Digital Image Processing

COSC 6380/4393

Lecture – 9

Sept. 19th, 2023

Pranav Mantini

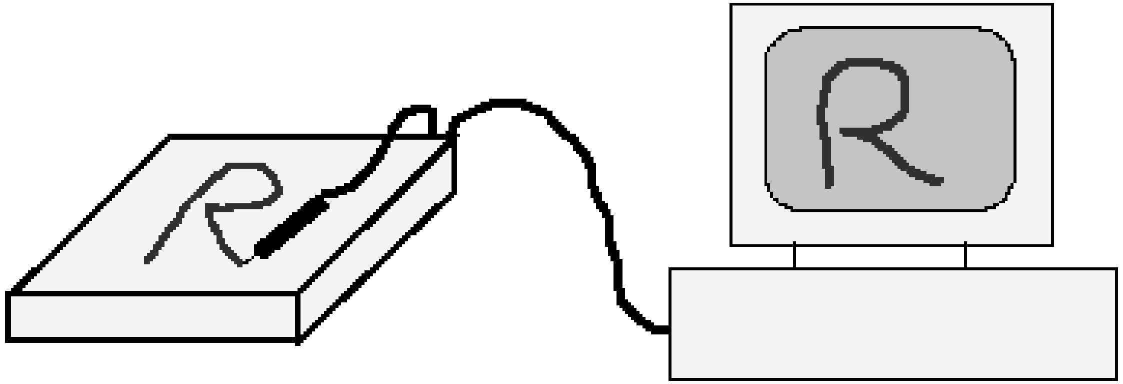
Slides from Dr. Shishir K Shah and Frank (Qingzhong) Liu

Review: BINARY IMAGES

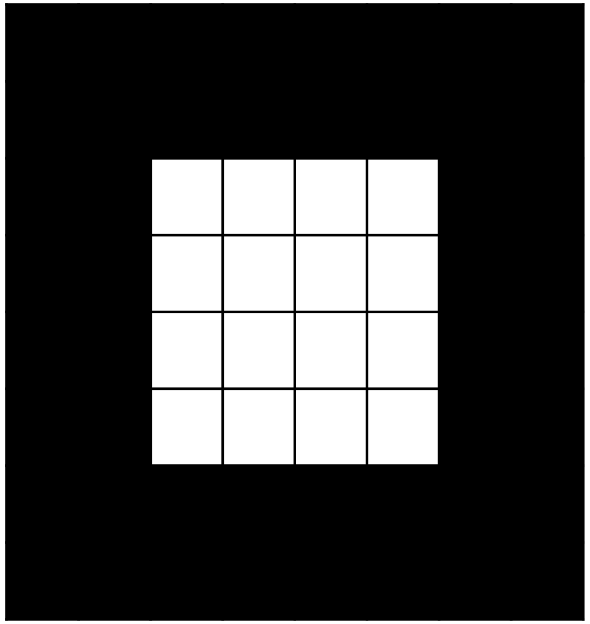
- How do **binary images** arise?
- Since **binary = bi-valued,** the (logical) values '0' or '1' usually indicate the **absence** or **presence** of an **image** property in an associated gray-level image:
  - Points of high or low intensity (brightness)
  - Points where an object is present or absent
  - More abstract properties, such as smooth vs. nonsmooth, etc.
- **Convention** - We will make the associations
- '1' = BLACK
- '0' = WHITE

Review: BINARY IMAGE GENERATION

- **Tablet-Based Input:**
- Binary images can derive from **simple sensors** with binary output
- Simplest example: **tablet, resistive pad,** or **light pen**
- All pixels **initially** assigned value '0':
  I = [I(i, j)], I(i, j) = '0' for all (i, j) = (row column)
- When pressure or light is applied at (i_0 , j_0), the image is assigned the value '1': I(i_0 , j_0) = '1'
- This continues until the user completes the drawing



Review: Grey Level → Binary Image



8X8 image → white box on black background

$Threshold$(T) →

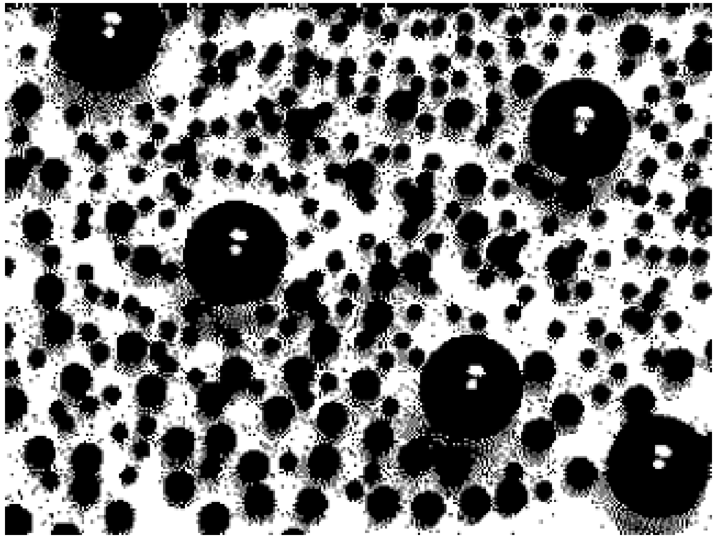
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

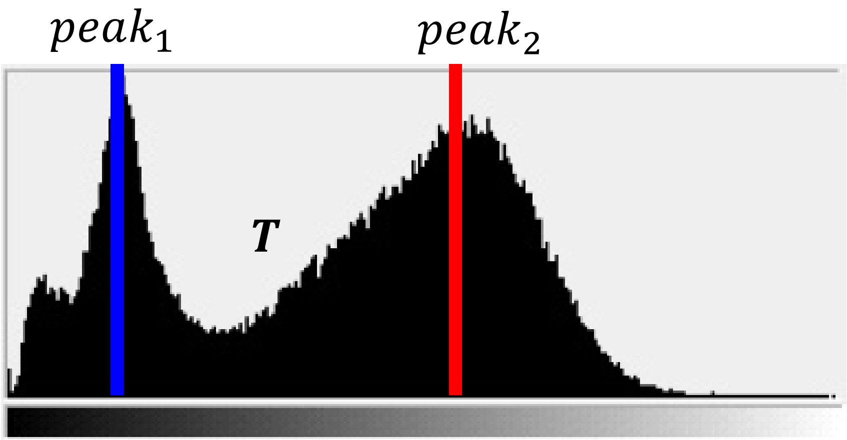
Binary image

What is good value of T?

