Image pyramids
Slide credits: Mubarak Shah

Samuel Cheng

School of ECE
University of Oklahoma

Spring, 2018

Table of Contents

Reference

cited: 7343

From Wikipedia
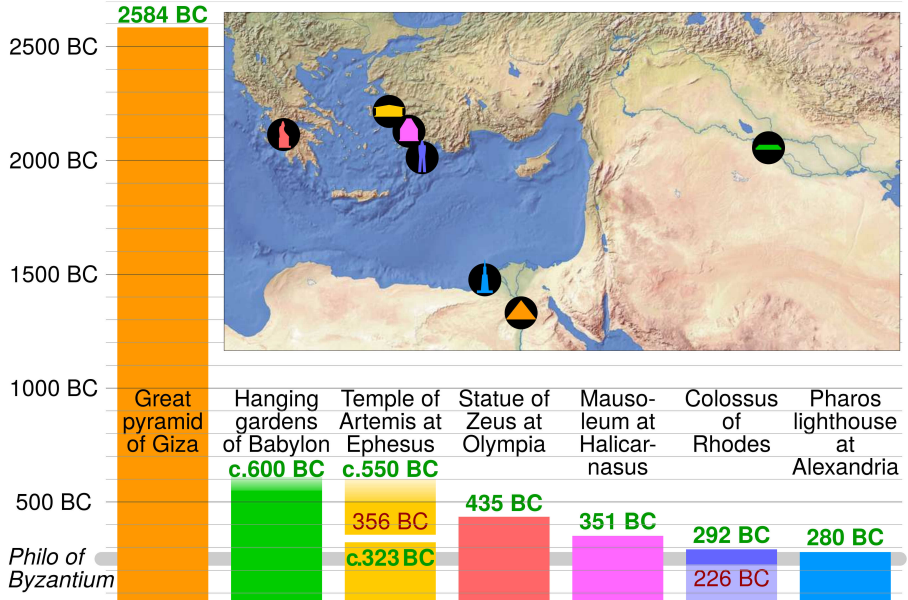
Image pyramid

- Very useful for representing images
- Pyramid is built as multiple resolution approximations of a same image
- Each level in the pyramid is 1/4 of the size of the previous level
- Lowest level has highest resolution
- Highest level has lowest resolution

https://www.pyimagesearch.com/2015/03/16/image-pyramids-with-python-and-opencv/
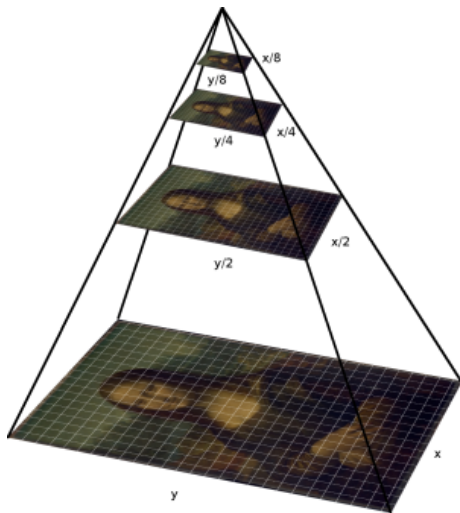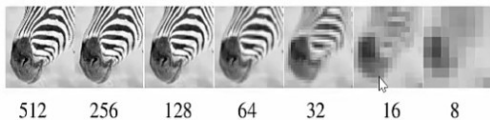
Gaussian pyramid

Source: Forsyth

Laplacian pyramid

512 256 128 64 32 16 8
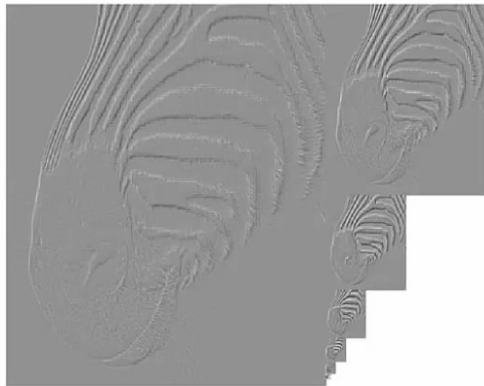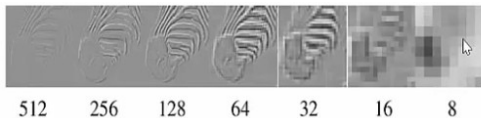
Source: Forsyth

