Image processing

Image operations

- Operations on an image
  - Linear filtering
  - Non-linear filtering
  - Transformations
  - Noise removal
  - Segmentation

Linear Filters

- General process:
  - Form new image whose pixels are a weighted sum of original pixel values, using the same set of weights at each point.
- Properties
  - Output is a linear function of the input
  - Output is a shift-invariant function of the input (i.e. shift the input image two pixels to the left, the output is shifted two pixels to the left)
- Example: smoothing by averaging
  - form the average of pixels in a neighbourhood
- Example: smoothing with a Gaussian
  - form a weighted average of pixels in a neighbourhood
- Example: finding a derivative
  - form a weighted average of pixels in a neighbourhood

Convolution

- Represent these weights as an image, H
- H is usually called the **kernel**
- Operation is called **convolution**
- Properties:

- Convolution is *commutative*.

$$c = a \otimes b = b \otimes a$$

- Convolution is *associative*.

$$c = a \otimes (b \otimes c) = (a \otimes b) \otimes c = a \otimes b \otimes c$$

- Convolution is *distributive*.

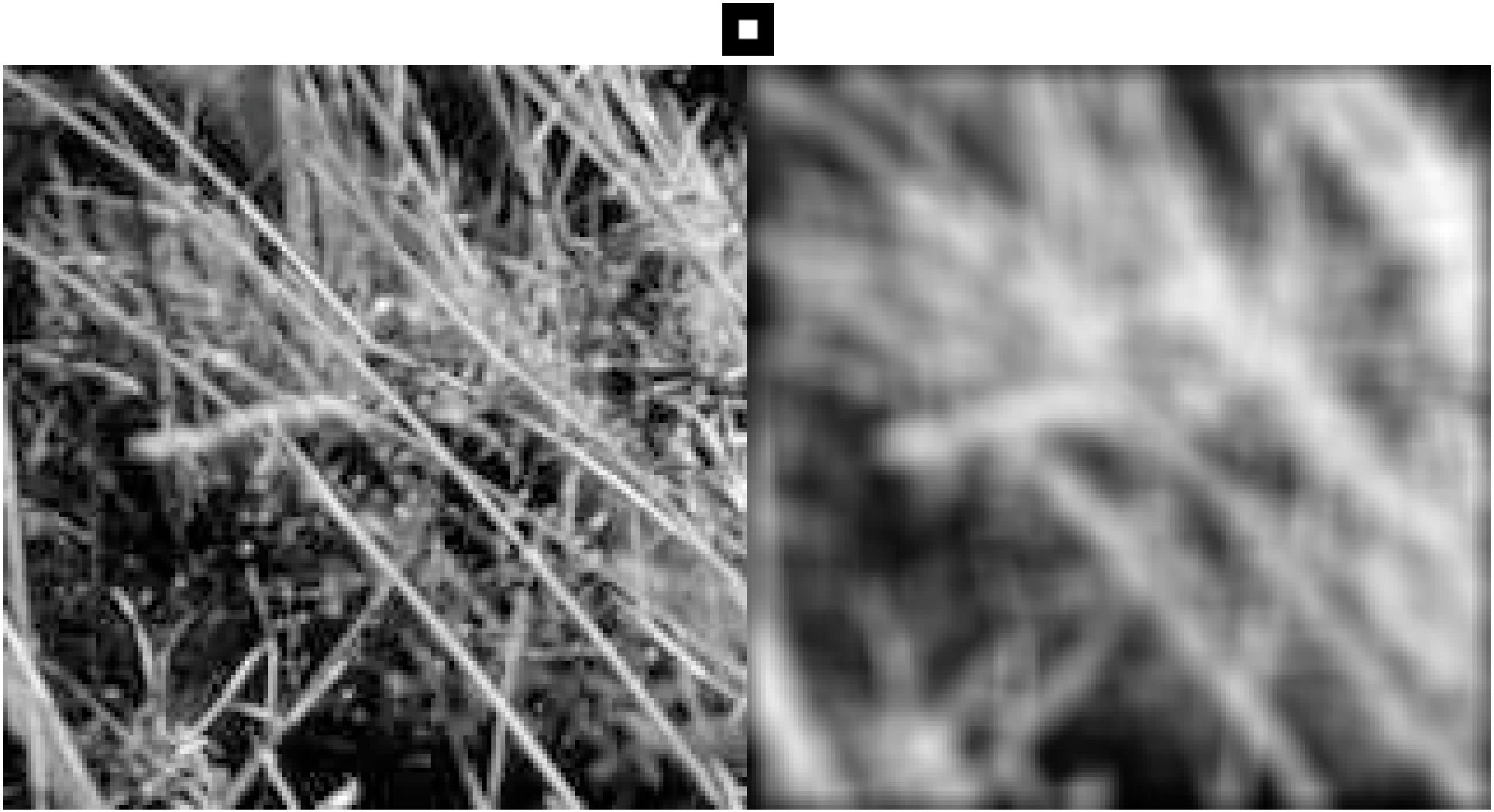
$$c = a \otimes (b + d) = (a \otimes b) + (a \otimes d)$$

