2 Image Processing

Syllabus

Pixel transforms, color transforms, histogram processing, histogram equalization, filtering, convolution, Fourier transformation and its applications in sharpening, blurring and noise removal.

Contents

Point operators

The most basic image processing transformations are point operators, in which the value of each output pixel is determined solely by the value of the associated input pixel (plus, potentially, some globally collected information or parameters). Adjustments to brightness and contrast are examples of such operators. Point processes are another name for such procedures.

2.1 Pixel Transforms

- A function that takes one or more input images and produces an output image is called as a image processing operator. This can be written as in the continuous domain.

$$g(x) = h(f(x)) \text{ or } g(x) = h(f_0(x), \ldots, fn(x)), \quad (2.1.1)$$

- where x is in the function D-dimensional domain (typically D = 2 for images) and the functions f and g are the operate across a range, which can be the scalar or vector-valued, for example, for color images or 2D motion. For discrete (sampled) images, the domain consists of a finite number of pixel locations, x = (i, j), and this can be written

$$g(i, j) = h(f(i, j)). \quad (2.1.2)$$

- Two commonly used in point processes operation are multiplication and addition with a constant,

$$g(x) = af(x) + b. \quad (2.1.3)$$

- The gain and the bias parameters, a > 0 and b, are frequently referred to as the contrast and brightness controls, respectively.
- bias and gain parameters can also be spatially varying is given as,

$$g(x) = a(x)f(x) + b(x), \quad (2.1.4)$$

For example, when modeling vignetting in an optical system or simulating the graded density filter used by photographers to selectively darken the sky. Multiplicative gain (both global and spatially variable) is a linear operation that follows the notion of superposition,

$$h(f_0 + f_1) = h(f_0) + h(f_1) \quad (2.1.5)$$

Image squaring (which is frequently employed to derive a local estimate of the energy in a band-pass filtered signal) is not a linear operator. The linear mix operator is another extensively used dyadic (two-input) operator,

$$g(x) = (1 - \alpha)f_0(x) + \alpha f_1(x). \quad (2.1.6)$$

By varying α from $0 \rightarrow 1$, this operator can be used to perform a temporal cross-dissolve between two images or videos, as seen in slide shows and film production, or as a component of the image morphing algorithms .

The gamma correction, which is used to remove the non-linear mapping between input radiance and quantization pixel values, is a widely used non-linear transform that is typically performed to images before further processing. To invert the sensor's gamma mapping, with a gamma value of $\gamma \approx 2.2$ being an acceptable fit for most digital cameras.

$$g(x) = [f(x)]^{1/\gamma} \quad (2.1.7)$$

2.2 Color Transforms

While color images can be treated as arbitrary vector-valued functions or collections of the independent bands, it usually makes sense to think about them as highly correlated signals with strong connections to the image formation process, sensor design, and human perception. Consider the effect of adding a constant value to all three channels to brighten a picture. Adding the same value to each color channel not only enhances the perceived intensity of each pixel, but it can also change the hue and saturation of the pixel.

(a) Source image (b) Extracted foreground object F (c) Alpha matte α shown in grayscale (d) New composite C

Fig. 2.2.1 Image matting and compositing

After modifying (e.g., brightening) the luminance Y, chromaticity coordinates or even simpler color ratios can be utilized to re-compute a valid RGB image with the same hue and saturation. For improved visibility, Fig. 2.2.1 illustrates some color ratio images multiplied by the middle gray value. Color balancing (for example, to compensate for incandescent lighting) can be achieved by multiplying each channel with a different scale factor, or by a more involved process of mapping to XYZ color space, changing the nominal white point, and mapping back to RGB, which can be written down using a linear 3×3 color twist transform matrix.

Compositing and matting

It's common to wish to clip a foreground object from one image and set it on top of another in various picture editing and visual effects tools. Matting is the technique of

eliminating an object from an image, whereas compositing is the process of incorporating it into another image (without obvious artifacts). The foreground object's intermediate representation between these two stag. In addition to the three color RGB channels, an alpha-matted image contains a fourth alpha channel α (or A) that describes the relative amount of opacity or fractional coverage at each pixel . Transparency is the polar opposite of opacity. Pixels inside the object are fully opaque ($\alpha = 1$), whereas pixels outside the object are completely transparent ($\alpha = 0$). The perceived apparent jaggies that arise when only binary opacities are employed are hidden by pixels on the object's edge varying smoothly between these two extremes.

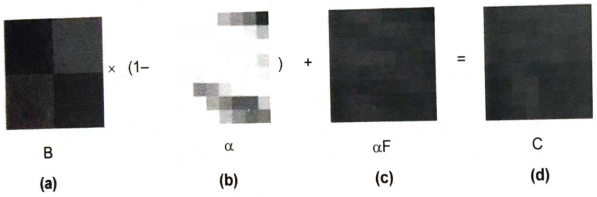


Fig. 2.2.2 Compositing equation C = (1 − α)B + αF . The images are taken from a close-up of the region of the hair in the upper right part of the lion.

