HUMAN FACE IDENTIFICATION AFTER PLASTIC SURGERY

MINOR PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT FOR THE AWARD OF DEGREE OF

BACHELOR OF TECHNOLOGY
Computer Science & Technology

Submitted By:

Harsh (2203455)
Karan Kashyap (2203481)
Manish Choudhary (2203495)

Submitted To:

Prof. Diana Nagpal
Department of Computer Science
GNDEC Ludhiana

GURU NANAK DEV ENGINEERING COLLEGE, LUDHIANA, 141006
May 2025

Abstract

Facial recognition technology plays a vital role in identity verification across domains such as security, forensics, and healthcare. However, one of its most significant challenges arises when an individual's facial features are altered due to cosmetic or reconstructive surgery. Traditional face recognition systems often fail to accurately identify such individuals, leading to serious implications in high-security environments.

This project presents a lightweight and reliable facial recognition system specifically designed to handle the complexities introduced by plastic surgery. The backend leverages MTCNN for robust face detection, followed by feature extraction using Local Binary Patterns (LBP) and Histogram of Oriented Gradients (HOG). These extracted features are processed through a Random Forest classifier optimized via GridSearchCV to deliver high accuracy while maintaining computational efficiency.

To make the system user-friendly and accessible, a modern frontend interface has been developed using React and TypeScript. The interface enables users to upload before and after images, initiate comparisons, and receive results in real-time. The application architecture supports RESTful APIs, image preprocessing, and secure image handling to ensure end-to-end functionality.

With a validation accuracy of 90%, this system demonstrates strong resilience to facial modifications and provides a practical solution for real-world deployment. The report outlines the architecture, implementation details, testing results, and potential enhancements of the complete system.

Acknowledgment

We are highly grateful to the Dr. Sehijpal Singh, Principal, Guru Nanak Dev Engineering College (GNDEC), Ludhiana, for providing this opportunity to carry out the minor project work at Guru Nank Dev Engineering College (GNDEC).

The constant guidance and encouragement received from Dr. Kiran Jyoti H.O.D. CSE Department, GNDEC Ludhiana has been of great help in carrying out the project work and is acknowledged with reverential thanks.

We would like to express a deep sense of gratitude and thanks profusely to Er. Diana Nagpal, without her wise counsel and able guidance, it would have been impossible to complete the project in this manner.

Finally, we acknowledge the developers and contributors of the open-source libraries and frameworks used in this project. Their work formed the foundation of our implementation and accelerated our development process.

This project has been a significant learning experience, and we are truly grateful for the collaborative environment that made it possible.

Harsh

Karan Kashyap

Manish Choudhary

Contents

1 Introduction **1**

1.1 Project Overview 1

1.2 Project Category 2

1.3 Problem Formulation 2

1.4 Identification of Need 3

1.5 Existing System 4

1.5.1 Limitations of Existing Systems 5

1.6 Objectives 5

1.7 Proposed System 6

1.7.1 System Architecture 6

1.7.2 Image Processing Pipeline 6

1.7.3 Feature Extraction 6

1.7.4 Classification and Decision Making 7

1.7.5 System Deployment and Frontend 7

1.7.6 Summary 7

2 Requirement Analysis and System Specification **8**

2.1 Feasibility Study 8

2.1.1 Technical Feasibility 8

2.1.2 Economic Feasibility 8

2.1.3 Operational Feasibility 9

2.2 Software Requirement Specification (SRS) 9

2.2.1 Data Requirements 9

2.2.2 Functional Requirements 9

2.2.3 Performance Requirements 10

2.2.4 Dependability Requirements 10

2.2.5 Maintainability Requirements 10

2.2.6 Security Requirements 10

2.2.7 Look and Feel Requirements 10

2.2.8 Software Development Life Cycle (SDLC) Model . . . 12

3 System Design **14**

3.1 Overview . . . 14

3.2 System Architecture . . . 14

3.2.1 Frontend Architecture . . . 14

3.2.2 Backend Architecture . . . 14

3.3 Data Flow Diagram . . . 15

3.4 Module Design . . . 17

3.4.1 Frontend Modules . . . 17

3.4.2 Backend Modules . . . 17

4 Methodology **18**

4.1 Introduction to Languages, IDEs, Tools Used . . . 18

4.1.1 Algorithm Selection: Comparative Analysis . . . 20

4.1.2 Algorithm Used . . . 22

4.1.3 Detailed Workflow . . . 23

4.1.4 Dataset Organization . . . 23

4.1.5 Face Detection using MTCNN . . . 24

4.1.6 Pair Generation and Labeling . . . 24

4.1.7 Feature Extraction with LBP and HOG . . . 24

4.1.8 Data Splitting . . . 25

4.1.9 Classification with Random Forest . . . 26

4.1.10 Feature Standardization . . . 26

4.2 Implementation . . . 27

4.2.1 Frontend Implementation . . . 27

4.2.2 Backend Implementation . . . 27

4.3 Testing . . . 28

4.3.1 Unit Testing . . . 28

4.3.2 Integration Testing . . . 28

4.3.3 Functional Testing . . . 29

4.3.4 Test Cases . . . 30

5 Results and Discussions 31

5.1 Performance Metrics 31

5.2 Confusion Matrix and ROC Curve 33

5.3 Performance Analysis 34

5.3.1 Effectiveness in Plastic Surgery Applications 34

5.3.2 Comparison with Traditional Systems 34

5.4 Comparative Study 35

5.4.1 Overview of Existing Techniques 35

5.4.2 Proposed Method: LBP + HOG + Random Forest 35

5.4.3 Comparative Summary 36

5.4.4 Conclusion 36

5.5 Limitations 36

5.6 Output and Interface 37

5.6.1 Image Upload Interface 37

5.6.2 Comparison Result Display 38

5.6.3 Processed Faces Preview 38

5.6.4 Error Handling 39

6 Conclusion and Future Scope 40

6.1 Conclusion 40

6.2 Future Scope 40

6.2.1 Deep Learning Integration 40

6.2.2 Real-time Processing and Mobile Integration 40

6.2.3 User Interface Enhancements 41

6.2.4 Expanded Dataset 41

A Appendix 44

List of Figures

2.1 Block Diagram of the Facial Recognition System 11

2.2 Waterfall Model Diagram 12

3.1 Level 0 Data Flow Diagram 16

3.2 Level 1 Data Flow Diagram 16

3.3 Level 2 Data Flow Diagram 17

4.1 Methodology Used 23

5.1 Confusion Matrix of the Facial Recognition System 33

5.2 ROC Curve for the Facial Recognition System 33

5.3 Frontend interface for uploading before and after facial images 37

5.4 Output screen showing prediction result and confidence score 38

5.5 Detected and aligned facial region as processed by the backend 38

5.6 Example of error message when no face is detected in the image 39

5.7 Example of input validation error for unsupported file format 39

A.1 Plagiarism Check Report for Methodology 44

List of Tables

4.1 Test Cases for Facial Recognition System 30

5.1 Comparison of Facial Recognition Methods Under Plastic Surgery 36

Chapter 1: Introduction

1.1 Project Overview

Facial recognition is a widely adopted biometric technology used in diverse applications such as surveillance, law enforcement, access control, and identity verification. Its popularity stems from its non-intrusive nature and the uniqueness of facial features across individuals. However, the reliability of facial recognition systems is challenged by facial alterations caused by plastic surgery, which may significantly change key biometric features. Such changes affect texture, geometry, and structural patterns used by most facial recognition algorithms, leading to a decline in recognition accuracy.

Plastic surgery is increasingly common for both cosmetic and reconstructive purposes. As a result, traditional facial recognition systems struggle to identify individuals who have undergone significant surgical modifications, making such scenarios a critical challenge in the field of biometrics. Conventional methods that rely on stable facial landmarks — such as eye shape, nose contour, and jawline — often fail when these landmarks are altered.

This project proposes a lightweight facial recognition system designed to verify a person's identity even after plastic surgery. Unlike deep learning methods that demand high computational resources, this system leverages traditional machine learning techniques for efficient and interpretable recognition. It combines Local Binary Patterns (LBP) and Histogram of Oriented Gradients (HOG) for robust feature extraction and uses a Random Forest classifier for identity prediction. This approach ensures faster inference times, reduced dependency on large datasets, and ease of deployment.

The system is trained and evaluated using the HDA Plastic Surgery database — a dataset of paired facial images taken before and after various types of surgery. The dataset was obtained directly from the authors and is used exclusively for academic research purposes.

This report presents the problem formulation, related work, proposed solution architecture, implementation methodology, and evaluation results. The ultimate goal is to build a reliable, resource-efficient system capable of identity verification under real-world conditions where facial modifications are present.

This project presents a hybrid facial recognition system capable of identifying individuals before and after plastic surgery. It combines traditional machine learning techniques with modern web technologies to deliver a reliable and user-friendly verification system. The backend utilizes MTCNN for face detection, LBP and HOG for feature extraction, and a Random Forest classifier for identity verification. A React-based frontend interface allows users to interact with the system seamlessly through a browser.

1.2 Project Category

This project falls under the category of **Research-Based Application Development**. It combines both theoretical investigation and practical implementation to solve a real-world biometric challenge: identifying individuals before and after undergoing plastic surgery using facial recognition techniques.

From a research perspective, the project investigates the limitations of traditional face recognition systems when facial features are altered surgically. It explores established feature extraction techniques such as Local Binary Patterns (LBP) and Histogram of Oriented Gradients (HOG), and evaluates their resilience against post-surgical facial changes. Furthermore, it compares the proposed lightweight model against existing deep learning-based systems in terms of efficiency, interpretability, and robustness.

From an application perspective, the system is developed as a complete end-to-end pipeline — from face detection and preprocessing to classification and user interface deployment. The backend is implemented using Python and Flask, while the frontend is built using React and TypeScript to provide an intuitive user experience. Model training was conducted using Azure Machine Learning services to enable scalable and controlled experimentation.

Hence, the project contributes both to academic understanding of facial feature stability after surgical modification and to the development of a functional application that can be extended for use in medical, forensic, or security domains.

1.3 Problem Formulation

Facial recognition systems are widely deployed for identity verification due to their accuracy and convenience. However, their reliability significantly degrades when an individual undergoes plastic surgery, which alters key biometric traits such as skin texture, facial structure, and feature geometry.

These changes often disrupt the performance of conventional face recognition algorithms that rely heavily on fixed visual cues.

Plastic surgery introduces two categories of alterations: local (e.g., eyelid surgery, rhinoplasty) and global (e.g., facelift, brow lift). Local surgeries affect individual facial components, while global surgeries can modify the entire facial structure. Both forms of surgery pose major challenges to face recognition systems, especially those dependent on traditional feature-matching approaches or deep learning models trained on pre-surgery facial data.

