บทความวิจัย (Research Article)

Face and Hand Gesture Recognition System Using MediaPipe

Sethakarn Prongnuch[1,*], Tanawat Sangkhakul[2], Nutchanon Angchoun[2],
Wuttipong Pinsarmsak[2] and Kant Charoenjit[2]

[1] Department of Robotics Engineering, Faculty of Engineering and Industrial Technology, Suan Sunandha Rajabhat University
[2] Department of Computer Engineering, Faculty of Engineering and Industrial Technology, Suan Sunandha Rajabhat University

* Corresponding Author: sethakarn.pr@ssru.ac.th, Tel: 084-1782616
(Received: April 15, 2025; Revised: May 12, 2025; Accepted: May 19, 2025)

Abstract

Current recognition systems face challenges related to accuracy, real-time speed, and limited device resources. This paper presents a face and hand gesture recognition system using MediaPipe. The research aims to create a real-time face and hand gesture recognition system utilizing MediaPipe, OpenCV, and SciPy, running on a Raspberry Pi 4 board with a webcam. The experiment involved a sample group of 100 participants. The experimental results show that the system achieves a success rate of approximately 95% in unlocking a security door for individuals whose information is stored in the database, despite limitations in camera angles and lighting. The highlight of this research is its presentation of a dual authentication method using the cosine similarity technique for security systems. The system's performance and accuracy will be further evaluated in real-world scenarios, such as its application in educational institutions within the Thai context.

Keywords: Face Recognition, Hand Gesture Recognition, OpenCV, MediaPipe, SciPy

Please cite this article as: S. Prongnuch, T. Sangkhakul, N. Angchoun, W. Pinsarmsak and K. Charoenjit, "Face and Hand Gesture Recognition System Using MediaPipe," *The Journal of Engineering and Industrial Technology, Kalasin University*, vol. 3, no. 4, pp. 45-58, 2025.

บทความวิจัย (Research Article)

ระบบการรู้จำใบหน้าและท่าทางมือโดยใช้มีเดียไปป์

เศรษฐกาล โปร่งนุช[1,*], ธนวัฒน์ สังขกุล[2], นัชชานนท์ เอ้งฉ้วน[2], วุฒิพงศ์ ปิ่นเสริมศักดิ์[2] และ กานต์ เจริญจิตร[2]

[1] สาขาวิชาวิศวกรรมหุ่นยนต์ คณะวิศวกรรมศาสตร์และเทคโนโลยีอุตสาหกรรม มหาวิทยาลัยราชภัฏสวนสุนันทา
[2] สาขาวิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์และเทคโนโลยีอุตสาหกรรม มหาวิทยาลัยราชภัฏสวนสุนันทา

*ผู้ประสานงานบทความต้นฉบับ: sethakarn.pr@ssru.ac.th โทรศัพท์: 084-1782616

(รับบทความ: 12 เมษายน 2568; แก้ไขบทความ: 12 พฤษภาคม 2568; ตอบรับบทความ: 19 พฤษภาคม 2568)

บทคัดย่อ

ระบบการรู้จำในปัจจุบันเผชิญกับความท้าทายที่เกี่ยวข้องกับความแม่นยำ ความเร็วตามเวลาจริง และทรัพยากรที่มีจำกัด บทความนี้นำเสนอระบบการรู้จำใบหน้าและท่าทางมือโดยใช้มีเดียไปป์ งานวิจัยนี้มีวัตถุประสงค์เพื่อสร้างระบบการรู้จำใบหน้าและท่าทางมือตามเวลาจริงโดยใช้คลังโปรแกรมมีเดียไปป์ โอเพนซีวี และไซพาย ซึ่งทำงานบนบอร์ดราสเบอร์รี่พาย รุ่นที่ 4 พร้อมเว็บแคม การทดลองนี้เลือกใช้กลุ่มตัวอย่างจำนวน 100 คน ผลการทดลองแสดงให้เห็นว่าระบบมีอัตราความสำเร็จในการปลดล็อคประตูรักษาความปลอดภัยสำหรับบุคคลที่มีข้อมูลเก็บไว้ในฐานข้อมูล ร้อยละ 95 เนื่องจากมีข้อจำกัดของมุมกล้องและแสง จุดเด่นของงานวิจัยนี้คือการนำเสนอแนวทางการตรวจสอบสิทธิ์แบบคู่ที่ใช้เทคนิคความคล้ายคลึงโคไซน์สำหรับระบบความปลอดภัย สามารถนำระบบนี้ไปประเมินประสิทธิภาพและความแม่นยำเพิ่มเติมในสถานการณ์โลกแห่งความเป็นจริง อาทิเช่น การประยุกต์ใช้ในสถานศึกษาตามบริบทไทย

คำสำคัญ: การรู้จำใบหน้า การรู้จำท่าทางมือ โอเพนซีวี มีเดียไปป์ ไซพาย

การอ้างอิงบทความ: เศรษฐกาล โปร่งนุช, ธนวัฒน์ สังขกุล, นัชชานนท์ เอ้งฉ้วน, วุฒิพงศ์ ปิ่นเสริมศักดิ์ และ กานต์ เจริญจิตร, "ระบบการรู้จำใบหน้าและท่าทางมือโดยใช้มีเดียไปป์," *วารสารวิศวกรรมและเทคโนโลยีอุตสาหกรรม มหาวิทยาลัยกาฬสินธุ์*, ปีที่ 3, ฉบับที่ 4, หน้า 45-58, 2568.

