UNIVERSITY OF GHANA

A COMPARATIVE STUDY OF FINGERPRINT MATCHING ALGORITHMS

ID	FULL NAME
10736694	Terence Ugo Nacciarone Quashie
10729461	Abdul-Aziz Abubakar Saddick

UNIVERSITY OF GHANA

A COMPARATIVE ANALYSIS OF FINGERPRINT MATCHING ALGORITHMS

BY

TERENCE UGO NACCIARONE QUASHIE

ABDUL-AZIZ ABUBAKAR SADDICK

Thesis submitted to the Department of Computer Science of the
University of Ghana, Legon, in partial fulfilment of the requirements for the award of Bachelor of Science degree in Computer Science

NOVEMBER 2022

DECLARATION

Candidates' Declaration

We hereby declare that this thesis is the result of our own original research and that no part of it has been presented for another degree in this university or elsewhere.

Candidate's Signature …… Date ……

Name: Quashie Terence Ugo Nacciarone ID: **10736694**

Candidate's Signature …… Date ……

Name: Abdul-Aziz Abubakar Saddick ID: **10729461**

Supervisors' Declaration

We hereby declare that the preparation and presentation of the thesis were supervised in accordance with the guidelines on supervision of thesis laid down by the University of Ghana.

Principal Supervisor's Signature …… Date ……

Name: Dr. Agbo Tettey Soli Micheal

ABSTRACT

The success of fingerprint patterns systems known as the AFIS (Automatic Fingerprint Identification Systems), in criminal investigations has spurred its application in a second generation geared towards civilization processes such as biometric authentication. Two of the most popular techniques developed are the Minutiae and the Scale Invariant Feature Transformation (SIFT) each with its advantages and disadvantages. The performance of these two techniques on a set of altered fingerprints were examined. A python-based programme was developed to run both algorithms of 900 images of fingerprints (both normal and altered). The SIFT algorithm gave 37% good matches as compared to 28% using Minutiae. On the other hand, Minutiae gave higher (almost 59%) no match as against the SIFT's low (almost 21%) no match. The SIFT algorithm is more likely to accurately identify subjects with (some sort of distortion on their fingerprints) such as cuts, dust or skin-oil on the fingerprint or on the scanner. Minutiae is less likely to identify subjects with distortions or aberrations on their fingerprints. Minutiae algorithm used 0.0000095 seconds to process 500 images while the same number of images were processed by the SIFT algorithm in 0.0000799 seconds Minutiae based algorithm will run faster on larger datasets as compared to SIFT, however its accuracy is reduced. Both algorithms, however, have good performance, running under 3 milli seconds on a database containing 500,000 sample fingerprint images.

KEY WORDS

Minutiae

Scale Invariant Feature Transformation (SIFT)

Python Programming Language,

Graphical User Interface (GUI)

Fingerprint,

Gaussian Scale Space,

ACKNOWLEDGEMENTS

Our first and greatest acknowledgement is to the Holy Trinity, Father, Son and Holy Spirit, and Allah for being with us all these years and especially during the trying moments of this project.

We would like to thank our supervisor, Dr. Agbo Tettey Soli Micheal and lecturers who taught and mentored us during this course.

Our appreciation also goes to our friends, especially Etornam, for giving us access to his computer while we had issues with ours

DEDICATION

To our parents and brothers,

Andy, Doris, Denis, and Louis Quashie

Abdul-Aziz, Bariyatu, Muna, Abdul-Rahman

Table of Contents

List of Figures

List of Tables

Chapter 1: Introduction

In 1983 when the Home Ministry Office, UK, concluded that no two individuals can have the same fingerprints, it set in motion a series of events that led to the widespread use of fingerprint pattern systems, known as the AFIS (Automatic Fingerprint Identification Systems). These systems are actively used by law enforcement agencies all over the world today. In fact, these fingerprint matching systems have become so successful in criminal investigations that the term fingerprint has become synonymous with the word inherent characteristic or unique characteristic.

The success of fingerprint identification systems has spurred a wide spiral of its application beyond the forensic domain to a "second generation" geared towards civilization application such as biometric authentication. The second generation of fingerprint identification systems operate automatically and are deployed in more mainstream applications that deal with a larger cross-section of society such as unlocking mobile phones or gaining access to a secure app such as a banking or blockchain app or gaining access to homes using smart locks.

It is an undeniable fact that the current implementation of fingerprint identification systems has positively reduced cases of identity theft in our society. However, the "million-dollar" question still remains; "Will these fingerprint biometric technologies work all the time? Will they work everywhere and, in all contexts, reliably identifying and authenticating a person". These are questions that have pushed researchers to develop newer and more efficient methods of automatic fingerprint identification and matching.

One of the design criteria for building such an automatic and reliable fingerprint verification system is that the underlying sensing, representation and matching technologies must be very robust. It is essential that these individual components have little to no degradation in their

performance in the long term. One way to address these requirements of robust performance is to adopt a robust representation scheme that captures all discriminatory information to a high degree of accuracy.

The most popular representation of these discriminatory information is through local landmarks known as M**inutiae**. This method evolved from the system of visually matching fingerprints used by forensic experts. **The minutiae-based algorithm** works by locating these local landmarks where fingerprint ridges either terminate or bifurcate and then match minutiae relative placements between a given fingerprint sample and the stored template. A good quality fingerprint contains between 25 and 100 minutiae depending on sensor resolution and finger placement on the sensor. The minutiae-based system is a well-known method for fingerprint verification.

Although the minutiae based-system is robust and the most widely used method of fingerprint verification, it struggles when given poor quality fingerprint impressions arising from dry fingers or fingers mutilated by scars and scratches. There is also anecdotal evidence that a fraction of the population may have fingerprints with relatively small number of minutiae thereby making it more vulnerable to failures.

This short-coming of the minutiae-based system meant that there was need to extend characteristic feature matching beyond minutiae points. For this, techniques such as the **Scale Invariant Feature Transformation (SIFT)** was introduced, an object matching algorithm. SIFT works by constructing a scale space from which descriptors are extracted and used in the matching process.

Using a public domain fingerprint dataset, this project will implement the two algorithms and compare their performance and useability.

Objectives

Algorithm Implementation

The minutiae-based algorithm and the scale invariant feature transform (SIFT) algorithm will be implemented using the python programming language. Both algorithms will recognize unique features in the same fingerprint sample.

Feature Recognition

Following the implementation, both algorithms would be tested against sample data to test their feature recognition capabilities.

Matching Using Features

Using the same fingerprint sample from the dataset, the matching capabilities of each algorithm will be tested along with their performance documented and represented in a graphical form. The fingerprint sample would also be distorted to further test the matching capabilities of both algorithms.

Compare Performance

Based on the values obtained from the previous objectives, the performance of both algorithms on the sample data will be tested using various performance metrics such as total time to process an image will be taken and analyzed.

Sample Data

The Kaggle Coventry Fingerprint dataset will be used in the undertaking of this project. The database contains over 6,000 (Six Thousand) sampled fingerprint images, as well over 14,000 (Fourteen Thousand) alterations of these fingerprint images. The sample images are a monotone

set with a resolution of 96 x 103 pixels. A sample will be taken from this pool and compared with their altered version using both algorithms.

Procedure

A theoretical approach will be taken detailing the important constituents of both algorithms including the various techniques used in their implementation. These two algorithms will then be implemented using the python programming language. Various python libraries will be used to visualize the data in graphical form such as **PYQT5** for the user interface where the path to the sample image can be specified for the algorithm to run.

Analyses

The performance of the two algorithms will be measured, compared and represented in a graph and tabular form to visualize the differences between the two algorithms. The performance metrics to be used include

i. Time to complete algorithm

ii. Number of minutiae extracted and successfully matched

Discussions of Findings, Conclusion and Recommendations

Results from the analyses will be discussed and conclusions drawn. Recommendations for further work will also be suggested.

Chapter 2: Fingerprint Matching Algorithms

A fingerprint matching algorithm is one that revolves around comparing previously stored templates of fingerprints against candidate fingerprints usually for the purpose of authentication (Wang, Gavrilova, Luo, & Rokne, 2006). The algorithm allows for a person to be identified or verified

Two types of algorithms are prevalent; these are:

1. Scale-Invariant Feature Transformation, and
2. Minutiae-based Fingerprint Recognition and Matching

Scale-Invariant Feature Transformation

Scale-Invariant Feature Transformation **(SIFT)** is a computer vision algorithm to describe, detect and match local features in digital images. It locates certain keypoints and then furnishes them with **quantitative information** or **descriptors** that can be used for object recognition. (Wechsler & Li, 2014) The Scale-Invariant Feature Transform (SIFT) algorithm consists of two successive and independent operations. These are

i. The detection of interesting points (keypoints)

ii. The extraction of a descriptor associated to each of the keypoints.

