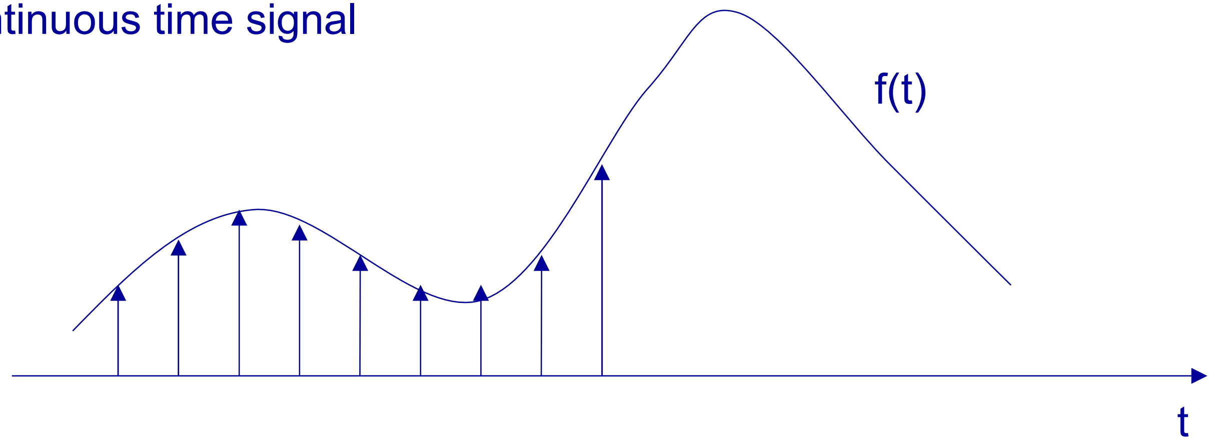
# Sampling in 2D

# Sampling in 1D

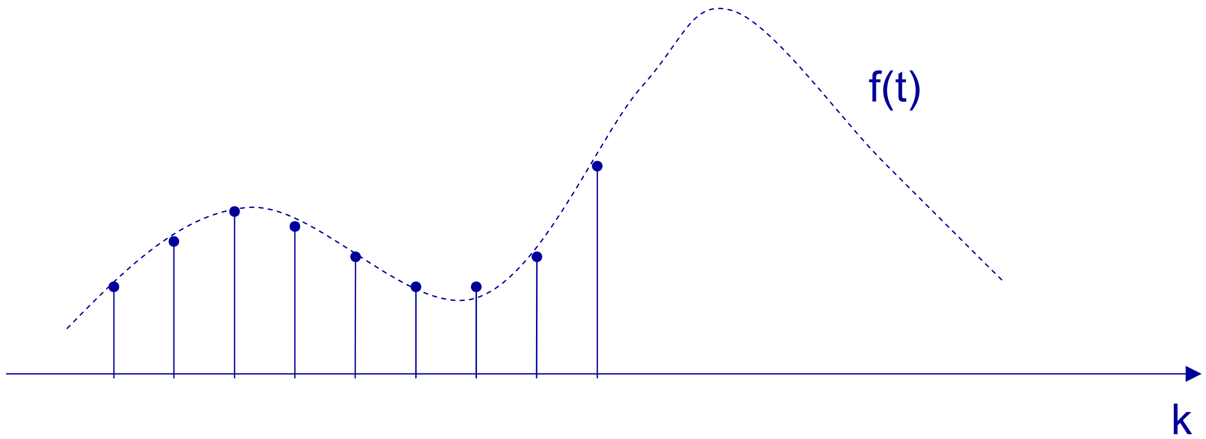
Continuous time signal



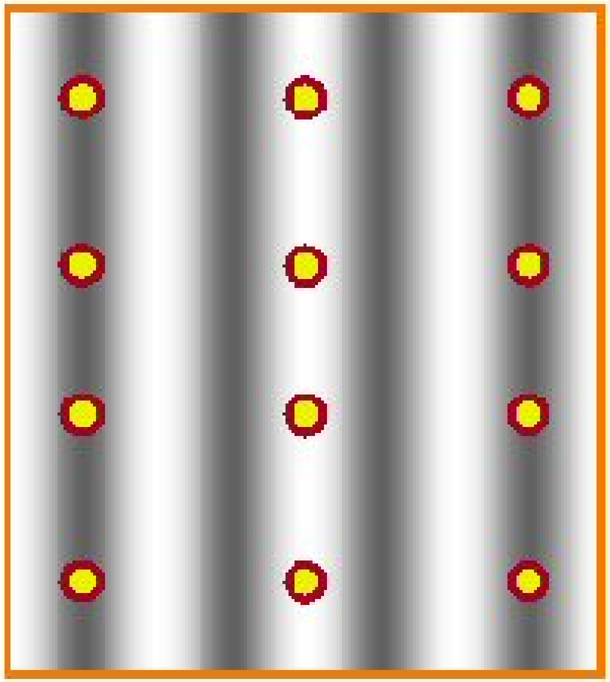

Discrete time signal

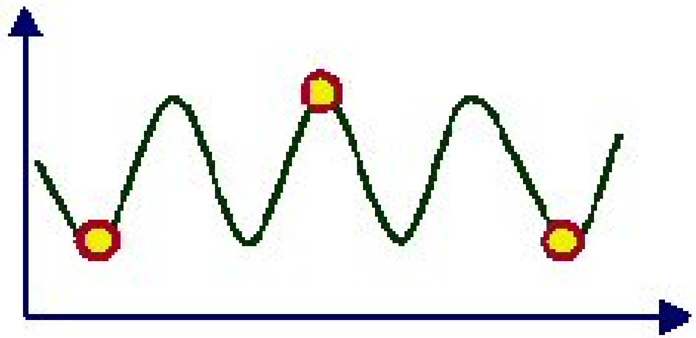
$$f[k] = f(kT_s) = f(t)\sum_k \delta(t - kT_s)$$

comb



# Nyquist theorem (1D)





At least 2 sample/period are needed to represent a periodic signal

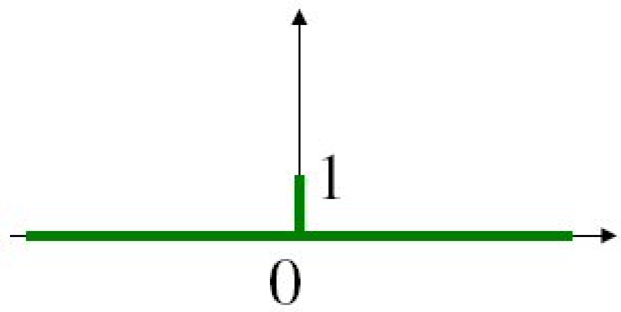
$$T_s \leq \frac{1}{2}\frac{2\pi}{\omega_{\max}}$$

$$\omega_s = \frac{2\pi}{T_s} \geq 2\omega_{\max}$$

# Delta pulse

- 1D Dirac pulse

$$\begin{cases} \delta(x) = 1 \text{ if } x=0 \\ \delta(x) = 0 \text{ else} \end{cases}$$

