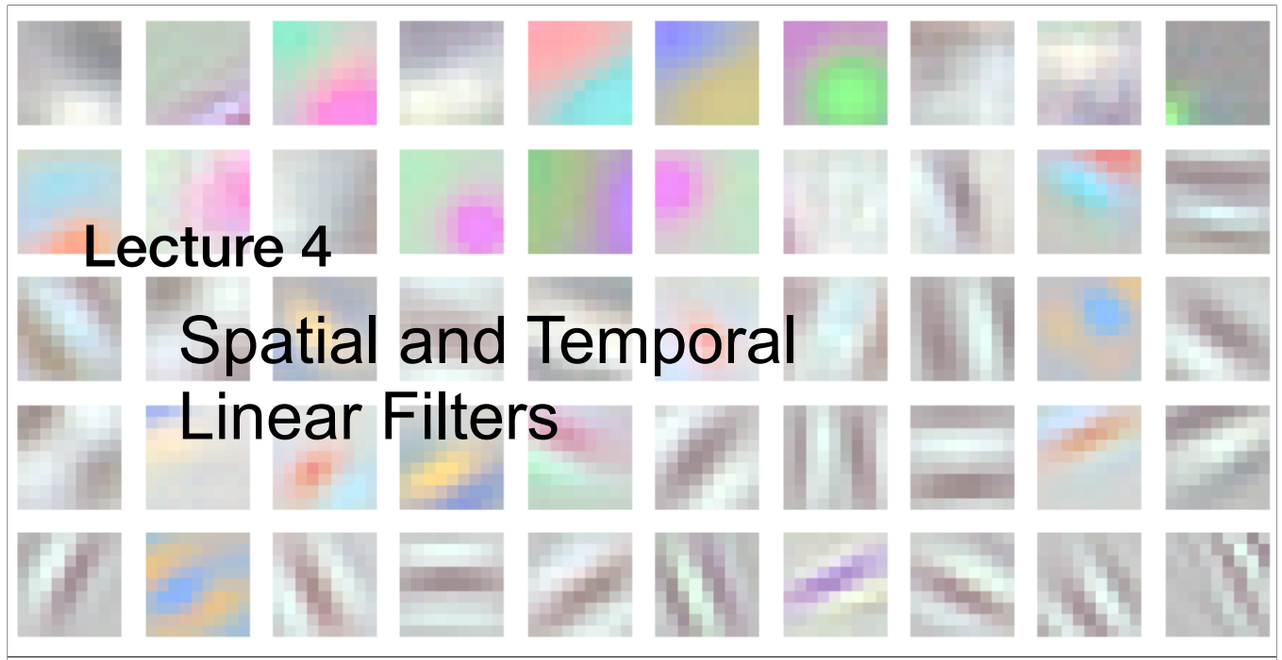


Lecture 4

Spatial and Temporal Linear Filters

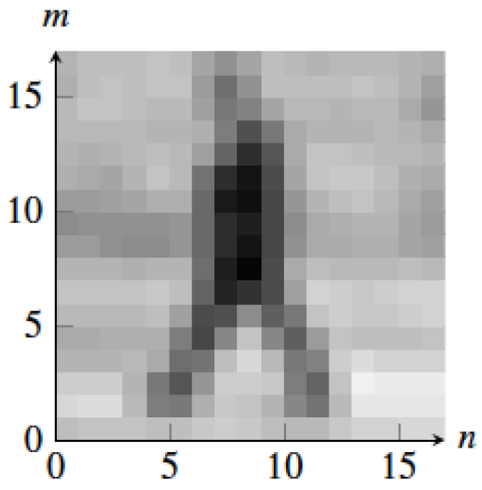
Sara Beery, Kaiming He, Vincent Sitzmann, Mina Konaković Luković

Announcements

- Pset 1 is out (due next Thu)
- Lecture recordings, lecture slides, course notes available at https://advances-in-vision.github.io/schedule.html
- Register for Piazza and Canvas (participation on Piazza rewarded)
- Max 3 person team for final project
- Check new tutorial rooms on webpage/Canvas

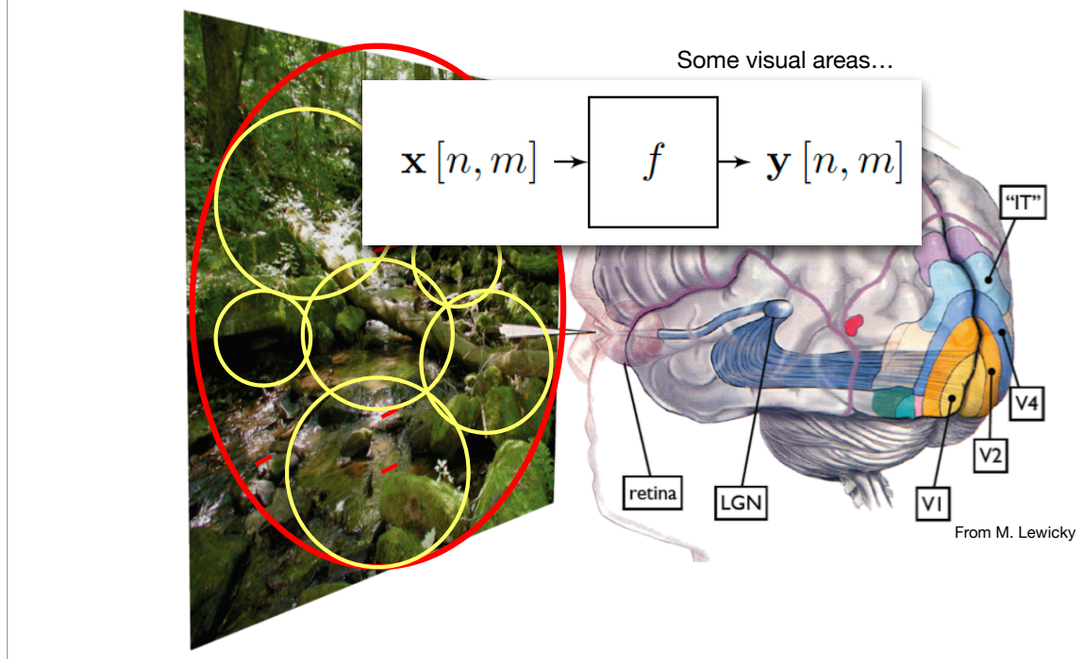
Remember, an image is just an array of numbers

What we see



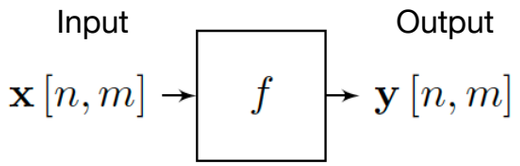
What the machine gets

$$\mathbf{I} = \begin{bmatrix}
160 & 175 & 171 & 168 & 168 & 172 & 164 & 158 & 167 & 173 & 167 & 163 & 162 & 164 & 160 & 159 & 163 & 162 \\
149 & 164 & 172 & 175 & 178 & 179 & 176 & 118 & 97 & 168 & 175 & 171 & 169 & 175 & 176 & 177 & 165 & 152 \\
161 & 166 & 182 & 171 & 170 & 177 & 175 & 116 & 109 & 169 & 177 & 173 & 168 & 175 & 175 & 159 & 153 & 123 \\
171 & 174 & 177 & 175 & 167 & 161 & 157 & 138 & 103 & 112 & 157 & 164 & 159 & 160 & 165 & 169 & 148 & 144 \\
163 & 163 & 162 & 165 & 167 & 164 & 178 & 167 & 77 & 55 & 134 & 170 & 167 & 162 & 164 & 175 & 168 & 160 \\
173 & 164 & 158 & 165 & 180 & 180 & 150 & 89 & 61 & 34 & 137 & 186 & 186 & 182 & 175 & 165 & 160 & 164 \\
152 & 155 & 146 & 147 & 169 & 180 & 163 & 51 & 24 & 32 & 119 & 163 & 175 & 182 & 181 & 162 & 148 & 153 \\
134 & 135 & 147 & 149 & 150 & 147 & 148 & 62 & 36 & 46 & 114 & 157 & 163 & 167 & 169 & 163 & 146 & 147 \\
135 & 132 & 131 & 125 & 115 & 129 & 132 & 74 & 54 & 41 & 104 & 156 & 152 & 156 & 164 & 156 & 141 & 144 \\
151 & 155 & 151 & 145 & 144 & 149 & 143 & 71 & 31 & 29 & 129 & 164 & 157 & 155 & 159 & 158 & 156 & 148 \\
172 & 174 & 178 & 177 & 177 & 181 & 174 & 54 & 21 & 29 & 136 & 190 & 180 & 179 & 176 & 184 & 187 & 182 \\
177 & 178 & 176 & 173 & 174 & 180 & 150 & 27 & 101 & 94 & 74 & 189 & 188 & 186 & 183 & 186 & 188 & 187 \\
160 & 160 & 163 & 163 & 161 & 167 & 100 & 45 & 169 & 166 & 59 & 136 & 184 & 176 & 175 & 177 & 185 & 186 \\
147 & 150 & 153 & 155 & 160 & 155 & 56 & 111 & 182 & 180 & 104 & 84 & 168 & 172 & 171 & 164 & 168 & 167 \\
184 & 182 & 178 & 175 & 179 & 133 & 86 & 191 & 201 & 204 & 191 & 79 & 172 & 220 & 217 & 205 & 209 & 200 \\
184 & 187 & 192 & 182 & 124 & 32 & 109 & 168 & 171 & 167 & 163 & 51 & 105 & 203 & 209 & 203 & 210 & 205 \\
191 & 198 & 203 & 197 & 175 & 149 & 169 & 189 & 190 & 173 & 160 & 145 & 156 & 202 & 199 & 201 & 205 & 202 \\
153 & 149 & 153 & 155 & 173 & 182 & 179 & 177 & 182 & 177 & 182 & 185 & 179 & 177 & 167 & 176 & 182 & 180
\end{bmatrix}$$



Slides from lewicki

Signals and systems

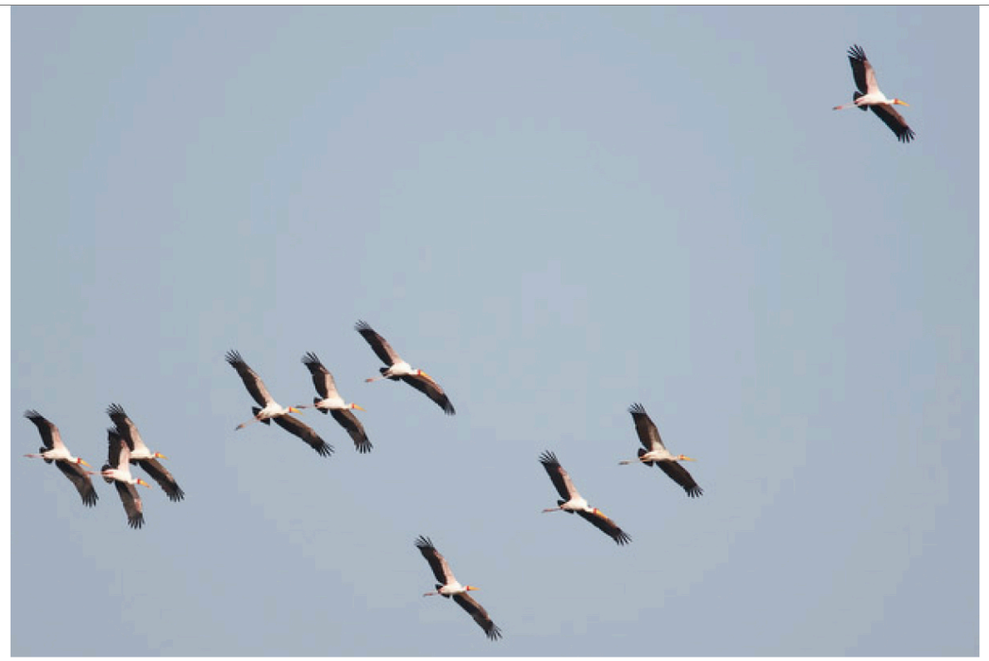


One important class of systems is the set of linear systems.

A function f is linear if it satisfies:

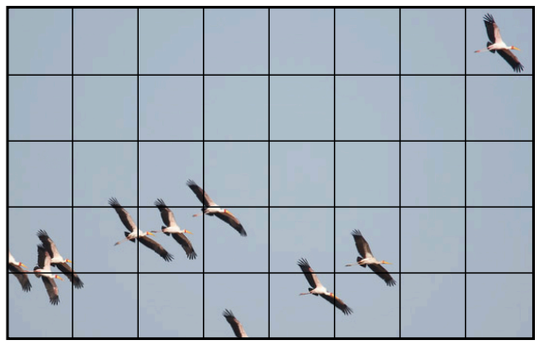
$$f(\alpha \mathbf{x}) = \alpha f(\mathbf{x})$$

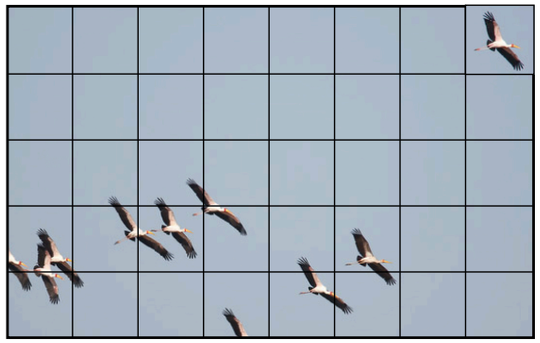
$$f(\mathbf{x} + \mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$$

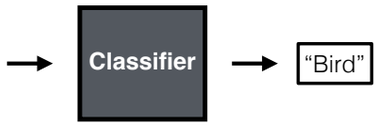


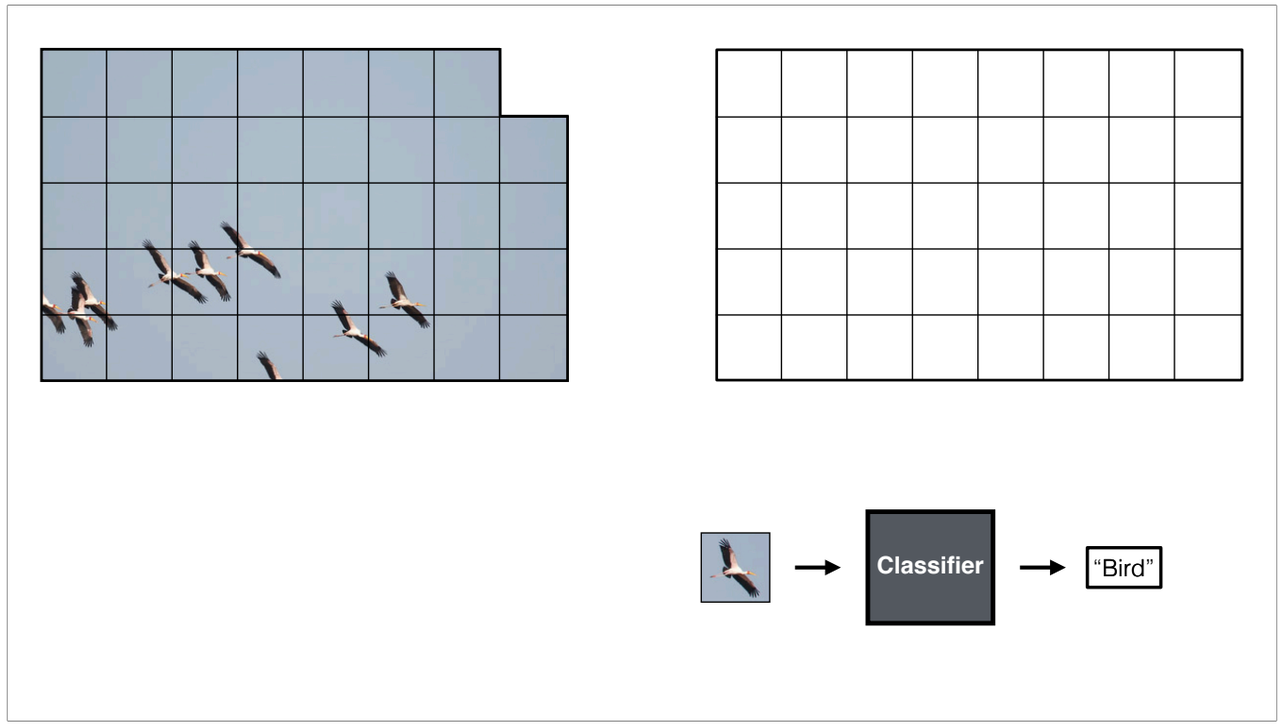
We need translation invariance

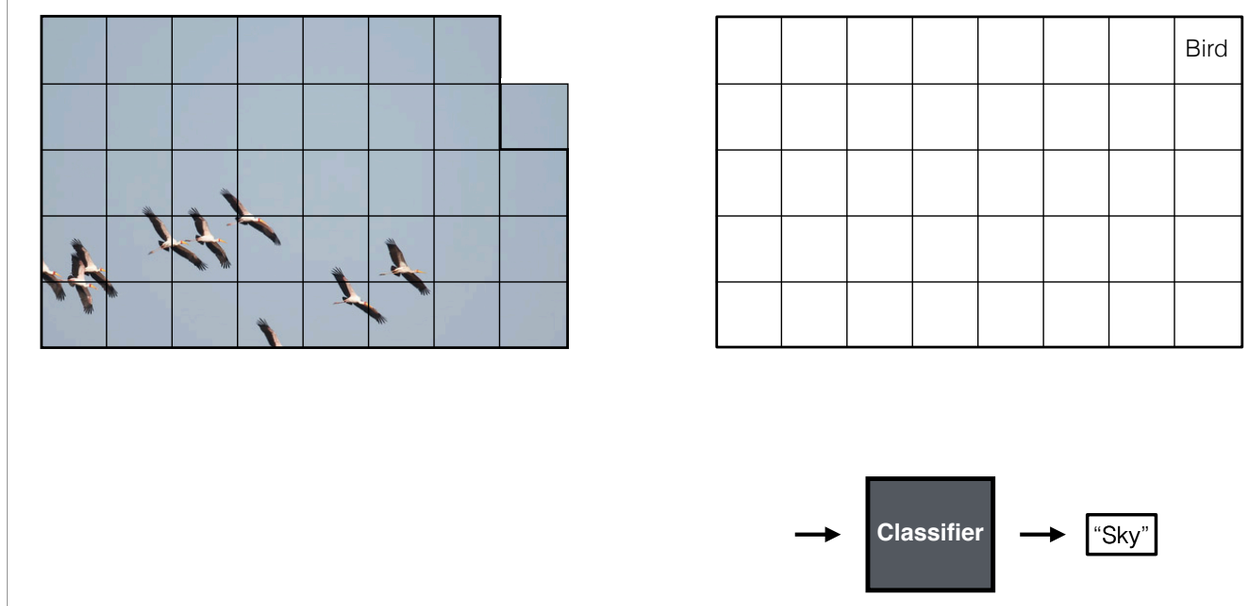
Now we also want translation invariant operations. We can have linear translation invariant and non-linear translation invariant. We will focus on linear translation invariant