Image pyramids

Things to learn today

- Gaussian and Laplacian pyramid
    - Reduce
    - Expand
- Applications of Laplacian pyramid
    - Image compression
    - Image composting

Reduce operation

$$g_l = REDUCE[g_{l-1}]$$

$$\underbrace{g_l(i,j)}_{l\text{-level}} = \sum_{m=-2}^{2} \sum_{n=-2}^{2} \tilde{w}(m,n)\underbrace{g_{l-1}(2i-m, 2j-n)}_{l-1\text{-level}}$$

Reduce operation

$$g_l = REDUCE[g_{l-1}]$$

$$\underbrace{g_l(i,j)}_{l\text{-level}} = \sum_{m=-2}^{2} \sum_{n=-2}^{2} \tilde{w}(m,n)\underbrace{g_{l-1}(2i-m, 2j-n)}_{l-1\text{-level}}$$

Remark

Note that it is different from convolution that we skip every one sample per dimension for the input

1-D case

$$g_l(i) = \sum_{m=-2}^{2} w(m) g_{l-1}(2i - m)$$

1-D case

$$g_l(i) = \sum_{m=-2}^{2} w(m)g_{l-1}(2i - m)$$

1-D case

$$g_l(i) = \sum_{m=-2}^{2} w(m)g_{l-1}(2i - m)$$

Image pyramids
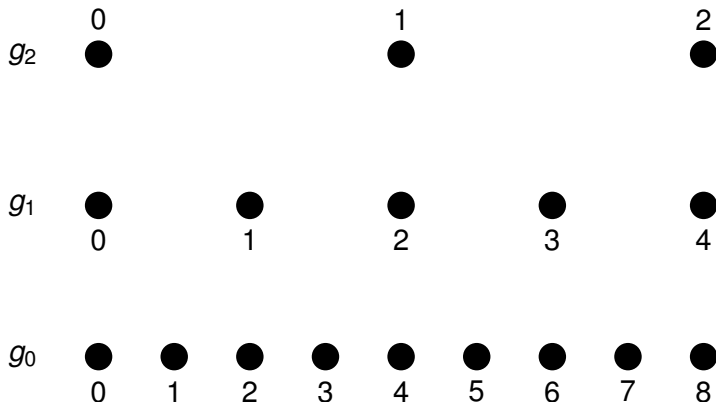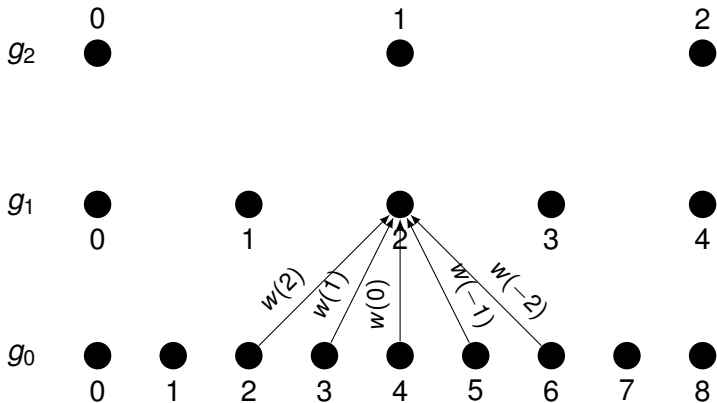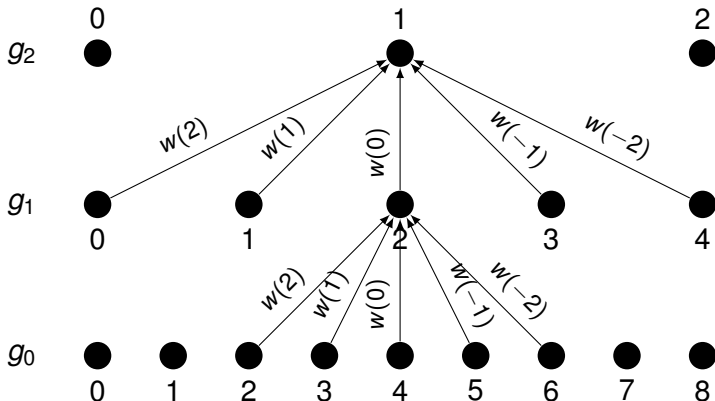
Expand operation

