Project report on

Fingerprint Identification

Group Members:

1. Mooga Shanmukha Teja BT23ECI036
2. Chilla Vivek Reddy BT23ECI037
3.Aman Panchal BT23ECI041
4.Bakki Sylvester Mohan Raj BT23ECI057

A report submitted for the partial fulfilment of the requirements of the course
Digital Image Processing

Submission Date: 12/11/2025

Under the guidance of:
Dr. Tapan Jain

Department of Electronics and Communication Engineering

भारतीय सूचना प्रौद्योगिकी संस्थान, नागपुर
Indian Institute of Information Technology, Nagpur

Table of contents:

Chapter No.	Particular	Page No.
1	Introduction	2
2	Objective	3
3	Methodology	4
4	Results and Observation	6
5	Code	10
6	Conclusion	13

Chapter 1: Introduction

Biometric authentication offers a secure and reliable means of verifying identity through unique physical traits. Among these, fingerprints remain the most popular due to their uniqueness, stability, and ease of capture. Each individual's fingerprint ridge pattern is distinct and remains unchanged throughout life, making it a dependable identifier.

This project presents a fingerprint recognition system using Python and OpenCV, applying Scale-Invariant Feature Transform (SIFT) for feature detection and Fast Library for Approximate Nearest Neighbors (FLANN) for matching. SIFT detects distinctive, stable keypoints in fingerprint images, while FLANN efficiently matches them to find similarities.

The system automatically compares fingerprint images from an "unknowns" folder with those in a "database" folder and determines if a match exists based on a similarity threshold. The approach is simple, efficient, and robust to rotation and scale variations, making it suitable for lightweight biometric verification and academic use.

Chapter 2 - Literature:

Fingerprint recognition has been widely researched as one of the most reliable biometric identification methods. Early systems relied on minutiae-based matching, where ridge endings and bifurcations are detected and compared. While accurate, this method requires heavy preprocessing — such as enhancement, binarization, and thinning — making it sensitive to noise, skin condition, and image quality (Jain et al., 1997).

To overcome these limitations, image-based and texture-based approaches were developed. These methods treat the fingerprint as a textured pattern rather than focusing only on minutiae. Ross and Jain (2003) showed that combining both minutiae and texture information improves robustness when fingerprints are distorted or partially captured.

With advancements in computer vision, feature-based techniques like SIFT (Scale-Invariant Feature Transform) and SURF (Speeded-Up Robust Features) became popular. SIFT, introduced by Lowe (1999), detects stable keypoints in an image that are invariant to rotation, scale, and brightness. It provides a more flexible representation of fingerprints without the need for strict alignment.

For faster comparison, FLANN (Fast Library for Approximate Nearest Neighbors) by Muja and Lowe (2009) enables quick matching of high-dimensional SIFT descriptors, reducing computational time significantly. This makes real-time fingerprint comparison feasible even with larger datasets.

More recent work has explored deep learning approaches, such as convolutional neural networks (CNNs), for fingerprint classification and matching. However, these methods require large datasets and high computing power, which makes them less suitable for small-scale or academic implementations.

Hence, this project adopts a SIFT + FLANN–based fingerprint recognition system, balancing accuracy, simplicity, and speed. It eliminates the need for complex minutiae extraction while ensuring robustness to rotation, scaling, and partial distortions, making it ideal for lightweight biometric applications.

Chapter 3 – Methodology :

The proposed fingerprint recognition system identifies individuals by comparing an input fingerprint image with those stored in a database. It is implemented using Python and the OpenCV library, which provide efficient tools for image feature detection and matching. The system mainly consists of four stages: input acquisition, feature extraction, feature matching, and decision-making.

3.1 Input Acquisition

The system automatically reads fingerprint images from two folders: one containing the known fingerprints (database) and another containing the fingerprints to be identified (unknowns). Each image is pre-processed to ensure it is suitable for analysis. This setup allows the system to compare multiple images automatically without any manual steps.

3.2 Feature Extraction Using SIFT

The Scale-Invariant Feature Transform (SIFT) algorithm is used to extract unique and stable points, called keypoints, from each fingerprint image. These keypoints represent distinctive details such as ridge endings, curves, and intersections.

SIFT is robust because it can detect the same features even if the fingerprint image is rotated, resized, or slightly distorted. This makes it ideal for biometric comparison.

At the end of this stage, every fingerprint is represented by a set of

numerical features that describe its unique structure.

3.3 Feature Matching Using FLANN

After feature extraction, the Fast Library for Approximate Nearest Neighbors (FLANN) algorithm is used to compare the keypoints of two fingerprints. FLANN quickly finds which features from one image are most similar to those in another image.

The similarity between two fingerprints is calculated based on the number of closely matching features. A higher number of matching points indicates a stronger likelihood that both fingerprints belong to the same person.

