Matlab
(see Homework 1: Intro to Matlab)

Starting Matlab from Unix:
- matlab & OR
- matlab –nodisplay

Image representations in Matlab:
- Unsigned 8bit values (when first read)
  - Values in range [0, 255], 0 = black, 255 = white
- Double precision floating point
  - By convention, values in range [0.0, 1.0]; 0.0 = black; 1.0 = white
- Colour images have 3 values at each pixel: RGB
  - [0.0 0.0 0.0] = black; [1.0 1.0 1.0] = white; [1.0 0.0 0.0] = red
  - Sometimes accessed through a colour map (lookup table)

Linear Filters
(Reading: 7.1, 7.5-7.7)

- **Linear filtering:**
  - Form a new image whose pixels are a weighted sum of the original pixel values, using the same set of weights at each point

Correlation

I

10	11	10	0	0	1
9	10	11	1	0	1
10	9	10	0	2	1
11	10	9	10	9	11
9	10	11	9	99	11
10	9	9	11	10	10

F

1/9

1	1	1
1	1	1
1	1	1

O

X	X	X	X	X	X
X	10				X
X					X
X					X
X					X
X	X	X	X	X	X

1/9.(10x1 + 11x1 + 10x1 + 9x1 + 10x1 + 11x1 + 10x1 + 9x1 + 10x1) =
1/9.(90) = 10

Slide credit: Christopher Rasmussen

Convolution

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

- Same as correlation, but with kernel reversed
- Represent the linear weights as an image, ***F***
- ***F*** is called the **kernel**
- Center origin of the kernel F at each pixel location
- Multiply weights by corresponding pixels
- Set resulting value for each pixel

- Image, ***R***, resulting from convolution of ***F*** with image ***H***, where u,v range over kernel pixels (in 1D):

$$R_{ij} = \sum_{u,v} H_{i-u,j-v} F_{uv}$$

Warning: the textbook mixes up H and F

Correlation compared to Convolution

Definition: Correlation

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j)\, I\,(X+i, Y+j)$$

Definition: Convolution

$$I'(X,Y) = \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(i,j)\, I\,(X-i, Y-j)$$
$$= \sum_{j=-k}^{k} \sum_{i=-k}^{k} F(-i,-j)\, I\,(X+i, Y+j)$$

NOTE: If $F(X,Y) = F(-X,-Y)$ then correlation $\equiv$ convolution

Linear Filtering (warm-up slide)

Original

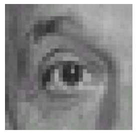
0	0	0
0	1	0
0	0	0

?

Linear Filtering (warm-up slide)

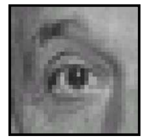
Original

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline\end{array}$$



Filtered
(no change)

Linear Filtering

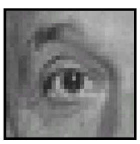


Original

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 1 \\ \hline 0 & 0 & 0 \\ \hline\end{array}$$

?

Linear Filtering



Original

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 1 \\ \hline 0 & 0 & 0 \\ \hline\end{array}$$

Shifted left
By 1 pixel

Linear Filtering

Original

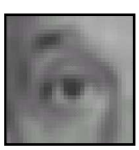
$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$

?

Linear Filtering

Original

$$\frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$



Blur (with a
box filter)

Linear Filtering

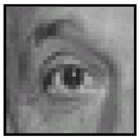
Original

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline\end{array} - \frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}$$

?

(Note that filter sums to 1)

Linear Filtering

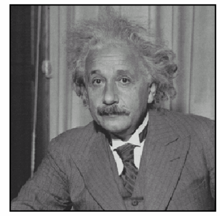


Original

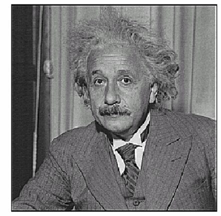
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sharpening filter
- Accentuates differences with local average
- Also known as Laplacian

Sharpening



before



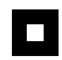
after

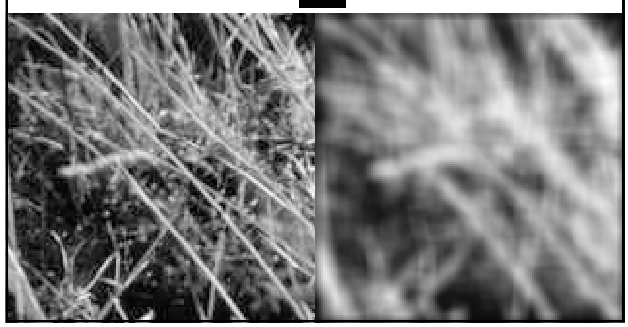
Average filter (box filter)

- Mask with positive entries, that sum to 1.
- Replaces each pixel with an average of its neighborhood.
- If all weights are equal, it is called a **box** filter.

