# Pattern Recognition & Artificial Neural Network in Pattern Recognition

**Unit 7:**

# Image Pattern

- An **image pattern** refers to a *recognizable arrangement or structure of visual elements (pixels, shapes, features)* within an image that conveys meaningful information. It is often defined by a specific combination of:
    - **Color** or intensity
    - **Texture**
    - **Shape or geometry**
    - **Spatial arrangement**

**Examples of Image Patterns:**

- A handwritten digit like “7” in MNIST dataset
- A human face in a photo
- A tumor shape in a medical scan
- A traffic sign in road surveillance images

# Pattern Recognition in Images

- **Pattern Recognition** is the automated process of **classifying or labeling** an input pattern (e.g., an image, object, or feature vector) into one of several predefined categories or classes.
- **It involves:**
- Understanding and extracting **relevant features**
- Applying **machine learning/statistical techniques**
- Making **decisions** based on learned rules

# Image Pattern Recognition System Components

| Step | Description |
|---|---|
| **1. Sensing** | Image acquisition via cameras or sensors |
| **2. Preprocessing** | Noise removal, normalization, contrast enhancement |
| **3. Segmentation** | Dividing image into regions of interest (e.g., separating digits in license plate) |
| **4. Feature Extraction** | Identifying informative elements (edges, corners, texture) |
| **5. Classification** | Using algorithms like neural networks, SVMs, or nearest neighbor to assign a class label |
| **6. Post-processing** | Error correction or smoothing of decision boundaries |

# 7.2 General Steps of Pattern Recognition

**1. Sensing (Data Acquisition)**

- **Description:**
- This is the **first stage**, where data (usually an image or a video frame) is acquired using a **sensor** such as:
- A camera
- Scanner
- Medical imaging device (e.g., MRI, CT scan)
- **Key Points:**
- Determines the quality and resolution of the input.
- Poor sensing can lead to downstream errors.

# 7.2 General Steps of Pattern Recognition

**2. Preprocessing**

- **Description:**
- Involves **enhancing image quality** and making the input suitable for further processing. It helps **reduce noise**, normalize brightness or contrast, and correct distortions.
- **Common Techniques:**
- **Noise removal**: Using filters (Gaussian, median)
- **Normalization**: Adjusting scale or intensity
- **Binarization**: Converting grayscale image to binary
- **Geometric correction**: Straightening skewed documents
- **Purpose:**
- To ensure **consistency** and reduce **irrelevant variability** in the data.

# 7.2 General Steps of Pattern Recognition

- **3. Segmentation**
- **Description:**
- The process of **dividing an image into meaningful regions** or objects (patterns).
- **Types:**
- **Edge-based segmentation**: Detecting boundaries
- **Region-based segmentation**: Grouping similar pixels
- **Thresholding**: Separating foreground from background
- **Example:**
- In face recognition, segmentation separates the face from the background.

# 7.2 General Steps of Pattern Recognition

- **4. Feature Extraction**
- **Description:**
- Extracts **informative attributes** (features) from the segmented regions that are crucial for distinguishing between classes.
- **Types of Features:**
- **Shape-based**: Area, perimeter, compactness
- **Texture-based**: Entropy, energy, contrast
- **Color-based**: Color histogram, dominant color
- **Keypoint-based**: SIFT, SURF, ORB
- **Goal:**
- To transform raw image data into a **concise and meaningful representation**.

# 7.2 General Steps of Pattern Recognition

**5. Feature Selection / Dimensionality Reduction (Optional)**

- **Description:**
- Reduces the number of features by removing **redundant or irrelevant data**, which:
- Improves classifier performance
- Reduces computational load
- **Techniques:**
- PCA (Principal Component Analysis)
- LDA (Linear Discriminant Analysis)

# 7.2 General Steps of Pattern Recognition

- **6. Classification / Decision Making**
- **Description:**
- Classifies the extracted feature vector into one of the **predefined classes** using a decision function or classifier.
- **Types of Classifiers:**
- **Statistical**: Bayes Classifier
- **Geometrical**: Nearest neighbor
- **Machine Learning**: SVM, k-NN, Decision Trees
- **Neural Networks**: Perceptron, CNN, Hopfield Network
- **Output:**
- A **label or class** indicating the pattern type.

