Why is computer vision difficult?

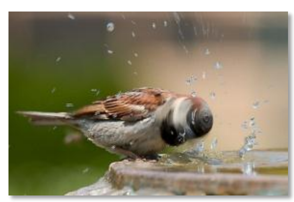
Viewpoint variation

Illumination

Scale

Why is computer vision difficult?

Intra-class variation

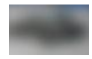


Motion (Source: S. Lazebnik)

Background clutter

Occlusion

Challenges: local ambiguity



slide credit: Fei-Fei, Fergus & T

But there are lots of cues we can exploit…

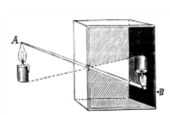
Source: S. Lazebr

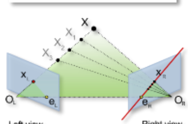
Bottom line

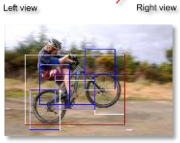
- Perception is an inherently ambiguous problem
  - Many different 3D scenes could have given rise to a particular 2D picture

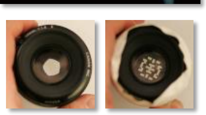
  - We often need to use prior knowledge about the structure of the world

Course overview (tentative)









1. Low-level vision
   – image processing, edge detection, feature detection, cameras, image formation
2. Geometry and algorithms
   – projective geometry, stereo, structure from motion, Markov random fields
3. Recognition
   – face detection / recognition, category recognition, segmentation
4. Light, color, and reflectance
5. Advanced topics

Projects (tentative)

- Roughly five projects
- First one will be done solo, others in groups
- You can discuss the projects on a whiteboard, but all code must be your (or your group's) own

- First project to be released today or tomorrow

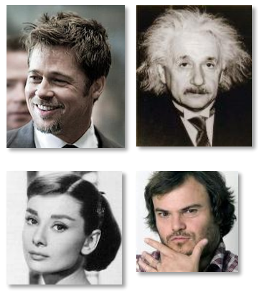
Project: Image Scissors

Project: Feature detection and matching

Project: Creating panoramas

Project: Recognition

Location recognition



Face recognition

Object category recognition

Grading

- Occasional quizzes (at the beginning of class)
- One prelim, one final exam

- Rough grade breakdown:
  - Quizzes: 5%
  - Midterm: 15%
  - Programming projects: 60%
  - Final exam: 15%

Late policy

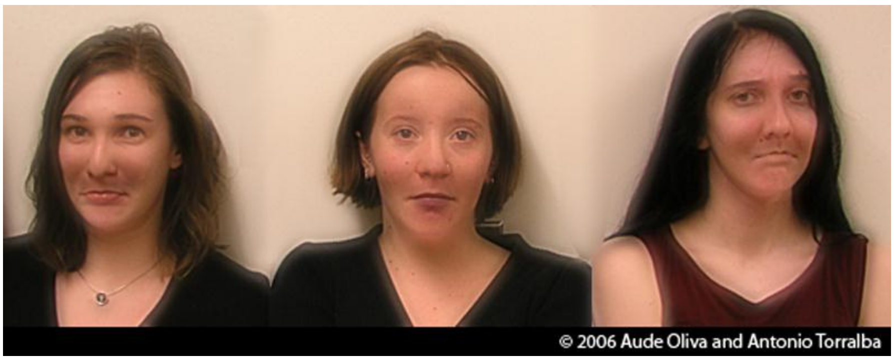
- Two “late days” will be available for the semester
- Late projects will be penalized by 25% for each day it is late, and no extra credit will be awarded.

Questions?