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

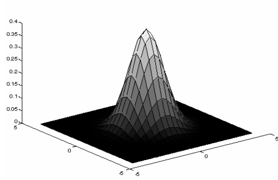
Example: Smoothing with a box filter





Smoothing with a Gaussian

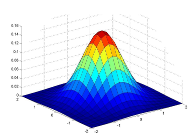
- Smoothing with a box actually doesn't compare at all well with a defocussed lens
- Most obvious difference is that a single point of light viewed in a defocussed lens looks like a fuzzy blob; but the box filter would give a little square.

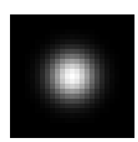


- A Gaussian gives a good model of a fuzzy blob
- It closely models many physical processes (the sum of many small effects)

Gaussian Kernel

- Idea: Weight contributions of neighboring pixels by nearness





0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

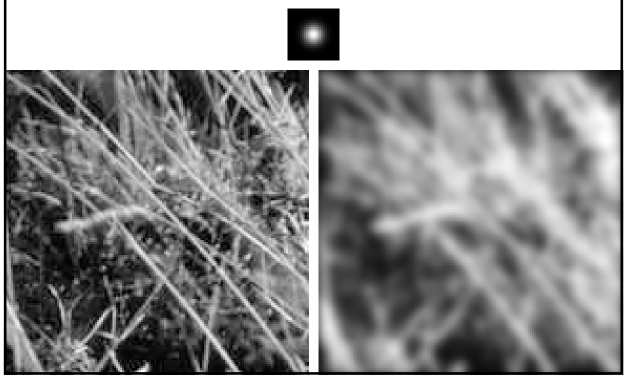
5 x 5, σ = 1

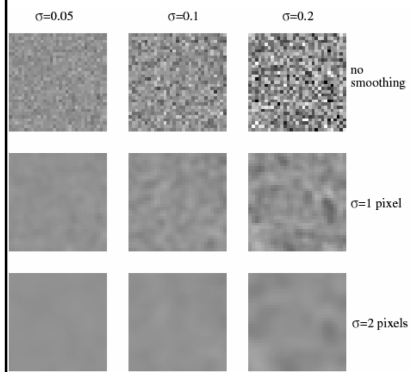
$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

- Constant factor at front makes volume sum to 1 (can be ignored, as we should re-normalize weights to sum to 1 in any case).

Slide credit: Christopher Rasmussen

Smoothing with a Gaussian





Smoothing reduces pixel noise:

Each row shows smoothing with Gaussians of different width; each column shows different amounts of Gaussian noise.

Efficient Implementation

- Both the BOX filter and the Gaussian filter are **separable** into two 1D convolutions:
  - First convolve each row with a 1D filter
  - Then convolve each column with a 1D filter.
  - (or vice-versa)

Separability of the Gaussian filter

For example, recall the 2D Gaussian

$$G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} \exp^{-\frac{x^2+y^2}{2\sigma^2}}$$
$$= \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{x^2}{2\sigma^2}}\right) \left(\frac{1}{\sqrt{2\pi}\sigma} \exp^{-\frac{y^2}{2\sigma^2}}\right)$$

The 2D Gaussian can be expressed as the product of two functions, one a function of *x* and the other a function of *y*

In this case, the two functions are the (identical) 1D Gaussian

Differentiation and convolution

- Recall, for 2D function, f(x,y):

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

- This is linear and shift invariant, so must be the result of a convolution.

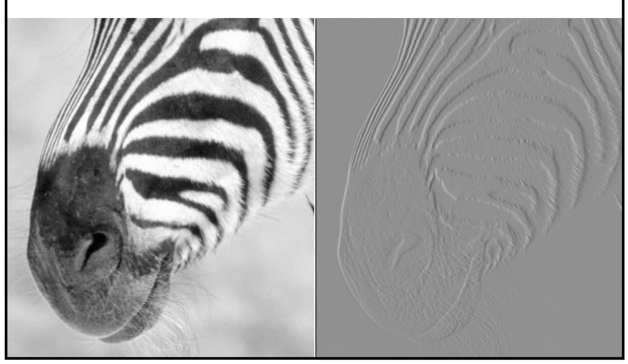
- We could approximate this as

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

(which is obviously a convolution)

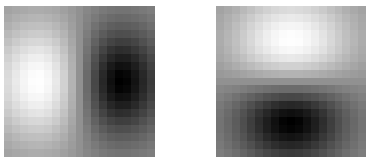
-1	**1**

Vertical gradients from finite differences



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 is a set of dot products
- Insight
  - filters look like the effects they are intended to find
  - filters find effects they look like



