Computer Vision

Spring 2006 15-385,-685

Instructor: S. Narasimhan

Wean 5403

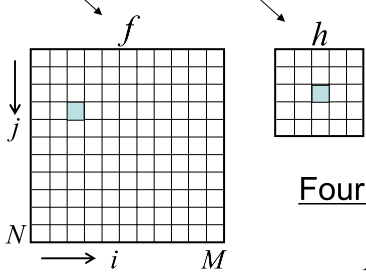
T-R 3:00pm – 4:20pm

Image Processing and Filtering (continued)

Lecture #6

Images are Discrete and Finite

$$f(x,y) \longrightarrow \boxed{h(x,y)} \longrightarrow g(x,y)$$



<u>Convolution</u>

$$g(i,j)=\sum_{m=1}^{M}\sum_{n=1}^{N} f(m,n)h(i-m,j-n)$$

<u>Fourier Transform</u>

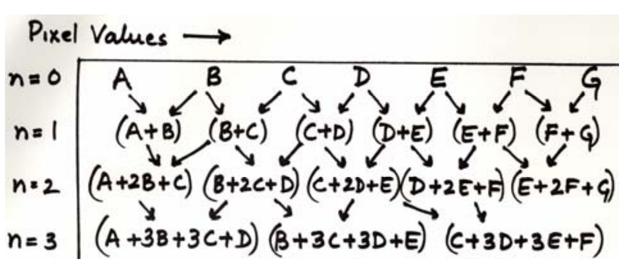
$$F(u,v)=\sum_{m=1}^{M}\sum_{n=1}^{N} f(m,n)e^{-i2\pi\left(\frac{mu}{M}+\frac{nv}{N}\right)}$$

<u>Inverse Fourier Transform</u>

$$f(k,l)=\frac{1}{MN}\sum_{u=1}^{M}\sum_{v=1}^{N} F(u,v)e^{i2\pi\left(\frac{ku}{M}+\frac{lv}{N}\right)}$$

Averaging

Let's think about averaging pixel values



For *n*=2, convolve pixel values with $\begin{bmatrix}1 & 2 & 1\end{bmatrix}$

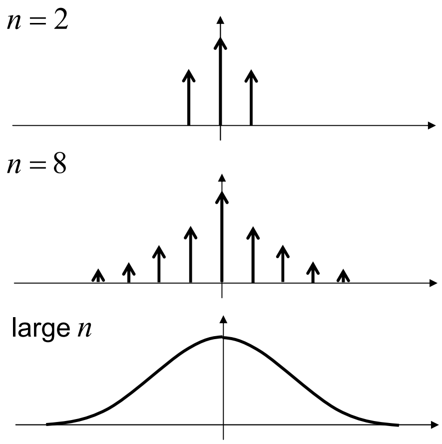
2D images:

(a) use $\begin{bmatrix}1 & 2 & 1\end{bmatrix}$ then $\begin{bmatrix}1\\2\\1\end{bmatrix}$ or (b) use $\begin{bmatrix}1 & 2 & 1\end{bmatrix} * \begin{bmatrix}1\\2\\1\end{bmatrix} = \begin{bmatrix}1 & 2 & 1\\2 & 4 & 2\\1 & 2 & 1\end{bmatrix}$

Which is faster? $(a)\,O(2(n+1))$ $(b)\,O\left((n+1)^2\right)$

Averaging

The convolution kernel

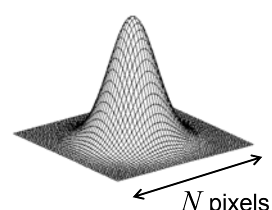


$\boxed{\text{Repeated averaging} \approx \text{Gaussian smoothing}}$

Gaussian Smoothing

Gaussian kernel

$$h(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\left(\frac{i^2+j^2}{\sigma^2}\right)}$$



Filter size $N \propto \sigma$ …can be very large (truncate, if necessary)

$$g(i,j) = \frac{1}{2\pi\sigma^2} \sum_{m=1} \sum_{n=1} e^{-\frac{1}{2}\left(\frac{m^2+n^2}{\sigma^2}\right)} f(i-m, j-n)$$

2D Gaussian is separable!

$$g(i,j) = \frac{1}{2\pi\sigma^2} \sum_{m=1} e^{-\frac{1}{2}\frac{m^2}{\sigma^2}} \sum_{n=1} e^{-\frac{1}{2}\frac{n^2}{\sigma^2}} f(i-m, j-n)$$

$\boxed{\text{Use two 1D Gaussian filters}}$

Gaussian Smoothing

- A Gaussian kernel gives less weight to pixels further from the center of the window

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

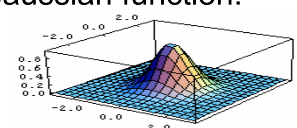
$F[x, y]$

$$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$H[u, v]$

- This kernel is an approximation of a Gaussian function:

$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



Gaussian Smoothing

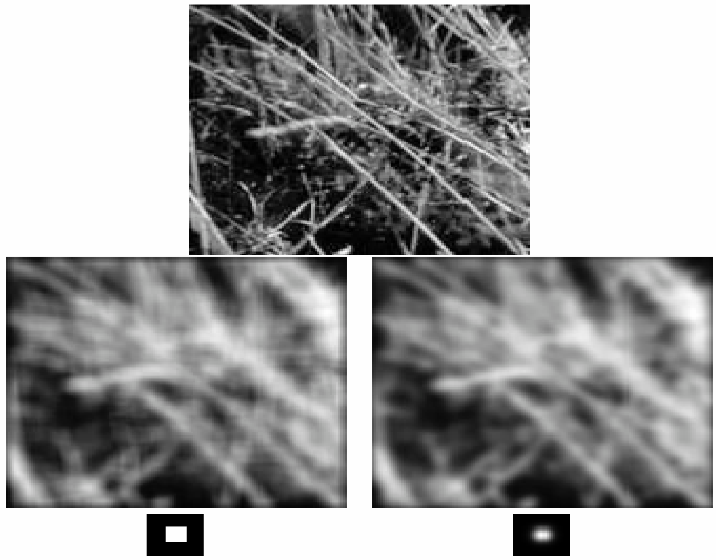
original

$\sigma = 2$

$\sigma = 2.8$

$\sigma = 4$

Mean vs. Gaussian filtering



Gaussian Smoothing

by Charles Allen Gillbert

by Harmon & Julesz

http://www.michaelbach.de/ot/cog_blureffects/index.html

Gaussian Smoothing


http://www.michaelbach.de/ot/cog_blureffects/index.html

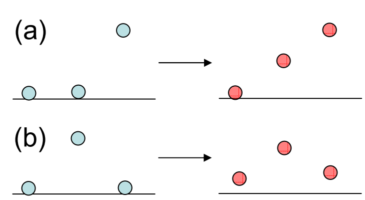
Border Problem

Border Problem

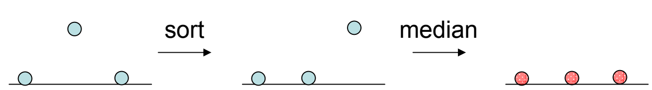
- Ignore
  - Output image will be smaller than original
- Pad with constant values
  - Can introduce substantial 1st order derivative values
- Pad with reflection
  - Can introduce substantial 2nd order derivative values

Median Filter

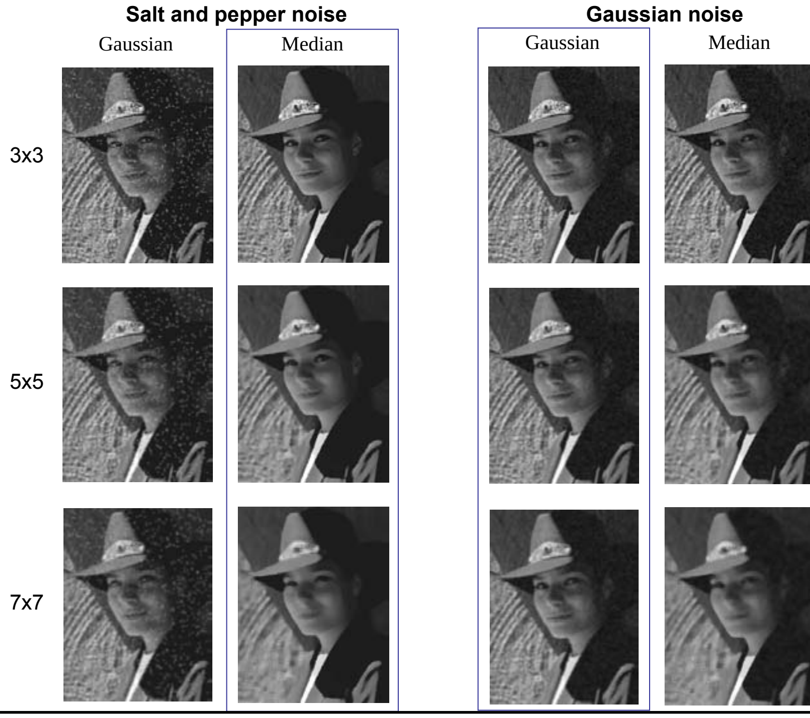
- Smoothing is averaging
  - (a) Blurs edges
  - (b) Sensitive to outliers



- Median filtering
  - Sort $N^2 - 1$ values around the pixel
  - Select middle value (median)



  - Non-linear (Cannot be implemented with convolution)



Correlation

template

How do we locate the template in the image?

