

Penn
Engineering

ONLINE LEARNING

Video 2.1
Kostas Daniilidis

A Beginner's Guide To Understanding Convolutional Neural Networks

Convolutional Neural Networks (CNNs) +3

What is a convolutional neural network?

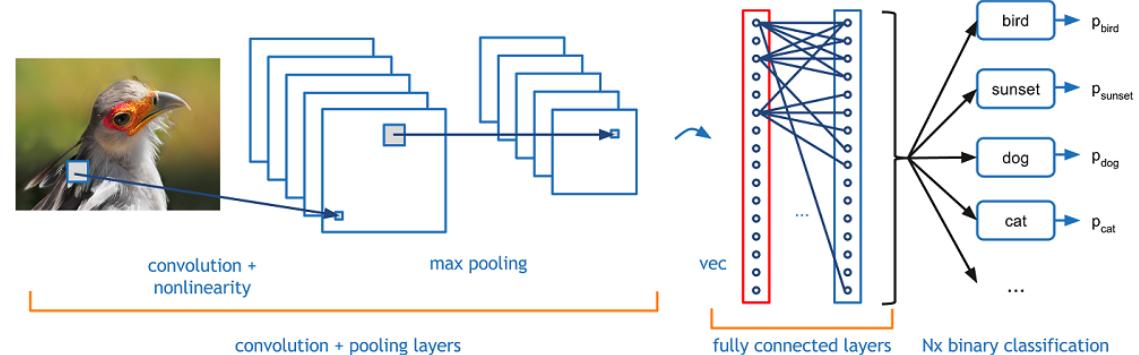
[Answer](#) [Request](#) Follow 76 Comment Share 2 Downvote

UNDERSTANDING CONVOLUTIONAL NEURAL NETWORKS FOR NLP

When we hear about Convolutional Neural Network (CNNs), we typically think of Computer Vision. CNNs were responsible for major breakthroughs in Image Classification and are the core of most Computer Vision systems today, from Facebook's automated photo tagging to self-driving cars.

patterns; with more than one, you could look for patterns of patterns. Take the case of image recognition, which tends to rely on a contraption called a “convolutional neural net.” (These were elaborated in a [seminal 1998 paper](#) whose lead author, a Frenchman named Yann LeCun, did his postdoctoral research in Toronto under Hinton and now directs a huge A.I. endeavor at

Next, Zweig and co optimized their own deep-learning systems based on convolutional neural networks with varying number of layers, each of which processes a different aspect of speech. They then used the



digital input and send output to other nodes. Layers upon layers of these nodes make up so-called **convolutional** neural networks, which, with sufficient training data, have become better and better at identifying images.

Inside a **Convolutional** Neural Network

But how does this filtering work? The secret is in the addition of two new types of layers: **convolutional** and pooling layers. We'll break the process down below, using the example of a network designed to do just one thing: determine whether a picture contains a grandma or not.



Got a tip? [Let us know.](#)

News ▾ Video ▾ Events ▾ Crunchbase

DISRUPT NY Find out how startups are transforming real estate investing at Disrupt NY [Get tickets](#)

convolutional neural networks

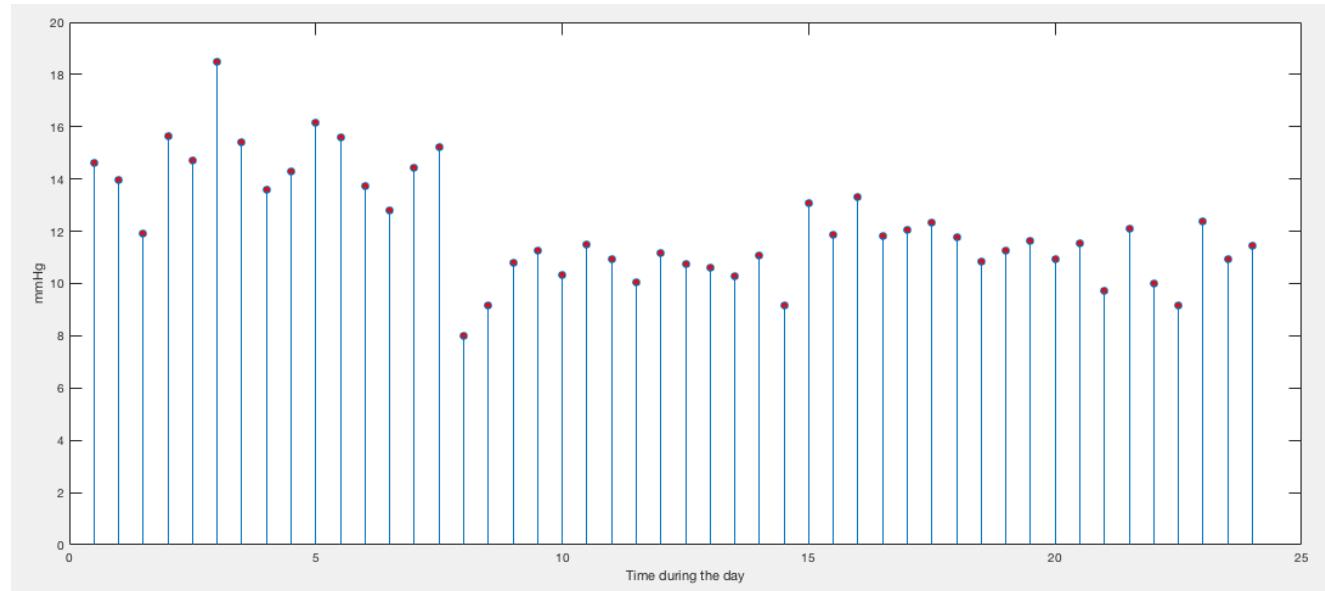
Convolution

$$(f \star g)(t) = \int_{-\infty}^{\infty} f(t')g(t - t')dt'$$

Is Convolution that convoluted?

Let us start with discrete 1D signals !

For example, my systolic blood pressure measured with a wearable over 24 hours on 1/16/2016:



$s[n]$

Interested in the largest drop?

15	14	12	16	15	18	15	14	14	16	16	14	13	14	15	8	9	11	11	10	11
-1		1																		
-1		1																		
-1		1																		
-1		1																		

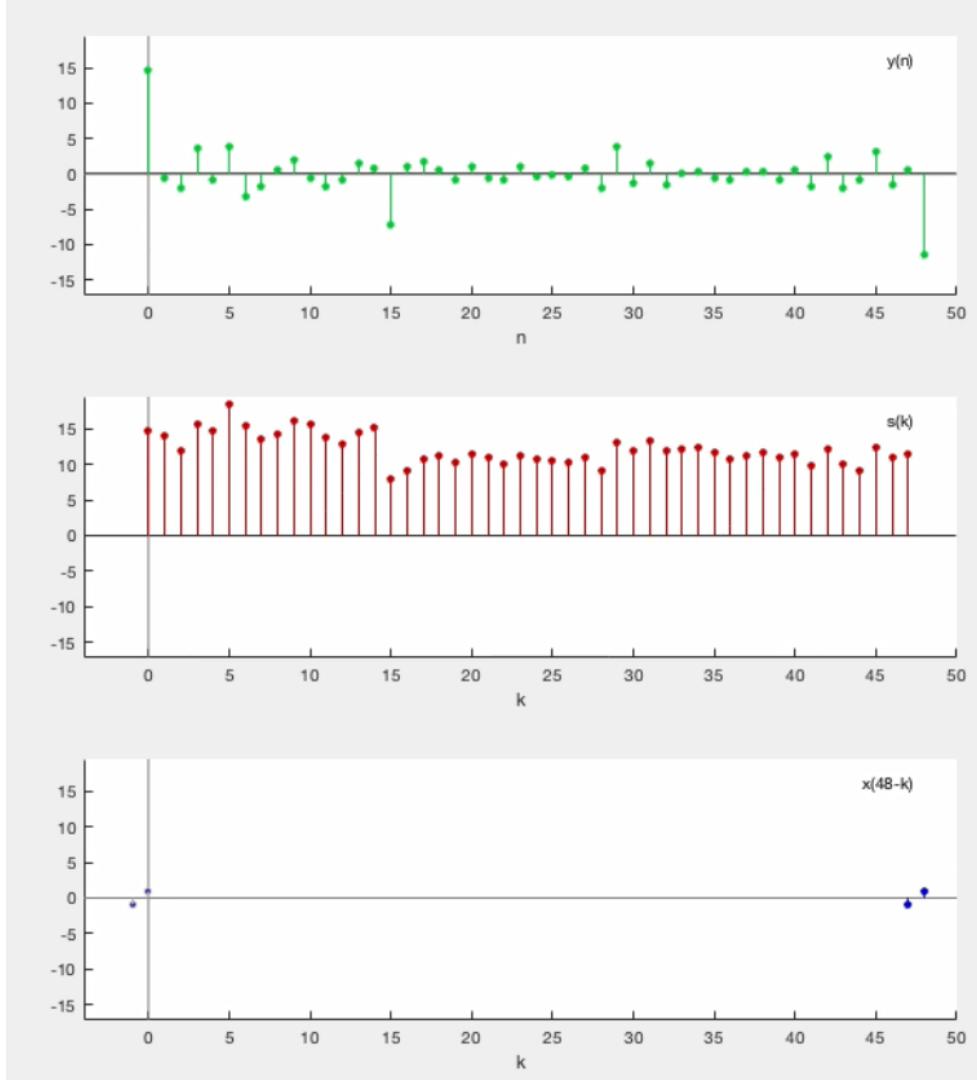
We are moving a sliding window and at every position we take the scalar product between the mask and the signal.

-1	-2	4	-1	4	-4	-1	0	2	0	-2	-1	1	1	-7	1	2	0	-1
----	----	---	----	---	----	----	---	---	---	----	----	---	---	----	---	---	---	----

Let us visualize it while in action:

We will call this operation **correlation** between two discrete signals and we can write it as

$$y[n] = \sum_{k=1}^N s[k] h[k - n]$$



Convolution differs from correlation by a reflection:

We take the “mask” $d[k]$ $\begin{matrix} 1 \\ -1 \\ 1 \end{matrix}$

and reflect it to $d[-k]$ $\begin{matrix} 1 \\ -1 \end{matrix}$

and then we shift to
 $d[-(k-n)] = d[n-k]$

$$y[n] = \sum_{k=1}^N s[k] h[n - k]$$

$\begin{matrix} 1 \\ -1 \end{matrix}$

15 14 12 16 15 18 15 14 14 16 16 14 13 14 15 8 9 11 11 10 11

The informal term for the convolution mask is “filter mask”, its values are called “filter weights” or convolution weights.

Let us do another operation on the systolic pressure signal.

Taking the local average over a window of two hours.

The averaging mask would look like
 $[1/4 \ 1/4 \ 1/4 \ 1/4]$

Does the convolution differ from correlation?

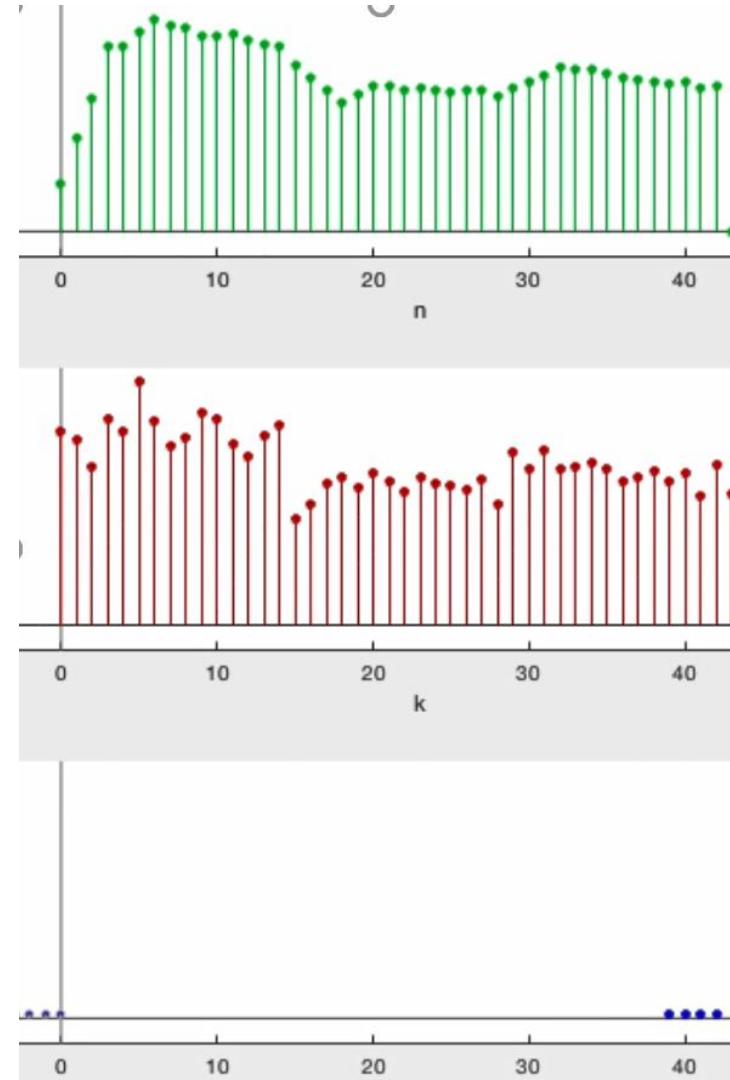
The informal term for the convolution mask is “filter mask”, its values are called “filter weights” or convolution weights.

Let us do another operation on the systolic pressure signal.

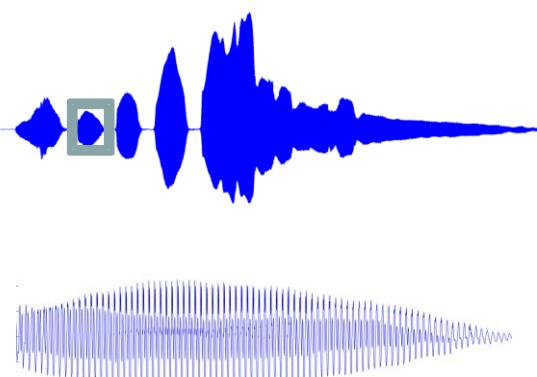
Taking the local average over a window of two hours.

The averaging mask would look like
 $[1/4 \ 1/4 \ 1/4 \ 1/4]$

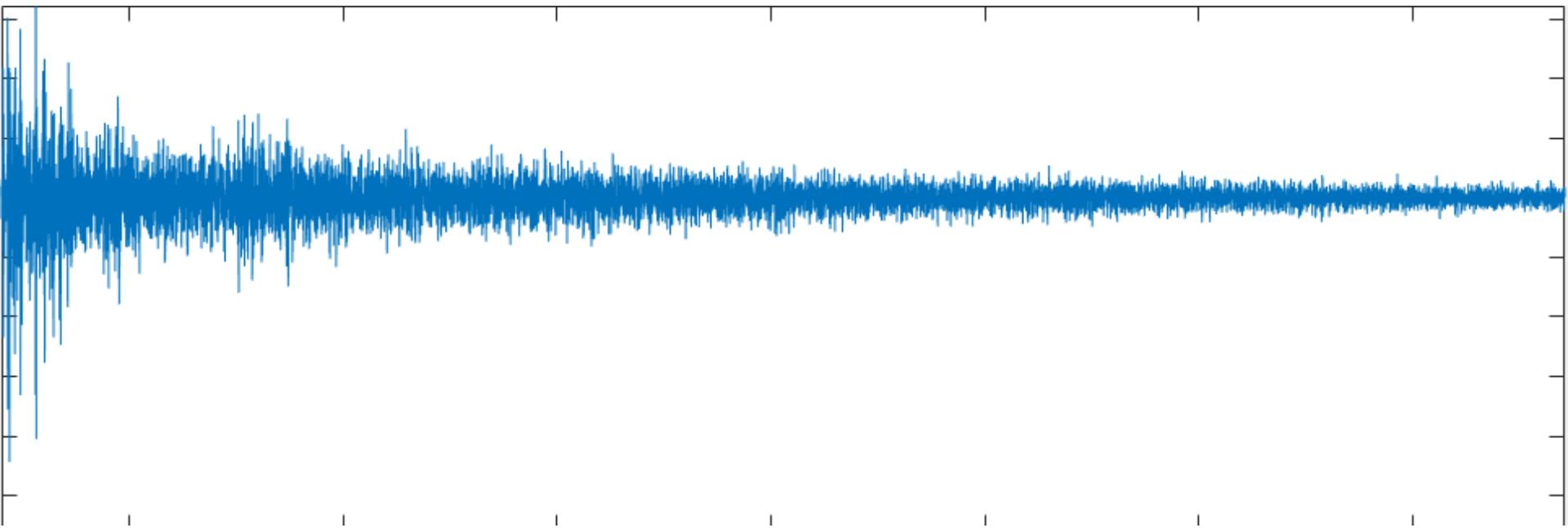
Does the convolution differ from correlation?



Let us move to the Academy of Music!

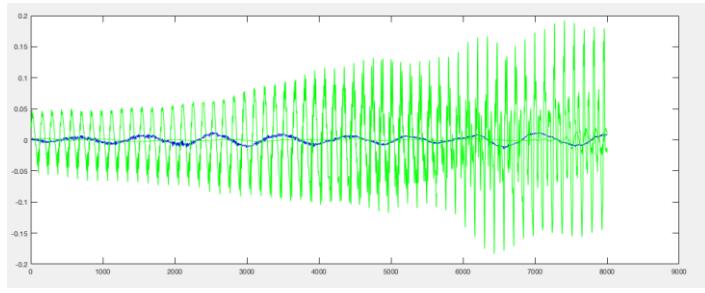


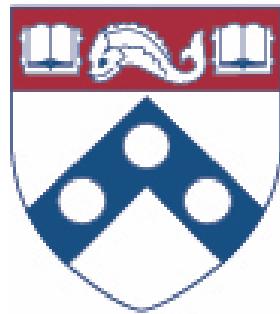
Sound is reflected creating multiple echos



Now let us take the song $s[n]$ and convolve it with the pistol $h[n]$

$$y[n] = \sum_{k=1}^N s[k] h[n - k]$$





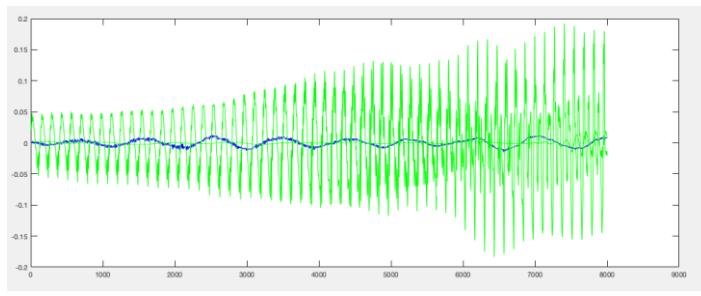
Penn
Engineering

ONLINE LEARNING

Video 2.2
Kostas Daniilidis

In the last lecture the song $s[n]$ and convolve it with the pistol $h[n]$

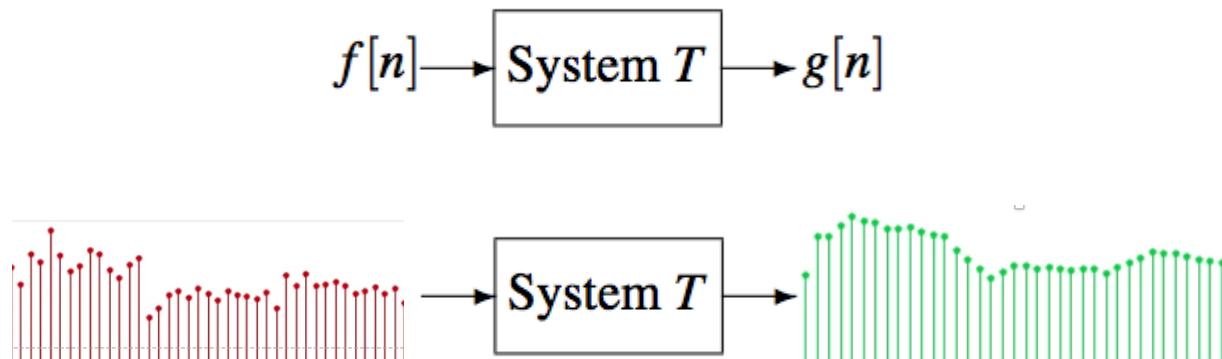
$$y[n] = \sum_{k=1}^N s[k] h[n - k]$$



The systems' view

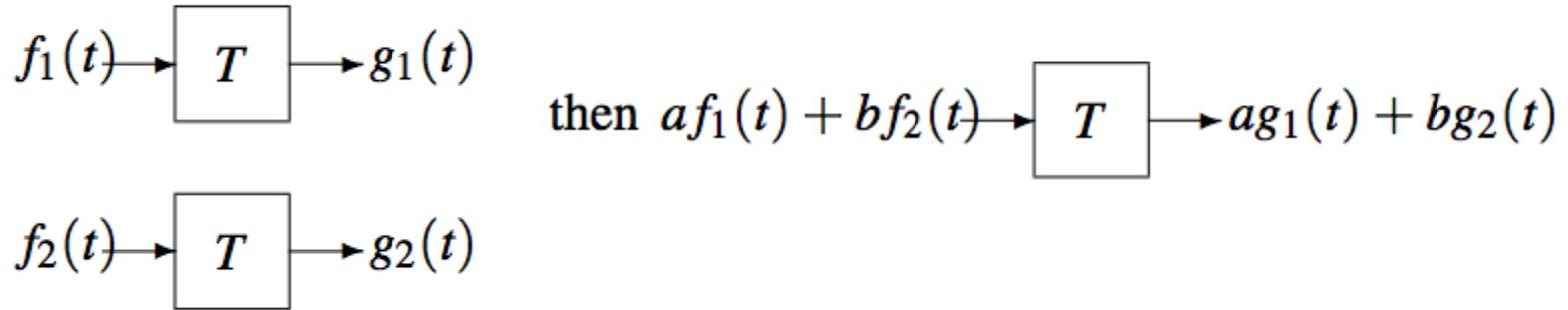
Consider it as a system with input and output. In the previous examples, input might have been

- Audio like a song
- Blood pressure measurements
- In general any 1D signal



Linear System

- We will say that a system is linear if



Example

$T\{f\}(t) = f(t) - f(t-1)$ is linear:

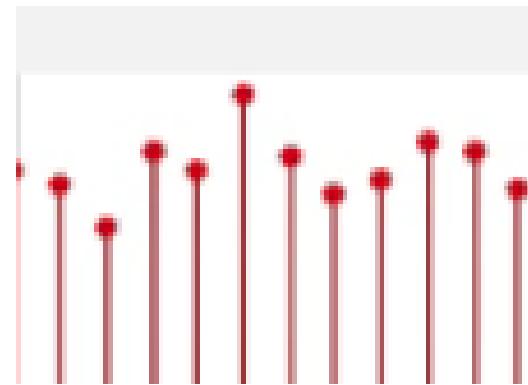
$$g_1(t) = f_1(t) - f_1(t-1)$$

$$g_2(t) = f_2(t) - f_2(t-1)$$

$$\begin{aligned} T\{af_1 + bf_2\}(t) &= [af_1(t) + bf_2(t)] - [af_1(t-1) + bf_2(t-1)] \\ &= a(f_1(t) - f_1(t-1)) + b(f_2(t) - f_2(t-1)) \\ &= aT\{f_1\}(t) + bT\{f_2\}(t) \end{aligned}$$

What is non-linear?

$$g(t) = \max(f(t), f(t-1), f(t-2))$$



Shift-invariant (or time-invariant) system



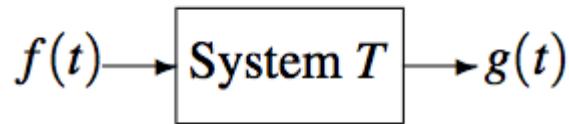
Examples:

- $T\{f\}(t) = f(t) - f(t - 1)$ is shift-invariant.

$$\begin{aligned} T\{f(t - t_0)\} &= f(t - t_0) - f(t - t_0 - 1) \\ &= g(t - t_0) \end{aligned}$$

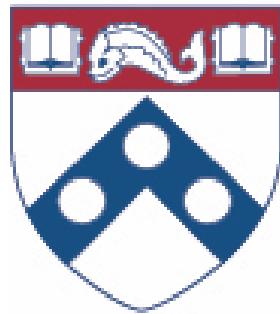
- $g(t) = tf(t)$ is not shift-invariant.

Is there a formula for describing
linear shift-invariant systems?



Yes, the convolution! Discrete or continuous

$$g(t) = \int_{-\infty}^{\infty} f(t')h(t-t')dt' \quad g[n] = \sum_{k=-\infty}^{\infty} f[k] h[n-k]$$



Penn
Engineering

ONLINE LEARNING

Video 2.3
Kostas Daniilidis

What is a filter?

Last lecture's main result: Linear shift-invariant systems can be written as convolutions!

$$g(t) = \int_{-\infty}^{+\infty} f(t')h(t-t')dt' = f(t) \star h(t)$$

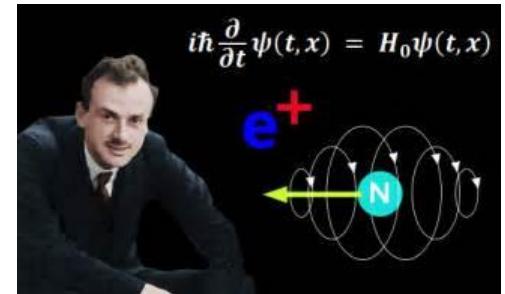
But what exactly is this $h(t)$ or $h[n]$ that we call filter?

Box function

$$\text{rect}(t) = \begin{cases} 1, & |t| \leq 1/2 \\ 0, & \text{otherwise} \end{cases}$$

$$\text{rect}(t/a) = \begin{cases} 1/a, & |t| \leq a/2 \\ 0, & \text{otherwise} \end{cases}$$

Dirac function



$$\delta(t) \doteq \lim_{a \rightarrow 0} \frac{1}{a} \operatorname{rect}(t/a)$$

It follows from definition of the Dirac function that

$$\delta(t) = 0 \quad \text{for all } t \neq 0$$

and

$$\int_{-\infty}^{+\infty} \delta(t) dt = 1$$

Absorption property

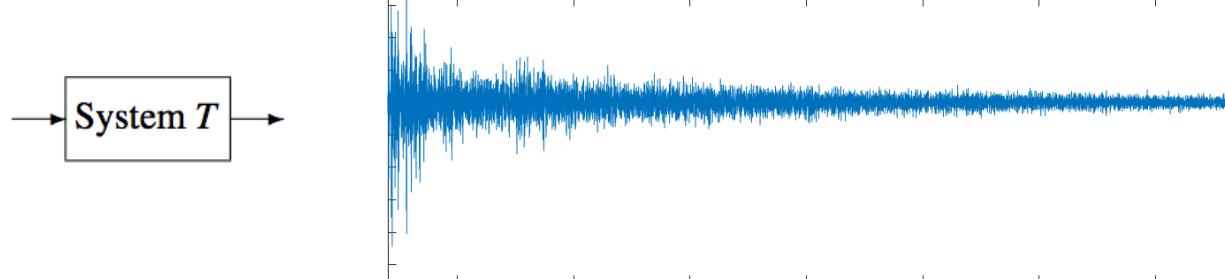
$$\int_{-\infty}^{+\infty} \delta(t) f(t) dt = f(0)$$

or in a more general form:

$$\int_{-\infty}^{+\infty} \delta(t - t_0) f(t) dt = f(t_0)$$

What happens when the input to an LSI is a Dirac?

$$\int_{-\infty}^{+\infty} \delta(t') h(t - t') dt' = h(t)$$



A filter $h(t)$ or $h[n]$ is the response of the system to the Dirac impulse.

Example. Let an LSI system with impulse response defined as

$$h(t) = (1/2)(\delta(t+1) - \delta(t-1))$$

$$\begin{aligned} g(t) &= \int_{-\infty}^{+\infty} f(t')h(t-t')dt' \\ &= \int_{-\infty}^{+\infty} f(t-u)h(u)du \\ &= \int_{-\infty}^{+\infty} f(t-u)(1/2)(\delta(u+1) - \delta(u-1))du \\ &= (1/2) \int_{-\infty}^{+\infty} f(t-u)\delta(u+1)du - (1/2) \int_{-\infty}^{+\infty} f(t-u)\delta(u-1)du \\ &= (1/2)(f(t+1) - f(t-1)) \end{aligned}$$

Dirac description of sampling

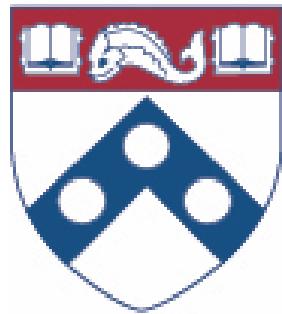
- What do we get discrete signals out of analog?

Sampling is multiplication with the comb

Sampling is a multiplication of the signal by the comb function $\text{III}_T(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$. T is the sampling interval:

$$f_s(t) = f(t) \sum_{n=-\infty}^{\infty} \delta(t - nT)$$

(S for “sampled”) After sampling, we forget about the T : we just obtain a sequence of numbers, that we note $f_s[k]$.

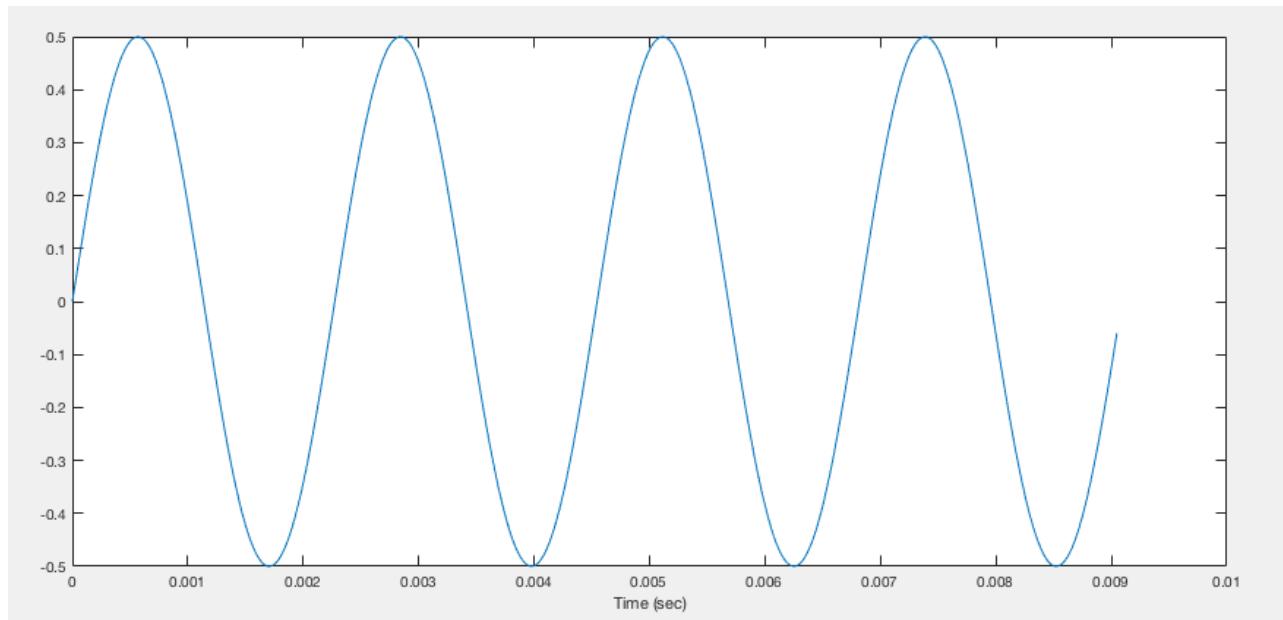


Penn
Engineering

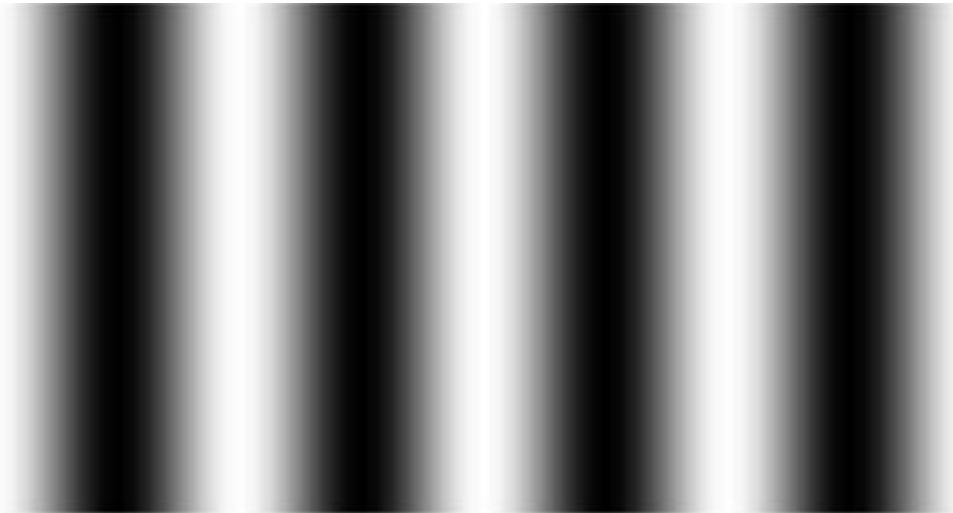
ONLINE LEARNING

Video 2.4
Kostas Daniilidis

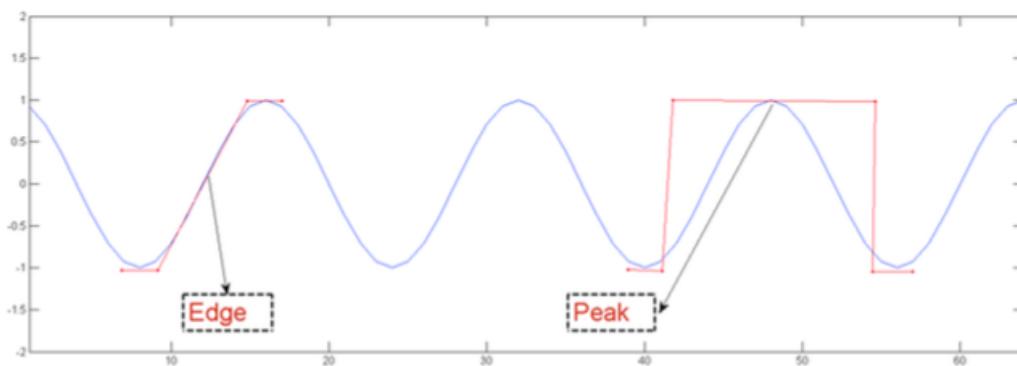
Back to music: sine wave 440Hz



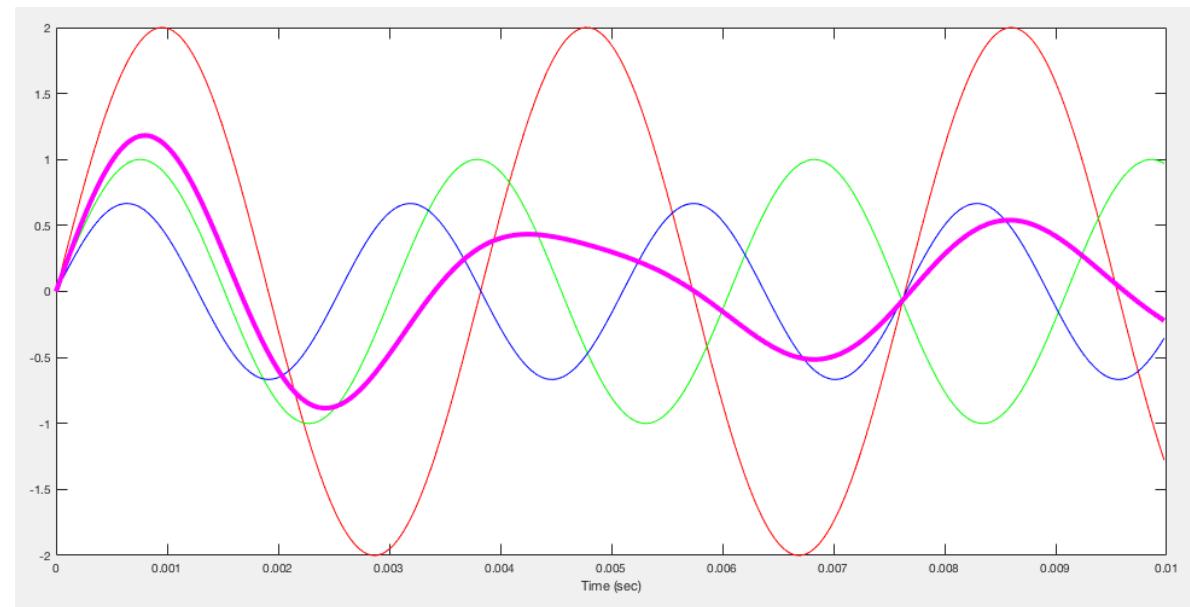
Or images....