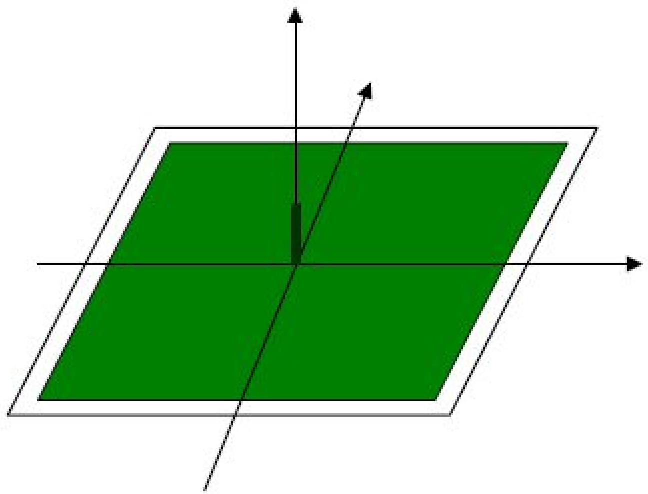



- 2D Dirac pulse

$$\begin{cases} \delta(x,y) = 1 \text{ if } x=0 \text{ and } y=0 \\ \delta(x,y) = 0 \quad \text{else} \end{cases}$$

which corresponds to :

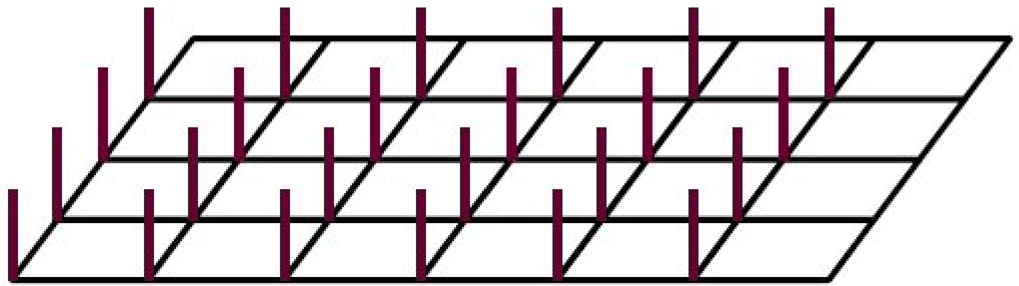
$$\delta(x,y) = \delta(x)\,\delta(y)$$



# Dirac *brush*

- 1D sampling: Dirac comb (or Shah function)

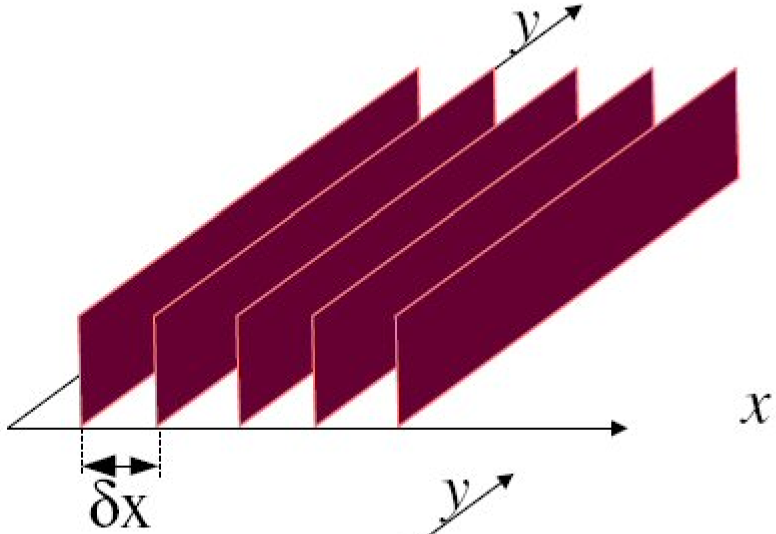
- 2D sampling : Dirac « brush »



# Comb

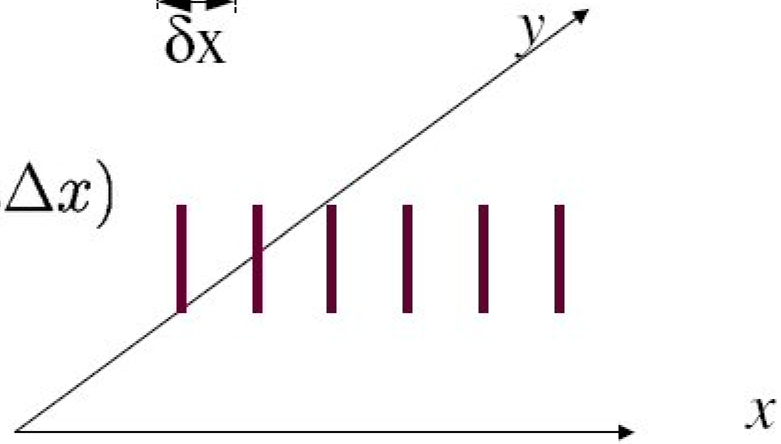
- Extended comb :

$$p_x(x,y) = \sum_{m=-\infty}^{\infty} \delta(x - m\Delta x)$$




- Comb :

