CSci 4968 and 6270
Computational Vision,
Fall Semester, 2011 Lectures 2&3, Image Processing

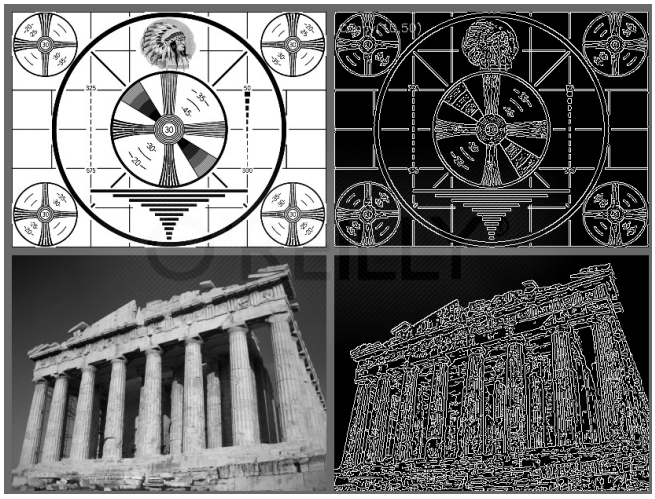
1 Introduction

How Do We Start Working with Images?

- Corners, boundaries, homogeneous regions, textures?
- How do we approach this from the point of view of a computation?

What We Want Are (Low-Level) "Features"

- Edges, perhaps better called "edgel elements" or edgels, are locations of maximal "significant" changes in intensity in one spatial direction.
- Corners are locations of maximal significant changes in intensity in two spatial directions.
- "Interest points" are distinctive, repeatably-detectable image locations.
  - In practice, techniques for corner detection and for interest point detection are largely the same.
- See Figure 1.



http://www.oreilly.de

http://ivt.sourceforge.net/demos/HarrisDemo.png

Figure 1: Example results of edge detection (left) and corner detection (right).

Basis of Extracting These Features

- Each feature is "local" to a small region of the image.

- Operations are applied to each pixel using its surrounding neighborhood of pixels, looking for "distinguishing" properties.
  - These distinguishing properties are generally based on some form of differentiation, implemented in the discrete domain.
- First, however, we must prepare or preprocess the image.
- Afterwards we must analyze and interpret the results of applying these operations to extract the features.

Overview of These Lectures

- Images
- Pixel-level operations
- Neighborhood operations
  - Gaussian smoothing
  - Differentiation
- Fourier transformations (notes only)
- Pyramids
- Binary operations and mathematical morphology
- Non-linear filters
- Concluding example of (primitive) change detection

2 Images

Images

- Photographic images of 12 mega-pixels (4K by 3K) are common — huge amount of data
- Often we will work with much smaller images, such as 640x480, during our computations
- Video resolution is typically anywhere from 320x240 up to 1024x768, with frame-rates of 30 images per second or more.
- Most camera chips have alternating patterns of red, green and blue sensors, with twice as many green as either red or blue, laid out in a "Bayar" pattern
- In order to form image pixels, RGB values are interpolated.
- Image cooordinate systems have the origin in the upper left corner with x across and y down.

Many Different Color Spaces

- Cameras and displays usually work with RGB (red-green-blue) values, each recorded in the range $[0, 255]$.
- Printers often work with CMYK (cyan, magenta, yellow and key (black)).
- RGB is also not "perceptually uniform", meaning that adding a fixed offset $\boldsymbol{\Delta c}$ to a given (r, g, b) vector has different meaning to the human eye, depending on the value of (r, g, b).
  - Aside: this is a first example of thinking of pixel values in terms of vectors and considering vector addition.
- Other color spaces (see `http://www.easyrgb.com` for conversions):
  - CIE and its variants, including L*a*b, which are designed for perceptual uniformity
  - XYZ
  - HSV (hue, saturation, value)

Color Has Played a Surprisingly Small Role in Computer Vision

- Image color measurements of the same surface differ substantially under different light sources, viewpoints, and cameras. Think back to our examples from the introductory lecture.
  - Color constancy is hard!
  - Despite this, humans tend to be good at it. (This poses one challenge to taking good quality photos!)
- Historically, color has played a surprisingly small role in computer vision research and applications.
- One important application: finding skin and faces in images (start with Jones and Rehg, *IJCV* 2002)
- Growing use in scene and context recognition.
- We will focus primarily on grayscale images

3 Pixel Operations

Converting to Grayscale

One of the first steps is converting from RGB to grayscale.

- A "perfect" conversion is not possible because (r, g, b) depends on the properties of the camera.
- Why is $i = \|(r, g, b)\|$ a problem?

