Analysis for fingerprinting fusion algorithms

Alonso Bravo, Alvaro Lopez, Adrian Crisp

Abstract
One of the issues when looking into developing fingerprinting is finding the best algorithm or mix of algorithms to extract and match features in fingerprints. This is known as fusion algorithms, this is where our project aims to solve and answer the benefits and ways of improving singular feature extraction matching algorithms by pairing the two feature extractors together to get a deeper understanding of how fingerprinting algorithms work and where they are able to show their strengths as well as weaknesses. Firstly we review existing algorithms and approaches to implement our singleton algorithms for local, global, and ridge based feature extractors. After that we then research and find algorithms for fusion matching algorithms that utilize pairs of these algorithms to have as a comparison point. After researching we start implementation for image pre-processing which will be consistent for all our matching algorithms as a base line and our findings will use the FCV 2000 DB, specifically the 2000 DB3 database for fingerprints, which was mainly used for testing.

Keywords: Fusion, Minutiae, Performance

1. Introduction

Fingerprints and fingerprinting as a whole have a distinct and noticeable issue that occurs when fingerprints from the same individual can have some drastic differences in features extracted by algorithms due to obstructions like noise, orientation, smudges, lightness and darkness in images etc.. This causes the issue that a fingerprint given by the same individual should match up even if the given fingerprint is given at a later time than what the initial fingerprint was taken at. In this paper we look to find, which sets of algorithms work best to extract and match the extracted fingerprint data best to show, which algorithms excel at matching and those algorithms that can struggle when not all fingerprints are given in perfect conditions.

2. Pre-Processing

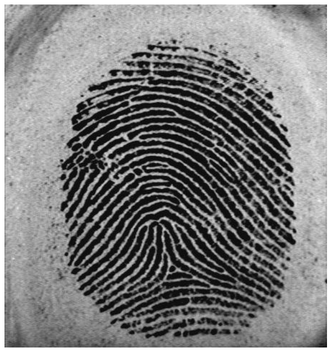

Fig.1. FCV2000 Database 3 Image 1 Person 1 used in primary testing.

For the fingerprints that were used in this study in the dataset FCV2000 Database (1-4) many had very clear issues as mentioned seen throughout the images[4]. The data contained 10 images per-person per dataset across 8 different individuals where each data set contained 80 total images(Insert Reference Num). The first step to feature extraction was to convert the images into grayscale, which was done using OpenCV Sobel Operator which took the X-axis and Y-axis greyscale images and recombined them into the gray scale image of the gradient image. The next step was to have Normalization occur, which helps gauge the intensity of the image and helps clear any noise that can be seen through the image. This is a crucial step in pre-processing the image to make contrasting the ridges and valleys easier for future steps. After comes orientation, which for this algorithm is a basic overlayment of the images we used here to match up overlapping ridge line angles given in the images. Post orientation we get into thinning to more clearly define the ridges and valleys for proper feature extraction where we divided the image into blocks and iteratively checked sections to remove and smudge the images.

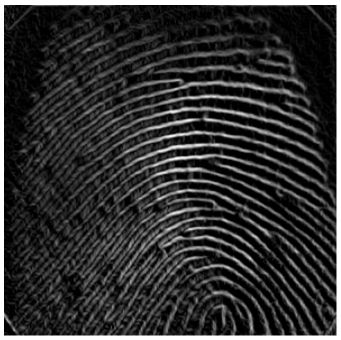

Fig.2. FCV2000 Database 1 Image 1 Person 1 where image is the resulting gradient image from X &Y sobel operator.

2.1 Singularity Detection

The Poincare singularity detection algorithm is used to detect singularity points in fingerprints, mainly the loop, delta and whorl, which can be defined as 180%, -180%, and 360% respectively. This process is done by an orientation field estimate being computed on the fingerprint essentially turning the ridge points into angle values . From there the Poincare index can be calculated by the following equation, $\text{Poincare}_{G,C}(i,j) = \Sigma_{k(0-7)}\text{angle}(d_{k,dk+1)}\%180)$.
In doing so by taking a 3x3 area in the fingerprint the angles of the 8 surrounding ridgelines are computed to see if they match and predefined classes if so they are marked and used for ridge-based feature extraction.

