Midterm is on Thursday!

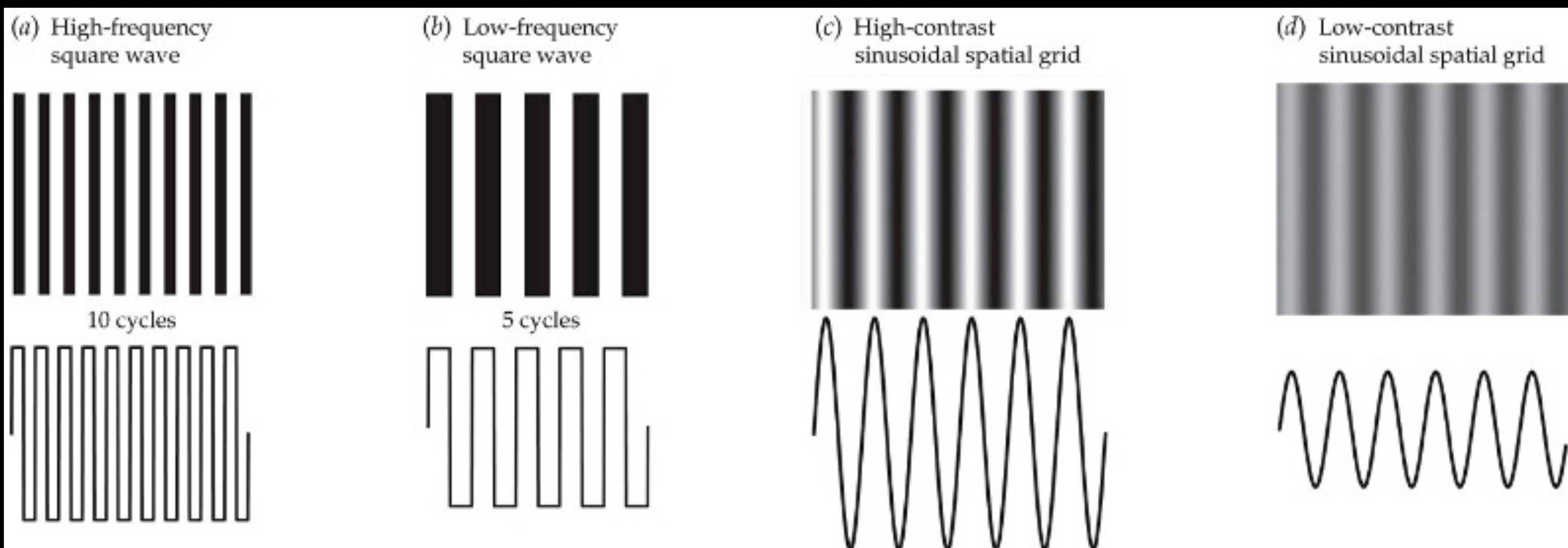
Project presentations are
May 17th, 22nd and 24th

Next week there is a strike on campus.
Class is therefore cancelled on Tuesday.
Please work on your presentations instead!

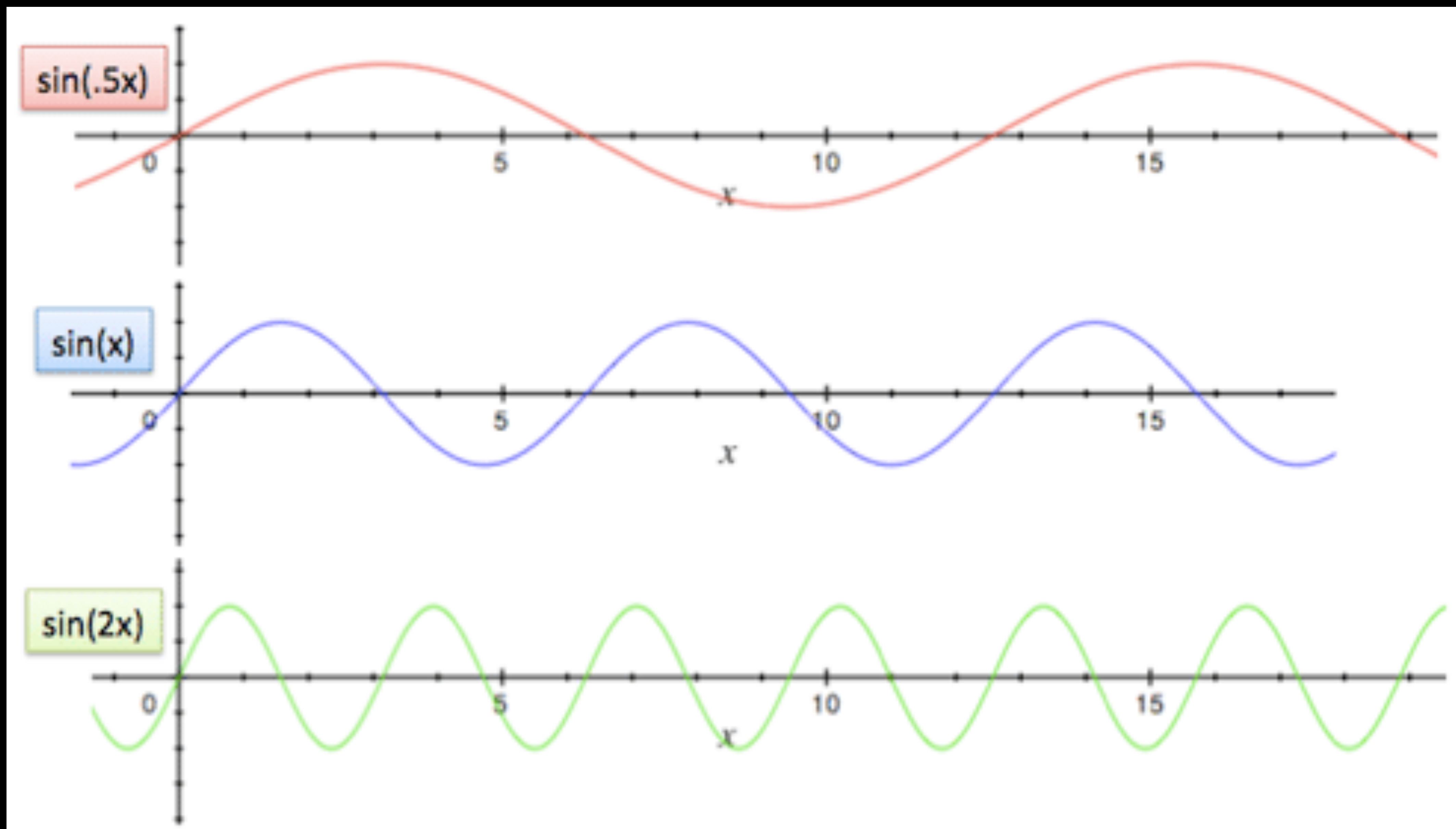
REVIEW

From results on practice midterm:
Half of the class needs another review!

