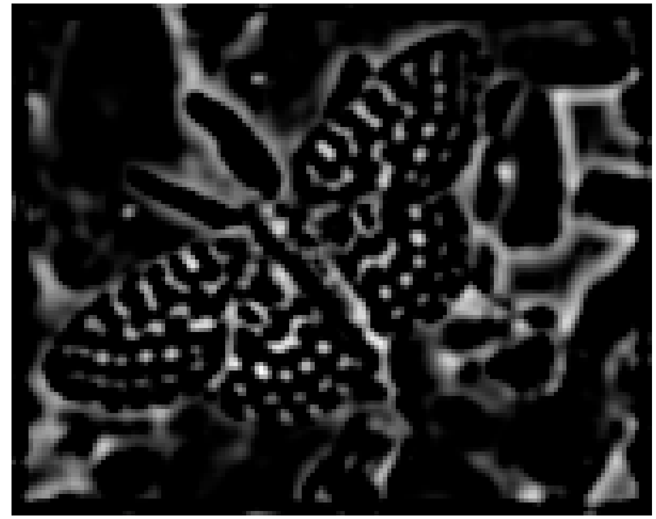
Normalized correlation

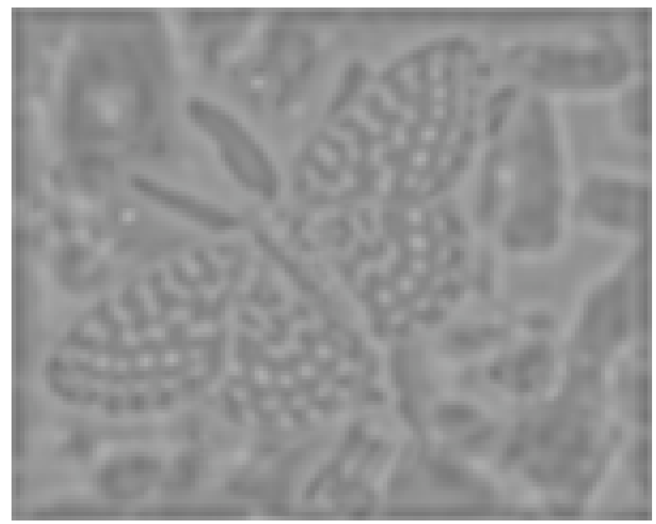
- Think of filters of a dot product
  - **problem**: brighter parts give bigger results even if the structure is same (often not what you want)
  - can think in terms of angle (i.e, its cosine)
  - **normalized** correlation output is filter output, divided by root sum of squares of values over which filter lies
- Tricks:
  - ensure that filter has a zero response to a constant region (helps reduce response to irrelevant background)
  - subtract image average when computing the normalizing constant (i.e. subtract the image mean in the neighbourhood)
  - absolute value can deal with contrast reversal (when you don't want to differentiate between white left of black and vice versa)



