Multiscale Image Transforms

Goal: Develop filter-based representations to decompose images into *component* parts, to extract features/structures of interest, and to attenuate noise.

Motivation:

- extract image features such as edges and corners
- isolate potentially independent image components
  - different locations, scales, orientations
  - independent measurement (evidence)
- redundancy reduction and image modeling for
  - efficient coding
  - image enhancement/restoration
  - image analysis/synthesis
- predictable behaviour under deformation
  - through time (motion) or between views (stereo)

Examples:

- DFT/DCT (global and blocked)
- Gabor Transform, Gabor wavelets
- Haar Transform
- Laplacian Pyramid
- Steerable Pyramid

Readings: Chapters 7, 8, and Sections 9.1-9.2 of Forsyth and Ponce.
Matlab Tutorials: imageTutorial.m and pyramidTutorial.m.

Linear Transform Framework

Projection Vectors: Let $\vec{\mathbf{I}}$ denote a 1D signal, or a vectorized representation of an image (so $\vec{\mathbf{I}} \in \mathcal{R}^N$), and let the transform be

$$\vec{\mathbf{a}} \;=\; \mathbf{P}^T \vec{\mathbf{I}} \,. \tag{1}$$

Here,

- $\vec{\mathbf{a}} = [a_0, ..., a_{M-1}] \in \mathcal{R}^M$ are the transform coefficients.
- The columns of $\mathbf{P} \;=\; [\vec{\mathbf{p}}_0, \vec{\mathbf{p}}_1, ..., \vec{\mathbf{p}}_{M-1}]$ are the projection vectors; the m^{th} coefficient, a_m, is the inner product $\vec{\mathbf{p}}_m{}^T \vec{\mathbf{I}}$
- When $\mathbf{P}$ is complex-valued, we should replace $\mathbf{P}^T$ by the conjugate transpose $\mathbf{P}^{*T}$

Sampling: The transform $\mathbf{P}^T \in \mathcal{R}^{M \times N}$ is said to be *critically sampled* when $M = N$. Otherwise it is *over-sampled* when $M > N$, or *under-sampled* when $M < N$.

Basis Vectors: For many transforms of interest there is a corresponding basis matrix $\mathbf{B}$ satisfying

$$\vec{\mathbf{I}} \;=\; \mathbf{B}\,\vec{\mathbf{a}} \,. \tag{2}$$

The columns $\mathbf{B} \;=\; [\vec{\mathbf{b}}_0, \vec{\mathbf{b}}_1, ..., \vec{\mathbf{b}}_{M-1}]$ are called basis vectors as they form a linear basis for $\vec{\mathbf{I}}$:

$$\vec{\mathbf{I}} \;=\; \sum_{m=0}^{M-1} a_m \,\vec{\mathbf{b}}_m$$

Linear Transform Framework (cont)

Completeness

- the forward transform (1) is complete, encoding all image structure, if it is invertible.
- when critically sampled, it is complete if $\mathbf{B} = (\mathbf{P}^T)^{-1}$ exists.
- if over-sampled, the transform is complete if $rank(\mathbf{P}) = N$.
  In this case $\mathbf{B}$ is not unique – one choice is the pseudoinverse

$$\mathbf{B} = (\mathbf{P}^T\mathbf{P})^{-1}\,\mathbf{P}^T$$

- if under-sampled, then $rank(\mathbf{P}) < N$ and it is not invertible in general.

Self-Inverting

- the transform is self-inverting if $\vec{\mathbf{b}}_m = \alpha\vec{\mathbf{p}}_m$ for some constant α.
- in the critically-sampled, self-inverting case the transform is orthogonal (unitary), up to the constant α (e.g., the DFT).

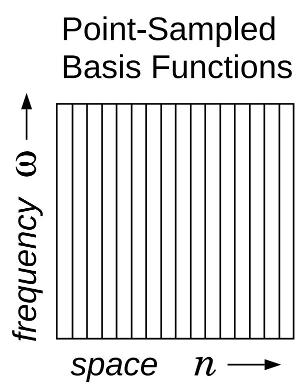
Global Transforms

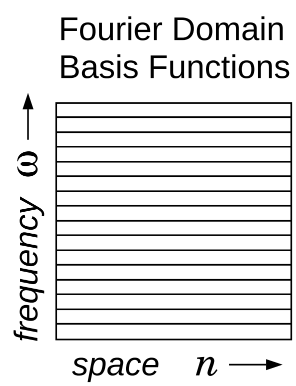
Point-Sampled Representation

- The sampled representation from the CCD array. The projection functions are shifted impulses, $\delta(n-k, m-l)$, which are, of course, orthogonal
- *Problem:*
  - Ideal localization in space, but global in Fourier domain.
    Therefore, no scale or orientation specificity.
  - We also find significant correlations among samples

Fourier Transform (DFT)

- DFT encodes image as a sum of *global* sinusoids: $e^{i\vec{\omega}_k\vec{n}}$
- localized in Fourier domain
- critically sampled for complex-valued signals
- *Problem:* not localized in space.





Gabor Transform

Joint Localization: Dennis Gabor (1946) showed that the Gaussian minimizes joint uncertainty (the product of variances) in space and Fourier domain.
The Fourier transform of a Gaussian function is a Gaussian:

$$g(x) \;=\; \frac{1}{\sqrt{2\pi}\sigma}\, e^{-x^2/2\sigma^2} \quad , \quad \hat{g}(\omega) \;=\; e^{-\omega^2\sigma^2/2} \; .$$

The product of their variances is 1.

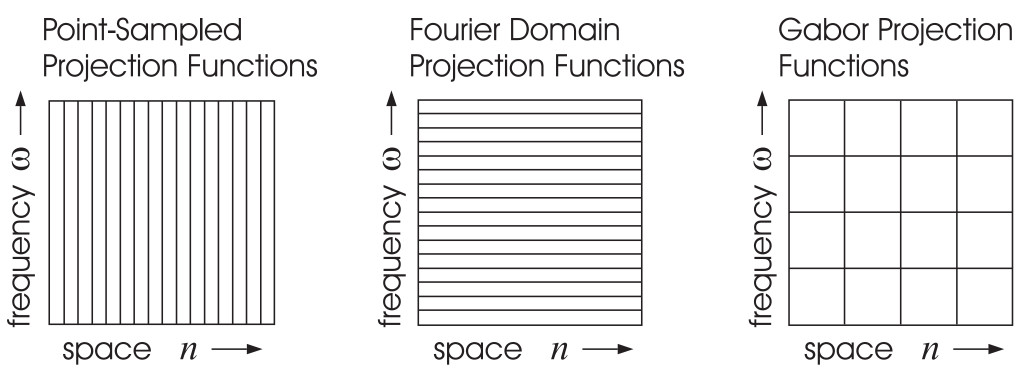
Gabor Transform (*aka* the Gaussian windowed Fourier Transform):

