Medical Informatics, Radiomics, and Image Analysis for Computer-aided Diagnosis

Course Material for GIAN Program at NITK Surathkal

Foreign Faculty:
Prof. Rangaraj Mandayam Rangayyan,
Professor Emeritus, Schulich School of Engineering, University of Calgary, Canada

Simplified Course Material prepared by:
Dr. Agughasi Victor Ikechukwu,
Department of CSE (Artificial Intelligence), MIT Mysore, India

Course Date: 21 February 2025, Day 5 (Analysis of Oriented Patterns)
Venue: ECES 203 Seminar Hall, South-wing ECE
Course Webpage: https://sites.google.com/view/nitkecelecture/home

Foreword by Prof. Rangaraj Mandayam Rangayyan

"It is my pleasure to welcome you to this session of the GIAN Program at NITK Surathkal on 'Medical Informatics, Radiomics, and Image Analysis for Computer-aided Diagnosis.' This material, focusing on 'Analysis of Oriented Patterns,' illustrates how powerful and instructive it can be to apply practical, hands-on methods in image analysis and pattern recognition. By exploring real-world images—such as those from biomedical imaging, mammography, and materials science—students gain an essential understanding of how to implement and interpret key algorithms for orientation detection, texture characterization, and automated decision support. These practical experiments and exercises enhance our ability to see how theoretical concepts translate into effective tools for diagnosing diseases, studying tissue properties, and designing computer-aided systems that can elevate both research and clinical outcomes. I encourage you to engage with the code snippets, the orientation-based filters, and the transform-based strategies described in this course material. By deepening your appreciation for the interplay of science, engineering, and AI-driven methods, you will be better equipped to innovate in the exciting domains of medical image analysis and radiomics. I wish you a productive and inspiring learning experience."

Contents

1 Introduction

1.1 Motivation for Oriented Pattern Analysis

A broad range of images contain structures exhibiting *directional coherence*. These structures can be fibers, edges, roads, vessels, or elongated ridges. Detecting and quantifying such orientations is critical in fields like:

- **Biomedical imaging**: Collagen fiber alignment in ligaments, vascular structures, muscle fibers, mammography (fibroglandular tissue, ducts, spicules).
- **Remote sensing**: Identifying roads, buildings, farmland alignments in aerial or satellite images.
- **Materials science**: Studying paper grain, textile weaves, carbon fiber composites, or any fibrous material.

Why orientation matters:

1. *Segmentation* of regions: Distinguishing texture patches or structural components based on dominant direction.
2. *Quantitative assessment*: Evaluating how ordered or disordered a region is (e.g., an injured ligament's healing progress).
3. *Early detection*: Subtle distortions or changes in orientation can be an early indicator of anomalies (e.g., architectural distortion in breast images).

1.2 Scope and Applications

This document covers:

- **Foundational approaches** (Fourier-domain fan filtering, gradient-based orientation histograms, Gabor filters),
- **Transform-based** detection (Hough, Radon transforms),
- **Statistical tools** for summarizing the distribution of orientations (moments, entropy),
- **Use cases** in medical imaging and beyond.

2 Image Representations and Preliminaries

2.1 Basic Notation and Definitions

Let $f(x, y)$ represent a 2D grayscale image, where x and y are spatial coordinates. We typically assume:

- $0 \leq x < M$, $0 \leq y < N$ for an $M \times N$ image,
- $f(x, y)$ can be integer (e.g., 8-bit grayscale) or float (normalized).

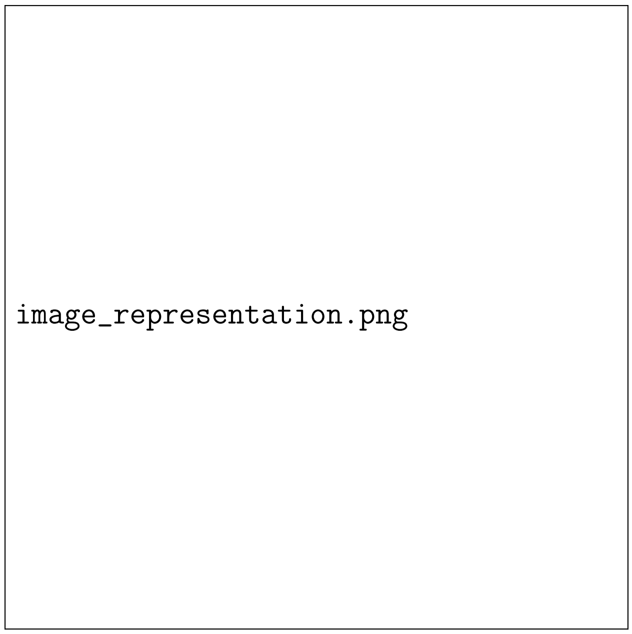

Figure 1: Schematic of a 2D grayscale image $f(x, y)$ with coordinates.

