AUVSI Foundation and ONR
Image Processing Primer Document
for
Autonomous Vehicle Competitions

Information Assembled by:
Dave Novick
for
Association for
Unmanned Vehicle Systems International (AUVSI) Foundation

US Navy Office of Naval Research (ONR)

Updated January 2014

Maritime RobotX Challenge

Introduction

Image manipulation can be divided into three categories:

- Image Processing — image in → image out
- Image Analysis — image in → measurements out
- Image Understanding — image in → high-level description out

This primer will focus on the concepts of image processing, restricting ourselves to two-dimensional (2D) image processing.

An image defined in the "real world" is considered to be a function of two real variables, for example, *a(x,y)* with *a* as the amplitude (e.g. Brightness) of the image at the real coordinate position *(x,y)*. An image may be considered to contain sub-images sometimes referred to as *regions-of-interest*, ROIs. The amplitudes of a given image will almost always be either real numbers or integer numbers. The latter is usually a result of a quantization process that converts a continuous range (say, between 0 and 100%) to a discrete number of levels.

Digital Image

A digital image *a[m,n]* described in a 2D discrete space is derived from an analog image *a(x,y)* in a 2D continuous space through a sampling process that is frequently referred to as digitization. The 2D continuous image *a(x,y)* is divided into *N rows* and *M columns*. The intersection of a row and column is termed a *picture element*, *image element* or *pixel.* The value assigned to the integer coordinates [*m,n*] with {*m*=0,1,2,...,M-1} and {*n*=0,1,2,...N-1} is *a[m,n]*.

There are standard values for the various parameters encountered in digital image processing. These values can be caused by video standards, by algorithmic requirements, or by the desire to keep digital circuitry simple. Quite frequently we see cases of $M=N=2^K$ where {K=8,9,10}

Color Space

The use of color in image processing is motivated by two principal factors. First, color is a powerful descriptor that often simplifies object identification and extraction from a scene. Second, humans can discern thousands of color shades and intensities, compared to about two dozen shades of gray. Most color models in use today are oriented either toward hardware (such as for color monitors) or toward applications where color manipulation is a goal (such as in the creation of color graphics for animation). In terms of digital image processing, the hardware-oriented models most commonly used in practice are the RGB (red, green, blue); the CMY (cyan, magenta, yellow); CMYK (cyan, magenta, yellow, black); and the HSI (hue, saturation, intensity) model which corresponds closely with the way humans describe and interpret color. The HSI model also has the advantage that it decouples the color and gray-scale techniques.

In the RGB model, each color appears in its primary spectral components of red, green, and blue. The number of bits used to represent each pixel in RGB space is called the *pixel depth*. Consider an RGB

image in which each of the red, green, and blue images is an 8-bit image. Each RGB color pixel (triplet value of (R,G,B)) is said to have a depth of 24 bits. Cyan, magenta and yellow are the secondary colors of light, or, alternatively, the primary colors of pigments.

When humans view a color object, we describe it by its hue, saturation, and brightness. Hue is an attribute associated with the dominant wavelength in a mixture of light waves. When we call an object red, orange, or yellow, we are specifying its hue. Saturation refers to the relative purity or the amount of white light mixed with a hue. The pure spectrum colors are fully saturated. Colors such as pink (red and white) and lavender (violet and white) are less saturated, with the degree of saturation being inversely proportional to the amount of white light added. The HSI color model decouples the intensity component from the color-carrying information (hue and saturation). Figure 1 shows an example of RGB and HIS color separation.