- One applies a Gaussian window at a point (n_0, m_0), followed by a DFT (like a blocked DFT/DCT transform, in which the image is broken into non-overlapping square blocks on which the DFT/DCT is applied, but with Gaussian window instead of a square window):

$$\mathcal{F}\,[\, g(n - n_0, m - m_0)\, I(n, m)\,]$$

- The resulting projection directions (often called Gabor functions), along with their Fourier spectra are given by

$$p_k(n) \;=\; g(n)\, e^{i\omega_k n} \quad , \quad \hat{p}_k(\omega) \;=\; \hat{g}(\omega - \omega_k)$$



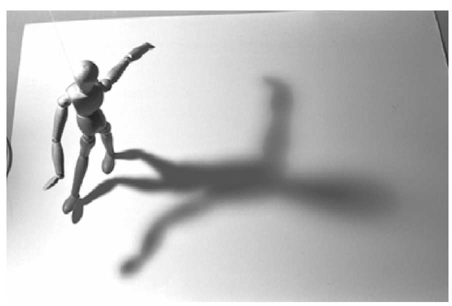
Gabor projection functions are *smooth* and *compact* in both space and frequency domain. They are complex-valued, and for smaller bandwidths (e.g., less than an octave) they are approximately a quadrature pair. The transform coefficients are also complex-valued.
But these projection functions are non-orthognal, and the resulting basis functions are not local, nor well-behaved.

Multiscale Image Transforms

Salient image structure occurs at multiple scales, suggesting some degree of position- and scale-invariant processing.

1) Objects and their parts occur at multiple scales:

2) Cast shadows cause edges to occur at many scales:

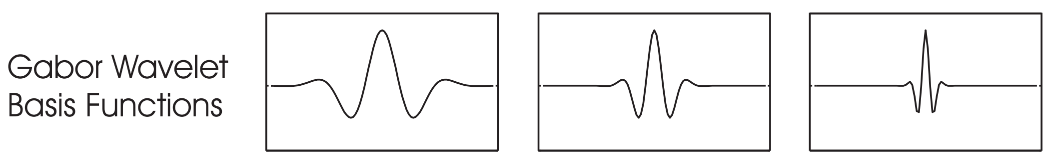


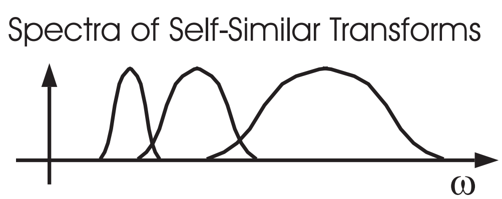
3) Objects may project into the image at different scales:

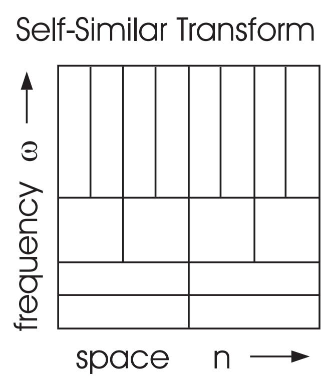
Self-Similar Multiscale Transforms

Goal: The filter support should grow with scale, and be well matched to scale-dependent correlation lengths in images. The represenation should exhibit scale-invariant properties, as objects project to images at different scales depending on distance from camera.

Scale Self-Similarity: Let the basis functions be dilations and translations of a "mother" function, so they all have the same shape, differing in scale and position only.





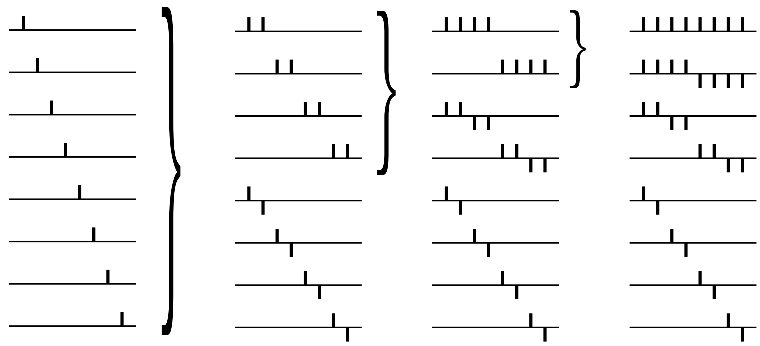


Examples:

- Gabor wavelets
- Haar Transform
- Laplacian Pyramid
- Steerable Pyramid

Haar Transforms

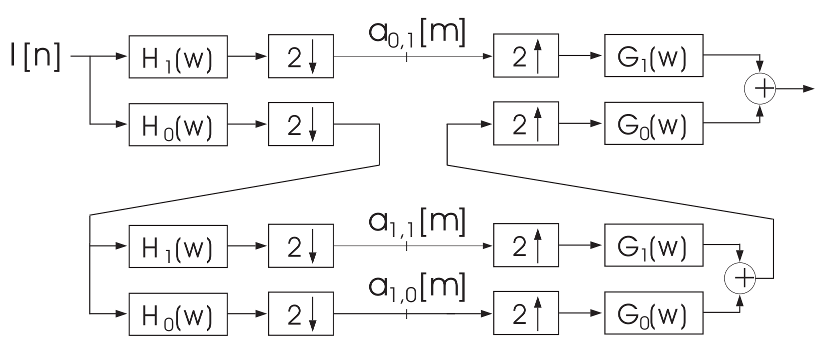
Originally described by A. Haar (1909). Each step creates two channels: one simply averages adjacent elements (i.e., low-pass channel); and one takes difference between adjacent elements (i.e., a high-pass channel). Both are down-sampled by 2.



