APPM4058A & COMS7238A: Digital Image Processing

Hairong Wang

School of Computer Science & Applied Mathematics
University of the Witwatersrand, Johannesburg

2019-2-27

Contents

WITS UNIVERSITY

Outline

WITS UNIVERSITY

The 2D discrete Fourier transform

Let $f(x, y)$ for $x = 0, 1, \ldots, M-1$, and $y = 0, 1, \ldots, N-1$ denote a digital image of size $M \times N$ pixels. The 2D discret Fourier transform (DFT), $F(u, v)$ of $f(x, y)$ is

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

where $u = 0, 1, \ldots, M-1$, and $v = 0, 1, \ldots, N-1$.

- The frequency domain is the coordinate system spanned by $F(u, v)$ with u and v as frequency variables.
- Analogous to the above is the spatial domain spanned by $f(x, y)$ with x and y as the spatial variables.

WITS UNIVERSITY

The inverse DFT

The inverse DFT is given by

$$f(x, y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) e^{j2\pi(ux/M + vy/N)}, \tag{2}$$

where $x = 0, 1, \ldots, M - 1$ and $y = 0, 1, \ldots, N - 1$.

WITS UNIVERSITY

The 2D DFT cont.

- The DFT of $f(x, y)$, $F(u, v)$, is complex in general.
- To analyze the transform visually, we compute its spectrum, i,e, magnitude.

$$|F(u, v)| = \left(R^2(u, v) + I^2(u, v)\right)^{1/2}, \tag{3}$$

where R(u,v) and $I(u, v)$ are the real and imaginary parts of $F(u, v)$.

- The phase angle of the transform is defined as

$$\phi(u, v) = arctan\left(\frac{I(u, v)}{R(u, v)}\right) \tag{4}$$

- Using (3) and (4), $F(u, v)$ can be expressed in polar form

$$F(u, v) = |F(u, v)|e^{j\phi(u,v)}, \tag{5}$$

- The power spectrum is defined as the square of the magnitude,

$$P(u, v) = |F(u, v)|^2 \tag{6}$$

WITS UNIVERSITY

The 2D DFT cont.

- Fourier transform is conjugate symmetric about origin. That implies the Fourier spectrum is symmetric about the origin, i.e.,

$$|F(u,v)| = |F(-u,-v)| \tag{7}$$

- $F(u,v)$ is infinitely periodic in both u and v direction,

$$F(u,v) = F(u+k_1M,v) = F(u,v+k_2N) = F(u+k_1M,v+k_2N), \tag{8}$$

where the periodicity is determined by M and N.

.

WITS UNIVERSITY

Computing the 2D DFT

- The 2D DFT is computed using function

$$F = fft2(f) \quad \text{in Matlab, or} \quad F = fftpack.fft2(f) \quad \text{in Python Scipy} \tag{9}$$

  returns a FT that is size of $M \times N$, with the origin of the data at the top left, and with four quarter periods meeting at the center of the frequency rectangle.

- The Fourier spectrum is obtained by

$$S = abs(F) \quad \text{in Matlab, or} \quad S = numpy.abs(F) \quad \text{in Python} \tag{10}$$

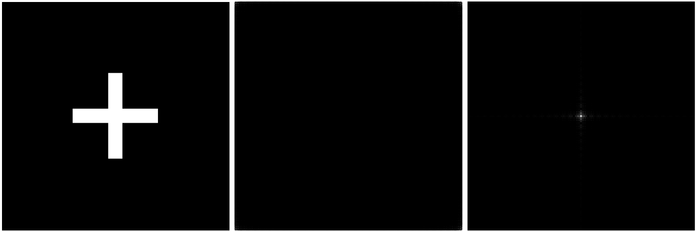
  which is the square root of the sum of the squares of real and imaginary parts of FT.