To composite a new (or foreground) image on top of an old (background) image, the over operator, first proposed by Porter and Duff (1984) and then studied extensively by Blinn, is used,

$$C = (1 - \alpha)B + \alpha F. \qquad (2.2.1)$$

This operator attenuates the influence of the background image B by a factor $(1 - \alpha)$ and then adds in the color (and opacity) values corresponding to the foreground layer F , as shown in Fig. 2.2.2.

In many situations, it is convenient to represent the foreground colors in pre-multiplied form, i.e., to store (and manipulate) the αF values directly. For various reasons, including the ability to blur or resample (e.g., rotate) alpha-matted images without any additional complexities, the pre-multiplied RGBA representation is preferable (just treating each RGBA band independently). Because the pure un-multiplied foreground colors F remain constant (or move slowly) in the neighborhood of the object, they are employed for matting utilizing local color consistency.

2.3 Histogram Processing

A histogram is a graph that shows frequency of anything. Usually histogram have bars that represent frequency of occurring of data in the whole data set. The histogram of a picture, like other histograms, shows frequency. In contrast, a picture histogram shows the frequency of pixel intensity levels. In an image histogram, the gray level intensities are shown on the x axis, and the frequency of these intensities is shown on the y axis. The x axis and the y axis are the two axes of a histogram. The x axis contains event whose frequency you have to count. The y axis contains frequency.

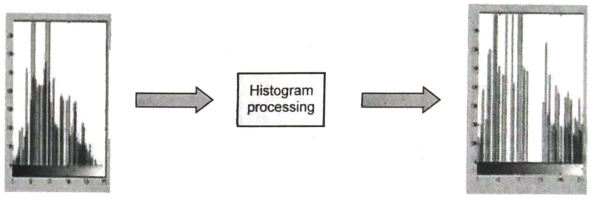


Fig. 2.3.1 Histogram Processing

The histogram of a digital image with gray levels in the range [0, L–1] is a discrete function of the form

$$\mathbf{H(rk) = nk}$$

where rk is the k^{th} gray level and nk is the number of pixels in the image having the level rk.

A normalized histogram is given by the equation

$$p(rk) = nk/n \text{ for } k = 0,1,2,\ldots..,L-1$$

P(rk) gives the estimate of the probability of occurrence of gray level rk. The sum of all components of a normalized histogram is equal to 1.The histogram plots are simple plots of p(rk)=nk versus rk.

The components of the histogram are concentrated on the low (dark) side of the gray scale in the dark image. In the case of a bright image, the histogram components are skewed to the gray scale's high end. A low contrast image's histogram will be narrow and centered near the middle of the gray scale. The histogram components in the high contrast image cover a wide range of gray scale. The end result will be an image with a lot of gray level details and a wide dynamic range.

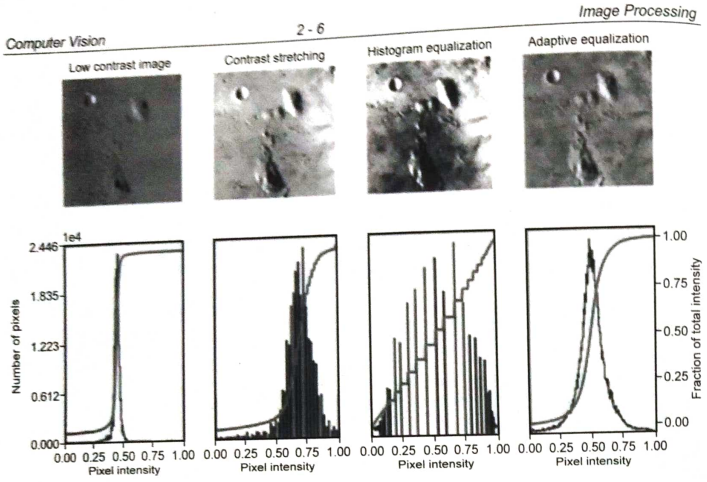


Fig. 2.3.2 Histogram equalization

2.4 Histogram Equalization

Equalization of histograms is a typical approach for improving the appearance of photographs. Assume that we have a largely dark image. The visual detail is compressed towards the dark end of the histogram, and the histogram is skewed towards the lower end of the grey scale. If we could 'stretch out' the grey levels at the dark end to produce a more uniformly distributed histogram then the image would become much clearer.

Let there be a continuous function with r being gray levels of the image to be enhanced. The range of r is [0, 1] with r = 0 representing black and r = 1 representing white. The transformation function is of the form. One approach might be to look at the darkest and brightest pixel values in an image and map them to pure black and pure white. Another approach might be to find the average value in the image, push it towards middle gray, and expand the range so that it more closely fills the displayable values S = T(r) where $0 < r < 1$.

It produces a level s for every pixel value r in the original image.

