Fundamentals of Digital Image Processing

Roger L. Easton, Jr.

22 November 2010

Contents

References

Center for Image Processing in Education: lots of links to software and images
http://www.evisual.org/homepage.html

ImageJ software for image processing and analysis in Java, evolution of *NIHImage*
http://rsb.info.nih.gov/ij/

Image 2000 (from NASA)
http://www.ccpo.odu.edu/SEES/ozone/oz_i2k_soft.htm

Scion Image Processing Software (for PC and MAC-OS)
http://www.scioncorp.com/frames/fr_scion_products.htm

Hypercube Image Analysis Software (for PC and MAC-OS)
http://www.tec.army.mil/Hypercube/

GIMP Image Processing Software (Gnu-IMP) (free for PC, MacOS, Linux)
http://www.gimp.org/

Irfanview (free image processing viewer with some processing capability)
http://www.irfanview.com/

Gregory A. Baxes, **Digital Image Processing, Principles and Applications,** John Wiley & Sons, New York, 1994.

Ronald N. Bracewell, **Two-Dimensional Imaging**, Prentice Hall, Englewood Cliffs, 1995.

Ronald N. Bracewell, **The Fourier Transform and Its Applications** (Second Edition, Revised), McGraw-Hill, 1986.

Ronald N. Bracewell, **The Hartley Transform**, Oxford University Press, New York, 1986.

R.N. Bracewell, "The Fourier Transform", **Scientific American**, June 1989, pp.86-95.

Kenneth R. Castleman, **Digital Image Processing**, Prentice Hall, Englewood Cliffs, 1996.

E.O.Brigham, **The Fast Fourier Transform and its Applications**, Prentice Hall, Englewood Cliffs, 1988.

Michael P. Ekstrom, (Ed.), **Digital Image Processing Techniques**, Academic Press, New York, 1984.

B.R. Frieden, **Probability, Statistical Optics, and Data Testing**, Third Edition, Springer-Verlag, Berlin, 2002.

Jack D. Gaskill, **Linear Systems, Fourier Transforms, and Optics**, John Wiley & Sons, New York, 1978.

Rafael C. Gonzalez and Richard E. Woods, **Digital Image Processing**, Second Edition, Prentice Hall, Upper Saddle River, 2002.

Jae S. Lim, **Two-Dimensional Signal and Image Processing**, Prentice Hall, Englewood Cliffs, 1990.

Paul J. Nahin, **An Imaginary Tale**, Princeton University Press, Princeton NJ, 1998.

A. Nussbaum and R. Phillips, **Contemporary Optics for Scientists and Engineers**, Prentice-Hall, 1976.

Wayne Niblack, **An Introduction to Digital Image Processing,** Prentice Hall, Englewood Cliffs, 1986.

J. Anthony Parker, **Image Reconstruction in Radiology**, CRC Press, Boca Raton FL, 1990.

William K. Pratt, **Digital Image Processing**, Second Edition, John Wiley & Sons, New York, 1991.

Azriel Rosenfeld and Avinash C. Kak, **Digital Picture Processing**, Second Edition, Academic Press, San Diego, 1982.

Craig Scott, **Introduction to Optics and Optical Imaging**, IEEE Press, New York, 1998.

J.S.Walker, **Fast Fourier Transforms** 2nd Edition, CRC Press, New York, 1996.

Chapter 1

Basic Principles of Digital Image Processing

During the last two decades or so, inexpensive and powerful digital computers have become widely available and have been applied to a multitude of tasks. By hitching computers to new imaging detectors and displays, very capable systems for creating, analyzing, and manipulating imagery have been constructed and are being applied in many arenas. For example, they now are used to reconstruct X-ray and magnetic resonance images (MRI) in medicine, to analyze multispectral aerial and satellite images for environmental and military uses, to read Universal Product Codes that specify products and prices in retail stores, to name just a few.

Since I first taught a predecessor of this course in 1987, the capabilities of inexpensive imaging tools (cameras, printers, computers) have exploded (no surprise to you, I'm sure). This has produced a wide and ever-expanding range of applications that we could not even envision in those days. To give an idea of the change over the last two decades, consider the set of data shown below, which is copied directly from the first edition of Digital Image Processing by Gonzalez and Woods:

These data represent a 64×64 5-bit image ($2^5 = 32$ gray values). This data set was entered by hand (with only 4 mistakes) in 1988 by Ranjit Bhaskar, an imaging science graduate student for use by students. The image was rendered using the so-called "overstrike" routine on a line printer, where "dark" pixels were printed using several overprinted characters and lighter pixels by sparse characters (e.g. "." and "-"). The subject of the image is shown on the next page:

This course will investigate the basic principles of digital imaging systems and introduce useful applications; many simple examples will be provided to illustrate the concepts. First, a definition:

IMAGE: A reproduction or imitation of form of a person or thing.
The optical counterpart of an object produced by a lens, mirror, etc.
..................................Noah Webster

We normally think of an *image* in the sense of a *picture*, i.e., a planar representation of the brightness, , i.e., the amount of light reflected or transmitted by an object.

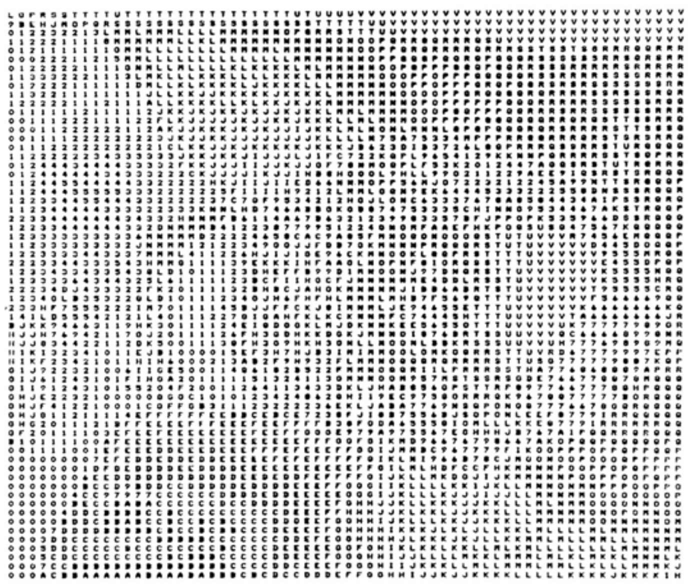


Figure 1.1: *Coded* 64 × 64 *5-bit image (32 gray values)*

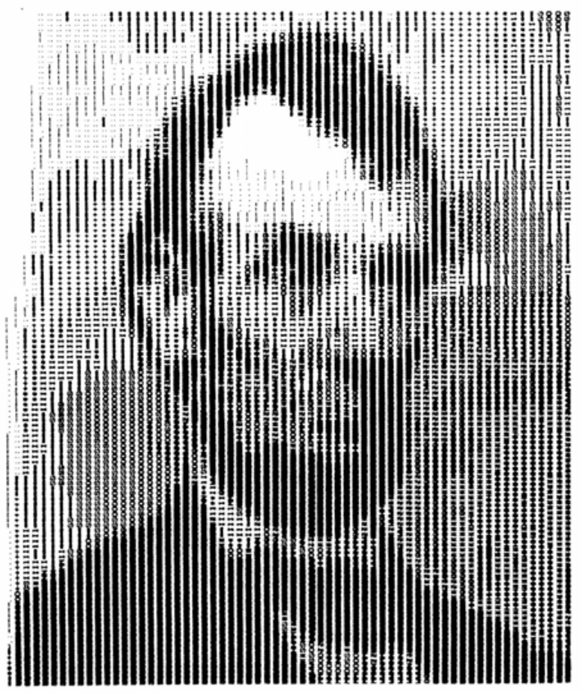
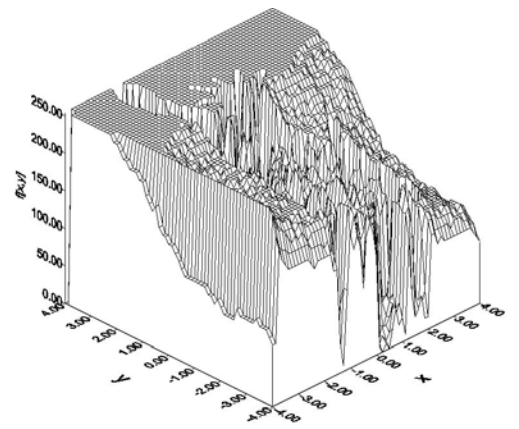


Figure 1.2: *Data in previous photo rendered using "overstrike" printout with line printer – this is how we used to do it, folks!*

An image is usually a function of two spatial variables, e.g., $f[x,y]$, which represents the brightness f at the Cartesian location $[x,y]$. Obviously, it also may be graphed in three dimensions, with brightness displayed on the z-axis.



Function of Two Spatial Coordinates
$f[x,y]$

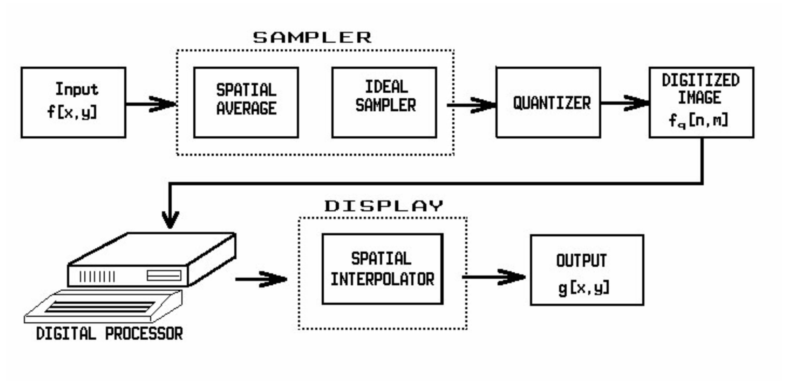
Image Representation of
$f[n,m]$

It is more and more common to deal with images that have more than two coordinate dimensions, e.g.,

$f[x,y,t_n]$	monochrome "movie", discrete set of images over time
$f[x,y,\lambda]$	spectral image with continuous domain of wavelengths
$f[x,y,\lambda_n]$	multispectral image, discrete set of wavelengths
$f[x,y,t]$	time-varying monochrome image over continuous time domain
$f[x,y,t_n]$	time-varying monochrome image with discrete time samples (cinema)
$f[x,y,z]$	3-D monochrome image (e.g., optical hologram)
$f[x,y,t_n,\lambda_m]$	discrete samples in time and wavelength, e.g., color movie
$f[x,y,z,t,\lambda]$	reality

It is generally fesible to "cut" 2-D slices from these multidimensional functions to create images, but the images need not be "pictorial." For example, consider the 2-D slices "cut" from the 3-D function spatial-temporal function $f[x,y,t]$; the 2-D slice $f[x,y;t=t_0]$ is pictorial but $f[x,y=y_0,t]$ is not. That said, the units of the axes have no effect on the computations; it is perfectly feasible for computers to process and display $f[x,y=y_0,t]$ as to do the same for $f[x,y;t_0]$.

After converting image information into an array of integers, the image can be manipulated, processed, and displayed by computer. Computer processing is used for image enhancement, restoration, segmentation, description, recognition, coding, reconstruction, transformation



1.1 Digital Processing

The general digital image processing system may be divided into three components: the input device (or digitizer), the digital processor, and the output device (image display).

1. The digitizer converts a continuous-tone and spatially continuous brightness distribution $f\,[x,y]$ to an discrete array (the digital image) $f_q[n,m]$, where n, m, and f_q are integers.

2. The digital processor operates on the digital image $f_q[n,m]$ to generate a new digital image $g_q[k,\ell]$, where k, ℓ, and g_q are integers. The output image may be represented in a different coordinate system, hence the use of different indices k and ℓ.

3. The image display converts the digital output image $g_q[k,\ell]$ back into a continuous-tone and spatially continuous image $g\,[x,y]$ for viewing. It should be noted that some systems may not require a display (e.g., in machine vision and artificial intelligence applications); the output may be a piece of information. For example, a digital imaging system that was designed to answer the question, Is there evidence of a cancerous tumor in this x-ray image?, ideally would have two possible outputs (YES or NO), , i.e., a single bit of information.

Note that the system includes most of the links in what we call the imaging chain. We shall first consider the mathematical description of image digitizing and display devices, and follow that by a long discussion of useful processing operations.

1.2 Digitization

Digitization is the conversion of a continuous-tone and spatially continuous brightness distribution $f\,[x,y]$ to an discrete array of integers $f_q[n,m]$ by two operations which will be discussed in turn:

(A) **SAMPLING** – a function of continuous coordinates $f[x,y]$ is evaluated on a discrete matrix of samples indexed by $[n,m]$. You probably saw some discussion of sampling in the course on linear and Fourier mathematics.

(B) **QUANTIZATION** – the continuously varying brightness f at each sample is converted to a one of set of integers f_q by some nonlinear thresholding process.

The digital image is a matrix of picture elements, or *pixels* if your ancestors are computers. Video descendents (and imaging science undergraduates) often speak of *pels* (often misspelled *pelz*). Each matrix element is an integer which encodes the brightness at that pixel. The integer value is called the *gray value* or *digital coun*t of the pixel.

Computers store integers as **BI**nary digi**TS**, or *bits* (0,1)

2 bits can represent: $00_\triangle = 0.$, $01_\triangle = 1$, $10_\triangle = 2.$, $11_\triangle = 3.$;a total of $2^2 = 4$ numbers.

(where the symbol "$\triangle$" denotes the binary analogue to the decimal point "." and thus may be called the "binary point," which separates the ordered bits with positive and negative powers of 2).

$$\begin{gathered} m \text{ BITS can represent } 2^m \text{ numbers} \\ \Longrightarrow 8 \text{ BITS} = 1 \text{ BYTE} \Longrightarrow 256 \text{ decimal numbers, } [0,255] \\ \Longrightarrow 12 \text{ BITS} = 4096 \text{ decimal numbers, } [0,4095] \\ \Longrightarrow 16 \text{ BITS} = 2^{16} = 65536 \text{ decimal numbers, } [0,65535] \end{gathered}$$

Digitized images contain finite numbers of data "bits" and it probably is apparent that the process of quantization discards some of the content of the image, i.e., the quantized image differs from the unquantized image, so errors have been created. Beyond that, we can consider the "amount" of "information" in the quantized image, which is defined as the number of bits required to store the image. The number of bits of "information" usually is smaller than the the number of bits of "data" (which is merely the product of the number of image pixels and the number of bits per pixel). The subject of information content is very important in imaging and will be considered in the section on image compression. We will discuss digitizing and reconstruction errors after describing the image display process.

Chapter 2

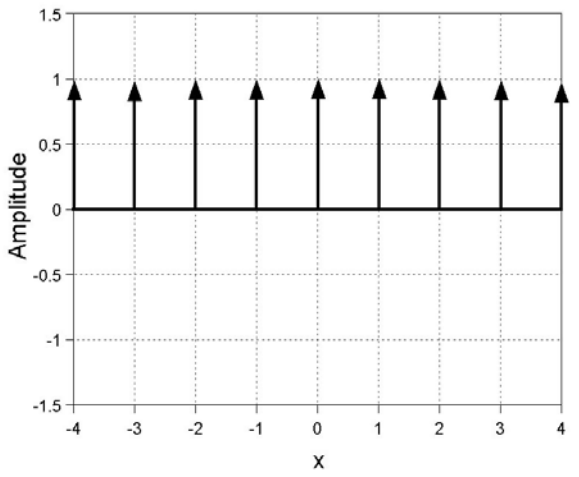
Review of Sampling

The process of “sampling” derives a discrete set of data points at (usually) uniform spacing. In its simplest form, sampling is expressed mathematically as multiplication of the original image by a function that “measures” the image brightness at discrete locations of infinitesimal width/area/volume in the 1-D/2-D/3-D cases:

$$\begin{aligned}
f_s[n\cdot\Delta x] &= f[x]\cdot s[x;n\cdot\Delta x]\\
&\text{where:}\\
f[x] &= \text{brightness distribution of input image}\\
s[x;n\cdot\Delta x] &= \text{sampling function}\\
f_s[n\cdot\Delta x] &= \text{sampled input image defined at coordinates } n\cdot\Delta x
\end{aligned}$$

The ideal sampling function for functions of continuous variables is generated from the so-called “Dirac delta function” $\delta[x]$, which is defined by many authors, including Gaskill. The ideal sampling function is the sum of uniformly spaced “discrete” Dirac delta functions, which Gaskill calls the *COMB* while Bracewell calls it the *SHAH*:

$$\begin{aligned}
COMB[x] &\equiv \sum_{n=-\infty}^{+\infty}\delta[x-n]\\
s[x;n\cdot\Delta x] &\equiv \sum_{n=-\infty}^{+\infty}\delta[x-n\cdot\Delta x]\equiv\frac{1}{\Delta x}COMB\left[\frac{x}{\Delta x}\right]
\end{aligned}$$



The *COMB* function defined by Gaskill (called the *SHAH* function by Bracewell).

For the (somewhat less rigorous) purpose we have here, we may consider the sampling function to just "grab" the value of the continuous input function $f[x,y]$ at the specific locations separated by the uniform intervals Δx and Δy, where $\Delta x = \Delta y$:

$$f_s[n,m] = f[n \cdot \Delta x, m \cdot \Delta y]$$

In other words, we are sweeping some unimportant and possibly confusing mathematical details under the rug.

2.0.1 Ideal Sampling of 1-D function

Multiplication of the input $f[x]$ by a *COMB* function merely evaluates $f[x]$ on the uniform grid of points located at $n \cdot \Delta x$, where n is an integer. Because it measures the value of the input at an infinitesmal point, this is a mathematical idealization that cannot be implemented in practice. Even so, the discussion of ideal sampling usefully introduces some essential concepts.

Consider ideal sampling of a sinusoidal input function with spatial period X_0 that is ideally sampled at intervals separated by Δx:

$$f[x] = \frac{1}{2}\left(1 + \cos\left[\frac{2\pi x}{X_0} + \phi_0\right]\right)$$

The amplitude of the function at the sample indexed by n is:

$$\begin{aligned} f_s[n \cdot \Delta x] &= \frac{1}{2} \cdot \left(1 + \cos\left[2\pi\left(\frac{n \cdot \Delta x}{X_0}\right) + \phi_0\right]\right) \\ &= \frac{1}{2} \cdot \left(1 + \cos\left[2\pi n \cdot \left(\frac{\Delta x}{X_0}\right) + \phi_0\right]\right) \end{aligned}$$

The dimensionless parameter $\frac{\Delta x}{X_0}$ in the second expression is the ratio of the sampling interval to the spatial period (wavelength) of the sinusoid and is a measurement of the fidelity of the sampled image. For illustration, examples of sampled functions obtained for several values of $\frac{\Delta x}{X_0}$ are:

$$\text{Case I: } X_0 = 12 \cdot \Delta x \implies \frac{\Delta x}{X_0} = \frac{1}{12}, \phi_0 = 0 \implies f_s[n] = \frac{1}{2} \cdot \left(1 + \cos\left[\frac{\pi n}{6}\right]\right)$$

$$\text{Case II: } X_0 = 2 \cdot \Delta x \implies \frac{\Delta x}{X_0} = \frac{1}{2}, \phi_0 = 0 \implies f_s[n] = \frac{1}{2} \cdot (1 + \cos[\pi n]) = \frac{1}{2}[1 + (-1)^n]$$

$$\text{Case III: } X_0 = 2 \cdot \Delta x \implies \frac{\Delta x}{X_0} = \frac{1}{2}, \phi_0 = -\frac{\pi}{2} \implies f_s[n] = \frac{1}{2} \cdot (1 + \sin[\pi n]) = \frac{1}{2}$$

$$\text{Case IV: } X_0 = \frac{4}{3} \cdot \Delta x \implies \frac{\Delta x}{X_0} = \frac{3}{4}, \phi_0 = 0$$

$$\implies f_s[n] = \frac{1}{2} \cdot \left(1 + \cos\left[\frac{2\pi n}{4/3}\right]\right) = \frac{1}{2} \cdot \left(1 + \cos\left[3\pi \frac{n}{2}\right]\right)$$

$$\text{Case V: } X_0 = \frac{4}{5} \cdot \Delta x \implies \frac{\Delta x}{X_0} = \frac{5}{4}, \phi_0 = 0$$

$$\implies f_s[n] = \frac{1}{2} \cdot \left(1 + \cos\left[\frac{2\pi n}{5/4}\right]\right) = \frac{1}{2} \cdot \left(1 + \cos\left[8\pi \frac{n}{5}\right]\right)$$

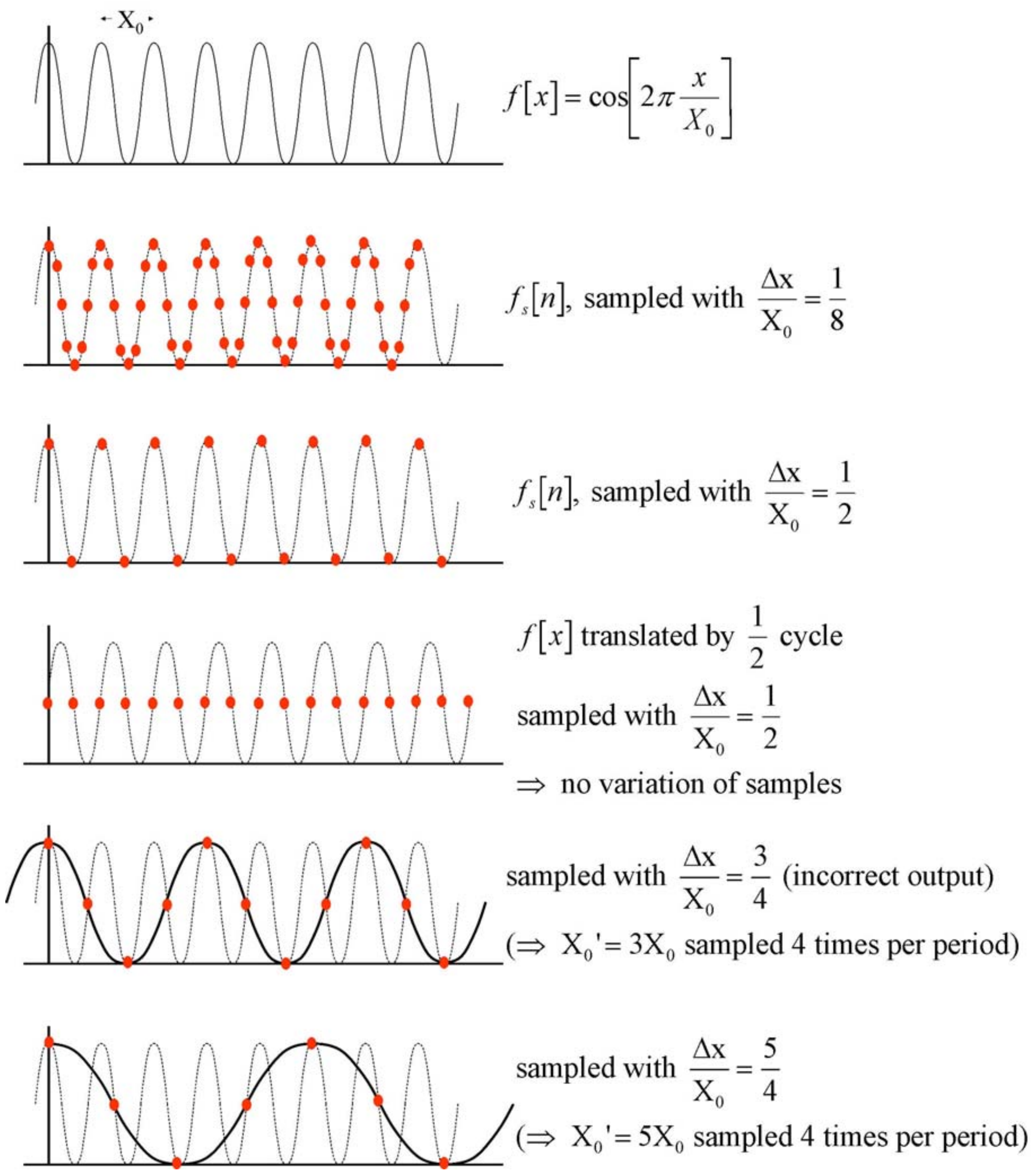
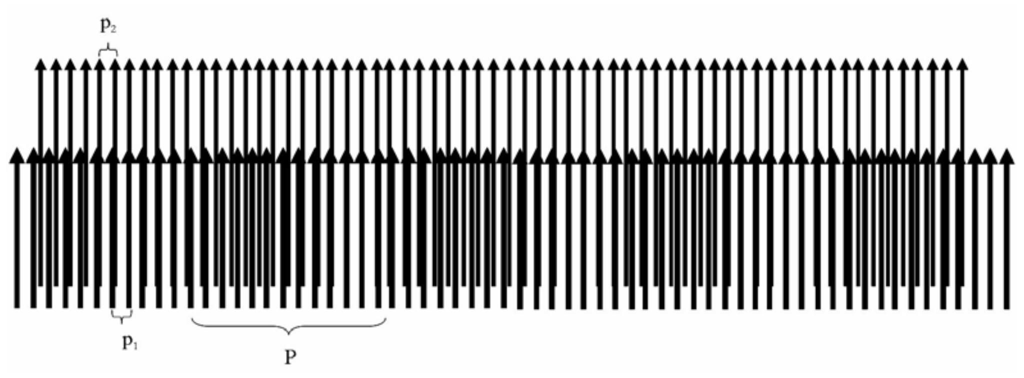


Illustration of samples of the biased sinusoids with the different values of $\frac{\Delta x}{X_0}$ listed in the table. The last three cases illustrate "aliasing."

The output evaluated for $\frac{\Delta x}{X_0} = \frac{1}{2}$ depends on the phase of the sinusoid; if sampled at the extrema, then the sampled signal has the same dynamic range as $f[x]$ (i.e., it is fully modulated), show no modulation, or any intermediate value. The interval $\Delta x = \frac{X_0}{2}$ defines the *Nyquist sampling limit.* If $\frac{\Delta x}{X_0} > \frac{1}{2}$ sample per period, then the same set of samples could have been obt5ained from a sinusoid with a longer period and a different sampling interval Δx. For example, if $\frac{\Delta x}{X_0} = \frac{3}{4}$, then the reconstructed function appears as though obtained from a sinudoid with period $X_0' = 3X_0$ if sampled with $\frac{\Delta x}{X_0'} = \frac{1}{4}$. In other words, the data set of samples is ambiguous; the same samples could be obtained from more than one input, and thus we cannot distinguish among the possible inputs based only on knowledge of the samples.



2.1 Aliasing – Whittaker-Shannon Sampling Theorem

As just demonstrated, the sample values obtained from a sinusoid which has been sampled fewer than two times per period will be identical to those obtained from a sinusoid with a longer period. This ambiguity about which original function produced the set of samples is called *aliasing* in sampling, but similar effects show up whenever periodic functions are multiplied or added. In other disciplines, these go by different names such as beats, Moiré fringes, and heterodyning. To illustrate, consider the product of two sinusoidal functions with the different periods $X_1$and X_2(and thus spatial frequencies $\xi_1 = \frac{1}{X_1}, \xi_2 = \frac{1}{X_2}$), which may be written as the *sum* of two sinusoids with different spatial frequencies:

$$\cos[2\pi\xi_1 x] \cdot \cos[2\pi\xi_2 x] = \frac{1}{2}\cos[2\pi(\xi_1+\xi_2)x] + \frac{1}{2}\cos[2\pi(\xi_1-\xi_2)x]$$

(note that the converse is also true; the sum of two sinusoids with the same amplitude and different frequencies may be written as the product of two sinusoids). The second term in the expression for the product oscillates slowly and is the analog of the aliased signal.

Though the proof is beyond our mathematical scope at this time, we can state that a sinusoidal signal that has been sampled without aliasing may be reconstructed without error from the (infinite set of) its ideal samples. This will be demonstrated in the section on image displays. Also without proof, we make the following claim:

Any 1-D or 2-D function may be expressed as a unique sum of 1-D or 2-D sinusoidal components with (generally) different amplitudes, frequencies, and phases.

This is the principle of Fourier analysis, which determines the set of sinusoidal components from the function. The set of amplitudes and phases of the sinusoidal components expressed as a function of frequency are the *Fourier components* of the function.

If the sinusoidal representation of $f[x]$ has a component with a maximum spatial frequency $\xi_{\max}$, and if we sample $f[x]$ so that this component is sampled without aliasing, then all sinusoidal components of $f[x]$ will be adequately sampled and $f[x]$can be perfectly reconstructed from its samples. Such a function is band-limited and $\xi_{\max}$ is the cutoff frequency of $f[x]$. The corresponding minimum spatial period is $X_{\min} = \frac{1}{\xi_{\max}}$. Thus the sampling interval Δx can be found from:

$$\frac{\Delta x}{X_{\min}} < \frac{1}{2} \Longrightarrow \Delta x < \frac{X_{\min}}{2} \Longrightarrow \Delta x < \frac{1}{2\xi_{\max}}$$

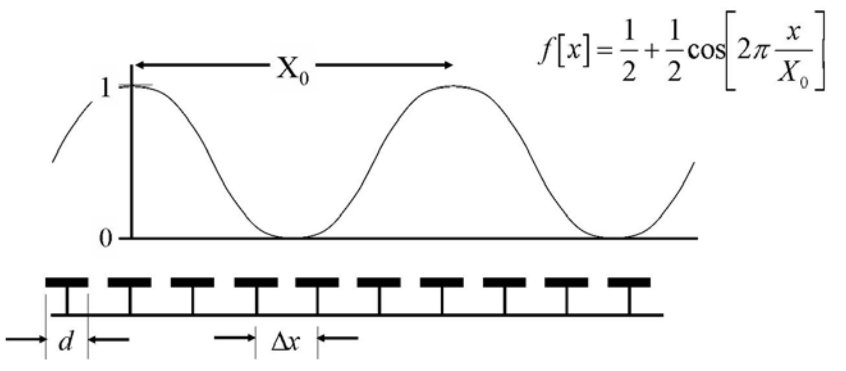
This is the *Whittaker-Shannon sampling theorem.* The limiting value of the sampling interval $\Delta x = \frac{1}{2\xi_{\max}}$ defines the *Nyquist sampling limit.* Sampling more or less frequently than the Nyquist limit is *oversampling* or *undersampling*, respectively.

$$\begin{aligned}\Delta x &> \frac{1}{2\xi_{\max}} \Longrightarrow \textit{undersampling}\\ \Delta x &< \frac{1}{2\xi_{\max}} \Longrightarrow \textit{oversampling}\end{aligned}$$

The Whittaker-Shannon Sampling Theorem is valid for all types of sampled signals. An increasingly familiar example is digital recording of audio signals (e.g., for compact discs or digital audio tape). The sampling interval is determined by the maximum audible frequency of the human ear, which is generally accepted to be approximately 20kHz. The sampling frequency of digital audio recorders is 44,000 $\frac{\text{samples}}{\text{second}}$which translates to a sampling interval of $\frac{1}{44{,}000\,\text{s}} = 22.7\,\mu\text{s}$. At this sampling rate, sounds with periods greater than $2 \cdot 22.7\,\mu\text{s} = 45.4\,\mu\text{s}$ (or frequencies less than $(45.4\,\mu\text{s})^{-1} = 22\,\text{kHz}$) can theoretically be reconstructed perfectly, assuming that $f[t]$ is sampled perfectly (i.e., at a point). Note that if the input signal frequency is greater than the Nyquist frequency of 22 kHz, the signal will be aliased and will appear as a lower-frequency signal in the audible range. Thus the reconstructed signal will be wrong. This is prevented by ensuring that no signals with frequencies above the Nyquist limit is allowed to reach the sampler; higher frequencies are filtered out before sampling.

2.2 Realistic Sampling – Averaging by the Detector

In fact, it is not possible to "grab" the amplited of a signal at specific locations with infinitesimal "width" (infinitesimal "support"). The measurement of a finite signal over the infinitesimally small area in the real world would produce an infinitesimal result. Rather a real system performs "realistic sampling," where the continuous input is measured over finite areas located at uniformly spaced samples by using a detector with finite spatial (or temporal) size. The measured signal is a spatial integral of the input signal over the detector area, which blurs the image. For example, in the common case where we assume that each sensor element has the same response over its full area, we can calculate the sample value by integrating the signal over the detector size. In the figure, the sensor elements are separated by Δx and each has width d_0:


$f[x]=\frac{1}{2}+\frac{1}{2}\cos\left[2\pi\frac{x}{X_0}\right]$
X_0
1
0
d
Δx

For a biased sinusoidal signal of the form:

$$f[x]=\frac{1}{2}\left(1+\cos\left[\frac{2\pi x}{X_0}+\phi_0\right]\right)$$

we define its *modulation* as:

$$m=\frac{f_{\max}-f_{\min}}{f_{\max}+f_{\min}};\ 0\leq m\leq 1\ \mathit{if}\ f_{\min}\geq 0$$

Note that modulation is only defined for nonnegative (i.e., biased) sinusoids. The analogous quantity for a nonnegative square wave is called *contrast.*

The biased sinusoidal signal is averaged over the detector area, e.g., the sampled value at $n=0$ is:

$$\begin{aligned} f_s[n=0] &= \frac{1}{d_0}\int_{-\frac{d_0}{2}}^{+\frac{d_0}{2}} f[x]\ dx \\ &= \frac{1}{d_0}\int_{-\infty}^{+\infty} f[x]\cdot RECT\left[\frac{x}{d_0}\right]\ dx \end{aligned}$$

$$\textit{where: } RECT\left[\frac{x}{d_0}\right]\equiv\begin{cases} 1\ \mathit{if} & |x|<\dfrac{d_0}{2} \\ \frac{1}{2}\ \mathit{if}\ |x|=\dfrac{d_0}{2} & \Longrightarrow\ x=\pm\dfrac{d_0}{2} \\ 0\ \mathit{if} & |x|>\dfrac{d_0}{2} \end{cases}$$

For $f[x]$ as defined above, the set of samples is derived by integrating $f[x]$ over the area of width d centered at coordinates that are integer multiples of Δx:

$$\begin{aligned}
&\frac{1}{d_0}\int_{n\cdot\Delta x-\frac{d_0}{2}}^{n\cdot\Delta x+\frac{d_0}{2}} \frac{1}{2}\left(1+\cos\left[\frac{2\pi x}{X_0}+\phi_0\right]\right)dx \\
&=\frac{1}{2d_0}\left(\int_{n\cdot\Delta x-\frac{d_0}{2}}^{n\cdot\Delta x+\frac{d_0}{2}} dx+\int_{n\cdot\Delta x-\frac{d_0}{2}}^{n\cdot\Delta x+\frac{d_0}{2}} \cos\left[\frac{2\pi x}{X_0}+\phi_0\right]\,dx\right) \\
&=\frac{1}{2d_0}\left[\left(n\cdot\Delta x+\frac{d_0}{2}\right)-\left(n\cdot\Delta x-\frac{d_0}{2}\right)\right]+\frac{1}{2d_0}\left.\frac{\sin\left[\frac{2\pi x}{X_0}+\phi_0\right]}{\frac{2\pi}{X_0}}\right|_{x=n\cdot\Delta x-\frac{d_0}{2}}^{x=n\cdot\Delta x+\frac{d_0}{2}} \\
&=\frac{1}{2}+\frac{1}{2d_0}\frac{\sin\left[2\pi n\cdot\frac{\Delta x}{X_0}+\frac{\pi d_0}{X_0}+\phi_0\right]-\sin\left[2\pi n\cdot\frac{\Delta x}{X_0}-\frac{\pi d_0}{X_0}+\phi_0\right]}{\left(\frac{2\pi}{X_0}\right)}
\end{aligned}$$

By defining

$$\begin{aligned}
\alpha &= 2\pi n\cdot\frac{\Delta x}{X_0}+\phi_0 \\
\beta &= \frac{\pi d_0}{X_0}
\end{aligned}$$

and applying the trigonometric identities:

$$\begin{aligned}
&\sin[\alpha\pm\beta]=\sin\alpha\cos\beta\pm\cos\alpha\sin\beta \\
\Longrightarrow\ &\sin[\alpha+\beta]-\sin[\alpha-\beta]=2\cos\alpha\sin\beta,
\end{aligned}$$

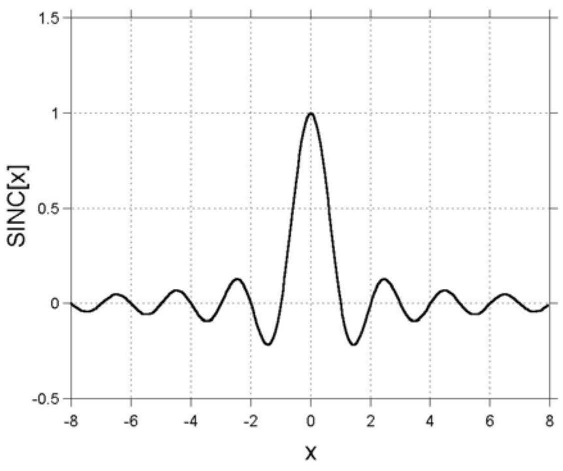
we find an expression for the integral over the detector area:

$$\begin{aligned}
f_s[n]&=\frac{1}{2}+\frac{1}{2d_0}\left(2\cos\left[2\pi n\cdot\frac{\Delta x}{X_0}+\phi_0\right]\cdot\frac{\sin\left[\frac{\pi d_0}{X_0}\right]}{\frac{2\pi}{X_0}}\right) \\
&=\frac{1}{2}+\frac{1}{2d_0}\left(2\cos\left[2\pi n\cdot\frac{\Delta x}{X_0}+\phi_0\right]\cdot d_0\cdot\frac{\sin\left[\frac{\pi d_0}{X_0}\right]}{2\pi\frac{d_0}{X_0}}\right)
\end{aligned}$$

The last multiplicative term may be simplified by applying the definition of the special function

$$\begin{aligned}
&SINC[\alpha]\equiv\frac{\sin[\pi\alpha]}{\pi\alpha} \\
\Longrightarrow\ &f_s[n]=\frac{1}{2}\left(1+SINC\left[\frac{d_0}{X_0}\right]\cdot\cos\left[2\pi n\cdot\frac{\Delta x}{X_0}+\phi_0\right]\right)
\end{aligned}$$

which varies with period X_0 and detector width d_0.



$SINC[x] \equiv \dfrac{\sin[\pi x]}{\pi x}$*; note that the value at the origin is unity, which may be demonstrated via l'Hôpital's rule.*

Note that for constant functions, the period $X_0 = \infty$ and the resulting value of the $SINC$ function is:

$$\lim_{X_0 \to \infty} \left\{ SINC\left(\frac{d_0}{X_0}\right) \right\} = 1$$

which means that uniform weighted averaging has no effect on any constant input function. The samples of cosine of period X_0 obtained with sampling interval Δx in the two cases are:

$$\begin{aligned} \textit{Realistic}: f_s[n] &= \frac{1}{2} \cdot \left(1 + SINC\left[\frac{d_0}{X_0}\right] \cdot \cos\left[2\pi \cdot n \cdot \frac{\Delta x}{X_0} + \phi_0\right]\right) \\ \textit{Ideal}: \ f_s[n] &= \frac{1}{2} \cdot \left(1 + \cos\left[2\pi \cdot n \cdot \frac{\Delta x}{X_0} + \phi_0\right]\right) \end{aligned}$$

where d_0 is the width of the detector. The amplitude of the realistic case is multiplied by a factor of $SINC\left[\frac{d_0}{X_0}\right]$, which is less than unity everywhere except at the origin, , i.e., where $d_0 = 0$ or $X_0 = \infty$. As the detector size increases relative to the spatial period of the cosine (i.e., as $\frac{d_0}{X_0}$ *increases*) , then $SINC\left[\frac{d_0}{X_0}\right] \to 0$ and the modulation of the sinusoid decreases.

The modulation of the image of a sine-wave of period X_0, or spatial frequency $\xi = \frac{1}{X_0}$, is reduced by a factor $SINC\left[\frac{d_0}{X_0}\right] = SINC[d_0\xi_0]$.

Example of Reduced Modulation due to Prefiltering

The input function $f[x]$ has a period of 128 units with two periods plotted. It is the sum of six sinusoidal components plus a constant:

$$f[x] = \frac{1}{2} + \frac{1}{2}\sum_{n=1}^{6} \frac{(-1)^{n-1}}{n} \sin\left[2\pi \frac{(2n-1)\,x}{256}\right].$$

The periods of the component sinusoids are:

$$\begin{aligned}
X_1 &= \frac{128}{1} \text{ units} \Longrightarrow \xi_1 = \frac{1}{128}\frac{\text{cycles}}{\text{unit}} \simeq 0.0078\frac{\text{cycles}}{\text{unit}} \\
X_2 &= \frac{128}{3} \text{ units} \simeq 42.7 \text{ units} \Longrightarrow \xi_2 = \frac{3}{128}\frac{\text{cycles}}{\text{unit}} \simeq 0.023\frac{\text{cycles}}{\text{unit}} \\
X_3 &= \frac{128}{5} \text{ units} = 25.6 \text{ units} \Longrightarrow \xi_3 = \frac{5}{128}\frac{\text{cycles}}{\text{unit}} \simeq 0.039\frac{\text{cycles}}{\text{unit}} \\
X_4 &= \frac{128}{7} \text{ units} \simeq 18.3 \text{ units} \Longrightarrow \xi_4 = \frac{7}{128}\frac{\text{cycles}}{\text{unit}} \simeq 0.055\frac{\text{cycles}}{\text{unit}} \\
X_5 &= \frac{128}{9} \text{ units} \simeq 14.2 \text{ units} \Longrightarrow \xi_4 = \frac{9}{128}\frac{\text{cycles}}{\text{unit}} \simeq 0.070\frac{\text{cycles}}{\text{unit}} \\
X_6 &= \frac{128}{11} \text{ units} \simeq 11.7 \text{ units} \Longrightarrow \xi_4 = \frac{11}{128}\frac{\text{cycles}}{\text{unit}} \simeq 0.086\frac{\text{cycles}}{\text{unit}}
\end{aligned}$$

The constant bias of 0.5 ensures that the function is positive. The first sinusoidal component ($X_{01} = 128$ units) is the fundamental and carries most of the modulation of the image; the other components (the higher harmonics) have less amplitude. The spatial frequency of each component is much less than the Nyquist limit of 0.5.

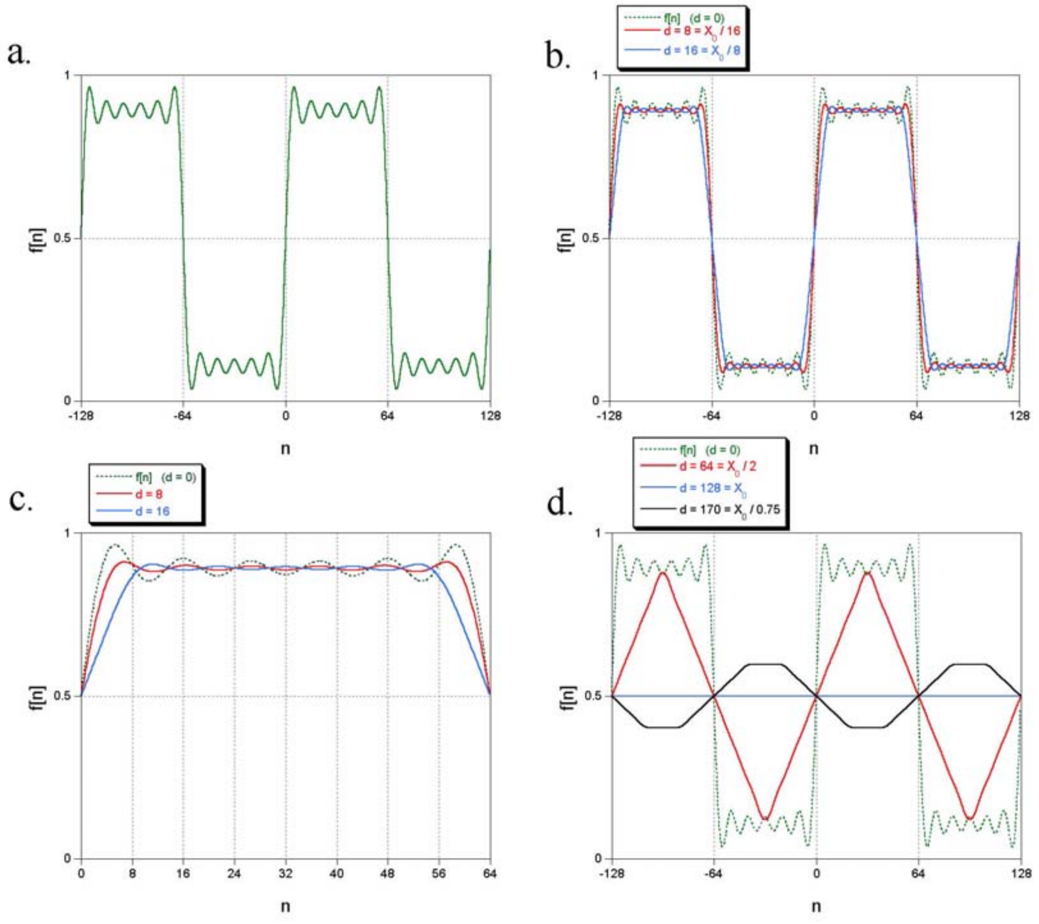


Illustration of the reduction in modulation due to "prefiltering": (a) input function $f\,[n]$; (b) result of prefiltering with uniform averagers of width $d = 0$, $d = \frac{X_0}{16}$, and $d = \frac{X_0}{8}$; (c) magnified view of (b), showing the change in the signal; (d) result offiltering with uniform averagers of width $d = \frac{X_0}{2}$, $d = X_0$, and $d = \frac{X_0}{0.75}$, showing the "contrast reversal" in the last case.

$$\begin{aligned}
SINC\left[\frac{d_0}{X_{01}}\right] &= SINC\left[d_0\xi_1\right] = SINC\left[8\cdot\frac{1}{128}\right] \simeq 0.994\\
SINC\left[\frac{d_0}{X_{02}}\right] &= SINC\left[d_0\xi_2\right] = SINC\left[8\cdot\frac{3}{128}\right] \simeq 0.943\\
SINC\left[\frac{d_0}{X_{03}}\right] &= SINC\left[d_0\xi_3\right] = SINC\left[8\cdot\frac{5}{128}\right] \simeq 0.847\\
SINC\left[\frac{d_0}{X_{04}}\right] &= SINC\left[d_0\xi_4\right] = SINC\left[8\cdot\frac{7}{128}\right] \simeq 0.714\\
SINC\left[\frac{d_0}{X_{05}}\right] &= SINC\left[d_0\xi_5\right] = SINC\left[8\cdot\frac{9}{128}\right] \simeq 0.555\\
SINC\left[\frac{d_0}{X_{06}}\right] &= SINC\left[d_0\xi_6\right] = SINC\left[8\cdot\frac{11}{128}\right] \simeq 0.385
\end{aligned}$$

Note that the modulation of sinusoidal components with shorter periods (higher frequencies) are diminished more severely by the averaging. A set of prefiltered images for several different averaging widths is shown on a following page. If the detector width is 32 units, the resulting modulations are:

$$\begin{aligned}
SINC\left[d_0\xi_1\right] &= SINC\left[32\cdot\frac{1}{128}\right] \simeq 0.900\\
SINC\left[d_0\xi_2\right] &= SINC\left[32\cdot\frac{3}{128}\right] \simeq 0.300\\
SINC\left[d_0\xi_3\right] &\simeq -0.180\\
SINC\left[d_0\xi_4\right] &\simeq -0.129\\
SINC\left[d_0\xi_5\right] &\simeq -0.100\\
SINC\left[d_0\xi_6\right] &\simeq +0.082
\end{aligned}$$

Note that the components with periods X_{04} and X_{05} have negative modulation, , i.e., $f_{\max} < f_{\min}$. The contrast of those components is reversed. As shown, the sampled image looks like a sawtooth with a period of 128 units.

If the detector size is 128, each component is averaged over an integral number of periods and the result is just the constant bias; the modulation of the output is zero:

$$\begin{aligned}
SINC\left[d_0\xi_1\right] &= SINC\left[\frac{128}{128}\right] = SINC\left[1\right] = 0\\
SINC\left[d_0\xi_2\right] &= SINC\left[128\cdot\frac{7}{42}\right] = SINC\left[3\right] = 0
\end{aligned}$$

For a detector width of 170 units, the modulations are:

$$\begin{aligned}
&SINC\left[d_0\xi_1\right] = SINC\left[170 \cdot \frac{1}{128}\right] \simeq -0.206 \\
&SINC\left[d_0\xi_2\right] = SINC\left[170 \cdot \frac{3}{128}\right] \simeq -0.004 \\
&SINC\left[d_0\xi_3\right] = SINC\left[170 \cdot \frac{5}{128}\right] \simeq +0.043 \\
&SINC\left[d_0\xi_4\right] = SINC\left[170 \cdot \frac{7}{128}\right] \simeq -0.028 \\
&SINC\left[d_0\xi_5\right] \simeq -0.004 \\
&SINC\left[d_0\xi_6\right] \simeq +0.021
\end{aligned}$$

Because the first (largest amplitude) sinusoidal component has negative modulation, so does the resulting image. The overall image contrast is reversed; darker areas of the input become brighter in the image.

Chapter 3

Review of Quantization

3.1 Quantization: A/D Conversion

The process of *quantization* converts the continuously valued irradiance ("lightness" or "brightness") measured at a sample (or a derived signal, such as the voltage measured from that signal) to one of a discrete set of *gray levels* (sometimes called *digital counts* though its acronym of "DC" may be confused with that for "direct current"). For example, the measured signal at the sample located at $[x_0, y_0]$ might be $f[x_0, y_0] = 1.234567890 \cdots \frac{\mathrm{W}}{\mathrm{mm}^2}$. This number is converted to some integer value, perhaps $f_q = 42$. The range of allowed integers is determined by various factors in the quantizer. For example, we might have $0 \leq f_q \leq (f_q)_{\max}$ where the maximum value $(f_q)_{\max}$ is determined by the number of bits in the quantizer: $(f_q)_{\max} = 255 = 2^8 - 1$ for an 8-bit quantizer and $(f_q)_{\max} = 2^{12} - 1 = 4095$ for the (increasingly common) 12-bit quantizer in scientific digital cameras.

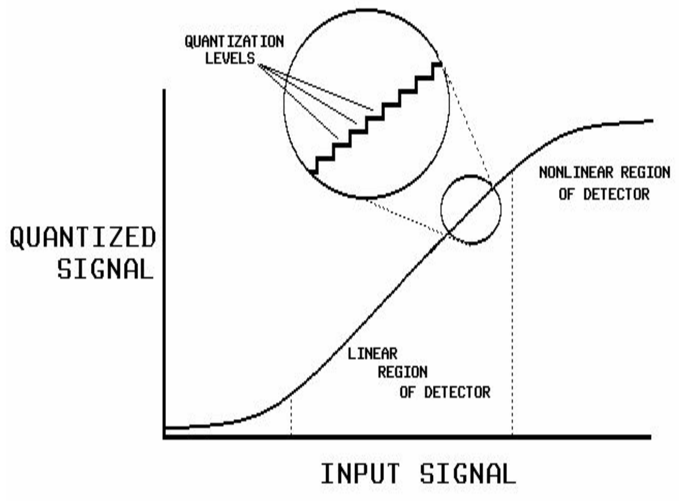
3.1.1 Tone Transfer Function

1. Quantization is significantly affected by detector parameters, including its *dynamic range* and *linearity*. The dynamic range of a detector image may be defined as the range of brightness (irradiance) over which a change in the input signal produces a *detectable* change in the measured output of the sensor. Note that the input and output quantities need not be identical; for example, in old-fashioned emulsion photography, the input might be measured as power per unit area (e.g., $\frac{\mathrm{W}}{\mathrm{mm}^2}$) and the resulting output in the dimensionless optical density D where $0 \leq D \leq \infty$. The effect of the detector on the measurement may be described by a *transfer characteristic* or *tone-transfer function* (TTF, also called the *tone-transfer curve* TTC), which is merely a graph of the output value as a function of the various inputs to the sensor. The shape of the transfer characteristic may be used as a figure of merit for the measurement process. A detector is linear if the TTC is a straight line, i.e., if an incremental change in input from any level produces a fixed incremental change in the output. Of course, all real detectors have a limited dynamic range, i.e., they will not respond at all to light intensity below some minimum value and their

response will not change for intensities above some maximum. All realistic detectors are therefore nonlinear, but there may be some regions over which they are more-or-less linear, with nonlinear regions at either end. A common such example is photographic film; the TTC is the *Hurter-Driffield* (*H&D*) curve, which is a graph of the recorded optical density of the emulsion as a function of the logarithm of the input irradiance (which might be measured in $\frac{\text{W}}{\text{mm}^2}$). Another very important example in digital imaging is the video camera, whose TTC maps input light intensity to output voltage. The transfer characteristic of a video camera is approximately a power law:

$$V_{out} = c_1 B_{in}^{\gamma} + V_0$$

where V_0 is the threshold voltage for a dark input and γ (gamma) is the exponent of the power law. The value of γ depends on the specific detector but typical values are $\gamma \simeq 1.7$ for a vidicon camera and $\gamma \simeq 1$ for an image orthicon. The value of gamma for a digital sensor (CCD) is inherently very close to unity, but are often modified in software to approximate photographic emulsions.



Possible tone-transfer function of quantizer applied to photographic emulsion, showing the "toe" on the left, "linear region" in the center, and "shoulder" on the right.

The *resolution*, or *step size* b_0, of the quantizer is the difference in the measured signal at the center of adjacent gray levels. It makes little sense to quantize the sampled signal with a resolution b_0 that is smaller than the signal uncertainty due to noise in the detector system. Thus the effective number of levels is often less than the maximum available as determined by the number of bits.

Conversion from a continuous range to discrete levels requires a thresholding operation (e.g.,truncation or rounding). Some range of input brightnesses will map to a single output level, e.g., all measured irradiances between 0.76 and 0.77$\frac{\text{W}}{\text{mm}^2}$ might map to gray level 59. Threshold conversion is a nonlinear operation, i.e., the threshold of a sum of two inputs is not necessarily the sum of the thresholded outputs. The concept of linear operators will be discussed extensively later, but we should say at this point that the nonlinearity due to quantization makes it inappropriate to analyze the complete digital imaging system (digitizer, processor, and display) by common linear methods. This problem is usually ignored, as is appropriate for large numbers of quantized levels that are closely spaced so that the digitized image appears continuous. Because the brightness resolution of the eye-brain is limited, quantizing to only 50 levels is satisfactory for many images; in other words, 6bits of data is often sufficient for images to be viewed by humans.

Quantization is performed by some kind of *digital comparator*, which is an *inherently* nonlinear electronic device that accepts an analog input and produces one of two possible output values that depend on whether the analog input is smaller or larger than some reference signal; if smaller, the output is a value that may be normalized to "0" and if) or if larger, the output is a "1". For example, if the reference of the comparator is set at $0.5\,\mathrm{V}$ and if the input is $f_{\text{input}} = 0.3\,\mathrm{V}$, then the output is $\mathcal{Q}\{0.3\,\mathrm{V}\} = 0$. If the input voltage is doubled $f_{\text{input}} = 0.6\,\mathrm{V}$, then the output is $\mathcal{Q}\{0.6\,\mathrm{V}\} = 1$. Note that this behavior clearly does not satisfy the requirement for a linear operator $\mathcal{L}$, where the output is doubled if the input is doubled:

$$\textit{if } \mathcal{L}\{f\} = g, \textit{ then } \mathcal{L}\{2f\} = 2g \textit{ for a linear operator } \mathcal{L}$$

The more general expression for a linear operator is::

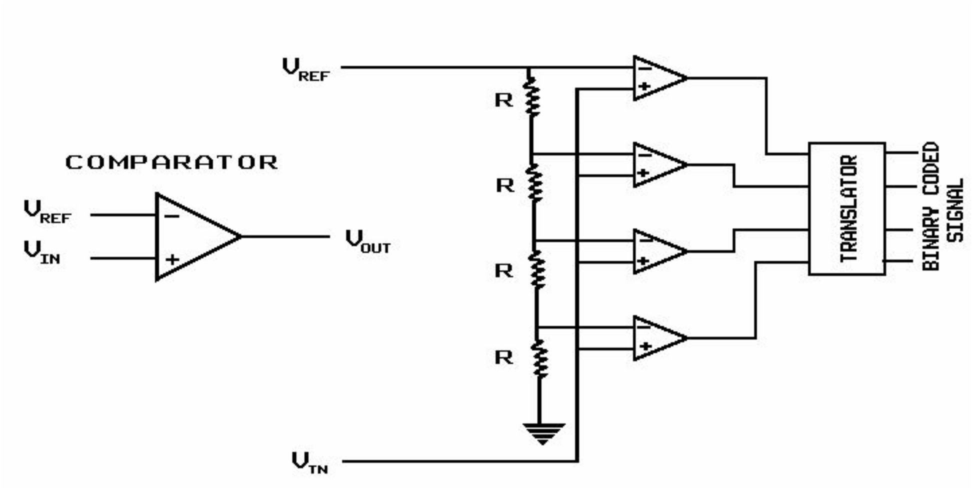
$$\textit{if } \mathcal{L}\{f_n\} = g_n, \textit{ then } \mathcal{L}\left\{\sum_{n=1}^{N} \alpha_n f_n\right\} = \sum_{n=1}^{N} \alpha_n g_n \textit{ for a linear operator } \mathcal{L}$$

where $\{\alpha_n\}$ is a set of numerical (generally complex-valued) weights.

The simplest quantizer converts an analog input voltage to a 1-bit digital output and can be constructed from an ideal differential amplifier, where the output voltage V_{out} is proportional to the difference of the input signal V_{in} and some reference V_{ref} that is provided by a calibrated source:

$$V_{\text{out}} = \alpha(V_{\text{in}} - V_{\text{ref}})$$

If the weighting factor α is sufficiently large to approximate ∞, then the output voltage will be $+\infty$ if $V_{\text{in}} > V_{\text{ref}}$ and $-\infty$ if $V_{\text{in}} < V_{\text{ref}}$. In this simple example, we would assign the digital value "0" to a negative output where $V_{\text{in}} < V_{\text{ref}}$ and "1" to a positive output such that $V_{\text{in}} > V_{\text{ref}}$. A quantizer with better resolution may be constructed by cascading several such digital comparators with different (but generally evenly spaced) reference voltages. A digital translator converts the set of comparator signals to the binary code. The 1-bit comparator and a 2-bit analog-to-digital converter (ADC) are shown in the figure:



Comparator and 2-Bit analog-to-digital converter (ADC). The comparator may be interpreted as an amplifier with "infinite" gain, so that its output is a "high" voltage if $V_{in} > V_{ref}$ and a "low" voltage otherwise. The schematic of the ADC consists of 4 comparators whose reference voltages are determined by the voltage divider with the resistor ladder.

In the most common scheme of uniform quantization, the step size b is fixed at the value:

$$b = \frac{f_{\max} - f_{\min}}{2^m - 1}$$

where $f_{\max}$ and $f_{\min}$ are the extrema of the measured irradiances of the image samples and m is the number of bits of the quantizer.

If the darkest and brightest samples of a continuous-tone image have measured irradiances $f_{\min}$ and $f_{\max}$ respectively, and the image is to be quantized using m bits (2^m graylevels), then we may define a set of uniformly spaced levels $f_q$that span the dynamic range via:

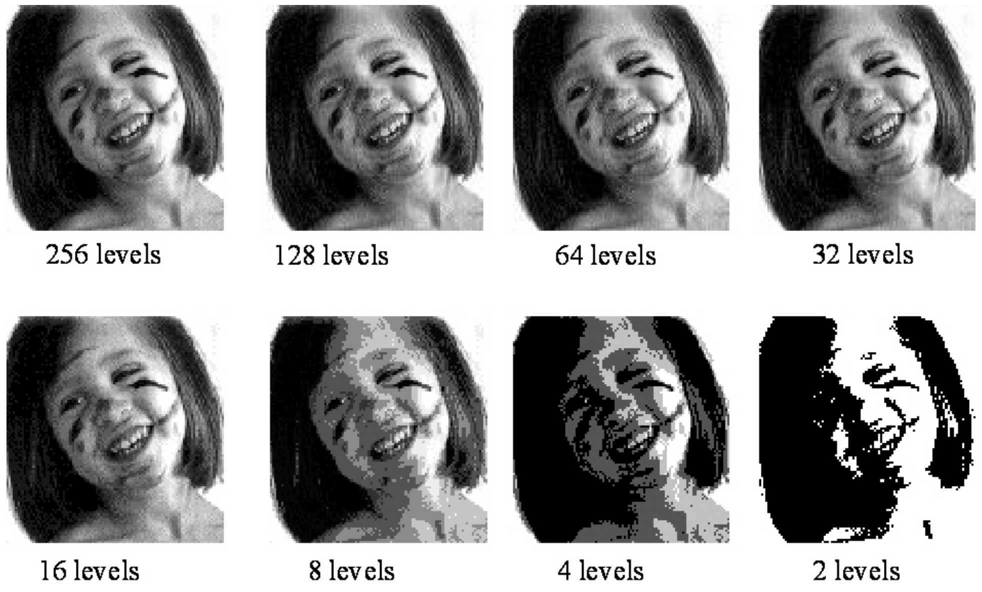
$$f_q[x,y] = \mathcal{Q}\left\{\frac{f[x,y] - f_{\min}}{b}\right\} = \mathcal{Q}\left\{\frac{f[x,y] - f_{\min}}{f_{\max} - f_{\min}} \cdot (2^m - 1)\right\}$$

where $\mathcal{Q}\{\ \}$ represents the nonlinear truncation or rounding operation, e.g., $\mathcal{Q}\{3.657\} = 3$ if $\mathcal{Q}$ is truncation or 4 if $\mathcal{Q}$ is rounding. The form of $\mathcal{Q}$ determines the location of the decision levels where the quantizer jumps from one level to the next. The image irradiances are reconstructed by assigning all pixels with a particular gray level f_q to the same irradiance value $E[x,y]$, which might be defined by "inverting" the quantization relation. The reconstruction level is often placed between the decision levels by adding a factor $\frac{b}{2}$:

$$\hat{E}[x,y] = \left(f_q[x,y] \cdot \frac{E_{\max} - E_{\min}}{2^m - 1}\right) + E_{\min} + \frac{b}{2}$$

Usually (of course), $\hat{E}[x,y] \neq E[x,y]$ due to the quantization, i.e., there will be quantization error. The goal of optimum quantization is to adjust the quantization scheme to reconstruct the set of image irradiances which most closely approximates the ensemble of original values. The criterion which defines the goodness of fit and the statistics of the original irradiances will determine the parameters of the quantizer, e.g., the set of thresholds between the levels.

The quantizer just described is *memoryless*, i.e., the quantization level for a pixel is computed independently that for any other pixel. The schematic of a memoryless quantizer is shown below. As will be discussed, a quantizer with memory may have significant advantages.



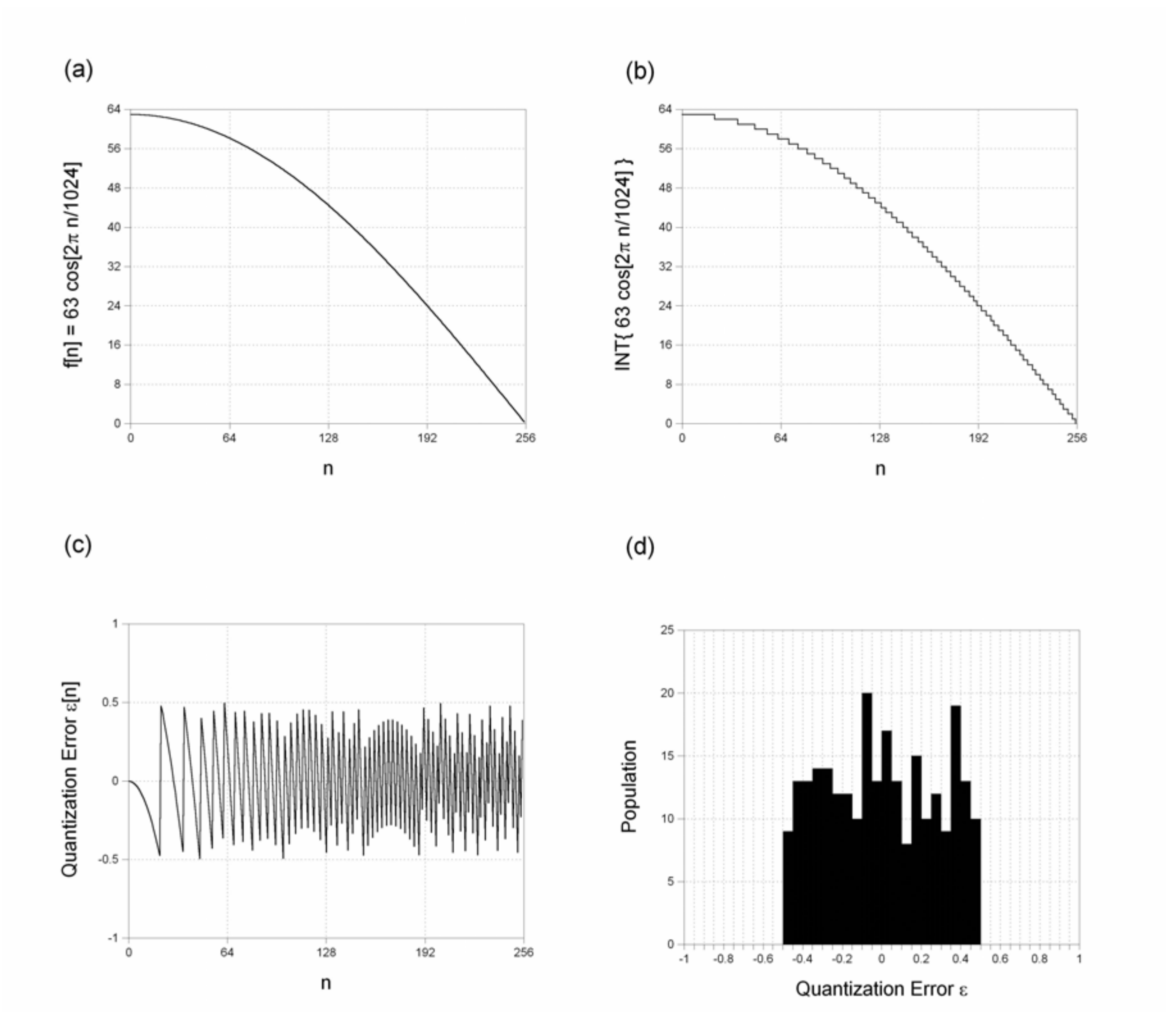
Effect of different bit depths (numbers of gray values) on image appearance from 8 bits (256 levels) down to 1 bit (2 levels). Not the "contouring" apparent for 2 and 3 bits (4 and 8 levels).

3.2 Quantization Error ("Noise")

The difference between the true input irradiance (or brightness) and the corresponding irradiance corresponding to the computed digital level is the *quantization error* at that pixel:

$$\epsilon\left[n\cdot\Delta x, m\cdot\Delta y\right] \equiv f\left[n\cdot\Delta x, m\cdot\Delta y\right] - f_q\left[n\cdot\Delta x, m\cdot\Delta y\right].$$

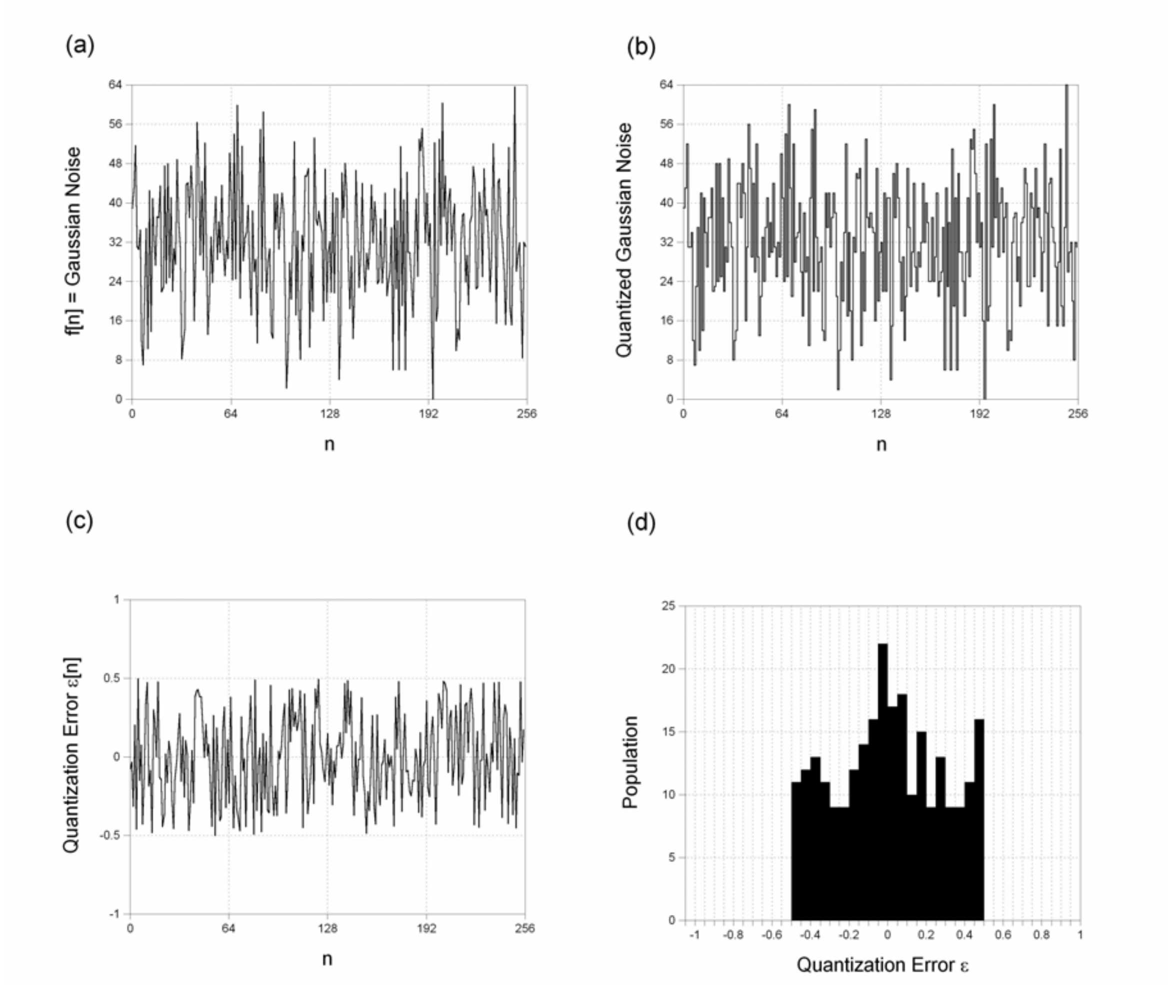
Note that the quantization error is bipolar in general, i.e., ϵ may take on positive or negative values. It often is useful to describe the statistical properties of the quantization error, which will be a function of both the type of quantizer and the input image. However, if the difference between quantization steps (i.e., the width of a quantization level) is b, is constant, the quantization error for most images may be approximated as a uniform distribution with mean value $\langle\epsilon\left[n\right]\rangle = 0$ and variance $\langle(\epsilon_1[n])^2\rangle = \frac{b^2}{12}$. The error distribution will be demonstrated for two 1-D 256-sample images. The first is a section of a cosine sampled at 256 points and quantized to 64 levels separated by $b = 1$:



Statistics of quantization noise: (a) $f[n] = 63 \cdot \cos\left[2\pi \frac{n}{1024}\right]$ *where* $0 \leq n \leq 255$ *(one quarter of a cycle); (b) after quantization by rounding to nearest integer; (c) quantization error* $\varepsilon[n] = f[n] - f_q[n]$*, showing that* $-\frac{1}{2} \leq \varepsilon \leq +\frac{1}{2}$*; (d) histogram of 256 samples of quantization error, which shows that the error is approximately uniformly distributed.*

The computed mean of the error $\epsilon_1[n] = f_1[n] - Q\{f_1[n]\}$ is $\langle \epsilon_1[n] \rangle = -5.1 \cdot 10^{-4}$ and the variance is $\langle \epsilon_1^2[n] \rangle = 0.08 \simeq \frac{1}{12}$. That the distribution of the errors is approximately uniform is shown by the histogram.

