CS6670: Computer Vision

Noah Snavely

Lecture 2: Edge detection and resampling

From Sandlot Science

Administrivia

- New room starting Thursday: HLS B11

Administrivia

- Assignment 1 (feature detection and matching) will be out Thursday
  - Turning via CMS: https://cms.csuglab.cornell.edu/
- Mailing list: please let me know if you aren't on it

Reading

- Szeliski: 3.4.1, 3.4.2

Last time: Cross-correlation

Let F be the image, H be the kernel (of size 2k+1 x 2k+1), and G be the output image

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

This is called a **cross-correlation** operation:

$$G = H \otimes F$$

Last time: Convolution

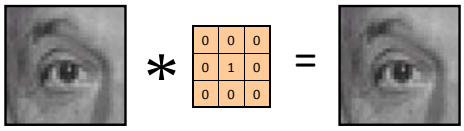
- Same as cross-correlation, except that the kernel is "flipped" (horizontally and vertically)

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

This is called a **convolution** operation:

$$G = H * F$$

Linear filters: examples

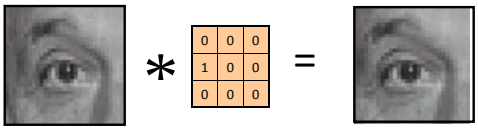


Original

Identical image

Source: D. Lowe

Linear filters: examples

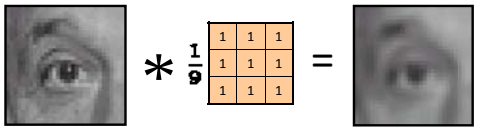


Original

Shifted left
By 1 pixel

Source: D. Lowe

Linear filters: examples

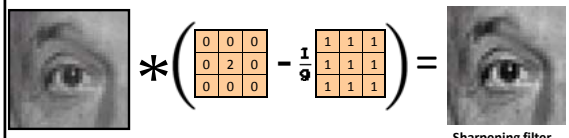


Original

Blur (with a mean filter)

Source: D. Lowe

Linear filters: examples



Original

Sharpening filter
(accentuates edges)

Source: D. Lowe

Image noise

Original image
$F[x, y]$

Gaussian noise
$F[x, y] + \mathcal{N}(0, \sigma)$

Salt and pepper noise
(each pixel has some chance of being switched to zero or one)

http://theory.uchicago.edu/~ejm/pix/20d/tests/noise/index.html

Gaussian noise

$F[x, y] + \mathcal{N}(0, 5\%)$ σ = 1 pixel σ = 2 pixels σ = 5 pixels

Smoothing with larger standard deviations suppresses noise, but also blurs the image

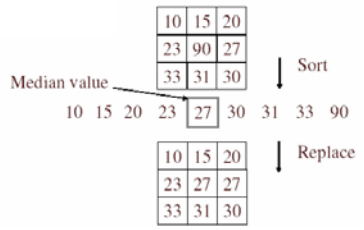
Salt & pepper noise – Gaussian blur

p = 10% σ = 1 pixel σ = 2 pixels σ = 5 pixels

- What's wrong with the results?

Alternative idea: Median filtering

- A **median filter** operates over a window by selecting the median intensity in the window

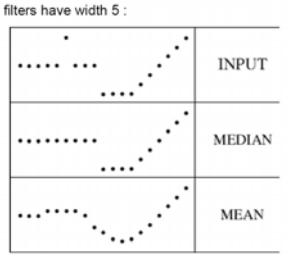


- Is median filtering linear?

Source: K. Grauman

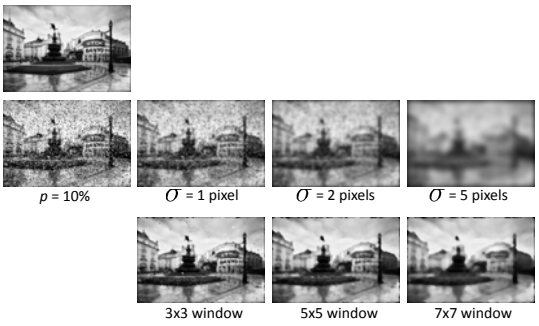
Median filter

- What advantage does median filtering have over Gaussian filtering?



