Advanced

# Image Processing and Stochastic Modeling

## TP3: Fourier

Prof. Sviatoslav Voloshynovskiy,
Olga Taran <olga.taran@unige.ch>,
Maurits Diephuis.

Stochastic Information Processing Group

April 5, 2017

UNIVERSITÉ DE GENÈVE
FACULTÉ DES SCIENCES
Département d'informatique

SIP stochastic information processing

## Submission

Please archive your report and codes in "Name_Surname.zip" (replace "Name" and "Surname" with your real name), and upload to "`Assignments/TP3: Fourier`" on `https://chamilo.unige.ch` before Thursday, May 3 2017, 23:59 PM. Note, **the assessment is mainly based on your report, which should include your answers to all questions and the experimental results**.

The Fourier series expresses periodic functions (or signals) as the sum of a possibly infinite set of simple oscillating functions, namely sinuses and cosines or complex exponentials.

The Discrete Fourier Transform (DFT) has a very fast implementation named the Fast Fourier Transform (FFT). Practically, you will do al your work with the FFT.

# 1 Introduction

Let there be a discrete sampled signal $f(x)$, with length $M$:

$$f(x) = (1, 2, 4, 5, 3, 0, \ldots, 7)$$

Although not noted explicitly, such a signal is represented with respect to a set of canonical basis vectors in $\mathbb{R}^M$. This means that the signal is made out from a weighted set of basis vectors:

$$\begin{aligned} f(x) = & \, 1(1, 0, 0, 0, 0, 0, \ldots, 0) \\ & + 2(0, 1, 0, 0, 0, 0, \ldots, 0) \\ & + 4(0, 0, 1, 0, 0, 0, \ldots, 0) \\ & + \ldots \\ & + 7(0, 0, 0, 0, 0, 0, \ldots, 1) \end{aligned}$$

In short:

$$f(x) = \sum_{k=0}^{M-1} a_k b_k(x)$$

where $b_k(x)$ are the canonical basis vectors and $a_k$ is defined as follows:

$$a_k = \sum_{l=0}^{M-1} f(l) b_k(l)$$

Hence, in linear algebra terms the weights $a_k$ are simply the inner product between the signal $f(x)$ and the corresponding basis vector $b_k(x)$.

A signal can be represented with more than set of basis vectors, and a particular powerful set is formed by the Fourier basis, which consists of sinusoids of varying frequency and phase:

$$\begin{aligned} f(x) = a_0 + a_1\cos(x) + a_2\cos(2x) + a_3\cos(3x) + \cdots \\ + b_1\sin(x) + b_2\sin(2x) + b_3\sin(3x) + \cdots \end{aligned}$$

This expansion is called the *trigonometric series* or the *Fourier series*.

## 1D Discrete Fourier Transform

The 1-dimensional Discrete Fourier Transform (DFT) for a discrete function $f(x)$, $x = 0, 1, \ldots, M-1$ is defined as:

$$\mathcal{F}(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(x)e^{\frac{-i2\pi ux}{M}}$$

where $u = 0, 1, \ldots, M-1$. Its inverse is defined as:

$$f(x) = \sum_{u=0}^{M-1} \mathcal{F}(u)e^{\frac{i2\pi ux}{M}}$$

Noting that $e^{i\theta} = \cos(\theta) + i\sin(\theta)$, the DFT can be refractored into:

$$\mathcal{F}(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(x)\Big\{\cos\Big(\frac{2\pi ux}{M}\Big) + i\sin\Big(\frac{2\pi ux}{M}\Big)\Big\}$$

Note that **each** term of the Fourier Transform, that is, the value of $\mathcal{F}(u)$ for **each** value of $u$, is composed of the **sum** of **all** values of the function $f(x)$. These values of $f(x)$ are in turn multiplied by sines and cosines of various frequencies.

### DFT Magnitude, Phase and Powerspectrum

As you can see, the components of the Discrete Fourier Transform are complex. It is therefore convenient to express the following properties in polar coordinates. Given:

$$\mathcal{F}(u) = |\mathcal{F}(u)|e^{-i\phi(u)}$$

The *magnitude* of the Fourier transform is defined as:

$$|\mathcal{F}(u)| = \sqrt{Re^2(u) + Im^2(u)}$$

The *phase angle* or *phase magnitude*:

$$\phi(u) = tan^{-1}\left(\frac{Im(u)}{Re(u)}\right)$$

The *power spectrum* is defined as the square of the magnitude:

$$\begin{aligned} P(u) &= |\mathcal{F}(u)|^2 \\ &= Re^2(u) + Im^2(u) \end{aligned}$$

**Exercise**

The first Fourier coefficient ($u = 0$) is called the DC component.