2.2 Intensity, Gradients, and Local Orientation

Local orientation at a pixel (x, y) is often estimated from the *gradients*:

$$G_x = \frac{\partial f}{\partial x}, \quad G_y = \frac{\partial f}{\partial y}.$$

Then the orientation angle θ is:

$$\theta = \operatorname{atan2}(G_y, G_x),$$

which typically ranges from $-\pi$ to π. For orientation analysis, we often map angles to $[0, \pi)$ or $[0°, 180°)$, since directions θ and $\theta + \pi$ can be considered the same orientation.

2.3 Spatial and Frequency Domains

Many oriented structures in the spatial domain show *distinct directional patterns* in the frequency domain. A line at angle ϕ in spatial coordinates transforms to a perpendicular orientation in frequency space, often approximated by a sinc function oriented orthogonally.

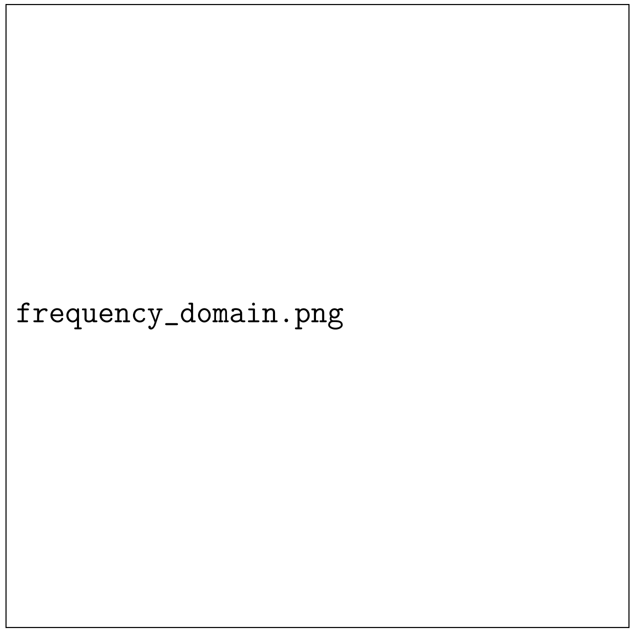

Figure 2: Frequency domain representation of oriented patterns.

3 Orientation Distributions and Rose Diagrams

3.1 Concept of Orientation and Direction Binning

A typical approach to *global* orientation analysis is to bin all local angles into intervals (e.g., 10-degree increments). The result is a histogram $H(k)$, where k indexes the bin. If the orientation is strongly aligned, the histogram has a dominant bin.

3.2 Constructing a Rose Diagram

A *rose diagram* is a polar plot of the orientation histogram. Let:

- $\theta(n)$ be the central angle of bin n,
- $p(n)$ be the normalized fraction of pixels (or sum of gradient magnitudes) having angle $\theta(n)$.

When plotting:

1. Draw a circle with angles from $0°$ to $180°$.
2. At each angle $\theta(n)$, draw a sector with radius proportional to $p(n)$.

This visually resembles flower petals.

rose_diagram.png

Figure 3: Example rose diagram of orientation distribution.

3.3 Example Computation and Visualization

Algorithm (conceptual):

1. Compute local orientation θ_{ij} at each pixel (or at each gradient location).
2. (Optional) Weight each orientation by $\|\nabla f\|$, the gradient magnitude.
3. Discretize angle range $[0, \pi)$ into N bins; accumulate weighted counts.
4. Normalize the resulting histogram $\{p(1), \ldots, p(N)\}$ so $\sum p(n) = 1$.
5. Plot radially to form a rose diagram.

4 Statistical Measures of Orientation

4.1 Angular Moments

Let $\{\theta(n)\}$ be the angle bins, $\{p(n)\}$ be the probabilities or normalized weights. The *angular moment* of order k is:

$$M_k = \sum_{n=1}^{N} [\theta(n)]^k \, p(n).$$

In practice, we often focus on:

- M_1: The *mean angle*,
- M_2: The *second central moment* about M_1.

4.1.1 Mean Angle

$$M_1 = \sum_{n=1}^{N} \theta(n) \, p(n).$$

If orientation is unimodal, M_1 approximates the main direction. If the distribution is multimodal, M_1 alone might be misleading.

4.1.2 Second Central Moment

$$M_2 = \sum_{n=1}^{N} [\theta(n) - M_1]^2 \, p(n).$$

Interpretation:

- *Small* M_2 means a *narrow* spread (high coherence),
- *Larger* M_2 means the angles are more widely scattered.

4.2 Angular Entropy

Entropy measures the *randomness* or *disorder* of an orientation distribution:

$$H = -\sum_{n=1}^{N} p(n) \, \log_2[p(n)].$$

- If all orientation energy is in *one* bin (perfect alignment), $H \approx 0$.
- If orientation is *uniformly* distributed across many bins, H is *maximal*.

4.3 Principal Axis via Spatial Moments

Another approach computes *spatial moments* of the image intensity:

$$m_{pq} = \sum_x \sum_y x^p \, y^q \, f(x, y).$$

Define the *centroid* $\bar{x} = \frac{m_{10}}{m_{00}}$, $\bar{y} = \frac{m_{01}}{m_{00}}$. Then the *central* second moments are:

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p \, (y - \bar{y})^q \, f(x, y).$$

The orientation θ^* at which the *moment of inertia* is minimal satisfies:

$$\tan(2\theta^*) = \frac{2\mu_{11}}{\mu_{20} - \mu_{02}}.$$

Limitation: Captures only a *single* global orientation, ignoring multiple directional components.

