Table of Contents

Image Processing 1
Commonly Used Operators 1
Linear Filtering 2
Separable Filtering 3
Band-Pass and Steerable Filters 4
Integral Images 5
Non-Linear Filtering 5
Fourier Transforms 6
Properties of Fourier Transforms 7
Fourier Transform Pairs 9
2D Fourier Transforms 10
Discrete Cosine Transform 10
Wiener Filtering 10

Image Processing

Image processing operators take one (or more) input image and produce an output image: $g(\vec{x})=h(f(\vec{x}))$, where g is the output image and f is the input image.

Commonly Used Operators

- **Gain and bias**: $g(\vec{x})=a\,f(\vec{x})+b$, where a and b are gain and bias parameters, respectively (note that gain and bias need not be constants, and thus $g(\vec{x})=a(\vec{x})\,f(\vec{x})+b(\vec{x})$).
- **Linear blend**: $g(\vec{x})=(1-\alpha)\,f_0(\vec{x})+\alpha\,f_1(\vec{x})$, where $\alpha\in[0,1]$.
- **Gamma correction**: $g(\vec{x})=[f(\vec{x})]^{\frac{1}{\gamma}}$, is a non-linear operator useful for removing the non-linear mapping between input radiance and quantized pixel values (radiance is the quantity of radiation that is emitted from a surface and reaches the sensor). Usually, $\gamma\approx 2.2$ is an adequate value.
- **Histogram equalization**: In this technique, we plot the histogram of individual color channels and luminance values, from which relevant statistics can be computed and used to enhance images, by way of histogram equalization. First we need to find an intensity mapping function $f(I)$ such that the resulting histogram is flat. To do this, we integrate the distribution $h(I)$ to obtain the cumulative distribution $c(I)$:

$$c(I)=\frac{1}{N}\sum_{i=0}^{I}h(i)$$

where N is the number of pixels in the image. Then, for any given intensity, we can look up its corresponding percentile $c(I)$ and determine the final value that pixels should have (apply $f(I)=c(I)$ with $c\in[0,255]$). This technique can be used locally in the image but care must be exercised in order not to introduce brightness discontinuities).

Linear Filtering

Linear filtering operators involve weighted combinations of pixels in small neighborhoods. A linear filter is defined as:

$$g(i,j)=\sum_{k,l} f(i+k,j+l)h(k,l)$$

where the entries in the kernel $h(k,l)$ are often called filter coefficients. The above formulation is known as a correlation operator. A common variant of this formula is:

$$g(i,j)=\sum_{k,l} f(i-k,j-l)h(k,l)=\sum_{k,l} f(k,l)h(i-k,j-l)$$

and is known as the convolution operator, where h is called the impulse response function. When the kernel function is convolved with a Dirac delta function, the kernel simply reproduces itself: $h*\delta=h$.

Both correlation and convolution operators are known as Linear Shift Invariant (LSI) operators. Linearity means that the relationship between the input and the output of the system is a linear function. That is to say, if input $f_1(x)$ produces response $g_1(x)$ and input $f_2(x)$ produces response $g_2(x)$, then the scaled and summed input $a_1 f_1(x)+a_2 f_2(x)=a_1 g_1(x)+a_2 g_2(x)$, where a_1 and a_2 are real constants. This can be extended to any number of terms such that for $a_1, a_2, \ldots, a_n$ the input

$$\sum_{i=1}^{n} a_i f_i(x)$$

produces the output

$$\sum_{i=1}^{n} a_i g_i(x)$$

In particular, we have that input function

$$\int_{-\infty}^{\infty} a_w f_w(x)\,dx$$

produces output

$$\int_{-\infty}^{\infty} a_w g_w(x)dx$$

Separable Filtering

If a kernel function $h(x,y)$ can be expressed as $h_1(x)h_2(y)$, then an input image can be convolved with the kernel in an efficient manner. For instance, if a mask for the kernel is chosen to have size $m \times m$ and the kernel is not separable, then the time complexity for an image convolution is given by $O(n_1 n_2 m^2)$ where $n_1 \times n_2$ is the size of the input image. Conversely, if the kernel is separable, then the time complexity of the convolution becomes $O(n_1 n_2 m)$, a profound difference.

Here are some examples of simple, separable, linear filters:

- The averaging (or box) filter:

$$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

  expressed in separable form as:

$$\frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \end{bmatrix}$$

- The tent filter:

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

  expressed in separable form as:

$$\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

- Convolution of the tent filter with itself yields the cubic approximating spline which is called the Gaussian kernel.

- Approximate Gaussian kernels can be obtained by successive convolutions of the box filter.

- These filters are called low-pass filters because they attenuate high frequencies and let lower ones pass through.

