Lecture-12

Hough Transform
Examples

Hough Space

Theta is from -90 to +90
Fitting Lines In an Image

Fitting Lines In an Image
Fitting lines in an image

Fitting Circles
Fitting Circles

Detect yellow line in the middle
Use gray levels instead of edges
Increment the parameter space by gray level at a pixel instead of by 1.

Detecting Lines in Gray Level Images

Detect yellow line in the middle
Use gray levels instead of edges
Increment the parameter space by gray level at a pixel instead of by 1.
Pyramids

• Very useful for representing images.
• Pyramid is built by using multiple copies of image.
• Each level in the pyramid is 1/4 of the size of previous level.
• The lowest level is of the highest resolution.
• The highest level is of the lowest resolution.
Gaussian Pyramids

\[ g_l(i, j) = \sum_{m=\lfloor i/2 \rfloor}^{2} \sum_{n=\lfloor j/2 \rfloor}^{2} w(m, n) g_{l/2}(2i + m, 2j + n) \]

\[ g_l = REDUCE[g_{l/2}] \]

Convolution

\[ f \bigotimes h = h(x, y) \]
Gaussian Pyramids

\[ g_{l,n}(i, j) = \sum_{p=1}^{2} \sum_{q=1}^{2} w(p, q) g_{1,n\square}(\frac{i \square p}{2}, \frac{j \square q}{2}) \]

\[ g_{l,n} = EXPAND[ g_{l,n\square} ] \]

Reduce (1D)

\[ g_{l}(i) = \sum_{m=1}^{2} \hat{w}(m) g_{l\square}(2i + m) \]

\[ g_{l}(2) = \hat{w}(2)g_{l\square}(4 \square 2) + \hat{w}(1)g_{l\square}(4 \square 1) + \hat{w}(0)g_{l\square}(4) + \hat{w}(1)g_{l\square}(4 + 1) + \hat{w}(2)g_{l\square}(4 + 2) \]

\[ g_{l}(2) = \hat{w}(2)g_{l\square}(2) + \hat{w}(1)g_{l\square}(3) + \hat{w}(0)g_{l\square}(4) + \hat{w}(1)g_{l\square}(5) + \hat{w}(2)g_{l\square}(6) \]
Reduce

Gaussian Pyramid

Expand (1D)

\[
g_{l,n}(i) = \prod_{p=\square}^{2} \hat{w}(p) g_{l,n\square}(i \square p)\]

\[
g_{l,n}(4) = \hat{w}(\square 2) g_{l,n\square}(\frac{4 \square 2}{2}) + \hat{w}(\square 1) g_{l,n\square}(\frac{4 \square 1}{2}) +
\]

\[
\hat{w}(0) g_{l,n\square}(\frac{4}{2}) + \hat{w}(1) g_{l,n\square}(\frac{4+1}{2}) + \hat{w}(2) g_{l,n\square}(\frac{4+2}{2})
\]

\[
g_{l,n}(4) = \hat{w}(\square 2) g_{l,n\square}(1) + \hat{w}(0) g_{l,n\square}(2) + \hat{w}(2) g_{l,n\square}(3)
\]
Expand (1D)

\[ g_{l,n}(i) = \prod_{p=1}^{2} \hat{w}(p) g_{l,n[p]} \left( \frac{i - p}{2} \right) \]

\[ g_{l,n}(3) = \hat{w}(\|2\|) g_{l,n[\|]} \left( \frac{3\|2\|}{2} \right) + \hat{w}(\|1\|) g_{l,n[\|]} \left( \frac{3\|1\|}{2} \right) + \]

\[ \hat{w}(0) g_{l,n[\|]} \left( \frac{3}{2} \right) + \hat{w}(1) g_{l,n[\|]} \left( \frac{3 + 1}{2} \right) + \hat{w}(2) g_{l,n[\|]} \left( \frac{3 + 2}{2} \right) \]

\[ g_{l,n}(3) = \hat{w}(\|1\|) g_{l,n[\|]} (1) + \hat{w}(1) g_{l,n[\|]} (2) \]

