Binary Images

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

Uses: Industrial Inspection


Uses: Document Analysis, Text Recognition

- Handwritten digits
- Natural text (after detection)

Uses: Medical/Bio Data

- Source: D. Kim et al., Cytometry 35(1), 1999
Perceptual and Sensory Augmented Computing

**Uses: Blob Tracking & Motion Analysis**

- **Frame Differencing**
  - Image Source: Kristen Grauman

- **Background Subtraction**
  - Image Source: Tobias Jäggi

**Uses: Shape Analysis, Free-Viewpoint Video**

- **Silhouette**
  - Visual Hull Reconstruction
  - Image Source: Blue-c project, ETH Zurich

**Uses: Intensity Based Detection**

- **Looking for dark pixels...**

  \[
  \text{fg_pix} = \text{find}(\text{im} < 65);
  \]

  Slide Credit: Kristen Grauman

**Uses: Color Based Detection**

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

  \[
  \text{fg_pix} = \text{find}(\text{hue} > t1 \& \text{hue} < t2);
  \]

  Slide Credit: Kristen Grauman

**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
    \[ F_{\text{\text{1-sided}}}^i[j] = \begin{cases} 1, & \text{if } F_{\text{\text{i}}}^i[j] \geq T \\ 0, & \text{otherwise} \end{cases} \]
  - Two-sided
    \[ F_{\text{\text{2-sided}}}^i[j] = \begin{cases} 1, & \text{if } T_1 \leq F_{\text{\text{i}}}^i[j] \leq T_2 \\ 0, & \text{otherwise} \end{cases} \]
  - Set membership
    \[ F_{\text{\text{set}}}^i[j] = \begin{cases} 1, & \text{if } F_{\text{\text{i}}}^i[j] \in Z \\ 0, & \text{otherwise} \end{cases} \]

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 then 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

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

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
1. Precompute a cumulative grayvalue histogram \(h\).
2. For each potential threshold \(T\)
   a) Separate the pixels into two clusters according to \(T\)
   b) Look up \(n_1, n_2\) in \(h\) and compute both cluster means
   c) Compute \(\sigma_{\text{between}}^2(T) = n_1(T)n_2(T) [\mu_1(T) - \mu_2(T)]^2\)
3. Choose \(T^* = \arg\max_T \sigma_{\text{between}}^2(T)\)
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?

Additional Improvements

- Document images often contain a smooth gradient
  
  \( \Rightarrow \) Try to fit that gradient with a polynomial function

Polynomial Surface Fitting

- Polynomial surface of degree $d$
  \[ f(x, y) = \sum_{i+j=d} b_{ij} x^i y^j \]
- For an image pixel \((x_0, y_0)\) with intensity $I_0$, this means
  \[ b_{0,0} + b_{1,0} x_0 + b_{0,1} y_0 + b_{2,0} x_0^2 + b_{1,1} x_0 y_0 + \cdots + b_{0,3} y_0^3 = I_0 \]
- Least-squares estimation, e.g. for $d = 3$

\[
\begin{bmatrix}
1 & x_0 & y_0 & x_0^2 & x_0 y_0 & y_0^2 \\
1 & x_1 & y_1 & x_1^2 & x_1 y_1 & y_1^2 \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
1 & x_n & y_n & x_n^2 & x_n y_n & y_n^2 \\
\end{bmatrix}
= 
\begin{bmatrix}
b_{0,0} \\
b_{1,0} \\
b_{0,1} \\
b_{2,0} \\
\end{bmatrix}
\]

\[
A b = I
\]

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.