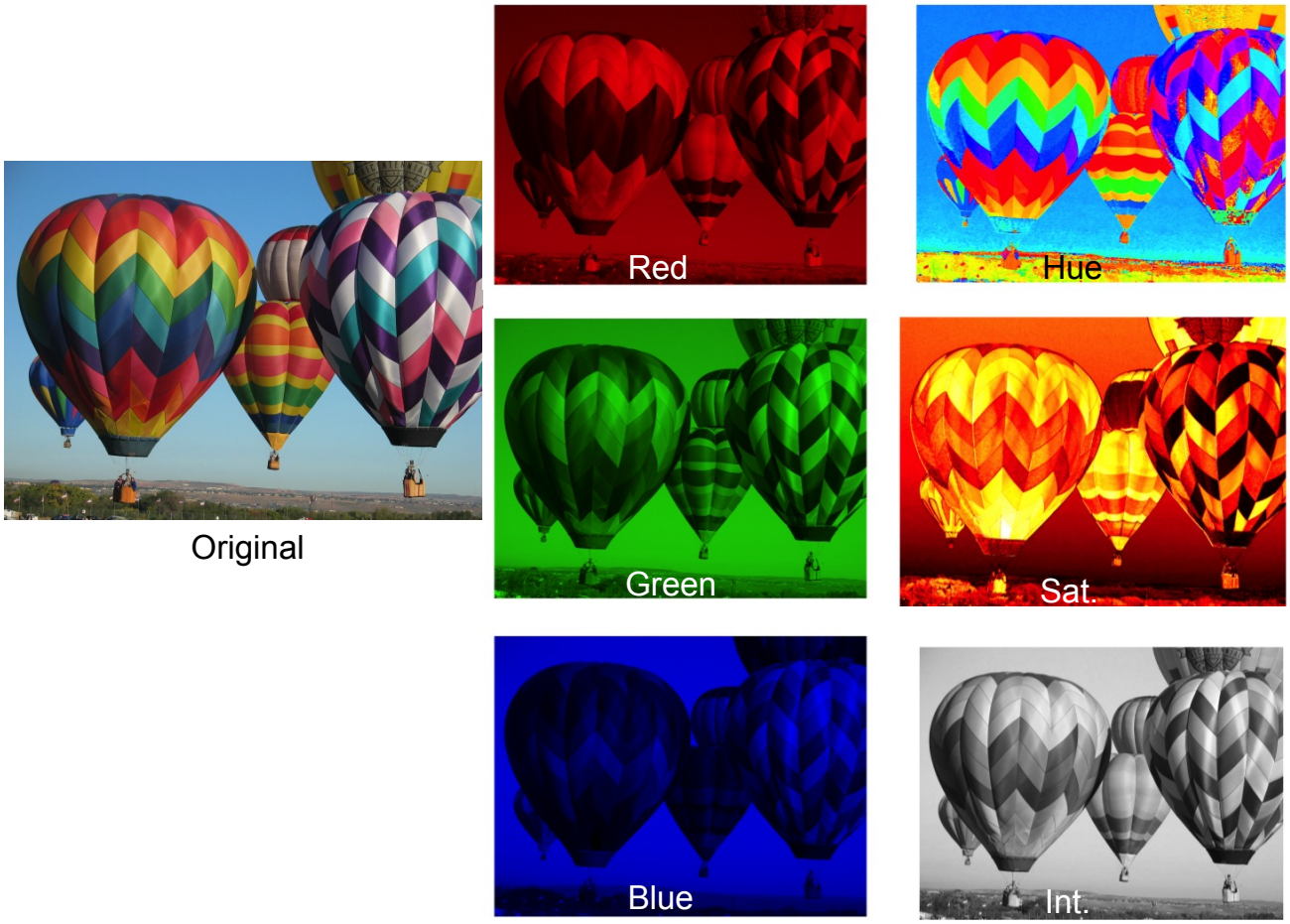


Figure 1: Color Space showing RGB and HSI

Characteristics of Image Operations

There is a variety of ways to classify and characterize image operations. The reason for doing so is to understand what type of results we might expect to achieve with a given type of operation or what might be the computational burden associated with a given operation.

Types of operations

The types of operations that can be applied to digital images to transform an input image *a[m,n]* into an output image *b[m,n]* (or another representation) can be classified into three categories: *Point* – the output value at a specific coordinate is dependent only on the input value at the same coordinate. *Local* – the output value at a specific coordinate is dependent on the input values in the *neighborhood* of that same coordinate. *Global* – the output value at a specific coordinate is dependent on all the values in the input image.

Types of neighborhoods

A pixel *p* at coordinates (x,y) has four horizontal and vertical neighbors whose coordinates are given by:

$$(x+1, y), (x-1,y), (x,y+1), (x,y-1)$$

This set of pixels, called the *4-neighbors* of *p* , is denoted by $N_4(p)$. Each pixel is a unit distance from (x,y), and some of the neighbors of *p* lie outside the digital image if (x,y) is on the border of the image.

The four diagonal neighbors of *p* have coordinates

$$(x+1, y+1), (x+1, y-1), (x-1, y+1), (x-1,y-1)$$

These points, together with the 4-neighbors, are called the *8-neighbors* of *p*, denoted by $N_8(p)$. As before, some of the points in the neighborhood fall outside the image if (x,y) is on the border of the image. An example of each is shown in Figure 2.

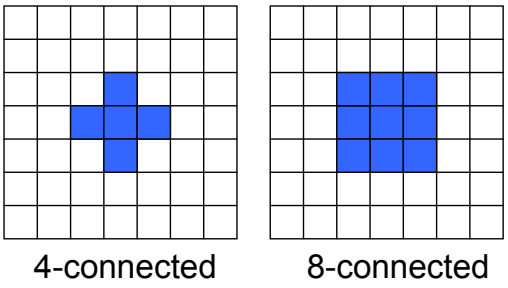


Figure 2: 4 and 8 connected neighborhoods

Video Parameters

We do not propose to describe the processing of dynamically changing images in this introduction. It is appropriate, given that many static images are derived from video cameras and frame grabbers, to mention the standards that are associated with the three standard video schemes that are currently in worldwide use: NTSC, PAL and SECAM. The information is summarized in *Table 1*.

Table 1. Various video schemes

	NTSC	PAL	SECAM
Images / second	29.97	25	25

ms / image	33.37	40.0	40.0
Lines / image	525	625	625
Aspect ratio	4:3	4:3	4:3
µs / line	63.56	64	64

In an interlaced image the odd numbered lines (1,3,5,...) are scanned in half of the allotted time (20 ms in PAL) and the even numbered lines (2,4,6,...) are scanned in the remaining half. The image display must be coordinated with the scanning format. The reason for interlacing the scan lines of a video image is to reduce the perception of flicker in a displayed image. If one is planning to use images that have been scanned from an interlaced video source, it is important to know if the two half-images have been appropriately "shuffled" by the digitization hardware or if this should be implemented in software. Further, the analysis of moving objects requires special care with interlaced video to avoid "zigzag" edges.

Tools

Certain tools are central to the processing of digital images. These include mathematical tools such as *convolution*, *Fourier* analysis, and *statistical* descriptions.

Convolution

The principal approach in defining a neighborhood about a point (x,y) is to use a square or rectangle subimage area, *T*, centered at (x,y). The center of the subimage is moved from pixel to pixel starting at a known location, i.e. at the top left corner. The operator *T* is applied at each location (x,y) to yield the output at that location. The process utilized only the pixels in the area of the image spanned by the neighborhood. Although other neighborhood shapes, such as approximations to a circle, sometimes are used, square and rectangle arrays are by far the most predominant because of their ease of implementation.

Fourier Transforms

The French mathematician Jean Baptiste Joseph Fourier found that any function that periodically repeats itself can be expressed as the sum of sines and/or cosines of different frequencies, each multiplied by a different coefficient (the *Fourier series*). Even functions that are not periodic can be expressed as the integral of sines and/or cosines multiplied by a weighting function (the *Fourier transform*). Both representations share the important characteristic that a function can be reconstructed (recovered) completely via an inverse process, with no loss of information. They allow us to work in the "Fourier domain" and then return to the original domain of the function without losing any information.

Given an image *a* and its Fourier transform *A*, then the forward transform goes from the spatial domain (either continuous or discrete) to the frequency domain which is always continuous.

$$A = F\{a\}$$

The inverse Fourier transform goes from the frequency domain back to the spatial domain.

$$a = F^{-1}\{A\}$$

Statistics

In image processing it is quite common to use simple statistical descriptions of images and sub-images. These can include the mean, median, and standard deviation. The average brightness of a region is defined as the sample mean of the pixel brightness within that region. The standard deviation is an estimate of the underlying brightness probability distribution. The mode of the distribution of pixels is the most frequent brightness value.

Noise

Images acquired through modern sensors may be contaminated by a variety of noise sources. By noise we refer to stochastic variations as opposed to deterministic distortions such as shading or lack of focus. We will assume the use of modern, charge-coupled device (CCD) cameras where photons produce electrons that are commonly referred to as photoelectrons. Nevertheless, most of the observations we shall make about noise and its various sources hold equally well for other imaging modalities.

