CS4670/5670: Computer Vision

Kavita Bala

Lecture 3: Filtering and Edge detection

Announcements

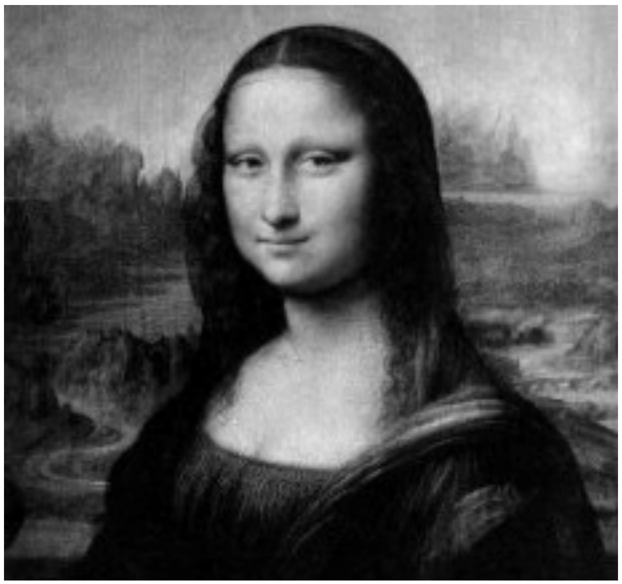
- PA 1 will be out later this week (or early next week)
  - due in 2 weeks
  - to be done in groups of two – please form your groups ASAP
- Piazza: make sure you sign up
- CMS: mail to Megan Gatch (mlg34@cornell.edu)

Mean filtering/Moving Average

- Replace each pixel with an average of its neighborhood
- Achieves smoothing effect
  - Removes sharp features

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

Filters: Thresholding



$$g(m,n) = \begin{cases} 255, & f(m,n) > A \\ 0 & otherwise \end{cases}$$

Linear filtering

- One simple version: linear filtering
  - Replace each pixel by a linear combination (a weighted sum) of its neighbors
  - Simple, but powerful
  - Cross-correlation, convolution
- The prescription for the linear combination is called the “kernel” (or “mask”, “filter”)

10	5	3
4	6	1
1	1	8

Local image data

0	0	0
0	0.5	0
0	1	0.5

kernel

	8	

Modified image data

Source: L. Zhang

Filter Properties

- Linearity
  - Weighted sum of original pixel values
  - Use same set of weights at each point
  - S[f + g] = S[f] + S[g]
  - S[k f + m g] = k S[f] + m S[g]

Linear Systems

- Is mean filtering/moving average linear?

- Is thresholding linear?

Filter Properties

- Linearity
  - Weighted sum of original pixel values
  - Use same set of weights at each point
  - S[f + g] = S[f] + S[g]
  - S[p f + q g] = p S[f] + q S[g]
- Shift-invariance
  - If f[m,n] $\xrightarrow{S}$ g[m,n], then f[m-p,n-q] $\xrightarrow{S}$ g[m-p, n-q]
  - The operator behaves the same everywhere

Overview

- Two important filtering operations
  - Cross correlation
  - Convolution
- Sampling theory
- Multiscale representations

Cross-correlation

Let F be the image, H be the kernel (of size 2*k*+1 x 2*k*+1), and G be the output image

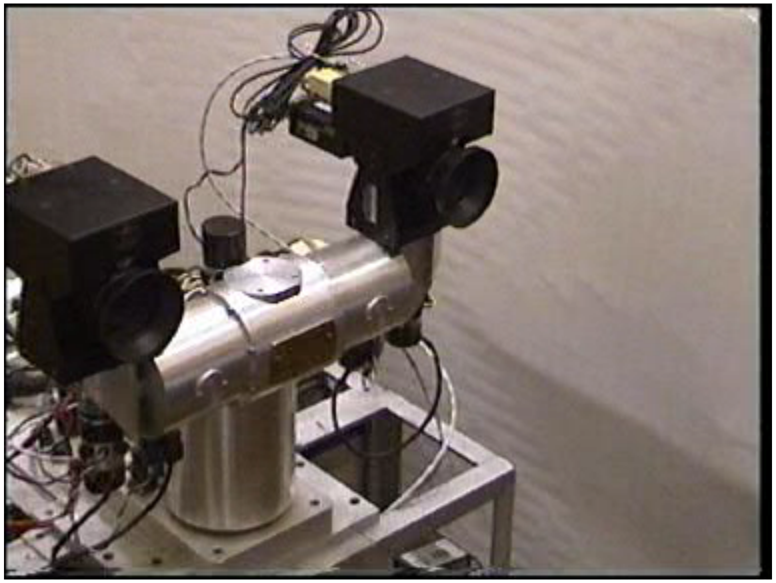
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] F[i+u, j+v]$$

This is called a **cross-correlation** operation:

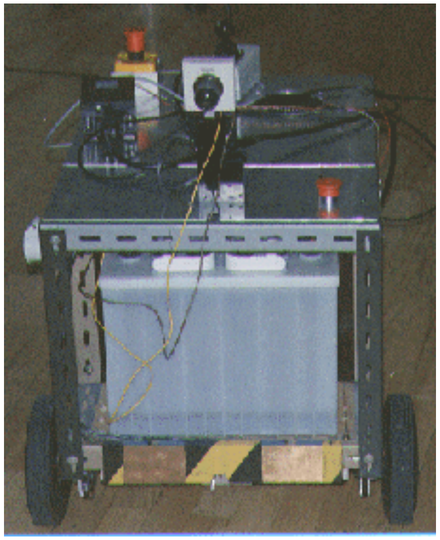
$$G = H \otimes F$$

- Can think of as a "dot product" between local neighborhood and kernel for each pixel

Stereo head



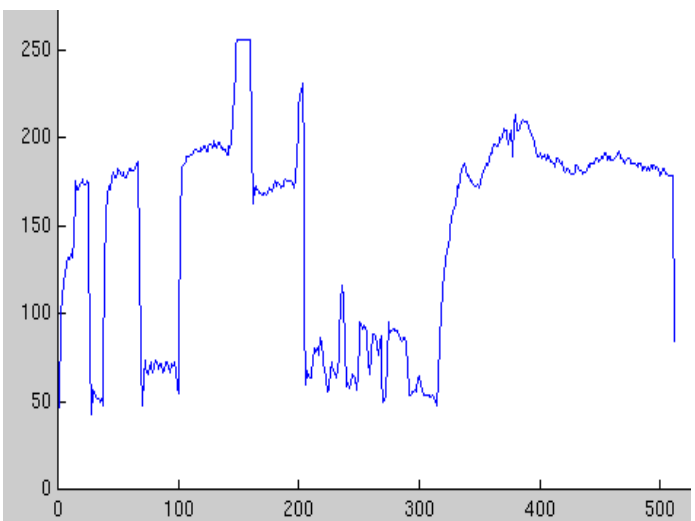
Camera on a mobile vehicle

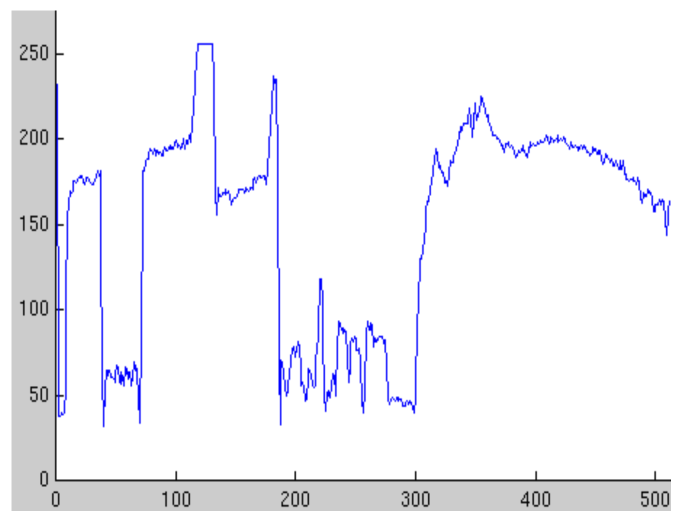


(COURTESY SONY)

