Image/ Object Feature extraction

Texture of image

- General Textures are rough, silky, bumpy
- Image with
  - rough texture has a large difference between pixel intensities
  - smooth texture has little difference between pixel intensities
- Textures in images quantify Grey Level differences
- Second order texture features consider the relationship between groups of two pixels in the original image
- First order texture measures are statistics of pixel values, like variance

  and do not consider pixel relationships

rough

smooth

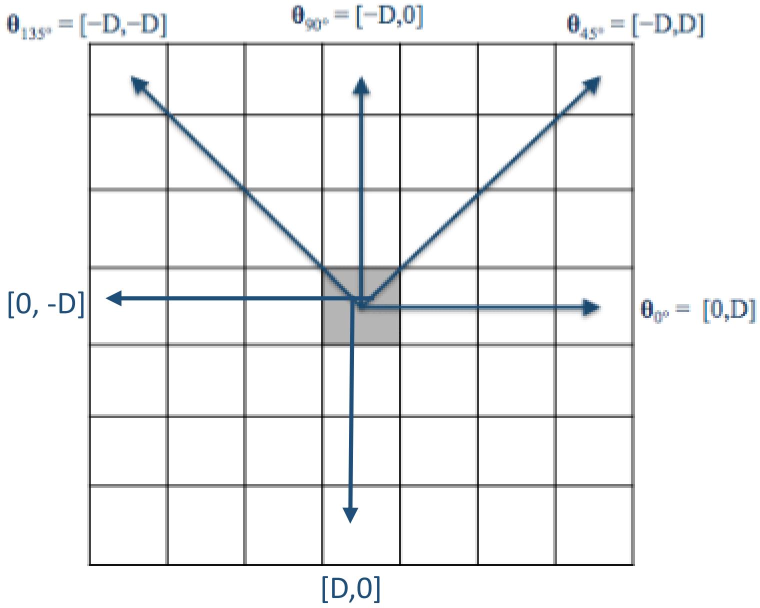
gray-level co-occurrence matrix (GLCM)

- Statistical method of examining texture
- Considers the spatial relationship of pixels
- Shows how often pairs of pixel with specific values and with specified relationship occur in an image
- Statistical measures can be extracted from this matrix
- Statistics provide information about the texture of an image

Create GLCM

- Is a tabulation of how often different combinations of pixel brightness values (grey levels) occur in an image
- Calculate how often a pixel with the intensity (gray-level) value *i* occurs in a specific spatial relationship to a pixel with the value *j*
- Example of Spatial relationship is pixel of interest and the pixel to its immediate right (horizontally adjacent)
- Each element is the sum of the number of times pixels with value, i occurred in specified relationship to a pixel with value, j in the image
- Number of gray levels in the image determines the size of GLCM
- If image has 8 levels, GLCM has the size of 8×8
- GLCM reveals certain properties about the spatial distribution of the gray levels in the texture image

GLCM example



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

2-bit Image matrix

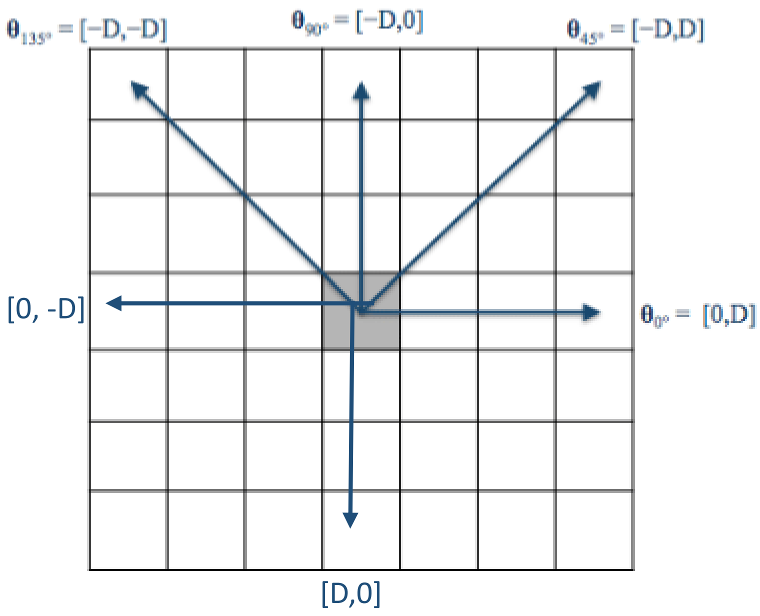
Pixel values, j →

Pixel values, i →

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

GLCM for (0,1) spatial relationship

GLCM example



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

2-bit Image matrix

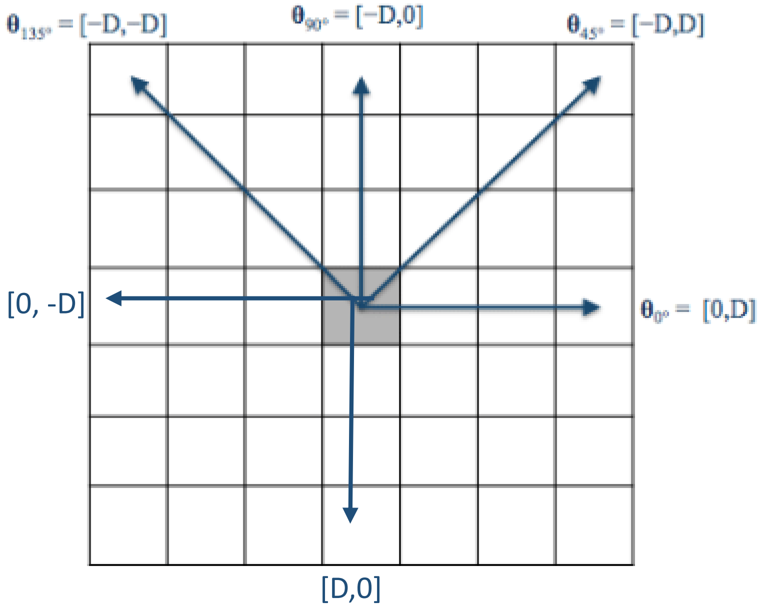
Pixel values, j →

Pixel values, i →

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

GLCM for (0,1) spatial relationship

GLCM example



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

2-bit Image matrix

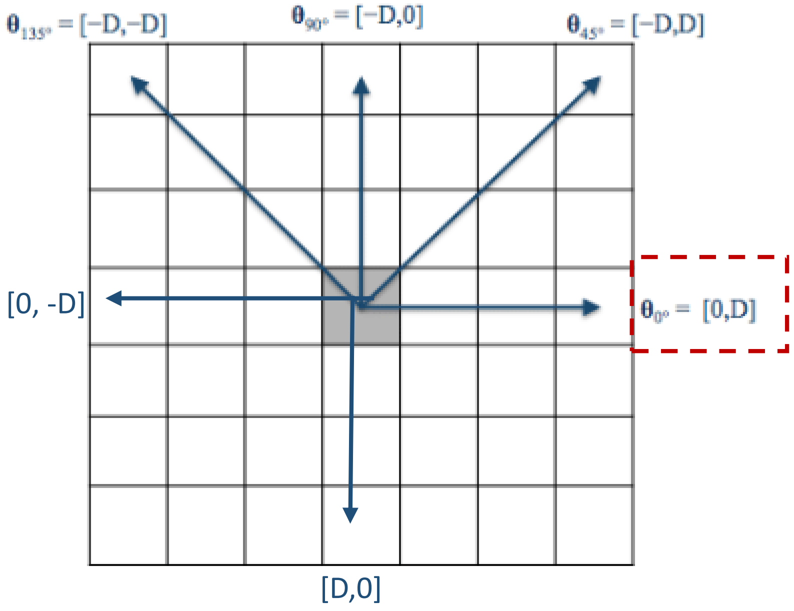
Pixel values, j →

Pixel values, i →

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

GLCM for (0,1) spatial relationship

GLCM example



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

2-bit Image matrix

Pixel values, j →

Pixel values, i →

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

GLCM for (0,1) spatial relationship

- (0,1) relation- target pixel is 1 unit on away on right side of reference pixel
- Other relationships like (1,0), (0,-2) etc are possible
- For each relationship one GLCM is constructed

Features of GLCM

- Square
  - Reference pixels have the same range of possible values as the neighbor pixels
  - Range of row and column numbers is same
- Number of rows and columns is number of level of the image
  - For 3-bit image, it has 8 rows and 8 columns
  - It is too large to for computation.
  - For 8-bit data, image is scaled to 4 bit (16 x 16 matrix) or 5 bit (32x32 matrix)
  - Also, there are many 0s because spatial relationship may not have pairs
  - Since GLCM works on joint probability distribution, result may skewed

Features of GLCM

- Symmetrical matrix around diagonal
  - GLCM is not symmetric
  - Texture calculations are best performed on a symmetrical matrix
  - Symmetry will be achieved if each pixel pair is counted twice: once "forwards" and once "backwards“
  - This operation would cover border pixels

Features of GLCM

- If image contains objects of a variety of shapes and sizes and they are arranged in horizontal and vertical directions

  use offsets of varying direction and distance

- Spatial relationship can be in any direction and any distance between the pixels of interest

GLCM example

- Pixel in the last column do not have neighbour pixels
- Transpose changes the spatial relationship
- Add original and transposed GLCM
- Addition generates almost symmetric matrix
- Also, border pixels are also counted in GLCM

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

2-bit Image matrix

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

Transpose GLCM

Pixel values, j →

Pixel values, i →

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

GLCM for (0, 1) spatial relationship

4	2	1	0
2	4	0	0
1	0	6	1
0	0	1	2

Symmetric GLCM

Features of GLCM

- Lines parallel to the diagonal are one cell away from the diagonal
- These lines represent pixel pairs with a difference of one grey level (0-1, 1-2, 2-3 etc.)
- Similarly, values in two cells away from the diagonal show how many pixels have intensity difference of 2
- The farther away from the diagonal, the greater the difference between pixel grey levels

Normalized Symmetric GLCM

- Normalized matrix shows probability of occurrence of each pair of pixels
- sum of elements = 24
- Normalized symmetric GLCM = element of symmetric GLCM/24
- The diagonal elements represent pixel pairs with no grey level difference
- If there are high probabilities in these elements, then the image has low contrast
- That is most of the pixels are identical to their neighbours

4	2	1	0
2	4	0	0
1	0	6	1
0	0	1	2

Symmetric GLCM

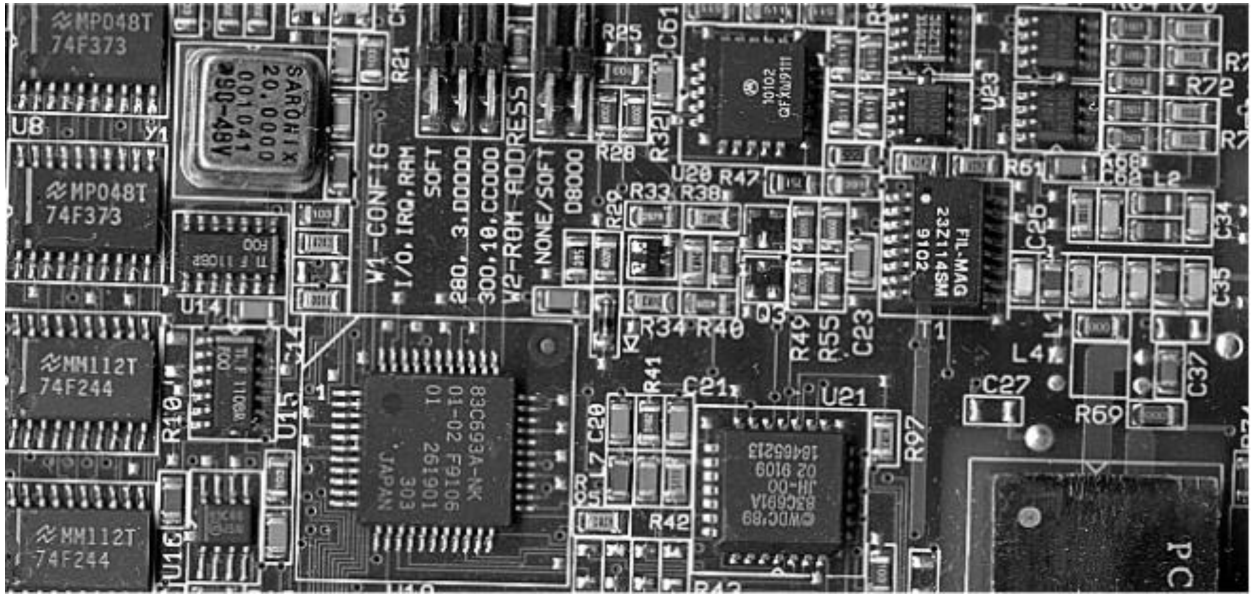
0.166	0.083	0.042	0
0.083	0.166	0	0
0.042	0	0.25	0.042
0	0	0.042	0.083

