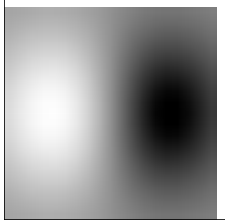
ISTA 352

Lecture 37
Image Analysis (IV, edges and segmentation)

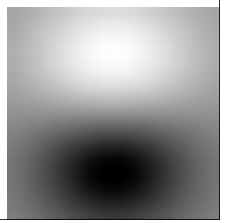
Filters are templates

- Applying a filter at some **point** can be seen as taking a dot-product between the image and some vector
- Filtering the image yields a set of dot products

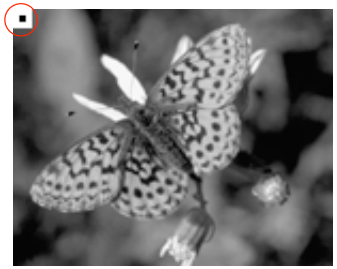
- Useful intuition
  - Filters look like the effects they are intended to find.
  - Filters find effects that look like them.
  - Remember to flip your filter if you are implementing correlation using convolution.

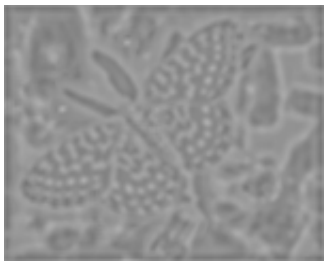


Filters for steps in X (left) and Y (right). The step in X goes from high-to-low. Convolving with it finds high-to-low steps due to the flip.

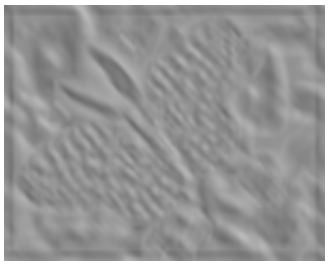


Filter for a dark spot





Filter for a dark bar at 135^0



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)
  - example: detecting change with a step filter
  - normalized correlation: output is filter output, divided by root sum of squares of values over which filter lies

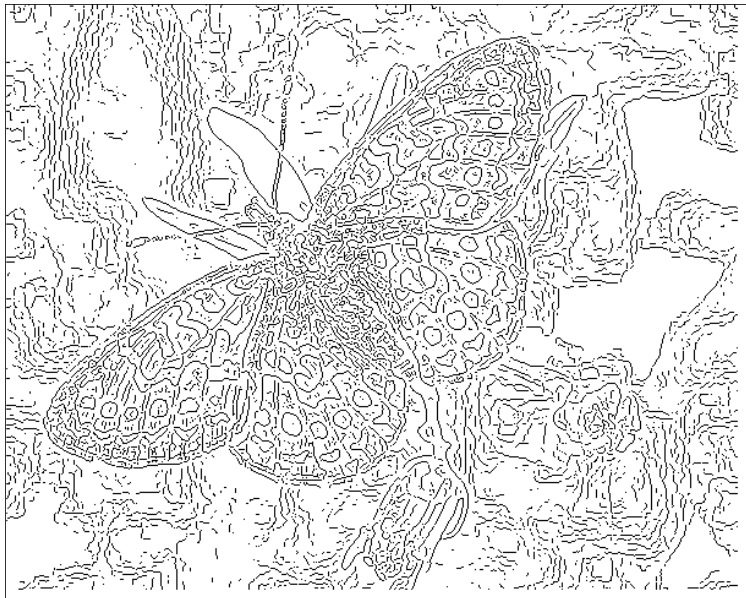
$$\frac{\mathbf{h} \bullet \mathbf{f}}{|\mathbf{f}|}$$

(|**h**| is not relevant to relative values for each of the pixels)

  - tip for constructing h: consider template filters that have zero response to a constant region (reduces response to irrelevant background).

