Available online at https://journal.rescollacomm.com/index.php/ijqrm/index

International Journal of Quantitative Research and Modeling

e-ISSN 2721-477X
p-ISSN 2722-5046

Vol. 7, No. 2, pp. 136-147, 2026

A CNN-Based Bimodal Biometric Authentication System Using Face and Iris Recognition

Oluwakemi C. Abikoye[1*], Ridwan Opeyemi Kasope[2], Kafayat Odunayo Tajudeen[3], Abimbola Ganiyat Akintola[4]

[1,2,4] *Department of Computer Science, University of Ilorin, Ilorin*
[3] *Department of Computer Science, Al-Hikmah University, Ilorin,*

**Corresponding author email: kotajudeen@alhikmah.edu.ng*

Abstract

In the evolving landscape of information technology, this study presents a bimodal biometric authentication system that combines facial and iris recognition using advanced Convolutional Neural Networks (CNNs) to address the escalating security concerns of personal and sensitive information. The bimodal approach leverages the unique textures of facial and iris features to create a robust and secure authentication mechanism, demonstrating high accuracy (95.76%) and precision value of 97.83%, low False Acceptance Rate (2.54%) and False Rejection Rate (14.29%). The system framework integrates the strengths of both facial and iris modalities, mitigating vulnerabilities inherent in unimodal systems and advancing the field of biometric authentication by providing a resilient solution against emerging cyber threats, enhancing the reliability of user identification, and contributing to safer digital environments across various domains.

Keywords: Unimodal authentication, bimodal authentication, feature extraction, fusion

1. Introduction

In an era where humans heavily rely on authentication for protection, security and privacy are essential for every individual and organisation (Saxena & Varshney, 2021). Authentication is an important aspect of our daily lives; humans' biological, behavioural, and physical characteristics are usually applied in an entity's verification in various applications (Vallabhadas & Sandhya, 2023). Biometric authentication makes use of stored information to verify an individual when they request access to their account. The change from password-based authentication to biometric authentication is significant in the authentication methods as it does not require any memorization (Barkadehi et al., 2018; Khade et al., 2021). The emerging trend is the multimodal biometric system, which has been proven to be more effective than the single biometric authenticator (Vensila & Wesley, 2023). Traditional single-factor authentication methods are increasingly susceptible to parody bimodality and manipulation, which increases the demand for sophisticated solutions. Bimodal biometric authentication, leveraging the fusion of two distinct biometric modalities, offers a promising approach to enhance security and information protection (Mansour et al., 2024).

Finally, bimodal biometric authentication, integrating the distinctive features of both facial and iris recognition, embodies a holistic approach towards enhancing accuracy, resilience, and user acceptance (Ammour et al., 2020). The objective of this research is to reshape secure authentication through the creation of a pioneering bimodal biometric system that combines facial and iris recognition methods, leveraging advanced convolutional neural networks (CNNs) as its foundation.

2. Literature Review

Hafeez et al. (2022) introduced an enhanced iris recognition system that incorporated image registration and an edge detection method for effective feature extraction, along with a proposed Principal Component Analysis (PCA)-based alternative approach utilizing a similarity score for identification. Through experiments on a custom-developed database, the first proposed system demonstrated a noteworthy reduction in computation time, achieving 6.56 seconds, and significantly enhanced accuracy, reaching 99.73%. In contrast, the PCA-based method exhibited lower accuracy than the proposed system. These results underscored the efficacy of the improved iris recognition system,

emphasizing its potential to reduce computation time and elevate accuracy levels, thereby contributing to the advancement of reliable biometric verification techniques.

Ammour et al. (2020) emphasized that the growing interest in multimodal biometrics stems from its ability to overcome the limitations of single biometric modalities, significantly enhancing recognition rates. Their study introduced a feature extraction technique for a bimodal biometric system using face and iris modalities. The proposed method employed a multi-resolution 2D Log-Gabor filter for iris feature extraction and singular spectrum analysis (SSA) combined with wavelet transform for facial features. The relevant features from both modalities were fused at a hybrid level. The evaluation was performed using a chimeric database incorporating ORL, FERET, and CASIA v3.0 Iris datasets. The experimental results demonstrated the robustness of the proposed multimodal biometric system.

