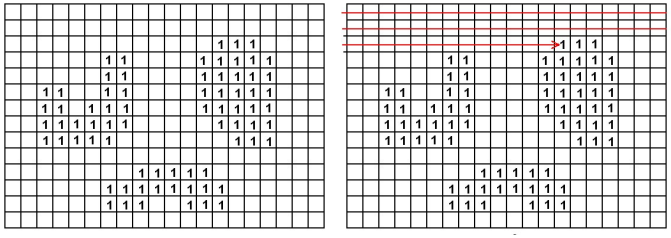
Component labeling: Example



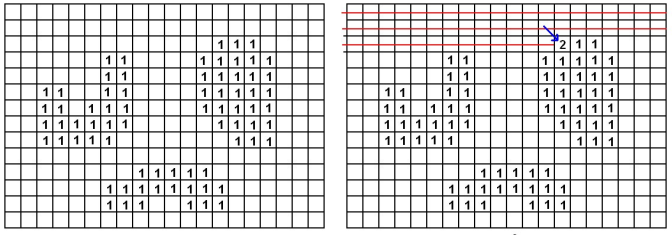
Component labeling: Example



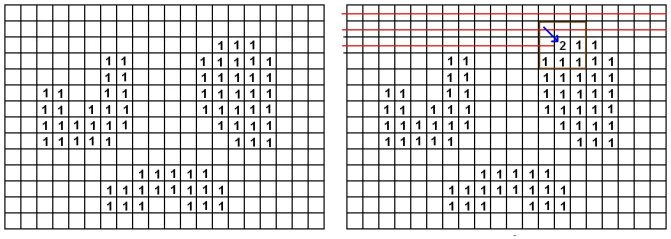
Component labeling: Example



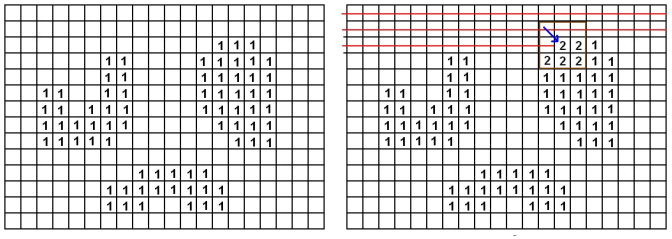
Component labeling: Example



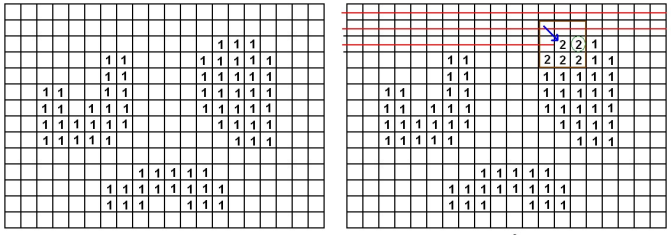
Component labeling: Example



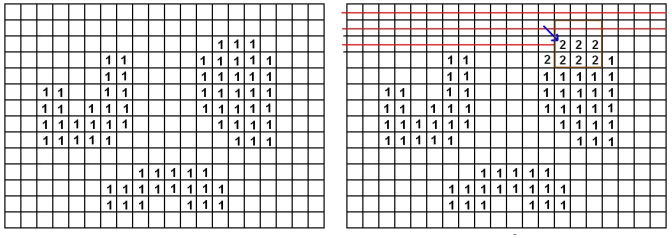
Component labeling: Example



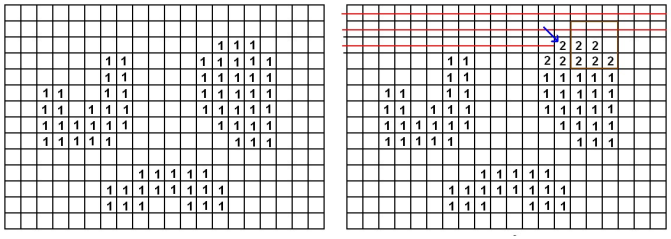
Component labeling: Example



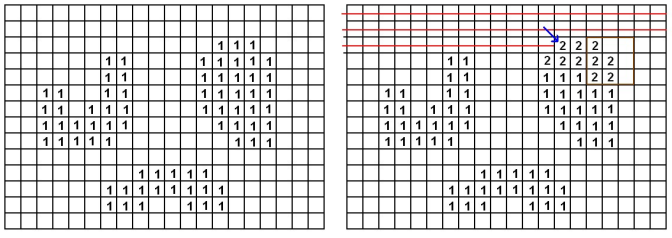
Component labeling: Example



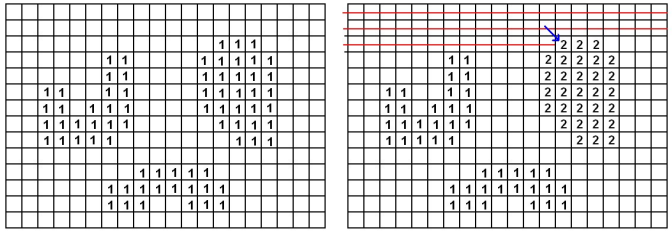
Component labeling: Example



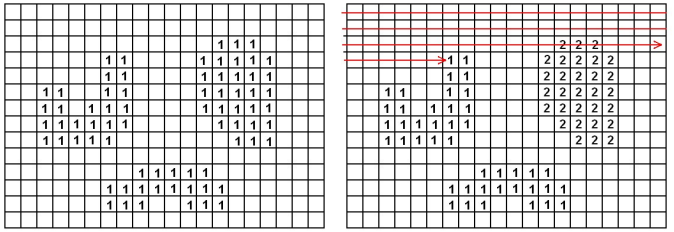
Component labeling: Example



