Image analysis

CS/CME/BIOPHYS/BMI 279
Fall 2015
Ron Dror

A two-dimensional image can be described as a function of two variables f(x,y). For a grayscale image, the value of f(x,y) specifies the brightness of the image pixel at position (x,y). (For a color image, the function specifies a red, green, and blue value for each position (x,y), but in this class, we'll be concerned primarily with grayscale images.)

If an image is obtained by a real-world instrument such as a digital camera or microscope, then the image is never perfect; some random perturbations are introduced, due both to imperfections in the instrument and to physical laws that limit measurement accuracy. We refer to these random perturbations as "noise."

In the following sections, we will go through some basic techniques to reduce noise in images, and briefly introduce edge-finding and principal component analysis as an example of extracting information from images.

Noise Reduction

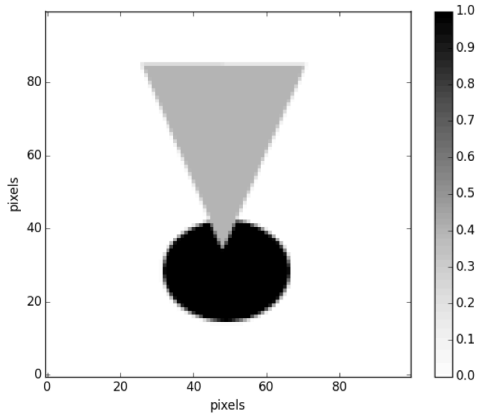
Noises in images may have different origins and therefore different characteristics. For instance, in Gaussian noise, a normally distributed random value is added to each pixel (fig. 1b). On the other hand, salt-and-pepper noise describes a situation where random pixels get replaced by extremely dark or bright values (fig. 1c).

In an uncorrupted image, the values of neighboring pixels are expected to be correlated (because an object or surface generally spans multiple neighboring pixels). Noise in neighboring pixels, on the other hand, is often uncorrelated (this is the case for the Gaussian noise and salt-and-pepper noise added in fig. 1). The most basic way of reducing the noise in an image is thus by "averaging" neighboring pixels. Such methods are often referred to as "filtering." Commonly used filters include mean filter (a.k.a. uniform filter), which replaces the value of a pixel by the mean value of an area centered at the pixel; Gaussian filter, which is similar to a mean filter but weighted in favor of pixels closer to the center; and median filter, which replaces the value of a pixel by the median of nearby pixels.

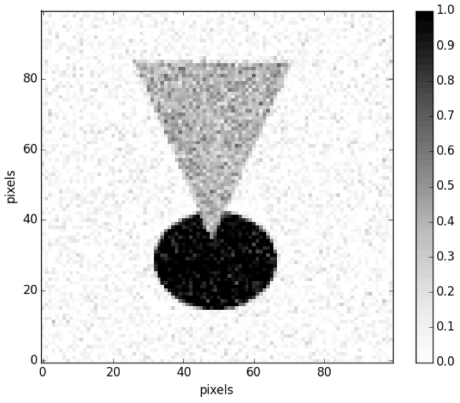
Fig. 2 demonstrates the effects of these filters on image noise. The Gaussian noise may be reduced, but not completely removed, by any of the filters. A Gaussian filter typically yields the smoothest image, whereas a similarly sized mean or median filter typically leaves blocky traces. The salt-and-pepper noise, on the other hand, can be completely removed by a median filter (if it affects only a small fraction of the pixels in any given region), but not by a mean or Gaussian filter.

Mathematically, applying a mean filter or Gaussian filter is equivalent to convolving the image with a window function; that window function is uniform over a small area in the case of a mean filter, or a Gaussian function in the case of a Gaussian filter. In order not to change the overall brightness of the image, the window functions are typically normalized such that their integrals evaluate to 1.

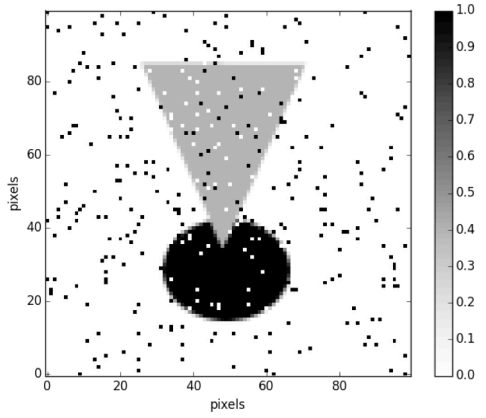
The Convolution Theorem states that the convolution between two functions in the real space is equivalent to their multiplication in Fourier space. Fig. 3 shows the window functions for a mean and Gaussian filter, along with their respective Fourier transforms. In Fourier space, both these functions have greater magnitudes (absolute values) at low frequencies and smaller magnitudes at high frequencies. The mean and Gaussian filters thus reduce the magnitudes of high-frequency components of the image while preserving ("passing") the low-frequency components. They are thus referred to as low-pass filters. (A median filter is not a convolution, so relating it quantitatively to the Fourier components is difficult.)