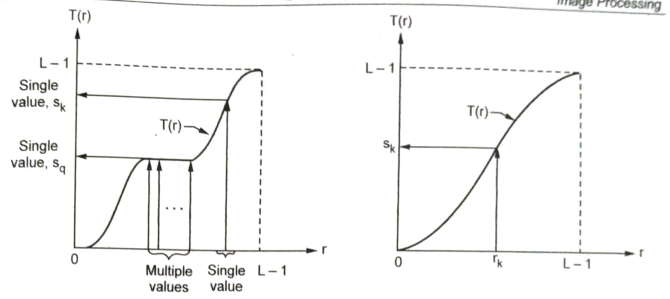


Fig. 2.4.1 monotonically increasing function (a) multiple value mapped to single value (b)one-to-one mapping

The transformation function is assumed to fulfill two condition T(r) is single valued and monotonically increasing in the internal $0 < T(r) < 1$ for $0 < r < 1$.The transformation function should be single valued so that the inverse transformations should exist. Monotonically increasing condition preserves the increasing order from black to white in the output image. The second conditions guarantee that the output gray levels will be in the same range as the input levels. The gray levels of the image may be viewed as random variables in the interval [0.1]. The most fundamental descriptor of a random variable is its probability density function (PDF) $P_r(r)$ and $P_s(s)$ denote the probability density functions of random variables r and s respectively. Basic results from an elementary probability theory states that if $P_r(r)$ and T_r are known and $T - 1(s)$ satisfies conditions (a), then the probability density function $P_s(s)$ of the transformed variable is given by the formula

$$P_s(s) = P_r r \left| \frac{dr}{ds} \right|$$

input image and by the chosen transformations function. This is the cumulative distribution function of r.A transformation function of a particular importance in image processing.

$$s = T\left(\int_0^r P_r(r)\, w\, (dw)\, r \right)$$

L is the total number of possible gray levels in the image.

One method is to look at the image's darkest and brightest pixel values and transfer them to pure black and white. Another option is to find the image's average value, shift it towards middle gray, and then widen the range so that it more closely fills the

displayable. We can compute key statistics such as the lowest, maximum, and average intensity values using this distribution. It's worth noting that the image has an abundance of dark and light values, but the mid-range values are mainly unpopulated. Wouldn't it be better if we could brighten a few at the same time ? One popular answer to this question is to perform histogram equalization, i.e., to find an intensity mapping function f(I) such that the resulting histogram is flat. The trick to finding such a mapping is the same one that people use to generate random samples from a probability density function, which is to first compute the cumulative distribution function

Think of the original histogram h(I) as the distribution of grades in a class after some exam. The answer is to integrate the distribution h(I) to obtain the cumulative distribution c(I),

$$c(I) = \frac{1}{N}\sum_{i=0} h(i) = c(I-\frac{1}{N}) + 1h(I) \qquad \ldots(2.4.1)$$

where N is the number of pixels in the image or students in the class. For any given grade or intensity, we can look up its corresponding percentile c(I) and determine the final value that pixel should take. When working with eight-bit pixel values, the I and c axes are rescaled from [0, 255].

(a) Original image (b) Block histogram equalization (c) Full locally adaptive equalization

Fig. 2.4.2 Locally adaptive histogram equalization

The result of applying f (I) = c(I) to the original image. As we can see, the resulting histogram is flat; so is the resulting image (it is "flat" in the sense of a lack of contrast and being muddy looking). One way to compensate for this is to only partially compensate for the histogram unevenness, e.g., by using a mapping function f (I) = αc(I) + (1 − α)I, which is a linear blend between the cumulative distribution function and the identity transform (a straight line). The resulting image maintains more of its original grayscale distribution while having a more appealing balance. Another potential problem with histogram equalization (or, in general, image brightening) is that noise in dark regions can be amplified and become more visible.

2.5 Filtering

Another technique to alter an image is to apply filters to it. And, unlike a point operation, the filter generates a new pixel value by using more than one pixel. Smoothing filters, for example, replace a pixel value with the average of its nearby pixel values. Linear and non-linear filters are the two types of filters.

Linear filter :

A linear filter operates on the pixel value in the support region in a linear fashion (i.e.,as weighted summation). The 'filter matrix' defines the support region, which is represented as H(i,j). The size of H is referred to as the 'filter region,' and the filter matrix has its own coordinate system, with I representing the column index and j representing the row index. The origin location is nicknamed the "hot s" because it is in the heart of it.

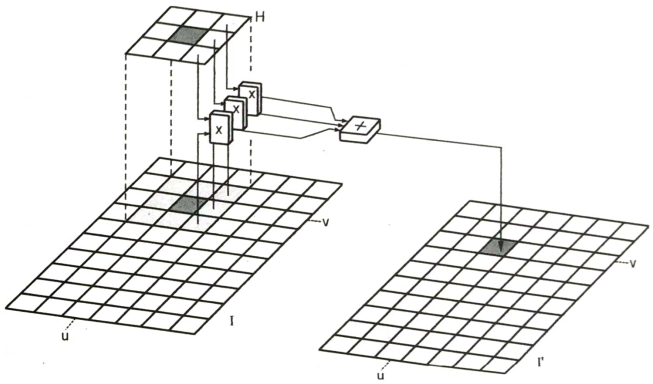


Fig. 2.5.1 Linear filter with filter matrix

Applying weight median filter to the image I, a hotspot location is at the orange shade (center of the filter matrix H)

