Computer Vision - Lecture 2

Binary Image Analysis

25.10.2012

Bastian Leibe
RWTH Aachen
http://www.vision.rwth-aachen.de/
leibe@vision.rwth-aachen.de
Announcements

• Please subscribe to the lecture on the Campus system!
  ➢ Important to get email announcements and L2P access!
  ➢ Bachelor students please also subscribe
Binary Images

- Just two pixel values
- Foreground and background
- Regions of interest

![Binary Image Example]

Slide credit: Kristen Grauman
Uses: Industrial Inspection

R. Nagarajan et al. “A real time marking inspection scheme for semiconductor industries“, 2006

B. Leibe
Uses: Document Analysis, Text Recognition

- Handwritten digits
- Natural text (after detection)
- Scanned documents

Source: Till Quack, Martin Renold
Uses: Medical/Bio Data

Source: D. Kim et al., Cytometry 35(1), 1999
Uses: Blob Tracking & Motion Analysis

Frame Differencing

![Frame Differencing Example](image1)

Background Subtraction

![Background Subtraction Example](image2)
Uses: Shape Analysis, Free-Viewpoint Video

Silhouette

Medial axis

Visual Hull Reconstruction

Blue-c project, ETH Zurich

B. Leibe
Uses: Intensity Based Detection

- Looking for dark pixels...

```matlab
fg_pix = find(im < 65);
```
Uses: Color Based Detection

- Looking for pixels within a certain color range...

\[
fg\_pix = \text{find}(\text{hue} > t1 \land \text{hue} < t2);
\]
Issues

• How to demarcate multiple regions of interest?
  - Count objects
  - Compute further features per object

• What to do with “noisy” binary outputs?
  - Holes
  - Extra small fragments
Outline of Today’s Lecture

• Convert the image into binary form
  ➢ Thresholding

• Clean up the thresholded image
  ➢ Morphological operators

• Extract individual objects
  ➢ Connected Components Labeling

• Describe the objects
  ➢ Region properties
Thresholding

- Grayscale image $\Rightarrow$ Binary mask
- Different variants
  - One-sided
    \[
    F_T[i, j] = \begin{cases} 
    1, & \text{if } F[i, j] \geq T \\
    0, & \text{otherwise}
    \end{cases}
    \]
  - Two-sided
    \[
    F_T[i, j] = \begin{cases} 
    1, & \text{if } T_1 \leq F[i, j] \leq T_2 \\
    0, & \text{otherwise}
    \end{cases}
    \]
  - Set membership
    \[
    F_T[i, j] = \begin{cases} 
    1, & \text{if } F[i, j] \in Z \\
    0, & \text{otherwise}
    \end{cases}
    \]
Selecting Thresholds

- Typical scenario
  - Separate an object from a distinct background

- Try to separate the different grayvalue distributions
  - Partition a bimodal histogram
  - Fit a parametric distribution (e.g. Mixture of Gaussians)
  - Dynamic or local thresholds

- In the following, I will present some simple methods.
- We will see some more general methods in Lecture 6...
A Nice Case: Bimodal Intensity Histograms

Ideal histogram, light object on dark background

Actual observed histogram with noise

Source: Robyn Owens

B. Leibe
Not so Nice Cases...

- How to separate those?

- Threshold selection is difficult in the general case
  - Domain knowledge often helps
  - E.g. Fraction of text on a document page (\(\Rightarrow\) histogram quantile)
  - E.g. Size of objects/structure elements

Source: Shapiro & Stockman
Global Binarization [Otsu’79]

- Search for the threshold $T$ that minimizes the within-class variance $\sigma_{\text{within}}$ of the two classes separated by $T$

\[
\sigma_{\text{within}}^2(T) = n_1(T)\sigma_1^2 + n_2(T)\sigma_2^2(T)
\]

where

\[
n_1(T) = |\{I(x,y) < T\}|, \quad n_2(T) = |\{I(x,y) \geq T\}|
\]

- This is the same as maximizing the between-class variance $\sigma_{\text{between}}$

\[
\sigma_{\text{between}}^2(T) = \sigma^2 - \sigma_{\text{within}}^2(T)
\]

\[
= n_1(T)n_2(T) [\mu_1(T) - \mu_2(T)]^2
\]
Algorithm

Precompute a cumulative grayvalue histogram $h$.