Result Comparison
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

Cleaning the Binarized Results

• Results of thresholding often still contain noise

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

Dilation

• Definition
  - “The dilation of \( A \) by \( B \) is the set of all displacements \( z \), such that \((\hat{B})\), and \( A \) overlap by at least one element.”
• Effects
  - If current pixel \( z \) is foreground, set all pixels under \((\hat{B})\) to foreground.
  - Grow features
  - Fill holes

Erosion

• Definition
  - “The erosion of \( A \) by \( B \) is the set of all displacements \( z \), such that \((\hat{B})\), is entirely contained in \( A \).”
• Effects
  - If not every pixel under \((\hat{B})\) is foreground, set the current pixel \( z \) to background.
  - Erode connected components
  - Shrink features
  - Remove bridges, branches, noise

Effects

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

⇒ Morphological Operators
**Effects**

- **Dilation with circular structuring element**
- **Erosion with circular structuring element**

**Opening**

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

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

**Effect of Opening**

- Feature selection through **size** of structuring element
- Original image
- Thresholded
- Opening with small structuring element
- Opening with larger structuring element

**Closing**

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

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

**Effect of Closing**

- Fill holes in thresholded image (e.g. due to specularities)
- Original image
- Thresholded
- Closing with circular structuring element

Size of structuring element determines which structures are selectively filled.
Example Application: Opening + Closing

Original image

Structuring element

Eroded image

Dilated image

Closing

B. Leibe

Application: Blob Tracking

Absolute differences from frame to frame

Eroding

Thresholding

Dilated image

Eroded image

B. Leibe

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.
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

### Connected Components Labeling

- Goal: Identify distinct regions

### Connected Components Examples

- Connected components of 1’s from thresholded image
- Connected components of cluster labels

### Connectedness

- Which pixels are considered neighbors?

### 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).

### 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.
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.

Equivalence table

[Red, blue, green, yellow]

Application: Segmentation of a Liver

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

Outline of Today’s Lecture

Image Source: D. Kim et al., Cytometry 35(1), 1999
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: \( \mu_x = \frac{1}{A} \sum_{(x,y) \in R} x \), \( \mu_y = \frac{1}{A} \sum_{(x,y) \in R} y \)

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: \( \mu_r = \frac{1}{K} \sum_{k=0}^{K-1} \| (x_k, y_k) - (\mu_x, \mu_y) \| \)
- Variance of radial distance: \( \sigma_r^2 = \frac{1}{K} \sum_{k=0}^{K-1} \| (x_k, y_k) - (\mu_x, \mu_y) \| - \mu_r \|^2 \)

Invariant Descriptors

- Often, we want features independent of location, 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 \( S \).
- Interpretation:
  - 0th central moment: area
  - 2nd central moment: variance
  - 3rd central moment: skewness
  - 4th central moment: kurtosis

Moment Invariants ("Hu Moments")

- Normalized central moments
  \[ \eta_{pq} = \frac{\mu_{pq}}{\mu_{00}} \]
  \[ \gamma = \frac{p+q}{2} + 1 \]
- From those, a set of invariant moments can be defined for object description.
  \[ \phi_1 = \eta_{20} + \eta_{02} \]
  \[ \phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2 \]
  \[ \phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \]
  \[ \phi_4 = (\eta_{40} + \eta_{22})^2 + (\eta_{22} + \eta_{04})^2 \]
- Robust to translation, rotation & scaling, but don’t expect wonders (still summary statistics).
Moment Invariants

\[ \phi_i = (\eta_{i0} - 3\eta_{i2})(\eta_{i0} + \eta_{i2})^2(\eta_{i0} - \eta_{i0})^2 \]
\[ + (3\eta_{i2} - \eta_{i0})(\eta_{i2} + \eta_{i0})(3(\eta_{i0} + \eta_{i2})^2 - (\eta_{i2} + \eta_{i0})^2) \]
\[ \phi_1 = (\eta_{i0} - \eta_{i2})(\eta_{i0} + \eta_{i2})^2 - (\eta_{i2} + \eta_{i0})^2 \]
\[ + 4\eta_{i1}(\eta_{i0} + \eta_{i2})(\eta_{i2} + \eta_{i0}) \]
\[ \phi_2 = (3\eta_{i2} - \eta_{i0})(\eta_{i0} + \eta_{i2})(\eta_{i0} - \eta_{i0})^2 - 3(\eta_{i2} - \eta_{i0})^2 \]
\[ + (3\eta_{i0} - \eta_{i2})(\eta_{i2} + \eta_{i0})(3(\eta_{i0} + \eta_{i2})^2 - (\eta_{i2} + \eta_{i0})^2) \]

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 2nd moment matrix (Matlab: \( \text{eig}(A) \))

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

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?
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);
```