Final Review

Image Processing
CSE 166
Lecture 18
Topics covered

• Basis vectors
• Matrix-based transforms
• Basis images
• Wavelet transform
• Image compression
• Image watermarking
• Morphological image processing
• Edge detection
• Segmentation
Matrix-based transforms

\[ T(u) = \sum_{x=0}^{N-1} f(x)r(x,u) \quad \text{Forward transform} \]

\[ f(x) = \sum_{u=0}^{N-1} T(u)s(x,u) \quad \text{Inverse transform} \]

where

- \( x \) is a spatial variable
- \( u \) is a transform variable
- \( T(u) \) is the transform of \( f(x) \)
- \( f(x) \) is the inverse transform of \( T(u) \)
- \( r(x,u) \) is a forward transformation kernel
- \( s(x,u) \) is an inverse transformation kernel
General inverse transform using basis vectors

\[ f(x) = T(0)s(x, 0) + T(1)s(x, 1) + \ldots + T(N - 1)s(x, N - 1) \]
Matrix-based transforms using orthonormal basis vectors

• In vector form

\[ T(u) = \langle s(x, u), f(x) \rangle \]
\[ T(u) = \langle s_u, f \rangle \]

where \( s_u = \begin{bmatrix} s(0, u) \\ s(1, u) \\ \vdots \\ s(N - 1, u) \end{bmatrix} \) and \( f = \begin{bmatrix} f(0) \\ f(1) \\ \vdots \\ f(N - 1) \end{bmatrix} \)

\[ T(u) = s_u^H f \text{ for complex vectors} \]
\[ T(u) = s_u^T f \text{ for real vectors} \]
Matrix-based transforms using orthonormal basis vectors

- In matrix form

\[
\begin{bmatrix}
T(0) \\
T(1) \\
\vdots \\
T(N-1)
\end{bmatrix} =
\begin{bmatrix}
s_0^H \\
s_1^H \\
\vdots \\
s_{N-1}^H
\end{bmatrix}
\begin{bmatrix}
f(0) \\
f(1) \\
\vdots \\
f(N-1)
\end{bmatrix}
\]

for complex vectors

\[
\begin{bmatrix}
T(0) \\
T(1) \\
\vdots \\
T(N-1)
\end{bmatrix} =
\begin{bmatrix}
s_0^T \\
s_1^T \\
\vdots \\
s_{N-1}^T
\end{bmatrix}
\begin{bmatrix}
f(0) \\
f(1) \\
\vdots \\
f(N-1)
\end{bmatrix}
\]

for real vectors

\[
t = Af \\
f = A^Ht \text{ for complex vectors} \\
f = A^Tt \text{ for real vectors}
\]
Matrix-based transforms using biorthonormal basis vectors

• In matrix form

\[
\begin{bmatrix}
T(0) \\
T(1) \\
\vdots \\
T(N-1)
\end{bmatrix}
= \begin{bmatrix}
\tilde{s}_0^H \\
\tilde{s}_1^H \\
\vdots \\
\tilde{s}_{N-1}^H
\end{bmatrix}
\begin{bmatrix}
f(0) \\
f(1) \\
\vdots \\
f(N-1)
\end{bmatrix}
\text{for complex vectors}
\]

\[
\begin{bmatrix}
T(0) \\
T(1) \\
\vdots \\
T(N-1)
\end{bmatrix}
= \begin{bmatrix}
\tilde{s}_0^T \\
\tilde{s}_1^T \\
\vdots \\
\tilde{s}_{N-1}^T
\end{bmatrix}
\begin{bmatrix}
f(0) \\
f(1) \\
\vdots \\
f(N-1)
\end{bmatrix}
\text{for real vectors}
\]

\[t = \tilde{A} f\]

\[f = A^H t \text{ for complex vectors}\]
\[f = A^T t \text{ for real vectors}\]
Matrix-based transform

Example: 8-point DFT of \( f(x) = \sin(2\pi x) \)

\[ s_u(x) \leftrightarrow s(x,u) \text{ or } A = [s_0 \ s_1 \ldots s_7]^T \]

real part + imaginary part
Matrix-based transforms

• Discrete Fourier transform (DFT)
• Discrete Hartley transform (DHT)
• Discrete cosine transform (DCT)
• Discrete sine transform (DST)
• Walsh-Hadamard (WHT)
• Slant (SLT)
• Haar (HAAR)
• Daubechies (DB4)
• Biorthogonal B-spline (BIOR3.1)
Basis vectors of matrix-based 1D transforms