Positive responses

Zero mean image, -1:1 scale

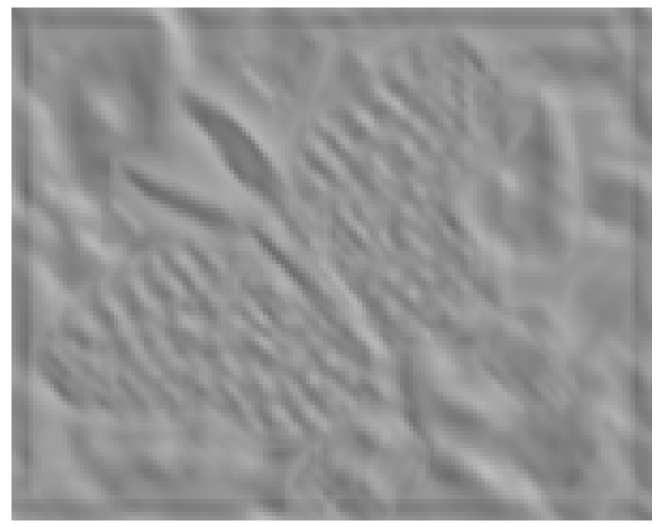
Zero mean image, -max:max scale



Positive responses

Zero mean image, -1:1 scale

Zero mean image, -max:max scale



Finding Edges

- Edges reveal much about images, and experience suggests considering them is useful (but there is debate!)
- Edge representations can be seen as information compression (because boundary is fewer pixels than the inside)
- Edges are the result of many different things
  - simple material change (step edge, corners)
  - illumination change (often soft, but not always)
  - shading edges and bar edges in inside corners
- An edge is basically where the images changes---hence finding images is studying changes (differentiation)

Differentiation and convolution

- Recall

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

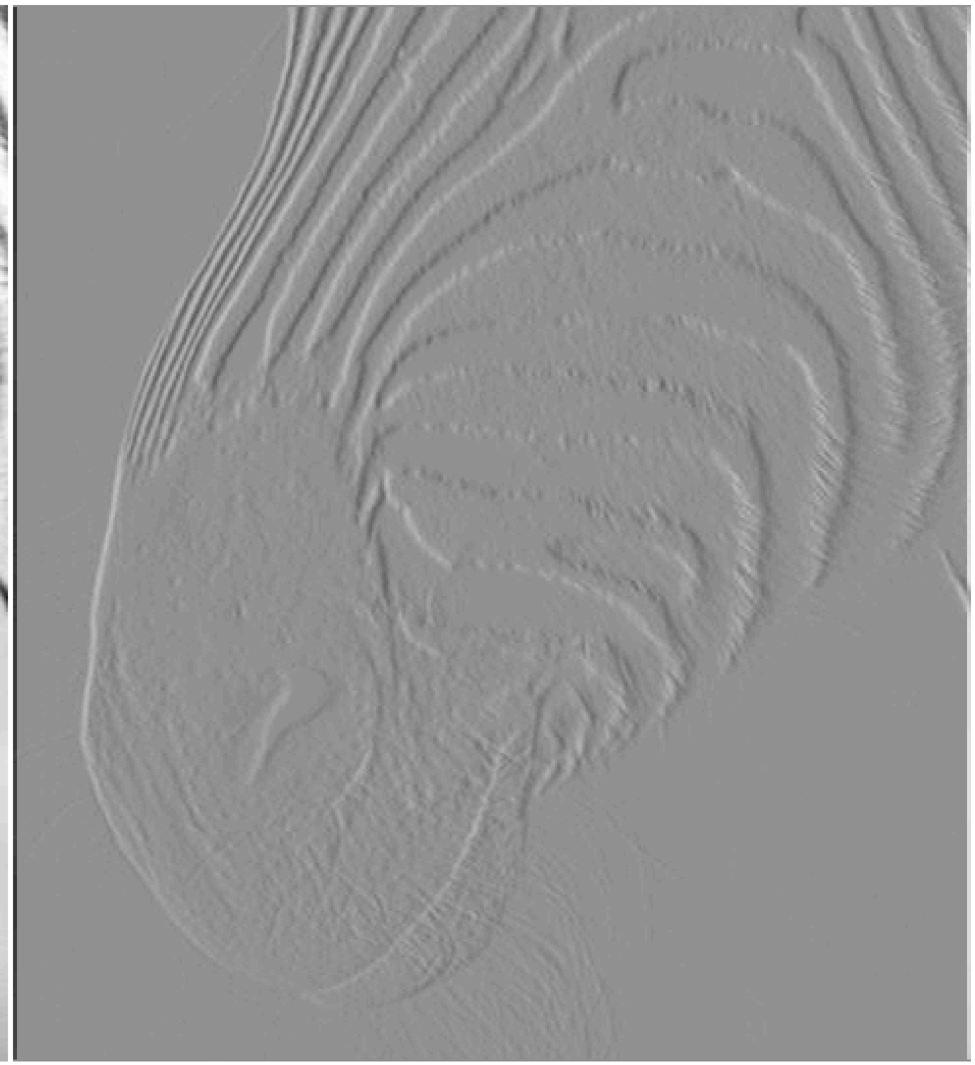
- Now this is linear and shift invariant, so must be the result of a convolution.