Expand

**Gaussian Pyramid**

\[ g_{l,1} \]

\[ g_{l,1,1} = \text{EXPAND}[g_{1,1}] \]
Convolution Mask

\[ [w(2), w(1), w(0), w(1), w(2)] \]

Convolution Mask

- Separable
  \[
  w(m, n) = \hat{w}(m)\hat{w}(n)
  \]
- Symmetric
  \[
  \hat{w}(i) = \hat{w}(\square i)
  \]

\[ [c, b, a, b, c] \]
Convolution Mask

- The sum of mask should be 1.
  \[ a + 2b + 2c = 1 \]

- All nodes at a given level must contribute the same total weight to the nodes at the next higher level.
  \[ a + 2c = 2b \]
Convolution Mask

\[ \hat{w}(0) = a \]
\[ \hat{w}(1) = \hat{w}(1) = \frac{1}{4} \]
\[ \hat{w}(2) = \hat{w}(2) = \frac{1}{4} \times \frac{a}{2} \]

a=.4 GAUSSIAN, a=.5 TRINGULAR

Triangular
Approximate Gaussian

Gaussian

\[ g(x) = e^{\frac{-x^2}{2\sigma^2}} \]
Gaussian

\[ g(x) = e^{\frac{-x^2}{2\sigma^2}} \]
Separability

Algorithm

- Apply 1-D mask to alternate pixels along each row of image.
- Apply 1-D mask to alternate pixel along each column of resultant image from previous step.
Gaussian Pyramid

- Similar to edge detected images.
- Most pixels are zero.
- Can be used for image compression.

\[
L_1 = g_1 \bigtriangleup EXPAND[g_2]
\]
\[
L_2 = g_2 \bigtriangleup EXPAND[g_3]
\]
\[
L_3 = g_3 \bigtriangleup EXPAND[g_4]
\]

Laplacian Pyramids
Coding using Laplacian Pyramid

• Compute Gaussian pyramid

$$g_1, g_2, g_3, g_4$$

• Compute Laplacian pyramid

$$L_1 = g_1 \cdot \text{EXPAND}[g_2]$$
$$L_2 = g_2 \cdot \text{EXPAND}[g_3]$$
$$L_3 = g_3 \cdot \text{EXPAND}[g_4]$$
$$L_4 = g_4$$

• Code Laplacian pyramid
Decoding using Laplacian pyramid

- Decode Laplacian pyramid.
- Compute Gaussian pyramid from Laplacian pyramid.

\[
g_4 = L_4 \\
g_3 = \text{EXPAND}[g_4] + L_3 \\
g_2 = \text{EXPAND}[g_3] + L_2 \\
g_1 = \text{EXPAND}[g_2] + L_1
\]

- is reconstructed image.
Image Compression (Entropy)

Huffman Coding (Example-1)

A_1

P=.5

A_2

P=.25

A_3

P=.125

A_4

P=.125
Huffman Coding

Entropy  \[ H = \sum_{i=0}^{255} p(i) \log_2 p(i) \]

\[ H = 0.5 \log_5 0.25 \log_2 0.125 \log_2 0.125 = 1.75 \]

Image Compression

1.58

1

0.73
Combining Apple & Orange

Combining Apple & Orange
Algorithm

- Generate Laplacian pyramid $L_0$ of orange image.
- Generate Laplacian pyramid $L_a$ of apple image.
- Generate Laplacian pyramid $L_c$ by copying left half of nodes at each level from apple and right half of nodes from orange pyramids.
- Reconstruct combined image from $L_c$.

Quad Trees

- Data structure to represent regions
- Three types of nodes: gray, black and white
- First generate the pyramid, then:
  - If type of pyramid is black or white then return else
    - Recursively find quad tree of SE quadrant
    - Recursively find quad tree of SW quadrant
    - Recursively find quad tree of NE quadrant
    - Recursively find quad tree of NW quadrant
    - Return
Chain Code

- A simple technique to represent a shape of boundary.
- Each directed line segment is assigned a code.
- Chain code is integer obtained by putting together the codes of all consecutive line segments.
- Shape number is a normalized chain code, which is invariant to translation and rotation.