$N = 16$

real part  imaginary part  Standard basis (for reference)
Basis vectors of matrix-based 1D transforms

N = 16
Matrix-based transforms in two dimensions

\[ T(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y)r(x, y, u, v) \quad \text{Forward transform} \]

\[ f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u, v)s(x, y, u, v) \quad \text{Inverse transform} \]

where

- \( x, y \) are spatial variables
- \( u, v \) are transform variables
- \( T(u, v) \) is the transform of \( f(x, y) \)
- \( f(x, y) \) is the inverse transform of \( T(u, v) \)
- \( r(x, y, u, v) \) is a forward transformation kernel
- \( s(x, y, u, v) \) is an inverse transformation kernel
Matrix-based transforms in two dimensions

- If \( r \) and \( s \) are separable and symmetric, and \( M = N \), then
  - For orthonormal basis vectors
    \[
    T = AFA^T
    \]
    \[
    F = A^*TA^* \text{ for complex vectors}
    \]
    \[
    F = A^T TA \text{ for real vectors}
    \]
  - For biorthonormal basis vectors
    \[
    T = \tilde{A}F\tilde{A}^T
    \]
    \[
    F = A^*T\tilde{A}A^* \text{ for complex vectors}
    \]
    \[
    F = A^T\tilde{A}TA \text{ for real vectors}
    \]
Matrix-based transforms in two dimensions using basis images

- Inverse transform

\[
f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u, v)s(x, y, u, v)
\]

\[
F = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u, v)S_{u,v}
\]

where

\[
F = \begin{bmatrix}
    f(0, 0) & f(0, 1) & \cdots & f(0, N - 1) \\
    f(1, 0) & f(1, 1) & \cdots & f(1, N - 1) \\
    \vdots & \vdots & \ddots & \vdots \\
    f(N - 1, 0) & f(N - 1, 1) & \cdots & f(N - 1, N - 1)
\end{bmatrix}
\]

\[
S_{u,v} = \begin{bmatrix}
    s(0, 0, u, v) & s(0, 1, u, v) & \cdots & s(0, N - 1, u, v) \\
    s(1, 0, u, v) & s(1, 1, u, v) & \cdots & s(1, N - 1, u, v) \\
    \vdots & \vdots & \ddots & \vdots \\
    s(N - 1, 0, u, v) & s(N - 1, 1, u, v) & \cdots & s(N - 1, N - 1, u, v)
\end{bmatrix}
\]

Each \(S_{u,v}\) is a basis image
Basis images of matrix-based 2D transforms

Standard basis images (for reference)

8-by-8 array of 8-by-8 basis images
Basis images of matrix-based 2D transforms

Discrete Fourier transform (DFT) basis images

real part  imaginary part
Basis images of matrix-based 2D transforms

Discrete Hartley transform (DHT) basis images
Basis images of matrix-based 2D transforms

Discrete cosine transform (DCT) basis images
Basis images of matrix-based 2D transforms

Discrete sine transform (DST) basis images
Basis images of matrix-based 2D transforms

Walsh-Hadamard transform (WHT) basis images
Basis images of matrix-based 2D transforms

Slant transform (SLT) basis images
Basis images of matrix-based 2D transforms

Haar transform (HAAR) basis images
Wavelet transforms

• A scaling function is used to create a series of approximations of a function or image, each differing by a factor of 2 in resolution from its nearest neighboring approximations.
• Wavelet functions (wavelets) are then used to encode the differences between adjacent approximations.
• The discrete wavelet transform (DWT) uses those wavelets, together with a single scaling function, to represent a function or image as a linear combination of the wavelets and scaling function.
Scaling functions 
and set of basis vectors

• Father scaling function

\[ \phi(x) \]

• Set of basis functions

\[ \{ \phi_{j,k}(x) \}, \text{ where } \phi_{j,k}(x) = 2^{j/2} \phi(2^j x - k) \quad \forall \ j, k \in \mathbb{Z} \]

– Integer translation \( k \)
– Binary scaling \( j \)