3. Singleton Feature extraction

For this section the three main feature extraction algorithms implementations will be overviewed in Ridge, Local, and Global. As an overview for these extraction methods this section will discuss the general idea behind these extraction methods. Ridge-based focuses on matching the ridges in fingerprints and seeing how much ridge overlap occurs. Local focuses on extracting and matching minutiae points locally, usually implementing algorithms that involve neighbor matching, then calculating a similarity score. Global focuses on finding and matching minutiae on a global scale, the implementation of which can vary.

3.1 Ridge-Based Feature extraction

The first step for Ridge-Based Feature extraction is to apply edge detection to the fingerprint, where we used the Hough Transform to detect the lines of the fingerprint. Note: fingerprint is aligned via poincare indexing. Feature extraction is accomplished by taking lines in the fingerprint as polar coordinates $\rho = x\cos\theta + y\sin\theta$, where ρ is the perpendicular distance from the origin to the line, and θ is the angle of this perpendicular. These polar coordinates are saved in an accumulator array transforming the image into coordinates where there is the quantized pair (ρ,θ). From here the edge pixels calculate which has the most lines passing through it to classify it as a line in the image with a given orientation from 0-180% based on the given fingerprint.

3.1.2 Ridge-Based Matching

Given the extracted data of the fingerprint the corresponding polar coordinates are matched given their pairs of (ρ,θ) and estimates a set of rotation and translation parameters $(\Delta\theta,\Delta x,\Delta y)$ that best align the query fingerprint image with the template fingerprint image.For each set of these parameters, the query image is aligned with the template. A matching score is then calculated for each alignment[1]. The final matching score for the fingerprint comparison is the highest score obtained across all sets of transformation parameters.
The matrix C is not symmetric, reflecting the fact that the alignment of ridge i with ridge j can differ from the alignment of ridge

j with ridge i. For an exact match (genuine fingerprints), C becomes a diagonal matrix where each diagonal element corresponds to the number of pixels in the respective ridge.
In cases of genuine matching, high-valued elements in C are expected to be located near the main diagonal. In contrast to this, impostor matches would show low-valued elements spread throughout the matrix. To further separate genuine and imposter samples the matching penalizes samples where ridge-crossings have increased C non-zero values(imposter).

3.2 Local Feature extraction

For our local feature extraction we used OpneCV's Scale-Invariant Feature Transform (SIFT) feature descriptor. The goal of sift is to find key points in the image. These key points tend to be areas that have high contrast. The four main steps SIFT takes are:

1. Scale-space Extrema Detection: Potential keypoints are identified through using the difference of gaussians (DoG) to approximate the Laplacian of Gaussian (LoG). This helps detect the minutiae at various scales.
2. Keypoint Localization: After the detection of potential keypoints, low contrast points are eliminated in order to preserve only the highest quality keypoints. This refinement process often includes an expansion of a Taylor series of the scale-space funcion. A Hessian matrix is also used to discard points that are poorly localized along edges.
3. Orientation Assignment: An orientation is assigned for each keypoint. This is based on the local gradient directions of the fingerprint image. This is important for matching fingerprints that might be orientated differently.
4. Keypoint Descriptor Creation: Each keypoint is given a descriptor. This descriptor is a 10x16 neighborhood around the keypoint. This neighborhood is divided into smaller 4x4 sub-blocks. Each sub block contains an 8-bin orientation histogram. The concatenation of the histograms forms a descriptor vector for each key point. This captures the local structure around the keypoint.

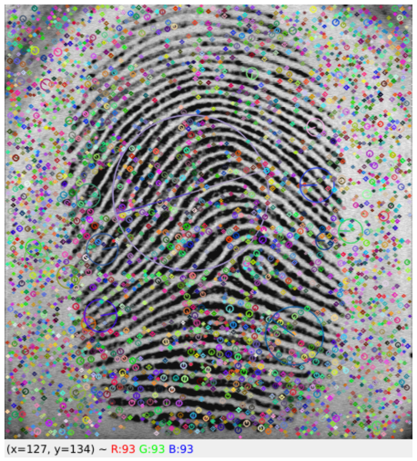

Fig.3 . FVC2000 DB3 image. SIFT descriptors visualized.