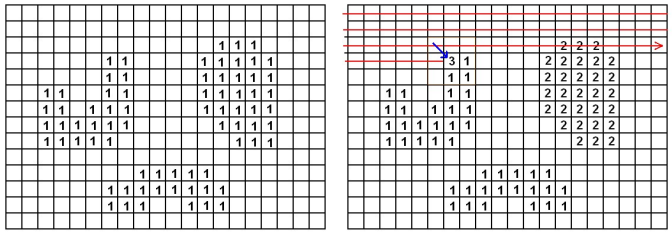
Component labeling: Example



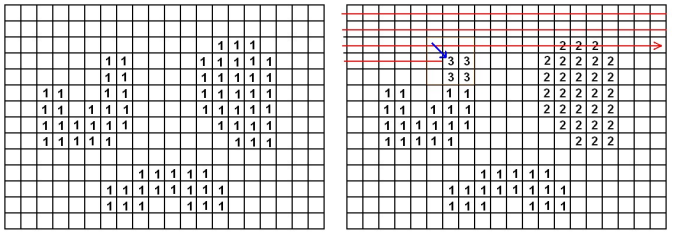
Component labeling: Example



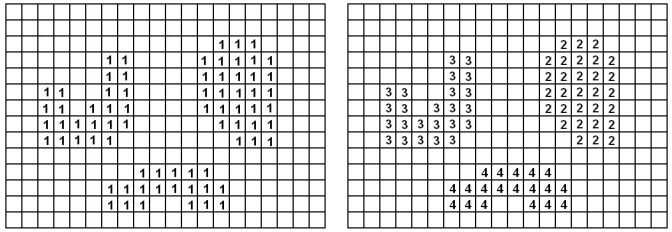
Component labeling: Example



Component labeling: Example



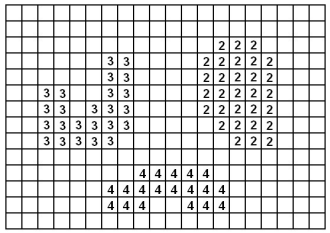
Component labeling: Example



Component labeling: Two-pass algorithm

- Flood-fill type algorithm is simple to understand
  - Implemented by using Queue data structure
  - Problem: Stack overflow in case of large components
- **Two-pass algorithm –** an efficient algorithm which is free from this problem
  - May be found *in Section 13.3.2 of "*Digital Image Processing and Analysis*", B. Chanda and D. Dutta Majumder,* PHI Learning, New Delhi, 2011.