• Basis of the function space spanned by

\[ \phi_{j,k}(x) \text{ for } j = j_0 \text{ and } k \in \mathbb{Z} \]

\[ V_{j_0} = \{ \phi_{j_0,k} \} \quad \forall \ k \in \mathbb{Z} \]
Scaling function, multiresolution analysis

1. The scaling function is orthogonal to its integer translates.

2. The function spaces spanned by the scaling function at low scales are nested within those spanned at higher scales.
   \[ V_{-\infty} \subset \cdots \subset V_{-1} \subset V_0 \subset V_1 \subset \cdots \subset V_\infty \]

3. The only function representable at every scale (all \( V_j \)) is \( f(x) = 0 \).

4. All measurable, square-integrable functions can be represented as \( j \to \infty \).
Wavelet functions

• Given father scaling function $\phi(x)$, there exists a mother wavelet function $\psi(x)$ whose integer translations and binary scalings

$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k) \quad \forall \ j, k \in \mathbb{Z}$$

span the difference between any two adjacent scaling spaces

• The orthogonal compliment of $V_{j_0}$ in $V_{j_0+1}$ is $W_{j_0}$

$$V_{j_0+1} = V_{j_0} \oplus W_{j_0} \quad \text{for } j = j_0 \text{ and } k \in \mathbb{Z}$$

$$\langle \phi_{j_0,k}(x), \psi_{j_0,l}(x) \rangle = 0 \text{ for } k \neq l$$
Relationship between scaling and wavelet function spaces

\[ V_2 = V_1 \oplus W_1 = V_0 \oplus W_0 \oplus W_1 \]

\[ V_1 = V_0 \oplus W_0 \]

\[ W_j \text{ is orthogonal complement of } V_j \text{ in } V_{j+1} \]

\[ V \text{ is basis of the function space spanned by scaling function} \]

Union
Scaling function coefficients and wavelet function coefficients

• Refinement (or dilation) equation

\[ \phi(x) = \sum_{k \in \mathbb{Z}} h_\phi(k) \sqrt{2} \phi(2x - k) \]

where \( h_\phi(k) \) are scaling function coefficients

• And

\[ \psi(x) = \sum_{k} h_\psi(k) \sqrt{2} \psi(2x - k) \]

where \( h_\psi(k) \) are wavelet function coefficients

• Relationship

\[ h_\psi(k) = (-1)^k h_\phi(1 - k) \]
1D discrete wavelet transform

for real signals

• Forward

\[ T_\phi(j_0, k) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f(x) \phi_{j_0,k}(x) \]

Approximation

\[ T_\psi(j, k) = \frac{1}{\sqrt{N}} \sum_{x=0}^{N-1} f(x) \psi_{j,k}(x) \]

Details

• Inverse

\[ f(x) = \frac{1}{\sqrt{N}} \sum_{k=0}^{2^{j_0} - 1} T_\phi(j_0, k) \phi_{j_0,k}(x) + \frac{1}{\sqrt{N}} \sum_{j=j_0}^{J-1} \sum_{k=0}^{2^j - 1} T_\psi(j, k) \psi_{j,k}(x) \]
2D discrete wavelet transform
for real signals

• Forward

\[ T_\phi(j_0, m, n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \phi_{j_0, m, n}(x, y) \]

\[ T_\psi^i(j, m, n) = \frac{1}{\sqrt{MN}} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \psi^{i}_{j, m, n}(x, y), \quad i = \{H, V, D\} \]

where

\[ \phi_{j, m, n}(x, y) = 2^{j/2} \phi(2^j x - m, 2^j y - n) \]

\[ \psi^{i}_{j, m, n}(x, y) = 2^{j/2} \psi^{i}(2^j x - m, 2^j y - n), \quad i = \{H, V, D\} \]

\[ \psi^H(x, y) = \psi(x)\phi(y) \]

\[ \psi^V(x, y) = \phi(x)\psi(y) \]

\[ \psi^D(x, y) = \psi(x)\psi(y) \]

Directional wavelets
2D discrete wavelet transform
for real signals

• Inverse

\[
f(x, y) = \frac{1}{\sqrt{MN}} \sum_{m=0}^{2^{j_0}-1} \sum_{n=0}^{2^{j_0}-1} T_\phi(j_0, m, n) \phi_{j_0,m,n}(x, y)
\]

\[+ \frac{1}{\sqrt{MN}} \sum_{i=H,V,D} \sum_{j=j_0}^{J-1} \sum_{m=0}^{2^j-1} \sum_{n=0}^{2^j-1} T_\psi^i(j, m, n) \psi_{j,m,n}(x, y)\]
2D discrete wavelet transform