45	60	98	127	132	133	137	133
46	65	98	123	126	128	131	133
47	65	96	115	119	123	135	137
47	63	91	107	113	122	138	134
50	59	80	97	110	123	133	134
49	53	68	83	97	113	128	133
50	50	58	70	84	102	116	126
50	50	52	58	69	86	101	120

f(x, y)

*

0.1	0.1	0.1
0.1	0.2	0.1
0.1	0.1	0.1

h(x, y)

=

69	95	116	125	129	132
68	92	110	120	126	132
66	86	104	114	124	132
62	78	94	108	120	129
57	69	83	98	112	124
53	60	71	85	100	114

g(x, y)

Fig. 2.5.2 Original image function I with filter matrix h

Neighborhood filtering (convolution): To create the image on the right, the image on the left is convolved with the filter in the middle. The source neighborhood for the light green destination pixel is indicated by the light blue pixels. where the sign of the offsets in f has been reversed. This is called the convolution operator,and h is then called the impulse response function.

$$g = f * h,$$

Applying the filter

To apply the filter to the image, please follow these step.

- Move the filter matrix over the image I and H(0,0) must go along with the current image position (u,v).
- Multiply each filter coefficient H(i,j) with the corresponding image element I(u+i,v+j).
- Average all result from the previous step and it is the result for the current location I(u,v).

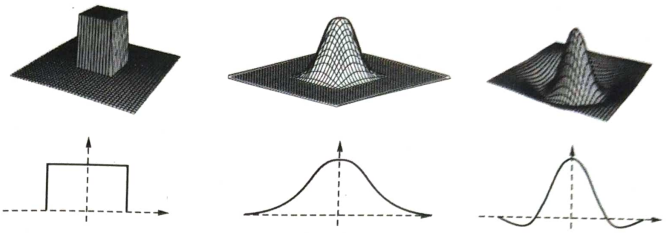
$$I'(u, v) \leftarrow \sum_{i=-1}^{i=1} \sum_{j=-1}^{j=1} I(u + i, v + j) \cdot H(i, j)$$

Equation Linear filter type

Smoothing Filter (Only positive integers are used in this filter.)

- Box filter. This filter's members are all the same.
- Gaussian filter. The weight of a filter member is determined by its position. The largest weight is received at the center of the filter, and it diminishes as you move away from it.

Different Filter



0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

(a)

0	1	2	1	0
1	3	5	3	1
2	5	9	5	2
1	3	5	3	1
0	1	2	1	0

(b)

0	0	−1	0	0
0	−1	−2	−1	0
−1	−2	16	−2	−1
0	−1	−2	−1	0
0	0	−1	0	0

(c)

Fig. 2.5.3 3D structure, 2D structure and example of filter (a) Box filter (b) Gaussian filter and (c) Laplace filter

Laplace or Mexican hat filter. Some of the members of this filter are negative filters, and it can be calculated by adding the positive and negative members together.

Properties of Linear Filter

This process is known as "Linear Convolution." The convolution operation is described as the equation for two-dimensional functions I and H.

$$I'(u, v) \leftarrow \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(u + i, v + j) \cdot H(i, i)$$

$$I' = I * H$$

this operation provide the similar result with the linear filter with the filter function which reflect in both horizontal and vertical axis. The convolution matrix H can be called kernel.

Properties of Linear Convolution

- Commutativity
- Linearity

- Associativity
- Seperability: The kernel H can be represented as the convolution of multiple kernels and can separated in a pair dimensional kernel x and y.

Non-Linear Filters

The consequence of reducing noise with a smoothing filter (a linear filter) is a burred image structure, line, and edge. This problem was solved using non-linear filters, which act in a non-linear fashion.

Types of non-linear filters

- **Minimum and Maximum Filters :** The minimum and maximum values in the original image's moving region R are the result of the minimum and maximum filters. These filters were described as follows :

$$I'(u, v) \leftarrow \min \{I(u+i, v+j) \mid (i, j) \in R\},$$

$$I'(u, v) \leftarrow \max \{I(u+i, v+j) \mid (i, j) \in R\}$$

- **Median Filter :** The result was computed in the same way that the minimum and maximum filters were computed. The median filter produces the median of all values in moving area R. This filter is commonly used to eliminate salt and pepper noise from images. This filter's definition was :

$$I'(u, v) \leftarrow \text{median} \{I(u+i, v+j) \mid (i, j) \in R\}$$

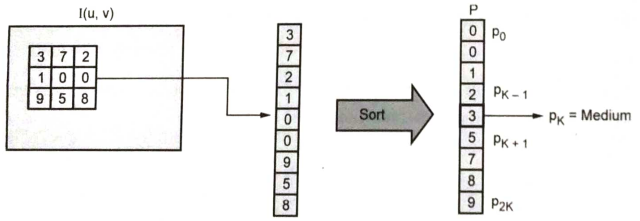


Fig. 2.5.4 Median filter for sort

Weight Median Filter : The word "Weight" refers to the extension of each member in the kernel according to the weight matrix W, which is similar to the median filter. The filter member closest to the hot location will have a higher weight than the others.