- Result is:

$$R_{ij} = \sum_{u,v} H_{i-u,j-v} F_{uv}$$

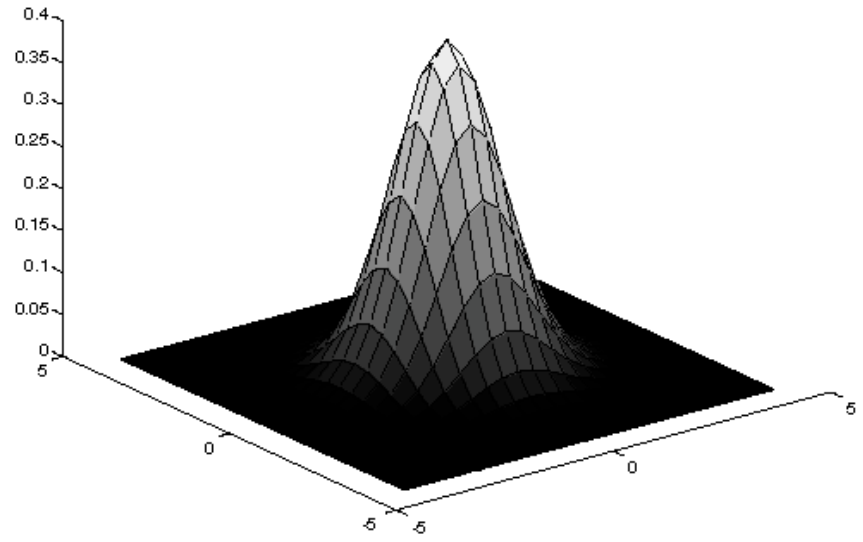
- Notice the order of indices
  - all examples can be put in this form
  - it's a result of the derivation expressing any shift-invariant linear operator as a convolution.

Example: Smoothing by Averaging



Smoothing with a Gaussian

- Smoothing with an average actually doesn't compare at all well with a defocussed lens
  - Most obvious difference is that a single point of light viewed in a defocussed lens looks like a fuzzy blob; but the averaging process would give a little square.