For each potential threshold $T$

1.) Separate the pixels into two clusters according to $T$
2.) Look up $n_1$, $n_2$ in $h$ and compute both cluster means
3.) Compute $\sigma^2_{between}$

Choose

$$T^* = \arg \max_T \left[ \sigma^2_{between}(T) \right]$$
Local Binarization [Niblack’86]

- Estimate a local threshold within a small neighborhood window $W$

\[ T_W = \mu_W + k \cdot \sigma_W \]

where $k \in [-1,0]$ is a user-defined parameter.

Effect:

What is the hidden assumption here?
Effects

Original image

Global threshold selection (Otsu)

Local threshold selection (Niblack)
Additional Improvements

- Document images often contain a smooth gradient

⇒ Try to fit that gradient with a polynomial function

Source: S. Lu & C. Tan, ICDAR’07
Surface Fitting

- Polynomial surface of degree $d$

$$f(x, y) = \sum_{i+j=0}^{d} b_{i,j} x^i y^j$$

- Least-squares estimation, e.g. for $d=3 \ (m=10)$

$$Ab = I$$

\[
\begin{bmatrix}
1 & x_0 & y_0 & x_0^2 & x_0y_0 & \cdots & x_0^3 \\
1 & x_1 & y_1 & x_1^2 & x_1y_1 & \cdots & x_1^3 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\
1 & x_n & y_n & x_n^2 & x_ny_n & \cdots & x_n^3 \\
\end{bmatrix}
\begin{bmatrix}
b_0 \\
b_1 \\
\vdots \\
b_m \\
\end{bmatrix}
= 
\begin{bmatrix}
I_0 \\
I_1 \\
\vdots \\
I_n \\
\end{bmatrix}
\]

Solution with pseudo-inverse:
$$b = (A^T A)^{-1} A^T I$$

Matlab (using SVD):
$$b = I \backslash A$$
Surface Fitting

- Iterative Algorithm
  1.) Fit parametric surface to all points in region.
  2.) Subtract estimated surface.
  3.) Apply global threshold (e.g. with Otsu method)
  4.) Fit surface to all background pixels in original region.
  5.) Subtract estimated surface.
  6.) Apply global threshold (Otsu)
  7.) Iterate further if needed...

- The first pass also takes foreground pixels into account.
  - This is corrected in the following passes.
  - Basic assumption here: most pixels belong to the background.
Outline of Today’s Lecture

• Convert the image into binary form
  ➢ Thresholding

• Clean up the thresholded image
  ➢ Morphological operators

• Extract individual objects
  ➢ Connected Components Labeling

• Describe the objects
  ➢ Region properties

Image Source: D. Kim et al., Cytometry 35(1), 1999
Cleaning the Binarized Results

- Results of thresholding often still contain noise

Necessary cleaning operations
- Remove isolated points and small structures
- Fill holes

⇒ Morphological Operators

Image Source: D. Kim et al., Cytometry 35(1), 1999
Morphological Operators

• Basic idea
  - Scan the image with a structuring element
  - Perform set operations (intersection, union) of image content with structuring element

• Two basic operations
  - Dilation (Matlab: imdilate)
  - Erosion (Matlab:imerode)

• Several important combinations
  - Opening (Matlab: imopen)
  - Closing (Matlab: imclose)
  - Boundary extraction

Matlab:
>> help strel
Dilation

• Definition
  
  ➢ “The dilation of $A$ by $B$ is the set of all displacements $z$, such that $\hat{(B)}_z$ and $A$ overlap by at least one element”.
  
  ➢ $((\hat{B})_z$ is the mirrored version of $B$, shifted by $z$)

• Effects
  
  ➢ If current pixel $z$ is foreground, set all pixels under $(B)_z$ to foreground.
  