$$2D \text{ Signal } f(x, y) = \cos\left(\frac{2\pi x}{16}\right)$$

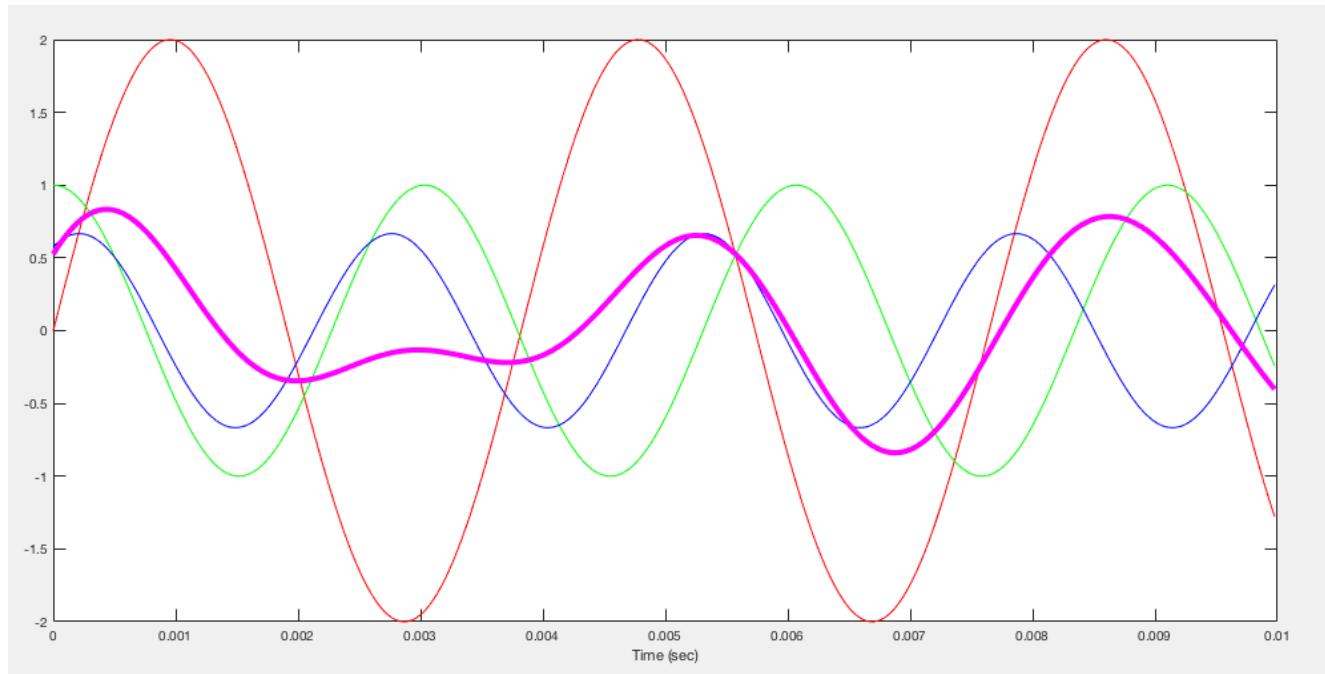


Play three notes together: C-major **chord** consists of a superposition of C (262Hz), E(330HZ) and G(392Hz)



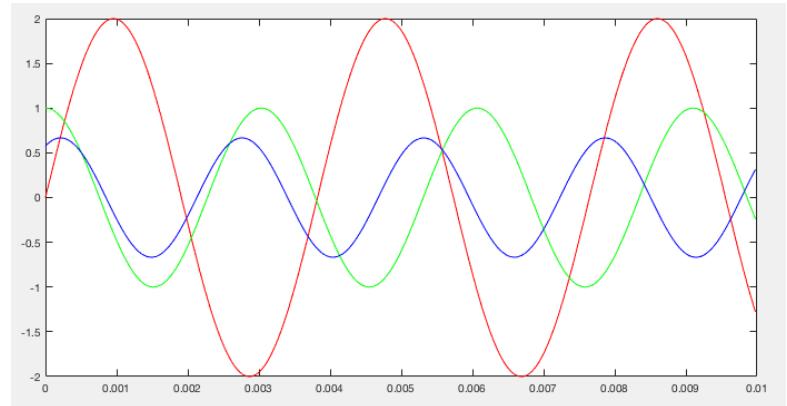
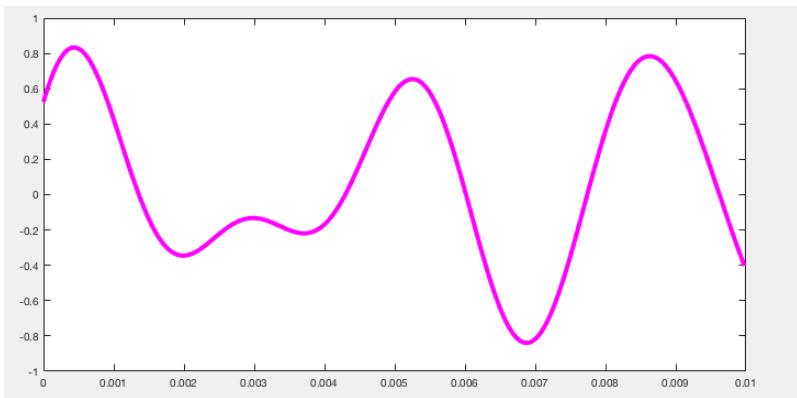
$$A_1 \sin(2\pi f_1 t) + A_2 \sin(2\pi f_2 t) + A_3 \sin(2\pi f_3 t)$$

We could also change the phases:

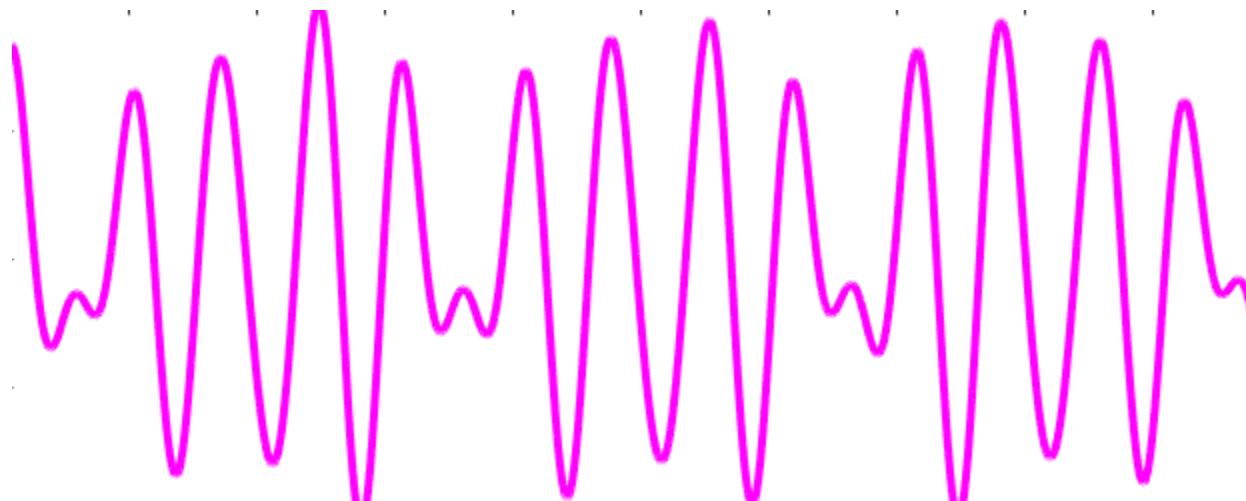


$$A_1 \sin(2\pi f_1 t) + A_2 \sin(2\pi f_2 t + \frac{\pi}{2}) + A_3 \sin(2\pi f_3 t + \frac{\pi}{3})$$

Can we solve the inverse problem?



Yes, we can decompose a function into “sinusoids” if it is periodic: Fourier series



$$f(t) = \sum_{n=-\infty}^{\infty} F[n] e^{j2\pi \frac{n}{T} t}$$

The complex Fourier coefficients $F[n]$ can be found with the Fourier Transform

$$F[n] = \int_{-T/2}^{T/2} f(t) e^{-j2\pi \frac{n}{T} t} dt$$

$$j^2 = -1$$

Quick reminder on complex numbers:

- $a + jb \in \mathbb{C}, j^2 = -1$
- $e^{j\omega t} = \cos(\omega t) + j \sin(\omega t).$
- $a + jb = re^{j\theta}$ with $r = \sqrt{a^2 + b^2},$
 $\theta = \text{atan2}(b, a)$

Why we need complex numbers?

To account for the phase

$$f(t) = \sum_{n=-\infty}^{\infty} F[n] e^{j2\pi \frac{n}{T} t}$$

or

$$f(t) = \sum_{n=-\infty}^{\infty} A_k \left(\cos\left(2\pi \frac{n}{T} t\right) + j \sin\left(2\pi \frac{n}{T} t\right) \right)$$

Let us agree on a notation:

$$F[n] = \int_{-T/2}^{T/2} f(t) e^{-j2\pi \frac{n}{T} t} dt$$

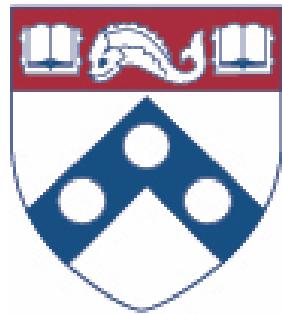
Definition of the Fourier Transform:

$$f(t) \xrightarrow{\bullet} F(\omega) = \mathcal{F}\{f(t)\}$$

$$F : \mathbb{R} \rightarrow \mathbb{C}$$

$$F(\omega) = \int_{-\infty}^{\infty} f(t) e^{-j\omega t} dt,$$

where ω denotes the frequency. This definition is sometimes called non-unitary Fourier transform, with angular frequency (ω is referred to as angular frequency, and $s = \frac{n}{T}$ is the ordinary frequency in Hz such that $\omega = 2\pi s$).



Penn
Engineering

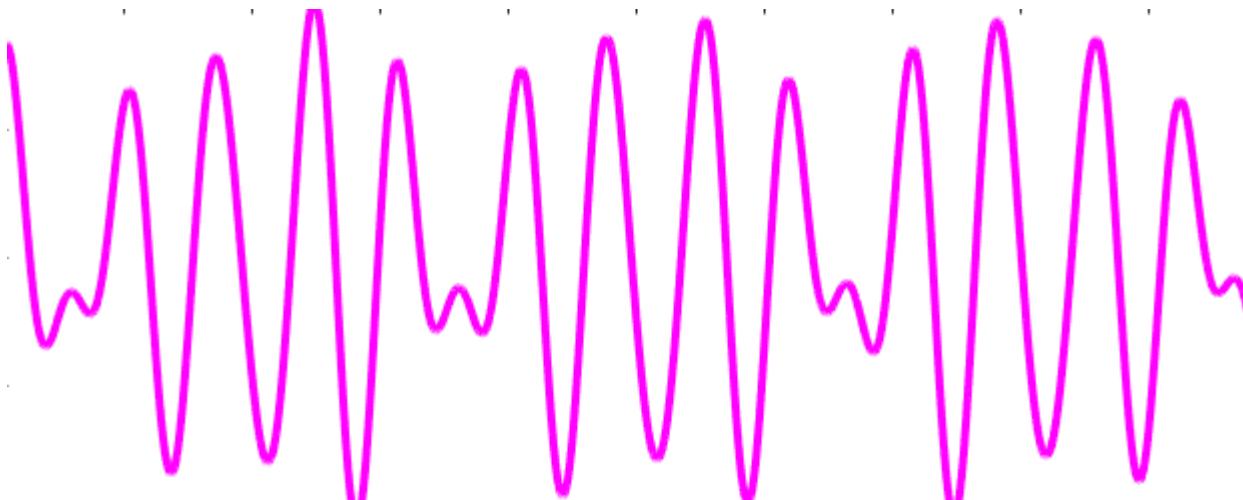
ONLINE LEARNING

Video 2.5
Kostas Daniilidis

Some facts about the Fourier Transform

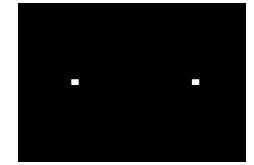
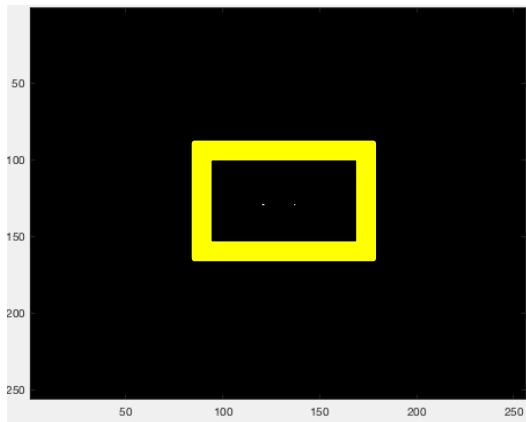
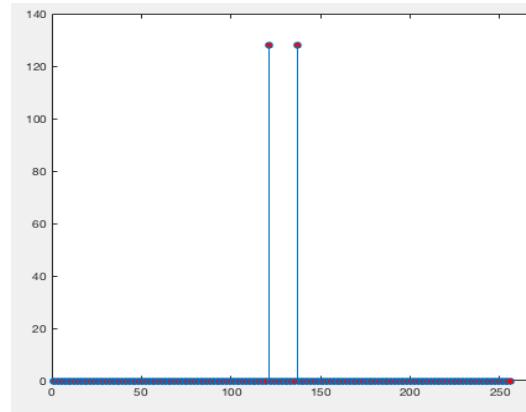
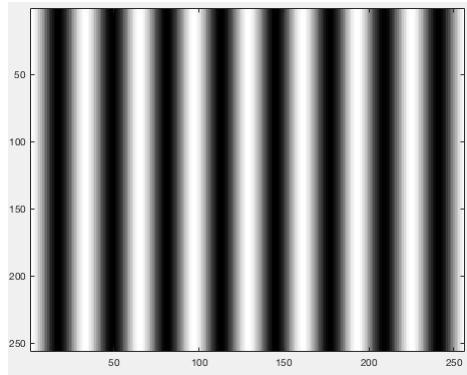
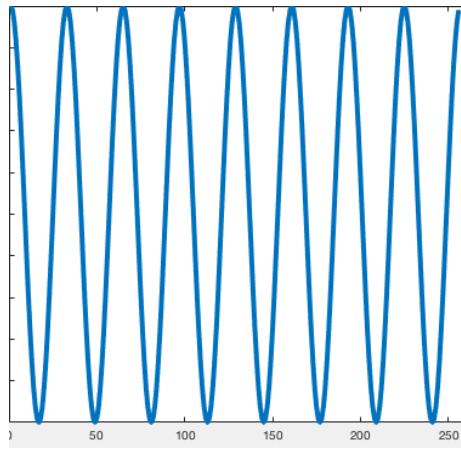


We learnt about the Fourier series of periodic signals

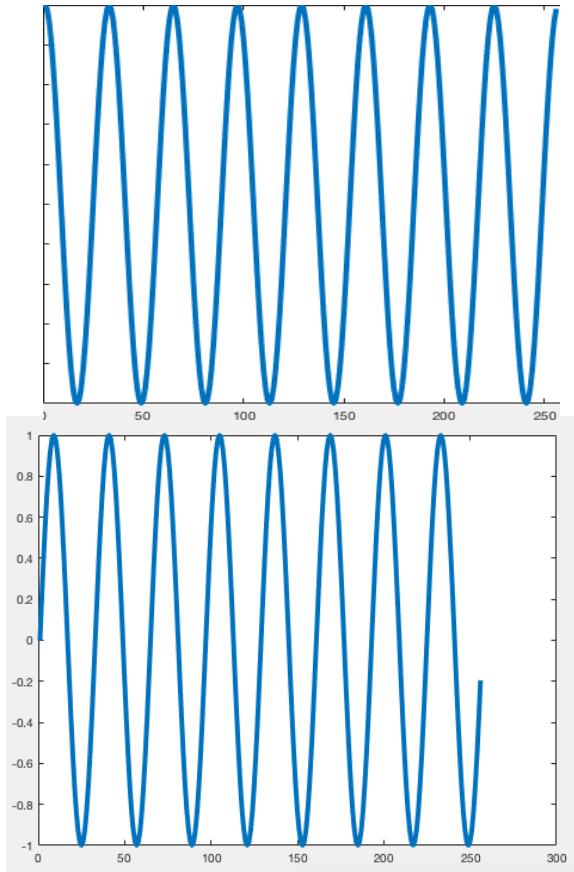


$$f(t) = \sum_{n=-\infty}^{\infty} F[n] e^{j2\pi \frac{n}{T} t}$$

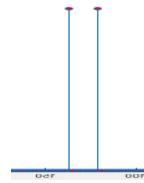
Fourier Transform of a Cosine



Do sine and cosine have the same Fourier transform



Only in the magnitude $|F[n]|$!



They differ in the phase !

$$\cos(\omega_0 t) \quad \text{---} \bullet \quad 2\pi \cdot \frac{1}{2} (\delta(\omega - \omega_0) + \delta(\omega + \omega_0))$$
$$\sin(\omega_0 t) \quad \text{---} \bullet \quad 2\pi \cdot \frac{1}{2j} (\delta(\omega - \omega_0) - \delta(\omega + \omega_0))$$

Fourier of the harmonic exponential

$$e^{j\omega_0 t} \longleftrightarrow 2\pi\delta(\omega - \omega_0)$$

Two phase-related facts

Shift theorem

$$f(t - t_0) \xrightarrow{\text{---}} F(\omega)e^{-j\omega t_0}$$

Example: for $f(t)$ even, $f(t - \frac{T}{2}) \xrightarrow{\text{---}} F(\omega)e^{-j\omega\frac{T}{2}}$, where $F(\omega)$ is real.

Modulation theorem

$$f(t)e^{j\omega_0 t} \xrightarrow{\text{---}} F(\omega - \omega_0)$$

Multiplying by a complex exponential causes a shift in the frequency domain.

Let us look at some more Fourier transforms: **Dirac**

Recall the absorption property

$$\int_{-\infty}^{\infty} \delta(t - t_0) f(t) dt = f(t_0)$$

Then we have:

$$\int_{-\infty}^{\infty} \delta(t) e^{-j\omega t} dt = 1$$

We will remember the two following results:

$$\begin{aligned} \delta(t) &\xrightarrow{\text{↔}} 1 \\ 1 &\xrightarrow{\text{↔}} 2\pi\delta(t) \quad (\text{DC-component}) \end{aligned}$$

Fourier of the comb function $\text{III}(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$:

$$\sum_{n=-\infty}^{\infty} \delta(t - nT) \circ\bullet \frac{1}{|T|} \sum_{n=-\infty}^{\infty} \delta(s - \frac{n}{T})$$

(or $\frac{2\pi}{|T|} \sum_{n=-\infty}^{\infty} \delta(\omega - \frac{2\pi n}{T})$ when using the non-unitary Fourier Transform with angular frequency).

Step and sgn functions

Preliminary: The step function is defined such that $\frac{d}{dt}u(t) = \delta(t)$

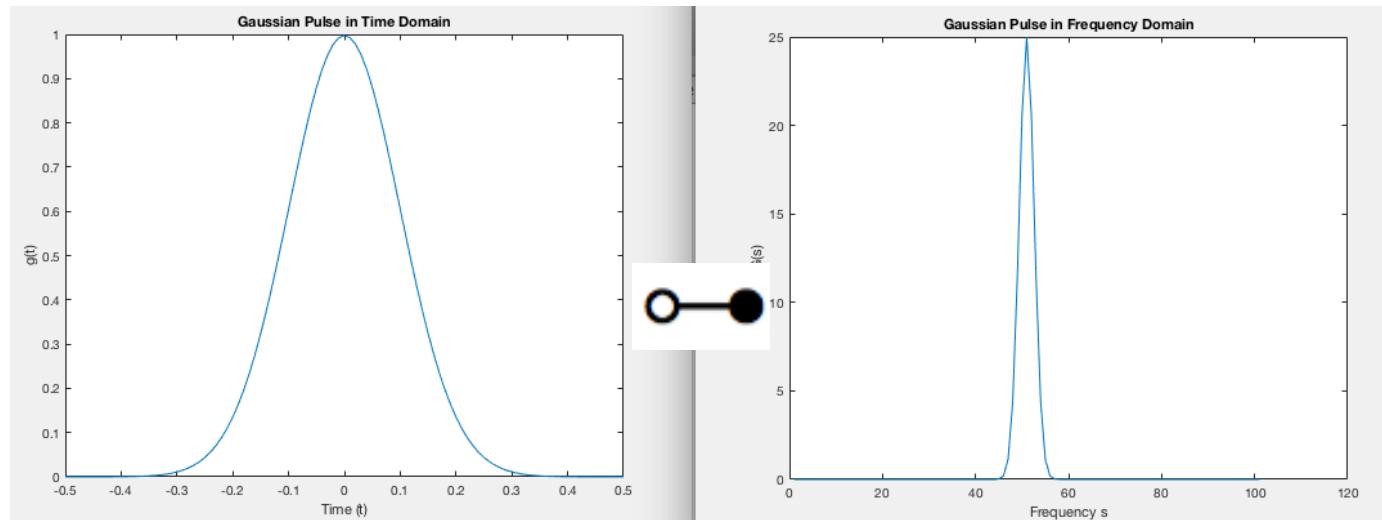
The sign function comes handy: $\text{sgn}(t) = 2u(t) - 1$, $u(0) = 1/2$, or equivalently $u(t) = \frac{1}{2} + \frac{1}{2}\text{sgn}(t)$.

We have the following straightforward Fourier transforms:

$$u(t) \circledast \mathcal{U}(\omega) = \frac{1}{2}\delta(\omega) + \frac{1}{j\omega}$$

$$\frac{d}{dt}\text{sgn}(t) = 2\delta(t) \circledast j\omega\mathcal{F}(\text{sgn}(t)) = 2.$$

Fourier Transform of the Gaussian Function



$$f(t) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{t^2}{2\sigma^2}}$$



$$F(s) = \frac{1}{\sigma \sqrt{2\pi}} e^{-2\pi^2\sigma^2 s^2}$$

Convolution Theorem

Convolution in frequency means multiplication in time or space

$$f(t) \rightarrow \boxed{h(t)} \rightarrow g(t) = \int_{-\infty}^{\infty} g(t')h(t-t')dt' = f(t) * h(t)$$

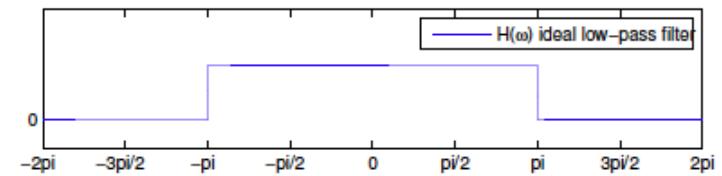
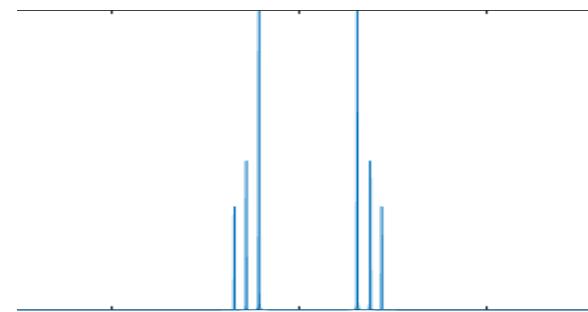
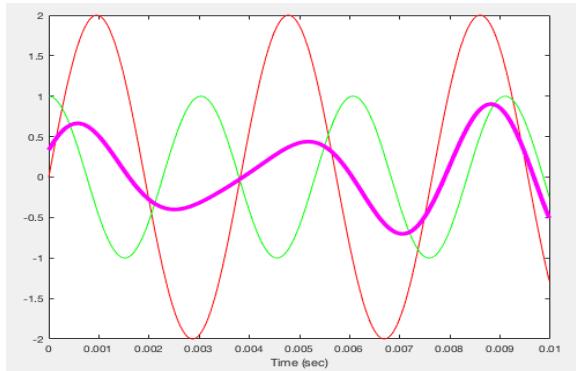
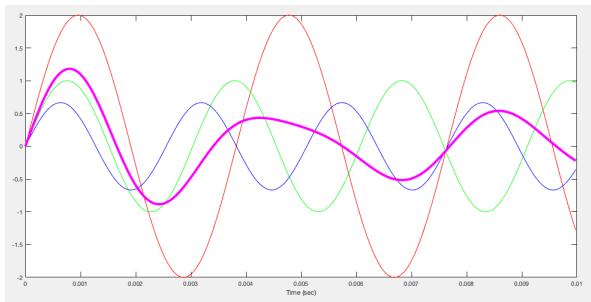
\bullet \bullet \bullet

$$F(\omega) \qquad \qquad H(\omega) \qquad \qquad G(\omega)?$$

$$f(t) * h(t) \circ \bullet F(\omega)H(\omega)$$

Convolution Theorem: Low Pass Filtering

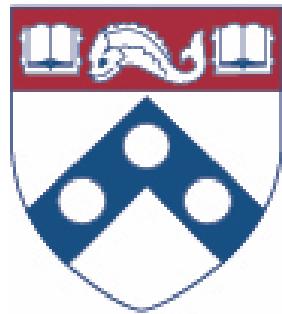
$$f(t) * h(t) \rightleftharpoons F(\omega)H(\omega)$$



Convolution Theorem II

Convolution in frequency means multiplication in time or space

$$f(t)h(t) \longleftrightarrow F(\omega) * H(\omega)$$



Penn
Engineering

ONLINE LEARNING

Video 2.6
Kostas Danillidis

Sampling: Converting analog time or space to digital



Optical Illusion: Wagon Wheel - YouTube

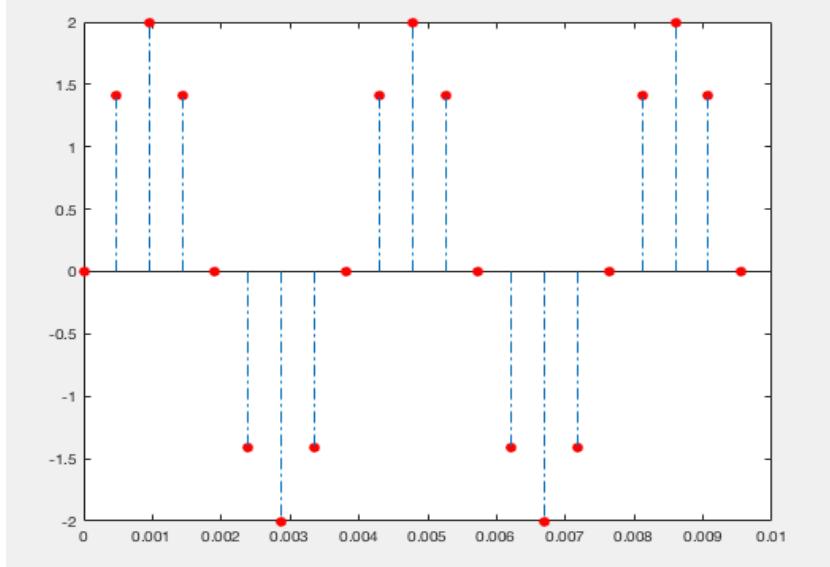
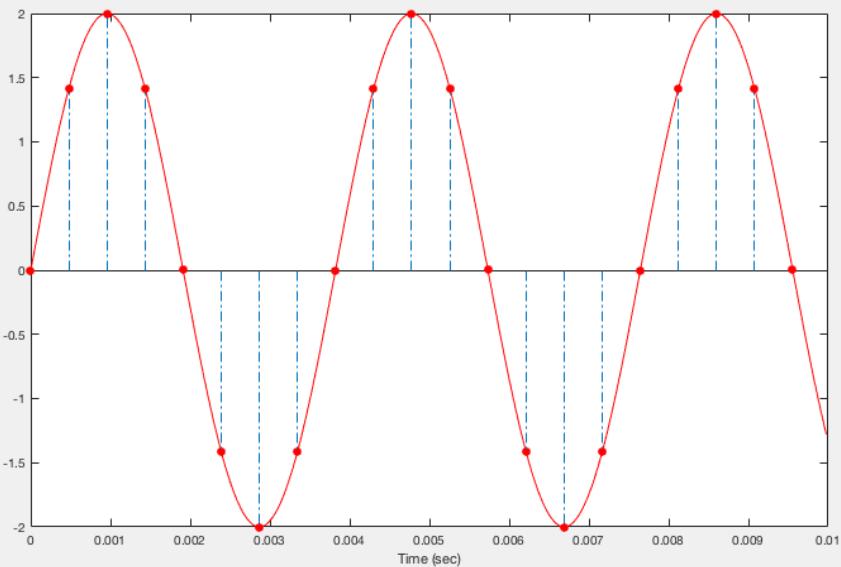


<https://www.youtube.com/watch?v=tMk14AEpRd8>

Jan 14, 2016 - Uploaded by Cognic

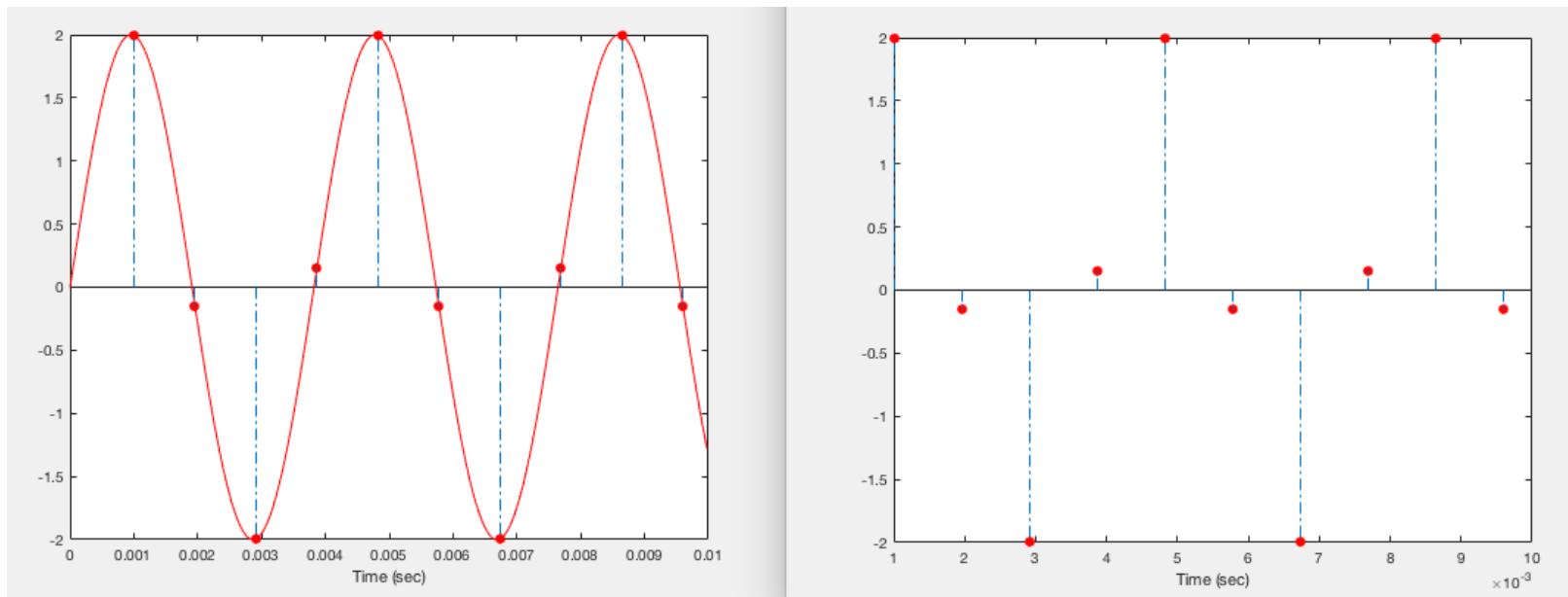
Optical Illusions and Mind Tricks to make people understand something. We see and feel and smell and touch ...

Back to the sine-wave: sampling interval 1/8 of wavelength

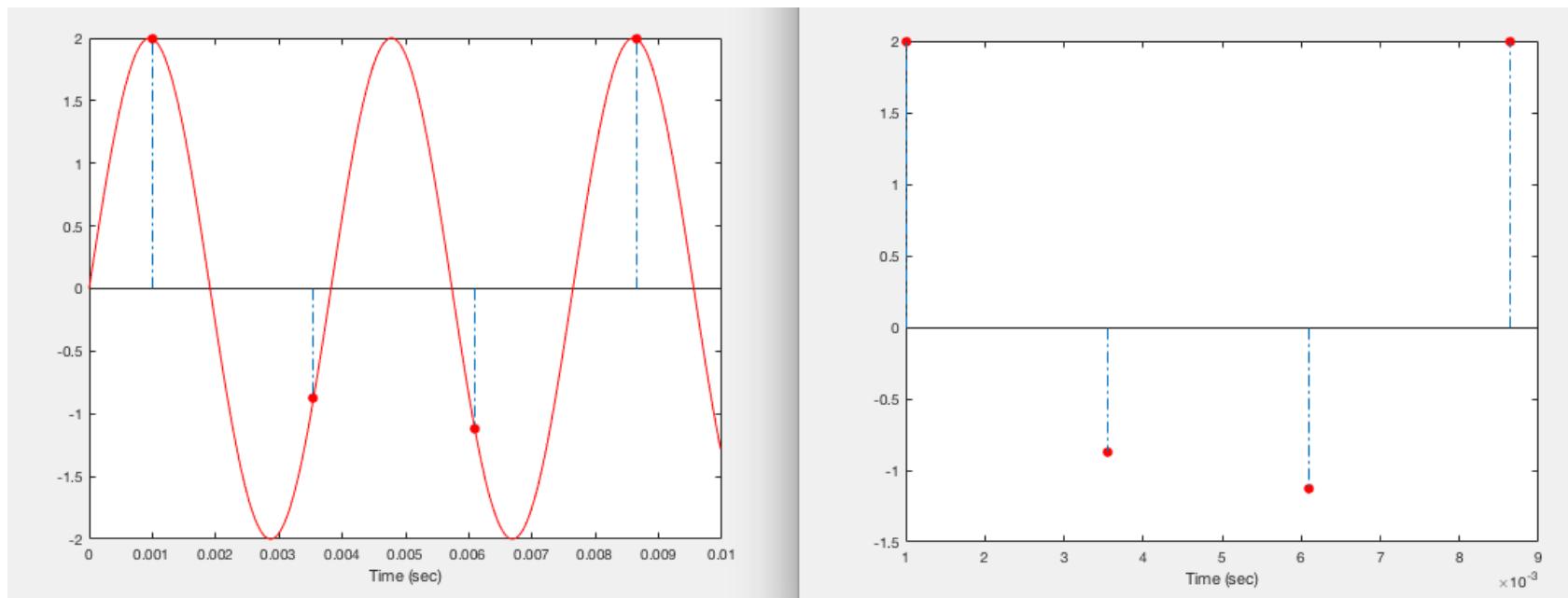


For note A (440 Hz), sampling frequency of 44.1kHz means 100 samples per wavelength !

Increasing sampling interval to 1/4 of wavelength



Increasing sampling interval to more than $\frac{1}{2}$ of wavelength



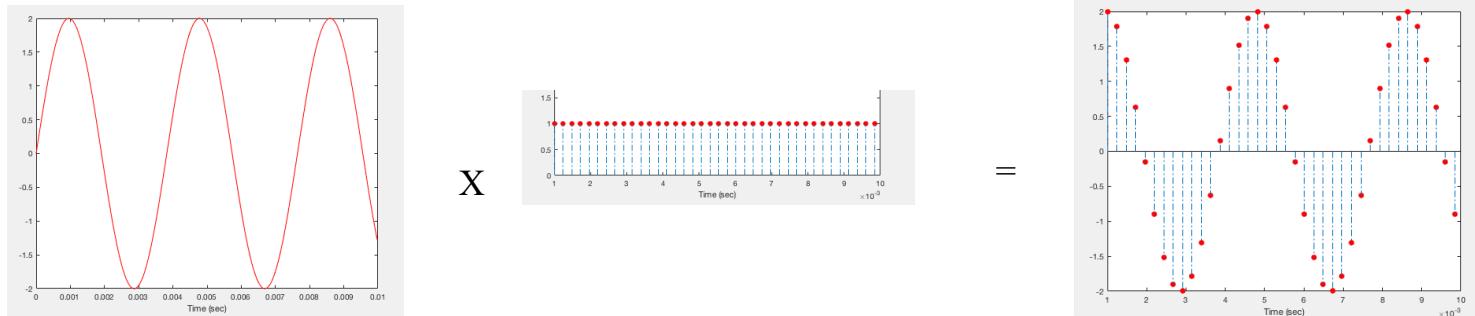
We cannot reconstruct the original wave!

What does sampling mean?

Sampling is a multiplication of the signal by the comb function $\text{III}_T(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$. T is the sampling interval:

$$f_s(t) = f(t) \sum_{n=-\infty}^{\infty} \delta(t - nT)$$

(S for “sampled”) After sampling, we forget about the T : we just obtain a sequence of numbers, that we note $f_s[k]$.



Multiplication in time means convolution in frequency!

Question 1: what is the Fourier transform of the comb ?

$$\sum_{n=-\infty}^{\infty} \delta(t - nT) \circledast \sum_{n=-\infty}^{\infty} \delta(\omega - \frac{2\pi n}{T}) = \sum_{n=-\infty}^{\infty} \delta(s - \frac{n}{T})$$

A comb with inverse sampling interval!

Question 2: What means convolution with a comb?

$$F_S(\omega) = F(\omega) * \sum_{n=-\infty}^{\infty} \delta(\omega - \frac{2\pi n}{T})$$

It is creating copies of the "spectrum" at inverse time intervals!

Question 3: Can we reconstruct the original spectrum from the “xeroxed” spectrum?

Only if the copies do not overlap ! **Aliasing!**



Shannon-Nyquist Theorem



An analog signal can be reconstructed from sampling only if the sampling frequency is at least twice the maximum frequency; equivalently, if the sampling interval is less equal the minimum half the minimum wavelength.

Obviously the analog signal has to be **bandlimited**!

How can the reconstruction be realized?