SIFT is useful for fingerprint matching because it produces unique descriptors on each fingerprint making it a convenient technique to match two fingerprint images. Due to the nature of these descriptors, comparison between samples can be undertaken (regardless of how distorted the image being used for comparison is) because a match is still generated.

Descriptors

Descriptors are quantitative information about a keypoint. These descriptors are invariant against various transformations such as image translation, rotation and scaling which might make images look different although they represent the same object. (Edmund, 2016)

SIFT descriptors have also proved to be robust to a wide family of image transformations including viewport changes, noise, blur, scene deformation and contrast changes while remaining discriminative enough for matching purposes. Since these descriptors are robust, they are usually used for matching pairs of images. (Otero, Delbracio, & Mauricio, 2014).

Keypoints

Keypoints are interesting points whose center position and characteristic scale are accurately located. For each keypoint, the size, center and orientation are normalized. Due to this normalization the keypoints remain invariant to any translation, scale change or rotation.

SIFT detects a series of keypoints from multiscale image representation. This multiscale representation consists of a set of increasingly blurred images. From these keypoints, descriptors are generated.

Video stabilization is another popular application of the SIFT method, however, the scope of this research will be limited to image recognition, more specifically finger print image recognition.

Obtaining keypoints and descriptors

To obtain these keypoints and descriptors, various steps and frameworks are constructed and used.

The Gaussian Scale-Space Construction

Keypoints in SIFT are invariant to scale-changes. In order to attain this scale invariance, SIFT is built on a Gaussian Scale-Space.

A Gaussian Scale-Space is a multiscale image representation simulating the family of all possible zoom-outs through increasingly blurred versions of the input image (Otero, Delbracio, & Mauricio, 2014)This blurring process simulates the loss of detail produced when a scene is photographed from farther and farther. The scale-space, therefore provides SIFT with scale invariance as it can be interpreted as the simulation of a set of snapshots of a given scene taken at different distances. This scale space is constructed by applying a variable gaussian operator on an input image. This is also known as **Gaussian Blurring**.

Gaussian blurring is a technique used by the SIFT algorithm to properly prepare images for scale space construction. This technique is a widely used effectively in graphics software, typically to reduce image noise or image graininess. However, in computer-vision-based algorithms, it is used as a preprocessing technique in order to enhance images at different scales (Haddad & Akansu, 1991)

In other areas of computer vision, the Gaussian blur is also used as a way to detect edges. Since most edge detection algorithms are sensitive to noise, the gaussian blur serves as a way to reduce the noise in order to make edge detection more accurate.

The Gaussian blur technique will be used to prepare fingerprint images for the detection of where the fingerprint arc, whorl or loop begins.

Difference of Gaussians

Another refinement technique used is the Difference of Gaussians **(DoG)**. This refers to a feature enhancement technique that involves the subtraction of one Gaussian Blurred version of an original image from another less blurred version of the original (Davidson, 2016).This results in a set of Gaussian-Smoothed images known as octaves.

The Difference of Gaussians technique ensures that the spatial information that lie between the range of frequencies are preserved between the two blurred images, these include visibility of edges and any other keypoints present in the digital image.

3D extrema

In SIFT, candidate keypoints are defined as the **3D extrema** of the normalized scale-space. The extrema are detected by observing each image point in the **Difference of Gaussian (DoG)**. A point is decided as a local minimum or maximum when its value is smaller or larger than all its surrounding neighbor points by a certain amount. If an extremum is decided as unstable or is placed on an edge, it is removed because it cannot be reliably detected again with small variation of the viewport or lighting changes.

Calculation of 3D extrema

Continuous 3D extrema of the digital DoG are calculated in two steps. Firstly, the 3D discrete extrema are extracted from each octave with pixel precision. Next, their location is refined through interpolation of the digital DoG by using a quadratic model such as the **Taylor expansion**.

The resulting image is then compared to its neighbors to detect the 3D discrete maxima and minima. These comparisons are possible due to the auxiliary images in the DoG.

Although this process produces candidate keypoints that can be used, it is prone to noise and as such produces unstable detections. The candidate keypoints chosen may therefore be flawed since they are constrained to the sampling grid.

Filtering Unstable Candidate Keypoints

Noisy images produce erroneous candidate keypoints thereby making them unstable and unlinked to any particular structure in the image.

SIFT attempts to eliminate these false detections by discarding those candidate keypoints found outside the DoG threshold on distance ratio defined as:

(*distance to nearest candidate keypoint neighbor) : (distance to second nearest candidate keypoint neighbor)***.***

Unstable key-points are those on the edges of the image. These candidate keypoints are difficult to precisely locate due to the fact that an edge is invariant to translations along its principal axis. Such detections do not help define covariant keypoints and are also discarded.

Orientation Assignment

After a set of stable candidate keypoints have been generated, an orientation is assigned to each of these keypoints to make them invariant to rotation (Singh, 2019). This is done by computing the magnitude and orientation for each keypoint and then constructing a histogram to determine the peak orientation for a particular keypoint.

An example is shown below in Figure 1

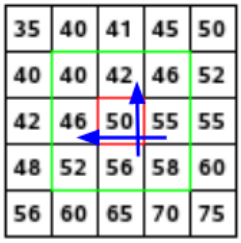

Figure 1:Pixel matrix

Consider the matrix above which is a matrix of pixels. To compute the orientation and magnitude for the pixel in red, the gradients in both the x and y directions are calculated as follows

$$Gx\ (gradient\ of\ x) is\ obtained\ by\ subtracting\ 46\ from\ 55\ giving\ us\ Gx = 9$$

$$Gy\ (gradient\ of\ y) is\ obtained\ by\ subtracting\ 42\ from\ 56\ giving\ us\ Gy = 14$$

$$Magnitude = \sqrt{(G_x)^2 + \left(G_y\right)^2} = 16.64$$

$$\Phi = atan(Gy / Gx) = atan(1.55) = 57.17$$

The magnitude represents the intensity of the pixel while the orientation represents the direction of the pixel. From this a histogram is created by plotting the magnitude and orientation value for all the pixels to obtain the peak orientation. An example is shown below in Figure 2*;*

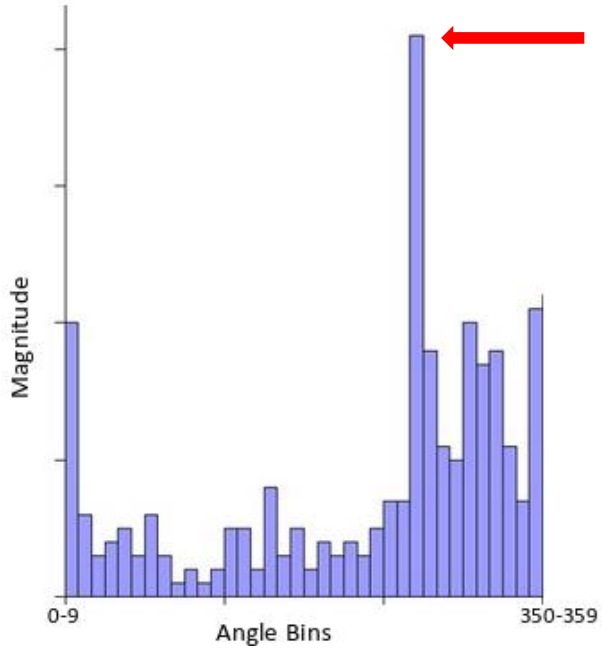

Figure 2: Histogram for orientation assignment

At some point the histogram peaks (arrowed in figure 2) and from this the orientation of the keypoint is determined making it invariant to rotations.

Keypoint Descriptor

From the unique orientation-invariant keypoints, ***Descriptors*** are obtained. Descriptors fall into two categories (Singh, 2019):

i. Descriptors based on properties of the image that are already rotation-invariant

ii. Descriptors based on a normalization with respect to the reference orientation

For the scope of this project, we will use descriptors based on category (i) above

Matching of Descriptors

Descriptors between two images are matched by identifying their nearest neighbors. In the event that the descriptors are too close to each other due to image noise, the ratio of the closest distance to second closest distance is taken.

