

Linear Filtering

Chul Min Yeum
Assistant Professor
Civil and Environmental Engineering
University of Waterloo, Canada

CIVE 497 – CIVE 700: Smart Structure Technology
Last updated: 2021-12-06



UNIVERSITY OF WATERLOO
FACULTY OF ENGINEERING

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
 - Color depth (8bit or 16 bit). Thus:

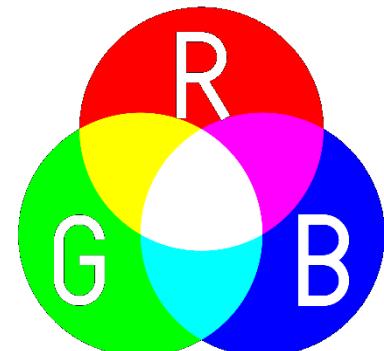
$$f: [0, w] \times [0, h] \rightarrow [0, 1],$$

$w = \text{width}, h = \text{height}$

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

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

Q. What is the color depth?



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

Example: 8bit Vs 16 bit Images

Supplement

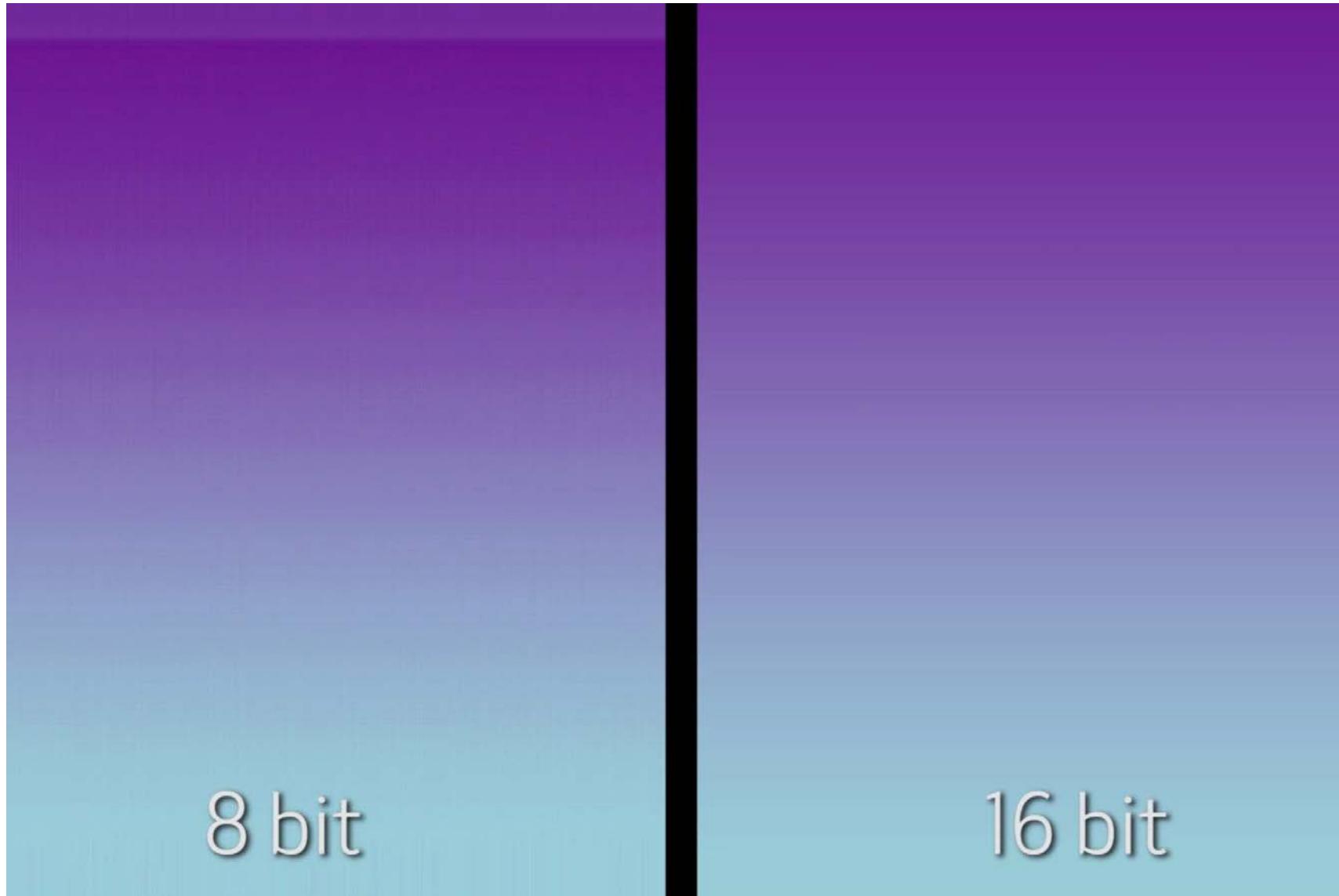
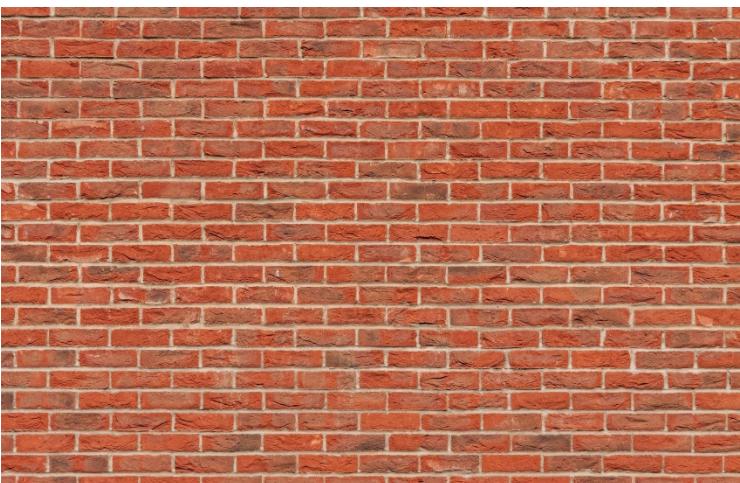


Image as Functions (Continue)

RGB



Gray

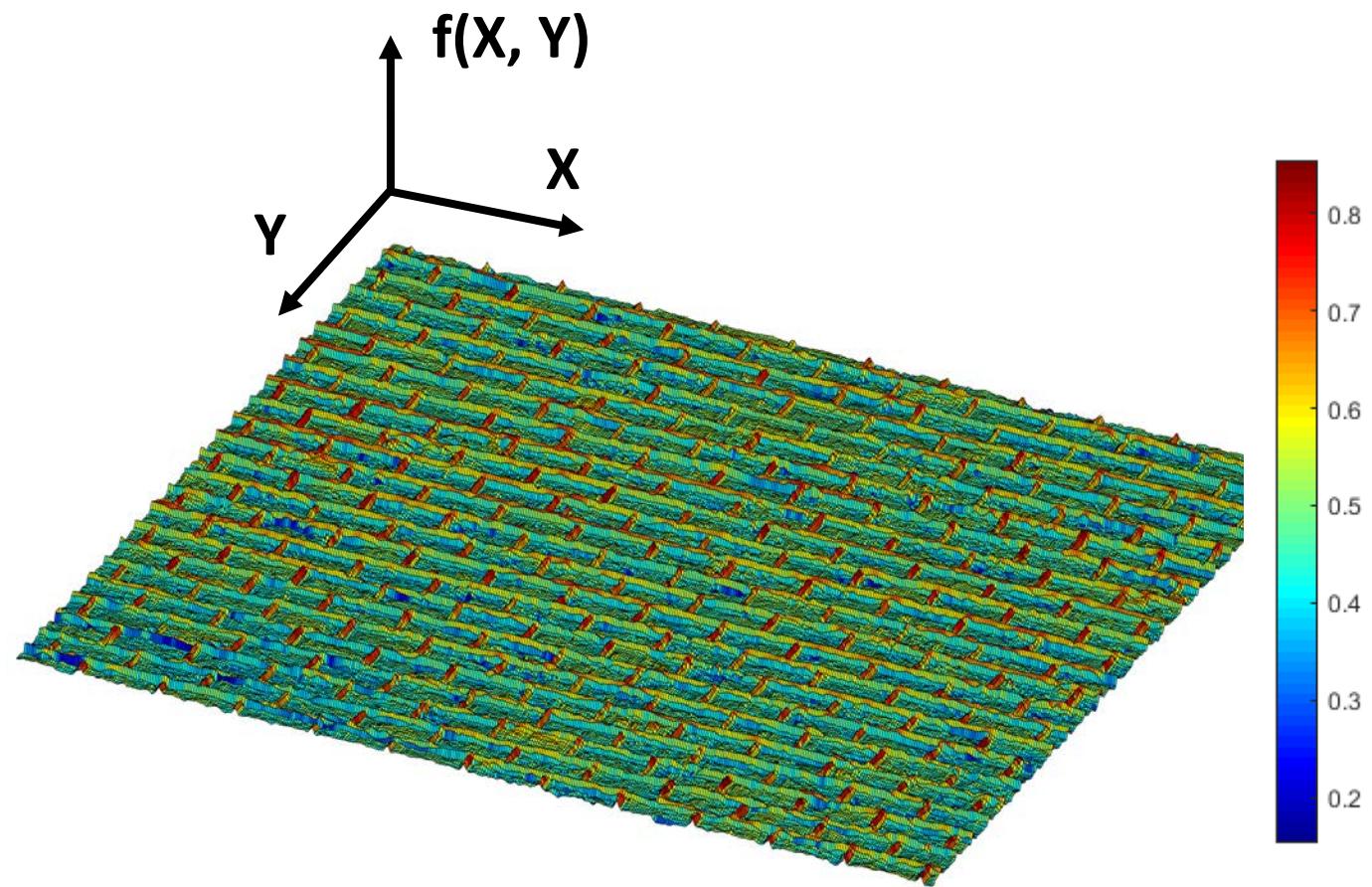
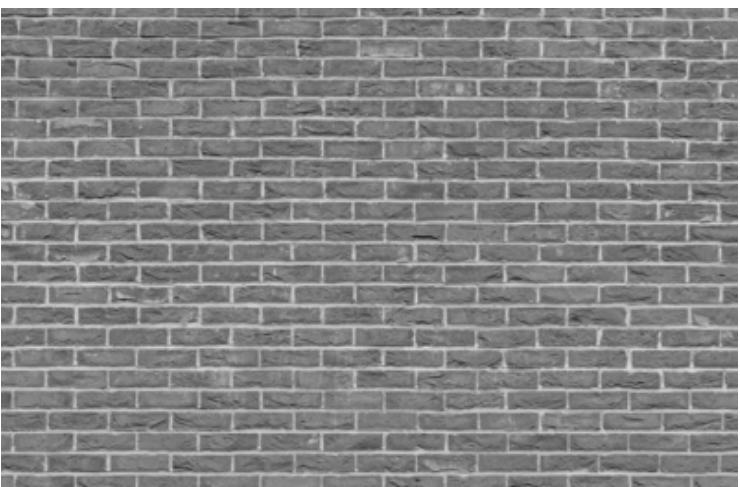
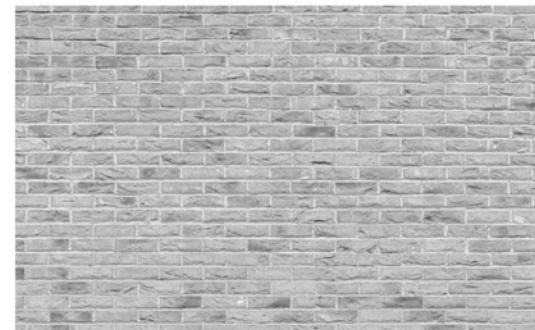
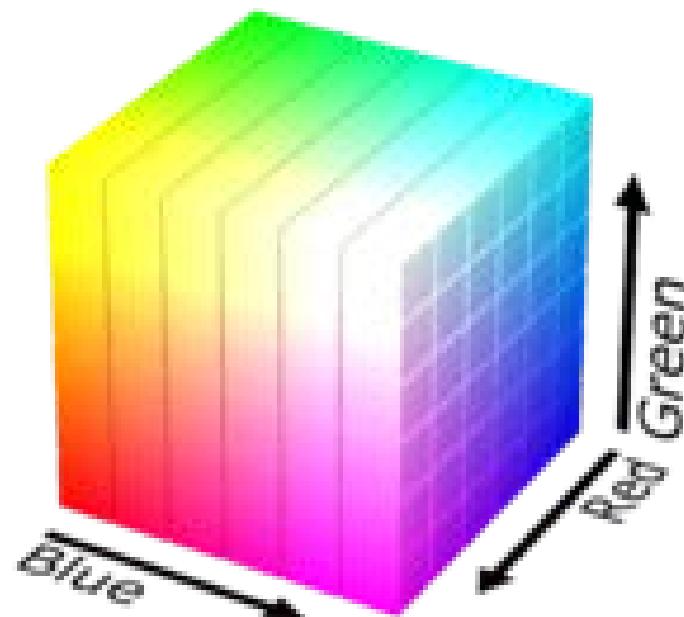
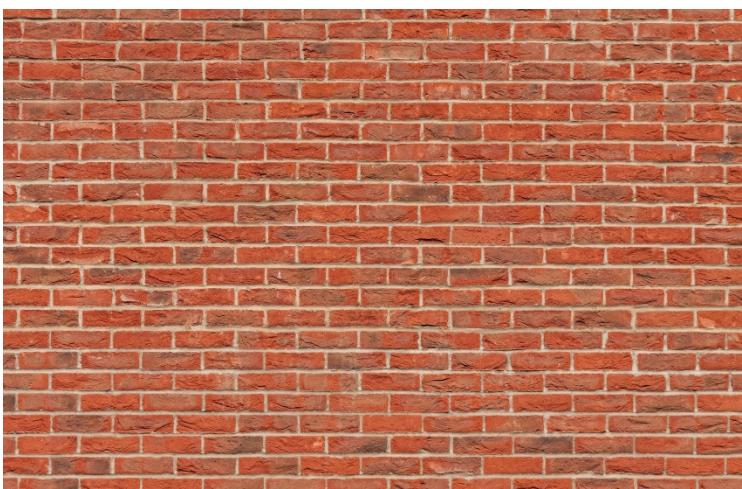
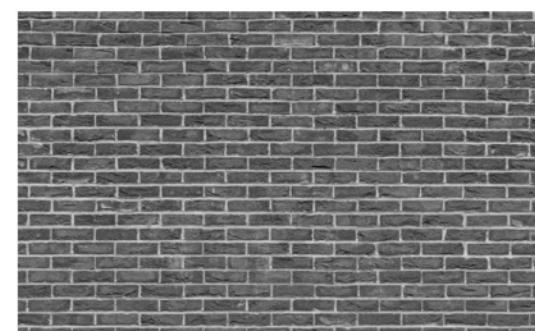