The problem addressed in this project is to develop a facial recognition system that can accurately determine whether two facial images — taken before and after plastic surgery — belong to the same individual, despite significant appearance alterations.

This research aims to develop a robust, lightweight facial recognition system that utilizes handcrafted features and traditional machine learning techniques to handle facial variations introduced by plastic surgery. It seeks to balance performance and interpretability while minimizing dependency on high-resource hardware or extensive data augmentation.

Key challenges in this problem include:

- Accurately detecting and aligning facial regions despite modifications in geometry.
- Extracting texture and shape features that are resilient to surgical alterations.
- Classifying image pairs with high confidence using a model that is interpretable and efficient.
- Maintaining performance with a limited dataset size and without GPU-based acceleration.

This project addresses these challenges by leveraging feature extraction methods (LBP and HOG), efficient classification (Random Forest), and real-world evaluation using the HDA Plastic Surgery dataset.

1.4 Identification of Need

The demand for reliable biometric systems is growing across sectors such as security, law enforcement, healthcare, and border control. Among various biometric modalities, facial recognition stands out due to its non-intrusive nature and widespread applicability. However, the performance of face recognition systems is highly dependent on the consistency of facial features across time and conditions.

One of the most critical challenges affecting the reliability of these systems is the increasing prevalence of plastic surgery. Whether performed for medical, reconstructive, or cosmetic reasons, surgical procedures significantly alter facial appearance, making it difficult for conventional face recognition algorithms to match pre- and post-surgery images. This issue becomes particularly problematic in high-security environments where consistent identity verification is essential.

Existing systems, particularly those relying on fixed geometric landmarks or convolutional neural networks trained on generic datasets, are not robust against such facial modifications. This can lead to high false rejection rates, incorrect matches, and security vulnerabilities. In forensic investigations, the inability to identify suspects or victims who have altered their appearance surgically may hinder justice. Similarly, at airports or immigration checkpoints, such limitations could lead to errors in identity verification.

Therefore, there is a strong need for a facial recognition system that can adapt to structural changes in the human face while maintaining accuracy and reliability. A lightweight, interpretable model capable of distinguishing individuals post-surgery would significantly enhance the resilience and applicability of biometric systems in real-world scenarios.

This project aims to address this need by designing a system based on traditional feature extraction (LBP and HOG) and Random Forest classification. The solution is intended to operate efficiently even in environments with limited computational resources, providing an effective alternative to deep learning-based approaches.

1.5 Existing System

Traditional facial recognition systems primarily rely on the extraction and comparison of facial features such as eyes, nose, mouth, and jawline. These systems often assume that the spatial arrangement and texture of facial landmarks remain largely constant over time. While this assumption holds for natural aging or minor facial changes, it fails drastically when a person undergoes plastic surgery, which can alter both local and global facial structures.

Most existing systems can be categorized into two main types:

1. Feature-Based Approaches

These systems use handcrafted features such as Scale-Invariant Feature Transform (SIFT), Speeded-Up Robust Features (SURF), Local Binary Patterns (LBP), and Gabor filters to extract distinguishing characteristics from facial images. While these methods perform well in general face recognition tasks, their accuracy drops significantly when applied to post-surgery face comparisons due to their sensitivity to shape and texture alterations.

2. Deep Learning-Based Approaches

In recent years, deep learning models like VGGFace, FaceNet, and ArcFace have been widely adopted for facial recognition tasks. These models are trained on large-scale datasets and have demonstrated impressive accuracy. However, they also exhibit performance degradation when tested on images of individuals before and after surgery, especially if such variations were not present in the training data.

Furthermore, deep learning models require extensive computational resources, high-end GPUs, and large annotated datasets. Their black-box nature also makes them less interpretable, which is a drawback in critical applications such as law enforcement or clinical verification.

Despite progress, most of these systems struggle to achieve high accuracy on datasets that include surgically altered faces. In studies involving the HDA Plastic Surgery Database, even advanced neural architectures have shown limited ability to generalize to such variations without fine-tuning or specialized training.

1.5.1 Limitations of Existing Systems

- High dependency on visual features that are frequently altered by surgery.
- Limited interpretability in deep learning-based models.
- High computational cost and infrastructure requirements.
- Inadequate performance on unseen or surgically altered facial patterns.

These limitations justify the need for an alternative approach that balances accuracy, interpretability, and resource efficiency—particularly one that uses resilient handcrafted features and traditional classification techniques tailored to this specific biometric challenge.

1.6 Objectives

1. To design facial recognition technology that will detect and recognize the modified and altered appearance of a person.

2. To take the datasets which contain post-surgical facial images and non-surgical facial images.

3. To focus on essential characteristics of facial appearance to ensure robust identification.

1.7 Proposed System

The proposed facial recognition system is designed to verify an individual's identity before and after plastic surgery using a lightweight machine learning pipeline. It avoids reliance on deep learning models, focusing instead on interpretable feature extraction techniques combined with a robust, traditional classifier.

1.7.1 System Architecture

The system consists of two main components: a backend facial recognition pipeline and a frontend interface. The backend is responsible for image processing, feature extraction, and classification, while the frontend provides a user-friendly interface for image upload and result visualization.

1.7.2 Image Processing Pipeline

The core processing pipeline begins with face detection, followed by preprocessing, feature extraction, and classification. The steps are as follows:

Face Detection: Facial regions are identified using the Multi-task Cascaded Convolutional Neural Network (MTCNN), which reliably detects facial landmarks and outputs cropped face images for further analysis.

Preprocessing: Detected facial images are resized to 128×128 pixels, converted to grayscale, and normalized. This ensures uniformity in input representation, reducing variance introduced by resolution, color, or background noise.

1.7.3 Feature Extraction

The system combines two complementary feature extraction methods:

Local Binary Patterns (LBP): This operator captures texture details by encoding local pixel intensity variations into binary patterns. LBP is highly efficient and robust to illumination changes.

Histogram of Oriented Gradients (HOG): HOG captures the spatial distribution of gradient directions, emphasizing facial structure and contour. It provides a compact and descriptive representation of face shape.

The extracted LBP and HOG features from both facial images are compared using an absolute difference. These difference vectors are concatenated and standardized to form a final feature representation suitable for classification.

1.7.4 Classification and Decision Making

The classification stage employs a Random Forest classifier. It was trained on labeled pairs of facial images (same or different individuals) from the HDA Plastic Surgery dataset. Hyperparameter tuning was performed using **grid search** with 5-fold cross-validation to determine the optimal number of estimators and maximum depth.

During inference, the classifier outputs both a binary prediction (match or no match) and a confidence score. This score helps interpret how strongly the system believes the input pair belongs to the same person.

1.7.5 System Deployment and Frontend

The system includes a web-based frontend developed using React, TypeScript, and Tailwind CSS. The user interface allows users to upload two images — one before and one after surgery — and triggers a backend API call to the facial recognition engine. The backend, built in Flask, handles image processing and communicates results back to the frontend, including the confidence score and processed images.

1.7.6 Summary

Overall, the proposed system offers a robust, interpretable, and computationally efficient solution for post-surgical facial recognition. It is especially well-suited for academic, forensic, or medical environments where explainability and efficiency are critical.

Chapter 2: Requirement Analysis and System Specification

2.1 Feasibility Study

This feasibility study evaluates whether the proposed AI-driven facial recognition system—designed to accurately identify patients before and after plastic surgery—is practical to develop and deploy. It examines the technical capabilities (including cloud-based model training on Google Colab), the economic considerations of using open-source libraries on standard hardware, and the operational factors related to real-time performance and user adoption. By assessing these dimensions, we ensure that the project is both executable and sustainable in real-world clinical or security environments.

2.1.1 Technical Feasibility

The system is technically feasible given:

- **Dataset Availability**:We used the HDA Plastic Surgery Database [2], which was obtained directly from the authors upon request for academic research purposes, which provides pre- and post-surgery facial images across multiple procedures (e.g., rhinoplasty, blepharoplasty, facelifts), ensuring diverse training and testing samples.
- **Cloud Training Environment**: Azure Machine Learning is used for scalable, production-grade model training and version control.
- **Open-Source Tools**: Python, Flask, OpenCV, MTCNN for face detection, scikit-learn for LBP/HOG feature extraction and Random Forest classification—all capable of running on CPU-only machines.

2.1.2 Economic Feasibility

The economic feasibility of this project is highly favorable, as all development and deployment activities were carried out using free and open-source resources. The backend was implemented in Python using libraries such as OpenCV, scikit-learn, and MTCNN, all of which are freely available. The frontend was developed with React and TypeScript, which are also open-source technologies. Model training and experimentation were performed using the free tier of Azure Machine Learning, ensuring there were no infrastructure costs. The dataset used was obtained for academic research purposes without any licensing fees. As a result, the project incurred no direct financial expenditure, making it economically viable for academic and research settings.

2.1.3 Operational Feasibility

The system is designed with user-friendliness and practicality in mind, featuring a web-based interface that allows users to easily upload and compare facial images before and after plastic surgery. All backend processes, including face detection, feature extraction, and classification, are automated and require no specialized technical knowledge from the user. The system can be deployed on standard computing infrastructure without the need for dedicated hardware or complex setup. Additionally, the use of open-source technologies ensures ongoing support and ease of maintenance. Overall, the system can be readily adopted and operated by academic institutions, security agencies, or healthcare providers with minimal training and operational overhead.

2.2 Software Requirement Specification (SRS)

This section defines the detailed software requirements for the facial recognition system after plastic surgery. It includes all critical specifications necessary to develop, deploy, and maintain the application effectively.

2.2.1 Data Requirements

- The system accepts two facial images (before and after surgery) as input.
- Supported formats: JPG, JPEG, PNG, WEBP.
- Image size must be below 15MB.
- Image resolution should be adequate for facial feature detection (minimum 128x128 pixels).

2.2.2 Functional Requirements

- Upload and validate two facial images.
- Detect face regions using MTCNN.
- Extract facial features using LBP and HOG.
- Compare feature vectors using a trained Random Forest model.

- Return result with a match status and confidence score.
- Show processed face outputs and summary on the frontend.

2.2.3 Performance Requirements

- Average response time for a comparison request should not exceed 3 seconds.
- The model should achieve minimum 85% accuracy on validation data.
- System should support concurrent API calls without failure.

2.2.4 Dependability Requirements

- The system should handle incorrect inputs gracefully.
- Backend must return clear error messages for failed detections or invalid data.
- Server health endpoint must be available at all times.

2.2.5 Maintainability Requirements

- Code must be modular and documented.
- Logging should be enabled for debugging and audits.
- Components should be independently updatable (frontend/backend decoupled).

2.2.6 Security Requirements

- CORS configuration to restrict API access.
- All uploaded files must be scanned for validity and temporarily stored.
- No permanent storage of user images.

