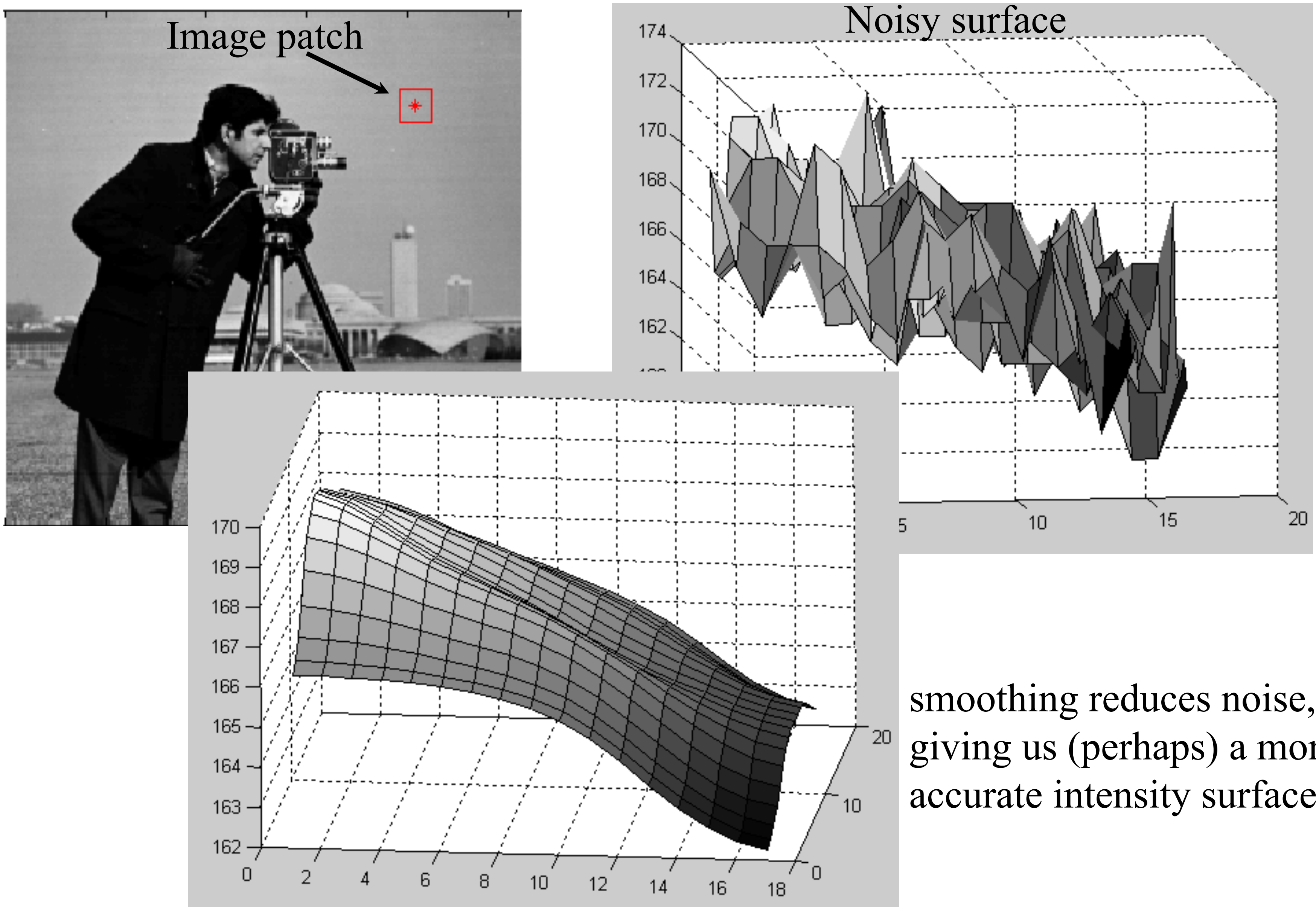
Lecture 4: Smoothing

Background reading alternatives:

- T&V Section 2.3.3 and Chapter 3
- Parts of Szeliski Section 3.2 are relevant
- Forsyth&Ponce, Chapter 8
- Jain et.al. Section 4.5

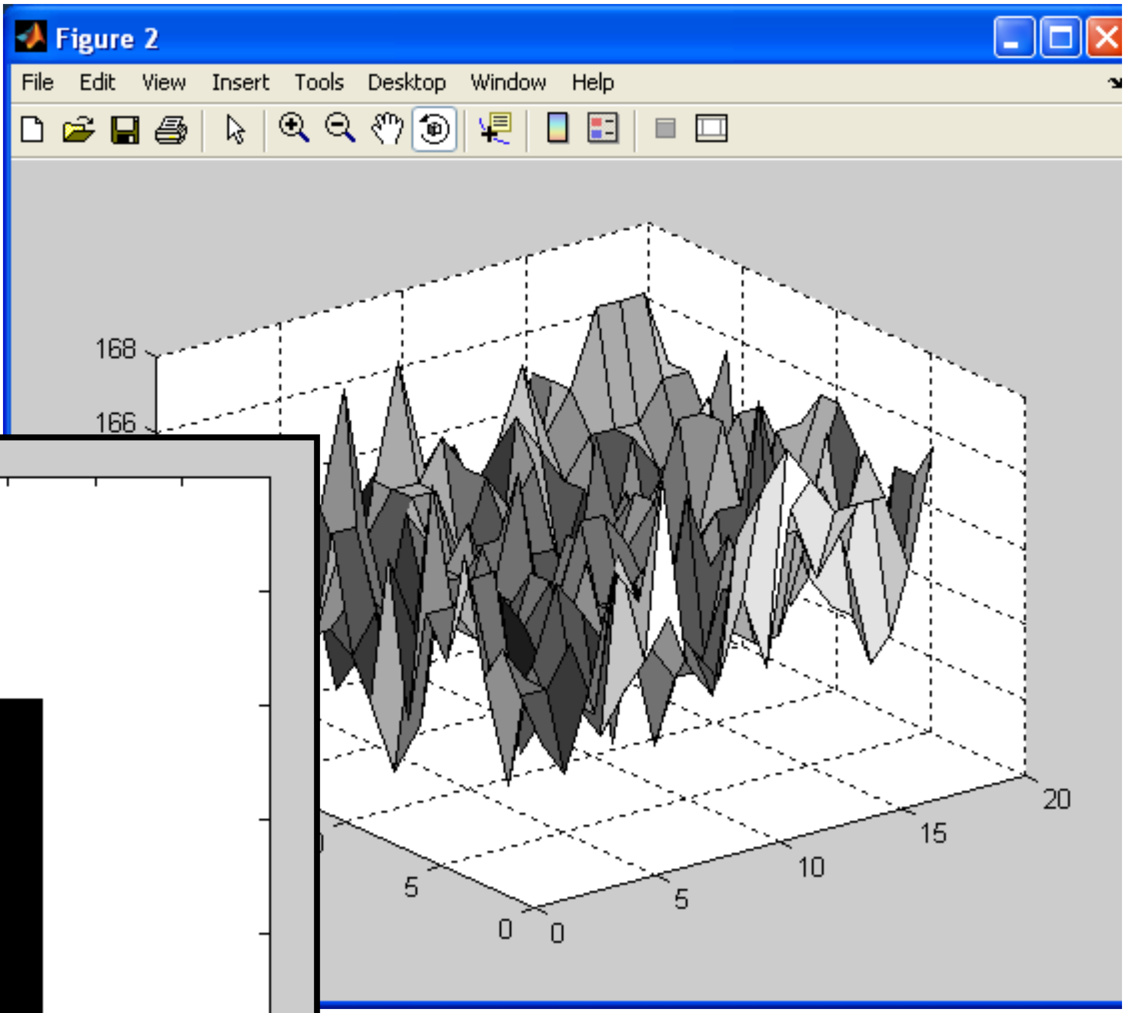
Today: Smoothing Reduces Noise

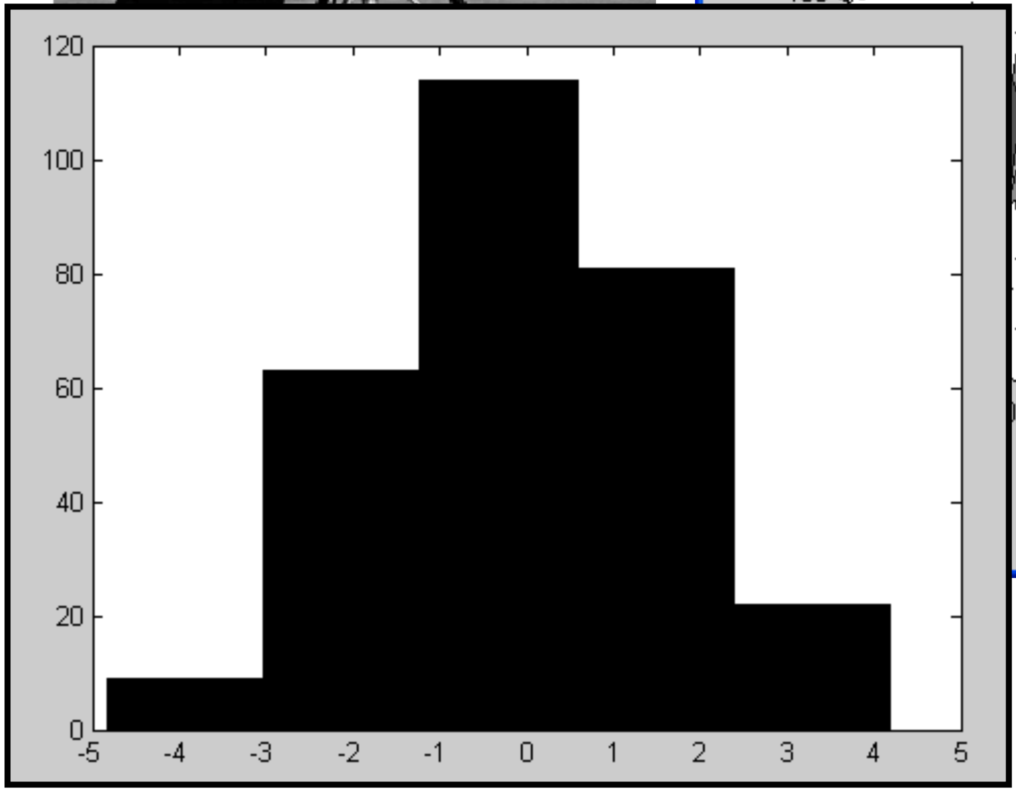


smoothing reduces noise, giving us (perhaps) a more accurate intensity surface.