The second image is comprised of 256 samples of Gaussian distributed random noise in the interval [0,63]. The image is quantized to 64 levels.Again, the error $\epsilon_2[n]$ is uniformly distributed in the interval $[-0.5, +0.5]$ with mean $4.09 \cdot 10^{-2} \simeq 0$ and variance $\sigma^2 = \langle \epsilon_2^2[n] \rangle >= 0.09 \simeq \frac{1}{12}$.



Statistics of quantization error for Gaussian distributed random noise: (a) $f[n]$ with $\mu \cong 27.7, \sigma \cong 11$; (b) after quantization by rounding to nearest integer; (c) quantization error $\varepsilon[n] = f[n] - f_q[n]$, showing that $-\frac{1}{2} \leq \varepsilon \leq +\frac{1}{2}$; (d) histogram of 256 samples of quantization error, which shows that the error is approximately uniformly distributed.

The total quantization error is the sum of the quantization error over all pixels in the image:

$$\epsilon = \sum_i \sum_j \epsilon [n \cdot \Delta x, m \cdot \Delta y] \, .$$

An image with large bipolar error values may thus have a small total error. The mean-squared error (the average of the squared error) is a better descriptor of the fidelity of the quantization:

$$\epsilon^2 = \frac{1}{N} \sum_i \sum_j \epsilon^2 [n \cdot \Delta x, m \cdot \Delta y] \, ,$$

where N is the number pixels in the image. If the irradiance is measured in $\frac{\mathrm{W}}{\mathrm{mm}^2}$, ϵ^2 will have units of $\left(\frac{\mathrm{W}}{\mathrm{mm}^2}\right)^2$, so it is common to evaluate the square root to compute the *root-mean-squared error* (RMS), which has the same dimensions as the error (or as the signal):

$$\text{RMS Error} = \sqrt{\epsilon^2} = \sqrt{\frac{1}{N}\sum_i \sum_j \epsilon^2\left[n \cdot \Delta x, m \cdot \Delta y\right]}.$$

It should be obvious that the RMS error obtained for one image using different quantizers will depend on the choice used, and that the RMS error from one quantizer will differ for different images. It should also be obvious that it is desirable to minimize the RMS error in an image. The brute-force method for minimizing quantization error is to add more bits to the ADC, which increases the cost of the quantizer and the memory required to store the image. There also is a limit to the effectiveness of such an approach, since the output from a quantizer with a very small step size will be noisy.

3.2.1 Signal-to-Noise Ratio

The signal-to-noise power ratio of an analog signal is most rigorously defined as the dimensionless ratio of the variances of the signal and noise:

$$SNR \equiv \frac{\sigma_f^2}{\sigma_n^2}$$

where the variance σ^2 of a signal is a measure of the spread of its amplitude about the mean value:

$$\sigma_f^2 = \int_{-\infty}^{+\infty} \left[f[x] - \langle f[x]\rangle\right]^2 dx \rightarrow \frac{1}{X_0}\int_{-\frac{X_0}{2}}^{+\frac{X_0}{2}} \left[f[x] - \langle f[x]\rangle\right]^2 dx$$

Thus a large *SNR* means that there is a larger variation of the signal amplitude than of the noise amplitude. The definition of *SNR* as the ratio of variances may have a large range of values – easily several orders of magnitude – and the numerical values may become unwieldy. The range of *SNR* may be compressed by expressing it on a logarithmic scale as a dimensionless quantity called a *bel*:

$$SNR = \log_{10}\left(\frac{\sigma_f^2}{\sigma_n^2}\right) = \log_{10}\left(\frac{\sigma_f}{\sigma_n}\right)^2 = 2\log_{10}\left(\frac{\sigma_f}{\sigma_n}\right) \quad [bels]$$

This definition of *SNR* is even more commonly expressed in units of tenths of a bel, or *decibels*, so that the integer value is more precise:

$$SNR = 10\ \log_{10}\left(\frac{\sigma_f^2}{\sigma_n^2}\right) = 20\ \log_{10}\left(\frac{\sigma_f}{\sigma_n}\right) \quad [decibels]$$

Under this definition, the SNR is 10 dB if the signal variance is ten times larger than the noise variance and 20 dB if the standard deviation is ten times larger than that of the noise.

The variances obviously depend on the independent statistics (specifically on the histograms) of the signal and noise. The variances do not depend on the "arrangement" of the gray levels (i.e., the numerical "order" or "pictorial" appearance of the pixel gray values) in the image. Since the noise often is determined by the measurement equipment, a single measurement of the noise variance often is used for many signal amplitudes. However, the signal variance must be measured each time. Consider a few examples of common 1-D signals.

3.2.2 Example: Variance of a Sinusoid

The variance of a sinusoid with amplitude A_0 is easily computed by direct integration:

$$\begin{aligned} f[x] &= A_0 \cos\left[2\pi \frac{x}{X_0}\right] \Longrightarrow \langle f[x]\rangle = 0 \\ \sigma_f^2 &= \frac{1}{X_0}\int_{-\frac{X_0}{2}}^{+\frac{X_0}{2}} \left[f[x] - \langle f[x]\rangle\right]^2 dx = \frac{1}{X_0}\int_{-\frac{X_0}{2}}^{+\frac{X_0}{2}} \left(A_0 \cos\left[2\pi\frac{x}{X_0}\right]\right)^2 dx \\ &= \frac{A_0^2}{X_0}\int_{-\frac{X_0}{2}}^{+\frac{X_0}{2}} \frac{1}{2}\left(1+\cos\left[4\pi\frac{x}{X_0}\right]\right) dx = \frac{A_0^2}{2X_0}(X_0+0) \end{aligned}$$

where the trigonometric identity

$$\cos^2[\theta] = \frac{1}{2}(1+\cos[2\theta])$$

has been used. The final result is:

$$\boxed{\sigma_f^2 = \frac{A_0^2}{2} \text{ for sinusoid with amplitude } A_0}$$

Note that the variance of the sinusoid does not depend on the period (i.e., on the spatial frequency) or on the initial phase – it is a function of the histogram of the values in a period and not of the "ordered" values. It also does not depend on any "bias" (additive constant) in the signal. The standard deviation of the sinusoid is just the square root of the variance:

$$\sigma_f = \frac{A_0}{\sqrt{2}} \text{ for sinusoid with amplitude } A_0$$

3.2.3 Example: Variance of a Square Wave:

The variance of a square wave also is easily evaluated by integration of the thresholded sinusoid:

$$f[x] = A_0 \; SGN\left[\cos\left[2\pi\frac{x}{X_0}\right]\right] \Longrightarrow \langle f[x]\rangle = 0$$

$$\sigma_f^2 = \frac{1}{X_0}\int_{-\frac{X_0}{2}}^{+\frac{X_0}{2}} \left[f[x] - \langle f[x]\rangle\right]^2 dx = \frac{1}{X_0}\left(\int_{-\frac{X_0}{4}}^{+\frac{X_0}{4}} (-A_0)^2\, dx + \int_{+\frac{X_0}{4}}^{+\frac{3X_0}{4}} (+A_0)^2\, dx\right)$$

$$= \frac{1}{X_0}\left(A_0^2\frac{X_0}{2} + A_0^2\frac{X_0}{2}\right) = A_0^2$$

$$\Longrightarrow \boxed{\sigma_f^2 = A_0^2 \text{ for square wave with amplitude } A_0}$$

$$\Longrightarrow \boxed{\sigma_f = A_0 \text{ for square wave with amplitude } A_0}$$

Note that:

$$(\sigma_f)_{\text{square wave, amplitude } A_0} > (\sigma_f)_{\text{sinusoid, amplitude } A_0}$$

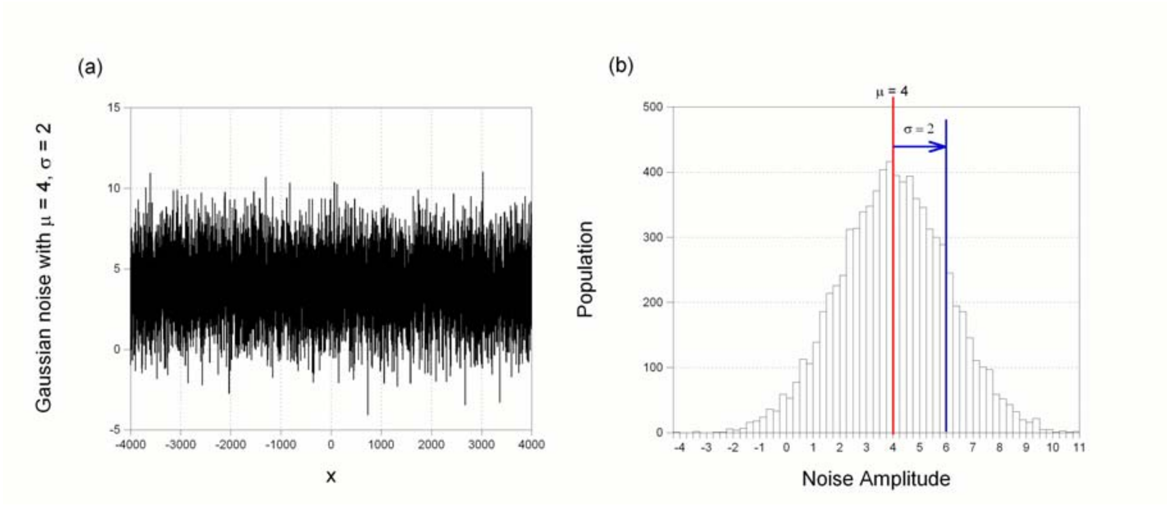
which makes intuitive sense, because the amplitude of the square wave is often more "distant" from its mean than the sinusoid is.

3.2.4 Variance of "Noise" from a Gaussian Distribution

A set of amplitudes selected at random from a probability distribution with a Gaussian "shape" is call Gaussian noise. The most common definition of the Gaussian distribution is:

$$p[n] = \frac{1}{\sqrt{2\pi\sigma^2}}\exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

where μ is the mean value of the distribution and σ^2 is the variance. This shows that the Gaussian distribution is completely specified by these two parameters. The standard deviation σ is a measure of the "width" of the distribution and so influences the range of output amplitudes.



One exemplar and the histogram of 8192 samples from the Gaussian distribution with mean value $\mu = 4$ and standard deviation $\sigma = 2$:

$$p[n] = \frac{1}{\sqrt{2\pi}} \exp\left[-\left(\frac{n-4}{2}\right)^2\right]$$

3.2.5 Approximations to *SNR*

Since the variance depends on the statistics of the signal, it is common (though less rigorous) to approximate the variance by the square of the dynamic range (peak-to-peak signal amplitude $f_{\max} - f_{\min} \equiv \Delta f$). In most cases, $(\Delta f)^2$ is larger (and often much larger) than σ_f^2. In the examples of the sinusoid and the square wave already considered, the approximations are:

$$\text{Sinusoid with amplitude } A_0 \Longrightarrow \sigma_f^2 = \frac{A_0^2}{2},\ \ (\Delta f)^2 = (2A_0)^2 = 4A_0^2 = 8 \cdot \sigma_f^2$$
$$\text{Square wave with amplitude } A_0 \Longrightarrow \sigma_f^2 = A_0^2,\ \ (\Delta f)^2 = (2A_0)^2 = 4A_0^2 = 4 \cdot \sigma_f^2$$

For Gaussian noise with variance $\sigma^2 = 1$ and mean μ, the dynamic range of the noise technically is infinite, but since few amplitudes exist that are outside of the interval of four standard deviations measured from the mean. This means that the extrema of the Gaussian distribution may be approximated by $f_{\max} \cong \mu + 4\sigma$, $f_{\min} \cong \mu - 4\sigma$, leading to $\Delta f \cong 8\sigma$. The estimate of the variance of the signal is then $(\Delta f)^2 \cong 64\sigma_f^2$, which is (obviously) 64 times larger than the actual variance. Because this estimate of the signal variance is too large, the estimates of the *SNR* thus obtained will be too optimistic.

Often, the signal and noise of images are measured by photoelectric detectors as differences in electrical potential in volts; the signal dynamic range is $V_f = V_{\max} - V_{\min}$, the average noise voltage is V_n, and the signal-to-noise ratio is:

$$SNR = 10 \log_{10}\left(\frac{V_f^2}{V_n^2}\right) = 20 \log_{10}\left(\frac{V_f}{V}\right) \quad [dB]$$

As an aside, we mention that the *signal amplitude* (or *level*) of analog electrical signals often is described in terms of dB measured relative to some fixed reference signal. If the reference level is $V = 1\,\text{V}$ (regardless of the impedance), then the signal level is measured in units of "dBV:"

$$level = 10\ \log_{10}\left(V_f^2\right)\ dBV = 20\ \log_{10}\left(V_f\right)\quad dBV$$

The level is measured relative to 1 mV (across an impedance of $75\,\Omega$) is in units of "dBmV:"

$$level = 10\ \log_{10}\left(\left(\frac{V_f}{10^{-3}\,\text{V}}\right)^2\right)\ dBV = 60\ \log_{10}\left(V_f\right)\quad dBmV$$

3.2.6 *SNR* of Quantization

We now finally get to the goal of this section: to determine the signal-to-noise ratio of quantization of a signal. Clearly the quantized signal exhibits an error from the original analog signal, which leads to the quantity of "noise" and thus a measure of SNR. . Though in a strict sense the input signal and the type of quantizer determine the probability density function of the quantization error, it usually is appropriate to assume that the error due to quantization is uniformly distributed, i.e., the probability density function is a rectangle. In the case of an m-bit uniform quantizer (2^m gray levels) where the levels are spaced by intervals of width b_0 over the full analog dynamic range of the signal, the error due to quantization will be (approximately) uniformly distributed over this interval b_0; this will be demonstrated in the following examples. If the nonlinearity of the quantizer is *rounding*, the mean value of the error is 0; if *truncation* to the next lower integer, the mean value is $-\frac{b_0}{2}$. Any book on probability and statistics likely will demonstrate that the variance and standard deviation of uniformly distributed noise over a range b_0 are:

$$\sigma_n^2 = \frac{b_0^2}{12}$$
$$\sigma_n = \sqrt{\frac{b_0^2}{12}} = \frac{b_0}{\sqrt{12}}$$

This also may be derived quite easily by evaluating the terms in the expression for the variance:

$$\sigma_n^2 = \left\langle n^2\right\rangle - \left\langle n\right\rangle^2$$

where the noise function is

$$p_n = \frac{1}{b_0} RECT\left[\frac{n - \frac{b_0}{2}}{b_0}\right]$$

This means that the mean value $\langle n \rangle = \frac{b_0}{2}$ and the mean value of n^2 is:

$$\begin{aligned}
\langle n^2 \rangle &= \int_{-\infty}^{+\infty} n^2 \cdot p\,[n]\;\;dn = \frac{1}{b_0} \int_{-\infty}^{+\infty} n^2 \cdot RECT \left[\frac{n - \frac{b_0}{2}}{b_0} \right] \;\; dn \\
&= \frac{1}{b_0} \int_0^{b_0} n^2 \;dn = \frac{1}{b_0} \cdot \left. \frac{n^3}{3} \right|_{n=0}^{n=b_0} = \frac{1}{b_0} \cdot \frac{b_0^3}{3} = \frac{b_0^2}{3}
\end{aligned}$$

So the variance is:

$$\begin{aligned}
\sigma_n^2 &= \langle n^2 \rangle - \langle n \rangle^2 \\
&= \frac{b_0^2}{3} - \left(\frac{b_0}{2} \right)^2 = b_0^2 \left(\frac{1}{3} - \frac{1}{4} \right) = \frac{b_0^2}{12} \quad QED
\end{aligned}$$

$$\boxed{\sigma_n^2 = \frac{b_0^2}{12} \textit{ for uniformly distributed noise over range } b_0}$$

For an m-bit quantizer and a signal with with maximum and minimum amplitudes $f_{\max}$ and $f_{\min}$ (and range $\Delta f = f_{\max} - f_{\min}$), the width b_0 of a quantization level (the step size) has the value:

$$b_0 = \frac{f_{\max} - f_{\min}}{2^m} \equiv \frac{\Delta f}{2^m} = (\Delta f) \cdot 2^{-m}$$

If we can assume that the quantization noise is uniformly distributed, then the variance of the quantization noise is well defined:

$$\sigma_n^2 = \frac{b_0^2}{12} = \frac{1}{12} \left(\frac{\Delta f}{2^m} \right)^2 = (\Delta f)^2 \cdot \left(12 \cdot 2^{2m}\right)^{-1}$$

The *SNR* is the ratio of the variance of the signal to that of the noise due to quantization:

$$\frac{\sigma_f^2}{\sigma_n^2} = \sigma_f^2 \cdot \frac{12 \cdot 2^{2m}}{(\Delta f)^2}$$

This may be expressed on a logarithmic scale to yield the (sort of) simple expression:

$$\begin{aligned}
SNR &= 10 \cdot \log_{10} \left[\sigma_f^2 \cdot 12 \cdot 2^{2m} \right] - 10 \cdot \log_{10} \left[(\Delta f)^2 \right] \\
&= 10 \cdot \log_{10} \left[\sigma_f^2 \right] + 10 \cdot \log_{10} [12] + 20 \cdot m \cdot \log_{10} [2] - 10 \cdot \log_{10} \left[(\Delta f)^2 \right] \\
&\cong 10 \cdot \log_{10} \left[\sigma_f^2 \right] + 10 \cdot 1.079 + 20 \cdot m \cdot 0.301 - 10 \cdot \log_{10} \left[(\Delta f)^2 \right] \\
&\cong 6.02 \cdot m + 10.8 + 10 \cdot \log_{10} \left[\left(\frac{\sigma_f^2}{(\Delta f)^2} \right) \right] \quad [dB]
\end{aligned}$$

The third term obviously depends on the parameters of *both* the signal and the quantizer. This equation demonstrates one result immediately – that the *SNR* of quantization increases by $\gtrapprox 6\ dB$ for every bit added to the quantizer (again, assuming uniform distribution of the noise). If using the (poor) estimate that $\sigma_f^2 = (\Delta f)^2$, then the third term evaluates to zero and the approximate *SNR* is:

$$\boxed{SNR \text{ for quantization to } m \text{ bits} \cong 6.02 \cdot m + 10.8 + 10 \cdot \log_{10}[1]) = 6.02 \cdot m + 10.8 \quad [dB]}$$

The statistics (and thus the variance) may be approximated for many types of signals (*e.g.*, music, speech, realistic images) as resulting from a random process. The histograms of these signals usually are peaked at or near the mean value μ and the probability of a gray level decreases for values away from the mean; the signal approximately is the output of a gaussian random process with variance σ_f^2. By selecting the dynamic range of the quantizer Δf to be sufficiently larger than σ_f, few (if any) levels should be clipped by the quantizer. For a Gaussian random process, virtually none of the values will be clipped if the the maximum and minimum levels of the quantizer are four standard deviations from the mean level:

$$f_{\max} - \mu_f = \mu_f - f_{\min} = \frac{\Delta f}{2} = 4 \cdot \sigma_f$$

In other words, we may choose the step size between levels of the quantizer to satisfy the criterion:

$$\Delta f = 8 \cdot \sigma_f \Longrightarrow \frac{\sigma_f^2}{(\Delta f)^2} = \frac{1}{64}$$

In this case, the *SNR* of the quantization process becomes:

$$\begin{aligned} SNR &= 6.02 \cdot m + 10.8 + 10 \cdot \log_{10}\left[\frac{1}{64}\right] \\ &= 6.02 \cdot m + 10.8 + 10 \cdot (-1.806) \\ &= 6.02 \cdot m - 7.26 \; [dB] \end{aligned}$$

which is $\cong$ 18 dB less than the (optimistic) estimate obtained by assuming that $\sigma_f^2 \cong (\Delta f)^2$.

This expression for the *SNR* of quantizing a Gaussian-distributed random signal with measured variance σ_f^2 may be demonstrated by quantizing that signal to m bits over the range $f_{\min} = \mu - 4\sigma_f$ to $f_{\max} = \mu + 4\sigma_f$, and computing the variance of the quantization error σ_n^2. The resulting *SNR* should satisfy the relation:

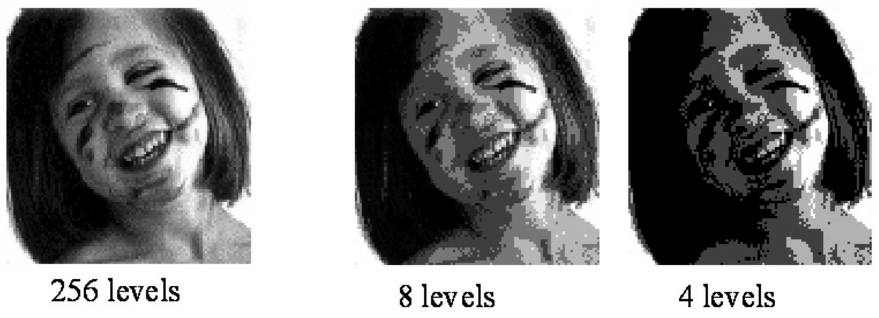
$$SNR = 10 \cdot \log_{10}\left[\frac{\sigma_f^2}{\sigma_n^2}\right] = (6.02 \cdot m - 7.26) \;\; dB$$

This equation may be applied to demonstrate that the *SNR* of a noise-free analog signal after quantizing to 8 bits is $\cong$ 41 dB. The *SNR* of the same signal quantized to 16 bits (common in CD players) is approximately 89 dB. The best *SNR* that can be obtained from analog recording (such as on magnetic tape) is about 65 dB, which is equivalent to that from a signal digitized to 12 bits per sample or 4096 levels.

The flip side of this problem is to determine the effective number of quantization bits after digitizing a noisy analog signal. This problem was investigated by Shannon in 1948. The analog signal is partly characterized by its bandwidth $\Delta\nu$ [Hz], which is the analog analogue of the concept of digital data rate [bits per second]. The bandwidth is the width of the region of support of the signal spectrum (its Fourier transform).

3.3 Quantization to Few Bits

If quantizing to a small number of bits, say 4 or fewer so that the number of gray values is 16 or fewer, then the image appearance often suffers markedly. This was visible in the example of bit depth shown earlier; the examples for 2 and 3 bits are repeated below. The "false contouring" shown was a common occurence in the "old days" when display capabilities were much more limited than currently. A special case is quantization to a single bit for two gray values (black and white); this is called *halftoning* from its original use to convert gray-scale images (photographs) to bitonal images for printing in newspapers or books.



False contouring becomes visible in an image with slowly varying gray values if quantized to few bits.

3.3.1 Improved Gray Scale (IGS) Quantization

The IGS quantization is a standard technique (considered in Gonzalez and Woods) that has a rather glorified name for a fairly simple concept. The process is well defined and reduces the effects of false contouring by adding a random number of size approximately equal to the quantization step size to the gray value of each pixel BEFORE quantization. The recipe for the process is:

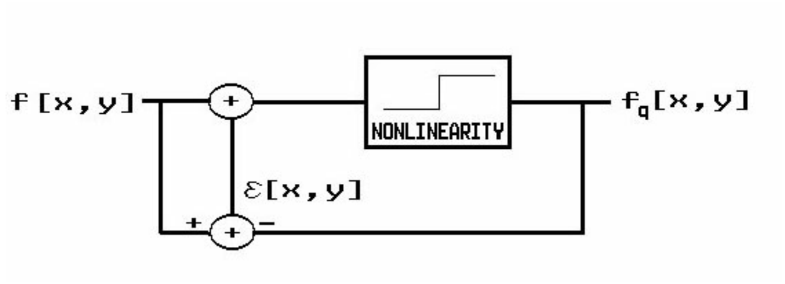
1. Set initial SUM to binary value $(0000\ 0000)_2$
2. If most signficant four bits of current pixel evaluate to 1111, then set the new sum to those four bits + 0000, otherwise set new sum with the four most significant bits plus the 4 least signficant bits of the old sum.

Example

Index	Gray	Binary Code	Sum of 8-bit binary and 4-bit error	4-bit Gray
$n-1$	N/A	0000_2	0000 0000	N/A
n	108	0110 1100_2	0110 1100 + 0000 0000 = (0110) (1100)	0110_2= 6.
$n+1$	139	1000 1011_2	1000 1011 + 0000 (1100) = (1001) (0111)	1001_2= 9.
$n+2$	135	1000 0111_2	1000 0111 + 0000 (0111) = (1000) (1110)	1000_2= 8.
$n+3$	244	1111 0100_2	1111 0100 + 0000 (1110) = (1111) (1111) + 0000 0011	1111_2= 15.
$n+4$	200	1100 1000_2	1100 1000 + 0000 (0011) = (1100) 1011	1100_2= 12.

3.3.2 Quantizers with Memory – Error Diffusion

Another way to reduce quantization error is to use a quantizer with memory, so that the quantized value at a pixel is determined in part by the quantization error at nearby pixels. A schematic diagram of the quantizer with memory is shown below:



Flow chart for "quantizer with memory" = error-diffusion quantization

A simple method for quantizing with memory that generally results in reduced total error without *a priori* knowledge of the statistics of the input image and without adding much additional complexity of computation was introduced by Floyd and Steinberg (**SID 36**, 1976) as a means to simulate gray level images on binary image displays and is known as error diffusion. It is easily adapted to multilevel image quantization. As indicated by the name, in error diffusion the quantization error is from one pixel is used to in the computation of the levels of succeeding pixels. In its simplest form, all quantization error at one pixel is subtracted from the gray level of the next pixel. The subsequent quantization of the next pixel therefore reduces the local quantization error.

1-D Error Diffusion of 1-D Functions

In the 1-D case, the quantization level at sample location x is the gray level of the sample minus the error $\epsilon[x-1]$ at the preceding pixel:

$$\begin{aligned} f_q[x] &= Q\{f[x] - \epsilon[x-1]\} \\ \epsilon[x] &= f[x] - f_q[x] \\ &= f[x] - Q\{f[x] - \epsilon[x-1]\} \end{aligned}$$

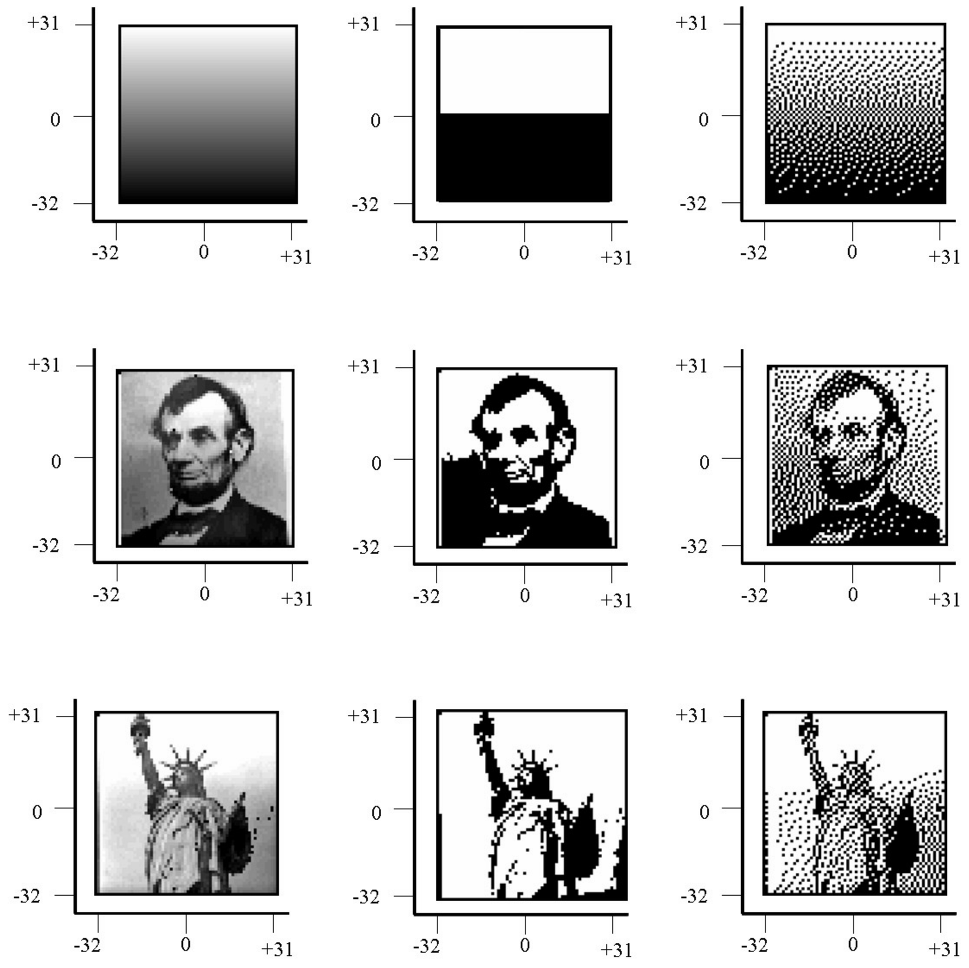
Usually, the error is reset to zero at the beginning of each line. You can see that this is simple to implement and the results are often very useful. Several years back, John Knapp (CIS undergraduate and graduate student) used 1-D error diffusion to improve the reconstructions of computer-generated holograms for several different applications and the results were often excellent (§23.4 in my book *Fourier Methods in Imaging*).

2-D Error Diffusion

In the 2-D case, it is possible (and often desirable) to divide and apply the resulting quantization error weight among different as-yet unquantized pixels. The "standard" Floyd-Steinberg algorithm quantizes the $N \times N$ image in a raster from the upper left corner (sequence of pixels: first row from left to right, then second row from left to right, etc. to the bottom line of pixels). At each pixel, the quantization error is evaluated and divided among four pixels yet to be quantized: $\frac{7}{16}$ of this result is added to the next pixel in line, $\frac{5}{16}$ to the pixel below, $\frac{3}{16}$ to the pixel below and "forward", and $\frac{1}{16}$ to the pixel below and "afterward." A figure is helpful:

	□	$\frac{7}{16}$
$\frac{3}{16}$	$\frac{5}{16}$	$\frac{1}{16}$

These errors are subtracted from the subsequent pixels, and the ensemble of results can significantly affect the quantized state of a pixel. The end result is an array of bitonal pixels (black and white) where the density of white pixels is proportional to the gray value. The examples demonstrate the effects of binary quantization on gray-level images. The images of the ramp demonstrate that why the binarizer with memory is often called *pulse-density modulation.* Note that more fine detail is preserved in binarized images from the quantizer with memory (e.g.,the sky in Liberty and the shoulder in Lincoln). This is accomplished by possibly enhancing the local binarization error.



2-D error-diffused quantization for three gray-scale images: (a) linear ramp, after quantizing at the midgray level, after Floyd-Steinberg error diffusion at the midgray level; (b) same sequence for "Lincoln"; (c) same sequency for "Liberty." The error-diffused images convey more information about the more raplidly varying structure in the scenes.

Error diffusion has applications beyond quantization – for example, Eschbach and Knox have used it to create algorithms for edge enhancement.

A discussion of the use of error diffusion in ADC was given by Anastassiou (**IEEE Trans. Circuits and Systems, 36**, 1175, 1989).

3.3.3 Halftoning

This is the process of converting a gray-scale image to a bitonal (1-bit) image. All processes do some kind of thresholding operation and depend on the limited resolution of the eye to "locally average" the resulting bits to give the illusion of gray scale. The "pattern dithering" processes are standard in newspaper reproduction, while the error-diffused dithering works well on electronic displays.

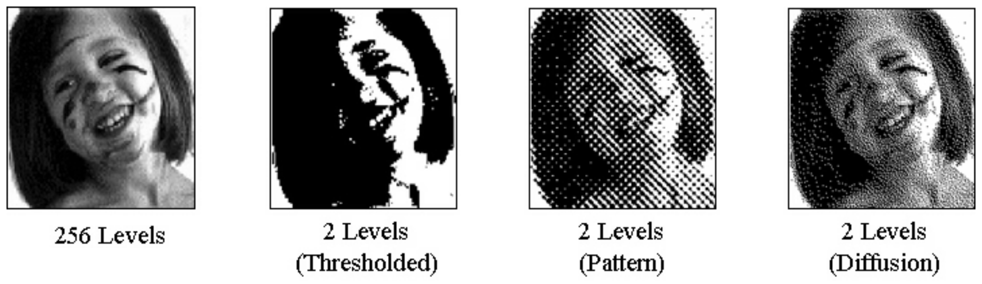


Figure 3.1: *Results of different halftoning algorithms: (a) original image; (b) after thresholding at 50% gray; (c) pattern "dither"; (d) error-diffused "dither."*

Chapter 4

Image Processing Operations

Once the image data has been sampled, quantized, and stored in the computer, the next task is change the image for whatever purpose, e.g., to extract additional information from the data, to make the image appear better. The various image processing operations $\mathcal{O}\{\ \}$ are applied to the digital input image $f_s[n \cdot \Delta x,\ m \cdot \Delta y]$ to obtain a (generally) different output $g_s[n' \cdot \Delta x, m' \cdot \Delta y]$. From this point on, all images may be considered as sampled data and thus the subscript s will be ignored and the coordinates will be labeled by $[x, y]$. The general operator has the form

$$\mathcal{O}\{f[n,m]\} = g[k,\ell]$$

though most often the input and output indices will be identical: $[k,\ell] = [n,m]$.

The various operators $\mathcal{O}$ can categorized based on the number and location of the pixels of the input image f that affect the computation of a particular output pixel $[x, y]$. One possible set of categories is:

1. **Geometrical Operations:** The gray value f at pixel $[n,m]$ is remapped to the new location $[k,\ell]$. This may be interpreted as a mapping of $f[n,m]$ to obtain $g[n,m] = f[k,\ell]$: this class of operator may be used to magnifiy, minify, rotate, or warp images and is essential in many disciplines, including cartography, medical imaging, ...

2. **Point Operations on single images:** The gray value of the output image g at a particular pixel $[n_0, m_0]$ depends ONLY on the gray value of the same pixel in f; examples of these operations include contrast stretching, segmentation based on gray value, and histogram equalization;

3. **Point Operations on multiple images:** The gray value of the output pixel $g[n_0, m_0]$ depends on the gray values of the same pixel in a set of input images $f[x_0, y_0, t_n]$ or $f[x_0, y_0, \lambda_n]$; examples are segmentation based on variations in time or color; multiple-frame averaging for noise smoothing, change detection, and spatial detector normalization;

4. **Neighborhood Operations on one image:** The gray value of g at a particular pixel $[n_0, m_0]$ depends on the gray values of pixels in the neighborhood

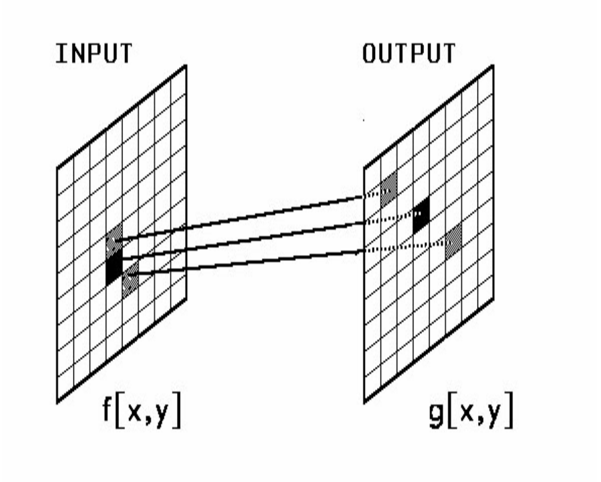
of of the same pixel in $f[n_0, m_0]$; examples include convolution (as for image smoothing or sharpening), and spatial feature detection (e.g., line, edge, and corner detection);

5. **Neighborhood Operations on multiple images:** This is just a generalization of (3); the pixel $g[n_0, m_0]$ depends on pixels in the spatial and temporal (or spectral) neighborhood of $[x_0, y_0, t_n$ or $\lambda_n]$. spatial / temporal convolution or spatial / spectral convolution

6. **Operations based on Object "Shape"** (e.g., "structural" or "morphological") operations: The gray level of the output pixel is determined by the object class to which a pixel belongs; examples include classification, segmentation, data compression, character recognition;

7. **"Global" Operations:** The gray value of the output image at a specific pixel depends on the gray values of all of the pixels of $f[n, m]$; these include image transformations, e.g., Fourier, Hartley, Hough, Haar, Radon transforms

4.1 Geometrical Operations



Schematic of geometrical operation; the gray values of pixels of $f[n, m]$ are "unchanged" (except for interpolation effects) by the operation, but the gray values are moved to different locations.

Geometrical operations change the spatial relationships between image pixels and may be used to correct distortions due to recording geometry, scale changes, rotations, perspective (keystoning), or due to curved or irregular object surfaces. In this section, we will define the procedures for specifying and implementing a range of geometrical operations.

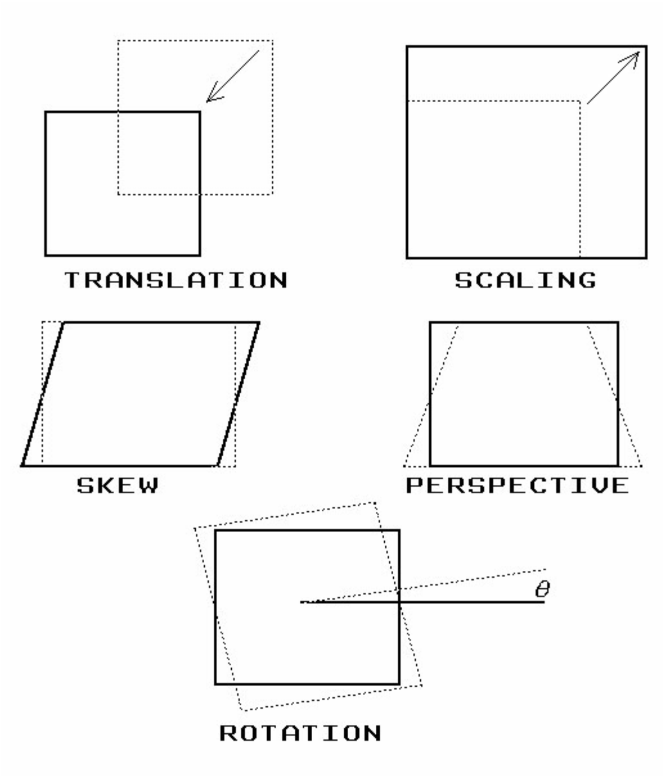
4.1.1 Common Geometrical Operations

$[x', y']$ are the coordinates of the input image that are mapped to $[x, y]$ in the output image.

$$x' = x, y' = y \implies g[x,y] = f[x',y'] = f[x,y] \implies \text{identity transformation}$$
$$x' = x + x_0, \quad y' = y + y_0 \implies g[x,y] = f[x + x_0, y + y_0] \implies \text{translation by } [x_0, y_0]$$
$$x' = ax, \quad y' = by \implies g[x,y] = f[M_x x, M_y y] \implies \text{spatial } \textit{scaling}$$
$$x' = -x, \quad y' = +y \implies g[x,y] = f[-x,+y] \implies \text{left-right } \textit{reversal}\text{, mirror image}$$
$$x' = -x, \quad y' = -y \implies g[x,y] = f[-x,-y] \implies \text{rotation by } 180^\circ$$
$$x' = x + \alpha y, \quad y' = y \implies g[x,y] = f[x + \alpha y, y] \implies \textit{skew}$$
$$x' = x + \alpha xy, \quad y' = y \implies g[x,y] = f[x + \alpha xy, y] \implies \text{perspective distortion}$$

$$\left\{\begin{array}{l} x' = \alpha[x,y] = x\cos\theta - y\sin\theta \\ y' = \beta[x,y] = x\sin\theta + y\cos\theta \end{array}\right\} \implies \text{rotation thru } \theta$$

$$\cos\pi = -1, \sin\pi = 0 \implies x' = -x, y' = -y \implies g[x,y] = f[-x,-y] \implies \text{rotation by } 180^\circ$$



Examples of useful geometrical operations.

We can think of two obvious cases, where the operation is specified at the outset (e.g., you need to rotate the image by $\theta = 30^\circ$) or where you need to determine the mathematical operation necessary to "warp" an "input" image to match a second image (the "reference").

In theory, it is possible (though exhausting!) to describe geometric transformations via a lookup table of input/output coordinates, i.e., the table would specify new output coordinates $[x', y']$ for each input location $[x, y]$. Equivalently, the coordinate lookup table could specify the input coordinates $[x, y]$ that map to a specific output pixel $[x', y']$. For example, the lookup table moves pixels upward and to the right by one in each direction is:

n	m	k	ℓ
0	0	$+1$	$+1$
-1	-1	0	0
1	2	2	3
$\vdots$	$\vdots$	$\vdots$	$\vdots$

For an $N \times M$ image, such a lookup table would contain $N \cdot M$ ordered pairs ($\Longrightarrow$ 262,144 pairs for a 512×512 image). Such a lookup table could specify any arbitrary geometric transform, i.e., input pixels in the same neighborhood could move to locations that are very far apart in the output image. Besides using large blocks of computer memory, coordinate lookup tables are more general than usually necessary.

In realistic applications, neighboring pixels in the input image will remain in close proximity in the output image, i.e., the coordinates of pixels in the same neighborhood will be transformed in similar manner (adjacent input pixels remain adjacent in the output image). Such coordinate mappings are also called *rubber-sheet transformations* or *image warping* and may be specified by a set of parametric equations that specify the output coordinates for a given input position. For example, the transformed coordinates x' and y' could be specified as a function of both the x and y coordinates by functions α and β:

$$x' = \alpha[x, y]$$
$$y' = \beta[x, y].$$

The gray level f at the location $[x, y]$ is transferred to the new coordinates $[x', y']$ to create the output image $f[x', y']$. This sequence of operations equivalent to defining a "new" image $g[x, y]$ in the same coordinates $[x, y]$ but with different gray levels; hence:

$$\mathcal{O}\{f[x, y]\} = g[x, y] = f[x', y'] = f[\alpha[x, y], \beta[x, y]].$$

4.1.2 Power Series for Coordinates

In nearly all imaging appliations, the output image preserves the arrangement of pixel gray values in neighborhoods, which means that the transformation of the continuous coordinates must be continuous, i.e., the derivatives of the functions $\alpha[x, y]$ and

$\beta[x,y]$ must be finite everywhere. Such a requirement is automatically satisfied if the functions have the forms of power series of the input coordinates with nonnegative powers:

$$x' = \alpha[x,y] \equiv \sum_{n=0}^{\infty}\sum_{m=0}^{\infty} a_{nm}x^n y^m = a_{00} + a_{10}x + a_{01}y + a_{11}xy + a_{20}x^2 + \cdots$$
$$y' = \beta[x,y] \equiv \sum_{n=0}^{\infty}\sum_{m=0}^{\infty} b_{nm}x^n y^m = b_{00} + b_{10}x + b_{01}y + b_{11}xy + b_{20}x^2 + \cdots$$

In theory, any geometrical operation may be specifed in this way. In practice, the infinite series must be truncated, and the upper limits determine the rage of possible transformations.

For example, if we select:

$$\begin{aligned} a_{10} &= b_{01} = 1 \\ a_{00} &= x_0 \\ b_{00} &= y_0 \\ a_{nm} &= b_{nm} = 0 \text{ otherwise} \end{aligned}$$

The operation reduces to:

$$\begin{aligned} x' &= x + x_0 \\ y' &= y + y_0 \end{aligned}$$

which is a translation of the origin from $[0,0]$ to $[x_0, y_0]$. If we select

$$\begin{aligned} a_{00} &= b_{00} = 0 \\ a_{10} &= M_x \\ b_{01} &= M_y \\ a_{nm} &= b_{nm} = 0 \text{ otherwise} \\ \Longrightarrow\ x' &= M_x \cdot x \\ y' &= M_y \cdot y \end{aligned}$$

then the operation is a magnification along the two axes by scale factors of M_x and M_y such that the origin is unchanged.

4.1.3 Affine Transformation

Under many conditions, only three terms are needed in each series:

$$\begin{aligned} x' &= a_{00} + a_{10}x + a_{01}y \\ y' &= b_{00} + b_{10}x + b_{01}y \end{aligned}$$

This defines an *affine* transformation and may be considered as a *linear transformation* (specified by the scale factors) followed by a *translation* (specified by the constants a_{00} and b_{00}). Note that the affine transformation does not describe the skew and perspective examples.

4.1.4 Bilinear Transformation – Pseudoinverse Calculation

If we add the fourth term in both series, i.e., a_{nm} and $b_{nm} = 0$ for $n, m \geq 2$, then we have a *bilinear* transformation:

$$\begin{aligned} x' &\cong a_{00} + a_{10}x + a_{01}y + a_{11}xy \\ y' &\cong b_{00} + b_{10}x + b_{01}y + b_{11}xy \end{aligned}$$

There are eight unknown coefficients in the transformation, and thus (at least) eight independent equations of $[x, y]$ and $[x', y']$ are needed to find a solution for the coefficients a_{nm} and b_{nm}. Knowledge of the coordinate transformation of the vertices of a quadrilateral is sufficient to find a solution for this transformation. In other words, knowledge of the mapping at (at least) four locations

$$\begin{aligned} [x_1, y_1] &\rightarrow [x'_1, y'_1] \\ [x_2, y_2] &\rightarrow [x'_2, y_2] \\ [x_3, y_3] &\rightarrow [x'_3, y'_3] \\ [x_4, y_4] &\rightarrow [x'_4, y_4] \end{aligned}$$

will allow calculation of the eight coefficients. These four locations are sometimes called *control points.*

The solution may be cast in matrix notation where the known inputs $[x_n, y_n]$ and outputs $[x'_n, y'_n]$ are arranged as column vectors in the matrices $\underline{\mathbf{X}}$ and $\underline{\mathbf{X}}'$, respectively, and the unknown coefficients a_{ij} and b_{ij} are the rows of the matrix $\underline{\mathbf{A}}$ in this expression:

$$\underline{\mathbf{A}} \text{ (unknown)} \bullet \underline{\mathbf{X}} \text{ (known)} = \underline{\mathbf{X}}' \text{ (known)}$$

$$\begin{bmatrix} a_{00} & a_{10} & a_{01} & a_{11} \\ b_{00} & b_{10} & b_{01} & b_{11} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \\ x_1y_1 & x_2y_2 & x_3y_3 & x_4y_4 \end{bmatrix} = \begin{bmatrix} x'_1 & x'_2 & x'_3 & x'_4 \\ y'_1 & y'_2 & y'_3 & y'_4 \end{bmatrix}$$

If the matrix $\underline{\mathbf{X}}$ is square (**and** if no three of the known points lie on the same straight line), then the inverse matrix $\underline{\mathbf{X}}^{-1}$ exists and may be found via a straightforward calculation:

$$(\underline{\mathbf{A}} \bullet \underline{\mathbf{X}} \bullet)\, \underline{\mathbf{X}}^{-1} = \underline{\mathbf{X}}' \bullet \underline{\mathbf{X}}^{-1}$$
$$\Longrightarrow \boxed{\underline{\mathbf{A}} = \underline{\mathbf{X}}' \bullet \underline{\mathbf{X}}^{-1}}$$

More control points may be used in a bilinear fit, thus making the problem "overdetermined" (more equations than unknowns). A unique solution of an overdetermined problem may not exist if there is uncertainty ("noise") in the data. Under such conditions, either a *least-squares solution* (which minimizes the total squared error, and thus the mean-squared error) may be calculated. Alternatively, it is possible to apply the control points locally to determine appropriate local transformations for different sections of the image. If the distortion cannot be adequately represented by a power series with eight coefficents, then more than four control points are required.

4.2 Least-Squares Solution for Affine Transformation

The procedure for computing the least-squares solution for the coefficients of a geometric transformation is quite easy in matrix notation. For example, consider the simpler affine transformation that adds a constant translation to the coordinate and applies a magnification in each orthogonal direction. The coordinate equations are:

$$x' = a_{00} + a_{10}x + a_{01}y$$
$$y' = b_{00} + b_{10}x + b_{01}y$$

where the coefficients a_{nm} and b_{nm} must be calculated. The system of equations has six unknown quantities and so requires six equations to obtain a solution. Since each control point (input-output coordinate pair) yields equations for both $x$and y, three control points are needed. If more control points (say five) are available and consistent, the extras may be ignored and the the matrix inverse computed as before. If the positions of the control points are uncertain, the equations will be inconsistent and the matrix inverse will not exist. Under these conditions, the additional control points will improve the estimate of the transformation. The computation of the coefficients which minimizes the squared error in the transformation is called the least-squares solution, and is easily computed using matrix algebra as a pseudoinverse.

Now consider the process if we have measured five pairs of known control points, the matrix transformation is composed of the (as-yet unknown) 2 row by 3 column matrix $\underline{\mathbf{A}}$, the known 3 row by 5 column matrix $\underline{\mathbf{X}}$ of input locations, and the 2 row by 5 column matrix $\underline{\mathbf{X}}'$ of measured output locations of the control points:

$$\underline{\mathbf{A}} \bullet \underline{\mathbf{X}} = \underline{\mathbf{X}}'$$

$$\begin{bmatrix} a_{00} & a_{10} & a_{01} \\ b_{00} & b_{10} & b_{01} \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & x_5 \\ y_1 & y_2 & y_3 & y_4 & y_5 \end{bmatrix} = \begin{bmatrix} x_1' & x_2' & x_3' & x_4' & x_5' \\ y_1' & y_2' & y_3' & y_4' & y_5' \end{bmatrix}$$

If $\underline{\mathbf{X}}$ were square (and invertible), we would compute $\underline{\mathbf{X}}^{-1}$to find $\underline{\mathbf{A}}$ via:

$$\underline{\mathbf{A}} = \underline{\mathbf{X}}' \bullet \underline{\mathbf{X}}^{-1}$$

However, $\underline{\mathbf{X}}$ is NOT square in this case, and thus its inverse $\underline{\mathbf{X}}^{-1}$ does not exist. We may evaluate a *pseudoinverse* of $\underline{\mathbf{X}}$ that implements a least-squares solution for $\underline{\mathbf{A}}$ via the following steps for the the general case where $\underline{\mathbf{X}}$ has p rows and q columns ($p = 3$ and $q = 5$ in the example above).

1. multiply both sides of the equation from the right by the *transpose* matrix $\underline{\mathbf{X}}^T$, which is obtained from $\underline{\mathbf{X}}$ by exchanging the rows and columns to obtain a matrix with q rows and p columns. The result is:

$$(\underline{\mathbf{A}} \bullet \underline{\mathbf{X}}) \bullet \underline{\mathbf{X}}^T = \underline{\mathbf{X}}' \bullet \underline{\mathbf{X}}^T$$

2. The associativity of vector multiplication allows the second and third matrices on the left-hand side to be multiplied first:

$$(\underline{\mathbf{A}} \bullet \underline{\mathbf{X}}) \bullet \underline{\mathbf{X}}^T = \underline{\mathbf{A}} \bullet \left(\underline{\mathbf{X}} \bullet \underline{\mathbf{X}}^T\right)$$

3. The matrix $\underline{\mathbf{X}} \bullet \underline{\mathbf{X}}^T$ is $p \times p$ square. In the example above, the product of the two matrices is 3×3 square:

$$\underline{\mathbf{X}} \bullet \underline{\mathbf{X}}^T = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & x_5 \\ y_1 & y_2 & y_3 & y_4 & y_5 \end{bmatrix} \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \\ 1 & x_4 & y_4 \\ 1 & x_5 & y_5 \end{bmatrix}$$

$$= \begin{bmatrix} 5 & x_1+x_2+x_3+x_4+x_5 & y_1+y_2+y_3+y_4+y_5 \\ x_1+x_2+x_3+x_4+x_5 & x_1^2+x_2^2+x_3^2+x_4^2+x_5^2 & x_1y_1+x_2y_2+x_3y_3+x_4y_4+x_5y_5 \\ y_1+y_2+y_3+y_4+y_5 & x_1y_1+x_2y_2+x_3y_3+x_4y_4+x_5y_5 & y_1^2+y_2^2+y_3^2+y_4^2+y_5^2 \end{bmatrix}$$

$$= \begin{bmatrix} 5 & \sum_{n=1}^{5} x_n & \sum_{n=1}^{5} y_n \\ \sum_{n=1}^{5} x_n & \sum_{n=1}^{5} x_n^2 & \sum_{n=1}^{5} x_n y_n \\ \sum_{n=1}^{5} y_n & \sum_{n=1}^{5} x_n y_n & \sum_{n=1}^{5} y_n^2 \end{bmatrix}$$

Since $\underline{\mathbf{X}} \bullet \underline{\mathbf{X}}^T$ is square, there is some chance that its inverse $\left(\underline{\mathbf{X}} \bullet \underline{\mathbf{X}}^T\right)^{-1}$ exists. If this is the case, then we can multiply the left-hand side of the equation by this inverse from the right; the result yields the desired coefficients $a_{k\ell}$ and $b_{k\ell}$ within the matrix $\underline{\mathbf{A}}$:

$$\underline{\mathbf{A}} \bullet \left(\underline{\mathbf{X}} \bullet \underline{\mathbf{X}}^T\right) \bullet \left(\underline{\mathbf{X}} \bullet \underline{\mathbf{X}}^T\right)^{-1} = \underline{\mathbf{A}}$$

4. If we perform the same series of steps on the right-hand side, we obtain the desired formula for the *pseudoinverse* $\underline{\mathbf{X}}^\dagger$

$$\underline{\mathbf{X}}^\dagger \equiv \underline{\mathbf{X}}^T \bullet \left(\underline{\mathbf{X}} \bullet \underline{\mathbf{X}}^T\right)^{-1} \Longrightarrow \boxed{\underline{\mathbf{A}} = \underline{\mathbf{X}}' \bullet \left(\underline{\mathbf{X}}^T \bullet \left(\underline{\mathbf{X}} \bullet \underline{\mathbf{X}}^T\right)^{-1}\right) \equiv \underline{\mathbf{X}}' \bullet \underline{\mathbf{X}}^\dagger}$$

The expression for the affine transformation obtained from five control points is:

$$\begin{bmatrix} a_{00} & a_{10} & a_{01} \\ b_{00} & b_{10} & b_{01} \end{bmatrix} = \begin{bmatrix} x'_1 & x'_2 & x'_3 & x'_4 & x'_5 \\ y'_1 & y'_2 & y'_3 & y'_4 & y'_5 \end{bmatrix} \bullet \left(\begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \\ 1 & x_4 & y_4 \\ 1 & x_5 & y_5 \end{bmatrix} \bullet \left(\begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 & x_5 \\ y_1 & y_2 & y_3 & y_4 & y_5 \end{bmatrix} \bullet \begin{bmatrix} 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \\ 1 & x_3 & y_3 \\ 1 & x_4 & y_4 \\ 1 & x_5 & y_5 \end{bmatrix} \right)^{-1} \right)$$

Example: Exact Control Points

Consider an example where we select four control points in a square that are mapped exactly without error

$$[x_1, y_1] = [0, 0] \rightarrow [x'_1, y'_1] = [0, 0]$$
$$[x_2, y_2] = [100, 0] \rightarrow [x'_2, y'_2] = [100, 100]$$
$$[x_3, y_3] = [100, 100] \rightarrow [x'_3, y'_3] = [0, 200]$$
$$[x_3, y_3] = [0, 100] \rightarrow [x'_3, y'_3] = [-100, 100]$$

$$\underline{\mathbf{X}} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 100 & 100 & 0 \\ 0 & 0 & 100 & 100 \end{bmatrix}$$

$$\underline{\mathbf{X}}' = \begin{bmatrix} x'_1 & x'_2 & x'_3 & x'_4 \\ y'_1 & y'_2 & y'_3 & y'_4 \end{bmatrix} = \begin{bmatrix} 0 & 100 & 0 & -100 \\ 0 & 100 & 200 & 100 \end{bmatrix}$$

$$\left(\underline{\mathbf{X}} \bullet \underline{\mathbf{X}}^T\right) = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 100 & 100 & 0 \\ 0 & 0 & 100 & 100 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 1 & 100 & 0 \\ 1 & 100 & 100 \\ 1 & 0 & 100 \end{bmatrix} = \begin{bmatrix} 4 & 200 & 200 \\ 200 & 20\,000 & 10\,000 \\ 200 & 10\,000 & 20\,000 \end{bmatrix}$$

$$\left(\underline{\mathbf{X}} \bullet \underline{\mathbf{X}}^T\right)^{-1} = \begin{bmatrix} \frac{3}{4} & -\frac{1}{200} & -\frac{1}{200} \\ -\frac{1}{200} & \frac{1}{10\,000} & 0 \\ -\frac{1}{200} & 0 & \frac{1}{10\,000} \end{bmatrix}$$

$$\underline{\mathbf{X}}^\dagger \equiv \underline{\mathbf{X}}^T \bullet \left(\underline{\mathbf{X}} \bullet \underline{\mathbf{X}}^T\right)^{-1} = \begin{bmatrix} 1 & 0 & 0 \\ 1 & 100 & 0 \\ 1 & 100 & 100 \\ 1 & 0 & 100 \end{bmatrix} \cdot \begin{bmatrix} \frac{3}{4} & -\frac{1}{200} & -\frac{1}{200} \\ -\frac{1}{200} & \frac{1}{10\,000} & 0 \\ -\frac{1}{200} & 0 & \frac{1}{10\,000} \end{bmatrix} = \begin{bmatrix} \frac{3}{4} & -\frac{1}{200} & -\frac{1}{200} \\ \frac{1}{4} & \frac{1}{200} & -\frac{1}{200} \\ -\frac{1}{4} & \frac{1}{200} & \frac{1}{200} \\ \frac{1}{4} & -\frac{1}{200} & \frac{1}{200} \end{bmatrix}$$

:

$$\underline{\mathbf{A}} = \begin{bmatrix} a_{00} & a_{10} & a_{01} \\ b_{00} & b_{10} & b_{01} \end{bmatrix} = \underline{\mathbf{X}}' \bullet \left(\underline{\mathbf{X}}^T \bullet \left(\underline{\mathbf{X}} \bullet \underline{\mathbf{X}}^T\right)^{-1}\right)$$

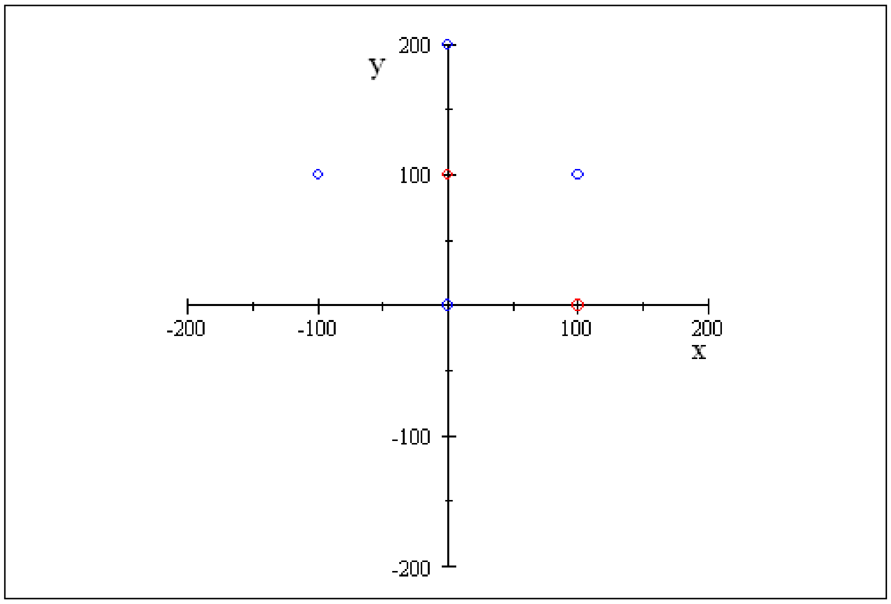
$$= \begin{bmatrix} 0 & 100 & 0 & -100 \\ 0 & 100 & 200 & 100 \end{bmatrix} \cdot \begin{bmatrix} \frac{3}{4} & -\frac{1}{200} & -\frac{1}{200} \\ \frac{1}{4} & \frac{1}{200} & -\frac{1}{200} \\ -\frac{1}{4} & \frac{1}{200} & \frac{1}{200} \\ \frac{1}{4} & -\frac{1}{200} & \frac{1}{200} \end{bmatrix} = \begin{bmatrix} 0 & 1 & -1 \\ 0 & 1 & 1 \end{bmatrix}$$

$$\Longrightarrow a_{00} = b_{00} = 0, a_{10} = b_{10} = b_{01} = 1, a_{01} = -1$$

This is a rotation and a scaling, which we can see by factoring out a constant:

$$\begin{bmatrix} a_{10} & a_{01} \\ b_{10} & b_{01} \end{bmatrix} = \begin{bmatrix} 1 & -1 \\ 1 & 1 \end{bmatrix} = \sqrt{2} \cdot \begin{bmatrix} \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \end{bmatrix} = \sqrt{2} \cdot \begin{bmatrix} \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \\ +\sin\frac{\pi}{4} & \cos\frac{\pi}{4} \end{bmatrix}$$

So the image is scaled (magnified) by $\sqrt{2}$ and rotated by $\theta = +\frac{\pi}{4} = 45°$.



$$\begin{aligned}
&\textit{Check}:\\
&\quad x' = a_{00} + a_{10}x + a_{01}y\\
&\quad y' = b_{00} + b_{10}x + b_{01}y\\
[0,0] &\to [0,0]\\
[100,0] &\implies 100 \to 0 + 1\cdot 100 + (-1)\cdot 0 = 100, 0\\
&\to 0 + 1\cdot 100 + 1\cdot 0 = 100 \implies [x_2', y_2'] = [100,100]\\
[100,100] &\implies 100 \to 0 + 1\cdot 100 + (-1)\cdot 100 = 0, 100\\
&\to 0 + 1\cdot 100 + 1\cdot 100 = 200 \implies [x_3', y_3'] = [0,200]\\
[0,100] &\implies 100 \to 0 + 1\cdot 0 + (-1)\cdot 100 = -100, 100\\
&\to 0 + 1\cdot 0 + 1\cdot 100 = 100 \implies [x_4', y_4'] = [-100,100]
\end{aligned}$$

Example: Control Points with Small Error

If the four control points are not exactly located, we obtain an approximate solution with the smallest least-squares error. The same input points are mapped to incorrect output points

$$\begin{aligned}
[x_1, y_1] &= [0,0] \to [x_1', y_1'] = [1,-1]\\
[x_2, y_2] &= [100,0] \to [x_2', y_2'] = [95,98]\\
[x_3, y_3] &= [100,100] \to [x_3', y_3'] = [2,196]\\
[x_3, y_3] &= [0,100] \to [x_3', y_3'] = [-98,99]
\end{aligned}$$

$$\underline{\mathbf{X}} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ x_1 & x_2 & x_3 & x_4 \\ y_1 & y_2 & y_3 & y_4 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 100 & 100 & 0 \\ 0 & 0 & 100 & 100 \end{bmatrix}$$

$$\underline{\mathbf{X}}' = \begin{bmatrix} x'_1 & x'_2 & x'_3 & x'_4 \\ y'_1 & y'_2 & y'_3 & y'_4 \end{bmatrix} = \begin{bmatrix} 1 & 95 & 2 & -98 \\ -1 & 98 & 196 & 99 \end{bmatrix}$$

$$\underline{\mathbf{X}}^\dagger \equiv \underline{\mathbf{X}}^T \bullet \left(\underline{\mathbf{X}} \bullet \underline{\mathbf{X}}^T\right)^{-1} = \begin{bmatrix} \frac{3}{4} & -\frac{1}{200} & -\frac{1}{200} \\ \frac{1}{4} & \frac{1}{200} & -\frac{1}{200} \\ -\frac{1}{4} & \frac{1}{200} & \frac{1}{200} \\ \frac{1}{4} & -\frac{1}{200} & \frac{1}{200} \end{bmatrix}$$

:

$$\underline{\mathbf{A}} = \begin{bmatrix} a_{00} & a_{10} & a_{01} \\ b_{00} & b_{10} & b_{01} \end{bmatrix} = \underline{\mathbf{X}}' \bullet \left(\underline{\mathbf{X}}^T \bullet \left(\underline{\mathbf{X}} \bullet \underline{\mathbf{X}}^T\right)^{-1}\right)$$

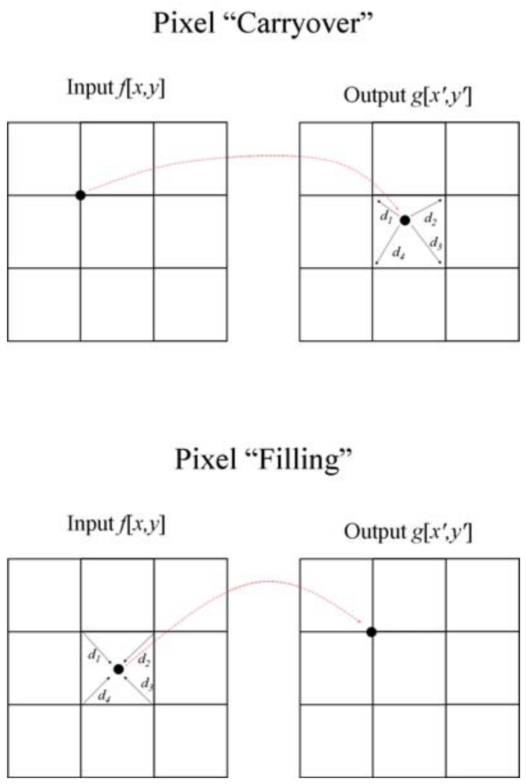
$$= \begin{bmatrix} 1 & 95 & 2 & -98 \\ -1 & 98 & 196 & 99 \end{bmatrix} \cdot \begin{bmatrix} \frac{3}{4} & -\frac{1}{200} & -\frac{1}{200} \\ \frac{1}{4} & \frac{1}{200} & -\frac{1}{200} \\ -\frac{1}{4} & \frac{1}{200} & \frac{1}{200} \\ \frac{1}{4} & -\frac{1}{200} & \frac{1}{200} \end{bmatrix} = \begin{bmatrix} -\frac{1}{2} & \frac{97}{100} & -\frac{24}{25} \\ -\frac{1}{2} & \frac{49}{50} & \frac{99}{100} \end{bmatrix}$$

$$\Longrightarrow a_{00} = b_{00} = -\frac{1}{2}, a_{10} = \frac{97}{100}, a_{01} = -\frac{24}{25}, b_{10} = \frac{49}{50}, b_{01} = \frac{99}{100}$$

This is indicates that we need to add a translation of the function by $-\frac{1}{2}$ along each direction.

4.3 Pixel Transfers

As already mentioned, a geometrical transformation may be implemented by specifying the where each pixel of the input image is located in the output grid, or by specifying the location of each output pixel on the input grid. Except for certain (and usually uninteresting) transformations, pixels of the input image will rarely map exactly to pixels of the output grid. It is therefore necessary to interpolate the gray value from pixels with noninteger coordinates (i.e., nongrid points) to pixels with integer coordinates that lie upon the grid. The method that transfers the input pixel coordinates and interpolates the gray value on the output grid is called *pixel carry-over* by K.R. Castleman in his book *Digital Image Processing.* It may also be called an *input-to-output* mapping. The algorithm that locates the output pixel upon the input grid and interpolates the gray value of the input pixels is called *pixel filling* or an *output-to-input* mapping.



Interpolation of pixel gray value in geometrical transformation: (a) pixel "carryover," where the gray value of a single input pixel is interpolated in the output array; (b) pixel "filling," where the interpolation is performed at the transformed location of the output pixel in the input array.