4.4 Interpretation of Statistical Orientation Measures

- **Rose diagram**: Provides a *visual* representation.
- **Moments and entropy**: Provide *numeric* descriptors.
- **Principal axis**: Good if the pattern is dominantly oriented in one direction, but not suitable for multi-lobed distributions.

5 Directional Filtering Techniques

5.1 Fourier-Domain Sector (Fan) Filters

A line in the spatial domain $\leftrightarrow$ a sinc function in the frequency domain. This leads to *fan* (or *sector*) filters:

1. Take the 2D FFT of image $F(u, v)$.
2. Define a filter $H(u, v)$ that is $= 1$ inside a wedge (angle $\theta_0 \pm \Delta\theta$) and $= 0$ elsewhere.
3. Multiply: $F'(u, v) = F(u, v) \cdot H(u, v)$.
4. Take inverse FFT to get a partial reconstruction containing only those orientations.

Practical issues:

- Ideal fan filters have *sharp edges* $\implies$ spatial ringing artifacts.
- Butterworth or raised-cosine edges help reduce ringing.

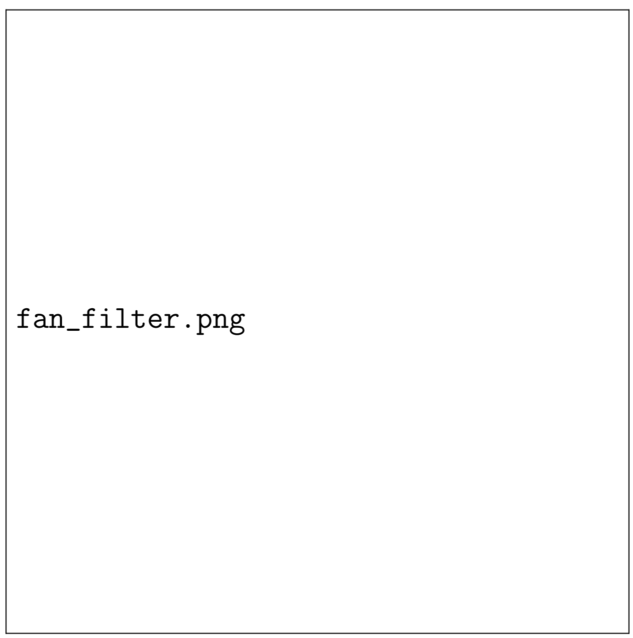

Figure 4: Illustration of a Fourier-domain fan filter.

5.2 Design Challenges and Artifacts

- *Truncation* in the frequency domain may cause spurious edges or rings in the inverse-transformed image.
- Large or abrupt orientation changes can lead to *mixed* bins in histograms if subdivided too finely.

- The DC component often passes through all orientations and can wash out thresholding steps.

5.3 Gabor Filters: Background and Properties

A *Gabor filter* is a Gaussian modulated by a sinusoid. In 2D:

$$g_{\theta,f_0}(x,y) = \exp\left[-\frac{1}{2}(\alpha x'^2 + \beta y'^2)\right]\cos(2\pi f_0 x'),$$

with

$$\begin{cases} x' = x\cos\theta + y\sin\theta, \\ y' = -x\sin\theta + y\cos\theta. \end{cases}$$

- **Local frequency**: f_0,
- **Orientation**: θ,
- **Bandwidth** controlled by α, β.

5.3.1 Gabor vs. Other Filters

- Gabor filters mimic *receptive fields* in the human visual cortex.
- Provide a *joint spatial-frequency* representation.
- Often used in texture segmentation, edge detection, and orientation analysis.

5.4 Tuning Gabor Filters for Multi-Scale, Multi-Orientation Analysis

- Choose a set of *scales* $\{f_1, f_2, \ldots\}$ or $\{\sigma_1, \sigma_2, \ldots\}$.
- Choose a set of *orientations* $\{\theta_1, \theta_2, \ldots, \theta_K\}$.
- Convolve the image with each filter, producing a set of *response images* that highlight different directions/frequencies.

6 Transforms for Oriented Structures

6.1 The Hough Transform for Line Detection

The *Hough transform* for lines parameterizes lines by (ρ, θ):

$$\rho = x \cos\theta + y \sin\theta.$$

For each pixel (x, y) with significant gradient or edge strength, we *vote* for all valid (ρ, θ) pairs. Peaks in *Hough space* indicate strong linear structures.

`hough_transform.png`

Figure 5: Example of Hough transform for line detection.

6.1.1 Applications and Shortcomings

- Effective for binary edges or strong lines.
- May create *spurious* lines if the structure is thick or curved.
- Quantization in ρ, θ can degrade accuracy.