- Show that the DC component for a discrete signal $f(x)$ with zero mean is 0.

## The DFT matrix

Let there be a 1D discrete signal $f(x)$ of length $M$:

$$\mathbf{f} = [f_0, f_1, f_2, \ldots, f_{M-1}]$$

The DFT result $\mathcal{F}(u)$ of $f(x)$ is some discrete sequence:

$$\mathbf{F} = [\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2, \ldots, \mathcal{F}_{M-1}]$$

The DFT definition is:

$$\mathcal{F}(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) e^{\frac{-i2\pi ux}{M}}$$

This finite sum can be expressed as a matrix multiplication:

$$\mathbf{F} = \boldsymbol{\mathcal{W}}\mathbf{f}$$

where $\boldsymbol{\mathcal{W}}$ is a $M \times M$ matrix and $m \in \{1, \ldots, M\}$, whose coefficients are defined by:

$$\begin{aligned} \mathcal{W}_{m,m} &= \frac{1}{M} e^{\frac{-i2\pi mm}{M}} && \text{by definition} \\ &= \frac{1}{M} \omega^{mm} && \text{substituting:} \quad \omega = e^{\frac{-i2\pi}{M}} \end{aligned}$$

$\boldsymbol{\mathcal{W}}$ which is otherwise known as the *DFT matrix* can be then worked out as follows:

$$\boldsymbol{\mathcal{W}} = \frac{1}{M} \begin{pmatrix} 1 & 1 & 1 & 1 & \ldots & 1 \\ 1 & \omega^1 & \omega^2 & \omega^3 & \ldots & \omega^{M-1} \\ 1 & \omega^2 & \omega^4 & \omega^6 & \ldots & \omega^{2(M-1)} \\ 1 & \omega^3 & \omega^6 & \omega^9 & \ldots & \omega^{3(M-1)} \\ 1 & \omega^4 & \omega^8 & \omega^{12} & \ldots & \omega^{4(M-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 1 & \omega^{M-1} & \omega^{2(M-1)} & \omega^{3(M-1)} & \ldots & \omega^{(M-1)^2} \end{pmatrix}$$

**Example**

Suppose the 1D discrete function $f(x)$ is defined as follows:

$$\mathbf{f} = [f_1, f_2, f_3, f_4]$$

This means that $M = 4$ which is enough information to build up the DFT matrix $\boldsymbol{\mathcal{W}}$:

$$\begin{aligned}
\omega &= e^{\frac{-i2\pi}{M}} && \text{by definition} \\
&= e^{\frac{-i2\pi}{4}} && \text{substitution} \\
&= e^{\frac{-i\pi}{2}} \\
&= \cos\left(\frac{-\pi}{2}\right) + i\sin\left(\frac{-\pi}{2}\right) && e^{i\theta} = \cos(\theta) + i\sin(\theta) \\
&= -i
\end{aligned}$$

Matrix $\boldsymbol{\mathcal{W}}$ then becomes:

$$\boldsymbol{\mathcal{W}} = \frac{1}{4}\begin{pmatrix} 1 & 1 & 1 & 1 \\ 1 & -i & (-i)^2 & (-1)^3 \\ 1 & (-i)^2 & (-i)^4 & (-i)^6 \\ 1 & (-i)^3 & (-i)^6 & (-i)^9 \end{pmatrix}$$

**Exercise**

Let the 1D discrete function $f(x)$ be defined as follows: $\mathbf{f} = [5, 4, 3, 2, 1]$.

- Determine the Discrete Fourier Transform of $f(x)$ using the DFT matrix by hand.
- Verify your answer with Matlab `fft`.
- What happens if we double the signal in length? Let $\mathbf{f} = [5, 4, 3, 2, 1, 5, 4, 3, 2, 1]$.

## 2D Discrete Fourier Transform

The 1D DFT can be extended straightforwardly to the 2D case. The 2D DFT of a function $f(x, y)$ or an $M \times N$ image, where $u$ and $v$ are the frequency values and $x$ and $y$ the spacial or image values.

$$\mathcal{F}(u, v) = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x, y)e^{(-i2\pi(\frac{ux}{M} + \frac{vy}{N}))}$$