Robert Collins
CMPEN454

Empirical Evidence for Sensor Noise being roughly Additive, Zero-mean, Gaussian

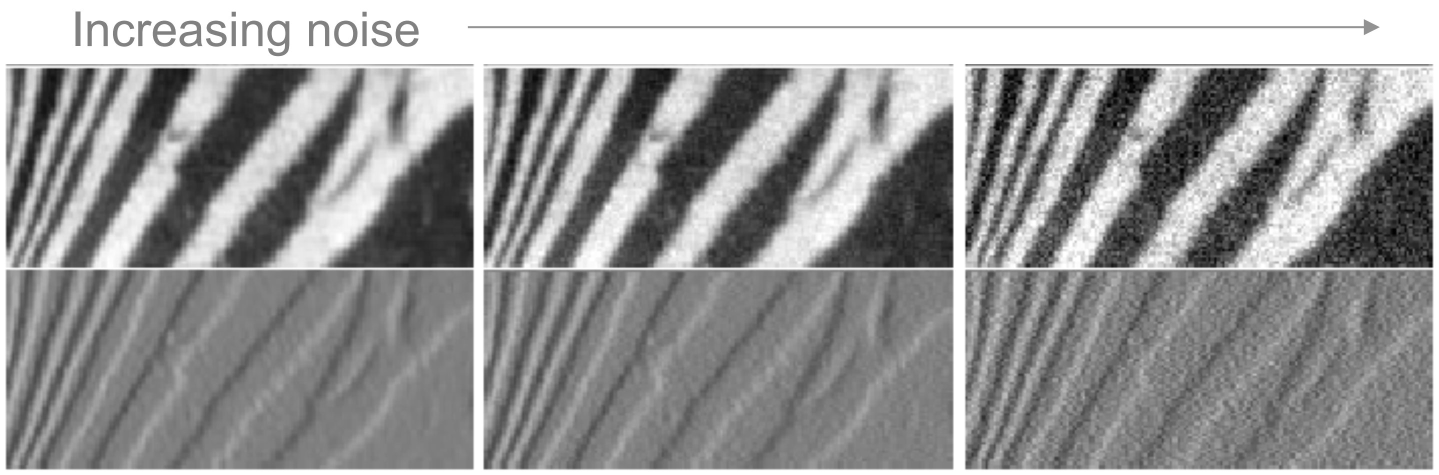




Mean = 164 Std = 1.8

Another motivation: Derivatives and Noise

- First derivative operator is affected by sensor noise

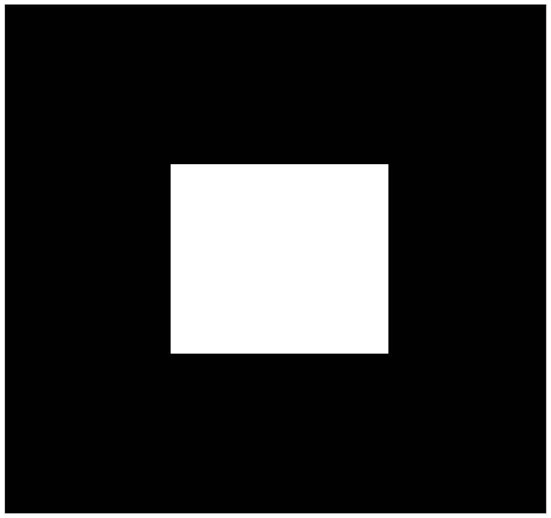


- Numerical derivatives can amplify noise!
  (particularly higher order derivatives)

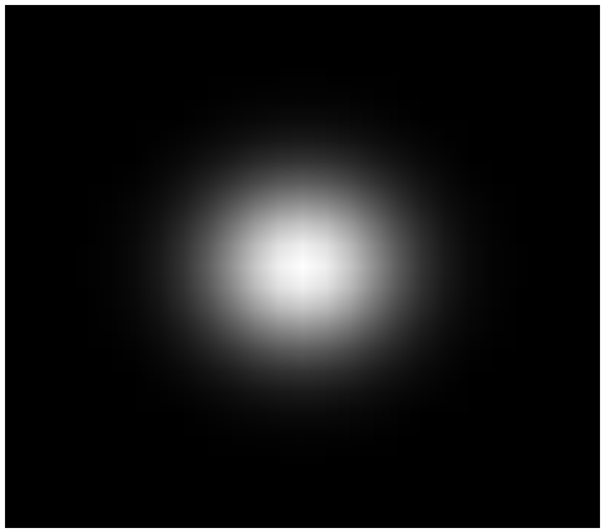
Preview

- We will talk about two smoothing filters
  - Box filter (simple averaging)
  - Gaussian filter (center pixels weighted more)

box filter



Gaussian filter



Averaging / Box Filter

- Filter with positive entries that sum to 1.
- Replaces each pixel with an average of its neighborhood.
- Called a BOX filter.

Box filter

$$1/9 \; \begin{array}{|c|c|c|} \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

important point:

since this is a linear operator, we can take the average around each pixel by convolving the image with this 3x3 filter!

Why Averaging Reduces Noise

- Intuitive explanation: variance of noise in the average is smaller than variance of the pixel noise (assuming zero-mean Gaussian noise).
- Sketch of more rigorous explanation

(see https://onlinecourses.science.psu.edu/stat414/node/166 for details)

$$A = \frac{1}{N}\sum_{i=1}^{N} I_m$$

$I_m = s_m + n_m$ with n being i.i.d. $G(0,\sigma^2)$

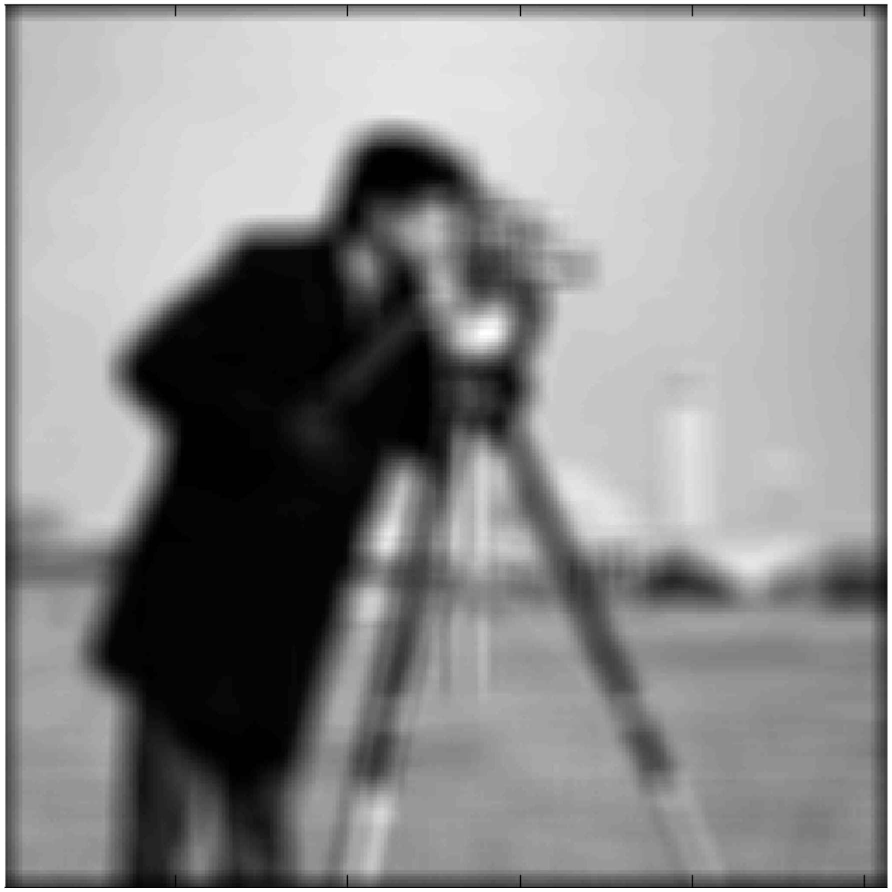
$$E(A) = \frac{1}{N}\sum s_m \longleftarrow \boxed{\text{Expected value is noise-free mean of the sample region}}$$

$$var(A) = E\left[(A - E(A))^2\right] = \frac{\sigma^2}{N} \longleftarrow \boxed{\text{With variance smaller than noise variance}}$$

Smoothing with Box Filter

original

Convolved with 11x11 box filter



Drawback: smoothing reduces fine image detail

Important Point about Smoothing

Averaging reduces noise (by reducing variance), leading to a more "accurate" estimate.