The standard ratio for this distance is 0.8 and if they are greater than this, the points are rejected. This ensures that 90% of false matches are eliminated while only discarding 5% of correct matches. (Tyagi, 2019)

Using descriptors for matching purposes work due to the following reasons:

- Keypoints are extracted at different scales and blur levels and all subsequent computations are performed within the scale space framework. This makes the descriptors invariant to image scaling and small changes in perspective.
- Computation relative to a reference orientation makes the descriptors robust against rotation.
- The descriptor information is stored relative to the keypoint position and thus invariant against translations.
- Many potential keypoints are discarded if they are deemed unstable or hard to locate precisely. The remaining keypoints are thus relatively immune to image noise.
- The histograms are normalized meaning the descriptors do not store the magnitudes of the gradients, only their relationships to each other. This makes the descriptors invariant against global, uniform illumination changes.

- The histogram values are 'thresholded' to reduce the influence of large gradients. This makes the information partly immune to local, non-uniform changes in illumination. (Edmund, 2016)

Use of SIFT on Fingerprint Images

Since SIFT keypoints are limited only by the condition of the local minima or maxima in a given scale space, a large number of keypoints on a fingerprint image are detected. These keypoints are determined by a set of parameters including the number of octaves. Typical fingerprints contain up to a thousand keypoints (Park, Pankanti, & Jain, 2008).

Use of SIFT on fingerprint images can be summarized into three (3) steps summarized in the flow chart below in ***Figure 3***

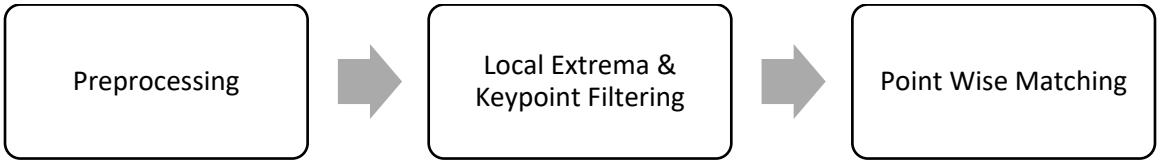

Figure 3: Flow for fingerprint matching using SIFT

Preprocessing (Obtaining scale space)

The first phase, which involves obtaining the scale space of the fingerprint image, is done by blurring the images using a **Gaussian blurring** method to simulate the different zoom levels of the image indicated by the orange arrow (→) in figure 4. Further blurring is done as indicated on its left neighbor or by the green arrow (→) as shown in ***Figure 4***.

To further make the scale space robust, the images can be scaled-down to better simulate different zoom levels as indicated by the blue arrow (→). Each row is known as an **octave**. Image representation of generating scale space shown in Figure 4.

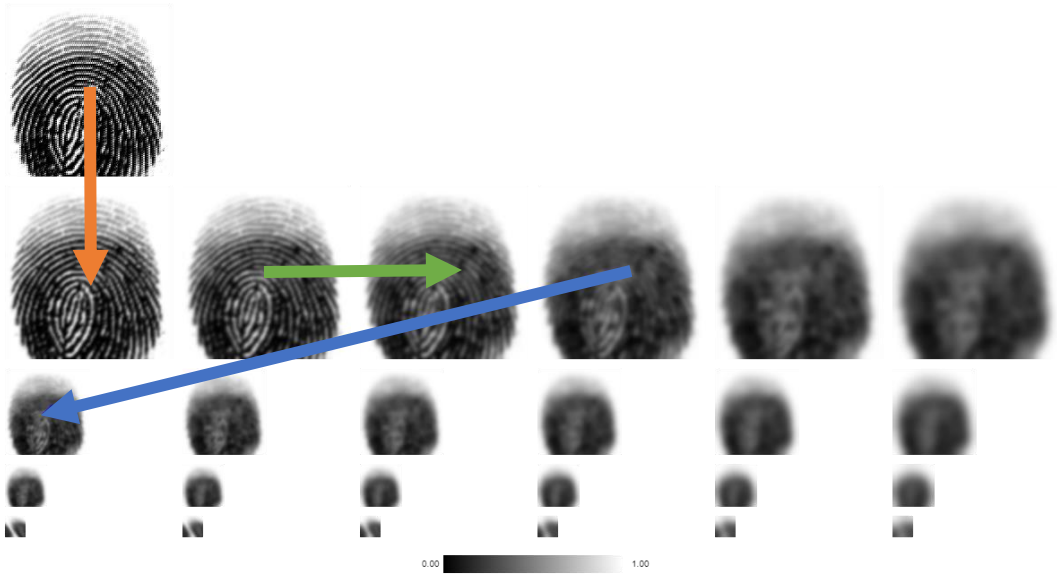

Figure 4: Scale space of fingerprints

Local Extrema & Descriptor Extraction

To find the local Extrema in this phase, we first compute the **Difference of Gaussians** (DoG).

This can be done by gray-scaling each image in the octave as shown in Figure 5 and calculating

the difference between each pixel in the adjacent image as shown in Figure 6.

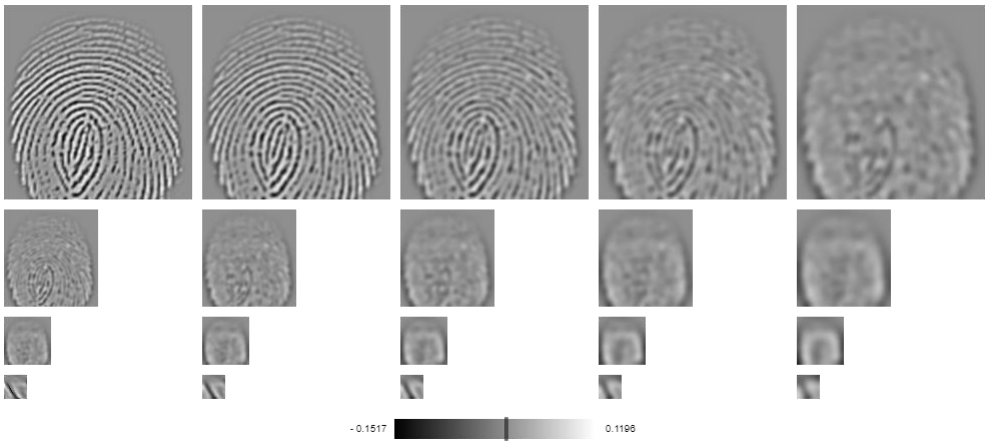

Figure 5: Grayscaling images

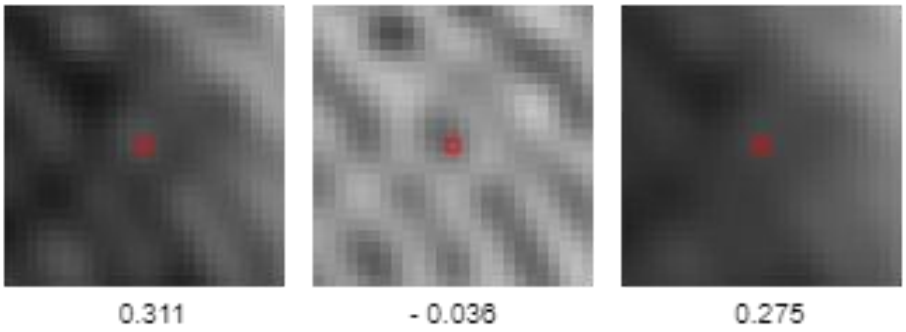

Figure 6: Calculated differences in pixel

The extremum/candidate keypoint then becomes a pixel whose gray value is larger than all of its neighboring pixels as shown below in Figure 7.

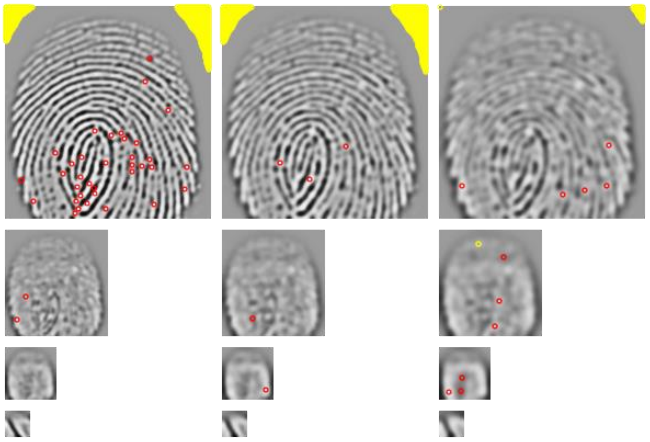

Figure 7: Candidate keypoints/extrema
Area marked in yellow ( ) are indeed extrema/candidate keypoints, but their absolute values are so low that they are discarded.

