# FINGERPRINT SIMILARITY DETECTION

*Submitted in partial fulfillment of the requirements for the degree of*

# Bachelor of Technology

in

# Computer Science Engineering

*By*


| NAME | REG NO |
|---|---|
| BANDI HARSHAVARDHAN REDDY | 20BCE2265 |
| GOPA LAASYA LALITHA PRIYA | 20BCE2255 |
| KAGANA KARTHIKEYA | 20BCE0490 |
| TEJOMURTULA SARATCHANDRA HEMANTH | 20BCE0510 |

**Under the guidance of**

**Dr.Anusha N**

**SCOPE**

**VIT, Vellore.**



November 2022

# DECLARATION

I hereby declare that the thesis entitled "FINGERPRINT SIMILARITY DETECTION" submitted by me, for the award of the degree of *Bachelor of Technology in Computer Science Engineering* to VIT is a record of bonafide work carried out by me under the supervision of Dr. Anusha N.

I further declare that the work reported in this thesis has not been submitted and will not be submitted, either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university.

Place : Vellore

Date :10-11-22

**Signature of the Candidate**

Kagana karthikeya

# CERTIFICATE

This is to certify that the thesis entitled "FINGERPRINT SIMILARITY DETECTION" submitted by **kagana Karthikeya & 20BCE0490**, **SCOPE**, VIT, for the award of the degree of *Bachelor of Technology in Computer Science Engineering*, is a record of bonafide work carried out by him / her under my supervision during the period, 01. 12. 2018 to 30.04.2019, as per the VIT code of academic and research ethics.

The contents of this report have not been submitted and will not be submitted either in part or in full, for the award of any other degree or diploma in this institute or any other institute or university. The thesis fulfills the requirements and regulations of the University and in my opinion meets the necessary standards for submission.

Place : Vellore

Date :10-11-22 **Signature of the Guide**

**Internal Examiner** **External Examiner**

Head of the Department

Programme

# ACKNOWLEDGEMENTS

I would want to convey my heartfelt gratitude to Dr. Anusha N, my mentor, for her invaluable advice and assistance in completing my project. She was there to assist me every step of the way, and her motivation is what enabled me to accomplish my task effectively. I would also like to thank all of the other supporting personnel who assisted me by supplying the equipment that was essential and vital, without which I would not have been able to perform efficiently on this project.

I would also want to thank the University of VIT for accepting my project in my desired field of expertise.

**Student Name**

- **Kagana karthikeya**

# Executive Summary

The fingerprint is considered a dominant biometric trait due to its acceptability, reliability, high-security level, and low cost. Due to the high demand for fingerprint identification system deployments, a lot of challenges are keeping arising in each system's phase including fingerprint image enhancement, feature extraction, feature matching, and fingerprint classification. Machine learning techniques introduce non-traditional solutions to the fingerprint identification challenges. This project emphasizes the implementation of fingerprint similarity detection code to solve fingerprint problems even though they are altered. This project contributes as ground truth for developing machine learning-based algorithms for fingerprint identification in the near future.

# CONTENTS

# List of Figures

# List of Tables

# List of Abbreviations

SIFT- Scale Invariant Feature Transform

SURF- Speeded Up Robust Features

BRISK- Binary Robust Invariant Scalable Keypoints

AKAZE- Accelerated KAZE

Open -CV – Open Source Computer Vision Librar

# 1. INTRODUCTION

## 1.1. OBJECTIVE

Now a days fingerprints found in the crime scenes are not very clear and accurate it's hard to find out the real match for the found culprit's fingerprint in the crime scene.

So we are Downloading the fingerprint dataset from Kaggle website and using the Z-cut and obliviated versions of the original finger prints to find out the correct fingerprint.

We are also using GIMP Open-source software for modifying the fingerprints we got from the Kaggle dataset to make the fingerprints still hard to find it's match.

Our main Objectives are to alter the fingerprints which are obtained from the dataset and implement fingerprint similarity detection and to detect the fingerprints even if they are highly altered and find the highest accuracy level.

## 1.2. MOTIVATION

In recent times the crimes are increasing exponentially day to day and the fingerprints and the proofs found in the crime scene are not all accurate enough to give a conclusion on guilt culprit, so we are developed a fingerprint matching algorithm to find the best match of the fingerprints found in the crime scene with real time fingerprints.

