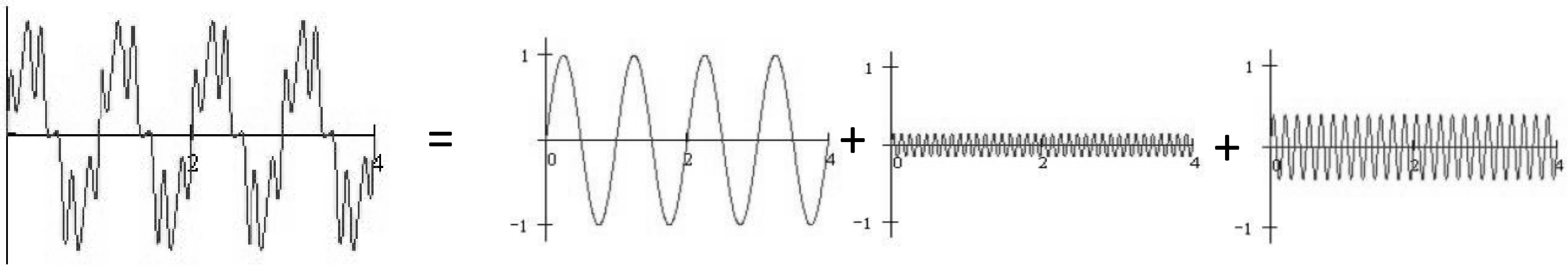
Introduction to Digital Image Processing

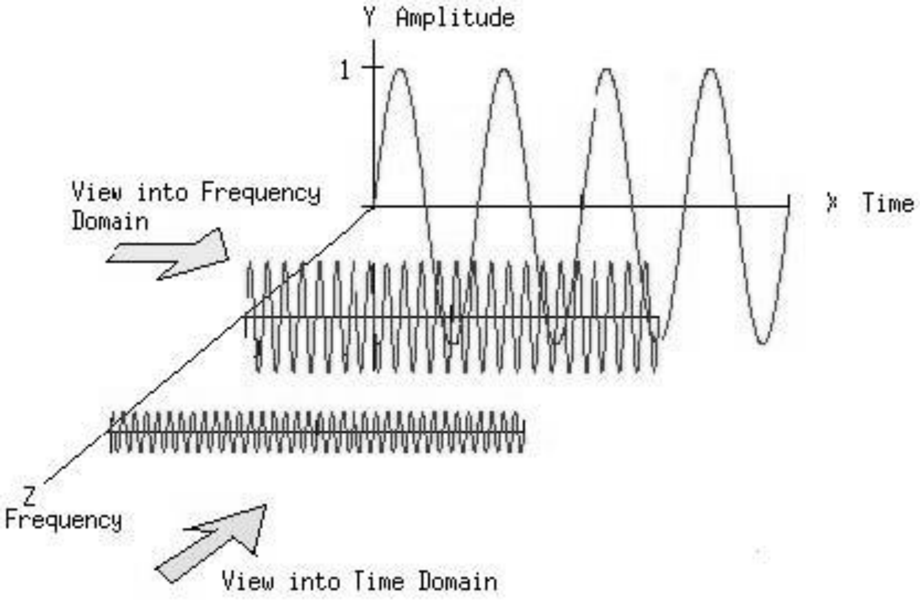
— Fourier Transform

Xiaohui Yuan

Department of Computer Science and Engineering
University of North Texas
xiaohui.yuan@unt.edu

Signals in Time and Frequency Domains





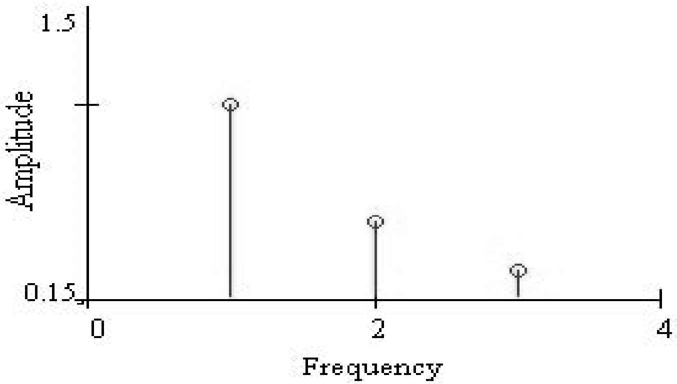
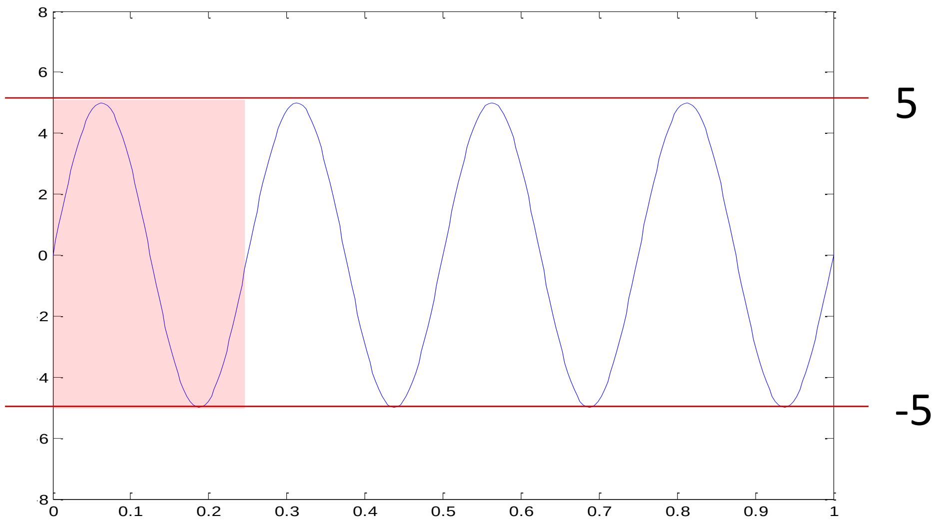


Photo by Digital Image Processing (3rd Edition), by Rafael C. Gonzalez and Richard E. Woods, Pearson, ISBN-13: 978-0131687288

Continuous and Digitized Sine Wave

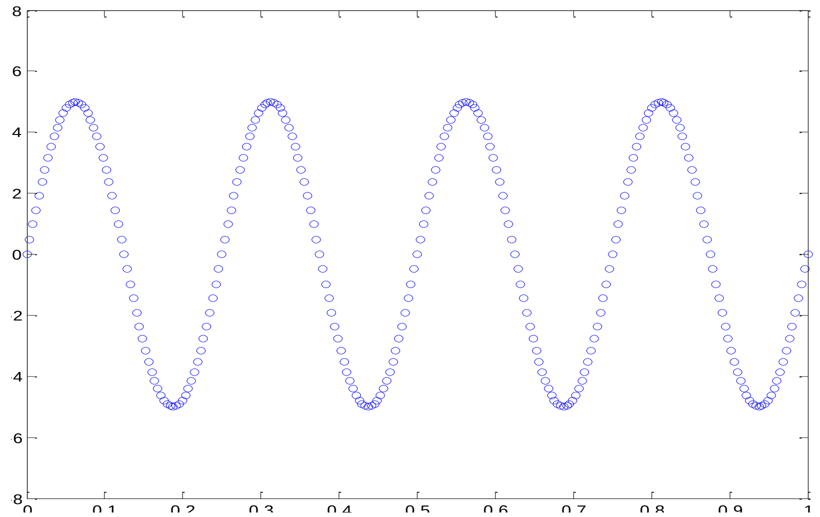


$5\sin(4t)$

The amplitude of this signal is 5

The frequency is at 4 Hz

The unit for frequency is cycles/second, a.k.a. Hertz (Hz)

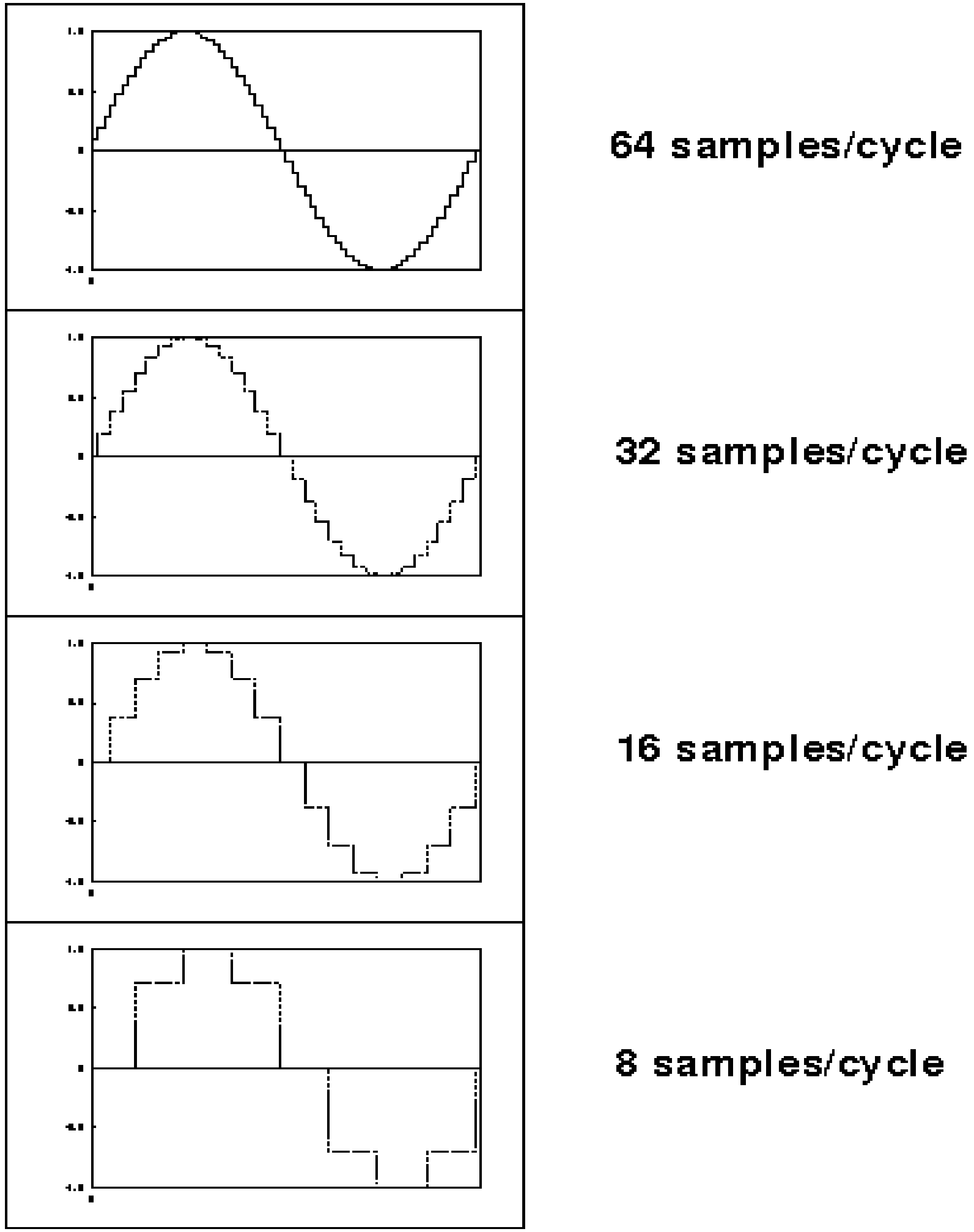


Sampling rate = 256 samples/second

Sampling duration = 1 second

Sampling Rate

- The sampling rate is the rate at which signal amplitude is digitized from the original waveform.
- A high sampling rate allows the waveform to be more accurately represented, yet it requires more storage space.
- *What is the lowest possible sampling rate or how many samples per second are needed to fully recover the original signal?*



Nyquist Theorem

- We can digitally represent the frequencies of a signal up to half the sampling rate, i.e., Nyquist frequency.
    - If a signal is sampled at a rate greater than twice its highest frequency, it is possible to recover the original signal.
- Frequencies above the Nyquist frequency "fold over" that appear to be lower frequencies.
    - This corruption is in the form of additional frequency components being introduced into the sampled function, which is called aliasing.