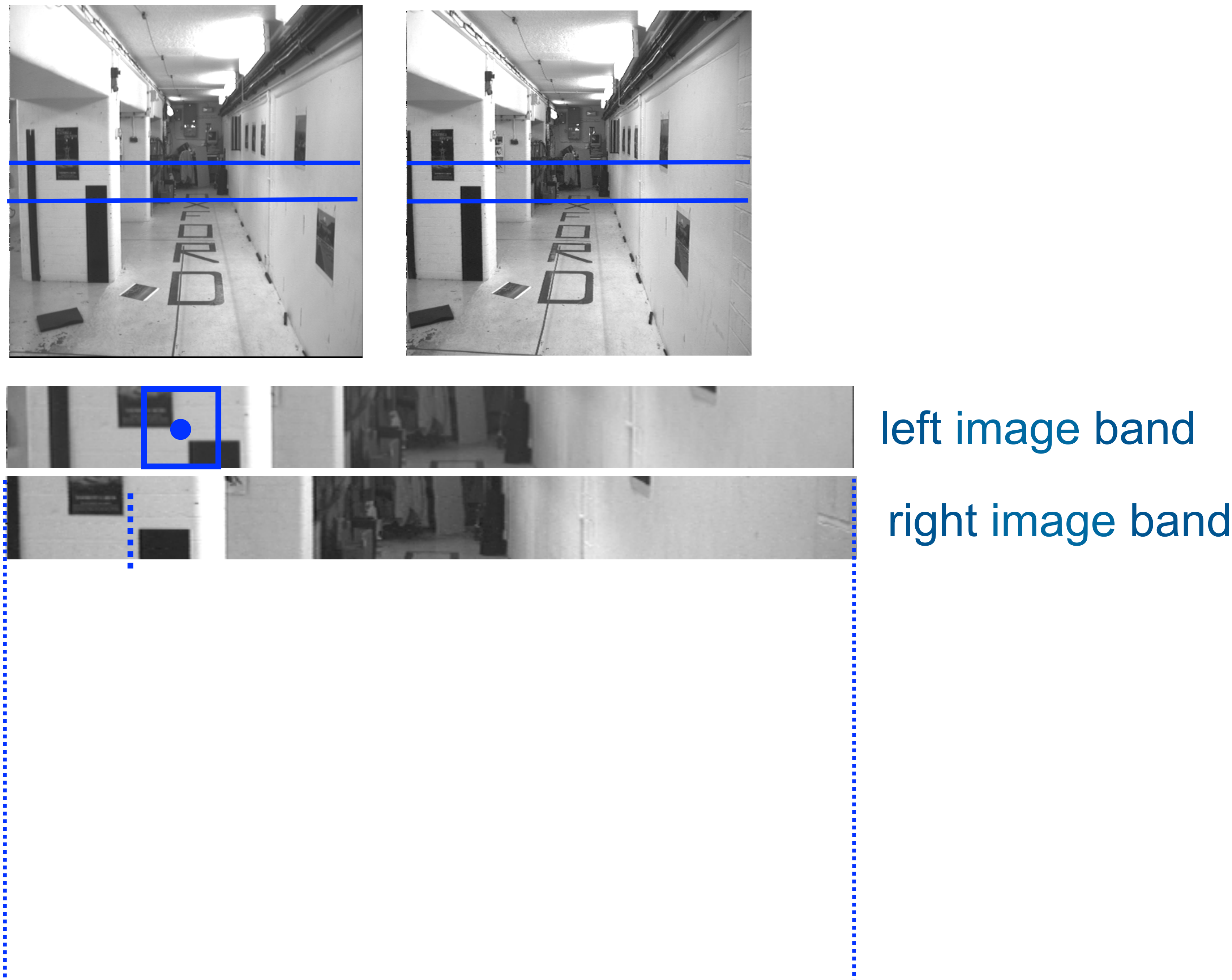
Example image pair – parallel cameras

Intensity profiles





- Clear correspondence between intensities, but also noise and ambiguity

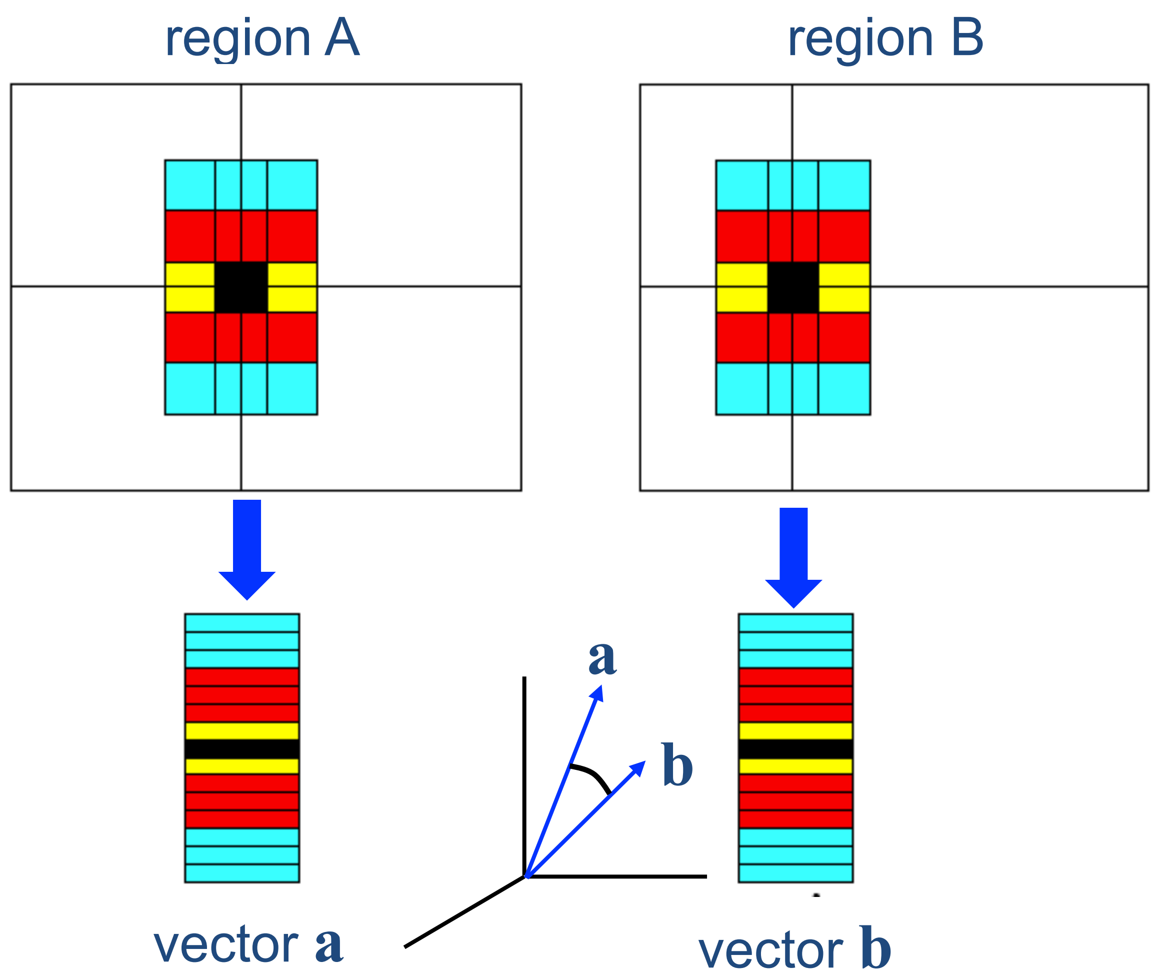

left image band
right image band

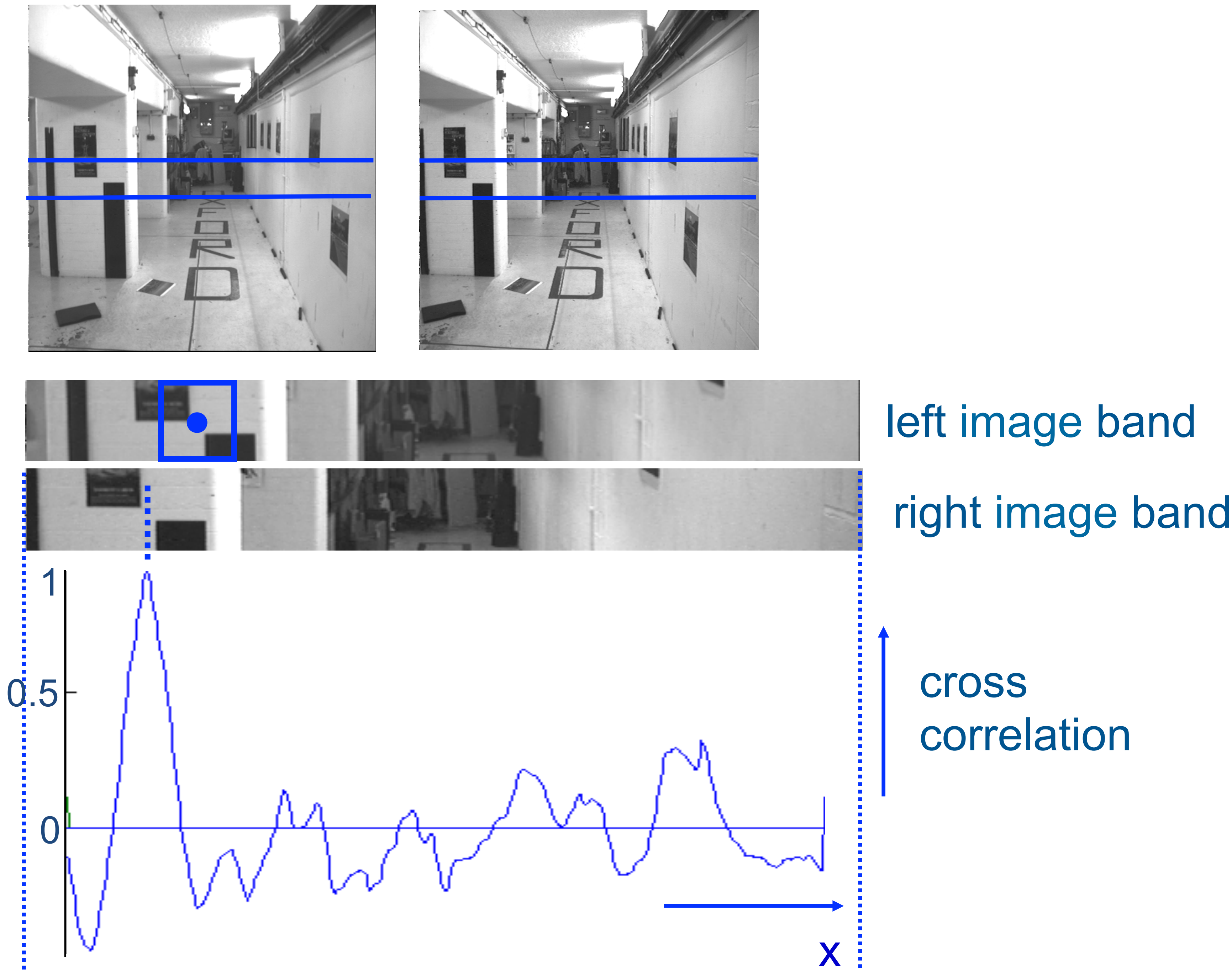
Normalized Cross Correlation

write regions as vectors

$$\mathtt{A} \rightarrow \mathbf{a}, \ \ \mathtt{B} \rightarrow \mathbf{b}$$

$$\text{NCC} = \frac{\mathbf{a}.\mathbf{b}}{|\mathbf{a}||\mathbf{b}|}$$





left image band
right image band
1
0.5
0
cross
correlation
x

Cross-correlation