- S can be displayed as an image;
- Use *fftshift* to shift the origin of the transform to the center of the frequency rectangle.

$$Fc = fftshift(F), \text{ in Matlab, or } Fc = fftpack.fftshift(F), \text{ in Python Scipy} \tag{11}$$

  where F is obtained using *fft2*, and Fc is the centered transform.

Example - visualizing the Fourier spectrum



(a) (b) (c)

Figure: (a) An image; (b) Fourier spectrum of (a); (c) Centered spectrum of (a).

Computing the 2D DFT cont.

- The range of the Fourier spectrum is very large
- This can be handled via a *log* transform,

$$S2 = log(1 + abs(Fc)). \tag{12}$$

- Function *ifftshift* in both Matlab and Scipy reverses the shifting, i.e.,

$$F = ifftshift(Fc). \tag{13}$$

Computing the 2D DFT cont.

To compute the phase angle, use

$$phi = atan2(I, R) \quad \text{in Matlab, or} \quad phi = numpy.arctan2(I, R) \quad \text{in Python} \tag{14}$$

where I and R are the imaginary and real parts of F, respectively. They can be obtained using $I = imag(F)$ and $R = real(F)$ in Matlab or Python Numpy.

- The *phi* is a matrix of same size as I and R, with its elements are angles of radian in $[-\pi, \pi]$ measured with respect to real axis.
- We can also use

  $$phi = angle(F) \tag{15}$$

  in both Matlab and Python Numpy, without extracting the imaginary and real parts explicitly.

Computing the 2D DFT cont.

The inverse of DFT can be obtained using Matlab or Python Scipy

$$f = \mathit{ifft2}(F). \tag{16}$$

Basics

The foundation in both spatial and frequency domain filtering is the convolution theorem,

$$f(x, y) \bigstar h(x, y) \Leftrightarrow F(u, v)H(u, v), \tag{17}$$

and, conversely,

$$f(x, y)h(x, y) \Leftrightarrow F(u, v) \bigstar H(u, v), \tag{18}$$

where $\bigstar$ indicates the convolution of two functions.

Basics cont.

- Images and their transforms are periodic for DFT.
- Convolving periodic functions can cause interference between adjacent periods, if the periods are close. This interference is referred to as wraparound error.
- The wraparound error can be avoided by padding the functions with zeros.

Basics cont.

- Assume $f(x, y)$ and $h(x, y)$ are of size $A \times B$ and $C \times D$, respectively. To form padded functions for f and h of size $p \times Q$, where
$$P \geq A + C - 1, \tag{19}$$
and
$$Q \geq B + D - 1. \tag{20}$$
- If both f and h are of the same size, $M \times N$, then
$$P \geq 2M - 1, \tag{21}$$
and
$$Q \geq 2N - 1. \tag{22}$$

Basics cont.

The periodic sequences, or padded image and filter, are formed by extending $f(x,y)$ and $h(x,y)$ as follows:

$$f_p(x,y) = \begin{cases} f(x,y) & 0 \leq x \leq A-1 \quad \text{and} \quad 0 \leq y \leq B-1 \\ 0 & A \leq x \leq P \quad \text{or} \quad B \leq y \leq Q \end{cases} \tag{23}$$

and

$$h_p(x,y) = \begin{cases} h(x,y) & 0 \leq x \leq C-1 \quad \text{and} \quad 0 \leq y \leq D-1 \\ 0 & C \leq x \leq P \quad \text{or} \quad D \leq y \leq Q \end{cases} \tag{24}$$

- Implement a `[P, Q]=paddedsize(size(f), size(h))` function in Matlab or Python to compute the size for the padded image.

Basics cont.

Figure: (a) The original image 'square'; (b) image lowpass filtered in the frequency domain without padding; (c) image lowpass filtered in the frequency domain with padding.

Basics cont.