Multiplying the spectrum of the sampled signal with the rectangle-function

$$\Pi(t) = \begin{cases} \frac{1}{2\pi} & -\pi \leq \omega \leq \pi \\ 0 & \text{anywhere else} \end{cases}$$

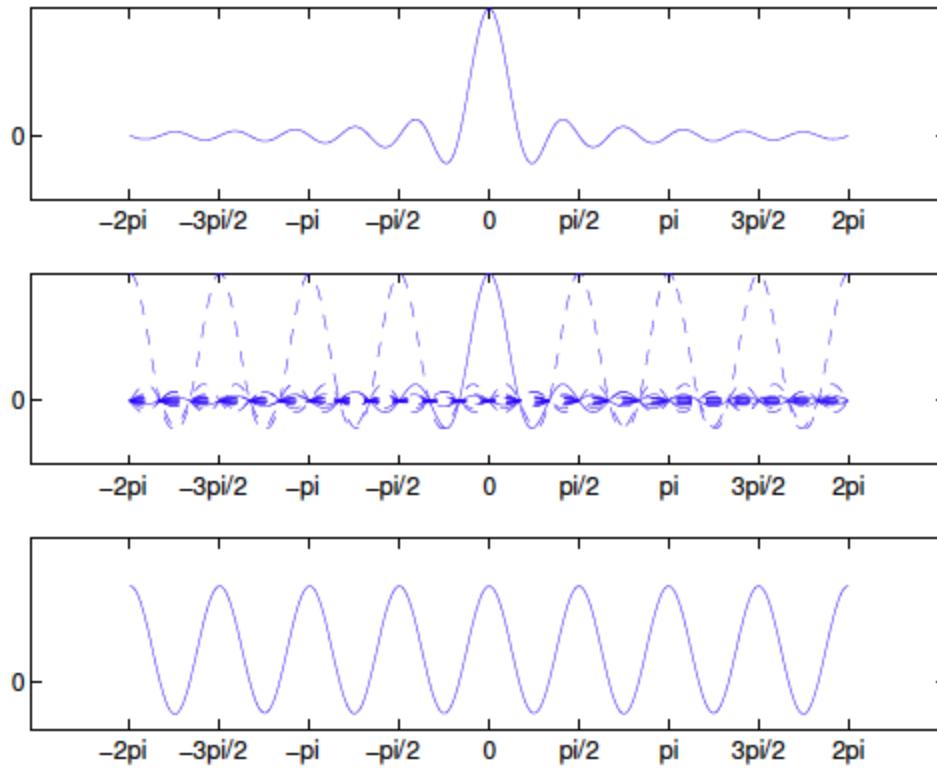
Which function has the box as Fourier-transform?

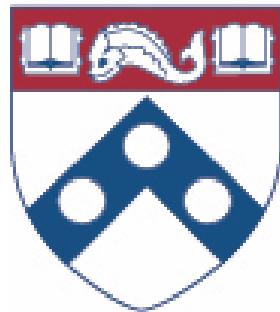
$$\begin{aligned}\mathcal{F}^{-1}\{\text{rect}(\omega)\} &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \text{rect}(\omega) e^{j\omega t} d\omega & \frac{1}{2\pi} \text{sinc}\left(\frac{t}{2}\right) \circ \bullet \text{rect}(\omega) \\ &= \frac{1}{2\pi} \int_{-\frac{1}{2}}^{\frac{1}{2}} e^{j\omega t} d\omega & \frac{1}{2\pi} \text{sinc}(\pi t) \circ \bullet \frac{1}{2\pi} \text{rect}\left(\frac{\omega}{2\pi}\right) \\ &= \frac{1}{2\pi} \frac{1}{jt} [e^{j\omega t}]_{-\frac{1}{2}}^{\frac{1}{2}} \\ &= \frac{1}{2\pi} \text{sinc}\left(\frac{t}{2}\right)\end{aligned}$$

Multiplication in frequency means convolution in time (space)

Reconstruction means convolution of the sampled signal with sinc

$$\begin{aligned}f_{reconstr}(t) &= f(t) \sum_{n=-\infty}^{\infty} \delta(t - nT) * \frac{1}{2\pi} \text{sinc}(\pi t) \\&= \sum_{n=-\infty}^{\infty} f[nT] \frac{1}{2\pi} \text{sinc}(\pi(t - nT))\end{aligned}$$





Penn
Engineering

ONLINE LEARNING

Video 2.7
Kostas Daniilidis

Discrete time signals have still continuous Fourier transforms (CFT)!

$$f[n] \leftrightarrow \sum_{n=0}^{L-1} f[n] e^{-j\omega n}$$

Discrete time signals have continuous **and periodic** Fourier transforms **with period 2π !**

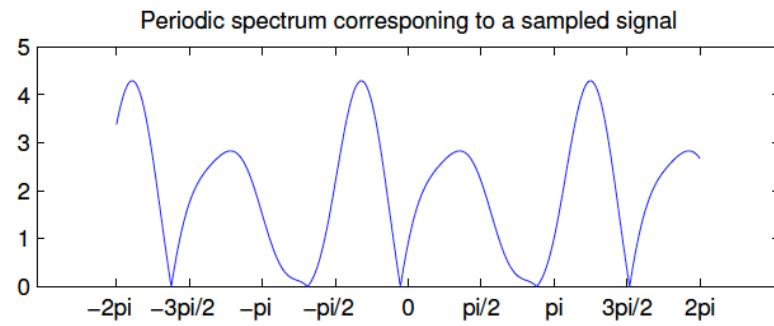
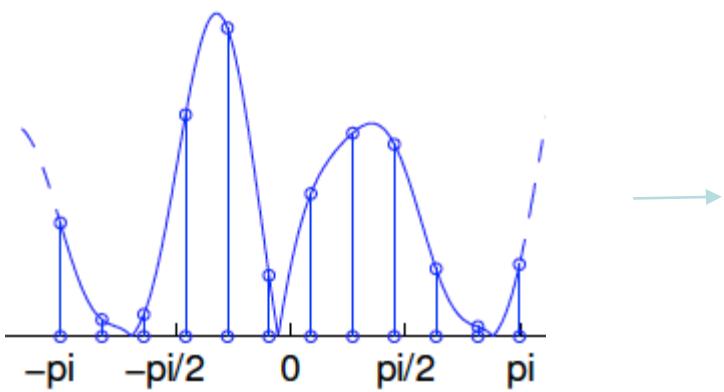
That's why we consider only the interval $(-\pi, \pi]$ when we talk about the CFT of a discrete signal!

Example. Let $f[n]$ a discrete time signal and $h[n]$ a discrete time signal defined as:

$$h[n] = \begin{cases} 1, & n = 0 \\ -1, & n = 1 \\ 0, & \text{elsewhere} \end{cases}$$

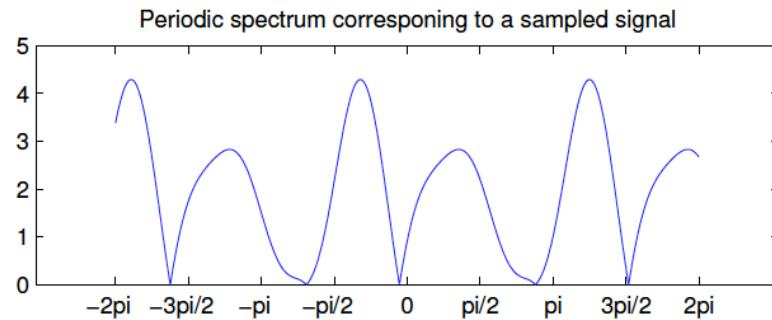
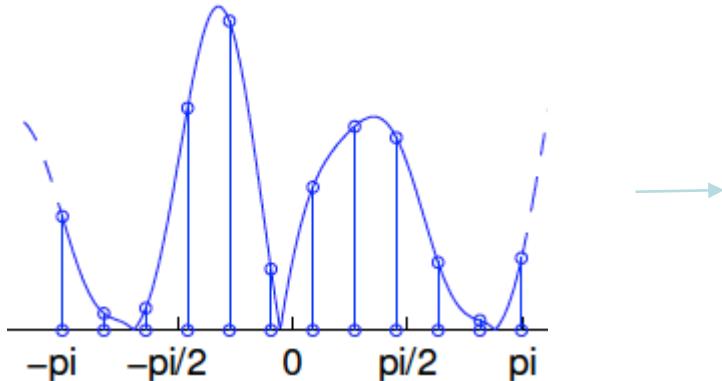
$$H(\omega) = \sum_n h[n]e^{-j\omega n} = h[0]e^{-j\omega \cdot 0} + h[1]e^{-j\omega \cdot 1} = 1 - e^{-j\omega}$$

What does discrete frequency mean?



What does sampling in the frequency domain correspond to in the time/space domain? It corresponds to a **convolution**:

- **FREQUENCY DOMAIN:** multiplication with $\sum \delta(\omega - \frac{2\pi k}{L})$
- **TIME DOMAIN:** convolution with $\sum \delta[n - kL]$, equivalent to **replication of the signal at multiples of its length**.



Why is the DFT of a discrete time finite signal same as the CFT?

Because sampling the frequency domain only means replicating the original signal.

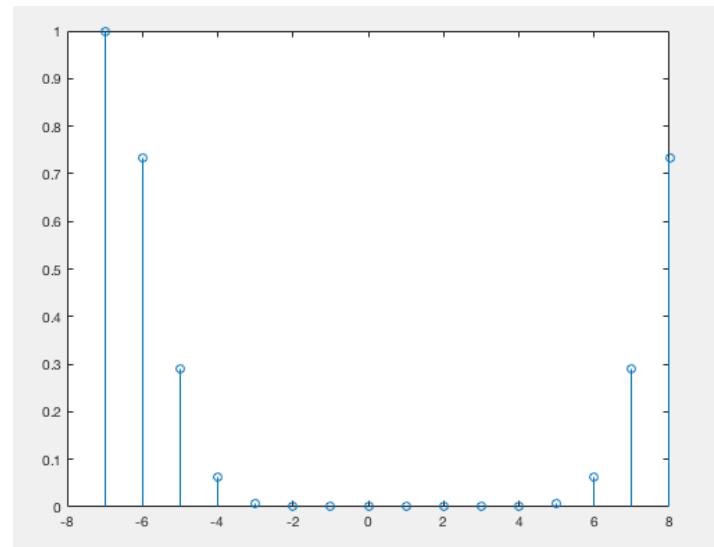
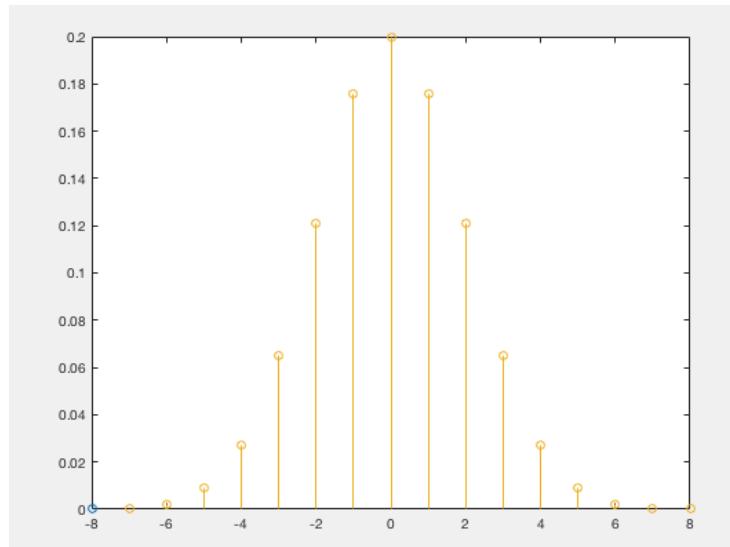
Some M_L

```
support = 16;
sigma = 2;
d = -floor(support/2)+1:floor(support/2);
g = (1/(sigma*sqrt(2*pi)))*exp(-d.^2/(2*sigma^2));

figure(1)
stem(d,g);|
```

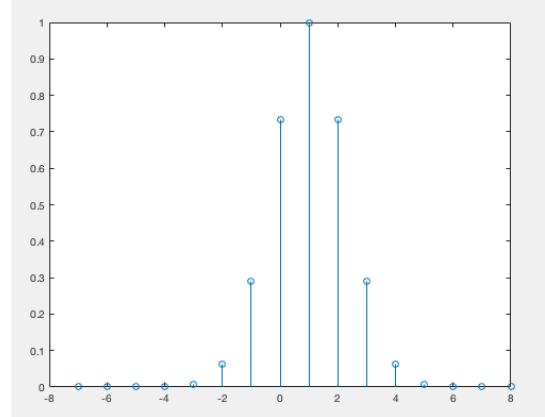
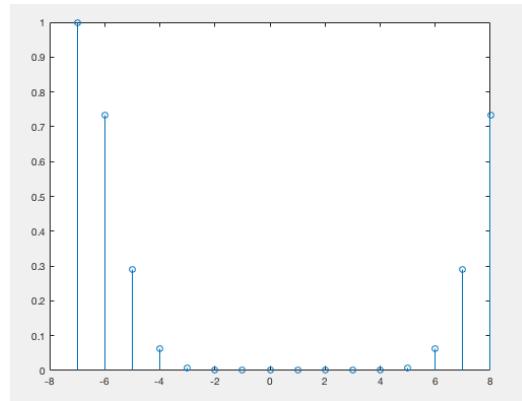
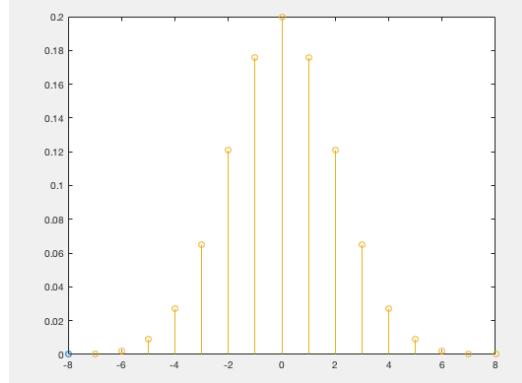


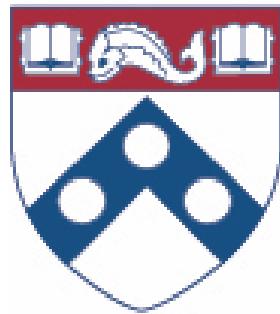
```
figure;
stem(d,abs(fft(g)));
```



fftshift...

fftshift(fft(f)) performs a shift by $\frac{L}{2}$
(modulation of the signal by $e^{\frac{-j2\pi L/2}{L}} = e^{-j\pi} = -1$)





Penn
Engineering

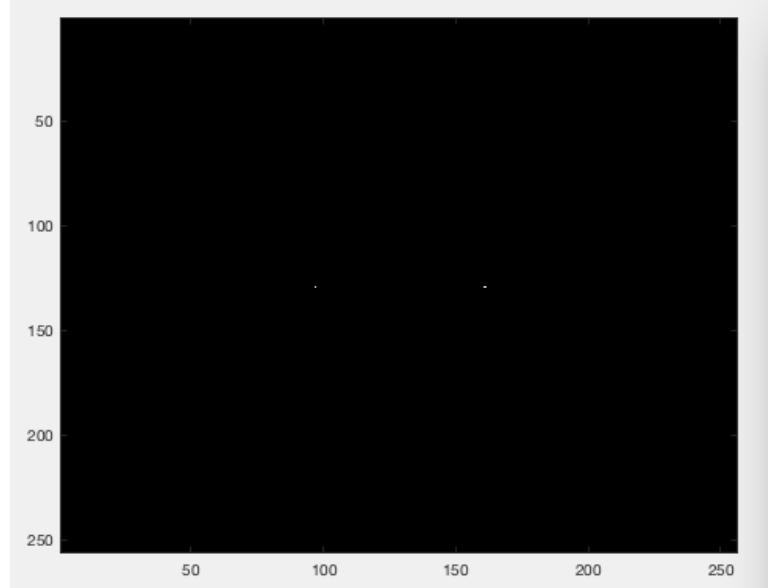
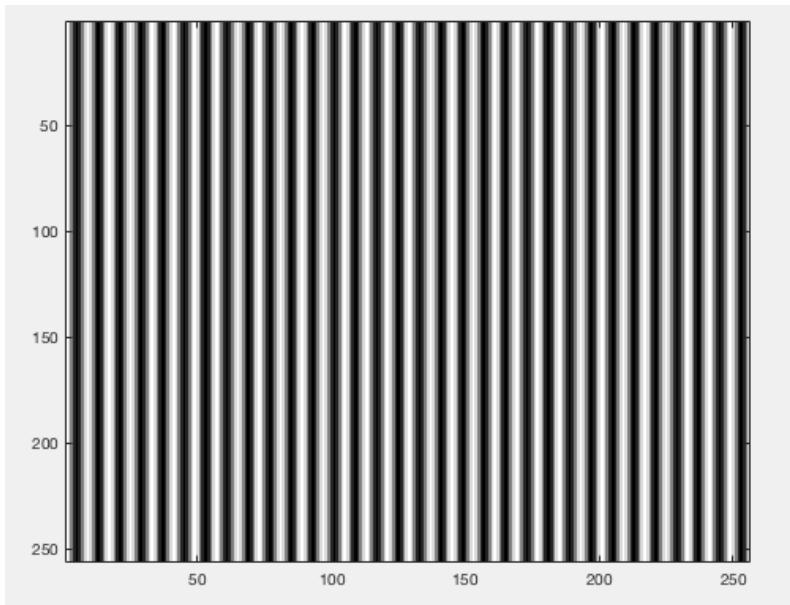
ONLINE LEARNING

Video 2.8
Kostas Daniilidis

2D Fourier Transform

$$f(x, y) \leftrightarrow \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-j\omega_x x + \omega_y y} dx dy$$
$$F(\omega_x, \omega_y) \leftrightarrow \left(\frac{1}{2\pi} \right) \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(\omega_x, \omega_y) e^{j\omega_x x + j\omega_y y} d\omega_x d\omega_y$$

Example: 1D-cosine as an image



$$f(x, y) = \cos(\omega_0 x) \quad f(x, y) \leftarrow \frac{1}{2}(\delta(\omega_x - \omega_0) + \delta(\omega_x + \omega_0)) \cdot \delta(\omega_y)$$

Separable functions

$$f(x, y) = f_1(x)f_2(y) \Leftrightarrow \int_{-\infty}^{\infty} f_1(x)e^{-j\omega_x x} dx \int_{-\infty}^{\infty} f_2(y)e^{-j\omega_y y} dy = F_1(\omega_x)F_2(\omega_y)$$

$$f(x, y) = \cos(\omega_1 x) \cos(\omega_2 y)$$



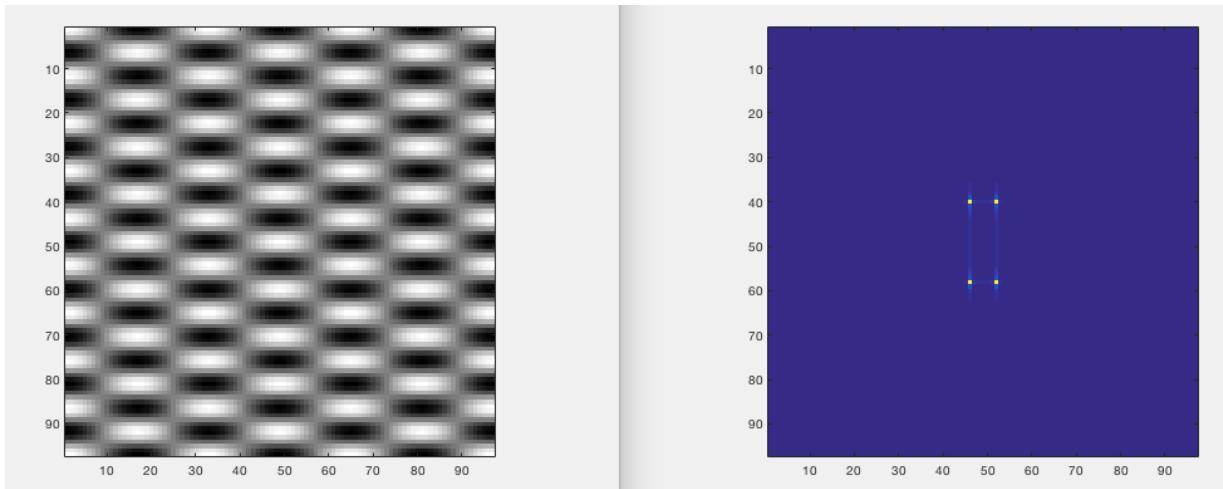
$$\frac{1}{2}(\delta(\omega_x - \omega_1) + \delta(\omega_x + \omega_1)) \frac{1}{2}(\delta(\omega_y - \omega_2) + \delta(\omega_y + \omega_2))$$

Separable functions

$$f(x, y) = \cos(\omega_1 x) \cos(\omega_2 y)$$



$$\frac{1}{2}(\delta(\omega_x - \omega_1) + \delta(\omega_x + \omega_1)) \frac{1}{2}(\delta(\omega_y - \omega_2) + \delta(\omega_y + \omega_2))$$

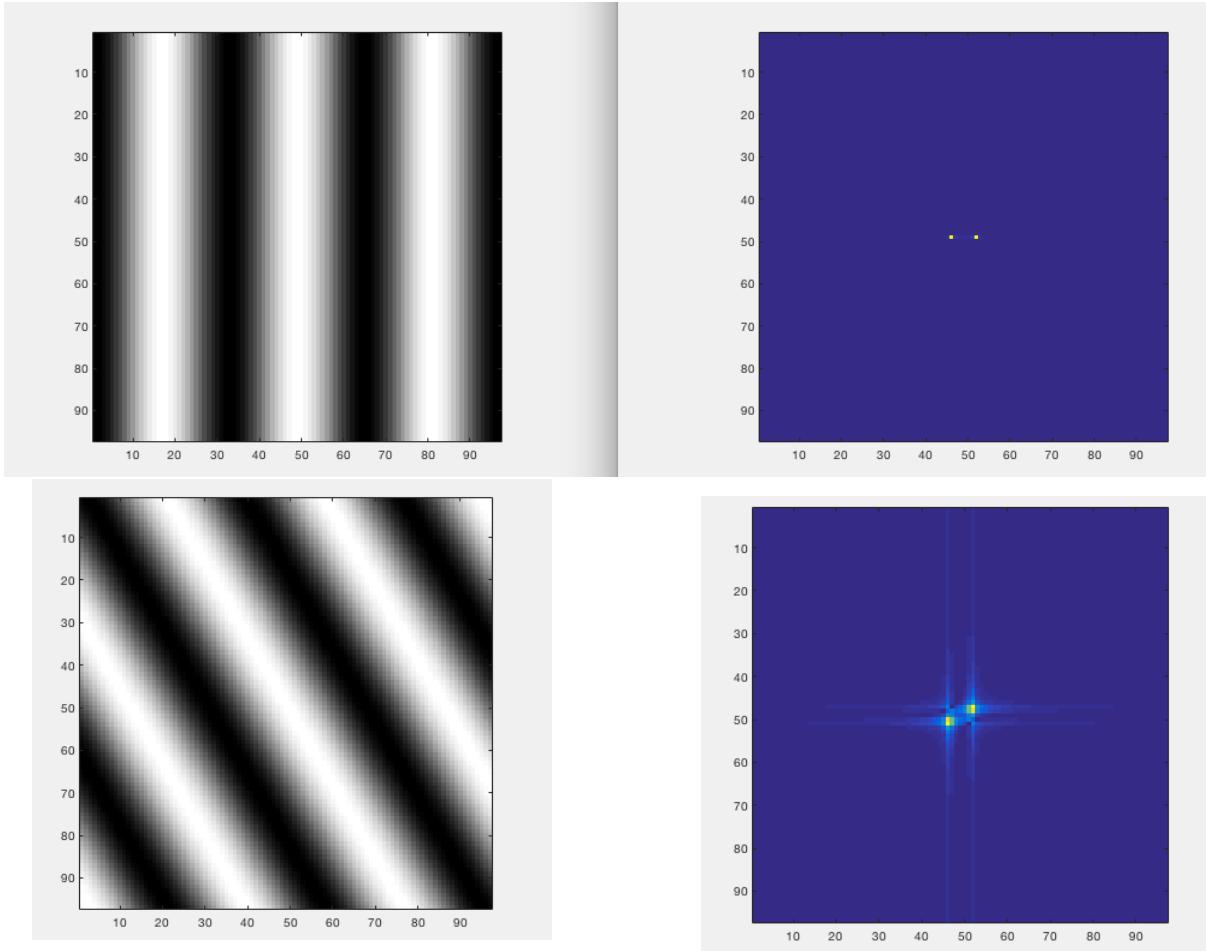


Shift Theorem in 2D

$$f(x - x_0, y - y_0) \xrightarrow{\text{FT}} F(\omega_x, \omega_y) e^{-j(\omega_x x_0 + \omega_y y_0)}$$

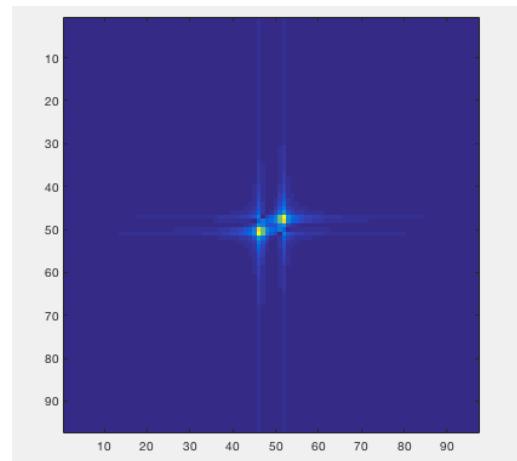
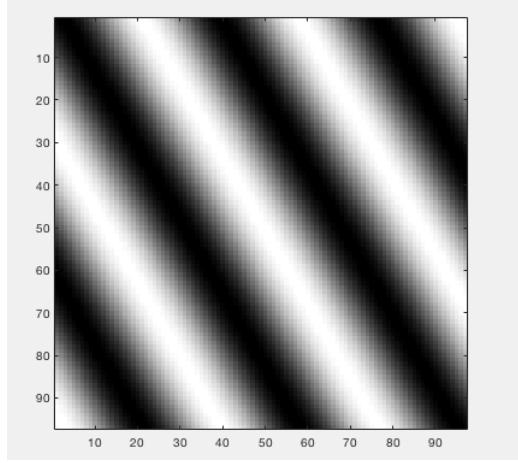
If we know the phases of two 1D signals we can recover how their relative displacement?
But can we do that for 2D images?

2D rotation

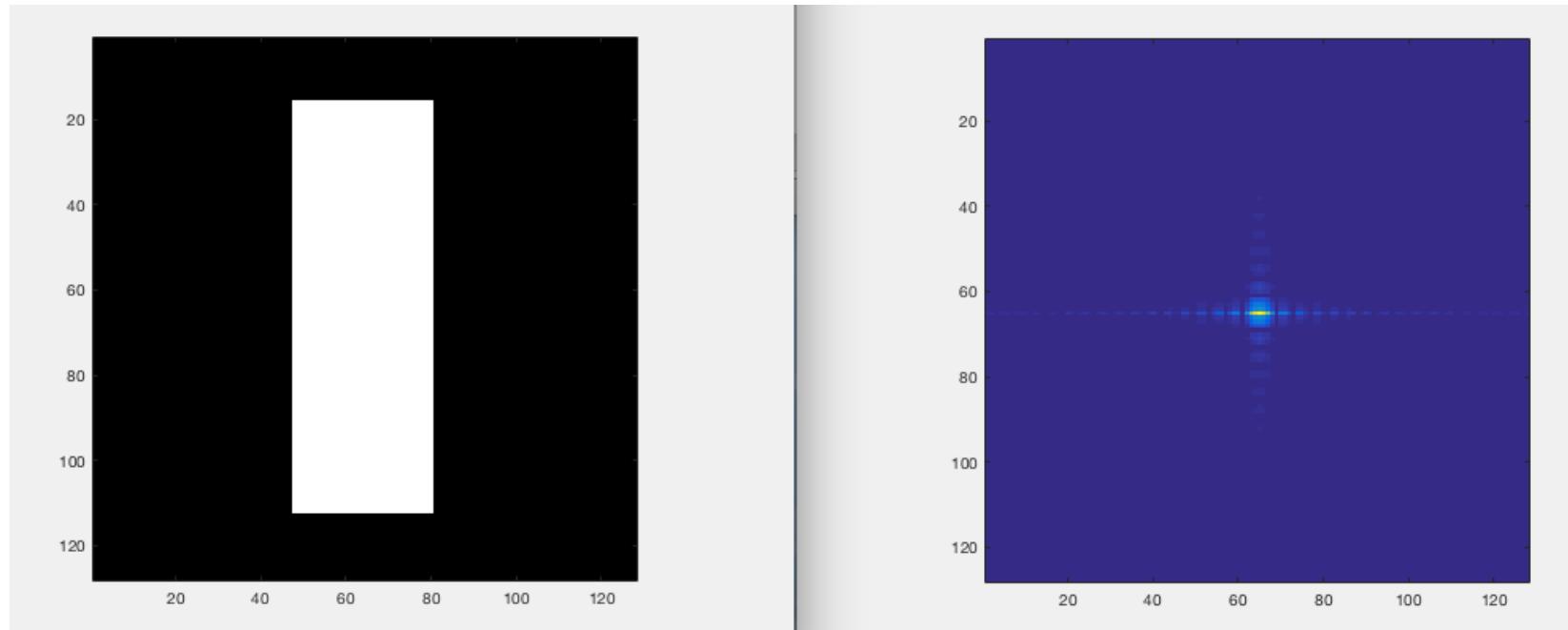


2D rotation

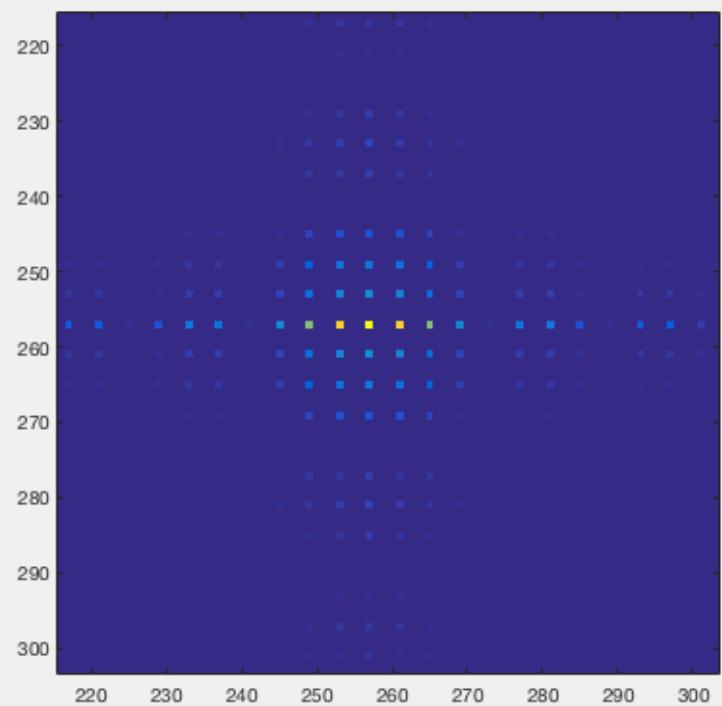
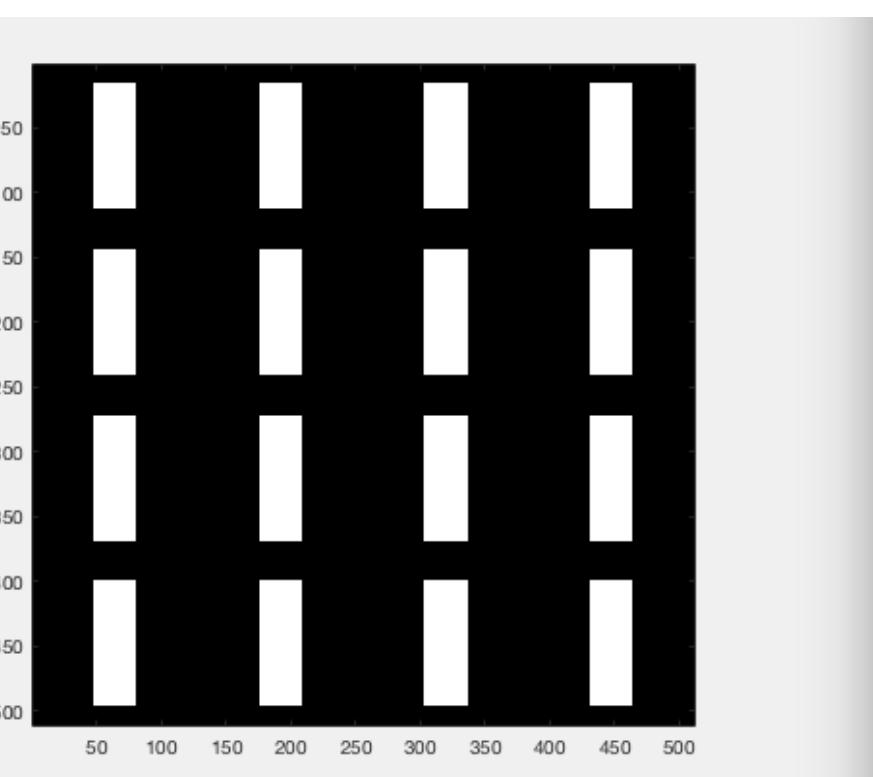
$$f * R \begin{bmatrix} x \\ y \end{bmatrix}) \circledcirc \mathcal{F} \left(R \begin{bmatrix} \omega_x \\ \omega_y \end{bmatrix} \right)$$



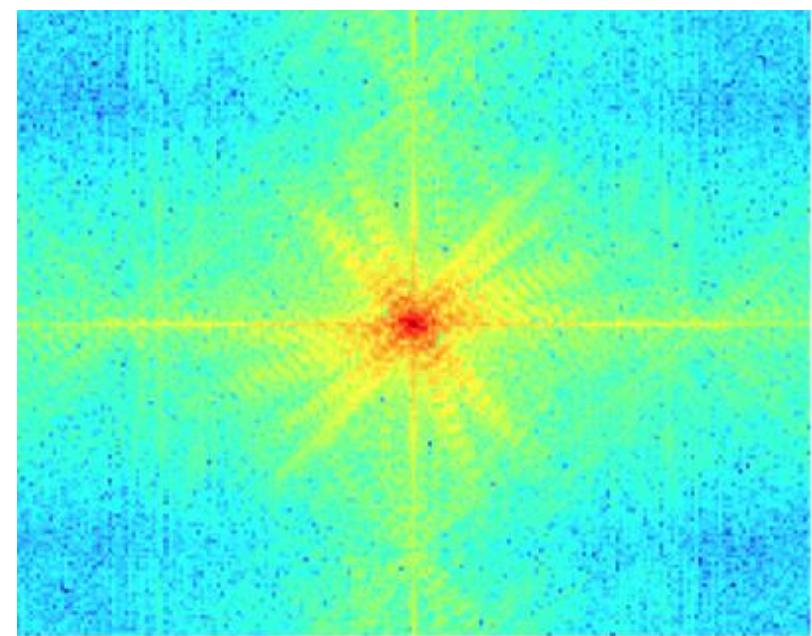
2D Fourier of a box



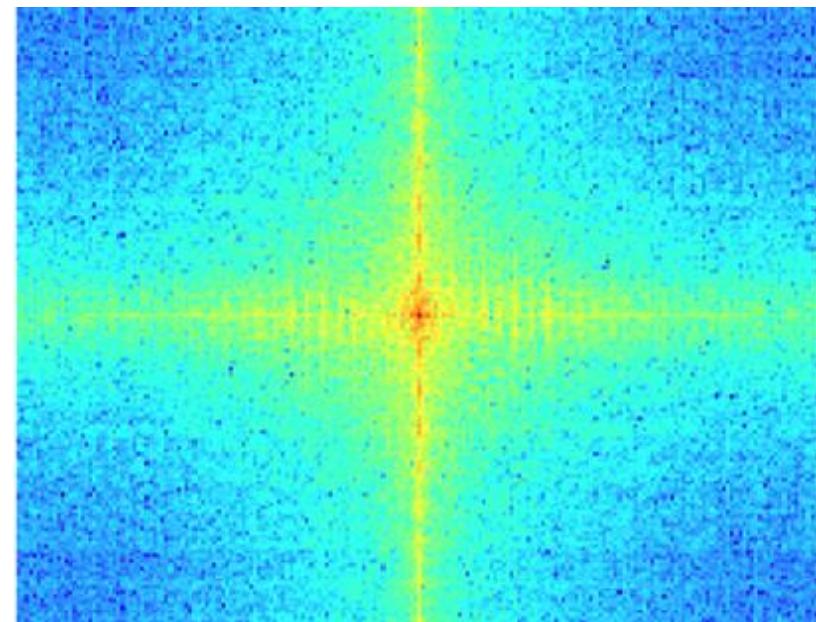
How do we model other periodic patterns?