Let F be the image, H be the kernel (of size 2*k*+1 x 2*k*+1), and G be the output image

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

This is called a **cross-correlation** operation:

$$G = H \otimes F$$

- Can think of as a "dot product" between local neighborhood and kernel for each pixel

Convolution

- Same as cross-correlation, except that the kernel is "flipped" (horizontally and vertically)

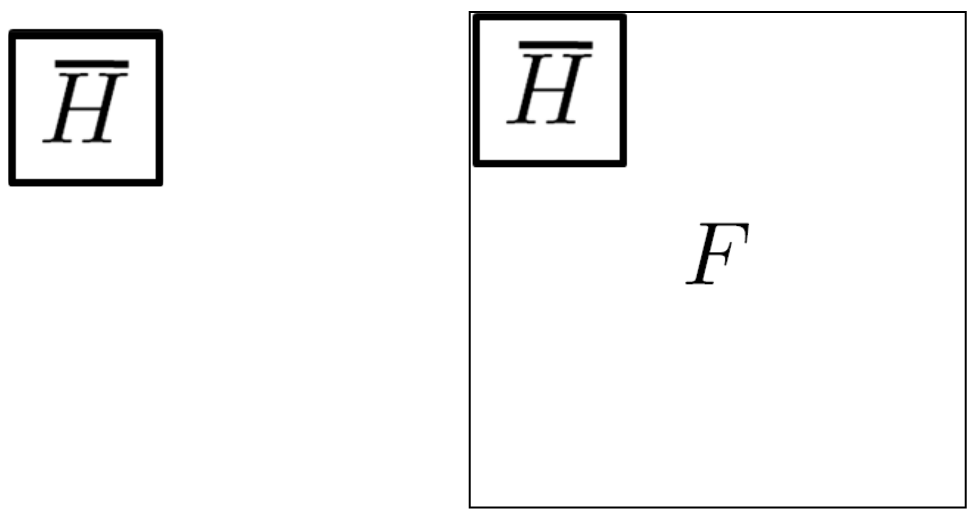
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

This is called a **convolution** operation:

$$G = H * F$$

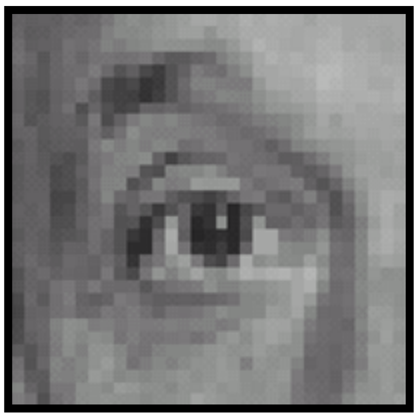
- Convolution is **commutative** and **associative**