Photon Noise

When the physical signal that we observe is based upon light, then the quantum nature of light plays a significant role. A single photon at $\lambda = 500$ nm carries an energy of $E = h\nu = hc/\lambda = 3.97 \times 10^{-19}$ Joules. Modern CCD cameras are sensitive enough to be able to count individual photons. The noise problem arises from the fundamentally statistical nature of photon production. We cannot assume that, in a given pixel for two consecutive but independent observation intervals of length *T,* the same number of photons will be counted. Photon production is governed by the laws of quantum physics which restrict us to talking about an average number of photons within a given observation window. If the appropriate formula for the SNR (in dB) is:

$$SNR = 10\log_{10}\rho T$$

where ρ is the rate or intensity parameter measured in photons per second. It is critical to understand that even if there were no other noise sources in the imaging chain, the statistical fluctuations associated with photon counting over a finite time interval *T* would still lead to a finite signal-to-noise ratio (*SNR*).

Thermal Noise

An additional, stochastic source of electrons in a CCD well is thermal energy. Electrons can be freed from the CCD material itself through thermal vibration and then, trapped in the CCD well, be indistinguishable from "true" photoelectrons. By cooling the CCD chip it is possible to reduce significantly the number of "thermal electrons" that give rise to thermal noise or *dark current*. As the integration time *T* increases, the number of thermal electrons increases. The probability distribution of thermal electrons is also a Poisson process where the rate parameter is an increasing function of temperature. There are alternative techniques (to cooling) for suppressing dark current and these

usually involve estimating the *average* dark current for the given integration time and then subtracting this value from the CCD pixel values before the A/D converter. While this does reduce the dark current *average*, it does not reduce the dark current *standard deviation* and it also reduces the possible dynamic range of the signal.

On-Chip Electronic Noise

This noise originates in the process of reading the signal from the sensor, in this case through the field effect transistor (FET) of a CCD chip.

KTC Noise

Noise associated with the gate capacitor of an FET is termed *KTC noise* and can be non-negligible, The output RMS value of this noise voltage is given by:

$$\sigma_{KTC} = \sqrt{kT/C}$$

where C is the FET gate switch capacitance, k is Boltzmann's constant, and T is the absolute temperature of the CCD chip measured in K. This value is a "one time" noise per pixel that occurs during signal readout, and thus independent of the integration time. Proper electronic design that makes use, for example, of correlated double sampling and dual-slope integration can almost completely eliminate KTC noise.

Amplifier Noise

The standard model for this type of noise is additive, Gaussian, and independent of the signal. In modern well-designed electronics, amplifier noise is generally negligible. The most common exception to this is in color cameras where more amplification is used in the blue color channel than in the green channel or red channel leading to more noise in the blue channel.

Quantization Noise

Quantization noise is inherent in the amplitude quantization process and occurs in the analog-to-digital converter (ADC). Quantization noise can usually be ignored as the total *SNR* of a complete system is typically dominated by the smallest *SNR*. In CCD cameras this is photon noise.

Cameras

The cameras and recording media available for modern digital image processing applications are changing at a significant pace. Nevertheless, the techniques that are used to characterize the CCD camera remain "universal" and the presentation that follows is given in the context of modern CCD technology for purposes of illustration.

Linearity

It is generally desirable that the relationship between the input physical signal (e.g. photons) and the output signal (e.g. voltage) be linear. In practice the relationship between input a and output c is frequently given by:

$$c = gain \times a^{\gamma} + offset$$

where γ is the *gamma* of the recording medium. For a truly linear recording system we must have $\gamma = 1$ and *offset* = 0. Unfortunately, the offset is almost never zero and thus we must compensate for this if the intention is to extract intensity measurements.

Sensitivity

There are two ways to describe the sensitivity of a camera. First, we can determine the minimum number of detectable photoelectrons. This can be termed the *absolute* sensitivity. Second, we can describe the number of photoelectrons necessary to change from one digital brightness level to the next, that is, to change one *analog-to-digital unit* (ADU). This can be termed the *relative* sensitivity.

SNR

As described previously, in modern camera systems the noise is frequently limited by:

- amplifier noise in the case of color cameras
- thermal noise which, itself, is limited by the chip temperature K and the exposure time, T
- photon noise which is limited by the photon production rate and the exposure time

Thermal noise (Dark current)

Using cooling techniques based upon Peltier cooling elements it is straightforward to achieve chip temperatures of 230 to 250 K. This leads to low thermal electron production rates. As a measure of the thermal noise, we can look at the number of seconds necessary to produce a sufficient number of thermal electrons to go from one brightness level to the next, an ADU, in the absence of photoelectrons. This last condition—the absence of photoelectrons—is the reason for the name *dark current*.

Photon noise

It should be possible to increase the *SNR* by increasing the integration time of the image and thus "capturing" more photons. The pixels in CCD cameras have, however, a finite well capacity.

Shading

Virtually all imaging systems produce shading. This means that if the physical input image $a(x,y)$ = *constant*, then the digital version of the image will not be constant. The source of the shading might be outside the camera such as in the scene illumination, or the result of the camera itself where *gain* and *offset* might vary from pixel to pixel.

Pixel Form

It is important to know the geometry for a given camera/digitizer system. In Figure 3 we define possible parameters associated with a camera and digitizer and the effect they have upon the pixel.

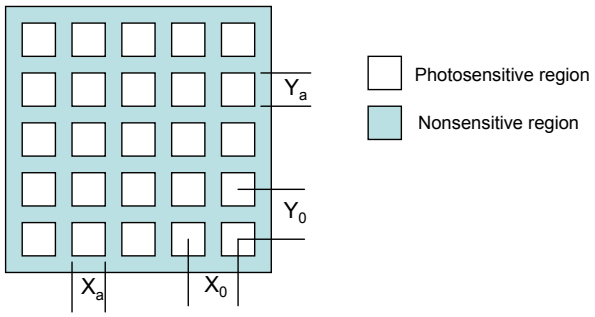


Figure 3: Pixel geomtery

The parameters X_0 and Y_0 are the spacing between the pixel centers, and represent the sampling distances. The parameters X_a and Y_a are the dimensions of that portion of the camera surface that is sensitive to light.