$$\hat{g}_l = EXPAND[\hat{g}_{l-1}]$$

$$\hat{g}_l(i,j) = \sum_{n=-2}^{2} \sum_{m=-2}^{2} \tilde{w}(m,n)\hat{g}_{l-1}\left(\frac{i-n}{2}, \frac{j-m}{2}\right)$$

Remark

EXPAND is approximately the inverse operation of REDUCE

S. Cheng (OU ECE)   Image pyramids   Feb 2017   12 / 23

1-D case

$$\hat{g}_l(i) = \sum_{m=-2}^{2} w(m)\hat{g}_{l-1}\left(\frac{i-m}{2}\right)$$

1-D case

$$\hat{g}_l(i) = \sum_{m=-2}^{2} w(m)\hat{g}_{l-1}\left(\frac{i-m}{2}\right)$$

$$\hat{g}_l(2) = w(-2)\hat{g}_{l-1}\left(\frac{2+2}{2}\right) + w(-1)\hat{g}_{l-1}\left(\frac{2+1}{2}\right)$$
$$+ w(0)\hat{g}_{l-1}\left(\frac{2}{2}\right) + w(1)\hat{g}_{l-1}\left(\frac{2-1}{2}\right) + w(2)\hat{g}_{l-1}\left(\frac{2-2}{2}\right)$$

1-D case

$$\hat{g}_l(i) = \sum_{m=-2}^{2} w(m)\hat{g}_{l-1}\left(\frac{i-m}{2}\right)$$

$$\hat{g}_l(2) = w(-2)\hat{g}_{l-1}\left(\frac{2+2}{2}\right) + \cancel{w(-1)\hat{g}_{l-1}\left(\frac{2+1}{2}\right)}$$
$$+ w(0)\hat{g}_{l-1}\left(\frac{2}{2}\right) + \cancel{w(1)\hat{g}_{l-1}\left(\frac{2-1}{2}\right)} + w(2)\hat{g}_{l-1}\left(\frac{2-2}{2}\right)$$

1-D case

$$\hat{g}_l(i) = \sum_{m=-2}^{2} w(m)\hat{g}_{l-1}\left(\frac{i-m}{2}\right)$$

$$\hat{g}_l(2) = w(-2)\hat{g}_{l-1}\left(\frac{2+2}{2}\right) + \cancel{w(-1)\hat{g}_{l-1}\left(\frac{2+1}{2}\right)}$$
$$+ w(0)\hat{g}_{l-1}\left(\frac{2}{2}\right) + \cancel{w(1)\hat{g}_{l-1}\left(\frac{2-1}{2}\right)} + w(2)\hat{g}_{l-1}\left(\frac{2-2}{2}\right)$$
$$= w(-2)\hat{g}_{l-1}(2) + w(0)\hat{g}_{l-1}(1) + w(2)\hat{g}_{l-1}(0)$$

