Linear Filtering

Juan Park, Graduate Research Assistant
Chul Min Yeum, Assistant Professor
Civil and Environmental Engineering
University of Waterloo, Canada

CIVE 497 – CIVE 700: Smart Structure Technology

UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING

Last updated: 2020-01-24

Table of contents

Title (slide number)

Bigger Picture: How Can We Count the Blocks?

Given an image, what are the challenges to using computer vision to count blocks?

- Camera Angle/Perspective
- Lighting differences (tree shade, glare, time of day, cloudy)
- Texture variations (corrosion, cracks, material color change)

Example: Automated Brick Counting for Façade Construction Progress Estimation

In order of operation:

(a) Gaussian smoothing (linear filter)

(b) Color thresholding

(c) Erosion (morphological operation)

(d) Dilation (morphological operation)

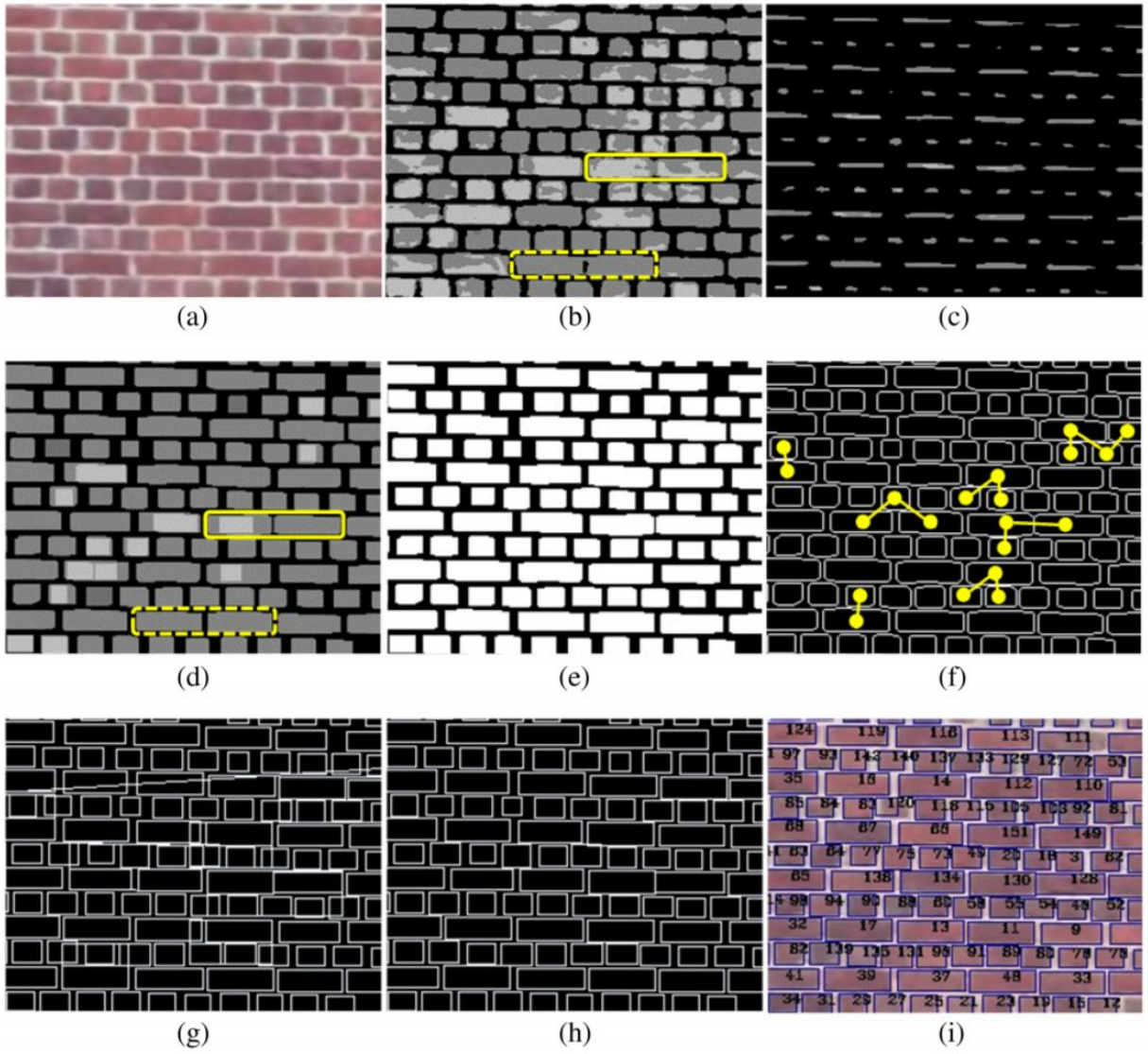
(e) Gray thresholding

(f) Laplace filtering

(g) Minimum area rectangle approximation

(h) Rectangle size filtering

(i) Final result



https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%29CP.1943-5487.0000423

Example: Automated Brick Counting for Façade Construction Progress Estimation

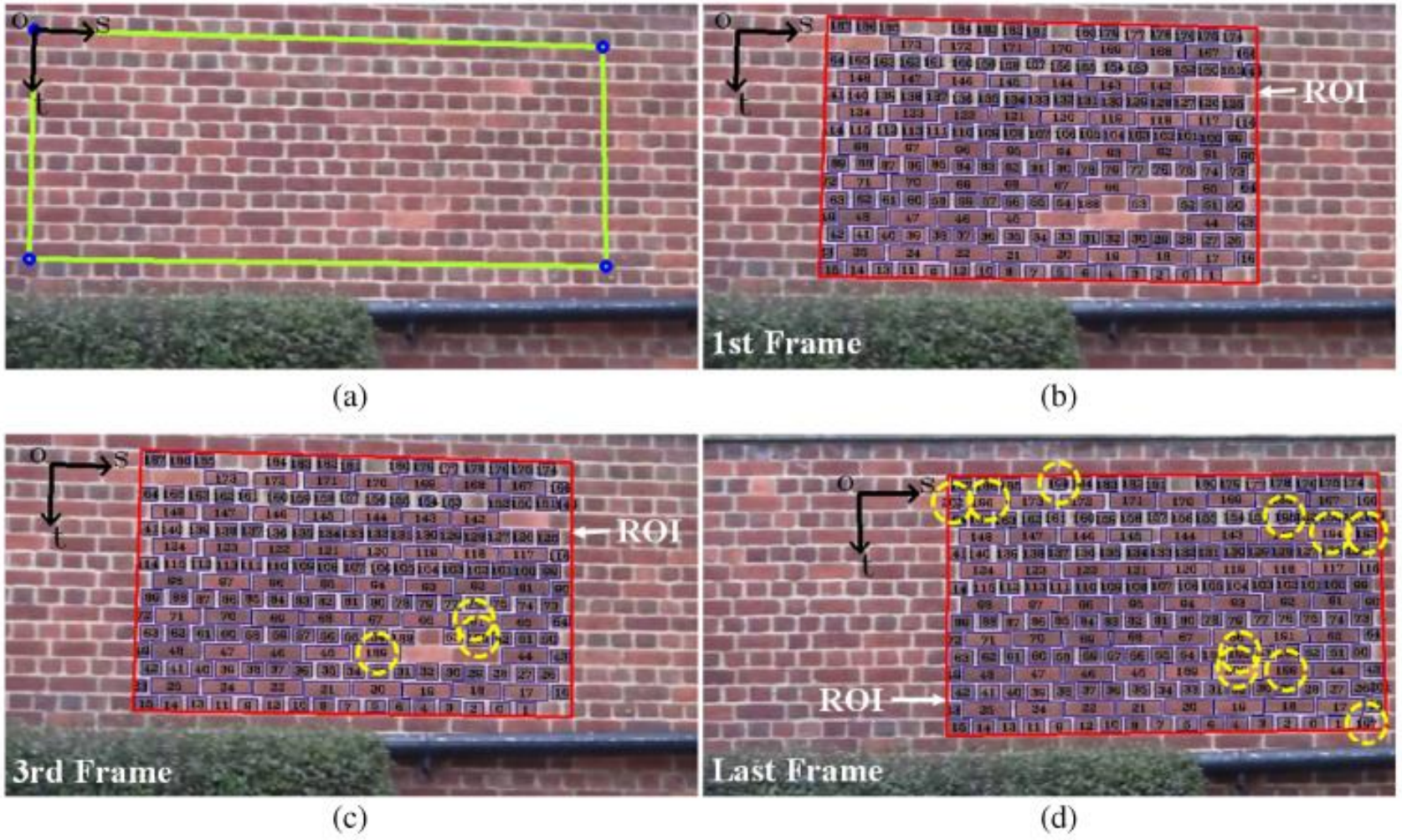
Brick Detecting using Video:

(a) Manually selecting wall coordinate system to compute homography

(b) 1st frame detections

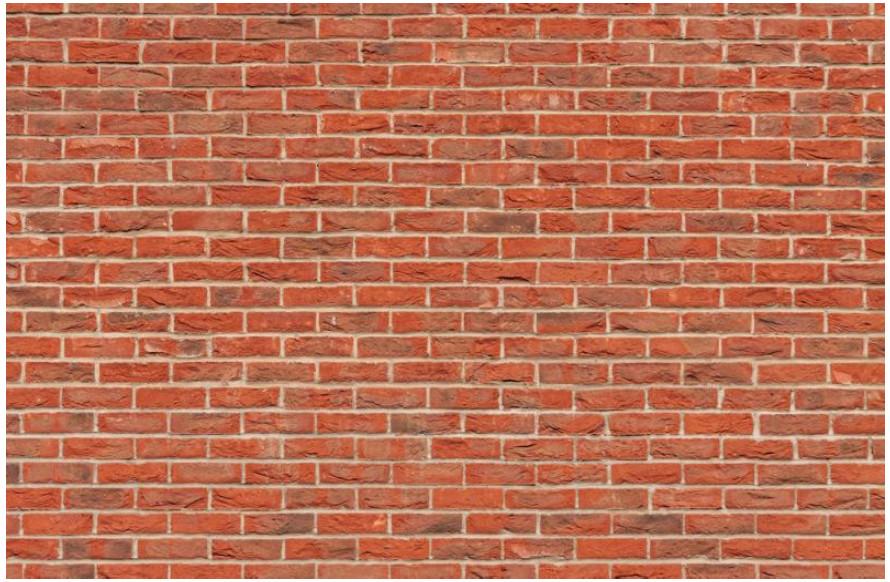
(c) Adding additional detections to first set of detections

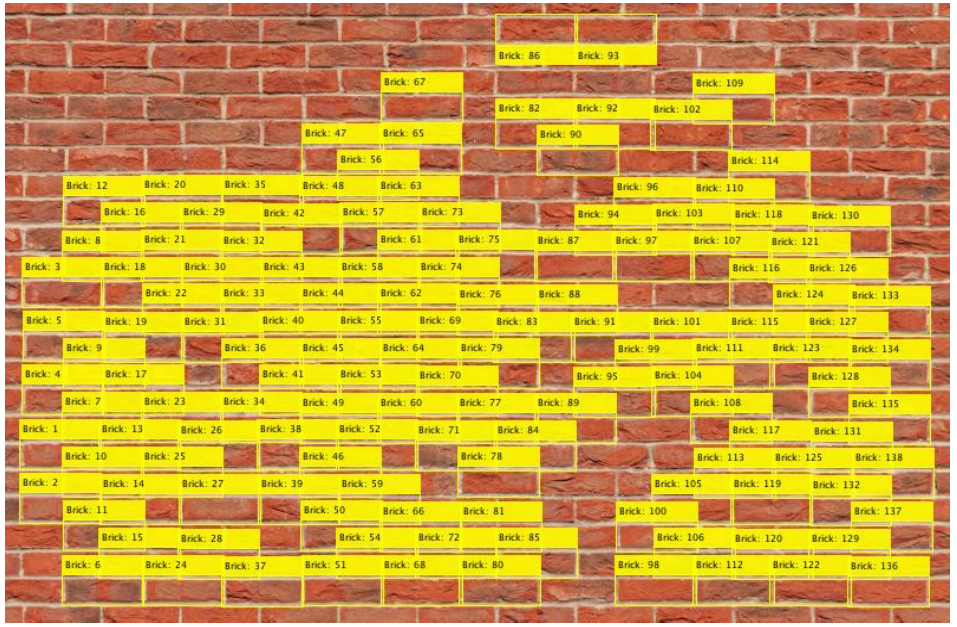
(d) Same as (c).



https://ascelibrary.org/doi/pdf/10.1061/%28ASCE%29CP.1943-5487.0000423

Template Matching





Count_brick_wall.m

Image as Functions

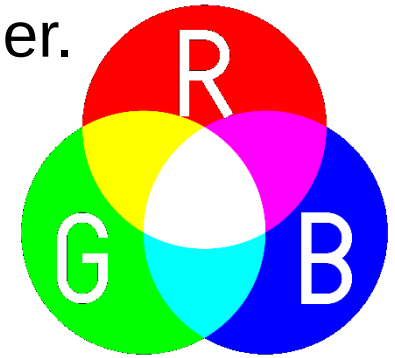
- We can think of an image as a function f, mapping $\mathbb{R}^2$ $(space) \rightarrow \mathbb{R}$ $(intensity)$:
  - $f(x, y)$ gives the intensity at point (x, y)
  - Realistically, images are rectangles, with a finite intensity range
  - (0-1 or 0-255). Thus:

$$f: [0, w] \times [0, h] \rightarrow [0,1],$$

$$w = width, h = height$$

- A color image is just three “grayscale” images pasted together.

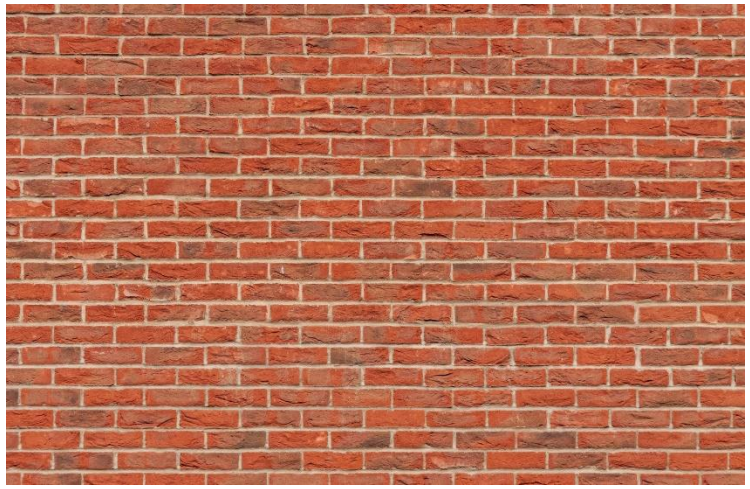
$$f(x, y) = \begin{bmatrix} r(x, y) \\ g(x, y) \\ b(x, y) \end{bmatrix}$$, r=red, g=green, b=blue



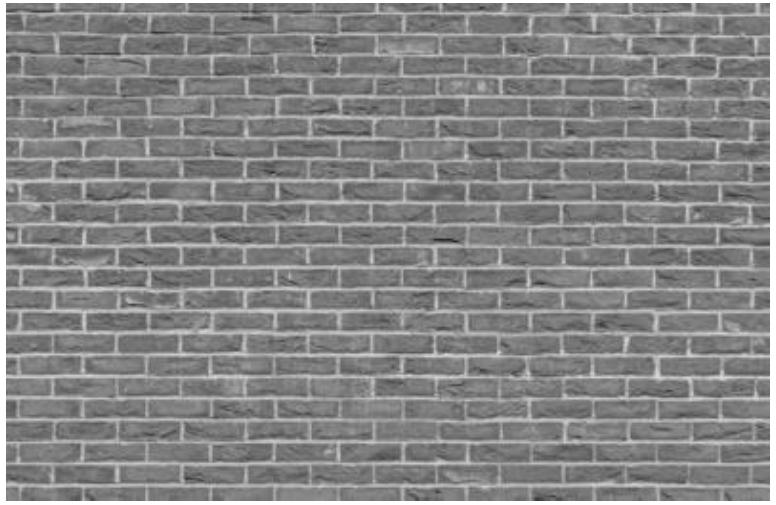
*RGB isn’t the only color space! There are others, like YCbCr, sRGB, CMY…

Image as Functions (Continue)