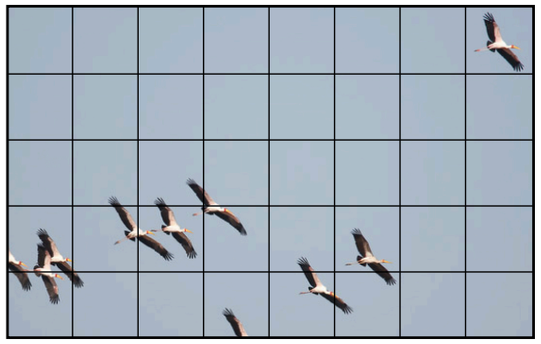





Classifier
“Bird”


Classifier
“Bird”

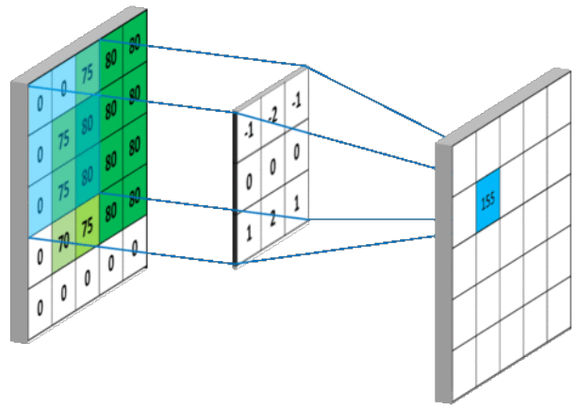

Bird
Classifier
“Sky”



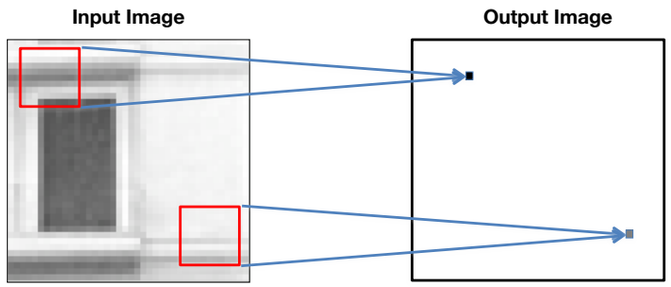
f

Sky	Sky	Sky	Sky	Sky	Sky	Sky	Bird
Sky	Sky	Sky	Sky	Sky	Sky	Sky	Sky
Sky	Sky	Sky	Sky	Sky	Sky	Sky	Sky
Bird	Bird	Bird	Sky	Bird	Sky	Sky	Sky
Sky	Sky	Sky	Bird	Sky	Sky	Sky	Sky

Convolution

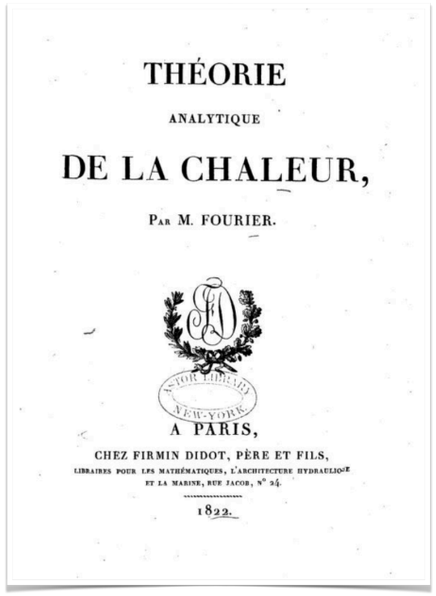


Convolution



The same weighting occurs within each window

Fourier series


THÉORIE

ANALYTIQUE

DE LA CHALEUR,

PAR M. FOURIER.

A PARIS,

CHEZ FIRMIN DIDOT, PÈRE ET FILS,

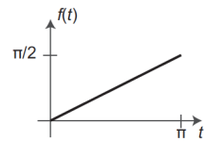
LIBRAIRES POUR LES MATHÉMATIQUES, L'ARCHITECTURE HYDRAULIQUE ET LA MARINE, RUE JACOB, N° 24.

1822.

Any function *f*(*t*) with *t* in the interval (0,π) could be expressed as an infinite linear combination of harmonically related sinusoids:

$$f(t) = a_1 \sin(t) + a_2 \sin(2t) + a_3 \sin(3t) + \ldots \quad \text{with} \quad a_n = \frac{2}{\pi}\int_0^{\pi} f(t)\sin(nt)dt$$

One of Fourier's original examples of sine series is the expansion of the ramp signal:



$$\frac{1}{2}t = \sin(t) - \frac{1}{2}\sin(2t) + \frac{1}{3}\sin(3t) - \frac{1}{4}\sin(4t) + \ldots$$

The Discrete Fourier Transform

Discrete Fourier Transform (DFT) transforms a signal $f[n]$ into $F[u]$ as:

$$F[u] = \sum_{n=0}^{N-1} f[n] \exp\left(-2\pi j \frac{un}{N}\right)$$

The inverse of the DFT is:

$$f[n] = \frac{1}{N} \sum_{u=0}^{N-1} F[u] \exp\left(2\pi j \frac{un}{N}\right)$$

The signal $f[n]$ is a weighted linear combination of complex exponentials with weights $F[u]$

Visualizing the image Fourier transform

$$F[u,v] = \sum_{n=0}^{N-1}\sum_{m=0}^{M-1} f[n,m] \exp\left(-2\pi j\left(\frac{un}{N}+\frac{vm}{M}\right)\right)$$

The values of $F[u,v]$ are complex.

Using the real and imaginary components:

$$F[u,v] = Re\{F[u,v]\} + jImag\{F[u,v]\}$$

Or using a polar decomposition:

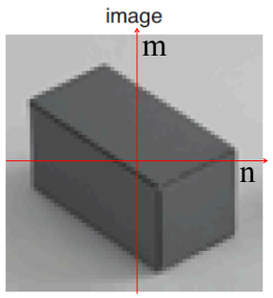
$$F[u,v] = A[u,v]\exp(j\theta[u,v])$$

Amplitude

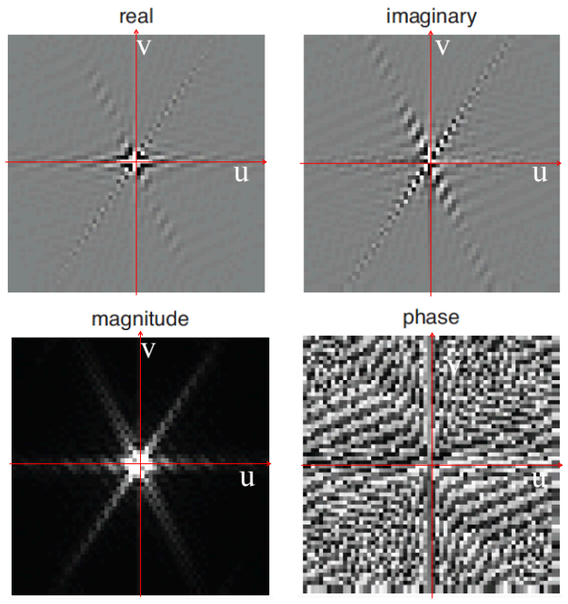
Phase

Visualizing the image Fourier transform

$f[n, m]$

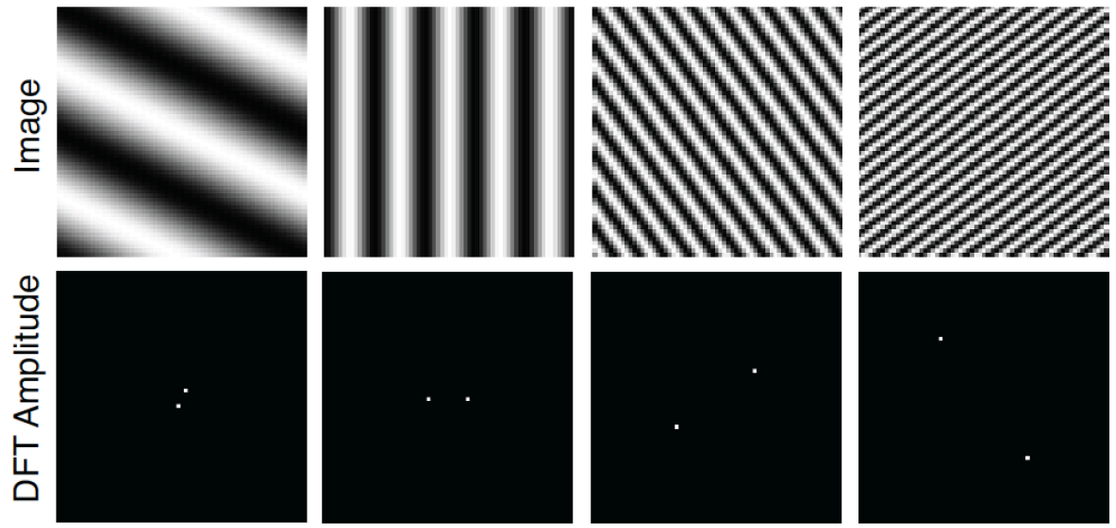


$F[u, v]$



Location information goes into the phase, strength goes into magnitude

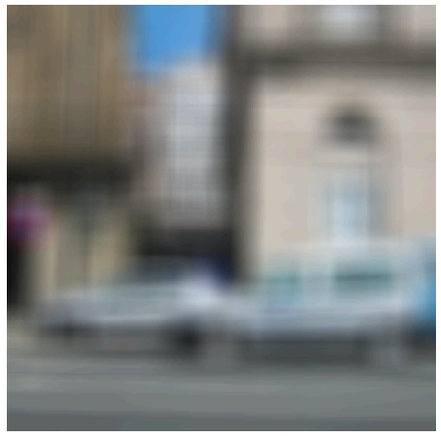
Simple Fourier transforms



Images are 64x64 pixels. The wave is a cosine, therefore DFT phase is zero.

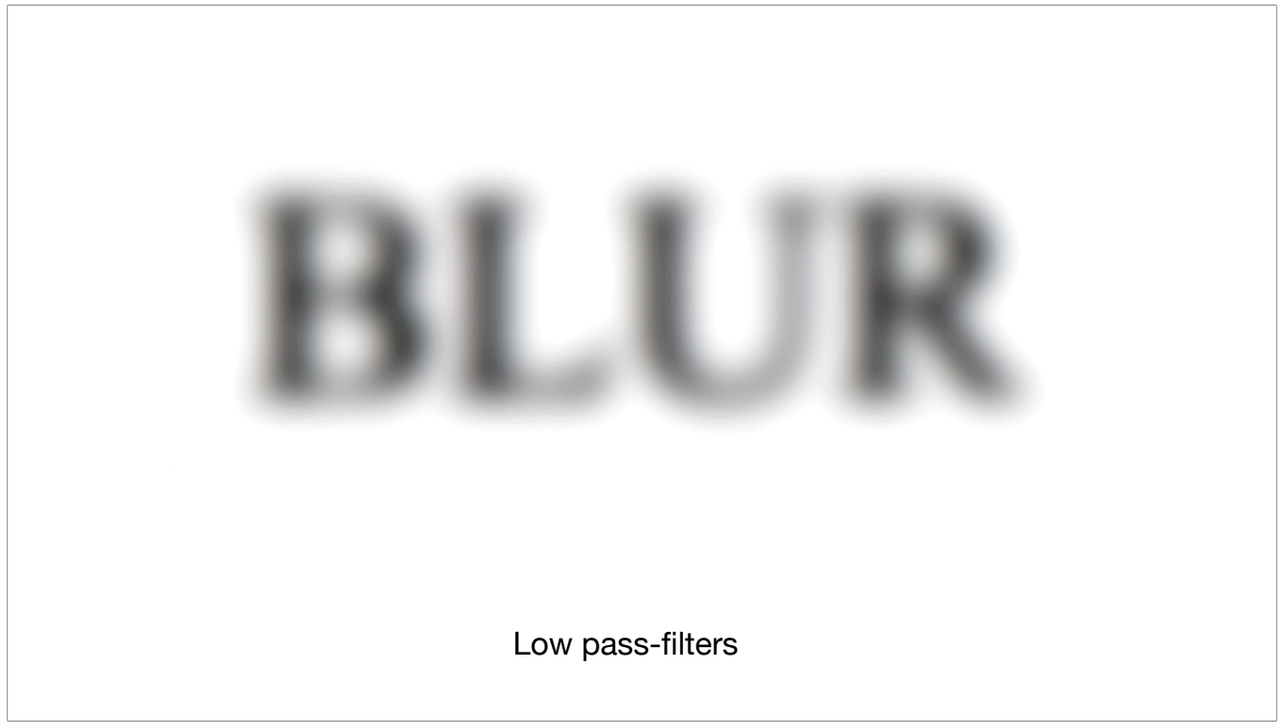
High frequency means further fro m the origin

Today: A collection of useful filters in space and time, and aliasing.



Low-pass filters

Band-pass filters



Low pass-filters

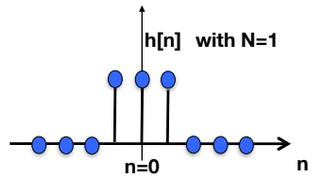
Box filter

$$h_{N,M}[n,m] = \begin{cases} 1 & \text{if } -N \le n \le N \text{ and } -M \le m \le M \\ 0 & \text{otherwise} \end{cases}$$

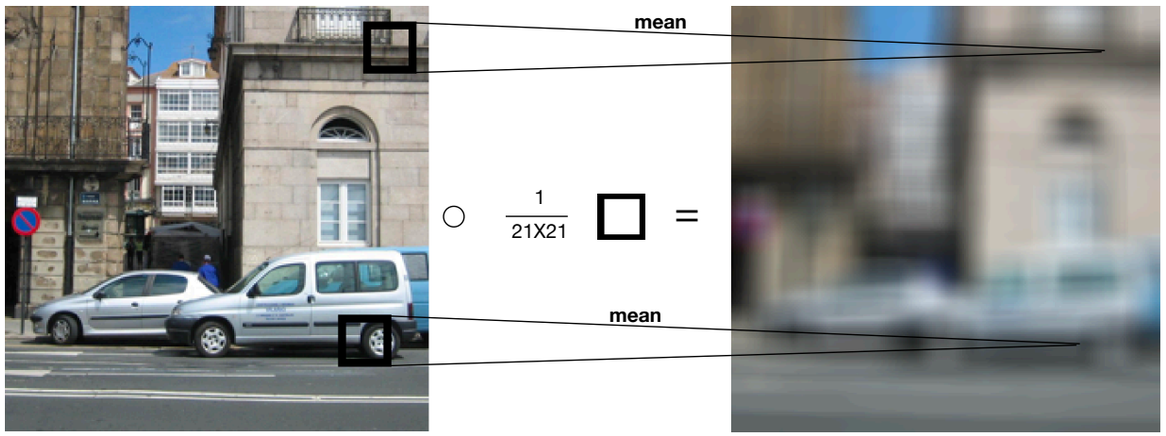
2N+1

1	1	…	1
1	1		1
1	1		1
…			
1	1	1	1

2M+1



Box filter

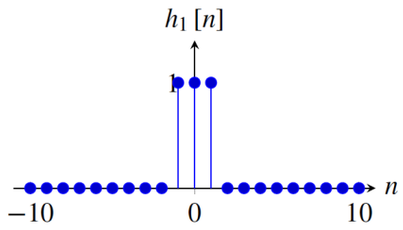


What does it do?

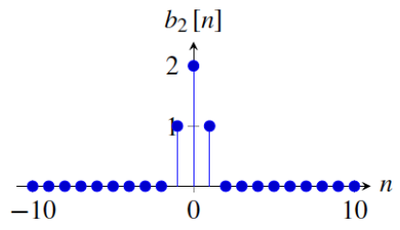
- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)

3-tap box filter versus b2[n]

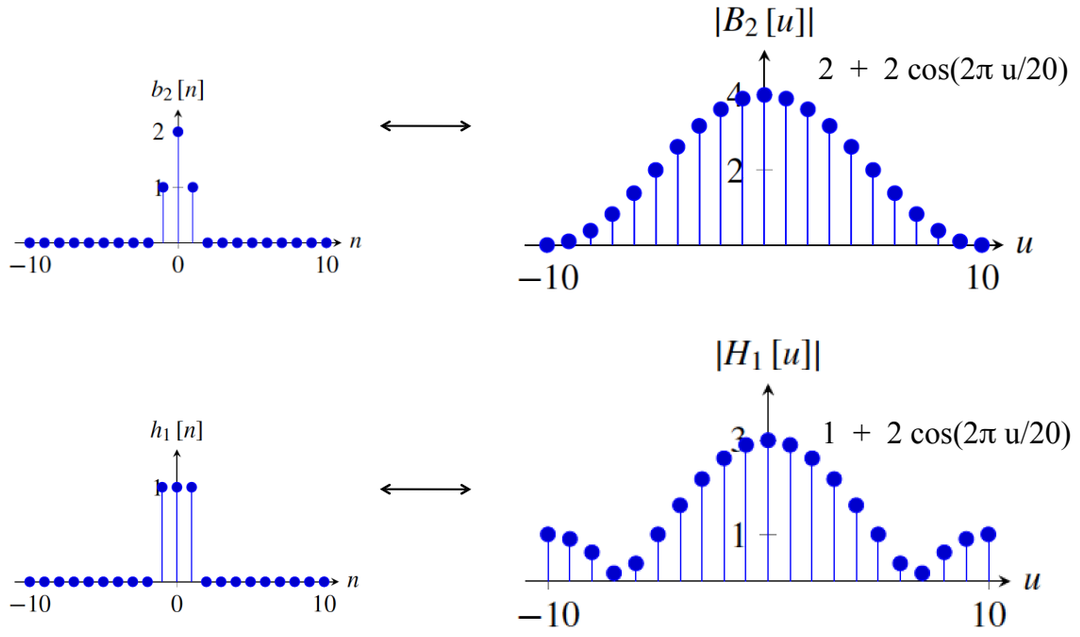
[1 1 1]



[1 2 1]



Which one is a better low-pass filter?