CS4670/5670: Intro to Computer Vision

Noah Snavely

Lecture 1: Images and image filtering



Hybrid Images, Oliva et al., http://cvcl.mit.edu/hybridimage.htm

CS4670: Computer Vision

Noah Snavely

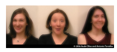
Lecture 1: Images and image filtering

Hybrid Images, Oliva et al., http://cvcl.mit.edu/hybridimage.htm

CS4670: Computer Vision

Noah Snavely

Lecture 1: Images and image filtering

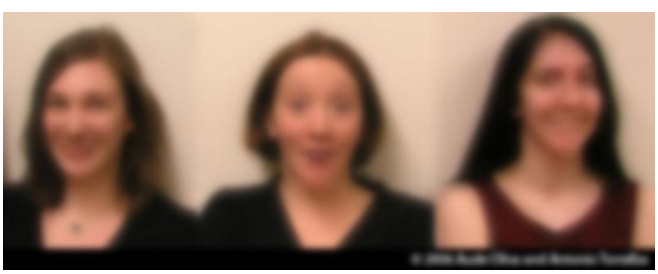


Hybrid Images, Oliva et al., http://cvcl.mit.edu/hybridimage.htm

CS4670: Computer Vision

Noah Snavely

Lecture 1: Images and image filtering



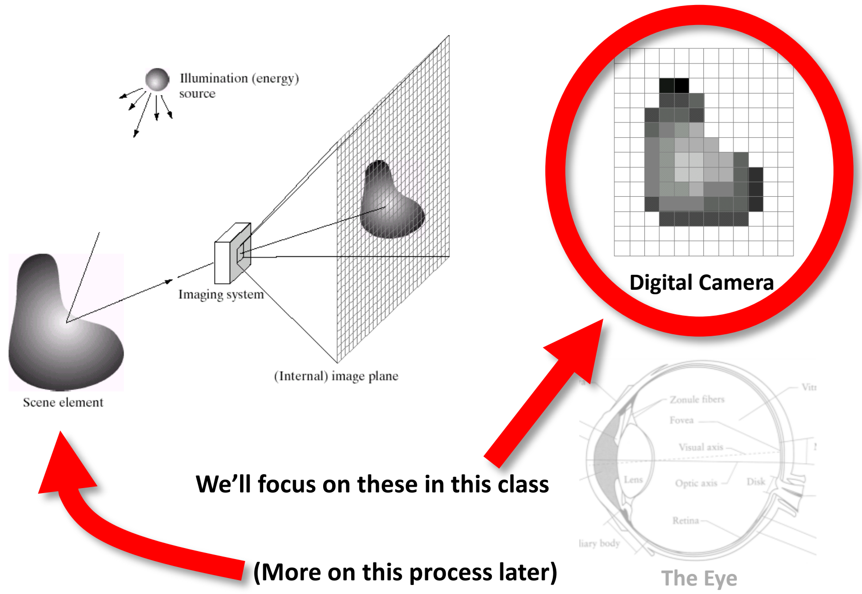
Hybrid Images, Oliva et al., http://cvcl.mit.edu/hybridimage.htm

Reading

- Szeliski, Chapter 3.1-3.2

What is an image?

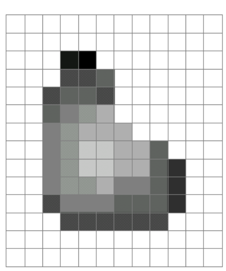
What is an image?



Source: A. Efros

What is an image?

- A grid (matrix) of intensity values



=

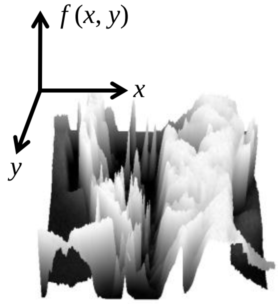
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	20	0	255	255	255	255	255	255	255
255	255	255	75	75	75	255	255	255	255	255	255
255	255	75	95	95	75	255	255	255	255	255	255
255	255	96	127	145	175	255	255	255	255	255	255
255	255	127	145	175	175	175	255	255	255	255	255
255	255	127	145	200	200	175	175	95	255	255	255
255	255	127	145	200	200	175	175	95	47	255	255
255	255	127	145	145	175	127	127	95	47	255	255
255	255	74	127	127	127	95	95	95	47	255	255
255	255	255	74	74	74	74	74	74	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255

(common to use one byte per value: 0 = black, 255 = white)

What is an image?