6.2 Radon Transform

A *Radon transform* of an image integrates intensities along radial lines at a set of angles. Essentially:

$$R(\rho, \theta) = \int_{-\infty}^{+\infty} f(\rho\cos\theta - t\sin\theta, \rho\sin\theta + t\cos\theta)\, dt.$$

In practice, discrete approximations exist, and it can be used for tomographic reconstructions or line detection.

6.3 Hough-Radon Hybrid Approaches

- Instead of voting by 1, we *accumulate* the actual intensity or gradient magnitude in the Hough space.
- Reduces random noise.
- Retains strong orientation signals, especially for thick or broad lines.

7 Detailed Use Cases

7.1 Collagen Fiber Analysis in Healing Ligaments

7.1.1 Background

After ligament injury, initial scar tissue is disorganized. Over time, collagen fibers reorient along the original ligament axis. Entropy and spread in orientation can quantify healing progress.

7.1.2 Methodology

1. Acquire SEM images of the ligament.
2. Use Fourier sector filtering or Gabor to isolate oriented fibers.
3. Threshold or measure the amplitude of the filtered response.
4. Construct rose diagram $\rightarrow$ compute entropy.
   - Normal ligaments have *low* entropy (one strong orientation peak).
   - Scar tissue has *higher* entropy, multiple or broad peaks.

7.1.3 Observations

Over a healing period (e.g., 6–14 weeks), entropy typically *decreases* as fibers realign. Joint immobilization or controlled movement can accelerate proper alignment.

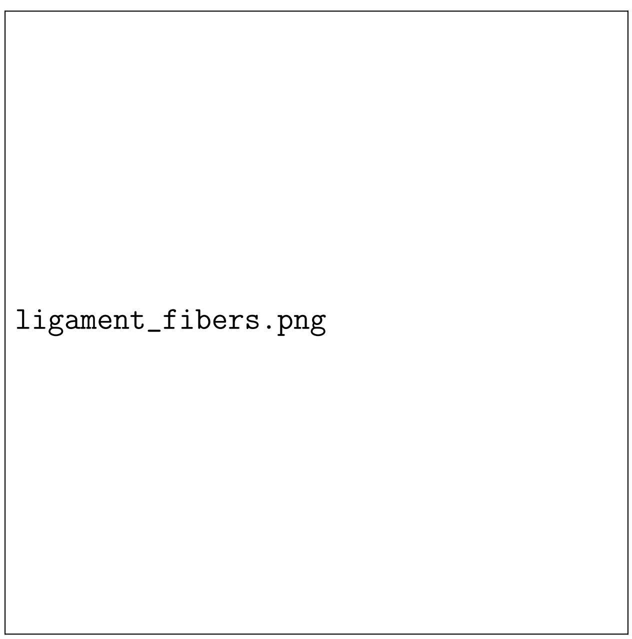

Figure 6: SEM image of collagen fibers in a healing ligament.

7.1.4 Extended Applications and Use Cases

- Medical Rehabilitation: Real-time monitoring of ligament healing in athletes using wearable sensors and orientation analysis, predicting recovery timelines to prevent re-injury (e.g., tracking an athlete's knee ligament recovery after ACL surgery).
- Veterinary Medicine: Analyzing collagen alignment in injured tendons of racehorses, improving treatment plans for faster recovery and performance optimization.
- Research in Tissue Engineering: Studying synthetic scaffolds for tissue regeneration, ensuring fiber orientation mimics natural ligaments for better biomechanical properties.

7.2 Microvascular Structures and Vascular Orientation

7.2.1 Motivation

Blood vessels in healing or pathological tissue may proliferate or appear more disorganized than in healthy tissue.

7.2.2 Skeletonization Approach

1. Threshold the image to isolate vessels.
2. Skeletonize to reduce them to one-pixel-thick lines.
3. Compute the local tangent orientation at each skeleton pixel.
4. Bin orientations, measure coverage.
5. Compare normal vs. pathological orientation distributions.

7.2.3 Extended Applications and Use Cases

- Ophthalmology: Real-time analysis of retinal vessel orientation in diabetic retinopathy patients, detecting early signs of disease progression for timely intervention (e.g., monitoring vessel changes in a patient's retina during an eye exam).
- Cardiovascular Health: Assessing vascular orientation in cardiac imaging to diagnose congenital heart defects, aiding surgeons in planning procedures.
- Environmental Biology: Studying blood vessel patterns in marine organisms (e.g., coral polyps) to assess environmental stress, supporting conservation efforts.

7.3 Architectural Distortion in Mammograms

7.3.1 Definition

Architectural distortion: subtle swirl or radial line patterns *without* a distinct mass. Often missed in routine screening.