$|B_2[u]|$
$b_2[n]$
2
1
n
−10
0
10
2 + 2 cos(2π u/20)
4
2
u
−10
10
$|H_1[u]|$
$h_1[n]$
1
n
−10
0
10
3
1 + 2 cos(2π u/20)
1
u
−10
10

h1[n] vs b2[n]

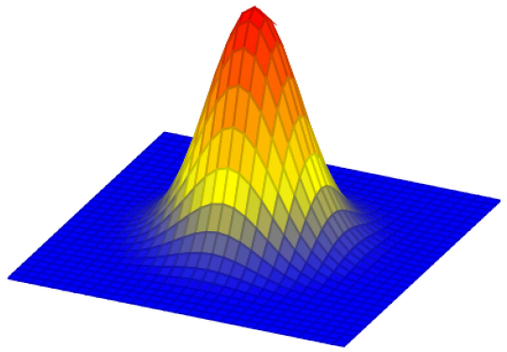
$[1, 1, 1] \circ [\ldots, 1, -1, 1, -1, 1, -1, \ldots] = [\ldots, -1, 1, -1, 1, -1, 1, \ldots]$

$[1, 2, 1] \circ [\ldots, 1, -1, 1, -1, 1, -1, \ldots] = [\ldots, 0, 0, 0, 0, 0, 0, \ldots]$

Gaussian filter

In the continuous domain:

$$g(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp -\frac{x^2 + y^2}{2\sigma^2}$$



Gaussian filter

Continuous Gaussian:

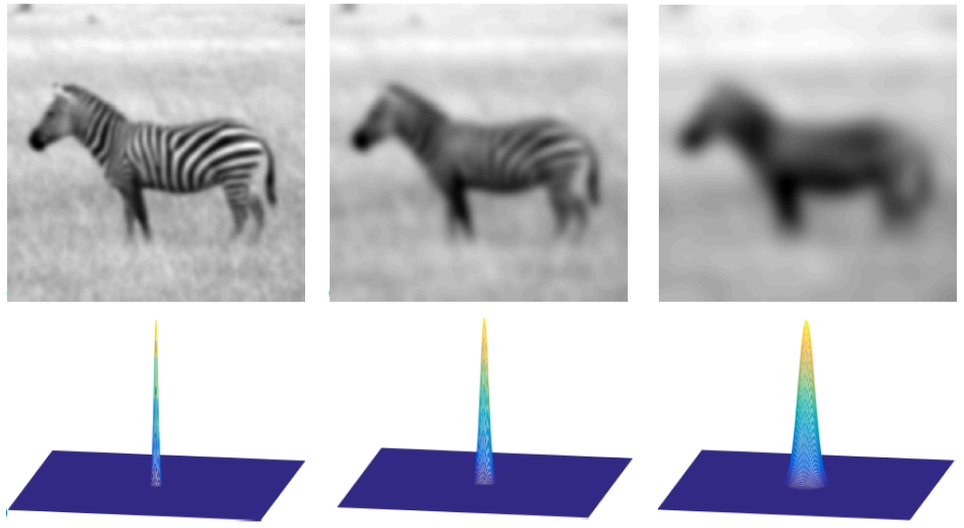
$$g(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp -\frac{x^2 + y^2}{2\sigma^2}$$

Discretization of the Gaussian:

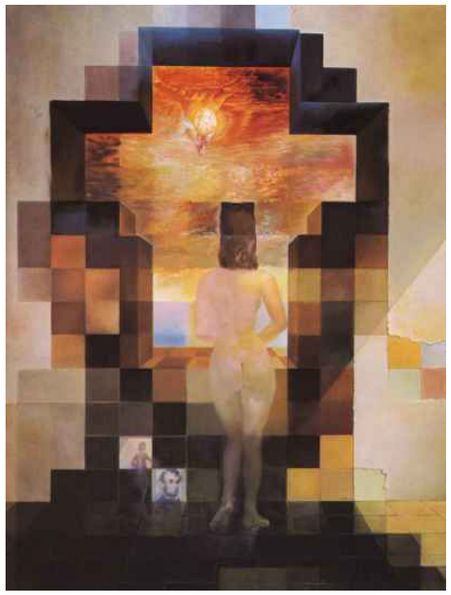
$$g[m, n; \sigma] = \exp -\frac{m^2 + n^2}{2\sigma^2}$$

Scale

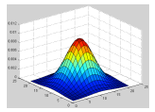
$$g[m,n;\sigma] = \exp -\frac{m^2+n^2}{2\sigma^2}$$

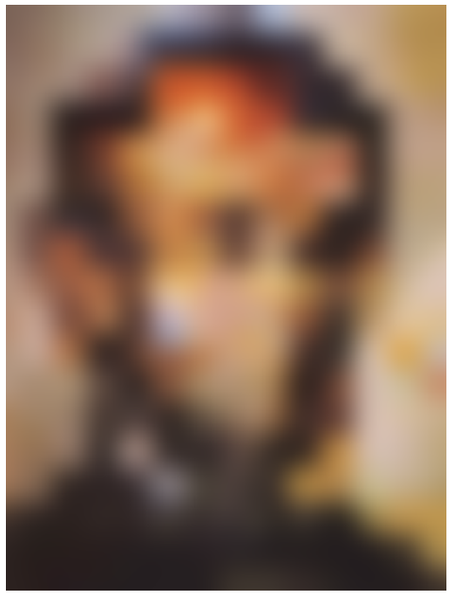


Gaussian filter for low-pass filtering



Dali





Properties of the Gaussian filter

$$g(x, y; \sigma) = \frac{1}{2\pi\sigma^2} \exp -\frac{x^2 + y^2}{2\sigma^2}$$

- The (continuous) Fourier transform of a Gaussian is another Gaussian

$$G(u, v; \sigma) = \exp -2\pi^2 (u^2 + v^2)\sigma^2$$

- The convolution of two n-dimensional Gaussians is an n-dimensional Gaussian.

$$g(x, y; \sigma_1) \circ g(x, y; \sigma_2) = g(x, y; \sigma_3)$$

where the variance of the result is the sum

$$\sigma_3^2 = \sigma_1^2 + \sigma_2^2$$

(it is easy to prove this using the FT of the Gaussian)

Binomial filter

- Binomial coefficients provide a compact approximation of the Gaussian coefficients using only **integers**
- The simplest blur filter (low pass) is [1 1]
- Binomial filters in the family of filters obtained as successive convolutions of [1 1]

Binomial filter

$$b_1 = [1 \;\; 1]$$

$$b_2 = [1\ 1] \circ [1\ 1] = [1\ 2\ 1]$$

$$b_3 = [1\ 1] \circ [1\ 1] \circ [1\ 1] = [1\ 3\ 3\ 1]$$

Binomial filter

b_1				1		1				$\sigma_1^2 = 1/4$
b_2				1	2	1				$\sigma_2^2 = 1/2$
b_3			1	3	3	1				$\sigma_3^2 = 3/4$
b_4		1	4	6	4	1				$\sigma_4^2 = 1$
b_5		1	5	10	10	5	1			$\sigma_5^2 = 5/4$
b_6	1	6	15	20	15	6	1			$\sigma_6^2 = 3/2$
b_7	1	7	21	35	35	21	7	1		$\sigma_7^2 = 7/4$
b_8	1	8	28	56	70	56	28	8	1	$\sigma_8^2 = 2$

Properties of binomial filters

- Sum of the values is 2^n
- The variance of b_n is $\sigma^2 = n/4$
- The convolution of two binomial filters is also a binomial filter

$b_n \circ b_m = b_{n+m}$ with a variance: $\sigma_n^2 + \sigma_m^2 = \sigma_{n+m}^2$

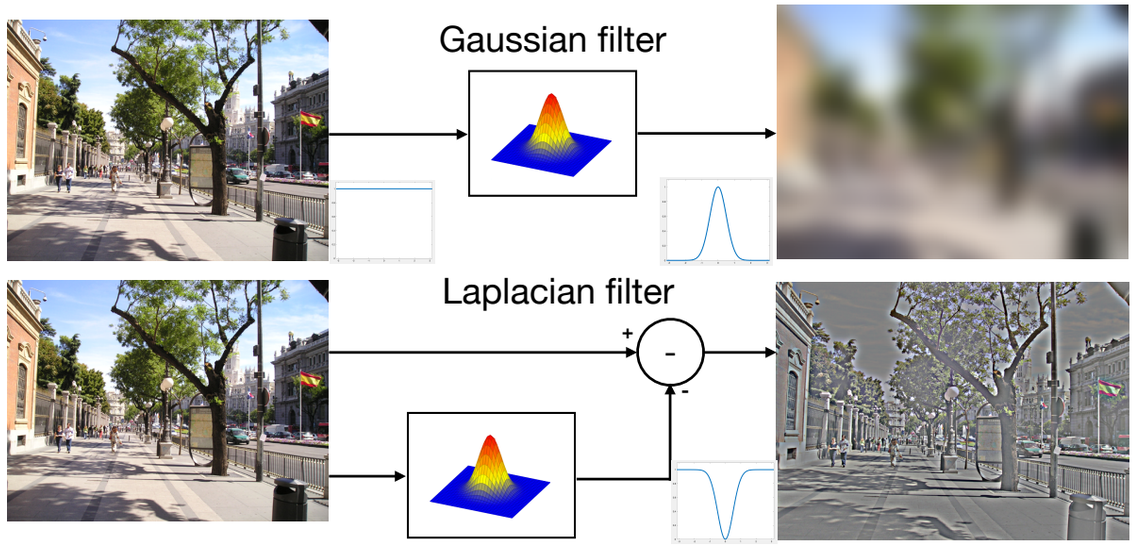
Note: These properties are analogous to the gaussian property in the continuous domain (but the binomial filter is different than a discretization of a gaussian)

B2[n]

$$b_{2,2} = b_{2,0} \circ b_{0,2} = \begin{bmatrix} 1 & 2 & 1 \end{bmatrix} \circ \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

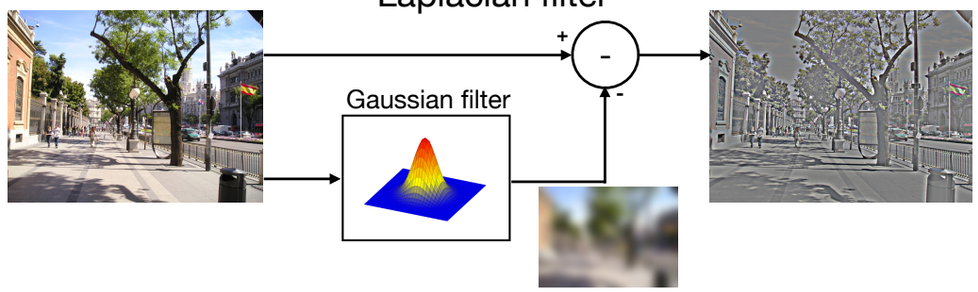
2D version by convolving 2 1D filters

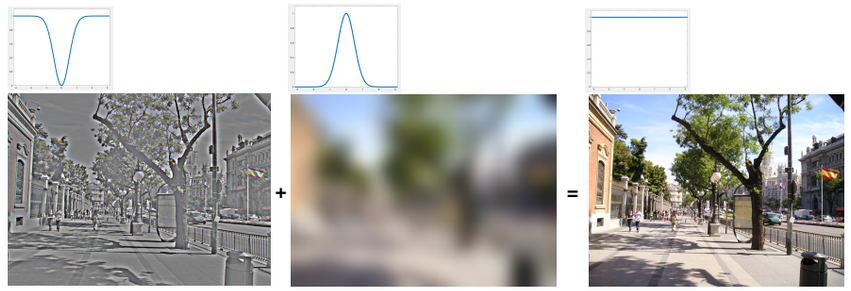
What about the opposite of blurring?

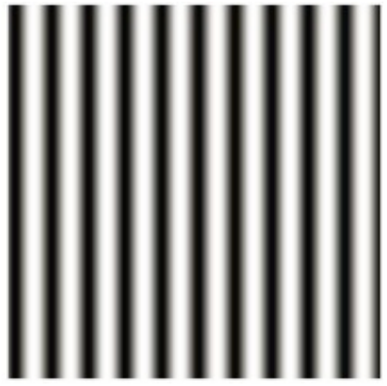


Sharpening

Laplacian filter







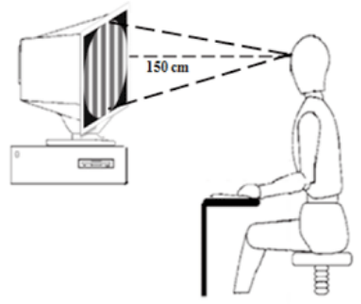


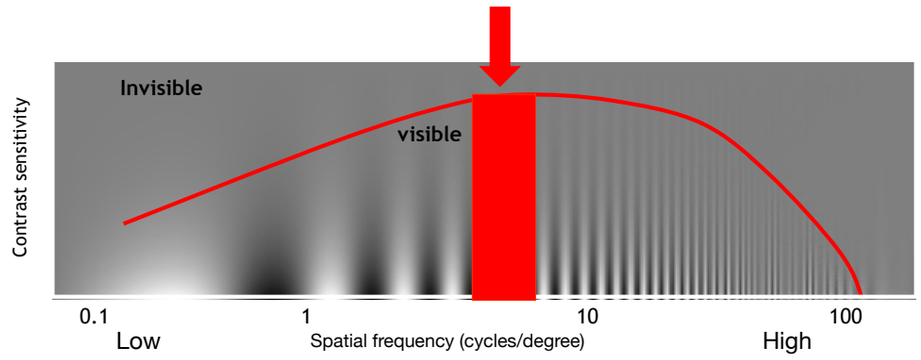
Figure 1. Stimulus presentation scheme. The stimuli were originally calibrated to be seen at a distance of 150 cm in a 19" display.

Our system responds to different special frequencies differently

Contrast Sensitivity Function

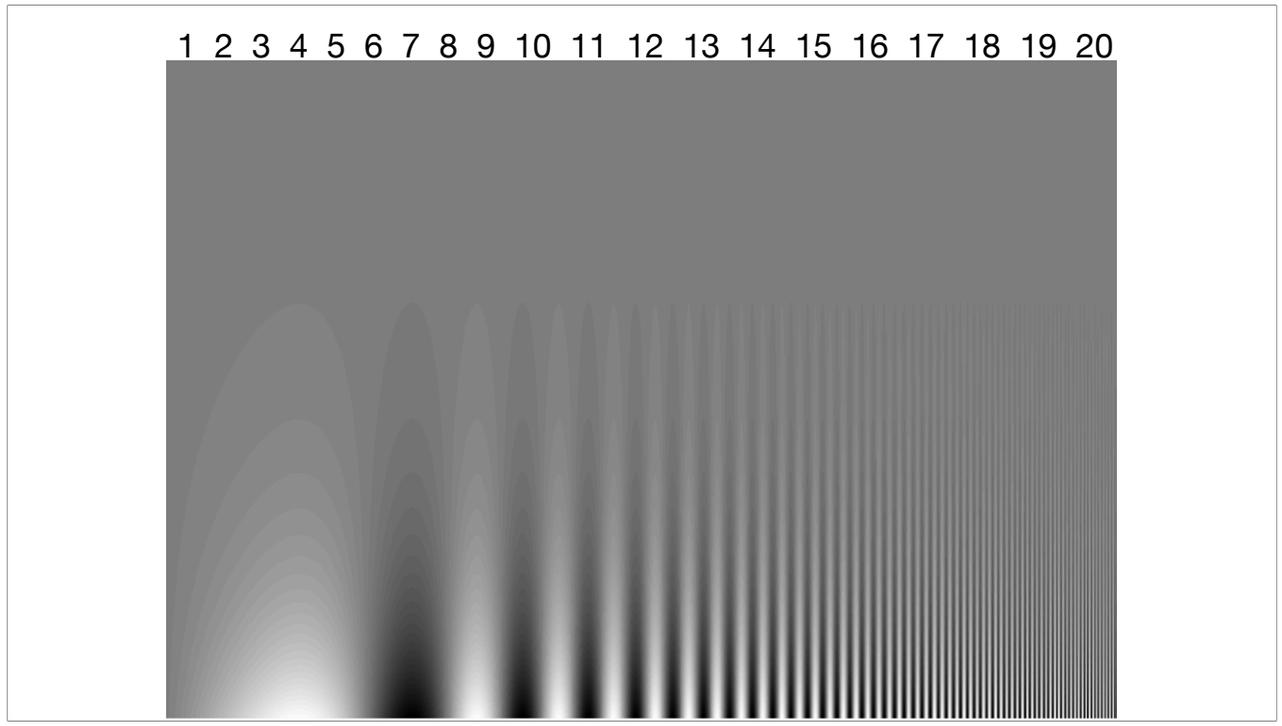
Blackmore & Campbell (1969)

Maximum sensitivity
~ **6** cycles / degree of visual angle



Things that are very close
and/or large are hard to see

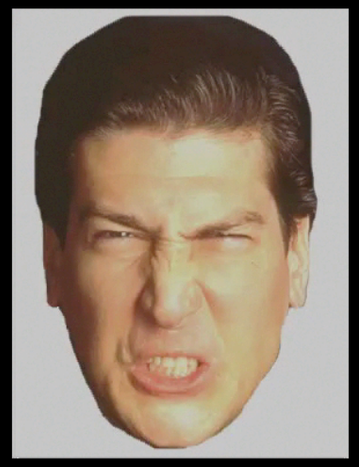
Things far away
are hard to see



When you’re far away you just see the low spatial frequencies, when you are close you see the high spatial frequencies