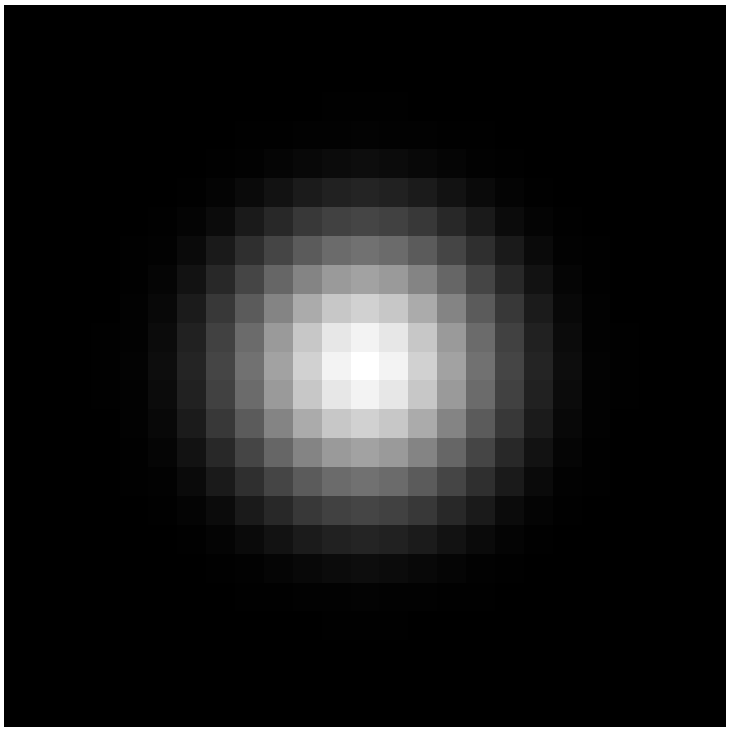


- A Gaussian gives a good model of a fuzzy blob

For a filter size N by M,

$$R_{ij} = \sum_{u=1:N} \sum_{v=1:M} H_{uv} F_{i+u-1, j+v-1}$$

An Isotropic Gaussian

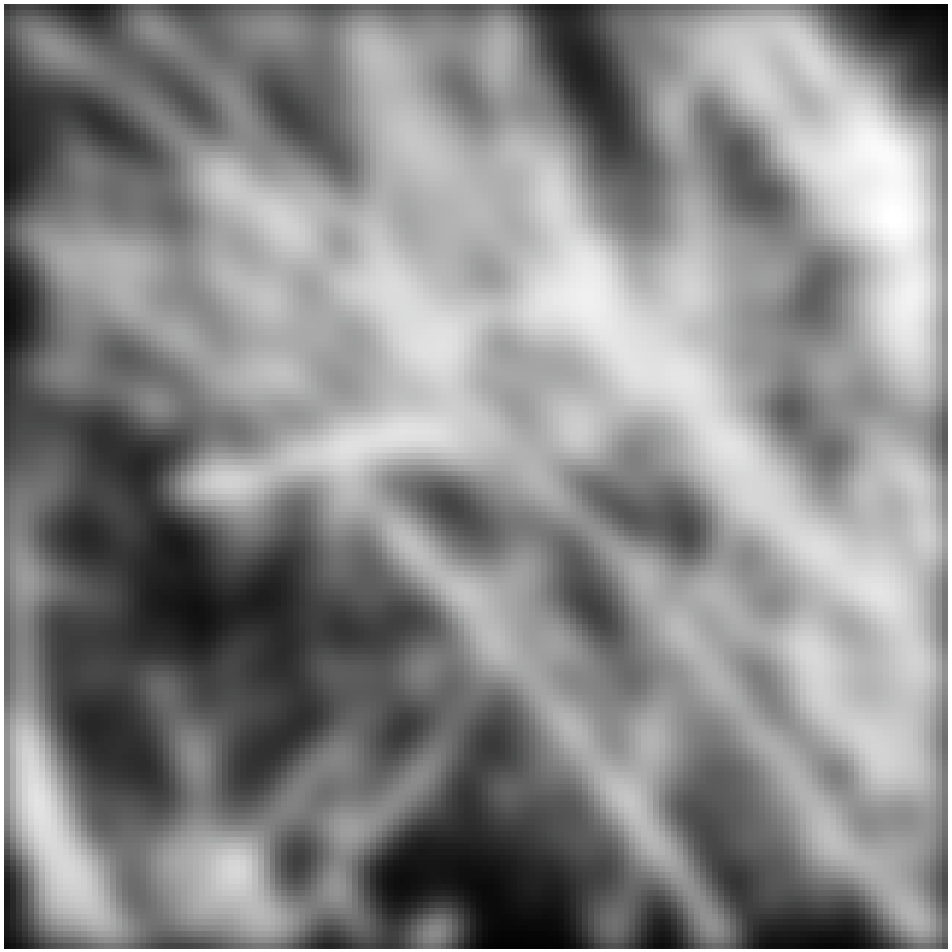


- The picture shows a smoothing kernel proportional to

$$H \sim \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)$$

(which is a reasonable model of a circularly symmetric fuzzy blob)

Smoothing with a Gaussian



Differentiation and convolution

- Recall

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

- We could approximate this as

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

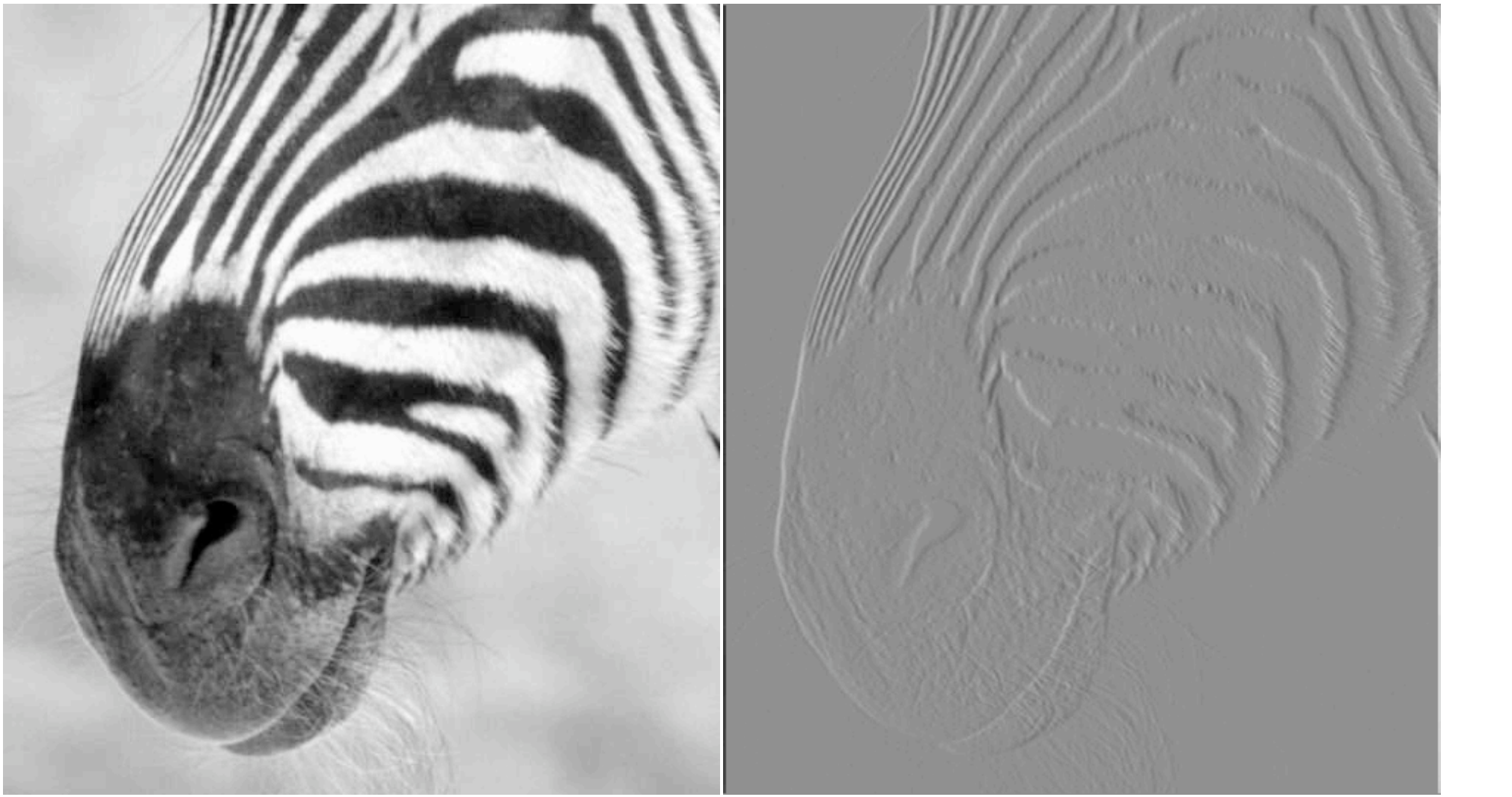
- Now this is linear and shift invariant, so must be the result of a convolution.