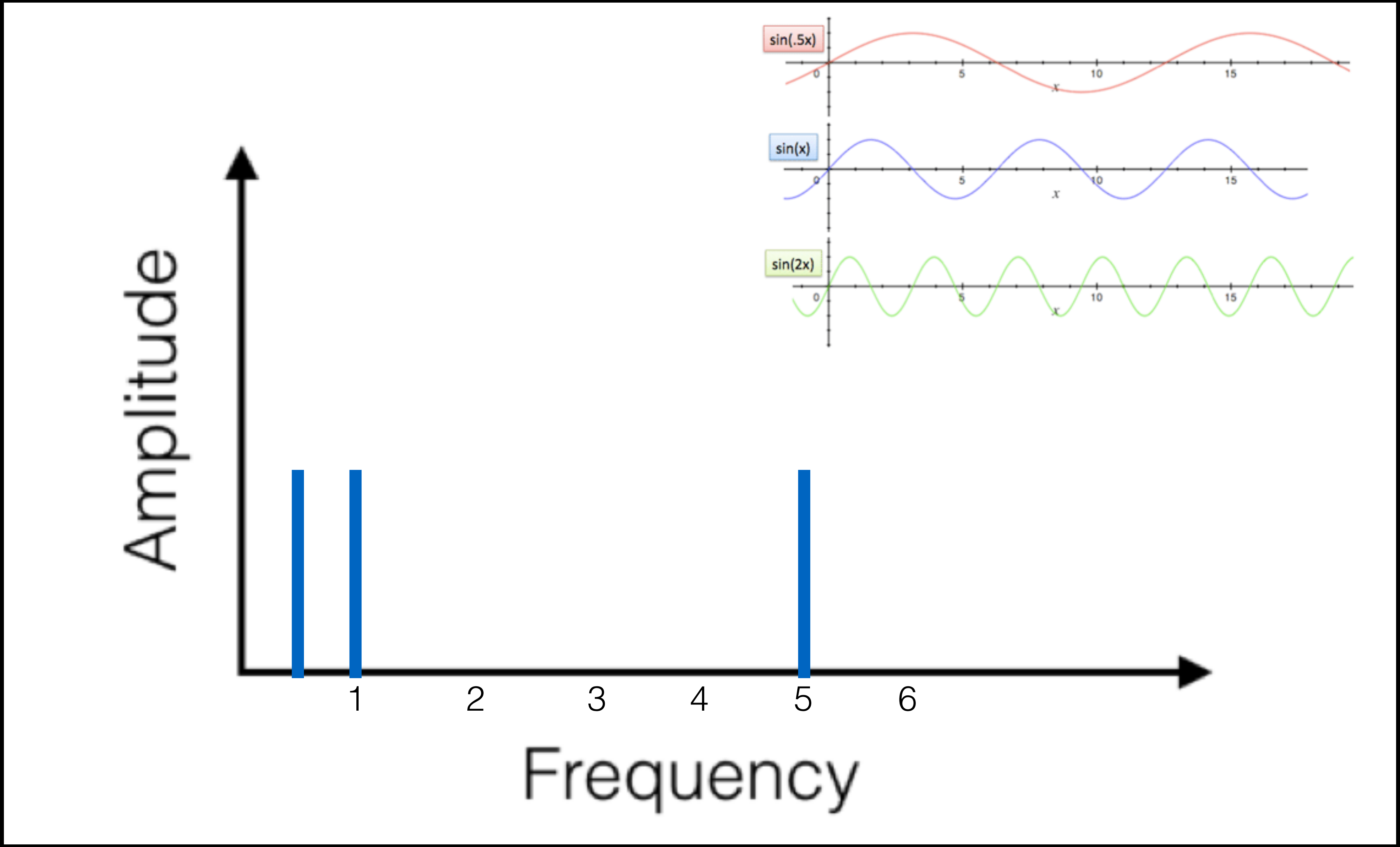
Spatial frequency



1D Frequency Spectrum

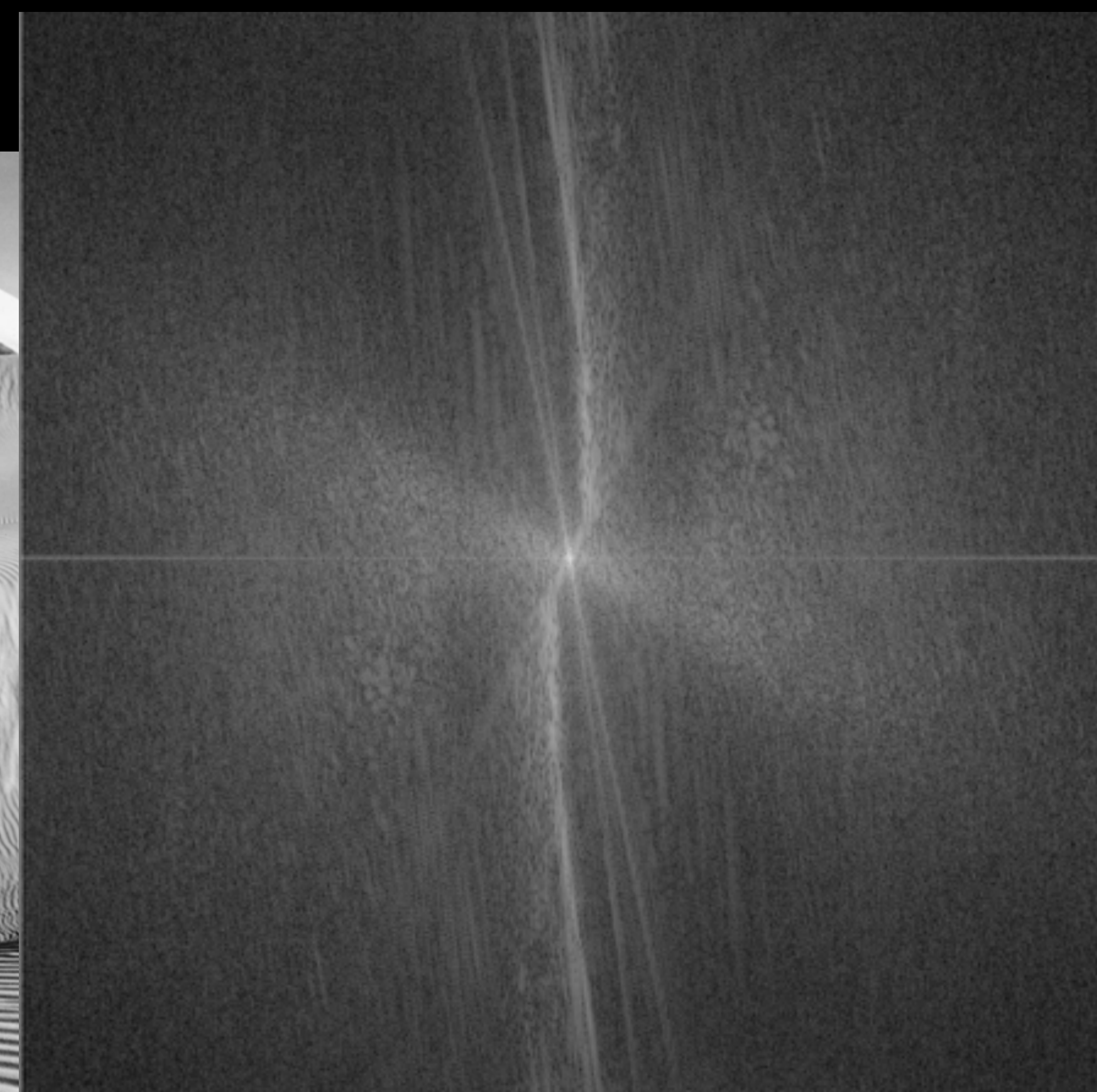


Frequency spectrum

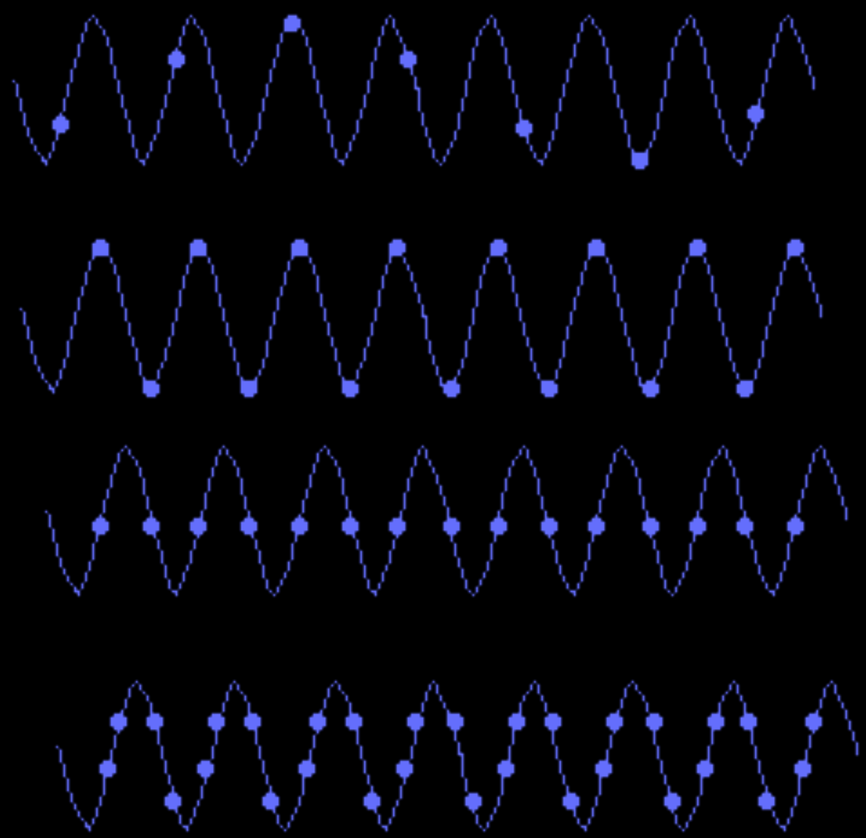


2D Fourier Transform Power Spectrum