Arora, Bhatia and Kukreja (2020) introduced a novel approach to multimodal biometric systems focusing on fusing features extracted from the face and the iris using deep learning. Facial and iris features were separately extracted using CNN models, and the feature vectors from the final CNN layers were then fused to achieve improved individual classification. The effectiveness of the proposed multimodal system was evaluated using the CASIA-Face V5 and IITD iris datasets, and the results demonstrated its superiority in efficiency, reliability, and robustness compared to unimodal biometric systems. This research contributes to advancing biometric verification systems by offering a more secure and accurate approach through the fusion of facial and iris features, mitigating the limitations of unimodal systems.

Abikoye et al. (2018) found that the feature extraction technique is critically important for iris recognition performance. They compared Gabor Wavelet Transform, SIFT, and Haar Wavelet, using a CASIA dataset. The Gabor Wavelet outperformed the others in accuracy and efficiency, suggesting its exceptional ability to capture iris details into discriminating features, making it a strong choice for robust iris recognition systems. The study emphasizes the critical importance of selecting the right feature extraction algorithm to avoid undermining the classification capabilities.

Abikoye et al. (2014) explored the use of Fast Wavelet Transform (FWT) for iris feature extraction, a crucial step in iris recognition systems for personal identification. The authors highlighted the unique texture of the iris, comprising interlacing features that collectively form a distinct signature for each individual. Emphasizing the importance of iris feature extraction, the researchers utilized the FWT algorithm, citing its speed and low computational complexity as advantages over other methods. By employing the FWT algorithm, Abikoye et al. (2014) successfully generated iris feature codes, which could be used for subsequent comparison and identification processes, contributing to the advancement of iris recognition systems.

3. Materials and Methods

3.1. Data Collection

This data collection process involved a total of 108 participants. For the face data, a mobile phone equipped with a high-resolution camera was used. Specifically, the Redmi Note 12's 50MP main camera was used. The iris data was collected using a dedicated iris camera, the VistaEY2H from Vista Imaging.

3.2. Face image preprocessing

Preprocessing is a critical step in ensuring accurate and consistent results in image analysis. For face detection and alignment, the Viola-Jones algorithm was employed to detect and localize face regions, followed by the detection of facial landmarks such as eyes, nose, and mouth to align and centre the faces. To enhance image quality, Gaussian filters and histogram equalization were utilized, improving clarity for subsequent processing. Normalization and resizing techniques, including scaling, cropping, and padding, were applied to maintain consistency across all images. Finally, Gaussian blurring was used for noise reduction, smoothing out images and eliminating extraneous noise for cleaner and more reliable inputs.

Viola jones algorithm for face detection (Younis & Ramo, 2023):

a. Begin
b. While (max number of frame <> 0)
c. Begin
d. Find the first frame that has a face region by implementing the viola jones algorithm and give it value as frame= 1.
e. If frame=1 do
  A. Implement viola jones algorithm.
  B. Crop the face region and save it in the im matrix.
f. Else
  A. Implement viola jones algorithm.
  B. Calculate the number of regions as the face in the count value.
  C. If count=0 then
    1) Implement the NCC algorithm to find High Peak as the face region.

2) Crop the highest peak region.

3) Implement the Manhattan technique between the face template of the previous frame and the current region frame.

D. Else if count > 1

1) Crop all regions that represent a face.

2) Implement Manhattan technique between face template of the previous frame and current regions frame and return distances: D1, D2...Dn

3) Find max D, and return the face region of it.

E. End if

F. End if

G. Exchange the im matrix with the current face region.

g. End if

h. End while

i. End.

Gaussian filter algorithm from (Kocamuha, n.d.) for Image enhancement

```
for j = 1:252
   for i = 1:252
      sum = 0.0;
      for k1 = 1: length(h)
         for k2 = 1: length(h)
            sum = □□□ + □□□□□□□(□(□ + □1 − 1, □ + □2 − 1)) ∗ h(□2, □1)
         end
      end
      filtered (j, i) = sum;
   end
end
```

Histogram equalisation for image enhancement from Kocamuha, E. (n.d.)

