Image Pyramids

Goal: Develop filter-based representations to decompose images into information at multiple scales, to extract features/structures of interest, and to attenuate noise.

Motivation:

- extract image features such as edges at multiple scales
- redundancy reduction and image modeling for
  - efficient coding
  - image enhancement/restoration
  - image analysis/synthesis

Examples:

- Gaussian Pyramid
- Laplacian Pyramid

Matlab Tutorials: imageTutorial.m and pyramidTutorial.m (up to line 200).

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 $\vec{\mathbf{I}}$:

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

Linear Transform Framework (cont)

Completeness

- the transform 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 of P^T

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

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

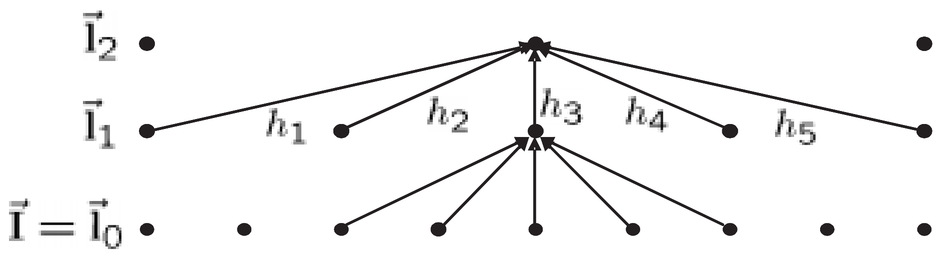
Self-Inverting

- the transform is self-inverting if $PP^T = \alpha I_N$ for some constant α. Here I_N is the $N \times N$ identity matrix. In this case, the basis matrix is simply $B = \frac{1}{\alpha} P$.
- in the critically sampled case the transform is orthogonal (unitary), up to the constant α.

Example. The Fourier transform is a critically sampled, complex-valued, self-inverting linear transform (remember to use the conjugate transpose $\mathbf{P}^{*T}$).

Gaussian Pyramid

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

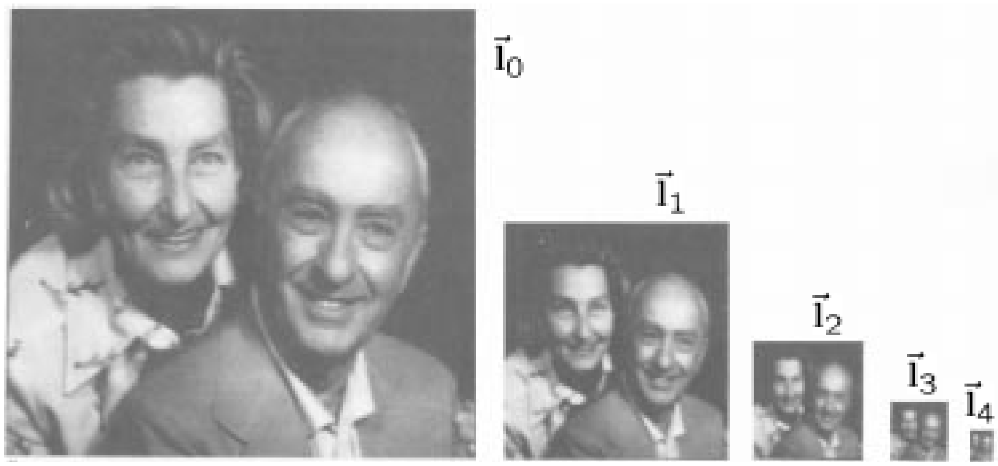
$$\vec{\mathbf{I}}_{k+1} \;=\; \mathbf{R}\vec{\mathbf{I}}_k \;=\; \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & \\ 0 & 0 & 1 & 0 & 0 & \cdots \\ 0 & 0 & 0 & 0 & 1 & \\ & & \vdots & & & \ddots \end{bmatrix}}_{\text{down-sampling}} \underbrace{\begin{bmatrix} \ddots & & & \\ & -\,\mathbf{h}\,- & & \\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

Typical weights for the impulse response from binomial weights

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

Gaussian Pyramid (cont)

Example of image and next four pyramid levels:



First three levels scaled to be the same size:

Properties of Gaussian pyramid:

- used for multi-scale edge estimation,
- efficient to compute coarse scale images. Only 5-tap 1D filter kernels are used,
- highly redundant, coarse scales provide much of the information in the finer scales.