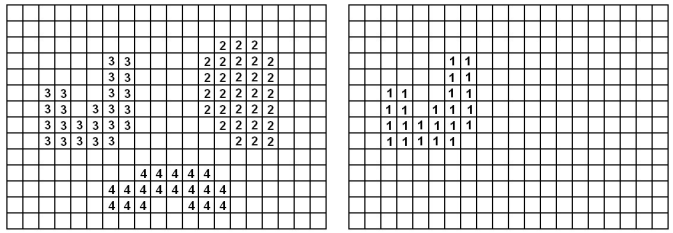
Component Counting: Example



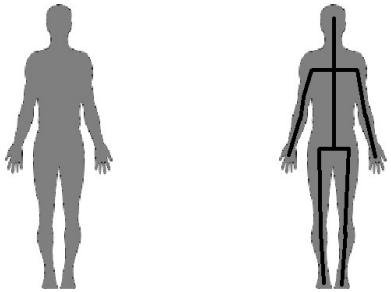
- Number of distinct labels indicates number of components/objects.
- Number of pixels having a unique label is the size of that component.
- Here no. of distinct label is 3 (2, 3 and 4)
- The sizes: 30, 24 and 19

Component extracting: Example

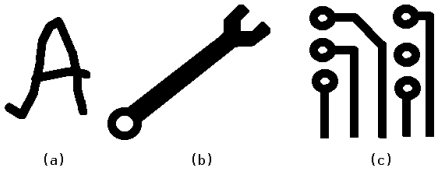
Component extracting FILTER: $b(i,j) = \begin{cases} 1 & \text{if } label(i,j) = 3 \\ 0 & \text{otherwise} \end{cases}$



Structural properties



Structural property

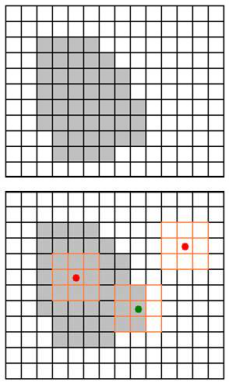


Thinning: **Search and delete process**

Removes (from the object) only those ***boundary*** pixels whose deletion

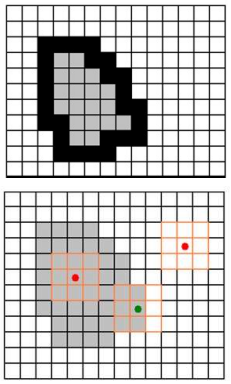
1. does not change connectivity locally
2. does not reduce the length of already thinned curve

Boundary detection



- Boundary or border is a connected set of pixels of object, which is in the neighbourhood of background.
- Check neighbourhood of every object pixel, if there is a background pixel then object pixel is a boundary pixel.

Boundary detection

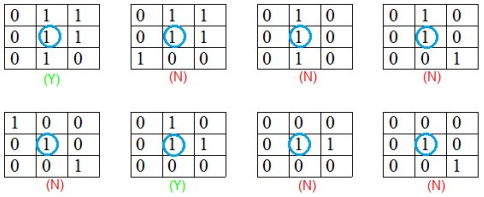


- Boundary or border is a connected set of pixels of object, which is in the neighbourhood of background.
- Check neighbourhood of every object pixel, if there is a background pixel then object pixel is a boundary pixel.

Boundary removal

Boundary pixels may be removed (from north or east or south or west at a time) **provided**

- Removal does not disconnect the object.
- Removal does not shorten the curve.



(Considering 8-connecteivity)

Thinning: steps

Original image **After removal of outer layer**

Thinning: steps

Original image **After removal of outer layer**

Thinning: steps

Original image **After removal of outer layer**

Thinning: steps

Original image **After removal of outer layer**

Thinning: steps

Original image **After removal of outer layer**

Thinning: steps

Original image **After removal of outer layer**

Thinning: steps

Original image **After removal of outer layer**

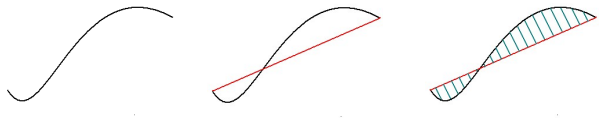
Thinning: steps

Original image **Final output (skeleton)**

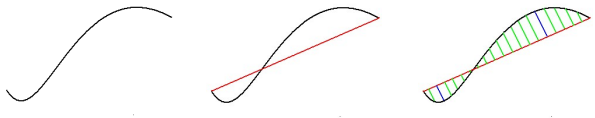
Polygonal approximation

- **Objective:** Representing object boundary in a simpler way.
- **Advantage:** Compaction of boundary information.
- **Method:** Representing the curve line by a straight line segment

