# Introduction to Medical Imaging (5XSA0)

## Frequency domain processing

**Sveta Zinger**

**Video Coding and Architectures Research group, TU/e**
( s.zinger@tue.nl )





## What do we consider in frequency domain processing?

- **Filtering in the frequency domain via the Fourier transform**
  - can be used for image enhancement, restoration, compression
- **How to perform frequency domain processing in Matlab**

(The slides are based on "Digital Image Processing Using Matlab", R. C. Gonzalez, R. E. Woods, S. L. Eddins)





## 2D Discrete Fourier Transform

- **For image *f(x,y)* (*x*=0,1,2,…,*M*-1 and *y*=0,1,2,…,*N*-1), discrete Fourier transform (DFT) is**

$$F(u,v)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)e^{-j2\pi(ux/M+vy/N)}$$

where $u$=0,1,2,…,$M$-1 and $v$=0,1,2,…,$N$-1

- **Frequency domain is the coordinate system spanned by *F(u,v)* with *u* and *v* as frequency variables**





## Inverse Discrete Fourier Transform

- **Inverse DFT is given by**

$$f(x,y)=\frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)e^{j2\pi(ux/M+vy/N)}$$

where $x$=0,1,2,…,$M$-1 and $y$=0,1,2,…,$N$-1,
the values $F(u,v)$ are called Fourier coefficients

- ***F(0,0)* is the DC component (Direct current – from electrical engineering) of the Fourier transform**





## Analyzing a transform – (1)

- **Even if *f(x,y)* is real, the transform in general is complex**
- **Spectrum – the magnitude of *F(u,v)* – is the principal method of visually analyzing a transform**
- **Fourier spectrum is defined as**

$$|F(u,v)|=\left[R^2(u,v)+I^2(u,v)\right]^{1/2}$$

where $R(u,v)$ and $I(u,v)$ represent the real and imaginary components of $F(u,v)$





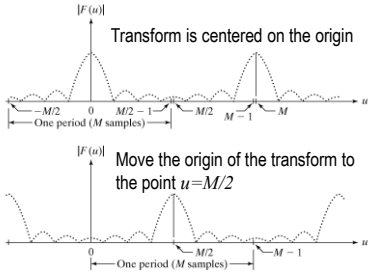
## Analyzing a transform – (2)

- **Fourier spectrum is symmetric about the origin**

$$|F(u,v)=|F(-u,-v)|$$

- **DFT is infinitely periodic in both *u* and *v* directions, the periodicity is determined by *M* and *N***
- **Image obtained by taking the inverse DFT is also infinitely periodic; DFT implementations compute only one period – *M x N***

## Analyzing a transform – (3)



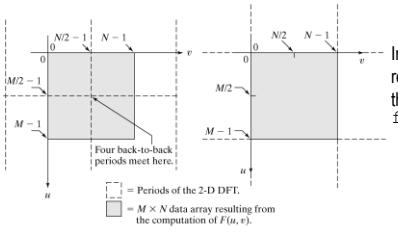

a b

**FIGURE 4.1** (a) Fourier spectrum showing back-to-back half periods in the interval $[0, M-1]$. (b) Centered spectrum in the same interval, obtained by multiplying $f(x)$ by $(-1)^x$ prior to computing the Fourier transform.





## Analyzing a transform – (4)




In Matlab, data is rearranged after the transform using `fftshift`

a b

**FIGURE 4.2** (a) $M \times N$ Fourier spectrum (shaded), showing four back-to-back quarter periods contained in the spectrum data. (b) Spectrum obtained by multiplying $f(x, y)$ by $(-1)^{x+y}$ prior to computing the Fourier transform. Only one period is shown shaded because this is the data that would be obtained by an implementation of the equation for $F(u, v)$ .





## Computing and visualizing 2D DFT in Matlab

* **DFT and its inverse are obtained in practice using a Fast Fourier Transform (FFT): `F=fft2(f)`**
  - it is necessary to pad the input image with zeros: `F=fft2(f,P,Q)` – pads the input so that the resulting function is of size $P$ x $Q$
  - Fourier spectrum: `S=abs(F)`
  - Inverse Fourier transform: `f=ifft2(F);`
    - obtain an image containing only real values: `f=real(ifft2(F))`



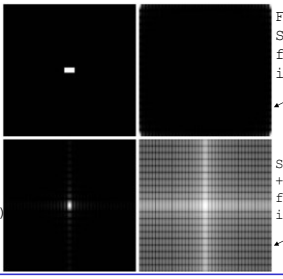


## Computing and visualizing 2D DFT in Matlab: example

a b
c d

**FIGURE 4.3** (a) A simple image. (b) Fourier spectrum. (c) Centered spectrum. (d) Spectrum visually enhanced by a log transformation.

```
Fc=fftshift(F);
figure;
imshow(abs(Fc),[])
```



```
F=fft2(f);
S=abs(F);
figure;
imshow(S,[])
```

```
S2=log(1+
+abs(Fc));
figure;
imshow(S2,[])
```