Finding Edges

- Edges reveal much about images
- 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)



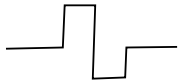
fine scale
high
threshold

Differentiation and convolution

- We approximate derivative in x direction by

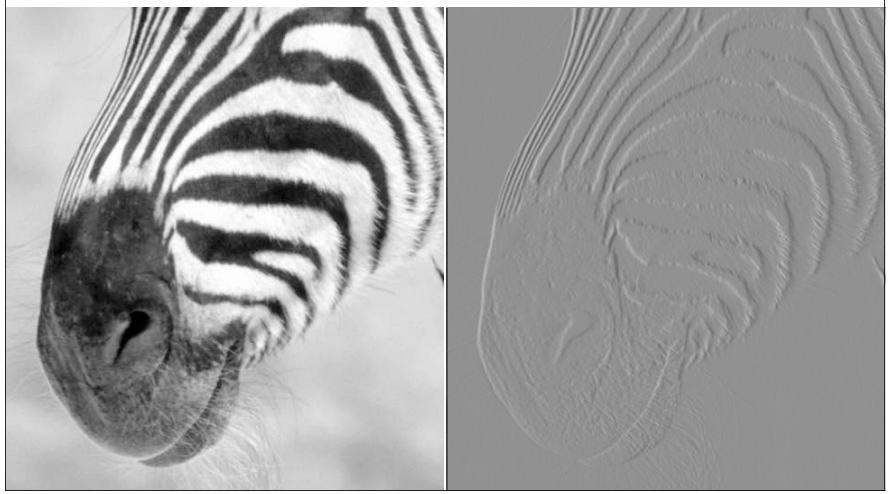
$$\frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}, y) - f(x_n, y)}{\Delta x}$$

- This is convolution by (we have already seen this)



1	-1

Recall finite difference filter (x-direction)



Effects of 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

image with added Gaussian noise (sigma=1)

image with added Gaussian noise (sigma=16)

Finite differences and noise

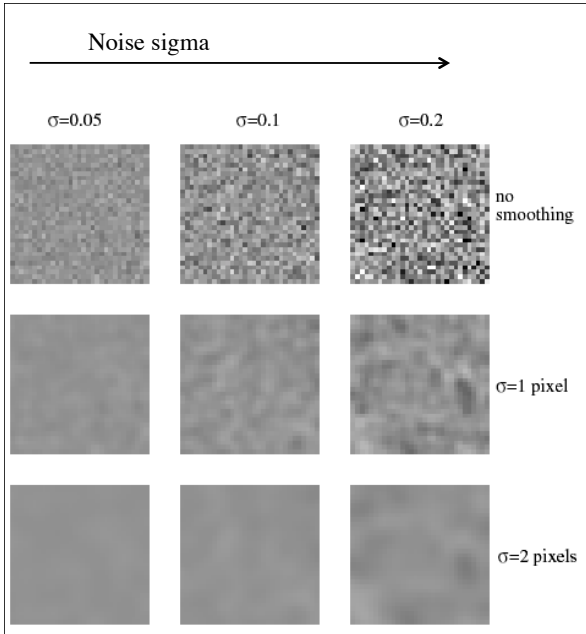
- Finite difference filters respond strongly to noise
  - Noise is not correlated across adjacent pixels, but the pixels tend to be correlated (small change on average)
  - A detected change is thus often just due to noise
  - Thus differences lock onto the noise!
- The larger the noise, the bigger such a response

Smoothing reduces noise

- Generally expect pixels to "be like" their neighbors
  - surfaces turn slowly
  - relatively few reflectance changes
- Expect noise processes to be independent from pixel to pixel
- This implies that some kind of averaging or smoothing should suppress noise, for appropriate noise models
- If we are concerned about outliers we might use a median filter
  - Non-linear, so we give up lots of nice properties.

Smoothing with a Gaussian filter

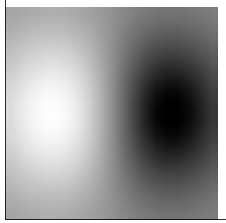
- For some noise models, a Gaussian filter optimal
- A Gaussian filter exposes edges at a specific scale
  - Noise creates high frequency edges, which we generally do not want
- Degree of smoothing <==> scale
  - the parameter in the symmetric Gaussian
  - as this parameter goes up, more pixels are involved in the average
  - and the image gets more blurred
  - and noise is more effectively suppressed

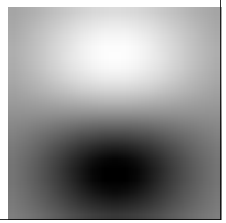


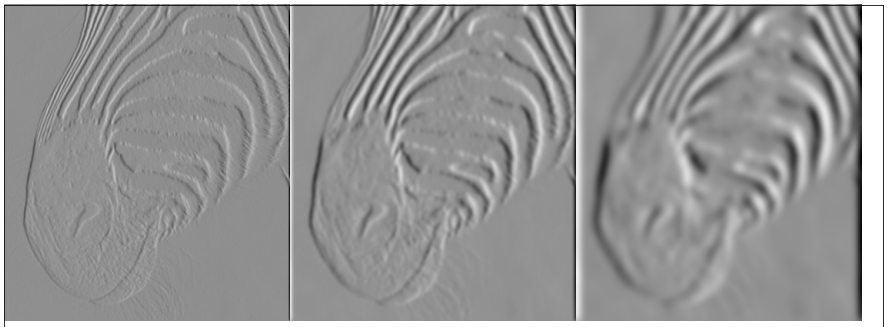
The effects of smoothing
Each row shows smoothing with gaussians of different width; each column shows different realizations of an image of gaussian noise.

