Image Analysis

Motivation

Computer vision

Input: digital images

Output: information about the world

Image Analysis

Input is a regular array of discrete samples of a 2D continuous function representing color

e.g., Color at (x,y)

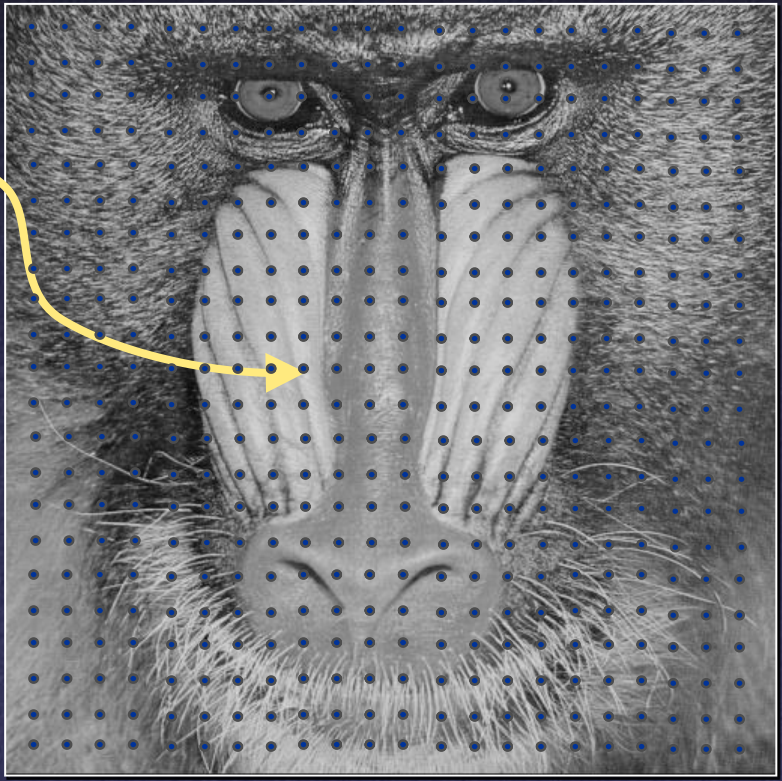
Color image

Output is info about structure of image

Image Analysis

For now, let's consider only gray-level images

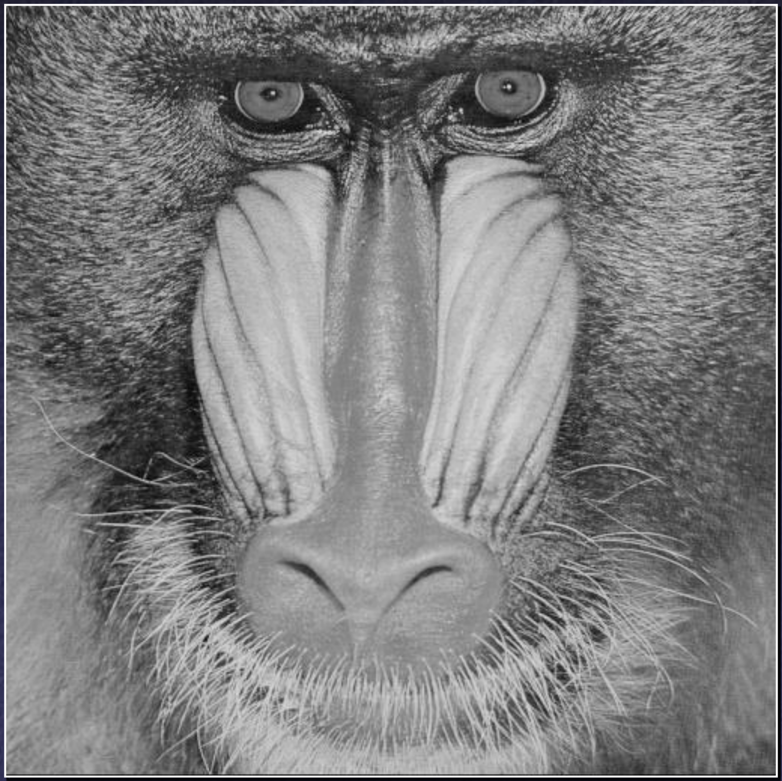
e.g., Luminance at (x,y)



Gray-level image

Image Analysis

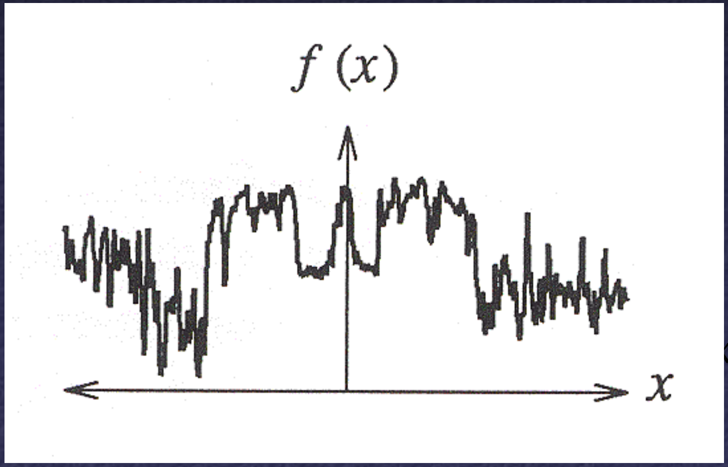
For now, let's ignore the discrete sampling

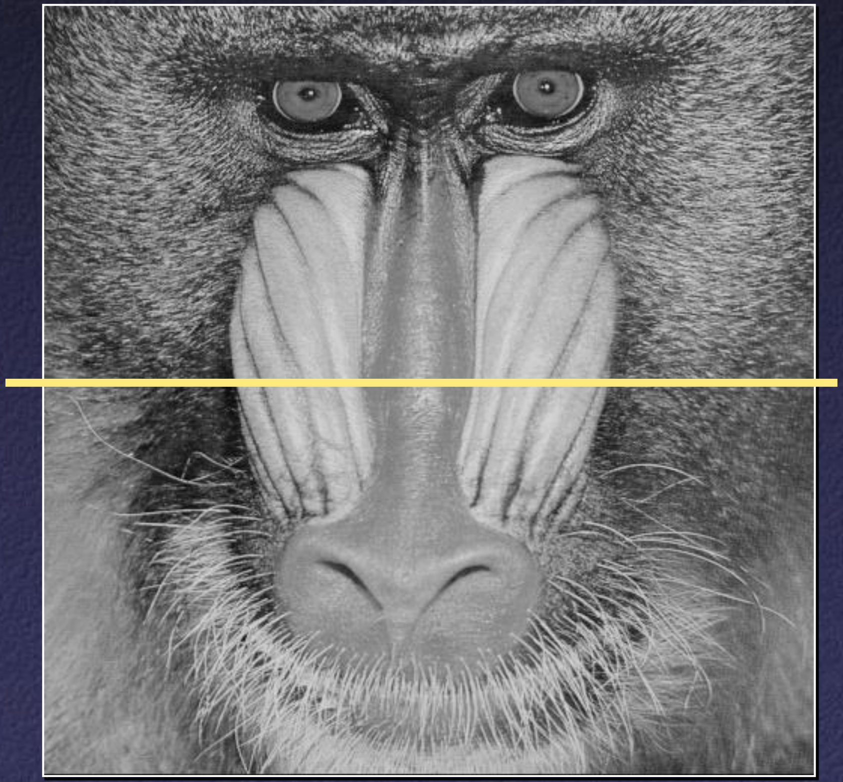


Gray-level function

Image Analysis

For now, let's consider only one horizontal scanline





Gray-level function

Image Analysis

How do we analyze 1D continuous functions?

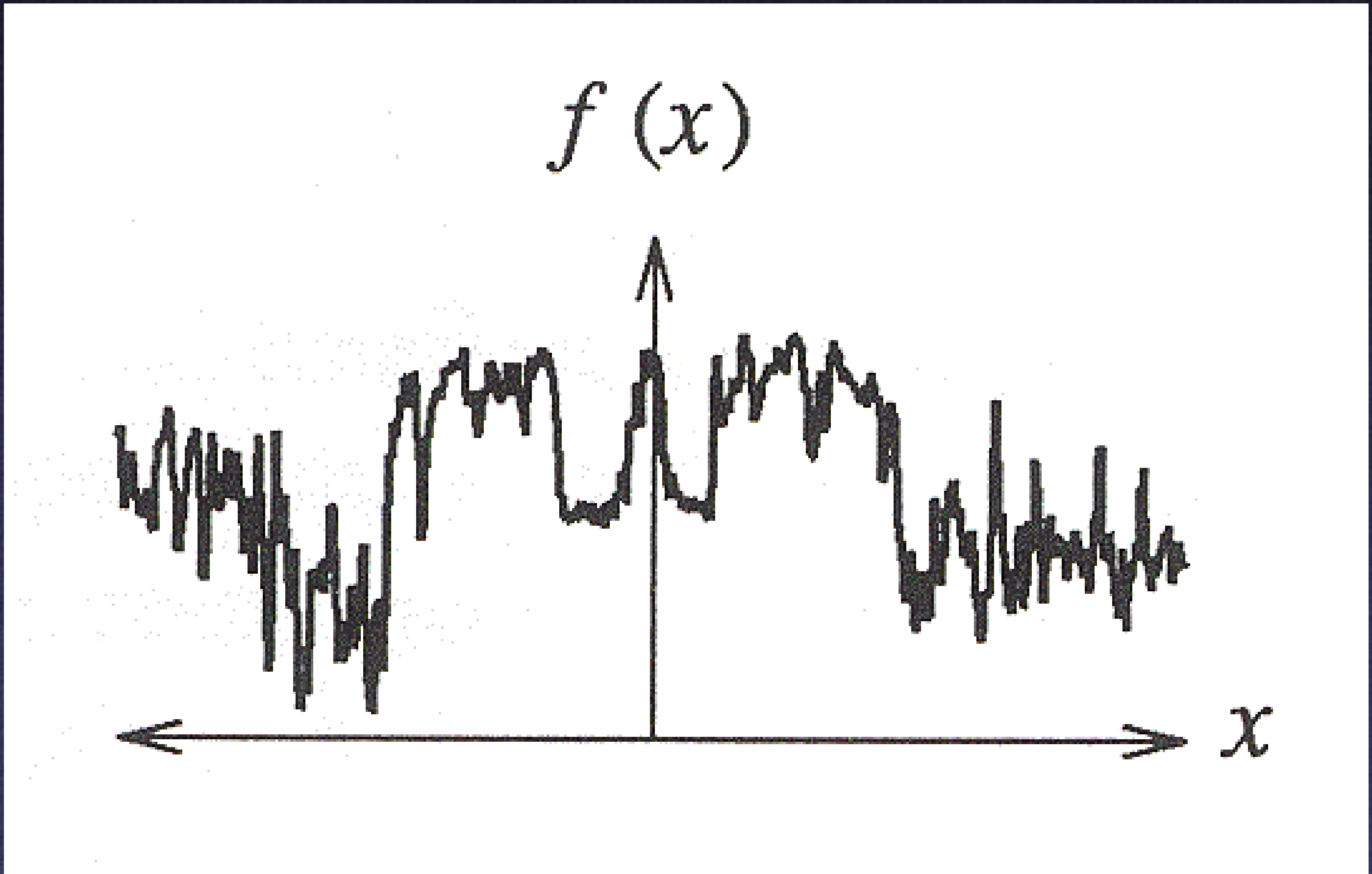
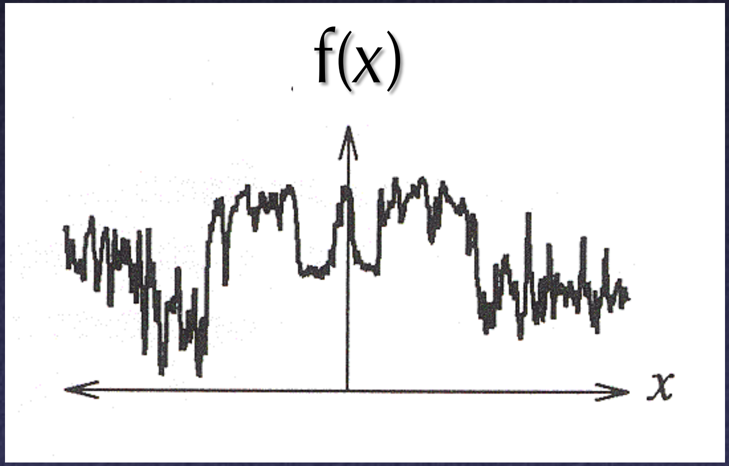


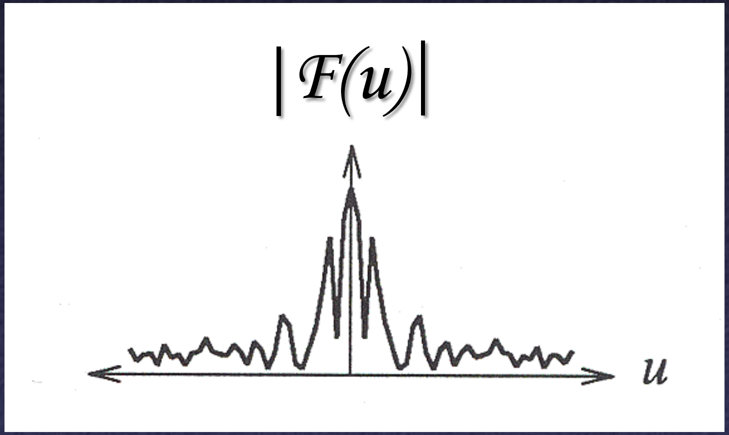
Image Analysis

How do we analyze 1D continuous functions?

- One useful tool is frequency analysis



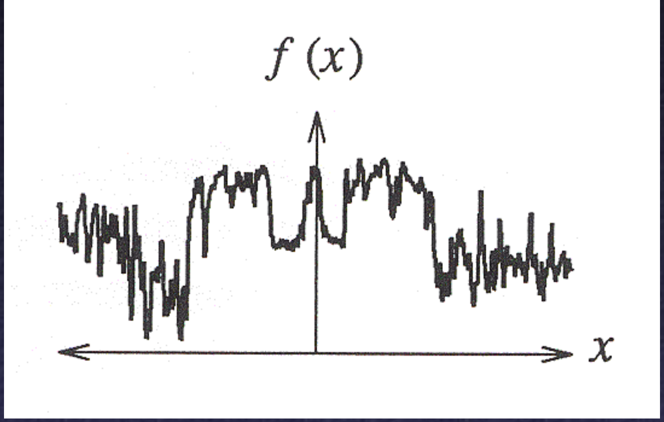
Spatial domain

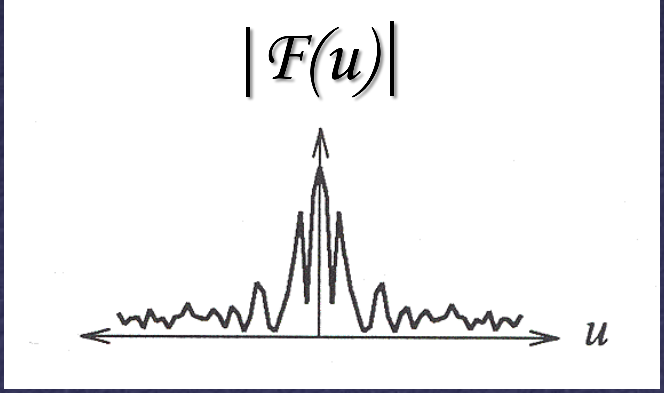


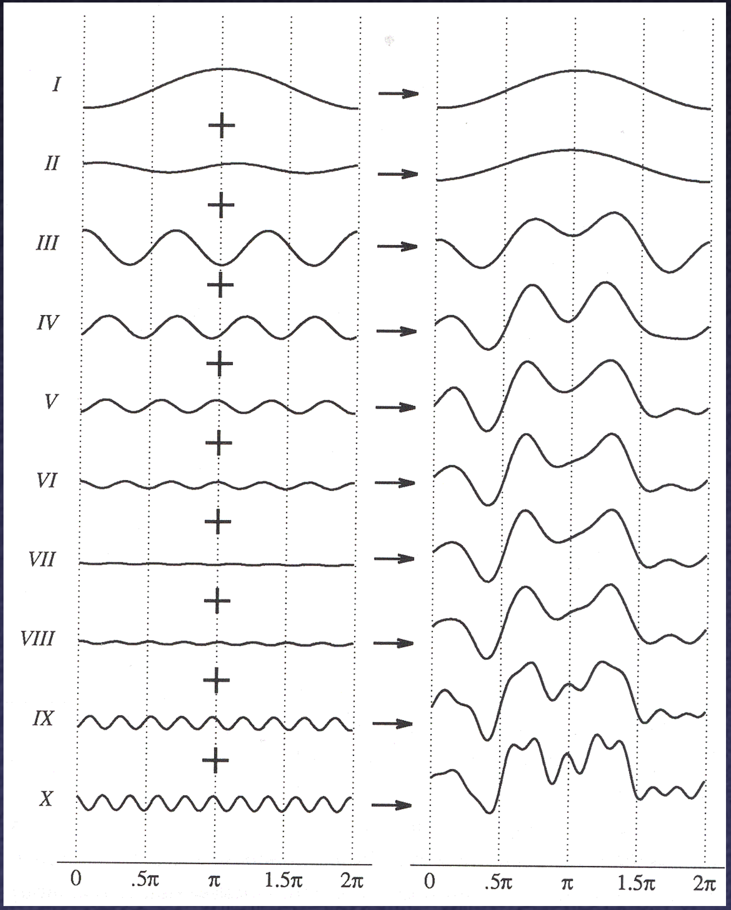
Frequency domain

Frequency Analysis

Any f(x) can be written as a sum of periodic functions







Frequency Analysis

Fourier transform of function *f* is

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-i 2\pi x u} dx$$

F(u) is a function of frequency u describing how much of each frequency *f* contains