- Minimize

$$E(i,j) = \sum_m \sum_n \left[f(m,n) - t(m-i,n-j) \right]^2$$
$$= \sum_m \sum_n \left[f^2(m,n) + t^2(m-i,n-j) - 2f(m,n)t(m-i,n-j) \right]^2$$

- Maximize

$$R_{tf}(i,j) = \sum_m \sum_n t(m-i,n-j) f(m,n)$$

Cross-correlation

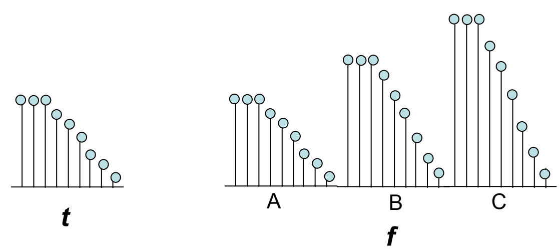
Cross-correlation

$$R_{tf}(i,j)=\sum_{m}\sum_{n} t(m-i,n-j)f(m,n) \qquad R_{tf}=t\otimes f$$

Note: $t\otimes f \neq f\otimes t$

$$R_{ff}=f\otimes f$$ Auto-correlation

Problem:



$R_{tf}(C)>R_{tf}(B)>R_{tf}(A)$ We need $R_{tf}(A)$ to be the maximum!

Normalized Correlation

- Account for energy differences

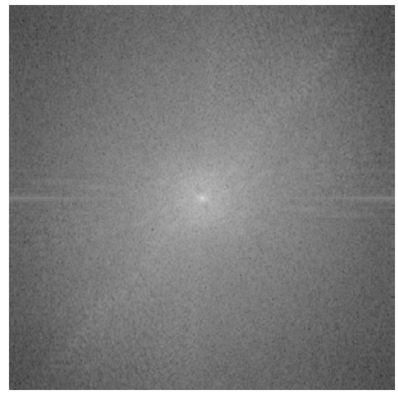
$$N_{tf}(i,j)=\frac{\sum_{m}\sum_{n} t(m-i,n-j)f(m,n)}{\left[\sum_{m}\sum_{n} t^2(m-i,n-i)\right]^{1/2}\left[\sum_{m}\sum_{n} f^2(m,n)\right]^{1/2}}$$

$\otimes$

$=$

Image Processing in the Fourier Domain

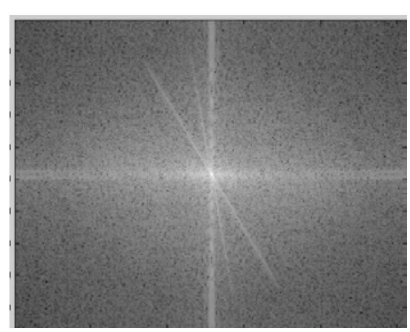
Magnitude of the FT



Does not look anything like what we have seen

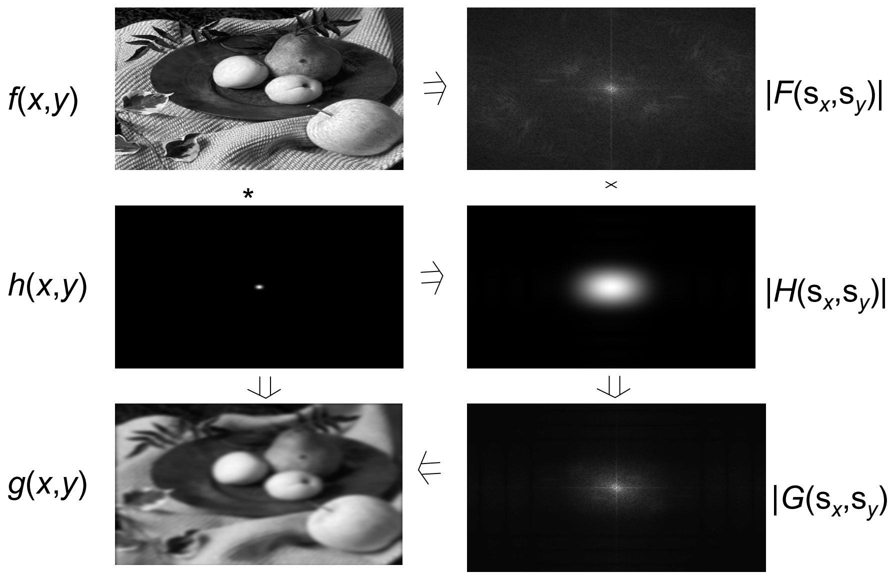
Image Processing in the Fourier Domain

Magnitude of the FT

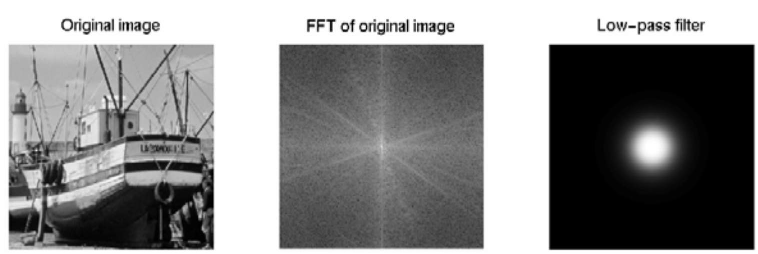


Does not look anything like what we have seen

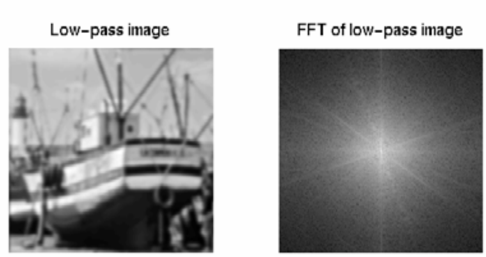
Convolution is Multiplication in Fourier Domain