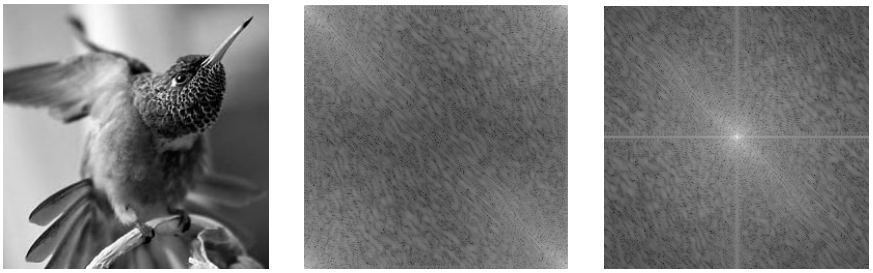
This expression must be evaluated for $u = 0, 1, 2, \cdots, M-1$ and $v = 0, 1, 2, \cdots, N-1$. Similarly, the inverse 2D DFT is defined as:

$$f(x, y) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} \mathcal{F}(u, v)e^{(i2\pi(\frac{ux}{M} + \frac{vy}{N}))}$$

Again, we emphasize that **each** term of $\mathcal{F}(u, v)$ contains **all** values of $f(x, y)$, modified by the multiplication of the exponential terms.

The *magnitude spectrum*, *phase* and *power spectrum* are defined as follows:

$$\begin{aligned}
|\mathcal{F}(u, v)| &= \sqrt{Re^2(u, v) + Im^2(u, v)} \\
\theta(u, v) &= tan^{-1}\Big(\frac{Im(u, v)}{Re(u, v)}\Big) \\
P(u, v) &= |\mathcal{F}(u, v)|^2
\end{aligned}$$



**(a)** **(b)** **(c)**

```
F = fft2(im);
figure;imshow(log(abs(F)+eps), []);
figure;imshow(log(abs(fftshift(F))+eps), []);
```

**(d)**

**Figure 1** – Examples of the *magnitude* of the Fourier Transform of a grayscale image and the shifting of $\mathcal{F}(0,0)$ to the center of the $M \times N$ image. To visualize the log of the magnitude has been taken and zero values in the magnitude have been increased with a tiny offset.

It is also important to note that the following holds:

$$\mathcal{F}(f(x,y)(-1)^{x+y}) = \mathcal{F}\Big(u - \frac{u-M}{2}, v - \frac{v-N}{2}\Big)$$

which states that the origin of the Fourier Transform of $[f(x,y)(-1)^{x+y}]$ (or $\mathcal{F}(0,0)$) is located at $u = \frac{M}{2}, v = \frac{N}{2}$. In other words, multiplying function $f(x,y)$ with $(-1)^{x+y}$ shifts the origin of $\mathcal{F}(u,v)$ to $(\frac{M}{2}, \frac{N}{2})$ which is the center of the $M \times N$ area this DFT occupies. In Matlab this can be done via `fftshift`. Note that $M$ and $N$ are integers and if they are not even, but odd, 1 should be added to the offset, i.e. $u = \frac{M}{2} + 1, v = \frac{N}{2} + 1$. See Figure 1.

## Image Phase and Magnitude

### Exercise

- Use the Fourier Transform to ascertain the *phase* and *magnitude* of the a grayscale image.
- Reconstruct the image via the inverse Fourier Transform twice. Once using only the phase and once using only the magnitude of the original image. For the inverse tangent, use Matlab `atan2`, which is the four-quadrant inverse tangent.
- What can you conclude from the two reconstructions.

### Multiplication and Convolution

Convolution in the time domain, equals multiplication in the Fourier domain:

$$\mathcal{F}\{f * g\} = \mathcal{F}\{f\} \cdot \mathcal{F}\{g\}$$
$$f * g = \mathcal{F}^{-1}\{\mathcal{F}\{f\} \cdot \mathcal{F}\{g\}\}$$

We can use this to determine the (auto) correlation between signals $f$ and $g$.

Formally, for (discrete) signals $f$ and $g$, of length $N$ it can be shown that :

$$\mathcal{F}^{-1}\{\mathcal{F}\{f\}^* \cdot \mathcal{F}\{g\}\}_n = (f \star g_N)_n$$

where $\star$ denotes cross-correlation and $\mathcal{F}\{f\}^*$ is the complex conjugate of $\mathcal{F}\{f\}$.

#### Exercise

- Implement a matlab function that determines the 2D (auto) correlation in the Fourier domain.
- Test your code on the image `cameraman` and on `cross.png` on `Chamilo`. Describe what you see.
- Implement the (auto) correlation again, but now in the spatial domain, using the Matlab function `xcorr2` or
  `c = conv2(a, rot90(conj(b),2));`.
- Measure the time differences it takes the Fourier domain version and the Spatial domain version to finish. This can be done with Matlab `tic`. Use the image `radcliffe.jpg` from `Dokeos`.

