CS559: Computer Graphics

Lecture 3: Image Sampling and Filtering

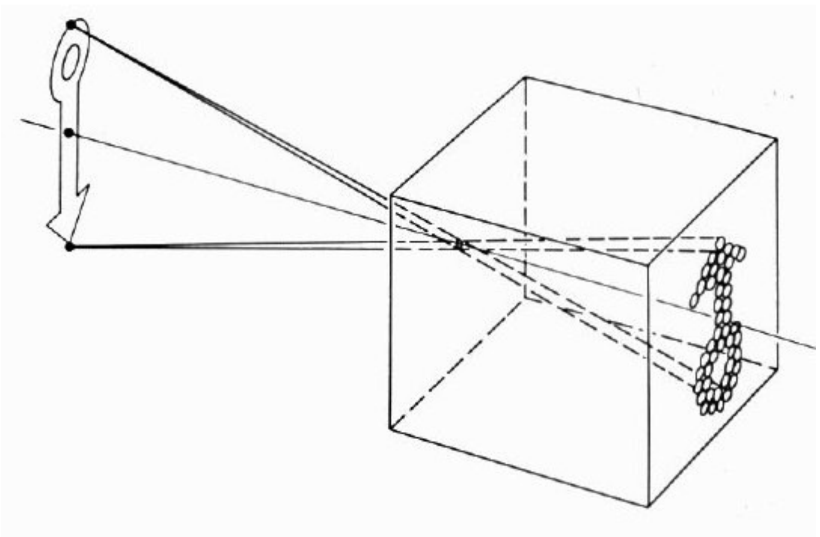
Li Zhang

Spring 2010

Announcement

- Today's office hour moves to 4-5pm this Friday.

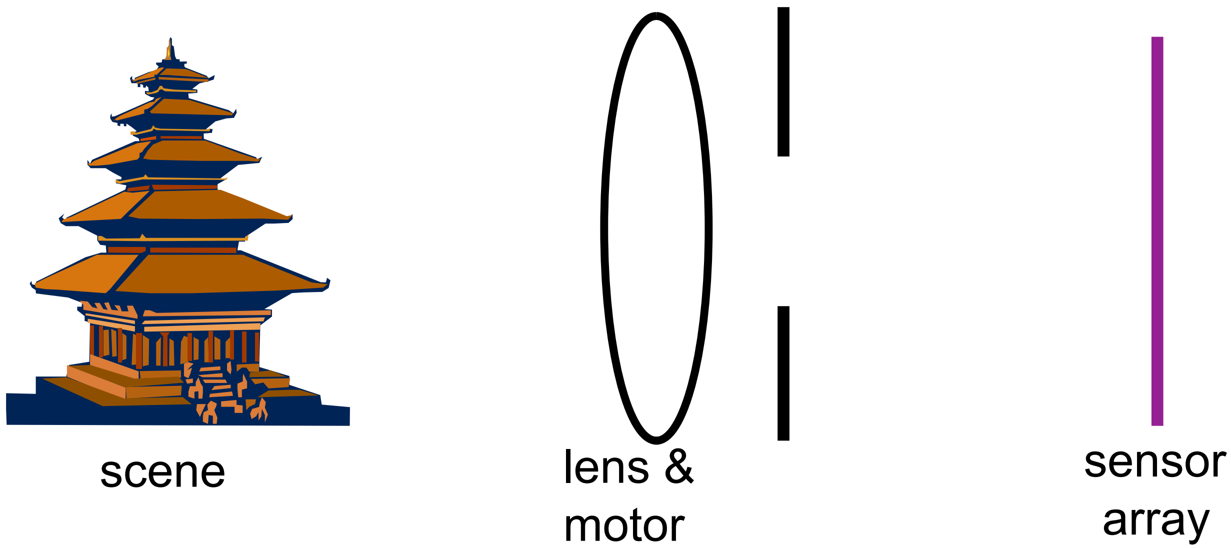
Last time: Image Formation in Cameras



- The first camera
  - 5th B.C. Aristotle, Mozi (Chinese: 墨子)
  - How does the aperture size affect the image?

http://en.wikipedia.org/wiki/Pinhole_camera

Last time: Image Formation in Cameras



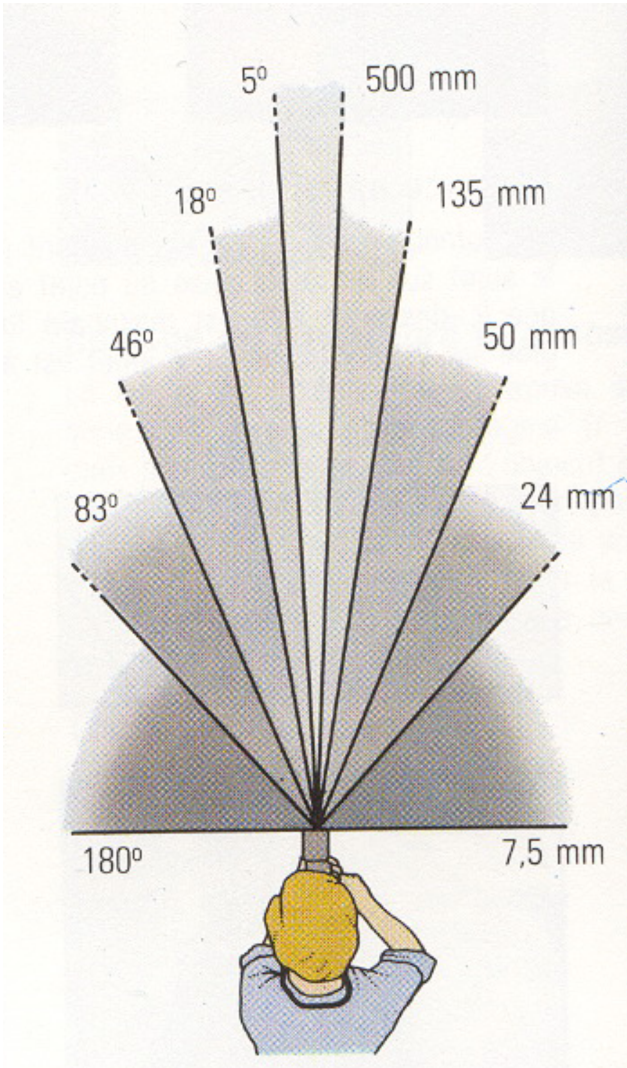
- A digital camera replaces film with a sensor array
- Each cell in the array is a light-sensitive diode that converts photons to electrons