3.4 Decision and Output

Once all comparisons are complete, the system determines the fingerprint with the highest similarity score. If this score exceeds a set threshold, the fingerprint is declared a match; otherwise, it is marked as no match found.

The final result is displayed on the screen, showing whether the unknown fingerprint matches any record in the database.

Chapter 4 - Code:

```
import cv2
import os

# --- 1. Improved Logic: Returns Score AND Visual Data ---
def get_match_details(img1, img2):
    # Initialize SIFT
    sift = cv2.SIFT_create()

    # Keypoints and Descriptors
    kp1, des1 = sift.detectAndCompute(img1, None)
    kp2, des2 = sift.detectAndCompute(img2, None)

    # Safety check
    if des1 is None or des2 is None:
        return 0, None, None, None

    # FLANN Matcher
    index_params = dict(algorithm=1, trees=10)
    search_params = dict(checks=50)
    flann = cv2.FlannBasedMatcher(index_params, search_params)

    matches = flann.knnMatch(des1, des2, k=2)

    # Lowe's Ratio Test
    good_points = []
    for m, n in matches:
        if m.distance < 0.7 * n.distance:
            good_points.append(m)

    # Return: Score, Keypoints1, Keypoints2, The List of Matches
    return len(good_points), kp1, kp2, good_points

# --- 2. Helper to Show Images ---
def show_comparison(title, img1, kp1, img2, kp2, matches):
    # Draw the matches
    result_img = cv2.drawMatches(img1, kp1, img2, kp2, matches, None)
```

```
    # Resize logic (incase images are huge)
    # We resize to a fixed height of 600px to fit your screen
    h, w = result_img.shape[:2]
    new_h = 600
    scale = new_h / h
    new_w = int(w * scale)
    result_img = cv2.resize(result_img, (new_w, new_h))

    cv2.imshow(title, result_img)
    print(f"   >>> DISPLAYING: {title} (Press any key to continue...)")
    cv2.waitKey(0) # Waits for you to press a key
    cv2.destroyAllWindows()

# --- 3. Main Execution ---
def main():
    input_folder = "unknowns"
    database_folder = "database"
    valid_ext = ('.png', '.jpg', '.jpeg', '.bmp', '.tif', '.tiff')

    if not os.path.exists(input_folder):
        print(f"ERROR: '{input_folder}' missing.")
        return

    unknown_files = [f for f in os.listdir(input_folder) if
f.lower().endswith(valid_ext)]

    print(f"--- STARTING VISUAL ANALYSIS ---")
    print(f"Processing {len(unknown_files)} images.\n")

    for u_file in unknown_files:
        print(f" Processing: {u_file}")

        u_path = os.path.join(input_folder, u_file)
        u_img = cv2.imread(u_path)
        if u_img is None: continue

        # ----------------------------------------
        # CHECK 1: DATABASE (Who is this?)
        # ----------------------------------------
        best_score = 0
        best_match_data = None # Stores (name, img, kp, mp)
```

```
    if os.path.exists(database_folder):
        for db_file in os.listdir(database_folder):
            if not db_file.lower().endswith(valid_ext): continue

            db_path = os.path.join(database_folder, db_file)
            db_img = cv2.imread(db_path)
            if db_img is None: continue

            score, kp1, kp2, good_points = get_match_details(u_img, db_img)

            if score > best_score:
                best_score = score
                # Save data so we can draw it later
                best_match_data = (db_file, db_img, kp2, good_points, kp1)

    # REPORT & SHOW DATABASE RESULT
    if best_score > 20:
        name, db_img, db_kp, matches, u_kp = best_match_data
        print(f"    MATCH FOUND: {name} (Score: {best_score})")

        # SHOW THE IMAGE
        show_comparison(f"MATCH: {u_file} vs {name}", u_img, u_kp,
db_img, db_kp, matches)
    else:
        print(f"    Unknown Identity.")

    # ----------------------------------------
    # CHECK 2: UNKNOWNS (Is there a copy here?)
    # ----------------------------------------
    for other_file in unknown_files:
        if other_file == u_file: continue

        other_path = os.path.join(input_folder, other_file)
        other_img = cv2.imread(other_path)
        if other_img is None: continue

        score, kp1, kp2, good_points = get_match_details(u_img, other_img)

        if score > 20:
            print(f"    DUPLICATE FOUND: Same as '{other_file}' (Score:
```

```
{score})")

            # SHOW THE IMAGE
            show_comparison(f"DUPLICATE: {u_file} vs {other_file}", u_img,
kp1, other_img, kp2, good_points)

        print("-" * 50)

if _name_ == "_main_":
    main()
```

Chapter 5- Results and Observation:

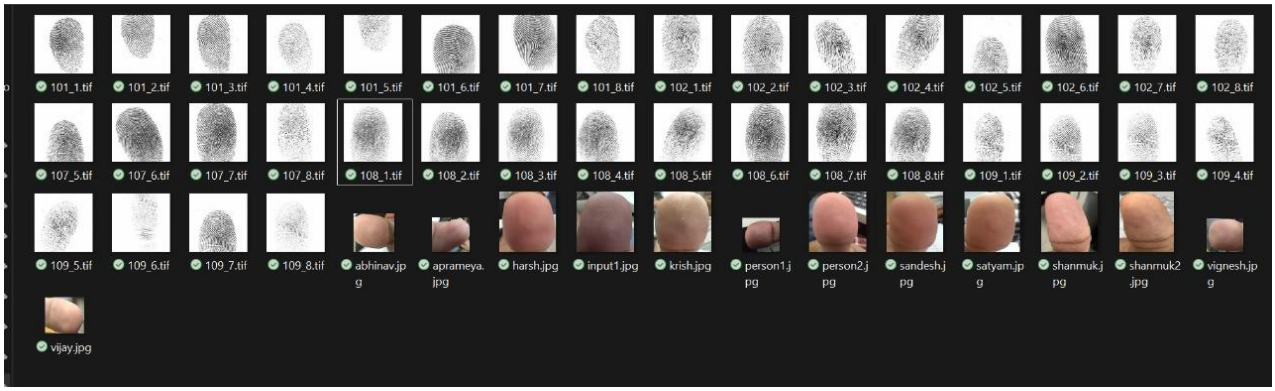

Fig 1: Database folder containing reference fingerprint images used for matching.

Fig 2: Test fingerprint 1 image given as input.

Fig 3: Test fingerprint 2 image given as input (in TIF format).

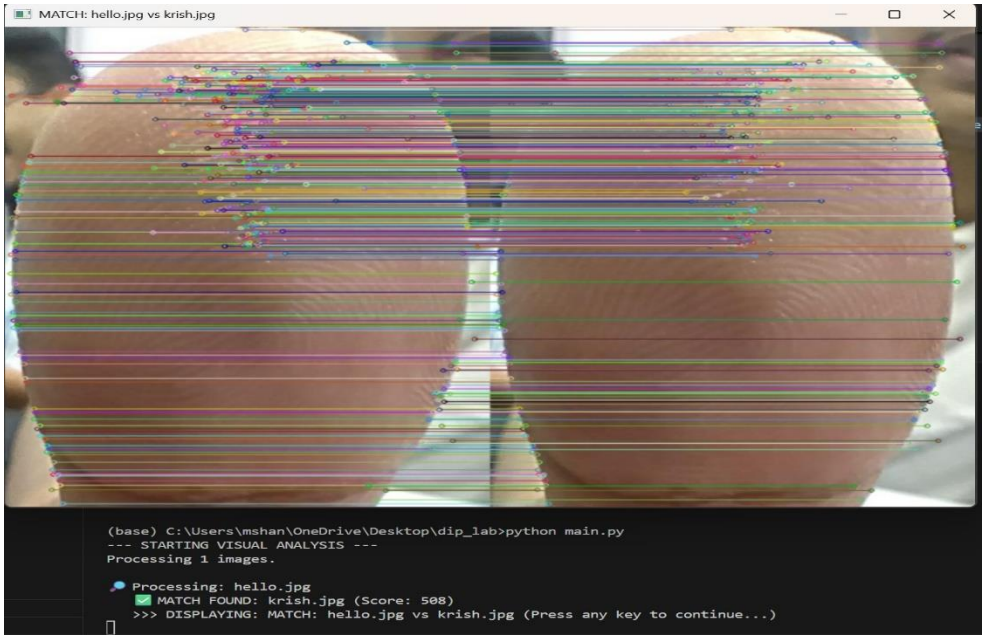

Fig 4:Output showing the matched fingerprint 1 from the database and similarity score.