- We could approximate this as

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

(which is obviously a convolution; it's not a very good way to do things, as we shall see)

Finite differences (x-direction)



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
- 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)

image with added
Gaussian noise
(sigma=1)

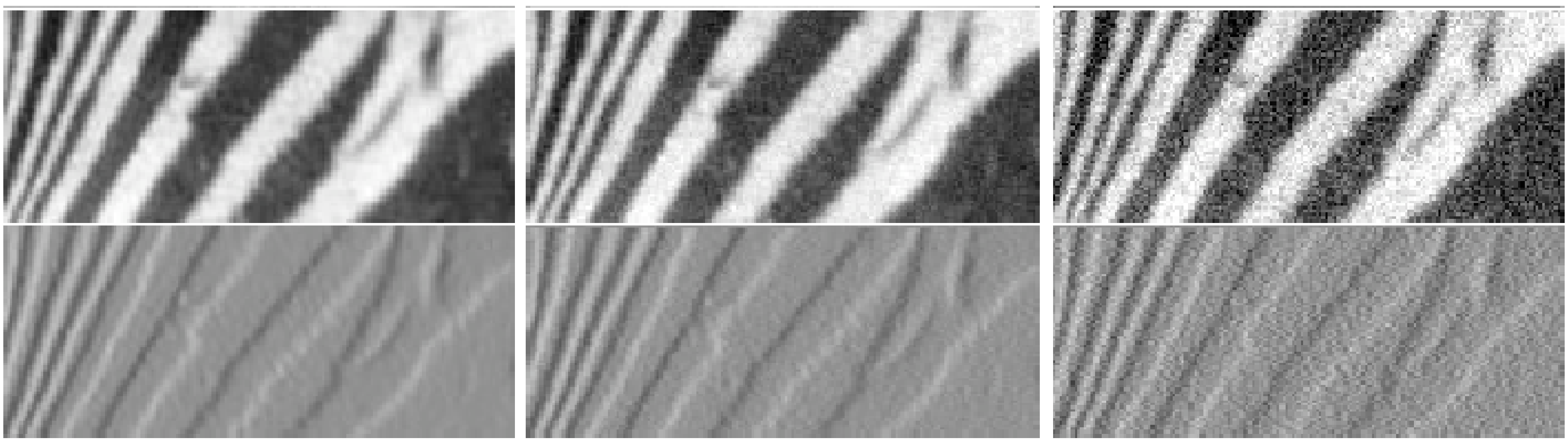
image with added
Gaussian noise
(sigma=16)

Finite differences and noise

- Finite difference filters respond strongly to noise
  - noise is not correlated across adjacent pixels, but the pixels tend to be correlated-->thus differences lock onto the noise!
- Generally, the larger the noise the stronger the response