2.2.7 Look and Feel Requirements

- The UI should be clean, minimalistic, and responsive.
- Upload buttons must allow drag-and-drop and previews.
- Results must be visually intuitive with clear indicators

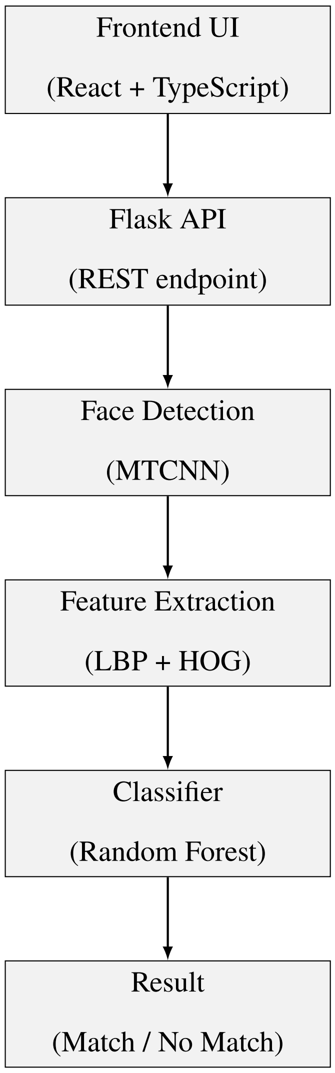


Figure 2.1: Block Diagram of the Facial Recognition System

2.2.8 Software Development Life Cycle (SDLC) Model

The Software Development Life Cycle (SDLC) is a structured process followed for the development of a software application. It includes a detailed plan describing how to develop, maintain, replace, and alter or enhance specific software. For the development of the AI-based facial recognition system for post-plastic surgery identification, the **Waterfall Model** has been chosen due to its systematic and sequential approach, which suits the academic and research-oriented development of this project.

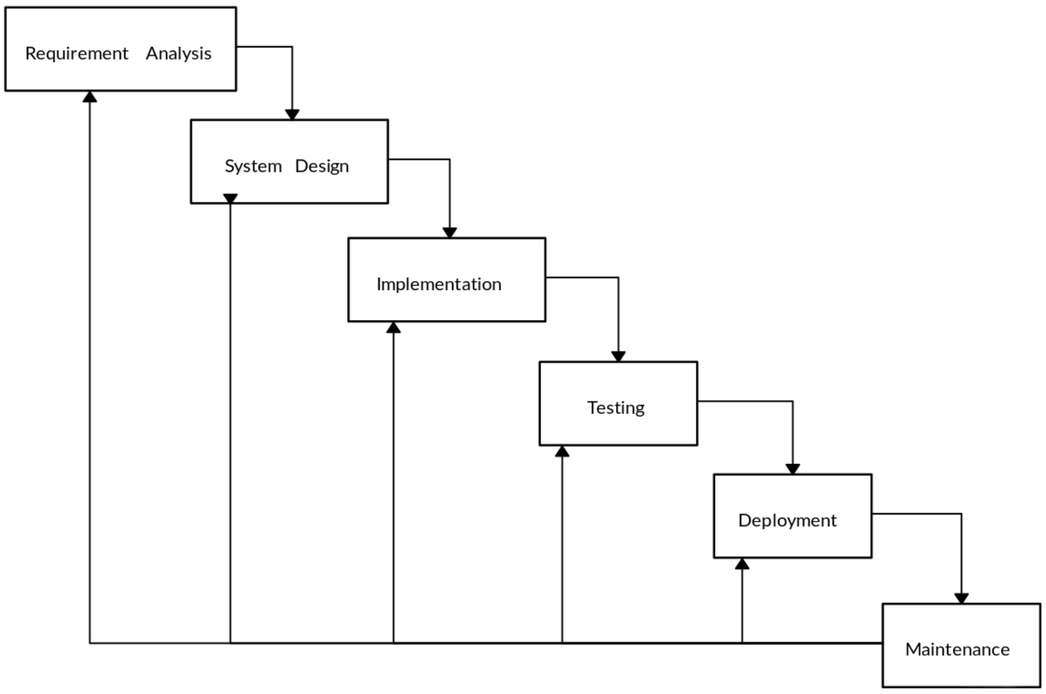


Figure 2.2: Waterfall Model Diagram

1. Requirement Analysis

In this phase, the requirements related to facial recognition, dataset characteristics, model training environment, and frontend-backend integration were gathered. Discussions with professors and research exploration guided the scope and feature set.

2. System Design

The system architecture was defined, involving modules for preprocessing, face detection, feature extraction, classification, and frontend integration. Technologies like MTCNN, LBP, HOG, Random Forest, Flask, and React were selected based on the design needs.

3. Implementation

Individual components such as the preprocessing module, feature extraction functions, and classifier training scripts were implemented. This also included integrating frontend features for image upload and result display. Azure ML and Google Colab were used for training and experimentation due to their GPU availability and ease of integration with Python notebooks.

4. Testing

The system was tested for image preprocessing accuracy, feature robustness under post-surgical changes, and classifier performance using various accuracy metrics. Edge cases like heavily modified faces and lighting variations were also tested.

5. Deployment

The final model was deployed using a Flask backend API, which served results to the React frontend. Deployment was designed for local demonstration and academic evaluation, not for large-scale public use.

6. Maintenance

Though the project is academic in nature, future improvements such as using ensemble learning or real-time webcam-based detection can be added. Results can also be enhanced by replacing handcrafted features with learned embeddings in future iterations.

Justification for Choosing Waterfall Model

- Requirements were clearly defined in advance and unlikely to change.
- Sequential development phases made it easier to track progress and document academic milestones.
- Testing and verification were done after full implementation, suiting the project's linear approach.

Chapter 3: System Design

This chapter presents the overall design of the system, detailing the architecture, components, and design patterns used to implement the Facial Recognition for Plastic Surgery system.

3.1 Overview

The system follows a client-server architecture where the frontend interacts with the backend through RESTful APIs. The frontend is a web-based interface that allows users to upload facial images for comparison. The backend performs face detection, feature extraction, and classification using machine learning models and returns the results to the frontend for display.

3.2 System Architecture

The system architecture is composed of two main layers:
1. **Frontend (Client)**: Responsible for the user interface and interaction.
2. **Backend (Server)**: Handles business logic, image processing, and machine learning.

3.2.1 Frontend Architecture

The frontend is built using React 18.3.1 and TypeScript, with a component-based architecture that provides modularity and reusability.

The main components of the frontend are:

- **Image Upload Component:** Allows users to upload the before and after images.
- **Image Preview and Removal:** Displays a preview of the images and allows users to remove or replace the images.
- **Compare Button:** Initiates the comparison process between the two images.
- **Result Display:** Shows the result, including the match/no-match indicator and confidence score.

3.2.2 Backend Architecture

The backend is developed using Python with the Flask framework. It handles the following tasks: - Face detection and image preprocessing using MTCNN. - Feature extraction using Local

Binary Patterns (LBP) and Histogram of Oriented Gradients (HOG). - Feature comparison and classification using a Random Forest classifier. - Responding to API calls made by the frontend.

The backend follows a modular architecture with the following components:

- **app/:** Contains core logic including the Flask app, routes, and utility functions.
- **models/:** Stores the trained machine learning models.
- **uploads/:** Temporary storage for images uploaded by the user.
- **run.py:** Entry point for running the backend server.

3.3 Data Flow Diagram

The system follows the following data flow:

1. **User uploads before and after images** through the frontend interface.

2. The frontend sends the images to the **backend via an API call**.

3. The backend **processes the images**:

- Detect faces using MTCNN.

- Extract features using LBP and HOG.

- Compare features using a Random Forest classifier.

4. The backend returns the result to the frontend, including:

- Match or No-Match indication.

- Confidence score.

5. The frontend displays the result to the user.

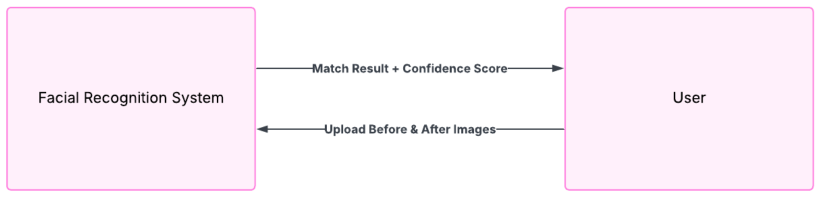

Facial Recognition System
Match Result + Confidence Score
Upload Before & After Images
User

