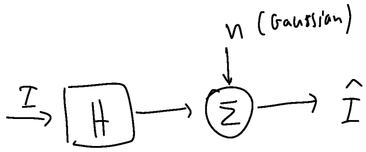
Noise

- sampling noise (aliasing)
- color quantization
- Noise model:



Signal to noise ratio

$$SNR = \frac{E_s}{E_n} = \frac{\sigma_s^2}{\sigma_n^2} = \frac{\frac{1}{n}\sum_{i,j}\left(I(i,j) - \bar{I}\right)^2}{\sigma_n^2}$$

σ_n^2 = variance for multiple frames of a static scene

OR

variance in a uniform image region

SNR [db]

$$SNR[db] = 10 \log_{10} \frac{E_s}{E_n}$$

10 db $\Rightarrow$ E_s is 10 times larger than E_n

13 db $\Rightarrow$ E_s is 20 times larger than E_n

3 db $\Rightarrow$ E_s is 2 times larger than E_n

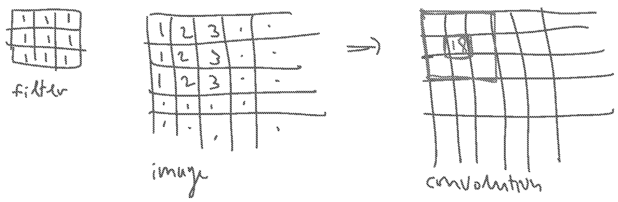
Noise filtering

- Remove noise using smoothing
- Smooth using convolution

filter dimensions

$$I_A(i,j) = I(i,j) * A(i,j) = \sum_{h=-\frac{m}{2}}^{\frac{m}{2}} \sum_{k=-\frac{n}{2}}^{\frac{n}{2}} A(h,k)\, I(i-h, j-k)$$

convolution



Convolution properties

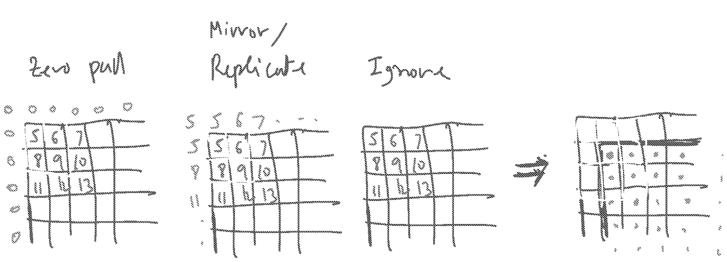
$$f * g = g * f$$

$$f * (g * h) = (f * g) * h$$

$$f * (g + h) = (f * g) + (f * h)$$

$$\frac{d}{dx}(f * g) = \frac{d}{dx} f * g = f * \frac{d}{dx} g$$

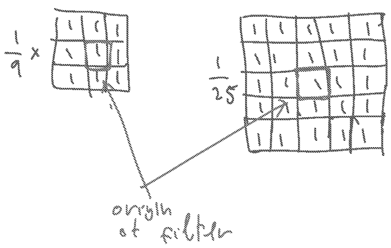
Convolution boundaries



- store result in new image
- store result in float array

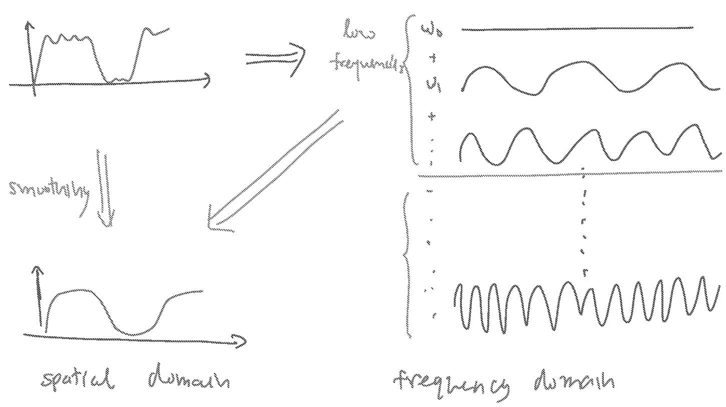
Smoothing using convolution

- convolution is a linear filter
- simple smoothing filter:



Low pass filter interpretation

Smoothing = removing high frequencies in image



Other applications of convolution

Blurring: $\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

Sharpening: $\frac{1}{9}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 18 & -1 \\ -1 & -1 & -1 \end{bmatrix}$

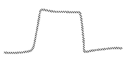
Vertical edge detection: $\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$

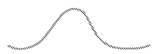
Horizontal edge detection: $\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Gaussian filter

* Give higher weight to pixels near the center

$$\frac{1}{25}\begin{bmatrix}1&1&1&1&1\\1&1&1&1&1\\1&1&\boxed{1}&1&1\\1&1&1&1&1\\1&1&1&1&1\end{bmatrix} \longrightarrow \frac{1}{273}\begin{bmatrix}1&4&7&4&1\\4&16&26&16&4\\7&26&\boxed{41}&26&7\\4&16&26&16&4\\1&4&7&4&1\end{bmatrix}$$





* 2D "Gaussian": $G_\sigma(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$

Separable implementation

$$I_G = I * G = \sum_i \sum_j I(i,j)\, e^{-\frac{i^2+j^2}{2\sigma^2}}$$

$$= \sum_i e^{-\frac{i^2}{2\sigma^2}} \sum_j I(i,j)\, e^{-\frac{j^2}{2\sigma^2}}$$

$$= (I * G_y) * G_x = I * G_x * G_y$$

Instead of convolving with a 2D Gaussian convolve with 1D Gaussian along rows then along columns

Complexity of separable implementation

$M \times N$ image

$m \times m$ filter

one 2D pass : $M \times N \times m^2$ operations

Two 1D passes : $2 \times M \times N \times m$ operations

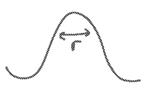
$$2MNm < MNm^2$$

Repeated application of Gaussians

$$I * G_{\sigma_1} * G_{\sigma_2} = I * G_{\sqrt{\sigma_1^2+\sigma_2^2}}$$

$$2 \times (M \times N \times m^2) < (M \times N \times (2m)^2)$$

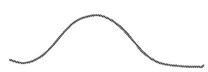
Selecting the Gaussian variance



68% of mass within $\pm\sigma$
95% of mass within $\pm 2\sigma$
99.7% of mass within $\pm 3\sigma$