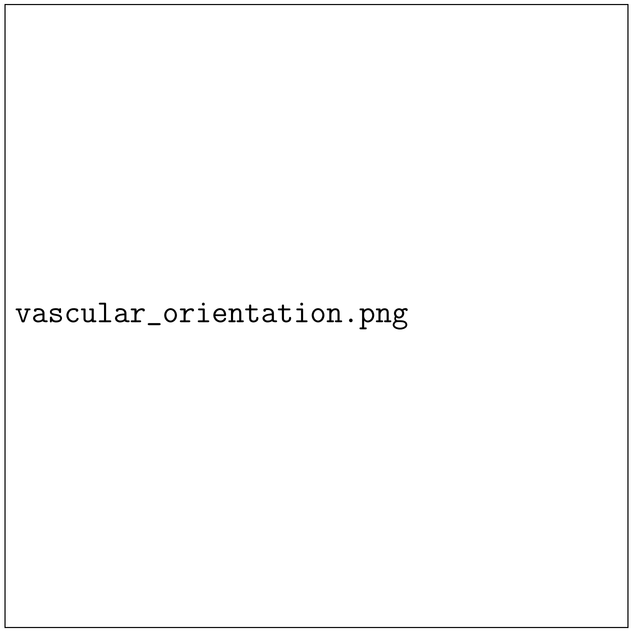

Figure 7: Vessels in a retinal image with orientation analysis.

7.3.2 Algorithmic Detection

1. Segment or crop the fibroglandular disc region.
2. Use a Gabor filter bank to identify oriented edges or spicules.
3. Combine filter responses to build a vector orientation field.
4. Large local changes in orientation or multiple crossing lines can be indicators of distortion.
5. Entropy or angular moment of the difference in orientation distribution (left vs. right breast) can signal asymmetry.

Extended Applications and Use Cases

- Breast Cancer Screening: Real-time mammogram analysis in clinics to detect subtle distortions, reducing false negatives and improving early detection rates for patients (e.g., identifying distortion in a routine mammogram during a health check).
- Medical Training: Training radiologists using orientation-based AI tools to recognize distortion patterns, enhancing diagnostic accuracy in underserved regions.
- Public Health Campaigns: Using orientation analysis in large-scale screening programs to prioritize high-risk patients, improving resource allocation in cancer prevention initiatives.

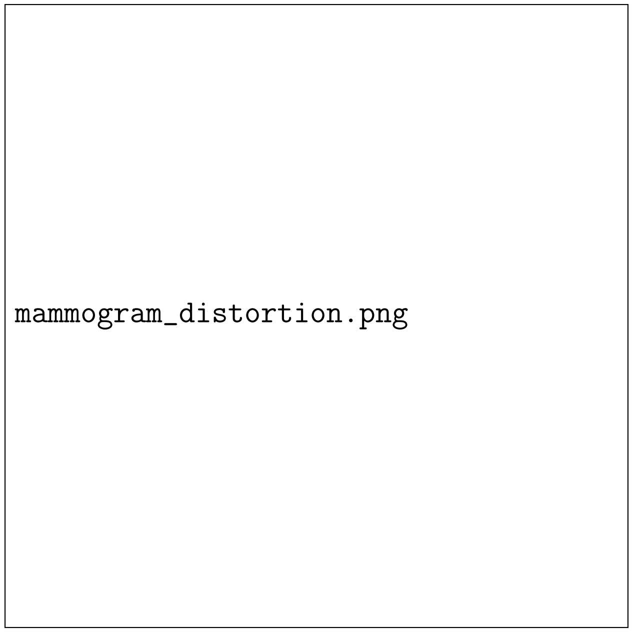

Figure 8: Mammogram showing architectural distortion with orientation analysis.

7.4 Other Applications (Textiles, Remote Sensing)

- **Textiles**: Checking fabric weave uniformity and yarn orientation in real-time quality control systems for textile manufacturing, ensuring consistent product quality (e.g., detecting misaligned threads in a factory).
- **Paper Industry**: Analyzing paper fiber orientation to gauge mechanical strength during production, reducing waste and improving paper durability for packaging (e.g., optimizing paper for shipping boxes).
- **Aerial Photos**: Detecting road alignments and farmland rows in real-time satellite imagery for urban planning and agricultural monitoring, aiding disaster response and crop management (e.g., mapping roads after a natural disaster).

remote_sensing.png

Figure 9: Satellite image showing road and farmland orientation analysis.

8 Mathematical Background and Key Equations

8.1 Spatial Moments and Principal Axes

Recall:

$$m_{pq} = \iint x^p \, y^q \, f(x, y) \, dx \, dy, \quad \bar{x} = \frac{m_{10}}{m_{00}}, \quad \bar{y} = \frac{m_{01}}{m_{00}}.$$

The *central* moments μ_{pq} define rotation invariants, and the minimal moment of inertia axis solves:

$$\tan(2\theta^*) = \frac{2\mu_{11}}{\mu_{20} - \mu_{02}}.$$

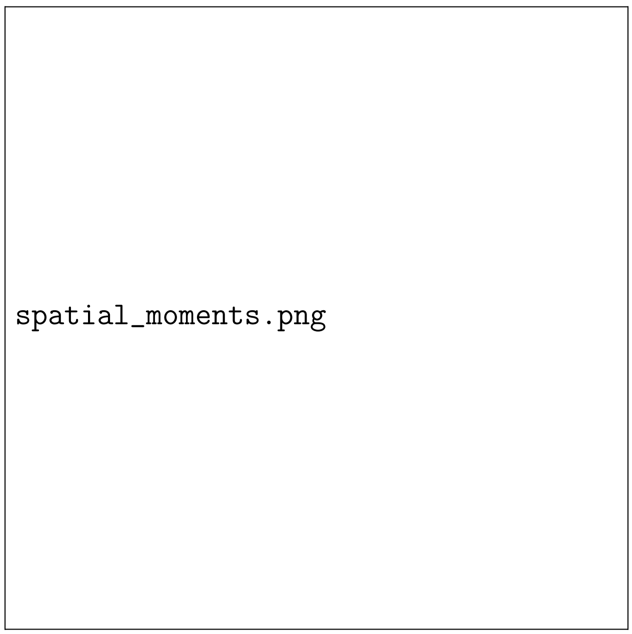

Figure 10: Diagram of spatial moments and principal axes calculation.

