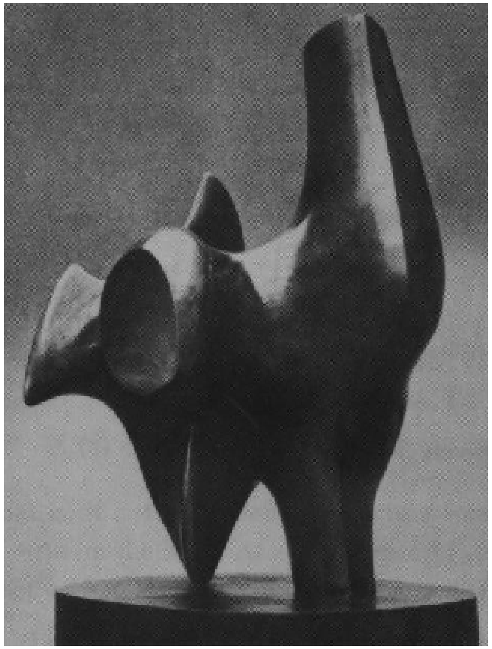
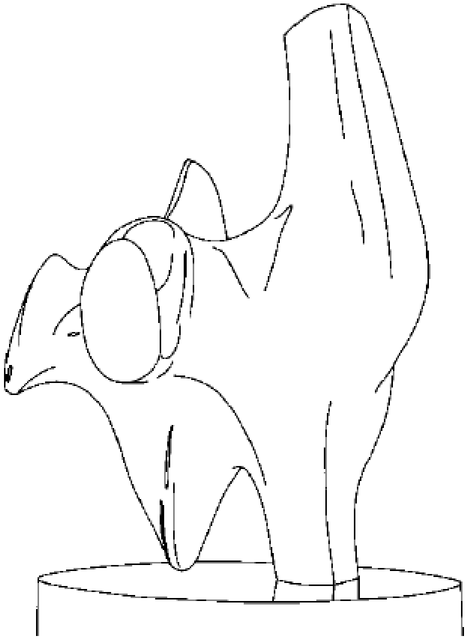
CS 664 Lecture 6
Edge and Corner Detection, Gaussian Filtering

Prof. Dan Huttenlocher
Fall 2003

Edge Detection

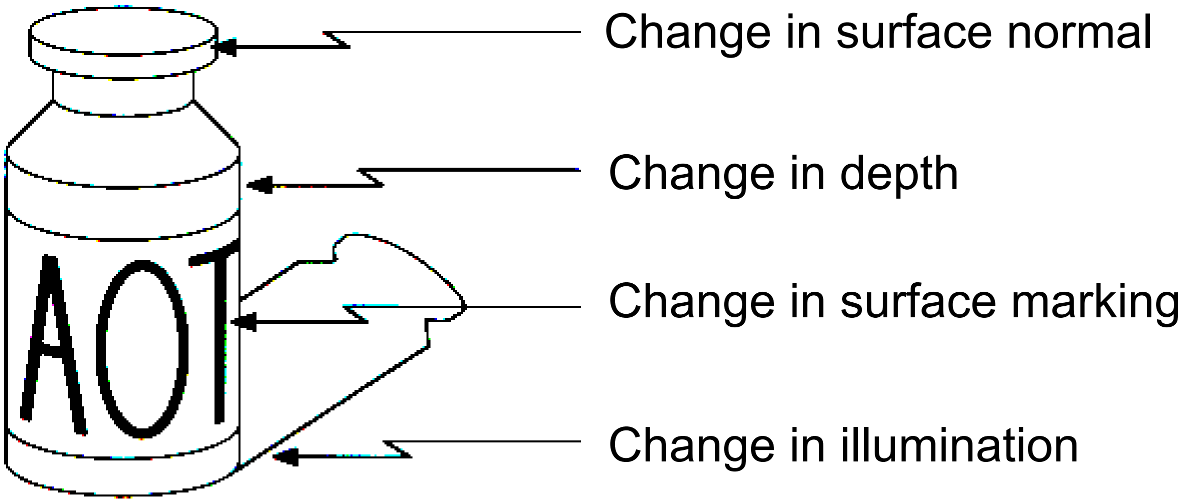
- Convert a gray or color image into set of curves
  - Represented as binary image
- Capture properties of shapes





Several Causes of Edges

- Sudden changes in various properties of scene can lead to intensity edges
  - Scene changes result in changes of image brightness/color



Detecting Edges

- Seek sudden changes in intensity
  - Various derivatives of image
- Idealized continuous image I(x,y)
- Gradient (first derivative), vector valued

$$\nabla I = (\partial I/\partial x, \partial I/\partial y)$$

- Squared gradient magnitude

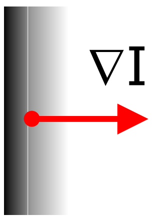
$$\|\nabla I\|^2 = (\partial I/\partial x)^2 + (\partial I/\partial y)^2$$

  - Avoid computing square root
- Laplacian (second derivative)

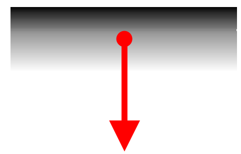
$$\nabla^2 I = \partial^2 I/\partial x^2 + \partial^2 I/\partial y^2$$

The Gradient

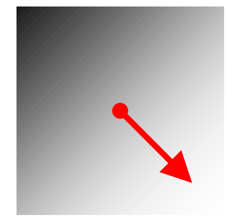
- Direction of most rapid change



$\nabla I = (\partial I/\partial x, 0)$



$\nabla I = (0, \partial I/\partial y)$



$\nabla I = (\partial I/\partial x, \partial I/\partial y)$

- Gradient direction is atan$(\partial I/\partial y, \partial I/\partial x)$
  - Normal to edge
- Strength of edge given by grad magnitude
  - Often use squared magnitude to avoid computing square roots

Finite Differences

- Images are digitized
  - Idealized continuous underlying function I(x,y) realized as discrete values on a grid I[u,v]
- Approximations to derivatives (1D)

  $dF/dx \approx F[u+1] - F[u]$

  $d^2F/dx^2 \approx F[u-1] - 2F[u] + F[u+1]$

1	0	1	0	10	11	11	0	1	
−1	1	−1	10	1	0	−11	1	0	dF: edge at extremum
−2	2	−2	11	−9	−1	−11	12	−2	d^2F: edge at zero crossing

- Second derivative symmetric about edge

Discrete Gradient

- Partial derivatives are estimated at boundaries between adjacent pixels
  - E.g., pixel and next one in x,y directions
- Yields estimates at different points in each direction if use x,y directions

2	1	2	3
1	0	1	2
1	0	1	2
2	1	2	3

- Generally use 45° directions to solve this
  - Magnitude fine, but gradient orientation needs to be rotated to correspond to axes

Estimating Discrete Gradient

- Gradient at u,v with 45° axes
  - Down-right: $\partial I/\partial x' \approx I[u+1,v+1]-I[u,v]$
  - Down-left: $\partial I/\partial y' \approx I[u,v+1]-I[u+1,v]$
- Handle image border, e.g., no change

u,v	u+1,v
u,v+1	u+1,v+1

2	1	1	2
7	6	7	6
1	6	7	7
2	2	7	6

I

4	6	5	0
-1	1	0	0
1	1	-1	0
0	0	0	0

$\partial I/\partial x'$

0	6	5	5
0	-5	-1	1
0	-4	-5	0
0	0	0	0

$\partial I/\partial y'$

16	72	50	25
1	26	1	1
1	17	26	0
0	0	0	0

$\| I \|^2$

Discrete Laplacian

- Laplacian at u,v

  $\partial^2 I/\partial x^2 = I[u-1,v]-2I[u,v]+I[u+1,v]$

  $\partial^2 I/\partial y^2 = I[u,v-1]-2I[u,v]+I[u,v+1]$

  $\nabla^2 I$ is sum of directional second derivatives:
  $I[u-1,v]+I[u+1,v]+I[u,v-1]+I[u,v+1]-4I[u,v]$
- Can view as 3x3 mask or stencil
  - Value at u,v given by sum of product with I
- Grid yields poor rotational symmetry
  - Weighted sum of two masks