Properties:

- critically-sampled and self-inverting (orthogonal)
- local in space (compact) but not continuously differentiable
- broad ringing frequency spectrum due to top-hat spatial window, and therefore massive aliasing in each band (like blocked DCT).
- very efficient to compute with pyramid scheme and addition

Analysis / Synthesis Diagram:

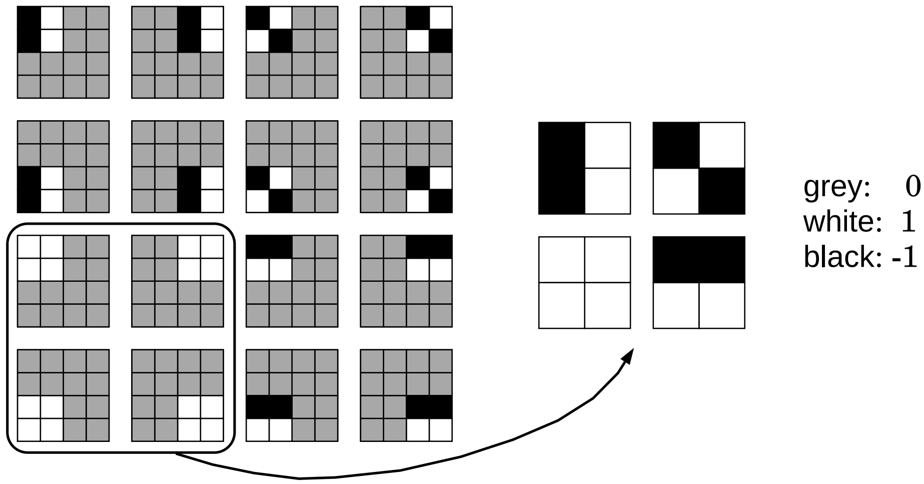


Analysis/Synthesis system diagram for
a 2-level cascaded pyramid filter bank

This is an analysis-synthesis diagram for a general 2-level cascaded pyramid (where the low-pass portion is further filtered). It shows the recursive construction of the transform. For the Haar transform, h_0 and h_1 are low-pass and high-pass filters that compute sums and differences (respectively) of adjacent pixels. Moreover, $G_j(\omega) = H_j(-\omega)$, and so the transform can be shown to be self-inverting. Finally, although there is aliasing in the individual channels of the Haar transform, one can show that, upon reconstruction, the aliasing in the transform channels cancels, so reconstruction is exact.

2D Haar Transforms

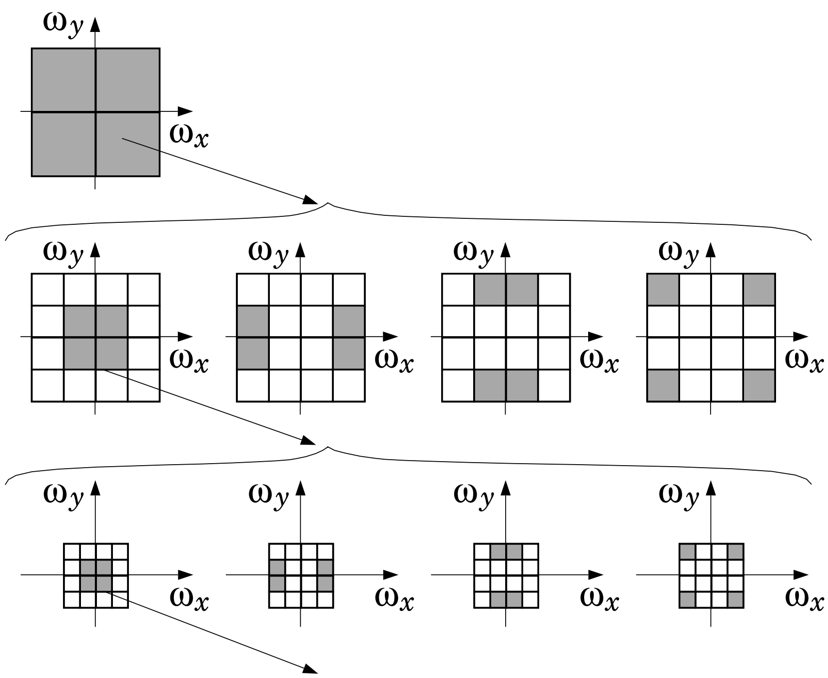
Recursive design of 2D Haar basis functions:



Separable 2D filters:

$$\begin{pmatrix}1\\1\end{pmatrix}\begin{pmatrix}1 & 1\end{pmatrix} \quad \begin{pmatrix}1\\1\end{pmatrix}\begin{pmatrix}1 & -1\end{pmatrix} \quad \begin{pmatrix}1\\-1\end{pmatrix}\begin{pmatrix}1 & 1\end{pmatrix} \quad \begin{pmatrix}1\\-1\end{pmatrix}\begin{pmatrix}1 & -1\end{pmatrix}$$

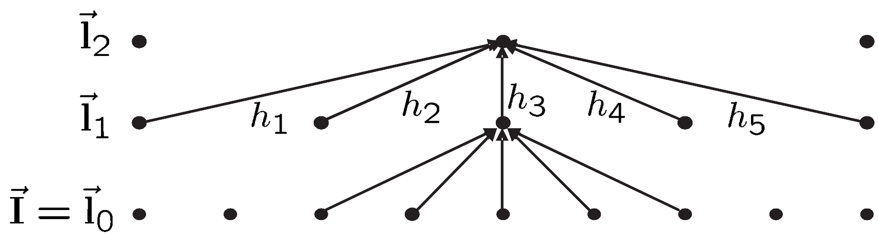
Idealized band-splitting in the frequency domain:



Gaussian Pyramid

Sequence of low-pass, down-sampled images, $[\vec{\mathbf{I}}_0,\ \vec{\mathbf{I}}_1,\ ...,\ \vec{\mathbf{I}}_N]$.

Usually constructed with a separable 1D kernel $\mathbf{h} = [h_1, h_2, h_3, h_4, h_5]$, and a down-sampling factor of 2 (in each direction):



In matrix notation (for 1D) the mapping from one level to the next has the form:

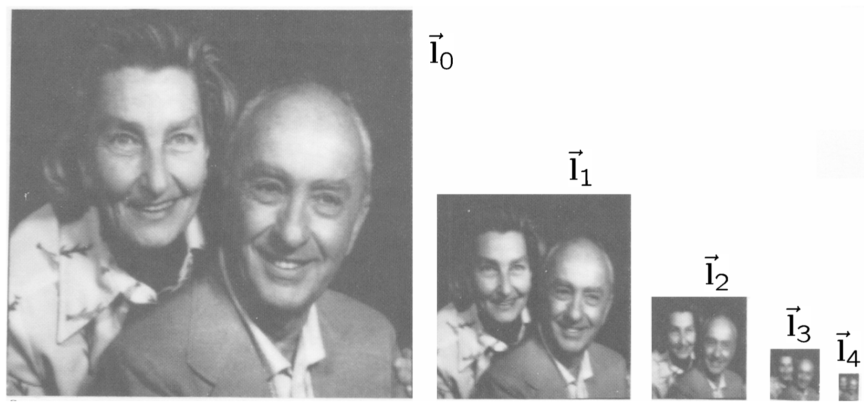
$$\vec{\mathbf{I}}_{k+1} \;=\; \mathbf{R}\,\vec{\mathbf{I}}_k \;=\; \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 1 & \\ & & \vdots & & & \ddots \end{bmatrix}}_{\text{down-sampling}} \underbrace{\begin{bmatrix} \ddots & & & \\ & -\,\mathbf{h}\,- & & \\ & & -\,\mathbf{h}\,- & \\ & & & -\,\mathbf{h}\,- \\ & & & & \ddots \end{bmatrix}}_{\text{convolution}} \vec{\mathbf{I}}_k$$

