Jean Baptiste Joseph Fourier (1768-1830)

had crazy idea (1807

Any univariate function c
be rewritten as a weighte
sum of sines and cosines
different frequencies.

- Don't believe it?
  - Neither did Lagrange, Laplace, Poisson and other big wigs
  - Not translated into English until 1878!
- But it's (mostly) true!
  - called Fourier Series
  - there are some subtle restrictions

...the manner in which the author arrives at these equations is not exempt of difficulties and...his analysis to integrate them still leaves something to be desired on the score of generality and even rigour.

Fourier, Joseph (1768-1830)

French mathematician who discovered that any periodic motion can be written as a superposition of sinusoidal and cosinusoidal vibrations. He developed a mathematical theory of heat in *Théorie Analytique de la Chaleur (Analytic Theory of Heat)*, (1822), discussing it in terms of differential equations.

Fourier was a friend and advisor of Napoleon. Fourier believed that his health would be improved by wrapping himself up in blankets, and in this state he tripped down the stairs in his house and killed himself. The paper of Galois which he had taken home to read shortly before his death was never recovered.

SEE ALSO: Galois

Additional biographies: MacTutor (St. Andrews), Bonn

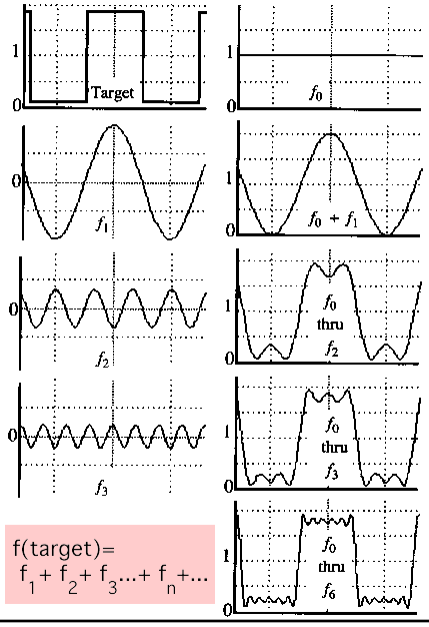
How would math have changed if the Slanket or Snuggie had been invented?

A sum of sines

Our building block:

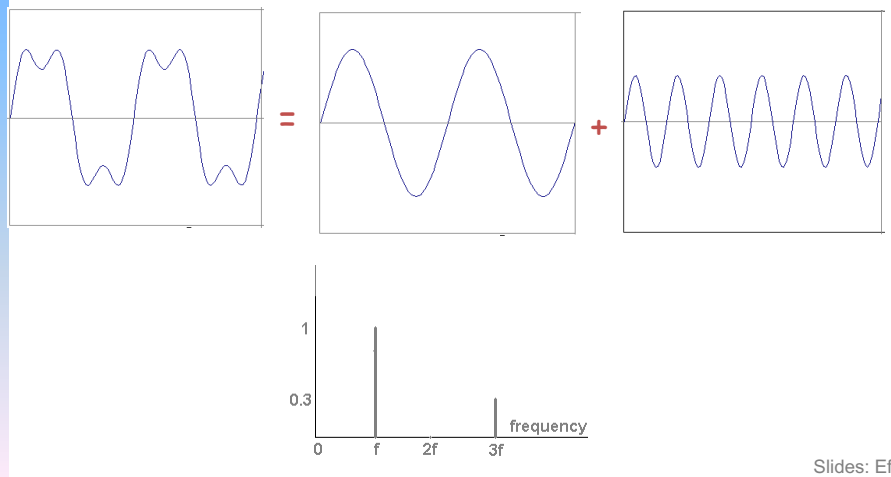
$$A\sin(\omega x + \phi)$$

Add enough of them to get any signal *g(x)* you want!



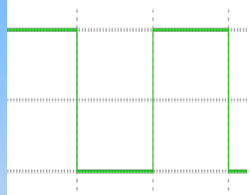
Frequency Spectra

- example : $g(t) = \sin(2\pi f\, t) + (1/3)\sin(2\pi(3f)\, t)$

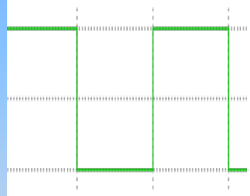


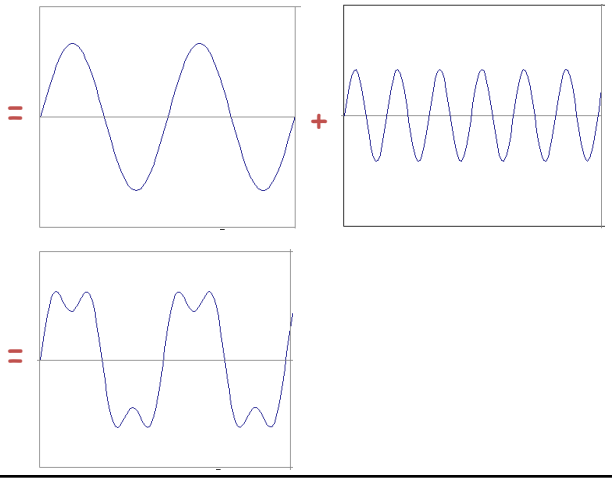
Slides: Efros

Frequency Spectra

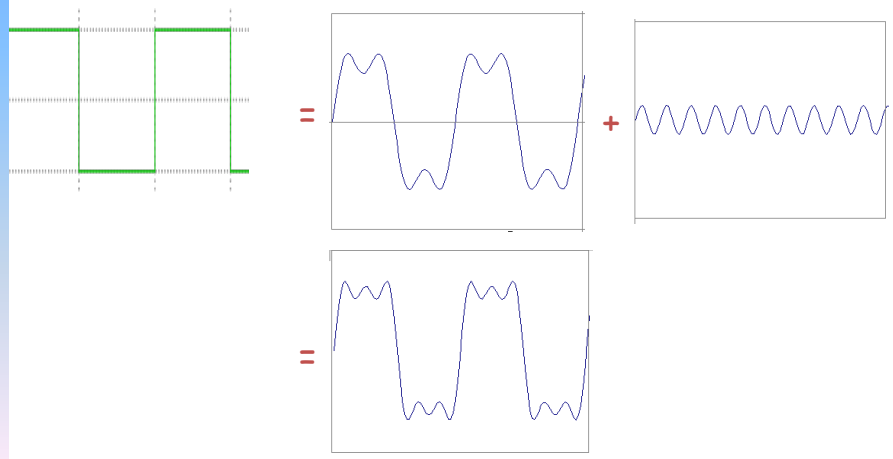


Frequency Spectra

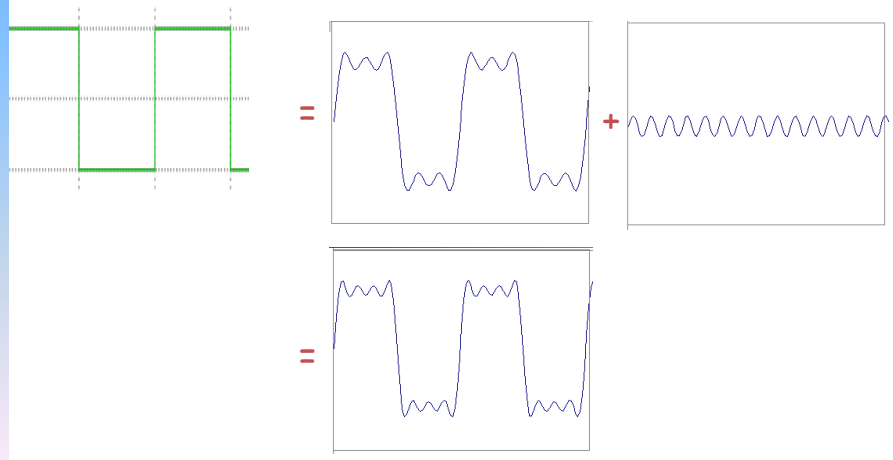




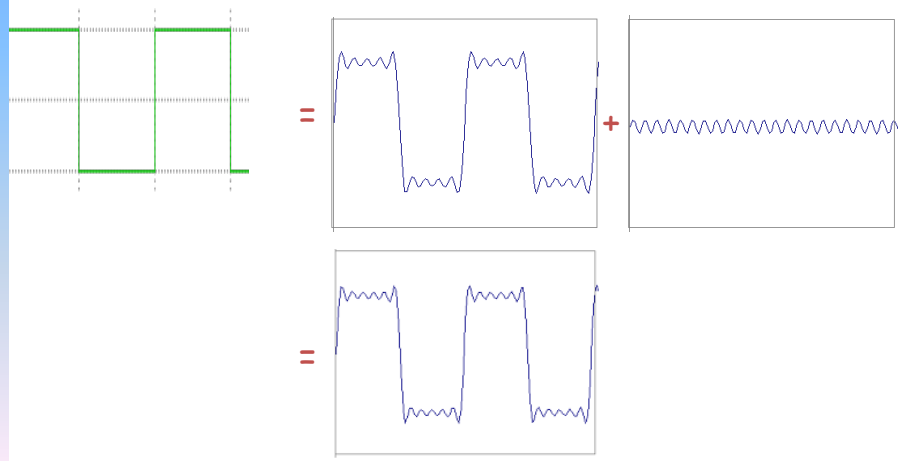
Frequency Spectra



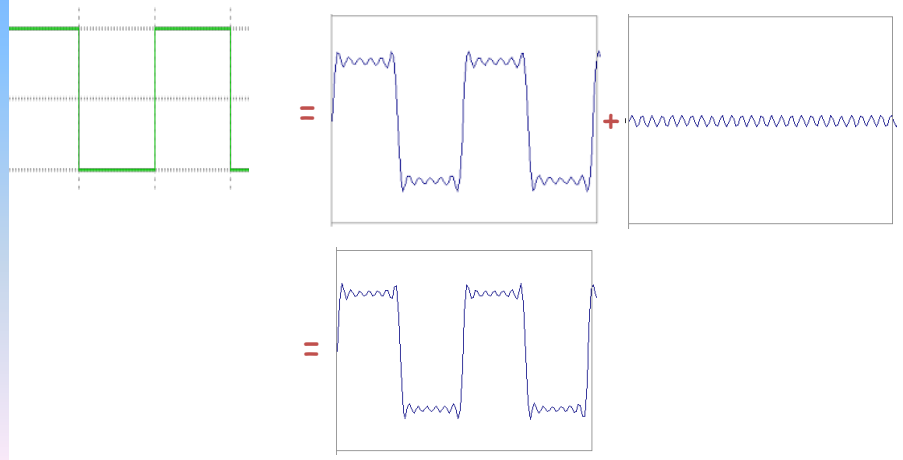
Frequency Spectra



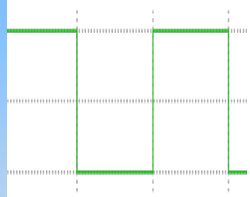
Frequency Spectra



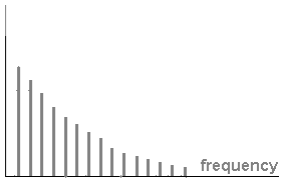
Frequency Spectra



Frequency Spectra

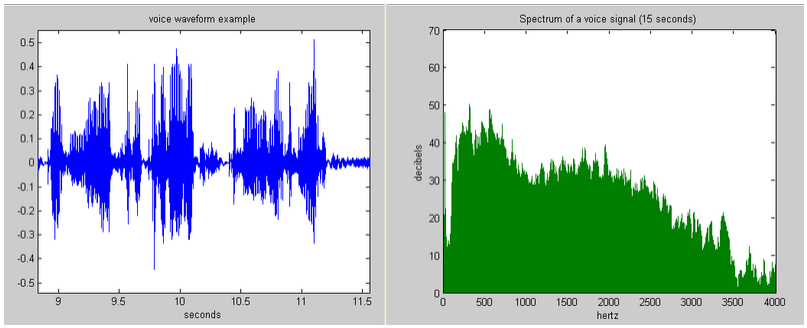


$$= A\sum_{k=1}^{\infty}\frac{1}{k}\sin(2\pi kt)$$



Example: Music

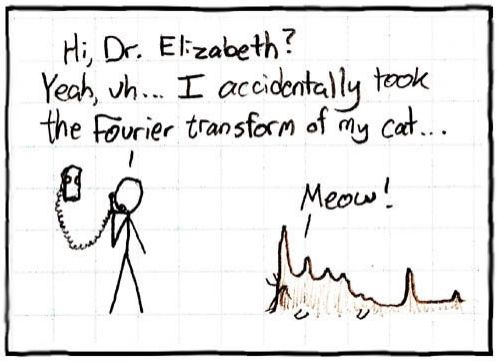
- We think of music in terms of frequencies at different magnitudes



