**FACE DETECTION AND RECOGNITION**

**USING HCC & LBHP ALGORITHMS**

**ABSTRACT:**

The Face Detection and Recognition using HCC and LBHP Algorithms project is designed to provide an efficient and accurate real-time facial recognition system. This web-based application integrates Haar Cascade Classifier (HCC) for face detection and Local Binary Histogram Patterns (LBHP) for face recognition. The combination of these algorithms ensures a robust approach to identifying individuals in dynamic environments.

Haar Cascade Classifier (HCC) is a machine learning-based approach that detects faces by analyzing pixel intensity differences in predefined rectangular regions. It employs integral images for fast computation and uses AdaBoost to select the most relevant Haar-like features, enabling rapid and efficient face detection. The classifier consists of multiple stages arranged in a cascade, filtering out non-face regions early to improve processing speed.

Local Binary Histogram Patterns (LBHP) is an extension of Local Binary Patterns (LBP), which enhances facial recognition by encoding local texture information. LBHP divides a facial image into small grids, computes local binary patterns, and generates histograms representing the distribution of pixel intensities. The histograms are then compared using Euclidean distance or Chi-square distance to identify the most similar face in the database. This method is computationally efficient and performs well under varying lighting conditions and facial expressions.

The system is implemented as a Django-based web application, utilizing Python and OpenCV for image processing. The backend, powered by Django, manages user data, stores extracted facial features, and processes recognition requests, while the frontend, developed using HTML, CSS, and JavaScript, provides an intuitive user interface. The application continuously captures video input, detects faces in real-time using HCC, extracts LBHP features, and compares them against a pre-existing database for identification.

**TABLE DESIGN:**

1. **User Table**

|  |  |
| --- | --- |
| **Column Name** | **Data Type** |
| User\_Id | INTEGER |
| Name | VARCHAR (50) |
| Registered\_On | DATETIME |

1. **Face Data Table**

|  |  |
| --- | --- |
| **Column Name** | **Data Type** |
| Face\_Id | INTEGER |
| User\_Id | INTEGER |
| Lbhp\_Features | BLOB |
| Image\_Path | VARCHAR (30) |
| Added\_On | DATETIME |

**PROJECT DESCRIPTION:**

The project "Face Detection and Recognition using HCC and LBHP Algorithms" aims to develop an efficient and accurate facial recognition system. This system is designed to identify individuals in real-time using a combination of Haar Cascade Classifier (HCC) for face detection and Local Binary Histogram Patterns (LBHP) for face recognition. The project is implemented as a web-based application using Django, Python, and OpenCV.

**OBJECTIVES:**

* To develop a robust face detection and recognition system.
* To implement real-time face detection using Haar Cascade Classifier (HCC).
* To enhance facial recognition accuracy using Local Binary Histogram Patterns (LBHP).
* To create a user-friendly web interface for managing and recognizing faces.

**METHODOLOGY:**

1. **Face Detection using Haar Cascade Classifier (HCC):**
   * HCC is a machine learning-based approach that detects faces by analyzing pixel intensity differences in predefined rectangular regions.
   * It employs integral images for fast computation and uses AdaBoost to select the most relevant Haar-like features.
   * The classifier consists of multiple stages arranged in a cascade, filtering out non-face regions early to improve processing speed.
2. **Face Recognition using Local Binary Histogram Patterns (LBHP):**
   * LBHP is an extension of Local Binary Patterns (LBP) that enhances facial recognition by encoding local texture information.
   * LBHP divides a facial image into small grids, computes local binary patterns, and generates histograms representing the distribution of pixel intensities.
   * The histograms are compared using Euclidean or Chi-square distance to identify the most similar face in the database.
3. **System Implementation:**
   * The system is implemented as a Django-based web application.
   * The backend, powered by Django, manages user data, stores extracted facial features, and processes recognition requests.
   * The frontend, developed using HTML, CSS, and JavaScript, provides an intuitive user interface.
   * The application continuously captures video input, detects faces in real-time using HCC, extracts LBHP features, and compares them against a pre-existing database for identification.

**FEATURES:**

* **Real-Time Face Detection:** The system captures video input and detects faces in real-time using the Haar Cascade Classifier.
* **Accurate Face Recognition:** The system uses Local Binary Histogram Patterns to recognize faces with high accuracy.
* **User Management:** The web application allows users to upload and manage facial images.
* **Video Processing:** The system can process video files to detect and recognize faces at different timestamps.
* **Intuitive Interface:** The frontend provides a user-friendly interface for interacting with the system.