$$p_x(x,y) = \delta(y) \sum_{m=-\infty}^{\infty} \delta(x - m\Delta x)$$



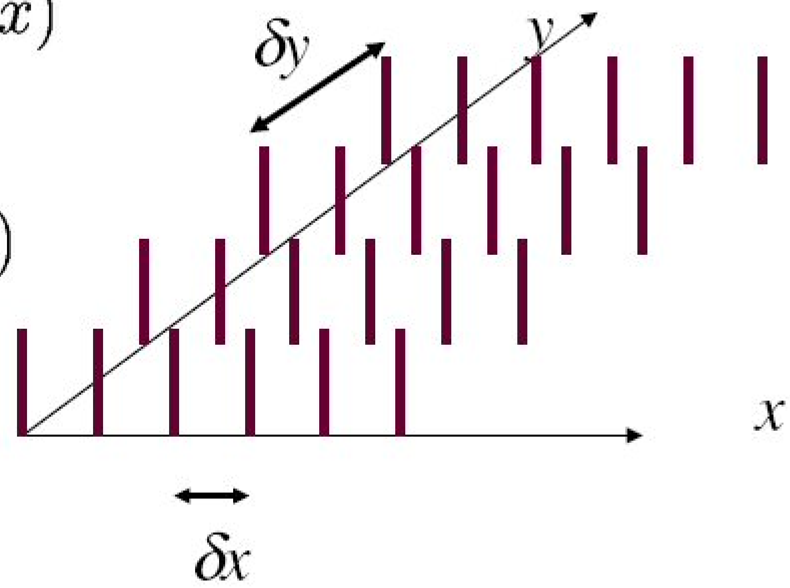
# Brush

- Brush = product of 2 extended combs

$$p_x(x,y) = \sum_{m=-\infty}^{\infty} \delta(x - m\Delta x)$$

$$p_y(x,y) = \sum_{m=-\infty}^{\infty} \delta(y - n\Delta y)$$

$$b(x,y) = p_x(x,y)p_y(x,y)$$



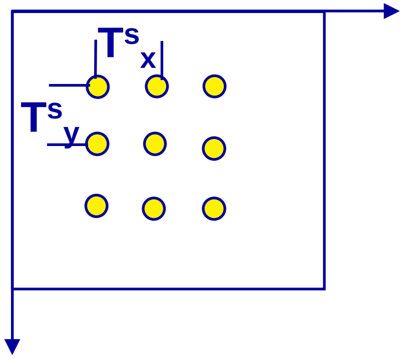
# Nyquist theorem

- Sampling in p-dimensions

$$s_T(\vec{x}) = \sum_{k \in Z^p} \delta(\vec{x} - kT)$$

$$f_T(\vec{x}) = f(\vec{x}) s_T(\vec{x})$$

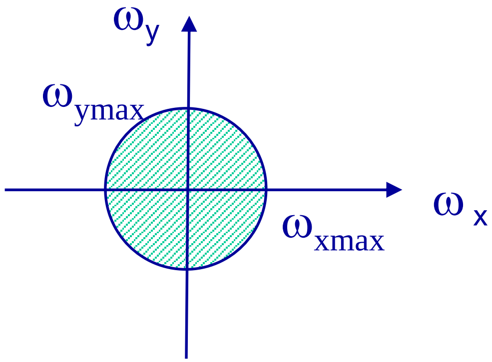
2D spatial domain



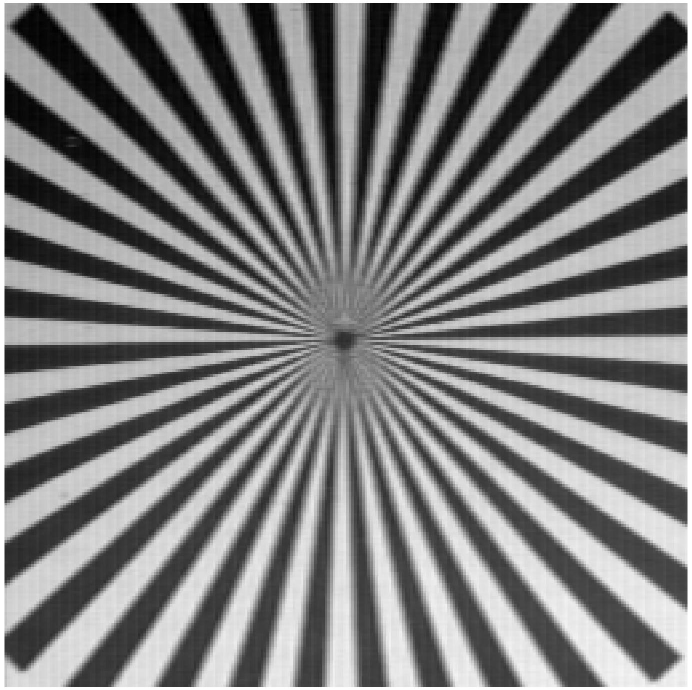

- Nyquist theorem

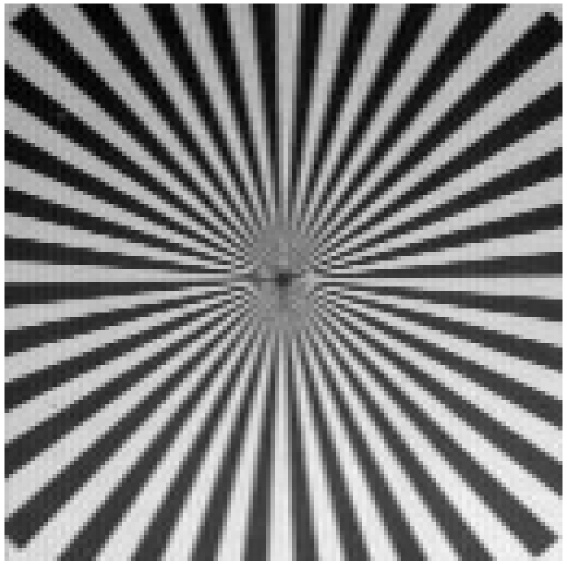
$$\begin{cases} \omega_x^s \geq 2\omega_{x\max} \\ \omega_y^s \geq 2\omega_{y\max} \end{cases} \Rightarrow \begin{cases} T_x^s \leq 2\pi \dfrac{1}{2\omega_{x\max}} \\ T_y^s \leq 2\pi \dfrac{1}{2\omega_{y\max}} \end{cases}$$