Review: Example: How to find T

- Determine peaks
- Choose T between peaks(say average)





Review: Algorithm

$Initialize\ T = K/2$

Do

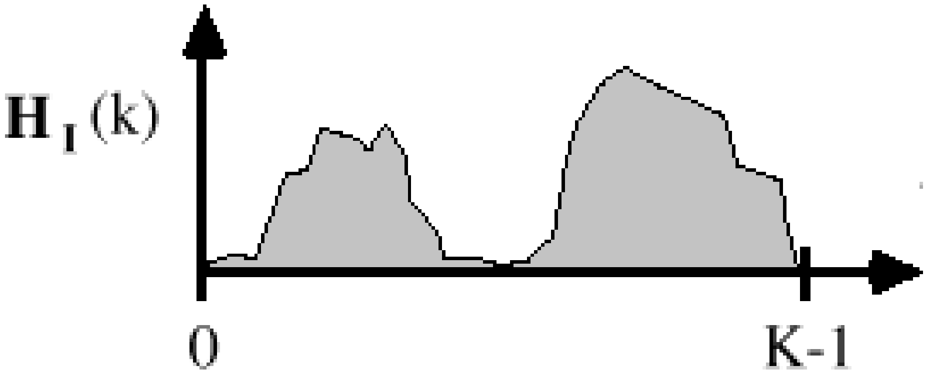
$\quad Compute\ \mu_1 = E(X)\forall X < T$

$\quad Compute\ \mu_2 = E(X)\forall X \geq T$

$\quad Set\ T = \dfrac{\mu_1 + \mu_2}{2}$

$While\ \Delta\mu_1 != 0\ \&\Delta\mu_2 != 0$

AKA: Expectation Maximization (simple version)

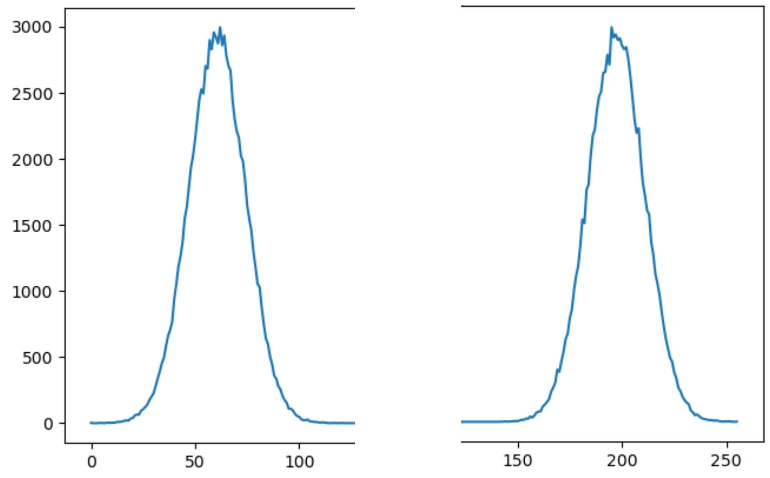


bimodal histogram
well separated peaks

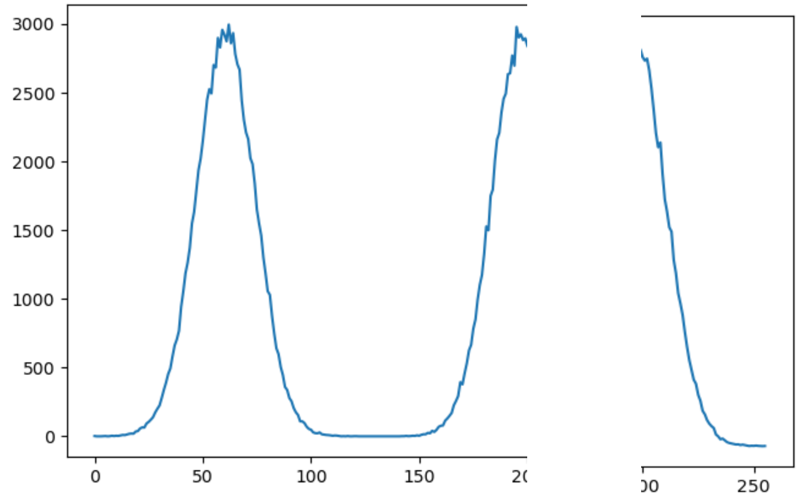
Otsu’s Binarization

- Popular thresholding method
- It works on the histogram of the image
- It assumes that the histogram is bimodal