Laplacian Pyramid

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

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

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

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

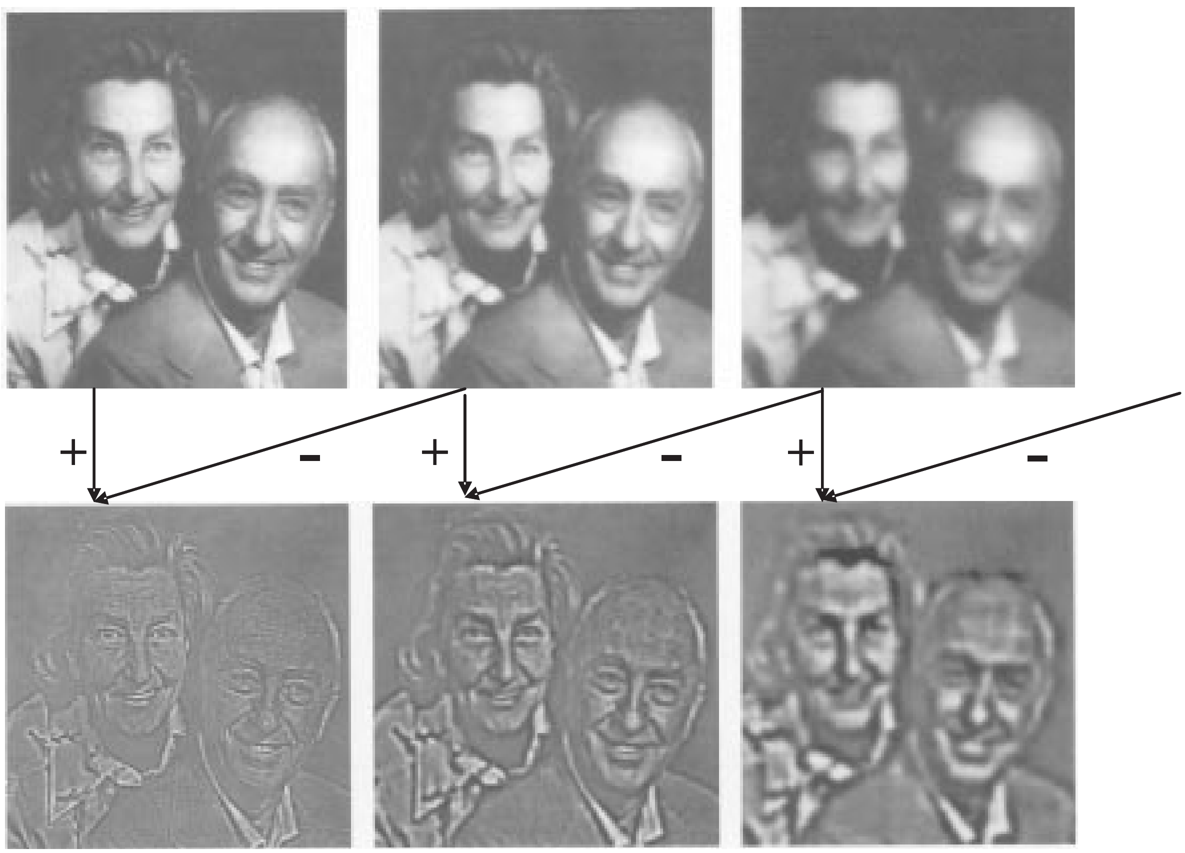
$$\mathbf{E}\,\vec{\mathbf{I}}_{k+1} = \underbrace{\begin{bmatrix} \ddots & & & & \\ & -\mathbf{g}- & & & \\ & & -\mathbf{g}- & & \\ & & & -\mathbf{g}- & \\ & & & & \ddots \end{bmatrix}}_{\text{convolution}} \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 & \\ 0 & 0 & 0 & 0 & \\ 0 & 1 & 0 & 0 & \cdots \\ 0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

