- Many operations are easier, both computationally and conceptually, in the frequency domain.
- We transform the time domain signal to the frequency domain, perform the operation of interest there, and then transform the altered signal back to the time domain.

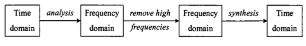
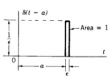


FIGURE 2.1 Processing a signal in the frequency domain.

- Também conhecida como função Dirac ou função delta
- Modela a propriedade de uma fonte puntual tendo largura infinitesimal e área (volume) unitária

$$\delta(x-x_0)=0 \quad \text{for } x \neq x_0,$$
$$\int_{-\infty}^{+\infty}\delta(x-x_0)\,dx = 1,$$



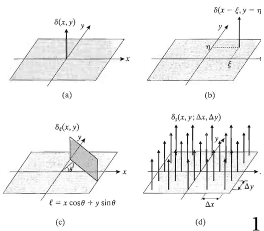


Figure 2.3 Signals derived from the point impulse: (a) point impulse $\delta(x,y)$, (b) shifted point impulse $\delta(x-\xi, y-\eta)$, (c) line impulse $\delta_\ell(x,y)$, and (d) sampling function $\delta_s(x,y;\Delta x,\Delta y)$.

sifting let $s(x)$ be continuous at $x = x_0$, then

$$\int_{-\infty}^{+\infty} s(x)\,\delta(x-x_0)\,dx = s(x_0);$$

scaling

$$\int_{-\infty}^{+\infty} A\,\delta(x)\,dx =$$

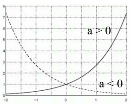
this is a special case of sifting.

$$\iint_{-\infty}^{+\infty} s(\vec{r})\,\delta(\vec{r}-\vec{r}_0)\,d\vec{r} = s(\vec{r}_0).$$

Exponential *exp*

$\exp(ax) = e^{ax}$

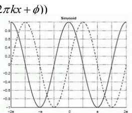
- when *a* > 0 then *exp* increases with increasing *x*
- when *a* < 0 then *exp* approximates 0 with increasing *x*



Complex exponential / sinusoid:

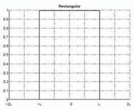
$$Ae^{i(2\pi kx+\phi)} = A(\cos(2\pi kx+\phi) + i\sin(2\pi kx+\phi))$$

As before
- the *cos* term is the signal's real part
- the *sin* term is the signal's imaginary part
- *A* is the amplitude, φ the phase shift, *k* determines the frequency



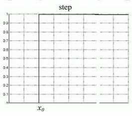
Rectangular function:

$$\Pi(\frac{x}{2L}) = 1 \quad \text{for } |x| < L$$
$$= \frac{1}{2} \quad \text{for } |x| = L$$
$$= 0 \quad \text{for } |x| > L$$



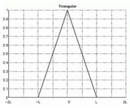
Step function:

$$u(x-x_0) = 0 \quad \text{for } x < x_0$$
$$= \frac{1}{2} \quad \text{for } x = x_0$$
$$= 1 \quad \text{for } x > x_0$$



Triangular function:

$$Tri(\frac{x}{2L}) = 1 - \frac{|x|}{L} \quad \text{for } |x| < L$$
$$= 0 \quad \text{for } |x| > L$$

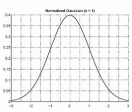


Normalized Gaussian:

$$G_n(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-(x-\mu)^2}{2\sigma^2}}$$

μ is the mean
σ is the standard deviation

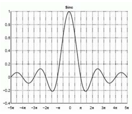
- normalized means that the integral for all *x* is 1



Sinc function:

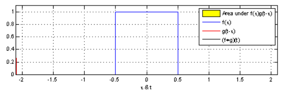
$$sinc(x) = \frac{\sin(\pi x)}{\pi x}$$

- sinc(0) = 1 (L'Hopital's rule)



- Um sinal f(x,y) é separável se existirem dois sinais unidimensionais $f_1(x)$ e $f_2(y)$ tais que:
  - $f(x,y) = f_1(x)\,f_2(y)$
- São limitados, pois podem somente modelar variações independentes em cada variável
- São apropriados em certas situações, pois são mais simples de operar do que funções em 2D ou 3D.