Source: K. Grauman

Salt & pepper noise – median filtering

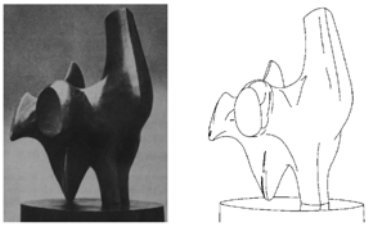


p = 10% σ = 1 pixel σ = 2 pixels σ = 5 pixels

3x3 window 5x5 window 7x7 window

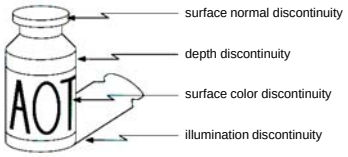
Questions?

Edge detection



- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels

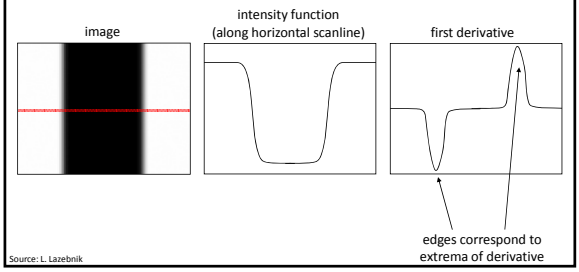
Origin of Edges



- Edges are caused by a variety of factors

Characterizing edges

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



Source: L. Lazebnik

Image derivatives

- How can we differentiate a *digital* image F[x,y]?
  - Option 1: reconstruct a continuous image, *f*, then compute the derivative
  - Option 2: take discrete derivative (finite difference)

$$\frac{\partial f}{\partial x}[x,y] \approx F[x+1,y] - F[x,y]$$

How would you implement this as a linear filter?

$\frac{\partial f}{\partial x}$: H_x $\quad$ $\frac{\partial f}{\partial y}$: H_y

Source: S. Seitz

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 increase in intensity

$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$ $\quad$ $\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$ $\quad$ $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

The *edge strength* is given by the gradient magnitude:

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

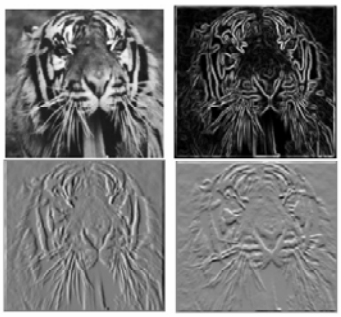
The gradient direction is given by:

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

- how does this relate to the direction of the edge?

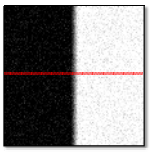
Source: Steve Seitz

Image gradient

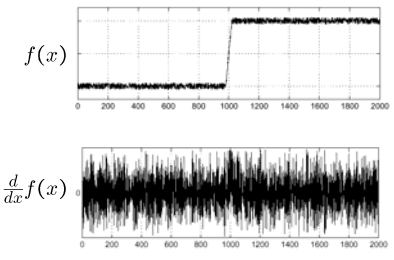


Source: L. Lazebnik

Effects of noise



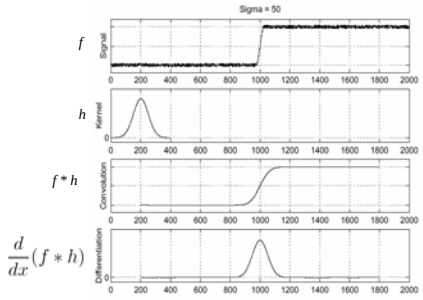
Noisy input image



Where is the edge?

Source: S. Seitz

Solution: smooth first

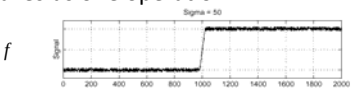


To find edges, look for peaks in $\frac{d}{dx}(f * h)$

Source: S. Seitz

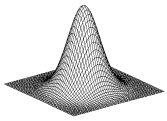
Associative property of convolution

- Differentiation is convolution, and convolution is associative: $\frac{d}{dx}(f * h) = f * \frac{d}{dx}h$
- This saves us one operation:



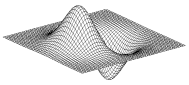
Source: S. Seitz

2D edge detection filters



Gaussian

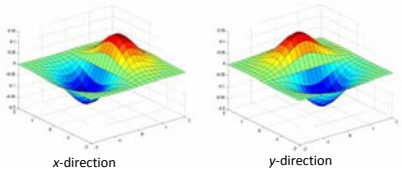
$$h_\sigma(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{2\sigma^2}}$$



derivative of Gaussian (*x*)

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

Derivative of Gaussian filter



x-direction *y*-direction

The Sobel operator

- Common approximation of derivative of Gaussian

$\frac{1}{8}$

-1	0	1
-2	0	2
-1	0	1

s_x

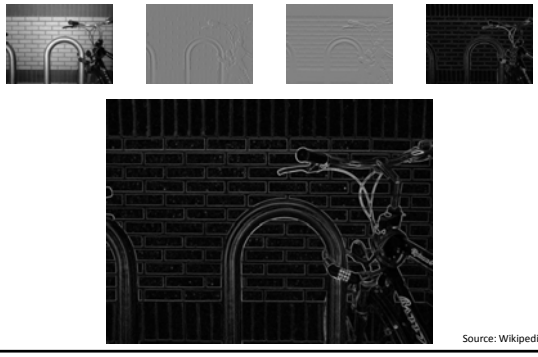
$\frac{1}{8}$

1	2	1
0	0	0
-1	-2	-1

s_y

- The standard defn. of the Sobel operator omits the 1/8 term
  - doesn't make a difference for edge detection
  - the 1/8 term **is** needed to get the right gradient value

Sobel operator: example



Source: Wikipedia

Example

- original image (Lena)

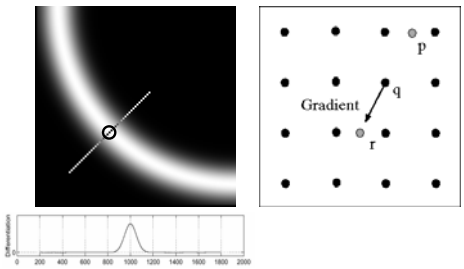
Finding edges

gradient magnitude

Finding edges

thresholding

Non-maximum supression



- Check if pixel is local maximum along gradient direction
  - requires *interpolating* pixels p and r

Finding edges

thresholding

Finding edges

thinning

(non-maximum suppression)

Canny edge detector

MATLAB: `edge(image,'canny')`

1. Filter image with derivative of Gaussian
2. Find magnitude and orientation of gradient

3. Non-maximum suppression
4. Linking and thresholding (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

Source: D. Lowe, L. Fei-Fei

Canny edge detector

- Still one of the most widely used edge detectors in computer vision

J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

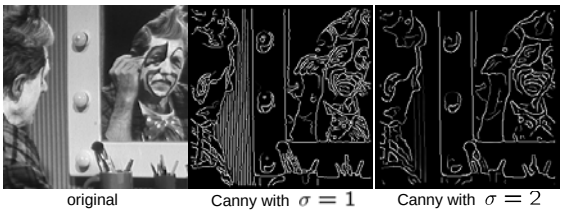
- Depends on several parameters:

σ : width of the Gaussian blur

high threshold

low threshold

Canny edge detector

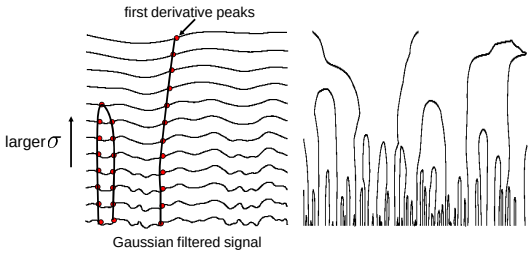


original Canny with $\sigma = 1$ Canny with $\sigma = 2$

- The choice of σ depends on desired behavior
  - large σ detects "large-scale" edges
  - small σ detects fine edges

Source: S. Seitz

Scale space (Witkin 83)

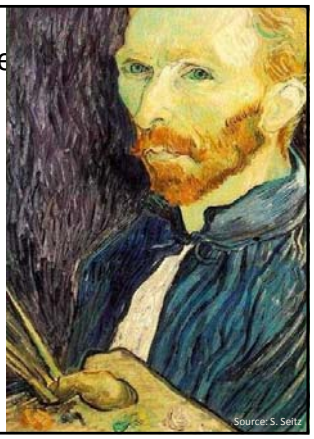


- Properties of scale space (w/ Gaussian smoothing)
  - edge position may shift with increasing scale (σ)
  - two edges may merge with increasing scale
  - an edge may ***not*** split into two with increasing scale

Questions?