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

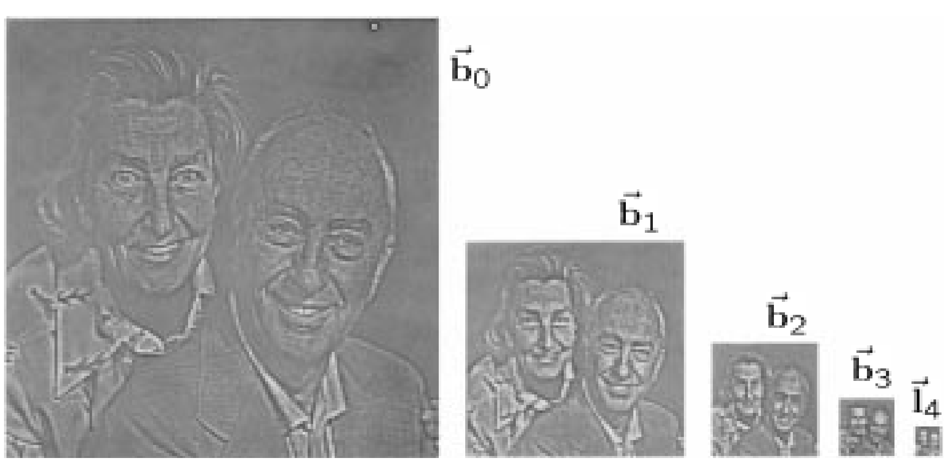
The **Laplacian pyramid** with L levels is given by $[\vec{\mathbf{b}}_0, \vec{\mathbf{b}}_1, ..., \vec{\mathbf{b}}_{L-1}, \vec{\mathbf{I}}_L]$. The representation is overcomplete by a factor of roughly $\frac{4}{3}$ for 2D images (i.e. $1 + 1/4 + 1/16 + ... = 4/3$).

Laplacian Pyramid (cont)

Construction of the Laplacian bands:



A Laplacian pyramid with four levels:



Laplacian Pyramid (cont)

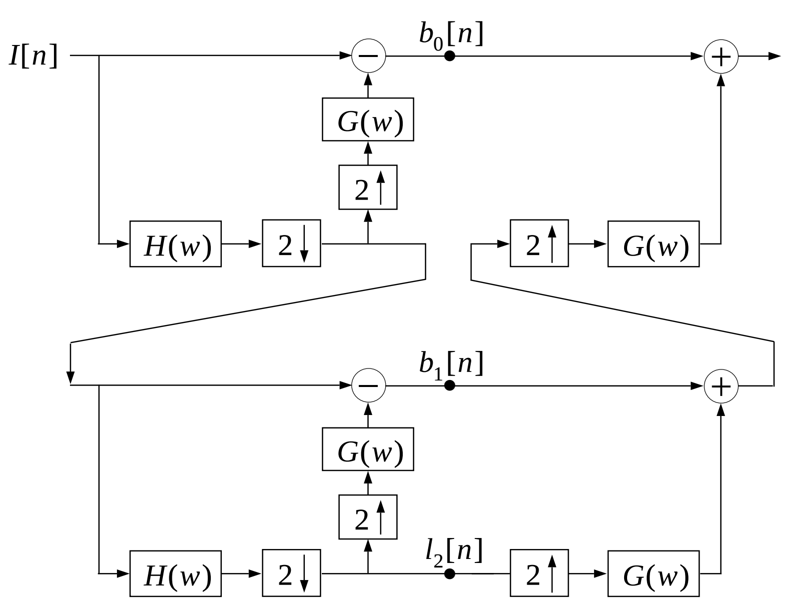
Construction: of $[\vec{\mathbf{b}}_0, \vec{\mathbf{b}}_1, ..., \vec{\mathbf{b}}_{L-1}, \vec{\mathbf{l}}_L]$.

