Discrete Fourier Transform

Ajit Rajwade

CS 663

Fourier Transform

- You have so far studied the Fourier transform of a 1D or 2D continuous (analog) function.
- The functions we deal with in practical signal or image processing are however discrete.
- We need an equivalent of the Fourier transform of such discrete signals.

Towards the Discrete Fourier Transform

- Given a continuous signal *f(t)*, we will consider signal *g(t)*, obtained by sampling *f(t)* at equally-spaced time instants.

- So *g(t)* is the pointwise product of *f(t)* with an impulse train $s_{\Delta T}(t)$ given by:

$$s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} \delta(t - n\Delta T)$$

$$g(t) = f(t)s_{\Delta T}(t)$$

The impulse functions here are Dirac delta functions.

Towards the Discrete Fourier Transform

$$G(\mu) = \mathsf{F}\ (g(t))(\mu) = \mathsf{F}\ (f(t)s_{\Delta T}(t))(\mu) = F(\mu) * S(\mu)$$

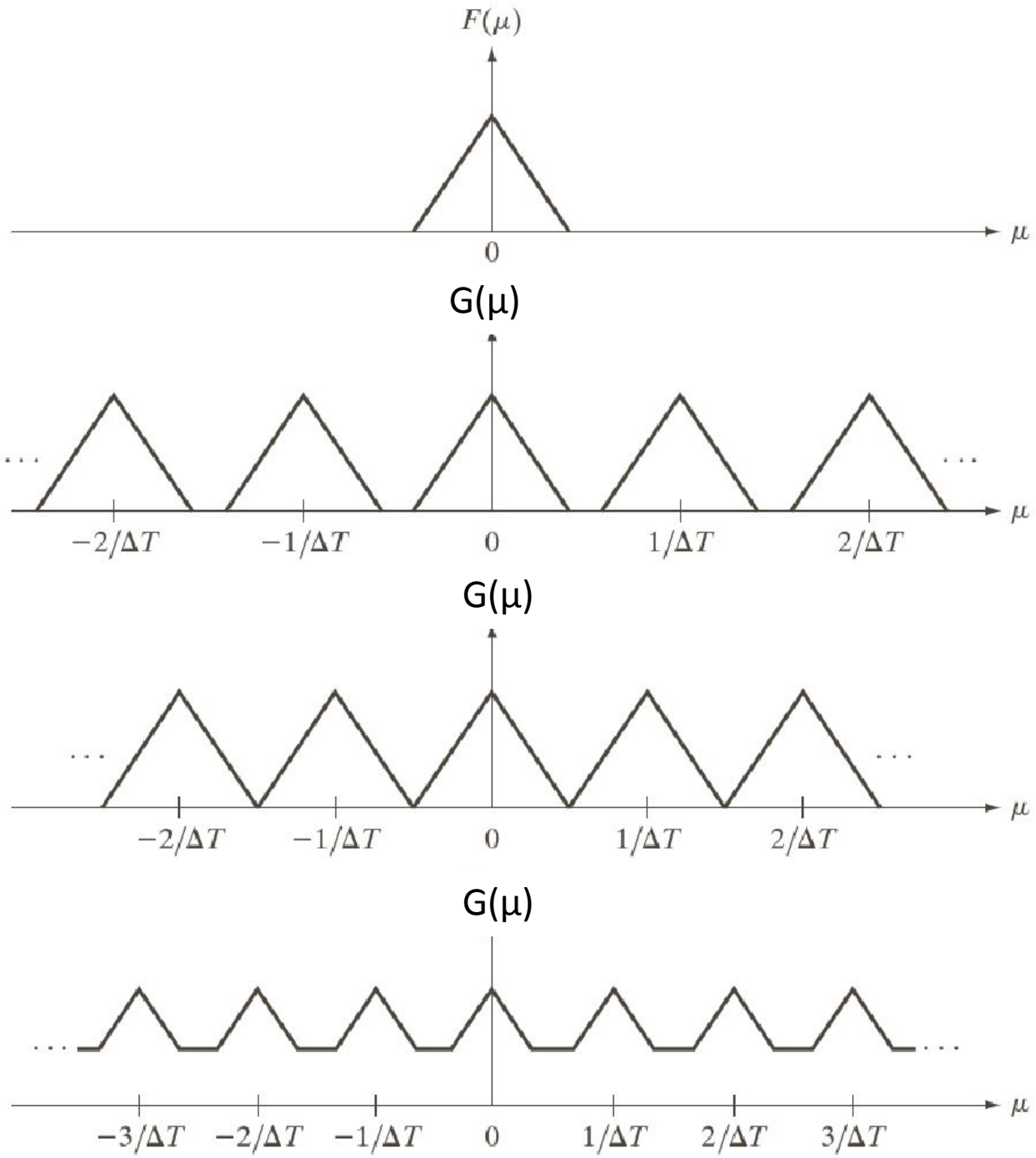
where $$S(\mu) = \frac{1}{\Delta T}\sum_{n=-\infty}^{\infty}\delta(\mu - n/\Delta T)$$

Check the book for the derivation for this (see Appendix at the end of the slides)

$$\therefore G(\mu) = \int_{-\infty}^{\infty} F(\tau)S(\mu-\tau)d\tau = \frac{1}{\Delta T}\int_{-\infty}^{\infty} F(\tau)\sum_{n=-\infty}^{\infty}\delta(\mu-\tau-n/\Delta T)d\tau$$

$$= \frac{1}{\Delta T}\sum_{n=-\infty}^{\infty} F(\mu - n/\Delta T)$$

This shows that the Fourier transform *G(μ)* of the sampled signal *g(t)* is an infinite periodic sequence of copies of *F(μ)* which is the Fourier transform of *f(t)*. The periodicity of *G(μ)* is given by the sampling period ΔT. Also though *g(t)* is a sampled function, its Fourier transform is continuous.



a
b
c
d

FIGURE 4.6
(a) Fourier transform of a band-limited function.
(b)–(d) Transforms of the corresponding sampled function under the conditions of over-sampling, critically-sampling, and under-sampling, respectively.

Towards the Discrete Fourier Transform

- Let us now try to derive *G(μ)* directly in terms of *g(t)* instead of *F(μ)*.

$$G(\mu) = \mathrm{F}\ (g(t)) = \int_{-\infty}^{\infty} g(t)\exp(-j2\pi\mu t)dt$$

$$= \int_{-\infty}^{\infty}\sum_{n=-\infty}^{\infty} f(t)\delta(t-n\Delta T)\exp(-j2\pi\mu t)dt$$

$$= \sum_{n=-\infty}^{\infty}\int_{-\infty}^{\infty} f(t)\delta(t-n\Delta T)\exp(-j2\pi\mu t)dt$$

$$= \sum_{n=-\infty}^{\infty} f_n \exp(-j2\pi\mu n\Delta T)$$

f(t)δ(t-nΔT) is the sampled function (a Dirac impulse weighted by f(t)). To get the actual sample values (the strength of the impulse), we do an integration, i.e. of the following form:

$$f_n = \int_{-\infty}^{\infty} f(t)\delta(t-n\Delta T)dt = f(n\Delta T)$$

Towards the Discrete Fourier Transform

- We have seen earlier that the Fourier transform of *g(t)* is periodic with period $1/\Delta T$, so it is of interest to characterize one period.

- Consider that we take some *M* equally spaced samples of *G(μ)* over one period from $\mu = 0$ to $1/\Delta T$.

- This gives us frequencies of the form $\mu = u/M\Delta T$ where $u = 0,1,2\ldots,M-1$.

Towards the Discrete Fourier Transform

- Plugging these values for μ we now have:

$$G(u) = F_d(u) = \sum_{n=0}^{M-1} f_n \exp(-j2\pi un / M)$$

- This is called the Discrete Fourier Transform of the sequence $\{f_n\}$ where $n = 0,1,\ldots,M\text{-}1$.

- Given the sequence $\{F_d(u)\}$ where $u = 0,1,\ldots M\text{-}1$, we can recover $\{f_n\}$ using:

$$f(n) = f_n = \frac{1}{M} \sum_{u=0}^{M-1} F_d(u) \exp(j2\pi un / M)$$

This is the inverse Discrete Fourier Transform (IDFT).

Towards the Discrete Fourier Transform

- Consider:

$$G(u) = F_d(u) = \sum_{n=0}^{M-1} f_n \exp(-j2\pi un / M)$$

$$f(n) = \frac{1}{M} \sum_{u=0}^{M-1} F_d(u) \exp(j2\pi un / M)$$

- It can be proved that plugging in the expression for *f(n)* into the expression for $F_d(u)$ yields the identity $F_d(u) = F_d(u)$.

- Also plugging in the expression for $F_d(u)$ into the expression for *f(n)* yields the identity *f(n) = f(n)*.

Towards the Discrete Fourier Transform

- In some books, the following expressions are used:

$$F_d(u) = \frac{1}{\sqrt{M}} \sum_{n=0}^{M-1} f_n \exp(-j2\pi un / M)$$

$$f(n) = \frac{1}{\sqrt{M}} \sum_{u=0}^{M-1} F_d(u) \exp(j2\pi un / M)$$

MATLAB does NOT follow this convention – it follows the one on the earlier slide.

- Note that the above expressions can be written in the following matrix vector format:

$$\mathbf{f} = \mathbf{U}\mathbf{F_d}$$

$$\mathbf{F_d} = \mathbf{U}^T\mathbf{f}$$

Vectors of size *M* by 1

U turns out to be an **orthonormal** *M* by *M* matrix – called the *discrete Fourier basis matrix* or *DFT matrix*. The square root of *M* in the denominator is required for **U** to be orthonormal, else it would be proportional to an orthonormal matrix.

$\mathbf{f} = \mathbf{U}\mathbf{F}_\mathbf{d}$ Vectors of size *M* by 1

$\mathbf{F}_\mathbf{d} = \mathbf{U}^T\mathbf{f}$ **U** is an orthonormal *M* by *M* matrix – called the discrete Fourier basis matrix or DFT matrix. The square root of *M* is required for **U** to be orthonormal, else it would be proportional to an orthonormal matrix.

$$\mathbf{f} = \mathbf{U}\mathbf{F}_d, \mathbf{f} \in \mathbf{R}^M, \mathbf{F}_d \in \mathbf{C}^M, \mathbf{U} \in \mathbf{C}^{M \times M}; \mathbf{C} = \text{complex domain}, \mathbf{R} = \text{real domain}$$

$$\begin{pmatrix} f(0) \\ f(1) \\ . \\ . \\ f(M-1) \end{pmatrix} = \frac{1}{\sqrt{M}} \begin{pmatrix} e^{j2\pi(0)(0)/M} & e^{j2\pi(0)(1)/M} & . & . & e^{j2\pi(0)(M-1)/M} \\ e^{j2\pi(1)(0)/M} & e^{j2\pi(1)(1)/M} & . & . & e^{j2\pi(1)(M-1)/M} \\ . & . & . & . & . \\ . & . & . & . & . \\ e^{j2\pi(M-1)(0)/M} & e^{j2\pi(M-1)(1)/M} & . & . & e^{j2\pi(M-1)(M-1)/M} \end{pmatrix} \begin{pmatrix} F_d(0) \\ F_d(1) \\ . \\ . \\ F_d(M-1) \end{pmatrix}$$

$$= \frac{1}{\sqrt{M}} \begin{pmatrix} e^{j2\pi(0)(0)/M} \\ e^{j2\pi(1)(0)/M} \\ . \\ . \\ e^{j2\pi(M-1)(0)/M} \end{pmatrix} F_d(0) + \frac{1}{\sqrt{M}} \begin{pmatrix} e^{j2\pi(0)(1)/M} \\ e^{j2\pi(1)(1)/M} \\ . \\ . \\ e^{j2\pi(M-1)(1)/M} \end{pmatrix} F_d(1) + \ldots + \frac{1}{\sqrt{M}} \begin{pmatrix} e^{j2\pi(0)(M-1)/M} \\ e^{j2\pi(1)(M-1)/M} \\ . \\ . \\ e^{j2\pi(M-1)(M-1)/M} \end{pmatrix} F_d(M-1)$$

Notice in the last equality how the signal **f** is being represented as a linear combination of column vectors of the DFT matrix. The coefficients of the linear combination are the *discrete Fourier coefficients*!

Sampling in time and frequency

- Remember that the function *g(t)* was created by sampling *f(t)* with a period of ΔT.

- And the spacing between the samples in the frequency domain (to get the DFT from G(μ)) is $1/ M\Delta T$ since $\mu = u/M\Delta T$ where $u = 0,1,2\ldots,M-1$.

- Likewise the *range of frequencies* spanned by the DFT is also inversely proportional to ΔT.

DFT properties

- Linearity: $\mathcal{F}(af+bg)(u) = a\mathcal{F}(f)(u) + b\mathcal{F}(g)(u)$
- Periodicity: $F(u) = F(u + kM)$ for integer k,
- And so is the inverse DFT since $f(n) = f(n+kM)$.
- The DFT is in general complex. Hence it has a *magnitude* $|F(u)|$ and a *phase*.

Clarification about DFT

- We have seen earlier that the Fourier transform $G(\mu)$ of the sampled version $g(t)$ of analog signal $f(t)$ is continuous.
- We also saw earlier that we take some M equally spaced samples of $G(\mu)$ over one period from $\mu = 0$ to $1/\Delta T$.
- This way the DFT and the discrete signal both were vectors of M elements.
- Obtaining the DFT given the signal (or the signal given its DFT) is an efficient operation owing to the orthonormality of the discrete fourier matrix (why efficient? Because for an orthonormal matrix, inverse = transpose).
- Why don't we take more than M samples in the frequency domain?
- If we did, the aforementioned computational efficiency would be lost. The inverse transform would require a matrix pseudo-inverse which is costly.
- Also the columns of the orthonormal matrix $\mathbf{U}^T$(size M x M) constitute a basis: i.e. any vector in M-dimensional space can be **uniquely** represented using linear combination of the columns of that matrix. If you took more than M samples in the frequency domain, that **uniqueness** would be lost as $\mathbf{U}^T$ would now have size M' x M where $M' > M$.

Convolution of discrete signals

- Discrete equivalent of the convolution is:

$$(f * h)(n) = \sum_{m=0}^{M-1} f(m)h(n-m)$$

- Due to the periodic nature of the DFT and IDFT, the convolution will also be periodic.

- The above formula represents one period of that periodic resultant signal.

Convolution of discrete signals

- The convolution theorem from continuous signals has an extension to the discrete case as well:

$$\mathsf{F}\ (f * h)(u) = F(u)H(u)$$

- Therefore discrete convolution can be implemented using product of DFTs followed by an IDFT.

- But in doing so, one has to account for **periodicity issues** to avoid *wrap-around artifacts* (see next slide).

Convolution of discrete signals

- Consider the discrete convolution:

$$(f * h)(n) = \sum_{l=0}^{M-1} f(l)h(n-l)$$

- To convolve *f* with *h*, you need to (1) flip *h* about the origin, (2) translate the flipped signal by an amount *n*, and (3) compute the sum in the above formula.
- Steps 2 and 3 are repeated for each value of *n*.
- The variable *n* ranges over all integers required to completely slide *h* over *f*.
- If *h* and *f* have size *M*, then *n* has to range up to 2*M*-1.

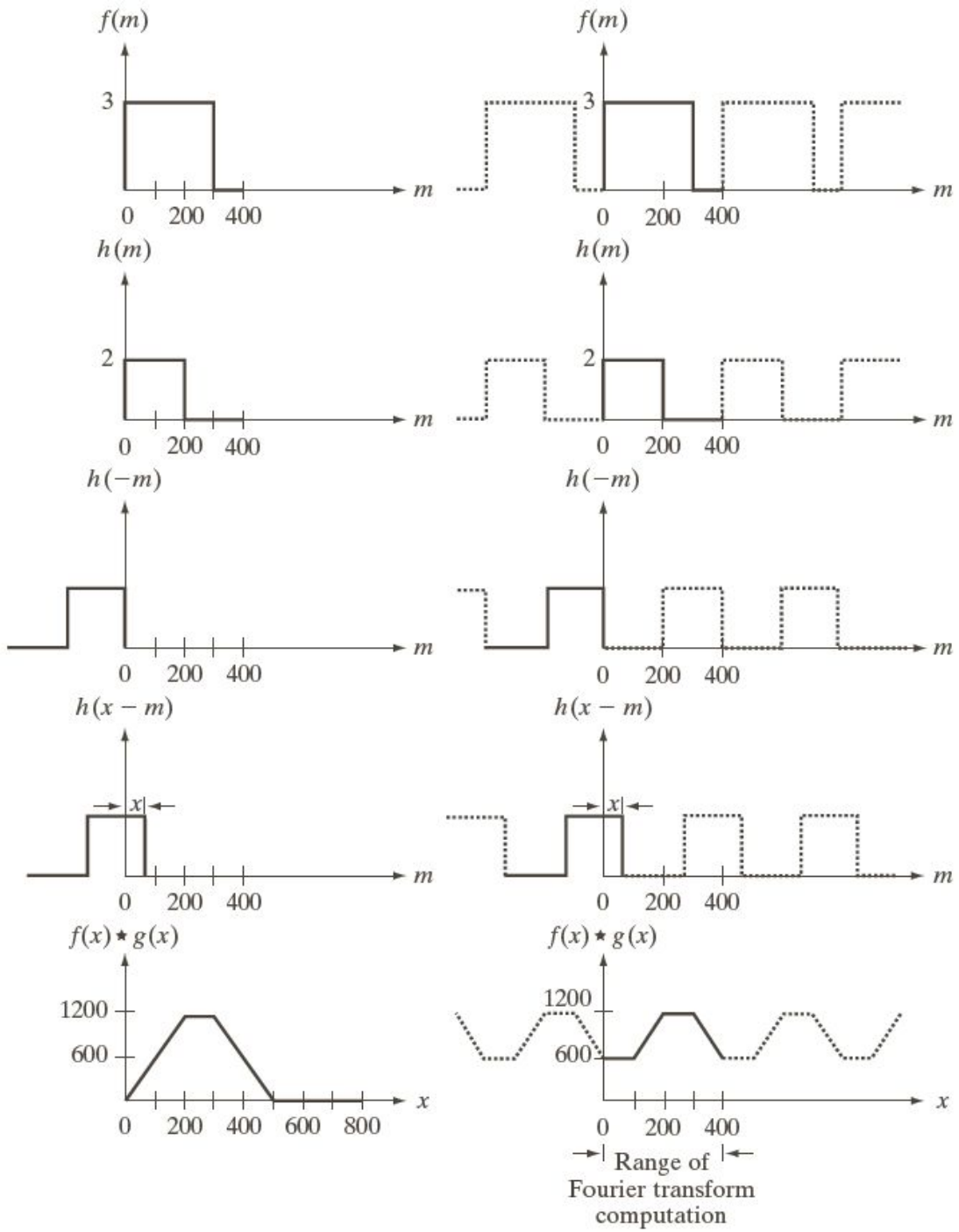
Convolution of discrete signals

- In other words, the resultant signal must have length of $2M$-1.

- The convolution can be implemented in the time/spatial domain – and MATLAB has a routine called **conv** which does the job for you!

- Now imagine you tried to implement the convolution using a DFT of M-point signals followed by an M-point IDFT.