Smoothing 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







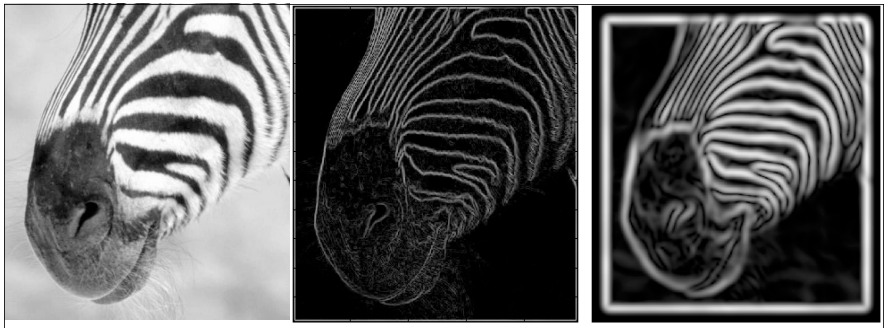
1 pixel 3 pixels 7 pixels

Horizontal derivative magnitude at different scales

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

Gradients and edges

- The gradient is a vector made of two components
  - 1) The (smoothed) difference in X (from one convolution) for the first component
  - 2) The (smoothed) difference in Y (from a different convolution) as the second component
- The magnitude of the gradient vector is a measure of edge strength
- The direction of the gradient vector gives us a direction perpendicular to the edge direction.



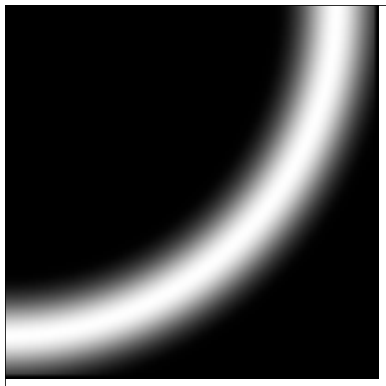
Gradient magnitudes of zebra at two different scales

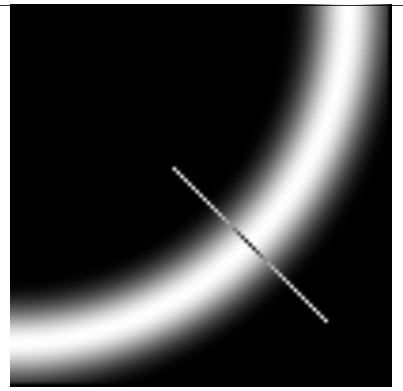
Three major issues:

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

Non-maximal suppression

- 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.