- ➔ Step 1: Get the input image
- ➔ Step 2: Generate the histogram for the image
- ➔ Step 3: Find the local minima of the image
- ➔ Step 4: Divide the histogram based on the local minima
- ➔ Step 5: Have the specific grey levels for each partition of the histogram
- ➔ Step 6: Apply the histogram equalisation on each partition

Gaussian blur algorithm for Image noise reduction

- Step 1: Create a small matrix representing the gaussian distribution
- Step 2: Add a border around the original image to prevent edge effects during the blurring process.
- Step 3: Slide the gaussian kernel over the image, element-wise multiplying the kernel with the corresponding image pixels. Sum the products to get the blurred pixel value.
- Step 4: Normalise and divide the blurred pixel value by the sum of the kernel elements as the original.
- Step 5: apply step 3-4 to every pixel in the image to produce the final blurred image.
- Step 6: Discard the added border, returning the blurred image to its original size.

3.3. Iris data preprocessing

Preprocessing of iris images is vital for accurate and reliable iris recognition. The first step, iris localization, involves using the Circular Hough transform from the OpenCV module to efficiently detect and isolate the iris region. Next, iris segmentation is performed using the Canny edge detector to accurately separate the iris from other eye components, ensuring only relevant texture and pattern information is extracted. Finally, normalization is achieved through Daugman's rubber-sheet model, converting the circular iris region into a standardized rectangular form for consistent representation, facilitating precise iris pattern matching and recognition.

Circular Hough transform Algorithm for Iris localization

$Input$:

(image: $\{I_{i,j}\}_{i,j}$): $a2Darrayofpixels\ (grayscaleimage)$

(radius_range = $(r_{\min}, r_{\max})$): $range\ of\ possible\ radii\ for\ the\ circles$

(edge_threshold): $threshold\ for\ edge\ detection\ (used\ in\ edge\ detection\ step)$
(accumulator_threshold): $threshold\ to\ determine\ if\ a\ point\ in\ the\ accumulator\ array\ is\ considered\ a\ circle\ center$

$Output$:
(detected_circles): $listofdetectedcircles, eachrepresentedas((x, y, r))$
$Step\ 1$: $Edge\ Detection$
edges = EdgeDetection(image,edge_threshold)

$Step\ 2$: $Initialize\ Accumulator\ Array$
[min_radius = $r_{\min}$]
[max_radius = $r_{\max}$]
[accumulator = zeros$\big((W, H, r_{\max} - r_{\min} + 1)\big)$]
$where\ (W)\ and\ (H)\ are\ the\ width\ and\ height\ of\ the\ image, respectively.$

$Step\ 3$: $Voting\ in\ Accumulator\ Array$:
$Foreach$(edge_pixel ∈ edges):
[$\big(x_{\text{edge}}, y_{\text{edge}}\big)$ = coordinates of edge_pixel]
$For(r) from(r_{\min}) to(r_{\max})$:
[θ = 0° to 360°]
[$a = x_{\text{edge}} - r \cdot \cos(\theta)$]
[$b = y_{\text{edge}} - r \cdot \sin(\theta)$]
$If\ (a)\ and\ (b)\ are\ within\ image\ bounds$:
[accumulator$[a, b, r - r_{\min}]$ += 1]

$Step\ 4$: $Detect\ Circles\ from\ Accumulator\ Array$:
detected_circles = []
$For(a = 0$ to $W - 1)$:
$For(b = 0$ to $H - 1)$:
$For(r = 0$ to $r_{\max} - r_{\min})$:
If(accumulator$[a, b, r] \geq$ accumulator_threshold):
[detected_circles.append$\big((a, b, r + r_{\min})\big)$]
$Return$ (detected_circleis)

$FunctionEdgeDetection$((image,edge_threshold)):
$Implementedgedetection$
[edges = 2D array of detected edge pixels]
$Return$(edges)

Canny edge detection Algorithm for iris edges detection
$AlgorithmCannyEdgeDetector$((image,low_threshold,high_threshold)):
$Input$:
(image: $\{I_{i,j}\}_{i,j}$): $a\ 2D\ array\ of\ pixels(grayscaleimage)$
(low_threshold): $lower\ bound\ for\ hysteresis\ thresholding$
(high_threshold): $upper\ bound\ for\ hysteresis\ thresholding$

$Output$:
(edges): $2D\ binary\ array\ where\ edge\ pixels\ are\ marked$
$Step\ 1$: $Gaussian\ Blur$:
blurred_image = GaussianBlur(image)

$Step\ 2$: $Gradient\ Calculation$:
(gradient_magnitude,gradient_direction) = ComputeGradients(blurred_image)

$Step\ 3$: $Non - Maximum\ Suppression$:
suppressed_image = NonMaximumSuppression(gradient_magnitude,gradient_direction)

$Step$ 4: $Double\ Threshold$:
(strong_edges,weak_edges) = DoubleThreshold(suppressed_image,low_threshold,high_threshold)

$Step$ 5: $Edge\ Tracking\ by\ Hysteresis$:
edges = EdgeTrackingByHysteresis(strong_edges,weak_edges)
$Return$(edges)

$FunctionGaussianBlur$((image)):
blurred_image = Convolve(image,GaussianKernel( ))
$Return$(blurred_image)

$FunctionComputeGradients$((image)):
gradient_x = Convolve(image,SobelKernelX( ))
gradient_y = Convolve(image,SobelKernelY( ))
gradient_magnitude = $\sqrt{\text{gradient\_x}^2 + \text{gradient\_y}^2}$
gradient_direction = arctan 2 (gradient_y,gradient_x)
$Return$(gradient_magnitude,gradient_direction)

$FunctionNonMaximumSuppression$((gradient_magnitude,gradient_direction)):
suppressed_image = zeros_like(gradient_magnitude)
$For\ each\ pixel\ ((i,j))\ in$ (gradient_magnitude):
angle = gradient_direction[i][j]
if is_local_maximum(gradient_magnitude, i, j,angle):
suppressed_image[i][j] = gradient_magnitude[i][j]
$Return$(suppressed_image)

$FunctionDoubleThreshold$((suppressed_image,low_threshold,high_threshold)):
strong_edges = zeros_like(suppressed_image)
weak_edges = zeros_like(suppressed_image)
$For\ each\ pixel((i,j))\ in$ (suppressed_image):
if suppressed_image[i][j] $\geq$ high_threshold:
strong_edges[i][j] = 1
elif suppressed_image[i][j] $\geq$ low_threshold:
weak_edges[i][j] = 1
$Return$(strong_edges,weak_edges)

$FunctionEdgeTrackingByHysteresis$((strong_edges,weak_edges)):
edges = copy(strong_edges)
$For\ each\ pixel((i,j))\ in$ (strong_edges):
if strong_edges[i][j] == 1:
TrackEdge(edges,weak_edges, i, j)
$Return$(edges)
$FunctionTrackEdge$((edges,weak_edges, i, j)):
$Foreachneighbor((ni,nj))of((i,j))$:
if weak_edges[ni][nj] == 1 and edges[ni][nj] == 0:
edges[ni][nj] = 1
[TrackEdge(edges,weak_edges, ni, nj)]

$Function\ is\ local\ \ maximum$((gradient_magnitude, i, j,angle)):
$Check\ if\ the\ current\ pixel\ is\ a\ local\ maximum\ in\ the\ gradient\ direction$:
$If angle\ is\ in\ the\ range\ of\ 0\ degrees(horizontal)$:
return gradient_magnitude[i][j] > gradient_magnitude[i][$j-1$] and gradient_magnitude[i][j] > gradient_magnitude[i][$j+1$]
$If\ angle\ is\ in\ the\ range\ of\ 45\ degrees(diagonal)$:
return gradient_magnitude[i][j] > gradient_magnitude[$i-1$][$j+1$] and gradient_magnitude[i][j] > gradient_magnitude[$i+1$][$j-1$]
$If\ angle\ is\ in\ the\ range\ of\ 90\ degrees(vertical)$:

return gradient_magnitude$[i][j]$ > gradient_magnitude$[i-1][j]$ and gradient_magnitude$[i][j]$
> gradient_magnitude$[i+1][j]$
$-$ *If angle is in the range of* 135 *degrees* (*diagonal*):
return gradient_magnitude$[i][j]$ > gradient_magnitude$[i-1][j-1]$ and gradient_magnitude$[i][j]$
> gradient_magnitude$[i+1][j+1]$
Return(False)

FunctionConvolve((image,kernel)):
output_image = convolve2d(image,kernel,mode =′ *same*′)
Return(output_image)

FunctionGaussianKernel( ):
Return Gaussian kernel for blurring

FunctionSobelKernelX( ):
Return Sobel kernel for gradient calculation in the x direction

Function SobelKernelY():
Return Sobel kernel for gradient calculation in the y direction

Daugman rubbersheet model for image normalization

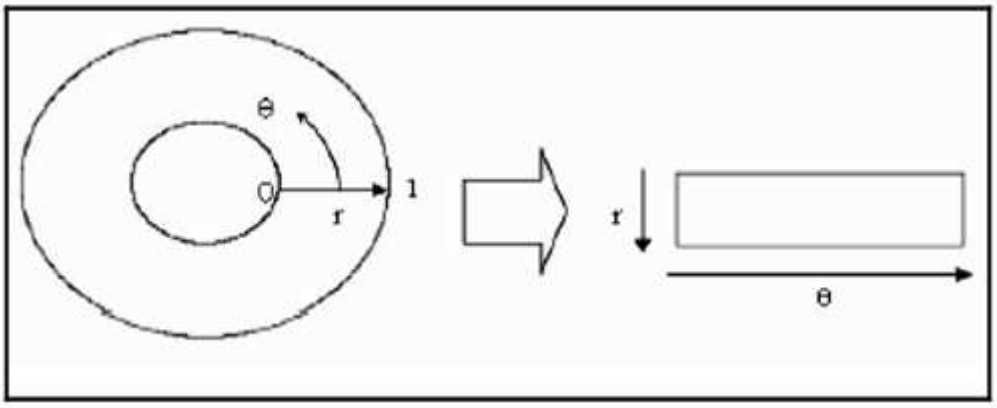

Figure 1: Daugman rubber sheet model *from the study made by (Omran & AlShemmary, 2020)*

3.4 Face feature extraction

For the face modality, a custom CNN was used to detect key landmarks and extract facial features. The CNN consists of layers that detect edges, textures, and complex facial features, culminating in a face embedding that numerically represents these features. This embedding is used to calculate a similarity score with enrolled face templates. The model was fine-tuned with a pre-trained VGGFace2 model to improve accuracy and robustness.

Pseudo code for extracting face features using a custom CNN

```
# Function to preprocess the input face image
def preprocess_face_image(face_image):
    resized_image = resize_image (face_image, (224, 224))
    normalized_image = normalize_image(resized_image)
    tensor_image = convert_to_tensor(normalized_image)
    return tensor_image

# Define the CNN architecture for face feature extraction
def custom_face_cnn():
    model = Sequential()
    # Add convolutional and pooling layers
    model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(224, 224, 3)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
    # Add more convolutional and pooling layers as needed
    model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    # Flatten the output and add dense layers
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    # Output layer for face embedding
    model.add(Dense(128, activation='linear'))
    return model

# Function to extract face embedding
def extract_face_embedding(face_image, model):
    # Preprocess the input face image
    preprocessed_image = preprocess_face_image(face_image)
    # Pass the image through the CNN model to get the embedding
    face_embedding = model.predict(preprocessed_image)
    return face_embedding

# System usage
face_image = load_face_image('path_to_face_image.jpg')
face_cnn_model = custom_face_cnn()
face_embedding = extract_face_embedding(face_image, face_cnn_model)
```

3.5 Iris feature extraction

For the iris modality, a custom CNN was used to extract iris features. The CNN layers detect textures, patterns, and complex features like ridges and furrows, producing an iris embedding that numerically represents unique iris patterns. This embedding calculates a similarity score with enrolled iris templates. The model was enhanced and fine-tuned using a pre-trained IrisNet101 model to improve feature extraction and performance.

Pseudo code for extracting iris features using a custom CNN

```
# Function to preprocess the input iris image
def preprocess_iris_image(iris_image):
    resized_image = resize_image(iris_image, (224, 224))
    normalized_image = normalize_image(resized_image)
    tensor_image = convert_to_tensor(normalized_image)s
    return tensor_image
# Define the CNN architecture for iris feature extraction
def custom_iris_cnn():
    model = Sequential()
    # Add convolutional and pooling layers
    model.add(Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(224, 224, 3)))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    # Add more convolutional and pooling layers as needed
    model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
    model.add(MaxPooling2D(pool_size=(2, 2)))
    # Flatten the output and add dense layers
    model.add(Flatten())
    model.add(Dense(128, activation='relu'))
    # Output layer for iris embedding
```

```
    model.add(Dense(128, activation='linear'))
    return model
# Function to extract iris embedding
def extract_iris_embedding(iris_image, model):
    preprocessed_image = preprocess_iris_image(iris_image)
    iris_embedding = model.predict(preprocessed_image)
    return iris_embedding
# System usage
iris_image = load_iris_image('path_to_iris_image.bmp')
iris_cnn_model = custom_iris_cnn()
iris_embedding = extract_iris_embedding(iris_image, iris_cnn_model)
```