- Owing to the assumed periodicity, you would get an undesirable wrap-around effect. See next slide for an illustration.



a f
b g
c h
d i
e j

FIGURE 4.28 Left column: convolution of two discrete functions obtained using the approach discussed in Section 3.4.2. The result in (e) is correct. Right column: Convolution of the same functions, but taking into account the periodicity implied by the DFT. Note in (j) how data from adjacent periods produce wraparound error, yielding an incorrect convolution result. To obtain the correct result, function padding must be used.

Convolution of discrete signals

- How does one deal with this conundrum?

- If **f** has length *M* and **h** has length *K*, then you should zero-pad both sequences so that they now have length at least *M*+*K*-1.

- Then compute *M*+*K*-1 point DFT for both, multiply the DFTs and compute the *M*+*K*-1 point IDFT.

Convolution of discrete signals: 2 methods

- Consider you want to convolve signal *f* having *N* elements with signal *h* having *K* elements.

- You can use the **conv** routine in MATLAB – which by default produces a signal of *N*+*K*-1 elements for you. It takes care of zero-padding internally for you.

- The other (equivalent) alternative is to:

❑ Append *f* with *K*-1 zeros.
❑ Append *h* with *N*-1 zeros.
❑ Compute the *N*+*K*-1 point DFT of *f* and *h* (using **fft**).
❑ Multiply the two DFTs point-wise.
❑ Compute the IDFT of the result – this gives you a signal with *N*+*K*-1 elements.

- In both cases, you may wish to extract the first *N* elements of the resultant signal (note that the trailing *K*-1 elements may not be zeros!).

Why convolution using Fourier transforms?

- The time complexity of computing the convolution of a signal of length M with another of length M is $O(M^2)$.

- With a DFT computed naively, it would remain $O(M^2)$ – the complexity of multiplying a M x M matrix with a M x 1 vector.

$$\mathbf{f} = \mathbf{U}\mathbf{F}_\mathbf{d}$$

$$\mathbf{F}_\mathbf{d} = \mathbf{U}^T\mathbf{f}$$

Why convolution using Fourier transforms?

- But there's a smarter way of doing the same which computes the DFT in O(M log M) time.
- That is called the **Fast Fourier Transform**, discovered by Cooley and Tukey in 1965.
- It is based on a **divide and conquer** algorithmic strategy.
- The same strategy can be used to compute the IDFT in O(M log M) time.
- Hence convolution can now be computed in O(M log M) time.

Towards the Fast Fourier Transform

$$F(u) = \sum_{n=0}^{M-1} f_n \exp(-j2\pi un / M) = \sum_{n=0}^{M-1} f_n W_M^{un}$$

where $W_M = \exp(-j2\pi / M)$

Let $M = 2^n = 2K$

Even indices

Odd indices

$$F(u) = \sum_{n=0}^{2K-1} f_n W_{2K}^{un} = \sum_{n=0}^{K-1} f_{2n} W_{2K}^{u(2n)} + \sum_{n=0}^{K-1} f_{2n+1} W_{2K}^{u(2n+1)}$$

$$\therefore F(u) = \sum_{n=0}^{K-1} f_{2n} W_K^{u(n)} + \sum_{n=0}^{K-1} f_{2n+1} W_K^{u(n)} W_{2K}^{u} \longrightarrow \boxtimes \; W_{2K}^{u(2n)} = W_K^{u(n)}$$

Towards the Fast Fourier Transform

$$\therefore F(u) = \sum_{n=0}^{K-1} f_{2n} W_K^{u(n)} + \sum_{n=0}^{K-1} f_{2n+1} W_K^{u(n)} W_{2K}^{u}$$

$$\text{Define } F_{even}(u) = \sum_{n=0}^{K-1} f_{2n} W_K^{u(n)}, F_{odd}(u) = \sum_{n=0}^{K-1} f_{2n+1} W_K^{u(n)}$$

$$\therefore F(u) = F_{even}(u) + F_{odd}(u) W_{2K}^{u} \text{ for } u = 0,1,\ldots,K-1$$

$$\text{and } F(u+K) = F_{even}(u) - F_{odd}(u) W_{2K}^{u} \text{ for } u = 0,1,\ldots,K-1$$

$$\text{as } W_{2K}^{u+K} = W_{2K}^{u} W_{2K}^{K} = W_{2K}^{u} \exp(-j2\pi K / 2K) = -W_{2K}^{u}$$

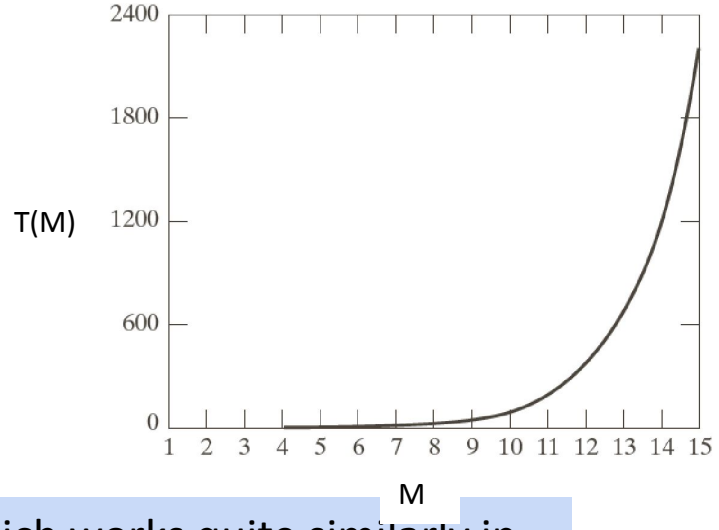
The *M*-point DFT computation for *F* is split up into **two halves**. The **first half** requires two *M*/2-point DFT computations – one for F_{even}, one for F_{odd}. The **second half** follows directly without any additional transform evaluations once F_{even} and F_{odd} are computed. Now F_{even} and F_{odd} can be further split up **recursively**.

Towards the Fast Fourier Transform

The *M*-point DFT computation for *F* is split up into **two halves**. The **first half** requires two *M*/2-point DFT computations – one for F_{even}, one for F_{odd}. The **second half** follows directly without any additional transform evaluations once F_{even} and F_{odd} are computed. Now F_{even} and F_{odd} can be further split up **recursively**.

This gives:

$$T(1) = c \text{ (constant)}$$
$$T(M) = 2T(M/2) + M$$
$$= 2(2T(M/4) + M/2) + M = 2^2 T(M/2^2) + 2M$$
$$= \ldots$$
$$= 2^k T(M/2^k) + kM$$
$$= O(M \log M)$$



There is also a fast inverse Fourier transform which works quite similarly in O(*M* log *M*) time. The speedup achieved by the fast fourier transform over a naïve DFT computation is rather dramatic for large *M*!

MATLAB implementation

- The Fast Fourier Transform is implemented in MATLAB directly – there are the routines **fft** and **ifft** for the inverse.

2D-DFT

- Given a 2D discrete signal (image) $f(x,y)$ of size W_1 by W_2, its DFT is given as:

$$F_d(u,v) = \frac{1}{\sqrt{W_1 W_2}} \sum_{x=0}^{W_1-1} \sum_{y=0}^{W_2-1} f(x,y) \exp(-j2\pi(ux/W_1 + vy/W_2))$$

$$f(x,y) = \frac{1}{\sqrt{W_1 W_2}} \sum_{u=0}^{W_1-1} \sum_{v=0}^{W_2-1} F(u,v) \exp(j2\pi(ux/W_1 + vy/W_2))$$

2D-DFT computation

- Given a 2D discrete signal (image) $f(x,y)$ of size W_1 by W_2, its DFT is given as:

$$F_d(u,v) = \frac{1}{\sqrt{W_1 W_2}} \sum_{x=0}^{W_1-1} \sum_{y=0}^{W_2-1} f(x,y) \exp(-j2\pi vy / W_2) \exp(-j2\pi ux / W_1)$$

Compute row-wise FFT, followed by a column-wise FFT. Overall complexity is $W_1 W_2 \log(W_1 W_2)$.

$$f(x,y) = \frac{1}{\sqrt{W_1 W_2}} \sum_{u=0}^{W_1-1} \sum_{v=0}^{W_2-1} F(u,v) \exp(j2\pi(ux / W_1 + vy / W_2))$$

MATLAB implementation

- The FFT for 2D arrays or images is implemented in MATLAB routines **fft2** and **ifft2**.

2D-DFT properties

- **Linearity:** $\mathcal{F}(af+bg)(u,v) = a\mathcal{F}(f)(u,v) + b\mathcal{F}(g)(u,v)$ where a and b are scalars.

- **Periodicity:** $F(u,v) = F(u + k_1W_1,v) = F(u,v + k_2W_2) = F(u + k_1W_1,v + k_2W_2)$ for integer k_1 and k_2.

- And so is the inverse DFT since $f(x,y) = f(x + k_1W_1,y) = f(x,y + k_2W_2) = f(x + k_1W_1,y + k_2W_2)$ for integer k_1 and k_2.

2D-DFT: Magnitude and phase

- The phase carries very critical information.

- If you retain the magnitude of the Fourier transform of an image, but change its phase, the appearance drastically changes.

IFT (Magnitude of FT of Salman Khan times phase of P V Sindhu)

IFT (Magnitude of FT of P V Sindhu times phase of Salman Khan)

2D-DFT properties

- Translation:

$$\mathsf{F}\ (f(x-x_0, y-y_0))(u,v) = F(u,v)\exp(-j2\pi(ux_0/W_1 + vy_0/W_2))$$

$$F(u-u_0, v-v_0) = \mathsf{F}(\ f(x,y)\exp(j2\pi(xu_0/W_1 + yv_0/W_2)))(u,v)$$

- Note that translation of a signal does not change the magnitude of the Fourier transform – only its phase.

2D-DFT properties

- The Fourier transform of a rotated image yields a rotated version of its Fourier transform

$$\mathrm{F}\ (f(x,y))(u,v) = F(u,v) \rightarrow$$

$$\mathrm{F}\ (f(x\cos\phi - y\sin\phi, x\sin\phi + y\cos\phi))(u,v) =$$

$$F(u\cos\phi - v\sin\phi, u\sin\phi + v\cos\phi)$$

- If you consider conversion to polar coordinates, then we have:

$$x = r\cos\theta, y = r\sin\theta$$

$$u = \omega\cos\theta, v = \omega\sin\theta$$

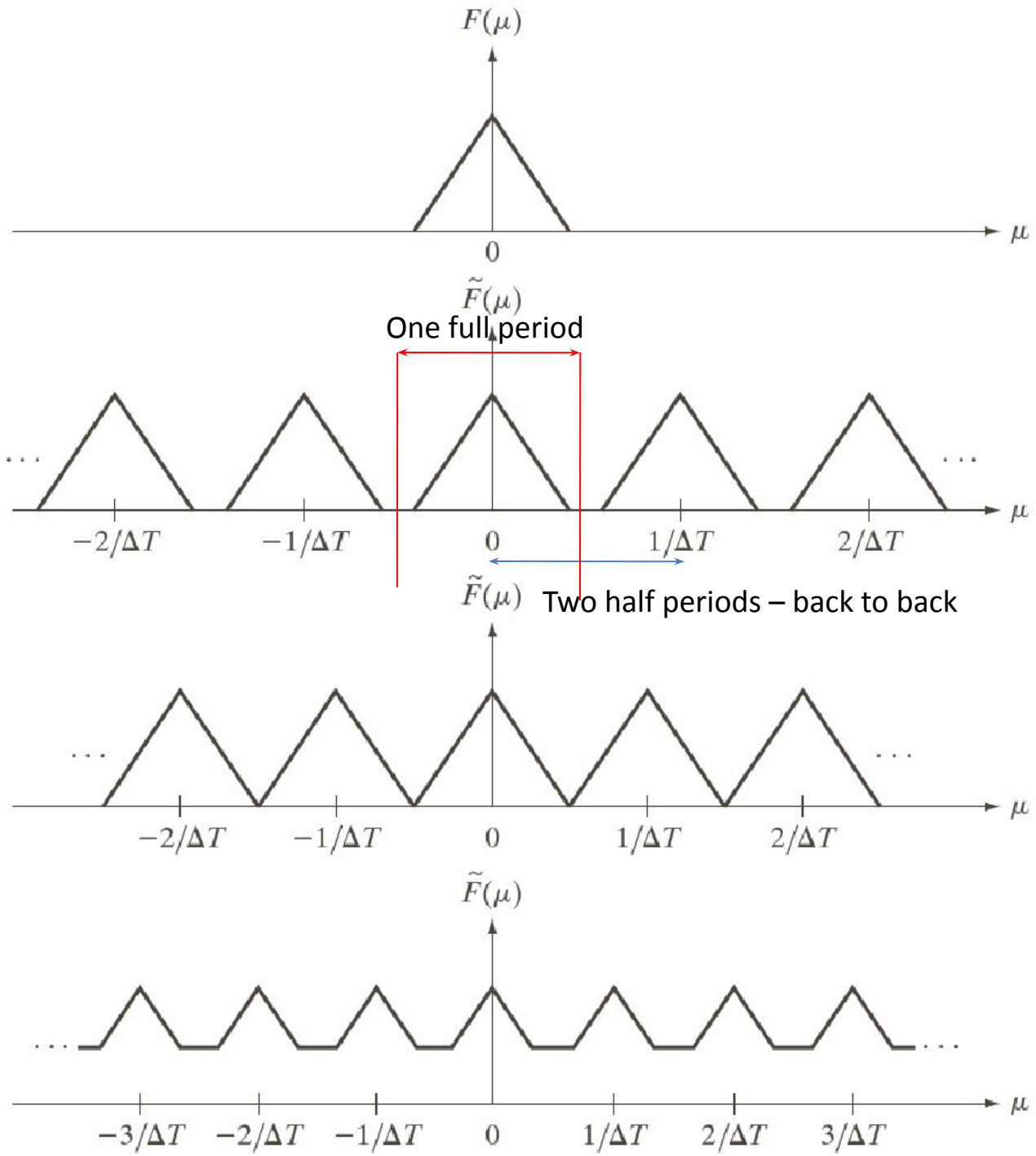
$$\mathrm{F}\ (f(x,y))(u,v) = F(u,v) \rightarrow$$

$$\mathrm{F}\ (f(r\cos(\theta+\phi), r\sin(\theta+\phi)))(u,v) =$$

$$F(\omega\cos(\theta+\phi), \omega\sin(\theta+\phi))$$

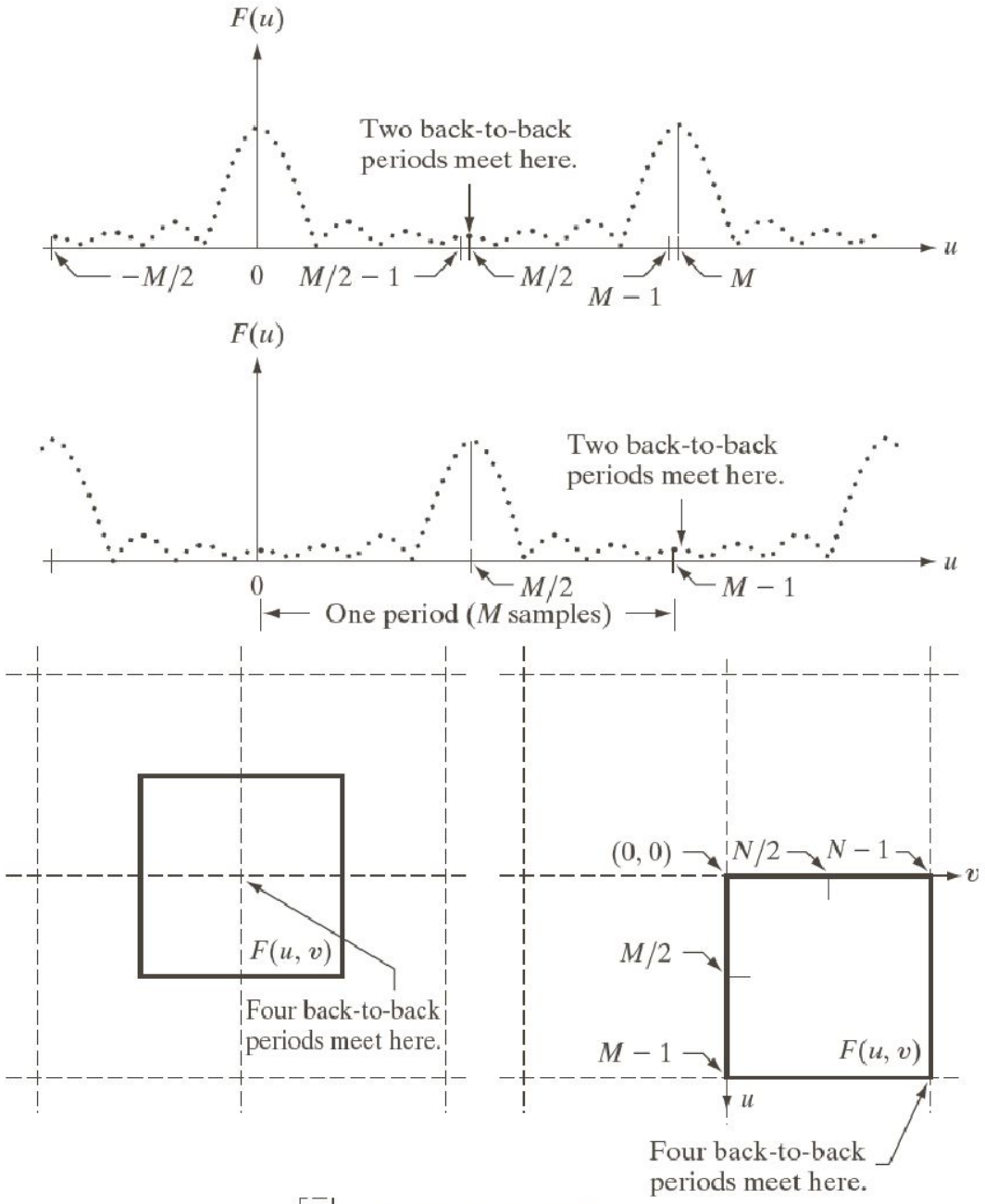
2D-DFT visualization in MATLAB

- In the formula for the DFT, the frequency range from $u = 0$ to $u = M-1$ actually represents two half periods back to back meeting at $M/2$ (see next slide).
- It is more convenient to instead view a complete period of the DFT instead.
- For that purpose we visualize $F(u-M/2)$ instead of $F(u)$ in 1D, and $F(u-W_1/2,v-W_2/2)$ instead of $F(u,v)$ in 2D.
- Thereby frequency $u = 0$ or $u = v = 0$ now occurs at the center of the displayed Fourier transform.
- Note that $F(u-W_1/2,v-W_2/2) = \mathcal{F}(f(x,y)(-1)^{x+y})(u,v)$, so the centering operation is easy to implement.



a
b
c
d

FIGURE 4.6
(a) Fourier transform of a band-limited function.
(b)–(d) Transforms of the corresponding sampled function under the conditions of over-sampling, critically-sampling, and under-sampling, respectively.



a
b
c d

FIGURE 4.23 Centering the Fourier transform. (a) A 1-D DFT showing an infinite number of periods. (b) Shifted DFT obtained by multiplying $f(x)$ by $(-1)^x$ before computing $F(u)$. (c) A 2-D DFT showing an infinite number of periods. The solid area is the $M \times N$ data array, $F(u, v)$, obtained with Eq. (4.5-15). This array consists of four quarter periods. (d) A Shifted DFT obtained by multiplying $f(x, y)$ by $(-1)^{x+y}$ before computing $F(u, v)$. The data now contains one complete, centered period, as in (b).