1-D case

$$\hat{g}_l(i) = \sum_{m=-2}^{2} w(m)\hat{g}_{l-1}\left(\frac{i-m}{2}\right)$$

$$\hat{g}_l(2) = w(-2)\hat{g}_{l-1}\left(\frac{2+2}{2}\right) + \cancel{w(-1)\hat{g}_{l-1}\left(\frac{2+1}{2}\right)}$$
$$+ w(0)\hat{g}_{l-1}\left(\frac{2}{2}\right) + \cancel{w(1)\hat{g}_{l-1}\left(\frac{2-1}{2}\right)} + w(2)\hat{g}_{l-1}\left(\frac{2-2}{2}\right)$$
$$= w(-2)\hat{g}_{l-1}(2) + w(0)\hat{g}_{l-1}(1) + w(2)\hat{g}_{l-1}(0)$$

$$\hat{g}_l(5) = w(-2)\hat{g}_{l-1}\left(\frac{5+2}{2}\right) + w(-1)\hat{g}_{l-1}\left(\frac{5+1}{2}\right) + w(0)\hat{g}_{l-1}\left(\frac{5}{2}\right)$$
$$+ w(1)\hat{g}_{l-1}\left(\frac{5-1}{2}\right) + w(2)\hat{g}_{l-1}\left(\frac{5-2}{2}\right)$$

1-D case

$$\hat{g}_l(i) = \sum_{m=-2}^{2} w(m)\hat{g}_{l-1}\left(\frac{i-m}{2}\right)$$

$$\hat{g}_l(2) = w(-2)\hat{g}_{l-1}\left(\frac{2+2}{2}\right) + \cancel{w(-1)\hat{g}_{l-1}\left(\frac{2+1}{2}\right)}$$
$$+ w(0)\hat{g}_{l-1}\left(\frac{2}{2}\right) + \cancel{w(1)\hat{g}_{l-1}\left(\frac{2-1}{2}\right)} + w(2)\hat{g}_{l-1}\left(\frac{2-2}{2}\right)$$
$$= w(-2)\hat{g}_{l-1}(2) + w(0)\hat{g}_{l-1}(1) + w(2)\hat{g}_{l-1}(0)$$

$$\hat{g}_l(5) = \cancel{w(-2)\hat{g}_{l-1}\left(\frac{5+2}{2}\right)} + w(-1)\hat{g}_{l-1}\left(\frac{5+1}{2}\right) + \cancel{w(0)\hat{g}_{l-1}\left(\frac{5}{2}\right)}$$
$$+ w(1)\hat{g}_{l-1}\left(\frac{5-1}{2}\right) + \cancel{w(2)\hat{g}_{l-1}\left(\frac{5-2}{2}\right)}$$

1-D case

$$\hat{g}_l(i) = \sum_{m=-2}^{2} w(m)\hat{g}_{l-1}\left(\frac{i-m}{2}\right)$$

$$\hat{g}_l(2) = w(-2)\hat{g}_{l-1}\left(\frac{2+2}{2}\right) + \cancel{w(-1)\hat{g}_{l-1}\left(\frac{2+1}{2}\right)}$$
$$+ w(0)\hat{g}_{l-1}\left(\frac{2}{2}\right) + \cancel{w(1)\hat{g}_{l-1}\left(\frac{2-1}{2}\right)} + w(2)\hat{g}_{l-1}\left(\frac{2-2}{2}\right)$$
$$= w(-2)\hat{g}_{l-1}(2) + w(0)\hat{g}_{l-1}(1) + w(2)\hat{g}_{l-1}(0)$$

$$\hat{g}_l(5) = \cancel{w(-2)\hat{g}_{l-1}\left(\frac{5+2}{2}\right)} + w(-1)\hat{g}_{l-1}\left(\frac{5+1}{2}\right) + \cancel{w(0)\hat{g}_{l-1}\left(\frac{5}{2}\right)}$$
$$+ w(1)\hat{g}_{l-1}\left(\frac{5-1}{2}\right) + \cancel{w(2)\hat{g}_{l-1}\left(\frac{5-2}{2}\right)}$$
$$= w(-1)\hat{g}_{l-1}(3) + w(1)\hat{g}_{l-1}(2)$$
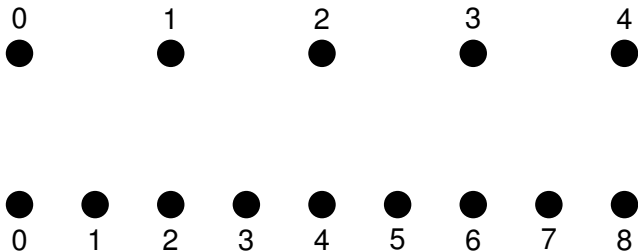
1-D case