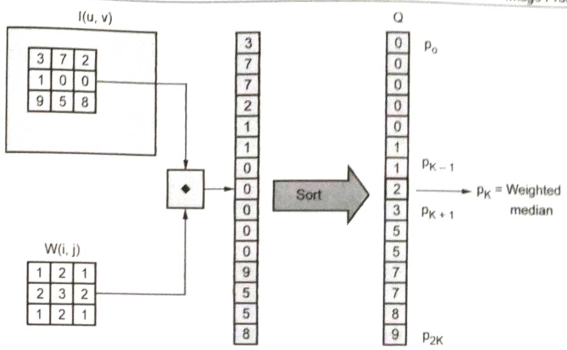


Fig. 2.5.5 Weighted median filter with weight matrix for sort

2.6 Convolution

Overlay a revolving mask on the image. This is how it is done. Place the mask's center at the middle of each picture element. Multiply the corresponding parts, then put them together, and paste the result into the image element where you want to set the mask's center. To begin, breakdown the input signal into a series of impulses, each of which can be seen as a scaled and shifted delta function. Second, each impulse produces an output that is a scaled and shifted replica of the impulse response. Finally, by summing these scaled and shifted impulse responses, the overall output signal may be obtained. In other applications, the impulse response is referred to by a different term. The impulse response is known as the filter kernel, the convolution kernel, or simply the kernel if the system under consideration is a filter. The point spread function is the name given to the impulse response in image processing. While these phrases are employed in a variety of ways, they all refer to the same signal produced by a system when the input is a delta function. Convolution, like multiplication, addition, and integration, is a formal mathematical operation. Convolution takes two signals and produces a third signal, whereas addition takes two numbers and produces a third number.

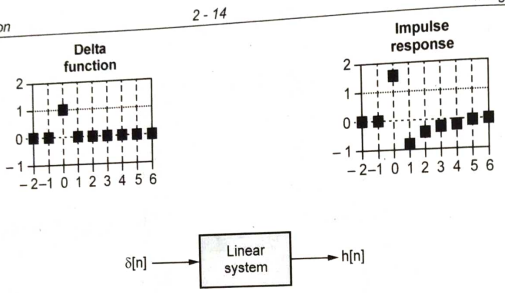


Fig. 2.6.1 Delta function and impulse response

Convolution is utilized in many branches of mathematics, including probability and statistics. Convolution is a mathematical term that describes the relationship between three signals of interest : The input signal, the impulse response, and the output signal, in linear systems. When convolution is employed with linear systems, the notation is shown in Fig. 2.6.2. A linear system having an impulse response, h[n], receives an input signal, x[n], and produces an output signal, y[n]. x[n] * h[n] = y[n] in equation form.

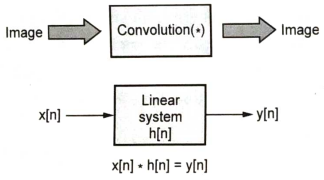


Fig. 2.6.2 Convoluted with impulse response

Expressed in words, the input signal convolved with the impulse response is equal to the output signal. Just as addition is represented by the plus (+), and multiplication by the cross (×), convolution is represented by the star (*). It is unfortunate that most programming languages also use the star to indicate multiplication.

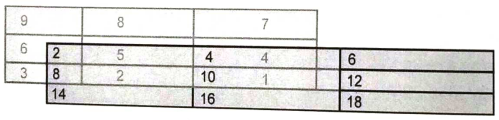


Fig. 2.6.3 Shows convolution steps

The mask is represented by the red box, and the values in orange are the mask's values. The image owns the black color box and its values. The value for the first pixel of the image will now be calculated as,

$$\begin{aligned}\text{First pixel} &= (5*2) + (4*4) + (2*8) + (1*10)\\ &= 10 + 16 + 16 + 10\\ &= 52\end{aligned}$$

Place 52 in the original image at the first index and repeat this procedure for each pixel of the image.

2.7 Fourier Transformation

A complex-valued function of frequency, whose magnitude (absolute value) is the amount of that frequency present in the original function, and whose argument is the phase offset of the fundamental sinusoid at that frequency, is the Fourier transform of a function of time. Although the Fourier transform is not confined to time functions, the original function's domain is typically referred to as the time domain. The Fourier Transform is an important image processing tool which is used to divide a picture into its sine and cosine components. The image in the Fourier or frequency domain is represented by the output of the transformation, whereas the spatial domain equivalent is represented by the input image. Each point in the Fourier domain image indicates a frequency contained in the spatial domain image.