These extrema usually exist as a result of image noise. The extrema are then filtered using the quadratic **Taylor expansion** of scale space-function. This is an iterative process that refines the location of a keypoint.

Following the first filtering, keypoints that lie on the edges are then identified and discarded. These points are not invariant to translations parallel to edge direction hence they are discarded.

In the example being used here, the remaining keypoints are shown in Figure 8:

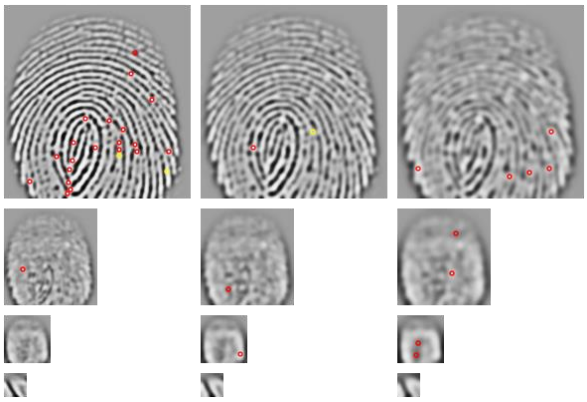

Figure 8: Refined keypoints

A reference orientation is now calculated making the keypoints invariant to rotation as described earlier in ***Orientation Assignment*** resulting in descriptors.

Descriptors that do not have enough pixels to compute a reference orientation are discarded.

Descriptors without a dominating orientation are also discarded resulting in the image as shown in Figure 9.

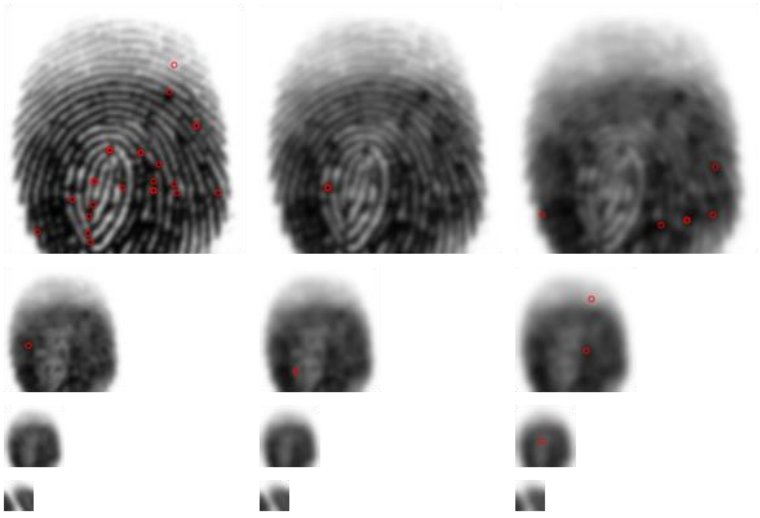

Figure 9: New keypoint after orientation assignment

Point Wise Matching

This final set of descriptors are then compared to the descriptors of another image even if they are depicted with different illumination, slightly distorted or with a different perspective. This comparison is effective/valid as discussed in the section under ***Matching of Descriptors*** above.

Figure 10 illustrates this comparison between a fingerprint and its distorted version.

Figure 10: Matching of two descriptors

Minutiae-based Fingerprint Extraction and Recognition

The minutiae-based fingerprint extraction and recognition algorithm is a **set of techniques** used to extract feature points or descriptors known as **"minutiae"**. Minutiae are small plot points on a fingerprint. They include characteristics such as **ridge bifurcation, the ridge ending** and **the orientation**.

A ridge bifurcation is defined as the point where a ridge diverges into branch ridges, whereas a ridge ending is the point where a ridge ends abruptly. The orientation on the other hand is defined as the local ridge orientation of the associated ridge (Zaeri, 2011)

The number of minutiae points on each fingerprint sample differ from one fingerprint image to another. Depending on the resolution of the fingerprint image, one fingerprint can generate up to 100 minutiae features (Zaeri, 2011)

These extracted minutiae features are then compared to the feature points of other fingerprint template images for fingerprint recognition and matching. The minimum number of matching pairs of minutiae required to declare that two template fingerprints match is at least 12 minutiae features (Amina, Dominique, & Youcef, 2022).

Procedures involved in extracting minutiae features

To obtain these minutiae points, a series of frameworks are constructed and used. These are explained below.

Normalization

In an ideal fingerprint image, lines of ridges flow in a constant direction. However, due to certain conditions such as, cuts and wetness of the skin, incorrect finger pressure when taking a reading or image noise generated as a result of the sensor, poor quality images are generated which may

lead to the generation of multiple false minutiae hence the need for normalization. (Cao & Wang, 2016)

Normalization, also referred to as histogram stretching or contrast stretching, is a process which involves changing the range of pixel intensity values (Fisher, Perkins, Walker, & Wolfart, 2003). The end result is a high contrast image with much clearer details**.** In the case of fingerprint images, it makes finer details such as ridges much more "refined".

In minutiae extraction, one of the most popular choices of normalization is the Gabor Filter. **The Gabor filter** is a linear filter used as a normalization technique. It also functions as a tool for feature extraction, edge detection and texture analysis. The Gabor filter amplifies a band of frequencies and rejects the others (Shah, 2018). By rejecting some of these color frequencies, the variation between gray values along the ridges are also reduced thereby making it easier to have more contrast in the image.

Binarization/Segmentation

Image binarization is the process of taking a normalized grayscale image and converting all its pixels it black and white, essentially reducing the color information from 255 shades of gray to two colors, black and white with values of 0 and 255 respectively.

Binarization is done to preserve the characteristics of the ridge structure while removing some of the cohesion between patterns. (Erwin, Karo, Sari, Aziza, & Putra, 2019). The next step after normalization is usually to binarize that image.

The process of binarization works by finding a threshold value in the color histogram. The threshold value is a value that divides the histogram into two parts. Each representing one of two

layers; the background layer and the object itself, and case, the pattern contained in the fingerprint.

The threshold value cannot be accurately calculated but a good prediction can be made by dividing the image into a $w \times w$ block and its mean and variance, calculated for each block.

Next, the average, the mean and variance of all the blocks are calculated. From this average, the relative mean and variance are also calculated. The threshold value for foreground, in this case the ridges of the fingerprint, will be represented as the average of the variance and the background, the average mean. (Al-Najjar & Sheta, 2008)

Fast Fourier Transform

The Fast Fourier Transform is an image enhancement technique used to transform an image between the spatial and frequency domain**.** Following the binarization process, some of the ridges may be broken as a result of having their pixel values greater than the threshold value and as such assigned values of 255 or white. The Fast Fourier Transform (FFT) is then used to reconnect these broken ridges.

Any image represented in a frequency domain has two major components.

- High Frequency components which correspond to the edges in the image
- Low Frequency components which correspond to the smooth regions of the image

Fast Fourier transformation decomposes the source image into its spectral information. For instance, its **directional information** which includes respective sines and cosines which reveal a repeating pattern within the image. This orientation data allows the reconnection of broken ridges.

By transforming the source image into the frequency domain, the FFT preserves all original data. (Amina, Dominique, & Youcef, 2022) in addition to further removing noise that may exist in the image

Dominant frequencies refer to a sinusoid which consists of the following:

- Spatial Frequency – Deals with brightness of the image
- Magnitude – Deals with contrast
- Phase – Refers to color information

Thinning

Thinning is an image processing technique that involves reducing the thickness of each line of ridge pattern till the width is shrunk enough to become a single or one (1) pixel. This process is done to identify the exact pattern of the fingerprint which in turn makes feature/minutiae extraction possible. (Al-Ani, 2013)

For an image to be considered properly thinned, each ridge should be thinned to its central pixel. After an image is thinned, further removal of pixels is not possible.