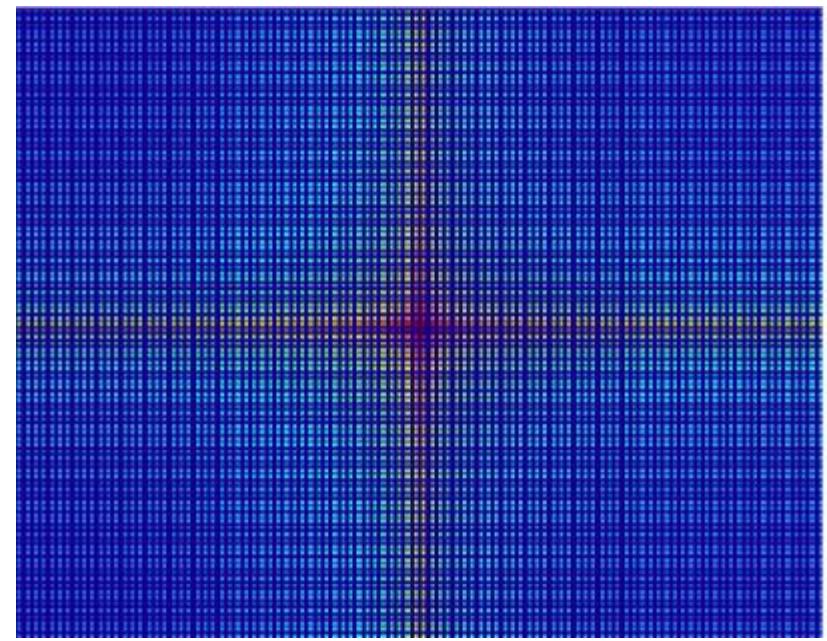
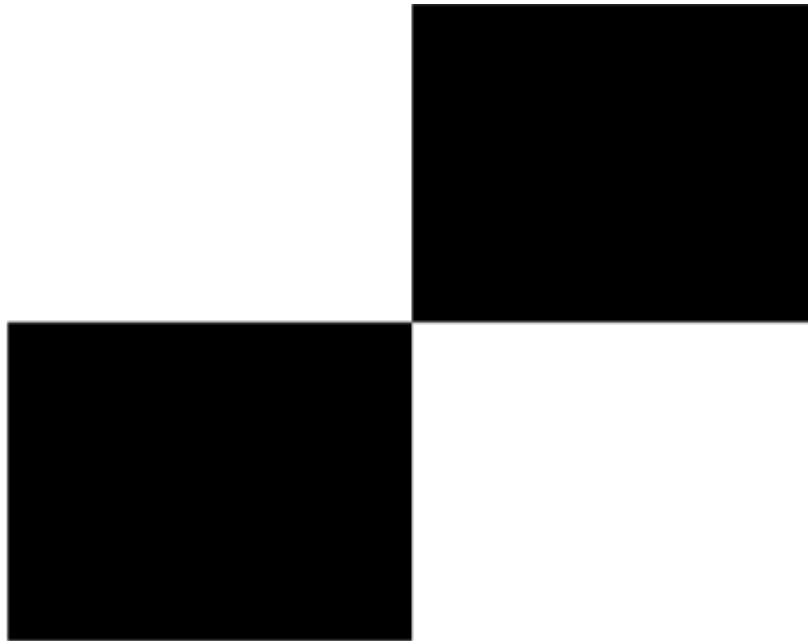
Clue about orientation of edges

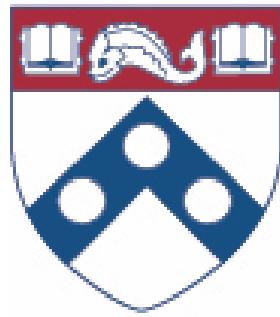


Clue about periodicity



Clues about contrast





Penn
Engineering

ONLINE LEARNING

Video 2.9
Jianbo Shi

Edge Formation Factors

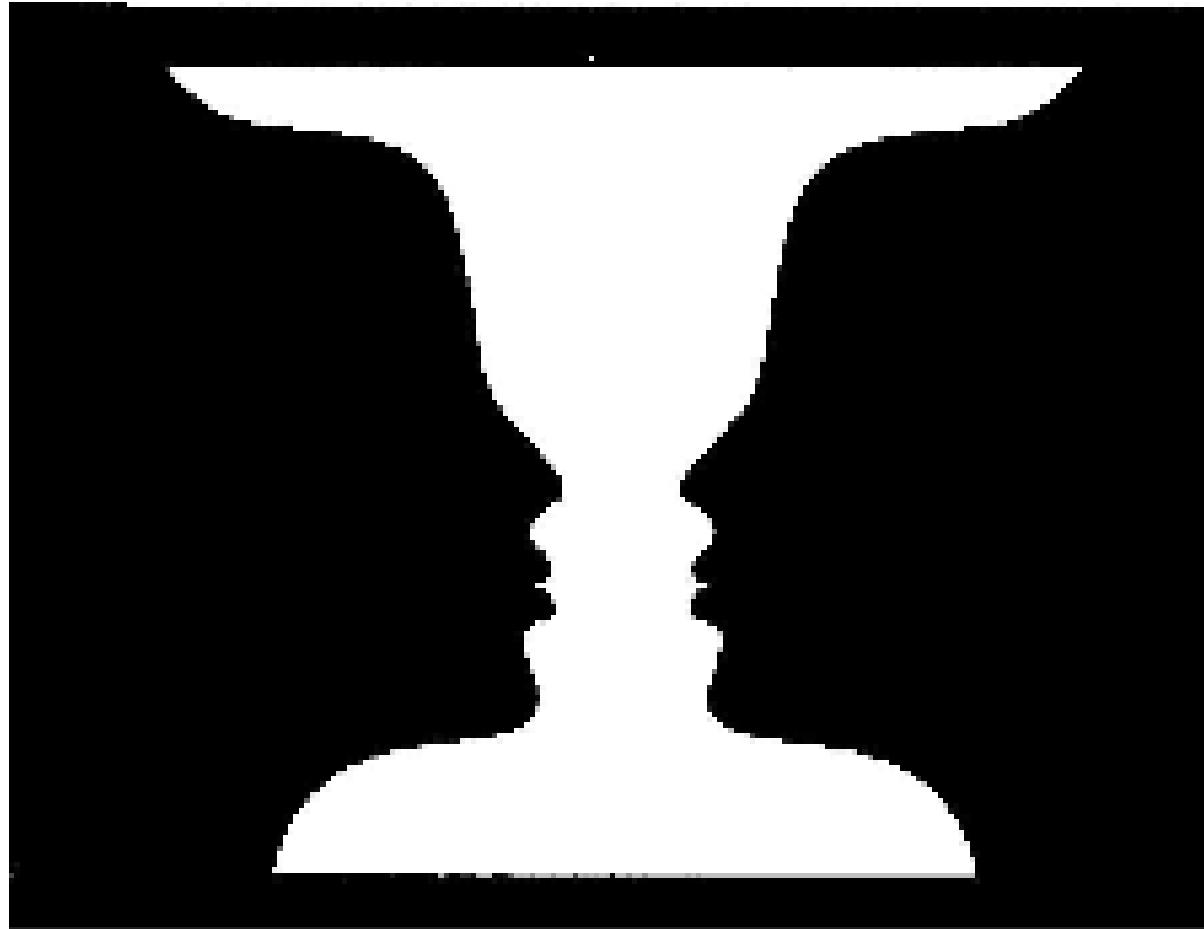


Surface normal discontinuity

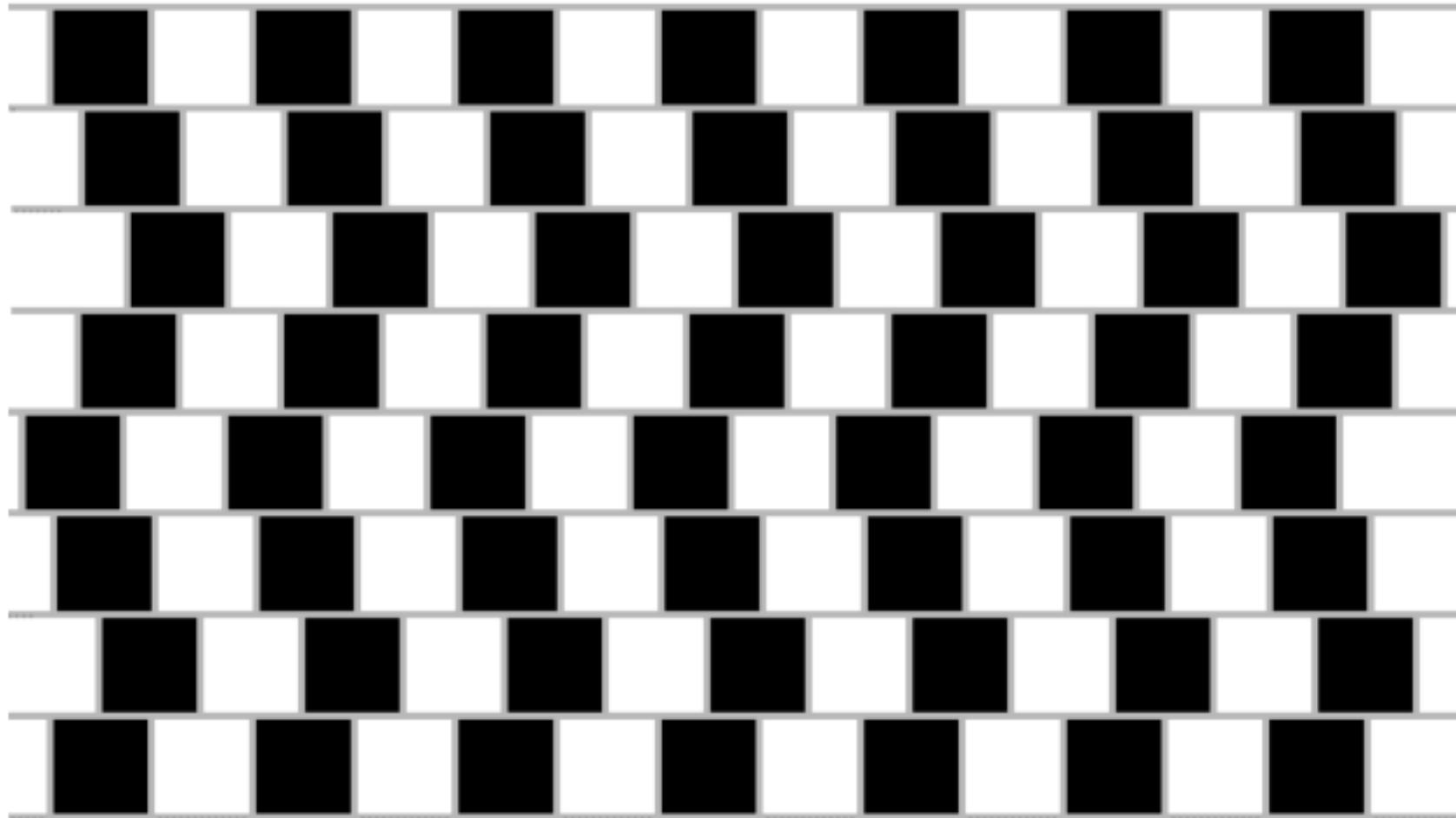
What's in front?

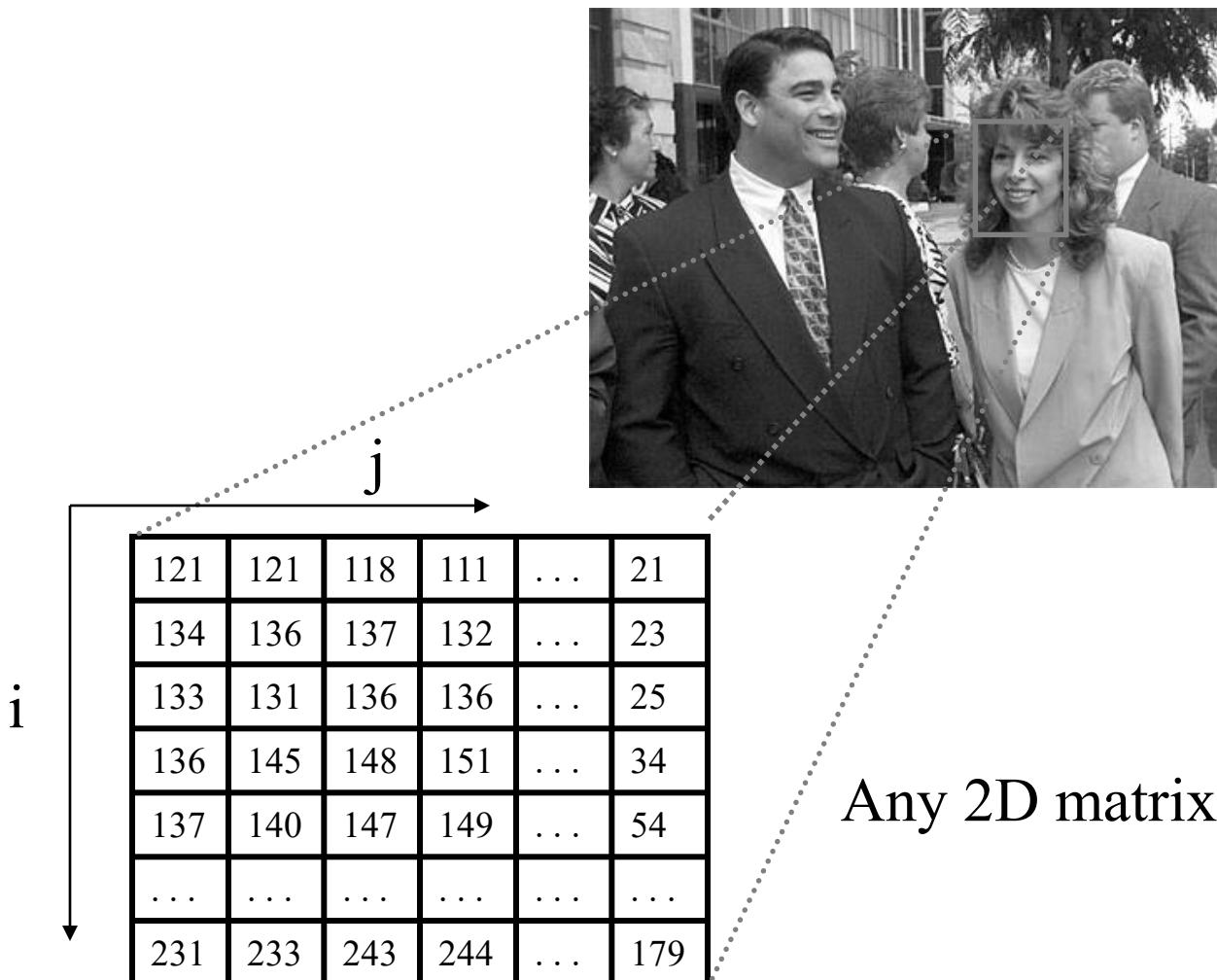


What it is?



The rows of black and white squares are all parallel.
The vertical zigzag patterns disrupt our horizontal perception.





Any 2D matrix can be seen as an image

Linear Filtering

Image I

200	130	20
255	100	10
200	100	30

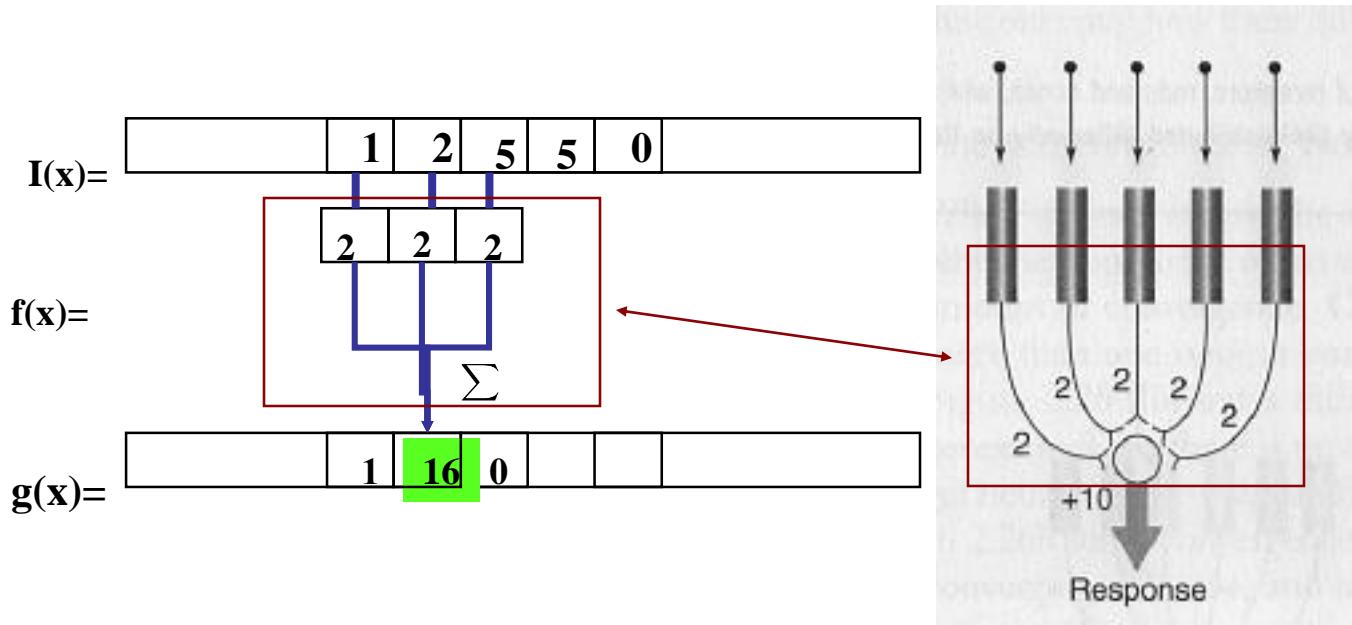
Kernel f

0.5	0	0
0	1.5	0.5
0	-0.5	0

	205	

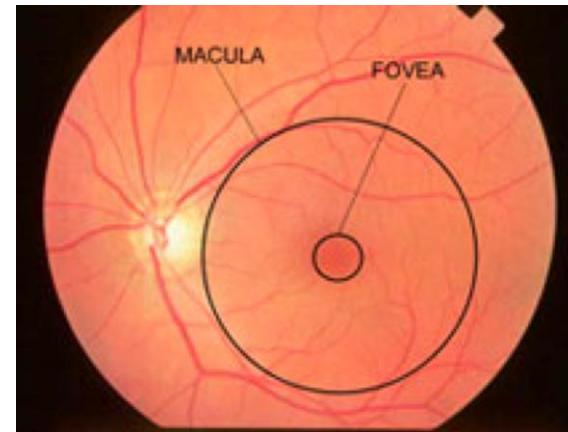
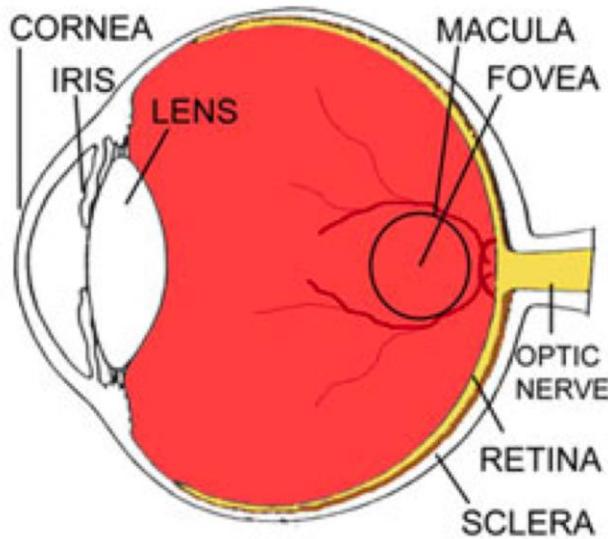
Image *filtering*: Replacing each pixel value by weighted average of its neighbors

Simple Neural Network



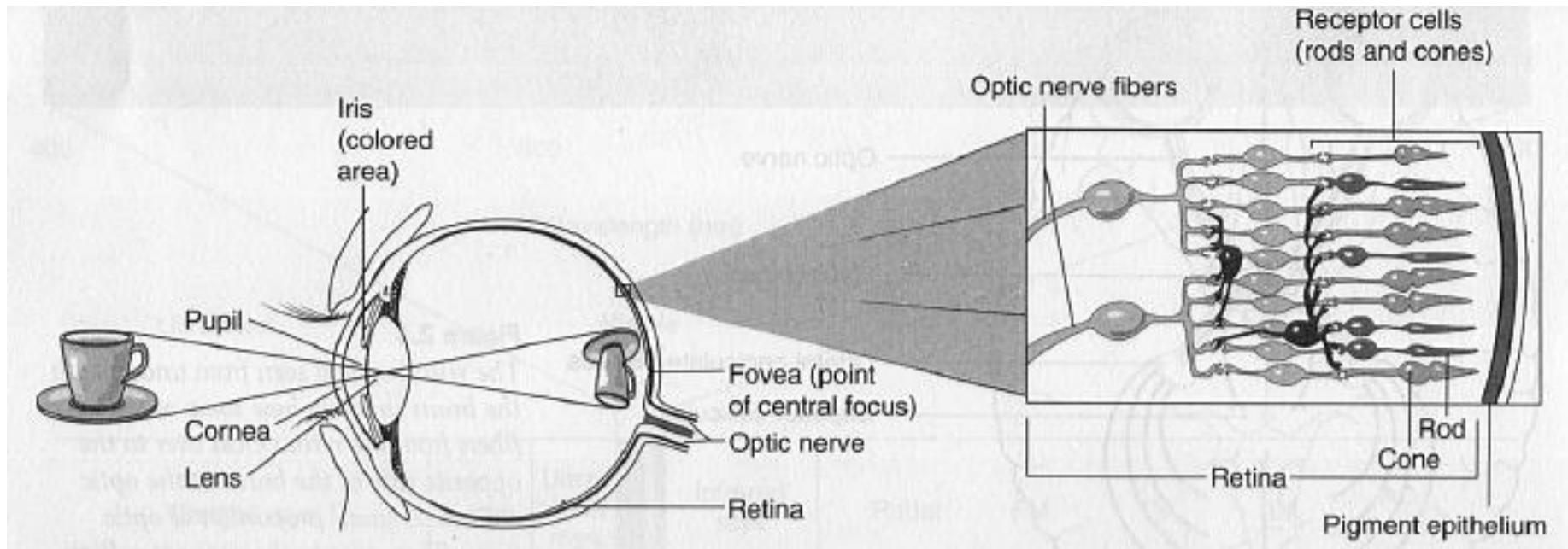
This leads to parallel computation

Anatomy of eye



The macula is the center of vision (and retina) and the fovea (FAZ) is the focal point approximately only 0.4mm in diameter. Reading, driving, etc. is all performed here.

Photo-sensors



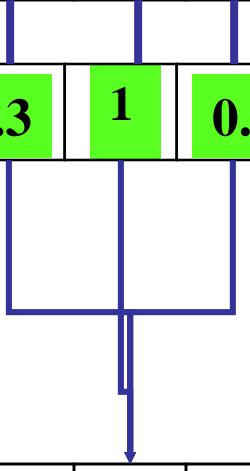
- 1) Light pass through retina cells, excites Rod and Cone
- 2) Cone: color(spectral) sensitive, R,G,B, 6 Million
- 3) Rod: more photo sensitive, peak at 580nm(yellow), 120 M
- 4) What happens if you miss one type of Cone cells?

$I(x) =$

	0	0	1	0	0	
--	---	---	---	---	---	--

0.3	1	0.6
-----	---	-----

$f(x) =$



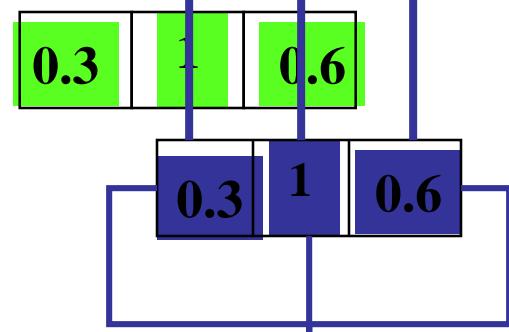
$g(x) =$

	0	0.6				
--	---	-----	--	--	--	--

$I(x) =$

	0	0	1	0	0	
--	---	---	---	---	---	--

$f(x) =$



$g(x) =$

	0	0.6	1			
--	---	-----	---	--	--	--

$I(x) =$

	0	0	1	0	0	
--	---	---	---	---	---	--

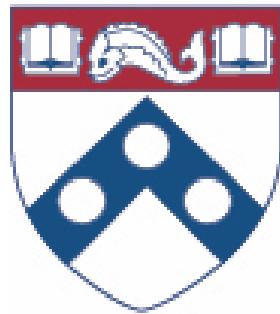
$f(x) =$

0.3	1	0.6
0.3	1	0.6

0.3	1	0.6
-----	---	-----

$g(x) =$

	0	0.6	1	0.3	
--	---	-----	---	-----	--

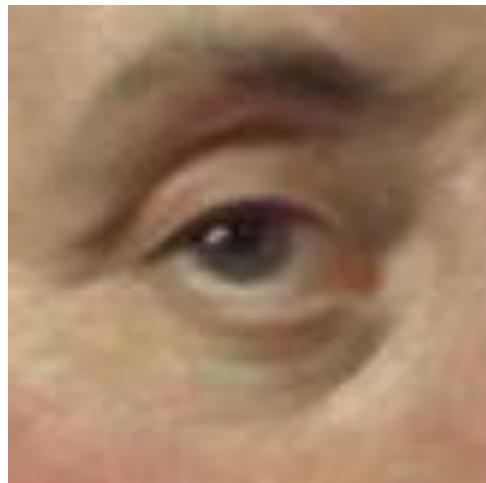


Penn
Engineering

ONLINE LEARNING

Video 2.10
Jianbo Shi

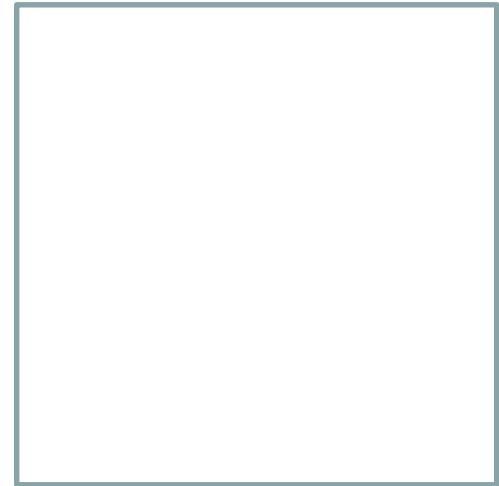
Linear Filter



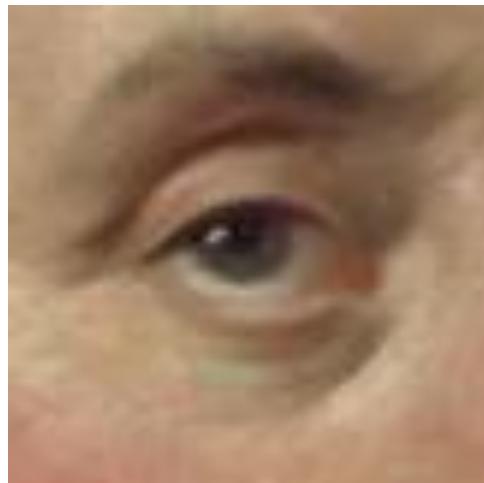
Image

0	0	0
0	1	0
0	0	0

Filter
Kernel



Linear Filter



Image

0	0	0
0	1	0
0	0	0

Kernal

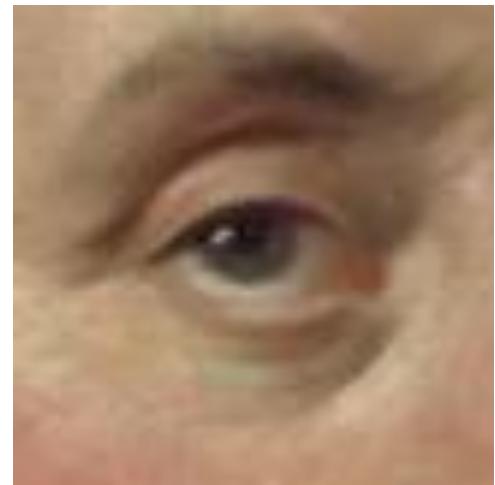
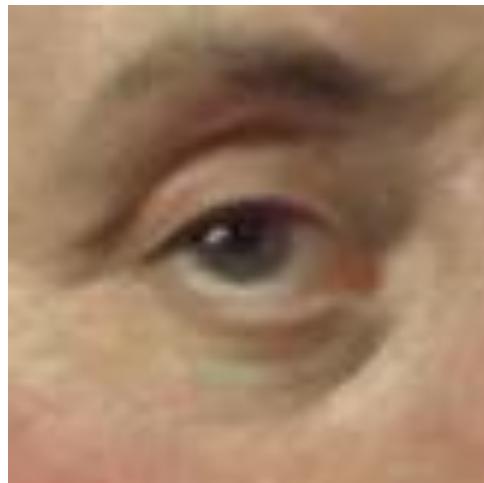


Image no change

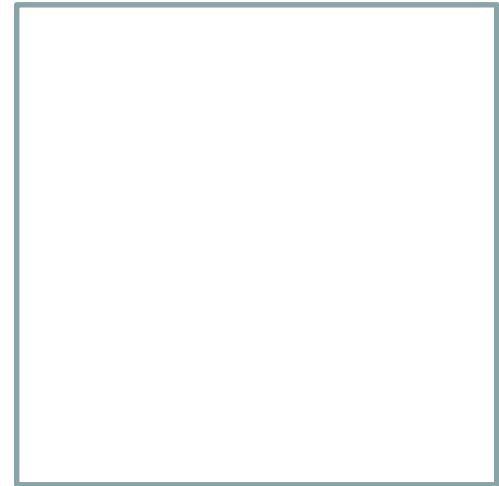
Linear Filter



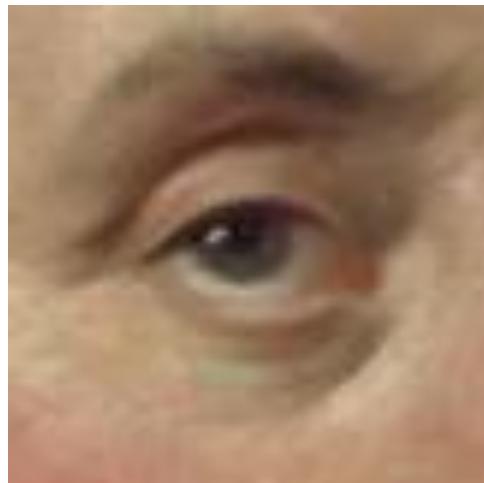
Image

0	0	0
1	0	0
0	0	0

Kernal



Linear Filter



Image

0	0	0
1	0	0
0	0	0

Kernal

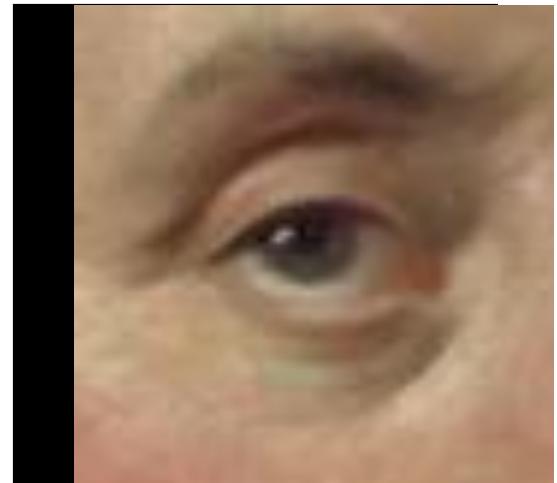
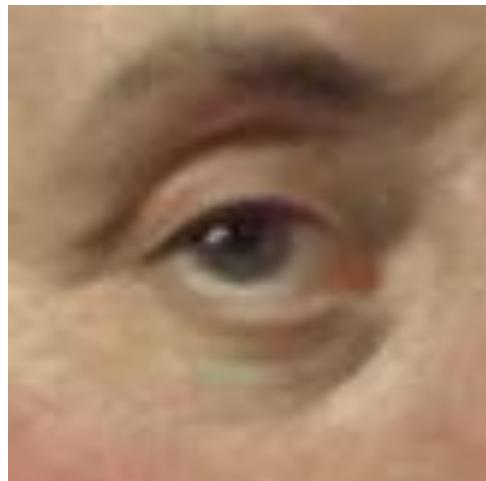


Image Shifted

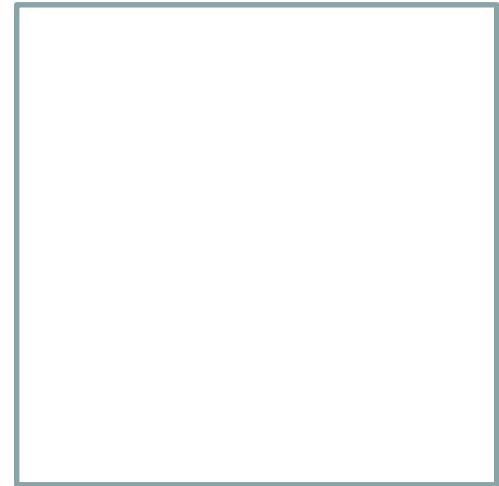
Linear Filter



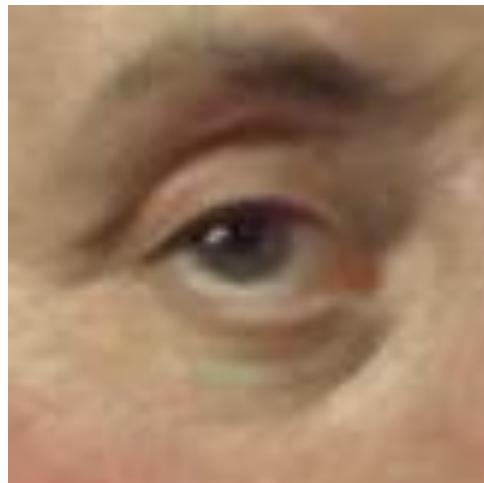
Image

0	0	0
0.3	0.3	0.3
0	0	0

Kernal



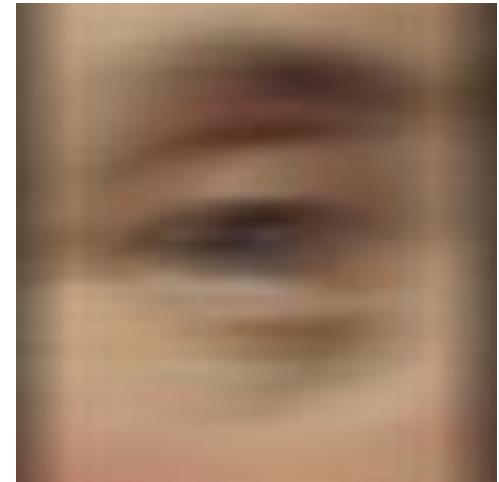
Linear Filter



Image

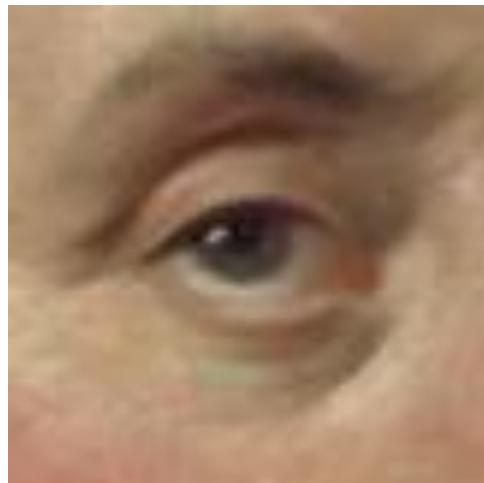
0	0	0
0.3	0.3	0.3
0	0	0

Kernal



Blurred
(horizontal)

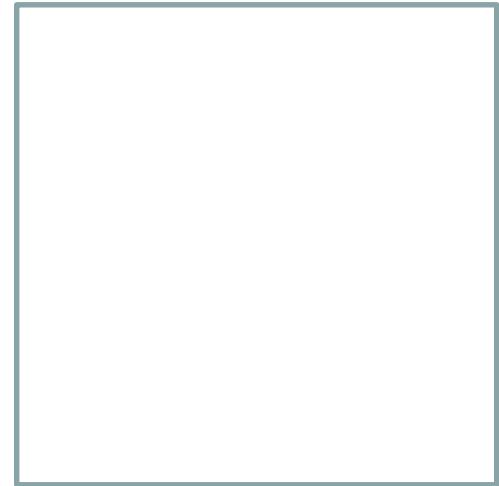
Linear Filter



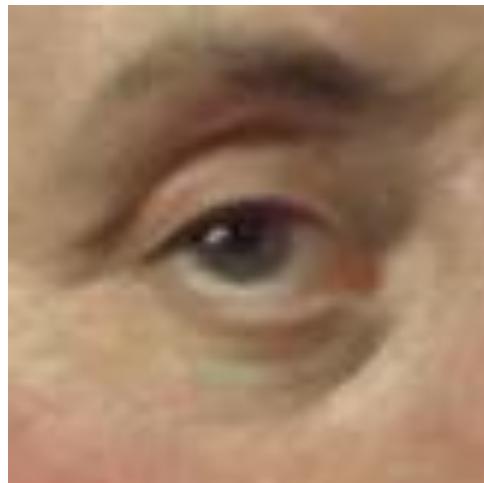
Image

0	0.3	0
0	0.3	0
0	0.3	0

Kernal



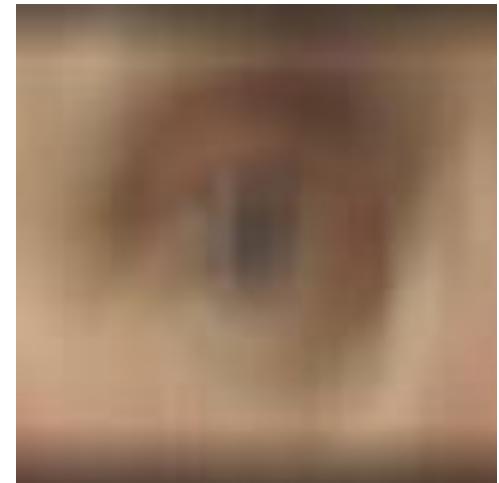
Linear Filter



Image

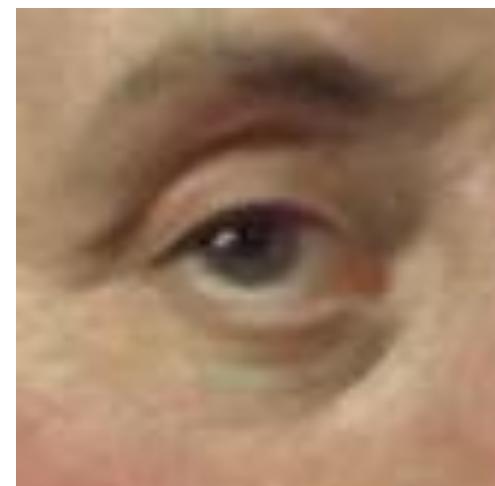
0	0.3	0
0	0.3	0
0	0.3	0

Kernal



Blurred (vertical)

Linear Filter



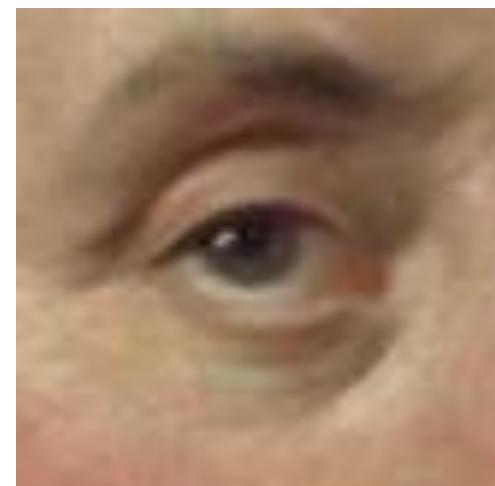
Image

$$\begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0.3 & 0.3 & 0.3 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

