FOURIER TRANSFORM

An image \( f(h, k) \) can be represented by its frequency components as follows

\[
f(h, k) = \frac{1}{4\pi^2} \int_{-\pi}^{\pi} \int_{-\pi}^{\pi} F(u, v) e^{ihu} e^{jkv} dudv \tag{1}
\]

\( F(u, v) \) is the Fourier transform of the image. It is given by

\[
F(u, v) = \frac{1}{MN} \sum_{h=1}^{M} \sum_{k=1}^{N} f(h, k) e^{-ihu} e^{-jkv} \tag{2}
\]

- \( F(u, v) \) is a continuous function and \( j^2 = -1 \).
- The values near the origin in the \((u, v)\) plane are the low frequency components.
- The values distant from the origin are the high frequency components.
- \( F(u, v) \) is a complex function, i.e., it has a magnitude and a phase.
- \( F(0, 0) \) is called the DC component.
- From equation (2), \( F(0, 0) \) is the average value of the image pixels.
- In the frequency domain, filtering becomes a simple multiplication. Thus it is not a problem to use large filters, unlike the space domain where large masks can be computationally expensive.
- Filtering in the frequency domain becomes a multiplication, that is

\[
G(u, v) = H(u, v) \times F(u, v) \tag{3}
\]

where \( G, H \) and \( F \) are the Fourier transforms of \( g \) (resulting image), \( h \) (filter), and \( f \) (original image), respectively. A block diagram of filtering in the frequency domain is shown in figure 1.

- Usually the magnitude in the Fourier transform is more important since it contains more information. However, in order to preserve all the information in the image we need to use both magnitude and phase.
- The Fourier transform can be used in many applications including image enhancement and compression.
- For linear filtering operations, the space domain is more useful than the frequency domain. However, sometimes it is easier to process the image in the frequency domain. For example, frequency domain makes large filtering operations faster.
- The range of the magnitude in the Fourier transform can be large for this reason, we use \( \sqrt{F(u, v)} \) or \( \log(|F(u, v)|) \) to visualize the magnitude.
- Every pixel in the Fourier transform is a frequency value.
- Matlab command \texttt{fft2} can be used to obtain the Fourier transform. Matlab command \texttt{fftsift} allows to move the DC (zero frequency component) to the center as shown in figure 2. Commands \texttt{fft2} and \texttt{fftsift} are illustrated in figure 3.
- Matlab command \texttt{ifft2} allows to obtain the inverse Fourier transform. An image and its FFT are shown in figure 4 top, the Fourier transforms of low pass and high pass filters are also shown in figure 4.

I. DISCRETE COSINE TRANSFORM

The discrete cosine transform is closely related to the Fourier transform. Here, the transformation is a linear combination of cosine functions. That is, the basis function is a
cosine function instead of the exponential function.

There are 4 types of DCT based on the boundary conditions; DCT II is the most commonly used. Assume $M = N$ (square image), the DCT is defined as

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

(4)

The normalization constant is given by

$$c(k) = \begin{cases} \frac{1}{\sqrt{2}} & \text{for } k = 0 \\ 0 & \text{otherwise} \end{cases}$$

(5)

What is the most obvious difference between DCT and DFT? DCT is real while DFT is complex.

DCT II is widely used in image compression. High frequency coefficients are less significant than the DC term and some low frequency coefficients. DCT allows to place the significant and nonsignificant components as follows:

- The most important values to the human eye are placed in the upper left corner of the matrix.
- The least significant values are placed mostly in the lower right corner of the matrix. This is illustrated in figure 5.

JPEG image compression method is based on the DCT. An example of the application of DCT is shown in figures 6 and 7, where figure 6 shows the original image and 7 shows the corresponding DCT.

II. WAVELET TRANSFORM

In the Fourier transform and the DCT, the frequency domain is decoupled from the spatial domain, you know the frequencies present in an image but you do not know their coordinates. Wavelet transforms solves this problem partially. The basis here is not a sine or a cosine, it is a function called wavelet.
The 1D continuous wavelet is given by

$$c(s, \tau) = \int_{R} f(t) \psi_{s,\tau}^{*}(t) d\tau$$  \hspace{1cm} (6)$$

where

- superstar $^*$ means complex conjugate
- $\psi^*$: mother wavelet.
- $s \in \mathbb{R}^+ - \{0\}, \tau \in \mathbb{R}$.

Note that $c(t)$ is a function of $s$ and $\tau$. Wavelets are generated from $\psi^*$ as follows

$$\psi(t)_{s,\tau} = \frac{1}{\sqrt{s}} \psi(\frac{t - \tau}{s})$$  \hspace{1cm} (7)$$

where

- $\frac{1}{\sqrt{s}}$ is a normalization factor
- $\tau$ is a shift in time
- $s$ is a change in the scale. Large values mean long wavelength

A. Example:

Shannon wavelet

$$\psi(t) = 2 \text{sinc}(2t) - \text{sinc}(t)$$  \hspace{1cm} (8)$$

An illustration of the Shannon wavelet for $s = 3, \tau = 4$ is shown in figure 8. The inverse wavelet transform is given by

$$f(t) = \int \int c(s, \tau) \psi_{s,\tau}(t) d\tau ds$$  \hspace{1cm} (9)$$

where

- $f(t)$ : the time series
- $\psi_{s,\tau}(t)$ : wavelet with scale $s$ and time $\tau$.

