Outline

• Review
  ▫ Morphology
  ▫ Connected Components
• Filtering for Detection
• Pyramids
• Wavelets
Morphological Image Processing

- Filtering done on binary images
  - Images with two values [0,1], [0, 255], [black, white]
  - Typically, this image will be obtained by thresholding
    \[ g(x, y) = \begin{cases} 
    1 & f(x, y) > T \\
    0 & f(x, y) \leq T 
    \end{cases} \]

- Morphology is concerned with the structure and shape

- In morphology, a binary image is convolved with a structuring element $s$ and results in a binary image

- See Chapter 9 of Gonzalez and Woods for a more complete treatment

- Matlab
Mathematical Morphology

- Tool for extracting image components that are useful in the representation and description of region shape
  - Boundaries, skeletons, convex hull, etc.
- The language of mathematical morphology is set theory
  - A set represents an object in an image
- This is often useful in video processing because of the simplicity of processing and emphasis on “objects”
  - Handy tool for “clean up” of a thresholded image
Morphological Operations

- **Threshold operation**
  \[ \theta(f, t) = \begin{cases} 
  1 & f \geq t \\
  0 & \text{else} 
\end{cases} \]

- **Structuring element**
  - \( s \) – e.g. 3 x 3 box filter (1’s indicate included pixels in the mask)
  - \( S \) – number of “on” pixels in \( s \)

- **Count of 1s in a structuring element**
  - \( c = f \otimes s \)
  - Correlation (filter) raster scan procedure

- **Basic morphological operations can be extended to grayscale images**

- **Dilation**
  - \[ \text{dilate}(f, s) = \theta(c, 1) \]
  - Grows (thickens) 1 locations

- **Erosion**
  - \[ \text{erode}(f, s) = \theta(c, S) \]
  - Shrinks (thins) 1 locations

- **Opening**
  - \[ \text{open}(f, s) = \text{dilate}(\text{erode}(f, s), s) \]
  - Generally smooth the contour of an object, breaks narrow isthmuses, and eliminates thin protrusions

- **Closing**
  - \[ \text{close}(f, s) = \text{erode}(\text{dilate}(f, s), s) \]
  - Generally smooth the contour of an object, fuses narrow breaks/separations, eliminates small holes, and fills gaps in a contour
Morphology Example

- **Dilation** - grows (thickens) 1 locations
- **Erosion** - shrink (thins) 1 locations
- **Opening** - generally smooth the contour of an object, breaks narrow isthmuses, and eliminates thin protrusions
- **Closing** - generally smooth the contour of an object, fuses narrow breaks/separations, eliminates small holes, and fills gaps in a contour

Note: Black is “1” location

Figure 3.21  Binary image morphology: (a) original image; (b) dilation; (c) erosion; (d) majority; (e) opening; (f) closing. The structuring element for all examples is a $5 \times 5$ square. The effects of majority are a subtle rounding of sharp corners. Opening fails to eliminate the dot, since it is not wide enough.
Binary Image Logic Operations

- Extension of basic logic operators
  - NOT, AND, OR, XOR

- Often use for “masking”
Structuring Elements

- The structuring element (SE) can be any shape
  - This is a mask of “on” pixels within a rectangular container
  - Typically, the SE is symmetric

![Diagram of structuring elements](image)
Erosion

- Retain only pixels where the entire SE is overlapped
  - \( A \ominus B = \{ z | (B)_z \subseteq A \} \)
  - \( A \) – image
  - \( B \) – SE
  - \( z \) – displacements (x,y locations)

\[
A \ominus B = \{ z | (B)_z \subseteq A \}
\]

\( A \ominus B \)

\[
A - image \quad B - SE \quad z - displacements (x,y locations)
\]

\( A \ominus B \)

\( A \ominus B \)

\( A \ominus B \)

\( A \ominus B \)

\( A \ominus B \)

\( A \ominus B \)

\( \)
Dilation

- Output image is “on” anywhere the SE touches an “on” pixel
  - $A \oplus B = \{z | (B)_z \cap A \neq \emptyset\}$
    - $A$ – image
    - $B$ – SE
    - $z$ – displacements ($x,y$ locations)
Other Morphological Operations

• Boundary extraction
  ▫ \( \beta(A) = A - (A \ominus B) \)
  ▫ Subtract erosion from original
  ▫ Notice this is an edge extraction

• Convex hull (\( H \))
  ▫ Smallest convex set that contains another set \( S \)
  ▫ This is often done for a collection of 2D or 3D points
  ▫ \texttt{bwconvhull.m}
Connected Components

• Semi-global image operation to provide consistent labels to similar regions
  ▫ Based on adjacency concept
• Most efficient algorithms compute in two passes

\[ X_k = (X_{k-1} \oplus B) \cap A \]

![Diagram](image)

Figure 3.23  Connected component computation: (a) original grayscale image; (b) horizontal runs (nodes) connected by vertical (graph) edges (dashed blue)—runs are pseudocolored with unique colors inherited from parent nodes; (c) re-coloring after merging adjacent segments.

• More computational formulations (iterative) exist from morphology
More Connected Components

- Typically, only the “white” pixels will be considered objects
  - Dark pixels are background and do not get counted
- After labeling connected components, statistics from each region can be computed
  - Statistics describe the region – e.g. area, centroid, perimeter, etc.