Square pixels

Square sampling implies that $X_0=Y_0$ or $X_0/Y_0=1$. It is not uncommon, however, to find frame grabbers where $X_0/Y_0=1.1$ (4:3). The risk associated with non-square pixels is that isotropic objects scanned with non-square pixels might appear isotropic on a camera-compatible monitor, but analysis of the objects (such as length-to-width ratio) will yield nonisotropic results.

The ratio X_0/Y_0 can be determined for any specific camera/digitizer system by using a calibration test chart with known distances in the horizontal and vertical direction. These charts are straightforward to make with modern laser printers. The test chart can then be scanned and the sampling distances X_0 and Y_0 determined.

Spectral Sensitivity

Sensors, such as those found in cameras and film, are not equally sensitive to all wavelengths of light. The spectral sensitivity for the CCD sensor is given in .

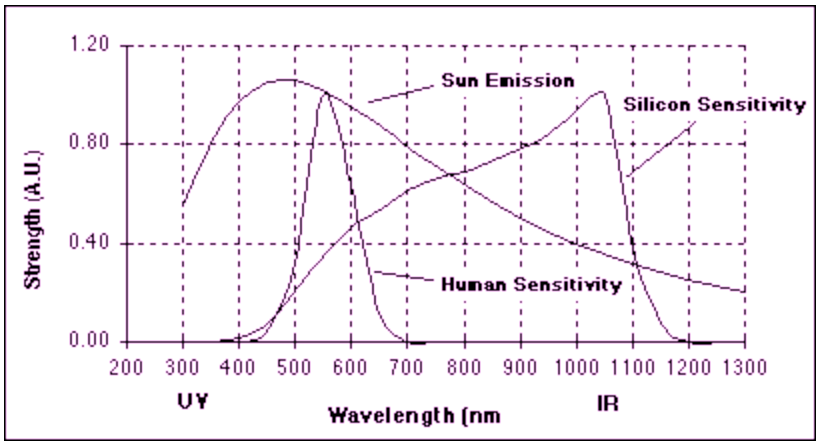


Figure 4: Spectral Sensitivity

The high sensitivity of silicon in the infra-red means that, for applications where a CCD (or other silicon-based) camera is to be used as a source of images for digital image processing and analysis, consideration should be given to using an IR blocking filter. This filter blocks wavelengths above 750 nm. Thus, it prevents "fogging" of the image from the longer wavelengths found in sunlight. Alternatively, a CCD-based camera can make an excellent sensor for the near infrared wavelength range of 750 nm to 1000 nm.

Shutter Speeds (Integration Time)

The length of time that an image is exposed (that photons are collected) may be varied in some cameras, or may vary on the basis of video formats. For reasons that have to do with the parameters of photography, this exposure time is usually termed *shutter speed* although integration time would be a more appropriate description.

Algorithms

In this section we will describe operations that are fundamental to digital image processing. These operations can be divided into four categories: operations based on the image histogram, on simple mathematics, on convolution, and on mathematical morphology. Further, these operations can also be described in terms of their implementation as a point operation, a local operation, or a global operation.

Histogram-Based Operations

An important class of point operations is based upon the manipulation of an image histogram or *region* histogram.

Contrast stretching

Frequently, an image is scanned in such a way that the resulting brightness values do not make full use of the available dynamic range. This can be easily observed in the histogram of the brightness values shown in Figure 5. By stretching the histogram over the available dynamic range we attempt to correct this situation. If the image is intended to go from brightness 0 to brightness 2^B-1 (where B is the number of bits quantifying the brightness) then one generally maps the 0% value (or *minimum*) to the value 0 and the 100% value (or *maximum*) to the value 2^B-1.

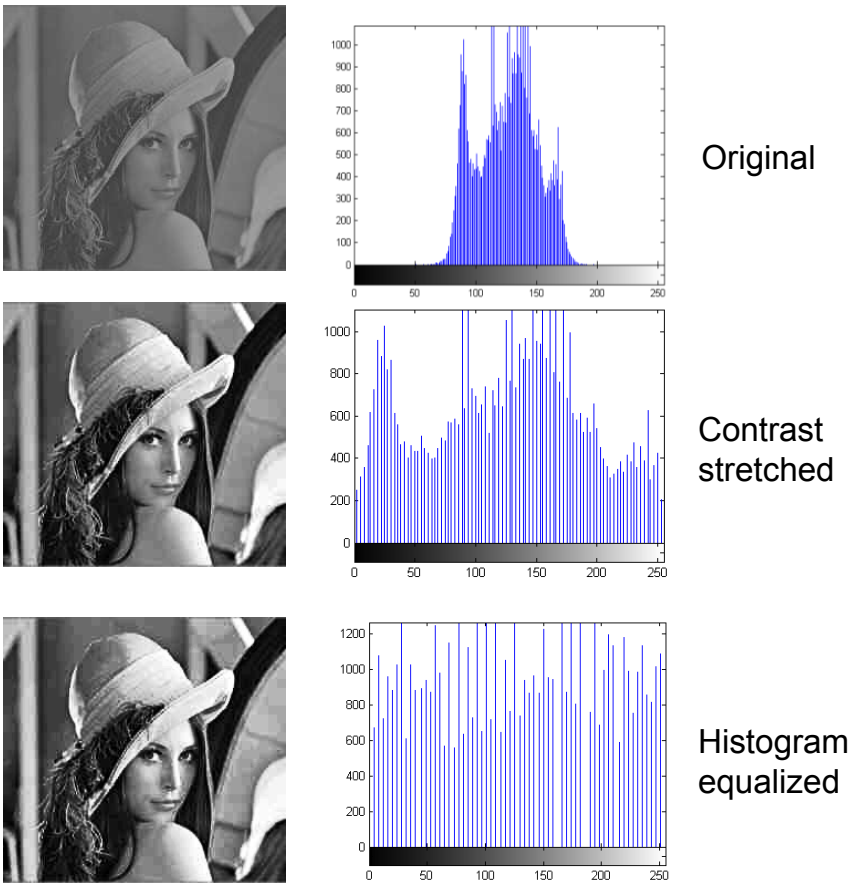


Figure 5: Contrast stretching & Histogram equalization

Equalization

When comparing two or more images on a specific basis, such as texture, it is common to first normalize their histograms to a "standard" histogram. This can be especially useful when the images have been acquired under different circumstances. The most common histogram normalization technique is *histogram equalization* where one attempts to change the histogram through the use of a function *b=f(a)* into a histogram that is constant for all brightness values, see Figure 5. This would correspond to a brightness distribution where all values are equally probable. Unfortunately, for an arbitrary image, one can only approximate this result.