Decomposition

\[ T_\varphi(j+1, k, l) \]

\[ T_\varphi(j, k, l) \quad T_\psi^H(j, k, l) \]

\[ T_\psi^V(j, k, l) \quad T_\psi^D(j, k, l) \]

Approximation

Horizontal details

Vertical details

Diagonal details
2D discrete wavelet transform

3-level wavelet decomposition
Wavelets in image processing

1. Wavelet transform
2. Alter transform
3. Inverse wavelet transform
Wavelet-based edge detection

- Zero horizontal details
- Zero lowest scale approximation

Edges

Vertical edges
Wavelet-based noise removal

Noisy image

Zero highest resolution details

Zero details for all levels

Threshold details

FIGURE 7.28
Modifying a DWT for noise removal: (a) a noisy CT of a human head; (b), (c) and (e) various reconstructions after thresholding the detail coefficients; (d) and (f) the information removed during the reconstruction of (c) and (e).
(Original image courtesy Vanderbilt University Medical Center.)
Data compression

• Data redundancy

\[ R = 1 - \frac{1}{C} \]

where compression ratio

\[ C = \frac{b}{b'} \]

where

\[ b \text{ and } b' \text{ are the number of bits in two different representations of the same information} \]
Data redundancy in images

Coding redundancy

Spatial redundancy

Irrelevant information

Does not need all 8 bits

Information is unnecessarily replicated

Information is not useful
Image information

- Entropy

\[ \tilde{H} = - \sum_{k=0}^{L-1} p_r(r_k) \log_2(p_r(r_k)) \]

where

- \( L \) is the number of intensity or gray levels
- \( r_k \) is input image intensity or gray level value \( k \)
- \( p_r(r_k) \) is normalized histogram of input image

- It is not possible to encode input image with fewer than \( \tilde{H} \) bits/pixel
Fidelity criteria, objective (quantitative)

• Total error
\[ \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left( \hat{f}(x, y) - f(x, y) \right) \]

• Root-mean-square error
\[ e_{\text{rms}} = \left( \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left( \hat{f}(x, y) - f(x, y) \right)^2 \right)^{1/2} \]

• Mean-square signal to noise ratio (SNR)
\[ \text{SNR}_{\text{rms}} = \frac{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \hat{f}(x, y)^2}{\sum_{x=0}^{M-1} \sum_{y=0}^{N-1} \left( \hat{f}(x, y) - f(x, y) \right)^2} \]
Fidelity criteria subjective (qualitative)

<table>
<thead>
<tr>
<th>Value</th>
<th>Rating</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Excellent</td>
<td>An image of extremely high quality, as good as you could desire.</td>
</tr>
<tr>
<td>2</td>
<td>Fine</td>
<td>An image of high quality, providing enjoyable viewing. Interference is not objectionable.</td>
</tr>
<tr>
<td>3</td>
<td>Passable</td>
<td>An image of acceptable quality. Interference is not objectionable.</td>
</tr>
<tr>
<td>4</td>
<td>Marginal</td>
<td>An image of poor quality; you wish you could improve it. Interference is somewhat objectionable.</td>
</tr>
<tr>
<td>5</td>
<td>Inferior</td>
<td>A very poor image, but you could watch it. Objectionable interference is definitely present.</td>
</tr>
<tr>
<td>6</td>
<td>Unusable</td>
<td>An image so bad that you could not watch it.</td>
</tr>
</tbody>
</table>
Approximations

Objective (quantitative) quality
rms error (in intensity levels)

5.17  15.67  14.17

Subjective (qualitative) quality, relative

(a) is better than (b).
(b) is better than (c)
Compression system

Encoder

Mapper → Quantizer → Symbol coder → Compressed data for storage and transmission

Decoder

Symbol decoder → Inverse mapper → \( \hat{f}(x, y) \) or \( \hat{f}(x, y, t) \)
Compression methods

- Huffman coding
- Golomb coding
- Arithmetic coding
- Lempel-Ziv-Welch (LZW) coding
- Run-length coding
- Symbol-based coding
- Bit-plane coding
- Block transform coding
- Predictive coding
- Wavelet coding
Block-transform coding