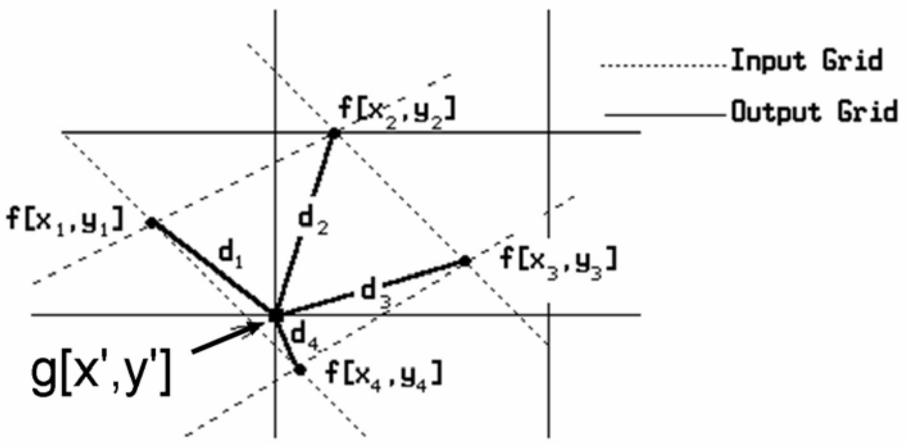
4.4 Pixel Interpolation

Gray-value interpolation is based on the relative distances of the nearest neighbors to or from the geometrically transformed pixel. In the simplest case of *nearest-neighbor* or *zeroth-order* interpolation), all of the gray value is transferred to or from the nearest pixel. However, since the distances of the four neighbors must be evaluated to determine which is closest, it generally is quite easy to "upgrade" nearest-neighbor calculations to *bilinear* interpolation. This method divides the gray level of the non-integer pixel among its four nearest "integer" neighbors in inverse proportion to the distance; if the transferred pixel is equidistant from the four neighbors, then its gray value is divided equally, if it is much closer to one of the four grid points, most of its gray value is transferred to that pixel. For pixel filling, the gray value at the output pixel $g[x', y']$ is determined from the following equation, where x_n, y_n are the coordinates of the nearest pixel of the input grid to the transformed pixel, and d_n are the respective distances:

$$g[x',y'] = \begin{cases} \dfrac{\sum_{n=1}^{4} \dfrac{f[x_n,y_n]}{d_n}}{\sum_{n=1}^{4} \dfrac{1}{d_n}} & \text{if all } d_n > 0 \\ f[x_n,y_n] & \text{if} \quad d_n = 0 \end{cases}$$

$$= \frac{\frac{1}{d_1} f[x_1,y_1] + \frac{1}{d_2} f[x_2,y_2] + \frac{1}{d_3} f[x_3,y_3] + \frac{1}{d_4} f[x_4,y_4]}{\frac{1}{d_1} + \frac{1}{d_2} + \frac{1}{d_3} + \frac{1}{d_4}}$$

Note that if the transformed location is identical to a pixel site, the distance to that site is $d = 0$ and the entire gray value is assigned to that location.



Schematic of interpolation of pixel gray value. The input and output grids of pixels are shown as solid and dashed lines, respectively. The distances of the output pixel from the four nearest neighbors are labeled $d_1 - d_4$.

The gray level of the transformed pixel can be divided among more of the neighboring pixels by using higher-order interpolation, e.g., cubic spline, etc.

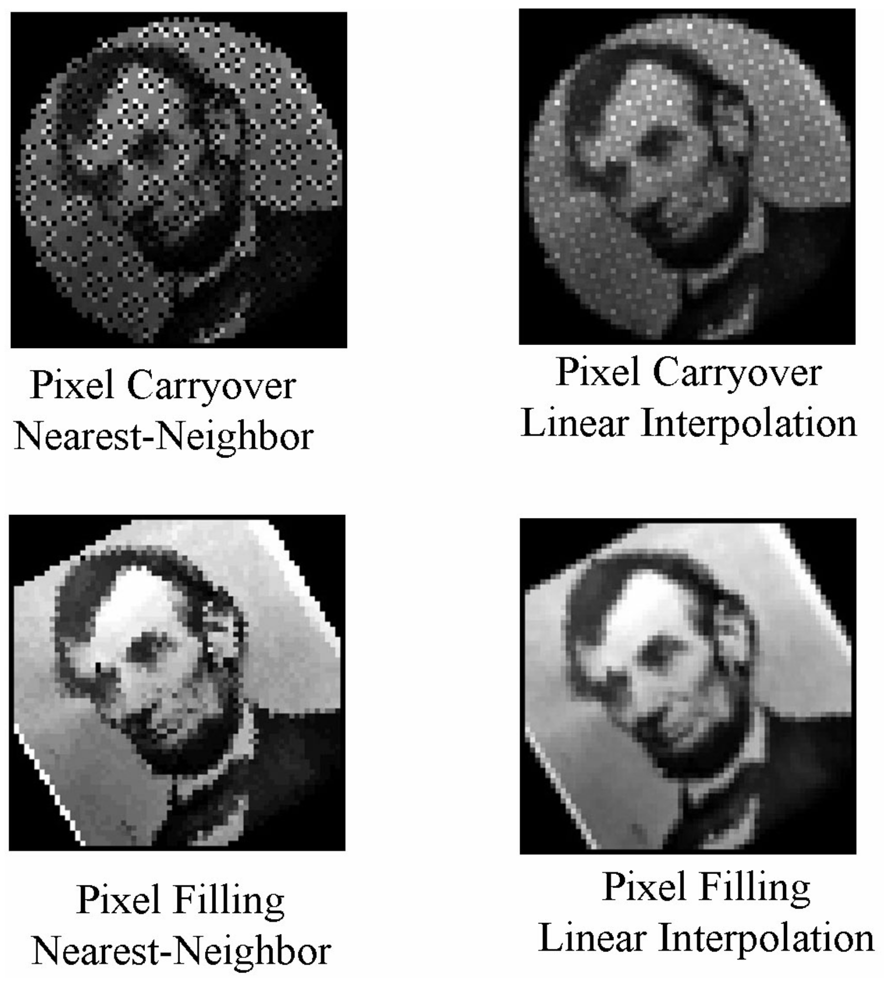


Figure 4.1: *The outputs obtained by rotating the image of Lincoln by* $\theta = 30^\circ$ *using four different pixel mappings: the first row are from pixel carryover (input to output) with* 0^{th}*-order interpolation ("nearest neighbor") and* 1^{st}*-order ("linear"); the second row are from pixel filling (output to input) with* 0^{th} *and* 1^{st}*-order interpolation. Note that the linear interpolation slightly "blurs" the image.*

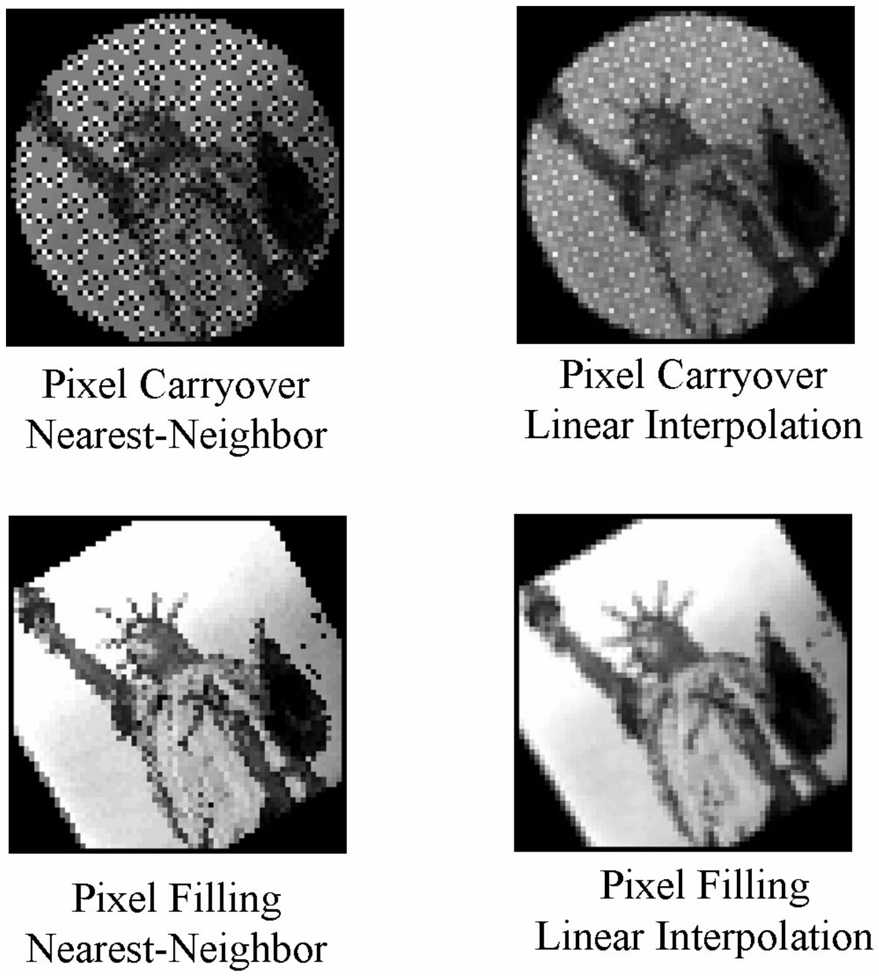

Pixel Carryover Nearest-Neighbor
Pixel Carryover Linear Interpolation
Pixel Filling Nearest-Neighbor
Pixel Filling Linear Interpolation

Figure 4.2: *Same operations applied to "Liberty"*

Chapter 5

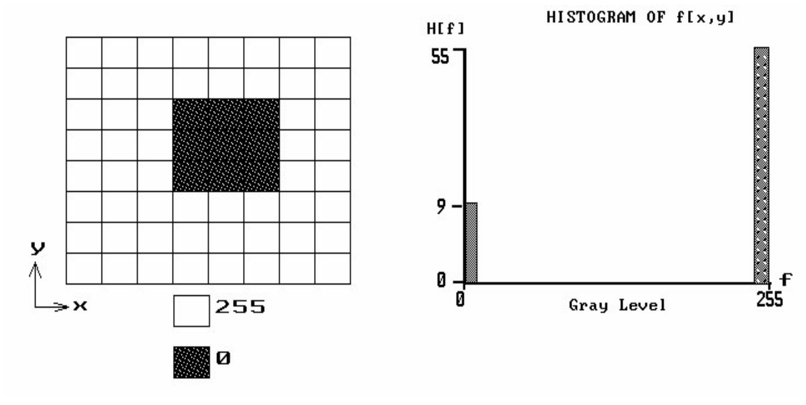
Point Operations

The gray value of each pixel in the output image $g[x,y]$ depends on the gray value of only the corresponding pixel of the input image $f[x,y]$. Every pixel of $f[x,y]$ with the same gray level maps to a single (usually different) gray value in the output image.

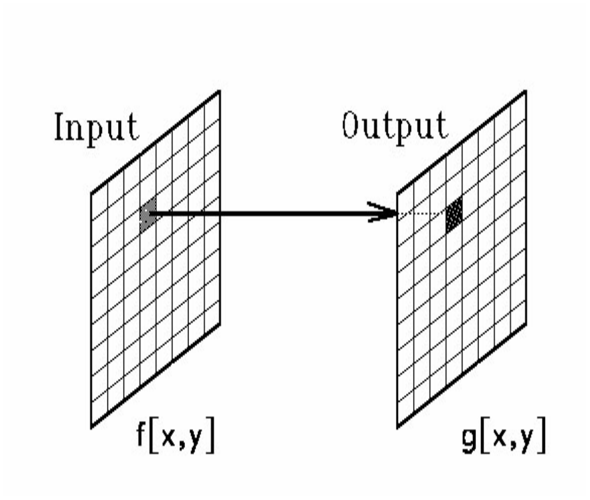
In a point operation, the only available parameter to be processed is the gray value of that one pixel. Therefore, the action of a point operation affects all pixels with the same input gray level f_0 the same; they are changed to the same output gray value g_0. When designing the action of the point operation, it often is very useful to know the pixel population H as a function of gray level f; such a plot is the histogram $H[f]$ of the image. The amplitude of the histogram at gray value f is proportional to the probability of occurrence of that gray value.

5.1 Image Histograms

The histogram of the 2-D image $f[x,y]$ plots the population of pixels with each gray level f.



The histogram may be represented as a 1-D function $H[f]$, where the independent variable is the gray value f and the dependent variable is the number of pixels H with that level. The histogram depicts a particular feature of the image: the population



of gray levels. It represents the simplest *feature* of the image, where a feature is a metric of some useful distinguishing characteristic of the image. The histogram may be called a *feature space* and it is possible to construct image processing operations to *segment* image pixels into component groups based on that feature.

The histogram often contains valuable global information about the image. For example, most pixels in a low-contrast image are contained in a few gray levels over a narrow range, so the histogram is concentrated within that small range of gray levels.. An image with a bimodal histogram (the histogram of gray values exhibits two modes $\Longrightarrow$ a histogram with two "peaks") often consists of a foreground object (whose pixels are concentrated around a single average gray value) on top of a background object whose pixels have a different average gray value.

Because all pixels in the image must have some gray value in the allowed range, the sum of populations of the histogram bins must equal the total number of image pixels N:

$$\sum_{f=0}^{f_{\max}} H[f] = N$$

where $f_{\max}$ is the maximum gray value ($f_{\max} = 255$ for an 8-bit quantizer). The histogram function is a scaled replica of the probability distribution function of gray levels in that image. The discrete probability distribution function $p[f]$ must satisfy the constraint:

$$\sum_{f=0}^{f_{\max}} p[f] = 1$$

and therefore the probability distribution and the histogram are related by the simple expression:

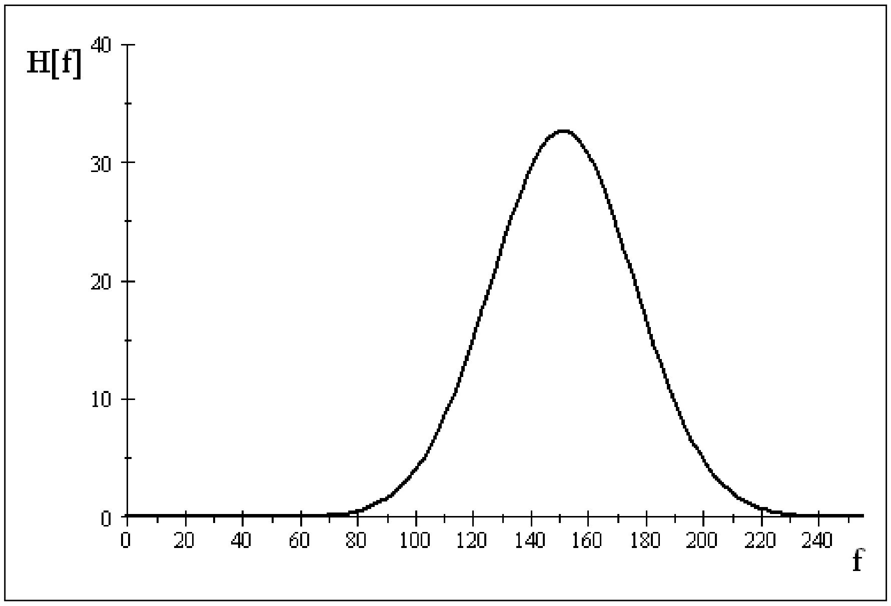
$$p[f] = \frac{1}{N}\, H[f]$$

5.1.1 Histograms of Typical Images

In realistic images, the gray values of an object are not identical but are "spread" about some mean value. The shape of the histogram for a realistic case follows the familiar "bell curve" or Gaussian distribution. The mathematical expression for a Gaussian distribution of gray values is an expression of the probability of occurence of gray value f where the mean value (and also most likely) is specified by μ. The "width" of a cluster may be quantified by metrics such as the *full width at half maximum* (FWHM) or *half width at half maximum* (HWHM), which are easy to measure but less useful than a true statistical measure. A more appropriate metric of cluster width is *standard deviation* σ (or its square σ^2, which is called the *variance*). The Gaussian probability has the form:

$$p[f] = \sqrt{\frac{1}{2\pi\sigma^2}} \cdot \exp\left[-\frac{(f-\mu)^2}{2\sigma^2}\right]$$

The factor $(f-\mu)^2$ in the exponent indicates that the Gaussian curve is symmetric about its mean value. Consider an example of gray values that follow a Gaussian distribution centered about a mean value $\mu = 155$ with standard deviation $\sigma = 25$ for a total number of pixels $N = 2048$ The histogram of this theoretical case indicates that gray values near the mean value (e.g., for $|f-\mu| \lessapprox 50 = 2 \cdot \sigma$) occur most frequently, while those far from the mean (e.g., $|f-\mu| \gtrapprox 3 \cdot \sigma$) are much less likely.



Simulation of histogram of a single object class with gray value centered at $f = 155$. The histogram has a Gaussian shape with standard deviation $\sigma = 25$.

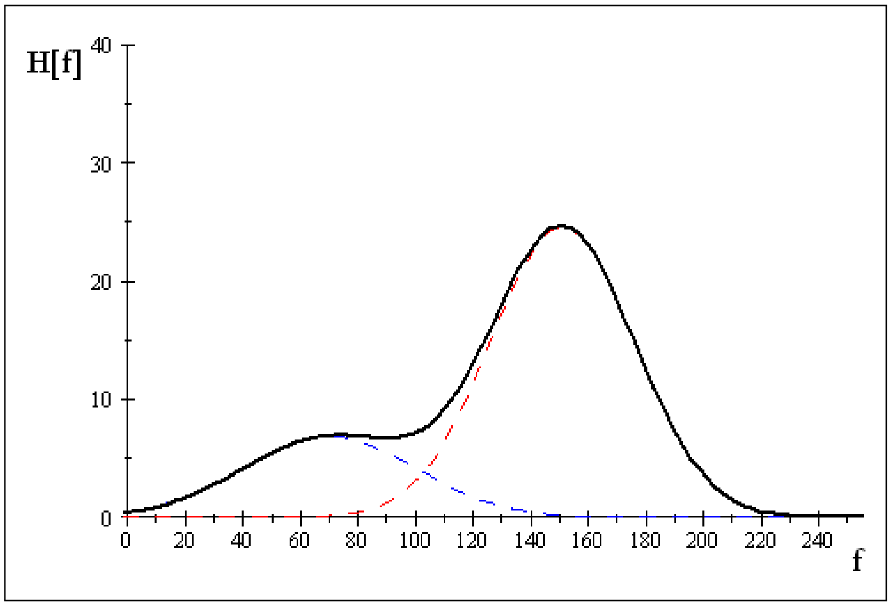
Since every pixel in the image takes on one of the allowed gray values, the summation of the histogram values must equal the number of pixels in the image:

$$\sum_{f=0}^{f_{\max}} H[f] = N$$

In real images with more than one object, the histograms tend to include multiple Gaussian curves added together. Consider a theoretical example that simulates the histogram of the image of Greek text. The histogram is composed of two Gaussian clusters with means and standard deviations:

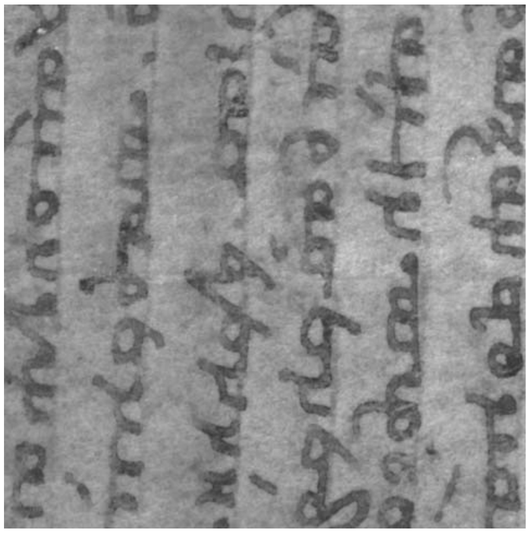
$$N_1 = 1536 : \mu_1 = 151, \ \sigma_1 = 25$$
$$N_2 = 512 : \mu_2 = 70, \ \sigma_2 = 30$$

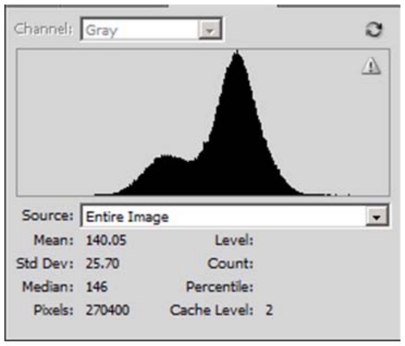
so that the cluster with the larger mean has three times as many pixels. The histogram of the individual clusters is shown (red and blue, respectively) and their sum, which shows the overlap of the clusters.



Simulation of the histogram of the Greek text image. The histograms (probability density functions) of the individual clusters are shown individually (red and blue) and as the sum (black). Note that the histogram clusters "overlap."

This theoretical histogram is very similar to the histogram of a real image of a manuscript with dark pixels on a lighter background. This is perhaps the most common class of image: a "smallish foreground" object (with a narrowish range of gray values) on top of a "larger background" object with a different range of gray values:





Source: Entire Image

Mean: 140.05 Level:

Std Dev: 25.70 Count:

Median: 146 Percentile:

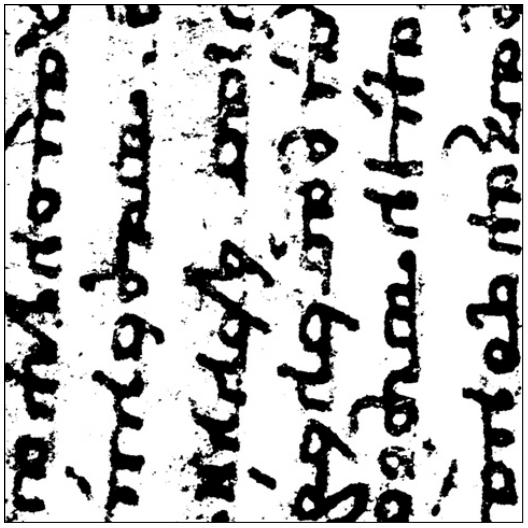
Pixels: 270400 Cache Level: 2

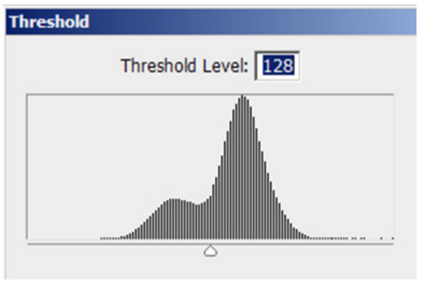
Image of historical manuscript with faded dark text on a lighter background; the histogram shows evidence of two "peaks:" a small darker peak (foreground text) and a large lighter peak (background parchment)

Note that many pixels belonging to the "lightish" background parchment form a "tall" cluster in the histogram whose shape follows the common "bell curve" of probability, which also called a *Gaussian* curve. In the example shown, the darker pixels in the image due to the handwritten characters form a broader cluster (indicating that the cluster has a larger variance) centered about a smaller gray value. The broad cluster has a smaller maximum value since there are fewer dark pixels than light.

Segmentation by Thresholding

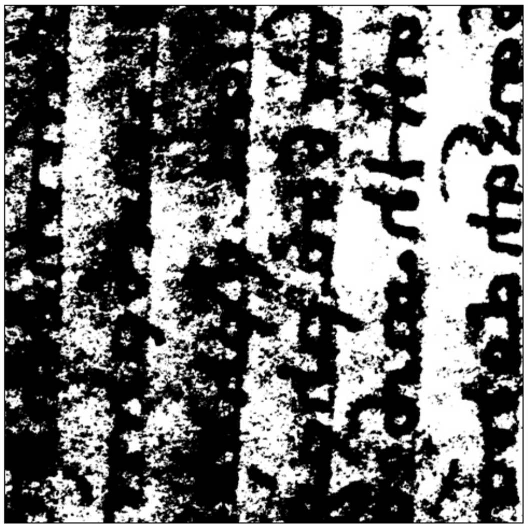
It often is desirable to construct an image where the spread of gray values is eliminated by "segmenting" the pixels belonging to one object from those belonging to the other(s). The most common method for doing this is to "threshold" the image, i.e., set the gray values of pixels below some chosen threshold level to BLACK and those above that level to WHITE. For example, we can threshold the image of text to produce a bitonal output; two examples are shown:

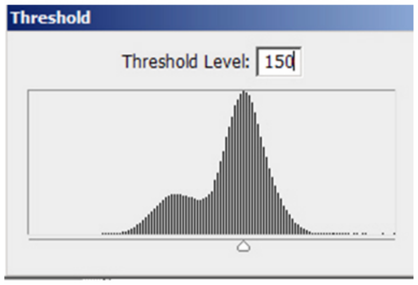




Threshold

Threshold Level: 128





Threshold

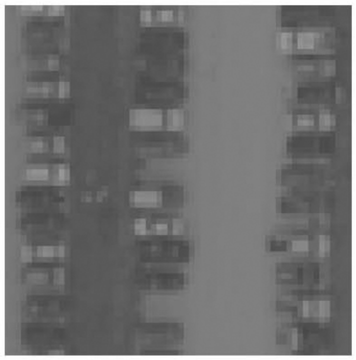
Threshold Level: 150

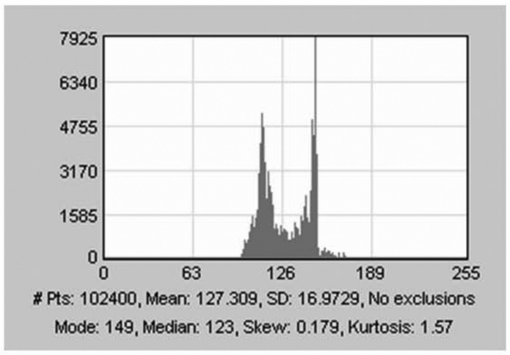
Gray-scale image of old Greek text thresholded at two different gray values; if thresholded at the lower level (top), then most of the background is correctly identified as "white" while some of the text is misidentified as "white"; if thresholded at the higher level, much of the background is misidentified as foreground (black).

Note that the process did not work well in this example; the "spread" in gray value due to the illumination dominated the second thresholded image. These results demonstrate the (obvious) fact that segmentation should work best if the peaks in the histogram are "narrow" and "far apart." The pixels that are correctly identified as belonging to the class of interest (e.g., if a dark text pixel is dark in the thresholded image) are "true positives" and those correctly identified as NOT belonging to that class are "true negatives." Obviously the incorrectly identified pixels are "false positives" (if a parchment pixel is thresholded to "dark" and thus identified as text) and "false negatives."

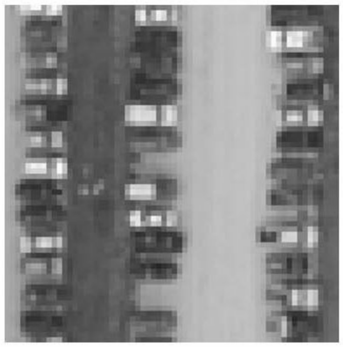
5.1.2 Other Examples of Histograms

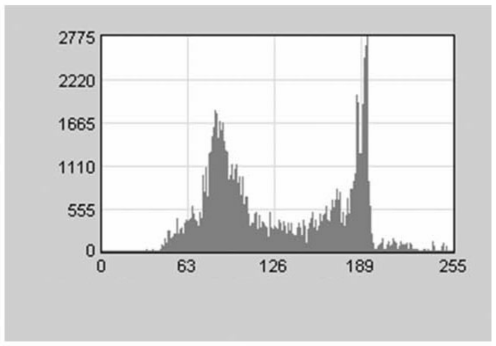
Consider the histogram of a an aerial photogram of a parking lot. The histogram exhibits two obvious clusters, but we can see that these do not correspond to a neat foreground and background.




7925
6340
4755
3170
1585
0
0
63
126
189
255
# Pts: 102400, Mean: 127.309, SD: 16.9729, No exclusions
Mode: 149, Median: 123, Skew: 0.179, Kurtosis: 1.57

Histogram of "low-contrast" image shows that most of the pixels are concentrated in a narrow range of gray value.





Histogram of an image with better contrast better fills the available dynamic range of the image.

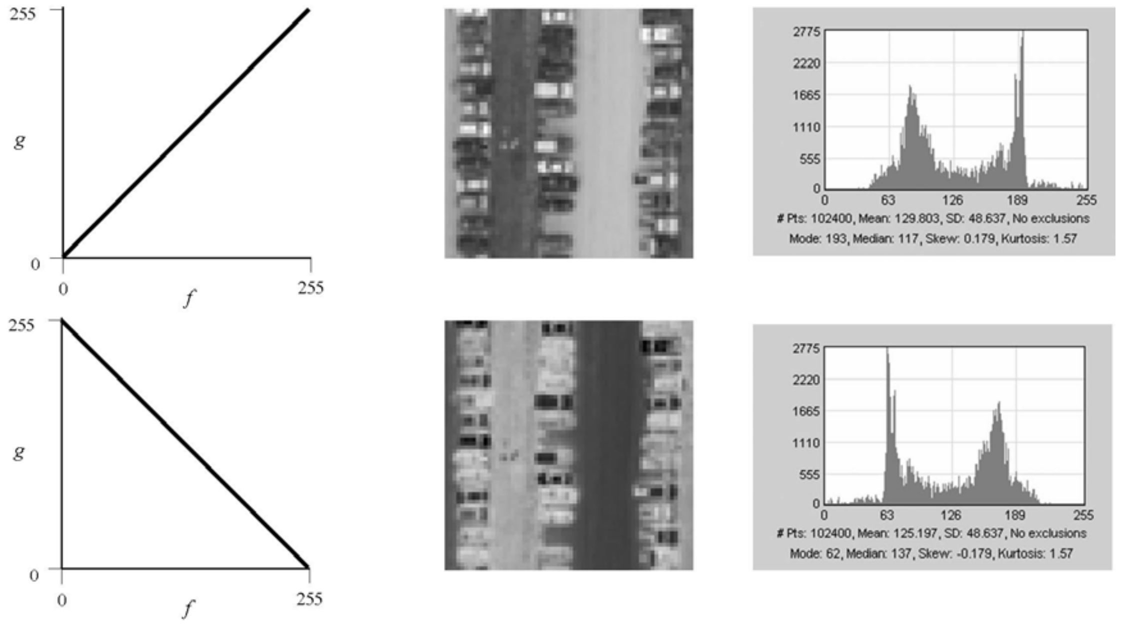
From these results, it is possible to see that more information is conveyed to the viewer if more gray values are occupied. From this observation, it is a "short hop" to see that images with every gray value occupied equally convey the maximum possible amount of "information" about the scene. This is the basis for an important concept in information theory that was pioneered by Claude Shannon in the 1940s. Shannon showed that an appropriate definition of the quantity of information in an image $f[x,y]$ such that the probability of the particular gray value f is represented as $p[f]$:

$$I[f] = -\sum_{f=0}^{f_{\max}} p[f] \cdot \log_2[p[f]] \qquad \left[\frac{\text{bits}}{\text{pixel}}\right]$$

As we shall see shortly, this equation may be analyzed to see that *the maximum information content in an image results when all gray levels are equally populated*; in other words, a flat histogram corresponds to maximum information content.

5.1.3 Histogram Modification for Image Enhancement

In point processing, the only parameter in the pixel transformation is the pixel gray value, which means that all pixels of the same gray level are transformed identically. An example of a such an operation is the "spreading out" of a compact histogram (resulting from a low-contrast image) over the available dynamic range to make the pixel contrast more visible. The mapping from input gray level f to output level g is called a *lookup table*, or *LUT*. Lookup tables may be graphically plotted as transformations $g[f]$ that relate the input gray level (plotted on the x-axis) to the output gray level (on the y-axis). For example, the output resulting from the first mapping below is identical to the input, while the output derived from the second mapping has inverted contrast, i.e., white$\rightarrow$black.



First row: The identity lookup table $g[f] = f$, the resulting image $g[x, y] = f[x, y]$ and its histogram; second row: the "negative" lookup table $g[f] = 255 - f$, the resulting image, and its histogram.

As already mentioned, the image histogram is proportional to the probability distribution of gray levels in the image. The action of any lookup table on an image may be modeled as a transformation of probabilities. Recall that the area under any continuous probability density $p[f]$ or discrete probability distribution p_f is unity:

$$\int_0^{\infty} p[f]\ df = 1 \implies \sum_{n=0}^{\infty} p_n = 1$$

For histograms, the corresponding equations are:

$$\int_0^{f_{\max}} H[f]\ df = \sum_{f=0}^{f_{\max}} H[f] = N \text{ (total number of pixels)},$$

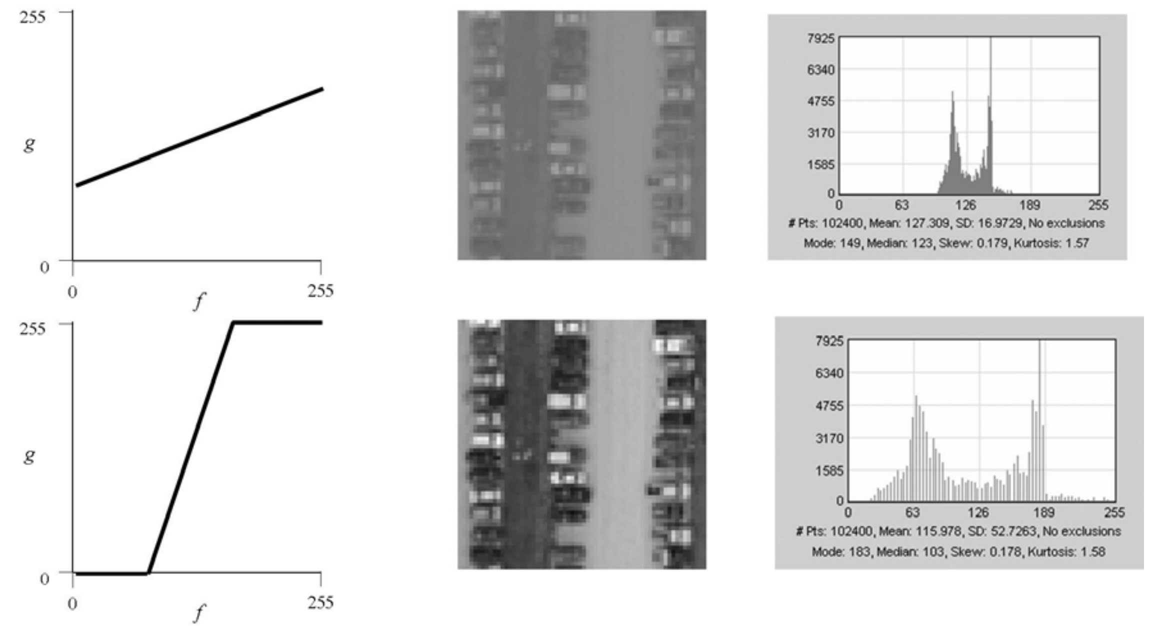
which merely states that every image pixel has some gray level between 0 and $f_{\max}$. Similarly for the output image:

$$\sum_{g=0}^{g_{\max}} H[g] = N.$$

The input and output histograms $H[f]$ and $H[g]$ may be easily related to the lookup table transformation $g[f]$ by using the basic principles of probability theory. Conservation of pixel number requires that incremental areas under the two histograms must match, i.e., if input gray level f_0 becomes output level g_0, then:

$$H[f_0]\ df = H[g_0]\ dg \text{ (continuous gray levels)} \implies H[f_0] = H[g_0] \quad \text{(discrete gray levels)}$$

These equations merely state that all input pixels with level f_0 are mapped to level g_0 in the output.



Pts: 102400, Mean: 127.309, SD: 16.9729, No exclusions
Mode: 149, Median: 123, Skew: 0.179, Kurtosis: 1.57

Pts: 102400, Mean: 115.978, SD: 52.7263, No exclusions
Mode: 183, Median: 103, Skew: 0.178, Kurtosis: 1.58

A lookup table that decreases the contrast, the resulting image, and the concentrated histogram; the linear contrast enhancement lookup table, its result when applied to the low-contrast image, and the "spread-out" histogram.

5.1.4 Jones Plots

It may be useful to plot the histogram of the input image, the lookup table ("gray-level transformation"), and the histogram of the output image as shown below. The input histogram (upside down) is at the lower-right; the output histogram (rotated 90° counterclockwise) is at the upper left; the lookup table is at the upper right. The new gray level g_0 is determined by mapping the value f_0 through the curve $g[f]$ onto the vertical axis. In this case, the gray levels of the original low-contrast image are spread out to create a higher-contrast image.

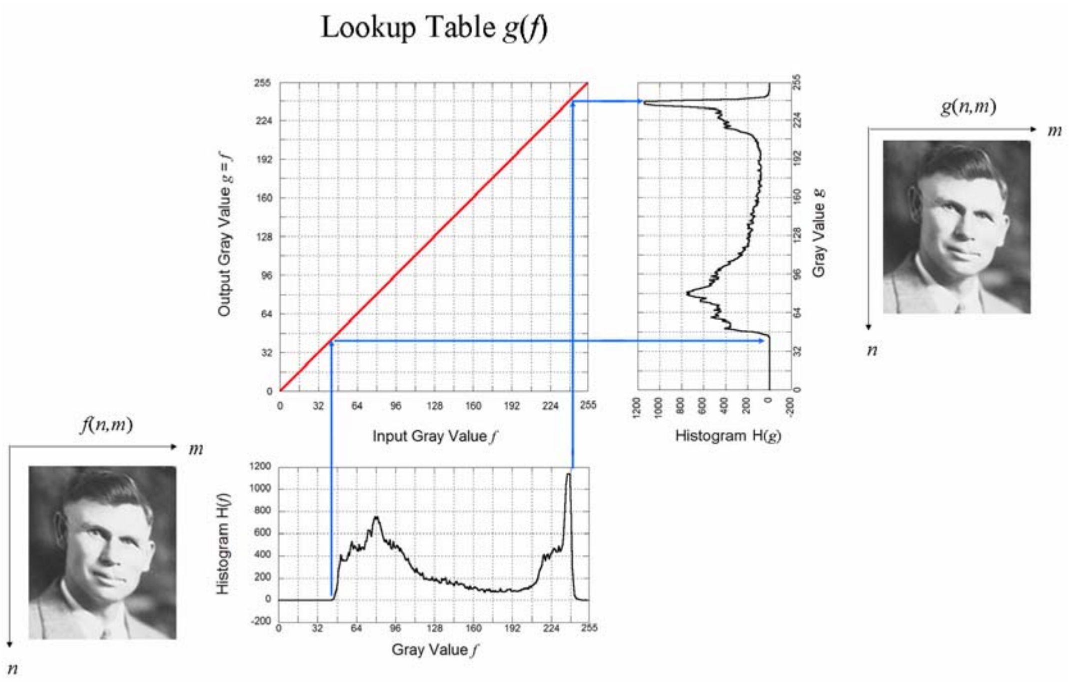


Figure 5.1: *Identity lookup table: the input image $f[x,y]$ and its histogram $H[f]$ are shown along the bottom. The lookup table $g[f] = f$ is shown as the red line at the rising angle of $45°$. The mappings of two gray values are shown in blue; the line from the input histogram is "reflected" through the lookup table to create the value in the output histogram $H[g]$ (on the right). The output image is $g[x,y] = f[x,y]$.*

Identity Lookup Table

The first example of a point operation is the (trivial) identity operator, where the gray value of the "output" pixel is identical to that of the input pixel, i.e., level "0" maps to "0," "1" to "1," etc. The lookup table is the line $g = f$, which rises at an angle of 45°.

"Inversion" or "Negative" Lookup Table

The second example computes the "inverse" or "negative" of the digital image. Actually, both of these names are somewhat misleading, since the output value remains a positive integer within the available dynamic range. The lookup table evaluates the new gray value

$$g[f] = f_{\max} - f$$

where $f_{\max} = 255$ in these 8-bit images. This indicates this lookup table is more appropriately called the "complement" instead of the "negative" or "inverse."

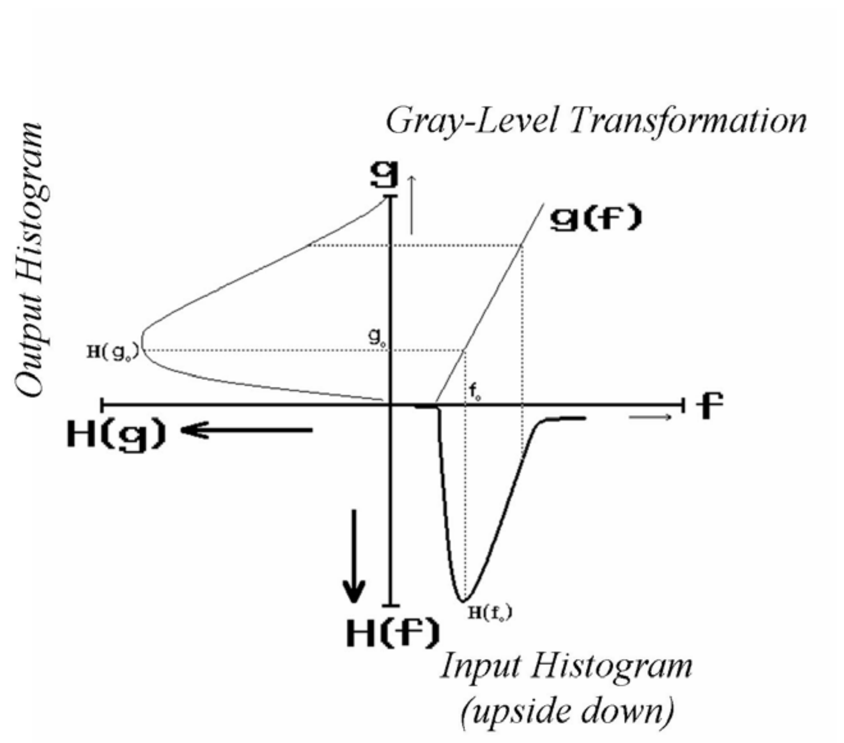


Figure 5.2: *Jones plot for contrast enhancement for a continuous histogram.*

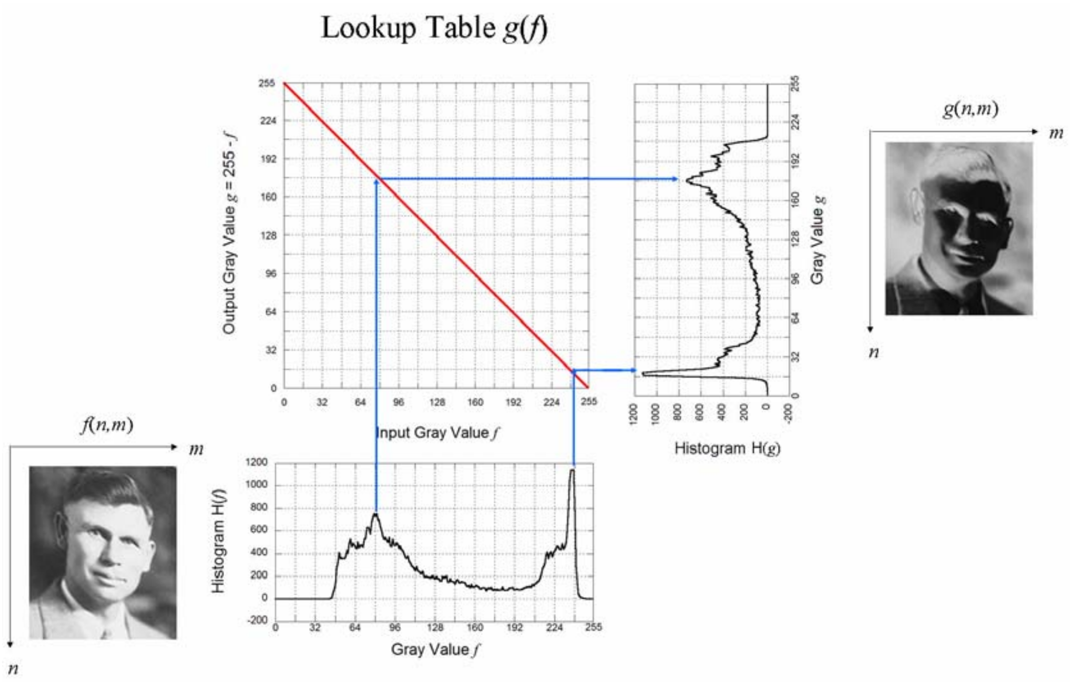


Figure 5.3: *Complement lookup table: The lookup table* $g[f] = f_{\max} - f$ *is shown as the red line at the falling angle of* $45°$*. "Dark" input pixels (at the left side of the histogram) are mapped to "bright" values. The output histogram* $H[g]$ *is "reversed" (flipped end for end).*

As already mentioned, the image histogram is proportional to the probability distribution of gray levels in the image. The action of any lookup table on an image may be modeled as a transformation of probabilities. The probability of a pixel having one of the available gray values must be unity:

$$\sum_{f=0}^{f_{\max}} p_f = 1$$

The corresponding equation for the histogram is:

$$\sum_{f=0}^{f_{\max}} H(f) = N \text{ (total number of pixels)},$$

which merely states that every image pixel has some gray level between 0 and $f_{\max}$. Similarly for the output image:

$$\sum_{g=0}^{g_{\max}} H(g) = N.$$

The input and output histograms $H(f)$ and $H(g)$ are related by the lookup table transformation $g(f)$ via the basic principles of probability theory. The fact that the number of pixels must be conserved requires that the values of the two histograms at corresponding gray levels must match, so that if input gray level f_0 becomes output level g_0, then:

$$H[f_0] = H[g_0] \quad \text{(discrete gray levels)}$$

These equations merely state that all input pixels with level f_0 are mapped to level g_0 in the output.

Contrast Enhancement

Probably the most common use of digital image processing is to enhance the contrast of images. In other words, the input image $f[x,y]$ is transformed to the output image $g[x,y]$ via a mapping $g[f]$. The histograms of the input and output images, $H(f)$ and $H(g)$, are related by the lookup table transformation $g(f)$ via the basic principles of probability theory. The fact that the number of pixels must be conserved requires that the values of the two histograms at corresponding gray levels must match, so that if input gray level f_0 becomes output level g_0, then:

$$H[f_0] = H[g_0] \quad \text{(discrete gray levels)}$$

These equations merely state that all input pixels with level f_0 are mapped to level g_0 in the output.The schematic of the process is shown:

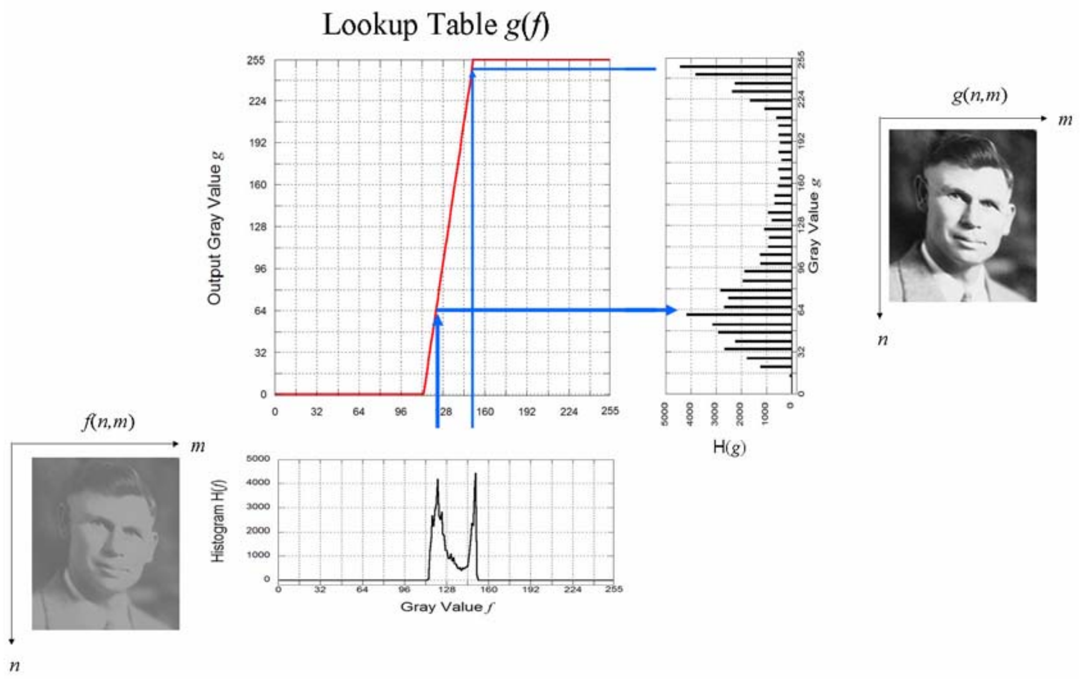


Figure 5.4: *Demonstration of contrast enhancement: the input image $f(x,y)$ and its histogram $H(f)$ are shown along the bottom. The lookup table $g(f)$ exhibits a steep linear slope over the range of occupied gray values f. Adjacent gray values are "spread apart" by the action of the lookup table (blue lines show the mapping for two levels).*

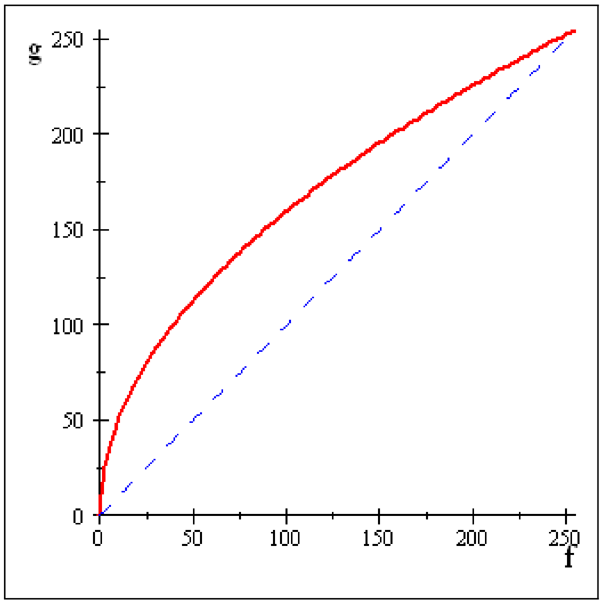
Ad Hoc Pixel Operations for Image Enhancement

Many images have low contrast at small gray values (in the "shadows") and fewer pixels with large gray values ("highlights"). These images should have their small gray values "spread out" and the large gray values "pushed together." An appropriate *ad hoc* lookup table is the appropriately scaled square root function:

$$g[f] = 255 \cdot \sqrt{\frac{f}{255}} = \sqrt{255 \cdot f}$$

$$\sqrt{255 \cdot 200}$$

$: 225.83 : 195.58 : 5\sqrt{510} = 112.92$



Possible lookup table for "ad hoc" enhancement of common 8-bit images. The curve $g[f] = \sqrt{255 \cdot f}$ *(red) is compared to the identity lookup table* $g[f] = f$ *(black). The square-root LUT enhances contrast in shadow regions (dark values, where* $\frac{dg}{df} > 1$*) while compressing gray values in highlights (light values where* $\frac{dg}{df} < 1$*).*

5.2 Cumulative Histogram

Given an N-pixel image $f[x,y]$ with gray values in the range $0 \leq f \leq f_{\max}$ and having histogram $H[f]$, then the *Cumulative Histogram* evaluated at specific gray value f_0 is the number of pixels with gray value $\leq f_0$:

$$C[f_0] = \sum_{f=0}^{f_0} H[f] = \frac{1}{N} \sum_{f=0}^{f_0} p[f]$$

Clearly, the cumulative histogram evaluated at the maximum gray value must yield the number of pixels in the image:

$$C[M] = \sum_{f=0}^{f_{\max}} H[f] = N$$

In the case of a continuous probability distribution, the cumulative histogram is an integral over gray level:

$$C[f_0] = \int_{f=0}^{f_0} H[f]\ df$$

If the image f has the "flat" histogram just described, so that $H[f]$ is constant for all values of f, then the value in each bin must be the ratio of the number of pixels and the number of gray values:

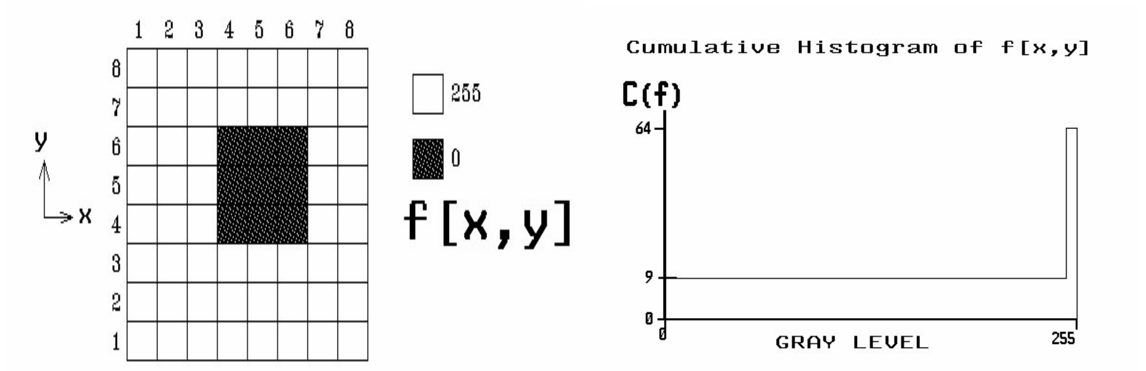
$$\textit{flat histogram} \implies H[f] = \frac{N}{f_{\max}+1}$$

The corresponding cumulative histogram is:

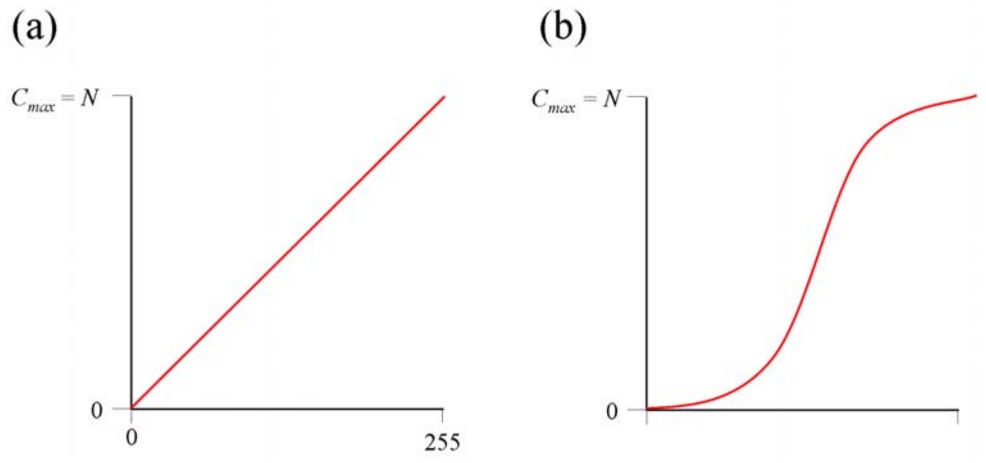
$$C[f_0] = \sum_{f=0}^{f_0} H[f] = \left(\frac{N}{f_{\max}+1}\right) \cdot \sum_{f=0}^{f_0} 1 = \frac{N}{f_{\max}+1} \cdot f_0$$

which increases by equal numbers of pixels as f_0 is incremented. On the appropriately scaled histogram graph, the cumulative histogram resulting from a flat histogram (maximum information content) is a straight line at 45°.

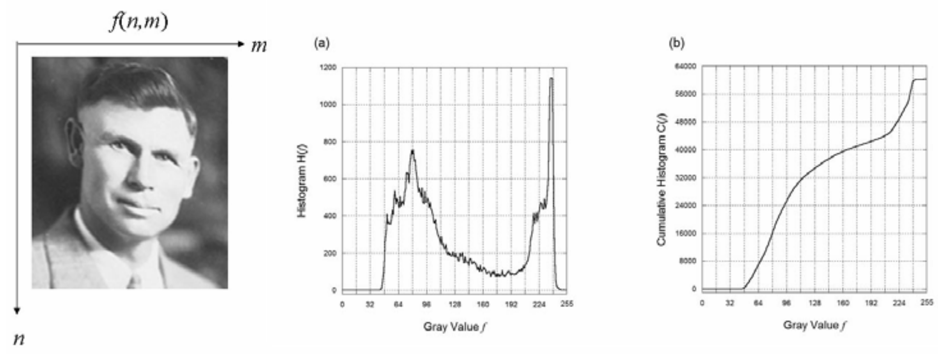
The cumulative histogram is used to derive the mapping that maximizes the global visibility of changes in image gray value (histogram *equalization*) and for deriving an output image with a specific histogram (histogram *specification*).

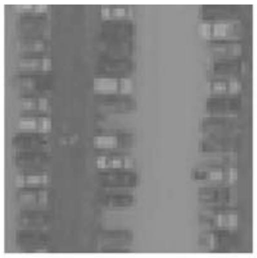


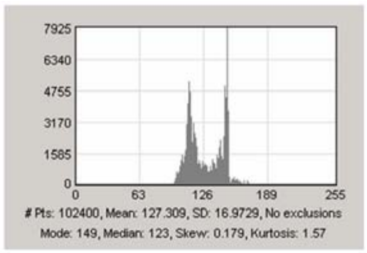
If $H[f]$ is "flat" (meaning that every gray level "has" the same number of pixels), then the associated cumulative histogram $C[f]$ increases by the same number of pixels for each gray value, and thus forms a linear "ramp" function. Such an image has maximum "contrast" of pixels, meaning that such an image exhibits the most subtle changes over the entire range of gray value; the *information content* of the image is maximized. The cumulative histogram of an image with less information will rises more rapidly at those gray levels where most pixels lie (often about the "mid-gray" value) and more slowly over the other levels. Examples are shown in the figure.



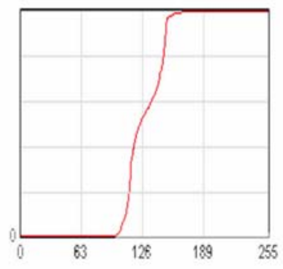
Cumulative histograms: (a) of an image with the same number of pixels in each gray value (thus maximizing the ability to distinguish small changes in gray value); (b) of an image with the pixels concentrated at a mid-gray value.



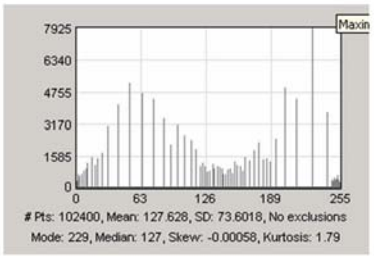

Low-Contrast Image



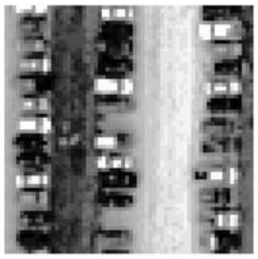
Low-Contrast Histogram

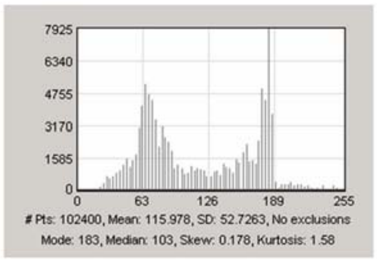


Cumulative Histogram
$\propto$ Lookup Table



Equalized Histogram


Contrast-Enhanced Image



Histogram After
Linear Enhancement

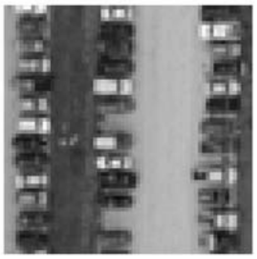


Image After
Linear Enhancement

Histogram Equalization ("Flattening")

We just mentioned that the maximum "quantity" of information is determined by the probability distribution of gray values, and thus by the histogram. The quantity of information I in an image $f[x,y]$ was specified by Claude Shannon in the late 1940s:

$$I[f] = -\sum_{f=0}^{f_{\max}} p[f] \cdot \log_2(p[f]) \qquad \left[\frac{\text{bits}}{\text{pixel}}\right]$$

where $p[f]$ is the probability of occurence of gray value f. The meaning of information content is very important in image *compression*, which is the task of discarding "unnecessary" (or "redundant") image data to reduce the required storage space or transmission time. As it happens, the information content in an image is maximized if all gray levels are equally populated, so that $H(f_0) = H(f_1) = \cdots = H(f_{\max})$. This ensures that the differences in gray level within the image are spread out over the widest possible range and thus maximizes the ability to distinguish differences in gray values.

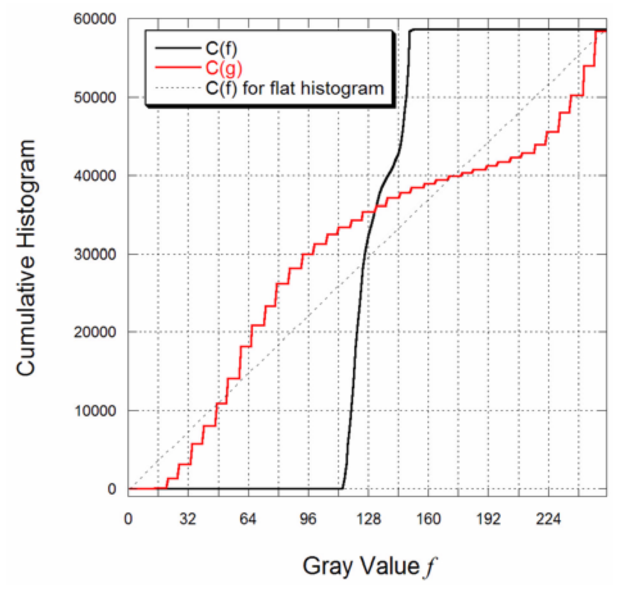


Figure 5.5: *Cumulative histogram of input function $f[x,y]$, which is scaled to produce the (nonlinear) lookup table for histogram equalization.*

The process of maximizing the visibility of image information is called histogram *equalization* or *flattening*, and is an operation available in many computer image processing programs. It is easy to describe and implment, BUT it is not appropriate for all applications and it is important to understand the reason why.

The lookup table $g(f)$ for histogram flattening is proportional to the cumulative histogram of the input image $C[f]$. The mathematical derivation of the appropriate $g[f]$ is straightforward. Assume the point operation (lookup table) $\mathcal{O}\{f[x,y]\} = g[x,y]$ equalizes the output histogram, i.e., the output histogram is "flat," so that $H(g)$ is some constant. For simplicity, assume that gray levels are continuous and that the lookup transformation $g[f]$ is monotonically increasing:

$$g[f_0] = g_0$$
$$g[f_0 + \Delta f] = g_0 + \Delta g$$

Since the lookup table $g[f]$ must be a monotonically increasing function, then the corresponding inverse operation must exist (call it g^{-1}), so that $g^{-1}[g_0] = f_0$. Because the number of pixels must be conserved (each pixel in $f[x,y]$ is also in $g[x,y]$), then the probabilities must satisfy the relation:

$$\begin{aligned} p[f]\ \Delta f &= p[g]\ \Delta g \\ &\Longrightarrow \frac{H[f]}{N} \cdot \Delta f = \frac{H[g]}{N} \cdot \Delta g \\ &\Longrightarrow H[f] \cdot \Delta f = H[g] \cdot \Delta g \end{aligned}$$

but $H[g]$ is constant by assumption (flat histogram), so substitute $H[g] = k$, a constant:

$$H \cdot \Delta f = k \cdot \Delta g$$

Sum (or integrate in the continuous case) both sides over the range of allowed levels from $f = 0$ to $f = f_0$. The summation evaluates the output gray level g_0 for input level f_0:

$$\sum_0^{f_0} H[f] = k \cdot \sum_0^{f_0} \Delta g = k \cdot g[f_0]$$

$$\textit{but} \ \sum_0^{f_0} H[f] = C[f_0] \ \textit{by definition of cumulative histogram}$$

$$\Longrightarrow k \cdot g[f_0] = C[f_0]$$

$$\Longrightarrow g[f_0] = \frac{1}{k} \cdot C[f_0]$$

where the value The proportionality constant k may be evaluated for the number R of available gray levels (dynamic range) and the number of pixels N

$$k = \frac{N}{R}$$

The lookup table that equalizes the image histogram is:

$$g_{flat}[f_0] = \frac{R}{N} \cdot C[f_0]$$

Note that gray value "0" of the equalized image takes on the scaled value of the cumulative histogram:

$$g_{flat}[f = 0] = \frac{R}{N} \cdot C[f = 0] = \frac{R}{N} \cdot H[0]$$

because the first element of the cumulative histogram is identical to the first element of the histogram: $C[0] = H[0]$. If lots of pixels in the original image have gray value "0", then $H[0]$ would be large, and perhaps so large that the first occupied gray value of the equalized image would not be zero. For example, if half of the pixels of $f[x,y]$ are black with value "0", then the first occupied level of $g[x,y]$ would be the midgray value. This clearly is not an efficient use of gray values, so the first occupied gray value of $g[x,y]$ often is set to zero by subtracting $g[0]$ from the computed gray values:

$$\begin{aligned} g_{flat}[f = f_0] &= \frac{R}{N} \cdot (C[f_0] - C[0]) \\ &= \frac{R}{N} \cdot C[f_0] - \frac{R}{N} \cdot H[0] \end{aligned}$$

which ensures that $g_{flat}[f = 0] = 0$.

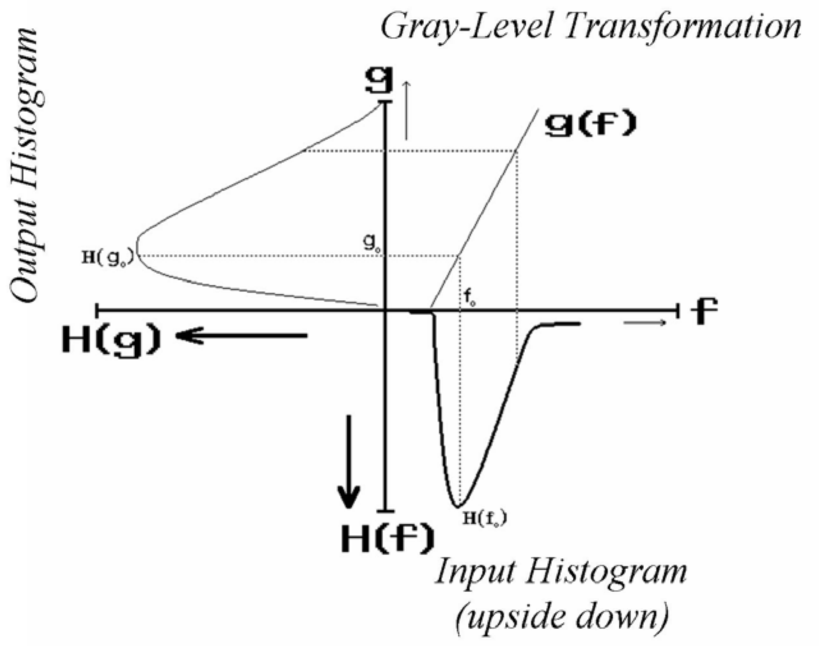
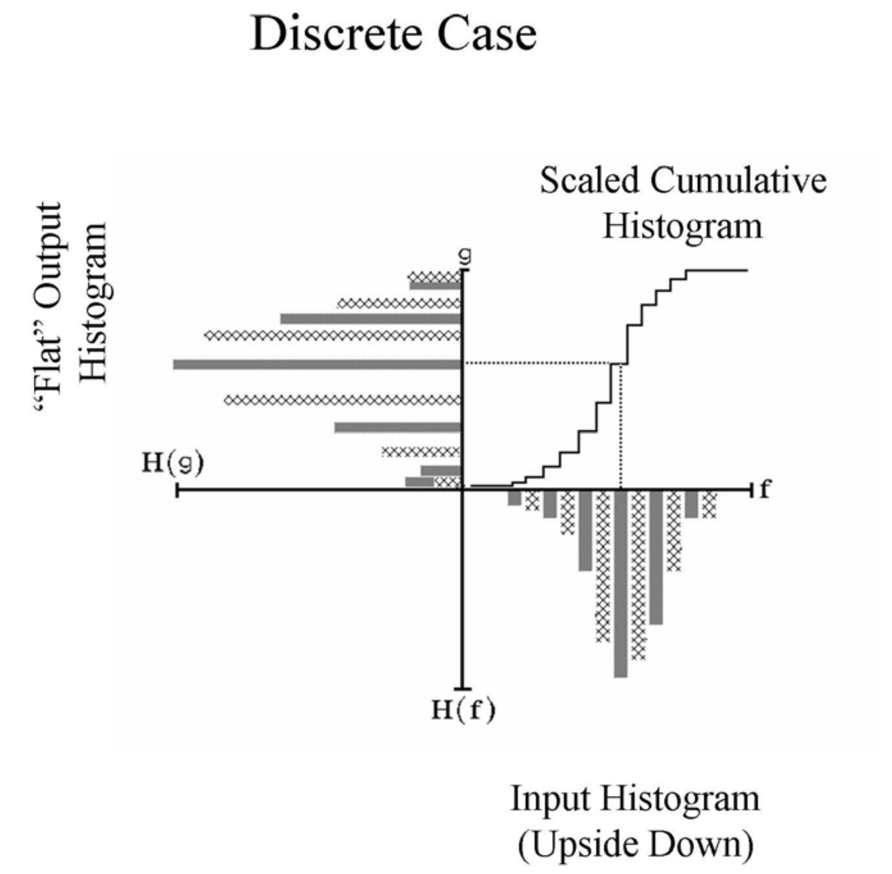


Figure 5.6: *Jones plot for contrast enhancement for a continuous histogram.*

Since all pixels with the same discrete gray level f_0 are treated identically by the transformation, the histogram is "flattened" by spreading densely occupied gray values into "neighboring," yet sparsely occupied, gray levels. The resulting histogram is as "flat" as can be obtained without basing the mapping on features other than gray level.