This is a convolutio: (but it's not a very good way to do things, as we shall see)

$$\frac{\partial h}{\partial x} \propto h_{i+1,j} - h_{i-1,j} \qquad \frac{\partial h}{\partial x} = \sum_u \sum_v D_{i-u, j-v} F_{u,v}$$

$$D = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

Finite differences



Noise

- Simplest noise model
  - independent stationary additive Gaussian noise
  - the noise value at each pixel is given by an independent draw from the same normal probability distribution

For an image *F*,
the measured value G:

$$G_{u,v} = F_{u,v} + n_{u,v}$$

$$n \overset{D}{\sim} N(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

- Issues
  - this model allows noise values that could be greater than maximum camera output or less than zero
  - for small standard deviations, this isn't too much of a problem - it's a fairly good model
  - independence may not be justified (e.g. damage to lens)
  - may not be stationary (e.g. thermal gradients in the ccd)

sigma=1

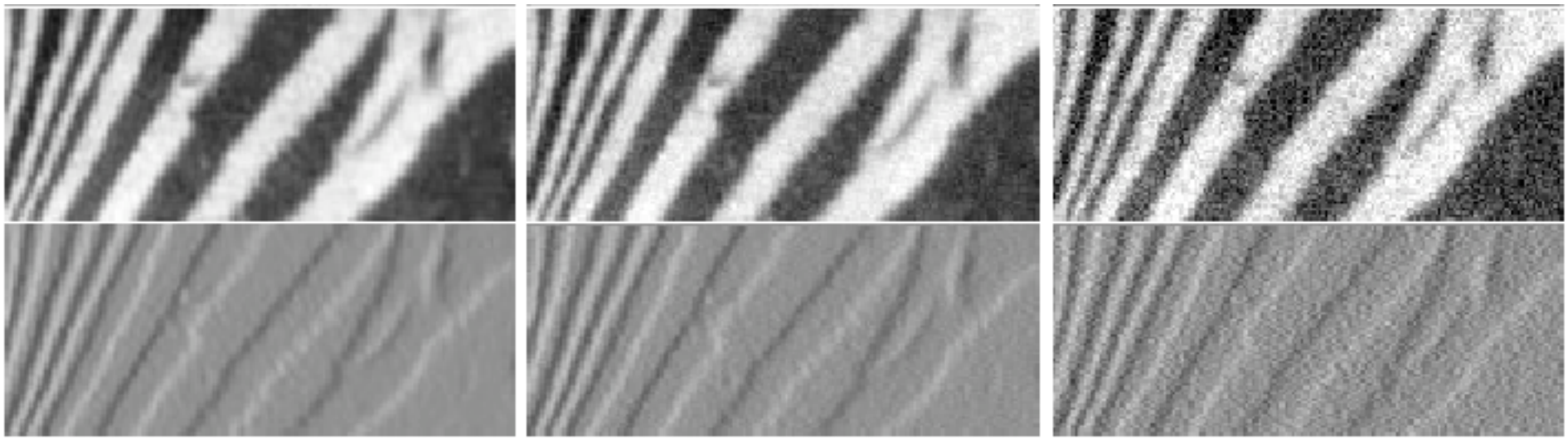
sigma=16

Finite differences and noise

- Finite difference filters respond strongly to noise
  - obvious reason: image noise results in pixels that look very different from their neighbours
- Generally, the larger the noise the stronger the response

- What is to be done?
  - intuitively, most pixels in images look quite a lot like their neighbours
  - this is true even at an edge; along the edge they're similar, across the edge they're not
  - suggests that smoothing the image should help, by forcing pixels different to their neighbours (=noise pixels?) to look more like neighbours

Finite differences responding to noise



Increasing noise ->
(this is zero mean additive gaussian noise)