T=125



T=200



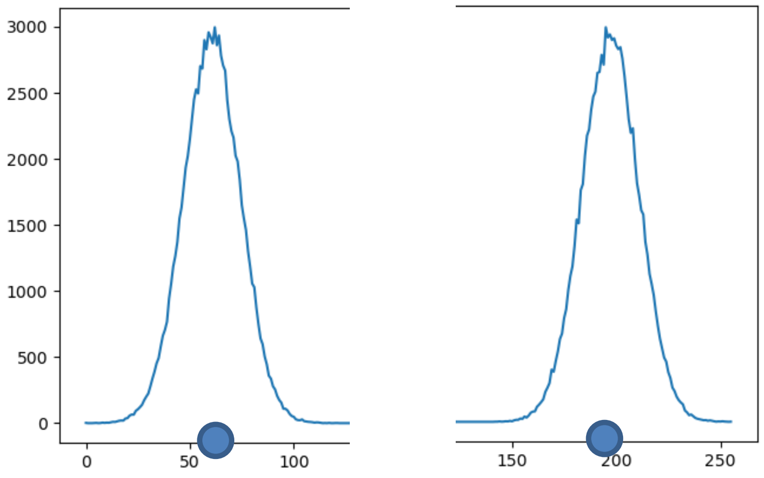
Compute a statistical value

Compute a statistical value

Compare

Pick Threshold

T=125



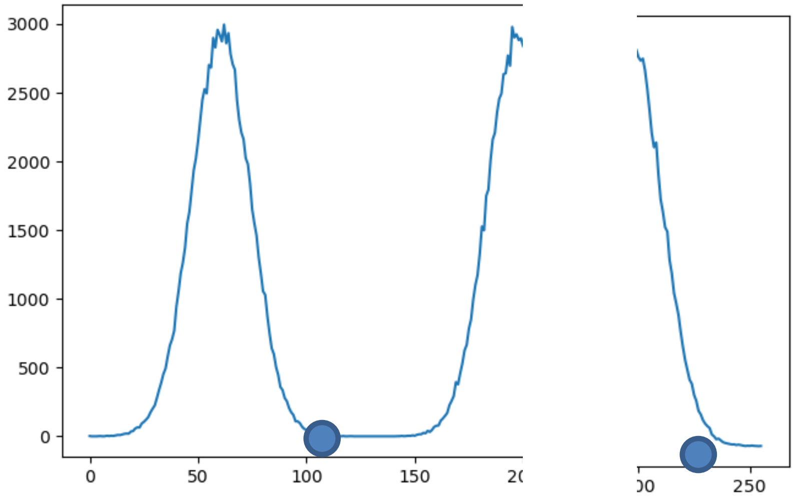
Mean = 62.03

Var = 189.3

Mean = 195.8

Var = 190.0

T=200



Mean = 107.54

Var = 3709.54

Mean = 225.4

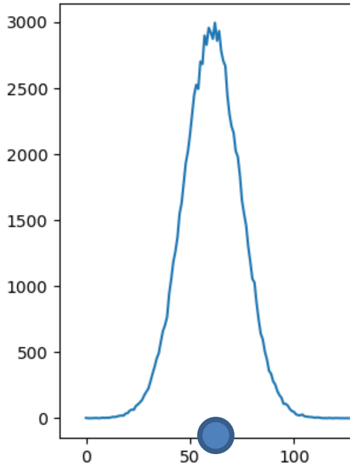
Var = 259.1

Intra-class Variance

Compare intra-class var

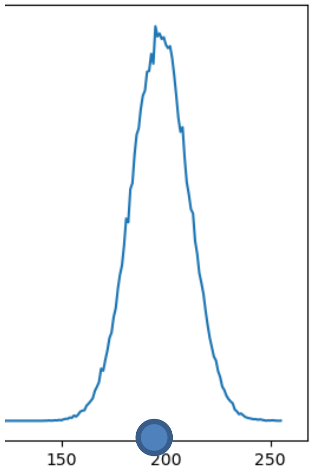
Pick Threshold that minimizes this value

T=125



Mean = 62.03

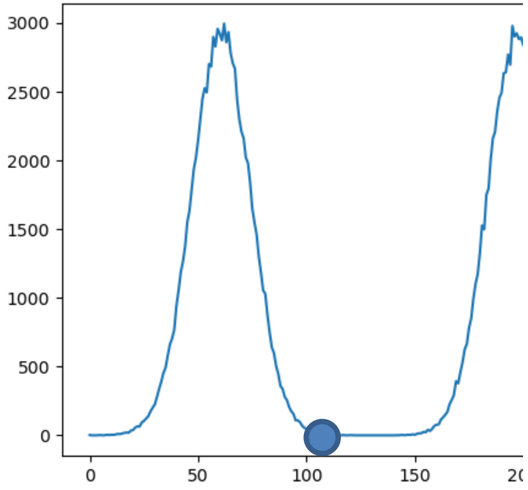
Var = 189.3 +



Mean = 195.8

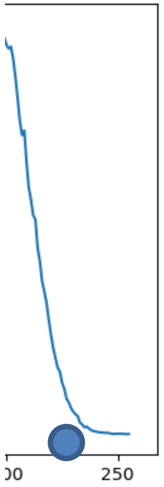
Var = 190.0

T=200



Mean = 107.54

Var = 3709.54 +



Mean = 225.4

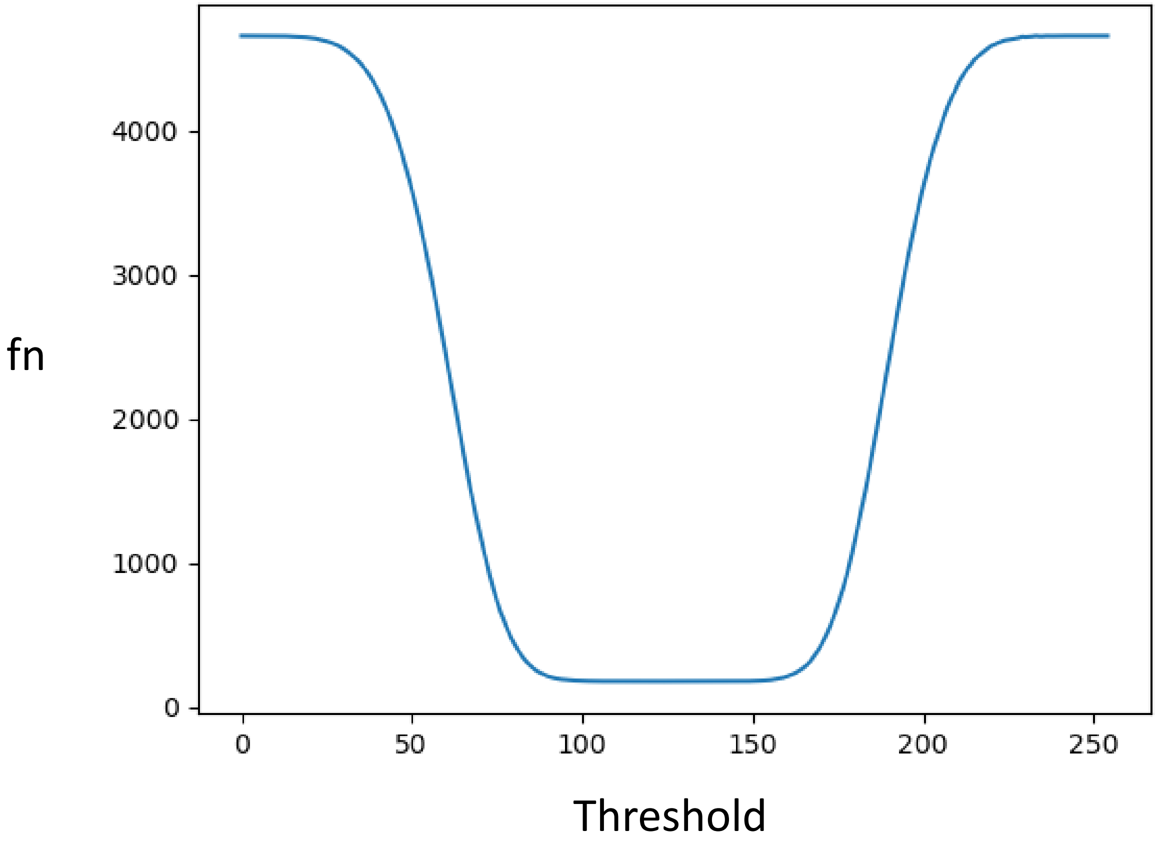
Var = 259.1

364 Intra-class Variance 4370

Compare intra-class var

Pick Threshold that minimizes this value

$$fn = w_1\, var1 + \; w_2 var2$$



Extreme values

Low values on either side of the valley

Algorithm

1. Compute Histogram
2. Compute probabilities
3. Iterate through all possible threshold values (t=0 to t= 255)
    1. Compute weights (q_1, q_2)
    2. Compute mean (μ_1, μ_2)
    3. Compute intra-class variance $(\boldsymbol{\sigma_1^2}, \boldsymbol{\sigma_2^2})$
    4. Compute weighted sum of intra-class variance $(\boldsymbol{q_1\sigma_1^2} + \boldsymbol{q_2\sigma_2^2})$
4. Pick threshold that minimizes the weighted sum of intraclass variance