Typical weights for the impulse response from binomial coefficients

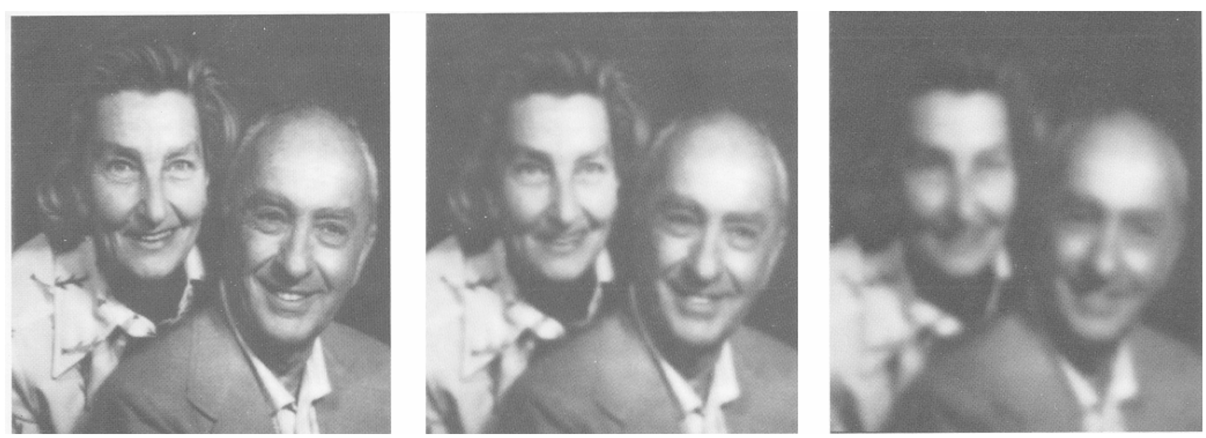
$$\mathbf{h} = \frac{1}{16}[1,\ 4,\ 6,\ 4,\ 1]$$

Gaussian Pyramid (cont)

Example of original image and four more pyramid levels:



First three levels scaled to be the same size:



Properties of Gaussian pyramid:

- used for multi-scale edge estimation
- efficient for computing coarse-scale images (only separable 5-tap kernels are used)
- highly redundant (coarse-scale information is duplicated in fine scale images)

Laplacian Pyramid

Over-complete decomposition based on difference-of-lowpass filters; the image is recursively decomposed into low-pass and highpass bands (like the Haar Transform).

- Each band of the Laplacian pyramid is the difference between two adjacent low-pass images of the Gaussian pyramid, $[\vec{\mathbf{l}}_0,\ \vec{\mathbf{l}}_1,\ ...,\ \vec{\mathbf{l}}_N]$. That is:

$$\vec{\mathbf{b}}_k \ = \ \vec{\mathbf{l}}_k - \mathbf{E}\,\vec{\mathbf{l}}_{k+1}$$

where $\mathbf{E}\,\vec{\mathbf{l}}_{k+1}$ is an up-sampled, smoothed version of $\vec{\mathbf{l}}_{k+1}$ (so that it will have the same dimension as $\vec{\mathbf{l}}_k$), i.e.,

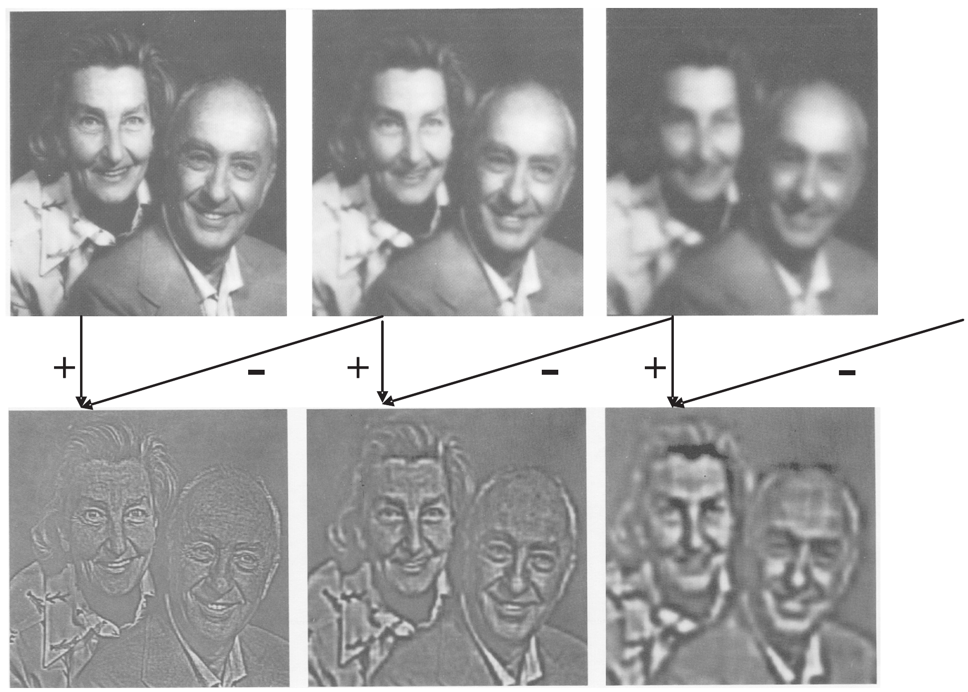
$$\mathbf{E}\,\vec{\mathbf{l}}_{k+1} \ = \ \underbrace{\begin{bmatrix} \ddots & & & \\ & -\,\mathbf{g}\,- & & \\ & & -\,\mathbf{g}\,- & \\ & & & -\,\mathbf{g}\,- \\ & & & & \ddots \end{bmatrix}}_{\text{convolution}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & \\ 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \\ & & \vdots & & \ddots \end{bmatrix}}_{\text{up-sampling}} \vec{\mathbf{l}}_{k+1}$$

Often the filters used to construct the Gaussian and Laplacian pyramids, $\mathbf{g}$ and $\mathbf{h}$, are identical.