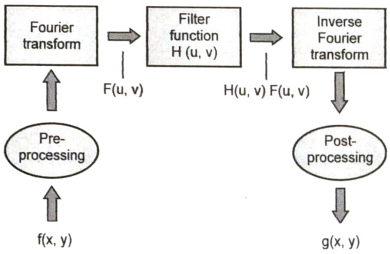


Fig. 2.7.1 Fourier transformation

1-D Fourier Transformation and its Inverse

Fourier transform is one of the most commonly used techniques in (linear) signal processing and control theory. It provides one-to-one transform of signals from / to a time-domain representation f(t) to/from a frequency domain representation F(ξ). It allows a frequency content (spectral) analysis of a signal. FT is suitable for periodic

signals. If the signal is not periodic then the windowed FT or the linear integral transformation with time (spatially in 2D) localized basis function, e.g., wavelets, Gabor filters can be used. $F\{f(t)\} = F(\xi)$, where ξ [Hz = s −1] is a frequency and $2\pi\xi$ [s −1] is the angular frequency

Fourier Tx	Inverse Fourier Tx
$F(\xi) = \int_{-\infty}^{\infty} f(t)\, e^{-2\pi i \xi t}\, dt$	$F(\xi) = \int_{-\infty}^{\infty} F(t)\, e^{2\pi i \xi t}\, d\xi$

Convolution (in functional analysis) is an operation on two functions f and h, which produces a third function (f * h), often used to create a modification of one of the input functions. Convolution is an integral 'mixing' values of two functions, i.e., of the function h(t), which is shifted and overlayed with the function f(t) or vice-versa. The limits can be constraint to the interval [0, t], because we assume zero values of functions for the negative argument.

$$(f * t)(t) = (h * f)(t) = \int_0^t f(\tau)\, h(t-\tau)\, d\tau = \int_0^t f(\tau - \tau)\, h(\tau)\, d\tau$$

Fourier Tx, properties

Property	f(t)	F(ξ)
Linearity	$af_1(t) + b f_2(t)$	$aF_1(\xi) + bF_2(\xi)$
Duality	F(t)	F(− ξ)
Convolution	(f * g) (t)	F(ξ)
Product	f(t) g(t)	(F * G) (ξ)
Time Shift	$f(t - t_0)$	$e^{-2\pi i \xi t} f(t)$
Frequency shift	$e^{2\pi i \xi_0 t} f(t)$	$F(\xi - \xi_0)$
Differentiation	$\frac{df(t)}{dt}$	$2\pi i \xi F(\xi)$
Multiplication by t	t f(t)	$\frac{i}{2\pi}\frac{dF\xi}{d\xi}$
Time scaling	f (a t)	$\frac{1}{a} F(\xi/a)$

Area in time	$F(0) = \int_{-\infty}^{\infty} f(t)\, dt$	Area function f(t)
Area in freq	$F(0) = \int_{-\infty}^{\infty} f(\xi)\, d\xi$	Area under F(ξ)
Parseval's th.	$\int_{-\infty}^{\infty} \lvert f(t)\rvert^2\, dt = \int_{-\infty}^{\infty} \lvert f(\xi)\rvert^2\, d\xi$	f energy = F energy

Discrete Fourier transform

Let f(n) be an input signal (a sequence), n = 0, . . . , N − 1. Let F(k) be a Frequency spectrum (the result of the discrete Fourier transformation) of a signal f(n). If there is a single variable, continuous function f(n) , then Fourier transformation F (k) may be given as,

Discrete Fourier transformation

$$F(k) = \sum_{n=0}^{N-1} F(n)\, e^{-\frac{2\pi i}{N} kn}$$

Inverse discrete Fourier transformation

$$f(n) = \frac{1}{N} \sum_{k=0}^{N-1} F(k)\, e^{\frac{2\pi i}{N} kn}$$

Because the DFT is a sampled Fourier Transform, it does not contain all of the frequencies that make up an image, but only a sufficient number of samples to fully characterize the spatial domain image. The number of frequencies is the same as the number of pixels in the spatial domain image, indicating that the spatial and Fourier domain images are of equal size.

For a square image of size N×N, **the two-dimensional DFT** is given by :

$$F(k, l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f(i, j)\, e^{-i2\pi\left(\frac{ki}{N} + \frac{lj}{N}\right)}$$

The exponential term is the basis function corresponding to each point F(k,l) in the Fourier space, and f(a,b) is the picture in the spatial domain. The value of each point F(k,l) is generated by multiplying the spatial picture with the associated base function and summing the result, as shown in the equation.

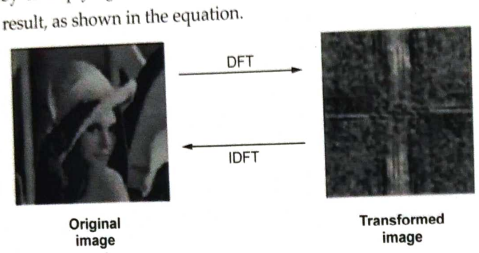


Fig. 2.7.2 Applying DFT and IDFT

Fast Fourier Transform (FFT)