## 1.3 BACKGROUND

Fingerprint matching is basically extracting the keypoints and descriptors from the fingerprint image and matching them with the original fingerprints and getting the best score / minutiae score.

We are using Open CV which is a library of programming functions mainly aimed at real-time computer vision and the key points are extracted using Feature detecting algorithms and key points and descriptors are matched with the original image using matching algorithms in Open -CV.

# 2.PROJECT DESCRIPTION AND GOALS

In this project we implement Fingerprint Detection with which we can obtain the altered fingerprints left behind in the criminal scene and then detect the real fingerprints with high accuracy by which we can identify the criminal. Similarly, we can use this for many other applications like safety pin codes.

The Goals of the project are:

- To alter the fingerprints which are obtained from the dataset.
- To implement fingerprint similarity detection.
- To detect the fingerprints even if they are highly altered.
- To find the highest minutiae score.

# 3.TECHNICAL SPECIFICATION

We are using Open CV which is a library of programming functions mainly aimed at real-time computer vision and the key points are extracted using Feature detecting algorithms and key points and descriptors are matched with the original image using matching algorithms in Open -CV.

The OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

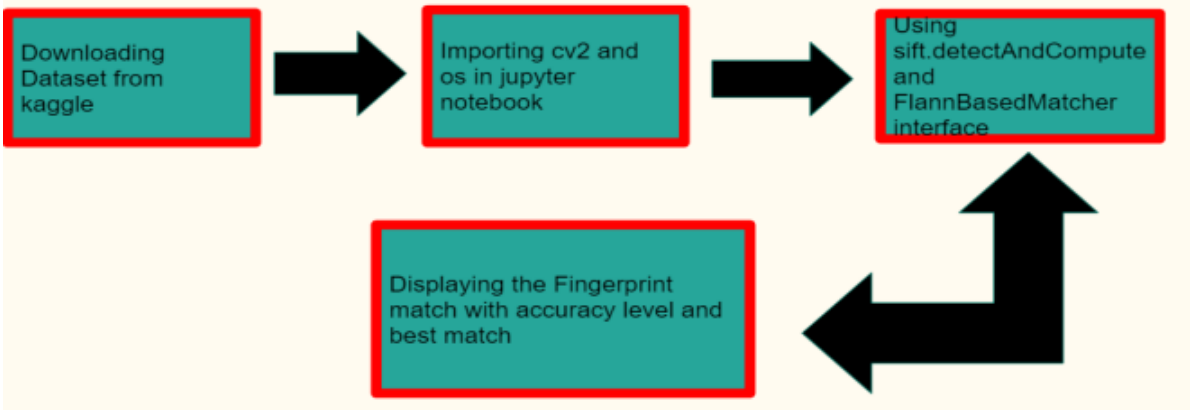
The feature detection algorithm used here is SIFT.SIFT means Scale-Invariant Feature Transform it is proposed by David Lowe it has four main steps which are feature point detection, localization, orientation assignment, and feature descriptor generation. It locates certain key points and then furnishes them with quantitative information which can, for example, be used for object recognition. The descriptors are supposed to be invariant against various transformations which might make images look different although they represent the same objects.

The matching algorithm used is FLANN algorithm. FLANN (Fast Library for Approximate Nearest Neighbors) is an image matching algorithm for fast approximate nearest neighbor searches in high dimensional spaces.

It contains a collection of algorithms optimized for fast nearest neighbor search in large datasets and for high dimensional features. It works faster than BFMatcher for large datasets.

These methods project the high-dimensional features to a lower-dimensional space and then generate the compact binary codes. Benefiting from the produced binary codes, fast image search can be carried out via binary pattern matching or Hamming distance measurement, which dramatically reduces the computational cost and further optimizes the efficiency of the search.

# 4. DESIGN APPROACH AND DETAILS




In our project design, we alter the real fingerprints with different varieties as shown in the image below. We have done this to give a real time example of fingerprint matching, as we don't get clear fingerprints all the time in real life. Here, we are implementing the accuracy level by comparing the fingerprints available in the dataset with the original fingerprint. We will receive the accuracy match level as the output. Our design consists of mainly two parts one matching matching of the keypoints and the other is other is extraction of keypoints. The extraction of keypoints is done by SIFT algorithm and matching is done by Flann-Based matcher which is present in Open-CV library and a best score is extracted from the above on which we will know weather the finger print found in the crime scene is accepted or not.