Minutiae Extraction

In this stage, all minutiae feature points, that is, ridge ending and bifurcations have been refined and are ready for extraction. Appropriate minutiae feature points are extracted by applying a filter of a 3×3 matrices over the region of interest (ROI) in the thinned image by following some rules:

i. If the central pixel is ‘1’ and the sum of pixels inside the block is ‘2’, then that central pixel is a ridge termination

ii. If the central pixel is ‘1’ and the sum is ‘4’ then the central pixel is a bifurcation

iii. If the central pixel is some other value than ‘1’, then the central pixel is a regular pixel with no unique point (Al-Najjar & Sheta, 2008)

Region of interest (ROI)

Region of interest (ROI) is the area of the thinned image of interest. The region of interest is found by applying some morphological operations such as opening area, filling, erosion or closing. Specifying the ROI allows for minutiae suppression outside the region. (Al-Najjar & Sheta, 2008)

Minutiae Orientation

Once the minutiae points have been determined, the orientation for both the ridge termination and bifurcation are calculated.

Termination Orientation

This is found by using a table of 3×3 dimensions for different angles of theta. The positions of pixels connected to the center in a 3×3 ROI block. By keeping the edge pixels, the first non-zero pixels are taken and its location compared to the table to find the corresponding angle for that termination.

Bifurcation Orientation

For each bifurcation, there are three bounding non-zero pixels hence the procedure for termination orientation calculation is applied three times instead of once. (Al-Najjar & Sheta, 2008)

Terminating False Minutiae

False minutiae consist of broken ridges, minutiae adjacent to each other and minutiae near the borders. Algorithms such as the Fast Fourier Transform aim to reduce the generation of these

points, however, as they are not perfect, some false minutiae will still be generated. (Maddala & Tangellapally, 2010). False minutiae adversely affect matching and as such they need to be trimmed down or removed.

Minutiae Matching

Once the false minutiae have been trimmed, a set of the pixel coordinates and orientation angles are generated from the source image. This data is then compared to the data set of the target fingerprint to determine if the source and the target fingerprint images match. (Al-Najjar & Sheta, 2008)

Using Minutiae Extraction Method on Fingerprint Images

The following set of images depict the various phases of the minutiae extraction method

Fingerprint sample | Normalized Image | Binarized Image | Thinned Image | Region of Interest | Extracted Minutiae

Chapter 3: Methodology

The aim of the research is to effectively compare two fingerprint matching algorithms; minutiae matching and Scale Invariant Feature Transformation (SIFT) matching. Implementation of both algorithms were done using the Python Programming language accompanied by various libraries such as PYQT5 for the Graphical User Interface (GUI), NumPy for various math functions, OpenCV2 library for image processing as well as matplotlib for data visualization. Detailed in the next sections are the various processes and ways that all these tools were synthesized together to collect data.

Research Design

This design method of the research combines various techniques to seek answers to "What's" and "How's" (Team Leverage Edu, 2021). In this case:

i. "**What** are fingerprint matching algorithms?"

ii. "**How** do these fingerprint matching algorithms work?"

iii. "**How** are they implemented?"

iv. "**Which** algorithm is more accurate?"

v. "**What** are the times each algorithm takes to finish?"

From these questions, the results collected were analyzed to find patterns and make interpretations.

Data/Information Collection

To understand the science behind fingerprint identification and matching, various papers were consulted and referenced on the two methods chosen. Implementation of the two algorithms varied and as such multiple sources were combined to obtain a general implementation of both algorithms.

Dataset

For this research, the Sokoto Coventry Fingerprint Dataset (SOCOFing) was used. This dataset is a biometric fingerprint database designed for academic purposes. It consists of over six thousand (6,000) fingerprint images from over six hundred (600) subjects with unique attributes such as labels for gender, finger and hand name. Also included with the dataset, are three different levels of altercations for obliteration, z-cuts and central rotations (Ruizagara, 2018). A summary of the nature of the dataset is given below:

Table 1: Nature of Dataset used

Fingerprint Sets	**Real**	**Altered - Easy**	**Altered - Medium**	**Altered – Hard**
Dimension	96 x 103	96 x 103	96 x 103	96 x 103
Image Type	Bitmap (.BMP)	Bitmap (.BMP)	Bitmap (.BMP)	Bitmap (.BMP)
Image Size (KB)	38.7	10.7	10.7	10.7
Number of Images	6,000	17,931	17,067	14,272

Each level of altercation further had three levels of obstruction

Table 2: Levels of Obstruction

Altered – Easy	**Altered – Medium**	**Altered – Hard**
CR (Cross Region Cut)	CR (Cross Region Cut)	CR (Cross Region Cut)
OBL (Obliteration)	OBL (Obliteration)	OBL (Obliteration)
Z-Cut	Z-Cut	Z-Cut

For this study, 900 images of fingerprints were used.

Algorithm Implementation

To realize these two algorithms, the Python Programming language was used accompanied with various libraries.

Python Programming Language

Python version **3.9.13 MSC v.1929 64bit (AMD64)** was used throughout the implementation of both algorithms. This version was chosen for its stability, ease of use as well as the volume of computer vision and mathematical computation libraries supported by major backers. The libraries which were used include the following

i. NumPy
ii. OpenCV2
iii. Matplotlib
iv. PyQt5

NumPy

NumPy is a library for scientific computing in Python, it provides a variety of routines for fast operations on arrays including mathematical, logical, selecting, basic linear algebra, I/O as well as discrete Fourier Transforms (NumPy Developers., 2022). The version being used for this project is ***version 1.23.1***

OpenCV2

OpenCV2 is a Python library started by intel's Gary Bradsky in 1999, it a set of Python bindings designed to solve computer vision problems (OpenCV, 2022). It is originally a C++ library that uses Python Wrappers to create modules, making code run as fast as its original implementation in C++. ***Version 4.6.0.66*** is used for this project.

Matplotlib

Matplotlib is a comprehensive library for creating static, animated and interactive visualizations in Python. Matplotlib allows a diagrammatic representation of all unique points on a fingerprint, as well as its corresponding matching points on the sample image (Hunter, 2007).

PyQt5

PyQt5 is a Graphical User Interface (GUI) framework that wraps around the C++ library, Qt. It allows the construction of GUI that hide the abstraction of code (pythonpyqt, 2020). Using PyQt5, an algorithm comparison program has been built which allows users to load images and run both comparisons, results are displayed side by side in tabs in such a way that the various phases of each algorithm can be analyzed directly.

Fingerprint Matching

Each image was processed with both algorithms two times to reduce false acceptancy rate (FAR) and false rejection rate (FRR) and compared to its three (3) respective altered versions with their matching scores. Graphs were drawn to visually represent the match score between the two algorithms

Algorithm Completion Time

Both algorithms were tested on a computer system with the following specifications

Operating System	Windows 10 Pro 64-bit (10.0, Build 19044)
Processor	Intel(R) Core™ i7-10870H CPU @ 2.20Ghz (16 CPUs)
Memory	16384MB RAM

The time taken to complete both algorithms on the same set of data was recorded and represented in graphs

Validity of Data

To ensure that consistent results were obtained, the following procedures were put in place:

- All algorithms were run on the same set of images in succession
- The same versions of the Python Programming Language with its various libraries were used on both computer systems
- Both algorithms were run using the same configurations
- All match scores were taken on the first instance of running the algorithm to simulate a real-world experience

Limitations

- Both algorithms could not be fully optimized due to time limitation (hence each algorithm may take a longer time to compute on slower computers)
- The latest version of the Python Programming Language was not available

Chapter 4: Results and Discussions

The results of running both algorithms on the 900 fingerprints were visualized using a GUI (figure 11). The results were given in an Excel sheet and grouped into two (2) sections namely

i. SIFT RESULTS

ii. MINUTIAE RESULTS

Each of these sections had the following information

i. Fingerprint Image

ii. Match Score

iii. Time Taken

iv. Verdict

Fingerprint Image

This section shows the original fingerprint image paired against its altered version. Altered versions were further broken down into sub categories, namely

- CR (Cross Region Cut)
- OBL (Obliteration)
- Z-Cut

Time Taken

Time Taken refers to the time for total completion of the algorithm; that is, from initialization to match score generation.

Match Score

Match score refers to the number of matches that were produced when images in the sub altercation category were compared to the original image.

Verdict

Verdict refers to the conclusion generated by each algorithm after it had completed its processing on both images. Each algorithm had different criteria for drawing a conclusion.