The determination of the wavelet coefficients at a fixed scale is a convolution operation, that is

$$c(s, \tau) = \int f(t) \psi((t - \tau)/s)$$  \hspace{1cm} (10)$$

Not any function can be chosen to be a basis function for the wavelet. Wavelets is a very active research topic in machine vision. It has many applications including image compression and object recognition.

**EDGE DETECTION**

Edges are among the most important features in an image. They characterize the boundary between different objects or regions. An edge can be defined as a significant local change in the intensity level. There exist different 1 D profile models for edges:

- step: the gray level jumps abruptly from one value to another one.
- ramp: the gray level changes gradually from one value to another one.
- line: the gray level jumps abruptly from one value to another one and goes back to the initial gray level
- roof: the gray level changes gradually to a value and then back to the initial value.
The four models are shown in figure 9. Line and step edges are not very common in digital images. Many digital cameras perform smoothing operations.

Since edges are defined as a significant change in the gray level, we need an operator that is sensitive to change to detect edges. This operator is the derivative. Since an image is a function of two variables, we use partial derivatives.

**B. Gradient**

Let \( f(x, y) \) be a 2D function, its gradient is given by:

\[
\nabla f(x, y) = \left[ \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right]
\]  

(11)

An important property of the gradient is that it is a vector, i.e., it has a magnitude and a phase. We can also use the second derivative for edge detection. Consider figures 10 and 11, we have

- \( f \) shows a change in the gray level.
- In the first derivative an edge corresponds to a peak.
- In the second derivative, an edge corresponds to zero crossing.

Since images are discrete data, the continuous gradient cannot be used, an approximation of the gradient is necessary to perform edge detection in digital images.

**C. First and second derivative approximation**

Using the definition of the derivative, we can write

\[
f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}
\]

(12)

If we take \( h = 1 \), we get

\[
f'(x) \simeq f(x + 1) - f(x)
\]

(13)

and

\[
f''(x) \simeq f(x + 2) - 2f(x + 1) + f(x)
\]

(14)

or

\[
f''(x) \simeq f(x + 1) - 2f(x) + f(x - 1)
\]

(15)

for the second derivative. The gradient is a vector: it has a magnitude and a phase. Its magnitude is given by:

\[
|\nabla f(x, y)| = \sqrt{\left( \frac{\partial f}{\partial x} \right)^2 + \left( \frac{\partial f}{\partial y} \right)^2}
\]

(16)

and can be approximated by

\[
|\nabla f(x, y)| \simeq \left| \frac{\partial f}{\partial x} \right| + \left| \frac{\partial f}{\partial y} \right|
\]

(17)

\[
|\nabla f(x, y)| \simeq \max \left\{ \left| \frac{\partial f}{\partial x} \right|, \left| \frac{\partial f}{\partial y} \right| \right\}
\]

(18)

The phase of the gradient is

\[
\alpha = \tan^{-1} \left( \frac{\frac{\partial f}{\partial y}}{\frac{\partial f}{\partial x}} \right)
\]

(19)

The phase and the magnitude characterize the direction and the strength of the edge, respectively. The angle obtained by equation (19) is perpendicular to the edge direction, for example \( \alpha = 0 \) corresponds to a vertical. Recall that

\[
f' = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h}
\]

(20)

For \( h = 1 \):

\[
f' = f(x+1) - f(x)
\]

(21)

We use the difference instead of the derivative. For the partial derivatives, this translates to:

\[
\frac{\partial f}{\partial x} = \nabla_x f = f(x + 1, y) - f(x, y)
\]

(22)

and

\[
\frac{\partial f}{\partial y} = \nabla_y f = f(x, y + 1) - f(x, y)
\]

(23)

Note that:

- \( j \) corresponds to the \( x \)-direction.
- \( i \) corresponds to the negative \( y \)-direction.
This is illustrated in figure 12. Noting the relationship between 
(i, j) and (x, y), the gradient can be approximated by the 
difference as follows

\[ \nabla_x f = f(i, j + 1) - f(i, j) \]  
\[ \nabla_y f = f(i, j) - f(i + 1, j) \]

We have \( A - B \) (see figure 12) because \( i \) is the opposite 
direction of \( y \), that is \( f(i, j) \sim f(x, y+1) \) and \( f(i+1, j) \sim f(x, y) \).
The gradient approximated by these difference equations has 
the following mask:

\[
\begin{bmatrix}
-1 & 1 \\
1 & -1
\end{bmatrix}
\]  
and
\[
\begin{bmatrix}
1 & 1 \\
1 & -1
\end{bmatrix}
\]

Recall that \( \nabla_x f, \nabla_y f \) are the directional components of 
the gradient, where:

- \( \nabla_x f \) is sensitive to the vertical edges.
- \( \nabla_y f \) is sensitive to the horizontal edges.