Mathematics-Based Operations

In this section binary arithmetic and ordinary arithmetic are distinguished. In the binary case there are two brightness values "0" and "1". In the ordinary case we begin with 2^B brightness values or levels but the processing of the image can easily generate more levels. For this reason many *software* systems provide 16 or 32 bit representations for pixel brightness in order to avoid problems with arithmetic overflow.

Binary operations

Operations based on binary (Boolean) arithmetic are the basis for a powerful set of tools called

mathematical morphology. The operations are point operations and thus admit a variety of efficient implementations including simple look-up tables. The basic set of binary operations include NOT, OR, AND, and XOR (exclusive OR). Each operation is applied on a pixel-by-pixel basis.

Arithmetic-based operations

The gray-value point operations that form the basis for image processing are based on ordinary mathematics and include those in *Table 2*.

Table 2. Arithmetic-based image processing

Operation	Definition	Preferred data type
ADD	c = a + b	Integer
SUB	c = a - b	Integer
MUL	c = a●b	Integer or floating point
DIV	c = a / b	Floating point
LOG	c = log(a)	Floating point
EXP	c = exp(a)	Floating point
SQRT	c = sqrt(a)	Floating point
TRIG.	c = sin/cos/tan(a)	Floating point
INVERT	c = $(2^B - 1)$ - a	Integer

Convolution-based Operations

Convolution, the mathematical, *local* operation is central to modern image processing. The basic idea is that a window of some finite size and shape is scanned across the image. The output pixel value is the weighted sum of the input pixels within the window where the weights are the values of the filter assigned to every pixel of the window itself. The window with its weights is called the *convolution kernel*. The convolution is performed by sliding the kernel over the image, generally starting at the top left corner and moving the kernel through all the positions within the boundaries of the image. Each kernel position corresponds to a single output pixel, the value is which is calculated by multiplying together the kernel value and the underlying image pixel value for each of the cells in the kernel, and then adding all these numbers together, as shown in Figure 6.

a_{00}	a_{01}	a_{02}	a_{03}	a_{04}	a_{05}			…	a_{0n}
a_{10}	a_{11}	a_{12}	a_{13}	a_{14}					
a_{20}	a_{21}	a_{22}	a_{23}						
a_{30}									
a_{40}									
…									…
a_{m0}								…	a_{mn}

k_{00}	k_{01}	k_{02}
k_{10}	k_{11}	k_{12}
k_{20}	k_{21}	k_{22}

kernel

Figure 6: Convoluting an image

Smoothing Operations

These algorithms are applied in order to reduce noise and/or to prepare images for further processing such as segmentation. We distinguish between linear and non-linear algorithms where the former are amenable to analysis in the Fourier domain and the latter are not. We also distinguish between implementations based on a rectangular support for the filter and implementations based on a circular support for the filter.

Linear Filters

Several filtering algorithms will be presented together with the most useful supports.

- Uniform filter – The output image is based on local averaging of the input filter where all the values with the filter support have the same weight. See Figure 7
- Gaussian filter – The use of the Gaussian kernel for smoothing has become extremely popular. This has to do with certain properties of the Gaussian distribution as well as several application areas such as edge finding and scale space analysis. See Figure 7
- Other – The Fourier domain approach offers the opportunity to implement a variety of smoothing algorithms. The smoothing filters will then be *lowpass filters*. In general it is desirable to use a lowpass filter that has zero phase so as not to produce phase distortion when filtering the image.

Non-Linear Filters

A variety of smoothing filters have been developed that are not linear. While they cannot, in general, be submitted to Fourier analysis, their properties and domains of application have been studied extensively.

- Median filter – A median filter is based upon moving a window over an image (as in a

convolution) and computing the output pixel as the median value of the brightness within the input window. If the widow is (J, K) in size we can order the J●K pixels in brightness values from smallest to largest. If J●K is odd, then the medial will be the (J●K+1)/2 entry in the list of ordered brightness. Unlean a mean filter, the value selected will be exactly equal to one of the existing brightnesses (the output of integers will be an integer, and not a float that is generated by the mean). See Figure 7

- Kuwahara filter – Edges play an important role in our perception of images as well as the analysis of images. As such it is important to be able to smooth images without disturbing the sharpness and, if possible, the position of edges. A filter that accomplishes this goal is termed an *edge-preserving filter* and one particular example is the Kuwahara filter. Although this filter can be implemented for a variety of different window shapes, the algorithm will be described for a square window of size J = K = 4L + 1 where L is an integer. The window is partitioned into four regions. In each of the four regions (i=1, 2, 3, 4), the mean brightness m_i and the $variance_i$, are measured. The output value of the center pixel in the window is the mean value of that region that has the smallest variance.