8.2 Fourier Transform and Bandpass Filtering

The 2D Fourier transform:

$$F(u, v) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y) \, e^{-j2\pi(ux+vy)} \, dx \, dy.$$

A *fan filter* in frequency space can be an ideal wedge or a smooth radial band:

$$H(u, v) = \begin{cases} 1, & \theta(u, v) \in [\theta_0 - \Delta, \theta_0 + \Delta], \\ 0, & \text{otherwise.} \end{cases}$$

8.3 Angular and Circular Statistics

Circular mean $\bar{\theta}$ and *circular variance* s_θ^2 can be computed via:

$$\bar{\theta} = \operatorname{atan2}\left(\frac{1}{N}\sum_k \sin\theta_k, \frac{1}{N}\sum_k \cos\theta_k\right),$$

$$s_\theta^2 = 1 - \sqrt{\left(\frac{1}{N}\sum_k \cos\theta_k\right)^2 + \left(\frac{1}{N}\sum_k \sin\theta_k\right)^2}.$$

9 Implementation Examples in Python

9.1 Simple Gradient-Based Orientation Estimation

```python
import cv2 # Import OpenCV for image processing
import numpy as np # Import NumPy for numerical operations
import matplotlib.pyplot as plt # Import Matplotlib for visualization

def compute_orientation_map(image): # Define function to compute orientation map
    """
    Compute local orientation at each pixel via Sobel gradients.
    Returns orientation angles in degrees [0..180).
    """
    # Convert image to float32 for gradient computation
    img_float = np.float32(image)
    # Compute horizontal gradient using Sobel operator (dx = 1, dy = 0)
    gx = cv2.Sobel(img_float, cv2.CV_32F, 1, 0, ksize=3)
    # Compute vertical gradient using Sobel operator (dx = 0, dy = 1)
    gy = cv2.Sobel(img_float, cv2.CV_32F, 0, 1, ksize=3)

    # Calculate orientation in radians using arctangent (atan2)
    angles = np.arctan2(gy, gx) # Range: -pi to pi
    # Convert angles to degrees and map to 0180 degrees
    angles_deg = (np.degrees(angles) + 180) % 180
    return angles_deg # Return orientation angles in degrees

# Load a grayscale image for analysis (replace 'sample.png' with actual path)
img = cv2.imread('sample.png', 0)
# Compute orientation map for the loaded image
angles_deg = compute_orientation_map(img)

# Visualize the orientation map using a color map
plt.imshow(angles_deg, cmap='jet') # Display orientation as a color map
plt.colorbar(label='Orientation(degrees)') # Add color bar with label
plt.title('Local-Orientation-Map') # Set plot title
plt.show() # Display the plot
```

9.2 Rose Diagram Computation

```
import cv2 # Import OpenCV for image processing
import numpy as np # Import NumPy for numerical operations
import matplotlib.pyplot as plt # Import Matplotlib for visualization

def rose_diagram(angles_deg, mag=None, num_bins=18): # Define function for rose diagram
    """
    angles_deg: array of orientation angles in degrees [0..180).
    mag: optional array of same shape for weighting by gradient magnitude.
    num_bins: number of bins for [0..180) degrees.
    Returns normalized histogram p(n).
    """
    # Use ones if no magnitude is provided
    if mag is None:
        mag = np.ones_like(angles_deg)
    # Flatten arrays for histogram calculation
    ang_flat = angles_deg.ravel()
    mag_flat = mag.ravel()

    # Compute histogram of angles, weighted by magnitude if provided
    hist, bin_edges = np.histogram(ang_flat, bins=num_bins, range=(0, 180), weights=
        mag_flat)
    # Normalize histogram to sum to 1, avoiding division by zero
    hist_norm = hist / (hist.sum() + 1e-6)
    return hist_norm, bin_edges # Return normalized histogram and bin edges

# Compute gradients for weighting (replace with actual image from previous step)
gx = cv2.Sobel(img, cv2.CV_32F, 1, 0) # Horizontal gradient using Sobel
gy = cv2.Sobel(img, cv2.CV_32F, 0, 1) # Vertical gradient using Sobel
mag = np.sqrt(gx**2 + gy**2) # Calculate gradient magnitude
angles_deg = compute_orientation_map(img) # Get orientation angles

# Compute rose diagram with gradient magnitude weighting
hist_norm, bin_edges = rose_diagram(angles_deg, mag=mag, num_bins=36)

# Calculate center angles for plotting
center_angles = 0.5 * (bin_edges[:-1] + bin_edges[1:])
# Create bar plot for rose diagram
plt.figure()
plt.bar(center_angles, hist_norm, width=5.0) # Plot bars with specified width
plt.title('Rose Diagram of Orientation (Weighted by gradient magnitude)') # Set title
plt.xlabel('Orientation (degrees)') # Label x-axis
plt.ylabel('Normalized Weight') # Label y-axis
plt.show() # Display the plot
```