3.2.2 Local Matching

We used OpenCV's Fast Library for Approximate Nearest Neighbors (FLANN) based feature matcher to match the SIFT Descriptors. The general steps for FLANN are:

1. Building the FLANN Index: FLANN builds an index of feature descriptors, this allows for efficient nearest neighbor searches. This index is essentially a data structure designed to speed up the search of the nearest descriptors. FLANN offers many algorithms depending on the feature extractor, but in our case we used the KD-Tree algorithm.
2. Searching for Nearest Neighbors: FLANN takes each feature descriptor from one image and finds the closest

descriptor in the other image. It finds the neighbors that are close to being the nearest.

3. Matching the Features: The output is a set of pairs of matching features. A pair consists of a feature from the first image and the closest feature in the second image.
4. We then apply the Lowe's Ratio Test to the feature set. This helps eliminate and filter out the bad matches. Matches are kept only if the ratio of the distance of the nearest neighbor and the distance of the second nearest neighbor is under a certain threshold.

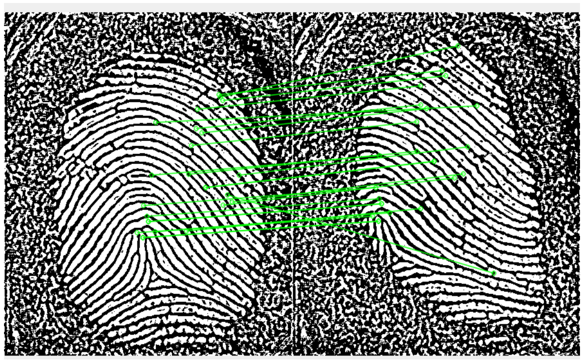
Fig.4. FCV2000 Database 3 Image 1 and 2 from person 1. Local Feature matching is done (green lines to correlating minutiae) Note: Matching was done on pre-processed image.

3.3 Global Feature extraction

Global Feature Extraction was done using Minutia Cylinder Code(MCC), which has a Local Structure Representation where each minutia is represented as m = $\{x_m, y_m, \theta_m\}$, where x_m, y_m are the coordinates and θ_m is the direction. The local structure is a cylinder centered on the minutia's location and oriented according to the minutia's direction. The cylinder is enclosed in a cuboid, divided into cells[2]. Each cell is identified by indices (i,j,k). Cell Value calculation is done for each cell, $C_m(i,j,k)$ using $\Sigma_{mt\in N_{pmij}} C^S_M(m_t, pm_{ij}) \text{x}\, C^D_M(m_t, d'_k)$ where C^S_M and C^D_M represent the spatial and directional contributions from each neighboring minutia m_t. The next step is to get the spatial and directional contributions where each neighboring minutia contributes spatially and directionally to the cell value. The spatial contribution is a function of the Euclidean distance between the minutia and the cell's center, often modeled using a Gaussian function. The directional contribution is based on the angular difference between the cell's associated angle and the direction of the neighboring minutia. After we get the cylinder validity where a cylinder is considered valid if it has a sufficient number of valid cells and contributing minutiae. The validity of a cell depends on whether its center lies within the intersection of the cylinder's base and an enlarged convex hull of all minutiae. For efficient computation a bit-based implementation of this representation is used. Here, the values in the cells are constrained to binary values, allowing the use of bitwise operations for matching and similarity calculations.

3.3.2 Global Matching

To match the extracted MCC's firstly the local similarity computation is done, which involves calculating the local similarity between cylinders. A matrix is then created containing all local similarities.From there a local similarity sort is done which sorts all local similarities and selects the top similarities to compute the global score. This is calculated as local similarity between minutia a in A and b in B is denoted as (a,b)The matrix S contains all these local similarities, where $S[r,c]=(a_r,b_c)$.The local similarities are sorted, and the top n_p minutiae-index pairs are selected, where n_p is a parameter that partially depends on the number

of minutiae in the two templates. These pairs are represented as
P = {($r_{t,}$$C_t$)|t = 1,...,$n_p$}, where 1≤$r_t$≤$n_A$ and ≤C_t≤$n_{B.}$ The global score S(A,B) is calculated as the average of the corresponding local similarities:

$$S(A,B) = \frac{1}{np} \sum_{(r,c)\in P} S[r,c]$$

The value of n_p= min(n_p+[Z(min(n_A,n_B);P;P)X(max_{np}-min_{np})], Here, P and P are parameters, Z is the sigmoid function, max_{np} and min_{np} are parameters, and [x] denotes the floor function (rounding down).