- We can think of a (grayscale) image as a **function**, f, from R^2 to R:
  - $f(x,y)$ gives the **intensity** at position (x,y)

snoop

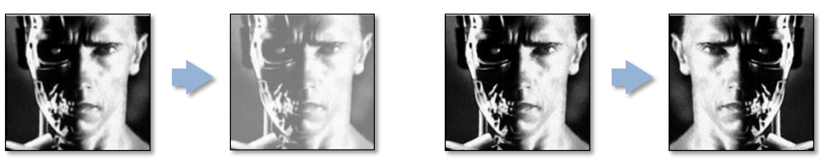


3D view

  - A **digital** image is a discrete (**sampled**, **quantized**) version of this function

Image transformations

- As with any function, we can apply operators to an image



$g(x,y) = f(x,y) + 20$ $\qquad$ $g(x,y) = f(-x,y)$

- We'll talk about a special kind of operator, *convolution* (linear filtering)

Question: Noise reduction

- Given a camera and a still scene, how can you reduce noise?

Take lots of images and average them!

What's the next best thing?

Source: S. Seitz

Image filtering

- Modify the pixels in an image based on some function of a local neighborhood of each pixel

10	5	3
4	5	1
1	1	7

Local image data

Some function →

	7	

Modified image data

Source: L. Zhang

Linear filtering

- One simple version: linear filtering (cross-correlation, convolution)
  - Replace each pixel by a linear combination (a weighted sum) of its neighbors
- The prescription for the linear combination is called the “kernel” (or “mask”, “filter”)

10	5	3
4	6	1
1	1	8

Local image data

0	0	0
0	0.5	0
0	1	0.5

kernel

	8	

Modified image data

Source: L. Zhang

Cross-correlation

Let F be the image, H be the kernel (of size 2*k*+1 x 2*k*+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:

$$G = H \otimes F$$

- Can think of as a “dot product” between local neighborhood and kernel for each pixel

Convolution

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

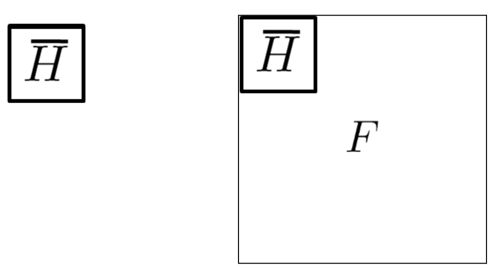
$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] F[i-u, j-v]$$

This is called a **convolution** operation:

$$G = H * F$$

- Convolution is **commutative** and **associative**

Convolution



Adapted from F. Durand

Mean filtering

H $*$ F $=$ G

H:

F:

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

G:

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Linear filters: examples

Original

$*$

0	0	0
0	1	0
0	0	0

$=$

Identical image

Source: D. Lowe

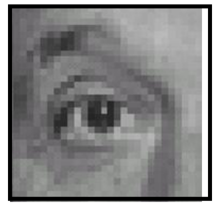
Linear filters: examples

Original

*

0	0	0
1	0	0
0	0	0

=



Shifted left
By 1 pixel

Source: D. Lowe

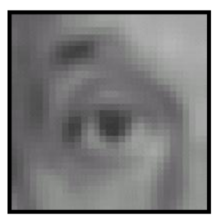
Linear filters: examples

Original

* $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

=



Blur (with a mean filter)

Source: D. Lowe

Linear filters: examples

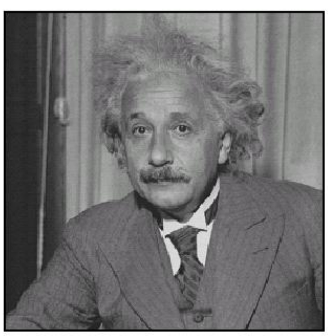
Original

$$* \left(\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \right) =$$

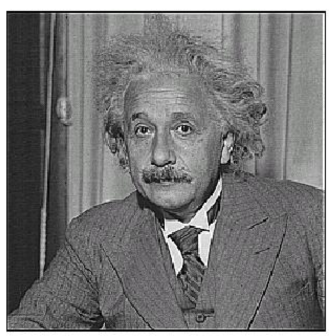
Sharpening filter
(accentuates edges)