filter width $m = 5\sigma \implies \sigma \leq \frac{m}{5}$

$G_\sigma(x) = e^{-\frac{x^2}{2\sigma^2}} \longrightarrow$



$\frac{1}{2.5}$ [0.14 | 0.61 | 1 | 0.61 | 0.14] $\sigma = 1$

$\frac{1}{38}$ [1 | 9 | 18 | 9 | 1] $\sigma = \frac{5}{6}$

$\frac{1}{16}$ [1 | 4 | 6 | 4 | 1] $\sigma = 1$

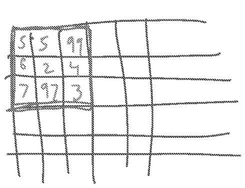
Median filter

$$I_{med}(i,j) = \text{median}\{I(x,y) \mid (x,y) \in win(i,j)\}$$

2, 3, 4, 5, 5, 6, 7, 97, 99

↑

median value = 5



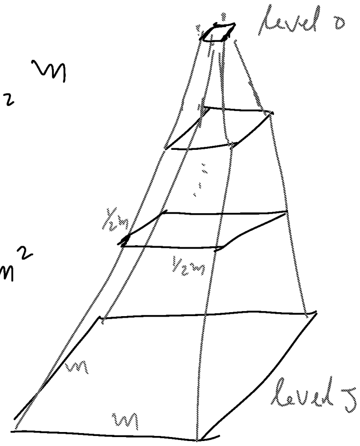
average = 25

Image pyramids

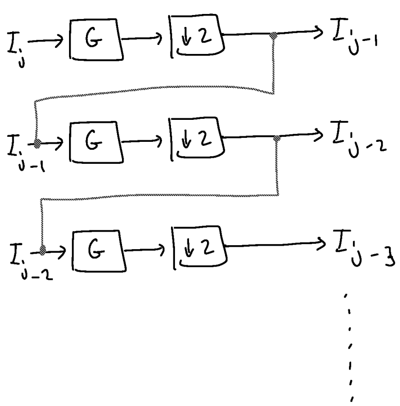
$$m = 2^J \Rightarrow J = \log_2 m$$

total # pixels =

$$m^2 + \frac{1}{4}m^2 + \frac{1}{16}m^2 + \cdots < \frac{4}{3}m^2$$



Gaussian pyramids



Pyramid:

I_J

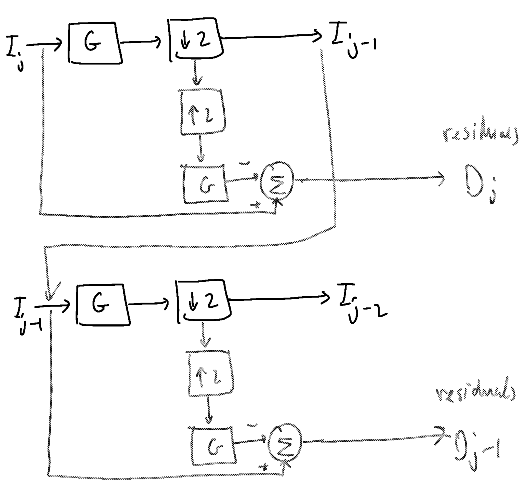
I_{J-1}

$\vdots$

I_0

useful for analysis

Laplacian pyramids



Pyramid:

D_J

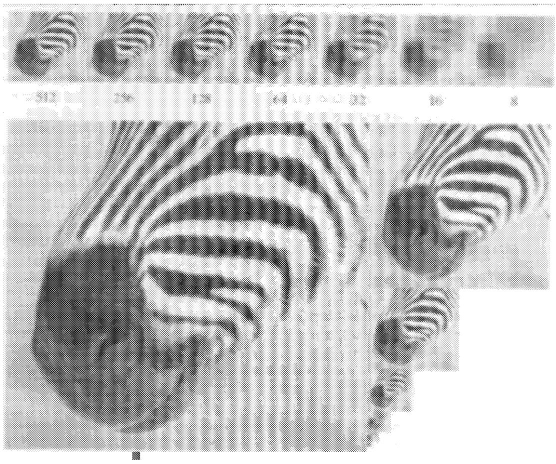
D_{J-1}

$\vdots$

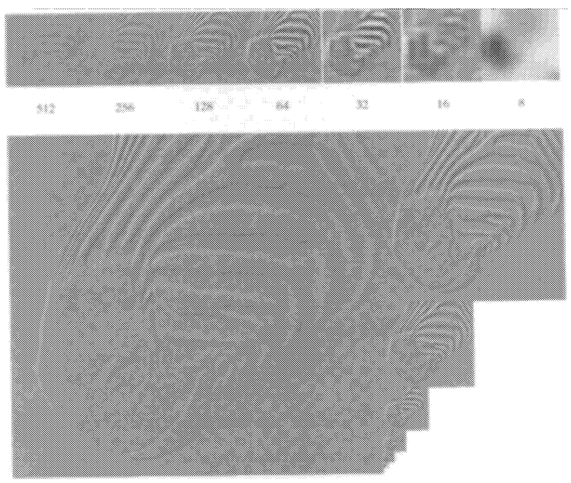
I_0

useful for compression (small residuals)

Example (Gaussian pyramid)



Example (Laplacian pyramid)



Color distribution in Gaussian and Laplacian pyramids