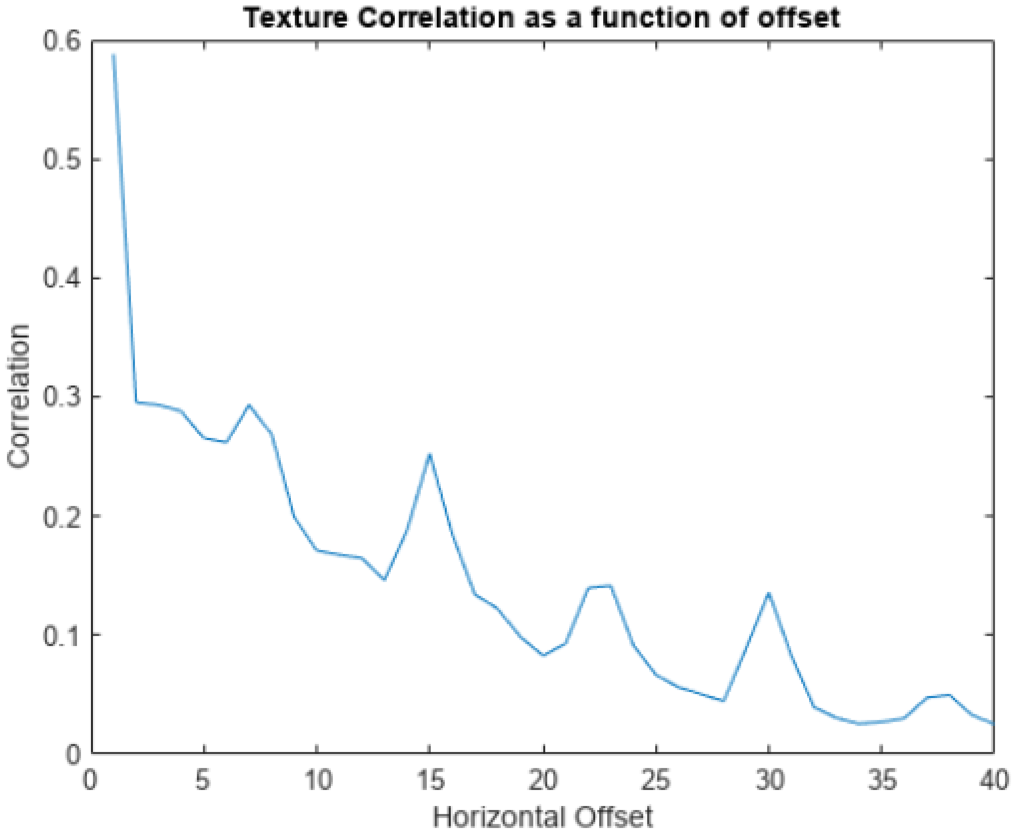
Normalized GLCM, Probability, p(i,j)

Application of texture image

- In remote sensing and medical imaging, value of a pixel is not important
- Pixel-to-pixel relationships is more important
- Ex: Forest can be smooth or variety of trees
- Generate GLCM for a small area of the image and determine texture features

Texture feature (correlation)





Statistics derived from Example GLCM

GLCM

j

i

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

- Contrast: 2.8947
- Dissimilarity: 1.31
- Energy: 0.1191
- Homogeneity: 0.565

Contrast of image derived from GLCM

- Also known as sum of squares variance or inertia
- Uses weights related to the *distance from the GLCM diagonal*
- Returns a intensity contrast between a pixel and its neighbor over the image
- Values on the GLCM diagonal show no contrast
- Contrast is more for pixels which are away from the diagonal
- Range is from 0 to the number of elements of GLCM
- If contrast = 0, image has pixels with constant intensity
- Increases exponentially as intensity difference between reference and target pixel increases
- Contrast is defined for a specific relationship used for GLCM
- For (0,1) relationship, it shows contrast in horizontal direction and with next pixel

Contrast derived from Example GLCM

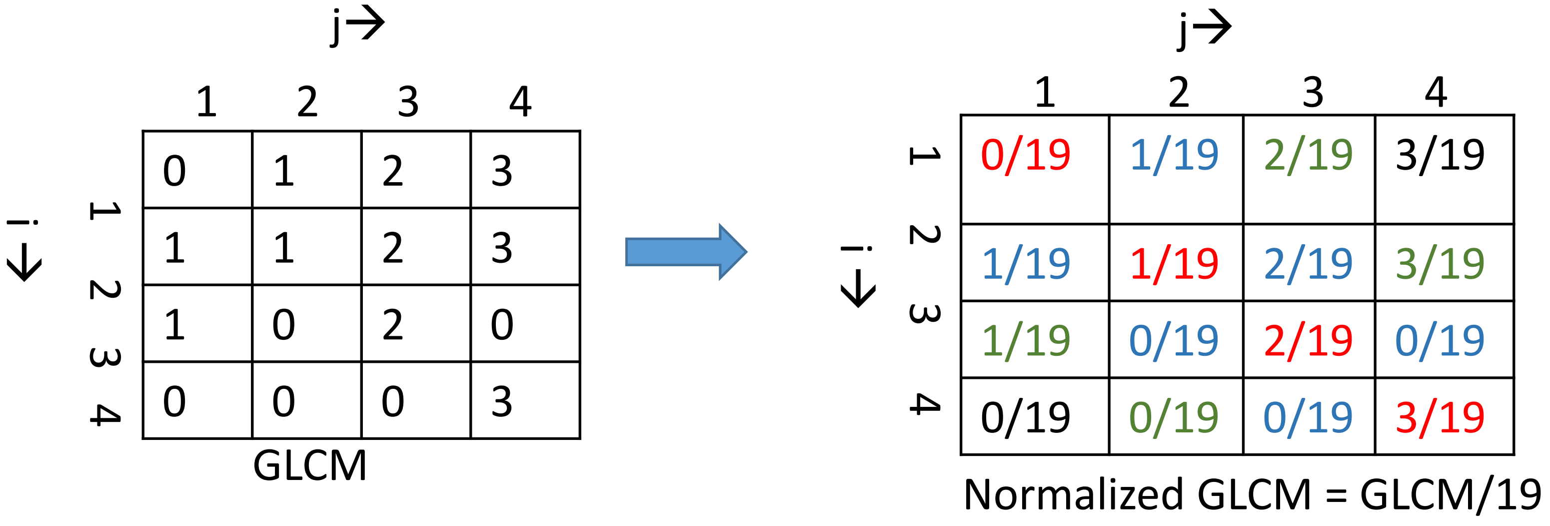
j→

i→	1	2	3	4
1	0	1	2	3
2	1	1	2	3
3	1	0	2	0
4	0	0	0	3

GLCM

Sum of elements of GLCM = 19

Contrast derived from Example GLCM



i \ j	1	2	3	4
1	0	1	2	3
2	1	1	2	3
3	1	0	2	0
4	0	0	0	3

GLCM

Sum of elements of GLCM = 19

i \ j	1	2	3	4
1	0/19	1/19	2/19	3/19
2	1/19	1/19	2/19	3/19
3	1/19	0/19	2/19	0/19
4	0/19	0/19	0/19	3/19

Normalized GLCM = GLCM/19

- Elements of normalized GLCM = Probability, p(i,j)
- Red elements show pairs of pixels with same value
- Blue elements show pairs of pixels with difference in pixel value of 1
- Green elements show pairs of pixels with difference in pixel value of 2

Contrast derived from Example GLCM

j→

i↓

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

GLCM

Sum of elements of GLCM = 19

j→

i↓

	1	2	3	4
1	0/19	1/19	2/19	3/19
2	1/19	1/19	2/19	3/19
3	1/19	0/19	2/19	0/19
4	0/19	0/19	0/19	3/19

Normalized GLCM = GLCM/19

Elements of normalized GLCM = Probability, p(i,j)

$$Contrast = \sum_{i,j} |i-j|^2 p(i,j)$$

- Contrast = $(1/19) + 2^2 \times (1/19) + (1/19) + 2^2 \times (2/19) + (2/19) + 3^2 \times (3/19) + 2^2 \times (3/19)$
- Contrast: 2.8947

Same procedure can be applied to symmetrical GLCM

Dissimilarity derived from Example GLCM

j→

i→

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

GLCM

Sum of elements of GLCM = 19

j→

i→

	1	2	3	4
1	0/19	1/19	2/19	3/19
2	1/19	1/19	2/19	3/19
3	1/19	0/19	2/19	0/19
4	0/19	0/19	0/19	3/19

Normalized GLCM = GLCM/19

$$dissimilarity = \sum_{i,j} |i - j| p(i,j)$$

- Dissimilarity increases linearly with pixel difference

Contrast = (1/19) + 2×(1/19) + (-1)(1/19) + (-2)×(2/19)+ (-1)(2/19) + (-3)×(3/19) + (-2)×(3/19)

Same procedure can be applied to symmetrical GLCM

Energy derived from GLCM

- Returns the sum of squared elements in the GLCM
- Range is from 0 to 1
- Energy is 1 for a constant image
- The property Energy is also known as uniformity or angular second moment

$$Energy = \sum_{i,j} p(i,j)^{\,2}$$

Energy derived from Example GLCM

j→

i↓	1	2	3	4
1	0	1	2	3
2	1	1	2	3
3	1	0	2	0
4	0	0	0	3

GLCM

j→

i↓	1	2	3	4
1	0/19	1/19	2/19	3/19
2	1/19	1/19	2/19	3/19
3	1/19	0/19	2/19	0/19
4	0/19	0/19	0/19	3/19

Probability, p(i,j) = GLCM(i,j)/19

$$Energy = \sum_{i,j} p(i,j)^{\,2}$$

- Energy = $4(1/19)^2 + 3(2/19)^2 + 3(3/19)^2$

  = 0.1191

Same procedure can be applied to symmetrical GLCM

Homogeneity derived from GLCM

- Is inverse of the contrast
- Decreases exponentially if more pixels are away from the diagonal
- Returns a value that measures the closeness of the intensity distribution
- Range is from 0 to 1
- Homogenity is 1 for diagonal GLCM

$$homogeneity = \sum_{i,j} \frac{p(i,j)}{1 + |i - j|}$$

Homogeneity derived from Example GLCM

j→

i→

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

GLCM

j→

i→

	1	2	3	4
1	0/19	1/19	2/19	3/19
2	1/19	1/19	2/19	3/19
3	1/19	0/19	2/19	0/19
4	0/19	0/19	0/19	3/19

Probability, p(i,j) = GLCM(i,j)/19

$$homogeneity = \sum_{i,j} \frac{p(i,j)}{1+|i-j|}$$

homogeneity = (1/19)[(1/2+1/3+1/2+1)]+(2/19)[(1/3+1/2+1/1)]+(3/19)[(1/4+1/3+1/1)]

= (1/19)[0.5+0.33+0.5+1+0.66+1+2+0.75+1+3]

= 0.5652

Same procedure can be applied to symmetrical GLCM

Entropy derived from GLCM

i ↓

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

GLCM

i ↓

	1	2	3	4
1	0/19	1/19	2/19	3/19
2	1/19	1/19	2/19	3/19
3	1/19	0/19	2/19	0/19
4	0/19	0/19	0/19	3/19

Probability, p(i,j) = GLCM(i,j)/19

- If p(i,j) is small (i.e. occurrences of that pixel combination is low), entropy is large
- Measure of how pairs pixels with specific relationship are equally distributed

$$entropy = \sum_{i,j} -p(i,j)\ln(p(i,j))$$

entropy = -4(1/19)ln(1/19) - 3(2/19)ln(2/19) -3(3/19)ln(3/19)