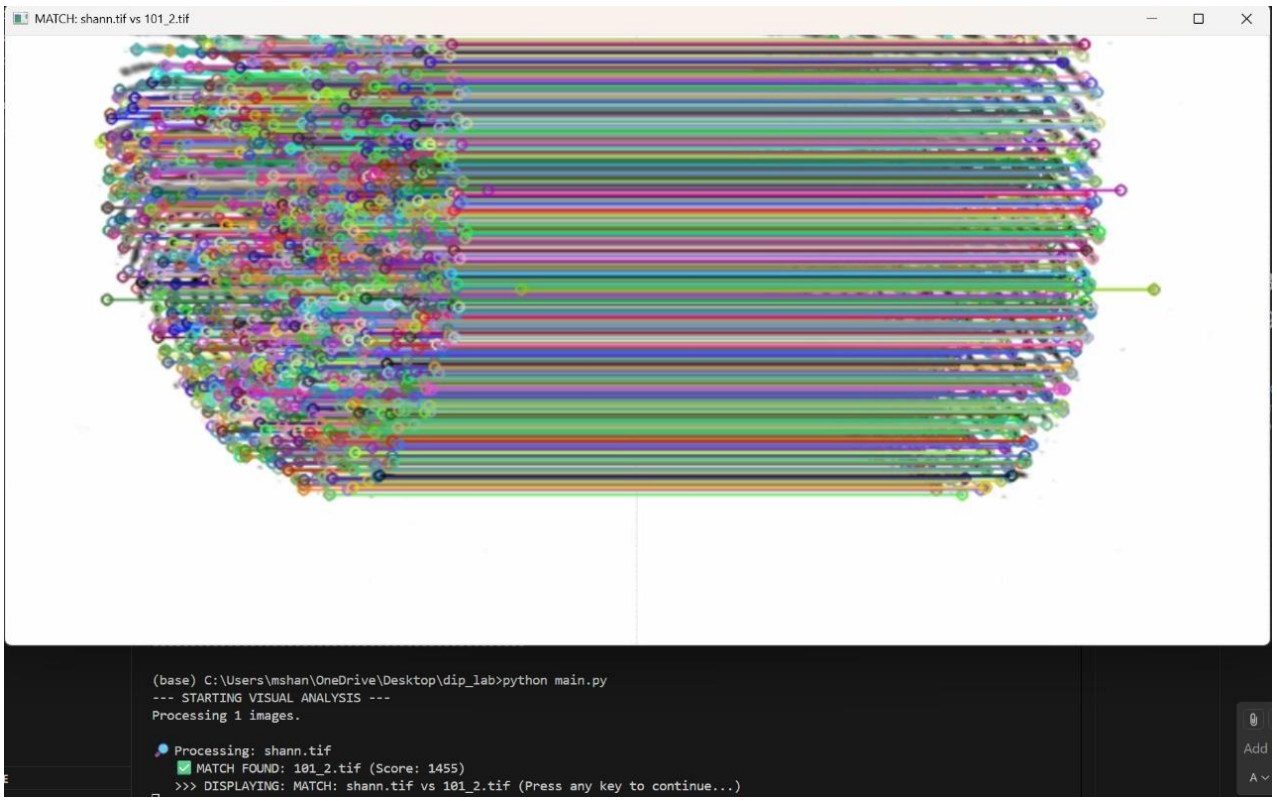

Fig 5 :Output showing the matched fingerprint 2 (TIF format) from the database and similarity score.

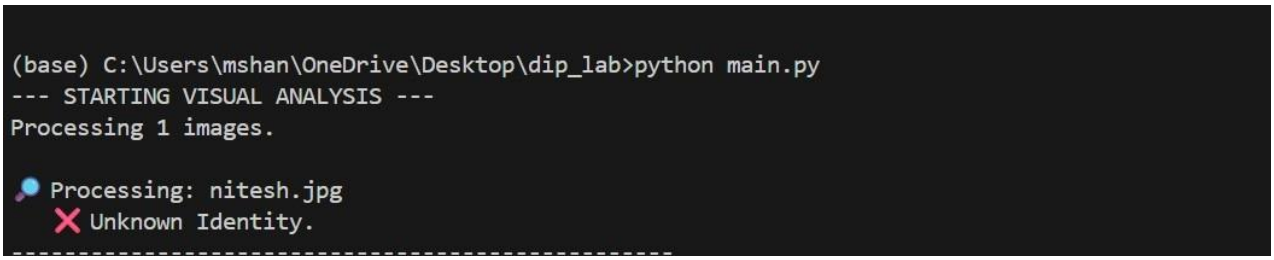

Fig 6 : The system displayed *"Unknown Identity"*, indicating that the input fingerprint did not correspond to any fingerprint stored in the database.

Chapter 6 – Conclusion:

The fingerprint recognition system developed using Python and OpenCV successfully identifies fingerprints by comparing them with images stored in a database. By utilizing SIFT (Scale-Invariant Feature Transform) for keypoint detection and FLANN (Fast Library for Approximate Nearest Neighbors) for feature matching, the system achieves accurate and reliable results even when the input images vary slightly in scale or rotation.

The results demonstrate that this feature-based approach is effective for small-scale biometric verification tasks. It eliminates the complexity of traditional minutiae extraction while maintaining good recognition accuracy. The system efficiently distinguishes between matching and non-matching fingerprints, providing clear textual output and visual confirmation for each comparison.

Overall, the project proves that combining SIFT and FLANN provides a simple, fast, and robust method for fingerprint recognition. It can be used as a foundation for developing more advanced biometric systems with improved accuracy, larger databases, and real-time verification capabilities.