Biometric System Using K-Nearest Neighbors and Harris Corner Detection

Abdulaziz Zyad Al-Said

Abstract

Bio-metric systems play a crucial role in authentication and identification, leveraging unique physical or behavioral traits for identity verification. This paper explores the implementation and intricacies of a biometric system based on a combination of K-Nearest Neighbors (KNN) for template matching and Harris Corner Detection for feature extraction. The system is trained on fingerprint images and evaluated on various scenarios, including noisy, altered, and rotated images.

I. Introduction

Biometric systems have gained prominence due to their ability to provide secure and convenient identity verification. This paper focuses on a biometric system designed for fingerprint recognition using the KNN algorithm and Harris Corner Detection.

II. Public Datasets

This system runs with two public data-sets for the training, and those are FVC (Fingerprint Verification Competition) 2004 and Sokoto Coventry Fingerprint Dataset (SOCOFing). The SOCOFing data-set comes with an altered set off the original with noise added from easy, medium, and hard levels of difficulty for recognition.

III. Preprocessing

The preprocessing module encompasses two main functionalities: adding noise and rotating images. Adding Gaussian noise simulates real-world variations in fingerprint images, while the rotation process enhances the system's robustness by considering different orientations of fingerprints.

A. Adding Noise

The function `add_noise` introduces Gaussian noise to an image, and `add_noise_to_folder` applies this to an entire image data-set. The formula used is:

$$\text{noisy}(x, y) = \text{clip}(\text{original}(x, y) + \text{Gauss}(x, y), 0, 255) \tag{1}$$

where $\text{Gauss}(x, y)$ is a random sample from a Gaussian distribution.

B. Rotating Images

The function `rotate_image` rotates an image by a specified angle using an affine transformation. This is achieved by applying a rotation matrix to each pixel. The formula is:

$$\text{rotated\_image}(x, y) = \text{original}(x', y')$$

where (x', y') is obtained by applying the rotation matrix to (x, y).

C. Image Preprocessing

The `preprocess_image` function converts images to gray-scale, normalizes pixel values, and resizes them. It ensures uniformity in image dimensions for further processing.

D. Minutiae Extraction

The `extract_minutiae` function employs Harris Corner Detection to identify potential minutiae points. Harris Corner Detection is a corner detection operator that identifies key points where the intensity of the image changes in multiple directions. The algorithm involves the computation of a Harris matrix for each pixel and the subsequent detection of corners based on eigenvalues of this matrix.

The key steps in Harris Corner Detection are as follows:

1) Compute gradients I_x and I_y using Sobel operators.
2) Compute products of gradients: $I_{xx} = I_x^2$, $I_{yy} = I_y^2$, and $I_{xy} = I_x \cdot I_y$.
3) Apply a Gaussian filter to the computed products.
4) Compute the Harris response: $R = \det(M) - k \cdot (\text{trace}(M))^2$, where M is the Harris matrix.
5) Threshold the response to identify corners.

The detected corners represent potential minutiae points.

IV. Minutiae Extraction and K-Nearest Neighbors (KNN)

The minutiae extraction process, as described in the previous sub-section, is crucial for identifying key points in fingerprint images. These minutiae points serve as distinctive features used by the K-Nearest Neighbors (KNN) algorithm for template matching.

A. K-Nearest Neighbors (KNN)

KNN is a supervised machine learning algorithm widely used for classification and regression tasks. In the context of bio-metric systems, KNN operates by comparing the minutiae features of an input fingerprint against stored templates in the database. The algorithm classifies the input fingerprint based on the class labels of its nearest neighbors in the feature space.

The key steps in the KNN algorithm for fingerprint recognition are as follows:

1) Feature Vector: Represent each fingerprint template as a feature vector, where each element corresponds to a minutiae point's characteristics.
2) Normalization: Normalize the feature vectors to ensure consistent scales for effective distance computation.
3) Distance Calculation: Measure the distance between the feature vector of the input fingerprint and those in the database, typically using Euclidean distance.
4) Nearest Neighbors: Identify the k-nearest neighbors with the smallest distances.
5) Majority Voting: Classify the input fingerprint based on the majority class of its nearest neighbors.

The KNN algorithm is implemented using the scikit-learn library, providing a flexible and efficient framework for fingerprint template matching.

B. Minutiae Extraction and KNN Integration

The minutiae extraction process, employing Harris Corner Detection, generates feature vectors used by the KNN algorithm. Each minutiae point's characteristics contribute to the feature vector, allowing the system to capture the unique patterns of individual fingerprints.

The combined approach of Harris Corner Detection for minutiae extraction and KNN for template matching enhances the system's accuracy and robustness. The experiment results, as discussed in the following sections, provide insights into the system's performance under various conditions, showcasing its effectiveness in real-world scenarios.

V. Matching and Evaluation

The matching and evaluation phase of the bio-metric system involves a detailed analysis of the `match_fingerprint` function, which plays a pivotal role in determining the accuracy and reliability of the system's fingerprint matching process.