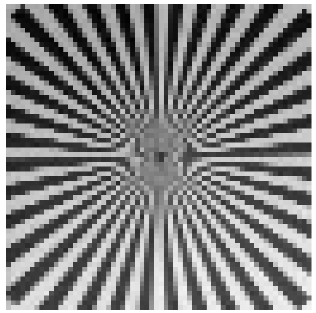
2D Fourier domain



# Spatial aliasing



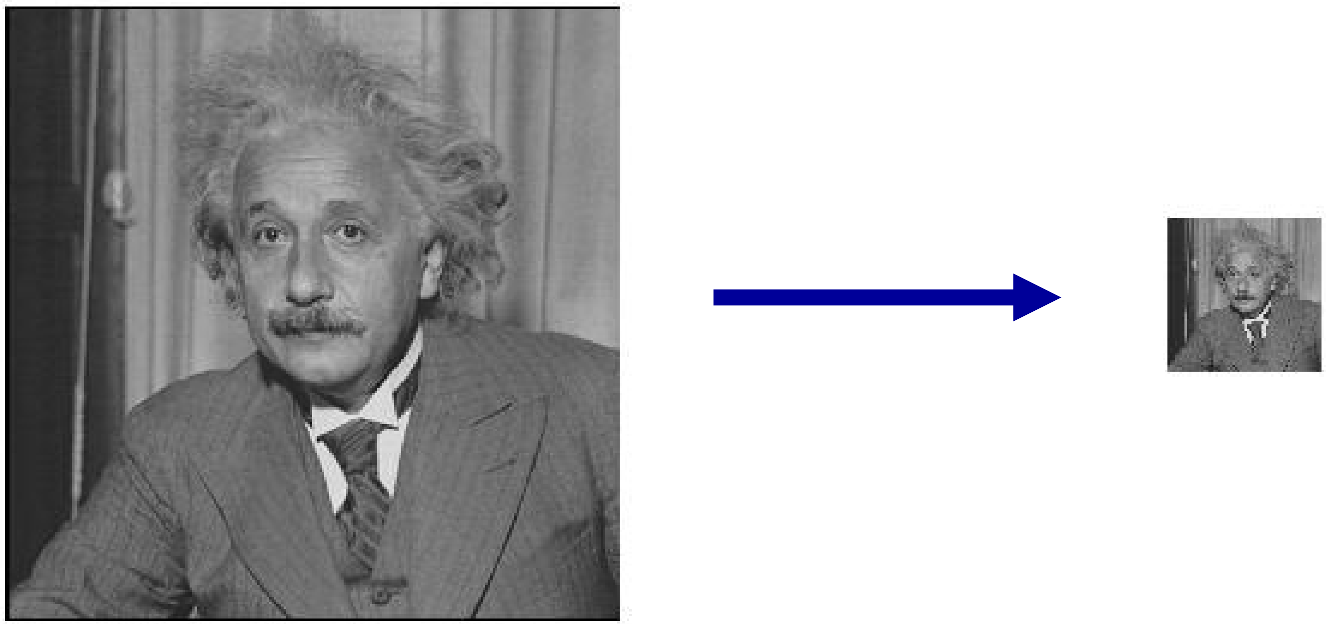




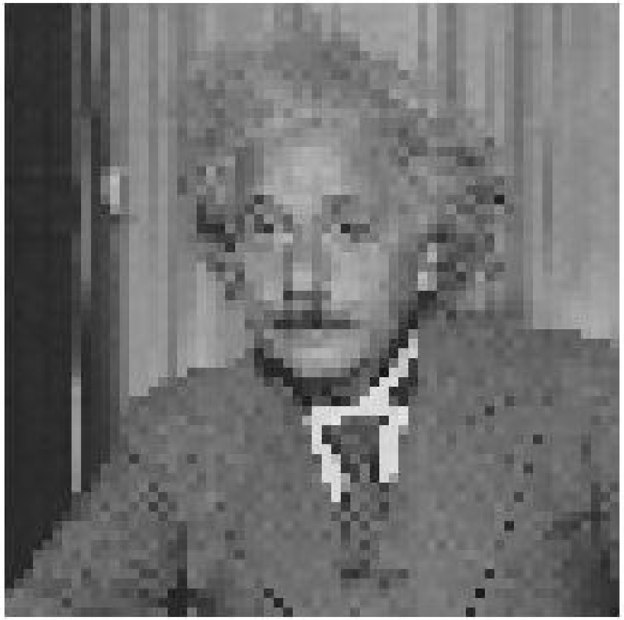
# Resampling

- Change of the sampling rate
  - Increase of sampling rate: Interpolation or upsampling
    - Blurring, low visual resolution
  - Decrease of sampling rate: Rate reduction or downsampling
    - Aliasing and/or loss of spatial details

# Downsampling

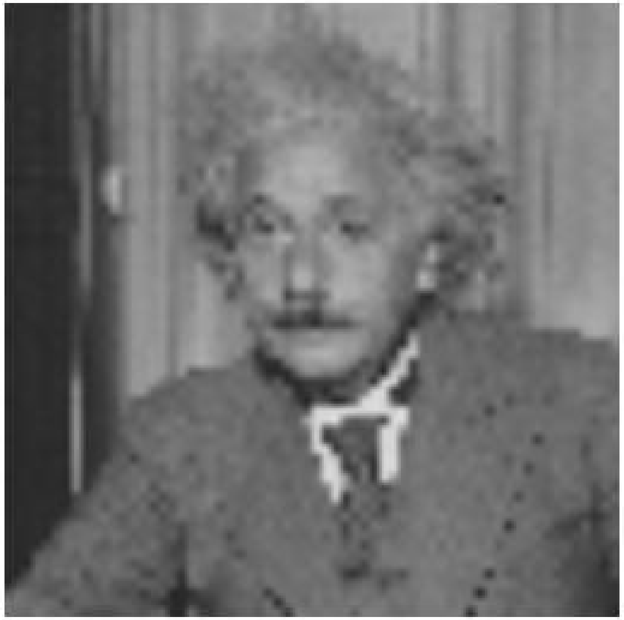


# Upsampling



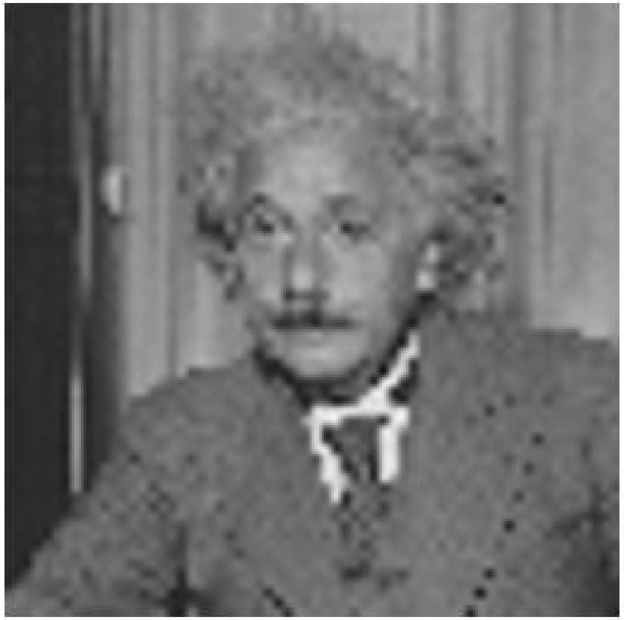
nearest neighbor (NN)

# Upsampling



bilinear

# Upsampling



bicubic

# Quantization

# Scalar quantization

- A scalar quantizer $Q$ approximates $X$ by $\tilde{X}=Q(X)$, which takes its values over a finite set.
- The quantization operation can be characterized by the MSE between the original and the quantized signals

$$d = E\{(X-\tilde{X})^2\}.$$

- Suppose that $X$ takes its values in $[a, b]$, which may correspond to the whole real axis. We decompose $[a, b]$ in $K$ intervals $\{(y_{k-1}, y_k]\}_{1\leq k\leq K}$ of variable length, with $y_0=a$ and $y_K=b$.
- A scalar quantizer approximates all $x \in (y_{k-1}, y_k]$ by $x_k$:

$$\forall x \in (y_{k-1}, y_k], \qquad Q(x) = x_k$$

# Scalar quantization

- The intervals $(y_{k-1}, y_k]$ are called *quantization bins*.
- Rounding off integers is an example where the quantization bins