The local areas under the input and flattened histograms must match; where $H[f]$ is large, the interval Δf is spread out to a larger Δg, thus enhancing contrast. Where $H[f]$ is small, the interval Δf maps to a smaller Δg, thus reducing the differences in gray value.

Adjacent well-populated gray levels are spread out, thus leaving gaps (i.e. unpopulated levels) in the output histogram. Pixels in adjacent sparsely populated gray levels of $f[x,y]$often are merged into a single level in $g[x,y]$. In practice, information about gray-level differences at gray values with few pixels may be lost in the output image, due to combining those pixels into single levels.



Jones plot for contrast enhancement in the discrete case.

Example of Histogram Equalization – 1-D "Image" Because the equalization process acts on the histograms (and thus only indirectly on the image), the mathematical operation does not depend on the number of spatial dimensions in the input image; the process works as well for 1-D as 2-D images (or 3-D or 4-D or ...). For simplicity of presentation, consider equalization of the histogram of a 1-D function. This case considers an "image" in the form of a decaying exponential with 256 pixels quantized to 6 bits (so that $0 \leq f \leq 63$). The object is shown in (a) its histogram in (b), and its cumulative histogram in (c). Note that the histogram exhibits significant clustering; there are more "dark" than "light" pixels. The lookup table for histogram equalization is a scaled replica of the cumulative histogram and is shown in (d). The cumulative histogram of the equalized output image is $C[g]$ in (e), and the output histogram $H[g]$ in (f). Note that the form of the output image in (g) is approximately linear, significantly different from the decaying exponential object in (a). In other words, the operation of histogram equalization changed BOTH the spatial character as well as the quantization. The gray levels with large populations (dark pixels) pixels have been spread apart in the equalized image, while levels with few pixels have been compressed together.

Example of Histogram Equalization – 2-D "Image" An example of equalizing of the histogram of a 2-D image is shown, where again the cumulative histogram is flattening of 1-D function.jpg

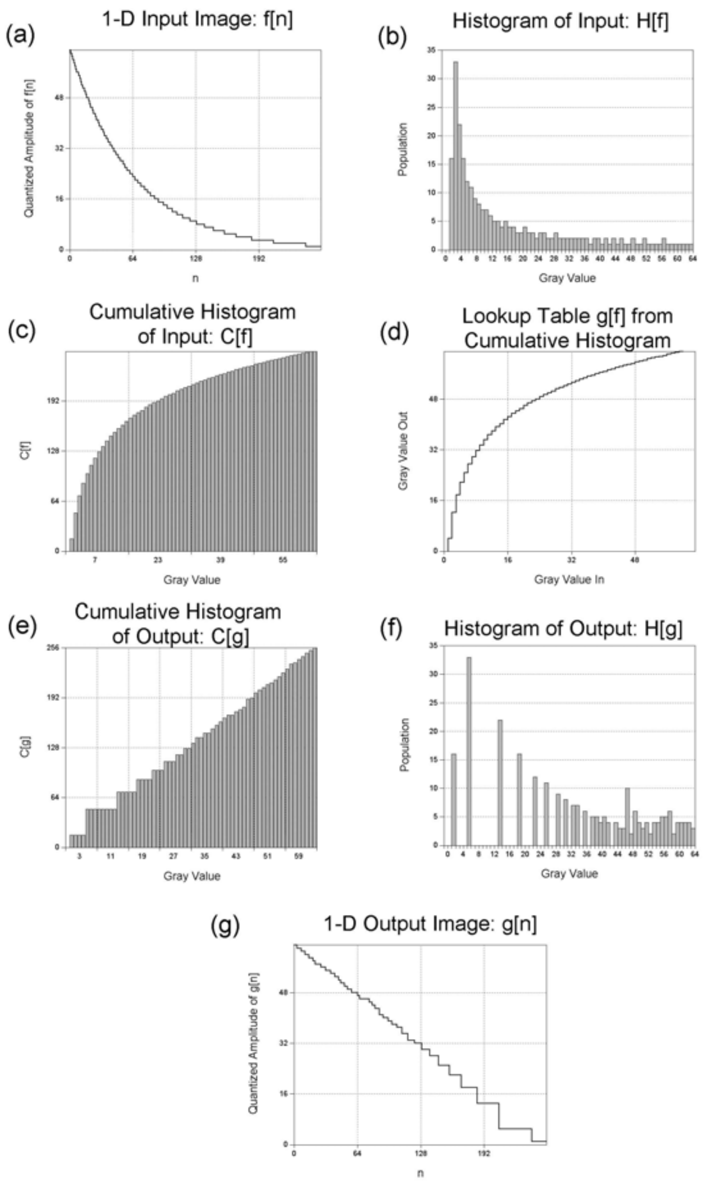


Figure 5.7: *Illustration of histogram flattening of a 1-D function: (a) 256 samples of $f[n]$, which is a decaying exponential quantized to 64 levels; (b) its histogram $H[f]$, showing that the smaller population of larger gray values; (c) cumulative histogram $C[f]$; (d) Lookup table, which is scaled replica of $C[f]$ (note that it is nonlinear); (e) Cumulative histogram of output $C[g]$, which more closely resembles a linear ramp; (f) histogram $H[g]$, which shows the wider "spacing" between levels with large populations; (g) .Output image $g[n]$ after quantization.*

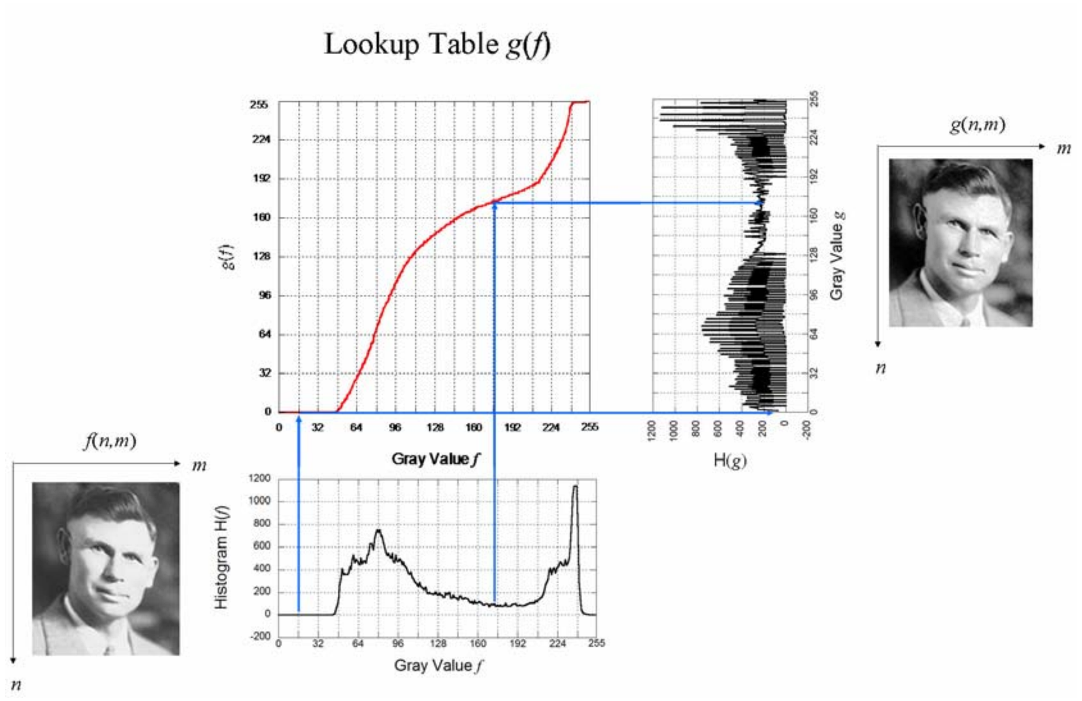


Figure 5.8: *Schematic of histogram equalization: the input image $f[x,y]$ and its histogram $H[f]$ are shown along the bottom. The lookup table $g[f]$ is a scaled replica of the cumulative histogram $C[f]$. The output image $g[f]$ and histogram $H[g]$ are shown on the right. Adjacent gray values with many pixels are spread apart due to the steepness of the cumulative histogram.*

not a linear function.

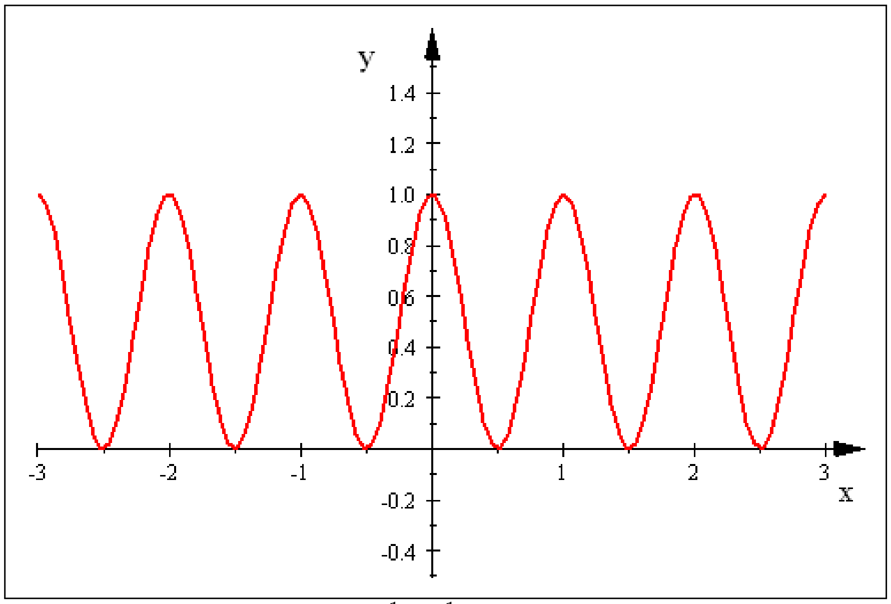
5.2.1 Nonlinear Nature of Histogram Equalization

Clearly the equalization lookup tables in the 1-D and 2-D examples just considered are not straight lines, which means that the gray value g of the "output" pixel is NOT proportional to f; this ensures that the gray-scale mapping of histogram equalization is (usually) NOT linear. Also, histogram equalization is based on the histogram of the image, it is a "custom" process for each image and the lookup table is almost never a straight-line graph. In other words, histogram equalization is inherently "nonlinear," which has significant impact on the "character" of the processed image.

Though the details of the effect of a nonlinear operation on an image are beyond the scope of this class, it is easy to see one significant impact of a nonlinear operation.

Consider a 1-D input that is a biased cosine function:

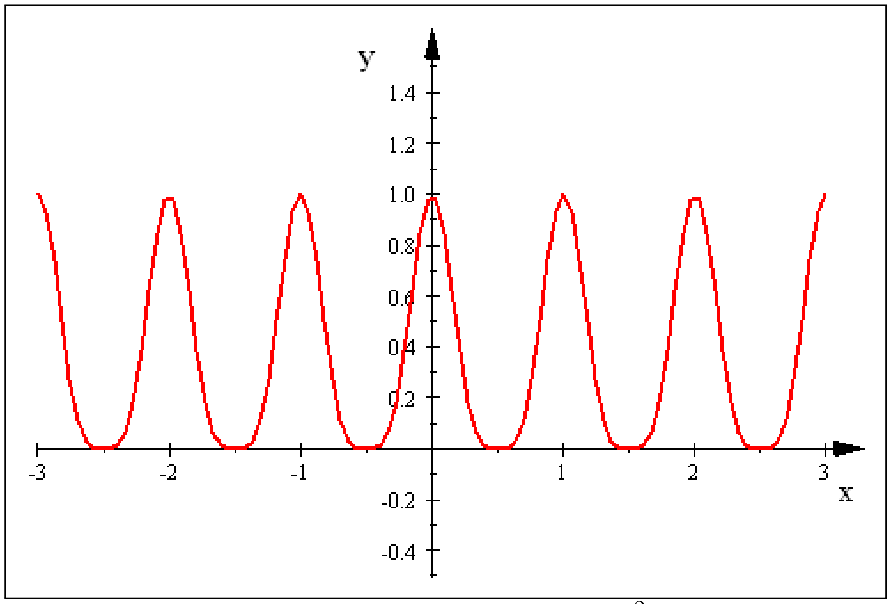
$$\begin{aligned}
f[x] &= \frac{1}{2}+\frac{1}{2}\cos[2\pi\xi_0 x] \\
&= \frac{1}{2}\cdot 1[x]+\frac{1}{2}\left(\frac{\exp[+2\pi i\xi_0 x]+\exp[-2\pi i\xi_0 x]}{2}\right) \\
&= \frac{1}{2}\cdot\exp[+2\pi i\cdot 0\cdot x]+\frac{1}{2}\left(\frac{\exp[+2\pi i\xi_0 x]+\exp[-2\pi i\xi_0 x]}{2}\right)
\end{aligned}$$



$f[x]=\frac{1}{2}+\frac{1}{2}\cos[2\pi\xi_0 x]$

which is composed of complex exponential functions with spatial frequencies $\xi = 0$ and $\xi = \pm\xi_0$. Now perform a simple nonlinear operation by evaluating the square of $f[x]$:

$$(f[x])^2 = \left(\frac{1}{2}+\frac{1}{2}\cos[2\pi\xi_0 x]\right)^2$$



$(f[x])^2 = \left(\frac{1}{2} + \frac{1}{2}\cos[2\pi\xi_0 x]\right)^2$

The square has the same limits $0 \leq (f[x])^2 \leq 1$, but the function looks "different." We can re-express it using the same method for $f[x]$:

$$\begin{aligned}
(f[x])^2 &= \left(\frac{1}{2} + \frac{1}{2}\cos[2\pi\xi_0 x]\right)^2 \\
&= \frac{3}{8} + \frac{1}{2}\cos[2\pi\xi_0 x] + \frac{1}{8}\cos[4\pi\xi_0 x] \\
&= \frac{3}{8} + \frac{1}{2}\cos[2\pi\xi_0 x] + \frac{1}{8}\cos[2\pi\cdot(2\xi_0)\cdot x] \\
&= \frac{3}{8}\cdot 1[x] + \frac{1}{2}\left(\frac{\exp[+2\pi i\xi_0 x] + \exp[-2\pi i\xi_0 x]}{2}\right) \\
&\quad + \frac{1}{8}\cdot\left(\frac{\exp[+2\pi i\cdot(2\xi_0)\cdot x] + \exp[-2\pi i\cdot(2\xi_0)\cdot x]}{2}\right)
\end{aligned}$$

$$\begin{aligned}
(f[x])^2 = \frac{3}{8}\cdot 1[x] + \frac{1}{2}\left(\frac{\exp[+2\pi i\xi_0 x] + \exp[-2\pi i\xi_0 x]}{2}\right) \\
+ \frac{1}{8}\cdot\left(\frac{\exp[+2\pi i\cdot(2\xi_0)\cdot x] + \exp[-2\pi i\cdot(2\xi_0)\cdot x]}{2}\right)
\end{aligned}$$

which shows it to be composed of complex exponential functions with spatial frequencies $\xi = 0$, $\xi = \pm\xi_0$, AND $\xi = \pm 2\xi_0$. In other words, the squaring operation "added" two additional spatial frequencies (that is, an additional real-valued sinusoidal function) "out of thin air." This is a fundamental effect on *any* nonlinear operation (not just of evaluating the square). The addition of other sinusoidal components complicates the process of comparing two images before and after a nonlinear operation.

For subjective applications, where the visual "appearance" of the output image is the only concern, the nonlinearity typically poses no problem. However, if two images with different histograms are to be compared in a quantitatively meaningful way (e.g., to detect seasonal changes from images taken from an airborne platform), then independent histogram equalization of the two images before comparison is not appropriate because the two operations are (generally) different. Nonlinear operations produce unpredictable effects on the spatial frequency content of the scene, as you will see in the class in linear mathematics. Images should either be compared after applying linear mappings are based on pixels of known absolute "brightness", or after the histogram specification process discussed next.

Nonlinear mappings are used deliberately in many applications – a complete nonlinear system is a "compandor", a composite word including "compressor" and "expandor". The process of companding is used to maintain the signal dynamic range and thus improve the "signal-to-noise" ratio in a noise reduction system. A common companding system used in audio systems is the well-known *Dolby noise reduction* system which is still used for recording analog audio signals on magnetic tape, which generates an audible noise signal (called tape "hiss") even if no signal is recorded. The Dolby system works by boosting the amplitude of low-level high-frequency input signals before recording; this is called "pre-emphasis." The unavoidable tape hiss is recorded along with the boosted signal. The boost decreases with increasing level of the input signal. The inverse process of de-emphasis (attenuation) is performed on playback, so that the recorded signal is faithfully reproduced but the recorded tape hiss is attenuated by the de-emphasis. Compandors are also used in digital imaging systems to preserve highlights and shadow detail in digital imaging systems.

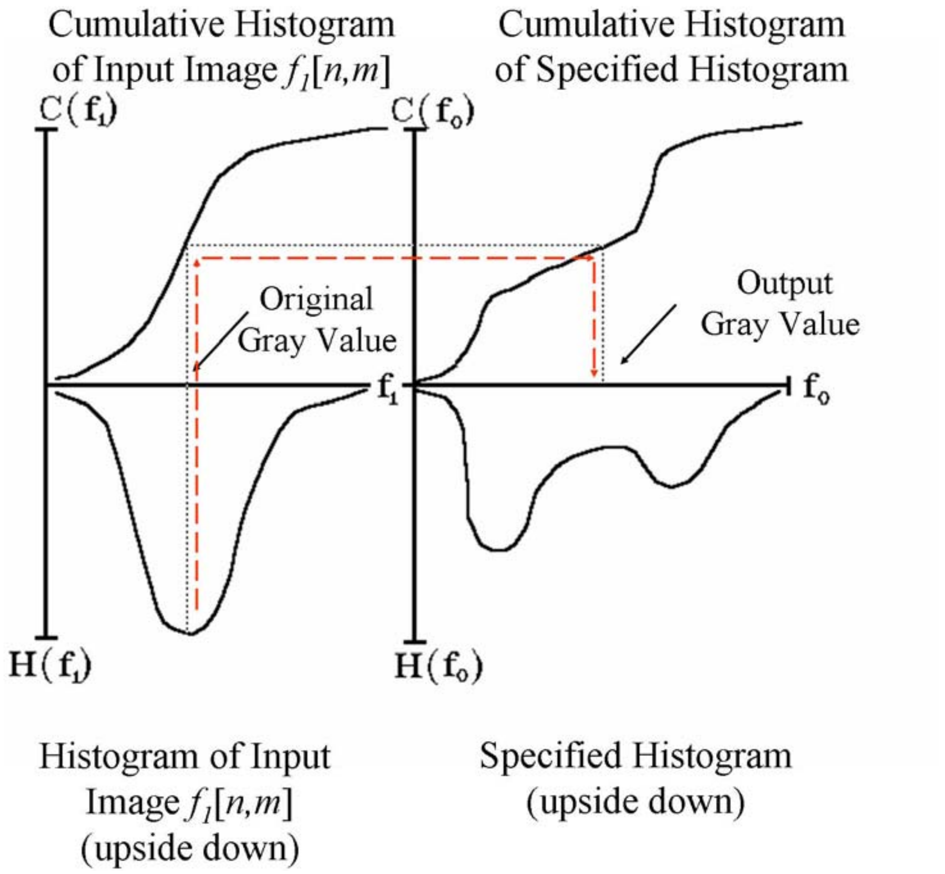
5.2.2 Histogram Specification or "Matching"

It is often useful to transform the histogram of an image to create a new image whose histogram "matches" match that of some reference image $f_{ref}[x,y]$. This process, histogram specification, is a generalization of histogram equalization and allows direct comparison of images perhaps taken under different conditions, e.g., LANDSAT images taken through different illuminations or atmospheric conditions. The required transformation of the histogram of f_1 to $H[f_{Ref}]$ may be derived by first equalizing the histograms of both images:

$$\mathcal{O}_{REF}\{f_{REF}[x,y]\} = e_{REF}[x,y]$$
$$\mathcal{O}_1\{f_1[x,y]\} = e_1[x,y]$$

where $e_n[x,y]$ is the image of $f_n[x,y]$ with a flat histogram obtained from the operator $\mathcal{O}\{\ \}$; the histograms of e_{REF} and e_1 are "identical" (both are flat). The inverse of the lookup table tranformation for the reference image is $\mathcal{O}^{-1}\{g_{REF}\} = f_{REF}$. The lookup table for histogram specification of the input image is obtained by first deriving the lookup tables that would flatten the histograms of the input and reference image. It should be noted that some gray levels will not be specified by this transformation and so must be interpolated. The functional form of the operation is:

$$g_1[x,y]\ \textit{(with specified histogram)} = \mathcal{O}_{REF}^{-1}\{\mathcal{O}_1\{f_1\}\} = \left[\mathcal{O}_{REF}^{-1}\cdot\mathcal{O}_1\right]\{f_1\} \propto \mathcal{C}_{REF}^{-1}\{\mathcal{C}_1\{f_1\}\}$$


Cumulative Histogram of Input Image $f_1[n,m]$
Cumulative Histogram of Specified Histogram
$C(f_1)$
$C(f_0)$
Original Gray Value
Output Gray Value
f_1
f_0
$H(f_1)$
$H(f_0)$
Histogram of Input Image $f_1[n,m]$ (upside down)
Specified Histogram (upside down)

Schematic of Histogram Specification: given input image $f_1[x,y]$ and desired "reference" histogram $H[f_0]$, the input gray value is mapped through its cumulative histogram $C[f_1]$ and the "inverse" of the reference cumulative histogram $C[f_0]$ to find the "output" gray value f_0.

5.3 Examples of Point Operations

In point processing, the only parameter available in the pixel transformation is the gray value of that pixel; all pixels of the same gray level must be transformed identically by a point process. The mapping from input gray level f to output level g is called a *lookup table* or *LUT*. Lookup tables often are graphed as functions $g(f)$ that relate the input gray level f (plotted on the x-axis) to the output gray level g (on the y-axis). One such lookup-table operation is the "spreading out" of the compact histogram of a low-contrast image over the full available dynamic range to make the image information more visible; as shown in a subsequent example.

5.4 Application of Histograms to Tone-Transfer Correction

Histogram specification may be used to correct a nonlinear tone-transfer curve of the digitizer to ensure that the overall tone transfer is linear. The recorded image $g_1[n \cdot \Delta x, m \cdot \Delta y]$ is obtained from the sampled input image $f[n \cdot \Delta x, m \cdot \Delta y]$ through the transfer curve (lookup table) $g_1[f]$, which may be measured by digitizing a linear step wedge. The inverse of the transfer curve may be calculated and cascaded as a second lookup table g_2 to linearize the total transfer curve:

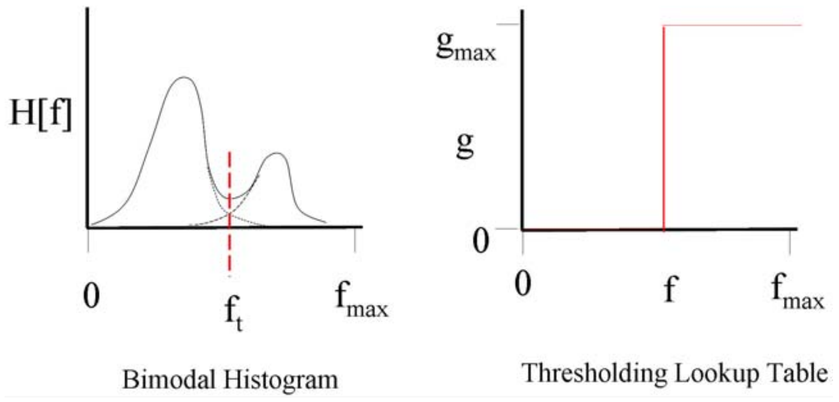
$$\begin{aligned} g_2[g_1(f[n \cdot \Delta x, m \cdot \Delta y])] &= f[n \cdot \Delta x, m \cdot \Delta y] \\ &\Longrightarrow g_2[g_1] = 1 \\ &\Longrightarrow g_2[x] = g_1^{-1}[x] \end{aligned}$$

Note that the display may be linearized in similar fashion. Consider a nonlinear digitizer transfer curve of the form $g_1[f] = \sqrt{f}$. The correction curve necessary to linearize the system is:

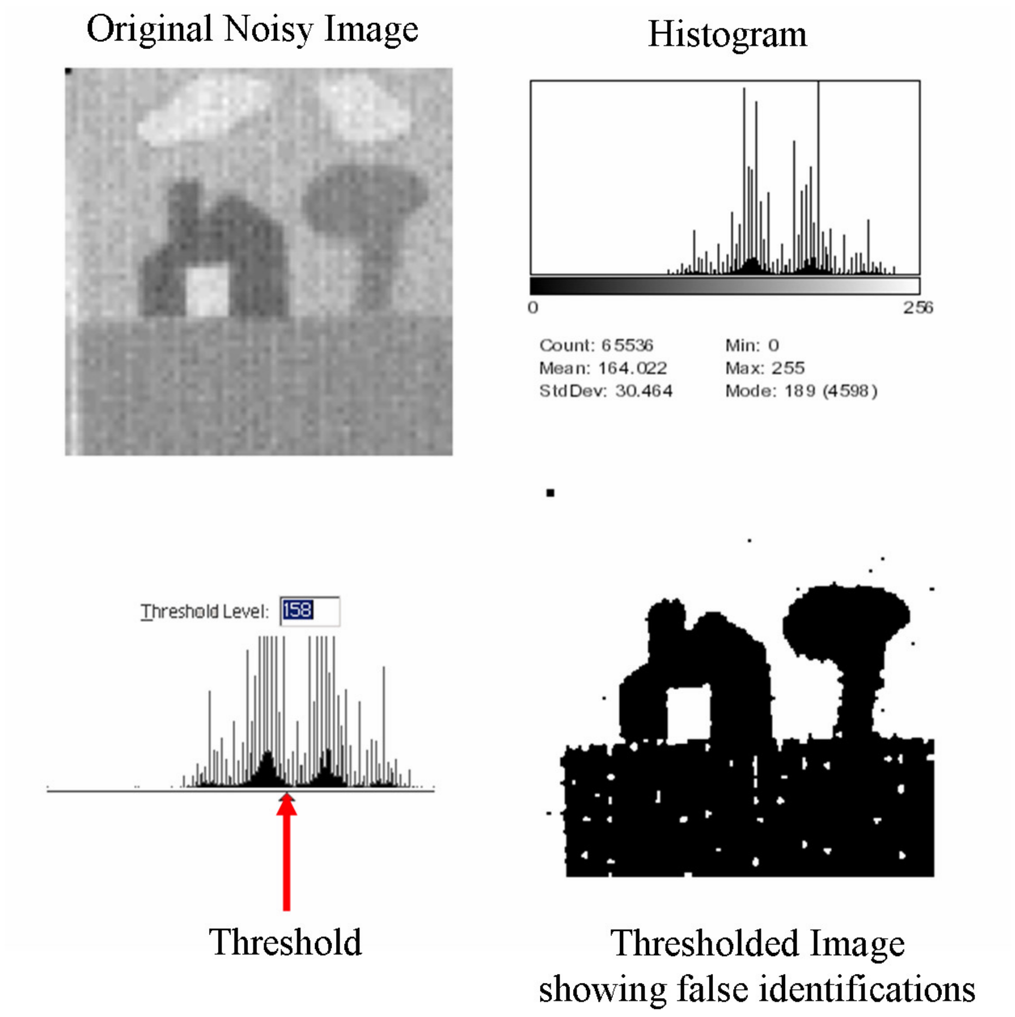
$$\begin{aligned} g_2[f_1[f]] = g_2\left[\sqrt{f}\right] &= f \\ \Longrightarrow g_2[x] &= x^2 \end{aligned}$$

5.5 Application of Histograms to Image Segmentation

Obviously, histograms may be used to distinguish among objects in the image that differ in gray level; this is the simplest example of segmentation in a *feature space.* Consider the bimodal histogram that often indicates the presence of a brighter object on a darker background. A gray value f_T may be determined form the histogram and used as a threshold to segment the "foreground" object. The threshold lookup table maps all pixels with gray levels greater than f_T to white and all others to black. If the histogram clusters are disjoint and the threshold is well chosen (and if the image really DOES contain a bright foreground object), a binary image of the foreground object will result. In this case, the histogram likely is composed of two overlapping Gaussian clusters, and thus some pixels likely will be misclassified by the threshold. Segmentation based on gray level only will be imperfect; there will be false positive pixels (background pixels classified as foreground), and false negative (foreground classified as background). In the crude 64×64 5-bit image shown below, several objects are distinguishable, but the histogram exhibits only two obvious clusters ("dark" and "bright"). Segmentation based on this histogram will be unsatisfactory. A theme of the study of image processing operations will be to improve segmentation by gathering or processing data to create histograms with compact and distinguishable clusters.



Bimodal histogram, showing the intermixing of the "tails" of the clusters for the two object classes, which produce false identifications in the image created by the thresholding lookup table.



Segmentaion of noisy image using histogram that contains four "obvious" clusters of pixels. The histogram is thresholded at $f = 158$, which segmented the sky, clouds, and door from the grass, house, and tree, but some white pixels appear in the grass ("false positives") and some black pixels in the sky ("false negatives")

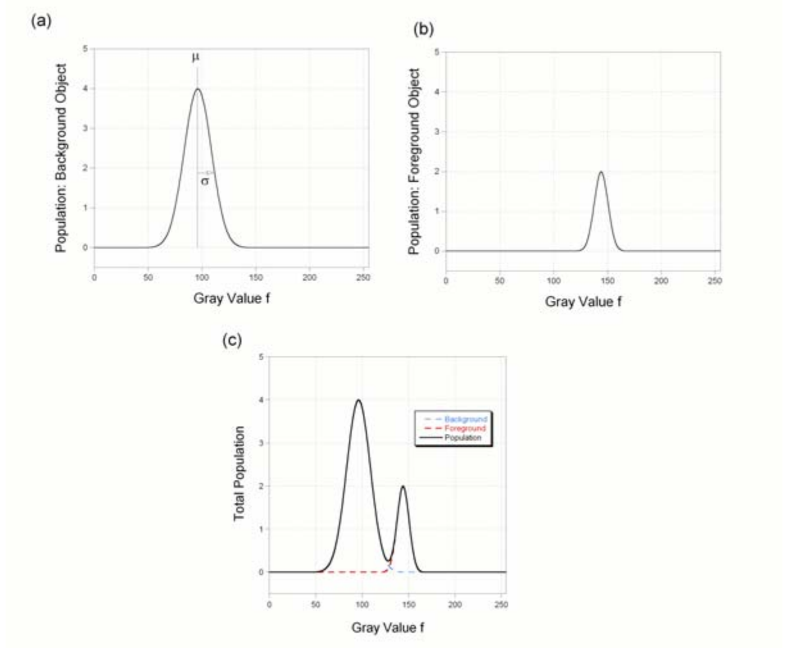
Mahalanobis Distance

The relative separation of the histogram peaks (and thus the ability to segment by thresholding) is sometimes characterized by a metric known as the "Mahalanobis distance." It is used to estimate the likelihood that a "test pixel" belongs to a specific set of pixels (e.g., to the foreground or to the background). If we specify the "location" of the pixel as its gray value f and the mean and standard deviation of the histogram peak of the class n of pixels (e.g., the foreground might be class 1 and the background would be class 2) as μ_n and σ_n, respectively. then we can think of the Mahalanobis distance in an intuitive way. Calculate a normalized distance on the histogram to be:

$$d_n \equiv \frac{f - \mu_n}{\sigma_n}$$

The minimum of the set of distances $\{d_n\}$ determines the most likely class to which the pixel might belong.

Other nonlinear mappings may be used for segmentation. For example, the upper LUT on the left maps background pixels to black and foreground pixels to their original gray level. The other is a *level slicer*; gray levels below f_1 and above f_2 map to zero while those with $f_1 < f\,[x, y] < f_2$ are thresholded to white.



Since a very common task of digital image analysis is segmenting object classes, this indicates that a goal is to find an image histogram where the clusters are "narrow" and spaced "far apart;" so that the Mahalonobis distance is (very) large. A recurring theme in the rest of the course will be methods for increasing the Mahalonobis distance, either by processing the existing image or by rendering the image in a different way..

Chapter 6

Point Operations on Multiple Images

$$g[x,y] = \mathcal{O}\{f[x,y,t_n]\}$$
$$g[x,y] = \mathcal{O}\{f[x,y,\lambda_n]\}$$
$$g[x,y] = \mathcal{O}\{f[x,y,z_n]\}$$

The output pixel $g[x,y]$ is a function of the gray value of that pixel in several input images. The input frames may differ in time, wavelength, depth (if they are slices of a 3-D scene), resolution, etc. Most commonly, the gray values of the multiple inputs are combined by arithmetic operations (e.g. addition, multiplication); binary images (i.e. two gray values) may be combined via logical operators (e.g. AND, XOR, etc.). It is also very common to generate a multidimensional histogram from the multiple inputs and use the interpreted data to segment the image via multispectral thresholding.

Applications:

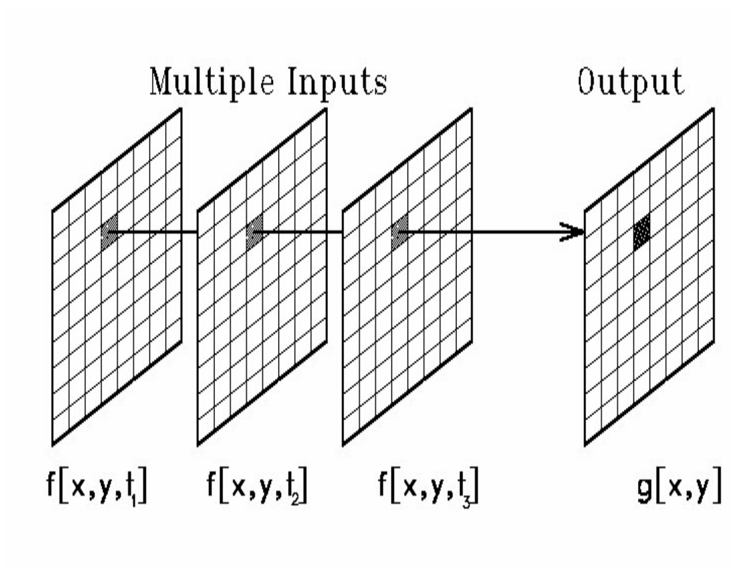
1. Image segmentation using multispectral information

2. Averaging multiple frames for noise reduction

3. Change detection by subtraction

4. Windowing images by mask or template multiplication

5. Correct for detector nonuniformity by division

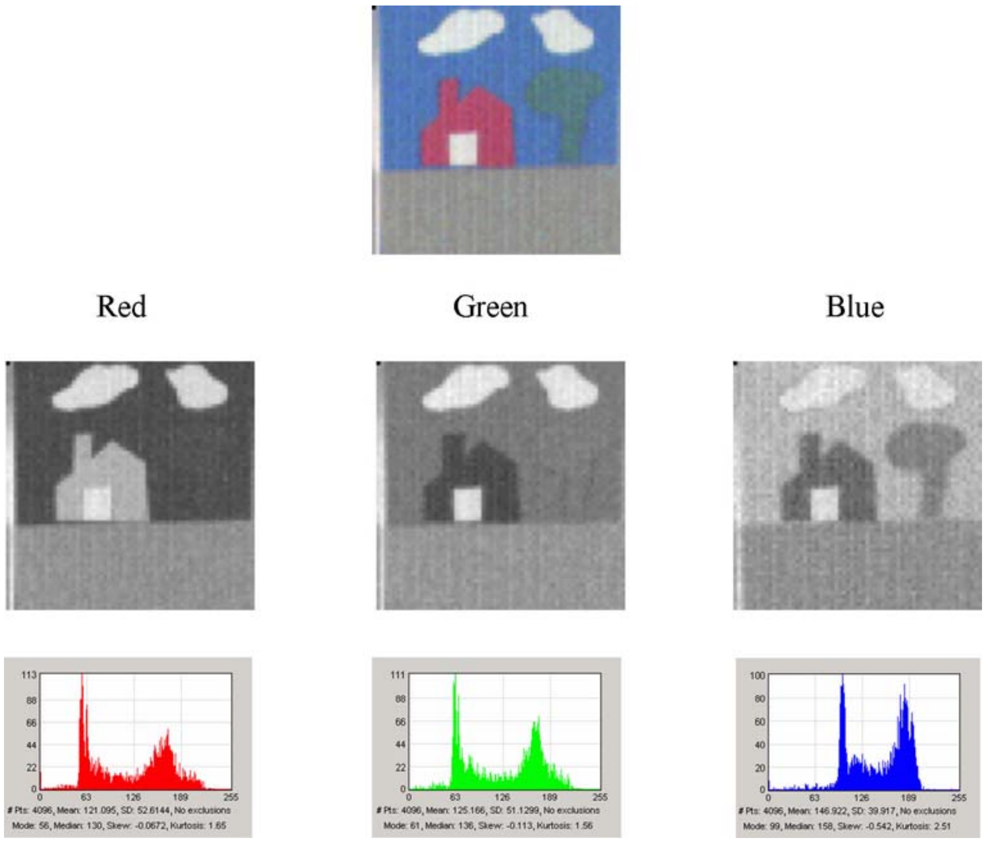


6.1 Color and "Spectral" Images

You are no doubt already familiar with the concept of combining images taken in red, green, and blue light to make a color image. If we call the three images $f[x, y, \lambda_n]$, where n is an index referring to red, green, and blue, then the color image may be written as:

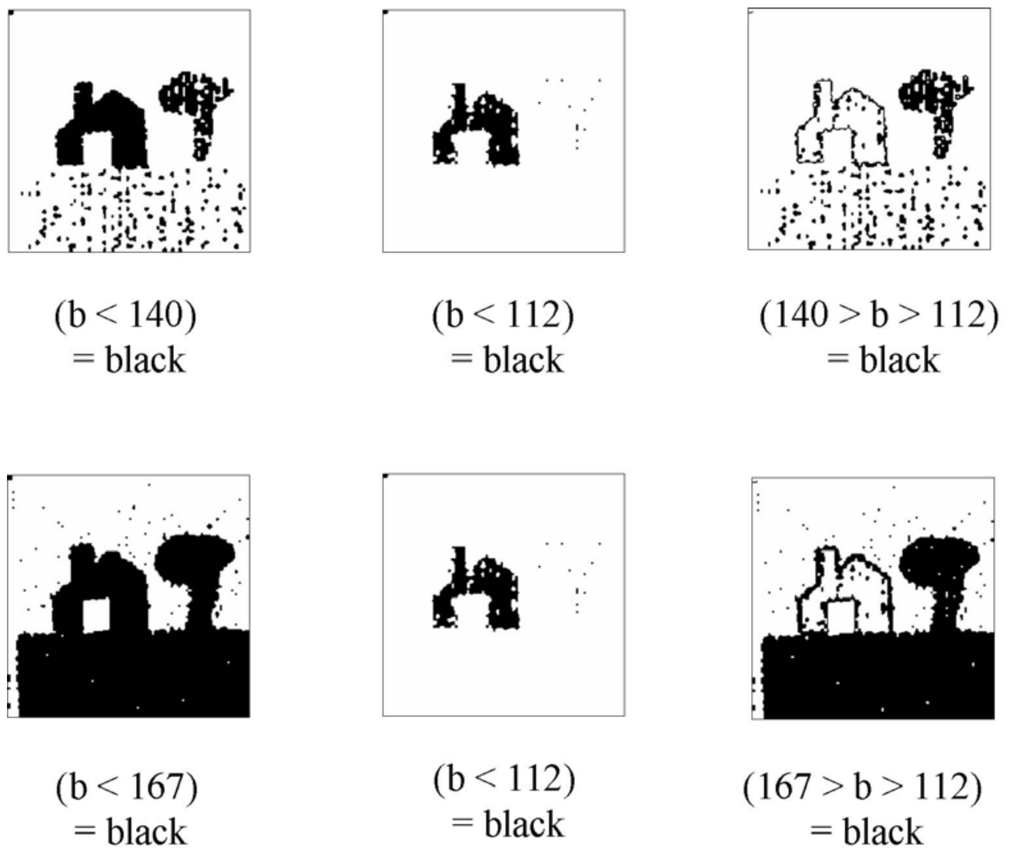
$$\begin{aligned} g[x, y] &= \alpha \cdot f[x, y, \lambda_r] + \beta \cdot f[x, y, \lambda_g] + \gamma \cdot f[x, y, \lambda_b] \\ &\equiv \alpha \cdot f_R[x, y] + \beta \cdot f_G[x, y] + \gamma \cdot f_B[x, y], \end{aligned}$$

where the coefficients $[\alpha, \beta, \gamma]$ are weighting constants determined by the original spectral filtration and sensitivity of the recording process. Normally, the gray values of the same pixel in the three images are different, but values for nearby pixels are typically correlated. For example, a red object will be bright in the red image and darker in the green and blue. The component images for a crude color image in red-green-blue (RGB) space is shown below; note that the histograms do not exhibit easily distinguishable clusters which we already know means that it will be difficult to segment the objects from the image.



Simple 3-band color image, the individual monochromatic images, and their histograms. Note that the gray values of the "tree" are approximately equal to those of "sky" in the red band and approximately equal to those of both "sky" and "grass" in the green band. The tree is difficult to segment from a single image.

None of the 1-D histograms exhibits distinct pixel clusters for each of the objects (house, tree, grass, sky, and white clouds with the door), which leads to the conclusion that the five objects cannot be segmented from any of these single gray-scale images. That said, it is clear that the spectral reflectance of the pixels belonging to "house" is significantly different from that of the other objects, which means that the "house" pixels may be segmented from the other objects fairly easily in a single image, (e.g. by a simple threshold in the blue image, as shown in the figure for a threshold level of 112). However, the reflectance of the pixels belonging to tree and to sky in the red image are virtually identical (since the tree "disappeared" into the sky in that image), and is only slightly different in the blue image. We will have no luck segmenting tree from sky in the red image, and results of attempts to do so in the blue image are shown:



Two attempts to segment "tree" from blue image alone. The first sets all pixels to black in the range $112 \leq b \leq 140$*, which produces a very noisy "tree." Pixels in the second image are thresholded to black if they lie in the range* $112 \leq b \leq 167$*. The segmented pixels include the "grass."*

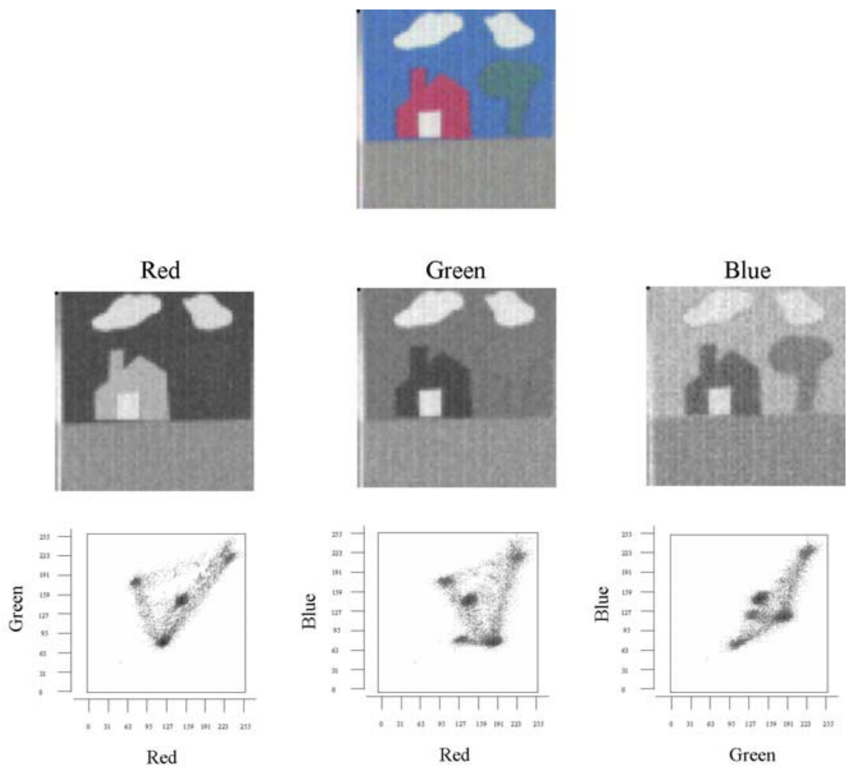
The results are quite "noisy," with lots of incorrectly identified pixels., so we need to find a different way.

6.2 Multispectral Histograms for Segmentation

To successfully segment the "Tree" pixels from the color image, it is very helpful to combine the information in two or more colors to produce histograms with larger Mahalonobis distances. One way is to use a multidimensional (2-D or 3-D) histogram generated from the triplet $[f_R, f_G, f_B]$ of gray values at each pixel. Often only two colors are used to generate a 2-D histogram because of the difficulty of displaying three dimensions of information by conventional means. For example, pixels having a particular gray level f_R in the red image and a level f_G in the green image are counted in bin $[f_R,\ f_G]$. The resulting matrix is the image of a 2-D feature space, i.e., the bin with the largest number of pixels can be displayed as white and unpopulated bins as black. The histogram can be used for image segmentation as before, but the extra information obtained from the second color usually ensures better results.

6.2.1 Multidimensional Histograms of Color Images

Since the color image is "three-dimensional", the corresponding histogram should have three Cartesian axes: "R", "G", and "B". This is difficult to display, so we shall introduce the idea by looking at the histograms of pairs of color bands. The 2-D histograms ("scatterplots") of pairs of the grayscale images of the "house-tree" scene were plotted using the *Hypercube* Image Analysis Software (available for free from *http://www.tec.army.mil/Hypercube/*).



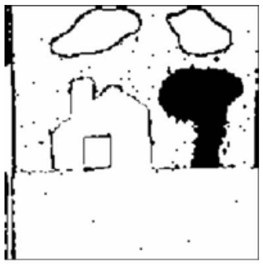
The three 2-D histograms of each pair of channels of the three color bands. You should be able to identify the objects in each of the histograms.

Note the cluster of pixels with large gray values (i.e., towards the upper right corner) in all images, which are generated by the pixels that are bright in all three bands (the white clouds and door). From the appearance of the single bands, you should be able to identify clusters for the house, tree, sky, etc. The image can be segmented by setting pixels within certain intervals of red, green, and blue to white and the others to black. In other words, we threshold pixel clusters within the histogram.

The multidimensional histogram concept may be extended easily any number of dimensions, though visual representation becomes difficult even for three. The concept of the multidimensional histogram is the basis for multispectral segmentation in many areas of image processing.

Segmentation from 3-D Histogram

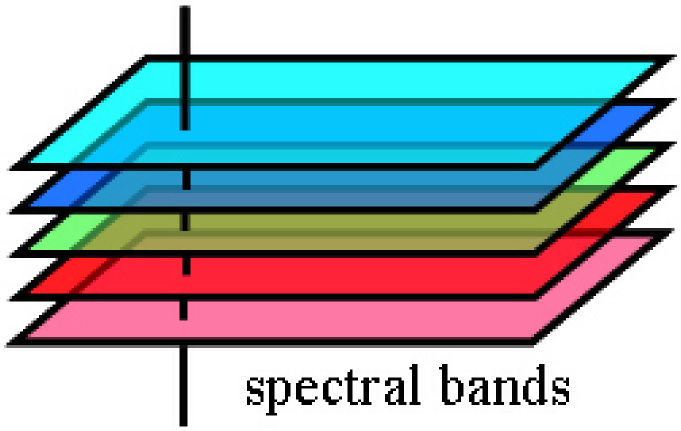
It is often easier to identify distinct and well-separated clusters from the multidimensional histogram rather than from the individual images. The tree segmented from the B-G histogram is shown:



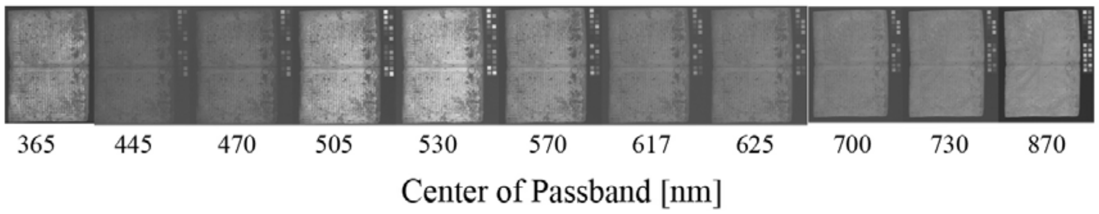
Segmentation of "tree" pixels from 3-D histogram of RGB image. The black pixels satisfy three constraints: $r \geq 106, g \geq 195,$ *and* $b \leq 167$.

6.2.2 Spectral Images

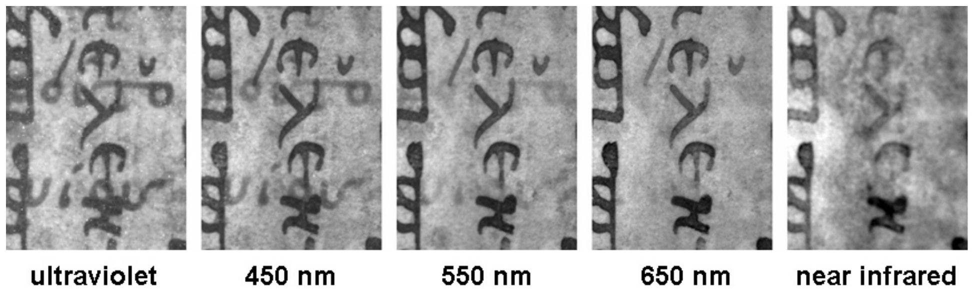
It has become increasingly common to collect images in more than three spectral bands, including bands that are invisible to the human eye. Astronomers utilized the idea for many years to classify stars based on color by measuring the brightness on photographic plates taken through color filters. The technology has advanced to the point where the technique is now quite common and is variously called "multispectral" or "hyperspectral" imaging. The dividing line between these two classes is historical and is based on the fact that it was difficult to collect registered images in many spectral bands. Loosely speaking, "multispectral" imaging involves images in 3-10 fairly broad passbands, while hyperspectral images have many more (often 200 or more) and narrower bands. I try to avoid the controversy by just calling it "spectral" imaging..



Concept of a spectral image; registered (i.e., "aligned") images are collected in multiple bands of wavelengths.



Sequence of monochrome images of a page of the Archimedes palimpsest taken under illumination at different wavelengths from 365 nm *(in the near-ultraviolet region of the spectrum) to* 870 nm *(in the near-infrared region). Note the variation in appearance with wavelength.*

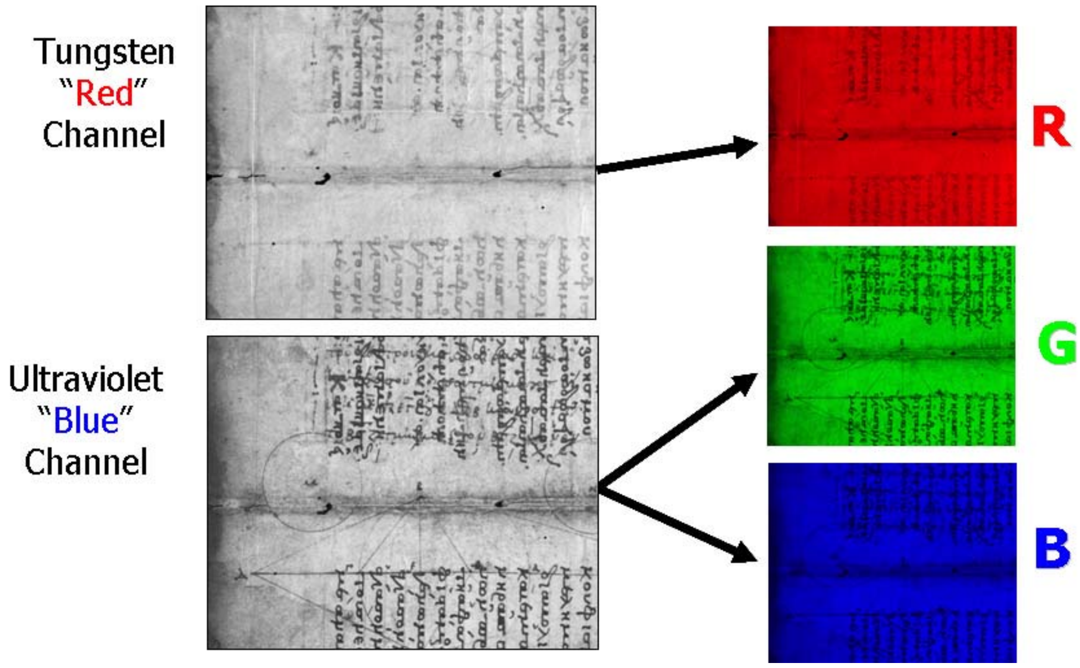


Images of a small section of a page at different wavelengths, showing the variation in appearance due to the variation in spectral reflectivity of the various object classes.

The additional bands complicates the display and analysis of the histogram without a high-powered computer. Several different analysis techniques are available; we will focus on principal component analysis (PCA) and introduce it using the simplest 2-D case.

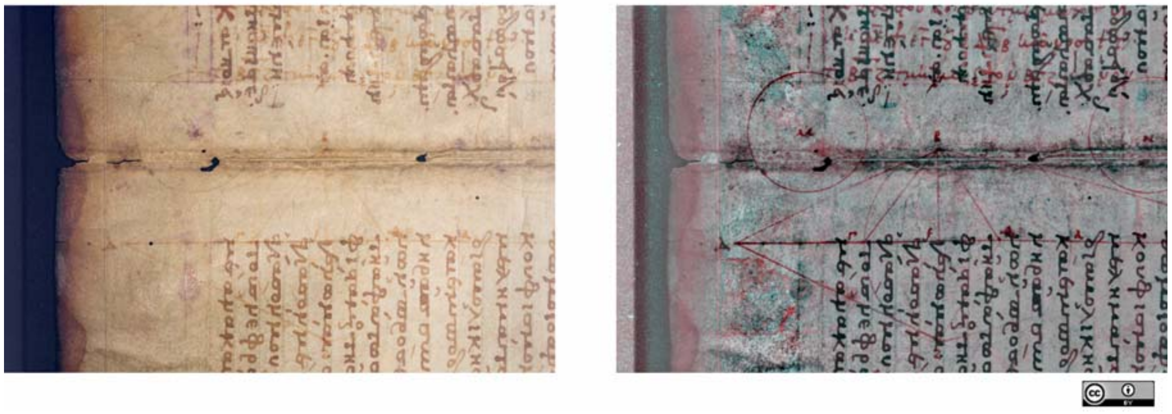
Example: Pseudocolor Rendering of the Archimedes Palimpsest

One of the most successful tools applied to assist the reading of the original texts in the 10^{th}-century parchment manuscript known as the Archimedes Palimpsest rendered the two texts in "pseudocolor." The technique is based on two observations: that the "reddish" traces of erased text virtually disappear if viewed under red illumination, while the later overtext remains visible, and that both texts are visible in the visible fluorescent emission when the page is illuminated by ultraviolet light. The pseudocolor image is created by combining the red band of a color image under low-wattage tungsten illumination (a very "reddish" light) with the blue channel of a color image under ultraviolet illumination. Though not discussed here, both images are "preprocessed" to enhance the contrast and balance the average gray value. The image rendering scheme is shown in the figure:



Schematic of pseudocolor rendering of the Archimedes palimpsest (Creative Commons Attribution License, Copyright retained by the Owner of the Archimedes Palimpsest).

Since the pixels corresponding to the original undertext are "bright" in the red channel and darker in the blue and green, these pixels appear "reddish" in the pseudocolor image, while the overtext pixels are dark in both images and appear neutral "gray" or "black" in the final image. The result of the process is shown by comparison to the visual appearance; note that the color of the original text provides a "cue" to the scholar of the source of the text; this is perhaps most visible near the top of the page.

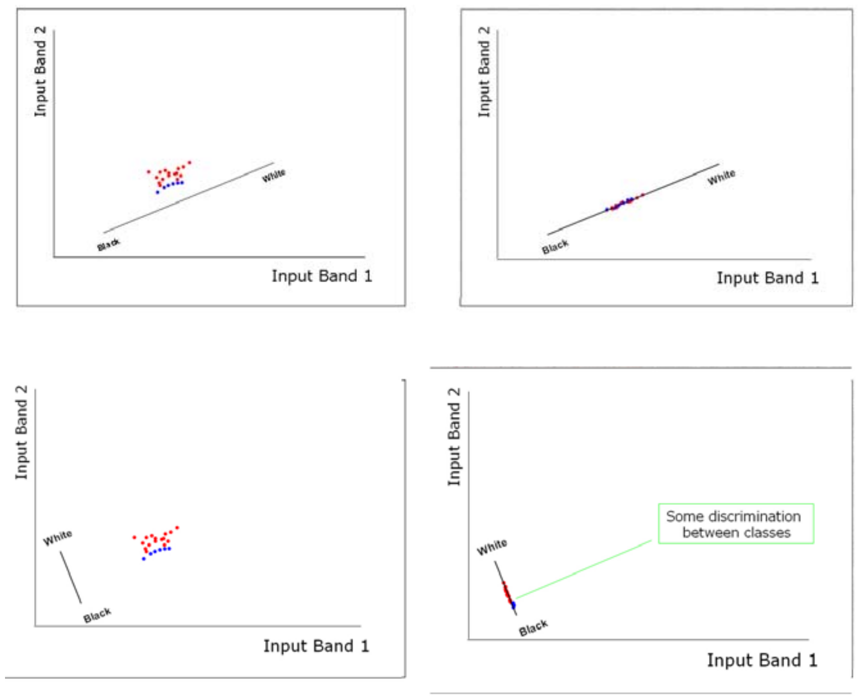


Comparison of visual appearance of gutter region of f.093v-092r (left) with pseudocolor rendering (right), which shows the color reddish "cue" for the scholars about the source of the original text (Creative Commons license, copyright retained by the Owner of the Archimedes Palimpsest)

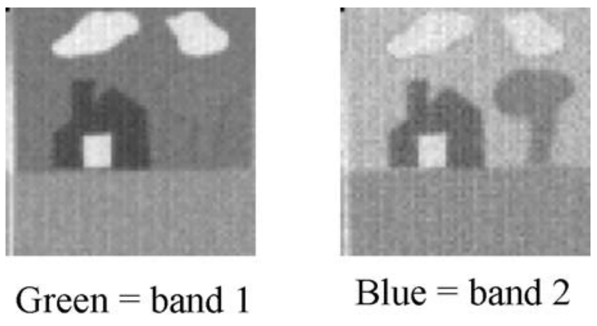
6.2.3 Principal Component Analysis – PCA

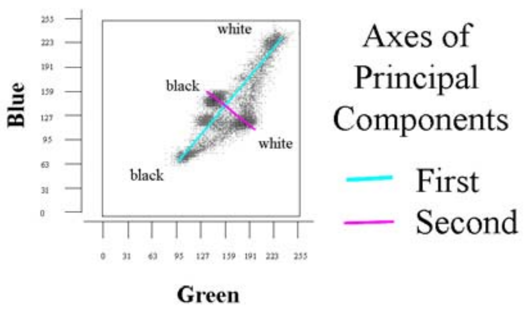
Reference: Schowengerdt, Remote Sensing

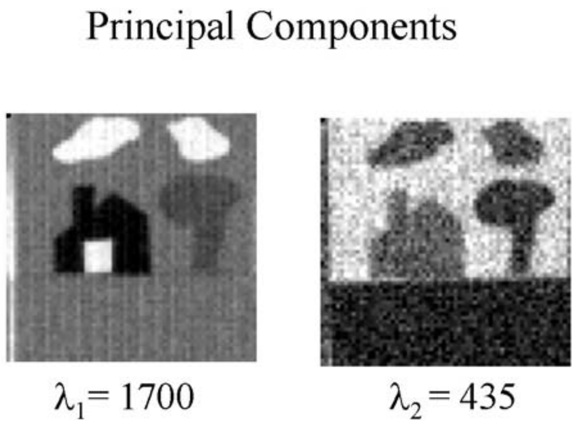
The information in the different bands of a multispectral image (e.g., an RGB color image) often are highly correlated, meaning that some or many bands may be visually similar. In other words, the information content of a multispectral image often is quite "redundant." It often is convenient to reduce or even eliminate this redundancy by expressing the image in terms of "axes" other than the original "bands." The data image is transformed to a new coordinate system by rigidly rotating axes to align with these other "directions" and the image data then projected onto these new axes. In principal components, the rotation produces a new multispectral image with a diagonal covariance matrix, so that there is no covariance between the various bands. One axis of the rotated coordinate system is aligned with the direction of maximum variance of the image, the second axis is perpendicular to the first and aligned with the direction of the second largest variance, etc. The bands in the principal component image are thus arranged in order from largest to smallest variance.



For example, consider a 2-band image that was created from the blue and green bands of the simple image, with zeros inserted into the red band. The 2-D histogram, blue vs. green, also is shown. The principal components were evaluated using "Hypercube." Since the third component image was black, only two principal components are needed to fully represent the two-band image. In the outputs, note that the "lightest" objects in the image, the clouds and door, appear in the first PC as "white," while the darkest structure in the two images is the house, which appears "black" in the 1st PC. In the second PC, the door has "faded into the house" because its projected gray value is in the mid range along this axis.

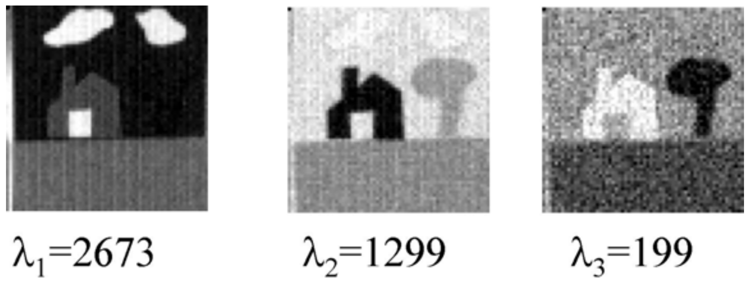






Principal components of a two-band image, computed in "Hypercube": the image is created by inserting zeros in the red band. The 2-D histogram blue vs. green is shown. The first principal component projects the gray values onto the "cyan" axis that includes most of the image variance. The second principal component projects the data onto the magenta axis. The images are shown. Note that the first principal component shows the clouds and door as white and the house as black. The door is not visible in the second principal component, as its gray value is the same as that of the house.

The principal components of the 3-band RGB image are also shown. The "sky" is quite dark in the first PC of the 3-D scene – the maximum contrast is between the sky and clouds in the 3-band histogram. Note that the 3rd PC is quite noisy – this image depicts the smallest range of variance, where the image noise dominates.



The three principal components of the RGB house+tree image. Note that the third component image is quite noisy.

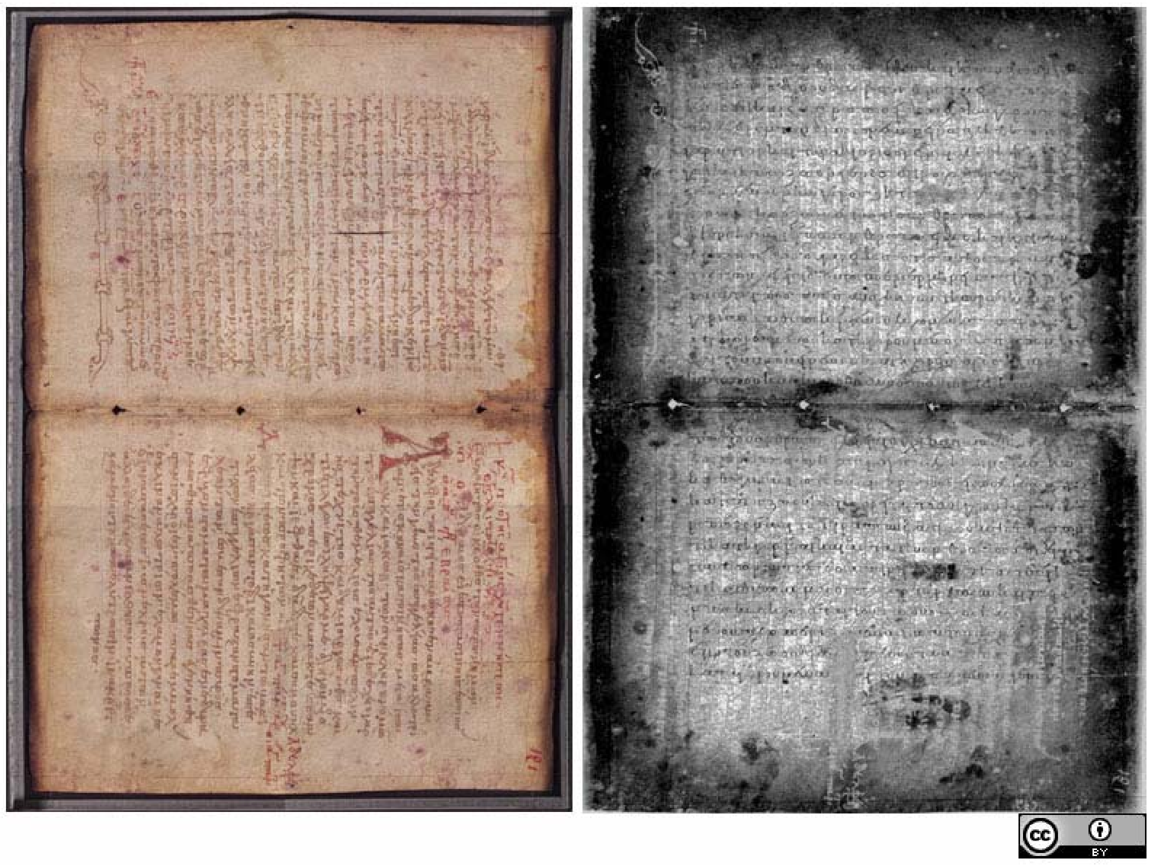
Since PCA decomposes the N bands of data into orthogonal (i.e., independent) components, the high-order bands (with small variances) of many realistic scenes are dominated by small random fluctuations ("noise") in the data. This reason for this effect is that the N bands of image generally are not truly independent, but rather exhibit correlations between the bands. PCA thus provides

Example: PCA Processing of one Treatise in the Archimedes Palimpsest

The text of one of the original manuscripts in the Archimedes Palimpsest did not respond to the pseudocolor rendering. These pages comprised a commentary by Alexander of Aphrodisias on Aristotle's "Categories." Virtually the only characters that indicated the source of the text spelled out "Aristotle" in the gutter of one page.

The text on these pages was eventually revealed by principal component analysis applied only to the RGB image collected under ultraviolet illumination. This fluorescent image is dominated by blue emission, so the blue band is noticeably brighter than the green, which is in turn brighter than the red. Therefore, the first PCA band (largest variance) approximates the blue channel of the fluorescence image. The second and third bands that are orthogonal to the first convey the low-variance text information

ENVI eigenvectors – first vector is first row, ordered by band



Comparison of original appearance of f.120v-121r of the Archimedes Palimpsest to PCA band 3 from the RGB image under UV illumination, showing the visibility of the original text. (Creative Commons Attribution License, Copyright retained by the Owner of the Archimedes Palimpsest).

The components of the three eigenvectors ordered by projection on the red, green, and blue axes, respectively are:

$$PCA_1 \implies \begin{bmatrix} \textit{projection onto "red" axis} \\ \textit{projection onto "green" axis} \\ \textit{projection onto "blue" axis} \end{bmatrix} = \begin{bmatrix} +0.366 \\ +0.447 \\ +0.829 \end{bmatrix} ;\ |PCA_1| = 1$$

$$PCA_2 \implies \begin{bmatrix} +0.857 \\ +0.220 \\ -0.467 \end{bmatrix} ;\ |PCA_2| = 1$$

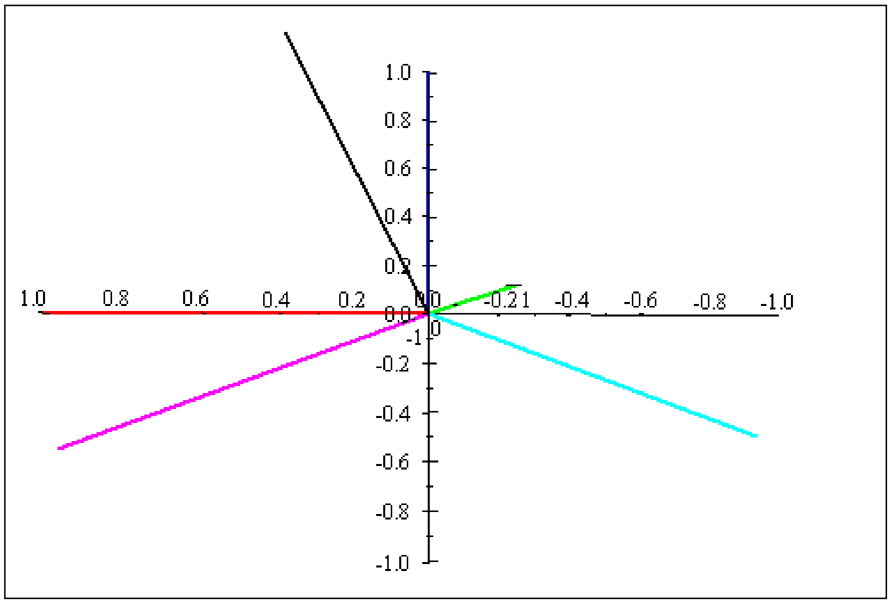
$$PCA_3 \implies \begin{bmatrix} -0.391 \\ +0.867 \\ -0.308 \end{bmatrix} ;\ |PCA_3| = 1$$

As expected, the first principal component is pointed within the octant in the 3-D volume defined by the positive color axes, which means that the pixel values in first PCA are weighted *sums* of the red, green, and blue pixel values (all projections are positive). Since the three vectors are scaled to have unit length, the three components are the direction cosines of the lines measured relative to the three axes, so the angles of a PCA axis measured from the color axes may be evaluated via the inverse cosine function:

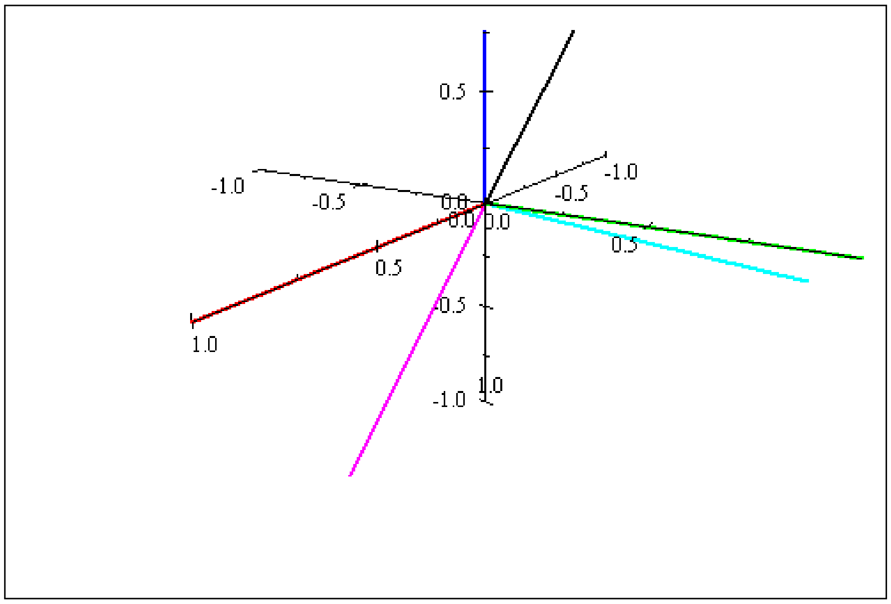
$$\begin{aligned}
\text{angle of } PCA_1 \text{ from blue axis: } & \cos^{-1}\left(\frac{0.829}{1}\right) \cong 0.593 \text{ radians } \cong 34^\circ \\
\text{angle of } PCA_1 \text{ from green axis: } & \cos^{-1}\left(\frac{0.447}{1}\right) \cong 1.074 \text{ radians } \cong 63.5^\circ \\
\text{angle of } PCA_1 \text{ from red axis: } & \cos^{-1}\left(\frac{0.336}{1}\right) \cong 1.228 \text{ radians } \cong 70.4^\circ
\end{aligned}$$

Note that the components of the first eigenvector are ordered with the largest projection in the direction of the "blue" axis (so that the angle of PCA_1 from the blue axis is smallest), the second largest along "green," and the third largest along "red." This confirms the hypothesis that the UV fluorescence is dominated by emission in the blue region of the spectrum, which was expected because this region is closest in wavelength to the UV stimulation at $\lambda_0 = 365\,\text{nm}$.