The response of a linear filter to noise

- Do only stationary independent additive Gaussian noise with zero mean (non-zero mean is easily dealt with)
- **Generalized Average (Mean)**:
  - output is a weighted sum of inputs
  - so we want mean of a weighted sum of zero mean normal random variables
  - must be zero

$$G_{u,v} = F_{u,v} + n_{u,v}$$

$$n \overset{D}{\sim} N(\mu, \sigma^2)$$

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

$$\mu = \sum_{i=1}^{N} w_i x_i \qquad \sum_{i=1}^{N} w_i = 1$$

$$R_{i,j} = \sum_{u,v} w_{i-u,j-v} G_{u,v} = \sum_{u,v} w_{i-u,j-v} \left(F_{u,v} + n_{u,v} \right)$$

$$R_{i,j} = \sum_{u,v} w_{i-u,j-v} F_{u,v} + \sum_{u,v} w_{i-u,j-v} n_{u,v}$$

$$R_{i,j} = \hat{F}_{u,v} + \mu_{u,v}$$

Response linear filter to noise

Properties of sums of Gaussian random variables

$$\frac{1}{N}\sum_{i=1:N} n_i = \mu$$

$$Var\left[\frac{1}{N}\sum_{i=1:N} n_i\right] = \frac{1}{N}\sum_{i=1:N} E\left[n_i^2\right] = \frac{1}{N}\sum_{i=1:N} \sigma_i^2$$

- **Variance**:
  - recall
    - variance of a sum of random variables is sum of their variances
    - variance of constant times random variable is constant^2 times variance
  - then if σ is noise variance and kernel is w, variance of response is

$$Var\left[\sum_{u,v} w_{i-u,j-v} n_{u,v}\right]$$

$$= \sum_{u,v} E\left[w_{i-u,j-v}^2 n_{u,v}^2\right]$$

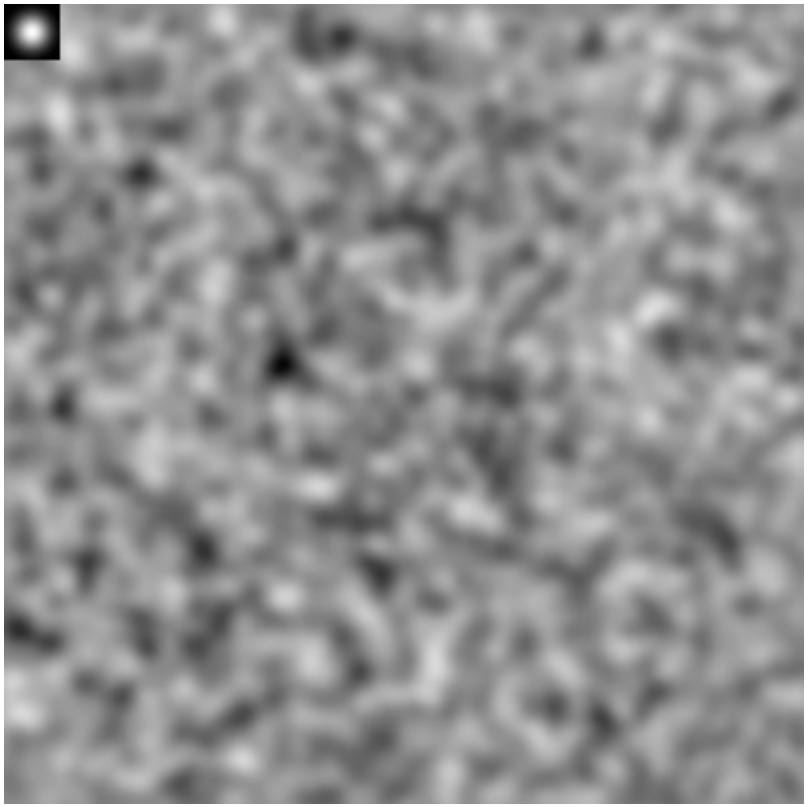
$$= \sum_{u,v} w_{i-u,j-v}^2 E\left[n_{u,v}^2\right]$$

