CS4501: Introduction to Computer Vision

Frequency, Edges, and Corners

Various slides from previous courses by:
D.A. Forsyth (Berkeley / UIUC), I. Kokkinos (Ecole Centrale / UCL). S. Lazebnik (UNC / UIUC), S. Seitz (MSR / Facebook), J. Hays (Brown / Georgia Tech), A. Berg (Stony Brook / UNC), D. Samaras (Stony Brook) . J. M. Frahm (UNC), V. Ordonez (UVA).

Last Class

- Google Colaboratory
- Recap on Convolutional Operations
- Image Gradients: The Sobel Operator
- Frequency Domain

Today’s Class

- Frequency Domain
- Filtering in Frequency
- Edge Detection - Canny
- Corner Detection - Harris

From Time to Frequency

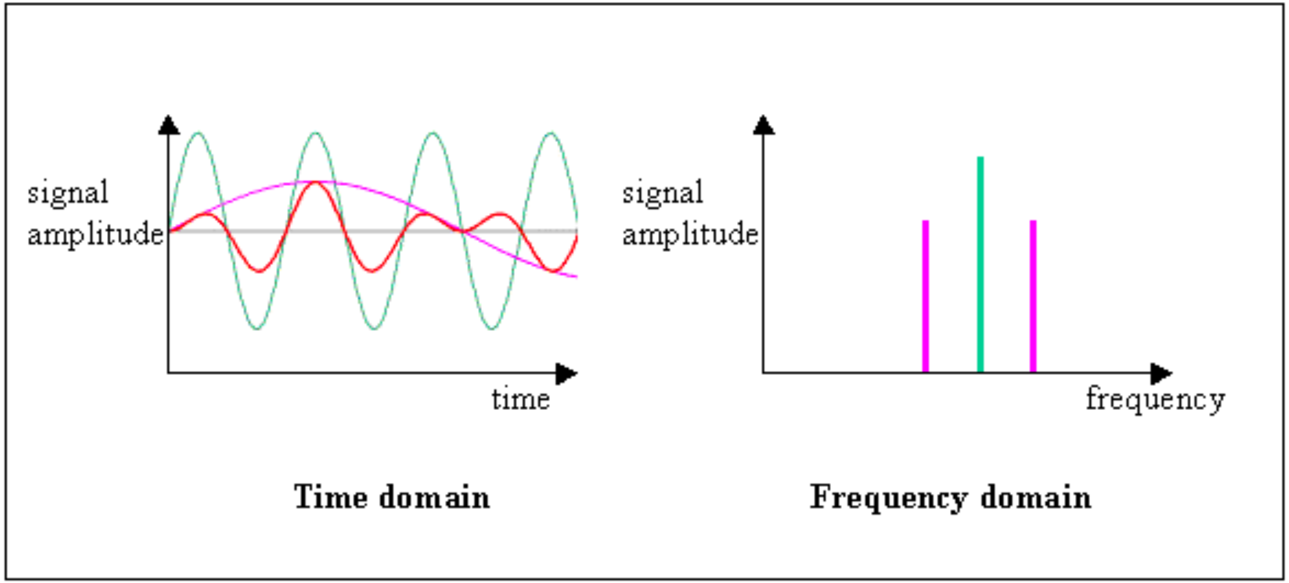
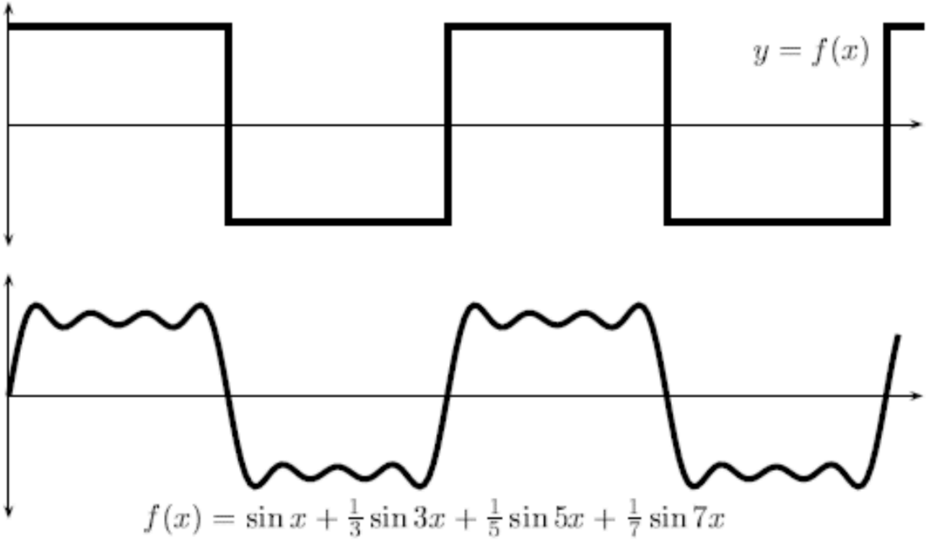


Figure by National Instruments

Example

Square wave

Approximation Using sines



Slide by Emmanuel Agu

Crucial Missing Step (Leap) from Last Class

$$f(t) = \frac{a_0}{2} + \sum_{n=1}^{\infty} (a_n \cos n\omega_0 t + b_n \sin n\omega_0 t)$$

$$\downarrow$$

$$f(t) = \sum_{n=-\infty}^{\infty} c_n \exp(in\omega_0 t)$$

http://www.personal.soton.ac.uk/jav/soton/HELM/workbooks/workbook_23/23_6_complex_form.pdf

Discrete
Fourier
Transform

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

Inverse
Discrete
Fourier
Transform

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

More generally for images (2D DFT and iDFT)

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) \exp\left[-2\pi i \left(\frac{xu}{M} + \frac{yv}{N}\right)\right]$$

$$f(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) \exp\left[2\pi i \left(\frac{xu}{M} + \frac{yv}{N}\right)\right]$$

Keep in mind Euler's Equation

$$e^{ix} = \cos x + i \sin x$$

We can compute the real and the imaginary part of the complex number.

Fourier Transform

- Fourier transform stores the magnitude and phase at each frequency
  - Magnitude encodes how much signal there is at a particular frequency
  - Phase encodes spatial information (indirectly)
  - For mathematical convenience, this is often notated in terms of real and complex numbers

Amplitude: $A = \pm\sqrt{R(\omega)^2 + I(\omega)^2}$

Phase: $\phi = \tan^{-1}\frac{I(\omega)}{R(\omega)}$

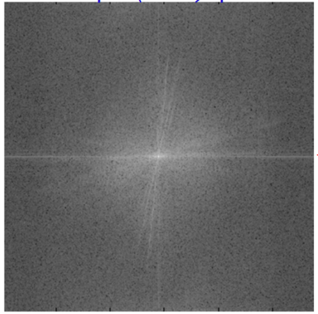
Slide by James Hays

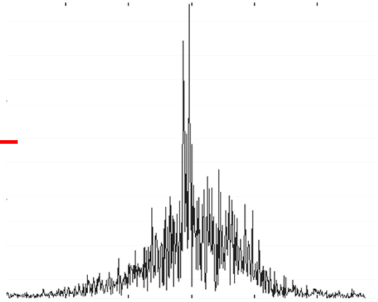
Discrete Fourier Transform - Visualization

$|F(u,v)|$

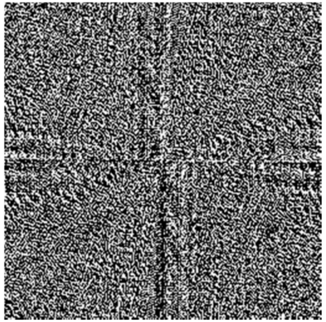
cross-section

$f(x,y)$





phase $F(u,v)$



- |f(u,v)| generally decreases with higher spatial frequencies
- phase appears less informative

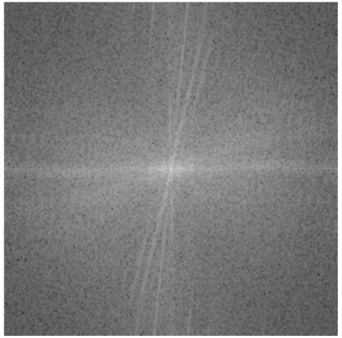
Slide by A. Zisserman

Image Filtering in the Frequency Domain

original

$f(x,y)$

$|F(u,v)|$



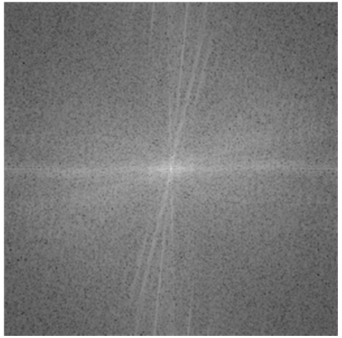
Slide by A. Zisserman

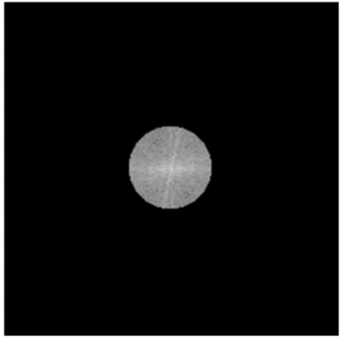
Image Filtering in the Frequency Domain

original

$f(x,y)$

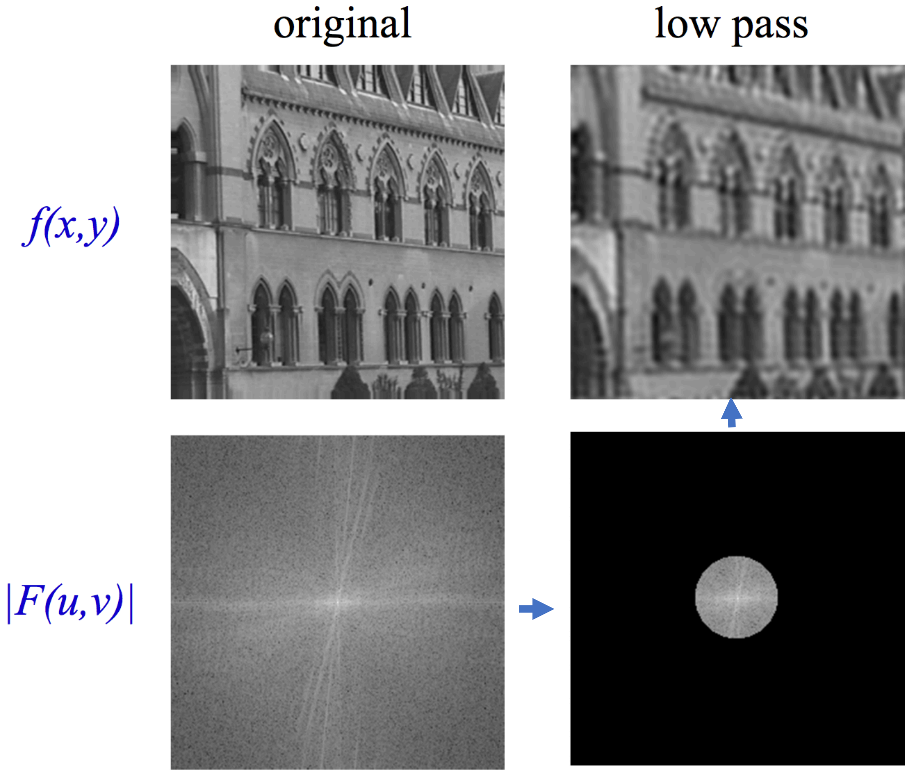
$|F(u,v)|$





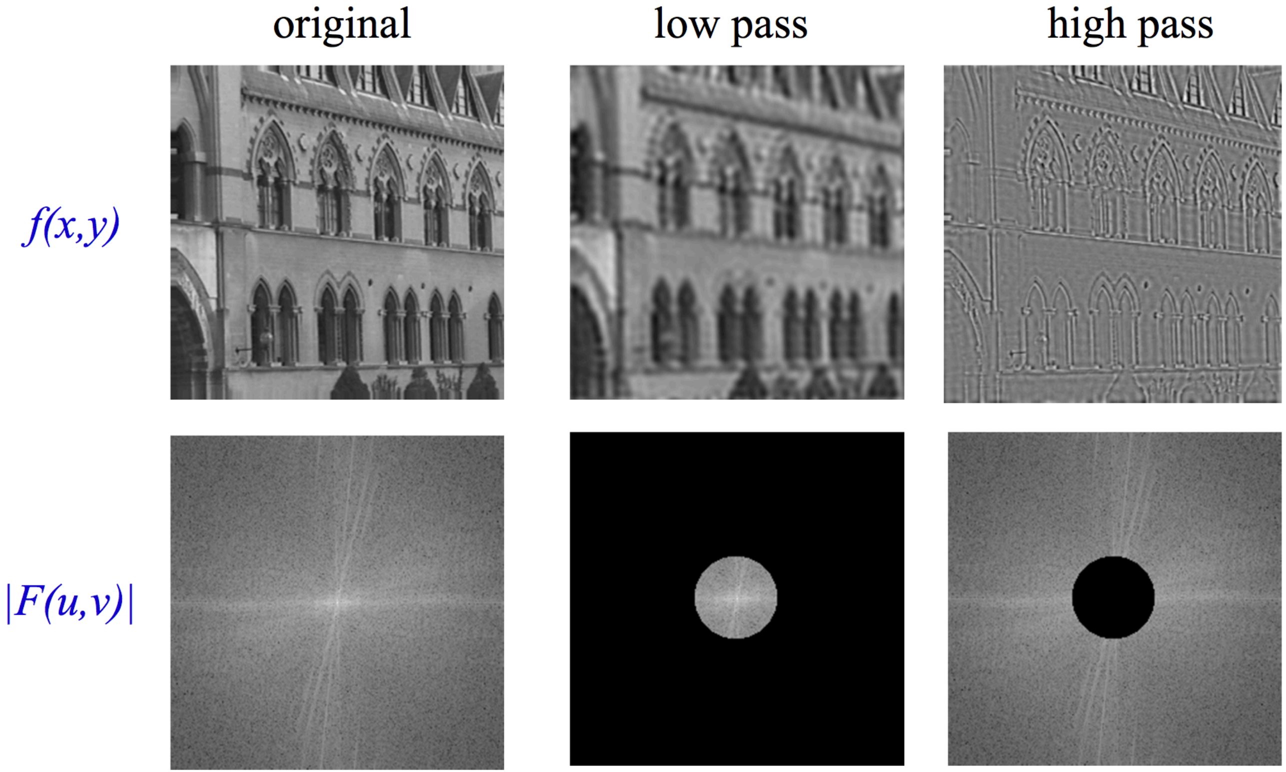
Slide by A. Zisserman

Image Filtering in the Frequency Domain



Slide by A. Zisserman

Image Filtering in the Frequency Domain



Slide by A. Zisserman

The Convolution Theorem

- The Fourier transform of the convolution of two functions is the product of their Fourier transforms

$$\mathrm{F}[g * h] = \mathrm{F}[g]\,\mathrm{F}[h]$$

- **Convolution** in spatial domain is equivalent to **multiplication** in frequency domain!

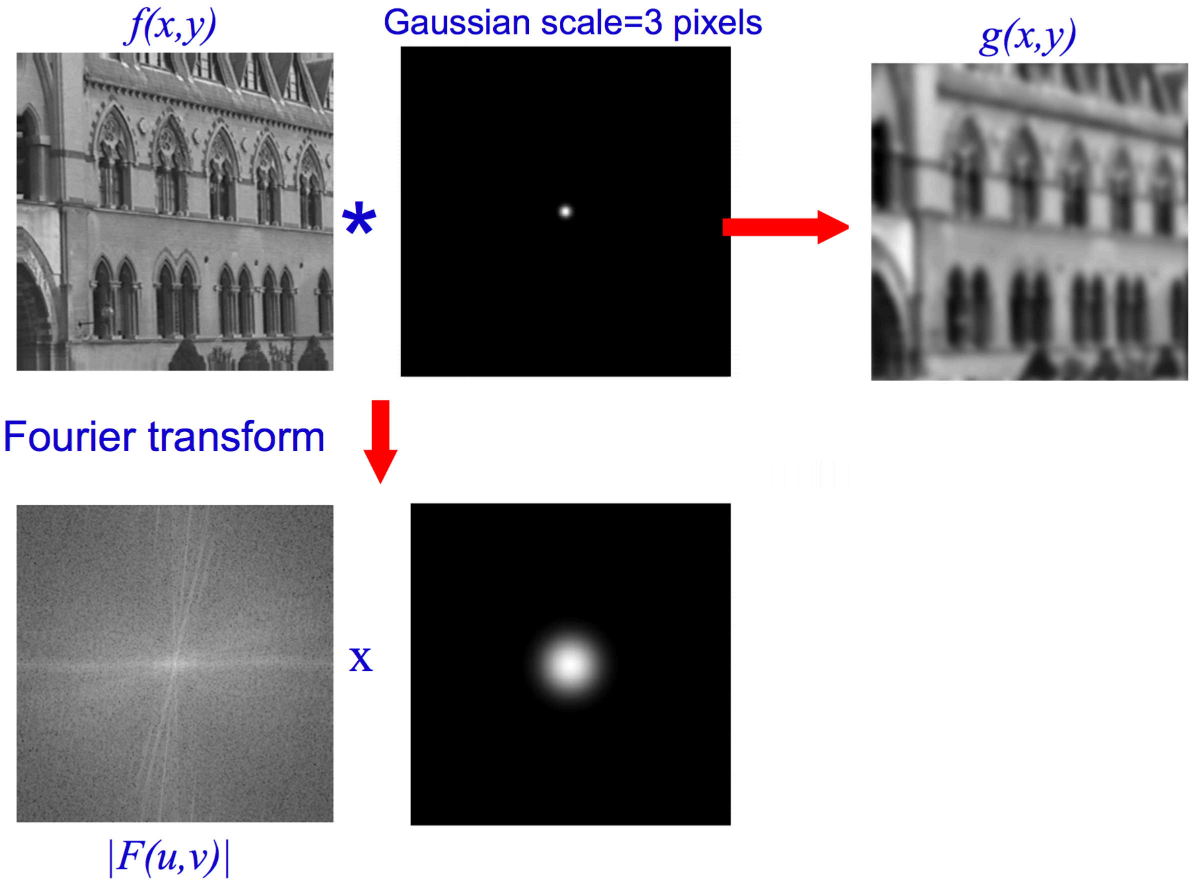
$$g * h = \mathrm{F}^{-1}[\mathrm{F}[g]\,\mathrm{F}[h]]$$

How can this be useful?

Blurring in the Time vs Frequency Domain

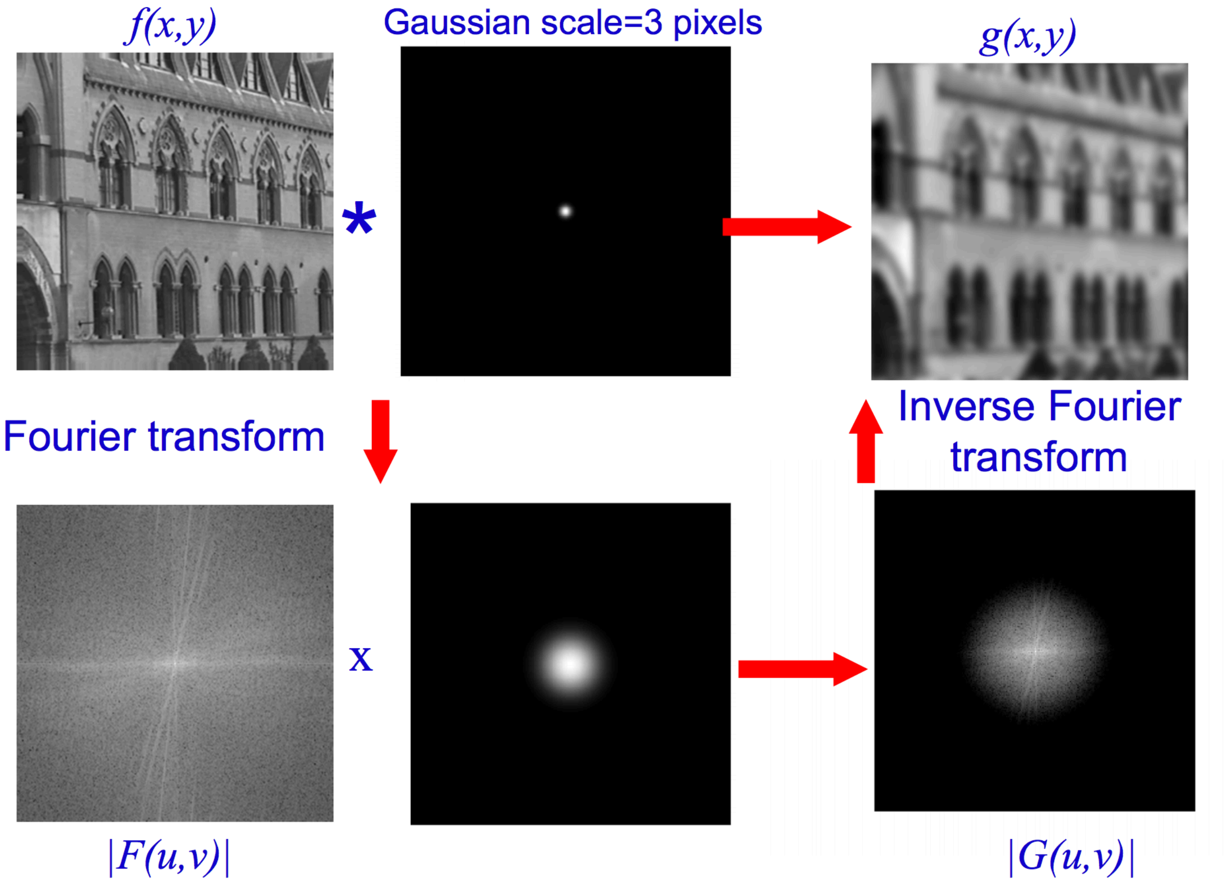
Example by A. Zisserman

Blurring in the Time vs Frequency Domain



Example by A. Zisserman

Blurring in the Time vs Frequency Domain

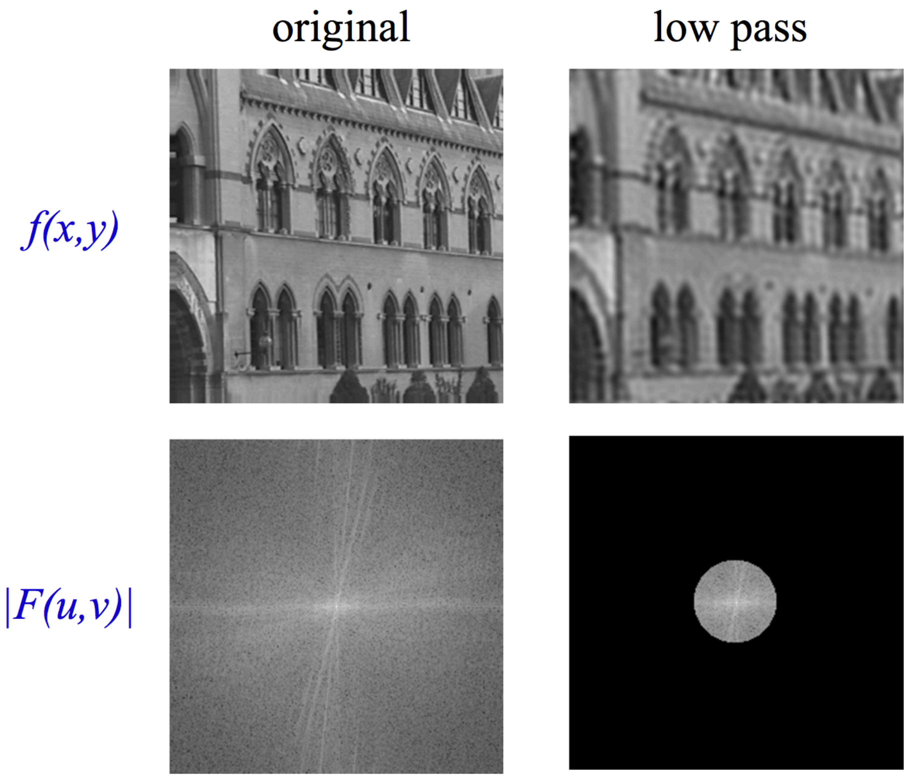


Example by A. Zisserman

Why Frequency domain?

- Because the Discrete Fourier Transform can be computed fast using the Fast Fourier Transform FFT algorithm.
- Because the running time does not depend on the size of the kernel matrix.
- However rarely used these days because most filters used in Computer vision are 3x3, 5x5, e.g. relatively small.

Final Thoughts – JPEG Image Compression



- Small amount of information can recover almost the original image with some loss in resolution.
- Images are dominated by low frequency information. e.g. no need to store repeated pixels.
- In practice JPEG uses a simpler transformation called Discrete Cosine Transform DCT.

Edge Detection

Edge Detection

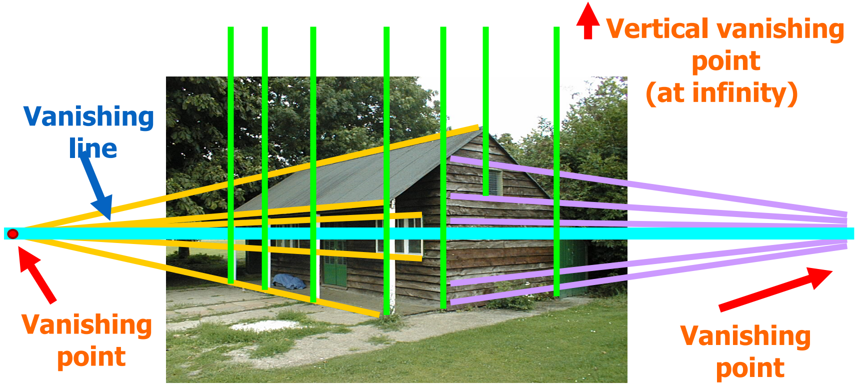
- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels
- **Ideal:** artist’s line drawing (but artist is also using object-level knowledge)

Source: D. Lowe

Why do we care about edges?

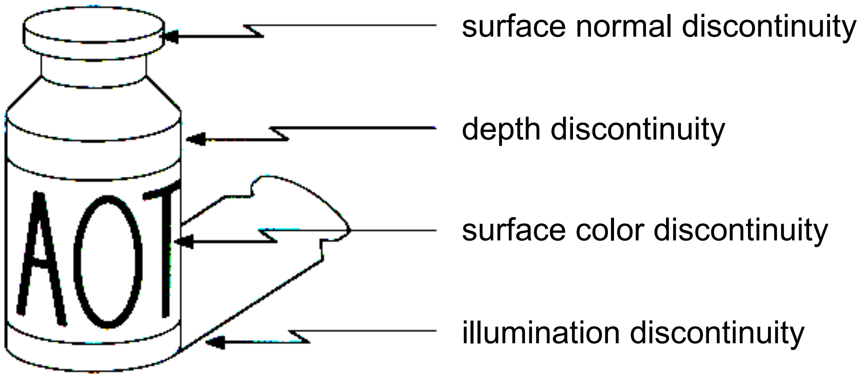
- Extract information, recognize objects

- Recover geometry and viewpoint



Source: J. Hays

Origin of Edges



- Edges are caused by a variety of factors

Source: Steve Seitz

Edge Detection

- Back to Sobel

But Ideally we want an output where:

g(x,y) = 1 if edge

g(x,y) = 0 if background

Edge Detection

- Sobel + Thresholding

$$g(x,y) = \begin{cases} 1, & f(x,y) \geq \tau \\ 0, & f(x,y) < \tau \end{cases}$$

- Sobel + Thresholding

$$g(x,y) = \begin{cases} 1, & f(x,y) \geq \tau \\ 0, & f(x,y) < \tau \end{cases}$$

Problems:

- Edges are too wide: We want 1-pixel wide edges if possible.
- Lots of disconnected edges: We want to respect continuity or connectivity.

Solution: Canny edge detector

- Suppress Noise
- Compute gradient magnitude and direction
- Non-Maximum Suppression
- Hysteresis and connectivity analysis

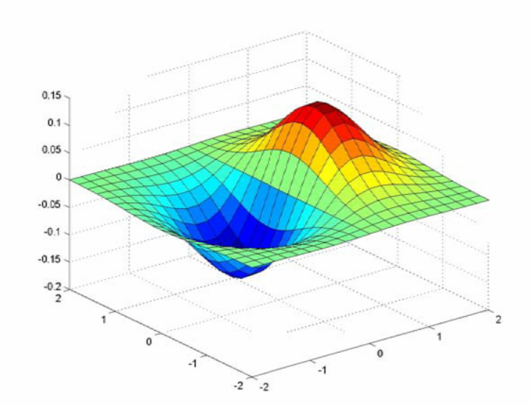
Similar to Sobel:
Blurring + Gradients

Example

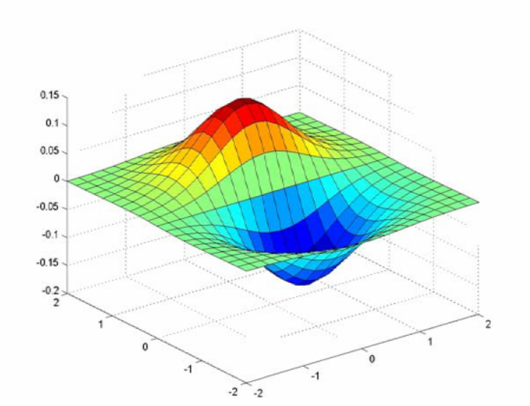
- original image

Source: Juan C. Niebles and Ranjay Krishna.

Derivative of Gaussian filter

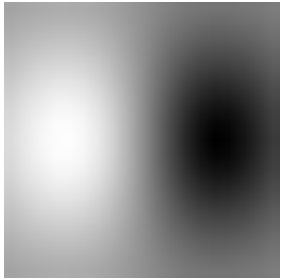


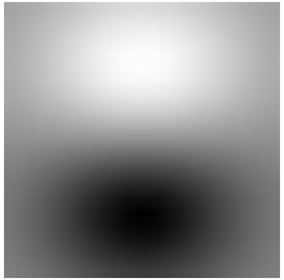
x-direction



y-direction

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

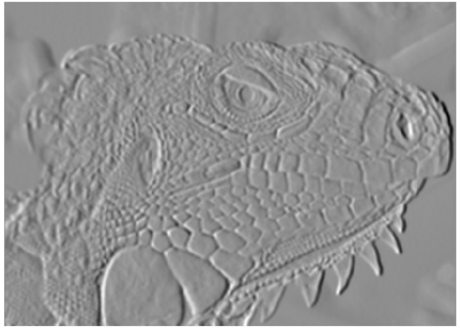




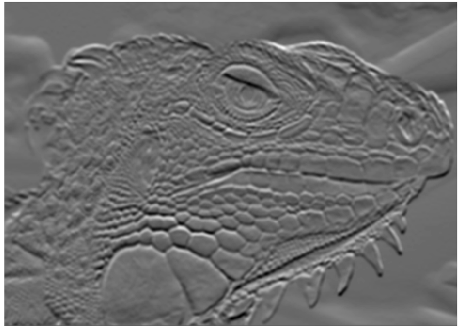
$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Source: Juan C. Niebles and Ranjay Krishna.

Compute gradients (DoG)



