Colorado School of Mines

Image and Multidimensional Signal Processing

Professor William Hoff
Dept of Electrical Engineering & Computer Science
http://inside.mines.edu/~whoff/
Frequency Domain Filters
Filtering in Frequency Domain

- If we want to do a spatial filtering operation
  \[ g(x, y) = h(x, y) \ast f(x, y) \]

- By the convolution theorem, we can transform the mask and the image to the frequency domain and do the operation there
  \[ \mathcal{F} \{ h(x, y) \ast f(x, y) \} = H(u, v)F(u, v) \]
  - then
  \[ g(x, y) = \mathcal{F}^{-1} \{ H(u, v)F(u, v) \} \]

Notes:
- The convolution kernel is the same size as the image (you have to pad the kernel with zeros if necessary)
- Multiplication is point-by-point, of complex numbers
Advantages of Filtering in Frequency Domain

- Cost (number of operations) of the computation of Fast Fourier Transform is $O(MN \log MN)$
  where $MN =$ number of points in image
- The total cost of filtering in the frequency domain is dominated by FFT
- Compare this to convolution in spatial domain - it is $O( (mn)(MN) )$

Convolution in frequency domain faster for large kernels (when mn gets much larger than log(MN))
Fourier-Domain Filtering in Matlab

- Need to pad filter to be same size as image
  - Can do this by setting the point in the lower right corner
    \[ h(\text{size}(f,1), \text{size}(f,2)) = 0; \]
  - where \( \text{size}(f) \) is the size (#rows, #cols) of the image
  - Matlab expands the filter and fills new values to zero

- The inverse Fourier Transform (\texttt{ifft2}) should yield a real image
  - But take \texttt{real} of final result (to get rid of tiny imaginary values)
Example

• Read image and take Fourier transform

\[
f = \text{imread('circuit.jpg');}
F = \text{fft2(double(f));}
\]

% Shift to center, take abs to see spectrum, take log to see small values
figure, imshow(log(abs(fftshift(F))), []);

• Create filter

\[
N=20;
h = \text{ones(N)/(N*N)};
h(\text{size(f,1)}, \text{size(f,2)}) = 0;
H = \text{fft2(h)};
figure, imshow(log(abs(fftshift(F))), []);
\]

• Apply filter, go back to spatial domain

\[
G = F .* H;
g = \text{real(ifft2(G));}
figure, imshow(g, []);
\]
Low Pass Filters

• Box filter

\[ h(x) \quad H(u) \quad f \ast h \]

Convolution w/ step edge

• Ideal low pass filter

\[ h(x) \quad H(u) \quad f \ast h \]

Convolution w/ step edge
Ideal Low Pass Filter in 2D

FIGURE 4.10 (a) Perspective plot of an ideal lowpass filter transfer function. (b) Filter displayed as an image. (c) Filter radial cross section.
FIGURE 4.12 (a) Original image. (b)–(f) Results of ideal lowpass filtering with cutoff frequencies set at radii values of 5, 15, 30, 80, and 230, as shown in Fig. 4.11(b). The power removed by these filters was 8, 5.4, 3.6, 2, and 0.5% of the total, respectively.

Note “ringing” near sharp edges
Matlab Example

• Create image $H$ of a disk in center
• Multiplication in freq domain
  \[ G = H \times F \]
• `ifft2(G)`
• Note ringing (do `improfile`)

![Image of a disk in center](image1.png)

![Graph of distance along profile](image2.png)
f = double imread('moon.tif');
h = size(f,1);
w = size(f,2);

% Create ideal low pass filter - a circle in middle of image
R = 30; % cutoff frequency
H = zeros(h,w);
for v=1:h
    for u=1:w
        if (v-h/2)^2 + (u-w/2)^2 < R^2
            H(v,u) = 1;
        end
    end
end
imshow(H, []);

H = ifftshift(H); % put zero freq in upper left corner
figure, imshow(H, []);

F = fft2(f);
G = H .* F;
g = real(ifft2(G));
figure, imshow(g, []);
Gaussian Lowpass Filter

- A Gaussian in the spatial domain also has the form of a Gaussian in the frequency domain.
- No ringing, but allows high frequencies to pass.