The expression
$$s_o(x) = \int_{-\infty}^{+\infty} s_i(\xi)h(x-\xi)d\xi = s_i * h$$
is called *convolution*, defined as:
$$s_1(x) * s_2(x) = \int_{-\infty}^{+\infty} s_1(\xi)s_2(x-\xi)d\xi = \int_{-\infty}^{+\infty} s_1(x-\xi)s_2(\xi)d\xi$$



$$s_1(x,y) * s_2(x,y)$$
$$= \iint_{-\infty}^{+\infty} s_1(x-\xi, y-\zeta)\,s_2(\xi,\zeta)\,d\xi\,d\zeta,$$

- Propriedades:
  - $s_1 * s_2 = s_2 * s_1$ (comutativa)
  - $(s_1*s_2)*s_3 = (s_1*s_2)*s_3$ (associativa)
  - $s_1*(s_2+s_3) = s_1*s_2 + s_1*s_3$ (distributiva)

Now assume the input is a complex sinusoid with $Ae^{2\pi iks}$ then:

$$s_o(x) = \int_{-\infty}^{+\infty} Ae^{2\pi ik(x-\xi)}h(\xi)d\xi$$
$$= Ae^{2\pi ikx}\int_{-\infty}^{+\infty} e^{-2\pi ik\xi}h(\xi)d\xi$$
$$= Ae^{2\pi ikx}H$$

for now, assume $\varphi=0$

H is called the *Fourier Transform* of *h*(*x*):

$$H = \int_{-\infty}^{+\infty} e^{-2\pi ik\xi}h(\xi)d\xi$$

- *H* is also often called the *transfer function* or *filter*

H scales, and maybe phase-shifts, the input sinusoid S_i

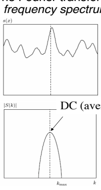
In essence, we have now two alternative representations:
- determine the effect of *L* on s_i by convolution with *h*: $s_i * h$
- determine the effect of *L* on s_i by multiplication with *H*: $S_i \cdot H$

$$s_i * h \leftrightarrow S_i \cdot H$$

Since convolution is expensive for wide *h*, the multiplication may be cheaper
- but we need to perform the Fourier transforms of s_i and *h*

Theory developed by Joseph Fourier (1768-1830)

The Fourier transform of a signal s(x) yields its *frequency spectrum S*(*k*)



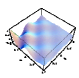
forward transform
$$S(k) = F\{s(x)\} = \int_{-\infty}^{+\infty} s(x)e^{-2\pi ikx}dx$$

inverse transform
$$s(x) = F^{-1}\{S(k)\} = \int_{-\infty}^{+\infty} S(k)e^{2\pi ikx}dk$$

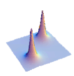
The Fourier transform generalizes to higher dimensions

Consider the 2D case:

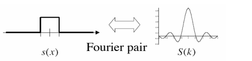
forward transform
$$S(k,l) = F\{s(x,y)\} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} s(x,y)e^{-2\pi i(kx+ly)}dxdy$$



inverse transform
$$s(x,y) = F^{-1}\{S(k,l)\} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} S(k,l)e^{2\pi i(kx+ly)}dkdl$$



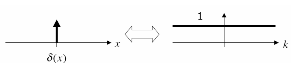
$$S(k) = F\{A\Pi(\frac{x}{2L})\} = \int_{-\infty}^{+\infty} A\Pi(\frac{x}{2L})e^{-i2\pi kx}dx = \int_{-L}^{+L} Ae^{-i2\pi kx}dx$$
$$= -\frac{A}{2\pi ki}(e^{-i2\pi kL} - e^{i2\pi kL}) = \frac{A}{2\pi k}2\sin(2\pi kL)$$
$$= 2AL\ \text{sinc}(2\pi kL)$$



We see that a finite signal in the *x*-domain creates an infinite signal in the *k*-domain (the frequency domain)
- the same is true vice versa

For s(*x*)=δ(*x*):

$$S(k) = F\{\delta(x)\} = \int_{-\infty}^{+\infty} \delta(x)e^{-i2\pi kx}dx = e^{-i2\pi k0} = 1$$