- One commonly-used linear combination (see `http://en.wikipedia.org/wiki/Grayscale`) is
$$i = 0.3r + 0.59g + 0.11b.$$
- The Matlab `rgb2gray` command from the image processing toolbox uses
$$i = 0.2989r + 0.5870g + 0.1140b.$$
- Why does green play such a strong role?
- More sophisticated methods are often used in software such as Photoshop.

Figure 2: Color-to-gray scale conversion example.

Computing Histograms

This is a straightforward operation, described here mathematically

- Let $f(i, j)$ be the image, with range $[0, 255]$.
- Then the histogram is
$$h(k) = \text{card}\{(i, j)|f(i, j) = k\},$$
with the domain of the function h being $[0, 255]$.
- What does the histogram tell us about the image?

PDFs and CDFs from Histograms

- If there are N pixels in the image, the histogram is converted to a (discrete) "probability density function" as
$$\text{pdf}(k) = h(k)/N,$$
with the range of function pdf now being $[0, 1]$.

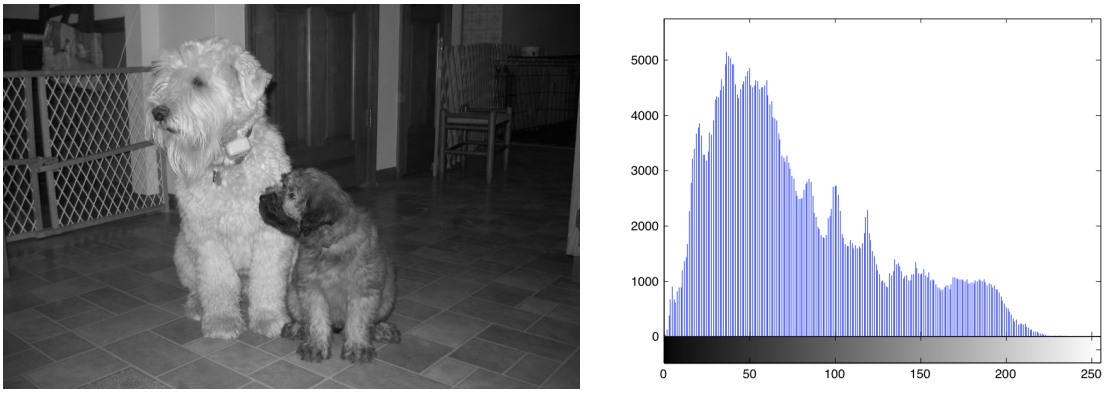


Figure 3: Histogram of the "dogs" gray-scale image.

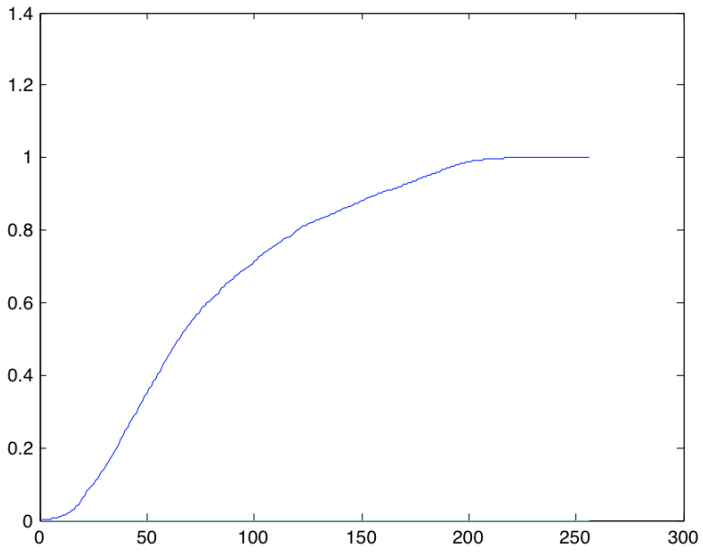


Figure 4: Plot of the cumulative density function of the dogs image

- We get the "cumulative distribution", cdf, as

$$\mathrm{cdf}(k) = \sum_{k_0=0}^{k} p(k_0),$$

  or recursively as $\mathrm{cdf}(0) = p(0)$ and

$$\mathrm{cdf}(k) = \mathrm{cdf}(k-1) + \mathrm{pdf}(k), \qquad k > 0.$$

Intensity Mapping Based on Histograms

The pdf and the cumulative distributions can be used in a number of different ways:

- Linear ramp: We can set a min and max intensity based on the cumulative distribution, map these to 0 and 255, and map the remaining intensities accordingly.

Before After

Figure 5: Before and after histogram equalization.

- Histogram equalization: We can adjust the intensities so that the resulting image has a (approximately) constant histogram.
- More sophisticated processes are possible.