**Encoder**

Input image \((M \times N)\) -> Contract \(n \times n\) subimage -> Forward transform -> Quantizer -> Symbol encoder -> Compressed image

**Decoder**

Compressed image -> Symbol decoder -> Inverse transform -> Merge \(n \times n\) subimage -> Decompressed image
Block-transform coding

- Example: discrete cosine transform

\[ T(u, v) = \sum_{x=0}^{n-1} \sum_{y=0}^{n-1} g(x, y) \alpha(u) \alpha(v) \cos \left( \frac{(2x + 1)u\pi}{2n} \right) \cos \left( \frac{(2y + 1)v\pi}{2n} \right) \]

where

\[ \alpha(u) = \begin{cases} \sqrt{\frac{1}{n}} & \text{if } u = 0 \\ \sqrt{2/n} & \text{otherwise} \end{cases} \quad \text{and} \quad \alpha(v) = \begin{cases} \sqrt{1/n} & \text{if } v = 0 \\ \sqrt{2/n} & \text{otherwise} \end{cases} \]

- Inverse

\[ g(x, y) = \sum_{u=0}^{n-1} \sum_{v=0}^{n-1} T(u, v) \alpha(u) \alpha(v) \cos \left( \frac{(2x + 1)u\pi}{2n} \right) \cos \left( \frac{(2y + 1)v\pi}{2n} \right) \]
Block-transform coding

4x4 subimages (4x4 basis images)

Walsh-Hadamard transform

Discrete cosine transform
Block-transform coding

8x8 subimages

Fourier transform
- Retain 32 largest coefficients

Walsh-Hadamard transform

Cosine transform
- Error image
- rms error
  - 2.32
  - 1.78
  - 1.13

Lower is better
Block-transform coding

Reconstruction error versus subimage size

DCT subimage size: 2x2  4x4  8x8
JPEG uses block DCT-based coding

Compression reconstruction  Scaled error image  Zoomed compression reconstruction

25:1

Compression ratio

52:1
Predictive coding model

**Encoder**

**Decoder**
Predictive coding

Example: previous pixel coding

Input image

Prediction error image

Histograms

Std. dev. = 45.60
Entropy = 7.25

Std. dev. = 15.58
Entropy = 3.99
Wavelet coding

Encoder

Input image → Wavelet transform → Quantizer → Symbol encoder → Compressed image

Decoder

Compressed image → Symbol decoder → Inverse wavelet transform → Decompressed image
Wavelet coding

Detail coefficients below 25 are truncated to zero

<table>
<thead>
<tr>
<th>Decomposition Level (Scales or Filter Bank Iterations)</th>
<th>Approximation Coefficient Image</th>
<th>Truncated Coefficients (%)</th>
<th>Reconstruction Error (rms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$256 \times 256$</td>
<td>74.7%</td>
<td>3.27</td>
</tr>
<tr>
<td>2</td>
<td>$128 \times 128$</td>
<td>91.7%</td>
<td>4.23</td>
</tr>
<tr>
<td>3</td>
<td>$64 \times 64$</td>
<td>95.1%</td>
<td>4.54</td>
</tr>
<tr>
<td>4</td>
<td>$32 \times 32$</td>
<td>95.6%</td>
<td>4.61</td>
</tr>
<tr>
<td>5</td>
<td>$16 \times 16$</td>
<td>95.5%</td>
<td>4.63</td>
</tr>
</tbody>
</table>
JPEG-2000 uses wavelet-based coding

Compressing reconstruction  Scaled error image  Zoomed compression reconstruction

25:1

Compression ratio

52:1

CSE 166, Spring 2020
JPEG-2000 uses wavelet-based coding

Compression reconstruction  Scaled error image  Zoomed compression reconstruction

75:1

Compression ratio

105:1
Image watermarking

• Visible watermarks
• Invisible watermarks
Visible watermark

\[ f_w = (1 - \alpha) f + \alpha w \]

Watermarked image

Original image minus watermark
Invisible image watermarking system

Encoder

Decoder
Invisible watermark

Example: watermarking using two least significant bits

Original image

JPEG compressed

Extracted watermark

Two least significant bits

Fragile invisible watermark
Invisible watermark

Example: DCT-based watermarking

Watermarked images (different watermarks)