Image as Functions (Continue)

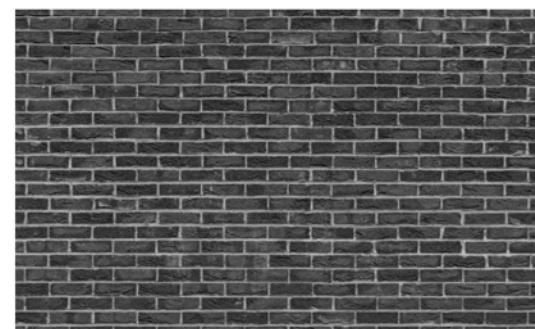
RGB



R



G



B

Grayscale in each channel 5

Linear Filtering?

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

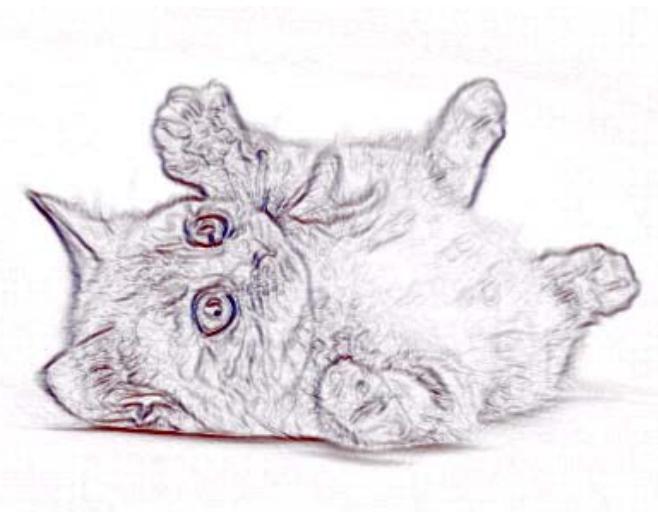
Linear Filtering?

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

Noisy image

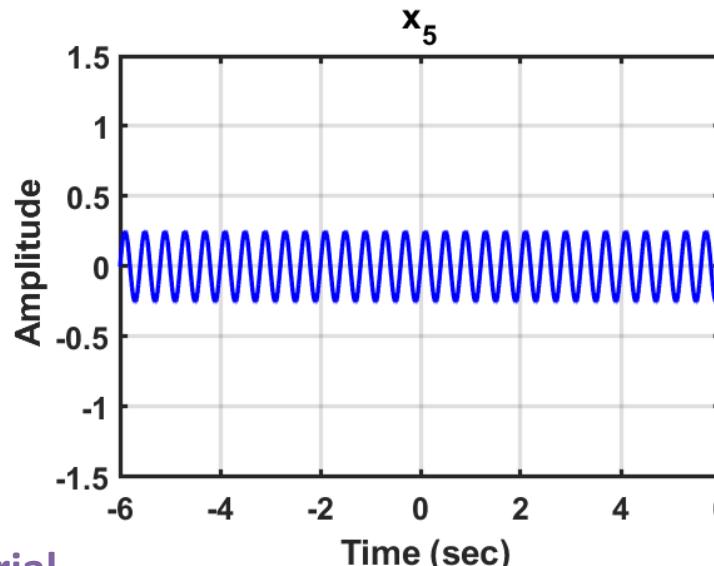
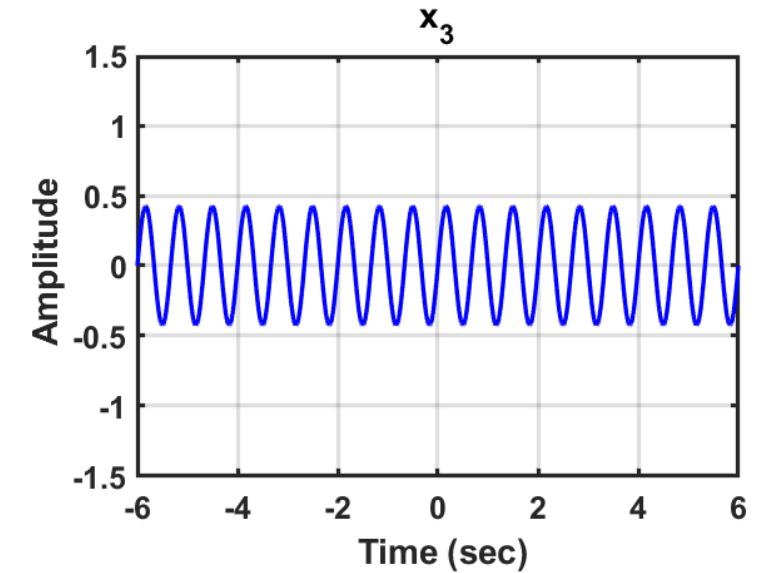
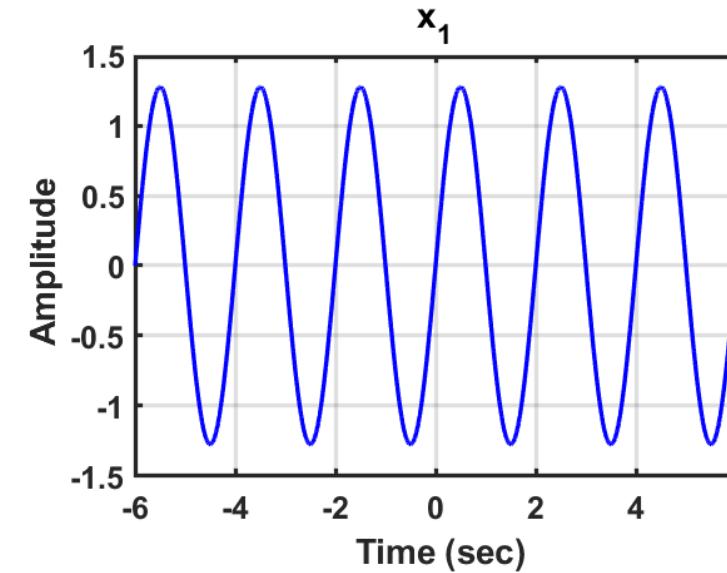
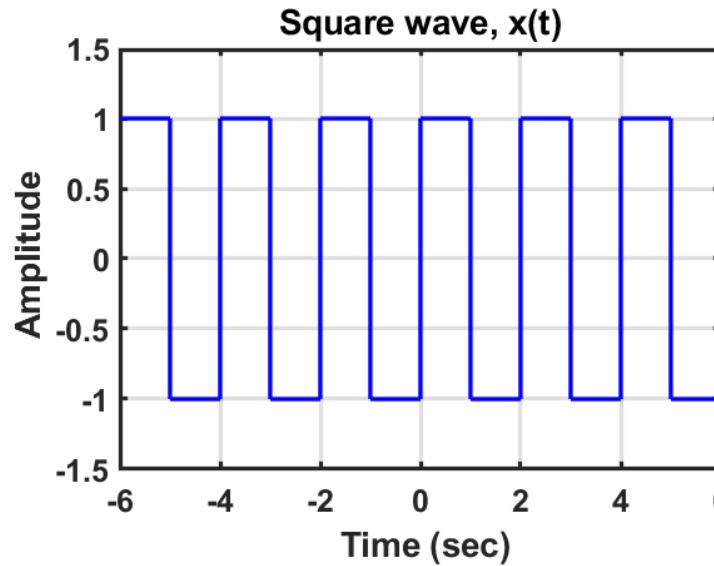


Denoised image



Recall: Fourier Series of a Square Wave

Supplement



$$x(t) = \sum_{n=1}^{\infty} \frac{2}{n\pi} (1 - \cos n\pi) * \sin\left(\frac{2\pi nt}{T_p}\right) = \sum_{n=1}^{\infty} x_n(t)$$

Recall: Fourier Series of a Square Wave (Continue)

Supplement

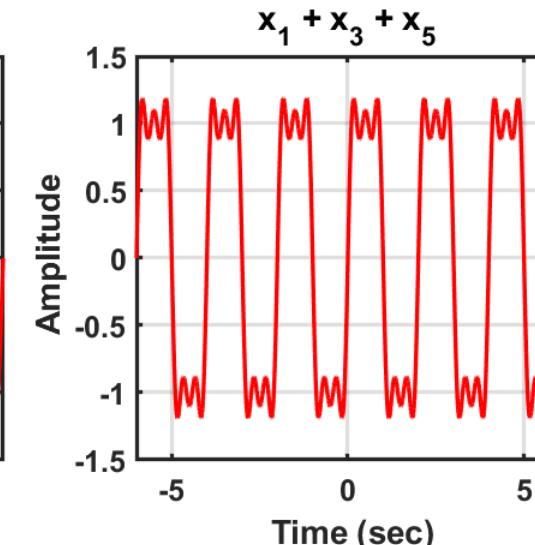
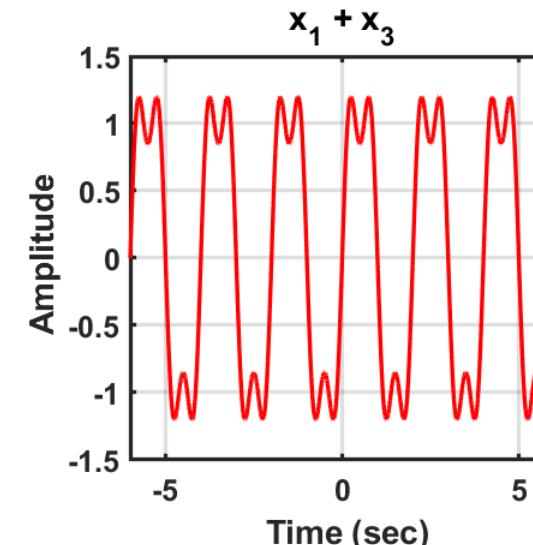
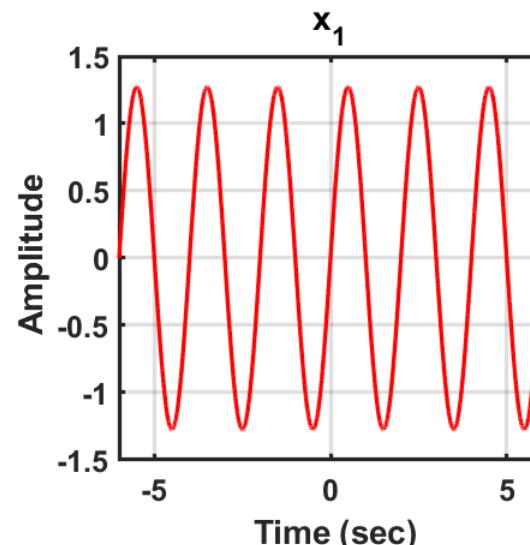
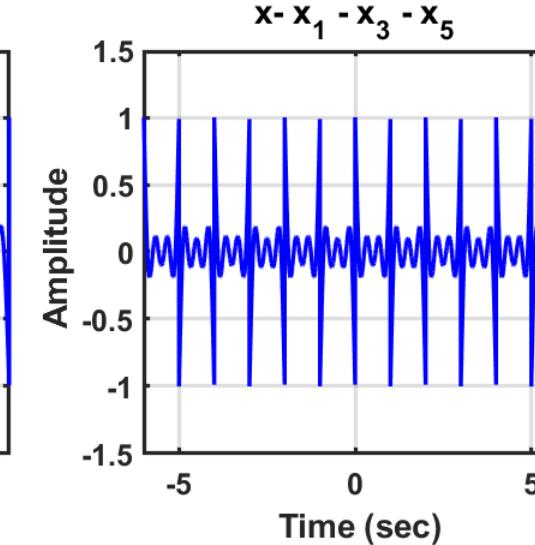
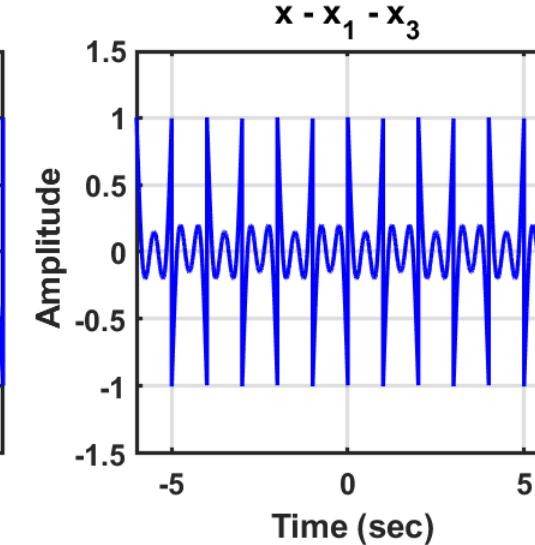
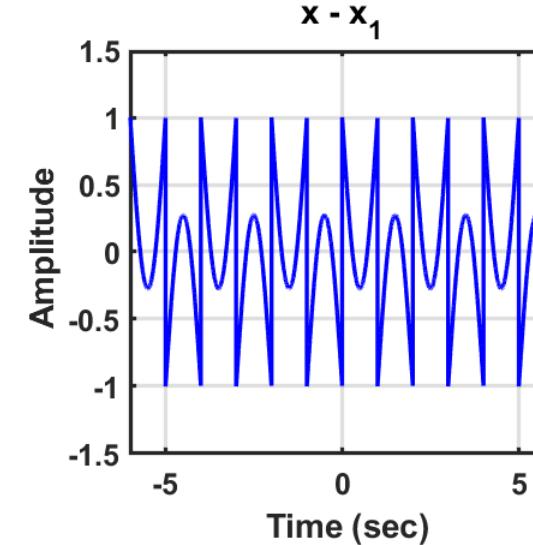
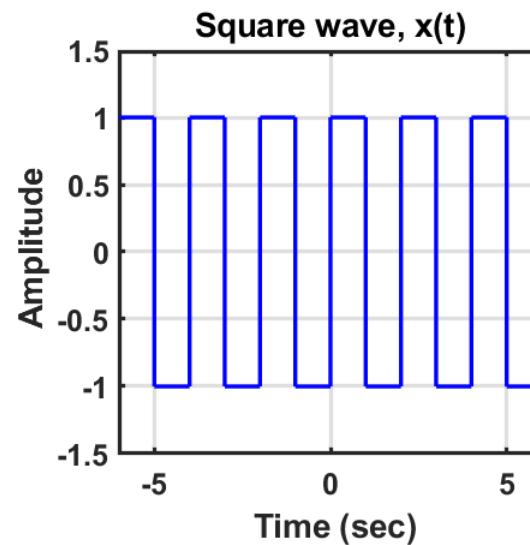


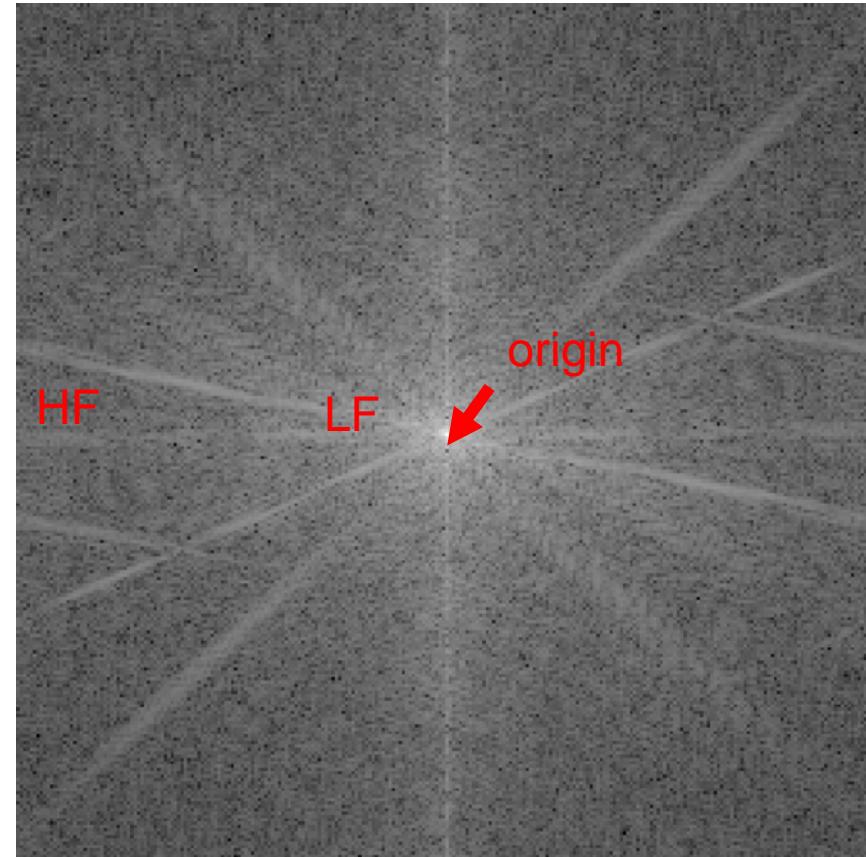
Image in a Frequency Domain

- Here is an image and its Fourier transform

Low frequency (LF) \approx solid colors
High frequency (HF) \approx noise/edges



2D image



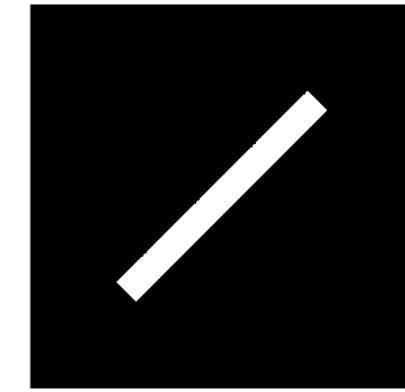
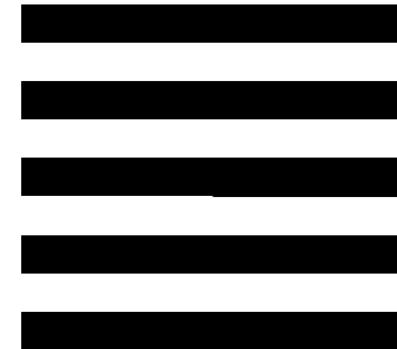
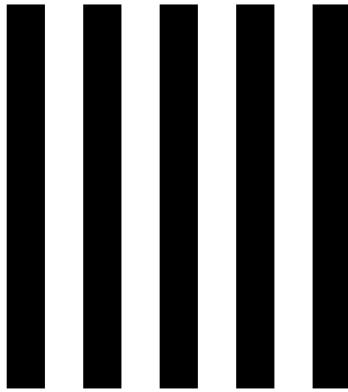
2D FFT

Y-axis

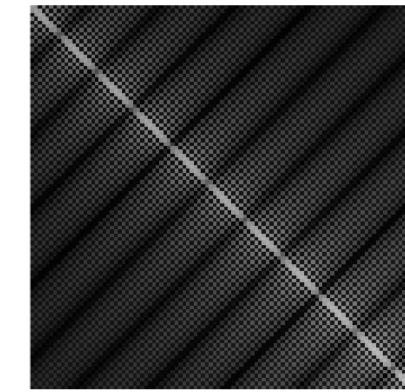
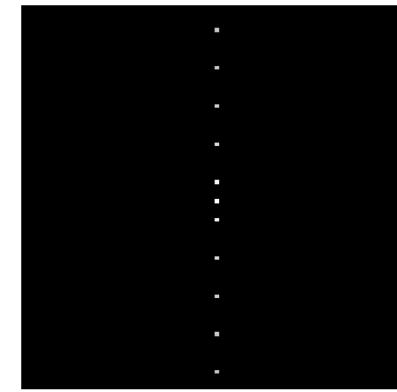
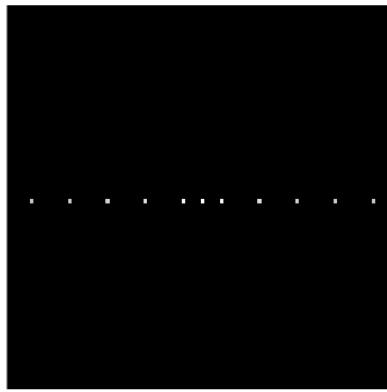
Image in a Frequency Domain (Patterns)

Supplement

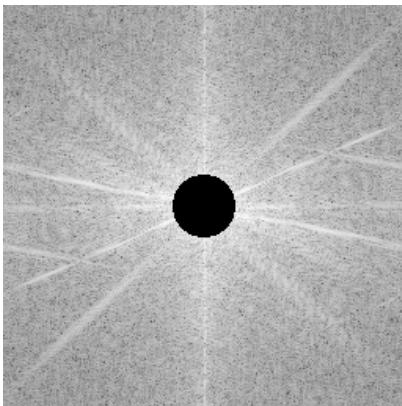
Image



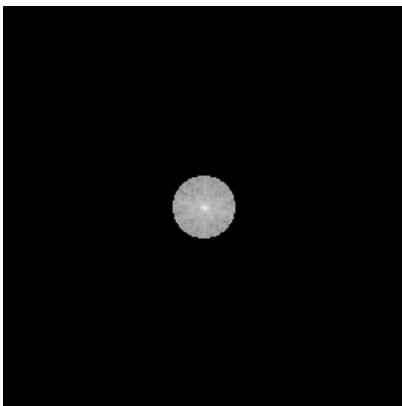
Frequency



How Filtering Works in Frequency



Remove low frequency (only pass high frequency)



Remove high frequency (only pass low frequency)



Highlighting edge



Blurring

Linear Filtering: Simple Example

- One simple version : linear filtering (cross-correlation or convolution)
 - Replace each pixel by a linear combination of its neighbors
- The prescription for the linear combination (how to compute the value using its own and nearby pixels) is called the “kernel” (or “mask” or “filtering”)

1	1	1
1	1	1
1	1	1



1	1	1
1	1	1
1	1	1

Kernal

0	0	0
0	9	0
0	0	0

Modified image data

Local image data

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):

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

This is called a cross-correlation operation:

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

$$g = h \otimes f$$

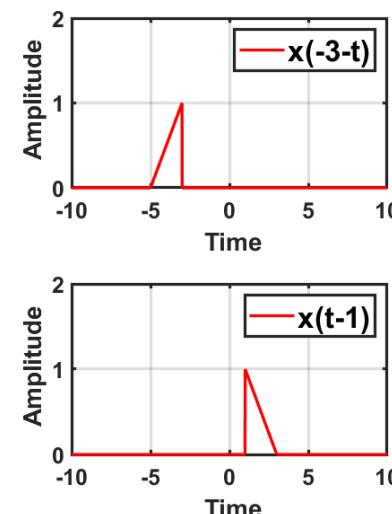
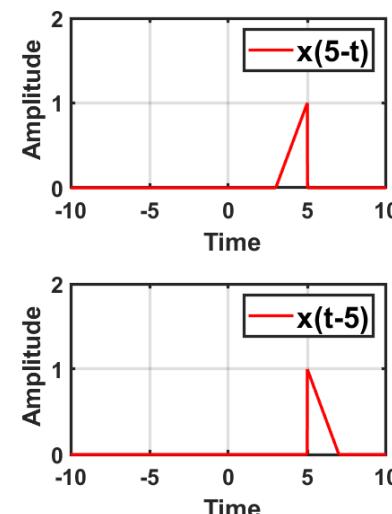
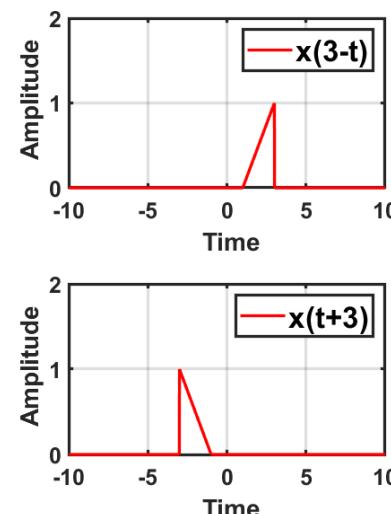
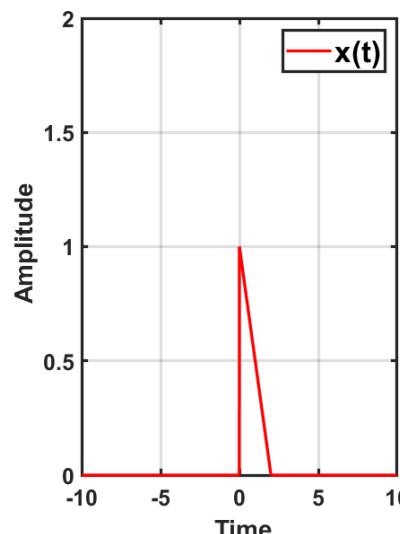
Convolution

Comparison between Cross-Correlation and Convolution

Supplement

Cross-correlation

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



Kernel in the cross-correlation

Convolution (commutative)

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

Replace $t - \tau$ to τ'

$$h(t) \otimes x(t) = \int_{-\infty}^{\infty} x(\tau')h(t - \tau')d\tau'$$

Kernel in the convolution

2D Cross-Correlation Using a Kernel

Cross-correlation is a measure of similarity between the 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 :**

Q. Why is this a measure of similarity?

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

Again, this is called a cross-correlation operation:

$$G = H \otimes F$$

2D Cross-Correlation Example

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

Point i, j

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

\otimes

$H[0, 0]$

1	1	1
1	1	1
1	1	1

=

4	7	5	4	2
6	9	7	5	3
4	6	6	5	4
3	4	4	3	2
1	1	2	3	4

$F[x, y]$

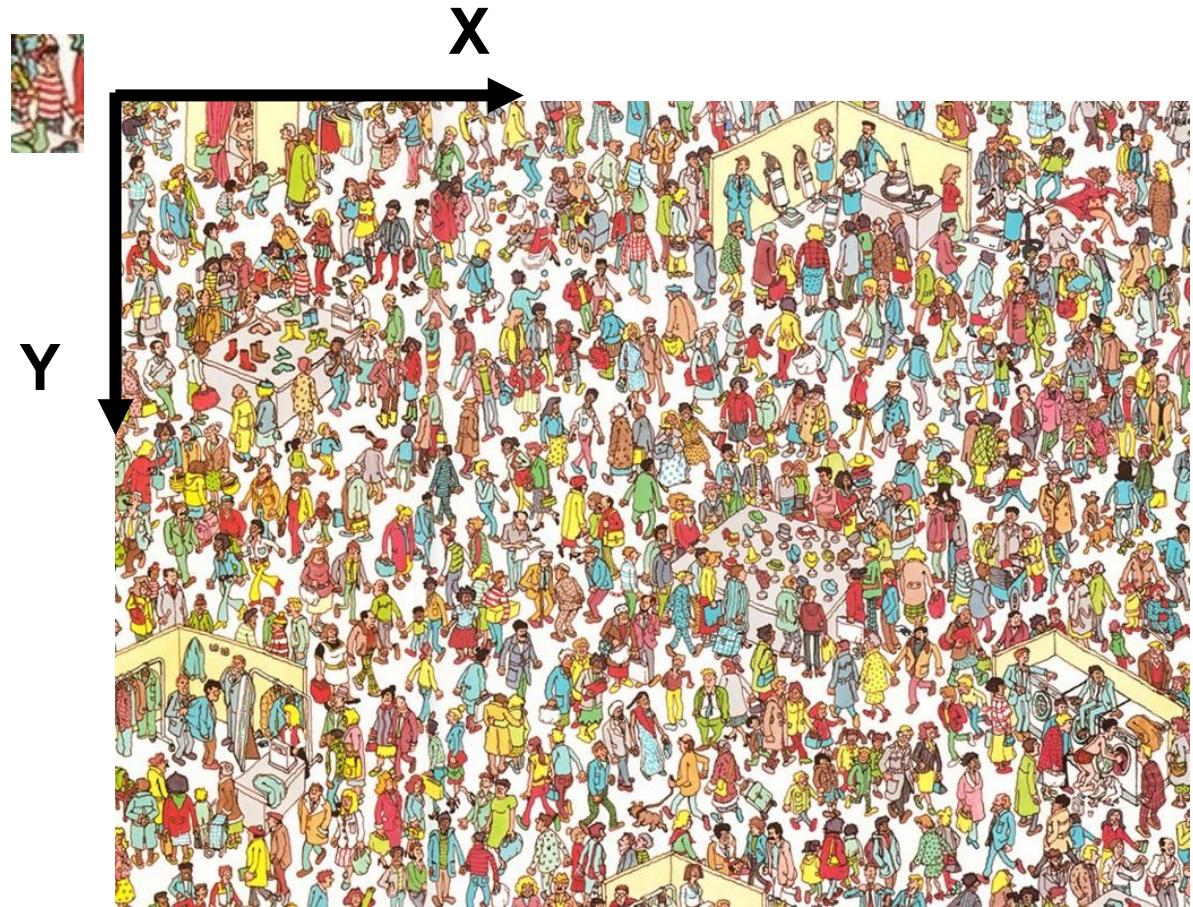
$H[x, y]$

$G[x, y]$

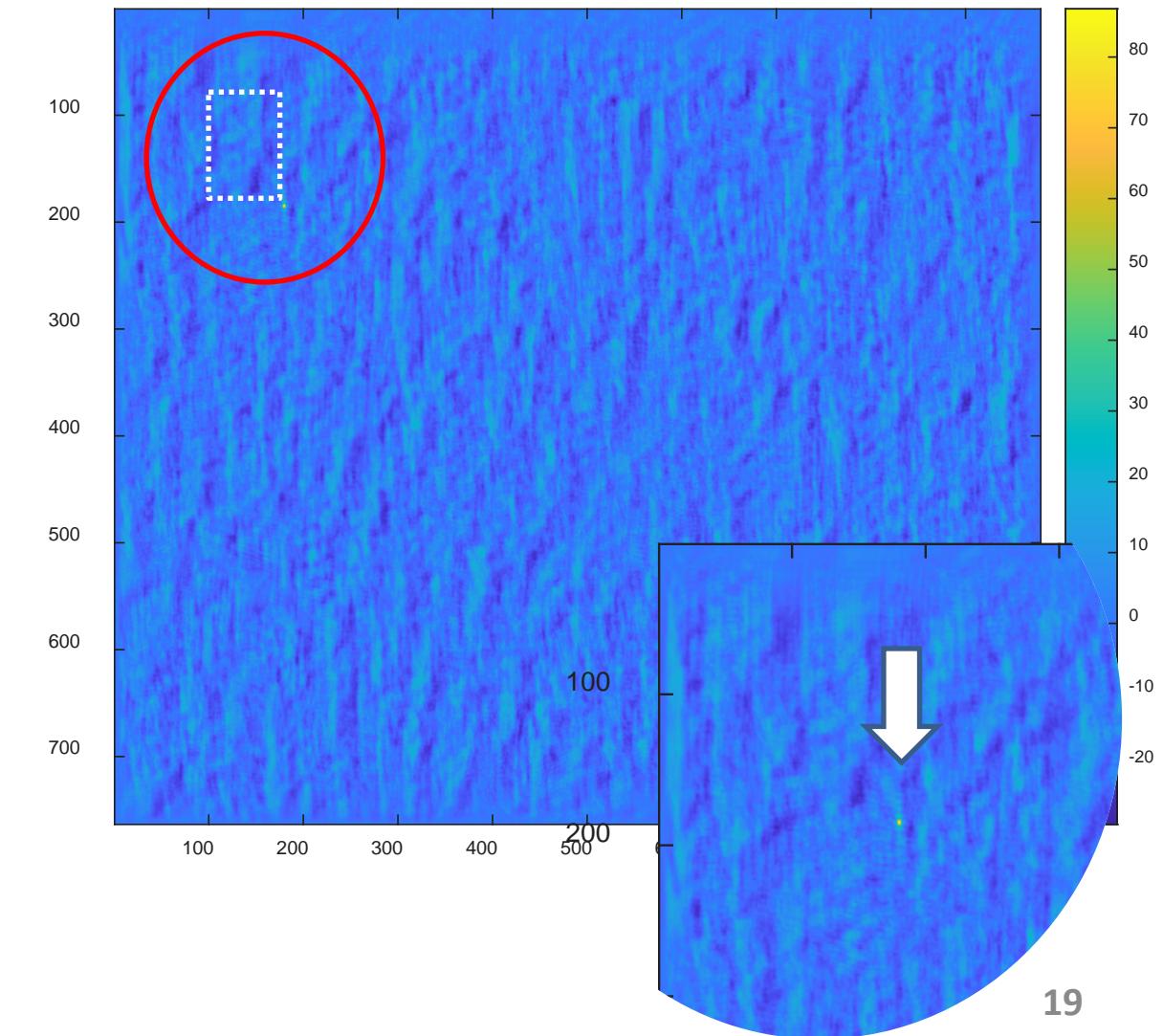
Computation of the Cross-Correlation Between Two Matrices



Example: Finding Waldo from the Image (Template Matching)



Cross-correlation: Scanning the (pre-processed) image with a waldo image.

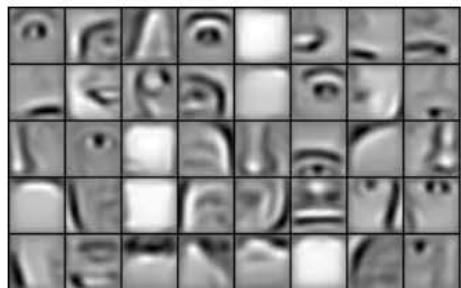


Challenges of the Template Matching

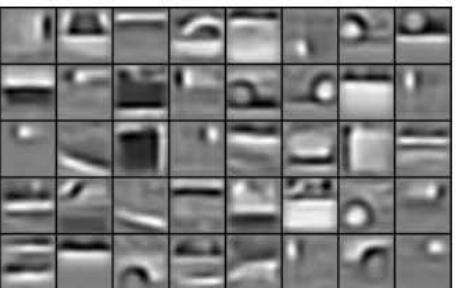


Applications: Convolution Neural Network

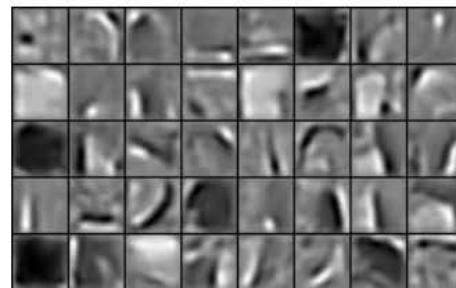
faces



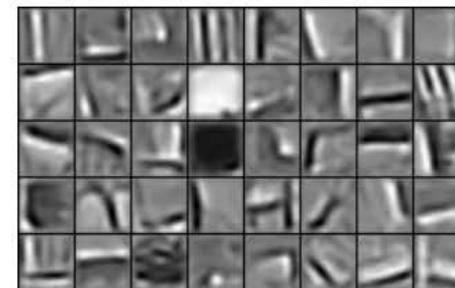
cars



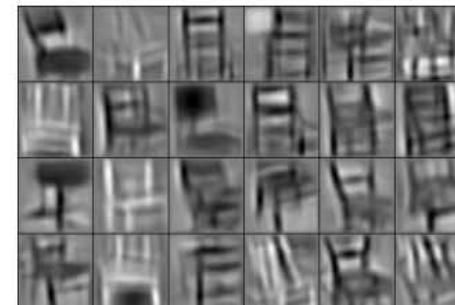
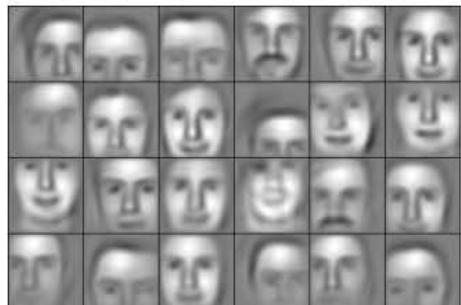
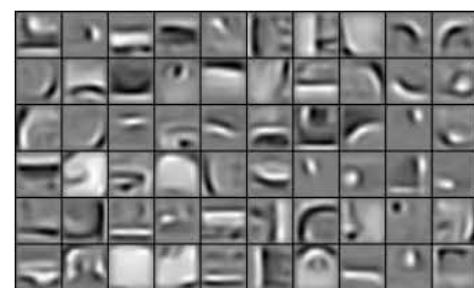
elephants



chairs



faces, cars, airplanes, motorbikes



2D Convolution

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

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: $(F * H)' = F' * H = F * H'$

Linear Filter: Generating an Identical Image

0	0	0
0	1	0
0	0	0

H_i

0	0	0
0	0	1
0	0	0

H_s

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

H_r

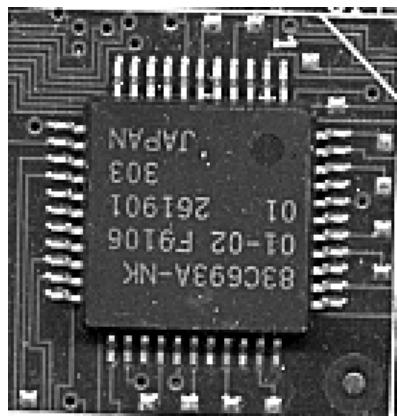
-0.11	-0.11	-0.11
-0.11	1.89	-0.11
-0.11	-0.11	-0.11

H_p

0.08	0.12	0.08
0.12	0.20	0.12
0.08	0.12	0.08

H_g

Generating an Identical Image

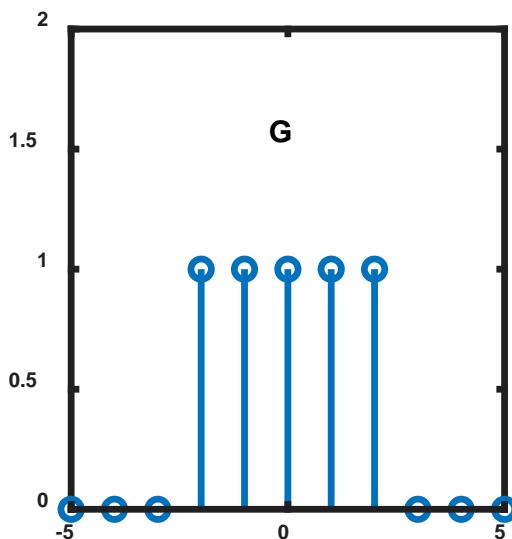
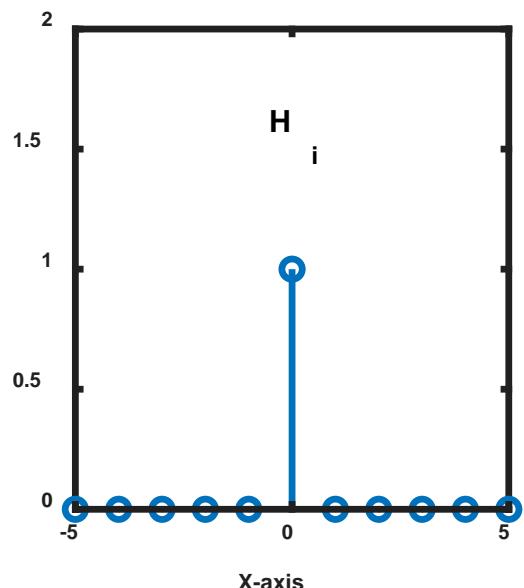
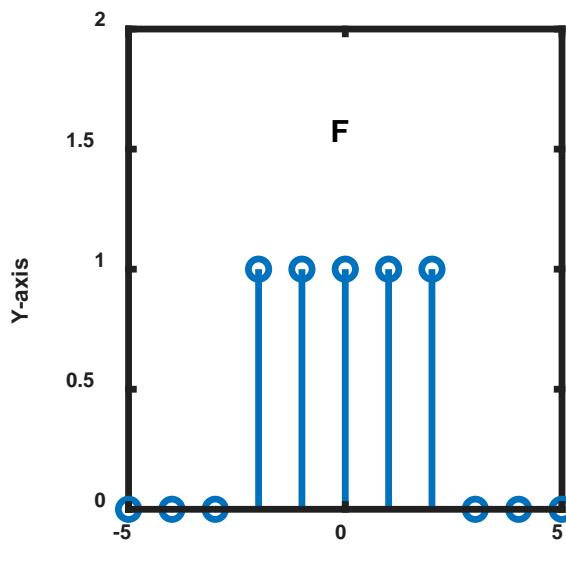
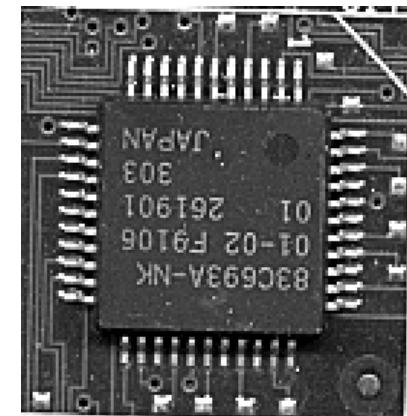


*

0	0	0
0	1	0
0	0	0

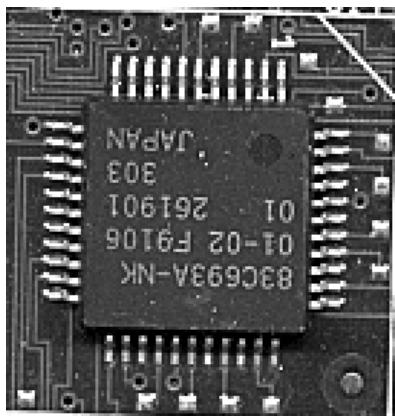
H_i

||



$$F * H_i = G$$

Shifting Pixels

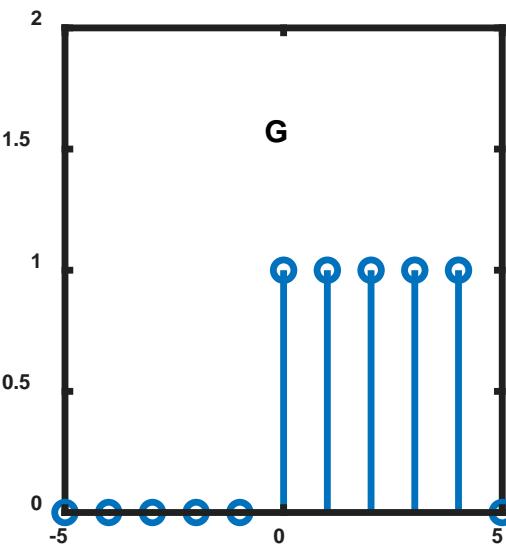
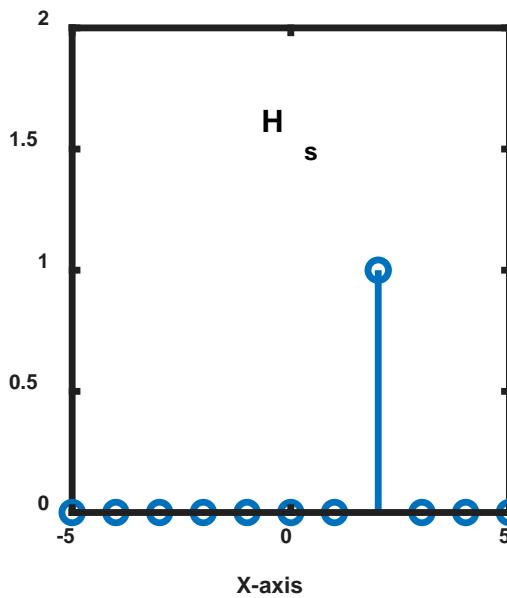
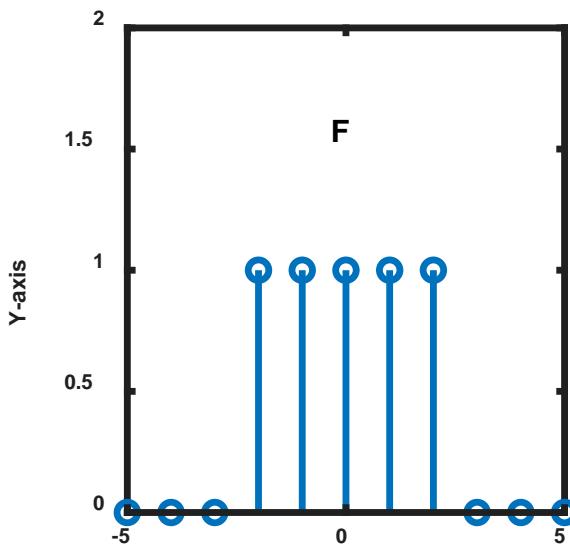