Recall that the Dirac is an extremely thin rect function
- the frequency spectrum is therefore extremely broad (1 everywhere)

This illustrates a key feature of the Fourier Transform:
- the narrower the *s*(*x*), the wider the *S*(*k*)
- sharp objects need higher frequencies to represent that sharpness

Scaling:

$$S(k) = 2AL\ \text{sinc}(2\pi kL)$$

$$F\{s(ax)\} = \frac{1}{|a|}S(\frac{k}{a}) \qquad \begin{matrix} a{>}1 \text{ shrinks } s \\ a{<}1 \text{ stretches } s \end{matrix}$$

Linearity: $F\{as_1(x)+bs_2(x)\} = F\{as_1(x)\} + F\{bs_2(x)\}$

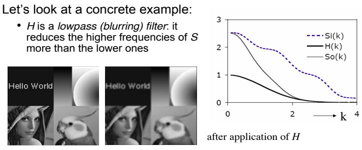
Translation: $F\{s(x-x_0)\} = S(k)e^{-2\pi ix_0k}$ phase shift

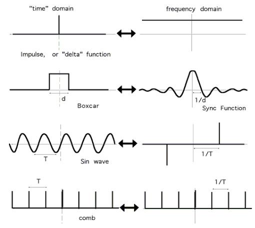
Convolution: $F\{s_1(x) * s_2(x)\} = S_1(k) \cdot S_2(k)$

$F\{s_1(x) \cdot s_2(x)\} = S_1(k) * S_2(k)$

$$s_0(x) = \int_{-\infty}^{+\infty} S_i(k)e^{2\pi ikx}H(k)dk$$
$$s_o(x) = s_i(x) * h(x) \leftrightarrow S_i(k) \cdot H(k) = S_o(k)$$

Let's look at a concrete example:
- *H* is a *lowpass (blurring) filter*: it reduces the higher frequencies of *S* more than the lower ones

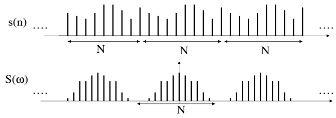




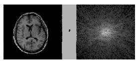
Discrete Fourier Transform (DFT)
- assumes that the signal is discrete and finite

$$S(k) = \sum_{n=0}^{N-1} s(n)e^{\frac{-i2\pi kn}{N}} \qquad s(n) = \frac{1}{N}\sum_{n=0}^{N-1} S(k)e^{\frac{i2\pi kn}{N}}$$

- now we have only N samples, and we can calculate N frequencies
- the frequency spectrum is now discrete, and it is periodic in N



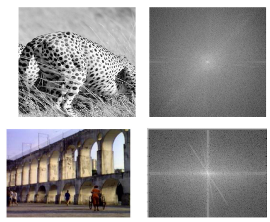
The 2D transform:



$$S(k,l) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} s(n,m)e^{\frac{-i2\pi(kn+lm)}{NM}}$$
$$s(n,m) = \frac{1}{NM}\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} S(k,l)e^{\frac{i2\pi(kn+lm)}{NM}}$$

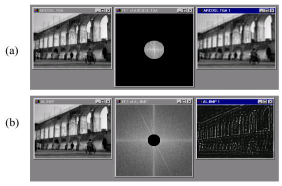
Separability:

$$S(k,l) = \frac{1}{NM}\sum_{m=0}^{M-1} e^{\frac{-i2\pi lm}{M}} P(k,m) \quad \text{where } P(k,m) = \sum_{n=0}^{N-1} s(n,m)e^{\frac{-i2\pi kn}{N}}$$
$$s(n,m) = \frac{1}{NM}\sum_{l=0}^{M-1} e^{\frac{-i2\pi lm}{M}} p(n,l) \quad \text{where } p(n,l) = \sum_{k=0}^{N-1} S(n,m)e^{\frac{-i2\pi kn}{N}}$$

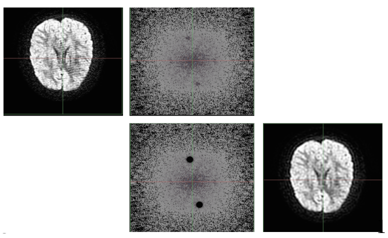


A. Efros, CMU

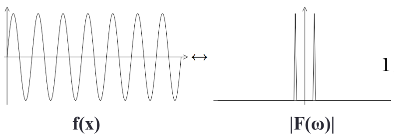
(a) Lower frequencies (close to origin) give overall structure
(b) Higher frequencies (periphery) give detail (sharp edges)