Two graphical renderings of the three PCA eigenvectors are shown in the plots, where the Cartesian red, green, and blue axes are shown in their corresponding colors and the three PCA bands are rendered in black, magenta, and cyan, respectively



One rendering of axes of 3-band PCA of Alexander page under ultraviolet illumination compared to RGB axes: PCA band 1 in black, band 2 in magenta, and band 3 in cyan. Note that PCA band 1 (in black) is a weighted sum of RGB and is pointed more in the direction of the blue axis,

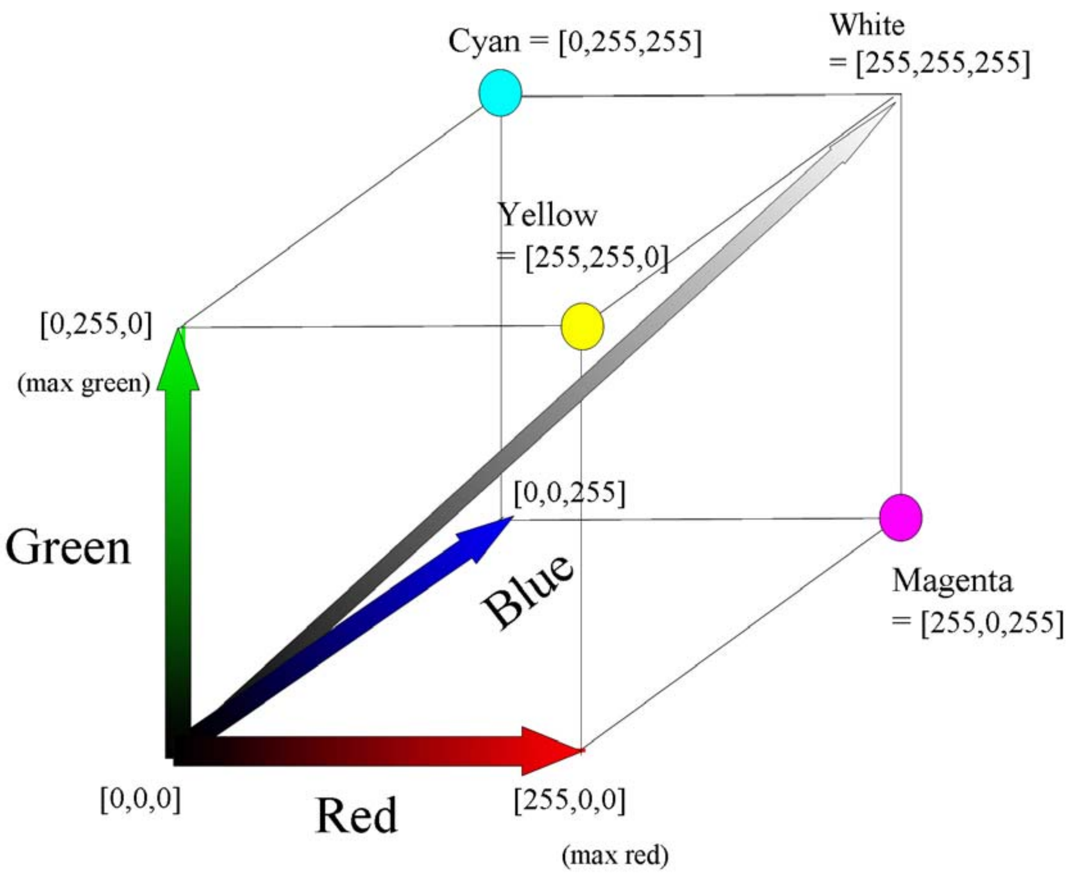


Different orientation of display for axes of 3-band PCA of Alexander page under ultraviolet illumination compared to RGB axes: PCA band 1 in black, band 2 in magenta, and band 3 in cyan.

6.3 Color Spaces

6.3.1 Red, Green, Blue

This representation is usually graphed in a Cartesian system analogous to $[x, y, z]$, as shown:



RGB coordinates (8 bit) displayed as a Cartesian system. Locations with the same value in each coordinate are "neutral", e.g., $[0, 0, 0]$ *is "black",* $[255, 255, 255]$ *is "white", and others are "gray." The additive primaries (red, green blue) form the Cartesian axes and the "subtractive" primaries are sums of pairs of additive primaries: magenta = red + blue; yellow = red + green, cyan = green + blue.*

6.3.2 Hue, Saturation, Lightness (or Brightness, or Value):

This is the representation that led to Thomas Young's theory of color in the early 1800s:

- *Hue* corresponds to the common definition of color, e.g., "red", "orange", "violet" etc., specified by the *dominant wavelength* in a spectrum distribution, though a "dominant" may not actually be present

- *Saturation* (also called *chroma*): an expression for the "strength" or "purity" of a color. The intensity of a very saturated color is concentrated near the dominant wavelength. Looked at another way, saturation is measured as the amount of "gray" in proportion to the hue. All colors formed from one or two primaries have 100% saturation; if some amount of the third primary is added to a color formed from the other two, then there is at least some "gray" in the color and the saturation decreases. The saturation of a pure white, gray, or black scene (equal amounts of all three primaries) is zero. A mixture of a purely saturated color (e.g., "red") and white produces a "desaturated red", or "pink". Saturation is reduced if surface reflections are present.

$$\begin{array}{ccc} R & & 0 \\ G & = & 204 \\ B & & 153 \end{array} \Longrightarrow \begin{array}{ccc} H & & 116 \\ S & = & 255 \\ L & & 102 \end{array} \Longrightarrow S = \frac{255}{255} = 100\%$$

$$\begin{array}{ccccccc} R & & 25 & & 25 & & 0 \\ G & = & 204 & = & 25 & + & 179 \\ B & & 153 & & 25 & & 128 \end{array} \Longrightarrow \begin{array}{ccc} H & & 115 \\ S & = & 199 \\ L & & 115 \end{array} \Longrightarrow S = \frac{199}{255} = 78\%$$

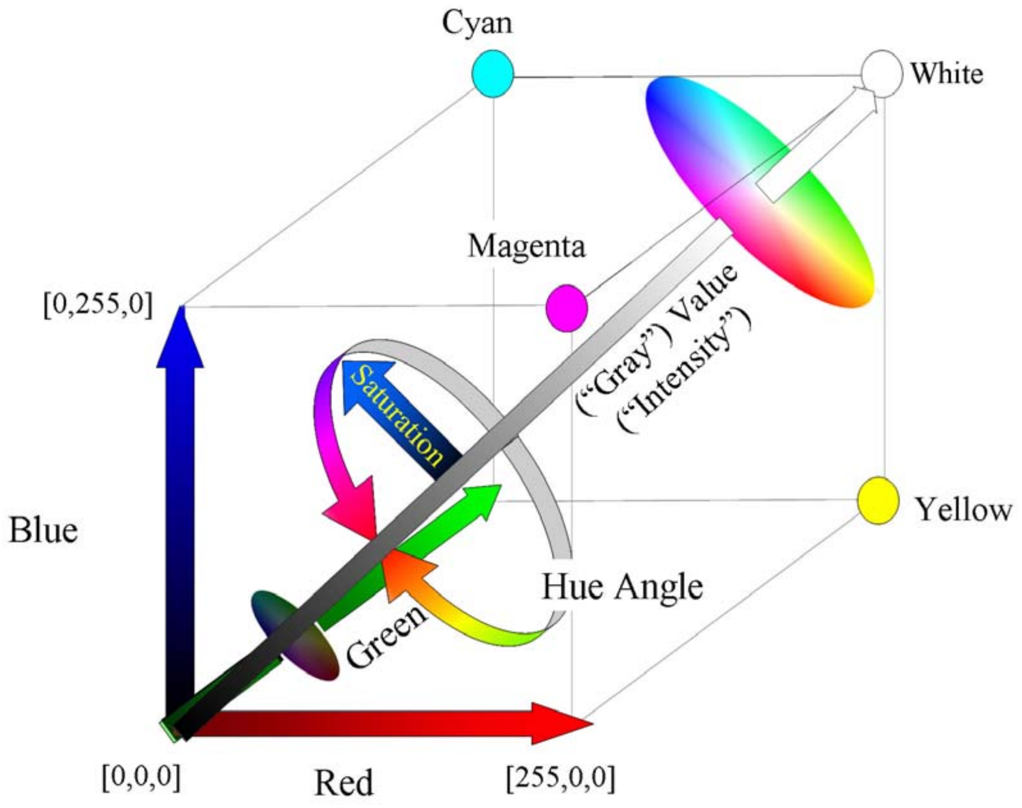
$$S = \frac{hue - gray}{hue} = \frac{115 - 25}{115} = 78\%$$

In more scientific terms, the saturation is the relative bandwidth of the visible output from a light source. A source with a narrow bandwidth emits a "purer" color and thus has a large saturation. As saturation decreases, colors appear more "washed-out".

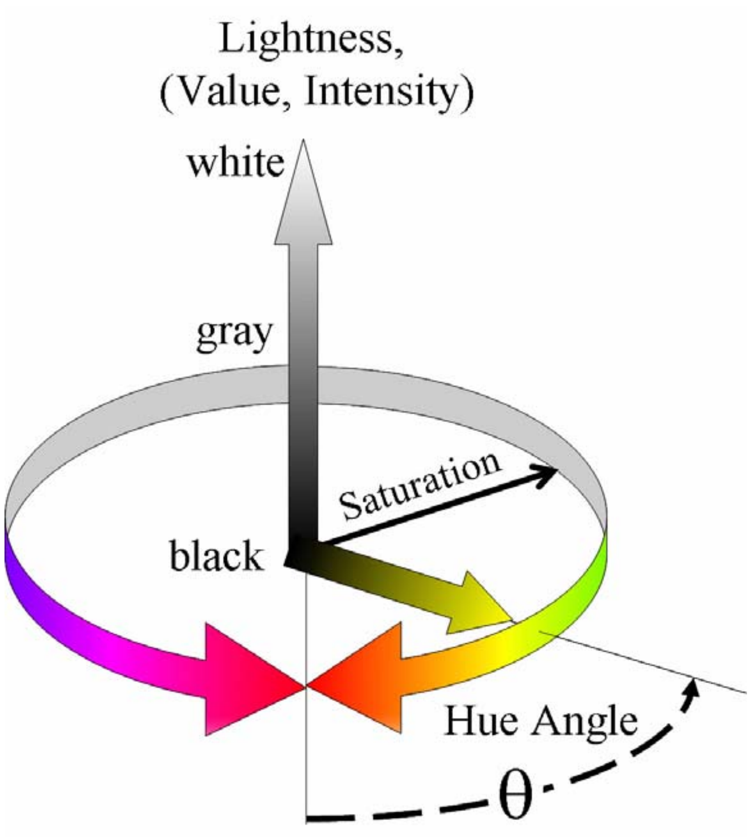
- *Brightness*: sensation of intensity of a light, from dark through dim to bright.

- *Lightness*: a relative expression of the intensity of the energy output reflected by a surface; "blackness", "grayness", "whiteness" of a visible light source. It can be expressed as a total energy value or as the amplitude at the wavelength where the intensity is greatest.

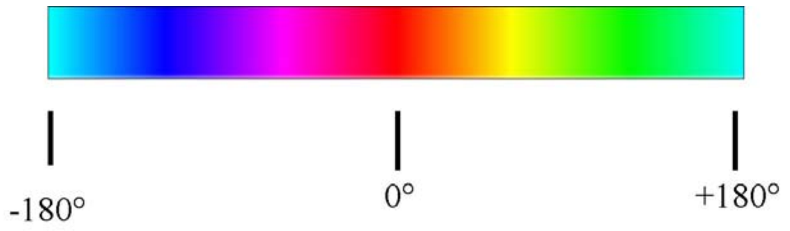
HSB is often represented in a cylindrical coordinate system analogous to (r, θ, z). The saturation coordinate is plotted along the radial axis, the hue in the azimuthal direction, and the lightness as the vertical (z) axis. The hue determines the frequency of light or the position in the spectrum or the relative amounts of red, green and blue. The hue is a continuous and periodic scale that often is measured in an "angle" in angular degrees (e.g., the "hue angle"), though it also may be normalized to be compatible with 8-bit numerical representations. The hue located at the extrema (e.g., angles of $\pm 180°$) are identical, as shown in the figure from the hue adjustment in Adobe Photoshop$^{\text{TM}}$. Microsoft Windows color dialogs use HSB but call the third dimension "luminosity" or "lightness". It ranges from 0% (black) to 100% (white). A pure hue is 50% luminosity, 100% saturation. The hue angles are shown, where red corresponds to an angle of $0°$.



Schematic view of RGB and HSV coordinate systems: the RGB system is Cartesian and HSV is cylindrical with the "value" axis along the diagonal "gray" axis in RGB. The radial distance from the "gray" axis defines the "saturation" (two examples near the ends of the "lightness" axis are shown) and the azimuth angle about the "lightness" ("gray") axis is the "hue angle."

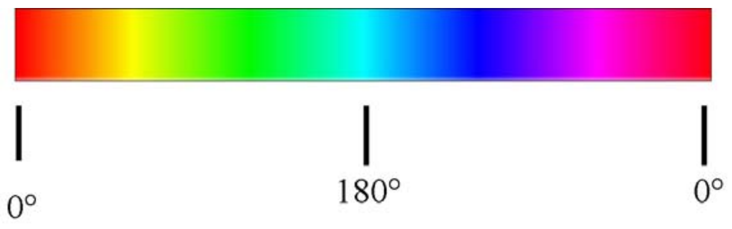


Coordinate system for "Hue, Saturation, Lightness (value, intensity)" which corresponds to the cylindrical system $(r, \theta, z) \to (S, H, L)$



Hue angle representation used in Adobe Photoshop^{TM}. The hue at angle $0°$ is "red" , "blue" is at $\theta = -120°$, "magenta" at $\theta = -60°$, yellow at $\theta = +60°$, and green at $\theta = +120°$; the hues at at $\theta = \pm 180°$ are identically "cyan".

The hue may be rotated about the azimuth and "wraps around" at cyan ($\theta = 180°$).



The hue axis after rotation by $180°$, showing the "wraparound" at the edge of the axis.

Hue is represented by 8-bit integer values in other applications, such as Powerpoint$^{\text{TM}}$. A list of the primary colors for different hue angles is shown in the table.Note that the additive primaries Red, Green, and Blue are located at $0°$ and $\mp 120°$, while the subtractive primaries Magenta, Yellow, and Cyan are at $\mp 60°$ and $180°$. Colors at opposite sides of the hue circle (separated by $180°$) are *complementary*, so that the sum of two complementary colors produces white.

The sum of monochromatic yellow ($\lambda = 580\,\text{nm}$) and monochromatic blue ($\lambda = 480\,\text{nm}$) produces white light that looks just as while as the sum of all visible wavelengths

Color	Photoshop$^{\text{TM}}$ Hue Angle θ ($°$)	Powerpoint$^{\text{TM}}$ Hue $[0, 255]$
cyan	$\pm 180°$	$127 \Longrightarrow \frac{1}{2}$
green	$+120°$	$85 \Longrightarrow \frac{1}{3}$
yellow	$+60°$	$42 \Longrightarrow \frac{1}{6}$
red	$0°$	0
magenta	$-60°$	$213 \Longrightarrow \frac{5}{6}$
blue	$-120°$	$170 \Longrightarrow \frac{2}{3}$

The table of colors in Photoshop$^{\text{TM}}$ are:

Color	R	G	B	H	S	L
cyan	0	255	255	127	255	128
green	0	255	0	85	255	128
yellow	255	255	0	42	255	128
red	255	0	0	0	255	128
magenta	255	0	255	213	255	128
blue	0	0	255	170	255	128

As another set of cxamples, consider the HSL coordinatges for different "yellows":

Color	R	G	B	H	S	L
yellow	255	255	0	42	255	128
yellow	131	128	0	42	255	64
yellow	129	255	255	42	255	192
yellow	246	240	0	42	255	120
yellow	195	192	64	42	128	128
gray	128	128	128	42	0	128

The RGB model scores on its simplicity, so what are the advantages of the HSL colour model? I think there are several:

- You can generate grey scales using only one parameter - the luminosity when saturation is set to 0.
- You can vary the colour by varying the hue alone such that the brightness remains unchanged
- You can fade or darken several colours, or whole bitmaps, such that the lightness (or darkness) stay in step
- I suppose it comes down to that the HSL model is easier to use visually because it suits the eye, whereas the RGB model is easier to use in programming.

The classic RGB colour space used in GDI+ is excellent for choosing or defining a specific colour as a mixture of primary colour and intensity values but what happens if you want to take a particular colour and make it a bit lighter or a bit darker or change its saturation. For this you need to be able to use the HSL (Hue, Saturation and Luminance) colour space.

6.3.3 Conversion from RGB to HSL

Recall the transformations between Cartesian and cylindrical coordinates:

$$\begin{aligned} r &= \sqrt{x^2+y^2} & x &= r\cos[\theta] \\ \theta &= \tan^{-1}\left[\tfrac{y}{x}\right] & y &= r\sin[\theta] \\ z &= z & z &= z \end{aligned}$$

These cannot be written as matrix-vector products and thus are nonlinear transformations. This observation suggests that the transformation between RGB and HSL also is nonlinear, as is in fact true. The scheme for computing HSL from RGB is:

1. Normalize the three values $[R, G, B]$ to the interval $[0, 1]$;

2. Find the maximum and minimum of the three values for a pixel; these are $color_{\max}$ and $color_{\min}$;

3. If all three RGB values are identical, then the hue and saturation for that pixel are both 0 (the pixel is "gray");

4. Compute the "lightness" L via

$$L = \frac{color_{\max} + color_{\min}}{2}$$

5. Test the lightness value L to find the saturation S using the conditions:

$$\text{If } L < 0.5 \text{ then } S = \frac{color_{\max} - color_{\min}}{color_{\max} + color_{\min}}$$
$$\text{If } L > 0.5 \text{ then } S = \frac{color_{\max} - color_{\min}}{2 - (color_{\max} + color_{\min})}$$

6. Compute the hue H via:

(**a**) If $\text{color}_{\max} = R$ then

$$H = \frac{G - B}{color_{\max} - color_{\min}}$$

(**b**) If $\text{color}_{\max} = G$ then

$$H = 2 + \frac{B - R}{color_{\max} - color_{\min}}$$

(**c**) If $\text{color}_{\max} = B$ then

$$H = 4 + \frac{R - G}{color_{\max} - color_{\min}}$$

7. Convert L and S back to percentages, and H into an angle in degrees (i.e., scale it from $-180° \leq H \leq +180°$).

Example: $[R, G, B] = [25, 204, 53]$

$$R = 25 \rightarrow \frac{25}{255} \cong 0.098,$$
$$G = 204 \rightarrow \frac{204}{255} = 0.800,$$
$$B = 53 \rightarrow \frac{53}{255} \cong 0.208$$

$$
\begin{aligned}
color_{\max} &= 0.800 \\
color_{\min} &= 0.098 \\
L &= \frac{0.8 + 0.098}{2} = 0.449 < 0.5 \\
S &= \frac{0.8 - 0.098}{0.8 + 0.098} \cong 0.782 \\
H &= 2 + \frac{0.208 - 0.098}{0.8 - 0.098} \cong 2.157 \ radians
\end{aligned}
$$

$$
\begin{aligned}
L &= 0.449 \cdot 255 = 115 \\
S &= 0.782 \cdot 255 = 199 \\
H &= \frac{2.157}{\pi} \cdot 180^\circ = 123.6^\circ
\end{aligned}
$$

Conversion from HSL to RGB

1. If $S = 0$, define $L = R = G = B$,Otherwise, test L :

$$
\begin{aligned}
&\text{If } L < 0.5, \text{ then } \alpha = L \cdot (S + 1) \\
&\text{If } L \geq 0.5, \text{ then } \alpha = L + S - L \cdot S
\end{aligned}
$$

2. Set

$$\beta = 2.0 \cdot L - \alpha$$

3. Normalize hue angle H to the range $[0, 1]$ by dividing by 360°

4. For each of R, G, B, compute γ as follows:

$$
\begin{aligned}
&\text{for } R, \quad \gamma = H + \frac{1}{3} \\
&\text{for } G, \quad \gamma = H \\
&\text{for } B, \quad \gamma = H - \frac{1}{3}
\end{aligned}
$$

5.

$$\text{If } \gamma < 0, \text{ then } \gamma = 1 + \gamma$$

6. For each of $[R, G, B]$, do the following test:

$$\begin{aligned}
&\text{If } \gamma < \frac{1}{6}, \text{ then } color = \beta + (\alpha - \beta) \cdot 6 \cdot \gamma \\
&\text{If } \frac{1}{6} \le \gamma < \frac{1}{2}, \text{ then } color = \alpha \\
&\text{If } \frac{1}{2} \le \gamma < \frac{2}{3}, \text{ then } color = \beta + (\alpha - \beta) \cdot (4 - 6\gamma)
\end{aligned}$$

7. Scale back to the range $[0, 100]$

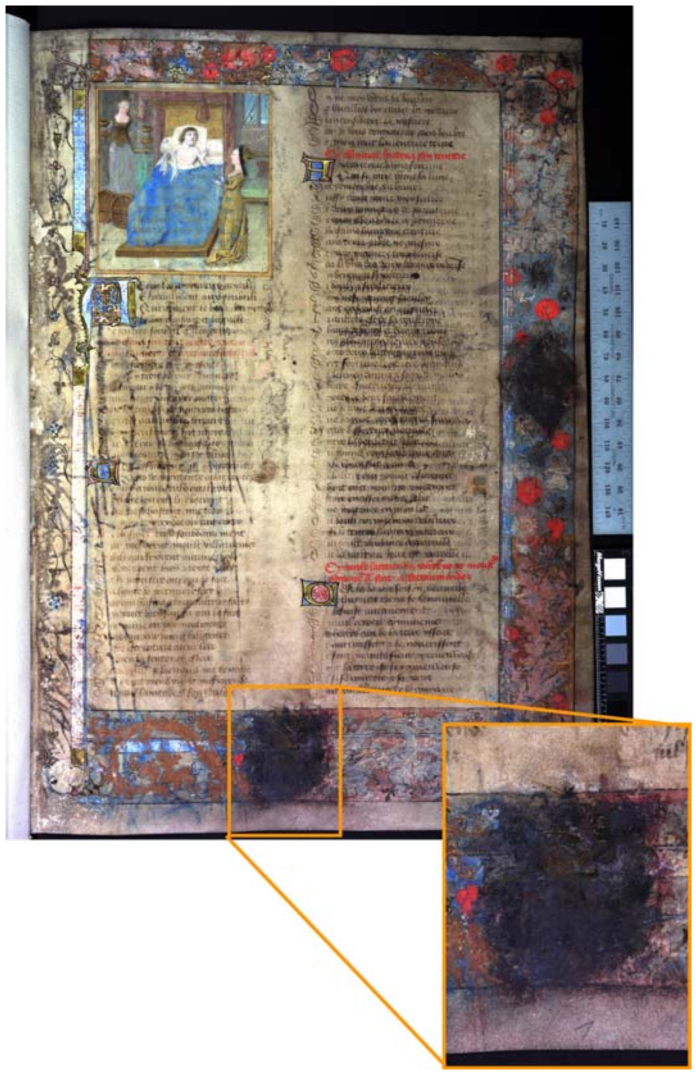
8. Repeat steps 6 and 7 for the other two colors.

Word Descriptors of Colors

- Hue: a “pure” color, i.e. one containing no black or white.
- Shade: a “dark” color, i.e. one produced by mixing a hue with black
- Tint: a “light” color, i.e. one produced by mixing a hue with white
- Tone: color produced by mixing a hue with a shade of grey.

6.3.4 Example: Wax “Coat of Arms” in a French Epic Poem

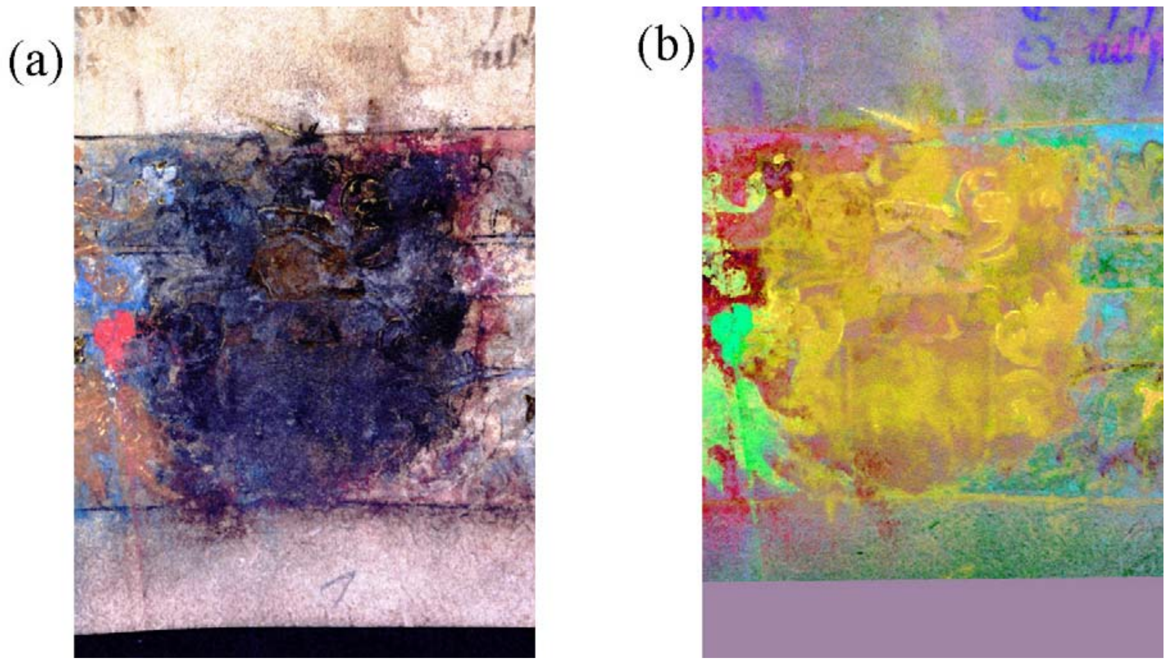
The color spaces just considered may be combined with PCA analysis to make an image tool that is useful for enhancing image features. The 14^{th}-century epic poem “Les Esches d’Amour” (the Chess of Love), written in Middle French, was damaged during the Allied bombing raids on Dresden in February 1945. Though these raids are famous for the firestorm inflicted upon the city, the damage to the manuscript was actually by water from fractured supply mains. The first page of the manuscript includes a wax “Armorial Achievement” (more commonly known as a coat of arms) that is now not decipherable to the eye.



First page of "Les Esches d'Amour" (The Chess of Love), showing coat of arms at the bottom of the page.

In an attempt to enhance the visibility of any internal structure in the coat of arms, the RGB histogram of the color image was equalized. Other than confirming the suspicion that the top of the coat of arms was a unicorn, little additional information was revealed.

Images were collected under 13 bands of illumination from light emitting diodes that spanned the wavelength region from ultraviolet at $\lambda \cong 365\,\text{nm}$ through infrared at $\lambda \cong 1050\,\text{nm}$. The 13 PCA bands were evaluated and examined, where it was noted that the first band with the largest variance conveyed little information about the structure in the coat of arms. PCA bands 2-4 were inserted in the blue, green, and red bands, respectively, of a color image and the hue angle was rotated to reveal structure.



Processed images of of coat of arms in "Les Esches d'Amour": (a) after equalizing the histogram, showing unicorn at top; (b) RGB pseudocolor of PCA bands 2-4 from a 12-band spectral image after rotating the hue angle; note the enhanced visibility of the unicorn at top, the shield with a second unicorn at the bottom, and a "lion rampant" to the left of the shield.

6.4 Time-Sequence Images: Video

The other common example a 3-D image plots time on the third image axis. Perhaps the most obvious use is in motion pictures, where the different frames of the movie are the time samples of the 3-D spatial image $f[x, y, t_n]$. The illusion of continuous motion is created because of the photochemical response of the rods and cones in the retina. The time duration of the process is the source of the phenomenon of "persistence of vision." The persistence is shorter if image is brighter (movie projectors have rotary shutters that show each frame two or even three times). Movies were originally taken at 16 frames per second, later increased to 24 fps in US. This is the reason why motion in old movies is too fast. The frame rate in Europe) is 25 fps, related to the AC frequency of rate is 50 Hz. For this reason, American movies shown in Europe finish in only 96% of the time in the US.

The second most familiar example of time-sampled imagery is video, where the 3-D scene $f[x, y, t]$ is sampled in both time and space to convert the scene to a 1-D function of time that I shall call $s[t]$. It took more than 50 years to develop video hardware to scan scenes. The first scanning systems were mechanical, using either a system that scanned light reflected from an illuminated scene, or an illumination system that scanned a beam of light over the scene and collected any reflected light. One of the primary developers of mechanically scanned video was the Scotsman John

Logie Baird. The hardware of electronic scanning was developed through the efforts of such "illuminaries" as the American Philo T. Farnsworth, who demonstrated a working video system in the 1920s.

Video systems commonly use an "interlaced scan" that alternates scans of the even and odd lines of the full frame, so that the eye is presented with half of the image information every 1/60 s. This is less objectionable to the eye than the original "progressive-scanning" systems that presented a full frame every 1/30 s.

cathode rays (stream of electrons from the "cathode" to the anode)

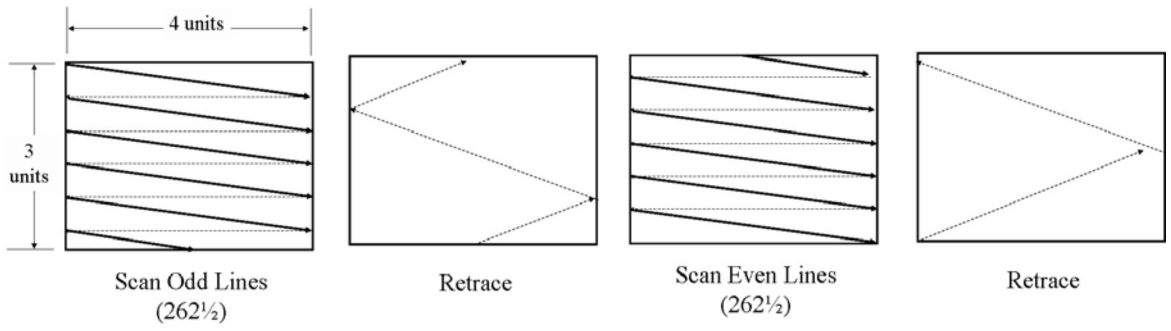
http://www.ee.washington.edu/conselec/CE/kuhn/ntsc/95x4.htm**??**

Response of retina through chemical process, implies a time to occur, implies persistence of vision,

movies originally 16 frames per second, increased to 24 fps in US (reason why old movies move too fast), 25 fps in Europe (AC rate is 50 Hz – American movies shown in Europe finish faster than here – one way to save time!)

persistence is shorter if image is brighter (movie projectors have rotary shutter to show each frame twice, or even three times)

reason for interlacing video



Interlaced format of NTSC video raster: one frame in 1/30 second is composed of two fields, each taking 1/60 second. The first "field" includes 262.5 odd lines and the second "field" has 262.5 even lines. Lines 248-263 in the first field and lines 511-525 in the second field are "blanked" – this is the "retrace" time for the CRT beam and is the time when additional information (e.g., closed captioning) is transmitted.

Note that lines number 248 to 263 and 511 to 525 are typically blanked to provide time for the beam to return to the upper left hand corner for the next scan; this is the "flyback" time that is used to transmit other signals (such as closed captioning or the second audio program SAP).

However lines number 248 to 263 and 511 to 525 are typically blanked to provide time for the beam to return to the upper left hand corner for the next scan.

6.5 Color-Space Transformations for Video Compression

References:

Pratt, Digital Image Processing, Second Edition, §2,3

Falk, Brill, Stork, Seeing the Light, §10

Glassner, Principles of Digital Image Synthesis, §1

As indicated by the previous discussion, visible-light color images are represented by triads of monochrome images. Decomposition of color images into RGB is very common, as it resembles the color detection mechanism of the eye that uses three types of color-sensitive detectors (the trichromatic cones) that have broad-band responses centered at approximately 420, 530, and 560nm. The color community specifies the cones as *S, M*, and *L*, for "short", "medium", and "long". The corresponding colors at the wavelengths of maximum sensitivity are roughly blue, green, and yellow, though type *L* has significant sensitiivity in the red. Within the neural network of the eye, the three cone signals are weighted and combined to generate the three channels that are transmitted to the brain. Roughly speaking, one channel corresponds to the lightness or luminance of the scene (i.e., the black-and-white video signal), and is a weighted sum (integral) of *S,M*, and L. This information is transmitted to the brain with full resolution. The other two transmitted signals are weighted differences (derivatives) of the *S, M*, and *L* and describe the chrominance of the scene; these are transmitted with reduced resolution, thus preserving information deemed more important during evolutionary development. Broadcast transmission of color video signals is roughly modeled on the weighted summing of cone signals for transmission to the brain.

In digital imaging, the three raw cone signals generated by the human visual system can be represented as three digital images: roughly the brightness in blue, green, and red light. The weighted summing of the cone signals for transmission to the brain may be modeled as a linear operation applied to the 3-element vector (*R,G,B*). The operation is implemented by an invertible 3×3 matrix. The requirement that the matrix be invertible ensures that the 3-element output vector is equivalent to the $[R, G, B]$ input vector; the input $[R, G, B]$ vector may be generated from the output vector. However, note that if the output values are quantized, as required before subsequent digital processing, then the cascade of forward and inverse transformations may not yield the identical triplet of RGB values.

Particular color transformations have been developed for use in many different applications, including various schemes for image transmission. Consider the transformation used in the color video standard in North America, that was developed by the National Television Standards Committee (NTSC), which converts RGB values to "luminance" Y (used by "black-and-white" receivers).and two "chrominance" channels I and Q via:

$$\begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

Note that the sum of the weights applied to the luminance channel Y is 1, while the sums of the weights of the two chrominance channels are both 0. In other words, the luminance channel is a weighted *sum* of RGB, while the chrominance channels are weighted *differences*. If the input R, G, and B are in the range [0,255], so will be the range of Y. The chrominance signals (I-Q in the former, U-V in the latter) are weighted differences; in both cases, the sum of the weights is zero, so that both chrominance channels of any gray input pixel (where $R = G = B$) is zero. The range of allowed chrominances is bipolar and fills the dynamic range: for 8-bit RGB inputs, $-103 \leq I \leq 152$, a total of 256 levels. The positive polarity of "I" is reddish (often described as "orange"), and the negative polarity is "green+blue" or cyan; hence the Q information is sometimes called the "orange-cyan" axis. The positive polarity of "Q" is "red+blue", or purple, and the negative polarity is green, so the I information is the "purple-green" axis.

In the context of linear systems, Y is the result of "spectral lowpass filtering," while I and Q are the outputs of what may be loosely described as "spectral highpass filters."

The transformation of a "mid-gray" pixel where the red, green, and blue images are identically α is:

$$\begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} \alpha \\ \alpha \\ \alpha \end{bmatrix} = \begin{bmatrix} \alpha \\ 0 \\ 0 \end{bmatrix}$$

so that the luminance is the gray value of the three colors and the two chrominance channels are both zero.

The transformation from YIQ back to RGB is the inverse of the forward matrix operator:

$$\begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix}^{-1} \cong \begin{bmatrix} 1.0 & 0.956 & 0.621 \\ 1.0 & -0.273 & -0.647 \\ 1.0 & -1.104 & 1.701 \end{bmatrix} \textit{(rounded)}$$

$$\begin{bmatrix} 1.0 & 0.956 & 0.621 \\ 1.0 & -0.273 & -0.647 \\ 1.0 & -1.104 & 1.701 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

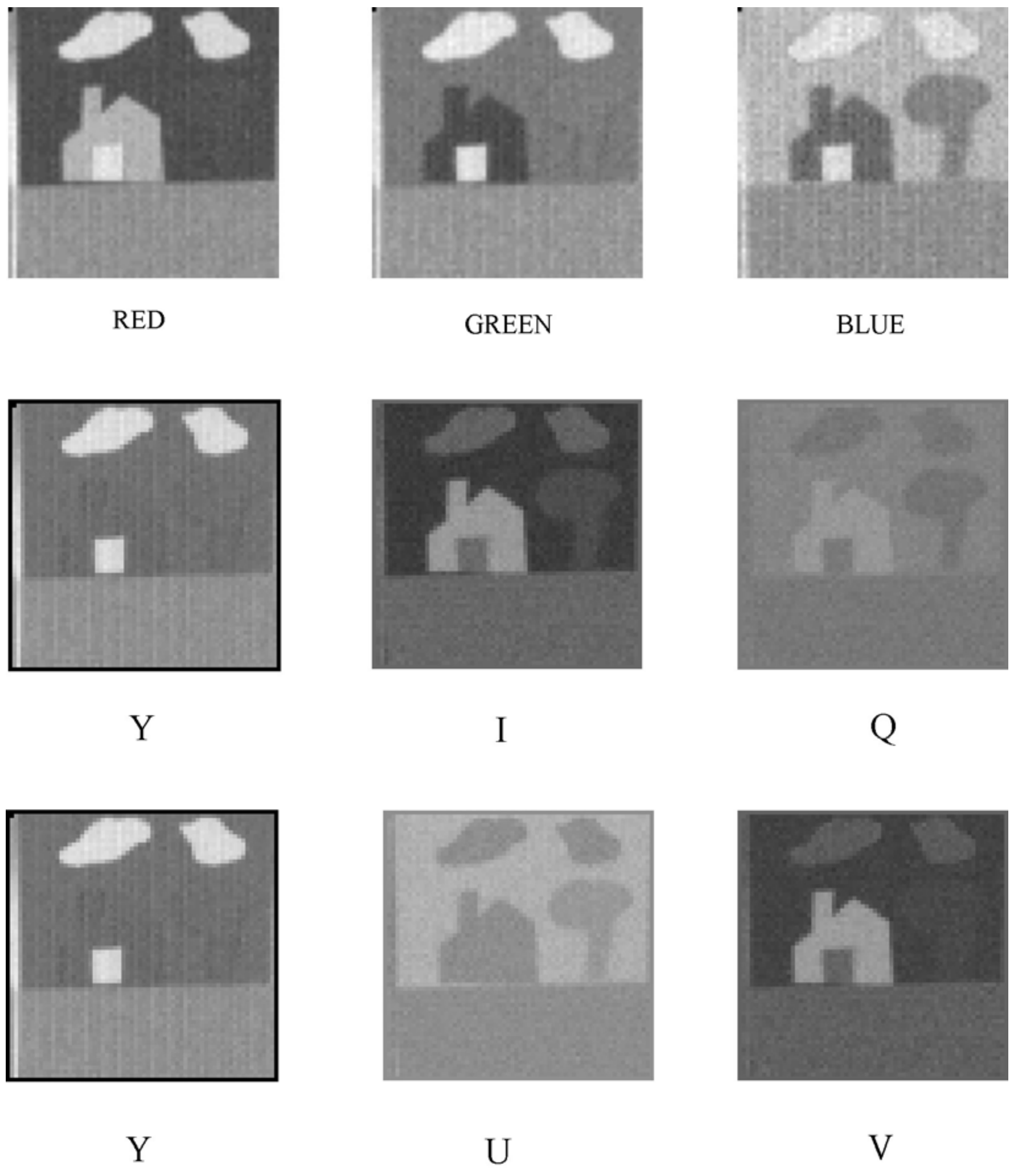
Note that the R, G, and B channels all include 100% of the luminance channel Y.

Other color transformations are used in other video systems. The two common systems used in Europe and Asia are *PAL* ("Phase Alternation by Line") and *SECAM* ("Systeme Electronique Couleur Avec Memoire"). Each broadcasts 625 lines at 25 frames per second and uses the *YUV* triad of luminance and chrominance, which is similar but not identical to *YIQ*. The transformation to *YUV* is:

$$\begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.148 & -0.289 & 0.437 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} Y \\ U \\ V \end{bmatrix}$$

The range of allowed chrominances is bipolar: for 8-bit RGB inputs $-144 \leq U \leq 111$ and $-98 \leq V \leq 157$ (each with 256 levels). The NTSC *YIQ* standard was selected over the *YUV* because tests indicated that more *I*-*Q* data could be discarded without affecting the subjective quality of the image presented to the viewer.

Obviously there exists an infinite number of invertible 3×3 matrices, and thus of invertible color transformations. The 3-D histograms of a particular input image before and after transformation generally will differ. Therefore segmentation of objects with particular similar colors likely will be easier or more successful in a particular color space.



Color image decomposition; all images displayed with full dynamic range (i.e., maximum possible gray value is mapped to "white" and minimum to "black"); first row shows the RGB images, second row the NSTC analog video transmission YIQ; the third row shows YUV.

6.6 Segmentation by Logical Operations on Multiple Images

It is quite feasible to segment a multispectral image by combining binary images by logical operations, e.g., the Boolean AND, OR, NAND (Not AND), and XOR (eXclusive OR) operators. These can be applied in the example just discussed to combine the segmented images obtained from 1-D histograms. Below are shown the images obtained from the green and blue image after thresholding pixels in the ranges $13 \leq f_g \leq 18$ and $16 \leq f_b \leq 21$ to white. By combining these images via a Boolean AND the result on the right is obtained. Note the noise (false identifications) are still quite prevalent in the segmented image.

6.7 Arithmetic Operations on Multiple Images

6.7.1 Multiple-Frame Averaging

Consider a series of video or movie images of an invariant (i.e. stationary) object $f[x,y]$ corrupted by additive noise which changes from pixel to pixel and frame to frame:

$$g[x,y,t_i] = f[x,y] + n[x,y,t_i]$$

where $n[x,y,t_i]$ is a random number selected from a Gaussian distribution with $\mu = 0$. The additive noise will tend to obscure the "true" image structure $f[x,y]$. One common problem in image processing is to enhance the visibility of the invariant objects in the image by attenuating the noise. If the gray values n are truly random, i.e., all values of n in the range $(-\infty, \infty)$ can exist with equal likelihood, then little can be done to improve the image. Fortunately, in realistic imaging problems the probability of each value of n is determined by some probability density function (histogram) $p[n]$ and we say that the noise is stochastic. The most common probability distributions in physical or imaging problems are the uniform, Poisson, and normal. Less common, but still physically realizable, distributions are the Boltzmann (negative exponential) and Lorentzian. A general discussion of stochastic functions is beyond the scope of the immediate discussion, though we will go into more detail later while reviewing statistical filters. Interested readers should consult Frieden's book *Probability, Statistical Optics, and Data Testing* (Springer-Verlag, 1991) for detailed discussions of physically important stochastic processes. At this point, we will state without proof that the important probability density function in physical problems is the normal distribution $N[\mu, \sigma^2]$, which may be completely characterized by two parameters: its mean value μ and the variance σ^2 (or its equivalent, the standard deviation σ). The histogram of noise gray values in the normal distribution is:

$$\begin{aligned} p[n] &= H[n[x,y,t_i]] \propto p[n] \\ &= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(n-\mu)^2}{2\sigma^2}\right] \equiv N\left(\mu, \sigma^2\right) \end{aligned}$$

This is the equation for the familiar bell curve with maximum value located at $n = \mu$ and with full width of 2σ measured at approximately 20% of the maximum amplitude. In the special case where the mean value $\mu = 0$, the normal distribution commonly is called a Gaussian distribution. The remainder of the discussion in this section will assume that the additive noise has been selected at random from a normal distribution.

It is probably clear intuitively that an image created by averaging a collection of noise images $n[x,y,t_i]$ over time will tend toward a uniform image whose gray level is the mean μ of the noise, i.e., the variations in gray level about the mean will "cancel out":

$$\frac{1}{N}\sum_{i=1}^{N-1} n[x,y.t_i] \cong \mu \cdot 1[x,y]$$

If the sequence of input images includes an invariant object on a background of additive normal noise, the visibility of the object will be enhanced in the average image:

$$\begin{aligned}\frac{1}{N}\sum_{i=1}^{N-1}(f[x,y]+n[x,y.t_i]) &= \frac{1}{N}\sum_{i=1}^{N-1} f[x,y]+\frac{1}{N}\sum_{i=1}^{N-1} n[x,y.t_i]\\ &\cong f[x,y]+\mu\cdot 1[x,y]\end{aligned}$$

This result is proven directly below, but may be seen more easily if the reader is familiar with the statistical concept that the probability density function of the sum of N random variables is the N-fold convolution of the individual probability density functions.

To quantify the visibility of the object in a noisy image, it is necessary to quantify the visibility of the noise, i.e., the variability of gray level due to the stochastic signal. The average gray value due to noise is:

$$\langle n[x,y]\rangle = \int_{-\infty}^{+\infty} n[x,y]\ \ p[n]\ \ dn = \mu$$

where $\langle\ \rangle$ denotes the averaged value of the quantity and $p[n]$ is the probability density function of the noise. This is obviously not appropriate to the problem because it does not describe the variability of gray value. A useful quantity is the variance of the noise which describes the average of the difference between the square of the gray value due to the noise and the mean:

$$\begin{aligned}\sigma^2[n] = \left\langle(n-\mu)^2\right\rangle &= \int_{-\infty}^{+\infty}(n-\mu)^2\ \ p[n]\ \ dn\\ &= \int_{-\infty}^{+\infty}\left(n^2-2n\mu+\mu^2\right)\ p[n]\ \ dn\\ &= \int_{-\infty}^{+\infty} n^2\ p[n]\ \ dn - 2\mu\int_{-\infty}^{+\infty} n\ p[n]\ \ dn + \mu^2\int_{-\infty}^{+\infty} p[n]\ \ dn\\ &= \int_{-\infty}^{+\infty} n^2\ p[n]\ \ dn - 2\mu\cdot\mu+\mu^2\cdot 1\\ &= \left\langle n^2\right\rangle - 2\mu^2+\mu^2\end{aligned}$$

$$\sigma^2[n] = \left\langle n^2\right\rangle - \mu^2$$

A measure of the relative visibility of the signal and the noise is the ratio of the signal to the noise variance, and is often called the signal-to-noise ratio (*S/N* or *SNR*). It may be expressed as amplitude or power:

$$\text{Power SNR} \equiv \frac{f^2[x,y]}{\sigma^2[n]}$$

$$\text{Amplitude SNR} \equiv \sqrt{\frac{f^2[x,y]}{\sigma^2[n]}} = \frac{f[x,y]}{\sigma[n]}$$

After averaging P frames of signal plus noise, the gray level at the pixel $[x,y]$ will approach:

$$\begin{aligned} g[x,y] &= \frac{1}{P}\sum_{i=1}^{P}\left(f[x,y] + n[x,y,t_i]\right) \\ &= \frac{1}{P}\sum_{i=1}^{P} f[x,y] + \frac{1}{P}\sum_{i=1}^{P} n[x,y,t_i] \\ &= \sum_{i=1}^{P}\frac{f[x,y]}{P} + \sum_{i=1}^{N}\frac{n[x,y,t_i]}{P} \\ &\cong f[x,y] + \mu \end{aligned}$$

The variance of the noise at pixel $[x,y]$ after averaging P frames is:

$$\begin{aligned} \sigma^2[n] &= \left\langle n^2[x,y]\right\rangle - \mu^2 = \sum_{i=1}^{P}\left(\frac{n[x,y,t_i]}{P}\right)^2 - \mu^2 \\ &= \frac{1}{P^2}\sum_{i=1}^{P} n[x,y,t_i]\sum_{j=1}^{P} n[x,y,t_j] - \mu^2 \\ &= \frac{1}{P^2}\sum_{i=1}^{P}\left(n[x,y,t_i]\right)^2 + \frac{2}{P^2}\sum_{i>j}^{P} n[x,y,t_i]\cdot n[x,y,t_j] - \mu^2 \end{aligned}$$

If the noise values are selected from the same distribution, all terms in the first sum on the right are identical:

$$\begin{aligned} \frac{1}{P^2}\sum_{i=1}^{P}\left(n[x,y,t_i]\right)^2 &= \frac{1}{P^2}\sum_{i>j}^{P}\left(\sigma_i^2 + \mu^2\right) \\ &= \frac{1}{P^2}\left(P\cdot\left(\sigma_i^2+\mu^2\right)\right) \\ &= \frac{\left(\sigma_i^2+\mu^2\right)}{P} \end{aligned}$$

Because the noise values are uncorrelated by assumption, the second term on the right is just the square of the mean:

$$\begin{aligned}\frac{1}{P^2}\sum_{i=1}^{P}(n[x,y,t_i])^2+\frac{2}{P^2}\sum_{i>j}^{P}n[x,y,t_i]\cdot n[x,y,t_j]-\mu^2 &= 2\cdot\left(\frac{\mu\cdot\mu}{2}\right)\\ &=\mu^2\end{aligned}$$

The variance of the average image is:

$$\sigma^2[n]=\frac{(\sigma_i^2+\mu^2)}{P}+\mu^2-\mu^2=\frac{\sigma_i^2+\mu^2}{P}$$

If the mean value μ of noise is 0 (Gaussian noise), then the variance of the sum is reduced by a factor of N and standard deviation is reduced by $\sqrt{P}$:

$$\begin{aligned}\sigma^2[n] &= \frac{\sigma_i^2}{P}\\ \sigma[n] &= \sqrt{\sigma^2[n]}=\frac{1}{\sqrt{P}}\sigma_i[n]\end{aligned}$$

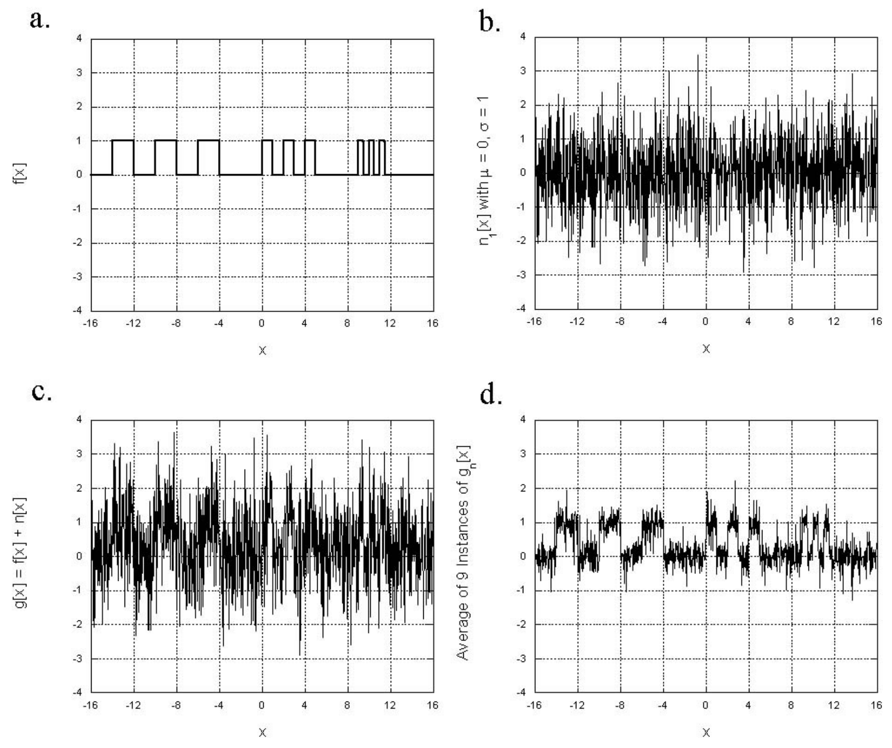
The amplitude SNR of the averaged image is:

$$SNR_{out}=\frac{f[x,y]}{\sqrt{\sigma^2[n]}}=\frac{f[x,y]}{\left(\frac{\sigma_i[n]}{\sqrt{P}}\right)}=\sqrt{P}\cdot\frac{f[x,y]}{\sigma_i}=\sqrt{P}\cdot SNR_{in}$$

$$\boxed{SNR_{out}=\sqrt{P}\cdot SNR_{in}}$$

Thus the effect of averaging multiple frames which include additive noise from a Gaussian distribution is to decrease the width of the histogram of the noise by a factor of $\left(\sqrt{P}\right)^{-1}$, which increases the signal-to-noise ratio of the image by $\sqrt{P}$. For example, a video image from a distant TV station is often contaminated by random noise ("snow"). If the image is stationary (i.e., does not vary with time), its signal-to-noise ratio can be increased by averaging; if 90 frames of video ($\cong$ 3 sec) are averaged, the SNR of the output image will increase by a factor of $\sqrt{90}\cong 9.5$.

If the noise is correlated to some degree from frame to frame (i.e., is not totally random), then averaging will not improve the SNR so rapidly. For example, consider imaging of a submerged object through a water surface. Wave motion will distort the images but there will be some correlation between frames. The improvement in SNR might be only $P^{0.25}\cong 3$ for $P=90$ frames.



Averaging of independent noise samples: (a) signal $f[x]$; (b) one realization of Gaussian noise $n_1[x]$ with $\mu = 0$ and $\sigma = 1$; (c) $f[x] + n_1[x]$; (d) $\frac{1}{9}\sum_{i=1}^{9}(f[x] + n_i[x])$, showing the improved signal-to-noise ratio of the averaged image.

6.7.2 Required Number of Bits for image Sums, Averages, and Differences

If two 8-bit images are added, then the gray value of the output image lies in the interval $[0, 510]$; these 511 possible gray values require 9 bits of data to represent fully. If constrained to 8 useful bits of data in the output image, half of the gray-scale variations data in the summation must be discarded. In short, it is necessary to requantize the summation image to fit is within the same dynamic range. If two 8-bit images are averaged, then the gray values of the resulting image are in the interval $[0, 255]$, but half-integer values are virtually guaranteed, thus ensuring that the average image also has 9 bits of data unless requantized.

The central limit theorem indicates that the histogram of an image resulting from a summation and/or average should approach a Gaussian form.

6.7.3 Image Subtraction

Change Detection

Subtraction of images of the same scene recorded at different times will highlight pixels whose gray value has changed in the interval:

$$g[x,y] = f[x,y,t_1] - f[x,y,t_0].$$

Invariant pixels will subtract to 0, pixels that have $\left\{\begin{array}{c}\text{brightened}\\ \text{dimmed}\end{array}\right\}$ will have $\left\{\begin{array}{c}\text{positive}\\ \text{negative}\end{array}\right\}$ gray level. This technique is applied to motion/change detection and may be interpreted as a time derivative.

$$\frac{\partial f[x,y,t]}{\partial t} = \lim_{\Delta t\to 0} \frac{f[x,y,t+\Delta t] - f[x,y,t]}{\Delta t}$$

For multitemporal digital images, the smallest nonvanishing time interval Δt is the interval between frames ($\Delta t = t_1 - t_0$). The time derivative is the difference image of adjacent frames:

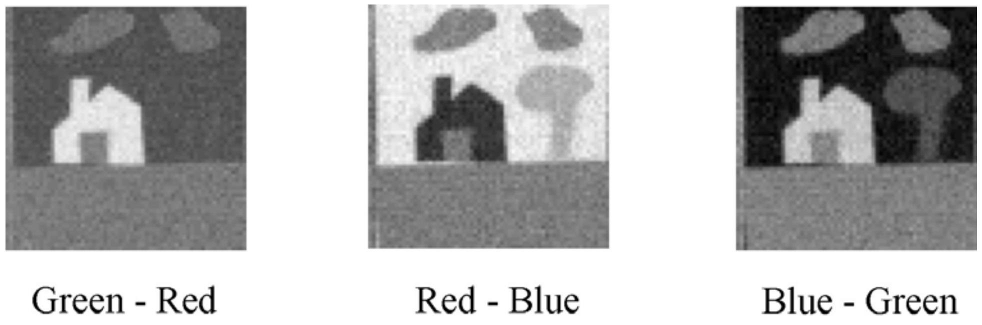
$$\frac{\partial f[x,y,t]}{\partial t} \to \partial_t\{f[x,y,t]\} \equiv \frac{f[x,y,t_0+(t_1-t_0)] - f[x,y,t_0]}{t_1-t_0} = \frac{f[x,y,t_1]-f[x,y,t_0]}{t_1-t_0}.$$

$$f[x,y,t_1] - f[x,y,t_0]$$

The difference image $g[x,y]$ is bipolar and must be scaled to fit the available discrete dynamic range [0-255] for display. Often $g_0 = 0$ is mapped to the mid-level gray (e.g. 127), the maximum negative level is mapped to 0, the brightest level to 255, and the intervening grays are linearly compressed to fit.

6.7.4 Difference Images as Features

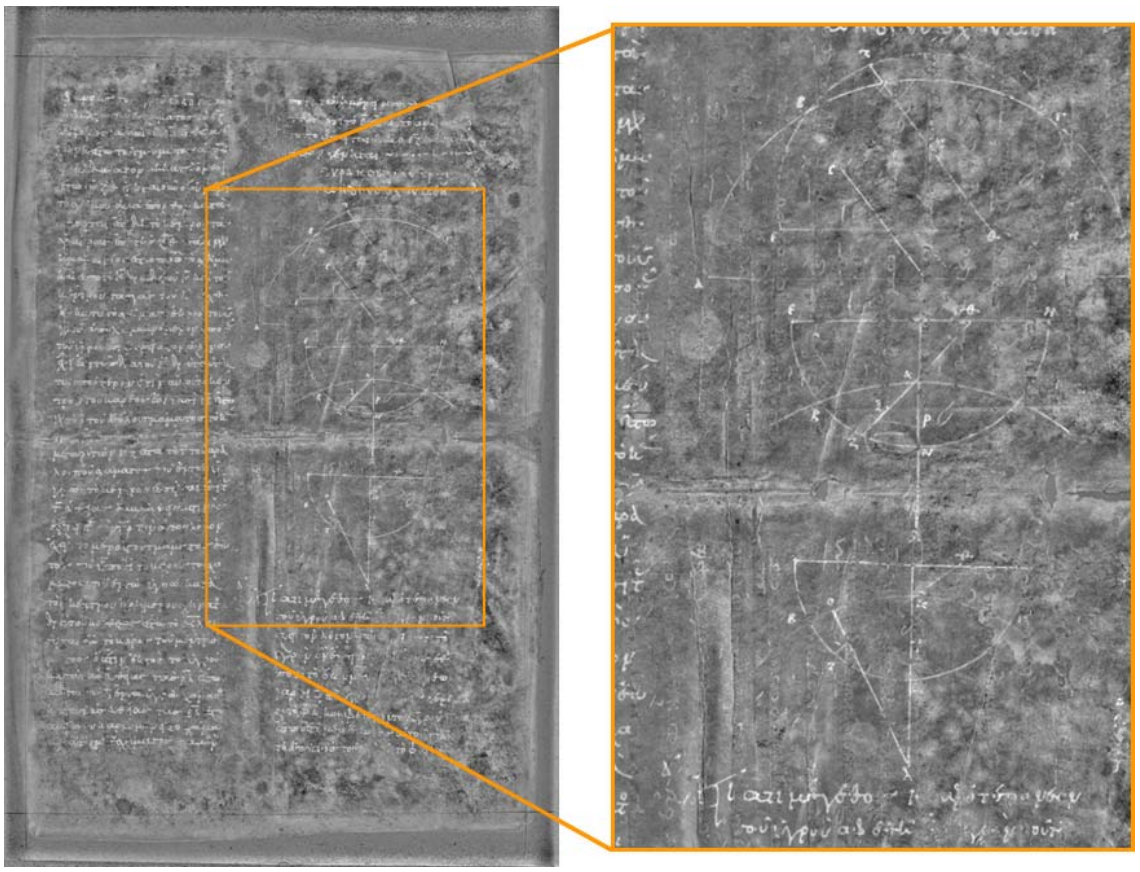
In feature extraction and recognition applications (e.g. remote sensing), linear combinations (i.e. sums and differences) of multispectral imagery may supply additional useful information for classification. For example, the difference image of spectral band 3 (red) and 4 (infrared) (out of a total of seven bands) imaged by LANDSAT helps classify urban vs. rural features in the image. In the simple house-tree image, the visibility of the house is enhanced in Red-Green and Blue-Red images.



Note that the difference images are noticeably noisier, especially Blue-Green; difference images enhance all variations in gray value, whether desired or not. The concerns about displaying bipolar gray values of time derivatives exist here as well.

Example: Spectral Differences in the Archimedes Palimpsest

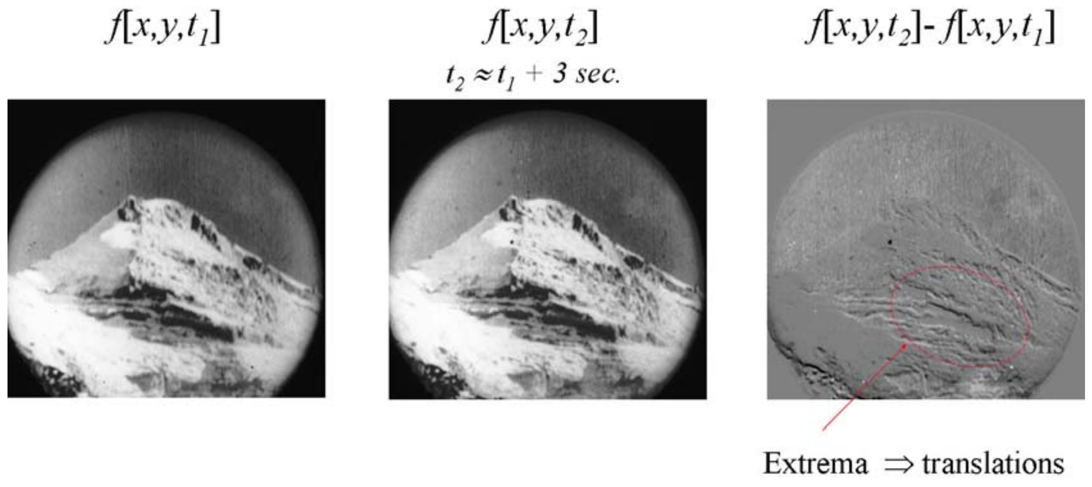
If the "foreground" object of interest has a different spectrum than the "background," then it may be possible to segment or enhance the foreground by displaying a difference of the two images. One useful example is the Archimedes palimpsest, where the erased original "undertext" is faint and reddish, while the later "overtext" is darker and more neutral. The reddish original text "disappears" into the parchment under red light, but shows rather well in the blue channel of a color image taken under fluorescent illumination. These two images are "balanced" to equalize the local mean value and variance over a window size that is larger than a character and then the UV blue channel is subtracted from the tungsten red channel; since the Archimedes text is brighter in the red and the later text is approximately the same value in both, the desired undertext has a positive value in the difference image, while the later prayerbook text and the parchment both appear with approximate value "0". The image is then scaled (the contrast is enhanced) to the full available dynamic range.



Example of spectral image subtraction in the Archimedes palimpsest; the blue channel of a color image under ultraviolet illumination is subtracted from the red channel of a color image under tungsten illumination, yielding a result where the pixels with equal brightness subtract to "0" and pixels that are brighter in the red tungsten image (such as the original text) generate positive values that appear as "white." (Creative Commons Attribution license, copyright retained by the Owner of the Archimedes Palimpsest)

Example: Time Differences in Imagery of Mallory Expedition

The British expedition to Mount Everest in 1924 ended with the deaths of George Mallory and Andrew Irvine. John Noel, the expedition cinematographer, set up his camera with a telephoto lens at base camp, approximately a 3-mile line of sight from the summit. Two frames of the movie film, spaced with $\Delta t \cong 3\,\mathrm{s}$ are shown. The images were subtracted in an attempt to segment pixels that had changed over that time span, with the goal of identifying any pixels that might correspond to the climbers. From the result, it is apparent that pixels had changed due to imperfections in the imaging system or due to refractive translations of the scene.



Difference of two images of the same scene taken with $\Delta t \cong 3\,\mathrm{s}$*; the images are from a movie made by John Noel of the summit cone of Mount Everest during the British expedition of 1924, during which George Mallory and Andrew Irvine died on the mountain (Mallory's body was discovered on 1 May 1999). One image has been translated by a small distance due to camera or optical effects, creating "edge" regions. Pixels with the same gray value are mapped to "midgray" and pixels in the second image that are brighter or darker map to whiter or darker, respectively.*

Number of Bits in Difference Image

If two 8-bit images with values in the intervals $[0, 255]$ are subtracted, then the gray values of the resulting image lie in the range $[-255, +255]$, for 511 possible values, requiring 9 bits of data to represent. Thus if we want to obtain 8 useful bits of data in the output image, we have to discard potentially half of the data in the difference image.

6.7.5 "Mask" or "Template" Multiplication: Image Mattes

Pixel-by-pixel multiplication of two images is useful to mask out sections of an image, perhaps to be replaced by objects from other images. This is occasionally useful in segmentation and pattern recognition, but is essential in image synthesis, such as for special effects in movies. Pixel-by-pixel image multiplication also is an essential part of local neighborhood and global image operations, which will be discussed in subsequent sections.

Lumière brothers

Green screen

6.7.6 Image Division

Images recorded by a system with spatially nonuniform response are functions of both the input distribution $f[x, y]$ and the spatial sensitivity curve $s[x, y]$, $0 \leq s[x, y] \leq 1$:

$$g[x,y] = f[x,y] \cdot s[x,y]$$

This is a deterministic multiplicative degradation of the image; the image may be restored by dividing out the noise. An estimate of the true image brightness can be computed at each pixel by division:

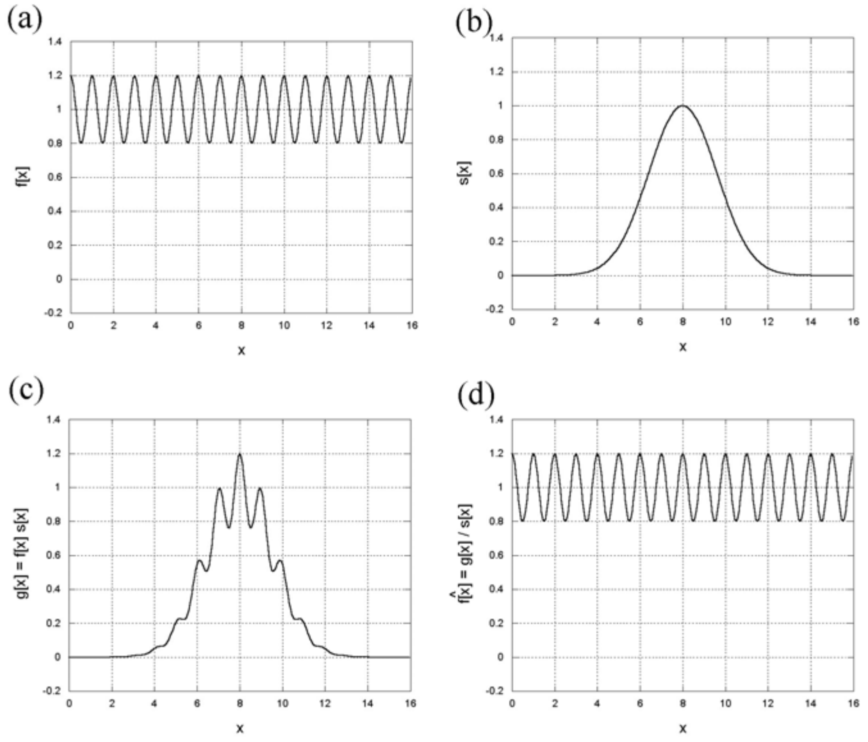
$$\hat{f}\,[x,y] = \frac{g[x,y]}{s[x,y]} \cong f\,[x,y]$$

(n.b., no image information is recorded at pixels where $s[x,y] = 0$ and thus the true value cannot be recovered at those pixels). This technique has been applied to remotely sensed imagery where information about image brightness is critical. Note that errors in the sensitivity function greatly distorts the recovered signal. Similarly, additive noise creates big problems in image division.