Slide: Hoiem

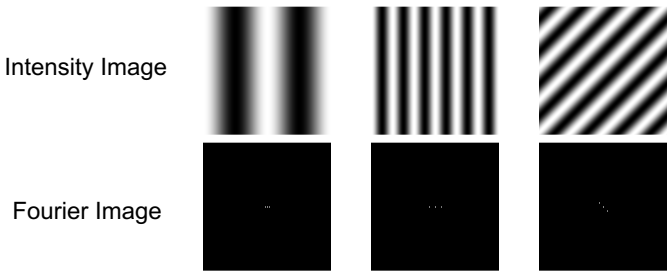
Other signals

- We can also think of all kinds of other signals the same way



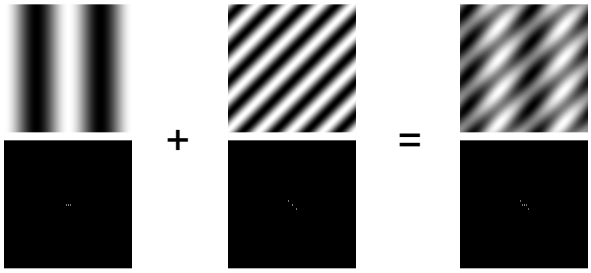
xkcd.com

Fourier analysis in images



http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering

Signals can be composed



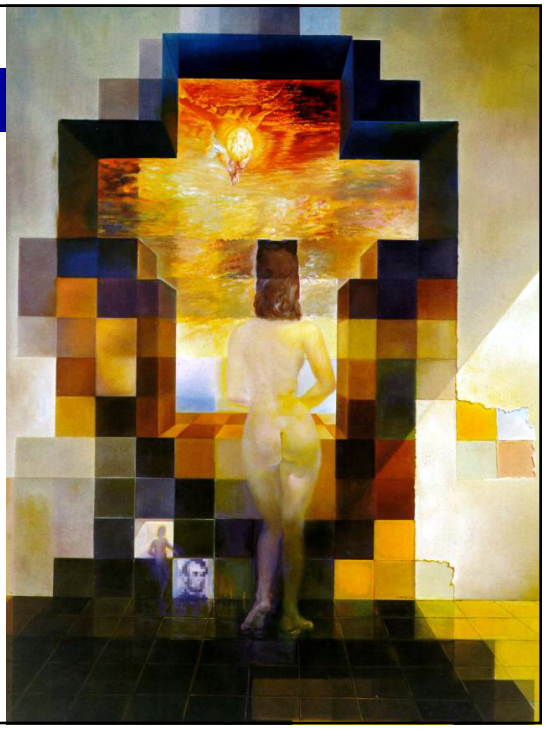
http://sharp.bu.edu/~slehar/fourier/fourier.html#filtering
More: http://www.cs.unm.edu/~brayer/vision/fourier.html

Fourier Transform

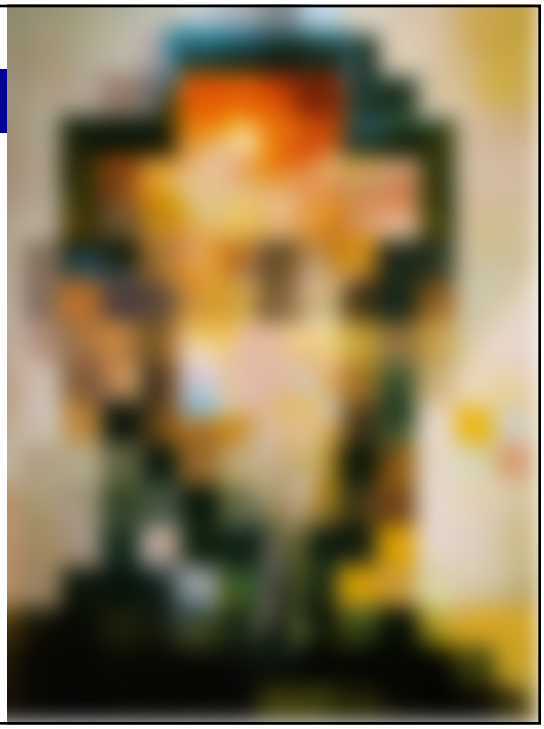
- Fourier transform stores the magnitude and phase at each frequency
  - Magnitude encodes how much signal there is at a particular frequency
  - Phase encodes spatial information (indirectly)
  - For mathematical convenience, this is often notated in terms of real and complex numbers

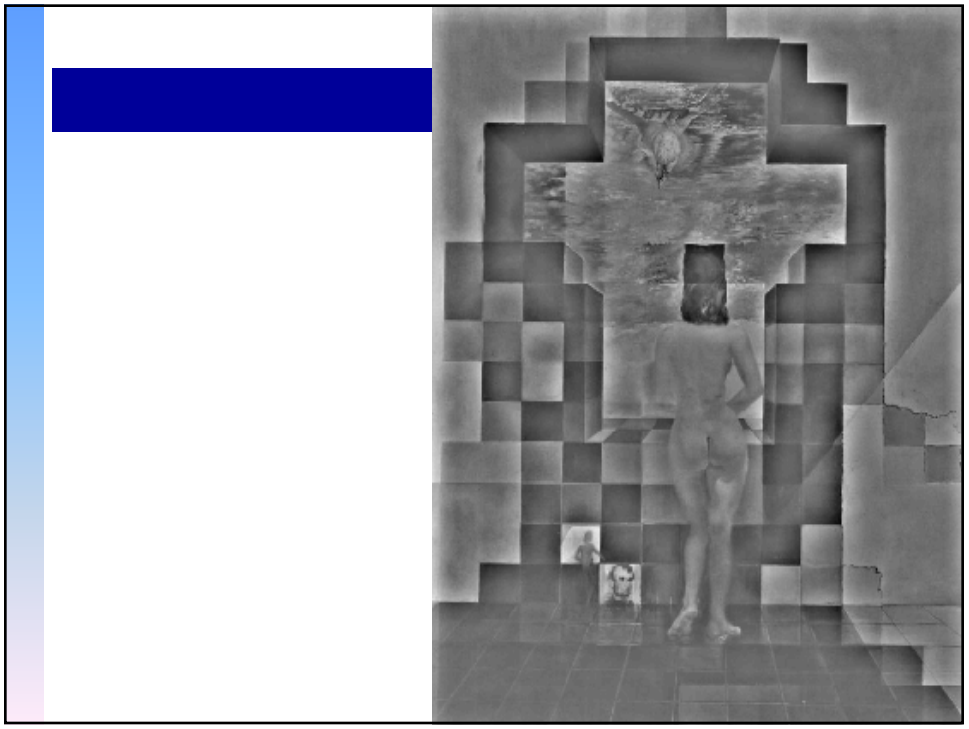
Amplitude: $A = \pm\sqrt{R(\omega)^2 + I(\omega)^2}$ Phase: $\phi = \tan^{-1}\frac{I(\omega)}{R(\omega)}$

Salvador Dali invented Hybrid Images?



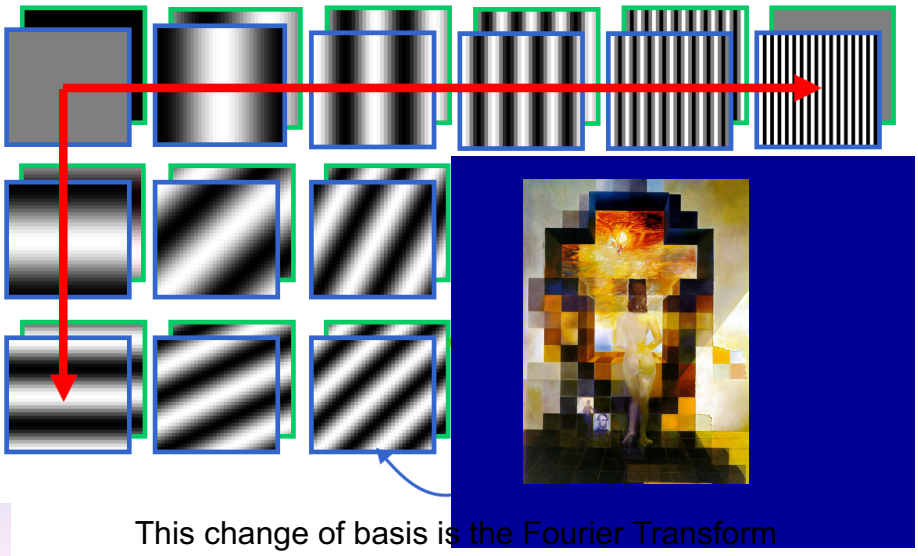
Salvador Dali
"Gala Contemplating the Mediterranean Sea, which at 30 meters becomes the portrait of Abraham Lincoln", 1976





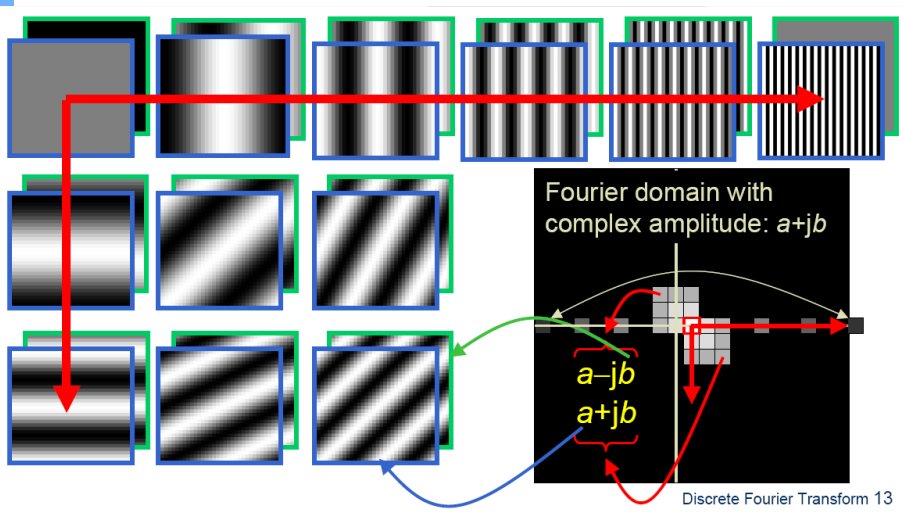
Fourier Bases

Teases away fast vs. slow changes in the image.



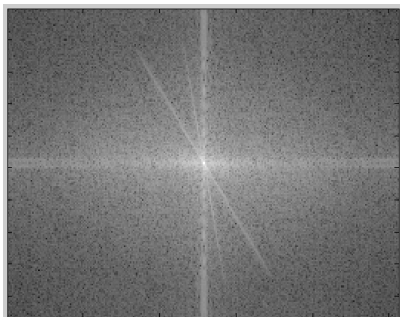
This change of basis is the Fourier Transform

Fourier Bases



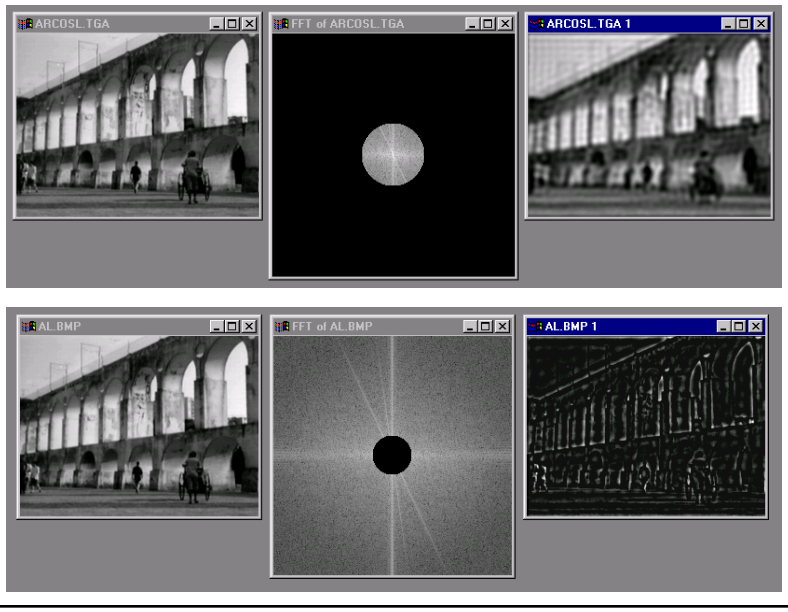
in Matlab, check out: imagesc(log(abs(fftshift(fft2(im)))));

Man-made Scene



Can change spectrum, then reconstruct

Low and High Pass filtering



Computing the Fourier Transform

$$H(\omega) = \mathcal{F}\{h(x)\} = Ae^{j\phi}$$

Continuous

$$H(\omega) = \int_{-\infty}^{\infty} h(x)e^{-j\omega x}dx$$

Discrete

$$H(k) = \frac{1}{N}\sum_{x=0}^{N-1} h(x)e^{-j\frac{2\pi kx}{N}}$$ *k = -N/2..N/2*

Fast Fourier Transform (FFT): NlogN

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!**

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

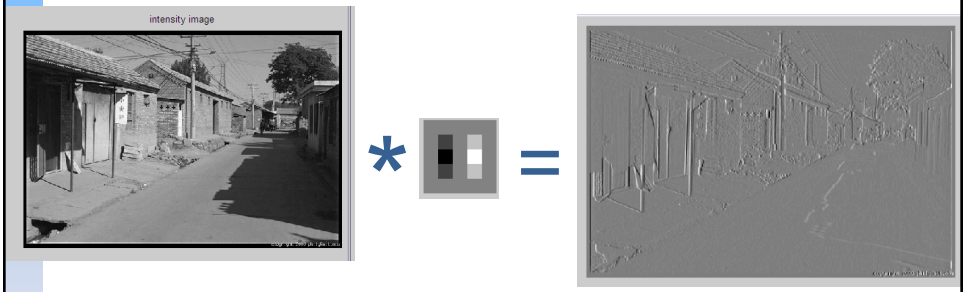
Properties of Fourier Transforms

- **Linearity** $\mathcal{F}[ax(t) + by(t)] = a\mathcal{F}[x(t)] + b\mathcal{F}[y(t)]$
- **Fourier transform of a real signal is symmetric about the origin**
- **The energy of the signal is the same as the energy of its Fourier transform**

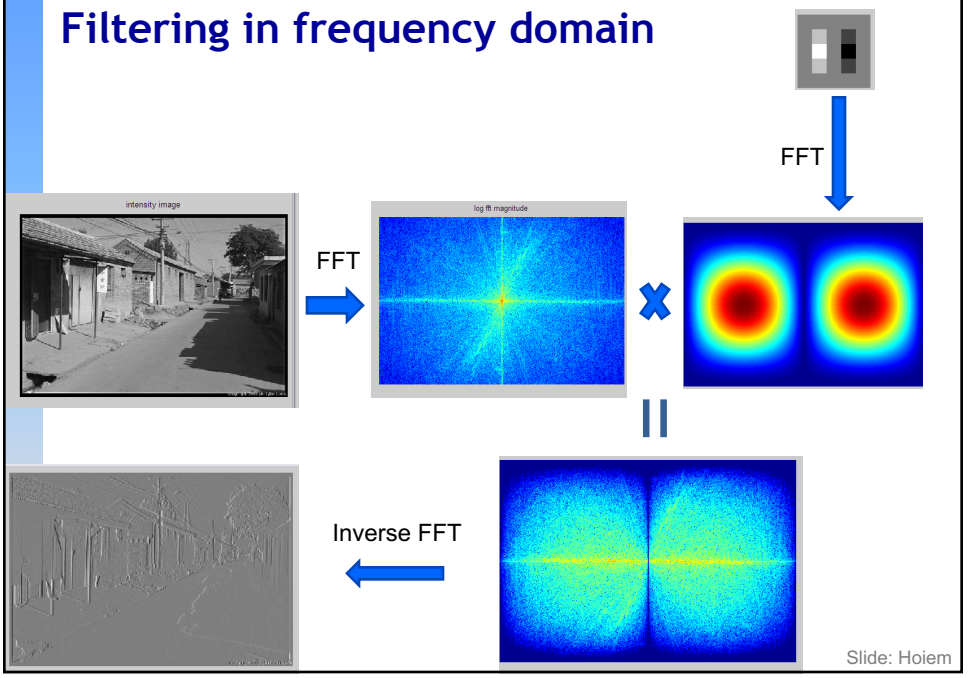
See Szeliski Book (3.4)

Filtering in spatial domain

1	0	-1
2	0	-2
1	0	-1



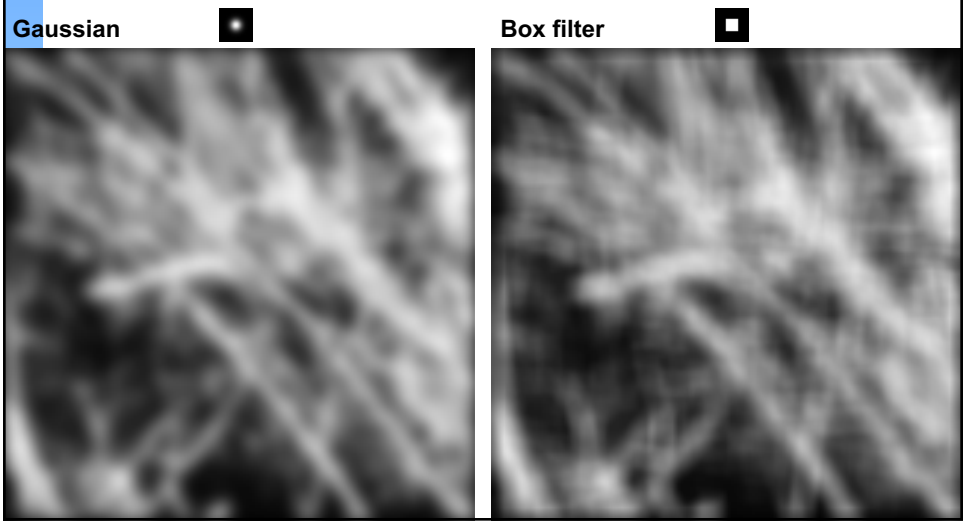
Filtering in frequency domain



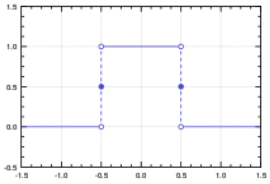
Slide: Hoiem

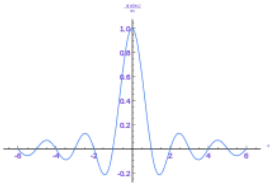
Filtering

Why does the Gaussian give a nice smooth image, but the square filter give edgy artifacts?



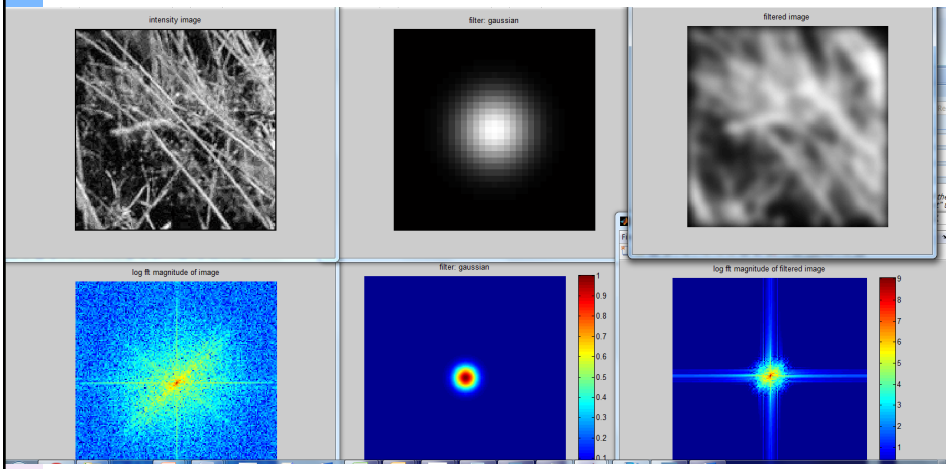
Sinc Filter



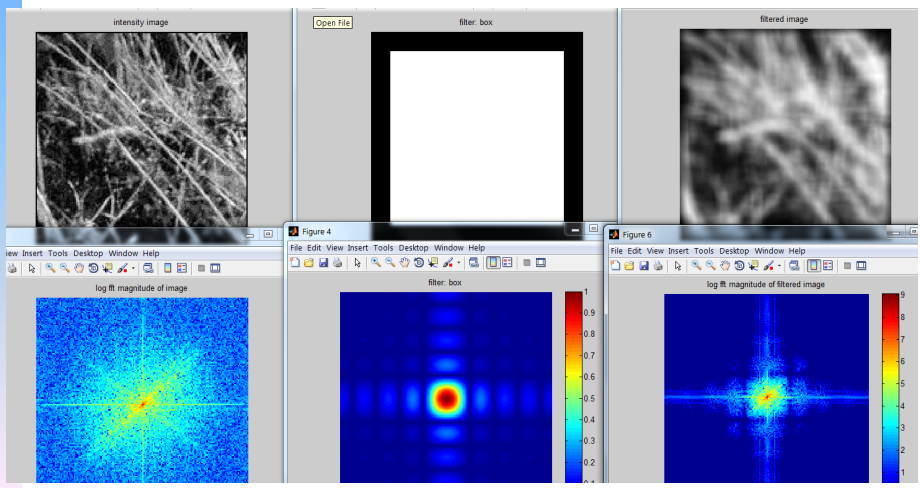


Frequency Domain Spatial Domain

Gaussian



Box Filter

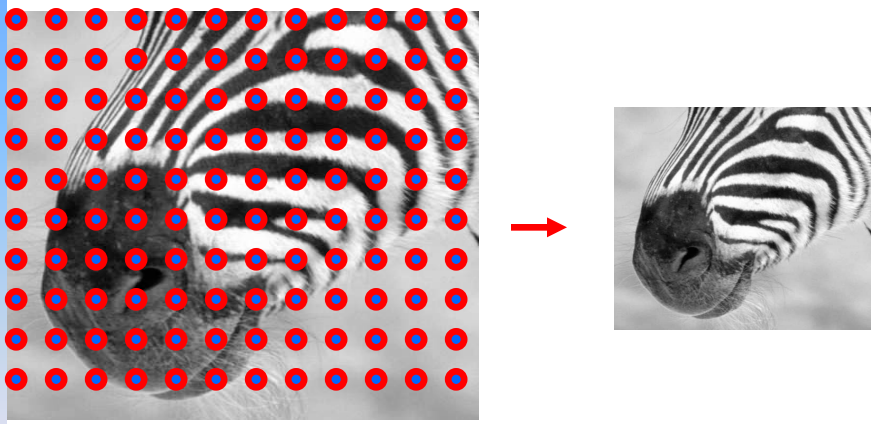


Sampling

Why does a lower resolution image still make sense to us? What do we lose?

Image: http://www.flickr.com/photos/igorms/136916757/

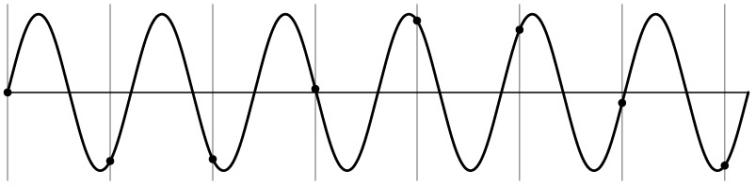
Subsampling by a factor of 2



Throw away every other row and column to create a 1/2 size image

Aliasing problem

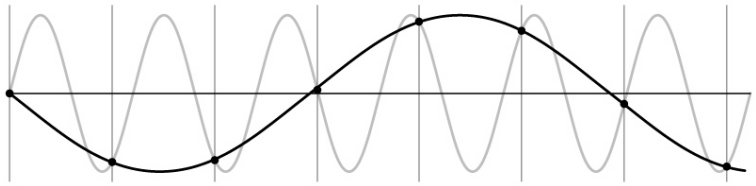
- **1D example (sinewave):**



Source: S. Marschner

Aliasing problem

- 1D example (sinewave):



Source: S. Marschner

Aliasing problem

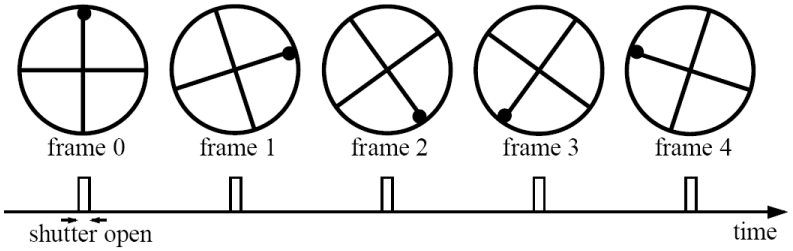
- Sub-sampling may be dangerous….
- Characteristic errors may appear:
  - "car wheels rolling the wrong way in movies"
  - "Checkerboards disintegrate in ray tracing"
  - "Striped shirts look funny on color television"

Source: D. Forsyth

Aliasing in video

Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what's happening.

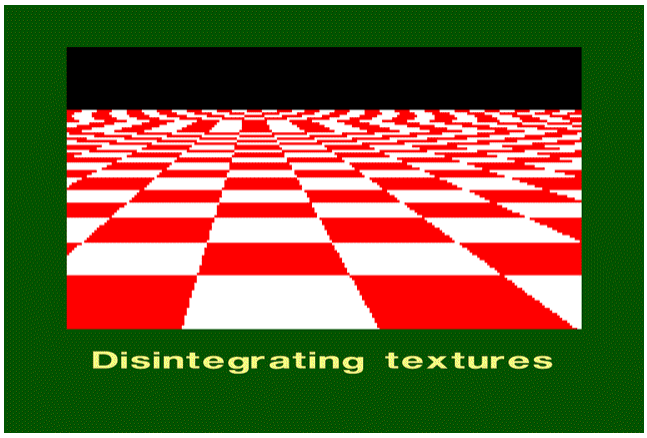
If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):



Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)

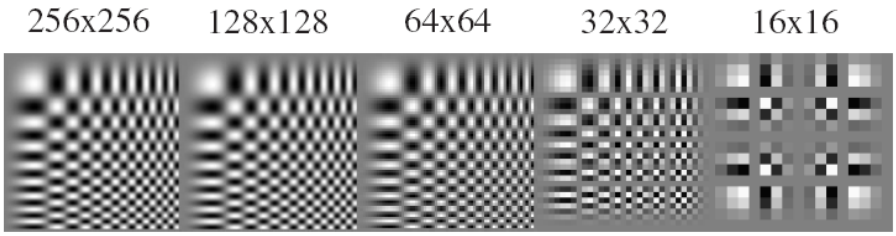
Slide by Steve Seitz

Aliasing in graphics



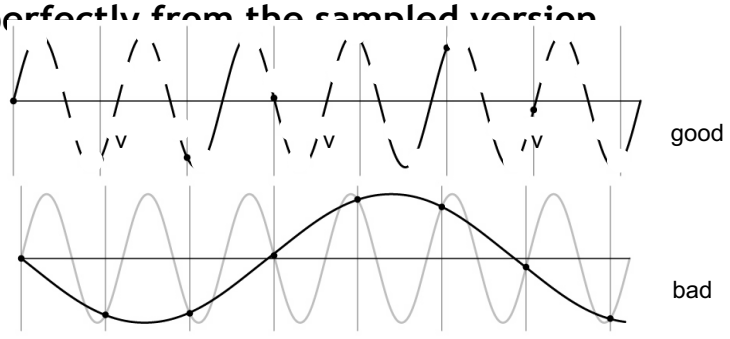
Source: A. Efros

Sampling and aliasing



Nyquist-Shannon Sampling Theorem

- When sampling a signal at discrete intervals, the sampling frequency must be $\geq 2 \times f_{max}$
- f_{max} = max frequency of the input signal
- This will allows to reconstruct the original perfectly from the sampled version



Anti-aliasing

Solutions:

- **Sample more often**
- **Get rid of all frequencies that are greater than half the new sampling frequency**
  - **Will lose information**
  - **But it's better than aliasing**
  - **Apply a smoothing filter**

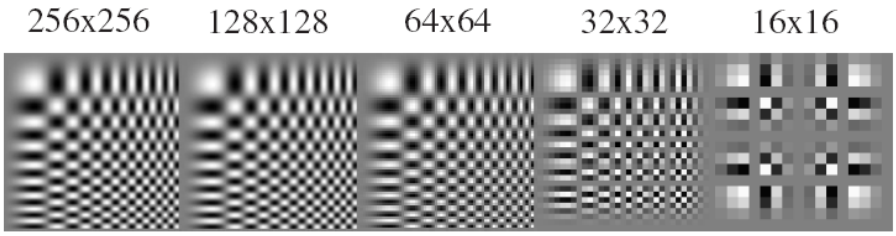
Algorithm for downsampling by factor of 2

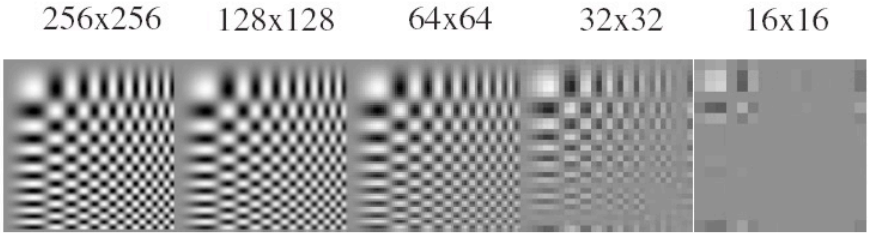
1. Start with image(h, w)
2. Apply low-pass filter

   im_blur = imfilter(image, fspecial('gaussian', 7, 1))
3. Sample every other pixel

   im_small = im_blur(1:2:end, 1:2:end);

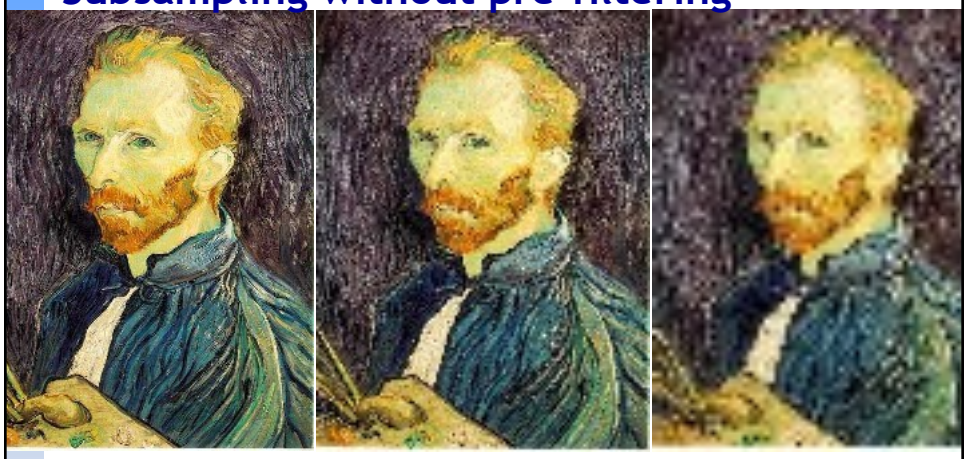
Anti-aliasing





Forsyth and Ponce 2002

Subsampling without pre-filtering



1/2 1/4 (2x zoom) 1/8 (4x zoom)

Slide by Steve Seitz

Subsampling with Gaussian pre-filtering

Gaussian 1/2 G 1/4 G 1/8

Slide by Steve Seitz

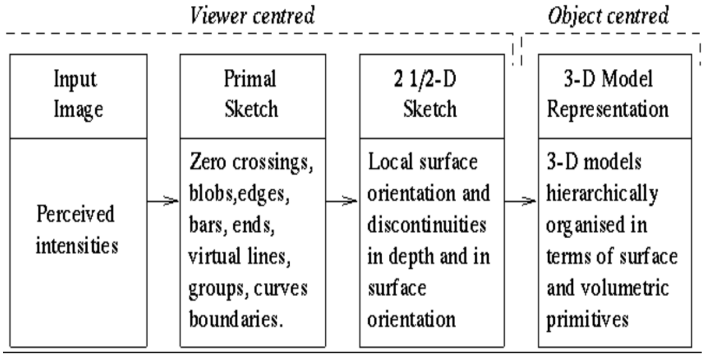
A biref anlaogy wtih txet

- Waht mttares is waht hpapens on wrod baounrdies
- Mocpera iwht htsi sesm

48

Primal Sketch

- Early vision: invariants, moments, pattern recognition
- David Marr, late'70s
  - Inspiration from biological vision
  - Image representation in terms of `sketch'



49

Primal Sketch

- Julesz, `Textons, the fundamental Elements of visual perception'
  - boundaries between different texture patterns can be observed pre-attentively
- Sketch components
  - Edges
  - Ridges
  - Blobs
  - Junctions

50

Feature Extraction

- Features: local meaningful detectable
  - Points
  - Edges
    - Step edges
    - Line edges
  - Contours
    - Closed contours are boundaries
  - Regions

Goals for low-level image representation

- Compact

  00000001111111000001111111111000 =

  -6- 01 -3- 10 -4- 01 -7- 10

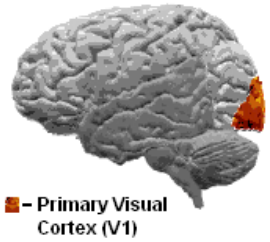
- Generic
  - Same for all tasks, objects, images considered

- Sufficient
  - No need to look back into the image

52

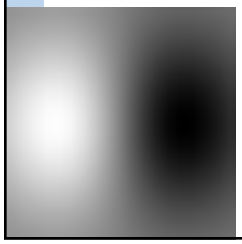
Purpose

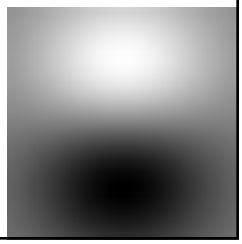
- Extract compact, generic, representation of image that carries sufficient information for higher-level processing tasks
- Essentially what area V1 does in our visual cortex.



Filters are templates

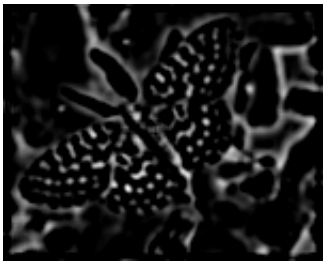
- Applying a filter at some point can be seen as taking a dot-product between the image and some vector
- Filtering the image is a set of dot products
- Insight
  - filters look like the effects they are intended to find
  - filters find effects they look like

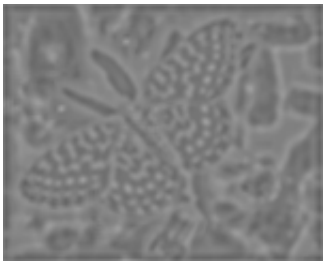


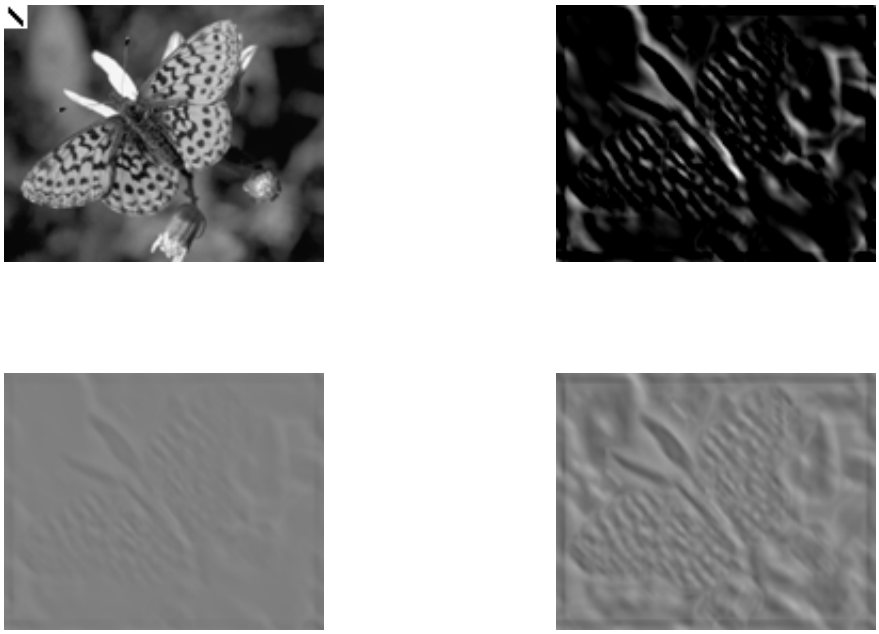


Normalized correlation

- Think of filters of a dot product
  - now measure the angle
  - i.e normalised correlation output is filter output, divided by root sum of squares of values over which filter lies
- Tricks:
  - ensure that filter has a zero response to a constant region (helps reduce response to irrelevant background)
  - subtract image average when computing the normalizing constant (i.e. subtract the image mean in the neighbourhood)
  - absolute value deals with contrast reversal





Edge Detection Schemes

- The basic approach to edge detection is to compute "spatial derivatives" of the intensity image
- The act of taking spatial derivatives is usually approximated by convolution

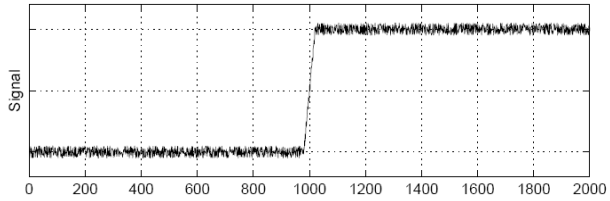
Gradients and edges

- Points of sharp change in an image are interesting:
  - change in reflectance
  - change in object
  - change in illumination
  - noise
- Sometimes called edge points

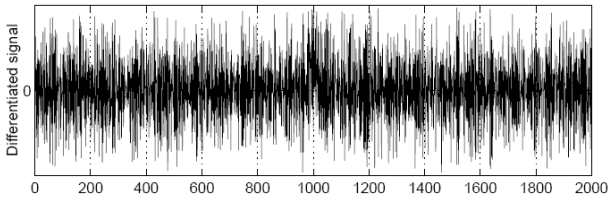
- General strategy
  - determine image gradient
  - now mark points where gradient magnitude is particularly large wrt neighbours (ideally, curves of such points).

Finding Edges

- Consider a 1D signal: `I(x)`



- Obvious strategy: look for maxima/minima in `I'(x)`:

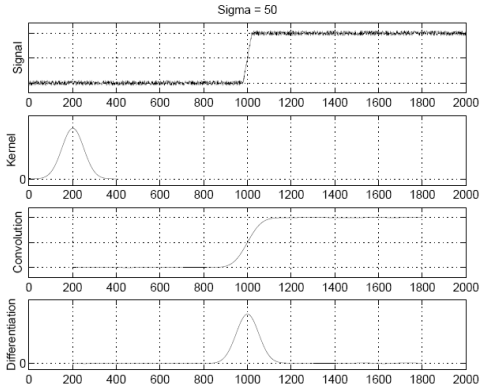


- Need to smooth to suppress noise

Slide: R. Cipolla, S. Seitz

1-D Edge Detection

- Convolve the 1-D signal with a Gaussian kernel to give s(x).
- Compute derivate of resulting smoothed signal to give s'(x).
- Find maxima/minima of s'(x)
- Threshold



$$g_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma^2}\right)$$

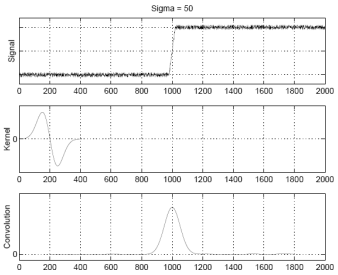
$$s(x) = g_\sigma(x) * I(x) = \int_{-\infty}^{+\infty} g_\sigma(u) I(x-u)\, du = \int_{-\infty}^{+\infty} g_\sigma(x-u) I(u)\, du$$

Slide: R. Cipolla, S. Seitz

1-D Edge Detection

- Note that

$$s'(x) = \frac{d}{dx}\left[g_\sigma(x) * I(x)\right] = g'_\sigma(x) * I(x)$$



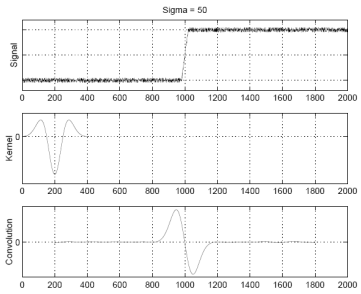
$g_\sigma(x)$ $g'_\sigma(x)$

$$s''(x) = g''_\sigma(x) * I(x)$$

Slide: R. Cipolla, S. Seitz

1-D Edge Detection

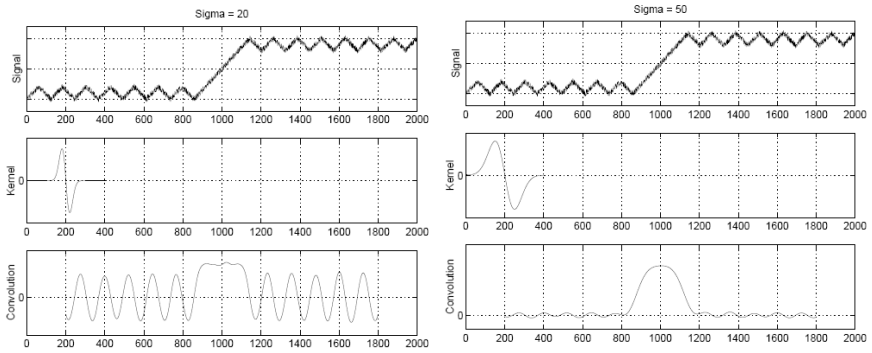
- Looking for maxima/minima of s'(x) is same as zero-crossings of s''(x), so easier to convolve with Laplacian of Gaussian, g_σ''(x):



Slide: R. Cipolla, S. Seitz

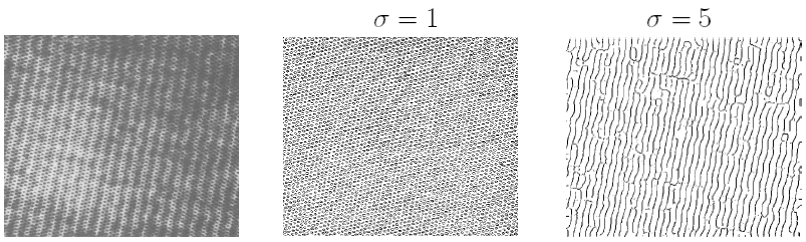
Multi-scale 1D edge detection

- As σ increases, the smoothing increases and only the central edge survives.



Multi-scale Edge Detection

- Degree of smoothing controls the scale at which we analyze the image
- Image of a dish cloth:



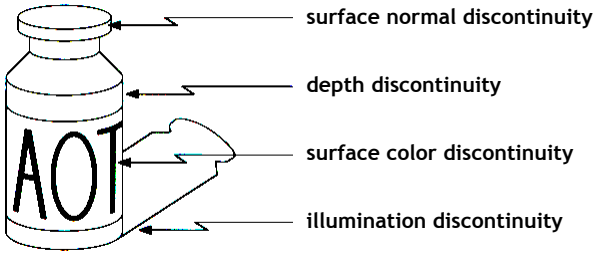
- Fine scales are sensitive to noise

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

Origin of Edges

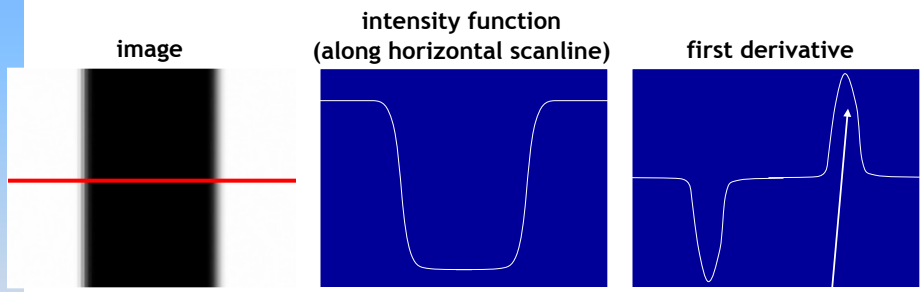


- Edges are caused by a variety of factors

Source: Steve Seitz

Characterizing edges

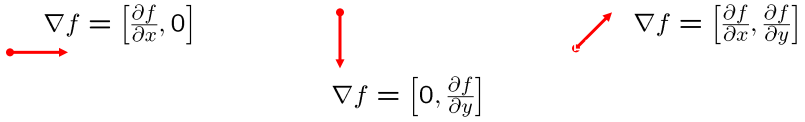
- An edge is a place of rapid change in the image intensity function



edges correspond to
extrema of derivative

Image gradient

- The gradient of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$



The gradient points in the direction of most rapid increase in intensity

The gradient direction is given by $\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$

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

The *edge strength* is given by the gradient magnitude

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

Source: Steve Seitz

Differentiation and convolution

- Recall, for 2D function, f(x,y):

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

- This is linear and shift invariant, so must be the result of a convolution.

- We could approximate this as

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

- (which is obviously a convolution)

-1	1

Source: D. Forsyth, D. Lowe

Finite difference filters

- Other approximations of derivative filters exist:

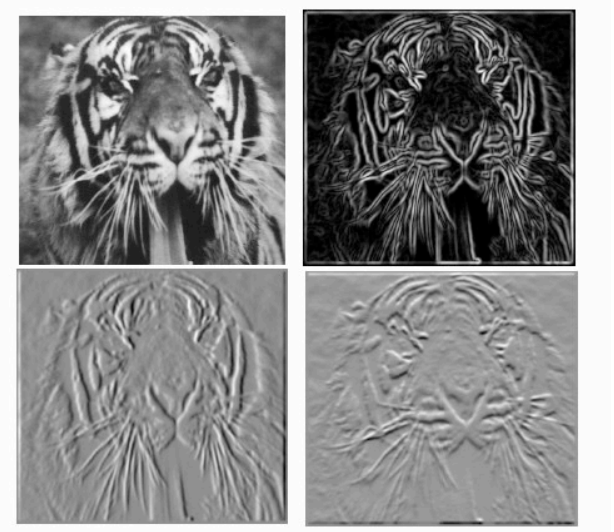
Prewitt: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

Source: K. Grauman

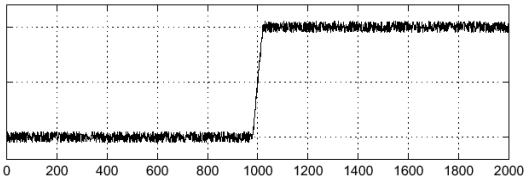
Finite differences: example



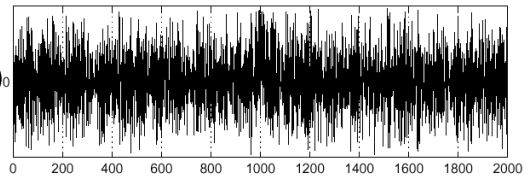
- Which one is the gradient in the x-direction (resp. y-direction)?

Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal



$\frac{d}{dx} f(x)$



Where is the edge?

Source: S. Seitz

Effects of noise

- Finite difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What is to be done?

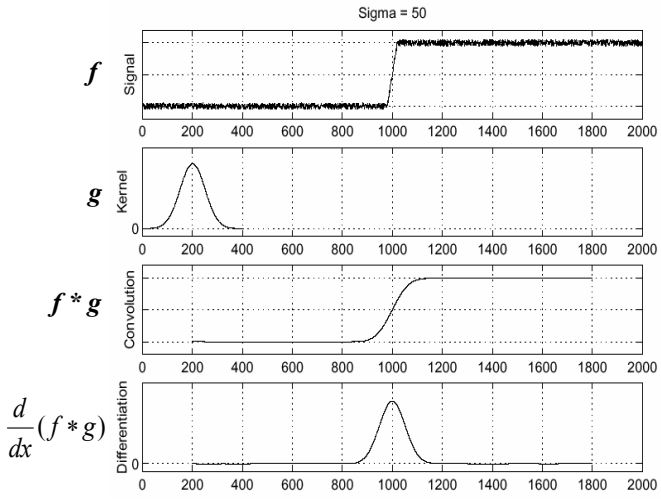
Source: D. Forsyth

Effects of noise

- Finite difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What is to be done?
  - Smoothing the image should help, by forcing pixels different to their neighbors (=noise pixels?) to look more like neighbors

Source: D. Forsyth

Solution: smooth first



- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

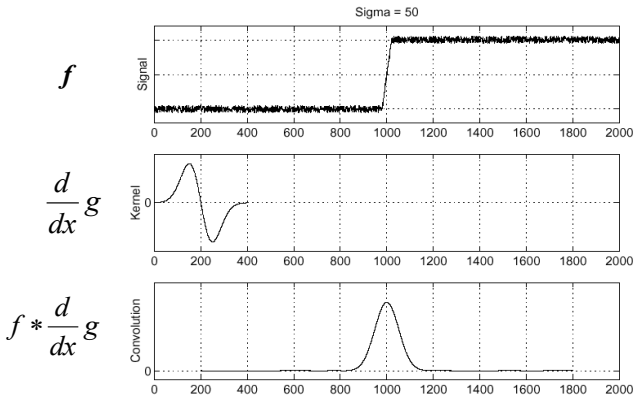
Source: S. Seitz

Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative:

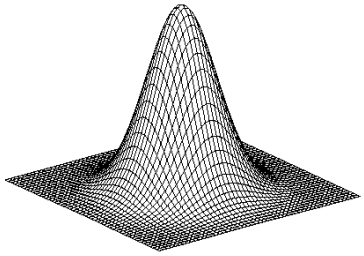
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- This saves us one operation:

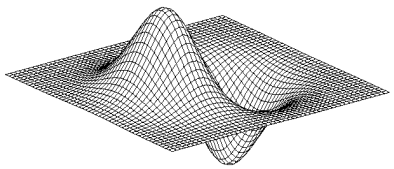


Source: S. Seitz

Derivative of Gaussian filter

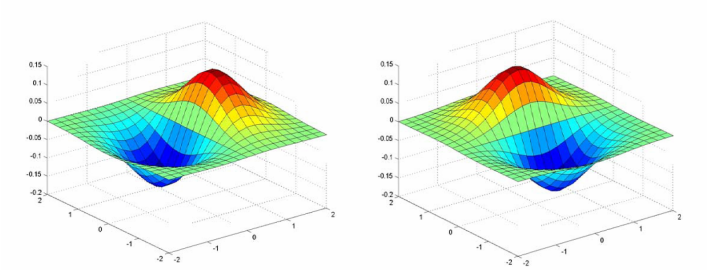


* [1 -1] =



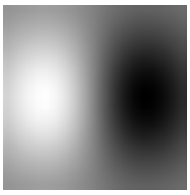
- Is this filter separable?

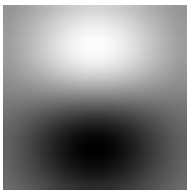
Derivative of Gaussian filter



x-direction

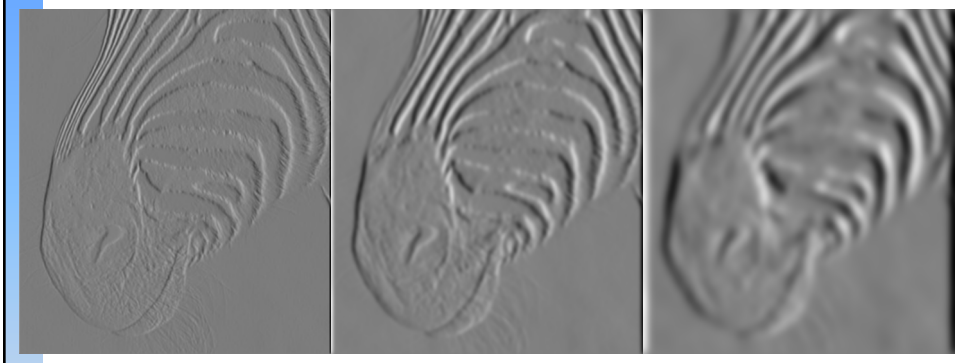
y-direction





- Which one finds horizontal/vertical edges?

Tradeoff between smoothing and localization



1 pixel

3 pixels

7 pixels

- Smoothed derivative removes noise, but blurs edge. Also finds edges at different "scales".

Source: D. Forsyth

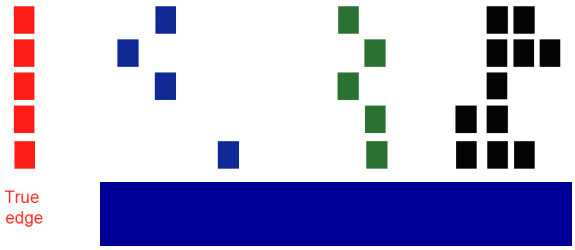
Implementation issues

- The gradient magnitude is large along a thick "trail" or "ridge," so how do we identify the actual edge points?
- How do we link the edge points to form curves?