A. Fingerprint Matching Process

The `match_fingerprint` function employs the K-Nearest Neighbors (KNN) algorithm to assess the similarity between a query fingerprint and the templates stored in the system's database. This process involves the following steps:

1) Feature Extraction: Extract relevant features from the query fingerprint using the minutiae information obtained through Harris Corner Detection. These features are essential for creating a representative feature vector that captures the unique characteristics of the fingerprint.
2) KNN Algorithm: Utilize the KNN algorithm to find the k-nearest neighbors in the feature space. The distance metric, often Euclidean distance, is computed between the feature vector of the query fingerprint and those of the templates in the database.
3) Confidence Thresholding: Introduce a confidence threshold to determine whether the query fingerprint is accepted as a match or rejected. The threshold serves as a decision boundary, allowing the system to balance between false positives and false negatives.
4) Matching Decision: Based on the distances calculated and the confidence threshold applied, the system makes a decision regarding the acceptance or rejection of the query fingerprint match.

```
function match_fingerprint(query_fingerprint, database_templates,
                                              confidence_threshold):
    features_query = extract_features(query_fingerprint)
    distances = calculate_distances(features_query, database_templates)
    min_distance = min(distances)

    if min_distance < confidence_threshold:
        return "Match"
    else:
        return "No Match"
```

B. Performance Metrics

The system's performance is comprehensively evaluated using a set of key metrics:

1) Accuracy: The accuracy metric represents the ratio of correctly matched fingerprints to the total number of fingerprints. It provides an overall measure of the system's correctness.
2) False Acceptance Rate (FAR): FAR measures the rate at which the system incorrectly accepts an imposter fingerprint as a match. It is crucial for assessing the system's vulnerability to false positive identifications.
3) False Rejection Rate (FRR): FRR measures the rate at which the system incorrectly rejects a genuine fingerprint as a non-match. This metric is essential for evaluating the system's sensitivity to false negatives.
4) Precision: Precision represents the ratio of true positive matches to the total number of accepted matches. It provides insights into the reliability of the system when it claims a positive match.
5) Recall (Sensitivity): Recall, also known as sensitivity, is the ratio of true positive matches to the total number of genuine fingerprints. It measures the system's ability to correctly identify positive matches.

6) F1 Score: The F1 score is the harmonic mean of precision and recall. It provides a balanced measure that considers both false positives and false negatives, offering a comprehensive assessment of the system's performance.

The computation of these metrics involves comparing the system's decisions with ground truth labels, enabling a thorough evaluation of its capabilities in distinguishing between genuine and imposter fingerprints.

This detailed matching and evaluation process, along with the metrics employed, ensures a robust and thorough assessment of the biometric system's performance in fingerprint recognition.

VI. Experimental Results

The system is tested on altered, noisy, and rotated images to evaluate its robustness under various conditions. Metrics are calculated for each scenario, providing insights into the system's performance. The altered and rotated images had a near perfect acceptance rate, while the noise injection function seemed to read closer to 91.2 percent

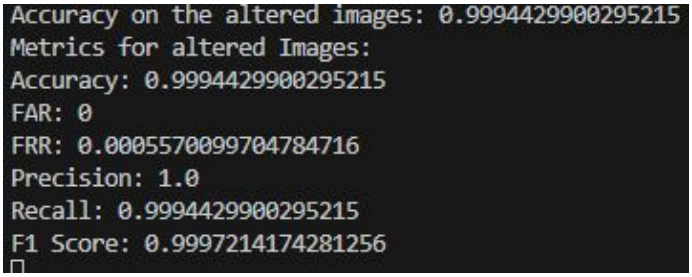

Fig. 1. Altered data

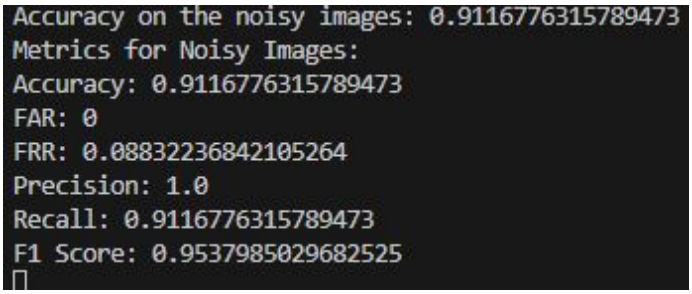

Fig. 2. Noisy data

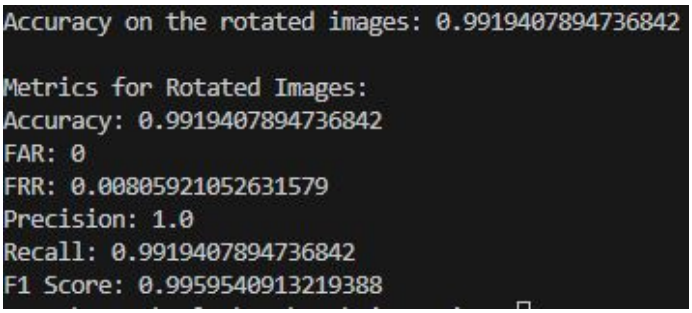

Fig. 3. rotated data

VII. Conclusion

This paper presents a comprehensive bio-metric system combining KNN and Harris Corner Detection for fingerprint recognition. The experimental results showcase its effectiveness under various conditions. Future work may involve exploring additional preprocessing techniques and extending the system to handle larger data-sets.

Original Project

The original intention of this project was to create a simple bio-metric system and test various attacks to obtain metrics of their success/failure rate and judge how strong the fingerprint as a bio-metric really is. The method would have been to utilize a fingerprint scanner instead of a public data-set that could introduce more security risks. Since using a public data-set completely removes a large section of the bio-metric system flowchart it prevents the use of popular/common attacks such as Spoofing. Due to the underestimation of the work that went into creating a fingerprint bio-metric system and the implementation with a public data-set the need to pivot the direction of the project was apparent.

Keywords

Bio-metric System, K-Nearest Neighbors, Harris Corner Detection, Fingerprint Recognition, Image Preprocessing, Template Matching.