(a)



(b)



(c)

Fig.1: (a) a noiseless image with 100x100 pixels, on top of which (b) Gaussian noise (c) salt-and-pepper noise were added

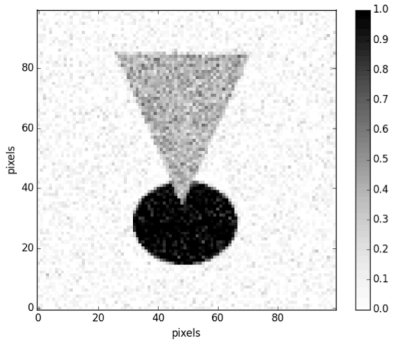
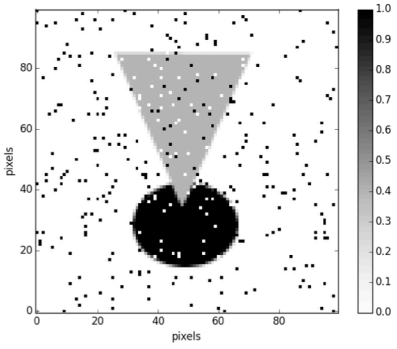
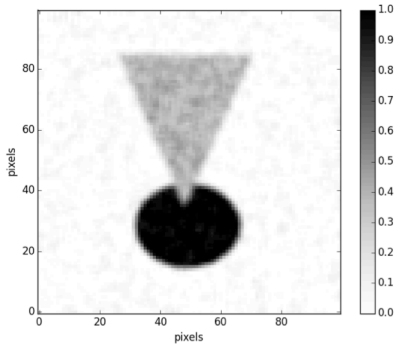
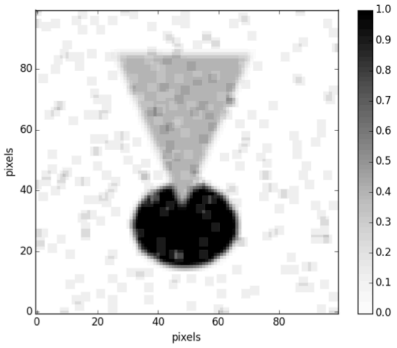
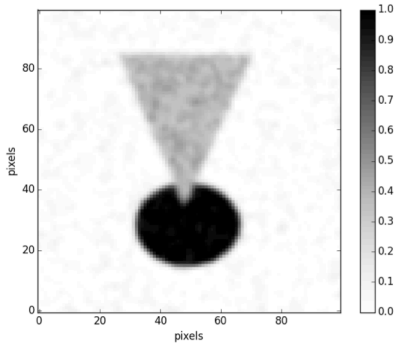
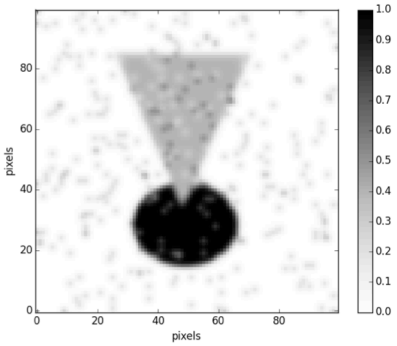
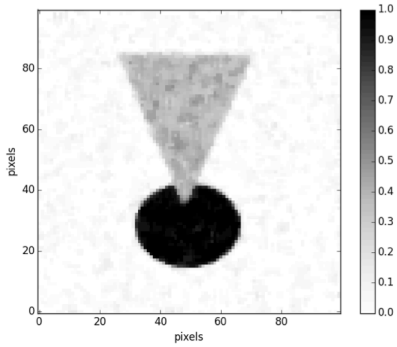
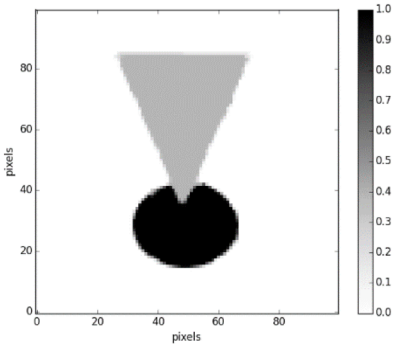
	Gaussian noise	Salt-and-pepper noise
Noisy images		
3x3 mean filter		
Gaussian filter with standard deviation $\sigma = 1$		
3x3 median filter		

Fig. 2: Effects of mean, Gaussian and median filters on image noise. The mean and median filters shown here consider a 3x3 block of surrounding pixels when computing the denoised value for each pixel.