RGB



Gray



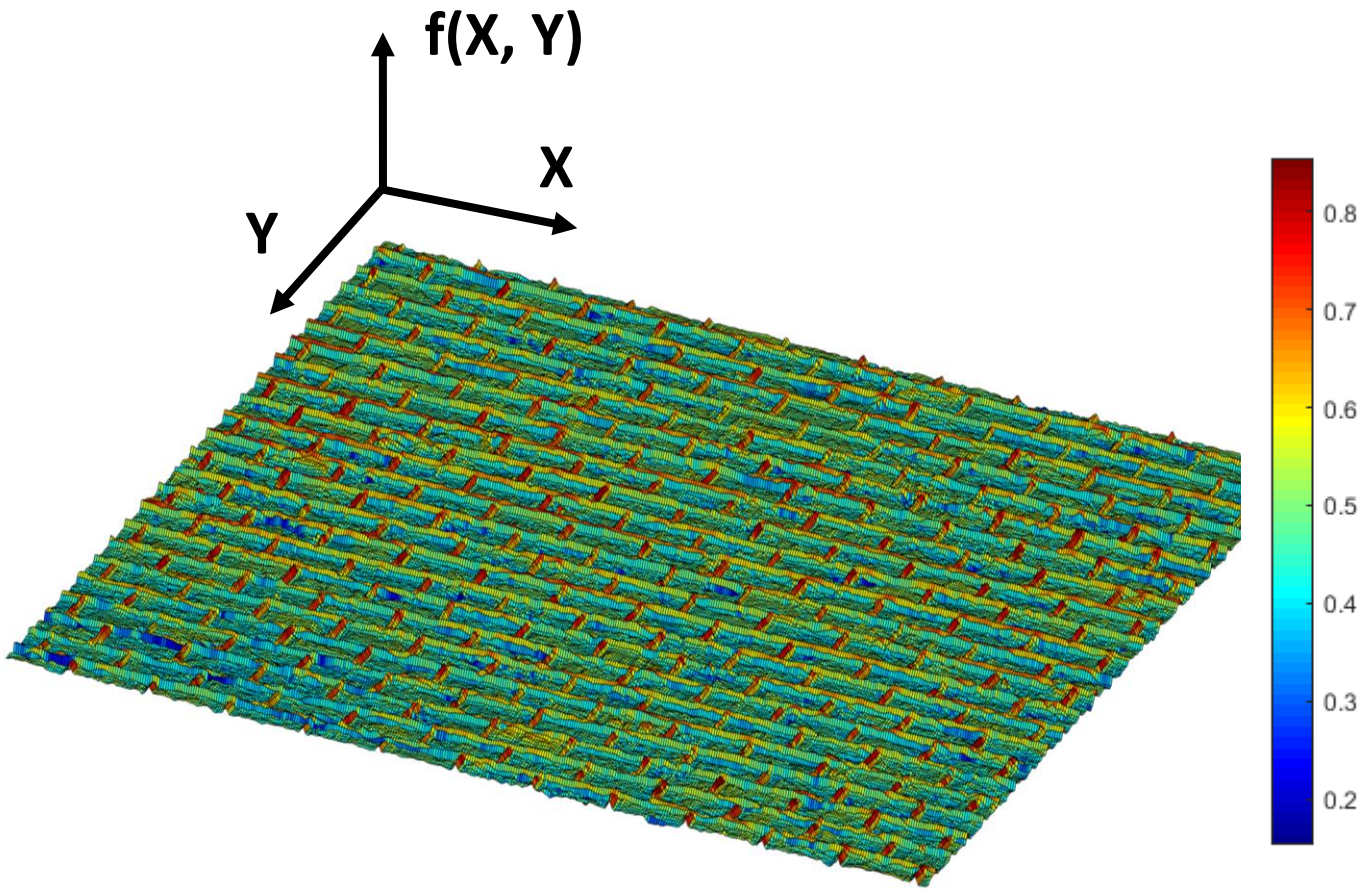
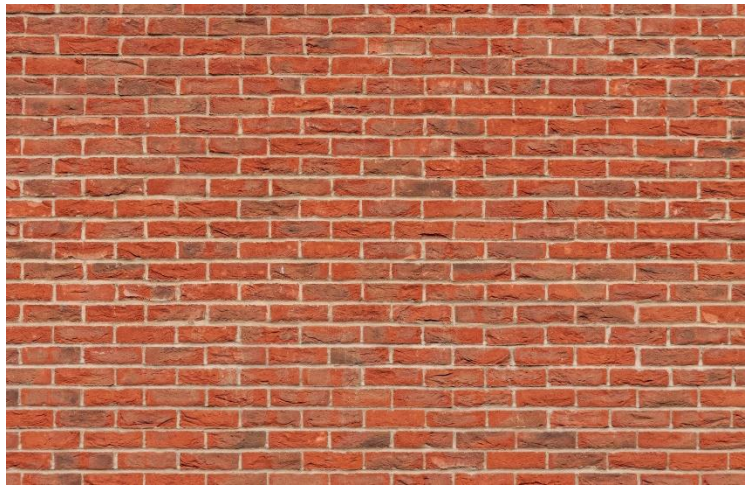
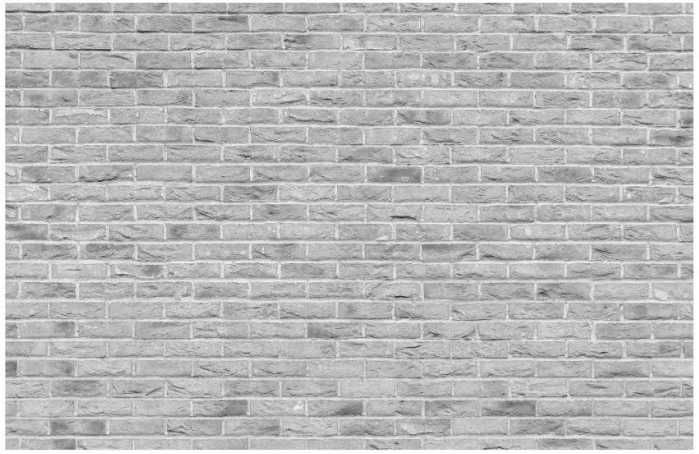


Image as Functions (Continue)

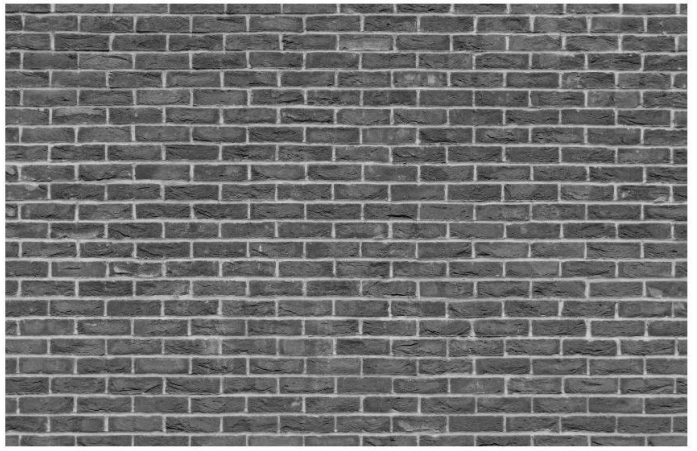
RGB



R



G



B

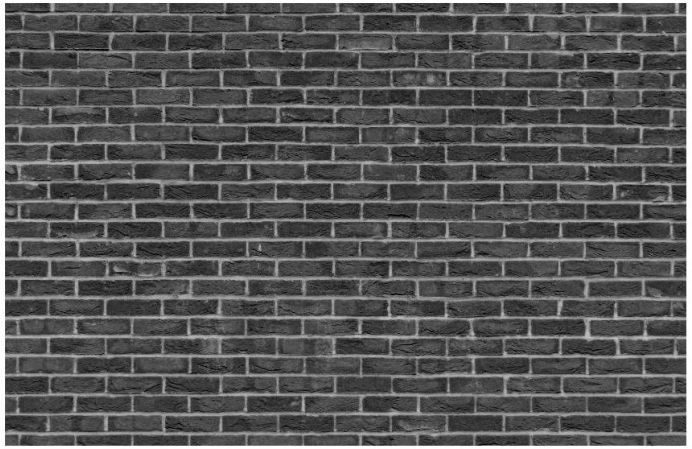
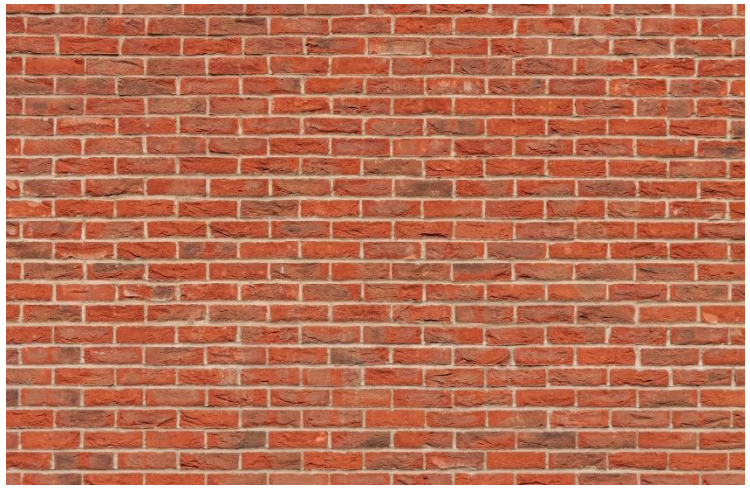
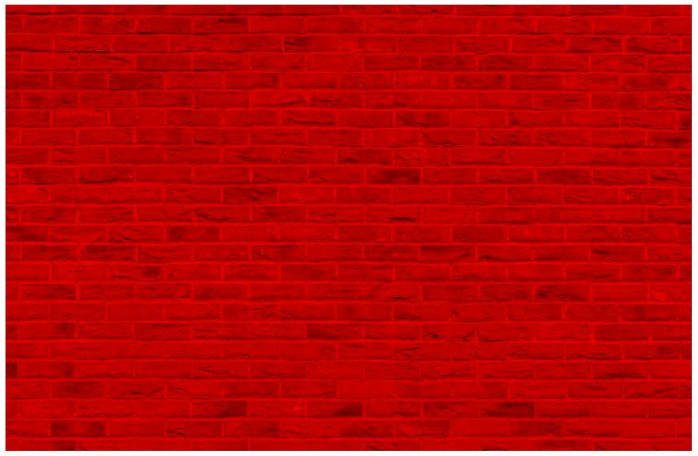


Image as Functions (Continue)

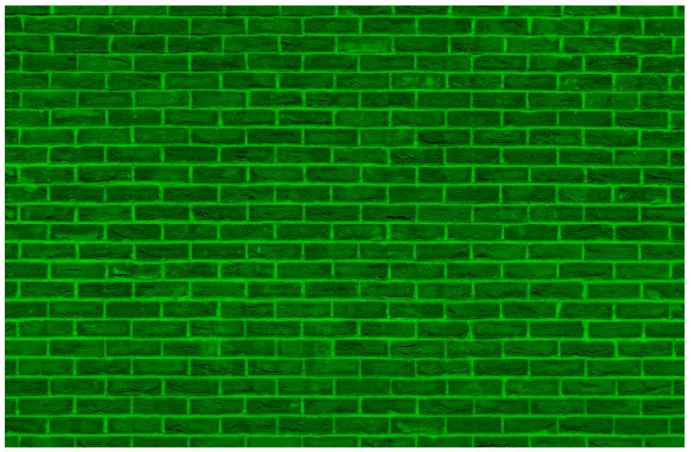
RGB



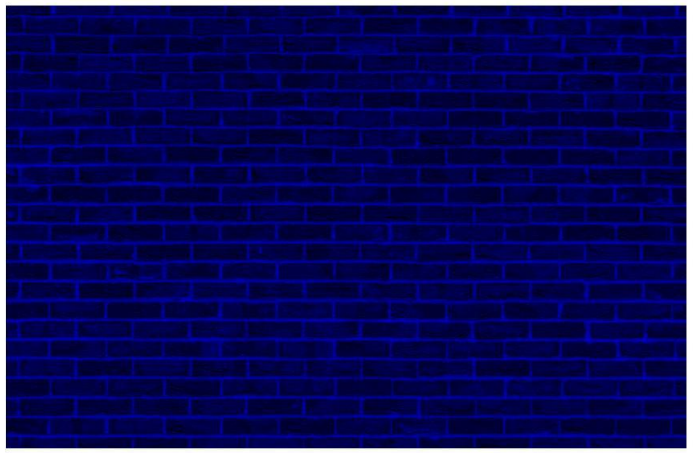
R



G



B



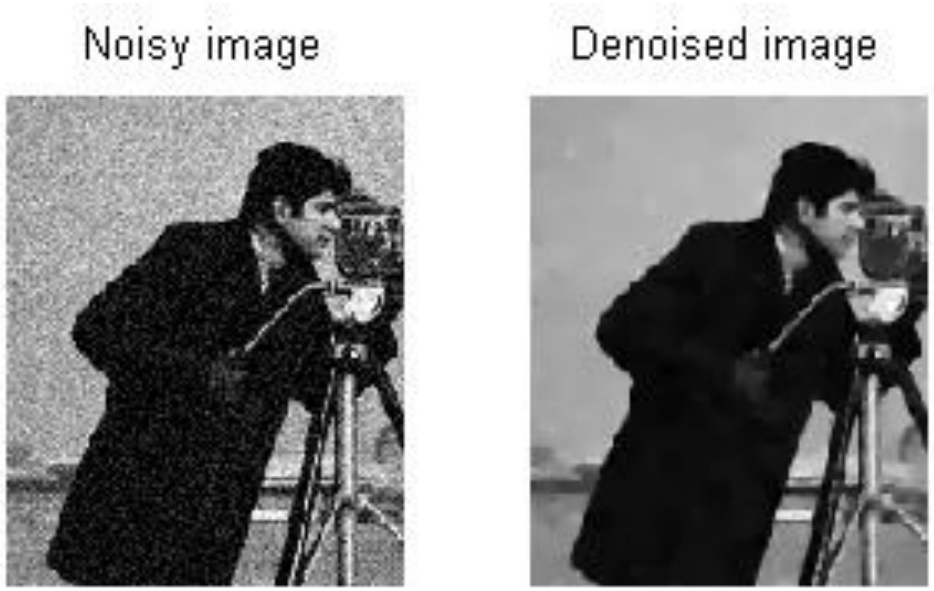
Linear Filtering?

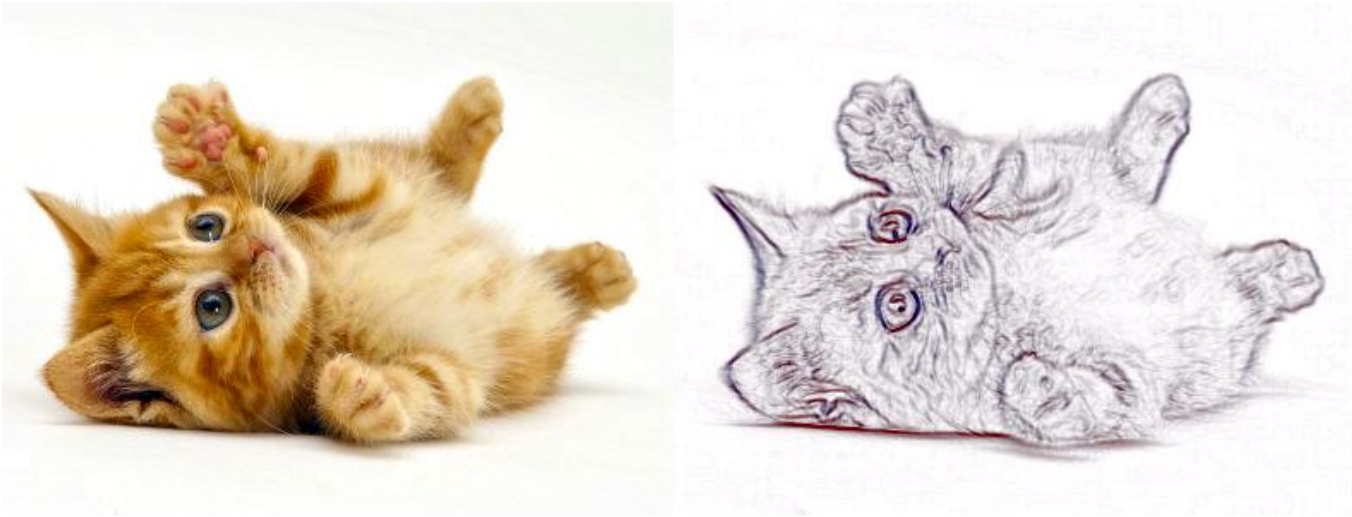
- **Filtering**: Modify or enhance data using a function (i.e. convolution kernel)
- **Linear**: The function follows linear properties of scaling and superposition.
  - Superposition: $\mathrm{h}(A + B) = h(A) + h(B)$
  - Scaling: $\mathrm{h}(\alpha A) = \alpha h(A)$
- Simply, it's linear combination: $\mathrm{h}(X, Y, Z) = aX + bY + cZ$

* h is a linear filter.

Linear Filtering?