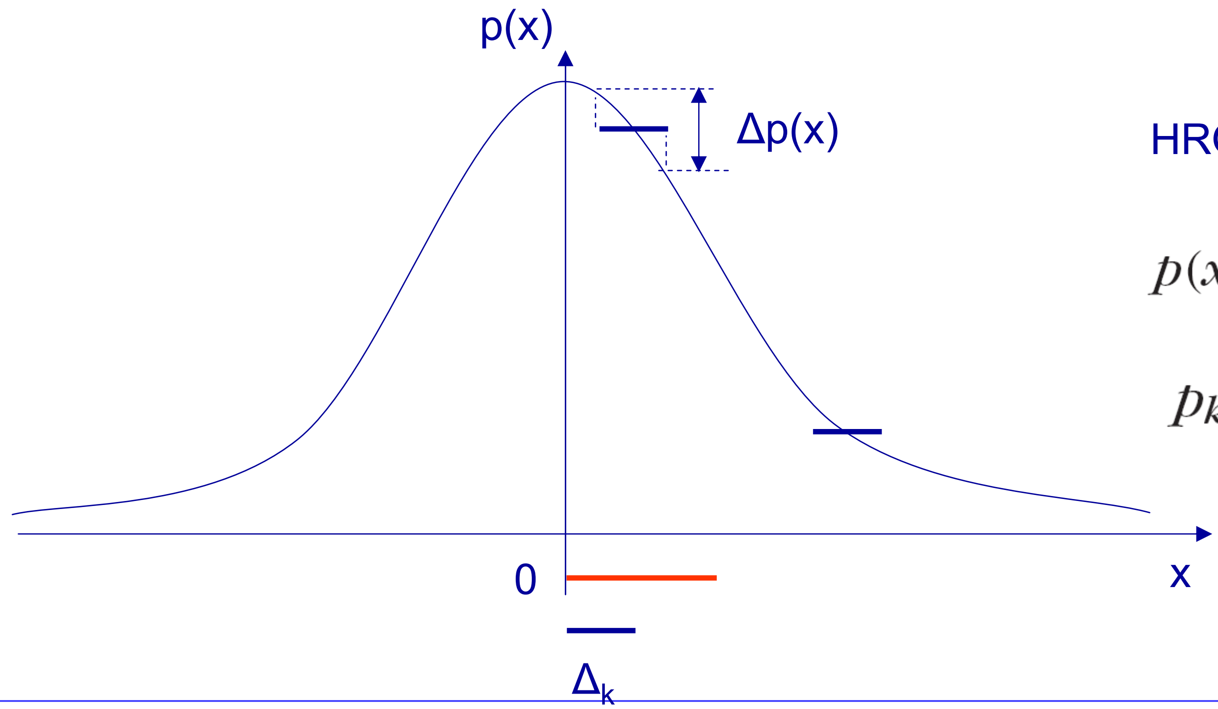
  $$(y_{k-1}, y_k] = (k-1/2, k+1/2]$$

  have size 1and $x_k = k$ for any $k \in Z$.
- qui
- High resolution quantization
  - Let $p(x)$ be the probability density of the random source $X$. The mean-square quantization error is

$$d = E\{(X - \tilde{X})^2\} = \int_{-\infty}^{+\infty} \left(x - Q(x)\right)^2 p(x)\, dx.$$

# HRQ

– A quantizer is said to have a *high resolution* if *p(x)* is *approximately constant* on each quantization bin. This is the case if the sizes *k* are sufficiently small relative to the rate of variation of *p(x)*, so that one can neglect these variations in each quantization bin.




HRQ: $\Delta p(x) \rightarrow 0$

$$p(x) = \frac{p_k}{\Delta_k} \quad \text{for } x \in (y_{k-1}, y_k],$$

$$p_k = \Pr\{X \in (y_{k-1}, y_k]\}.$$

# Scalar quantization

- Teorem 10.4 (Mallat): For a high-resolution quantizer, the mean-square error $d$ is minimized when $x_k=(y_k+y_{k+1})/2$, which yields

$$d = \frac{1}{12}\sum_{k=1}^{K} p_k \Delta_k^{\ 2}$$

**Proof.** The quantization error (10.15) can be rewritten as

$$d = \sum_{k=1}^{K} \int_{y_{k-1}}^{y_k} (x - x_k)^2 \, p(x) \, dx.$$

Replacing $p(x)$ by its expression (10.16) gives

$$d = \sum_{k=1}^{K} \frac{p_k}{\Delta_k} \int_{y_{k-1}}^{y_k} (x - x_k)^2 \, dx. \qquad \textbf{(10.18)}$$

One can verify that each integral is minimum for $x_k = (y_k + y_{k-1})/2$, which yields (10.17). ■

# Uniform quantizer

The uniform quantizer is an important special case where all quantization bins have the same size

$$y_k - y_{k-1} = \Delta \quad \text{for} \quad 1 \leqslant k \leqslant K.$$

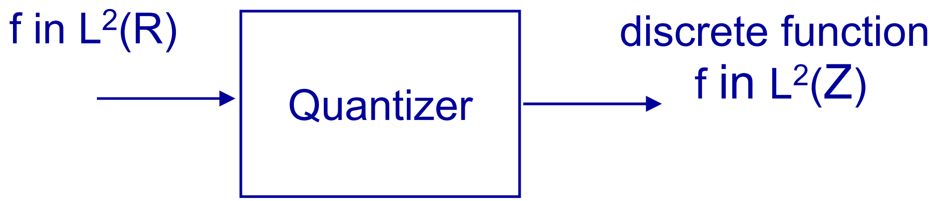
For a high-resolution uniform quantizer, the average quadratic distortion (10.17) becomes

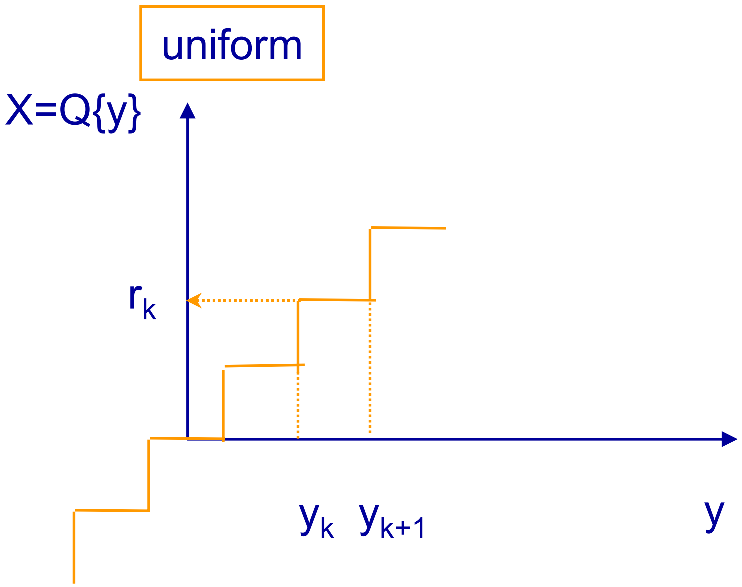
$$d = \frac{\Delta^2}{12} \sum_{k=1}^{K} p_k = \frac{\Delta^2}{12}. \qquad \textbf{(10.19)}$$