	1	
1	-4	1
	1	

1		1
	-4	
1		1

1	4	1
4	-20	4
1	4	1

Local Edge Detectors

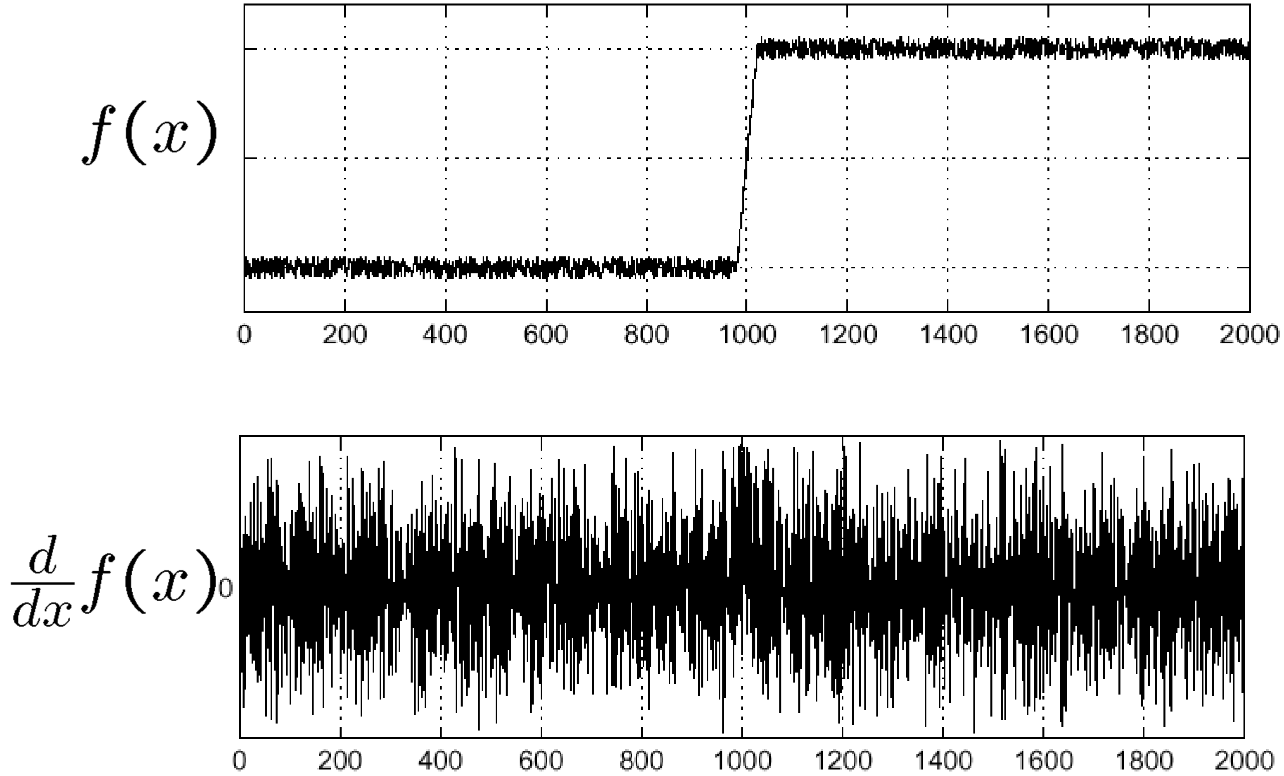
- Historically several local edge operators based on derivatives
  - Simple local weighting over small set of pixels
- For example Sobel operator
  - Derivatives in x and y
  - Weighted sum
  - 3x3 mask for symmetry
  - Today can do better with larger masks, fast algorithms, faster computers

-1		**1**
-2		**2**
-1		**1**

1	**2**	**1**
-1	**-2**	**-1**

Problems With Local Detectors

- 1D example illustrates effect of noise (variation) on local measures



Regions of Support

- Desirable to have edge detectors that operate over interval or region
- Low pass filtering of an image
  - Combining certain neighboring pixel values to produce "less variable" image
  - Often referred to as "smoothing" or as "blurring" the image
- Simple idea: mean filter – average values over w by h neighborhood

$$M[u,v] = (1/wh)\ \Sigma_{\mathbf{i}}\ \Sigma_{\mathbf{j}}\ F[u+i-(w-1)/2,\ v+j-(h-1)/2]$$

3x3 Mean Filter Example

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

?	?	?	?	?	?	?	?	?	?
?	0	10	20	30	30	30	20	10	?
?	0	20	40	60	60	60	40	20	?
?	0	30	60	90	90	90	60	30	?
?	0	30	50	80	80	90	60	30	?
?	0	30	50	80	80	90	60	30	?
?	0	20	30	50	50	60	40	20	?
?	10	20	30	30	30	30	20	10	?
?	10	10	10	0	0	0	0	0	?
?	?	?	?	?	?	?	?	?	?

$$M[u,v] = (1/9)\ \Sigma_i\ \Sigma_j\ F[u+i-1,\ v+j-1]$$

Border Pixels

- As usual with image operations the border cases need to be handled somehow
  - Produce smaller image by summing only when entire w by h window fits inside image
  - Sum only value inside image but produce full size image
    - In effect summing zeroes outside image
  - Assume value outside image some non-zero value
    - E.g., reflected copy of the image
- No right answer, reflection often least bad

Weighted Average Filter

- Sum of product with weights H

  $G[u,v] = \Sigma_{\mathbf{i}}\, \Sigma_{\mathbf{j}}\, H[i,j] F[u+i-(w-1)/2,\ v+j-(h-1)/2]$

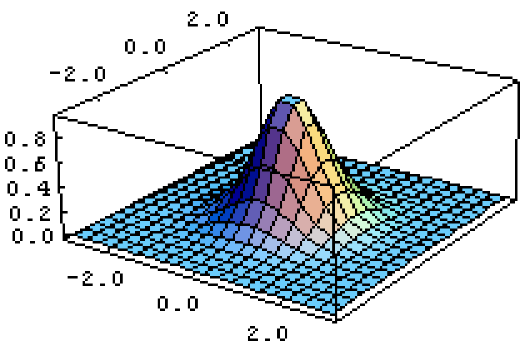
- Mean filter simply has H[I,j]=1/wh
  - Uniform weighting
- Note that entries of H should sum to 1
  - Otherwise performs overall scaling of the image
    - Consider 3x3 mask of 1's instead of 1/9's
- When averaging generally give central pixel most weight