However, the more accurate estimate is of the mean of a local pixel neighborhood! This might not be what you want.

Balancing act: smooth enough to "clean up" the noise, but not so much as to remove important image gradients.

Gaussian Smoothing Filter

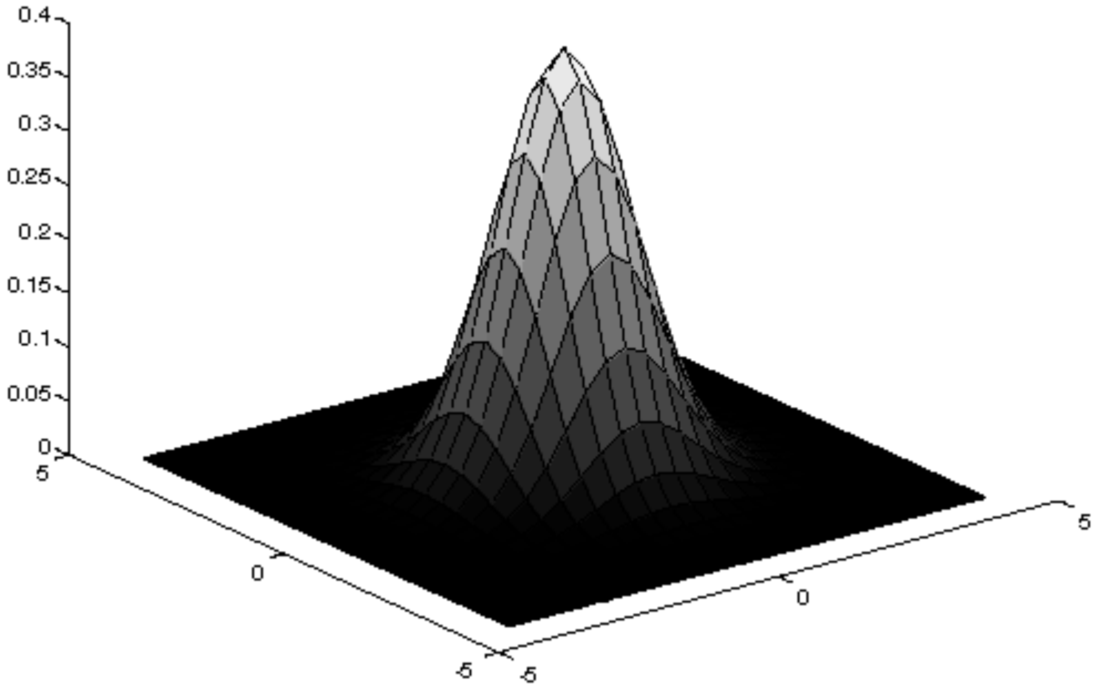
- Smoothing filter that does weighted averaging.
  - The coefficients are a 2D Gaussian.
  - Gives more weight at the central pixels and less weight to the neighbors.
  - The farther away the neighbors, the smaller the weight.

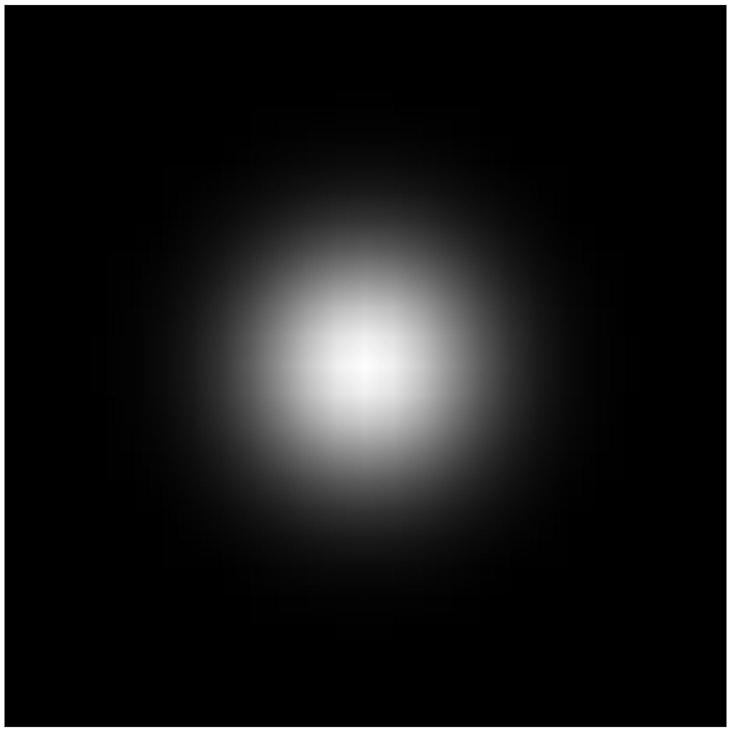
$$G_\sigma \equiv \frac{1}{2\pi\sigma^2}\exp\{-\frac{x^2+y^2}{2\sigma^2}\}$$

Confusion alert: there are now two Gaussians being discussed here (one for noise, one for smoothing). They are different.

Gaussian Smoothing Filter

An isotropic (circularly symmetric) Gaussian:



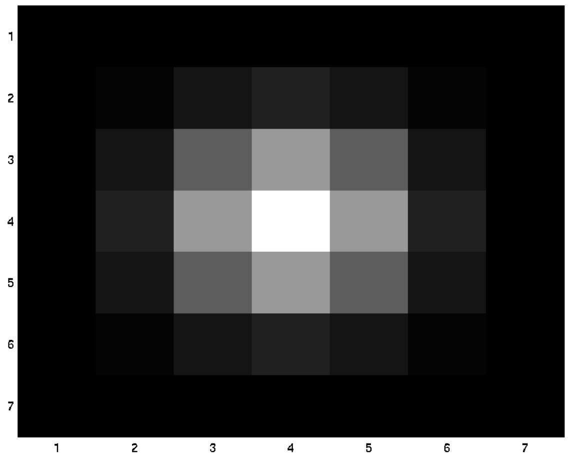


In Matlab

```
>> sigma = 1

sigma =

     1

>> halfwid = 3*sigma

halfwid =

     3

>> [xx,yy] = meshgrid(-halfwid:halfwid, -halfwid:halfwid);
>> tmp = exp(-1/(2*sigma^2) * (xx.^2 + yy.^2))

tmp =

    0.0001    0.0015    0.0067    0.0111    0.0067    0.0015    0.0001
    0.0015    0.0183    0.0821    0.1353    0.0821    0.0183    0.0015
    0.0067    0.0821    0.3679    0.6065    0.3679    0.0821    0.0067
    0.0111    0.1353    0.6065    1.0000    0.6065    0.1353    0.0111
    0.0067    0.0821    0.3679    0.6065    0.3679    0.0821    0.0067
    0.0015    0.0183    0.0821    0.1353    0.0821    0.0183    0.0015
    0.0001    0.0015    0.0067    0.0111    0.0067    0.0015    0.0001
```

Note: we have not included the normalization constant. Values of a Gaussian should sum to one. However, it is OK to ignore the constant as long as divide by it later.