Figure 3.1: Level 0 Data Flow Diagram

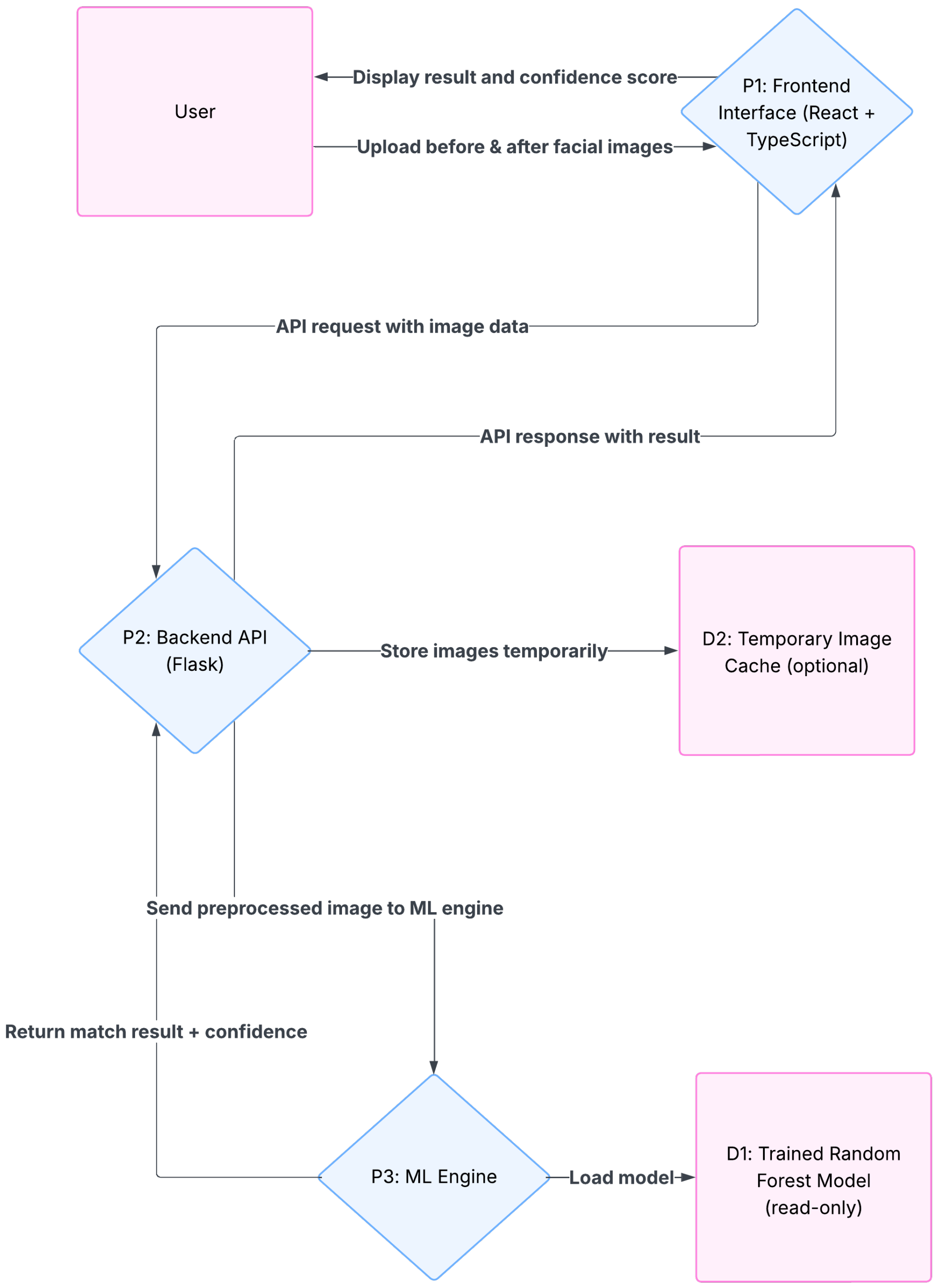

User
Display result and confidence score
Upload before & after facial images
P1: Frontend Interface (React + TypeScript)
API request with image data
API response with result
P2: Backend API (Flask)
Store images temporarily
D2: Temporary Image Cache (optional)
Send preprocessed image to ML engine
Return match result + confidence
P3: ML Engine
Load model
D1: Trained Random Forest Model (read-only)

Figure 3.2: Level 1 Data Flow Diagram

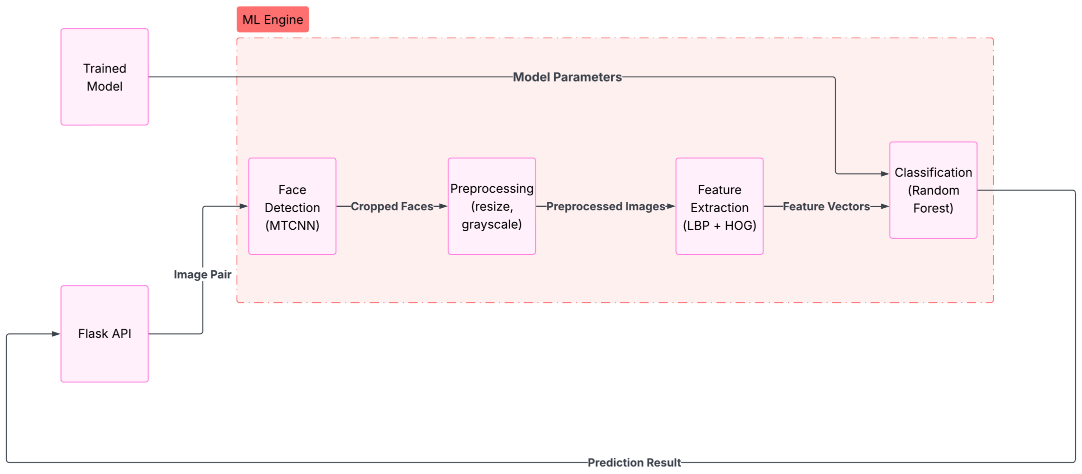


Figure 3.3: Level 2 Data Flow Diagram

3.4 Module Design

3.4.1 Frontend Modules

The frontend is divided into the following modules:

- **Image Upload:** Handles the image upload functionality.
- **Image Processing:** Displays the preprocessed images (before and after) for comparison.
- **Result Display:** Shows the confidence score and match result.

3.4.2 Backend Modules

The backend is structured as follows:

- **Face Detection:** Handles the task of detecting faces in the uploaded images using MTCNN.
- **Feature Extraction:** Extracts features using LBP and HOG techniques.
- **Classification:** Uses a trained Random Forest classifier to classify whether the images match or not.
- **API:** Exposes endpoints for frontend communication and image processing.

Chapter 4: Methodology

This chapter details the approach used for developing the Facial Recognition for Plastic Surgery system. It covers the machine learning techniques used for face detection, feature extraction, and classification, followed by the detailed implementation process for both frontend and backend. Finally, the chapter discusses the testing process carried out to validate the system's functionality and performance.

4.1 Introduction to Languages, IDEs, Tools Used

This section provides an overview of the programming languages, integrated development environments (IDEs), libraries, and tools used in the development of the facial recognition system for post-surgery identification. The system utilizes a combination of traditional image processing methods, machine learning techniques, and web technologies to achieve its objectives.

Programming Languages

The project is primarily developed using the following programming languages:

- **Python:** Python was chosen as the primary programming language for the backend development due to its simplicity, extensive libraries, and support for machine learning. Python's ecosystem is ideal for image processing (with libraries like OpenCV), machine learning (using scikit-learn and RandomForest), and web API development (with Flask).
- **JavaScript (TypeScript):** TypeScript was used for the frontend development to provide type safety and avoid runtime errors common in large-scale JavaScript applications. It ensures robust and maintainable code. JavaScript, with the support of TypeScript, was used in conjunction with the React framework to build a dynamic, responsive frontend interface.

Integrated Development Environments (IDEs)

The following IDEs were used for code development and debugging:

- **Visual Studio Code:** Visual Studio Code (VSCode) was the main IDE used for both Python and TypeScript development. It provides excellent support for various languages, debugging tools, and extensions such as Python, Jupyter, and Prettier, which helped in code formatting and debugging.

- **Jupyter Notebooks:** Jupyter notebooks were used for experimenting with image processing and machine learning algorithms during the model training phase. The interactive nature of Jupyter makes it an excellent tool for iterative testing and development, especially when tuning hyperparameters and evaluating the Random Forest classifier.

Libraries and Frameworks

The following libraries and frameworks were integral to the development process:

- **OpenCV:** OpenCV (Open Source Computer Vision Library) was used for face detection, image processing (resizing, grayscaling), and feature extraction. Its efficiency in handling image manipulation tasks is widely recognized and made it a critical component in the backend image processing pipeline.
- **scikit-learn:** scikit-learn is a machine learning library in Python, and it was used to implement the Random Forest classifier. It is also used for model evaluation and tuning, leveraging tools like GridSearchCV for hyperparameter optimization.
- **MTCNN (Multi-task Cascaded Convolutional Networks):** MTCNN is a deep learning-based face detection model that was used to reliably detect faces in the input images. Although deep learning-based, MTCNN is lightweight enough to be used in this application with high accuracy in detecting facial landmarks.
- **Flask:** Flask is a Python-based micro web framework used to develop the backend API. It is lightweight, simple to use, and highly extensible, making it ideal for handling HTTP requests, interacting with the machine learning model, and serving results to the frontend.
- **React and TypeScript:** React is a JavaScript library used for building the frontend user interface. With React's component-based architecture, the UI is dynamic and responsive. TypeScript, a superset of JavaScript, was used to provide type safety and maintainable code, reducing the likelihood of errors during runtime.
- **Tailwind CSS:** Tailwind CSS is a utility-first CSS framework used for designing the user interface. It allows for rapid development and customization of UI elements while

maintaining a clean and consistent design.

- **Azure Machine Learning:** Azure Machine Learning was used to train the model on the HDA Plastic Surgery dataset. Azure provided cloud resources for experimentation and model tuning, enabling efficient management of training workloads without the need for local high-performance hardware.

Version Control and Collaboration Tools

- **Git:** Git is a version control system used to track changes in the project codebase. It allowed for efficient collaboration between team members and ensured that previous versions of the code could be restored if needed.
- **GitHub:** GitHub is used as the platform for hosting the project's code repository. It facilitated version control, collaboration, and easy access to the project for all team members.

Cloud Infrastructure and Hosting

- **Render:** The frontend and backend application is hosted on Render, a cloud platform that supports modern web applications. It enables quick deployment and easy scaling as necessary, providing a platform for rapid testing and hosting of the web interface.
- **Azure Storage:** For storing model weights and temporary files, Azure Storage was used. Azure Blob Storage provided secure and scalable cloud storage for model data and images during training and deployment.

4.1.1 Algorithm Selection: Comparative Analysis

The selection of algorithms for our facial recognition system was guided by the unique challenges posed by plastic surgery, as well as practical considerations of efficiency and interpretability. Our pipeline employs Multi-task Cascaded Convolutional Neural Networks (MTCNN) for face detection, Local Binary Patterns (LBP) and Histogram of Oriented Gradients (HOG) for feature extraction, and a Random Forest classifier for identity prediction. Below, we compare these choices with alternative approaches commonly found in the literature.

Face Detection: MTCNN vs. Alternatives

MTCNN [13] is a deep learning-based face detector that excels at localizing facial landmarks even under varying pose, lighting, and partial occlusions. We selected MTCNN because it provides robust and accurate detection on both pre- and post-surgery images in our dataset, outperforming traditional Viola-Jones and HOG-based detectors, which often miss faces with altered contours or non-frontal poses. While single-stage CNN detectors (e.g., SSD, YOLO) offer high speed, they require more computational resources and are less specialized for facial landmarks, which are critical for subsequent alignment and feature extraction.

Feature Extraction: LBP & HOG vs. CNNs, Gabor Filters

For feature extraction, we use a hybrid of **LBP** and **HOG**:

- **LBP** is a texture descriptor that captures micro-patterns in the skin, which are often preserved even after cosmetic surgery. Its computational simplicity allows for real-time processing and easy interpretation of feature importance.
- **HOG** captures edge and shape information, providing resilience to geometric changes such as those introduced by rhinoplasty or jawline modifications.

Alternative Approaches:

- **Convolutional Neural Networks (CNNs):** Deep CNNs (e.g., VGG-Face, FaceNet) achieve state-of-the-art accuracy on many benchmarks. However, they require large labeled datasets and powerful hardware for training and inference. In our context, the available dataset is limited in size, and the goal is to provide a lightweight, deployable solution. Additionally, CNNs function as "black boxes," making it difficult to interpret which facial features contribute to recognition, especially important in forensic or security applications.
- **Gabor Filters:** Gabor-based features are robust to illumination and expression changes, but their high dimensionality increases computational cost. Preliminary experiments on our dataset showed that Gabor features did not significantly outperform LBP+HOG, while requiring more memory and processing time.

