Binary Image Processing

Examples
Example

- Label connected components

assuming 4-connected
assuming 8-connected
Example

- Label connected components

assuming 4-connected

assuming 8-connected
Example

• Do erosion, then dilation (ie, opening) with structuring element
Example

- Do erosion, then dilation (i.e., opening) with structuring element

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]
Example

• Task:
  – Segment coins from the background
  – Namely, generate a binary (or “logical”) image which is white (1) where there are coins, and black (0) elsewhere
  – Want:
    • No gaps in the coins
    • No extraneous white pixels in the background

Image “eight.tif”
clear all
close all

I = imread('eight.tif');
imshow(I,[]);

B = im2bw(I, graythresh(I));    % threshold
B = imcomplement(B);            % we want black regions

S = strel('disk',1);        % define a small structuring element
B1 = imopen(B,S);       % get rid of small white regions

S = strel('disk', 5, 0);    % Need structuring element bigger than gaps
B2 = imclose(B1,S);     % Fill in gaps
figure, imshow(B2);

[L,n] = bwlabel(B2);     % find connected components
fprintf('n = %d
', n);
figure, imshow(L, []);
figure, imshow(L, []);

RGB = label2rgb(L);     % create false color image for visualization
figure, imshow(RGB);
original image

thresholded image, after morphological operations

labeled image

labeled image with false colors
Example

• Tasks
  – count number of balls
  – draw rectangles and/or crosshairs to mark each ball
  – find the average of the ball locations

This is image “D67.gif” from http://www.ux.uis.no/~tranden/brodatz.html. It is from the “Brodatz” image texture database, which is often used to test computer vision algorithms.
clear all
close all

I = imread('balls.gif');
imshow(I,[])

B = im2bw(I, graythresh(I));  % Threshold
B = ~B;  % Complement image

S = strel('disk', 5, 0);

I2 = imerode(B,S);
figure, imshow(I2);

[L, n] = bwlabel(I2);

blobs = regionprops(L);

for i=1:n
    rectangle('Position', blobs(i).BoundingBox, 'EdgeColor', 'r');
end

allCentroids = cat(1,blobs(:,).Centroid);  % Get all centroids

avgCentroid = mean(allCentroids);  % Get the average x,y
xavg = avgCentroid(1);
yavg = avgCentroid(2);

% Display crosshair at the average
figure(1);
line([xavg-10 xavg+10], [yavg yavg], 'Color', 'r');
line([xavg xavg], [yavg-10 yavg+10], 'Color', 'r');
Finding CCC Targets

• **Approach:**
  - Threshold the image
    • For most images, a global threshold is fine (ie, the Otsu algorithm).
    • For images with poor contrast or where the target is small, local thresholding may be needed.

```matlab
clear all
close all

I = imread('robot.jpg');
imshow(I,
I = rgb2gray(I);

% Global threshold
W = im2bw(I, graythresh(I));  % White regions
```
Finding CCC Targets (continued)

- If there are a lot of tiny (noise) white regions, you may need to “clean up” the thresholded image.
  - Otherwise you will have too many regions to process.
- You can use morphological “opening” to do this.
  - Use a structuring element that is larger than the noise regions.
    
    ```matlab
    S = strel('disk', radius, 0);
    W2 = imopen(W, S);
    figure, imshow(W2);
    ```

- Then, find connected components and extract their region properties (e.g., centroid, area, bounding box)

  ```matlab
  [LW, nw] = bwhlabel(W2); % Get labeled image LW
  blobsWhite = regionprops(LW); % All white blobs
  ```
Finding CCC Targets (continued)

- Complement the image, and repeat the process to find all black blobs.

- Now check every possible pair of white and black blobs. Pseudocode:

```plaintext
for each white blob
    for each black blob
        Get centroid of white blob
        Get centroid of black blob
        if distance between centroids < thresh
            % We have a possible CCC; draw a
            % crosshair at its centroid or draw a
            % rectangle around its bounding box
            end if
    end for
end for
```
Finding CCC Targets (continued)

• To find the distance between two centroids, you can use Matlab’s “norm” function.
  – “norm” computes the norm of a vector, which is its length

```matlab
bc = blobsBlack(ib).Centroid; % Get black blob centroid
wc = blobsWhite(iw).Centroid; % Get white blob centroid
if norm(bc-wc) < thresh
  % Got a possible CCC; draw a rectangle
  :
```

If additional tests are needed

• You can check if the black bounding box encloses the white bounding box

• A bounding box is represented as $[x_0 \ y_0 \ w \ h]$, where
  – $x_0, y_0$ are the coordinates of the upper left point
  – $w, h$ are the width and height

\[
\begin{align*}
\text{>> blobsWhite(1).BoundingBox} \\
\text{ans} = \\
0.5000 & \quad 59.5000 & \quad 175.0000 & \quad 273.0000
\end{align*}
\]
If additional tests are needed

- You can check the “circularity” ratio of the combined (white and black) region.
- The circularity ratio is defined as \[ R = \frac{4\pi \cdot \text{area}}{\text{perim}^2} \]
- For a perfect circle, \( R = 1 \)
  - All other shapes have \( R \) less than 1
  - For example, a square has \( R = \frac{\pi}{4} \approx 0.78 \)
  - So you might check if \( R \) > than some minimum value like 0.78

- Note – by default, “bwlabel” just computes area, centroid, and bounding box.
- To have it compute “Perimeter” too, use:

```matlab
blobs = regionprops(L, 'Area', 'BoundingBox', 'Centroid', 'Perimeter');
```