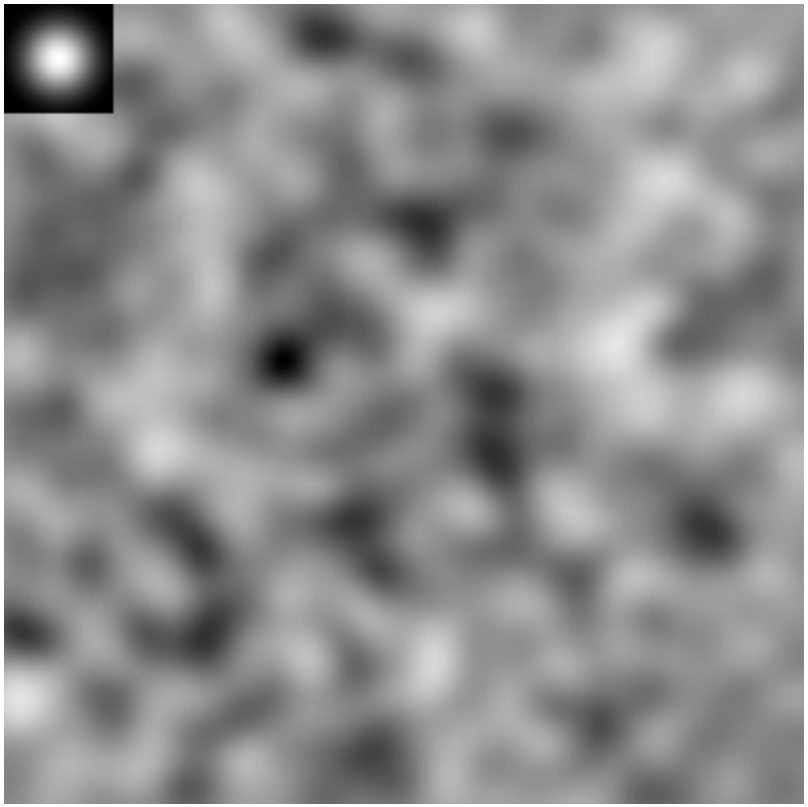
$$= \sum_{u,v} w_{i-u,j-v}^2 \sigma^2$$

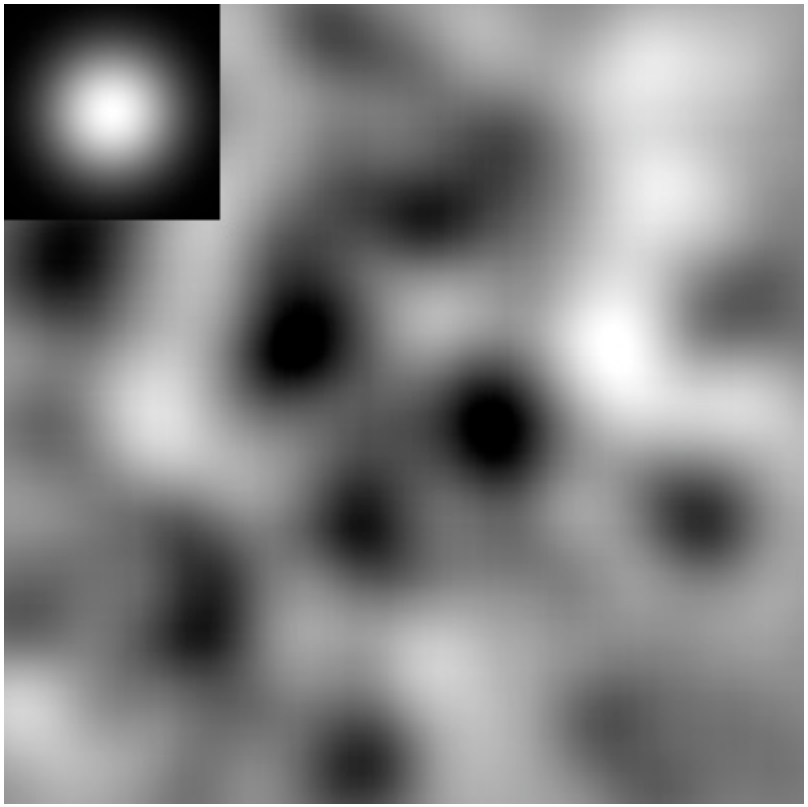
$$= \sigma^2 \sum_{u,v} w_{i-u,j-v}^2$$

Filter responses are correlated

- over scales similar to the scale of the filter
- Filtered noise is sometimes useful
  - looks like some natural textures, can be used to simulate fire, etc.