  $\Rightarrow$ Expand connected components
  $\Rightarrow$ Grow features
  $\Rightarrow$ Fill holes

Image Source: R.C. Gonzales & R.E. Woods
Erosion

• Definition
  - “The erosion of $A$ by $B$ is the set of all displacements $z$, such that $(B)_z$ is entirely contained in $A$.”

• Effects
  - If not every pixel under $(B)_z$ is foreground, set the current pixel $z$ to background.
  - Erode connected components
  - Shrink features
  - Remove bridges, branches, noise
Effects

Original image

Dilation with circular structuring element

Erosion with circular structuring element

Image Source: http://homepages.inf.ed.ac.uk/rbf/HIPR2/
Effects

Original image

Dilation with circular structuring element

Erosion with circular structuring element

Image Source: http://homepages.inf.ed.ac.uk/rbf/HIPR2/
Opening

• Definition
  - Sequence of Erosion and Dilation
    \[ A \circ B = (A \ominus B) \oplus B \]

• Effect
  - \( A \circ B \) is defined by the points that are reached if \( B \) is rolled around inside \( A \).
  - \( \Rightarrow \) Remove small objects, keep original shape.

Image Source: R.C. Gonzales & R.E. Woods
Effect of Opening

- Feature selection through *size* of structuring element

Original image

Thresholded

Opening with small structuring element

Opening with larger structuring element

Effect of Opening

- Feature selection through *shape* of structuring element

*Input Image* → *Opening with circular structuring element*

- *How could we have extracted the lines?*
Closing

• Definition
  - Sequence of Dilatation and Erosion
    \[ A \cdot B = (A \oplus B) \ominus B \]

• Effect
  - \( A \cdot B \) is defined by the points that are reached if \( B \) is rolled around on the outside of \( A \).
  - \( \Rightarrow \) Fill holes, keep original shape.

Image Source: R.C. Gonzales & R.E. Woods
Effect of Closing

- Fill holes in thresholded image (e.g. due to specularities)

Size of structuring element determines which structures are selectively filled.

Image Source: http://homepages.inf.ed.ac.uk/rbf/HIPR2/
Example Application: Opening + Closing

Original image

Opening

Closing

Erosion

Dilation

Dilation

Erosion

Structuring element

Eroded image

Dilated image

Source: R.C. Gonzales & R.E. Woods

B. Leibe
Application: Blob Tracking

Absolute differences from frame to frame

Slide credit: K. Grauman  B. Leibe
Thresholding

Slide credit: K. Grauman
Morphological Boundary Extraction

• Definition
  - First erode $A$ by $B$, then subtract the result from the original $A$.
  \[ \beta(A) = A - (A \ominus B) \]

• Effects
  - If a $3 \times 3$ structuring element is used, this results in a boundary that is exactly 1 pixel thick.
Morphology Operators on Grayscale Images

- Dilation and erosion are typically performed on binary images.
- If image is grayscale: for dilation take the neighborhood max, for erosion take the min.

Slide credit: Kristen Grauman
Outline of Today’s Lecture

- Convert the image into binary form
  - Thresholding

- Clean up the thresholded image
  - Morphological operators

- Extract individual objects
  - Connected Components Labeling

- Describe the objects
  - Region properties

Image Source: D. Kim et al., Cytometry 35(1), 1999
Connected Components Labeling

- **Goal:** Identify distinct regions

Binary image

Connected components labeling

Sources: Shapiro & Stockman, Chandra
Connected Components Examples

connected components of 1’s from thresholded image

connected components of cluster labels

Source: Pinar Duygulu

B. Leibe
## Connectedness

- Which pixels are considered neighbors?

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>[i, j]</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 4-connected

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

### 8-connected

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Source: Chaitanya Chandra
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 and New object](image)

What happens in these cases?