X-Derivative of Gaussian



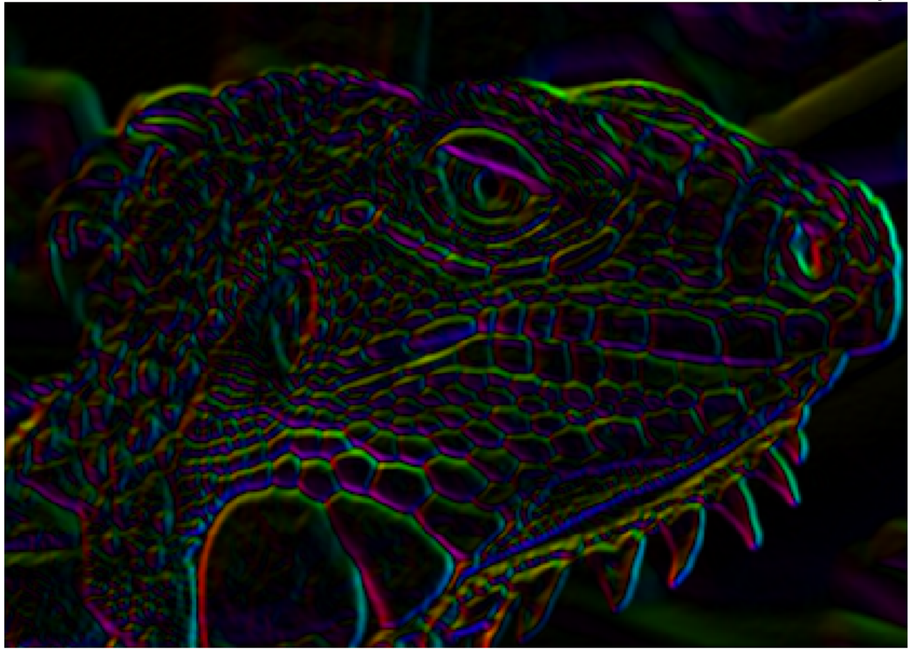
Y-Derivative of Gaussian

Gradient Magnitude

Source: J. Hays

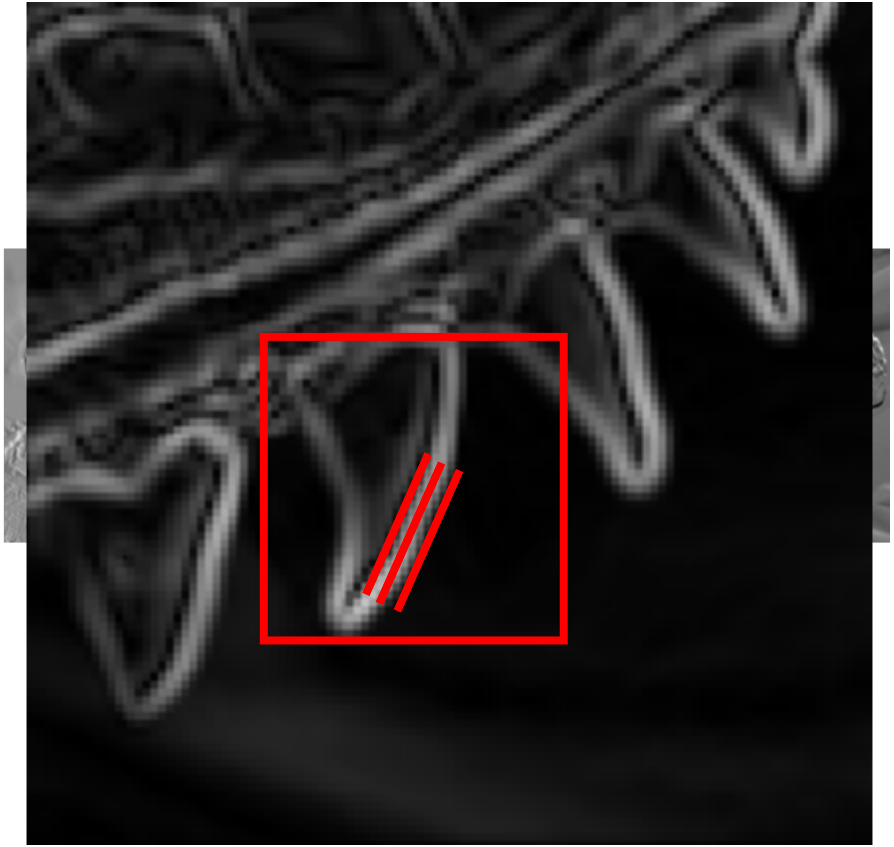
Get orientation at each pixel

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



Source: J. Hays

Compute gradients (DoG)



Gradient Magnitude

Source: Juan C. Niebles and Ranjay Krishna.

Canny edge detector

- Suppress Noise
- Compute gradient magnitude and direction
- Apply Non-Maximum Suppression
  - Assures minimal response

Source: Juan C. Niebles and Ranjay Krishna.

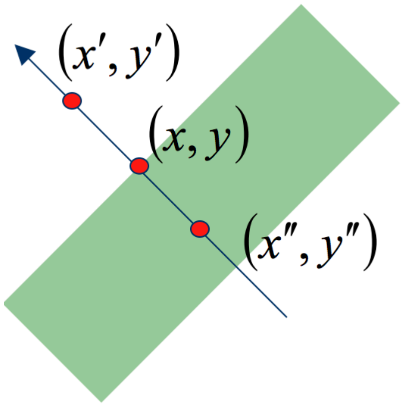
Non-maximum suppression

- Edge occurs where gradient reaches a maxima
- Suppress non-maxima gradient even if it passes threshold
- Only eight angle directions possible
  - Suppress all pixels in each direction which are not maxima
  - Do this in each marked pixel neighborhood

Source: Juan C. Niebles and Ranjay Krishna.

Remove spurious gradients

$|\nabla G|(x, y)$ is the gradient at pixel (x, y)



$$M(x,y) = \begin{cases} |\nabla G|(x,y) \text{ if } |\nabla G|(x,y) > |\nabla G|(x',y') \\ \qquad\qquad \& |\nabla G|(x,y) > |\nabla G|(x'',y'') \\ 0 \qquad\qquad \text{otherwise} \end{cases}$$