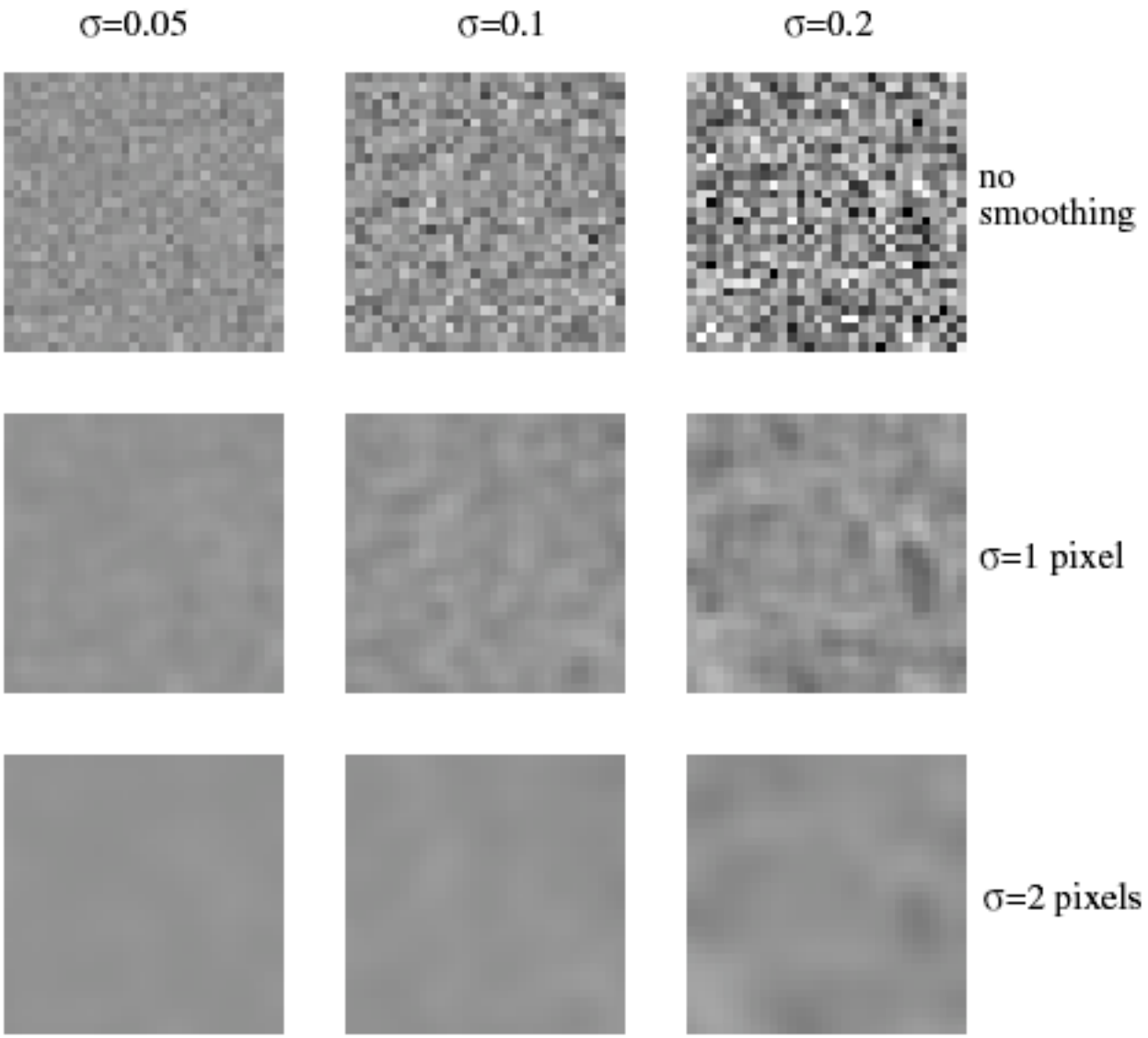






Smoothing reduces noise

- Generally expect pixels to “be like” their neighbours
  - surfaces turn slowly
  - relatively few reflectance changes
- Generally expect noise processes to be independent from pixel to pixel
- Implies that smoothing suppresses noise, for appropriate noise models
- Scale
  - the parameter in the symmetric Gaussian
  - as this parameter goes up, more pixels are involved in the average
  - and the image gets more blurred
  - and noise is more effectively suppressed

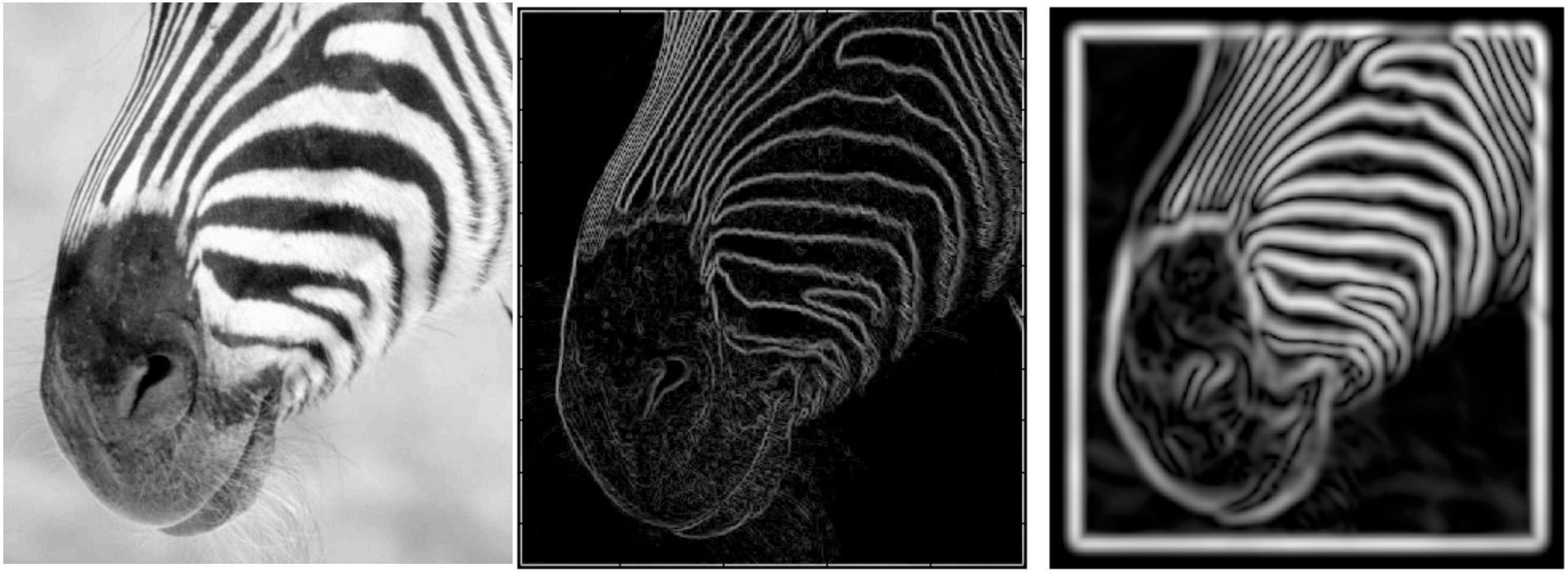


The effects of smoothing
Each row shows smoothing with gaussians of different width; each column shows different realisations of an image of gaussian noise.