**MODULES:**

* Home
* About
* Live Face Recognition
  + Single Face Recognition
  + Multi Face Recognition
* Pre-Recorded File Recognition
  + Single File Recognition
  + Multi File Recognition

**DATA FLOW DIAGRAM:**

**LEVEL 0:**

Start

Home

About

Live Recognition

File Recognition

Multi Face Recognition

Single Face Recognition

Single File Recognition

Multi File Recognition

**LEVEL 1:**

User inputs test images

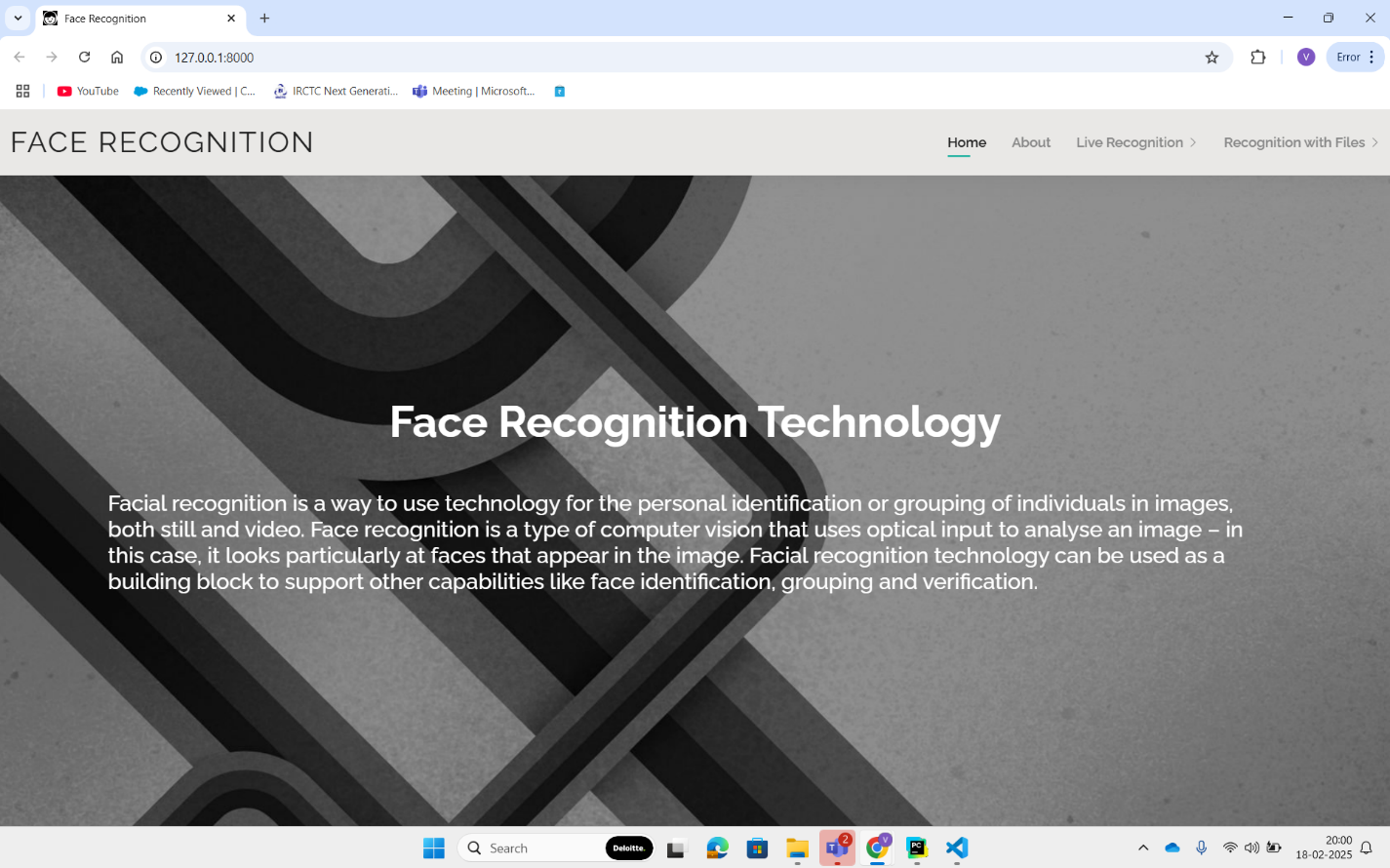
Extract facial features

Compare with existing face data

Show the result in feed

**LEVEL 1:**

**SAMPLE SCREENSHOTS:**

****

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated