# Face Detection & Recognition Algorithms

## 🔍 Part 1: **Face Detection with Haar Cascades**

### ✅ What is Haar Cascade?

Haar Cascade is an **object detection algorithm** that detects objects like **faces**, **eyes**, **cars**, etc. It was proposed by **Viola and Jones** in 2001.

### ⚙️ How Haar Cascade Works (Step-by-Step)

#### 1. **Haar Features Extraction**

* Haar-like features are simple patterns like:
  + Edge features
  + Line features
  + Rectangle patterns

#### 2. **Integral Image**

* Speeds up calculation by storing pixel sums in a special way.
* Allows computing Haar features in constant time.

#### 3. **Adaboost Training**

* Selects the most **important** features using a technique called **Adaboost**.
* Combines weak classifiers into a strong classifier.

#### 4. **Cascade of Classifiers**

* Applies multiple stages:
  + Fast reject non-faces in early stages
  + Apply complex filters only on likely face areas
* **Improves speed** by discarding most regions early.

### Manual Insight

When you run detectMultiScale():

* It scans the image at multiple scales and locations.
* For each region:
  + Computes Haar features.
  + Uses cascade model to classify face or not.

✅ **Only regions passing all stages are labeled as “face.”**

## Code Recap (Simplified)

faces = face\_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5)

## Part 2: **Face Recognition with LBPH**

### What is LBPH?

**Local Binary Patterns Histogram (LBPH)** is a **texture-based** face recognition technique. It doesn't look at **color or shape**, but **texture patterns**.

### ⚙How LBPH Works (Step-by-Step)

#### 1. **Divide Image into Cells**

* Convert image to grayscale and split into small grids (e.g. 8x8).

#### 2. **Compare Pixels with Neighbors**

For each pixel in a cell:

* Compare it with its 8 surrounding pixels.
* If neighbor ≥ center pixel → write 1, else 0.

Example:

Center pixel = 100  
Neighbors: [110, 90, 130, 80, 105, 120, 70, 95]  
Binary: [ 1 , 0 , 1 , 0 , 1 , 1 , 0 , 0 ] → 10101100 → Decimal = 172

This number becomes the **LBP code** of that pixel.

#### 3. **Build Histograms**

* For each cell, count the occurrence of each LBP code.
* Combine all histograms → **LBP feature vector**

#### 4. **Compare with Training Histograms**

* Calculate **distance (Euclidean or Chi-Square)** between the test image’s histogram and training histograms.
* Return the **closest match** (minimum distance).

### Manual Insight

When you run recognizer.predict():

* It transforms the test face into a histogram.
* Compares it with each trained histogram.
* Returns:
  + **Label (person ID)**
  + **Confidence (lower = better match)**

## 🔄 Code Recap (Simplified)

label, confidence = recognizer.predict(face\_image)

## ⚖️ Summary Table

| Feature | Haar Cascade | LBPH |
| --- | --- | --- |
| Used for | Face Detection | Face Recognition |
| Algorithm Type | Machine learning, boosted stages | Pattern-matching (histograms) |
| Input | Any image | Cropped grayscale face image |
| Output | Face location (x, y, w, h) | Person ID, confidence score |
| Speed | Very fast | Slower but accurate |
| Handles Lighting | Poorly | Well |
| Custom Training | Not possible (pre-trained only) | Yes (you train with images) |

## 💡 Visual Summary (Text)

**Haar Cascade** 📷 Image → 🔲 Sliding Window → 📐 Haar Features → ✅ Face Detected

**LBPH** 🖼️ Grayscale Face → 💠 LBP Codes → 📊 Histograms → 🔎 Match with Trained