Machine Vision

Frequency-Domain Methods for Image Processing in MATLAB

Lab Activity Sheet 7

Author: Dr. Mostapha Kalami Heris

Introduction

Frequency-domain methods are an important part of modern image processing and computer vision. They provide a complementary way to represent and manipulate images, alongside the more familiar spatial-domain techniques. Instead of working directly with pixel values and local neighborhoods, frequency-domain methods describe an image in terms of how its intensity varies across different spatial frequencies.

In this lab, you will work with practical frequency-domain tools in MATLAB to understand how filters behave, how they can be designed in the frequency domain, and how they affect real images. You will connect these ideas to concepts from earlier labs on spatial filtering, contrast enhancement, and segmentation.

By the end of this lab, you should be able to:

- explain the basic ideas behind frequency-domain representations of images;
- describe how low, mid, and high spatial frequencies relate to smooth regions, edges, textures, and noise;
- compare spatial-domain and frequency-domain approaches and identify when each is more suitable;
- use key MATLAB functions to build frequency grids, inspect frequency responses, design 2-D filters, and apply them to images.

Why Frequency-Domain Techniques Matter

In the spatial domain, you have already seen filters such as averaging, Gaussian, and Laplacian kernels. These filters operate directly on pixel neighborhoods through convolution or correlation. Spatial filters are intuitive and powerful, especially when you need local operations or nonlinear methods such as median filtering or morphology. However, many important tasks are easier to understand and control in the frequency domain. The 2-D Fourier transform represents an image as a sum of sinusoidal components with different frequencies, magnitudes, and phases. In this representation:

- **low frequencies** correspond to slow variations and smooth regions;
- **high frequencies** correspond to edges, fine textures, and sharp transitions;
- **periodic patterns** appear as strong peaks at specific frequency locations.

Frequency-domain methods allow you to:

- design filters that act on precise frequency ranges;
- remove periodic noise by targeting isolated peaks in the spectrum;
- perform global smoothing or sharpening in a controlled way;
- implement large or complex filters more efficiently by working with transforms.

These capabilities are used in many real applications, including enhancement of medical images, document sharpening, removal of sensor artifacts, and preparation of images for feature detection and recognition.

Relationship Between Spatial and Frequency Domains

Spatial-domain and frequency-domain techniques are two different views of the same underlying signal. Convolution–multiplication duality states that:

- convolution in the spatial domain corresponds to multiplication in the frequency domain;
- multiplication in the frequency domain corresponds to convolution in the spatial domain.

This relationship explains why a smoothing kernel in the spatial domain corresponds to a low-pass filter in the frequency domain, and why an edge-enhancing kernel corresponds to a high-pass response. Many filters can be understood both as:

- a **spatial kernel**, and
- a **frequency response**.

In practice:

- **Spatial-domain filtering** is preferred for local or nonlinear operations, small kernels, or when neighborhood geometry matters.
- **Frequency-domain filtering** is preferred for global smoothing or sharpening, removal of periodic noise, or designing large filters.

Understanding both views helps you choose the right tool for each problem and combine them when necessary.

Typical Applications of Frequency-Domain Methods

Some common applications where frequency-domain techniques are particularly useful include:

- **Smoothing and denoising** — low-pass filtering reduces high-frequency noise.
- **Edge and detail enhancement** — high-pass or band-reject filters sharpen structures.
- **Texture analysis and segmentation** — band-pass filters isolate specific texture scales or orientations.
- **Removal of periodic noise** — notch filters suppress periodic interference.
- **Feature detection and recognition** — controlled frequency enhancement improves robustness for methods such as SIFT, SURF, and ORB.

In this lab, you will design and analyze 2-D frequency-domain filters and apply them to real images. You will connect their frequency responses to observable spatial-domain changes.

MATLAB Functions Used in This Lab

The following table lists MATLAB functions commonly used in frequency-domain image processing.

