Lecture 3: Binary image analysis

Thursday, Sept 6
• Sudheendra’s office hours
  – Mon, Wed 1-2 pm
  – ENS 31NR

• Forsyth and Ponce book
Binary images

- Two pixel values
- Foreground and background
- Regions of interest
Constrained image capture setting

Fig. 3 Schematic diagram of marking inspection setup at Texas Instruments

R. Nagarajan et al. A real time marking inspection scheme for semiconductor industries, 2006
Medical, bio data
Intermediate low-level cues

- Edges
- Motion
- Orientation

NASA robonaut
http://robonaut.jsc.nasa.gov/status/October_prime.htm
Outline

• Thresholding
• Connected components
• Morphological operators
• Region properties
  – Spatial moments
  – Shape
• Distance transforms
  – Chamfer distance
Thresholding

• Grayscale -> binary mask
• Useful if object of interest’s intensity distribution is distinct from background

\[ F_T[i, j] = \begin{cases} 
1 & \text{if } F[i, j] \geq T \\
0 & \text{otherwise.} 
\end{cases} \]

\[ F_T[i, j] = \begin{cases} 
1 & \text{if } T_1 \leq F[i, j] \leq T_2 \\
0 & \text{otherwise.} 
\end{cases} \]

\[ F_T[i, j] = \begin{cases} 
1 & \text{if } F[i, j] \in Z \\
0 & \text{otherwise.} 
\end{cases} \]

Selecting thresholds

- Partition a bimodal histogram
- Fit Gaussians
- Dynamic or local thresholds
A nice case: bimodal intensity histograms

Ideal histogram, light object on dark background

Actual observed histogram with noise

A nice case: bimodal intensity histograms

- Example
- Thresholding a bimodal histogram

- Otsu method (1979) : automatically select threshold by minimizing the weighted within-group variance of the two groups of pixels separated by the threshold.
Not so nice cases

- Threshold selection is an art, not a science
Connected components

• Identify distinct regions

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

a) binary image

1 1 0 1 1 1 0 2
1 1 0 1 0 1 0 2
1 1 1 1 0 0 0 2
0 0 0 0 0 0 0 2
3 3 3 3 0 4 0 2
0 0 3 0 4 0 2
5 5 0 3 0 0 0 2
5 5 0 3 0 2 2 2

b) connected components labeling

c) binary image and labeling, expanded for viewing

Shapiro and Stockman
Connected components

Connected components of 1’s from thresholded image

Connected components of cluster labels
Connectedness

- Which pixels are considered neighbors

4-connected

8-connected

Image from http://www-ee.uta.edu/Online/Devarajan/ee6358/BIP.pdf
Connected components

• Various algorithms to compute
  – Recursive (in memory)
  – Two rows at a time (image not necessarily in memory)
  – Parallel propagation strategy
Recursive connected components

• Find an unlabeled pixel, assign it a new label
• Search to find its neighbors, and recursively repeat to find their neighbors til there are no more
• Repeat

• Demo http://www.cosc.canterbury.ac.nz/mukundan/covn/Label.html
Sequential connected components

- Labeling a pixel only requires to consider its prior and superior neighbors.
- It depends on the type of connectivity used for foreground (4-connectivity here).

Same object  New object
(a)    (b)    (c)    (d)

What happens in these cases?

Equivalence table

Slide from J. Neira
• Process the image from left to right, top to bottom.

1. If the next pixel to process is 1-pixel:
   - Already processed

2. If both are, and have the same label, copy it.

3. If they have different labels:
   - superior? smallest?
     - 1. Copy the label from the prior.
     - 2. Reflect the change in the table of equivalences.

4. Otw, assign a new label.


• Re-label with the smallest of equivalent labels.

• Pixels of the same segment always have the same label.
Morphological operators

- Dilation
- Erosion
- Open, close
Dilation

- Expands connected components
- Grow features
- Fill holes

Before dilation

After dilation
Structuring elements

• Masks of varying shapes used to perform morphology

• Scan mask across foreground pixels to transform the binary image
Dilation / Erosion

• Dilation: if current pixel is foreground, set all pixels under S to foreground in output (OR)

• Erosion: if every pixel under S is foreground, leave as is; otherwise, set current pixel to background in output
Example for Dilation (1D)

Input image: 1 0 0 0 1 1 1 0 1 1

Structuring Element: 1 1 1

Output Image: 1 1

\[ g(x) = f(x) \oplus SE \]