# 7.2 General Steps of Pattern Recognition

- **7. Post-Processing (Optional)**
- **Description:**
- Improves the final results using:
- **Error correction**
- **Label smoothing**
- **Validation against constraints**
- **Example:**
- If a sequence of characters is recognized, a spell-check may refine the result.

# Example Workflow: Handwritten Digit Recognition

| Step | Description |
| --- | --- |
| Sensing | Scanning a handwritten digit |
| Preprocessing | Grayscale conversion, denoising |
| Segmentation | Isolating each digit |
| Feature Extraction | Extracting stroke orientation, shape, pixel intensity |
| Classification | Using an ANN or k-NN to identify digit |
| Output | Digit label (e.g., "3") |

# Why Follow These Steps

- Each step **modularizes** the process and isolates concerns.
- Errors in earlier stages **propagate**, so robust design is key.
- Flexible architecture: steps can be adapted for text, audio, images, or video.

# 7.3 Boundary Preprocessing, Boundary Feature Descriptors, Region Feature Descriptors

- Pattern recognition in image analysis often involves identifying and classifying objects based on their **shape and internal characteristics**. For this, two primary types of descriptors are used:
- **Boundary (Contour) descriptors**: Analyze the object outline.
- **Region descriptors**: Analyze the entire area enclosed by the boundary.
- Before extracting features, **preprocessing** is crucial to ensure accuracy.

# Boundary Preprocessing

- **Definition:**
- Boundary preprocessing involves preparing object contours for reliable feature extraction by reducing noise and simplifying shapes.
- 🔶 **Objectives:**
- Remove noise or irregularities
- Smooth contours
- Simplify boundary shapes for analysis

# Boundary Feature Descriptors

- These are features extracted **along the contour or edge** of an object, focusing on its **shape** and **outline**.

| Descriptor | Description | Example Use |
|---|---|---|
| **Perimeter** | Total length of boundary | Object size |
| **Chain Code** | Encodes contour using direction codes | Shape representation |
| **Curvature** | Rate of change of direction along boundary | Detect corners or inflection points |
| **Fourier Descriptors** | Converts boundary to frequency domain | Shape matching and normalization |
| **Shape Signature** | Radial distance from centroid vs. angle | Compact shape descriptor |

- **Example: Chain Code**
- Represents a boundary by connecting pixels using **4- or 8-directional codes**.
- E.g., for an edge: → = 0, ↑ = 1, ← = 2, ↓ = 3 (4-directional)
- **Example: Fourier Descriptors**
- Apply Discrete Fourier Transform (DFT) on boundary coordinates.
- Invariant to translation, rotation, and scaling.

# Region Feature Descriptors

- These descriptors analyze the **entire region inside a boundary**, focusing on area-based properties.

| Descriptor | Description | Example Use |
|---|---|---|
| **Area** | Number of pixels in the object | Size estimation |
| **Centroid** | Geometric center of the region | Object location |
| **Eccentricity** | Ratio of major to minor axis | Shape elongation |
| **Moment Invariants (Hu moments)** | Describe shape using statistical moments | Object recognition invariant to rotation |
| **Compactness** | (Perimeter2)/Area(\text{Perimeter}^2)/\text{Area} | Roundness metric |
| **Texture Features** | Energy, entropy, homogeneity (from GLCM) | Region texture analysis |

# Patterns and Pattern Classes

- **What is a Pattern?**
- A **pattern** is a set of **observations or data points** that can be identified and grouped based on certain **characteristic features or properties**. In the context of **image processing and pattern recognition**, a pattern typically refers to a visual structure that represents some meaningful object or concept.
- 🔶 **Example:**
- A handwritten digit “5”
- A fingerprint image
- A cat in a photograph
- A traffic sign in a street image
- Each of these can be **seen as a pattern** in an image that we want to detect, analyze, and classify.

