EL5123 --- Image Processing

Final Term Review

Yao Wang
Polytechnic Institute of NYU, Brooklyn, NY 11201

Topics Covered Before Midterm

- Image representation
- Color representation
- Quantization
- Contrast enhancement
- Spatial Filtering: noise removal, sharpening, edge detection
- Frequency domain representations
  - FT, DTFT, DFT
  - Implementation of linear filtering using DTFT and DFT

Topics Covered After Midterm

- Non-linear filtering: median, morphological filtering
- Image sampling, interpolation and resizing
- Image compression
  - Lossless coding: entropy bound, Huffman coding, runlength coding for bilevel images
  - Transform coding: unitary transform, quantization, runlength coding of coefficients, JPEG
  - Wavelet transform and JPEG2K, Scalability
- Geometric transformation
- Image Restoration

Non-Linear Filtering

- Convolution is a linear operation
  - g1=f1*h, g2=f2*h
  - (a1* f1+a2* f2)*h=a1* g1+a2*g2
- Linear filtering can be analyzed in frequency domain easily
- Non-linear filtering
  - Median
  - Rank-order filtering
  - Morphological filtering

Median Filter

- Problem with averaging or weighted averaging filter
  - Blur edges and details in an image
  - Not effective for impulse noise (Salt-and-pepper)
- Median filter:
  - Taking the median value instead of the average or weighted average of pixels in the window
    - Sort all the pixels in an increasing order, take the middle one
  - The window shape does not need to be a square
  - Special shapes can preserve line structures
- Median filter is a NON-LINEAR operation
- Generalization of median filtering
  - Rank-order filtering: taking the k-th largest value

Morphological Filtering

- Binary image
  - dilation, erosion, closing, opening
  - can be interpreted as set operation
  - More sophisticated operations can extract image features (skeleton, edges, etc.)
- Gray scale image
  - Dilation, erosion, closing, opening
  - Proofs of properties of the morphological filters not required.

Binary Dilation

- Dilation of set F with a structuring element H is represented by $F \oplus H$

$$F \oplus H = \{x : (\hat{H})_x \cap F \neq \Phi\}$$

where Φ represent the empty set.

- $G = F \oplus H$ is composed of all the points that when Ĥ shifts its origin to these points, at least one point of Ĥ is included in F.
- If the origin of H takes value "1", dilation expands the original image $F \subset F \oplus H$

Binary Erosion

- Erosion of set F with a structuring element H is represented by $F \ominus H$, and is defined as, $F \ominus H = \{x : (H)_x \subset F\}$

- $G = F \ominus H$ is composed of points that when H is translated to these points, every point of H is contained in F.
- If the origin of H takes value of "1", erosion shrinks the original image $F \ominus H \subset F$

Closing and Opening

- Closing

$$F \bullet H = (F \oplus H)\Theta H$$

  – Smooth the contour of an image
  – Fill small gaps and holes

- Opening

$$F \circ H = (F \Theta H) \oplus H$$

  – Smooth the contour of an image
  – Eliminate false touching, thin ridges and branches.

Morphological Processing for Grayscale Image

- Dilation $(f \oplus h)(x,y) = \max\{f(x-s, y-t) + h(s,t); (s,t) \in D_h\}$

- Erosion $(f \Theta h)(x,y) = \min\{f(x+s, y+t) - h(s,t); (s,t) \in D_h\}$

- Opening $f \circ h = (f \Theta h) \oplus h$

- Closing $f \bullet h = (f \oplus h) \Theta h$

- Can be thought of as non-linear filtering: replacing weighted sum by min/max operations

Sampling and Interpolation

- What determines necessary sampling frequency?
- Why is pre-filtering necessary?
- How to reconstruct a continuous image from samples
- Frequency domain interpretation of sampling and interpolation

Frequency Domain Interpretation of Sampling

- Sampling is equivalent to multiplication of the original signal with a sampling pulse sequence.

$$f_s(x,y) = f(x,y)\,p(x,y)$$

$$where \quad p(x,y) = \sum_{m,n} \delta(x - m\Delta x, y - n\Delta y)$$

- In frequency domain

$$F_s(u,v) = F(u,v) * P(u,v)$$

$$P(u,v) = \frac{1}{\Delta x \Delta y} \sum_{m,n} \delta(u - m f_{s,x}, v - n f_{s,y}) \quad \Rightarrow \quad F(u,v) = \frac{1}{\Delta x \Delta y} \sum_{m,n} F(u - m f_{s,x}, v - n f_{s,y})$$

$$where \quad f_{s,x} = \frac{1}{\Delta x},\ f_{s,y} = \frac{1}{\Delta y}$$

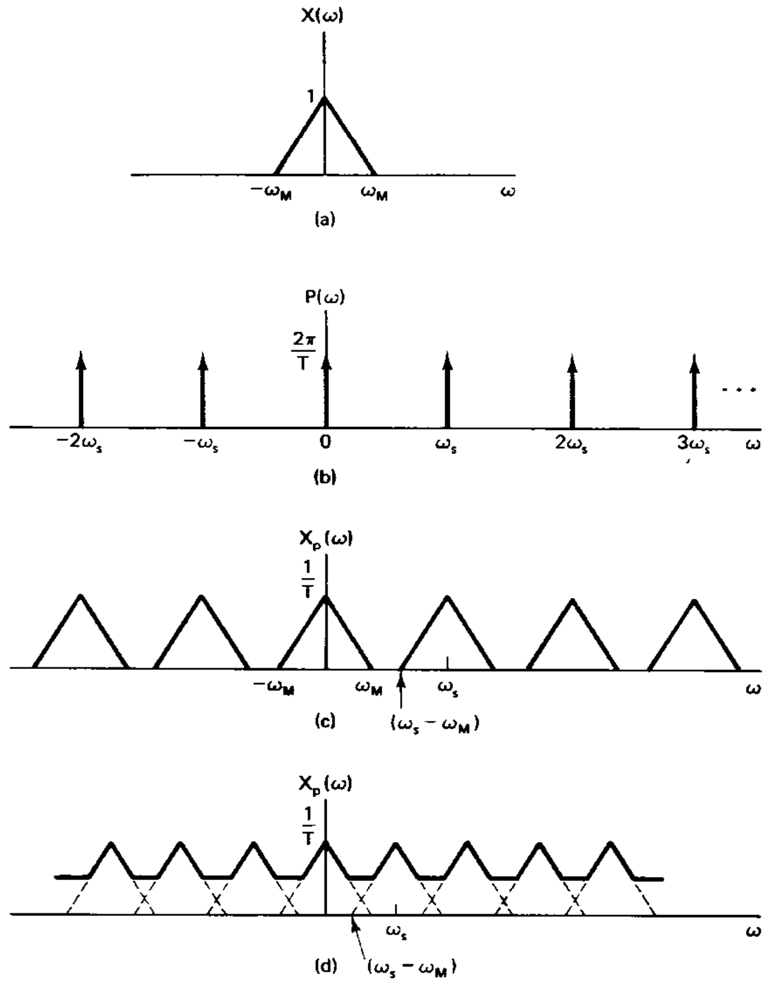
Frequency Domain Interpretation of Sampling in 1D

Original signal

Sampling impulse train

Sampled signal $f_s > 2f_m$

Sampled signal $f_s < 2f_m$ (Aliasing effect)



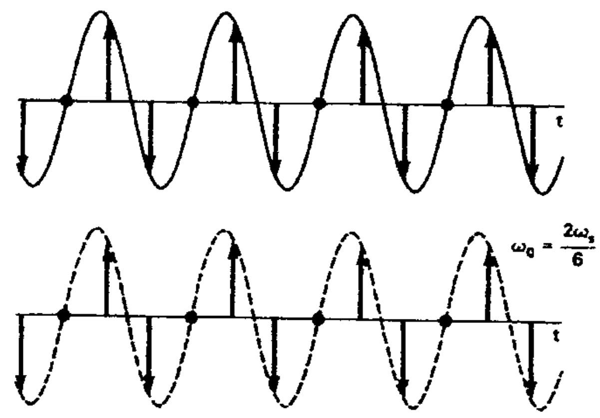
The spectrum of the sampled signal includes the original spectrum and its aliases (copies) shifted to *k f_s , k=+/- 1,2,3,…*
The reconstructed signal from samples has the frequency components upto *f_s/2.*

When $f_s < 2f_m$, aliasing occur.

Sampling of 1D Sinusoid Signals

Sampling above
Nyquist rate
$\omega_s = 3\omega_m > \omega_{s0}$

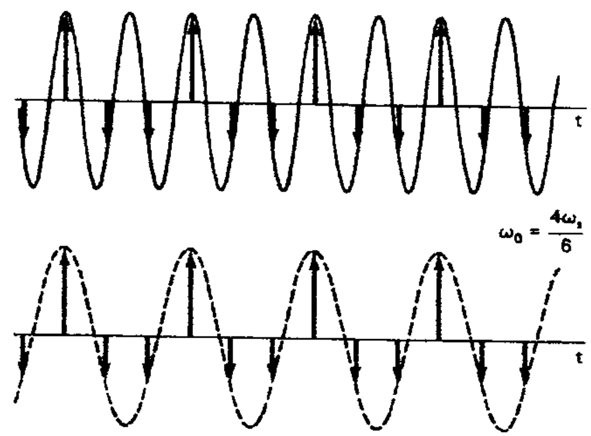
Reconstructed
=original



(b)

Sampling under
Nyquist rate
$\omega_s = 1.5\omega_m < \omega_{s0}$

Reconstructed
!= original

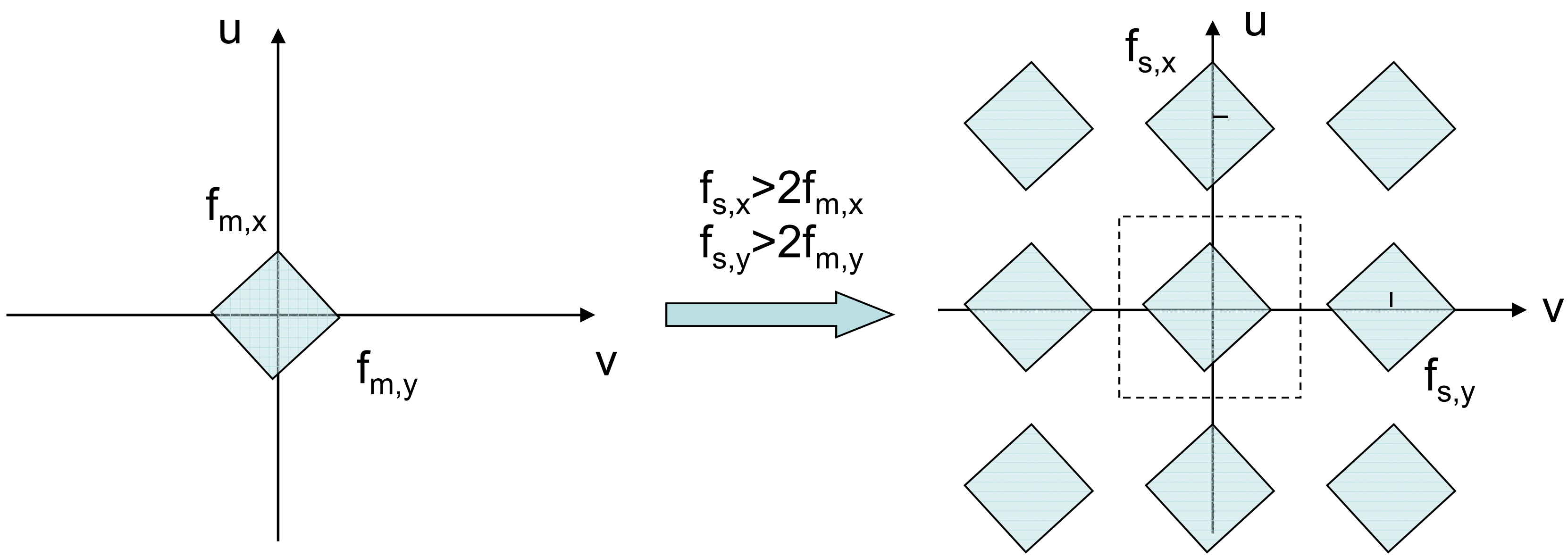


(c)

Aliasing: The reconstructed sinusoid has a lower frequency than the original!

Frequency Domain Interpretation of Sampling in 2D

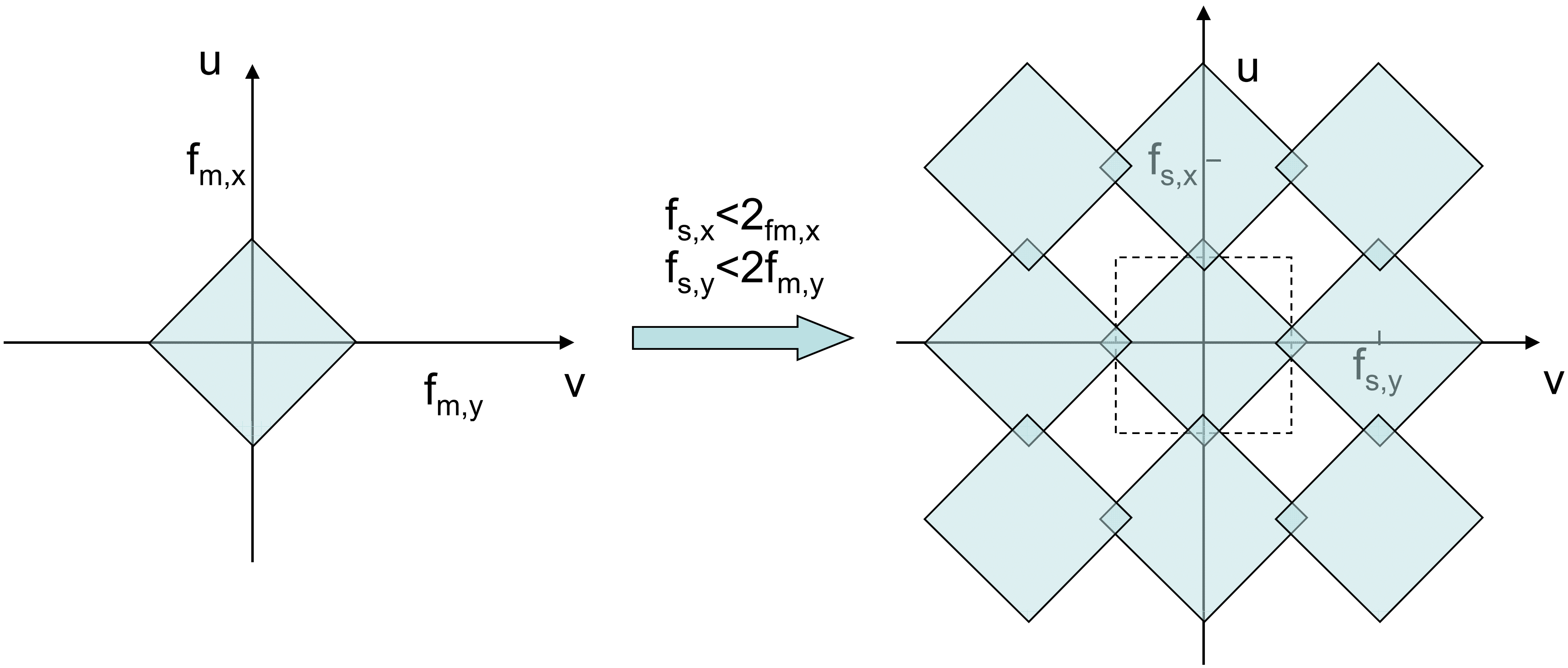
- The sampled signal contains replicas of the original spectrum shifted by multiples of sampling frequencies.



Original spectrum F(u,v)

Sampled spectrum $F_s(u,v)$

Illustration of Aliasing Phenomenon



Original spectrum F(u,v)

Sampled spectrum $F_s(u,v)$

Nyquist Sampling and Reconstruction Theorem

- In order to avoid aliasing, the sampling frequency $f_{s,x}$, $f_{s,y}$ must be at least twice of the highest frequency of the signal, known as *Nyquist sampling rate*.
- A band-limited image with highest frequencies at $f_{m,x}$, $f_{m,y}$ can be reconstructed perfectly from its samples, provided that the sampling frequencies satisfy: $f_{s,x} > 2f_{m,x}$, $f_{s,y} > 2f_{m,y}$
- The reconstruction can be accomplished by the ideal low-pass filter with cutoff frequency at $f_{c,x} = f_{s,x}/2$, $f_{c,y} = f_{s,y}/2$, with magnitude $\Delta x \Delta y$.