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

Reconstruction: of $\vec{\mathbf{I}}$ is exact (for any filters) and straightforward:

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

System Diagram: shows the filters and sampling steps used to compute the pyramid, and to then reconstruct the image from the transform coefficients. Gaussian pyramid levels are computed using $h(n)$. Filter $g(n)$ 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 practive:

- often use same filters for h and g (apply same operators for smoothing and interpolation in construction and reconstruction)
- use separable lowpass filters
- desire isotropy 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. $\hat{h}(0) = 1$:

$$\hat{h}(0) \;=\; \sum_{n=-2}^{2} h(n)\, e^{-i\,0\,n} \;=\; a_0 + a_1 + a_2 \;=\; 1.$$

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

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

- So there are two linear equations for the three unknowns a_0, a_1, and a_2. There is therefore one free degree of freedom.
- For example, choose $a_0 = \frac{6}{16}$, then $h(n)$ is the binomial 5-tap filter

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

On the name "Laplacian"

The well-known Laplacian derivative operator (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, $g(x;\sigma) = \frac{1}{\sqrt{2\pi}\sigma} e^{-x^2/2\sigma^2}$,

$$\begin{aligned}
\frac{dg(x;\sigma)}{dx} &= \frac{-x}{\sigma^2} g(x;\sigma) \\
\frac{d^2 g(x;\sigma)}{dx^2} &= \left(\frac{x^2}{\sigma^2} - 1\right) \frac{1}{\sigma^2} g(x;\sigma) \\
\frac{dg(x;\sigma)}{d\sigma} &= \left(\frac{x^2}{\sigma^2} - 1\right) \frac{1}{\sigma} g(x;\sigma)
\end{aligned}$$

Therefore

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

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 scale σ.

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.
- this quantization matches human contrast sensitivity
- vast majority of the coefficients are in high frequency bands.

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

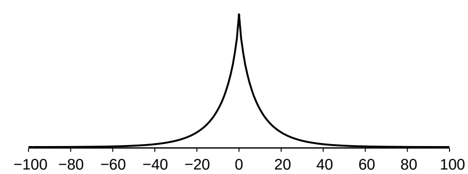
bits per pixel

Uses of Laplacian Pyramid: Restoration (Coring)

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

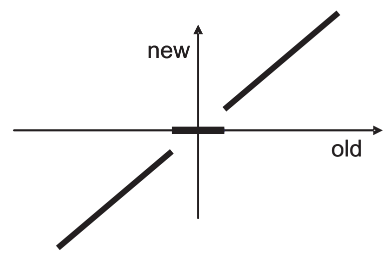
Histograms of transform coefficients are often well approximated by a so-called ”generalized Laplacian” density, $c\,e^{-|x/s|^k}$, where

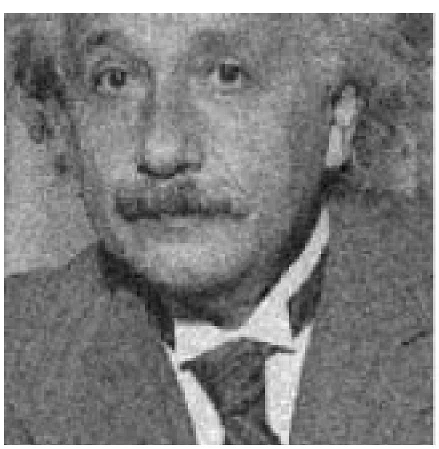
- γ is usually between 0.7 and 1.2
- s controls the variance
- peaked at 0, with heavy tails