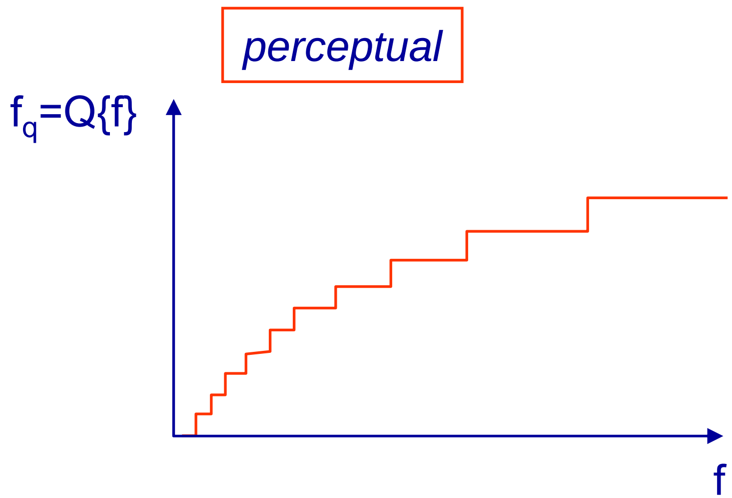
It is independent of the probability density $p(x)$ of the source.

# Quantization

- A/D conversion ⇨ quantization



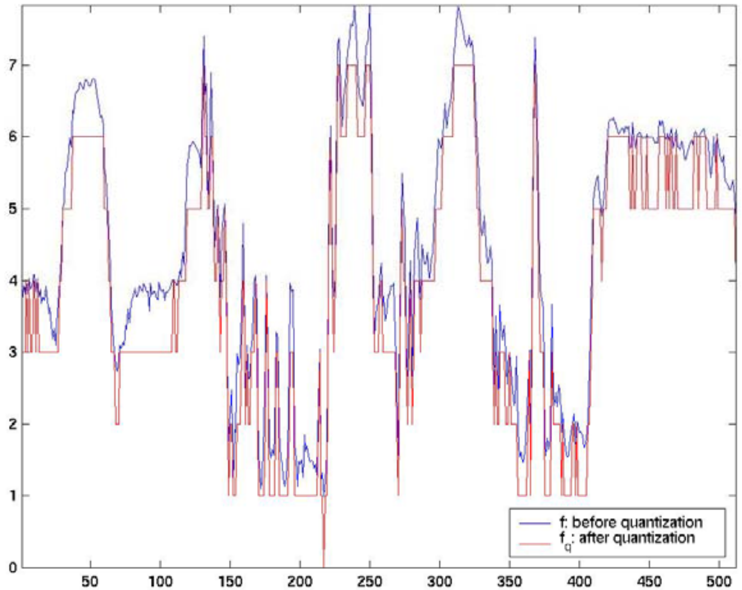




The sensitivity of the eye decreases increasing the background intensity (Weber law)

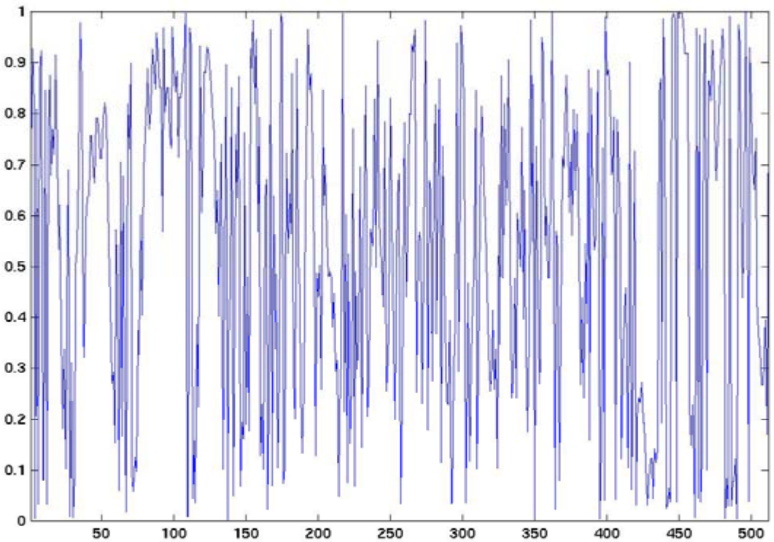
# Quantization

Signal before (blue) and after quantization (red) Q




Equivalent noise: $n=f_q - f$

additive noise model: $f_q=f+n$



# Quantization

original

5 levels

10 levels

50 levels

# Distortion measure

- Distortion measure

$$D = \mathrm{E}\left[(f_Q - f)^2\right] = \sum_{k=0}^{K} \int_{t_k}^{t_{k+1}} (f_Q - f)^2 \, p(f)\, df$$

  - The distortion is measured as the expectation of the mean square error (MSE) difference between the original and quantized signals.

$$PSNR = 20\log_{10}\frac{255}{MSE} = 20\log_{10}\frac{255}{\frac{1}{N\times M}\sqrt{\sum_{i=1}^{N}\sum_{j=1}^{M}\left(I_1[i,j]-I_2[i,j]\right)^2}}$$

- Lack of correlation with perceived image quality
  - Even though this is a very natural way for the quantification of the quantization artifacts, it is not representative of the *visual annoyance* due to the majority of common artifacts.
- Visual models are used to define perception-based image quality assessment metrics