- 2D Fourier transform of the *Dunes* photograph

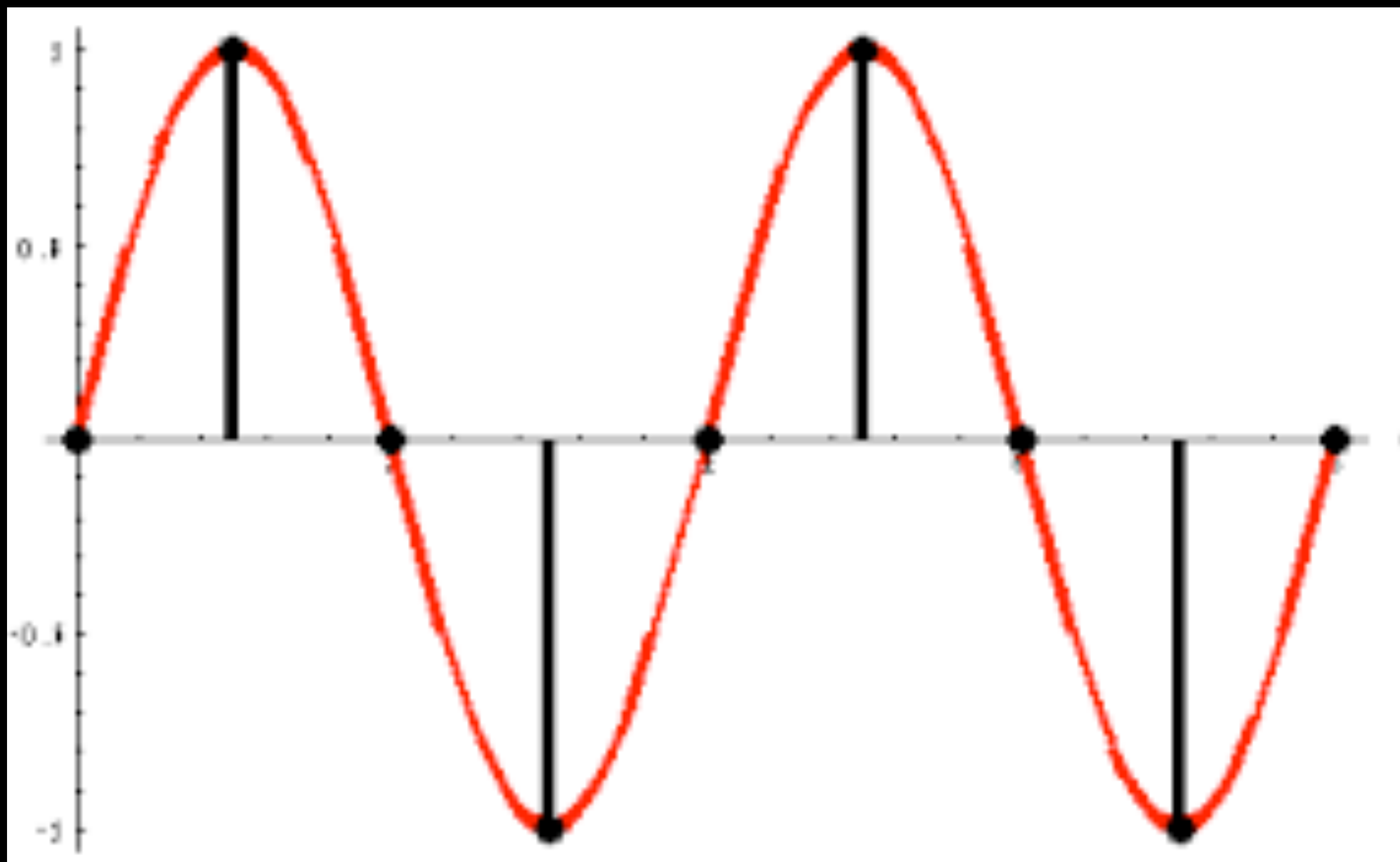


Sampling

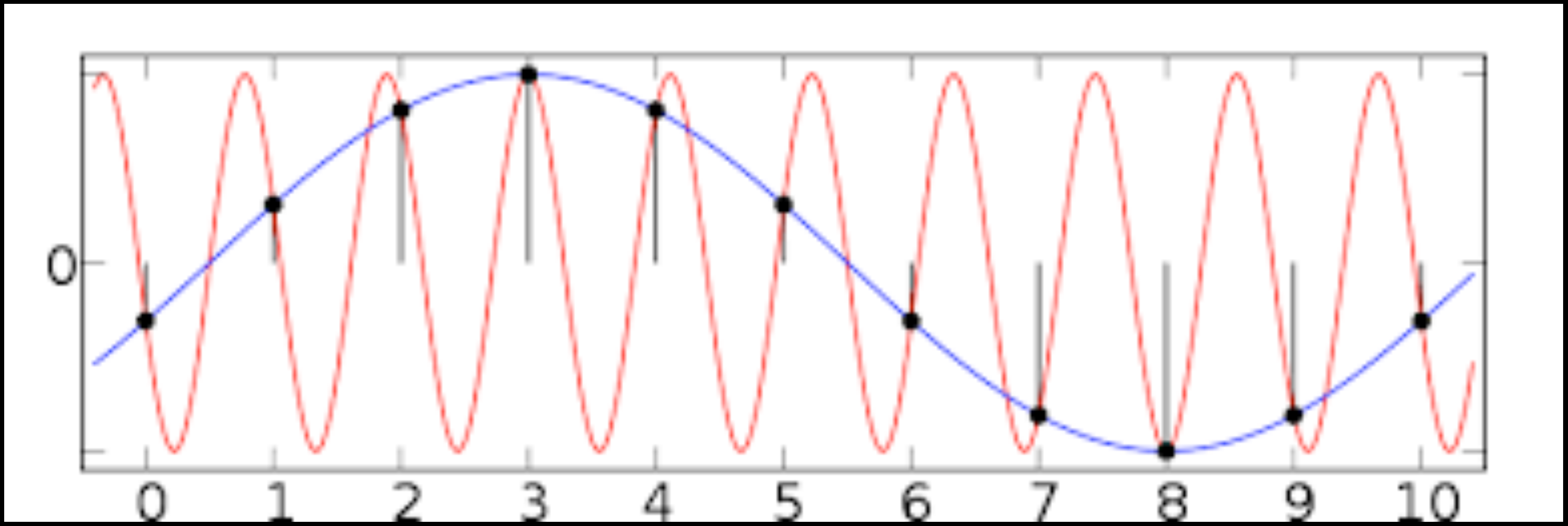


- Both in space and time
- Select acquisition speed
- Select sensor size / Optical magnification
- Aliasing