Same procedure can be applied to symmetrical GLCM

GLCM mean, variance

- GLCM mean is mean of number of times a pixel's occurrence with a specific relationship

$$\mu_i = \sum_{i,j} -ip(i,j) \qquad \mu_j = \sum_{i,j} -jp(i,j)$$

- Variance is dependent on the mean, and the dispersion around the mean
- It is not variance of grey levels in the original image

$$\sigma_i^2 = \sum_{i,j} p(i,j)(i-\mu_i\ )^2 \qquad \sigma_j^2 = \sum_{i,j} p(i,j)(j-\mu_j\ )^2$$

Mean and Variance derived from Example GLCM

j→

i↓

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

GLCM

j→

i↓

	1	2	3	4
1	0/19	1/19	2/19	3/19
2	1/19	1/19	2/19	3/19
3	1/19	0/19	2/19	0/19
4	0/19	0/19	0/19	3/19

Probability, p(i,j) = GLCM(i,j)/19

$$\mu_i = \sum_{i,j} -ip(i,j)$$

$$\mu_j = \sum_{i,j} -jp(i,j)$$

μ_i = -1/19[(1+2+2+3)+2(1+2+3)+3(1+2)+4(3)]
= 1/19[6+14+9+12]
= 2.15

μ_j = 1/19[(1+1)+2(1+1)+3(2+2+2)+4(3+3+3)]
= 1/19[2+4+18+36]
= 3.16

Same procedure can be applied to symmetrical GLCM

Mean and Variance derived from Example GLCM

j→

i→

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

GLCM

j→

i→

	1	2	3	4
1	0/19	1/19	2/19	3/19
2	1/19	1/19	2/19	3/19
3	1/19	0/19	2/19	0/19
4	0/19	0/19	0/19	3/19

Probability, p(i,j) = GLCM(i,j)/19

$$\sigma_i^2 = \sum_{i,j} p(i,j)(i-\mu_i\ )^2$$

$$\sigma_j^2 = \sum_{i,j} p(i,j)(j-\mu_j\ )^2$$

$\sigma_i = (1\text{-}2.15)^2 + 2(1\text{-}2.15)^2 + 3(1\text{-}2.15)^2$
$+ (2\text{-}2.15)^2 + (2\text{-}2.15)^2 + 2(2\text{-}2.15)^2 + 3(2\text{-}2.15)^2$
$+ (3\text{-}2.15)^2 + 2(3\text{-}2.15)^2 + 3(4\text{-}2.15)^2$
$= 12.38$

Same procedure is applied for σ_j

Correlation from GLCM

- Measure of how correlated a pixel is to its neighbor in the image
- Range is -1 to 1
- 1 or -1 for a perfectly positively or negatively correlated image.
- NaN for constant image

$$Correlation = \sum_{i,j} (i - u_i)(j - \mu_j)p(i,j)/(\sigma_i \sigma_j)$$

- Single patches of a particular ground cover usually have a higher correlation value within them than between adjacent objects

Same procedure can be applied to symmetrical GLCM

Correlation derived from Example GLCM

j→

i→	1	2	3	4
1	0	1	2	3
2	1	1	2	3
3	1	0	2	0
4	0	0	0	3

GLCM

j→

i→	1	2	3	4
1	0/19	1/19	2/19	3/19
2	1/19	1/19	2/19	3/19
3	1/19	0/19	2/19	0/19
4	0/19	0/19	0/19	3/19

Probability, p(i,j) = GLCM(i,j)/19

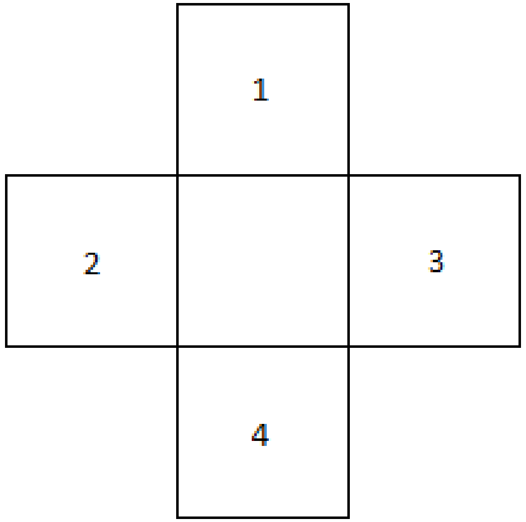
$$Correlation = \sum_{i,j} (i - u_i)(j - \mu_j)p(i,j)/(\sigma_i \sigma_j)$$

Correlation = 0.0783

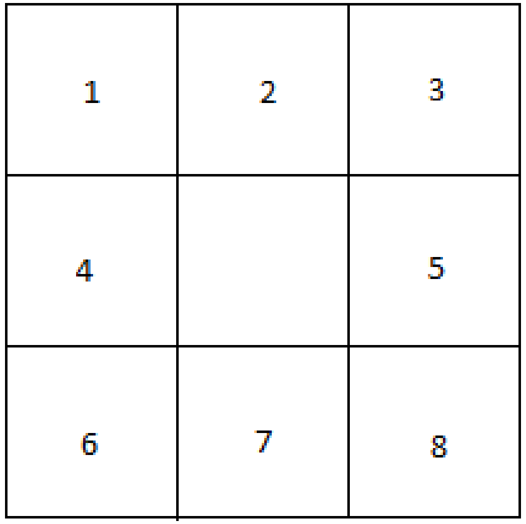
Connected Component Analysis

- Also known as Connected component labeling, blob extraction, or region labeling
- Used to determine the connectivity of “blob”-like regions in a binary image
- Used in computer vision to detect connected regions in binary images
- To extract separate objects from an image and describe these objects quantitatively

4- connectivity and 8-connectivty



4-connectivity



8-connectivity

Pixel Connectivity

- Pixels that are directly next to each other
- and belong to the foreground class can be considered to belong to the same object

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

Image

0	0	0	0	0	0	0	0
0	A	A	0	0	0	0	0
0	A	A	0	0	0	0	0
0	0	0	B	B	B	0	0
0	0	0	B	B	B	B	0
0	0	0	0	0	0	0	0

4-connectivity

Assign different colors two objects

0	0	0	0	0	0	0	0
0	A	A	0	0	0	0	0
0	A	A	0	0	0	0	0
0	0	0	A	A	A	0	0
0	0	0	A	A	A	A	0
0	0	0	0	0	0	0	0

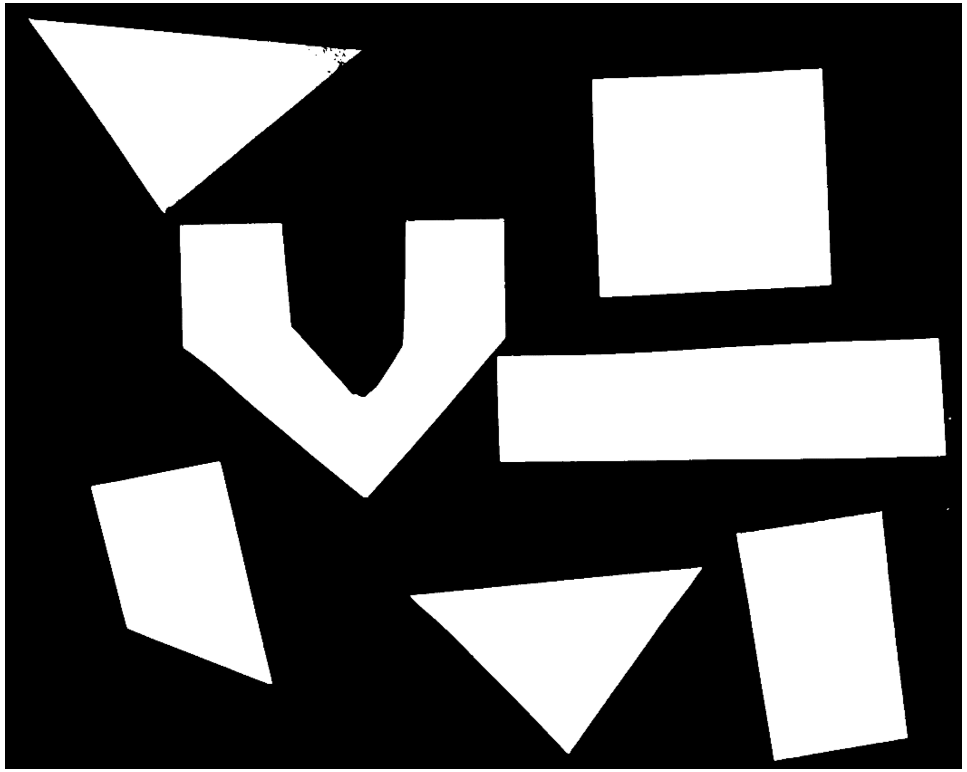
8-connectivity

Assign one color object

Connected Component Analysis for object detection

Objects with color are foreground and background is white

To count the number of objects, apply thresholding



7 objects are identified

Connected Component Analysis

- Connectivity can be 4-connected neighborhood or 8-connected neighborhood
- Scans an image and group its pixels into components based on pixel connectivity
- All pixels in a connected component share similar pixel intensity values and are in some way connected with each other
- Once all groups have been determined, each pixel is labeled with a graylevel or a color (color labeling) according to the component it was assigned to
- Extracting and labeling of various disjoint and connected components in an image is useful for automated image analysis applications

Steps for Connected Component Analysis

- Identifies connected pixel regions
- Identifies regions of adjacent pixels which have intensity from the given set, *V*
- For binary image V={1}.
- For gray image *V* can have a range of values
- Example: V = {51, 52, 53, ..., 77, 78, 79, 80}
- Scan the image by moving along a row until it comes to a point *p*
- Where *p* denotes the pixel to be labeled in the scanning process) for which V={1} or any other specified set

Algorithm for Connected Component Analysis for object detection

- Label each connected component (or blob) with the same unique label
- Can find total number of individual blobs
- Output is different for different type of connectivity
- There are two common ways of defining whether or not a component is connected
  - A pixel has 4 neighbors (sometimes called 4-connectivity)
  - A pixel has 8 neighbors

Algorithm for Connected Component Analysis for object detection

First pass: For each non-zero pixel, check its neighbours, if

- it has no non-zero neighbours, it is a new component, give it a new label
- it has one non-zero neighbor, these pixels are connected, give it the same label as the neighbour
- it has more than one non-zero neighbour, there are two cases:
  - neighbours have the same label, give the current pixel the same label
  - neighbours have different labels
    - Set the current pixel to the neighbours’ lowest label
    - Keep a note of the equivalences of all the connected labels
    - Record which different labels should be the same
    - Solve equivalence in the second pass

Algorithm for Connected Component Analysis for object detection

Second pass

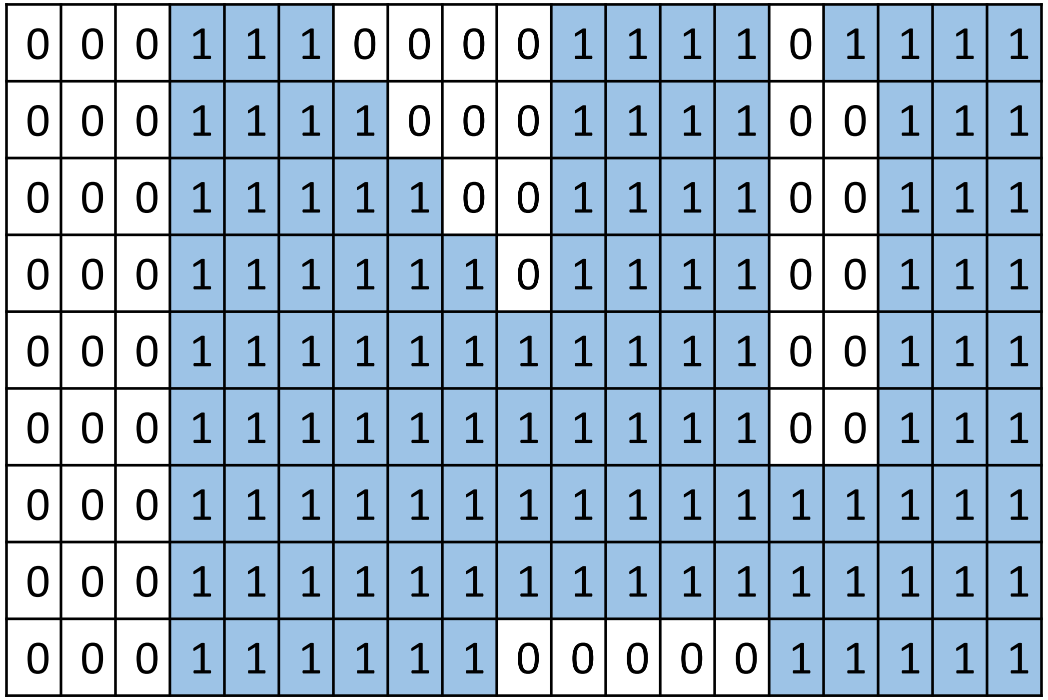
- For each pixel with a label,
- Check if this label has equivalent labels and change the label

