Linear Filters and Edges

Tuesday, Sept 8

Last time

- Various models for image “noise”
- Linear filters and convolution useful for
  - Image smoothing, removing noise
    - Box filter
    - Gaussian filter
    - Impact of scale / width of smoothing filter
- Separable filters more efficient
- Median filter: a non-linear filter, edge-preserving

Filter f = 1/9 x [1 1 1 1 1 1 1 1 1]

original image h   filtered

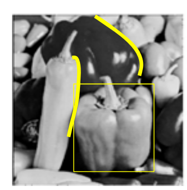
Filter f = 1/9 x [1 1 1 1 1 1 1 1 1]^T

original image h   filtered

Today

- Template matching
- Gradient images, derivative filters
  - Seam carving
- Edge detection

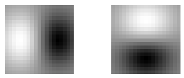
Filters for features



- Previously, thinking of filtering as a way to remove or reduce **noise.**
- Now, consider how filters will allow us to abstract higher-level “**features**”.
  - Map raw pixels to an intermediate representation that will be used for subsequent processing
  - Goal: reduce amount of data, discard redundancy, preserve what’s useful

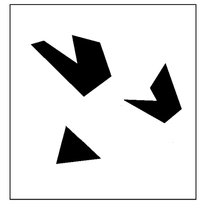
Template matching

- Filters as **templates**:
  Note that filters look like the effects they are intended to find --- “matched filters”

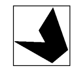


- Use normalized cross-correlation score to find a given pattern (template) in the image.
- Normalization needed to control for relative brightnesses.

Template matching



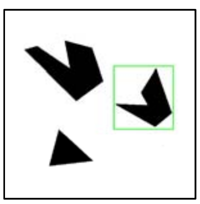
Scene



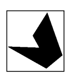
Template (mask)

A toy example

Template matching

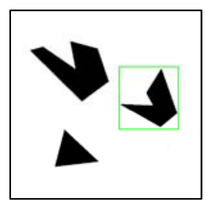


Detected template

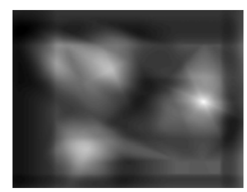


Template

Template matching



Detected template



Correlation map

Where’s Waldo?

Scene

Template

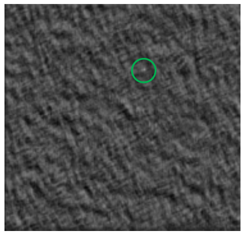
Where’s Waldo?

Detected template

Template

Where's Waldo?

Detected template



Correlation map

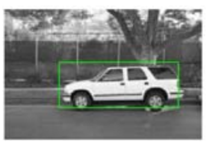
Template matching

Scene

Template

What if the template is not identical to some subimage in the scene?

Template matching



Detected template

Template

Match can be meaningful, if scale, orientation, and general appearance is right.

Edge detection

- **Goal**: map image from 2d array of pixels to a set of curves or line segments or contours.
- **Why?**

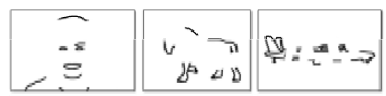
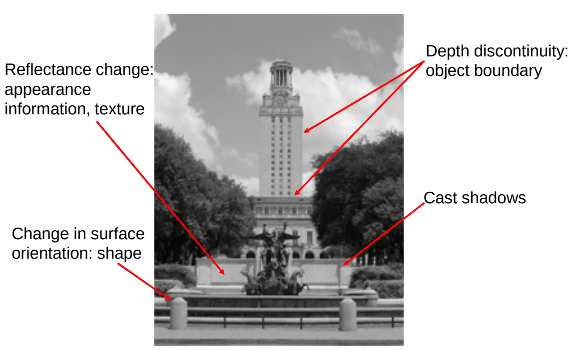


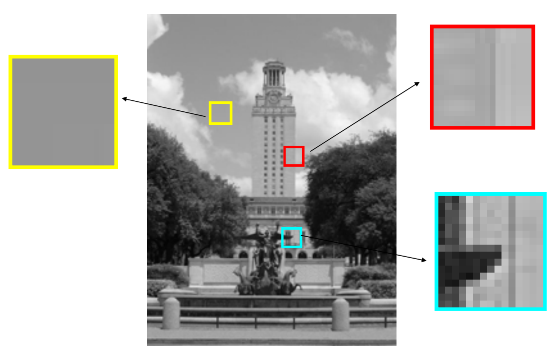
Figure from J. Shotton et al., PAMI 2007

- **Main idea**: look for strong gradients, post-process

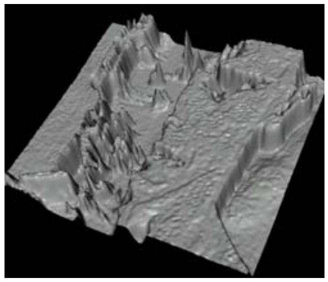
What can cause an edge?



