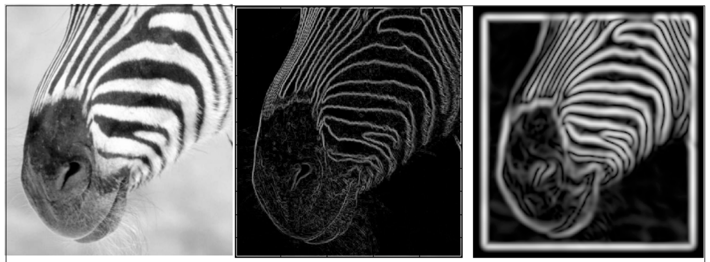
Gradients and edges

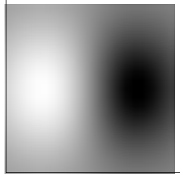
- Sources of points of sharp change in an image:
  - change in reflectance
  - change in object
  - change in illumination
  - noise!
- Sometimes called **edge points**
- General strategy
  - determine image gradient
  - mark points where gradient magnitude is particularly large compared to that of neighbours
  - attempt to promote linked edge points

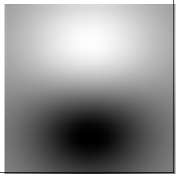


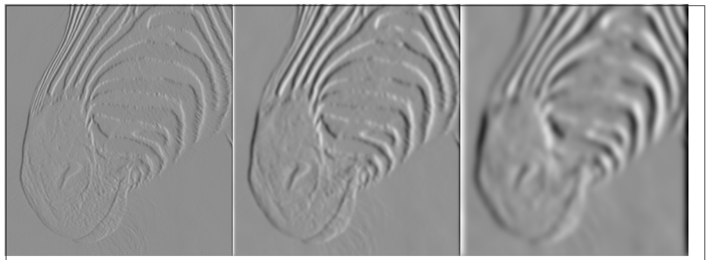
There are three major issues:
1) The gradient magnitude at different scales is different; which should we choose?
2) The gradient magnitude is large along thick trail; how do we identify the significant points?
3) How do we link the relevant points up into curves?

Smoothing and Differentiation

- Issue: noise
  - so we smooth before differentiation
  - this suggests a convolution to smooth, then a convolution to differentiate
  - but we can use a derivative of Gaussian filter
    - because differentiation is convolution, and convolution is associative







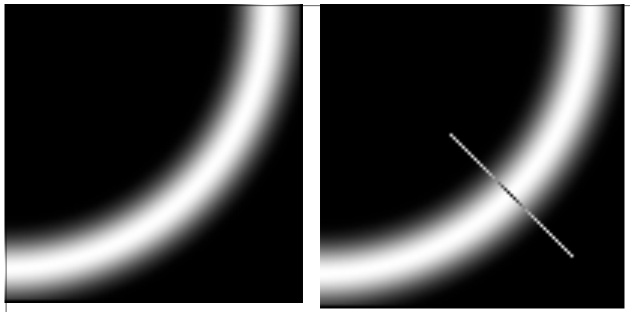
1 pixel 3 pixels 7 pixels

Horizontal derivative magnitude at different scales

The scale affects the estimates and the semantics of the edges recovered.

Non-maximal suppression (alg 8.2)

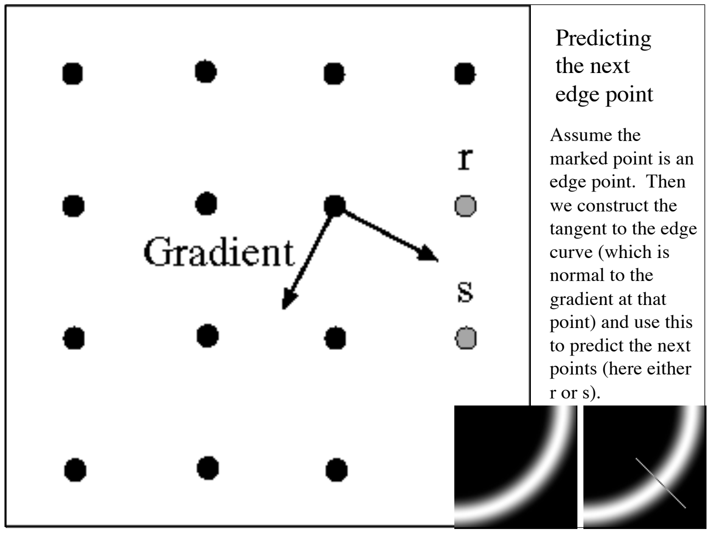
- Given a scale, how do we find edge points?
- If we set a threshold, then either lots of points near the edge are accepted, or none are.



We wish to mark points along the curve where the gradient magnitude is biggest in the gradient direction (best edge points).

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.



Predicting the next edge point

Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).