![Equivalence table](image)
Sequential Connected Components (2)

- Process the image from left to right, top to bottom:
  1.) If the next pixel to process is 1
     
     i.) If only one of its neighbors (top or left) is 1, copy its label.
     
     ii.) If both are 1 and have the same label, copy it.
     
     iii.) If they have different labels
           - Copy the label from the left.
           - Update the equivalence table.
     
     iv.) Otherwise, assign a new label.

- Re-label with the smallest of equivalent labels

Slide credit: J. Neira

B. Leibe
Application: Segmentation of a Liver

Application by Jie Zhu, Cornell University

Slide credit: Li Shen
Outline of Today’s Lecture

- Convert the image into binary form
  - Thresholding

- Clean up the thresholded image
  - Morphological operators

- Extract individual objects
  - Connected Components Labeling

- Describe the objects
  - Region properties
Region Properties

- From the previous steps, we can obtain separated objects.

- 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, we obtain

  - **Area:**
    \[ A = \sum_{(x, y) \in R} 1 \]

  - **Centroid:**
    \[ \bar{x} = \frac{1}{A} \sum_{(x, y) \in R} x \]
    \[ \bar{y} = \frac{1}{A} \sum_{(x, y) \in R} y \]

Source: Shapiro & Stockman
Circularity

- Measure the deviation from a perfect circle

  **Circularity:**
  \[
  C = \frac{\mu_R}{\sigma_R}
  \]

  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 \((x_k, y_k)\).

- **Mean radial distance:**
  \[
  \mu_R = \frac{1}{K} \sum_{k=0}^{K-1} \left\| (x_k, y_k) - (\bar{x}, \bar{y}) \right\|
  \]

- **Variance of radial distance:**
  \[
  \sigma_R^2 = \frac{1}{K} \sum_{k=0}^{K-1} \left[ \left\| (x_k, y_k) - (\bar{x}, \bar{y}) \right\| - \mu_R \right]^2
  \]

Source: Shapiro & Stockman

B. Leibe
Invariant Descriptors

- Often, we want features independent of location, orientation, scale.

\[ [a_1, a_2, a_3, \ldots] \quad [b_1, b_2, b_3, \ldots] \]

Feature space distance
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 $S$.

- Interpretation:
  - $0^{th}$ central moment: area
  - $2^{nd}$ central moment: variance
  - $3^{rd}$ central moment: skewness
  - $4^{th}$ central moment: kurtosis
Moment Invariants ("Hu Moments")

- Normalized central moments
  \[ \eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma}, \quad \gamma = \frac{p + q}{2} + 1 \]

- From those, a set of invariant moments can be defined for object description.
  \[
  \begin{align*}
  \phi_1 &= \eta_{20} + \eta_{02} \\
  \phi_2 &= (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \\
  \phi_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
  \phi_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2
  \end{align*}
  \]

- Robust to translation, rotation & scaling, but don’t expect wonders (still summary statistics).
Moment Invariants

\[ \phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})\left[ (\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2 \right] \]
\[ + (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})\left[ 3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2 \right] \]

\[ \phi_6 = (\eta_{20} - \eta_{02})\left[ (\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2 \right] \]
\[ + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \]

\[ \phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})\left[ (\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2 \right] \]
\[ + (3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})\left[ 3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2 \right] \]

Often better to use \( \log_{10}(\phi_i) \) instead of \( \phi_i \) directly…
Axis of Least Second Moment

- Invariance to orientation?
  - Need a common alignment

- Compute Eigenvectors of 2\textsuperscript{nd} moment matrix (Matlab: eig(A))

\[
\begin{bmatrix}
\mu_{20} & \mu_{11} \\
\mu_{11} & \mu_{02}
\end{bmatrix} = VDV^T = \begin{bmatrix}
v_{11} & v_{12} \\
v_{21} & v_{22}
\end{bmatrix} \begin{bmatrix}
\lambda_1 & 0 \\
0 & \lambda_2
\end{bmatrix} \begin{bmatrix}
v_{11} & v_{12} \\
v_{21} & v_{22}
\end{bmatrix}^T
\]

Axis for which the squared distance to 2D object points is minimized (maximized).
Summary: Binary Image Processing

• Pros
  - Fast to compute, easy to store
  - Simple processing techniques
  - Can be very useful for constrained scenarios

• Cons
  - Hard to get “clean” silhouettes
  - Noise is common in realistic scenarios
  - Can be too coarse a representation
  - Cannot deal with 3D changes
References and Further Reading

• More on morphological operators can be found in

• Online tutorial and Java demos available on
Questions ?
Demo “Haribo Classification”
You Can Do It At Home...