*

0	0	0
0	0	1
0	0	0

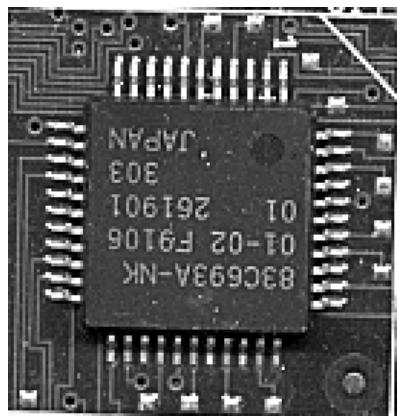
==

H_S



$$F * H_s = G$$

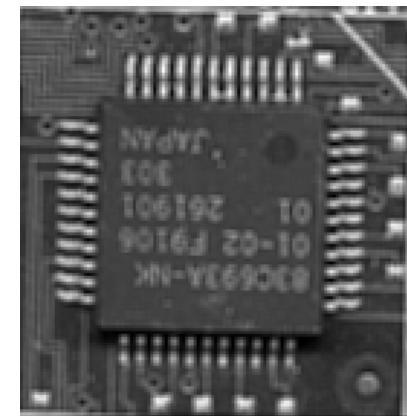
Blurring



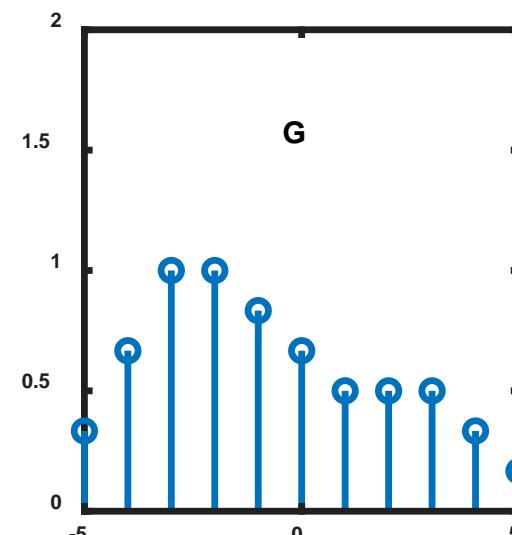
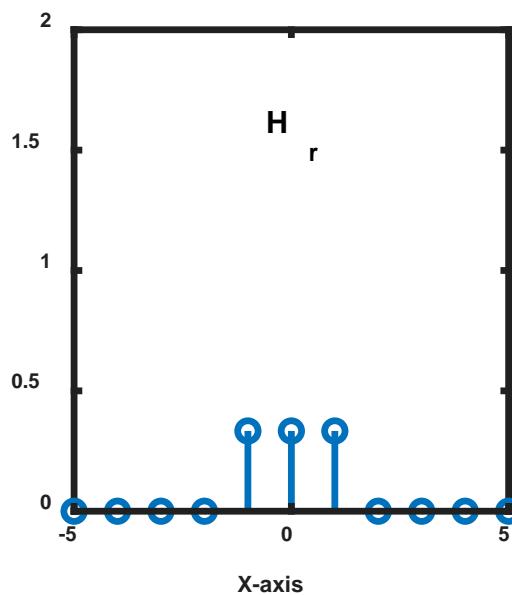
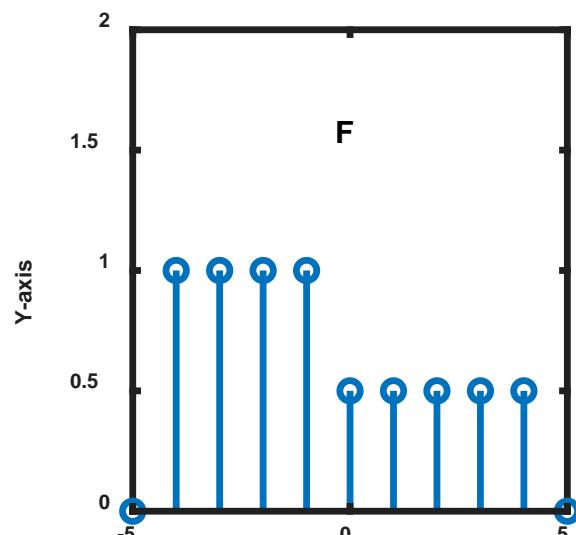
*

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

||

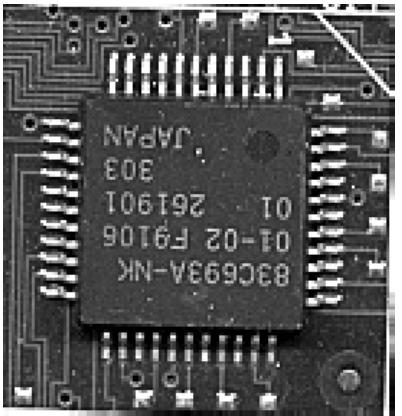


H_r



$$F * H_r = G$$

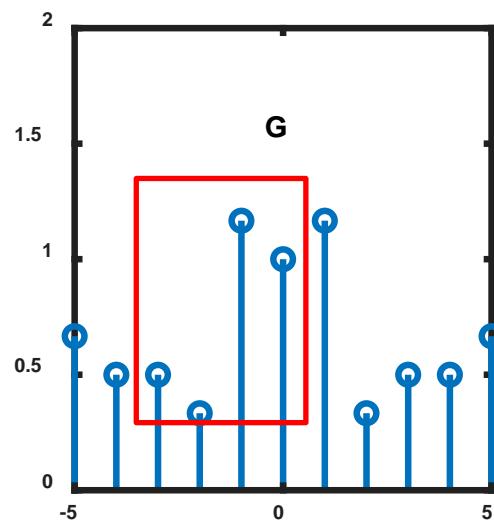
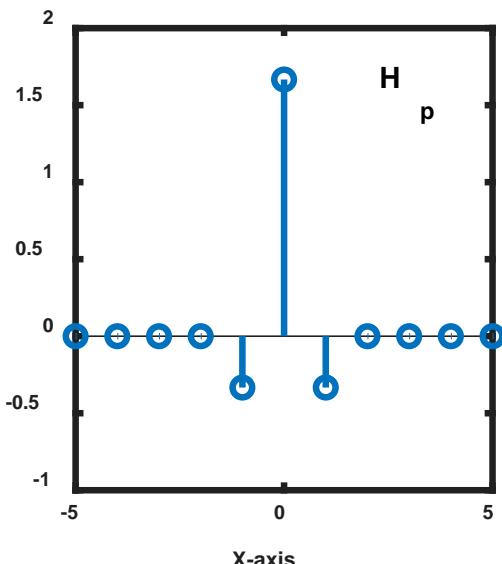
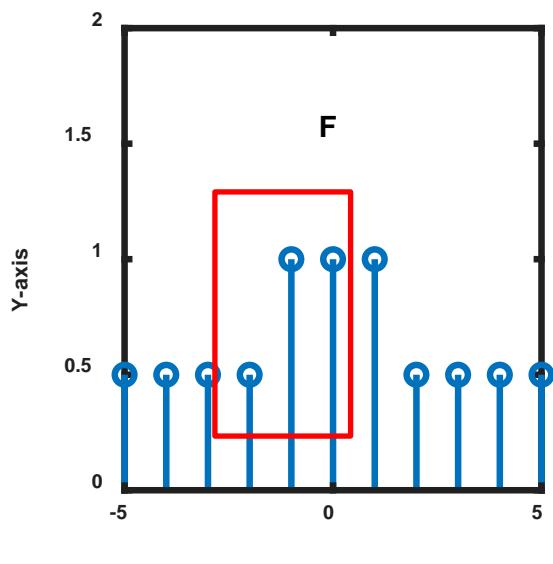
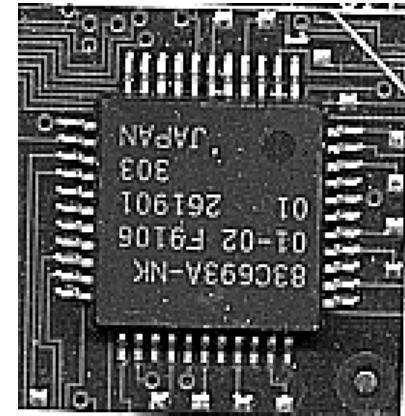
Sharpening