Low-pass Filtering

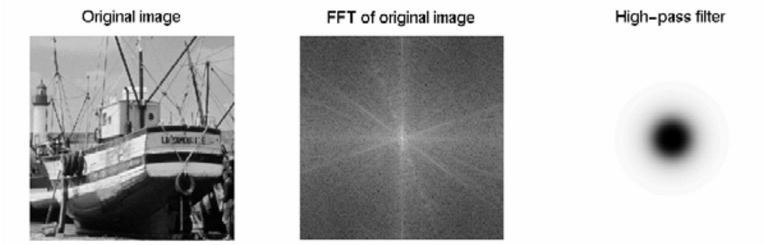


Let the low frequencies pass and eliminating the high frequencies.

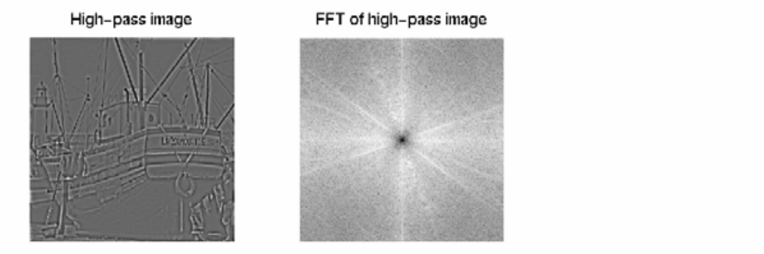


Generates image with overall shading, but not much detail

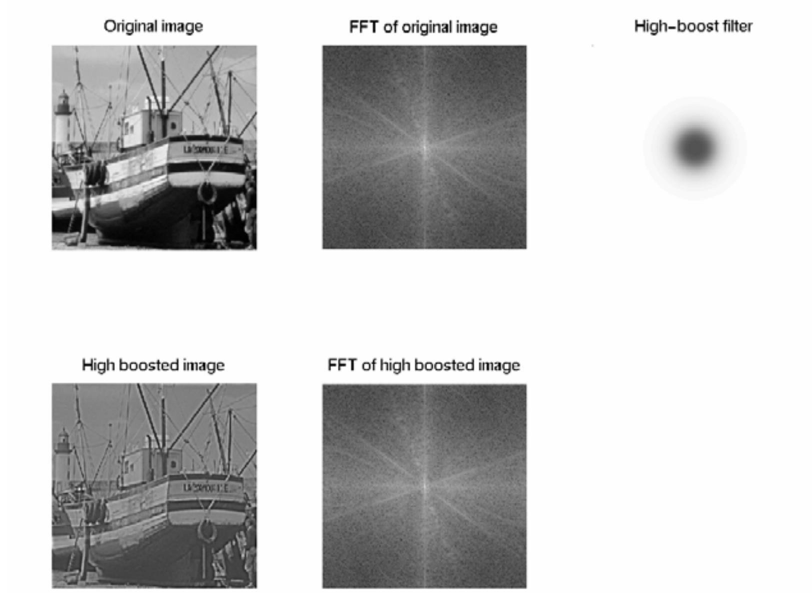
High-pass Filtering



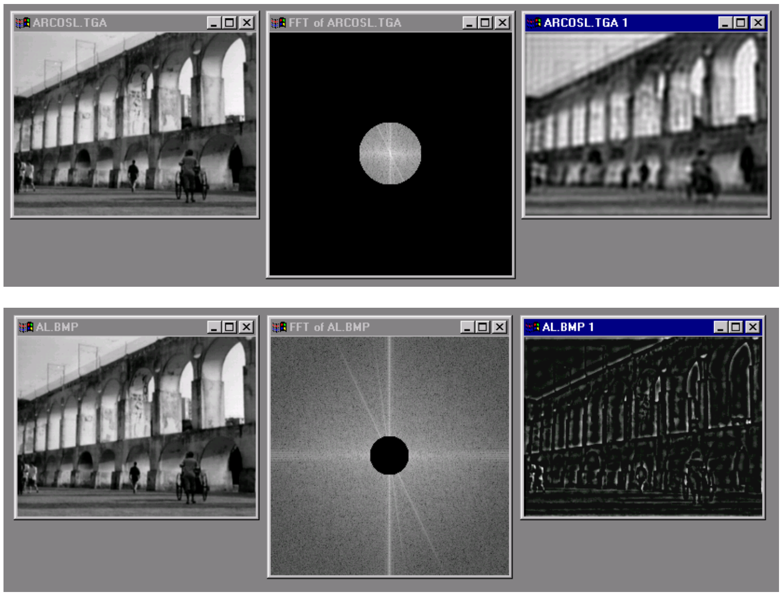
Lets through the high frequencies (the detail), but eliminates the low frequencies (the overall shape). It acts like an edge enhancer.