YungYu Chuang's slide

Last time: Image Formation in Cameras

Canon EF-S 60mm f/2.8 | Canon EF 100mm f/2.8 | Canon EF 180mm f/3.5

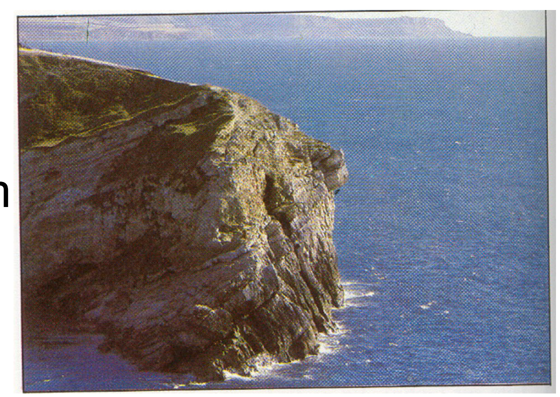
Last time: Image Formation in Cameras



24mm

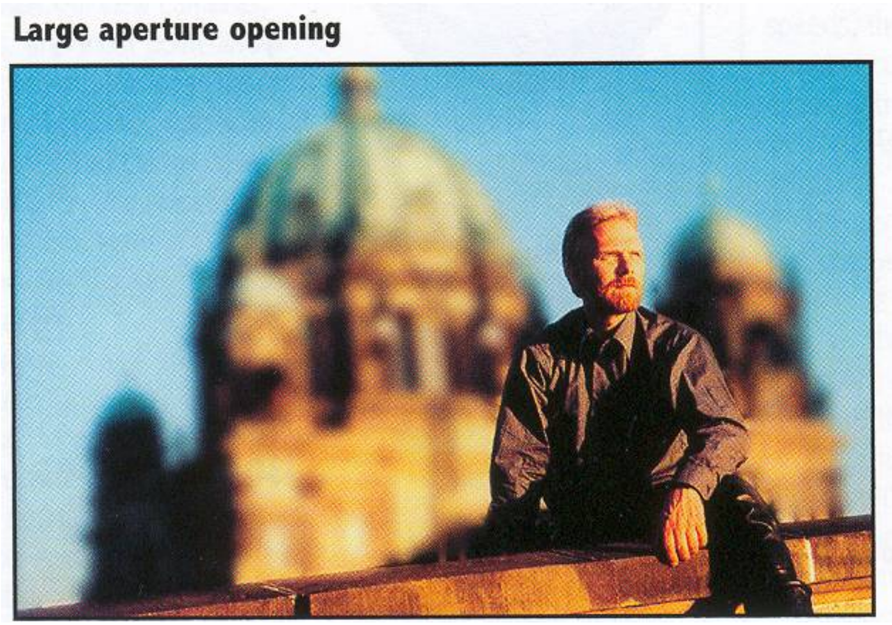
50mm

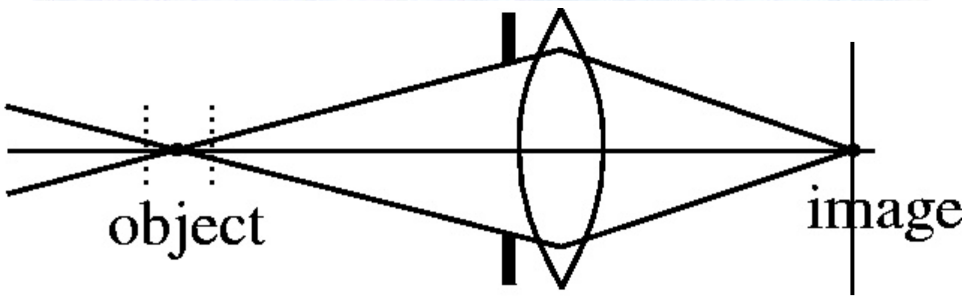
135mm

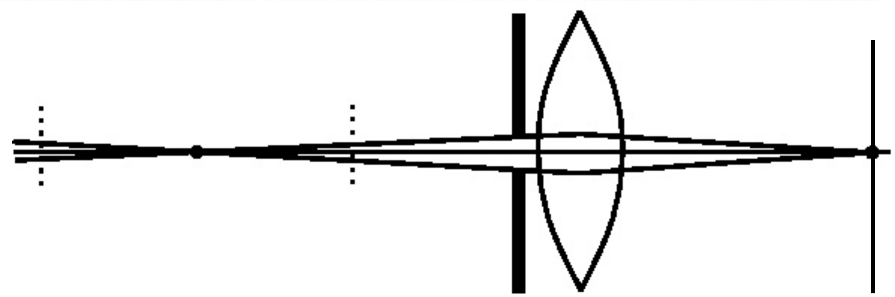


Frédo Durand's slide

Last time: Image Formation in Cameras



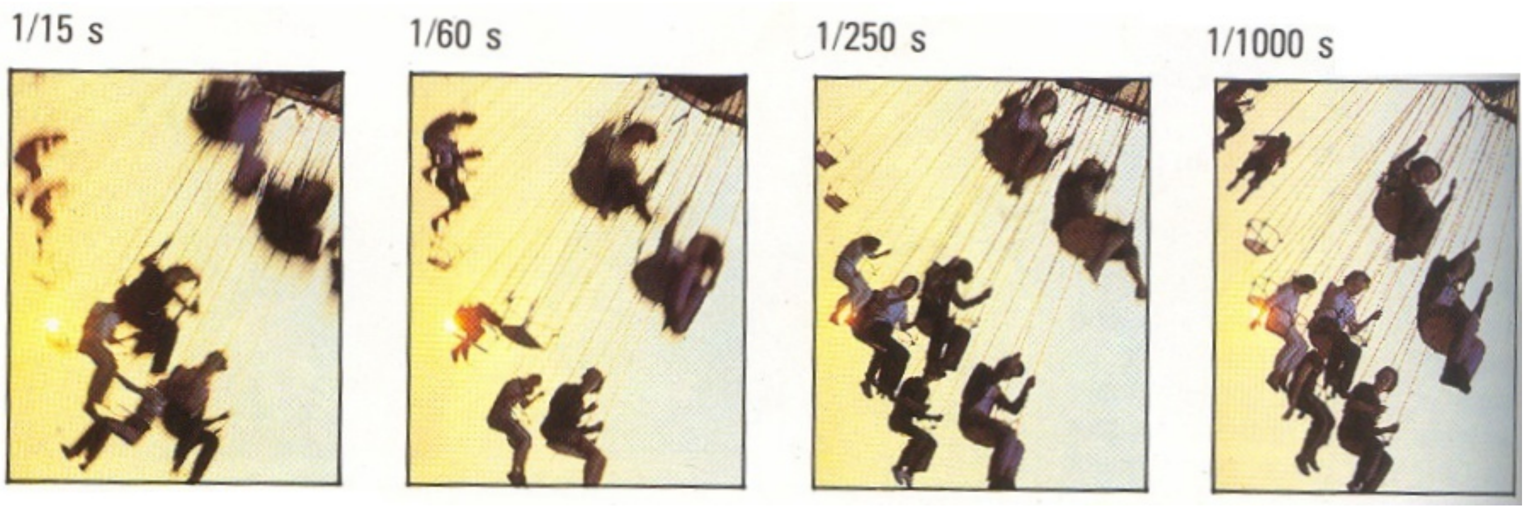




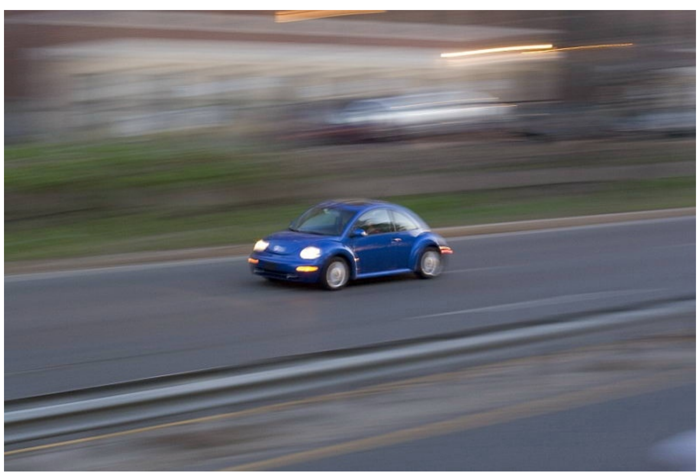
Changing the aperture size affects depth of field. A smaller aperture increases the range in which the object is approximately in focus

Last time: Image Formation in Cameras

- Slower shutter speed => more light, but more motion blur



- Faster shutter speed freezes motion

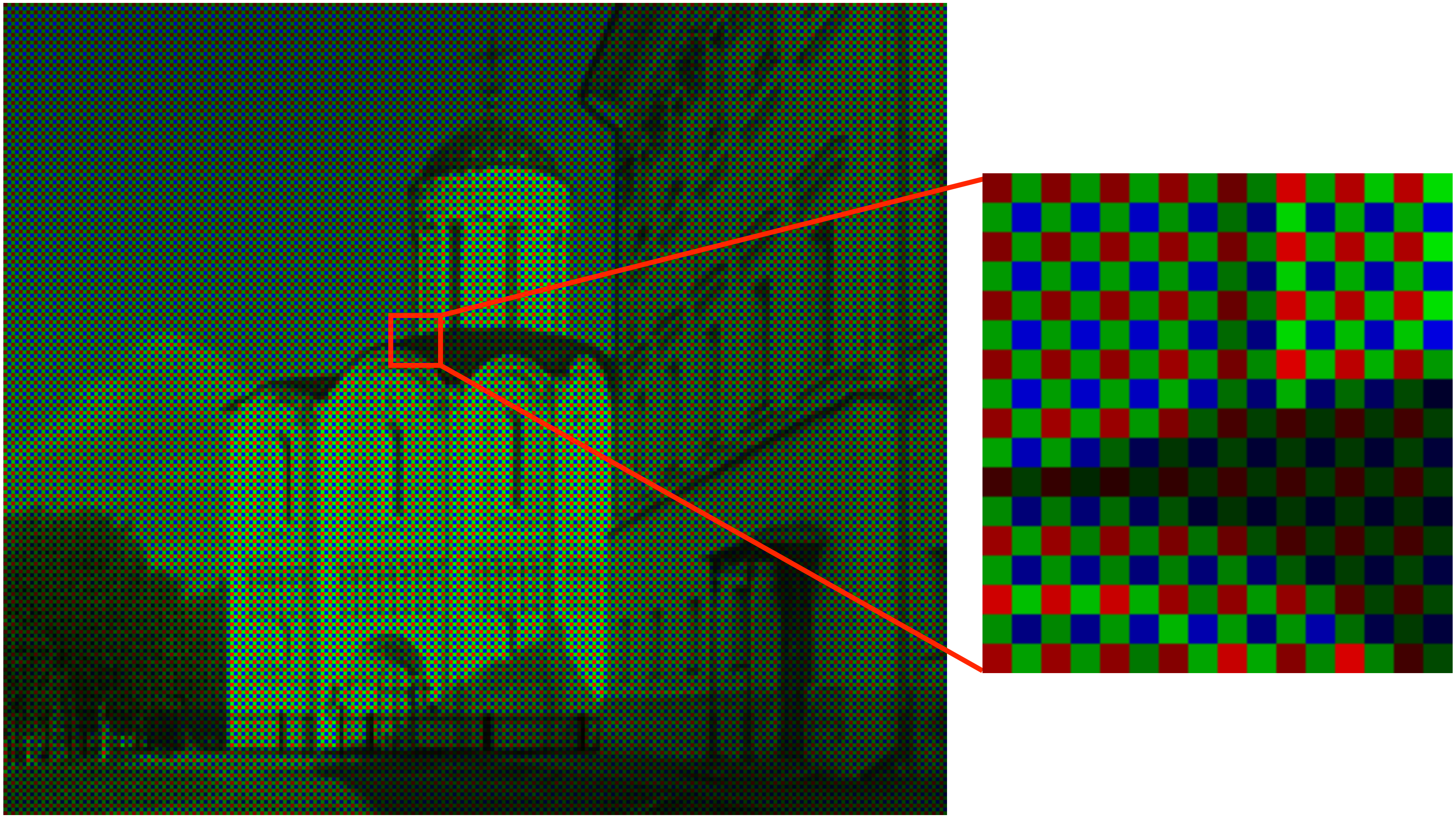


YungYu Chuang's slide

Last time: Image Formation in Cameras

- Field of View, Motion blur, Depth of Field
- Can all be simulated in OpenGL

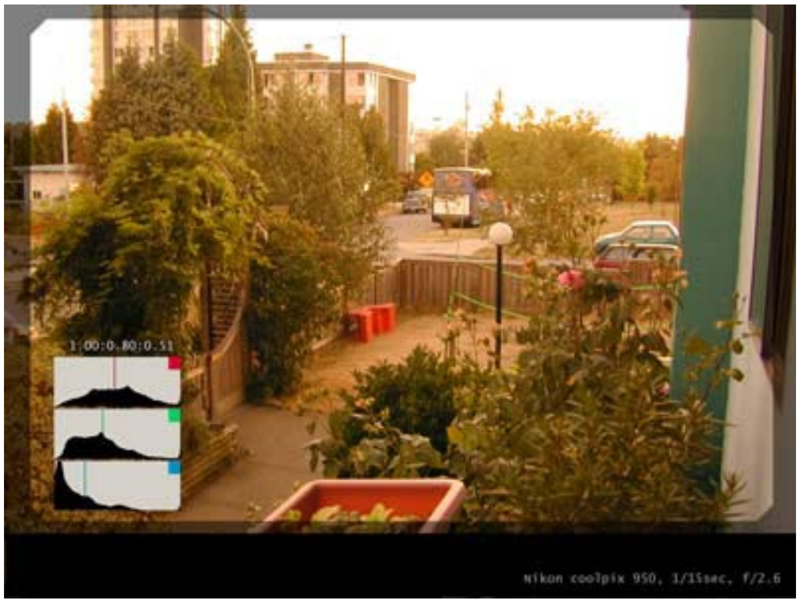
Last time: Image Formation in Cameras



Lecture 3-4: Image Re-sampling and Filtering

YungYu Chuang's slide

Last time: Image Formation in Cameras



warmer

automatic white balance

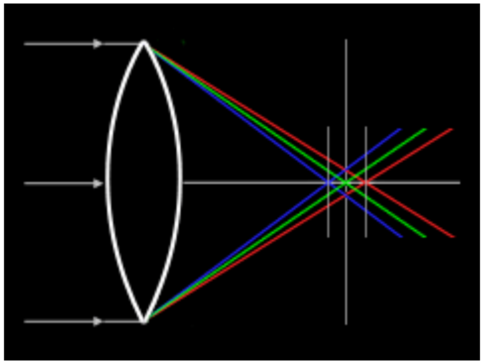
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 255/R'_w & 0 & 0 \\ 0 & 255/G'_w & 0 \\ 0 & 0 & 255/B'_w \end{bmatrix} \begin{bmatrix} R' \\ G' \\ B' \end{bmatrix}$$

YungYu Chuang's slide

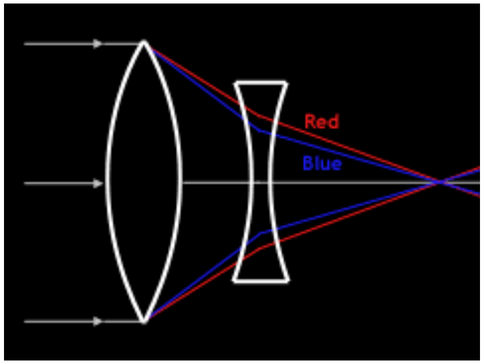
Last time: Image Formation in Cameras

- Bayer Pattern => color image, white balance
- Are good exercises for project 1.