Algorithm for Connected Component Analysis for object detection

- Identify object/s
- Visually it shows that image has one object

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

Image



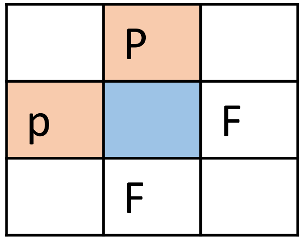
0	0	0	1	1	1	0	0	0	0	1	1	1	1	0	1	1	1	1
0	0	0	1	1	1	1	0	0	0	1	1	1	1	0	0	1	1	1
0	0	0	1	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1
0	0	0	1	1	1	1	1	1	0	1	1	1	1	0	0	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1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

Visually identified object

Algorithm for Connected Component Analysis

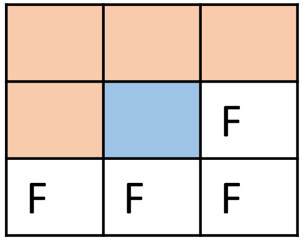
- Counter=0, label = 0
- Scan across row:
  - If pixel is 1 and connected, continue with lowest label above or behind (4-connected or 8-connected)
    - If 4-connectivity, 2 past values
    - If 8-connectivity, 4 past values
    - Record location of conflicts if any
  - If pixel is 1 and not connected, increment counter, and label = counter,
  - If pixel is 0 then label=0

4-connectivity



Has 2 past values and 2 future values

8-connectivity



Has 4 past values and 4 future values

Algorithm for Connected Component Analysis

- Counter=0 and label = 0 and Scan across row
- If pixel is 1 and connected, continue with lowest label above or behind (4-connected or 8-connected)
- If pixel is 1 and not connected, increment counter, and label = counter
- If conflict continue with lowest label and record the location

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

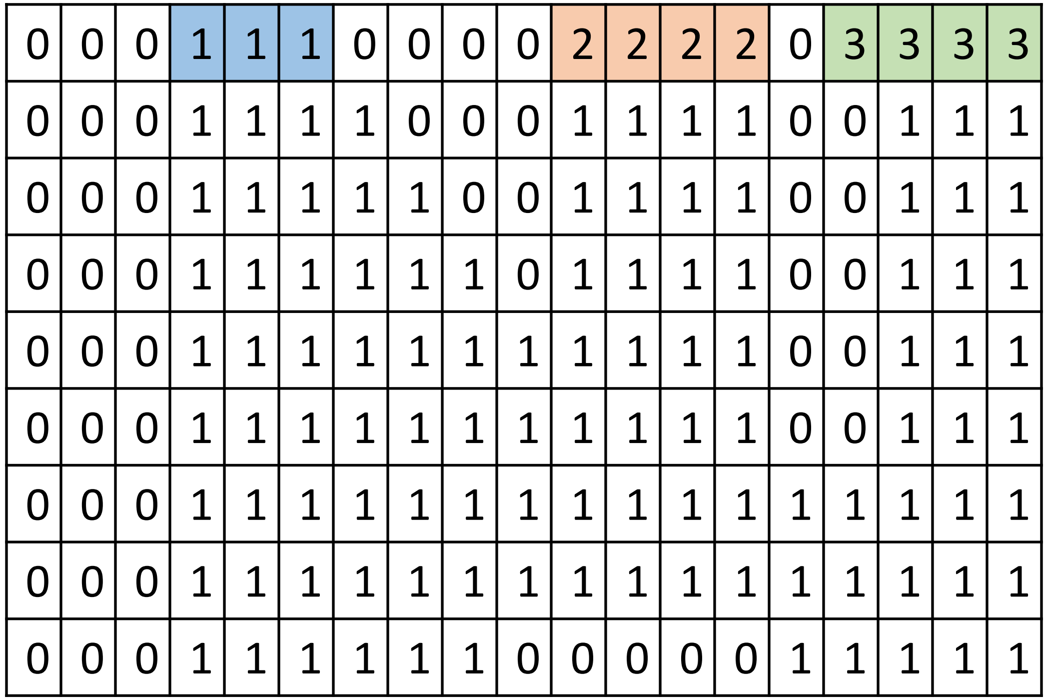
Image

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

Scanned first row

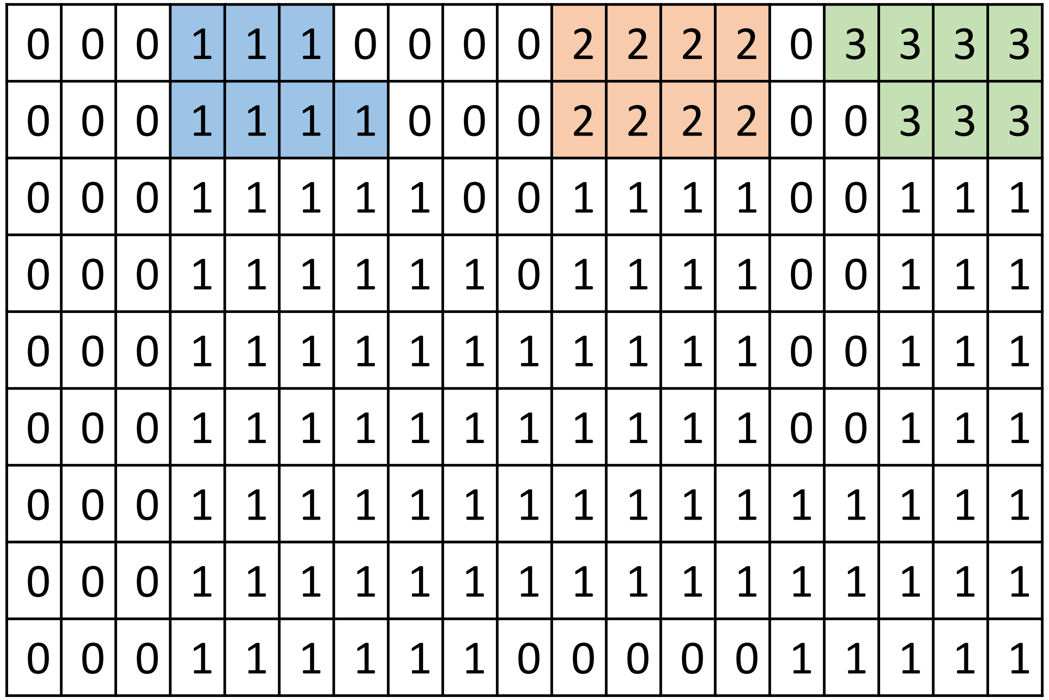
Algorithm for Connected Component Analysis

- Counter=0 and label = 0 and Scan across row
- If pixel is 1 and connected, continue with lowest label above or behind (4-connected or 8-connected)
- If pixel is 1 and not connected, increment counter, and label = counter
- If conflict continue with lowest label and record the location



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

Scanned first row

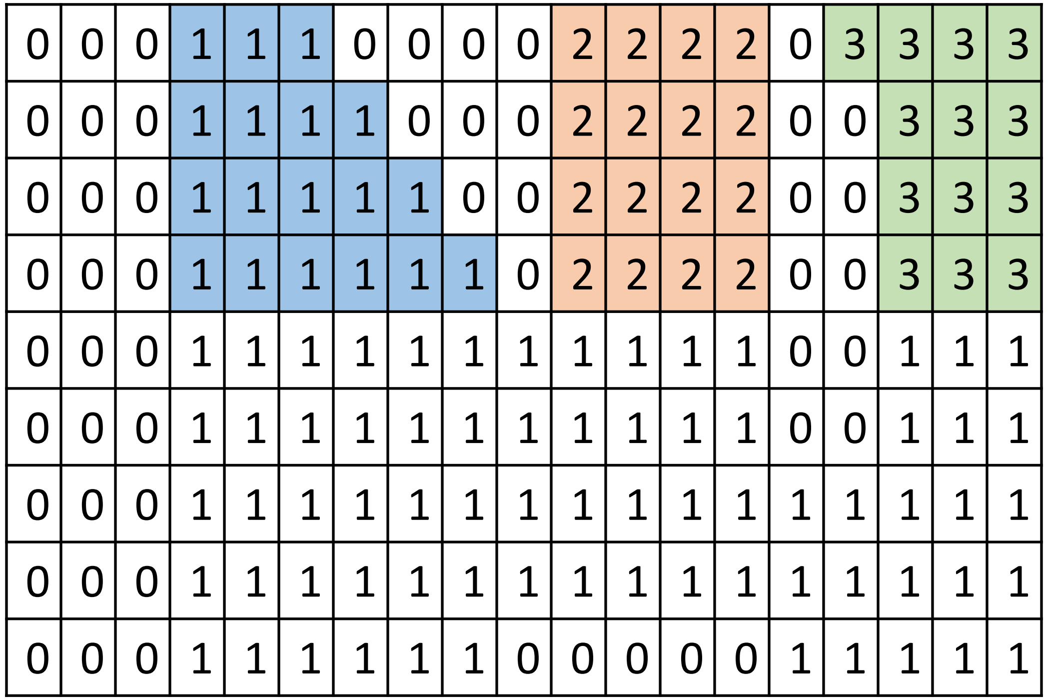


0	0	0	1	1	1	0	0	0	0	2	2	2	2	0	3	3	3	3
0	0	0	1	1	1	1	0	0	0	2	2	2	2	0	0	3	3	3
0	0	0	1	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1
0	0	0	1	1	1	1	1	1	0	1	1	1	1	0	0	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1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

Scanned second row

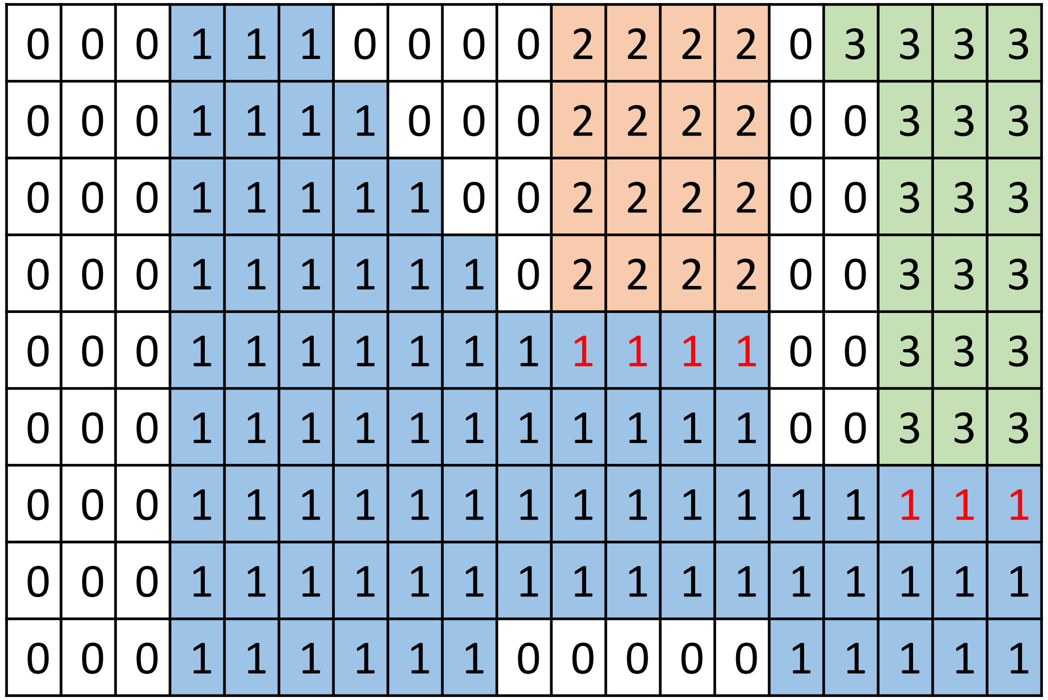
Algorithm for Connected Component Analysis