#### Hints

- Given images **f** and **g**, use `fft2(X,m,n)` where `m = size(f, 1) + size(g, 1)-1` and `m = size(f, 2) + size(g, 2)-1`.
- Use `fftshift` to set the DC component in the center.
- Use `abs` to obtain the magnitude of the final result.

## 2 Filtering in the Fourier Domain

#### Introduction

A 2D filter or transfer function is the Fourier domain is some function $H(u, v)$ that is deployed as follows:

$$G(u, v) = H(u, v). * \mathcal{F}(u, v)$$

where $\mathcal{F}(u, v)$ is the Fourier transform of some image or function $f(x, y)$. The final result in the spatial domain can be obtained running the inverse Fourier transform, i.e:

$$g(x, y) = \mathcal{F}^{-1}(G(u, v));$$

Now, as stated, in general the Fourier components are complex, yet the filters can be real. In this case, both the real and the imaginary components of $\mathcal{F}(u, v)$ are multiplied against the real component of the filter. This type of filtering is called *zero-phase-shift* filters as they don't modify the phase of the transform.

Although the proof will be omitted, the *convolution theorem* shows that filters in the spatial and in the frequency domain form a Fourier pair. Meaning one can take a spatial filter and obtain the corresponding filter in the frequency domain via the Fourier Transform and vise versa. Note that the Fourier Transform and its inverse are linear processes, so this principle only applies to linear filters.

Last but not least, why filter in the Fourier domain if one can do it in the spatial domain using (small) convolution masks as seen previously? The reason is two fold. To start, filters in the frequency domain are easily designed and can be understood intuitively. Secondly, and most important, they are much faster. Processing in the frequency domain becomes faster than the spatial domain for $N \times M$ images when $M, N > 32$.

### Example

Figure 2 shows an example of filtering in the Fourier domain using a filter that is basically a white circle mask. Although easily defined in the Fourier domain, one can see that this filter gives far from pleasing results.

### Gaussian Filter

An other easy example is formed by the Gaussian filter. Examples can be seen in Figure 3. A Gaussian low-pass filter in the spatial domain is defined as:

$$h(x) = \sqrt{2\pi}\sigma A e^{-2\pi^2\sigma^2 x^2}$$

and in the frequency domain:

$$H(u) = A^{(\frac{-u^2}{2\sigma^2})}$$

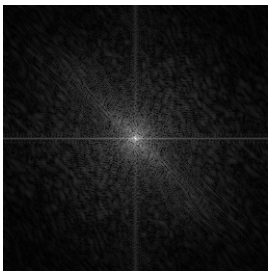
Alternatively, one can construct a high-pass filter using the so called *differences of Gaussian*:

$$H(u) = A^{(\frac{-u^2}{2\sigma_1^2})} - B^{(\frac{-u^2}{2\sigma_2^2})}$$

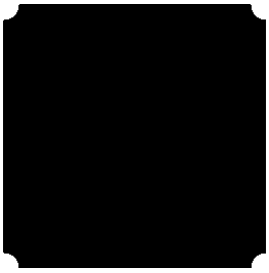
where $A > B$ and $\sigma_1 > \sigma_2$.

Note, in Figure 3, the reciprocal relationship between the filter defined in the filter domain and the one in the spatial domain. If filter $H(u)$ has a broad large profile (large value of $\sigma$), its spatial counterpart $h(x)$ has a narrow profile. In fact, as $\sigma$ approaches infinity, $H(u)$ will become a continuous function and $h(x)$ an impulse.