**D. Example**

Approximate the gradient at the middle pixel \( c_5 \) and 
determine the direction of the edge.

\[
C = \begin{bmatrix}
c_1 & c_2 & c_3 \\
c_4 & c_5 & c_6 \\
c_7 & c_8 & c_9
\end{bmatrix}
= \begin{bmatrix}
1 & 2 & 3 \\
2 & 2 & 2 \\
10 & 10 & 10
\end{bmatrix}
\]

\[
\nabla_x f = 2 - 2 \\n\nabla_y f = 2 - 10 \\n\nabla f = \sqrt{(c_5 - c_3)^2 + (c_5 - c_6)^2}
\]

Based on the values of \( \nabla_x f \) and \( \nabla_y f \), there is a horizontal 
edge. The direction of the edge is

\[
\text{atan2}(\nabla_y f, \nabla_x f) = -\pi/2
\]

which again corresponds to a horizontal edge. This can be 
confirmed from the image in (29).

**E. Another approximation**

It is possible to approximate the gradient using the diagonal 
elements instead of the direct neighbors:

\[
\nabla_x f = f(i, j) - f(i + 1, j + 1) \\n\nabla_y f = f(i + 1, j) - f(i, j + 1)
\]

The mask for this approximation is

\[
R_x = \begin{bmatrix}
1 & 0 \\
0 & -1
\end{bmatrix}
\]
\[
R_y = \begin{bmatrix}
0 & 1 \\
-1 & 0
\end{bmatrix}
\]

This mask is called the Roberts edge detector.

**F. Another approximation**

Another approximation that contains more neighbors is 
given by:

\[
\nabla_x f = f(i - 1, j + 1) + cf(i, j + 1) + f(i + 1, j + 1) \\n- f(i - 1, j - 1) - cf(i, j - 1) - f(i + 1, j - 1)
\]

and

\[
\nabla_y f = -f(i + 1, j - 1) - f(i + 1, j + 1) - cf(i + 1, j) \\n+ f(i - 1, j - 1) + cf(i - 1, j) + f(i - 1, j + 1)
\]

The mask for this approximation is as follows:

\[
M_x = \begin{bmatrix}
-1 & 0 & 1 \\
-c & 0 & c \\
-1 & 0 & 1
\end{bmatrix}
\]
\[
M_y = \begin{bmatrix}
1 & c & 1 \\
0 & 0 & 0 \\
-1 & -c & -1
\end{bmatrix}
\]

Two very important edge detectors are derived from \( M_x \) and 
\( M_y \). These are the Sobel and the Prewitt edge detectors.

**G. Sobel edge detector: c=2**

\[
S_x = \begin{bmatrix}
-1 & 0 & 1 \\
-2 & 0 & 2 \\
-1 & 0 & 1
\end{bmatrix}
\]
\[
S_y = \begin{bmatrix}
1 & 2 & 1 \\
0 & 0 & 0 \\
-1 & -2 & -1
\end{bmatrix}
\]

**H. Prewitt edge detector: c=1**

\[
P_x = \begin{bmatrix}
-1 & 0 & 1 \\
-1 & 0 & 1 \\
-1 & 0 & 1
\end{bmatrix}
\]
\[
P_y = \begin{bmatrix}
1 & 1 & 1 \\
0 & 0 & 0 \\
-1 & -1 & -1
\end{bmatrix}
\]
I. Example

Let

\[ A = \begin{bmatrix} 30 & 66 & 65 \\ 14 & 30 & 70 \\ 12 & 15 & 40 \end{bmatrix} \]

(46)

Use the Sobel and Prewitt edge detectors to calculate the magnitude and direction of the edge. Compare. The image corresponding to (46) is shown in figure 13.

- Using the Sobel mask:

  \[ \nabla_x f = 175 \]
  \[ \nabla_y f = 145 \]
  \[ \nabla f = 227.26 \]
  \[ \alpha = 39.64 \]

(47) \hspace{1cm} (48) \hspace{1cm} (49) \hspace{1cm} (50)

- Using the Prewitt mask:

  \[ \nabla_x f = 119 \]
  \[ \nabla_y f = 94 \]
  \[ \nabla f = 151.64 \]
  \[ \alpha = 38.30 \]

(51) \hspace{1cm} (52) \hspace{1cm} (53) \hspace{1cm} (54)

Low pass filter ⇒ Smoothing operation ⇒ Gaussian filter

High pass filter ⇒ Edge detection ⇒ Edge detectors