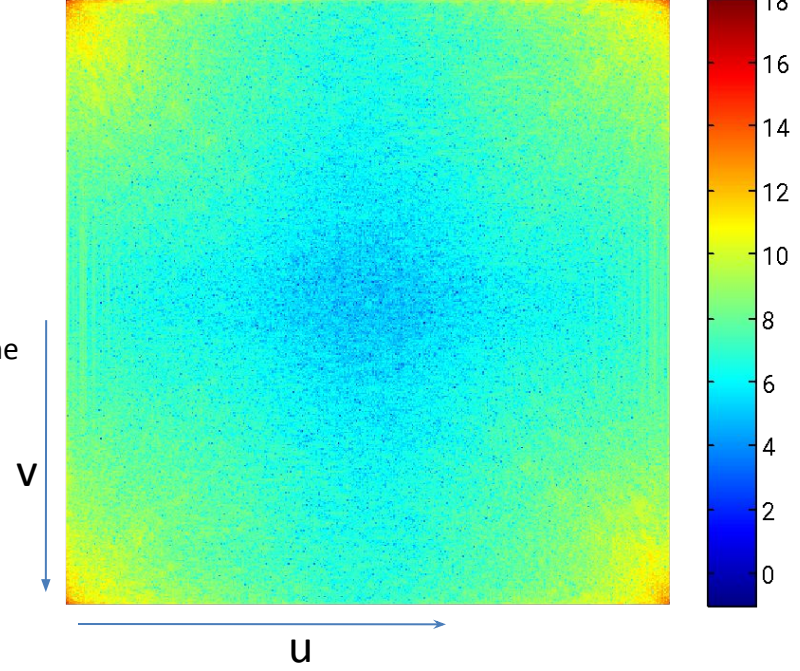
2D-DFT visualization in MATLAB

- To visualize the DFT of an image, we visualize the magnitude of the DFT.
- The Fourier transform is first centered as mentioned on the previous slide.
- Due to the large range of magnitude values, the DFT magnitudes are visualized on a logarithmic scale, i.e. we view $\log(|F(u,v)|+1)$ where the 1 is added for stability.

Without centering the fft

```
fim2 = fft2(im2);
absfim2 = log(abs(fim2)+1);
imshow(absfim1,[-1 18]);
colormap (jet); colorbar;
```

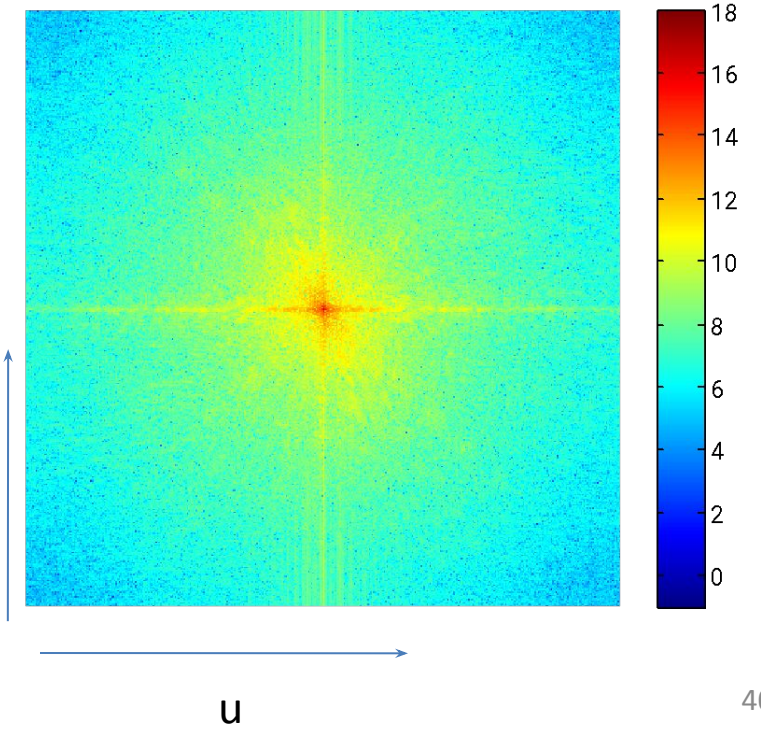
The (0,0) frequency lies at the top left corner



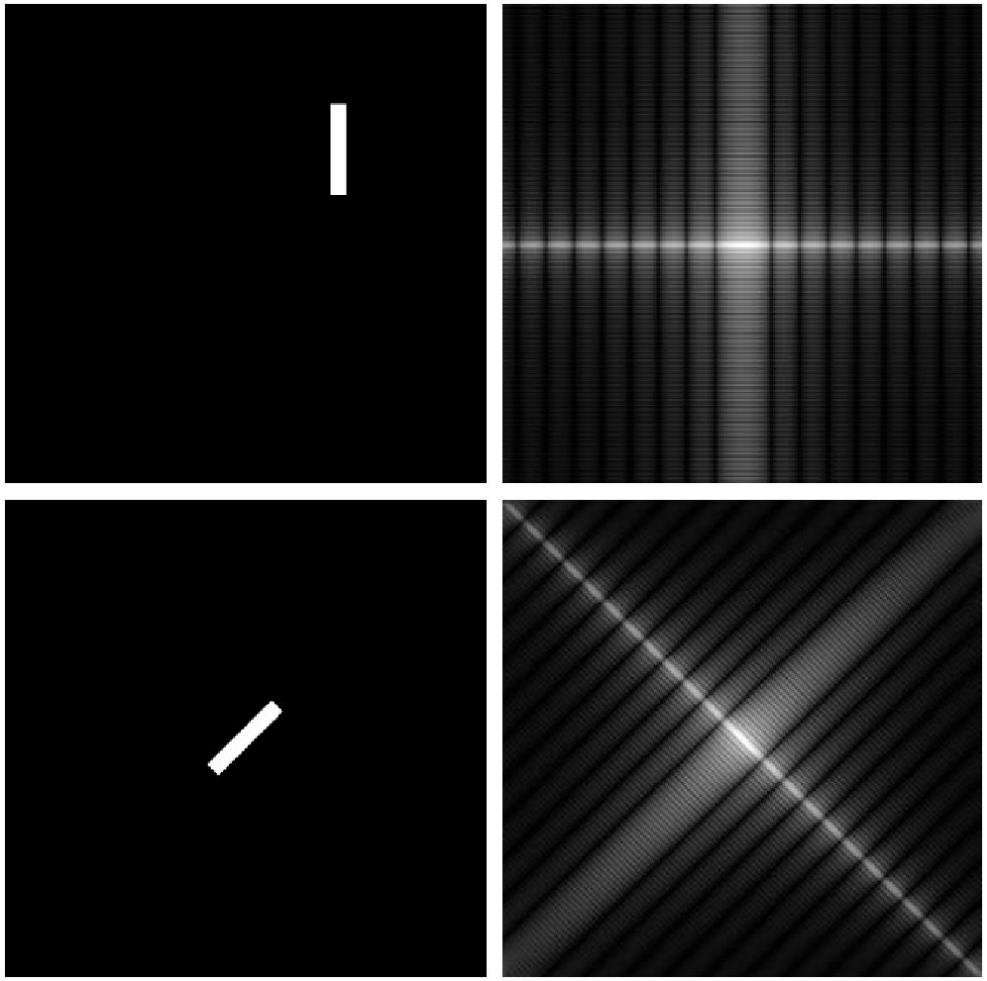
With centering the fft

```
fim2 = fftshift(fft2(im2));
absfim2 = log(abs(fim2)+1);
imshow(absfim1,[-1 18]);
colormap (jet); colorbar;
```

The (0,0) frequency lies in the
Center.



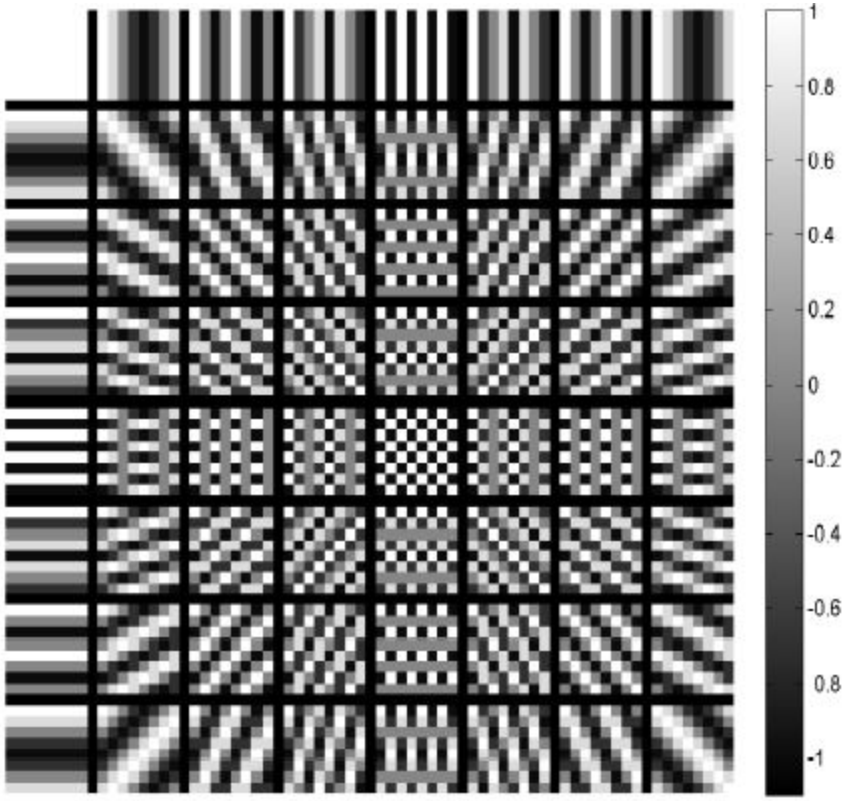
“Viewing” the rotation property of the DFT



a b
c d

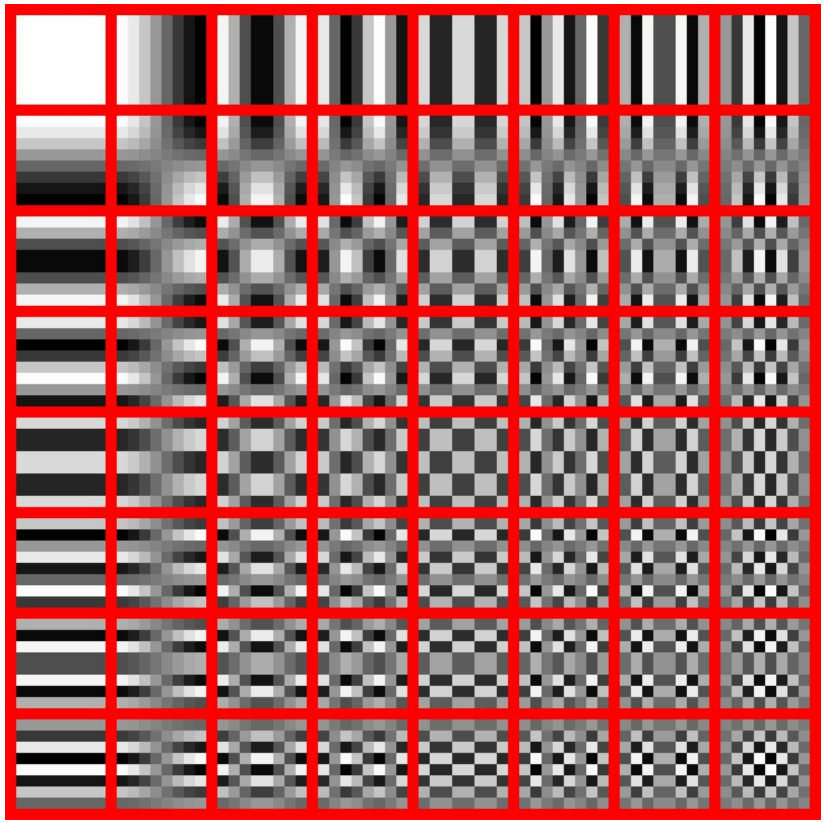
FIGURE 4.25
(a) The rectangle in Fig. 4.24(a) translated, and (b) the corresponding spectrum. (c) Rotated rectangle, and (d) the corresponding spectrum. The spectrum corresponding to the translated rectangle is identical to the spectrum corresponding to the original image in Fig. 4.24(a).

Visualizing DFT bases in 2D



Real part of DFT

Each image represents a column of the 2D DFT matrix reshaped to form an image. Only the real component is displayed here. Our theory says that any discrete image can be accurately represented as a linear combination of such basis images. The coefficients of the linear combination may be positive, negative or zero, and real or complex in value. Note that pixel (x,y) of the image for frequency (u,v) represents: Exp(j 2*pi*(ux + vy)/N)



Discrete cosine transform (for comparison only)

http://en.wikipedia.org/wiki/JPEG

2D convolution

- In MATLAB, 2D convolution can be implemented using the routine **conv2**.

- This can be very expensive if the signals you wish to convolve another with, are of large size.

- Hence one resorts to the convolution theorem which holds in 2D as well.

2D convolution

- The convolution theorem applies to 2D-DFTs as well:

  F $(f * h)(u,v) = F(u,v)H(u,v)$

- Consider an image *f* of size W_1 x W_2 which you want to convolve with a kernel *k* of size K_1 x K_2 using the DFT method.
- Then you should symmetrically zero-pad *f* and *k* so that they acquire size (W_1+K_1-1) x (W_2+K_2-1).
- Compute the DFTs of the zero-padded images using the FFT algorithm, point-wise multiply them and obtain the IDFT of the result.
- Extract the central W_1 x W_2 portion of the result – for the final answer.

Image Filtering: Frequency domain

- You have studied image filters of various types: mean filter, Gaussian filter, bilateral filter, patch-based filter.

- The former two are linear filters and the latter two aren’t.

- Linear filters are represented using convolutions and hence have a frequency domain interpretation as seen on the previous slides.

Image Filtering: Frequency domain

- Hence such filters can also be designed in the frequency domain.
- In certain applications, it is in fact more intuitive to design filters directly in the frequency domain.
- Why? Because you get to design directly which frequency components to weaken/eliminate and which ones to boost, and by how much.

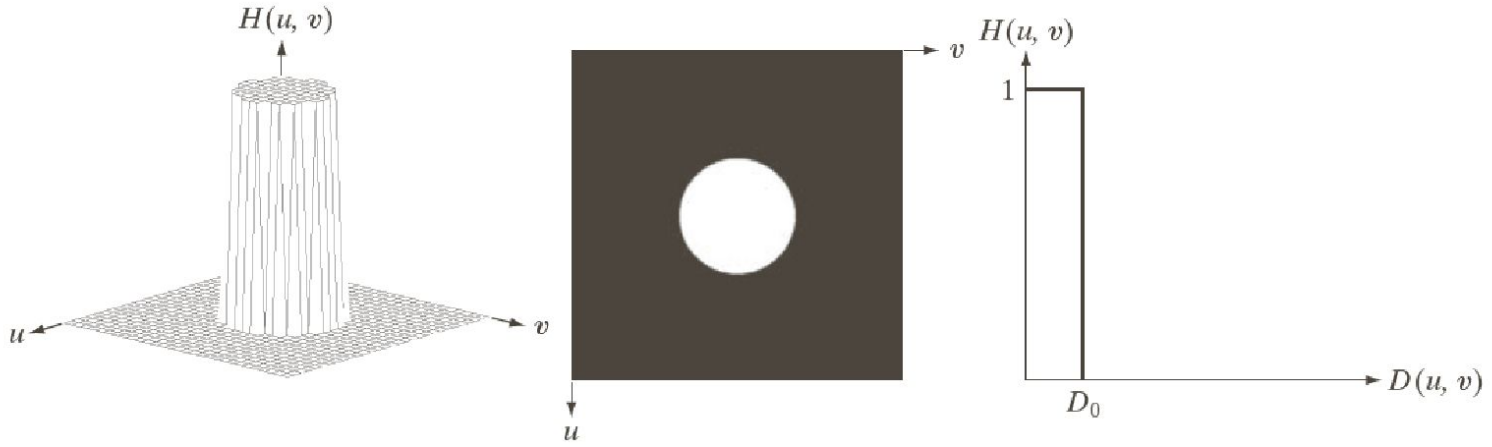
Low pass filters

- Edges and fine textures in images contribute significantly to the high frequency content of an image.
- When you smooth/blur an image, these edges and textures are weakened (or removed).
- Such filters allow only the low frequencies to remain intact and are called as **low pass filters**.
- In the frequency domain, an ideal low pass filter can be represented as follows:

$$H(u,v) = 1, \text{if } u^2 + v^2 \le D^2$$
$$= 0 \text{ otherwise}$$

Note: we are assuming (0,0) to be the center (lowest) frequency. Frequencies outside a radius of *D* from the center frequency are eliminated.

Low pass filters



a b c

FIGURE 4.40 (a) Perspective plot of an ideal lowpass-filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross section.

Low pass filters

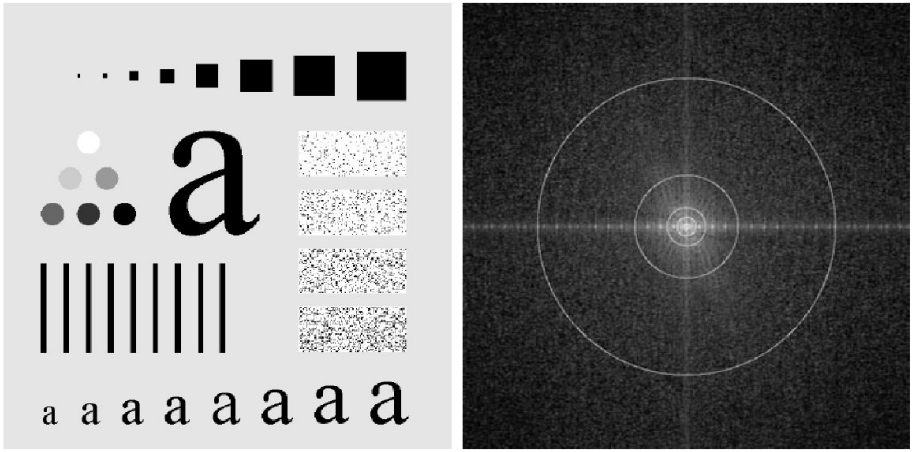
- To apply such a filter to an image *f* to create a filtered image *g*, we do as follows:

$$H(u,v) = 1, \text{if } u^2 + v^2 \leq D^2$$
$$= 0 \text{ otherwise}$$
$$G(u,v) = F(u,v)H(u,v)$$

- *D* is a design parameter of the filter – often called the **cutoff frequency**.

- This is called the **ideal** low pass filter as it completely eliminates frequencies outside the chosen radius.