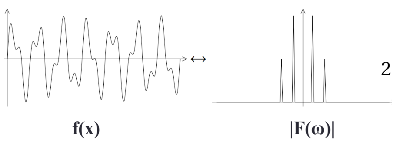
A. Efros, CMU



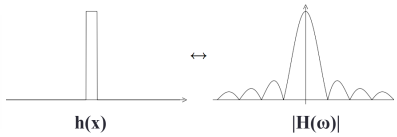
Single sine wave $f(x) = \sin(\omega x)$:



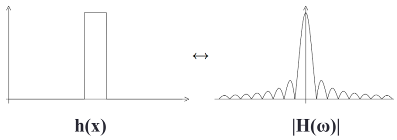
Sum of two sine waves
$f(x) = \sin(\omega x) + 0.5\sin(2\omega x)$:

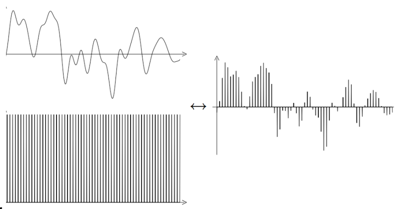


Box function:

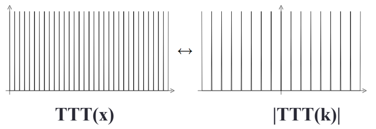


Wider box function:

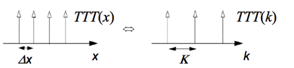




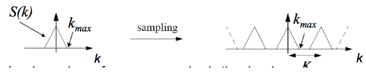
Wider comb function:

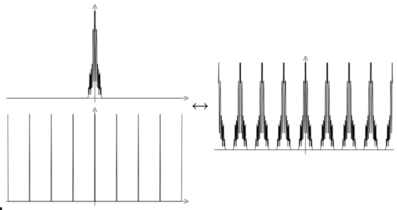


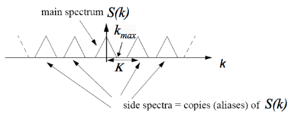
$$S_s(k) = S(k) * F\{\text{TTT}(x)\}, \quad \text{where } F\{\text{TTT}(x)\} = K\sum_{l=-\infty}^{+\infty}\delta(k - lK$$



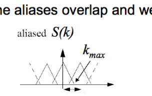
- the smaller *Δx* the wider *K* (recall the Fourier scaling theorem)
- sampling (the convolution of *TTT*(*k*) and *S*(*k*)) replicates the signal spectrum *S*(*k*) at integer multiples of sampling frequency *K*







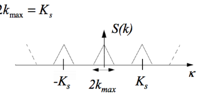
However, if we choose $K < 2\,k_{max}$ the aliases overlap and we get *aliasing*
- what does aliasing look like?



$$K > K_s = 2 \cdot k_{max}, \quad K_s \text{ is the } Nyquist\ rate$$

In other words:
- the samples only uniquely define the signal if:

$$S(k) = 0 \quad \forall |k| > k_{max}$$
$$\frac{1}{\Delta x} > 2k_{max} = K_s$$

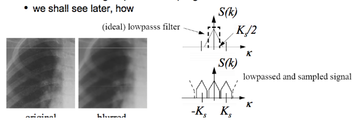


- this assumes that the signal is band-limited (*S*(*k*)=0 above K_s

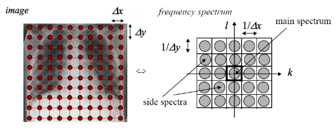
Usually signals are not band-limited
- recall the infinite spectrum of a sharp edge (for example: a bone)

To prevent the inevitable aliasing we must perform anti-aliasing before sampling the signal
- for example: when digitizing a radiograph of a bone or a chest

Anti-aliasing is done by low-pass filtering (blurring)
- band-limit the signal *prior* to sampling
- we shall see later, how