3.6 Fusion

To combine the face and iris extracted features, we used an approach called feature-level fusion, specifically the concatenation method. Feature-level fusion directly merges the feature vectors extracted from each modality before any classification or matching is performed. This allows for a richer and more comprehensive representation of the data.

3.7 Matching

The Euclidean distance between the feature vector of the new image and the template vectors is calculated to match a new image to a template. The Euclidean distance between two vectors A and B in n-dimensional space is given by:

$$[\mathrm{d}(\mathrm{x},\mathrm{y}) = \sqrt{\sum_{\mathrm{i}=1}^{\mathrm{n}} (\mathrm{x_i} - \mathrm{y_i})^2}]$$

Where:
- $(\mathrm{x})\mathrm{and}(\mathrm{y})$ are the face or iris embeddings (feature vectors) of the input and enrolled templates, respectively.
- $(\mathrm{x_i})$ and $(\mathrm{y_i})$ are the components of the vectors $(\mathrm{x})\mathrm{and}(\mathrm{y})$
- *n* is the dimensionality of the feature vectors.

3.8 Evaluation

To evaluate how well our bimodal authentication system is performing, several performance metrics were used. First, we calculated the accuracy to know what percentage of authentication attempts were correctly approved or denied. The precision was also measured, indicating the amount of genuine user attempts. FAR and FRR (False Acceptance Rate and False Rejection Rate, respectively) were used to calculate the percentage of incorrect authentications and rejections

False Acceptance Rate (FAR)

$$\mathrm{FAR} = \frac{\text{Number of False Acceptances}}{\text{Total Number of Impostor Attempts}} \times 100\%$$

False Rejection Rate (FRR)

$$\mathrm{FRR} = \frac{\text{Number of False Rejections}}{\text{Total Number of Genuine Attempts}} \times 100\%$$

4. Results and discussion

4.1. Face Image Acquired

A total of 108 face images were collected and processed to extract facial features.

Figure 2: The images above are of the users 001, 002, 003 and 004

4.2 Iris Image Acquired

A total of 180 iris images were collected and processed to extract iris features. The system captures both left and right iris images for each user.

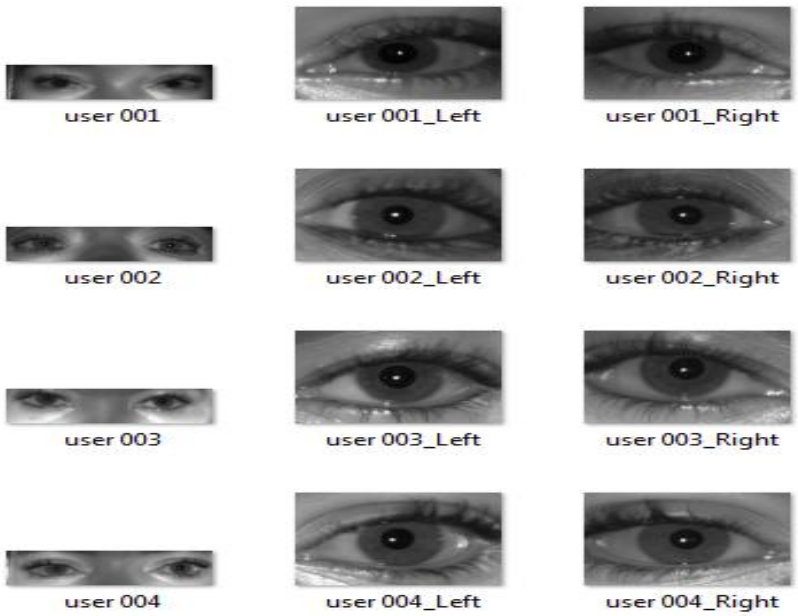

Figure 3: The images above are the dual, left and right irises of users 001, 002, 003 and 004 respectively.