Boosting High Frequencies



Most information at low frequencies!



Fun with Fourier Spectra

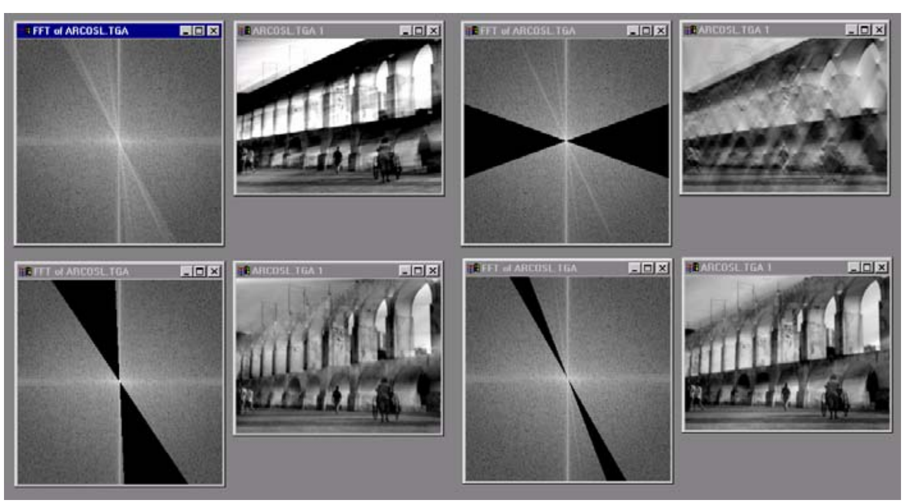
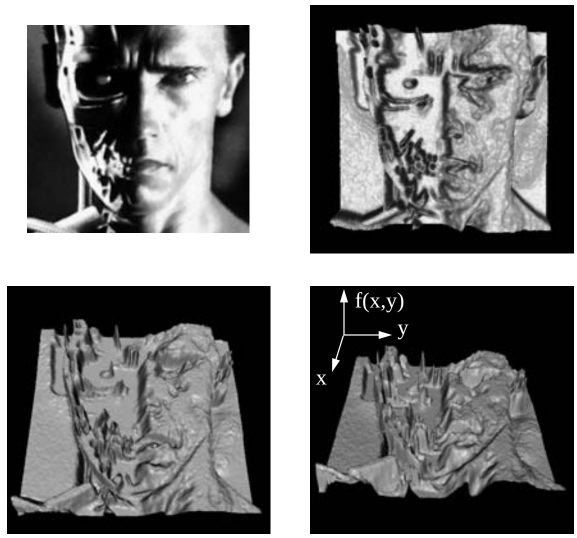


Image as a Discrete Function



Digital Images

The scene is

- **projected** on a 2D plane,
- **sampled** on a regular grid, and each sample is
- **quantized** (rounded to the nearest integer)

$$f(i,j) = \text{Quantize}\{f(i\Delta, j\Delta)\}$$

Image as a matrix

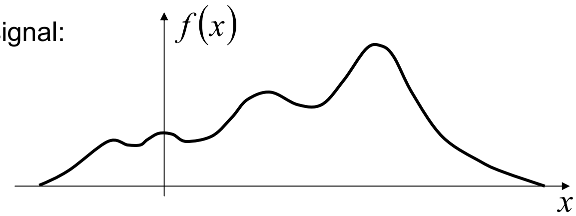
$j \rightarrow$

$i \downarrow$

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

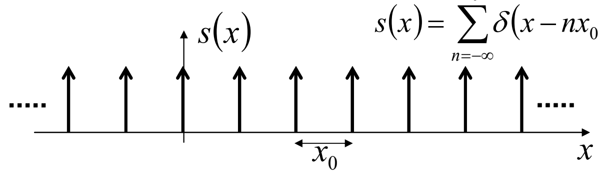
Sampling Theorem

Continuous signal:



Shah function (Impulse train):

$$s(x)=\sum_{n=-\infty}^{\infty}\delta(x-nx_0)$$



Sampled function:

$$f_s(x)=f(x)s(x)=f(x)\sum_{n=-\infty}^{\infty}\delta(x-nx_0)$$

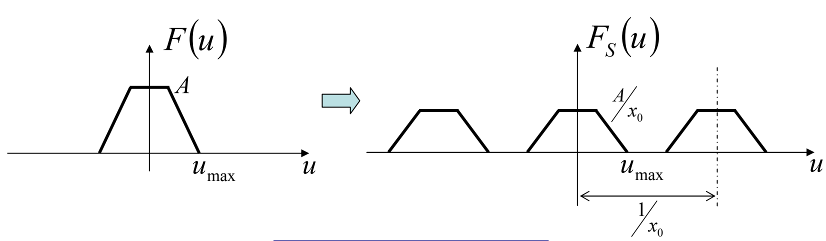
Sampling Theorem

Sampled function:

$$f_s(x)=f(x)s(x)=f(x)\sum_{n=-\infty}^{\infty}\delta(x-nx_0)$$

Sampling frequency $\frac{1}{x_0}$

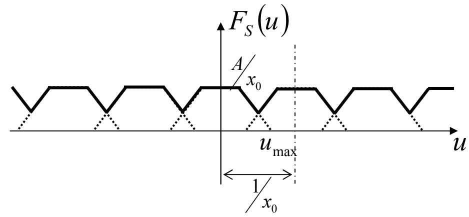
$$F_S(u)=F(u)*S(u)=F(u)*\frac{1}{x_0}\sum_{n=-\infty}^{\infty}\delta\left(u-\frac{n}{x_0}\right)$$