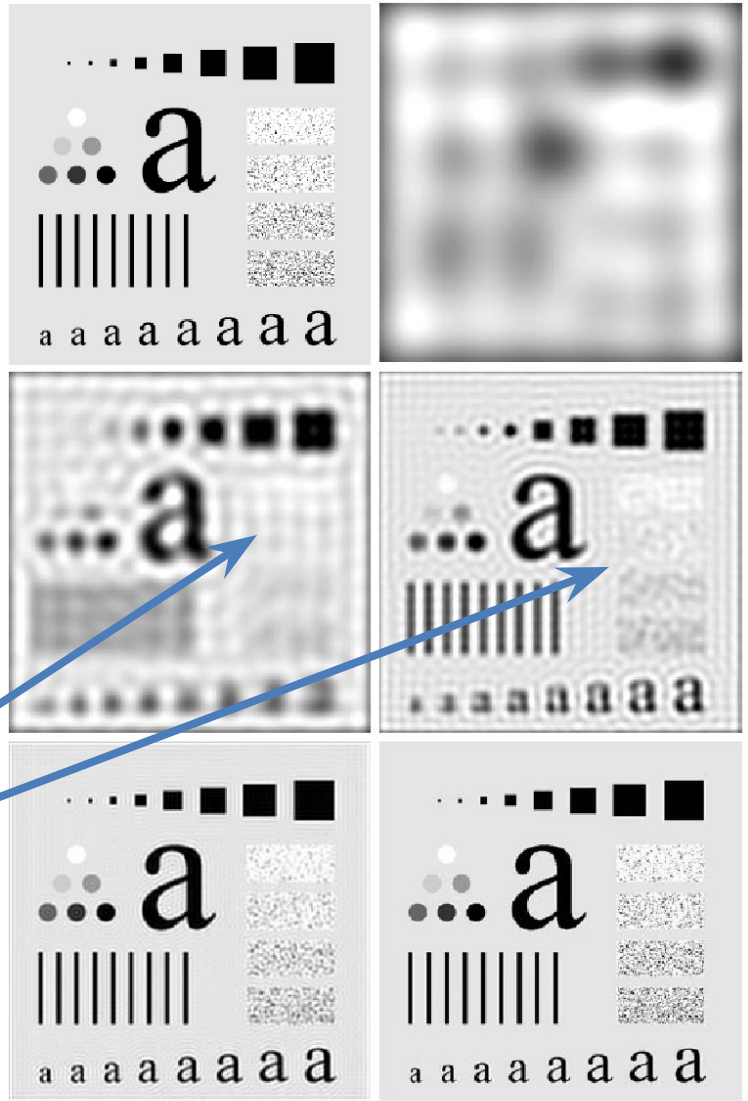
Low pass filters



a b

FIGURE 4.41 (a) Test pattern of size 688 × 688 pixels, and (b) its Fourier spectrum. The spectrum is double the image size due to padding but is shown in half size so that it fits in the page. The superimposed circles have radii equal to 10, 30, 60, 160, and 460 with respect to the full-size spectrum image. These radii enclose 87.0, 93.1, 95.7, 97.8, and 99.2% of the padded image power, respectively.

Notice the ringing artifacts around the edges!

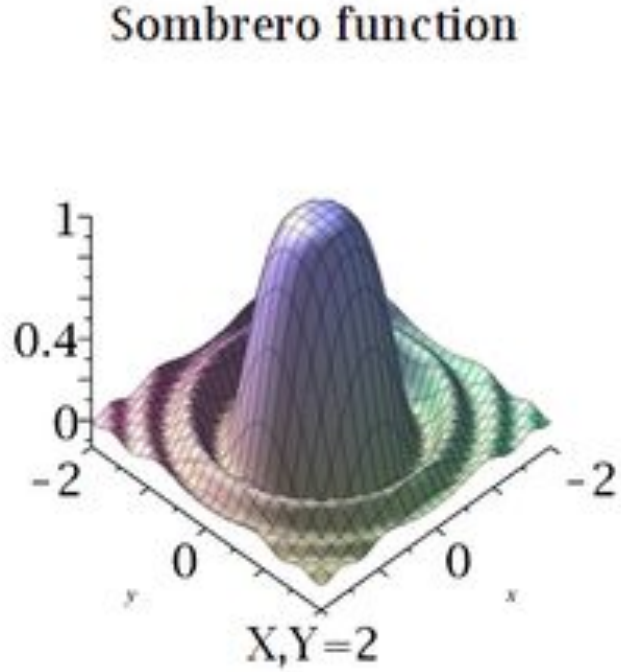


a b
c d
e f

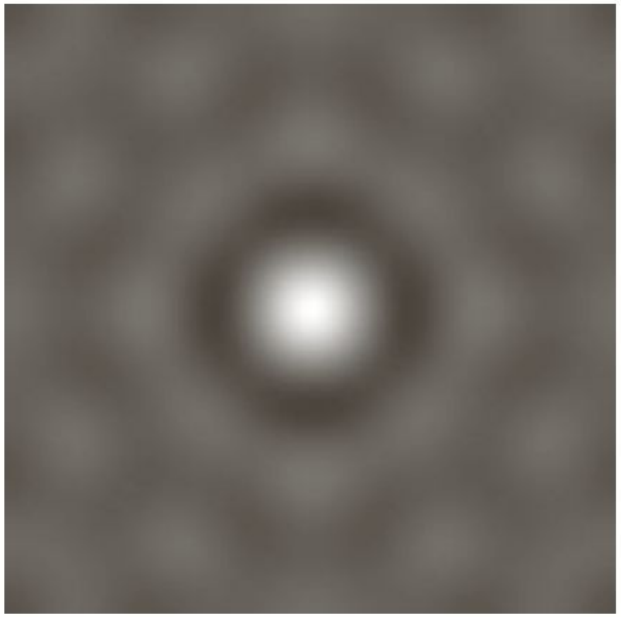
FIGURE 4.42 (a) Original image. (b)–(f) Results of filtering using ILPFs with cutoff frequencies set at radii values 10, 30, 60, 160, and 460, as shown in Fig. 4.41(b). The power removed by these filters was 13, 6.9, 4.3, 2.2, and 0.8% of the total, respectively.

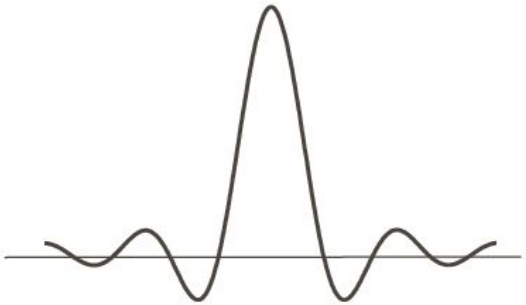
Low pass filters

- The ringing artifacts can be explained by the convolution theorem.
- The corresponding spatial domain filter is called the **jinc** function.
- The jinc is a 2D and circularly symmetric version of the sinc function – see also the next slide.
- Any cross section of the jinc is basically a sinc function.



https://en.wikipedia.org/wiki/Sombrero_function





a b

FIGURE 4.43
(a) Representation in the spatial domain of an ILPF of radius 5 and size 1000×1000. (b) Intensity profile of a horizontal line passing through the center of the image.

An image can be regarded as a weighted sum of Kronecker delta functions (impulses). Convolving with a sinc function means placing copies of the sinc function at each impulse. The central lobe of the sinc function causes the blurring and the smaller lobes give rise to the “ripple artifacts” or ringing.

Low pass filters: other types

- These ringing artifacts can be quite undesirable.

- Hence the ideal low pass filter is replaced by other types of low pass filters which **weaken** but do not totally eliminate the higher frequencies.

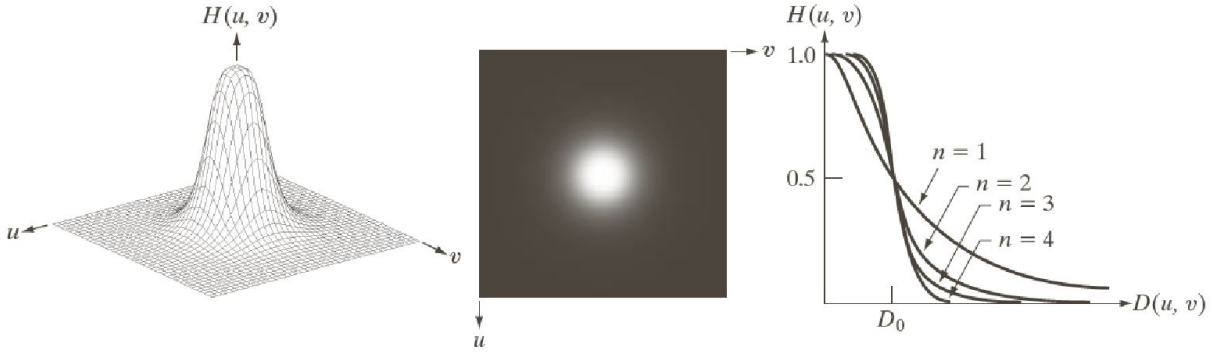
- For example:

$$H(u,v) = \frac{1}{1+(\sqrt{u^2+v^2}\,/\,D)^{2n}}, (n, D) = \text{filter parameters}$$

Butterworth filter:
D = cutoff frequency,
n = order of the filter

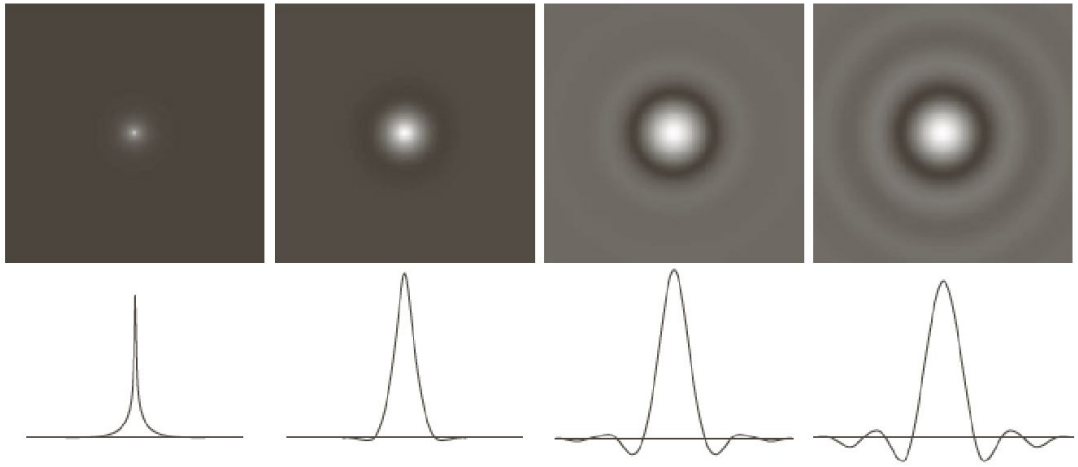
$$H(u,v) = \exp(-(u^2+v^2)/(2\sigma^2)), (\sigma) = \text{filter parameters}$$

Gaussian filter



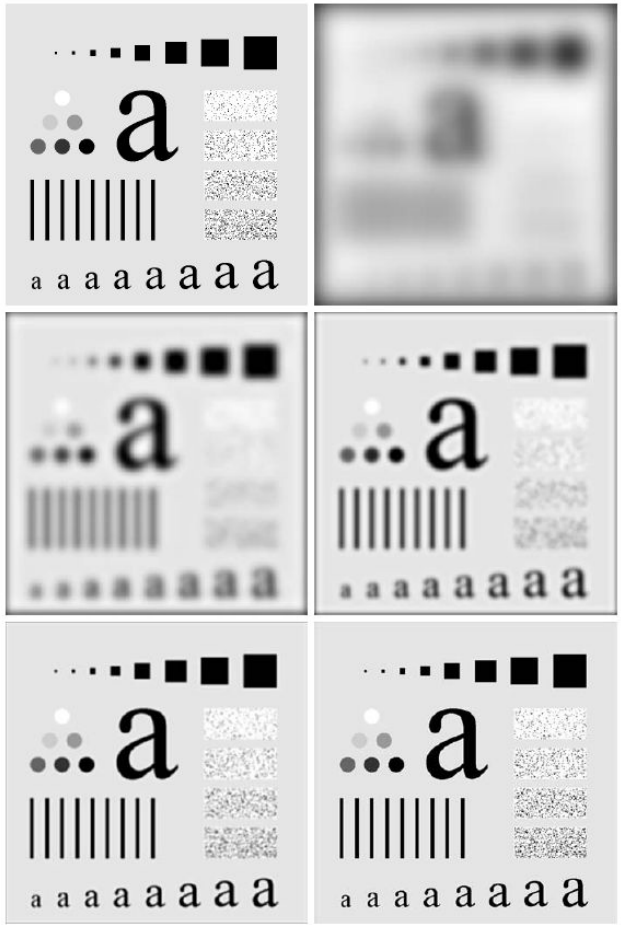
a b c

FIGURE 4.44 (a) Perspective plot of a Butterworth lowpass-filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections of orders 1 through 4.



a b c d

FIGURE 4.46 (a)–(d) Spatial representation of BLPFs of order 1, 2, 5, and 20, and corresponding intensity profiles through the center of the filters (the size in all cases is 1000 × 1000 and the cutoff frequency is 5). Observe how ringing increases as a function of filter order.



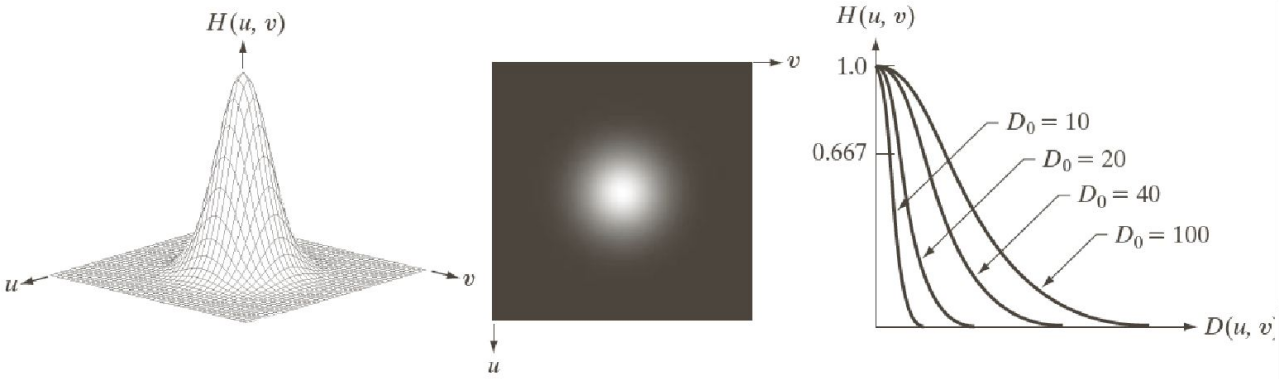
a b
c d
e f

FIGURE 4.45 (a) Original image. (b)–(f) Results of filtering using BLPFs of order 2, with cutoff frequencies at the radii shown in Fig. 4.41. Compare with Fig. 4.42.

Low pass filters: Butterworth

- As *D* increases, the Butterworth filter allows more frequencies to pass through without weakening (see previous slide).

- As the order *n* increases, the Butterworth filter weakens the higher frequencies more aggressively (why?) – which actually increases ringing artifacts (see previous slide).

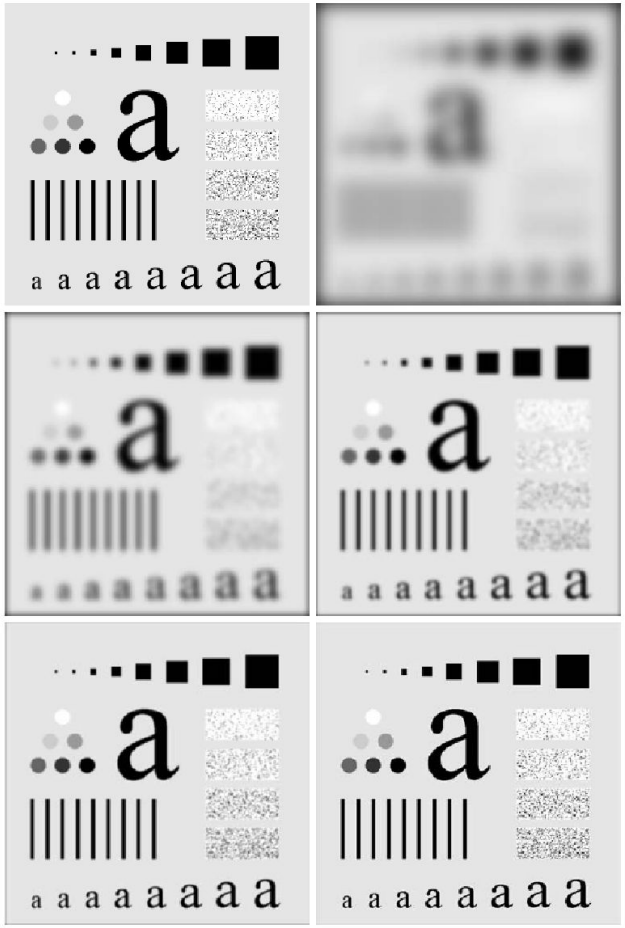


a b c

FIGURE 4.47 (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of D_0.

a b c

FIGURE 4.50 (a) Original image (784 × 732 pixels). (b) Result of filtering using a GLPF with $D_0 = 100$. (c) Result of filtering using a GLPF with $D_0 = 80$. Note the reduction in fine skin lines in the magnified sections in (b) and (c).



a b
c d
e f

FIGURE 4.48 (a) Original image. (b)–(f) Results of filtering using GLPFs with cutoff frequencies at the radii shown in Fig. 4.41. Compare with Figs. 4.42 and 4.45.

Low pass filters: Gaussian

- The spatial domain representation for the Gaussian LPF is basically a Gaussian.

- In fact, the following are Fourier transform pairs:

$$g(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(\frac{-x^2}{2\sigma^2}\right) \leftrightarrow G(u) = \exp\left(\frac{-u^2}{2(1/\sigma^2)}\right)$$

http://www.cse.yorku.ca/~kosta/CompVis_Notes/fourier_transform_Gaussian.pdf

Notice from the equations that a spatial domain Gaussian with high standard deviation corresponds to a frequency domain Gaussian with low cut off frequency! The extreme case of this is a constant intensity signal (Gaussian with very high standard deviation – infinite as such) whose Fourier transform is an impulse at the origin of the Frequency plane). Another extreme case is a spatial domain signal which is just an impulse – its Fourier transform has constant amplitude everywhere.

High pass filter

- It is a filter that allows high frequencies and eliminates or weakens the lower frequencies.

- The equation for the frequency response of an ideal high pass filter is given as:

$$H(u,v) = 0, \text{if } u^2 + v^2 \leq D^2$$
$$= 1 \text{ otherwise}$$

- In fact, given an LPF, an HPF can be constructed from it using:

$$H_{HP}(u,v) = 1 - H_{LP}(u,v)$$

TABLE 4.5

Highpass filters. D_0 is the cutoff frequency and n is the order of the Butterworth filter.