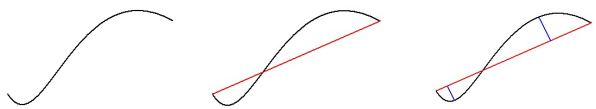
Polygonal approximation



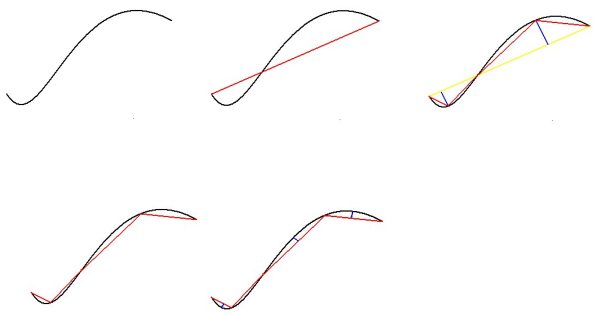
Polygonal approximation



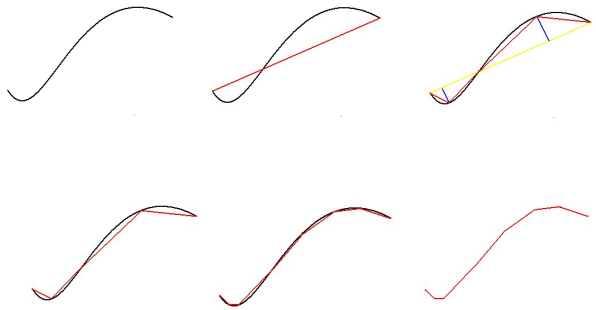
Polygonal approximation



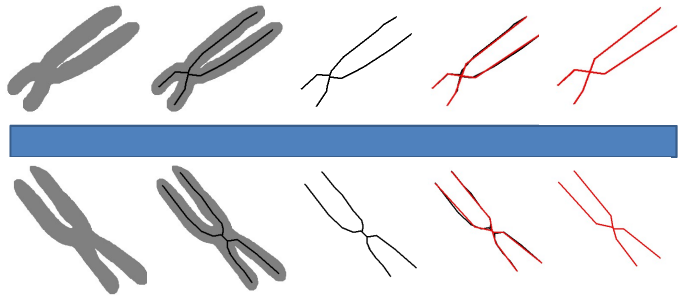
Polygonal approximation



Polygonal approximation



Skeleton: Example



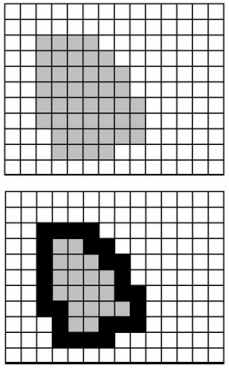
Structural features

- **Structural features that may be extracted:**
  - Number of strokes
  - Length of the strokes
  - Slopes of the strokes (horizontal, vertical, slant, etc.)
  - Number of loop
  - So on …

Distance transform

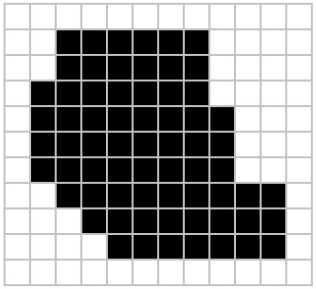
- Computes distance of an object pixel from background.
- It transforms a binary image to a multilevel image where interior pixels have *progressively* higher value than the boundary pixels.
  - So it provides shape information.
- Original object can be reconstructed from the pixels with *locally maximum value* of distance transform.
- These pixels (with *locally maximum value*) constitute the *skeleton* or *medial axis* of the object.