Only if $u_{max}\le\frac{1}{2x_0}$

Nyquist Theorem

If $u_{\max} > \dfrac{1}{2x_0}$



Aliasing

When can we recover $F(u)$ from $F_S(u)$?

Only if $u_{\max} \le \dfrac{1}{2x_0}$ (Nyquist Frequency)

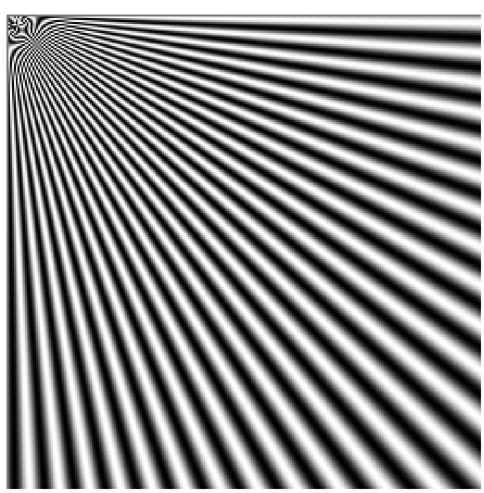
We can use

$$C(u) = \begin{cases} x_0 & |u| < 1/2x_0 \\ 0 & \text{otherwise} \end{cases}$$

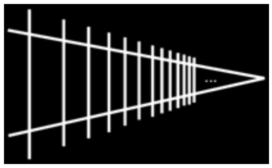
Then $F(u) = F_S(u)C(u)$ and $f(x) = \mathrm{IFT}[F(u)]$

Sampling frequency must be greater than $2u_{\max}$

Aliasing



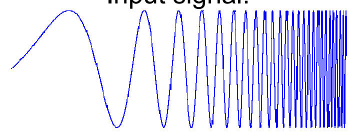
Alias: n., an assumed name



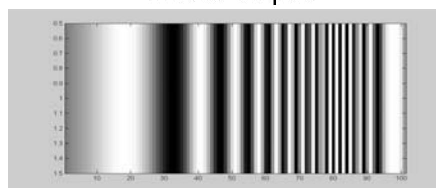
Picket fence receding into the distance will produce aliasing…

WHY?

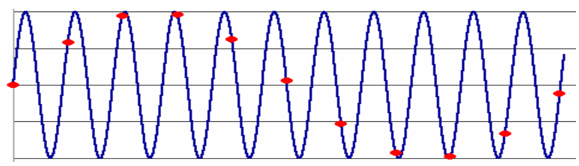
Input signal:



Matlab output:



x = 0:.05:5; imagesc(sin((2.^x).*x))



Alias!

Not enough samples

Image Scaling

This image is too big to fit on the screen. How can we reduce it?

How to generate a half-sized version?

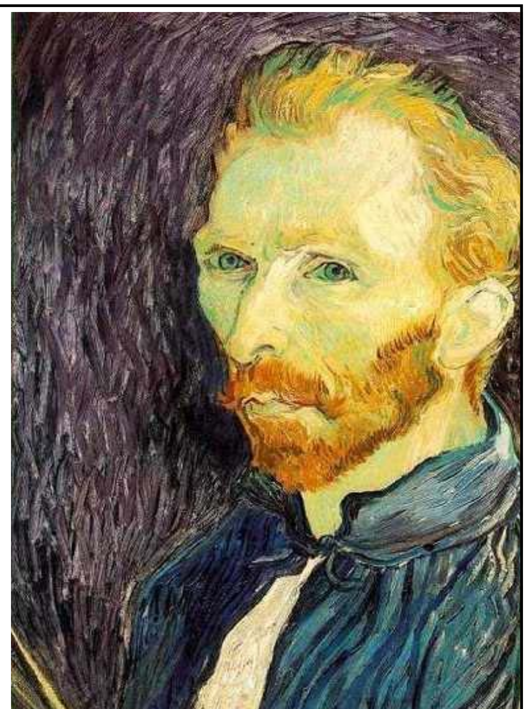
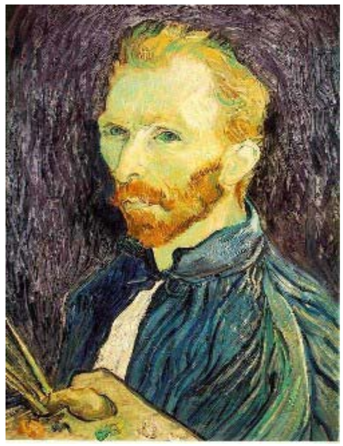