Adapted from T. Moeslund
Example for Dilation

Input image:
1 0 0 0 1 1 1 0 1 1

Structuring Element:
1 1 1

Output Image:
1 1
Example for Dilation

Input image:

```
1 0 0 0 1 1 1 0 1 1
```

Structuring Element:

```
1 1 1
```

Output Image:

```
1 1 0
```
**Example for Dilation**

<table>
<thead>
<tr>
<th>Input image</th>
<th>1 0 0 0 1 1 1 0 1 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structuring Element</td>
<td>1 1 1</td>
</tr>
<tr>
<td>Output Image</td>
<td>1 1 0 0</td>
</tr>
</tbody>
</table>
## Example for Dilation

<table>
<thead>
<tr>
<th>Input image</th>
<th>1 0 0 0 1 1 1 0 1 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Structuring Element</td>
<td>1 1 1</td>
</tr>
<tr>
<td>Output Image</td>
<td>1 1 0 1 1 1</td>
</tr>
</tbody>
</table>

The process of dilation involves applying the structuring element to the input image, resulting in the output image.
Example for Dilation

Input image:

```
1 0 0 0 1 1 1 1 0 1 1
```

Structuring Element:

```
1 1 1 1
```

Output Image:

```
1 1 0 1 1 1 1
```
Example for Dilation

Input image

| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |

Structuring Element

| 1 | 1 | 1 |

Output Image

| 1 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
Example for Dilation

Input image: 1 0 0 0 1 1 1 0 1 1

Structuring Element: 1 1 1

Output Image: 1 1 0 1 1 1 1 1 1
Example for Dilation

Input image

```
1 0 0 0 1 1 1 0 1 1
```

Structuring Element

```
1 1 1
```

Output Image

```
1 1 0 1 1 1 1 1 1 1
```

The object gets bigger and holes are filled!
Erosion

- Erode connected components
- Shrink features
- Remove bridges, branches, noise

Before erosion  After erosion
Example for Erosion (1D)

Input image

Structuring Element

Output Image

\[ g(x) = f(x) \Theta SE \]
Example for Erosion (1D)

Input image
1 0 0 0 1 1 1 0 1 1

Structuring Element
1 1 1

Output Image
0 0

\[ g(x) = f(x) \Theta SE \]
Example for Erosion

Input image: 1 0 0 0 1 1 1 0 1 1

Structuring Element: 1 1 1

Output Image: 0 0 0
Example for Erosion

Input image: 1 0 0 0 1 1 1 0 1 1

Structuring Element: 1 1 1

Output Image: 0 0 0 0
Example for Erosion

Input image

1 0 0 0 1 1 1 0 1 1

Structuring Element

1 1 1

Output Image

0 0 0 0 0
Example for Erosion

Input image

1 0 0 0 1 1 1 1 0 1 1

Structuring Element

1 1 1 1

Output Image

0 0 0 0 0 0 1
Example for Erosion

Input image

1 0 0 0 1 1 1 0 1 1

Structuring Element

1 1 1

Output Image

0 0 0 0 0 1 0
Example for Erosion

Input image

| 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |

Structuring Element

| 1 | 1 | 1 | 1 |

Output Image

| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
Example for Erosion

Input image

1 0 0 0 1 1 1 0 1 1

Structuring Element

1 1 1

Output Image

0 0 0 0 0 0 1 0 0 0 0
Example for Erosion

Input image: 1 0 0 0 1 1 1 0 1 1

Structuring Element: 1 1

Output Image: 0 0 0 0 0 1 0 0 0 1

The object gets smaller
Dilation / Erosion

- **Dilation**: if current pixel is foreground, set all pixels under S to foreground in output (OR)

- **Erosion**: if every pixel under S is foreground, leave as is; otherwise, set current pixel to background in output

Images by P. Duygulu
Opening

- Erode, then dilate
- Remove small objects, keep original shape
Closing

- Dilate, then erode
- Fill holes, but keep original shape
Application: blob tracking

Absolute differences from frame to frame
Erode
Application: blob tracking

• Background subtraction + blob tracking
Application: segmentation of a liver

Application by Jie Zhu, Cornell University

Slide credit: Li Shen
Region properties

Some useful features can be extracted once we have connected components, including

• Area
• Centroid
• Extremal points, bounding box
• Circularity
• Spatial moments
Area and centroid

- We denote the set of pixels in a region by $R$.