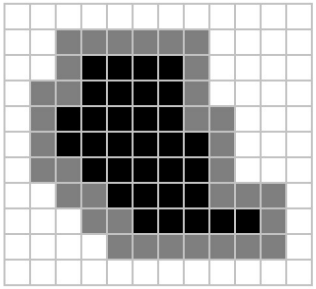
Boundary detection



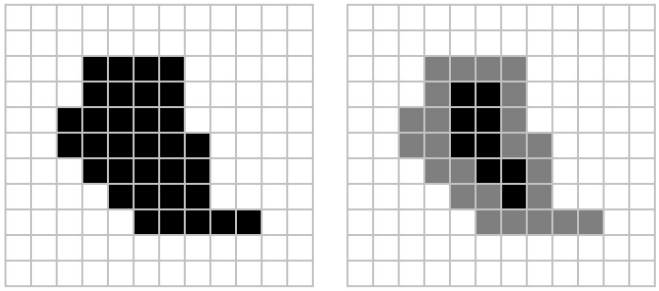
- Boundary or border is a connected set of pixels of object, which is in the neighbourhood of background.
- Check neighbourhood of every object pixel, if there is a background pixel then object pixel is a boundary pixel.

Distance transform computation

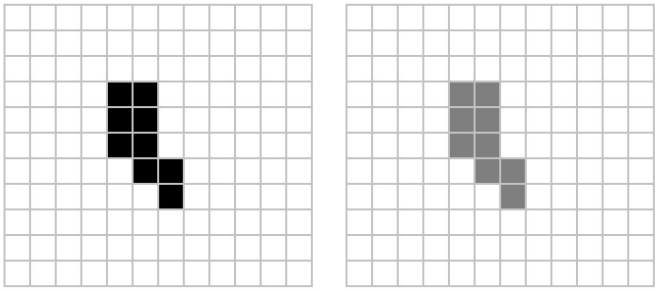




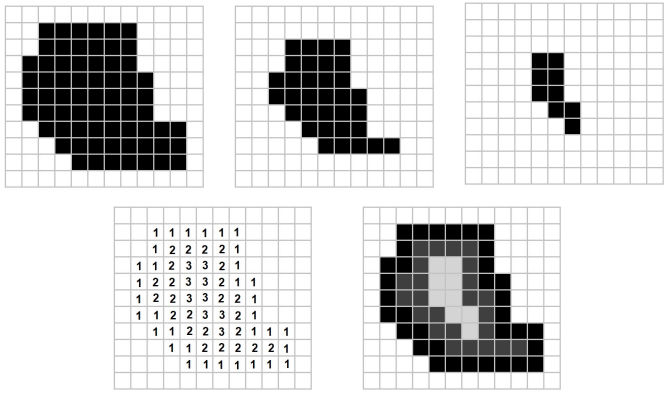
Distance transform computation



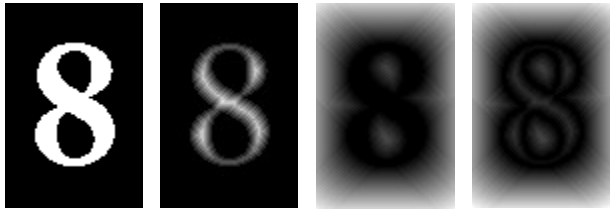
Distance transform computation



Distance transform computation



Distance transform: Example



(a) Binary image

(b) Distance transform of object pixels

(c) Distance transform of background pixels

(d) Combination of (b) and (c)

Distance transform: Two-pass algorithm

- A **Two-pass algorithm** starting from top-left corner may be used to compute the distance transform (*DT*)
  - The algorithm may be found *in pages 368-370 of "*Digital Image Processing and Analysis*", B. Chanda and D. Dutta Majumder,* PHI Learning, New Delhi, 2011.
- Local maxima of *DT(x,y)* constitute the *medial axis* (MA) of the component.
  - MA(x,y) is equivalent to skeleton that has obtained by thinning.
  - Points in MA(x,y) may not be connected. However, from MA(x,y) original component may be reconstructed.

Moment features

The graylevel at a pixel is considered as point mass at at that point.