Contrast and invariance



Recall : Images as functions

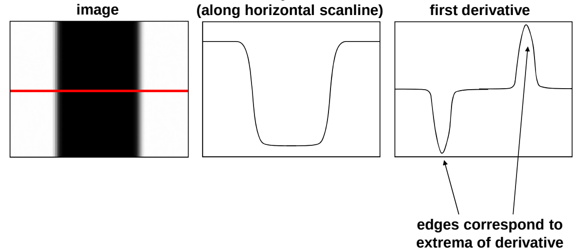


Edges look like steep cliffs

Source: S. Seitz

Derivatives and edges

An edge is a place of rapid change in the image intensity function.



Source: L. Lazebnik

Differentiation and convolution

For 2D function, f(x,y), the partial derivative is:

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

For discrete data, we can approximate using finite differences:

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

To implement above as convolution, what would be the associated filter?

Partial derivatives of an image

$$\frac{\partial f(x,y)}{\partial x}$$

$$\frac{\partial f(x,y)}{\partial y}$$

-1	1

-1
1

? or

1
-1

Which shows changes with respect to x?

(showing flipped filters)

Assorted finite difference filters

Prewitt: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$

Sobel: $M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$

Roberts: $M_x = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$; $M_y = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$

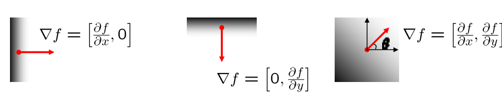
```
>> My = fspecial('sobel');
>> outim = imfilter(double(im), My);
>> imagesc(outim);
>> colormap gray;
```

Image gradient

The gradient of an image:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The gradient points in the direction of most rapid change in intensity



The gradient direction (orientation of edge normal) is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

The *edge strength* is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Slide credit Steve Seitz

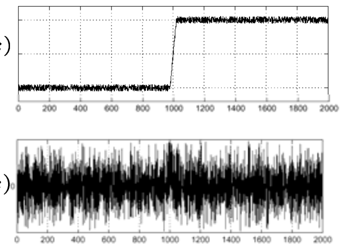
Effects of noise

Consider a single row or column of the image

- Plotting intensity as a function of position gives a signal

$f(x)$

$\frac{d}{dx}f(x)$



Where is the edge?

Slide credit Steve Seitz

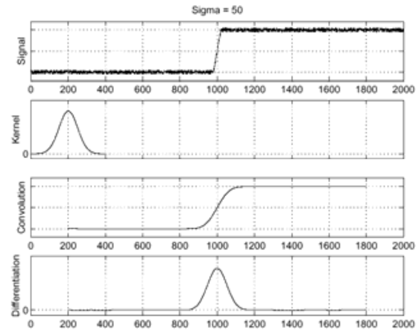
Solution: smooth first

f

h

$h \star f$

$\frac{\partial}{\partial x}(h \star f)$



Where is the edge? Look for peaks in $\frac{\partial}{\partial x}(h \star f)$

Derivative theorem of convolution

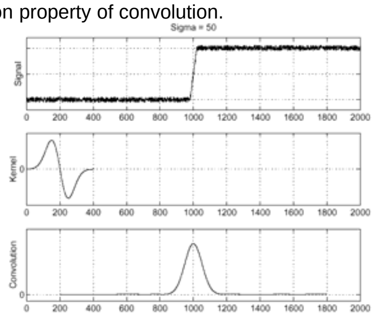
$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

Differentiation property of convolution.

f

$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) \star f$



Slide credit Steve Seitz

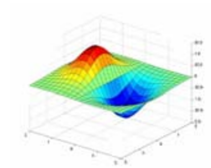
Derivative of Gaussian filter

$$(I \otimes g) \otimes h = I \otimes (g \otimes h)$$

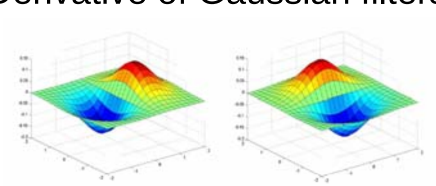
0.0030	0.0133	0.0219	0.0133	0.0030
0.0133	0.0596	0.0983	0.0596	0.0133
0.0219	0.0983	0.1621	0.0983	0.0219
0.0133	0.0596	0.0983	0.0596	0.0133
0.0030	0.0133	0.0219	0.0133	0.0030

$\otimes \begin{bmatrix} 1 & -1 \end{bmatrix}$

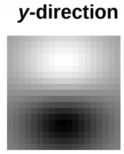
Why is this preferable?



Derivative of Gaussian filters



***x*-direction** ***y*-direction**



Source: L. Lazebnik

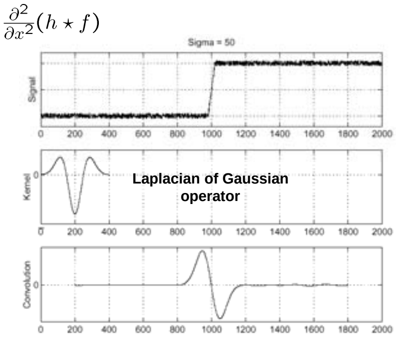
Laplacian of Gaussian

Consider $\frac{\partial^2}{\partial x^2}(h \star f)$

f

$\frac{\partial^2}{\partial x^2}h$

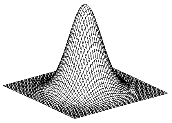
$(\frac{\partial^2}{\partial x^2}h) \star f$



Where is the edge? **Zero-crossings of bottom graph**

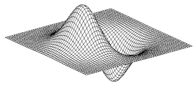
Slide credit: Steve Seitz

2D edge detection filters



Gaussian

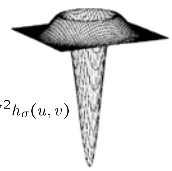
$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$



derivative of Gaussian

$\frac{\partial}{\partial x} h_\sigma(u,v)$

Laplacian of Gaussian



$\nabla^2 h_\sigma(u,v)$

- **∇^2 is the Laplacian operator:**

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

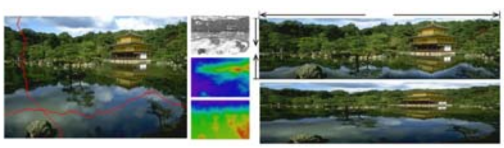
Slide credit: Steve Seitz

Mask properties

- Smoothing
  - Values positive
  - Sum to 1 → constant regions same as input
  - Amount of smoothing proportional to mask size
  - Remove "high-frequency" components; "low-pass" filter
- Derivatives
  - Opposite signs used to get high response in regions of high contrast
  - Sum to 0 → no response in constant regions
  - High absolute value at points of high contrast
- Filters act as templates
  - Highest response for regions that "look the most like the filter"
  - Dot product as correlation

Seam Carving

- Pset 1 out today, due Sept 21.
  - Programming problem uses gradients to do "seam carving":



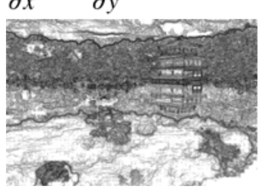
Seam Carving

Content-aware resizing

Traditional resizing

Seam Carving

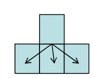
- Energy function: $e_1(\mathbf{I}) = |\frac{\partial}{\partial x}\mathbf{I}| + |\frac{\partial}{\partial y}\mathbf{I}|$



- Want to remove seams where they won't be very noticeable
- Choose seam based on minimum total energy path across image.

Seam Carving

- A vertical seam **s** is a list of column indices, one for each row, where each subsequent column differs by no more than one slot.



- Optimal 8-connected path (seam):

$$s^* = \min_{\mathbf{s}} E(\mathbf{s}) = \min_{\mathbf{s}} \sum_{i=1}^{n} e(\mathbf{I}(s_i))$$

can be computed with dynamic programming.

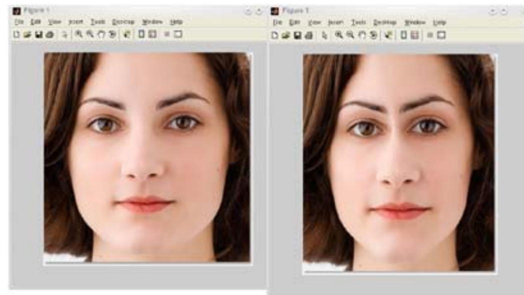
- Compute the cumulative minimum energy for all possible connected seams at each entry (i,j):

$$M(i,j) = e(i,j) + \min(M(i-1,j-1), M(i-1,j), M(i-1,j+1))$$

- Backtrack from min value in last row of *M* to pull out optimal seam path.
- *Example*

Some results from last year…

Andy Luong



Aaron Strahan

Birgi Tamersoy

(Kung Fu Panda on a diet)

Antonio Arocha

Matthew deWet

Andrew Harp

Coming up

- Thursday: finish edges, binary image analysis
- Pset 1 out today, due Sept 21.