4. Fused Feature Matching

Fused feature matching involves combining the results of multiple individual matching algorithms to improve the overall accuracy and robustness of the fingerprint recognition system. Some algorithms focus more on the ridges, bifurcations and terminations; some others may focus on correlation or histograms. This to say there is a benefit to using certain algorithms when trying to extract certain features from a fingerprint, therefore combining these techniques will create a more robust and accurate matching algorithm, rather than a singular matching algorithm. These combinations can help to strengthen the weaknesses that can be found when matching certain extracted features.

4.1 Ridge/Local Matching

The fusion matching algorithm implemented for Ridge/Local Matching was to stack the extracted features of the individual algorithms together and generate a result. To do this we first extracted both feature sets for an image, then combined the overall similarity score that was given by the matching algorithm. More weight was given once the matching algorithm was able to determine how many total ridges and minutiae were able to be extracted for the image.

4.2 Local/Global Matching

The fusion algorithm for implementing a local global matching algorithm is based upon using weights influenced by the local feature extraction algorithm. To achieve this the local features extracted were passed to the global matcher with positions P(X,Y,θ) and these are into the global matcher where they are added to the MCC and calculated. Then if there is a local proximity that is within a certain distance d between the neighboring minutiae points a weight is added to the global minutiae point and that is used to calculate the final similarity score in the same way that the mcc was originally calculated. Another weight is also added if the minutiae matched were found within the fingerprints convex hull (area with most minutiae extracted).

5 Results

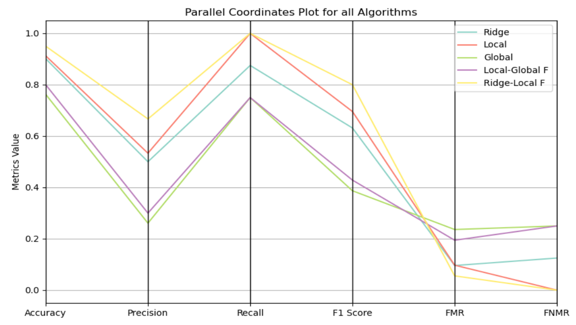

Fig.5. Line graph showcasing the resulting matching results for all matching algorithms (singleton & fused).

In this section we review the results and accuracy of each implementation of feature extractors and matching schemes.

5.1 Singleton Results

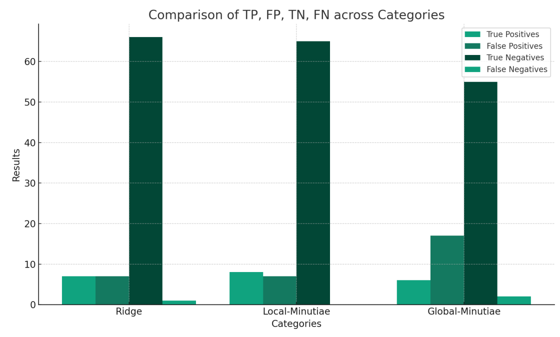

Fig.6. Bar graph showing the results from testing across 8 people in FCV 2000 DB3 for TP, TN, FP, FN in the singleton feature matchers.

The performance of the singleton algorithm aligned closely with anticipated outcomes for a singleton matching system. This was particularly evident in the proficiency of the local minutiae matcher throughout the dataset, which outperformed the global minutiae matching algorithm. The global matcher requires more meticulous thresholding and calibration to accurately match minutiae points in comparison. The ridge-based methodology encountered difficulties with fingerprints featuring numerous ridge-crossings that did not fully align with the original or comparative image. On the other hand, the local algorithm demonstrated superior effectiveness in distinct matches where the bifurcations and terminations in the pre-processed images were readily apparent. This result is in line with the expected capabilities of the algorithm.