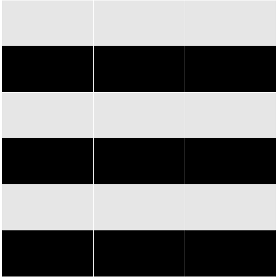


Figure: Implied infinite periodic sequence of the image 'square'. The thin white lines are not part of the image.

Basics cont.

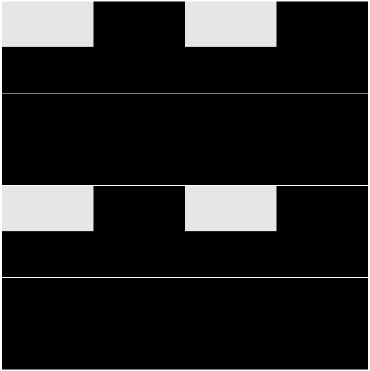


Figure: The same periodic sequence after padding with 0s. The thin white lines are not part of the image.

Basic steps in DFT filtering

- Convert the image to single or double;
- Obtain the padding parameters, and then create the padded image — i.e., extend the row and column ends with zeros.
- Obtain the FT for the padded image, or in Matlab you can use the following function.

$$F = fft2(f, P, Q); \tag{25}$$

- Obtain the desired filter in frequency domain, H, of the same size as the padded image.
- Multiply the transform by the filter

$$G = H. * F; \tag{26}$$

- Obtain the inverse of FT using Matlab or Python Scipy

$$g = ifft2(G); \tag{27}$$

- Crop the top left rectangle of g to obtain an image of the original size
- Convert the image to the class of the input image.

WITS UNIVERSITY

Outline

WITS UNIVERSITY

Obtaining frequency domain filter from spatial domain

- In general, filtering in spatial domain is more efficient computationally than frequency domain filtering when the filters are small.
- Filtering using an FFT algorithm can be faster than a spatial implementation when the filters become larger, such as having more than 32 elements.
- How to convert a spatial filter into an equivalent frequency domain filter? In Matlab,

$$H = freqz2(h, P, Q), \tag{28}$$

  where h is a 2D spatial filter, H is the corresponding filter in the frequency domain, and P and Q are the number of rows and columns in H.

Obtaining frequency domain filter from spatial domain cont.

- In order to obtain the corresponding filter in the frequency domain, preprocessing and postprocessing are often needed.

$$\begin{aligned}[P, Q] &= paddedsize(size(f), size(h)); \\ H &= freqz2(h, P, Q); \\ H1 &= ifftshift(H);\end{aligned} \tag{29}$$

- In Python, you can implement this by first padding your filter to the desired size of the frequency domain filter; then find the DFT of the padded filter.

Example

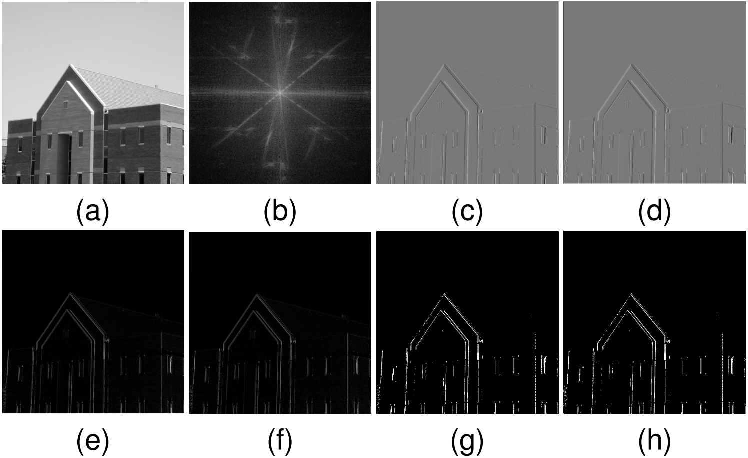


Figure: Lowpass filtering. (a) The original image; (b) The Fourier spectrum of (a); (c) Spatial domain filtering using a vertical Sobel mask, (d) Frequency domain filtering using a filter obtained from vertical Sobel; (e), (f) absolute values of (c) and (d), respectively; (g), (h) thresholded versions of (e), (f), respectively.