Lens related issues: Chromatic Abberation



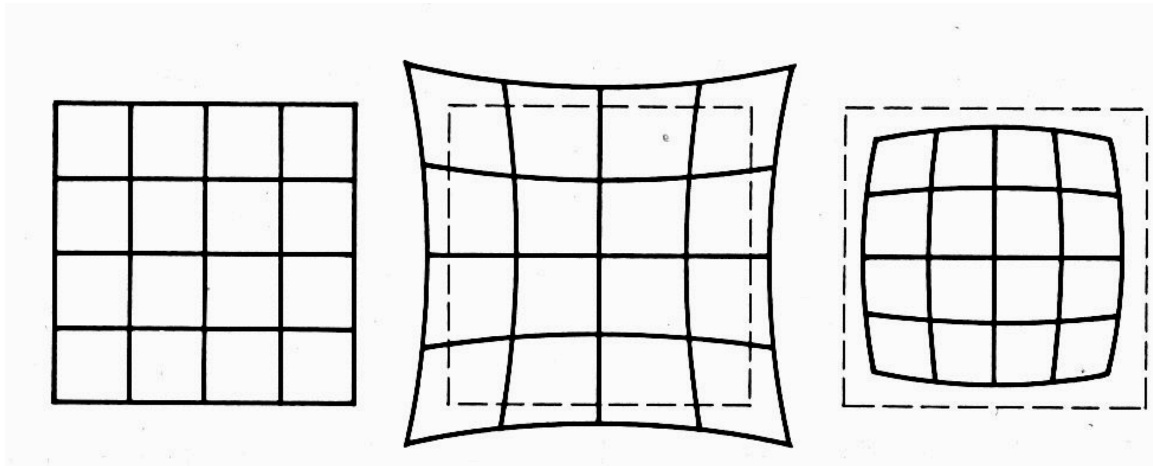
Lens has different refractive indices for different wavelengths.



Special lens systems using two or more pieces of glass with different refractive indexes can reduce or eliminate this problem.

http://www.dpreview.com/learn/?/Glossary/Optical/chromatic_aberration_01.htm

Lens related issues: Distortion



No distortion Pin cushion Barrel

- Radial distortion of the image
  - Caused by imperfect lenses
  - Deviations are most noticeable for rays that pass through the edge of the lens

Steve Seitz's slide

Correcting radial distortion

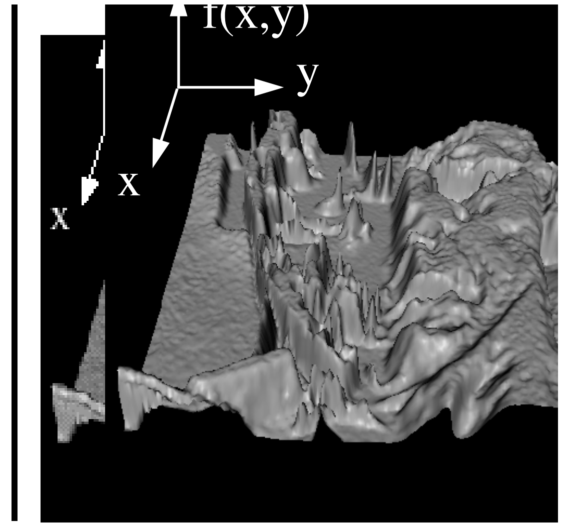
Lecture 6: Image Warping from Helmut Dersch

Steve Seitz's slide

Digital camera review website

- http://www.dpreview.com/
- http://www.imaging-resource.com/
- http://www.steves-digicams.com/

Image as a discreet function



Represented by a matrix:

j →

i ↓

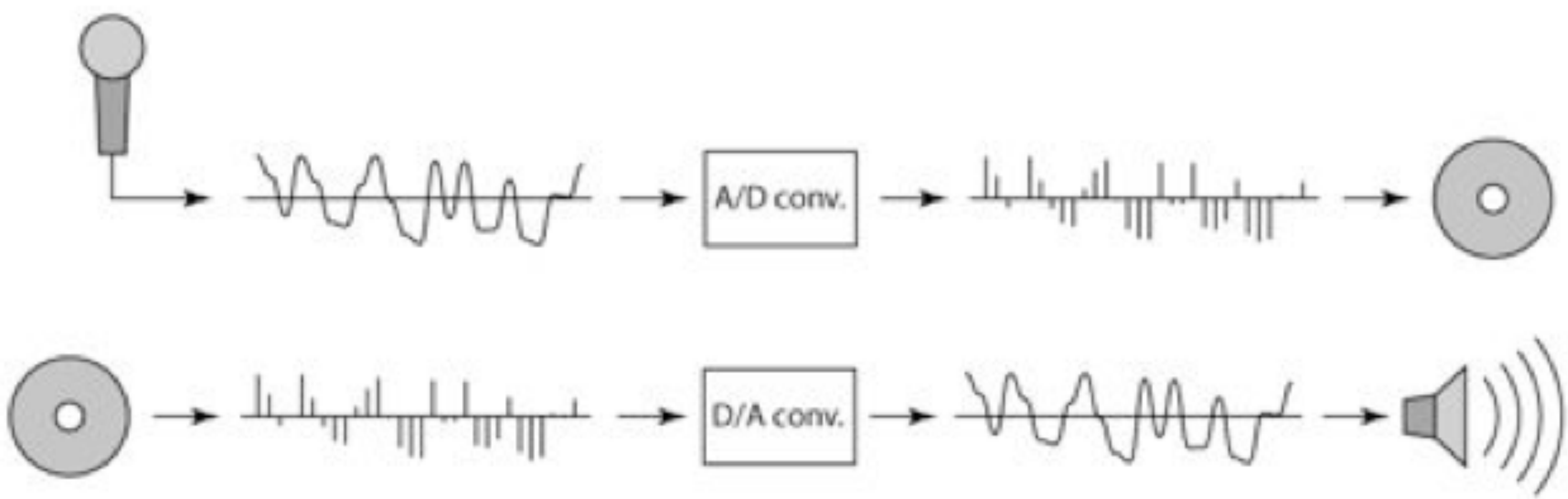
62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

Q1: How many discrete samples are needed to represent the original continuous function?

Q2: How to reconstruct the continuous function from the samples?

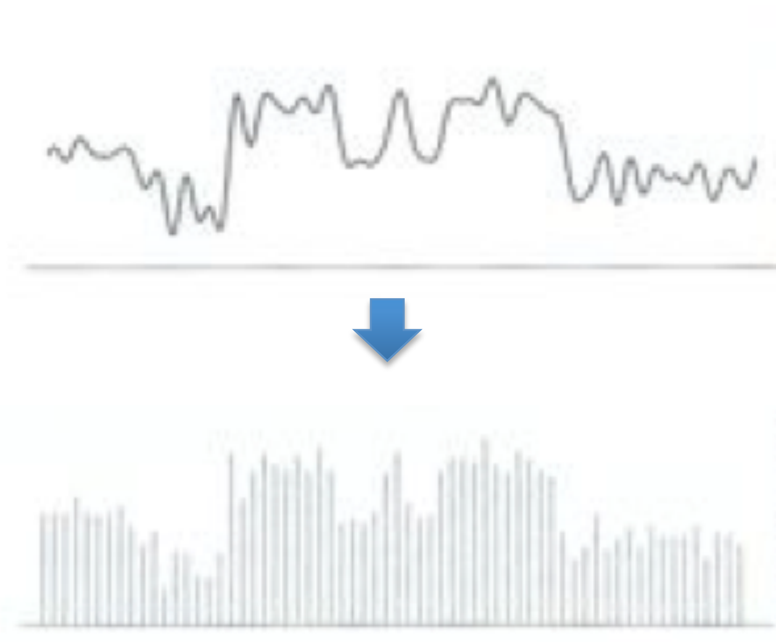
Sampling in digital audio

- Recording: sound to analog to samples to disc
- Playback: disc to samples to analog to sound again
  - How can we make sure we are filling in the gaps correctly?



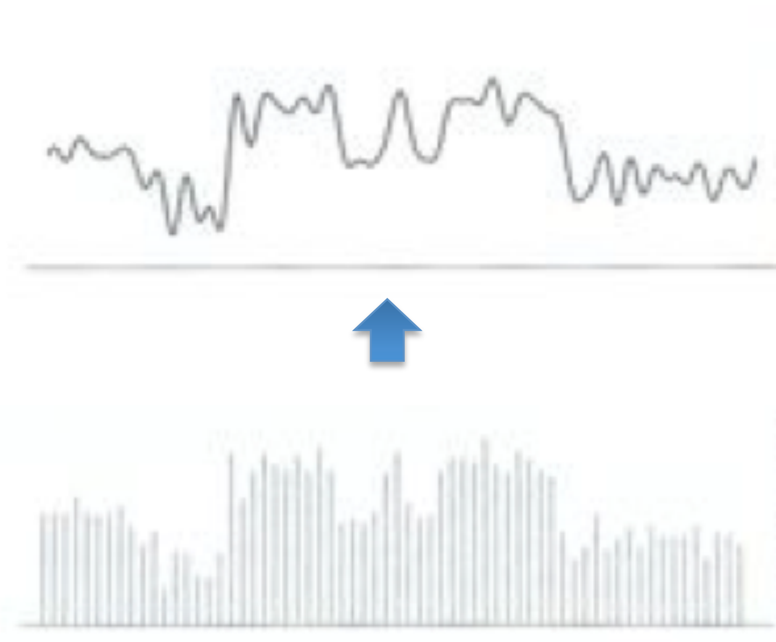
Sampled Representation in General

- How to store and compute with continuous functions?
- Sampling: write down the function's values at many points



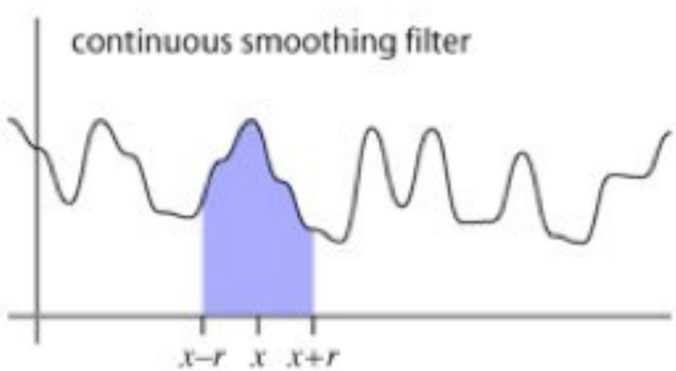
Sampled Representation in General

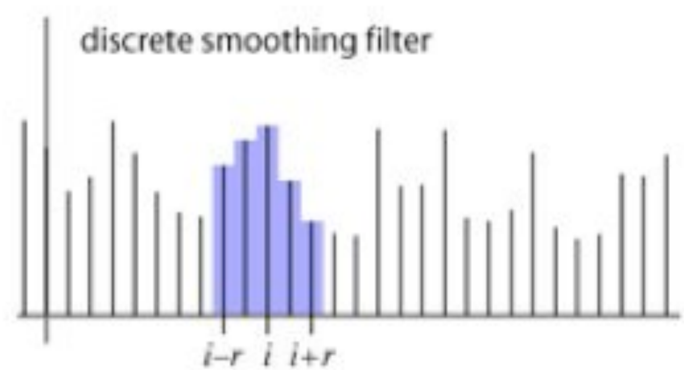
- Making samples back into a continuous function
  - For output
  - For analysis or processing
- Amounts to guessing what the function did in between