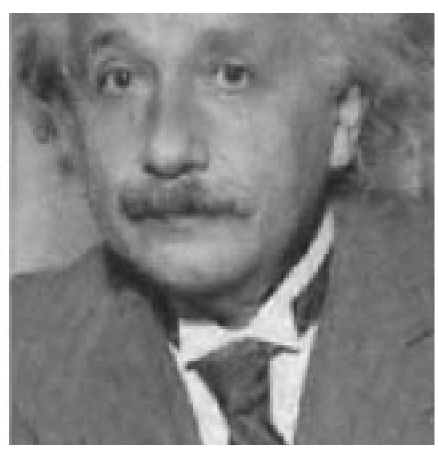
Coring:

- 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: Seamlessly stitch together images into an image mosaic (i.e., *register* the images and *blurring* the boundary), by smoothing the boundary in a scale-dependent way to avoid boundary aritfacts.

Method:

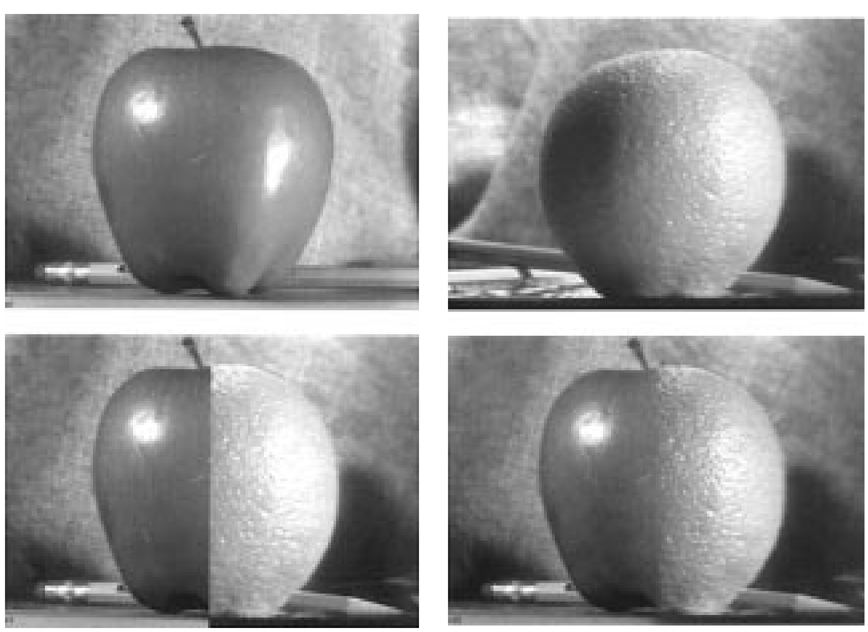
- assume images $I_1(\vec{\mathbf{n}})$ and $I_2(\vec{\mathbf{n}})$ are registered and let $m_1(\vec{\mathbf{n}})$ be a mask that is 1 at pixels where we want the brightness from $I_1(\vec{\mathbf{n}})$ and 0 otherwise (i.e., where we want to see $I_2(\vec{\mathbf{n}})$).
- create Gaussian pyramid for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

$$\begin{aligned} b_{0,j}(\vec{\mathbf{n}}) &= b_{1,j}(\vec{\mathbf{n}})\, l_j(\vec{\mathbf{n}}) + b_{2,j}(\vec{\mathbf{n}})\,(1 - l_j(\vec{\mathbf{n}})) \\ l_{0,L}(\vec{\mathbf{n}}) &= l_{1,L}(\vec{\mathbf{n}})\, l_L(\vec{\mathbf{n}}) + l_{2,L}(\vec{\mathbf{n}})\,(1 - l_L(\vec{\mathbf{n}})) \end{aligned}$$

- collapse blended pyramid to reconstruct 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 $I_1(\vec{n})$ and $I_2(\vec{n})$, 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

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

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


Image 1


Image 2


Composite