# Characteristics of a Pattern

| Characteristic | Description |
|---|---|
| **Features** | Numerical or categorical attributes extracted from data |
| **Structure** | Spatial or geometric arrangement (shape, edges) |
| **Class Membership** | Belongs to a group of similar patterns |

# What is a Pattern Class?

- A **Pattern Class** (also called a **Category** or **Label**) is a group or set of patterns that share **common features** and are treated as **equivalent** for the purpose of classification.
- 🔶 **Examples:**
- All images of the digit “5” → belong to the **class ‘5’**
- All images of cars → belong to the **class ‘vehicle’**
- All “stop” signs → belong to the **class ‘traffic_stop_sign’**
- Each pattern class defines **what the pattern represents** and is used by classifiers to assign labels to new input patterns.

# Types of Patterns

| Type | Description | Example |
|---|---|---|
| **Discrete** | Has distinct values | Letters A–Z, digits 0–9 |
| **Continuous** | Represented with real-valued features | Voice signal, temperature data |
| **Statistical** | Vary with probability distributions | Texture patches in images |
| **Structural** | Composed of sub-patterns with rules (grammar) | Face (eyes, nose, mouth) |

# Pattern Representation

- Patterns are typically represented in **feature space** (also known as **pattern space**), where each pattern is a **feature vector**:
- $\vec{x}$=[x1,x2,x3,...,xn] where x1,x2,x3 are the features (like width, height, color intensity, texture).

# Pattern Recognition Goal

- To design a system that can:

1. **Extract features** from a new input pattern
2. **Match** it against known classes using a classifier
3. **Label** the pattern with the most likely class

# Pattern Class Example

**Example: Handwritten Digit Recognition**

| Input Image | Feature Vector (e.g., pixel values, stroke angles) | Pattern Class |
|---|---|---|
| | [0.12, 0.89, 0.65, …] | Digit ‘5’ |
| | [0.34, 0.78, 0.67, …] | Digit ‘8’ |

# Pattern Class Separability

- A good pattern recognition system ensures that:
- **Patterns from the same class** cluster together in feature space
- **Patterns from different classes** are well-separated
- **Visual Representation (Feature Space)**
- Pattern Class 1: ● ● ● ●
- Pattern Class 2: ▲ ▲ ▲ ▲
- Pattern Class 3: ■ ■ ■ ■

# Applications of Pattern and Pattern Classes

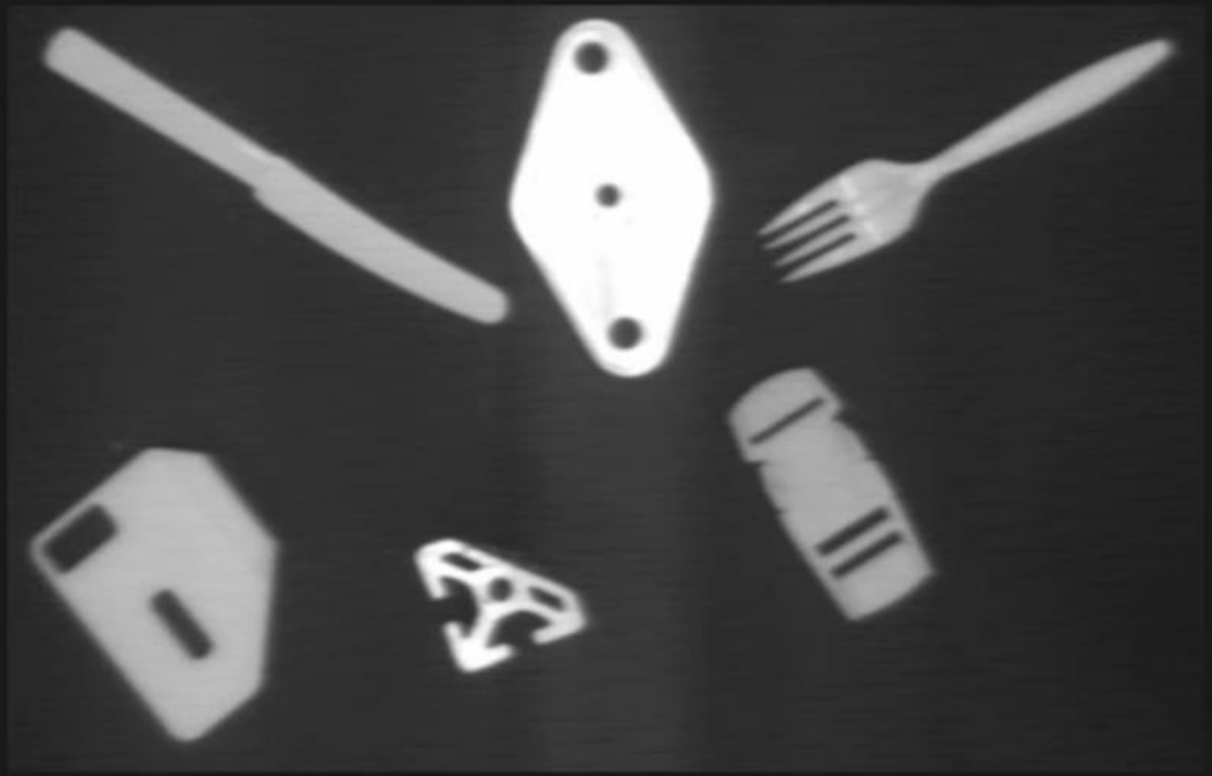
| Field | Example Pattern Classes |
|---|---|
| **Biometrics** | Fingerprint, iris, face |
| **Medical Imaging** | Tumor types: benign, malignant |
| **Traffic Systems** | Sign types: stop, yield, speed limit |
| **Document Analysis** | Characters: A-Z, digits, symbols |