Advantage of sampled representation

- Simplifying the job of processing a function
- Simple example: smoothing by averaging
  - Can be executed in continuous form (analog circuit design)
  - But can also be executed using sampled representation





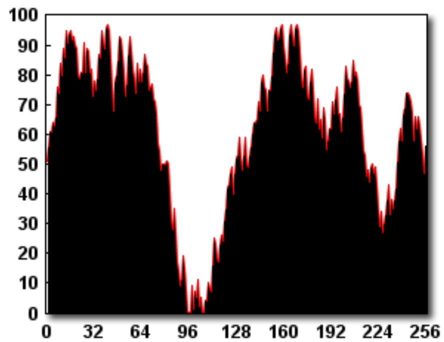
History of sampling

- 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 terminology has ECE flavor instead of CS

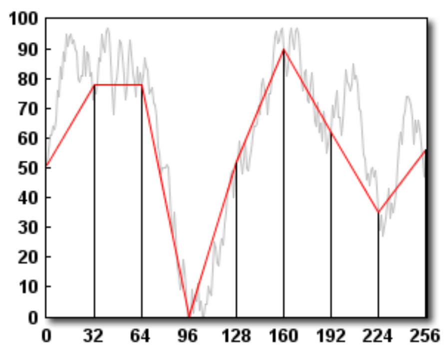
- Compressed Sensing 2004; sub-Nyquiest-Shannon criterion

Sampling a continuous function (1D)

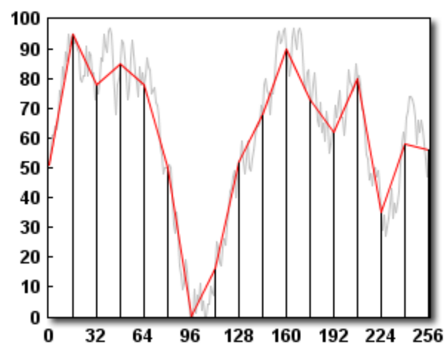
Continuous Function



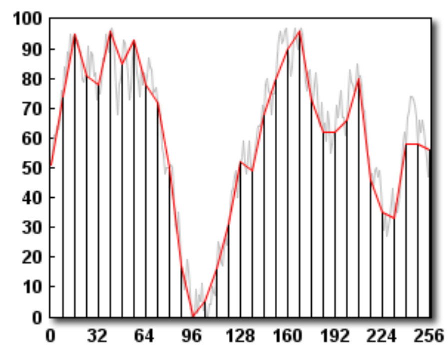
Discrete Samples



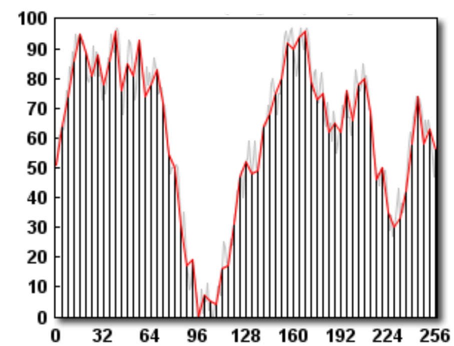
Sampling Period T = 32



Sampling Period T = 16



Sampling Period T = 8

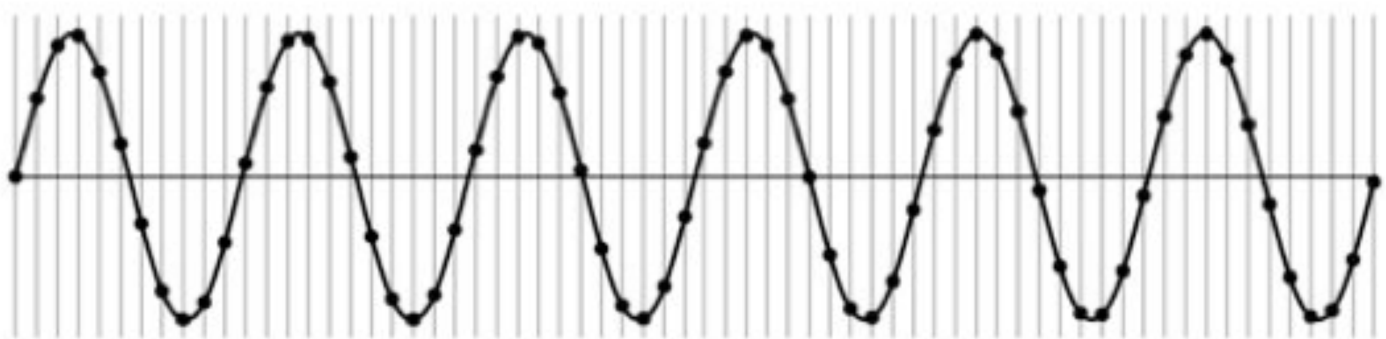


Sampling Period T = 4

The denser the better, but at the expense of storage and processing power

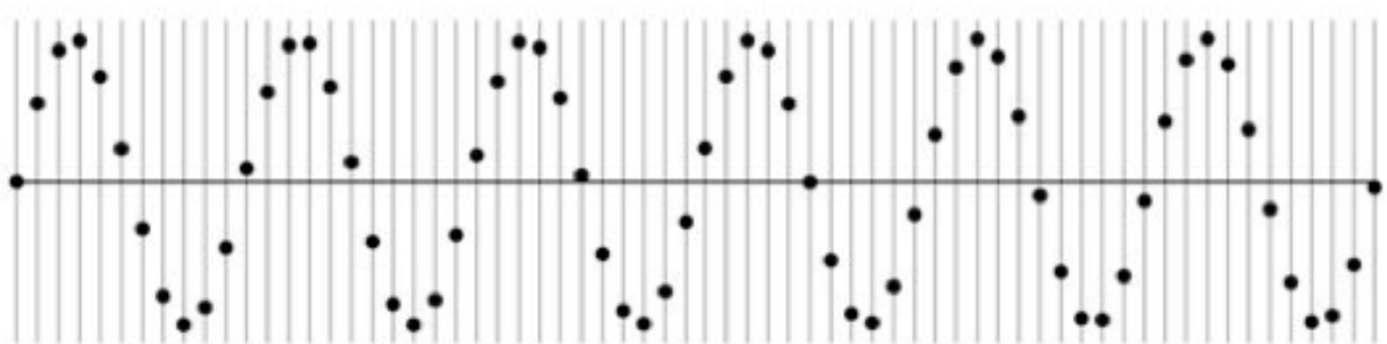
Under-sampling

- Sampling a sine wave

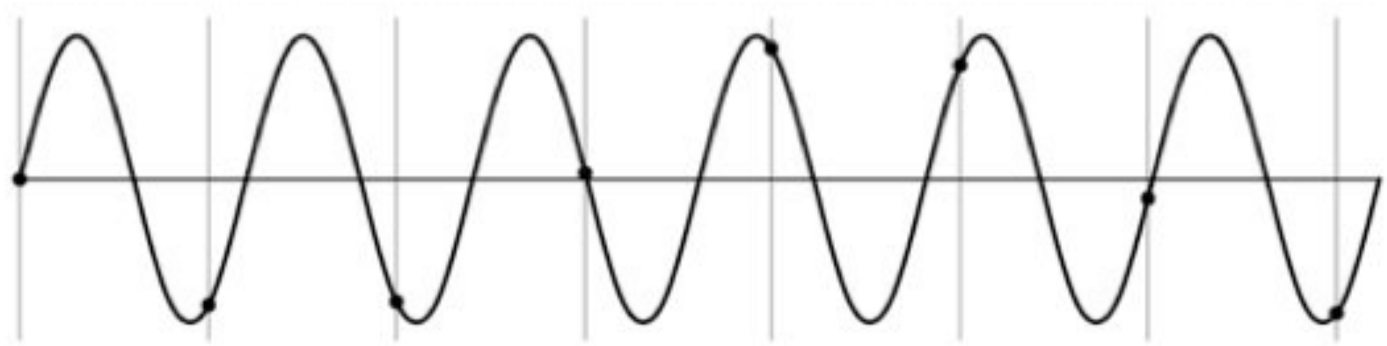


Under-sampling

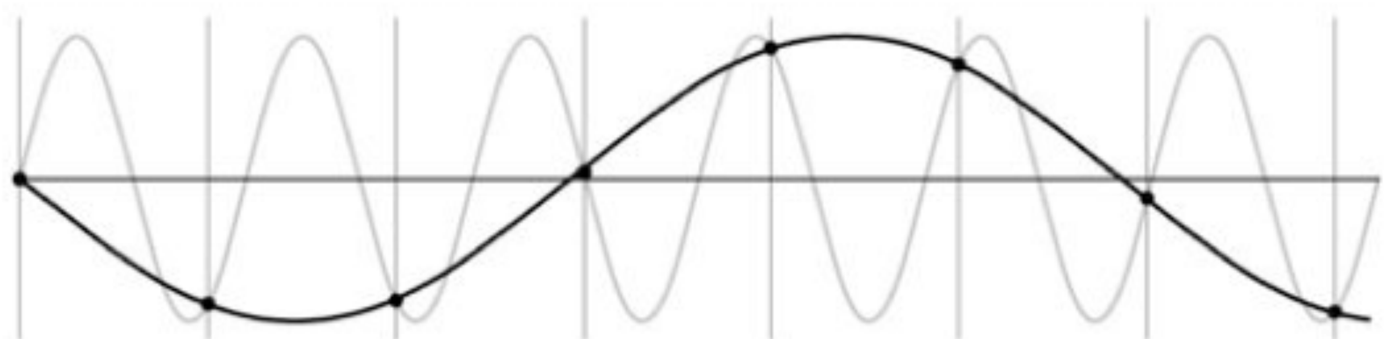
- Sampling a sine wave
- What if we "missed" things between the samples?
  - Unsurprising result: information is lost