Gaussian Smoothing Filter

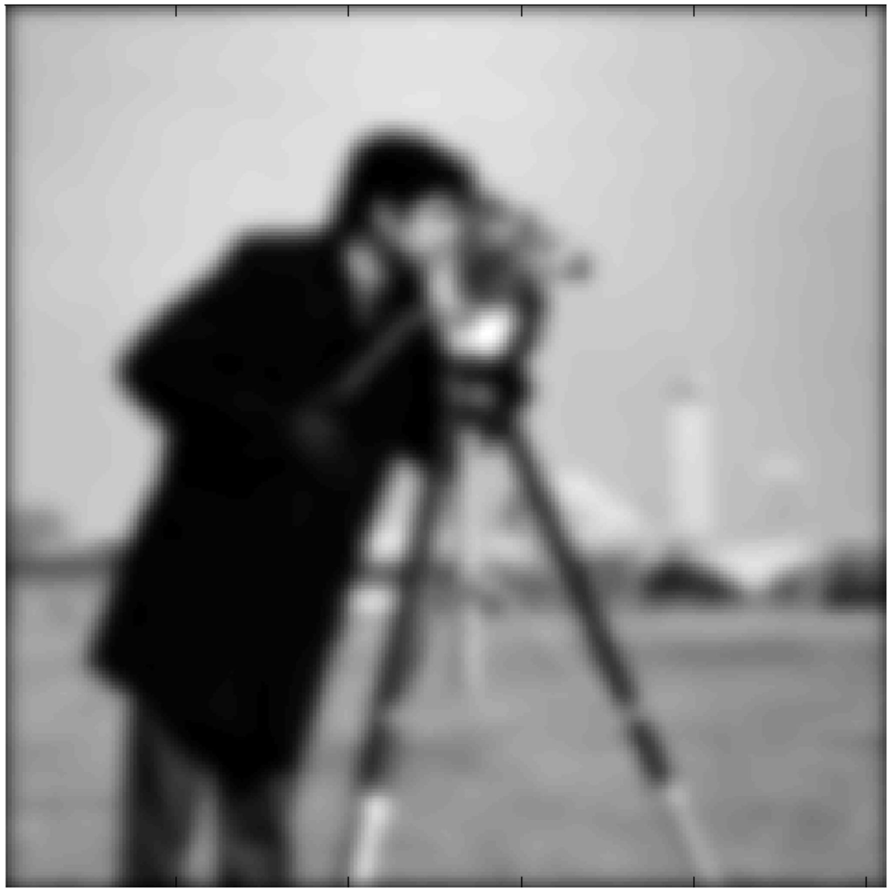


```
>> sigma = 1

sigma =

     1

>> halfwid = 3*sigma

halfwid =

     3
>> [xx,yy] = meshgrid(-halfwid:halfwid, -halfwid:halfwid);
>> gau = exp(-1/(2*sigma^2) * (xx.^2 + yy.^2))

gau =

    0.0001    0.0015    0.0067    0.0111    0.0067    0.0015    0.0001
    0.0015    0.0183    0.0821    0.1353    0.0821    0.0183    0.0015
    0.0067    0.0821    0.3679    0.6065    0.3679    0.0821    0.0067
    0.0111    0.1353    0.6065    1.0000    0.6065    0.1353    0.0111
    0.0067    0.0821    0.3679    0.6065    0.3679    0.0821    0.0067
    0.0015    0.0183    0.0821    0.1353    0.0821    0.0183    0.0015
    0.0001    0.0015    0.0067    0.0111    0.0067    0.0015    0.0001
```

Just another linear filter. Performs a weighted average.
Can be convolved with an image to produce a smoother image.