Filter
Kernel

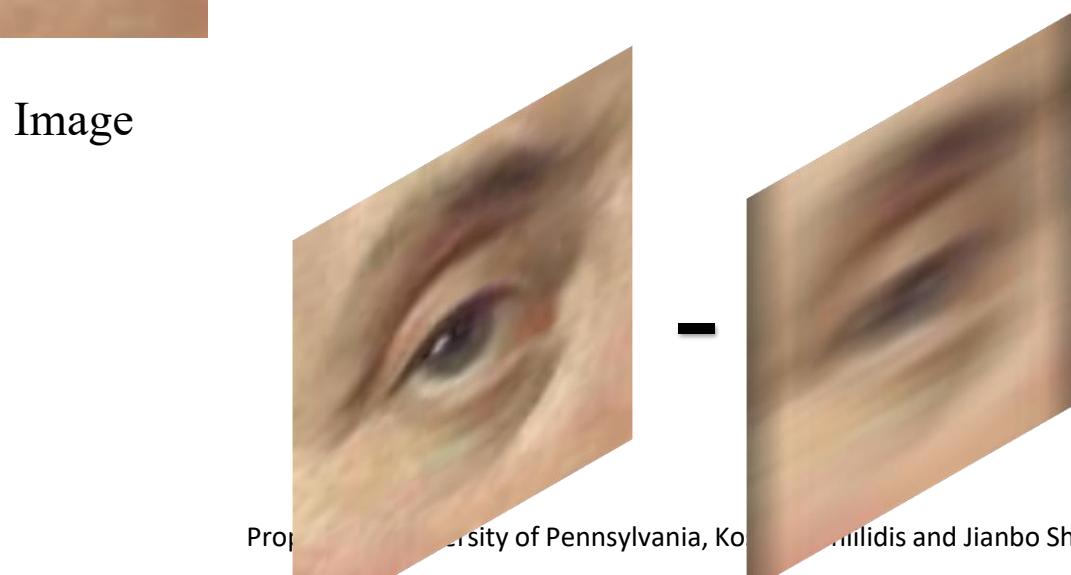


Linear Filter

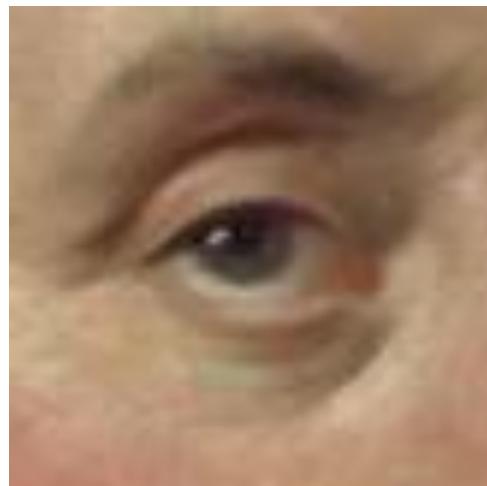


$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 \\ 0.3 & 0.3 & 0.3 \\ 0 & 0 & 0 \end{bmatrix}$$

Kernel Image



Linear Filter



$$\begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0.3 & 0.3 & 0.3 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

Kernal Image



Image

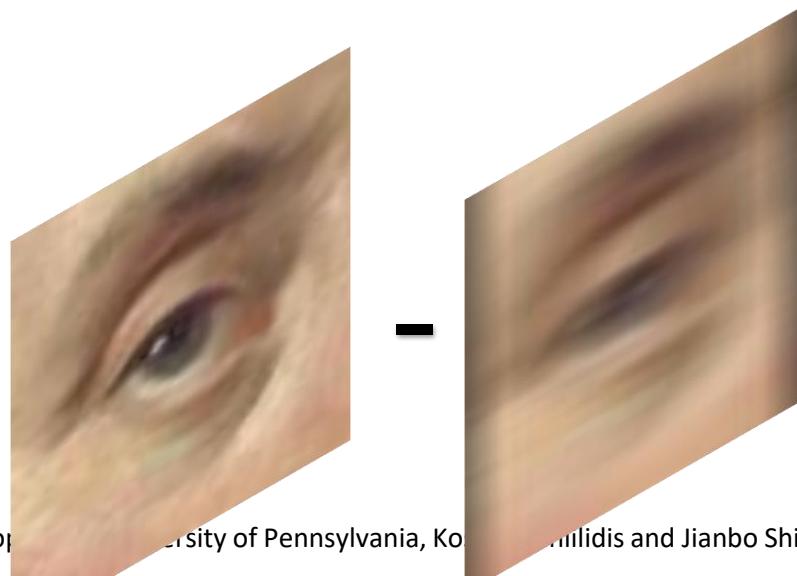


Image filtering

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

Output
Image

Input
Image

Kernal
Image

Image filtering

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

Image I 8x8

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	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

Kernel f
3x3

1	2	3
4	5	6
7	8	9

Same position

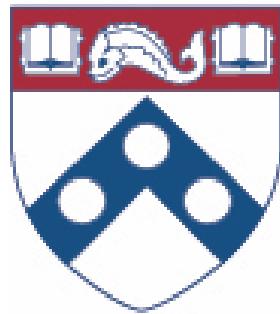
Register

a	b	c
d	e	f
g	h	i

Loop over every pixel

Calculate result = $a*1+b*2+\dots+i*9$

28	39	39	39	39	39	39	39	24
33	45	45	45	45	45	45	45	27
33	45	45	45	45	45	45	45	27
16	21	21	21	21	21	21	21	12
5	6	6	6	6	6	6	6	3
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

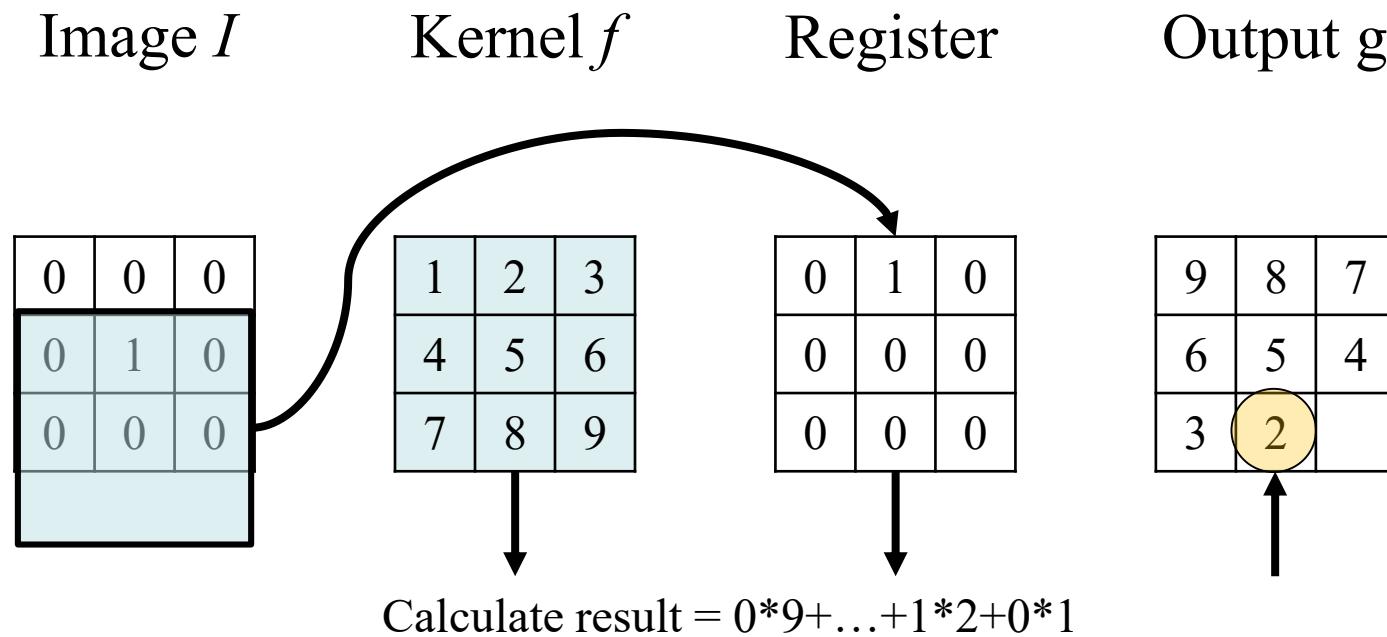


Penn
Engineering

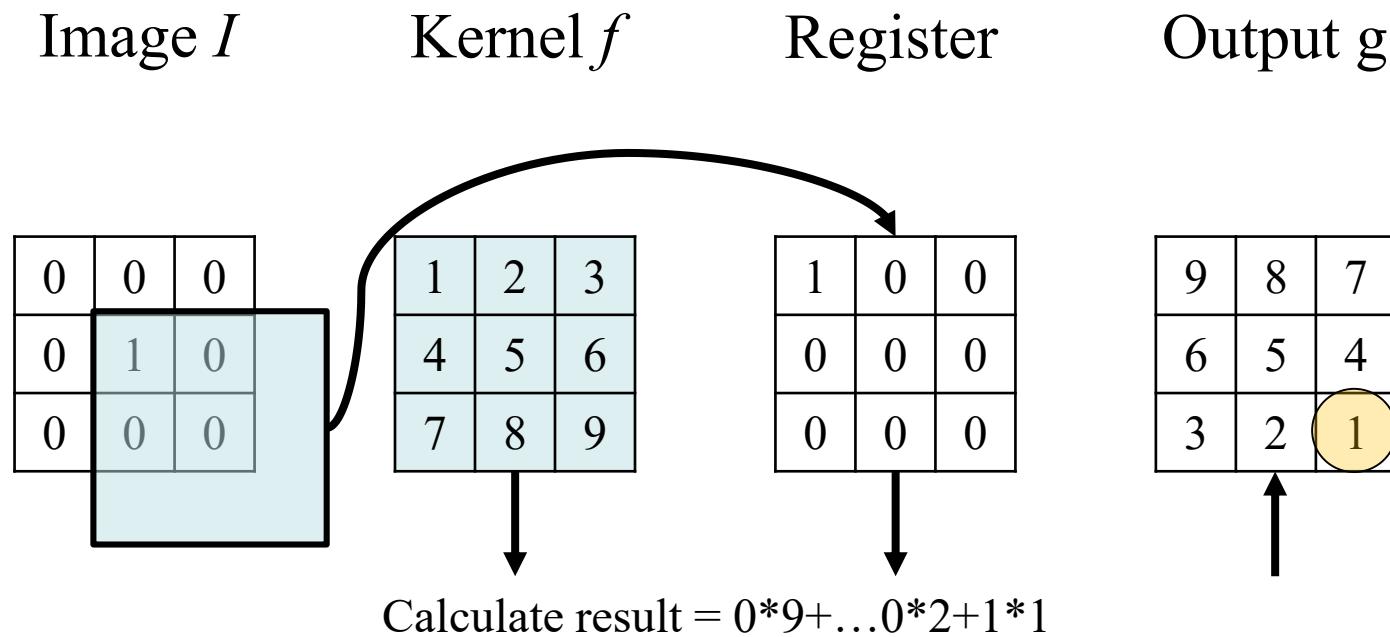
ONLINE LEARNING

Video 2.11
Jianbo Shi

Special case: impulse function



Special case: impulse function



<Note> The output is the kernel flipped left-right, up-down!

Convolution $I \otimes f$

- Let I be an Signal(image), Convolution kernel f ,

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

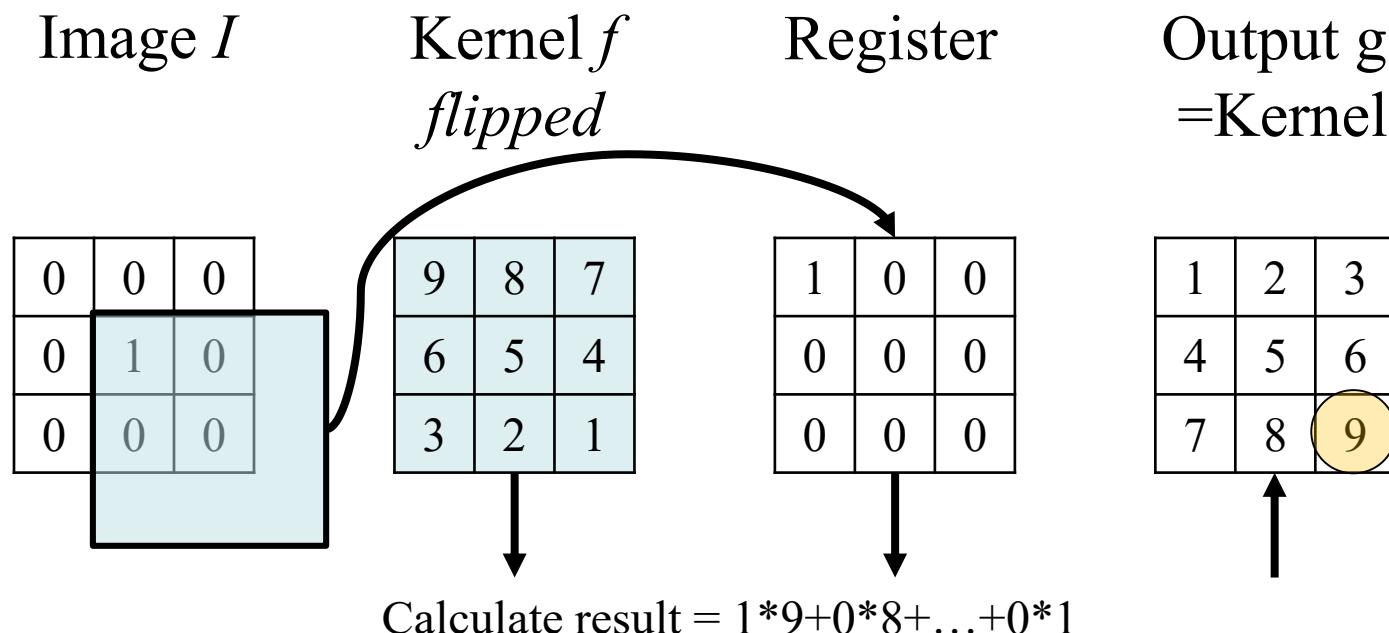
Output
Image

Input
Image

Kernal
Image

Convolution

- Convolution is filtering with kernel flipped



Impulse functions shift images

Image I

a	b	c
d	e	f
g	h	i

Kernel f

1	0	0
0	0	0
0	0	0

Kernel f'

0	0	0
0	0	0
0	0	1

Result $I \otimes f$

e	f	0
h	i	0
0	0	0

- In this case the resulting image shifted to the upper left

Impulse functions shift images

Image I

1	0	0
0	0	0
0	0	0

Kernel f

a	b	c
d	e	f
g	h	i

Kernel f'

i	h	g
f	e	d
c	b	a

Result $I \otimes f$

e	f	0
h	i	0
0	0	0

- In this case the resulting image shifted to the upper left

Convolution is associative

I

a	b	c
d	e	f
g	h	i

f

1	0	0
0	0	0
0	0	0

$I \otimes f$

e	f	0
h	i	0
0	0	0

f

1	0	0
0	0	0
0	0	0

I

a	b	c
d	e	f
g	h	i

$f \otimes I$

e	f	0
h	i	0
0	0	0

Linear independence

Image I

2	0	0
0	3	0
0	0	0

Kernel f
flipped

9	8	7
6	5	4
3	2	1

Output g

37	32	21
22	17	12
9	6	3

Decompose



1	0	0
0	0	0
0	0	0

$*2$



Intermediate

5	4	0
2	1	0
0	0	0

$*2$



Add together



0	0	0
0	1	0
0	0	0

$*3$



9	8	7
6	5	4
3	2	1

$*3$



Linear independence

Image I

0	1	0	1	0
0	1	0	1	0
0	1	0	1	0
0	1	0	1	0
0	1	0	1	0

5x5

Kernel f

1	0	0
0	1	0
0	0	0

Output $g = g_1 + g_2$

1	1	1	1	0
1	1	1	1	0
1	1	1	1	0
1	1	1	1	0
0	1	0	1	0

Kernel f_1

1	0	0
0	0	0
0	0	0

Output g_1

1	0	1	0	0
1	0	1	0	0
1	0	1	0	0
1	0	1	0	0
0	0	0	0	0

Output g_2

0	1	0	1	0
0	1	0	1	0
0	1	0	1	0
0	1	0	1	0
0	1	0	1	0

Kernel f_2

0	0	0
0	1	0
0	0	0

- Convolution is commutative

$$I \otimes f$$

Image I

a	b	c
d	e	f
g	h	i

Kernel f

1	0	0
1	0	0
1	1	0



Decompose

Image I

a	b	c
d	e	f
g	h	i

\otimes

0	0	0
1	0	0
0	0	0



e	f	0
h	i	0
0	0	0

b	c	0
e	f	0
h	i	0

0	0	0
0	0	0
1	0	0

0	0	0
a	b	c
d	e	f

- Convolution is commutative

$$f \otimes I$$

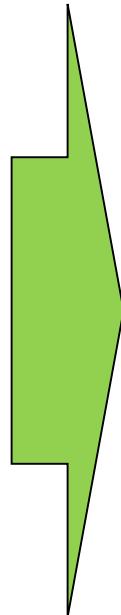
Image I

a	b	c
d	e	f
g	h	i

Kernel f

1	0	0
1	0	0
1	1	0

Decompose



1	0	0
0	0	0
0	0	0

0	0	0
1	0	0
0	0	0

0	0	0
0	0	0
1	0	0

0	0	0
0	0	0
0	1	0

a	b	c
d	e	f
g	h	i

0	0	0
a	b	c
d	e	f



e	f	0
h	i	0
0	0	0

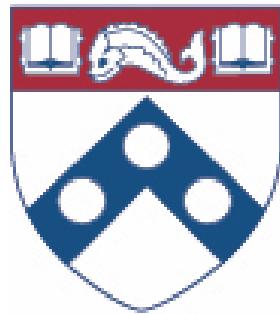
b	c	0
e	f	0
h	i	0

0	0	0
b	c	0
e	f	0

0	0	0
a	b	c
d	e	f

Proof of Commutative property

- $g[m, n] = I \otimes f = f \otimes I$
- $g[m, n] = I \otimes f = \sum_{k,l} I(m - k, n - l) * f(k, l)$
- Let $k' = m - k, l' = n - l$,
then $k = m - k', l = n - l'$
- $g[m, n] = \sum_{k',l'} I(k', l') * f(m - k', n - l') = f \otimes I$



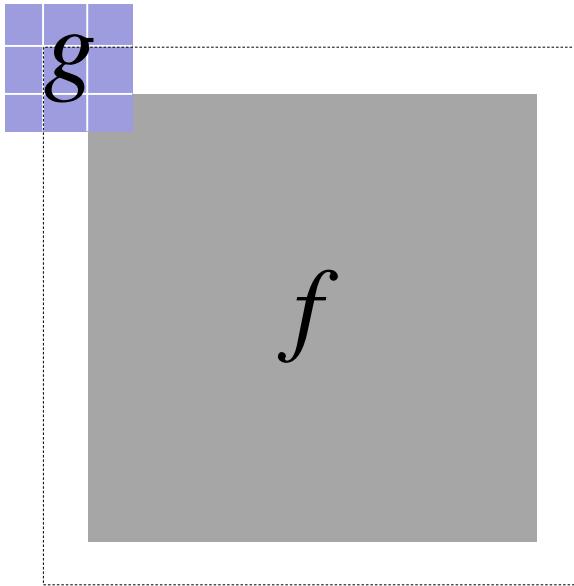
Penn
Engineering

ONLINE LEARNING

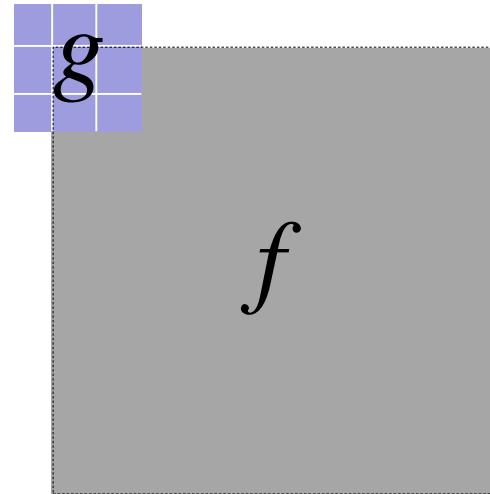
Video 2.12
Jianbo Shi

Output Size of Image Convolution

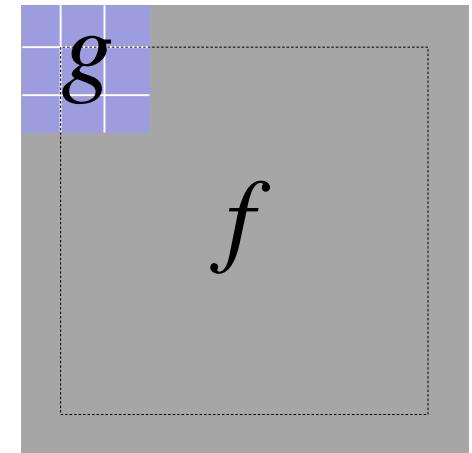
$$f \otimes g$$



Full



Same



Valid

`filter2(g, f, shape)` in MATLAB

Full: $\text{output_size} = \text{f_size} + \text{g_size} - 1$

Same: $\text{output_size} = \text{f_size}$

Valid: $\text{output_size} = \text{f_size} - (\text{g_size} - 1)$

2D visualization of convolution (full)

Image I

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	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

8x8

Kernel f

1	2	3
4	5	6
7	8	9

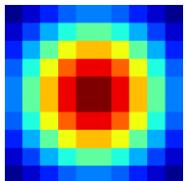
3x3

Output g

1	3	6	6	6	6	6	6	5	3
5	12	21	21	21	21	21	21	16	9
12	27	45	45	45	45	45	45	33	18
12	27	45	45	45	45	45	45	33	18
11	24	39	39	39	39	39	39	28	15
7	15	24	24	24	24	24	24	17	9
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	0

10x10

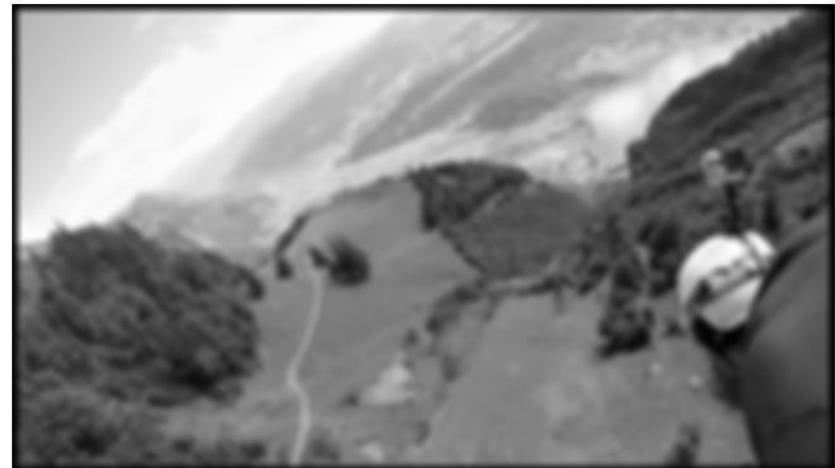
Output Size of Image Convolution



g : 10 x 10 Gaussian kernel



f : 640 x 360 resolution



Full

`filter2(g, f, shape)` in MATLAB

Full: $\text{output_size} = \text{f_size} + \text{g_size} - 1$

```
>> full = filter2(g, im, 'full');  
>> size(full)
```

ans =

369 649

2D visualization of convolution (same)

Image I

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	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

8x8

Kernel f

1	2	3
4	5	6
7	8	9

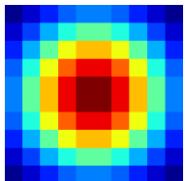
3x3

Output g

12	21	21	21	21	21	21	16
27	45	45	45	45	45	45	33
27	45	45	45	45	45	45	33
24	39	39	39	39	39	39	28
15	24	24	24	24	24	24	17
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

8x8

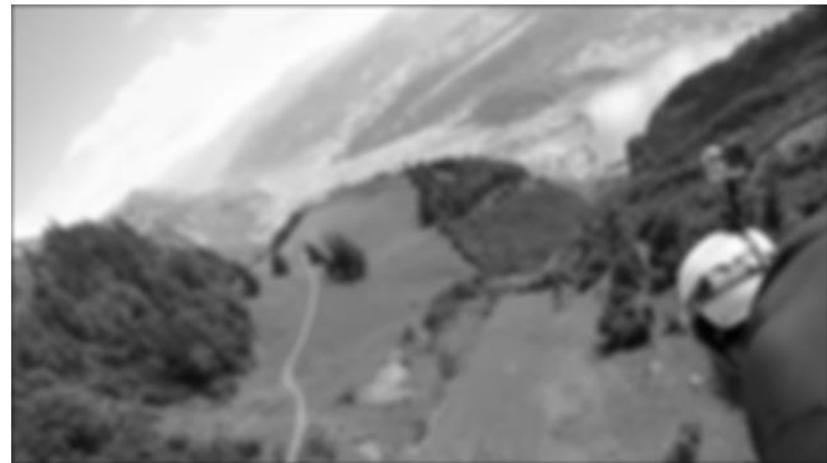
Output Size of Image Convolution



g : 10 x 10 Gaussian kernel



f : 640 x 360 resolution



Same

`filter2(g, f, shape)` in MATLAB

Full: $\text{output_size} = \text{f_size} + \text{g_size} - 1$

Same: $\text{output_size} = \text{f_size}$

```
>> same = filter2(g, im, 'same');  
>> size(same)
```

ans =

360 640

2D visualization of convolution (valid)

Image I

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	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

8x8

Kernel f

1	2	3
4	5	6
7	8	9

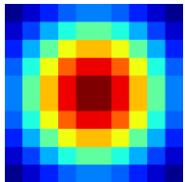
3x3

Output g

45	45	45	45	45	45
45	45	45	45	45	45
39	39	39	39	39	39
24	24	24	24	24	24
0	0	0	0	0	0
0	0	0	0	0	0

6x6

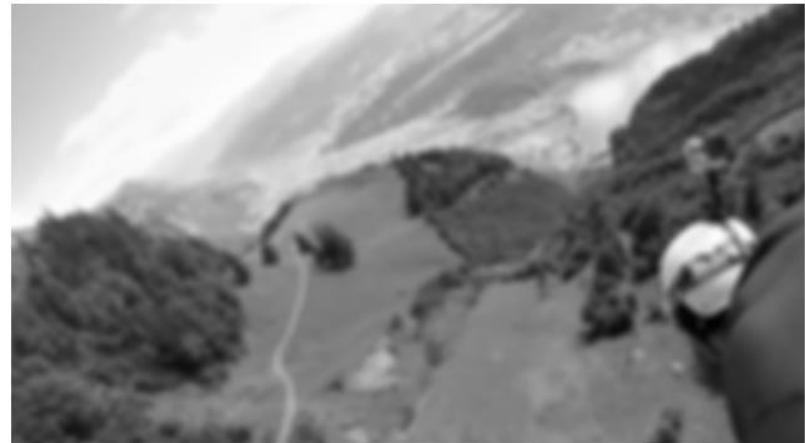
Output Size of Image Convolution



g : 10 x 10 Gaussian kernel



f : 640 x 360 resolution



Valid

$filter2(g, f, shape)$ in MATLAB

Full: $output_size = f_size + g_size - 1$

Same: $output_size = f_size$

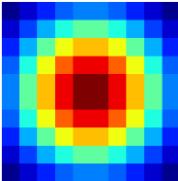
Valid: $output_size = f_size - (g_size - 1)$

```
>> valid = filter2(g, im, 'valid');  
>> size(valid)
```

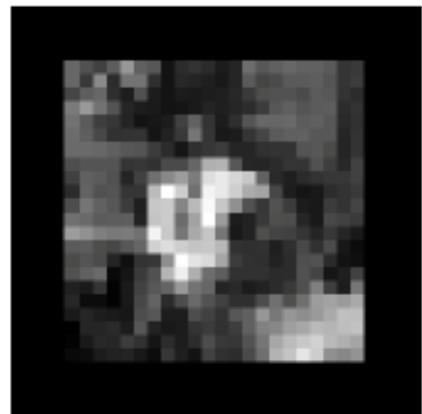
ans =

351 631

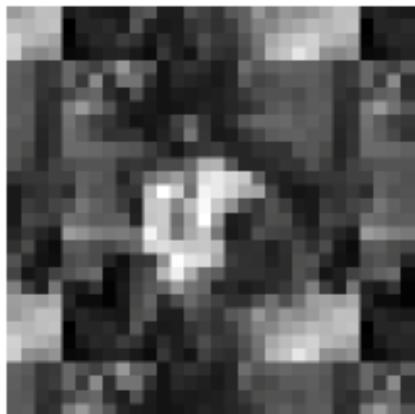
Image Boundary Effect



The filter window falls off at the edge of image.



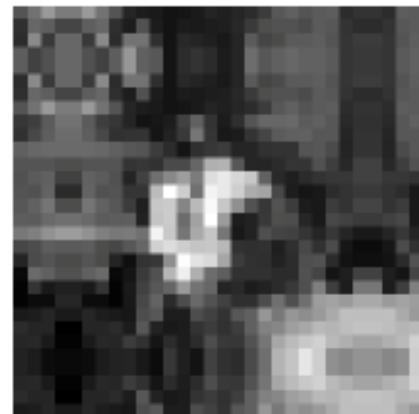
zero



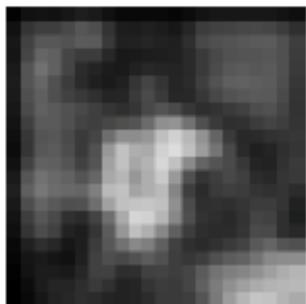
wrap



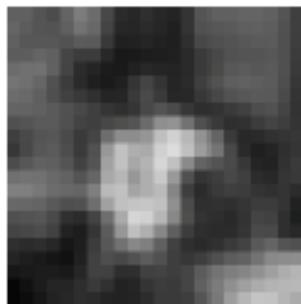
clamp



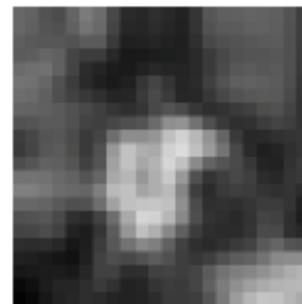
mirror



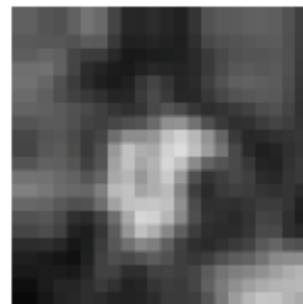
blurred zero



normalized zero



blurred clamp



blurred mirror

Image Extrapolation (Mirroring)

Code

```
J = imread('image.bmp');  
figure; imshow(J);
```



J

Image Extrapolation (Mirroring)

Code

```
boarder = 40;  
[nr,nc,nb] = size(J);  
J_big = zeros(nr+2*boarder, nc + 2*boarder,nb);  
J_big(boarder+1:boarder+nr,boarder+1:boarder+nc,:) = J;
```



Image Extrapolation (Mirroring)

Code

```
for i=1:border,  
    for j=1:border,  
        J_big(i,j,:) = J(border-i+1,border-j+1,:);  
    end  
end
```

Mirroring with respect to the border

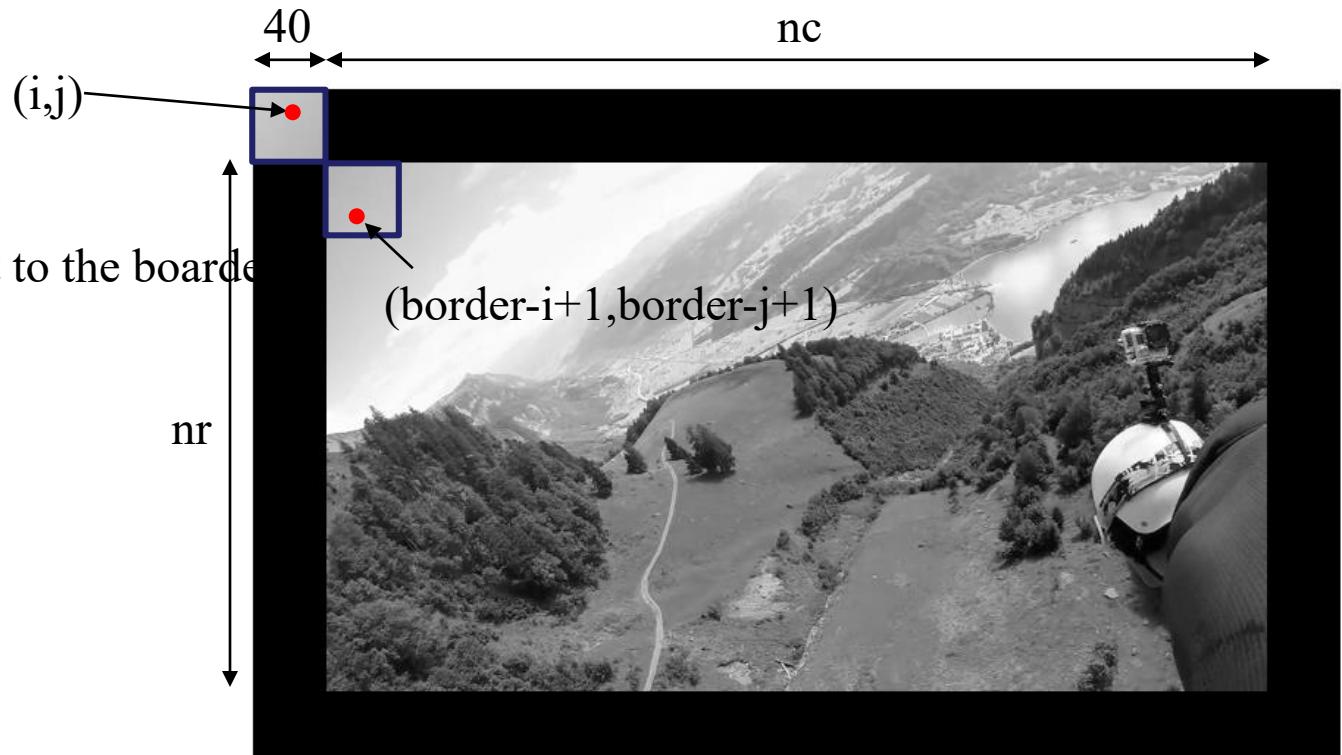


Image Extrapolation (Mirroring)

Code

```
for i=1:boarder,  
  for j=border+1:border+nc,  
    J_big(i,j,:) = J(border-i+1,j-border,:);  
  end  
end
```

Mirroring with respect to the border

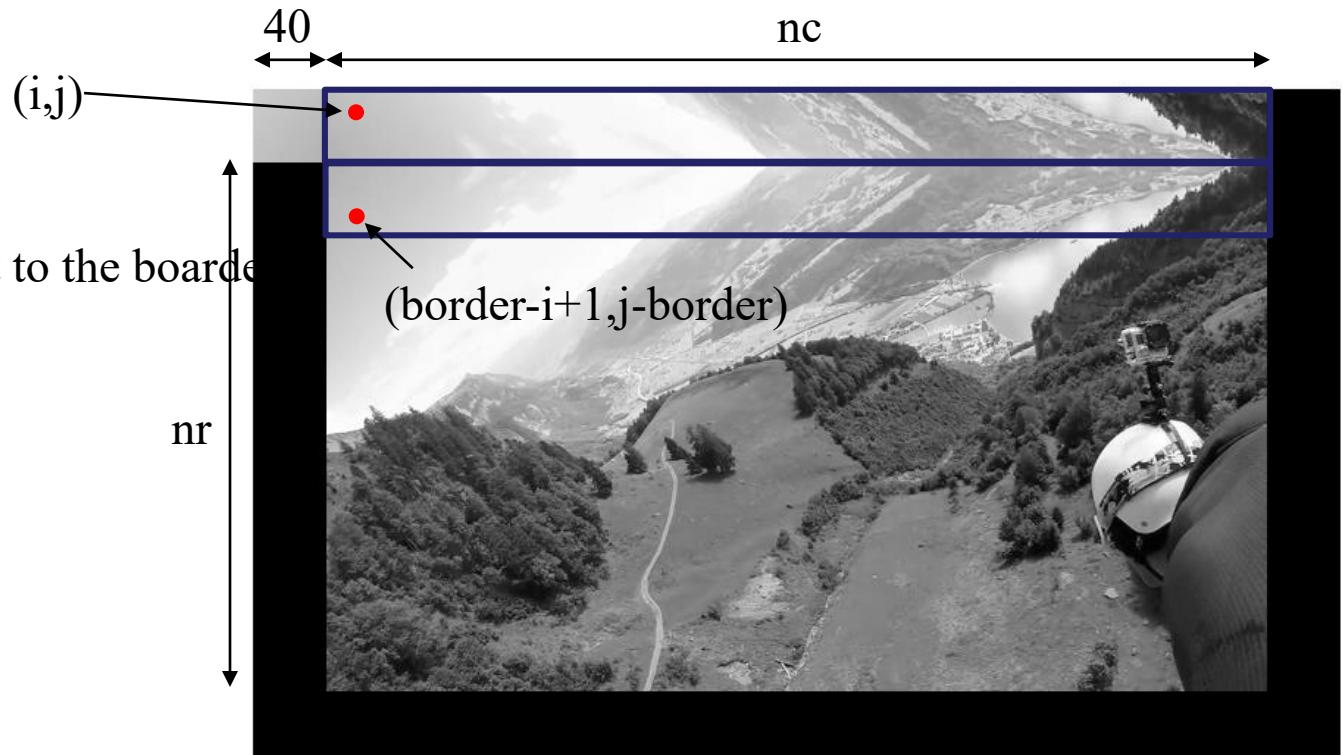


Image Extrapolation (Mirroring)

Code

```
for i=nr+border+1:border*2+nr,  
    for j=border+1:border+nc,  
        J_big(i,j,:) = J(2*nr-i+border+1,j-border,:);  
    end  
end
```