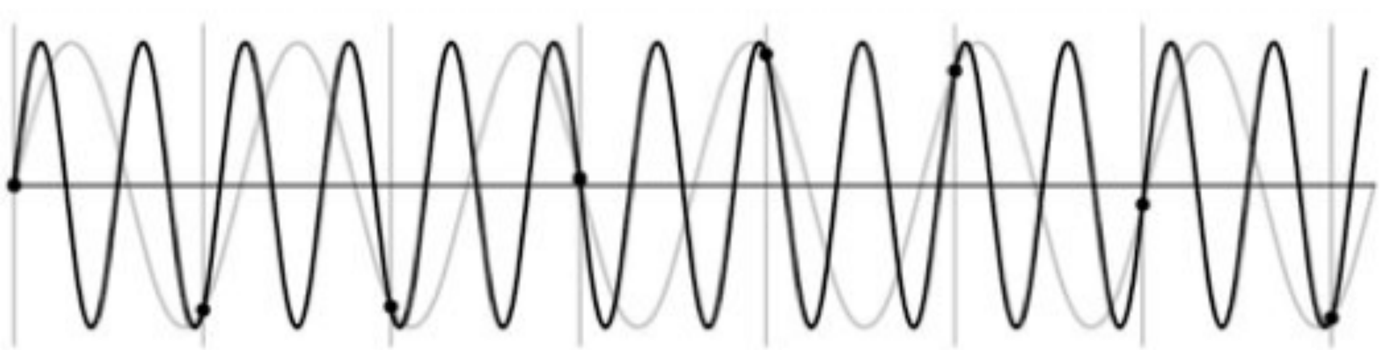
- Sampling a sine wave
- What if we "missed" things between the samples?
  - Unsurprising result: information is lost
  - Surprising result: indistinguishable from lower frequency



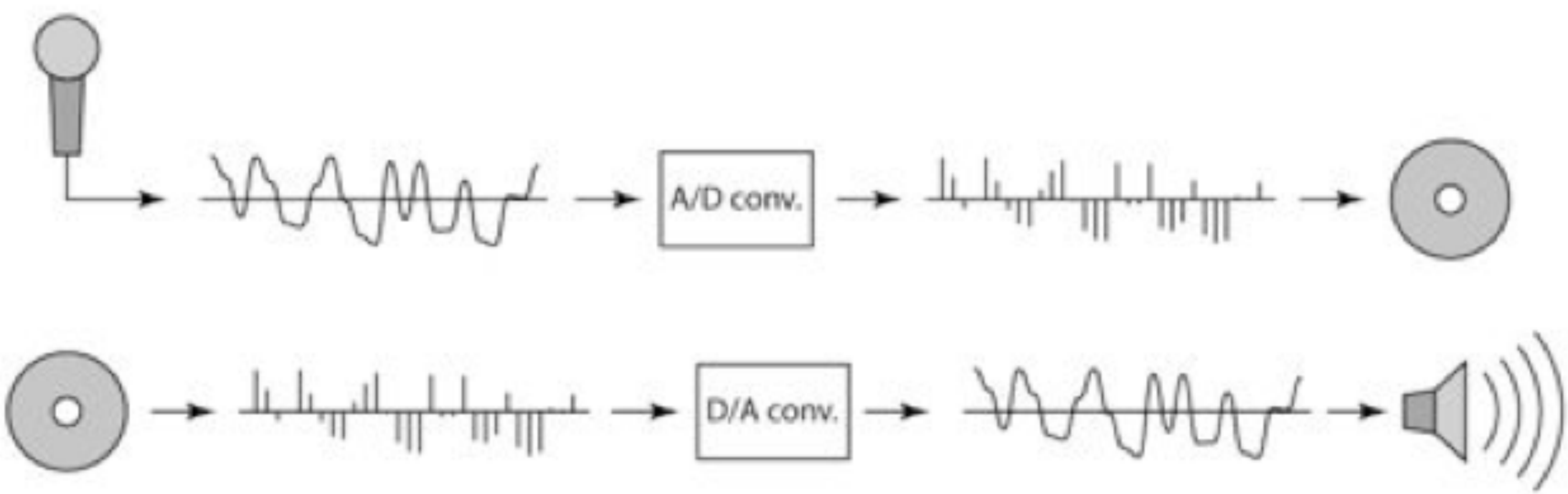
- Sampling a sine wave
- What if we “missed” things between the samples?
  - Unsurprising result: information is lost
  - Surprising result: indistinguishable from lower frequency



- Sampling a sine wave
- What if we "missed" things between the samples?
  - Unsurprising result: information is lost
  - Surprising result: indistinguishable from lower frequency
  - Also indistinguishable from high frequency
  - *Aliasing:* Insufficient samples to reconstruct original signal



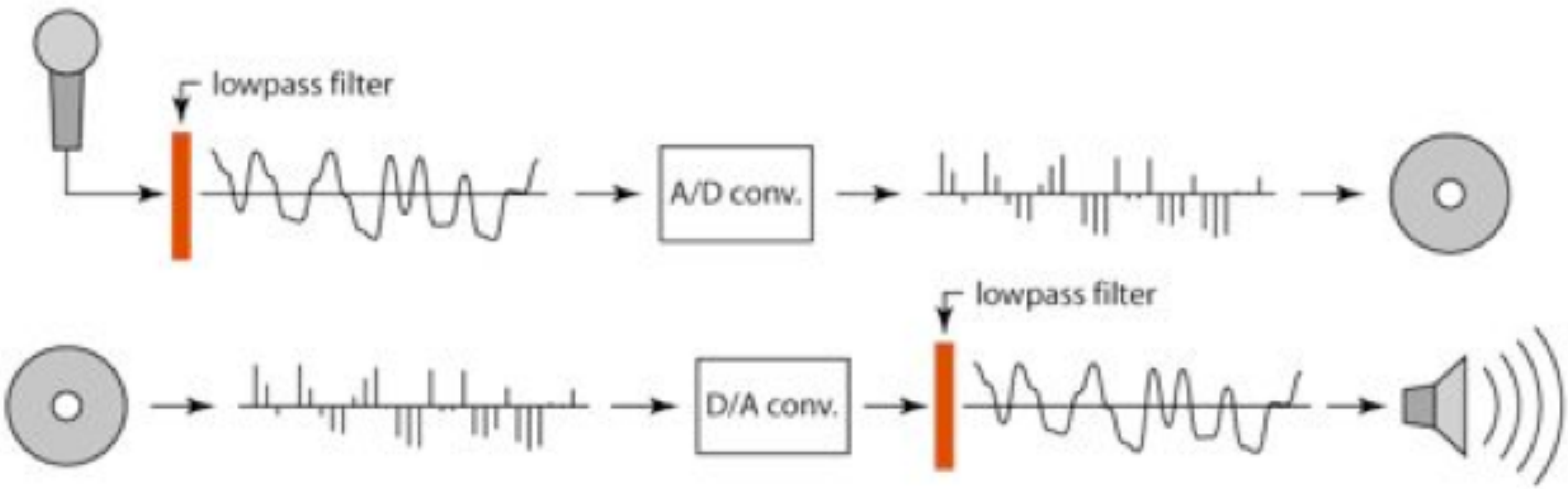
Preventing aliasing



Preventing aliasing

Introducing lowpass filters:

- remove high frequency leaving only safe low frequencies
- choose lowest frequency in reconstruction (disambiguate)



Linear filtering: a key idea

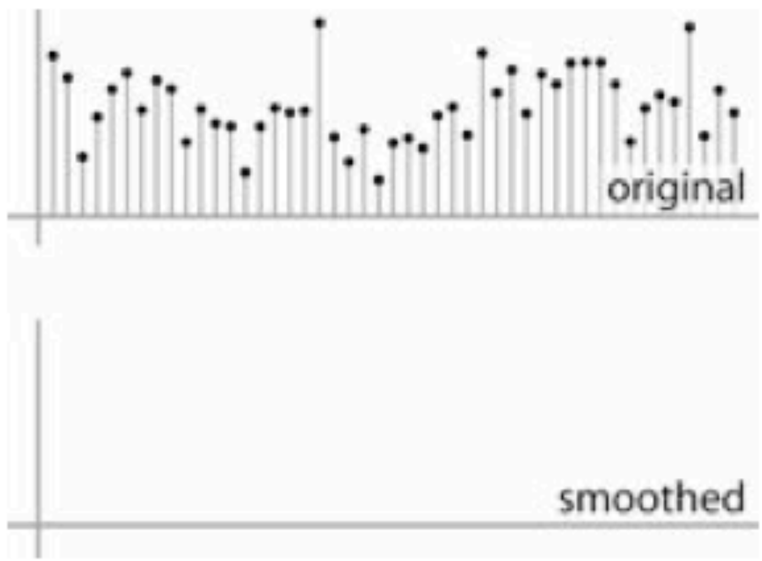
Transformation on signals; e.g.:

- bass/treble controls on stereo
- blurring/sharpening operations in image editing
- smoothing/noise reduction in tracking