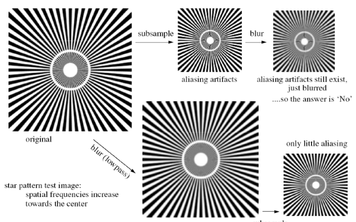


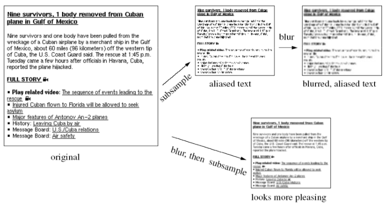
All of these concepts readily extend to higher dimensions



Main spectrum (S(k,l) must fit into the center box to prevent overlap with side-spectra (and aliasing)

$$\frac{1}{\Delta x} > 2\cdot k_{x\max} \qquad \frac{1}{\Delta y} > 2\cdot k_{y\max}$$





We observe: Anti-aliasing (i.e., blurring, lowpassing) must be applied before sampling

Now assume the input is a complex sinusoid with $Ae^{2\pi ikx}$ then:

$$s_0(x) = \int_{-\infty}^{+\infty} Ae^{2\pi ik(x-\xi)}h(\xi)d\xi$$

for now, assume $\varphi=0$

$$= Ae^{2\pi ikx}\int_{-\infty}^{+\infty} e^{-2\pi ik\xi}h(\xi)d\xi$$

$$= Ae^{2\pi ikx}H$$

H is called the *Fourier Transform* of *h*(*x*):

$$H = \int_{-\infty}^{+\infty} e^{-2\pi ik\xi}h(\xi)d\xi$$

- *H* is also often called the *transfer function* or *filter*

H scales, and maybe phase-shifts, the input sinusoid S_i

In essence, we have now two alternative representations:

- determine the effect of *L* on s_i by convolution with *h*: $s_i * h$
- determine the effect of *L* on s_i by multiplication with *H*: $S_i \cdot H$

$$s_i * h \leftrightarrow S_i \cdot H$$

Since convolution is expensive for wide *h*, the multiplication may be cheaper

- but we need to perform the Fourier transforms of s_i and *h*

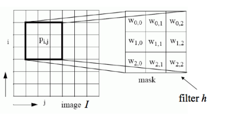
$$s_1(x,y) * s_2(x,y) = \iint_{-\infty}^{+\infty} s_1(x-\xi, y-\zeta)\, s_2(\xi,\zeta)\, d\xi\, d\zeta,$$

- Propriedades:
  - $s_1 * s_2 = s_2 * s_1$ (comutativa)
  - $(s_1 * s_2) * s_3 = (s_1 * s_2) * s_3$ (associativa)
  - $s_1 * (s_2 + s_3) = s_1 * s_2 + s_1 * s_3$ (distributiva)

We say *discrete filters* since they operate on a discretized signal, the image

- to implement discrete filters we use discrete convolution

$h * I$

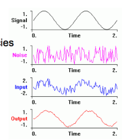


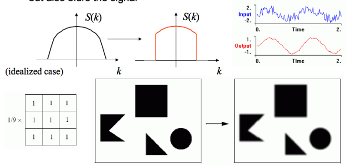
Procedure:

- place a weight matrix or *mask* at each pixel location p_{ij}
- this mask weighs the pixel's neighborhood and determines the output pixel's value
- important: do not replace the computed values into the original image, but write to an output image

Smoothing (averaging):

- also called *low-passing:* keeps the low frequencies, but reduces the high frequencies
- removes noise and jagged edges
- but also blurs the signal



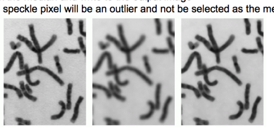


A non-linear filter, best used to remove speckle noise

- a regular smoothing filter would blur the speckles (and the signal)
- the median filter will eliminate the speckle and leave the signal as is

Procedure:

- convolve with a mask as usual
- but this time, for each mask position, sort the values under the mask
- pick the median and write to the output image
- the speckle pixel will be an outlier and not be selected as the median

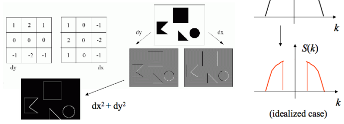


Edge detector / enhancer:

$\nabla I = \nabla h * I$ first derivative (gradient)

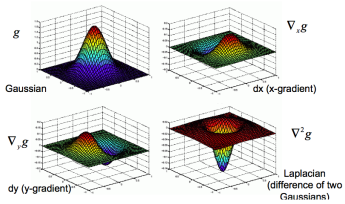
$\nabla^2 I = \nabla^2 h * I$ second derivative (Laplacian)

- also called *high-passing:* keeps the high frequencies, but reduces the low frequencies
- enhances edges and contrast
- but also enhances noise and jagged edges



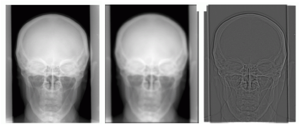
The Gaussian kernel is a popular filter function

- see book for 3x3 convolution masks

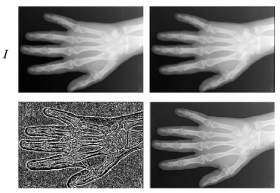


Several useful effects can be achieved by subsequent filtering with different masks (kernels) and/or multi-image operations

Subtracting a smoothed image from the original image leaves the edges (the high frequencies):



Places the enhanced edges on top of a smoothed original

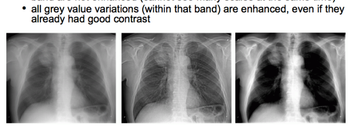


Windowing enhances contrast only for a specific range of grey levels (not sensitive to edges)

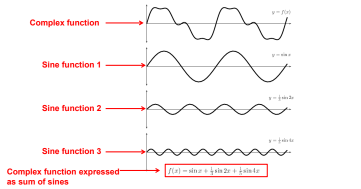
- strong edges with already good contrast are further enhanced

Edge enhancement (such as sharp masking) only boosts features within a certain frequency band

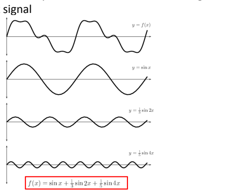
- this frequency band is determined by filter size -- features outside that band are not enhanced (cannot see many scales at the same time)
- all grey value variations (within that band) are enhanced, even if they already had good contrast



- **Main idea:** Any periodic function can be decomposed into a summation of sines and cosines



- Mathematially easier to analyze effects of transmission medium, noise, etc on simple sine functions, then add to get effect on complex signal



- If *f(x)* is **periodic function** of period *2T*
- **Fourier series expansion**

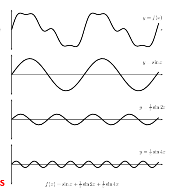
$$f(x) = a_0 + \sum_{n=1}^{\infty}\left(a_n \cos\frac{n\pi x}{T} + b_n \sin\frac{n\pi x}{T}\right)$$

Where

$$a_0 = \frac{1}{T}\int_{-T}^{T} f(x)dx$$

$$a_n = \frac{1}{T}\int_{-T}^{T} f(x)\cos\frac{n\pi x}{T}dx, \; n = 1,2,3,...$$

$$b_n = \frac{1}{T}\int_{-T}^{T} f(x)\sin\frac{n\pi x}{T}dx, \; n = 1,2,3,...$$



- a_n and b_n called **Fourier coefficients**

- (Almost) any periodic function *g(x)* with fundamental frequency ω_0 can be described as a sum of sinusoids

$$g(x) = \sum_{k=0}^{\infty}\left[A_k \cos(k\omega_0 x) + B_k \sin(k\omega_0 x)\right]$$

Infinite sum of sines — Cosines — Sines

- This infinite sum is called a **Fourier Series**
- Summed sines and cosines are multiples of the fundamental frequency (harmonics)
- A_k and B_k called **Fourier coefficients**
  - Not known initially but derived from original function *g(x)* during **Fourier analysis**

- Large speedups if *S* is large
- Example: *M* = 512 x 512, *S* = 32 x 32
- Direct computation:
  - 32^2 = 1024 multiplications for each pixel
  - Total multiplications for entire image = 512 x 512 x 1024 = **268,435,456** multiplications
- Using DFT:
  - Each row requires 4608 multiplications
  - Multiplications for rows = 4608 x 512 = 2,359,296 multiplications
  - Repeat for columns, DFT of image = 4718592 multiplications
  - Need same for DFT of filter and for inverse DFT.
  - Also need 512 x 512 multiplications for product of 2 transforms
  - Total multiplications = 4718592 x 3 + 262144 = **14,417,920**

DC Component

- Recall that:

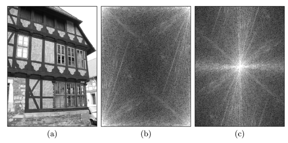
$$F(u,v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp\left[-2\pi i\left(\frac{xu}{M}+\frac{yv}{N}\right)\right]$$

- The value *F(0,0)* of the DFT is called the **dc coefficient**
- If we put *u* = *v* = 0, then

$$F(0,0) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\exp(0) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)$$

- Essentially *F(0,0)* is the sum of all terms in the original matrix

Centering DFT Spectrum



Original Image — Non-centered spectrum — Centered spectrum

Displaying Transforms

- As elements are complex numbers, we can view magnitudes $|F(u,v)|$ directly
- Displaying magnitudes of Fourier transforms called **spectrum** of the transform
- **Problem:** DC component much larger than other values
  - Displays white dot in middle surrounded by black
- So stretch transform values by displaying log of transform

$$\log(1+|F(u,v)|)$$

Examples of DFTs

- Suppose we have as input a constant image of all 1's, *f(x,y)* = 1
- The DFT yields just a DC component, 0 everywhere else
- In this example, DC component is sum of all elements = 64

64	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

DFT of Image

- Consider DFT of image with single edge
- For display, DC component shifted to center
- Log of magnitudes of Fourier Transform displayed

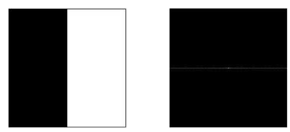
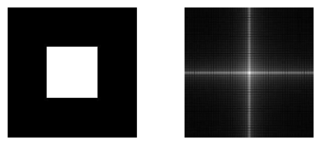


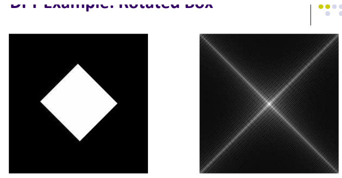
Image — DFT

DFT Example: A Box



Box — DFT

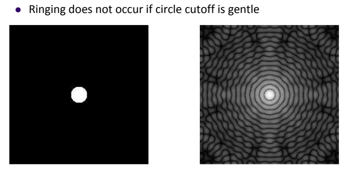
DFT Example: Rotated Box



Rotated Box — DFT

DFT Example: A Circle

- **Note:** Ringing caused by sharp cutoff of circle
- Ringing does not occur if circle cutoff is gentle

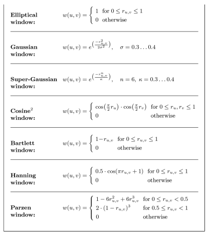


Circle — DFT

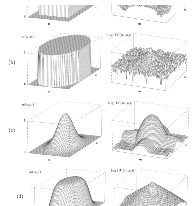
- DFT computation assumes image repeated horizontally and vertically
- Large intensity transition at edges = vertical and horizontal line in middle of spectrum after shifting
- Effects of sharp transitions affects many pixels

Windowing

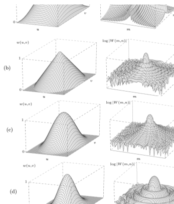
- Can multiply image by windowing function *w(u,v)* before DFT to reduce sharp transitions between ends of repeated images
- Ideally, causes image to drop off towards ends, reducing transitions



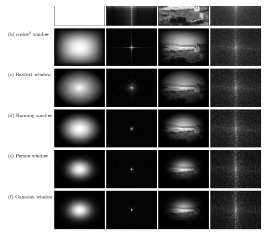
Some Proposed Windowing Functions



Some Proposed Windowing Functions



Some Proposed Windowing Functions



Application of Windowing Functions

- Conversa Prof. Choukri
- Aula prática na próxima quinta-feira com imagens usando FFT em 2D

Close enough to start a war
All that I have is on the floor
God only knows what we're fighting for
All that I say, you always say more