Function	**Purpose**	**Notes**
`fft2`	Two-dimensional discrete Fourier transform	Converts images to the frequency domain; use `fftshift` to center low frequencies.
`ifft2`	Inverse two-dimensional Fourier transform	Reconstructs spatial-domain images; use `real` to remove numerical artifacts.
`fftshift`	Shift zero-frequency to the center	Useful for spectrum visualization.
`ifftshift`	Undo `fftshift`	Required before applying `ifft2`.
`freqspace`	Generate 2-D frequency grids	Used with `fsamp2`, `fwind1`, `fwind2` to specify sampling locations.
`freqz2`	Compute 2-D frequency response	Displays magnitude response of 2-D FIR filters.
`fsamp2`	Design filters using frequency sampling	Produces filters matching a desired frequency response matrix.
`fwind1`	Window-based 2-D filter design (1-D window)	Applies a one-dimensional window to a desired response.
`fwind2`	Window-based 2-D filter design (2-D window)	Uses two-dimensional windows, such as Gaussian windows.
`ftrans2`	Transform 1-D FIR into 2-D FIR	Uses transformation methods such as McClellan transform.
`filter2`	Apply 2-D correlation	Performs spatial filtering using a kernel.
`imfilter`	General filtering function	Allows convolution, correlation, and boundary control.
`imshow`	Display images	Uses natural data ranges.
`imagesc`	Display scaled matrices	Useful for viewing frequency responses or kernels.

You will use a subset of these functions in the activities that follow.

Activity 1: Frequency Grids and 2-D Frequency Response

Objective

The goal of this activity is to construct two-dimensional frequency grids, create ideal frequency-domain masks, design simple filters using window-based methods, and visualise their frequency responses using MATLAB tools. This activity gives you a foundation for understanding how filters behave in the frequency domain. You will see how their shapes relate to smoothing, sharpening, and other image modifications.

Description

In this task, you will:

- generate a frequency grid using `freqspace`;
- create an ideal low-pass (or band-pass) response matrix;
- design a two-dimensional filter using a window method such as `fwind1`;
- visualise the resulting filter using `freqz2`;
- reflect on how the chosen window affects the characteristics of the filter.

Most of the code is provided so that you can focus on understanding the relationship between the desired frequency response and the resulting filter. At the end of the activity, you will explore variations with different cutoff radii, different window shapes, and alternative ideal responses.

Starter Code

```
% Generate 2-D frequency grid
[f1, f2] = freqspace(51, 'meshgrid');

% Create an ideal circular low-pass mask
D = sqrt(f1.^2 + f2.^2);
Hd = double(D <= 0.4);   % cutoff radius = 0.4

% Choose a 1-D window
w = hamming(51);

% Design the filter using fwind1
```

```
h = fwind1(Hd, w);

% Visualise the frequency response
freqz2(h);
title('2-D Frequency Response of Designed Filter');
```

Tasks

1. Use `freqspace` to generate a frequency grid of size 51×51.
2. Construct an ideal low-pass mask with a circular cutoff.
3. Apply `fwind1` to design a filter using a one-dimensional window.
4. Plot the resulting frequency response using `freqz2`.
5. Inspect the shape of the magnitude response and note how it relates to the ideal mask.

Open-Ended Exploration

The following tasks require your own experimentation:

- Change the cutoff radius (for example, 0.2, 0.3, 0.5) and observe how the designed filter changes.
- Replace the Hamming window with a Hann or Bartlett window by using `hann` or `bartlett`.
- Modify the ideal response to create a band-pass or notch filter instead of a low-pass filter.
- Compare how smooth the transition band is when you use different windows. Explain the differences that you observe.
- Explain why using a wider window produces a narrower transition band.

Reflection Questions

- How closely does the designed filter follow the shape of the ideal response?
- What is the effect of changing the cutoff radius?

- How does the chosen window influence the smoothness of the frequency response?
- Which window produced the most desirable result for your experiment, and why?

Activity 2: Designing 2-D Filters in the Frequency Domain

Objective

The goal of this activity is to design two-dimensional FIR filters using several frequency-domain methods and compare how well each method approximates a desired frequency response. You will use `fsamp2`, `fwind1`, and `fwind2` to create filters based on a target frequency mask and then inspect the resulting frequency responses using `freqz2`.
This activity includes some starter code. It also gives you several open-ended tasks so that you can explore how the different design methods behave. You will follow guided steps to construct a desired frequency response matrix and generate filters. After this, you will explore sampling density, window choices, and directional behaviour through your own experiments.