Ideal	Butterworth	Gaussian
$H(u,v) = \begin{cases} 1 & \text{if } D(u,v) \le D_0 \\ 0 & \text{if } D(u,v) > D_0 \end{cases}$	$H(u,v) = \dfrac{1}{1 + [D_0/D(u,v)]^{2n}}$	$H(u,v) = 1 - e^{-D^2(u,v)/2D_0^2}$

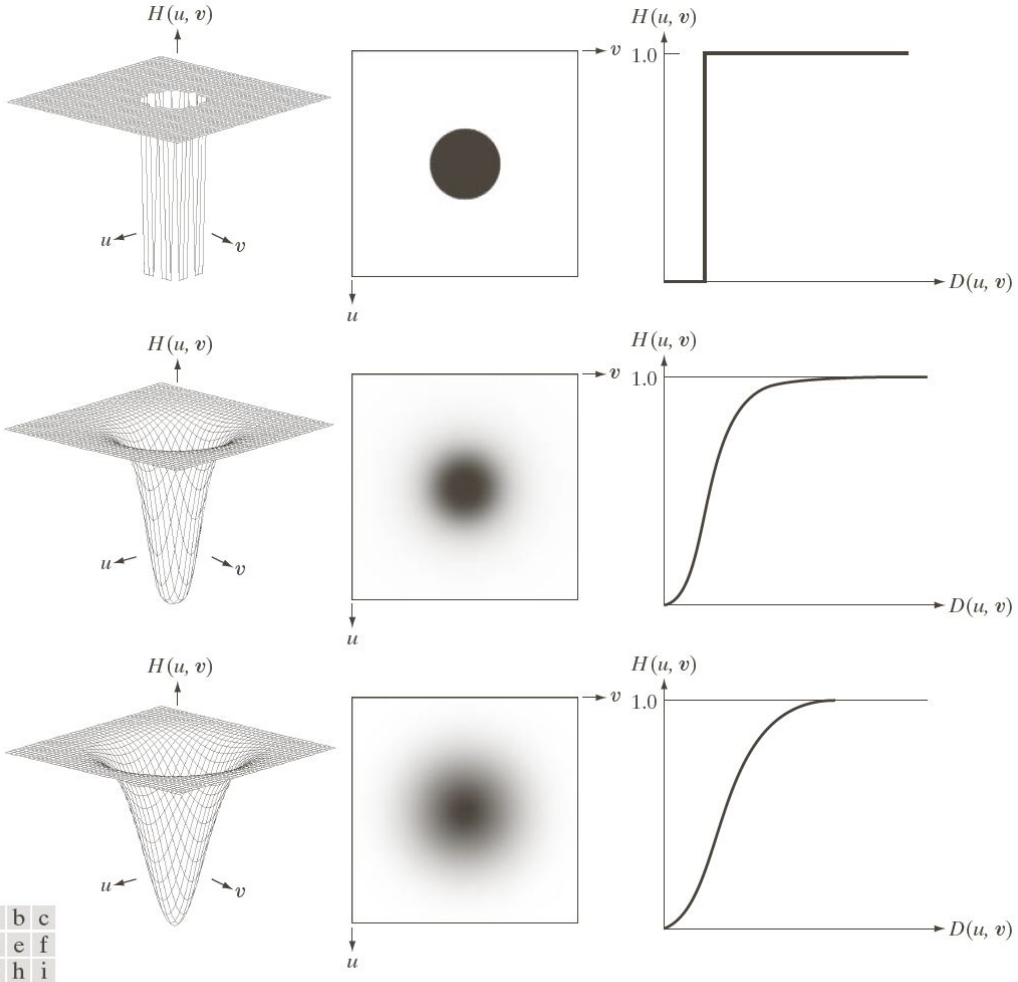
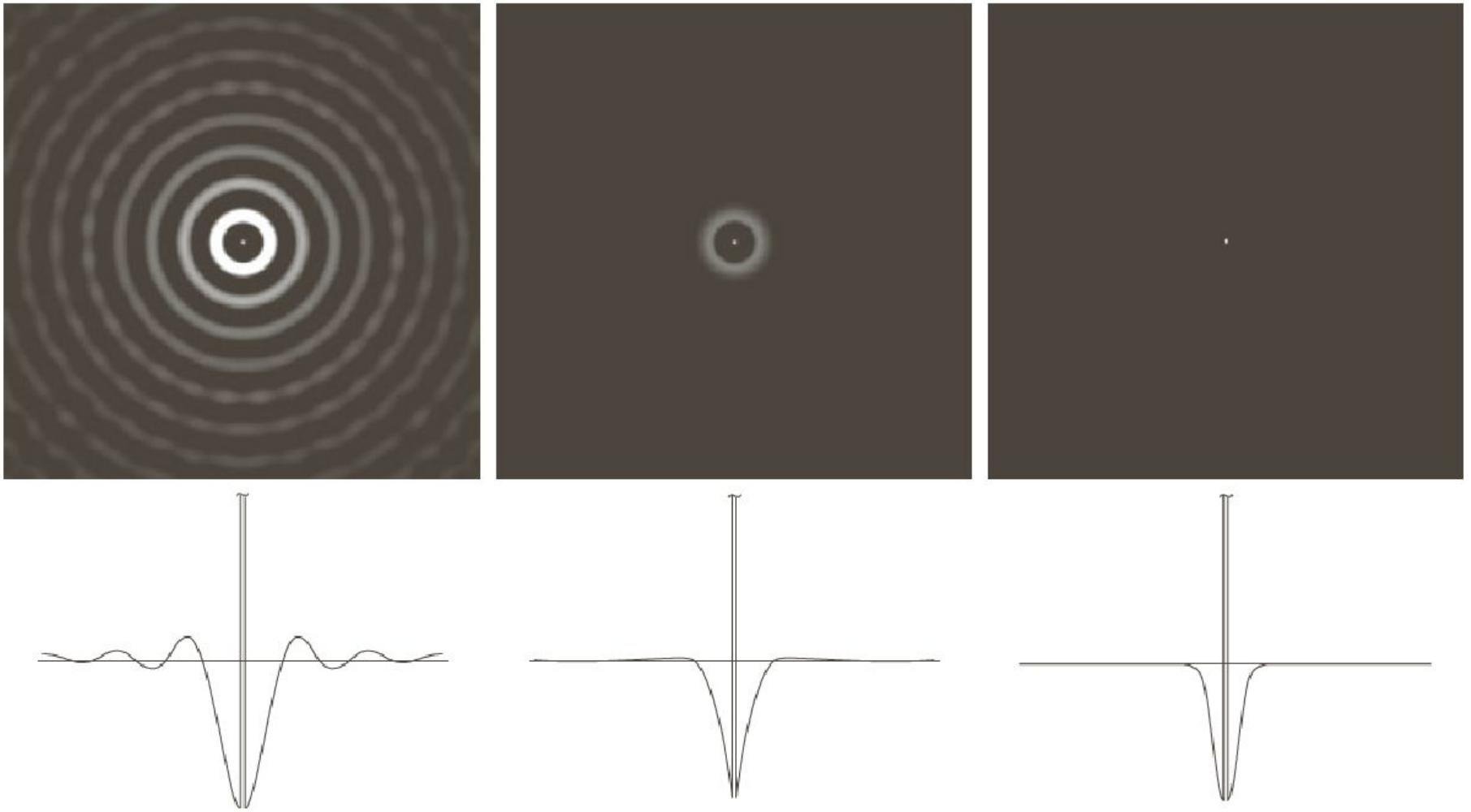


FIGURE 4.52 Top row: Perspective plot, image representation, and cross section of a typical ideal highpass filter. Middle and bottom rows: The same sequence for typical Butterworth and Gaussian highpass filters.



a b c

FIGURE 4.53 Spatial representation of typical (a) ideal, (b) Butterworth, and (c) Gaussian frequency domain highpass filters, and corresponding intensity profiles through their centers.

Ringing artifacts for an Ideal HPF!

a b c

FIGURE 4.54 Results of highpass filtering the image in Fig. 4.41(a) using an IHPF with $D_0 = 30$, 60, and 160.

Ringing artifacts gone!

a b c

FIGURE 4.55 Results of highpass filtering the image in Fig. 4.41(a) using a BHPF of order 2 with $D_0 = 30$, 60, and 160, corresponding to the circles in Fig. 4.41(b). These results are much smoother than those obtained with an IHPF.

Ringing artifacts for an Ideal HPF!

a b c

FIGURE 4.54 Results of highpass filtering the image in Fig. 4.41(a) using an IHPF with $D_0 = 30$, 60, and 160.

Ringing artifacts gone!

a b c

FIGURE 4.56 Results of highpass filtering the image in Fig. 4.41(a) using a GHPF with $D_0 = 30$, 60, and 160, corresponding to the circles in Fig. 4.41(b). Compare with Figs. 4.54 and 4.55.

High pass filters

- Note that any gradient-based operation on images is essentially a type of high-pass filter.

- This includes first order derivative filters or the Laplacian filter.

- Why? Consider the first order derivative filter in y direction.

$$g(x,y) = I(x,y+1) - I(x,y)$$
$$G(u,v) = I(u,v)(e^{j2\pi v/M} - 1) \longrightarrow G(u,0) = 0$$

High pass filters

- Consider the Laplacian filter:

$$g(x,y) = (I(x,y+1) + I(x+1,y) + I(x,y-1) + I(x-1,y) - 4I(x,y))/4$$

$$G(u,v) = \frac{(e^{-j2\pi v/M} + e^{-j2\pi u/M} + e^{j2\pi v/M} + e^{j2\pi u/M} - 4)}{4} I(u,v)$$

$$G(0,0) = 0$$

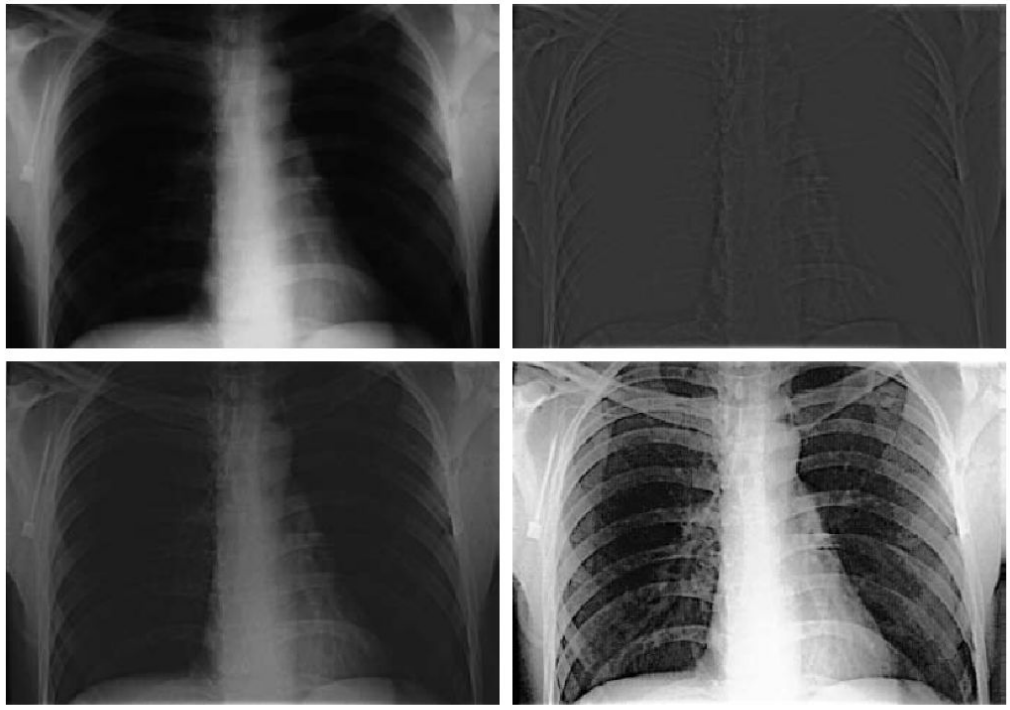
Boosting high frequencies

- In some applications, you neither wish to weaken the higher frequencies nor the lower frequencies.

- For example, in image sharpening applications you wish to **boost** the edges or textures (basically higher frequencies).

- In that case you wish to perform operations of the following form:

$$g(x,y) = \mathsf{F}^{-1}[(1 + kH_{HP}(u,v))F(u,v)]$$



a b
c d

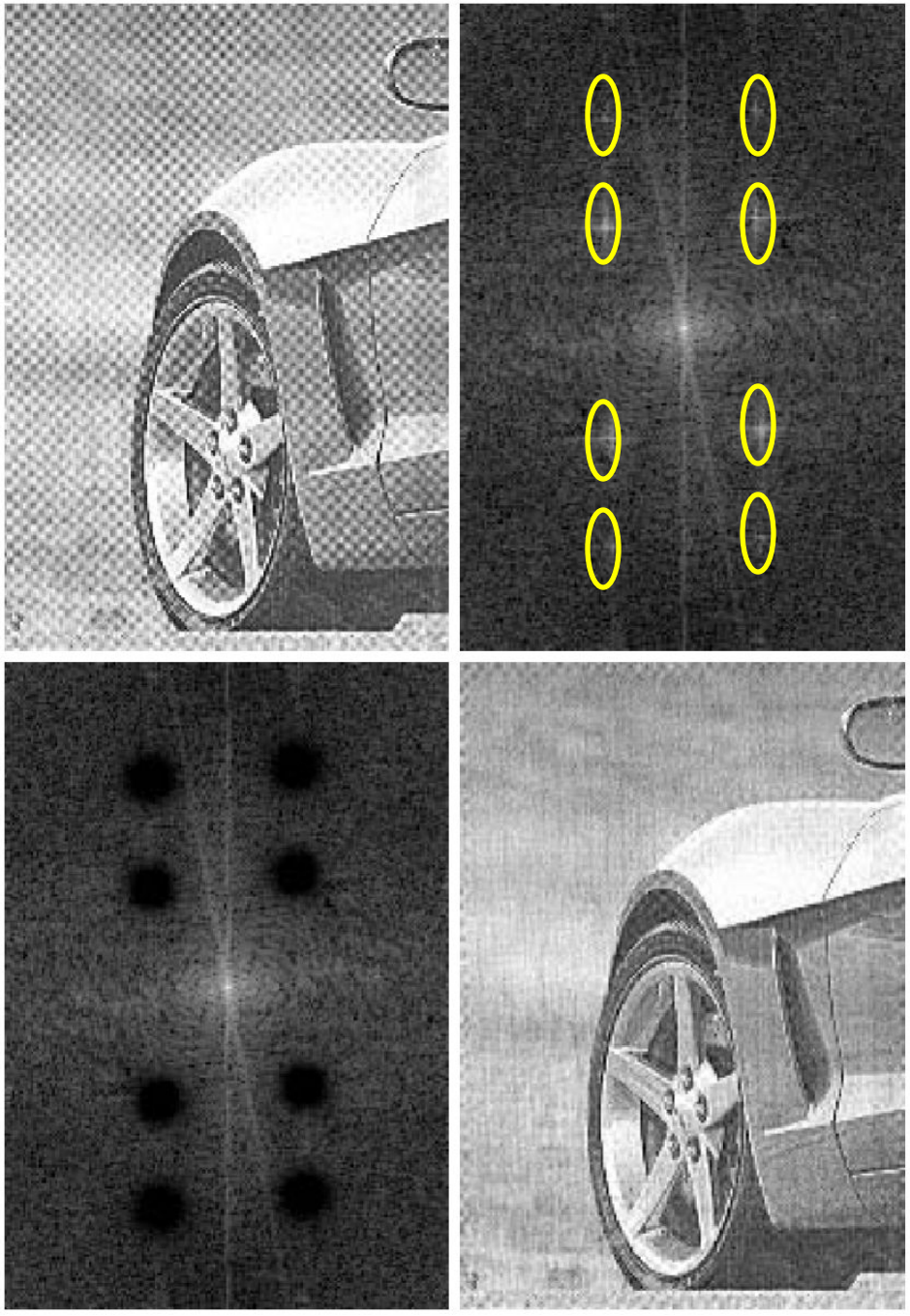
FIGURE 4.59 (a) A chest X-ray image. (b) Result of highpass filtering with a Gaussian filter. (c) Result of high-frequency-emphasis filtering using the same filter. (d) Result of performing histogram equalization on (c). (Original image courtesy of Dr. Thomas R. Gest, Division of Anatomical Sciences, University of Michigan Medical School.)

a b

FIGURE 4.58
(a) Original, blurry image. (b) Image enhanced using the Laplacian in the frequency domain. Compare with Fig. 3.38(e).

Notch filters

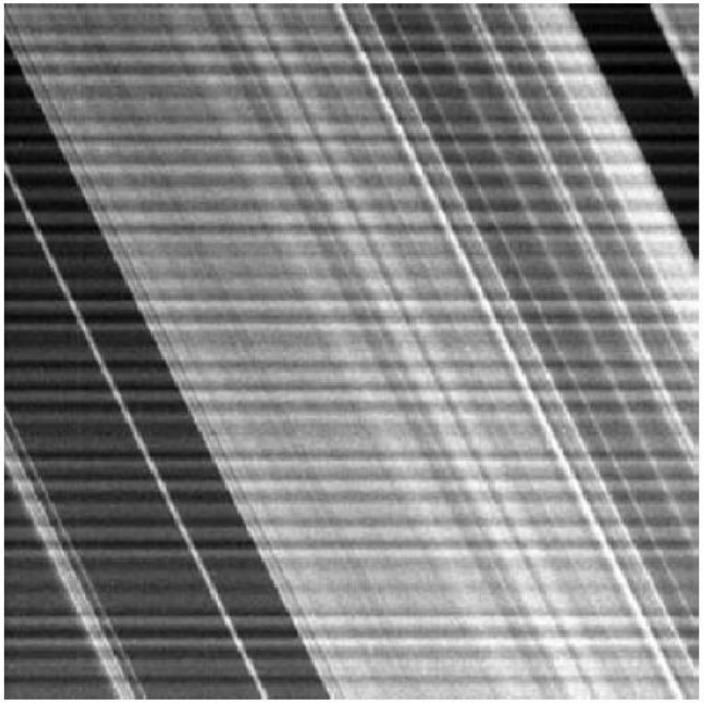
- There are applications where images are presented with **periodic noise patterns** – e.g. images scanned from old newspapers (see next slide).
- Usually, the amplitudes of the Fourier components of natural images are seen to decay with frequency.
- The Fourier transform of images with periodic patterns show unnatural peaks – unlike ordinary natural images.
- Such images can be restored using **notch filters**, which basically weaken or eliminate these unnatural frequency components (or any frequency components specified by the user).