Frequency Analysis

Fourier transform has real and imaginary parts:

$$
\begin{aligned}
\text{Magnitude:} \quad |F| &= \left[\Re(F)^2 + \Im(F)^2\right]^{1/2} \\
\text{Phase:} \quad \phi(F) &= \tan^{-1} \frac{\Im(F)}{\Re(F)}
\end{aligned}
$$

Real part	How much of a cosine of that frequency you need
Imaginary part	How much of a sine of that frequency you need
Magnitude	Amplitude of combined cosine and sine
Phase	Relative proportions of sine and cosine

Frequency Analysis

How does this work for 2D functions?

$$F(u,v) = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} f(x,y) e^{-j2\pi ux} dx \right] e^{-j2\pi vy} dy.$$

$$f(x,y) = \int_{-\infty}^{\infty} \left[\int_{-\infty}^{\infty} F(u,v) e^{j2\pi ux} du \right] e^{j2\pi vy} dv.$$

Frequency Analysis

The Fourier Transform is separable:

$$f(x,y) = f(x)g(y) \xrightarrow{\mathcal{F}} F(u,v) = F(u)G(v)$$

Proof:

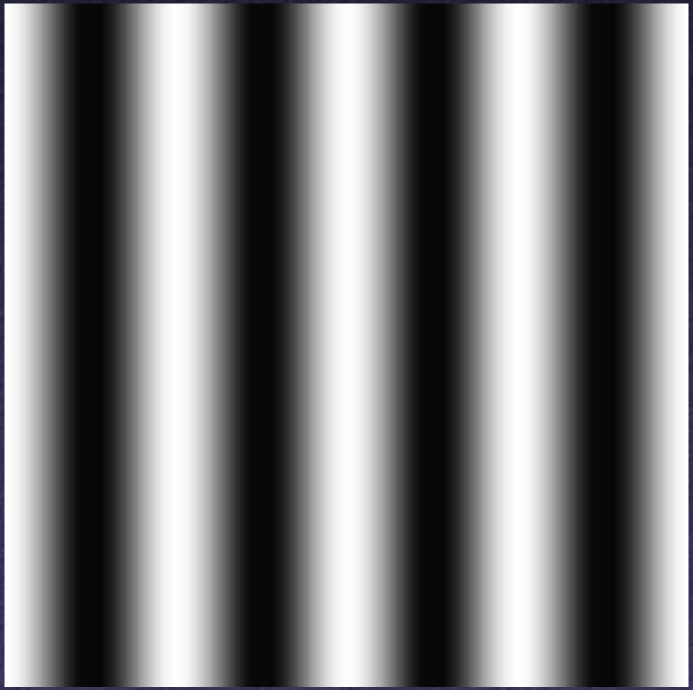
$$\begin{aligned} & F(u,v) \\ = & \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)e^{-j2\pi(ux+vy)}dx\,dy \\ = & \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x)g(y)e^{-j2\pi ux}e^{-j2\pi vy}\,dx\,dy \\ = & \int_{-\infty}^{\infty} f(x)e^{-j2\pi ux}\,dx \int_{-\infty}^{\infty} g(y)e^{-j2\pi vy}dy \\ = & F(u)\,G(v) \end{aligned}$$

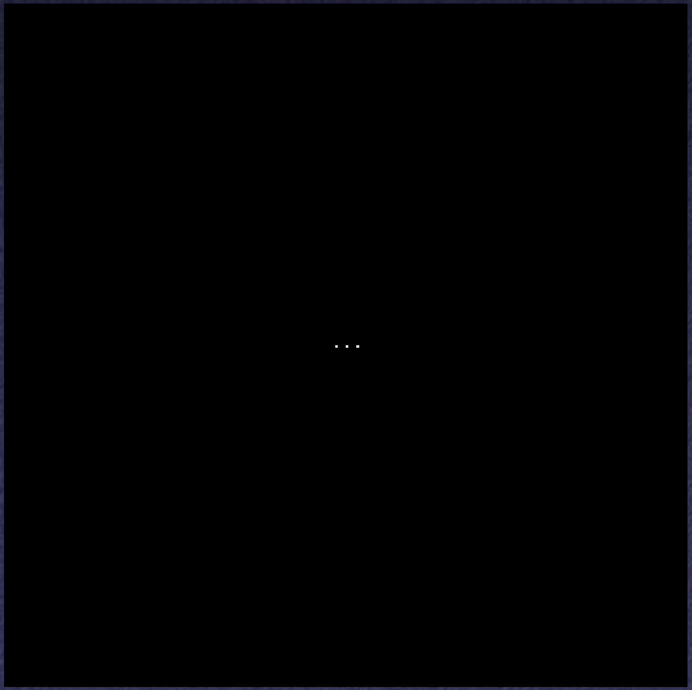
Frequency Analysis

Examples:

$f(x,y)$

$|F(u,v)|$



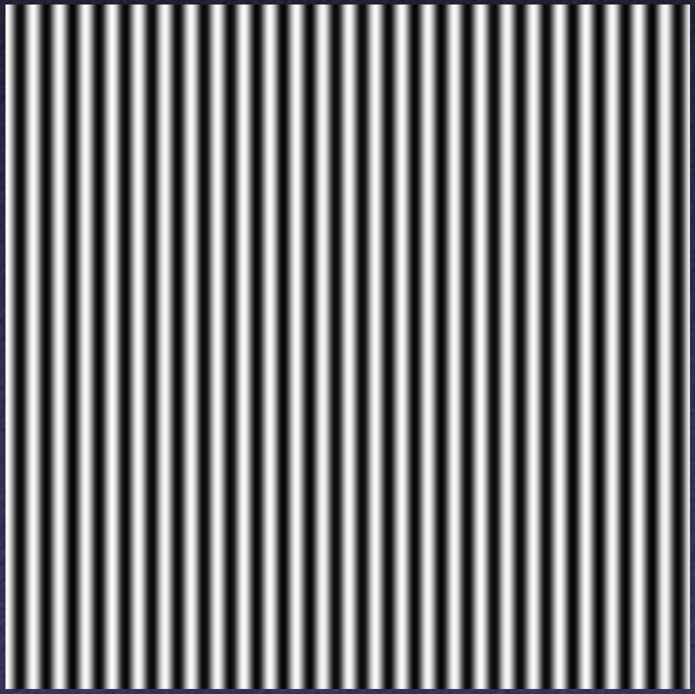


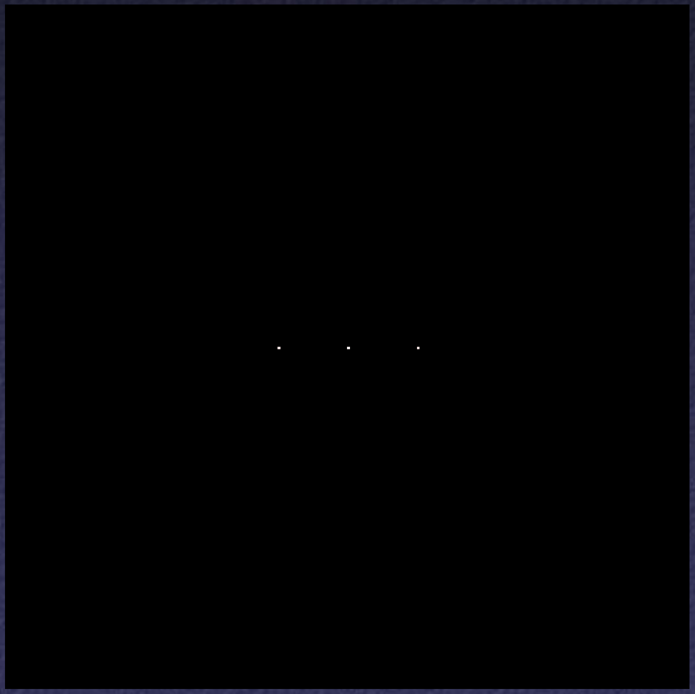
Frequency Analysis

Examples:

f(x,y)

|F(u,v)|

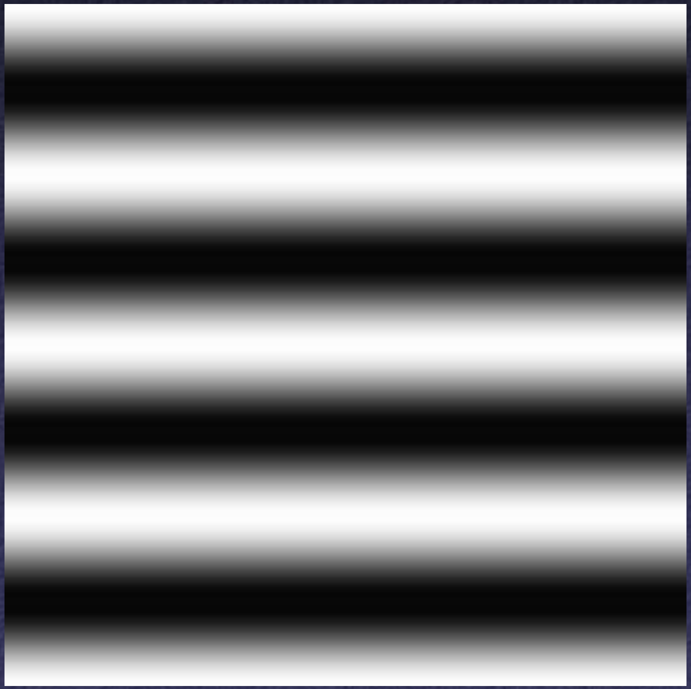




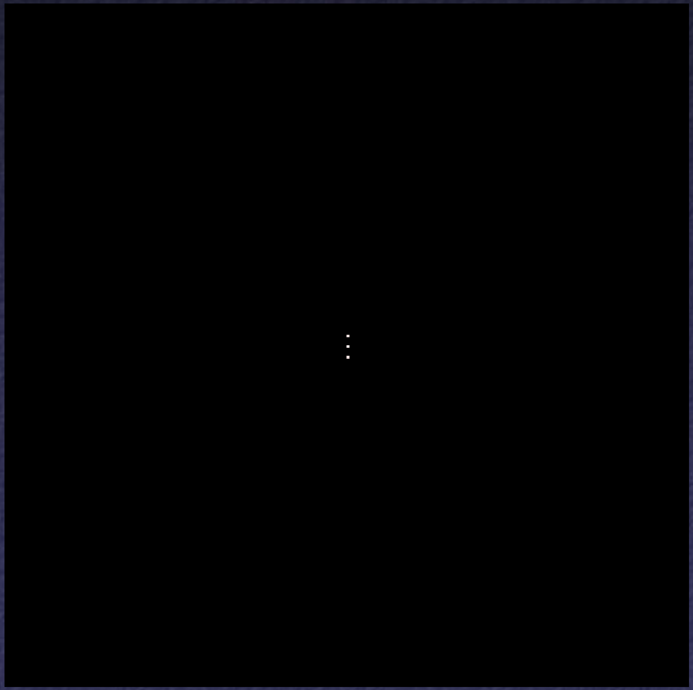
Frequency Analysis

Examples:

f(x,y)



|F(u,v)|



Frequency Analysis

Examples:

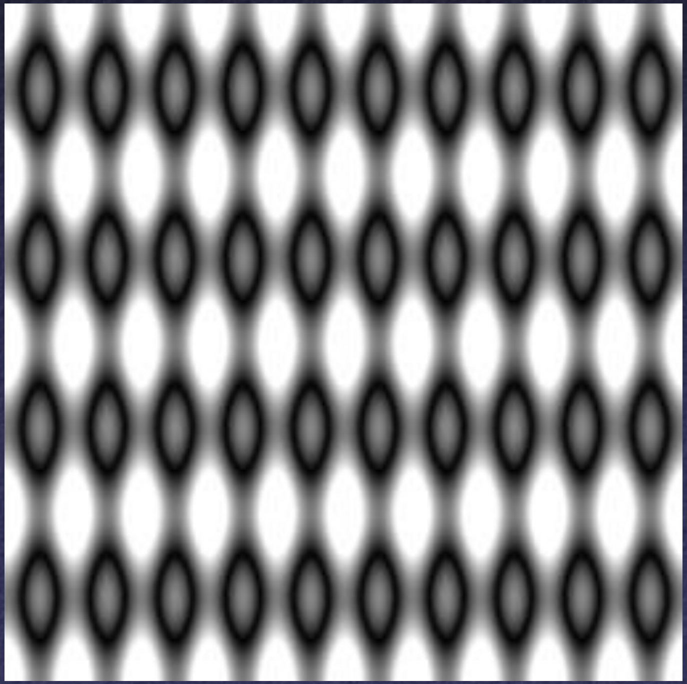
$f(x,y)$

$|F(u,v)|$

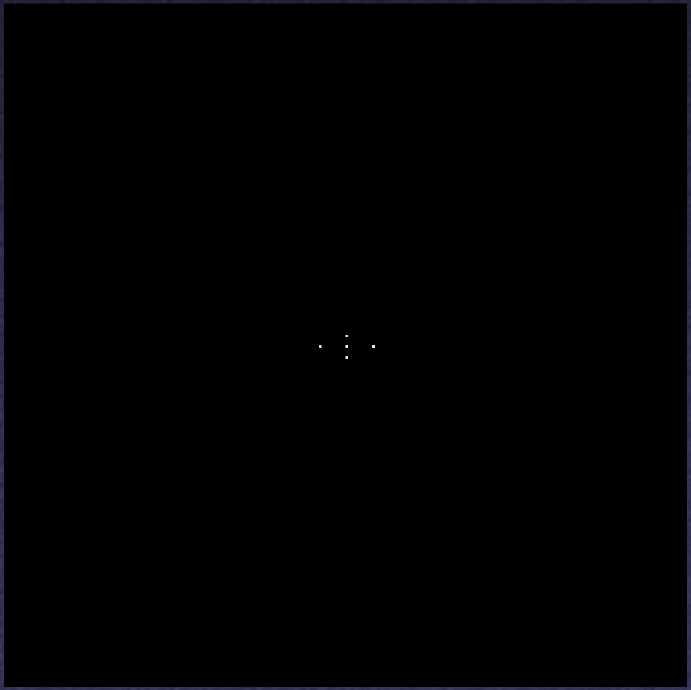
Frequency Analysis

Examples:

f(x,y)



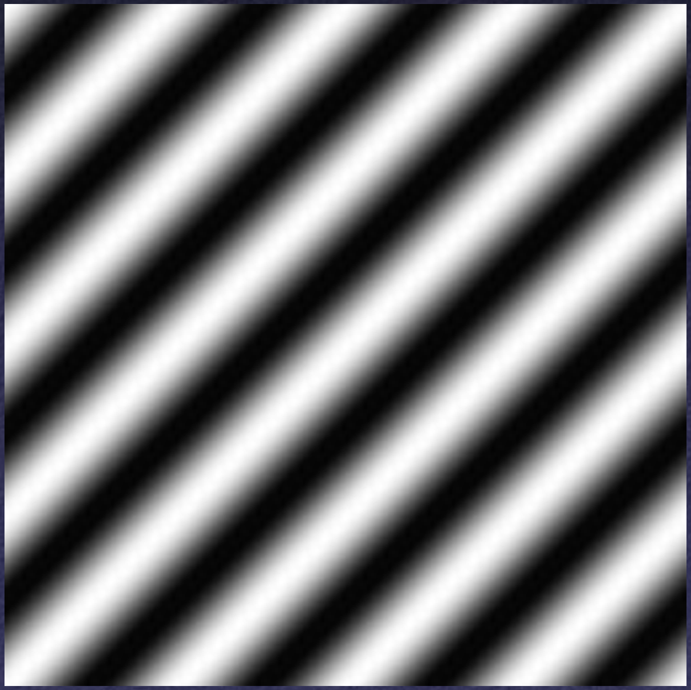
|F(u,v)|



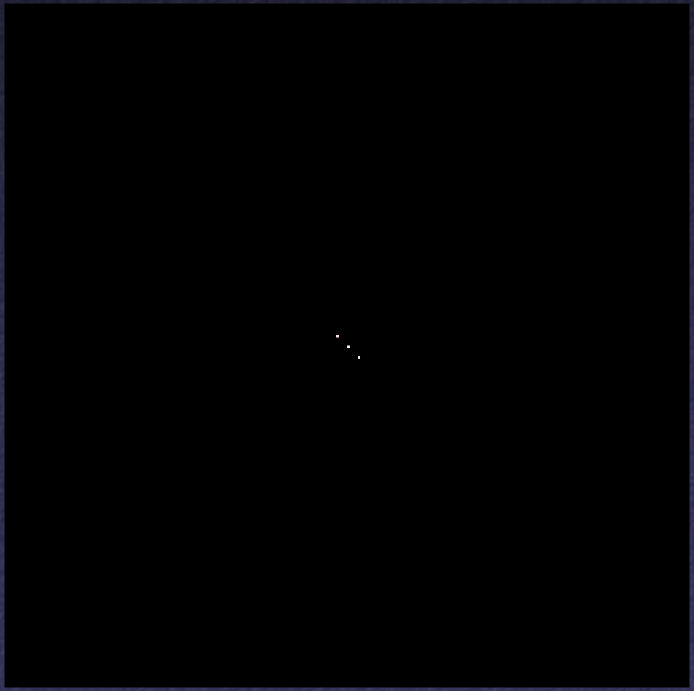
Frequency Analysis

Examples:

$f(x,y)$



$|F(u,v)|$

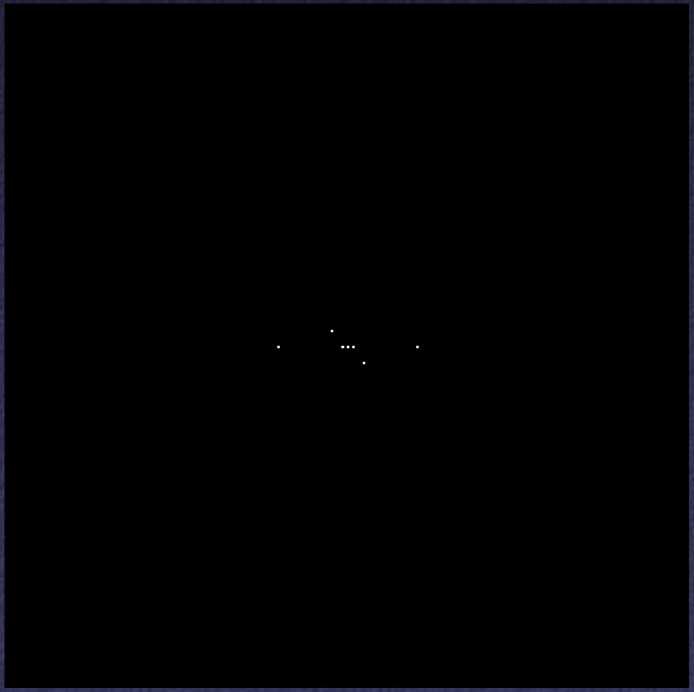


Frequency Analysis

Examples:

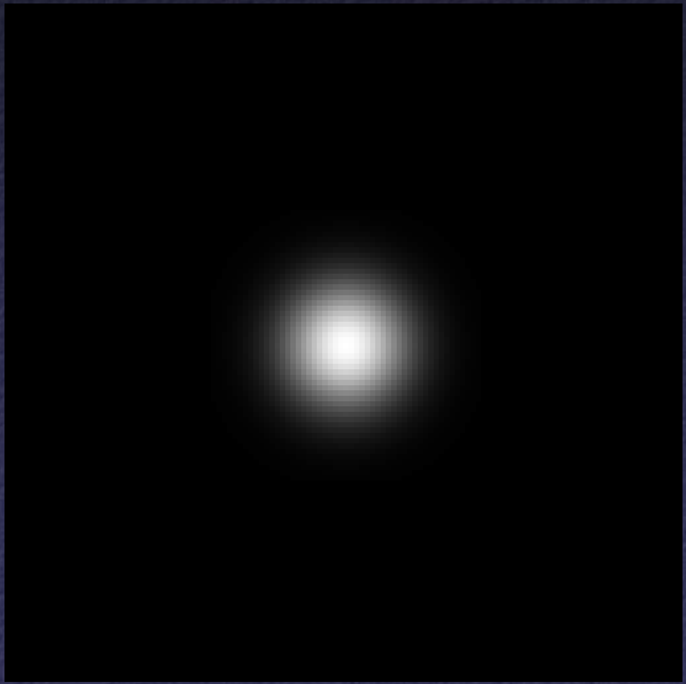
f(x,y)

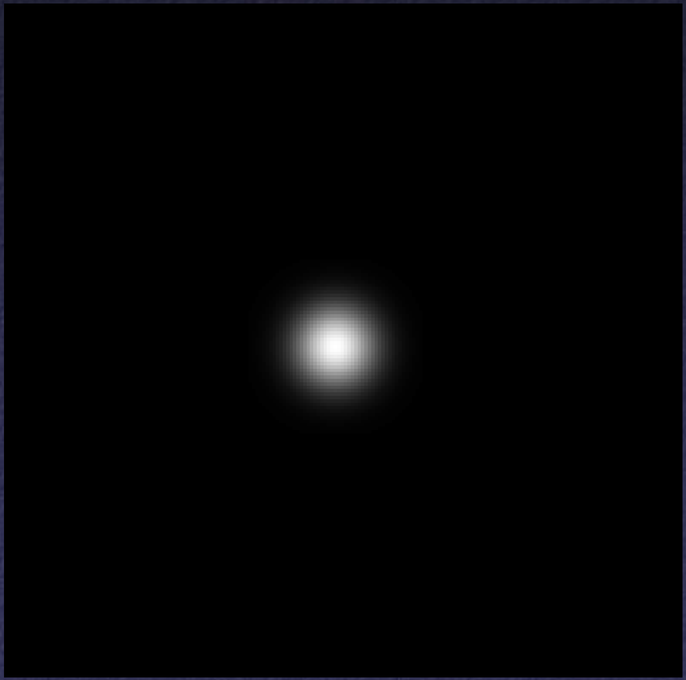
|F(u,v)|



Frequency Analysis

Examples: Gaussian





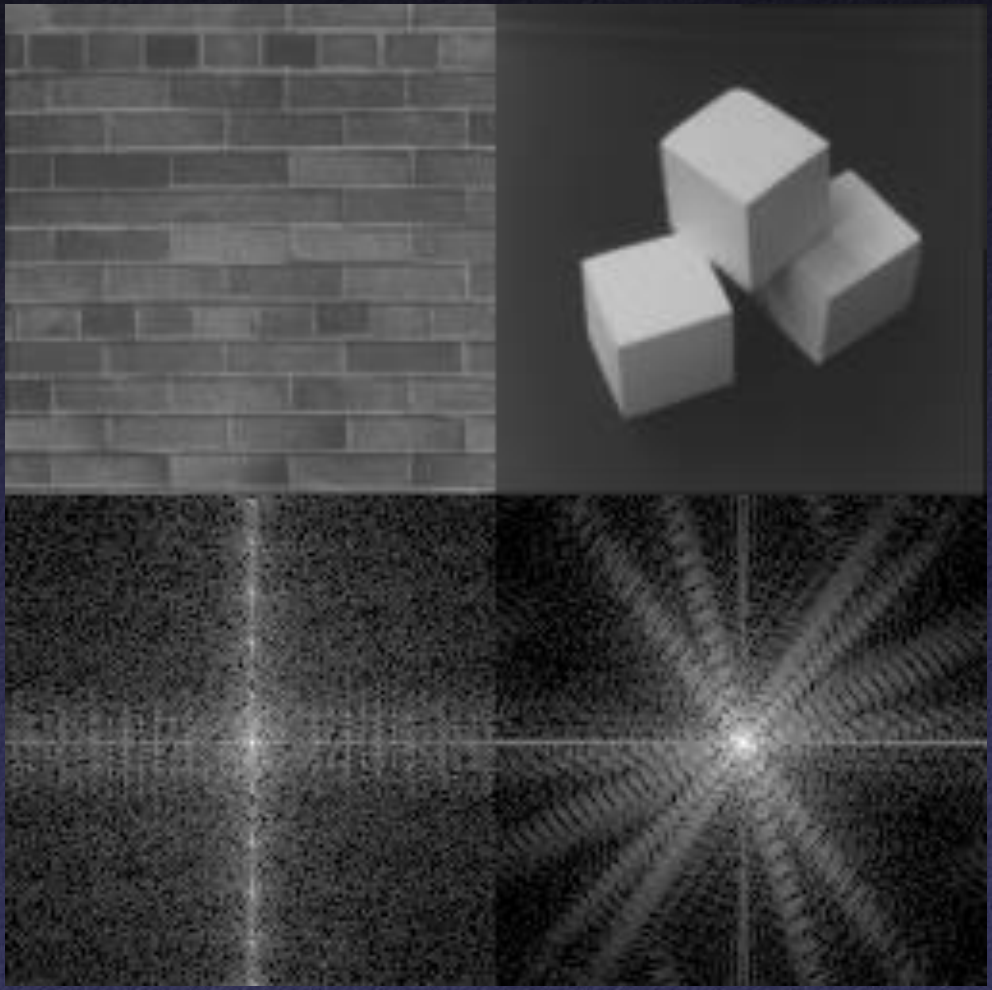
$$G_2(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Frequency Analysis

Examples:

f(x,y)

|F(u,v)|

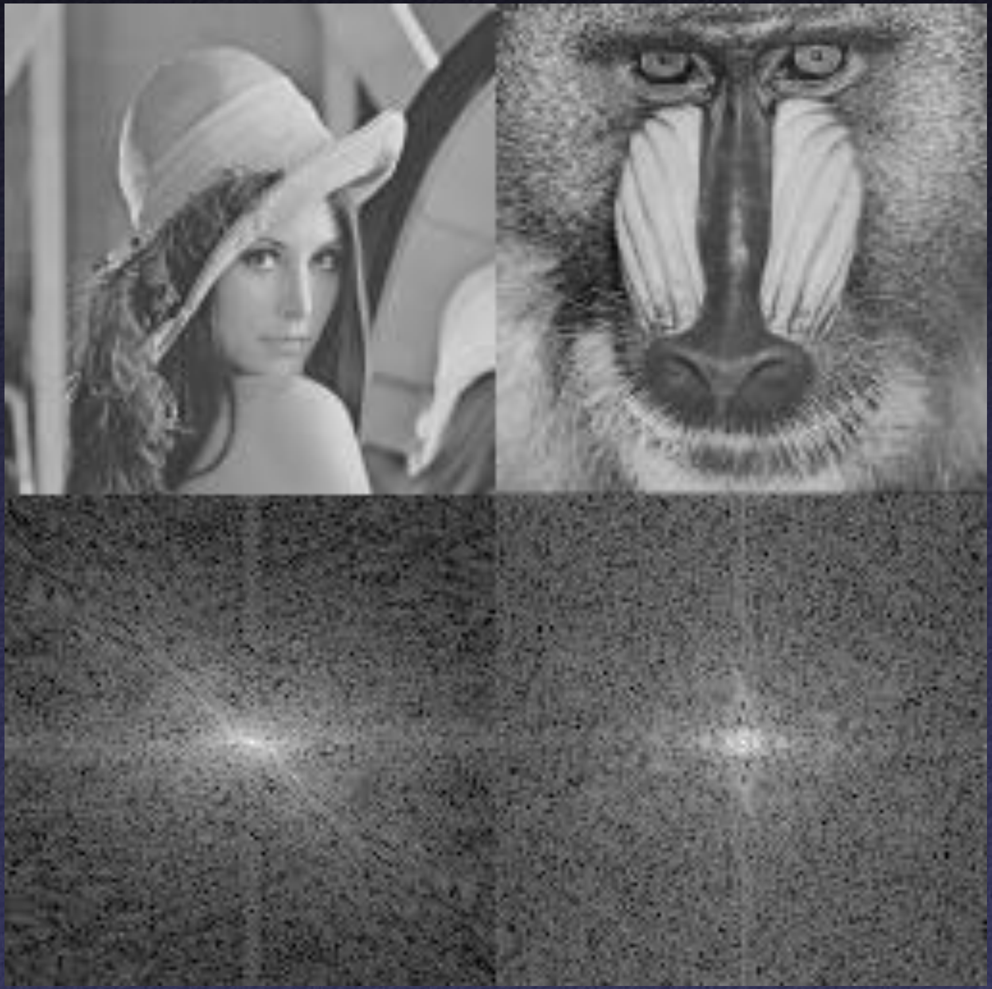


Frequency Analysis

Examples:

f(x,y)

|F(u,v)|



Frequency Analysis

The Fourier transform has an inverse:

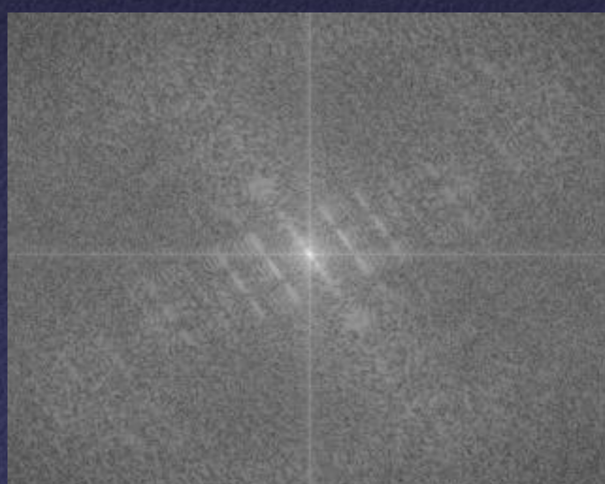
$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi xu} dx$$

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{+i2\pi ux} du$$

Application 1: Reducing Noise

f(x,y)

|F(u,v)|



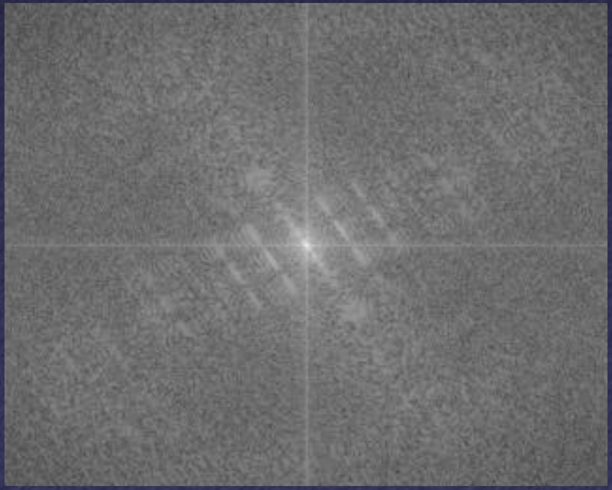
Noise is unwanted (random) energy in high frequencies

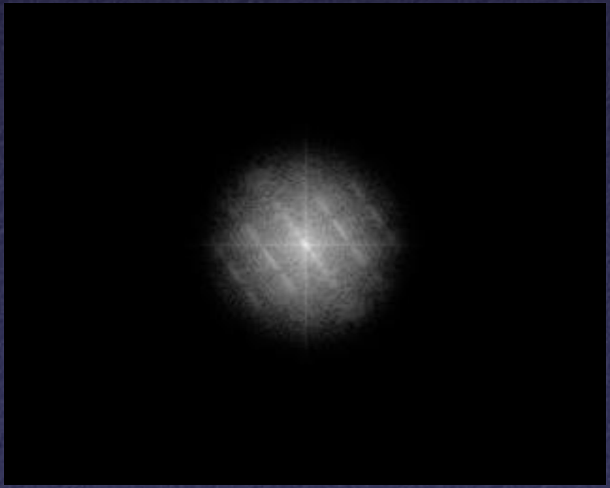
Zoomed

Application 1: Reducing Noise

f(x,y)

|F(u,v)|



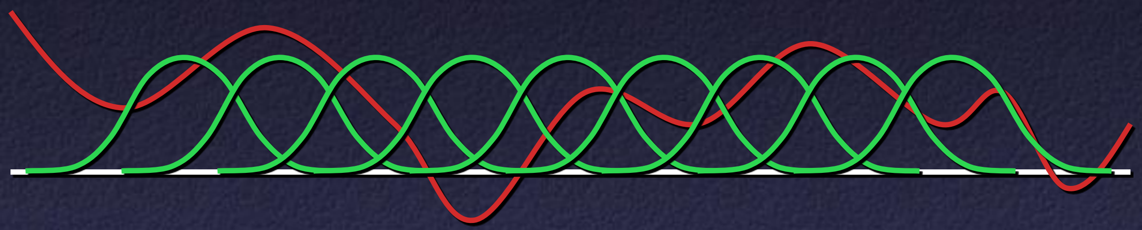


Original

High frequencies removed

Application 1: Reducing Noise

Can reduce noise by convolving image with a Gaussian filter



$$f(x) * g(x) = \int_{t=-\infty}^{\infty} f(t)\, g(x-t)\, dt$$

Gaussian Filters

What is a Gaussian filter?

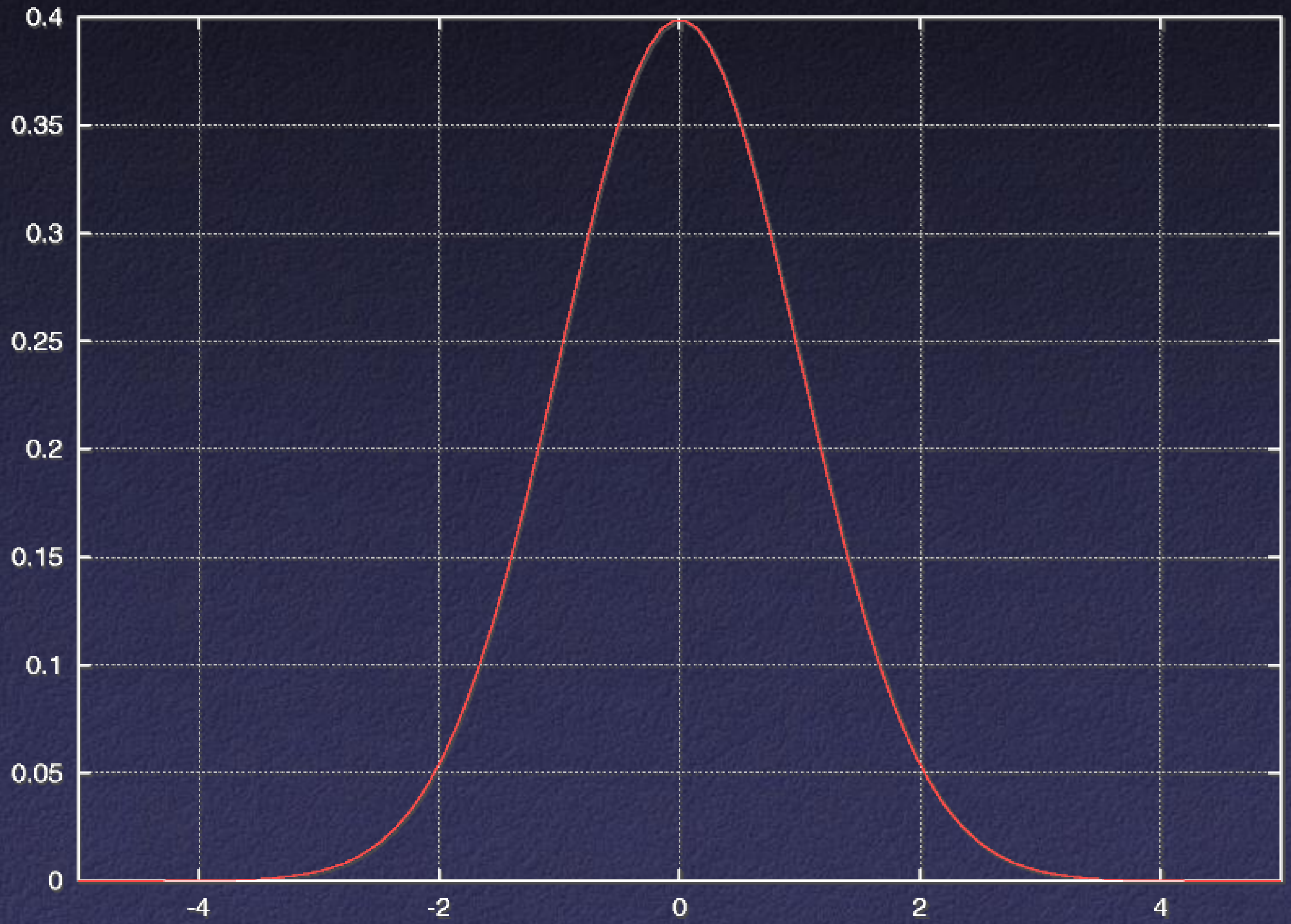
- One-dimensional Gaussian

$$G_1(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}}$$

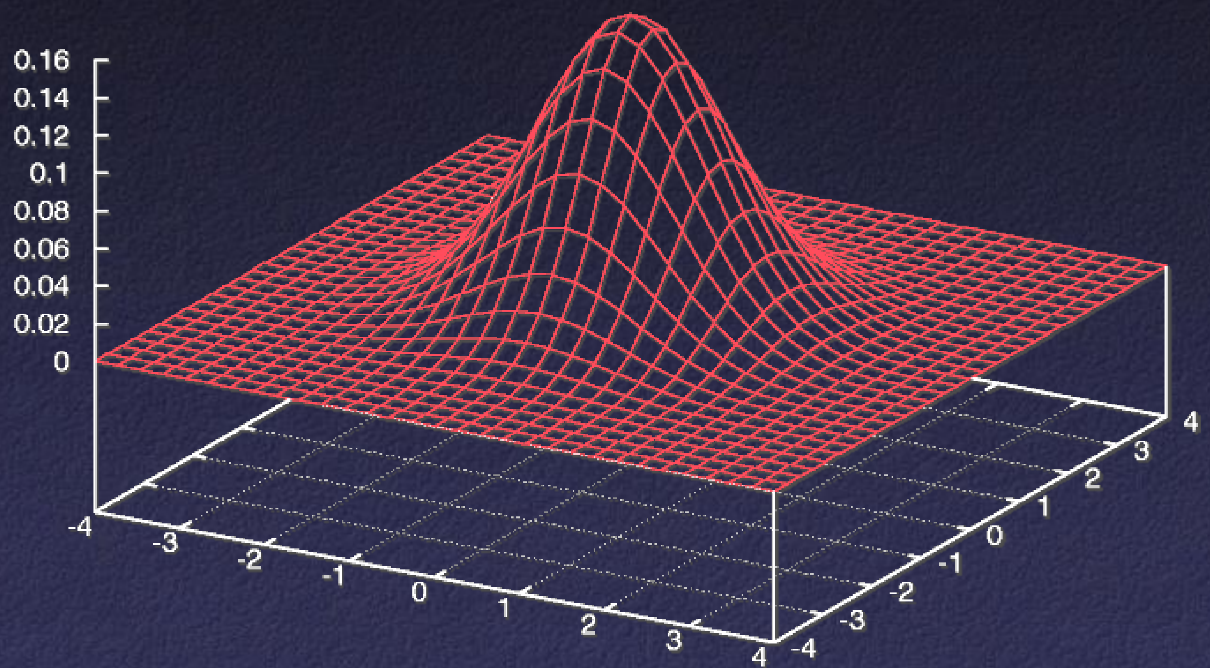
- Two-dimensional Gaussian

$$G_2(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Gaussian Filters

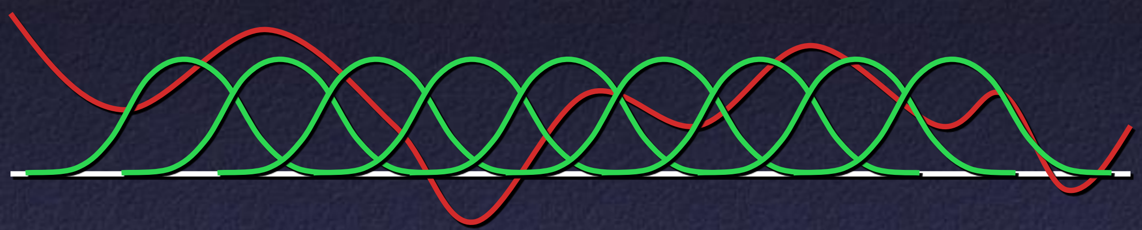


Gaussian Filters



Convolution

How do we convolve an image with a filter?

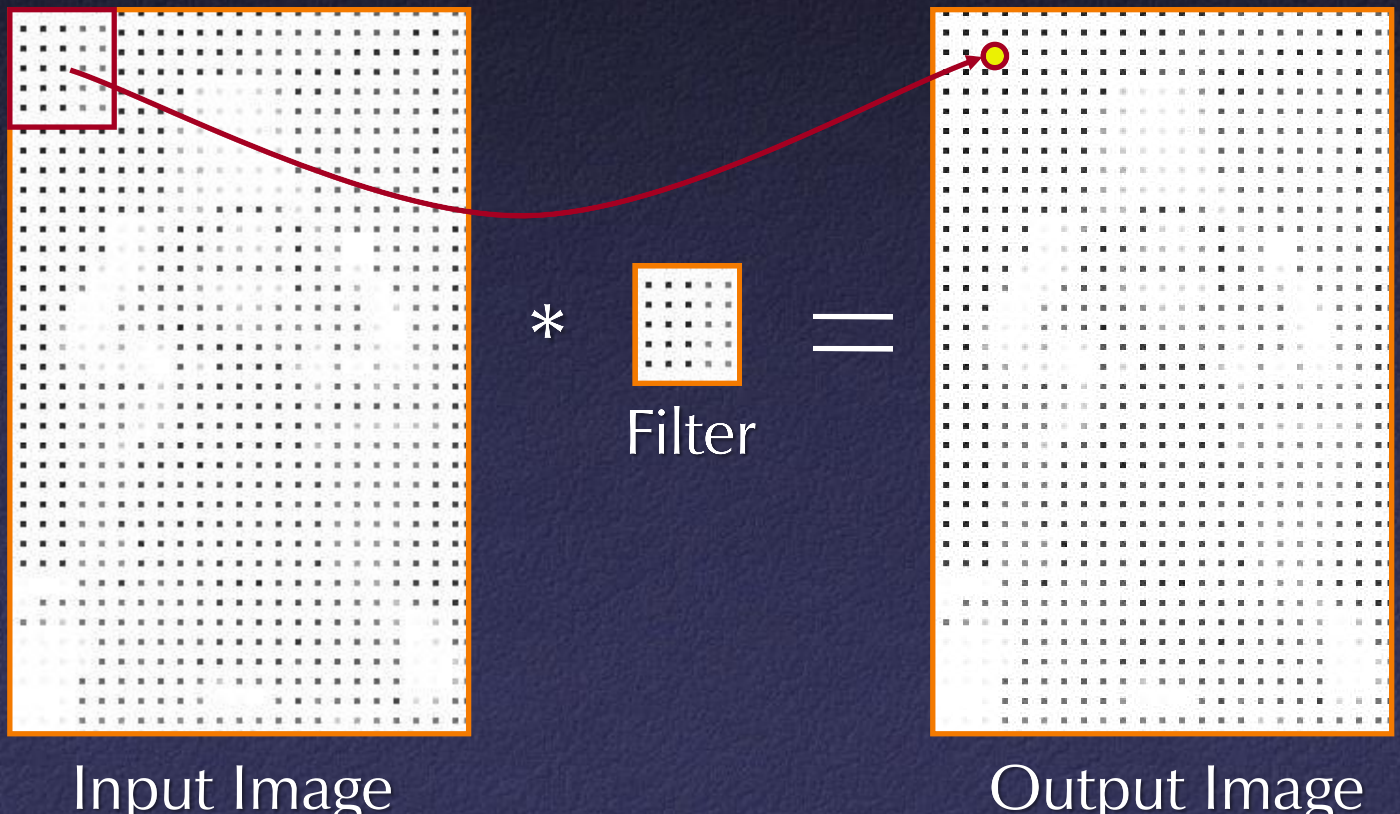


$$f(x) * g(x) = \int_{t=-\infty}^{\infty} f(t)\, g(x-t)\, dt$$

Convolution

Discrete convolution

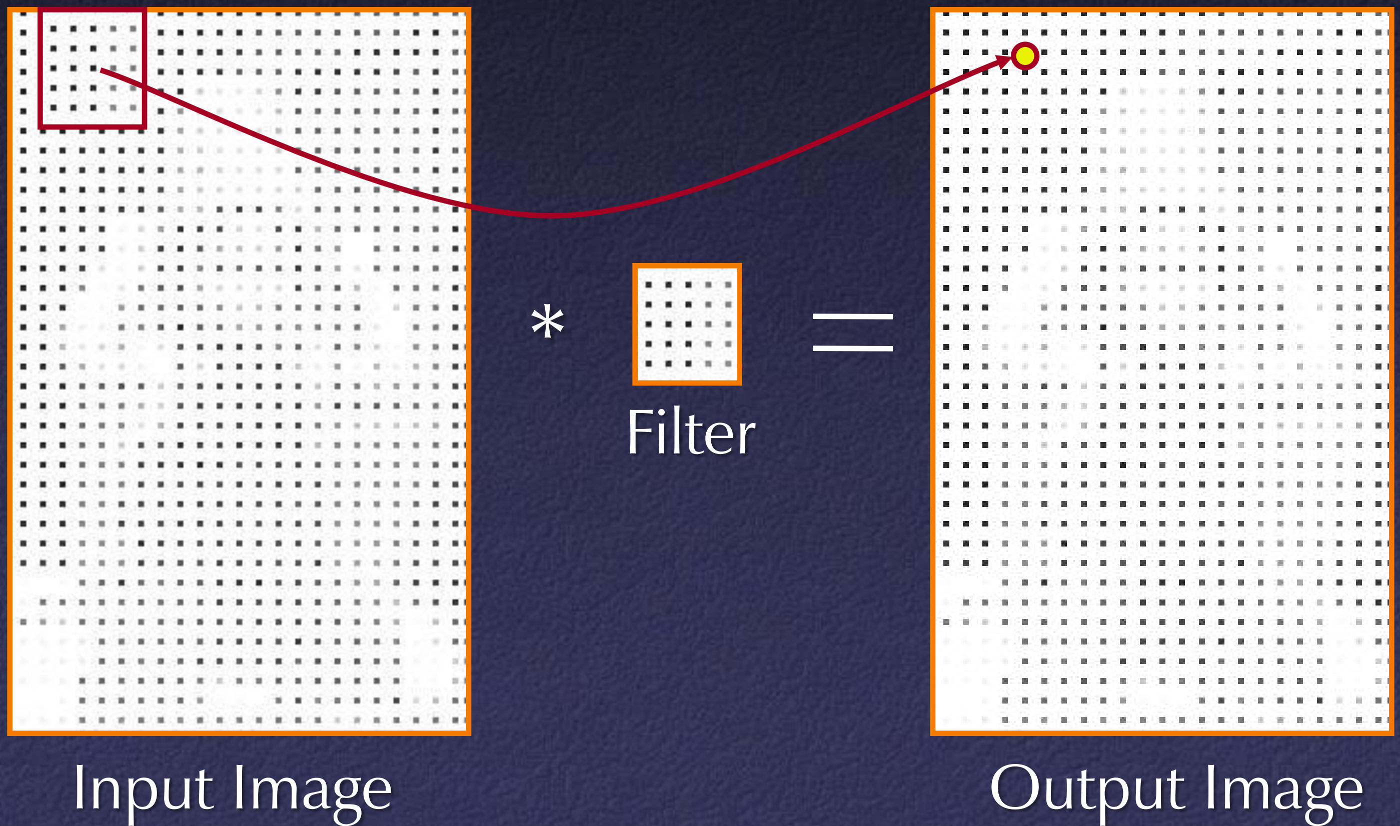
- Each output pixel is a linear combination of input pixels in neighborhood with weights prescribed by filter



Convolution

Discrete convolution

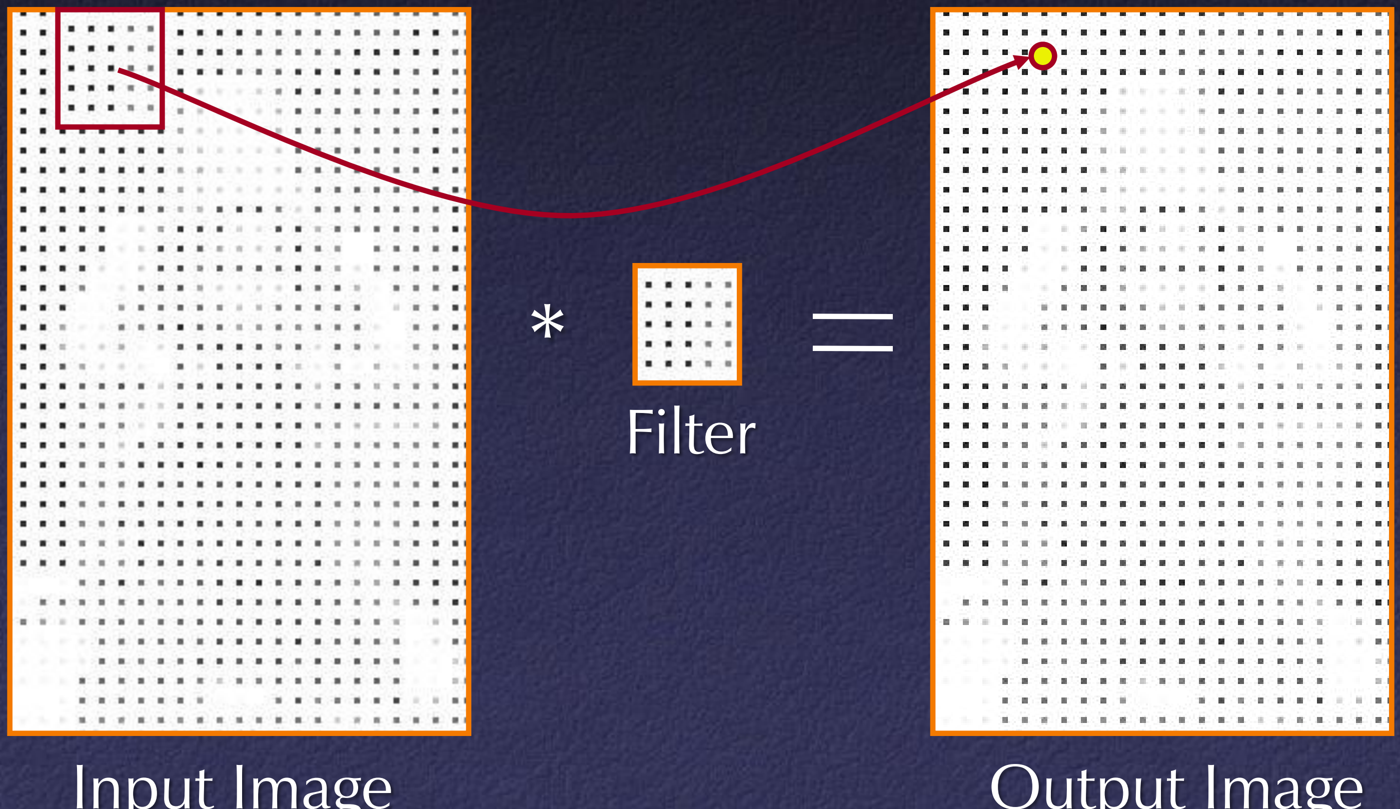
- Each output pixel is a linear combination of input pixels in neighborhood with weights prescribed by filter



Convolution

Discrete convolution

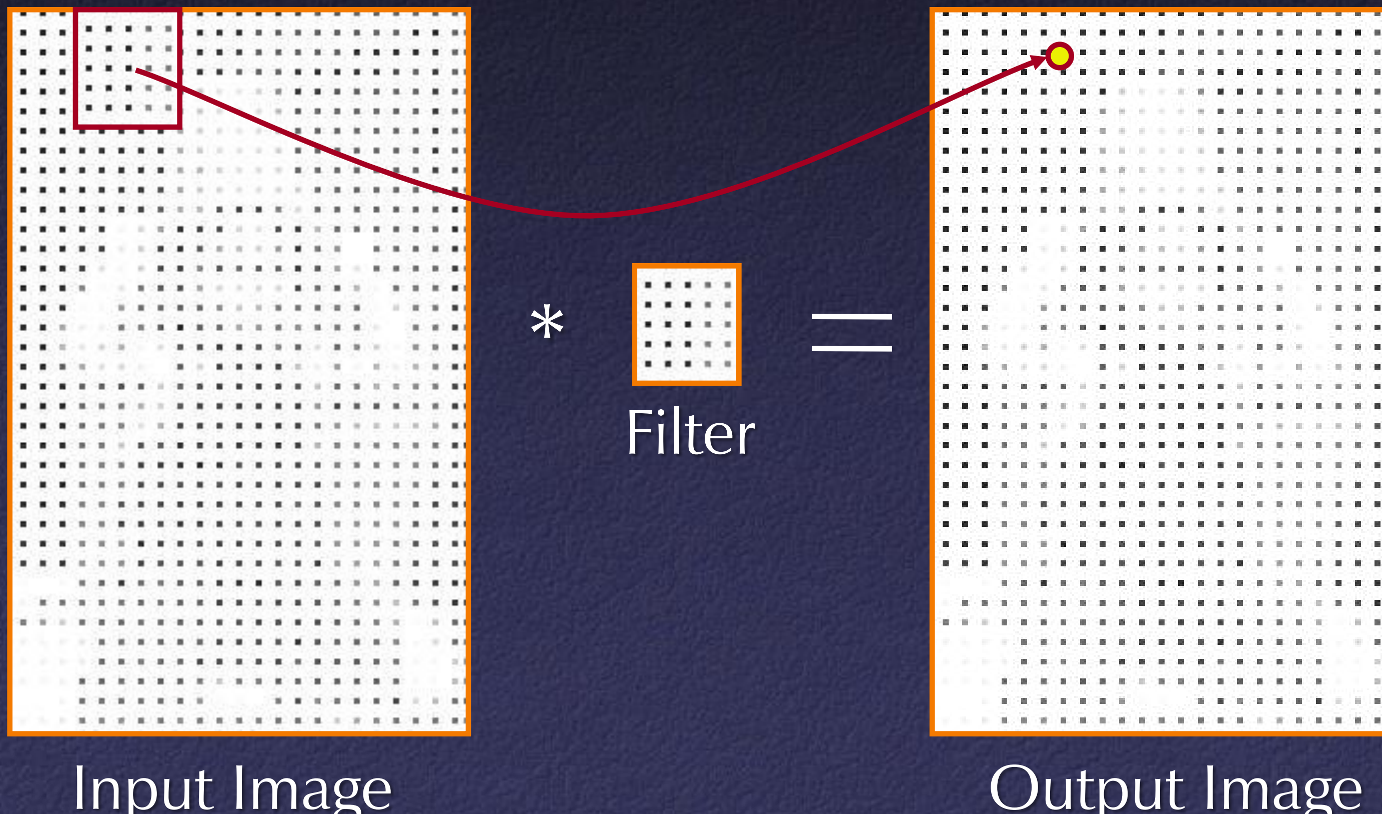
- Each output pixel is a linear combination of input pixels in neighborhood with weights prescribed by filter



Convolution

Discrete convolution

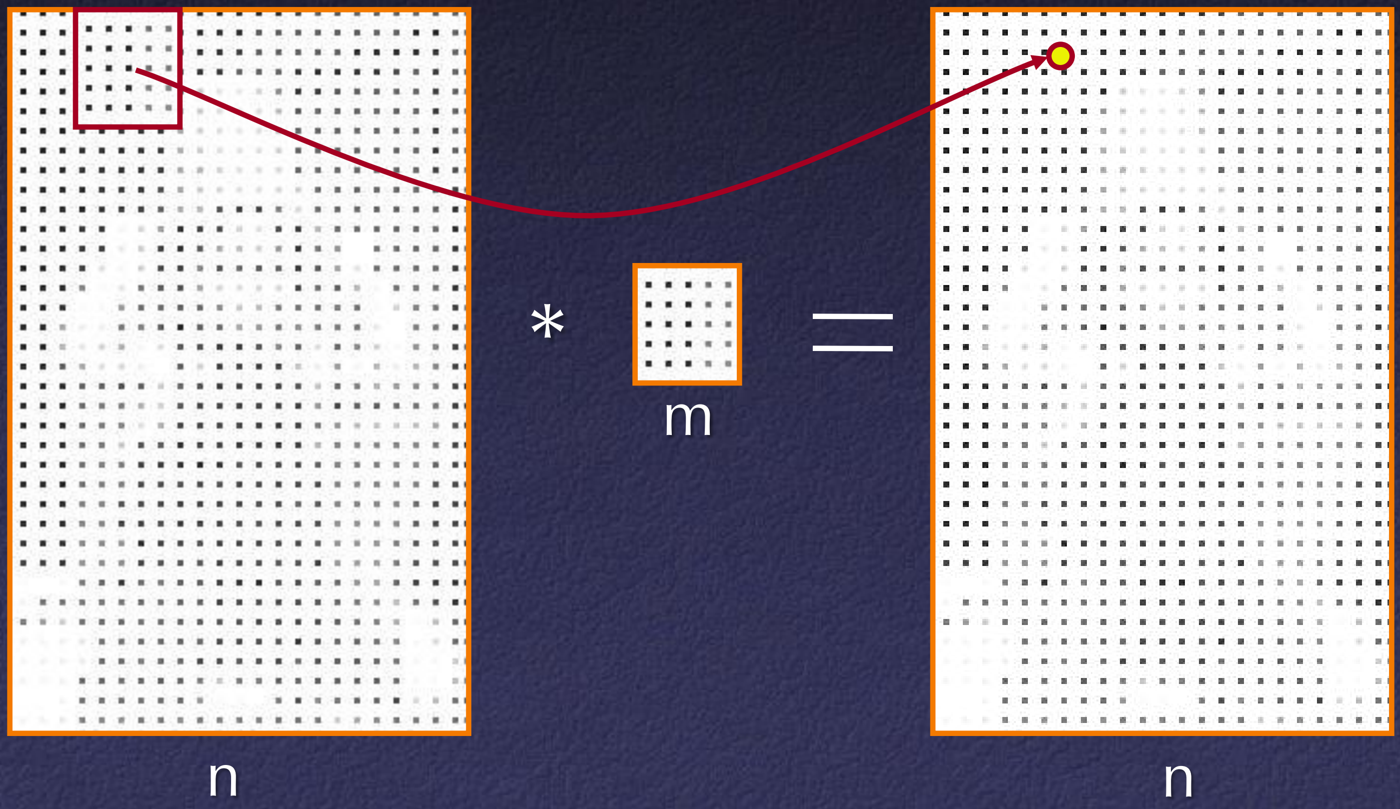
- Each output pixel is a linear combination of input pixels in neighborhood with weights prescribed by filter



Convolution

Discrete convolution

- Naïve process takes $O(n^2m^2)$ … OK for small filters (m)



Fourier Transform and Convolution

Useful fact: multiplication in frequency domain is same as convolution in spatial domain

$$f(x) * g(x) = \mathcal{F}^{-1}(\ \mathcal{F}(f(x))\ \mathcal{F}(g(x))\ )$$

Fourier Transform and Convolution

This provides a faster way to perform convolution for large filters:

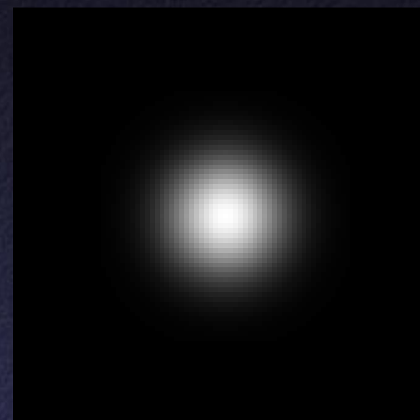
- Fast Fourier Transform (FFT) takes time
  $O(n \log n)$
- Thus, convolution can be performed in time
  $O(n \log n + m \log m)$

Fourier Transform and Convolution

Also, helps us reason about effects of specific filters

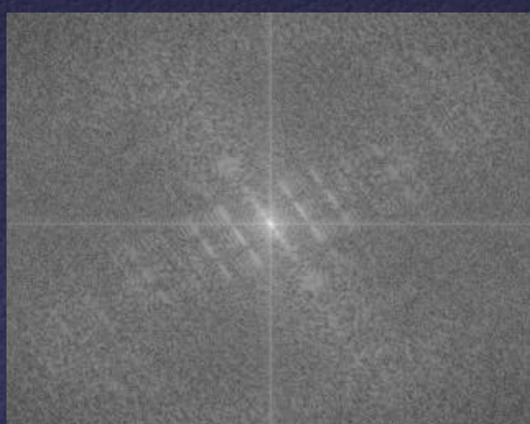
f(x,y)

*

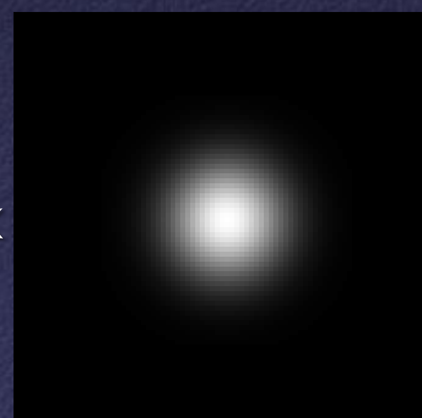


=

|F(u,v)|



x



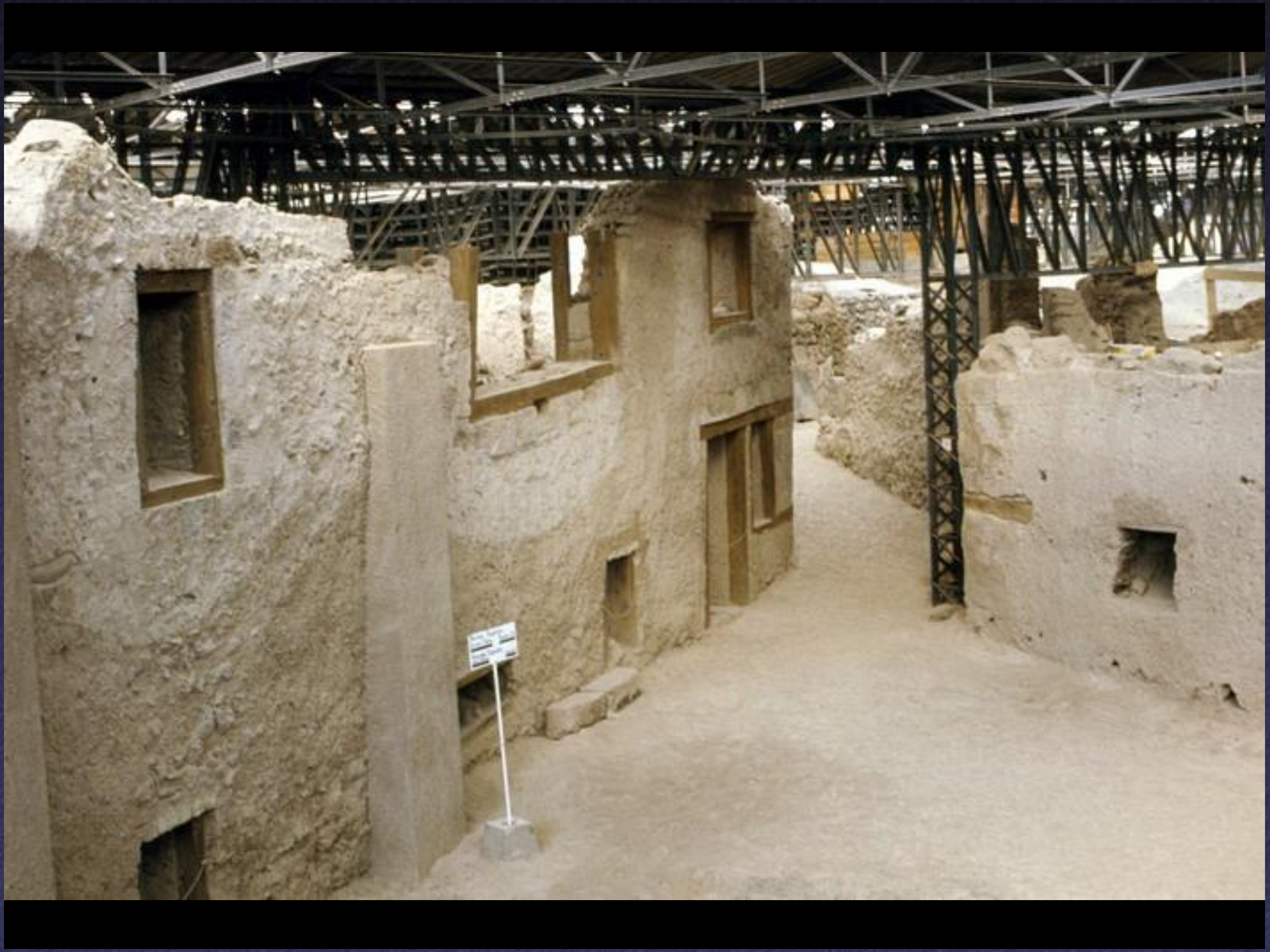
=

Gaussian

Application 2: Reconstructing Frescoes

Application 2: Reconstructing Frescoes

Akrotiri = buried city discovered in 1967

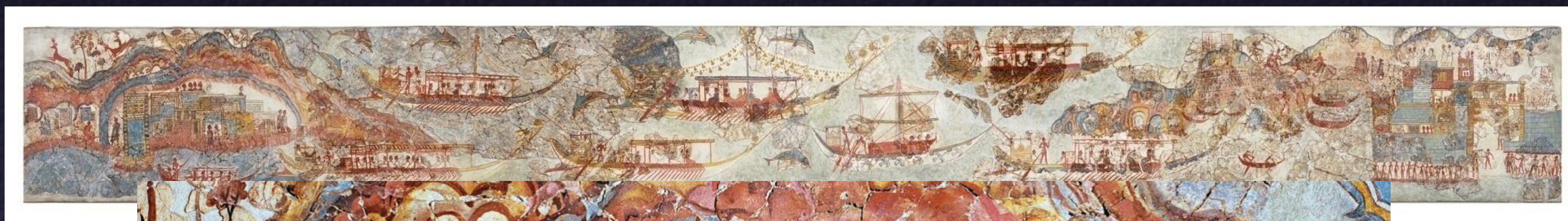


Application 2: Reconstructing Frescoes

Many walls were decorated with wall paintings

Application 2: Reconstructing Frescoes

Many walls were decorated with wall paintings



Application 2: Reconstructing Frescoes

… but most walls are shattered into fragments

Application 2: Reconstructing Frescoes

… but most walls are shattered into fragments

Application 2: Reconstructing Frescoes

… and re-assembling the fragments is difficult

Application 2: Reconstructing Frescoes

… and re-assembling the fragments is difficult

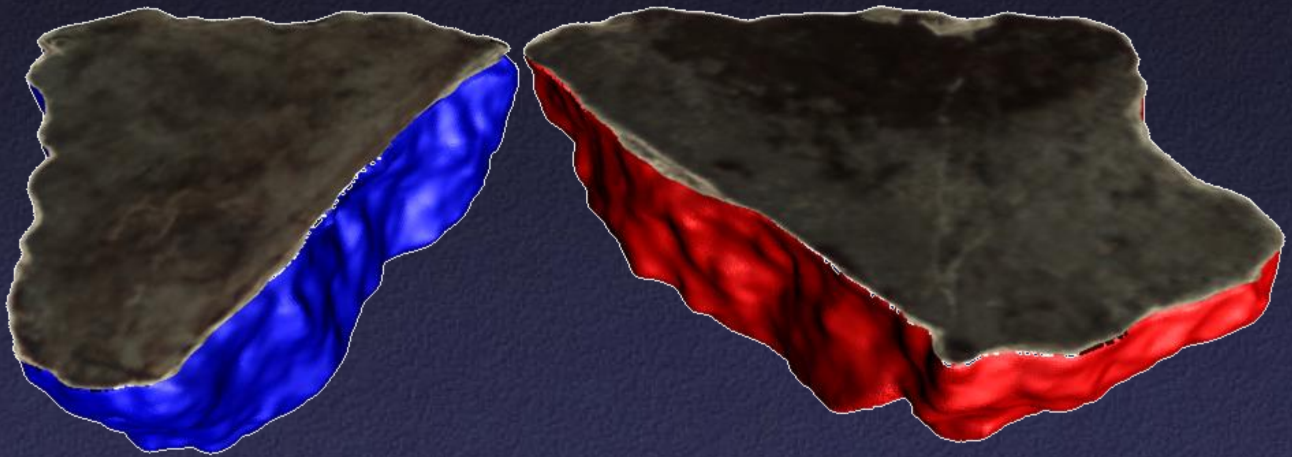
Application 2: Reconstructing Frescoes

Our project: scan surfaces of fragments

Fracture surface

Application 2: Reconstructing Frescoes

Our work: find matches between fragments



Application 2: Reconstructing Frescoes

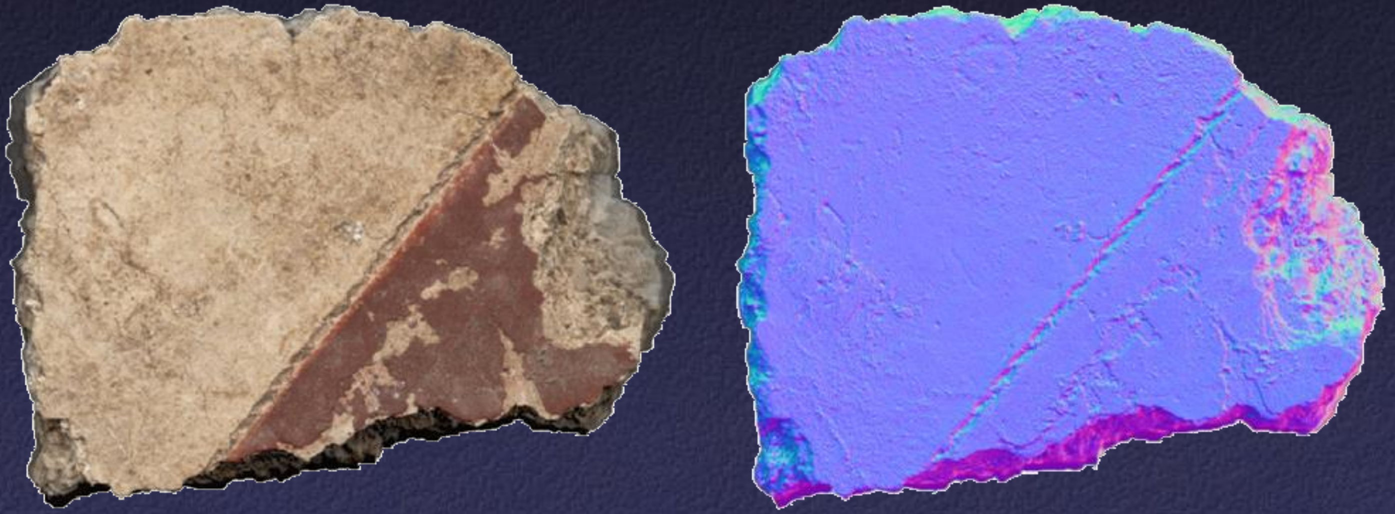
Our work: reconstruct fresco from matches

Candidate fragment matches

Reconstructed Fresco

Application 2: Reconstructing Frescoes

It turns out that subtle patterns in surface images are good cues for finding matches

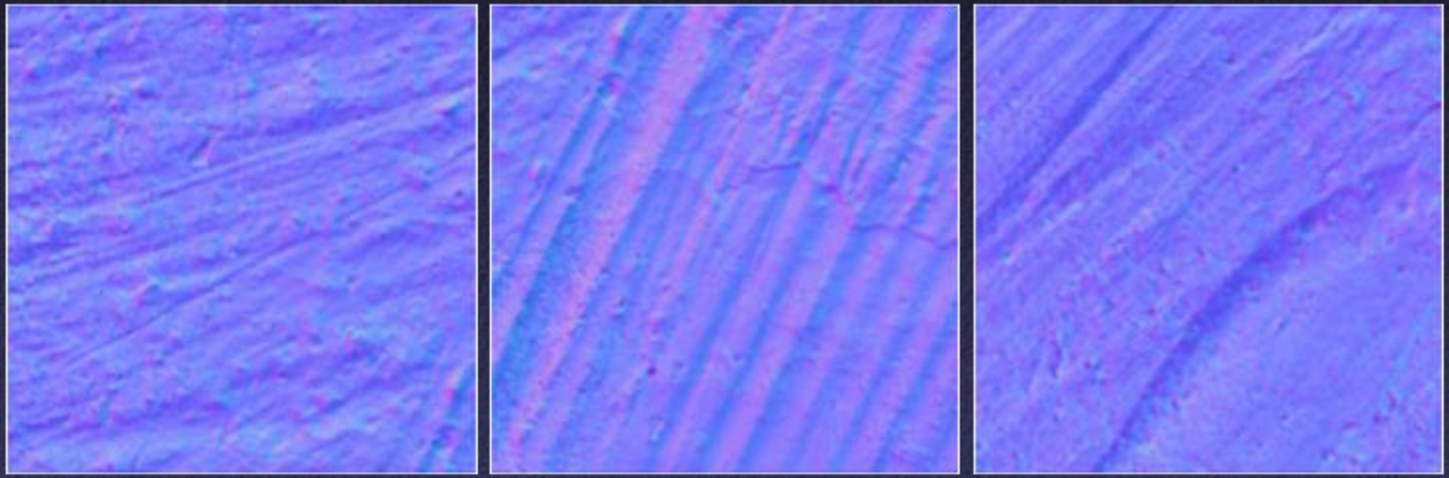


Surface patterns on a fresco fragment
(colors on right represent normal directions)

Toler-Franklin et al.

Application 2: Reconstructing Frescoes

Brush strokes appear as periodic functions with dominant frequency and orientation

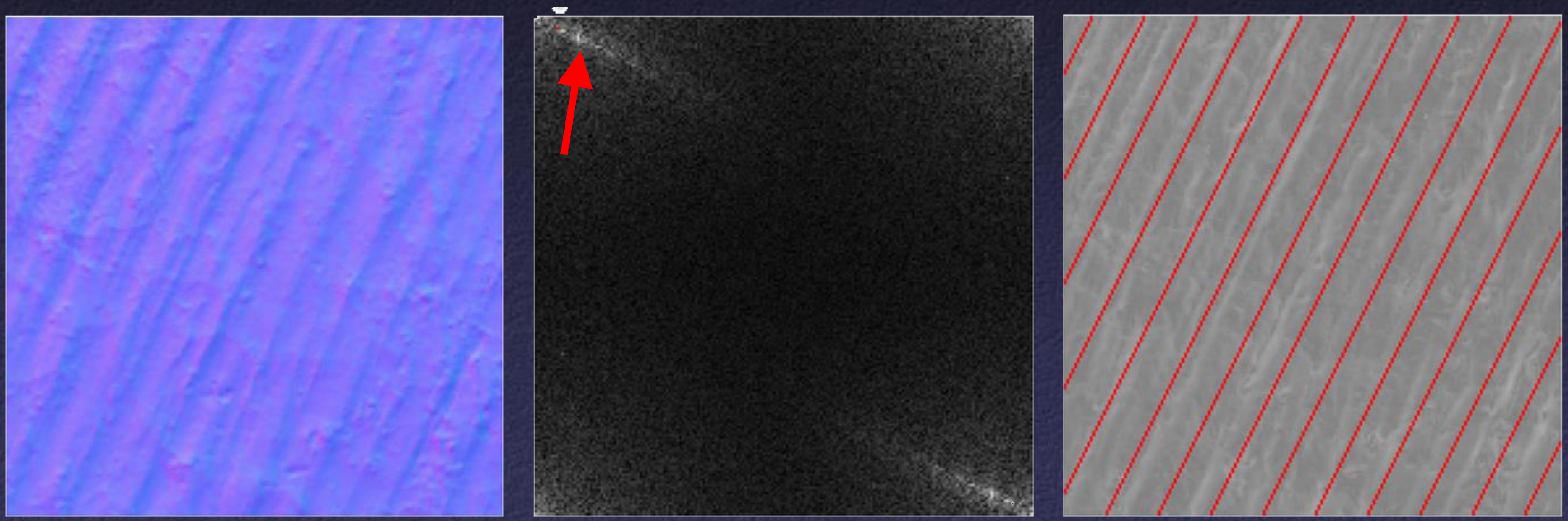


Brush patterns on different fresco fragments
(colors represent normal directions)

Toler-Franklin et al.

Application 2: Reconstructing Frescoes

Brush strokes appear as periodic functions with dominant frequency and orientation

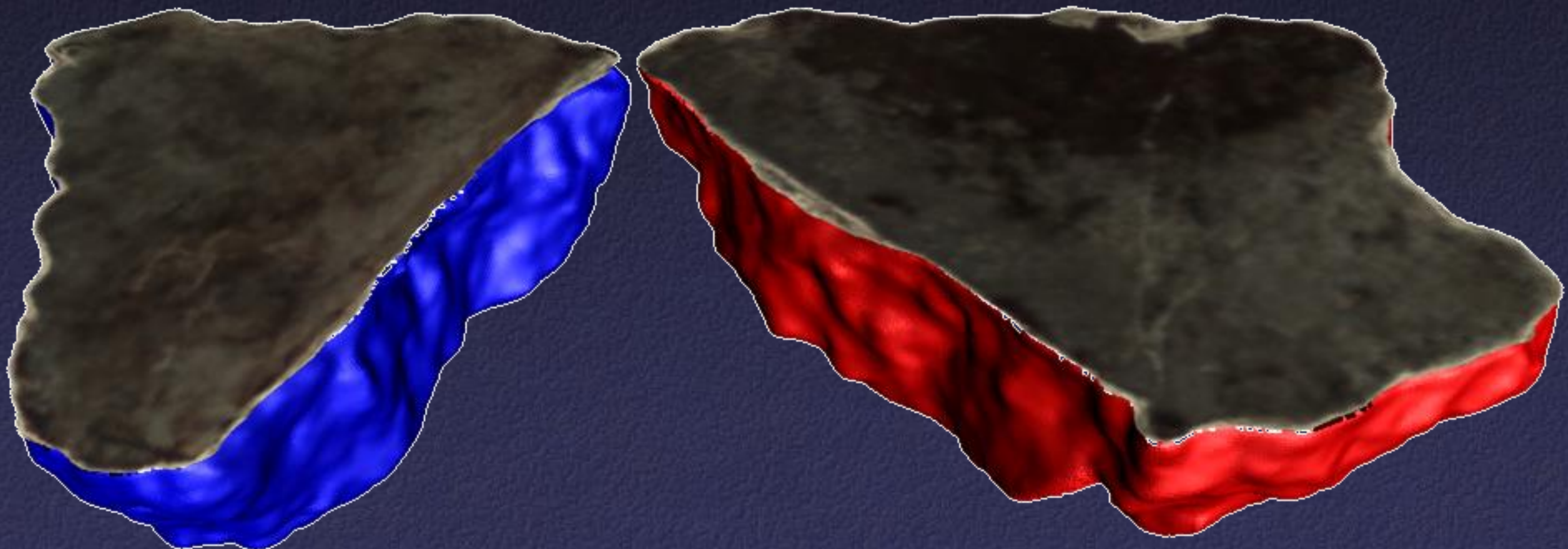


f(x) F(u) Dominant frequency and direction

Toler-Franklin et al.

Application 2: Reconstructing Frescoes

Consider alignment of brush strokes and other surface features when searching for matches



Toler-Franklin et al.

Image Analysis

What other tools do we have for analyzing functions?

Image Analysis

What other tools do we have for analyzing functions?