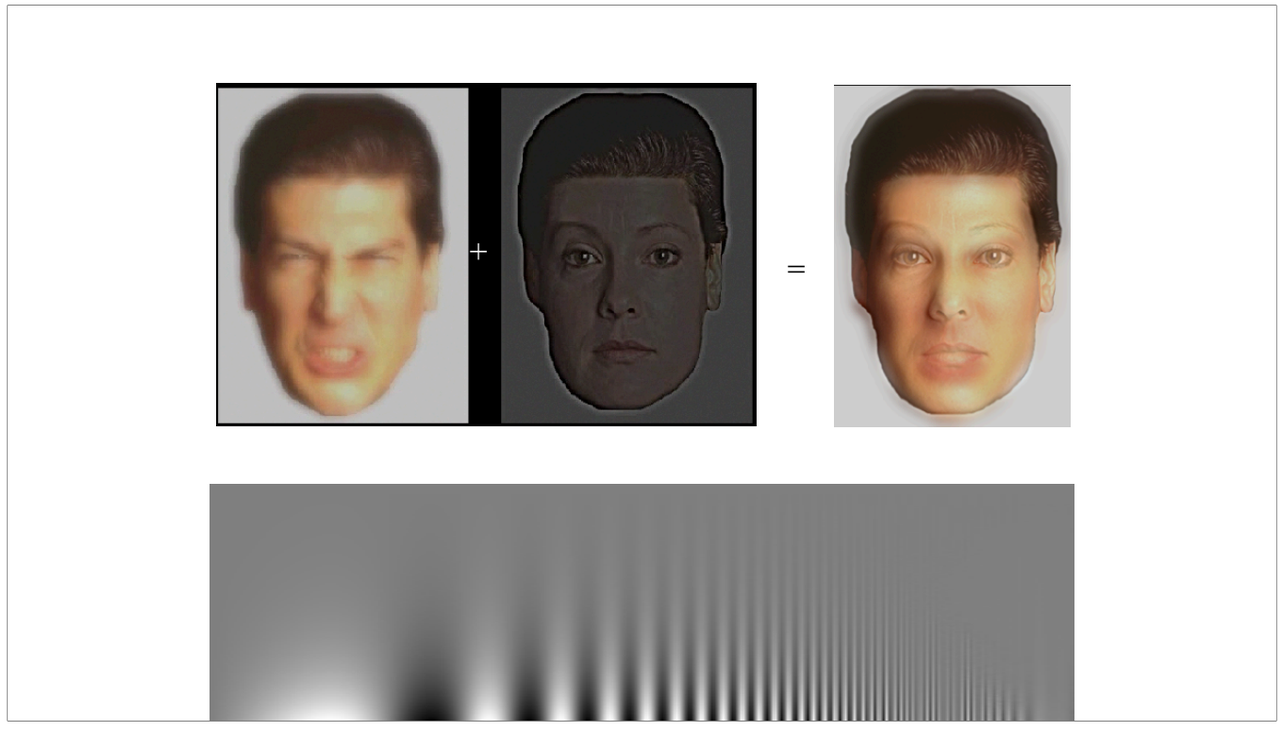
Hybrid Images

Oliva & Schyns



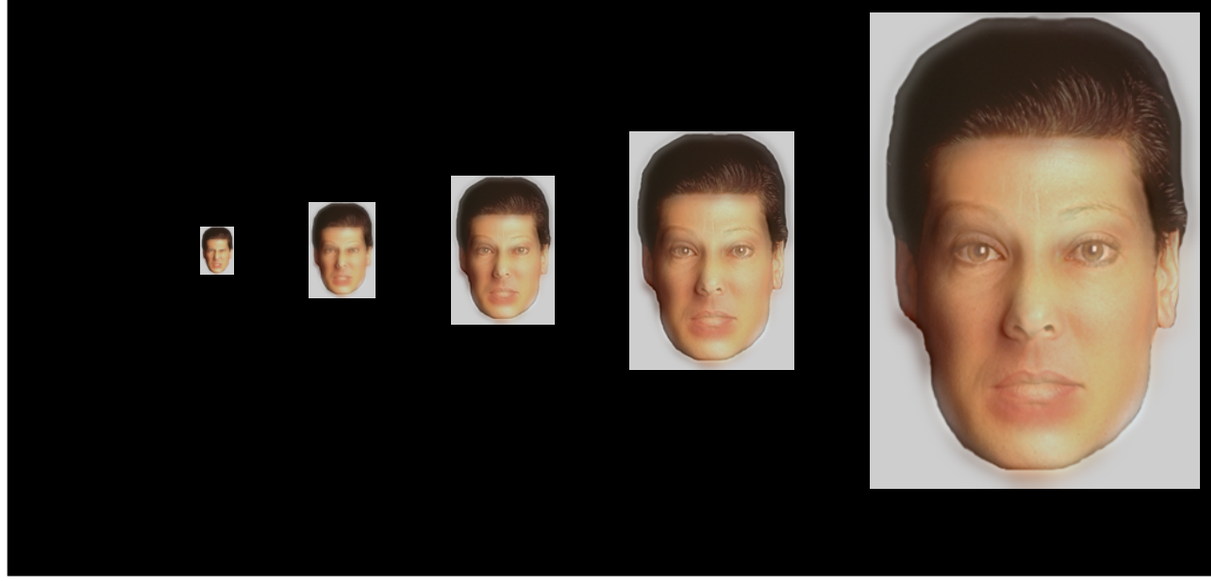
We start by taking two pictures, we isolate the details and contours in one, here the woman, and we blur the second face. as you can see, the man seems to go out of focus and the details of the woman are superposed.

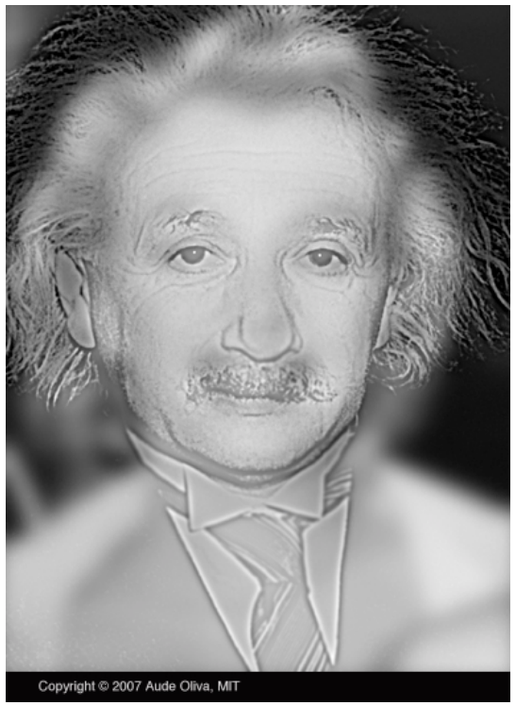
Hybrid Images

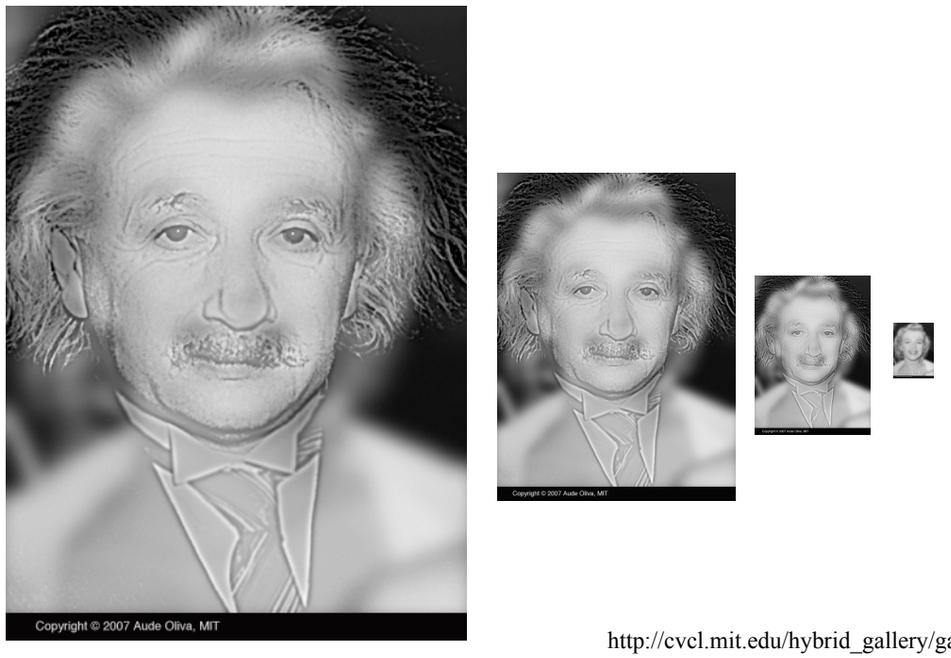


When you’re far away you just see the low spatial frequencies, when you are close you see the high spatial frequencies

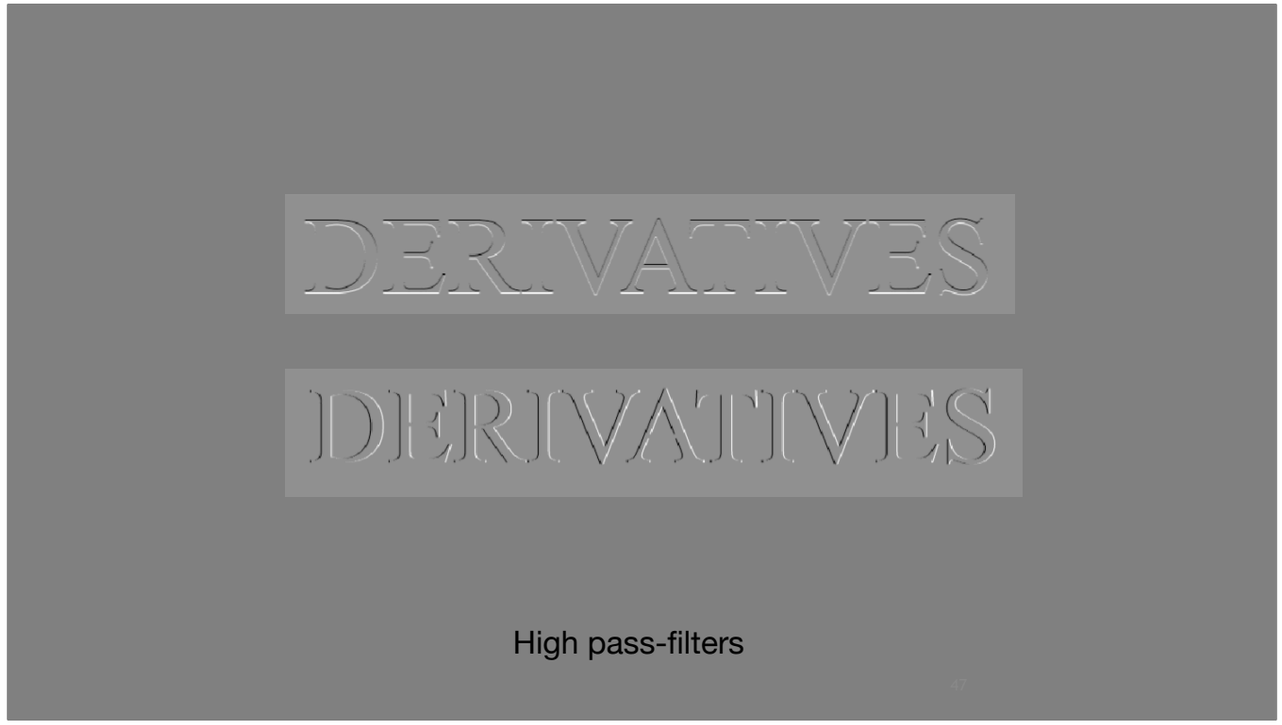
Hybrid Images




Copyright © 2007 Aude Oliva, MIT



http://cvcl.mit.edu/hybrid_gallery/gallery.html



High pass-filters

Finding edges in the image

Image gradient:

$$\nabla \mathbf{I} = \left(\frac{\partial \mathbf{I}}{\partial x}, \frac{\partial \mathbf{I}}{\partial y} \right)$$

Approximation image derivative:

$$\frac{\partial \mathbf{I}}{\partial x} \simeq \mathbf{I}(x, y) - \mathbf{I}(x-1, y)$$

Edge strength $E(x, y) = |\nabla \mathbf{I}(x, y)|$

Edge orientation: $\theta(x, y) = \angle \nabla \mathbf{I} = \arctan \frac{\partial \mathbf{I} / \partial y}{\partial \mathbf{I} / \partial x}$

Edge normal: $\mathbf{n} = \frac{\nabla \mathbf{I}}{|\nabla \mathbf{I}|}$

Image derivative [-1 1]

$$\frac{\partial \mathbf{I}}{\partial x} \simeq \mathbf{I}(x,y) - \mathbf{I}(x-1,y)$$

g[m,n]

⊗

[-1, 1]

h[m,n]

=

f[m,n]

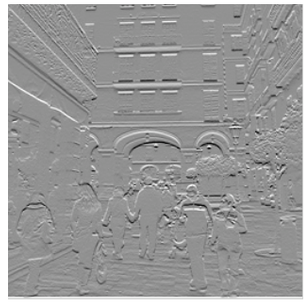
$[-1\ 1]^T$

g[m,n]

$\otimes$ $[-1, 1]^T$

h[m,n]

=



f[m,n]

Discrete derivatives

$$d_0 = [1, -1]$$

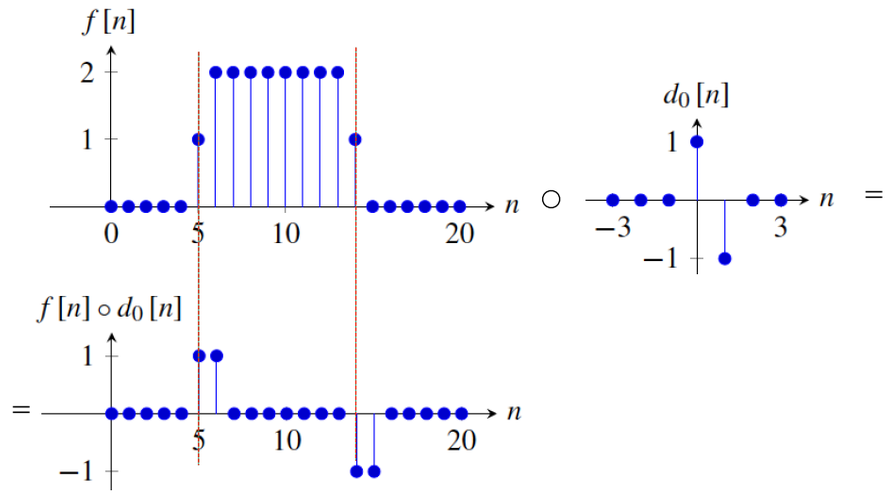
$$f \circ d_0 = f[n] - f[n-1]$$

$$d_1 = [1, 0, -1]/2$$

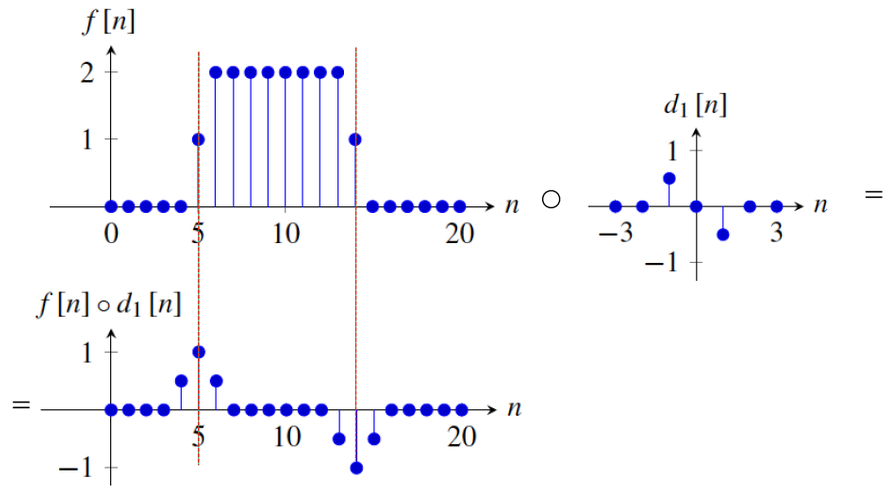
$$f \circ d_1 = \frac{f[n+1] - f[n-1]}{2}$$

This version can give you a better centering. In the first example the edge would be at one half pixel

Discrete derivatives



Discrete derivatives



Derivatives

We want to compute the image derivative:

$$\frac{\partial f(x,y)}{\partial x}$$

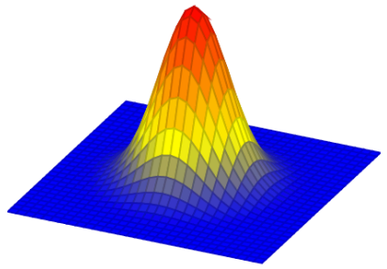
If there is noise, we might want to “smooth” it with a blurring filter

$$\frac{\partial f(x,y)}{\partial x} \circ g(x,y)$$

But derivatives and convolutions are linear and we can move them around:

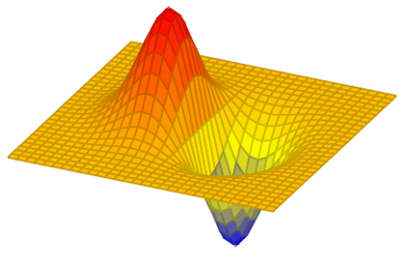
$$\frac{\partial f(x,y)}{\partial x} \circ g(x,y) = f(x,y) \circ \frac{\partial g(x,y)}{\partial x}$$

Gaussian derivatives



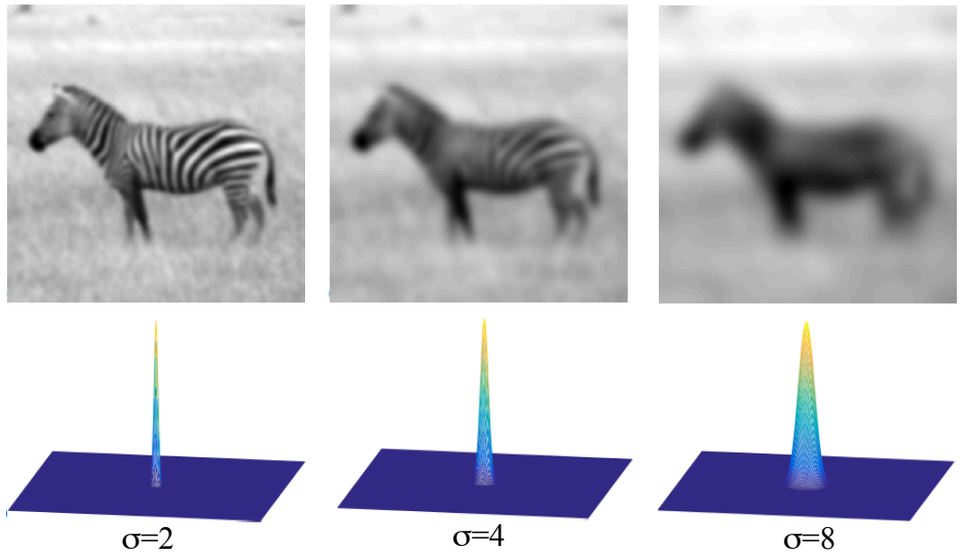
$$g(x,y;\sigma) = \frac{1}{2\pi\sigma^2}\exp -\frac{x^2+y^2}{2\sigma^2}$$

The continuous derivative is:



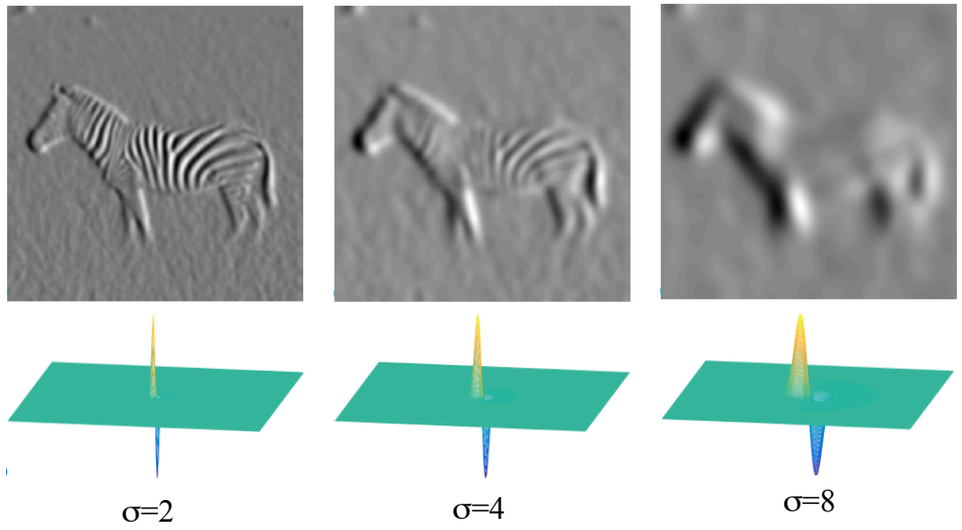
$$g_x(x,y;\sigma) = \frac{\partial g(x,y;\sigma)}{\partial x} = \\ = \frac{-x}{2\pi\sigma^4}\exp -\frac{x^2+y^2}{2\sigma^2} \\ = \frac{-x}{\sigma^2}g(x,y;\sigma)$$

Gaussian Scale



Analyze edges of the image (high-frequency edges) at a different scale

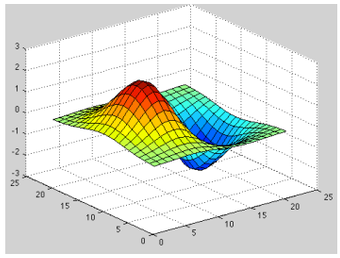
Derivatives of Gaussians: Scale



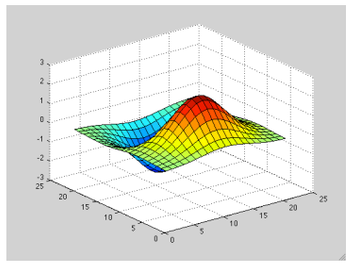
Edges at a different spatial scale - not picking strips, picking just edges of the body

Orientation

$$g_x(x,y) = \frac{\partial g(x,y)}{\partial x} = \frac{-x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

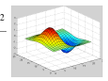


$$g_y(x,y) = \frac{\partial g(x,y)}{\partial y} = \frac{-y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

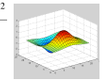


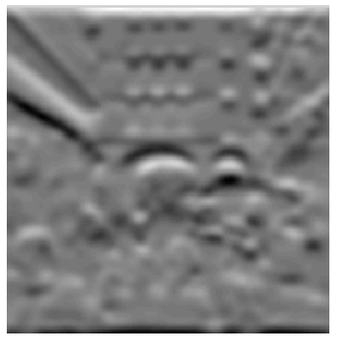
Orientation

$$g_x(x,y) = \frac{\partial g(x,y)}{\partial x} = \frac{-x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



$$g_y(x,y) = \frac{\partial g(x,y)}{\partial y} = \frac{-y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

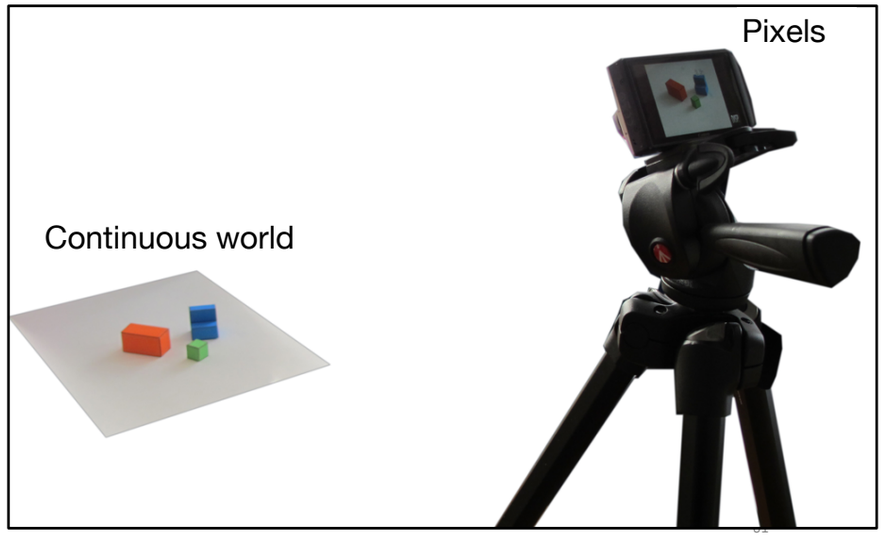




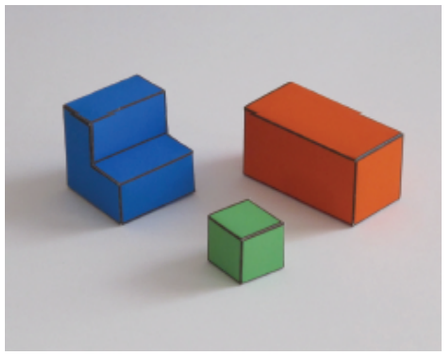
Sampling

You should always blur your image before you subsample it!

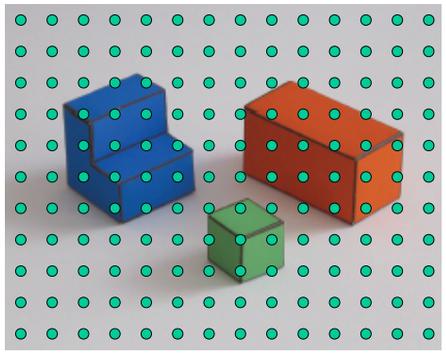
Sampling



Sampling



Sampling

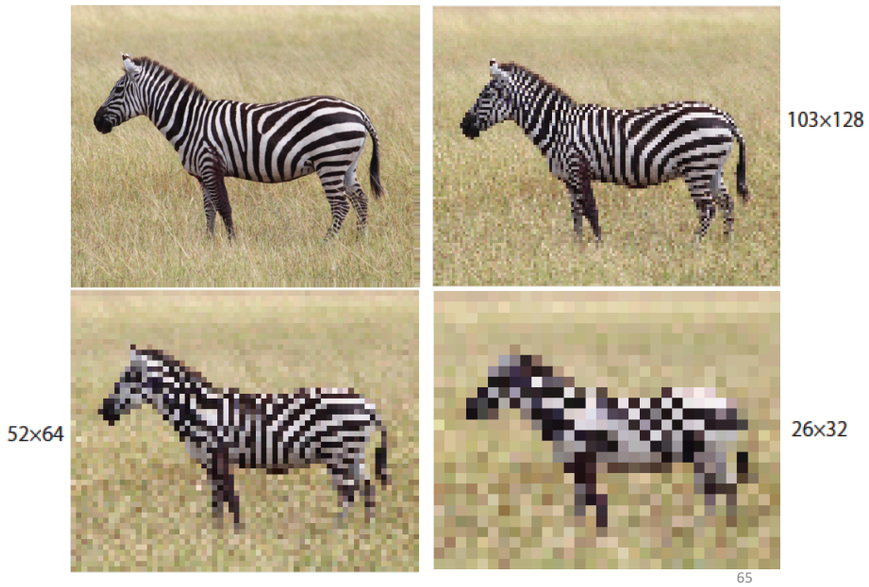


The question is: how many samples we need to capture the continuous thing well?

Aliasing

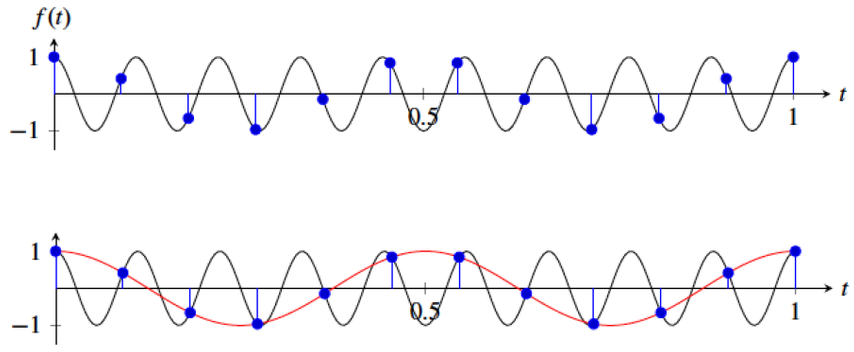
Let’s start with this continuous image (it is not really continuous…)

Aliasing



Subsample it (take every forth pixel or something)

Aliasing

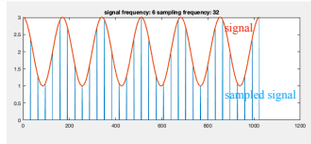


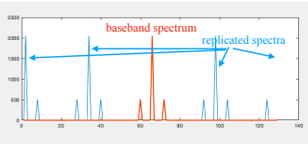
Both waves fit the same samples. Aliasing consists in “perceiving” the red wave when the actual input was the blue wave.

Red curve is the signal: sinusoid + constant

Blue shows sampled signal

spatial domain





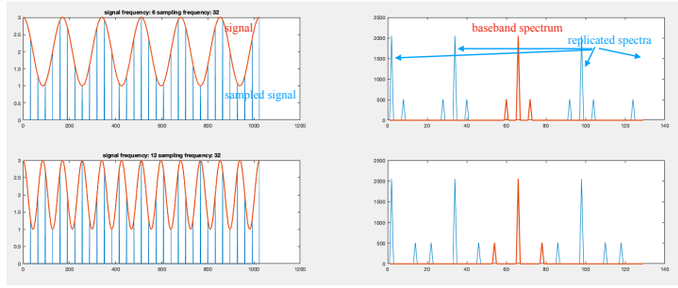
frequency domain

The number of replicas is equal to the sampling frequency

Red curve is the signal: sinusoid + constant

Blue shows sampled signal

spatial domain



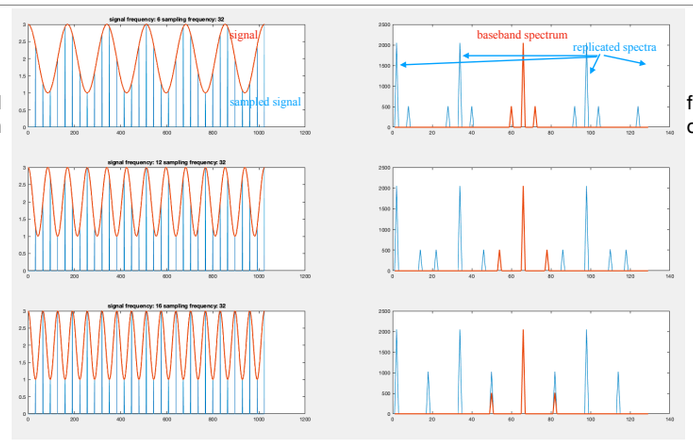
frequency domain

Red curve is the signal: sinusoid + constant

Blue shows sampled signal

spatial domain

frequency domain



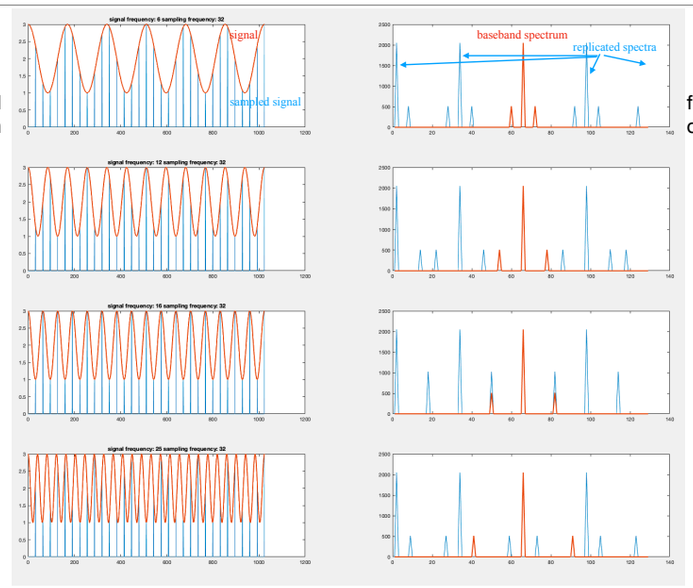
sampled at Nyquist frequency

Red curve is the signal: sinusoid + constant

Blue shows sampled signal

spatial domain

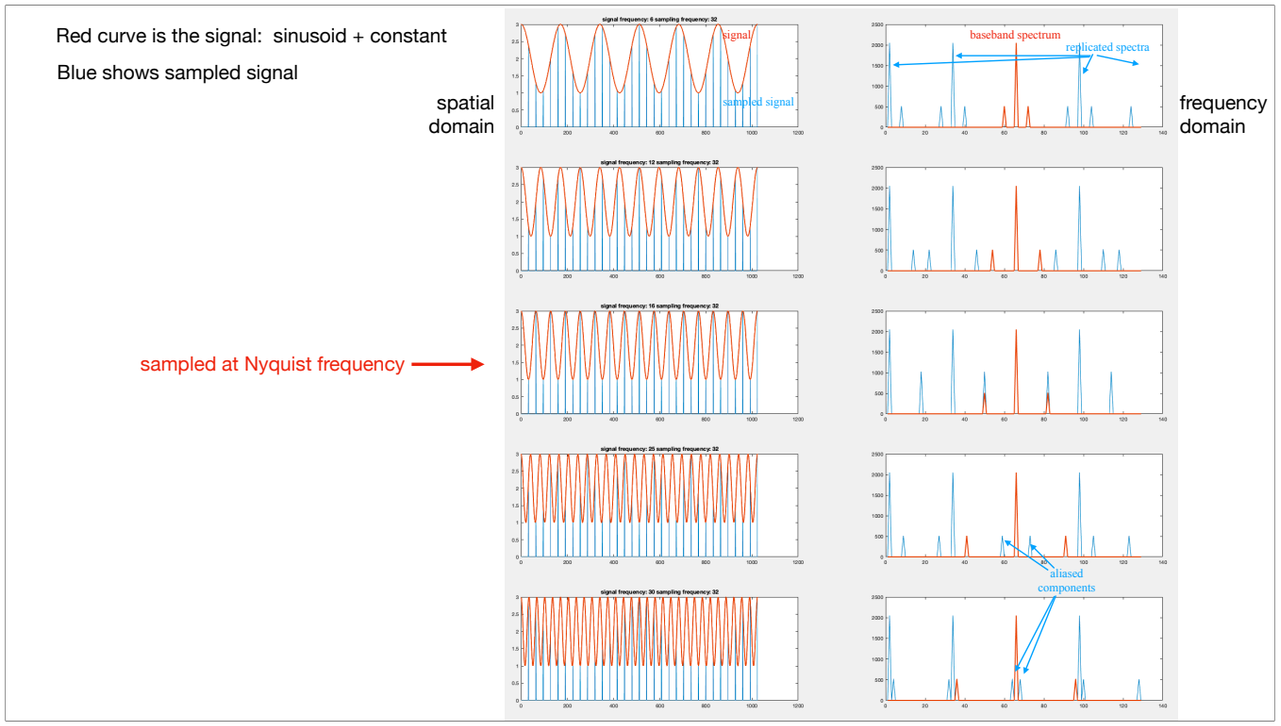
frequency domain



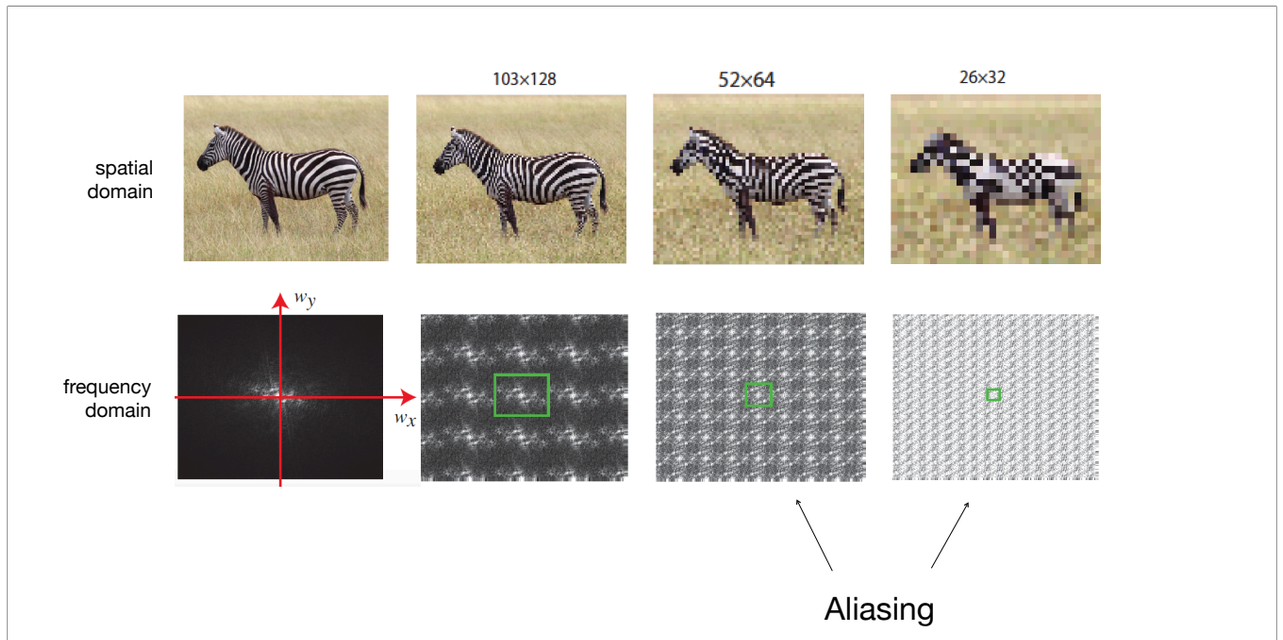
sampled at Nyquist frequency →

Red curve is the signal: sinusoid + constant

Blue shows sampled signal



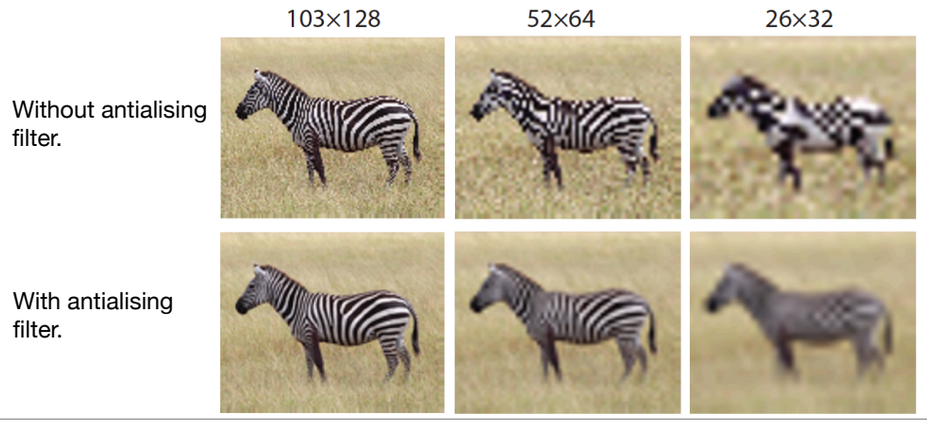
High frequency appears to be a low frequency of another thing. To avoid this situation, you do a low pass filter first, so you can capture at least 2 samples of your sinusoid.



You repeat copies by the spacing that is related to the sampling frequency.

Antialising filtering

Before sampling, apply a low pass-filter to remove all the frequencies that will produce aliasing: “blur before you subsample"



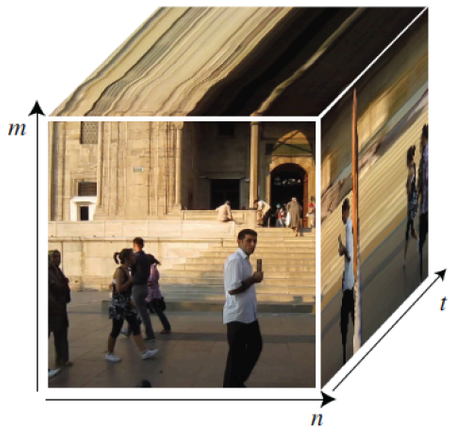
- **Temporal filtering**
- **Motion illusion**, involving aliasing, addressing whether humans match spatial patterns, or use temporal filters, to measure motion.

Temporal filtering

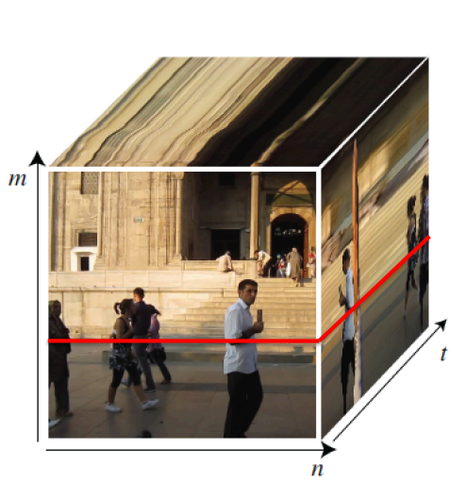
why filter videos over time?

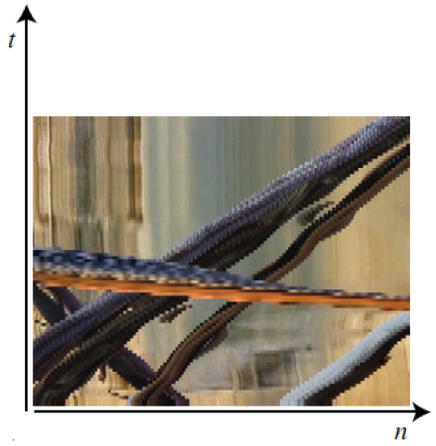
Sequences

time



Sequences

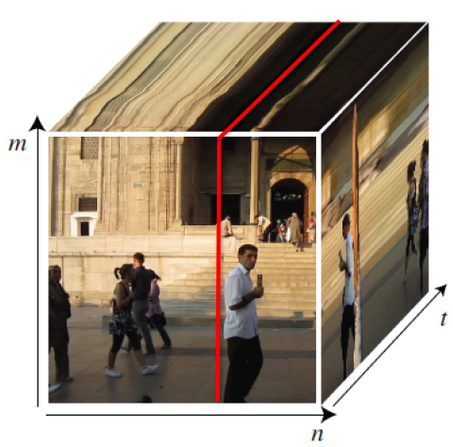




Cube size = 128x128x90

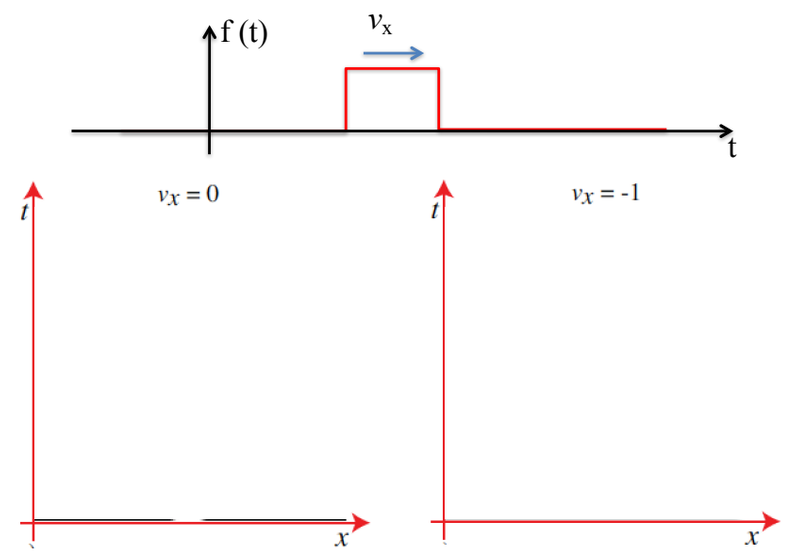
Slope says how fast it’s moving

Sequences

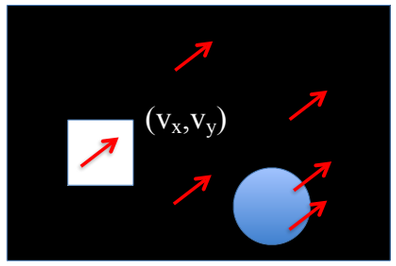


Cube size = 128x128x90

A box moving with speed v_x



Global constant motion



A global motion of the image can be written as:

$$f(x, y, t) = f_0(x - v_x t, y - v_y t)$$

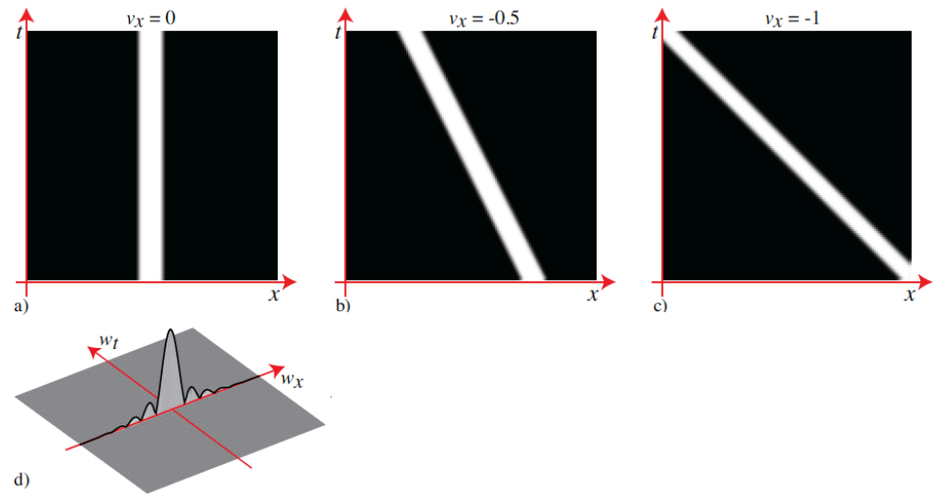
where:

$$f_0(x, y) = f(x, y, 0)$$

$$f(x,y,t) = f_0(x - v_x t, y - v_y t)$$

$$\updownarrow$$

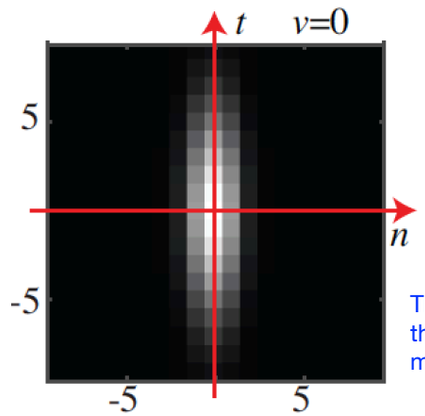
$$F(w_x, w_y, w_t) = F_0(w_x, w_y)\delta(w_t + v_x w_x + v_y w_y)$$



Our Fourier signal is the original signal multiplied by the delta function.
Fourier transformation of a box is a sink function (sin x/x)
Fourier domain is just a different representation that makes certain operations simpler. Like defining motion - just calculate the orientation of the signal to keep track of the direction of motion.

Temporal Gaussian

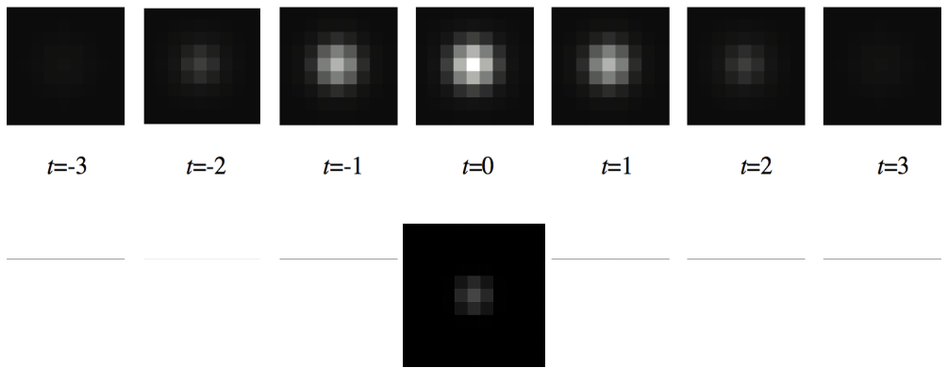
$$g(x, y, t; \sigma_x, \sigma_t) = \frac{1}{(2\pi)^{3/2}\sigma_x^2\sigma_t} \exp -\frac{x^2 + y^2}{2\sigma_x^2} \exp -\frac{t^2}{2\sigma_t^2}$$



This filter keeps stationary things sharp, and blurs moving things.

Filters remove some frequencies we don’t like and keep the ones we like.
Blurring more in time than in space with this filter.

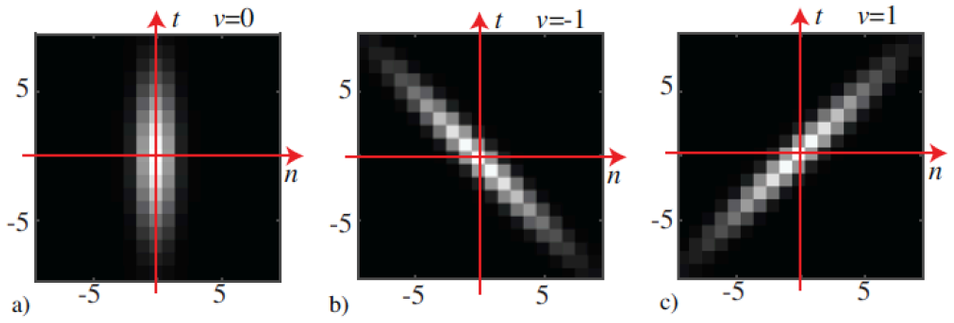
Spatio-temporal Gaussian



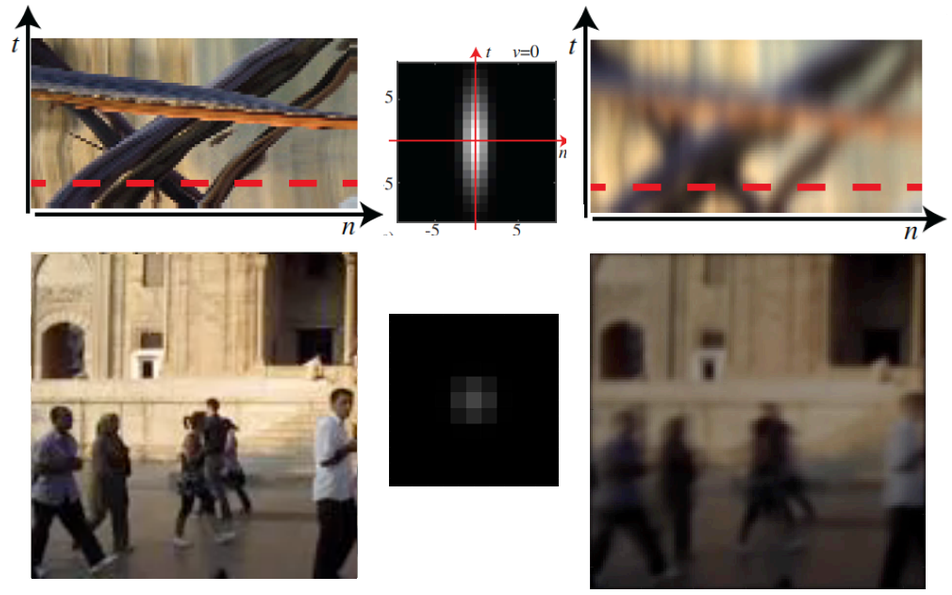
Spatio-temporal Gaussian

How could we create a filter that keeps sharp objects that move at some velocity **(vx , vy)** while blurring the rest?

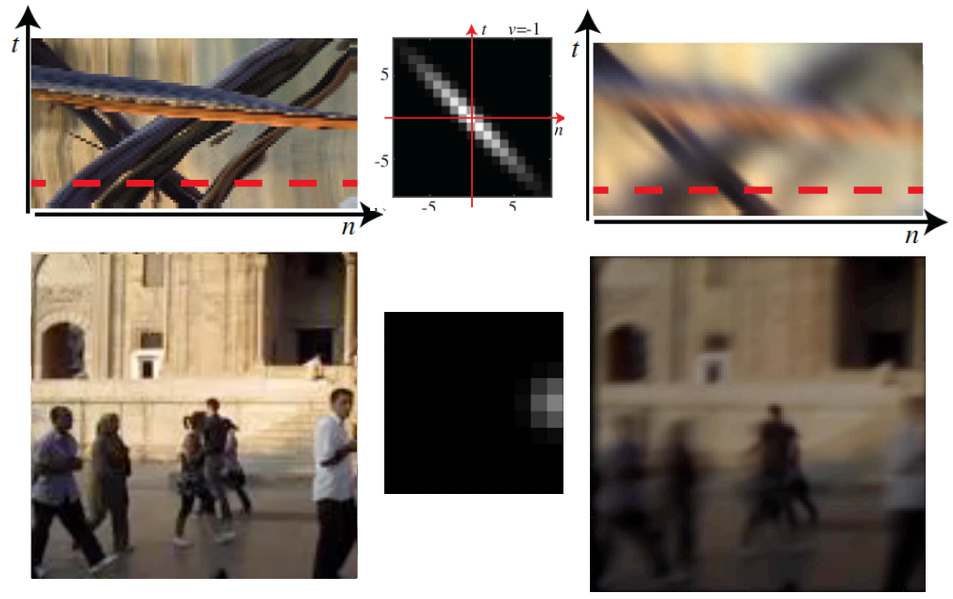
$$g_{v_x,v_y}(x,y,t) = g(x - v_x t, y - v_y t, t)$$



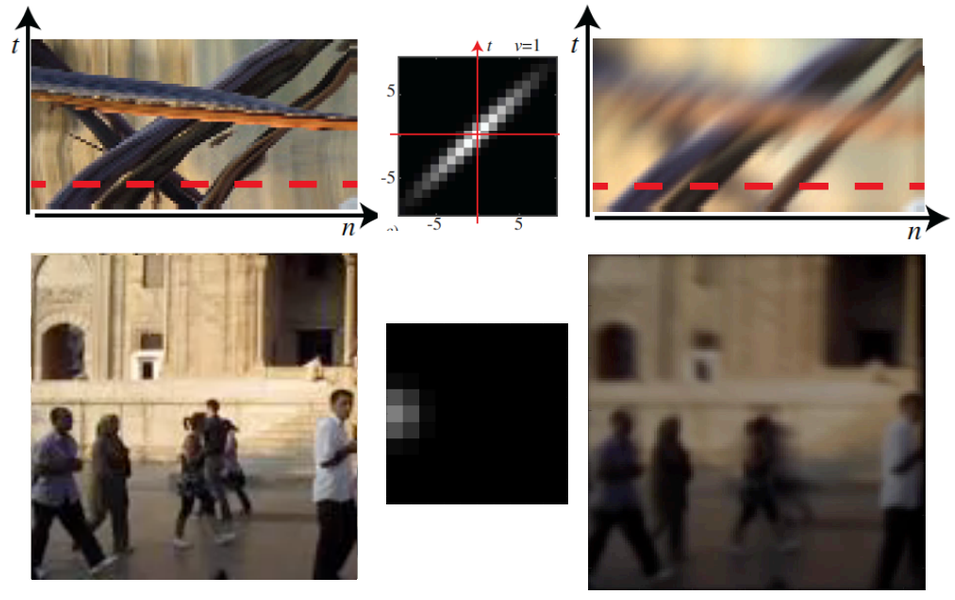
We can design Gaussians that pick up a particular direction.



Grab stationary objects

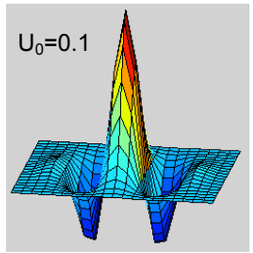


Grab people moving to the left…

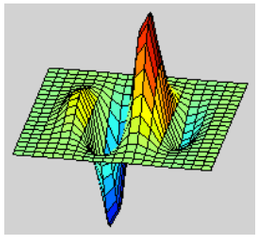

t
n
t
v=1
5
-5
-5
5
t
n

Quadrature pair of Gabor filters

$$\psi_c(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(2\pi u_0 x)$$



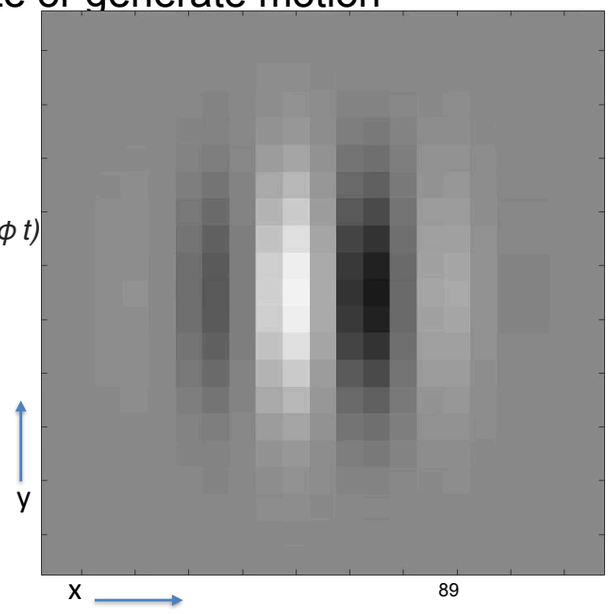
$$\psi_s(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \sin(2\pi u_0 x)$$



Gabor is a Gaussian wavelet multiplied by cosine

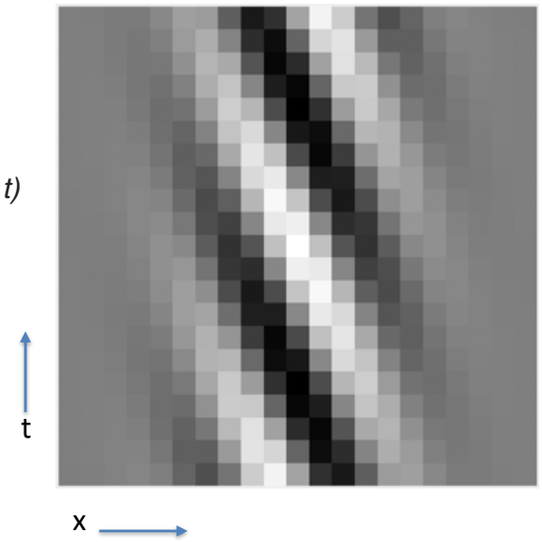
Using phase changes of local Gabor filters to analyze or generate motion

$$\psi_c(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos\left(2\pi u_0 x + \phi\, t\right)$$



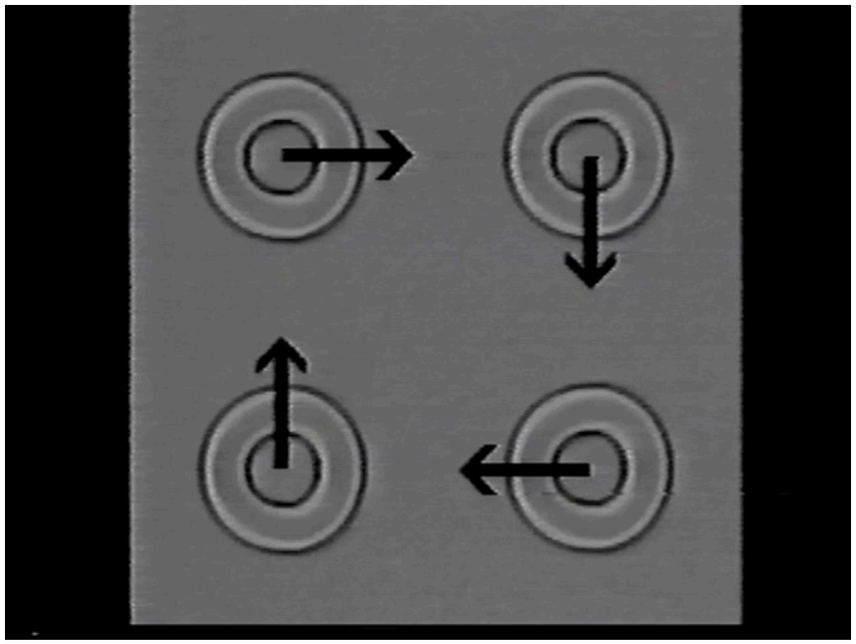
Space-time plot of the a slice through the patio-temporal filter of the previous slide

$$\psi_c(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(2\pi u_0 x + \phi\, t)$$



Orientation in space time is speed in real world

Motion without movement



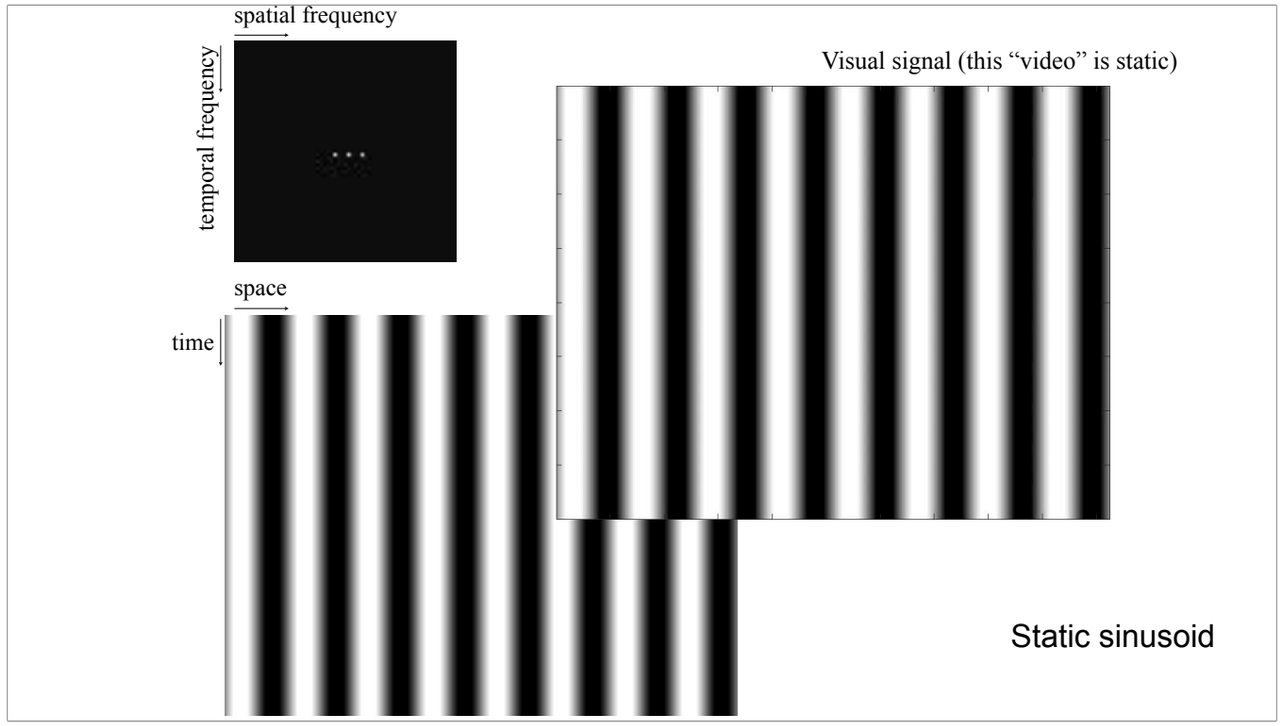
Spatio-temporal sampling illusion, due to
Edward Adelson and Jim Bergen

Evidence for filter-based analysis of motion in the human visual system shown via spatio-temporal visual illusion based on sampling

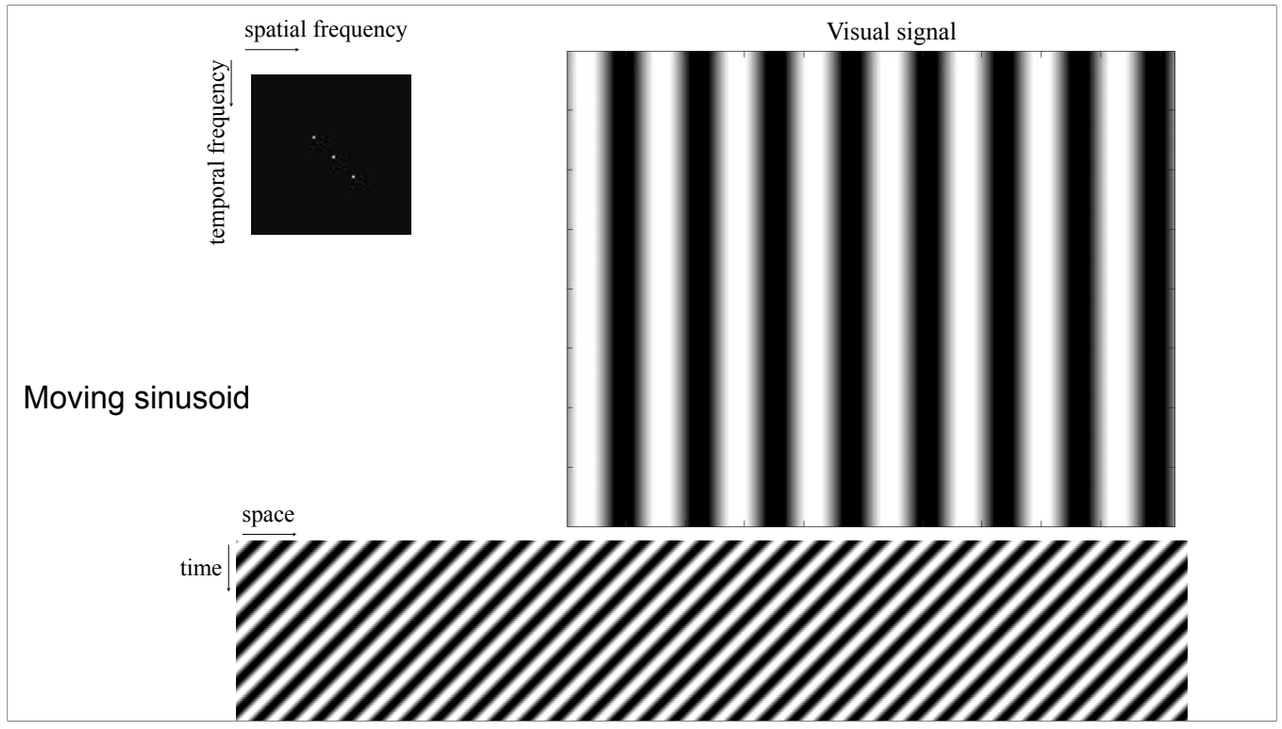
Two potential theories for how humans compute our motion perceptions:

(a) We match the pattern in the image that we see at one moment and compare it with what we see at subsequent times.
(b) We use spatio-temporal filters to measure spatio-temporal energy in order to measure local motion.

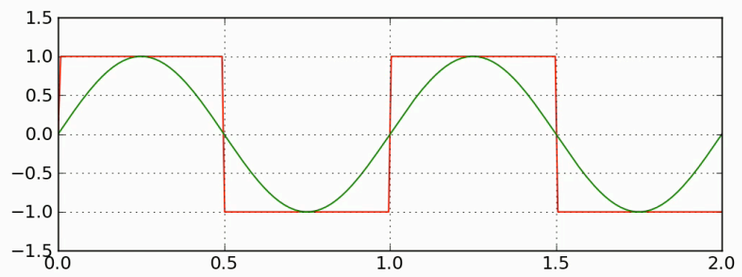
This illusion favors one theory over the other.



Start with a picture of a sin wave, a static wave that’s not moving in time

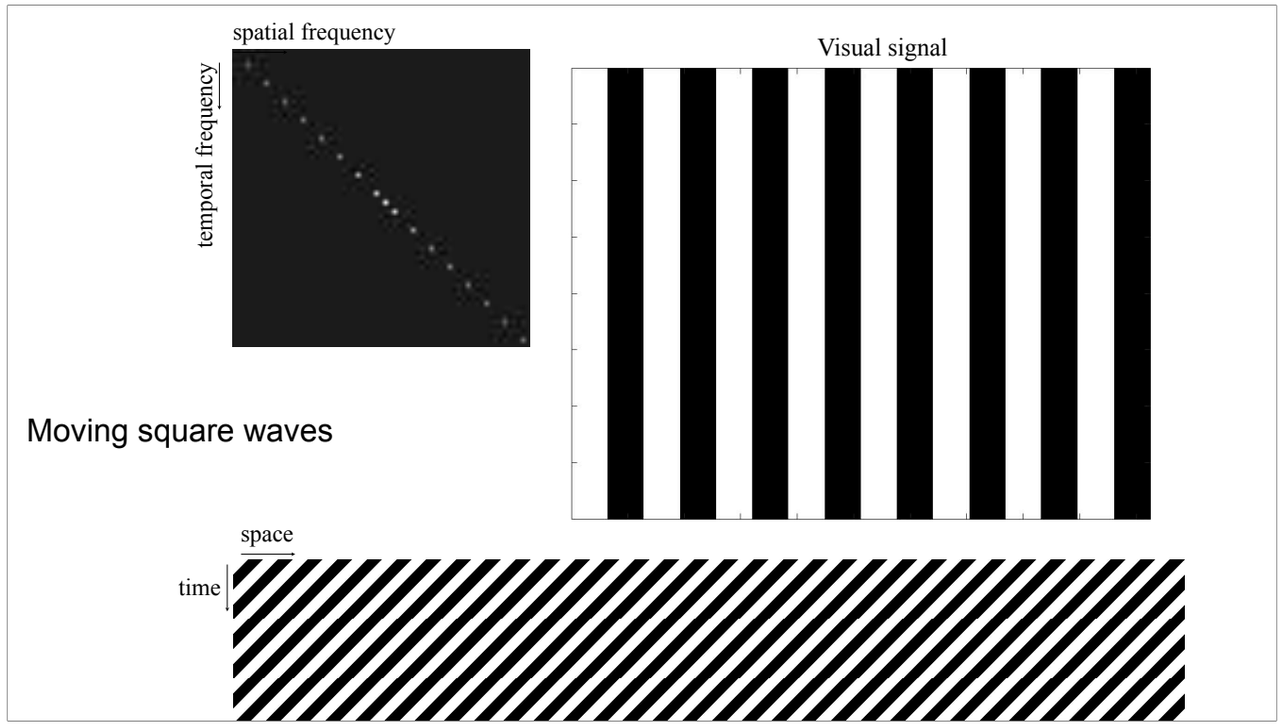

spatial frequency
temporal frequency
Visual signal
Moving sinusoid
space
time

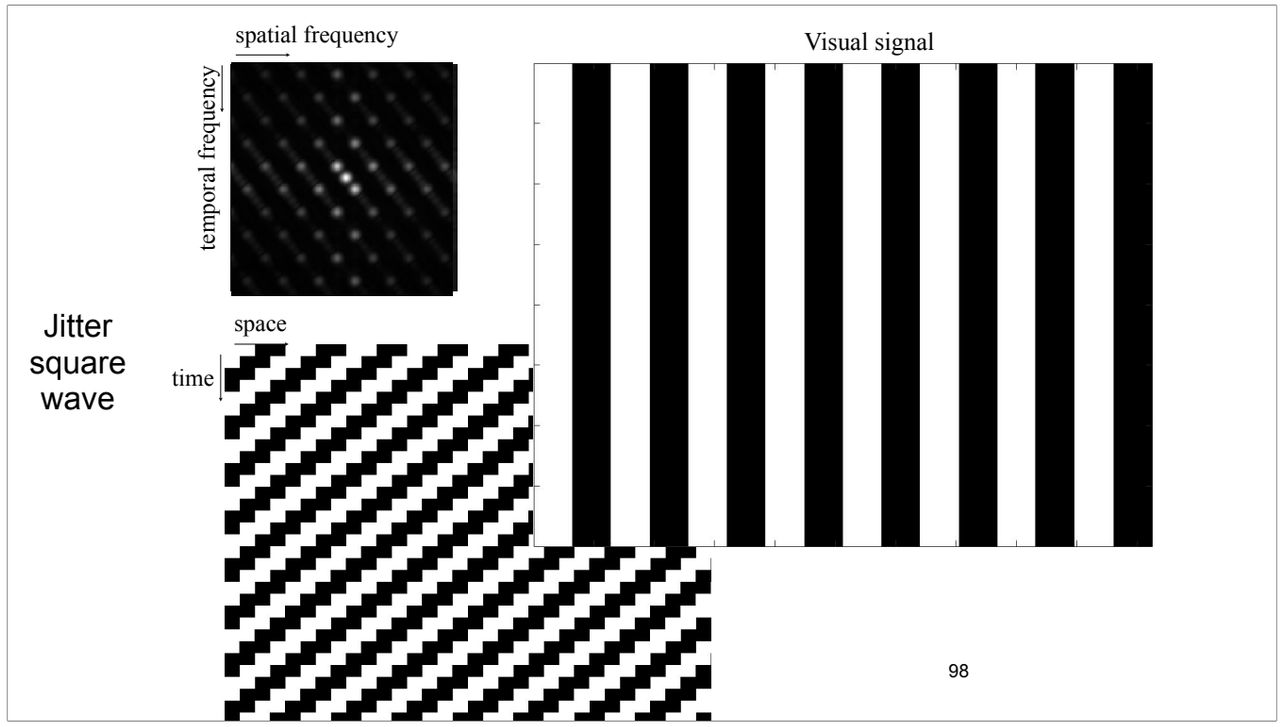
A square wave is an infinite sum of sinusoids