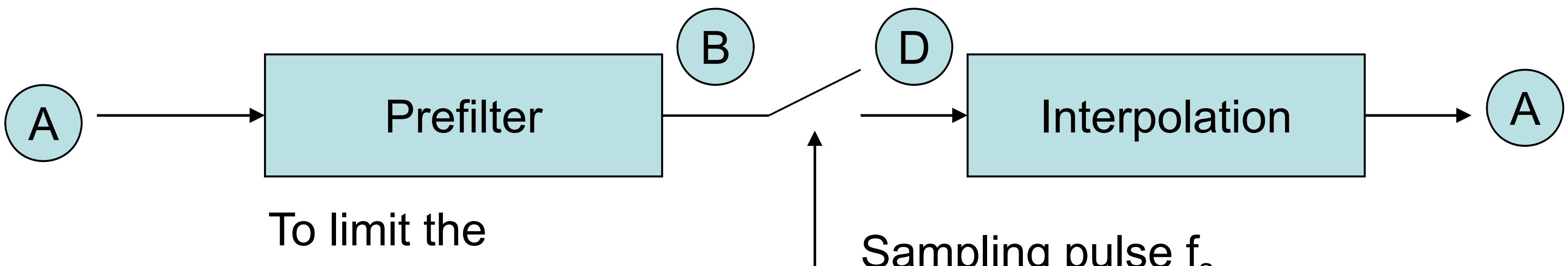
$$H(u,v) = \begin{cases} \Delta x \Delta y & |u| \le \frac{f_{s,x}}{2}, |v| \le \frac{f_{s,y}}{2} \\ 0 & otherwise \end{cases} \Leftrightarrow \quad h(x,y) = \frac{\sin \pi f_{s,x} x}{\pi f_{s,x} x} \cdot \frac{\sin \pi f_{s,y} y}{\pi f_{s,y} y}$$

- The interpolated image

$$\hat{f}(x,y) = \sum_m \sum_n f_s(m,n) \frac{\sin \pi f_{s,x}(x - m\Delta x)}{\pi f_{s,x}(x - m\Delta x)} \frac{\sin \pi f_{s,y}(y - m\Delta y)}{\pi f_{s,y}(y - m\Delta y)}$$

Applying Nyquist Theorem

- Two issues
  - The signals are not bandlimited.
  - The sinc filter is not realizable.
- A general paradigm



To limit the bandwidth to <=fs/2

Sampling pulse f_s

- Averaging
- Weighted average
- Truncated sinc

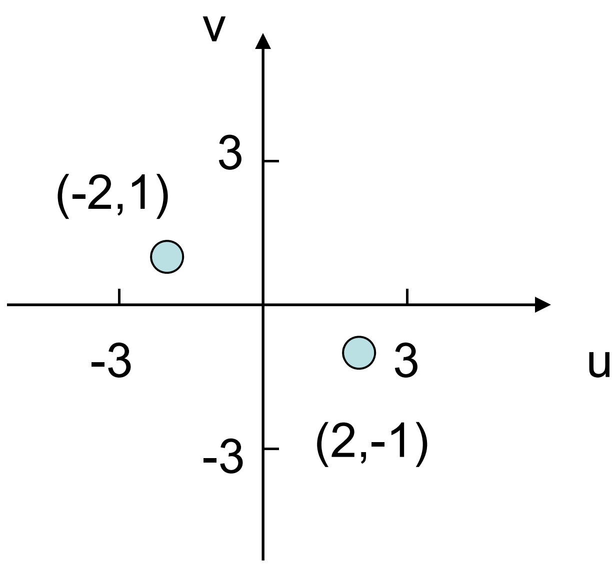
- Sample and hold
- Bilinear interpolation
- Bicubic
- Truncated sinc

Sampling a Sinusoidal Signal

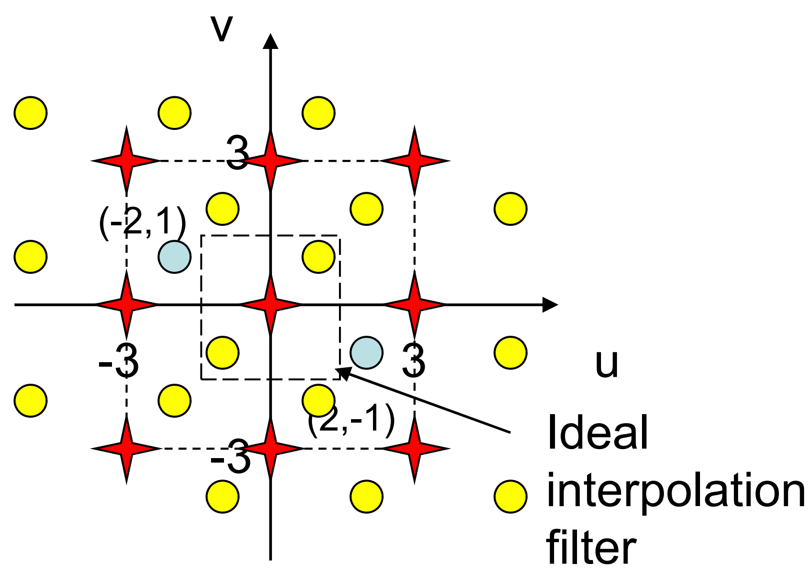
$$f(x,y)=\cos(4\pi x-2\pi y) \quad \Leftrightarrow \quad F(u,v)=\frac{1}{2}\left[\delta(u-2,v+1)+\delta(u+2,v-1)\right]$$

Sampled at Δx=Δy=1/3 $f_{s,x}=f_{s,y}=3$

Original Spectrum



Sampled Spectrum



Original pulse

Replicated pulse

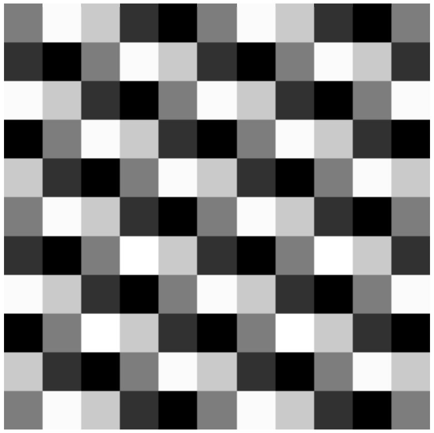
$$\hat{f}(x,y)=\cos(2\pi x+2\pi y)$$

Replication center

What if we use non-ideal interpolation filter?

Sampling in 2D: Sampling a 2D Sinusoidal Pattern

f(x,y)=sin(2*π*(3x+y))
Sampling: dx=0.01,dy=0.01
Satisfying Nyquist rate
$f_{x,max}=3$, $f_{y,max}=1$
$f_{s,x}=100>6$, $f_{s,y}=100>2$



f(x,y)=sin(2*π*(3x+y))
Sampling: dx=0.2,dy=0.2
(Displayed with pixel replication)
Sampling at a rate lower than Nyquist rate

Comparison of Different Interpolation Filters

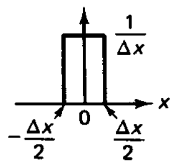
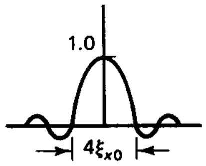
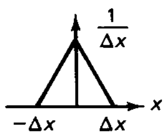
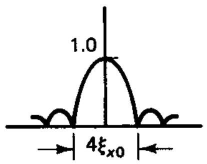
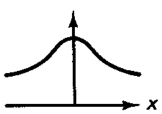
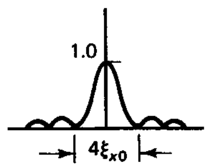
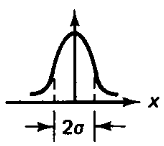
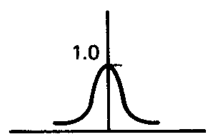
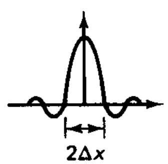
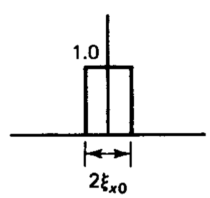
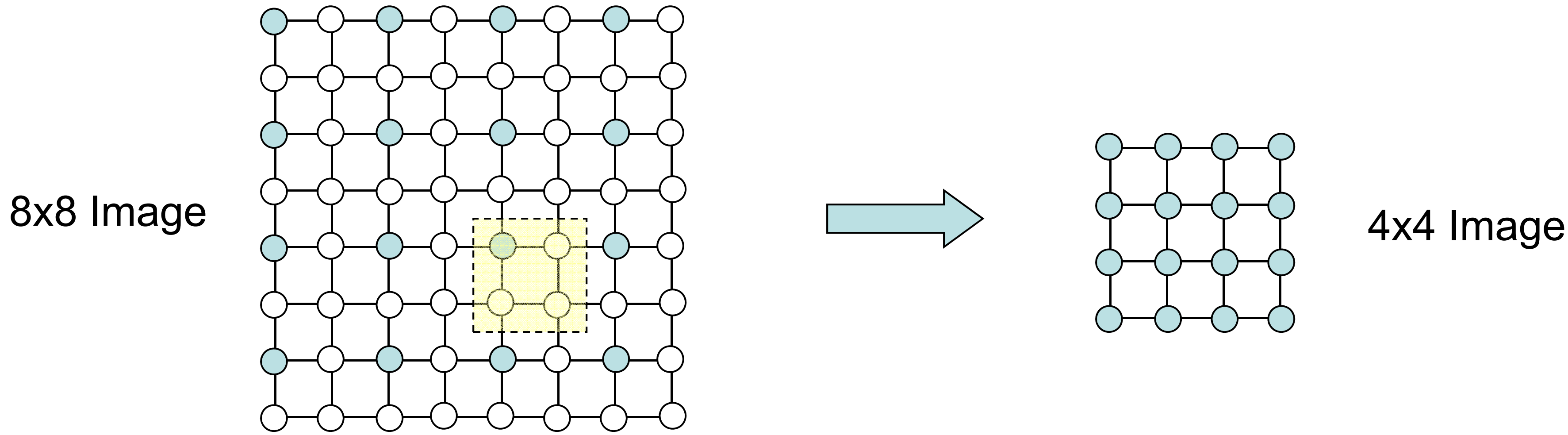
One-dimensional interpolation function	Diagram	Definition $p(x)$	Two-dimensional interpolation function $p_d(x, y) = p(x)p(y)$	Frequency response $P_d(\xi_1, \xi_2)$	$P_d(\xi_1, 0)$
Rectangle (zero-order hold) ZOH $p_o(x)$		$\frac{1}{\Delta x}\,\mathrm{rect}\left(\frac{x}{\Delta x}\right)$	$p_o(x)p_o(y)$	$\mathrm{sinc}\left(\frac{\xi_1}{2\xi_{x0}}\right)\mathrm{sinc}\left(\frac{\xi_2}{2\xi_{y0}}\right)$	
Triangle (first-order hold) FOH $p_1(x)$		$\frac{1}{\Delta x}\,\mathrm{tri}\left(\frac{x}{\Delta x}\right)$ $p_o(x) \circledast p_o(x)$	$p_1(x)p_1(y)$	$\left[\mathrm{sinc}\left(\frac{\xi_1}{2\xi_{x0}}\right)\mathrm{sinc}\left(\frac{\xi_2}{2\xi_{y0}}\right)\right]^2$	
nth-order hold $n = 2$, quadratic $n = 3$, cubic splines $p_n(x)$		$p_o(x) \circledast \cdots \circledast p_o(x)$ n convolutions	$p_n(x)p_n(y)$	$\left[\mathrm{sinc}\left(\frac{\xi_1}{\xi_{x0}}\right)\mathrm{sinc}\left(\frac{\xi_2}{\xi_{y0}}\right)\right]^{n+1}$	
Gaussian $p_g(x)$		$\frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{x^2}{2\sigma^2}\right]$	$\frac{1}{2\pi\sigma^2}\exp\left[-\frac{(x^2+y^2)}{2\sigma^2}\right]$	$\exp\left[-2\pi^2\sigma^2(\xi_1^2+\xi_2^2)\right]$	
Sinc		$\frac{1}{\Delta x}\,\mathrm{sinc}\left(\frac{x}{\Delta x}\right)$	$\frac{1}{\Delta x \Delta y}\,\mathrm{sinc}\left(\frac{x}{\Delta x}\right)\mathrm{sinc}\left(\frac{x}{\Delta y}\right)$	$\mathrm{rect}\left(\frac{\xi_1}{2\xi_{x0}}\right)\mathrm{rect}\left(\frac{\xi_2}{2\xi_{y0}}\right)$	

Image Resizing

- Image resizing:
  - Enlarge or reduce the image size (number of pixels)
  - Equivalent to
    - First reconstruct the continuous image from samples
    - Then Resample the image at a different sampling rate
  - Can be done w/o reconstruct the continuous image explicitly
- Image down-sampling (resample at a lower rate)
  - Spatial domain view
  - Frequency domain view: need for prefilter
- Image up-sampling (resample at a higher rate)
  - Spatial domain view
  - Different interpolation filters
    - Nearest neighbor, Bilinear, Bicubic

Down Sampling by a Factor of Two



- Without Pre-filtering (simple approach)

$$f_d(m,n) = f(2m,2n)$$

- Averaging Filter

$$f_d(m,n) = [f(2m,2n) + f(2m,2n+1) + f(2m+1,2n) + f(2m+1,2n+1)]/4$$

Problem of Simple Approach

- Aliasing if the new sampling rate is below the Nyquist sample rate = 2 * highest frequency in the signal
- We need to prefilter the signal before down-sampling
- Ideally the prefilter should be a low-pass filter with a cut-off frequency half of the new sampling rate.
  - In digital frequency of the original sampled image, the cutoff frequency is ¼.
- In practice, we may use simple averaging filter

Example: Image Down-Sample

Without prefiltering

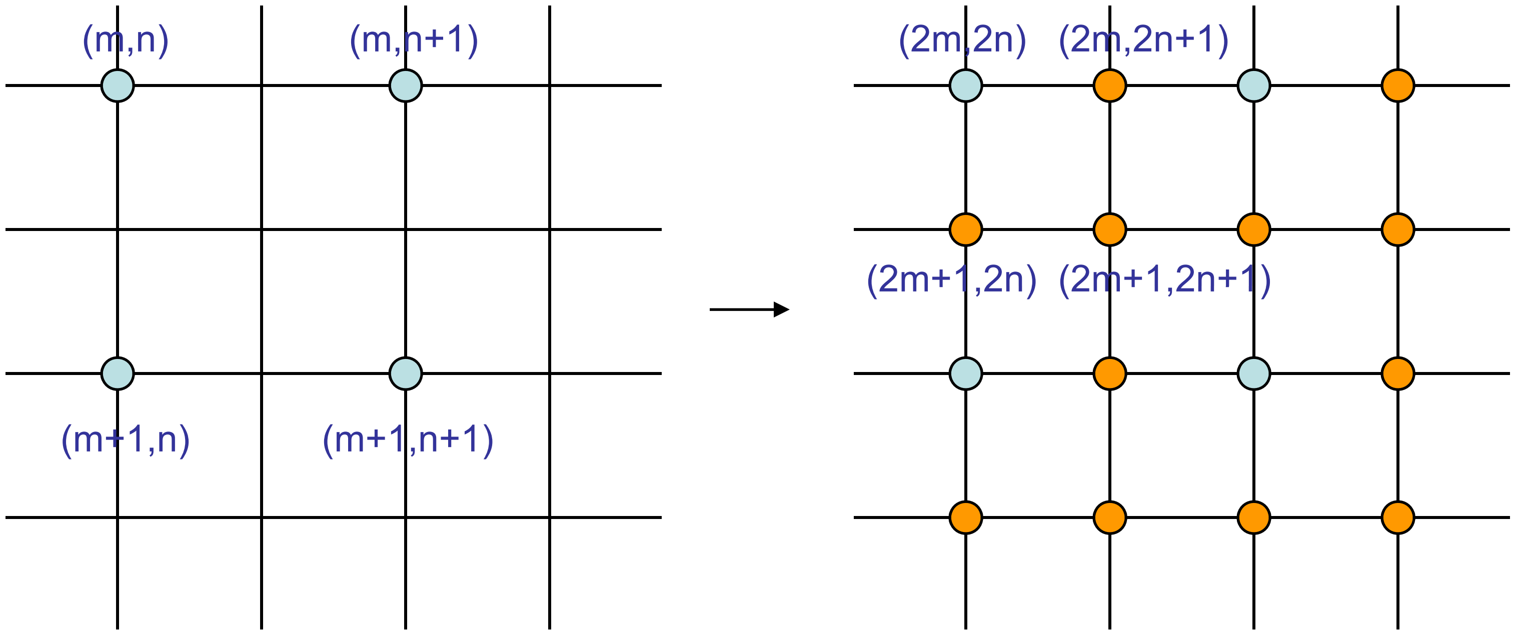
With prefiltering

Image Up-Sampling

- Produce a larger image from a smaller one
  - Eg. 512x512 -> 1024x1024
  - More generally we may up-sample by an arbitrary factor L
- Questions:
  - How should we generate a larger image?
  - Does the enlarged image carry more information?
- Connection with Interpolation of a continuous image from discrete image
  - First interpolate to continuous image, then sampling at a higher sampling rate, Lfs
  - Can be realized with the same interpolation filter, but only evaluate at $x=m\Delta x'$, $y=n\Delta y'$, $\Delta x'=\Delta x/L$, $\Delta y'=\Delta y/L$
  - Ideally using the sinc filter!