DISCUSSION OF HISTOGRAM TYPES

- We'll return to the histogram later in the context of **quantitative gray-level properties**
- Some general qualitative observations are worth making now
  - **Bimodal histograms** often imply **objects** and **background** of significantly different average brightnesses
  - **Bimodal histograms** are the easiest to threshold
  - The result of thresholding a **bimodal histogram** is (ideally) a simple binary image showing **object/background separation**
- Examples. Images of
  - Printed type
  - Blood cells in solution
  - Machine parts on an assembly line

HISTOGRAM TYPES

- **Multi-modal histograms** often occur when the image contains different objects of different average brightnesses on a uniform background
- **Flat or level histograms** usually imply more complex images, containing detail, non-uniform background, etc.
- Thresholding rarely gives perfect results
- Usually, some kind of **region correction** must be applied

THE BASIC LOGICAL OPERATIONS

- We will use only a **few simple** logical operations
- Suppose that X_1 ,..., X_n are **binary variables**
- For example, pixels from one or more binary images
- Here is the notation we will use:
- **Logical Complement**: $NOT(X_1)$ = complement of X_1
- **Logical AND**: $AND(X_1, X_2) = X_1 \wedge X_2$

X_1	$NOT(X_1)$
0	1
1	0

TRUTH TABLE

X_1	X_2	$X_1 \wedge X_2$
0	0	0
0	1	0
1	0	0
1	1	1

TRUTH TABLE

LOGICAL OPERATIONS

- **Multi-Variable Logical AND:**

$$AND(X_1, X_2, \ldots, X_n) = X_1 \wedge X_2 \wedge \cdots\cdots \wedge X_{n-1} \wedge X_n$$

$$= \bigwedge_{i=1}^{n} X_i$$

$$= 1 \qquad \text{if } X_1 = X_2 = X_3 = \cdots\cdots = X_{n-1} = X_n = 1 \text{ (all 1's)}$$

$$= 0 \qquad \text{otherwise}$$

- **Logical OR**: OR(X1, X2) = X1 ∨ X2

X_1	X_2	$X_1 \vee X_2$
0	0	0
0	1	1
1	0	1
1	1	1

TRUTH TABLE

LOGICAL OPERATIONS

- **Multi-Variable Logical OR:**

$$OR(X_1, X_2, \ldots, X_n) = X_1 \vee X_2 \vee \cdots\cdots \vee X_{n-1} \vee X_n$$

$$= \bigvee_{i=1}^{n} X_i$$

$$= 0 \qquad \text{if } X_1 = X_2 = X_3 = \cdots\cdots = X_{n-1} = X_n = 0 \text{ (all 0's)}$$

$$= 1 \qquad \text{otherwise}$$

SIMPLE BOOLEAN ALGEBRA PROPERTIES

1. NOT [NOT(X)] = X
2. $X1 \wedge X2 \wedge X3 = (X1 \wedge X2) \wedge X3$
   $= X1 \wedge (X2 \wedge X3)$
3. $X1 \vee X2 \vee X3 = (X1 \vee X2) \vee X3$
   $= X1 \vee (X2 \vee X3)$
4. $X1 \wedge X2 = X2 \wedge X1$
5. $X1 \vee X2 = X2 \vee X1$
6. $(X1 \wedge X2) \vee X3 = (X1 \vee X3) \wedge (X2 \vee X3)$
7. $(X1 \vee X2) \wedge X3 = (X1 \wedge X3) \vee (X2 \wedge X3)$
8. NOT($X1 \wedge X2$) = NOT($X1$) $\vee$ NOT($X2$)
9. NOT($X1 \vee X2$) = NOT($X1$) $\wedge$ NOT($X2$)

SIMPLE BOOLEAN ALGEBRA PROPERTIES

1. NOT [NOT(X)] = X
2. $X1 \wedge X2 \wedge X3 = (X1 \wedge X2) \wedge X3$
   $= X1 \wedge (X2 \wedge X3)$ (Associative Law)
3. $X1 \vee X2 \vee X3 = (X1 \vee X2) \vee X3$
   $= X1 \vee (X2 \vee X3)$ (Associative Law)
4. $X1 \wedge X2 = X2 \wedge X1$ (Commutative Law)
5. $X1 \vee X2 = X2 \vee X1$ (Commutative Law)
6. $(X1 \wedge X2) \vee X3 = (X1 \vee X3) \wedge (X2 \vee X3)$ (Distributive Law)
7. $(X1 \vee X2) \wedge X3 = (X1 \wedge X3) \vee (X2 \wedge X3)$ (Distributive Law)
8. $\mathrm{NOT}(X1 \wedge X2) = \mathrm{NOT}(X1) \vee \mathrm{NOT}(X2)$ (DeMorgan’s Law)
9. $\mathrm{NOT}(X1 \vee X2) = \mathrm{NOT}(X1) \wedge \mathrm{NOT}(X2)$ (DeMorgan’s Law)

BOOLEAN ALGEBRA

- **Binary Majority** (odd # of variables only)

X_1	X_2	X_3	$MAJ(X_1, X_2, X_3)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