*

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

||



Making the difference larger

Q. What's difference?

$$F * H_p = G$$

How the Sharpening Filter Works

0	0	0
0	2	0
0	0	0

$$2H_i$$

0.11	0.11	0.11
0.11	0.11	0.11
0.11	0.11	0.11

$$H_r$$

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

$$H_p$$

$$F * H_p = F * (H_i + H_i - H_r) = F + \boxed{(F - F * H_r)}$$

0	0	0
0	$\alpha+1$	0
0	0	0

$$\alpha$$

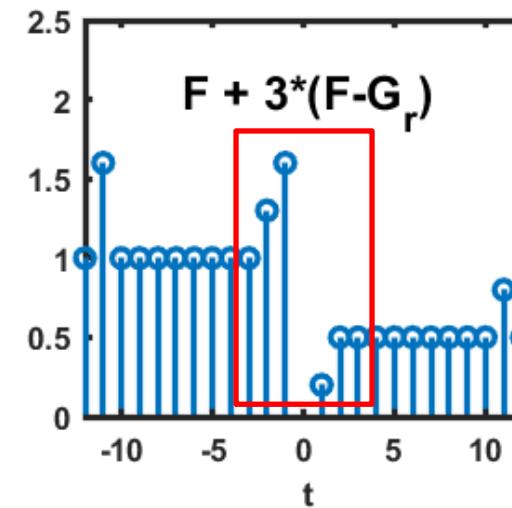
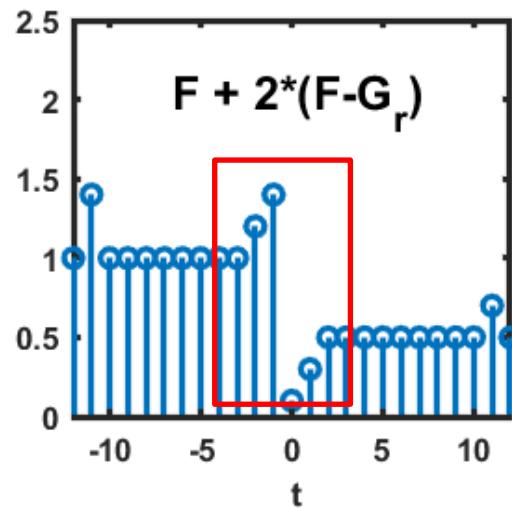
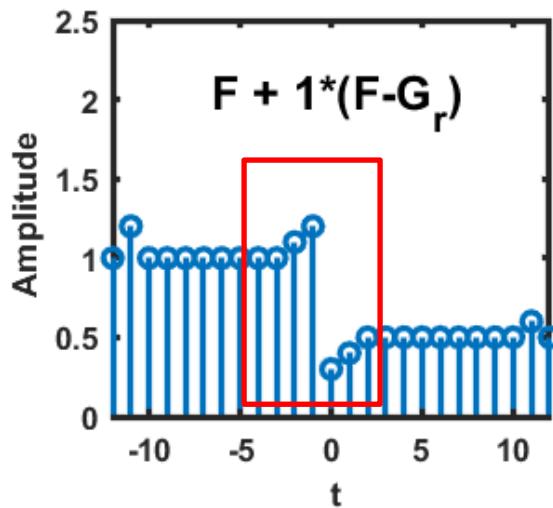
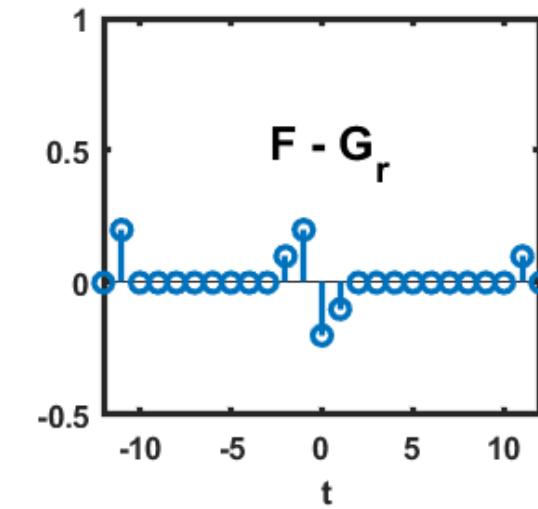
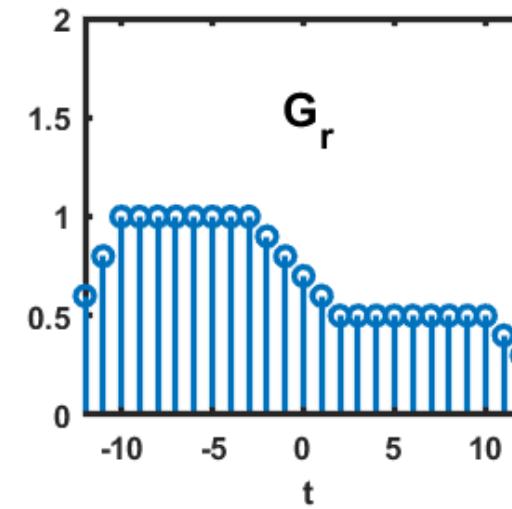
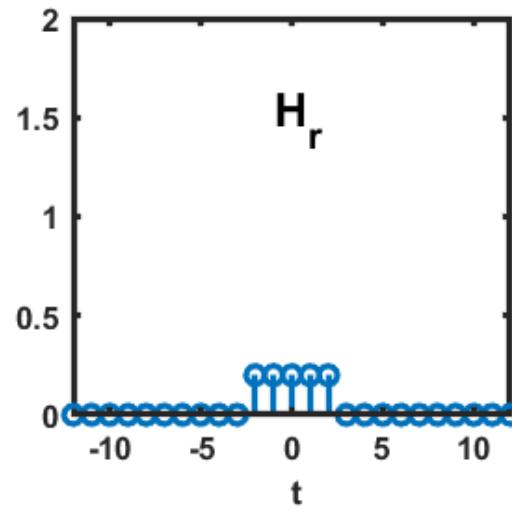
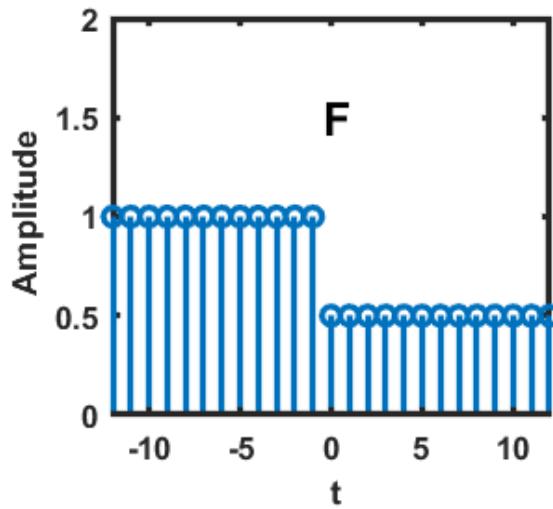
0.11	0.11	0.11
0.11	0.11	0.11
0.11	0.11	0.11

-0.11\alpha	-0.11\alpha	-0.11\alpha
-0.11\alpha	1+0.89\alpha	-0.11\alpha
-0.11\alpha	-0.11\alpha	-0.11\alpha

$$F + \alpha(F - F * H_r)$$

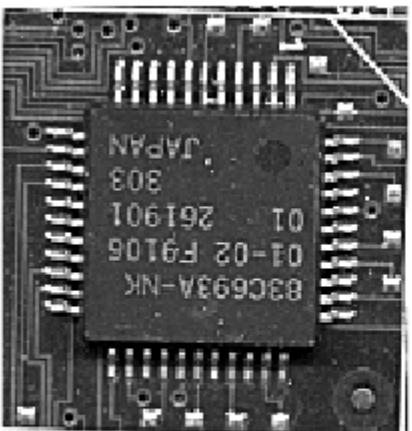
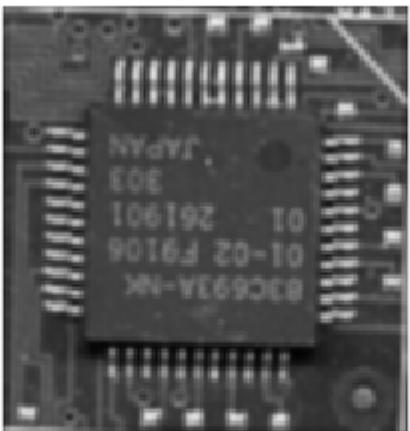
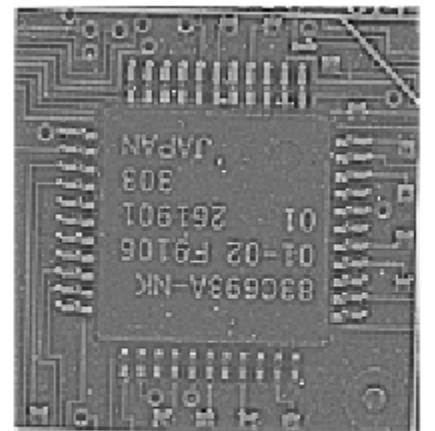
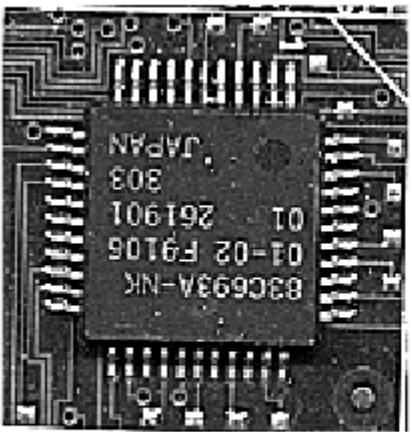
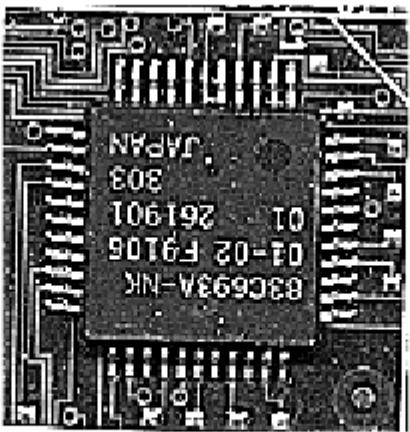
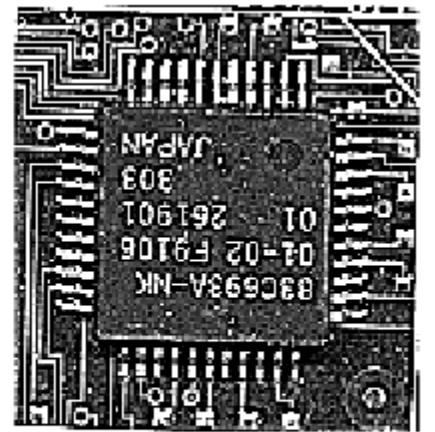
Adding this component in the original image

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



As α increases, the edges (discontinuities) are enhanced.

How the Sharpening Filter Works (Image Example)