Description

You will:

- construct a desired band-pass response using `freqspace`;
- design a filter using three different methods (`fsamp2`, `fwind1`, `fwind2`);
- evaluate each filter by inspecting its frequency response;
- compare the characteristics and limitations of each method.

This activity helps you understand how sampling resolution, window selection, and filter size influence the final frequency response.

Starter Code

```
% Generate 2-D frequency grid
[f1, f2] = freqspace(51, 'meshgrid');

% Desired band-pass response
```

```
D = sqrt(f1.^2 + f2.^2);
Hd = double(D >= 0.2 & D <= 0.5);

% Method 1: Frequency sampling (fsamp2)
h_fs = fsamp2(Hd);

% Method 2: Window-based design with 1-D window
w = hann(51);
h_w1 = fwind1(Hd, w);

% Method 3: Window-based design with 2-D window
w2 = fspecial('gaussian', [51 51], 10);
h_w2 = fwind2(Hd, w2);

% Visualise frequency responses
figure;
subplot(1,3,1); freqz2(h_fs); title('fsamp2');

subplot(1,3,2); freqz2(h_w1); title('fwind1 (1-D window)');

subplot(1,3,3); freqz2(h_w2); title('fwind2 (2-D window)');
```

Filter normalization: Filters designed with `fsamp2`, `fwind1`, and `fwind2` are not always normalised automatically. This can cause the filtered image to appear brighter or darker. If needed, normalise the filter with:

```
h = h / sum(h(:));
```

Tasks

1. Use `freqspace` to generate a `51 × 51` frequency grid.
2. Construct a band-pass mask that selects frequencies between 0.2 and 0.5.
3. Design filters using:
   - `fsamp2`,
   - `fwind1`,
   - `fwind2`.
4. Visualise each filter's frequency response with `freqz2`.

5. Compare the responses and describe how each method interprets the same ideal mask.

Open-Ended Exploration

- Change the frequency grid resolution (for example, 25×25 or 101×101) and observe how sampling density influences the results.
- Replace the Hann window with other windows such as `hamming` or `bartlett` and compare how smooth the transition band becomes.
- For `fwind2`, try different 2-D windows such as:
  - `fspecial('gaussian', ...)` with different standard deviations,
  - `fspecial('disk', ...)` for circular smoothing.
- Evaluate computational cost by timing each method using `tic` and `toc`.
- Explore directional behaviour by constructing anisotropic frequency masks (for example, horizontal or vertical bands).
- Write a short paragraph comparing the strengths and weaknesses of each method.

Reflection Questions

- Which method produced the frequency response closest to your desired bandpass shape?
- How did the choice of window affect the sharpness or smoothness of the transition region?
- What happens when you increase the sampling resolution of `Hd`?
- Which method is most suitable for precise control, and which is best for simplicity?
- Did any method introduce unexpected artefacts or asymmetries?

Activity 3: Applying Frequency-Domain Filters to Real Images

Objective

The goal of this activity is to apply the filters that you designed in the frequency domain to real images, compare their effects with standard spatial-domain filters, and explore how different frequency responses influence image smoothing, edge enhancement, and noise removal. This activity is more open-ended: a small amount of starter code is provided, and you will design most of the experiments yourself.

Description

You will:

- take one or more filters from Activities 1 and 2 (or design new ones);
- apply them to images using both spatial-domain convolution and FFT-based frequency-domain multiplication;
- compare results with standard spatial filters such as Gaussian smoothing or Laplacian sharpening;
- explore the effects of low-pass, high-pass, band-pass, and notch-type frequency responses on real images.

This activity connects the abstract frequency responses you have seen in previous tasks with visible changes in the images. You are encouraged to try different parameter settings, images, and filter shapes, and to document your observations.