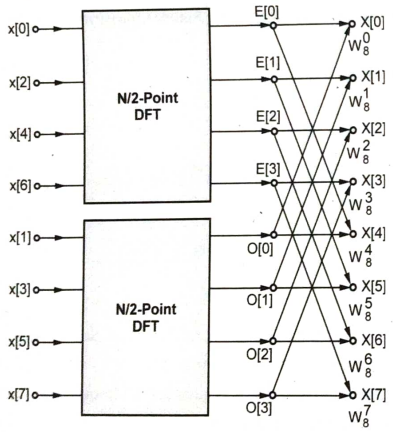


Fig. 2.7.3 Fast Fourier Transform (FFT)

A Fast Fourier Transform (FFT) is an algorithm that computes the discrete Fourier transform (DFT) of a sequence, or its inverse (IDFT). Fourier analysis converts a signal from its original domain (often time or space) to a representation in the frequency domain and vice versa. The DFT is obtained by decomposing a sequence of values into components of different frequencies. This operation is useful in many fields, but

computing it directly from the definition is often too slow to be practical. A Fast Fourier Transform (FFT) is an efficient algorithm to compute the discrete Fourier transform and its inverse. Statement : FFT has the complexity O(Nlog 2N). Example (according to numerical recepies in C) : A sequence of N = 106 , 1 µsecond computer. FFT 30 seconds of CPU time. DFT 2 weeks of CPU time, i.e., 1,209,600 seconds, which is about 40.000 × more.

2-D Fourier Transform

The Fourier transform can be applied to a larger number of dimensions. Many signals, for example, are 2D space functions defined on an x-y plane. Depending on whether the 2D signal is periodic or discrete, the two-dimensional Fourier transform has four possible forms.

Aperiodic, continuous signal, continuous, aperiodic spectrum

$$f(x, y) = \int_{-\infty}^{\infty}\int F(u, v)\, e^{j2\pi(ux + vy)}\, du\, dv$$

$$F(u, v) = \int_{-\infty}^{\infty}\int f(x, y)\, e^{-j2\pi(ux + vy)}\, dx\, dy$$

Where **u** and **v** are spatial frequencies in x and y directions, respectively, and F(x,y) is the 2D spectrum of f(x,y).

Aperiodic, discrete signal, continuous, periodic spectrum

$$F(u, v) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} f[m, n] e^{-j2\pi(umx_0 + vny_0)}$$

$$f[m, n] = \frac{1}{UV} \int_{0}^{U} \int_{0}^{V} F(u,v)\, e^{j2\pi(umx_0 + vny_0)}\, du\, dv$$

Where X and Y are periods of the signal in x and y directions, respectively, and $u_0 = \frac{1}{X}$ and $v_0 = 1/Y$ are the intervals between consecutive samples in the spectrum F[k,l].

2.8 Applications in Sharpening, Blurring and Noise Removal

The enhancement of images through the use of sharpening and noise reduction processes, which need some form of neighborhood processing, is another popular application of image processing.

Blurring An Image-Low Pass Filtering

Convolution in the spatial domain is identical to simple multiplication in the frequency domain, which is one of the most essential features of Fourier Transforms. The -convolve option blurs an image in the spatial domain by using small, square-sized, basic convolution filters (kernels). A low pass filter is what it's called. The most basic filter is a square array with equal weights. One of the most important aspects of Fourier Transforms is that convolution in the spatial domain is similar to simple multiplication in the frequency domain. Using modest, square-sized fundamental convolution filters, the -convolve option blurs an image in the spatial domain (kernels). It's referred to as a low pass filter. A square array with equal weights is the most basic filter. This is the same as a neighborhood or local average. The Gaussian-weighted, circularly shaped filter offered by either -gaussian-blur or -blur is another low pass filter.

A constant intensity white circle encircled by black is one sort of low pass blurring filter in the frequency domain. This filter corresponds to the magnitude (or real) component, and there is no (or a zero) phase (or imaginary) component. In the spatial domain, this filter would be analogous to a circularly shaped averaging convolution filter. Because convolution in the spatial domain is identical to multiplication in the frequency domain, all we have to do is forward Fourier transform the image, multiply the filter with the magnitude image, and then inverse Fourier transform the product. We can see that in the frequency domain, a tiny convolution filter corresponds to a huge circle. Multiplication is done with a -compose multiply setting and -composite.

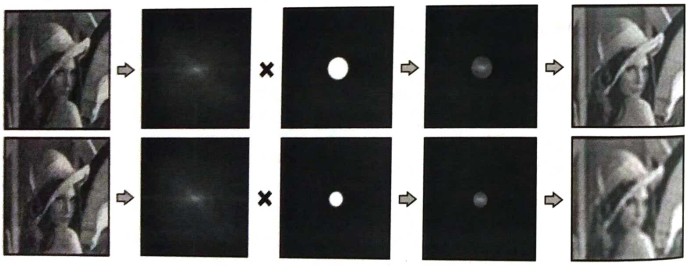


Fig. 2.8.1 Convolution filter corresponds to a huge circle

So let's try it with two different sizes of circular filters, one with a diameter of 48 (radius 24) and the other with a diameter of 32 (radius 32). (radius 16). We show both the spectrum and the masked spectrum, however the blurred effect is created using the masked magnitude. As a result, we can see that the image with the smaller diameter filter

has more blurring. In the generated photos, we also notice a 'ringing' or 'ripple' effect at the edges. This happens because, as we learned earlier, the Fourier Transform of a circle is a jinc function, which has decreasing oscillations as it moves away from the center. The jinc function and oscillations, on the other hand, are in the spatial domain rather than the frequency domain, as we saw before.