$$\hat{g}_l(i) = \sum_{m=-2}^{2} w(m)\hat{g}_{l-1}\left(\frac{i-m}{2}\right)$$

0   1   2   3   4

● ● ● ● ●

● ● ● ● ● ● ● ● ●
0 1 2 3 4 5 6 7 8

Remark

Note that since we have discard half of the coefficients during expansion, we should double the weights $w(m)$ comparing to those during reduction

1-D case
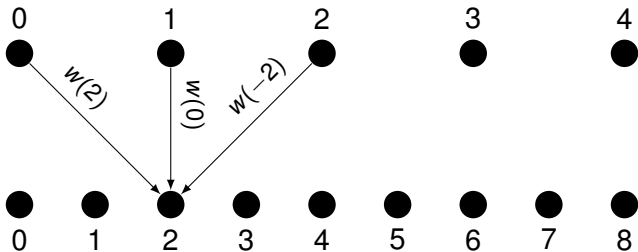
$$\hat{g}_l(i) = \sum_{m=-2}^{2} w(m)\hat{g}_{l-1}\left(\frac{i-m}{2}\right)$$

Remark

Note that since we have discard half of the coefficients during expansion, we should double the weights w(m) comparing to those during reduction

1-D case

$$\hat{g}_l(i) = \sum_{m=-2}^{2} w(m)\hat{g}_{l-1}\left(\frac{i-m}{2}\right)$$

Remark

Note that since we have discard half of the coefficients during expansion, we should double the weights $w(m)$ comparing to those during reduction

Design properties of convolution mask

Separable
- That is, $\tilde{w}(m, n) = w(m)w(n)$
- We can save computational cost (from N^2 to $2N$)

Design properties of convolution mask

Separable
- That is, $\tilde{w}(m, n) = w(m)w(n)$
- We can save computational cost (from N^2 to $2N$)

Symmetric
- That is, $w(i) = w(-i)$
- $[w(-2), w(-1), w(0), w(1), w(2)] = [c, b, a, b, c]$

Design properties of convolution mask

Separable
- That is, $\tilde{w}(m, n) = w(m)w(n)$
- We can save computational cost (from N^2 to $2N$)

Symmetric
- That is, $w(i) = w(-i)$
- $[w(-2), w(-1), w(0), w(1), w(2)] = [c, b, a, b, c]$

Normalized
- Sum of mask should be 1
- Thus, $a + 2b + 2c = 1$

Design properties of convolution mask

Separable
- That is, $\tilde{w}(m, n) = w(m)w(n)$
- We can save computational cost (from N^2 to $2N$)

Symmetric
- That is, $w(i) = w(-i)$
- $[w(-2), w(-1), w(0), w(1), w(2)] = [c, b, a, b, c]$

Normalized
- Sum of mask should be 1
- Thus, $a + 2b + 2c = 1$

"Unbiased"
- All nodes at a given level should contribute the same to nodes at the next level

Design properties of convolution mask

Separable
- That is, $\tilde{w}(m, n) = w(m)w(n)$
- We can save computational cost (from N^2 to $2N$)

Symmetric
- That is, $w(i) = w(-i)$
- $[w(-2), w(-1), w(0), w(1), w(2)] = [c, b, a, b, c]$

Normalized
- Sum of mask should be 1
- Thus, $a + 2b + 2c = 1$

"Unbiased"
- All nodes at a given level should contribute the same to nodes at the next level

Design properties of convolution mask

Separable
- That is, $\tilde{w}(m, n) = w(m)w(n)$
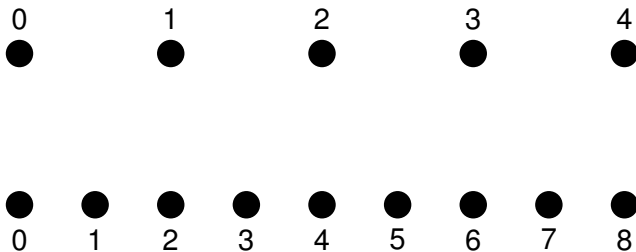- We can save computational cost (from N^2 to $2N$)