Outline

WITS UNIVERSITY

Generating filters directly in the frequency domain

- An ideal low-pass filter (ILPF) has the transform function

$$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) \leq D_0 \\ 0 & \text{if } D(u, v) > D_0 \end{cases} \tag{30}$$

  where D_0 is a positive number and $D(u, v)$ is the distance from point (u, v) to the center of the filter.
- Since $G = H. * F$, an ILPF “cuts off” all components of $F(u, v)$ outside the circle, and leaves unchanged all components on, or inside the circle.

Computing the distance in the frequency rectangle

- To implement filters in frequency domain, we need to create the meshgrid arrays for distance computation.

In Matlab

```
%set up range of variables
u=single(0:(M-1));
v=single(0:(N-1));
%compute the indices to use in meshgrid
idx=find(u>M/2);
u(idx)=u(idx)-M;
idy=find(v>N/2);
v(idy)=v(idy)-N;
[V,U]=meshgrid(v,u);
%compute the distance from every point to the origin
D=hypot(V,U);
```

Example

- Compute the distance from every point in a matrix to the origin (the point at (0,0)).
- Using meshgrid simplified the computation to `sqrt(V.^2 + U.^2)`.

```
M=5, N=7;
V =
0       1       2       3      -3      -2      -1
0       1       2       3      -3      -2      -1
0       1       2       3      -3      -2      -1
0       1       2       3      -3      -2      -1
0       1       2       3      -3      -2      -1

U =
0       0       0       0       0       0       0
1       1       1       1       1       1       1
2       2       2       2       2       2       2
-2      -2      -2      -2      -2      -2      -2
-1      -1      -1      -1      -1      -1      -1
```

Computing the distance in the frequency rectangle

In Python

```
#set up range of variables
import numpy as np
u=np.arange(0,5,1.0)
v=np.arange(0,7,1.0)
#compute the indices to use in meshgrid
idx=np.where(u>M/2)
u[idx]=u[idx]-M
idy=np.where(v>N/2)
v[idy]=v[idy]-N
V,U=np.meshgrid(v,u)
#D=np.sqrt(V**2+U**2)
D=V**2+U**2
```

WITS UNIVERSITY

```
array([
[ 0.,   1.,   4.,   9.,   9.,   4.,   1.],
[ 1.,   2.,   5.,  10.,  10.,   5.,   2.],
[ 4.,   5.,   8.,  13.,  13.,   8.,   5.],
[ 4.,   5.,   8.,  13.,  13.,   8.,   5.],
[ 1.,   2.,   5.,  10.,  10.,   5.,   2.]
])
```

```
>>> ft.fftshift(D)
array([
[13.,   8.,   5.,   4.,   5.,   8.,  13.],
[10.,   5.,   2.,   1.,   2.,   5.,  10.],
[ 9.,   4.,   1.,   0.,   1.,   4.,   9.],
[10.,   5.,   2.,   1.,   2.,   5.,  10.],
[13.,   8.,   5.,   4.,   5.,   8.,  13.]
])
```

To compute a filter in frequency domain

```
%assume we use a D0 equals to 5% of the padded image (with size
     P * Q) width
D0 = 0.05 * Q
H = exp(-(D.^2)/(2*(D0^2)))
```

Butterworth lowpass filter (BLPF)

- BLPF has the transform function

$$H(u, v) = \frac{1}{1 + (D(u, v)/D_0)^{2n}}, \tag{31}$$

  where n is the order of BLPF. $n = 2$ is often used.
- Compared to ILPF, BLPF does not have a sharp discontinuity at D_0.

Gaussian lowpass filter (GLPF)

- GLPF has the transform function

$$H(u,v) = e^{-D^2(u,v)/2\sigma^2}, \tag{32}$$

where σ is the standard deviation, and a measure of the spread of the Gaussian curve.