**FIGURE 4.17** (a) Perspective plot of a GLPF transfer function. (b) Filter displayed as an image. (c) Filter radial cross sections for various values of $D_0$. 
FIGURE 4.18 (a) Original image. (b)-(f) Results of filtering with Gaussian lowpass filters with cutoff frequencies set at radii values of 5, 15, 30, 80, and 230, as shown in Fig. 4.11(b). Compare with Figs. 4.12 and 4.15.
Butterworth Lowpass Filter

• Definition

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

- \( D(u, v) \) is distance from (0,0) to \((u, v)\)
- \( D_0 \) is cutoff frequency
- \( n \) is the “order” of the filter

• Properties

- For \( D(u, v) << D_0 \), \( H \approx 1 \)
- For \( D(u, v) >> D_0 \), \( H \approx 0 \)
- At \( D(u, v) = D_0 \), \( H = 1/2 \)

• Advantages

- Reduces “ringing” while keeping clear cutoff
- Tradeoff between amount of ringing and sharpness of cutoff
For large $n$, $H(u,v)$ approaches the ideal low pass filter.
FIGURE 4.16  (a)–(d) Spatial representation of BLPFs of order 1, 2, 5, and 20, and corresponding gray-level profiles through the center of the filters (all filters have a cutoff frequency of 5). Note that ringing increases as a function of filter order.
FIGURE 4.15 (a) Original image. (b)–(f) Results of filtering with BLPFs of order 2, with cutoff frequencies at radii of 5, 15, 30, 80, and 230, as shown in Fig. 4.11(b). Compare with Fig. 4.12.
Sharpening Filters

• Can obtain by

\[ H_{hp}(u, v) = 1 - H_{lp}(u, v) \]

• Types
  – Ideal high pass
  – Butterworth high pass
  – Gaussian high pass
FIGURE 4.24 Results of ideal highpass filtering the image in Fig. 4.11(a) with $D_0 = 15$, 30, and 80, respectively. Problems with ringing are quite evident in (a) and (b).
FIGURE 4.25 Results of highpass filtering the image in Fig. 4.11(a) using a BHPF of order 2 with $D_0 = 15, 30,$ and $80,$ respectively. These results are much smoother than those obtained with an ILPF.
Notch Filters

• A filter that rejects (or passes) specific frequencies
• Example: periodic noise corresponds to spikes or lines in the Fourier domain
• Can design a filter with zeros at those frequencies … this will remove the noise
• Examples:
  – Image mosaics
  – Scan line noise
  – Halftoning noise (moire patterns)
Steps in Notch Filtering

- Look at spectrum $|F(u,v)|$ of noisy image $f(x,y)$, find frequencies corresponding to the noise.

- Create a mask image $M(u,v)$ with notches (zeros) at those places, 1’s elsewhere.

- Multiply mask with original image transform; this zeros out noise frequencies:
  \[ G(u,v) = M(u,v) F(u,v) \]

- Take inverse Fourier transform to get restored image:
  \[ g(x,y) = \mathcal{F}^{-1}(G(u,v)) \]
FIGURE 4.64
(a) Sampled newspaper image showing a moiré pattern.
(b) Spectrum.
(c) Butterworth notch reject filter multiplied by the Fourier transform.
(d) Filtered image.
Matlab Example

