Binary Image Processing
Binary Images

• “Binary” means
  – 0 or 1 values only
  – Also called “logical” type (true/false)

• Obtained from
  – Thresholding gray level images
  – Or, the result of feature detectors

• Often want to count or measure shape of 2D binary image regions
Thresholding

- Convert gray scale image to binary (0s and 1s)
- Simplifies processing and computation of features
- Can use a single threshold value (global) or a local value (adaptive)

Thresholding in Matlab:

```
B = I > t;
```
Otsu’s Method for Global Thresholding

• Choose threshold to minimize the variance within groups
  \[ \sigma_w^2 = P_1 \sigma_1^2 + P_2 \sigma_2^2 \]

• Or equivalently, maximize the variance between groups
  \[ \sigma_B^2 = P_1 (m_1 - m_G)^2 + P_2 (m_2 - m_G)^2 \]

• where
  \[ P_1 = \sum_{i=0}^{k} p_i, \quad P_2 = \sum_{i=k+1}^{L-1} p_i \]

• \( m_G \) is the global mean; \( m_k \) is the mean of class \( k \)

Used in Matlab’s “graythresh” function
Matlab Examples

- Images
  - cameraman.tif, eight.tif, coins.png

- Functions
  - \( t = \text{graythresh}(I) \) % Otsu algorithm
  - \( BW = \text{im2bw}(I,t); \) % performs thresholding
Connected Components

• Define adjacency
  – 4-adjacent
  – 8-adjacent

• Two pixels are connected in S if there is a path between them consisting entirely of pixels in S

• S is a (4- or 8-) connected component ("blob") if there exists a path between every pair of pixels

• "Labeling" is the process of assigning the same label number to every pixel in a connected component
Example

- Hand label simple binary image

![Binary image](image1.png) ![Labeled image (4-connected)](image2.png) ![Labeled image (8-connected)](image3.png)

4-connected

8-connected
A Fast Labeling Algorithm

- One pass through image to assign temporary labels and record equivalences
- Second pass to replace temporary labels with equivalence labels
- Let
  - \( B(r,c) \) is the input binary image
  - \( L(r,c) \) is the output image of labels

Side note – faster labeling algorithms do exist. They use 2x2 blocks to search for connected components and use the fact that all the pixels within the block are 8-connected.

Let: \( B(\text{MAXROWS, MAXCOLS}) \) be the input binary image
Also, \( L(\text{MAXROWS, MAXCOLS}) \) will be the output label image

for \( r=1 \) to \( \text{MAXROWs} \)
  for \( c=1 \) to \( \text{MAXCOLS} \)
    if \( B(r,c) == 0 \) then
      \( L(r,c) = 0; \) % if pixel not white, assign no label
    else
      if \( B(r-1,c)==0 \) && \( B(r,c-1)==0 \)
        \( L(r,c) = \text{NUMLABEL}++; \) % New component label
      else if \( B(r-1,c)==1 \) && \( B(r,c-1)==0 \)
        \( L(r,c) = L(r-1,c); \) % Use label from above neighbor
      else if \( B(r,c-1)==1 \) && \( B(r-1,c)==0 \)
        \( L(r,c) = L(r,c-1); \) % Use label from left neighbor
      else if \( B(r,c-1)==1 \) && \( B(r-1,c)==1 \)
        \( L(r,c) = L(r-1,c); \) % Use either neighbor’s label
        record equivalence of \( L(r,c-1) \) and \( L(r-1,c) \)
    end
  end
end

Then go through \( L \) and replace temporary labels with equivalence labels
Example

Binary image

Temporary labels after 1\textsuperscript{st} pass

Final (equivalence) labels after 2\textsuperscript{nd} pass
Matlab Example

- Labeling connected components (white blobs)
  - `im2bw`
    - threshold to convert to binary image
  - `bwlabel`
    - do connected component labeling
    - generate an image of labels
  - `label2rgb`
    - for visualization
    - converts each label to a random color