Extracted robust invisible watermark
Reflection and translation

\[ \hat{B} = \{ w \mid w = -b, \text{ for } b \in B \} \]

Reflection

\[ (B)_z = \{ c \mid c = b + z, \text{ for } b \in B \} \]

Translation
Sets of pixels: objects and structuring elements (SEs)

Objects represented as sets

Structuring element represented as a set

Objects represented as a graphical image

Structuring element represented as a graphical image

Digital image

Digital structuring element

Border of background pixels around objects

Tight border around SE
Reflection about the origin

Origin

Don’t care elements
Erosion

Example: square SE

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

\[ I \ominus B = \{ z \mid (B)_z \subseteq A \text{ and } A \in I \} \cup \{ A^c \mid A^c \subseteq I \} \]

Complement of A (i.e., set of elements not in A)
Erosion

Example: elongated SE

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

\[ I \ominus B = \{ z \mid (B)_z \subseteq A \text{ and } A \in I \} \cup \{ A^c \mid A^c \subseteq I \} \]
**Dilation**

\[ A \oplus B = \{ z \mid (\hat{B})_z \cap A \neq \emptyset \} \]

**Examples**

- **Square SE**
  - Background
  - Image, \( I \)
  - \( \hat{B} = B \)
  - \( I \oplus B \)

- **Elongated SE**
  - \( \hat{B} = B \)
  - \( A \oplus B \)
  - \( I \oplus B \)
Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

Dilation Expands
Opening

Structuring element rolls along inner boundary

\[ A \odot B = (A \ominus B) \oplus B \]
Closing

Structuring element rolls along outer boundary

\[ A \cdot B = (A \oplus B) \ominus B \]
Opening and closing

Erosion

Opening

Dilation

Closing
Morphological image processing

Noisy input

Opening

Dilation

Erosion

Closing

Dilation

Erosion
Boundary extraction

Erosion

Set difference
Boundary extraction
Hole filling

Given point in hole

\[ X_k = (X_{k-1} \oplus B) \cap A^c \quad k = 1, 2, \ldots \]
Hole filling

Given points in holes

All holes filled
Given point in $A$
Connected components

X-ray image

Threshold (negative)

15 connected components

<table>
<thead>
<tr>
<th>Connected component</th>
<th>No. of pixels in connected comp</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>11</td>
</tr>
<tr>
<td>02</td>
<td>9</td>
</tr>
<tr>
<td>03</td>
<td>9</td>
</tr>
<tr>
<td>04</td>
<td>39</td>
</tr>
<tr>
<td>05</td>
<td>133</td>
</tr>
<tr>
<td>06</td>
<td>1</td>
</tr>
<tr>
<td>07</td>
<td>1</td>
</tr>
<tr>
<td>08</td>
<td>743</td>
</tr>
<tr>
<td>09</td>
<td>7</td>
</tr>
<tr>
<td>10</td>
<td>11</td>
</tr>
<tr>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>12</td>
<td>9</td>
</tr>
<tr>
<td>13</td>
<td>9</td>
</tr>
<tr>
<td>14</td>
<td>674</td>
</tr>
<tr>
<td>15</td>
<td>85</td>
</tr>
</tbody>
</table>
Image segmentation

• General approach
  1. Spatial filtering
  2. Additional processing
  3. Thresholding

• Global thresholding (simplest)

\[ g(x, y) = \begin{cases} 
1 & \text{if } f(x, y) > T \\
0 & \text{otherwise}
\end{cases} \]

where

\( T \) is threshold value
Image segmentation

Input

Edges

Segmentation

Edge-based

Region-based
Derivatives in 1D

• Forward difference

\[
\frac{\partial f(x)}{\partial x} = \frac{f(x + \Delta x) - f(x)}{\Delta x}
\]

• Backward difference

\[
\frac{\partial f(x)}{\partial x} = \frac{f(x) - f(x - \Delta x)}{\Delta x}
\]

• Central difference

\[
\frac{\partial f(x)}{\partial x} = \frac{f(x + \Delta x) - f(x - \Delta x)}{2\Delta x}
\]
Image derivatives
Detection of isolated points

Laplacian (second derivative)

Input

Segmentation

Threshold absolute value
Line detection

Input

Threshold absolute value

Double lines

Laplacian (second derivative)