Smoothing kernels may also be used to sharpen images. For example, since blurring the image reduces its high-frequency contents, adding some of the difference between the original and blurred image makes it sharper:

$$g(\vec{x}) = f(\vec{x}) + \gamma(f(\vec{x}) - h(k,l) * f(\vec{x}))$$

where $h(k,l)$ is a low-pass (smoothing) kernel, $f(\vec{x})$ is the input image, γ is a real number within $[0,1]$, and $g(\vec{x})$ is the output image.

Band-Pass and Steerable Filters

More sophisticated kernels are created by first smoothing images with a unit-area kernel

$$G(\vec{x};\sigma) = \frac{1}{2\pi\sigma^2}\exp\left\{\frac{-(x^2+y^2)}{2\sigma^2}\right\}$$

and then taking its first or second order derivatives. Such filters are known as band-pass filters since they filter out some low and high-frequency contents while letting a particular band pass.

The undirected second-order derivative of a 2D image is obtained with the Laplacian operator:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Blurring an image with a Gaussian and then taking its Laplacian is equivalent to convolving it directly with a Laplacian of a Gaussian (LoG) filter:

$$\nabla^2 G(\vec{x};\sigma) = \left(\frac{x^2+y^2}{\sigma^4} - \frac{2}{\sigma^2}\right) G(\vec{x};\sigma)$$

Likewise, an oriented filter may be derived by first smoothing an image with a Gaussian and then taking a directional derivative

$$\nabla \hat{u} = \frac{\partial}{\partial \hat{u}}$$

which is obtained by taking the dot product between the gradient field ∇ and a unit direction $\hat{u} = (\cos\theta, \sin\theta)$:

$$\hat{u} \cdot \nabla (G * f) = \nabla \hat{u} (G * f) = (\nabla \hat{u} G) * f$$

Consequently, the smoothed directional derivative filter is:

$$G_{\hat{u}} = u G_x + v G_y = u \frac{\partial G}{\partial x} + v \frac{\partial G}{\partial y}$$

where $\hat{u} = (u, v) = (\cos\theta, \sin\theta)$.

Integral Images

If an image is going to be repeatedly convolved with different averaging filters, we can pre-compute the summed area table, which is simply the running sum of all the pixel values from the origin of the image:

$$s(i, j) = \sum_{k=0}^{i} \sum_{l=0}^{j} f(k, l)$$

This running sum can be efficiently implemented with a recursive raster-scan algorithm:

$$s(i, j) = s(i-1, j) + s(i, j-1) - s(i-1, j-1) + f(i, j)$$

To find the sum area (integral) inside a rectangular sub-image $[i_0, i_1] \times [j_0, j_1]$, we simply combine four samples from the summed-area table in the following way:

$$s(i_0, \ldots, i_1, j_0, \ldots, j_1) = \sum_{i=i_0}^{i_1} \sum_{j=j_0}^{j_1} s(i_1, j_1) - s(i_1, j_0 - 1) - s(i_0 - 1, j_1) + s(i_0 - 1, j_0 - 1)$$

Non-Linear Filtering

Sometimes, linear filters do not yield adequate results and one must resort to the use of non-linear filters. A prime example of this is the elimination of shot noise (also known as salt and pepper noise) with a median filter. A median filter selects the median value from each pixel's neighborhood. Since the shot noise values usually lie well outside the correct values in the neighborhood, the median filter is able to eliminate this type of noise.

There is also the α -trimmed mean filter which computes the mean of subtending pixels, while excluding the α smallest and largest values.

Another option is to compute a weighted median in which each pixel is used a number of times depending on its distance from the center. This is equivalent to minimizing the weighted objective function:

$$\sum_{k,l} w(k,l)|f(i+k,j+l)-g(i,j)|^p$$

where $g(i,j)$ is the desired output value and $p=1$ for the weighted median. When $p=2$, then the function computes the weighted mean, which is equivalent to correlation after normalizing by the sum of the weights.

Alternatively, we can combine weighted filter kernels with efficient outlier rejection mechanisms. In the bilateral filter the output pixel value depends on a weighted combination of neighboring pixel values:

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

The weighted coefficients $w(i,j,k,l)$ depend on the product of a domain kernel

$$d(i,j,k,l)=\exp\left\{-\frac{(i-k)^2+(j-l)^2}{2\sigma_d^2}\right\}$$

and a data-dependent range kernel

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

When multiplied together, they yield the data-dependent bilateral weight function:

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

Fourier Transforms

Fourier transforms provide a means to analyze what a given filter does to the frequency content of images. As a starting point, we can imagine that we pass a sinusoid of known frequency through the filter and determine how much it is attenuated. Consider

$$s(x)=\sin(\omega x+\phi_i)$$

where $\omega=2\pi f$ is the angular frequency, f is the frequency, and ϕ_i is the phase. If we convolve the sinusoidal signal $s(x)$ with a filter that has impulse response $h(x)$ then we get a sinusoid of identical frequency but with a different amplitude and phase:

$$o(x)=h(x)*s(x)=A\sin(\omega x+\phi_o)$$

The magnitude A is called the gain of the filter and $\phi_o-\phi_i$ is its phase shift. If we use a complex-valued sinusoid such as:

$$s(x)=e^{i\omega x}=\cos(\omega x)+i\sin(\omega x)$$

Then, the previous convolution becomes:

$$o(x)=h(x)*s(x)=A\,e^{i\omega x+\phi}$$

In that sense, the Fourier Transform is simply a tabulation of the magnitude and phase response at each frequency:

$$H(\omega)=F[h(x)]=Ae^{i\phi}$$

We denote a Fourier Transform pair as:

$$h(x)\Leftrightarrow H(\omega)$$

In the continuous domain, the Fourier Transform is written as:

$$H(\omega)=\int_{-\infty}^{\infty}h(x)e^{-i\omega x}\,dx$$

and, in the discrete domain, as:

$$H(k)=\frac{1}{N}\sum_{x=0}^{N-1}h(x)e^{-i\frac{2\pi kx}{N}}$$

where N is the number of samples in the signal.

Properties of Fourier Transforms

- **Superposition**: The Fourier Transform of a sum of signals is the sum of their Fourier Transforms

- **Shift**: The Fourier Transform of a shifted signal is the transform of the original signal multiplied by a linear phase shift (complex sinusoid)

- **Reversal**: The Fourier Transform of a reversed signal is the complex conjugate of the transform of the signal

- **Convolution**: The Fourier Transform of a pair of convolved signals is the product of their transforms

- **Correlation**: The Fourier Transform of a correlation is the product of the first transform with the complex conjugate of the second one

- **Multiplication**: The Fourier Transform of the product of two signals is the convolution of their transforms

- **Differentiation**: The Fourier Transform of the derivative of a signal is the transform of the signal multiplied by the frequency

- **Domain scaling**: The Fourier Transform of a stretched signal is the equivalently compressed and scaled version of the original transform

- **Real Images**: The Fourier Transform of a real-valued signal is symmetric around the origin

- **Parseval's Theorem**: The energy (sum of squared values) of a signal is the same as the energy of its Fourier transform

Property	Signal	Transform
Superposition	$f_1(x)+f_2(x)$	$F_1(\omega)+F_2(\omega)$
Shift	$f(x-x_0)$	$F(\omega)e^{-i\omega x_0}$
Reversal	$f(-x)$	$F^*(\omega)$

Convolution	$f(x)*h(x)$	$F(\omega)H(\omega)$
Correlation	$f(x)\operatorname{corr}h(x)$	$F(\omega)H^*(\omega)$
Multiplication	$f(x)h(x)$	$F(\omega)*H(\omega)$
Differentiation	$f'(x)$	$i\omega F(\omega)$
Domain Scaling	$f(ax)$	$\frac{1}{a}F\left(\frac{\omega}{a}\right)$
Real-valued Signal	$f(x)=f^*(x)$	$F(\omega)=F(-\omega)$
Parseval's Theorem	$\sum_x f(x)^2$	$\sum_\omega F(\omega)^2$

Fourier Transform Pairs

Signal	Transform
Impulse: $\delta(x)$	1
Shifted Impulse: $\delta(x-x_0)$	$e^{-i\omega x_0}$
Box: $\operatorname{box}\left(\frac{x}{a}\right)$	$a\operatorname{sinc}(a\omega)$
Tent: $\operatorname{tent}\left(\frac{x}{a}\right)$	$a\operatorname{sinc}^2(a\omega)$
Gaussian: $G(x;\sigma)$	$\frac{\sqrt{2\pi}}{\sigma}G(\omega;\sigma^{-1})$
LoG: $\left(\frac{x^2}{\sigma^4}-\frac{1}{\sigma^2}\right)G(x;\sigma)$	$\frac{-\sqrt{2\pi}}{\sigma}\omega^2 G(\omega;\sigma^{-1})$
Gabor: $\cos(\omega_0 x)G(x;\sigma)$	$\frac{\sqrt{2\pi}}{\sigma}G(\omega\pm\omega_0;\sigma^{-1})$
Unsharp Mask: $(1-\gamma)\delta(x)-\gamma G(x;\sigma)$	$(1-\gamma)-\frac{\sqrt{2\pi}}{\sigma}G(\omega;\sigma^{-1})$

- $\operatorname{box}(x)=\begin{cases}1 & \text{if } |x|\le 1\\ 0 & \text{otherwise}\end{cases}$
- $\operatorname{sinc}(\omega)=\sin\frac{(\omega)}{\omega}$
- $\operatorname{tent}(x)=\max(0,1-|x|)$
- $G(x;\sigma)=\frac{1}{\sqrt{2\pi}\sigma}\exp\left(\frac{-x^2}{2}\sigma^2\right)$

2D Fourier Transforms

The extension from 1D Fourier Transforms to 2D Fourier Transforms is trivial and the continuous 2D transform is written as:

$$H(\omega_x, \omega_y) = \iint h(x, y) e^{-i(\omega_x x + \omega_y y)} dx\, dy$$

The discrete Fourier transforms easily follows:

$$H(k_x, k_y) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} h(x, y) \exp\left(-i 2\pi \frac{k_x x + k_y y}{MN}\right)$$

where M and N are the width and height of the signal.