# 5. SCHEDULE, TASKS AND MILESTONES

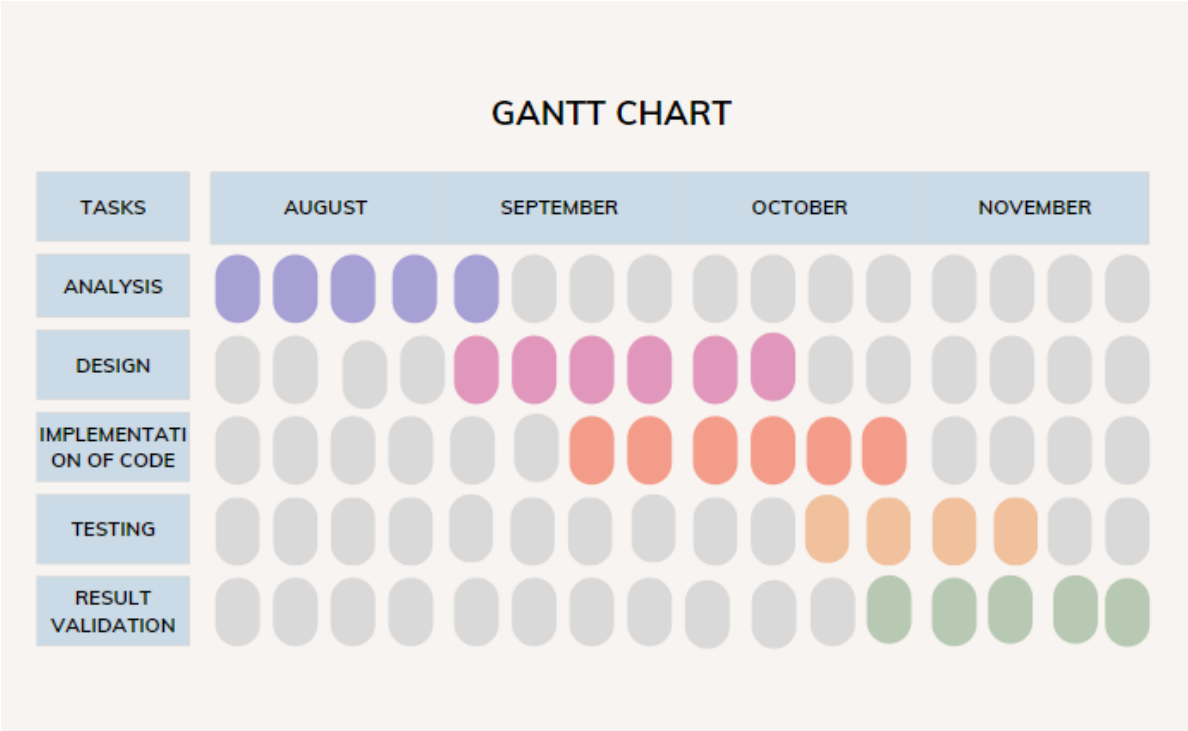



Fig-5.1

## 6.PROJECT DEMONISTRATION

```
import os
import cv2

sample =
cv2.imread("/home/sarath/Downloads/archive/SOCOFing/Altered/Altered
-Hard/9__M_Left_index_finger_Obl.BMP")

best_score = 0
filename = None
image = None
kp1, kp2, mp = None ,None, None
counter = 0
for file in [file for file in
os.listdir("/home/sarath/Downloads/archive/SOCOFing/Real")]:
    if counter % 10 == 0:
        print(counter)
        print(file)
    counter += 1
    fingerprint_image =
cv2.imread("/home/sarath/Downloads/archive/SOCOFing/Real/" + file)
    sift = cv2.SIFT_create()

    keypoints_1, descriptors_1 = sift.detectAndCompute(sample,
None)
    keypoints_2, descriptors_2 =
sift.detectAndCompute(fingerprint_image, None)
    matches = cv2.FlannBasedMatcher({'algorithm': 1,
'trees':10},{}).knnMatch(descriptors_1, descriptors_2, k=2)
    match_points = []
    for p, q in matches:
        if p.distance < 0.1 *q.distance:
            match_points.append(p)
    keypoints = 0
    if len(keypoints_1) < len(keypoints_2):
        keypoints = len(keypoints_1)
    else:
        keypoints = len(keypoints_2)
    if (len(match_points) / keypoints) *100 >best_score:
        best_score = (len(match_points) / keypoints) * 100
        filename = file
        image = fingerprint_image
```

```
        kp1, kp2, mp = keypoints_1, keypoints_2, match_points
print("BEST MATCH: " + filename)
print("SCORE" + str(best_score))
imagefile = cv2.imread(filename)
if best_score > 12:
    print("Matched as per criminal court norms")
else:
    print("Not matched")
result = cv2.drawMatches(sample,kp1,image,kp2,mp,None)
result = cv2.resize(result, None, fx=4, fy=4)
cv2.imshow("Result", result)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

## 7.RESULTS AND ANALYSIS