Symmetric
- That is, $w(i) = w(-i)$
- $[w(-2), w(-1), w(0), w(1), w(2)] = [c, b, a, b, c]$

Normalized
- Sum of mask should be 1
- Thus, $a + 2b + 2c = 1$

"Unbiased"
- All nodes at a given level should contribute the same to nodes at the next level

Design properties of convolution mask

Separable
- That is, $\tilde{w}(m, n) = w(m)w(n)$
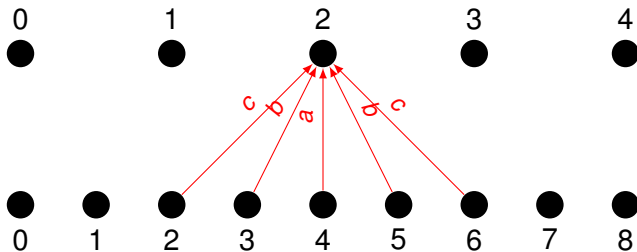- We can save computational cost (from N^2 to $2N$)

Symmetric
- That is, $w(i) = w(-i)$
- $[w(-2), w(-1), w(0), w(1), w(2)] = [c, b, a, b, c]$

Normalized
- Sum of mask should be 1
- Thus, $a + 2b + 2c = 1$

"Unbiased"
- All nodes at a given level should contribute the same to nodes at the next level

Design properties of convolution mask

Separable
- That is, $\tilde{w}(m, n) = w(m)w(n)$
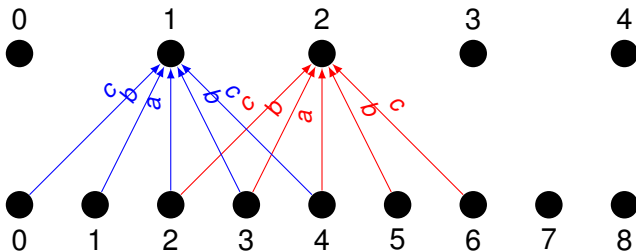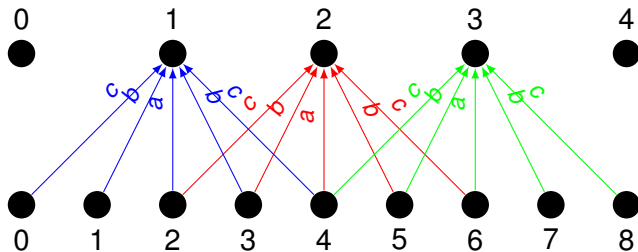- We can save computational cost (from N^2 to $2N$)

Symmetric
- That is, $w(i) = w(-i)$
- $[w(-2), w(-1), w(0), w(1), w(2)] = [c, b, a, b, c]$

Normalized
- Sum of mask should be 1
- Thus, $a + 2b + 2c = 1$

"Unbiased"
- All nodes at a given level should contribute the same to nodes at the next level

Design properties of convolution mask

Separable
- That is, $\tilde{w}(m, n) = w(m)w(n)$
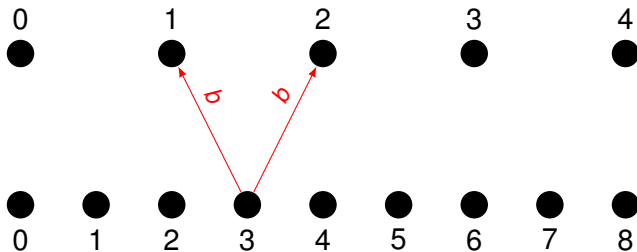- We can save computational cost (from N^2 to $2N$)

Symmetric
- That is, $w(i) = w(-i)$
- $[w(-2), w(-1), w(0), w(1), w(2)] = [c, b, a, b, c]$