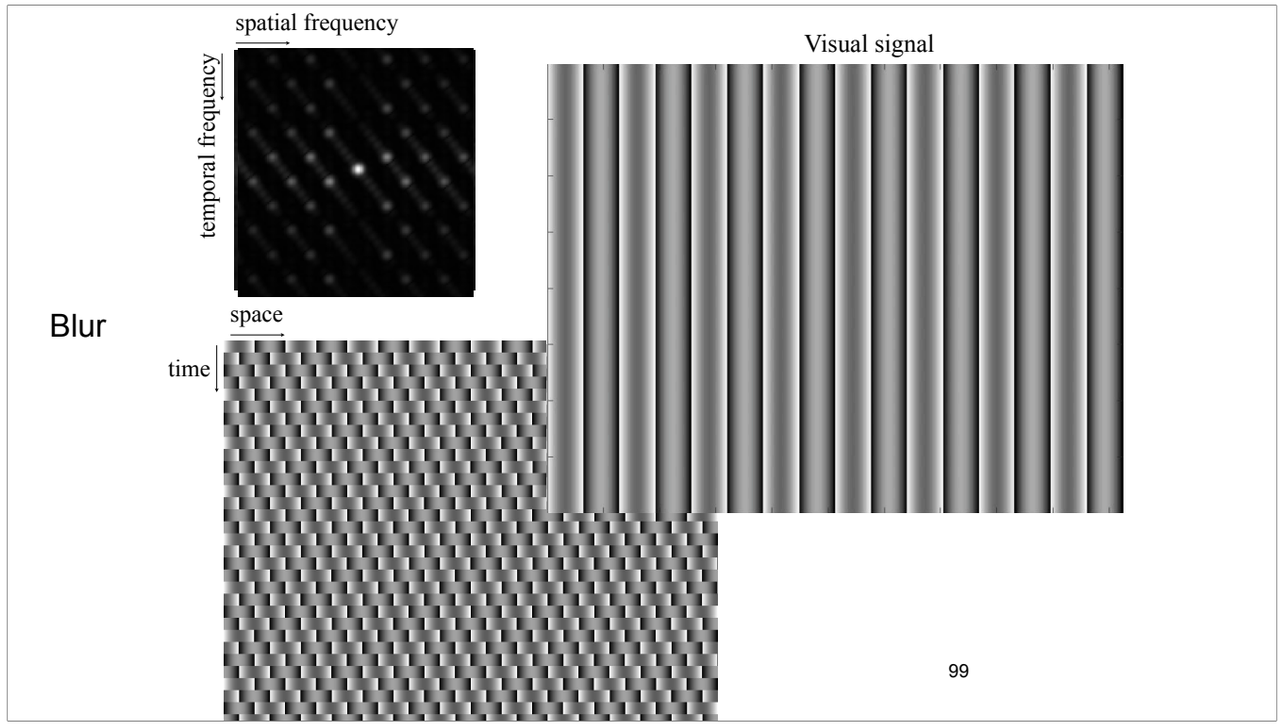


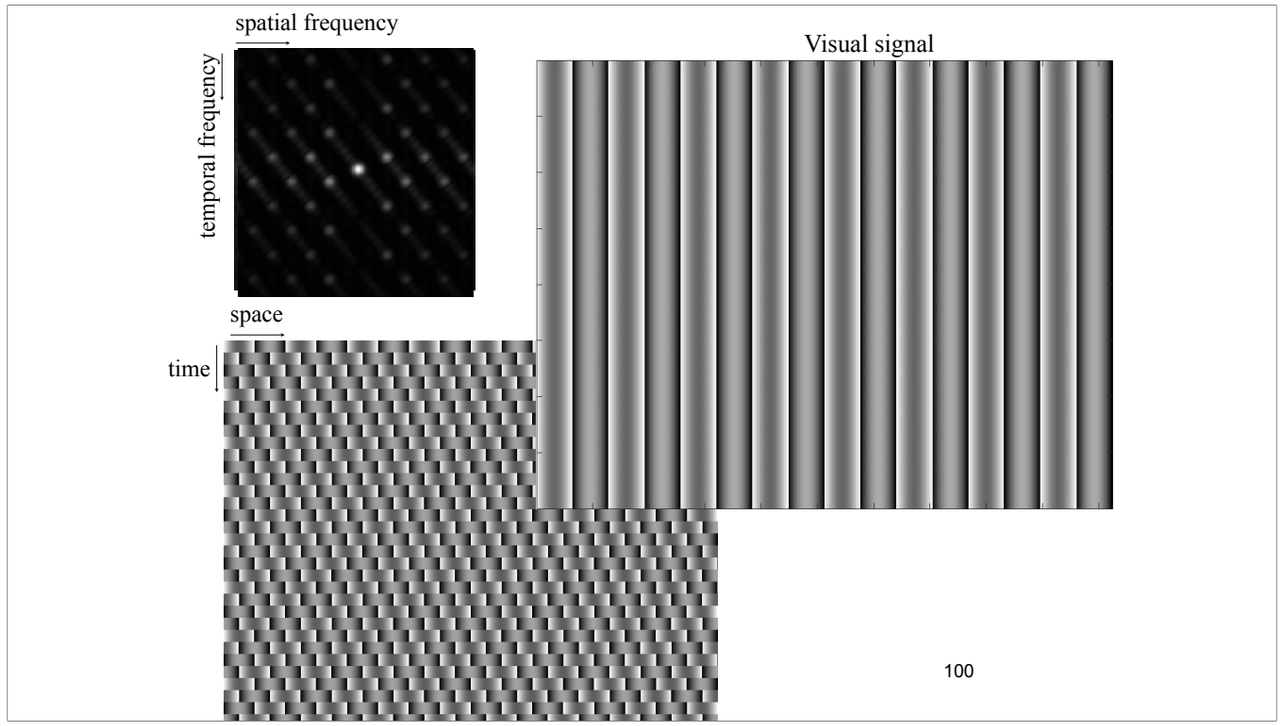
Instead of looking at sinusoids we are going to use square waves

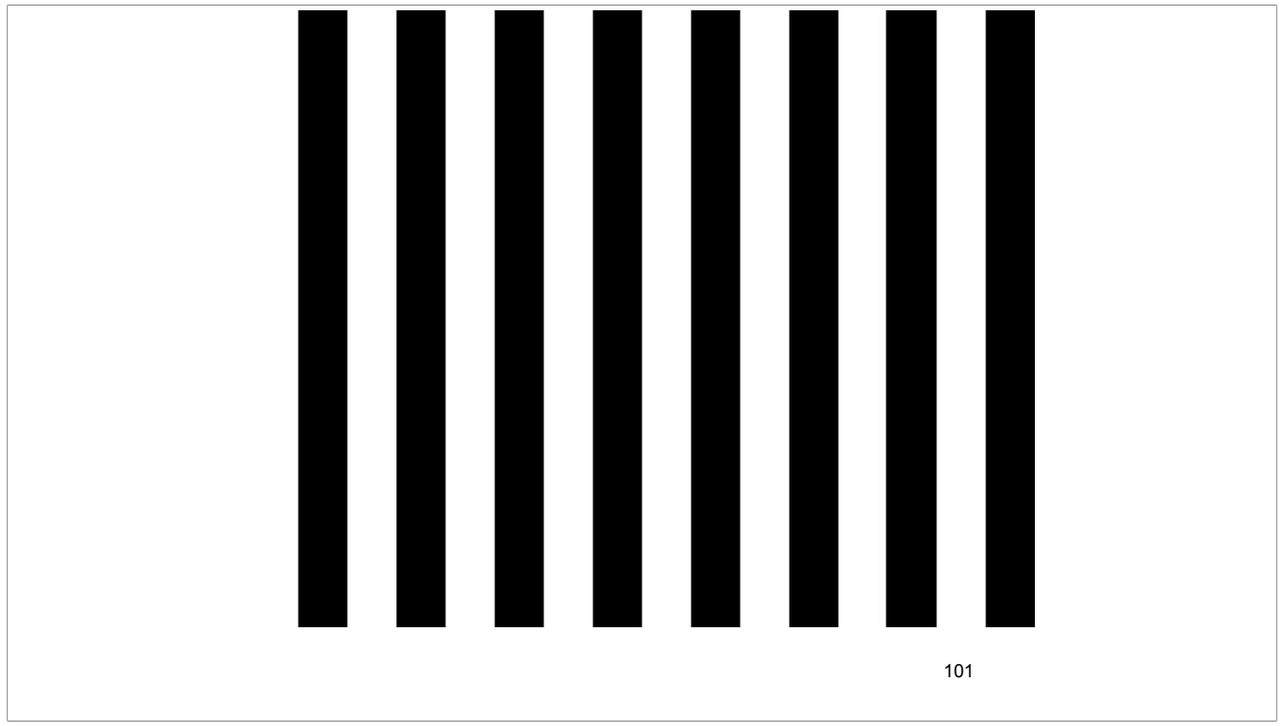
Add higher frequency sins, this is represented as a sum of a lot of sins

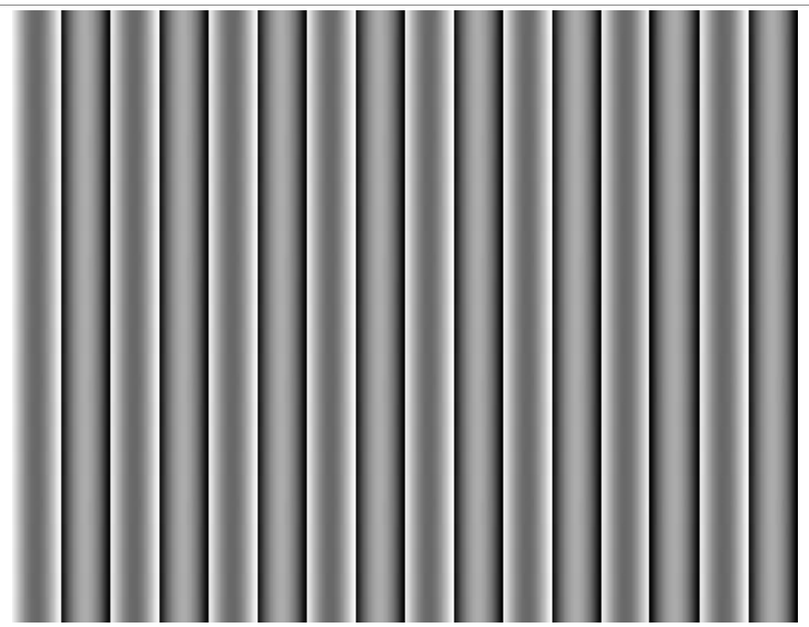

spatial frequency
temporal frequency
Visual signal
Moving square waves
space
time

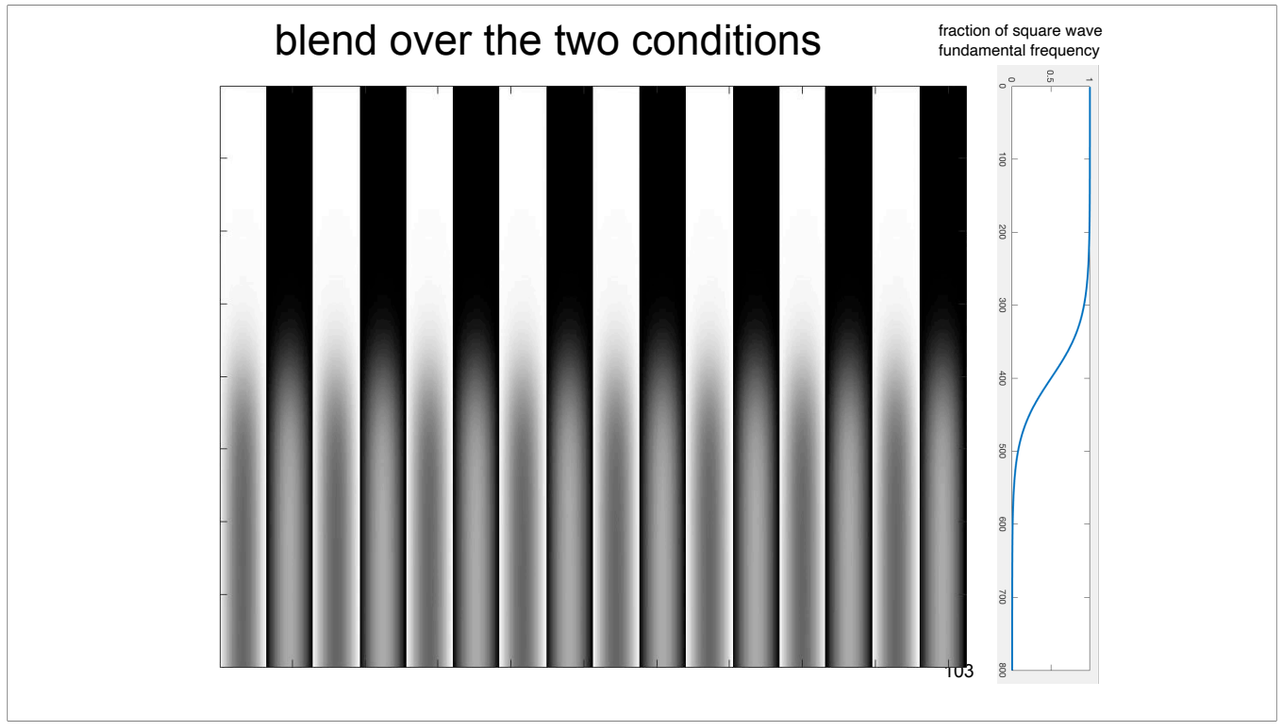

spatial frequency
temporal frequency
Visual signal
Jitter square wave
space
time


spatial frequency
temporal frequency
Visual signal
Blur
space
time

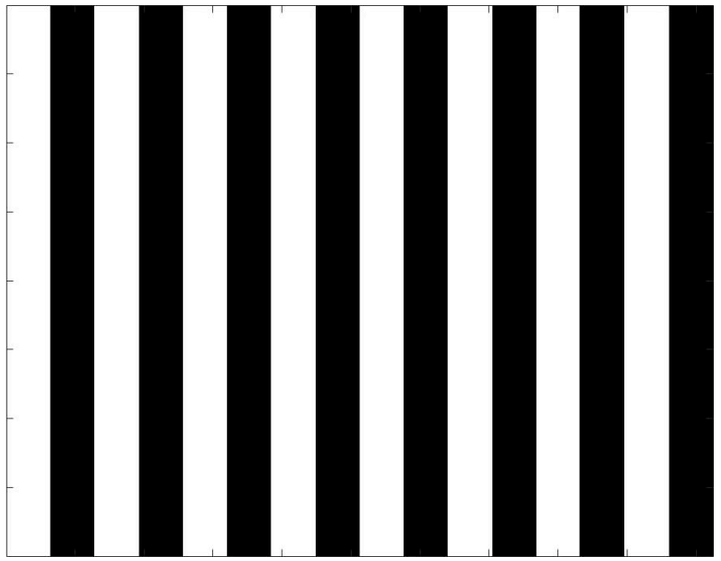

spatial frequency
temporal frequency
Visual signal
space
time
100


101



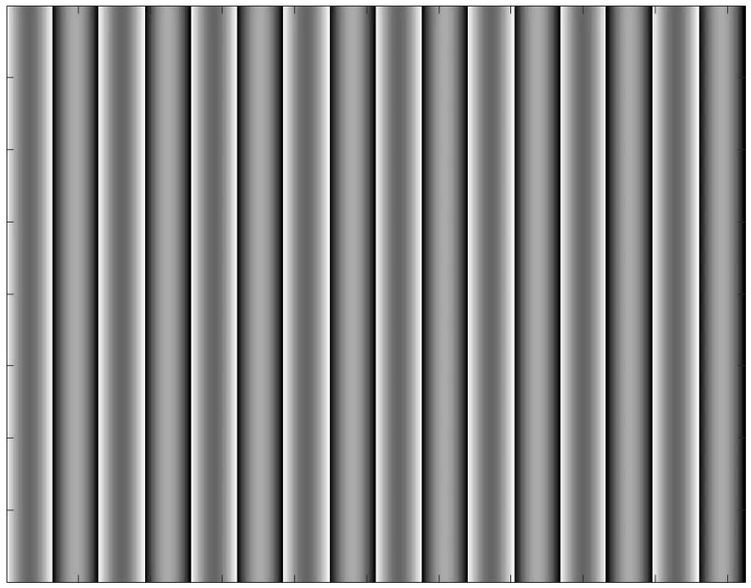

blend over the two conditions
fraction of square wave
fundamental frequency
0
0.5
1
0
100
200
300
400
500
600
700
800
103

faster display speed



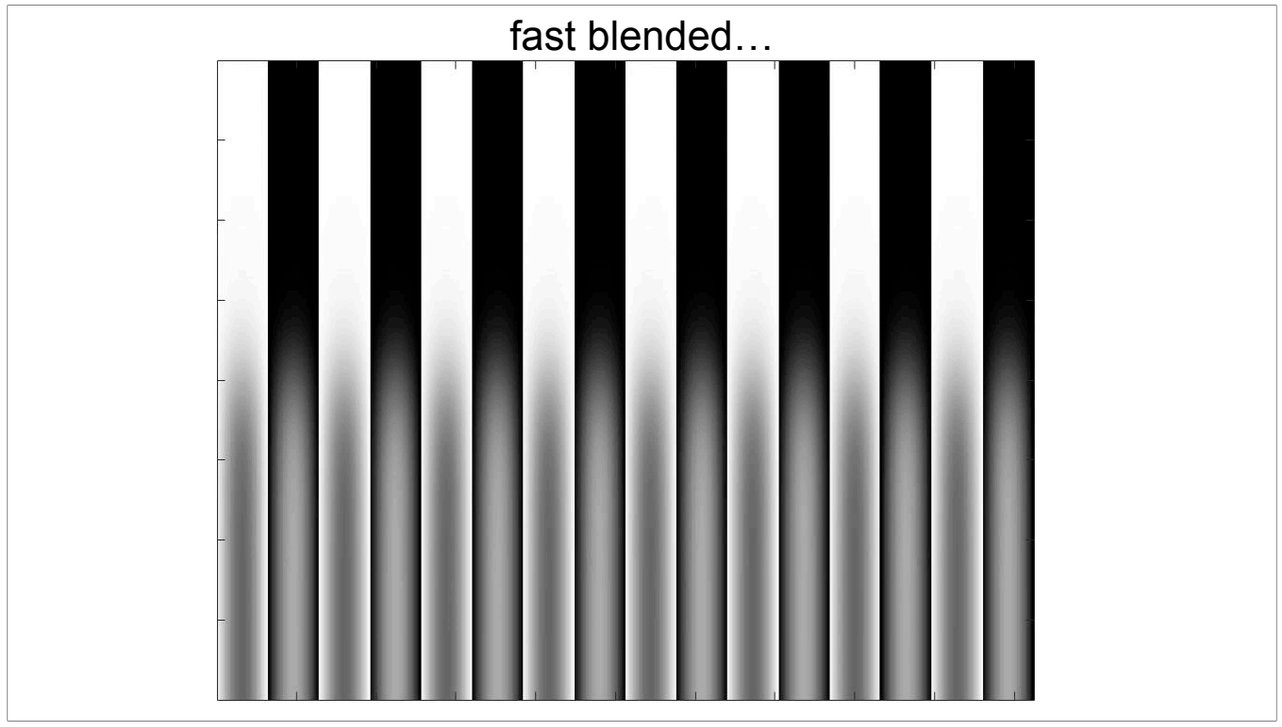
alpha: 1 squareFlag: 1 offset: period/4

faster display speed



105

alpha: 0 squareFlag: 1 offset: period/4


fast blended…

lecture summary

- We have “inverted U shaped” sensitivity to spatial frequencies, peaking at 6 cycles per degree.
- We discussed ways to filter out different spatial frequency components of an image.
- Aliasing: “blur before you subsample”.
- Spatio-temporal filtering enables motion analysis.
- Motion illusion gives evidence some temporal filtering mechanisms are involved in our motion processing.