```
94__M_Left_index_finger.BMP
5940
95__M_Left_index_finger.BMP
5950
96__M_Left_index_finger.BMP
5960
97__M_Left_index_finger.BMP
5970
98__M_Left_index_finger.BMP
5980
99__M_Left_index_finger.BMP
5990
9__M_Left_index_finger.BMP
BEST MATCH: 9__M_Left_index_finger.BMP
SCORE15.625
```

Fig-7.1

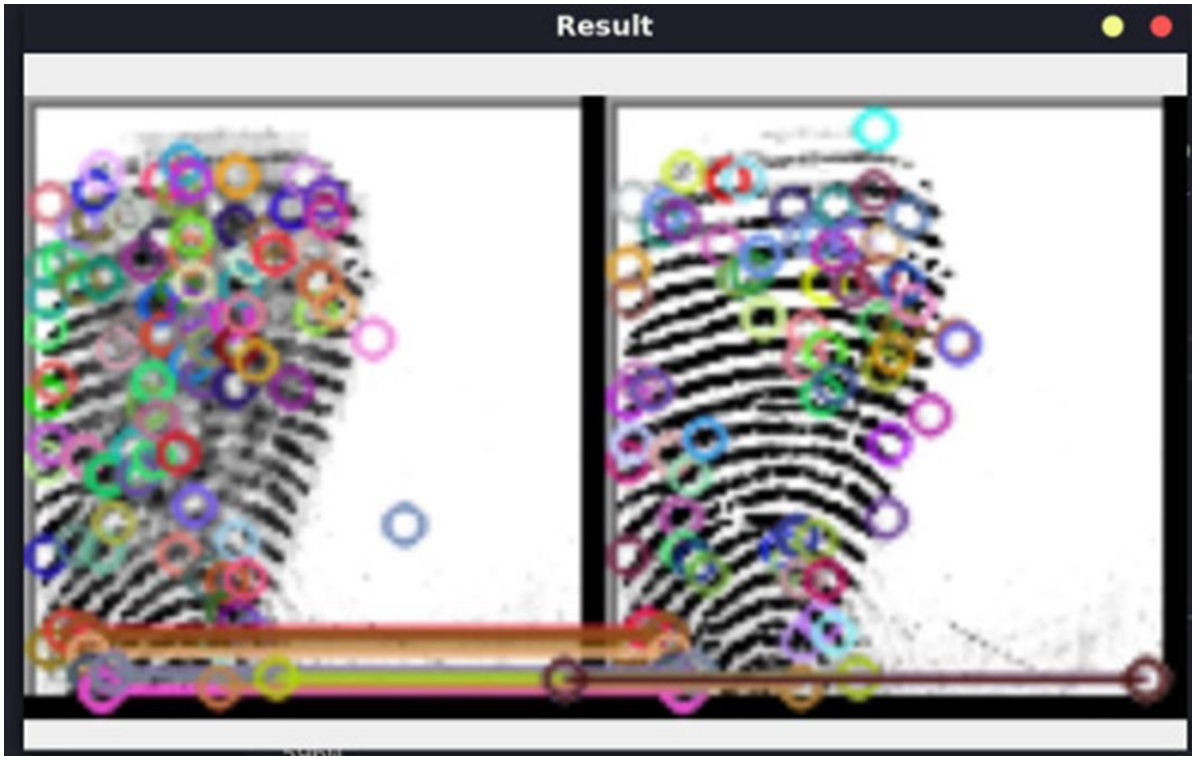



Fig-7.2

In this project the result what we get is the similarity matching of sample image with the real image. Now in this code we SIFT function (i.e., Scale Invariant Feature Transform) which allows us to extract the key points and descriptors for the individual image. Key points are points in the image that are particularly interesting and that stand out in some way, descriptors are ways of describing these key points. In the image above all the coloured circles represent the descriptors of the key points whereas the line joining these key points shows us the matching of key points in both the images. To Show the matching between the two images we use the FlannBasedSearch method. FLANN stands for Fast Library for approximate nearest neighbours. FlannBasedSearch is used to match the sample fingerprint image (altered image) with the real image in the database. At the end of our output, we get the best match of the sample fingerprint with real fingerprint with the accuracy level given below.

## 7.1 CONCLUSION:

| SIFT | SURF | BRISK | KAZE | AKAZE |
|---|---|---|---|---|
| Scale Invariant Feature Transform | Speeded Up Robust Features | Binary Robust invariant scalable keypoints | KAZE | Accelerated KAZE |
| SIFT is used to detect corners, blobs, circles, and so on. It is also used for scaling an image. It can detect features from the image irrespective of its size and orientation. | The SURF method (Speeded Up Robust Features) is a fast and robust algorithm for local, similarity invariant representation and comparison of images. | The BRISK algorithm is a feature point detection and description algorithm with scale invariance and rotation invariance | KAZE Features is a novel 2D feature detection and description method that operates completely in a nonlinear scale space. | Accelerated- KAZE (AKAZE), are considered as the first algorithms to detect features by building a scale space using nonlinear diffusion. |
| Slow but Accurate | SIFT is better than SURF in different scale images | SIFT is more Accurate than BRISK | KAZE does not perform down sampling | SIFT is more accurate than AKAZE |
| Features matched-470.8 | Features matched-178.7 | Features matched-127.5 | Features matched-212.8 | Features matched-231.5 |
| Time- 1.12 | Time-1.93 | Time-1.17 | Time-0.69 | Time-0.25 |