We will derive equations and look at examples in class.

4 Linear Neighborhood Operations and Convolution

Motivation

We want our neighborhood operations to accomplish two tasks as the basis for our feature extraction:

- We want to smooth the image to make noise and minor features less prominent.
- We want to locate potential boundaries:
  - Locations where the intensity is changing most rapidly.
  - Thinking in terms of calculus, these are locations of largest (in magnitude) first derivatives or zero second derivatives.

Noise Suppression as (Weighted) Averaging

- If all the pixels surrounding a given pixel are from the same surface, then we can reduce the noise and suppress detail by replacing the pixel with the average of its intensity and that of its neighors.
  - For example, we could center a 3x3 or a 5x5 neighbhorhood at each pixel.
- Conceptually, this is done simultaneously at each pixel in the image, requiring a second image on which to record the results.
- The average could be weighted, so that pixels further away from the center pixel are given less influence.

- See Figure 4

5x5

9x9

Figure 6: Examples of convolving/correlating with a box filter

Averaging as Correlation

Mathematically, we describe this as follows:

- Let f be the original image, and g be the output image.
- Let h be the image or function of weights.
- Then we write:
$$g(i,j) = \sum_{k,l} f(i+k, j+h)h(k,l),$$
or more compactly as
$$g = f \otimes h.$$
- Technically, this is called the *correlation* of f with h.
- In the case of local averaging, h is typically defined over a small domain, such as $[-2,2] \times [-2,2]$ for averaging over a 5×5 neighborhood.

Correlation and Convolution Are Closely Tied

- Convolution is defined as
$$g(i,j) = \sum_{k,l} f(i-k, j-l)h(k,l)$$
or $g = f * h$
- Here h is called the "impulse response" function because in the special case that f is the (discrete) impulse function — $f(0,0) = 1$ and $f(i,j) = 0$ when $i \neq 0$ or $j \neq 0$ — $f * h = h$.

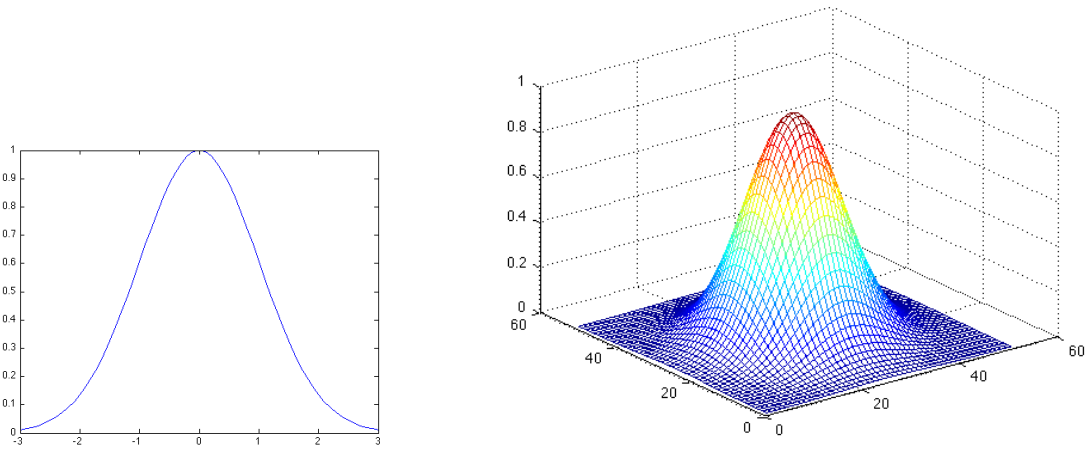
- For correlation, when f is the impulse function, $f \otimes h = -h$.
- Notice that when h is an even function — meaning that $h(k,l) = h(-k,-l)$ — correlation and convolution are equivalent.
- k is more commonly called the "kernel" or even the "operator".

Weight Dropoff

- The box filter produces an average over a neighborhood.
- Instead we might want a weighted average, with more weight given to the center pixel and less to the surrounding pixels.
- The question becomes, what is the appropriate weighting function?
- The answer, for several reasons, at least as long as we are considering linear filtering, is the *Gaussian function*.

Gaussian Weighting — Continuous Form

- Continuous form in 1d
$$g(x;\sigma) = \exp(-0.5x^2/\sigma^2).$$
- Continuous form in 2d
$$g(x,y;\sigma) = \exp\big(-\frac{x^2+y^2}{2\sigma^2}\big) = \exp\big(-\frac{x^2}{2\sigma^2}\big)\exp\big(-\frac{y^2}{2\sigma^2}\big).$$
- See Figure 4



1d gaussian 2d gaussian

Figure 7: Plot of the continuous Gaussian function in one-dimension and in two.