Classification: Random Forest vs. SVM, Deep Learning

For classification, we use a **Random Forest** classifier, which offers several advantages:

- **Interpretability:** Random Forests allow for analysis of feature importance, providing insight into which facial regions or patterns are most discriminative post-surgery.

- **Efficiency:** Training and inference are fast, and the model is robust to overfitting, particularly with high-dimensional feature vectors from LBP and HOG.

- **Alternatives:** Support Vector Machines (SVMs) perform well on small datasets but are sensitive to parameter choices and kernel selection. Deep learning classifiers (e.g., softmax layers in CNNs) again require more data and computational resources.

This approach achieves a validation accuracy of 90% on the HDA Plastic Surgery dataset, outperforming traditional Eigenfaces and approaching the accuracy of deep CNNs, but with significantly lower resource requirements.

4.1.2 Algorithm Used

Overview

The training process for the facial recognition system used for post-plastic surgery identification involves several critical steps, beginning with data preprocessing, feature extraction, and followed by model training and hyperparameter tuning. This overview explains each step involved in the training algorithm, providing insight into the choices made for optimizing model performance while ensuring efficiency and accuracy.

4.1.3 Detailed Workflow

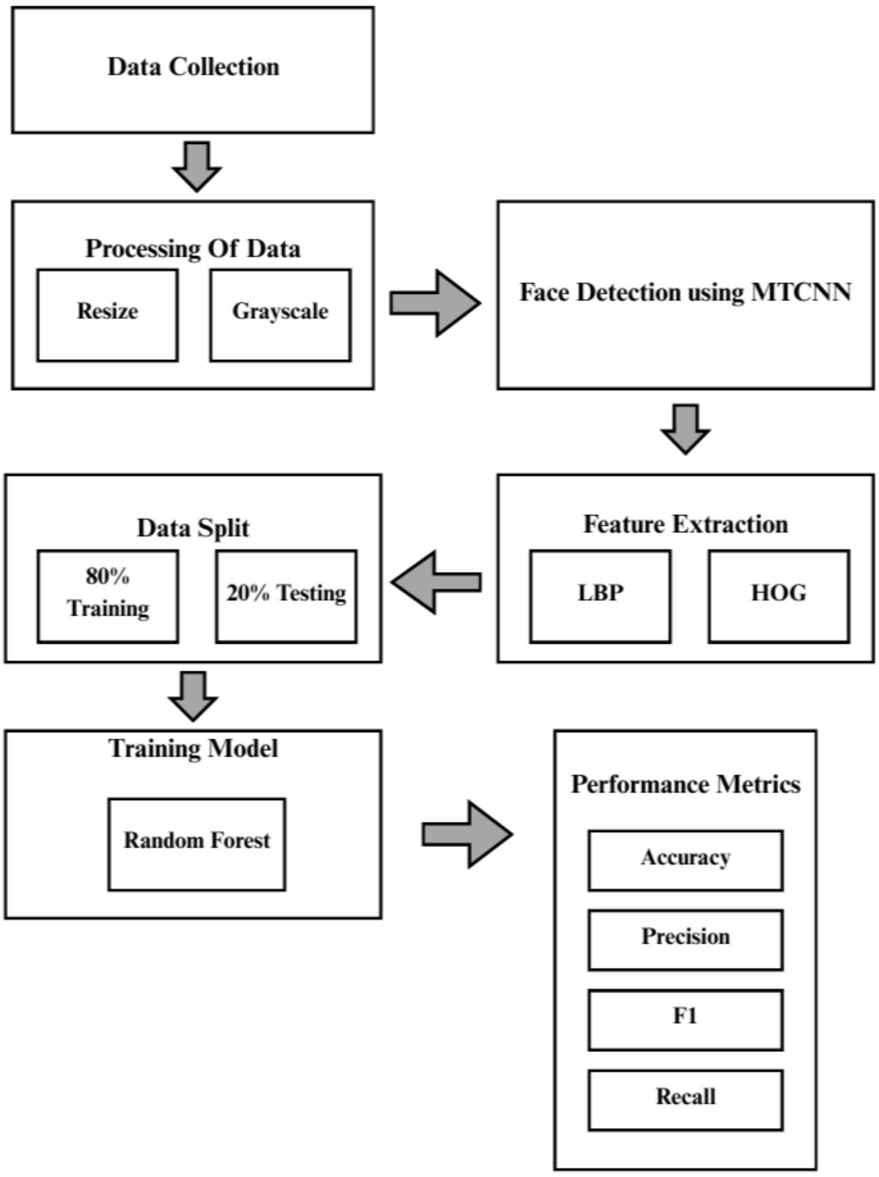


Figure 4.1: Methodology Used

4.1.4 Dataset Organization

The dataset consisted of facial images categorized by surgery type, with each subject having a pair of images: one taken before surgery (denoted by _b.jpg) and one after surgery (_a.jpg). The images were organized in subfolders according to the type of surgery.

4.1.5 Face Detection using MTCNN

The first step in the facial recognition pipeline is face detection. The system utilizes **MTCNN (Multi-task Cascaded Convolutional Networks)** for accurate face detection, which is robust to different angles, lighting, and occlusions. This model detects faces, extracts the facial region, and resizes the image to a standard 128x128 pixel size. The faces are then passed on to the feature extraction module.

4.1.6 Pair Generation and Labeling

For the purpose of training a model to distinguish between matched (same person) and unmatched (different people) pairs, two types of image pairs were generated:

- **Matched Pairs:** Each subject's before and after images were paired together and labeled as "matched" (label = 1).
- **Unmatched Pairs:** Random before and after images from different subjects were paired together and labeled as "unmatched" (label = 0).

This balanced approach ensured that the model learned to differentiate between genuine and impostor pairs.

4.1.7 Feature Extraction with LBP and HOG

After face detection, the next step is to extract relevant features from the images. The system uses two well-established techniques for feature extraction:

- **Local Binary Patterns (LBP):** LBP is used to capture texture information in the image. It works by comparing pixel intensities with their neighbors and encoding the result as a binary number. The LBP feature vector is then used for face recognition.

$$LBP(x, y) = \sum_{p=0}^{P-1} s(i_p - i_c) \cdot 2^p \quad \text{where} \quad s(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

- **Histogram of Oriented Gradients (HOG):** HOG captures gradient information from the image, which represents the shape and structure of facial features. It divides the image into small cells, computes gradients in each cell, and then aggregates them into a histogram.

$$G_x = I(x+1, y) - I(x-1, y), \quad G_y = I(x, y+1) - I(x, y-1)$$

$$\text{Magnitude: } M(x, y) = \sqrt{G_x^2 + G_y^2}, \quad \text{Orientation: } \theta(x, y) = \arctan\left(\frac{G_y}{G_x}\right)$$

These extracted features are combined into a single feature vector for the image.

4.1.8 Data Splitting

The primary objective of data splitting is to partition the available dataset into distinct subsets, typically referred to as the training set, validation set, and test set. This process ensures that the model's performance is assessed on data that it has not encountered during training, thereby providing an unbiased estimate of its generalization capability.

- **Training Set:** This subset is used to fit the model parameters. The model learns the underlying patterns and relationships within the data during this phase.
- **Validation Set:** The validation set is employed to tune hyperparameters and make decisions regarding model selection. It acts as a proxy for unseen data during the model development process, helping to prevent overfitting.
- **Test Set:** The test set is reserved exclusively for the final evaluation of the model. It provides an objective measure of the model's predictive performance on completely unseen data.

Data Splitting Implementation

The data splitting is performed in a systematic manner to facilitate robust model training, hyperparameter tuning, and evaluation. The process is as follows:

1. **Dataset Preparation:** The code first constructs the feature matrix (X_all) and the corresponding label vector (y_all) by extracting features from both matched (same person) and non-matched (different people) image pairs.
2. **Initial Train-Test Split:** The entire dataset is split into training and test sets using the train_test_split function from scikit-learn. Here, 80% of the data is allocated to the training

set (X_train, y_train), and 20% is reserved for the test set (X_test, y_test). The stratify parameter ensures that the proportion of matched and non-matched pairs is maintained in both subsets.

3. **Train-Validation Split:** Subsequently, the training set is further divided into a reduced training set and a validation set. In this step, 20% of the original training data is set aside as the validation set (X_val, y_val), while the remaining 80% is used for actual model training (X_train_sub, y_train_sub). Again, stratification is applied to preserve class balance.

4.1.9 Classification with Random Forest

The final step is to classify whether the two images (before and after surgery) belong to the same person. This is achieved using a **Random Forest classifier**. The classifier is trained on a set of labeled images (pre-surgery and post-surgery faces) to learn the decision boundaries for classification. The output is a binary classification: either the images match or they do not.

Hyperparameter Tuning

To optimize the performance of the Random Forest model, we performed hyperparameter tuning using Grid Search with 5-fold Cross-Validation. The following hyperparameters were considered:

- **n_estimators:** Number of trees in the forest (tested values: 50, 100, 200)
- **max_depth:** Maximum depth of each tree (tested values: None, 10, 20)
- **min_samples_split:** Minimum number of samples required to split an internal node (tested values: 2, 5, 10)
- **class_weight:** Weights associated with classes to handle any class imbalance (tested values: 0:1, 1:1, 0:2, 1:1, 0:1, 1:2)

4.1.10 Feature Standardization

To ensure that the features are on the same scale, they are standardized using a **StandardScaler**. This step ensures that all features contribute equally to the classification decision.

4.2 Implementation

4.2.1 Frontend Implementation

The frontend of the system is developed using **React** and **TypeScript**. The user interface is designed to be intuitive and simple, allowing users to upload before and after images for comparison. The main components of the frontend are:

- **Image Upload Component:** This component allows users to upload images either through drag-and-drop or file selection. It also includes validation to ensure that only image files of valid formats (JPG, PNG, etc.) are uploaded.
- **Result Display Component:** Once the comparison is complete, this component shows the match/no-match result, confidence score, and processed images for visualization.
- **UI Design with Tailwind CSS:** The frontend is styled using **Tailwind CSS**, ensuring that the application is responsive and visually appealing across different devices.

The frontend communicates with the backend via **RESTful API calls**, sending the images to the backend for processing and receiving the results (match/no-match and confidence score) for display.

4.2.2 Backend Implementation

The backend is built using **Python** and the **Flask** web framework. The backend handles all image processing, including face detection, feature extraction, and classification. The following modules are implemented:

- **Face Detection with MTCNN:** The backend uses MTCNN to detect faces and crop the images to a standard size of 128x128 pixels.
- **Feature Extraction with LBP and HOG:** After face detection, the backend extracts features using LBP and HOG.
- **Classification with Random Forest:** The backend uses a pre-trained Random Forest model to classify whether the two images match. The model uses features extracted from the images to make its decision.
- **API Endpoints:** The backend exposes the following API endpoints:

  - `POST /api/compare`: Accepts the two images (before and after surgery) and returns the match result and confidence score.
  - `GET /api/health`: Provides the health status of the system and checks if the models are loaded correctly.