Gaussian Filter

- Gaussian in two-dimensions

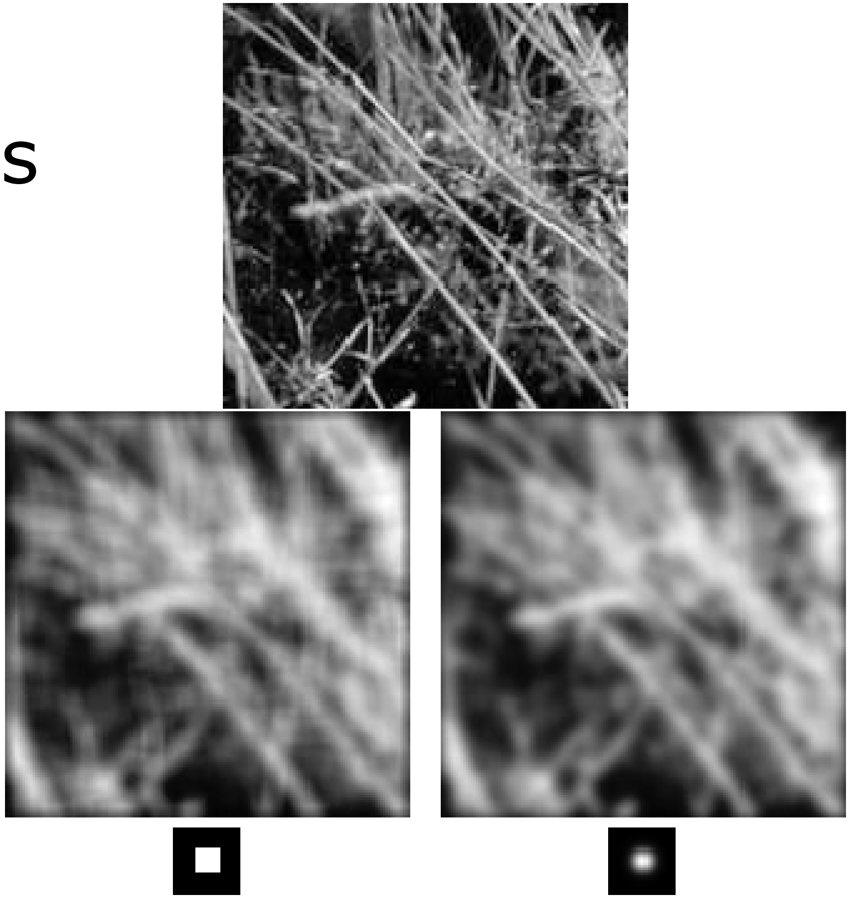
$$h(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$

- Weights center more
- Falls off smoothly
- Integrates to 1
- Larger σ produces more equal weights (blurs more)
- Normal distribution



Gaussian Versus Mean Filter

- Mean filter blurs but sharp changes remain as well
  - "Blocky"
- Gaussian not blocky looking
- Same area masks
  - But Gaussian small at borders



Cross Correlation

- The weighted summation operation is called the cross correlation

  $G[u,v] = \Sigma_i \Sigma_j H[i,j]F[u+i-(w-1)/2,\ v+j-(h-1)/2]$

  – Written as $G = H \otimes F$
    - Notation not consistent, sometimes written as $G = H \star F$, but we will use that for convolution
- Powerful operation
  – Every element of output G results from sum of product of two inputs H and F
  – Elements of output differ in shift of inputs H,F
  – Not that easy to grasp at first

Cross Correlation Examples

$$\underset{A}{\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}} \otimes \underset{B}{\begin{array}{|c|c|c|c|c|}\hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & a & b & c & 0 \\ \hline 0 & d & e & f & 0 \\ \hline 0 & g & h & i & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}} = \underset{G}{\begin{array}{|c|c|c|c|c|}\hline ? & ? & ? & ? & ? \\ \hline ? & a & b & c & ? \\ \hline ? & d & e & f & ? \\ \hline ? & g & h & i & ? \\ \hline ? & ? & ? & ? & ? \\ \hline \end{array}}$$

$$\underset{B}{\begin{array}{|c|c|c|}\hline a & b & c \\ \hline d & e & f \\ \hline g & h & i \\ \hline \end{array}} \otimes \underset{A}{\begin{array}{|c|c|c|c|c|}\hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}} = \underset{G}{\begin{array}{|c|c|c|c|c|}\hline ? & ? & ? & ? & ? \\ \hline ? & i & h & g & ? \\ \hline ? & f & e & d & ? \\ \hline ? & c & b & a & ? \\ \hline ? & ? & ? & ? & ? \\ \hline \end{array}}$$

Convolution

- Closely related operation that "flips" indices of H and F

  $G[u,v] = \Sigma_{\mathbf{i}} \Sigma_{\mathbf{j}} H[i,j]F[u-i+(w-1)/2, v-j+(h-1)/2]$

  – Written as $G = H \star F$

    - Again, notation not always consistent

- Note $\star$ and $\otimes$ same when H or F symmetric

  – I.e., unchanged when "flipped"

- Convolution has nice properties

  – Commutative: $A\star B=B\star A$

  – Associative: $A\star(B\star C)=(A\star B)\star C$

  – Distributive: $A\star(B+C)=(A\star B)+(A\star C)$

Convolution Examples

$$\underset{A}{\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}} * \underset{B}{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & a & b & c & 0 \\ 0 & d & e & f & 0 \\ 0 & g & h & i & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}} = \underset{G}{\begin{bmatrix} ? & ? & ? & ? & ? \\ ? & a & b & c & ? \\ ? & d & e & f & ? \\ ? & g & h & i & ? \\ ? & ? & ? & ? & ? \end{bmatrix}}$$

$$\underset{B}{\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}} * \underset{A}{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}} = \underset{G}{\begin{bmatrix} ? & ? & ? & ? & ? \\ ? & a & b & c & ? \\ ? & d & e & f & ? \\ ? & g & h & i & ? \\ ? & ? & ? & ? & ? \end{bmatrix}}$$

Identity for Convolution

- Unit impulse: one at origin, zero elsewhere
- Suggests why simple averaging produces "blocky" results
  - Consider a=b=… =i=K

a	b	c
d	e	f
g	h	i

*

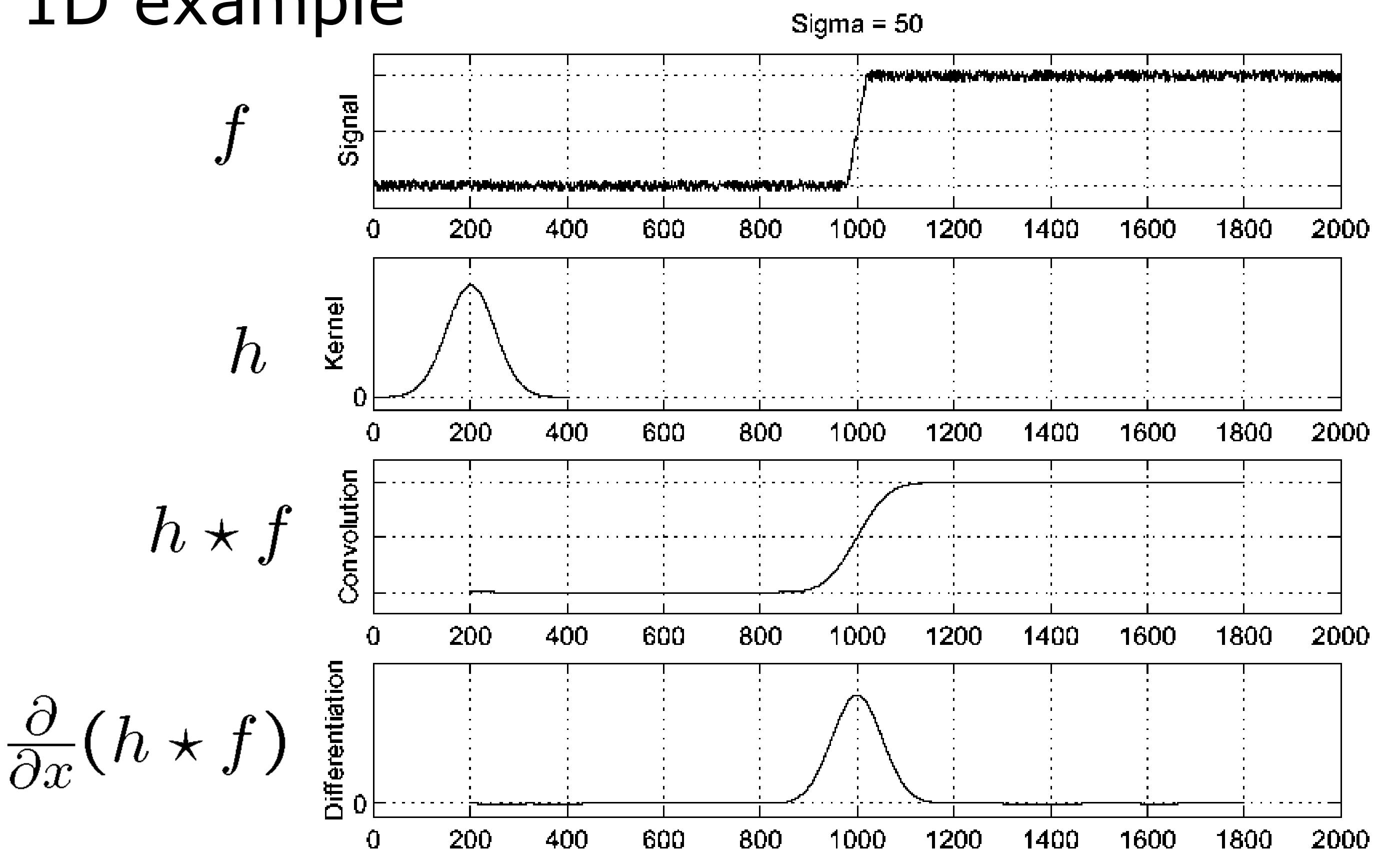
0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

=

?	?	?	?	?
?	a	b	c	?
?	d	e	f	?
?	g	h	i	?
?	?	?	?	?

Back to Edges: Derivatives

- Smooth and then take derivative
  - 1D example



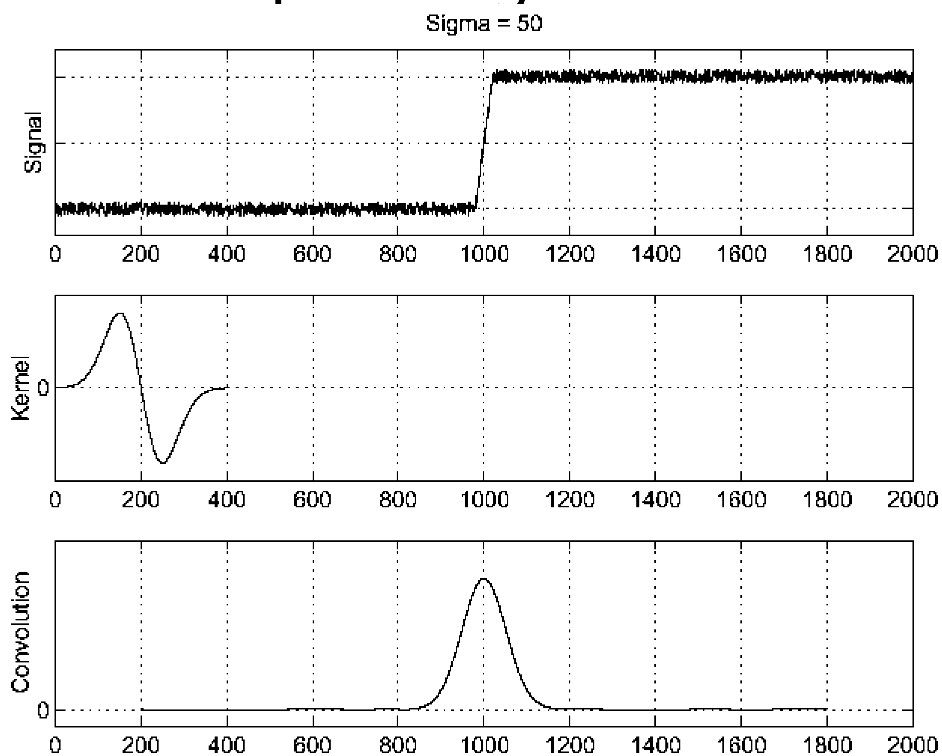
Derivatives and Convolutions

- Another useful identity for convolution is d/dx(A∗B)= (d/dx A)∗B = A∗(d/dx B)
  – Use to skip one step in edge detection

$$f$$

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

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



Derivatives Using Convolution

- When smoothing all weights of mask h are positive
  - Sum to 1
  - Maximum weight at center of mask
- Weights do not have to all be positive
  - Negative weights compute differences (derivatives)
  - E.g., Laplacian h =

1	4	1
4	-20	4
1	4	1

  - $h \star f = \nabla^2 f$
- Symmetry of h also gives us $h \star f = h \otimes f$
  - True for many masks; makes people sloppy

CORNELL

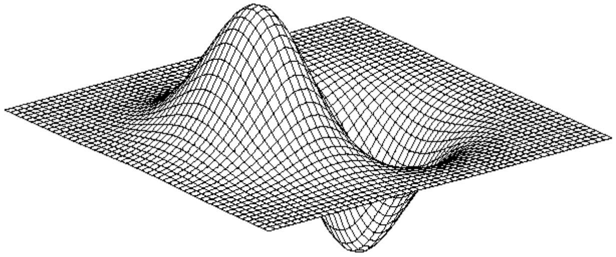
Linear Operators

- Linear shift invariant (LSI) system
  - Given a "black box" h: $f \rightarrow \boxed{h} \rightarrow g$
  - Linearity: $af_1+bf_2 \rightarrow \boxed{h} \rightarrow ag_1+bg_2$
  - Shift invariance: $f(x\text{-}u) \rightarrow \boxed{h} \rightarrow g(x\text{-}u)$
- Convolution with arbitrary h equivalent to these properties
  - Beyond this course to show it
- Linearity is "simple to understand" but real world not always linear
  - E.g., saturation effects

Area of Support for 2D Operators

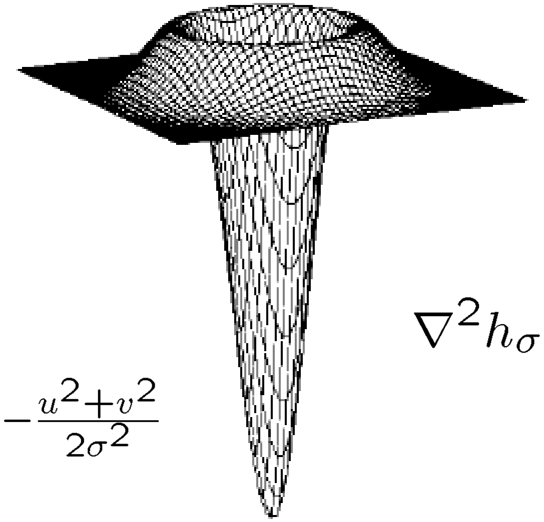
- Directional first derivatives and second derivative (Laplacian) of Gaussian
  - Sigma controls scale, larger yields fewer edges

Derivative of Gaussian



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

Laplacian of Gaussian



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

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

Gradient Magnitude

- Also use smoothed image

$$\|\nabla(I * h_\sigma)\| = ((\partial(I * h_\sigma)/\partial x)^2 + (\partial(I * h_\sigma)/\partial y)^2)^{.5}$$