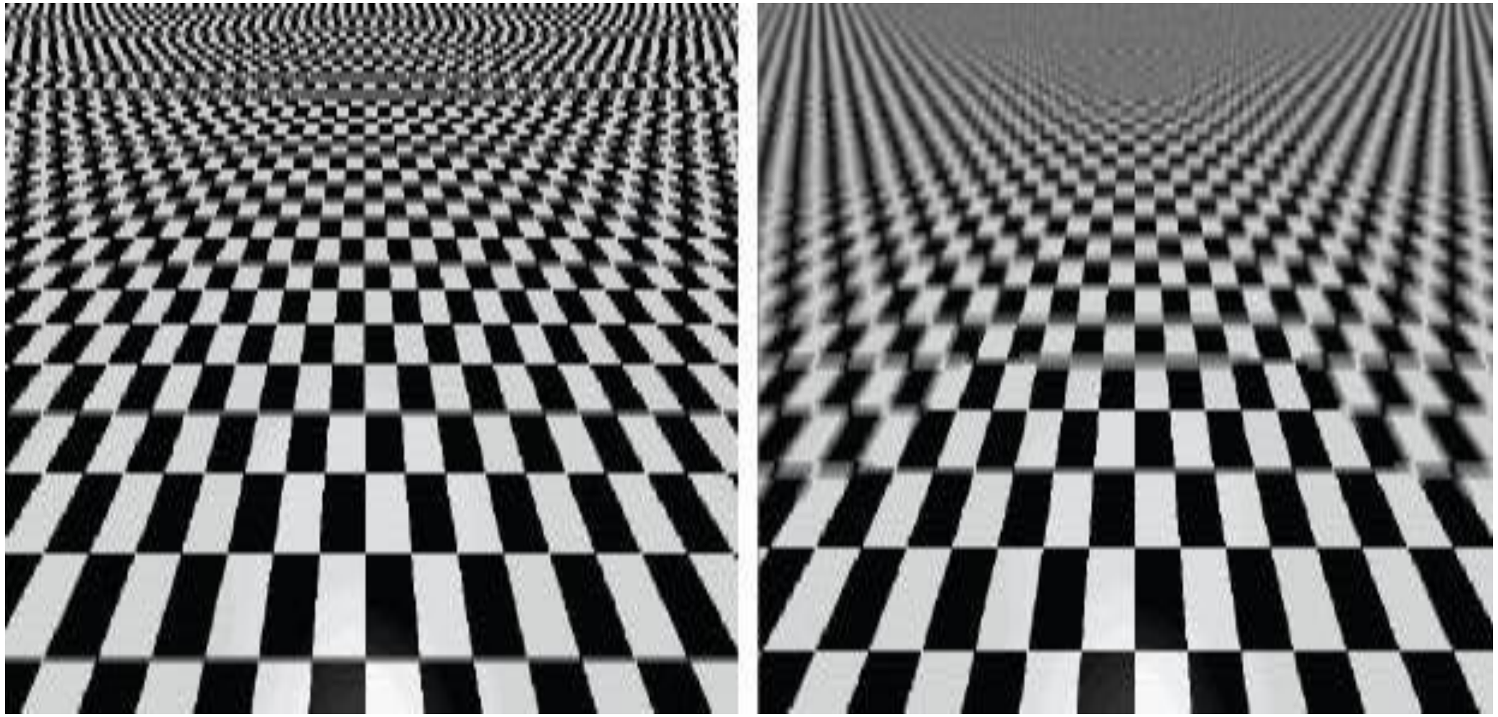


Photo by Digital Image Processing (3rd Edition), by Rafael C. Gonzalez and Richard E. Woods, Pearson, ISBN-13: 978-0131687288

Basis Functions for Signal Representation

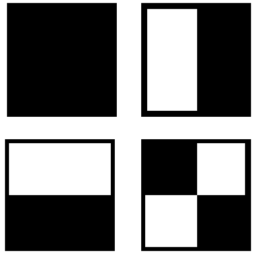
- We can describe a point as linear combination of orthogonal basis vectors:

$$x = a_1 v_1 + a_2 \mathrm{v}_2 + \cdots + \mathrm{a_n v_n}$$

- The standard basis for images is the set of unit matrices with one element being one and the rest being zero, e.g.,

$$\mathrm{I} = \begin{pmatrix} 2 & 1 \\ 6 & 1 \end{pmatrix} = 2\begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix} + 1\begin{pmatrix} 0 & 1 \\ 0 & 0 \end{pmatrix} + 6\begin{pmatrix} 0 & 0 \\ 1 & 0 \end{pmatrix} + 1\begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}$$

- Hadamard Basis Function
  - We can express the image with these new (normalized) basis vectors as:



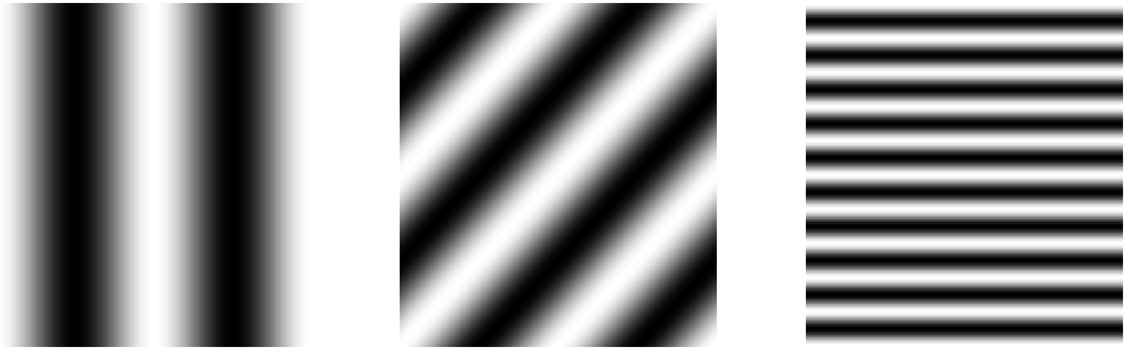
$$\mathrm{I} = \begin{pmatrix} 2 & 1 \\ 6 & 1 \end{pmatrix} = \frac{1}{2}\left[5\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} - 3\begin{pmatrix} -1 & 1 \\ -1 & 1 \end{pmatrix} + 2\begin{pmatrix} -1 & -1 \\ 1 & 1 \end{pmatrix} - 2\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}\right]$$

  - Coefficients of sum is the projection of image I onto the new basis
  - These are the coordinates of the image in the Hadamard space

$$\mathrm{I}_H = \mathcal{H}(\mathrm{I}) = \begin{pmatrix} 5 & -3 \\ 2 & -2 \end{pmatrix}$$

Sinusoidal Basis Function

- Binary-valued, rectangular wave pattern of Hadamard basis does not capture real image gradients well
- It can only be used for signals with limited range/size
- Idea: Use smoothly varying sinusoidal patterns at different densities and angles as the basis images



- Any periodic functions and signals can be expanded into a series of sine and cosine functions
    - The transformation takes one signal (or function) and turns it into another signal (or function) by "matching" with the basis signals at different frequencies.

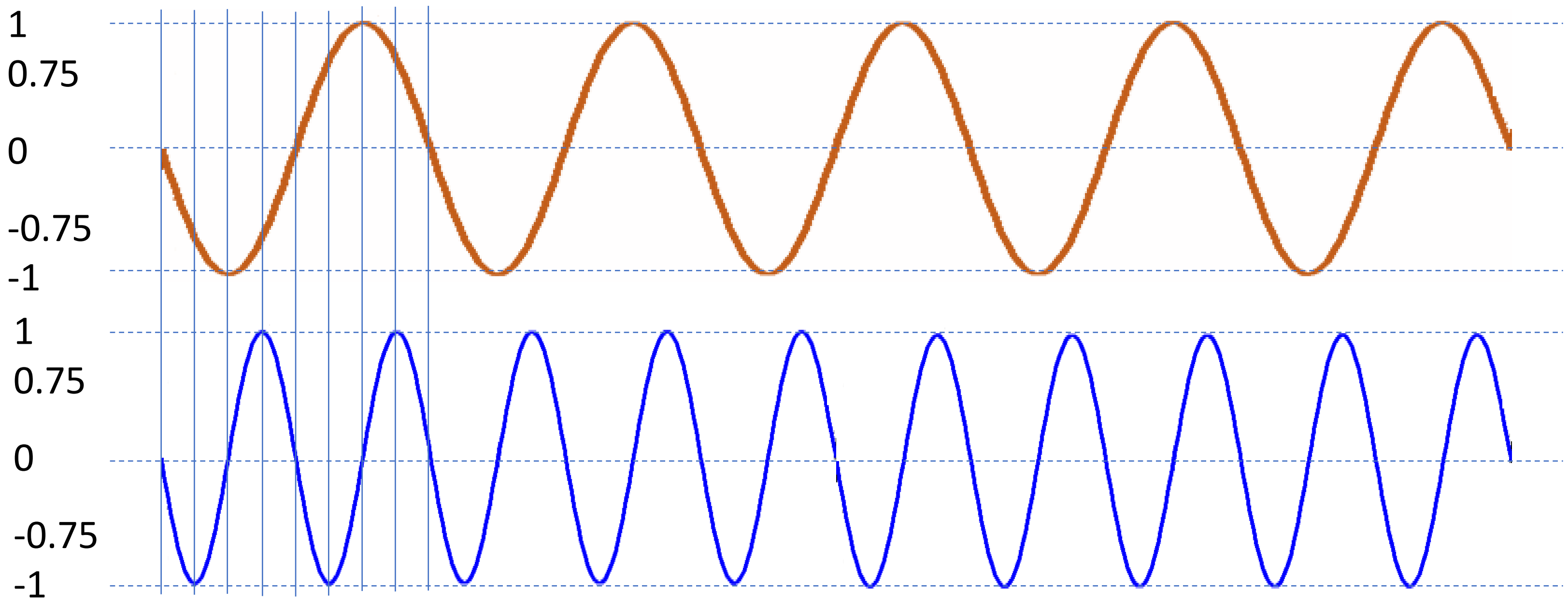
Fourier Transform

Forward Fourier transform: $F(w) = \int_{-\infty}^{\infty} f(t)e^{-iwt}\,dt$

Inverse Fourier transform: $f(t) = \frac{1}{2\pi}\int_{-\infty}^{\infty} F(w)e^{iwt}\,dw$

Euler's formula: $e^{i} = \cos x + i\sin x$

- The decomposition base functions serve as filters.
  - Only the signal component that has the matched frequency results in a non-zero response.
  - This is valid only in the infinite time (or frequency) domain and does not depend on the phase of a signal.