- Image “Clown.tif”

```matlab
f = imread('Clown.tif');
figure, imshow(f, []);

F = fftshift(fft2(double(f)));
S = log(abs(F));
imwrite( S/max(S(:)), 'mask.tif');

% Edit image 'mask.tif' with another application such as "Paint".
% Draw black squares or circles at noise locations. Save it back
% to 'mask.tif'.
pause;

M = imread('mask.tif');
M = M(:,:,1); % Use only first band of color image
M = double((M>0)); % Threshold, so 0's are at noise locations

G = M .* F;
g = real(ifft2(ifftshift(G)));```

Notes:
- A Butterworth filter would be a better mask to use than the circles I drew
- You should put the notches in symmetrical locations
- Example of horizontal scan lines
- Create a notch of vertical lines in frequency domain

**FIGURE 4.65**
(a) $674 \times 674$ image of the Saturn rings showing nearly periodic interference.
(b) Spectrum: The bursts of energy in the vertical axis near the origin correspond to the interference pattern. (c) A vertical notch reject filter.
(d) Result of filtering. The thin black border in (c) was added for clarity; it is not part of the data. (Original image courtesy of Dr. Robert A. West, NASA/JPL.)
Another Application of Frequency Domain Filtering

• Assume we have two images, slightly shifted (translated)

• We can do a cross correlation between the two images

• The location of the peak score corresponds to the amount of translation

• However, the location of the peak is at some integer number of pixels – can we find a subpixel shift?

• Use the method of “phase correlation”

Movie of intervascular ultrasound (IVUS) images of a coronary artery, taken 1/30 sec apart. The heart motion causes a shift between the images.

Data courtesy of Dr. James Chen at the U of Colorado Health Sciences Center
Phase Correlation

- A shift in the spatial domain corresponds to a linear change in the phase, in the frequency domain

\[ f(x - x_0, y - y_0) \Leftrightarrow F(u, v) e^{-j2\pi (ux_0/M + vy_0/N)} \]

- Assume

\[ f_2(x, y) = f_1(x - x_0, y - y_0) \]

- Then

\[ F_2(u, v) = F_1(u, v) e^{-j2\pi (ux_0/M + vy_0/N)} \]

\[ \frac{F_2(u, v)}{F_1(u, v)} = e^{-j2\pi (ux_0/M + vy_0/N)} \]

- The phase angle is

\[ \phi(u, v) = 2\pi (ux_0/M + vy_0/N) \]

A least squares fit plane to the observed phases gives \( x_0, y_0 \) to subpixel precision.
Matlab Example

• We will shift image horizontally and see if we can recover the amount of shift
  – Take ratio of Fourier transforms of the original and shifted images
  – Find phase (Matlab function \texttt{angle})
  – Plot phase vs. $u$, calculate shift

• We’ll shift the image by applying an affine transform:

$$\begin{bmatrix} x & y & 1 \end{bmatrix} = \begin{bmatrix} u & v & 1 \end{bmatrix} \begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}$$

  – A pure translation is

$$\begin{bmatrix} x & y & 1 \end{bmatrix} = \begin{bmatrix} u & v & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_0 & y_0 & 1 \end{bmatrix}$$

• Also use bilinear interpolation to create the new, shifted image
I1 = double(imread('cameraman.tif'));
F1 = fft2(I1);

% Make a shifted image
x0 = 2.5;  % horizontal shift in pixels
Tform = maketform('affine', ... 
    [1 0 0; 
    0 1 0; 
    x0 0 1]);
I2 = imtransform(I1, Tform, ... 
    'XData', [1 size(I1,2)], ...  % forces output to be
    'YData', [1 size(I1,1)]);  % .. same size as input
F2 = fft2(I2);

% The ratio of Fourier transforms is exp(-j*2*pi*u*x0/M)
F3 = F2 ./ F1;

phaseAngle = angle(F3);  % get phase in radians
figure, imshow(phaseAngle, []);

% The phase is 2*pi*u*x0/M.
% The slope of this plot is 2*pi*x0/M.
% So x0 = slope*M/(2*pi)
figure, plot(phaseAngle(2,1:50));
Summary / Questions

• The convolution theorem says that convolution in one domain (e.g., spatial) is equivalent to point-by-point multiplication in the other domain (e.g., frequency).
  – It gives us a way to understand the behavior of filters.

• Examples of some kinds of filters:
  – An “ideal lowpass filter” passes all frequencies with magnitudes below a specific level, and attenuates all frequencies above that level.
  – An “ideal highpass filter” does the opposite.
  – A “notch” filter rejects (or passes) frequencies at a specific point (the notch).

• What would a “bandpass filter” look like in the frequency domain?