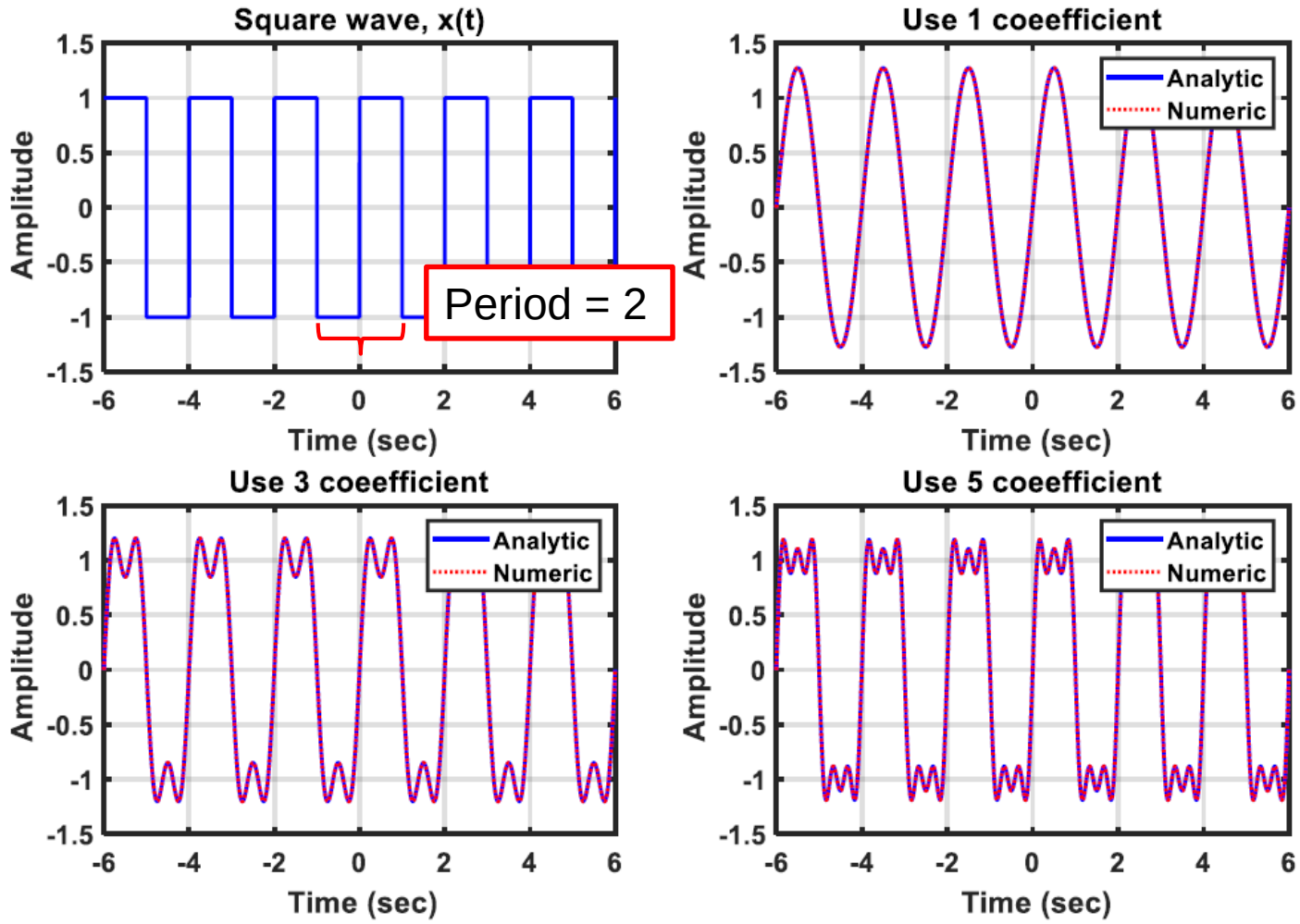
- Linear filtering is used to:
  - Reduce noise in data
  - Extract features from data





Recall: 1D Signal Interpreted using Fourier Series

- Remember the square wave function and its Fourier series approximations?



Recall: Fourier Coefficients

- We can approximate functions by using Fourier series:

$$x(t) = \sum_{n=-\infty}^{\infty} c_n e^{i2\pi nt/T_p}$$

Signal

As n approaches infinity, LHS = RHS

$$c_n = \frac{1}{T_p} \int_{-T_p/2}^{T_p/2} x(t) e^{-i2\pi nt/T_p} dt$$

Key point: we have to know a full period of the function!!!

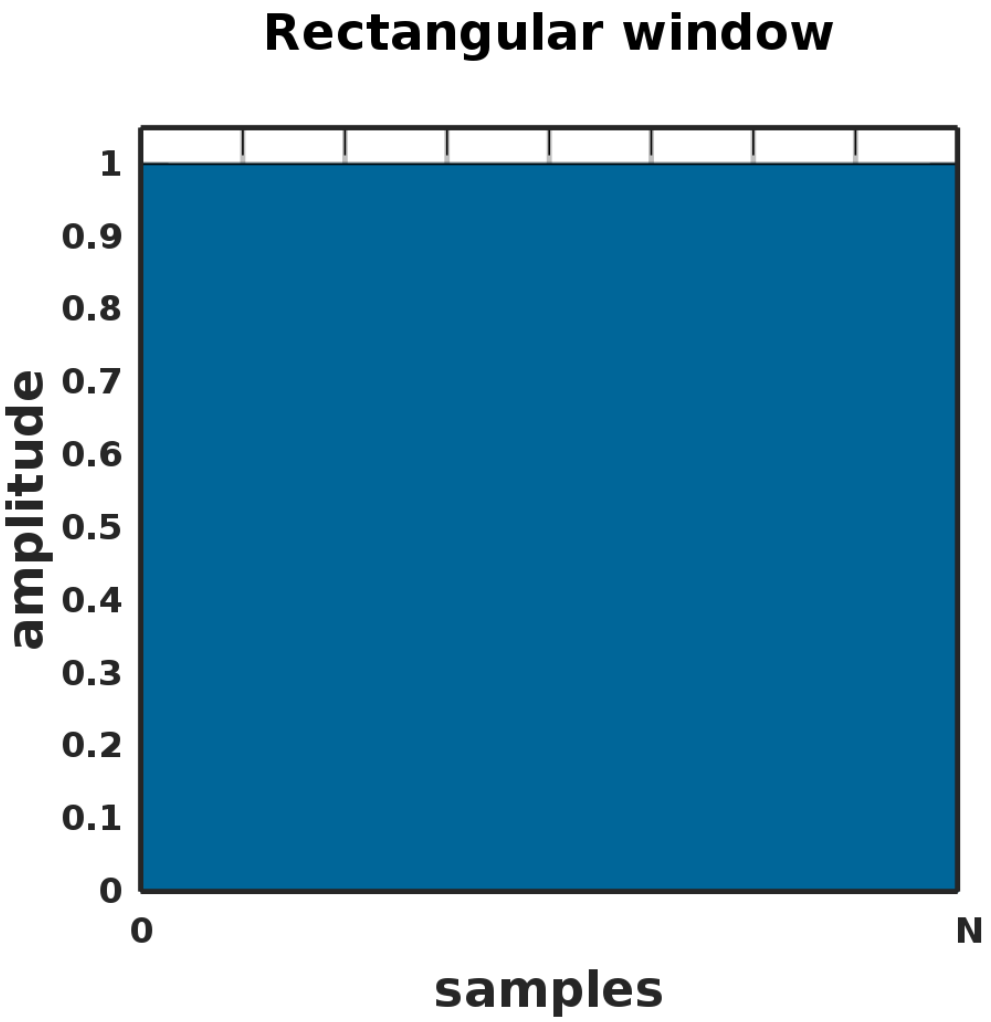
Tp = 2 for square function

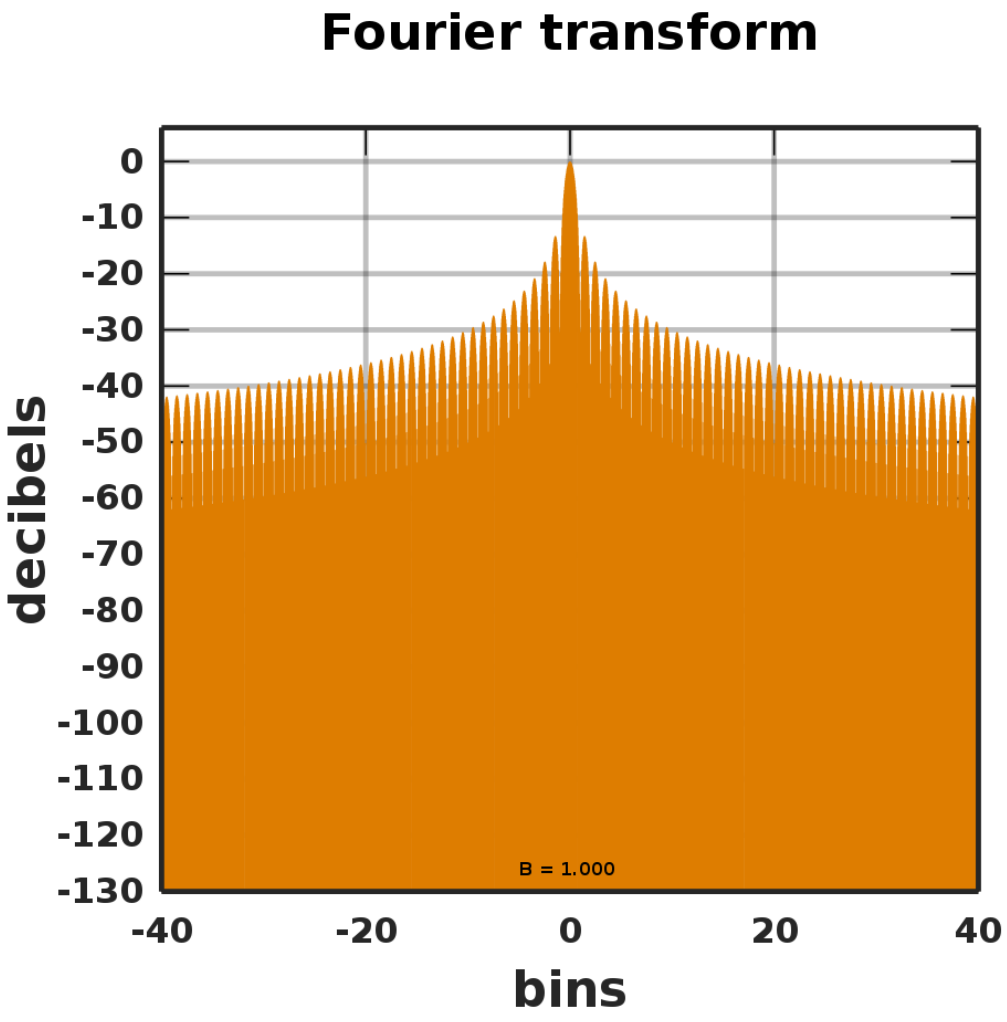
$$w = \frac{2\pi}{T_p} = 2\pi f$$

Recall: Fourier Series -> Fourier Transform

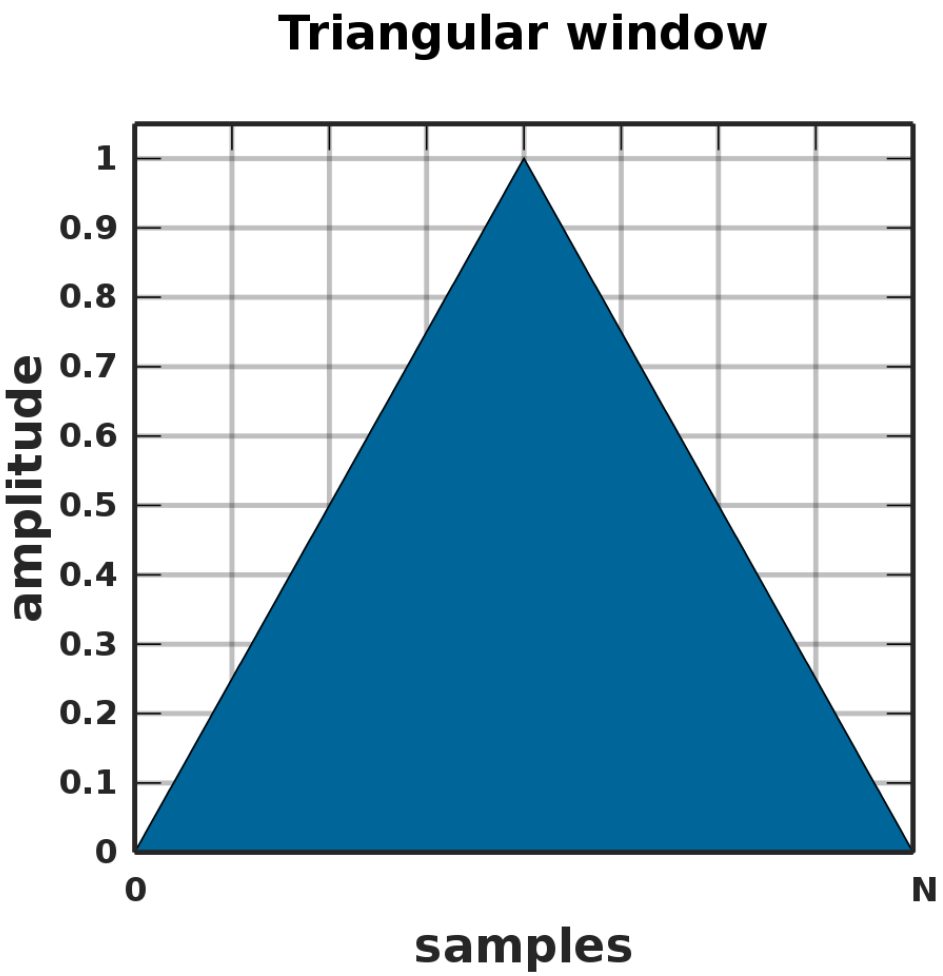
- For a practical, real signal, we do not know the period of the signal.
- Thus, **we assume $T_p = \infty$** (signal_processing2 slide 3).
- Now, when we calculate the amplitude for each value of n, we are essentially doing… a correlation analysis! This is one interpretation of the Fourier transform!
- Let's look at a square wave function repeating infinitely…