The Discrete Fourier Transform

- The Discrete Fourier Transform:

Discrete Fourier transform
$$F_n = \sum_{k=0}^{N-1} f_k e^{-\frac{2\pi i k n}{N}}$$

Inverse Discrete Fourier transform
$$f_k = \frac{1}{N} \sum_{n=0}^{N-1} F_n e^{\frac{2\pi i k n}{N}}$$

- Euler's formula: $e^{i\theta} = \cos\theta + i\sin\theta$

$$F(n) = \sum_{k=0}^{N-1} f(x)[\cos\left(\frac{2\pi kn}{N}\right) - isin\left(\frac{2\pi kn}{N}\right)]$$

$$f(k) = \frac{1}{N}\sum_{n=0}^{N-1} F(n)\,[\cos\left(\frac{2\pi kn}{N}\right) + isin\left(\frac{2\pi kn}{N}\right)]$$

Two Dimensional (2D) Discrete Fourier Transform

- The 2D Fourier basis uses the same family of complex sinusoidal functions and can be computed as follows:

  Forward 2D DFT

$$F(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) e^{-i2\pi(ux/M + vy/N)}$$

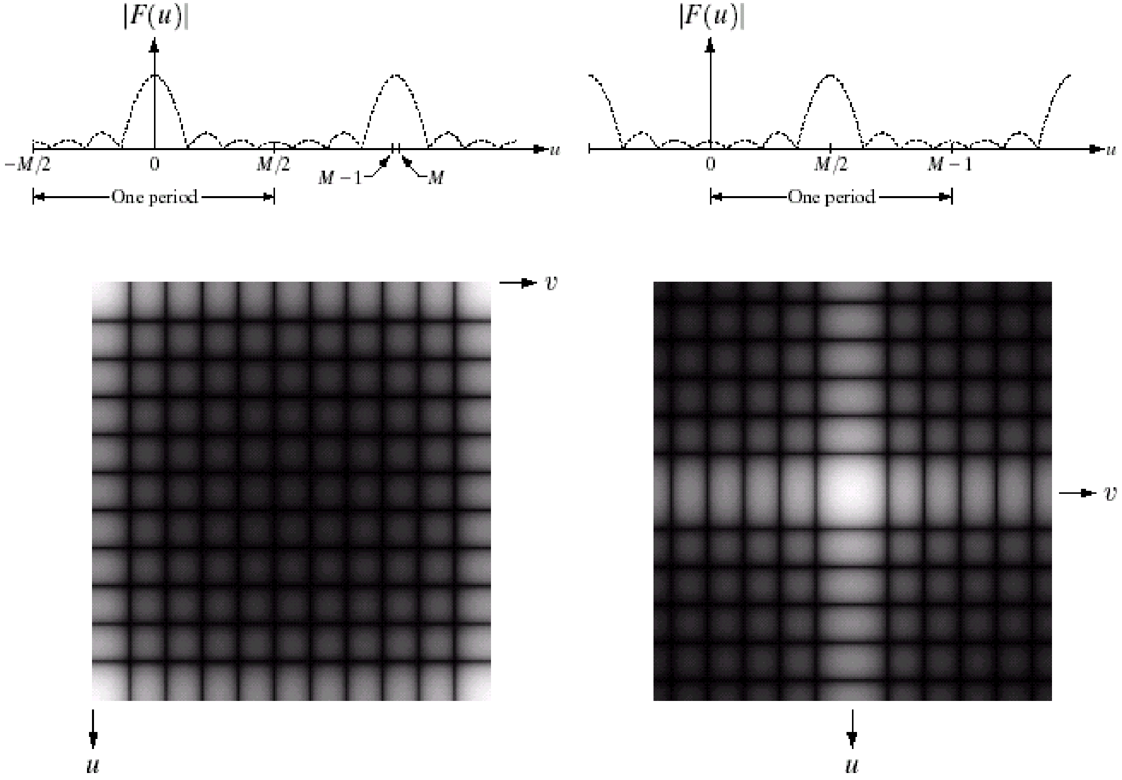
  Inverse 2D DFT

$$f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{i2\pi(ux/M + vy/N)}$$

  (u, v) are the frequency coordinates while (x, y) are the spatial coordinates
  M, N are the number of pixels along the x, y coordinates

Properties of Fourier Transform

Periodicity and conjugate symmetry



Separability

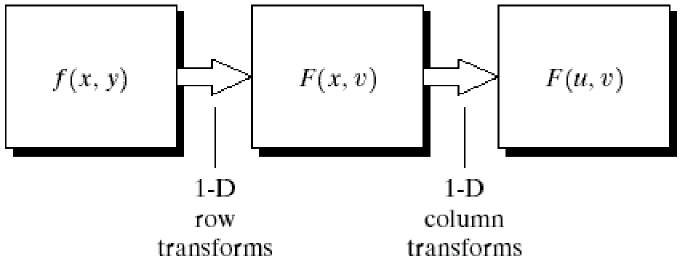
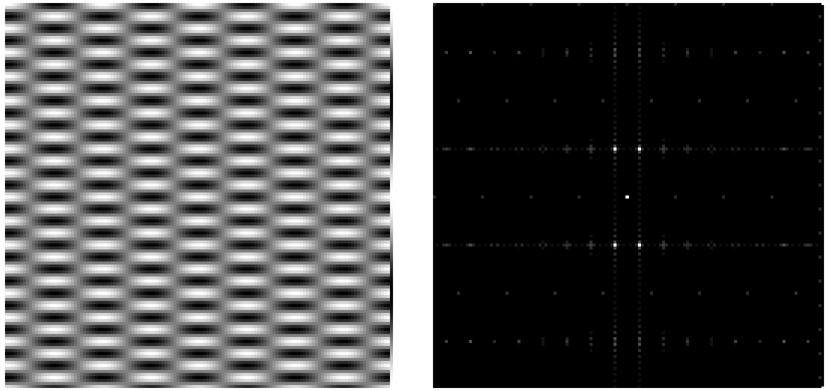


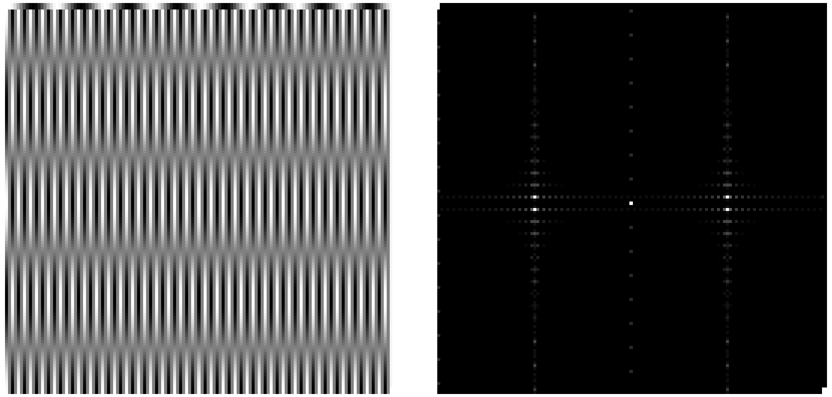
Photo by Digital Image Processing (3rd Edition), by Rafael C. Gonzalez and Richard E. Woods, Pearson, ISBN-13: 978-0131687288

Fourier Transform of Images

- Two images generated from two-dimensional sine waves are shown on the left
- The Fourier transform of the image consists of three peaks
  - Center peak gives the average (i.e., zero Hz) magnitude
  - The other two peaks are located at the center vertical direction and are spaced equally to the center
- The center is the average of all sine waves.
  - So, it is usually the brightest spot and is used as a point of reference



This image exclusively has 32 strips in the vertical direction.



This image exclusively has 8 strips in the horizontal direction.

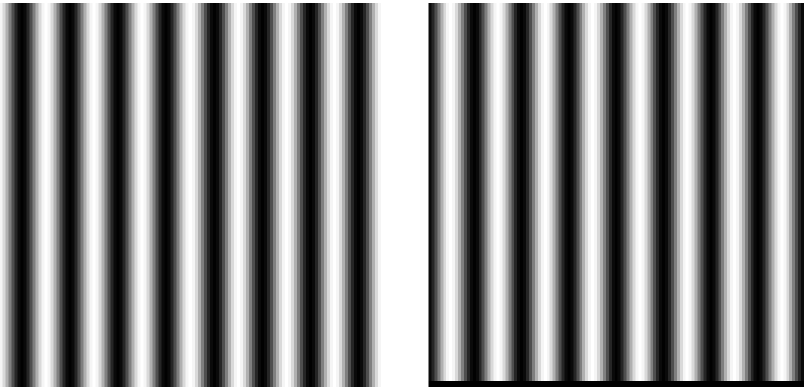
Photo by Digital Image Processing (3rd Edition), by Rafael C. Gonzalez and Richard E. Woods, Pearson, ISBN-13: 978-0131687288

Magnitude and Phase

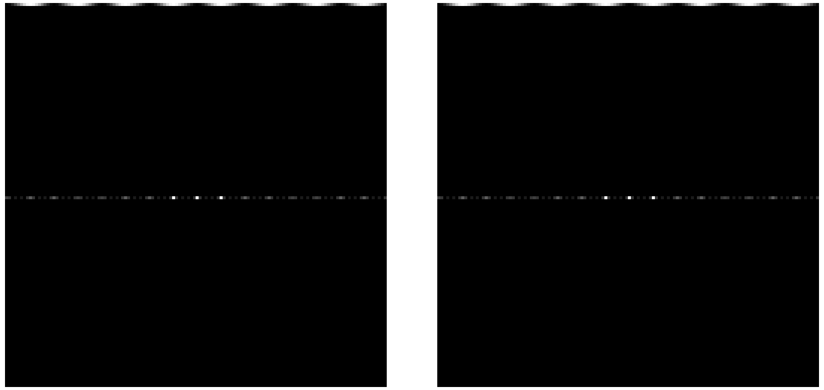
- Instead of representing the complex numbers as real and imaginary parts we can represent them as magnitude and phase, where they are computed as follows:

$$Magnitude = \sqrt{\mathrm{Re}^2 + \mathrm{Im}^2}$$
$$Phase = \arctan\left(\frac{\mathrm{Im}}{\mathrm{Re}}\right)$$

- <u>Magnitude</u> tells **how strong** a certain frequency component is in the image.
- <u>Phase</u> tells ‘**where’** that frequency component appears in the image.



These two images are shifted pi with respect to each other.

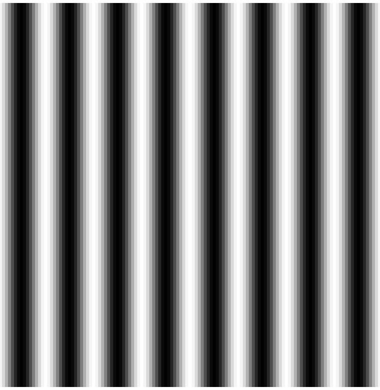


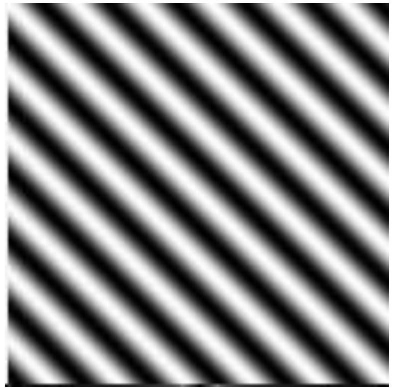
The above illustration of Fourier coefficient consists only the magnitude. No phase components are shown

Photo by Digital Image Processing (3rd Edition), by Rafael C. Gonzalez and Richard E. Woods, Pearson, ISBN-13: 978-0131687288

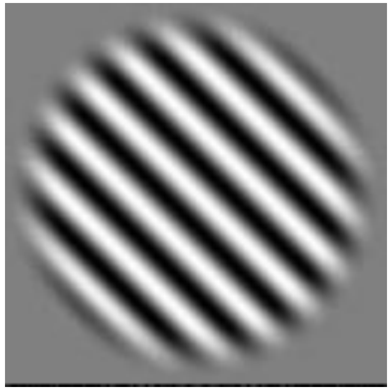
Reconstructions from the Phase coefficients

Rotation Effect

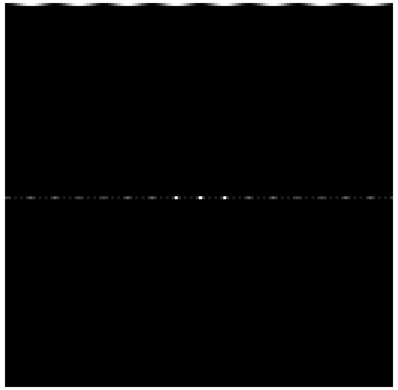


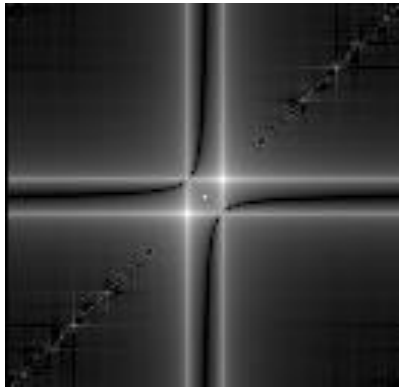


These two images are identical except the right one has been rotated 45 degrees.

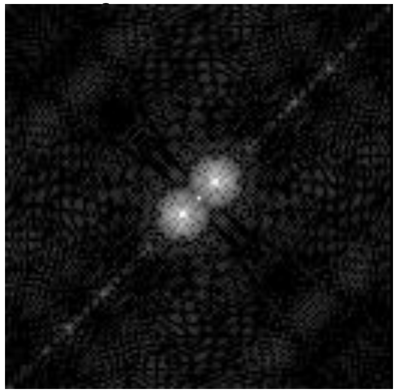


We can blur the image boundary and do Fourier

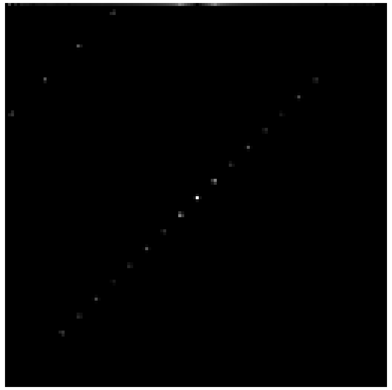




The FT always treats an image as a periodic array of horizontal and vertical sine curves. Since the image abruptly ends at the edges of the box it has a strong effect on the image.