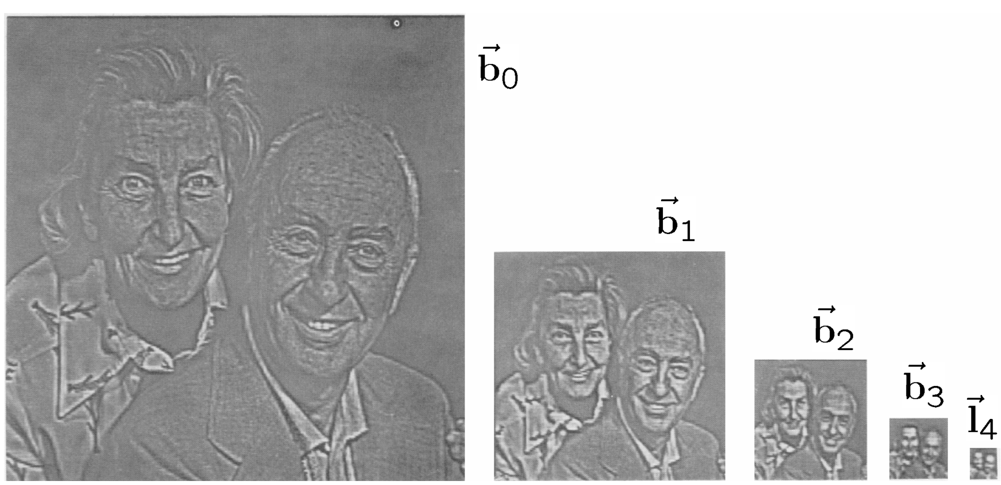
The **Laplacian pyramid** with L levels is given by $[\vec{\mathbf{b}}_0,\ \vec{\mathbf{b}}_1, ...,\ \vec{\mathbf{b}}_{L-1},\ \vec{\mathbf{l}}_L]$. The representation is overcomplete by a factor of roughly of $\frac{4}{3}$ for 2D images (i.e., 1 + 1/4 + 1/16 + ... = 4/3).

Laplacian Pyramid (cont)

Construction of the Laplacian bands:



A Laplacian pyramid with four levels:



The transform coefficients are the pixel values of these images.

Laplacian Pyramid (cont)

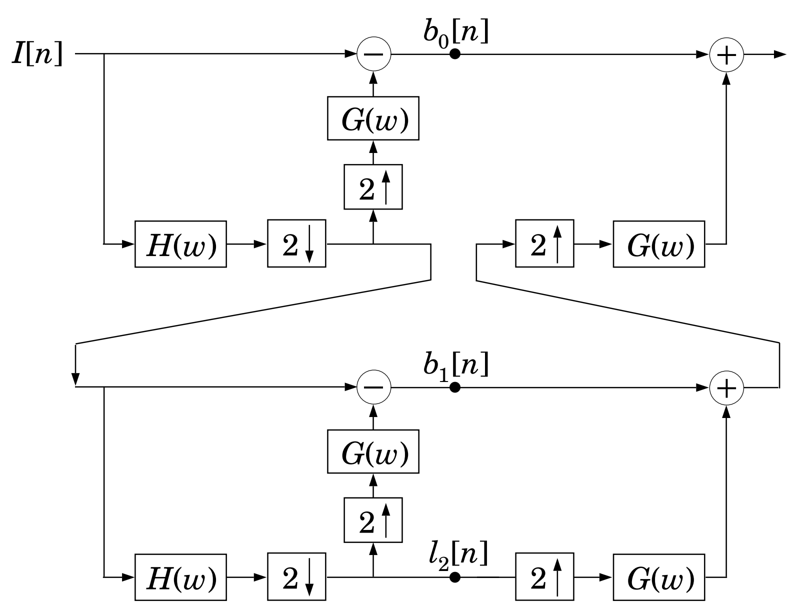
Construction of $[\vec{\mathbf{b}}_0, \vec{\mathbf{b}}_1, ..., \vec{\mathbf{b}}_{L-1}, \vec{\mathbf{l}}_L]$:

$$\begin{aligned}\vec{\mathbf{l}}_0 &= \vec{\mathbf{I}}\\ \vec{\mathbf{l}}_{k+1} &= \mathbf{R}\,\vec{\mathbf{l}}_k\\ \vec{\mathbf{b}}_k &= \vec{\mathbf{l}}_k - \mathbf{E}\,\vec{\mathbf{l}}_{k+1}\end{aligned}$$

Reconstruction of $\vec{\mathbf{I}}$ is exact for any filter (for $k = L-1, \ldots, 0$):

$$\begin{aligned}\vec{\mathbf{l}}_k &= \vec{\mathbf{b}}_k + \mathbf{E}\,\vec{\mathbf{l}}_{k+1}\\ \vec{\mathbf{I}} &= \vec{\mathbf{l}}_0\end{aligned}$$

System Diagram: shows the filters and sampling steps used for pyramid construction, and then image reconstruction from the transform coefficients. Gaussian pyramid levels are computed using $h(n)$ (with spectrum $H(\omega)$). Filter $g(n)$ (with spectrum $G(\omega)$) is used with up-sampling so that adjacent Gaussian levels can be subtracted.



Analysis/synthesis diagram for a 2-layer Laplacian pyramid

Laplacian Pyramid Filters

In practice:

- often use same filters for h and g (i.e., we apply the same operators for smoothing and interpolation in construction and reconstruction)
- use separable lowpass filters (for efficiency)
- desire isotropy for h and g so all orientations handled the same way.