Sharpening An Image - High Boost Filtering

The simplest technique to sharpen an image is to apply a high pass filter to it (without stretching it) and then mix it with the original. In this scenario, the high pass filtering is performed in the frequency domain, with the output being translated back to the spatial domain and blended with the original image.

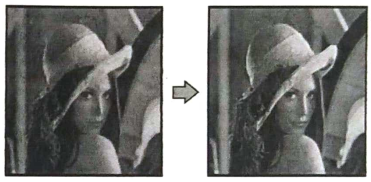


Fig. 2.8.2 Sharpen an image is to apply a high pass filter

Noise Removal - Notch Filtering

Patterned noise can be found in a lot of noisy photos. In the frequency domain, this type of noise is easier to remove because the patterns appear as a pattern of a few dots or lines. Remember that a basic sine wave is a repeating pattern that appears in the spectrum as simply three dots. To get rid of the noise, all one has to do is manually mask (or notch) away the dots or lines in the magnitude image. This is accomplished by translating to the frequency domain, creating a grayscale version of the spectrum, masking the dots or lines, thresholding it, multiplying the binary mask image with the magnitude image, and then transforming back to the spatial domain.

Let's try it on the clown image, which has a dither-like diagonally striped pattern. We start by transforming the clown image into magnitude and phase images.

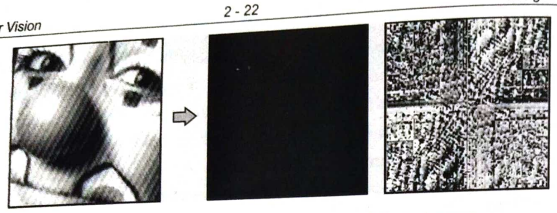


Fig. 2.8.3 Clown image into magnitude and phase images

Following that, we construct the spectrum image from the magnitude, but we add an extra step to push any image values at gray level zero to gray level 1, so that gray level zero can be preserved for the masked areas. Spectrum, a bash script, is also available to help with this. The magnitude image can be used as input, or the original image may be provided and it will be converted to the magnitude before generating the spectrum. In fact, the script does an adequate job of estimating the log scaling constant, so in fact, you may leave off the -s argument. We can detect four bright star-like spots in the spectrum, one in each quadrant. The bright dot and lines in the middle are ignored since they represent the image's DC or zero frequency components, i.e. the constant intensity regions. So we'll have to mask out those star-like spots immediately. We could open this image in ImageMagick and measure the locations and sizes of the star-like spots using the display functions, then use -draw to fill them with black circles or rectangles. However, it is easier to perform this interactively with a tool like GIMP or Photoshop and then threshold the output to a binary picture with ImageMagick. The outcomes are displayed below.

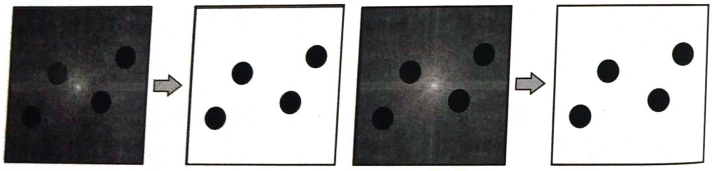


Fig. 2.8.4 Binary picture with ImageMagick

You can even color in the mask before removing it. Now we just multiply the mask by the magnitude and translate back to the spatial domain using the result and the phase image. A Gaussian shaped taper, as we did earlier, or even a simple linear taper of 5 pixels can be used. It's worth noting that -blur radiusx65000 will result in a linear taper. Using the script fft filter, both of the examples above become much easier.

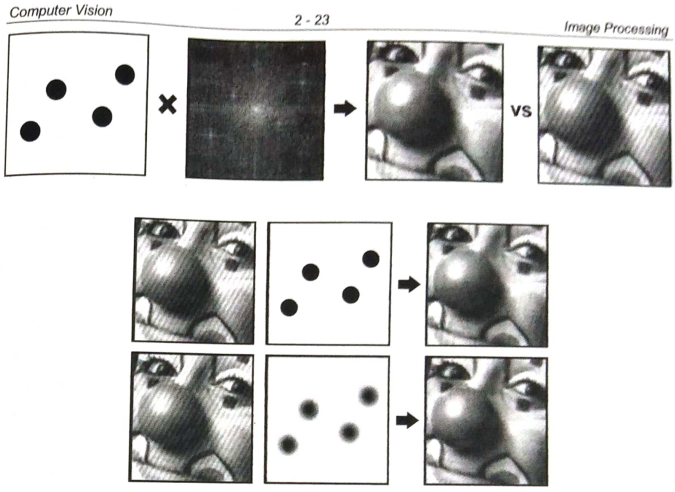


Fig. 2.8.5 Gaussian shaped taper

❑❑❑