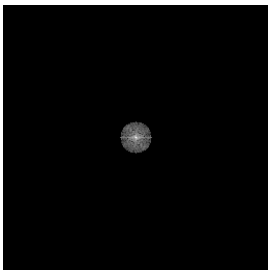
**(a)** Original



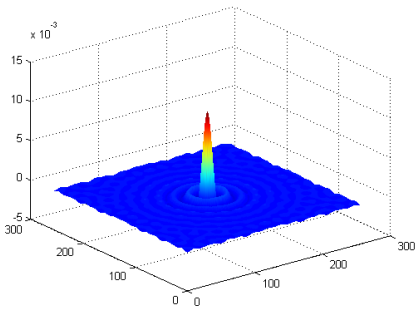
**(b)** DFT $\mathcal{F}(u, v)$



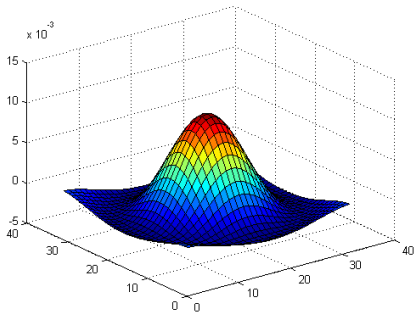
**(c)** $H(u, v)$



**(d)** $G(u, v) = H(u, v). * \mathcal{F}(u, v)$



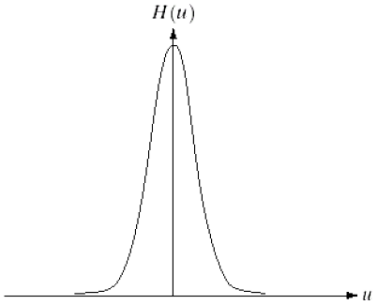

**(e)** $h(x, y)$



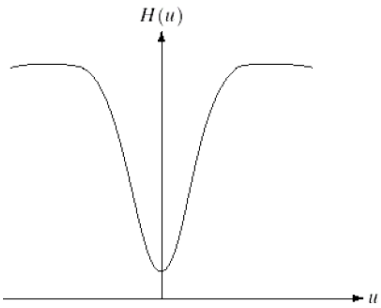

**(f)** $h(x, y)$ detail

**(g)** The result image $g(x, y)$

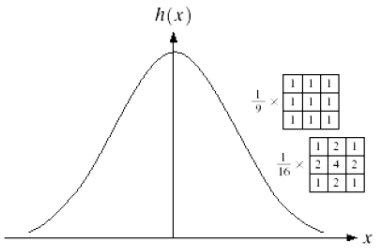
**Figure 2** – Simple filtering example in the Fourier domain. See the function `fourier_filter_example.m` on `Chamilo`.



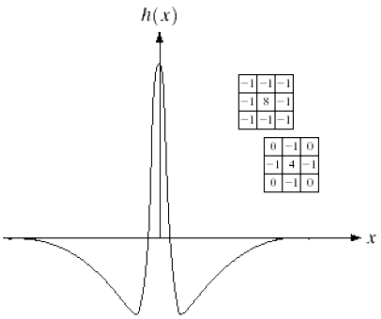

**(a)** $H(u)$, Gaussian low pass filter in frequency domain.




**(b)** $H(u)$, Gaussian high pass filter in frequency domain.




**(c)** $h(x)$, Gaussian low pass filter in spatial domain and 2 numerical approximations.




**(d)** $h(x)$, Gaussian high pass filter in spatial domain and 2 numerical approximations.

**Figure 3**

**Exercise**

- Implement and test a Gaussian low pass filter in the Fourier domain. Show the result and visualize the filter.
- Repeat the previous task, but with a Gaussian high pass filter in the Fourier domain.

## Digital Forensics

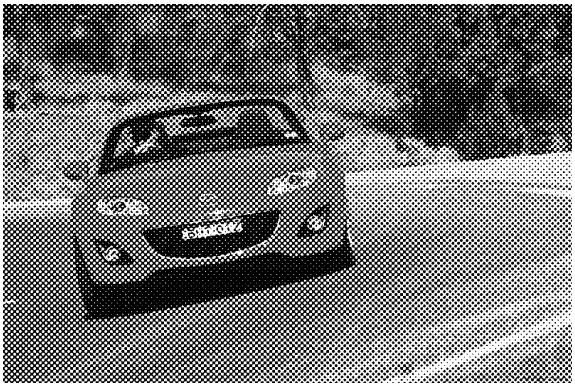
In the last short project, you are asked to pick one of the images (available on `Chamilo`) from Figure 4 and restore it to the best of your abilities.

**Exercise**

- Restore an image using filtering in the Fourier domain.

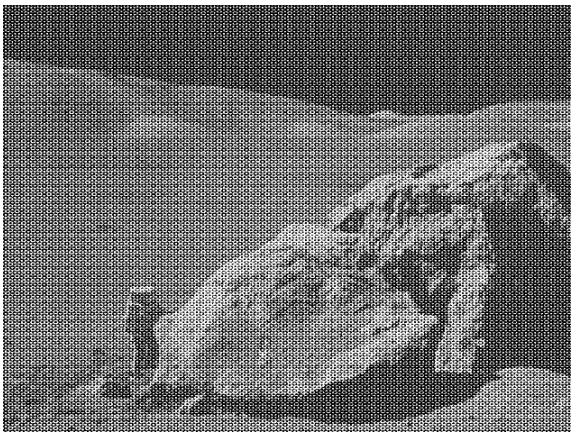
**Hints**

- Look up the definition for a *notch* filter.



**(a)**

**(b)**



**(c)**

**Figure 4** – Pick one of these images to restore.