Constraints on 5-tap lowpass filter h:

- even-symmetry means that taps are $h = \left(\frac{a_2}{2}, \frac{a_1}{2}, a_0, \frac{a_1}{2}, \frac{a_2}{2}\right)$.
- assume that dc signal is preserved, i.e. $\hat{h}(0) = 1$:

$$\hat{h}(0) \;=\; \sum_{n=-2}^{2} h(n)\, e^{-i\,0\,n} \;=\; a_0 + a_1 + a_2$$

- assume that spectrum decays to 0 at fold-over rate, i.e. $\hat{h}(\pi) = 0$:

$$\hat{h}(\pi) \;=\; \sum_{n=-2}^{2} h(n)\, e^{-i\,\pi\,n} \;=\; a_0 - a_1 + a_2$$

- So $a_1 = a_0 + a_2 = 0.5$, and there is one free constraint. For example, choose $a_0 = \frac{6}{16}$, then h is the binomial 5-tap filter:

$$h(n) = \frac{1}{16}\,(1, 4, 6, 4, 1)$$

Historical remark on name of pyramid: The well-known Laplacian filter (isotropic second derivative) is given by

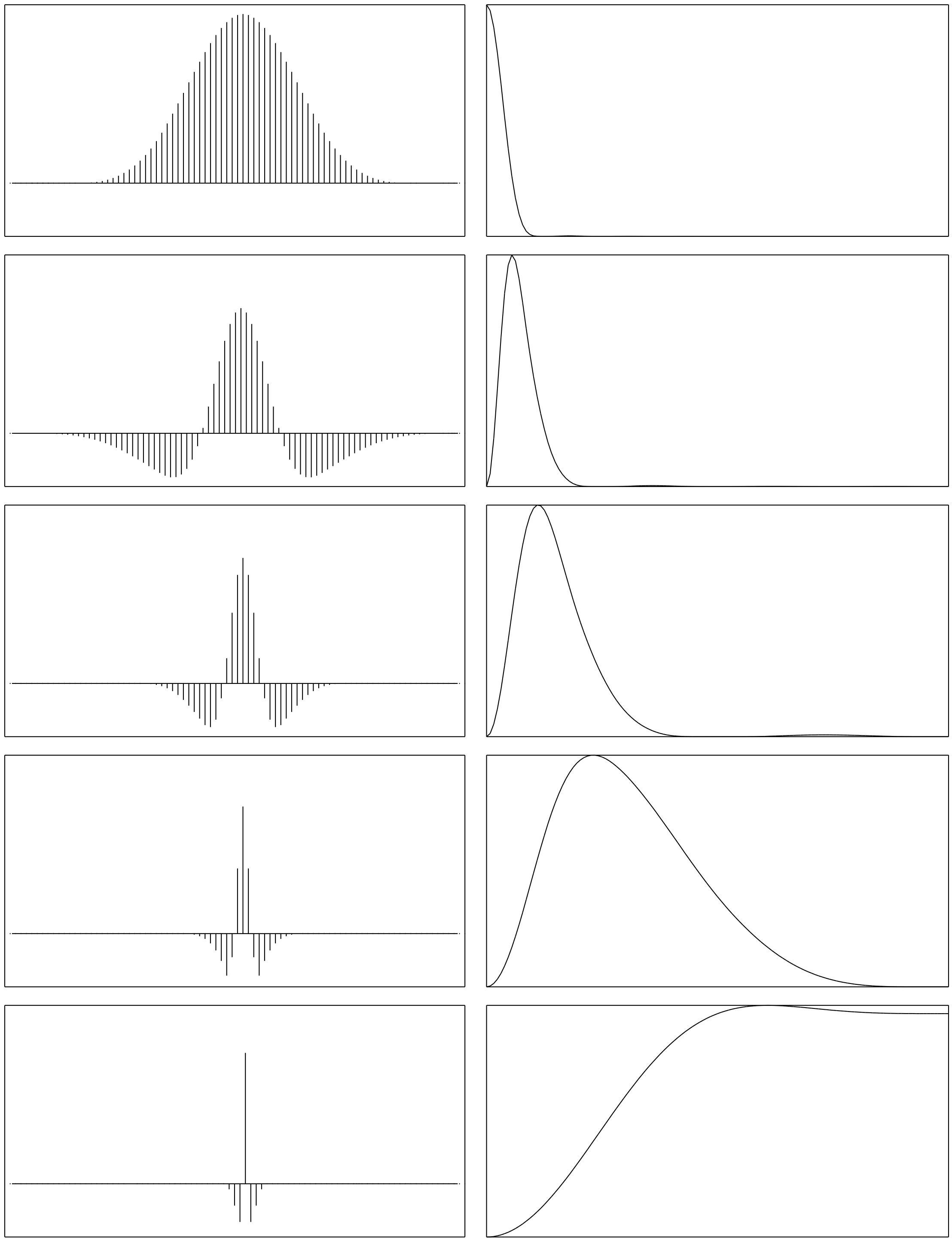
$$\nabla^2 f(x, y) \;=\; \frac{\partial^2 f}{\partial x^2} \;+\; \frac{\partial^2 f}{\partial y^2}$$

For Gaussian kernels, $g(x; \sigma) = \frac{1}{\sqrt{2\pi}\sigma}\, e^{-x^2/2\sigma^2}$,

$$\frac{d^2 g(x;\sigma)}{dx^2} \;=\; c_0\, \frac{d\, g(x;\sigma)}{d\sigma} \;\approx\; c_1\,(g(x;\sigma) - g(x;\sigma + \Delta\sigma))$$