- Counter=0 and label = 0 and Scan across row
- If pixel is 1 and connected, continue with lowest label above or behind (4-connected or 8-connected)
- If pixel is 1 and not connected, increment counter, and label = counter
- If conflict continue with lowest label and record the location



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

Scanned next rows

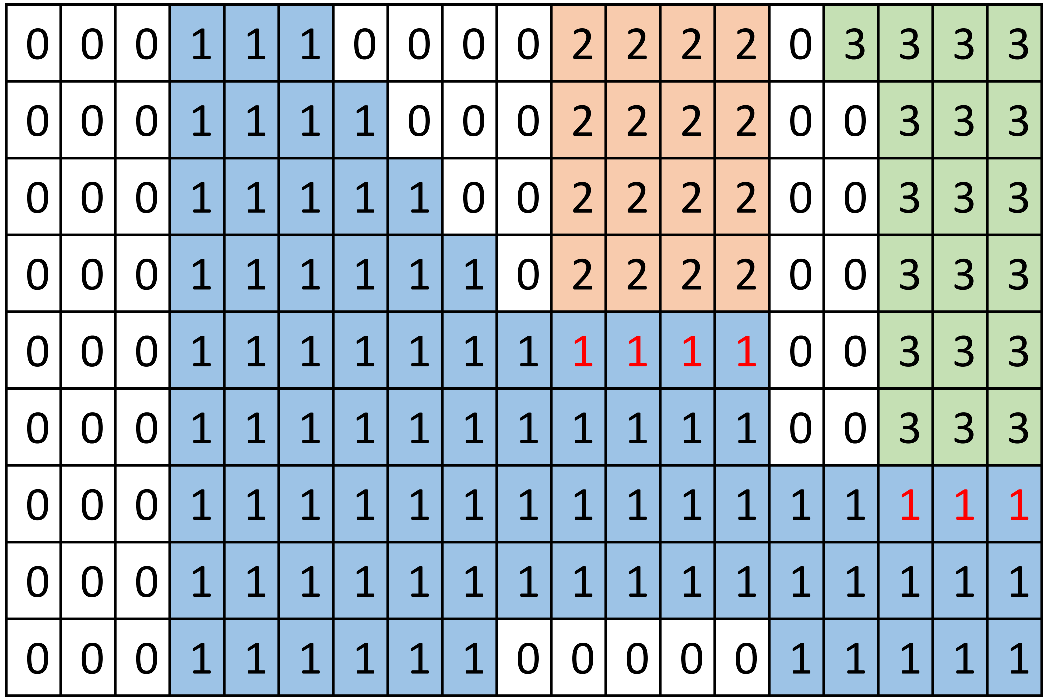


0	0	0	1	1	1	0	0	0	0	2	2	2	2	0	3	3	3	3
0	0	0	1	1	1	1	0	0	0	2	2	2	2	0	0	3	3	3
0	0	0	1	1	1	1	1	0	0	2	2	2	2	0	0	3	3	3
0	0	0	1	1	1	1	1	1	0	2	2	2	2	0	0	3	3	3
0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	3	3	3
0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	3	3	3
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1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

5th and 7th rows have conflict
1 ≡ 2, 1 ≡ 3

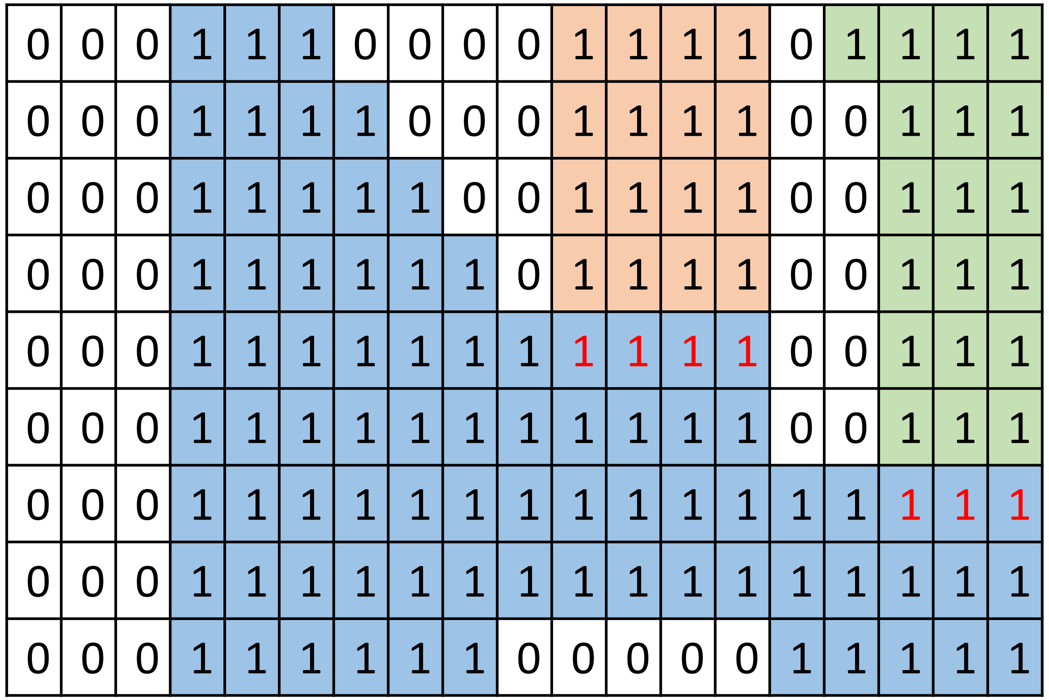
Algorithm for Connected Component Analysis

- Counter=0 and label = 0 and Scan across row
- If pixel is 1 and connected, continue with lowest label above or behind (4-connected or 8-connected)
- If pixel is 1 and not connected, increment counter, and label = counter
- If conflict continue with lowest label and record the location



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

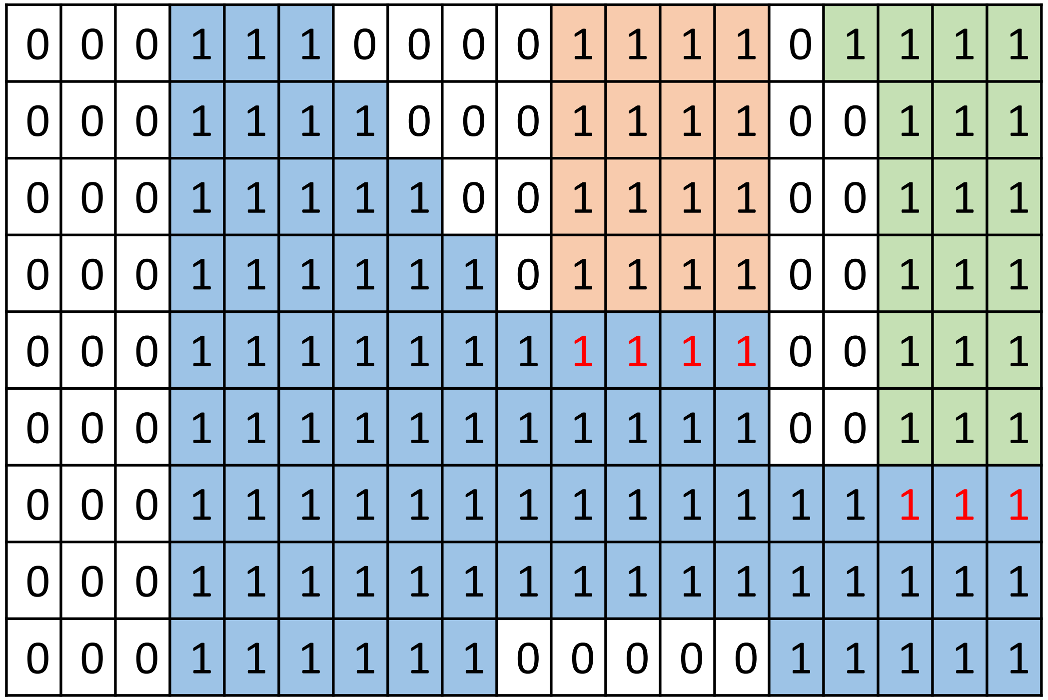
5th and 7th rows have conflict
1 ≡ 2, 1 ≡ 3



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

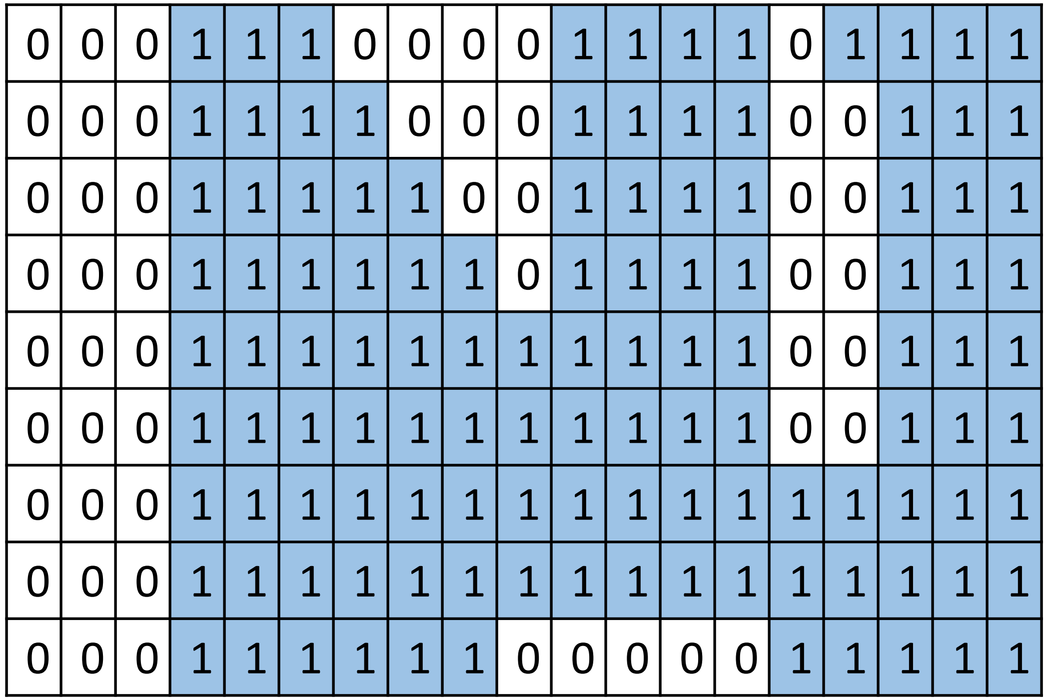
Change 2 and 3 to 1

Algorithm for Connected Component Analysis



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

Change 2 and 3 to 1

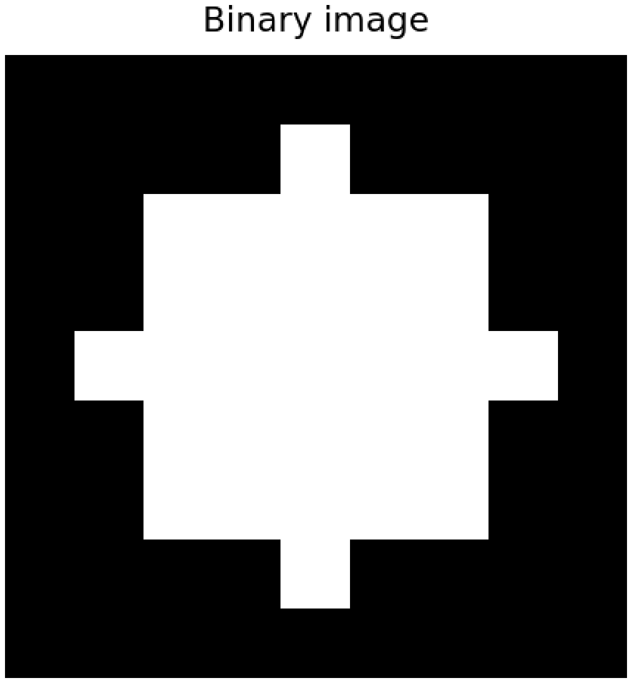


0	0	0	1	1	1	0	0	0	0	1	1	1	1	0	1	1	1	1
0	0	0	1	1	1	1	0	0	0	1	1	1	1	0	0	1	1	1
0	0	0	1	1	1	1	1	0	0	1	1	1	1	0	0	1	1	1
0	0	0	1	1	1	1	1	1	0	1	1	1	1	0	0	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	0	0	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1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