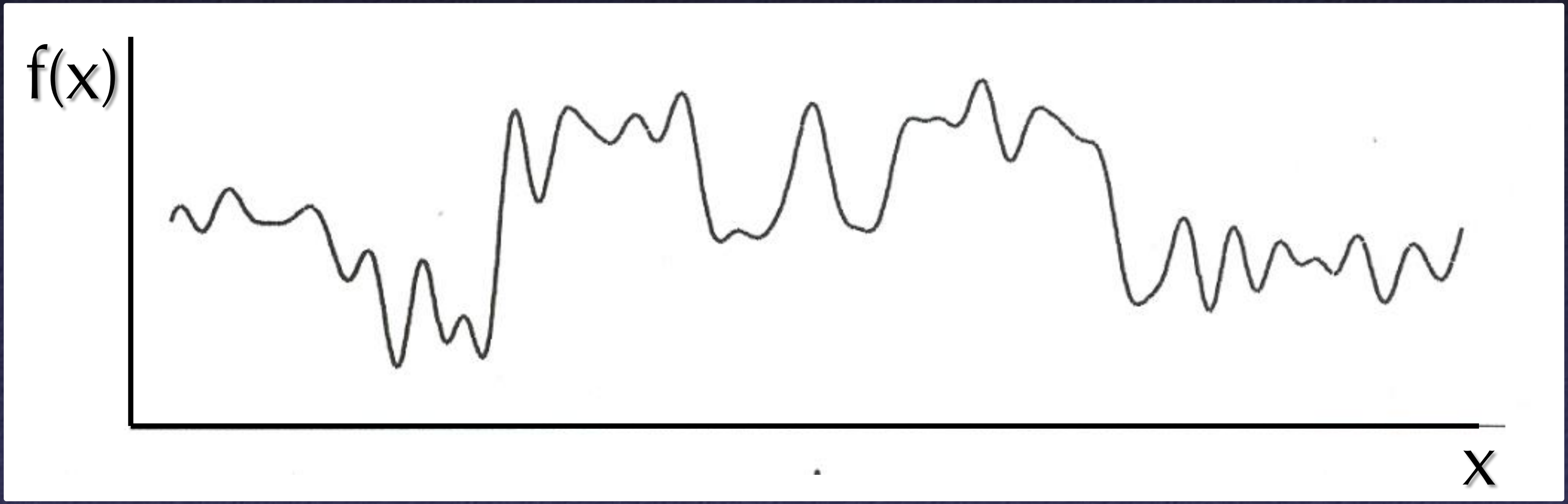


Image Analysis

What other tools do we have for analyzing functions?

- Let's look at gradients

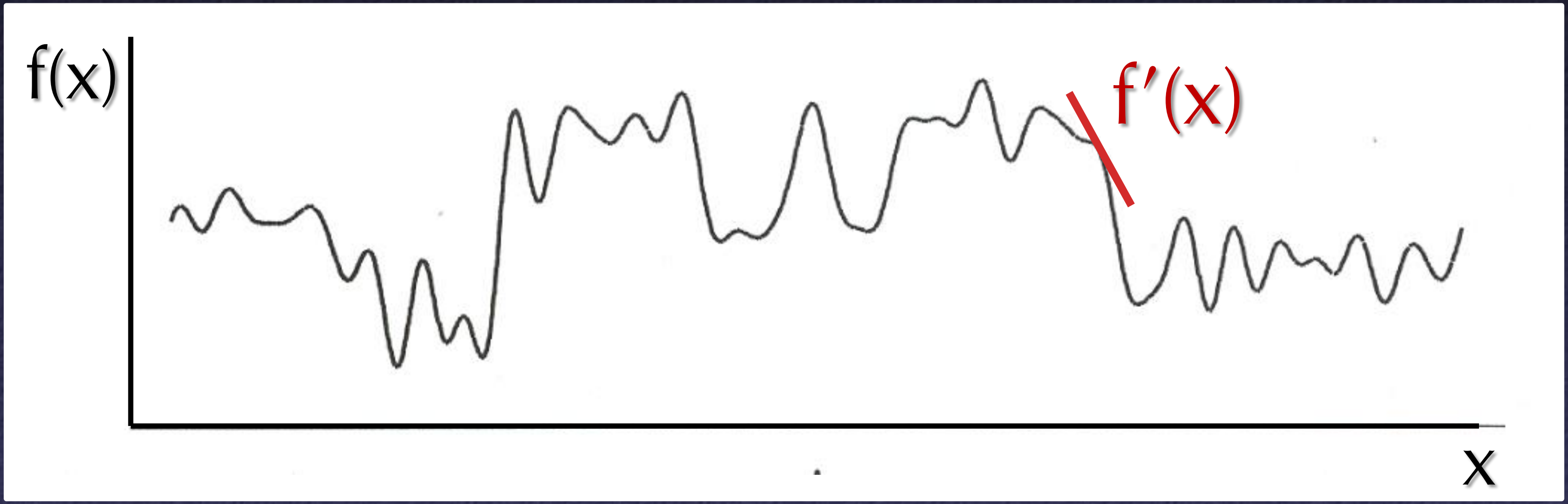


Image Gradients

For 2D function f(x,y), the partial derivative is:

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x+\varepsilon, y) - f(x,y)}{\varepsilon}$$

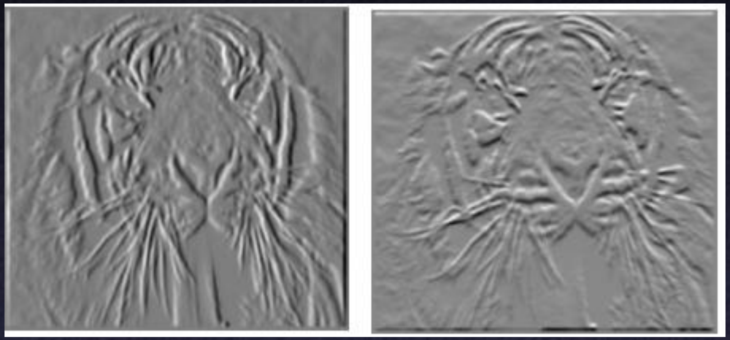
For discrete data, we can approximate using finite differences:

$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1, y) - f(x,y)}{1}$$

Image Gradients

The gradient of an image:

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



The magnitude of the gradient:

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

The direction of the gradient:

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

Computing Image Gradients

This is a convolution with two simple filters:

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

-1	1

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

-1
1

or

1
-1

Computing Image Gradients

Other common gradient filters:

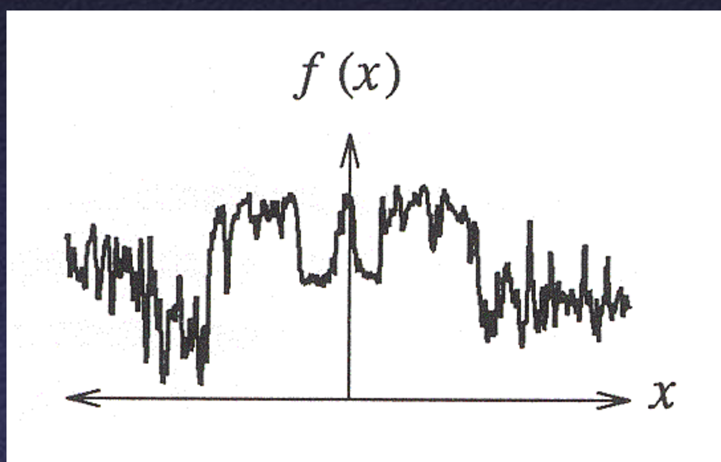
Prewitt: $$M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \quad ; \quad M_y = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

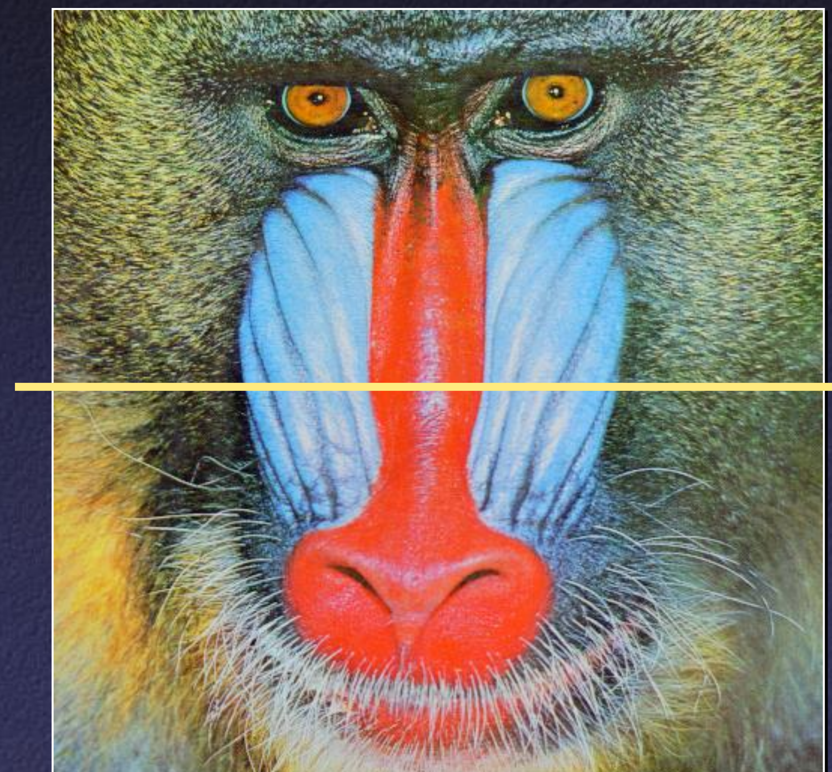
Sobel: $$M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \quad ; \quad M_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

Roberts: $$M_x = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline -1 & 0 \\ \hline \end{array} \quad ; \quad M_y = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline \end{array}$$

Computing Image Gradients

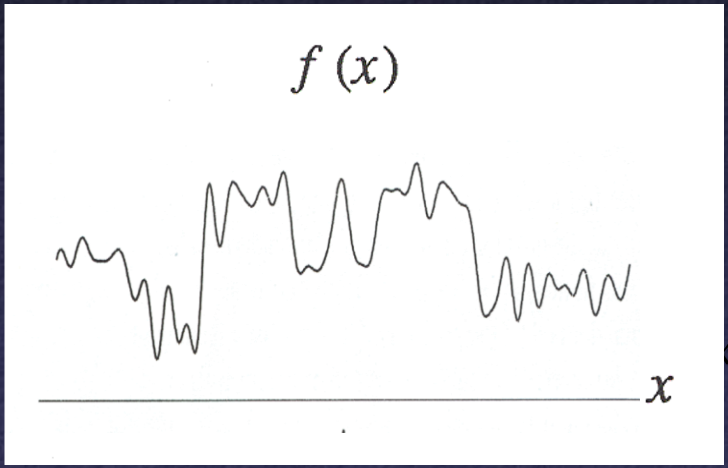
We usually limit high frequencies when computing gradient





Computing Image Gradients

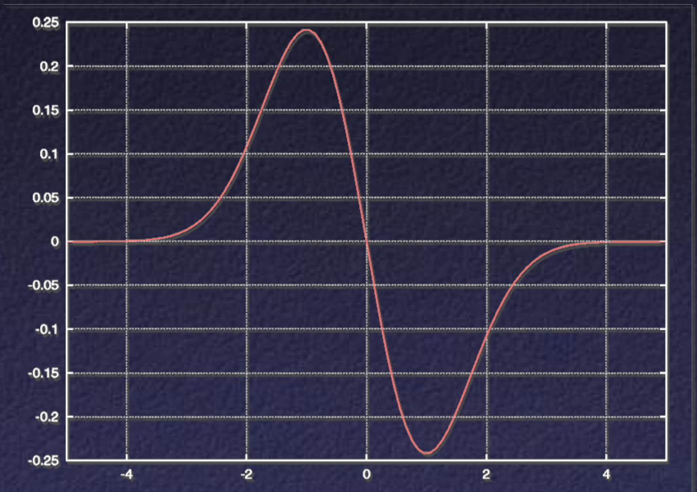
We usually limit high frequencies when computing gradient



Computing Image Gradients

Useful fact #1: differentiation "commutes" with convolution

$$\frac{df}{dx} * g = \frac{d}{dx}\left(f * g\right) = f * \frac{dg}{dx}$$



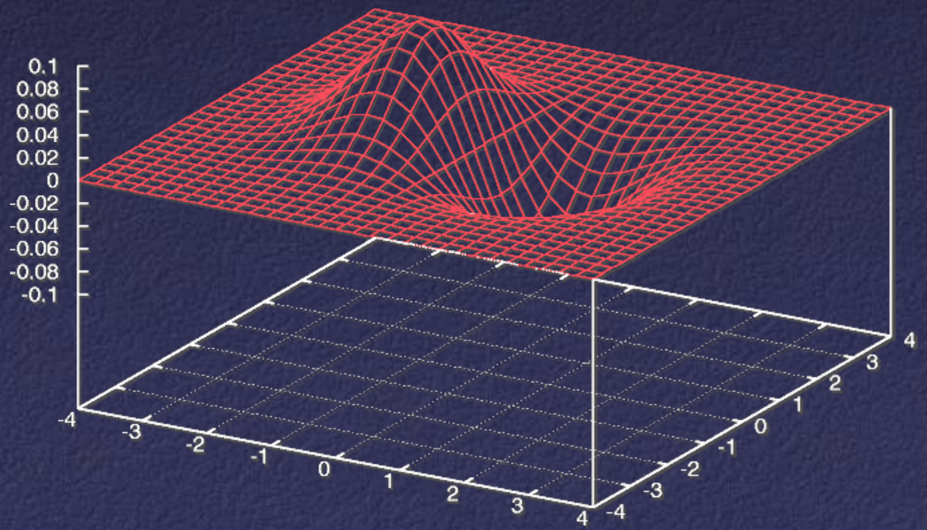
Useful fact #2: Gaussian is separable:

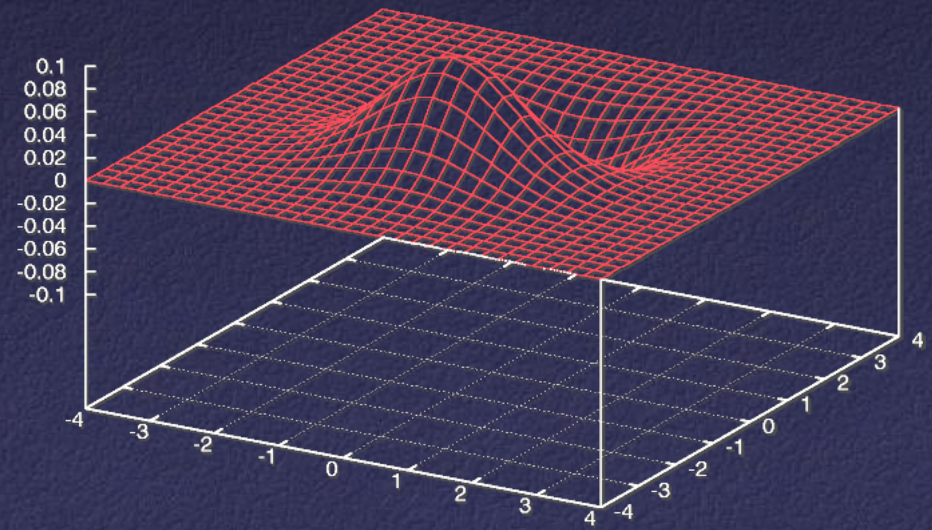
$$G_2(x,y) = G_1(x)G_1(y)$$

Computing Image Gradients

Thus, combine smoothing with gradient computation:

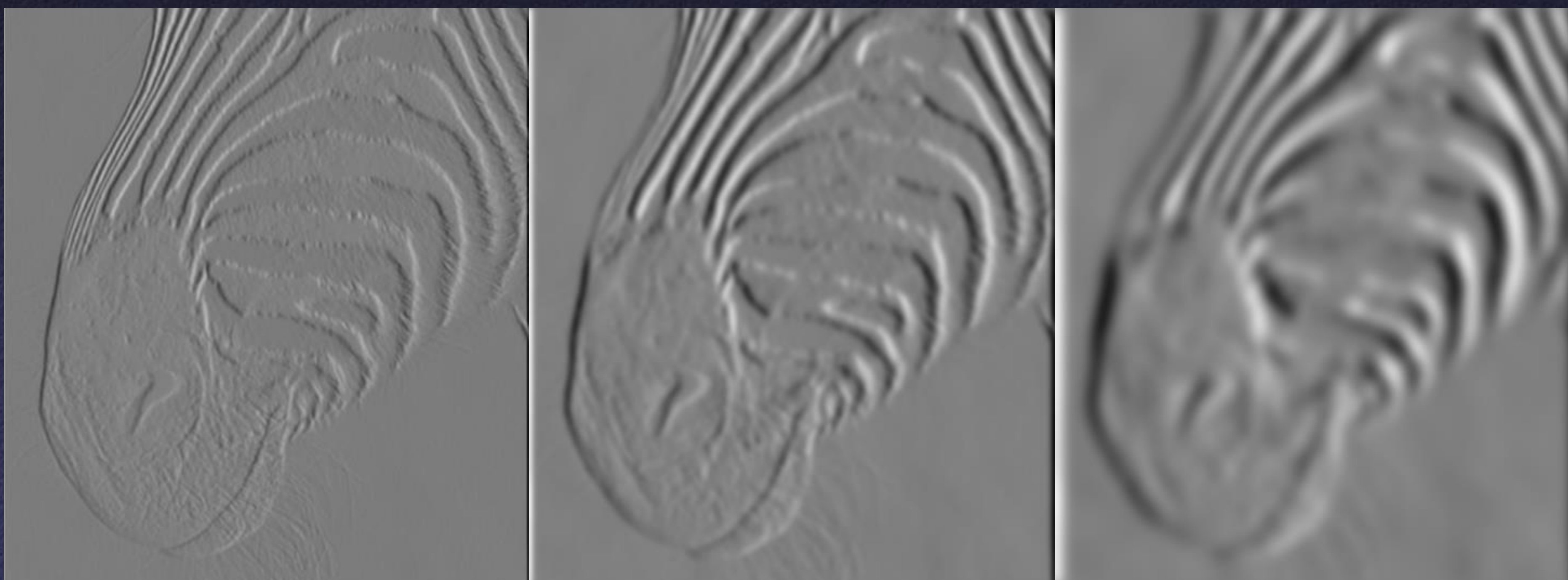
$$\nabla\left(f(x,y) * G_2(x,y)\right) = \begin{bmatrix} f(x,y) * \left(G_1'(x)G_1(y)\right) \\ f(x,y) * \left(G_1(x)G_1'(y)\right) \end{bmatrix} = \begin{bmatrix} f(x,y) * G_1'(x) * G_1(y) \\ f(x,y) * G_1(x) * G_1'(y) \end{bmatrix}$$