$(i,j)-\text{th}$ moment of the image $f(r,c)$ is given by

$$m(i,j)=\sum_r\sum_c r^i c^j f(r,c)$$

$m(0,0)$ is the mass (total intensity values) of object (image).

$(\bar{m}_r,\bar{m}_c)=\left(\frac{m(1,0)}{m(0,0)},\frac{m(0,1)}{m(0,0)}\right)$ is the centroid of the object.

Moment features

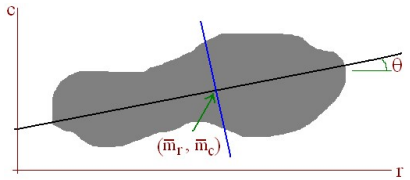
Central moment =

$$\bar{m}(i,j)=\sum_r\sum_c (r-\overline{m_r})^i(c-\overline{m_c})^j f(r,c)$$

$\bar{m}(i,j)$ is the measure of **symmetry** if $(i+j)$ is odd.

$\bar{m}(2.0)+\bar{m}(0,2)$ is the measure of **inertia** (rotation invariant)

Moment features



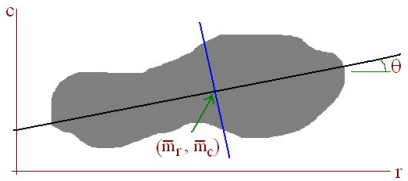
Principal or symmetric axis of the object passes through (α,β) and has slope θ, then

Equation: $(r-\alpha)sin\theta-(c-\beta)cos\theta=0$

Inertia: $I(\alpha,\beta,\theta)=\sum_{(r,c)}[(r-\alpha)sin\theta-(c-\beta)cos\theta]^2 f(r,c)$

Moment features

$(\bar{m}_r, \bar{m}_c)$



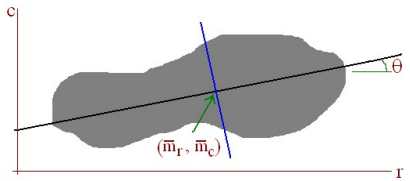
Principal or symmetric axis of the object

passes through $(\alpha = \bar{m}_r, \beta = \bar{m}_c)$ and has slope θ where

Equation: $(r - \bar{m}_r)sin\theta - (c - \bar{m}_c)cos\theta = 0$

Inertia: $I(\bar{m}_r, \bar{m}_c, \theta) = \sum_{(r,c)}[(r - \bar{m}_r)sin\theta - (c - \bar{m}_c)cos\theta]^2 f(r,c)$

Moment features

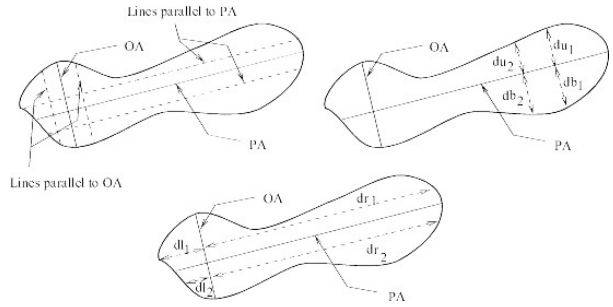


Principal or symmetric axis of the object

passes through $(\alpha = \bar{m}_r, \beta = \bar{m}_c)$ and has slope θ where

$$\tan^2\theta + \frac{\bar{m}(2,0) - \bar{m}(0,2)}{\bar{m}(1,1)}\tan\theta - 1 = 0$$

Aspect ratio and symmetry:



PA = Principal axis, OA = Axis orthogonal to PA

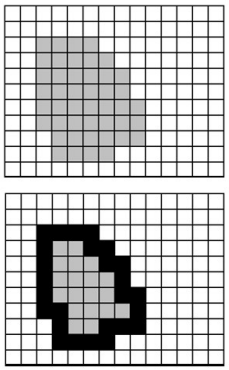
$$\text{Aspect ratio} = \frac{\max_i\{du_i\} + \max_i\{db_i\}}{\max_i\{dl_i\} + \max_i\{dr_i\}}$$

$$\text{Symmetry} = avg_i\{\frac{\min\{du_i, db_i\}}{\max\{du_i, db_i\}}\}$$

Value of ***Symmetry*** would be 1 for symmetric object and reduce for asymmetric objects.