Convolution



Adapted from F. Durand

Linear filters: examples

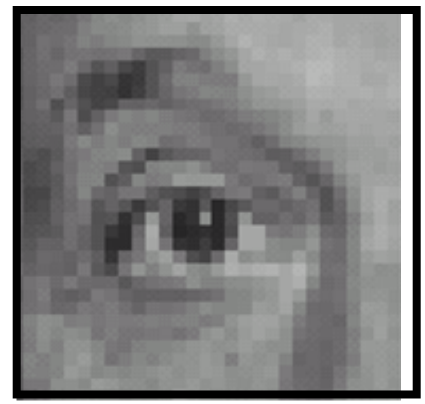


Original

*

0	0	0
1	0	0
0	0	0

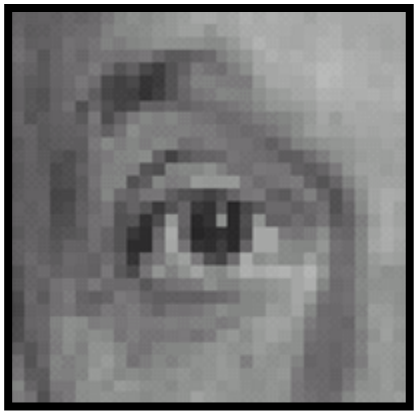
=



Shifted left
By 1 pixel

Source: D. Lowe

Linear filters: examples



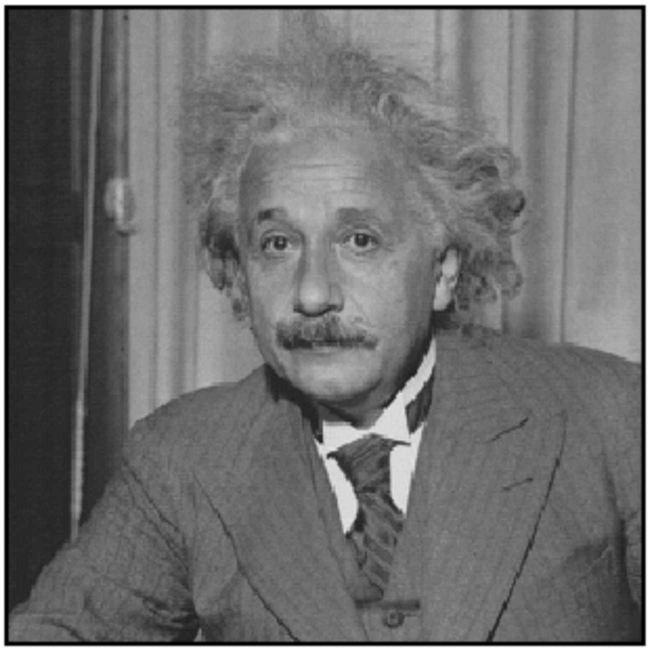
Original

$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) =$$

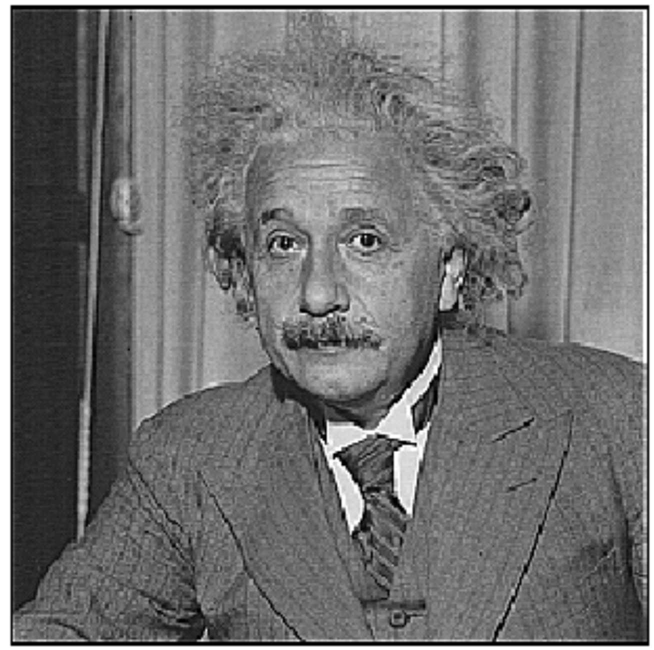
Sharpening filter
(accentuates edges)

Source: D. Lowe

Sharpening



before



after

Source: D. Lowe

Sharpening revisited

- What does blurring take away?

−

=

Let’s add it back:

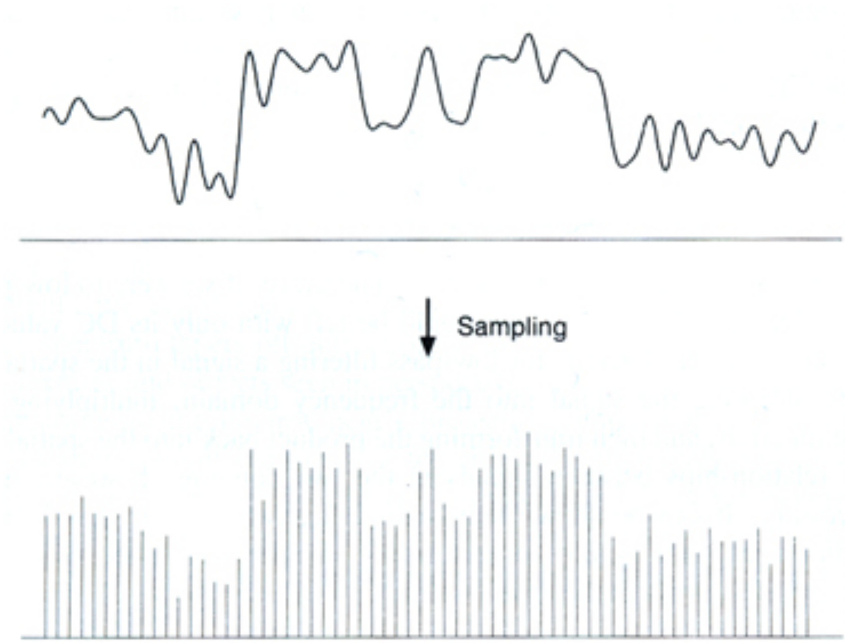
+ α

=

Source: S. Lazebnik

Sampling Theory

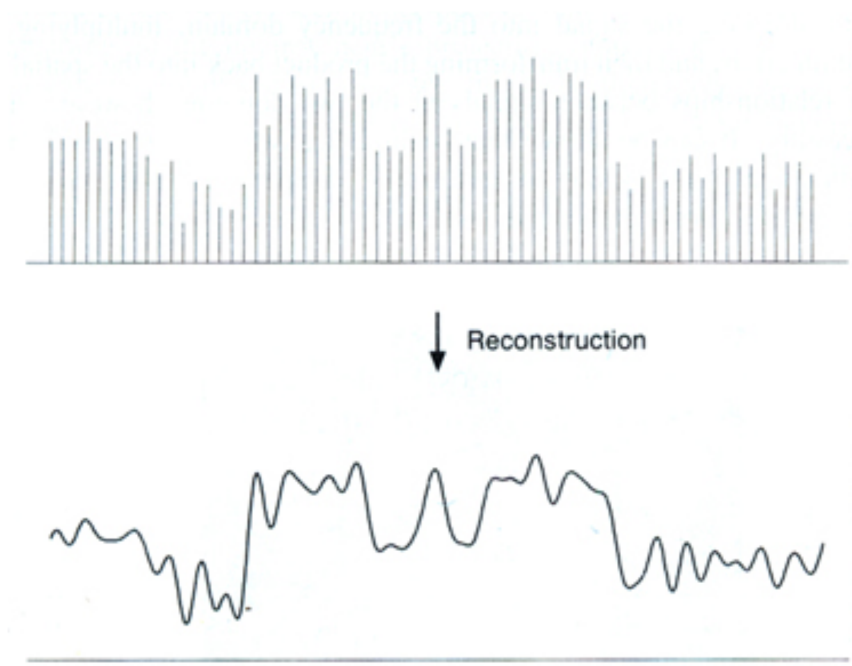
Sampled representations



[FvDFH fig.14.14b / Wolberg]

Reconstruction

- Making samples back into a continuous function
  - for output (need realizable method)
  - for analysis or processing (need mathematical method)



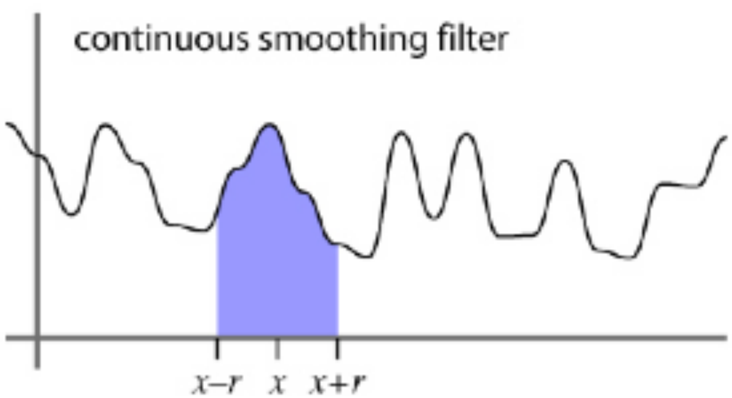
[FvDFH fig.14.14b / Wolberg]

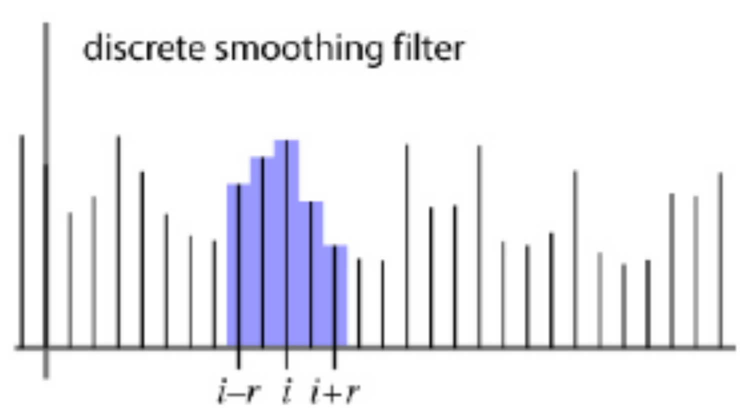
Roots of sampling

- Nyquist 1928; Shannon 1949
  - famous results in information theory
- 1940s: first practical uses in telecommunications
- 1960s: first digital audio systems
- 1970s: commercialization of digital audio
- 1982: introduction of the Compact Disc
  - the first high-profile consumer application
- This is why all the terminology has a communications or audio "flavor"
  - early applications are 1D; for us 2D (images) is important