Gaussian Weighting — Discrete Form

- Gaussian is non-zero over an infinite domain!
- Practically, however, it can be treated as 0 for values of x at and beyond a given multiple of σ, usually 2.5 or 3. In particular, at $x = 3\sigma$ $g(x;\sigma) \approx 0.011$, at $x = 2.5\sigma$ $g(x;\sigma) \approx 0.044$, while at $x = 0$ $g(0;\sigma) = 1$.
- To form the discrete Gaussian, we sample and normalize so that the sum of the weights is 1. You can do this either using integer values or floating point values of the kernel.
- For example, here is an integer-valued discretized form in 1d with $\sigma = 1.5$ and a cut-off of 2.5σ

	-3	-2	-1	0	1	2	3
1/27	1	3	6	7	6	3	1

- Similarly, with $\sigma = 1.5$ and a cut-off of 3σ we have

	-4	-3	-2	-1	0	1	2	3	4
1/131	1	5	14	28	35	28	14	5	1

- In both of these examples, I created the integer values shown in the kernel by
  1. normalizing each entry by the value of $\exp(-0.5 * w^2/\sigma^2)$
  2. rounding the result
  3. summing over the resulting kernel values to obtain the final normalizing term (1/27 in the first example and 1/131 in the second).
- In practice, other less-exact approximations are sometimes used. For example, you might cut-off at $\sigma = 2$ as well, especially for larger values of σ.
- Perhaps more simply, you may use float-point or double-precision weights throughout, but be sure that you normalize so that they sum to 1.0.

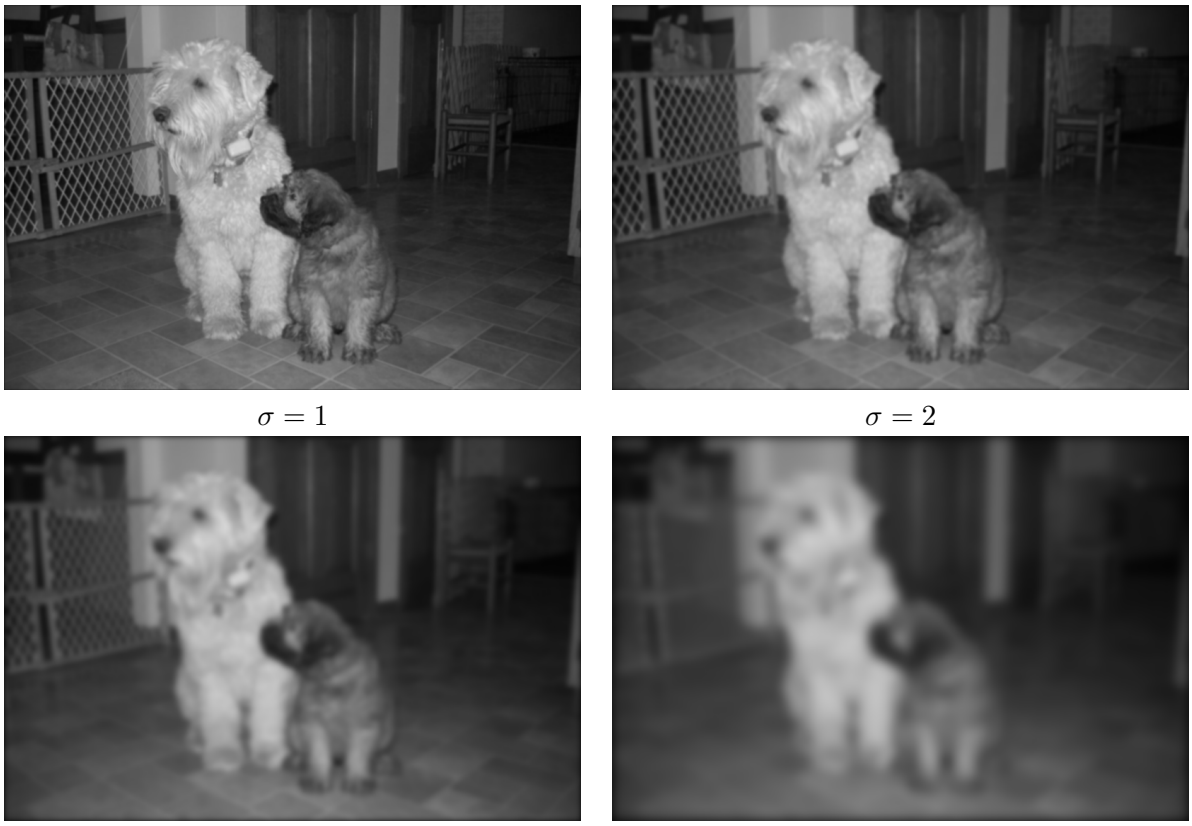
Increasing Sigma Increases the Smoothing

- The $\sigma = 1.5$ kernel has width $2w + 1 = 7$ (or 9 if a 3σ cut-off is used), whereas the $\sigma = 3$ kernel has width $2w + 1 = 15$ (or 19 if a 3σ cut-off is used).
- The practical effect is increasingly smoothed and blurred images as σ increases
- Examples are shown in Figure 4.