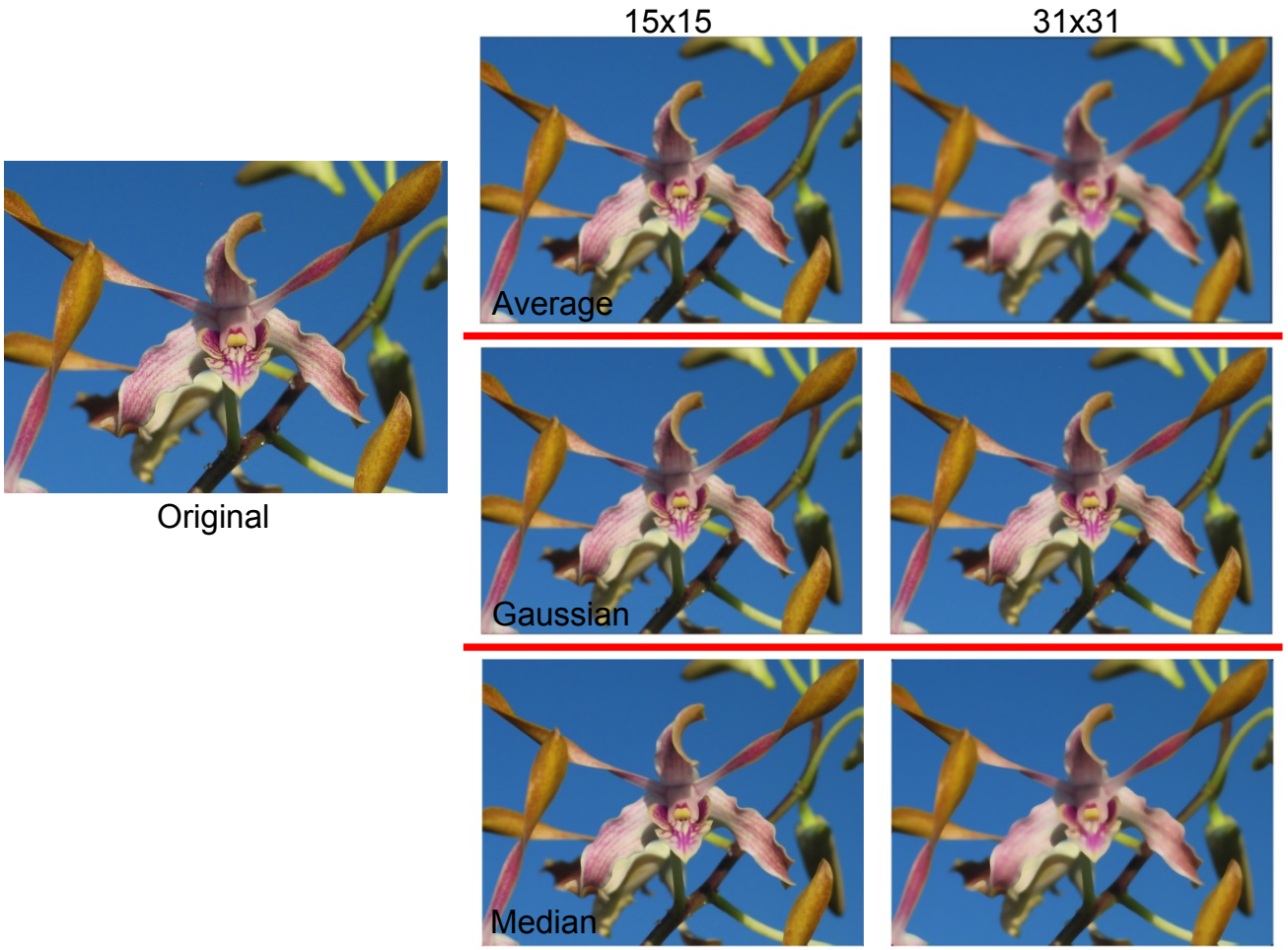


Figure 7: Linear, Gaussian and Median filter using a 15x15 and 31x31 kernel

Derivative-Based Operations

Just as smoothing is a fundamental operation in image processing, so is the ability to take one or more spatial derivatives of the image. The fundamental problem is that according to the mathematical definition of a derivative, this cannot be done. A digitized image is not a continuous function *a(x,y)* of the spatial variables, but rather a discrete function *a[m,n]* of the integer spatial coordinates. As a result the algorithms we will present can only be seen as *approximations* to be true spatial derivative of the original spatially-continuous image.

Taking the derivative multiplies the signal spectrum. This means that high frequency noise will be emphasized in the resulting image. The general solution to this problem is to combine the derivative operation with one that suppresses high frequency noise, in short, smoothing in combination with the desired derivative operation.

First Derivatives

As an image is a function of two (or more) variables it is necessary to define the direction in which the derivative is taken. For the two-dimensional case we have the horizontal direction, the vertical direction, or an arbitrary direction which can be considered as a combination of the two.

- Gradient filters – It is possible to generate a vector derivative description as the *gradient* of an image.
- Prewitt gradient filters – Both directions (horizontal and vertical) are separable. Each filter takes the derivative in one direction and smooths in the orthogonal direction using a one-dimensional version of a *uniform* filter. See Figure 8
- Sobel gradient filter – These filters are separable. Each filter takes the derivative in one direction and smoothes in the orthogonal direction using a one-dimensional version of a *triangular* filter. See Figure 8
- Gaussian gradient filters – In modern digital image processing one of the most common techniques is to use a Gaussian filter to accomplish the required smoothing and one of the gradient filters.

Original

Prewitt

Sobel

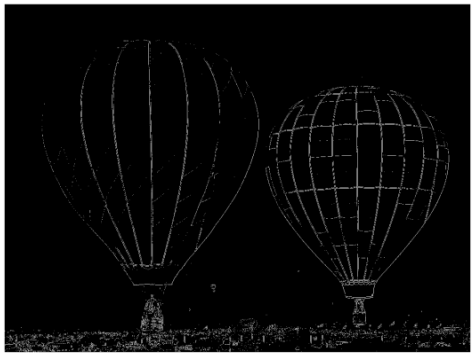
Figure 8: First order derivative functions

Second Derivatives

It is possible to compute higher-order derivatives of functions of two variables. In image processing the second derivatives or Laplacian play an important role.

- Basic second derivative filter.
- Frequency domain Laplacian. See Figure 9
- Gaussian second derivative filter – This is the straightforward extension of the Gaussian first derivative filter described above and can be applied independently in each dimension. We first apply Gaussian smoothing. Then the desired second derivative filter is applied. See Figure 9

Original



Laplacian

Laplacian
Of Gaussian

Figure 9: Second order derivative functions

Other Filters

A large number of filters, both linear and non-linear, are possible for image processing. It is therefore impossible to describe more then the basic types in this section. The description of other filters can be found in the reference literature as well as in the applications literature. It is important to use a small consistent set of relevant test images to understand the effect of a given filter or class of filters. The effect of filters on images can be frequently understood by the use of images that have pronounced regions of varying sizes to visualize the effect, or by the use of test patterns such as sinusoidal sweeps to visualize the effects in the frequency domain.

Morphology-Based Operations

We defined an image as an (amplitude) function of two, real (coordinate) variables *a(x,y)* or two, discrete variables *a[m,n]*. An alternative definition of an image can be based on the notion that an image consists of a set (or collection) of either continuous or discrete coordinates. In a sense the set corresponds to the points or pixels that belong to the objects in the image. For the moment we will consider the pixel values to be binary (0 or 1).

Fundamental definitions

The fundamental operations associated with an object are the standard set operations, *union*, *intersection* and *complement* plus *translation.*

Dilation and Erosion

Dilation, in general, causes objects to dilate or grow in size; *erosion* causes objects to shrink. The amount and the way that they grow or shrink depend upon the choice of the structuring element. The two most common structuring elements (given a Cartesian grid) are the 4-connected and 8-connected sets, N_4 and N_8. They are illustrated in Figure 10.