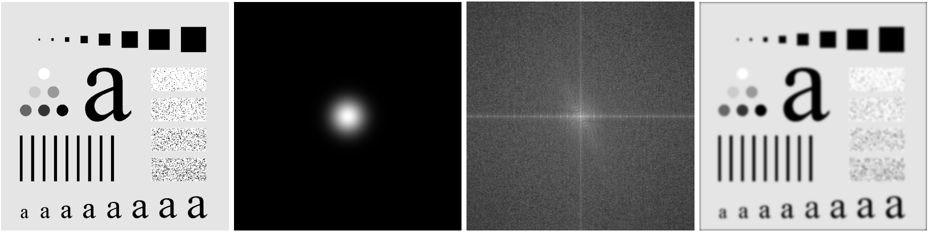
- By letting $\sigma = D_0$, we have

$$H(u,v) = e^{-D^2(u,v)/2D_0^2}, \tag{33}$$

where D_0 is the cutoff frequency.

WITS UNIVERSITY

Example using GLPF



(a) (b) (c) (d)

Figure: Lowpass filtering. (a) the original image; (b) Gaussian LPF shown as an image; (c) Spectrum of (a); (d) Filtered image.

Outline

Highpass frequency domain filtering

- Lowpass filtering and highpass filtering are a pair of opposite processes.
- Low pass filtering blurs an image, while highpass filtering sharpens the image;
- Given the transform function $H_{LP}(u, v)$ of a lowpass filter, the transform function for highpass filter is

$$H_{HP} = 1 - H_{LP}(u, v). \tag{34}$$

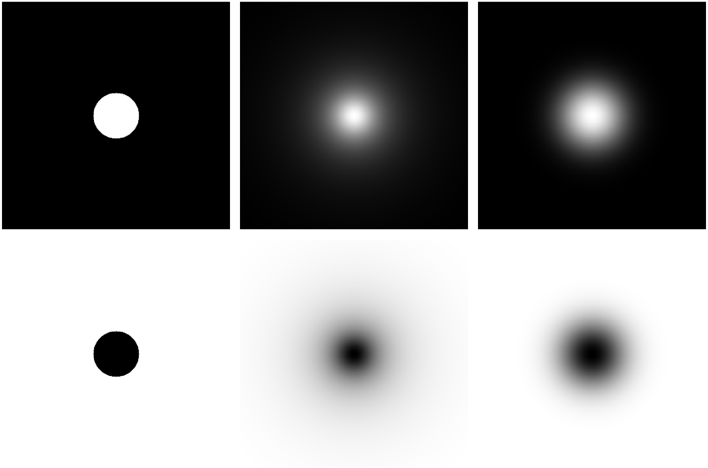
WITS UNIVERSITY

Highpass frequency domain filtering cont.

	Lowpass $H(u, v)$	Highpass $H(u, v)$
Ideal	$\begin{cases} 1 & \text{if } D(u, v) \leq D_0 \\ 0 & \text{if } D(u, v) > D_0 \end{cases}$	$\begin{cases} 0 & \text{if } D(u, v) \leq D_0 \\ 1 & \text{if } D(u, v) > D_0 \end{cases}$
Butterworth	$\frac{1}{1+(D(u,v)/D_0)^{2n}}$	$\frac{1}{1+(D_0/D(u,v))^{2n}}$
Gaussian	$e^{-D^2(u,v)/2\sigma^2}$	$1 - e^{-D^2(u,v)/2\sigma^2}$

Table: Frequency domain filters

Highpass frequency domain filtering cont.



(a) Ideal (b) Butterworth (c) Gaussian

Figure: Spatial and frequency domain lowpass and highpass filters.