TRUTH TABLE

BOOLEAN ALGEBRA

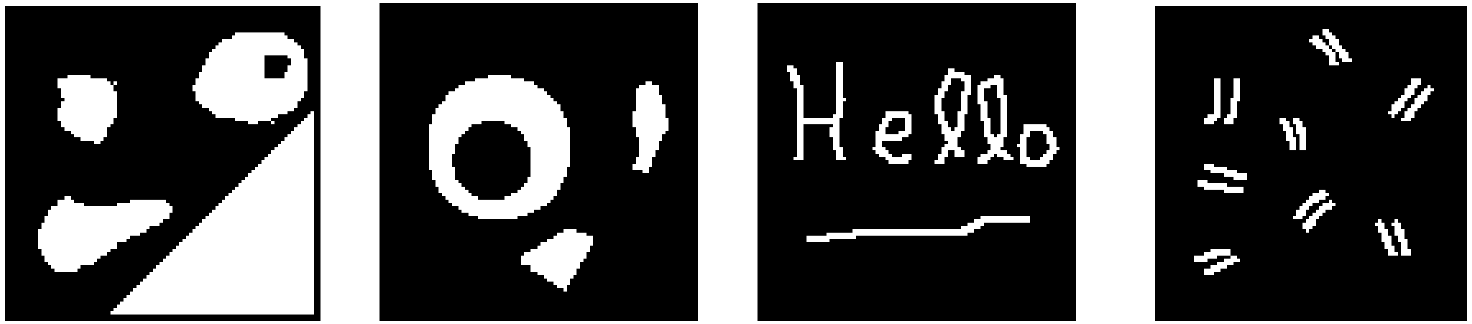
- **Multi-Variable Binary Majority:**
- $MAJ(X_1, X_2, \ldots, X_n) = 1$ if more 1's than 0's $= 0$ if more 0's than 1's
- **Comments**
  - **Any** binary operation can be created from 'NOT', 'AND', 'OR' - Boolean
  - Algebra is an entire math discipline built on these
  - However, we will restrict ourselves to using 'NOT', 'AND', 'OR', and 'MAJ' in a few simple applications

LOGICAL OPERATIONS ON IMAGES

- Let $\mathbf{I}_1$, $\mathbf{I}_2$, ..., $\mathbf{I}_n$ be binary images
- We define logical operations on images on a point-wise basis

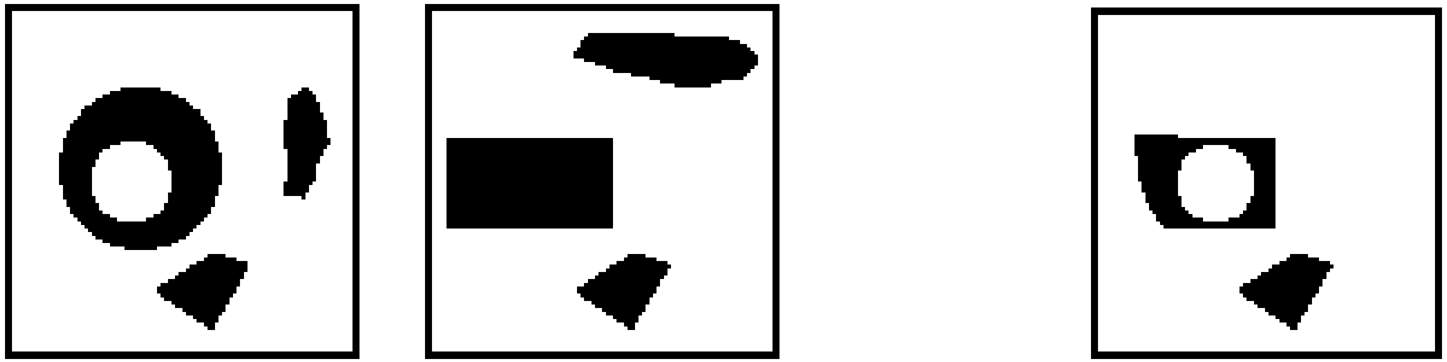
The complement of an image:

- $\mathbf{J}_1$ = NOT($\mathbf{I}_1$) if J_1 (i, j) = NOT[I_1 (i, j)] for all (i, j)
- This reverses the **contrast** - it creates a **binary negative**:



BINARY AND

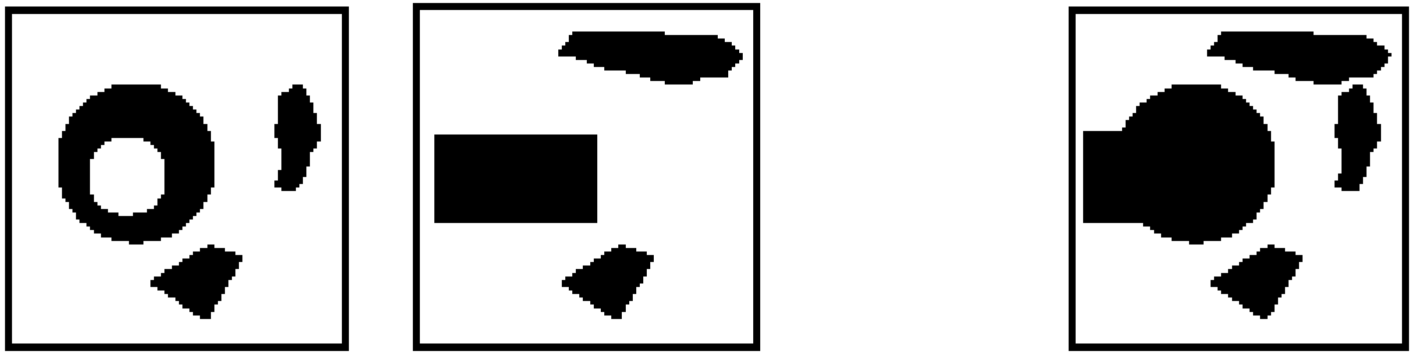
- The AND or intersection of two images:
- $\mathbf{J}_2 = \mathrm{AND}(\mathbf{I}_1 , \mathbf{I}_2) = \mathbf{I}_1 \wedge \mathbf{I}_2$ if $J_2 (i, j) = \mathrm{AND}[\ I_1 (i, j), I_2 (i, j)\ ]$ for all (i, j)
- Shows the overlap of BLACK regions in $\mathbf{I}_1$ and $\mathbf{I}_2$



$\mathbf{I}_1$ $\mathbf{I}_2$ $\mathbf{J}_2 = \mathbf{I}_1 \wedge \mathbf{I}_2$

BINARY OR

- The OR or union of two images:
- $\mathbf{J}_3 = \mathrm{OR}(\mathbf{I}_1 , \mathbf{I}_2) = \mathbf{I}_1 \vee \mathbf{I}_2$
  if $J_3 (i, j) = \mathrm{OR}[I_1 (i, j), I_2 (i, j)]$ for all (i, j)
- Shows the **overlap** of the WHITE regions in $\mathbf{I}_1$ and $\mathbf{I}_2$