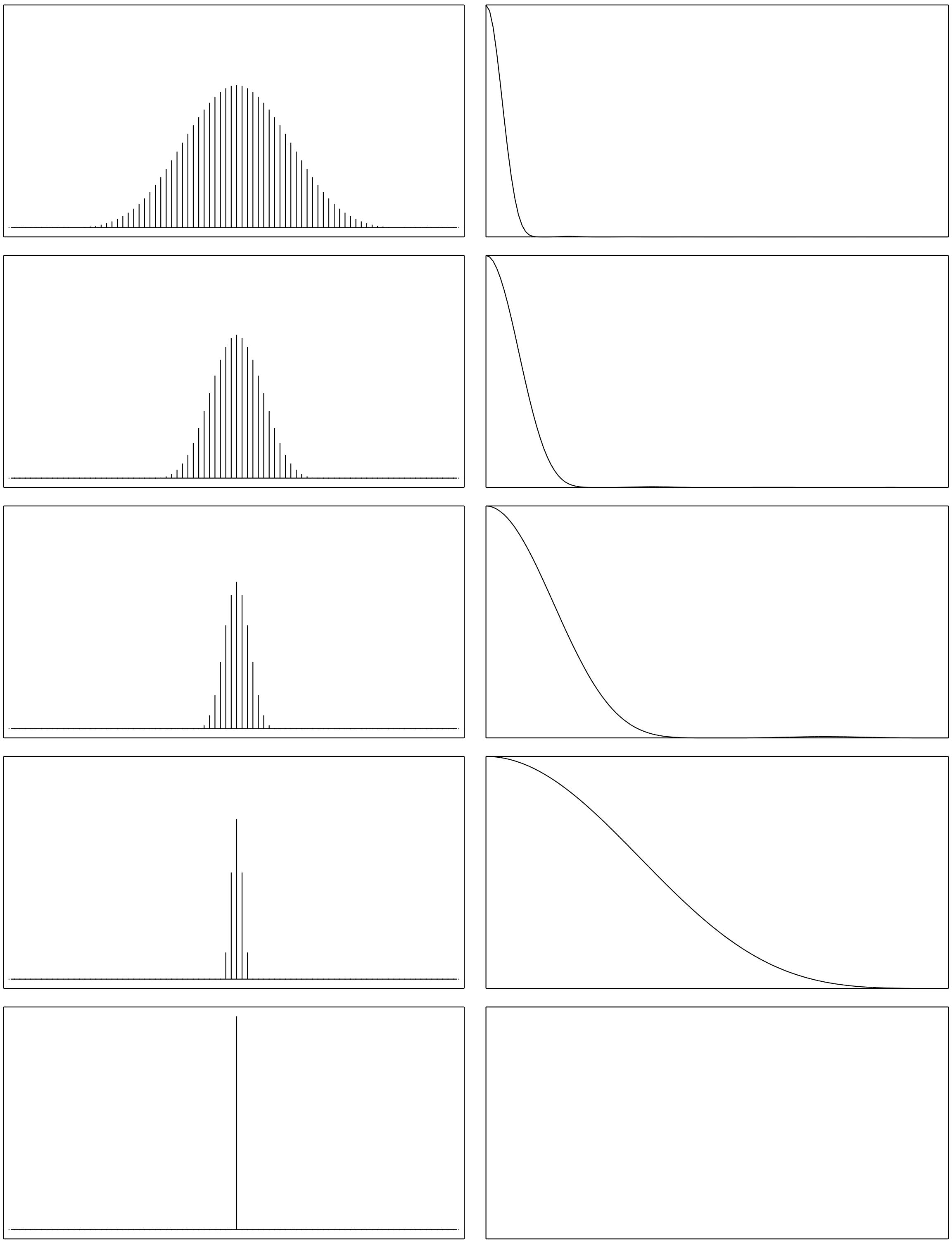
That is, if the low-pass filter h used to create the Laplacian pyramid is Gaussian, then the Laplacian pyramid levels approximate the second derivative of the image at different scales σ.

Laplacian Pyramid Projection Vectors:



Laplacian Projection Vectors   Fourier Spectra

Laplacian Pyramid Basis Vectors:



Laplacian Basis Vectors   Fourier Spectra

Uses of Laplacian Pyramid: Coding

Multiscale image representations are natural for image coding and transmission. The same basic ideas underly JPEG encoding.

Approach: Use quantization levels that become more coarse as one moves to higher frequency pass bands.

- high frequency coefficients are more coarsely coded (i.e., to fewer bits) than lower frequency bands.
- vast majority of the coefficients are in high frequency bands.
- this quantization matches human contrast sensitivity (roughly)

Advantages:

- eliminates blocking artifacts of JPEG at low frequencies because of the overlapping basis functions.
- approach also allows for progressive transmission, since low-pass representations are reasonable approximations to the image.
- coding and image reconstruction are simple

0.03 0.1 0.31 0.81 1.58

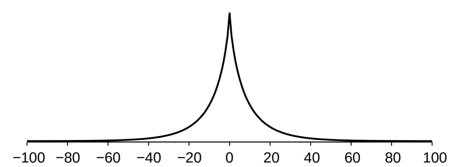
bits per pixel

Uses of Laplacian Pyramid: Restoration (Coring)

Transform coefficients for the Laplacian transform are often near zero. Significantly non-zero values are generally sparse.

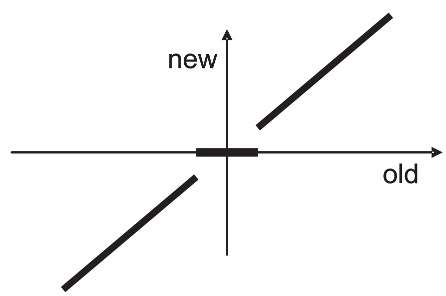
Histograms of transform coefficients are often well approximated by a so-called ”generalized Laplacian” density, $c\,e^{-|x/s|^k}$, where

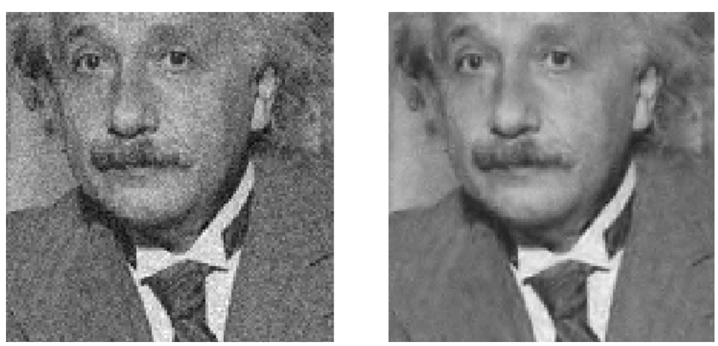
- k is usually between 0.7 and 1.2
- s controls the variance
- peaked at 0, with heavy tails



Coring:

- set all sufficiently small transform coefficients to zero,
- leave others unchanged, and possibly clip at large magnitudes.





Original image + additive noise (SNR = 9dB)

Cored image (SNR = 13.82dB)

Uses of Laplacian Pyramid: Image Compositing

Goal: Image stiching without visible seams. Register and mask the images. Then smooth the boundary neighbourhood in a scale-dependent way to avoid visible boundary aritfacts.

Method:

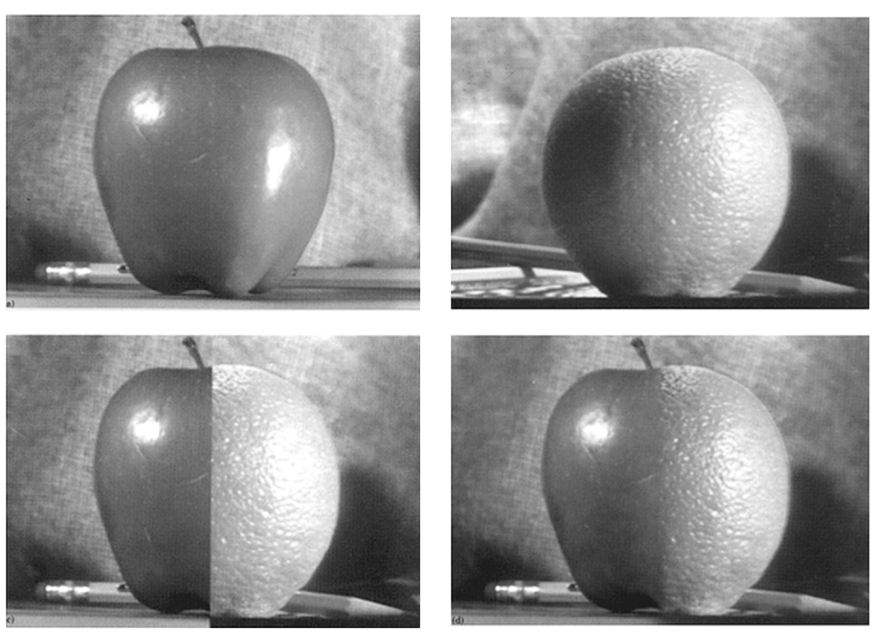
- assume images $I_1(\vec{\mathbf{n}})$ and $I_2(\vec{\mathbf{n}})$ are registered (aligned) and let $m_1(\vec{\mathbf{n}})$ be a mask that is 1 at pixels where we want the brightness from $I_1(\vec{\mathbf{n}})$ and 0 otherwise (i.e., where we want to see $I_2(\vec{\mathbf{n}})$).
- create Gaussian pyramid for $m_1(\vec{\mathbf{n}})$, denoted $\{l_0(\vec{\mathbf{n}}), l_1(\vec{\mathbf{n}}), ..., l_L(\vec{\mathbf{n}})\}$
- create Laplacian pyramids for $I_1(\vec{\mathbf{n}})$ and $I_2(\vec{\mathbf{n}})$, denoted by

  $$\{b_{1,0}(\vec{\mathbf{n}}), ..., b_{1,L-1}(\vec{\mathbf{n}}), l_{1,L}(\vec{\mathbf{n}})\} \quad \text{and} \quad \{b_{2,0}(\vec{\mathbf{n}}), ..., b_{2,L-1}(\vec{\mathbf{n}}), l_{2,L}(\vec{\mathbf{n}})\}$$

- create blended pyramid $\{b_{0,0}(\vec{\mathbf{n}}), ..., b_{0,L-1}(\vec{\mathbf{n}}), l_{0,L}(\vec{\mathbf{n}})\}$ where

  $$\begin{aligned} b_j(\vec{\mathbf{n}}) &= b_{1,j}(\vec{\mathbf{n}})\, l_j(\vec{\mathbf{n}}) + b_{2,j}(\vec{\mathbf{n}})\,(1 - l_j(\vec{\mathbf{n}})) \\ l_L(\vec{\mathbf{n}}) &= l_{1,L}(\vec{\mathbf{n}})\, l_L(\vec{\mathbf{n}}) + l_{2,L}(\vec{\mathbf{n}})\,(1 - l_L(\vec{\mathbf{n}})) \end{aligned}$$

- collapse the pyramid $\{b_1, b_2, ..., l_L\}$ to obtain composite image



Uses of Laplacian Pyramid: Enhancement

Goal: Create a high fidelity image from a set of images take with different focal lengths, shutter speeds, etc.

- Images with different focal lengths will have different image regions in focus.
- Images with different shutter speeds may have different contrast and luminance levels in different regions.

Approach:

- Given pyramids for two images $I_1(\vec{\mathbf{n}})$ and $I_2(\vec{\mathbf{n}})$, construct 2 or 3 levels of a Laplacian pyramid:

  $$\{b_{1,0}(\vec{\mathbf{n}}), ..., b_{1,L-1}(\vec{\mathbf{n}}), l_{1,L}(\vec{\mathbf{n}})\} \quad \text{and} \quad \{b_{2,0}(\vec{\mathbf{n}}), ..., b_{2,L-1}(\vec{\mathbf{n}}), l_{2,L}(\vec{\mathbf{n}})\}$$

- at level j, define a mask $m(\vec{\mathbf{n}})$ that is 1 when $|b_{1,j}(\vec{\mathbf{n}})| > |b_{2,j}(\vec{\mathbf{n}})|$ and 0 elsewhere.

- then form the blended pyramid with levels $b_{0,j}[\vec{n}]$ given by

  $$b_{0,j}(\vec{\mathbf{n}}) \;=\; m(\vec{\mathbf{n}})\, b_{1,j}(\vec{\mathbf{n}}) \;+\; (1 - m(\vec{\mathbf{n}}))\, b_{2,j}(\vec{\mathbf{n}})$$

- average the low-pass bands from the two pyramids.

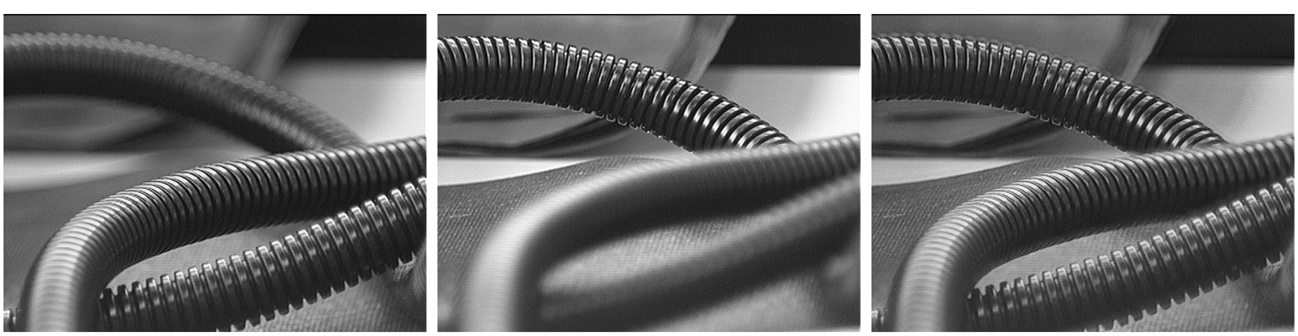


Image 1 Image 2 Composite

Further Readings

Books on Sections on Image Transforms:

Kenneth R Castleman, **Digital Image Processing**, Prentice Hall, 1995

Brian A Wandell, **Foundations of Vision**, Sinauer Press, 1995

Papers on Image Transforms and their Applications:

Peter J Burt and Edward H Adelson, ”A multiresolution spline with application to image mosaics.” *ACM Trans. on Graphics*, V. 2(4), 1983, pp. 217-236.

Peter J Burt and Edward H Adelson, ”The Laplacian pyramid as a compact image code.” *IEEE Trans. on Communications*, V. 31(4), 1983 pp. 532-540.

Eero P Simoncelli and Edward H Adelson, ”Subband transforms.” In **Subband Image Coding**, (ed.) John Woods. Kluwer Academic Publishers, Norwell, MA 1990.