Filtering

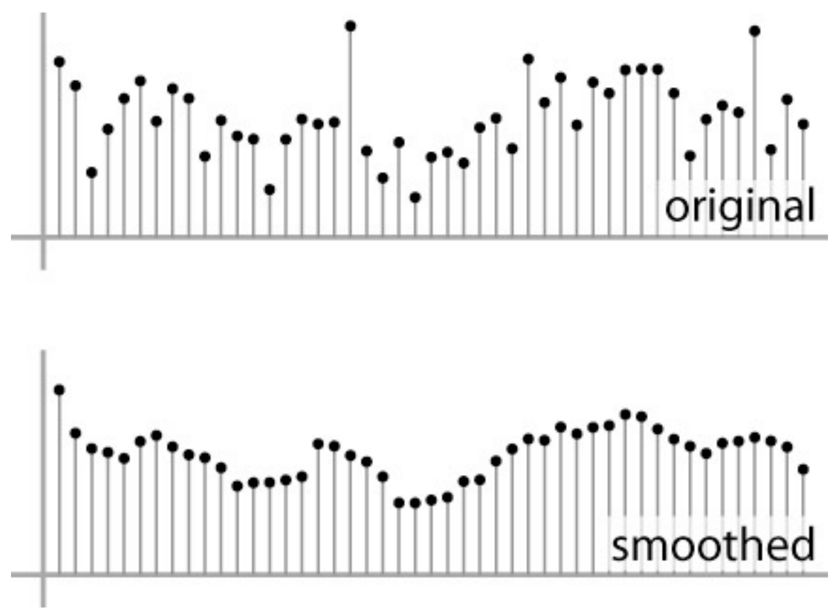
- Processing done on a function
  - in continuous form
  - also using sampled representation
- Simple example: smoothing by averaging





Convolution warm-up

- basic idea: define a new function by averaging over a sliding window
- a simple example to start off: smoothing



Convolution warm-up

- Same moving average operation, expressed mathematically:

$$b_{\text{smooth}}[i] = \frac{1}{2r+1} \sum_{j=i-r}^{i+r} b[j]$$

Discrete convolution

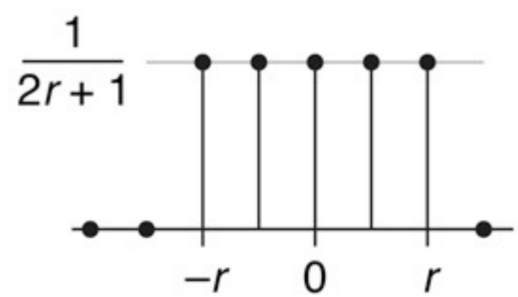
- Simple averaging: $b_{\text{smooth}}[i] = \frac{1}{2r+1} \sum_{j=i-r}^{i+r} b[j]$
  - –every sample gets the same weight
- Convolution: same idea but with *weighted* average

$$(a \star b)[i] = \sum_j a[j] b[i-j]$$

  - –each sample gets its own weight (normally zero far away)
- This is all convolution is: a moving weighted average

Filters

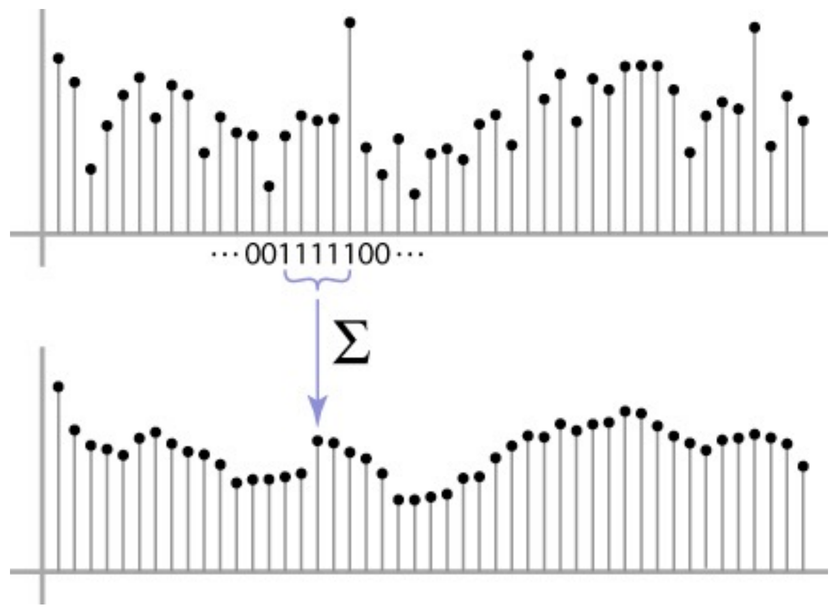
- Sequence of weights *a*[*j*] is called a *filter*
- Filter is nonzero over its *region of support*
  - –usually centered on zero: support radius *r*
- Filter is *normalized* so that it sums to 1.0
  - –this makes for a weighted average
    - not just any old weighted sum
- Most filters are symmetric about 0
  - –since for images we usually want to treat left and right the same



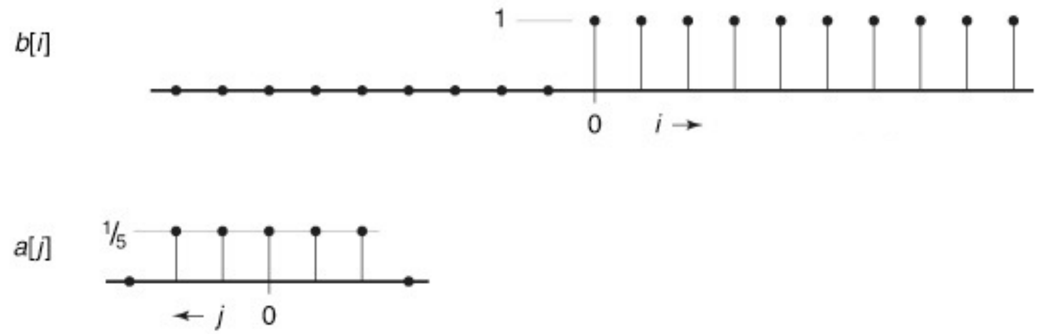
a box filter

Convolution and filtering

- Can express sliding average as convolution with a *box filter*
- a_{box} = [..., 0, 1, 1, 1, 1, 1, 0, ...]

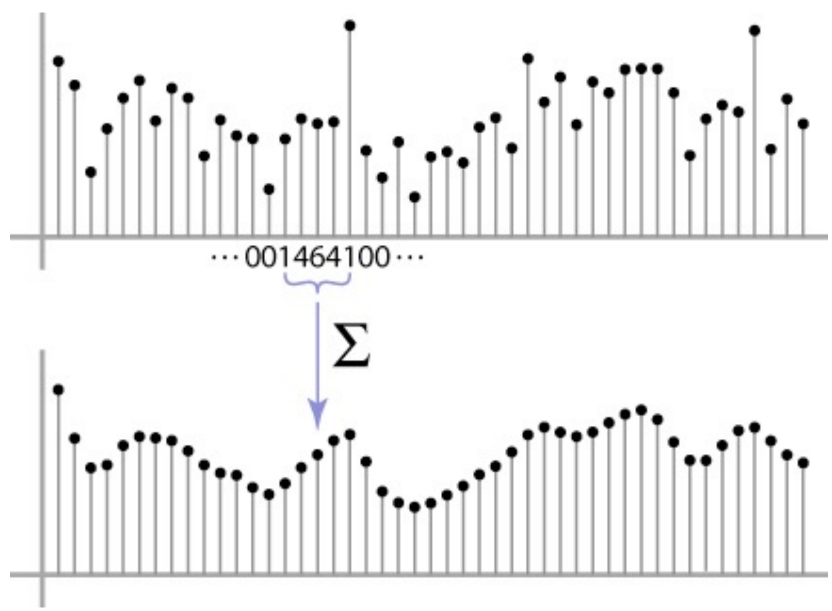


Example: box and step



Convolution and filtering

- Convolution applies with any sequence of weights
- Example: Bell curve (Gaussian-like)
  - [..., 1, 4, 6, 4, 1, ...]/16



And in pseudocode…

function convolve(sequence a, sequence b, int r, int i)
    $s = 0$
    for $j = -r$ to r
        $s = s + a[j]b[i - j]$
    return s

Discrete convolution

- Notation: $b = c \star a$
- Convolution is a multiplication-like operation
  - –commutative $a \star b = b \star a$
  - –associative $a \star (b \star c) = (a \star b) \star c$
  - –distributes over addition $a \star (b + c) = a \star b + a \star c$
  - –scalars factor out $\alpha a \star b = a \star \alpha b = \alpha(a \star b)$
  - –identity: unit impulse *e* = [..., 0, 0, 1, 0, 0, ...]