Discrete Cosine Transform

The Discrete Cosine Transform is a variant of the Fourier Transform often used to compress images in block-wise fashion. The 1D Cosine Transform is computed by taking the dot product of each N-wide block of pixels with a set of cosines of different frequencies:

$$F(k) = \sum_{i=0}^{N-1} \cos\left(\frac{\pi}{N}\left(i + \frac{1}{2}\right)k\right) f(i)$$

where k is the frequency index, and the one-half pixel offset is used to make the basis coefficients symmetric. The 2D version of the Discrete Cosine Transform is defined as:

$$F(k, l) = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \cos\left(\frac{\pi}{N}\left(i + \frac{1}{2}\right)k\right) \cos\left(\frac{\pi}{N}\left(j + \frac{1}{2}\right)l\right) f(i, j)$$

Just as with the Fast Fourier Transform (FFT), the 2D Discrete Cosine Transform is separable.

Wiener Filtering

Signal spectra capture a first-order description of spatial statistics. In particular, we can assume that an image is a sample from a correlated Gaussian random noise field combined with a statistical model of the measurement process.

To derive the Wiener filter, we analyze each frequency component from the Fourier Transform of a signal independently. The image formation process may

be written as:

$$o(x,y)=s(x,y)+n(x,y)$$

where $s(x,y)$ is the noiseless image (that we are trying to recover), $n(x,y)$ is the additive noise signal, and $o(x,y)$ is the observed noisy image. In the frequency domain, this can be written as:

$$O(\omega_x,\omega_y)=S(\omega_x,\omega_y)+N(\omega_x,\omega_y)$$

At each frequency (ω_x,ω_y) , we know from the image spectrum that the unknown transform component $S(\omega_x,\omega_y)$ has a prior distribution, or a way of describing the uncertainty before considering the image data itself.

To find this prior distribution, we assume that an image is a random noise field whose expected (mean) magnitude at each frequency is given by its power spectrum:

$$\langle S(\omega_x,\omega_y)^2\rangle=P_s(\omega_x,\omega_y)$$

Then we can say that the unknown transform component $S(\omega_x,\omega_y)$ has a prior distribution which is a zero-mean Gaussian with variance $P_s(\omega_x,\omega_y)$. There is also the noisy measurement $O(\omega_x,\omega_y)$ whose variance is $P_n(\omega_x,\omega_y)$ (the power spectrum of the noise) which is assumed to be constant: $P_n(\omega_x,\omega_y)=\sigma_n^2$. With Baye's rule, then we can write the posterior estimate of S as:

$$p(S|O)=\frac{p(O|S)\,p(S)}{\int_S p(O|S)\,p(S)}$$

where the denominator is used simply to make the distribution integrate to 1, as it should.

The prior distribution $p(S)$ is given by:

$$p(S)=\exp\left(\frac{-(S-\mu)^2}{2P_s}\right)$$

where μ is the expected mean at that frequency. The measurement distribution $p(O|S)$ is given by:

$$p(O|S)=\exp\left(\frac{-(S-O)^2}{2P_n}\right)$$

Taking the negative logarithm on both sides of the posterior estimate of S and setting $\mu = 0$ for simplicity, we obtain:

$$\begin{aligned} -\log p(S|O) &= -\log p(O|S) - \log p(S) + C \\ &= \frac{1}{2} P_n^{-1} (S-O)^2 + \frac{1}{2} P_s^{-1} S^2 + C \end{aligned}$$

which is the negative posterior log likelihood. The minimum of this quantity is obtained as:

$$S^* = \left(\frac{1}{1 + \frac{P_n}{P_s}} \right) O$$

and

$$W(\omega_x, \omega_y) = \frac{1}{1 + \frac{\sigma_n^2}{P_s(\omega_x, \omega_y)}}$$

is the Fourier Transform of the optimal Wiener filter to remove the noise from an image that has power spectrum $P_s(\omega_x, \omega_y)$. This method of deriving the filter can be extended to the case where the observed image is a noisy, blurred version of the ideal noiseless image:

$$o(x, y) = b(x, y) * s(x, y) + n(x, y)$$

where $b(x, y)$ is a known blurring kernel.