Accessing a webcam in Matlab:

```matlab
function out = webcam
    % uses "Image Acquisition Toolbox"
    adaptorName = 'winvideo';
    vidFormat = 'I420_320x240';
    vidObj1 = videoinput(adaptorName, 1, vidFormat);
    set(vidObj1, 'ReturnedColorSpace', 'rgb');
    set(vidObj1, 'FramesPerTrigger', 1);
    out = vidObj1;

    cam = webcam();
    img = getsnapshot(cam);
```
### Matlab Intro: Everything is a matrix

```
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

N = 5  % A scalar
v = [1 0 0]  % A row vector
v = [1; 2; 3]  % A column vector
v = v'   % Transpose a vector (row to column or column to row)
```

Tutorial adapted from W. Freeman, MIT
Matlab Intro: Matrix Index

```matlab
18    %% Matrix index
19    A = magic(4);
20        >> A =
21        16   2   3   13
22        5  11  10   8
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
```

Tutorial adapted from W. Freeman, MIT
Matlab Intro: Manipulating 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)) | isnan(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
```

Tutorial adapted from W. Freeman, MIT
Matlab Intro: Manipulating 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
```

```
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
```
Matlab Intro: 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;

myotherfunction.m

function [y, z] = myotherfunction(a, b)
% Function of two arguments with two return values
y = a + b;
z = a - b;
```

Tutorial adapted from W. Freeman, MIT
Matlab Intro: Try to Code in Matrix Ways

```matlab
% use for loops
A = [1 2 3 4];
for i = 1:4
    A(i) = A(i)^2;
end
% or use matrix
A = [1 2 3 4];
A = A.^2;
```

```matlab
% use for loops
A = [1 2 3 4; 5 6 7 8];
sum(A)
    >> ans =
       6   8  10  12
ASUM = sum(A, 2)
    >> ASUM =
        10
        26
for i = 1:size(A, 1)
    for j = 1:size(A, 2)
        APROB(i, j) = A(i, j)/ASUM(i);
    end
end
% use matrix
A = [1 2 3 4; 5 6 7 8];
ASUM = sum(A, 2)
APROB = A./repmat(ASUM, 1, size(A, 2));
repmat(ASUM, 1, size(A, 2))
    >> ans =
       10   10   10   10
       26   26   26   26
```

B. Leibe
Matlab Intro: Important Commands

- **whos**  
  - List variables in workspace
- **help**  
  - Get help for any command
- **lookfor**  
  - Search for keywords
- **clear/clear x**  
  - Erase a variable/all variables
- **save**  
  - Save the workspace
- **load**  
  - Load a saved workspace
- **keyboard**  
  - Enter debugging mode (until `dbquit`)
Morphology in Matlab

- \( N = \text{hist}(Y,M) \)
- \( \text{IM2} = \text{imerode}(	ext{IM}, \text{SE}); \)
- \( \text{IM2} = \text{imdilate}(	ext{IM}, \text{SE}); \)
- \( \text{IM2} = \text{imclose}(	ext{IM}, \text{SE}); \)
- \( \text{IM2} = \text{imopen}(	ext{IM}, \text{SE}); \)
- \( L = \text{bwlabel}(	ext{BW}, \text{N}); \)
- \( \text{STATS} = \text{regionprops}(L, \text{PROPERTIES}); \)
  - 'Area'
  - 'Centroid'
  - 'BoundingBox'
  - 'Orientation', ...

Slide credit: Kristen Grauman
Questions?