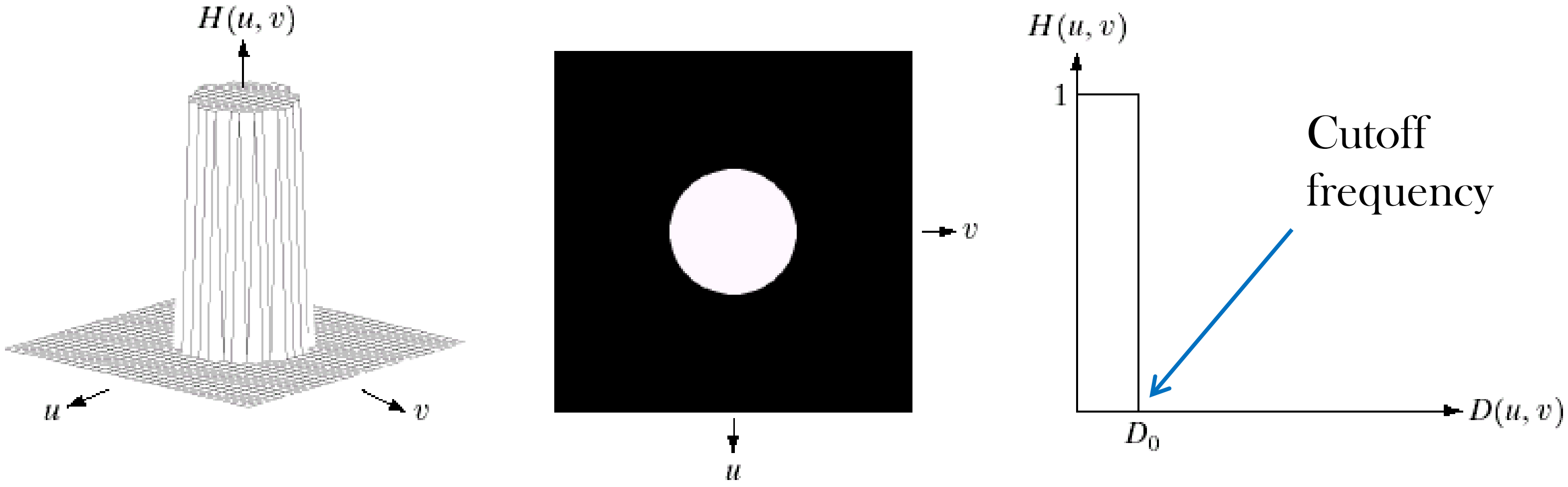
The result is better



The true frequency response

Photo by Digital Image Processing (3rd Edition), by Rafael C. Gonzalez and Richard E. Woods, Pearson, ISBN-13: 978-0131687288

Ideal Lowpass Filter (ILPF)



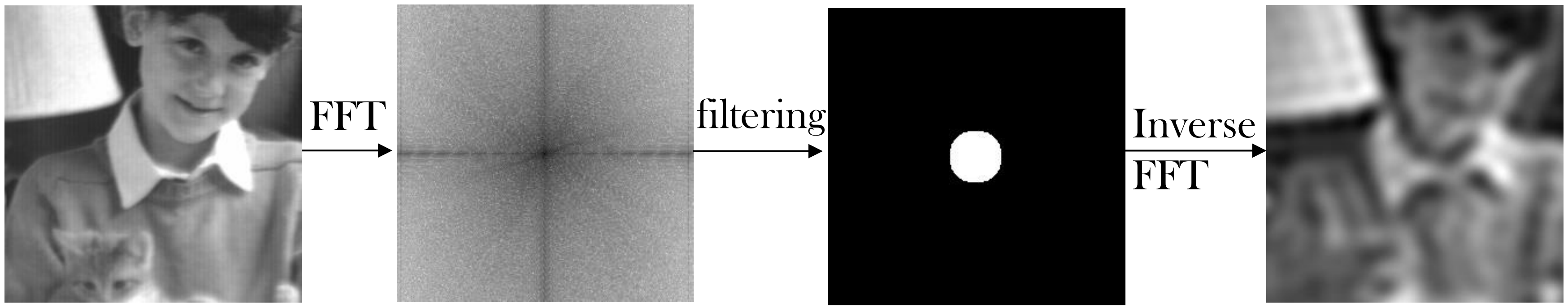


Photo by Digital Image Processing (3rd Edition), by Rafael C. Gonzalez and Richard E. Woods, Pearson, ISBN-13: 978-0131687288

Results of Ideal Lowpass Filters

Original image	Lowpass filter response with cutoff frequency at 5
Lowpass filter response with cutoff frequency at 15	Lowpass filter response with cutoff frequency at 30
Lowpass filter response with cutoff frequency at 80	Lowpass filter response with cutoff frequency at 230

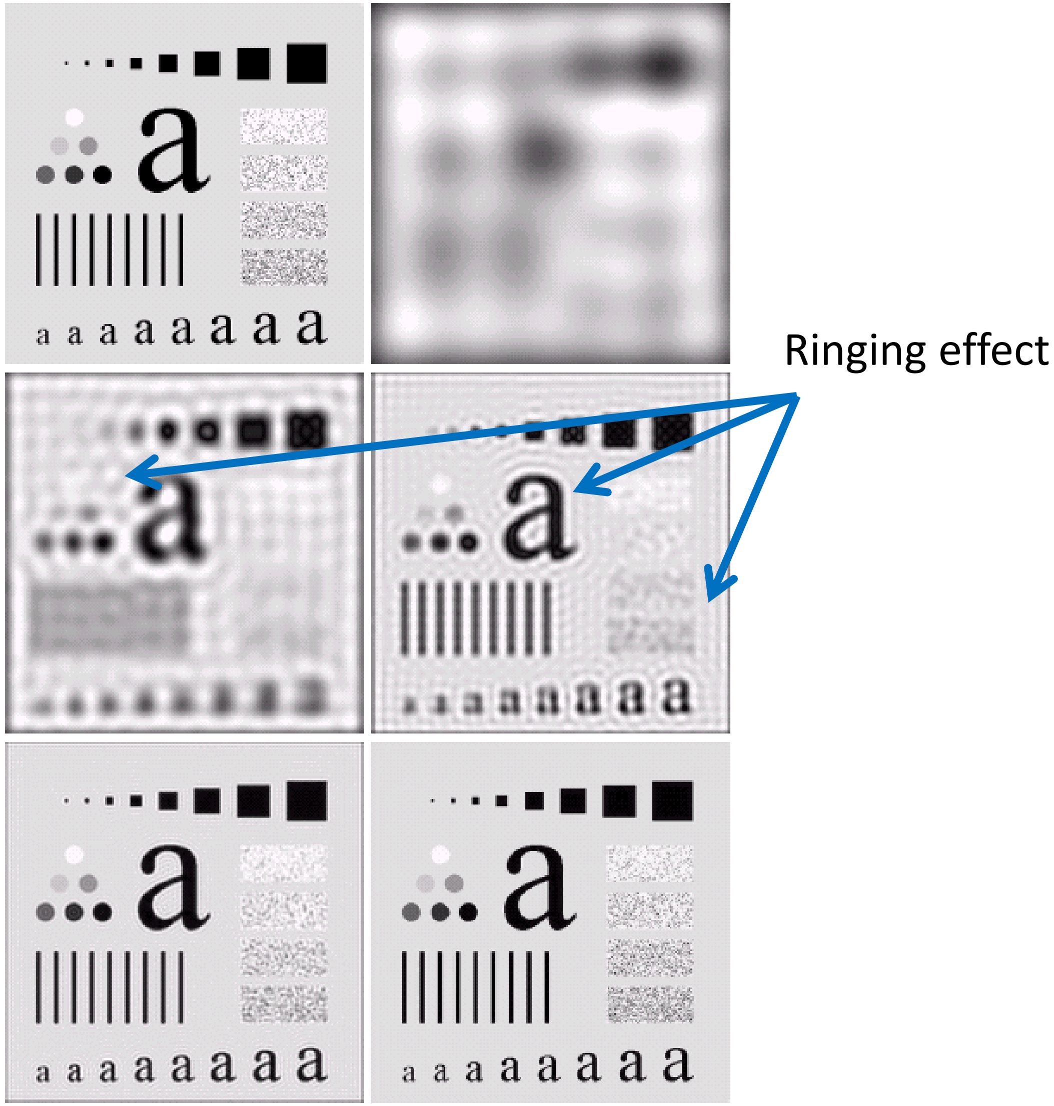
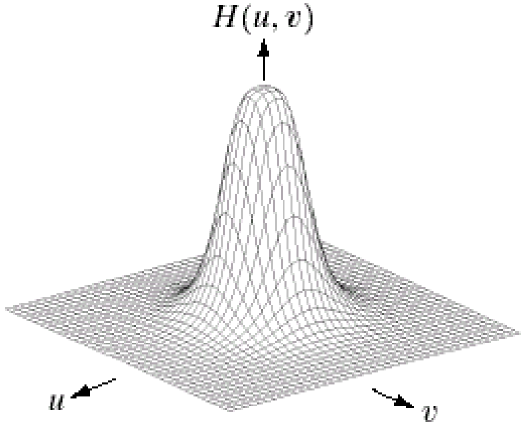


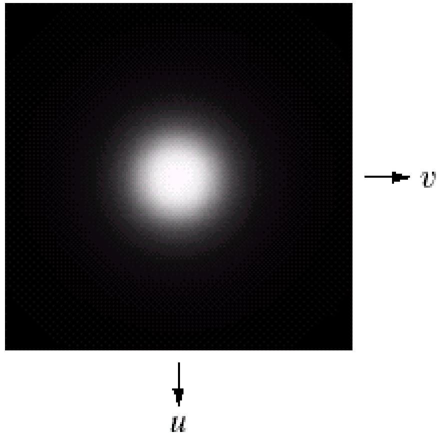
Photo by Digital Image Processing (3rd Edition), by Rafael C. Gonzalez and Richard E. Woods, Pearson, ISBN-13: 978-0131687288

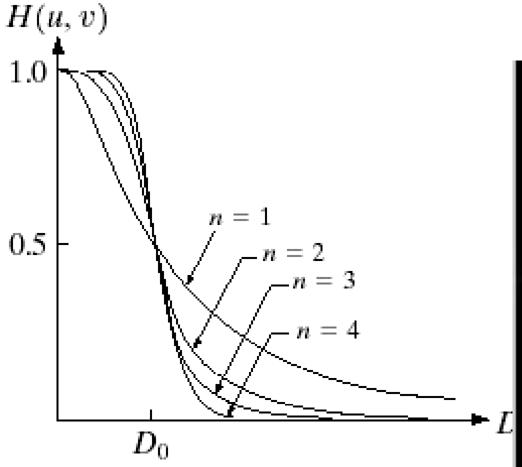
Butterworth Lowpass Filter (BLPF)

- Ideal filtering introduces unwanted artifacts (ringing) into the result.
- Butterworth filter uses smooth transition.