9.3 Fan Filtering in the Fourier Domain

```python
import cv2 # Import OpenCV for image processing
import numpy as np # Import NumPy for numerical operations

def apply_fan_filter(image, theta_center=0, theta_width=15): # Define fan filter function
    """
    Applies an ideal fan filter around theta_center  theta_width/2 (in degrees).
    Returns the spatial domain image containing primarily that orientation.
    """
    # Get image dimensions
    rows, cols = image.shape
    # Compute 2D Fourier transform of the image
    dft = np.fft.fft2(image)
    # Shift zero frequency to center for easier filtering
    dft_shift = np.fft.fftshift(dft)

    # Initialize mask for fan filter
    crow, ccol = rows // 2, cols // 2 # Center of the image
    mask = np.zeros_like(dft_shift, dtype=np.float32)

    # Convert angles to radians for computation
    theta_center_rad = np.radians(theta_center)
    theta_halfwidth = np.radians(theta_width / 2)

    # Iterate over image to create fan-shaped mask
    for r in range(rows):
        for c in range(cols):
            y = r - crow # Vertical distance from center
            x = c - ccol # Horizontal distance from center
            # Calculate angle in frequency domain
            angle = np.arctan2(y, x)
            # Compute angle difference, handling periodicity
            diff = abs((angle - theta_center_rad + np.pi) % (2 * np.pi) - np.pi)
            # Apply mask if angle is within the fan width
            if diff < theta_halfwidth:
                mask[r, c] = 1.0

    # Apply filter in frequency domain
    filtered_dft = dft_shift * mask
    # Shift back frequency components
    f_ishift = np.fft.ifftshift(filtered_dft)
    # Inverse Fourier transform to return to spatial domain
    img_back = np.fft.ifft2(f_ishift)
    # Take absolute value to get real image
    img_filtered = np.abs(img_back)
    return img_filtered # Return filtered image

# Apply fan filter to the grayscale image (replace with actual image)
img_filtered_0deg = apply_fan_filter(img, theta_center=0, theta_width=30)
```

9.4 Gabor Filter Bank Construction and Application

```
import cv2 # Import OpenCV for image processing
import numpy as np # Import NumPy for numerical operations

def build_gabor_kernels(num_orient=6, scales=[4, 8, 16], ksize=31): # Define function to
    build Gabor kernels
    """
    Returns a list of (kernel, theta, freq) for each orientation, scale.
    """
    kernels = [] # Initialize list to store kernels
    # Iterate over scales
    for scale in scales:
        lambd = scale # Wavelength proportional to scale
        # Iterate over orientations
        for i in range(num_orient):
            theta = np.pi * i / num_orient # Calculate orientation angle in radians
            sigma = 0.5 * lambd # Set Gaussian standard deviation
            gamma = 0.5 # Aspect ratio of the Gaussian envelope
            psi = 0 # Phase offset
            # Generate Gabor kernel using OpenCV
            kernel = cv2.getGaborKernel((ksize, ksize), sigma, theta, lambd, gamma, psi,
                ktype=cv2.CV_32F)
            kernels.append((kernel, theta, scale)) # Store kernel, angle, and scale
    return kernels # Return list of kernels

def apply_gabor_bank(image, kernels): # Define function to apply Gabor filters
    responses = [] # Initialize list for filter responses
    # Apply each kernel to the image
    for (k, theta, scale) in kernels:
        resp = cv2.filter2D(image, cv2.CV_32F, k) # Convolve image with kernel
        responses.append(resp) # Store response
    return responses # Return list of responses

# Load a grayscale image (replace 'ligament.png' with actual path)
img_gray = cv2.imread('ligament.png', 0)
# Build Gabor kernel bank for multi-scale, multi-orientation analysis
kernels = build_gabor_kernels(num_orient=8, scales=[4, 8, 16], ksize=31)
# Apply Gabor filters to the image
responses = apply_gabor_bank(img_gray, kernels)
# Analyze responses for max amplitude or orientation signature (example not shown)
```

9.5 Example Workflow for Oriented Pattern Analysis

1. *Load* the image (e.g., SEM image of a ligament).
2. *Preprocess* (remove noise, maybe enhance contrast).
3. *Apply* directional filtering (Gabor or fan filters).
4. *Threshold* the magnitude of the filtered response to isolate fibers/edges.
5. *Compute* rose diagram or orientation histogram.
6. *Derive* statistics: mean angle, second moment, entropy.
7. *Interpret* results (e.g., if entropy is high, the structure is more disorganized).

Figure 11: Flowchart of oriented pattern analysis workflow.