Normalized correlation

- Think of filters as a dot product of the filter vector with the image region
  - Now measure the angle between the vectors
    $a \cdot b = |a||b|\cos\theta$
  - Angle (similarity) between vectors can be measured by normalizing length of each vector to 1.
  - Normalized correlation: divide each correlation by square root of sum of squared values (length)

Application: Vision system for TV remote control

- uses template matching

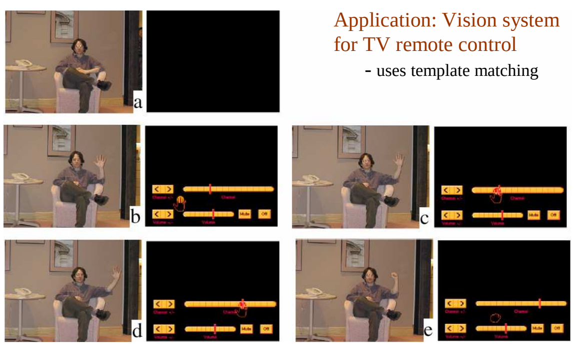


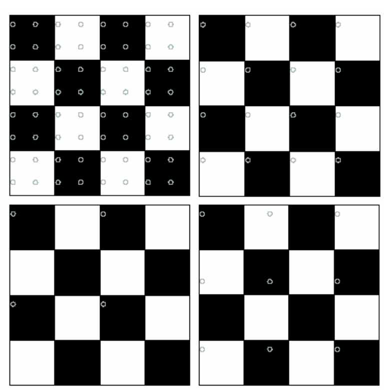
Figure from "Computer Vision for Interactive Computer Graphics," W.Freeman et al, IEEE Computer Graphics and Applications, 1998 copyright 1998, IEEE

We need scaled representations

- Find template matches at all scales
  - e.g., when finding hands or faces, we don't know what size they will be in a particular image
  - Template size is constant, but image size changes
- Efficient search for correspondence
  - look at coarse scales, then refine with finer scales
  - much less cost, but may miss best match
- Examining all levels of detail
  - Find edges with different amounts of blur
  - Find textures with different spatial frequencies (levels of detail)

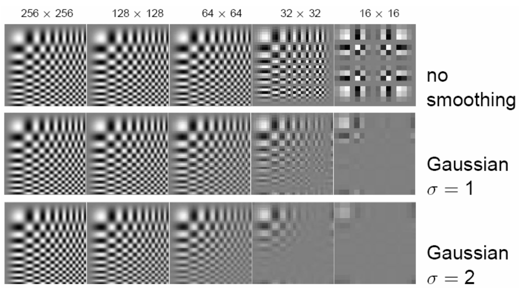
Aliasing

- We can't shrink an image by taking every second pixel
- If we do, characteristic errors appear
  - Examples shown in next few slides
  - Typically, small phenomena look bigger; fast phenomena can look slower
  - Common examples
    - Checkerboard patterns misrepresented in video games
    - Striped shirts look funny on colour television
    - Wagon wheels rolling the wrong way in movies



Resample the checkerboard by taking one sample at each circle. In the case of the top left board, new representation is reasonable. Top right also yields a reasonable representation. Bottom left is all black (dubious) and bottom right has checks that are too big.

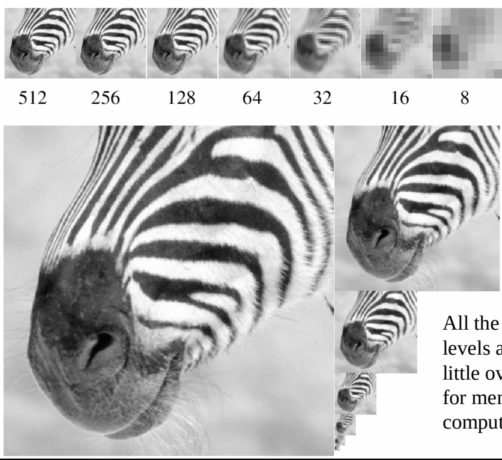
Resampling with prior smoothing



Forsyth & Ponce Figures 7.12–7.14 (top rows)

The Gaussian pyramid

- Create each level from previous one:
  - smooth and sample
- Smooth with Gaussians, in part because
  - a Gaussian*Gaussian = another Gaussian
  - G(x) * G(y) = G(sqrt(x^{2} + y^{2}))
- Gaussians are low pass filters, so the representation is redundant once smoothing has been performed
  - There is no need to store smoothed images at the full original resolution



All the extra levels add very little overhead for memory or computation!

Summary of Linear Filters

- **Linear filtering:**
  - Form a new image whose pixels are a weighted sum of original pixel values
- **Properties**
  - Output is a **shift-invariant** function of the input (same at each image location)

Examples:
- Smoothing with a box filter
- Smoothing with a Gaussian
- Finding a derivative
- Searching for a template

Pyramid representations
- Important for describing and searching an image at all scales