Assign a color to label 1

Distance Transform

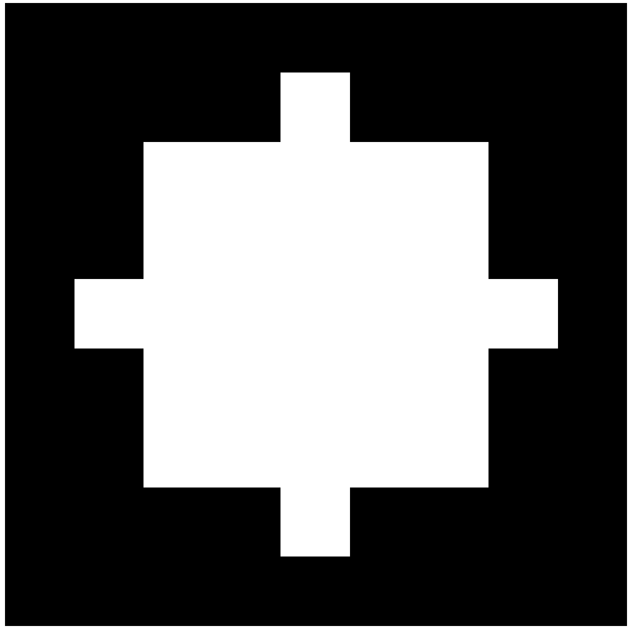
- A metric or measure of the separation of foreground pixels from background pixels
- Replaces each foreground pixel of a binary image with the distance to the closest background pixel
- Does not measure distance among foreground pixels
- Different grey levels are allocated to distance values



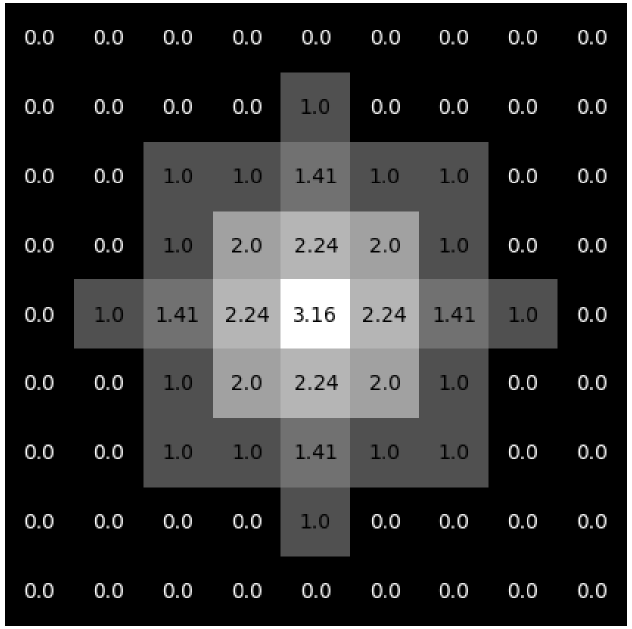
Distance Transform

- A metric or measure of the separation of foreground pixels from background pixels
- Replaces each foreground pixel of a binary image with the distance to the closest background pixel
- Does not measure distance among foreground pixels
- Different grey levels are allocated to distance values

Binary image



Distance transform

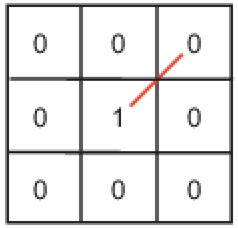


0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
0.0	0.0	1.0	1.0	1.41	1.0	1.0	0.0	0.0
0.0	0.0	1.0	2.0	2.24	2.0	1.0	0.0	0.0
0.0	1.0	1.41	2.24	3.16	2.24	1.41	1.0	0.0
0.0	0.0	1.0	2.0	2.24	2.0	1.0	0.0	0.0
0.0	0.0	1.0	1.0	1.41	1.0	1.0	0.0	0.0
0.0	0.0	0.0	0.0	1.0	0.0	0.0	0.0	0.0
0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Distance Transform

- Distance Metrics
  - Euclidean
    - Straight-line distance between p (foreground) and q (nearest background) pixels
    - Distance between two consecutive pixels is 1 unit

$$d = ((p_1 - q_1)^2 + (p_2 - q_2)^2)^{1/2}$$



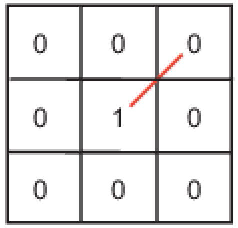
0	0	0
0	1	0
0	0	0

Image

Distance Transform

- Distance Metrics
  - Euclidean
    - Straight-line distance between p (foreground) and q (nearest background) pixels
    - Distance between two consecutive pixels is 1 unit

$$d = ((p_1 - q_1)^2 + (p_2 - q_2)^2)^{1/2}$$



0	0	0
0	1	0
0	0	0

Image

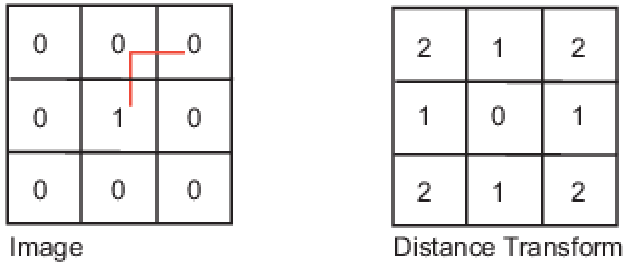
1.41	1.0	1.41
1.0	0.0	1.0
1.41	1.0	1.41

Distance Transform

Distance Transform

- City Block (Manhatten Distance)
  - Measures the path between the pixels based on a 4-connected neighborhood
  - Pixels with edges touching foreground pixel are 1 unit apart
  - Pixels diagonally touching are 2 units apart

$$d = |p_1 - q_1| + |p_2 - q_2|$$



0	0	0
0	1	0
0	0	0

Image

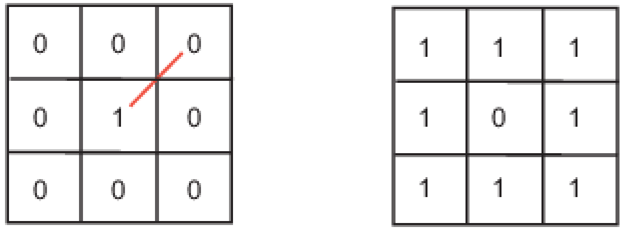
2	1	2
1	0	1
2	1	2

Distance Transform

City Block distance

Distance Transform

- Chessboard
  - measures the path between the pixels based on an 8-connected neighborhood
  - Pixels whose edges or corners touch are 1 unit apart



0	0	0
0	1	0
0	0	0

Image

1	1	1
1	0	1
1	1	1

Distance Transform

Chessboard

Distance Transform

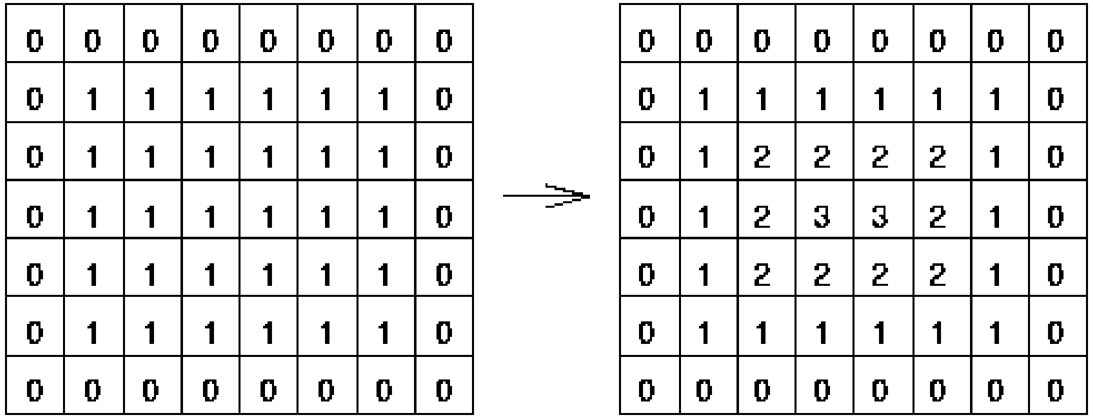
- Using chess board distance metric

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

→

Distance Transform

- Using chess board distance metric



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

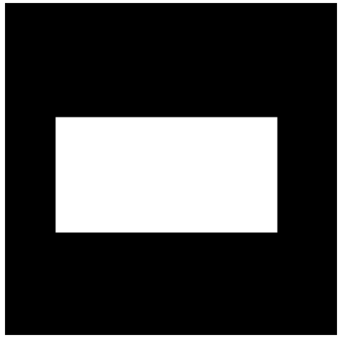
→

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

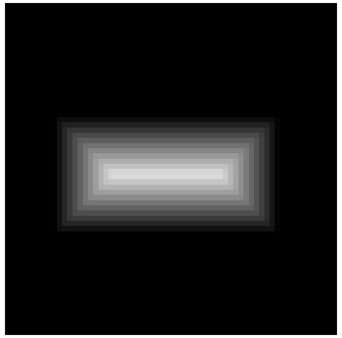
Distance Transform

- Dual of distance transform produces the distance transform for the background
- Can be considered as a process of inverting the original image and then applying the standard transform

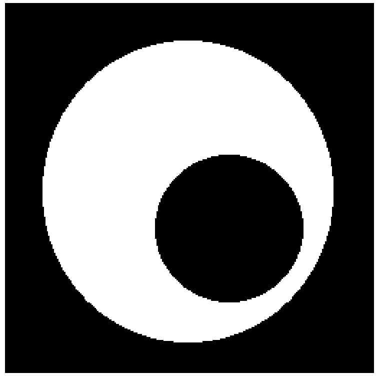
Examples: Distance Transform



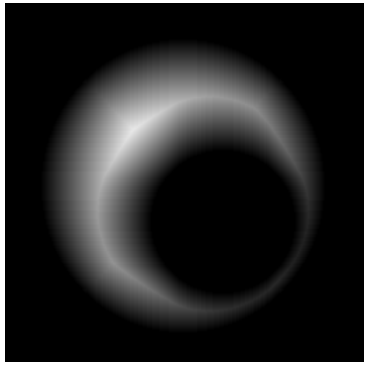
Binary image



Distance transform

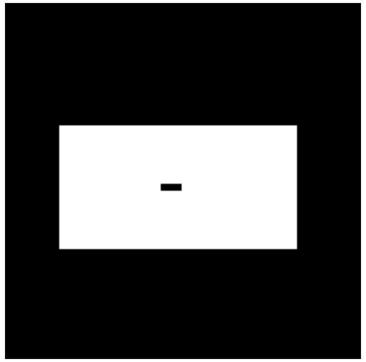


Binary image

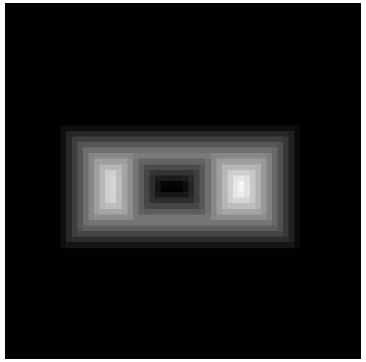


Distance transform

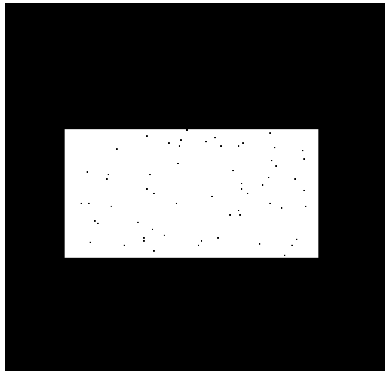
- distance transform is sensitive to small changes in the object