Normalized
- Sum of mask should be 1
- Thus, $a + 2b + 2c = 1$

"Unbiased"
- All nodes at a given level should contribute the same to nodes at the next level

Design properties of convolution mask

Separable
- That is, $\tilde{w}(m, n) = w(m)w(n)$
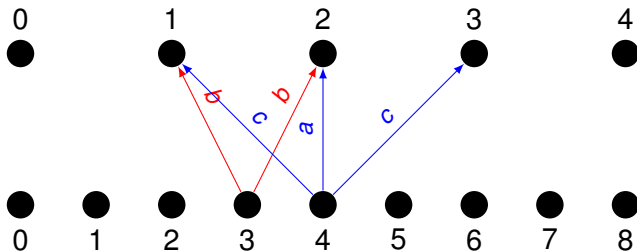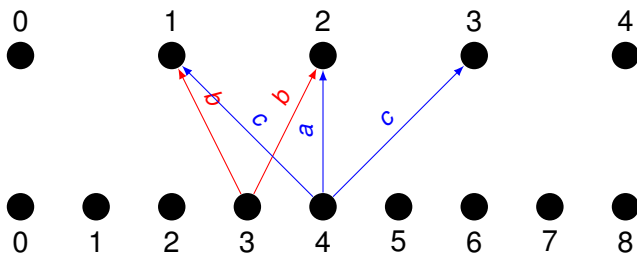- We can save computational cost (from N^2 to $2N$)

Symmetric
- That is, $w(i) = w(-i)$
- $[w(-2), w(-1), w(0), w(1), w(2)] = [c, b, a, b, c]$

Normalized
- Sum of mask should be 1
- Thus, $a + 2b + 2c = 1$

"Unbiased"
- All nodes at a given level should contribute the same to nodes at the next level ($2b = 2c + a$)
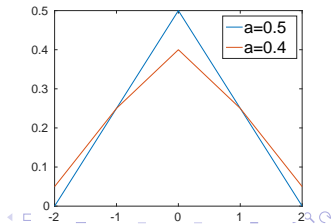
Mask parameters

- From the previous slide,

$$\begin{cases} a + 2c = 2b \\ a + 2b + 2c = 1 \end{cases} \Rightarrow 4b = 1 \Rightarrow b = \frac{1}{4}$$

- Substitute $b = \frac{1}{4}$ back to the equations, we have

$$\begin{cases} b = \frac{1}{4} \\ c = \frac{1}{4} - \frac{a}{2} \end{cases}$$

- $a = 0.4 \approx$ Gaussian, $a = 0.5 \approx$ triangular

Laplacian pyramids

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

$$L_1 = g_1 - EXPAND[g_2]$$
$$L_2 = g_2 - EXPAND[g_3]$$
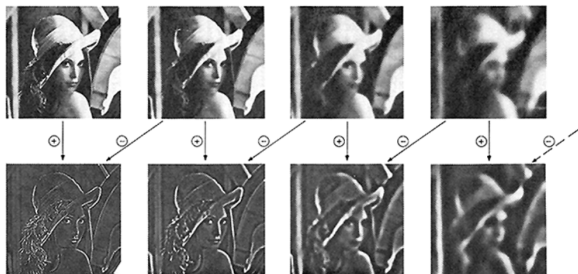$$L_3 = g_3 - EXPAND[g_4]$$

Image encoding

- Compute Gaussian pyramid

$$g_1, g_2, g_3, g_4$$

Image encoding

- Compute Gaussian pyramid

$$g_1, g_2, g_3, g_4$$

- Compute Laplacian pyramid

$$L_1 = g_1 - EXPAND[g_2]$$
$$L_2 = g_2 - EXPAND[g_3]$$
$$L_3 = g_3 - EXPAND[g_4]$$
$$L_4 = g_4$$

Image encoding

- Compute Gaussian pyramid

$$g_1, g_2, g_3, g_4$$

- Compute Laplacian pyramid

$$L_1 = g_1 - EXPAND[g_2]$$
$$L_2 = g_2 - EXPAND[g_3]$$
$$L_3 = g_3 - EXPAND[g_4]$$
$$L_4 = g_4$$