Computing Image Gradients

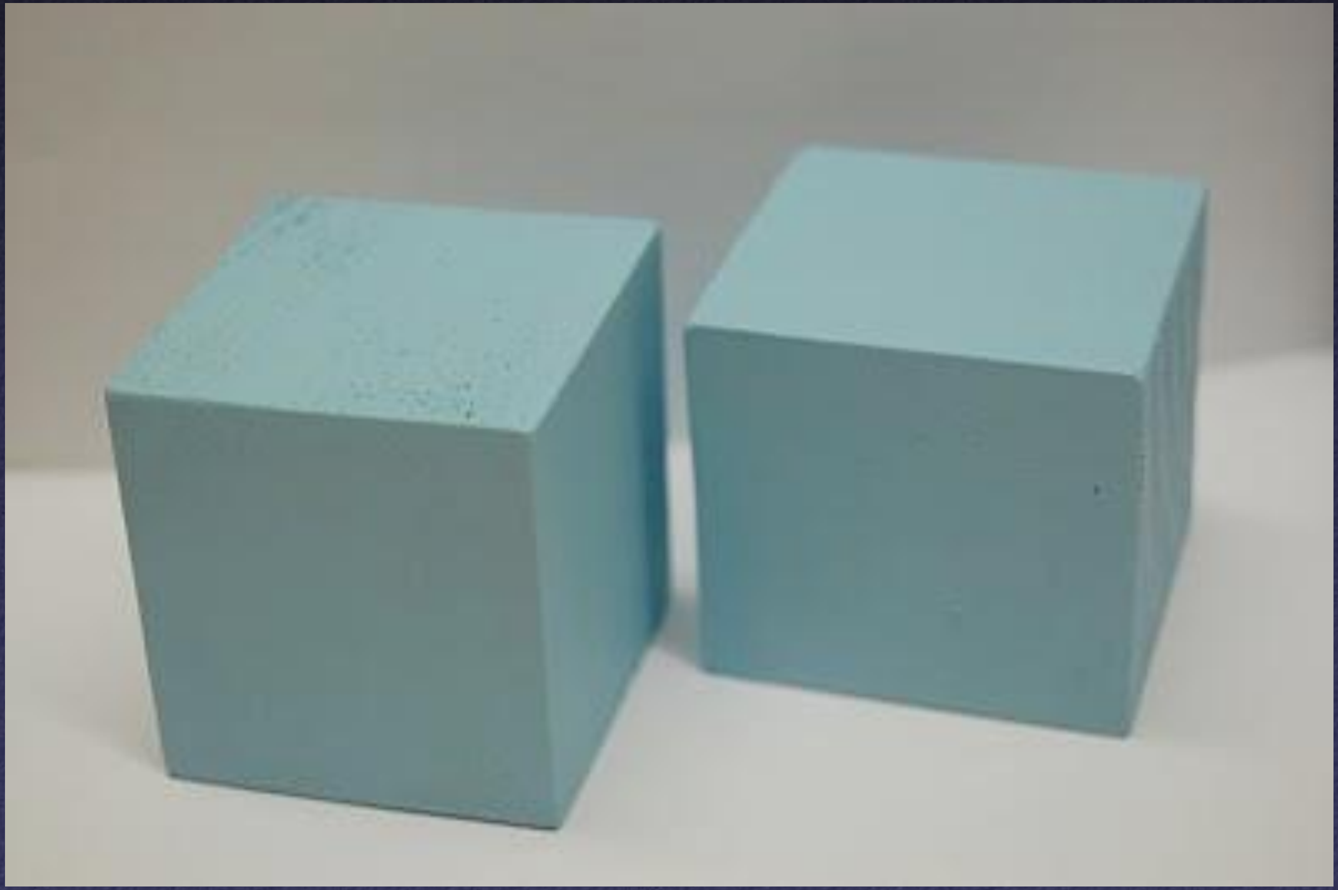
Can use different sigma to find gradients at different "scales"



1 pixel   3 pixels   7 pixels

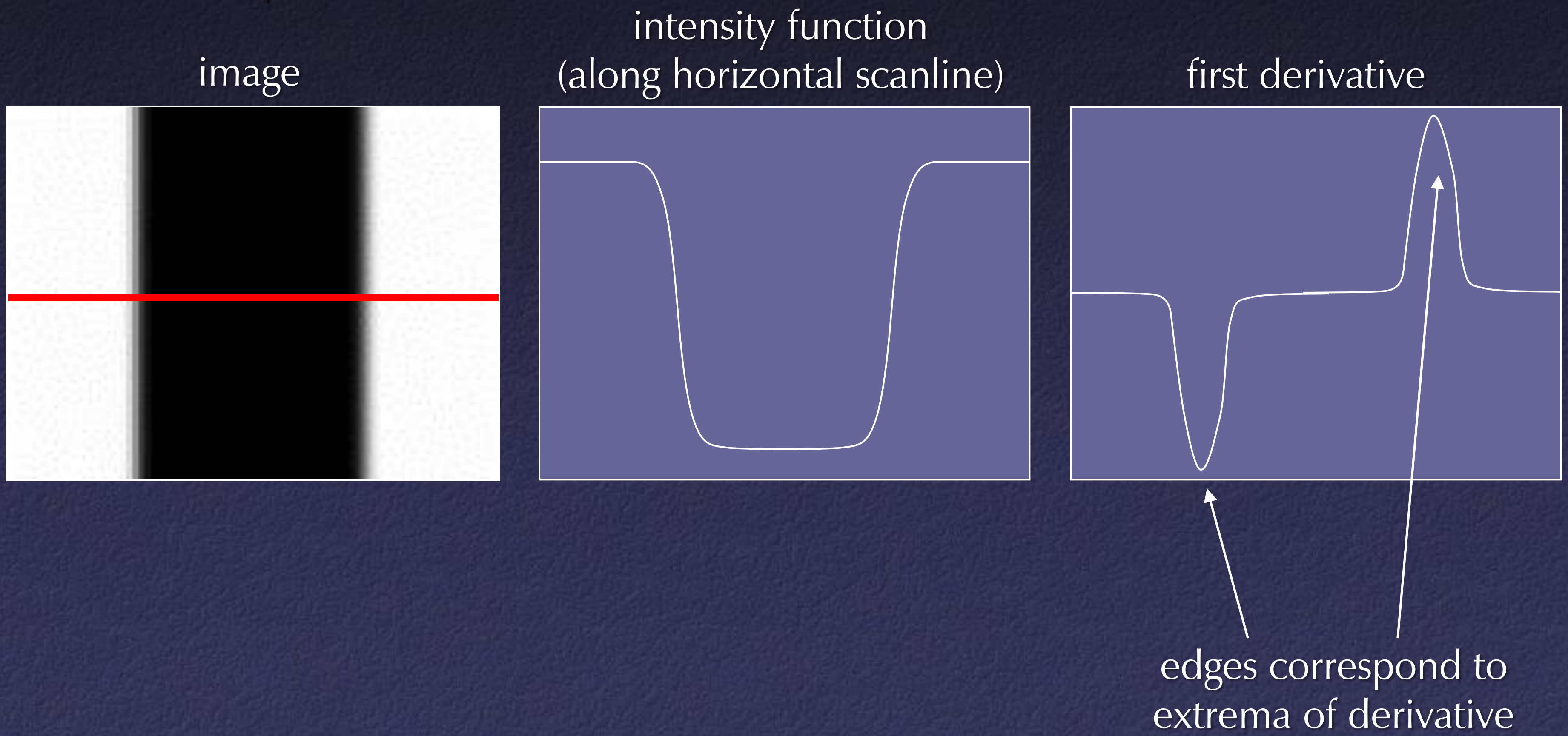
Gradient Analysis

How are image gradients useful?

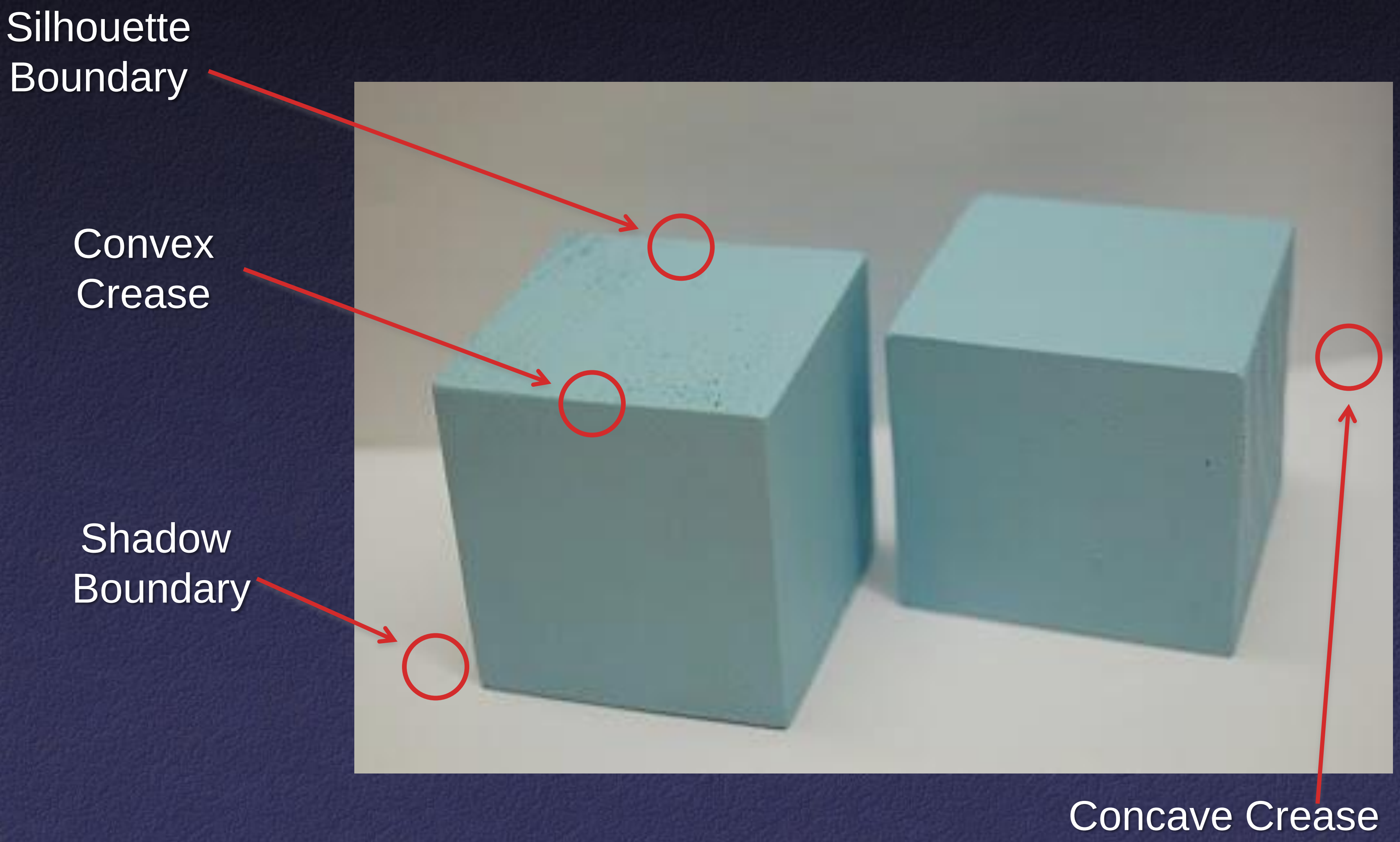


Edges

- An edge is a place of rapid change in the image intensity function



Edges


Silhouette Boundary
Convex Crease
Shadow Boundary
Concave Crease

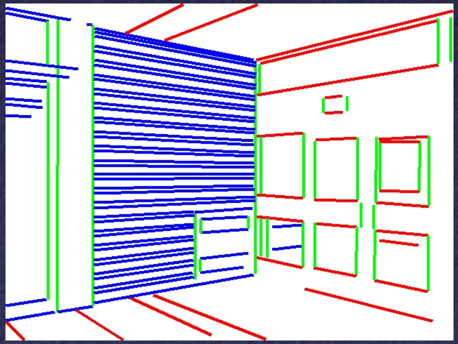
Edges

Edges

Edge Detection

Useful for many applications in vision

- Segmentation
- Camera pose estimation
- 3D reconstruction
- Object classification
- Object recognition
- etc.



Canny Edge Detector

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression
4. Hysteresis thresholding

Canny Edge Detector

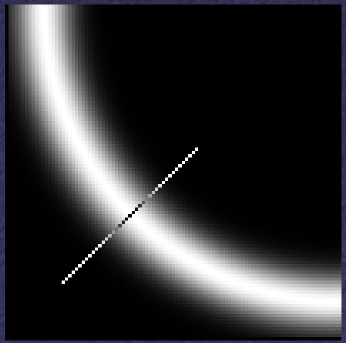
Original Image

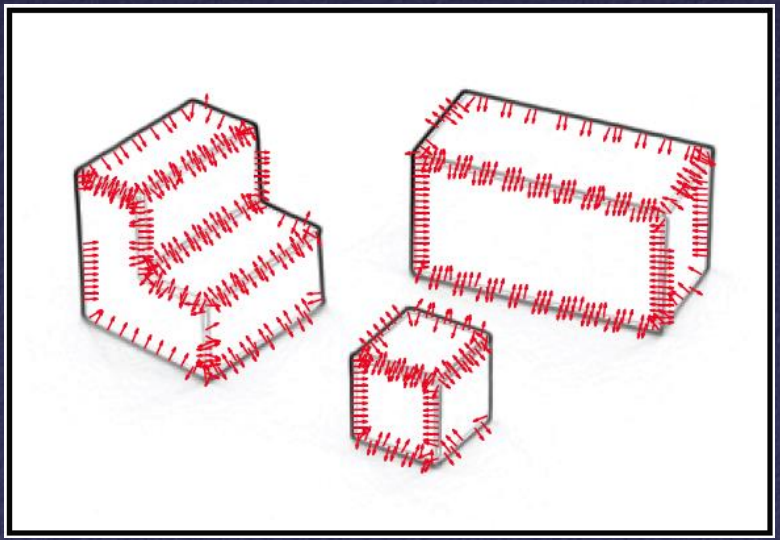
Smoothed Gradient Magnitude

Canny Edge Detector

Nonmaximum suppression

- Eliminate all but local maxima in *gradient magnitude* (sqrt of sum of squares of x and *y* components)
- At each pixel p look along *direction* of gradient: if either neighbor is bigger, set p to zero
- In practice, quantize direction to horizontal, vertical, and two diagonals
- Result: "thinned edge image"





Canny Edge Detector

Smoothed gradient magnitude

Non-maximum suppression

Canny Edge Detector

Final stage: thresholding

Simplest: use a single threshold

Better: use two thresholds

- Find chains of touching edge pixels, all $\geq \tau_{low}$
- Each chain must contain at least one pixel $\geq \tau_{high}$
- Helps eliminate dropouts in chains, without being too susceptible to noise
- "Thresholding with hysteresis"

Canny Edge Detector

Non-maximum suppression

Canny edges

Canny Edge Detector

Original Image

Canny edges

Summary of Canny Edge Detector

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - Thin wide “ridges” down to single pixel width
4. Hysteresis thresholding:
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

Summary of Today

Image analysis:

- Frequency analysis
  - Fourier transform
  - Convolution
- Gradient analysis
  - Edge detection