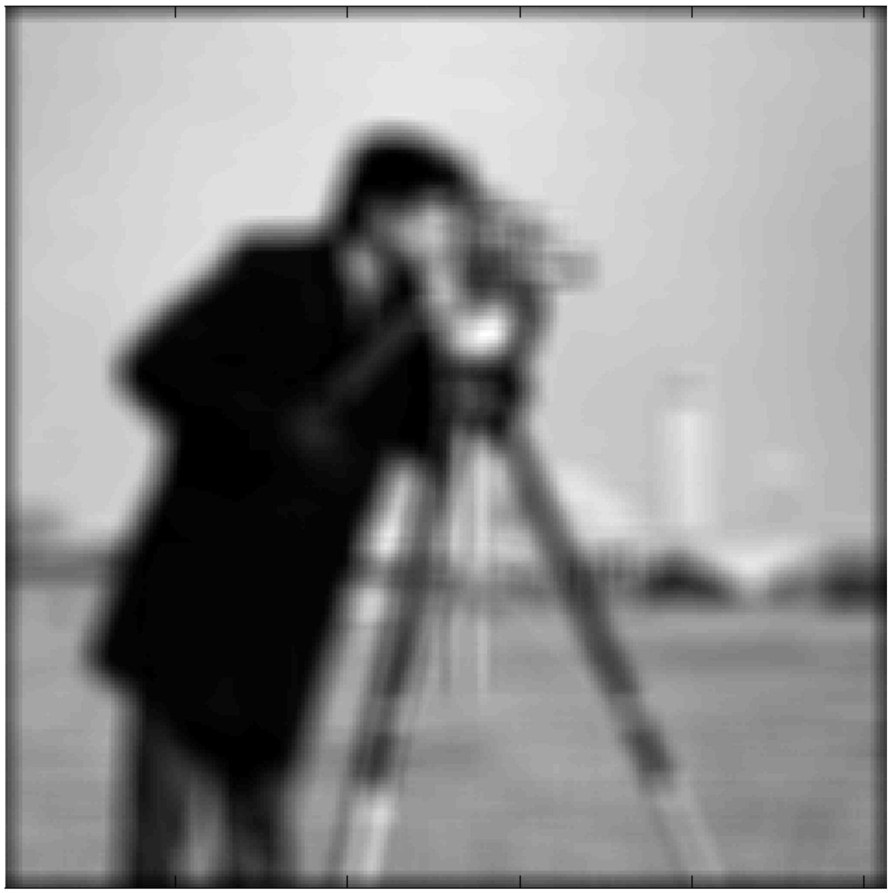
Gaussian Smoothing Example

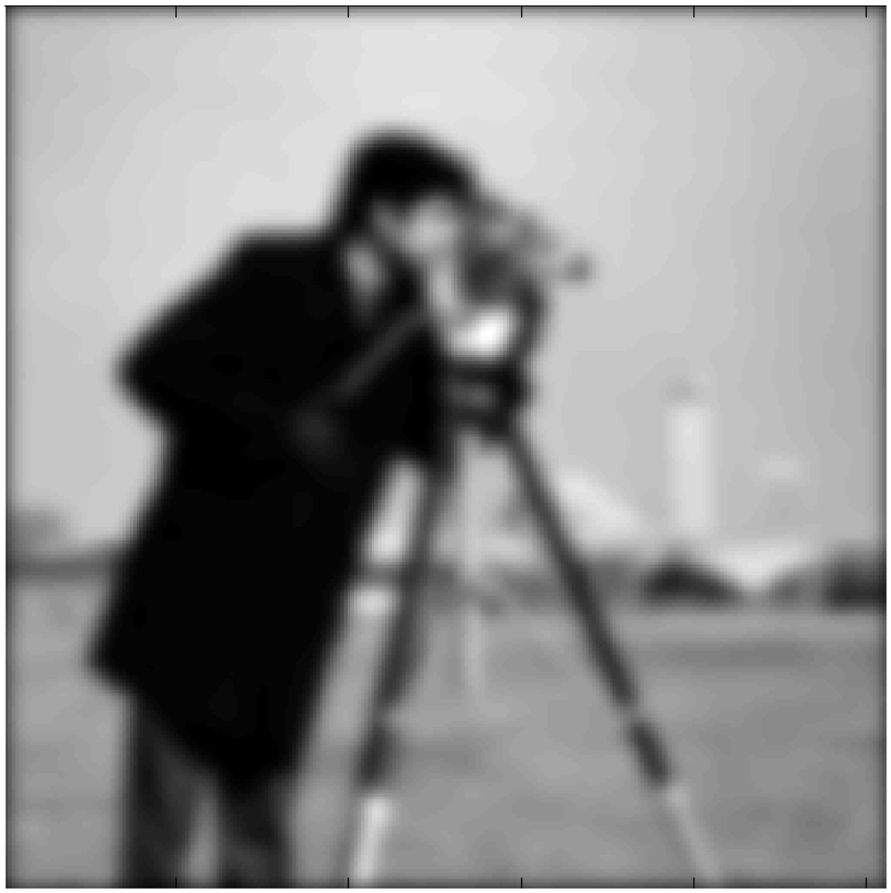
original


sigma = 3

Robert Collins
CMPEN454

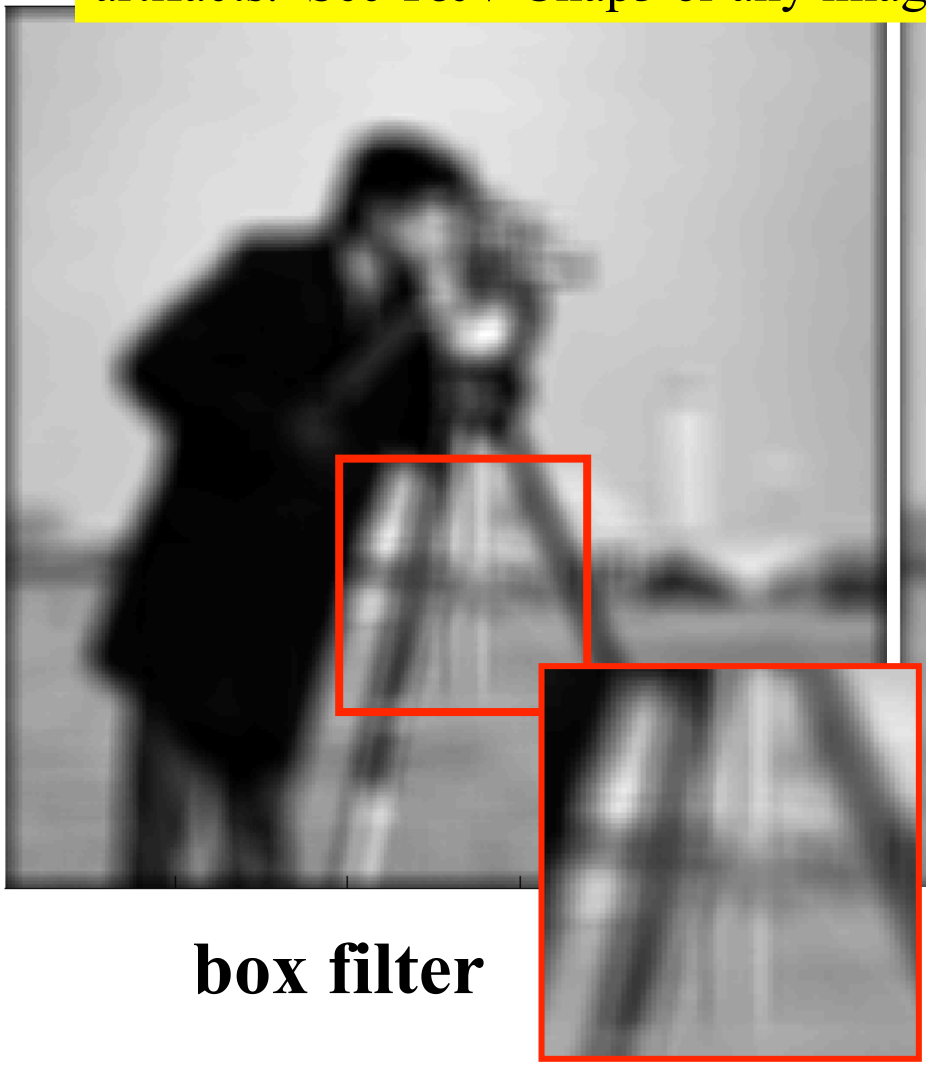
Box vs Gaussian


box filter

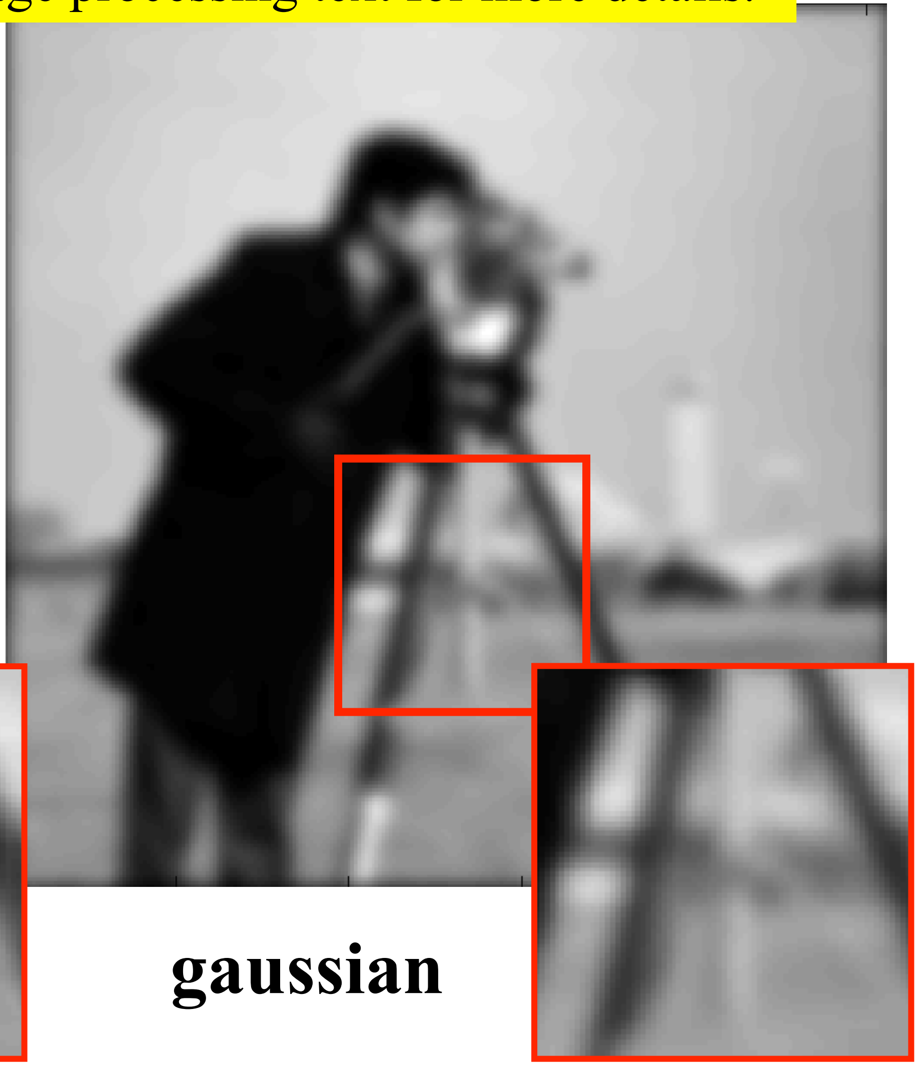

gaussian

Box vs Gaussian

Note: Gaussian is a true low-pass filter, so won't cause high frequency artifacts. See T&V Chap3 or any image processing text for more details.