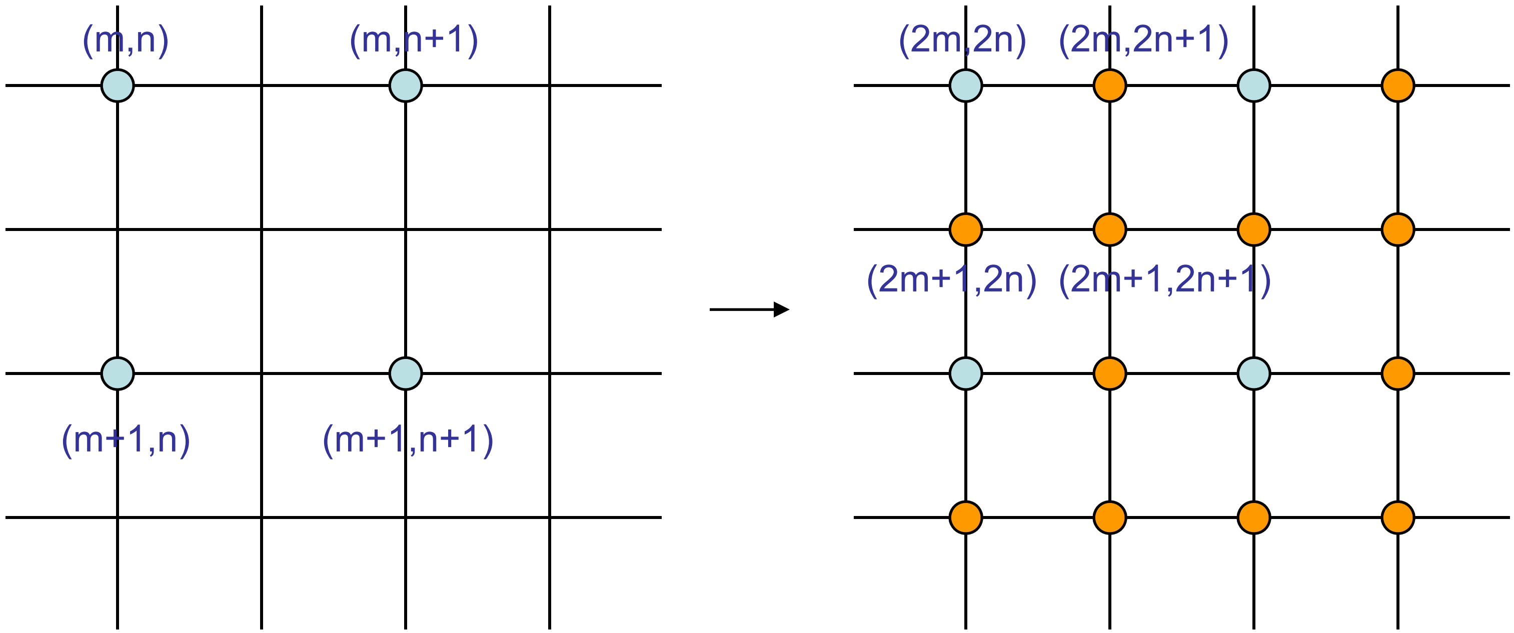
$$\hat{f}(x,y)=\sum_m\sum_n f_s(m,n)\frac{\sin \pi f_{s,x}(x-m\Delta x)}{\pi f_{s,x}(x-m\Delta x)}\frac{\sin \pi f_{s,y}(y-m\Delta y)}{\pi f_{s,y}(y-m\Delta y)}$$

Example: Factor of 2 Up-Sampling



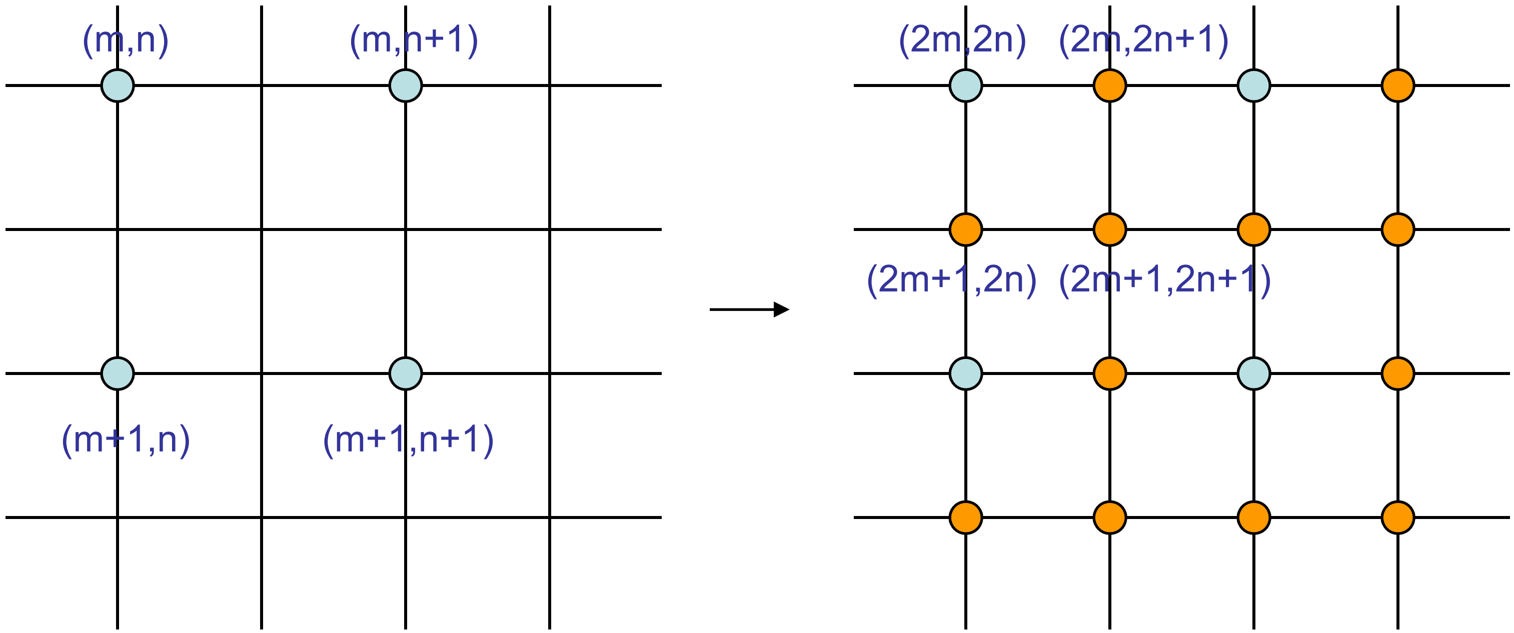
Green samples are retained in the interpolated image;
Orange samples are estimated from surrounding green samples.

Pixel Replication (0-th order)



Nearest Neighbor:
O[2m,2n]=I[m,n]
O[2m,2n+1]= I[m,n]
O[2m+1,2n]= I[m,n]
O[2m+1,2n+1]= I[m,n]

Bilinear Interpolation (1st order)



Bilinear Interpolation:

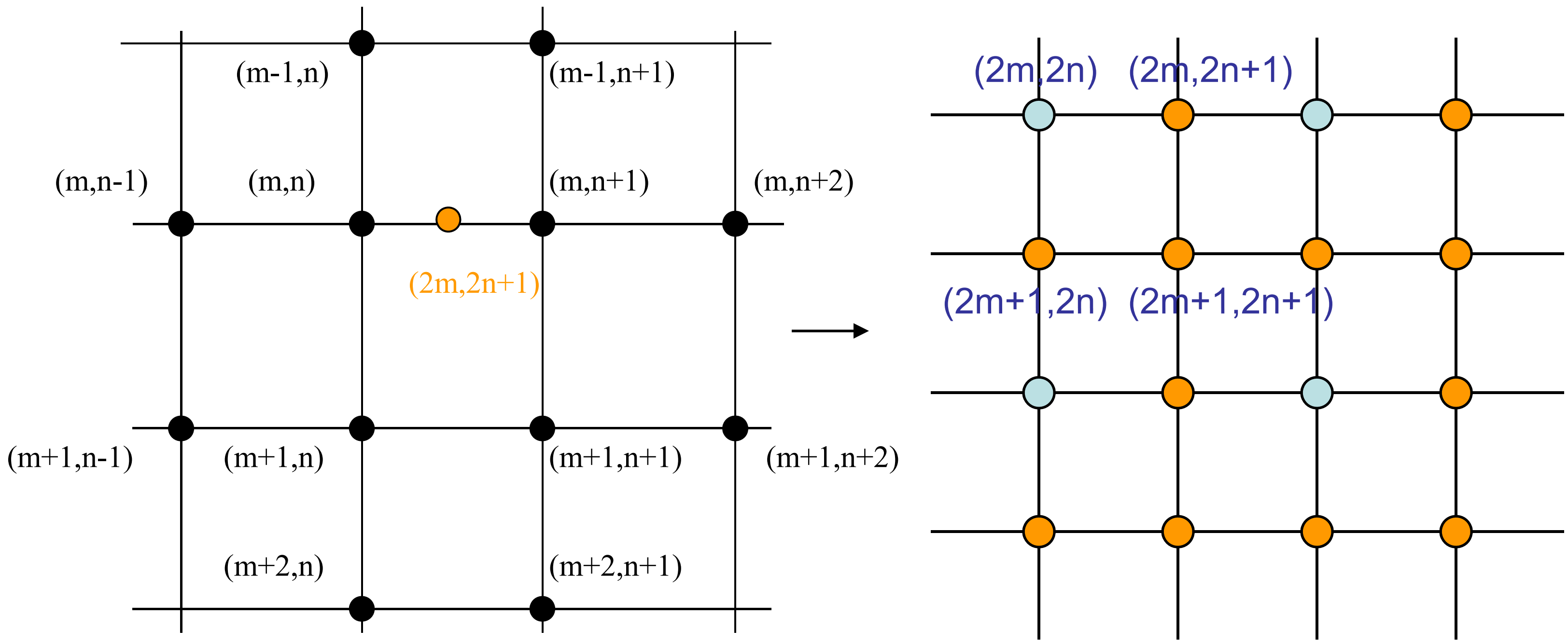
O[2m,2n]=I[m,n]

O[2m,2n+1]=(I[m,n]+I[m,n+1])/2

O[2m+1,2n]=(I[m,n]+I[m+1,n])/2

O[2m+1,2n+1]=(I[m,n]+I[m,n+1]+I[m+1,n]+I[m+1,n+1])/4

Bicubic Interpolation (3rd order)

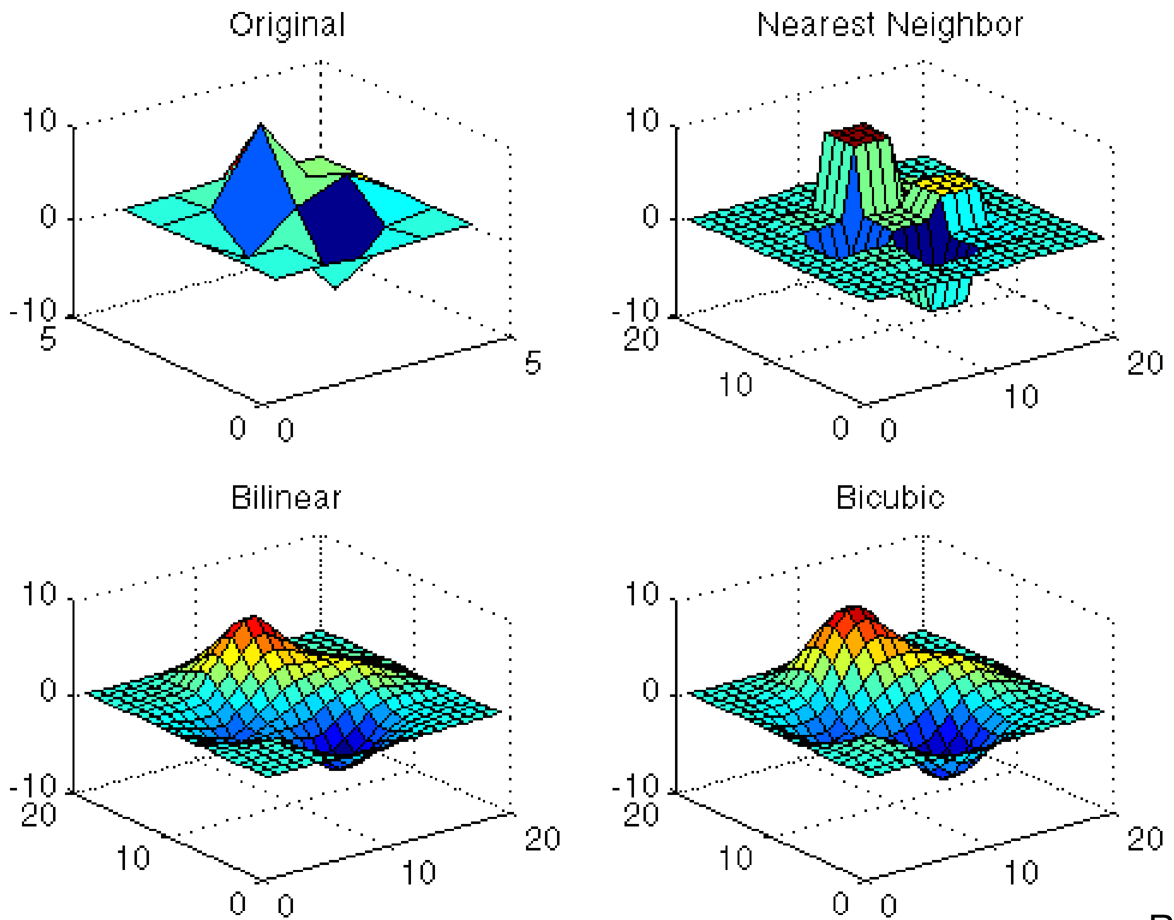


Bicubic interpolation in Horizontal direction

F[2m,2n]=I[m,n]
F[2m,2n+1]= -(1/8)I[m,n-1]+(5/8)I[m,n]+(5/8)I[m,n+1]-(1/8)I(m,n+2)

Same operation then repeats in vertical direction

Comparison of Interpolation Methods



Resize_peak.m

Up-Sampled from w/o Prefiltering

Original

Nearest neighbor

Bilinear

Bicubic

Up-Sampled from with Prefiltering

Original

Nearest neighbor

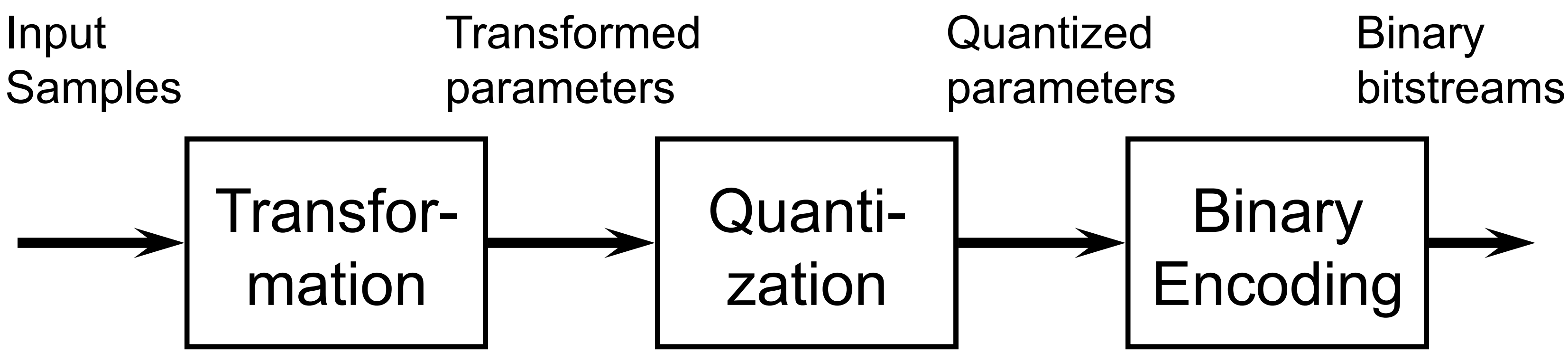
Bilinear

Bicubic

Image Compression

- Three major steps
  - Transformation, quantization, binary encoding
- Binary encoding
- Quantization
- Transformation:
  - Runlength
  - Linear transform
  - Prediction
- JPEG
- JPEG2000

A Typical Compression System



Prediction
Transforms
Model fitting
……

Scalar Q
Vector Q

Fixed length
Variable length
(Huffman,
arithmetic, LZW)

- Motivation for transformation ---
  To yield a more efficient representation of the original samples.