## Filtering in the frequency domain

* **Convolution theorem**

$$f(x,y) * h(h,y) \Leftrightarrow H(u,v)F(u,v)$$

and

$$f(x,y)h(h,y) \Leftrightarrow H(u,v) * F(u,v)$$

symbol "*" indicates convolution

$H(u,v)$ – filter transfer function

* **Frequency domain filtering: select a filter transfer function that modifies *F(u,v)* in a specified manner**



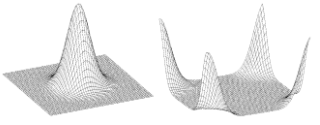


## Transfer function for lowpass filter: example

a b

**FIGURE 4.4** Transfer functions of (a) a centered lowpass filter, and (b) the format used for DFT filtering. Note that these are frequency domain filters.

after `fftshift`



Lowpass filter attenuates the high-frequency components of $F(u,v)$, while leaving the low frequencies relatively unchanged.

Result of lowpass filtering is image blurring (smoothing).

## Padding functions with zeros

For functions $f(x,y)$ and $h(x,y)$ of size $A$ x $B$ and $C$ x $D$ respectively, form two extended (padded) functions, both of size $P$ x $Q$, by appending zeros to $f$ and $g$.

Wraparound error is avoided by choosing $P \geq A + C - 1$ and $Q \geq B + D - 1$

If the functions are of the same size, $M$ x $N$, then the padding values are $P \geq 2M - 1$ and $Q \geq 2N - 1$





## Basic steps (1-2) in DFT filtering

Step-by-step procedure involving Matlab functions, where
$f$ – image to be filtered,
$g$ – result,
$H(u,v)$ – filter function of the same size as the padded image.

1. Obtain the padding parameters: `PQ=2*size(f).`
2. Obtain the Fourier transform with padding: `F=fft2(f,PQ(1),PQ(2)).`





## Basic steps (3-6) in DFT filtering

3. Generate a filter function $H$ of size $PQ1$ x $PQ2$. If the filter function is centered, let `H=fftshift(H)` before using the filter.
4. Multiply the transform by the filter: `G=H.*F;`
5. Obtain the real part of the inverse FFT of $G$: `g=real(ifft2(G));`
6. Crop the top left rectangle of the original size: `g=g(1:size(f,1),1:size(f,2));`





## Filtering procedure summarized

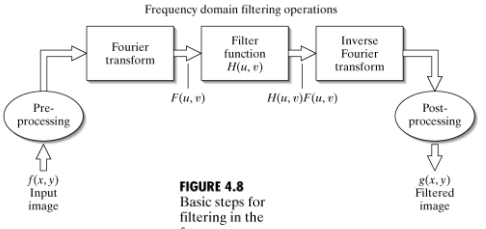



**FIGURE 4.8** Basic steps for filtering in the frequency domain.





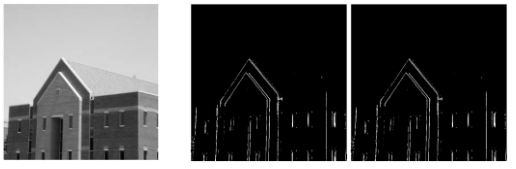
## Converting spatial filters into equivalent frequency domain filters

- **Choice between filtering in spatial or frequency domain may depend on the computational efficency**
- **Filter in the frequency domain:**
  **`H=freqz2(h,R,C),`**
  where $h$ is a 2D spatial filter,
  $R$ is the number of rows and
  $C$ is the number of columns that we wish filter H to have





## Spatial filtering (Sobel) versus equivalent frequency domain filtering: identical results



initial image | spatial filtering | frequency domain filtering (thresholded results)

## Creating meshgrid arrays for filters in the frequency domain

```
function [U V]=dftuv(M,N)
% DFTUV Computes meshgrid frequency matrices. U and V are both M-by-N
% Set up range of variables
u=0:(M-1); v=0:(N-1);
% Compute the indices for use in meshgrid
idx=find(u>M/2); u(idx)=u(idx)-M;
idy=find(v>N/2); v(idy)=v(idy)-N;
% Compute the meshgrid arrays
[U V]=meshgrid(v,u);
```