It is not necessary to process all the pixels in an object in order to compute a *dilation* or an *erosion*. We only have to process the boundary pixels. A number of "fast" algorithms can be found in the literature that are based on this result. The simplest dilation or erosion algorithms are frequently described as follows

- Dilation – Take each binary object pixel (with value "1") and set all background pixels (with value of "0") that are *C*-connected (the two most common structuring elements are the 4-connected and 8-connected sets) to that object pixel to the value "1".
- Erosion – Take each binary object pixel (with the value "1") that is *C*-connected to a background pixel and set the object pixel value to "0".

Opening and Closing

We can combine *dilation* and *erosion* to build two important higher order operations. The *opening* operation (erosion then dilation) can separate objects that are connected in a binary image. The *closing* operation (dilation then erosion) can fill in small holes and gaps. Both operations generate a certain amount of smoothing on an object's contour. See Figure 10 for some examples.

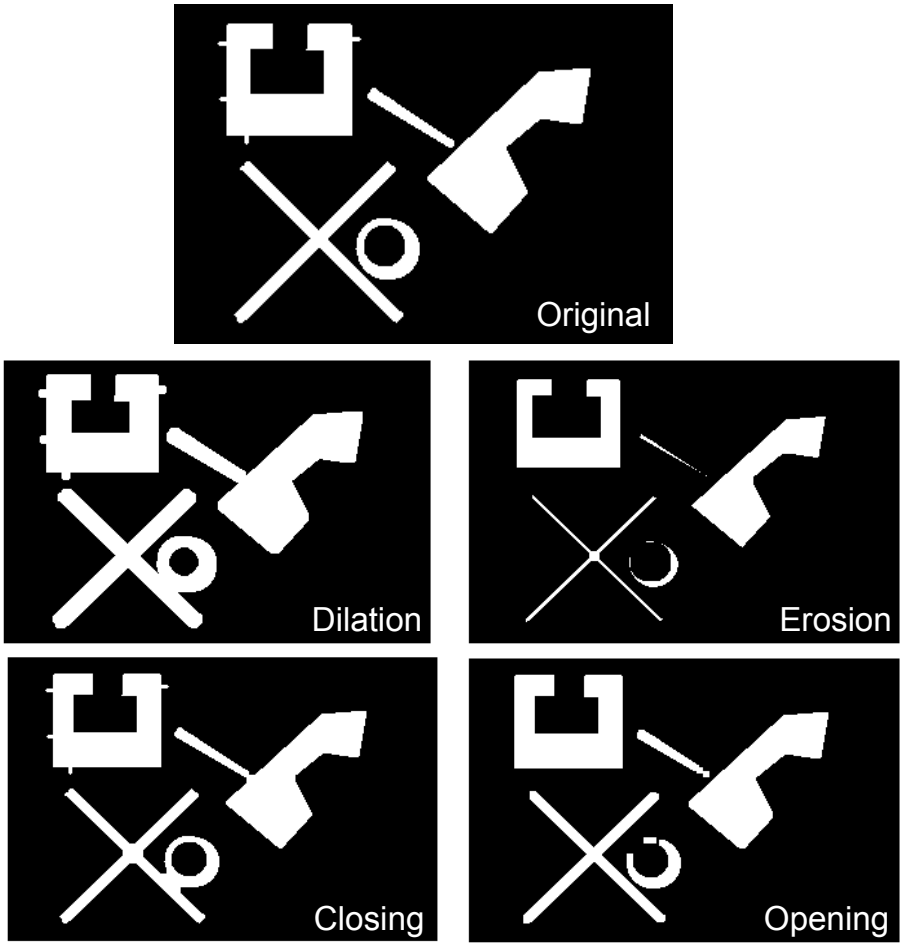


Figure 10: Examples of Morphological operations

Skeleton

Skeletonization is a process for reducing foreground regions in a binary image to a skeletal remnant that largely preserves the extent and connectivity of the original region while throwing away most of the original foreground pixels. The informal definition of a skeleton is a line representation of an object that is:

- one-pixel thick
- through the "middle" of the object
- preserves the topology of the object

One technique is to implement a *thinning*, an erosion that reduces the thickness of an object without permitting it to vanish. The alternative method is to first calculate the distance transform of the image. The skeleton then lies along the singularities in the distance transform. Figure 11 shows the original object and the skeleton represention in red.

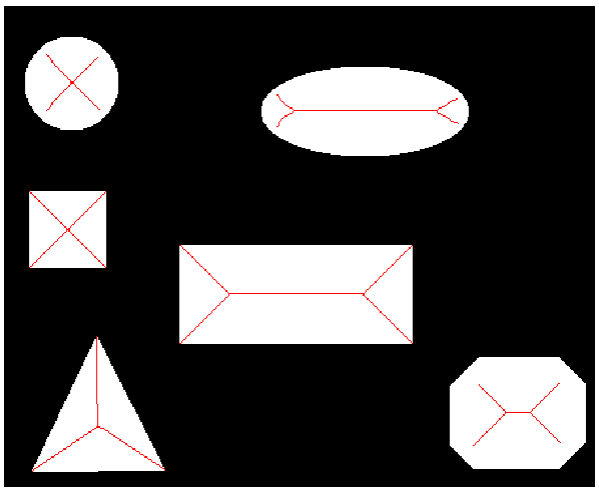


Figure 11: Skeletonization

Techniques

The algorithms presented can be used to build techniques to solve specific image processing problems. Without presuming to present the solution to all processing problems, the following examples are of general interest and can be used as models for solving related problems.

Basic Enhancement and Restoration Techniques

The process of image acquisition frequently leads to inadvertent image degradation. Due to mechanical problems, out-of-focus blur, motion, inappropriate illumination, and noise, the quality of the digitized image can be inferior to the original. The goal of *enhancement* is – starting from a recorded image *c[m,n]* – to produce the most visually pleasing image *â[m,n]*. The goal of *restoration* is – starting from a recorded image c[m,n] – to produce the best possible estimate *â[m,n]* of the original image *a[m,n]*. The goal of enhancement is beauty; the goal of restoration is accuracy.