Binary Encoding

- Binary encoding
  - To represent a finite set of symbols using binary codewords.
- Fixed length coding
  - N symbols represented by $(int)\ log_2(N)$ bits.
- Variable length coding
  - more frequently appearing symbols represented by shorter codewords (Huffman, arithmetic, LZW=zip).
- The minimum number of bits required to represent a source is bounded by its entropy.

Entropy of a Source

- Consider a source of N symbols, r_n, n=1,2,…,N. Suppose the probability of symbol r_n is p_n. The entropy of this source is defined as:

$$H = -\sum_{n=1}^{N} p_n \log_2 p_n \quad (bits)$$

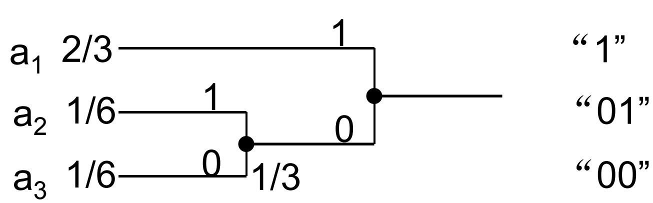
- Shannon Source Coding Theory: For an arbitrary source, a code can be designed so that the average length is bounded by

$$H \le l = \sum p_n l_n \le H + 1$$

- The Shannon theorem only gives the bound but not the actual way of constructing the code to achieve the bound
- Practical coding methods:
  - Huffman
  - LZW
  - Arithmetic coding

Huffman Coding

- Procedure of Huffman coding
  - Step 1: Arrange the symbol probabilities p_n in a decreasing order and consider them as leaf nodes of a tree.
  - Step 2: While there is more than one node:
    - Find the two nodes with the smallest probability and arbitrarily assign 1 and 0 to these two nodes
    - Merge the two nodes to form a new node whose probability is the sum of the two merged nodes.



$$l = \frac{2}{3}\times 1 + \frac{1}{6}\times 2 + \frac{1}{6}\times 2 = \frac{4}{3} = 1.33$$

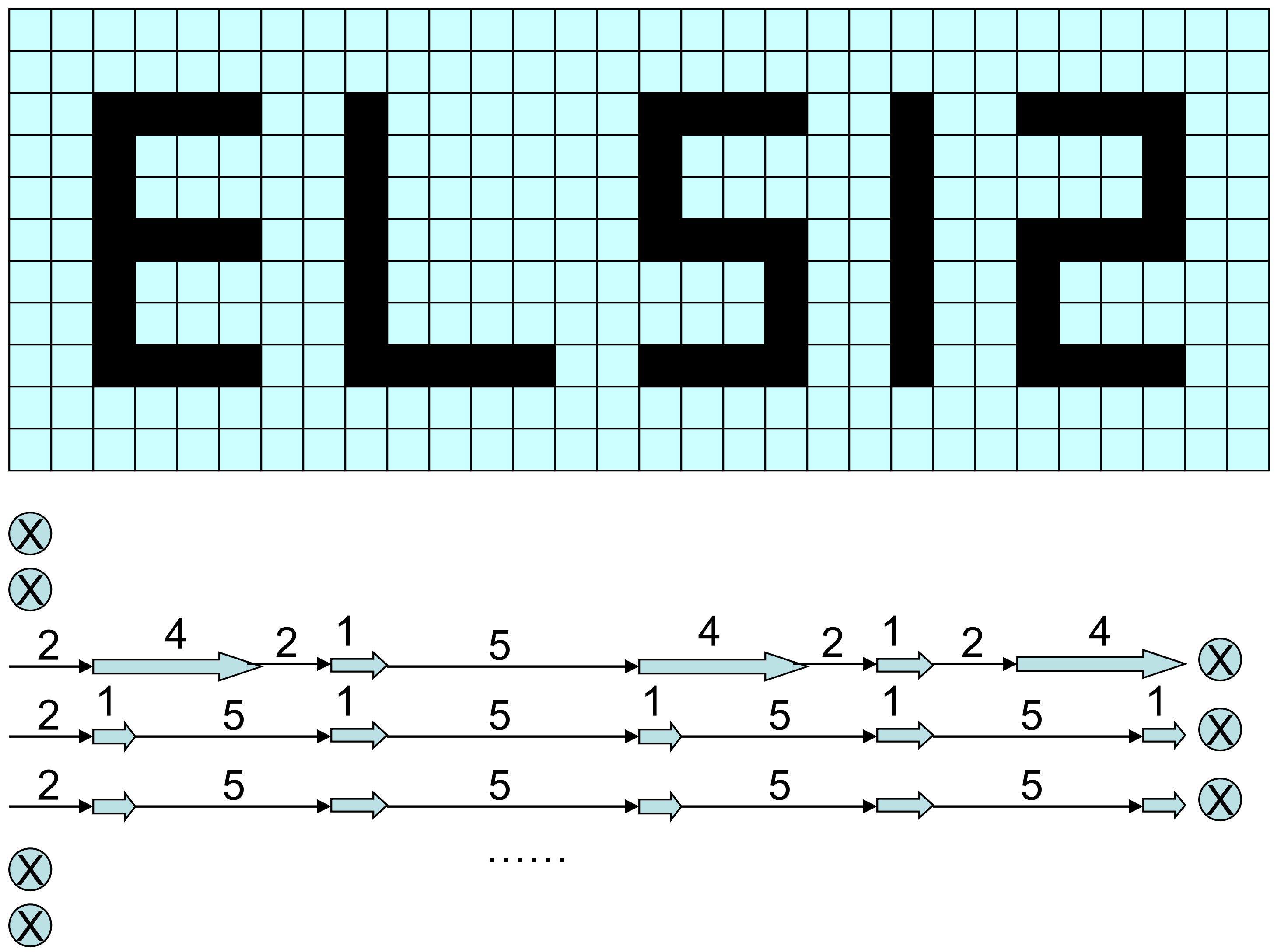
Improvement of Huffman Coding

- Disadvantage of Huffman coding:
  - At least one bit has to be used for each symbol
- Vector Huffman coding
  - Treat each group of M symbols as one entity and give each group a codeword.
  - Bit rate per M symbols bounded by the joint entropy of M symbols: $H_M \le R_M \le H_M + 1$
  - Bit rate per symbol bounded by $H_M/M \le R \le H_M/M + 1/M$
- Conditional Huffman coding:
  - The possible outcomes of a new sample depends on its neighboring samples
  - Described by the conditional probability
  - Build a different Huffman table for each possible neighborhood structure (context)
  - Bounded by the conditional entropy

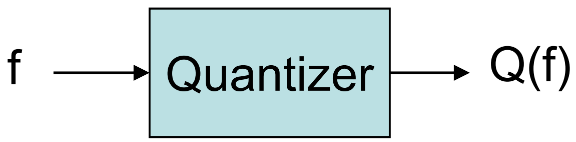
Runlength Coding of Bi-Level Images

- 1D Runlength Coding:
  - Count length of white and length of black alternatingly
  - Represent the last runlength using “EOL”
  - Code the white and black runlength using different codebook (Huffman Coding)
- 2D Runlength Coding:
  - Use relative address from the last transition in the line above
  - Used in Facsimile Coding (G3,G4)
  - Details of 2D run-length coding in the G3/G4 standards are not required.

Example of 1-D Runlength Coding



Quantization



Decision Levels $\{t_k, k = 1, \ldots, L+1\}$

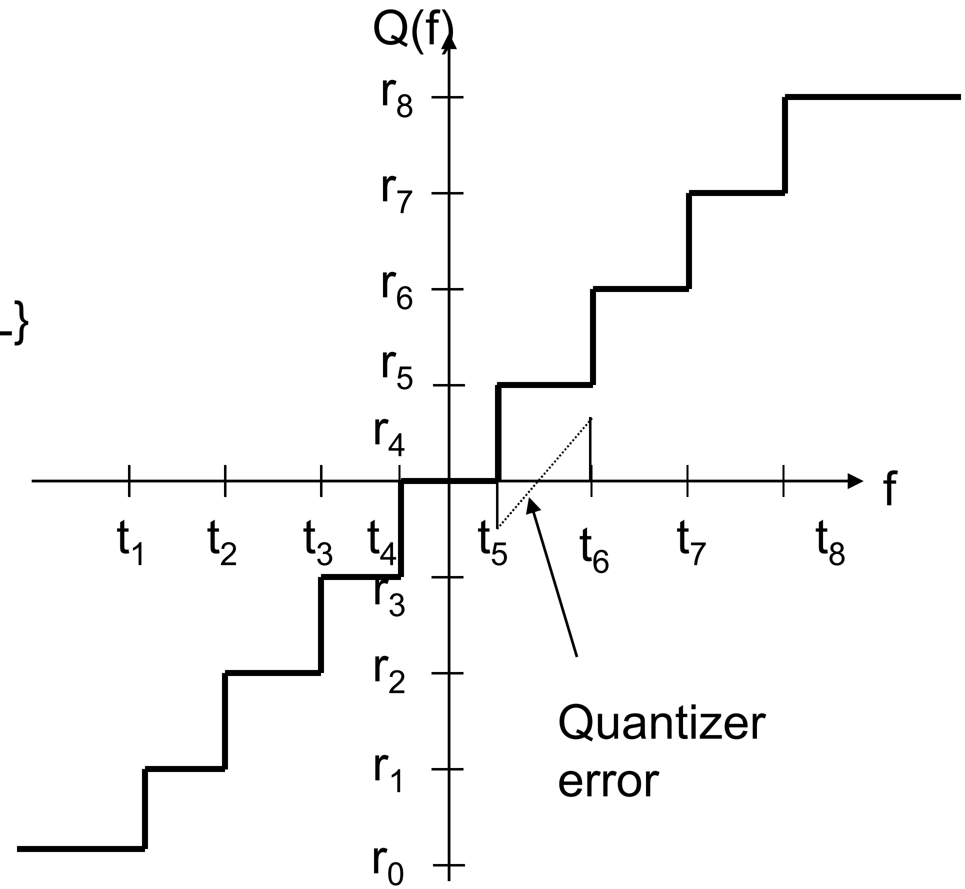
Reconstruction Levels $\{r_k, k = 1, \ldots, L\}$

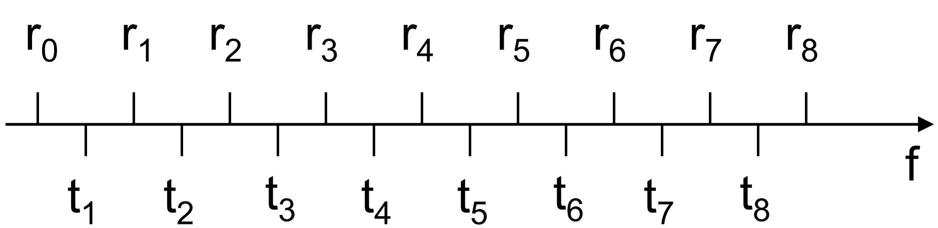
If $f \in [t_k, t_{k+1})$

Then $Q(f) = r_k$

L levels need $R = \lceil \log_2 L \rceil$ bits

$\lceil x \rceil$ returns the smallest integer that is bigger than or equal to x





Uniform Quantization

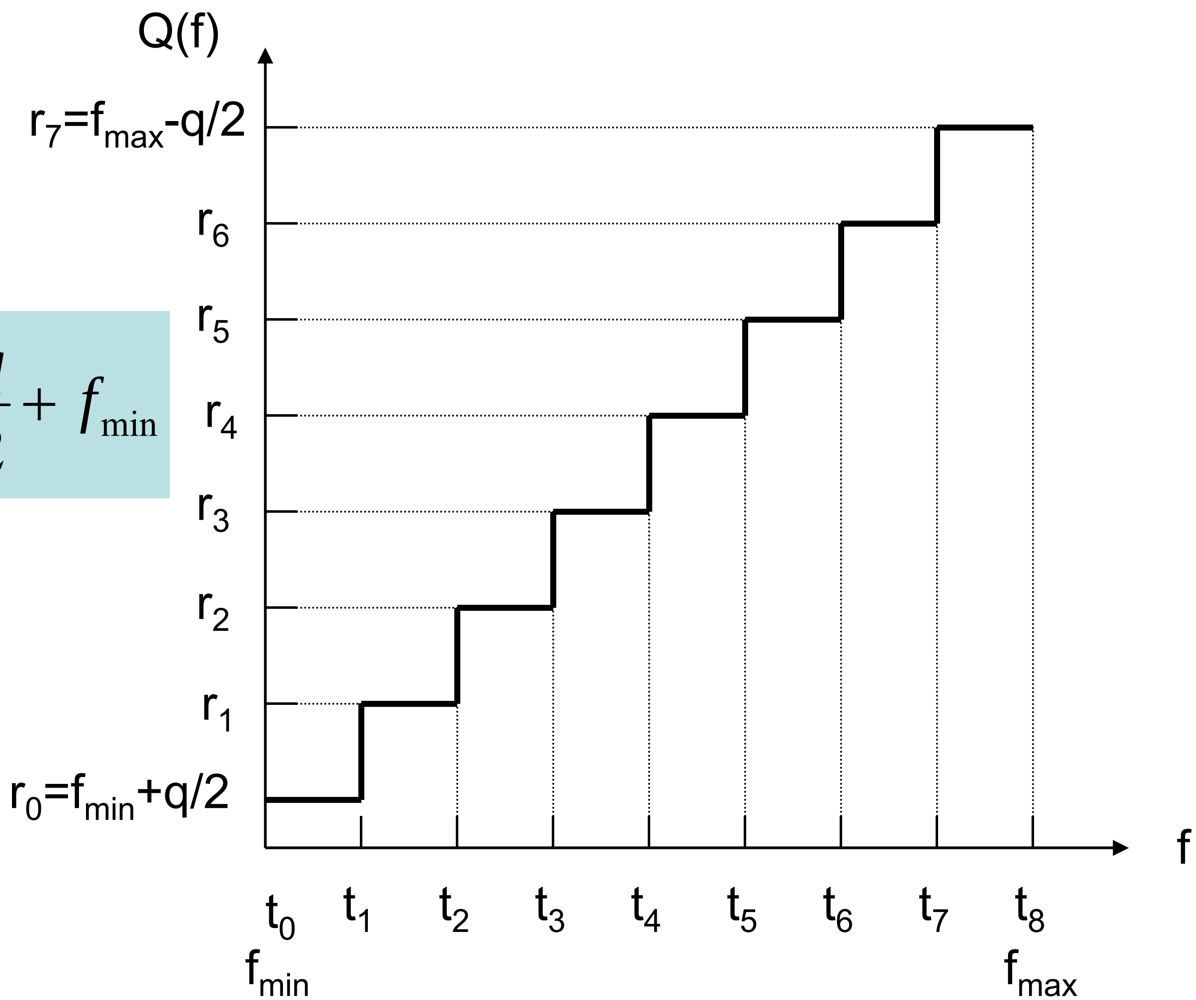
- Equal distances between adjacent decision levels and between adjacent reconstruction levels
  - $t_l - t_{l-1} = r_l - r_{l-1} = q$
- Parameters of Uniform Quantization
  - L: levels ($L = 2^R$)
  - B: dynamic range $B = f_{max} - f_{min}$
  - q: quantization interval (step size)
  - $q = B/L = B2^{-R}$

Uniform Quantization: Functional Representation

stepsize $q=(f_{max}-f_{min})/L$

$$Q(f)=\left\lfloor \frac{f-f_{\min}}{q} \right\rfloor * q + \frac{q}{2} + f_{\min}$$

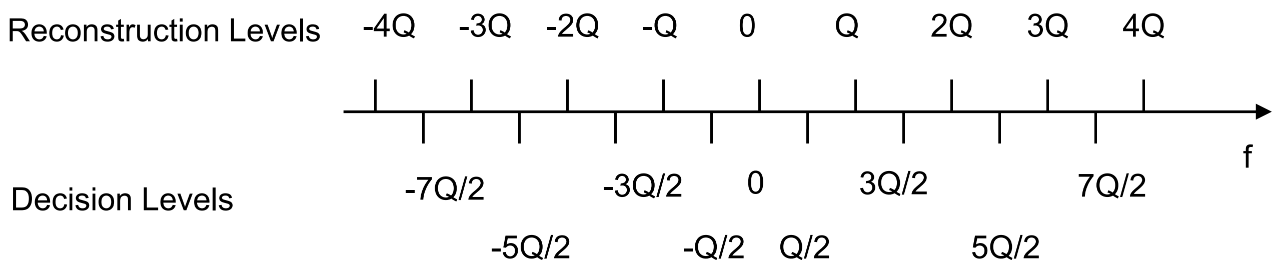
$\lfloor x \rfloor$ returns the biggest integer that is smaller than or equal to x



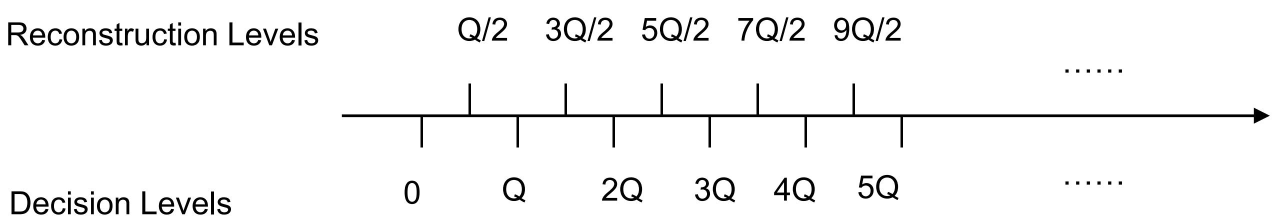
$I(f)=\left\lfloor \frac{f-f_{\min}}{q} \right\rfloor$ is called the reconstruction level index, which indicates which reconstruction level is used for f.