Image Sub-Sampling

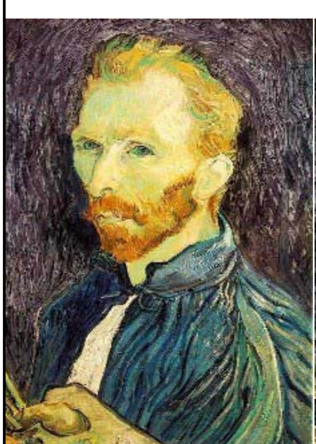


1/4

1/8

Throw away every other row and column to create a 1/2 size image
- called *image sub-sampling*

Image Sub-Sampling

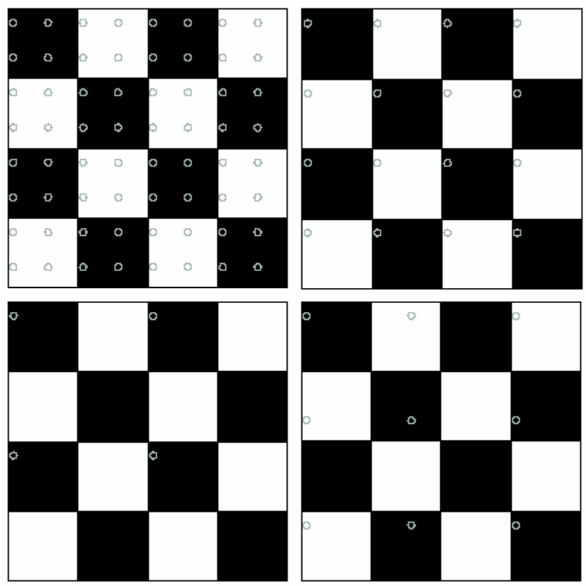


1/2

1/4 (2x zoom)

1/8 (4x zoom)

Good and Bad Sampling



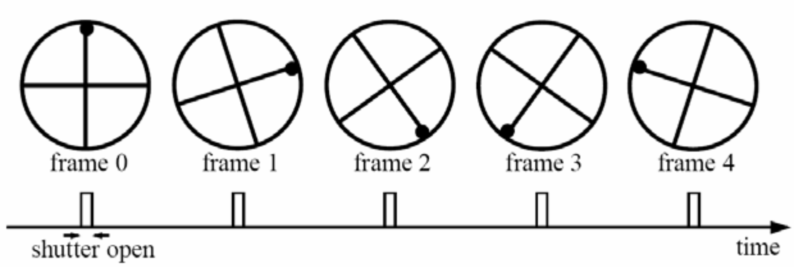
Good sampling:
•Sample often or,
•Sample wisely

Bad sampling:
•see aliasing in action!

Really bad in video

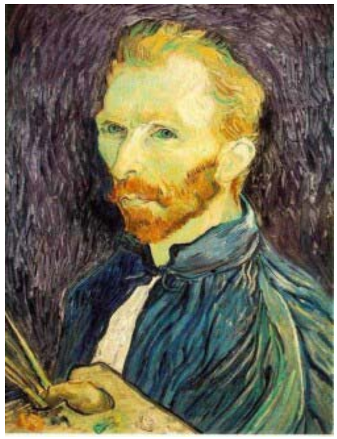
Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what's happening.

If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):



Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)

Sub-Sampling with Gaussian Pre-Filtering



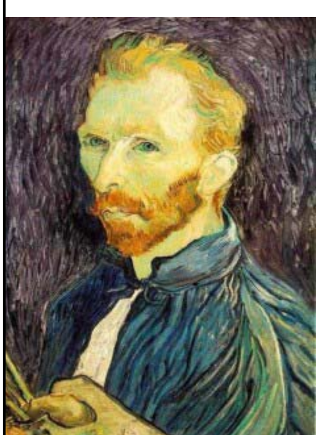
Gaussian 1/2

G 1/4

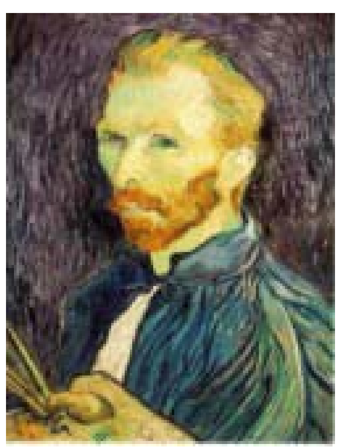
G 1/8

- Solution: filter the image, *then* subsample
  - Filter size should double for each ½ size reduction. Why?