Scale Invariant Feature Transformation (SIFT)

- If the number of matches were greater than 35 (> 35) the fingerprint images were considered a good a match.
- If the number of matches were greater than 18 but less than 35 the fingerprint images were considered a good match but with a low score.
- If the number of matches were less than 18 then the conclusion is that fingerprints do not match

Table 3 summarizes the relationship between verdict and match score.

Table 3: Relationship between Verdict and SIFT Match Score

Match Score	Verdict
Greater than 35	Good Match
Greater than 18 but less than 35	Match but with low score
Less than 18	No Match

Minutiae Matching Algorithm

- If the match score was greater than 7 the fingerprint images were considered to be a good match
- If the match score was greater than 3 but less than 7 the images were considered match with low score.

- If the match score was less than 3 then the fingerprint images were not considered to be a match

Table 4 summarizes the relationship between the match score and verdict.

Table 4: Relationship between Verdict and Minutiae Match score

Match Score	**Verdict**
Greater than 7	Good Match
Greater than 3 but less than 7	Match but with low score
Less than 3	No Match

Fingerprint image	Alteration Type	Match Score (SIFT)	Time (SIFT)	Verdict (SIFT)	Match Score (Minutiae)	Time (Minutiae)	Verdict (Minutiae)
1__M_Left_index_finger.BMP 1__M_Left_index_finger_CR.BMP	Easy	40	6.85421E-05	Fingerprints Are A Good Match!	1	1.23717E-05	Fingerprints do not match
1__M_Left_index_finger.BMP 1__M_Left_index_finger_Obl.BMP	Easy	44	6.68273E-05	Fingerprints Are A Good Match!	1	1.23392E-05	Fingerprints do not match
1__M_Left_index_finger.BMP 1__M_Left_index_finger_Zcut.BMP	Easy	46	6.52769E-05	Fingerprints Are A Good Match!	8	1.24693E-05	Fingerprints Are A Good Match
1__M_Left_little_finger.BMP 1__M_Left_little_finger_CR.BMP	Easy	31	6.58313E-05	Fingerprints Match With A Low Score!	0	1.23379E-05	Fingerprints do not match
1__M_Left_little_finger.BMP 1__M_Left_little_finger_Obl.BMP	Easy	35	6.8484E-05	Fingerprints Match With A Low Score!	0	1.24996E-05	Fingerprints do not match
1__M_Left_little_finger.BMP 1__M_Left_little_finger_Obl.BMP	Medium	27	6.77754E-05	Fingerprints Match With A Low Score!	0	1.2379E-05	Fingerprints do not match
1__M_Left_little_finger.BMP 1__M_Left_little_finger_Zcut.BMP	Medium	30	6.11174E-05	Fingerprints Match With A Low Score!	0	1.22169E-05	Fingerprints do not match
1__M_Left_middle_finger.BMP 1__M_Left_middle_finger_CR.BMP	Medium	20	5.32277E-05	Fingerprints Match With A Low Score!	0	1.33997E-05	Fingerprints do not match
1__M_Left_middle_finger.BMP 1__M_Left_middle_finger_Obl.BMP	Medium	27	5.79226E-05	Fingerprints Match With A Low Score!	3	1.33751E-05	Fingerprints Match With A Really Low Score
1__M_Left_middle_finger.BMP 1__M_Left_middle_finger_Zcut.BMP	Medium	26	5.35037E-05	Fingerprints Match With A Low Score!	0	1.35674E-05	Fingerprints do not match

Figure 11: Sample Results as shown on a GUI

Matches

After both algorithms run on the same data sample, the results of matches generated were collated and plotted on a graph (figure 13):

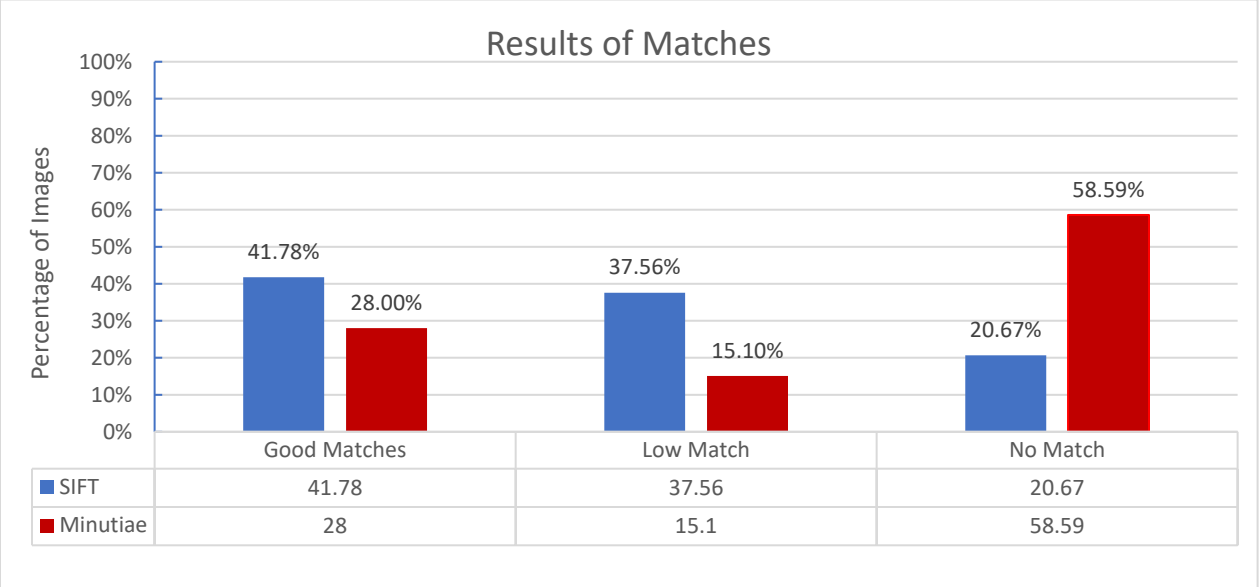

Figure 12: Results of Matches

SIFT has higher good matches than Minutiae by about 50% (figure 12). This means that using SIFT, 50% more good matches of fingerprints will be identified.

Minutiae has a lower percentage of low matches by about 60% (figure 12) implying that more fingerprints will not be matched using Minutiae than SIFT if the alteration is more pronounced.

Minutiae has a higher rejection rate of fingerprints than SIFT by about 65% which means using Minutiae will fail more fingerprints than SIFT especially if the fingerprints are altered heavily.

Algorithm Run Time

Cross Regional (CR) Images

The time taken to process Cross Regional (CR) images on both algorithms were plotted (figure 13) Trendlines were determined.