- What is to be done?
  - 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

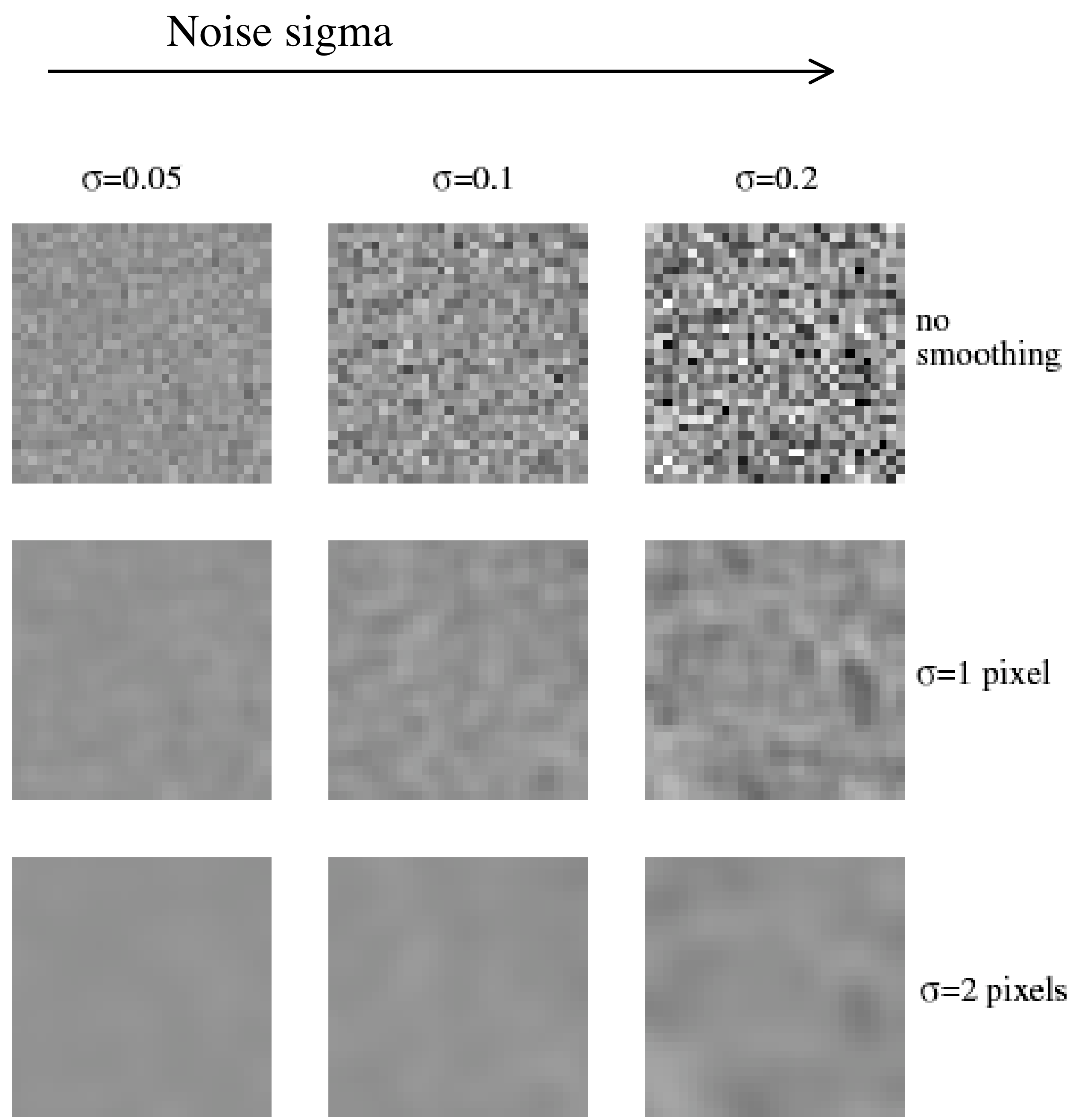


Increasing noise ->
(this is zero mean additive gaussian noise)

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 realizations of an image of gaussian noise.

