Filtering

Introduction to Computer Vision
CSE 152
Lecture 7

Announcements
- Assignment 2: Posted on web site, due 4/27
- See links on web page for reading on binary image processing (e-reserves)
- Text for filtering

Binary Image Processing: Basic Steps
1. Labeling pixels as foreground/background (0,1).
2. Morphological operators (sometimes)
3. Find pixels corresponding to a region
4. Compute properties of each region

Histogram-based Segmentation
- Select threshold
- Create binary image:
  \[ I(x,y) < T \rightarrow O(x,y) = 0 \]
  \[ I(x,y) > T \rightarrow O(x,y) = 1 \]

P-Tile Method
- If the size of the object is approx. known, pick \( T \) s.t. the area under the histogram corresponds to the size of the object:

“Peakiness” Detection Algorithm
- Find the two highest local maxima at a minimum distance apart, \( g_i \) and \( g_j \)
- Find lowest point between them, \( g_k \)
- Measure “peakiness”:
  \[ \min(\text{height}(g_i), \text{height}(g_j))/\text{height}(g_k) \]
- Find \((g_i, g_j, g_k)\) with highest peakiness
Four & Eight Connectedness

Recursive Labeling
Connected Component Exploration

Properties extracted from binary image
- A tree showing containment of regions
- Properties of a region
  1. Genus – number of holes
  2. Centroid
  3. Area
  4. Perimeter
  5. Moments (e.g., measure of elongation)
  6. Number of “extrema” (indentations, bulges)
  7. Skeleton

Procedure Label (Pixel)
BEGIN
Mark(Pixel) <- Marker;
FOR neighbor in Neighbors(Pixel) DO
  IF Image(neighbor) = 1 AND Mark(neighbor)=nil THEN
    Label(neighbor)
END
BEGIN Main
Marker <- 0;
FOR Pixel in Image DO
  IF Image(Pixel) = 1 AND Mark(Pixel)=nil THEN
    BEGIN
      Marker <- Marker + 1;
      Label(Pixel);
    END;
END
END
Globals:
Marker: integer
Mark: Matrix same size as Image, initialized to NIL

Moments
0 S = \{(x, y)|f(x, y) = 1\}

Given a pair of non-negative integers (j, k), the discrete (j, k)th moment of S is:
M_{jk}(S) = \sum_{(x, y)\in S} x^j y^k

M_{/j} = \sum_{x=1}^{n} \sum_{y=1}^{m} B(x, y) x / y^j
- Fast way to implement computation over n by m image or window
- One object

Area: Moment M_{00}
0 S = \{(x, y)|f(x, y) = 1\}
M_{jk}(S) = \sum_{(x,y)\in S} x^j y^k

Example:
M_{00}(S) = \sum_{(x,y)\in S} x^0 y^0 = \sum_{(x,y)\in S} 1 = \#(S)

Area of S !!
Computing the centroid with Moments

Example:

\[ S = \{(x, y) | f(x, y) = 1\} \]

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

\[ M_{10}(S) = \sum_{(x, y) \in S} x y, \quad M_{01}(S) = \sum_{(x, y) \in S} y \]

\[ \frac{M_{10}(S)}{M_{00}(S)} = \frac{\sum_{(x, y) \in S} x y}{\sum_{(x, y) \in S} y} = \bar{x} \quad \frac{M_{01}(S)}{M_{00}(S)} = \frac{\sum_{(x, y) \in S} y}{\sum_{(x, y) \in S} y} = \bar{y} \]

Center of gravity (Centroid) of \( S \)

Shape recognition by Moments

Recognition could be done by comparing moments:

However, moments \( M_{jk} \) are not invariant under:

- Translation
- Scaling
- Rotation
- Skewing

Central Moments

Given a pair of non-negative integers \((j, k)\) the central \((j, k)\)th moment of \( S \) is given by:

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

Normalized Moments

Given a pair of non-negative integers \((j, k)\) the normalized \((j, k)\)th moment of \( S \) is given by:

\[ m_{jk}(S) = \sum_{(x, y) \in S} \left( \frac{x - \bar{x}}{\sigma_x} \right)^j \left( \frac{y - \bar{y}}{\sigma_y} \right)^k \]

Central Moments

Given a pair of non-negative integers \((j, k)\) the central \((j, k)\)th moment of \( S \) is given by:

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

Normalized Moments

Given a pair of non-negative integers \((j, k)\) the normalized \((j, k)\)th moment of \( S \) is given by:

\[ m_{jk}(S) = \sum_{(x, y) \in S} \left( \frac{x - \bar{x}}{\sigma_x} \right)^j \left( \frac{y - \bar{y}}{\sigma_y} \right)^k \]
Region orientation from Second Moment Matrix

1. Compute second centralized moment matrix
   \[
   \begin{bmatrix}
   \mu_{20} & \mu_{11} \\
   \mu_{11} & \mu_{02}
   \end{bmatrix}
   \]
2. Compute Eigenvectors of Moment Matrix to obtain orientation
3. Eigenvalues are independent of orientation, translation!