What if fmin, fmax are not known?

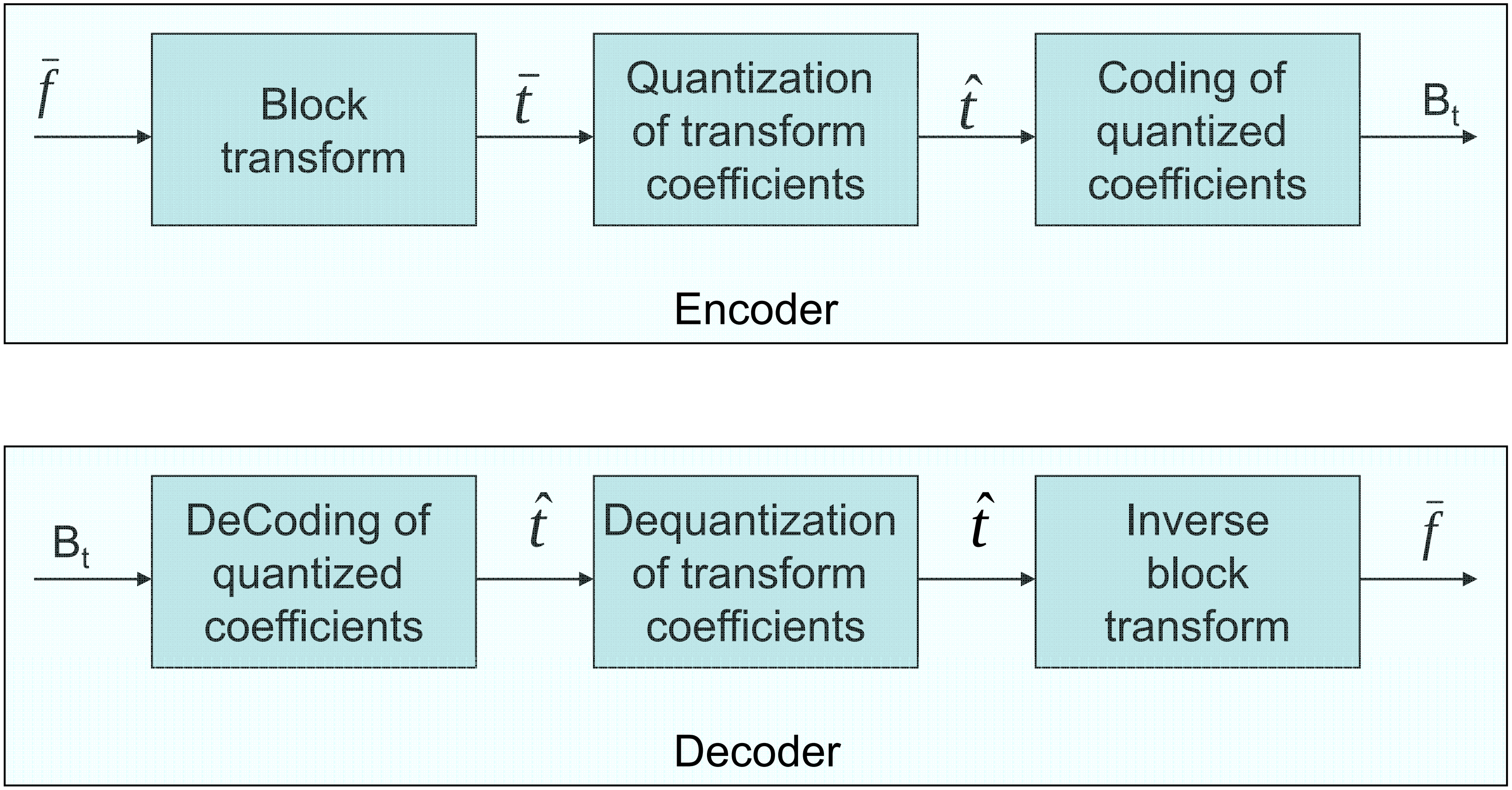
- For sources with zero mean (e.g. transform coefficients, prediction errors)
  With quantizer bins centered around zeros
  Quantize f to the bin: Qindex(f)=sign(f)* (int)[|f|+Q/2)/Q]
  Quantized value: Q(f)= Qindex(f)*Q



- For sources starting at 0 (e.g. original pixel values):
  - Quantize f to the bin: Qindex(f)=(int)[f/Q]
  - Quantized value: Q(f)=Qindex(f)*Q+Q/2



Transform Coding



Predictive Coding

- Motivation
  - The value of a current pixel usually does not change rapidly from those of adjacent pixels. Thus it can be predicted quite accurately from the previous samples.
  - The prediction error will have a non-uniform distribution, centered mainly near zero, and can be specified with less bits than that required for specifying the original sample values, which usually have a uniform distribution.

Linear Predictor

- Let f_0 represent the current pixel, and f_k, k = 1,2,…,K the previous pixels that are used to predict f_0. For example, if f_0=f(m,n), then f_k=f(m-i,n-j) for certain i, j ≥ 0. A linear predictor is

$$\hat{f}_0 = \sum_{k=1}^{K} a_k f_k$$

- a_k are called linear prediction coefficients or simply prediction coefficients.
- The key problem is how to determine a_k so that a certain criterion is satisfied.

LMMSE Predictor (1)

- The designing criterion is to minimize the mean square error (MSE) of the predictor.

$$\sigma_p^2 = E\{|f_0 - \hat{f}_0|^2\} = E\left\{\left|f_0 - \sum_{k=1}^{K} a_k f_k\right|^2\right\}$$

- The optimal a_k should minimize the error

$$\frac{\partial \sigma_p^2}{\partial a_l} = E\left\{\left(f_0 - \sum_{k=1}^{K} a_k f_k\right) f_l\right\} = 0, \quad l = 1,2,\ldots,K.$$

Let $R(k,l)=E\{f_k f_l\}$

$$\sum_{k=1}^{K} a_k R(k,l) = R(0,l), \quad l = 1,2,\ldots,K$$

LMMSE Predictor (2)

$$\sum_{k=1}^{K} a_k R(k,l) = R(0,l), \quad l = 1,2,\ldots,K$$

In matrix format

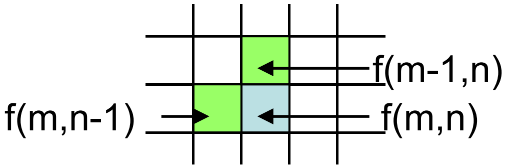
$$\begin{bmatrix} R(1,1) & R(2,1) & \ldots & R(K,1) \\ R(1,2) & R(2,2) & \ldots & R(K,2) \\ \vdots & \vdots & \ddots & \vdots \\ R(1,K) & R(2,K) & \cdots & R(K,K) \end{bmatrix} \begin{bmatrix} a_1 \\ a_1 \\ \vdots \\ a_K \end{bmatrix} = \begin{bmatrix} R(0,1) \\ R(0,2) \\ \vdots \\ R(0,K) \end{bmatrix} \Rightarrow Ra = r \Rightarrow a = R^{-1} r$$

The MSE of this predictor

$$\sigma_p^2 = E\{(f_0 - \hat{f}_0) f_0\} = R(0,0) - \sum_{k=1}^{K} a_k R(k,0) = R(0,0) - r^T a = R(0,0) - r^T R^{-1} r$$

An Example of Predictive Coder

- Assume the current pixel $f_0 = f(m, n)$ is predicted from two pixels, one on the left, $f_1 = f(m, n-1)$, and one on the top, $f_2 = f(m-1,n)$. Assume the pixel values have zero means. The correlations are: $R(0,0) = R(1,1) = R(2,2) = \sigma_f^2$, $R(0,1) = \rho_h\sigma_f^2$, $R(0,2) = \rho_v\sigma_f^2$, $R(1,2) = R(2,1) = \rho_d\sigma_f^2$.



$$\hat{f}(m,n) = a_1 f(m,n-1) + a_2 f(m-1,n)$$

$$f_0 = f(m,n)$$
$$f_1 = f(m,n-1)$$
$$f_2 = f(m-1,n)$$
$$R(0,0) = E\{f(m,n)f(m,n)\}$$
$$R(1,1) = E\{f(m,n-1)f(m,n-1)\}$$
$$R(2,2) = E\{f(m-1,n)f(m-1,n)\}$$
$$R(0,1) = E\{f(m,n)f(m,n-1)\}$$
$$R(0,2) = E\{f(m,n)f(m-1,n)\}$$
$$R(1,2) = E\{f(m,n-1)f(m-1,n)\}$$
$$R(2,1) = E\{f(m-1,n)f(m,n-1)\}$$

$$\begin{bmatrix} R(1,1) & R(2,1) \\ R(2,1) & R(2,2) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} R(0,1) \\ R(0,2) \end{bmatrix} \Rightarrow \begin{bmatrix} 1 & \rho_d \\ \rho_d & 1 \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \begin{bmatrix} \rho_h \\ \rho_v \end{bmatrix}$$

$$\Rightarrow \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \frac{1}{1-\rho_d^2} \begin{bmatrix} 1 & -\rho_d \\ -\rho_d & 1 \end{bmatrix} \begin{bmatrix} \rho_h \\ \rho_v \end{bmatrix} = \frac{1}{1-\rho_d^2} \begin{bmatrix} \rho_h - \rho_d\rho_v \\ \rho_v - \rho_d\rho_h \end{bmatrix}$$

$$\sigma_p^2 = R(0,0) - \begin{bmatrix} R(0,1) & R(0,2) \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \sigma_f^2 \left(1 - \frac{\rho_h^2 + \rho_v^2 - 2\rho_v\rho_d\rho_h}{1-\rho_d^2} \right)$$

if the correlation is isotropic, $\rho_h = \rho_v = \rho$,

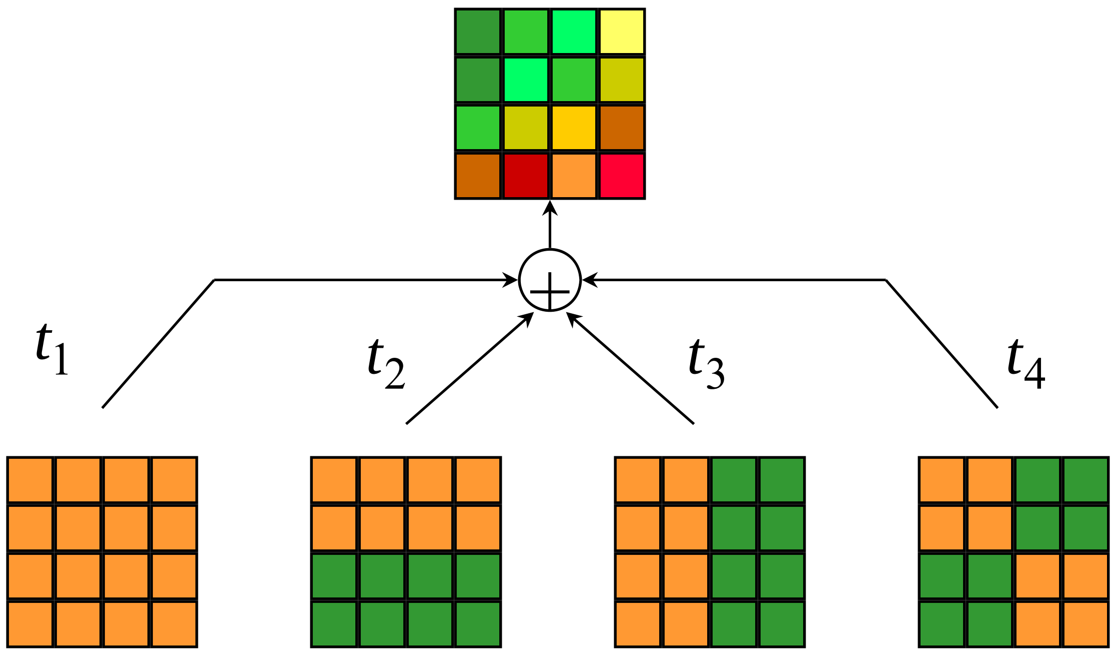
$$\begin{bmatrix} a_1 \\ a_2 \end{bmatrix} = \frac{\rho}{1+\rho_d} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$$\sigma_p^2 = \sigma_f^2 \left(1 - \frac{2\rho^2}{1+\rho_d} \right)$$

Note: when the pixel value has non-zero mean, the above predictor can be applied to mean-shifted values.

What is a Linear Transform

- Represent an image (or an image block) as the linear combination of some basis images and specify the linear coefficients.



One Dimensional Linear Transform

- Let C^N represent the N dimensional complex space.
- Let $\mathbf{h}_0$, $\mathbf{h}_1$, …, $\mathbf{h}_{N-1}$ represent N linearly independent vectors in C^N.
- For any vector $\mathbf{f} \in C^N$,

$$\mathbf{f} = \sum_{k=0}^{N-1} t(k)\mathbf{h}_k = \mathbf{B}\mathbf{t},$$

$$where \quad \mathbf{B} = [\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{N-1}], \mathbf{t} = \begin{bmatrix} t(0) \\ t(1) \\ \vdots \\ t(N-1) \end{bmatrix}.$$

$\Rightarrow$

$$\mathbf{t} = \mathbf{B}^{-1}\mathbf{f} = \mathbf{A}\mathbf{f}$$

f and t form a transform pair

Orthonormal Basis Vectors (OBV)

- $\{\mathbf{h}_k$, k=0,…N-1$\}$ are OBV if

$$<\mathbf{h}_k, \mathbf{h}_l> = \delta_{k,l} = \begin{cases} 1 & k = l \\ 0 & k \neq l \end{cases}$$

$$<\mathbf{h}_l, \mathbf{f}> = <\mathbf{h}_l, \sum_{k=0}^{N-1} t(k)\mathbf{h}_k> = \sum_{k=0}^{N-1} t(k) <\mathbf{h}_l, \mathbf{h}_k> = t(l) = \mathbf{h}_t^H \mathbf{f}$$

$$\mathbf{t} = \begin{bmatrix} \mathbf{h}_0^H \\ \mathbf{h}_1^H \\ \vdots \\ \mathbf{h}_{N-1}^H \end{bmatrix} \mathbf{f} = \mathbf{B}^H \mathbf{f} = \mathbf{A}\mathbf{f}$$

$$\mathbf{B}^{-1} = \mathbf{B}^H, \ or\ \mathbf{B}^H\mathbf{B} = \mathbf{B}\mathbf{B}^H = \mathbf{I}.$$

B is unitary

Definition of Unitary Transform

- Basis vectors are orthonormal
- $$t(k) = <\mathbf{h}_k, \mathbf{f}> = \sum_{n=0}^{N-1} h_k(n)^* f(n),$$

$$\mathbf{t} = \begin{bmatrix} \mathbf{h}_0^H \\ \mathbf{h}_1^H \\ \vdots \\ \mathbf{h}_{N-1}^H \end{bmatrix} \mathbf{f} = \mathbf{B}^H \mathbf{f} = \mathbf{A}\mathbf{f}$$

- $$f(n) = \sum_{k=0}^{N-1} t(k) h_k(n),$$

$$\mathbf{f} = \sum_{k=0}^{N-1} t(k)\mathbf{h}_k = [\mathbf{h}_0 \quad \mathbf{h}_1 \quad \cdots \quad \mathbf{h}_{N-1}]\mathbf{t} = \mathbf{B}\mathbf{t} = \mathbf{A}^H \mathbf{t}$$

Property of Unitary Transform

- Energy preservation: ||**f**||=||**t**||.