10 Challenges, Limitations, and Future Directions

10.1 Artifacts and Edge Effects

- Fourier domain filtering can cause *ringing* at high-contrast edges.
- Large or abrupt orientation changes can lead to *mixed* bins in histograms.

10.2 Multi-Modal Orientation Fields

Some images have *multiple strong directions*. Summaries like a single principal axis can be misleading. It may be necessary to detect or group multiple peaks in the orientation distribution.

10.3 Nonlinear and Wavelet-Based Extensions

- Wavelet transforms (like Gabor) can capture orientation at multiple scales with good localization.
- Curvelet or shearlet transforms also handle directional content, especially for curved edges or multi-direction patterns.

10.4 Potential Research Directions

- Improving *robustness* to noise and partial occlusion,
- Automated *multi-scale* orientation classification,
- Using *machine learning* or deep learning that incorporate orientation features (e.g., directional filters embedded in CNNs).

11 Summary and Conclusions

Oriented pattern analysis is a foundational topic in image processing and computer vision. It provides valuable tools for:

- *Segmenting* oriented textures (fibers, edges, spicules),
- *Quantifying* how aligned or chaotic structures are (using entropy, angular moments),
- *Detecting* subtle distortions (architectural distortion in mammography, scarring in ligaments).

Key methods covered:

1. Gradient-based orientation mapping,
2. Rose diagram for visualizing distributions,
3. Fourier fan filtering and Gabor filters for multi-scale orientation detection,
4. Hough and Radon transforms for line- or ridge-like structures.

Through these approaches, both *qualitative* (visual rose diagrams) and *quantitative* (entropy, principal axis) insights into *oriented patterns* can be obtained. These techniques find wide application in *biomedical imaging*, *remote sensing*, *materials science*, and beyond.

12 References and Further Reading

1. R. M. Rangayyan, *Biomedical Image Analysis*, CRC Press (chapters on oriented texture, Gabor filters, and directionality measures).

2. G. Granlund and H. Knutsson, *Signal Processing for Computer Vision*, Kluwer (advanced approaches to local orientation estimation).

3. M. Petrou and P. Bosdogianni, *Image Processing: The Fundamentals*, Wiley (covers morphological and transform-based methods for texture).

4. J. G. Daugman, 'Complete discrete 2D Gabor transforms by neural networks for image analysis and compression,' *IEEE Trans. Acoustics, Speech, Signal Processing*, 1988 (the classic Gabor wavelet approach).

5. X. Li, 'Hough transform for line detection: Implementation details,' *Pattern Recognition*, 1996 (practical discussion of Hough transform details).

6. W. K. Pratt, *Digital Image Processing*, Wiley (general reference to Fourier-based methods for orientation filtering).

Symbols and Notations Used in the Equations

$f(x, y)$: A 2D grayscale image, where x, y denote spatial coordinates.

M, N: The dimensions of the image (M rows, N columns).

G_x, G_y: Gradients of $f(x, y)$ in the x and y directions, respectively.

θ: Local orientation angle, computed as $\theta = \operatorname{atan2}(G_y, G_x)$.

M_k: The k^{th} angular moment of an orientation distribution.

$\theta(n)$: Center angle of the n^{th} orientation bin in a histogram.

$p(n)$: Normalized weight (probability) for the n^{th} orientation bin.

H: Angular entropy (measure of orientation disorder).

m_{pq}: Raw spatial moment of order (p, q) of an image $f(x, y)$.

$\bar{x}$, $\bar{y}$: Coordinates of the centroid, $\bar{x} = \frac{m_{10}}{m_{00}}$, $\bar{y} = \frac{m_{01}}{m_{00}}$.

μ_{pq}: Central moment of order (p, q), accounting for the centroid $(\bar{x}, \bar{y})$.

θ^*: Orientation at which the image's moment of inertia is minimal, satisfying $\tan(2\theta^*) = \frac{2\mu_{11}}{\mu_{20} - \mu_{02}}$.

$F(u, v)$: 2D Fourier transform of $f(x, y)$, with frequency variables (u, v).

$H(u, v)$: Fan-filter mask in the frequency domain, selecting frequencies in an angular wedge.

$g_{\theta, f_0}(x, y)$: 2D Gabor filter with orientation θ and center frequency f_0.

α, β: Scale parameters controlling the Gaussian envelope of a Gabor filter.

x', y': Rotated coordinates for Gabor filters: $x' = x \cos\theta + y \sin\theta$, $y' = -x \sin\theta + y \cos\theta$.

ρ, θ: Parameters for Hough transform lines or Radon transform integration (distance and angle).

$R(\rho, \theta)$: Radon transform of $f(x, y)$ integrating along rays at angle θ and offset ρ.

$\bar{\theta}$: Circular mean of angles, $\bar{\theta} = \operatorname{atan2}\Big(\frac{1}{N} \sum \sin\theta_k, \frac{1}{N} \sum \cos\theta_k\Big)$.

s_θ^2: Circular variance, $1 - \sqrt{\left(\frac{1}{N} \sum \cos\theta_k\right)^2 + \left(\frac{1}{N} \sum \sin\theta_k\right)^2}$.