 F  G_r  $F - G_r$  $F + (F - G_r)$  $F + 3(F - G_r)$  $F + 5(F - G_r)$

Gaussian Filter

```
clear; close all; clc;

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

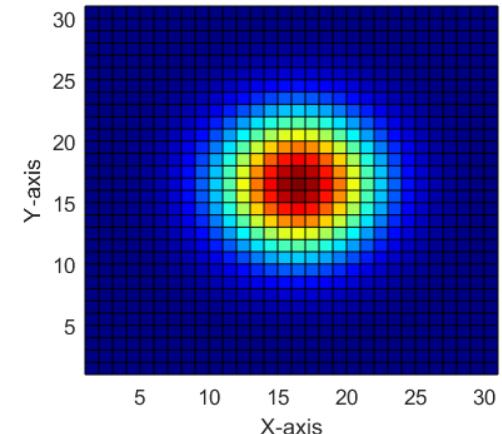
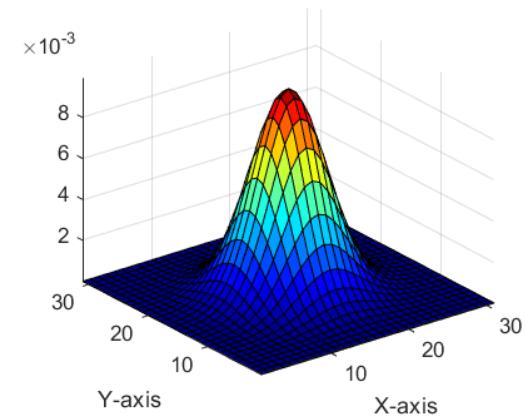
numSize = 3;
[x, y] = meshgrid(1:numSize);
x = x-(round(numSize/2));
y = y-(round(numSize/2));
sigma = 1;

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'); % from MATLAB built-in function
subplot(122); PlotMat(G_sigma,gca,'float'); % from equation

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

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

Effect of Gaussian Window Sizes

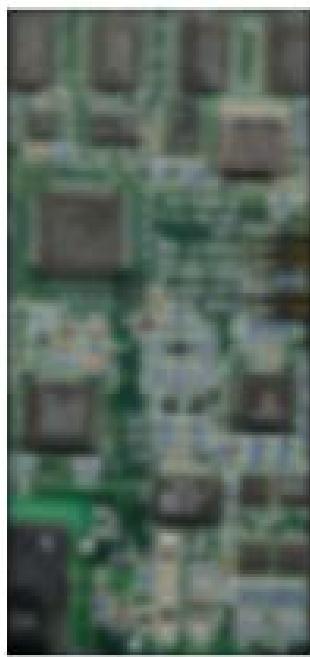


Original



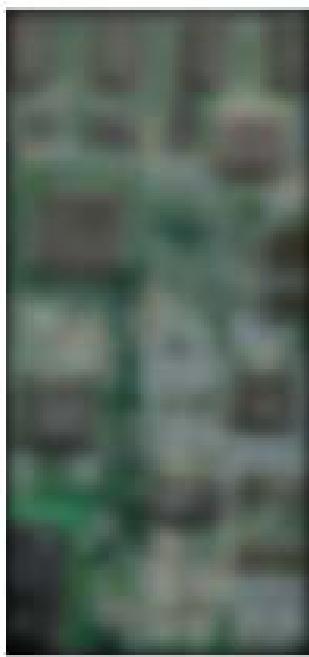
f_1

$$\sigma = 1$$



f_2

$$\sigma = 5$$



f_3

$$\sigma = 10$$



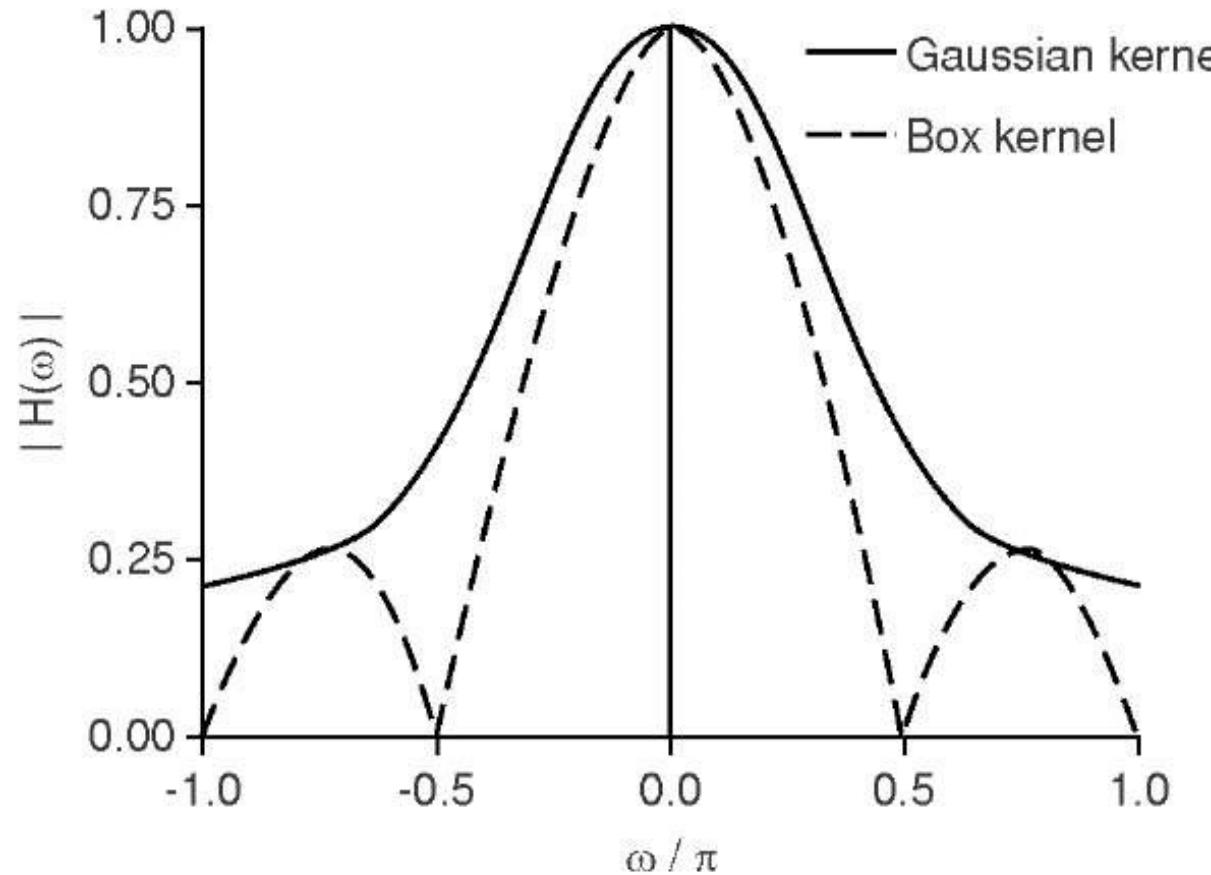
f_4

$$\sigma = 30$$

```
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

Supplement

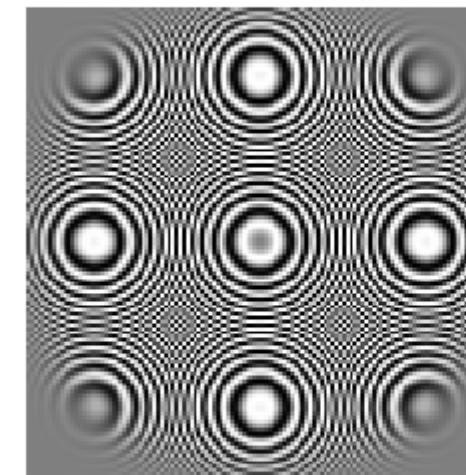
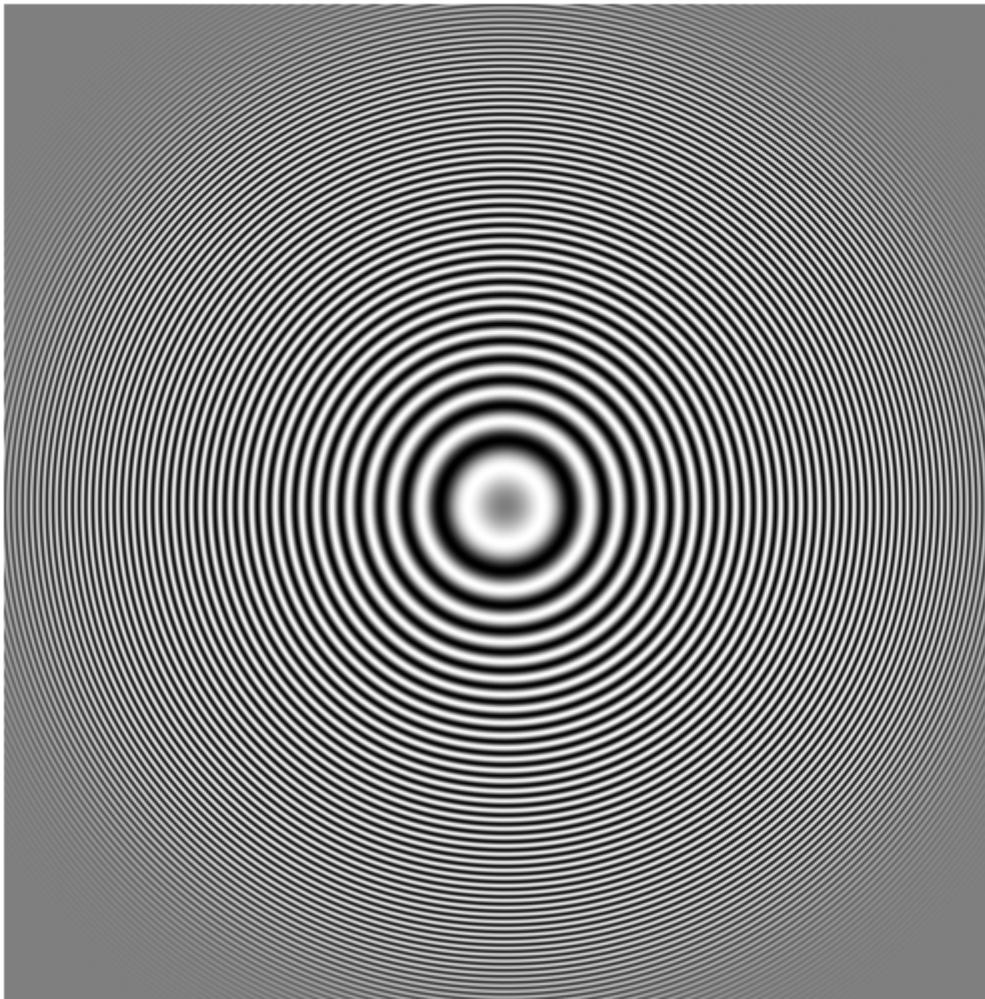


Gaussian

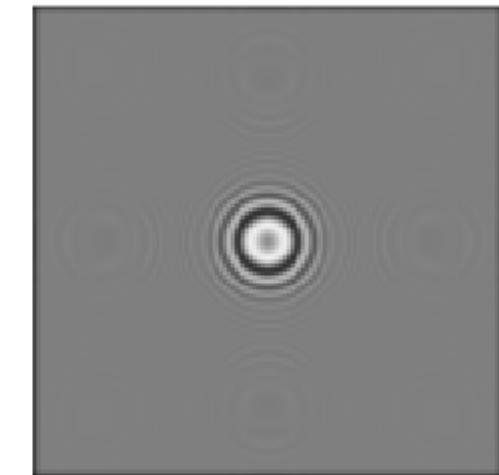


Box

original image



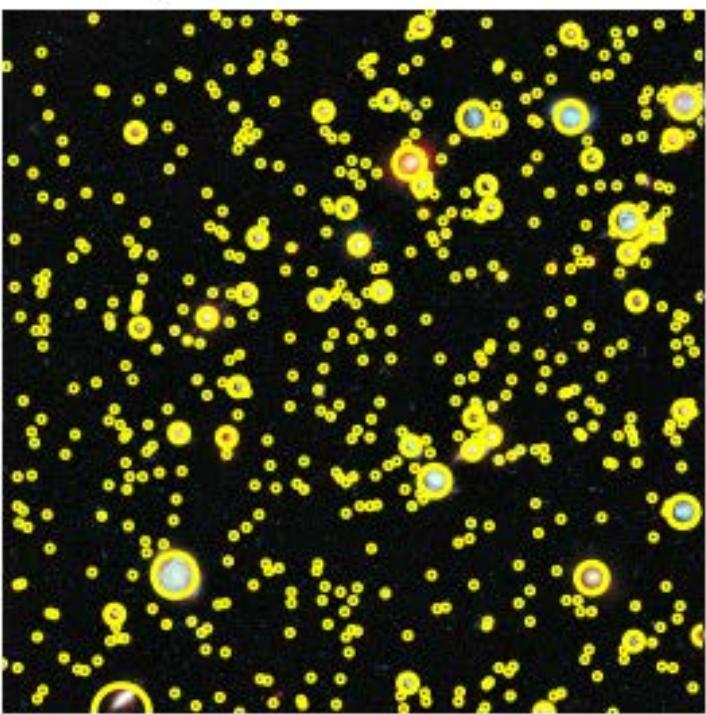
Resizing without
applying gaussian
filter



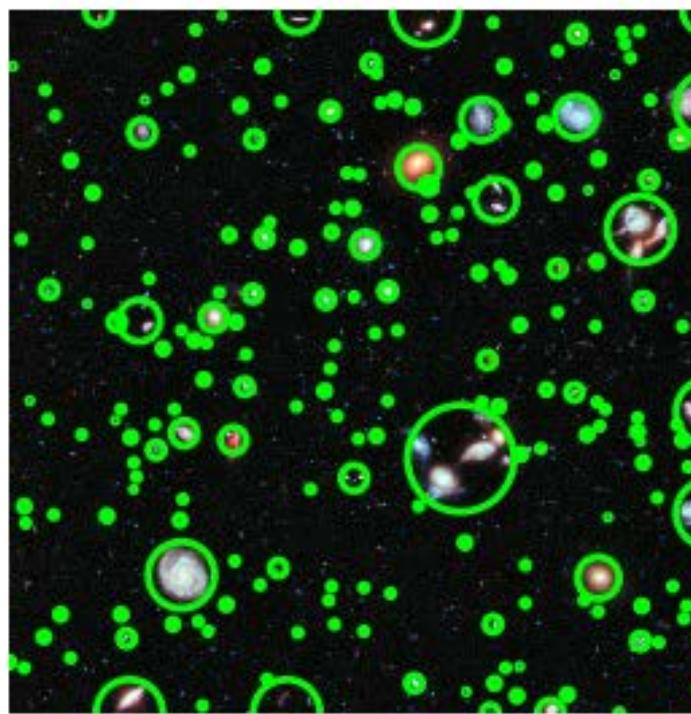
Resizing after
applying gaussian
filter

Applications: Blob Detection (for Feature Detection)

Laplacian of Gaussian



Difference of Gaussian



Determinant of Hessian