Edge Detection by Subtraction

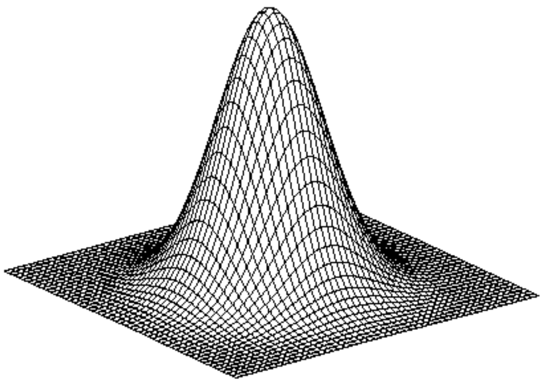
- Difference of image and smoothed version

Original

Smoothed

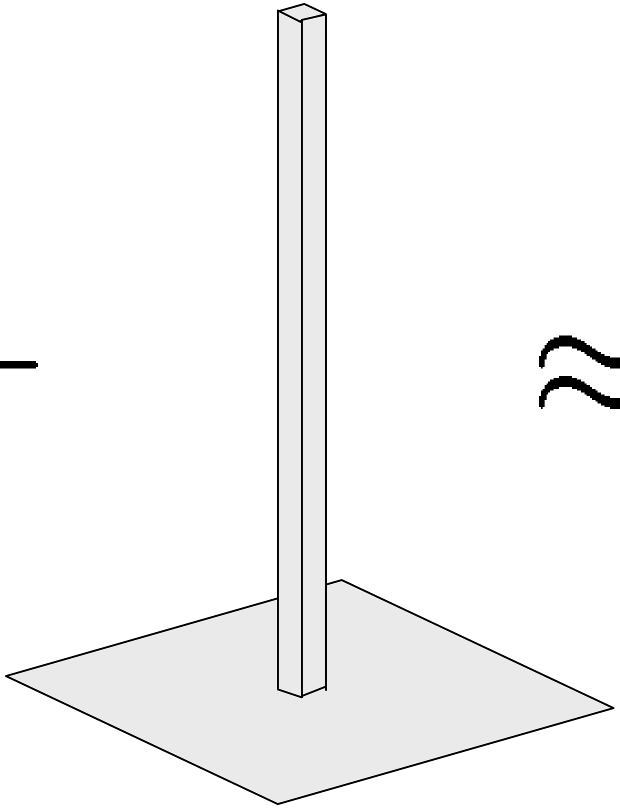
Difference
(brightened)

What Does This Do?



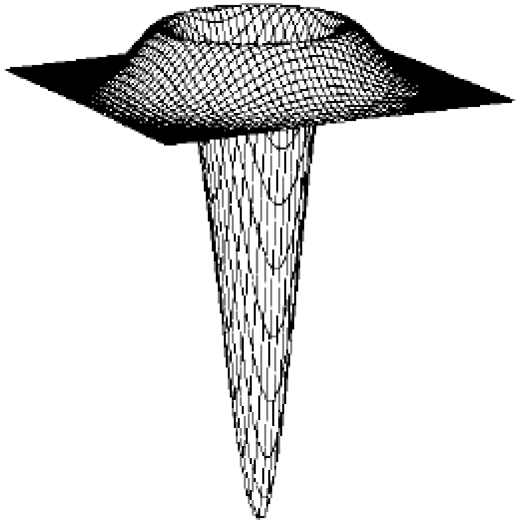
Gaussian

$-$



Impulse

$\approx$

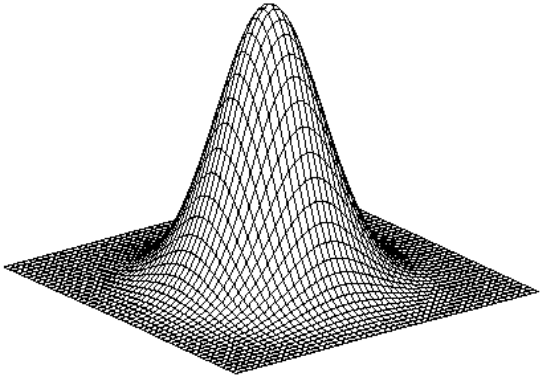


Laplacian of Gaussian

- More generally $(I \star h_{\sigma_1})-(I \star h_{\sigma_2}) \approx \nabla^2 I(I \star h_{\sigma_3})$

Efficient Gaussian Smoothing

- The 2D Gaussian is decomposable into separate 1D convolutions in x and y
- First note that product of two one-dimensional Gaussians



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

$$\frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-1}{2}\left(\frac{x^2}{\sigma^2}\right)} \frac{1}{\sqrt{2\pi}\sigma} e^{\frac{-1}{2}\left(\frac{y^2}{\sigma^2}\right)}$$

- Can view as product of two 1d vectors
  - Column vector times row vector each with values of 1d (sampled) Gaussian

Expressing as 1D Convolutions

- Use unit impulse as a notational trick
  - Continuous case: $\delta(x) = \infty$ when x is 0, else 0
  - Discrete case: $\delta[x] = 1$ when x is 0, else 0
  - $f \star \delta = f$
- $h_\sigma = h_{\sigma\mathbf{x}} \star h_{\sigma\mathbf{y}}$

$$\frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-1}{2}\left(\frac{x^2}{\sigma^2}\right)} \delta(y)$$

$$\frac{1}{\sqrt{2\pi\sigma}} e^{\frac{-1}{2}\left(\frac{y^2}{\sigma^2}\right)} \delta(x)$$

- $h_\sigma \star I = (h_{\sigma\mathbf{x}} \star h_{\sigma\mathbf{y}}) \star I = h_{\sigma\mathbf{x}} \star (h_{\sigma\mathbf{y}} \star I)$
  - Two 1D convolutions, don't sum the zeroes!

2D Gaussian as 1D Convolutions

0	0	0	0	0
0	16	16	16	0
0	16	16	16	0
0	16	16	16	0
0	0	0	0	0

*

1/4
1/2
1/4

=

0	4	4	4	0
0	12	12	12	0
0	16	16	16	0
0	12	12	12	0
0	4	4	4	0

*

1/4	**1/2**	**1/4**

=

0	0	0	0	0
0	16	16	16	0
0	16	16	16	0
0	16	16	16	0
0	0	0	0	0

*

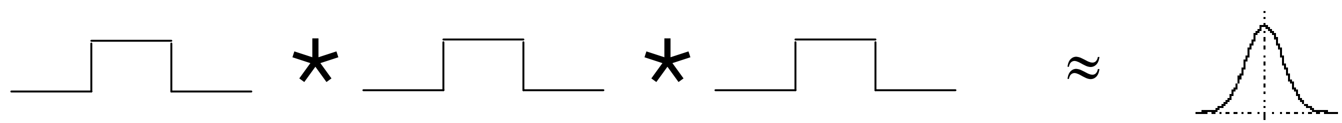
1/16	**1/8**	**1/16**
1/8	**1/4**	**1/8**
1/16	**1/8**	**1/16**

=

1	3	4	3	1
3	9	12	9	3
4	12	16	12	4
3	9	12	9	3
1	3	4	3	1

Fast 1D Gaussian Convolution

- Repeated convolution of box filters approximates a Gaussian
  - Application of central limit theorem, convolution of pdf’s tends towards normal distr.