Source: D. Lowe

Sharpening



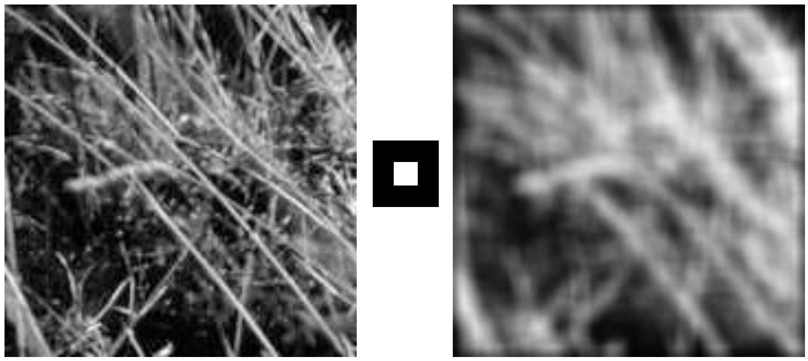
before



after

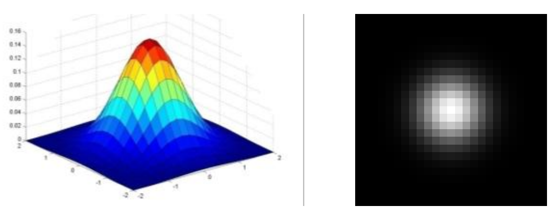
Source: D. Lowe

Smoothing with box filter revisited



Source: D. Forsyth

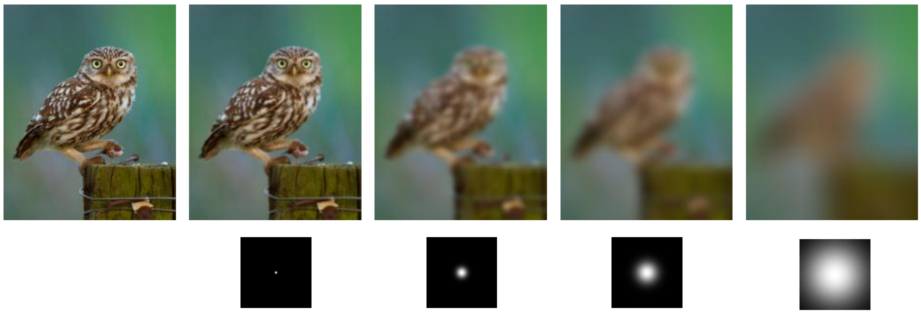
Gaussian Kernel



$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

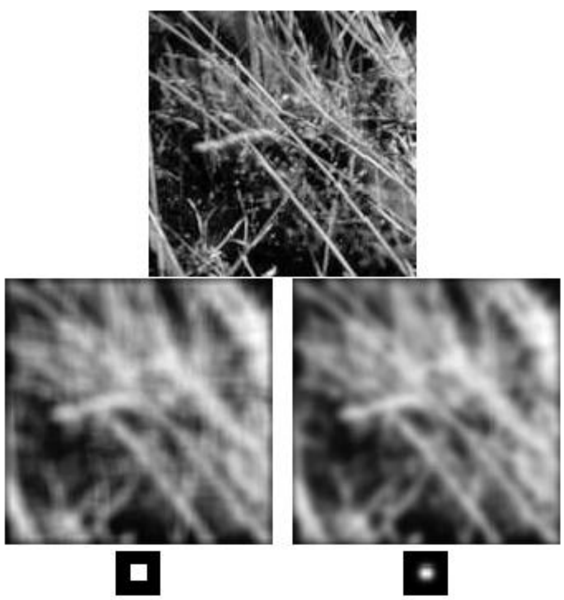
Source: C. Rasmussen

Gaussian filters



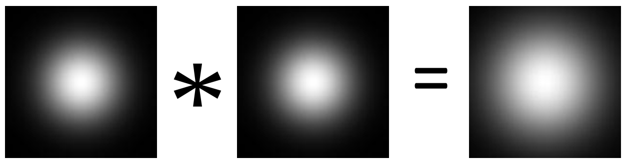
σ = 1 pixel | σ = 5 pixels | σ = 10 pixels | σ = 30 pixels

Mean vs. Gaussian filtering



Gaussian filter

- Removes "high-frequency" components from the image (low-pass filter)
- Convolution with self is another Gaussian



  – Convolving twice with Gaussian kernel of width σ = convolving once with kernel of width $\sigma\sqrt{2}$

Source: K. Grauman

Sharpening revisited

- What does blurring take away?