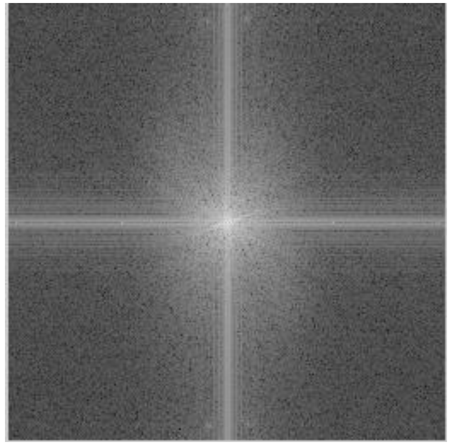
	Butterworth	Ideal
Low pass	$f_{Bl}(x) = \dfrac{1}{1 + (x/T)^{2n}}$	$f_{Il}(x) = \begin{cases} 1 & x < T \\ 0 & x \geq T \end{cases}$



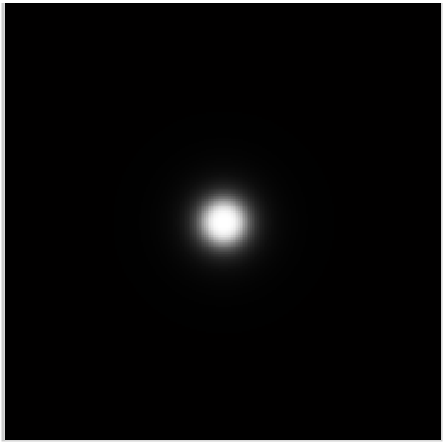




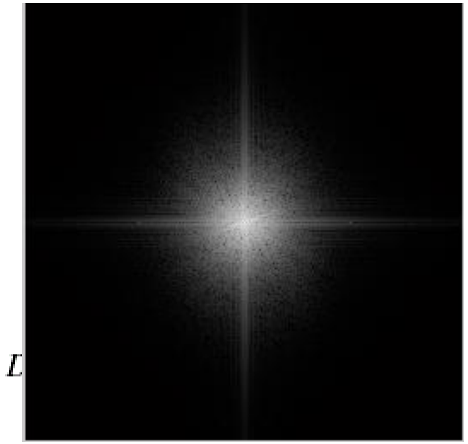
Fourier Coefficients



BLPF Response



Filter Response



Reconstruction

Photo by Digital Image Processing (3rd Edition), by Rafael C. Gonzalez and Richard E. Woods, Pearson, ISBN-13: 978-0131687288

Gaussian Lowpass Filter (GLPF)

- Gaussian lowpass filter follows a 2D Gaussian function as follows:

$$f(x, y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

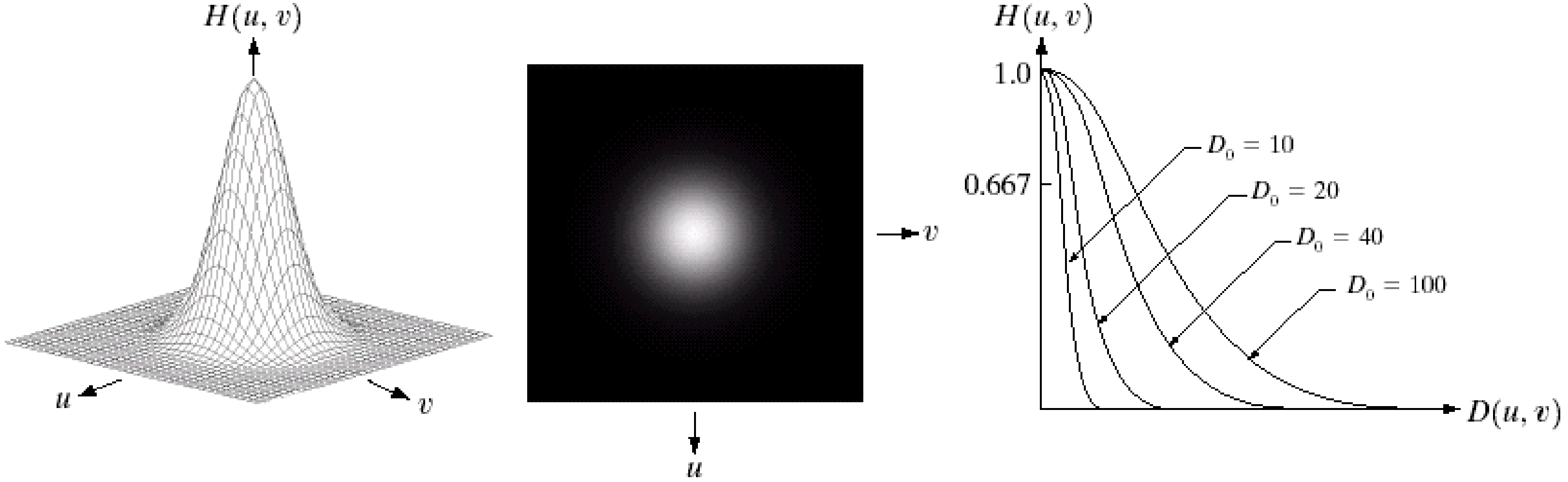
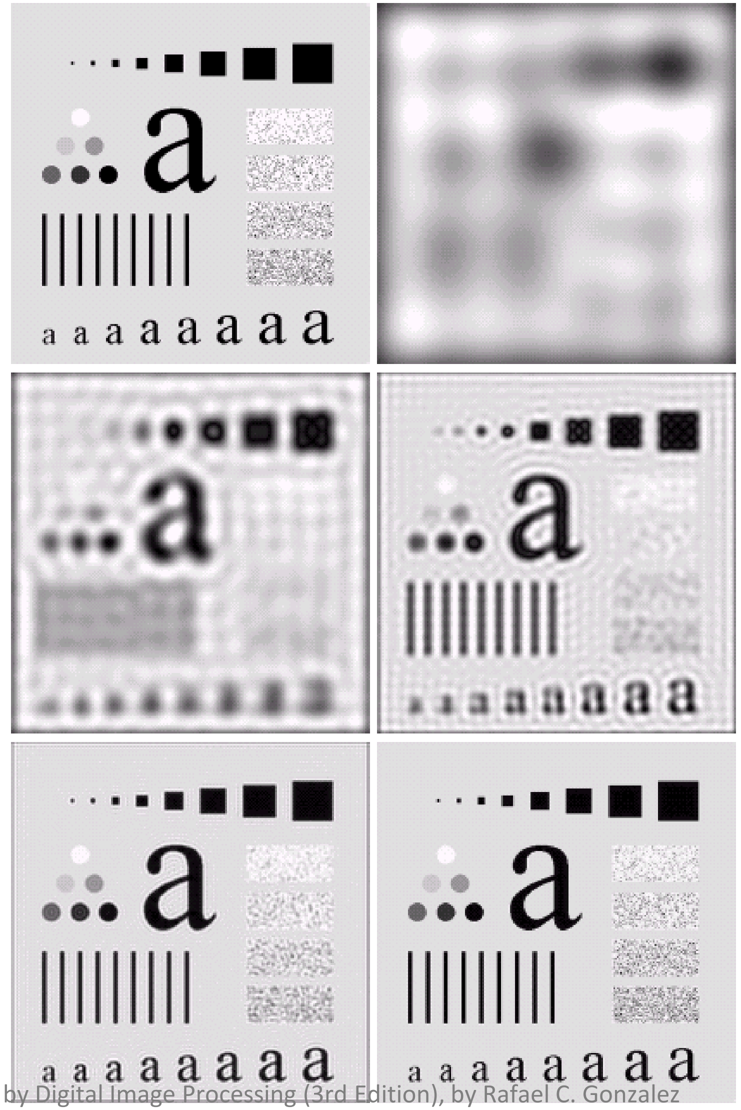
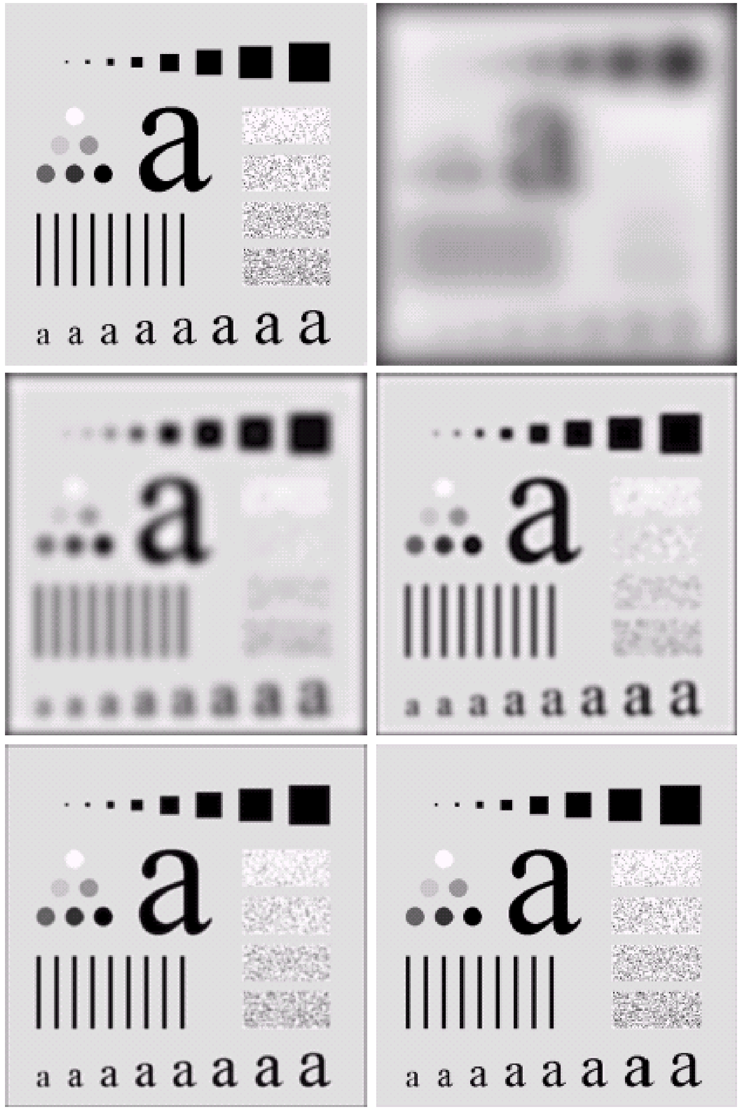


Photo by Digital Image Processing (3rd Edition), by Rafael C. Gonzalez
and Richard E. Woods, Pearson, ISBN-13: 978-0131687288

ILPF



BLPF



GLPF

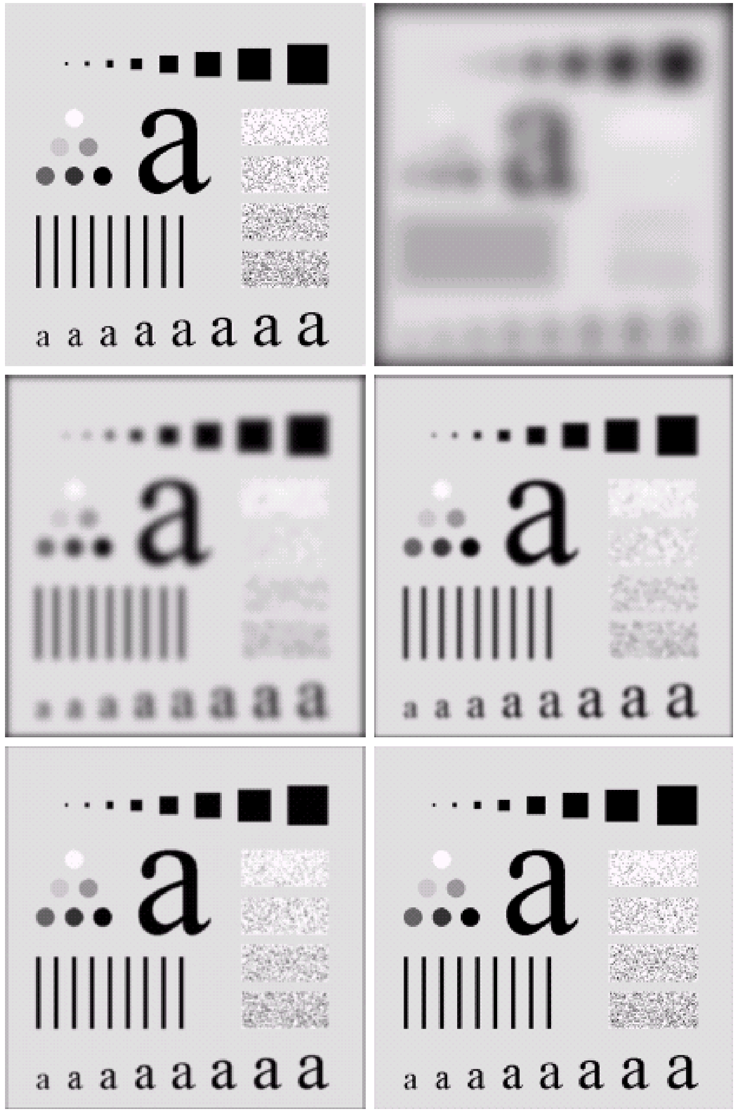
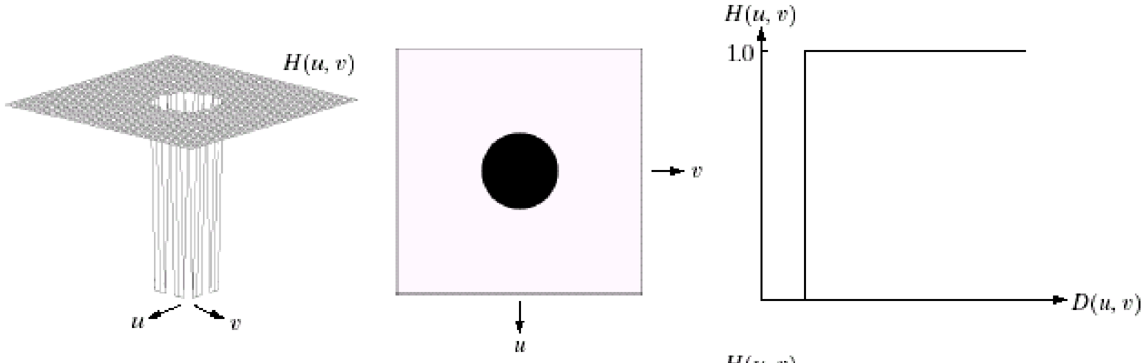