Mirroring with respect to the border

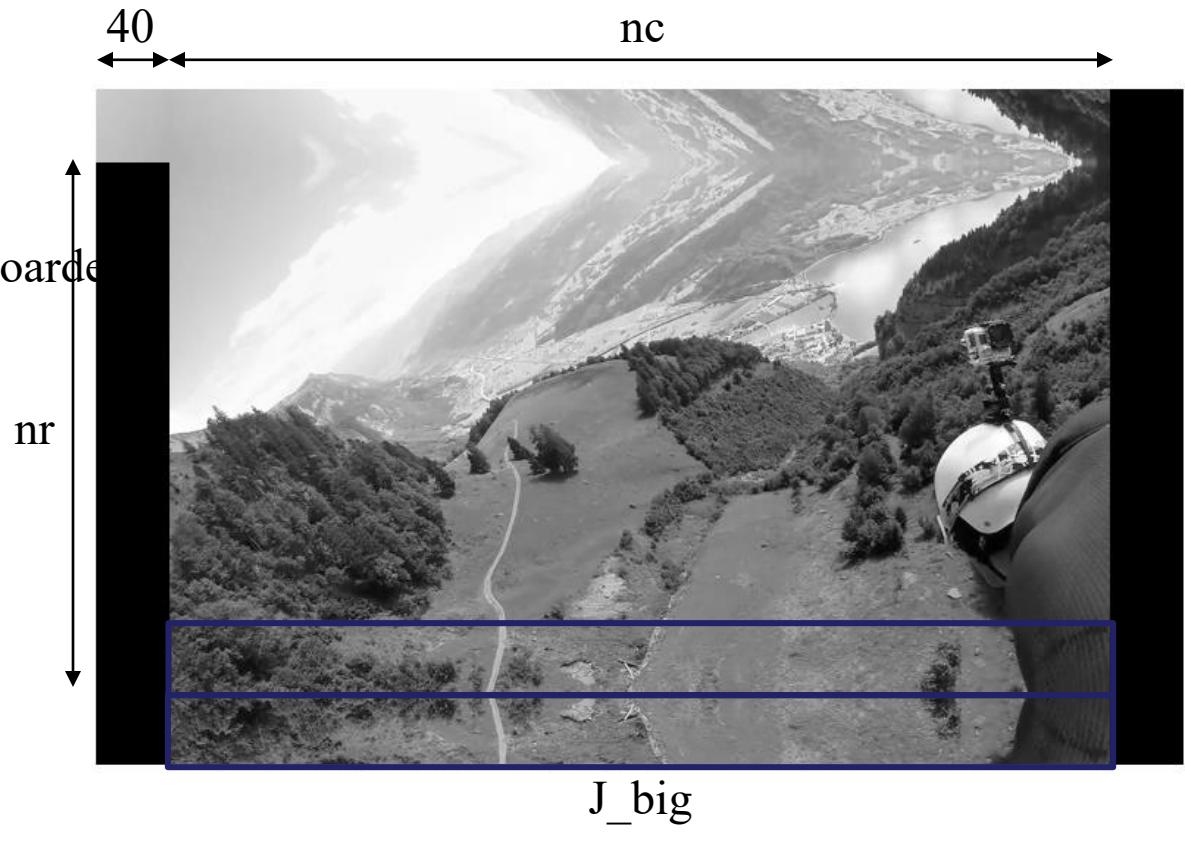
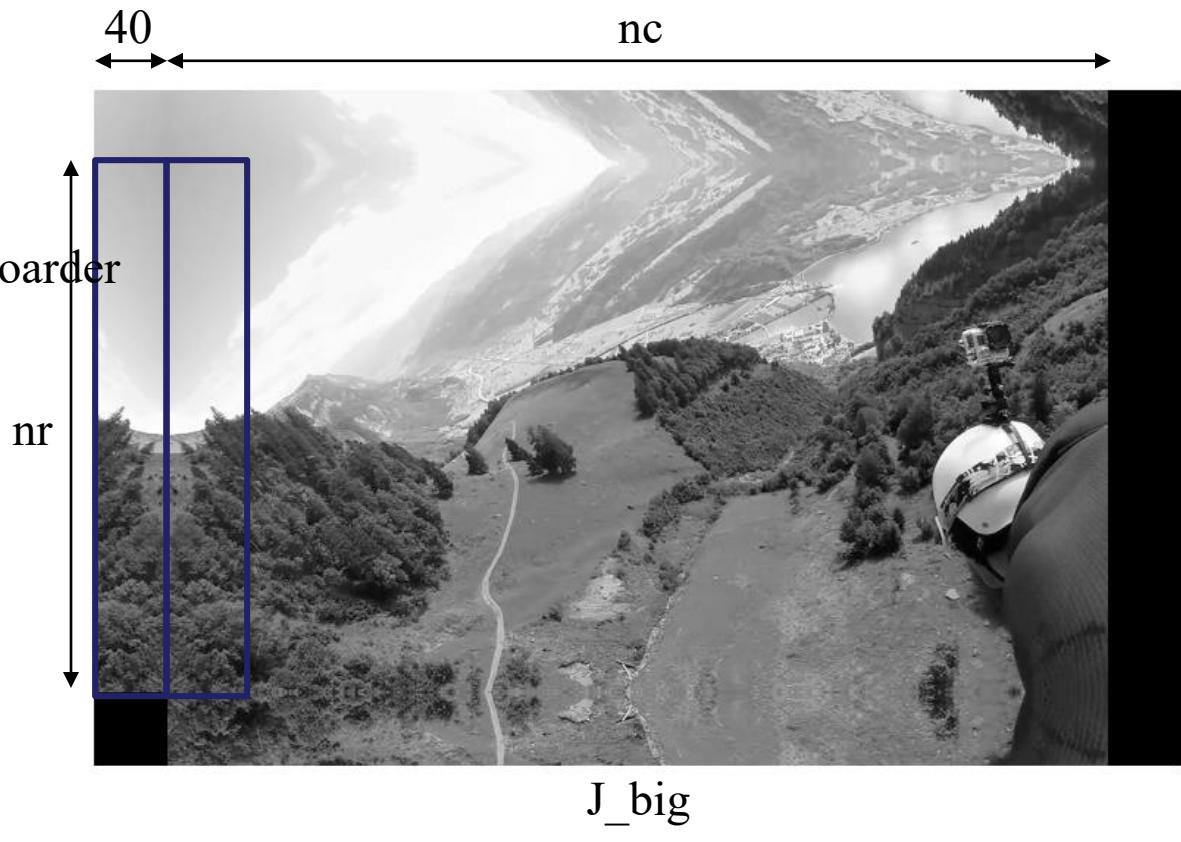
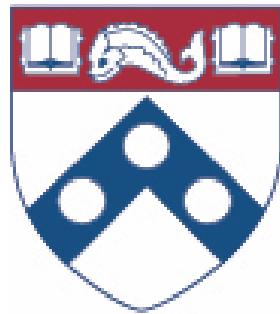


Image Extrapolation (Mirroring)

Code

```
for i=border+1:border+nr;  
    for j=1:border,  
        J_big(i,j,:) = J(i-border,border-j+1,:);  
    end  
end
```





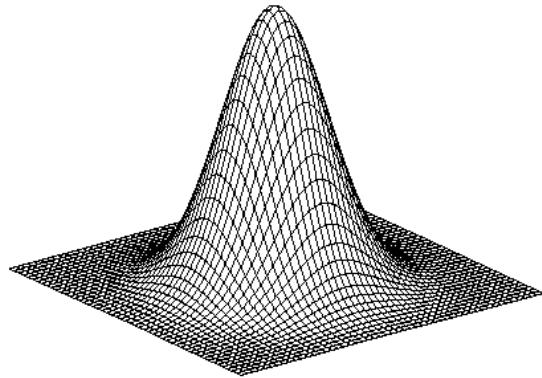
Penn
Engineering

ONLINE LEARNING

Video 2.13
Jianbo Shi

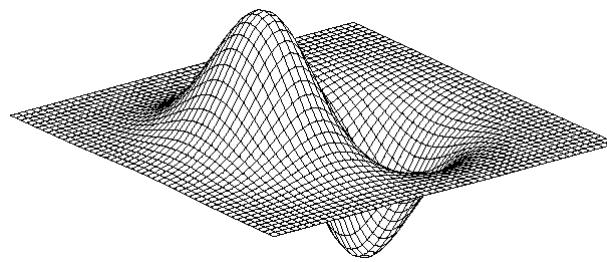
Examples of image operation as convolution

2D filters, more on this later...



Gaussian

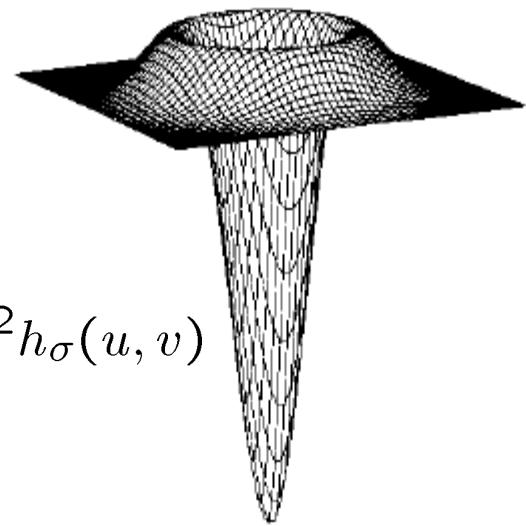
$$h_\sigma(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian

$$\frac{\partial}{\partial x} h_\sigma(u, v)$$

Laplacian of Gaussian

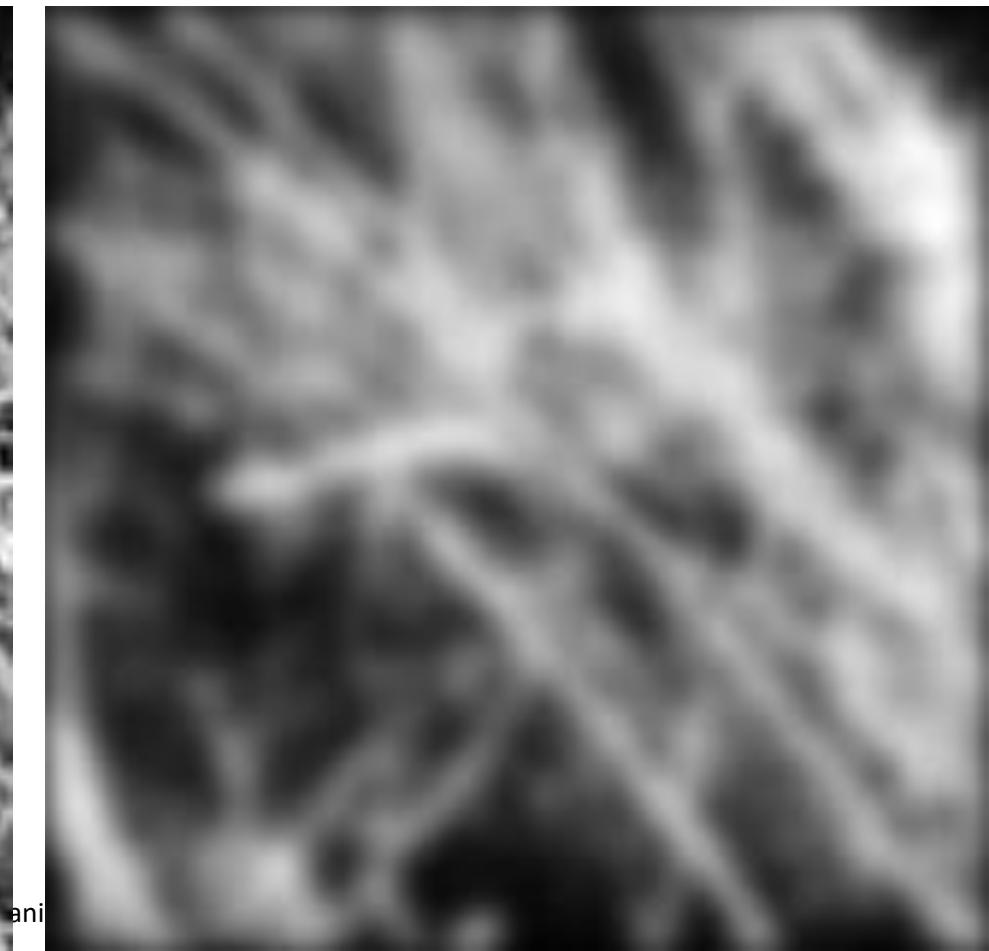


$$\nabla^2 h_\sigma(u, v)$$

- is the **Laplacian** operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Smoothing with a Gaussian

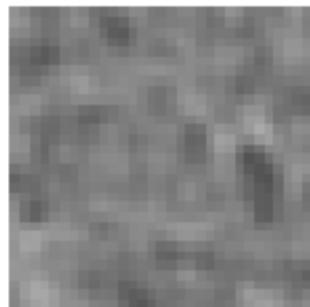
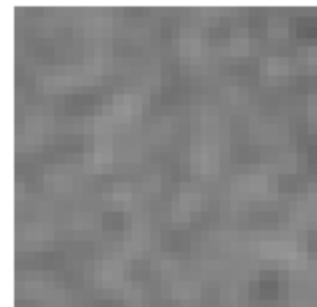
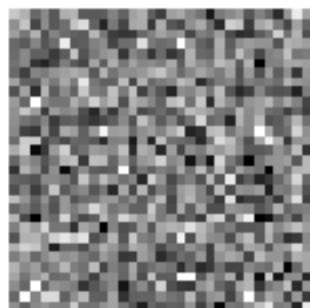
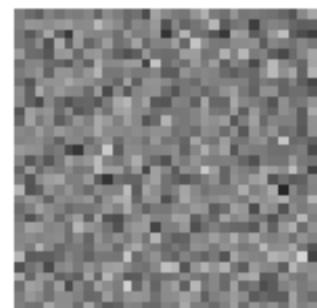


$\sigma=0.05$

$\sigma=0.1$

$\sigma=0.2$

no
smoothing



$\sigma=1$ pixel



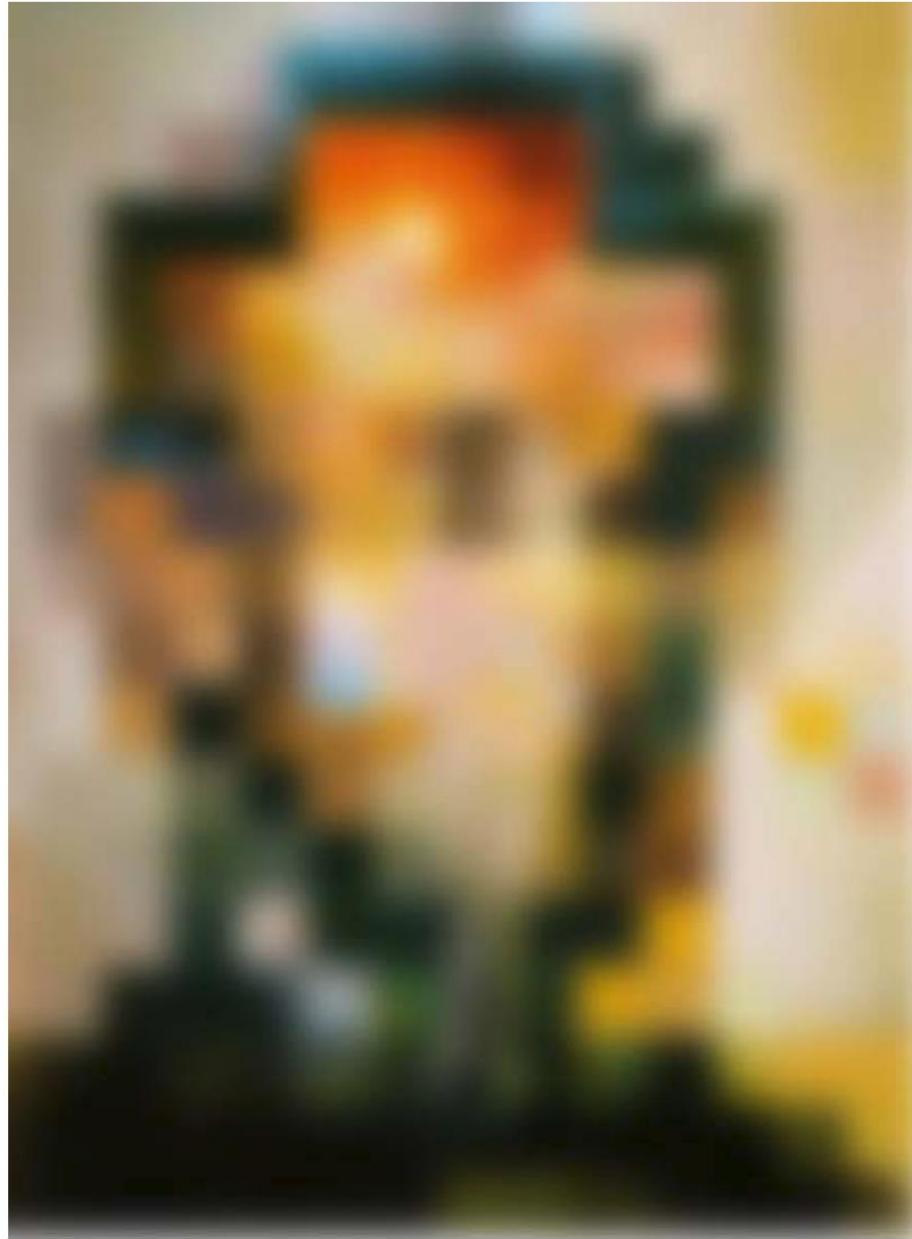
$\sigma=2$ pixels

The effects of smoothing

Each row shows smoothing with gaussians of different width; each column shows different realizations of an image of gaussian noise.



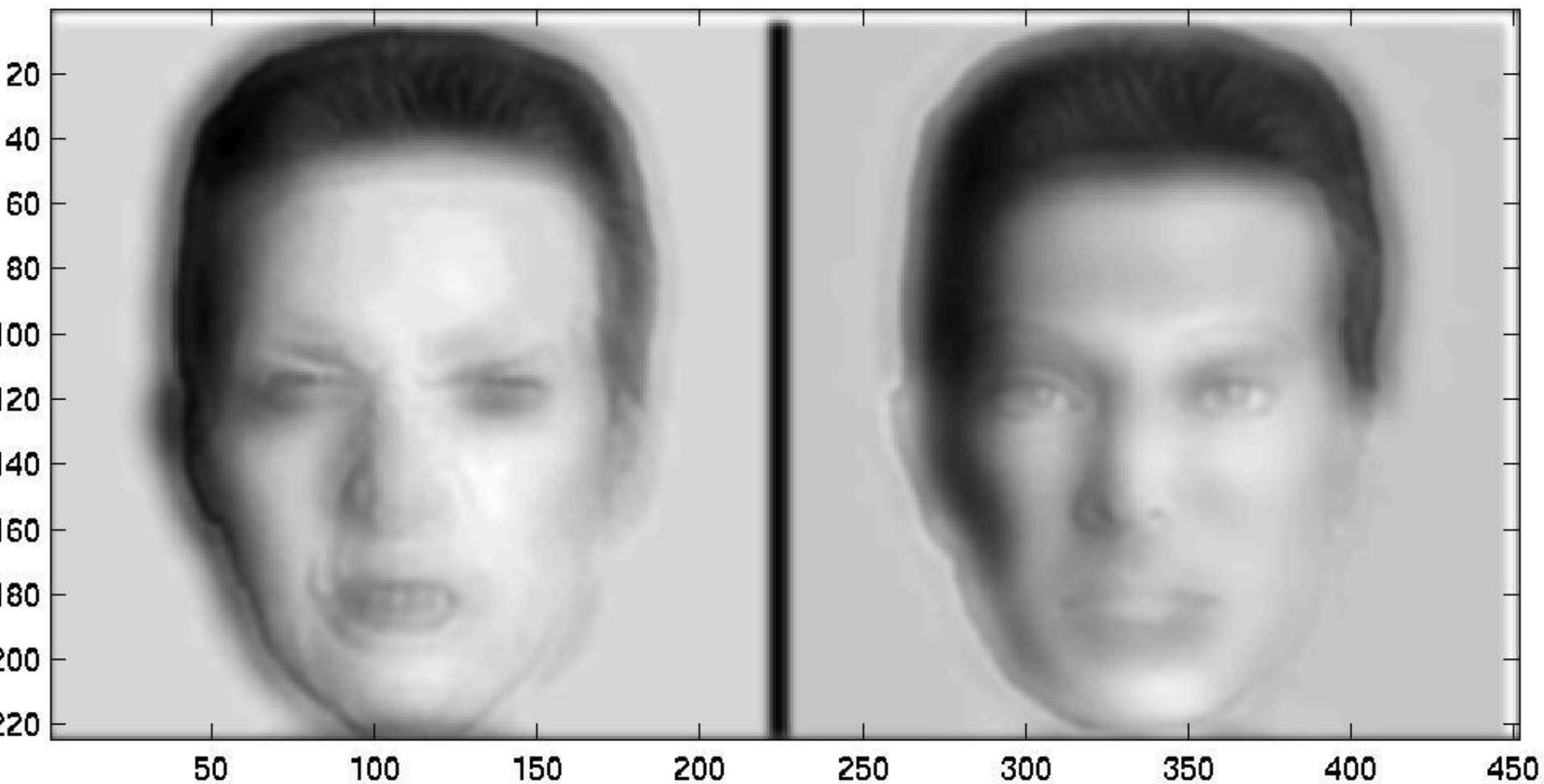
Salvador Dalí, "Gala Contemplating the Mediterranean Sea, which at 30 meters becomes the portrait of Abraham Lincoln", 1976

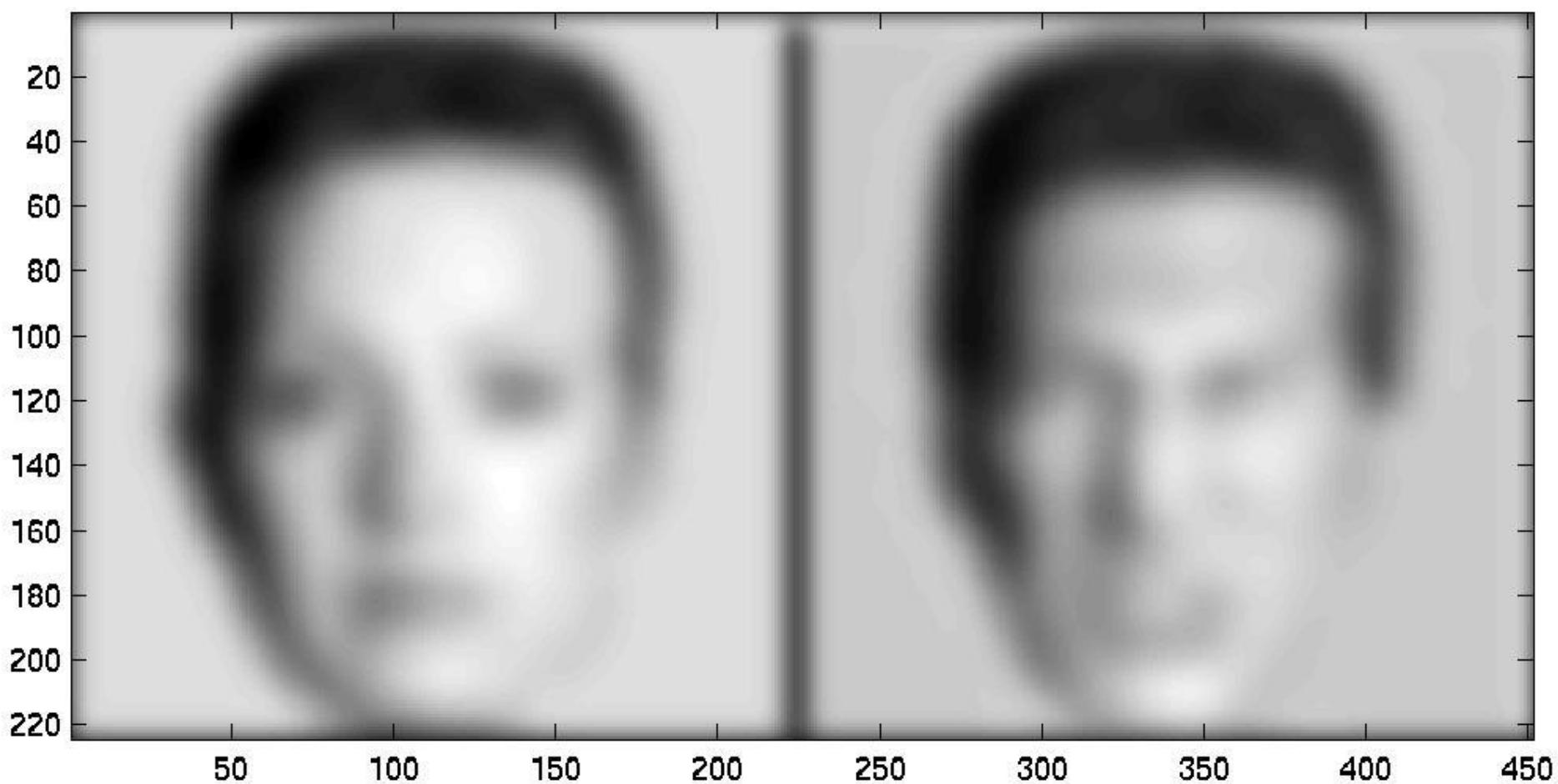


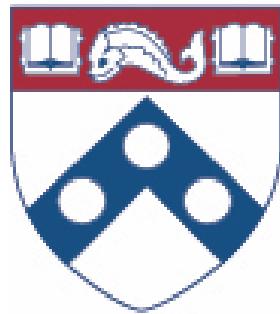
Salvador Dalí, "Gala Contemplating the Mediterranean Sea, which at 30 meters becomes the portrait of Abraham Lincoln", 1976

Image smoothing can remove noise, and also ...









Penn
Engineering

ONLINE LEARNING

Video 2.14
Jianbo Shi

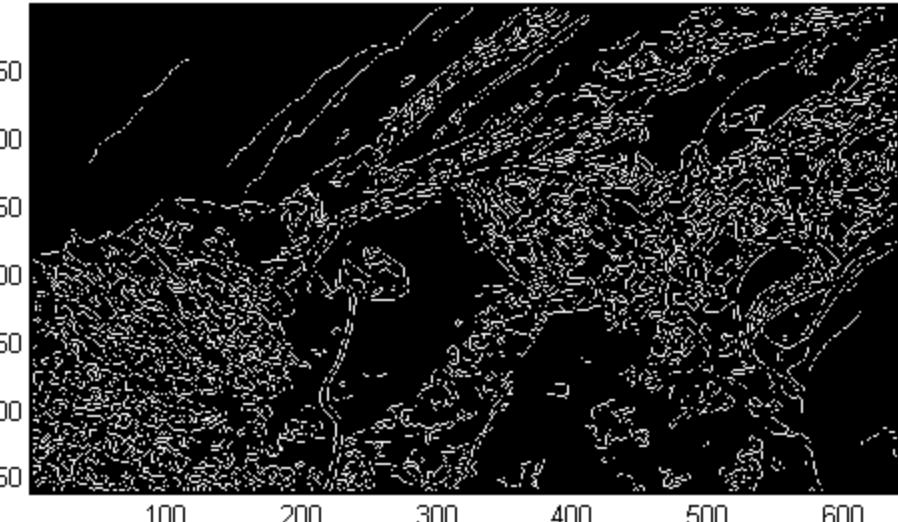
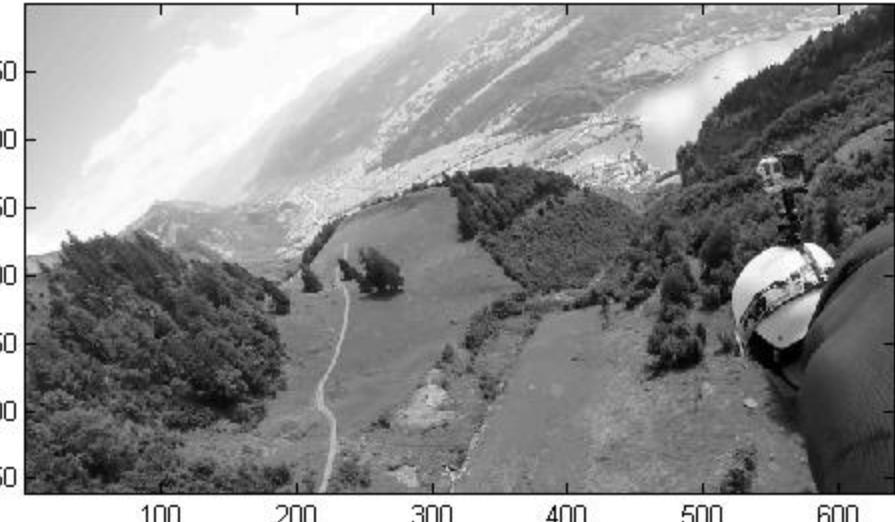
Edge Detection

Code

```
Jb = rgb2gray(J);
```

```
imagesc(Jb);axis image; colormap(gray);
```

```
bw = edge(Jb,'canny');
```



Canny Edge Detection

$$B(i,j) = \begin{cases} 1 & \text{if } I(i,j) \text{ is edge} \\ 0 & \text{if } I(i,j) \text{ is not edge} \end{cases}$$

Objective: to localize edges given an image.



Original image, I

Binary image indicating edge pixels

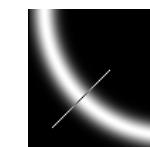


Edge map image, B



Canny Edge Detection

1. Filter image by derivatives of Gaussian
2. Compute magnitude of gradient
3. Compute edge orientation
4. Detect local maximum
5. Edge linking



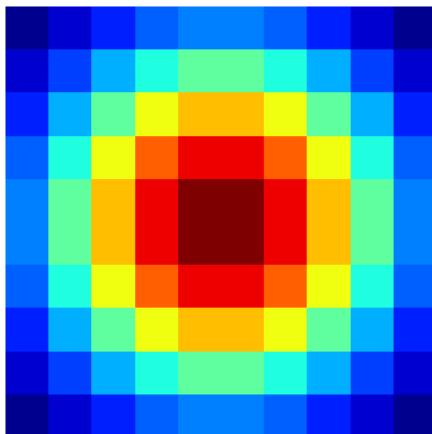
1) Compute Image Gradient

the first order derivative of Image I in x, and in y direction

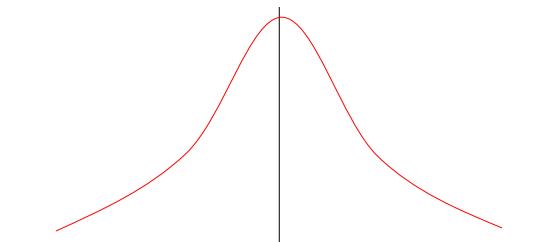
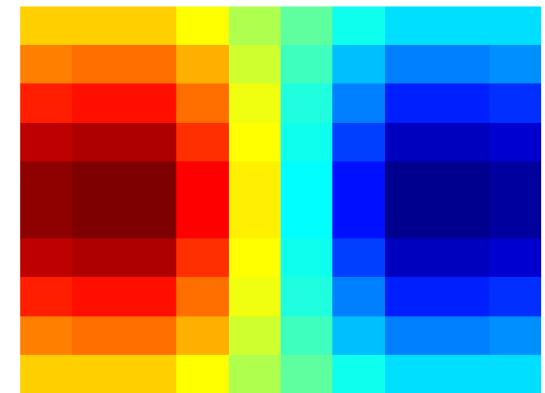
Edge Detection, Step 1, Filter out noise and compute derivative:

$$\left(\frac{\delta}{\delta x} \otimes G \right)$$

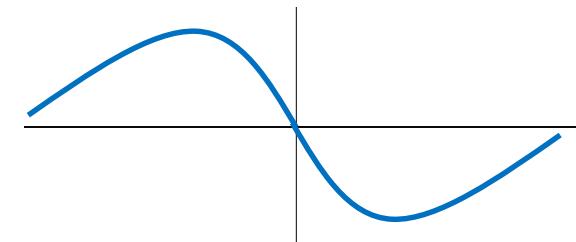
Gradient of Gaussian



$$\frac{\delta}{\delta x} \longrightarrow$$



$$\frac{\delta}{\delta x} \longrightarrow$$



Edge Detection, Step 1, Filter out noise and compute derivative:

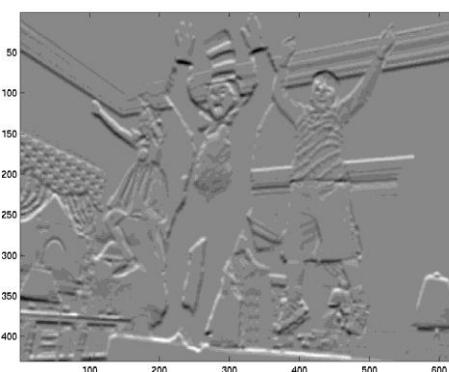
Image

$$\otimes \left(\frac{\delta}{\delta x} \otimes G \right)$$

Smoothed Derivative



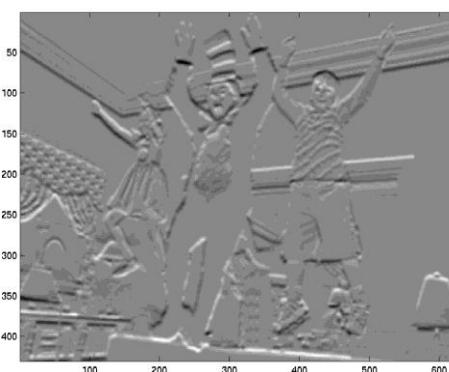
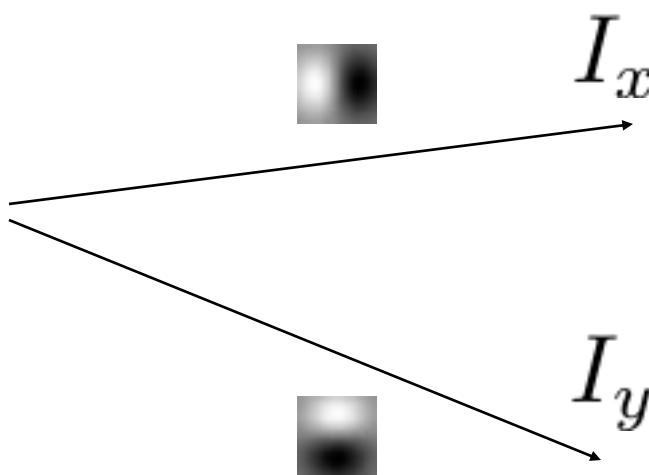
Two derivative filters are shown as small grayscale kernels. The top filter, labeled I_x , is a 3x3 kernel with a central value of 1 and negative values along the horizontal axis. The bottom filter, labeled I_y , is a 3x3 kernel with a central value of 1 and negative values along the vertical axis.



Edge Detection, Step 1, Filter out noise and compute derivative:

In matlab:

```
>> [dx,dy] = gradient(G); % G is a 2D gaussain  
>> Ix = conv2(I,dx,'same'); Iy = conv2(I,dy,'same');
```

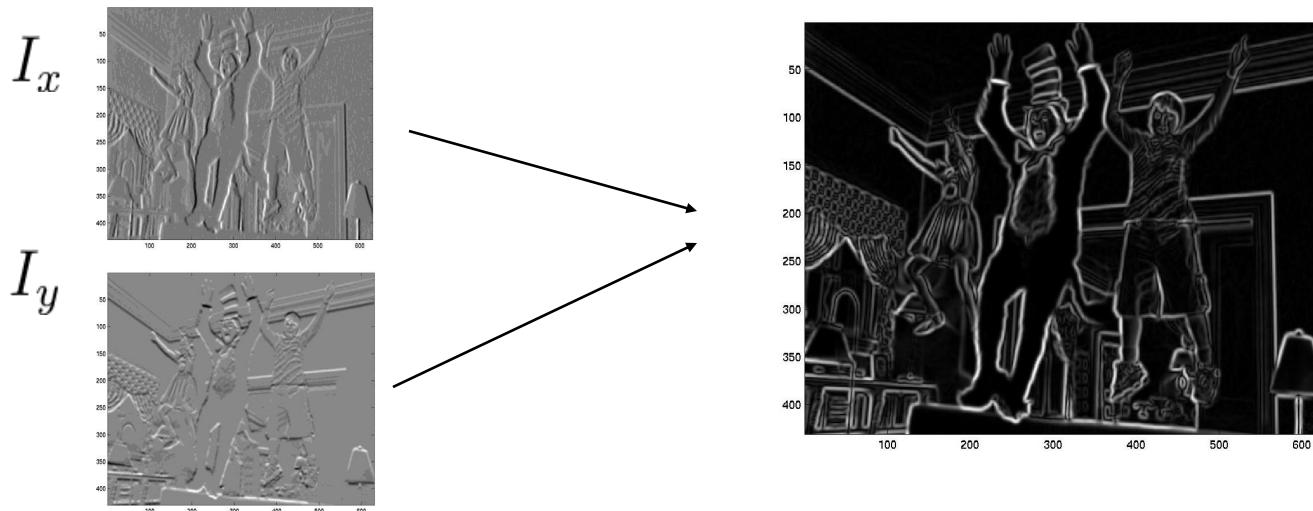


Edge Detection: Step 2

Compute the magnitude of the gradient

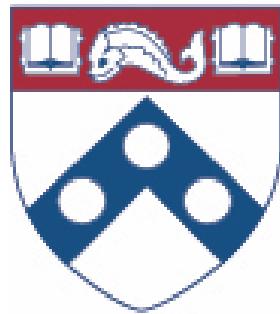
In Matlab:

```
>> Im = sqrt(Ix.*Ix + Iy.*Iy);
```



We know roughly where are the edges, but we need their precise location.