4.3 Case Study (Successful Authentication of user 002)

For a successful authentication, the UI workflow is as follows:

1. **User selects face image**: The user is guided to upload their face image.
2. **User selects iris image**: The user uploads their iris image.
3. **System processes and displays results**: The system compares the provided images with the enrolled templates. If a match is found, a success message along with the fusion score is displayed.

Authentication

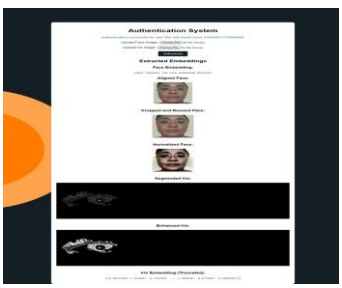

Figure 4: User interface for a successfully authenticated user.

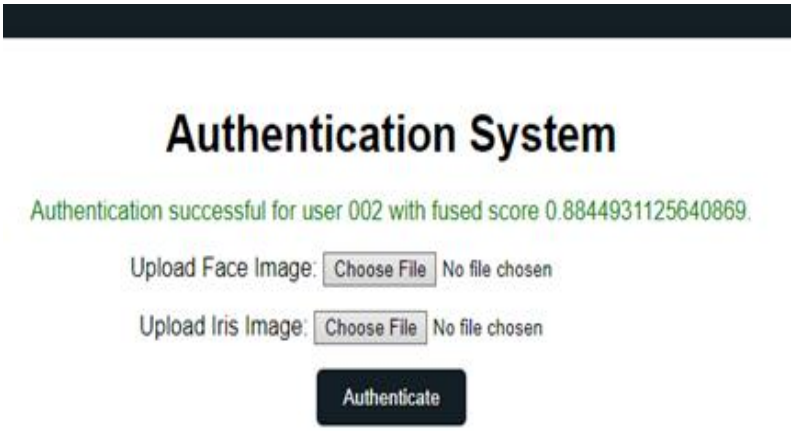

Figure 5: User 002 was authenticated successfully with a fusion score of 0.88.

Authentication outcome

The image displays the results of a biometric authentication system, detailing both face and iris preprocessing steps, followed by the generation of embeddings, and concluding with the authentication outcome. The face preprocessing involves aligning the face by detecting key facial landmarks, cropping, and resizing it to a standard dimension, and normalizing the pixel values to ensure consistency and improve model performance. In the iris preprocessing, the system segments the iris, isolating it from the rest of the eye, and then enhances this segmented iris to improve the quality of features extracted. Both the face and iris embeddings are numerical representations, capturing unique patterns from the respective images. These embeddings are used to verify identity by comparing them with stored data. The displayed authentication result indicates a successful verification of User 002, with a fused score of 0.8844931125640869. This score represents a combination of the face and iris embedding comparisons, where a value above a certain threshold confirms a match and leads to successful authentication.

4.4 Case Study (Failed authentication)

For a successful authentication, the UI workflow is as follows:

1. **User selects face image**: The user is guided to upload their face image.
2. **User selects iris image**: The user uploads their iris image.
3. **System processes and displays results**: The system compares the provided images with the enrolled templates. If a match is found, a success message along with the fusion score is displayed else authentication failed.

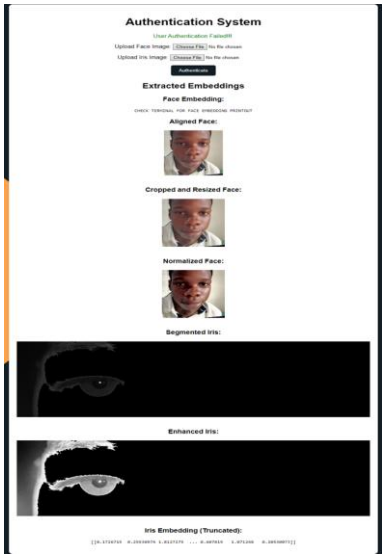

Figure 6: User interface a failed authentication

Figure 7: Failed authentication

Authentication outcome

The image depicts the results of the bimodal biometric authentication system where the user authentication process has failed. The system initially processes the face by aligning it using key facial landmarks, followed by cropping and resizing the face to a standard dimension for consistency. The image is then normalized to standardize the pixel values, which helps reduce the impact of lighting variations. For the iris, the system first segments it to isolate the iris from the rest of the eye, and then enhances the segmented iris to highlight important details for feature extraction. The face and iris embeddings, which are numerical representations of the unique patterns in the respective images, are generated next. These embeddings are used to compare the input data with stored data for identity verification. However, in this instance, the system indicates that the authentication process failed, meaning the generated embeddings did not sufficiently match the stored data, resulting in a failed authentication attempt.