Proof: $\|\mathbf{f}\| = \mathbf{f}^H\mathbf{f} = \mathbf{t}^H\mathbf{A}\mathbf{A}^H\mathbf{t} = \mathbf{t}^H\mathbf{t} = \|\mathbf{t}\|$

- Mean vector relation:

$$\boldsymbol{\mu}_t = \mathbf{A}\boldsymbol{\mu}_f, \quad \boldsymbol{\mu}_f = \mathbf{A}^H\boldsymbol{\mu}_t, \quad where$$
$$\boldsymbol{\mu}_f = E\{\mathbf{f}\}, and\ \boldsymbol{\mu}_t = E\{\mathbf{t}\}$$

- Covariance relation:

$$\mathbf{C}_t = \mathbf{A}\mathbf{C}_f\mathbf{A}^H, \mathbf{C}_f = \mathbf{A}^H\mathbf{C}_t\mathbf{A}, \quad where$$
$$\mathbf{C}_f = E\{(\mathbf{f}-\boldsymbol{\mu}_f)(\mathbf{f}-\boldsymbol{\mu}_f)^H\}, \quad \mathbf{C}_t = E\{(\mathbf{t}-\boldsymbol{\mu}_t)(\mathbf{t}-\boldsymbol{\mu}_t)^H\}$$

Proof:

$$\mathbf{t}-\boldsymbol{\mu}_t = \mathbf{A}(\mathbf{f}-\boldsymbol{\mu}_f) \quad \Rightarrow \quad \mathbf{C}_t = E\{\mathbf{A}(\mathbf{f}-\boldsymbol{\mu}_f)(\mathbf{f}-\boldsymbol{\mu}_f)^H\mathbf{A}^H\} = \mathbf{A}\mathbf{C}_f\mathbf{A}^H.$$

Two Dimensional Unitary Transform

- $\{H_{k,l}\}$ is an orthonormal set of basis images
- Forward transform

$$T(k,l) = <\mathbf{H}_{k,l}, \mathbf{F}> = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} H_{k,l}^*(m,n) F(m,n)$$

- Inverse transform

$$F(m,n) = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} T(k,l) H_{k,l}(m,n), \quad or$$

$$\mathbf{F} = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} T(k,l) \mathbf{H}_{k,l}$$

Example of 2D Unitary Transform

$$\mathbf{H}_{00} = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix}, \mathbf{H}_{01} = \begin{bmatrix} 1/2 & 1/2 \\ -1/2 & -1/2 \end{bmatrix}, \mathbf{H}_{10} = \begin{bmatrix} 1/2 & -1/2 \\ 1/2 & -1/2 \end{bmatrix}, \mathbf{H}_{11} = \begin{bmatrix} 1/2 & -1/2 \\ -1/2 & 1/2 \end{bmatrix},$$

$$\mathbf{F} = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \quad \Rightarrow \quad \begin{cases} T(0,0) = 5 \\ T(0,1) = -2 \\ T(1,0) = -1 \\ T(1,1) = 0 \end{cases}$$

Separable Unitary Transform

- Let $\mathbf{h}_k$, k=0, 1, …, M-1 represent a set of orthonormal basis vectors in C^M,
- Let $\mathbf{g}_l$, l=0, 1, …, N-1 represent another set of orthonormal basis vectors in C^N,
- Let $\mathbf{H}_{k,l}=\mathbf{h}_k\mathbf{g}_l^T$, or $H_{k,l}(m,n)=h_k(m)g_l(n)$.
- Then $\mathbf{H}_{k,l}$ will form an orthonormal basis set in C^{MxN}.
- Transform can be performed separately, first row wise, then column wise

Example of Separable Unitary Transform

- Example 1

$$\mathbf{h}_0 = \begin{bmatrix} 1/\sqrt{2} \\ 1/\sqrt{2} \end{bmatrix}, \mathbf{h}_1 = \begin{bmatrix} 1/\sqrt{2} \\ -1/\sqrt{2} \end{bmatrix}.$$

$$\mathbf{H}_{00} = \mathbf{h}_0 \mathbf{h}_0^T = \begin{bmatrix} 1/2 & 1/2 \\ 1/2 & 1/2 \end{bmatrix} \qquad \mathbf{H}_{01} = \mathbf{h}_0 \mathbf{h}_1^T = \begin{bmatrix} 1/2 & -1/2 \\ 1/2 & -1/2 \end{bmatrix}$$

$$\mathbf{H}_{10} = \mathbf{h}_1 \mathbf{h}_0^T = \begin{bmatrix} 1/2 & 1/2 \\ -1/2 & -1/2 \end{bmatrix} \qquad \mathbf{H}_{11} = \mathbf{h}_1 \mathbf{h}_1^T = \begin{bmatrix} 1/2 & -1/2 \\ -1/2 & 1/2 \end{bmatrix}$$

- 2D DFT

$$H_{k,l}(m,n) = \frac{1}{\sqrt{MN}} e^{j2\pi\left(\frac{km}{M} + \frac{ln}{N}\right)},$$

$$h_k(m) = \frac{1}{\sqrt{M}} e^{j2\pi\frac{km}{M}}, \quad g_l(n) = \frac{1}{\sqrt{N}} e^{j2\pi\frac{ln}{N}}$$

Why Using Transform?

- When the transform basis is chosen properly
  - Many coefficients have small values and can be quantized to 0 w/o causing noticeable artifacts
  - The coefficients are uncorrelated, and hence can be coded independently w/o losing efficiency.

Transform Basis Design

- Optimality Criteria:
  - *Energy compaction:* a few basis images are sufficient to represent a typical image.
  - *Decorrelation:* coefficients for separated basis images are uncorrelated.
- Karhunen Loeve Transform (KLT) is the Optimal transform for a given covariance matrix of the underlying signal (the vector of pixels in a block).
  - Must be computed for a given image or a collection of training data
  - KLT basis design not required!
- Discrete Cosine Transform (DCT) is close to KLT for images that can be modeled by a first order Markov process (*i.e.*, a pixel only depends on its previous pixel).
  - Fixed transform

Basis Images of DCT

$$h(m,n,u,v) = \alpha(u)\alpha(v)\cos\left[\frac{(2m+1)u\pi}{2N}\right]\cos\left[\frac{(2n+1)v\pi}{2N}\right]$$

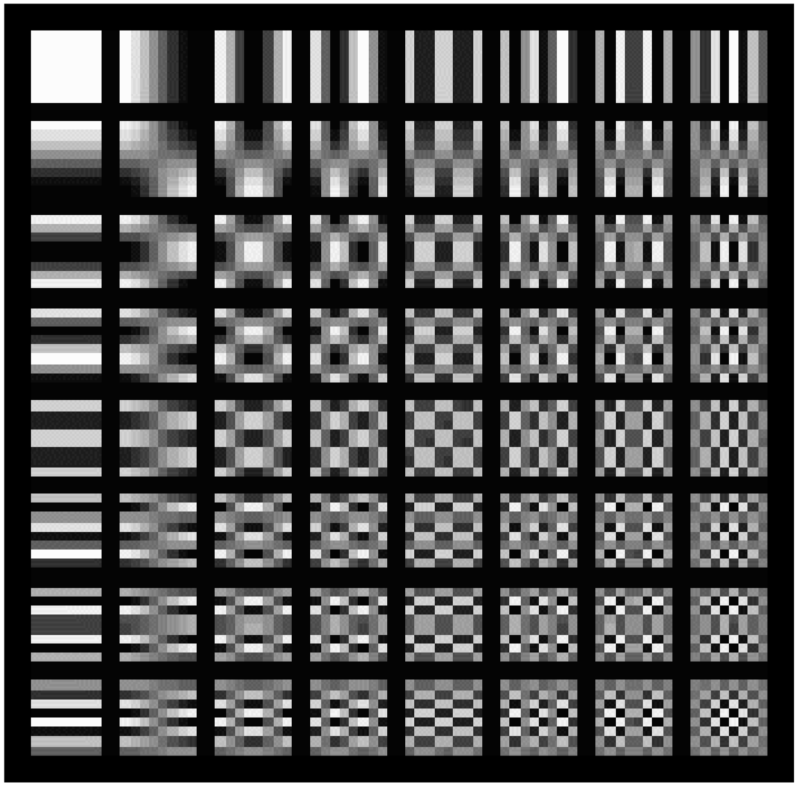
$$\text{where } \alpha(u) = \begin{cases} \sqrt{1/N} & u = 0 \\ \sqrt{2/N} & u = 1,\dots,N-1 \end{cases}$$

$$T(u,v) = \sum_{m=0}^{N-1}\sum_{n=0}^{N-1} f(m,n)h(m,n,u,v)$$

$$f(m,n) = \sum_{u=0}^{N-1}\sum_{v=0}^{N-1} T(u,v)h(m,n,u,v)$$

Low-Low

High-Low



Low-High

High-High

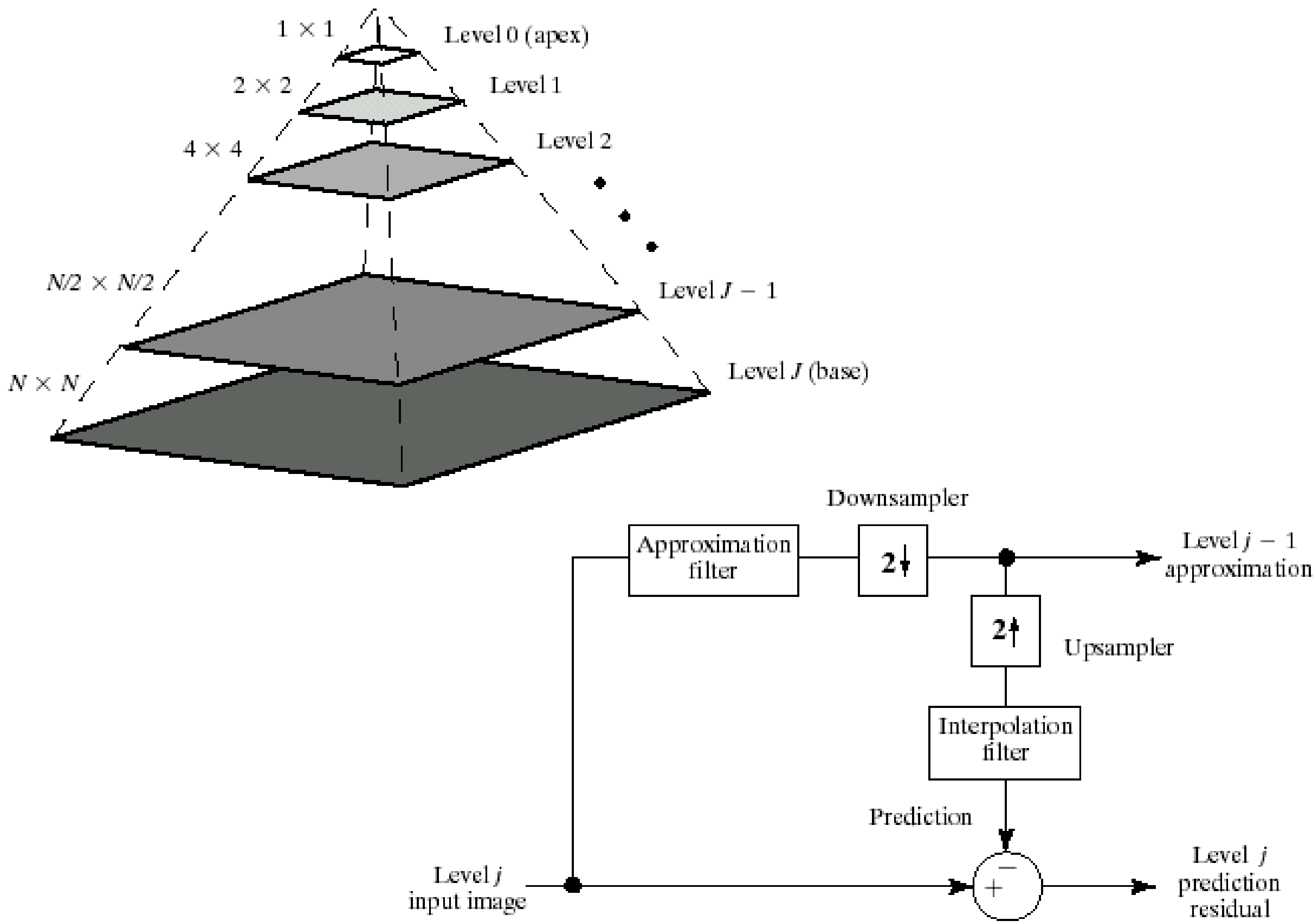
JPEG Image Coding Standard

- JPEG (Baseline) uses block-DCT
  - Divide an image into 8x8 non-overlapping blocks
  - For each block:
    - Apply Discrete Cosine Transform
    - Quantize DCT coefficients (uniform quantizer with different stepsizes for different coefficients)
    - Zig-zag ordering of quantized DCT coefficients
    - Create Run-length representation
    - Huffman coding of run-length symbols
- Pros and Cons
  - Good coding efficiency
  - Simple
  - Blocking artifacts at low bit rate
  - No scalability

Wavelet Transform

- Multi-resolution representation using a pyramid decomposition
- Wavelet transform through tree-structured subband decomposition

Multi-Resolution Representation (aka Pyramid Representation)



a b

FIGURE 7.2 (a) A pyramidal image structure and (b) system block diagram for creating it.