Threshold value
Line detection, specific directions

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>2</td>
<td>-1</td>
<td>-1</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>2</td>
<td>-1</td>
<td>2</td>
<td>-1</td>
</tr>
<tr>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>2</td>
<td>-1</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Horizontal, +45°, Vertical, -45°

Spatial filters
Line detection, specific directions

+45°

Negative values set to zero

Threshold
Edge models

- Step
- Ramp
- Roof edge
Edge models

Step

Ramp

Roof edge
Ramp edge

- Horizontal intensity profile
- First derivative
- Second derivative
- Zero crossing
- One point
- Two points
- First derivative
- Second derivative
Noise and image derivatives

Input  First derivative  Second derivative
Noise
Edge detection

1. Image smoothing for noise reduction
2. Detection of image points (edge point candidates)
3. Edge localization (select from candidates, set of edge points)
Gradient and edge direction

Gradient direction is orthogonal to edge direction
Gradient operators

Forward difference

-1

1

-1  1

-1  0  0  -1

0  1  1  0

Roberts

-1  -1  -1  -1  0  0  0  1

0  0  0  1  -1  0  1  1

1  1  1  1

Prewitt

-1  -2  -1  -1  0  0  0  1

0  0  0  1  -2  0  2  1

1  2  1  1  -1  0  1

Sobel
Gradients

- Input
- Magnitude of vertical gradient
- Magnitude of horizontal gradient
- Magnitude of gradient vector
Smooth image prior to computing gradients.
Results in more selective edges
Edge detection

1. Smooth the input image
2. Compute the gradient magnitude image
3. Apply nonmaximal suppression to the gradient magnitude image
4. Threshold the resulting image
Edge detection

Threshold magnitude of gradient vector

Without smoothing

With smoothing
Advanced edge detection

Input

Magnitude of gradient vector (with smoothing)

Marr-Hildreth

Canny

Figure 10.25 in textbook looks better
Canny edge detector

1. Smooth the input image with a Gaussian filter
2. Compute the gradient magnitude and angle images
3. Apply nonmaximal suppression to the gradient magnitude image
4. Use double thresholding and connectivity analysis to detect and link edges
Double thresholding

• Use a high threshold to start edge curves and a low threshold to continue them
  – Define two thresholds $T_H$ and $T_L$
  – Starting with output of nonmaximal suppression, find a point $q_0$, which is a local maximum greater than $T_H$
  – Start tracking an edge chain at pixel location $q_0$ in one of the two directions
  – Stop when gradient magnitude is less than $T_L$
Double thresholding

Single threshold

\[ T = 15 \]

\[ T = 5 \]

Double threshold

\[ T_H = 15 \text{ and } T_L = 5 \]
Thresholding

Histograms

Single threshold

Dual thresholds
Thresholding

• Single threshold

\[ g(x, y) = \begin{cases} 
1 & \text{if } f(x, y) > T \\
0 & \text{otherwise} 
\end{cases} \]

where \( T \) is threshold value

• Dual thresholds

\[ g(x, y) = \begin{cases} 
a & \text{if } f(x, y) > T_2 \\
b & \text{if } T_1 < f(x, y) \leq T_2 \\
c & \text{if } f(x, y) \leq T_1 
\end{cases} \]

where \( T_1 \) is first threshold value
\( T_2 \) is second threshold value
Noise and thresholding

Noise
Varying background and thresholding

Input

Intensity ramp

Product of input and intensity ramp
Basic global thresholding

Input

Intensity ramp

Threshold
Basic global thresholding

1. Select initial estimate of global threshold $T_0$

2. Group $G_1$ if $f(x, y) > T_i$
   Group $G_2$ if $f(x, y) \leq T_i$

3. $m_1$ is mean of intensity values in $G_1$
   $m_2$ is mean of intensity values in $G_2$

4. $T_i = \frac{1}{2}(m_1 + m_2)$

5. Repeat steps 2–4 until $|T_i - T_{i-1}| < \Delta T$

Value used to stop iterating
Optimum global thresholding

Input

Histogram

Basic global thresholding

Optimum global thresholding using Otsu’s method
Otsu’s method

1. Compute normalized histogram (similar to pdf)

2. Compute cumulative sums \( P_1(k) = \sum_{i=0}^{k} p_i \) \( k = 0, 1, \ldots, L - 1 \)
   (note \( P_2(k) = 1 - P_1(k) \) since \( P1 + P2 = 1 \))