- Quantize and code Laplacian pyramid (e.g., by Huffmann coding)

Image encoding

Image decoding
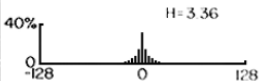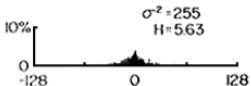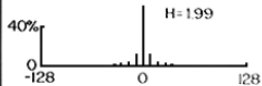
- Decode Laplacian pyramid

$$L_1, L_2, L_3, L_4$$

Image decoding

- Decode Laplacian pyramid

$$L_1, L_2, L_3, L_4$$

- Compute Gaussian pyramid from Laplacian pyramid

$$g_4 = L_4$$
$$g_3 = L_3 + EXPAND[g_4]$$
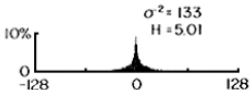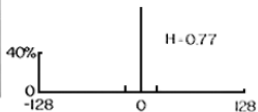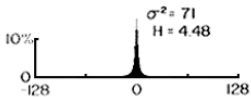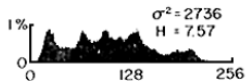$$g_2 = L_2 + EXPAND[g_3]$$
$$g_1 = L_1 + EXPAND[g_2]$$

Image decoding

- Decode Laplacian pyramid

$$L_1, L_2, L_3, L_4$$

- Compute Gaussian pyramid from Laplacian pyramid

$$g_4 = L_4$$
$$g_3 = L_3 + EXPAND[g_4]$$
$$g_2 = L_2 + EXPAND[g_3]$$
$$g_1 = L_1 + EXPAND[g_2]$$

- g_1 is the reconstructed image
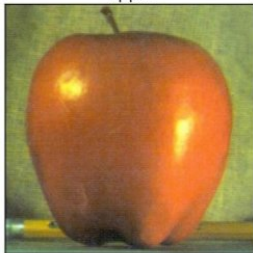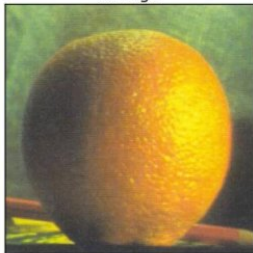
Compression result

(a)

(b)

(c)

(d)

Fig 8. Examples of image data compression using the Laplacian Pyramid code. (a) and (c) give the original "Lady" and "Walter" images, while (b) and (d) give their encoded versions of the data rates are 1.58 and 0.73 bits/pixel for "Lady" and "Walter," respectively. The corresponding mean square errors were 0.88 percent and 0.43 percent, respectively.

Image compositing

HW3

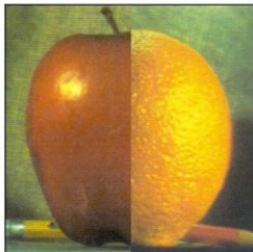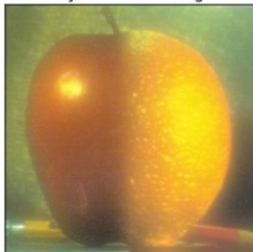## Algorithm for image composting

- Generate Laplacian pyramid L_o of the orange image

Image pyramids

HW3

Algorithm for image composting

- Generate Laplacian pyramid L_o of the orange image
- Generate Laplacian pyramid L_a of the apple image

HW3

Algorithm for image compositing

- Generate Laplacian pyramid L_o of the orange image
- Generate Laplacian pyramid L_a of the apple image
- Generate Laplacian pyramid L_c by
  - Copy left half of nodes at each level from apple pyramid
  - Copy right half of nodes at each level from orange pyramid

HW3

Algorithm for image compositing

- Generate Laplacian pyramid L_o of the orange image
- Generate Laplacian pyramid L_a of the apple image
- Generate Laplacian pyramid L_c by
  - Copy left half of nodes at each level from apple pyramid
  - Copy right half of nodes at each level from orange pyramid
- Reconstruct combined image from pyramid L_c