5.2 Fused Results

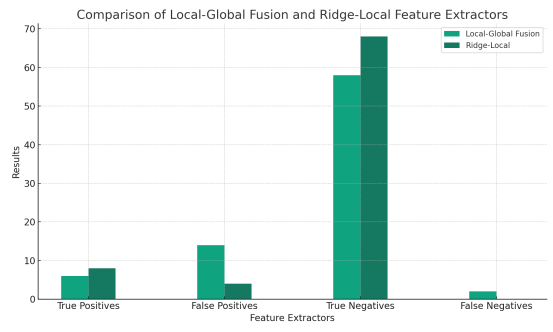

Fig.7. Bar graph showing the results from testing across 8 people in FCV 2000 DB3 for TP, TN, FP, FN in the fused feature matchers.

This bar graph displays the outcomes of FCV 2000 DB3 testing on 8 individuals for True Positive (TP), False Positive (FP), True Negative (TN), and False Negative (FN) using fused feature matchers. Two feature extractors were employed, incorporating Local-Global Fusion and Ridge-Local techniques. For Local-Global Fusion, a novel stacking method was implemented, involving minutiae extraction followed by local matching. Weights were assigned to influence minutiae matches, considering both matched minutiae and the convex hull surrounding centralized points. This approach proved effective in mitigating background noise. Ridge-Local Fusion Matching involved separately running local and ridge extraction on the same fingerprint, with features added to a profile. A similar process was applied to the comparison fingerprint, and subsequent data comparison determined fingerprint matches. The findings highlight the superiority of Ridge-Local fusion in minimizing false positives and improving overall accuracy in fingerprint matching, offering a valuable contribution to biometric recognition systems.

5.3 Conclusion

The results shown by this experiment were about what we were expecting to see form the following algorithms as the ridge/local and local having the higher accuracy ratings as opposed to the other matching algorithms fig.5. However one unseen result was the overall high FP rate given from the global algorithm in both the singleton and fusion matchers. This is likely due to the implementation relating to the neighbor matching algorithm and the proper thresholding to match accordingly. Overall however the project was successful in showing the effects fused algorithms can have on improving robustness and efficiency in fingerprint matching algorithms.

5.4 Future Work

Some future work that could be done following this paper is to implement the same matching algorithms using different feature extraction methods (i.e. surf, binary maps, frequency maps etc.). On the contrary, using different fusion matching techniques may also be beneficial as some algorithms may highlight the strengths and weaknesses of them in harmony, such that the strengths of one benefit the other and vice versa. Another important consideration is to get an EER of the dataset we are testing to see if there are certain images that are making all matchers struggle, as a baseline to help understand how efficient the matching algorithm truly is.

6. References

1. Marana, Aparecido, and Anil Jain. "Ridge-Based Fingerprint Matching Using Hough Transform." *Proceedings of the SIBGRAPI 2005.* 2005, doi: 10.1109/SIBGRAPI.2005.45.

2. Cappelli, Raffaele & Ferrara, Matteo & Maltoni, Davide. (2010). Minutia Cylinder-Code: A New Representation and Matching Technique for Fingerprint Recognition. IEEE transactions on pattern analysis and machine intelligence. 32. 2128-41. 10.1109/TPAMI.2010.52.

3. Alonso-Fernandez, Fernando, et al. "A Comparative Study of Fingerprint Image-Quality Estimation Methods." IEEE Transactions on Information Forensics and Security, vol. 2, no. 4, Dec. 2007, pp. 734–743, Institute of Electrical and Electronics Engineers (IEEE), doi:10.1109/tifs.2007.908228.

4.Mario, D, et al. “FCV 2000 Fingerprint Verification Competition.” *FVC2000*, 2000, bias.csr.unibo.it/fvc2000/.

5. OpenCV. (2023). OpenCV-Python Tutorials, https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html

6. Jain, A., & Ross, A. (2001). Fingerprint Matching Using Minutiae and Texture Features. IEEE Xplore. https://doi.org/10.1109/ICIP.2001.958106