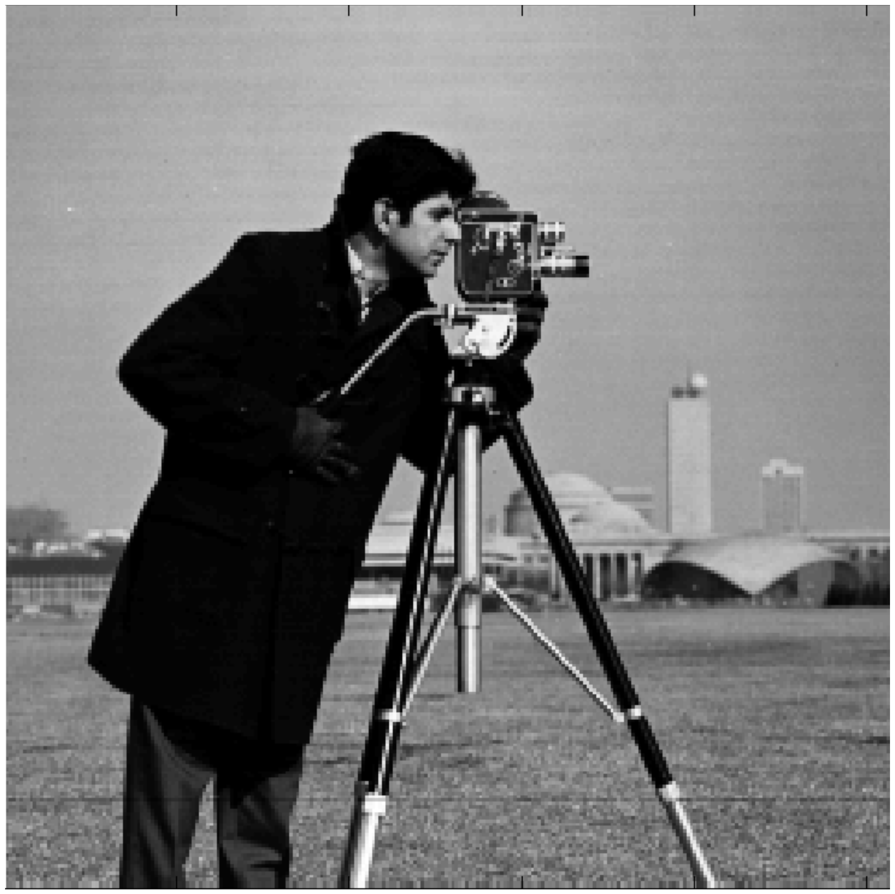


box filter

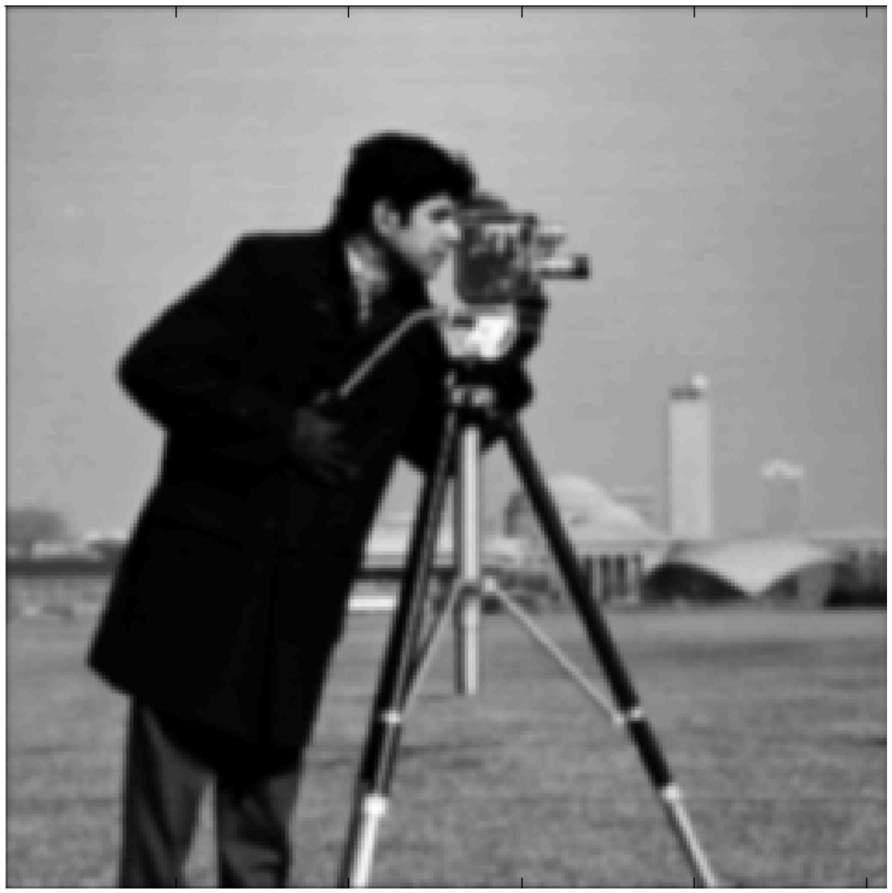


gaussian

Gaussian Smoothing at Different Scales



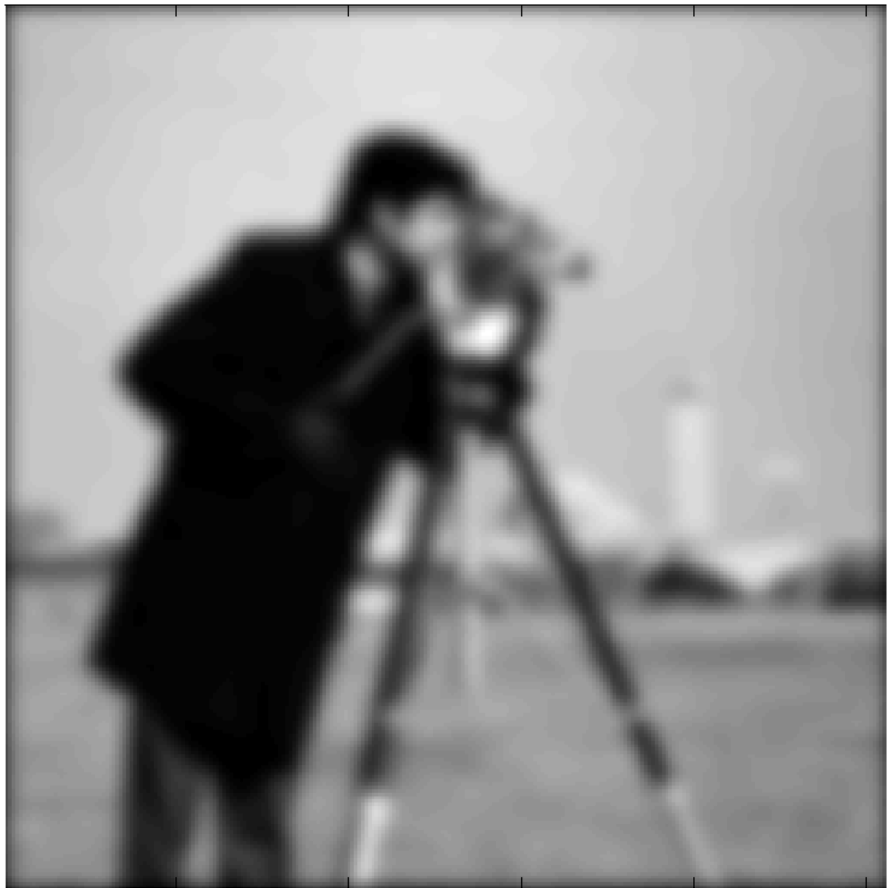
original



sigma = 1

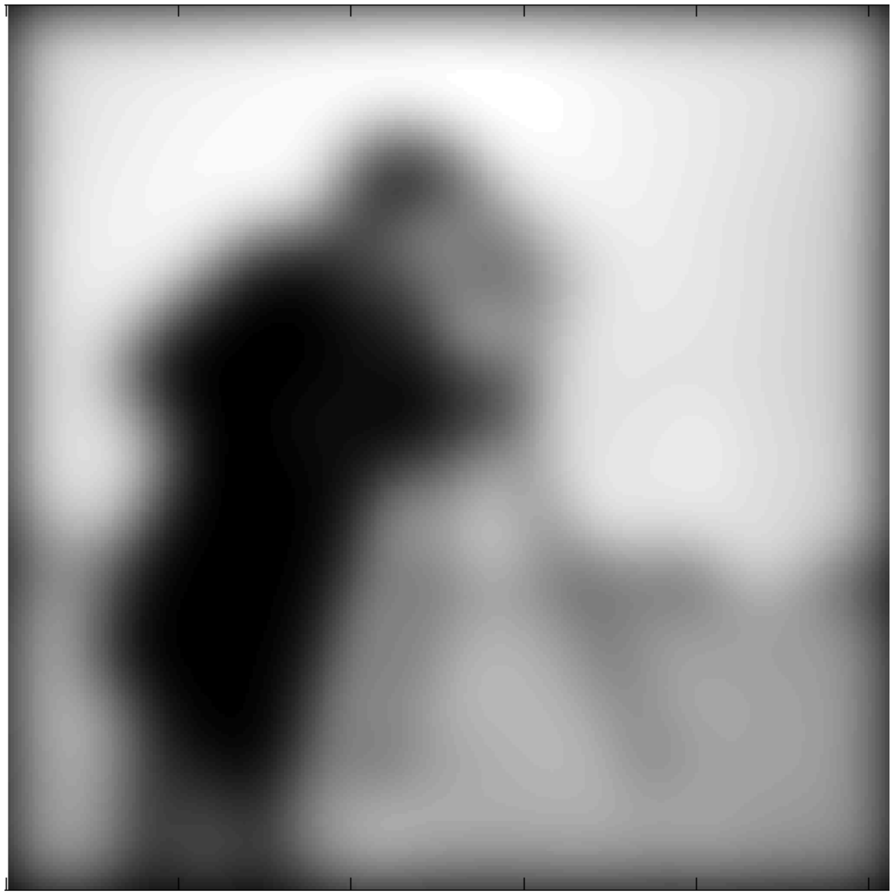
Gaussian Smoothing at Different Scales

original


sigma = 3

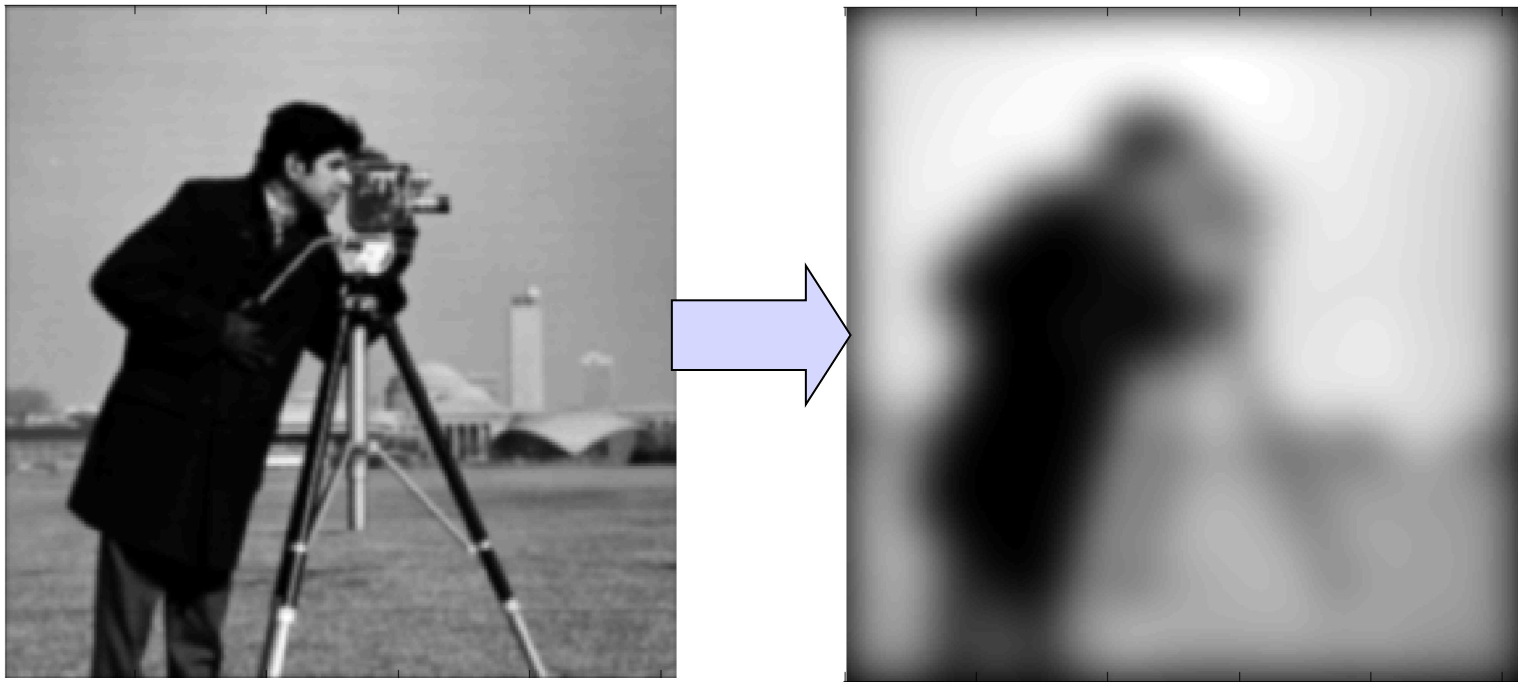
Gaussian Smoothing at Different Scales

original



sigma = 10

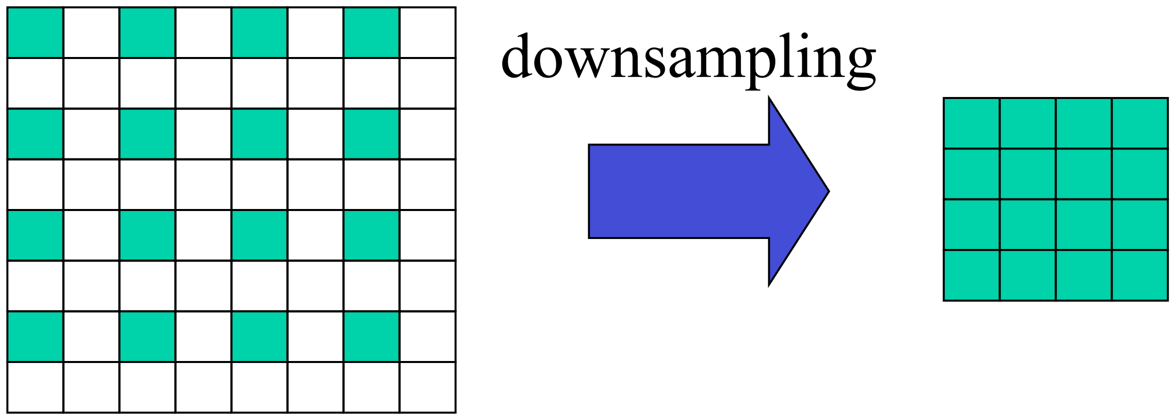
A Small Aside...



can you guess what border-handling method was used when convolving???