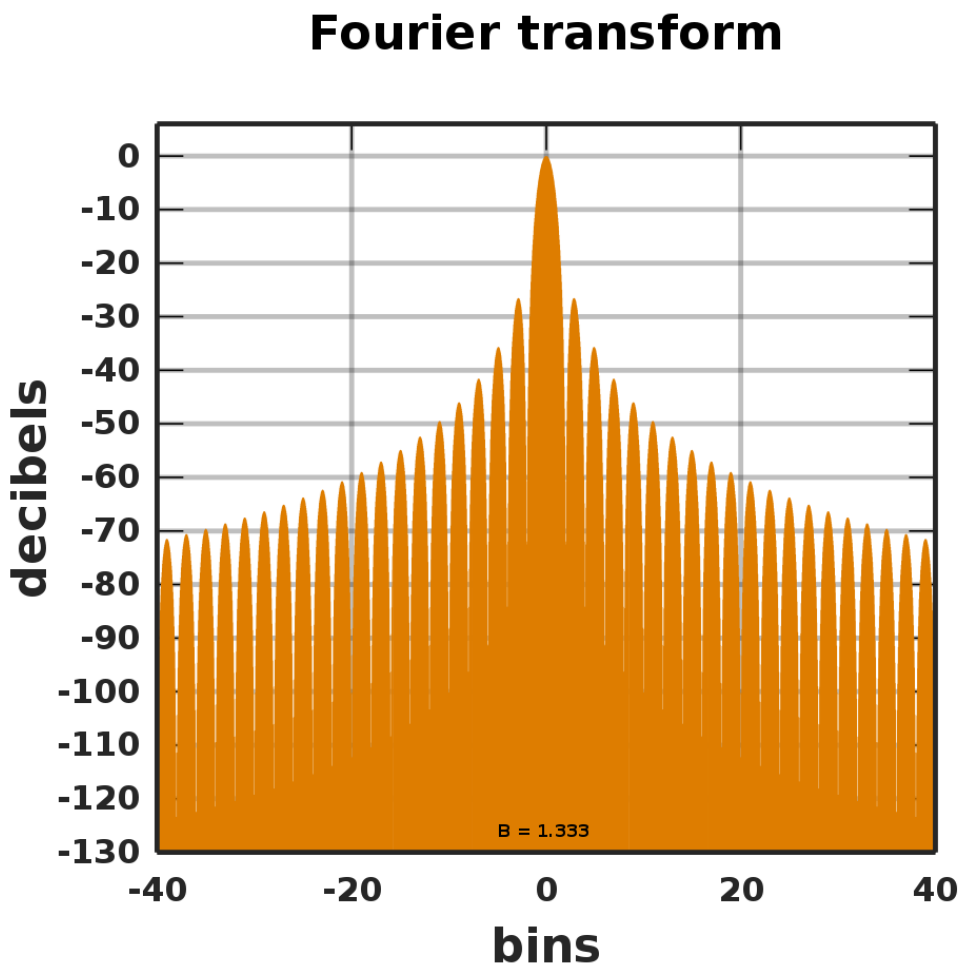
Example: Signal Filtering



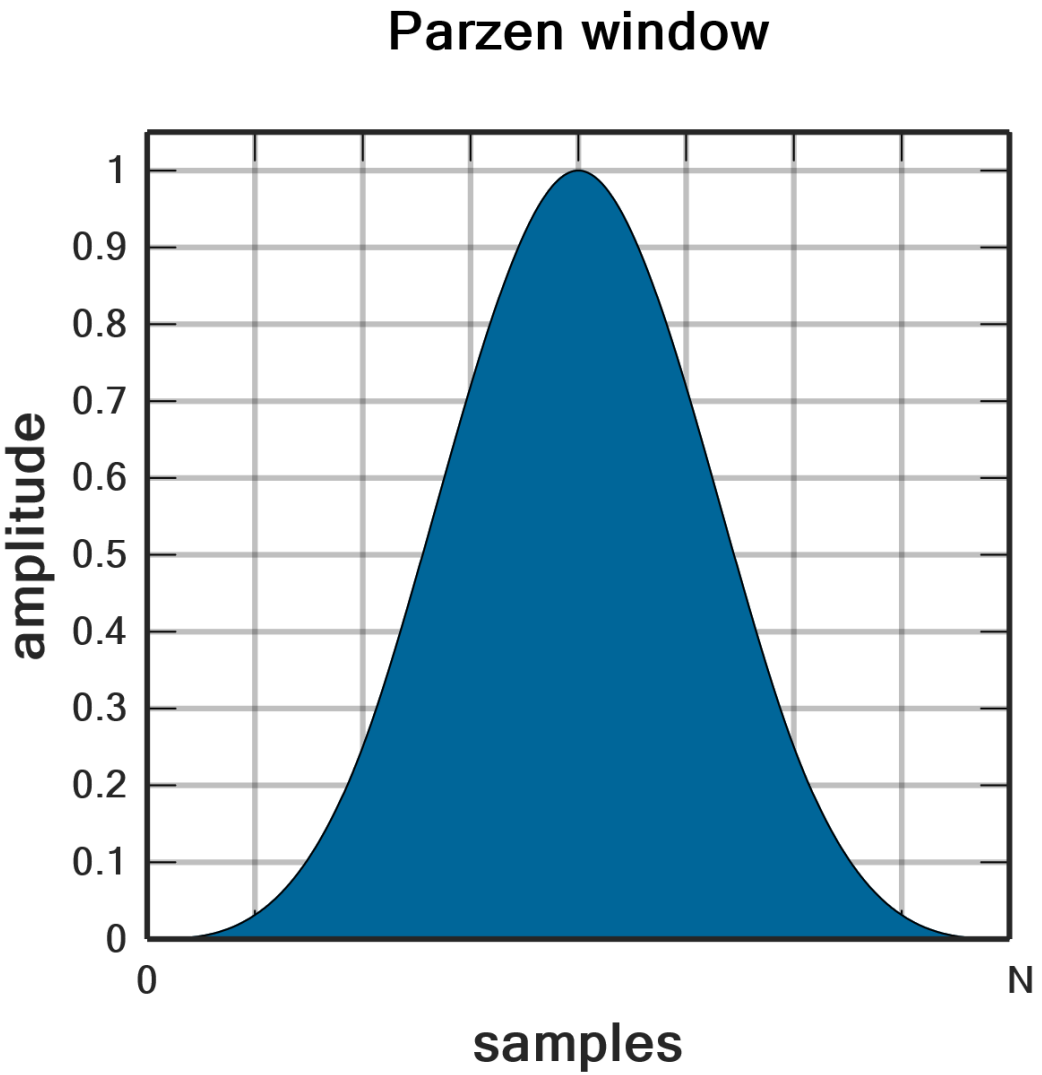


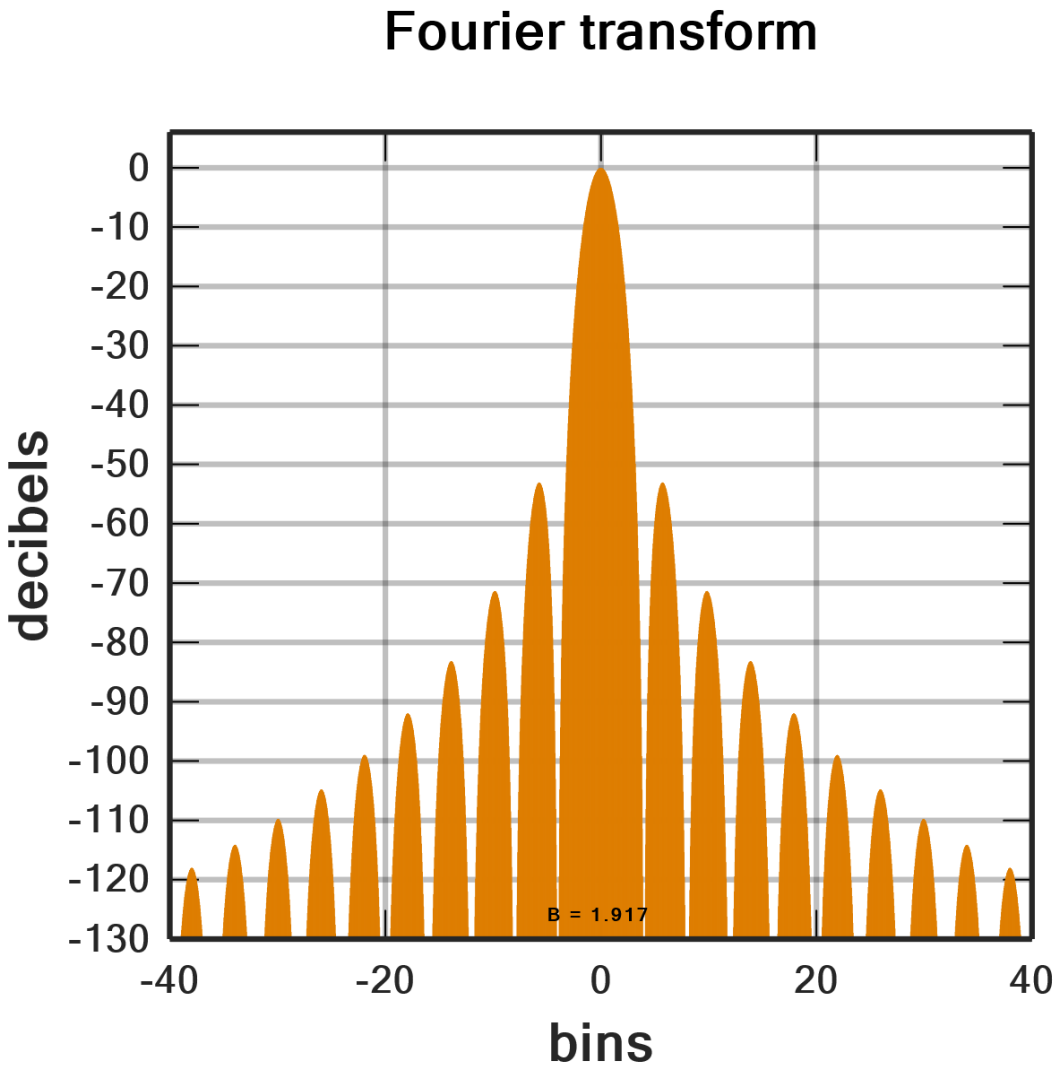
Example: Signal Filtering





Example: Signal Filtering





Example: Signal Filtering

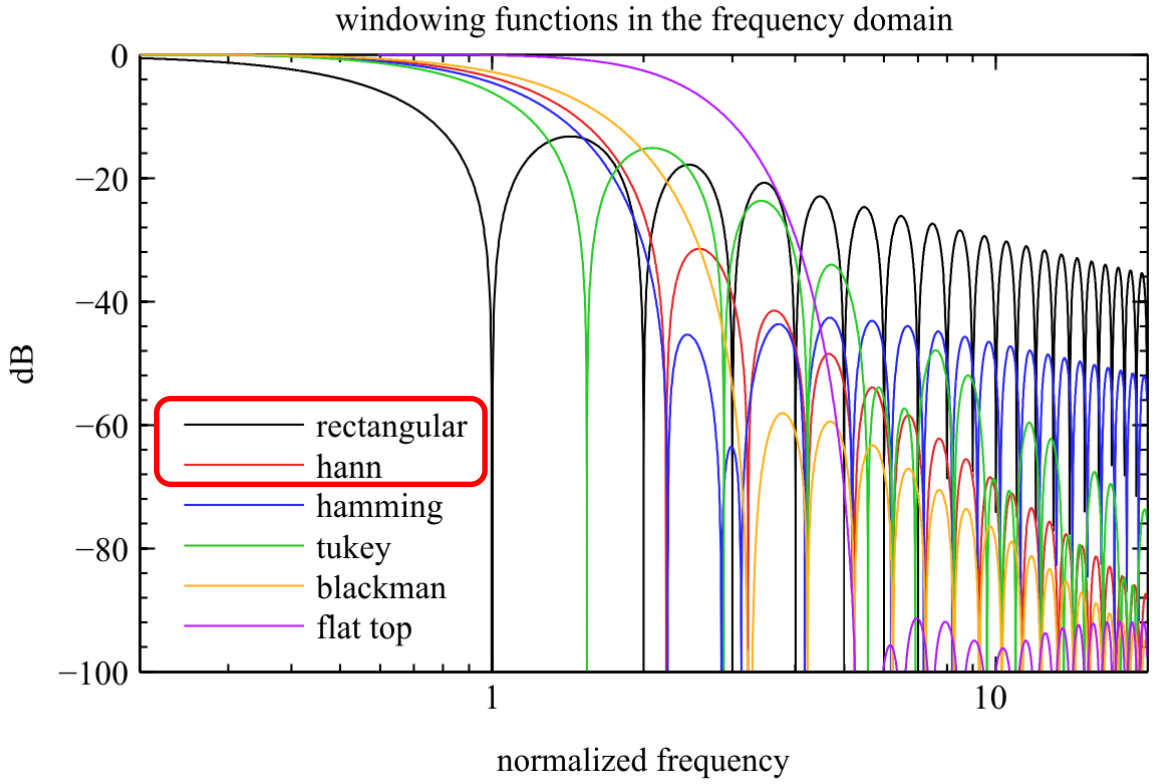
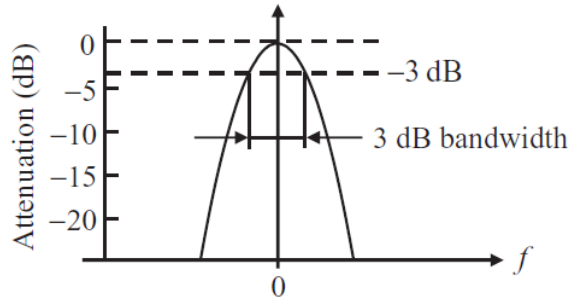


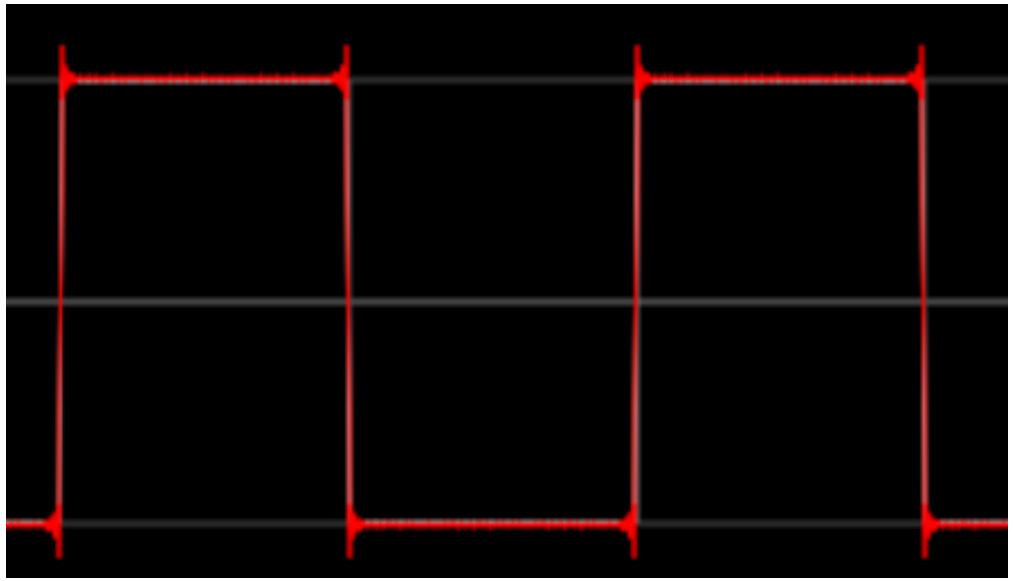
Table 4.2 Properties of some window functions

Window (length T)	Highest side lobe (dB)	Asymptotic roll-off (dB/octave)	3 dB bandwidth	Noise bandwidth	First zero crossing (freq.)
Rectangular	−13.3	6	$0.89\frac{1}{T}$	$1.00\frac{1}{T}$	$\frac{1}{T}$
Bartlett (triangle)	−26.5	12	$1.28\frac{1}{T}$	$1.33\frac{1}{T}$	$\frac{2}{T}$
Hann(ing) (Tukey or cosine squared)	−31.5	18	$1.44\frac{1}{T}$	$1.50\frac{1}{T}$	$\frac{2}{T}$
Hamming	−43	6	$1.30\frac{1}{T}$	$1.36\frac{1}{T}$	$\frac{2}{T}$
Parzen	−53	24	$1.82\frac{1}{T}$	$1.92\frac{1}{T}$	$\frac{4}{T}$



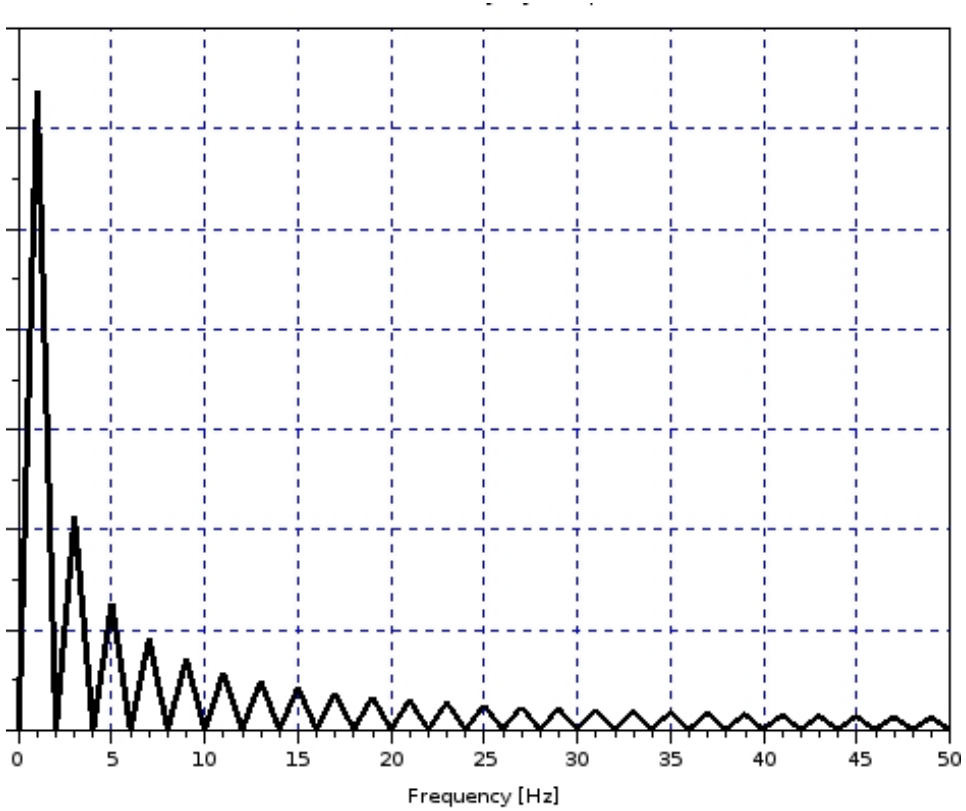
Recall: 1D Signal Interpreted using Fourier Series

- Let's looks at a square wave function repeating infinitely…



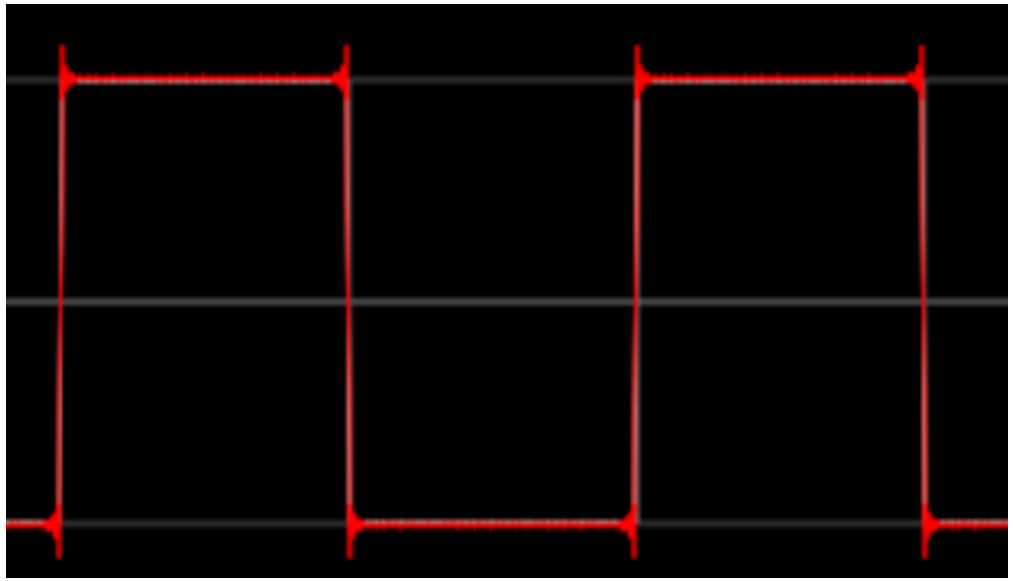
https://www.falstad.com/fourier/

- Now, assuming $T_p = \infty$, and calculate all c_n coefficients and get its magnitude to get the Fourier spectra:

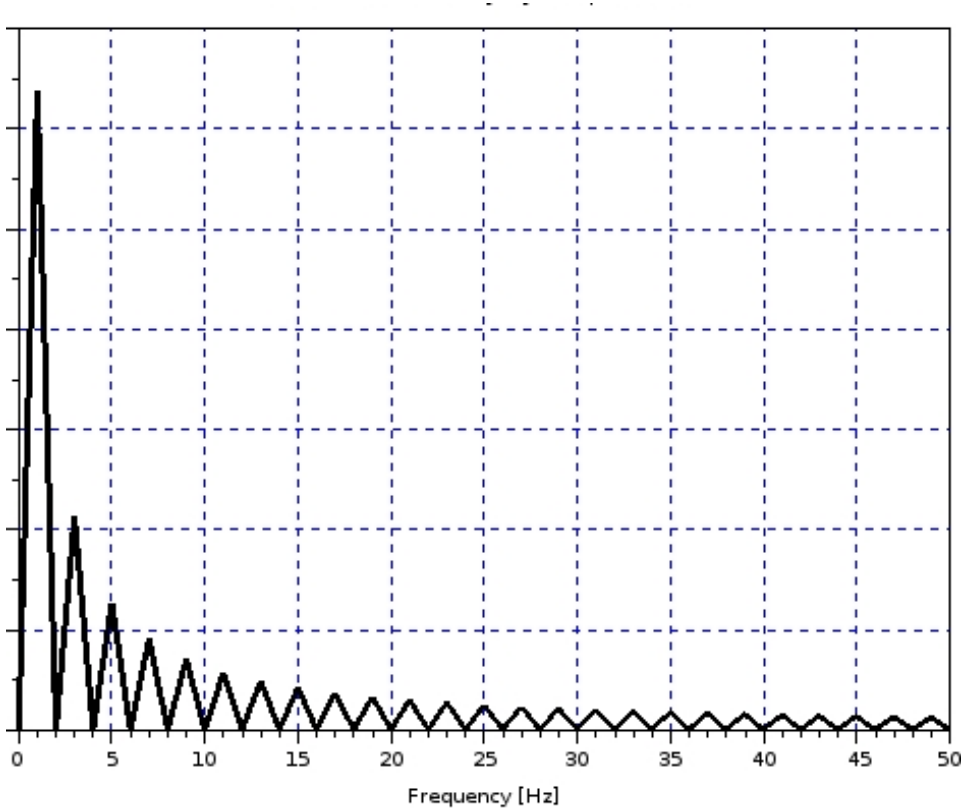


Recall: 1D Signal Interpreted using Fourier Series

- What would happen if we set lower frequencies (c_{1-8}) to 0?