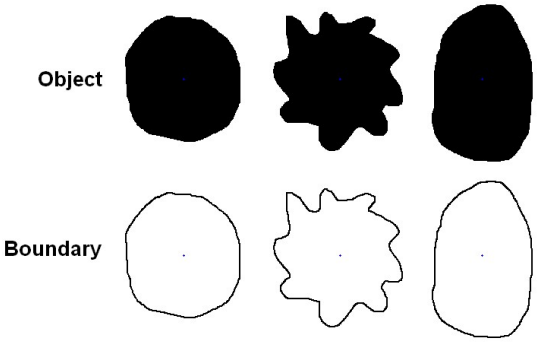
$$\text{Thinness} = \frac{avg_i\{du_i + db_i\}}{\max_i\{dl_i\} + \max_i\{dr_i\}}$$

Boundary detection

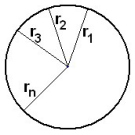


- Boundary or border is a connected set of pixels of object, which is in the neighbourhood of background.
- Check neighbourhood of every object pixel, if there is a background pixel then object pixel is a boundary pixel.

Boundary detection: Example



Geometric property: Circularity

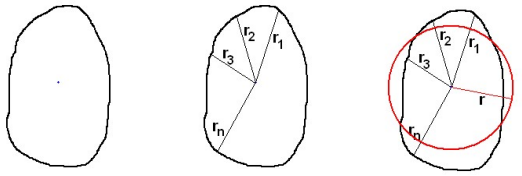


Average radius $r = \frac{1}{N}\sum_{n=1}^{N} r_n$

Max. difference in radii $d_r = \max_n\{r_n\} - \min_n\{r_n\}$

Note: In case of circle max. diff. in radii is zero.

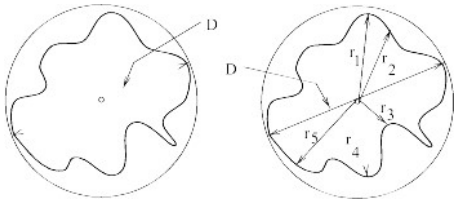
Circularity property: Example



$$\text{Pr } 1 = \frac{2|\text{Area of object} - \text{Area of circle of radius r}|}{\text{Area of object} + \text{Area of circle of radius r}}$$

$$\text{Pr } 2 = \frac{\text{Area of circle with } r_{max} - \text{Area of circle with } r_{min}}{\text{Area of circle of radius } r_{avg}}$$

Circularity property: Example

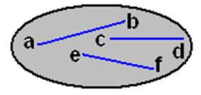


Minimum bounding circle (MBC) with centre at CG of the object.

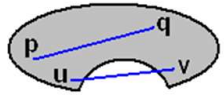
$$\text{Pr } 3 = \frac{\text{Area of object}}{\text{Area of MBC}} \qquad \text{Pr } 4 = \frac{\text{Area of circle with } r_{min}}{\text{Area of the MBC}}$$

Geometric property: Convexity

- An object is said to be a ***convex object***, if the straight line joining every pair of points of the object lies in the object.
- If the object is not convex, it is called ***concave object***.



Convex object

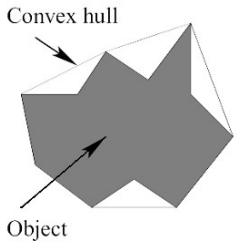


Concave object

Convexity measure

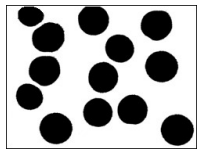
- ***Convex hull*** is a smallest filled polygon covering the object.
- Convexity of an object may be measured as

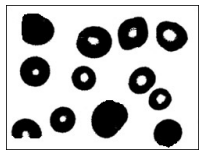
$$\text{Pr } 5 = \frac{\text{Area of the object}}{\text{Area of the convex hull}}$$



Topological property

- Euler number or genus
  = no. of objects – no. of holes
- Invariant under rubber-sheet transformation
- Examples:





Euler number= 13 4

Thank you !

Any question?

Square Grid