$\mathbf{I}_1$ $\mathbf{I}_2$ $\mathbf{J}_3 = \mathbf{I}_1 \vee \mathbf{I}_2$

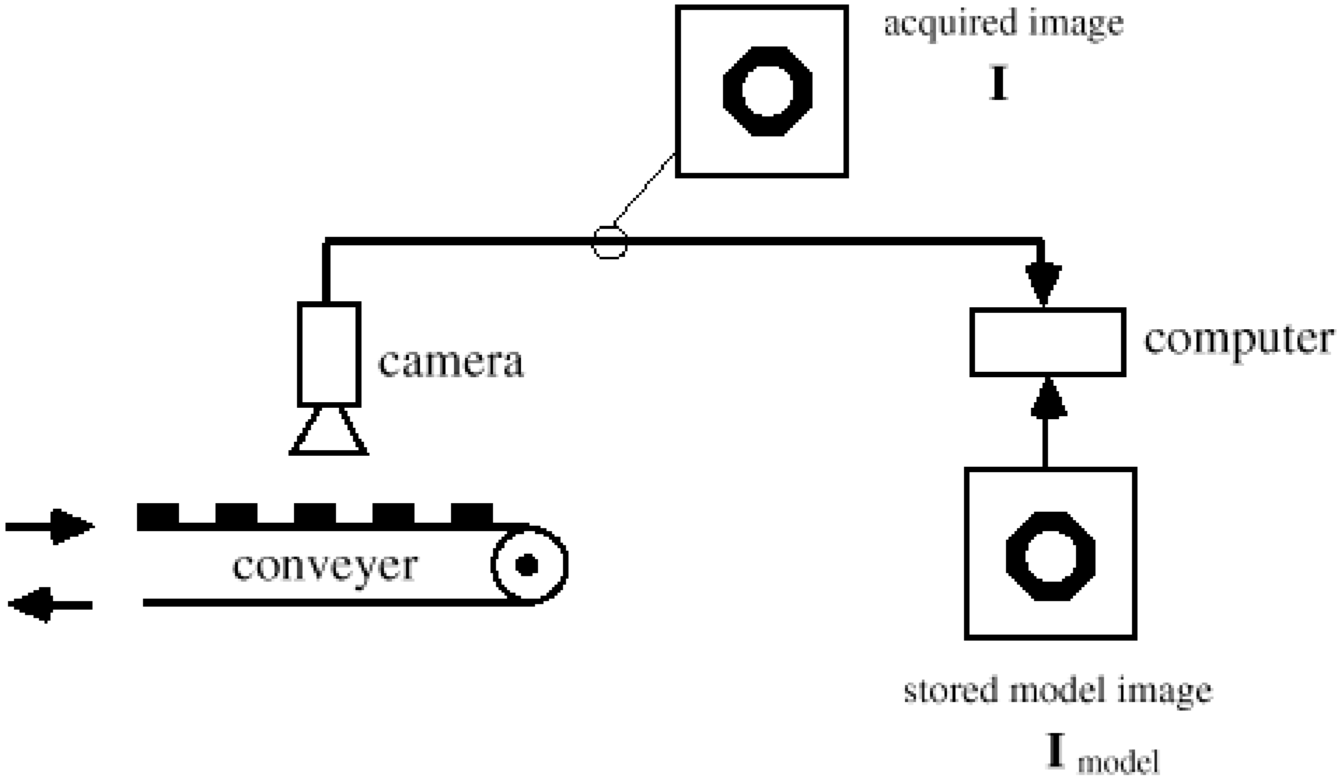
BINARY OPERATIONS

Comments

- The usefulness of **globally** applying AND, OR and MAJ to images is very limited.
- Later, we will find that AND, OR, and MAJ are very useful when applied to **small, local image regions**
- There are exceptions ...

EXAMPLE

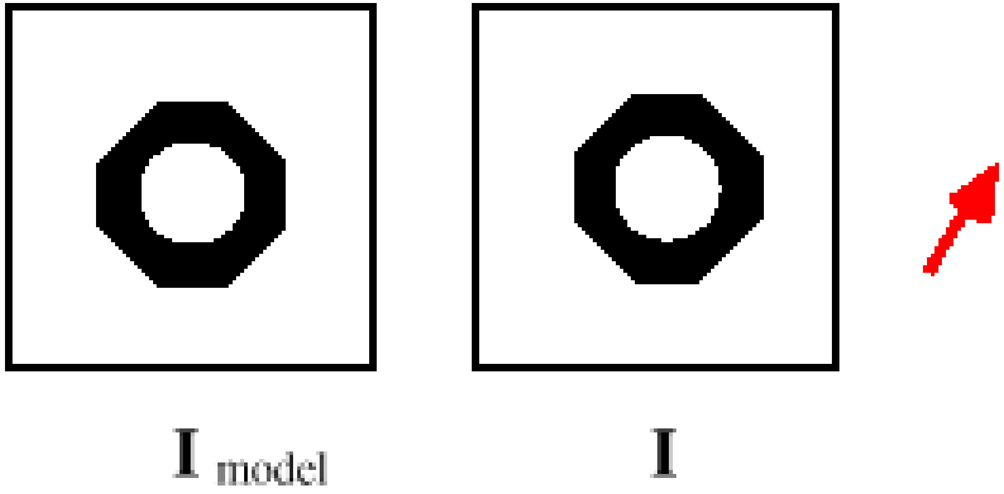
- An assembly-line image inspection system. Similar to many marketed by industry:



- **Objective**: Numerically compare the stored image $\mathbf{I}_{\text{model}}$ and the acquired image **I**

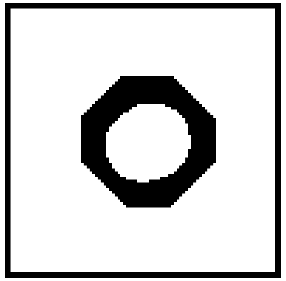
EXAMPLE

- Observe that the object in **I** has been shifted very slightly



Logical AND

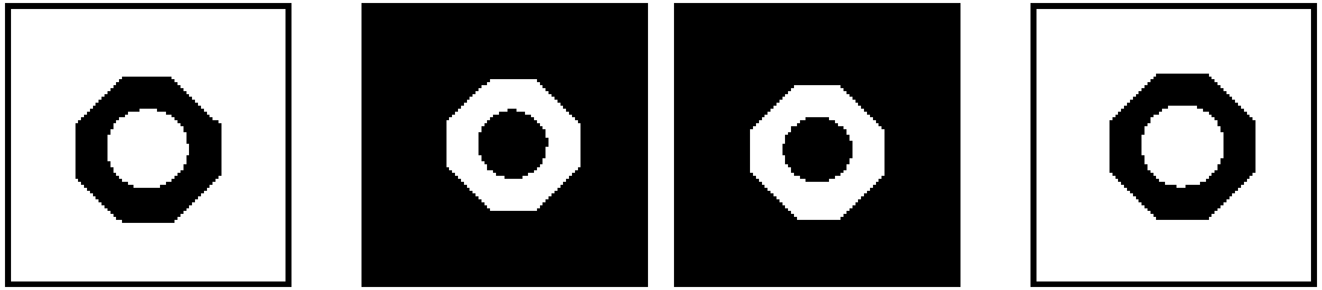
- The logical AND conveys the **overlap**