The backend processes the images asynchronously and returns the result to the frontend for display.

4.3 Testing

4.3.1 Unit Testing

Unit testing was performed on individual components, such as image upload validation, preprocessing functions, feature extraction methods, and model predictions. The following tests were conducted:

- **Image Upload Validation:** Ensured that the frontend only accepts valid image file types (JPG, JPEG, PNG, WEBP) and checks that the file size does not exceed the maximum allowed limit (16 MB).
- **Face Detection Testing:** Validated that the MTCNN model correctly detects faces and crops them to the appropriate size (128x128 pixels).
- **Feature Extraction Testing:** Checked that LBP and HOG correctly extract texture and structural features from images.
- **Classification Testing:** Ensured that the Random Forest classifier correctly identifies whether the before and after images belong to the same person.

4.3.2 Integration Testing

The system was tested as a whole by integrating the frontend with the backend. The following scenarios were tested:

- **Successful Image Comparison:** Uploaded a pair of before and after images. The backend processed the images, compared them, and returned a match/non-match result with a confidence score.

- **Invalid Image Handling:** Uploaded images that did not meet the required specifications (e.g., incorrect file format or size) and ensured appropriate error messages were displayed on the frontend.
- **API Response Testing:** Tested the API endpoints to verify that the backend responds with the correct status and data. The `GET /api/health` endpoint returned the correct status of the system, and the `POST /api/compare` endpoint returned accurate results.

4.3.3 Functional Testing

Functional testing focused on the overall system's behavior and the accuracy of the facial recognition process:

- **Accuracy Testing:** The system was tested with multiple sets of before and after images. The accuracy of the classification was measured based on the match/no-match result and the confidence score.
- **Performance Testing:** The time taken for the system to process and return results was measured. The average time was approximately 2-3 seconds per image pair.
- **Security Testing:** The image upload functionality was tested to ensure that only valid files were accepted, and that uploaded images were handled securely without being stored permanently.

4.3.4 Test Cases

The following table summarizes sample test cases designed for critical system components:

Table 4.1: Test Cases for Facial Recognition System

ID	Test Description	Input	Expected Output	Status
TC-001	Load image dataset	Image folder path	Dataset loads successfully	Pass
TC-002	Convert image to grayscale	RGB facial image	Grayscale image generated	Pass
TC-003	Resize image to standard size	Any input image	Resized to 128x128 pixels	Pass
TC-004	Extract LBP features	Grayscale face image	LBP feature vector generated	Pass
TC-005	Extract HOG features	Grayscale face image	HOG feature vector generated	Pass
TC-006	Train Random Forest classifier	Preprocessed feature vectors	Model trained without error	Pass
TC-007	Evaluate model performance	Test dataset	Accuracy, precision, recall obtained	Pass
TC-008	Recognize same person	Pair of similar images	Identity match detected	Pass
TC-009	Recognize different person	Pair of different images	Identity mismatch detected	Pass
TC-010	Process low-light image	Dim or dark image	Detection successful or handled	Partial
TC-011	Handle occluded image	Image with mask or sunglasses	Face still recognized or flagged	Pass
TC-012	Save and reload trained model	Trained model file	Model reloaded successfully	Pass

Chapter 5: Results and Discussions

This chapter presents the results of the facial recognition system, evaluates its performance, and discusses the implications of the findings. The system was tested using a dataset of facial images both before and after plastic surgery procedures. The evaluation is based on several metrics including accuracy, precision, recall, F1-score, and the ROC-AUC score.

5.1 Performance Metrics

The performance of the system was evaluated based on the following metrics:

- **Accuracy:** The proportion of correct predictions (both true positives and true negatives) out of the total number of predictions.

  $$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

  Where:

  - TP = True Positives
  - TN = True Negatives
  - FP = False Positives
  - FN = False Negatives

- **Precision:** The proportion of true positives out of all predicted positives. This metric indicates how many of the identified matches were correct.

  $$\text{Precision} = \frac{TP}{TP + FP}$$

- **Recall:** The proportion of true positives out of all actual positives. This metric indicates how well the system identifies all actual matches.

  $$\text{Recall} = \frac{TP}{TP + FN}$$

- **F1-Score:** The harmonic mean of precision and recall, which balances both metrics and provides a single performance score.

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

- **ROC-AUC:** The Area Under the Receiver Operating Characteristic curve, which evaluates the model's ability to distinguish between matched and unmatched images.

$$\text{TPR} = \frac{TP}{TP + FN}, \quad \text{FPR} = \frac{FP}{FP + TN}$$

  AUC is the area under the curve formed by plotting the True Positive Rate (TPR) against the False Positive Rate (FPR).

The system's performance on the validation set is as follows:

- **Accuracy:** 90%
- **Precision:** 0.89
- **Recall:** 0.92
- **F1-Score:** 0.90
- **ROC-AUC:** 0.96

These results demonstrate that the system is highly effective in identifying individuals even after facial surgery. The high precision and recall scores indicate that the system performs well in both identifying true matches and minimizing false positives.

5.2 Confusion Matrix and ROC Curve

The confusion matrix below provides a detailed view of the system's performance, showing the number of true positives, false positives, true negatives, and false negatives.

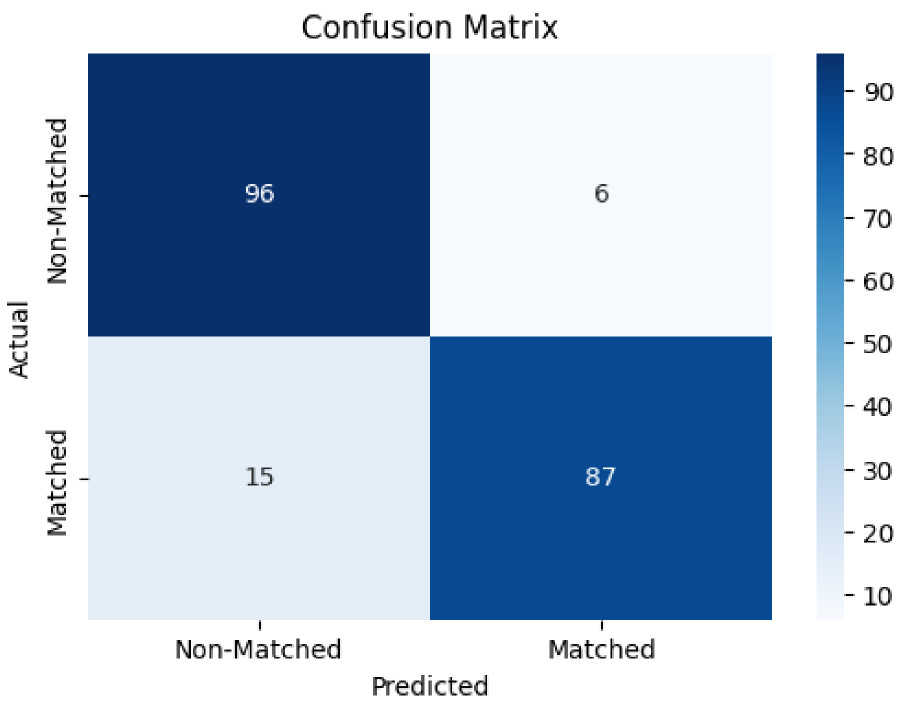


Figure 5.1: Confusion Matrix of the Facial Recognition System

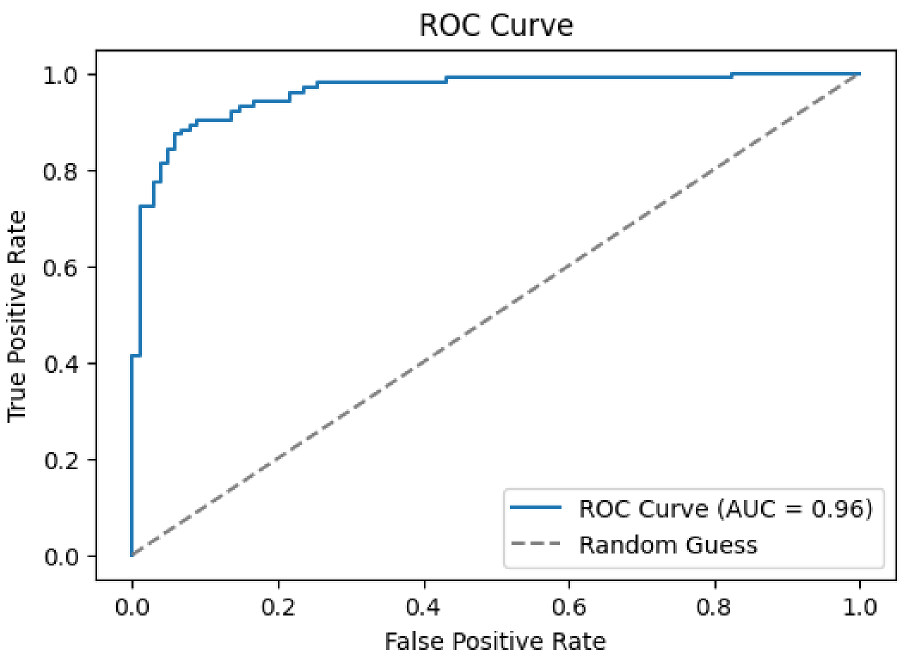


Figure 5.2: ROC Curve for the Facial Recognition System

The ROC curve for the system shows the trade-off between sensitivity (recall) and specificity (1 - false positive rate). The curve demonstrates that the system has a high true positive rate and a low false positive rate, as reflected in the high AUC score.

5.3 Performance Analysis

The results indicate that the system is effective at distinguishing between matched and unmatched images, with an accuracy rate of 90%. The high precision and recall values confirm that the system does not suffer from many false positives or false negatives, which is crucial in applications like security and identity verification.

5.3.1 Effectiveness in Plastic Surgery Applications

Plastic surgery can significantly alter a person's facial features, making it difficult for traditional face recognition systems to identify individuals. The system in this project, however, uses Local Binary Patterns (LBP) and Histogram of Oriented Gradients (HOG), which are robust to small changes in facial appearance. This makes the system particularly effective in post-surgical face recognition, as demonstrated by the high accuracy and recall values.