Scale, Rotation, Translation Invariance

1. Compute second normalized moment matrix
2. Eigenvectors give orientation of object.
3. Eigenvalues are translation, rotation, and scale invariance.

Binary System Summary

1. Acquire images and binarize (thresholding, color labels, etc.).
2. Possibly clean up image using morphological operators.
3. Determine regions (blobs) using connected component exploration
4. Compute position, area, and orientation of each blob using Moments
5. Compute features that are rotation, scale, and orientation invariant using Moments (e.g., Eigenvalues of Normalized Moments).

Filtering

Why perform filtering?

1. Reduce the effect of “noise”
2. Extract descriptions of a neighborhood of an image about a point
3. Extract features of an image (edges, texture, corners).
4. Enhance aspects of an image, visual improvement, etc.
What is image filtering?

- Modify the pixels in an image based on some function of a local neighborhood of the pixels.

![Image of a local image data and modified image data]

(From Bill Freeman)

Linear Filters

- **General process:**
  - Form a new image whose pixel values are a weighted sum of original pixel values, using the same set of weights at each point.

- **Properties**
  - Output is a linear function of the input
  - Output is a shift-invariant function of the input (i.e., shift the input image two pixels to the left, the output is shifted two pixels to the left)

- Example: smoothing by averaging
  - Form the average of pixels in a neighborhood

- Example: smoothing with a Gaussian
  - Form a weighted average of pixels in a neighborhood

- Example: finding a derivative
  - Form a weighted average of pixels in a neighborhood

Convolution

- **Equation:**
  $$ R(i, j) = \sum_{k=-m}^{m} \sum_{l=-m}^{m} K(h, k) I(i-h, j-k) $$

- **Kernel size:**
  - is $m+1$ by $m+1$

Note: Typically Kernel is relatively small in vision applications.
Convolution: \( R = K \ast I \)

**Kernel size is \( m+1 \) by \( m+1 \)**

\[
R(i, j) = \sum_{k=-m}^{m} \sum_{h=-m}^{m} K(h, k) I(i-h, j-k)
\]

\( m = 2 \)
Convolution: \( R = K \ast I \)

Kernel size is \( m+1 \) by \( m+1 \)

\[
R(i, j) = \sum_{h=0}^{m} \sum_{k=0}^{m} K(h, k) I(i-h, j-k)
\]

Linear filtering (warm-up slide)

Original coefficient 1.0 Pixel offset ?

Impulse Response

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{array}
\]

Linear filtering (warm-up slide)

Original coefficient 1.0 Pixel offset Filtered (no change)
Linear filtering

original

coefficient 1.0

Pixel offset

?

shift

original

coefficient 1.0

Pixel offset

shifted

Linear filtering

original

coefficient 0.3

Pixel offset

?

Blurring

original

coefficient 0.3

Pixel offset

Blurred (filter applied in both dimensions).

Blur examples

impulse

original

coefficient 8

Pixel offset

filtered

Blur examples

impulse

original

coefficient 8

Pixel offset

filtered

c

edge

original

coefficient 8

Pixel offset

filtered
Linear filtering (warm-up slide)

original

Linear filtering (no change)

original

Filtered (no change)

Linear filtering

original

(remember blurring)

original

Pixel offset

Blurred (filter applied in both dimensions).

Sharpening

original

Sharpened original

Sharpening

before

after
Smoothing by Averaging

**Kernel:**

- Noise is what we're not interested in.
  - We'll discuss simple, low-level noise today: Light fluctuations; Sensor noise; Quantization effects; Finite precision
  - Not complex: shadows; extraneous object.
- A pixel's neighborhood contains information about its intensity.
- Averaging noise reduces its effect.

**Noise**

- Simplest noise model
  - Independent stationary additive Gaussian noise
    \( N(i,j) = I(i,j) - I(i,j) \)
  - \( N(i,j) \) is a Gaussian Random Variable
  - The noise value at each pixel is given by an independent draw from the same normal probability distribution

- Issues
  - This model allows noise values that could be greater than maximum camera output or less than zero
  - For small standard deviations, this isn't too much of a problem; it's a fairly good model
  - Independence may not be justified (e.g., damage to lens)
  - May not be stationary (e.g., thermal gradients in the CCD)

**Average Filter**

- Mask with positive entries, that sum 1.
- Replaces each pixel with an average of its neighborhood.
- If all weights are equal, it is called a BOX filter.

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]
**An Isotropic Gaussian**

- The picture shows a smoothing kernel proportional to
  \[
  \exp \left( -\frac{x^2 + y^2}{2\sigma^2} \right)
  \]

  (which is a reasonable model of a circularly symmetric fuzzy blob)

---

**Smoothing by Averaging**

- Smoothing with a Gaussian
  - Kernel: \( \sigma \)
  - The effects of smoothing:
    - Each row shows smoothing with gaussians of different widths, with each column showing different realizations of an image of gaussian noise.

---

**Efficient Implementation**

- Both, the BOX filter and the Gaussian filter are separable:
  - First convolve each row with a 1D filter
  - Then convolve each column with a 1D filter.