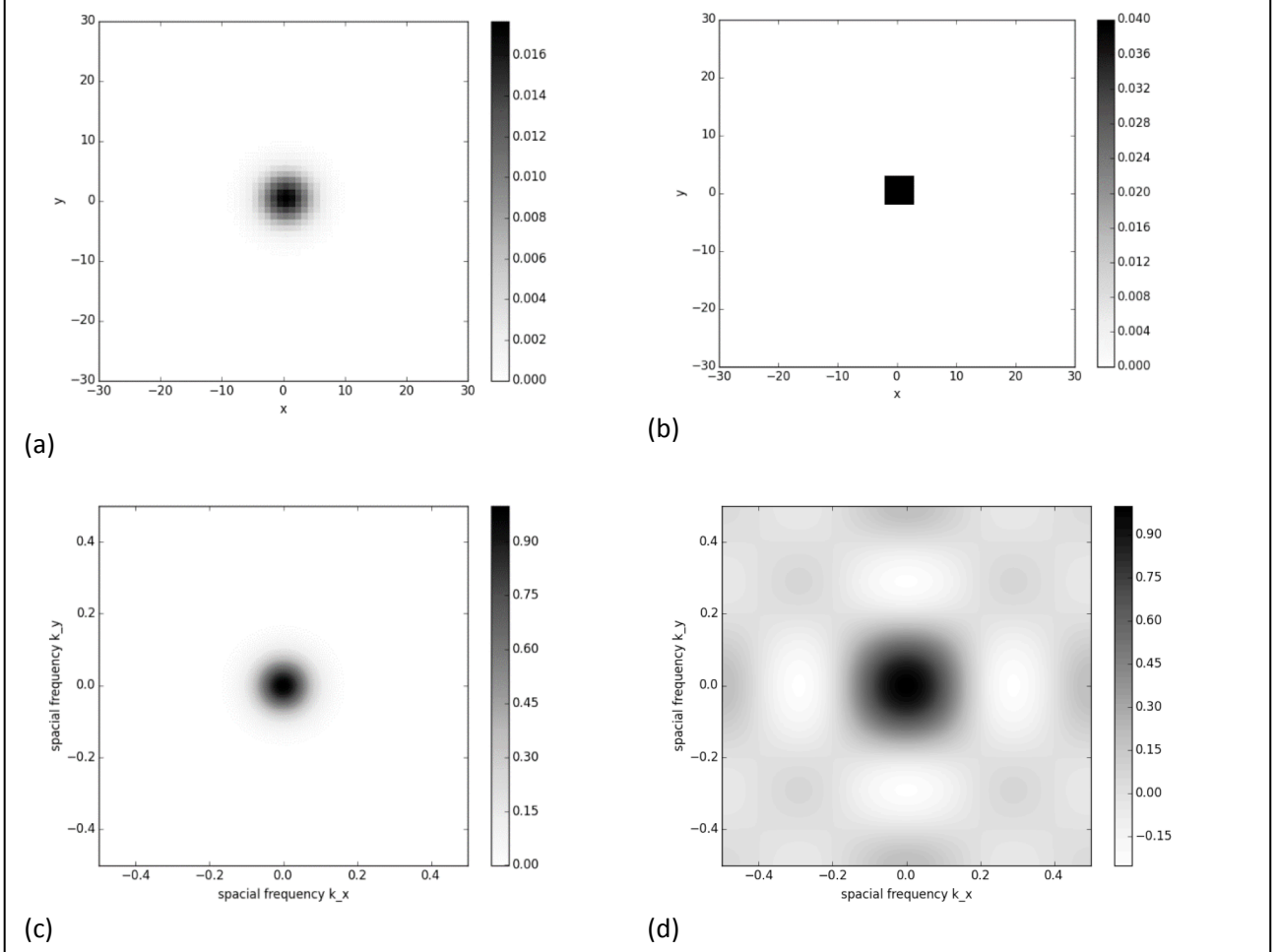


Fig. 3: (a) a Gaussian function with standard deviation $\sigma = 3$, (b) a 5x5 mean filter window function, and (c) & (d) their respective Fourier transforms

High-pass filtering

A high-pass filter preserves high frequency components while reducing the magnitude of low-frequency components. An easy way to high-pass filter an image is to subtract a low-pass filtered version of the image from the original image. Fig. 4 gives an example. High-pass filtering is useful for image sharpening (see lecture notes) and for highlighting edges in images. It will, however, tend to exacerbate noise.