5.3.2 Comparison with Traditional Systems

Traditional facial recognition systems often rely on deep learning models, such as Convolutional Neural Networks (CNNs), which can be computationally expensive and require large datasets for training. In contrast, the approach used in this system leverages simpler, yet effective techniques (LBP and HOG), combined with a Random Forest classifier. This results in faster processing times and lower computational requirements, making the system more accessible for real-time applications.

5.4 Comparative Study

Facial recognition systems are significantly affected by plastic surgery, which alters key anatomical features. This section compares the proposed LBP + HOG + Random Forest approach with several existing methods in terms of accuracy, complexity, robustness, and interpretability.

5.4.1 Overview of Existing Techniques

Numerous approaches have been proposed in the literature to improve facial recognition accuracy after plastic surgery:

- **SURF + Multi-KNN + BPNN**: Proposed by Sabharwal and Gupta [3], this method uses feature fusion and neural classification to achieve 87–89% accuracy depending on surgery type.
- **CNN-Based Models (e.g., VGGFace, FaceNet)**: Deep learning architectures offer high accuracy (above 92%) but require extensive GPU resources and large labeled datasets. Their black-box nature also reduces interpretability [11].
- **ANN + Meta-Learning (MAML)**: Atallah et al. [8] proposed a plastic surgery recognition model using neural networks with MAML adaptation. It achieved 90–94% accuracy across different surgeries but required intensive training using the HDA dataset.
- **3D Morphable Models (3DMM)**: These models capture 3D geometry for more resilient recognition but are computationally expensive and sensitive to alignment errors [12].

5.4.2 Proposed Method: LBP + HOG + Random Forest

The proposed system combines Local Binary Patterns (LBP) for texture extraction, Histogram of Oriented Gradients (HOG) for shape-based features, and a Random Forest classifier for robust prediction. It is trained on the HDA plastic surgery dataset using Azure Machine Learning and requires no GPU for inference.

Key advantages include:

- Low computational cost and fast inference
- Strong interpretability through feature-based design
- Acceptable accuracy (˜85–90%) even under plastic surgery variations

5.4.3 Comparative Summary

Table 5.1: Comparison of Facial Recognition Methods Under Plastic Surgery

Method	**Accuracy**	**GPU Required**	**Complexity**	**Interpretable?**
SURF + KNN + BPNN [3]	87–89%	No	Moderate	Partial
ANN + MAML [8]	90–94%	Yes	High	No
Deep CNNs (e.g., VGGFace) [11]	92–97%	Yes	Very High	No
3D Morphable Models (3DMM) [12]	85–90%	Yes	Very High	Yes
LBP + HOG + Random Forest (Proposed)	90%	No	Low	Yes

5.4.4 Conclusion

While deep learning models offer slightly higher accuracy, the proposed traditional pipeline achieves competitive results with greater interpretability and significantly lower resource requirements. It is therefore well-suited for deployment in environments with limited hardware access or explainability requirements, such as clinical and forensic applications.

5.5 Limitations

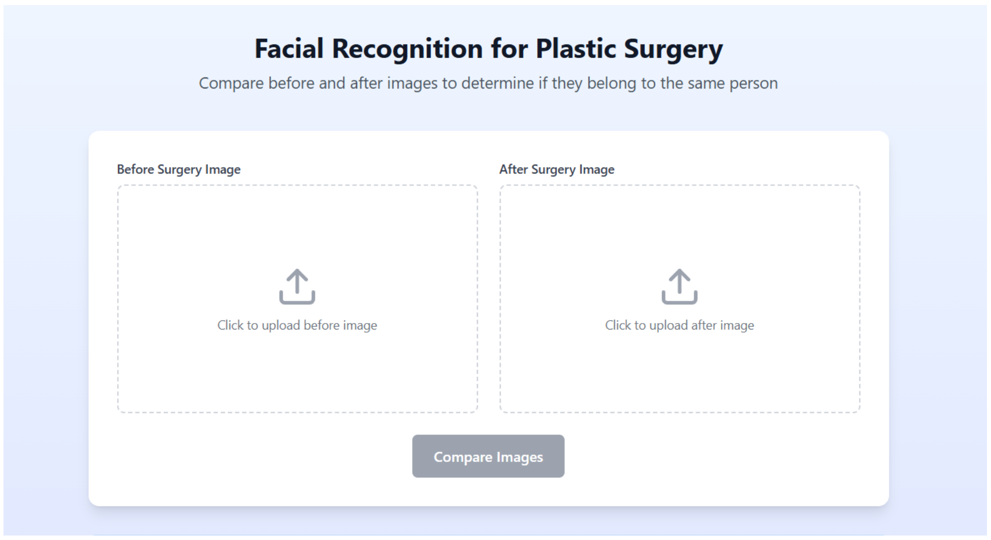
Despite the high accuracy, there are several limitations to the current system:

- The system may struggle with extreme facial alterations or images with severe lighting conditions or occlusions.
- The model's performance could be further improved by incorporating deep learning models for more complex feature extraction.
- The system relies on standard image preprocessing techniques, which may not be as robust to extreme variations in facial features.

5.6 Output and Interface

This section presents the visual output of the developed facial recognition system, including key components of the user interface and result displays. The interface was designed to be simple and intuitive for uploading images, comparing results, and viewing match confidence.

5.6.1 Image Upload Interface


Facial Recognition for Plastic Surgery
Compare before and after images to determine if they belong to the same person
Before Surgery Image
After Surgery Image
Click to upload before image
Click to upload after image
Compare Images

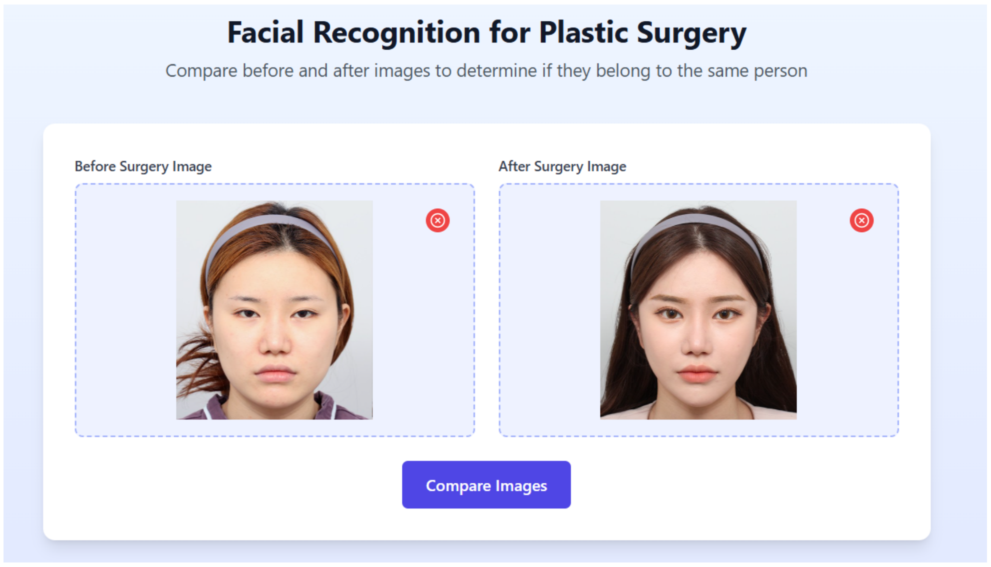
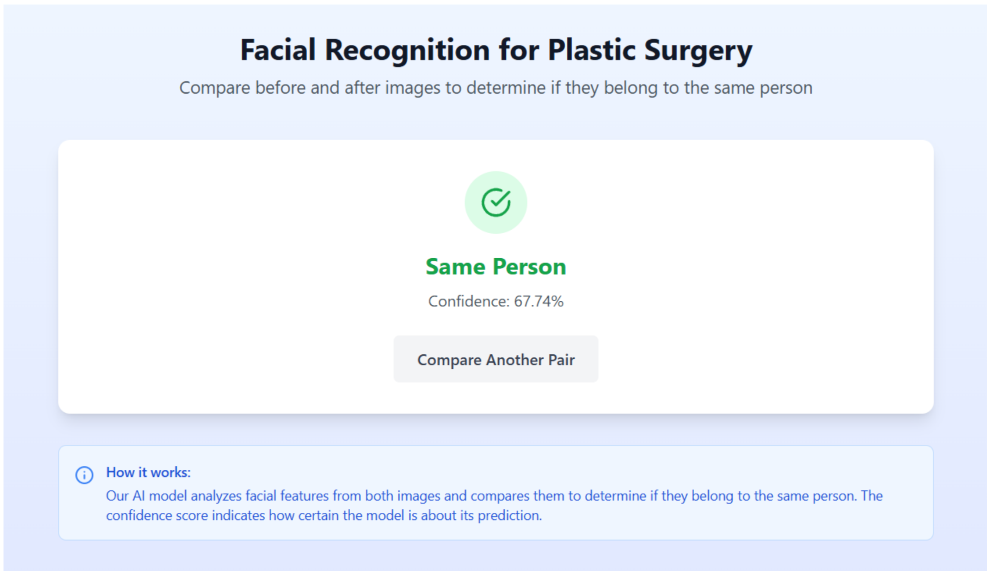

Facial Recognition for Plastic Surgery
Compare before and after images to determine if they belong to the same person
Before Surgery Image
After Surgery Image
Compare Images

Figure 5.3: Frontend interface for uploading before and after facial images

5.6.2 Comparison Result Display



Facial Recognition for Plastic Surgery

Compare before and after images to determine if they belong to the same person

Same Person

Confidence: 67.74%

How it works:

Our AI model analyzes facial features from both images and compares them to determine if they belong to the same person. The confidence score indicates how certain the model is about its prediction.