- If we want to find black blobs
  - `B=~B` or `B=imcomplement(B)`;
    - Flip black and white regions
  - then repeat steps
>> l = imread('Fig9.16(a).jpg');
>> imshow(l,[])
>> whos
   Name     Size         Bytes  Class  Attributes
  I  512x512      262144  uint8

>> BW = im2bw(l, graythresh(l));
>> figure, imshow(BW)
>> [L,num] = bwlabel(BW);
>> figure, imshow(L,[])
>> num

   num =

        17

>> RGB = label2rgb(L);
>> figure, imshow(RGB)
>> BW = imcomplement(BW);
>> [L,num] = bwlabel(BW);
>> RGB = label2rgb(L);
>> figure, imshow(RGB)
Binary Image Morphology

• Operations on binary images:
  – dilation and erosion
  – opening and closing

• Can be used to “clean up” image
  – shrink and expand regions
  – eliminate small regions or holes

• Operations are performed with a “structuring element” S
  – a small binary image
  – like a filter mask
Dilation

- Defined as
  \[ B \oplus S = \bigcup_{b \in B} S_b \]
- where
  - \( S_b \) is the structuring element \( S \), shifted to \( b \)
- Procedure
  - Sweep \( S \) over \( B \)
  - Everywhere the origin of \( S \) touches a 1, OR \( S \) with the result image
- Expands regions
Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

**FIGURE 9.5**
(a) Sample text of poor resolution with broken characters (magnified view).
(b) Structuring element.
(c) Dilation of (a) by (b). Broken segments were joined.
Erosion

• Defined as
  \[ B \Theta S = \{ b \mid b + s \in B, \forall s \in S \} \]

• Procedure
  – Sweep S over B
  – Everywhere S is completely contained in B, output a 1 at the origin of S

• Shrinks regions
Matlab

• To create a structuring element
  \[ S = \text{strel}('disk', 5) \]% disk shape, radius=5
  – Note that this creates a rounder shape:
    \[ S = \text{strel}('disk', 5, 0); \]

• Dilation
  \[ I_2 = \text{imdilate}(I,S); \]

• Erosion
  \[ I_2 = \text{imerode}(I,S); \]
Openings and Closings

• Opening
  – Erosion followed by dilation
  – Eliminate small regions and projections

\[ B \circ S = (B \Theta S) \oplus S \]

• Closing
  – Dilation followed by erosion
  – Fill in small holes and gaps

\[ B \bullet S = (B \oplus S) \Theta S \]

• Matlab functions: imopen, imclose
Example

• Segment all the dark regions in the lower half of the image “bag.png”
  – Namely, generate a binary (or “logical”) image which is white (1) in the regions of interest, and black (0) elsewhere
  – Want:
    • No gaps in the regions
    • No extraneous white pixels in the background

• Then do connected component labeling on these regions

Image “bag.png” from Matlab image processing toolbox demo folder

For this example, ignore the upper half of the image
Approach

• Threshold the image
  – Use “graythresh” to pick the threshold automatically
    \[
    B = \text{im2bw}(I, \text{graythresh}(I));
    \]

• Complement the resulting binary image
  \[
  B = \sim B; \quad \% \text{Make the target regions white}
  \]

• Clean up
  – Do an opening to get rid of small noise white regions
    \[
    B_2 = \text{imopen}(B, \text{SA});
    \]
  – where \( \text{SA} = \text{strel}('disk', \text{radius}, 0); \)
  – Do a closing to fill in gaps in the target regions
    \[
    B_3 = \text{imclose}(B_2, \text{SB});
    \]

• Do connected component labeling
  \[
  [L, n] = \text{bwlabel}(B_3);
  \]
Example (continued)

Original "bag.png" image

Segmented binary image after thresholding and morphological operations

After connected component labeling. Each color indicates a different label of a foreground object (and white represents the background)
Region Properties