Photo by Digital Image Processing (3rd Edition), by Rafael C. Gonzalez and Richard E. Woods, Pearson, ISBN-13: 978-0131687288

Highpass Filters

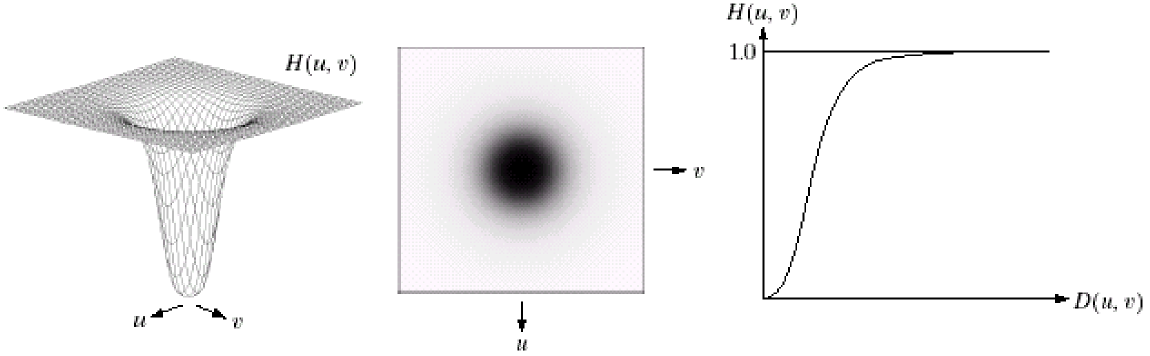
Ideal highpass filter

$$H(u,v) = \begin{cases} 0 & \text{if } \mathrm{D(u,v)} \le \mathrm{D}_0 \\ 1 & \text{if } \mathrm{D(u,v)} > \mathrm{D}_0 \end{cases}$$



Butterworth highpass filter

$$H(u,v) = \frac{1}{1 + [\mathrm{D}_0 / D(u,v)]^{2n}}$$



Gaussian highpass filter

$$H(u,v) = 1 - e^{-D^2(u,v)/2D_0^2}$$

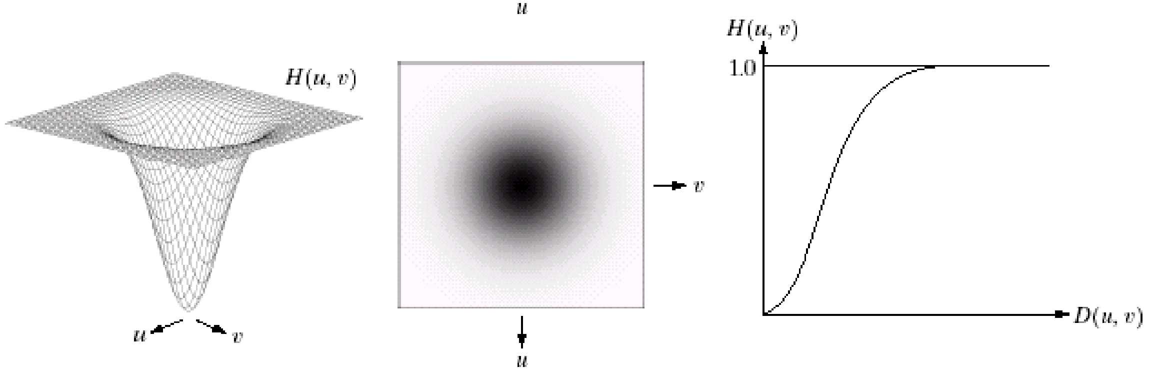


Photo by Digital Image Processing (3rd Edition), by Rafael C. Gonzalez
and Richard E. Woods, Pearson, ISBN-13: 978-0131687288

Highpass Filter Response

The cutoff frequencies for the three kinds of filters are 15, 30, and 80 from left to right, respectively

Results of IHPF

Results of BHPF

Results of GHPF

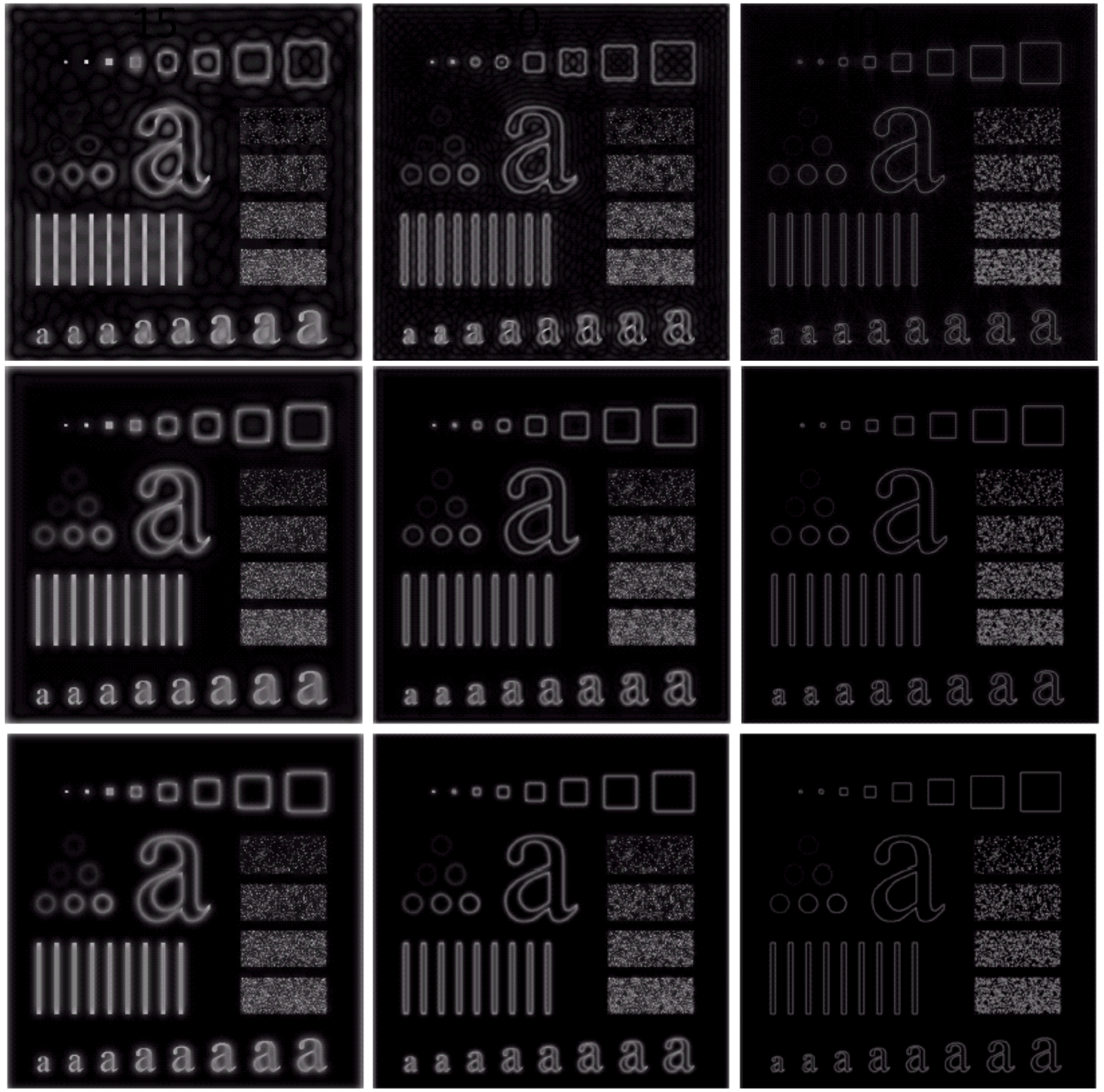
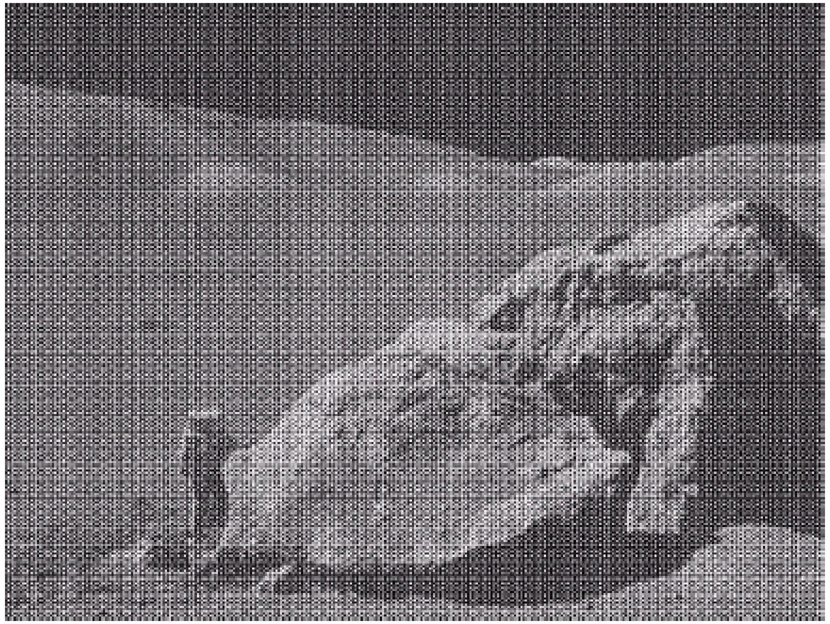


Photo by Digital Image Processing (3rd Edition), by Rafael C. Gonzalez and Richard E. Woods, Pearson, ISBN-13: 978-0131687288

Periodic Noise Removal

- Periodic noise typically arises due to various electrical or electromagnetic interferences
- It appears to be regular patterns in an image
- Frequency techniques (e.g., band reject filter) in the Fourier domain are most effective at removing periodic noise