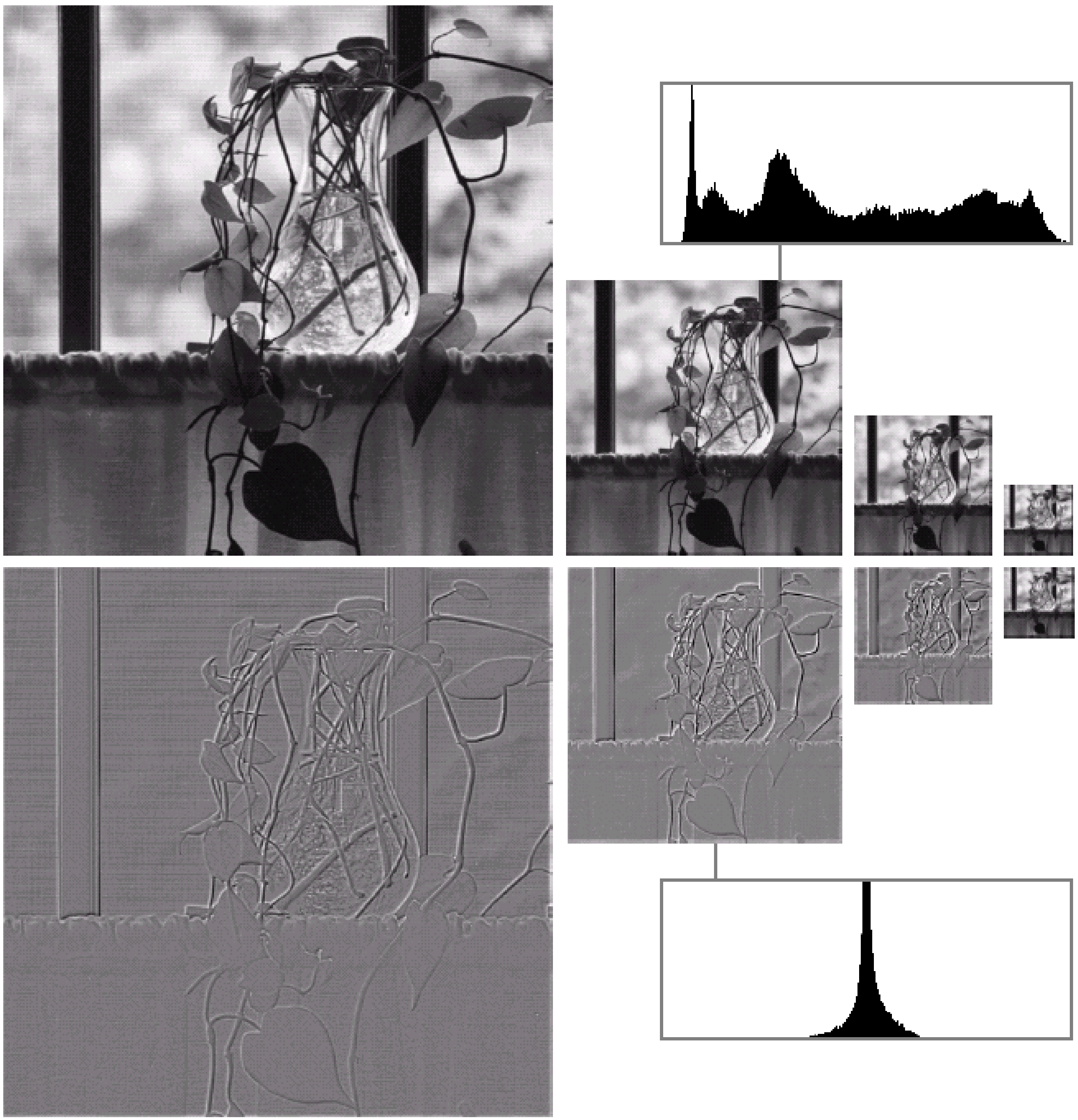
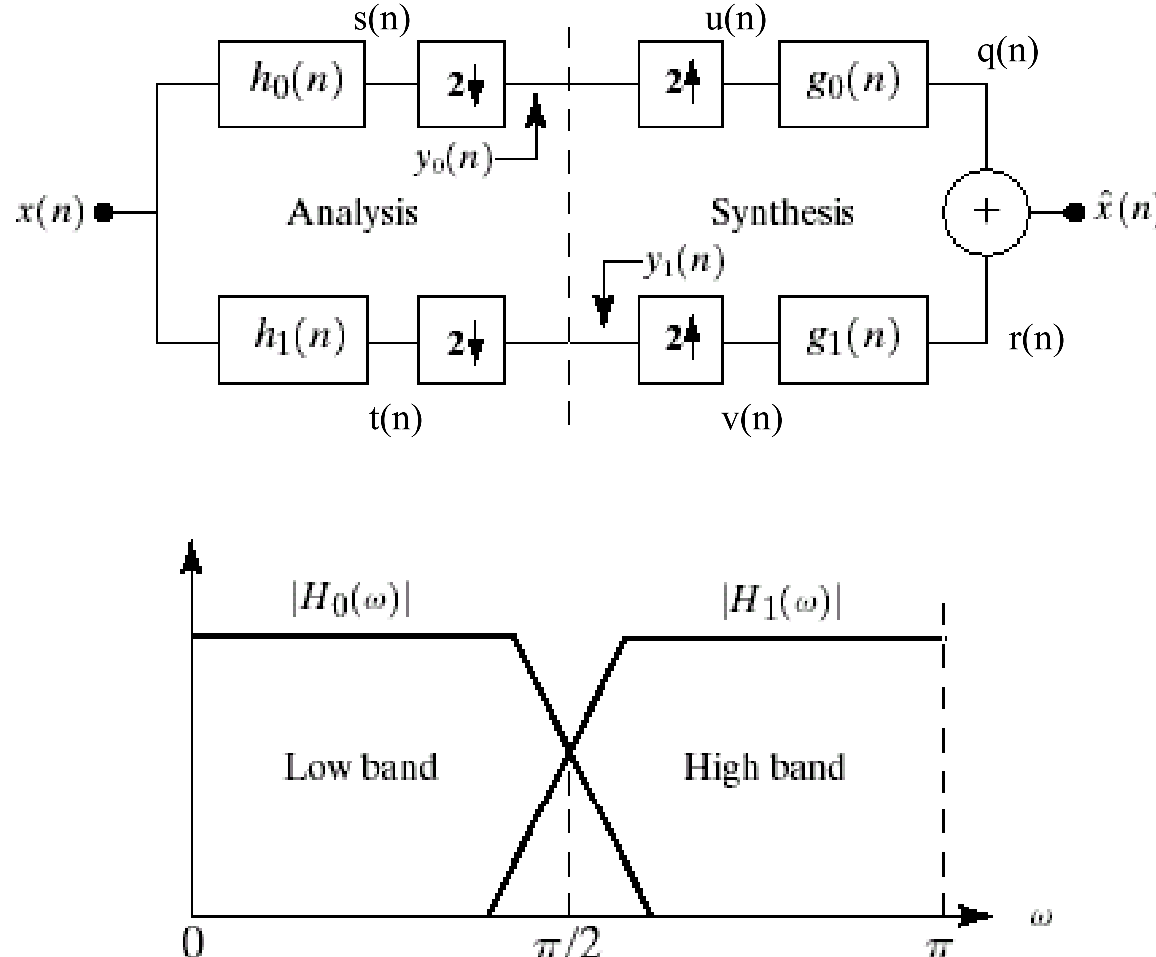


FIGURE 7.3 Two image pyramids and their statistics: (a) a Gaussian (approximation) pyramid and (b) a Laplacian (prediction residual) pyramid.

Two Band Filterbank

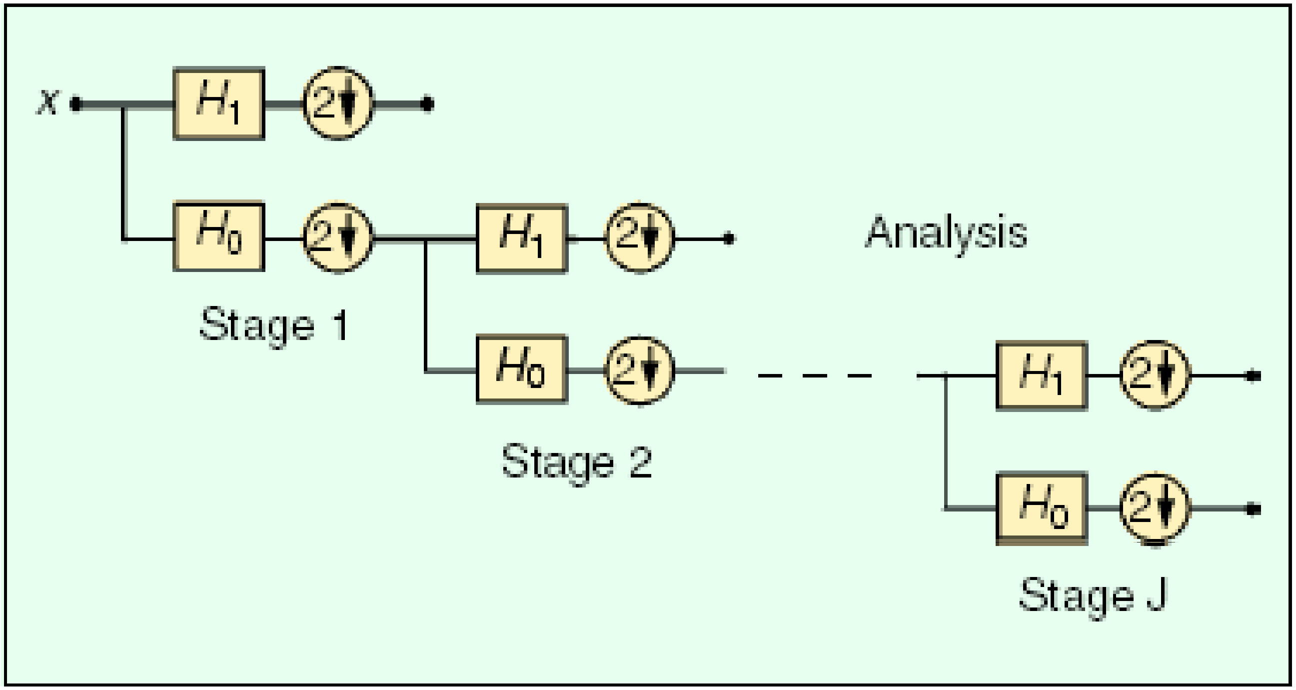


a
b

FIGURE 7.4 (a) A two-band filter bank for one-dimensional subband coding and decoding, and (b) its spectrum splitting properties.

h0: Lowpass filter, y0: a blurred and then down-sampled version of x
h1: Highpass filter, y1: edges in x
When the filters h0,h1, g0, g1 are designed appropriately,
X^=X (perfect reconstruction filterbank)

Iterated Filter Bank



▲ 3. Iterated filter bank. The lowpass branch gets split repeatedly to get a discrete-time wavelet transform.

From [Vetterli01]

How to Apply Filterbank to Images?

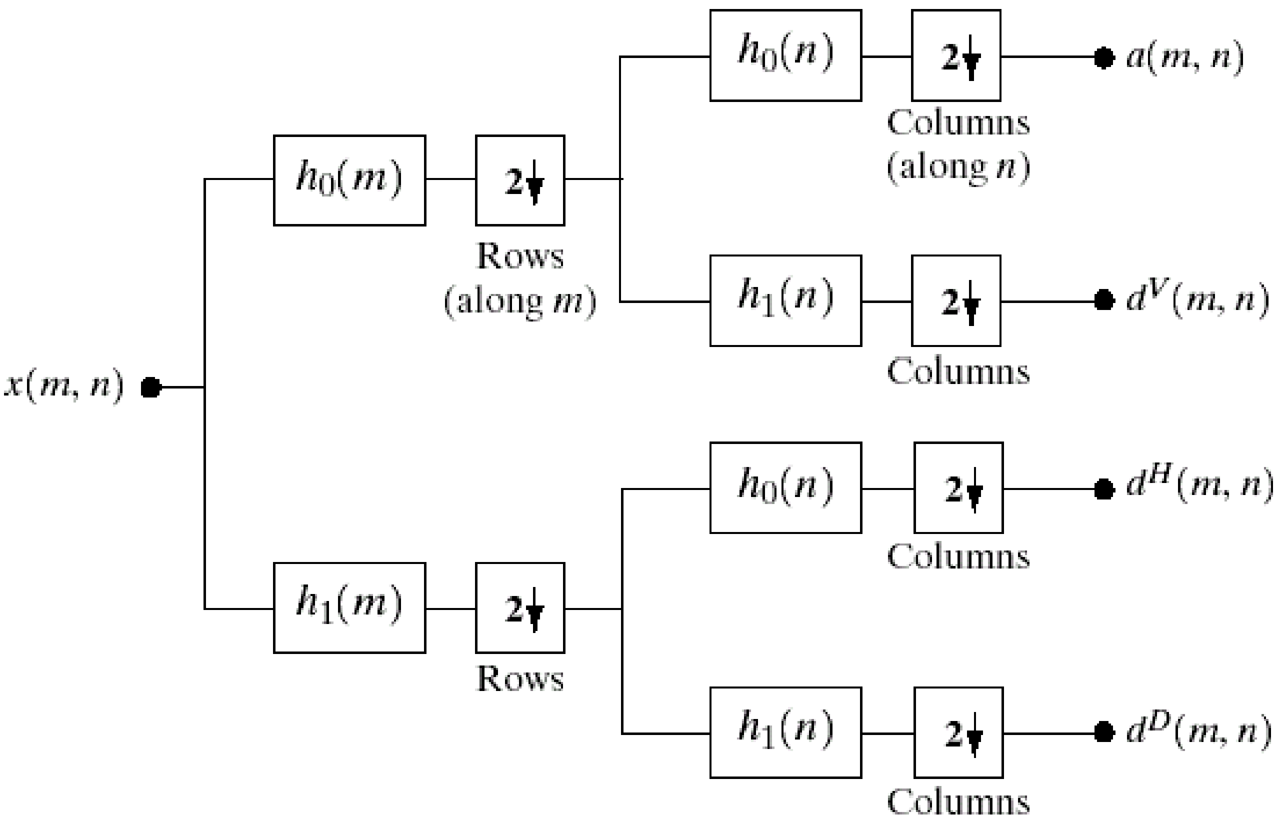
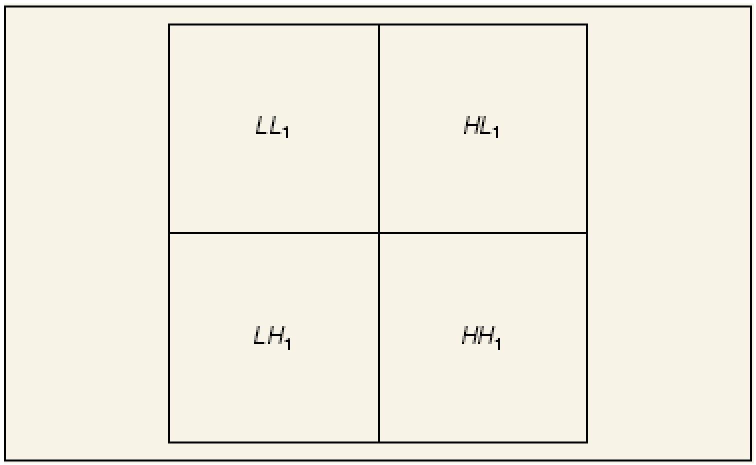


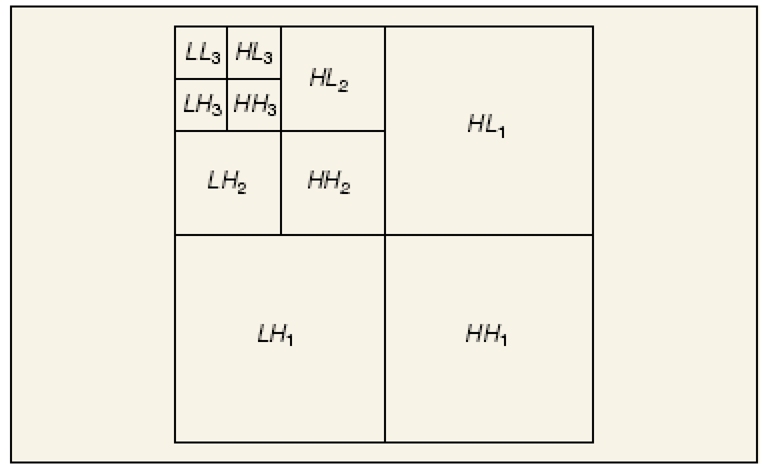
FIGURE 7.5 A two-dimensional, four-band filter bank for subband image coding.

2D decomposition is accomplished by applying the 1D decomposition along rows of an image first, and then columns.

Wavelet Transform for Images

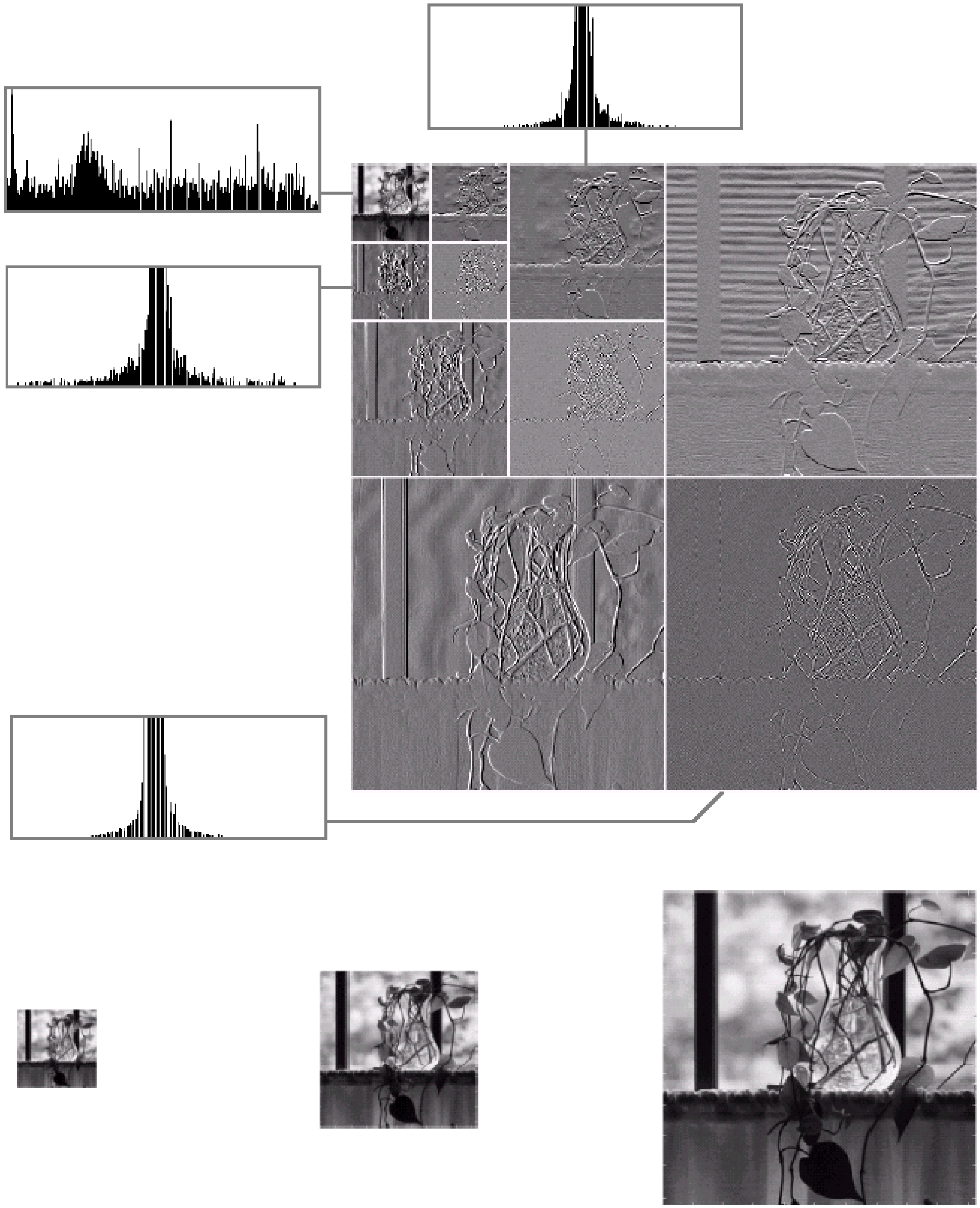


▲ 4. The subband labeling scheme for a one-level, 2-D wavelet transform.