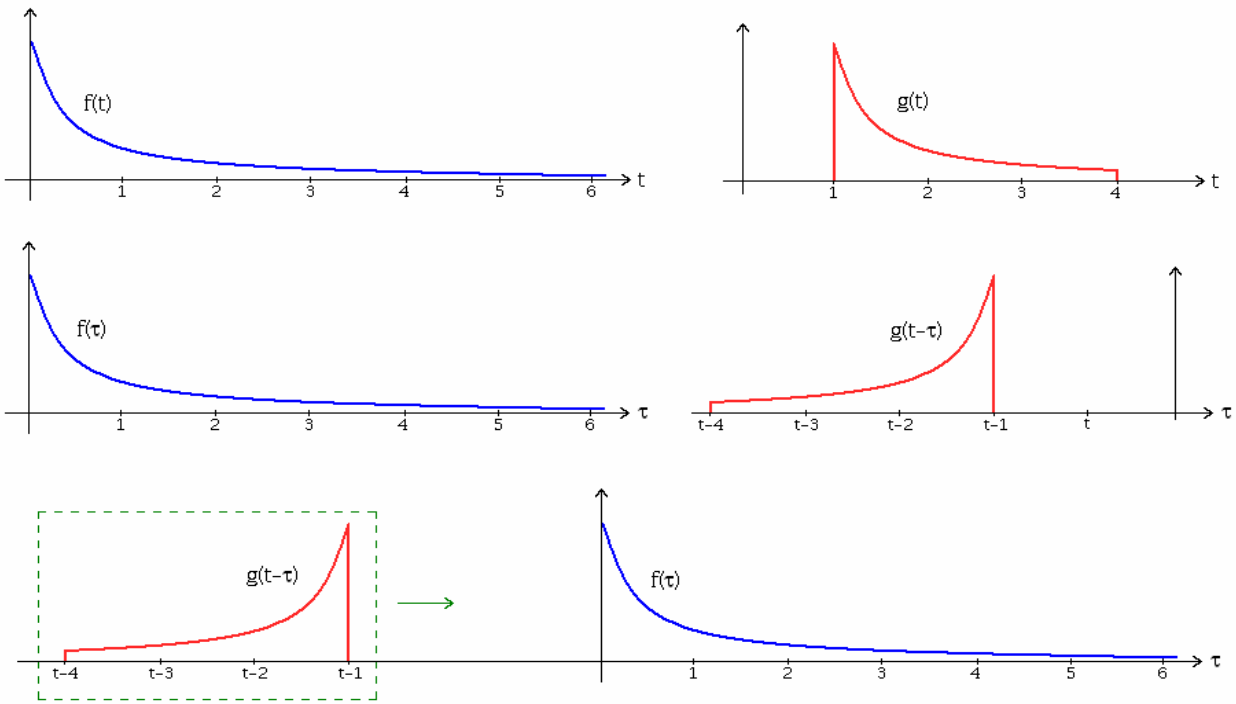
# Example

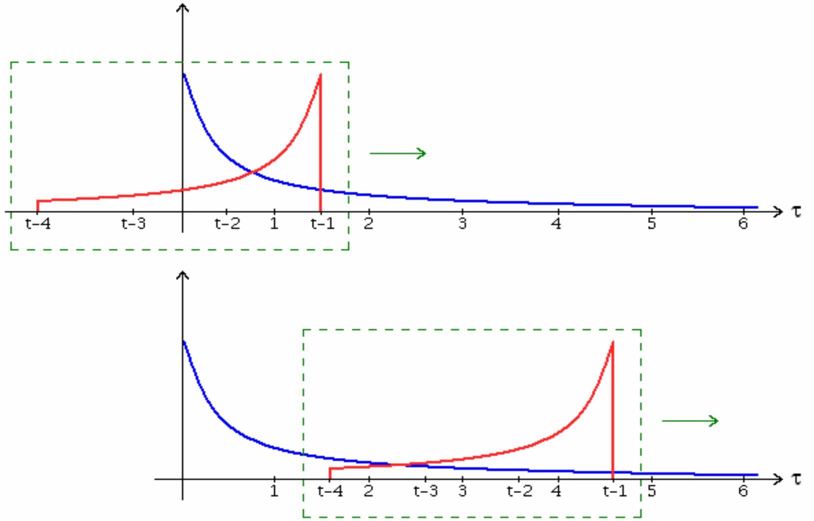
- The PSNR does not allow to distinguish among different types of distortions leading to the same RMS error between images

- The MSE between images (b) and (c) is the same, so it is the PSNR. However, the visual annoyance of the artifacts is different

# Convolution

# Convolution



$$c(t) = f(t) * g(t) = \int_{-\infty}^{+\infty} f(\tau) g(t-\tau) d\tau$$

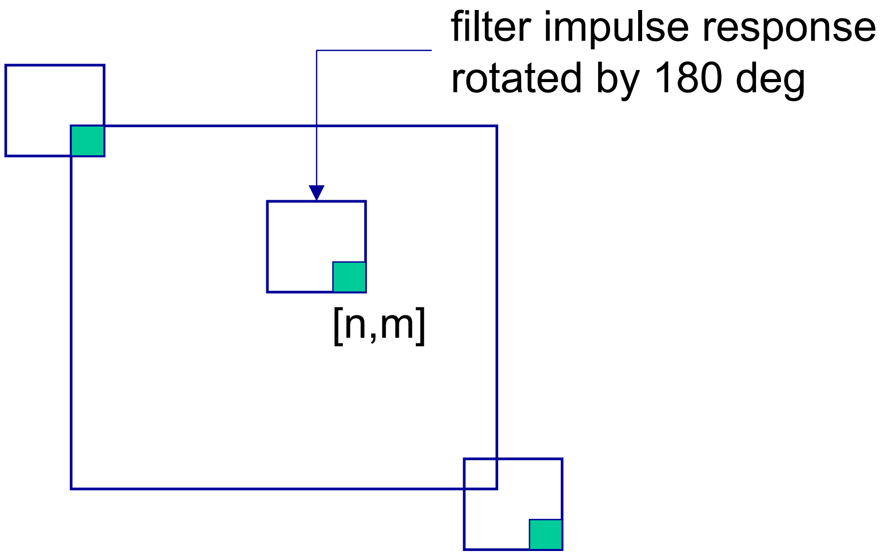
$$c[n] = f[n] * g[n] = \sum_{k=-\infty}^{+\infty} f[k] g[k-n]$$

# 2D Convolution

$$c(x,y) = f(x,y) \otimes g(x,y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(\tau,\nu) g(x-\tau, y-\nu) d\tau d\nu$$

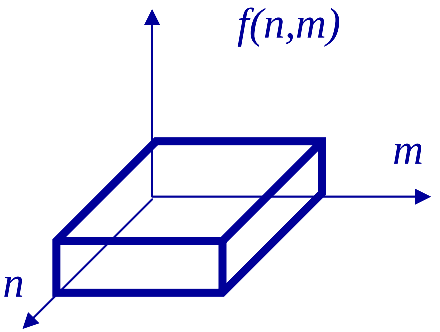
$$c[i,k] = \sum_{n=-\infty}^{+\infty} \sum_{m=-\infty}^{+\infty} f[n,m] g[i-n, k-m]$$

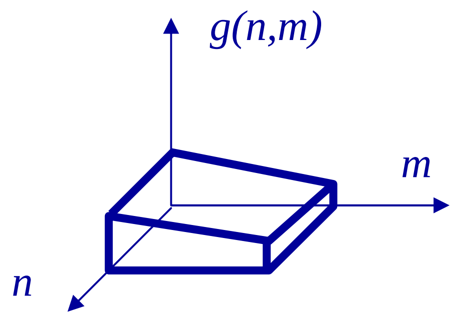
- Associativity
- Commutativity
- Distributivity



# 2D Convolution

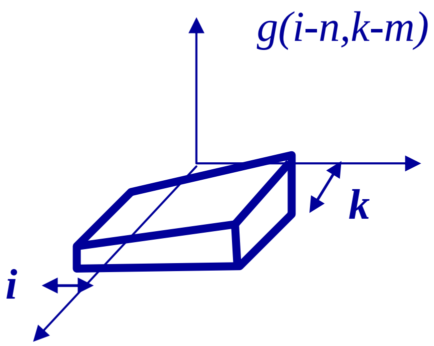
$$c[i,k] = \sum_{n=-\infty}^{+\infty} \sum_{m=-\infty}^{+\infty} f[n,m]g[i-n,k-m]$$