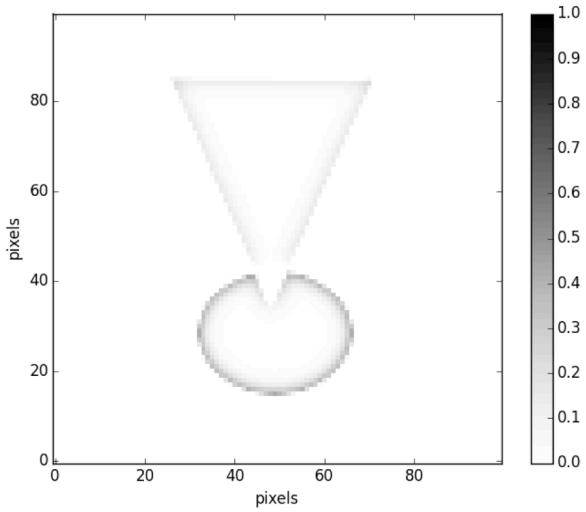


Fig. 4: Image in fig. 1a after applying a high-pass filter (by subtracting from the original image a version of the image filtered using a Gaussian filter with a standard deviation of $\sigma = 2$.

Principal Component Analysis

Suppose we have a large set of images, each 100x100 pixels. Each image has 10,000 pixels, so we can think of each image as corresponding to a point in a 10,000-dimensional space. The high dimensionality of the space makes it difficult to classify images or to identify the most meaningful patterns of variation between them. These tasks become easier if we can map the images to a lower-dimensional space. Principal component analysis is one common way to do this (and it's used in many fields, not just image analysis).

The basic idea of principal component analysis is to find a low-dimensional linear subspace (line, plane, etc.) that best fits the data points in the high-dimensional space. Then we can approximate each point in the high-dimensional space by the closest point in the low-dimensional linear subspace.

The first principal component (together with the mean of all the data) specifies the line that best fits the data, in the sense that if we compute the distance from all data points to that line, the sum of the squares of these distances is minimal (fig. 5). It turns out this is equivalent to saying that if we project all the data points onto this line (by finding the point on the line closest to each data point), the variance of those projected points is maximal.

Likewise, the first two principal components (together with the mean) specify the plane that best fits the data. One way to find the second principal component is to subtract from each point its projection onto the line specified by the first principal component. This gives us a new set of points, and the first principal component of the new set of points is the second principal component of the original data set.

In general, the first *N* principal components (together with the mean) specify the *N*-dimensional linear subspace that best fits the data. The principal component vectors will be orthogonal to one another.

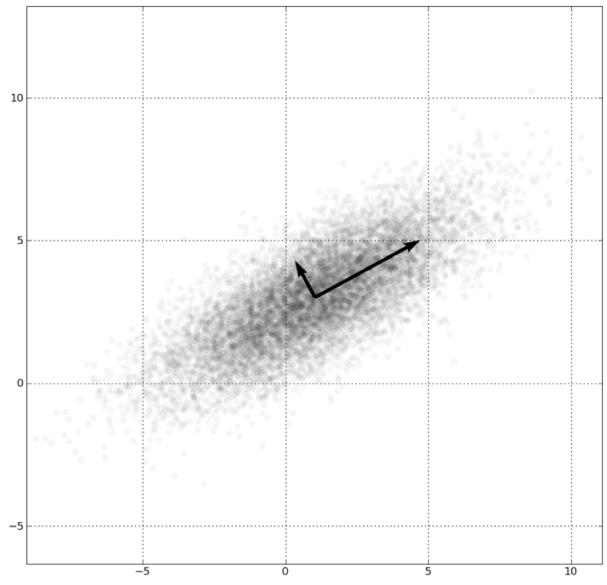


Fig. 5: The first principal component of this set of points is the vector specified by the longer arrow. The second principal component is specified by the shorter arrow. Image fromhttp://en.wikipedia.org/wiki/Covariance_matrix.