Table-7.1

In this project we have gathered a numerous number of fingerprints from the Kaggle dataset. Later, we altered the fingerprints in 3 different ways the obliterated, Z – Cut, Centre Rotation and in 3 different mediums easy, medium and hard. After the alteration, we tried to match the altered fingerprint with the original one present in the dataset. When we are trying to match the fingerprints, we can see the key points of altered fingerprints matching with the key points on original one. The higher the key points the higher is the accuracy. We can increase the accuracy by training the data set. Finally, we can get the resultant comparison with the corresponding key points matching with each other.

# 8.SUMMARY

Fingerprint recognition systems work by examining a finger pressed against a smooth surface. The finger's ridges and valleys are scanned, and a series of distinct points, where ridges and valleys end or meet, are called minutiae. These minutiae are the points the fingerprint recognition system uses for comparison. In order to identify an individual, the valleys (minutiae) and ridges are used which are found on the surface tips of a human finger.

Fingerprint matching is basically extracting the keypoints and descriptors from the fingerprint image and matching them with the original fingerprints and getting the best score / minutiae score.

We are using Open CV which is a library of programming functions mainly aimed at real-time computer vision and the key points are extracted using Feature detecting algorithms and key points and descriptors are matched with the original image using matching algorithms in Open -CV.

# 9 References


[1] Amandeep Kaur and Ameeta, Babita," Minutiae Extraction and Variation of Fast Fourier Transform on Fingerprint Recognition",International Journal of Engineering Research and General Science Volume 2, Issue 6, OctoberNovember, 2014. Domenico Bloisi, Luca Iocchi, Michele Fiorini, Giovanni Graziano, "Automatic Maritime Surveillance with Visual Target Detection", International Defense and Homeland Security Simulation Workshop (DHSS), September 2011.

[2] Vaibhav Jain and Ajay Kumar Singh, "An approach for Minutia Extraction in Latent Fingerprint Matching",International Journal,ISSN:2319–1058, Volume 6, No. 1 , pp. 51-58, October 2015.

[3] https://ieeexplore.ieee.org/document/8346440

[4] https://bolster.ai/blog/fast-image-matching-at-scale

[5] https://www.analyticsvidhya.com/blog/2021/06/feature-detection-description-and-matching-of-images-using-opencv/

[6] https://link.springer.com/chapter/10.1007/978-3-642-35326-0_52