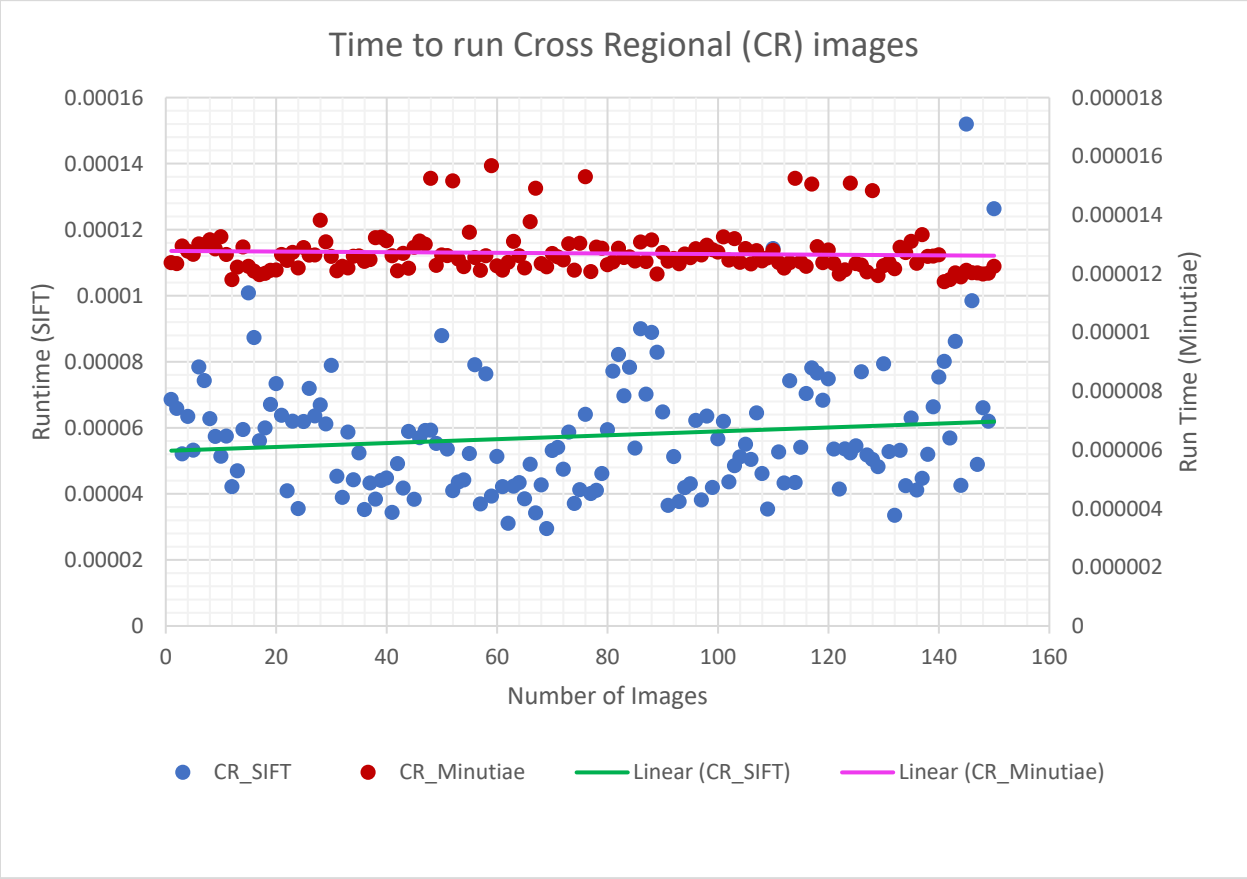

Figure 13: Time to run Cross Regional (CR) images

The average time to process a fingerprint image with a cross regional (CR) cut using the SIFT algorithm increases as the number of images increase (figure 13).

The trendline equation is ***y = 6E-08x + 5E-05***

The time to process a fingerprint image with a cross regional (CR) cut on average using the Minutiae algorithm reduces as the number of images increase (figure 13).

The trendline equation is ***y = -1E-09x + 1E-05***

Using the equations of the trendlines, the runtime for Cross Regional (CR) images for various populations were calculated and tabulated (table 5).

Table 5: Runtime for Cross Regional (CR) images

Runtime CR		
Number of Subjects	**SIFT**	**Minutiae**
500	0.0000799999999999999	0.0000095
5,000	0.00035	0.000005
50,000	0.0030499999999999998	0.00004

Obliterated (OBL) Images

The time taken to process Obliterated (OBL) images on both algorithms were plotted (figure 14). Trendlines were determined.

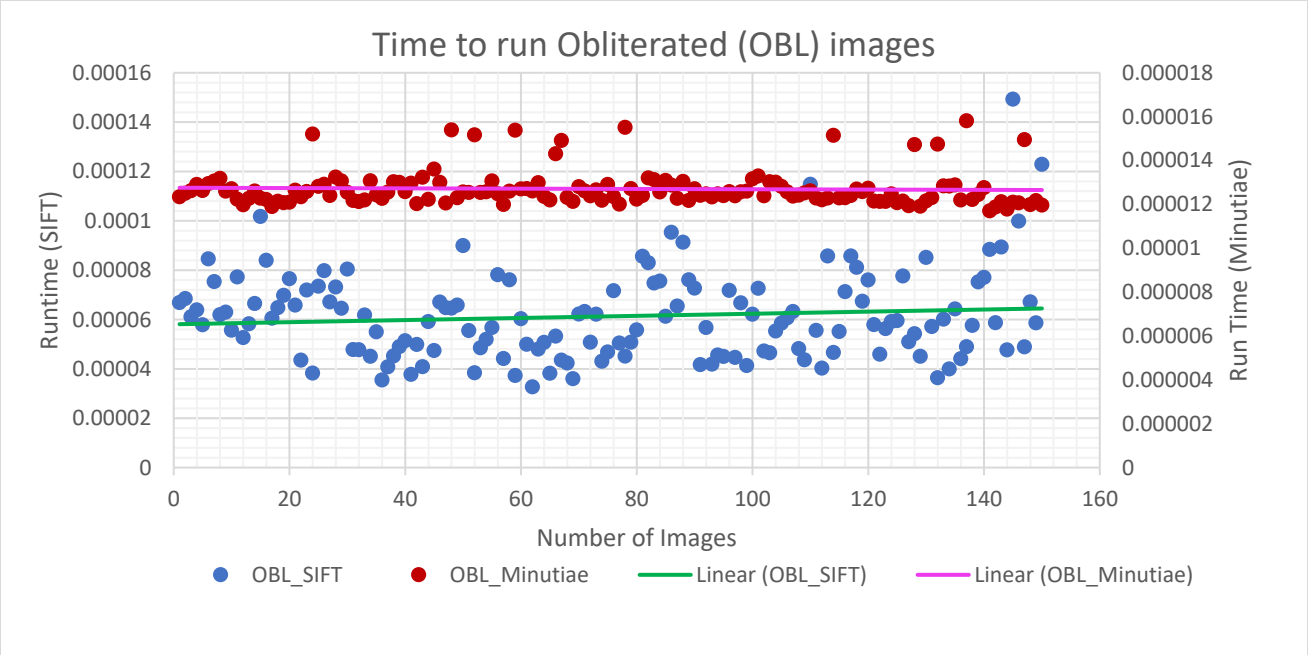

Figure 14: Runtime for OBL images

The average time to process a fingerprint image using the SIFT algorithm increases as the number of images increase (figure 14). The trendline equation is ***y = 4E-08x + 6E-05***

The time to process a fingerprint image on average using the Minutiae algorithm reduces as the number of images increase (figure 14). The trendline equation is ***y = -7E-10x + 1E-05***

Using the equations of the trendlines, the runtime for Obliterated images (OBL) for various populations were calculated and tabulated (table 6)

Table 6: Runtime for Obliterated (OBL) images

Runtime OBL		
Number of Subjects	**SIFT**	**Minutiae**
500	0.00008	0.00000965
5,000	0.00026000000000000003	0.0000065

Runtime OBL		
Number of Subjects	**SIFT**	**Minutiae**
50,000	0.00206	0.0000249999999999999

Z Cut Images

The time taken to process Z Cut images on both algorithms were plotted (figure 15) and trendlines determined.