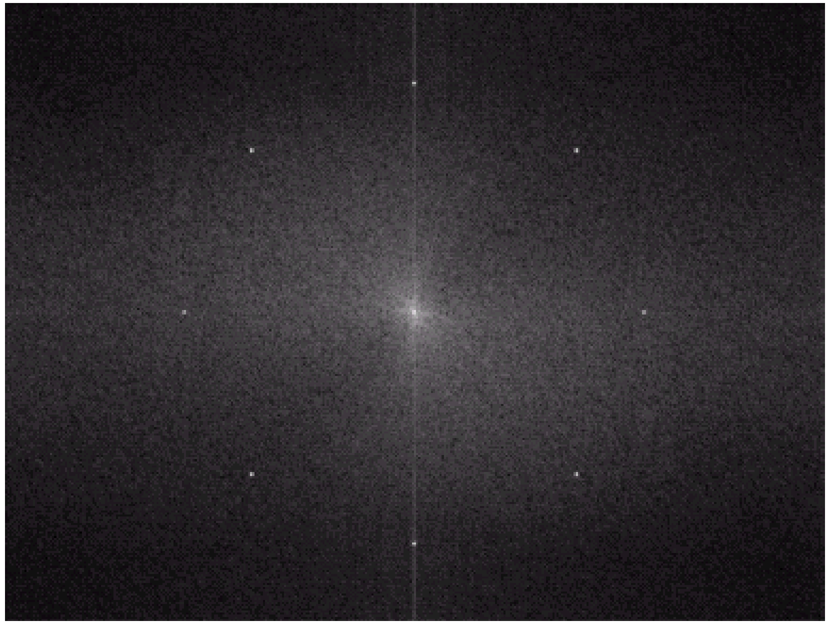
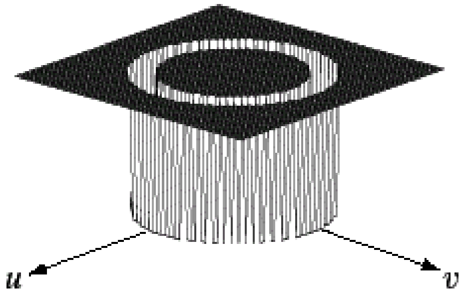


Photo by Digital Image Processing (3rd Edition), by Rafael C. Gonzalez and Richard E. Woods, Pearson, ISBN-13: 978-0131687288

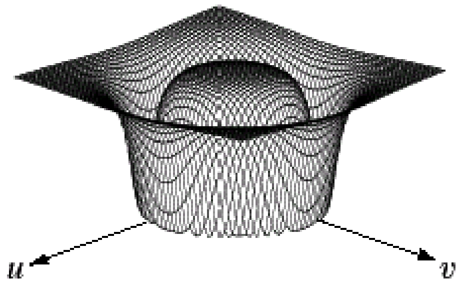
Band Reject Filters

- Removing periodic noise from an image involves removing a particular range of frequency components
- Band reject (stop) filters can be used for this purpose:

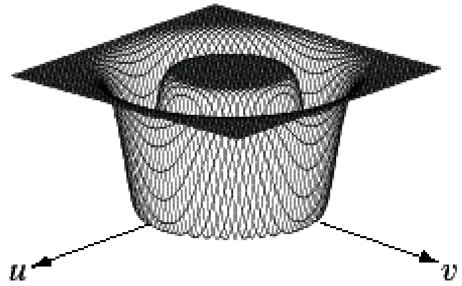
$$H(u,v) = \begin{cases} 1 & if\ D(u,v) < D_0 - \frac{W}{2} \\ 0 & if\ D_0 - \frac{W}{2} \le D(u,v) \le D_0 + \frac{W}{2} \\ 1 & if\ D(u,v) > D_0 + \frac{W}{2} \end{cases}$$



Ideal Band Reject Filter



Butterworth Band Reject Filter (of order 1)

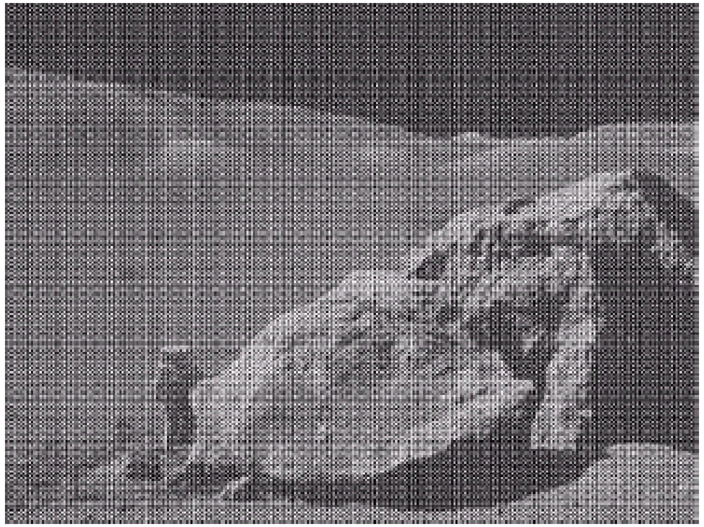


Gaussian Band Reject Filter

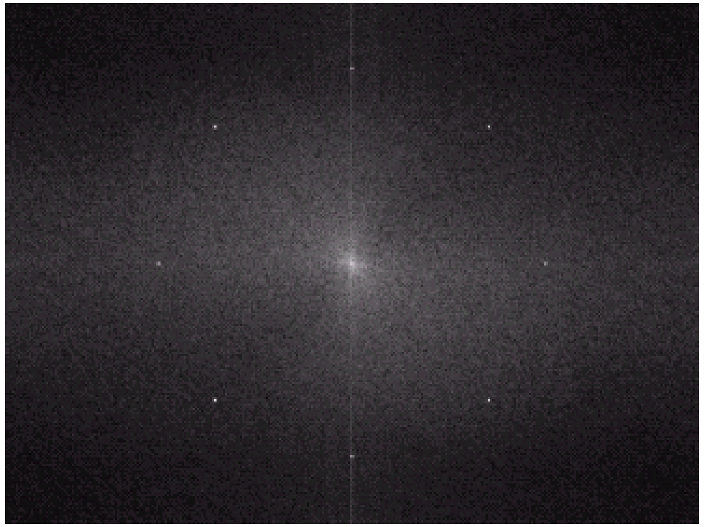
Photo by Digital Image Processing (3rd Edition), by Rafael C. Gonzalez and Richard E. Woods, Pearson, ISBN-13: 978-0131687288

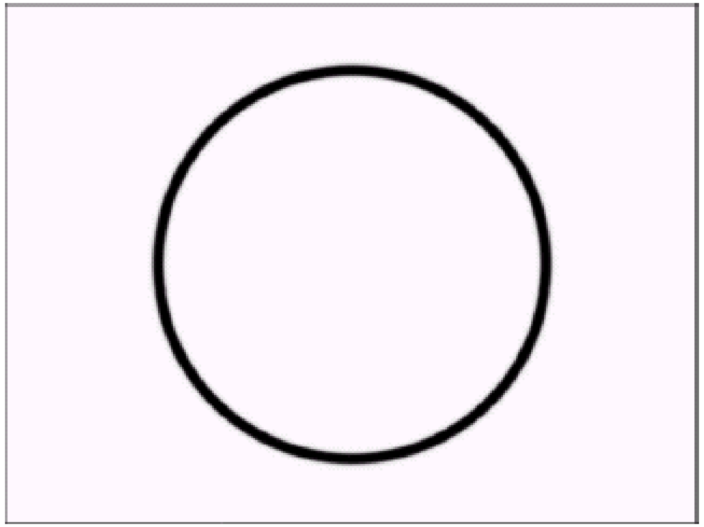
Band Reject Filter Result

Image corrupted by sinusoidal noise



Fourier spectrum of the corrupted image





Butterworth band reject filter

Filtered image

Photo by Digital Image Processing (3rd Edition), by Rafael C. Gonzalez and Richard E. Woods, Pearson, ISBN-13: 978-0131687288

Image Compression

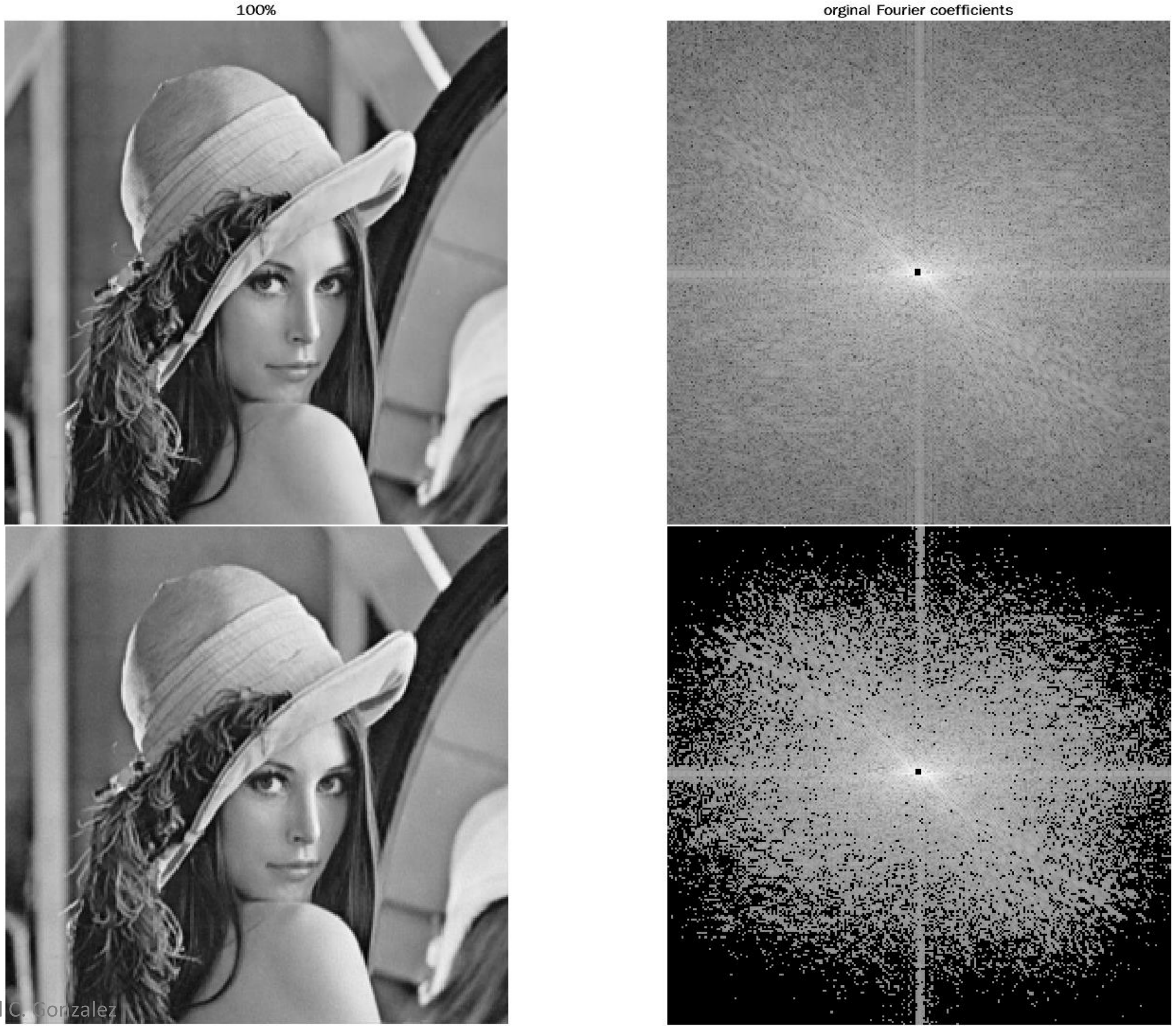


Photo by Digital Image Processing (3rd Edition), by Rafael C. Gonzalez and Richard E. Woods, Pearson, ISBN-13: 978-0131687288

Image Compression (JPEG)

- JPEG is developed based on the following observations
  - Image color usually varies slightly across an image, especially within an 8x8 block
  - Experiments indicate that humans are not very sensitive to the high-frequency components in images
    - Therefore, we can remove some high-frequency components using transform coding
  - Humans are much more sensitive to brightness (luminance) information than to color (chrominance)
- The steps of JPEG compression are the following
  1. Transform RGB to YUV and subsample the color
  2. Perform Discrete Cosine Transform on 8x8 image blocks
  3. Perform quantization
  4. Zig-zag ordering and run-length encoding
  5. Entropy coding

Discrete Cosine Transform

- Discrete Cosine Transform (DCT) is essentially the real part of the Fourier transform
- Forward DCT