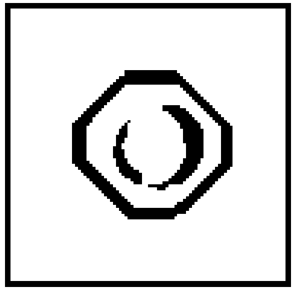
$\mathbf{I}_{\text{model}} \wedge \mathbf{I}$

- A measurement of the **displacement** is given by:
- $\text{XOR}(\mathbf{I}, \mathbf{I}_{\text{model}}) = \text{OR}\{\ \text{AND}[\mathbf{I}_{\text{model}}\ ,\ \text{NOT}(\mathbf{I})],\ \text{AND}[\text{NOT}(\mathbf{I}_{\text{model}}\ ),\ \mathbf{I}\ ]\}$

DISPLACEMENT



$\mathbf{I}_{\text{model}}$ NOT($\mathbf{I}$) NOT($\mathbf{I}_{\text{model}}$) $\mathbf{I}$



XOR($\mathbf{I}$, $\mathbf{I}_{\text{model}}$)

EXAMPLE

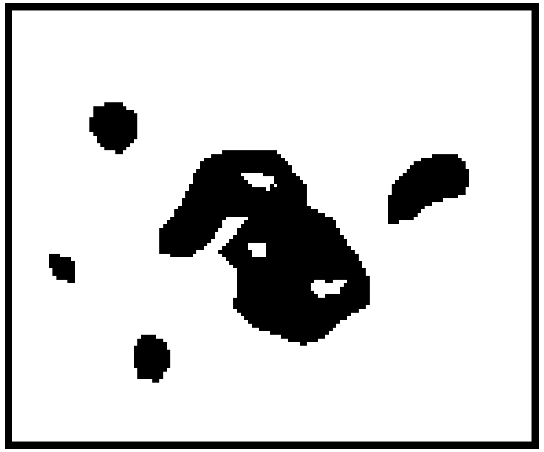
- XOR shows where the displacement errors occur
- To decide if there is a problem or flaw, the ratio or percentage

PERCENT = [# black pixels in XOR($\mathbf{I}$, $\mathbf{I}_{model}$)] / [# black pixels in $\mathbf{I}_{model}$]

- may be compared to a pre-determined tolerance percentage P
- If PERCENT > P, then the part may be **flawed or incorrectly placed**

BLOB COLORING

- A simple technique for **region classification** and **correction**
- **Motivation**: Gray-level image thresholding **usually** produces an imperfect binary image:
  - Extraneous blobs or holes due to noise
  - Extraneous blobs from thresholded objects of little interest
  - Nonuniform object/background surface reflectances

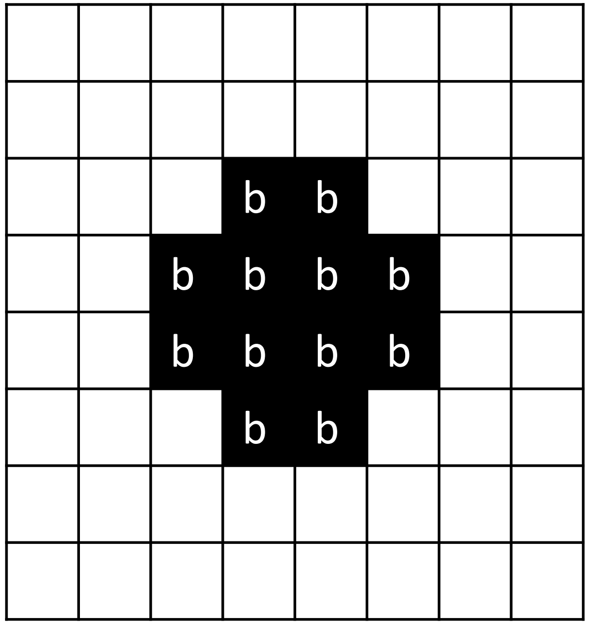


typical thresholded
image result

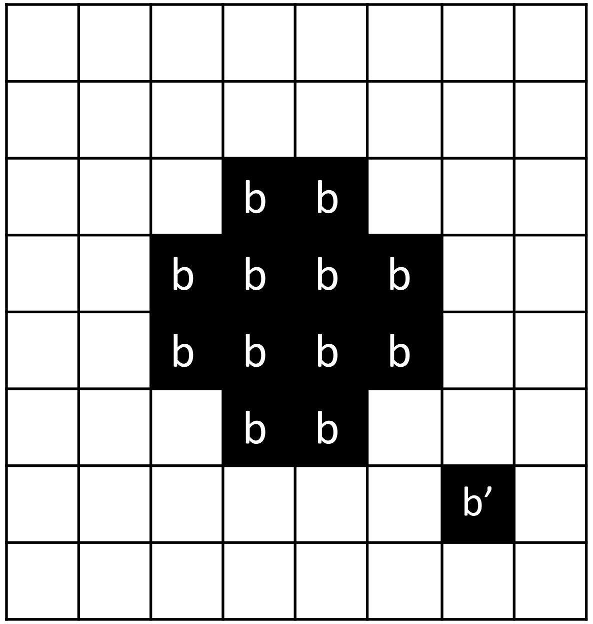
BLOB COLORING

- It is usually desired to **extract** a **small number of objects** or even a **single object** by thresholding
- Blob coloring is a very simple technique for **listing** all of the blobs or objects in a binary image

BLOB COLORING

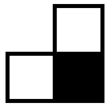


BLOB COLORING

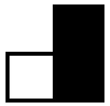


BLOB COLORING ALGORITHM

- For binary image **I**, define a "region color" array **R**:
- R(i, j) = **region number** of pixel I(i, j)
- Set **R** = **0** (all zeros) and k = 1 (k = region number counter)
- While scanning the image left-to-right and top-to-bottom **do**
  - if I(i, j) = 1 and I(i, j-1) = 0 and I(i-1, j) = 0 then
    - set R(i, j) = k and k = k + 1;

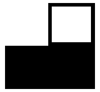


  - if I(i, j) = 1 and I(i, j-1) = 0 and I(i-1, j) = 1 then
    - set R(i, j) = R(i-1, j);

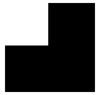


BLOB COLORING ALGORITHM (contd.)