Practical Application: Thumbnails

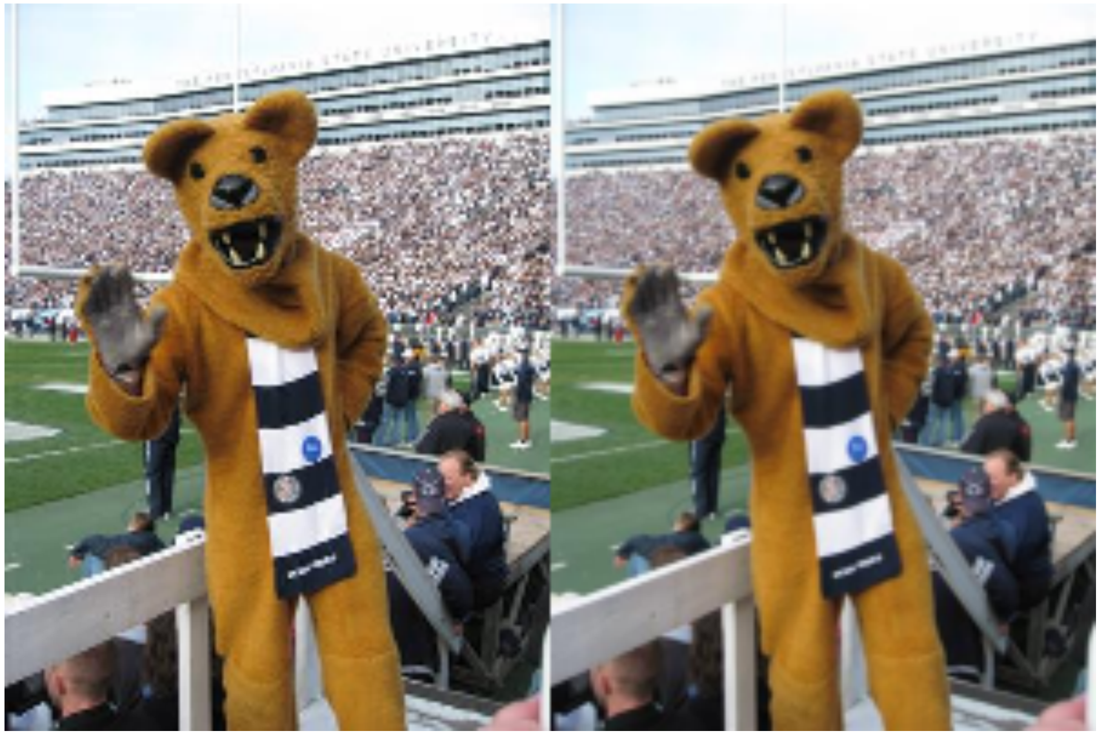
A common way to generate small thumbnail images is to repeatedly downsample an original full-size image.



However, naively applying downsampling/subsampling is a bad idea unless you have previously blurred/smoothed the image! Why? Because it causes high-frequency artifacts (aliasing) to appear.

Example : Thumbnails

original image
400x300

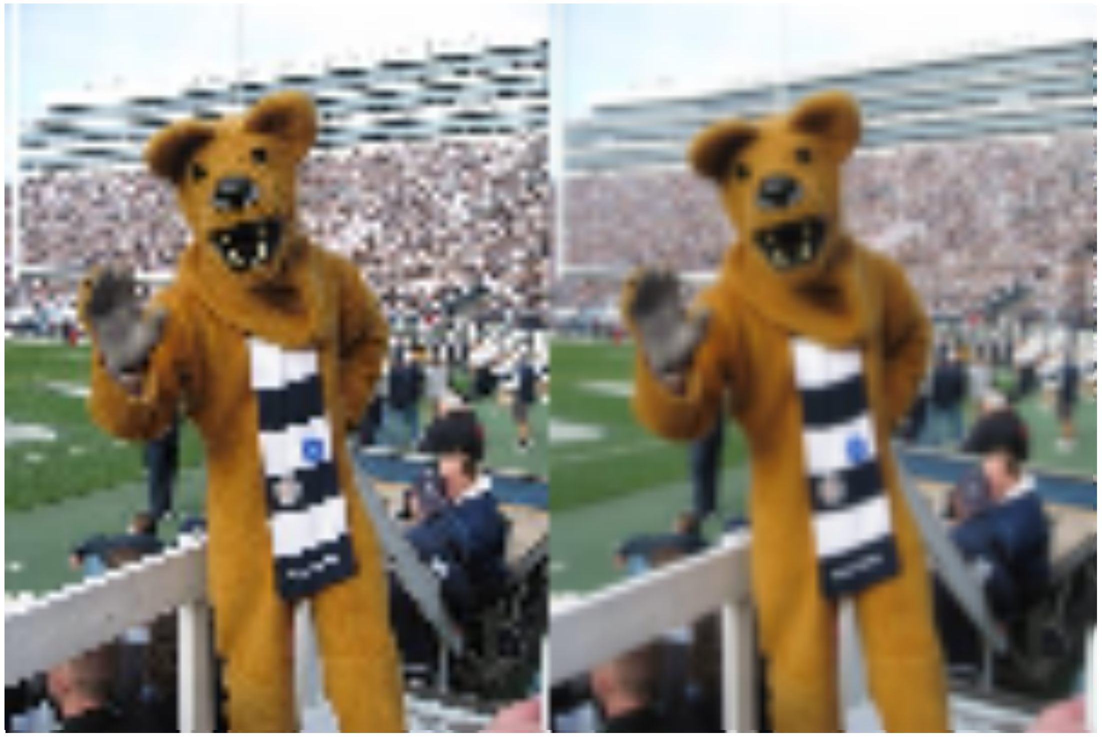


downsampling (left)
vs. smoothing then
downsampling (right)

200x150

Example : Thumbnails

original image
400x300

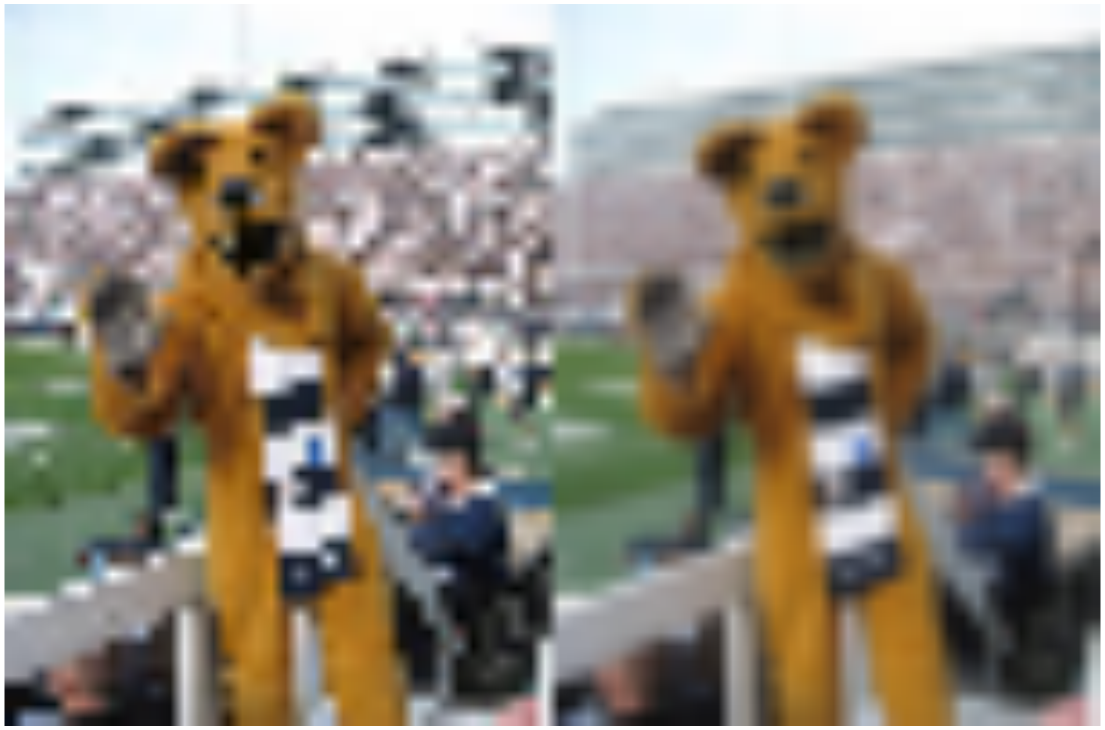


Downsampling twice (left) vs. applying smoothing then downsampling twice (right)