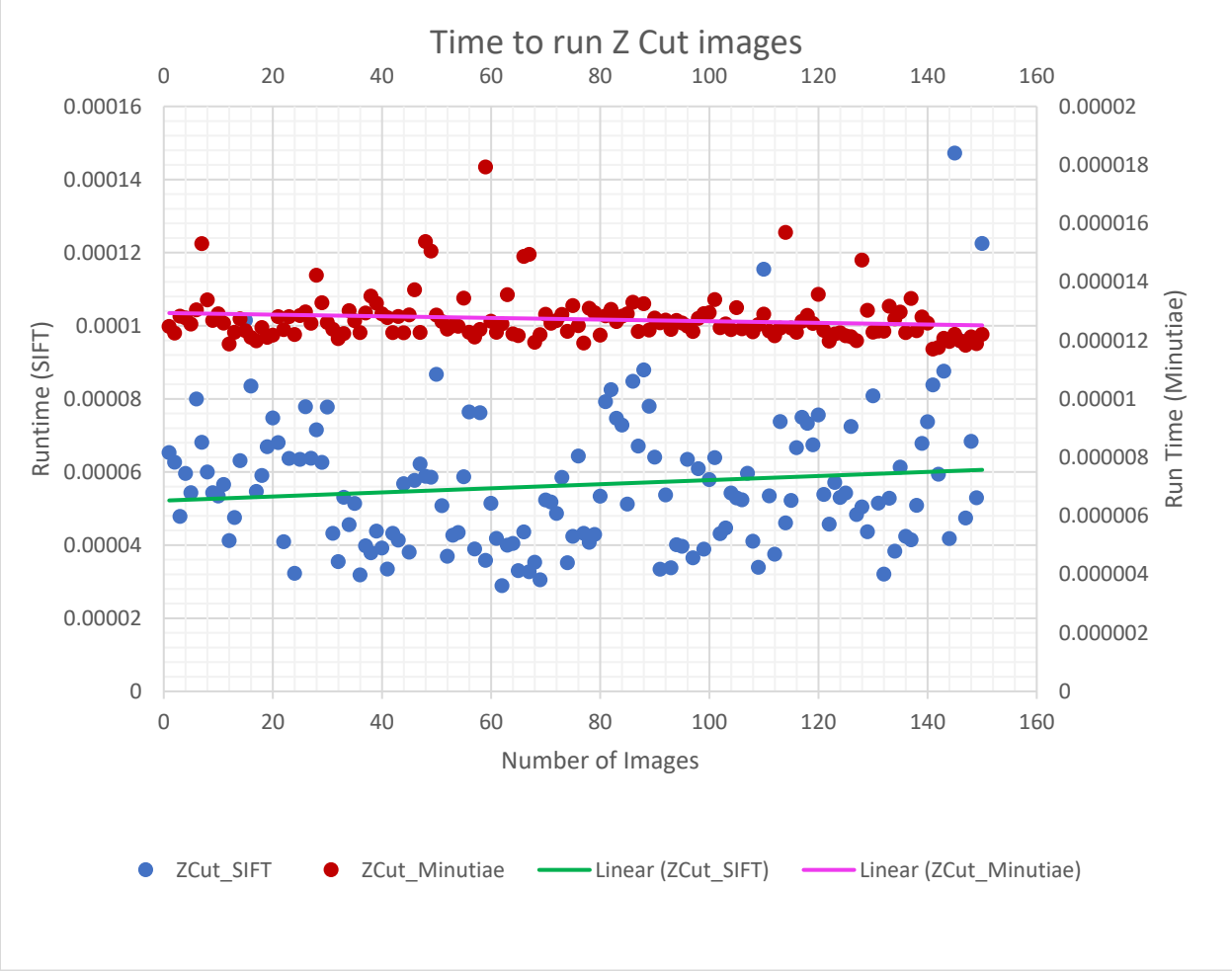

Figure 15: Runtime for Z-Cut images

The average time to process a fingerprint image using the SIFT algorithm increases as the number of images increase (figure 15). The trendline equation is ***y = 6E-08x + 5E-05***

The time to process a fingerprint image on average using the Minutiae algorithm reduces as the number of images increase (figure 15). The trendline equation is ***y = -3E-09x + 1E-05***

Using the equations of the trendlines, the runtime for Z Cut images for various populations were calculated and tabulated (table 7)

Table 7: Runtime for Z Cut images

Runtime Z-Cut		
Number of Subjects	**SIFT**	**Minutiae**
500	0.0000799999999999999	0.0000085
5,000	0.00035	0.00000499999
50,000	0.0030499999999999998	0.00014

Chapter 5: Conclusion

The two algorithms, Scale Invariant Feature Transformation (SIFT) and Minutiae-based algorithm can be implemented using the Python Programming language.

Both algorithms are able to extract unique features on sample data. These unique features can be used to match one fingerprint to another.

The processes can be visualized in a Graphical User Interface using the PyQt5 library and their performance including runtime collected.

Scale Invariant Feature Transformation (SIFT) is more likely to accurately identify subjects with (some sort of distortion on their fingerprints) such as cuts, dust or skin-oil on the fingerprint or on the scanner. Minutiae is less likely to identify subjects with distortions or aberrations on their fingerprints.

Minutiae based algorithm will run faster on larger datasets as compared to SIFT, however its accuracy is reduced.

Both algorithms have good performance, running under 3 milli seconds on a database containing 500,000 sample fingerprint images.

Recommendations and Future Work

- Both algorithms should be reimplemented in the latest version of python and the accompanying libraries as they can offer significant improvement to runtime
- Larger data sets should be used to validate the efficiency of each algorithm as the size of a dataset increases

- The criteria for the match scores should be varied to investigate/confirm the efficiency of the algorithms

References

Al-Ani, M. S. (2013, February 2). A Novel Thinning Algorithm for Fingerprint Recognition. *International Journal of Engineering Sciences*, 43-48.

Al-Najjar, Y., & Sheta, A. (2008). MINUTIAE EXTRACTION FOR FINGERPRINT RECOGNITION. *2008 5th International Multi-Conference on Systems, Signals and Devices* (pp. 1-5). As-Salt: INPROCEEDINGS.

Amina, Dominique, M., & Youcef, Z. (2022). Fast Fourier transform for fingerprint enhancement and features extraction with adaptive block size using orientation fields. *ITM Web of Conferences*, 10.

Cao, L., & Wang, Y. (2016, May 31). Fingerprint Image Enhancement and Minutiae Extraction Algorithm. Linnaeus, Li, Sweden.

Davidson, M. W. (2016, May 17). *Difference of Gaussians Edge Enhancement.* Retrieved from Micro.Magnet: https://micro.magnet.fsu.edu/primer/java/digitalimaging/processing/diffgaussians/index.html

Edmund. (2016). *SIFT - Scale-Invariant Feature Transform.* Retrieved from Weitz: http://weitz.de/sift/

Erwin, Karo, N. N., Sari, A. Y., Aziza, N., & Putra, H. K. (2019). The Enhancement of Fingerprint Images using. *Journal of Physics: Conference Series*.

Fisher, R., Perkins, S., Walker, A., & Wolfart, E. (2003). *Contrast Stretching.* Retrieved from HIPR2: https://homepages.inf.ed.ac.uk/rbf/HIPR2/stretch.htm

Haddad, R. A., & Akansu, A. N. (1991). A Class of Fast Gaussian Binomial Filters for Speech and Image Processing. *Transactions on Signal Processing*, 723-727.

Hunter, J. D. (2007). Matplotlib: A 2D graphics environment. *Computing in Science & Engineering, 9,* 90-95. doi:10.1109/MCSE.2007.55

Maddala, S., & Tangellapally, S. R. (2010, September). Implementation and Evaluation of NIST Biometric Image Software For Fingerprint Recognition. Blekinge, Sweden.

NumPy Developers. (2022). *What is NumPy?* Retrieved from NumPy: https://numpy.org/doc/stable/user/whatisnumpy.html

OpenCV. (2022, June 5). *Introduction to OpenCV-Python Tutorials*. Retrieved from OpenCV: https://docs.opencv.org/4.6.0/d0/de3/tutorial_py_intro.html

Otero, R., Delbracio, I., & Mauricio. (2014). Anatomy of the SIFT Method. *Image Processing On Line*, 370--396.

Park, U., Pankanti, S., & Jain, A. (2008). Fingerprint Verification Using SIFT Features . *SPIE Defense and Security Symposium*.

pythonpyqt. (2020). *What is PyQt?* Retrieved from LEARN PYTHON PYQT: https://pythonpyqt.com/what-is-pyqt/

Ruizagara. (2018). *Sokoto Coventry Fingerprint Dataset (SOCOFing)*. Retrieved from Kaggle: https://www.kaggle.com/datasets/ruizgara/socofing

Shah, A. (2018, June 17). *Through The Eyes of Gabor Filter*. Retrieved from Medium: https://medium.com/@anuj_shah/through-the-eyes-of-gabor-filter-17d1fdb3ac97

Singh, A. (2019, October 9). *A Detailed Guide to the Powerful SIFT Technique for Image Matching*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2019/10/detailed-guide-powerful-sift-technique-image-matching-

python/#:~:text=SIFT%20helps%20locate%20the%20local,detection%2C%20scene%20detection%2C%20etc.

Team Leverage Edu. (2021, November 17). *Research Desi*. Retrieved from Leverageedu: https://leverageedu.com/blog/research-design/

Tyagi, D. (2019, March 16). *Introduction to SIFT(Scale Invariant Feature Transform).* Retrieved from Medium: https://medium.com/data-breach/introduction-to-sift-scale-invariant-feature-transform-65d7f3a72d40

Wang, C., Gavrilova, M., Luo, Y., & Rokne, J. (2006, January). *An efficient algorithm for fingerprint matching.* Retrieved from ResearchGate: https://www.researchgate.net/publication/220927699_An_efficient_algorithm_for_fingerprint_matching/citations

Wechsler, H., & Li, F. (2014). Chapter 10 - Biometrics and Robust Face Recognition. In M. Kaufmann, *Conformal Prediction for Reliable Machine Learning* (pp. 189-215). 189-215: Kaufmann, Morgan.

Zaeri, N. (2011). *Minutiae-based Fingerprint Extraction and Recognition.* Rijeka: IntechOpen. doi:10.5772/17527