- if $I(i, j) = 1$ and $I(i, j-1) = 1$ and $I(i-1, j) = 0$ then
  - set $R(i, j) = R(i, j-1)$;



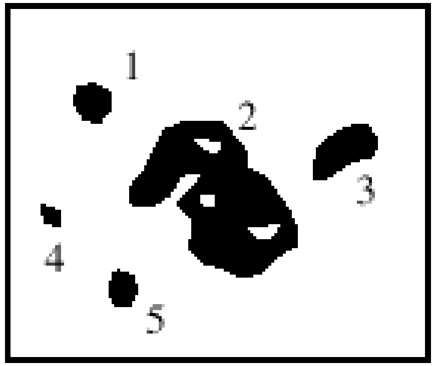
- if $I(i, j) = 1$ and $I(i, j-1) = 1$ and $I(i-1, j) = 1$ then
  - set $R(i, j) = R(i-1, j)$;
  - if $R(i, j-1)$ =/= $R(i-1, j)$ then
  - record $R(i, j-1)$ and $R(i-1, j)$ as equivalent (same color)



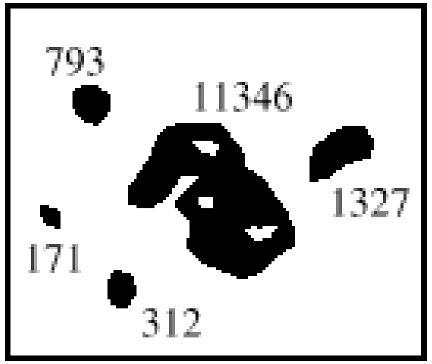
- Distinct integers or "colors" k are assigned to each blob
- Counting the pixels in each blob (by color) is then simple

EXAMPLE

- Using blob coloring



blob coloring result

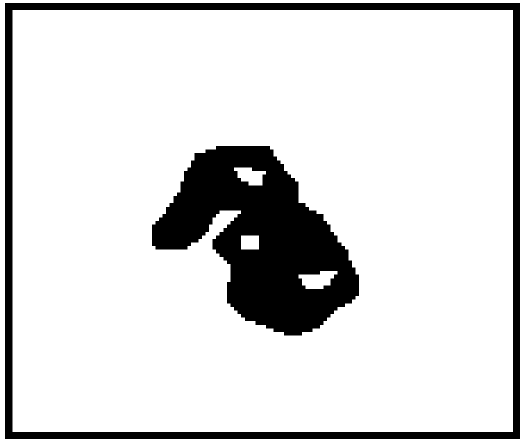


blob counting result

- "Color" of largest blob: **2**

REMOVING MINOR REGIONS

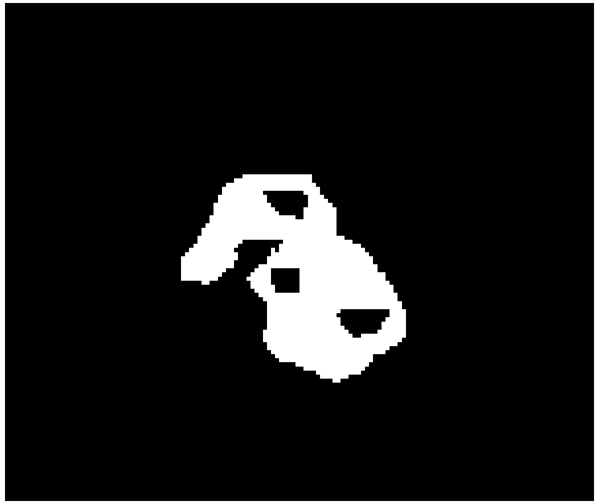
- Let m = "color" of largest region
- While scanning the image left-to-right and top-to-bottom **do**
- if I(i, j) = 1 and R(i, j) != m then
- set I(i, j) = 0;



minor region
removal

EXAMPLE

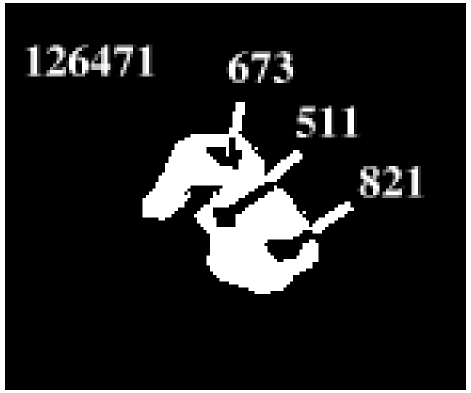
- The process is not complete! To obtain a cohesive, connected object, repeat the procedure on the WHITE pixels
- **Complement** the last result:



complement

EXAMPLE

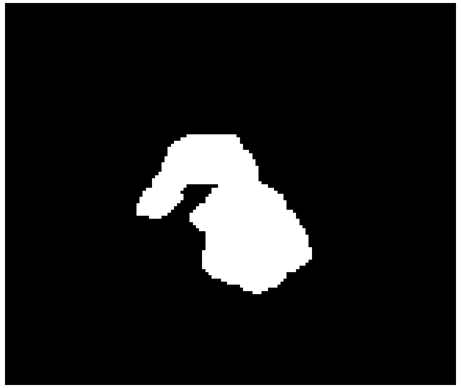
- Then apply all the same steps:



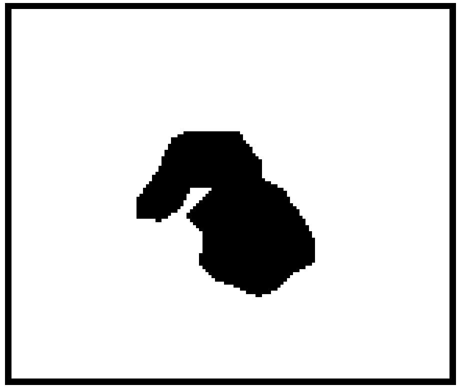
blob counting

- "Color" of largest blob: 1

EXAMPLE



minor region removal



complement

- Simple and effective, but doesn't "cure" everything