- assuming square pixels:
  
  area:
  \[
  A = \sum_{(r,c) \in R} 1
  \]

  centroid:
  \[
  \bar{r} = \frac{1}{A} \sum_{(r,c) \in R} r \\
  \bar{c} = \frac{1}{A} \sum_{(r,c) \in R} c
  \]

- $(\bar{r}, \bar{c})$ is generally not a pair of integers.
A second measure uses variation off of a circle:

circularity (2):

\[ C_2 = \frac{\mu_R}{\sigma_R} \]

where \( \mu_R \) and \( \sigma_R^2 \) are the mean and variance of the distance from the centroid of the shape to the boundary pixels \((r_k, c_k)\).

mean radial distance:

\[ \mu_R = \frac{1}{K} \sum_{k=0}^{K-1} ||(r_k, c_k) - (\bar{r}, \bar{c})|| \]

variance of radial distance:

\[ \sigma_R^2 = \frac{1}{K} \sum_{k=0}^{K-1} [|| (r_k, c_k) - (\bar{r}, \bar{c}) || - \mu_R]^2 \]

[Haralick]
Invariant descriptors

Often want features independent of position, orientation, scale.
Central moments

S is a subset of pixels (region).
Central \( (j,k)^{th} \) moment defined as:

\[
\mu_{jk} = \sum_{(x,y) \in S} (x - \bar{x})^j (y - \bar{y})^k
\]

- Invariant to translation of S.
Central moments

• 2\textsuperscript{nd} central moment: variance
• 3\textsuperscript{rd} central moment: skewness
• 4\textsuperscript{th} central moment: kurtosis
Axis of least second moment

• Invariance to orientation?
  Need a common alignment

Axis for which the squared distance to 2d object points is minimized.
Distance transform

- Image reflecting distance to nearest point in point set (e.g., foreground pixels).
Distance transform

Edge image

Distance transform image
Distance transform (1D)

Two pass O(n) algorithm for 1D $L_1$ norm

1. **Initialize**: For all $j$
   \[ D[j] \leftarrow 1_p[j] \]
2. **Forward**: For $j$ from 1 up to $n-1$
   \[ D[j] \leftarrow \min(D[j], D[j-1]+1) \]
3. **Backward**: For $j$ from $n-2$ down to 0
   \[ D[j] \leftarrow \min(D[j], D[j+1]+1) \]

Adapted from D. Huttenlocher

\[ \begin{array}{cccccccccc}
\infty & 0 & \infty & 0 & \infty & \infty & 0 & \infty \\
\infty & 0 & 1 & 0 & 1 & 2 & 3 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 2 & 1 & 0 & 1
\end{array} \]
Distance Transform (2D)

- 2D case analogous to 1D
  - Initialization
  - Forward and backward pass
    - Fwd pass finds closest above and to left
    - Bwd pass finds closest below and to right
- Note nothing depends on $0, \infty$ form of initialization
  - Can “distance transform” arbitrary array

Adapted from D. Huttenlocher
Chamfer distance

- Average distance to nearest feature

\[ D_{\text{chamfer}}(T, I) \equiv \frac{1}{|T|} \sum_{t \in T} d_I(t) \]
Chamfer distance

More on this and other distances later

Edge image

Distance transform image

D. Gavrila, DAGM 1999
Generalized distance transforms

• Same forward/backward algorithm applicable with different initialization
• Initialize with function values $F(x,y)$:

\[
\text{Generalized Distance Transform} \quad D(p) = \min_{q \in I} (\| p - q \| + F(q))
\]
Distance Transform vs. Generalized Distance Transform

Assuming \( F(p) = \begin{cases} 
0 & \text{if pixel } p \text{ is image feature} \\
\infty & \text{O.W.}
\end{cases} \)

then \( D(p) = \min_{q} \{ \| p - q \| + F(q) \} = \min_{q:F(q)=0} \| p - q \| \)

is standard Distance Transform (of image features)
Distance Transform vs.
Generalized Distance Transform

- For general $F(p)$

$$D(p) = \min_q \{ \| p - q \| + F(q) \}$$

is Generalized Distance Transform of $F(p)$

Location of $q$ is close to $p$, and $F(q)$ is small there

$F(p)$ may represent non-binary image features (e.g. image intensity gradient)

Slide credit Y. Boykov
Binary images

• Pros
  – Can be fast to compute, easy to store
  – Simple processing techniques available
  – Lead to some useful compact shape descriptors

• Cons
  – Hard to get “clean” silhouettes, noise common in realistic scenarios
  – Can be too coarse of a representation
  – Not 3d
Matlab