100x75

Example : Thumbnails

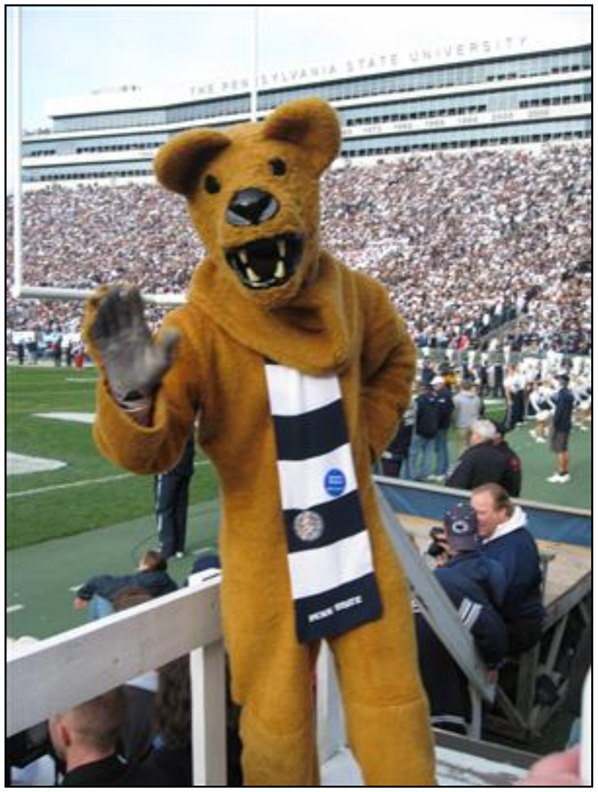
original image
400x300



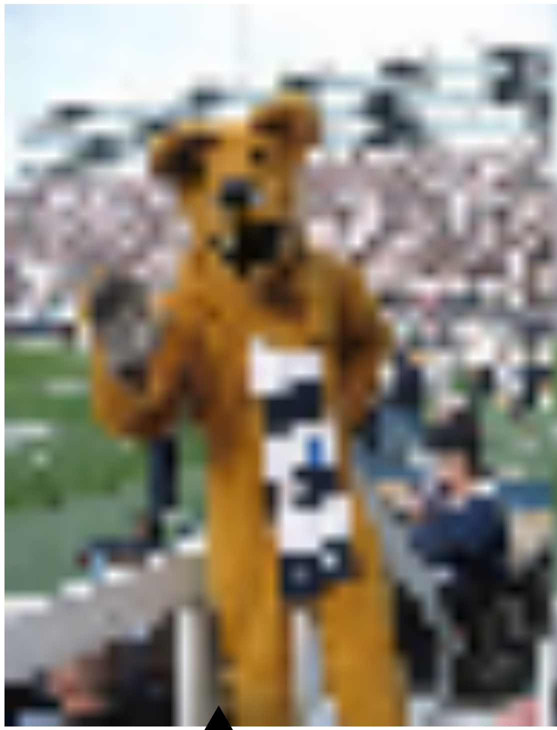
Downsampling 3 times (left) vs. applying smoothing then downsampling 3 times (right)

50x38

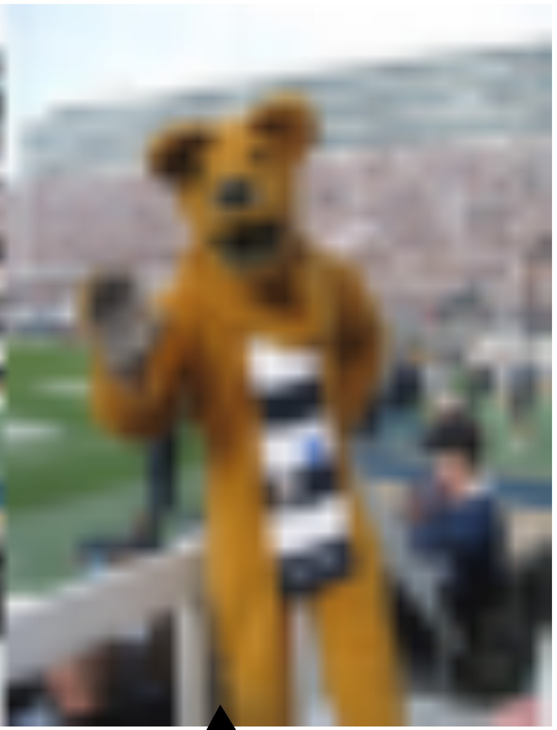
Example : Thumbnails



original image
400x300



Disfigured by high frequency artifacts

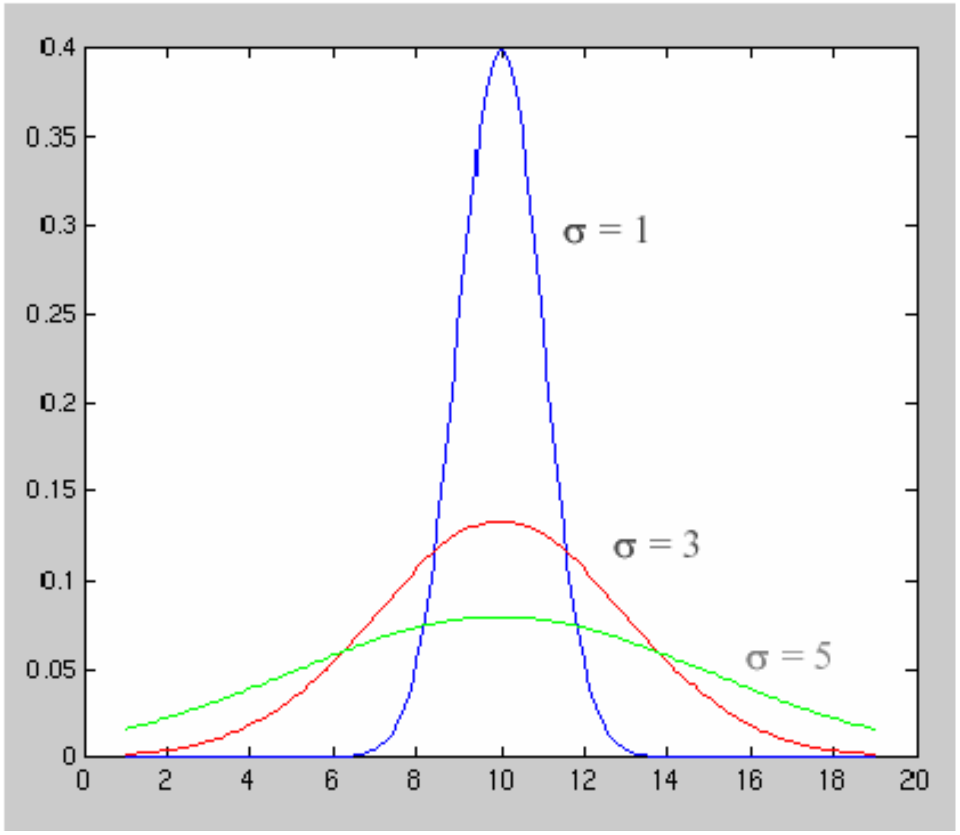


Loss of image resolution, but no high freq. artifacts

Implementation Details

How big should a digital Gaussian filter be?

- The std. dev σ of the Gaussian determines the amount of smoothing.
- Gaussian theoretically has infinite support, but we need a filter of finite size.
- 98.76% of the distribution falls between +/-2.5σ
- +/- 3σ covers over 99%



Efficient Implementation

- Both, the Box filter and the Gaussian filter are separable:
  - First convolve each row with a 1D filter
  - Then convolve each column with a 1D filter.

Separable Gaussian: **associativity**

$$G_\sigma * f = \left[g_{\sigma\rightarrow} * g_{\sigma\uparrow} \right] * f = g_{\sigma\rightarrow} * \left[g_{\sigma\uparrow} * f \right]$$

- Explanation sketch: write out convolution and use identity $e^A e^B = e^{A+B}$)

Efficient Implementation

- Cascaded Gaussians
  - – Repeated convolution by a small Gaussian simulates the effect of applying a larger one.
- G*(G*f) = (G*G)*f [associativity]

- Note: $G_{\sigma_1} * G_{\sigma_2} = G_{\sigma} \quad \sigma^2 = \sigma_1{}^2 + \sigma_2{}^2$

- explanation sketch: convolution in spatial domain is multiplication in frequency domain (Fourier space). Fourier transform of Gaussian is

$$\mathcal{F}_x\left[e^{-\frac{x^2}{2\sigma^2}}\right] = e^{-2\pi^2\sigma^2u^2}$$

$$e^{-2\pi^2\sigma_1^2u^2} e^{-2\pi^2\sigma_2^2u^2} = e^{-2\pi^2(\sigma_1^2+\sigma_2^2)u^2}$$