Non-maximal suppression (alg 8.2)

(See book, page 180)

For non-marked points with sufficiently large gradient

Find a maximum along gradient, marking max as edge point, others as non edge.

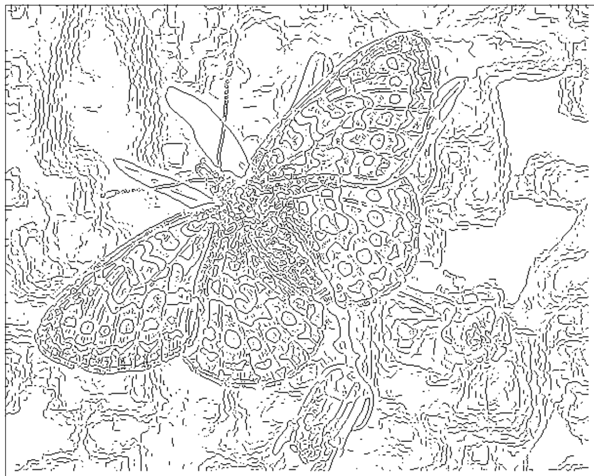
Follow chain by looking perpendicular to gradient for points which are local max in gradient direction, and marking them as edges if their gradient magnitude is big enough, and marking other visited points as non-edge.

Remaining issues

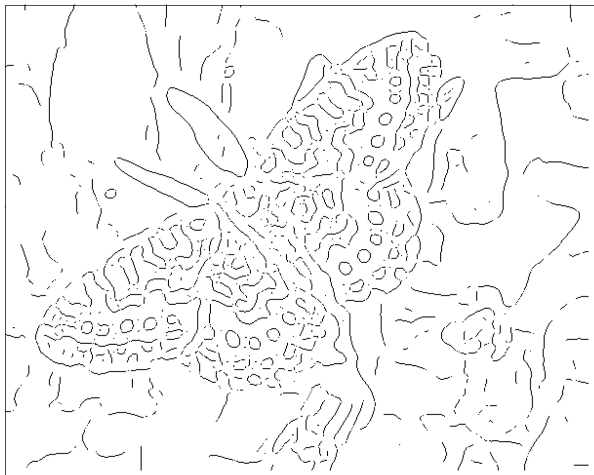
- Check that maximum value of gradient value is sufficiently large
  - **hysteresis** method
    - use a high threshold to start edge curves and a low threshold to continue them.

Notice

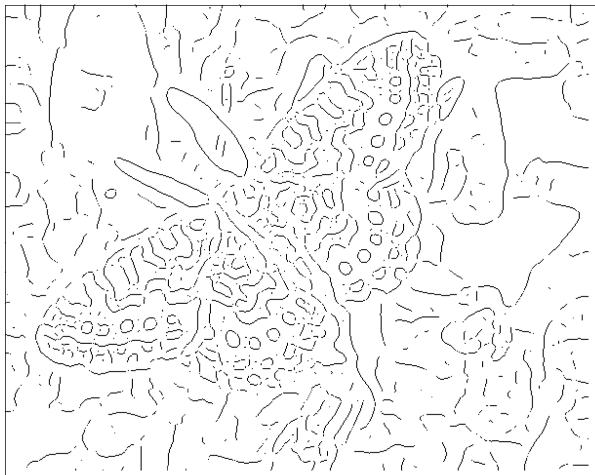
- Theory does not really match what happens at corners and edge detectors often do badly at them
- Edges aren't bounding contours (this is the hard part!)
- Scale affects contrast. Typically one analyzes images at different scales to find different structures.



fine scale
high
threshold



coarse
scale,
high
threshold



coarse
scale
low
threshold

Introduction to Fourier methods

- Very brief introduction. We don’t have time to go through the math!
- Fourier methods give insight into image processing
- Provides a principled way to think about reversing the effect of a convolution (e.g., deblurring).
- Provides a way to speed up convolution (depending on the work flow).

- SEE SUPPLEMENTARY MATERIAL AND OPTIONAL ASSIGNMENT FOR MORE DETAILS.

The Convolution Theorem

- Important result which can have practical impact (convolution theorem)

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

- (Depending on your workflow, using the DFT for convolution can save time).

- A strategy for inverting the effect of a convolution

$$a = F^{-1}\left(F(a)\right) = F^{-1}\left(\frac{F(a \otimes b)}{F(b)}\right)$$

Fourier Transform (practice)

- Because of the convolution theorem, the FT gives a convenient way to invert the effect of convolution.
  - For example, often blurring can be modeled as a convolution, and the FT gives a convenient way to think about de-blurring.
- Fast (O(n*log(n)) methods exist to compute discrete version of Fourier transform (DFT2 in Matlab, IDFT2 for the inverse).
- If we assume that the image is periodic and symmetric then only the cosine terms count and we can avoid imaginary components which can speed up and simplify some tasks (cosine transform; DCT2 in Matlab, IDCT2 for the inverse).