Can be mathematically by *convolution*

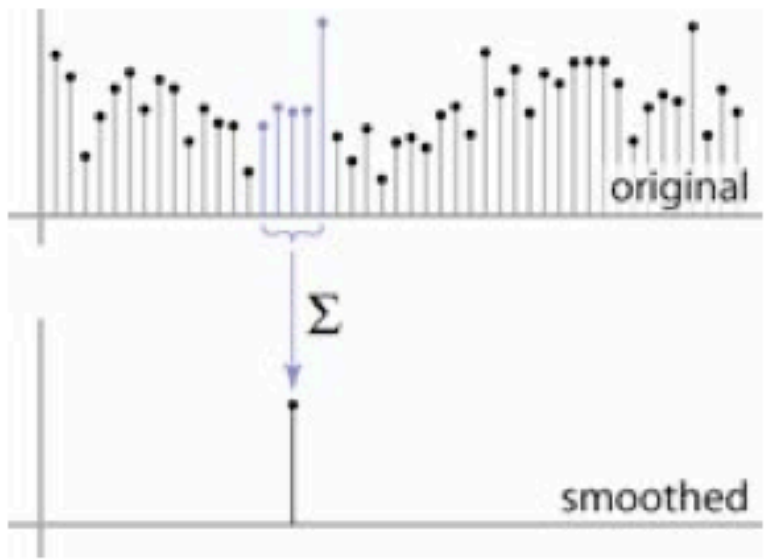
Convolution warm-up

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



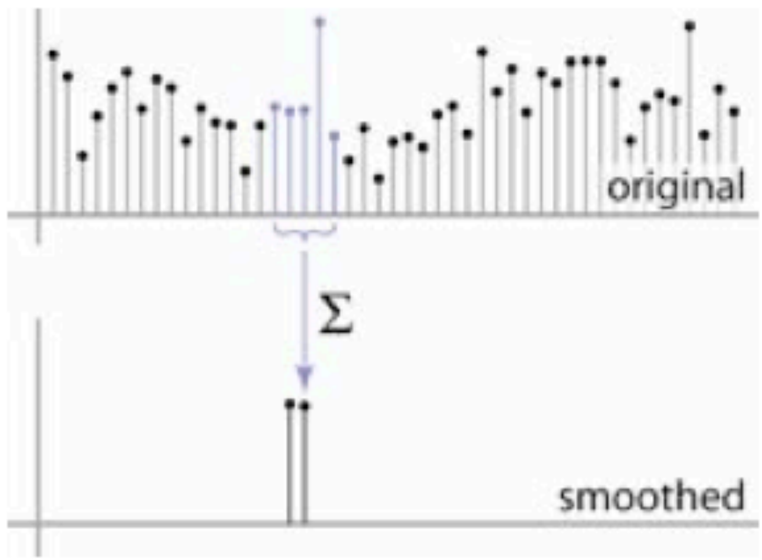
Convolution warm-up

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



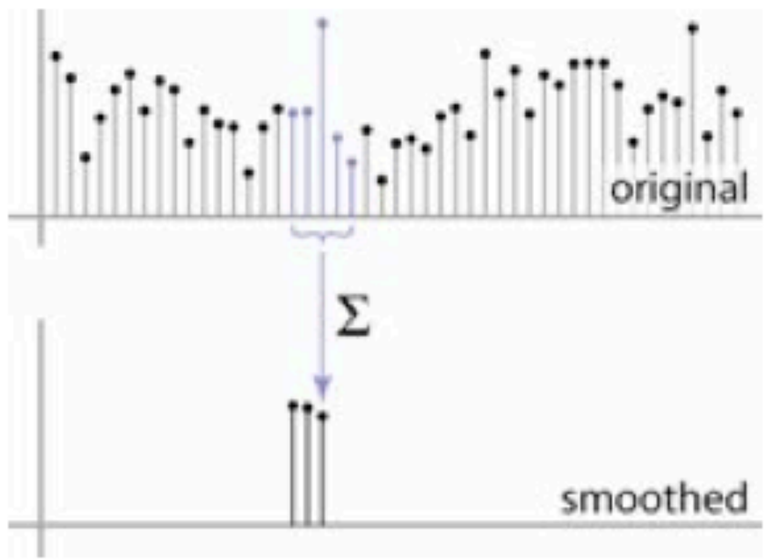
Convolution warm-up

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



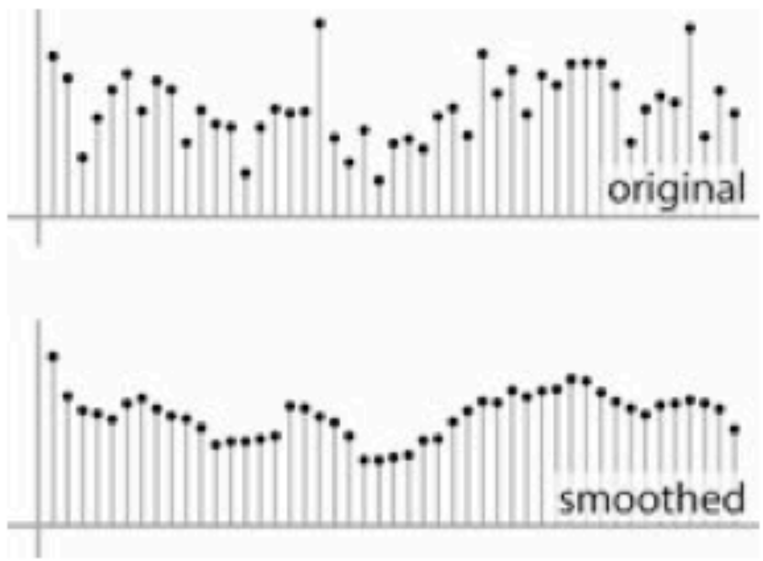
Convolution warm-up

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



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

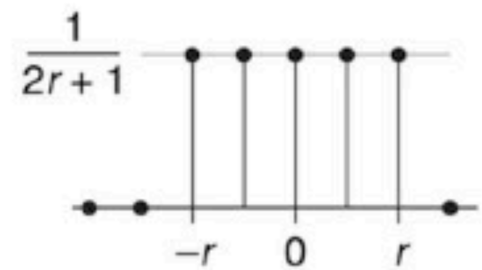
$$(a \star b)[i] = \sum_j a[j]b[i-j]$$

each sample gets its own weight (normally zero far away)

- This is all convolution is: it is a **moving weighted average**

Filters

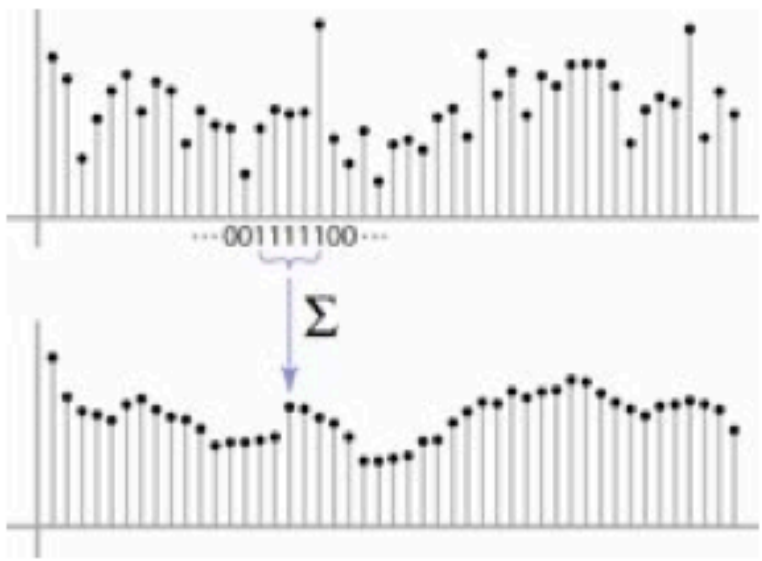
- Sequence of weights $a[j]$ is called a *filter*
- Filter is nonzero over its *region of support*
  - usually centered on zero: support radius r
- Filter is *normalized* so that it sums to 1.0
  - this makes for a weighted average, not just any old weighted sum
- Most filters are symmetric about 0
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



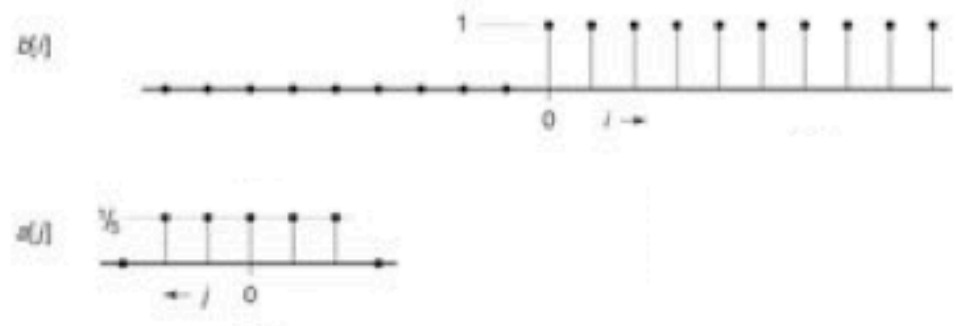
a box filter

Convolution and filtering

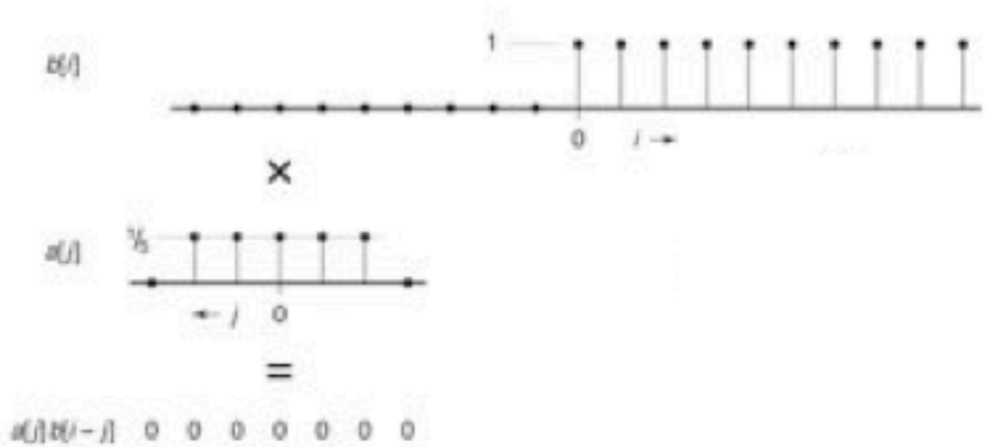
- Can express sliding average as convolution with a *box filter*
- $a_{\text{box}} = [\ldots, 0, 1, 1, 1, 1, 1, 0, \ldots]$