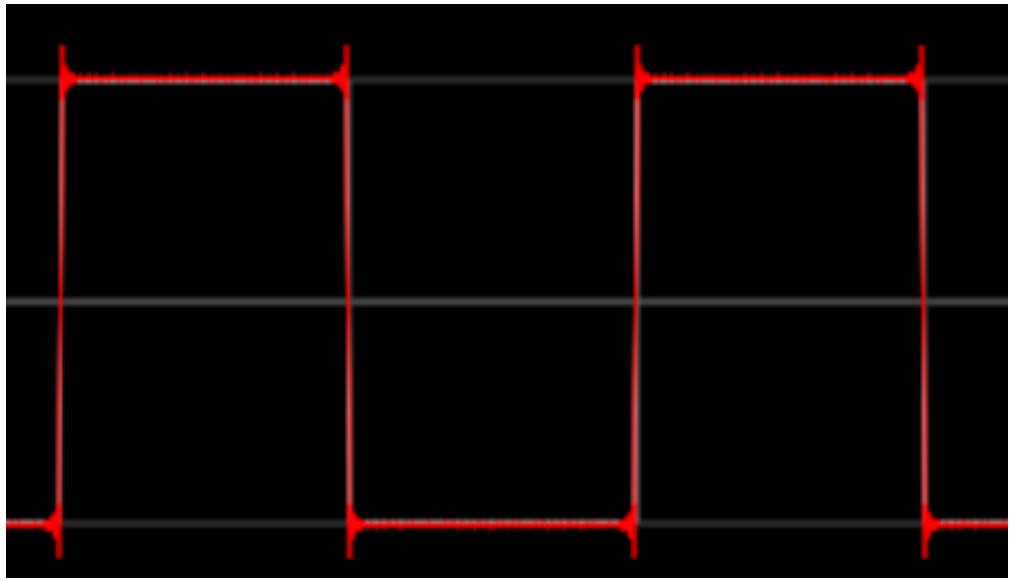


https://www.falstad.com/fourier/

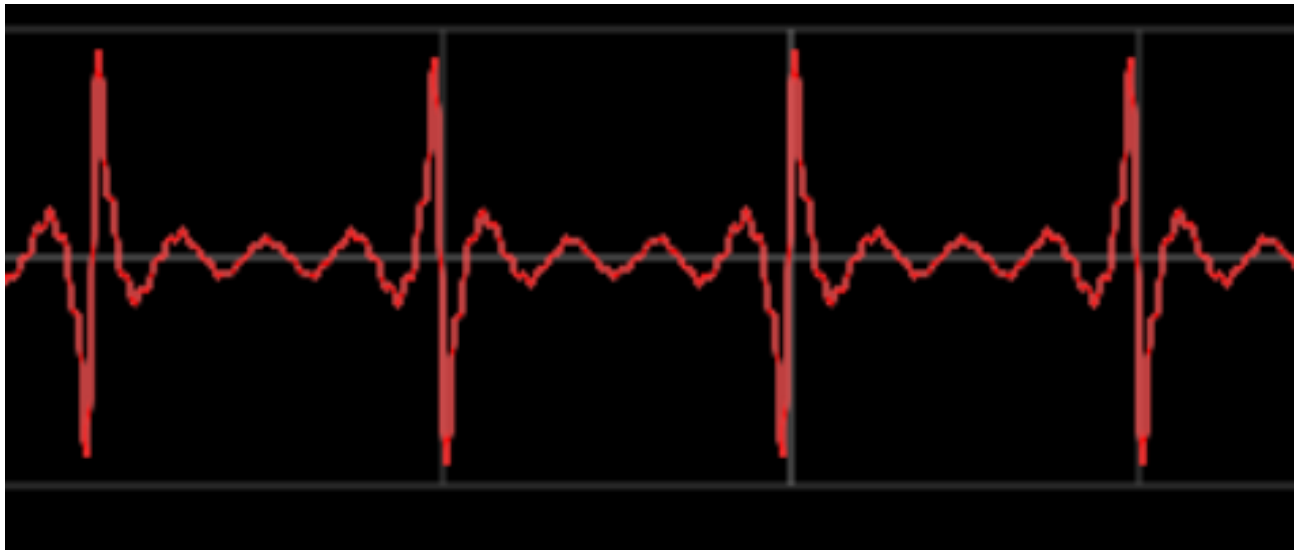


Think: What happens if we set $c_{1-8} = 0$?

- Square function approximation



- If we remove lower frequencies from the square wave approximation:



https://www.falstad.com/fourier/

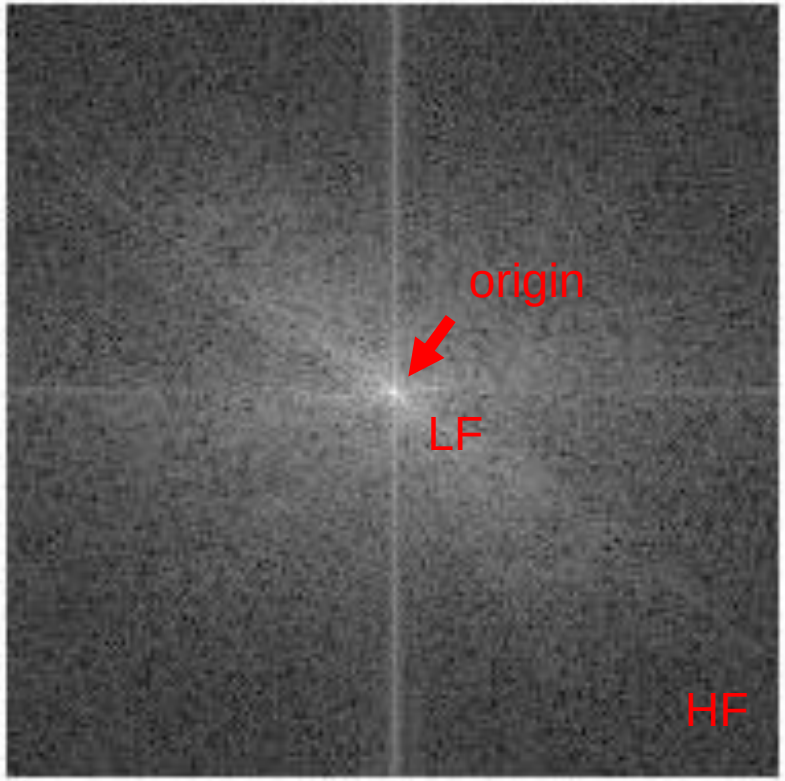
- What does it look like? Kind of like “Edges”, no?
- Let’s keep this in mind as we move to images…

Back to images: Image Interpreted in a Frequency Domain

- Here is an image and its Fourier transform (remember Tp=infinity):

Image

Fourier transform



Low frequency (LF) ≈ solid colors

High frequency (HF) ≈ noise/edges

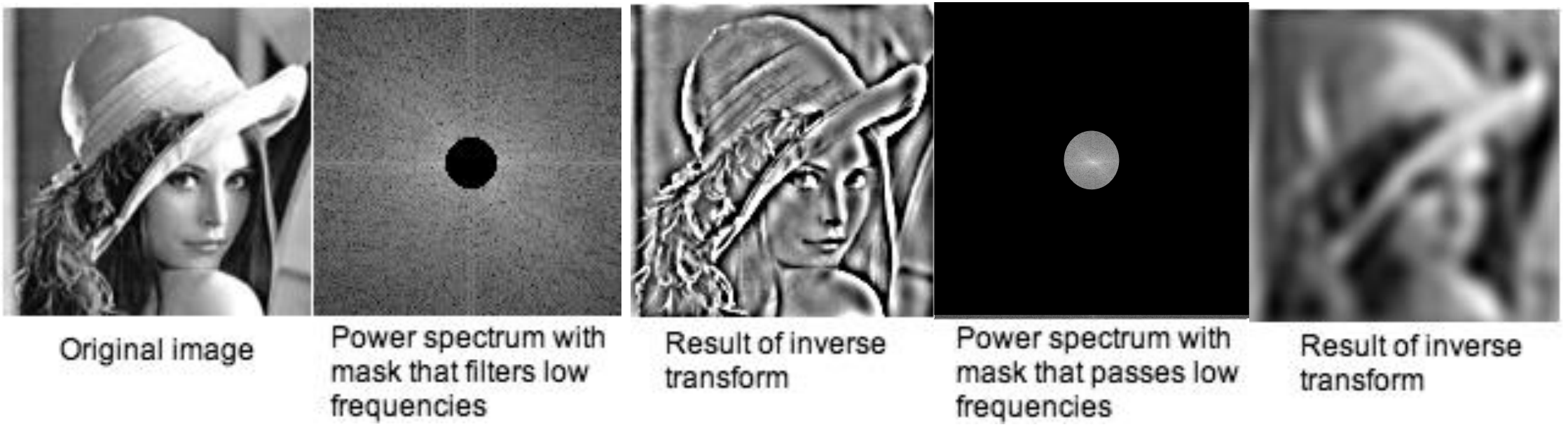
Frequency axis 1

Frequency axis 2

Back to images: Image Interpreted in a Frequency Domain

- If we remove low frequencies of an image, what do we get?
- How about high frequencies?

https://imagej.nih.gov/ij/docs/images/fft.jpg

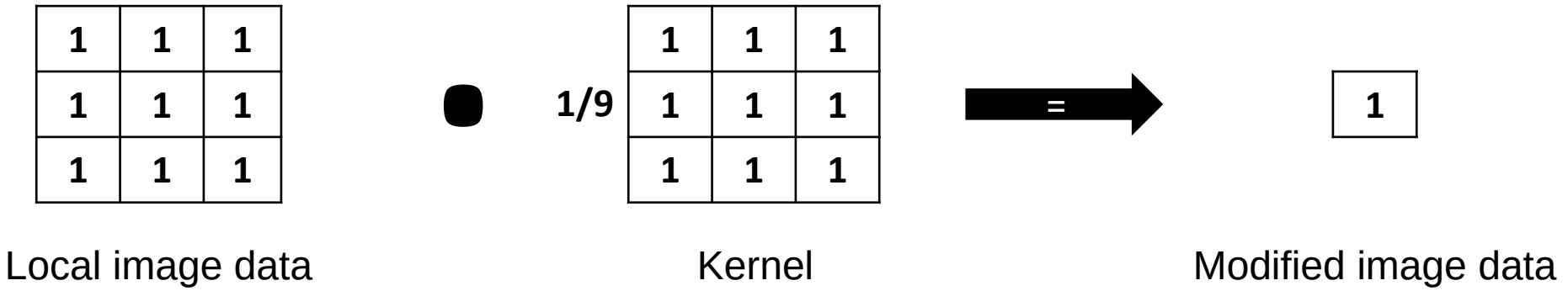


Original image | Power spectrum with mask that filters low frequencies | Result of inverse transform | Power spectrum with mask that passes low frequencies | Result of inverse transform

Key point: Multiplication in the frequency domain is convolution in the spatial/time domain!

Linear Filtering: Simple Example

- The linear function applied to data is called a “kernel/mask/filter”
- “How much is the data correlated to the kernel?”



- Key examples of linear filtering: Cross correlation and Convolution!

Cross-Correlation Definition

Cross-correlation is a measure of similarity between the signal and the kernel.

Let x be the signal, h be the kernel (basically convolution but inverted):

Remember for convolution, the sign is -!

$$h(t) \otimes x(t) = \int_{-\infty}^{\infty} h(\tau) x(t + \tau) d\tau$$

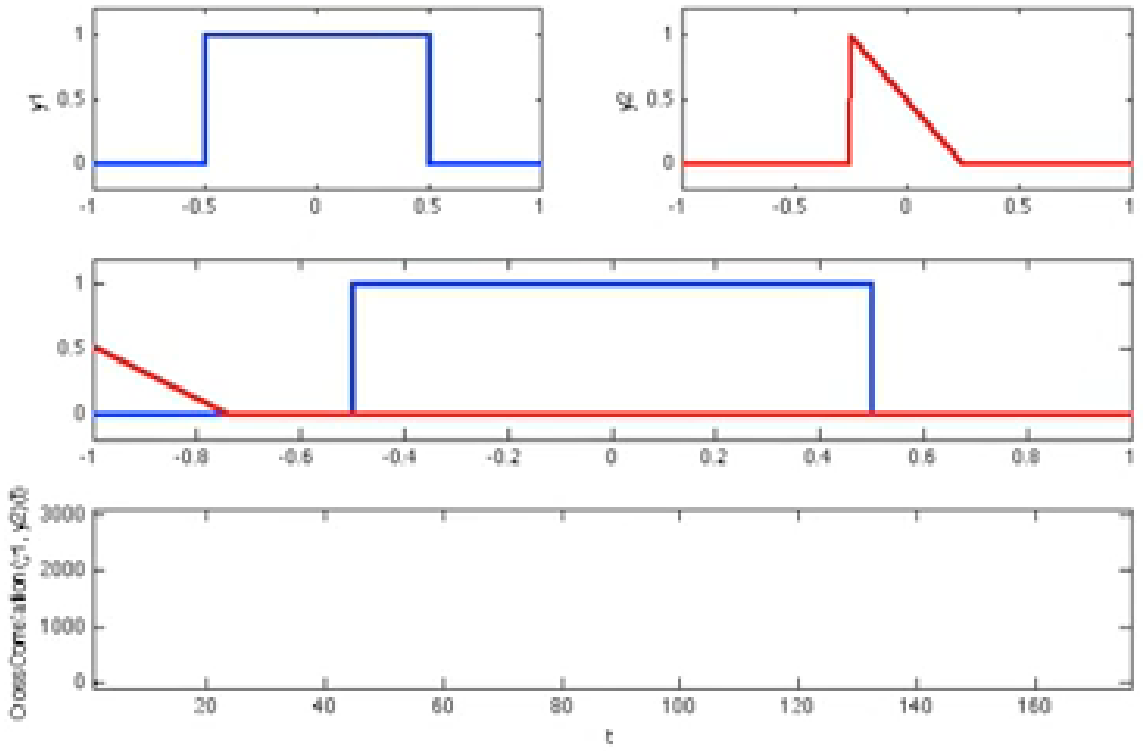
This is called a cross-correlation operation:

$$g = h \otimes f$$

Cross correlation vs convolution

Signal

Kernel



Cross correlation

Convolution

2D Cross-Correlation Using a Kernel

Cross-correlation is a measure of similarity between the signal (image) and the kernel. Let F be the image, H be the kernel (of size $2k+1$ by $2k+1$), G be the output image. **For G at point i, j:**

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

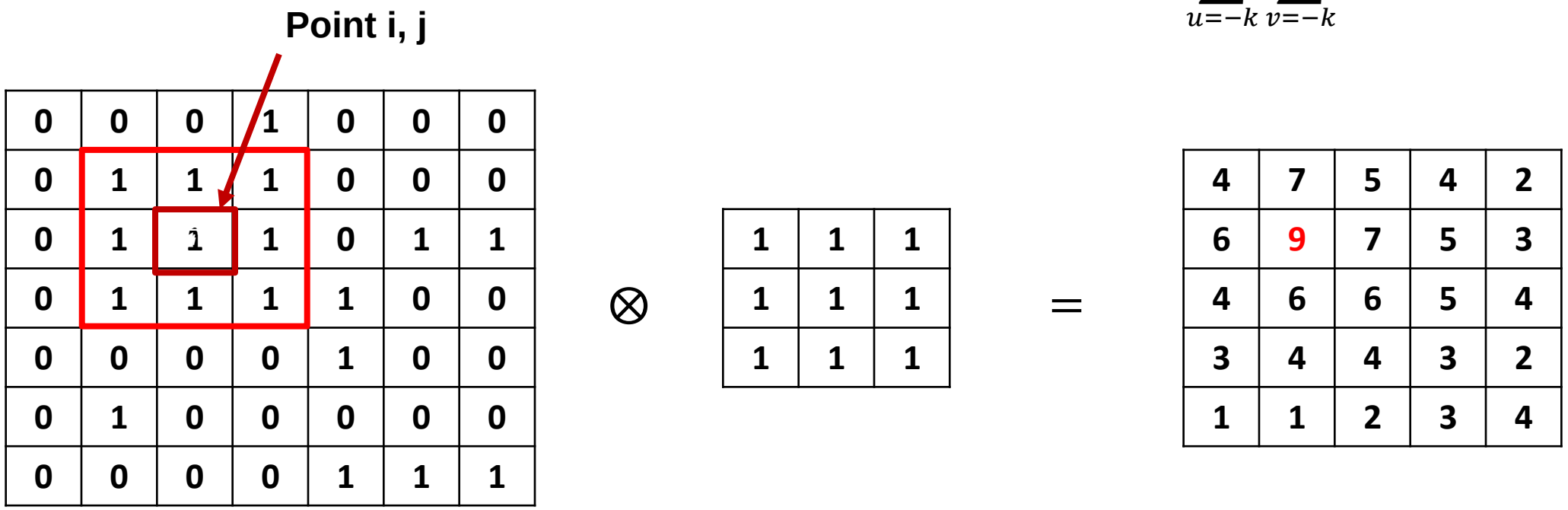
Again, this is called a cross-correlation operation:

$$G = H \otimes F$$

2D Cross-Correlation Example

Cross-correlation is a measure of similarity between two images. Cross correlation with a kernel can be viewed as comparing a litter “picture” or “region” of what you want to find across all sub-regions in the image.