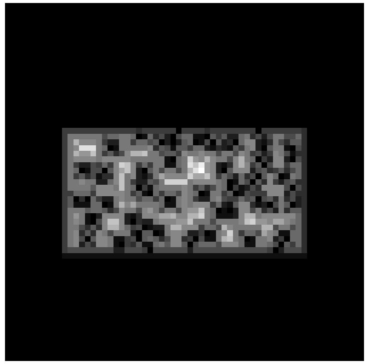
Binary image



Distance transform



Binary image

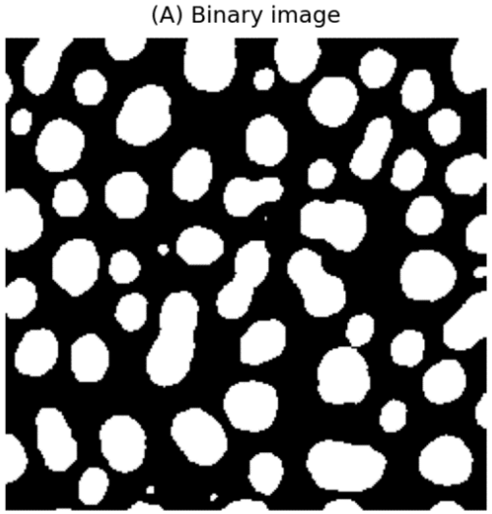


Distance transform

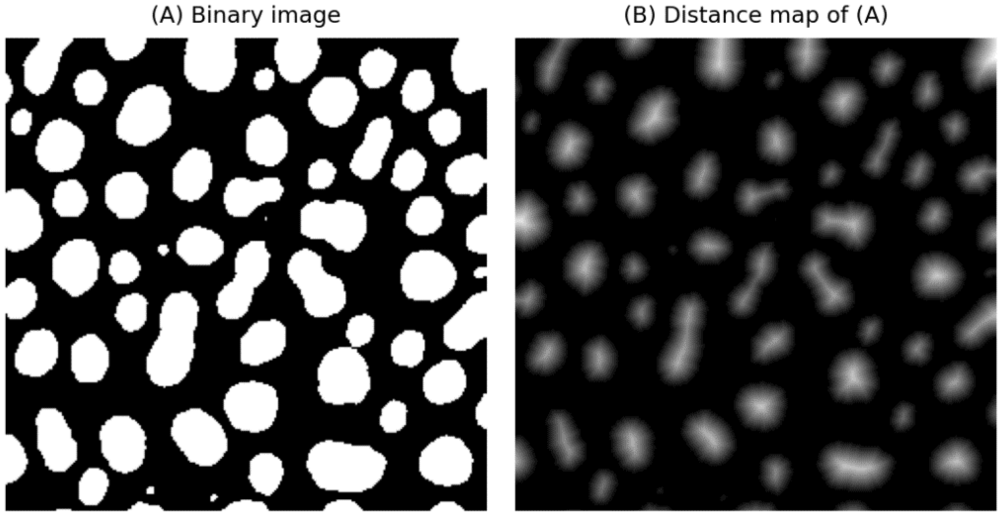
Why distance transform?

- Eroding or dilating binary images by a large amount can be very slow using for morphology operations or spatial
  - because large maximum or minimum structuring element/ filters are required
- Instead apply distance map (transform) and apply global threshold
  - Much faster than morphology operation

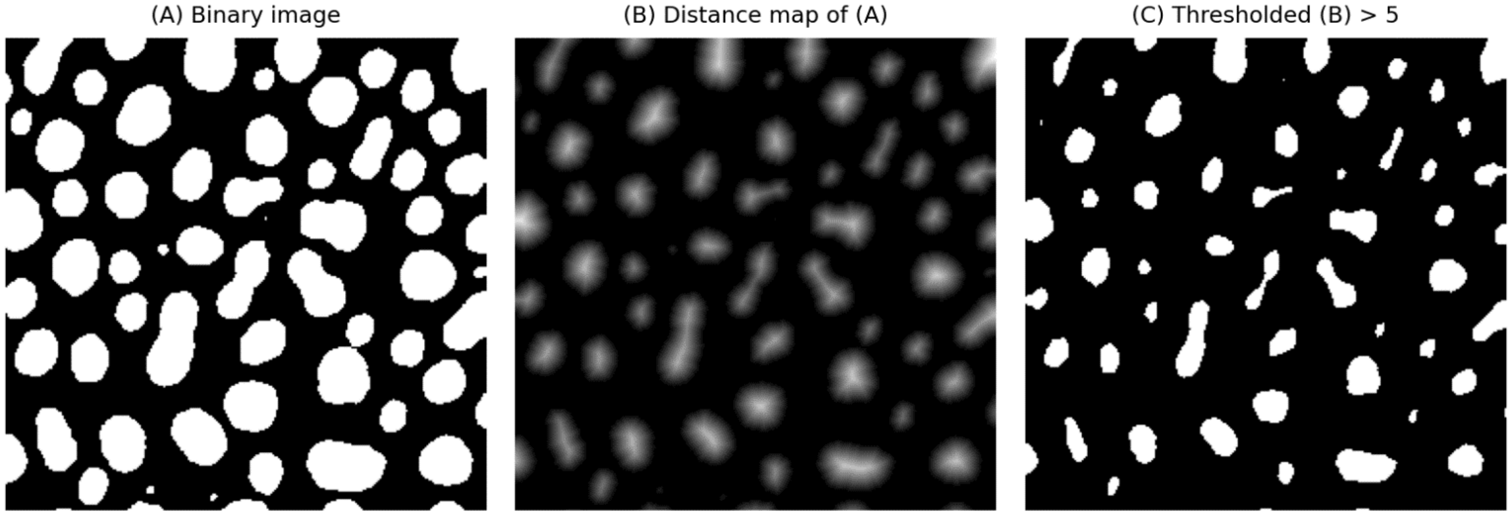
Application of distance transform



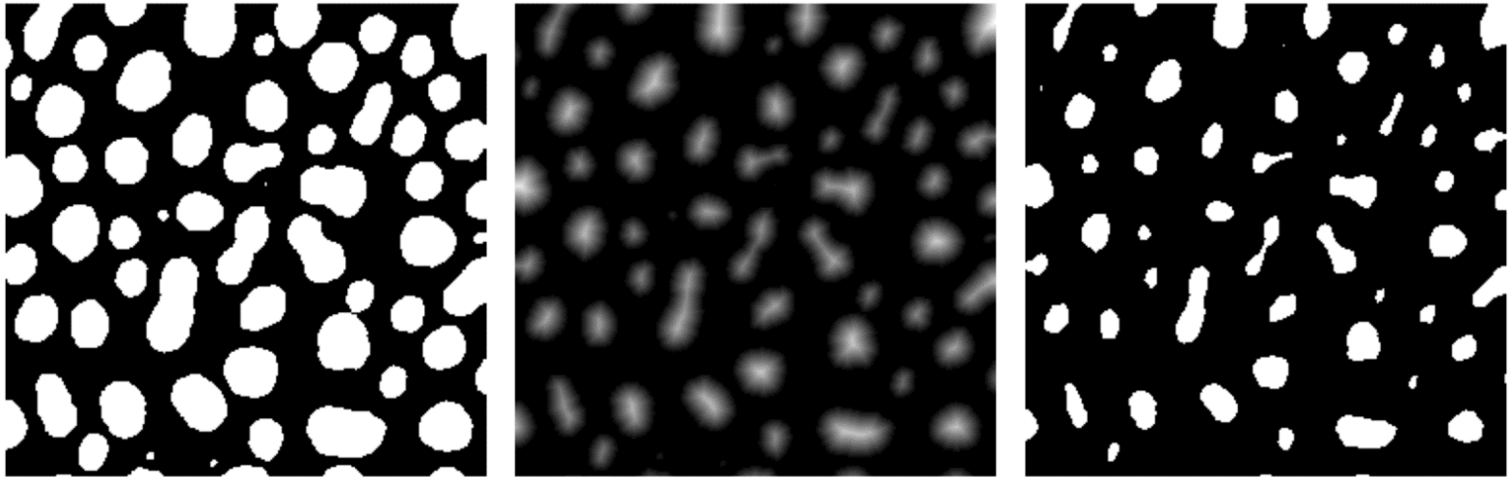
Application of distance transform



Application of distance transform



Application of distance transform

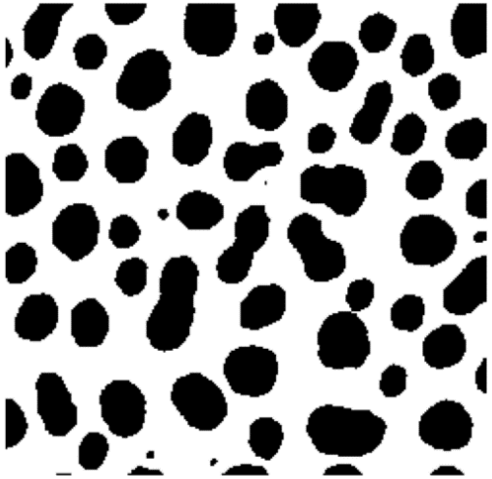


(A) Binary image

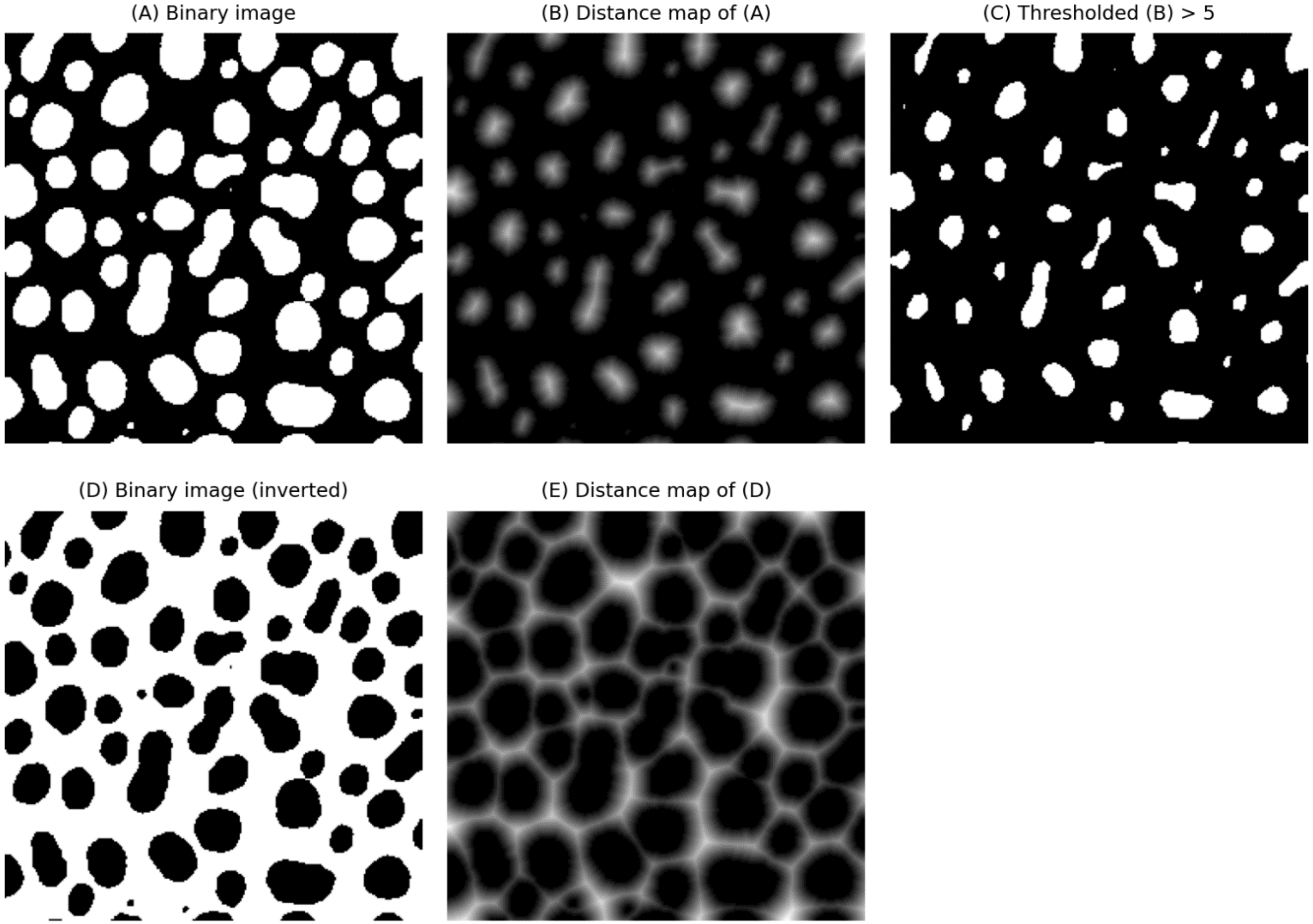
(B) Distance map of (A)