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).

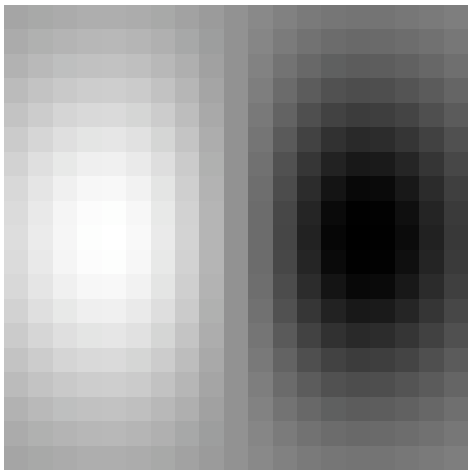


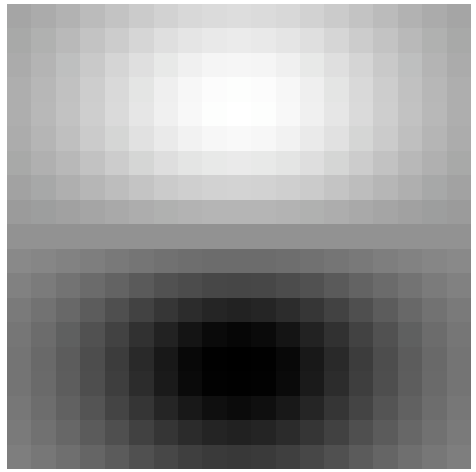
There are three major issues:

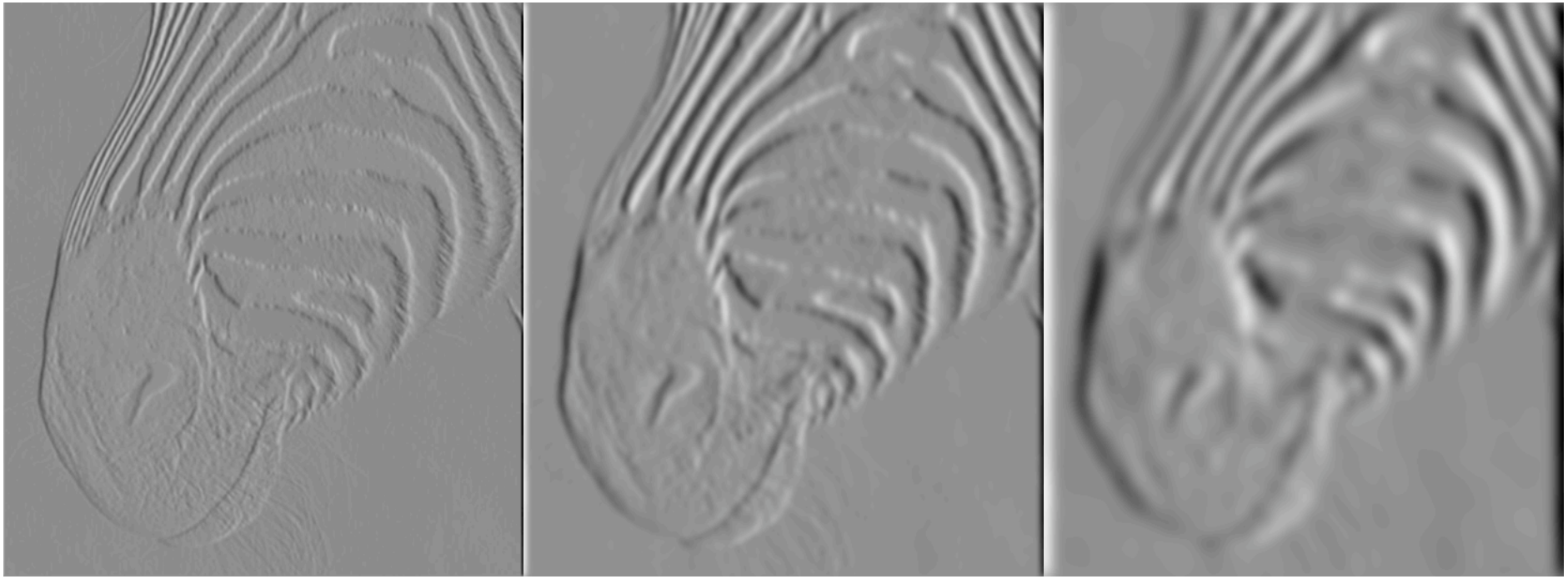
1) The gradient magnitude at different scales is different; which should we choose?
2) The gradient magnitude is large along thick trail; how do we identify the significant points?
3) How do we link the relevant points up into curves?

Smoothing and Differentiation

- Issue: noise
  - smooth before differentiation
  - two convolutions to smooth, then differentiate?
  - actually, no - we can use a derivative of Gaussian filter
    - because differentiation is convolution, and convolution is associative







1 pixel

3 pixels

7 pixels

The scale of the smoothing filter affects derivative estimates, and also the semantics of the edges recovered.