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$



Computation of the Cross-Correlation Between Two Matrices

F

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

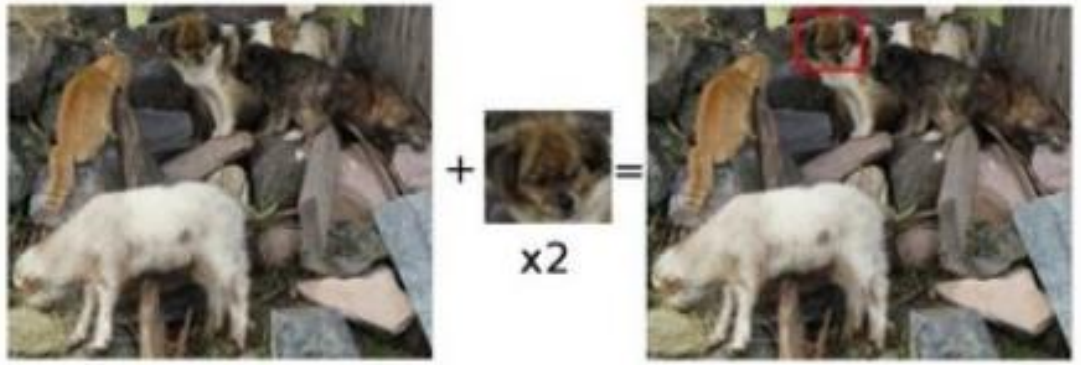
H

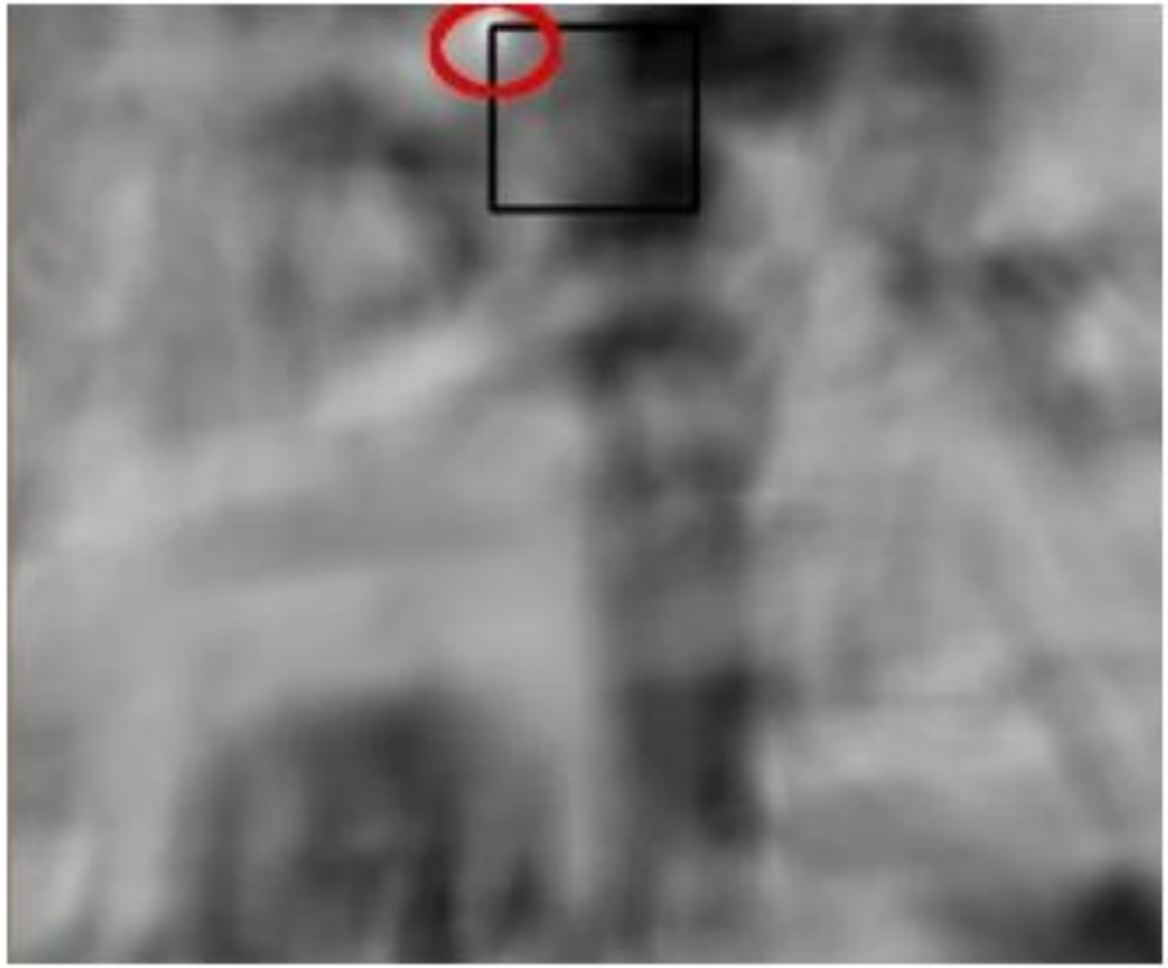
1	1	1
1	1	1
1	1	1

G

4	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0
0	0	0	0	0

Example: Finding Same Objects on Images





https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/template_matching/template_matching.html

Challenges (template matching) !!!!!!

2D Convolution

Let F be the image, H be the kernel (of size 2k+1 x 2k+1),

G be the output image

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

Only difference is that the kernel is "flipped" horizontally and vertically.

This is called a convolution operation:

$$G = H * F$$

Convolution Properties: Commutative, Associative, and Linearity

- Commutative: $F * H = H * F$
- Associative: $F * (H * L) = (H * F) * L$
- Linearity: $F * (H_1 + H_2) = F * H_1 + F * H_2$
- Relationship with differentiation: $(F * H)' = F' * H = F * H'$

Difference between Cross-Correlation and Convolution

X

A	B	C
D	E	F
G	H	I

Y

I	H	G
F	E	D
C	B	A

$$G = H * X = H \otimes Y$$

$\otimes$ cross-correlation

$*$ convolution

The basic difference between convolution and cross-correlation is that the convolution process flips the kernel horizontally and vertically.

Usage

Cross-Correlation: Process to measure a similarity between two signals.

Convolution: Process to transform a signal to another signal.

Guessing Game! Linear Filter: Generating an Identical Image

0	0	0
0	1	0
0	0	0

0	0	0
0	0	1
0	0	0

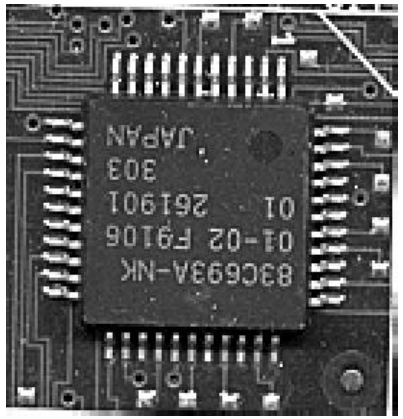
$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

-0.11	-0.11	-0.11
-0.11	1.89	-0.11
-0.11	-0.11	-0.11

0.08	0.12	0.08
0.12	0.20	0.12
0.08	0.12	0.08

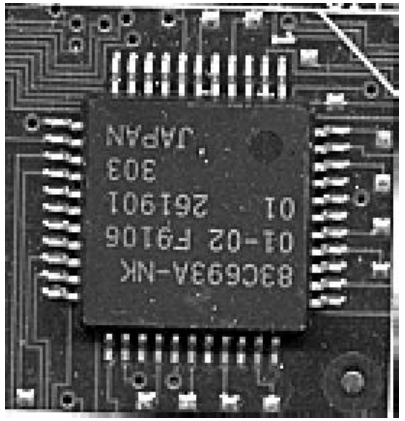
Guessing Game! Linear Filter: Generating an Identical Image

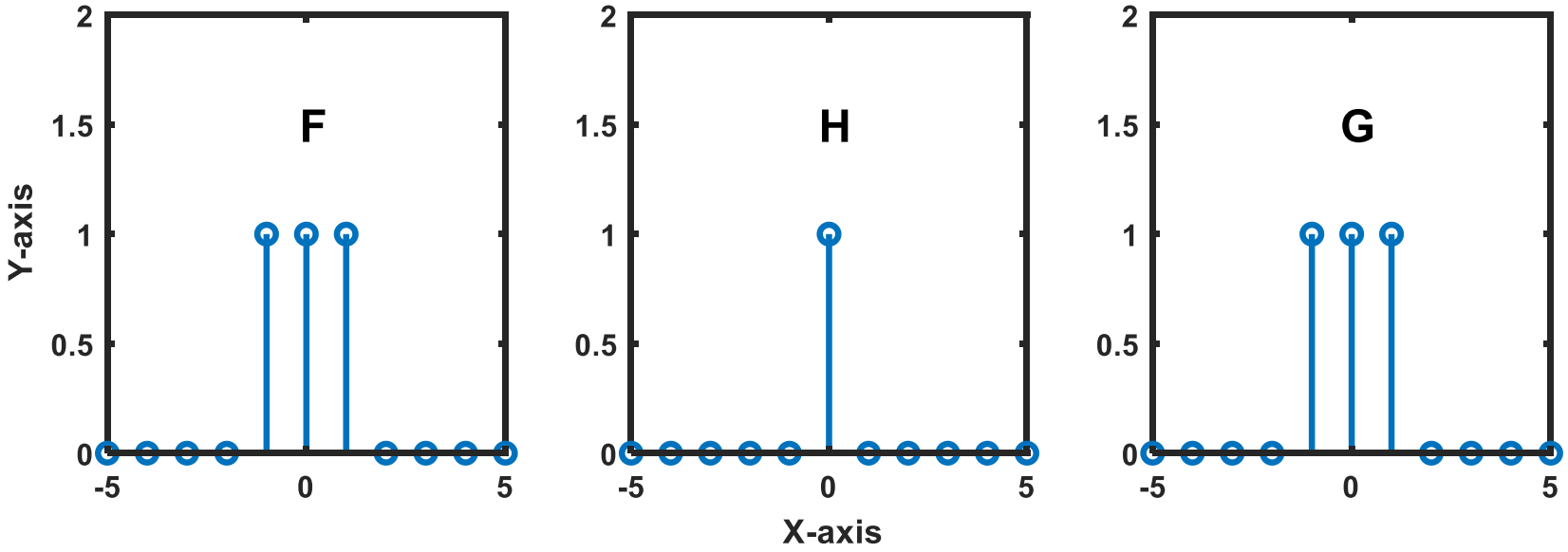


$*$

0	0	0
0	1	0
0	0	0

$=$





$$F * H = G$$

Guessing Game! Linear Filter: Shifting Pixels

0	0	0
0	1	0
0	0	0

0	0	0
0	0	1
0	0	0

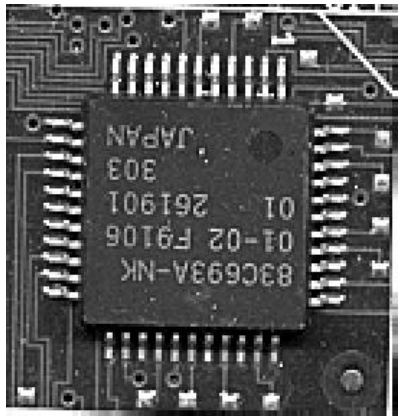
$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

-0.11	-0.11	-0.11
-0.11	1.89	-0.11
-0.11	-0.11	-0.11

0.08	0.12	0.08
0.12	0.20	0.12
0.08	0.12	0.08

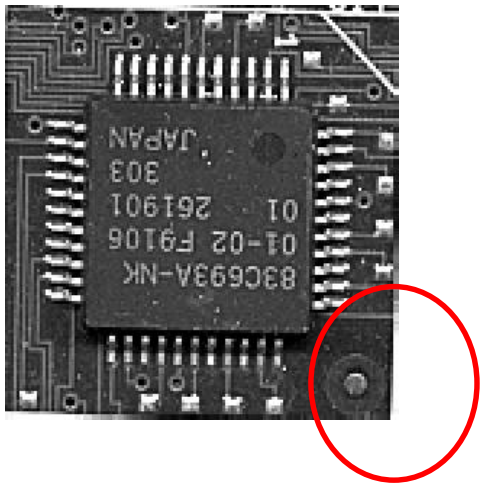
Guessing Game! Linear Filter: Shifting Pixels

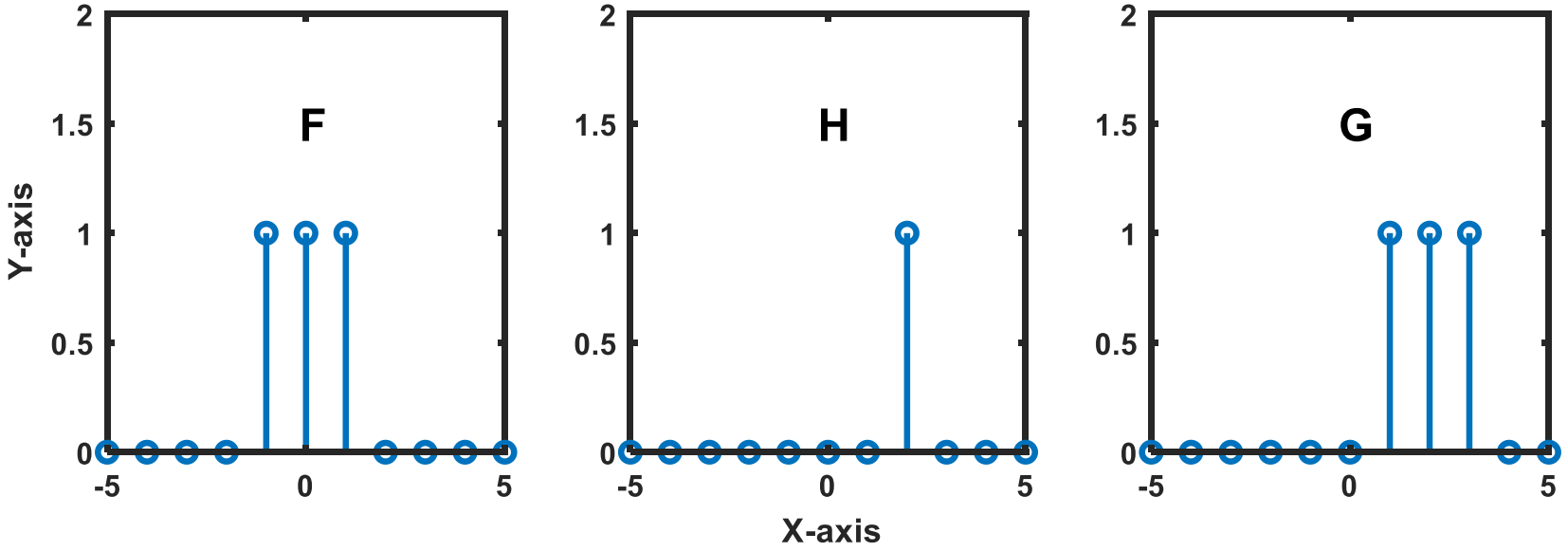


$*$

0	0	0
0	0	1
0	0	0

$=$





$$F * H = G$$

Guessing Game! Linear Filter: Blurring

0	0	0
0	1	0
0	0	0

0	0	0
0	0	1
0	0	0

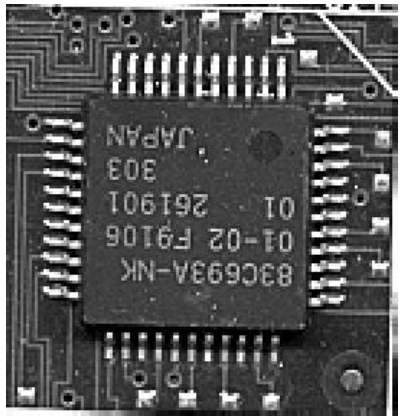
$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