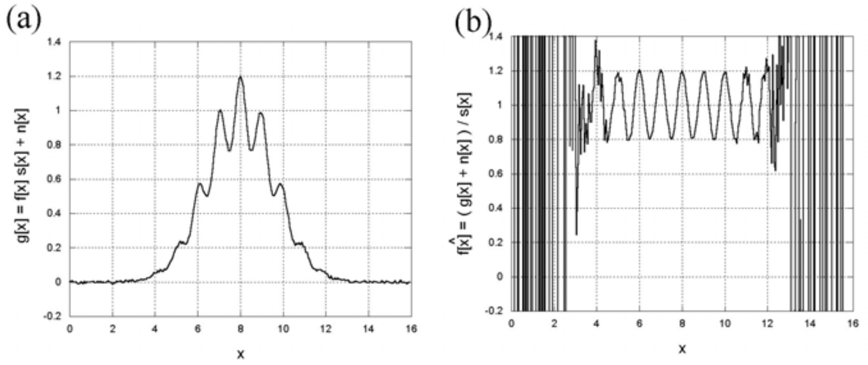
Image Division to Correct Spatial Sensitivity

Imaging systems often suffer from a consistent multiplicative error. One very common example is the variation in sensitivity of the pixels in a CCD sensor due to unavoidable variability in the properties of the substrate or in the manufacturing. It is possible to measure this error and correct for it via a subsequent division.

Consider a biased sine wave $f\,[x]$ recorded by an imaging system whose sensitivity falls off away from the origin. The image may be recovered completely if the sensitivity curve is nonzero everywhere and there is no noise in either the recorded signal or the estimate of the sensitivity.



1-D simulation of spatial compensation to correct for detector sensitivity: (a) original signal $f[x]$ is a biased sinusoid; (b) sensitivity function $s[x]$; (c) $g[x] = f[x] \cdot s[x]$; (d) correction $\hat{f}[x] = \dfrac{g[x]}{s[x]}$.



Spatial correction in the presence of noise: (a) $g[x] + n[x]$, where $n[x]$ is zero-mean Gaussian noise with $\sigma = 0.005$ (VERY small); (b) $\hat{f}[x] = \dfrac{g[x] + n[x]}{s[x]}$, showing the large errors where $|g[x]| \gtrapprox 0$ due to the incorrect division of two small values.

Noise in the estimate of the sensitivity results in distortion of the recovered signal; the effect is more severe where the SNR is low. The deviation of the added noise is 0.005.

Some other examples of image subtraction and division are considered after discussing convolution in the next chapter.

Chapter 7

Local Operations

In many common image processing operations, the output pixel is a weighted combination of the gray values of pixels in the neighborhood of the input pixel, hence the term local neighborhood operations. The size of the neighborhood and the pixel weights determine the action of the operator. This concept has already been introduced when we considered image prefiltering during the discussion of realistic image sampling. It will now be formalized and will serve as the basis of the discussion of image transformations.

$$g[x,y] = \mathcal{O}\{f[x \pm \Delta x, y \pm \Delta y]\}$$

7.1 Window Operators – Correlation

To introduce local neighborhood operators, consider the following process that acts on a 1-D input function $f[x]$ that is defined over a continuous domain:

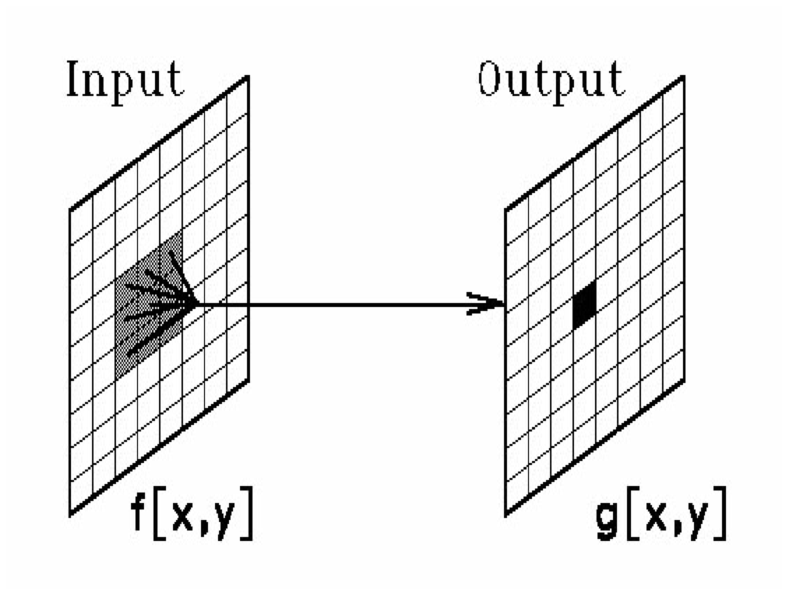
$$\mathcal{O}\{f[x]\} = \int_{-\infty}^{+\infty} f[\alpha] \cdot \gamma[\alpha - x] \; d\alpha$$

In words, this process computes the area of the product of two functions for each output location x: the input f and a second function γ that has been translated (shifted) by a distance x. The result of the operation is determined by the function $\gamma[x]$.

The process may be recast in a different form by defining a new variable of integration $u \equiv \alpha - x$:

$$\int_{-\infty}^{+\infty} f[\alpha] \cdot \gamma[\alpha - x] \; d\alpha \to \int_{u=-\infty}^{u=+\infty} f[x+u] \cdot \gamma[u] \; du$$

which differs from the first expression in that the second function $\gamma[u]$ remains fixed in position and the input function f is shifted by a distance $-x$. If the amplitude of the function γ is zero outside some interval in this second expression, then the integral need be computed only over the region where $\gamma[u] \neq 0$. The region where



the function $\gamma[x]$ is nonzero is called the support of γ, and functions that are nonzero over only a finite domain are said to exhibit *finite* or *compact* "support."

The 2-D versions of these expressions are:

$$\begin{aligned}\mathcal{O}\{f[x,y]\} &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f[\alpha,\beta]\cdot\gamma[\alpha-x,\beta-y]\ d\alpha\ d\beta \\ &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f[x+u,y+v]\cdot\gamma[u,v]\,du\ dv.\end{aligned}$$

The analogous process for sampled functions requires that the integral be converted to a discrete summation:

$$g[n,m] = \sum_{i=-\infty}^{+\infty}\sum_{j=-\infty}^{+\infty} f[i,j]\cdot\gamma[i-n,j-m] = \sum_{i=-\infty}^{+\infty}\sum_{j=-\infty}^{+\infty} f[i+n,j+m]\cdot\gamma[i,j].$$

In the common case where γ has compact support of size 3×3 samples, it may be considered to be a 3×3 matrix or window function:

$$\gamma[n,m] = \begin{array}{|c|c|c|}\hline \gamma_{-1,1} & \gamma_{0,1} & \gamma_{1,1} \\ \hline \gamma_{-1,0} & \gamma_{0,0} & \gamma_{1,0} \\ \hline \gamma_{-1,-1} & \gamma_{0,-1} & \gamma_{1,-1} \\ \hline \end{array},$$

The second expression reduces to:

$$g[n,m] = \sum_{i=-1}^{+1}\sum_{j=-1}^{+1} f[i+n,j+m]\cdot\gamma[i,j].$$

In words, this process scales the shifted function by the values of the matrix γ, and thus computes a weighted average of the input image $f[n,m]$. The operation derfined by this last equation is called the cross-correlation of the image with the window funtion $\gamma[n,m]$. The correlation operation often is denoted by $g[n,m] = f[n,m] \bigstar [n,m]$. The output image g at pixel indexed by [n,m] is computed by centering the window $\gamma[n,m]$ on that pixel of the input image $f[n,m]$, multiplying the window and input image pixel by pixel, and summing the products. This operation produces an output extremum at shifts [n,m] where the gray-level pattern of the input matches that of the window.

Examples of 3×3 windows:

$$\gamma_1[n,m] = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & +1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

$$\gamma_2[n,m] = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & +2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

$$\gamma_1[n,m] = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & +1 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

γ_1– the only pixel that influences the output $g[n,m]$ is the identical pixel in the input $f[n,m]$– this is the *identity* operator.

γ_2– the output pixel has twice the gray value of the input pixel – this is a uniform contrast stretching operator.

γ_3– the output pixel is identical to its right-hand neighbor in the input image – this operator moves the image one pixel to the left.

Once the general cross-correlation algorithm is programmed, many useful operations on the image $f[n,m]$ can be performed simply by specifying different values for the window coefficients.

7.2 Convolution

A mathematically equivalent but generally more convenient neighborhood operation is the CONVOLUTION, which has some very nice mathematical properties.

The definition of the convolution of a 1-D continuous input function $f[x]$ with a

1-D continuous function $h[x]$ is:

$$g[x] = f[x] * h[x] \equiv \int_{-\infty}^{\infty} d\alpha\ f[\alpha]\ \ h[x-\alpha]\,.$$

where α is a dummy variable of integration. As for the cross-correlation, the function $h[x]$ defines the action of the system on the input $f[x]$. By changing the integration variable to $u \equiv x - \alpha$, an equivalent expression for the convolution is found:

$$\begin{aligned} g[x] &= \int_{-\infty}^{\infty} f[\alpha]\ \ h[x-\alpha]\ \ d\alpha \\ &= \int_{u=+\infty}^{u=+\infty} f[x-u]\ \ h[u]\,(-du) \\ &= \int_{-\infty}^{\infty} f[x-u]\ \ h[u]\ \ du \\ &= \int_{-\infty}^{\infty} h[\alpha]\ \ f[x-\alpha]\ \ d\alpha \end{aligned}$$

where the dummy variable was renamed from u to α in the last step. Note that the roles of $f[x]$ and $h[x]$ have been exchanged between the first and last expressions, which means that the input function $f[x]$ and system function $h[x]$ can be interchanged.

The convolution of a continuous 2-D function $f[x,y]$ with a system function $h[x,y]$ is defined as:

$$\begin{aligned} g[x,y] = f[x,y] * h[x,y] &\equiv \iint_{-\infty}^{\infty} f[\alpha,\beta]\cdot h[x-\alpha,y-\beta]\,d\alpha\ d\beta \\ &= \iint_{-\infty}^{\infty} f[x-\alpha,y-\beta]\cdot h[\alpha,\beta]\ \ d\alpha\ d\beta \end{aligned}$$

Note the difference between the first forms for the convolution and the cross-correlation:

$$\begin{aligned} f[x,y] \bigstar [x,y] &= \iint_{-\infty}^{\infty} f[\alpha,\beta]\cdot\gamma[\alpha-x,\beta-y]\ \ d\alpha\ d\beta \\ f[x,y] * h[x,y] &\equiv \iint_{-\infty}^{\infty} f[\alpha,\beta]\cdot h[x-\alpha,y-\beta]\ \ d\alpha\ d\beta \end{aligned}$$

and between the second forms:

$$\begin{aligned} f[x,y] \bigstar [x,y] &\equiv \iint_{-\infty}^{\infty} f[x+u,y+v]\cdot\gamma[u,v]\,du\ dv \\ f[x,y] * h[x,y] &\equiv \iint_{-\infty}^{\infty} f[x-\alpha,y-\beta]\cdot h[\alpha,\beta]\,d\alpha\ d\beta \end{aligned}$$

The changes of the order of the variables in the first pair says that the function

γ is just shifted before multiplying by f in the cross-correlation, while the function h is flipped about its center (or equivalently rotated about the center by 180ž) before shifting. In the second pair, the difference in sign of the integration variables says that the input function f is shifted in different directions before multiplying by the system function γ for cross-correlation and h for convolution. In convolution, it is common to speak of filtering the input f with the kernel h. For discrete functions, the convolution integral becomes a summation:

$$g[n,m] = f[n,m] * h[n,m] \equiv \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f[i,j] \cdot h[n-i, m-j].$$

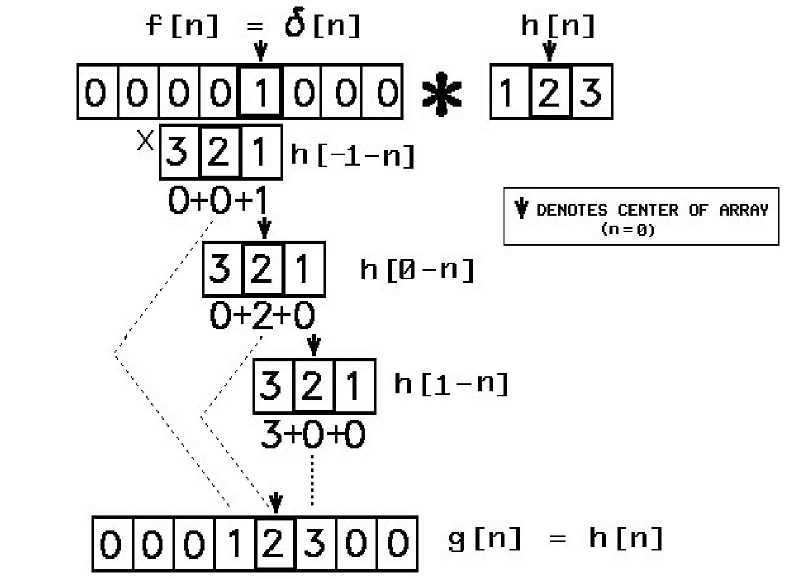
Again, note the difference in algebraic sign of the action of the kernel $h[n,m]$ in convolution and the window γ_{ij} in correlation:

$$f[n,m] \bigstar \gamma[n,m] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f[i,j] \cdot (\gamma[i-n, j-m])$$
$$f[n,m] * h[n,m] = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} f[i,j] \cdot (h[n-i, m-j]).$$

This form of the convolution results in the very useful property that convolution of an impulse function $\delta[n,m]$ with a system function $h[n,m]$ yields the system function:

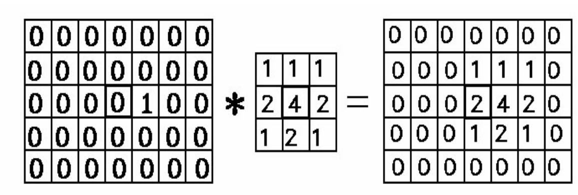
$$\delta[n,m] * h[n,m] = h[n,m]$$

where $\delta[n,m] \equiv 1$ for $n = m = 0$, and 0 otherwise. Consider a 1-D example of convolution:



Schematic of the sequence of calculations in 1-D discrete convolution. The 3-pixel *kernel* $h[n] = \begin{array}{|c|c|c|}\hline 1 & 2 & 3 \\ \hline \end{array}$ *is convolved with the input image that is 1 at one pixel and zero elsewhere. The output is a replica of* $h[n]$ *centered at the location of the impulse.*

The same type of result applies in two dimensions:


Schematic of 2-D discrete convolution with the 2-D kernel $h[n,m]$.

$$\delta[i-n, j-m] * h[n,m] = h[n,m]$$

Persons with signal/image processing background refer to $h[n,m]$ as the *impulse response* instead of the *kernel*, since it is the output of convolution with an impulse; optickers call it the point spread function (psf). The psf is often nonzero only in a finite neighborhood – e.g., 3×3 or 5×5.

Examples:

0	0	0
0	+1	0
0	0	0

$\Longrightarrow$ *identity*

0	0	0
0	0	+1
0	0	0

$\Longrightarrow$ *shifts image one pixel to right*

Discrete convolution is linear because it is defined by a weighted sum of pixel gray values, i.e.,

$$\begin{aligned} f[n,m] * (h_1[n,m] + h_2[n,m]) &\equiv \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} f[i,j] \cdot (h_1[n-i,m-j] + h_2[n-i,m-j]) \\ &= \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} (f[i,j] \cdot h_1[n-i,m-j] + f[i,j] \cdot h_2[n-i,m-j]) \\ &= \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} f[i,j] \cdot h_1[n-i,m-j] \\ &\qquad + \sum_{i=-\infty}^{+\infty} \sum_{j=-\infty}^{+\infty} f[i,j] \cdot h_2[n-i,m-j] \end{aligned}$$

$$\boxed{f[n,m] * (h_1[n,m] + h_2[n,m]) = f[n,m] * h_1[n,m] + f[n,m] * h_2[n,m]}$$

Sums or differences of kernels therefore create new kernels. For example, consider the sum of three 3×3 kernels:

$$h[n,m] = \frac{1}{3} \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & +1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} + \frac{1}{3} \begin{array}{|c|c|c|} \hline 0 & +1 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} + \frac{1}{3} \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & +1 & 0 \\ \hline \end{array}$$

The output image $g[n,m]$ is the average of three images: the input and copies shifted one pixel up and down. Therefore, each pixel in $g[n,m]$ is the average of three pixels in a vertical line; $g[n,m]$ is blurred vertically. Note that the kernels have been normalized so that the sum of the elements is unity. This ensures that the gray level of the filtered image will fall within the dynamic range of the input image, but they may not be integers. The output of a lowpass filter must typically be requantized.

7.2.1 Convolutions – Edges of the Image

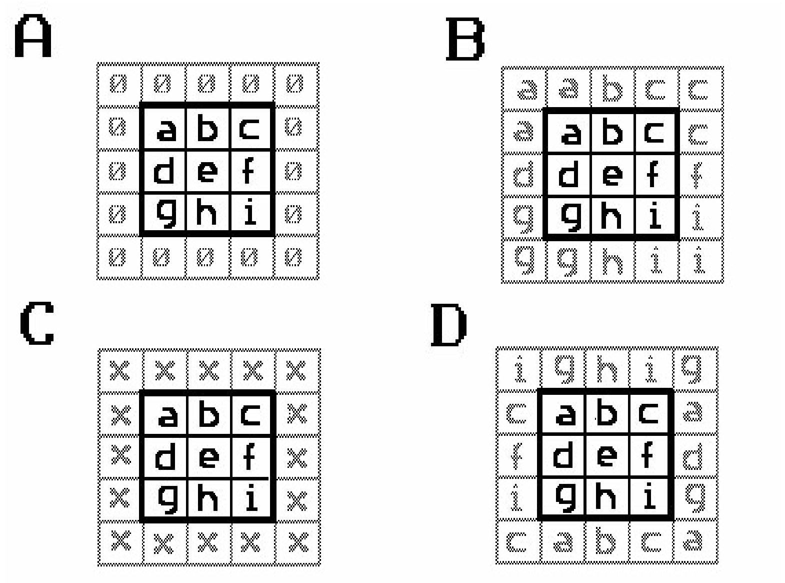
Because a convolution is the sum of weighted gray values in the neighborhood of the input pixel, there is a question of what to do near the edge of the image, i.e., when the neighborhood of pixels in the kernel extends "over the edge" of the imgae. The common solutions are:

1. consider any pixel in the neighborhood that would extend off the image to have gray value "0";
2. consider pixels off the edge to have the same gray value as the edge pixel;
3. consider that the convolution in any such case to be undefined; and
4. define any pixels over the edge of the image to have the same gray value as pixels on the opposite edge.

On the face of it, the fourth of these alternatives may seem to be ridiculous, but it is simply a statement that the image is assumed to be periodic, i.e., that:

$$f[n,m] = f[n+kN, m+\ell M]$$

where N and M are the numbers of pixels in a row or column, respectively, and k,ℓ are integers. In fact, this is the most common case, and will be treated in depth when global operators are discussed.



Possible strategies for dealing with the edge of the image in 2-D convolution: (a) the input image is padded with zeros; (b) the input image is padded with the same gray values "on the edge;" (c) values "off the edge" are ignored; (d) pixels off the edge are assigned the values on the opposite edge, this assumes that the input image is periodic.

The 3×3 image $f[n,m]$ is bold face, the assumed gray values of pixels off the edge of the image are in light face for the four cases. (If an "x" falls within the window, the output gray value is undefined)

7.2.2 Convolutions – Computational Intensity

Convolution with a serial processor can be very slow, especially for large kernels. For example, convolution of a 512^2-pixel image with an $M \times M$ kernel requires: $2 \cdot 512^2 \cdot M^2$ operations (multiplications and additions) for a total of $4.7 \cdot 10^6$ operations with a 3×3 kernel and $25.7 \cdot 10^6$ operations with a 7×7. Of course, it should be noted that the operations are performed on integer rather than floating-point data (at least to the point of image scaling). The increase in computations as M^2 ensures that convolution of large images with large kernels is not very practical by serial brute-force means. In the discussion of global operations to follow, we will introduce an alternative method for computing convolutions via the Fourier transform that requires many fewer operations for large images.

7.2.3 Smoothing Kernels – Lowpass Filtering

If all elements of a convolution kernel have the same algebraic sign, the operator $\mathcal{O}$ sums gray values of input pixels in the neighborhood; if the sum of the elements is zero, then the process computes a weighted average of the gray values. Averaging reduces the variability of the gray values of the input image; it smooths the function:

$$\boxed{\textit{Local averaging } \textbf{\textit{decreases}} \textit{ the "variability" (variance) of pixel gray values}}$$

$$\boxed{\textit{Local averaging "pushes" gray values towards the mean}}$$

For a uniform averaging kernel of a fixed size, rapidly varying functions (e.g., short-period, high-frequency sinusoids) will be averaged more than slowly varying terms. In other words, local averaging attenuates the high sinusoidal frequencies while passing the low frequencies relatively undisturbed – local averaging operators are lowpass filters. If the kernel size doubles, input sinusoids with twice the period (half the spatial frequency) will be equivalently affected. This action was discussed in the section on realistic sampling; a finite detector averages the signal over its width and reduces modulation of the output signal to a greater degree at higher frequencies.

$$\boxed{\textit{Local averaging operators are } \textbf{\textit{lowpass}} \textit{ filters}}$$

Obviously, averaging kernels reduce the visibility of additive noise by spreading the difference in gray value of noise pixel from the background over its neighbors. By analogy with temporal averaging, spatial averaging of noise increases SNR by the square-root of the number of pixels averaged if the noise is random and the averaging weights are identical.

Averaging can be directional, e.g.,:

$$h[n,m] = \frac{1}{3}\cdot \begin{array}{|c|c|c|}\hline 0 & +1 & 0 \\ \hline 0 & +1 & 0 \\ \hline 0 & +1 & 0 \\ \hline \end{array} \quad \textit{blurs vertically}$$

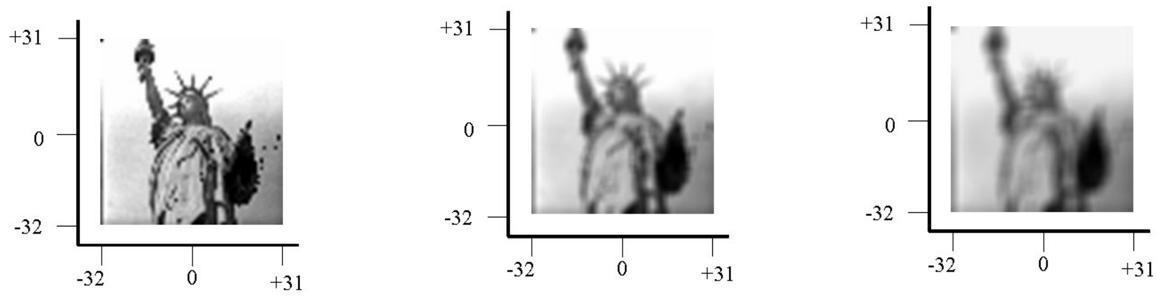
$$h[n,m] = \frac{1}{3}\cdot \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline +1 & +1 & +1 \\ \hline 0 & 0 & 0 \\ \hline \end{array} \quad \textit{blurs horizontally}$$

The elements of the kernel need not be identical, e.g.,

$$h[n,m] = \frac{1}{3}\begin{array}{|c|c|c|}\hline +0.25 & +0.25 & +0.25 \\ \hline +0.25 & +1 & +0.25 \\ \hline +0.25 & +0.25 & +0.25 \\ \hline \end{array}$$

averages over the entire window but the output is primarily influenced by the center pixel; the output blurred less than in the case when all elements are identical.

Lowpass-Filtered Images

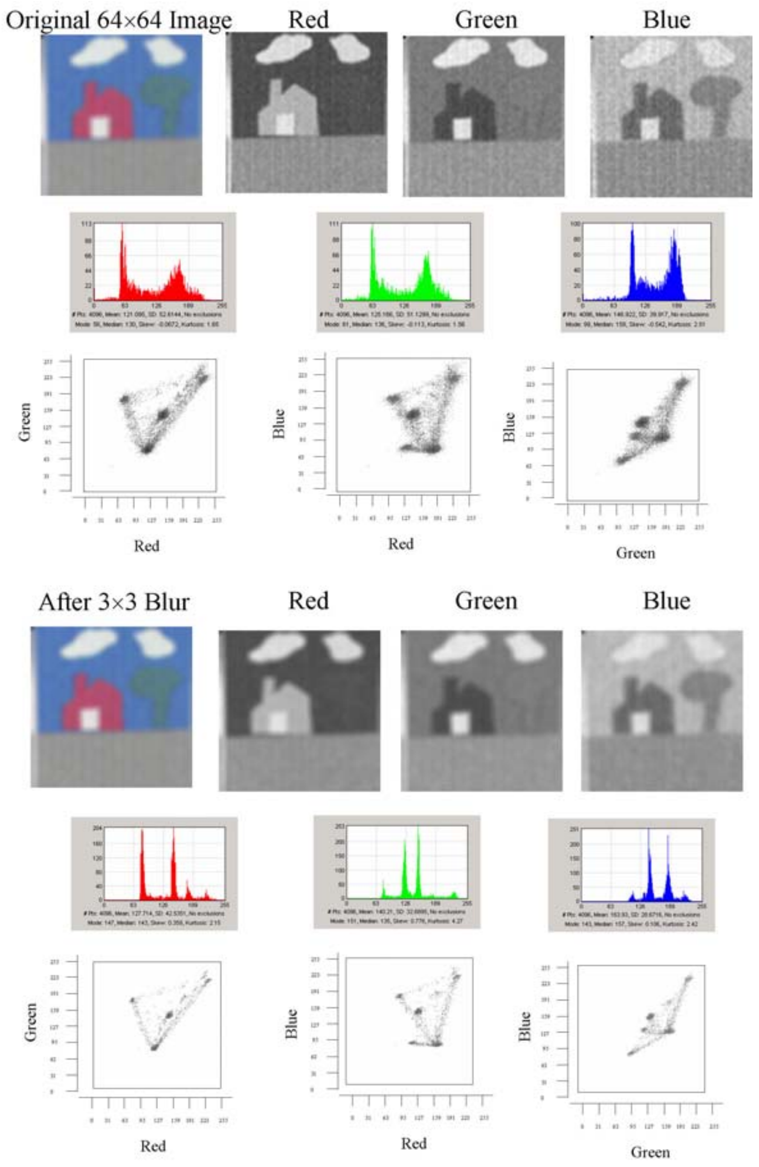


Examples of lowpass-filtered images: (a) original; (b) after 3×3 *local average; (c) after* 5×5 *local average.*

Effect of Lowpass Filtering on the Histogram

Because an averaging kernel reduces pixel-to-pixel variations in gray level, and hence the visibility of additive random noise in the image, we would expect that clusters of pixels in the histogram of an averaged image to be taller and thinner than in the original image. It should be easier to segment objects based on average gray level from the histogram of an averaged image. To illustrate, we reconsider the example of the house-tree image. The image in blue light and its histogram before and after averaging with a 3×3 kernel are shown below:

Note that there are four fairly distinct clusters in the histogram of the averaged image, corresponding to the house, grass/tree, sky, and clouds/door (from dark to bright). The small clusters at the ends are more difficult to distinguish on the original histogram.



Effect of blurring on the histogram: the 64×64 *color image, the histograms of the 3 bands, and the 3 2-D histograms are shown at top; the same images and histograms after blurring with a* 3×3 *kernel are at the bottom, showing the concentration of histogram clusters resulting from image blur.*

Note that the noise visible in uniform areas of the images (e.g., the sky in the blue image) has been noticeably reduced by the averaging, and thus the widths of the histogram clusters have decreased.

7.2.4 Differencing Kernels – Highpass Filters

From the previous discussion, it may be clear that the converse of the statement that local averaging reduces variability is also true:

Local Differencing ***increases*** *the variance of pixel gray values*

$$\boxed{\textit{Local Differencing "pushes" the gray values away from the mean}}$$
(and the new mean may be zero)

A kernel with both positive and negative terms computes differences of neighboring pixels. Adjacent pixels with identical gray levels will tend to cancel, while differences between adjacent pixels will tend to be emphasized. Since high-frequency sinusoids vary over shorter distances, differencing operators will enhance them and attenuate slowly varying (i.e., lower-frequency) terms.

$$\boxed{\textit{Differencing operators are } \textbf{\textit{highpass}} \textit{ filters}}$$

Subtraction of adjacent pixels results in output gray levels with values less than 0, just as in the case of image differencing for change detection considered in the last chapter. The output image must be biased up for display by adding some constant gray level to all image pixels, e.g., if the range of gray values of the difference image is, say, $[-g_0, g_0]$, then the negative gray values may be displayed in the interval $[0, 2 \cdot g_0]$ by adding the level g_0 to all pixels. Note that the maximum $g_{\max} = 2 \cdot g_0$ may be larger than the available range, so the output image generally must be requantized.

For an important example, consider the kernel:

$$h[n,m] = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline +1 & \text{-1} & 0 \\ \hline 0 & 0 & 0 \\ \hline\end{array} = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline +1 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline\end{array} + \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & \text{-1} & 0 \\ \hline 0 & 0 & 0 \\ \hline\end{array}$$
$$= \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline +1 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline\end{array} - \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & +1 & 0 \\ \hline 0 & 0 & 0 \\ \hline\end{array}$$

The output image is equivalent to the difference between an image shifted one pixel to the left and an unshifted image, which may be written in the form:

$$\begin{aligned}\frac{\partial}{\partial x} f[x,y] &= \lim_{\Delta x \to 0} \frac{f[x+\Delta x, y] - f[x,y]}{\Delta x} \\ &\implies \frac{\partial}{\partial x} f[x,y] \equiv f[(n+1)\cdot \Delta x, m \cdot \Delta y] - f[n \cdot \Delta x, m \cdot \Delta y] \\ &\implies \partial_x * f[n,m] = f[n+1,m] - f[n,m]\end{aligned}$$

because the minimum nonzero value of the translation $\Delta x = 1$ sample. The corresponding discrete partial derivative in the y-direction is:

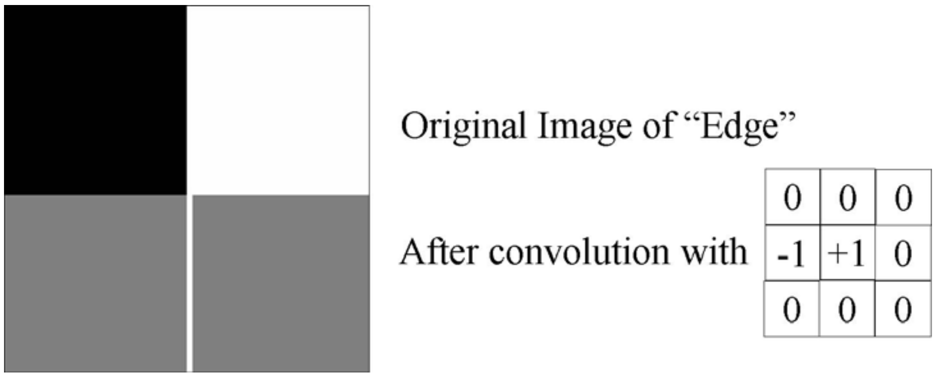
$$\partial_y * f[n,m] \equiv f[n,m+1] - f[n,m]$$

Discrete derivatives may be implemented via convolution with two specific discrete kernels that compute the differences of a translated replica of an image and the original:

$$\partial_x = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline +1 & -1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

$$\partial_y = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & -1 & 0 \\ \hline 0 & +1 & 0 \\ \hline \end{array}$$

This definition of the derivative effectively "locates" the edge of an object at the pixel immediately to the right or above the "crack" between pixels that is the actual edge.



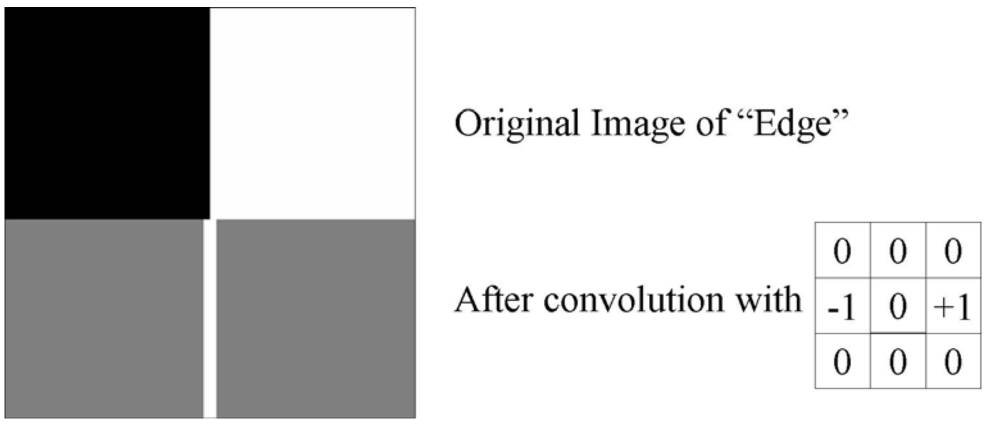
2-D Edge image on top (black on left, white on right) and the first derivative on bottom obtained by convolving with the 3×3 *first-derivative kernel in the x-direction, showing that the edge in the derivative image is located at the pixel to the right of the edge transition.*

A symmetric version of the derivative operators is sometimes used which takes the difference across two pixels:

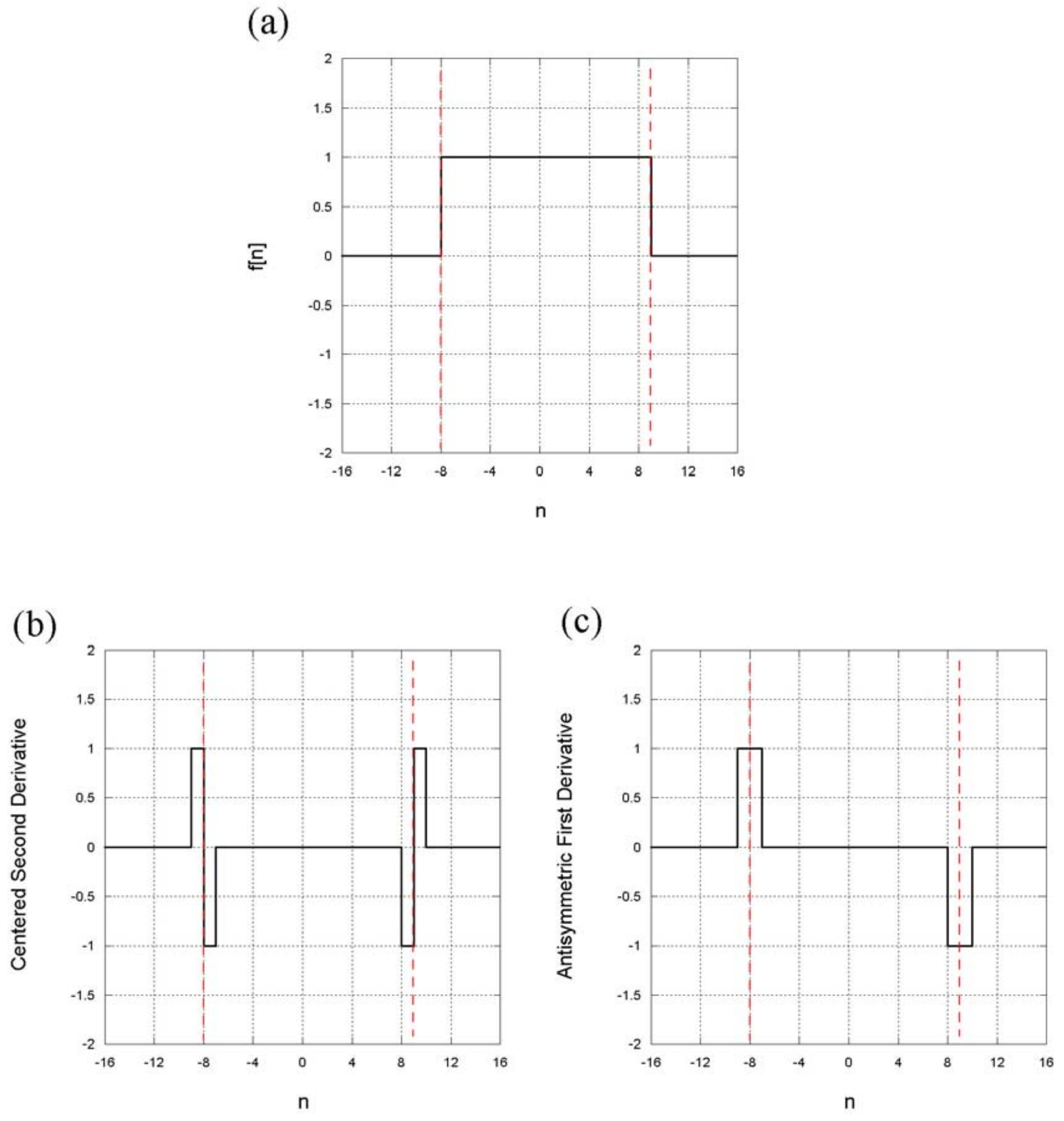
$$\partial_x = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline +1 & 0 & -1 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

$$\partial_y = \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & +1 & 0 \\ \hline \end{array}$$

These operators locate the edge of an object between two pixels symmetrically.



2-D Edge image and the first derivative obtained by convolving with the 3×3 *"symmetric" first-derivative kernel in the x-direction, showing that the edge is a two-pixel band symmetrically placed about the transition.*



First derivatives as edge detectors: (a) 1-D object with edges that both increase and decrease in brightness; (b) bipolar output from discrete first derivative $\begin{array}{|c|c|c|}\hline +1 & -1 & 0 \\ \hline \end{array}$ *where the dashed red line shows the location of the actual edge; (c) output of antisymmetric first derivative* $\begin{array}{|c|c|c|}\hline +1 & 0 & -1 \\ \hline \end{array}$ *, showing that the "edges" are two pixels wide. The ranges of the output images are* ± 1 *and so are usually rescaled to render the maximum value as white, "0" as midgray, and the minimum as black.*

Higher-Order Derivatives

The kernels for higher-order derivatives are easily computed since convolution is associative. The convolution kernel for the 1-D second derivative is obtained by autoconvolving the kernel for the 1-D first derivative:

$$\begin{aligned}
\frac{\partial^2}{\partial x^2} f[x,y] &= \frac{\partial}{\partial x}\left(\frac{\partial}{\partial x} f[x,y]\right) = \frac{\partial}{\partial x}\left(\lim_{\Delta x\to 0}\frac{f[x+\Delta x,y]-f[x,y]}{\Delta x}\right)\\
&= \lim_{\Delta x\to 0}\left(\lim_{\Delta x\to 0}\left(\frac{f[x+2\Delta x,y]-f[x+\Delta x,y]}{\Delta x}\right)-\lim_{\Delta x\to 0}\left(\frac{f[x+\Delta x,y]-f[x,y]}{\Delta x}\right)\right)\\
&= \lim_{\Delta x\to 0}\left(\frac{f[x+2\Delta x,y]-2f[x+\Delta x,y]+f\,[x,y]}{\Delta x}\right)\\
&\Longrightarrow\ \partial_x^2 * f[n,m] \equiv f[n+2,m]-2f[n+1,m]-f\,[n,m]
\end{aligned}$$

which may be evaluated by convolution with a five-element symmetric kernel:

$$\partial_x^2 = \begin{array}{|c|c|c|c|c|}\hline 0&0&0&0&0\\ \hline 0&0&0&0&0\\ \hline +1&-2&+1&0&0\\ \hline 0&0&0&0&0\\ \hline 0&0&0&0&0\\ \hline\end{array}$$

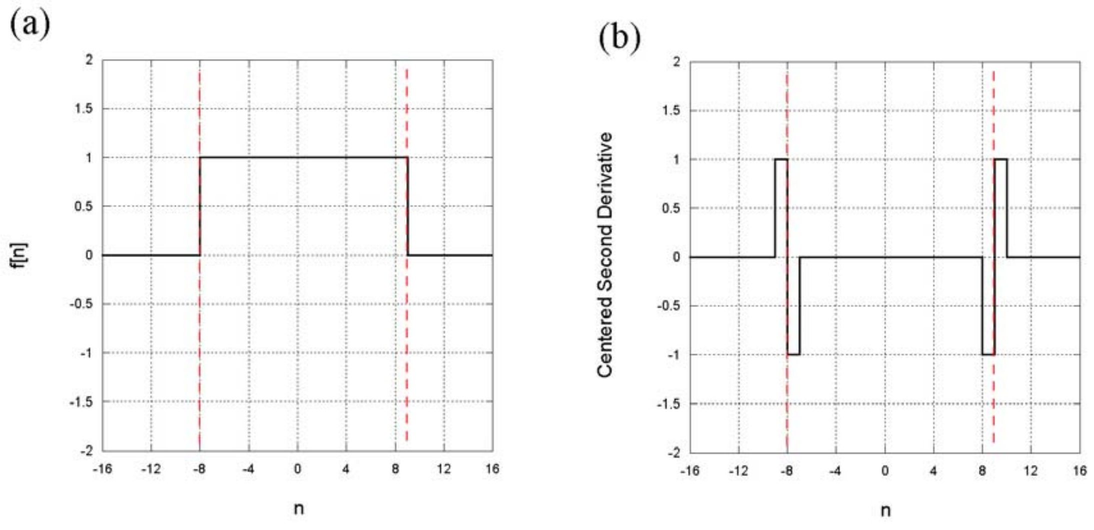
Usually, the kernel is translated to the right by one pixel to "center" the weights in a 3×3 kernel:

$$\hat{\partial}_x^2 = \begin{array}{|c|c|c|}\hline 0&0&0\\ \hline +1&-2&+1\\ \hline 0&0&0\\ \hline\end{array}$$

which generates the same image as cascaded first derivatives but for a shift to the right by a pixel. The corresponding 2-D second partial derivative kernels are:

$$\partial_y^2 = \begin{array}{|c|c|c|c|c|}\hline 0&0&0&0&0\\ \hline 0&0&0&0&0\\ \hline 0&0&+1&0&0\\ \hline 0&0&-2&0&0\\ \hline 0&0&+1&0&0\\ \hline\end{array}$$

$$\hat{\partial}_y^2 = \begin{array}{|c|c|c|}\hline 0 & +1 & 0 \\ \hline 0 & \text{-}2 & 0 \\ \hline 0 & +1 & 0 \\ \hline \end{array}$$



1-D rectangle object and the location of its edges obtained by convolving with the "centered" second derivative kernel $\begin{array}{|c|c|c|}\hline +1 & \text{-}2 & +1 \\ \hline \end{array}$ *in the x direction; the location of the true edges are shown as dashed lines. The edge in the output image is a pair of impulses with opposite signs. Again, the bipolar dynamic range must be rescaled.*

The derivation may be extended to derivatives of still higher order by convolving kernels to obtain the kernels for the 1-D third and fourth derivatives:

$$\partial_x^3 \equiv \partial_x * \partial_x * \partial_x$$

$$= \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline +1 & -1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline +1 & -1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline +1 & -1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

$$= \begin{array}{|c|c|c|c|c|c|c|}\hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline +1 & -3 & +3 & -1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \hline \end{array}$$

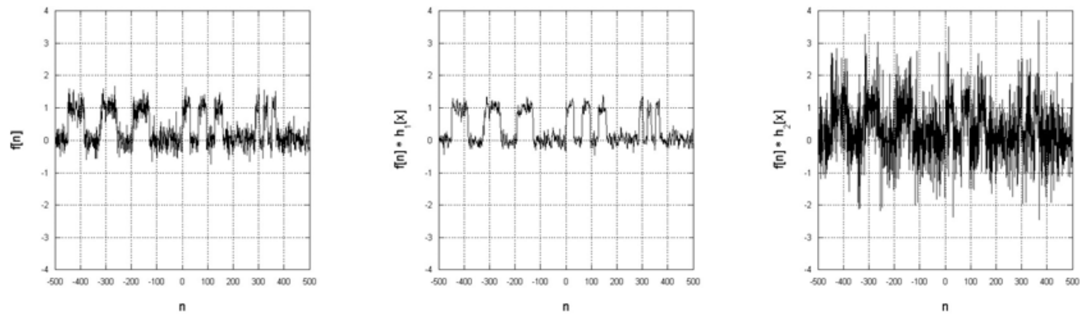
Of course, the higher-order derivatives are usually translated to reduce the size of the 1-D kernel and obtain:

$$\hat{\partial}_x^3 = \begin{array}{|c|c|c|c|c|}\hline +1 & -3 & +3 & -1 & 0 \\ \hline \end{array}$$

The gray-level extrema of the image produced by a differencing operator indicate pixels in regions of rapid variation, e.g., edges. The visibility of these pixels can be further enhanced by a subsequent thresholding operation.

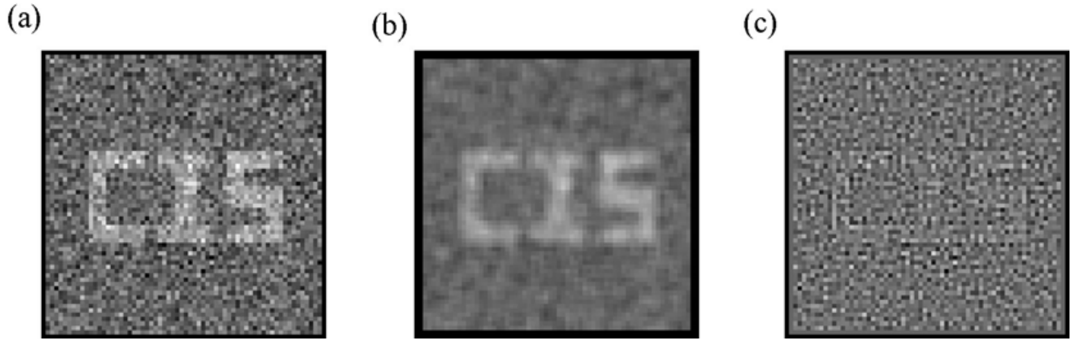
7.3 Effiects of Averaging and Differencing on Noisy Images

Because they compute weighted differences in pixel gray value, differencing operators will enhance the visibility of noise in an image. Consider the 1-D example where the input image $f[x]$is a 3-bar chart with added noise, so that the signal-to-noise ratio (SNR) of 4.The convolution of the input $f[x]$with an averaging kernel $h_1[x] = \frac{1}{3}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline \end{array}$ and with differencing kernel $h_2[x] = \begin{array}{|c|c|c|}\hline -1 & +3 & -1 \\ \hline \end{array}$ are shown below:



Effect of averaging and of sharpening applied to an image with noise: (a) $f[x]+n[x]$; (b) after averaging over 3-pixel neighborhood, showing reduction in noise; (c) after sharpening over 3-pixel neighborhood with $\begin{array}{|c|c|c|}\hline -1 & +3 & -1 \\ \hline \end{array}$*, showing increased noise variance.*

Note that the noise is diminished by the averaging kernel and enhanced by the differencing kernel. The 2-D case is shown below with a 3×3 uniform averager and a 3×3 sharpening kernel. The images were scaled to the same dynamic range. Again, note that the visibility of the letters is enhanced by averaging and diminished by differencing.



Effect of averaging and differencing operations on noisy image: (a) original image + noise; (b) after local averaging, showing the enhanced visibility of the letters; (c) after applying Laplacian sharpener (and extending the dynamic range), which enhances the noise relative to the signal.

7.3.1 Application of the Laplacian to Texture Segmentation

Since the Laplacian operator evaluates differences of gray values in the neighborhood, the values depend on both the pattern and magnitude of the local changes. For this reason, it may be used to segment regions that have the same average gray value but different "textures." Consider an image composed of two regions with the same mean value but different patterns of gray value

7.4

7.5 Applications of Differencing – Image Sharpening

7.5.1 Unsharp Masking

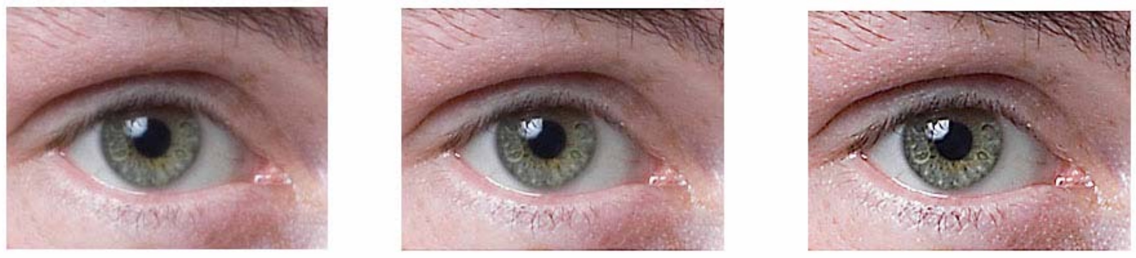
This digital technique that was adapted from a tool of photographic imaging that was developed in the 1930s to increase the apparent "sharpness" (the so-called "acutance") of images. It may be concisely described as the difference of an image and a blurry replica, where the difference operation was originally implemented as the sum of the blurry image and the original photographic negative. The steps in the photographic process are:

1. make a transparent positive contact print of the original scene by placing the copy emulsion in contact with the back of the original negative; the additional distance of the copy emulsion from the original slightly "blurs" the positive reproduction.

2. Place the blurred positive transparency just made in contact with the back of the original negative so that the two images are registered.

3. Make a positive print of the sandwich, which is the difference of the "sharply focused" positive print from the negative and a slightly blurry negative print from the blurred positive transparency.

In the resulting image, low-frequency features are partially canceled by the blurred transparency, thus relatively enhancing the high-frequency information in the original. In other words, the sinusoidal components of resulting image have larger amplitudes at the larger spatial frequencies, so the image appears "sharper." The "amount" of sharpening (i.e., the spatial frequency where enhancement is first noticed) is controlled by adjusting the amount of blurring; the more blurry the positive transparency, the sharper the final image.



Action of unsharp masking: (a) original appearance; (b) after unsharp masking (note highlight details at right side of eye and the lower lashes; (c) "oversharp" image. (credit, Wikipedia Commons)

We can think of unsharp masking in terms of the convolution operators that form the individual component images. The sharply focused image is produced by convolution with the identity kernel:

$$f[x,y] * \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & +1 & 0 \\ \hline 0 & 0 & 0 \\ \hline\end{array} = f[x,y]$$

We can generate the blurry image by convolution with the uniform averaging operator:

$$f[x,y] * \frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array} = f_1[x,y]$$

And the difference image may be written as

$$\begin{aligned} g[x,y] &= f[x,y] * \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & +1 & 0 \\ \hline 0 & 0 & 0 \\ \hline\end{array} - f[x,y] * \frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array} \\ &= f[x,y] * \left(\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & +1 & 0 \\ \hline 0 & 0 & 0 \\ \hline\end{array} - \frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}\right) \\ &= f[x,y] * \frac{1}{9}\begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & +8 & -1 \\ \hline -1 & -1 & -1 \\ \hline\end{array} \end{aligned}$$

This is single convolution operator $h[x,y]$ that will implement unsharp masking in in one step. Note that sum of the weights is zero, which means that the mean value of the output image will be zero.

7.5.2 Other Image Sharpeners

A different way to construct a sharpener is to add "positive" edge information to the original image. The edge information may be determined by applying some variety of edge detector. A very common edge detector is 2-D second derivative, which is called the "Laplacian". Recall the second derivative operator in the x- and y-directions:

$$\left(\frac{\partial}{\partial x}\right)^2 \Longrightarrow \partial_x^2 = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline +1 & -2 & +1 \\ \hline 0 & 0 & 0 \\ \hline\end{array}$$

$$\left(\frac{\partial}{\partial y}\right)^2 \Longrightarrow \partial_y^2 = \begin{array}{|c|c|c|}\hline 0 & +1 & 0 \\ \hline 0 & -2 & 0 \\ \hline 0 & +1 & 0 \\ \hline\end{array}$$

The sum of these two is:

$$\left(\frac{\partial}{\partial x}\right)^2 + \left(\frac{\partial}{\partial y}\right)^2 \equiv \nabla^2$$

$$\Longrightarrow \partial_x^2 + \partial_y^2 = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline +1 & -2 & +1 \\ \hline 0 & 0 & 0 \\ \hline\end{array} + \begin{array}{|c|c|c|}\hline 0 & +1 & 0 \\ \hline 0 & -2 & 0 \\ \hline 0 & +1 & 0 \\ \hline\end{array} = \begin{array}{|c|c|c|}\hline 0 & +1 & 0 \\ \hline +1 & -4 & +1 \\ \hline 0 & +1 & 0 \\ \hline\end{array}$$

This is the most common form of Laplacian operator. Note that the weights sum to zero, which means that the mean gray value of the image produced by this operator will be zero.

Now consider the result of subtracting the iamge produced by the Laplacian operator just specified from the original image. The convolution kernel may be specified as the difference of the identity and Laplacian operators:

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & +1 & 0 \\ \hline 0 & 0 & 0 \\ \hline\end{array} - \begin{array}{|c|c|c|}\hline 0 & +1 & 0 \\ \hline +1 & -4 & +1 \\ \hline 0 & +1 & 0 \\ \hline\end{array} = \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & +5 & -1 \\ \hline 0 & -1 & 0 \\ \hline\end{array}$$

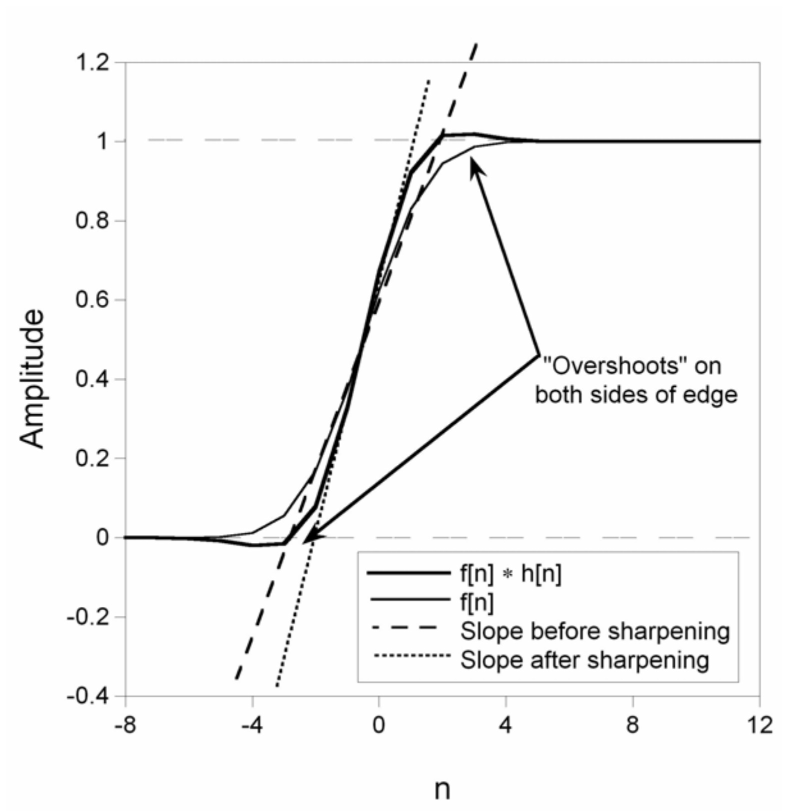
Note that the weights sum to one, which means that this operator will preserve the average gray value of the image.

We can also construct other variants of the Laplacian operator

The 1-D analogue of the Laplacian sharpener will subtract the 1-D second derivative from the identity:

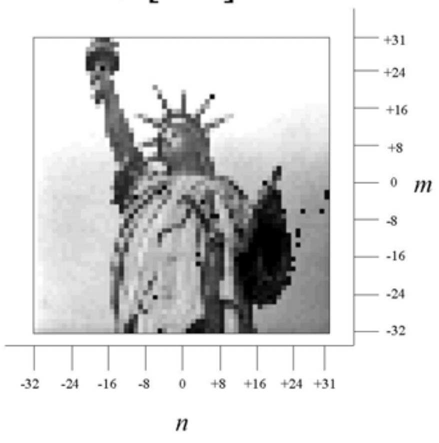
$$\begin{array}{|c|c|c|}\hline 0 & +1 & 0 \\ \hline\end{array} - \begin{array}{|c|c|c|}\hline +1 & -2 & +1 \\ \hline\end{array} = \begin{array}{|c|c|c|}\hline -1 & +3 & -1 \\ \hline\end{array}$$

It may be useful to apply this operator to a 1-D blurred edge to see the effect. In the vicinity of a blurred edge, the second derivative operator will enhance the edge information to sharpen the image

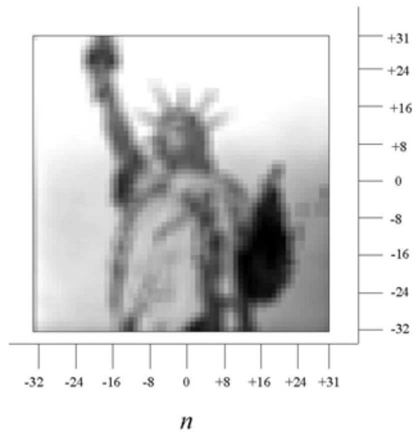


Action of 1-D 2nd-derivative sharpening operator on a "blurry" edge. The angle of the slope of the sharpened edge is "steeper", but the amplitude "overshoots" the correct value on both sides of the edge. In short, the output is not the ideal sharp edge.

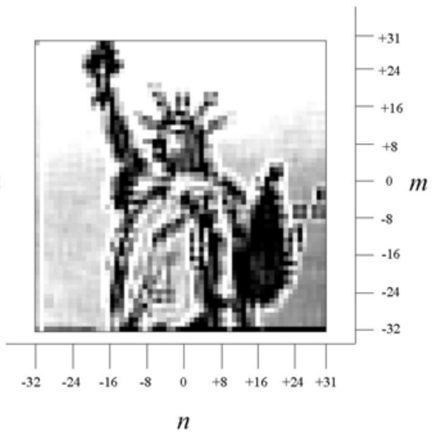
$f[n,m]$



$g[n,m] = f[n,m] * h[n,m]$



$$* \begin{bmatrix} -1 & -1 & -1 \\ -1 & +9 & -1 \\ -1 & -1 & -1 \end{bmatrix} =$$



7.5.3 Generalized Laplacian

We just showed how the isotropic Laplacian may be written as the difference of a 3×3 average and a scaled discrete delta function:

$$\begin{array}{|c|c|c|} \hline +1 & +1 & +1 \\ \hline +1 & -8 & +1 \\ \hline +1 & +1 & +1 \\ \hline \end{array} = \begin{array}{|c|c|c|} \hline +1 & +1 & +1 \\ \hline +1 & +1 & +1 \\ \hline +1 & +1 & +1 \\ \hline \end{array} - 9 \cdot \begin{array}{|c|c|c|} \hline 0 & 0 & 0 \\ \hline 0 & +1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

which suggests that the Laplacian operator may be generalized to include all operators that compute differences between a weighted replica of the original image and a copy blurred by some averaging kernel. One example of a generalized Laplacian may be constructed from the 2-D circularly symmetric continuous Gaussian impulse response:

$$h[x,y] = A\ \exp\left[-\pi\left(\frac{x^2+y^2}{\alpha^2}\right)\right]$$

where the decay parameter α determines the rate of attenuation of kernel values with radial distance from the origin. The amplitude parameter A often is selected to normalize the sum of the elements of the kernel to unity, thus ensuring that the process computes a weighted average that preserves the average gray value of the images. Amplitudes smaller than some threshold value are often set to zero. For example, a normalized discrete approximation of the Gaussian kernel with $\alpha = 2\cdot\Delta x$ might be approximated as:

$$h_1[n,m] = \frac{1}{2047}\begin{array}{|c|c|c|c|c|}\hline 1 & 11 & 23 & 11 & 1 \\ \hline 11 & 111 & 244 & 111 & 11 \\ \hline 23 & 244 & 535 & 244 & 23 \\ \hline 11 & 111 & 244 & 111 & 11 \\ \hline 1 & 11 & 23 & 11 & 1 \\ \hline\end{array} \cong \frac{1}{21}\begin{array}{|c|c|c|c|c|}\hline 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 1 & 2 & 1 & 0 \\ \hline 1 & 2 & 5 & 2 & 1 \\ \hline 0 & 1 & 2 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ \hline\end{array}$$

A corresponding generalized Laplacian operator is constructed by subtracting the 5×5 identity kernel from this discrete Gaussian:

$$h_2[n,m] = \frac{1}{21}\begin{array}{|c|c|c|c|c|}\hline 0 & 0 & 1 & 0 & 0 \\ \hline 0 & 1 & 2 & 1 & 0 \\ \hline 1 & 2 & 5 & 2 & 1 \\ \hline 0 & 1 & 2 & 1 & 0 \\ \hline 0 & 0 & 1 & 0 & 0 \\ \hline\end{array} - \frac{1}{21}\begin{array}{|c|c|c|c|c|}\hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 21 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 0 \\ \hline\end{array} = \frac{1}{21}\begin{array}{|c|c|c|c|c|}\hline 0 & 0 & +1 & 0 & 0 \\ \hline 0 & +1 & +2 & +1 & 0 \\ \hline +1 & +2 & -16 & +2 & +1 \\ \hline 0 & +1 & +2 & +1 & 0 \\ \hline 0 & 0 & +1 & 0 & 0 \\ \hline\end{array}$$

Note that the sum of the elements of this generalized Laplacian kernel is zero because of the normalization of the Gaussian kernel, which means the output will be a null image if the input is a uniform gray field.

The analogue of the Laplacian sharpening operator is constructed in the same manner as before, by subtracting the Laplacian (second derivative). In the original case:

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & +1 & 0 \\ \hline 0 & 0 & 0 \\ \hline\end{array} - \begin{array}{|c|c|c|}\hline 0 & +1 & 0 \\ \hline +1 & -4 & +1 \\ \hline 0 & +1 & 0 \\ \hline\end{array} = \begin{array}{|c|c|c|}\hline 0 & -1 & 0 \\ \hline -1 & +5 & -1 \\ \hline 0 & -1 & 0 \\ \hline\end{array}$$

With the generalized Laplacian:

$$\begin{array}{|c|c|c|c|c|}\hline 0&0&0&0&0\\\hline 0&0&0&0&0\\\hline 0&0&1&0&0\\\hline 0&0&0&0&0\\\hline 0&0&0&0&0\\\hline\end{array} - \frac{1}{21}\begin{array}{|c|c|c|c|c|}\hline 0&0&+1&0&0\\\hline 0&+1&+2&+1&0\\\hline +1&+2&-16&+2&+1\\\hline 0&+1&+2&+1&0\\\hline 0&0&+1&0&0\\\hline\end{array} = \frac{1}{21}\begin{array}{|c|c|c|c|c|}\hline 0&0&-1&0&0\\\hline 0&-1&-2&-1&0\\\hline -1&-2&+37&-2&-1\\\hline 0&-1&-2&-1&0\\\hline 0&0&-1&0&0\\\hline\end{array}$$

7.6 Directional Derivatives: Gradient

The *gradient* of a 2-D continuous function $f[x,y]$ constructs a 2-D vector at each coordinate whose components are the x- and y-derivatives:

$$\underline{\mathbf{g}}[x,y] = \nabla f[x,y] = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

This is a discrete version of a common operation in physics (particularly in electromagnetism). The image $f[n,m]$ is a *scalar* function which assigns a numerical gray value f to each coordinate $[n,m]$. The gray value f is analogous to terrain "elevation" in a map. In physics, the gradient of a scalar "field" $f[x,y]$ is the product of the vector operator ∇ (pronounced *del*) and the scalar "image" f, yielding $\nabla f[x,y]$. This process calculates a *vector* for each coordinate $[x,y]$ whose Cartesian components are $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$. Note that the 2-D vector ∇f may be represented in polar form as magnitude $|\nabla f|$ and direction $\mathbf{\Phi}\{\nabla f\}$:

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

$$\mathbf{\Phi}\{\nabla f[n,m]\} = \tan^{-1}\left[\frac{\left(\frac{\partial f}{\partial y}\right)}{\left(\frac{\partial f}{\partial x}\right)}\right]$$

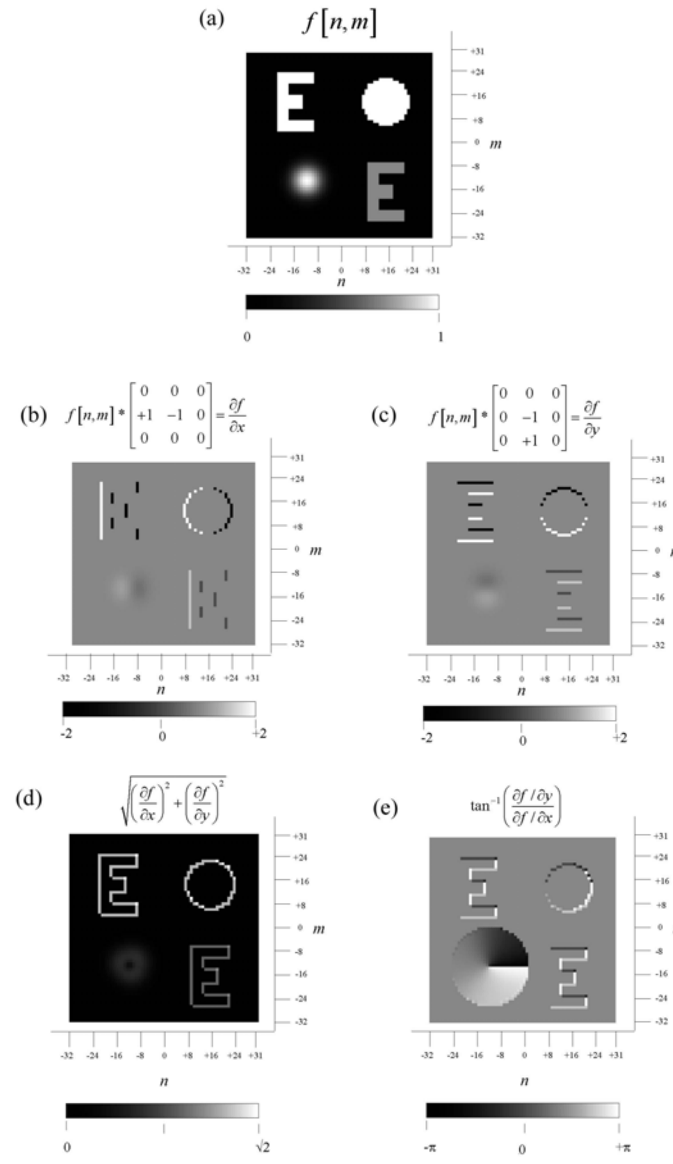
The vector points "uphill" in the direction of the maximum "slope" in gray level.

$$\underline{\mathbf{g}}[n,m] = \nabla f[n,m] = \begin{bmatrix}(\partial_x * f[n,m])\\(\partial_y * f[n,m])\end{bmatrix}$$

In image processing, the magnitude of the gradient often is approximated as the sum of the magnitudes of the components:

$$\left|\underline{\mathbf{g}}\,[n,m]\right| = |\nabla f\,[n,m]\,| = \sqrt{(\partial_x * f\,[n,m])^2 + (\partial_y * f\,[n,m])^2}$$
$$\cong |\partial_x * f\,[n,m]| + |\partial_y * f\,[n,m]|$$

The magnitude $|\nabla f|$ is the "slope" of the 3-D surface f at pixel $[n,m]$. The azimuth $\boldsymbol{\Phi}\,\{\nabla f\,[n,m]\}$ defines the compass direction where this slope points "uphill." The gradient is not a linear operator, and thus can neither be evaluated as a convolution nor described by a transfer function. The largest values of the magnitude of the gradient correspond to the pixels where the gray value "jumps" by the largest amount, and thus the thresholded magnitude of the gradient may be used to identify such pixels. In this way the gradient may be used as an "edge detection operator." An example of the gradient operator is shown.


(a) $f[n,m]$
(b) $f[n,m] * \begin{bmatrix} 0 & 0 & 0 \\ +1 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = \frac{\partial f}{\partial x}$
(c) $f[n,m] * \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & +1 & 0 \end{bmatrix} = \frac{\partial f}{\partial y}$
(d) $\sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$
(e) $\tan^{-1}\left(\frac{\partial f/\partial y}{\partial f/\partial x}\right)$
n
m
-32 -24 -16 -8 0 +8 +16 +24 +31
0
1
-2
+2
√2
-π
+π

Example of the discrete gradient operator $\nabla f\,[n,m]$*. The original object is the* *nonnegative function $f[n,m]$ shown in (a), which has amplitude in the interval $0 \leq f \leq +1$. The gradient at each pixel is the 2-D vector with components bipolar $\left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$. The two component images are shown in (b) and (c). These also may be displayed as the magnitude $\sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$ in (c) and the angle $\phi = \tan^{-1}\left[\frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}}\right]$ in (d). The extrema of the magnitude are located at corners and edges in $f[n,m]$.*

7.6.1 Roberts' Gradient

Often the gradient magnitude is approximated by replacing the Pythagorean sum of the derivatives by the sum of their magnitudes:

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

The magnitude of the gradient is always positive, which removes any difficulty from displaying bipolar data.