- \( N = \text{HIST}(Y,M) \)
- \( L = \text{BWLABEL}(BW,N) \);
- \( \text{STATS} = \text{REGIONPROPS}(L,\text{PROPERTIES}) \);
  - 'Area'
  - 'Centroid'
  - 'BoundingBox'
  - 'Orientation', ...
- \( \text{IM2} = \text{imerode}(IM,SE) \);
- \( \text{IM2} = \text{imdilate}(IM,SE) \);
- \( \text{IM2} = \text{imclose}(IM,SE) \);
- \( \text{IM2} = \text{imopen}(IM,SE) \);
- \([D, L] = \text{bwdist}(BW,\text{METHOD}) \);
• Everything is matrix

```matlab
1    %% Matrix Definition 1 %%
2    A = [1 2 3 4; 5 6 7 8];
3
4    %% Matrix Definition 2 %%
5    A = [1:1:4; 5:1:8];
6
7    %% Matrix Definition 3 %%
8    for i = 1:2
9        for j = 1:4
10            A(i,j) = (i-1)*4+j;
11        end
12    end
13 | 
14    %% Matrix Definition 4 %%
15    A = [];
16    A = zeros(n,n); %ones zeros eye rand randn
```

Tutorial adapted from W. Freeman, MIT 6.896
• Matrix index

13 % Matrix index
19 A = magic(4);
20    >> A =
21   16   2   3  13
22   5  11  10   6
23    9   7   6  12
24    4  14  15   1
25 A(1:2, 1:2)
26    >> ans =
27    16   2
28    5  11
29 A(:,1)
30    >> ans =
31    16
32    5
33    9
34    4
35 A(1,:)
36    >> ans =
37    16   2   3  13
38 A([6 7])
39    >> ans =
40    11   7
• Manipulate matrices

```matlab
A = [1 NaN 2 NaN; 3 Inf 4 Inf]; % 1/0=Inf 0/0=NaN
%% replace NaN/Inf with 0
for i = 1:size(A,1)
    for j = 1:size(A,2)
        if isnan(A(i,j)) || isinf(A(i,j))
            A(i,j) = 0;
        end
    end
end
%% 'find'
a = [0 1 0 2; 0 3 0 4]
>> a =
0 1 0 2
0 3 0 4
find(a)
>> ans = [3 4 7 8]'
[ii jj] = find(a);
>> ii = [1 2 1 2]
>> jj = [2 2 4 4]
nanflag = isnan(A)
>> nanflag =
0 1 0 1
0 0 0 0
infflag = isinf(A)
>> infflag =
0 0 0 0
0 1 0 1
```
• Manipulate matrices

```matlab
75 \%\%\% Matrix operation
76 A = [1 2; 3 4];
77 B = [1 1; 1 -1];
78 A*B
79    >> ans =
80     3  -1
81     7  -1
82 A.*B
83    >> ans =
84     1  2
85     3  -4
86 A/B == A * inv(B)
87    >> ans =
88     1.5  -0.5
89     3.5  -0.5
90 A./B
91    >> ans =
92     1  2
93     3  -4
```

```matlab
95 \%\%\% Matrix concatenation
96 A = [1 2]; B = [3 4];
97 C = [A B]
98    >> C =
99     1  2  3  4
100 C = [A;B]
101    >> C =
102     1  2
103     3  4
```

```
A = [1 2];
A*A' = 5
A.*A =
    1  4
sum(A.*A) = 5
```
• **Scripts and functions**
  
  - Scripts are m-files containing MATLAB statements
  
  - Functions are like any other m-file, but they accept arguments
  
  - Name the function file the same as the function name

```matlab
myfunction.m

function y = myfunction(x)
  % Function of one argument with one return value
  a = [-2 -1 0 1];  % Have a global variable of the same name
  y = a + x;
```

```matlab
myotherfunction.m

function [y, z] = myotherfunction(a, b)
  % Function of two arguments with two return values
  y = a + b;
  z = a - b;
```
• Try to code in matrix ways

```matlab
% use for loops
A = [1 2 3 4; 5 6 7 8];
sum(A)
>> ans =
     5  8 10 12
ASUM = sum(A,2)
>> ASUM =
     10
     26

% or use matrix
for i = 1:size(A,1)
    for j = 1:size(A,2)
        APROB(i,j) = A(i,j)/ASUM(i);
    end
end

% or matrix
A = [1 2 3 4; 5 6 7 8];
ASUM = sum(A,2)
APROS = A./repmat(ASUM, 1, size(A,2));
>> ans =
     10 10 10 10
     26 26 26 26
```
• whos
• help
• lookfor
• clear / clear x
• save
• load