$$[1\ 1\ 1] * [1\ 1\ 1] = [0\ 0\ 1\ 2\ 3\ 2\ 1\ 0\ 0]$$

$$* [1\ 1\ 1] = [0\ 1\ 3\ 6\ 7\ 6\ 3\ 1\ 0]$$

$$* [1\ 1\ 1] = [1\ 4\ 10\ 16\ 19\ 16\ 10\ 4\ 10]$$

Good Approximation to Gaussian

- Convolution of 4 unit height box filters of different widths yields low error
  - Wells, PAMI Mar 1986
- Simply apply each box filter separately
  - Also separate horizontal and vertical passes
  - Each box filter constant time per pixel
    - Running sum
- For Gaussian of given σ
  - Choose widths w_i such that $\Sigma_i (w_i^2-1)/12 \approx \sigma^2$
- In practice faster than explicit G_σ for $\sigma \approx 2$

What Makes Good Edge Detector

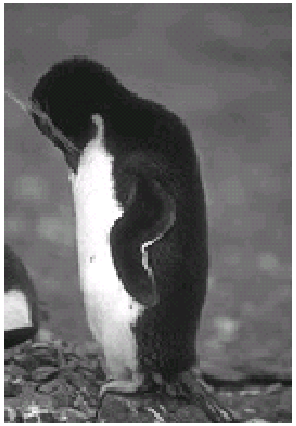
- Goals for an edge detector
  - Minimize probability of multiple detection
    - Two pixels classified as edges corresponding to single underlying edge in image
  - Minimize probability of false detection
  - Minimize distance between reported edge and true edge location
- Canny analyzes in detail 1D step edge
  - Shows that derivative of Gaussian is optimal with respect to above criteria
  - Analysis does not extend easily to 2D

Canny Edge Detector

- Based on gradient magnitude and direction of Gaussian smoothed image
  - Magnitude: $\|\nabla(G_\sigma \star I)\|$
  - Direction (unit vector): $\nabla(G_\sigma \star I) / \|\nabla(G_\sigma \star I)\|$
- Ridges in gradient magnitude
  - Peaks in direction of gradient (normal to edge) but not along edge
- Hysteresis mechanism for thresholding strong edges
  - Ridge pixel above lo threshold
  - Connected via ridge to pixel above hi threshold

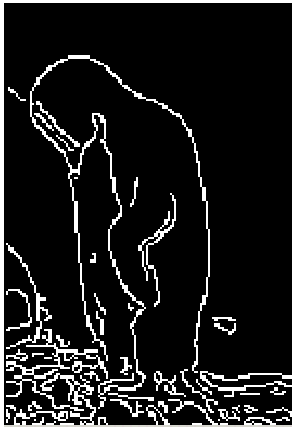
Canny Edge Definition

- Let $(\delta_x,\delta_y) = \nabla(G_\sigma \star I) / \| \nabla(G_\sigma \star I) \|$
  - Note compute without explicit square root
- Let $m = \| \nabla(G_\sigma \star I) \|^2$
- Non-maximum suppression (NMS)
  - $m(x,y) > m(x+\delta_{\mathbf{x}}(x,y), y+\delta_{\mathbf{y}}(x,y))$
  - $m(x,y) \geq m(x-\delta_{\mathbf{x}}(x,y), y-\delta_{\mathbf{y}}(x,y))$
  - Select “ridge points”
- Still leaves many candidate edge pixels
  - E.g., $\sigma=1$



Canny Thresholding

- Two level thresholding of candidate edge pixels (those that survive NMS)
  - Above lo and connected to pixel above hi
- Start by keeping (classifying as edges) all candidates above hi threshold
  - <u>Recursively</u> if pixel above lo threshold and adjacent to an edge pixel keep it
- Perform recursion using bfs/dfs
  - E.g., σ=1, lo=5, hi=10 and lo=10, hi=20



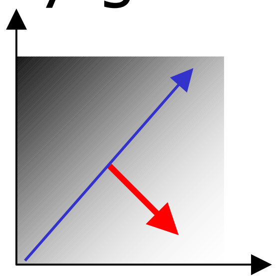
Corners

- Corner characterized by region with intensity change in two different directions

- Use local derivative estimates
  - Gradient oriented in different directions
- Not as simple as looking at gradient (partial derivatives) wrt coordinate frame

Corner Detectors

- Most detectors use local gradient estimate $I_x = \partial I / \partial x$ and $I_y = \partial I / \partial y$
  - Aggregated over rectangular region
- Seek substantial component to gradient in 2 distinct directions
  - Do so by finding coordinate axes normal to primary gradient direction



$(\partial I/\partial x,\ \partial I/\partial y)$

$(\partial I/\partial x',\ 0)$

- Also detects textured regions

Best Coordinate Frame

- Eigenvectors of scatter matrix

$$C = \begin{pmatrix} \Sigma\, I_x^2 & \Sigma\, I_x I_y \\ \Sigma\, I_x I_y & \Sigma I_y^2 \end{pmatrix}$$

- Major axis of points (I_x,I_y) in "gradient plane" – one for each pixel in region
- Orthogonal basis that best characterizes major elongation of points
  - Geometric view of eigenvector with largest eigenvalue

Simple Corner Detector

- Smooth image slightly
- Compute derivatives on 45° rotated axis
  - Eigenvectors thus oriented wrt that grid
  - Eigenvalues not affected
- Find eigenvalues λ_1,λ_2 of C ($\lambda_1<\lambda_2$)
  - If both large then high gradient in multiple directions
    - When λ_1 larger than threshold detect a corner
  - Eigenvalues can be computed in closed form

$$\begin{pmatrix} a & b \\ b & c \end{pmatrix}$$

$$\lambda_1 = \tfrac{1}{2}\left(a+c-\sqrt{(a-c)^2+4b^2}\right)$$

$$\lambda_2 = \tfrac{1}{2}\left(a+c+\sqrt{(a-c)^2+4b^2}\right)$$

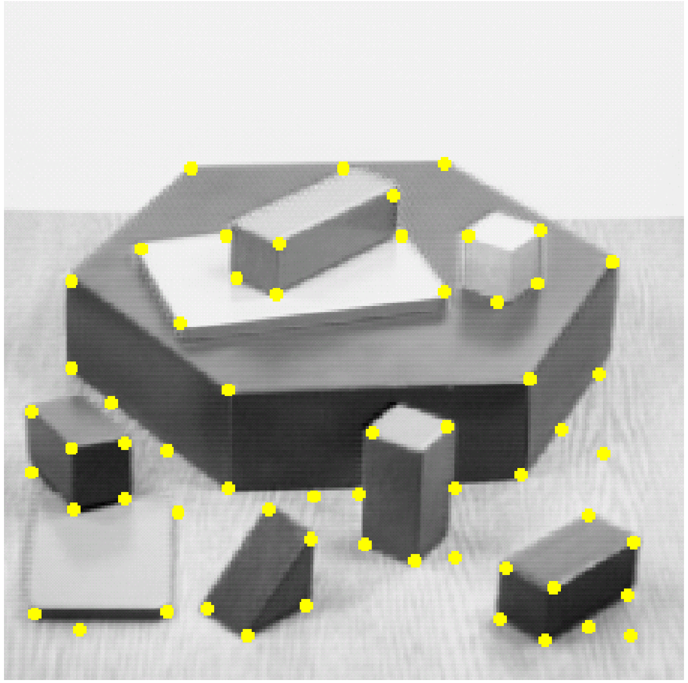
Corner Detectors

- Two most widely used
  - KLT (Kanade-Lucas-Tomasi) and Harris
- Both based on computing eigenvectors of scatter matrix of partial derivatives
  - Over some rectangular region
- Each has means of ensuring corners not too near one another
  - Enforcing distance between "good" regions
- KLT addresses change between image pair
  - Important for tracking corners across time