Let's add it back:

Source: S. Lazebnik

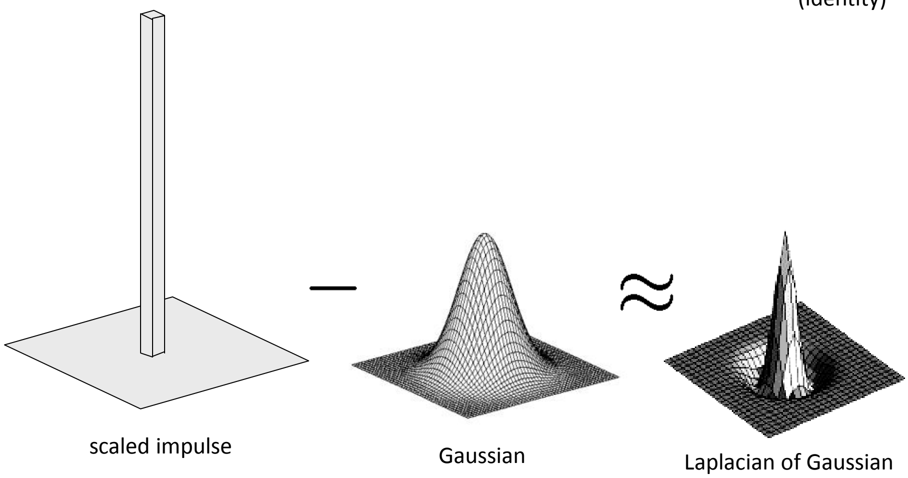
Sharpen filter

$$F + \alpha\,(F - \underbrace{F * H}_{\text{blurred image}})$$

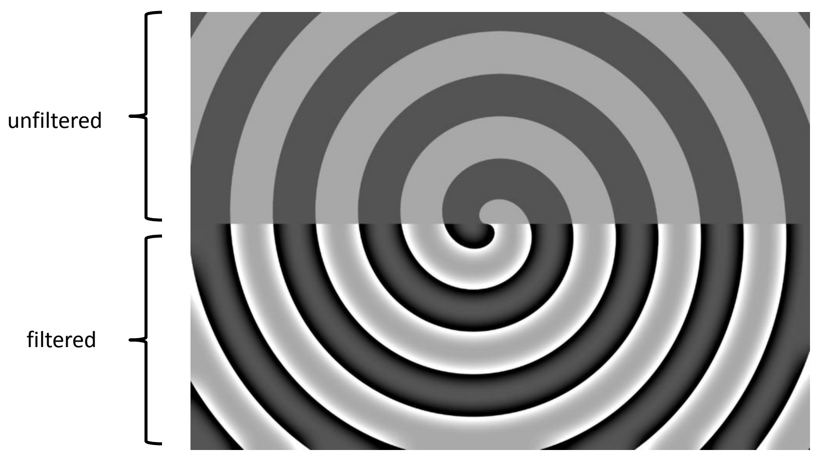
image

blurred image

unit impulse (identity)



Sharpen filter

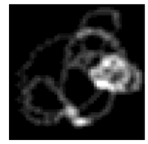


“Optical” Convolution

Camera shake

=

*



Source: Fergus, *et al.* “Removing Camera Shake from a Single Photograph”, SIGGRAPH 2006

Bokeh: Blur in out-of-focus regions of an image.

Source: http://lullaby.homepage.dk/diy-camera/bokeh.html

Questions?

- For next time:
  - Read Szeliski, Chapter 3.1-3.2