Discrete Partial Differentiation as Convolution

- Recall the limit definition of the derivative:

$$f'(x) = \lim_{\Delta x \to 0} \frac{f(x + \Delta(x)) - f(x)}{\Delta x}.$$



$\sigma = 1$ $\sigma = 2$

$\sigma = 4$ $\sigma = 8$

Figure 8: Examples of convolution with a Gaussian kernel at different scales.

- We will use this in class to derive a discrete approximation to the derivative as

$$f'(x) = \frac{f(x+1) - f(x-1)}{2}.$$

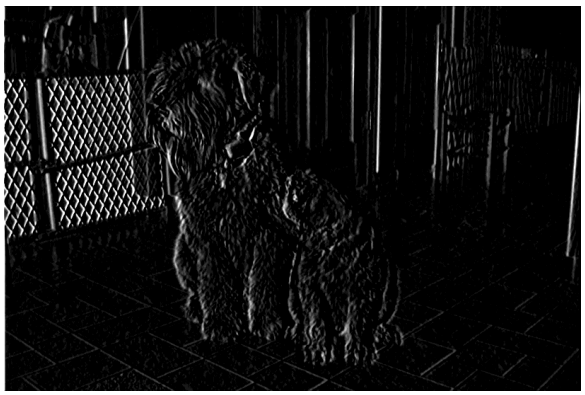
- Aside: the measurements here are in "pixel" units.
- This produces the following simple correlation kernel for differentiation:

-1	0	1
$-1/2$	0	$1/2$

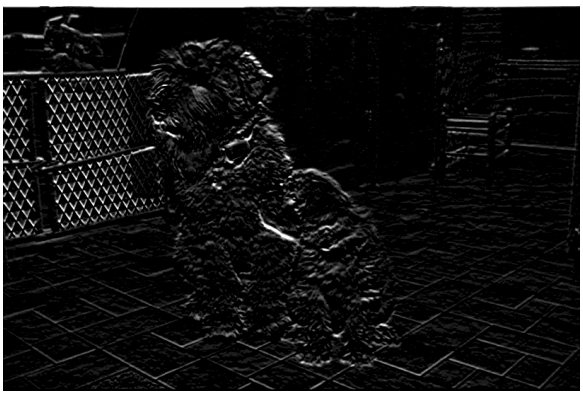
Partial Derivatives

In two dimensions,

- We obtain $\partial f/\partial x$ with the above kernel.
- We obtain $\partial f/\partial y$ by rotating the kernel 90° clockwise.



$\partial f/\partial x$



$\partial f/\partial y$

Figure 9: Examples of applying partial derivatives in the x and y directions.

Looking Ahead to Edge and Corner Detection

We will look at:

- Combining partial derivatives to compute gradient vectors
- Second derivative operators: Laplacian and Hessian
- Developing methods to "extract" edges and corners based on the results of smoothing and differentiation.

Implementation: Boundary Effects Can Be Handled in Several Ways

- Since the image domain is not infinite, there will be image locations (i, j) where computation of the convolution requires locations $(i + k, j + l)$ that are outside the image domain.
- This can be handled in several ways:
  - Assign the convolution result to 0 at these locations.
  - Mirror the values. For example, at the upper and left boundaries of the image $f(r, c) = f(-r, -c)$.
  - Assign $f(r, c) = 0$ to locations (r, c) outside the image domain for the purposes of computing the convolution.
  - Wrap-around, treating the image as a torus. If the image has m rows and n columns, then when $r < 0$ and $c < 0$, $f(r, c) = f(m + r, n + c)$.
- The choice between these depends on the context, as we will soon see.

Implementation: Separating 2D Convolutions Into Two 1D Convolutions

- We can show that convolution is commutative, i.e.

$$(f * g) * h = f * (g * h).$$

  Think of f as the original image and g and h as two successive convolution kernels.
- Implication: if we have a "large" kernel k and we can break it up into two (much) smaller kernels g and h, then rather than applying k to the image, we can apply g and h in succession.
- The most important case of this occurs when g is "vertical" — all its non-zero values are in a single column — and h is horizontal — all its non-zero values are in a single row.
- Both the box kernel and the Gaussian kernel are separable.