KLT Corner Detector

- Processing steps
  1. Compute I_x I_y locally at each pixel
     - Perhaps smooth image slightly first
  2. For each pixel compute C over d by d neighborhood centered around that pixel
     - Use box sum for all additions I_x^2, I_y^2, I_xI_y
  3. Compute smallest eigenvalue λ_1 of C at each pixel
  4. Select pixels above some threshold, in order of decreasing magnitude
     - Omit any pixel that is contained in neighborhood of previously included pixel

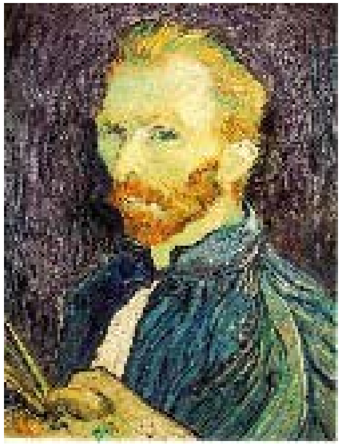
KLT Corner Detector Examples

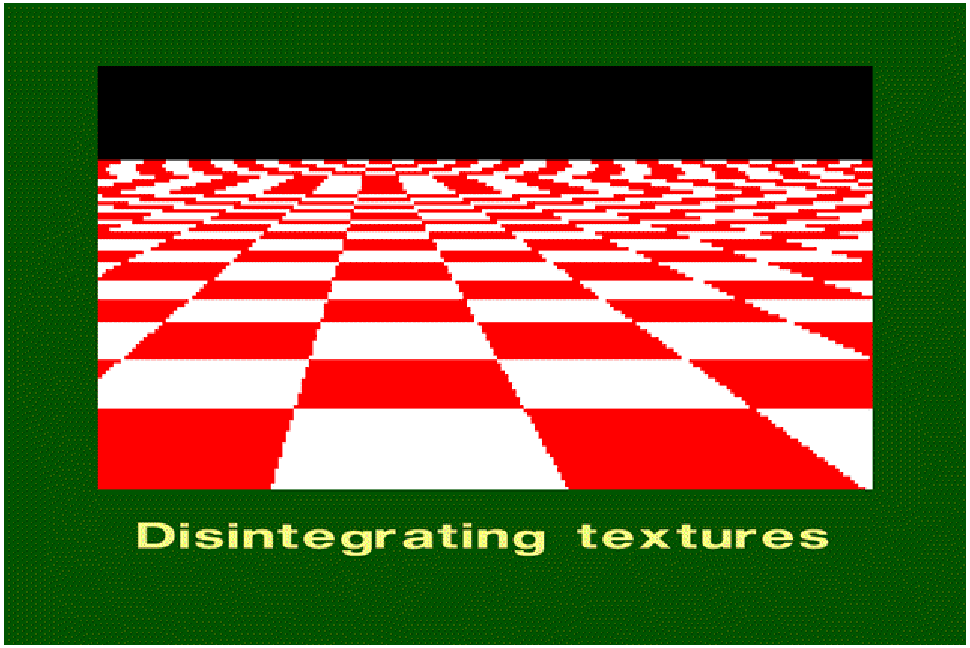


- KLT corner detection and tracking code available on the Web robotics.stanford.edu/~birch/klt/

Image Sub-Sampling

- To halve the resolution of an image seems natural to discard every other row & col
  - However produces poor looking results





Filter then Sub-Sample

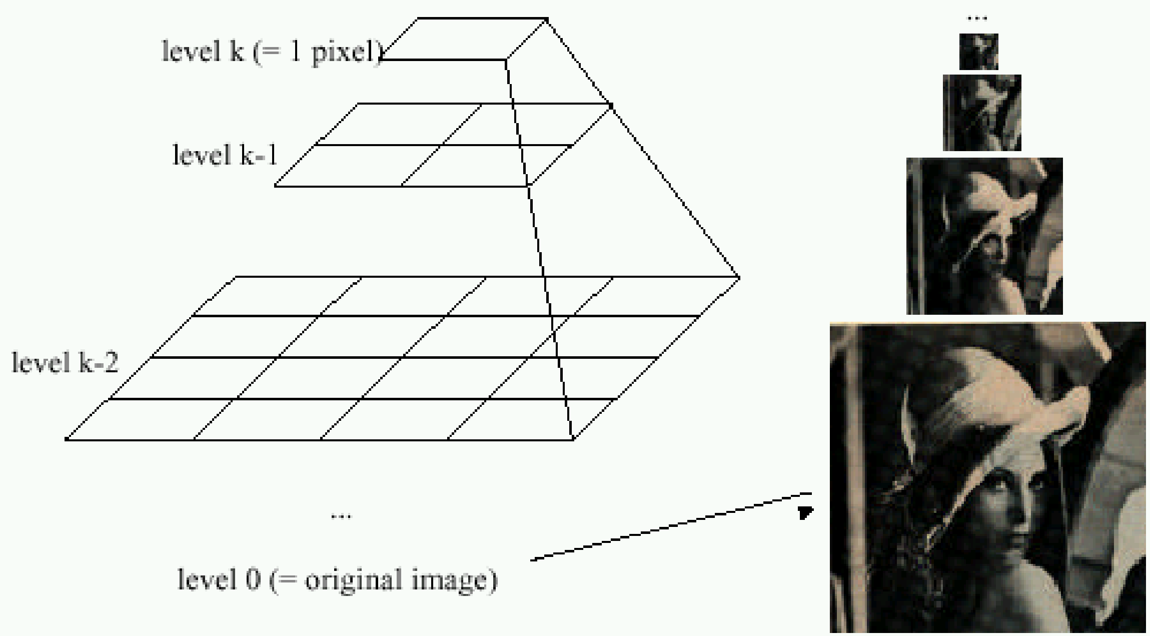
- Phenomenon known as aliasing
  - Need to remove high spatial frequencies
    - Can’t be represented accurately at lower resolution
    - E.g., 000111000111000111000111000
    - Downsample by 2: 00100100100100
    - Downsample by 4: 0100100
    - Downsample by 8: 0010
- Nyquist rate: need at least two samples per period of alternating signal
- Address by smoothing (lowpass filter)

Gaussian Filter and Sub-Sample

Gaussian Pyramid

- Filter and subsample at each level
  - Uses only 1/3 more storage than original

Idea: Represent NxN image as a “pyramid” of $1\mathrm{x}1$, $2\mathrm{x}2$, $4\mathrm{x}4$,…, $2^k\mathrm{x}2^k$ images (assuming $N=2^k$)