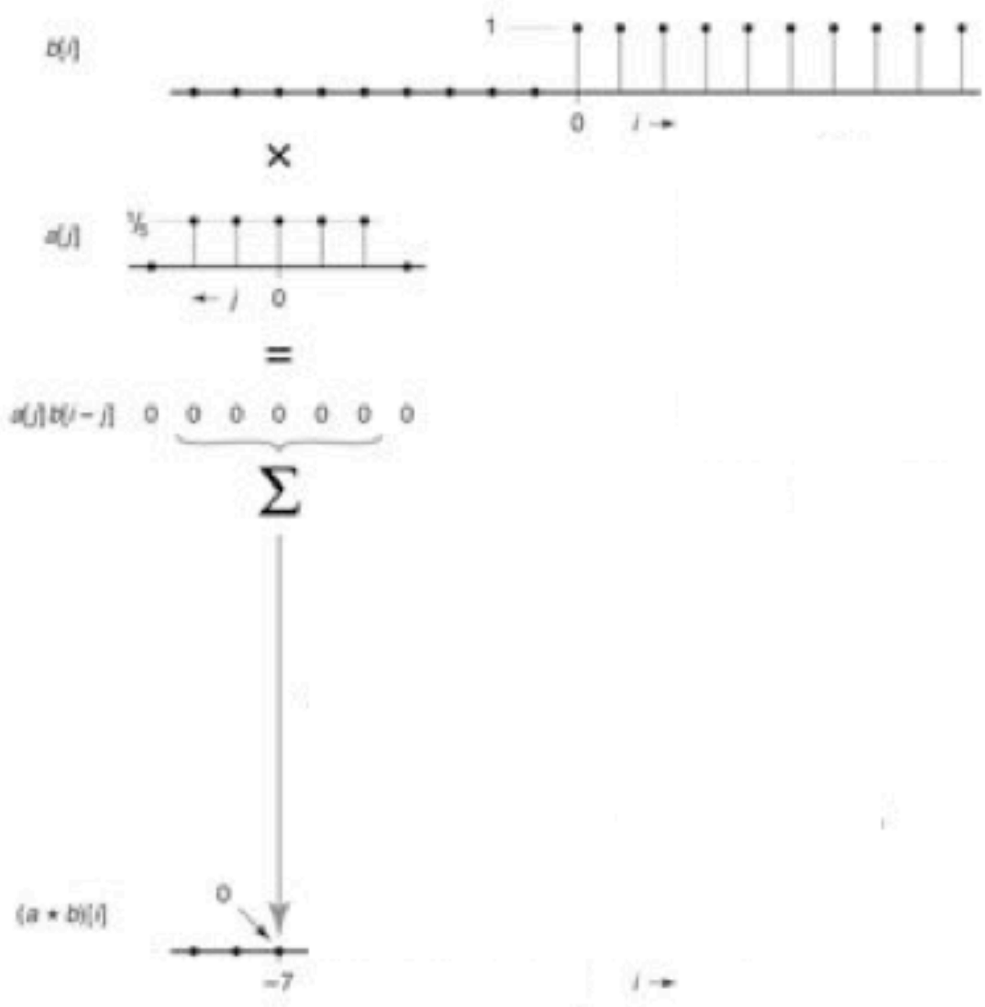
Example: box and step



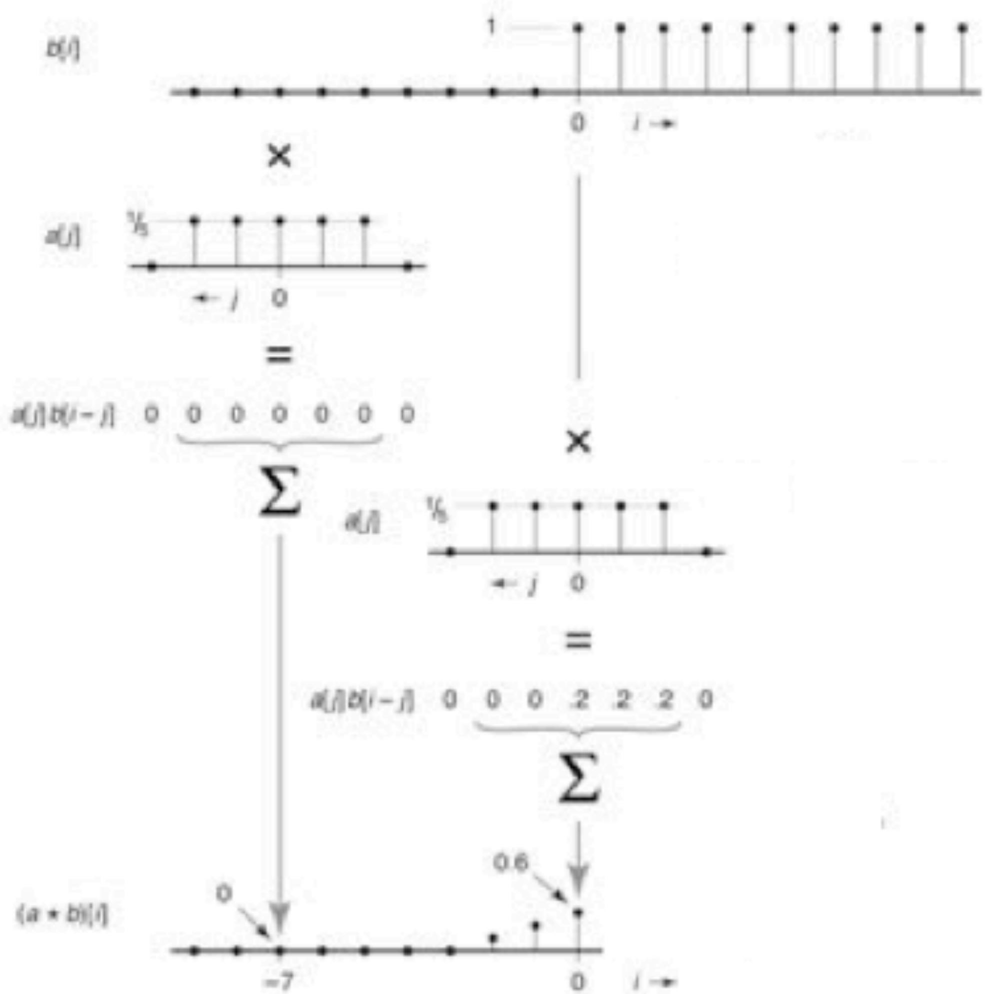
Example: box and step



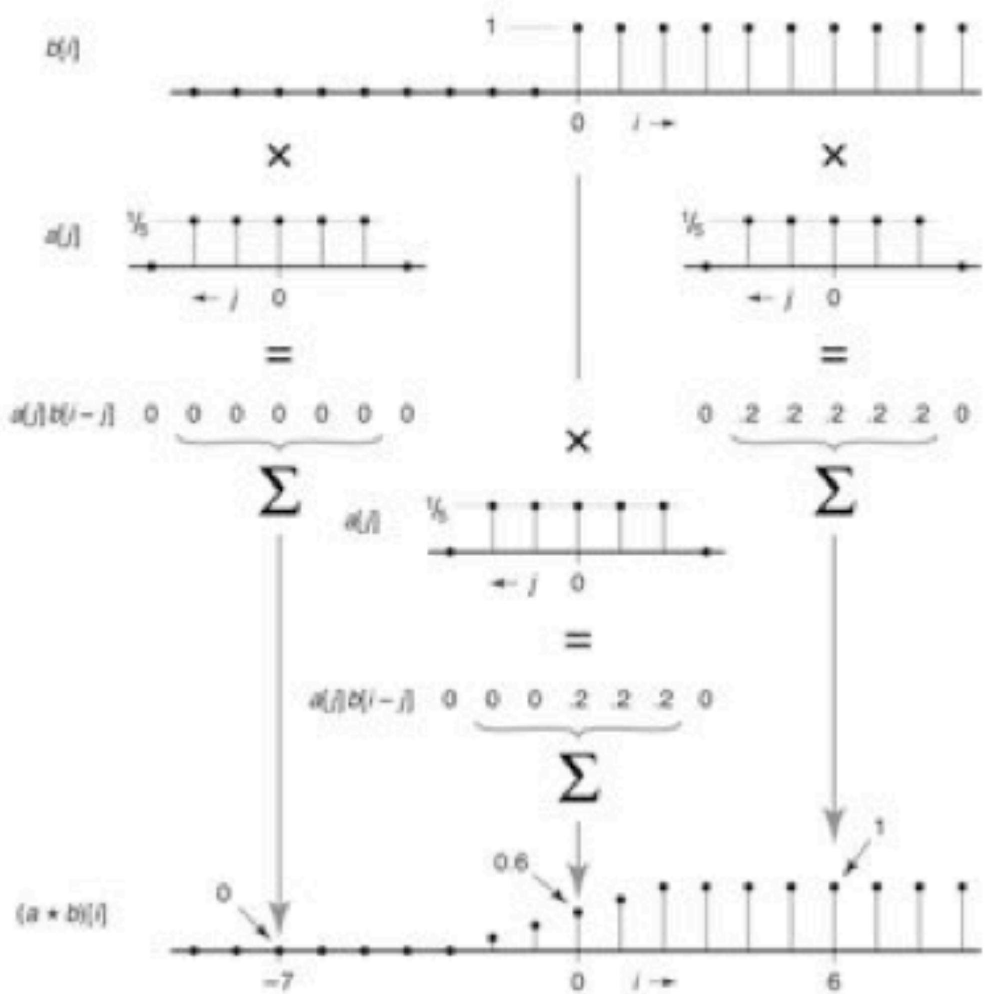
Example: box and step



Example: box and step

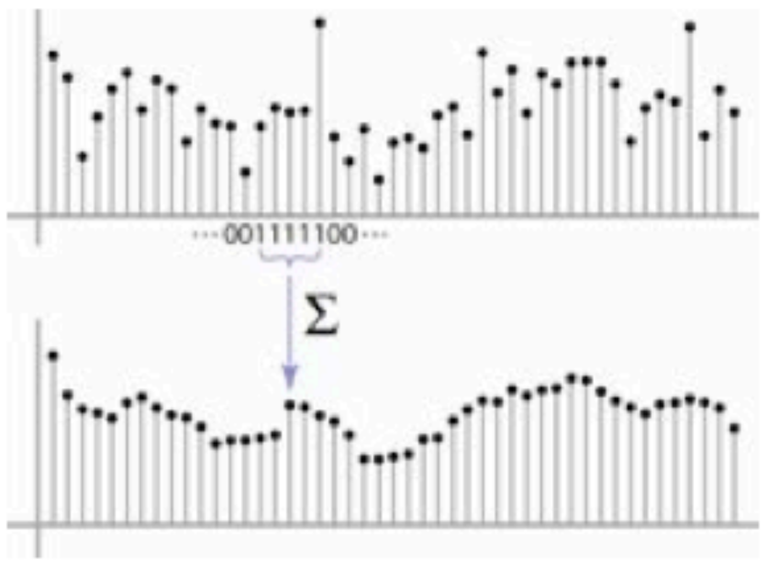


Example: box and step



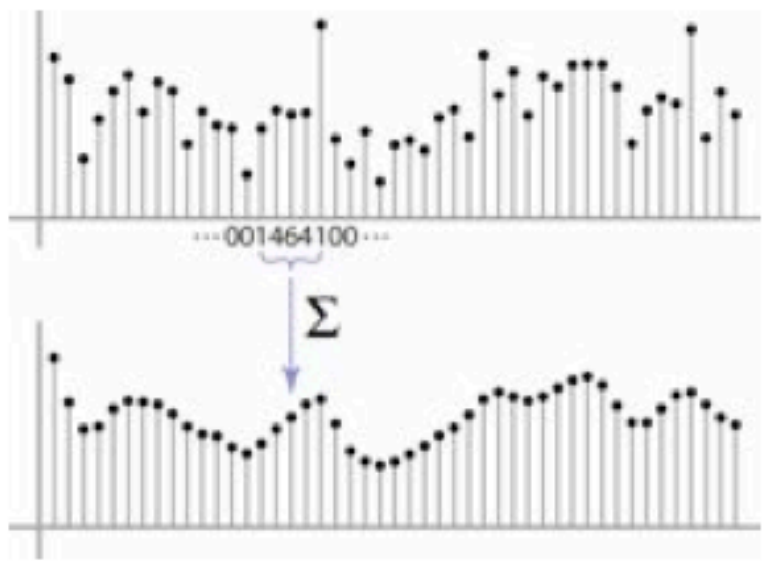
Convolution and filtering

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



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

Let’s take a break

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

a

0	0	0
0	0.5	0
0	0	0.5

ɐ

0.5	0	0
0	0.5	0
0	0	0

j →

i ↓

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

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

a

0	0	0
0	0.5	0
0	0	0.5

ɐ

0.5	0	0
0	0.5	0
0	0	0

j →

i ↓

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

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

a

0	0	0
0	0.5	0
0	0	0.5

ɐ

0.5	0	0
0	0.5	0
0	0	0

j →

i ↓

62	79	23	119	120	105	4	0
10	10	9	62	12	78	34	0
10	58	197	46	46	0	0	48
176	135	5	188	191	68	0	49
2	1	1	29	26	37	0	77
0	89	144	147	187	102	62	208
255	252	0	166	123	62	0	31
166	63	127	17	1	0	99	30

Linear filtering (warm-up slide)

original

$\star$

0	0	0
0	1	0
0	0	0

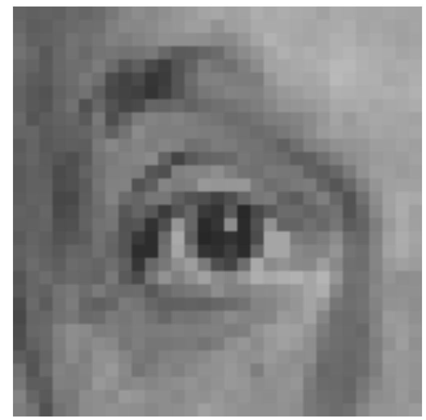
?