x' and **x**" are the neighbors of **x** along normal direction to an edge

Alper Yilmaz, Mubarak Shah Fall 2012, UCF

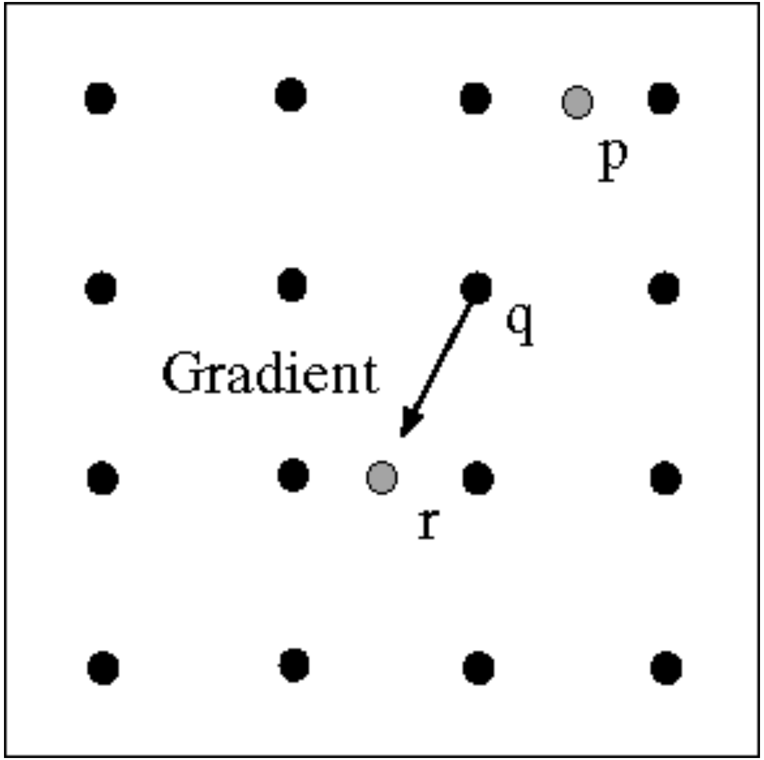
Source: Juan C. Niebles and Ranjay Krishna.

Non-maximum suppression

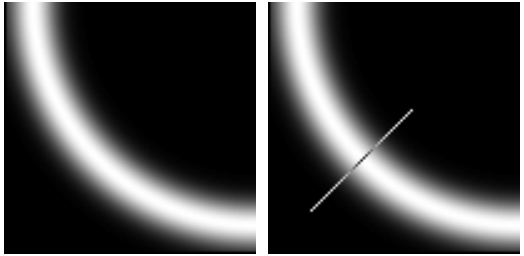
- Edge occurs where gradient reaches a maxima
- Suppress non-maxima gradient even if it passes threshold
- Only eight angle directions possible
  - Suppress all pixels in each direction which are not maxima
  - Do this in each marked pixel neighborhood

Source: Juan C. Niebles and Ranjay Krishna.

Non-maximum suppression



At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.



Source: D. Forsyth

Non-max Suppression

Before

After

Source: Juan C. Niebles and Ranjay Krishna.

Canny edge detector

- Suppress Noise
- Compute gradient magnitude and direction
- Apply Non-Maximum Suppression
  - Assures minimal response
- Use hysteresis and connectivity analysis to detect edges

Source: Juan C. Niebles and Ranjay Krishna.

Source: Juan C. Niebles and Ranjay Krishna.

Hysteresis thresholding

- Avoid streaking near threshold value
- Define two thresholds: Low and High
  - If less than Low, not an edge
  - If greater than High, strong edge
  - If between Low and High, weak edge

Source: Juan C. Niebles and Ranjay Krishna.

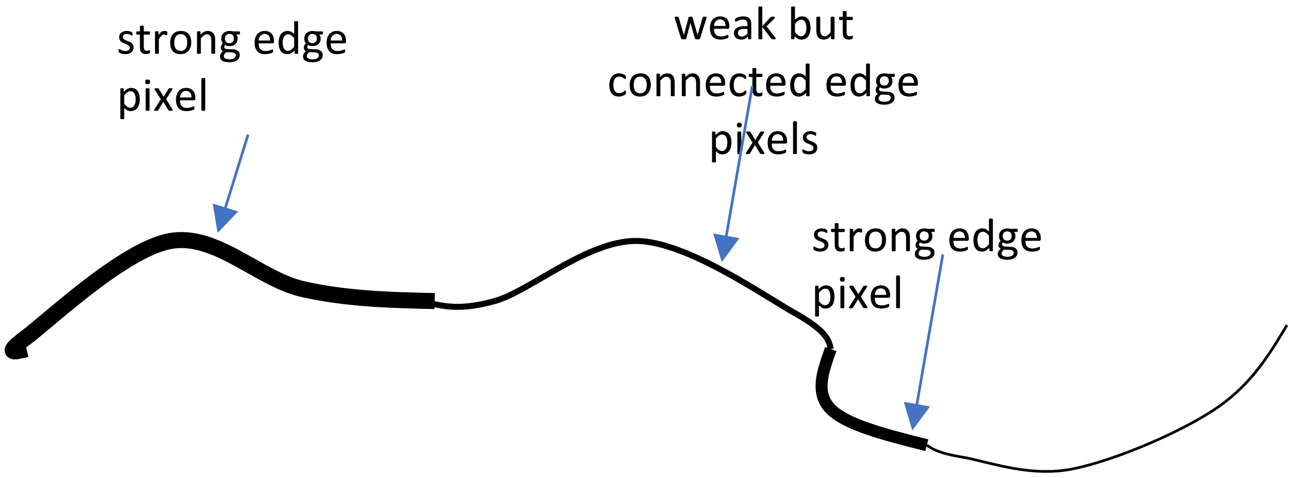
Hysteresis thresholding

If the gradient at a pixel is

- above High, declare it as an ‘strong edge pixel’
- below Low, declare it as a “non-edge-pixel”
- between Low and High
  - Consider its neighbors iteratively then declare it an “edge pixel” if it is connected to an ‘strong edge pixel’ directly or via pixels between Low and High

Source: Juan C. Niebles and Ranjay Krishna.

Hysteresis thresholding



Source: S. Seitz

Final Canny Edges

Source: Juan C. Niebles and Ranjay Krishna.

Canny edge detector

1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - Thin multi-pixel wide “ridges” down to single pixel width
4. Thresholding and linking (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

Source: Juan C. Niebles and Ranjay Krishna.

Canny Edge Detector

- Classic algorithm in Computer Vision / Image Analysis
- Commonly implemented in most libraries

- e.g. in Python you can find it in the skimage package. OpenCV also has an implementation with python bindings.

Corners (and Interest Points)

- How to find corners? What is a corner?

Questions?