    $a \star e = a$
- Conceptually no distinction between filter and signal

Discrete filtering in 2D

- Same equation, one more index

$$(a \star b)[i,j] = \sum_{i',j'} a[i',j']b[i-i',j-j']$$

  – now the filter is a rectangle you slide around over a grid of numbers
- Commonly applied to images
  – blurring (using box, gaussian, …)
  – sharpening
- Usefulness of associativity
  – often apply several filters one after another:
    - $(((a * b_1) * b_2) * b_3)$
  – this is equivalent to applying one filter:
    - $a * (b_1 * b_2 * b_3)$

And in pseudocode…

function convolve2d(filter2d a, filter2d b, int i, int j)

$s = 0$

$r = a$.radius

for $i' = -r$ to r **do**

  for $j' = -r$ to r **do**

    $s = s + a[i'][j']b[i - i'][j - j']$

return s

Optimization: separable filters

- basic alg. is $O(r^2)$: large filters get expensive fast!
- definition: $a_2(x,y)$ is *separable* if it can be written as: $a_2[i,j] = a_1[i]a_1[j]$
  - this is a useful property for filters because it allows factoring:

$$(a_2 \star b)[i,j] = \sum_{i'} \sum_{j'} a_2[i',j']b[i-i',j-j']$$

$$= \sum_{i'} \sum_{j'} a_1[i']a_1[j']b[i-i',j-j']$$

$$= \sum_{i'} a_1[i'] \left(\sum_{j'} a_1[j']b[i-i',j-j'] \right)$$

[Philip Greenspun]

two-stage resampling using a separable filter

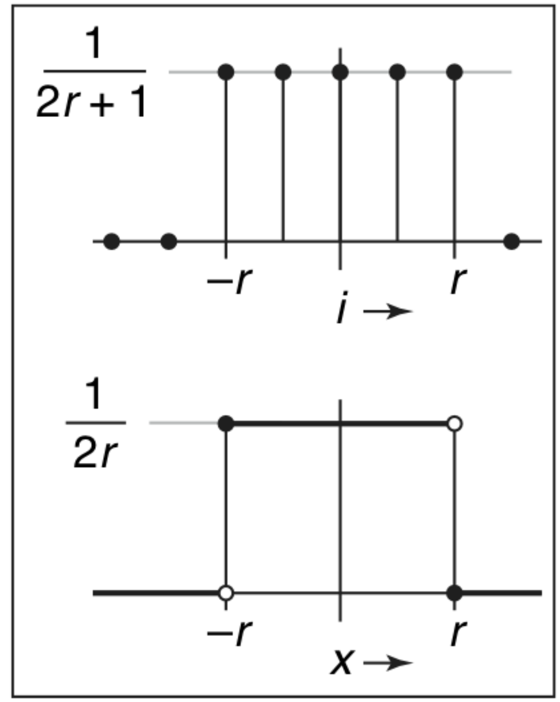
A gallery of filters

- Box filter
  - Simple and cheap
- Tent filter
  - Linear interpolation
- Gaussian filter
  - Very smooth antialiasing filter
- B-spline cubic
  - Very smooth
- ...

Box filter

$$a_{\text{box},r}[i] = \begin{cases} 1/(2r+1) & |i| \le r, \\ 0 & \text{otherwise.} \end{cases}$$

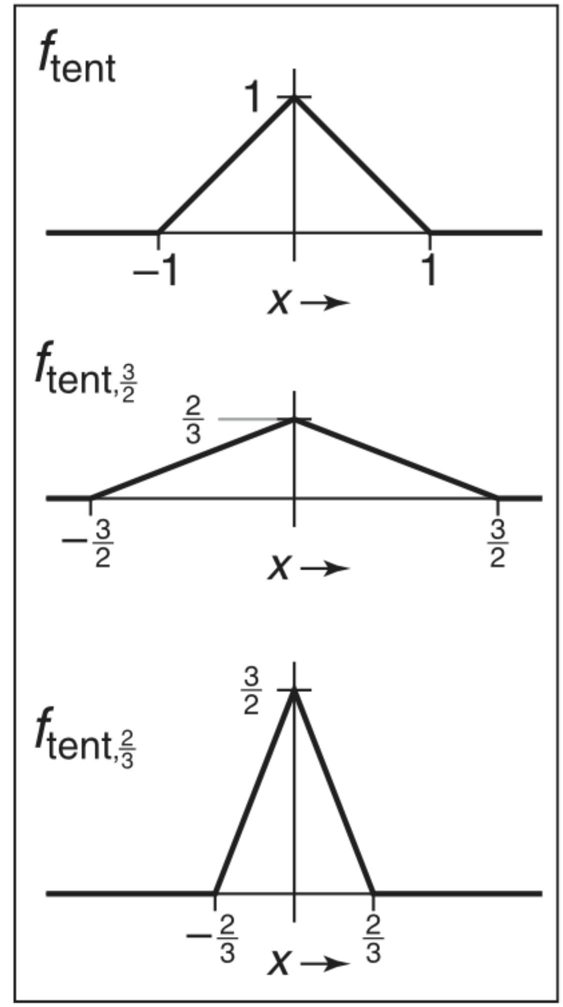
$$f_{\text{box},r}(x) = \begin{cases} 1/(2r) & -r \le x < r, \\ 0 & \text{otherwise.} \end{cases}$$



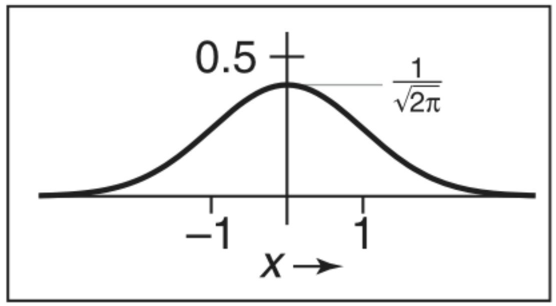
Tent filter

$$f_{\text{tent}}(x) = \begin{cases} 1 - |x| & |x| < 1, \\ 0 & \text{otherwise}; \end{cases}$$

$$f_{\text{tent},r}(x) = \frac{f_{\text{tent}}(x/r)}{r}.$$



Gaussian filter



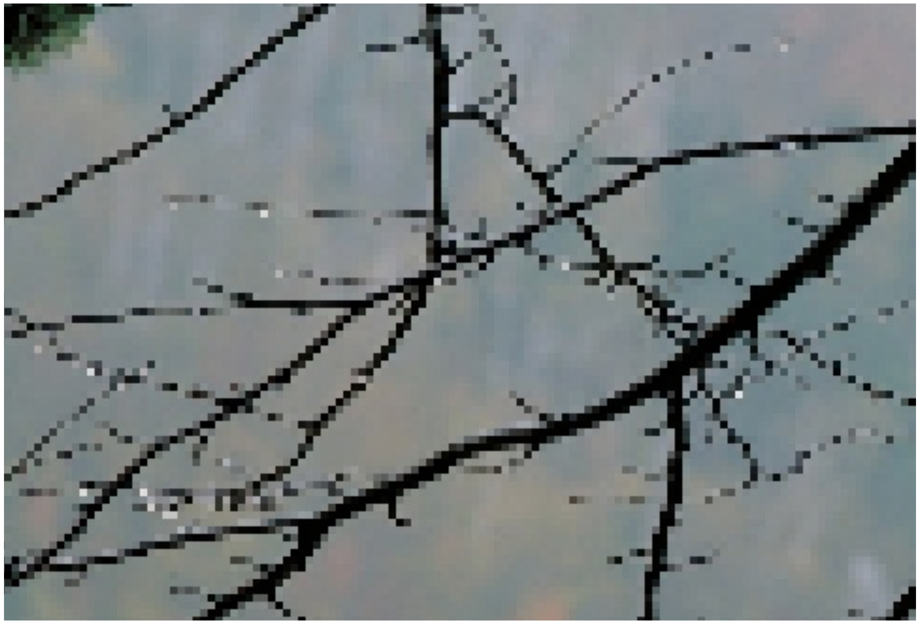
$$f_g(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}.$$