Source: D. Forsyth

Designing an edge detector

- Criteria for an "optimal" edge detector:
  - Good detection: the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)
  - Good localization: the edges detected must be as close as possible to the true edges
  - Single response: the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge



Source: L. Fei-Fei

Canny edge detector

- This is probably the most widely used edge detector in computer vision
- Theoretical model: step-edges corrupted by additive Gaussian noise
- Canny has shown that the first derivative of the Gaussian closely approximates the operator that optimizes the product of *signal-to-noise ratio* and localization

J. Canny, *A Computational Approach To Edge Detection*, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Source: L. Fei-Fei

Canny edge detector

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

- MATLAB: edge(image, 'canny')

Source: D. Lowe, L. Fei-Fei

Example

- original image (Lena)

Example

norm of the gradient

Example

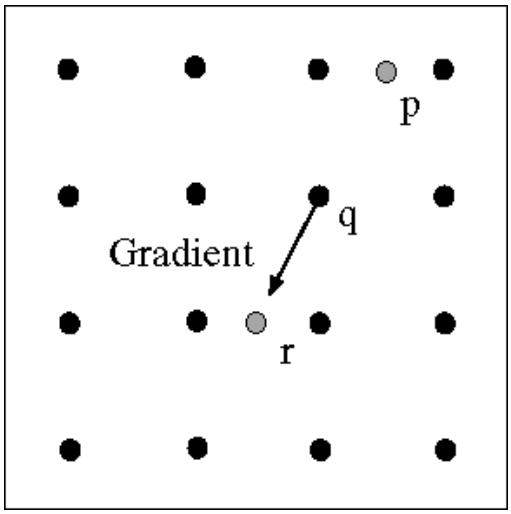
thresholding

Example

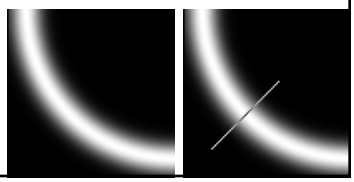
thinning

(non-maximum suppression)

Non-maximum suppression

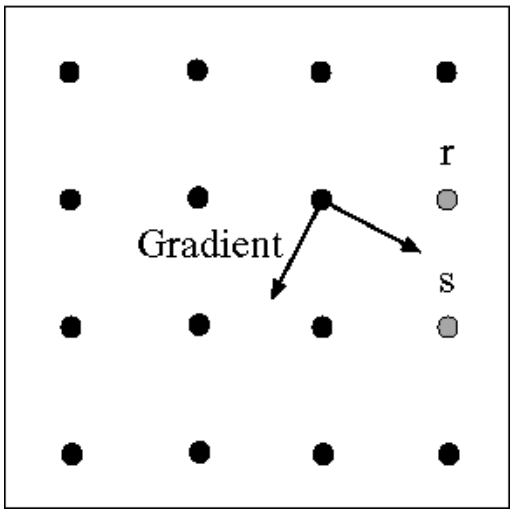


At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.

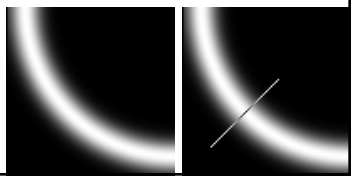


Source: D. Forsyth

Edge linking



Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).



Source: D. Forsyth

Hysteresis thresholding

- Check that maximum value of gradient value is sufficiently large
  - drop-outs? use hysteresis
    - use a high threshold to start edge curves and a low threshold to continue them.

Source: S. Seitz

Hysteresis thresholding

original image

high threshold (strong edges)

low threshold (weak edges)

hysteresis threshold

Source: L. Fei-Fei

Effect of σ (Gaussian kernel spread/size)

original | Canny with $\sigma = 1$ | Canny with $\sigma = 2$

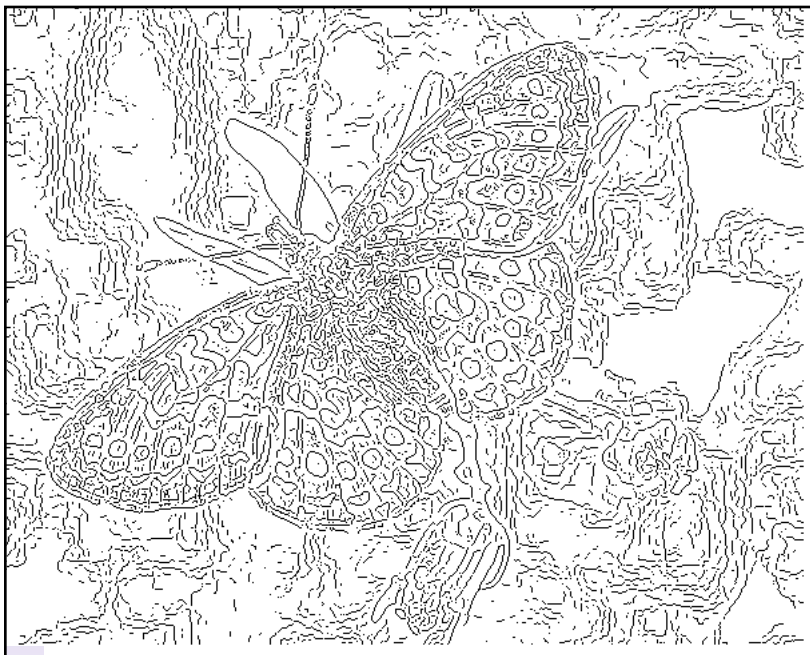
The choice of σ depends on desired behavior

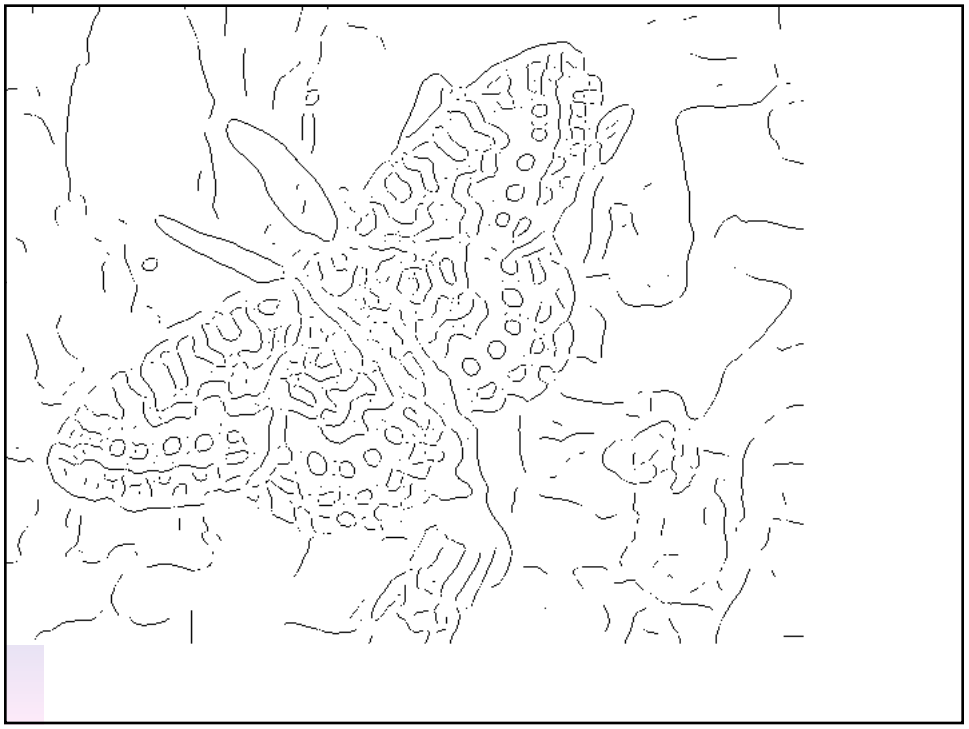
- large σ detects large scale edges
- small σ detects fine features

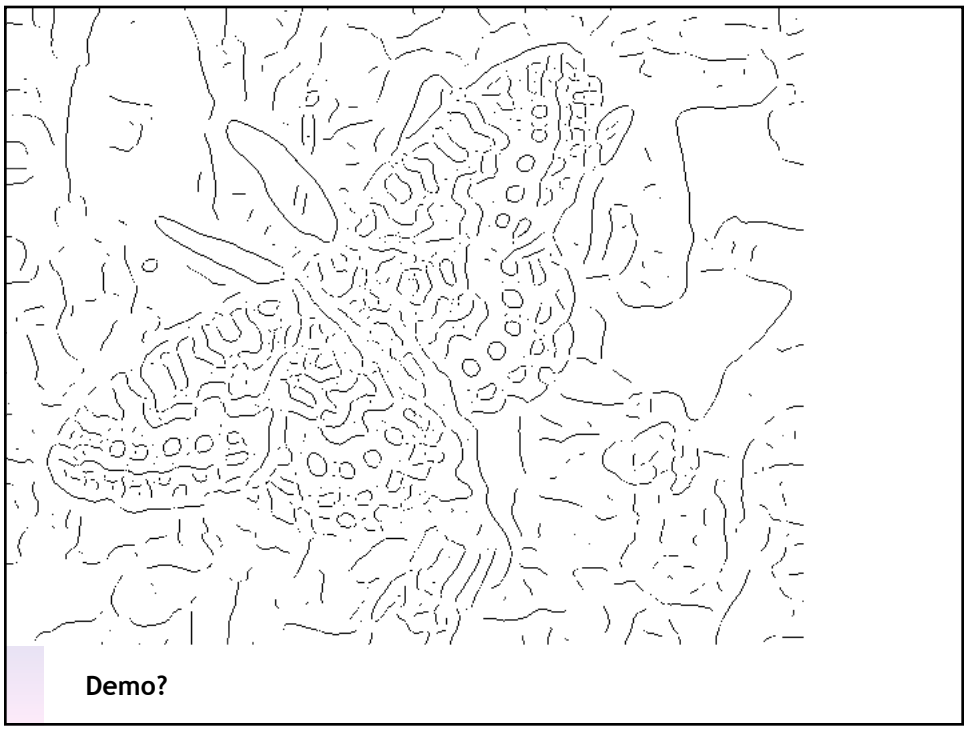
Source: S. Seitz

Notice

- Something nasty is happening at corners
- Scale affects contrast
- Edges aren't bounding contours







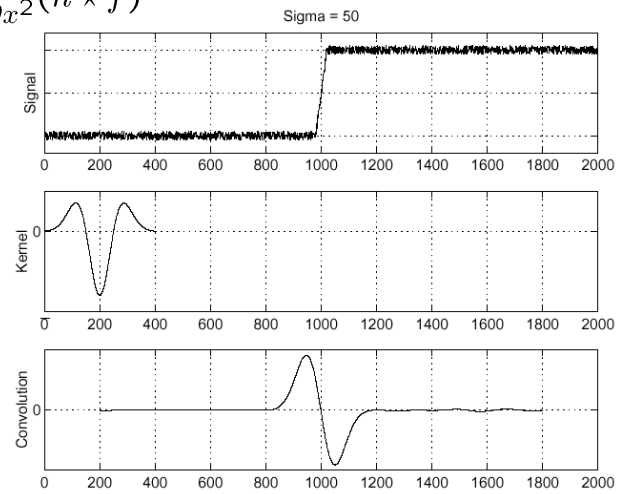
Demo?

2nd Derivative Detector: Laplacian of Gaussian (LoG)

- **Consider** $\frac{\partial^2}{\partial x^2}(h \star f)$

$\frac{\partial^2}{\partial x^2} h$

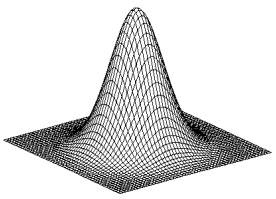
$(\frac{\partial^2}{\partial x^2} h) \star f$



Where is the edge? **Zero-crossings of bottom graph**

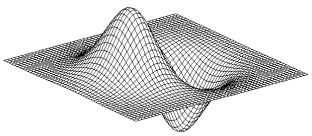
Slide credit: Steve Seitz **B. Leibe**

Summary: 2D Edge Detection Filters



Gaussian

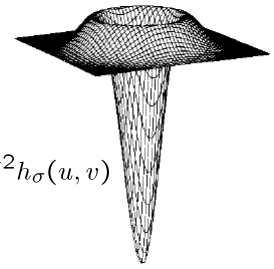
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



Derivative of Gaussian

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

Laplacian of Gaussian



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

- **This is the Laplacian operator:**

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

Slide credit: Kristen Grauman

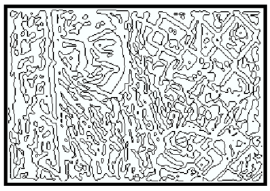
LoG Edge Detector

- The Laplacian-of-Gaussian (LOG) – cont.