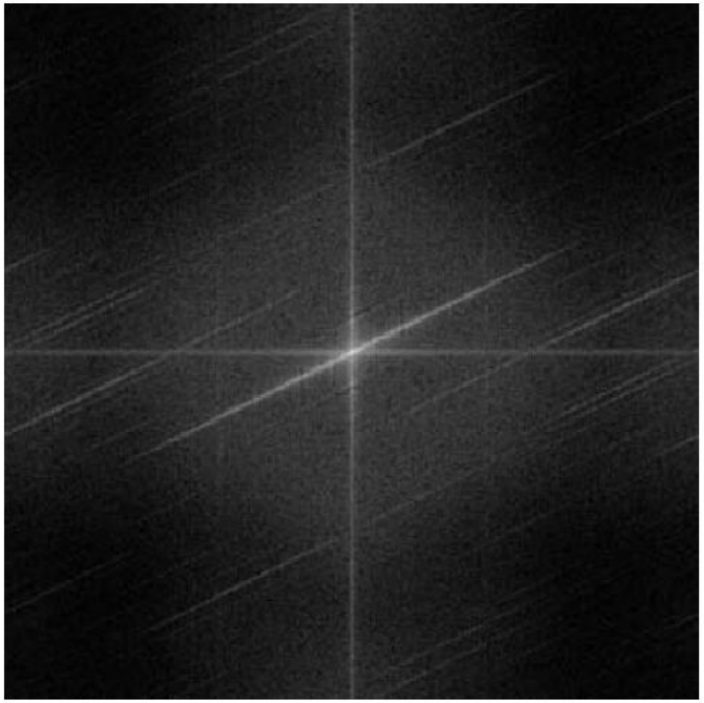


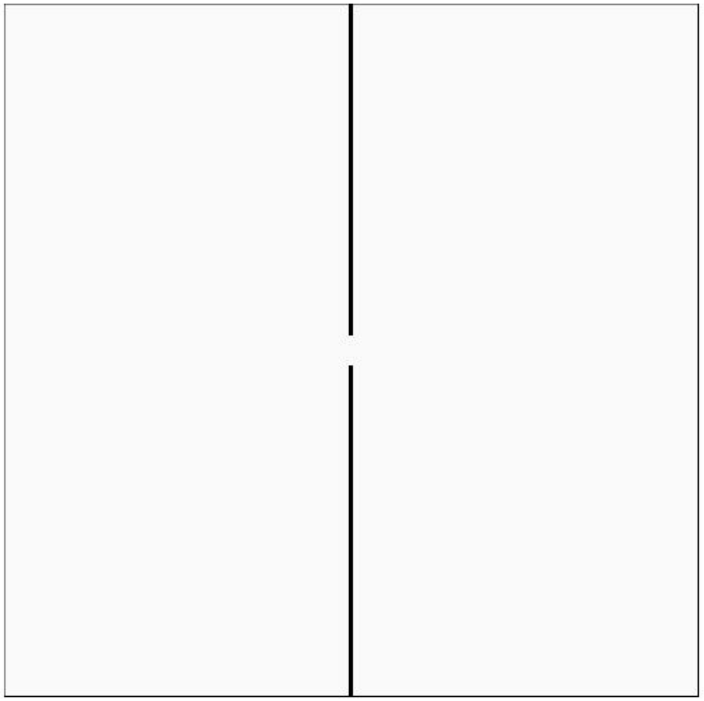
a b
c d

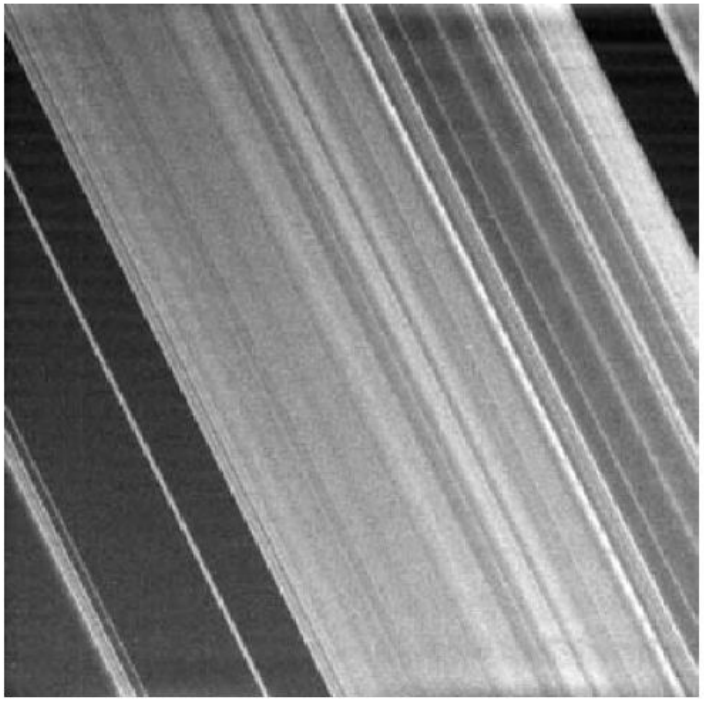
FIGURE 4.64
(a) Sampled newspaper image showing a moiré pattern. (b) Spectrum. (c) Butterworth notch reject filter multiplied by the Fourier transform. (d) Filtered image.

The yellow ellipses indicate unnatural peaks in the Fourier transform of the image with the superimposed interfering pattern. These are unnatural because typical images do not have such Fourier transforms. Therefore one can apply a notch filter to remove the interfering pattern.







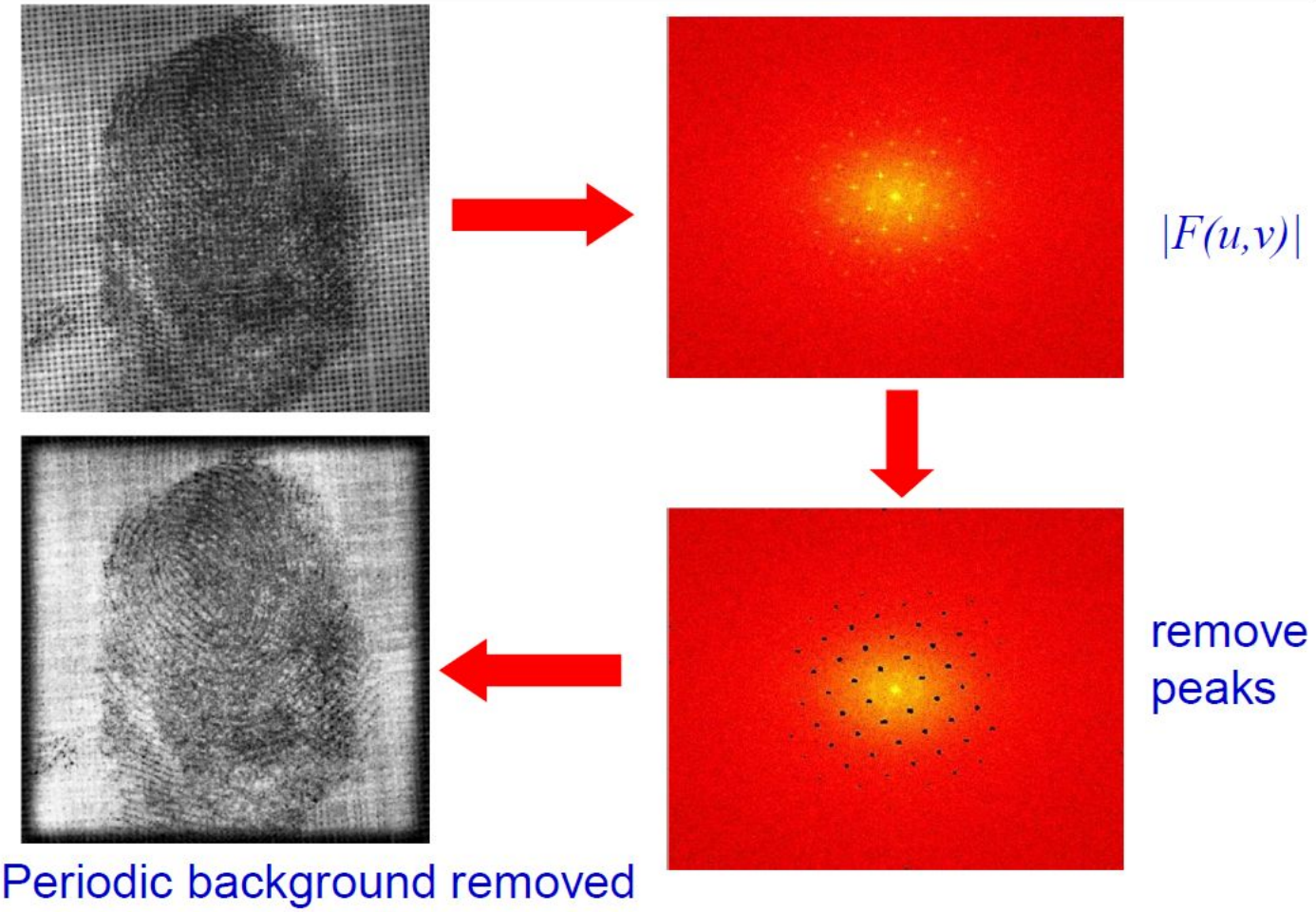


a b
c d

FIGURE 4.65
(a) 674 × 674 image of the Saturn rings showing nearly periodic interference. (b) Spectrum: The bursts of energy in the vertical axis near the origin correspond to the interference pattern. (c) A vertical notch reject filter. (d) Result of filtering. The thin black border in (c) was added for clarity; it is not part of the data. (Original image courtesy of Dr. Robert A. West, NASA/JPL.)

Courtesy: http://www.robots.ox.ac.uk/~az/lectures/ia/lect2.pdf

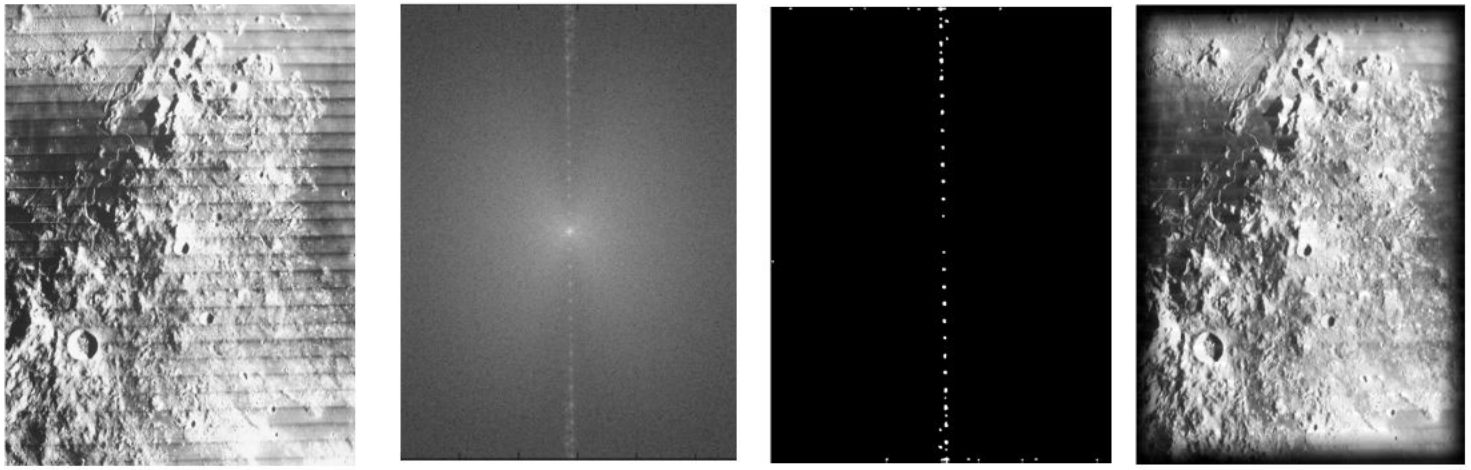
Example – Forensic application



Courtesy: http://www.robots.ox.ac.uk/~az/lectures/ia/lect2.pdf

Example – Image processing

Lunar orbital image (1966)



$|F(u,v)|$

remove peaks

join lines removed

Notch filters

- The expression for a notch filter frequency response is given as follows (Q = number of frequency components to suppress):

$$H_{NR}(u,v) = \prod_{i=1}^{Q} H_N(u,v;u^{(i)},v^{(i)},R)$$

$$H_{NR}(u,v;u^{(i)},v^{(i)},R) = 0 \text{ if } (u-u^{(i)})^2 + (v-v^{(i)})^2 \le R^2$$
$$= 1 \text{ otherwise}$$

Ideal notch reject filter

$$H_{NR}(u,v;u^{(i)},v^{(i)},R) = 1 - \exp\left(-((u-u^{(i)})^2 + (v-v^{(i)})^2)/R^2\right)$$

Gaussian notch reject filter

Algorithm for frequency domain filtering

- Consider you have to filter an image f of size H x W using one of the filters described in these slides.
- Zero pad the image symmetrically to create a new image f’ of size 2H x 2W.
- Compute the Fourier transform of f’ and center it using fftshift. Let the Fourier transform be F’(u,v).
- Design a frequency domain filter H(u,v) of size 2H x 2W with the zero frequency at index (H,W) of the 2D filter array.
- Compute the product F’(u,v)H(u,v).
- Compute the inverse Fourier transform of the product after applying ifftshift (to undo the effect of fftshift).
- Consider only the real part of the inverse Fourier transform and extract the central portion of size H x W.
- That gives you the final filtered image.
- Note that the zero padding is important. To understand the difference, see the results on the next slide with and without zero-padding.

Original image

Effect of Gaussian LPF with appropriate zero padding

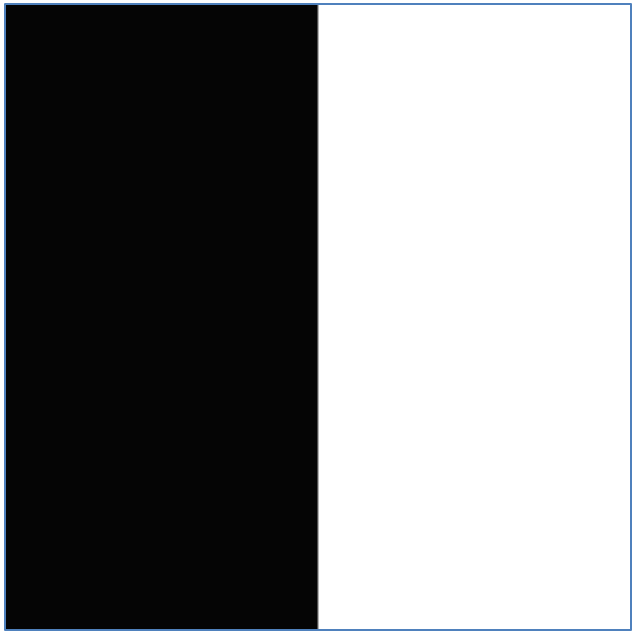
Effect of Gaussian LPF without appropriate zero padding – zero the border artifacts!

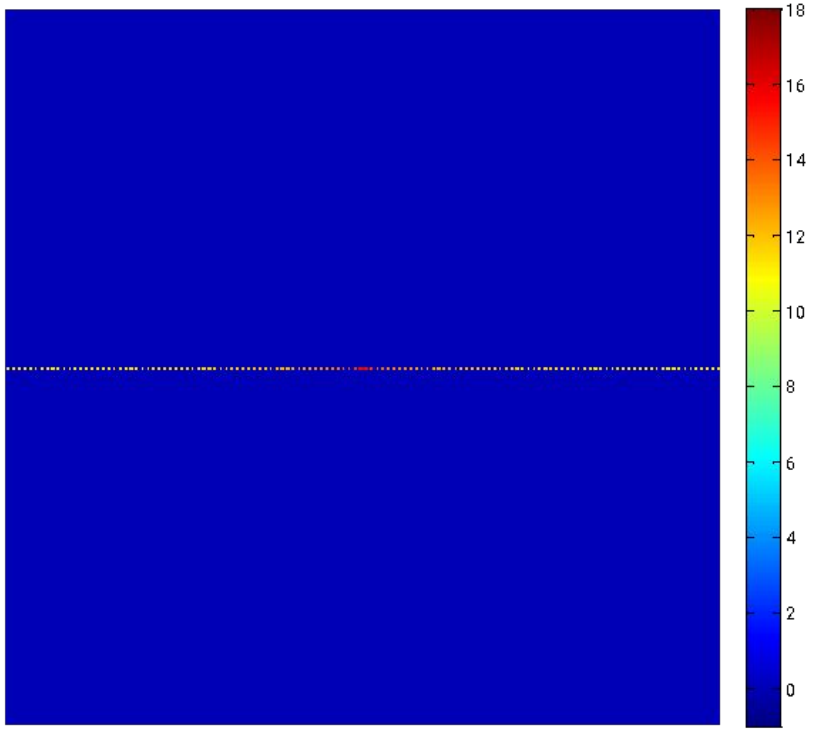
Interpreting the DFT of Natural Images

- In a few cases, one can guess some structural properties of an image by looking at its Fourier transform.

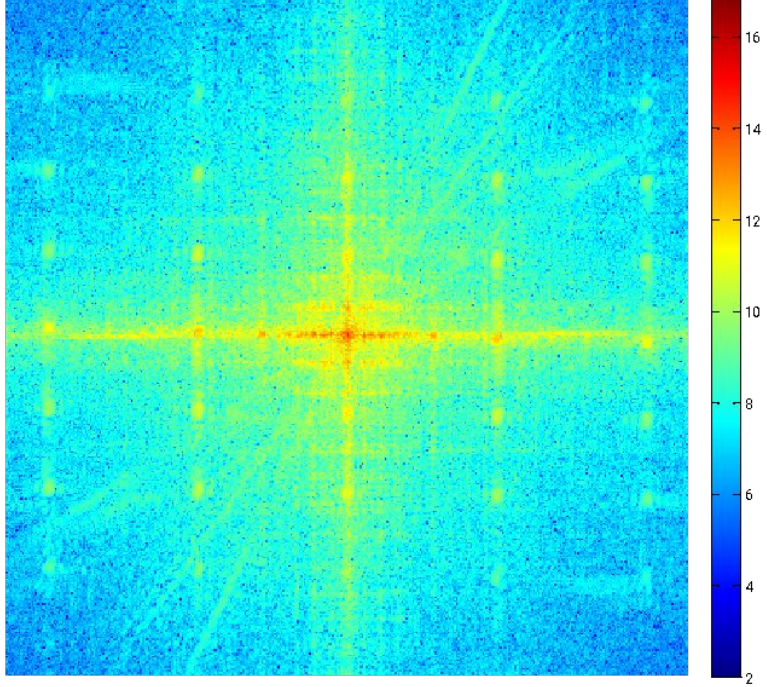
- Most natural images have stronger low frequency components as compared to higher frequency components. This is generally true, although its not a strictly monotonic relationship.

- For example, how does the DFT of a vertical edge image look like?

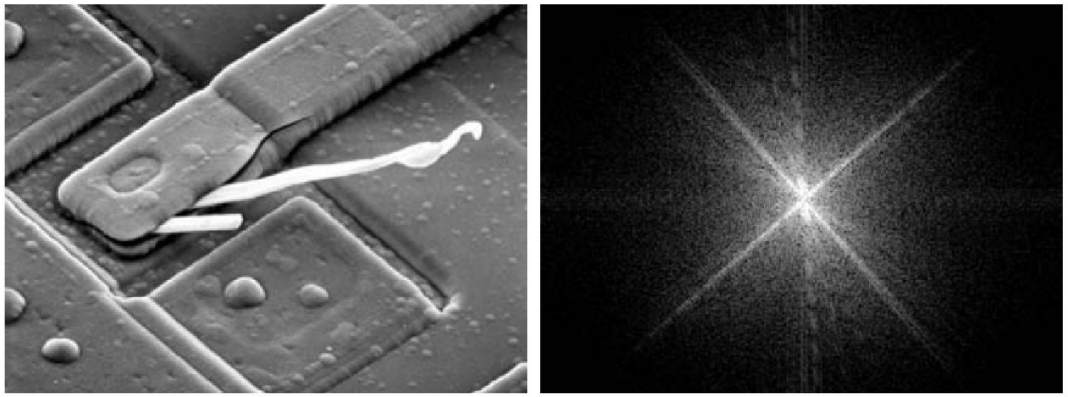




There is a sinc function along the *u* axis!