- Matlab functions
  - `bwconncomp.m`, `labelmatrix.m` (or `bwlabel.m`) - label image
  - `label2rgb.m` – color components for viewing
  - `regionprops.m` – calculate region statistics
Connected Component Example

Grayscale image

Threshold image

Opened Image

Labeled image – 91 grains of rice
Binary Filtering as Detection

• Filtering (correlation) can be used as a simple object detector
  ▫ Mask provides a search template
  ▫ “Matched filter” – kernels look like the effects they are intended to find

This is who I am. Nobody said you had to like it.
Correlation Masking

This is who I am. Nobody said you had to like it.

correlation

detected letter

0.9 max threshold

0.5 max threshold
Normalized Cross-Correlation

- Extension to intensity values
  - Handle variation in template and image brightness

Adapted from http://kurser.iha.dk/ee-ict-master/ticovi/
Where’s Waldo

Detected template

correlation map

Adapted from
http://kurser.iha.dk/ee-ict-master/ticovi/
Detection of Similar Objects

- Previous examples are detecting exactly what we want to find
  - Give the perfect template
- What happens with similar objects

- Works fine when scale, orientation, and general orientation are matched
- What to do with different sized objects, new scenes

Adapted from K. Grauman
Pyramids

• Image processing so far has input/output images of the same size
  ▫ Will want to change the size of an image
    • Interpolation to make small image larger
    • Decimation to operate on a smaller image

• Sometimes we may not know the appropriate resolution
  ▫ What size cars are we looking for in the previous example?
  ▫ Create a “pyramid” of images at different resolutions to do processing
    • Accelerates search process
    • Multi-scale representation and processing
Decimation

- Also known as downsampling
  - Decreases the size of an image by removing pixels
- Easiest form of decimation is subsampling by a factor of 2
  - Throw away pixels

- What can we say about the quality of this?
Subsampling without pre-filtering

Adapted from S. Seitz
Subsampling with Gaussian pre-filtering

- Improved results by first smoothing

Adapted from S. Seitz
Smooth to Avoid Aliasing

- Low-pass smoothing filter avoids aliasing
- In practice, the convolution can be evaluated at a reduced rate
  \[ g(i, j) = \sum_{k,l} f(k, l) h(ri - k, rj - l) \]
  - \( r \) – the downsampling rate

Figure 3.30  Signal decimation: (a) the original samples are (b) convolved with a low-pass filter before being downsampled.
Downsample Filters

- A number of filters can be used and have various performance
  - Amount of high-frequency removal
- Bilinear filter is the simplest
- Bicubic fits a 3rd degree polynomial to the neighborhood

<table>
<thead>
<tr>
<th>( n )</th>
<th>Linear</th>
<th>Binomial</th>
<th>Cubic ( a = -1 )</th>
<th>Cubic ( a = -0.5 )</th>
<th>Windowed sinc</th>
<th>QMF-9</th>
<th>JPEG 2000</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0.50</td>
<td>0.3750</td>
<td>0.5000</td>
<td>0.5000</td>
<td>0.4939</td>
<td>0.5638</td>
<td>0.6029</td>
</tr>
<tr>
<td>1</td>
<td>0.25</td>
<td>0.2500</td>
<td>0.3125</td>
<td>0.28125</td>
<td>0.2684</td>
<td>0.2932</td>
<td>0.2669</td>
</tr>
<tr>
<td>2</td>
<td>0.0625</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>0.0000</td>
<td>-0.0519</td>
<td>-0.0782</td>
</tr>
<tr>
<td>3</td>
<td>-0.0625</td>
<td>-0.03125</td>
<td></td>
<td>-0.0153</td>
<td>-0.0431</td>
<td>-0.0169</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
<td></td>
<td>0.0000</td>
<td>0.0198</td>
<td>0.0267</td>
<td></td>
</tr>
</tbody>
</table>

Table 3.4 Filter coefficients for 2x decimation. These filters are of odd length, are symmetric, and are normalized to have unit DC gain (sum up to 1). See Figure 3.31 for their associated frequency responses.

Figure 3.31 Frequency response for some 2x decimation filters. The cubic \( a = -1 \) filter has the sharpest fall-off but also a bit of ringing; the wavelet analysis filters (QMF-9 and JPEG 2000), while useful for compression, have more aliasing.
Interpolation

- Also known as upsampling
  - Increases the size of an image by inserting new pixels between existing ones
- This can be done with a modified convolution operation
  - \( g(i, j) = \sum_{k,l} f(k, l)h(i - rk, j - rl) \)
    - \( r \) – the upsampling rate
- Each new pixel is a weighted sum of samples
Interpolation Kernels

Notice some ringing with bilinear

Figure 3.28 Two-dimensional image interpolation: (a) bilinear; (b) bicubic \( (a = -1) \); (c) bicubic \( (a = -0.5) \); (d) windowed sinc (nine taps).
Multi-Resolution Pyramids

- Accelerates coarse-to-fine search algorithms
  - Faster search at lower resolution and higher resolution for better localization
- Detect objects at different scales
  - Use same template with different sized images = having different sized templates
- Perform multi-resolution blending operations
Laplacian (Gaussian) Pyramid

- [Burt and Adelson, 1983]
- Blur and subsample the original image by a factor of 2 at each level
Gaussian Pyramid

- Construction

- Repeat:
  - Filter
  - Subsample
- Until minimum resolution is reached
- The whole pyramid is only $4/3$ the size of the original image
  - Each higher level is $1/4$ the size of lower level

Adapted from S. Seitz