Highpass frequency emphasis filtering

- Highpass filters zero out the mean component, reducing the average value of an image to zero.
- The remedy – add an offset to a highpass filter, called high frequency emphasis filtering.

$$H_{HFE} = a + bH_{HP}(u, v), \tag{35}$$

where a is the offset, b is the multiplier, H_{HP} is the transform function of a highpass filter.

Example of highpass frequency domain filtering

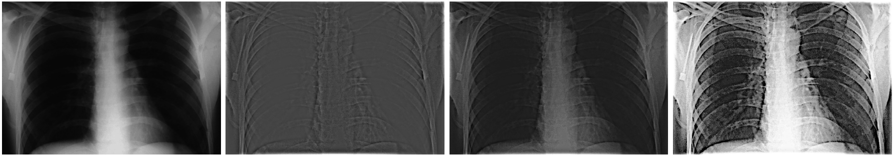

(a) (b) (c) (d)

Figure: Highpass filtering. (a) the original image; (b) the result of filtering using Butterworth filter; (c) Using highpass emphasis filtering; (d) After histogram equalization of (c).

Outline

Bandreject and bandpass filters

- Table 2 shows expressions for ideal, Butterworth, and Gaussian bandreject filters.
- We obtain a bandpass filter $H_{BP}(u,v)$ from a given bandreject filter $H_{BR}(u,v)$ using

$$H_{BP}(u,v) = 1 - H_{BR}(u,v). \tag{36}$$

	$H(u,v)$
Ideal	$\begin{cases} 0 & \text{for } D_0 - \frac{W}{2} \leq D(u,v) \leq D_0 + \frac{W}{2} \\ 1 & \text{otherwise} \end{cases}$
Butterworth	$\frac{1}{1+\left(\frac{WD(u,v)}{D^2(u,v)-D_0^2}\right)^{2n}}$
Gaussian	$1 - e^{-\left(\frac{D^2(u,v)-D_0^2}{WD(u,v)}\right)^2}$

Table: Bandreject filters. W is the width of the band, $D(u,v)$ is the distance from the center of the filter, D_0 is the radius of the center of the band.

Bandreject and bandpass filters example

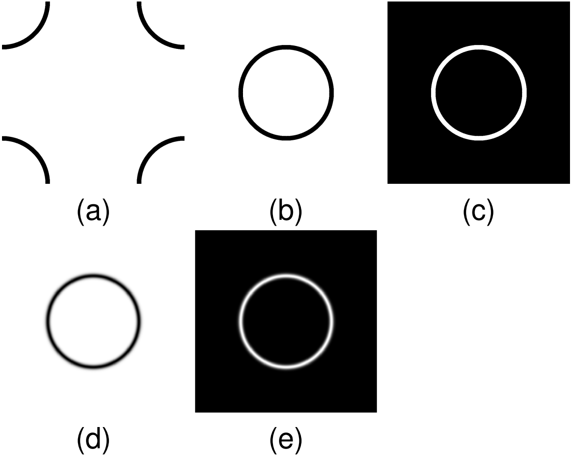


Figure: (a) An ideal bandreject filter with the origin at the top left corner; (b) An ideal bandreject filter with the origin at the center; (c) The corresponding bandpass filter of (b); (d) A Gaussian bandreject filter; (e) The corresponding bandpass filter of (d). Parameters used: $M = N = 800$, $D0 = 200$, $W = 20$.

References

- Further reading in [Gonzalez and Woods, 2008, Chapter 4], [Gonzalez et al., 2009, Chapter 4].
- Further reading at `http://www.cs.unm.edu/~brayer/vision/fourier.html`

Gonzalez, R. C. and Woods, R. E. (2008). *Digital Image Processing*. Pearson Prentice Hall, Upper Saddle River, NJ 07458, third edition.

Gonzalez, R. C., Woods, R. E., and Eddins, S. L. (2009). *Digital Imageage Processing using MATLAB*. Gatesmark Publishing.

WITS UNIVERSITY