-0.11	-0.11	-0.11
-0.11	1.89	-0.11
-0.11	-0.11	-0.11

0.08	0.12	0.08
0.12	0.20	0.12
0.08	0.12	0.08

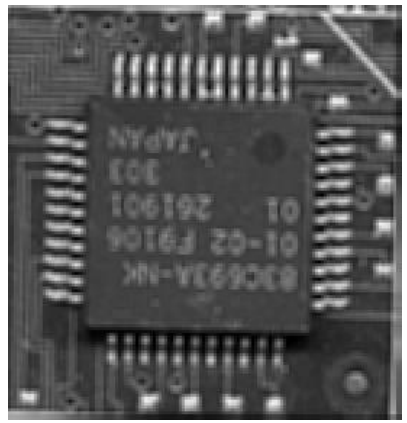
Guessing Game! Linear Filter: Blurring

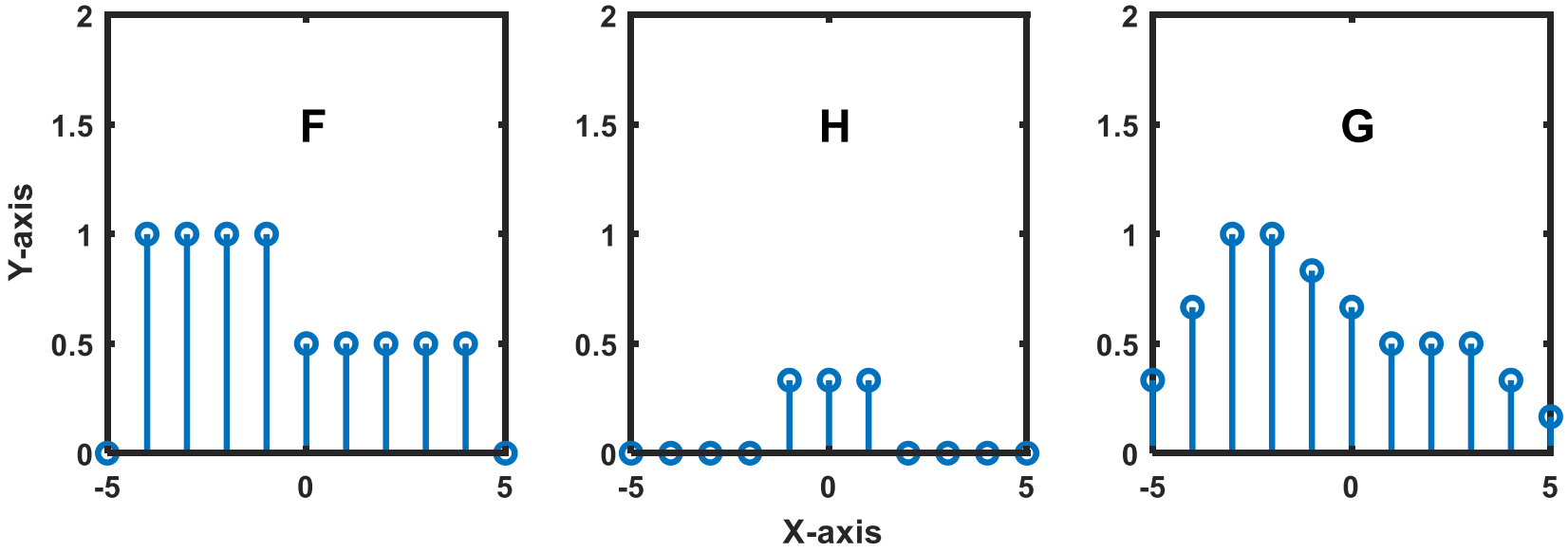


$*$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$=$





$$F * H = G$$

Guessing Game! Linear Filter: Sharpening

0	0	0
0	1	0
0	0	0

0	0	0
0	0	1
0	0	0

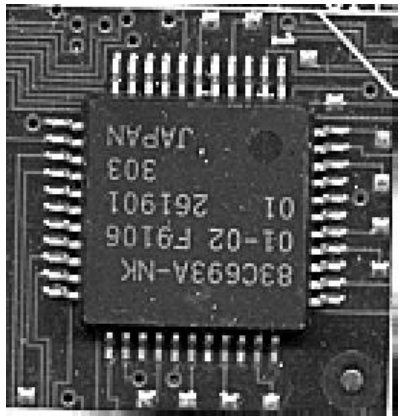
$$\frac{1}{9}$$

1	1	1
1	1	1
1	1	1

-0.11	-0.11	-0.11
-0.11	1.89	-0.11
-0.11	-0.11	-0.11

0.08	0.12	0.08
0.12	0.20	0.12
0.08	0.12	0.08

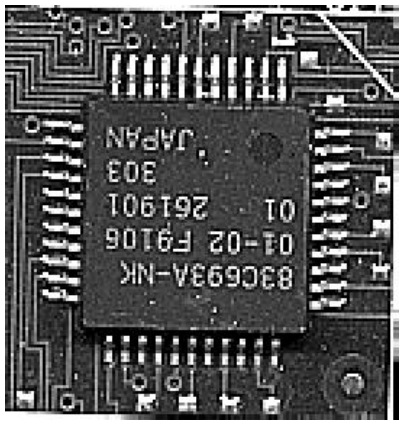
Guessing Game! Linear Filter: Sharpening

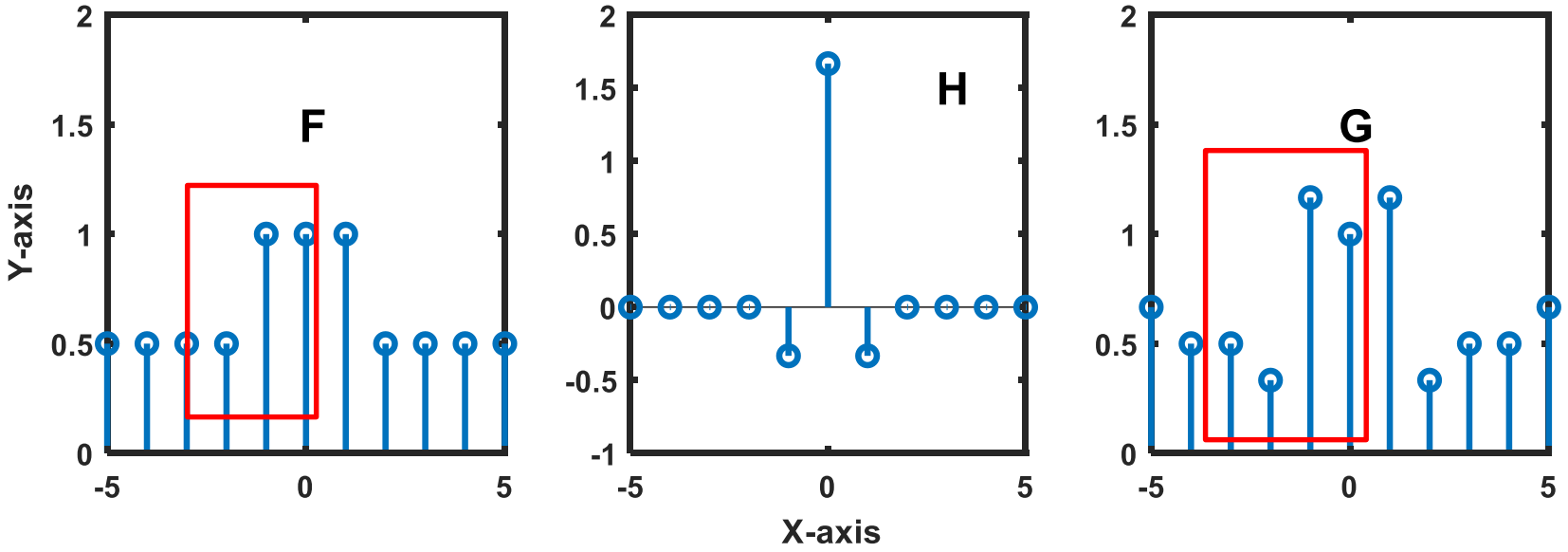


$*$

-0.11	-0.11	-0.11
-0.11	1.89	-0.11
-0.11	-0.11	-0.11

$=$





$$F * H = G$$

How the Sharpening Filter Works

0	0	0
0	2	0
0	0	0

$-$

0.11	0.11	0.11
0.11	0.11	0.11
0.11	0.11	0.11

$=$

-0.11	-0.11	-0.11
-0.11	1.89	-0.11
-0.11	-0.11	-0.11

$$F + (F - F * H) = G$$

0	0	0
0	α+1	0
0	0	0

$-\ \alpha$

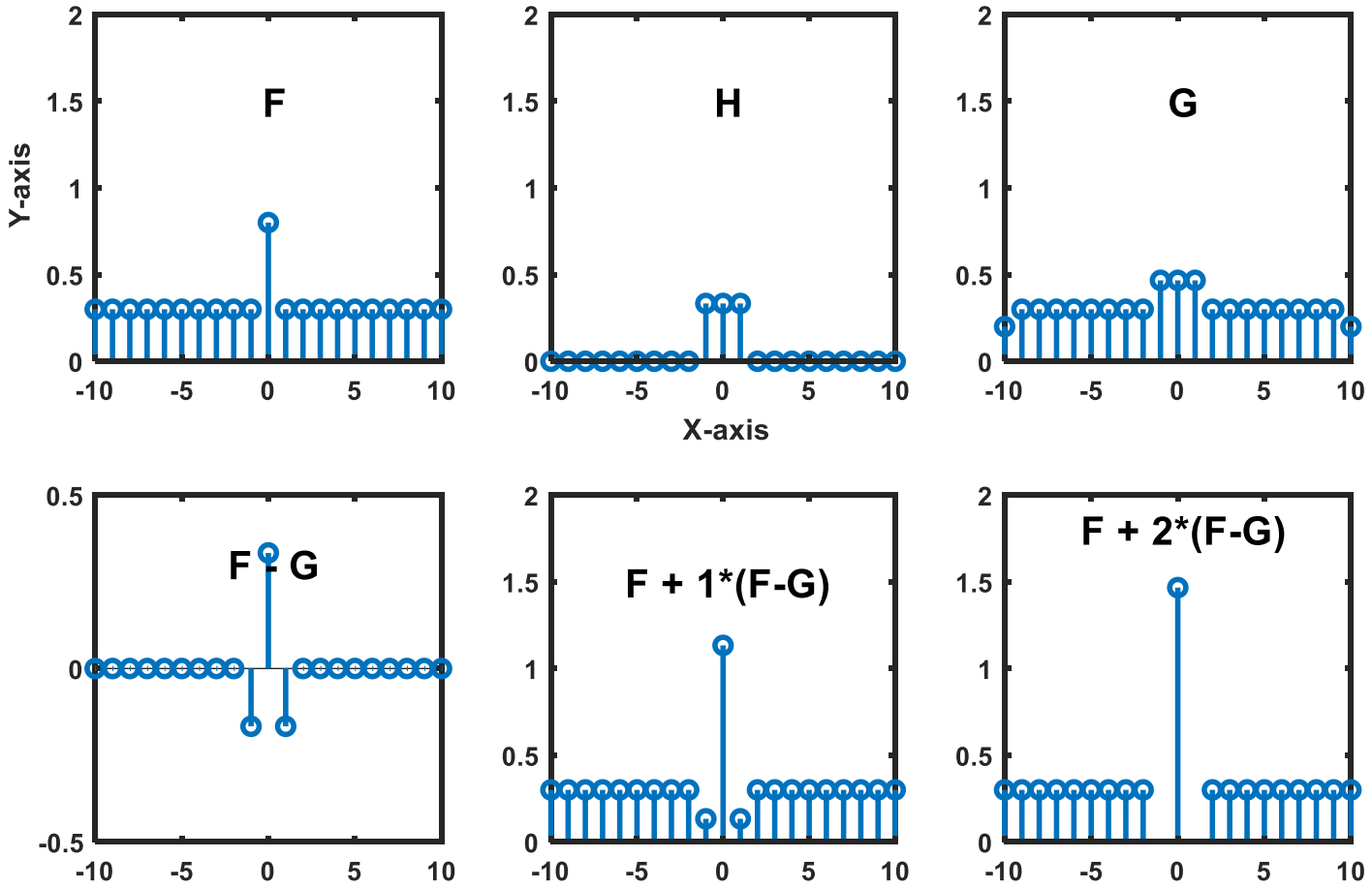
0.11	0.11	0.11
0.11	0.11	0.11
0.11	0.11	0.11

$=$

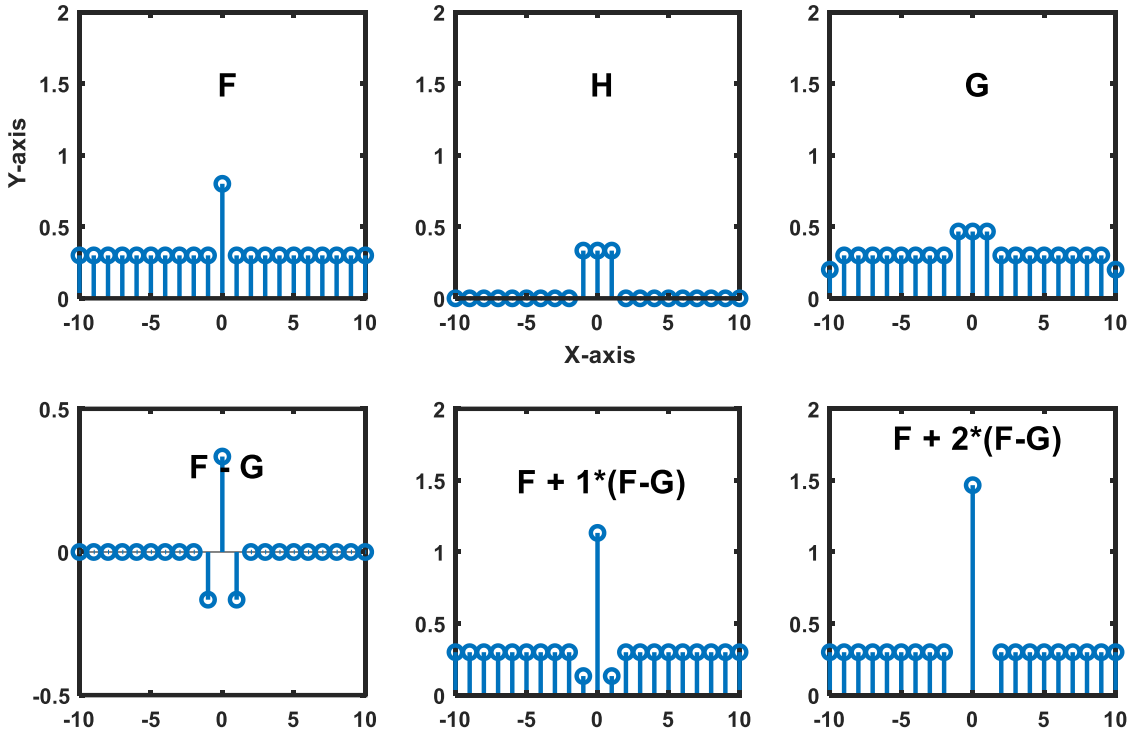
-0.11α	-0.11α	-0.11α
-0.11α	1+0.89α	-0.11α
-0.11α	-0.11α	-0.11α

$$F + \alpha(F - F * H) = G$$

How the Sharpening Filter Works (Signal Example) (Continue)

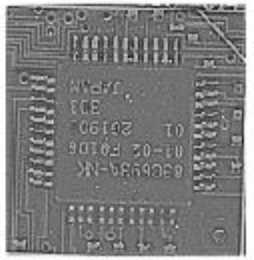


How the Sharpening Filter Works (Image Example)

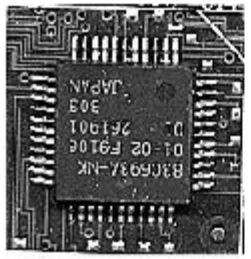


F

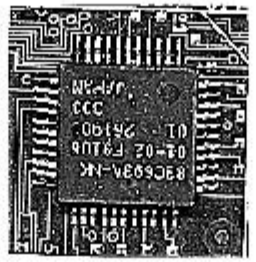
G



$F - G$



$F + (F - G)$



$F + 5(F - G)$

Guessing Game! Gaussian Kernel

0	0	0
0	1	0
0	0	0

0	0	0
0	0	1
0	0	0

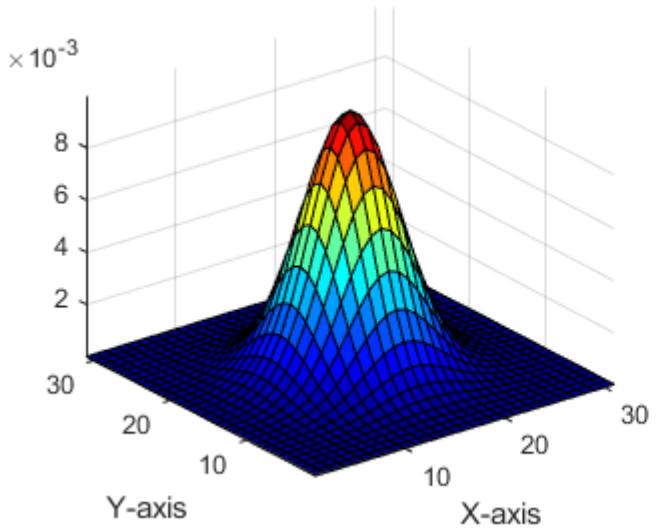
$$\frac{1}{9}$$

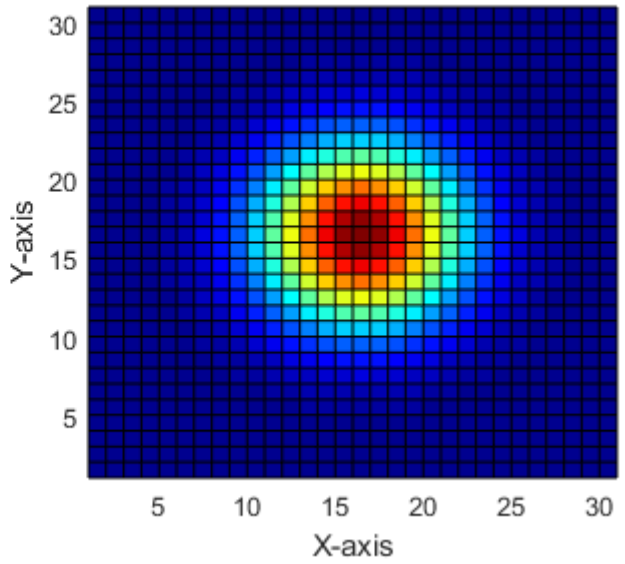
1	1	1
1	1	1
1	1	1

-0.11	-0.11	-0.11
-0.11	1.89	-0.11
-0.11	-0.11	-0.11

0.08	0.12	0.08
0.12	0.20	0.12
0.08	0.12	0.08

Gaussian Kernel





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

Gaussian Filter

```
clear; close all; clc;

h = fspecial('gaussian',3,1);

numSize = 3;
[x, y] = meshgrid(1:numSize);
x = x-(round(numSize/2));
y = y-(round(numSize/2));
sigma = 1;

G_sigma = 1/(2*pi*sigma^2)*exp(-(x.^2 + y.^2)/(2*sigma^2));
G_sigma = G_sigma/sum(G_sigma,'all');

figure(1);
subplot(121); PlotMat(h,gca,'float');
subplot(122); PlotMat(G_sigma,gca,'float');

fig2 = figure(2);
subplot(121); h = fspecial('gaussian', 31,4);
surf(h); axis tight; colormap(jet)
set(fig2,'Position', [100 100 800 300]);
xlabel('X-axis'); ylabel('Y-axis');
subplot(122); surf(h); view(0,90);
xlabel('X-axis'); ylabel('Y-axis'); axis tight
```

0.08	0.12	0.08
0.12	0.20	0.12
0.08	0.12	0.08

0.08	0.12	0.08
0.12	0.20	0.12
0.08	0.12	0.08

Effect of Gaussian Window Sizes

$f1$

Std=1

$f2$

Std=5

$f3$

Std=10

$f4$

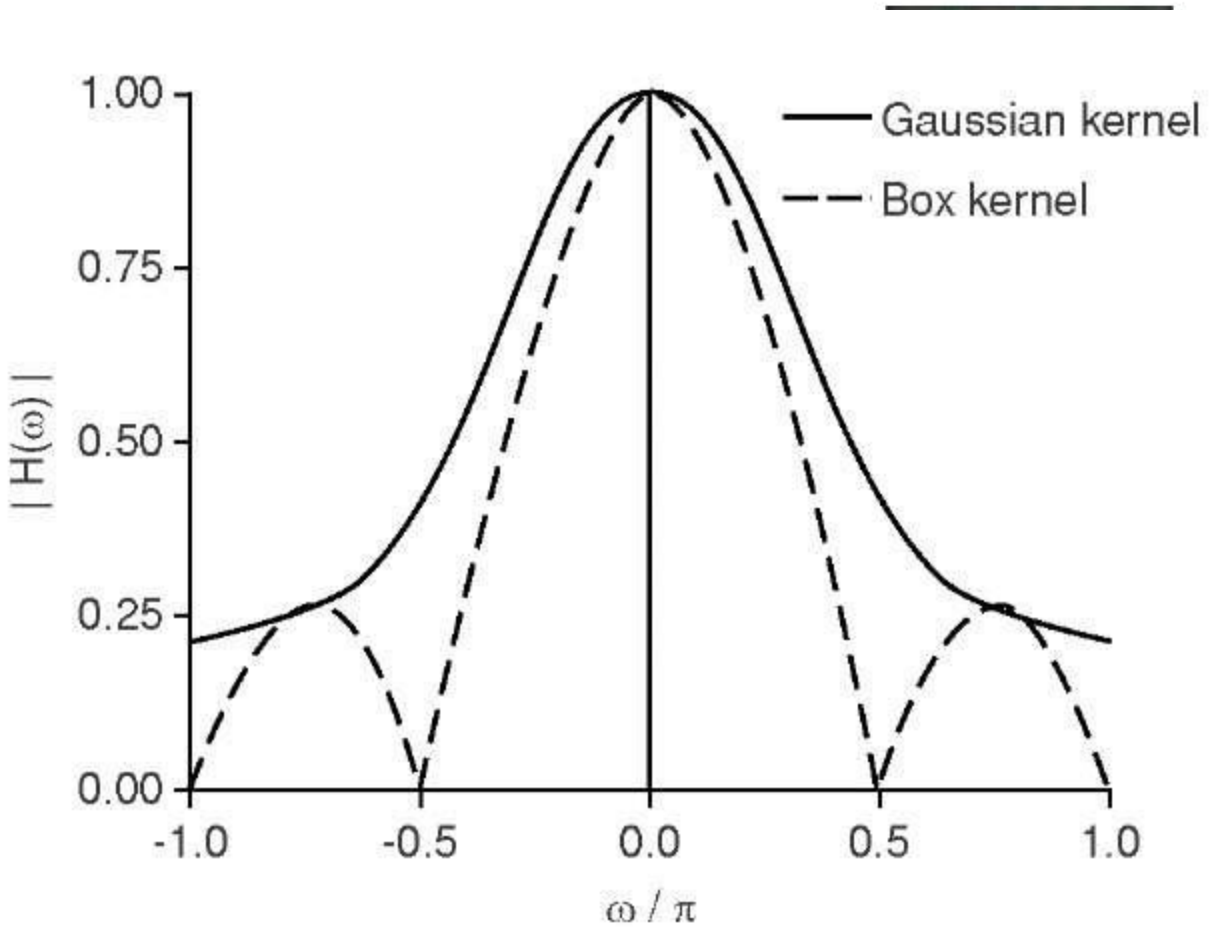
Std=30

```
f1 =  fspecial('gaussian', 101,1);
f2 =  fspecial('gaussian', 101,5);
f3 =  fspecial('gaussian', 101,10);
f4 =  fspecial('gaussian', 101,30);
```

Difference of the Between a Gaussian Filter and Box Filter

```
f1 = fspecial('gau
f2 = fspecial('ave

figure(3);
subplot(121); imsh
subplot(122); imsh
```

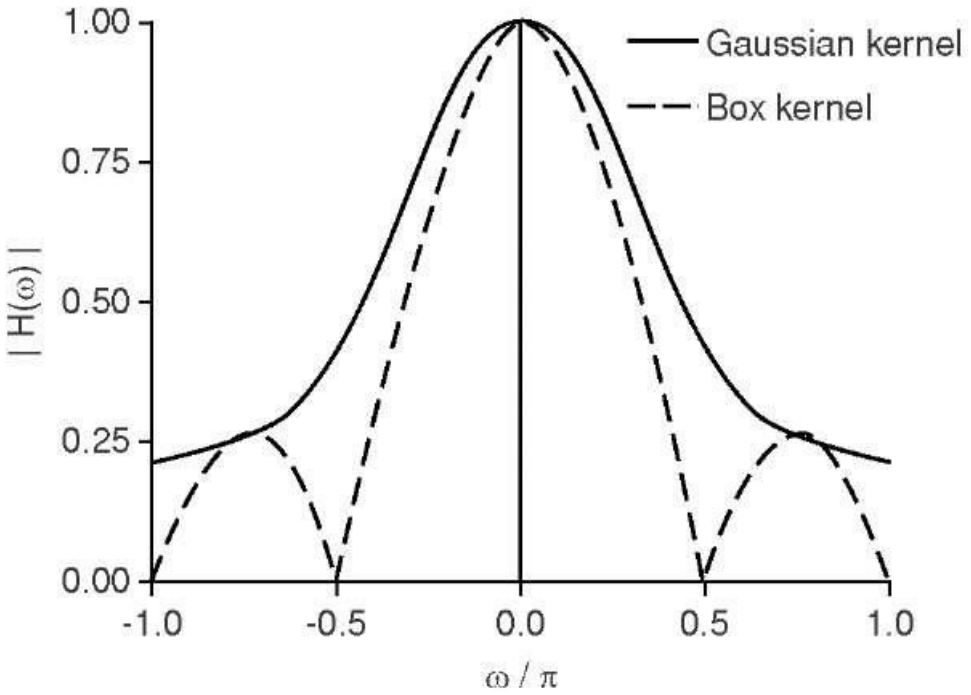


ıssian (f2)

https://dsp.stackexchang
should-be-considered-wh
function-when-smoothin

https://stackoverflow.com/questions/31131672/differe
nce-between-mean-and-gaussian-filter-in-result

Difference of the Between a Gaussian Filter and Box Filter



Box (f1)

Gaussian (f2)

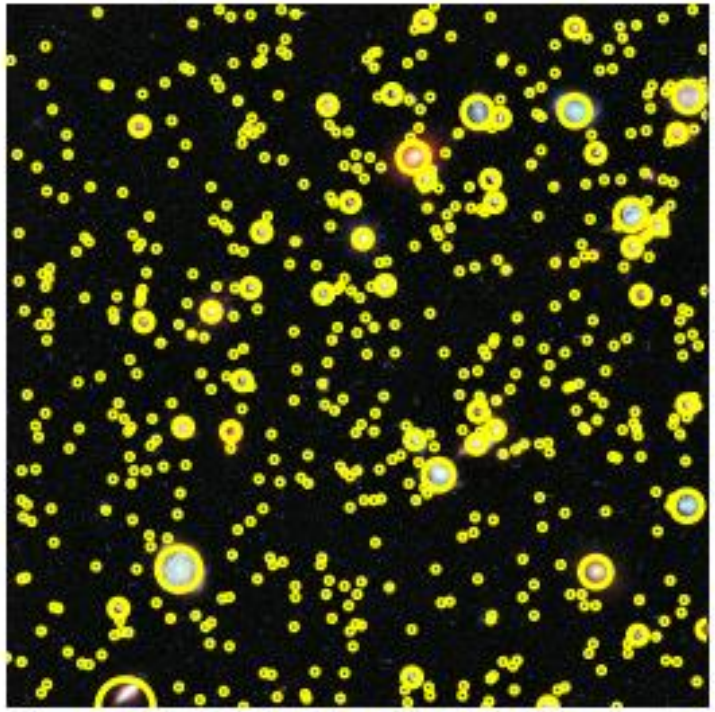
https://dsp.stackexchange.com/questions/208/what-should-be-considered-when-selecting-a-windowing-function-when-smoothing-a-t

https://stackoverflow.com/questions/31131672/difference-between-mean-and-gaussian-filter-in-result

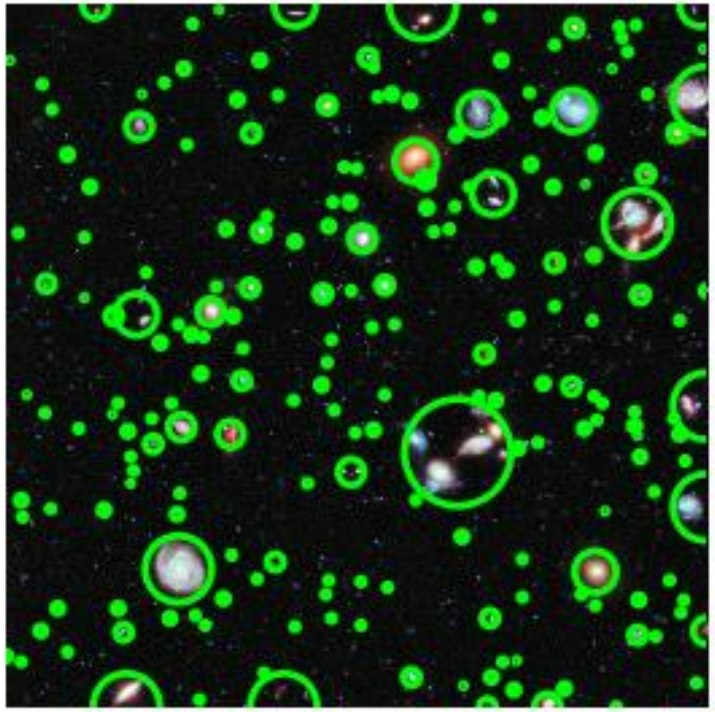
Applications: Edge detection

Applications: Blob Detection (for feature detection)

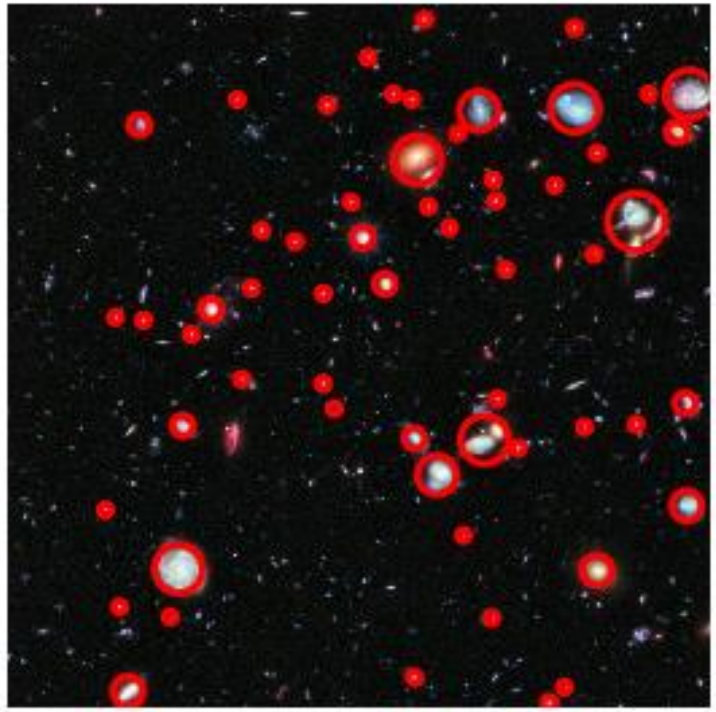
Laplacian of Gaussian



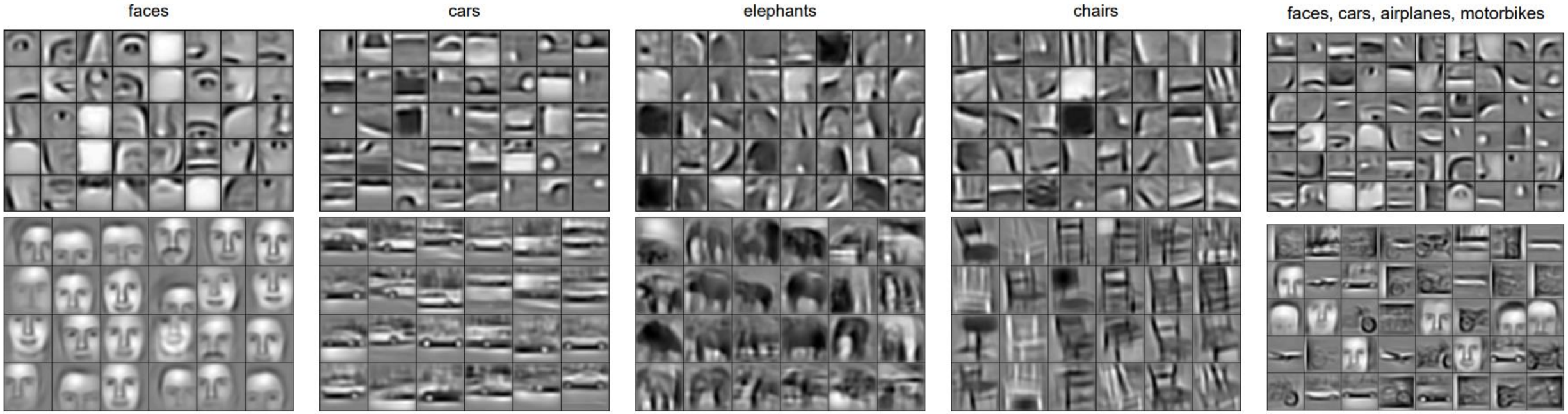
Difference of Gaussian



Determinant of Hessian



Applications: Convolution Neural Network



http://web.eecs.umich.edu/~honglak/icml09-ConvolutionalDeepBeliefNetworks.pdf

Slide Credits and References

- Lecture notes: Rob Fergus.
- Lecture notes: Steve Seitz
- Lecture notes: Mohammad Jahanshahi
- Lecture notes: Svetlana Lazebnik
- Lecture notes: Derek Hoiem
- Lecture notes: Ioannis (Yannis) Gkioulekas
- Lecture notes: Robert Collins
- Lecture notes: Jason Corso
- Lecture notes: Gordon Wetzstein
- Lecture notes: Noah Snavely