• Basic features
  – Area \[ A = \sum_{(r,c) \in R} 1 \]
  – Centroid \[ \bar{r} = \frac{1}{A} \sum_{(r,c) \in R} r, \quad \bar{c} = \frac{1}{A} \sum_{(r,c) \in R} c \]
  – Bounding box
    • The smallest rectangle containing the region
    • Can be specified by
      – The location of the upper left corner
      – The width and height

• Matlab function \texttt{regionprops(L)}
  – This function computes region properties
  – You pass in a “label” image, produced by “\texttt{bwlabel}”
  – It returns an array of structures – each contains the properties for one region
Matlab Structures

• A structure is an object with named "fields"

• Example

```matlab
>> I = imread('Fig9.16(a).jpg');
>> BW = im2bw(I);
>> [L,n] = bwlabel(BW);
>> blobs = regionprops(L);

>> blobs
blobs =
17x1 struct array with fields:
    Area
    Centroid
    BoundingBox
```

• Access fields using: `structurename.fieldname`

```matlab
>> blobs(1)
ans =
    Area: 2058
    Centroid: [15.7216 179.8717]
    BoundingBox: [0.5000 133.5000 34 93]
```
Properties from “regionprops”

```matlab
>> blobs(5)
an =

    Area: 2369
    Centroid: [93.5293 157.4690]
    BoundingBox: [65.5000 129.5000 57 57]
```

- **Centroid** is represented as \([xc, yc]\)
- **Bounding box** is represented as \([x0 y0 w h]\), where
  - \(x0, y0\) are the coordinates of the upper left point
  - \(w, h\) are the width and height
Matlab Graphics

- To draw a rectangle on an image:
  
  ```matlab
  rectangle('Position', [x y w h]), 'EdgeColor', 'r');
  ```

- To draw a line on an image:
  
  ```matlab
  line([x1 x2], [y1 y2], 'Color', 'r');
  ```
Matlab Example

- Draw bounding box around all blobs
- Draw cross hairs on the centroids

```matlab
clear all
close all

I = imread('Fig9.16(a).jpg');
B = im2bw(I, graythresh(I));    % Threshold image
imshow(B);
L = bwlabel(B);     % Do connected component labeling
blobs = regionprops(L); % Get region properties
for i=1:length(blobs)
    % Draw a rectangle around each blob
    rectangle('Position', blobs(i).BoundingBox, 'EdgeColor', 'r');
    % Draw crosshair at center of each blob
    c = blobs(i).Centroid; % Get centroid of blob
    line([c(1)-5 c(1)+5], [c(2) c(2)], 'Color', 'g');
    line([c(1) c(1)], [c(2)-5 c(2)+5], 'Color', 'g');
end
```
Concentric contrasting circle (CCC) target

• The target is a white ring surrounding a black dot

• This feature is fairly unique in the image, because the centroid of the white ring will coincide with the centroid of the black dot

• You can automatically find the target by finding a white region whose centroid coincides with the centroid of a black region

image “robot.jpg” on course website
CCC targets (continued)

• For more discrimination power, you can also place constraints on the binary regions, e.g.,
  – The white area must be > the black area
  – The white bounding box must enclose the black bounding box

• Sometimes local thresholding is better than global thresholding; especially when the lighting varies across the image

• For these simple black-and-white targets, a easy way to do local thresholding is:
  – Apply an averaging filter to the image (whose size is greater than the size of the target). This creates an image of local means.
  – Subtract the local average image from the original.
  – Threshold at zero.

\[
\begin{align*}
I &= \text{double}(I); \quad % \text{Convert image to double} \\
I2 &= \text{imfilter}(I, \text{fspecial('average', sz)}); \\
\text{Idiff} &= I - I2; \\
% \text{Segment white blobs} \\
\text{W} &= \text{Idiff}>0; \\
\text{figure, imshow(W, [])};
\end{align*}
\]