Image

This image is too big to fit on the screen. How can we generate a half-sized version?



Source: S. Seitz

Image sub-sampling

1/4

1/8

Throw away every other row and column to create a 1/2 size image - called *image sub-sampling*

Source: S. Seitz

Image sub-sampling

1/2 1/4 (2x zoom) 1/8 (4x zoom)

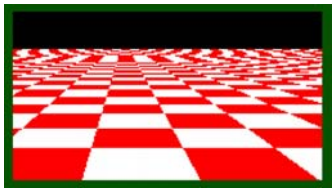
Why does this look so crufty?

Source: S. Seitz

Image sub-sampling

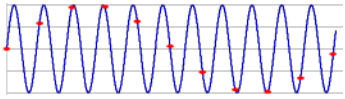
Source: F. Durand

Even worse for synthetic images



Source: L. Zhang

Aliasing



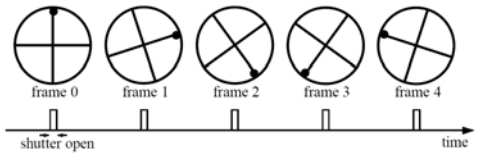
- Occurs when your sampling rate is not high enough to capture the amount of detail in your image
- Can give you the wrong signal/image—an *alias*
- To do sampling right, need to understand the structure of your signal/image
- Enter Monsieur Fourier…
- To avoid aliasing:
  - sampling rate ≥ 2 * max frequency in the image
    - said another way: ≥ two samples per cycle
  - This minimum sampling rate is called the **Nyquist rate**

Source: L. Zhang

Wagon-wheel effect

Imagine a spoked wheel moving to the right (rotating clockwise). Mark wheel with dot so we can see what's happening.

If camera shutter is only open for a fraction of a frame time (frame time = 1/30 sec. for video, 1/24 sec. for film):



Without dot, wheel appears to be rotating slowly backwards! (counterclockwise)

(See http://www.michaelbach.de/ot/mot_wagonWheel/index.html)

Source: L. Zhang

Gaussian pre-filtering

G 1/4

G 1/8

Gaussian 1/2

- Solution: filter the image, *then* subsample

Source: S. Seitz

Subsampling with Gaussian pre-filtering

Gaussian 1/2 G 1/4 G 1/8

- Solution: filter the image, *then* subsample

Source: S. Seitz

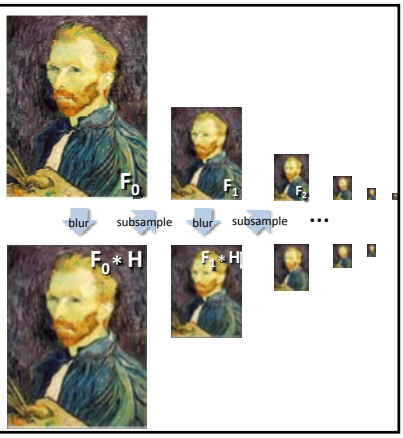
Compare with...

1/2 1/4 (2x zoom) 1/8 (4x zoom)

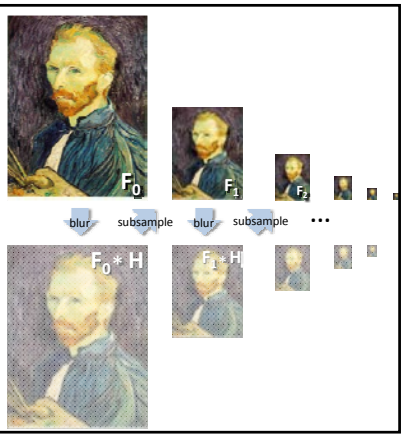
Source: S. Seitz

Gaussian pre-filtering

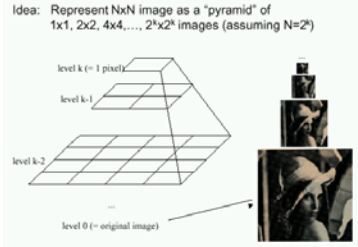
- Solution: filter the image, *then* subsample



Gaussian pyramid



Gaussian pyramids [Burt and Adelson, 1983]

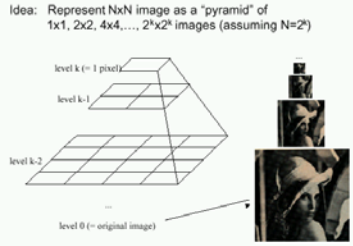


- In computer graphics, a *mip map* [Williams, 1983]
- A precursor to *wavelet transform*

Gaussian Pyramids have all sorts of applications in computer vision

Source: S. Seitz

Gaussian pyramids [Burt and Adelson, 1983]



- How much space does a Gaussian pyramid take compared to the original image?

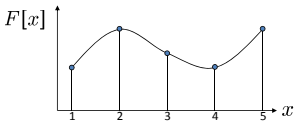
Source: S. Seitz

Questions?

Upsampling

- This image is too small for this screen:
- How can we make it 10 times as big?
- Simplest approach: repeat each row and column 10 times
- ("Nearest neighbor interpolation")

Image interpolation



d = 1 in this example

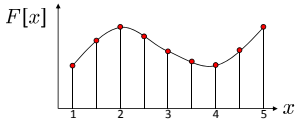
Recall how a digital image is formed

$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

- It is a discrete point-sampling of a continuous function
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale

Adapted from: S. Seitz

Image interpolation



d = 1 in this example

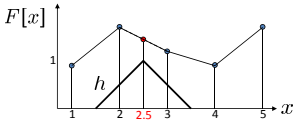
Recall how a digital image is formed

$$F[x, y] = \text{quantize}\{f(xd, yd)\}$$

- It is a discrete point-sampling of a continuous function
- If we could somehow reconstruct the original function, any new image could be generated, at any resolution and scale

Adapted from: S. Seitz

Image interpolation



d = 1 in this example

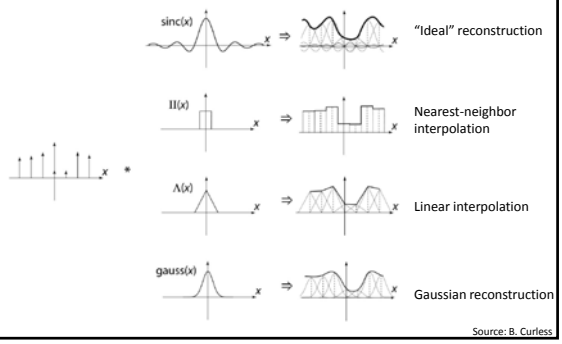
- What if we don't know f ?
  - Guess an approximation: $\tilde{f}$
  - Can be done in a principled way: filtering
  - Convert F to a continuous function:

    $f_F(x) = F(\frac{x}{d})$ when $\frac{x}{d}$ is an integer, 0 otherwise
  - Reconstruct by convolution with a *reconstruction filter*, h

    $\tilde{f} = h * f_F$

Adapted from: S. Seitz

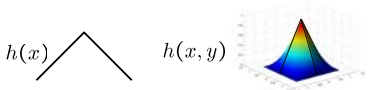
Image interpolation



Source: B. Curless

Reconstruction filters

- What does the 2D version of this hat function look like?



performs linear interpolation

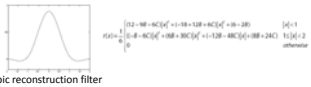
(tent function) performs **bilinear interpolation**

Often implemented without cross-correlation

- E.g., http://en.wikipedia.org/wiki/Bilinear_interpolation

Better filters give better resampled images

- **Bicubic** is common choice



Cubic reconstruction filter

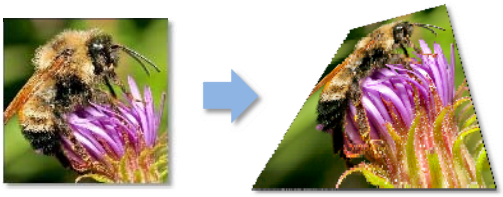
Image interpolation

Original image: x 10

Nearest-neighbor interpolation | Bilinear interpolation | Bicubic interpolation

Image interpolation

Also used for *resampling*



Questions?

Next time: image features