I

$I*(\Delta^2 G)$

Zero crossings of $I*(\Delta^2 G)$



From Bedros

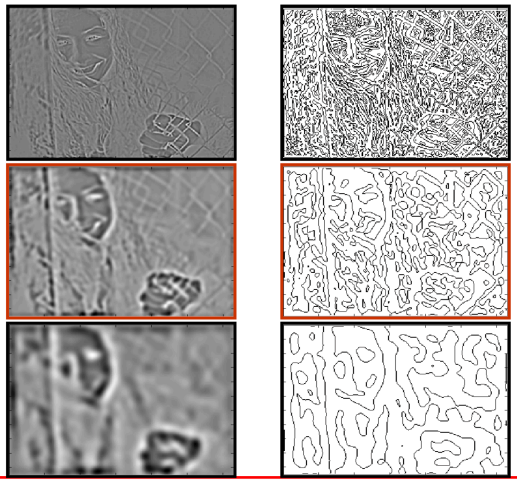
LoG Edge Detector

- The Laplacian-of-Gaussian (LOG) – cont.

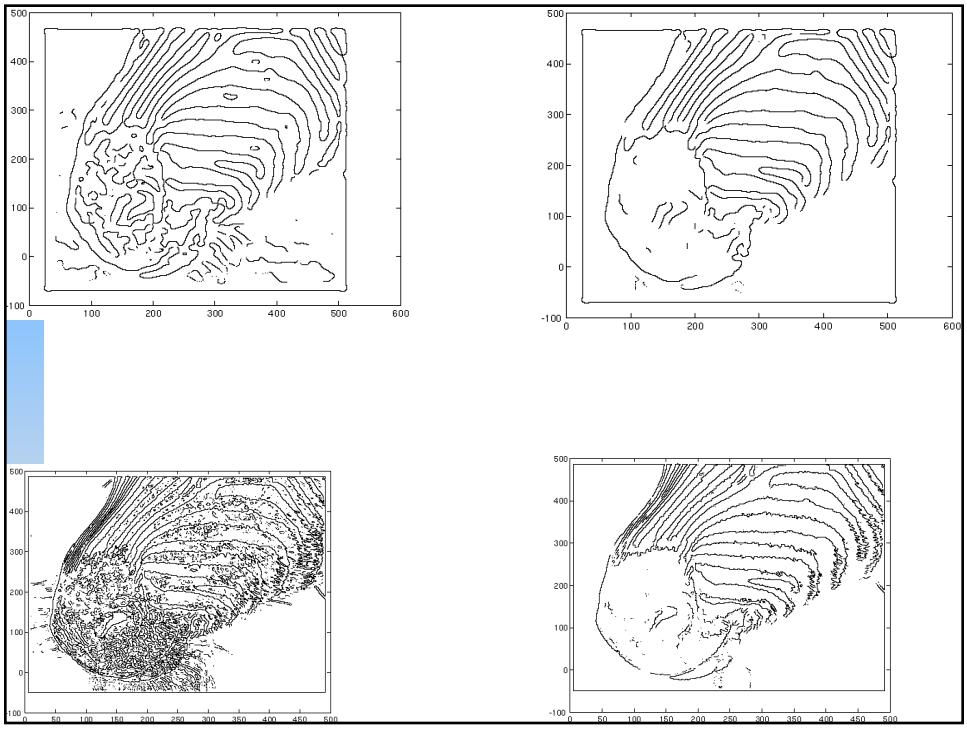
$\sigma = 1$

$\sigma = 3$

$\sigma = 6$



From Bedros



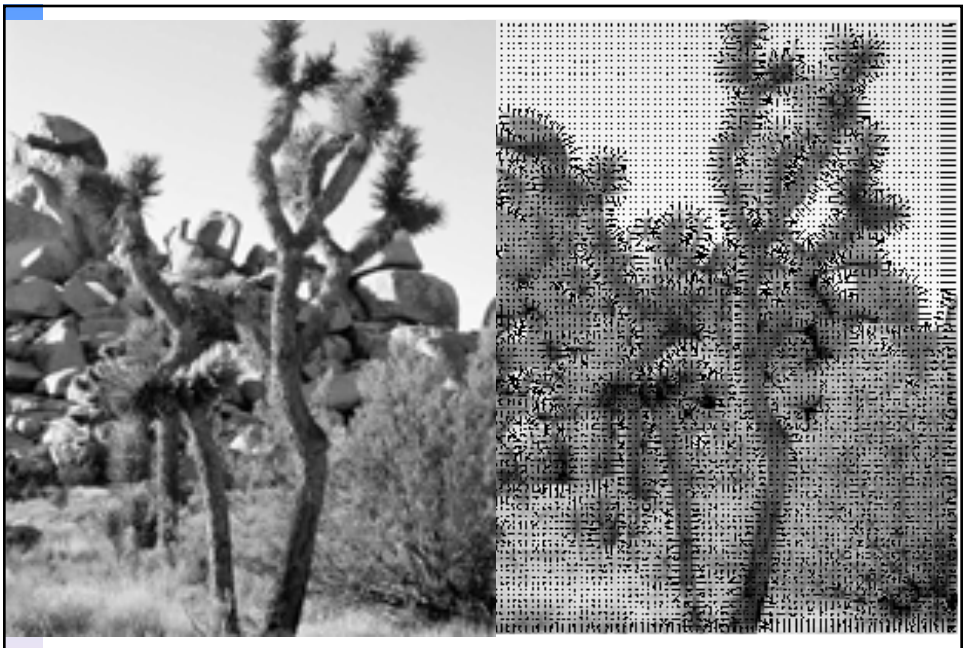
Orientation representations

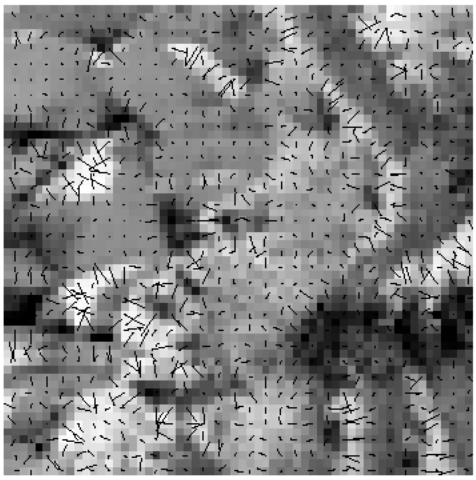
- The gradient magnitude is affected by illumination changes
  - but it's direction isn't
- We can describe image patches by the swing of the gradient orientation
- Important types:
  - constant window
    - small gradient mags
  - edge window
    - few large gradient mags in one direction
  - flow window
    - many large gradient mags in one direction
  - corner window
    - large gradient mags that swing

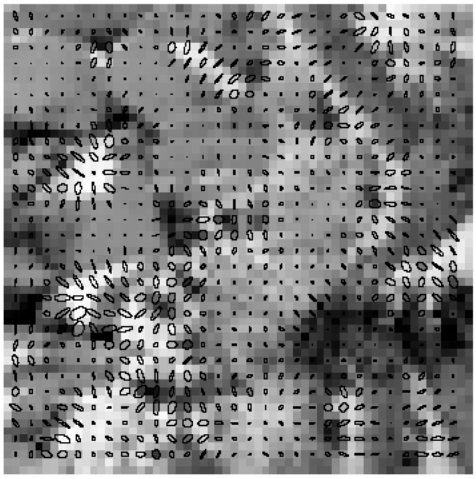
Representing Windows

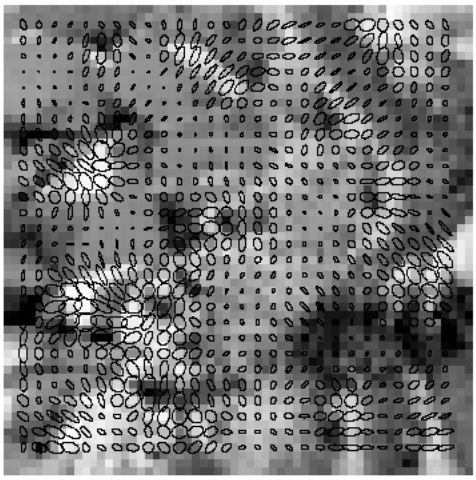
- **Types**
  - **constant**
    - small eigenvalues
  - **Edge**
    - one medium, one small
  - **Flow**
    - one large, one small
  - **corner**
    - two large eigenvalues

$$H = \sum_{window} (\nabla I)(\nabla I)^T$$





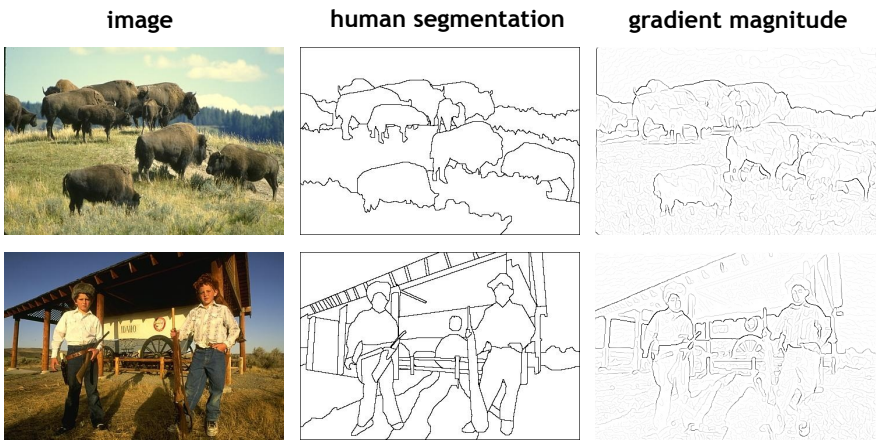




Disclaimer

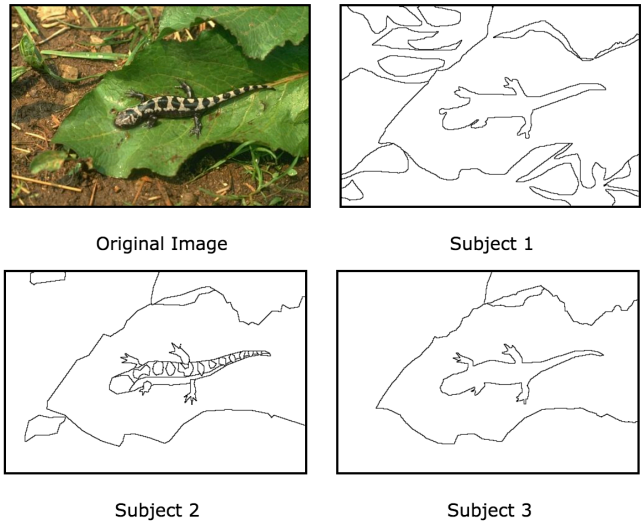
- No edge detection scheme is going to work perfectly in all cases. This is due to the fact that our notion of what constitutes a salient edge in the image is actually somewhat subtle.

Edge detection is just the beginning…



- **Berkeley segmentation database:**
  http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/

Even Humans Disagree



- **Berkeley segmentation database:**
  http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/