Sub-Sampling with Gaussian Pre-Filtering



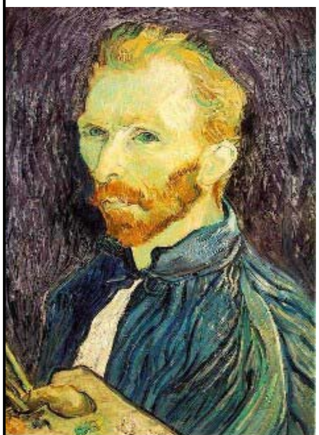
Gaussian 1/2



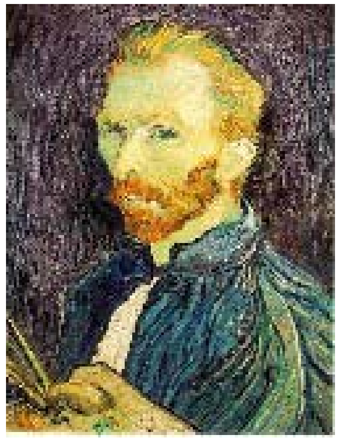
G 1/4

G 1/8

Compare with...



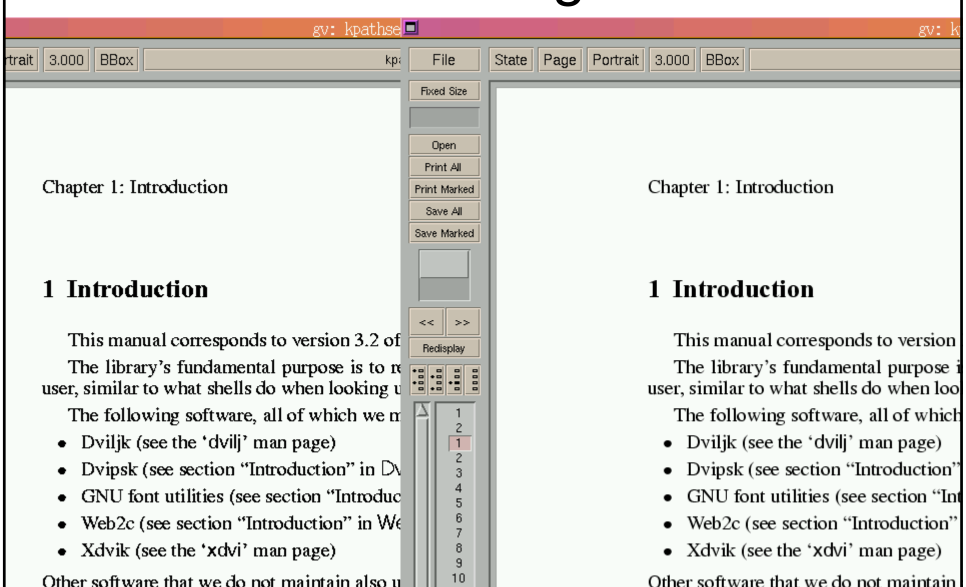
1/2



1/4 (2x zoom)

1/8 (4x zoom)

Aliasing



Canon D60 (w/ anti-alias filter)

Sigma SD9 (w/o anti-alias filter)

From Rick Matthews website, images by Dave Etchells

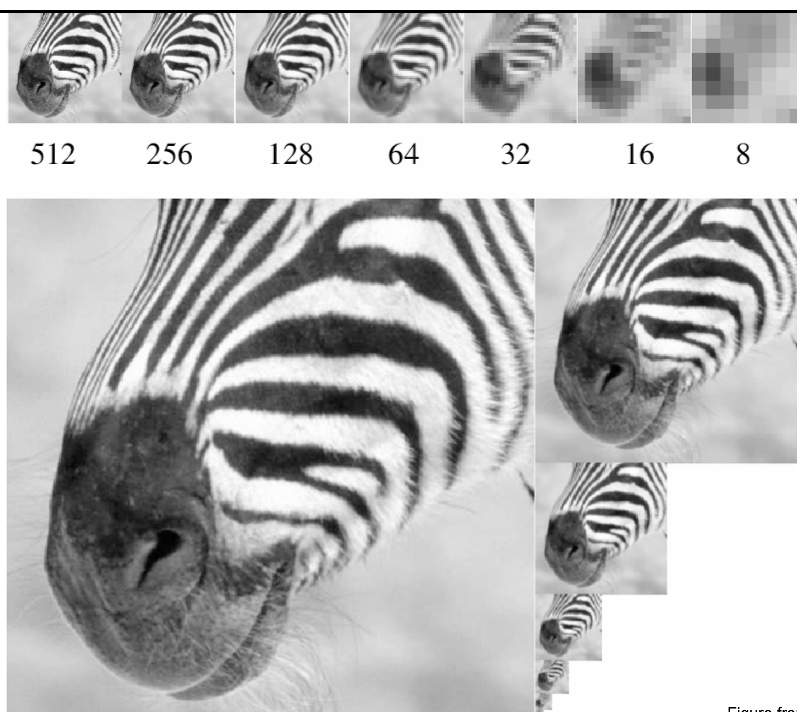


Figure from David Forsyth

Next Class

- Image Processing and Filtering (continued) – Edge Detection
- Horn, Chapter 6