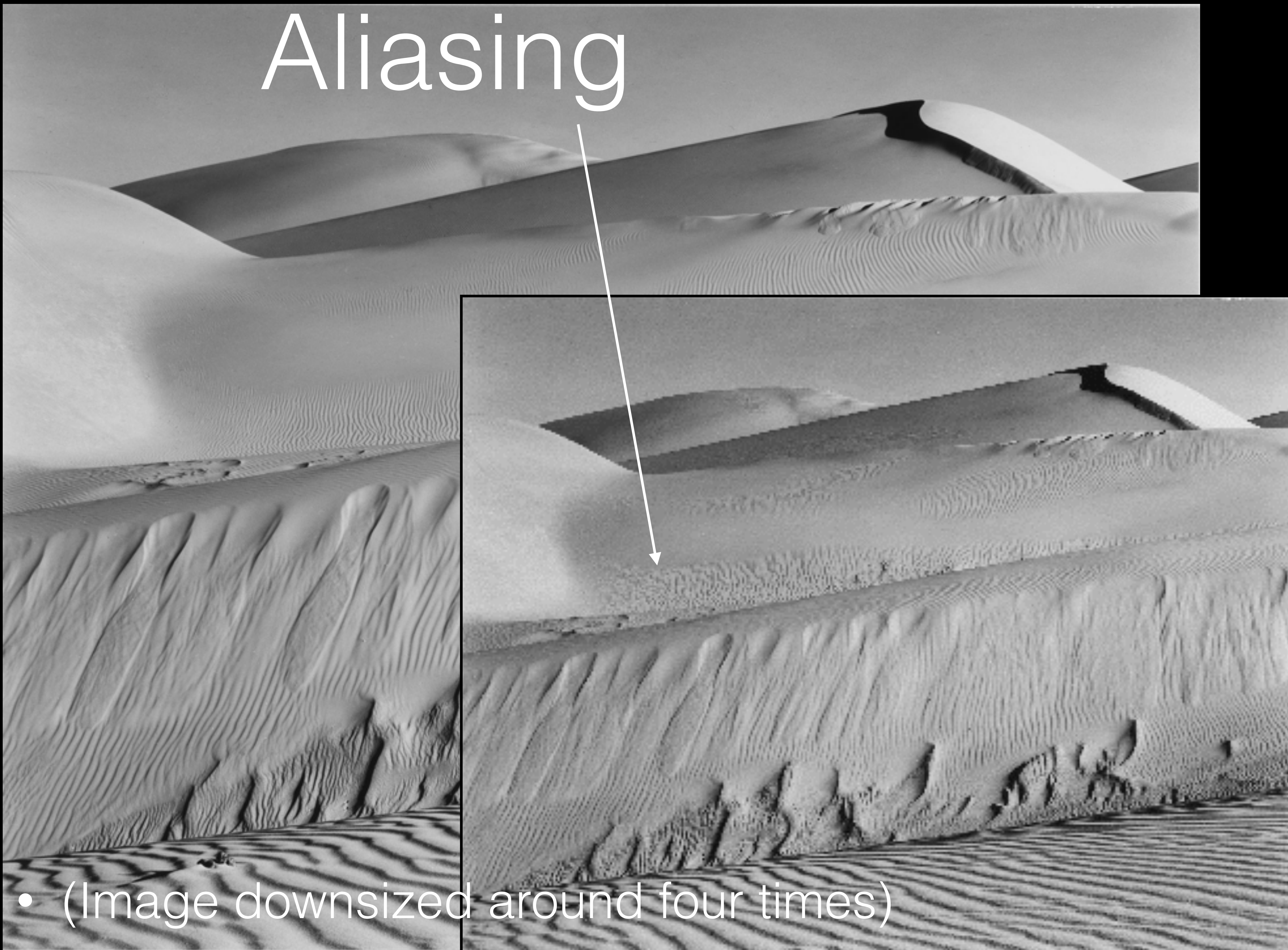
Nyquist sampling
= twice the frequency



Aliasing

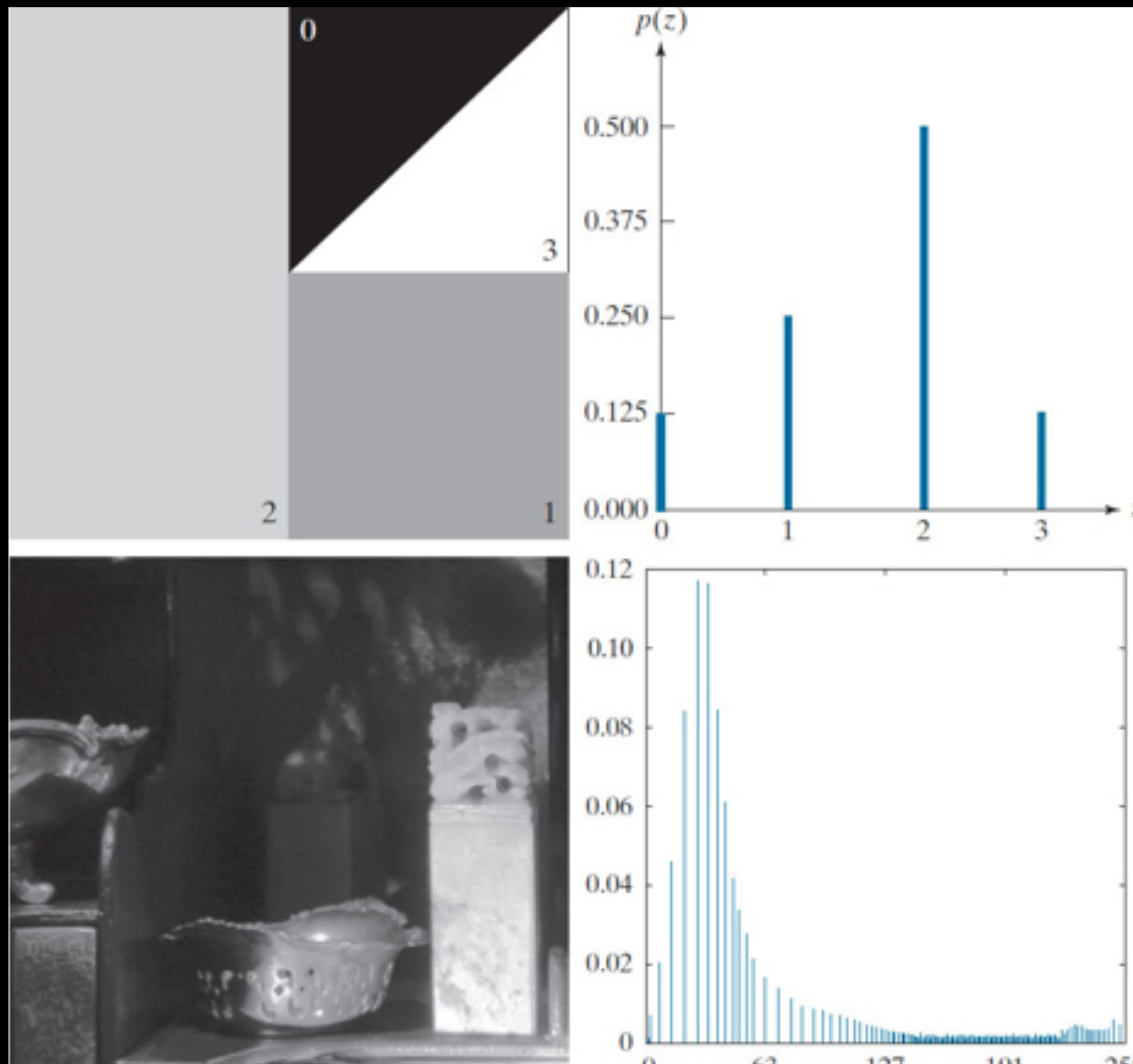


Aliasing

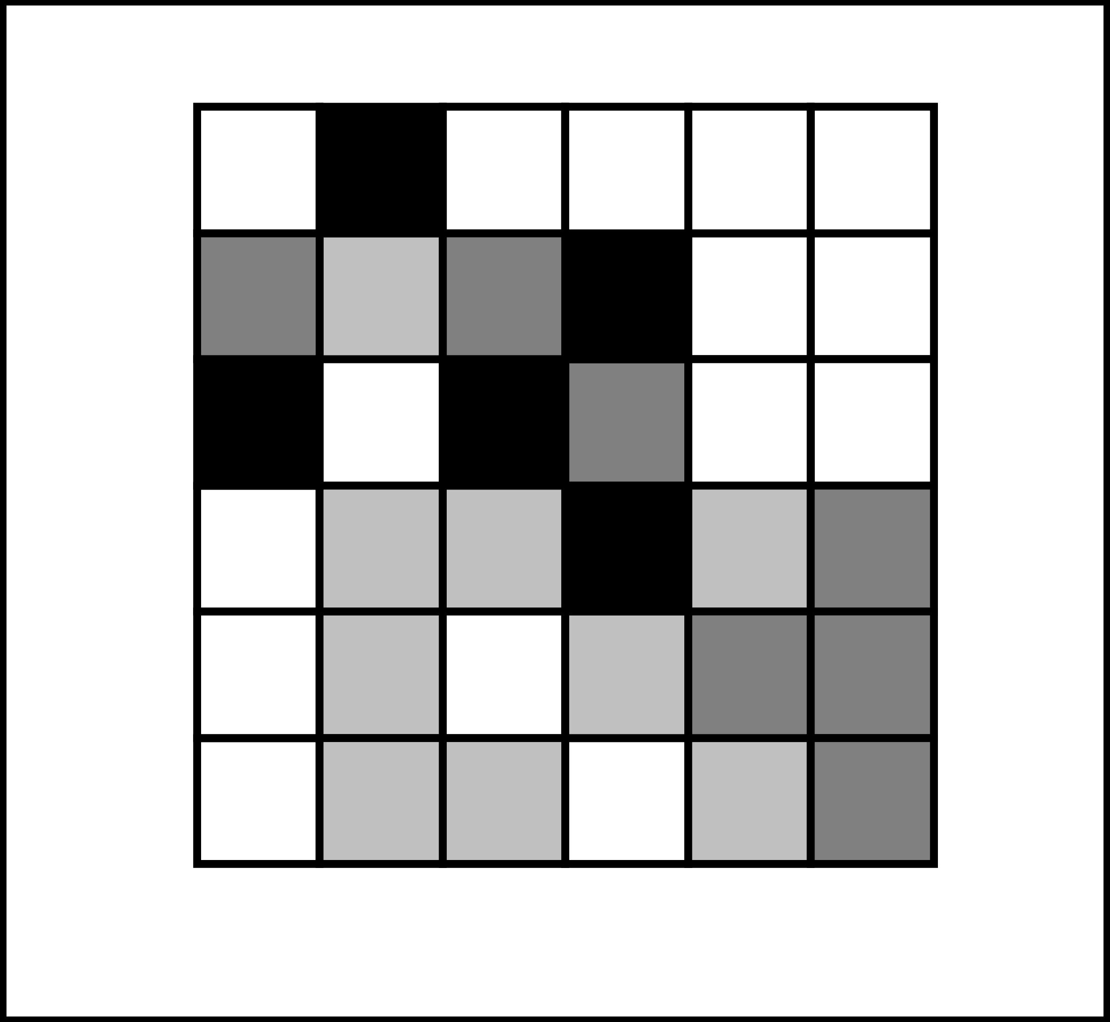


- (Image downsized around four times)

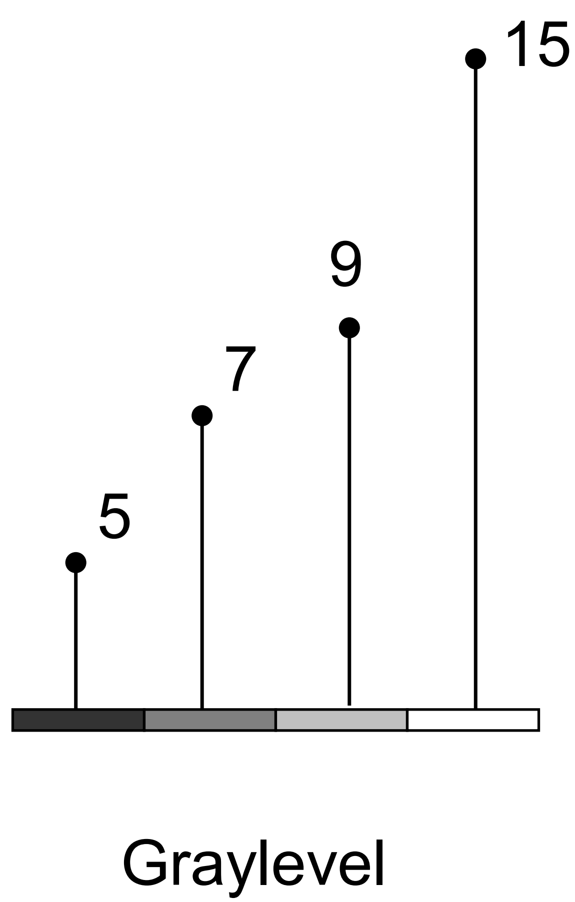
Histogram shows distribution of gray-levels in an image

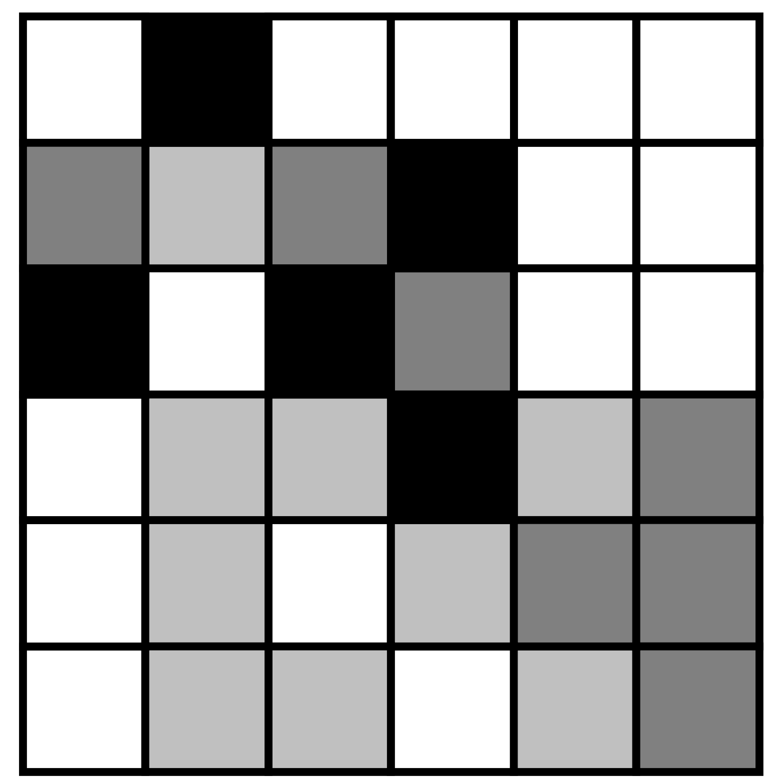


Lets plot the histogram of this synthetic image

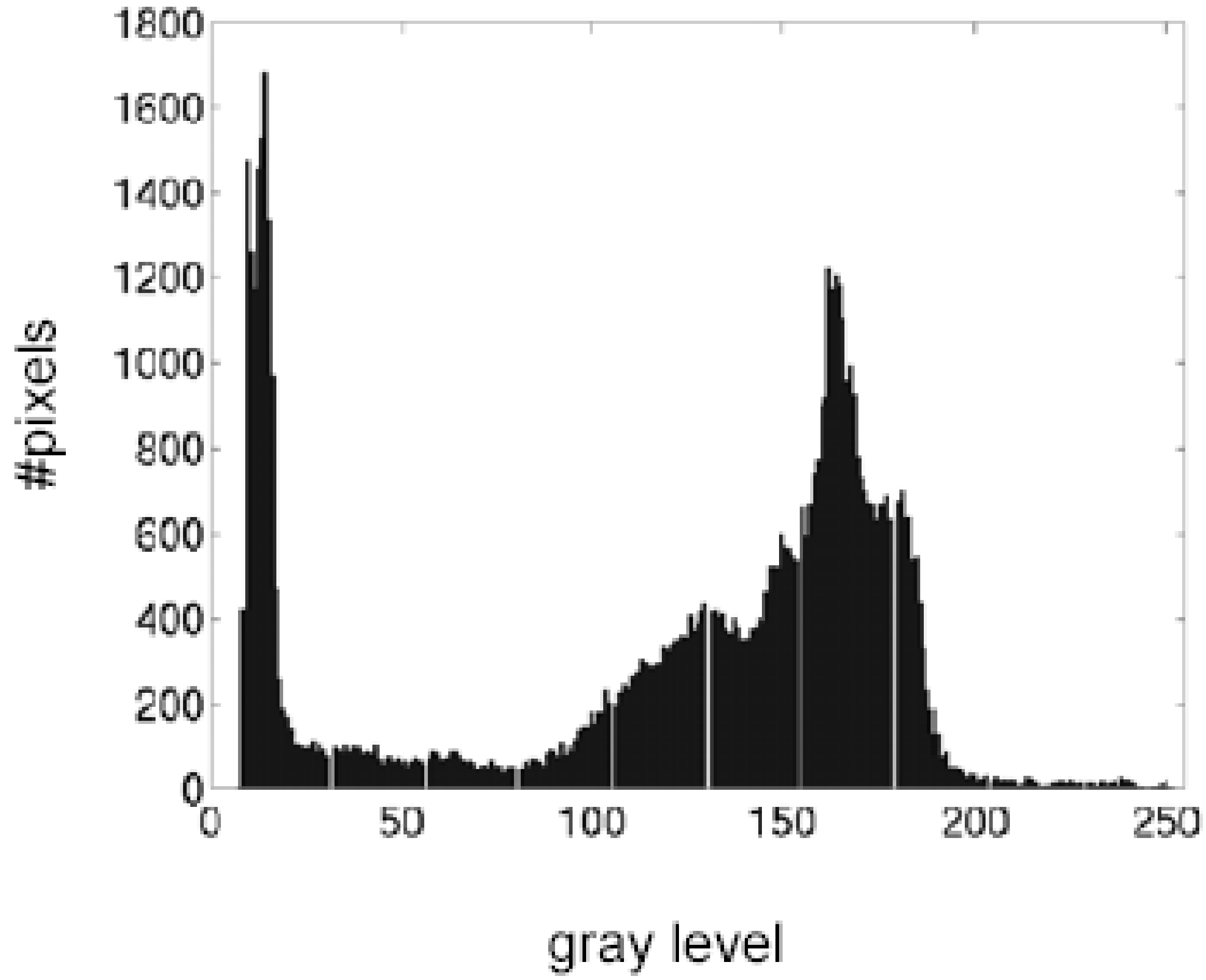


Histogram:





Grayscale Histogram

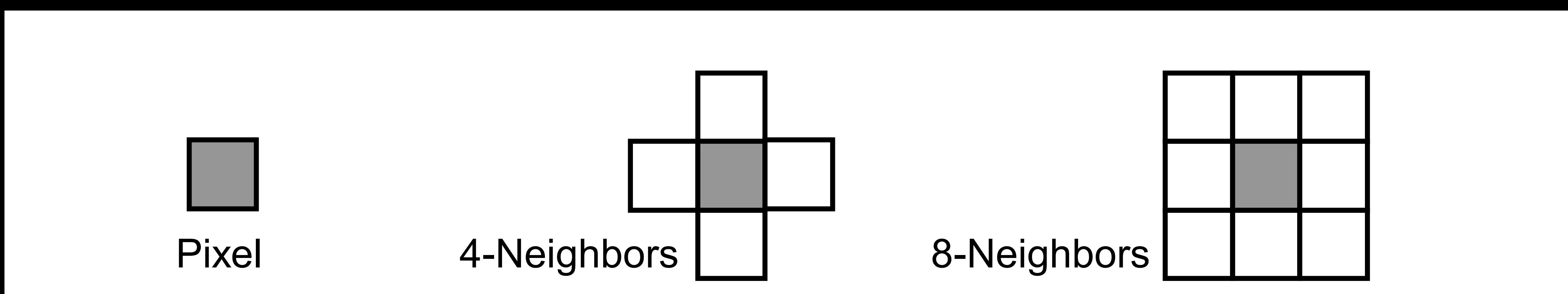


Cameraman image

Color Histogram

Image enhancement

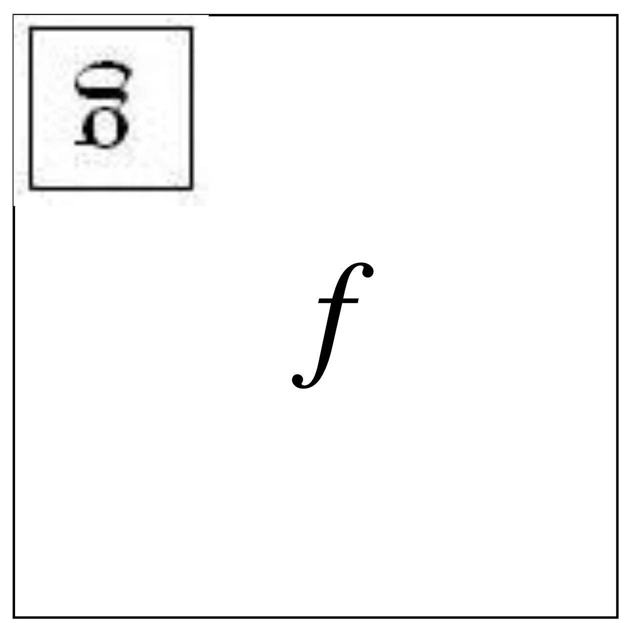
- In the pixel domain, for example:



Convolution

- Let *f* be the image and *g* be the kernel. The output of convolving *f* with *g* is denoted *f* * *g*.

$$(f * g)[m,n] = \sum_{k,l} f[m-k, n-l]\, g[k,l]$$



- Convention: kernel is “flipped”
- MATLAB: conv2 vs. filter2 (also imfilter)

Convolution properties

- **Linearity:** filter(f_1 + f_2) = filter(f_1) + filter(f_2)
- **Shift invariance:** same behavior regardless of pixel location: filter(shift(f)) = shift(filter(f))
- Theoretical result: any linear shift-invariant operator can be represented as a convolution

“Drag-and-Stamp”

- Local linear oprations on an image

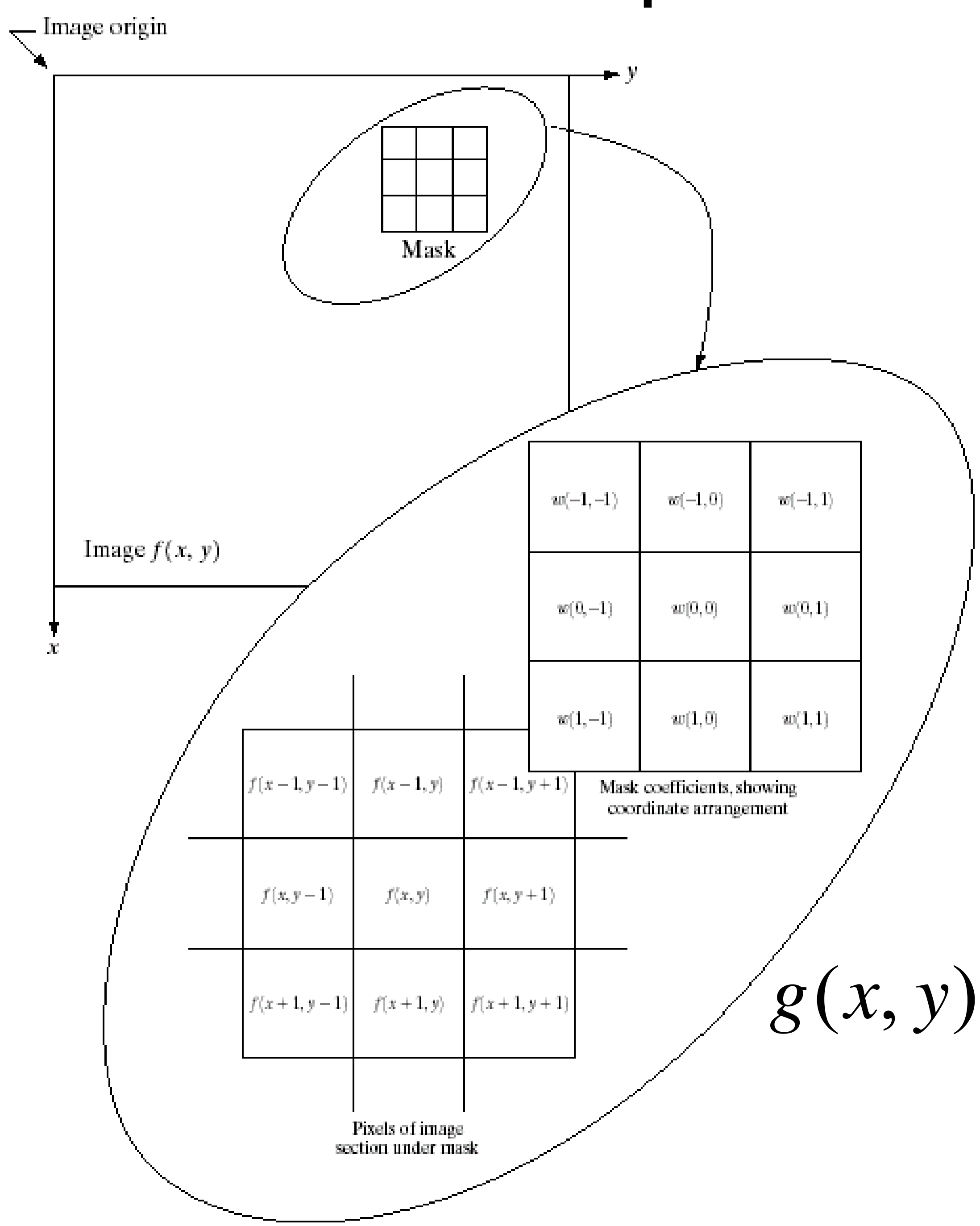


FIGURE 3.32 The mechanics of spatial filtering. The magnified drawing shows a 3 × 3 mask and the image section directly under it; the image section is shown displaced out from under the mask for ease of readability.

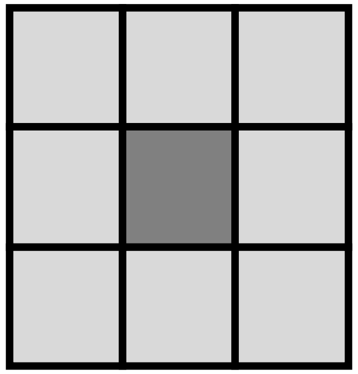
w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

Input: f(x,y), Output: g(x,y)

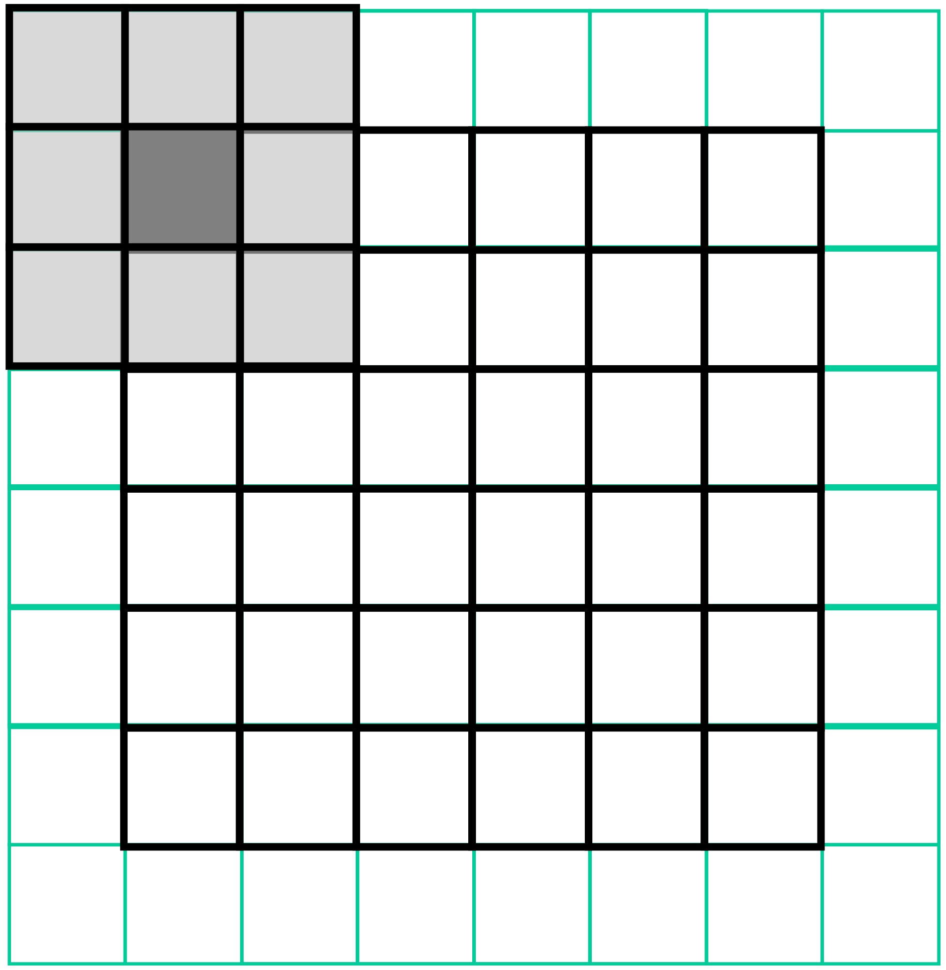
$$g(x, y) = w_1 f(x-1, y-1) + w_2 f(x-1, y) + \cdots$$
$$w_8 f(x+1, y-1) + w_9 f(x+1, y+1)$$

What to do at the edges of the image

- An important point: Edge Effects
  - To compute all pixel values in the output image, we need to fill in a “border”



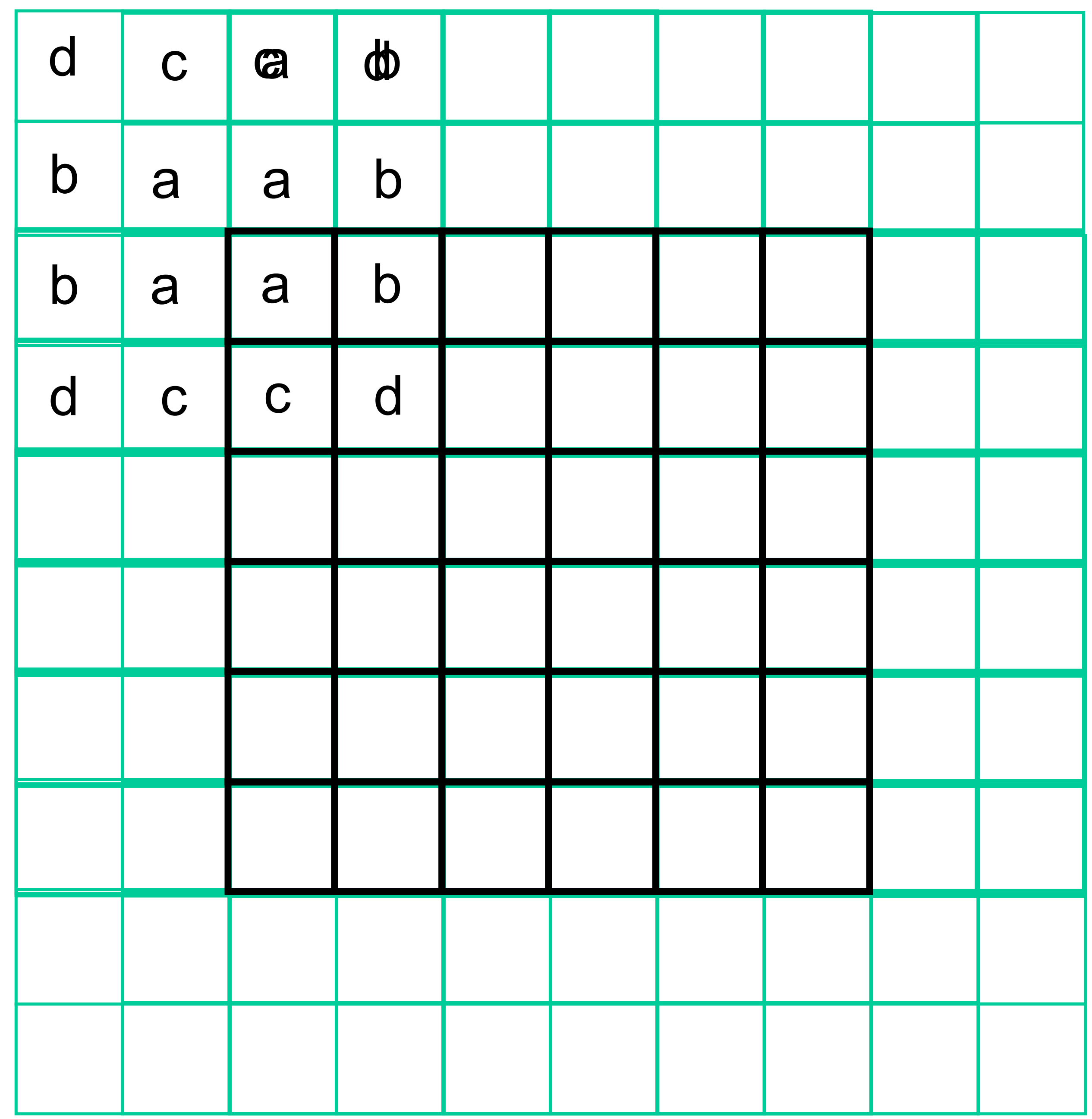
Mask dimension = 2M+1



Border dimension = M

Image Enhancement:Spatial Filtering Operation

- An important point: Edge Effects (Ex.: 5x5 Mask)
  - How to fill in a “border”
    - Zeros (Ringing)
    - Replication (Better)
    - Reflection (“Best”)



- Procedure:
  - Replicate row-wise
  - Replicate column-wise
  - Apply filtering
  - Remove borders

MATLAB methods:

• clip filter (black):	imfilter(f, g, 0)
• wrap around:	imfilter(f, g, ‘circular’)
• copy edge:	imfilter(f, g, ‘replicate’)
• reflect across edge:	imfilter(f, g, ‘symmetric’)

What happens:

- Filter kernel operations on matrix (image)
- “Drag and Stamp” operation
- Using MATLAB: filter2(A,B)
    - use: ’same’, ‘valid’ and ‘full’ parameters (ex: filter2(A,B,’full’)

Filtering example

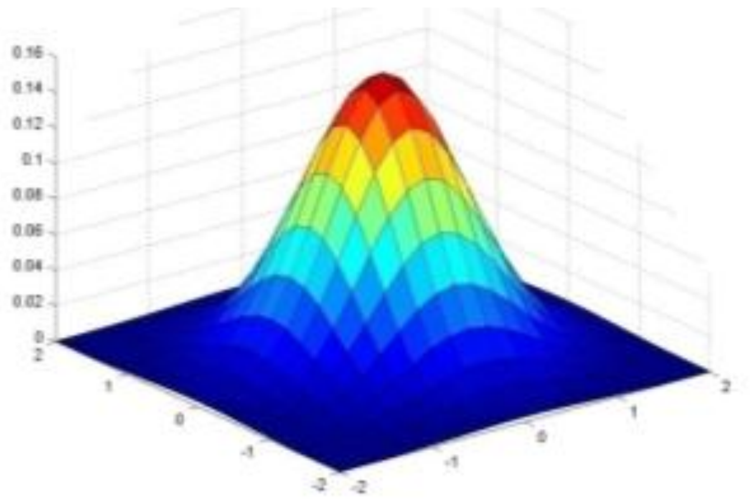
- MATLAB code how to apply filter kernel:

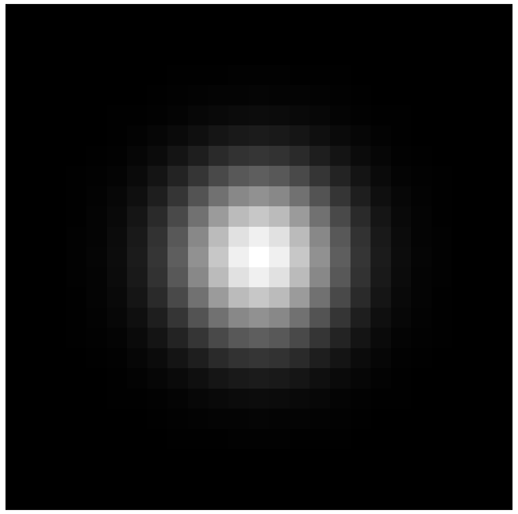
Gaussian blur filtering

- Filter with Gaussian kernel
- Filtering twice with a certain Gaussian is the same as filtering with this Gaussian convolved with itself -> MORE BLUR

Gaussian Kernel

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





0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

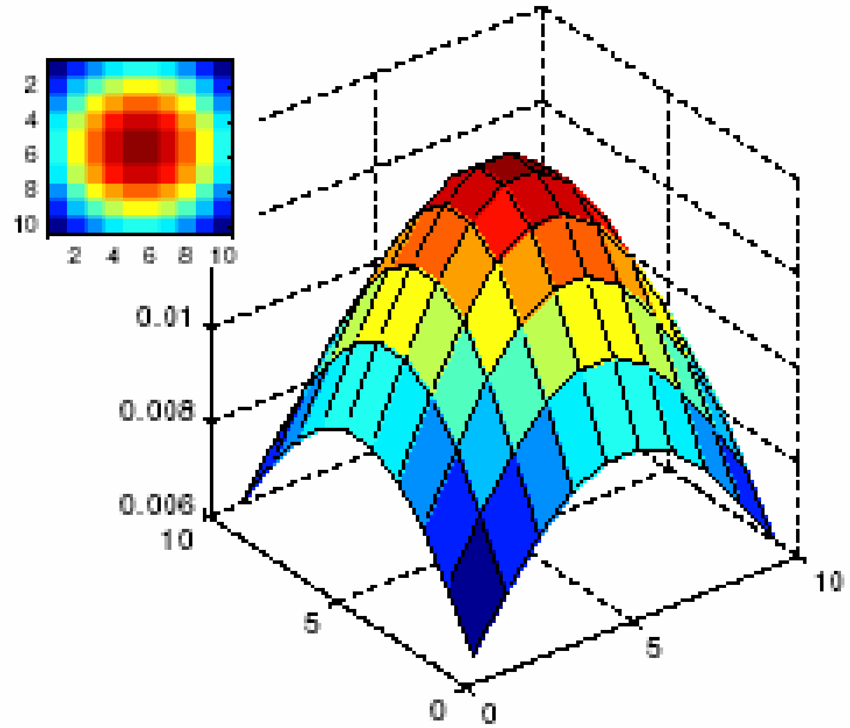
5 x 5, σ = 1

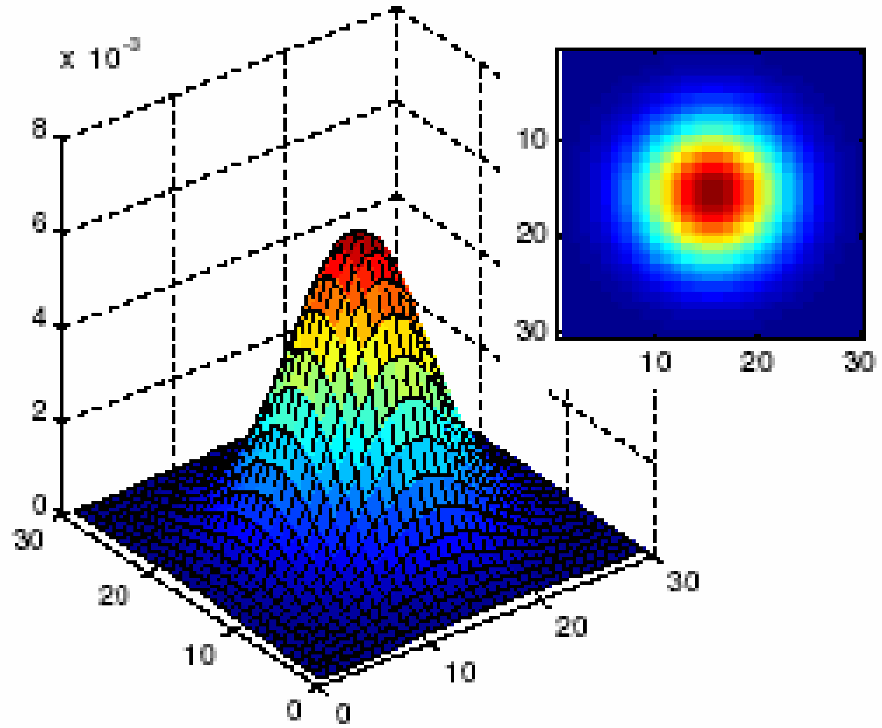
fspecial('gauss',5,1)

- Constant factor at front makes volume sum to 1 (can be ignored, as we should re-normalize weights to sum to 1 in any case)

Choosing kernel width

- Gaussian filters have infinite support, but discrete filters use finite kernels



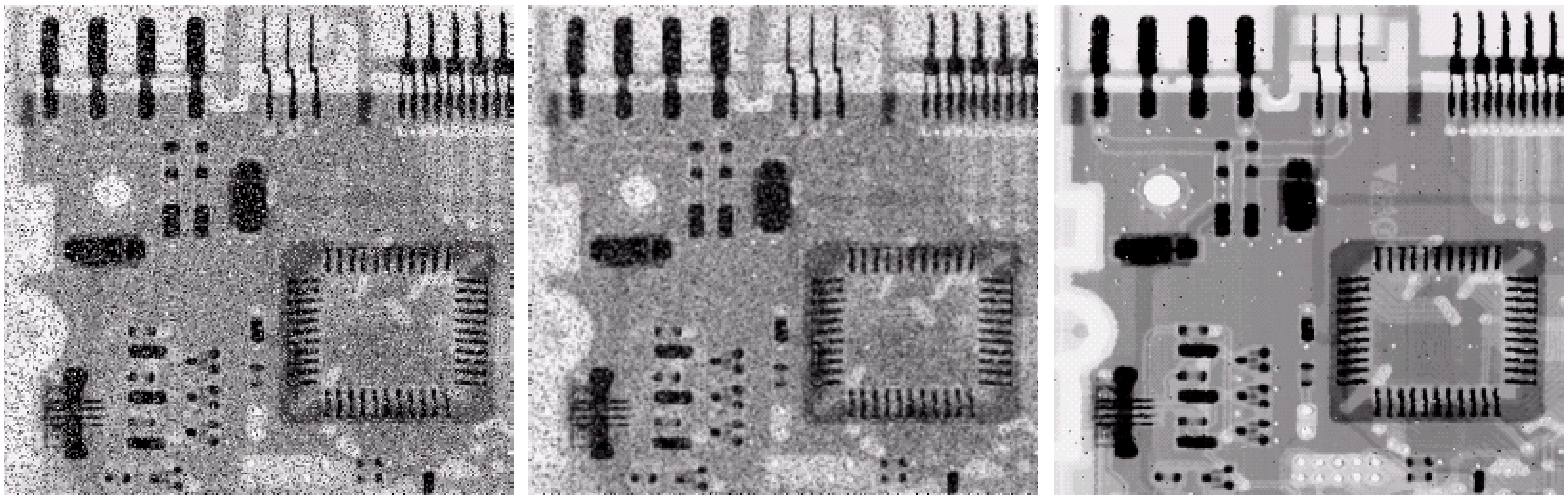


Example code MATLAB

```
ImageO = imread('MushroomDownload.jpg');
Image = ImageO(:,:,1);

H = fspecial('motion',20,45);
MotionBlur = imfilter(Image,H,'replicate');
subplot(2,1,2), imagesc(MotionBlur),axis equal, axis off,
colormap gray
subplot(2,1,1), imagesc(Image),axis equal, axis off, colormap
gray
```

Blurring can be used for denoising



a b c

FIGURE 3.37 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

- (Example from course textbook Gonzales & Wood)

Try this out on noisy data with median, gaussian etc…

Note: it's better to vectorize your code in MATLAB

```
tic
for t = 1:1024
   y(t) = sin(2*pi*t/512);
end
toc
```

```
tic
y = sin(pi*(0:(1/256):2));
toc
```

You can read up on this at:
http://www.mathworks.com/help/matlab/matlab_prog/vectorization.html

Fourier Transform

- Examples of spatial filtering in frequency space

Some Fourier Transform Pairs

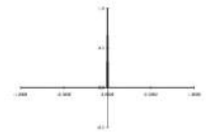
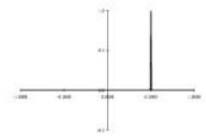
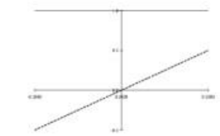
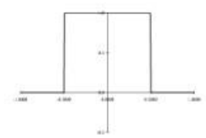
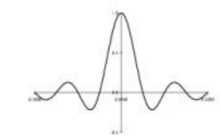
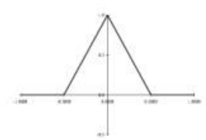
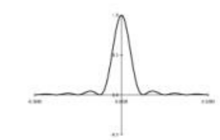
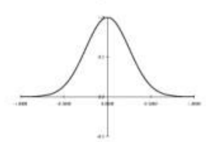
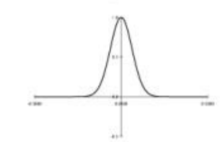
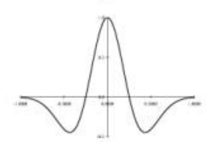
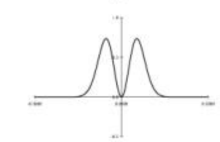
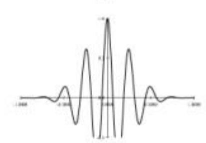
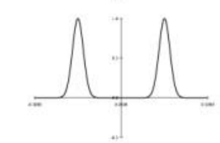
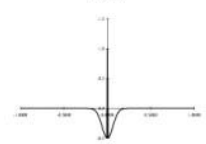
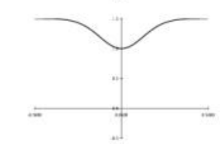
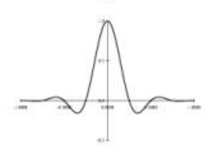
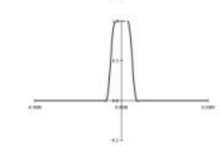
Name		Signal		Transform	
impulse		$\delta(x)$	$\Leftrightarrow$	1	
shifted impulse		$\delta(x-u)$	$\Leftrightarrow$	$e^{-j\omega u}$	
box filter		$\mathrm{box}(x/a)$	$\Leftrightarrow$	$a\,\mathrm{sinc}(a\omega)$	
tent		$\mathrm{tent}(x/a)$	$\Leftrightarrow$	$a\,\mathrm{sinc}^2(a\omega)$	
Gaussian		$G(x;\sigma)$	$\Leftrightarrow$	$\frac{\sqrt{2\pi}}{\sigma}G(\omega;\sigma^{-1})$	
Laplacian of Gaussian		$(\frac{x^2}{\sigma^4}-\frac{1}{\sigma^2})G(x;\sigma)$	$\Leftrightarrow$	$-\frac{\sqrt{2\pi}}{\sigma}\omega^2 G(\omega;\sigma^{-1})$	
Gabor		$\cos(\omega_0 x)G(x;\sigma)$	$\Leftrightarrow$	$\frac{\sqrt{2\pi}}{\sigma}G(\omega\pm\omega_0;\sigma^{-1})$	
unsharp mask		$(1+\gamma)\delta(x)$ $-\gamma G(x;\sigma)$	$\Leftrightarrow$	$(1+\gamma)-$ $\frac{\sqrt{2\pi}\gamma}{\sigma}G(\omega;\sigma^{-1})$	
windowed sinc		$\mathrm{rcos}(x/(aW))$ $\mathrm{sinc}(x/a)$	$\Leftrightarrow$	(see Figure 3.29)	

Table: Richard Szeliski, *Computer Vision and Applications*, Springer, 2010, ISBN 978-1-84882-935-0, p.137, http://szeliski.org/Book/.

Fourier Transform Pairs

Function, f(t)	**Fourier Transform, F(ω)**
Definition of Inverse Fourier Transform $f(t)=\frac{1}{2\pi}\int_{-\infty}^{\infty}F(\omega)e^{j\omega t}\,d\omega$	***Definition of Fourier Transform*** $F(\omega)=\int_{-\infty}^{\infty}f(t)e^{-j\omega t}\,dt$
$f(t-t_0)$	$F(\omega)e^{-j\omega t_0}$
$f(t)e^{j\omega_0 t}$	$F(\omega-\omega_0)$
$f(\alpha t)$	$\frac{1}{\lvert\alpha\rvert}F\left(\frac{\omega}{\alpha}\right)$
$F(t)$	$2\pi f(-\omega)$
$\frac{d^n f(t)}{dt^n}$	$(j\omega)^n F(\omega)$
$(-jt)^n f(t)$	$\frac{d^n F(\omega)}{d\omega^n}$
$\int_{-\infty}^{t} f(\tau)d\tau$	$\frac{F(\omega)}{j\omega}+\pi F(0)\delta(\omega)$
$\delta(t)$	1
$e^{j\omega_0 t}$	$2\pi\delta(\omega-\omega_0)$
$\mathrm{sgn}\,(t)$	$\frac{2}{j\omega}$

Function, f(t)	**Fourier Transform, F(ω)**
$j\frac{1}{\pi t}$	$\mathrm{sgn}(\omega)$
$u(t)$	$\pi\delta(\omega)+\frac{1}{j\omega}$
$\sum_{n=-\infty}^{\infty}F_n e^{jn\omega_0 t}$	$2\pi\sum_{n=-\infty}^{\infty}F_n\delta(\omega-n\omega_0)$
$\mathrm{rect}\left(\frac{t}{\tau}\right)$	$\tau\mathrm{Sa}\left(\frac{\omega\tau}{2}\right)$
$\frac{B}{2\pi}\mathrm{Sa}\left(\frac{Bt}{2}\right)$	$\mathrm{rect}\left(\frac{\omega}{B}\right)$
$\mathrm{tri}(t)$	$\mathrm{Sa}^2\left(\frac{\omega}{2}\right)$
$A\cos\left(\frac{\pi t}{2\tau}\right)\mathrm{rect}\left(\frac{t}{2\tau}\right)$	$\frac{A\pi}{\tau}\frac{\cos(\omega\tau)}{\left(\pi/2\tau\right)^2-\omega^2}$
$\cos(\omega_0 t)$	$\pi\left[\delta(\omega-\omega_0)+\delta(\omega+\omega_0)\right]$
$\sin(\omega_0 t)$	$\frac{\pi}{j}\left[\delta(\omega-\omega_0)-\delta(\omega+\omega_0)\right]$
$u(t)\cos(\omega_0 t)$	$\frac{\pi}{2}\left[\delta(\omega-\omega_0)+\delta(\omega+\omega_0)\right]+\frac{j\omega}{\omega_0^2-\omega^2}$
$u(t)\sin(\omega_0 t)$	$\frac{\pi}{2j}\left[\delta(\omega-\omega_0)-\delta(\omega+\omega_0)\right]+\frac{\omega^2}{\omega_0^2-\omega^2}$
$u(t)e^{-\alpha t}\cos(\omega_0 t)$	$\frac{(\alpha+j\omega)}{\omega_0^2+(\alpha+j\omega)^2}$

Lets do an experiment with spatial frequency imaging

- Take an image of the line pairs with your cell phone
- If you don't have one, come up here and take them with a camera
- Download to your computer and Fourier transform the data. Can you recognize the spatial frequencies from the image?