Linear filtering (warm-up slide)

original

★

0	0	0
0	1	0
0	0	0



Filtered
(no change)

Linear filtering

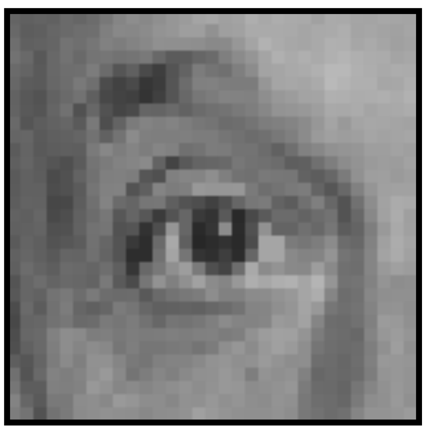
original

$\star$

0	0	0
1	0	0
0	0	0

?

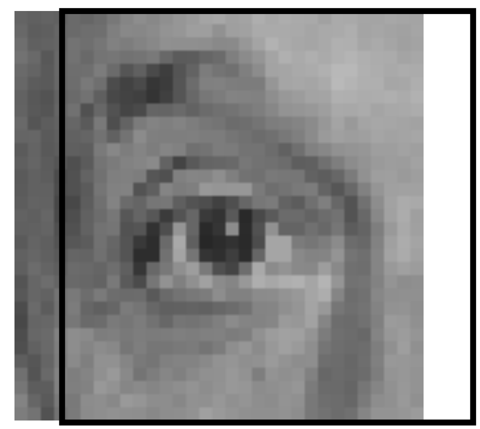
shift



original

★

0	0	0
1	0	0
0	0	0



shifted

Linear filtering

original

⋆

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

?

Blurring

original

★

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Blurred (filter applied in both dimensions).

Linear filtering (warm-up slide)

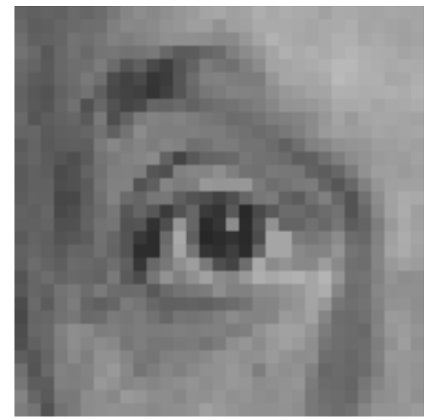
original

$$\star \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \right) \quad ?$$

Linear Filtering (no change)

original

$$\star \left(\begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \right)$$



Filtered
(no change)

Linear Filtering

original

$$\star \left(\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \begin{array}{|c|c|c|}\hline 1/9 & 1/9 & 1/9 \\ \hline 1/9 & 1/9 & 1/9 \\ \hline 1/9 & 1/9 & 1/9 \\ \hline \end{array} \right) \qquad ?$$

(remember blurring)

original

$\star$ (

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

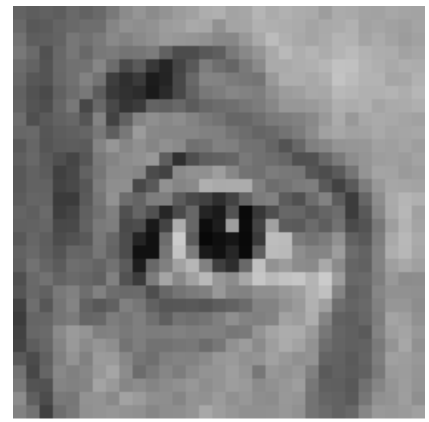
)

Blurred (filter applied in both dimensions).

sharpening

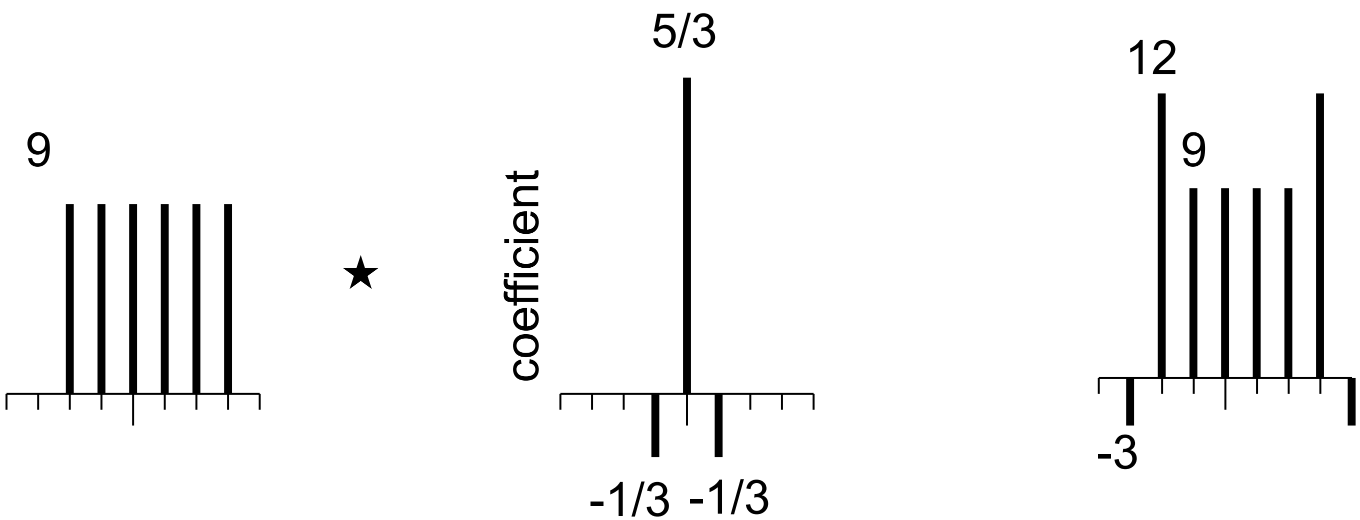
original

$$\star \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} \right)$$



Sharpened original

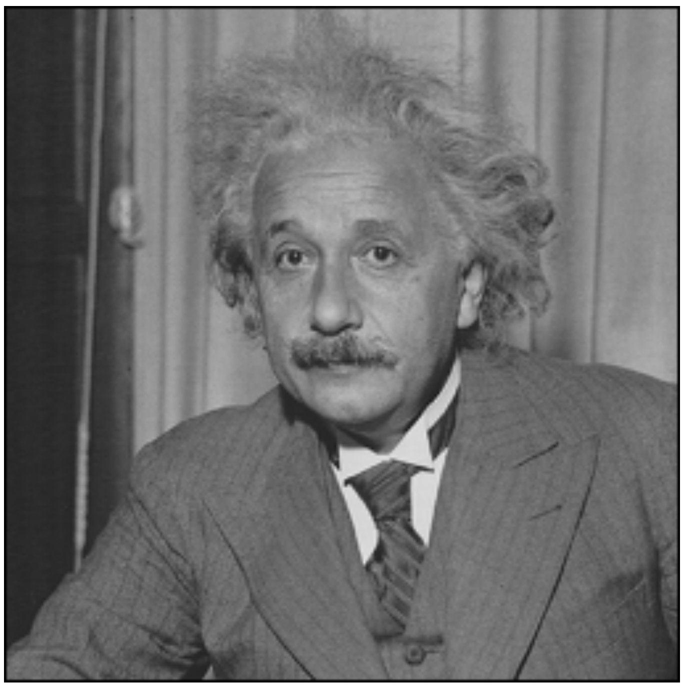
Sharpening example

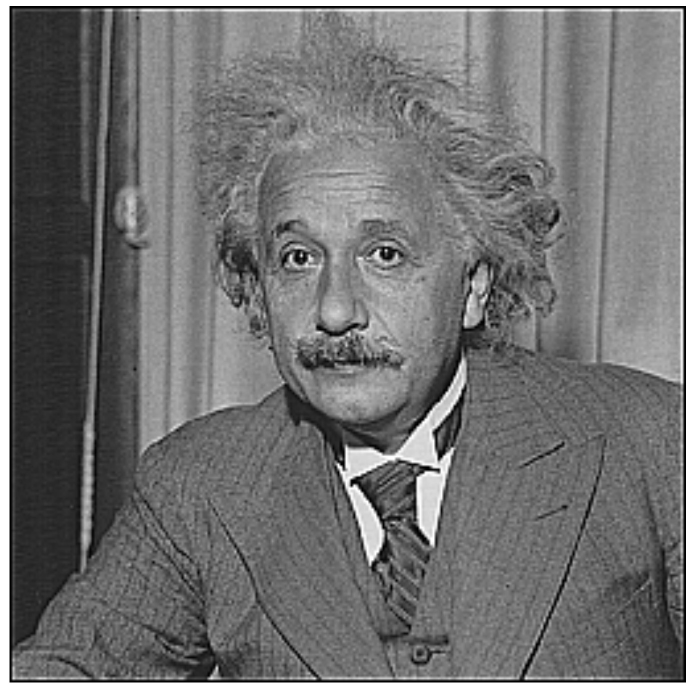


original

Sharpened (differences are accentuated; constant areas are left untouched).

Sharpening





before **after**

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, using gaussian, …)

  sharpening (impulse minus blur)

- 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:

$$\begin{aligned}(a_2 \star b)[i,j] &= \sum_{i'} \sum_{j'} a_2[i',j']b[i-i',j-j'] \\ &= \sum_{i'} \sum_{j'} a_1[i']a_1[j']b[i-i',j-j'] \\ &= \sum_{i'} a_1[i'] \left(\sum_{j'} a_1[j']b[i-i',j-j'] \right)\end{aligned}$$

Separable filtering

$$a_2[i,j] = a_1[i]a_1[j]$$

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

1	4	6	4	1

1
4
6
4
1

Separable filtering

$$a_2[i, j] = a_1[i]a_1[j]$$

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

0	0	0	0	0
0	0	0	0	0
1	4	6	4	1
0	0	0	0	0
0	0	0	0	0

0	0	1	0	0
0	0	4	0	0
0	0	6	0	0
0	0	4	0	0
0	0	1	0	0

Separable filtering

$$a_2[i, j] = a_1[i]a_1[j]$$

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

0	0	0	0	0
0	0	0	0	0
1	4	6	4	1
0	0	0	0	0
0	0	0	0	0

0	0	1	0	0
0	0	4	0	0
0	0	6	0	0
0	0	4	0	0
0	0	1	0	0

first, convolve with this

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

Separable filtering

$$a_2[i,j] = a_1[i]a_1[j]$$

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

0	0	0	0	0
0	0	0	0	0
1	4	6	4	1
0	0	0	0	0
0	0	0	0	0

0	0	1	0	0
0	0	4	0	0
0	0	6	0	0
0	0	4	0	0
0	0	1	0	0

second, convolve with this

first, convolve with this

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