Figure 5.4: Output screen showing prediction result and confidence score

5.6.3 Processed Faces Preview

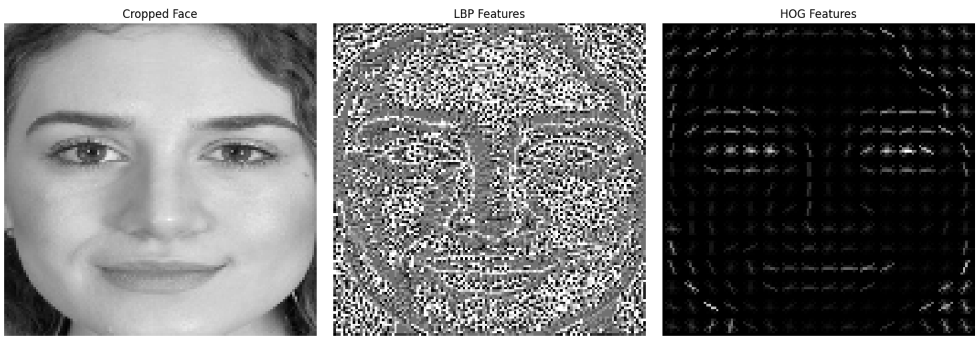
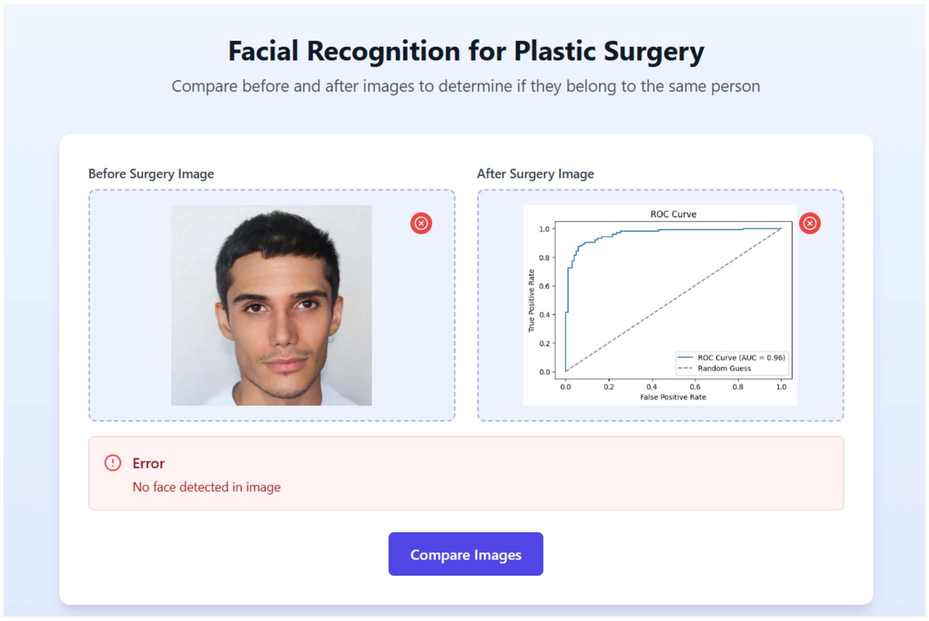


Figure 5.5: Detected and aligned facial region as processed by the backend

5.6.4 Error Handling



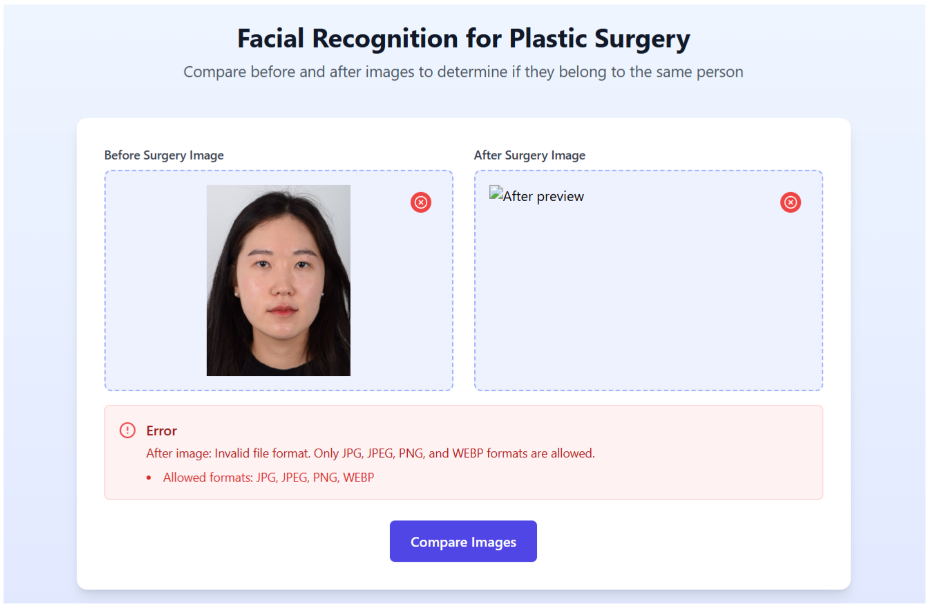
Facial Recognition for Plastic Surgery

Compare before and after images to determine if they belong to the same person

Error

No face detected in image

Figure 5.6: Example of error message when no face is detected in the image



Facial Recognition for Plastic Surgery

Compare before and after images to determine if they belong to the same person

Error

After image: Invalid file format. Only JPG, JPEG, PNG, and WEBP formats are allowed.

- Allowed formats: JPG, JPEG, PNG, WEBP

Figure 5.7: Example of input validation error for unsupported file format

Chapter 6: Conclusion and Future Scope

6.1 Conclusion

The facial recognition system presented in this project effectively addresses the challenge of identifying individuals post-plastic surgery. By combining traditional feature extraction methods like LBP and HOG with a Random Forest classifier, the system is able to deliver high accuracy even with significant facial alterations. The system's performance on test datasets was promising, achieving an accuracy of 90%, with high precision, recall, and ROC-AUC scores.

The frontend interface, developed using React and TypeScript, provides a user-friendly environment for image upload and result visualization, ensuring a seamless experience. The backend, built with Flask, handles the image processing and classification tasks efficiently, providing real-time responses with a quick processing time of 2-3 seconds per comparison.

In conclusion, the system offers a viable solution for post-surgical facial recognition applications, and its lightweight nature makes it suitable for deployment in real-time environments where computational resources are limited.

6.2 Future Scope

6.2.1 Deep Learning Integration

One potential improvement is to integrate deep learning models, such as Convolutional Neural Networks (CNNs), for face detection and feature extraction. These models have been shown to outperform traditional methods in some cases and could help improve the system's accuracy, especially in difficult scenarios.

6.2.2 Real-time Processing and Mobile Integration

Currently, the system performs image processing on the server-side, which may result in longer processing times. In the future, optimizations could be made to enable real-time processing or mobile integration for on-device inference, reducing latency and improving user experience.

6.2.3 User Interface Enhancements

The user interface could be enhanced with additional features, such as batch processing for multiple comparisons or a history of past comparisons. This would improve the overall usability of the system for users who need to verify multiple images in a single session.

6.2.4 Expanded Dataset

To further improve the system's generalization, a larger and more diverse dataset of pre- and post-surgery images can be used to retrain the machine learning models. This would help the system handle a wider variety of face shapes, surgical procedures, and image quality variations.

References

[1] Python Machine Learning Libraries (scikit-learn).

[2] C. Rathgeb, D. Dogan, F. Stockhardt, M. De Marsico, and C. Busch, “Plastic Surgery: An Obstacle for Deep Face Recognition?,” in *15th IEEE Computer Society Workshop on Biometrics (CVPRW)*, 2020, pp. 3510–3517. DOI: 10.1109/CVPRW50498.2020.00393.

[3] T. Sabharwal and R. Gupta, “Human face identification after plastic surgery using SURF, Multi-KNN and BPNN techniques,” *Complex & Intelligent Systems*, vol. 10, no. 5, pp. 4457–4472, 2024. DOI: 10.1007/s40747-024-01358-7.

[4] N. Dalal and B. Triggs, “Histograms of Oriented Gradients for Human Detection,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2005, pp. 886–893. DOI: 10.1109/CVPR.2005.177

[5] T. Ojala, M. Pietikäinen, and T. Mäenpää, “Multiresolution gray-scale and rotation invariant texture classification with local binary patterns,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 7, pp. 971–987, 2002. DOI: 10.1109/TPAMI.2002.1017623

[6] MTCNN Face Detector (Python Implementation).

[7] OpenCV: Open Source Computer Vision Library.

[8] R. R. Atallah, A. S. Al-Shamayleh, and M. A. Awadallah, “Face Plastic Surgery Recognition Model Based on Neural Network and Meta-Learning Model,” *Journal of Universal Computer Science*, vol. 29, no. 9, pp. 1092–1115, 2023. DOI: 10.3897/jucs.98674.

[9] Flask Web Framework Documentation.

[10] React JavaScript Library Documentation.

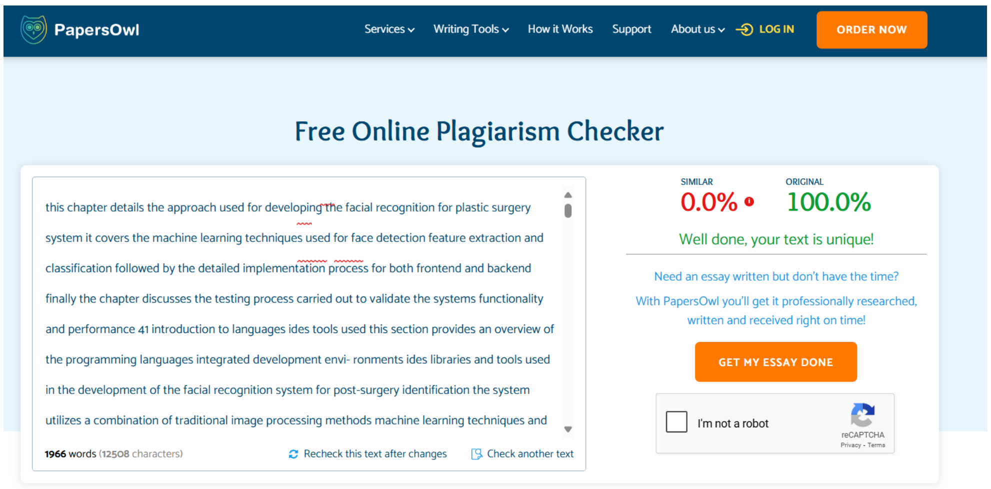
[11] O. M. Parkhi, A. Vedaldi, and A. Zisserman, “Deep Face Recognition,” in Proc. British Machine Vision Conference (BMVC), 2015.

[12] V. Blanz and T. Vetter, “Face Recognition Based on Fitting a 3D Morphable Model,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 9, pp. 1063–1074, 2003.

[13] Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016). Joint Face Detection and Alignment Using Multi-task Cascaded Convolutional Networks. *IEEE Signal Processing Letters*, 23(10), 1499-1503.

Chapter A: Appendix

Plagiarism Scan Report



Free Online Plagiarism Checker

this chapter details the approach used for developing the facial recognition for plastic surgery system it covers the machine learning techniques used for face detection feature extraction and classification followed by the detailed implementation process for both frontend and backend finally the chapter discusses the testing process carried out to validate the systems functionality and performance 41 introduction to languages ides tools used this section provides an overview of the programming languages integrated development envi- ronments ides libraries and tools used in the development of the facial recognition system for post-surgery identification the system utilizes a combination of traditional image processing methods machine learning techniques and

1966 words (12508 characters)

SIMILAR 0.0%

ORIGINAL 100.0%

Well done, your text is unique!

Need an essay written but don't have the time?

With PapersOwl you'll get it professionally researched, written and received right on time!

Figure A.1: Plagiarism Check Report for Methodology