## Lowpass frequency domain filters – (1)

* **Ideal lowpass filter (ILPF) has the transfer function**

$$H(u,v)=\begin{cases}1 & \text{if } D(u,v)\le D_0\\ 0 & \text{if } D(u,v)> D_0\end{cases}$$

where $D_0$ – specified nonnegative number,

$D(u,v)$ – distance from point (u,v) to the center of the filter

– Ideal filter "cuts off" (multiplies by 0) all components of $F$ outside the circle and leaves unchanged (multiplies by 1) all components on, or inside, the circle





## Lowpass frequency domain filters – (2)

* **Butterworth lowpass filter (BLPF) of order n, with a cutoff frequency at a distance $D_0$ from the origin, has the transfer function**

$$H(u,v)=\frac{1}{1+\left[D(u,v)/D_0\right]^{2n}}$$

– BLPF transfer function does not have a sharp discontinuity at $D_0$





## Lowpass frequency domain filters – (3)

* **Transfer function of a Gaussian lowpass filter (GLPF)**

$$H(u,v)=e^{-D^2(u,v)/2\sigma^2}$$

where $\sigma$ – standard deviation

– By letting $\sigma = D_0$, we obtain the expression for GLPF in terms of the cutoff parameter:

$$H(u,v)=e^{-D^2(u,v)/2D_0^2}$$





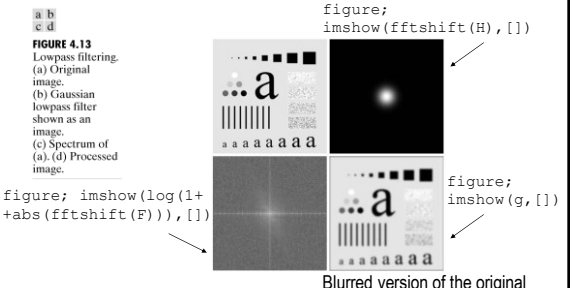
## Gaussian lowpass filter: example – (1)

```
PQ=2*size(f);
% Create meshgrid for the transfer function – you need dftuv.m
[U V]=dftuv(PQ(1),PQ(2));
D=sqrt(U.^2+V.^2);
% define the parameter for cut-off frequency D0
D0=0.05*PQ(2); % 5% of the padded image width
% Perform FFT of the original image and obtain the filter transfer function
F=fft2(f,PQ(1),PQ(2));
H=exp(-(D.^2)/(2*(D0^2)));
% Obtain and crop the resulting image
g=real(ifft2(H.*F));
g=g(1:size(f,1),1:size(f,2));
```





## Gaussian lowpass filter: example – (2)




Blurred version of the original

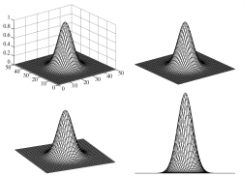
## 3D plots – (1)

- Draw a plot: `mesh`, `surf`
- Set or switch off axis: `axis`
- Switch on or off the grid: `grid`
- Change the viewing point:
  - `view`
  - Click on the "Rotate 3D" button in the figure window's toolbar



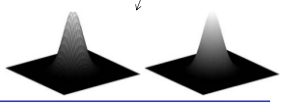


## 3D plots – (2)



a b
c d

**FIGURE 4.15** (a) A plot obtained using function `mesh`. (b) Axes and grid removed. (c) A different perspective view obtained using function `view`. (d) Another view obtained using the same function.



a b

**FIGURE 4.16** (a) Plot obtained using function `surf`. (b) Result of using the command `shading interp`.





## Sharpening frequency domain filters: highpass filtering

- Highpass filtering sharpens the image by attenuating the low frequencies and leaving the high frequencies of the Fourier transform relatively unchanged
- Transfer function of a highpass filter:

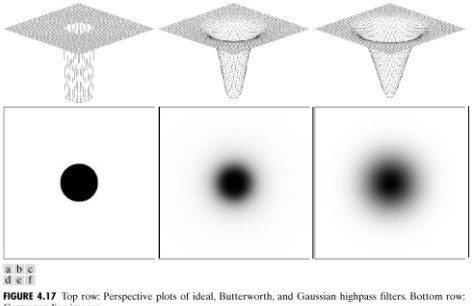
$$H_{hp}(u,v) = 1 - H_{lp}(u,v)$$

where $H_{lp}(u,v)$ – transfer function of the corresponding lowpass filter





## Highpas filters: examples



a b c
d e f

**FIGURE 4.17** Top row: Perspective plots of ideal, Butterworth, and Gaussian highpass filters. Bottom row: Corresponding images.





## Reference


- Rafael C. Gonzalez, Richard E. Woods, Steven L. Eddins, "Digital Image Processing Using Matlab", Pearson Education, 2004
  - Chapter 4