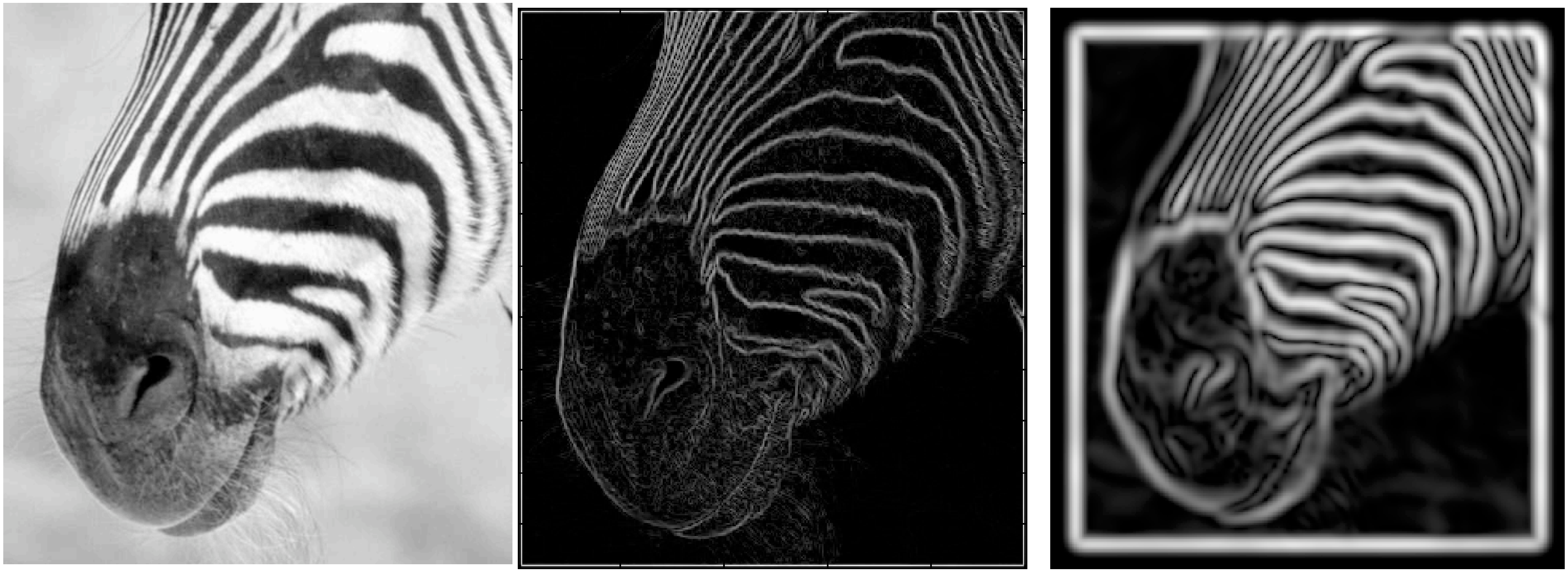


Smoothing sigma

Gradients and edges

- Sources of points of sharp change in an image:
  - change in reflectance
  - change in object
  - change in illumination
  - noise
- Sometimes called **edge points**

- General strategy
  - determine image gradient
  - mark points where gradient magnitude is particularly large wrt neighbours (ideally, curves of such points).

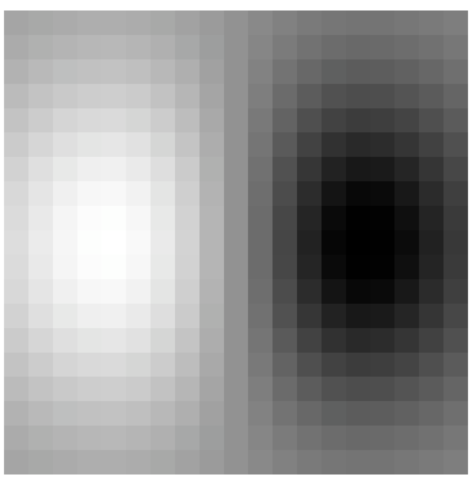


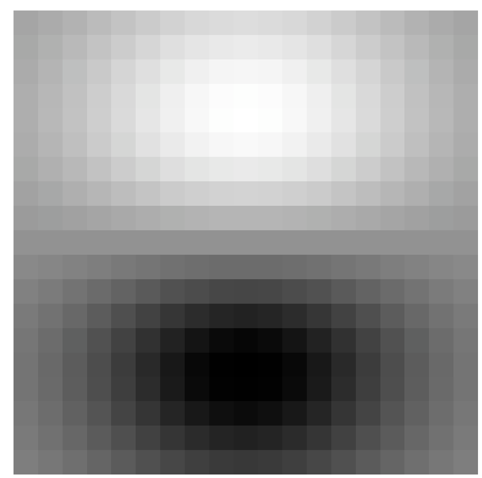
There are three major issues:

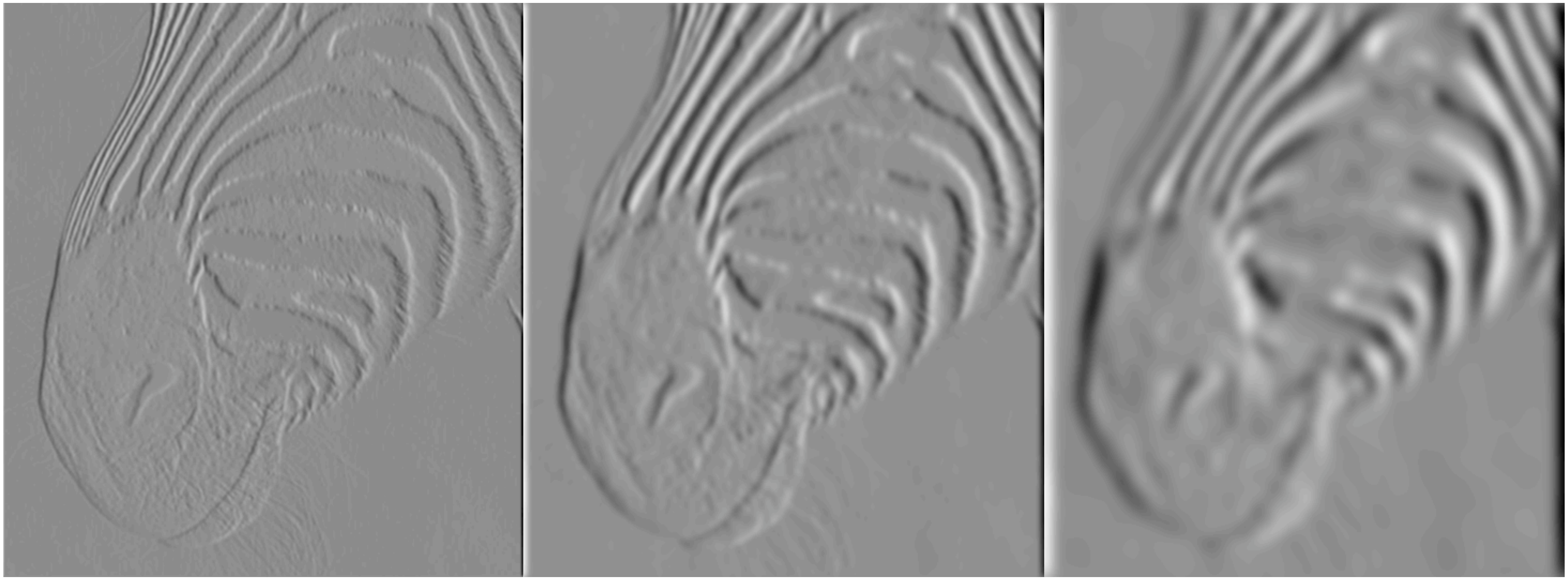
1) The gradient magnitude at different scales is different; which should we choose?
2) The gradient magnitude is large along thick trail; how do we identify the significant points?
3) How do we link the relevant points up into curves?