Strong horizontal and vertical edges = strong responses along *u* and *v* axes in the Fourier plane!



a b

FIGURE 4.29 (a) SEM image of a damaged integrated circuit. (b) Fourier spectrum of (a). (Original image courtesy of Dr. J. M. Hudak, Brockhouse Institute for Materials Research, McMaster University, Hamilton, Ontario, Canada.)

A strong edge in direction θ in XY plane = a strong response in the direction perpendicular to θ in UV plane

Interpreting the DFT

- For an image (or any 2D array) with Fourier transform F(u,v), F(0,0) is proportional to the average value of the image intensity – why?

- The DFT of a non-zero constant intensity image is basically an impulse at the zero frequency. What is the height of the impulse?

DFT limitations

- A strong response at frequency (u,v) (i.e. a large magnitude for F(u,v)) indicates the presence of a sinusoid with frequency (u,v) and with large amplitude *somewhere* in the image.

- In general, the Fourier transform *cannot* and *does not* tell us *where* in the image the sinusoid is located (read section 1 of http://users.rowan.edu/~polikar/WAVELETS/WTpart1.html).

- For that, you need to compute separate Fourier transforms over smaller regions of the image (Short time Fourier transform) – that will tell you which region(s) contained a particular sinusoidal component.

Fun with Fourier: Hybrid images

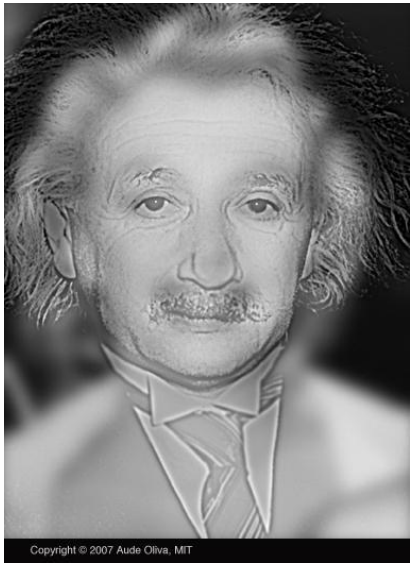
- Hybrid images are a superposition of an image J_1 with **strong low frequency components** and *weak higher frequency components*, and an image J_2 with **stronger higher frequency components** and weak lower frequency components.

- J_1 = apply LPF with some cutoff frequency *D* on an image.
- J_2 = apply HPF with some cutoff frequency *D* on another image.

- The appearance of the hybrid image changes with viewing distance!

Figure 3: Perceptual grouping between edges and blobs. The three images are perceived as a tiger when seen up-close and as a cheetah from far away. The differences among the three images is the degrees of alignment between the edges and blobs. Image a) contains two images superimposed without alignment. In image b), the eyes are aligned. And in image c), the head pose and the locations of eyes and mouth are aligned. Under proper alignment, the residual frequency band does not manage to build a percept. When seen up-close, it is difficult to see the cheetah's face, which is perfectly masked by the tiger's face. From far away, the tiger's edges are assimilated to the cheetah's face.

Tiger – when seen up-close, Cheetah – when seen from far



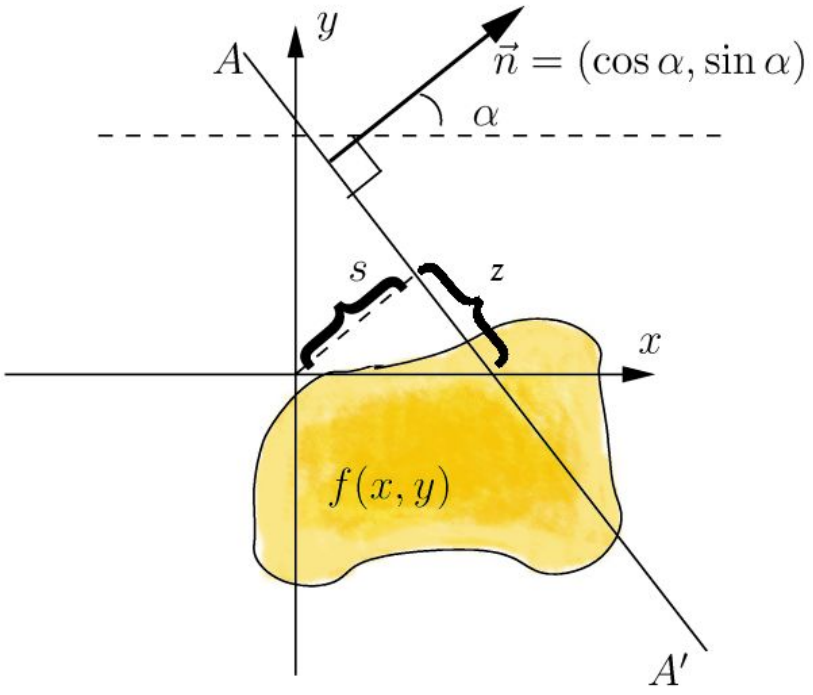
Einstein from nearby, Marilyn Monroe from far away (or if you squint)!

Refer to:
http://cvcl.mit.edu/hybrid_gallery/monroe_einstein.html
http://cvcl.mit.edu/publications/OlivaTorralb_Hybrid_Siggraph06.pdf (paper from a graphics journal!)

Fourier in action: tomographic reconstruction

- Tomography refers to the process of estimation of the internal structure of an object (tomos in Greek means interior).
- This is different from photographic imaging which images only the object surface.
- Tomography is typically accomplished via X Rays and the so called Radon transform.

Radon transform



Imagine a line was drawn through the 2D image in a certain direction α, and you integrated the intensity values along that line.

Now you repeat this for lines parallel to the original one but at different offsets.

Each such summation produces a "**bin**" of the **"tomographic projection"**.

The collection of bins form a 1D array which is called the **tomographic projection** or the **Radon transform** of the object in the direction α.

Reconstruction from the Radon Transform

- The Radon transform computation can be similarly repeated for multiple angles.

- The aim of tomographic reconstruction is to reconstruct the 2D image from a collection of its 1D tomographic projections in different angles.

Fourier transform of the Radon Transform

- The Radon transform is given as:

$$R(f) = g(\rho,\theta) = \int_{-\infty-\infty}^{+\infty+\infty} f(x,y)\delta(x\cos\theta + y\sin\theta - \rho)dxdy$$

- Its 1D Fourier transform w.r.t. ρ (keeping Θ fixed to some value) is given by:

$$G(\mu,\theta) = \int_{-\infty}^{+\infty} g(\rho,\theta)\exp(-j2\pi\mu\rho)d\rho$$

$$= \int_{-\infty-\infty-\infty}^{+\infty+\infty+\infty} f(x,y)\delta(x\cos\theta + y\sin\theta - \rho)\exp(-j2\pi\mu\rho)dxdyd\rho$$

$$= \int_{-\infty-\infty}^{+\infty+\infty} f(x,y)\left[\int_{-\infty}^{+\infty}\delta(x\cos\theta + y\sin\theta - \rho)\exp(-j2\pi\mu\rho)d\rho\right]dxdy$$

$$= \int_{-\infty-\infty}^{+\infty+\infty} f(x,y)\exp(-j2\pi\mu(x\cos\theta + y\sin\theta))dxdy$$

G(μ, Θ) is the Fourier transform of the projection of f(x,y) along some direction Θ.

Fourier transform of the Radon Transform

- Its 1D Fourier transform w.r.t. ρ (keeping Θ fixed) is given by:

$$G(\mu,\theta) = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y)\exp\left(-j2\pi(x\mu\cos\theta + y\mu\sin\theta)\right)dxdy$$

$$= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y)\exp\left(-j2\pi(xu+yv)\right)dxdy$$

$$\text{where we define } u = \mu\cos\theta, v = \mu\sin\theta$$

$$\therefore G(\mu,\theta) = \left[F(u,v)\right]_{u=\mu\cos\theta, v=\mu\sin\theta} = F(\mu\cos\theta, \mu\sin\theta)$$

The RHS of this equation is a slice of the 2D Fourier transform of f(x,y), i.e. F(u,v), along the angle Θ in the frequency plane, and passing through the origin

This equation above is called the **Projection Slice Theorem** or the **Fourier Slice Theorem**. It states that the Fourier transform of a projection of the 2D object along some direction Θ (i.e. G(μ, Θ)) is equal to a slice of the 2D Fourier transform of the object along the same direction Θ (in the frequency plane), passing through the origin.

Reconstruction from the Radon transfer

- The projection slice theorem thus tells you that you can fill up the 2D Fourier transform of the original image – since the 1D Fourier transform of the projection at angle θ is equal to a slice of the 2D Fourier transform of the image at the angle θ through the origin of the (u,v) plane.

- Thus given different projections, each at different angles, you can fill up the Fourier transform of the original image. The final inverse Fourier transform gives you a reconstruction of the image.

How does the Radon transform show up in tomographic acquisition?

- Let I_0 be the intensity of the X-ray beam sent through the object f at angle θ.
- The materials in the object absorb the X-rays partially.
- The remaining energy is sent out and is measured by a detector.
- As per Beer's law, the energy measured at the detector is given by:

$$q_\theta = I_0 \exp(-R_\theta f) \therefore \log\left(\frac{I_0}{q_\theta}\right) = R_\theta f$$

Measured projection vector (many bins) at angle θ

Radon transform of object f at angle θ

Fourier transforms in actual acquisition

- Computed tomography (in an indirect way via the Fourier slice theorem)
- Magnetic resonance imaging – the machine measures the Fourier transform of the object placed inside!
- Optics: in diffraction, only the squared magnitudes of the Fourier transform are measured – phase retrieval problem.

Applications of the Fourier transform in image processing

- Filtering: efficient implementation of the convolution
- Filtering: frequency domain filters
- Others: solving partial differential equations (not covered).

Appendix: Fourier transform of an impulse train

- Impulse train is given by:

$$s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} \delta(t - n\Delta T)$$

- It is a periodic signal with period ΔT, and hence it can be expressed by a Fourier series:

$$s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} c_n e^{j2\pi nt/\Delta T}; c_n = \frac{1}{\Delta T} \int_{-\Delta T/2}^{\Delta T/2} s_{\Delta T}(t) e^{-j2\pi nt/\Delta T} dt$$

Appendix: Fourier transform of an impulse train

- Simplifying the Fourier series coefficients, we have:

$$\therefore s_{\Delta T}(t) = \frac{1}{\Delta T}\sum_{n=-\infty}^{\infty} e^{j2\pi nt/\Delta T}$$

$$\therefore \mathbf{F}(s_{\Delta T}(t))(\mu) = \frac{1}{\Delta T}\sum_{n=-\infty}^{\infty}\mathbf{F}(e^{j2\pi nt/\Delta T}) = \frac{1}{\Delta T}\sum_{n=-\infty}^{\infty}\delta(\mu - n/\Delta T)$$

using Fourier shift theorem for the last equality

$$s_{\Delta T}(t) = \sum_{n=-\infty}^{\infty} c_n e^{j2\pi nt/\Delta T};$$

$$c_n = \frac{1}{\Delta T}\int_{-\Delta T/2}^{\Delta T/2} s_{\Delta T}(t)e^{-j2\pi nt/\Delta T}dt = \frac{1}{\Delta T}\int_{-\Delta T/2}^{\Delta T/2}\delta(t)e^{-j2\pi nt/\Delta T}dt$$

(the integral between the limits uses only a single impulse, the one at 0)

$$= \frac{1}{\Delta T}e^{0}\,(\text{using sifting property}) = \frac{1}{\Delta T}$$

Thus the Fourier transform of an impulse train with period ΔT is also an impulse train with period $1/\Delta T$. (Compare: The Fourier transform of a Gaussian is also a Gaussian with an inverted standard deviation). There are a few other functions that are their own Fourier transform.

Appendix: Sampling Theorem

- This is a very important theorem in signal processing.
- Different parts of it were independently discovered by Shannon, Nyquist, Whittaker and Kotelnikov.
- It is often called Shannon's sampling theorem or the Shannon-Nyquist sampling theorem, but it may well be called the Shannon-Nyquist-Whittaker-Kotelnikov sampling theorem.

Appendix: Sampling Theorem

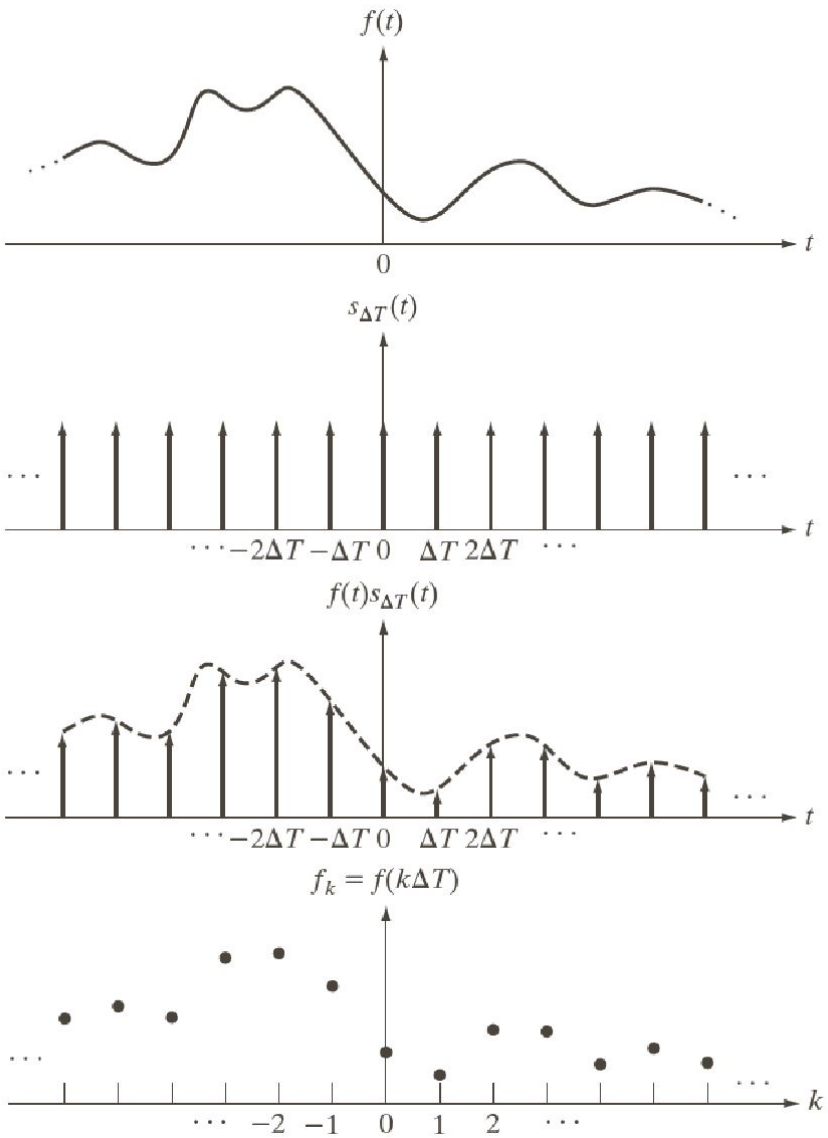
- The theorem states that any band-limited signal with maximum frequency B can be accurately reconstructed from its equi-spaced time-domain samples provided the sampling rate is more than 2B (equivalently the sampling period is less than 1/2B).

- The reconstruction of the original (analog) signal from its time-domain samples proceeds by sinc interpolation – we will see it later.

- An intuitive proof of the sampling theorem can be seen from the next four slides (pictorially).

Appendix: Sampling Theorem

f(t) is an underlying analog signal which is bandlimited, and whose equi-spaced time-domain samples we observe (second figure). The aim is to recover f(t) accurately from its sampled version. The gap between the samples is ΔT.

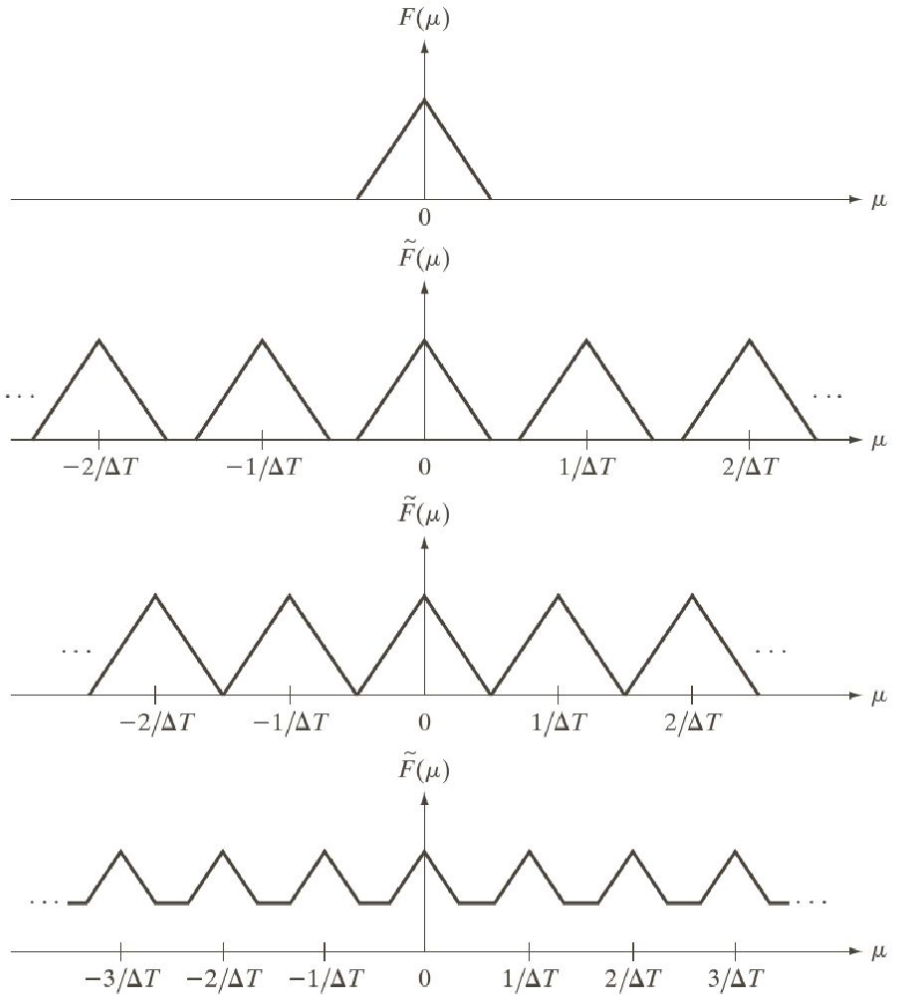


a
b
c
d

FIGURE 4.5
(a) A continuous function. (b) Train of impulses used to model the sampling process. (c) Sampled function formed as the product of (a) and (b). (d) Sample values obtained by integration and using the sifting property of the impulse. (The dashed line in (c) is shown for reference. It is not part of the data.)

Appendix: Sampling Theorem

F(μ) is the Fourier transform of f(t) (observe the bandlimitedness). In the second row, we see the Fourier transform of its sampled version – we have earlier proved that it would be equal to the summation of infinitely many copies of F(μ). If ΔT is too large, then the different periods will merge and cause aliasing. Hence ΔT has to be "small enough".