Penn
Engineering

ONLINE LEARNING

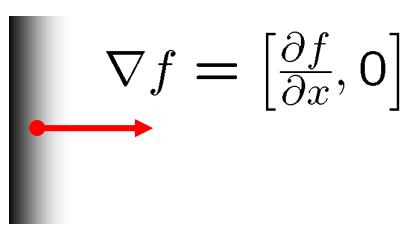
Video 2.15
Jianbo Shi

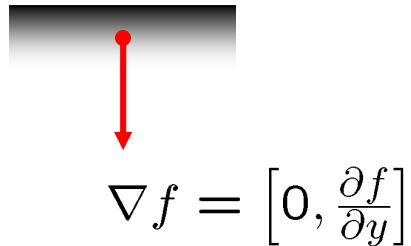
Finding the orientation of the edge

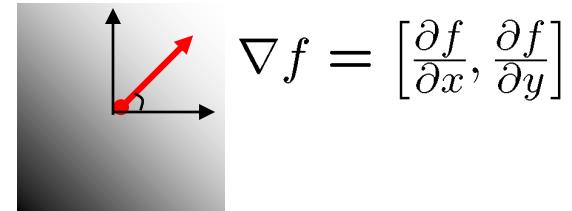
- The gradient of an image:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$

- The gradient points in the direction of most rapid change in intensity


$$\nabla f = \left[\frac{\partial f}{\partial x}, 0 \right]$$


$$\nabla f = \left[0, \frac{\partial f}{\partial y} \right]$$


$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]$$

- The image gradient direction is given by:

$$\theta = \tan^{-1} \left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x} \right)$$

- how does this relate to the direction of the edge?

$$\theta_{edge} = \tan^{-1} \left(- \frac{\delta f}{\delta x} / \frac{\delta f}{\delta y} \right)$$

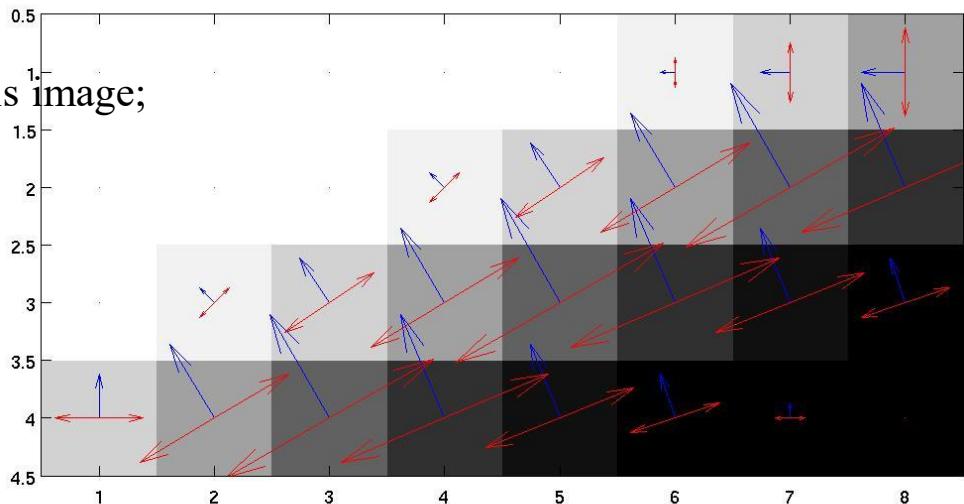
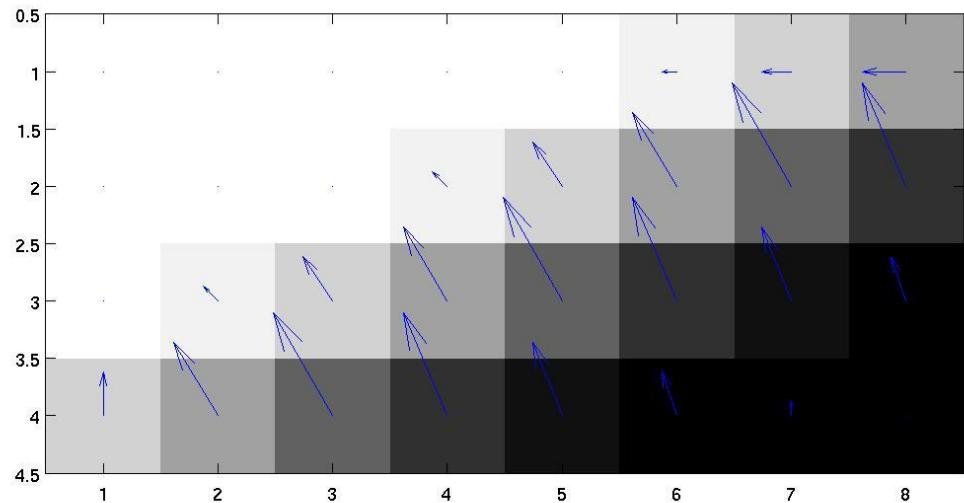
```
%% define image gradient operator
dy = [1;-1];
dx = [1,-1];
```

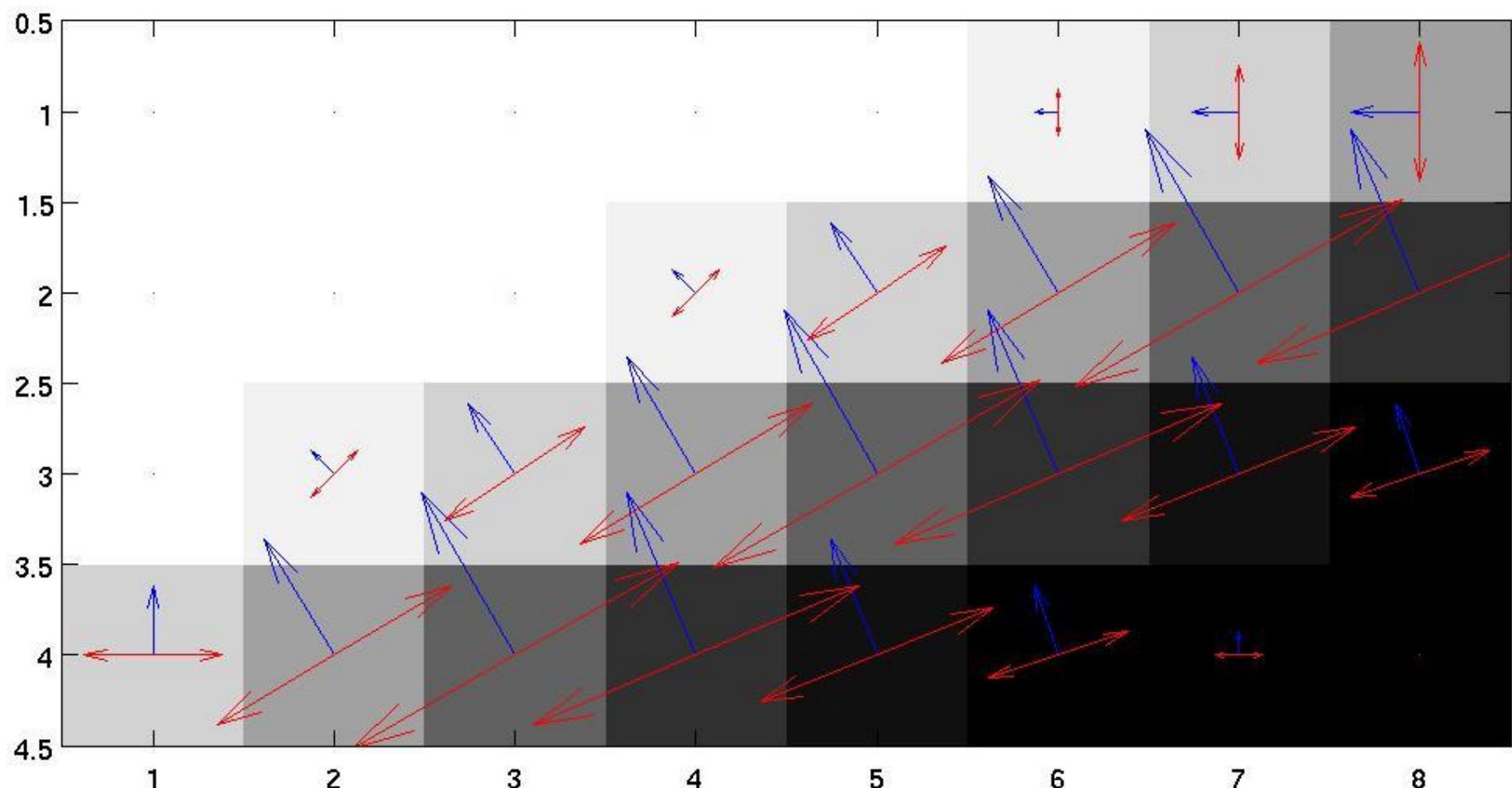
```
%% compute image gradient in x and y
AA_y = conv2(AA,dy,'same');
AA_x = conv2(AA,dx,'same');
```

```
Jy = AA_y(1:end-2,2:end-1);
Jy(1,:) = 0;
```

```
Jx = AA_x(2:end-1,1:end-2);
Jx(:,1) = 0;
```

```
%% display the image gradient flow
figure(3);clf;imagesc(J);colormap(gray);axis image;
hold on;
quiver(Jx,Jy);
quiver(-Jy,Jx,'r');
quiver(Jy,-Jx,'r');
```





```
[gx,gy] = gradient(J);  
mag = sqrt(gx.*gx+gy.*gy); imagesc(mag);colorbar
```

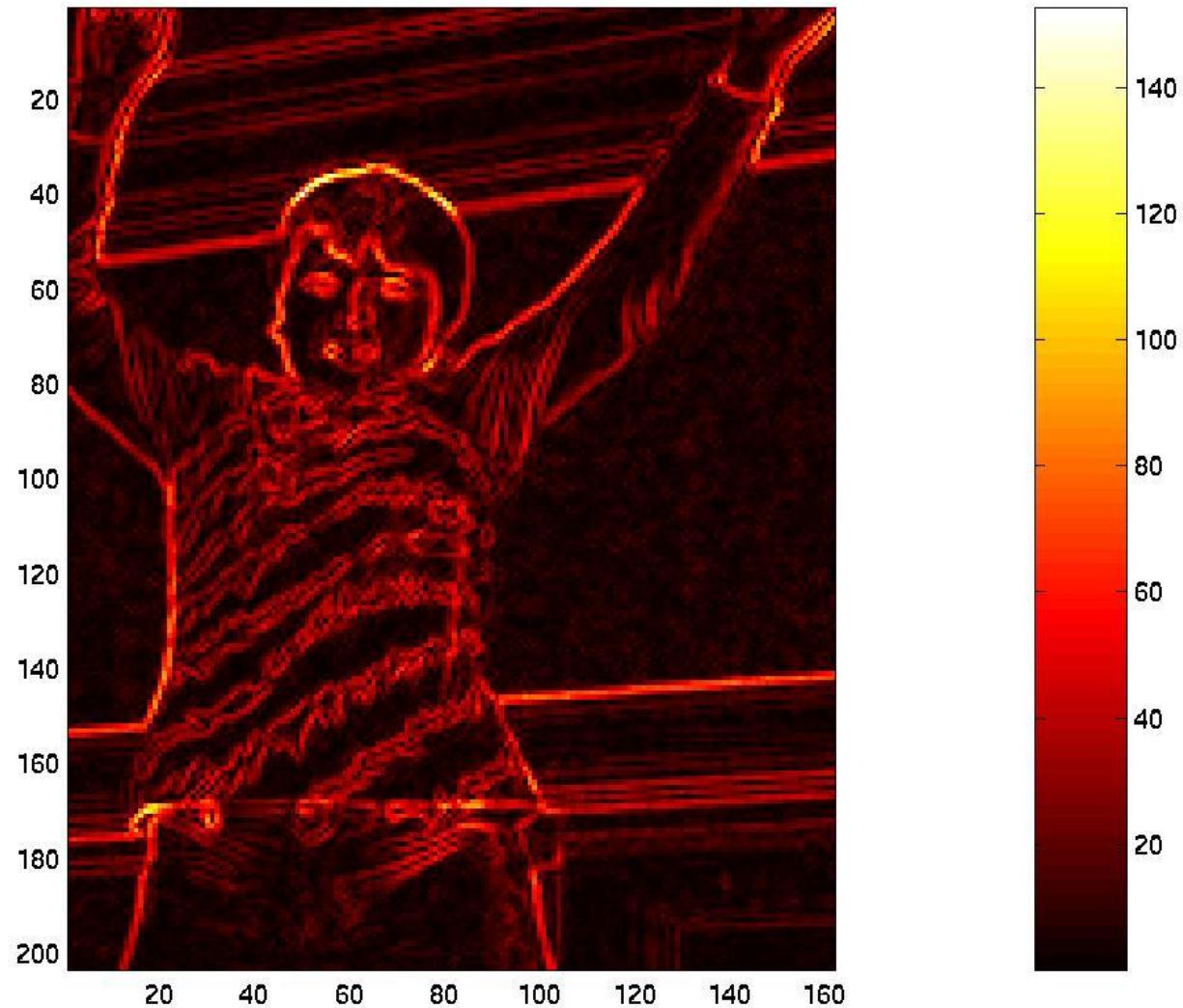
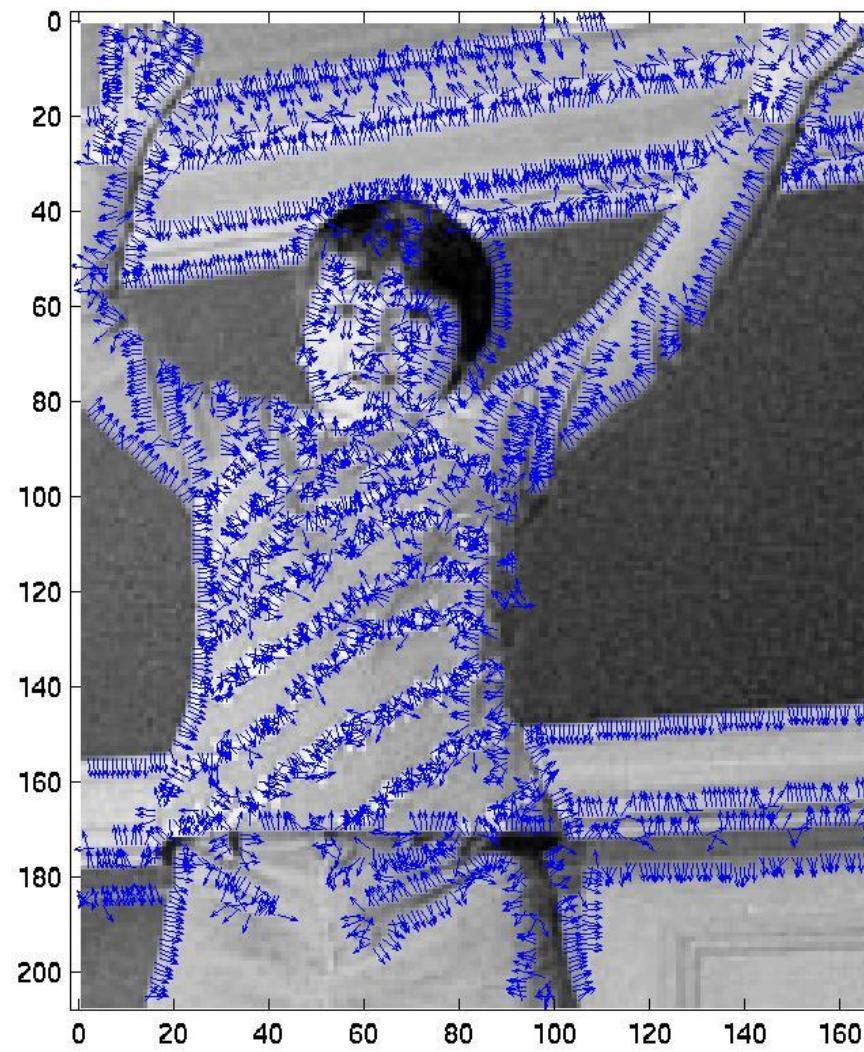
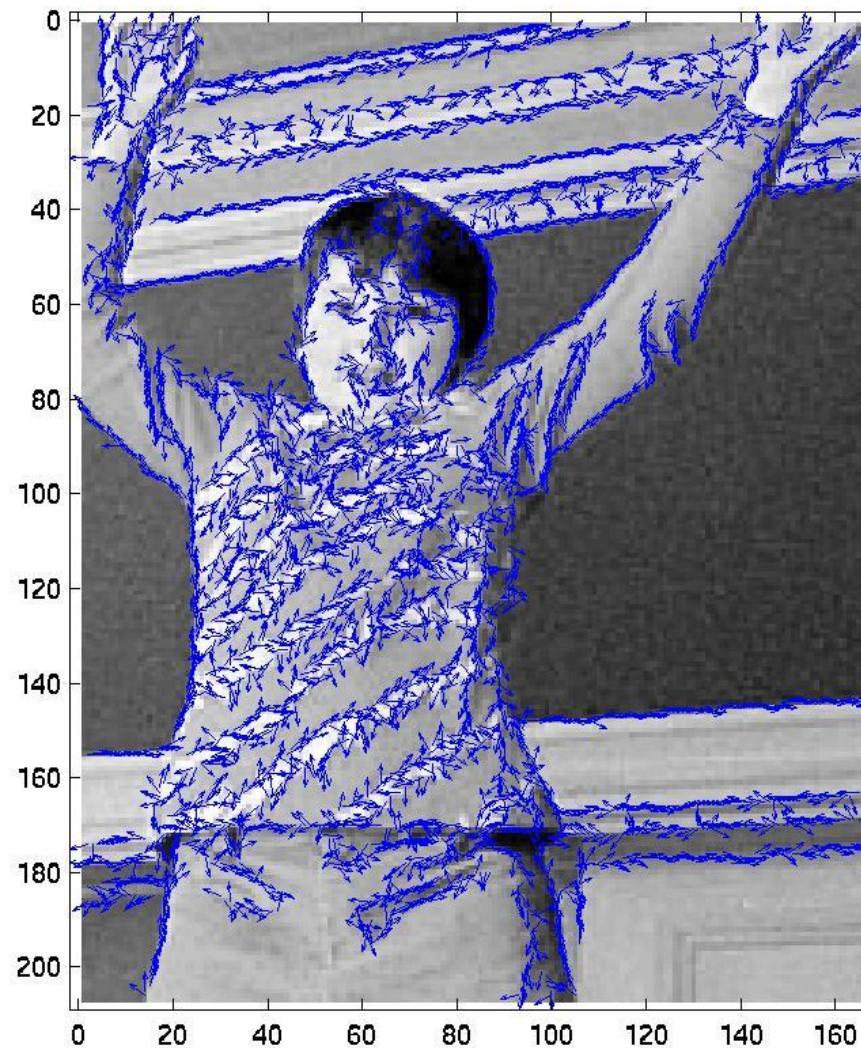


image gradient direction:



Edge orientation direction:



```
[gx,gy] = gradient(J);  
th = atan2(gy,gx); % or you can use:[th,mag] = cart2pol(gx,gy);  
imagesc(th.*(mag>20));colormap(hsv); colorbar
```

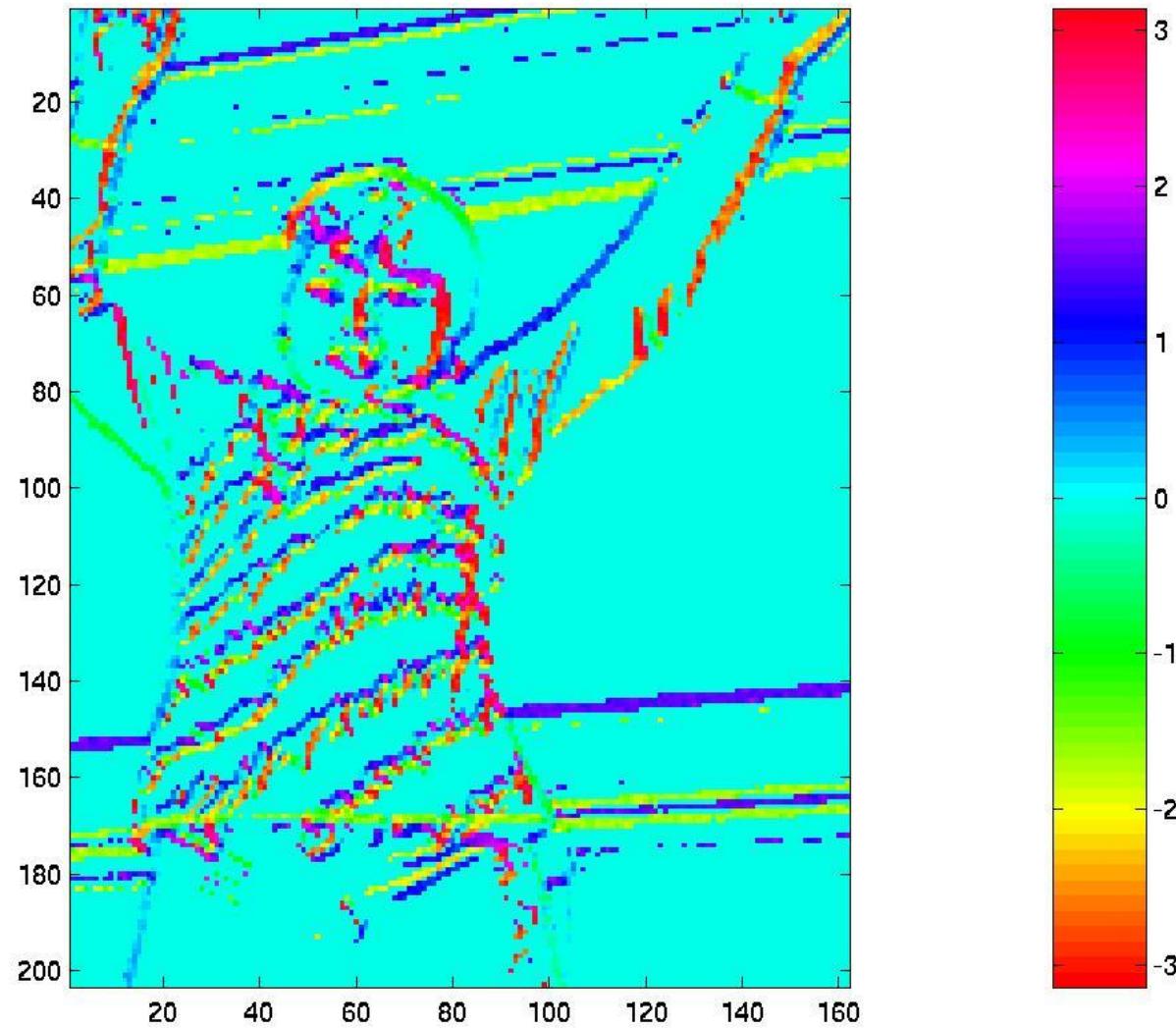
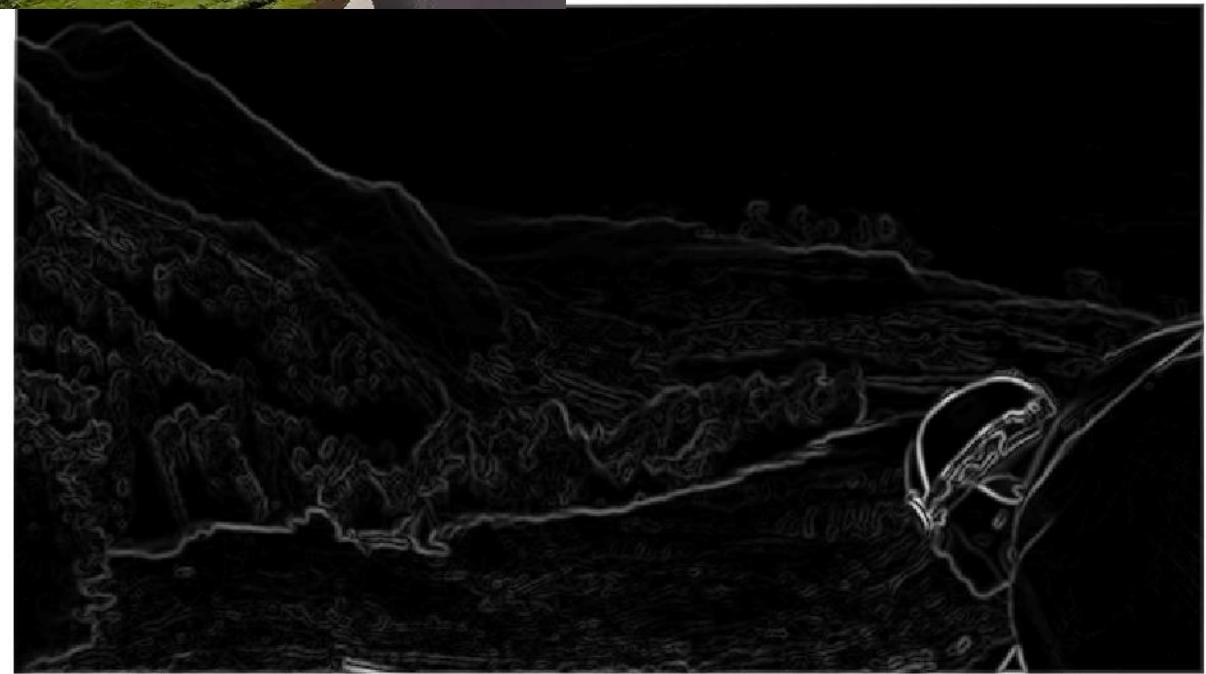
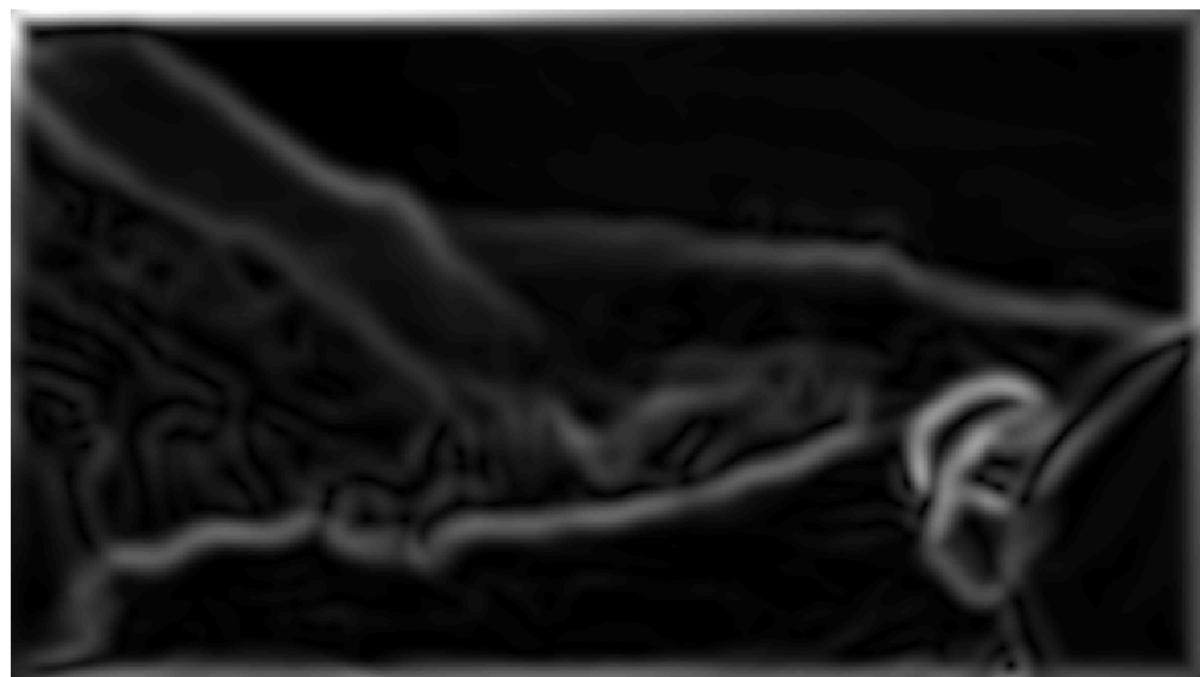
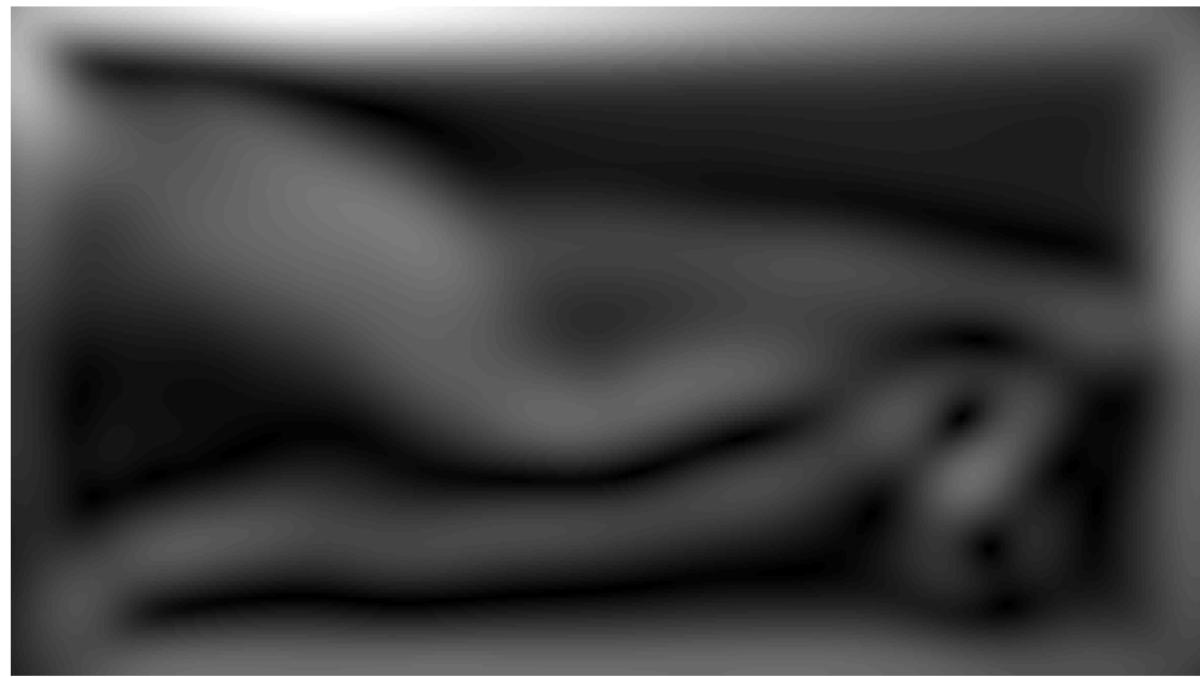




Image Scale₁₇₅



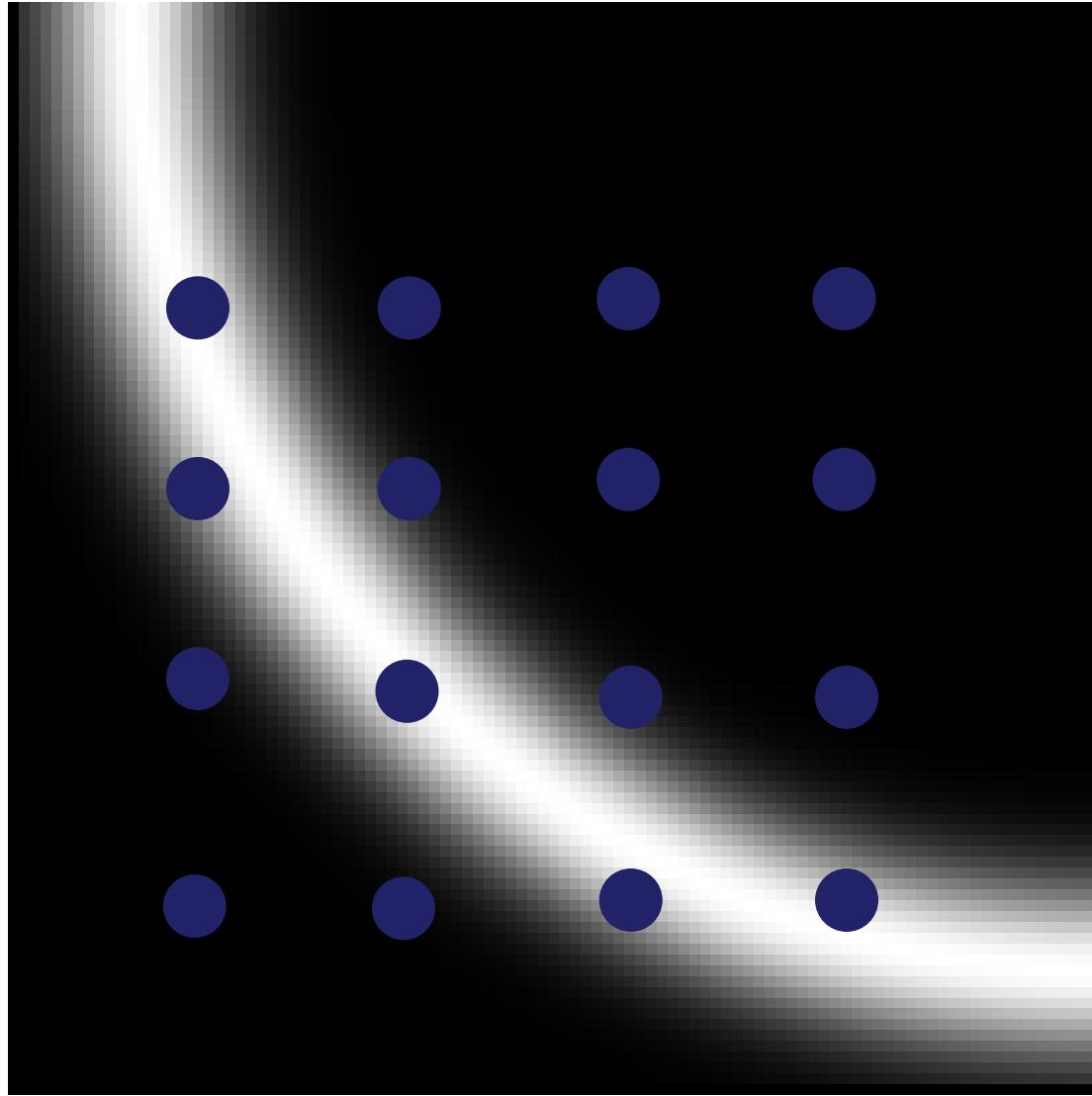




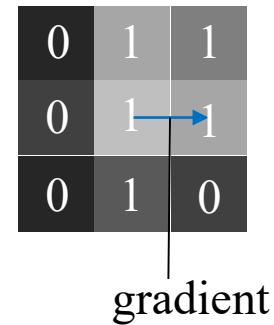
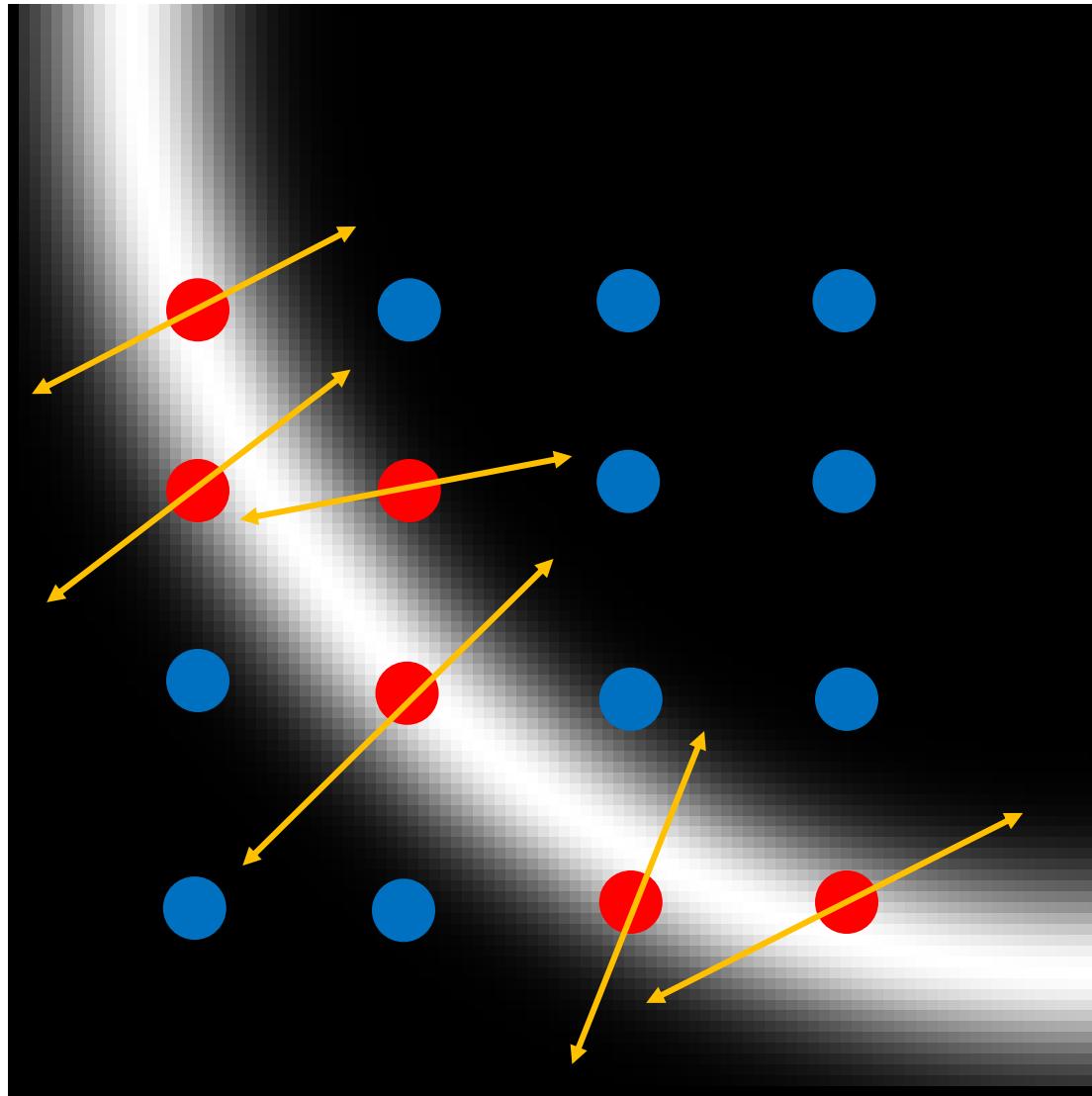


Different scale of image encodes different edge response.