▲ 6. The subband labeling scheme for a three-level, 2-D wavelet transform.

From [Usevitch01]



a
b c d

FIGURE 7.8 (a) A discrete wavelet transform using Haar basis functions. Its local histogram variations are also shown; (b)–(d) Several different approximations (64×64, 128×128, and 256×256) that can be obtained from (a).

JPEG2000

- Uses wavelet transforms
- Divide an image into tiles, process each tile of each color component separately
- For each tile
  - Apply wavelet transform
  - Divide the coefficients into code blocks
  - Represent each code block in bit planes
  - Code successive bit planes using context-based arithmetic coding
  - Resulting bits are packetized into a scalable bit stream
    - Coarse scale coefficients first
    - Higher bit planes first within each scale
- Details of JPEG2000 coding algorithms not required.

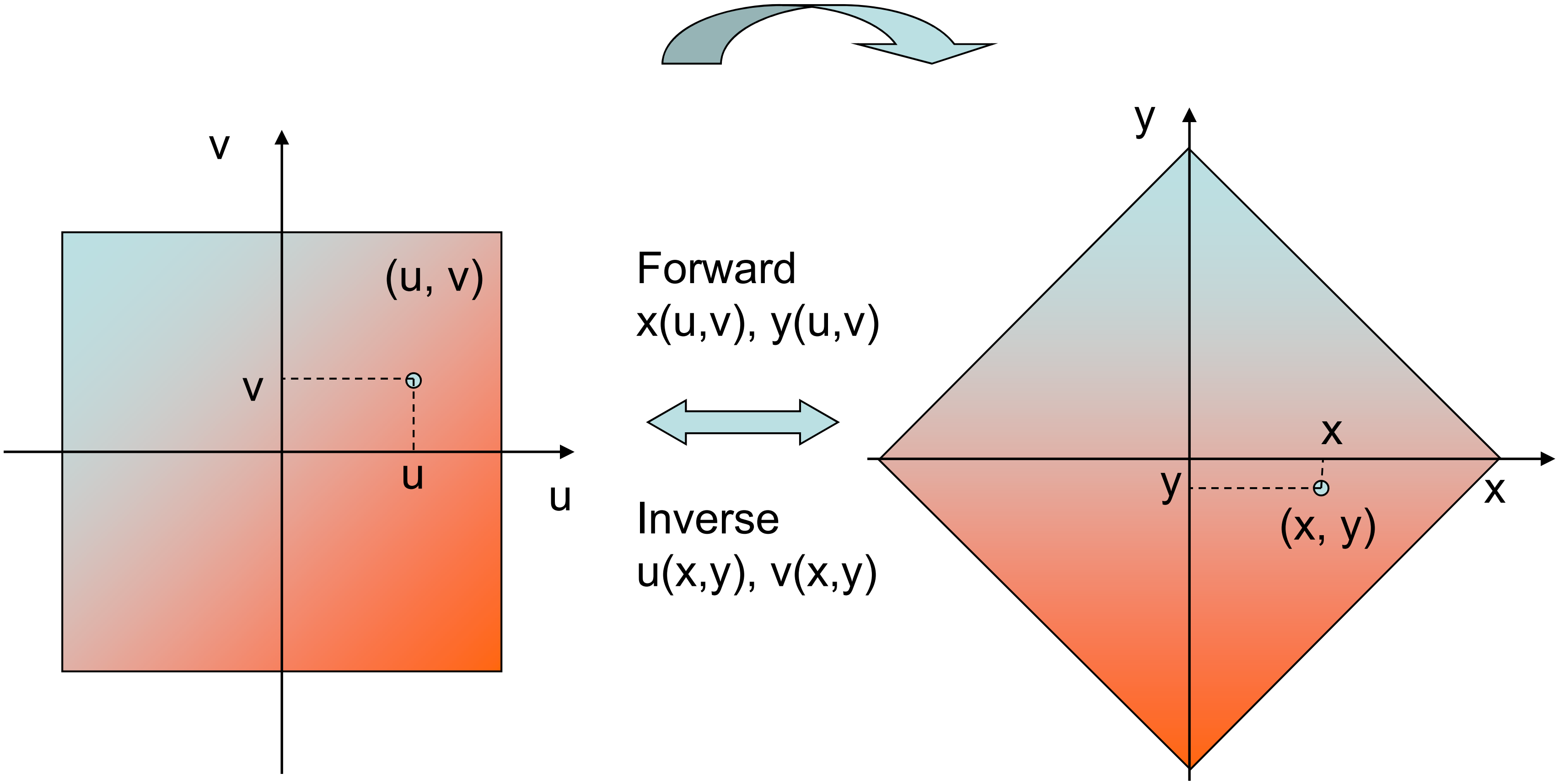
JPEG2000 Features

- Improved coding efficiency
  - Primarily due to efficient entropy coders for bit planes of wavelet coefficients
- Full quality scalability
  - From lossless to lossy at different bit rate
  - Enabled by bit plane coding
- Spatial scalability
  - Enabled by wavelet transform: code the coefficients from coarse to fine scale
- Region of interests
- More demanding in memory and computation time than JPEG

What is Geometric Transformation?

- So far, the image processing operations we have discussed modify the color values of pixels in a given image
- With geometric transformation, we modify the positions of pixels in a image, but keep their colors unchanged
  - To create special effects
  - To register two images taken of the same scene at different times or by different sensors
  - To morph one image to another

Illustration of Forward and Inverse Mapping Functions



Translation, Scaling, Rotations

- Should know the mapping functions for each
- Can be combined and represented with a general form of

$$\mathbf{x} = \mathbf{RS}(\mathbf{u} + \mathbf{t}) = \mathbf{Au} + \mathbf{b},$$
$$\mathbf{u} = \mathbf{A}^{-1}(\mathbf{x} - \mathbf{b}) = \mathbf{A}^{-1}\mathbf{x} + \mathbf{c},$$
$$\textit{with } \mathbf{A} = \mathbf{RS}, \mathbf{b} = \mathbf{RSt}, \mathbf{c} = -\mathbf{t}.$$

- Note that interchanging the order of operations will lead to different results.

Polynomial Warping

- The polynomial warping includes all deformations that can be modeled by polynomial transformations:

$$\begin{cases} x = a_0 + a_1 u + a_2 v + a_3 uv + a_4 u^2 + a_5 v^2 + \cdots \\ y = b_0 + b_1 u + b_2 v + b_3 uv + b_4 u^2 + b_5 v^2 + \cdots \end{cases}$$

- Special cases:
  - *Affine Mapping*, which has only first order terms:

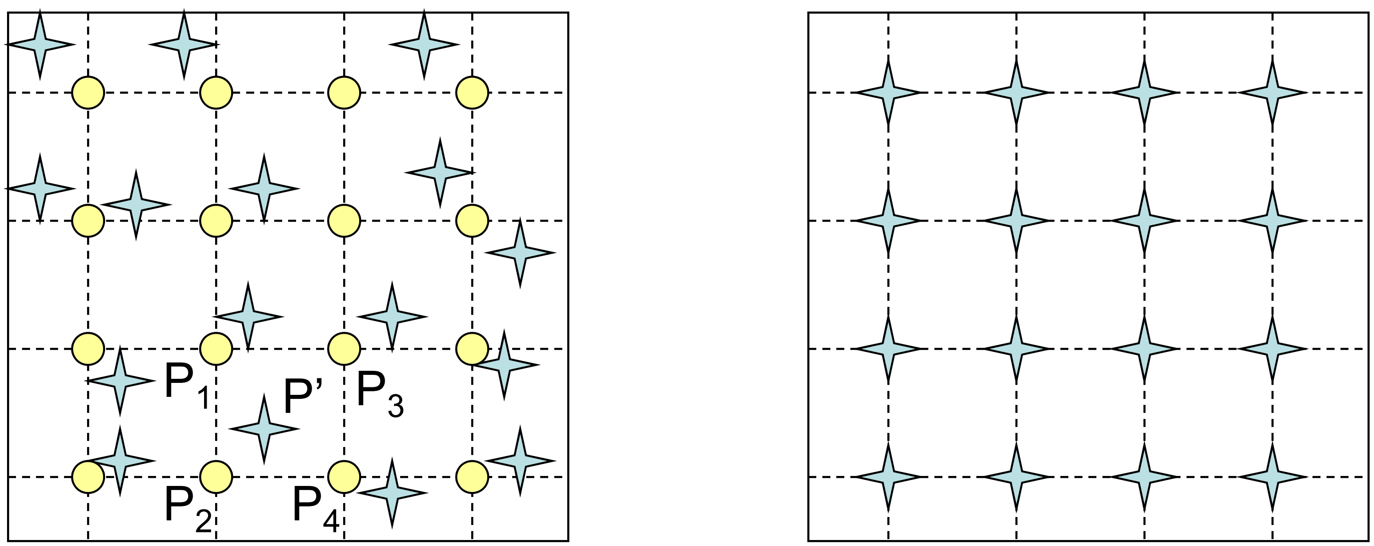
$$\begin{cases} x = a_0 + a_1 u + a_2 v \\ y = b_0 + b_1 u + b_2 v \end{cases}$$

  - Bilinear Mapping

$$\begin{cases} x = a_0 + a_1 u + a_2 v + a_3 uv \\ y = b_0 + b_1 u + b_2 v + b_3 uv \end{cases}$$

Image Warping by Inverse Mapping

- Inverse Mapping
  - For each point (x, y) in the image to be obtained, find its corresponding point (u, v) in the original image using the inverse mapping function, and let g(x, y) = f(u, v)



P' will be interpolated from P_1, P_2, P_3, and P_4

Image Registration

- Suppose we are given two images taken at different times of the same object. To observe the changes between these two images, we need to make sure that they are aligned properly. To obtain this goal, we need to find the correct mapping function between the two. The determination of the mapping functions between two images is known as the registration problem.
- Once the mapping function is determined, the alignment step can be accomplished using the warping methods.

How to determine the mapping function?

- Assume the mapping function is a polynomial of order N
- Step 1: Identify K≥N corresponding points between two images, i.e.

$$(u_i, v_i) \leftrightarrow (x_i, y_i), i = 1,2..., K.$$

- Step 2: Determine the coefficients a_i, b_i, i = 0,…,N-1 by solving

$$\begin{cases} x(u_i, v_i) = a_0 + a_1 u_i + a_2 v_i + \cdots = x_i, \\ y(u_i, v_i) = b_0 + b_1 u_i + b_2 v_i + \cdots = y_i, \end{cases} \quad i = 1,2,..., K$$

Image Registration Method (2)

- Step 2:

$$\mathbf{A}\mathbf{a} = \mathbf{x}, \quad \mathbf{A}\mathbf{b} = \mathbf{y}$$

where

$$\mathbf{A} = \begin{bmatrix} 1 & u_1 & v_1 & \cdots \\ 1 & u_2 & v_2 & \cdots \\ \vdots & \vdots & \vdots & \ddots \\ 1 & u_K & v_K & \cdots \end{bmatrix}, \mathbf{a} = \begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N-1} \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_0 \\ b_1 \\ \vdots \\ b_{N-1} \end{bmatrix}, \mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_K \end{bmatrix}, \mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_K \end{bmatrix}$$

If K = N, and the matrix **A** is non-singular, then

$$\mathbf{a} = \mathbf{A}^{-1}\mathbf{x}, \quad \mathbf{b} = \mathbf{A}^{-1}\mathbf{y}$$

If K > N, then we can use a least square solution

$$\mathbf{a} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{x}, \quad b = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T y$$

If K < N, or **A** is singular, then more corresponding feature points must be identified.

Image Morphing

- Image morphing has been widely used in movies and commercials to create special visual effects. For example, changing a beauty gradually into a monster.
- The fundamental techniques behind image morphing is image warping.
- Let the original image be f(**u**) and the final image be g(**x**). In image warping, we create g(**x**) from f(**u**) by changing its shape. In image morphing, we use a combination of both f(**u**) and g(**x**), to create a series of images in between f(**u**) and g(**x**),

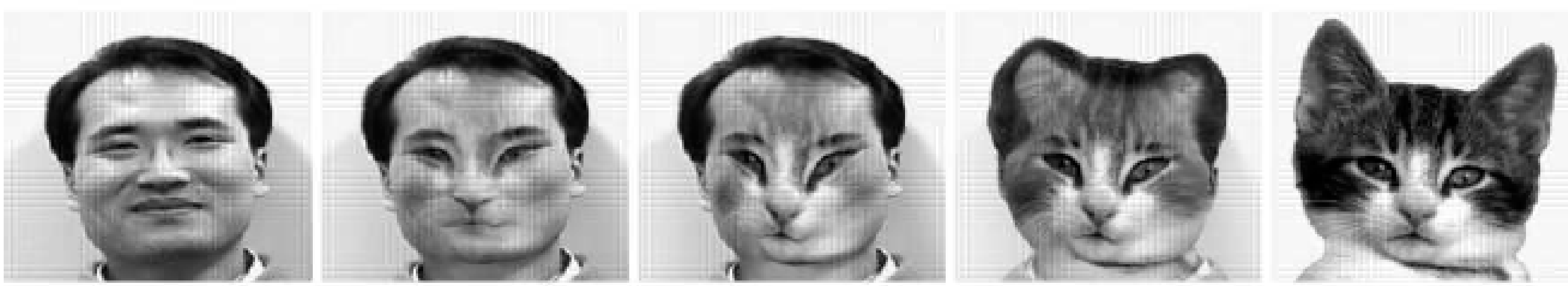
$$h_k(\mathbf{u} + s_k\mathbf{d}) = (1 - s_k) f(\mathbf{u}) + s_k g(\mathbf{u} + \mathbf{d}(\mathbf{u})), \quad k = 0,1,\ldots,K,$$
$$where\ s_k = k / K.$$

Examples of Image Morphing

Cross
Dissolve

I(t) = (1-t)*S+t*T

Mesh
based



George Wolberg, “Recent Advances in Image Morphing”,
Computer Graphics Intl. '96, Pohang, Korea, June 1996.

Image Restoration

- How to model the image degradation process
  - Transformation (linear or nonlinear) + Noise
  - Linear model
    - $g(x,y)= h(x,y)*f(x,y)+ n(x,y)$
- How to estimate h(x,y) for common degradation processes
  - Spatial blurring
  - Motion blurring
- How to restore the image with given h(x,y)?
  - Inverse filtering
  - Wiener filtering
  - Problems and fixes…

Final Exam Logistics

- Scheduled time: 12/22 1:30-4:30, RH615
- Closed-book, 1 sheet of notes allowed (double sided OK)
- Only cover topics after midterm exam
- See previous notes on topics that will not be covered.
- Office hour:
  - 12/20 4-6PM. Contact me for other times by email
  - Last two HW will be due on 12/15 5PM outside door of my office (LC256)
  - Solutions to last three HWs will be posted on my.poly by 12/16

Follow-Up Courses

- EL6123 - Video processing
  – TV systems, video sampling and format conversion, motion estimation, video coding techniques, video coding and communication standards, video transport over networks
  – Require a term project
  – http://eeweb.poly.edu/~yao/EL612/
- CS6643 - Computer vision
- Other related courses
  – EL7133: Digital signal processing
  – EL7163: Multiresolution signal processing
  – CS6533: Computer Graphics
  – EL5823: Medical Imaging I
  – EL5143: Multimedia Lab