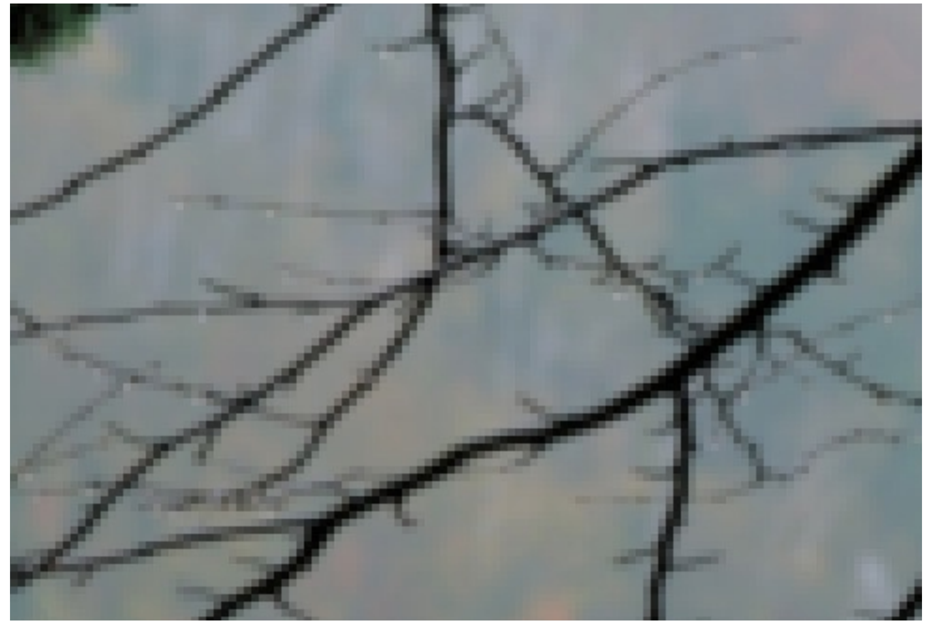
Reducing and enlarging

- Very common operation
  - –devices have differing resolutions
  - –applications have different memory/quality tradeoffs
- Also very commonly done poorly
- Simple approach: drop/replicate pixels
- Correct approach: use resampling

[Philip Greenspun]

1000 pixel width



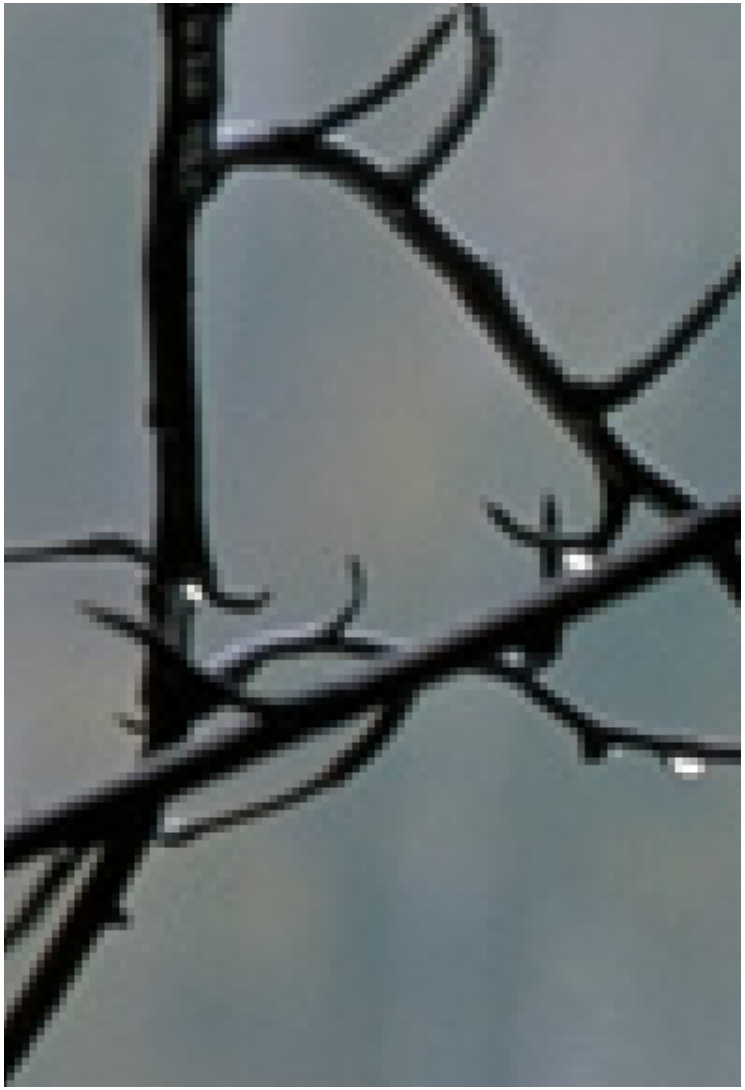


[Philip Greenspun]