Warning: when applying two consecutive 1D kernels instead of a 2D kernel, the intermediate results should be computed and stored in a higher precision than unsigned char (byte).

5 Very Brief Introduction to Fourier Transformations

- Understand the "frequencies" or the rate of variation in an image.
- Fundamental tool of signal processing and image processing.
- For us, the most important use is to analyze kernels.
- Works in the complex domain
- Mathematically, it is a change of basis (functions).

Discrete, 1d Mathematical Form

- Let $j \equiv \sqrt{-1}$, k be the "frequency" variable, let H be the Fourier transformation of h — i.e. $H = \mathcal{F}(h)$ — and let $h(x)$ be the kernel. Then,

$$\begin{aligned} H(k) &= \frac{1}{N}\sum_{x=0}^{N} h(x) e^{\frac{-j2\pi kx}{N}} \\ &= \frac{1}{N}\sum_{x=0}^{N} h(x)\big(\cos\big(\frac{2\pi kx}{N}\big) + j\sin\big(\frac{2\pi kx}{N}\big)\big) \end{aligned}$$

- The angular frequency is actually $k\frac{2\pi}{N}$.
- We can also compute an inverse Fourier transformation.

Understanding This With Some Examples

Look at different values of k:

- What happens when $k = 0$, $k = 1$, etc?
- What happens if we shift the image?

Important Properties

- Convolution in the "spatial domain" becomes multiplication in the Fourier domain.
- Combining with the inverse Fourier Transformation we get

$$f * h = \mathcal{F}^{-1}(\mathcal{F}(f)\mathcal{F}(h)).$$

Examples

- The Fourier transform of the Box filter — square with constant values — is the sinc function.
- The Fourier transform of the Gaussian is another Gaussian, proportional to

$$\exp\big(-\frac{\omega^2}{2(1/\sigma^2)}\big).$$

Implication

- Multiplying by the Gaussian in the Fourier domain attenuates higher frequencies — noise **and** sharp changes.
- This explains mathematically the averaging and blurring effects we already know about.
- The limited "support" in both the spatial and frequency domains is why the Gaussian is a popular smoothing filter.
  - It is also the only isotropic, separable 2d filter.
- The box filter, whose Fourier transform is the sinc function, is separable but not isotropic.

6 Image Pyramids

Forming a Pyramid

- Smoothing with a Gaussian removes higher frequencies
- After smoothing with $\sigma = 2$, we no longer need to represent the image with as many samples.
- Hence, we downsample by removing every other row and every other column.
- Repeating this recursively, we create a "pyramid" of images.
- Processing is faster on the smaller ("coarser") images at higher levels of the pyramid.
- More sophisticated pyramids may be created used wavelets
- The Matlab function `impyramid` may be used to create a pyramid based on an approximation to the Gaussian.

Aide: Sum-of-Squared Differences Matching

- Consider the images in Figure 6, with the pillow being the only common feature. The images are 1280x960.
- Suppose we wanted to find the region of the image (a) that best matched the pillow region from (b).
- Simply idea:
  - Place the "pillow" from (b) at each possible location in (a)
  - Compute the intensity differences at each overlapping pixel, square them and sum them.
  - Choose the location that minimizes this "sum of squared differences"
- What might be some reasons why this simple idea may not work?
- Scale is one of them (not the only one) — the pillow is 68 pixels high in the (a) image and 230 in the (b) image.
- After building a pyramid for (b), in the second level of the pyramid the pillow is about 62 pixels high.
  - Back to the figure(c) shows (b) after Gaussian smoothing with $\sigma = 4$ and (d) shows the result after downsampling by a factor of 4. The pillow in the downsampled image does not look blurred and looks more like the pillow in (a).
- The pillow is more likely to match correctly between level 2 of the pyramid for image (b) and level 0 of the pyramid for image (a) than between the original images.