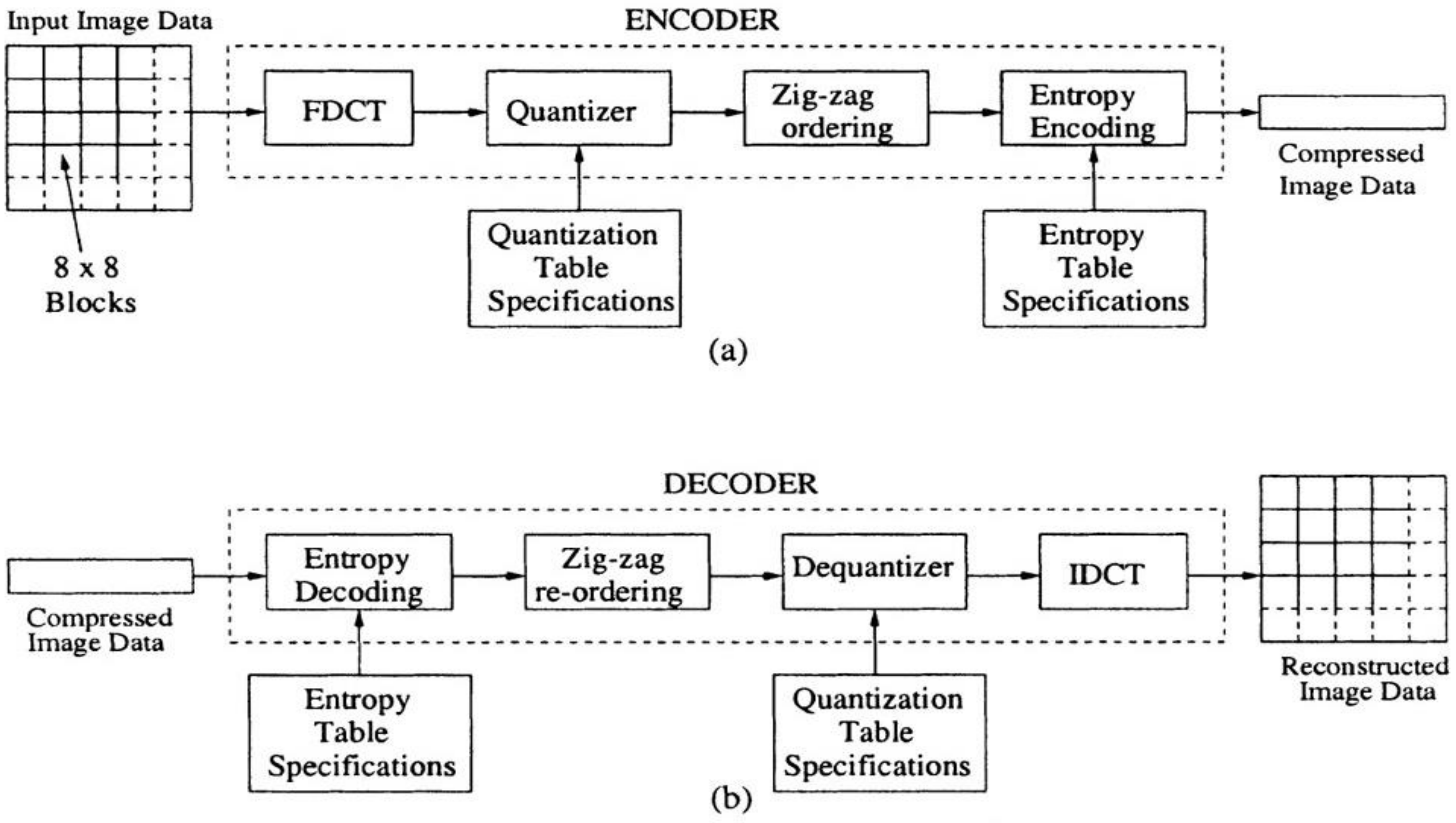
$$F(u,v)=\frac{2}{N}\sum_{i=0}^{N-1}\sum_{j=0}^{N-1}C(u)C(v)f(i,j)\cos\left(\frac{(2i+1)u\pi}{2N}\right)\cos\left(\frac{(2j+1)v\pi}{2N}\right)$$

- Inverse DCT

$$f(i,j)=\frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}C(u)C(v)F(u,v)\cos\left(\frac{(2i+1)u\pi}{2N}\right)\cos\left(\frac{(2j+1)v\pi}{2N}\right)$$

where $0\le i,j,u,v\le N-1$, $C(x)=\begin{cases}1/\sqrt{2} & (x=0)\\ 1 & (x\neq 0)\end{cases}$

JPEG Compression Overview



DCT on an Image Block and Quantization

- The image is divided up into 8x8 blocks
  - 2D DCT is performed on each block
  - The DCT is performed independently for each block
  - This is why, when a high degree of compression is requested, JPEG gives a “blocky” image result
- Quantization reduces the total number of bits
  - Divide each entry in the frequency space block by an integer, then round it

$$\hat{F}(u,v) = round(\frac{F(u,v)}{Q(u,v)})$$

- Use larger entries in *Q* for the higher spatial frequencies
- Multiple quantization matrices can be used, allowing the user to choose how much compression to use
  - Trades off quality vs. compression ratio

Typical Quantization Tables

Luminance Quantization Table

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Chrominance Quantization Table

17	18	24	47	99	99	99	99
18	21	26	66	99	99	99	99
24	26	56	99	99	99	99	99
47	66	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99
99	99	99	99	99	99	99	99

An Example

An 8×8 block from the Y image of 'Lena'

200	202	189	188	189	175	175	175
200	203	198	188	189	182	178	175
203	200	200	195	200	187	185	175
200	200	200	200	197	187	187	187
200	205	200	200	195	188	187	175
200	200	200	200	200	190	187	175
205	200	199	200	191	187	187	175
210	200	200	200	188	185	187	186

$f(i,j)$

515	65	-12	4	1	2	-8	5
-16	3	2	0	0	-11	-2	3
-12	6	11	-1	3	0	1	-2
-8	3	-4	2	-2	-3	-5	-2
0	-2	7	-5	4	0	-1	-4
0	-3	-1	0	4	1	-1	0
3	-2	-3	3	3	-1	-1	3
-2	5	-2	4	-2	2	-3	0

$F(u,v)$

After quantization

32	6	-1	0	0	0	0	0
-1	0	0	0	0	0	0	0
-1	0	1	0	0	0	0	0
-1	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

Zeros!

$\hat{F}(u,v)$

Reconstructed Fourier coefficients

512	66	-10	0	0	0	0	0
-12	0	0	0	0	0	0	0
-14	0	16	0	0	0	0	0
-14	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

$\tilde{F}(u,v)$

Inverse DCT transform

199	196	191	186	182	178	177	176
201	199	196	192	188	183	180	178
203	203	202	200	195	189	183	180
202	203	204	203	198	191	183	179
200	201	202	201	196	189	182	177
200	200	199	197	192	186	181	177
204	202	199	195	190	186	183	181
207	204	200	194	190	187	185	184

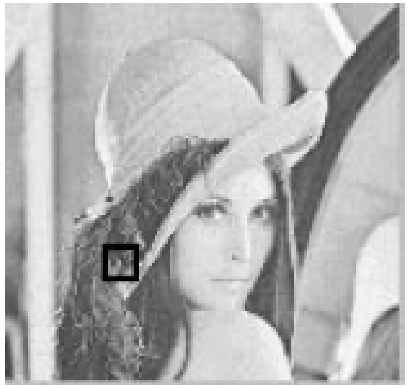
$\tilde{f}(i,j)$

Difference from the original

1	6	-2	2	7	-3	-2	-1
-1	4	2	-4	1	-1	-2	-3
0	-3	-2	-5	5	-2	2	-5
-2	-3	-4	-3	-1	-4	4	8
0	4	-2	-1	-1	-1	5	-2
0	0	1	3	8	4	6	-2
1	-2	0	5	1	1	4	-6
3	-4	0	6	-2	-2	2	2

$\epsilon(i,j) = f(i,j) - \tilde{f}(i,j)$

A Less Homogeneous Block



Another 8×8 block from the Y image of 'Lena'

70	70	100	70	87	87	150	187
85	100	96	79	87	154	87	113
100	85	116	79	70	87	86	196
136	69	87	200	79	71	117	96
161	70	87	200	103	71	96	113
161	123	147	133	113	113	85	161
146	147	175	100	103	103	163	187
156	146	189	70	113	161	163	197

$f(i, j)$

-80	-40	89	-73	44	32	53	-3
-135	-59	-26	6	14	-3	-13	-28
47	-76	66	-3	-108	-78	33	59
-2	10	-18	0	33	11	-21	1
-1	-9	-22	8	32	65	-36	-1
5	-20	28	-46	3	24	-30	24
6	-20	37	-28	12	-35	33	17
-5	-23	33	-30	17	-5	-4	20

$F(u, v)$

After quantization

-5	-4	9	-5	2	1	1	0
-11	-5	-2	0	1	0	0	-1
3	-6	4	0	-3	-1	0	1
0	1	-1	0	1	0	0	0
0	0	-1	0	0	1	0	0
0	-1	1	-1	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$\hat{F}(u,v)$

Cereals!

Reconstructed Fourier coefficients

-80	-44	90	-80	48	40	51	0
-132	-60	-28	0	26	0	0	-55
42	-78	64	0	-120	-57	0	56
0	17	-22	0	51	0	0	0
0	0	-37	0	0	109	0	0
0	-35	55	-64	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

$\tilde{F}(u,v)$

Inverse DCT transform

70	60	106	94	62	103	146	176
85	101	85	75	102	127	93	144
98	99	92	102	74	98	89	167
132	53	111	180	55	70	106	145
173	57	114	207	111	89	84	90
164	123	131	135	133	92	85	162
141	159	169	73	106	101	149	224
150	141	195	79	107	147	210	153

$\tilde{f}(i,j)$

Difference from Original

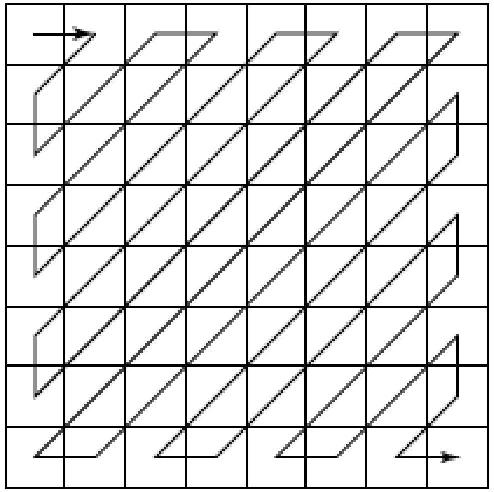
0	10	-6	-24	25	-16	4	11
0	-1	11	4	-15	27	-6	-31
2	-14	24	-23	-4	-11	-3	29
4	16	-24	20	24	1	11	-49
-12	13	-27	-7	-8	-18	12	23
-3	0	16	-2	-20	21	0	-1
5	-12	6	27	-3	2	14	-37
6	5	-6	-9	6	14	-47	44

$\epsilon(i,j) = f(i,j) - \tilde{f}(i,j)$

Run-Length Coding (RLC)

- After quantization we have many zero AC components
  - Note that most of the zero components are towards the lower right corner (high spatial frequencies)
  - To take advantage of this, use zigzag scanning to create a 64-vector
- Now the RLC step replaces values in a 64-vector by

  a pair (R, V)
  - R is the number of zeroes in the run and
  - V is the next non-zero value
- DC components are treated differently



−26, **−3, 0, −3**, −3, −6, 2, −4, 1 −4, 1, 1, 5, 1, 2, −1, 1, −1, **2**, 0, 0, 0, 0, 0, **−1, −1**, 0, ……,0.

The sequence can be expressed as: (0:-3),(1:-3),…,(0:2),(5:-1),(0:-1),EOB

Differential Pulse Code Modulation for the DC Coefficients

- Now we handle the DC coefficients
  - 1 DC component per block
  - DC coefficients may vary greatly over the whole image, but slowly from one block to its neighbor (once again, zigzag order)
  - Apply Differential Pulse Code Modulation (DPCM) for the DC coefficients
  - If the first five DC coefficients are 150, 155, 149, 152, 144, we come up with DPCM codes: 150, 5, -6, 3, -8

Entropy Coding

- Entropy coding is applied to the RLC coded AC coefficients and the DPCM coded DC coefficients
  - The baseline entropy coding method uses Huffman coding on images with 8-bit components
    - Encode the high/low probability symbols with short/long code length.
  - DPCM-coded DC coefficients are represented by a pair of symbols (SIZE, AMPLITUDE)
    - SIZE: number of bits to represent coefficient
    - AMPLITUDE: the actual bits