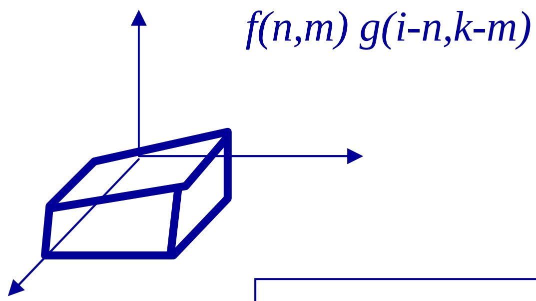



1. fold about origin
2. displace by ‘i’ and ‘k’

3. compute integral
   of the box




Tricky part: borders

- *(zero padding, mirror...)*

# Convolution

Filtering with filter h(x,y)

$$f_2(x, y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f_1(s, t)h(x - s, y - t)dsdt$$

- Convolution with a 2D Dirac pulse

  $f_2(x, y) = f_1(x, y)$ sampling property of the delta function

- Convolution a Dirac pulse shifted by $(x_0, y_0)$

  $f_2(x, y) = f_1(x - x_0, y - y_0)$

- Fourier transform…

  $F_2(u, v) = F_1(u, v)\ H(u, v)$

- … and vice versa

  $g(x,y) = f_1(x, y)\ f_2(x, y)$ then $G(u,v) = F_1(u, v) * F_2(u, v)$

# Convolution

- Convolution is a *neighborhood operation* in which each output pixel is the *weighted sum* of neighboring input pixels. The matrix of weights is called the *convolution kernel*, also known as the *filter*.
  - A convolution kernel is a correlation kernel that has been rotated 180 degrees.
- Recipe
  1. Rotate the convolution kernel 180 degrees about its center element.
  2. Slide the center element of the convolution kernel so that it lies on top of the (I,k) element of f.
  3. Multiply each weight in the rotated convolution kernel by the pixel of f underneath. Sum the individual products from step 3

  - zero-padding is generally used at borders but other border conditions are possible

# Example

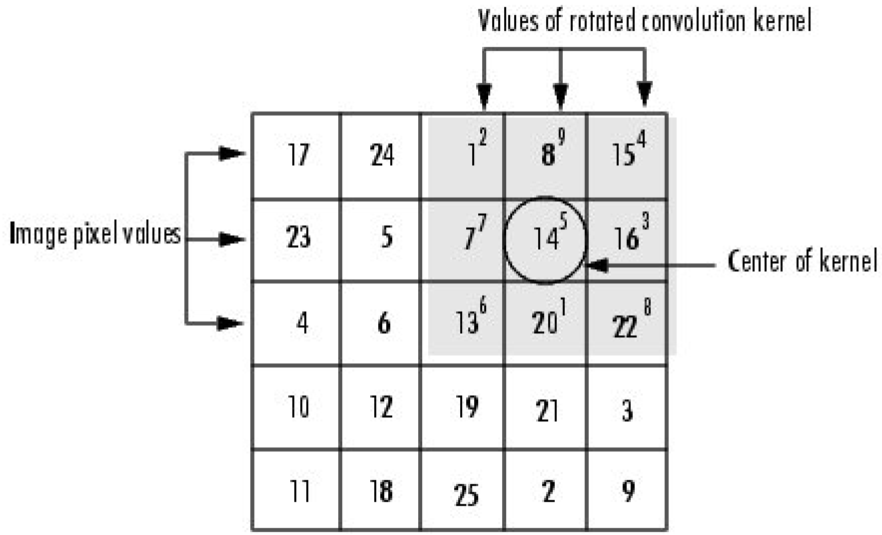
kernel

f = [17 24 1 8 15
23 5 7 14 16
4 6 13 20 22
10 12 19 21 3
11 18 25 2 9]

h = [8 1 6
3 5 7
4 9 2]

h'= [2 9 4
7 5 3
6 1 8]

$$1 \cdot 2 + 8 \cdot 9 + 15 \cdot 4 + 7 \cdot 7 + 14 \cdot 5 + 16 \cdot 3 + 13 \cdot 6 + 20 \cdot 1 + 22 \cdot 8 = 575$$




**Computing the (2,4) Output of Convolution**

# Correlation

- The operation called correlation is closely related to convolution. In correlation, the value of an output pixel is also computed as a weighted sum of neighboring pixels.
- The difference is that the matrix of weights, in this case called the correlation kernel, *is not rotated* during the computation.
- Recipe
  1. Slide the center element of the correlation kernel so that lies on top of the (2,4) element of f.
  2. Multiply each weight in the correlation kernel by the pixel of A underneath.
  3. Sum the individual products from step 2.

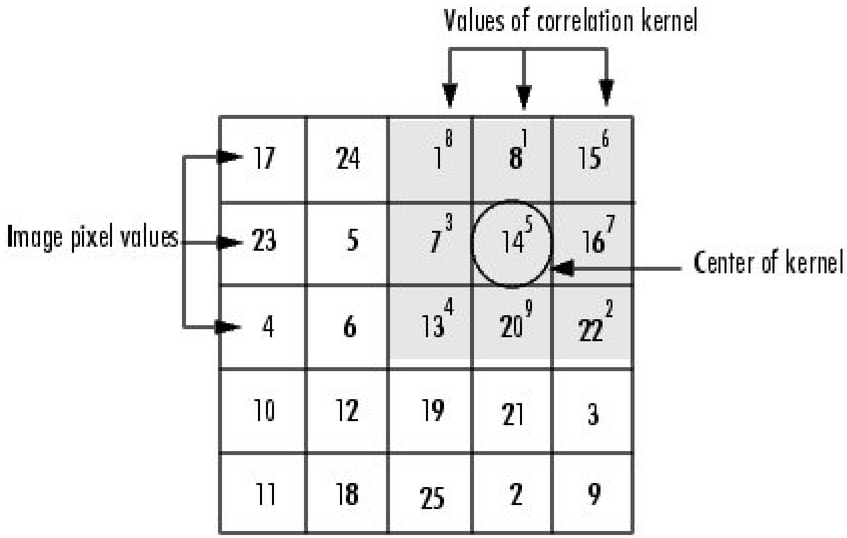
# Example

kernel

f = [17 24 1 8 15
23 5 7 14 16
4 6 13 20 22
10 12 19 21 3
11 18 25 2 9]

h = [8 1 6
3 5 7
4 9 2]

$$1 \cdot 8 + 8 \cdot 1 + 15 \cdot 6 + 7 \cdot 3 + 14 \cdot 5 + 16 \cdot 7 + 13 \cdot 4 + 20 \cdot 9 + 22 \cdot 2 = 585$$




Computing the (2,4) Output of Correlation