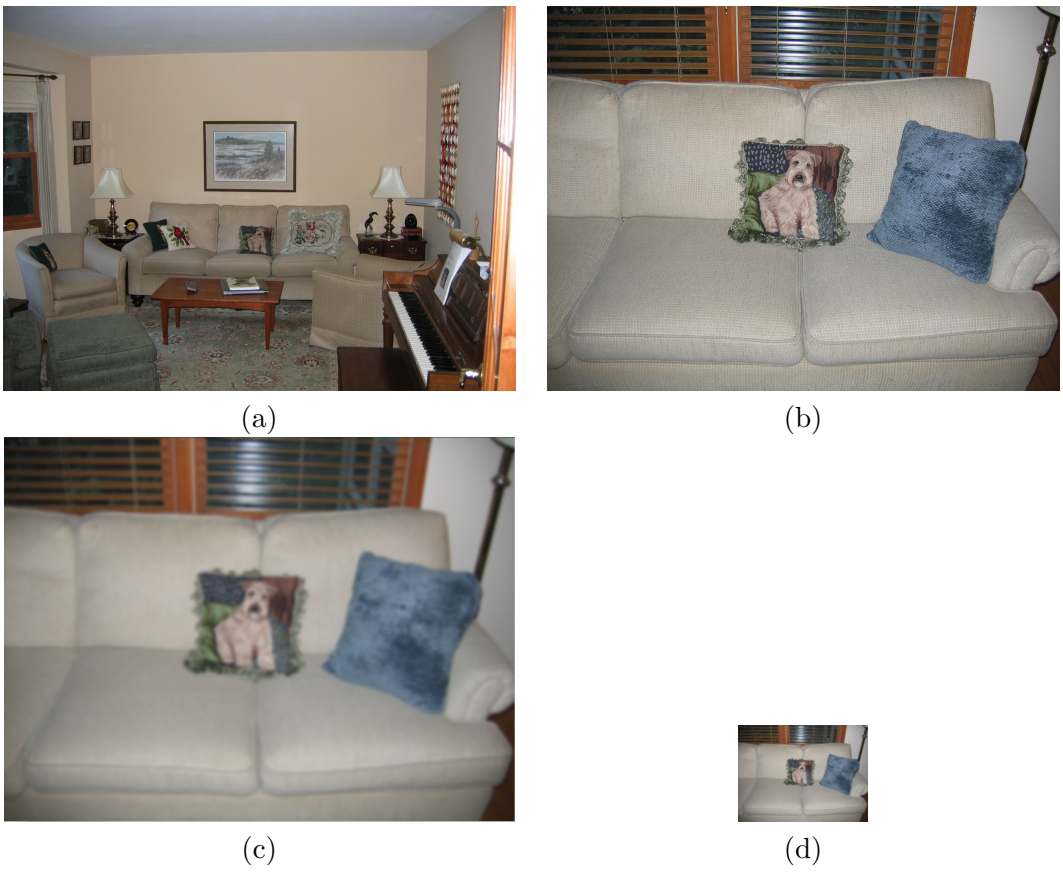


Figure 10: Two images, (a) and (b), with just the Wheaten Terrier pillow in common. The image regions containing the pillow are dramatically different in scale. Panel (c) shows (b) after smoothing with $\sigma = 4$ and (d) shows (c) after downsampling by a factor of four in each dimension.

7 Binary Image Analysis and Mathematical Morphology

Motivation for Binary Image Analysis and Mathematical Morphology

Morphology:

- Operations on binary image
- First example of non-linear operators
- Concerned with "structure" of image regions

Looking again, in Lecture 4 we will discuss non-linear operators that are more closely analogous to convolution filtering.

Aside on Thresholding

- Given an image f and a threshold θ, compute a new image:

$$b(x,y) = \begin{cases} 1 & f(x,y) \geq \theta \\ 0 & f(x,y) < \theta \end{cases}$$

- Ideally, separate "foreground" (pixels of interest) from "background" (pixels that can be safely ignored).
- Ordinary photographs usually are ill-suited to thresholding.
- Special-purpose images, such as CT scans and document images, often are.
- Images that result from applying transformations to one or more original (input) images may be as well.
- Matlab functions to examine: `graythresh` and `im2bw`.
- Histograms and pdfs can be used to help establish thresholds by looking for the bottom of the valley between two distributions.

Set Theoretic Approach

- We will think of our binary image as a set of locations:

$$B = \{(i,j) \mid b(i,j) = 1\}.$$

- We will apply a structuring element K to an image, which may also be thought of as a set of locations.
  - K plays a similar role in morphology that the convolution kernel plays in linear operations.
- Notationally, if $k \in K$ and $k = (x,y)$, then we write

$$B_k = \{(i+x, j+y) | (i,j) \in B\}.$$

In other words B_k is a shift of the set B by the vector (x,y).

Dilation and Erosion

The two fundamental operations of morphology:

- Dilation:
$$B \oplus K = \bigcup_{k \in K} B_k.$$
In other words, translate B by each element of K and take the union.
- Erosion:
$$B \ominus K = \bigcap_{k \in K} B_{-k}.$$
In other words, translate B by each $-k$ and take the intersection.
- We will look at examples in class.

Conceptual View

Assuming $(0, 0) \in K$.

- Dilation: Place the center of the structuring element on an image location (where there is a 1). All image locations covered by K are in the result of the dilation.
- Erosion: Same intuition as dilation, but with the mirror image of K and reversing the roles of 0 and 1 in the image.