# Scale-Invariant Feature Transform (SIFT)

- **SIFT** is a widely used feature detection and description algorithm developed by **David Lowe** in 1999. It is capable of detecting **distinctive and invariant key-points** in images for tasks such as object recognition, image stitching, and motion tracking.
- **Goal**: Extract **key-points** (interest points) that are **invariant** to:
- **Scale**
- **Rotation**
- **Translation**
- **Affine transformation**
- **Illumination changes**

# A Little Quiz

**How would you recognize the following types of object?**



Objects on an assembly line

# A Little Quiz

**How would you recognize the following types of object?**


License plates

# A Little Quiz

**How would you recognize the following types of object?**

# Why SIFT?

- **Challenges in Feature Matching:**
- Different camera zoom levels (scale change)
- Rotation of the object in the image
- Lighting variations
- Partial occlusion
- Noise
- SIFT handles these challenges by identifying features that remain **consistent** across transformations.

# Steps of the SIFT Algorithm

🔶 **Step 1: Scale-Space Extrema Detection**

- Construct a **scale-space representation** using **Gaussian filters** at different scales.
- Compute the **Difference of Gaussian (DoG)**:

$$\mathrm{DoG}(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$$

  Where $L$ is the image convolved with a Gaussian of scale $\sigma$.

- Detect **keypoints** as local maxima/minima in the DoG scale-space by comparing with neighbors in 3×3×3 grid.

# Steps of the SIFT Algorithm

## 🔶 Step 2: Keypoint Localization

- Fit a 3D quadratic function to determine the accurate position and scale of each keypoint.
- Reject unstable keypoints with **low contrast** or those poorly localized along edges (using Hessian matrix analysis).

# Steps of the SIFT Algorithm

## 🔶 Step 3: Orientation Assignment

- Calculate the **gradient magnitude** and **orientation** around each keypoint:

$$m(x, y) = \sqrt{(L_x)^2 + (L_y)^2}, \quad \theta(x, y) = \tan^{-1}\left(\frac{L_y}{L_x}\right)$$

- Build an **orientation histogram** (36 bins) and assign the dominant orientation.
- This allows the descriptor to be **rotation-invariant**.

# Steps of the SIFT Algorithm

## 🔶 Step 4: Keypoint Descriptor Generation

- Around the keypoint, construct a 16×16 neighborhood.
- Divide into 4×4 subregions.
- In each subregion, create an **8-bin orientation histogram.**
- Concatenate all 16 histograms → **128-dimensional descriptor**.

# Applications of SIFT

| Area | Use Case |
| --- | --- |
| Object Recognition | Detecting and identifying objects in cluttered scenes |
| Image Stitching | Finding overlapping features for panorama creation |
| 3D Reconstruction | Matching keypoints across images to derive depth |
| Robot Navigation | Landmark detection for SLAM |
| Medical Imaging | Matching anatomical structures across scans |