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





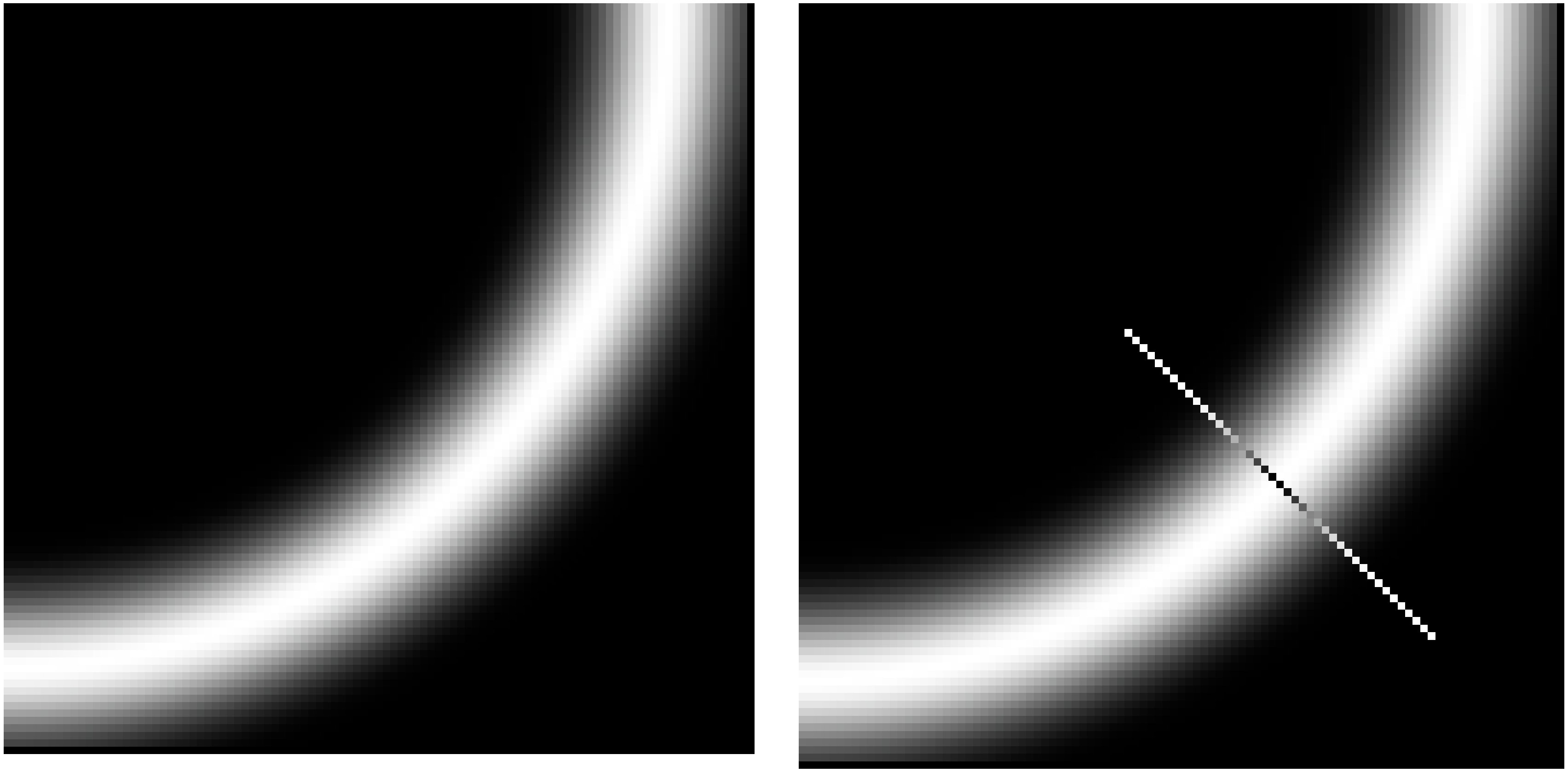


1 pixel

3 pixels

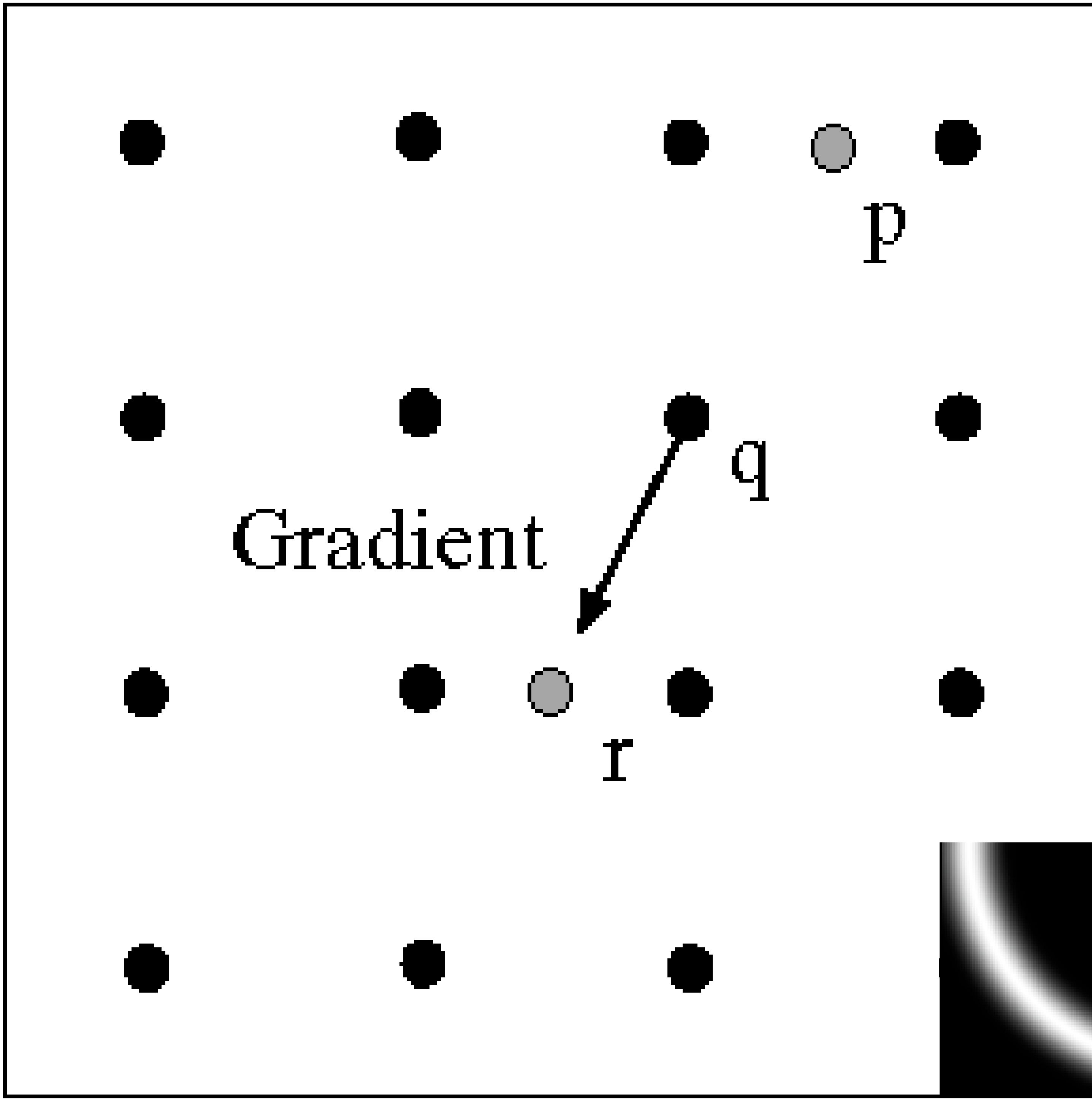
7 pixels

The scale of the smoothing filter affects derivative estimates (X direction is shown), and also the semantics of the edges recovered.



We wish to mark points along the curve where the gradient magnitude is biggest. We can do this by looking for a maximum along a slice normal to the curve (non-maximum suppression). These points should form a curve. There are then two algorithmic issues: at which point is the maximum, and where is the next one?

Non-maximum suppression



At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.