Opening and Closing

- Opening: erosion followed by dilation with the same structuring element:
$$B \bullet K = B \ominus K \oplus K.$$
- Closing: dilation followed by erosion with the same structuring element:
$$B \circ K = B \oplus K \ominus K.$$
- Intuitions:
  - Opening: Consider any location where K may be placed such that K is entirely "inside" the image (the binary 1 region). All locations covered by K will "survive" the opening operation.
  - Closing: Consider any location where K may be placed such that K is entirely "outside" the image. All such locations will remain 0.
  - "Open holes" and "close gaps".

Common Structuring Elements

- Disk of radius r
- Square of width w (usually odd)
- Oriented rectangles

Examples

We will play with some examples in class. Relevant Matlab functions include:

- `strel` — create a structuring element
- `imdilate`
- `imerode`
- `imclose`
- `imopen`

Connected Components Labeling

- Once we have binary image, we can gather pixels into regions and analyze the regions
- This is connected-component labeling
- Question about connectivity: 4-connected or 8-connected?
  - We'll investigate in class.
- Think of the image as a graph!
  - Pixels that are "1" are the vertices
  - Edges formed to neighboring "1" pixels based on either 4-connectedness or 8-connectedness
- Standard breadth-first search or depth-first search may be used to extract connected components
- Faster, more specialized algorithms may be applied
- First example of thinking of the image as a graph.

Region Analysis

- Once we have the sets of connected components, we can analyze each component to discover its area, center and second moments.
- Area and center are easy
- Second moments giving the major and minor axes of the region, are found from the eigenvectors of the 2x2 scatter matrix.

Application: Identifying Regions of Change

Problem:

> Given an approximately stationary camera monitoring an approximately stationary scene, find the regions of change when something or someone is moving.

Solution

- Align images to remove small camera motions
- Subtract images, pixel-by-pixel; compute the absolute value
- Apply a threshold
- Binary opening and binary closing to clean up
- Connected components and region analysis.

Matlab functions used:

- `imread`, `rgb2gray`, `imabsdiff`, `imtool`, `im2bw`, `strel`, `imopen`, `imclose`, `bwlabel`, `label2rgb`

8 Non-Linear Smoothing: Median and Bilateral Filtering

Motivation for Non-linear Smoothing

- Smoothing with a Gaussian blurs edges in addition to removing noise.
- Looking at an edge profile suggests several ways to help address this problem.
- In linear operations, only the pixels' location relative to each other are used in determining contributions — think of Gaussain weighting.
  - Stated more mathematically, this is a domain-only based determination of contribution.

- The idea of non-linear smoothing is to use the pixels' intensities — working in the "range" of the image function — to help decide how to smooth intensities.
- We pay a performance penalty for this.

Simplest Idea: Median Filtering

- At each pixel, (i, j), let $N(i, j)$ be the neighborhood (such as a 5x5 region) of pixels.
- Then, the median-filtered image is

$$g(i, j) = \text{median}\{f(k, l) \mid (k, l) \in N(i, J).$$

- Advantages:
  - Preserves ideal step edges perfectly.
- Disadvantage:
  - Computational cost
  - Doesn't smooth very much — best for "salt and pepper" noise, which is rare these days.
  - A variety of structures is lost. (We'll think about these in class.)

Bilateral Filtering

- Weighted averaging where the weights depend on both the difference in position AND the difference in gray scale.
- Standard weighted average formula

$$g(i, j) = \frac{\sum_{(k,l)\in\mathcal{N}(i,j)} f(k, l)w(i, j; k, l)}{\sum_{(k,l)\in\mathcal{N}(i,j)} w(i, j; k, l)}$$

  where $\mathcal{N}(i, j)$ is a neighborhood.
- Weight function is composed of a *domain kernel* and *range kernel*:

$$w(i, j; k, l) = d(i, j; k, l)r(i, j; k, l),$$

  with

$$d(i, j; k, l) = \exp\left(-\frac{(i - k)^2 + (j - l)^2}{2\sigma_d^2}\right)$$
$$r(i, j; k, l) = \exp\left(-\frac{\|f(i, j) - f(k, l)\|^2}{2\sigma_r^2}\right).$$

- The vector distance is used for image functions in case the pixels are color valued.

9 Summary of Low-Level Image Processing

- Histograms tell us about the spread of intensities and are useful for establishing thresholds.
- Smoothing and differentiation are achieved through convolution
- The Gaussian is the most widely used smoothing operator.
- Fourier transforms help to analyze the frequencyies of an image or a kernel.
- Image pyramids allow us to efficiently examine and match at different scales.
- Mathematical morphology and region statistics are used to analyze binary images.
- Image differencing, thresholding and morphology can be combined to form a motion detection and change analysis system.
- Non-linear filters are useful for preserving sharp edges while smoothing out details.