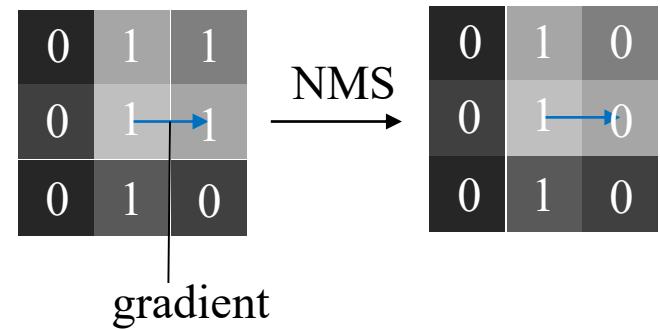
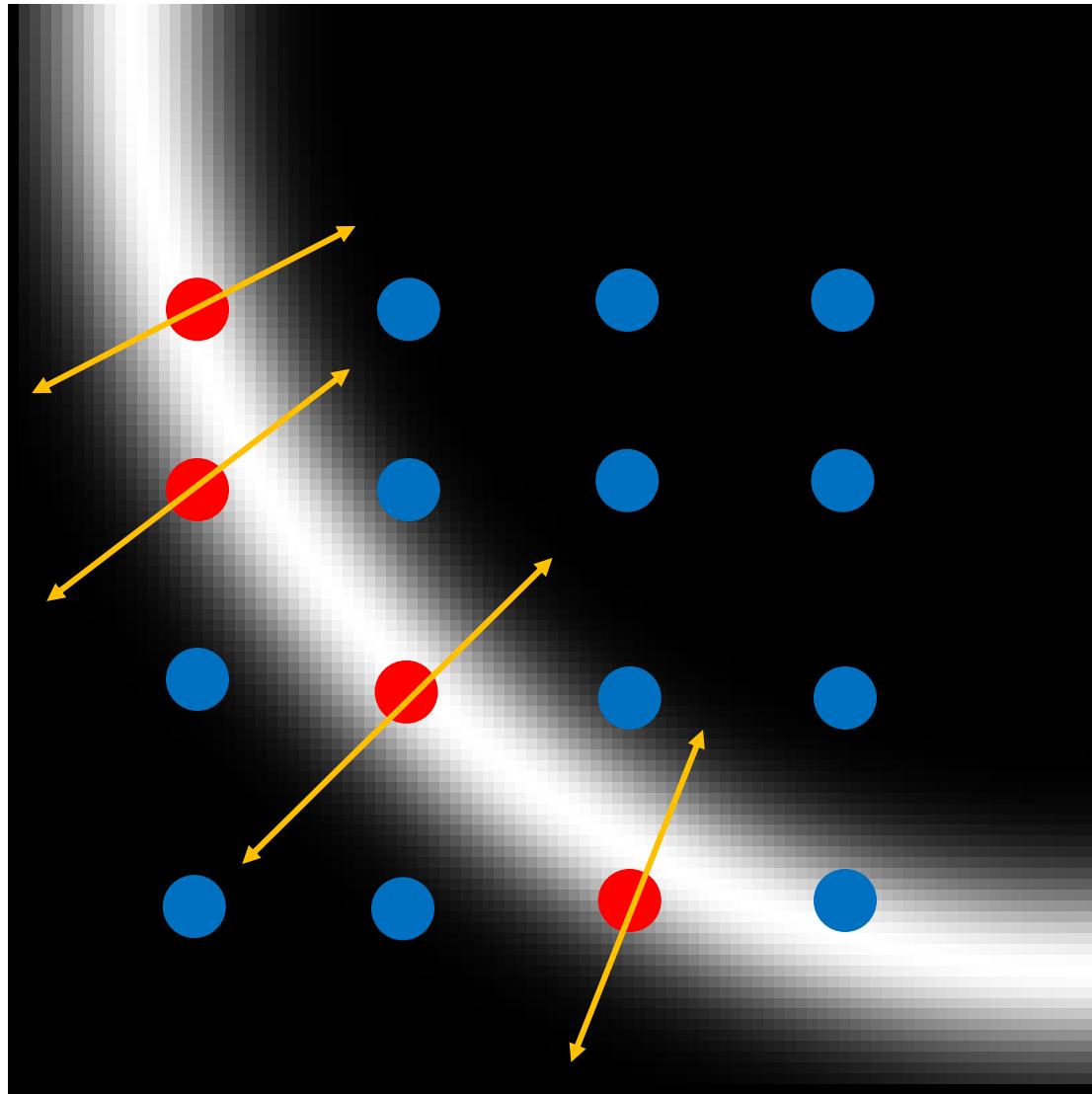
Discretized pixel locations



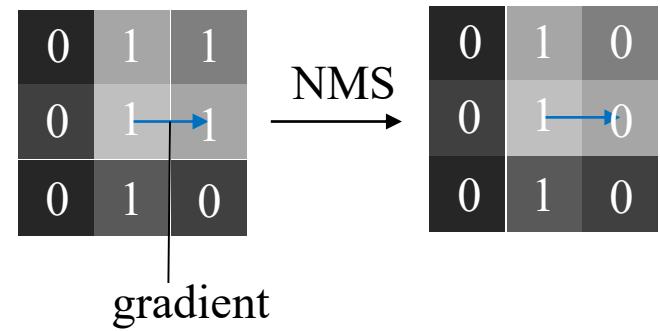
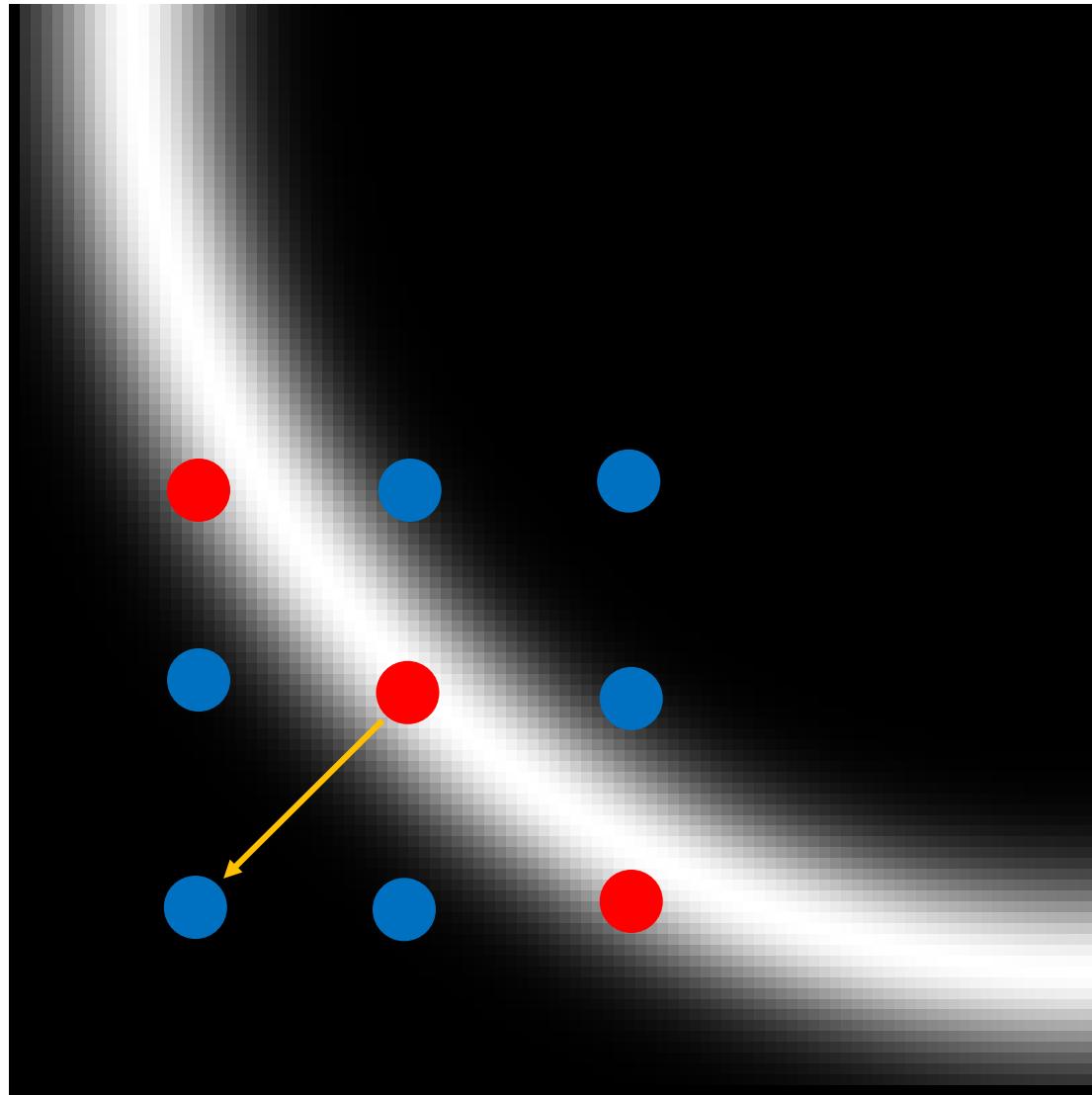
Thresholding

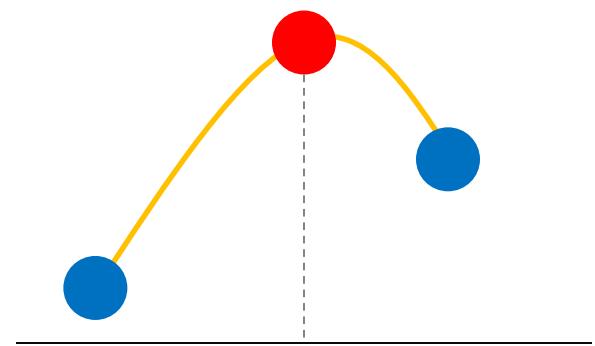
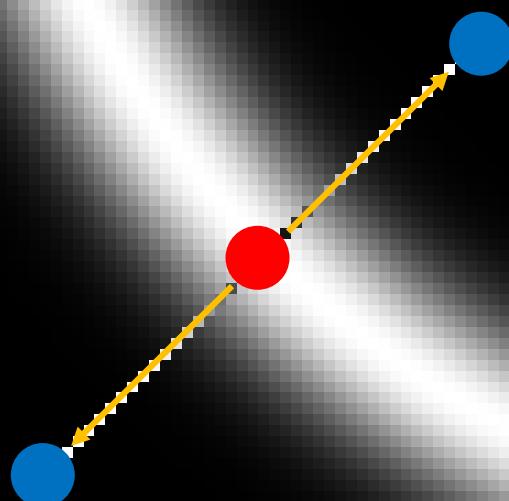


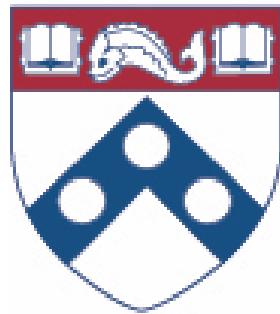
Non-maximum suppression along the line of the gradient



Gradient direction







Penn
Engineering

ONLINE LEARNING

Video 2.16
Jianbo Shi

Edge Detection

Code

```
Jb = rgb2gray(J);
```

```
imagesc(Jb);axis image; colormap(gray);
```

```
bw = edge(Jb,'canny');
```



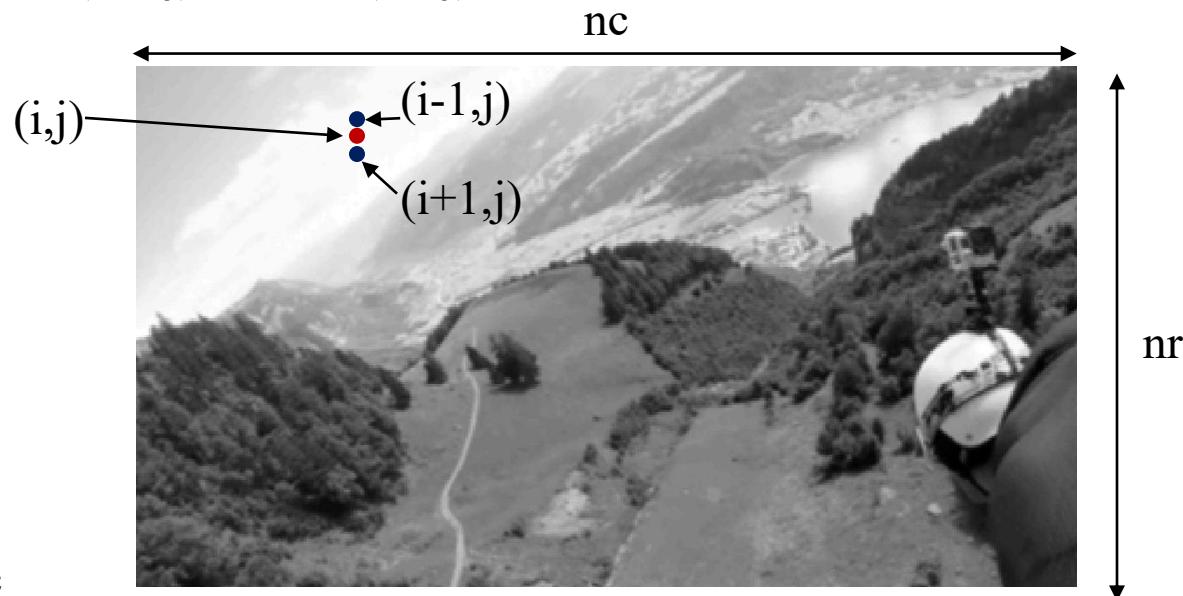
Numerical Image Filtering

Looping through all pixels

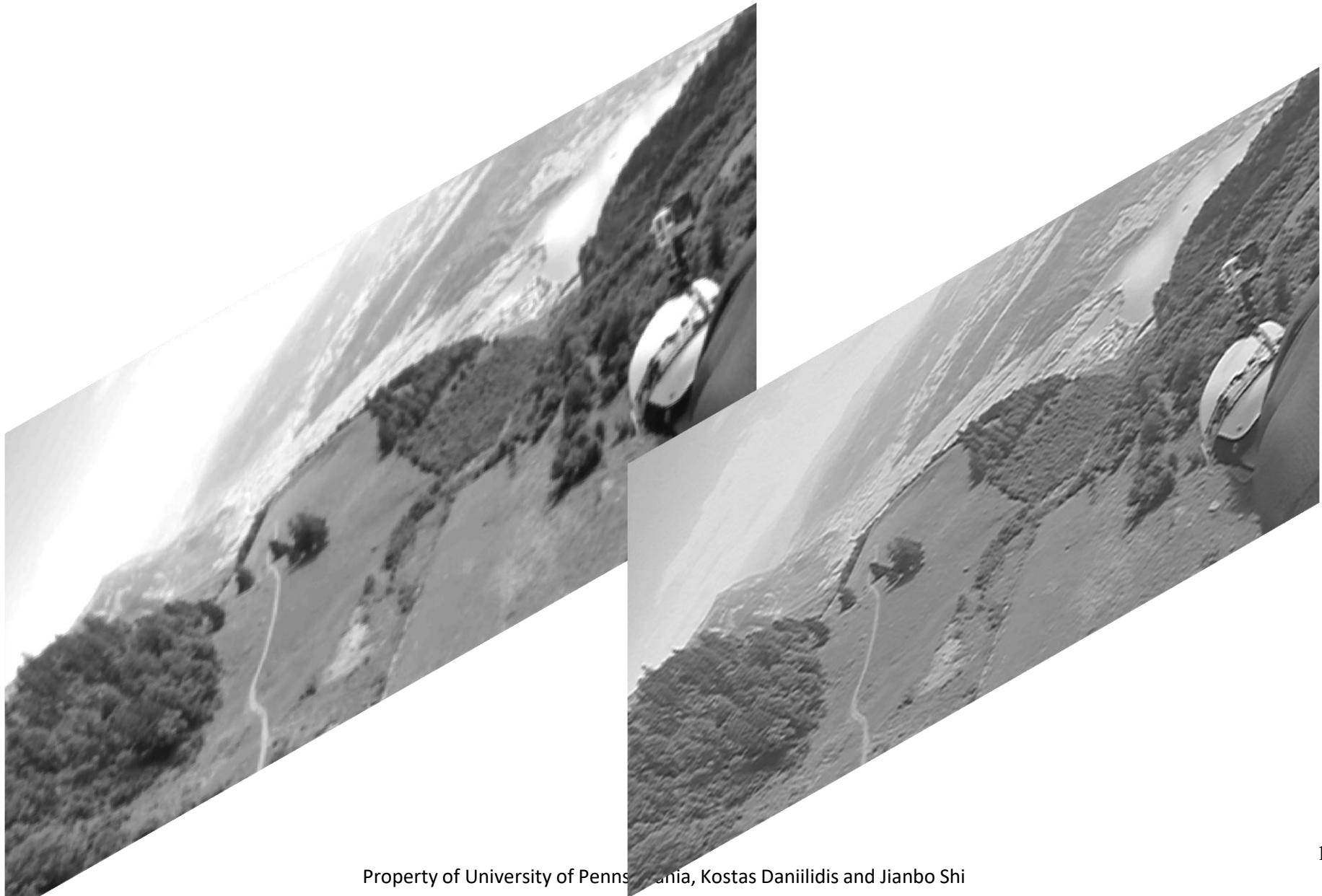
```
[nr,nc] = size(Jb);  
J_out = zeros(nr,nc);  
for i=1:nr,  
    for j=1:nc;  
        if (i<nr) && (i>1),  
            J_out(i,j) = 2*Jb(i,j) - 0.8*Jb(i+1,j) - 0.8*Jb(i-1,j);  
        else  
            J_out(i,j) = Jb(i,j);  
        end  
    end  
end  
figure; imagesc(J_out);  
colormap(gray)
```

Computation time: 0.050154 sec

Filter



Numerical Image Filtering



Convolution without Looping using meshgrid

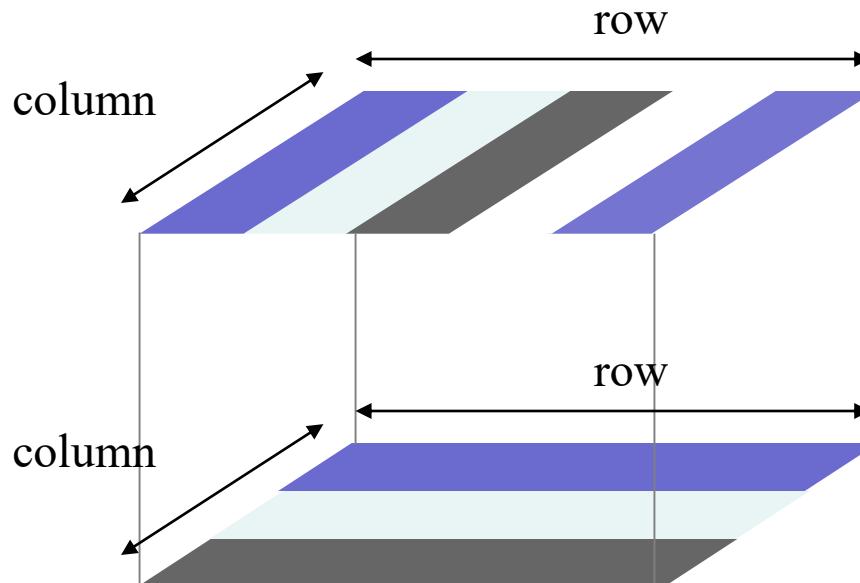
```
>> [x,y] = meshgrid(1:5,1:3)
```

x =

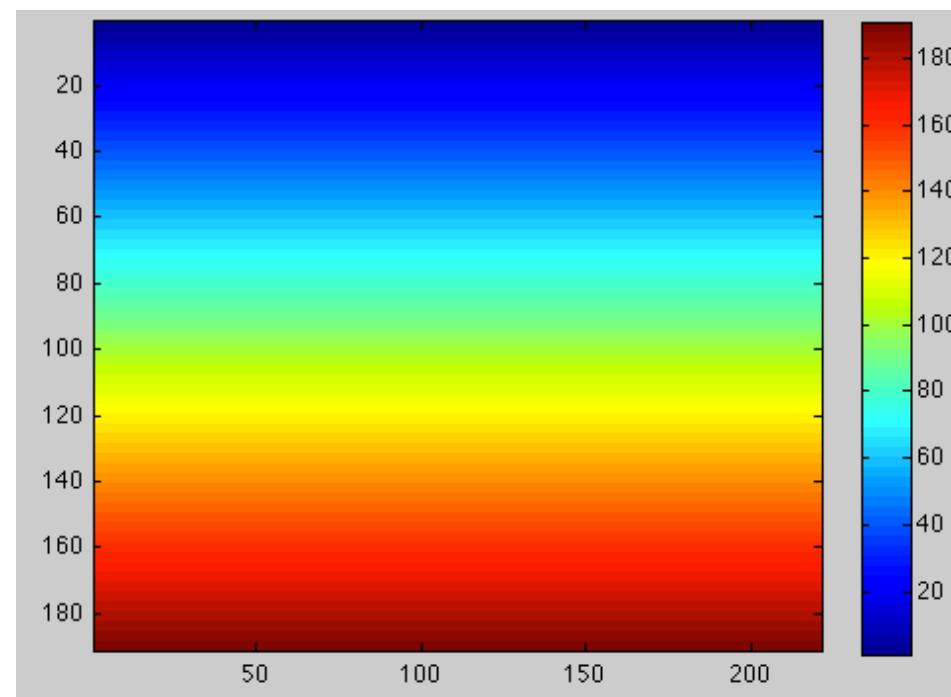
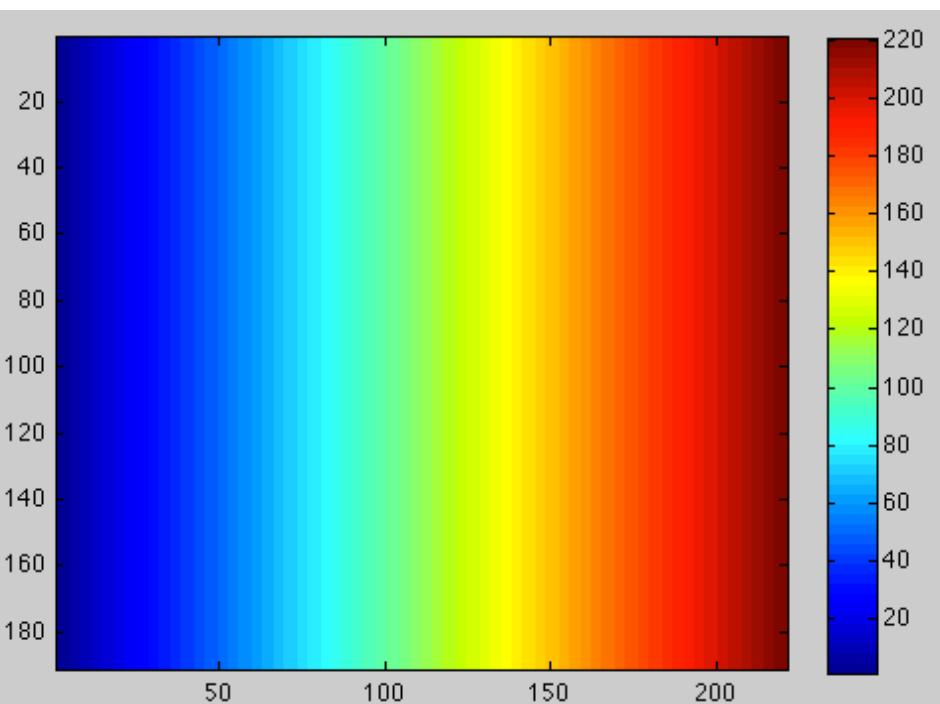
1	2	3	4	5
1	2	3	4	5
1	2	3	4	5

y =

1	1	1	1	1
2	2	2	2	2
3	3	3	3	3



```
[x,y] = meshgrid(1:nc,1:nr);  
figure(1); imagesc(x); axis image; colorbar; colormap(jet);  
figure(2); imagesc(y); axis image; colorbar; colormap(jet);
```



Convolution without Looping using meshgrid

```
[x,y] = meshgrid(1:nc,1:nr);
```

```
y_up = y-1;
```

```
y_down = y+1;
```

```
y_up = min(nr,max(1,y_up)); % keep y_up index within legal range of [1,nr]
```

```
y_down = min(nr,max(1,y_down));
```

```
ind_up = sub2ind([nr,nc],y_up(:,x(:))); % create linear index
```

```
ind_down = sub2ind([nr,nc],y_down(:,x(:)));
```

```
J_out = 2*Jb(:) - 0.8*Jb(ind_up) - 0.8*Jb(ind_down);
```

```
J_out = reshape(J_out, nr, nc);
```

```
figure; imagesc(J_out);colormap(gray)
```

Computation time: 0.024047 sec



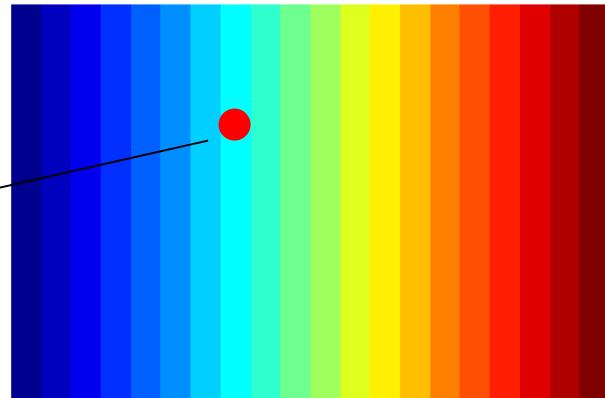
Convolution without Looping using meshgrid

```
[x,y] = meshgrid(1:nc,1:nr);
```

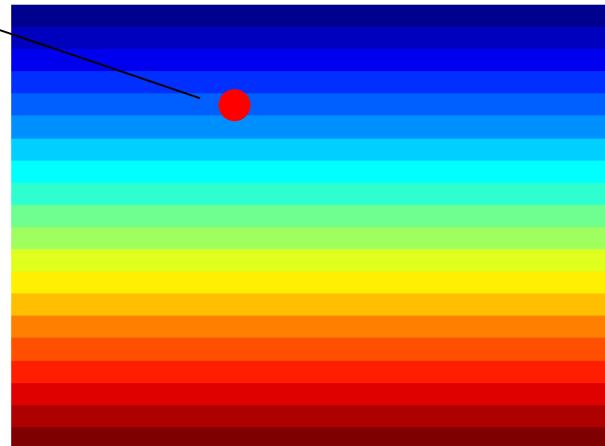
x and y are subscript indice.



Jb_{xy}



x



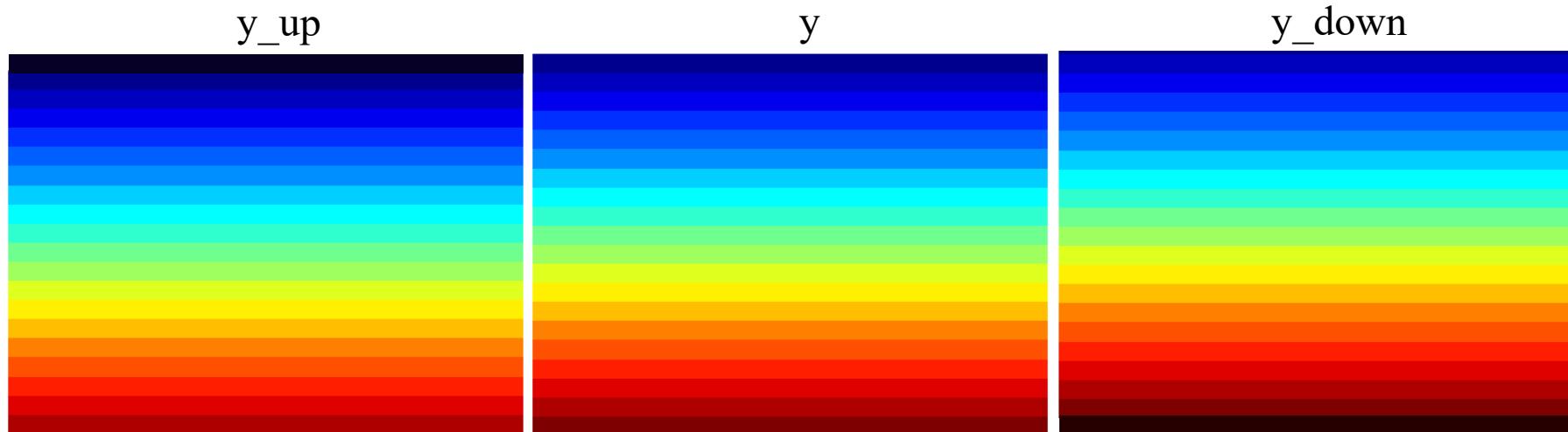
y

Convolution without Looping using meshgrid

```
[x,y] = meshgrid(1:nc,1:nr);
```

```
y_up = y-1;
```

```
y_down = y+1;
```



y_up =

0	0	0	0	0
1	1	1	1	1
2	2	2	2	2

y =

1	1	1	1	1
2	2	2	2	2
3	3	3	3	3

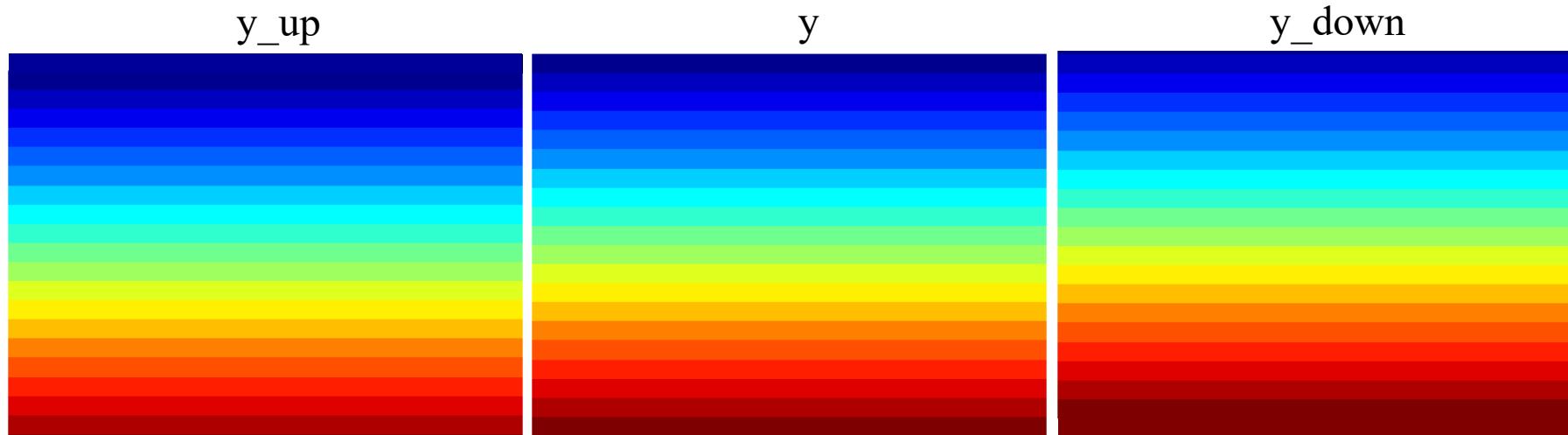
y_down =

2	2	2	2	2
3	3	3	3	3
4	4	4	4	4

Convolution without Looping using meshgrid

```
y_up = y-1;  
y_down = y+1;
```

```
y_up = min(nr,max(1,y_up)); % keep y_up index within legal range of [1,nr]  
y_down = min(nr,max(1,y_down));
```



y_up =

1	1	1	1	1
1	1	1	1	1
2	2	2	2	2

y =

1	1	1	1	1
2	2	2	2	2
3	3	3	3	3

y_down =

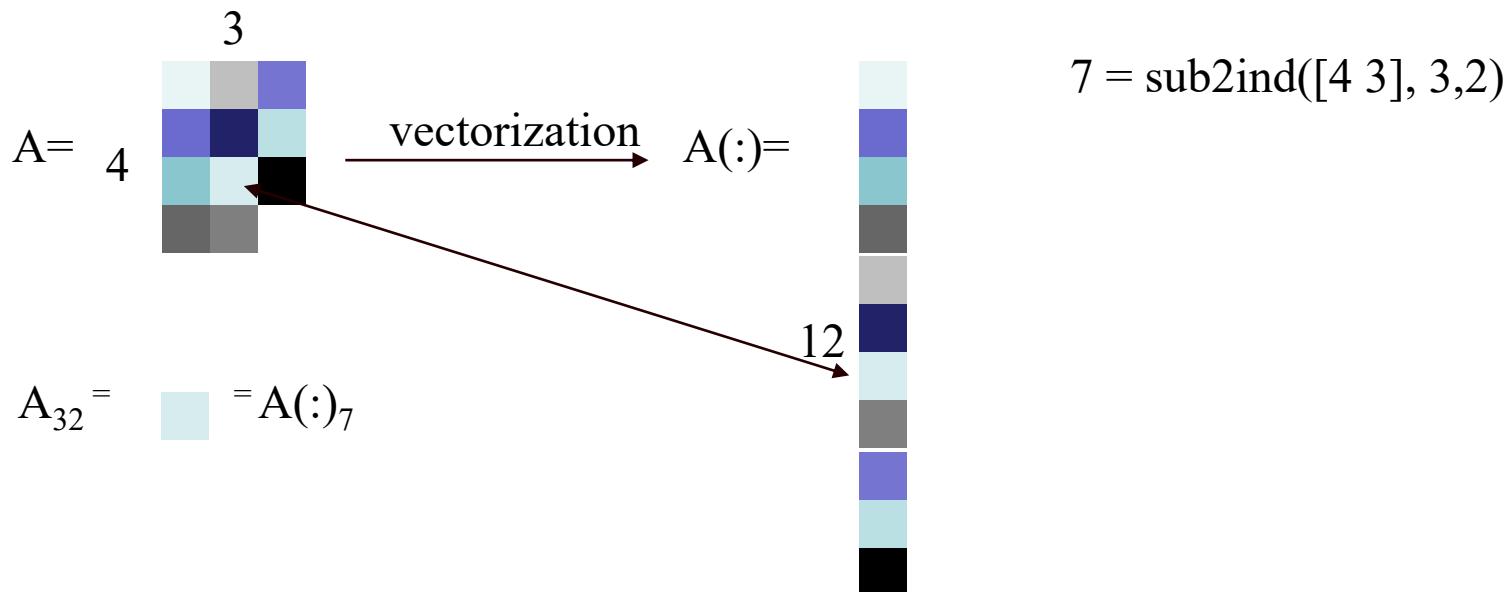
2	2	2	2	2
3	3	3	3	3
3	3	3	3	3

Convolution without Looping using meshgrid

```
y_up = min(nr,max(1,y_up)); % keep y_up index within legal range of [1,nr]  
y_down = min(nr,max(1,y_down));
```

```
ind_up = sub2ind([nr,nc],y_up(:,x(:)); % create linear index  
ind_down = sub2ind([nr,nc],y_down(:,x(:));
```

```
linear_index = sub2ind([n_row, n_col], row_subscript, col_subscript)
```

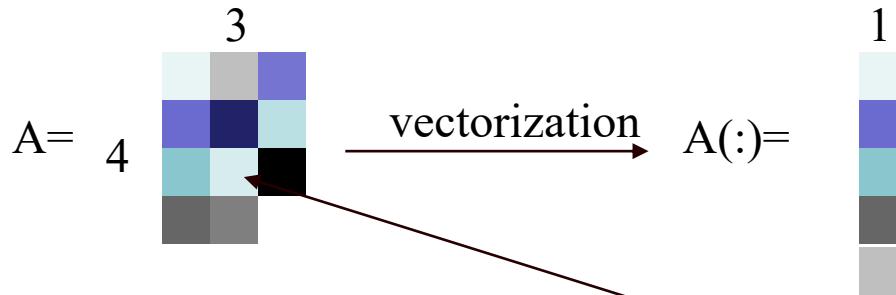


Convolution without Looping using meshgrid

```
y_up = min(nr,max(1,y_up)); % keep y_up index within legal range of [1,nr]  
y_down = min(nr,max(1,y_down));
```

```
ind_up = sub2ind([nr,nc],y_up(:,x(:)); % create linear index  
ind_down = sub2ind([nr,nc],y_down(:,x(:));
```

```
linear_index = sub2ind([n_row, n_col], row_subscript, col_subscript)
```

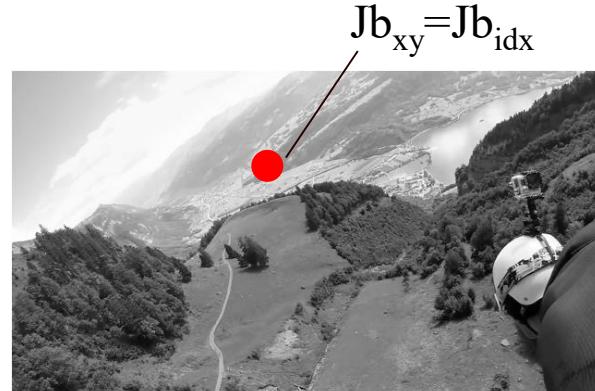


$$A_{32} = \text{[light blue square]} = A(:)_7$$

$7 = \text{sub2ind}([4 3], 3, 2)$

$\text{ind_up} = \underline{\text{sub2ind}([nr,nc], y_up(:,x(:))};$

Operation on vectors



Convolution without Looping using meshgrid

```
J_out = 2*Jb(:) - 0.8*Jb(ind_up) - 0.8*Jb(ind_down);
```

```
J_out = reshape(J_out, nr, nc);
```

```
figure; imagesc(J_out); colormap(gray)
```



=

J_{out}

2



-0.8

J_b



-0.8

$Jb(ind_up)$



$Jb(ind_down)$

With loop



Without loop



Computation time: 0.050154 sec

Computation time: 0.024047 sec