Noise suppression

The techniques available to suppress noise can be divided into those techniques that are based on temporal information and those that are based on spatial information. By temporal information we mean that a sequence of images are available that contain *exactly* the same objects and that differ only in the sense of independent noise realizations. If this is the case and if the noise is additive, then simple averaging of the sequence will produce a result where the mean value of each pixel will be unchanged.

If temporal averaging is not possible, then spatial averaging can be used to decrease the noise. This generally occurs, however, at the cost to image sharpness.

Segmentation

In the analysis of the objects in images it is essential to distinguish between the objects of interest and "the rest." This latter group is also referred to as the background. The techniques that are used to find the objects of interest are usually referred to as *segmentation techniques* – segmenting the foreground

from background. In this section we will look at two of the most common techniques – *thresholding* and *edge finding* – and we will present techniques for improving the quality of the segmentation result. It is important to understand that:

- there is no universally applicable segmentation technique that will work for all images, and
- no segmentation technique is perfect.

Thresholding

This technique is based upon a simple concept. A parameter Θ called *brightness I* is chosen and applied to the image *a[m,n]* as follows:

IF $a[m,n] \geq \Theta$	$a[m,n] = object = 1$
ELSE	$a[m,n] = background = 0$

This version of the algorithm assumes that we are interested in light objects on a dark background.

The output is the label "object" or "background" which can be represented as a Boolean variable "1" or "0". In principle, the test condition could be based upon some other property than simple brightness (for example, *If (Redness{a[m,n]}*$\geq \Theta_{red}$*),* but the concept is clear.

The central question in thresholding then becomes: How do we choose the threshold Θ? While there is no universal procedure for threshold selection that is guaranteed to work on all images, there are a variety of alternatives.

- Fixed threshold – One alternative is to use a threshold that is chosen independently of the image data. If it is known that one is dealing with very high-contrast images where the objects are very dark and the background is homogeneous and very light, then a constant threshold might be sufficiently accurate. By accuracy, we mean that the number of falsely-classified pixels should be kept to a minimum.
- Histogram-derived thresholds – In most cases the threshold is chosen from the brightness histogram of the region or image that we wish to segment. A variety of techniques have been devised to automatically choose a threshold starting from the gray-value histogram.
- Background-symmetry algorithm – This technique assumes a distinct and dominant peak for the background that is symmetric about its maximum.

Thresholding does not have to be applied to entire images but can be used on a region by region basis. In each region, a threshold is calculated and the resulting threshold values are put together (interpolated) to form a thresholding surface for the entire image. The regions should be of "reasonable" size so that there are a sufficient number of pixels in each region to make an estimate of the histogram and the threshold.

Edge finding

Thresholding produces a segmentation that yields all the pixels that, in principle, belong to the object or objects of interest in an image. An alternative to this is to find those pixels that belong to the borders of

the objects. Techniques that are directed to this goal are termed *edge finding techniques*. There is an intimate relationship between edges and regions.

- Gradient-based procedure – The central challenge to edge finding techniques is to find procedures that produce *closed* contours around the objects of interest. For objects of particularly high SNR, this can be achieved by calculating the gradient and then using a suitable threshold. A variety of smoothing techniques can be used to reduce the noise effects before the gradient operator is applied.
- Zero-crossing based procedure – A more modern view to handling the problem of edges in noise images is to use the zero crossing generated in the Laplacian of an image. The rationale starts from the model of an ideal edge, a step function, that has been blurred. The edge location is, according to the model, at that place in the image where the Laplacian changes sign, the zero crossing. As the Laplacian operation involves a second derivative, this means a potential enhancement of noise in the image. To pervent enhanced noise from dominating the search for zero crossings, a smoothing is necessary. Gaussian smoothing is usually used, and produces the technique of the Laplacian of Gaussian or LoG.

Binary mathematical morphology

The various algorithms that we have described for mathematical morphology can be put together to form powerful techniques for the processing of binary images and gray level images. As binary images frequently result from segmentation processes on gray level images, the morphological processing of the binary image result permits the improvement of the segmentation result.

- Salt-or-pepper filtering – Segmentation procedures frequently result in isolated "1" pixels in a "0" neighborhood (salt) or isolated "0" pixels in a "1" neighborhood (pepper)
- Filling holes in objects – To fill holes in objects we use the following procedure:
  1. *Segment* the image to produce binary representation of objects
  2. Compute the *complement* of a binary image as a *mask image*
  3. Generate a *seed image* as the border of the image
  4. *Propagate* the *seed* into the *mask*
  5. *Complement* result of propagation to produce the final result
- Removing border-touching objects – Objects that are connected to the image border are not suitable for analysis. To eliminate them we can use a series of morphological operations:
  1. *Segment* the image to produce binary *mask images* of objects
  2. Generate a *seed image* as the border of the image
  3. *Propagate* the *seed* into the *mask*
  4. Compute *XOR* of the propagation result and the *mask image* as the final result
- Exo-skeleton – The exo-skeleton of a set of objects is the skeleton of the background that contains the objects. The exo-skeleton produces a partition of the image into regions each of which contains one object. The actual skeletonization is performed without the preservation of end pixels and with the border set to "0". The procedure is described below:

1. *Segment* image to produce binary image.
2. Compute *complement* of binary image
3. compute *skeleton* with border set to "0"

- Touching objects – Segmentation procedures frequently have difficulty separating slightly touching, yet distinct, objects. The following procedure provides a mechanism to separate these objects and makes minimal use of "magic numbers." The exo-skeleton produces a partition of the image into regions each of which contains one object. The actual skeletonization is performed without the preservation of end pixels and with the border set to "0". The procedure is described below:

  1. *Segment* image to produce binary image
  2. Compute a "small number" of *erosions* with a 4-connected neighborhood
  3. Compute *exo-skeleton* of eroded result
  4. Complement e*xo-skeleton* result
  5. Compute *AND* of binary image and the complemented exo-skeleton

References

1. Gonzales, R.C and Woods, R.E., *Digital Image Processing Second Edition*. 2002, Upper Saddle River, New Jersey: Prentice Hall
2. Young, I.T., Gerbrands, J.J. and van Vliet, L.J., *Fundamentals of Image Processing*. 1998, PDF
3. Castleman, K.R., *Digital Image Processing Second Edition* 1996, Englewood Cliffs, New Jeresy: Prentice-Hall.