3. Compute cumulative means \( m(k) = \sum_{i=0}^{k} i p_i \) \( k = 0, 1, \ldots, L - 1 \)

4. Compute global mean \( m_G = \sum_{i=0}^{L-1} i p_i = m(L - 1) \)

5. Compute between-class variance \( \sigma_B^2(k) = \frac{(m_GP_1(k) - m(k))^2}{P_1(k)(1-P_1(k))} \) \( k = 0, 1, \ldots, L - 1 \)

6. Otsu’s threshold \( k^* \) is the value of \( k \) for which \( \sigma_B^2(k) \) is maximum
   (average values if multiple maxima)
Image smoothing to improve global thresholding

Otsu’s method

Without smoothing

With smoothing
Image smoothing does not always improve global thresholding

Without smoothing

With smoothing

Otsu’s method
Edges to improve global thresholding

Mask image (thresholded gradient magnitude)

Optimum global thresholding using Otsu’s method
Edges to improve global thresholding

Mask image (thresholded absolute Laplacian)

Input

Masked input

Optimum global thresholding using Otsu’s method
Variable thresholding

Input

Global thresholding

Local thresholding using standard deviations

Local standard deviations
Variable thresholding

Input (spot shading)

Global thresholding

Local thresholding using moving averages
Variable thresholding

Global thresholding

Local thresholding using moving averages

Input (sinusoidal shading)
Segmentation by region growing

Input X-ray image

Final seed image

Initial seed image

Output image
Segmentation by region growing

\[ f(x, y) \] input image
\[ S(x, y) \] seed image containing ones at locations of seed points and zeros elsewhere

1. Find all connected components in \( S(x, y) \) and reduce each connected component to one pixel. Set all such pixels as 1 (all others to zero)

2. Calculate \( f_Q(x, y) \) containing results of applying predicate \( Q \)

3. Calculate \( g(x, y) \) by appending to each seed point in \( S \) all 1-valued pixels in \( f_Q \) that are 8-connected to that seed point

4. Label each connected component in \( g(x, y) \) with a different region label (e.g., integers)
Segmentation by region growing

Difference image

Difference image thresholded using dual thresholds

Difference image thresholded with the smallest of the dual thresholds

Segmentation by region growing
Advanced segmentation methods

- \( k \)-means clustering
- Superpixels
- Graph cuts
Segmentation using $k$-means clustering

Input

Segmentation using $k$-means, $k = 3$
Segmentation using \textit{k}-means clustering

1. Specify an initial set of means $\mathbf{m}_i \quad i = 1, 2, \ldots, k$

2. Assign each sample to the cluster whose mean is closest
   (ties are resolved arbitrarily, so samples are assigned to only one cluster)

3. Update cluster means
   
   $\mathbf{m}_i = \frac{1}{|C_i|} \sum_{\mathbf{z} \in C_i} \mathbf{z} \quad i = 1, 2, \ldots, k$

   where $|C_i|$ is the number of samples in set $C_i$

4. If mean vectors do not converge, then return to step 2
   Test for convergence:
   
   $\sum_{i=1}^{k} \| \mathbf{m}_i^{(t)} - \mathbf{m}_i^{(t-1)} \| \leq T$

   where $T$ is a specified threshold
Superpixels

Input image of 480,000 pixels

Image of 4,000 superpixels with boundaries

Image of 4,000 superpixels
Superpixels

1,000 superpixels  500 superpixels  250 superpixels
Superpixels for image segmentation

Input image of 301,678 pixels

Segmentation using $k$-means, $k = 3$

Superpixel image (100 superpixels)

Segmentation using $k$-means, $k = 3$
Images as graphs

Simple graph with edges only between 4-connected neighbors

Stronger (greater weight) edges are darker
Graph cuts for image segmentation

Cut the weak edges
Graph cuts for image segmentation

Input

Smoothed input

Graph cut segmentation
CSE 166

• Image acquisition
• Geometric transformations and image interpolation
• Intensity transformations
• Spatial filtering
• Fourier transform and filtering in the frequency domain
• Image restoration
• Color image processing

• Basis vectors
• Matrix-based transforms
• Basis images
• Wavelet transform
• Image compression
• Image watermarking
• Morphological image processing
• Edge detection
• Segmentation