The gradient computed from the absolute values of the derivatives will preferentially emphasize outputs where both derivatives are "large," which will happen for diagonal edges. In short, the magnitude gradient will produce larger outputs for diagonal edges than for horizontal or vertical edges. A change of ± 1 gray value in the horizontal or vertical direction will product a gradient magnitude $|\nabla f| = 1$ gray value, while changes of ± 1 gray values along a 45° diagonal will generate a gradient magnitude of:

$$|\nabla f| = \sqrt{(\pm 1)^2 + (\pm 1)^2} = 1.4 \text{ gray values } > \sqrt{(\pm 1)^2 + (0)^2} \cong |\pm 1| + |\pm 1| = 2 \text{ gray values}$$

If the image structure is primarily horizontal or vertical, it may be desirable to replace the kernels for the x- and y-derivatives in the gradient operator by kernels for derivatives across the diagonals (by rotating by $\pm\frac{\pi}{4}$ radians):

$$\partial_{\pi/4} = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & \text{-1} & 0 \\ \hline +1 & 0 & 0 \\ \hline\end{array}, \quad \partial_{-\pi/4} = \begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & \text{-1} & 0 \\ \hline 0 & 0 & +1 \\ \hline\end{array}$$

The component operators for the Roberts' gradient are often considered to be:

$$\partial_1 = \begin{array}{|c|c|c|}\hline 0 & 0 & \text{-1} \\ \hline 0 & +1 & 0 \\ \hline 0 & 0 & 0 \\ \hline\end{array}, \quad \partial_2 = \begin{array}{|c|c|c|}\hline 0 & \text{-1} & 0 \\ \hline 0 & 0 & +1 \\ \hline 0 & 0 & 0 \\ \hline\end{array}$$

The gradient magnitude is computed as before and the result is the *Roberts' gradient,* which preferentially emphasizes horizontal or vertical features. A gradient magnitude which responds without preferential emphasis may be generated by summing the outputs of all four derivative kernels. For this application, the "untranslated" rotated operators are prefered to ensure that edges computed from all four kernels will overlap:

$$\begin{aligned}|\nabla f[x,y]| &= \sqrt{(\partial_x * f)^2 + (\partial_y * f)^2 + \left(\partial_{\pi/4} * f\right)^2 + \left(\partial_{-\pi/4} * f\right)^2} \\ &\cong |\partial_x * f| + |\partial_y * f| + \left|\partial_{\pi/4} * f\right| + \left|\partial_{-\pi/4} * f\right|\end{aligned}$$

Because the gradient operators compute gray-level differences, they will generate extreme values (positive or negative) where there are "large" changes in gray level, e.g., at edges. However, differencing operators also generate nonzero outputs due to "noise" in the images due to random signasl, quantization error, etc. If the variations due to noise are of similar size as the changes in gray level at an edge, identification of edge pixels will suffer.

7.6.2 "Laplacian of Gaussian"

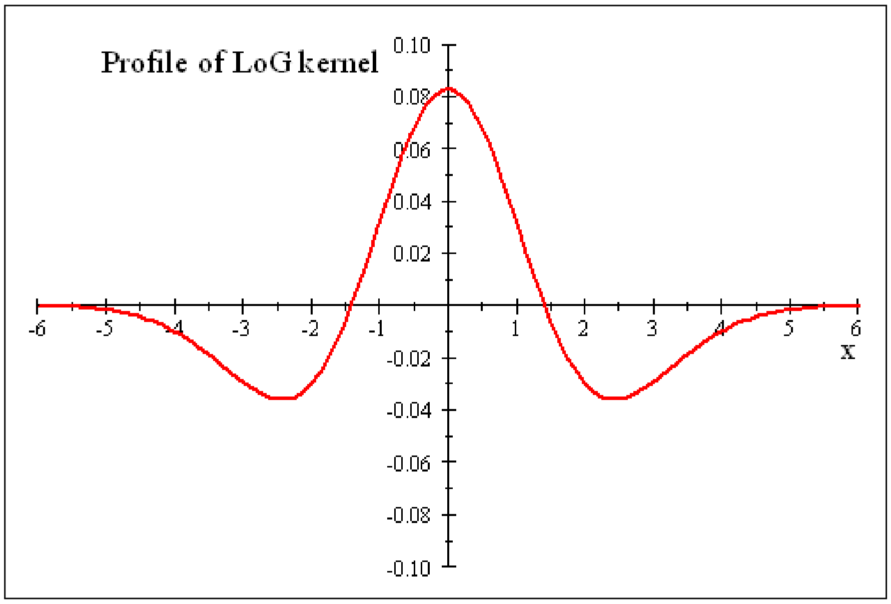
The so-called "Laplacian of Gaussian" (LoG) operator was introduced by Marr and Hildreth ("Theory of Edge Detection" **Proc. Royal Soc. London B207**, pp. 187-217, 1980) which blurs the images with a Gaussian averager and then performs a Laplacian edge detector. These are sometimes called "Marr operators." The output may be written as:

$$\begin{aligned}g[x,y] &= \nabla^2\{f[x,y] * h[x,y]\} \\ &= \left(\frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}\right)\left(f[x,y] * \frac{1}{2\pi\sigma^2}\exp\left[-\frac{x^2+y^2}{2\sigma^2}\right]\right)\end{aligned}$$

where $h[x,y]$ is the impulse response of the Gaussian function with standard deviation σ; the value of σ may be selected as a free parameter; the larger the value of σ, the wider the averaging of the Gaussian.

Since both operations (Laplacian and Gaussian blur) are implemented by convolution, we can get the same result by applying the Laplacian to the Gaussian kernel FIRST, thus producing a single kernel for the entire operation. Consider a 1-D example with continuous (nonsampled) coordinates, which we can easily extend to 2-D:

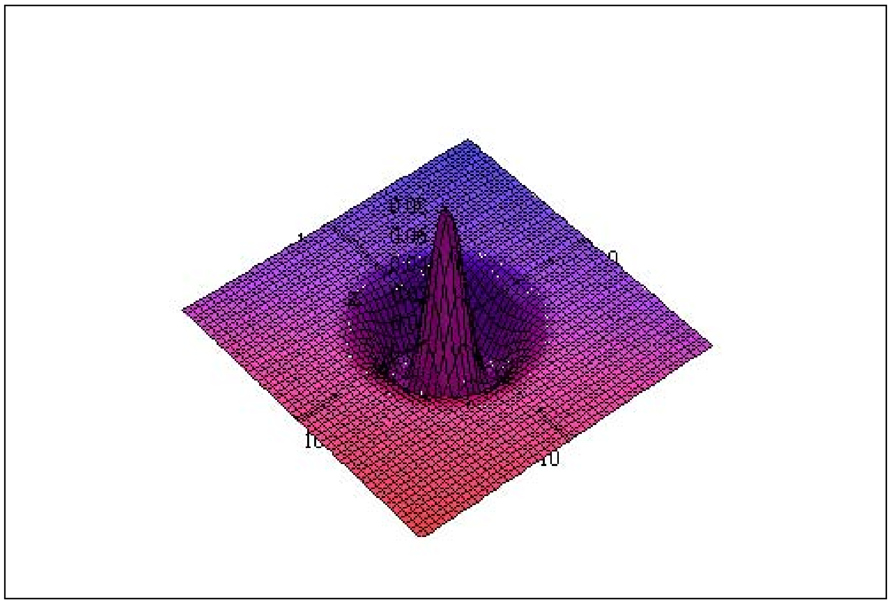
$$\begin{aligned}\frac{\partial^2}{\partial x^2}\left(\frac{1}{2\pi\sigma^2}\exp\left[-\frac{x^2}{2\sigma^2}\right]\right) &= \frac{1}{2\pi\sigma^2}\cdot\frac{\partial^2}{\partial x^2}\left(\exp\left[-\frac{x^2}{2\sigma^2}\right]\right)\\ &= \frac{1}{2\pi\sigma^2}\cdot\frac{\partial}{\partial x}\left(\frac{\partial}{\partial x}\exp\left[-\frac{x^2}{2\sigma^2}\right]\right)\\ &= \frac{1}{2\pi\sigma^2}\cdot\frac{\partial}{\partial x}\left(-\frac{x}{\sigma^2}\exp\left[-\frac{x^2}{2\sigma^2}\right]\right)\\ &= \frac{1}{2\pi\sigma^2}\cdot\left(-\frac{1}{\sigma^2}\exp\left[-\frac{x^2}{2\sigma^2}\right]-\frac{x}{\sigma^2}\left(-\frac{x}{\sigma^2}\exp\left[-\frac{x^2}{2\sigma^2}\right]\right)\right)\\ &= -\frac{1}{2\pi\sigma^4}\left(\frac{x^2}{\sigma^2}-1\right)\exp\left[-\frac{x^2}{2\sigma^2}\right]\end{aligned}$$



Profile of the LoG operator along the x-axis for $\sigma = 1.4$

The corresponding 2-D kernel is

$$\begin{aligned}\left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)\left(\frac{1}{2\pi\sigma^2}\exp\left[-\frac{x^2+y^2}{2\sigma^2}\right]\right) &= \left(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2}\right)\left(\frac{1}{2\pi\sigma^2}\exp\left[-\frac{x^2+y^2}{2\sigma^2}\right]\right)\\ &= \frac{1}{\pi\sigma^4}\left(1-\frac{x^2+y^2}{2\sigma^2}\right)\exp\left[-\frac{x^2+y^2}{2\sigma^2}\right]\end{aligned}$$



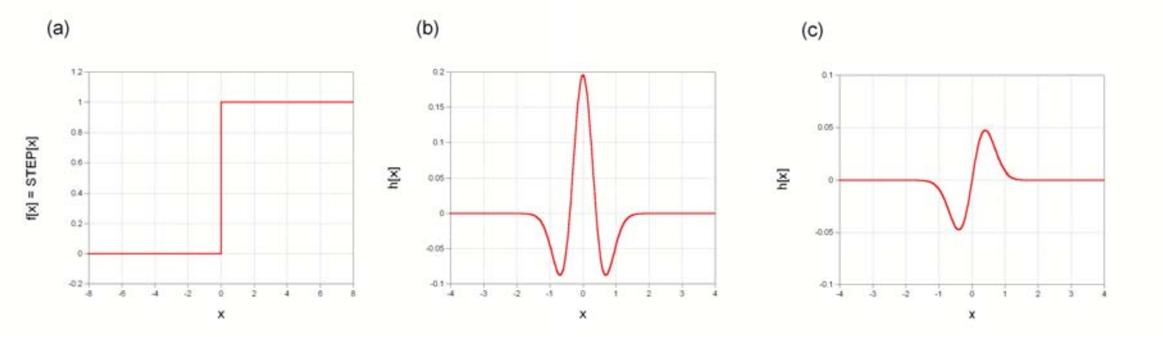
2-D impulse response of LoG operator with $\sigma = 1.4$

Because of the lateral extent of the kernel, the sampled version of the impulse response should be generated in a fairly large array, say 9×9 or larger. A 9×9 approximation of the LoG kernel with $\sigma = 1.4$ is

$$h[x,y] = \begin{array}{|c|c|c|c|c|c|c|c|c|}
\hline
0 & -1 & -1 & -2 & -2 & -2 & -1 & -1 & 0 \\
\hline
-1 & -2 & -4 & -5 & -5 & -5 & -4 & -2 & -1 \\
\hline
-1 & -4 & -5 & -3 & 0 & -3 & -5 & -4 & -1 \\
\hline
-2 & -5 & -3 & +12 & +24 & +12 & -3 & -5 & -2 \\
\hline
-2 & -5 & 0 & +24 & +40 & +24 & 0 & -5 & -2 \\
\hline
-2 & -5 & -3 & +12 & +24 & +12 & -3 & -5 & -2 \\
\hline
-1 & -4 & -5 & -3 & 0 & -3 & -5 & -4 & -1 \\
\hline
-1 & -2 & -4 & -5 & -5 & -5 & -4 & -2 & -1 \\
\hline
0 & -1 & -1 & -2 & -2 & -2 & -1 & -1 & 0 \\
\hline
\end{array}$$

The sum of the weights is zero, which means that the application of this operator will produce an image with mean gray value of zero.

The case of the 1-D LoG applied to an edge is shown below:



(a) 1-D edge; (b) 1-D impulse response of Laplacian of Gaussian; (c) output showing double response at edge.

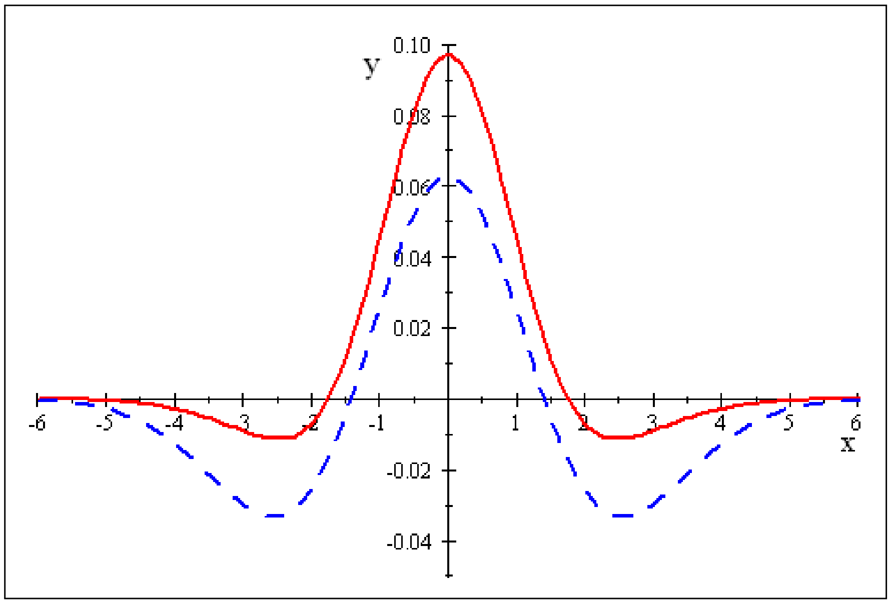
Difference of Gaussians

A difference of Gaussians (DoG) operation:

$$h[x,y] = \frac{1}{2\pi\sigma_1^2}\left(\frac{\sigma_1^2}{\sigma_2^2}\exp\left[-\frac{x^2+y^2}{2\sigma_2^2}\right] - \exp\left[-\frac{x^2+y^2}{2\sigma_1^2}\right]\right)$$

where $\sigma_1 > \sigma_2$. If $\sigma_1 > 1.6 \cdot \sigma_2$, then $h[x,y]$ closely approximates the LoG:

$$h[x,y] = \frac{1}{2\pi \cdot (1.6)^2}\left((1.6)^2 \exp\left[-\frac{x^2+y^2}{2}\right] - \exp\left[-\frac{x^2+y^2}{2\cdot(1.6)^2}\right]\right)$$



Profile of $h[x,0]$ for the "difference of Gaussians" kernel with $\sigma_1 = 1.6$ and $\sigma_2 = 1$ (in red) compared to LoG for $\sigma = 1.5$ (blue)

7.7 Nonlinear Filters

7.7.1 Median Filter

Probably the most useful nonlinear statistical filter is the local median, i.e., the gray value of the output pixel is the median of the gray values in a neighborhood, which is obtained by sorting the gray values in numerical order and selecting the middle value. To illustrate, consider the 3×3 neighborhood centered on the value "3" and the 9 values sorted in numerical order; the median value of "2" is indicated by the box and replaces the "3" in the center of the window:

$$\begin{array}{|c|c|c|}\hline 1 & 2 & 6 \\ \hline 2 & 3 & 5 \\ \hline 1 & 5 & 2 \\ \hline \end{array} \Longrightarrow \text{ordered sequence is } \begin{array}{|c|c|c|c|c|c|c|c|c|}\hline 1 & 1 & 2 & 2 & \boxed{2} & 3 & 5 & 5 & 6 \\ \hline \end{array} \text{ and the median is 2}$$

The nonlinear nature of the median can be recognized by noting that the median of the sum of two images is generally not equal to the sum of the medians. For example, the median of a second 3×3 neighborhood is "3"

$$\begin{array}{|c|c|c|}\hline 4 & 5 & 6 \\ \hline 3 & 1 & 2 \\ \hline 2 & 4 & 3 \\ \hline \end{array} \Longrightarrow \begin{array}{|c|c|c|c|c|c|c|c|c|}\hline 1 & 2 & 2 & 3 & \boxed{3} & 4 & 4 & 5 & 6 \\ \hline \end{array}$$

The sum of the two medians is $2+3 = 5$, but the sum of the two 3×3 neighborhoods produces a third neighborhood whose median of $6 \neq 2+3$

$$\begin{array}{|c|c|c|}\hline 5 & 7 & 12 \\ \hline 6 & 3 & 7 \\ \hline 3 & 9 & 5 \\ \hline \end{array} \Longrightarrow \begin{array}{|c|c|c|c|c|c|c|c|c|}\hline 3 & 3 & 5 & 5 & \boxed{6} & 7 & 7 & 9 & 12 \\ \hline \end{array}$$

confirming that the median of the sum is not the sum of the medians in this case.

The median requires sorting of the gray values, which may not be computed as a convolution. Its computation typically requires more time than a mean filter, but it has the advantage of reducing the modulation of signals that oscillate over a period less than the width of the median window while preserving the gray values of signals that are constant or monatonically increasing on a scale larger than the window size. This implies that the variance of additive noise will be reduced in a fashion similar to the mean filter, BUT it also tends to preserve edge structure. Unlike the mean filter, all gray values generated by the median must have been present in the original image, thus eliminating any need to requantize the processed gray values.

The statistics of the median-filtered image depend on the probability density function of the input signal, including the deterministic part and any noise. Thus predictions of the effect of the filter cannot be as specific as for the mean filter, i.e., given an input image with known statistics (mean, variance, etc.), the statistics of the output image are more difficult to predict. However, Frieden analyzed the statistical properties of the median filter by modeling it as a limit of a large number of discrete trials of a binomial probability distribution, which are often called "Bernouilli trials" (Frieden, "**Probability, Statistical Optics, and Data Testing**," Springer, p.257). The median of an odd number N of a set of samples for a set of gray values f_i taken from an input distribution with probability law (i.e., the histogram) $p_f[x]$ must be determined. Frieden applied the principles of Bernoulli trials to determine the probability density of the median of several independent sets of numbers. In other words, he sought to determine the probability that the median of the N numbers $\{f_n\}$ is x by evaluating the median of many independent such sets of N numbers selected from a known probability distribution $p_f[x]$. Frieden reasoned that, for each placement of the median window, a specific amplitude f_n of the N values is the median if three conditions are satisfied:

1. one of the N numbers satisfies the condition $x \leq f_n < x + \Delta x$

2. of the remaining $N-1$ numbers, $\dfrac{N-1}{2}$ exceed x and

3. $\dfrac{N-1}{2}$ of the remaining numbers are less than x.

The probability of the simultaneous occurence of these three events is the probability density of the output of the median window. For an arbitrary x, any one value f_n must either lie in the interval $(x \leq f < x + \Delta x)$, be larger than x, or less than x. In other words, each trial has three possible outcomes. These conditions define a sequence of Bernoulli trials with three outcomes, which produce results akin to those from the flipping of a "three-sided" coin where the probabilities of the three outcomes are not equal. In the more familiar case, the probability that N coin flips with two possible outcomes that have associated probability p and q will produce m "successes" (say, m heads) and $N-m$ "failures" (tails) is:

$$P_N[m] = \frac{N!}{(N-m)!m!} \cdot p^m \cdot (1-p)^{N-m}$$

The formula is easy to extend to the more general case of three possible outcomes; the probability that the result yields m_1 instances of the first possible outcome (say, "heads #1), m_2 of the second outcome ("head #2") and $m_3 = N - (m_1 + m_2)$ of the third ("head #3") is

$$\begin{aligned} P_N[m_1, m_2, m_3] &= P_N[m_1, m_2, N - (m_1 + m_2)] \\ &= \frac{N!}{m_1! m_2! (N - (m_1 + m_2))!} \cdot p_1^{m_1} \cdot p_2^{m_2} \cdot p_3^{m_3} \\ &= \frac{N!}{m_1! m_2! (N - (m_1 + m_2))!} \cdot p_1^{m_1} \cdot p_2^{m_2} \cdot (1 - (p_1 + p_2))^{N-(m_1+m_2)} \end{aligned}$$

where p_1, p_2, and $p_3 = 1 - (p_1 + p_2)$ are the respective probabilities of the three outcomes.

When applied to one sample of data, the median filter has three possible outcomes whose probabilities are known: (1) the sample amplitude may be the median (probability p_1) (2) the sample amplitude may be smaller than the median (probability p_2), and (3) it may be larger than the median (probability p_3).

$$\begin{aligned} p_1 &= P\left[x \leq f_n \leq x + \Delta x\right] = p_f\left[x\right] \\ p_2 &= P\left[f_n < x\right] = C_f\left[x\right] \\ p_3 &= P\left[f_n > x\right] = 1 - C_f\left[x\right] \end{aligned}$$

where $C_f\left[x\right]$ is the cumulative probability distribution of the continuous probability density function $p_f\left[x\right]$:

$$C_f\left[x\right] = \int_{-\infty}^{x} p_f\left[\alpha\right]\ d\alpha$$

In this case, the distibutions are continuous (rather than discrete), so the probability is the product of the probability density function $p_{med}\left[x\right]$ and the infinitesmal element dx. We substitute the known probabilities and the known number of occurences of each into the Bernoulli formula for three outcomes:

$$\begin{aligned} p_{med}\left[x\right]\ dx &= \frac{N!}{\left(\frac{N-1}{2}\right)! \cdot \left(\frac{N-1}{2}\right)! \cdot 1!} \left(C_f\left[x\right]\right)^{\frac{N-1}{2}} \cdot \left(1 - C_f\left[x\right]\right)^{\frac{N-1}{2}} \cdot p_f\left[x\right]\ dx \\ &= \frac{N!}{\left(\left(\frac{N-1}{2}\right)!\right)^2} \left(C_f\left[x\right]\right)^{\frac{N-1}{2}} \cdot \left[1 - C_f\left[x\right]\right]^{\frac{N-1}{2}}\ p_f\left[x\right]\ dx \end{aligned}$$

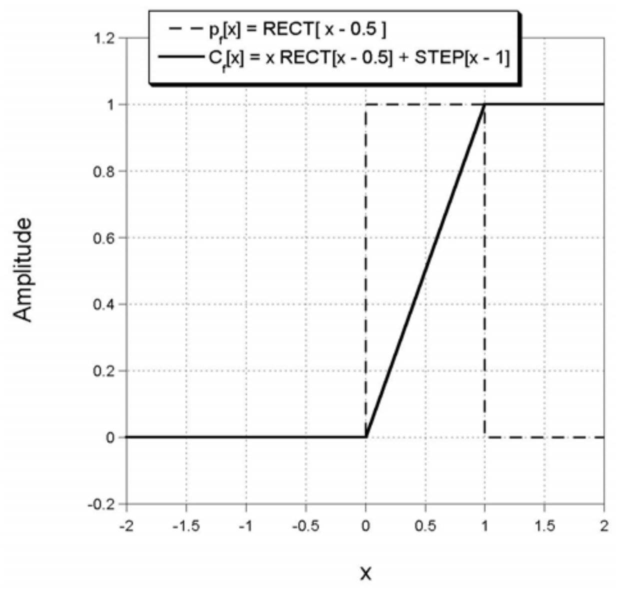
If the window includes $N = 3$, 5, or 9 values, the following probability laws for the median result:

$$\begin{aligned} N = 3 &\implies p_{med}\left[x\right]\ dx = \frac{3!}{(1!)^2} \left(C_f\left[x\right]\right)^1 \cdot \left(1 - C_f\left[x\right]\right)^{\frac{N-1}{2}} \cdot p_f\left[x\right]\ dx \\ &= 6(C_f\left[x\right]) \cdot \left(1 - C_f\left[x\right]\right)\ p_f\left[x\right]\ dx \\ N = 5 &\implies p_{med}\left[x\right]\ dx = \frac{5!}{(2!)^2} \left(C_f\left[x\right]\right)^2 \cdot \left[1 - C_f\left[x\right]\right]^2\ p_f\left[x\right]\ dx \\ &= 30(C_f\left[x\right])^2 \cdot \left(1 - C_f\left[x\right]\right)^2\ p_f\left[x\right]\ dx \\ N = 9 &\implies p_{med}\left[x\right]\ dx = 630(C_f\left[x\right])^4 \cdot \left(1 - C_f\left[x\right]\right)^4\ p_f\left[x\right]\ dx \end{aligned}$$

7.7.2 Example of Median Filter of Uniform Distribution

The statistical properties of the median will now be demonstrated for some simple examples of known probabilities. If the original pdf $p_f\left[x\right]$ is uniform over the interval $[0, 1]$, then it may be written as a rectangle function:

$$p_f\left[x\right] = RECT\left[x - \frac{1}{2}\right]$$



pdf of noise that is uniformly distributed over the interval $[0,1]$ and its associated cumulative probability distribution $F_c[x] = x \cdot RECT\left[x-\frac{1}{2}\right] + STEP[x-1]$

The associated cumulative probability distribution may be written in several ways, including:

$$\begin{aligned} C_f[x] &= \int_{-\infty}^{x} p_f(\alpha)d\alpha \\ &= x \cdot RECT\left[x-\frac{1}{2}\right] + STEP[x-1] \end{aligned}$$

so the product of the cumulative distribution and its complement is windowed by the rectangle to yield:

$$p_{\text{median}}[x]\;dx = \frac{N!}{\left(\left(\frac{N-1}{2}\right)!\right)^2}\left(x^{\frac{N-1}{2}} \cdot (1-x)^{\frac{N-1}{2}}\right)\;RECT\left[x+\frac{1}{2}\right]\;dx$$

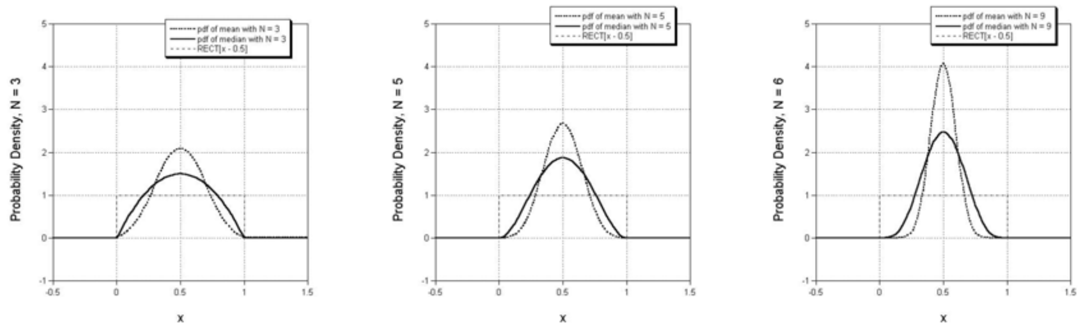
The pdfs of the output of median filters for $N = 3$, 5, and 9 are:

$$N=3 \implies p_{\text{median}}[x]\;dx = 6\left(x-x^2\right)\;RECT\left[x-\frac{1}{2}\right]\;dx$$

$$N=5 \implies p_{\text{median}}[x]\;dx = 30\left(x^4-2x^3+x^2\right)\;RECT\left[x-\frac{1}{2}\right]\;dx$$

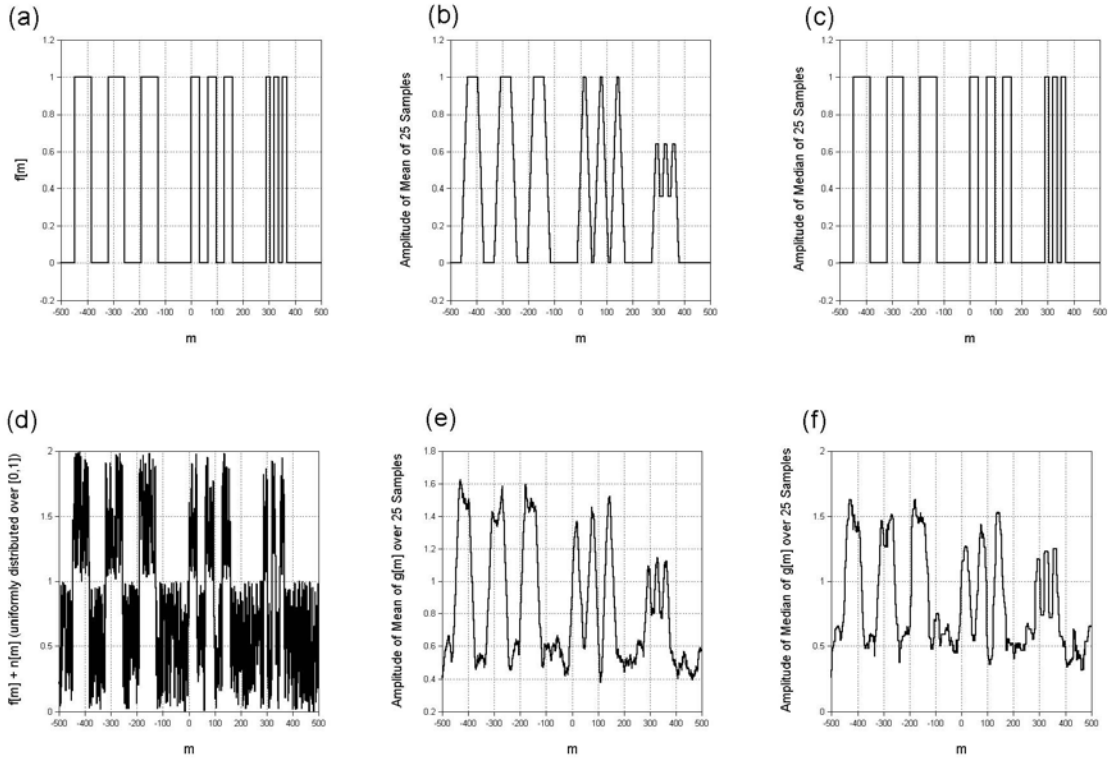
$$N=9 \implies p_{\text{median}}[x]\;dx = 630\cdot\left(x^8-4x^7+6x^6-4x^5+x^4\right)\;RECT\left[x-\frac{1}{2}\right]\;dx$$

are compared to the pdfs of the output of the mean filters in the figure:



Comparison of pdfs of mean and median filter for uniform probability density function $p_f[x] = RECT\left[x + \frac{1}{2}\right]$ for $N = 3$, 5, and 9. *Note that the pdf of the mean filter is "taller" and "skinnier" in all three cases, showing that it will reduce the variance more than the median filter.*

Just like the mean filter, the maximum value of $p_{\text{median}}[x]$ increases and its width decreases as the number of input values in the median window increases (as $N \uparrow$). The calculated pdfs for the median and mean filters over $N = 3$ and $N = 5$ samples for input values from a uniform probability distribution are shown below to the same scale. Note that the output distributions from the mean filter are taller than for the median, which indicates that the median filter does a poorer job over averaging noise than the mean (Frieden determined that the S/N ratio of the median filter is smaller than that of the mean by a factor of $\log_e[2] \cong 0.69$, so that there is a penalty in S/N of about 30% for the median filter relative to the averaging filter. Put another way, the standard deviation of the median of N samples decreases as $\left(\sqrt{N} \cdot \log_2[2]\right)^{-1}$ instead of $\left(\sqrt{N}\right)^{-1}$. The lesser noise reduction of the median filter is offset by its ability to preserve the sharpness of edges.



Comparison of mean and median filter: (a) bitonal object $f[m]$ defined over 1000 samples; (b) output of mean filter over 25 samples, showing reduction in contrast with decreasing period; (c) median of $f[m]$ over 25 samples, which is identical to $f[m]$; (d) $f[m]+n[m]$, which is uniformly distributed over interval $[0,1]$; (e) mean over 25 samples; (f) median over 25 samples. Note that the highest-frequency bars are better preserved by the median filter.

7.8 Median Filter and Gaussian Noise

Probably the most important application of the median filter is to attenuate Gaussian noise (i.e., the gray values are selected from a normal distribution with zero mean) without blurring edges. The central limit theorem indicates that the statistical character of noise which has been generated by summing random variables from different distributions will be Gaussian in character. The probability distribution function is the Gaussian with mean value μ and variance σ^2 normalized to unit area:

$$p_f[x] = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(x-\mu)^2}{2\sigma^2}\right]$$

The cumulative probability density of this noise is the integral of the Gaussian probability law, which is proportional to the error function:

$$\text{erf}\,[x] \equiv \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

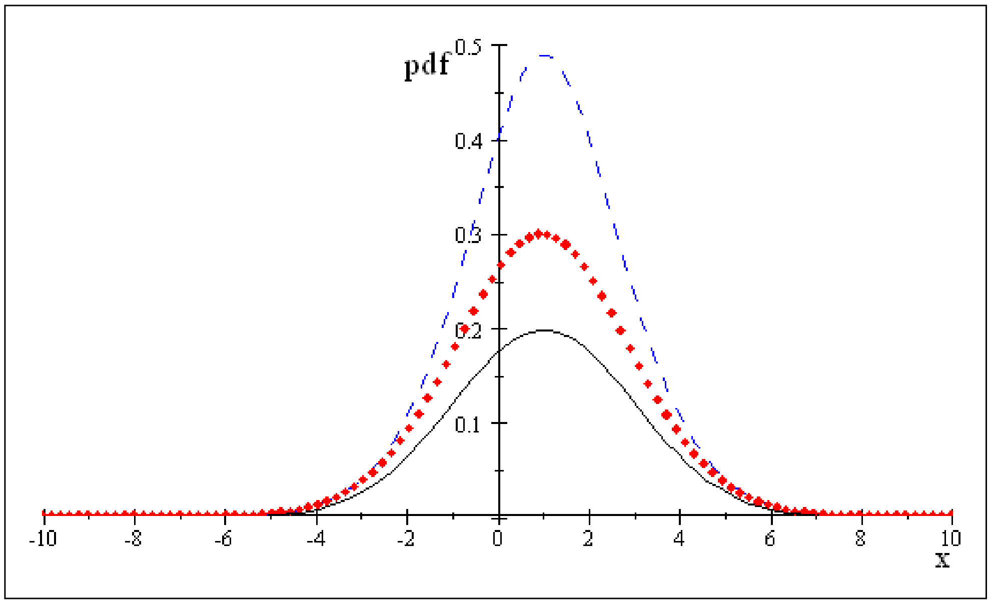
We can evaluate the cumulative density in terms of erf $[x]$:

$$C_f\,[x] = \int_{-\infty}^{x} p_c\,[x]\,dx \equiv \frac{1}{2} - \frac{1}{2\sqrt{2}\sigma}\,\text{erf}\left[\frac{x-\mu}{\sqrt{2}\sigma}\right] \quad \text{for } x \le \mu$$
$$C_f\,[x] = \int_{-\infty}^{x} p_c\,[x]\,dx \equiv \frac{1}{2} + \frac{1}{2\sqrt{2}\sigma}\,\text{erf}\left[\frac{x-\mu}{\sqrt{2}\sigma}\right] \quad \text{for } x \ge \mu$$

Therefore the probabilities of the different outcomes of the median filter are:

$$p_{med}\,[x]\;dx = \frac{N!}{\left(\left(\frac{N-1}{2}\right)!\right)^2}\left(\frac{1}{2} + \frac{1}{2\sqrt{2}\sigma}\,\text{erf}\left[\frac{x-\mu}{\sqrt{2}\sigma}\right]\right)^{\frac{N-1}{2}} \cdot \left(\frac{1}{2} - \frac{1}{2\sqrt{2}\sigma}\,\text{erf}\left[\frac{x-\mu}{\sqrt{2}\sigma}\right]\right)^{\frac{N-1}{2}} \cdot \frac{1}{\sqrt{2\pi}\sigma}\,\text{exp}\left[-\frac{x^2}{2\sigma^2}\right.$$

The error function is compiled and may be evaluated to plot the probability $p_{\text{median}}\,[x]$



pdf of Gaussian noise with $\mu = 1$, $\sigma = 2$ (black) and of the median for $N = 3$ (red), $N = 9$ (blue).

The graphs illustrate the theoretical averaging effects of the mean and median filters on Gaussian noise. The graphs are plotted on the same scale and show the pdf of the original Gaussian noise (on the left) and the output resulting from mean and median filtering over 3 pixels (center) and after mean and median filtering over 5 pixels (right). The calculated mean gray value and standard deviation for 2048 samples of filtered

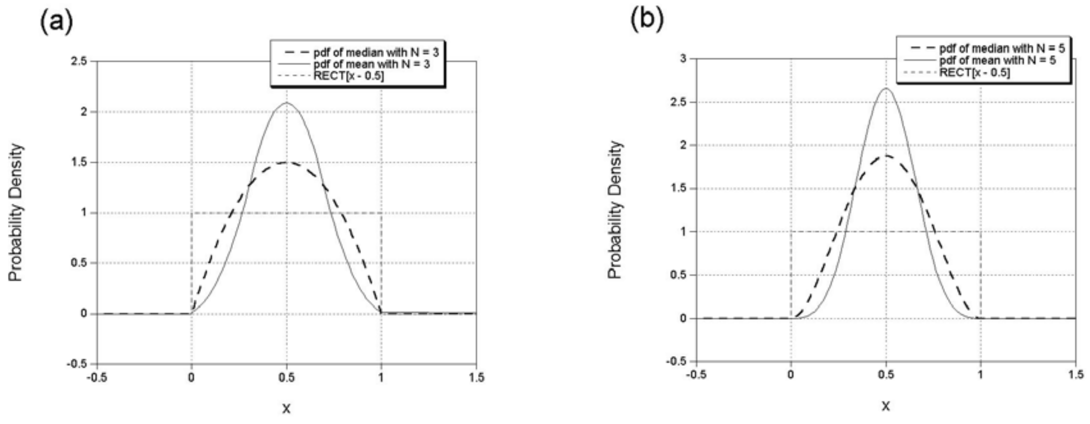


Figure 7.1: *Comparison of histograms resulting from mean and median filtering of noise that is uniformly distributed after averaging over $N = 3$ and $N = 5$. The histogram obtained from the mean filter is "taller" and "skinnier" than that from the median, showing that mean filters reduce the variance more than median filters for both cases. This advantage is offset by the edge-preserving property of the median.*

Gaussian noise yielded the following values:

$$\begin{aligned} \mu_{in} &= 0.211 \\ \sigma_{in} &= 4.011 \\ \mu_3 - mean &= 0.211 \\ \sigma_3 - mean &= 2.355 \\ \mu_3 - median &= 0.225 \\ \sigma_3 - median &= 2.745 \end{aligned}$$

7.9 Comparison of Histograms after Mean and Median Filter

The input is the blue house image. The first set is the original image and histogram, followed by the 3×3 mean-filtered image and the 3×3 median filtered. Note that the median does a better job of segmenting the peaks of the histogram while maintaining image sharpness (e.g. around the door of the house).

7.9.1 Effect of Window "Shape" on Median Filter

In the 2-D imaging case, the shape of the window over which the median is computed also affects the output image. For example, if the 2-D median is computed over a

5×5 window at the corner of a dark object on a bright background, the median will be the background value:

$$\text{median of } \begin{array}{|c|c|c|c|c|}\hline 0&0&0&0&0\\\hline 0&0&0&0&0\\\hline 1&1&1&0&0\\\hline 1&1&1&0&0\\\hline 1&1&1&0&0\\\hline \end{array} = 0$$

The median calculated over a full square window (3x3, etc.) will convert bright pixels at outside corners of bright object to dark pixels, i.e., the corners will be clipped; it will also convert a dark background pixel at the inside corner of a bright object to a bright pixel. It will also eliminate lines less than half as wide as the window. Corner clipping may be prevented by computing the median over a window that only includes 9 values arrayed along horizontal and vertical lines:

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

If applied to the pixel in the corner, we obtain

$$\text{median of } \begin{array}{|c|c|c|c|c|}\hline 0&0&0&0&0\\\hline 0&0&0&0&0\\\hline 1&1&1&0&0\\\hline 1&1&1&0&0\\\hline 1&1&1&0&0\\\hline \end{array} \times \begin{array}{|c|c|c|c|c|}\hline -&-&1&-&-\\\hline -&-&1&-&-\\\hline 1&1&1&1&1\\\hline -&-&1&-&-\\\hline -&-&1&-&-\\\hline \end{array} = \text{median of } \begin{array}{|c|c|c|c|c|}\hline -&-&0&-&-\\\hline -&-&0&-&-\\\hline 1&1&1&0&0\\\hline -&-&1&-&-\\\hline -&-&1&-&-\\\hline \end{array} = 1$$

This pattern also is effective when applied to thin lines without elimating them:

$$\text{median of}\begin{array}{|c|c|c|c|c|}\hline 0&0&0&0&0\\\hline 0&0&0&0&0\\\hline 1&1&1&1&1\\\hline 0&0&0&0&0\\\hline 0&0&0&0&0\\\hline\end{array}\times\begin{array}{|c|c|c|c|c|}\hline -&-&1&-&-\\\hline -&-&1&-&-\\\hline 1&1&1&1&1\\\hline -&-&1&-&-\\\hline -&-&1&-&-\\\hline\end{array}=\text{median of}\begin{array}{|c|c|c|c|c|}\hline -&-&0&-&-\\\hline -&-&0&-&-\\\hline 1&1&1&1&1\\\hline -&-&0&-&-\\\hline -&-&0&-&-\\\hline\end{array}=1$$

Other patterns of medians are also useful [**?**, Castleman, Digital Image Processing, Prentice-Hall, 1996, p. 249].

7.9.2 Other Statistical Filters (Mode, Variance, Maximum, Minimum)

The statistical mode in the window (i.e., the most common gray level) is a useful operator on binary images corrupted by isolated noise pixels ("salt-and-pepper noise"). The mode is found by computing a mini-histogram of pixels within the window and assigning the most common gray level to the center pixel. Rules must be defined if two or more gray levels are equally common, and particularly if all levels are populated by a single pixel. If two levels are equally populated, the gray level of center pixel is usually retained if it is one of those levels; otherwise one of the most common gray levels may be selected at random.

The *variance filter* σ^2 and *standard deviation filter* σ replace the center pixel with the variance or standard deviation of the pixels in the window, respectively. The variance filtering operation is

$$g[x,y]=\sum_{window}(f[x,y]-\mu)^2$$

where μ is the mean value of pixels in the window. The output of a variance or standard deviation operation will be larger in areas where the image is busy and small where the image is smooth. The output of the σ-filter resembles that of the isotropic Laplacian, which computes the difference of the center pixel and the average of the eight nearest neighbors.

The *Maximum* or *Minimum filter* obviously replace the gray value in the center with the highest or lowest value in the window. The MAX filter will dilate bright objects, while the MIN filter erodes them. These provide the basis for the so-called *morphological operators*. A "dilation" (MAX) followed by an "erosion" (MIN) defines the morphological "CLOSE" operation, while the opposite (erosion followed by dilation) is an "OPEN" operation. The "CLOSE" operation fills gaps in lines and removes isolated dark pixels, while OPENING removes thin lines and isolated bright pixels. These nonlinear operations are useful for object size classification and distance measurements.

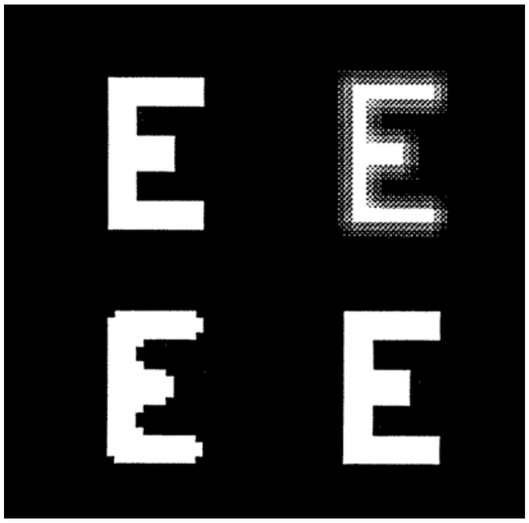
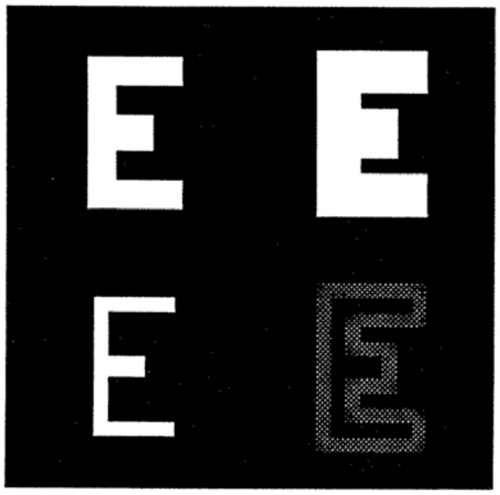


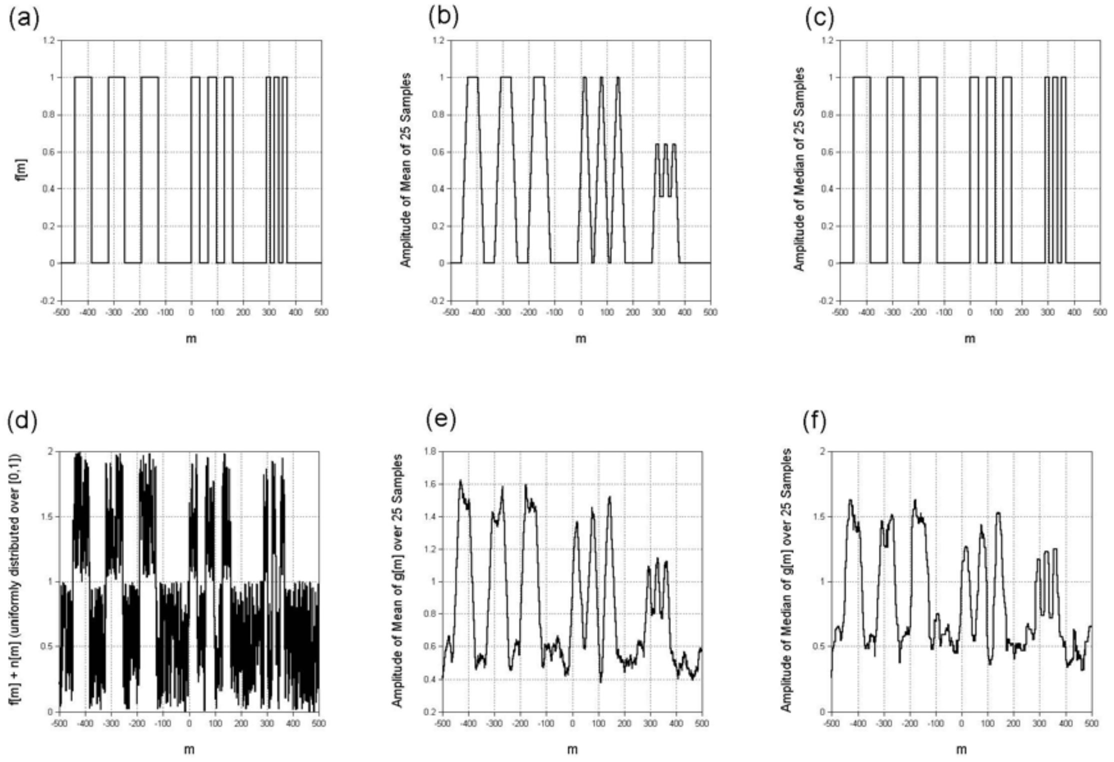
Figure 7.2: *E (upper left) and after* 3×3 *filtering by mean (upper right),* 3×3 *median (lower left), and* 3×3 *median with clipped corners (lower right)*



E (upper left) and after 3×3 *maximum filter (upper right),* 3×3 *minimum (lower left), and* 3×3 *variance (lower right).*

7.9.3 Examples of Nonlinear Filters

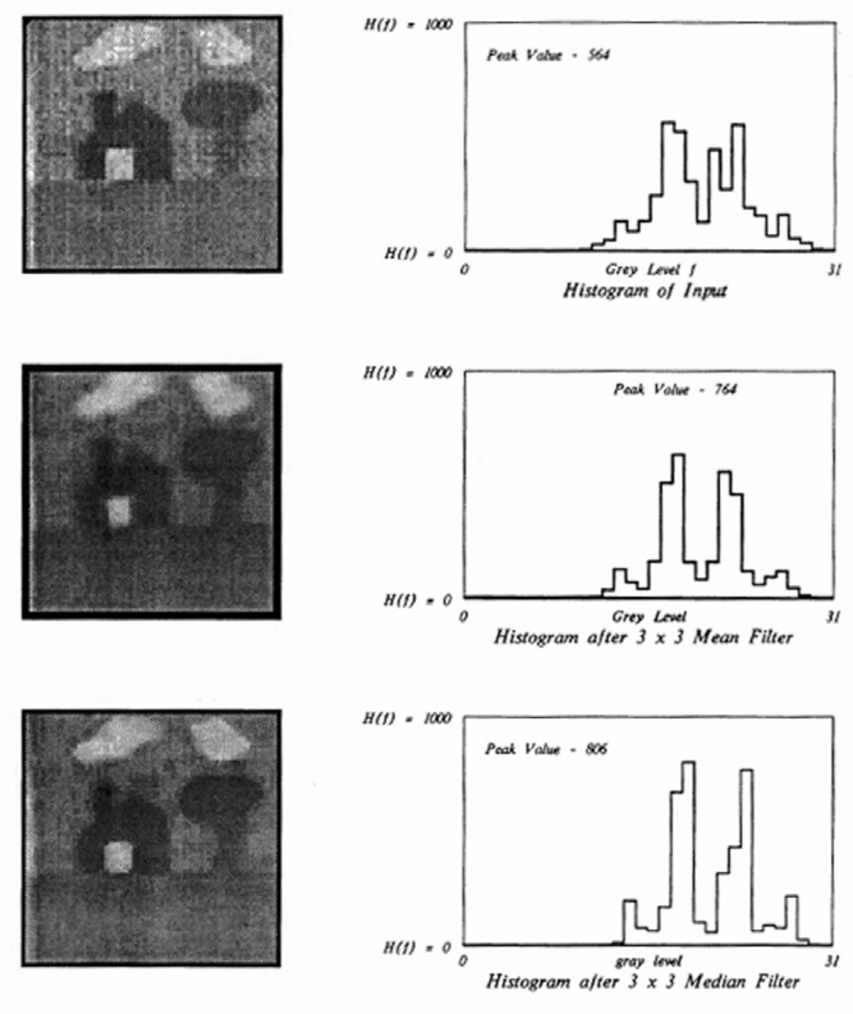
Consider 1-D examples of the filtering operations just reviewed:



Comparison of mean and median filter: (a) bitonal (two-level) object $f[m]$ defined over 1000 samples; (b) mean of $f[m]$ over 25 samples; (c) median of $f[m]$ over 25 samples, which is identical to $f[m]$ since the median evaluates to the bitonal values; (d) $f[m]+n[m]$, where the noise is uniformly distributed over interval $[0,1]$; (e) mean over 25 samples; (f) median over 25 samples. Note that the highest-frequency bars are (slightly) better preserved by the median filter.

Below are pictured the results of the filtering operations just reviewed. In each example, the upper-left quadrant is the original image and the other three are the outputs of the filters.

In the first sequence, the images were filtered with 3×3 operators. The first image includes the result of the 3×3 mean filter (upper right), 3×3 median (lower left), and 3×3 median with clipped corners (lower right). The second image is comprised of the 3×3 maximum filter (upper right), 3×3 minimum (lower left), and 3×3 variance.



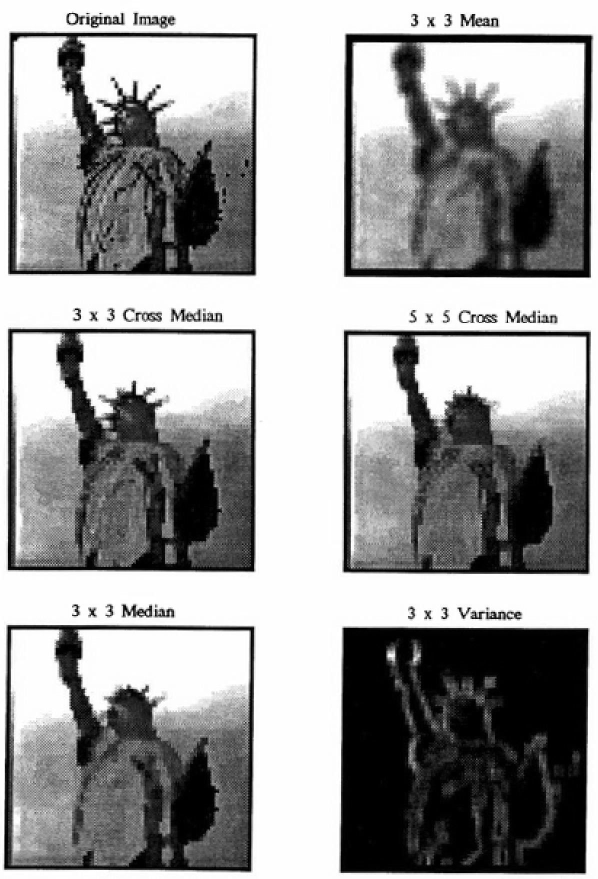
The second set of images is the same sequence of operations for 5×5 windows.

7.9.4 Nonlinear Filters on Images with Additive Gaussian Noise

The dynamic range of the original image is in the range $[0, 1]$ with added Gaussian noise ($\sigma = 0.5$). The image sequences are just as before.

7.9.5 Nonlinear Filters on Noise-Free Gray-Level Image

Note that the points of the crown are clipped by the 3×3 median filter, and that the edges are apparent in the 3×3 variance filter.



Note that the "points" of the crown are clipped by the 3 x 3 median filter, and that the edges are apparent in the 3 x 3 variance filter

7.10 Adaptive Operators

In applications such as edge enhancement or segmentation, it is often useful to "change", or "adapt" the operator based on conditions in the image. One example has already been considered: the nonlinear normalization used while convolving with a bipolar convolution kernel. For another example, it is possible to enhance differences in the direction of the local gradient (e.g. via a 1-D Laplacian) while averaging in the orthogonal direction. In other words, the operator used to enhance the edge information is determined by the output of the gradient operator. As another example, the size of an averaging neighborhood could be varied based on the statistics (e.g., the variance) of gray levels in the neighborhood.

In some sense, these adaptive operators resemble cascaded convolutions, but the resulting operation is not space invariant and may not be desribed by convolution with a single kernel. By judicious choice of algorithm, significant improvement of image quality may be obtained.

7.11 Convolution Revisited – Bandpass Filters

The parameters of a filter that determine its effect on the image are the size of the kernel and the algebraic sign of its coefficients. Kernels whose elements have the same algebraic sign are lowpass filters that compute spatial averages and attenuate the modulation of spatial structure in the image. The larger the kernel, the greater the attenuation. On the other hand, kernels that compute differences of neighboring gray levels will enhance the modulation of spatially varying structure while attenuating the brightness of constant areas. Note that the largest number of elements in a kernel with different algebraic signs is two; the spatial first derivative is an example.

We will now construct a hybrid of these two extreme cases that will attenuate the modulation of image structure that varies more slowly or rapidly than some selectable rate. In other words, the filter will pass a band of spatial frequencies and attenuate the rest of the spectrum; this is a bandpass filter. The bandpass filter will compute differences of spatial averages of gray level. For example, consider a 1-D image:

$$\begin{aligned} f[n] &= 1+\sum_{i=0}^{2}\cos\left[\frac{2\pi n(2^i)}{128}\right] = 1+\cos\left[\frac{2\pi n}{\infty}\right]+\cos\left[\frac{2\pi n}{128}\right]+\cos\left[\frac{2\pi n}{64}\right]+\cos\left[\frac{2\pi n}{32}\right] \\ &= 2+\cos\left[\frac{2\pi n}{128}\right]+\cos\left[\frac{2\pi n}{64}\right]+\cos\left[\frac{2\pi n}{32}\right] \end{aligned}$$

The spatial frequencies of the cosines are:

$$\begin{aligned} \xi_0 &= \tfrac{1}{\infty} = 0 \text{ cycles per sample} \Longrightarrow X_0 = \infty \\ \xi_1 &= \tfrac{1}{128} \simeq 7.8\cdot 10^{-2} \text{ cycles per sample} \Longrightarrow X_1 = 128 \text{ samples} \\ \xi_2 &= \tfrac{1}{64} \text{ cycles per sample} \Longrightarrow X_2 = 64 \text{ samples} \\ \xi_3 &= \tfrac{1}{32} \text{ cycles per sample} \Longrightarrow X_3 = 32 \text{ samples} \end{aligned}$$

This function is periodic over 128 samples, which is a common multiple of all of the finite-period cosines. The extreme amplitudes are +5 and +0.2466. Consider convolution of $f[n]$ with several kernels; the first set are 3-pixel averagers whose weights sum to unity, therefore preserving the mean gray level of $f[n]$:

$$h_1[n] = \boxed{0 \mid 1 \mid 0}$$

$$h_2[n] = \boxed{\tfrac{1}{4} \mid \tfrac{1}{2} \mid \tfrac{1}{4}}$$

$$h_3[n] = \boxed{\tfrac{1}{3} \mid \tfrac{1}{3} \mid \tfrac{1}{3}}$$

Obviously, $h_1[n]$ is the identity kernel, h_2 is a tapered averager that applies more weight to the center pixel, while h_3 is a uniform averager. Based on our experience with averaging filters, we know that $g_1[n] = f[n]*h_1[n]$ must be identical to $f[n]$, while the modulation of the output from $h_2[n]$ will be reduced a bit in g_2 and somewhat more in g_3. This expectation is confirmed by the computed maximum and minimum values:

$f_{\max} = 5$	$(g_1)_{\max} = 5$	$(g_2)_{\max} \cong 4.987$	$(g_3)_{\max} \cong 4.983$
$f_{\min} \cong 0.2466$	$(g_1)_{\min} \cong 0.2466$	$(g_2)_{\min} \cong 0.2564$	$(g_3)_{\min} \cong 0.2596$

The mean gray values of these images are identical:

$$\langle f \rangle = \langle g_1 \rangle = \langle g_2 \rangle = \langle g_3 \rangle = 2$$

We can define a contrast factor based on these maximum and minimum values that is analogous to the modulation, except that the image is not sinusoidal:

$$c_f \equiv \frac{f_{\max} - f_{\min}}{f_{\max} + f_{\min}}$$

The corresponding factors are:

$c_f = 0.906$	$c_1 \cong 0.906$	$c_2 \cong 0.9022$	$c_3 \cong 0.9019$

which confirms the expectation.

Now consider three 5-pixel averagers:

$$h_4[n] = \boxed{0} \boxed{0} \boxed{1} \boxed{0} \boxed{0}$$
$$h_5[n] = \boxed{\tfrac{1}{9}} \boxed{\tfrac{2}{9}} \boxed{\tfrac{3}{9}} \boxed{\tfrac{2}{9}} \boxed{\tfrac{1}{9}},$$
$$h_6[n] = \boxed{\tfrac{1}{5}} \boxed{\tfrac{1}{5}} \boxed{\tfrac{1}{5}} \boxed{\tfrac{1}{5}} \boxed{\tfrac{1}{5}}$$

Again, h_4 is the identity kernel that reproduces the modulation of the original image, h_2 is a tapered averager, and h_6 is a uniform averager. The computed maximum and minimum values for the images are:

$f_{\max} = 5$	$(g_4)_{\max} = 5$	$(g_5)_{\max} \cong 4.967$	$(g_6)_{\max} \cong 4.950$
$f_{\min} \cong 0.2466$	$(g_4)_{\min} \cong 0.2466$	$(g_5)_{\min} \cong +0.272$	$(g_6)_{\min} \cong 0.285$
$\langle f \rangle = 2$	$\langle g_4 \rangle = 2$	$\langle g_5 \rangle = 2$	$\langle g_6 \rangle = 2$
$c_f = 0.906$	$c_4 = 0.906$	$c_5 = 0.896$	$c_6 = 0.891$

The average over a larger number of samples reduces the modulation further but does not affect the mean gray values.

Now consider a kernel of significantly larger width:

$$h_7[n] = A_7 \cos\left[\frac{2\pi n}{64}\right] \cdot RECT\left[\frac{n}{32}\right]$$

where the scale factor A_7 is used to normalize $h_7[n]$ to unit area. The *RECT* function limits the support of the cosine to a finite width of 32 pixels. Note that the kernel is large and is nonnegative everywhere; this is another example of a tapered averaging kernel that weights pixels in the vicinity more heavily than more distant pixels. The corresponding uniform averaging kernel is:

$$h_8[n] = \frac{1}{32} RECT\left[\frac{n}{32}\right]$$

To simplify comparison of the results of h_7 and h_8, we will set $A_7 = \frac{1}{32}$ instead of a factor that ensures a unit area. The exact value of the scale factor will affect the output amplitudes and not the modulation. Based on the experience gained for $h_1, h_2, \cdots, h_6$, we expect that both h_7 and h_8 will diminish the modulation of spatially varying patterns, and that h_8 will have the larger effect. In fact, because the width of h_8 matches the period $X_3 = 32$, this cosine term will be attenuated to null amplitude. The effects on the amplitudes of the array are:

$f_{\max} = 5$	$(g_7)_{\max} \simeq 2.585$	$(g_8)_{\max} \cong 3.536$
$f_{\min} \cong 0.2466$	$(g_7)_{\min} \cong 0.593$	$(g_8)_{\min} \cong +1.205$
$\langle f \rangle = 2$	$\langle g_7 \rangle = 2$	$\langle g_8 \rangle = 2$
$c_f = 0.906$	$c_7 \cong 0.627$	$c_8 \cong 0.492$

which again confirms the expectation that the uniform averager reduces the contrast to a greater degree than the tapered averager, but neither affects the mean gray value.

If the width of the RECT function in the tapered averaging kernel h_7 is increased while the period of the constituent cosine function is retained, negative weights are added to the kernel. For example:

$$h_9[n] = A_9 \cos\left[2\pi\frac{n}{64}\right] \; RECT\left[\frac{n}{48}\right]$$

The constant A_9 may be chosen so that the area of h_9 is unity, yielding $A_9 = 0.06948$, or A_9 may be set to $\frac{1}{48}$, matching the normalization factor for the uniform averager:

$$h_{10}[n] = \frac{1}{48} RECT\left[\frac{n}{48}\right]$$

which simplifies comparison of the resulting amplitudes. Kernel h_9 computes the same weighted average as h_7 in the neighborhood of the pixel, but then subtracts a weighted average of distant pixels from it; it computes differences of average amplitudes. The effects of these operations on the extrema are:

$f_{\max} = 5$	$(g_9)_{\max} \simeq 1.532$	$(g_{10})_{\max} \cong 2.940$
$f_{\min} \cong 0.2466$	$(g_9)_{\min} \cong 0.118$	$(g_{10})_{\min} \cong +1.420$
$\langle f \rangle = 2$	$\langle g_9 \rangle \cong 0.5997$	$\langle g_{10} \rangle = 2$
$c_f = 0.906$	$c_9 \cong 0.857$	$c_{10} \cong 0.349$

Kernel h_{10} retained the mean value but further attenuated the contrast by pushing the amplitudes toward the mean. However, the difference-of-averages kernel h_9 actually increased the contrast and decreased the mean value.

This trend may be continued by increasing the width of the RECT and using equal scale factors:

$$h_{11}[n] = \frac{1}{64}\cos\left[2\pi\frac{n}{64}\right] RECT\left[\frac{n}{64}\right]$$
$$h_{12}[n] = \frac{1}{64} RECT\left[\frac{n}{64}\right]$$

Because the width of the RECT matches the period of the cosine in h_1, it may not be normalized to unit area. The extrema of these two processes are:

$f_{\max} = 5$	$(g_{11})_{\max} \simeq 0.712$	$(g_{12})_{\max} = 2 + \frac{2}{\pi} \cong 2.637$
$f_{\min} \cong 0.2466$	$(g_{11})_{\min} \cong -0.511$	$(g_{12})_{\min} \cong +1.364$
$\langle f \rangle = 2$	$\langle g_{11} \rangle \cong 0$	$\langle g_{12} \rangle = 2$
$c_f = 0.906$	$c_{11} \cong 6.08$ (?!!)	$c_{12} \cong 0.318$

The uniform averager h_1 continues to push the amplitudes toward the mean value of 2 and decreases the contrast, while the mean amplitude generated by the difference of averages kernel is now zero, which means that the minimum is less than zero.

Note that the output $g_{11}[n]$ looks like the kernel $h_{11}[n]$; in other words, the portion of $f[n]$ that was transmitted to $g_{11}[n]$ largely is a cosine of period 64. The distortion is due to cosines at other frequencies. Now, consider filters whose widths are equal to the period of $f[n]$:

$$h_{13}[n] = \cos\left[2\pi\frac{n}{128}\right] \cdot \frac{1}{128} RECT\left[\frac{n}{128}\right]$$
$$h_{14}[n] = \frac{1}{128} RECT\left[\frac{n}{128}\right].$$

The figures of merit for the gray values of these arrays are:

$f_{\max} = 5$	$(g_{13})_{\max} = 0.5$	$(g_{14})_{\max} = +2$
$f_{\min} \cong 0.2466$	$(g_{13})_{\min} \cong -0.5$	$(g_{14})_{\min} = +2$
$\langle f \rangle = 2$	$\langle g_{13} \rangle = 0$	$\langle g_{14} \rangle = 2$
$c_f = 0.906$	$c_{13} = \infty$	$c_{14} = 0.0$

The output of the bipolar kernel$h_{13}$is a sinusoid with period 64 and zero mean, while that of the averager h_{14} is the constant average value of $f[n]$. Note that the contrast parameter c_{13} is undefined because $f_{min} = -f_{max}$, while $c_{14} = 0$.

To summarize, kernels that compute differences of averages are wide and bipolar, and typically yield bipolar outputs. As the width of a difference-of-averages kernel is increased, the output resembles the kernel itself to a greater degree, which is bipolar with zero mean. On the other hand, increasing the width of an averaging operator results in outputs that approach a constant amplitude (the average value of the input); this constant is a cosine with infinite period. The difference-of-averages operator rejects BOTH slowly and rapidly varying sinusoids, and preferentially passes a particular sinusoidal frequency or band of frequencies. Thus, differences of averages operators are called bandpass filters.

$\boxed{\textit{Kernels of bandpass filters are wide, bipolar, and resemble the signal to be detected.}}$

7.11.1 Bandpass Filters for Images

We can extend the concept of the bandpass filter to images. For example, consider the cases shown below:

Because bandpass kernels are large, the computation will be time-consuming and the kernel will run over the edge of the image for small translations. These problems are by performing the convolution via the discrete Fourier transform.

7.12 Pattern Matching

We now explore the principles of the matched filter that may be applied to design kernels for locating specific gray-level patterns, such as edges at particular orientations, corners, isolated pixels, particular shapes, you name it. Particularly in the early days of digital image processing when computers were less capable than they are today, the computational intensity of the calculation often was an important issue. It was desirable to find the least intensive method for common tasks such as pattern detection, which generally meant that the task was performed in the space domain using a small convolution kernel rather than calculating a better approximation to the ideal result in the frequency domain. That said, the process of designing and applying a pattern-matching kernel illuminates some of the concepts and thus is worth some time and effort.

A common technique for pattern matching convolves the input image with a kernel of the same size as the reference pattern. The process and its limitations will be illustrated by example. Consider an input image $f[n,m]$ that is composed of two replicas of a real-valued nonnegative pattern $p[n,m]$ centered at coordinates $[n_1,m_1]$ and $[n_2,m_2]$ with respective amplitudes A_1 and A_2. The image also includes a bias $b \cdot 1[n,m]$:

$$f[n,m] = A_1 \cdot p[n-n_1, m-m_1] + A_2 \cdot p[n-n_2, m-m_2] + b \cdot 1[n,m]$$

Consider a kernel that is identical to the pattern but "reversed":

$$\hat{m}[n,m] = p[-n,-m]$$

which also is real valued and nonnegative within its region of support. The output from this matched filter autocorrelation of the pattern centered at those coordinates:

$$\begin{aligned} g[n,m] &= f[n,m] * \hat{m}[n,m] \\ &= A_1 \cdot p[n,m] \bigstar p[n,m]|_{n=n_1,m=m_1} + A_2 \cdot p[n,m] \bigstar p[n,m]|_{n=n_2,m=m_2} \\ &\qquad + b \cdot (1[n,m] * p[-n,-m]) \\ &= A_1 \cdot p[n,m] \bigstar p[n,m]|_{n=n_1,m=m_1} + A_2 \cdot p[n,m] \bigstar p[n,m]|_{n=n_2,m=m_2} \\ &\qquad + b \cdot \sum_{n,m} p[n,m] \end{aligned}$$

The last term is the spatially invariant output due to the constant bias convolved with the matched filter, which produces the sum of the product of the bias and the weights at each sample. The spatially varying autocorrelation functions rest on a bias proportional to the sum of the gray values p in the pattern. If the output bias is large, it can reduce the "visibility" of the autocorrelations in exactly the same way as the modulation of a nonnegative sinusoidal function. Therefore it is convenient to construct a matched filter kernel whose weights sum to zero by subtracting the average value:

$$\hat{m}[n,m] = p[-n,-m] - p_{average} \Longrightarrow \sum_{n,m} \hat{m}[-n,-m] = \sum_{n,m} \hat{m}[n,m] = 0$$

This condition ensures that the constant bias in the third term vanishes. This result determines the strategy for designing convolution kernels that produce outputs that have large magnitudes at pixels centered on neighborhoods that contain these patterns and small magnitudes in neighborhoods where the feature does not exist. For example, consider an image containing an "upper-right corner" of a brighter object on a darker background:

$f[n,m] =$

⋱	⋮	⋮	⋮	⋮	⋮	⋯
⋯	50	50	50	50	50	⋯
⋯	50	50	50	50	50	⋯
⋯	100	100	100	50	50	⋯
⋯	100	100	100	50	50	⋯
⋯	100	100	100	50	50	⋯
⋮	⋮	⋮	⋮	⋮	⋮	⋱

The task is to design a 3×3 kernel for locating this pattern in a scene:

$p[n,m] =$

50	50	50
100	100	50
100	100	50

The recipe tells us to rotate the pattern by π radians about its center to create $p[-n,-m]$:

$p[-n,-m] =$

50	100	100
50	100	100
50	50	50

The average weight in this 3×3 kernel is $\frac{650}{9} \cong 72.222$, which is subtracted from each element:

$$\begin{array}{|c|c|c|}\hline -22.222 & +27.778 & +27.778 \\ \hline -22.222 & +27.778 & +27.778 \\ \hline -22.222 & -22.222 & -22.222 \\ \hline \end{array} = (+22.222) \begin{array}{|c|c|c|}\hline -1 & +1.25 & +1.25 \\ \hline -1 & +1.25 & +1.25 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

The multiplicative factor may be ignored since it just scales the output of the convolution by this constant. Thus one realization of the unamplified 3×3 matched filter for upper-right corners is:

$\hat{m}[n,m] \cong$

−1	+1.25	+1.25
−1	+1.25	+1.25
−1	−1	−1

Though not really an issue now with faster computers, it was once considered more convenient to restrict the weights in the kernel to integer values. This may be done by redistributing the weights slightly. In this example, the fraction of the positive weights often is concentrated in the center pixel to produce the *Prewitt corner detector*:

$$\hat{m}[n,m] \cong \begin{array}{|c|c|c|} \hline -1 & +1 & +1 \\ \hline -1 & +2 & +1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

Note that that the upper-right corner detector contains a bipolar pattern that looks like a lower-left corner because of the rotation ("reversal") inherent in the convolution. Because $\hat{m}$ is bipolar, so generally is the output of the convolution with the input $f[n,m]$. The linearity of convolution ensures that the output amplitude at a pixel is proportional to the contrast of the feature. If the contrast of the upper-right corner is large and "positive," meaning that the corner is much brighter than the dark background, the output at the corner pixel will be a large and positive extremum. Conversely, a dark object on a very bright background will produce a large negative extremum. The magnitude of the image shows the locations of features with either contrast. The output image may be thresholded to specify the pixels located at the desired feature.