Starter Code

```
% Example: Apply a designed filter (h) in the spatial domain
I = im2double(imread('cameraman.tif'));

% Assume h is a 2-D FIR filter designed in Activities 1 or 2
I_filt_spatial = imfilter(I, h, 'replicate');

figure;
subplot(1,2,1); imshow(I, []); title('Original');
subplot(1,2,2); imshow(I_filt_spatial, []); title('Spatial-domain filtering');
```

```
% Example: Apply the same filter via FFT-based frequency-domain filtering
[M, N] = size(I);
H_pad = zeros(M, N);
[m_h, n_h] = size(h);

% Place h in the top-left corner and circularly shift to the centre
H_pad(1:m_h, 1:n_h) = h;
Hpad = circshift(Hpad, [-floor(mh/2), -floor(nh/2)]);

H = fft2(H_pad);

F = fft2(I);
G = H .* F;
I_filt_freq = real(ifft2(G));

figure;
subplot(1,2,1); imshow(I, []); title('Original');
subplot(1,2,2); imshow(I_filt_freq, []); title('Frequency-domain filtering');
```

Alternatively, you may use following snippet to calculate `H`, using `psf2otf` function.

```
[M, N] = size(I);
H = psf2otf(h, [M, N]);
```

Important: When implementing frequency-domain filtering manually, correct alignment of the filter is crucial. In the spatial domain, the center of the kernel is usually in the middle of the matrix. However, for `fft2`, the "origin" is at index (1,1). We use `circshift` (or the helper function `psf2otf`) to move the kernel center to (1,1) before transforming. If this step is skipped, the resulting filtered image may appear shifted by half the image size.
Note on padding: The size of the filter and the chosen padding method affect the final result. Simple zero-padding may introduce ringing or border artefacts, especially when the filter is large. If you observe such effects, try increasing the padded region or using alternative boundary strategies such as reflective or symmetric padding. Adjusting the padding strategy often improves the visual quality of the filtered image.

Tasks

1. Choose at least one filter `h` that you designed in Activities 1 or 2 (for example, a low-pass or band-pass filter).

2. Apply this filter to `cameraman.tif` using spatial-domain filtering with `imfilter` or `filter2`.

3. Implement FFT-based frequency-domain filtering as shown in the starter code to apply the same filter.

4. Compare the spatial-domain and frequency-domain outputs:

   - visually, by displaying them side-by-side;
   - numerically (optional), by computing the maximum absolute difference between the two results.

5. Repeat the experiment for at least one more image (for example, `rice.png` or `peppers.png`).

Open-Ended Exploration

Use the following prompts as a guide, but feel free to extend them:

- **Smoothing and denoising:** Design or select a low-pass filter and apply it to a noisy image (you may use `imnoise` to add noise). Compare it with `imgaussfilt` using the same approximate blur scale.

- **Edge and detail enhancement:** Construct a high-pass or band-pass filter in the frequency domain (for example, by subtracting a low-pass mask from 1). Apply it to an image and compare the result with Laplacian-based sharpening in the spatial domain.

- **Notch filtering for periodic noise:** Create a simple image with added periodic noise or choose an example where periodic patterns are visible. Design a notch filter in the frequency domain (zeroing out small regions in the spectrum) and evaluate how well the noise is removed.

- **Directional filtering:** Build a frequency mask that selects primarily horizontal or vertical frequencies (for example, a band around the f_x axis). Apply it to an image and observe which structures are emphasised or suppressed.

- **Efficiency considerations:** For a large filter size, measure computation times for spatial-domain filtering and FFT-based filtering using `tic` and `toc`. Comment on when the frequency-domain approach becomes more efficient.

Document your experiments by saving figures or notes for at least two filters and two images.

Reflection Questions

- How similar are the spatial-domain and frequency-domain results for the same filter `h`? Under what conditions do they match closely?
- Which types of image features (smooth regions, edges, textures, periodic patterns) are most clearly controlled in the frequency domain?
- Did you observe any artefacts such as ringing or border effects? How might padding choices or filter design influence these artefacts?
- For your experiments, when did frequency-domain filtering offer a clear advantage over spatial-domain filtering (for example, in terms of control or efficiency)?
- If you had to choose one filter from this activity for a practical application (such as denoising or enhancement), which one would you pick and why?