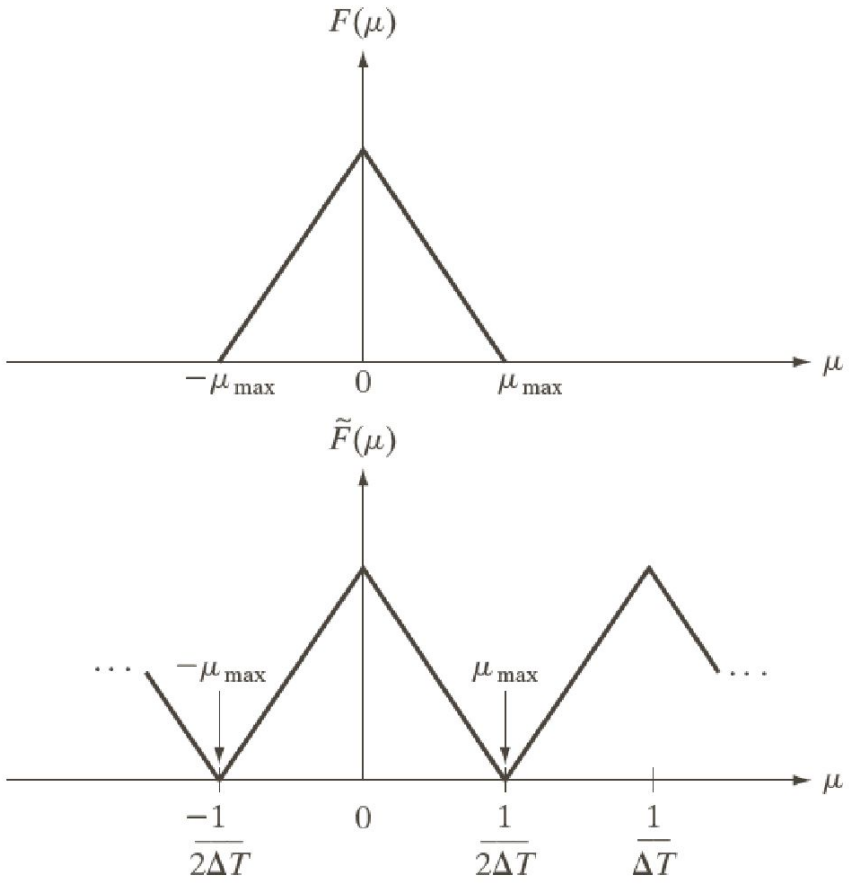


a
b
c
d

FIGURE 4.6 (a) Fourier transform of a band-limited function. (b)–(d) Transforms of the corresponding sampled function under the conditions of over-sampling, critically-sampling, and under-sampling, respectively.

Appendix: Sampling Theorem

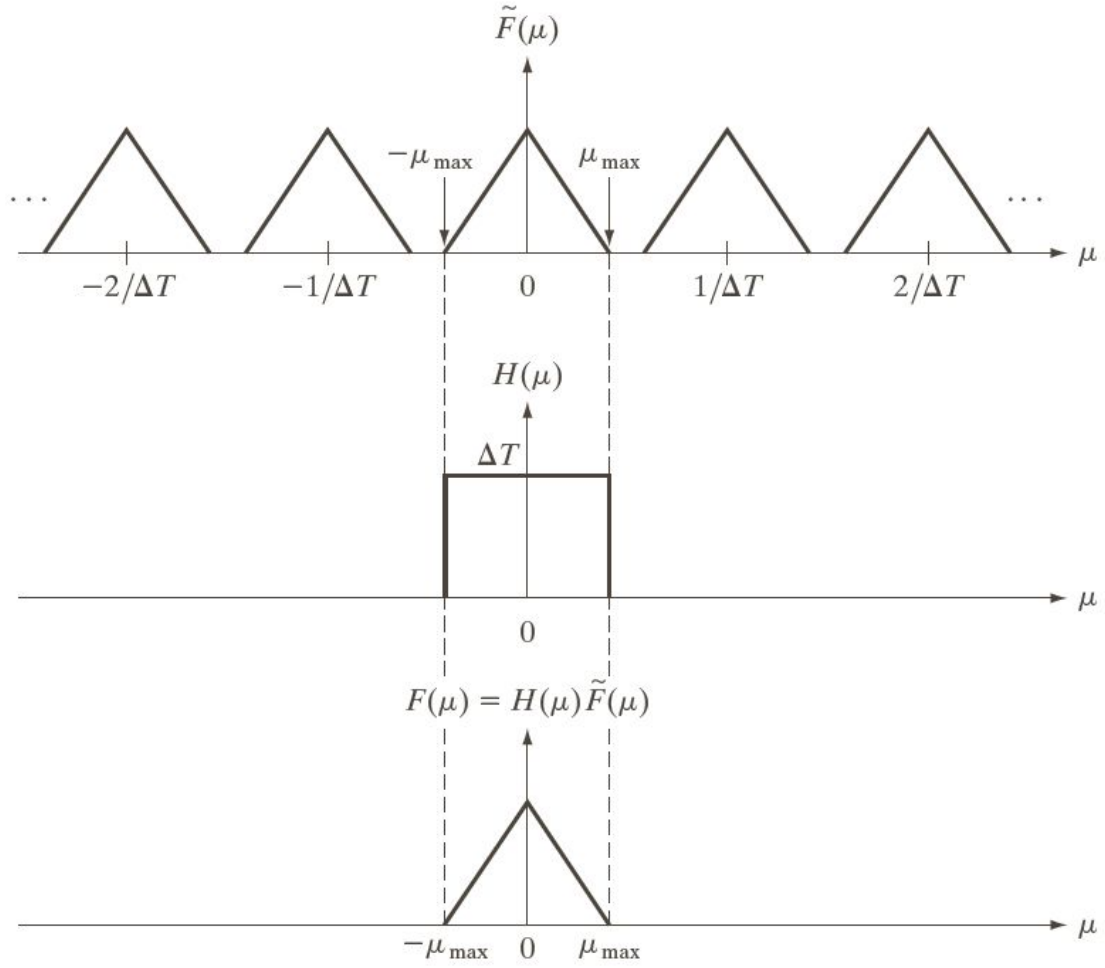
F(μ) is the Fourier transform of f(t) (observe the bandlimitedness). In the second row, we see the Fourier transform of its sampled version – we have earlier proved that it would be equal to the summation of infinitely many copies of F(μ). If ΔT is too large, then the different periods will merge and cause aliasing. Hence ΔT has to be "small enough". As seen in sub-figure b here, we must have $1/\Delta T > 2\mu_{max}$.



a
b

FIGURE 4.7
(a) Transform of a band-limited function.
(b) Transform resulting from critically sampling the same function.

Appendix: Sampling Theorem



a
b
c

FIGURE 4.8 Extracting one period of the transform of a band-limited function using an ideal lowpass filter.

Appendix: Sampling Theorem

- Refer to the previous slide. We have:

$$F(\mu) = H(\mu)\widetilde{F}(\mu) \text{ where } H(\mu) \text{ is a rectangular pulse in the frequency domain}$$

$$f(t) = (h * \widetilde{f})(t) \text{ where } \widetilde{f}(t) = f(t)s_{\Delta T}(t)$$

$$\therefore f(t) = \sum_{n=-\infty}^{\infty} f(n\Delta T)\text{sinc}[(t - n\Delta T)/(n\Delta T)]$$

Sampled version of f

The inverse fourier transform of a rect is a sinc

- This is called the Whittaker-Shannon interpolation formula. In practice, it requires infinite summation, and hence methods like bilinear or bicubic interpolation are preferred. It can be implemented in truncated form, with some error

Appendix: Sampling theorem in 2D

- A 2D signal is band-limited if its Fourier transform is 0 outside a rectangle established by intervals $[-\mu_{max},+\mu_{max}]$ and $[-v_{max},+v_{max}]$.

- The theorem states that such a continuous band-limited 2D signal can be accurately reconstructed from its equi-spaced spatial-domain samples provided the sampling rate is more than $2\mu_{max}$ in the X-direction and more than $2v_{max}$ in the Y-direction (equivalently the sampling period is less than $1/2\mu_{max}$ and $1/2\ v_{max}$).

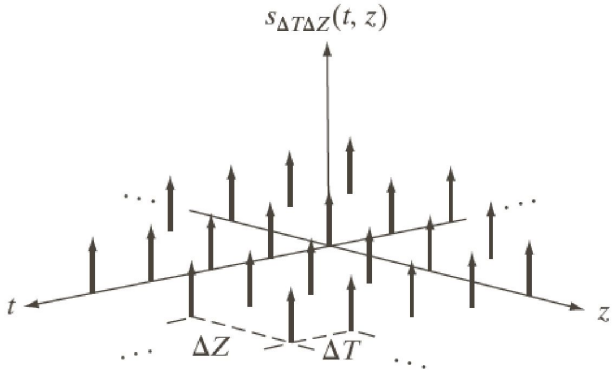
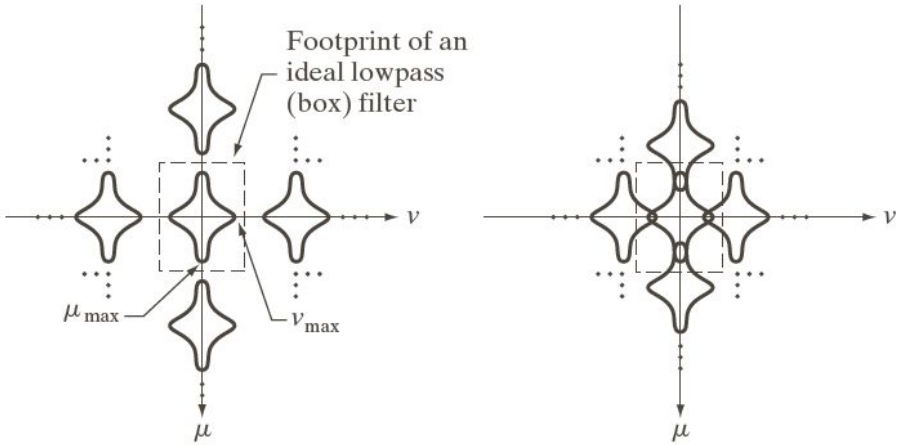


FIGURE 4.14
Two-dimensional impulse train.



a b

FIGURE 4.15
Two-dimensional Fourier transforms of (a) an over-sampled, and (b) under-sampled band-limited function.

Appendix: Sampling Theorem and Aliasing

- If a signal is sampled at a rate less than the sampling rate, its analog version cannot be accurately reconstructed.
- We get an erroneous version of that signal from its samples.
- This phenomenon is called aliasing, because the (genuinely) higher frequency components “masquerade as” (i.e. pretend to be) lower frequency components.
- Aliasing is illustrated in figure 4.6 of the book (also on slide 96 here).
- A certain amount of aliasing is inevitable in practical signal or image acquisition, as in practice the signals can only be of finite length. And we know that time-limited or space-limited signals cannot be bandlimited.

Appendix: Sampling Theorem and Aliasing in 1D

Analog signal (to be reconstructed from its samples) shown in solid line:
$f(t) = \sin(\pi t)$,
Time period = 2 second

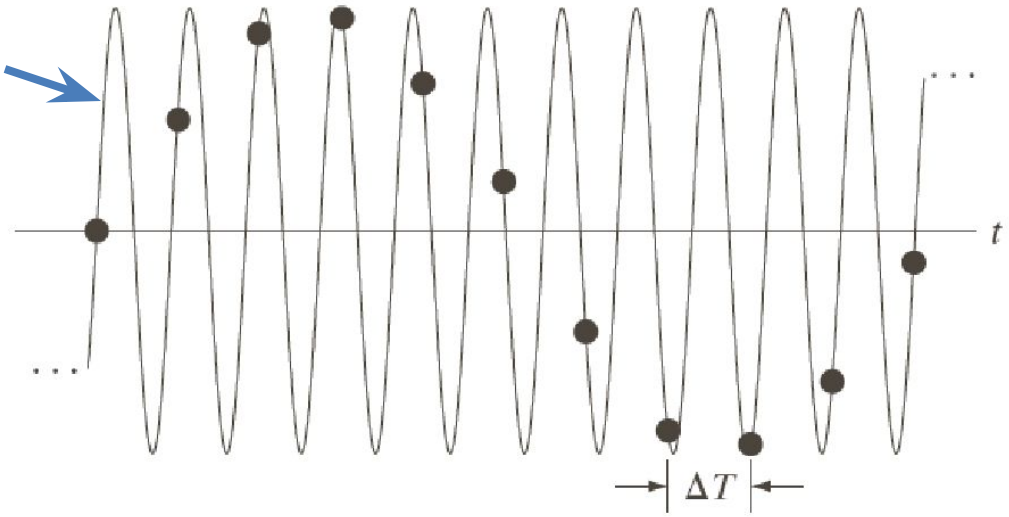
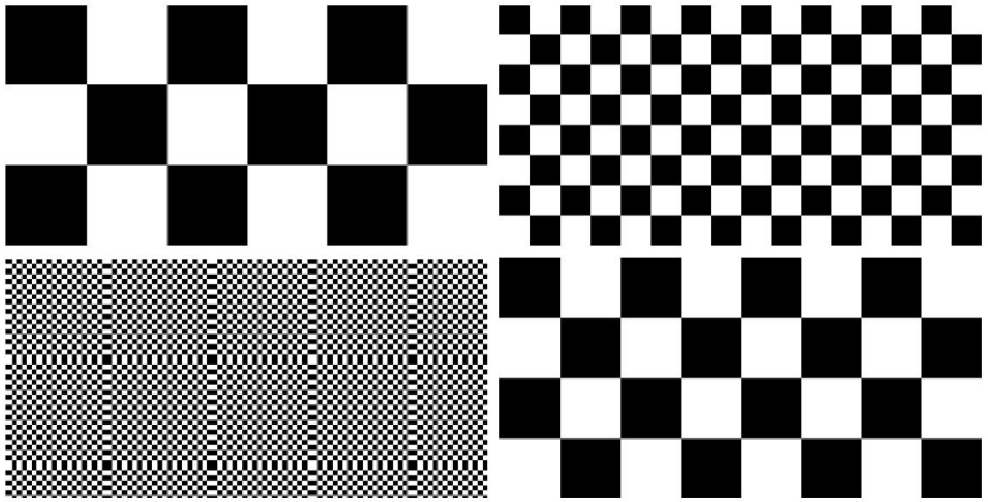


FIGURE 4.10 Illustration of aliasing. The under-sampled function (black dots) looks like a sine wave having a frequency much lower than the frequency of the continuous signal. The period of the sine wave is 2 s, so the zero crossings of the horizontal axis occur every second. ΔT is the separation between samples.

As per the sampling theorem, we need to sample this signal at a rate strictly **greater** than twice the largest frequency, i.e. greater than 2 x ½ = 1 sample per second. Equivalently the sampling rate must be strictly **less** than 1/(2 x ½) = 1 second. If you sample at exactly twice the largest frequency, i.e. 1 sample per second, you will collect values of $\sin(\pi t)$ for integer t, i.e. all zeroes. This will not allow for good reconstruction.

Appendix: Sampling Theorem and Aliasing in 2D (example 1)



a b
c d

FIGURE 4.16 Aliasing in images. In (a) and (b), the lengths of the sides of the squares are 16 and 6 pixels, respectively, and aliasing is visually negligible. In (c) and (d), the sides of the squares are 0.9174 and 0.4798 pixels, respectively, and the results show significant aliasing. Note that (d) masquerades as a "normal" image.

Consider a perfect imaging system that can faithfully capture images of size 96 x 96 pixels (one pixel is the resolution or sampling period of the system in both X and Y directions). Such a system can faithfully capture a checkerboard pattern in which each square has size 1 pixel or more. See top row where the square width is 16 and 6 pixels respectively. But there is a problem if the size of the squares is less than 1 pixel: see bottom row where the squares have width 0.9174 and 0.4798 respectively. These produce misleading patterns which can masquerade as normal images.

Appendix: Sampling Theorem and Aliasing in 2D (example 2)

"original" shirt texture

Aliased shirt texture (see appearance of slanting patterns from top right to bottom keft near the midde shirt button)

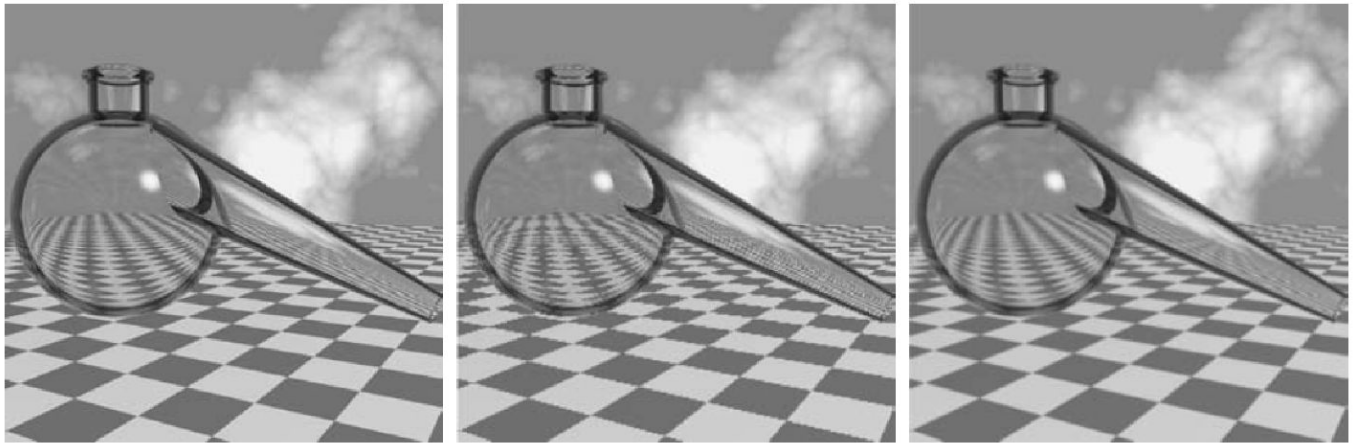
Appendix: Sampling Theorem and Aliasing in 2D (example 3)

a b c

FIGURE 4.17 Illustration of aliasing on resampled images. (a) A digital image with negligible visual aliasing. (b) Result of resizing the image to 50% of its original size by pixel deletion. Aliasing is clearly visible. (c) Result of blurring the image in (a) with a 3×3 averaging filter prior to resizing. The image is slightly more blurred than (b), but aliasing is not longer objectionable. (Original image courtesy of the Signal Compression Laboratory, University of California, Santa Barbara.)

Aliasing can be controlled by a small amount of blurring – this is of course not without loss of information, but at least it does not produce fake textures.

Appendix: Sampling Theorem and Aliasing in 2D (example 4)



a b c

FIGURE 4.18 Illustration of jaggies. (a) A 1024 × 1024 digital image of a computer-generated scene with negligible visible aliasing. (b) Result of reducing (a) to 25% of its original size using bilinear interpolation. (c) Result of blurring the image in (a) with a 5 × 5 averaging filter prior to resizing it to 25% using bilinear interpolation. (Original image courtesy of D. P. Mitchell, Mental Landscape, LLC.)

Jagged edges due to aliasing