4.5 System Evaluation

The preliminary evaluation of the system shows promising results, with the following performance metrics:

Figure 8: system evaluation report

Table System evaluation reports

Table 1: The evaluation of the system and their corresponding values.

Metric	Value
System Accuracy	95.76%
System Precision	97.83%
False Acceptance Rate (FAR)	2.54%
False Rejection Rate (FRR)	14.29%

These metrics shown from the terminal indicate that the system correctly approved and denied most authentication attempts during the evaluation. The accuracy value is 95.76% which shows that the system is perform very well and it is able to authenticate users, the precision value of 97.83% shows that most genuine user attempts were correctly authenticated, while the FAR and FRR values of 2.54% and 14.29% indicate a small percentage of incorrect authentications

False Rejection Rate (FRR) Report

Table 2: FRR reports for 3 tested users

User ID	False Rejections	FRR
001	6	31.58%
066	1	8.33%
072	0	0.00%

Combined FRR for All Users: 14.29%

5. Conclusion

The bimodal authentication system developed combines face and iris recognition for high accuracy and reliability. Using CNN-based models for feature extraction, the system underwent data collection, preprocessing, feature extraction, fusion of modalities, and matching. This integration enhances accuracy, security, and robustness, demonstrating effective identity verification and adaptability to biometric changes over time.

References

Bau, M. K. D., Setyawan, Y., & Jatipaningrum, M. T. (2023). Comparison of the K-Means and K-Medoids Algorithm Methods for Clustering Regencies/Cities in East Nusa Tenggara Based on the 2020 Human Development Index. *Journal of Industrial Statistics and Computation*, 8(1), 48-57.

Hidayat, Y., Purwandari, T., Sukono, et al. (2023). Mean-Value-at-Risk portfolio optimization based on risk tolerance preferences and asymmetric volatility. Mathematics, 11, 4761.

Markowitz, H. (1952). Portfolio selection. The Journal of Finance, 7(1), 77-91.

Parrak, R., & Seidler, J. (2010). Mean-Variance & Mean-VaR portfolio selection: A simulation based comparison in the Czech crisis environment. IES Working Paper 27/2010.

Pramesti, D. F., Furqon, M. T., & Dewi, C. (2017). Implementation of the K-Medoids Clustering Method for Grouping Forest and Land Fire Potential Data Based on Hotspot Distribution. Jurnal Pengembangan Teknologi Informasi dan Ilmu Komputer, 1(9), 723-732.

Puspaningsih, E. S., Di Asih, I. M., & Tarno, T. (2024). K-Medoids Clustering and Mean–Value-at-Risk for Portfolio Optimization of Jakarta Islamic Index Stocks. Indonesian Journal of Applied Statistics, 6(1), 85-95.

Rudianto, R. D., & Wijayanto, A. W. (2024). Comparative Analysis of K-Means and K-Medoids for Clustering Provinces Based on the Indonesian Democracy Index. Komputika: Jurnal Sistem Komputer, 13(1), 19-27.

Sarykalin, S., Serraino, G., & Uryasev, S. (2008). Value-at-Risk vs. Conditional Value-at-Risk in risk management and optimization. INFORMS Tutorials in Operations Research.

Wahid, A. J., & Saputra, J. (2025). Portfolio Performance Analysis with Jensen's Alpha Using Single Index Model and CAPM on IDX30 Stocks. International Journal of Quantitative Research and Modeling, 6(2), 274-283.

Wahid, A. J., Riaman, R., & Sukono, S. (2025). A Systematic Literature Review on Mean-CVaR Based Financial Asset Portfolio Weight Allocation Using K-Means Clustering. CAUCHY: Jurnal Matematika Murni dan Aplikasi, 10(2), 1069-1091.