Summary and Conclusion

This lab introduced some of the key concepts behind frequency-domain methods and demonstrated how they can be used to design, analyse, and apply two-dimensional filters for image processing. The activities guided you from fundamental frequency representations to more open-ended experimentation with real images. This section summarises the main ideas, highlights important observations, and provides directions for further study.

What You Learned

- **Frequency representation of images:** Images can be decomposed into spatial frequencies that correspond to smooth regions, edges, textures, or fine details.
- **Filter design in the frequency domain:** You used tools such as `freqspace`, `fwind1`, `fwind2`, and `fsamp2` to build filters that isolate or suppress specific frequency ranges.
- **Effects of filters on real images:** Low-pass filters smooth images, high-pass filters enhance edges, and band-pass or notch filters target specific structures or periodic noise.
- **Duality between spatial and frequency domains:** Convolution in the spatial domain corresponds to multiplication in the frequency domain, helping you understand when each approach is most effective.

- **Comparison of implementations:** Spatial-domain filtering and FFT-based filtering often produce similar results, but may differ in computational cost and boundary behaviour.

Main Observations From the Lab

- Window shape has a strong impact on the smoothness of the transition band.
- Sampling resolution affects the accuracy and symmetry of the designed filter.
- Frequency-domain filtering is especially useful for removing periodic noise.
- Large filters are more efficient to apply using the FFT than in the spatial domain.
- Sharp masks can introduce ringing, while smoother masks reduce artefacts.

Connections to Modern Machine Learning and Deep Learning

Although many deep learning methods operate in the spatial domain, frequency-domain ideas remain important in modern computer vision pipelines. Some examples include:

- **Fourier-based neural layers:** Some networks include layers that operate in the frequency domain to learn global structure more efficiently.
- **Spectral regularisation and augmentation:** Techniques that modify or constrain frequency components can help networks learn more stable or generalisable features.
- **Hybrid pipelines:** Combining classical frequency filters with learned models can improve denoising, super-resolution, image restoration, and texture analysis.
- **Fourier-domain loss functions:** Losses defined in the frequency domain help ensure that networks preserve both global and local frequency characteristics.
- **Preprocessing for deep learning:** Frequency-domain smoothing, sharpening, or noise removal can improve the quality of features passed to convolutional neural networks.

These connections show that classical frequency-domain techniques remain useful even in modern machine learning systems. In practice, you often design pipelines that combine spatial filters, frequency filters, and learned components to achieve the best results for a specific application.

Extensions and Further Applications

- **Advanced filter design:** Experiment with Gabor filters, steerable filters, or wavelet-based methods for texture and feature extraction.
- **Combining methods:** Use frequency-domain filters alongside spatial operations such as morphology, edge detection, or region-based segmentation.
- **Targeted noise suppression:** Apply notch filters to remove periodic noise from scanned documents, medical images, or patterned backgrounds.
- **Directional analysis:** Enhance horizontal, vertical, or diagonal structures using frequency masks that isolate specific orientations.

Relevance to Communication and Compression

Frequency-domain concepts play an essential role in information transmission and storage:

- **JPEG compression** uses the discrete cosine transform to represent image blocks efficiently.
- **Communication systems** (for example, OFDM) rely on frequency-domain signal structuring to transmit information reliably.
- **Speech and audio coding** depend on spectral shaping and frequency-selective filtering.
- **Data reduction** often exploits the fact that signals have compact or sparse frequency representations.

Understanding frequency-domain ideas will help you make connections across image processing, signal processing, machine learning, and modern communication systems.

Where to Go Next

If you wish to explore these topics further, consider:

- reviewing MATLAB documentation for functions such as `fft2`, `freqz2`, `fsamp2`, `fwind1`, and `fwind2`;
- reading about the discrete Fourier transform, sampling theory, and aliasing;

- exploring wavelets, Gabor filters, and other multiscale representations;
- studying how deep networks incorporate frequency-domain reasoning for tasks such as denoising, restoration, and super-resolution.

These topics will prepare you for advanced work in computer vision, signal processing, and intelligent systems.