This method of feature detection is not ideal. The output of this unamplified filter at a corner is the autocorrelation of the feature rather than the ideal 2-D discrete Dirac delta function. If multiple copies of the pattern with different contrasts are present in the input, it will be difficult or impossible to segment the desired features by thresholding the convolution alone. Another consequence of the unamplified matched filter is that features other than the desired pattern produce nonnull outputs, as shown in the output of the corner detector applied to a test object consisting of "E" at two different amplitudes as shown. The threshold properly locates the upper-right corners of the bright "E" and one point on the sampled circle, but misses the corners of the fainter "E". This shows that corners of some objects are missed (false negatives). If the threshold were set at a lower level to detect the corner of the fainter "E", other pixels will be incorrectly identified as corners (false positives). A simple method for reducing misidentified pixels is considered in the next section.

7.12.1 Other Matching Kernels

Prewitt Edge Detectors (gradients): (horizontal and vertical)

$$\frac{1}{3}\begin{array}{|c|c|c|} \hline +1 & 0 & -1 \\ \hline +1 & 0 & -1 \\ \hline +1 & 0 & -1 \\ \hline \end{array}, \frac{1}{3}\begin{array}{|c|c|c|} \hline -1 & -1 & -1 \\ \hline 0 & 0 & 0 \\ \hline +1 & +1 & +1 \\ \hline \end{array}$$

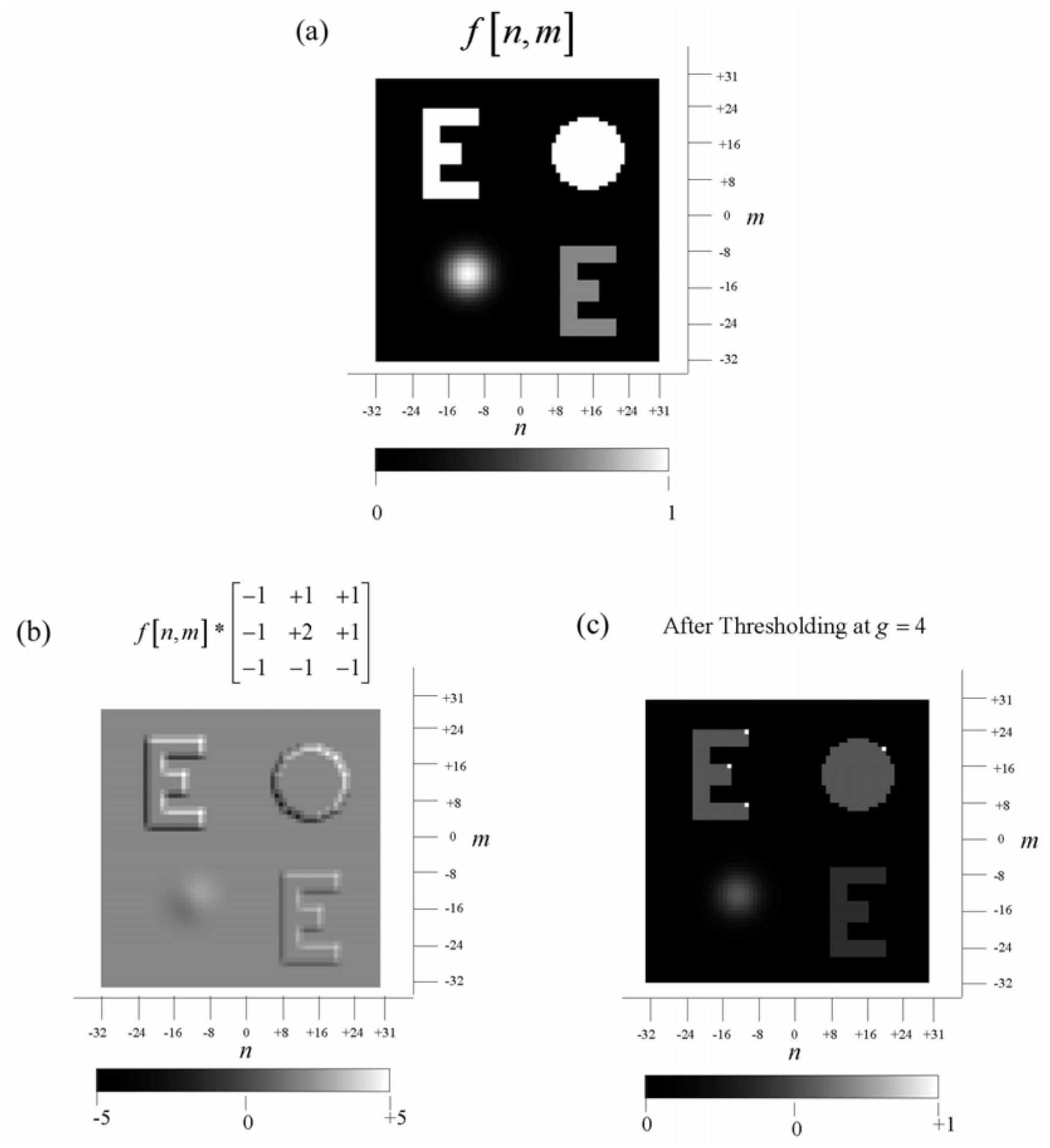


Figure 7.3: *Thresholding to locate features in the image: (a) $f[n,m]$, which is the nonnegative function with $0 \leq f \leq 1$; (b) $f[n,m]$ convolved with the "upper-right corner detector", producing the bipolar output $g[n,m]$ where $-5 \leq g \leq 4$. The largest amplitudes occur at the upper-right corners, as shown in (c) after thresholding at level 4 along with a "ghost" replica of the original image. This demonstrates detection of the upper-right corners of the high-contrast "E" and of the circle, but the corner of the low-contrast "E" is missed.*

Sobel Edge Detectors (gradients): (horizontal and vertical)

$$\frac{1}{4}\begin{array}{|c|c|c|}\hline +1 & 0 & -1 \\ \hline +2 & 0 & -2 \\ \hline +1 & 0 & -1 \\ \hline \end{array}\,,\ \frac{1}{4}\begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline 0 & 0 & 0 \\ \hline +1 & +1 & +1 \\ \hline \end{array}$$

Line Detectors: (horizontal and vertical)

$$\begin{array}{|c|c|c|}\hline -1 & +2 & -1 \\ \hline -1 & +2 & -1 \\ \hline -1 & +2 & -1 \\ \hline \end{array}\,,\ \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline +2 & +2 & +2 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

"Spot" Detectors:

$$\nabla^2 = \begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline -1 & +8 & -1 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$$

7.12.2 Normalization of Contrast of Detected Features

The recipe just developed allows creation of kernels for detecting pixels in neighborhoods that are "similar" to some desired pattern. However, the sensitivity of the process to feature contrast can significantly limit its utility. A simple modification can improve the classification. A normalized correlation measure $R[n,m]$ was defined by Hall (1979):

$$R[n,m] = \frac{f[n,m] * h[n,m]}{\sqrt{\sum_{n,m} (f[n,m])^2}\sqrt{\sum_{n,m} (h[n,m])^2}}$$

where the sums in the denominator are over ONLY the values of the input and kernel within the support of the latter. Note that the sum of the squares of the elements of the kernel $h[n,m]$ results in a constant scale factor k and may be ignored:

$$R[n,m] = k\left(\frac{f[n,m]}{\sqrt{\sum_{n,m} (f[n,m])^2}}\right) * h[n,m]$$

In words, this operation divides the convolution by the geometric or Pythagorean sum of gray levels under the kernel. The modification to the filter makes the entire

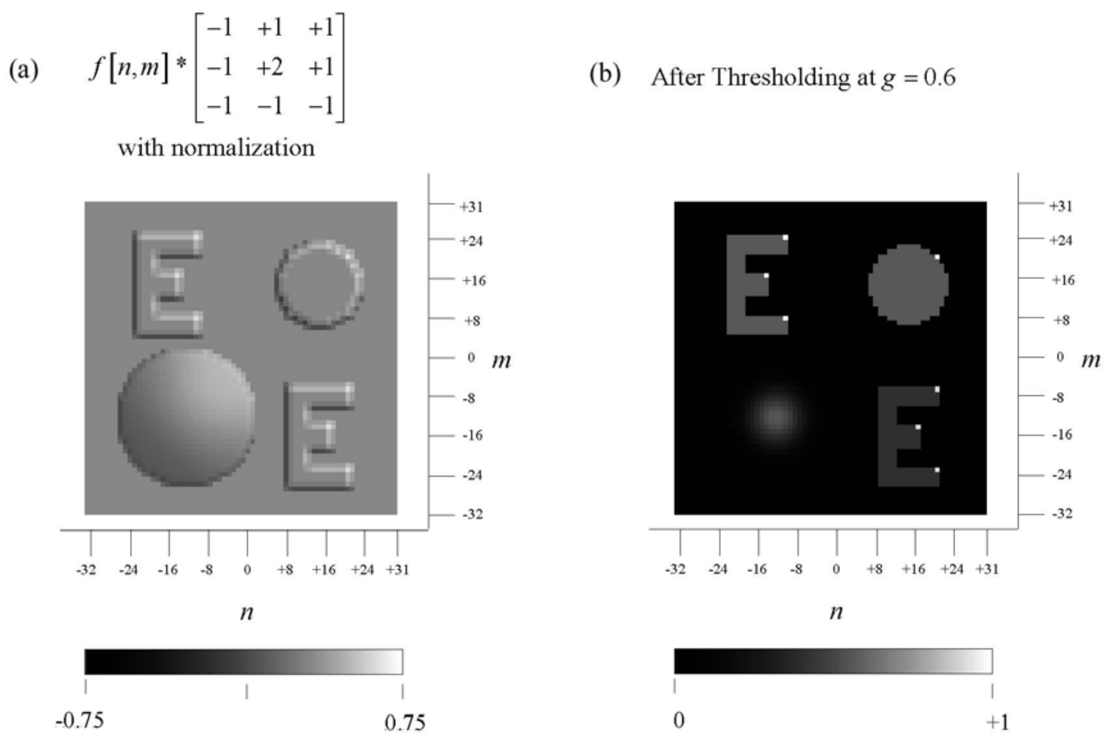


Figure 7.4: *The action of the nonlinear normalization of detected features using the same object: (a) $f[n,m]$ convolved with the upper-right corner detector with normalization by the image amplitude, producing the bipolar output $g[n,m]$ where $-0.60 \leq g \leq +0.75$; (b) image after thresholding at $+0.6$, with a "ghost" of the original image, showing the detection of the upper-right corners of both "E"s despite the different image contrasts.*

process shift variant and thus may not be performed by a simple convolution. The denominator may be computed by convolving $(f[n,m])^2$ with a uniform averaging kernel $s[n,m]$ of the same size as the original kernel $h[n,m]$ and then evaluating the square root

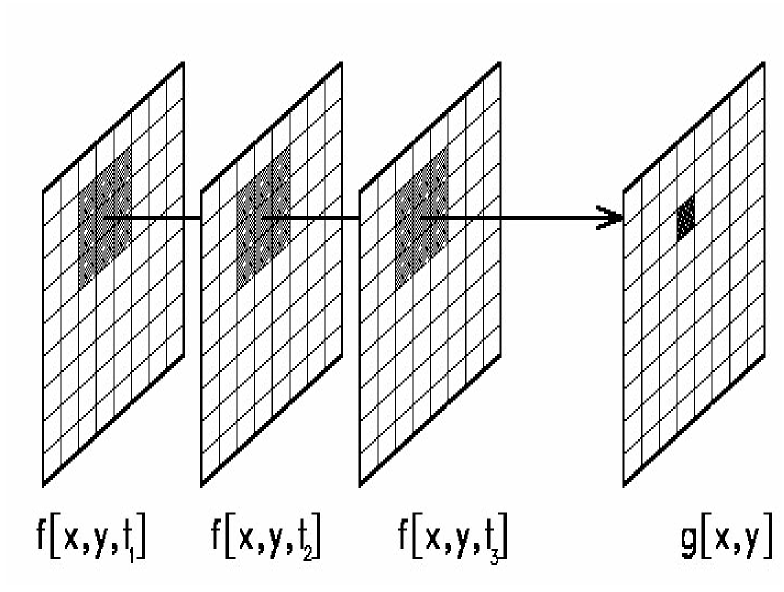
$$R[n,m] = k \frac{f[n,m] * h[n,m]}{\sqrt{(f[n,m])^2 * s[n,m]}}$$

The upper-right corner detector with normalization is shown in the figure, where the features of both "E"s are located with a single threshold.

7.13 Implementation of Filtering

7.13.1 Nonlinear and Shift-Variant Filtering

Special-purpose hardware (array processors) is readily available for implementing linear operations. These contain several memory planes and can store multiple copies



of the input. By shifting the addresses of the pixels in a plane and multiplying the values by a constant, the appropriate shifted and weighted image can be generated. These are then summed to obtain the filtered output. A complete convolution can be computed in a few clock cycles.

Other than the mean filter, the statistical filters are nonlinear, i.e., the gray value of the output pixel is obtained from those of the input pixel by some method other than multiplication by weights and summing. In other words, they are not convolutions and thus cannot be specified by a kernel. Similarly, operations that may be linear (output is a sum of weighted inputs) may use different weights at different locations in the image. Such operations are shift-variant and must be specified by different kernels at different locations. Nonlinear and shift-variant operations are computationally intensive and thus slower to perform unless special single-purpose hardware is used. However, as computer speeds increase and as prices fall, this is becoming less of a problem. Because of the flexibility of operations possible, nonlinear shift-variant filtering is a very active research area.

7.14 Neighborhood Operations on Multiple Images

7.14.1 Image Sequence Processing

It should now be obvious that we can combine the gray levels in neighborhoods of the input pixel in multiple images to obtain the output image $g[x, y]$. The multiple copies of $f[x, y]$may have been spread over time (e.g. video), over wavelength (e.g. RGB images), or some other parameter. In these cases, the image and kernel are functions of three coordinates.

7.14.2 Spectral + Spatial Neighborhood Operations

Additive noise is a common corrupter of digital images and may disrupt classification algorithms based on gray-level differences, *e.g.* in multispectral differencing to segment remote-sensing images. The noise can be attenuated by combining spatial averaging and spectral differencing, i.e.

$$g[x,y] = f[x,y,\lambda_1] * h[x,y] - f[x,y,\lambda_2] * h[x,y]$$

where

$$h[x,y] = \frac{1}{9}\begin{array}{|c|c|c|}\hline +1 & +1 & +1 \\ \hline +1 & +1 & +1 \\ \hline +1 & +1 & +1 \\ \hline \end{array}$$

. Since convolution and subtraction are linear, the order of operations can be interchanged:

$$g[x,y] = (f[x,y,\lambda_1] - f[x,y,\lambda_2]) * h[x,y]$$

These can be combined into a single 3-D operation using a 3-D kernel $h[x,y,\lambda]$

$$g[x,y] = f[x,y,\lambda_i] * h[x,y,\lambda_i]$$

where

$$h[x,y,\lambda_i] = -\frac{1}{9}\begin{array}{|c|c|c|}\hline +1 & +1 & +1 \\ \hline +1 & +1 & +1 \\ \hline +1 & +1 & +1 \\ \hline \end{array}, \frac{1}{9}\begin{array}{|c|c|c|}\hline +1 & +1 & +1 \\ \hline +1 & +1 & +1 \\ \hline +1 & +1 & +1 \\ \hline \end{array}, \frac{1}{9}\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

7.14.3 "Pseudounsharp Masking"

A very different variant of unsharp masking was developed by a group of astronomers, who needed to examine fine structure with low contrast (i.e. features distinguished by small brightness differences over short distance scales) that are hidden by a larger-scale brightness gradient across of the object or scene. Included among the significant low-contrast structural features are streamers in the solar corona that radiate outward from the solar surface. The details in the structure provide clues about the physical nature of the solar atmosphere and magnetic fields. However, the radial brightness gradient of the corona makes it very difficult to image the full length of the coronal streamers. The overall dynamic range of ground-based imagery of the solar corona is approximately three orders of magnitude (limited by atmospheric sky brightness). Imagery from air- or spacecraft may add an order of magnitude for a total dynamic range of 10,000. This may be recorded on a wide-range photographic negative ($\Delta D > 4$), but it is not possible to print that dynamic range by normal photographic techniques.

The problem of enhancing the visibility of small changes in coronal image brightness is similar to correction of detector sensitivity just considered and may be attacked by digital methods. The brightness gradient of the corona is analogous to the 1-D sensitivity function $s[x]$; division of the original image by the brightness gradient equalizes the background so that the variations across the image may be displayed on a medium with limited dynamic range. An estimate of the coronal brightness gradient may be estimated by averaging fitted curves of the radial brightness or by making a low-resolution (blurred) image. The latter is more commonly used, as it may be derived via simple local neighborhood digital operations to be considered next. The recovered image is the ratio of the measured high-resolution image and the image of the brightness gradient. This technique may be applied to archival photographic negatives of historical eclipses to provide additional information about the history of solar conditions and thus their effects on the earth's climate.

The process may be viewed in terms of the convolution operators. We start with the original image $r[x,y]$ which includes the coarse brightness gradient function $s[x,y]$ overlying the desired image $f[x,y]$ of the fine structure:

$$r[x,y] = f[x,y] \cdot s[x,y]$$

We can estimate $s[x,y]$ by blurring the image $r[x,y]$ sufficiently so that the fine details are all blurred together:

$$r[x,y] * h[x.y] \cong s[x,y]$$

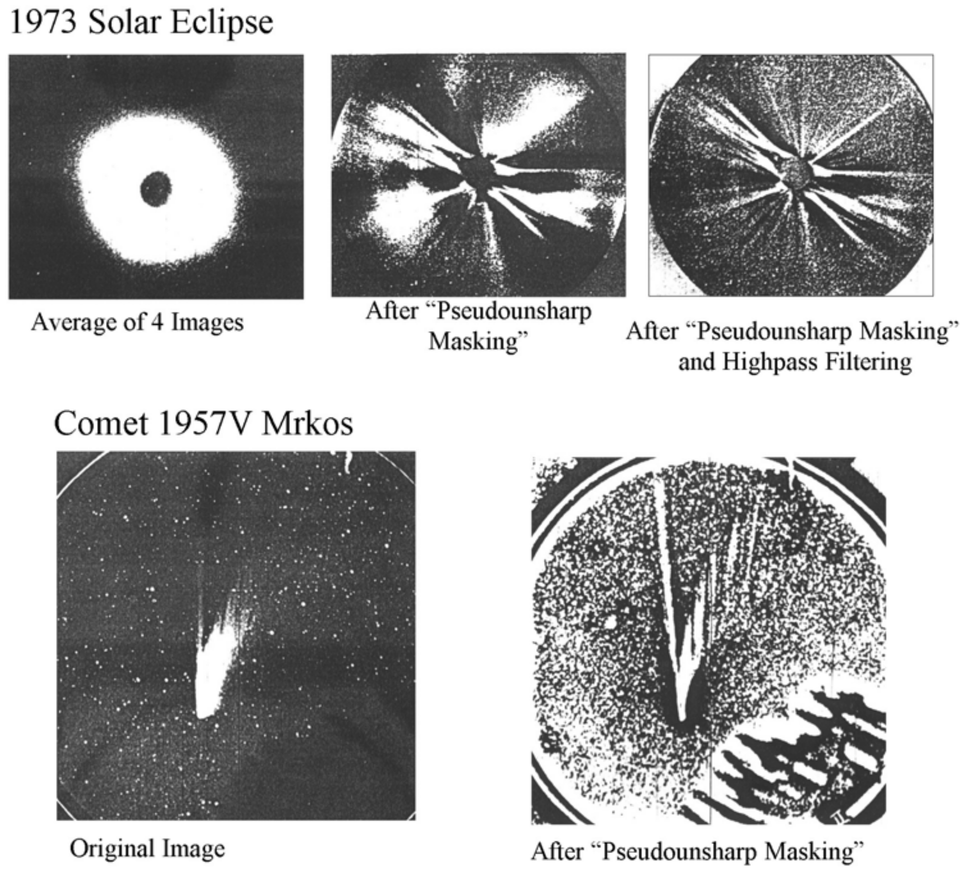
where $h[x,y]$ is a "large" averaging kernel, perhaps as large as 101×101 pixels or so. The output image is:

$$\frac{r[x,y]}{r[x,y] * h[x.y]} \cong \frac{f[x,y] \cdot s[x,y]}{s[x,y]} \cong f[x,y]$$

where the ratio image must be scaled to the available dynamic range of the display.

Examples of Image Division for Local Contrast Enhancement

Examples of recovered information from old imagery of a solar eclipse and a comet are shown below. The pictures are from *Enhancement of Solar Corona and Comet Details* by Matuska, *et al.*, **Proc. SPIE, 119**, pp. 28-35, 1977 and in **Optical Engineering, 17**(6), 661-665, 1978. The images are scans of xerographic prints from microfilm, which is the reason for the poor quality.


1973 Solar Eclipse
Average of 4 Images
After "Pseudounsharp Masking"
After "Pseudounsharp Masking" and Highpass Filtering
Comet 1957V Mrkos
Original Image
After "Pseudounsharp Masking"

Chapter 8

Global Operations

If a pixel in the output image g is a function of (almost) all of the pixels in $f[x,y]$, then $\mathcal{O}\{f[x,y]\}$ is a global operator. This category includes image coordinate transformations, of which the most important is the Fourier transform. These transformations derive new, usually equivalent, representations of images; for example, the Fourier transform maps from the familiar coordinate-space representation $f[x,y]$ to a new representation (a new image) whose brightness at each coordinate describes the quantity of a particular sinusoidal spatial frequency component present in $f[x,y]$. The sum of the component sinusoids is the original image. In other words, the Fourier transform generates the frequency-space representation $F[\xi,\eta]$ of the image $f[x,y]$. The coordinates of the image [x,y] have dimensions of length (e.g., mm) while the coordinates of the frequency representation $[\xi,\eta]$ have units of inverse length (e.g., $\frac{\text{cycles}}{\text{mm}}$). Global gray-level properties of

the image map to local properties in the Fourier transform, and vice versa. The frequency-space representation is useful for many applications, including segmentation, coding, noise removal, and feature classification. It also provides an avenue for performing other image operations, particularly convolution. Each output pixel is a function of the gray levels of all input pixels.

8.1 Relationship to Neighborhood Operations

The concept of a linear global operator is a simple extension of that of the linear local neighborhood operator. In that case, an output pixel was calculated by point-by-point multiplication of pixels in the input image by a set of weights (the kernel) and summing the products. The convolution at different pixels is computed by shifting the kernel. Recall that some accommodation must be made for cases where one or more elements of the the kernel are off the edge of the image.

In the case of a global operator, the set of weights is as large as the image and constitutes a "mask function", say $q[x,y]$. The output value obtained by applying a mask $q[x,y]$ to an input image $f[x,y]$ is:

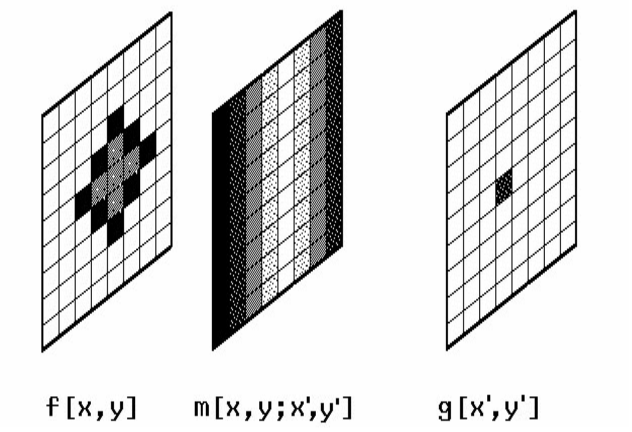
$$g = \iint f[x,y]\ \ q[x,y]\ \ dx\ dy$$

In the discrete case, the integral becomes a summation:

$$g = \sum_{n} \sum_{n} f[n,m] \;\; q[n,m]$$

Note that a translation of the mask by one pixel in any direction shifts some of its elements over the edge of the image. If we assume that the output in such cases is undefined, only a single output pixel is calculated from one mask function $q[n,m]$. In general, different outputs result from different masks, i.e., we can define an output pixel by using different masks for each coordinate pair [x',y']:

$$g[k,\ell] = \sum_{n} \sum_{m} f[n,m] \;\; q[n,m;k,\ell]$$



In general, the coordinates of g are different from those of f, and often even have different dimensions (units). The action of the operator is obviously determined by the form of the mask function. The most common example is the Fourier transform, where the mask function is:

$$q[x,y;\xi,\eta] = \cos[2\pi(\xi x + \eta y)] - i \;\; \sin[2\pi(\xi x + \eta y)] = \exp[-2\pi i(\xi x + \eta y)]$$

8.2 Discrete Fourier Transform (DFT)

If the input signal has been sampled at discrete intervals (of width Δx, for example), the Fourier integral over x reduces to a sum:

$$F[\xi] = \sum_{n=-\infty}^{+\infty} f[n \cdot \Delta x] \exp[-2\pi i \xi (n \cdot \Delta x)]$$

Recall that the Whittaker-Shannon sampling theorem states that a sinusoidal function must be sampled at a rate greater than than two samples per period (Nyquist frequency) to avoid aliasing. Thus, the minimum period X_{min} of a sampled sinusoidal function is two sample intervals ($2 \cdot \Delta x$ in the example above), which implies that the maximum spatial frequency in the sampled signal is:

$$\xi_{max} = \xi_{Nyq} = \frac{1}{X_{\min}} = \frac{1}{2 \cdot \Delta x}$$

$\xi_{\max}$ is measured in cycles per unit length (typically cycles per millimeter). Often the absolute scale of the digital image is not important, and the frequency is scaled to $\Delta x = 1$ pixel, i.e., the maximum spatial frequency is $\frac{1}{2}\,\frac{\text{cycle}}{\text{pixel}}$. The range of meaningful spatial frequencies of the DFT is $\frac{1}{2 \cdot \Delta x} > |\xi|$.

If the input function $f[x]$ is limited to N samples, the DFT becomes a finite sum:

$$F[\xi] \equiv \sum_{n=0}^{N-1} f[n \cdot \Delta x] \exp[-2\pi i \xi (n \cdot \Delta x)]$$
$$\text{or} \sum_{n=-\frac{N}{2}-1}^{\frac{N}{2}} f[n \cdot \Delta x] \exp[-2\pi i \xi (n \cdot \Delta x)]$$

The DFT of a 1-D sequence of N samples at regular intervals Δx can be computed at any spatial frequency ξ. However, it is usual to calculate the DFT of a sequence of frequencies (e.g., a total M) separated by a constant interval $\Delta\xi$. Each sample of the DFT of a real sequence of N pixels requires that N values each of the cosine and sine be computed, followed by $2N$ multiplications and $2N$ sums, i.e., of the order of N operations. The DFT at M spatial frequencies requires of the order of $M \cdot N$ operations. Often, the DFT is computed at N frequencies, thus requiring of the order of N^2 operations. This intensity of computation made calculation of the DFT a tedious and rarely performed task before digital computers. For example, a Fourier deconvolution of seismic traces for petroleum exploration was performed by Enders Robinson in 1951; it took the whole summer to do 32 traces by hand with a memoryless mechanical calculator. This task could now be done with the cheapest PC in less than a second. Even with mainframe digital computers into the 1960s, digital Fourier analysis was unusual because of the computation time. In 1965, J.W. Cooley and J.W. Tukey developed the Fast Fourier Transform algorithm, which substantially cut computation times and made digital Fourier analysis feasible.

8.3 Fast Fourier Transform (FFT)

The FFT was developed to compute discrete Fourier spectra with fewer operations than the DFT by sacrificing some flexibility. The DFT may compute the amplitude of sinusoidal components at any frequency within the Nyquist window, i.e., the DFT maps discrete coordinates $n \cdot \Delta x$ to a continuous set of frequencies ξ in the interval $[-\xi_{Nyq}, \xi_{Nyq}]$. The DFT may be computed at a single spatial frequency if desired. The FFT is a recursive algorithm that calculates the spectrum at a fixed discrete set of frequencies with a minimum number of repetitive calculations. The spectrum must be computed at all frequencies to obtain the values of individual spectral components. In the FFT, the amplitudes at N discrete equally spaced frequencies are computed in the interval $[-\xi_{Nyq}, \xi_{Nyq}]$ from N input samples. The frequency samples are indexed by the integer k and the interval between frequency samples is:

$$\begin{aligned}
\Delta\xi = \frac{1}{N}\cdot 2\xi_{Nyq} &= \frac{1}{N}\cdot\frac{2}{2\cdot\Delta x} = \frac{1}{N\cdot\Delta x}\\
\implies \xi_k = k\cdot\Delta\xi &= \frac{k}{N\cdot\Delta x}, \left[-\frac{N}{2}\leq k\leq\frac{N}{2}-1\right]\\
\implies N\cdot\Delta x\cdot\Delta\xi &= 1
\end{aligned}$$

If we substitute these specific frequencies into the DFT:

$$\begin{aligned}
F[k\cdot\Delta\xi] &= \sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1} f[n\cdot\Delta x]\exp[-2\pi ik\cdot\Delta\xi\cdot(n\cdot\Delta x)]\\
&= \sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1} f[n\cdot\Delta x]\exp\left[-2\pi ikn\cdot\frac{\Delta x}{N\cdot\Delta x}\right]
\end{aligned}$$

$$\text{but } \Delta\xi = \frac{1}{N\cdot\Delta x} \implies F[k\cdot\Delta\xi] = \sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1} f[n\cdot\Delta x]\exp\left[-\frac{2\pi ikn}{N}\right]$$

If Δx is assumed to be a dimensionless sample, then the Nyquist frequency is fixed at:

$$\xi_{Nyquist} = \frac{1}{2}\frac{\text{cycle}}{\text{sample}} = \pi\frac{\text{radians}}{\text{sample}}$$

Recall that the DFT assumes that the sample interval is Δx and computes a periodic spectrum with period $\frac{1}{\Delta x}$. In the FFT, the spectrum is assumed to be sampled at intervals $\Delta\xi = \frac{1}{N\cdot\Delta x}$, which implies in turn that the input function is periodic with period $N\cdot\Delta x$. If N is a power of 2 (e.g., 128, 256, 512, $\cdots$), there are only N distinct values of the complex exponential $\exp\left[-\frac{2\pi ink}{N}\right]$ to calculate. By using this fact, the number of required operations may be reduced and processing speeded up. The FFT of N samples requires of the order $N\cdot\log_2[N]$ operations vs. $\mathcal{O}\{N^2\}$ for the DFT.

Since both representations $f[n\cdot\Delta x]$ and $F[k\cdot\Delta\xi]$ are sampled and periodic, the inverse FFT is a finite summation and is proportional to:

$$f[n\cdot\Delta x] = C\cdot\sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} F[k\cdot\Delta\xi]\exp\left[+\frac{2\pi ink}{N}\right]$$

The proportionality constant C is required to ensure that $\mathcal{F}\{F[k]\} = f[n]$, and may be found by substituting the formula for the forward FFT for $F[k\cdot\Delta\xi]$:

$$f\left[n\cdot\Delta x\right]=C\cdot|\Delta x|\sum_{k=-\frac{N}{2}}^{+\frac{N}{2}-1}\left(\sum_{p=-\frac{N}{2}}^{+\frac{N}{2}-1}f[p\cdot\Delta x]\exp\left[-\frac{2\pi ipk}{N}\right]\right)\exp\left[+\frac{2\pi ink}{N}\right]$$

$$\begin{aligned}f\left[n\cdot\Delta x\right]&=C\cdot|\Delta x|\sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1}\left(\sum_{p=-\frac{N}{2}}^{\frac{N}{2}-1}f\left[p\cdot\Delta x\right]\exp\left[-\frac{2\pi ipk}{N}\right]\right)\exp\left[+\frac{2\pi ink}{N}\right]\\&=C\cdot\sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1}\sum_{p=-\frac{N}{2}}^{\frac{N}{2}-1}f\left[p\cdot\Delta x\right]\exp\left[-\frac{2\pi ik}{N}(n-p)\right]\\&=C\cdot\sum_{p=-\frac{N}{2}}^{\frac{N}{2}-1}f\left[p\cdot\Delta x\right]\sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1}\exp\left[-\frac{2\pi ik}{N}(n-p)\right]\end{aligned}$$

$$\sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1}\exp\left[-\frac{2\pi ik}{N}(n-p)\right]=\begin{cases}N\text{ if }n=p\\0\text{ if }n\neq p\end{cases}\equiv\delta\left[n-p\right]$$

$$\begin{aligned}C\cdot\sum_{p=-\frac{N}{2}}^{\frac{N}{2}-1}f\left[p\cdot\Delta x\right]\cdot\left(N\cdot\delta\left[n-p\right]\right)&=C\cdot N\cdot f\left[n\cdot\Delta x\right]\\&=f\left[n\cdot\Delta x\right]\text{ if }C=N^{-1}\end{aligned}$$

Thus $C=N^{-1}$ and the inverse FFT may be defined as:

$$f\left[n\cdot\Delta x\right]=\frac{1}{N}\sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1}F\left[k\cdot\Delta\xi\right]\exp\left[+\frac{2\pi ink}{N}\right]$$

The proportionality constant is a scale factor that is only significant when cascading forward and inverse transforms and may be applied in either direction. Many conventions (including mine) include the proportionality constant in the forward FFT:

$$\begin{aligned}
F[k\cdot\Delta\xi] = F\left[\frac{k}{N\cdot\Delta x}\right] &= \frac{1}{N}\sum_{n=-\frac{N}{2}}^{\frac{N}{2}-1} f[n\cdot\Delta x]\exp\left[-\frac{2\pi ink}{N}\right] \\
f[n\cdot\Delta x] &= \sum_{k=-\frac{N}{2}}^{\frac{N}{2}-1} F[k\cdot\Delta\xi]\exp\left[+\frac{2\pi ink}{N}\right] \\
N\cdot\Delta x\cdot\Delta\xi &= 1 \\
\xi_{Nyq} &= N\cdot\frac{\Delta\xi}{2} \\
x_{\max} &= N\cdot\frac{\Delta x}{2}
\end{aligned}$$

8.4 Fourier Transforms of Images

The concept of a 1-D Fourier transform can be easily extended to multidimensional continuous or discrete signals. The continuous 2-D transform is defined as:

$$\mathcal{F}_2\{f[x,y]\} \equiv F[\xi,\eta] = \iint_{-\infty}^{+\infty} f[x,y]\exp[-2\pi i(\xi x+\eta y)]\,dxdy$$

For a uniformly sampled discrete 2-D function $f[\ell,k]$, the transform is a summation:

$$\mathcal{F}_2\{f[n\cdot\Delta x, m\cdot\Delta y]\} = \sum_{n=-\infty}^{+\infty}\sum_{m=-\infty}^{+\infty} f[n,m]\exp[-2\pi i(\xi n\cdot\Delta x+\eta m\cdot\Delta y)]$$

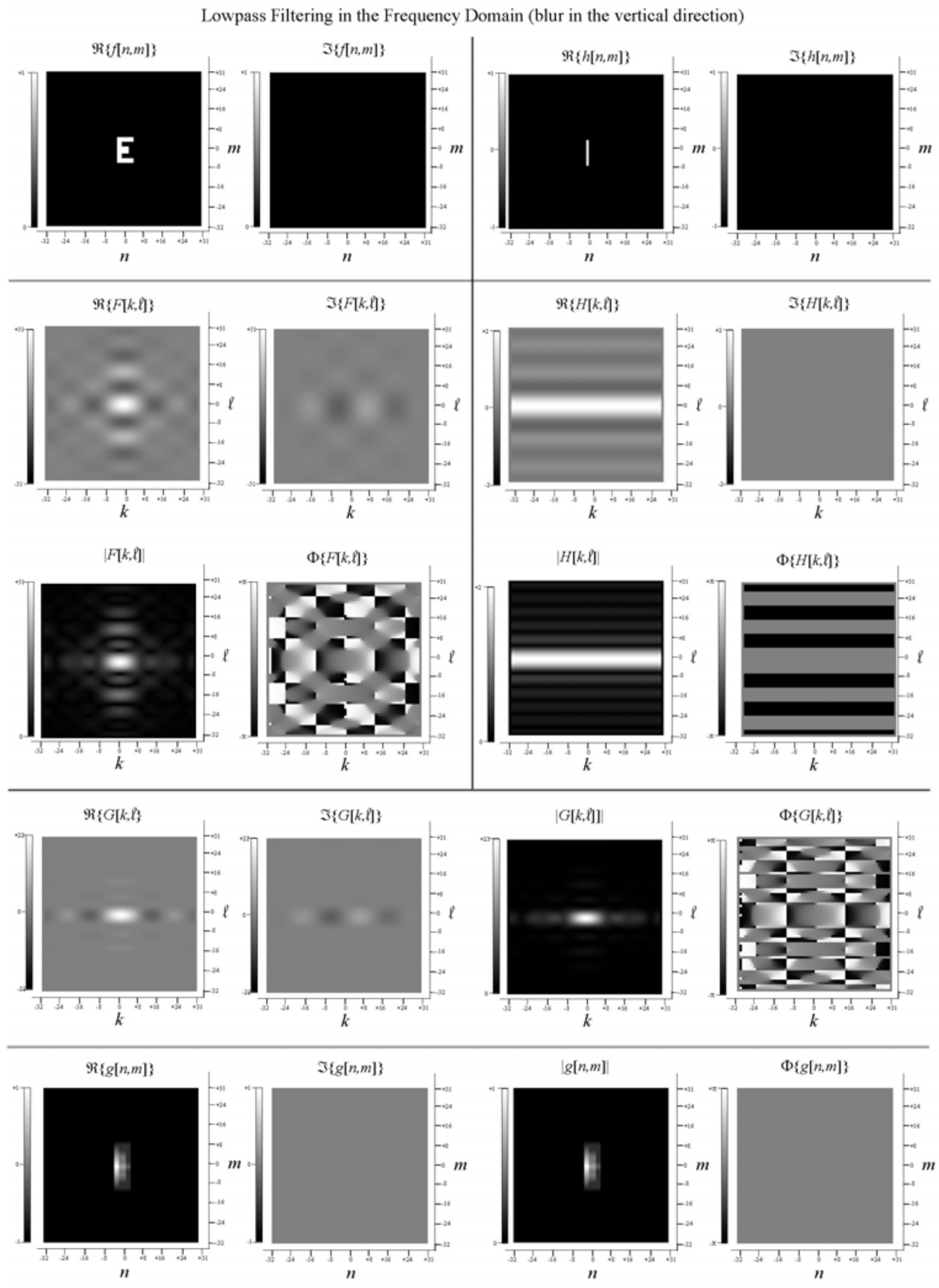
The Fourier transform of a real-valued 2-D function is Hermitian (even real part and odd imaginary part. The Fourier transform of the image of a 2-D cosine is a pair of delta-function spikes at a distance from the origin proportional to the frequency of the cosine, as shown on the next page. The polar angle of the spikes relative to the origin indicates the direction of variation of the cosine, while the brightness of the spikes is proportional to the amplitude of the cosine. Notice that the Fourier transform of the sum of two cosine waves is the sum of the individual transforms (i.e., the Fourier transform is linear).

The 2-D transform has the same properties mentioned before, including that global properties become local properties and vice versa. This is the primary reason why the Fourier transform is such a powerful tool for image processing and pattern recognition; $F[\xi,\eta]$ is uniquely defined for each $f[x,y]$, and the global properties of $f[x,y]$ are concentrated as local properties of $F[\xi,\eta]$.

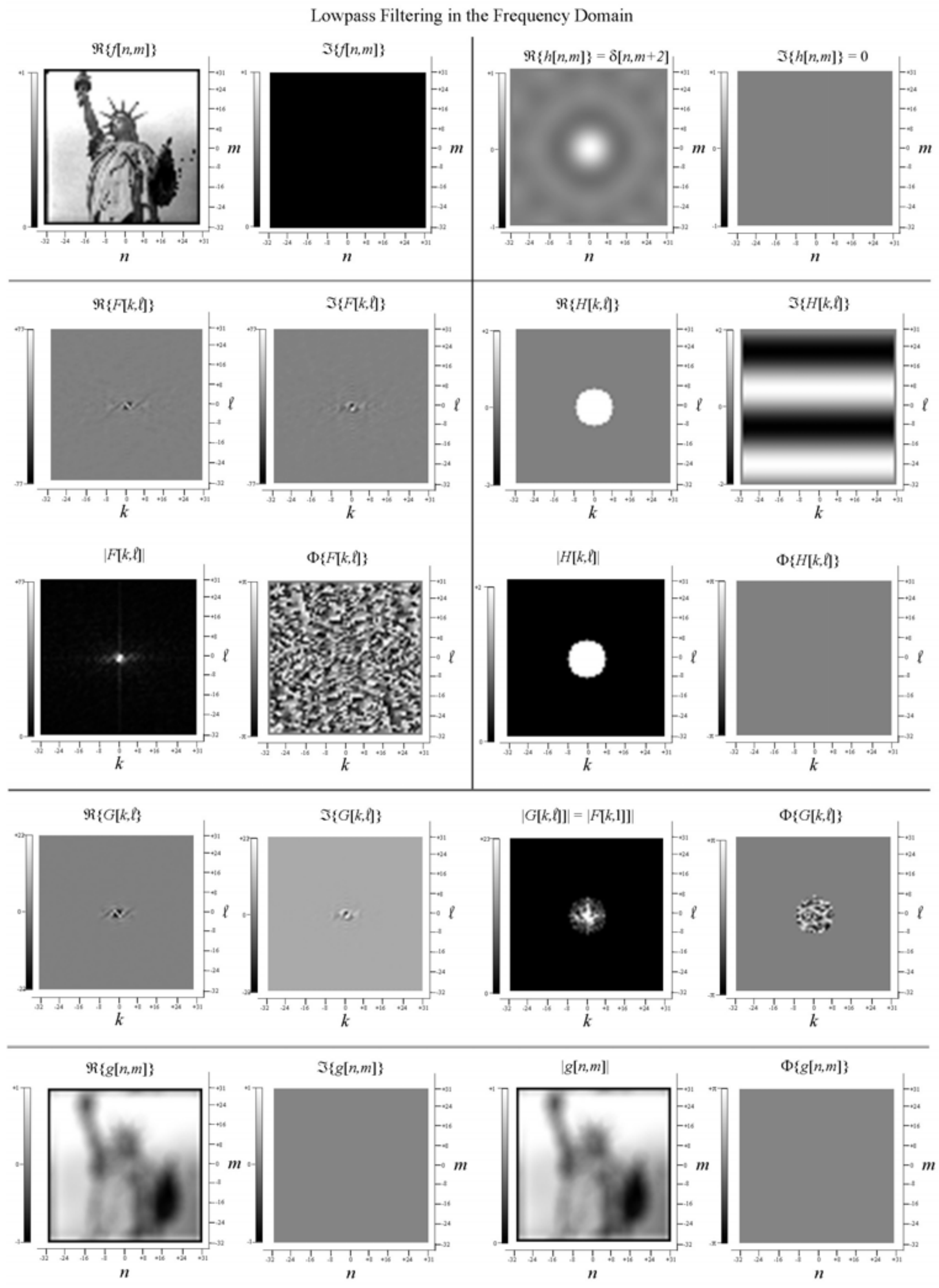
Local modification of $F[\xi,\eta]$ is called filtering and is intimately related to the local operation of convolution that we've already discussed. In fact, it is easily proven that the Fourier transform of a convolution is the product of the Fourier transforms of the component functions. This result is called the Filter Theorem:

$$\begin{aligned}\mathcal{F}_2\{f[x,y]*h[x,y]\} &= \mathcal{F}_2\{f[x,y]\}\cdot\mathcal{F}_2\{h[x,y]\}\\ &\Longrightarrow f*h=\mathcal{F}_2^{-1}\{F[\xi,\eta]\cdot H[\xi,\eta]\}\end{aligned}$$

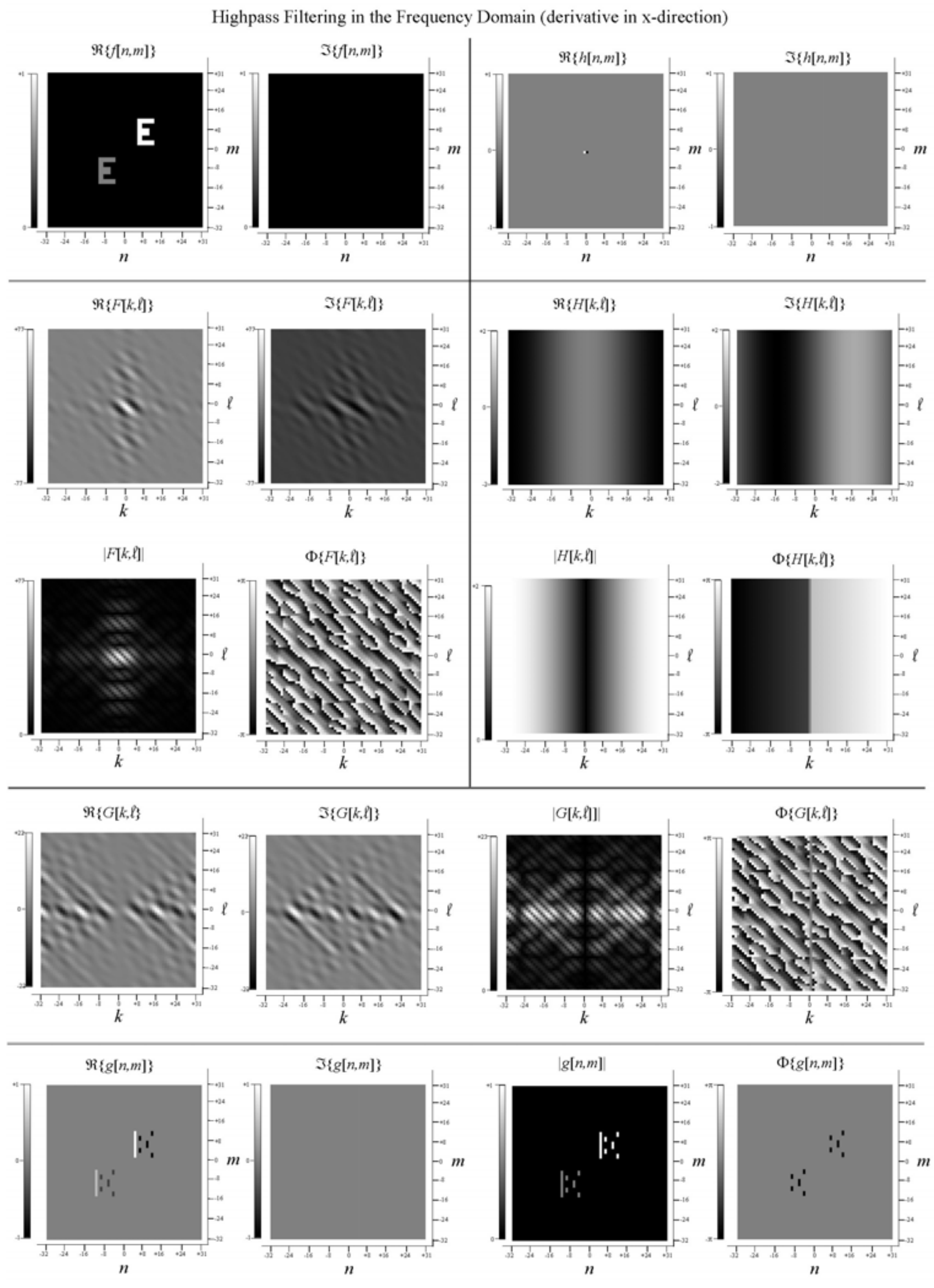
We have already given the name impulse response or point spread function to $h[x,y]$; the representation $H[\xi,\eta]$ is called the transfer function of the system.The most common reason for computing Fourier transforms of digital signals or images is the to use this path for convolution.



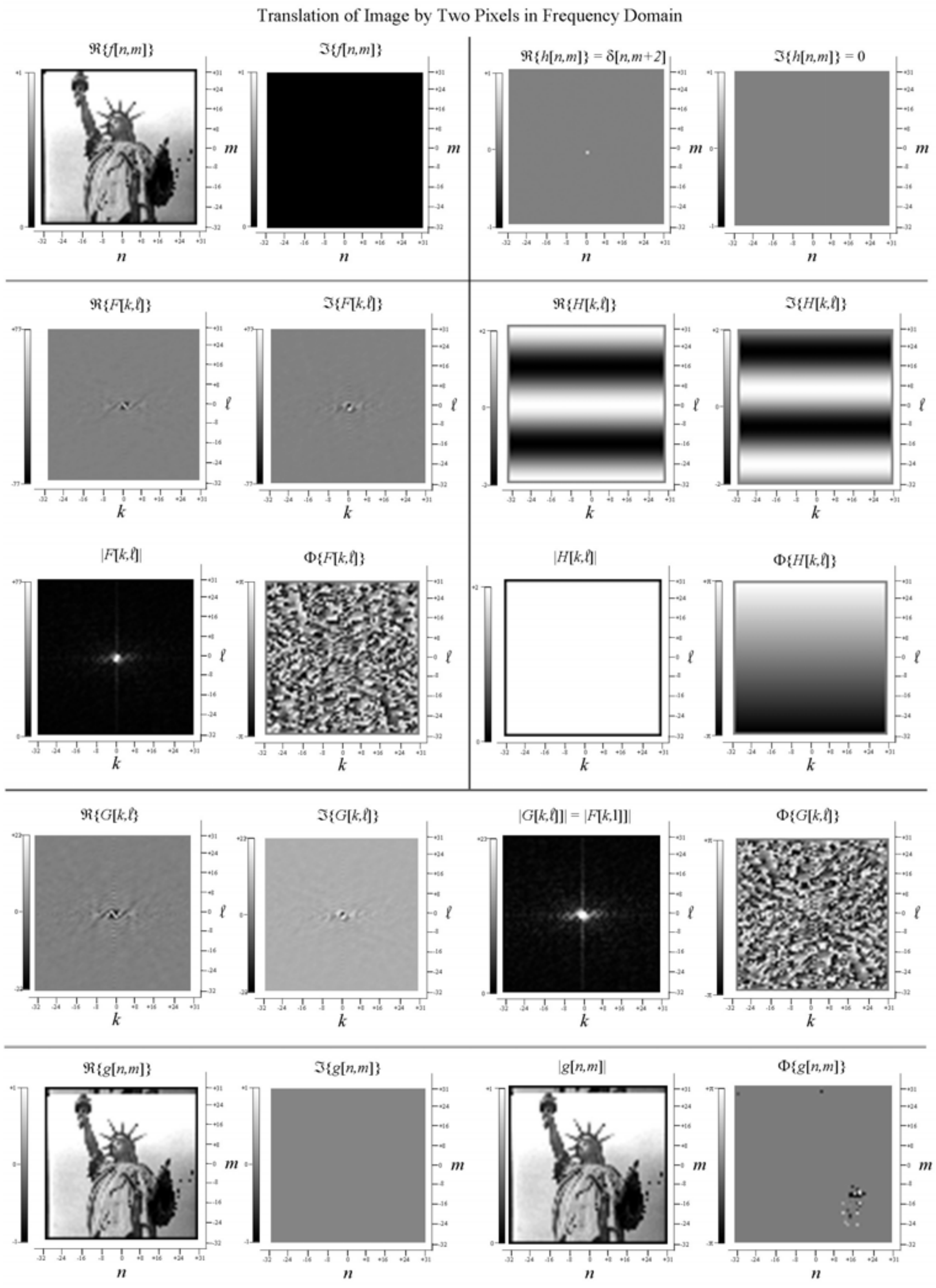
Local averaging of a 2-D image along a vertical direction in the frequency domain. The top row shows the real and imaginary parts of the object (a letter "E") and the kernel (a vertical averager). The next two rows show the Fourier transforms as real, imaginary, magnitude, and phase. The next row shows the products (real, imaginary, magnitude, phase), and the final row shows the output image as real, imaginary, magnitude, and phase.



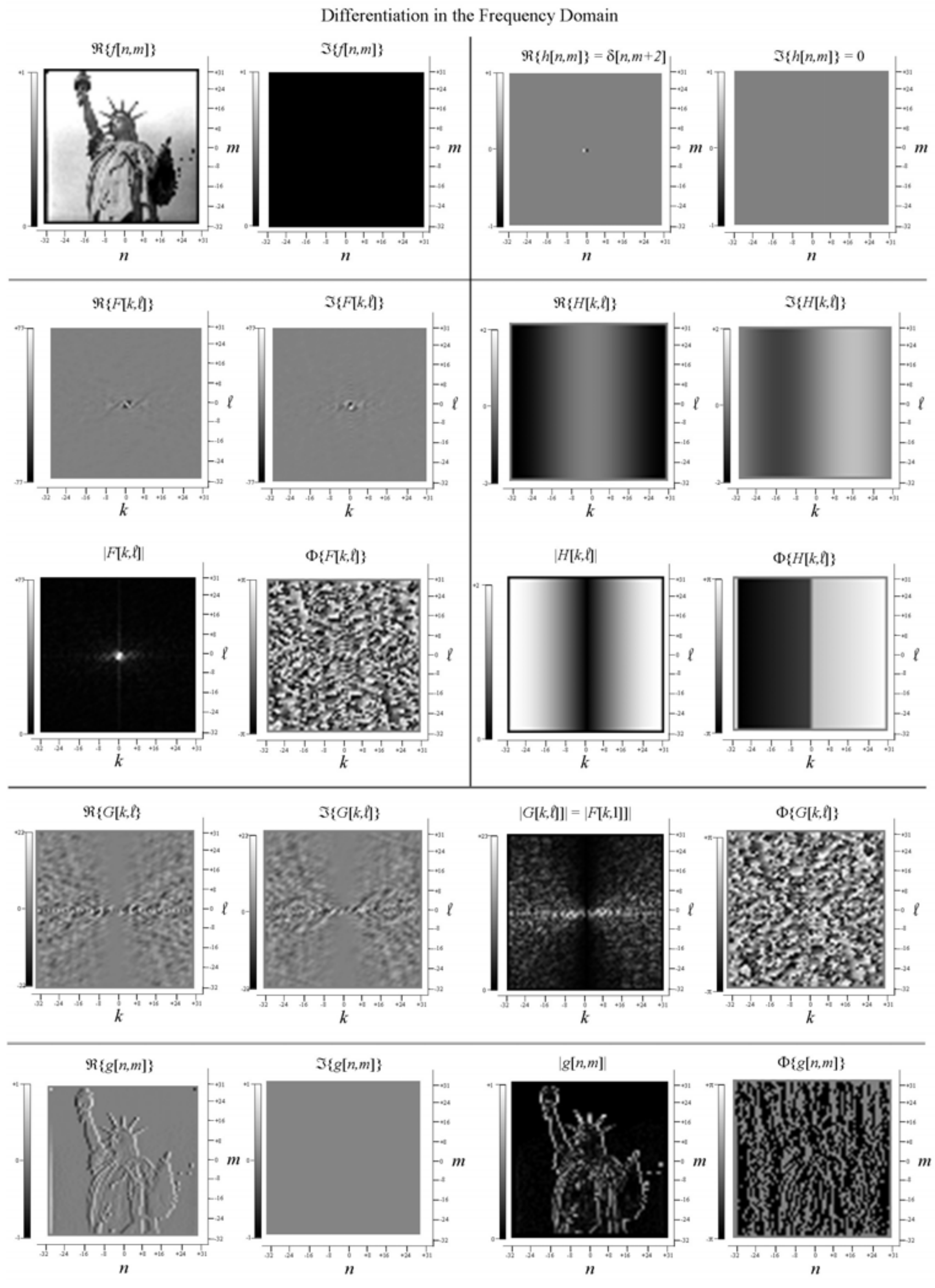
Lowpass filtering of "Liberty" in the frequency domain. In this case, sinusoidal frequencies outside of the circle are blocked completely. Since these frequencies that are necessary to reproduce edges are missing, the image is very blurry.



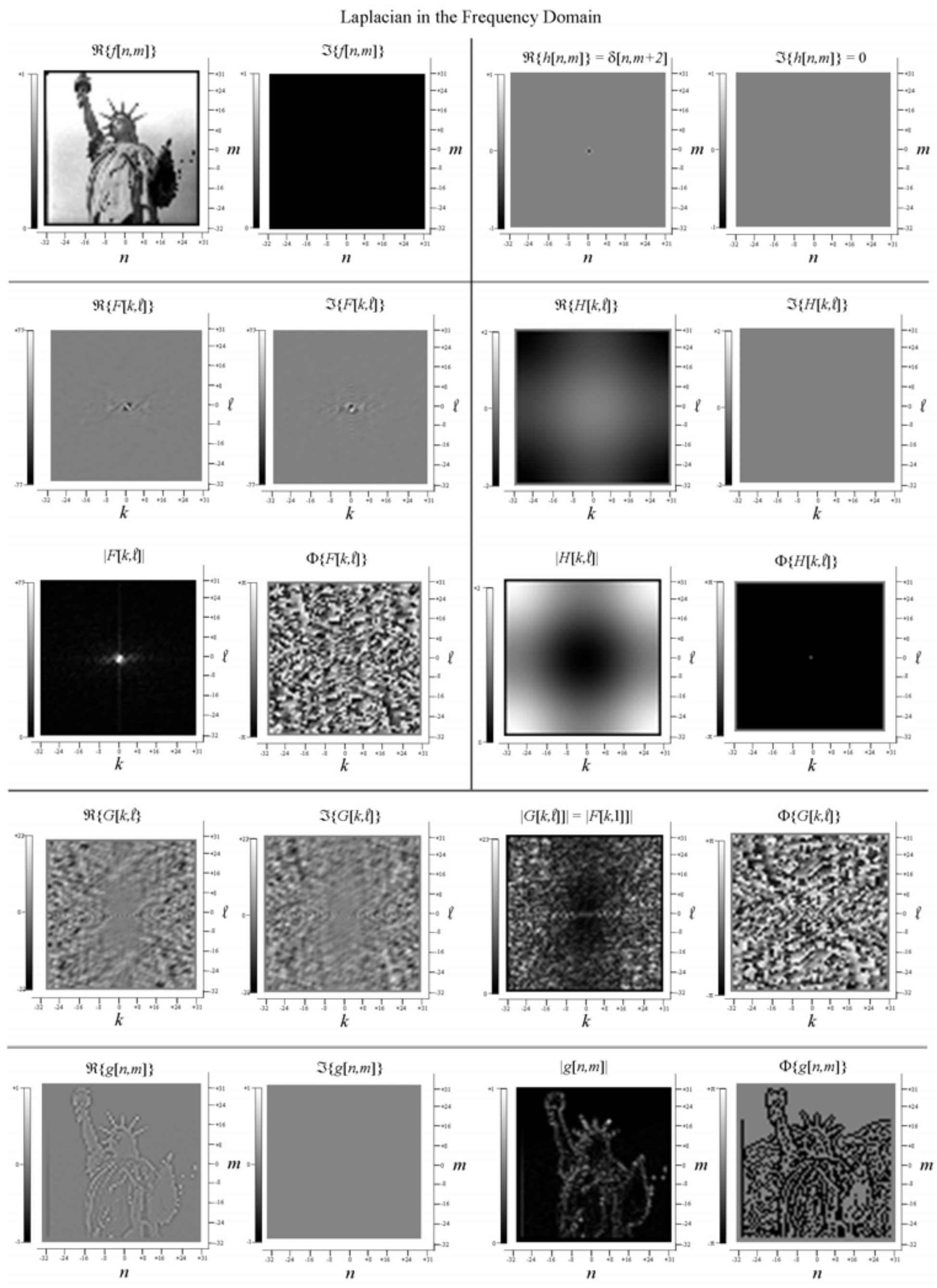
Derivative with respect to x of two "E" characters implemented in the frequency domain. This is a local differencer, and therefore a highpass filter, that amplifies the high-frequency terms in the spectrum of the object.



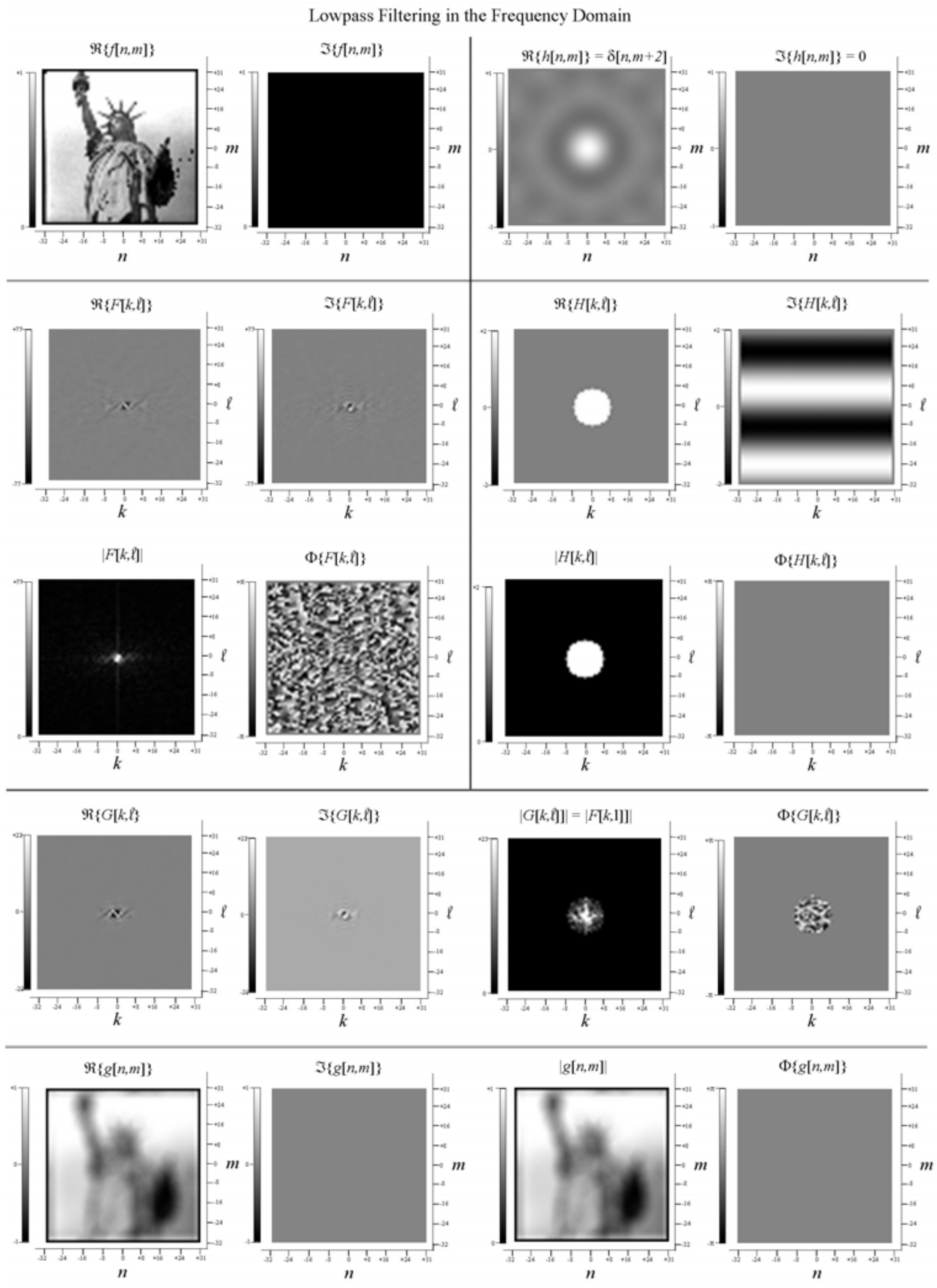
Translation of the image by two pixels in the vertical direction by actions in the frequency domain. The filter passes all spatial frequency components with change in magnitude, but it does change the phase by increments proportional to the spatial frequency.



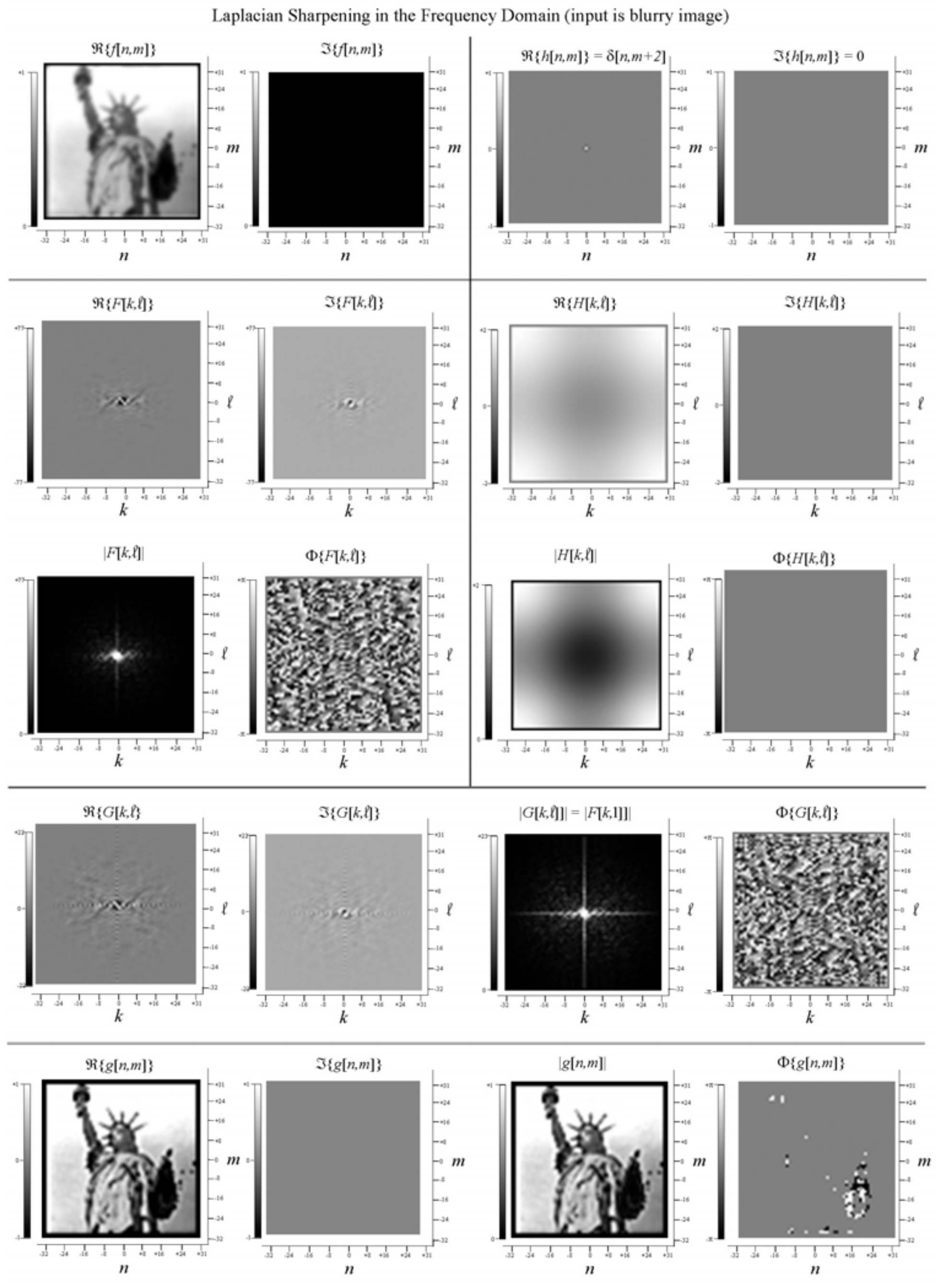
Derivative of "Liberty" with respect to x, which amplifies sinusoids with large frequencies that oscillate along the x-direction.



Laplacian of "Liberty", which is another example of highpass (or "highboost") filter that amplifies the sinusoids with large spatial frequencies in proportion.



Lowpass filtering of "Liberty" in the frequency domain. In this case, sinusoidal frequencies outside of the circle are blocked completely. Since these frequencies that are necessary to reproduce edges are missing, the image is very blurry.



Laplacian sharpener applied to a blurry version of "Liberty"