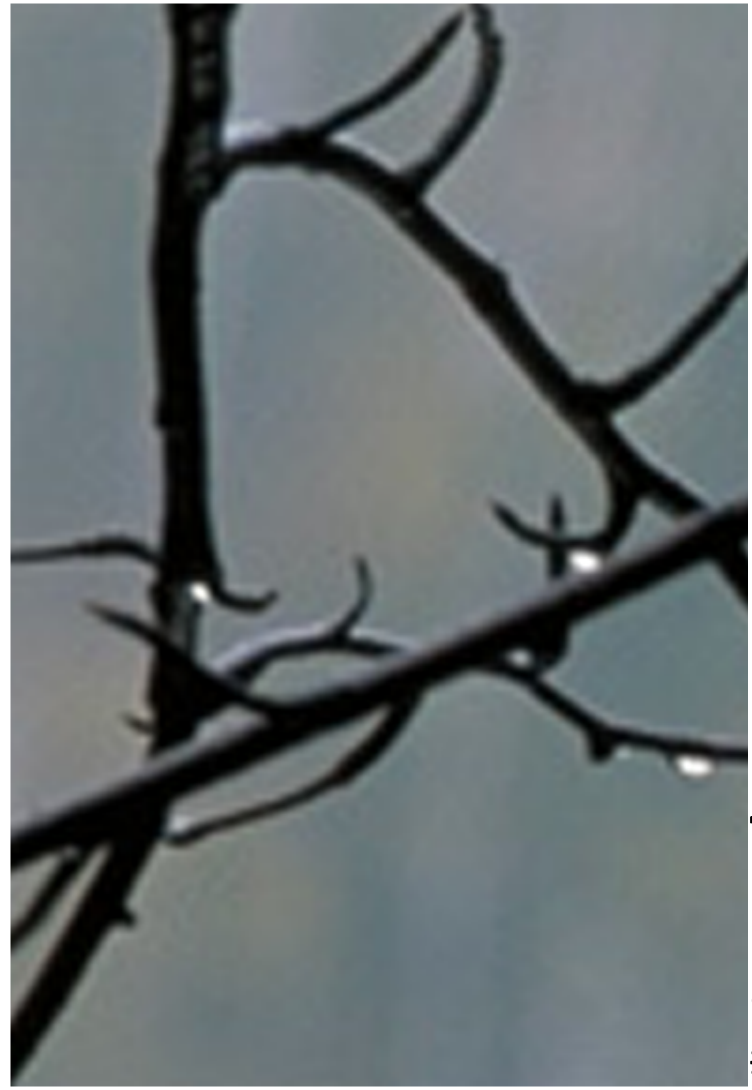
by dropping pixels

gaussian filter

250 pixel width



box reconstruction filter



bicubic reconstruction filter

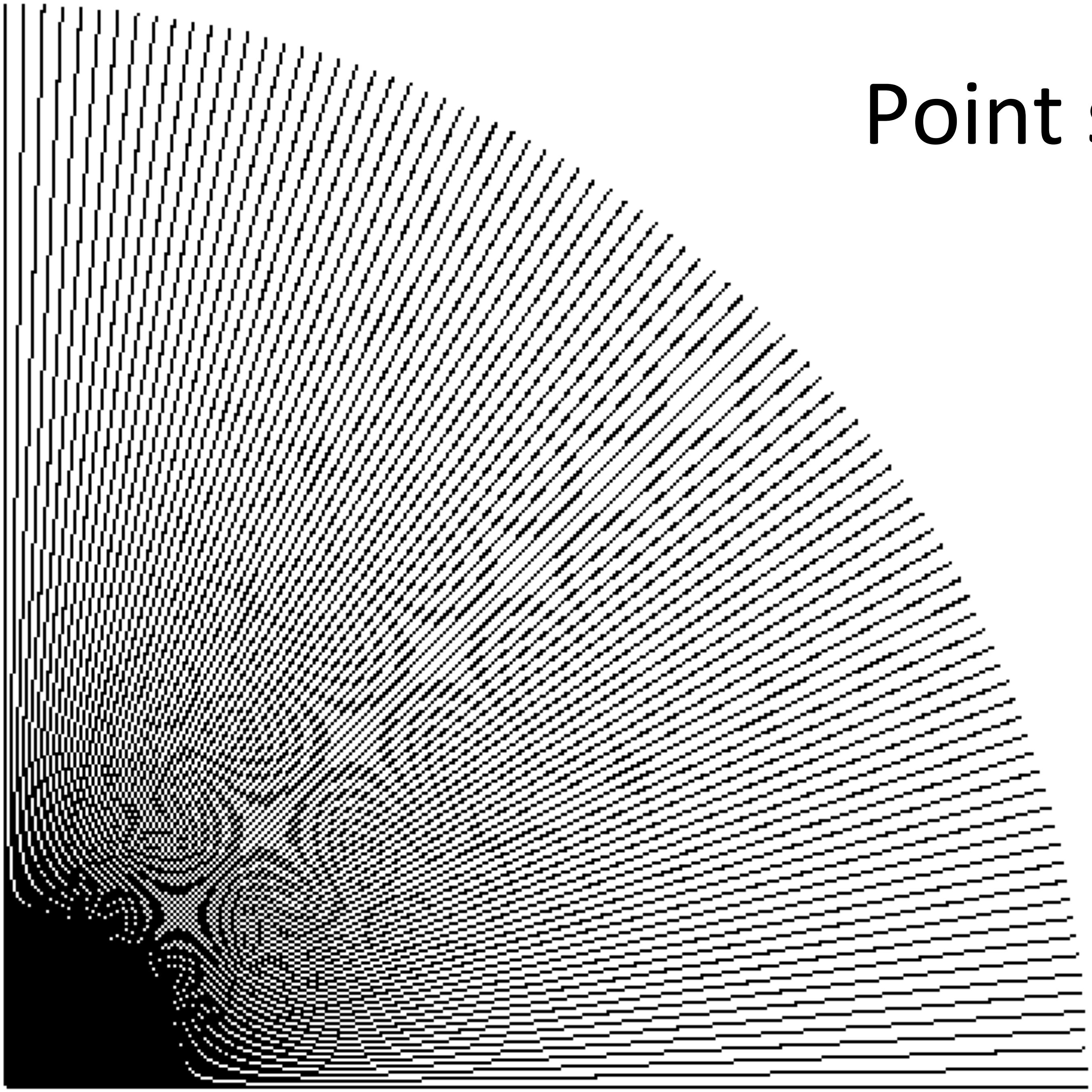
4000 pixel width

[Philip Greenspun]

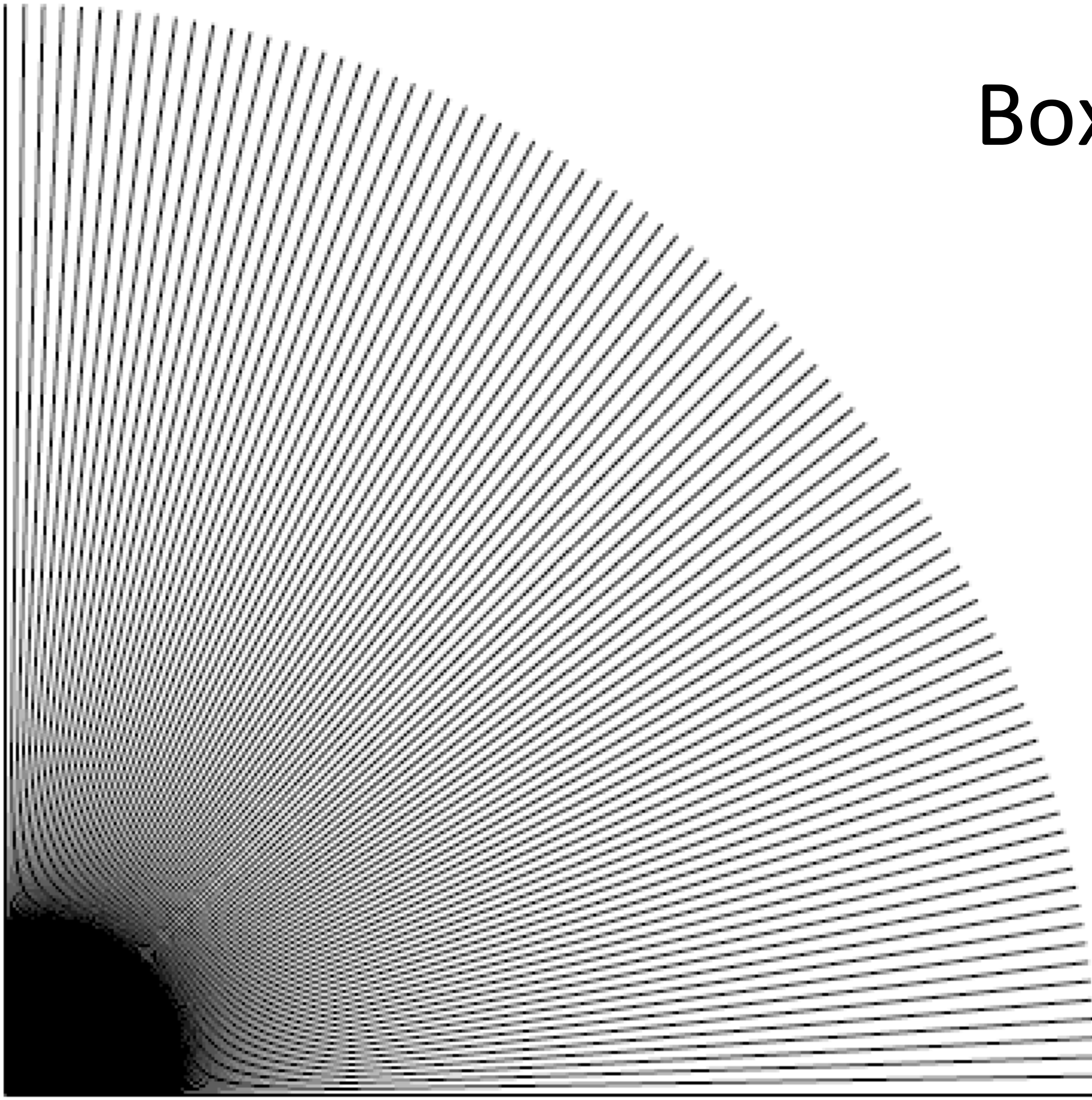
[Philip Greenspun]

original △|▽ box blur

sharpened △|▽ gaussian blur



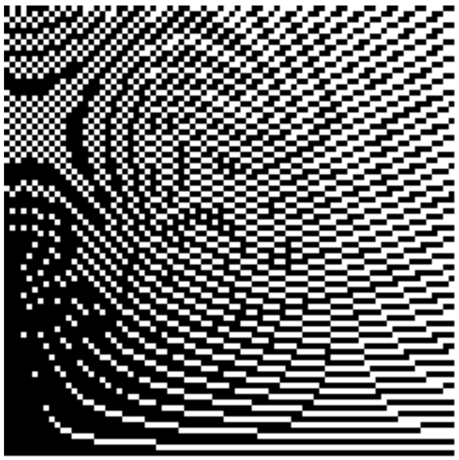
Point sampling in action



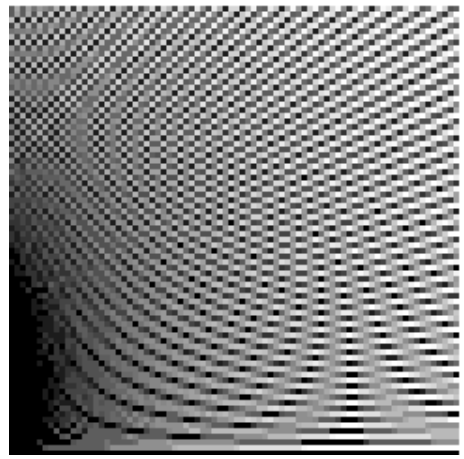
Box filtering in action

Gaussian filtering in action

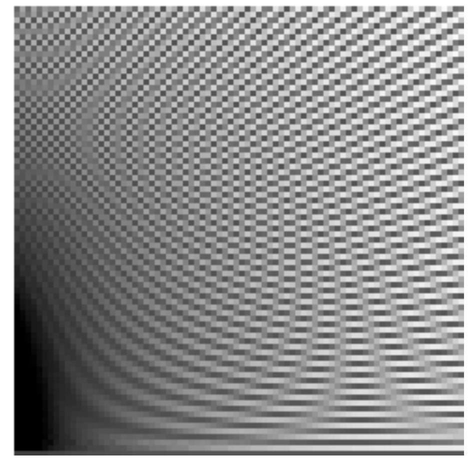
Filter comparison



Point sampling



Box filtering



Gaussian filtering