(C) Thresholded (B) > 5

(D) Binary image (inverted)



Application of distance transform



(A) Binary image

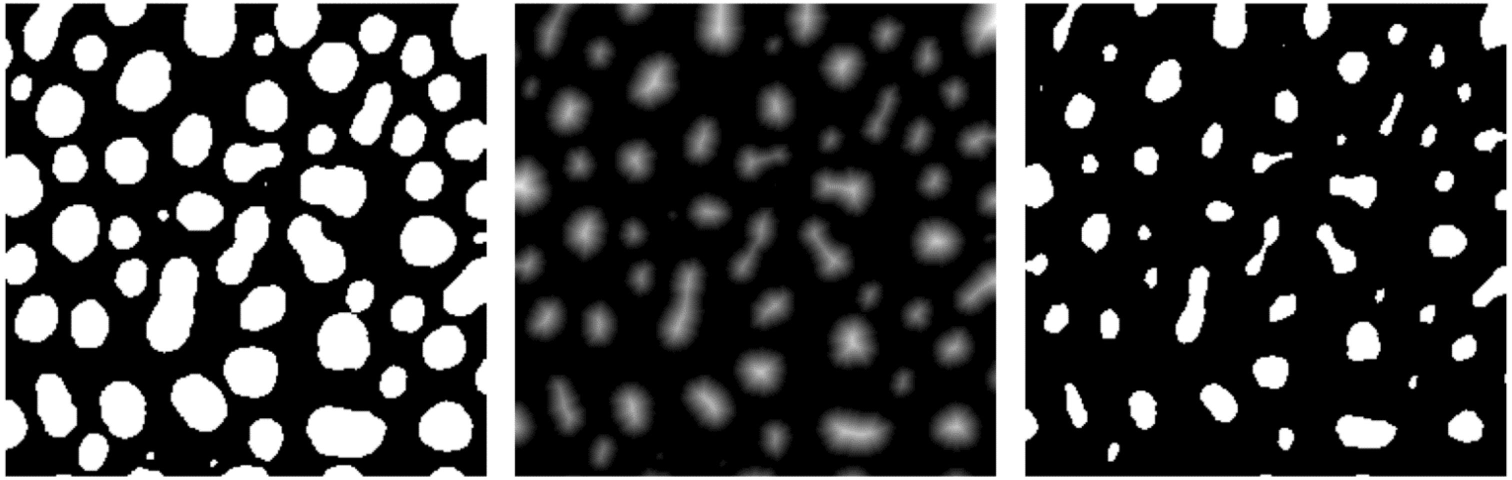
(B) Distance map of (A)

(C) Thresholded (B) > 5

(D) Binary image (inverted)

(E) Distance map of (D)

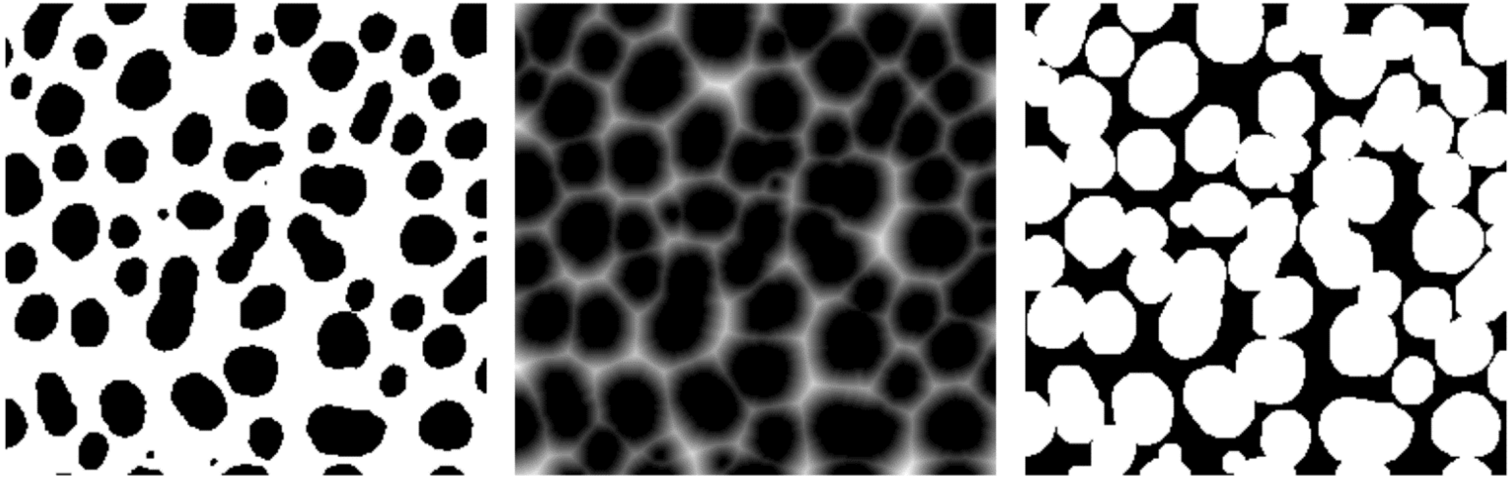
Application of distance transform



(A) Binary image

(B) Distance map of (A)

(C) Thresholded (B) > 5



(D) Binary image (inverted)

(E) Distance map of (D)

(F) Thresholded (E) < 5

Skeletonization /Medial Axis Transform(MAT)

- The skeleton is a binary image showing the skeleton of objects
- MAT can also be used to determine skeleton of an object
- It generates gray image where each point on the skeleton has intensity which represents

  its distance to a boundary in the original object
- MAT is the similar to distance transform but with all points away from skeleton are suppressed to zero

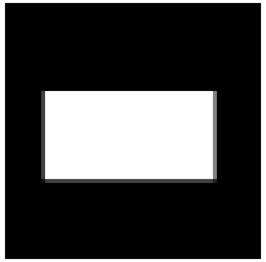
Skeletonization using morphology or MAT

Morphology

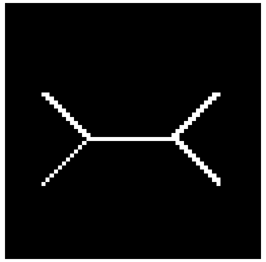
- Use thinning that successively erodes away pixels from the boundary
- while preserving the end points of line segments until no more thinning is possible
- Resulting image is skeleton

- MAT
  - Calculate the distance transform and apply threshold
  - more efficient than morphology

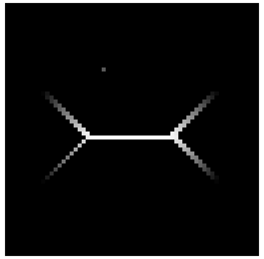
Skeletonization /Medial Axis Transform



Binary image

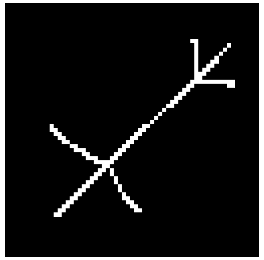


Morphology

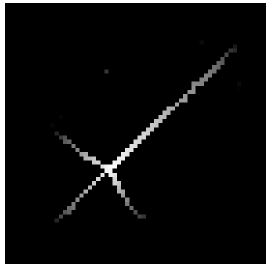


MAT

Binary image



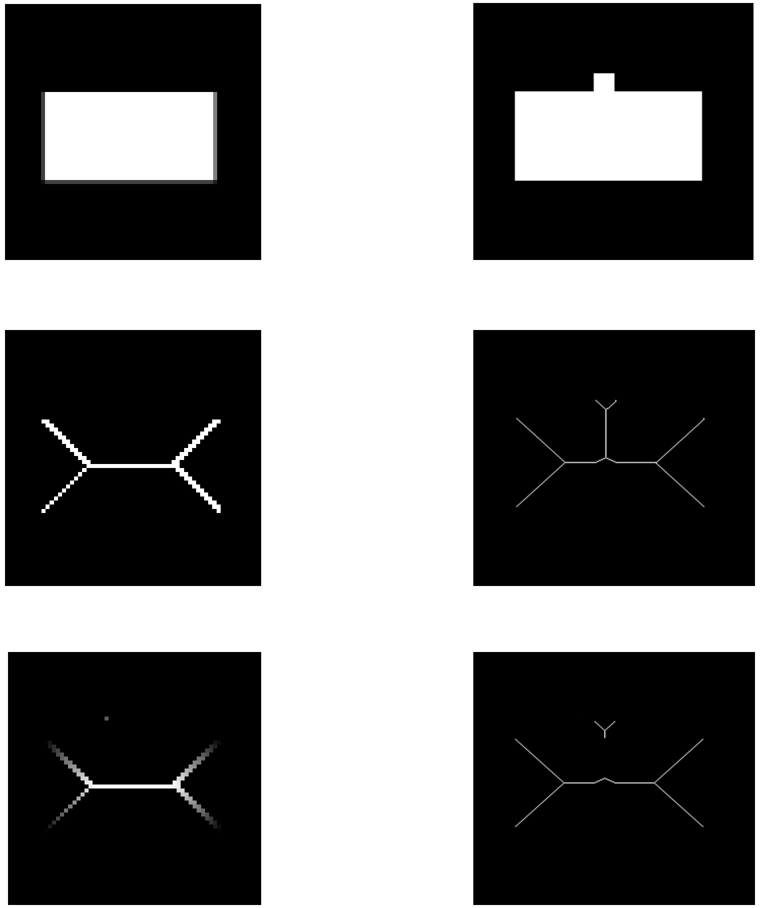
Morphology



MAT

Skeletonization /Medial Axis Transform

- Skeletonization using MAT is also very sensitive to noise



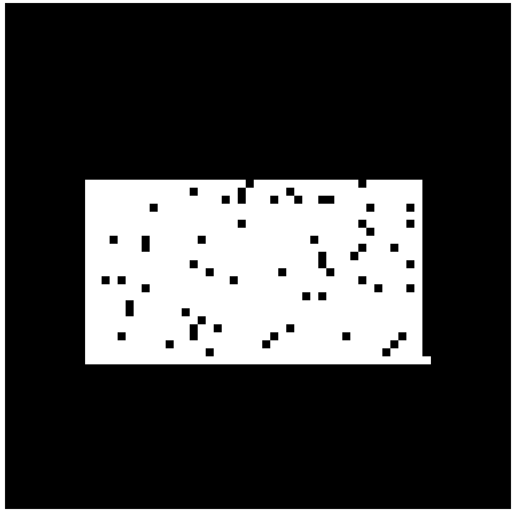
Binary image

Morphology

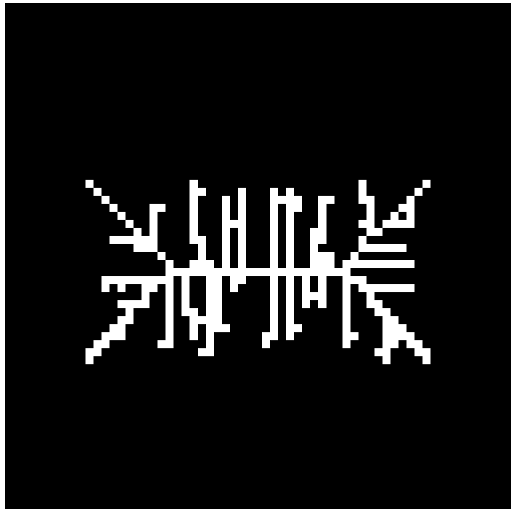
MAT

Skeletonization /Medial Axis Transform

- Skeletonization using MAT is also very sensitive to noise
- Apply averaging/ median filter to reduce noise before applying MAT



Binary image with salt and pepper noise



MAT