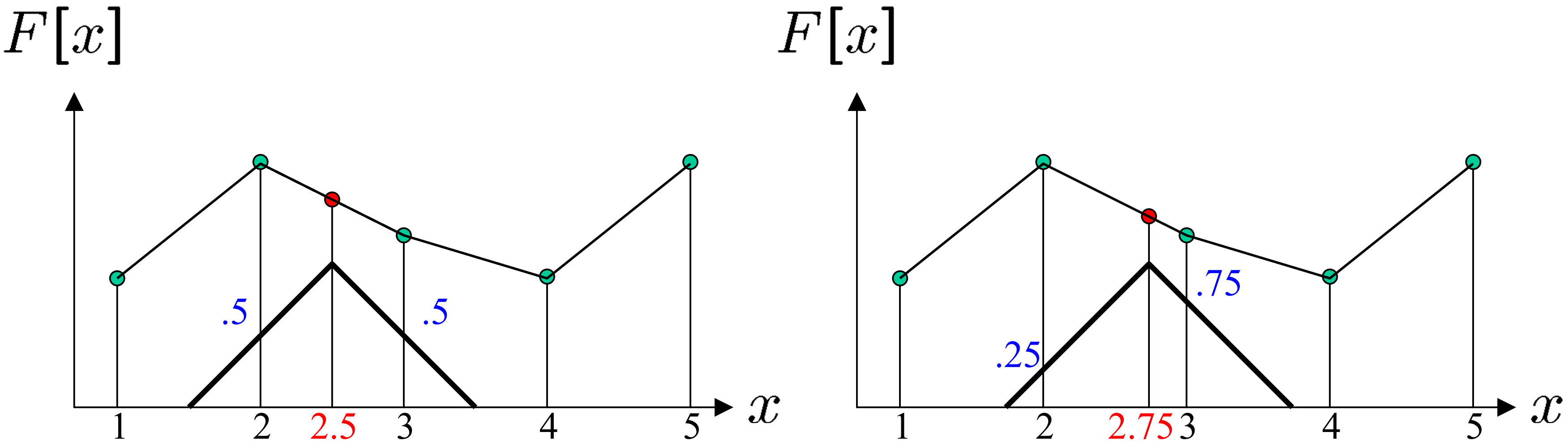
Sampling and Interpolation

- What if scale is not halving of the image
- What if want to upsample not downsample
- More general issue of constructing best samples on one grid given another grid
  - Often referred to as resampling
- If scaling down, first lowpass filter
- In both cases then map from one grid to another
  - Bilinear interpolation (2 by 2)
  - Bicubic interpolation (usually 4 by 4)

1D Linear Interpolation

- Compute intermediate values by weighted combination of neighboring values
  - Can view as convolution with “hat” on the original grid
    - E.g., equal spacing yields mask

.5	.5



Linear Interpolation by Convolution

- Implement by convolution with mask based on grid shift
  - If grid shifted to right by amount 0<a<1 then use mask [(1-a) a]
- For example grid shifted halfway between

.5	.5

*

0	2	4	4	8	6	6	4	2

1	3	4	6	7	6	5	3

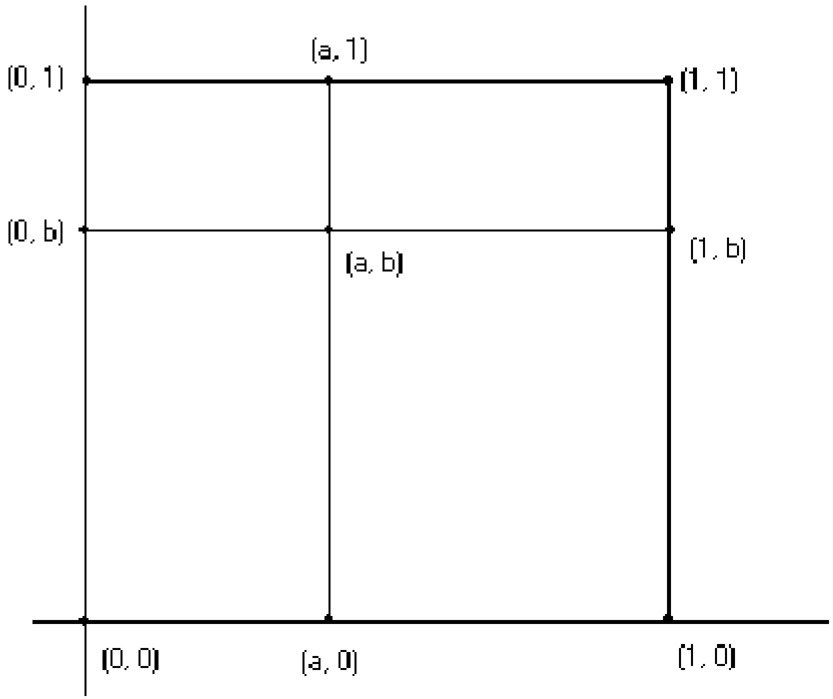
- Upsampled

0	1	2	3	4	4	4	6	8	7	6	6	6	5	4	3	2

Bilinear Interpolation

- Value at (a,b) based on four neighbors

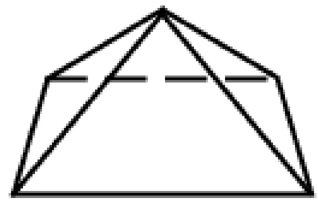
  $(1 - b)\ (1 - a)\ F_{0,0} + (1 - b)\ a\ F_{1,\,0}$
  $+\ b\ (1 - a)\ F_{0,1} + b\ a\ F_{1,\,0}$



Bilinear Interpolation by Convolution

- Convolution with two-dimensional function

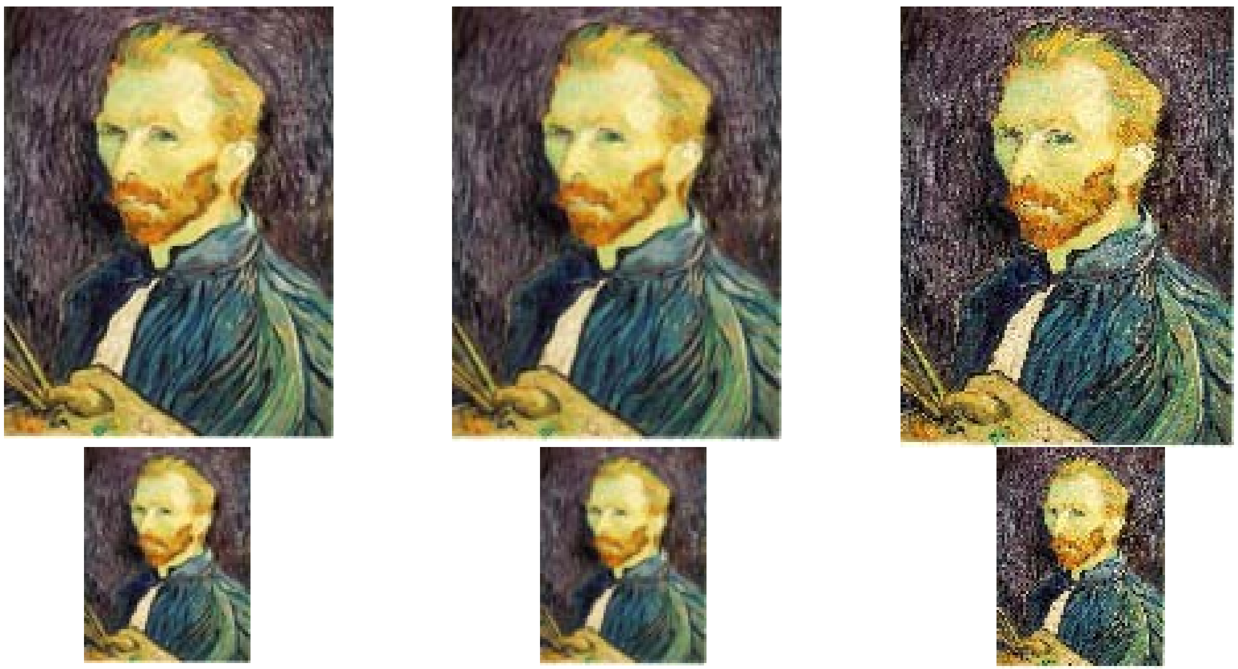
$h(x, y)$



- Perform two one-dimensional convolutions
  - Separable; simple to verify
  - New grid shifted down and to right by (a,b)
    - Where (as standard) origin of grid in upper left
  - Convolve horizontally with [(1-a) a)] then vertically with $[(1\text{-}b)\ b]^{T}$

Comparing Sampling Methods

- Bilinear filter and subsample, Gaussian filter and subsample, straight subsample



1/2

1/4

- Bicubic works better
  – Can also be implemented as convolution