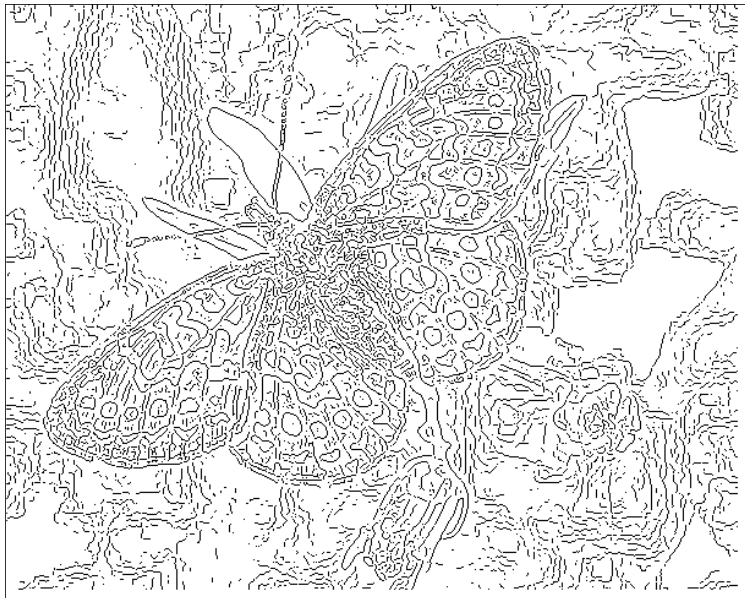




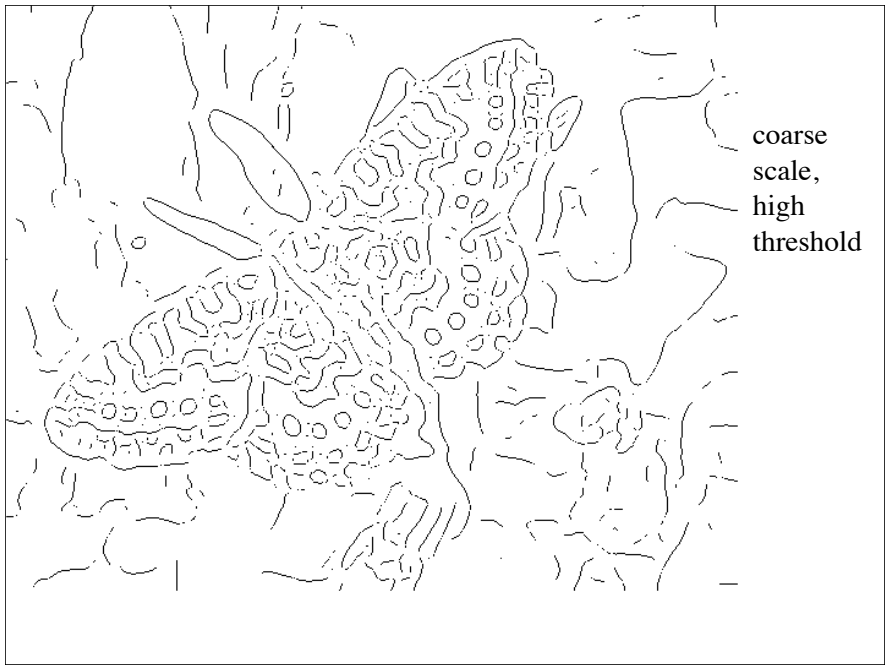
We wish to mark points along the curve where the gradient magnitude is biggest in the gradient direction (best edge points).

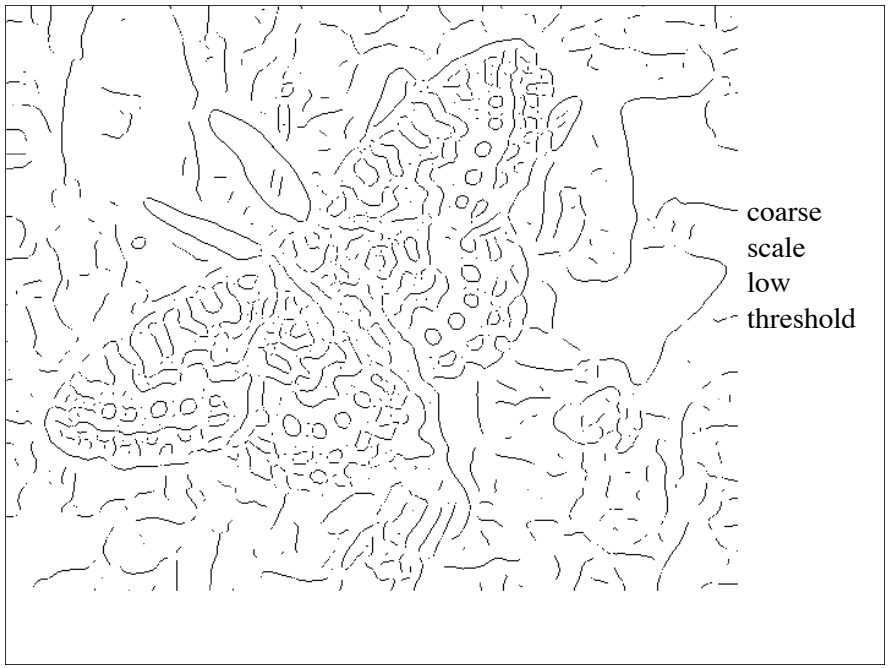
We can do this by looking for a maximum along a slice normal to the curve (non-maximum suppression).

These points should form a curve.



fine scale
high
threshold


coarse
scale,
high
threshold


coarse
scale
low
threshold