บทความวิจัย (Research Article)

1. Introduction

In the present era, facial recognition technology and hand gestures are increasingly being utilized. Research challenges in face and hand gesture recognition include environmental and visual limitations, data bias and representation issues, technical and computational difficulties, dataset limitations, user-centric and ethical considerations, and deployment in challenging environments.

For example, security systems combine facial recognition and hand gestures with artificial intelligence to verify a user's identity. This research proposes a method for picking up and placing hazardous objects using a hand gesture-controlled robotic arm system via a Kinect camera, designed for use in hazardous areas [1]. Numerous facial recognition research projects employ the You Only Look Once (YOLO) in combination with various algorithms [2-3]. Real-world applications are gaining popularity, particularly for implementation on low-power microcontroller boards [4]. Additionally, research using MediaPipe has enabled simple computer control through facial gesture recognition [5].

MediaPipe provides a suite of libraries and tools that enable the rapid implementation of artificial intelligence techniques in applications. By using MediaPipe as a ready-made model, developers can achieve high performance, multi-platform compatibility, and low resource consumption. Its applications span various domains, including virtual reality or augmented reality, disability assistance, gaming, healthcare, and education, all supported by an active open-source community driving its continuous development.

Therefore, this paper presents face and hand gesture recognition system using MediaPipe. The research objectives are divided into 5 issues as follows: 1) Study and apply Raspberry Pi 4 board to create a face and hand gesture recognition system. 2) Design a real-time face and hand gesture detection and comparison system by comparing the detected data with the database. 3) Write an algorithm for a face recognition and hand gesture detection system using MediaPipe. 4) Develop a security system that can access data from face and hand gestures. 5) Evaluate the performance of the created system in a real environment and test the accuracy of the data in recognizing faces and hand gestures.

The novelty of this article lies in the presentation of a dual authentication approach that utilizes the cosine similarity technique for security systems. The system demonstrates promising capabilities in hand gesture recognition using this method. This work underscores the need for greater robustness in both face and hand gesture recognition algorithms, particularly when deployed in real-world, dynamic environments. It also emphasizes the importance of addressing detection inconsistencies to ensure stable and reliable performance across all frames.

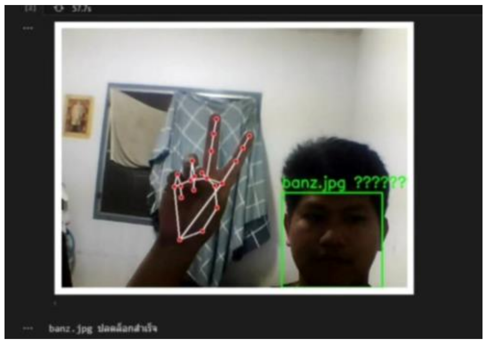
The paper is structured as follows: Section 2 reviews related work, Section 3 research methodology, Section 4 describes the experimental setup, Section 5 presents results, and Section 6 concludes.

2. Theory and Related Work

2.1 MediaPipe

MediaPipe is an open-source framework developed by Google Inc. It is a library designed for processing multi-modal data such as images, videos, audio, and sensor data. One of MediaPipe's key advantages is that it simplifies development and reduces the time required to create algorithms for computer vision, machine learning, artificial intelligence, and augmented reality (AR) [6]. As a result, algorithms using MediaPipe are more efficient and widely adopted. The features of the MediaPipe library comprises six key components:

- A structured pipeline framework with nodes for tasks such as data input, object detection, and processing.
- Core elements including graphs (data flow structure), nodes (functional units), and packets (data carriers).
- Pre-trained models for tasks like face and hand tracking, pose estimation, and 3D object detection.
- Landmark detection for identifying key points on faces, hands, or bodies, as illustrated in Figure 1.



```
banz.jpg ปลดล็อกสำเร็จ
```

Figure 1 Example of using Landmark detection function for recognition

- Cross-platform support for mobile, web, and desktop.
- Real-time, low-power processing.

2.2. OpenCV

OpenCV (Open Source Computer Vision Library) is a freely available software library designed for computer vision and machine learning applications. It was created to provide a standardized framework for computer vision tasks and to facilitate the adoption of machine perception in commercial products [7]. OpenCV stands out due to its extensive documentation and strong developer interest. Furthermore, studies have shown that OpenCV offers excellent performance in processing image data, including its use in computer vision frameworks for object monitoring [8].

2.3 SciPy

SciPy (Scientific Python) is a Python library built on NumPy, designed to support advanced scientific and numerical computations. Developed as an open-source project on GitHub [9], it offers tools for linear algebra, integration, optimization, and statistical analysis, enabling efficient solutions for complex mathematical and scientific problems. SciPy includes advanced capabilities such as derivative computation, Fourier transforms, and signal processing, with optimization functions designed for minimizing multivariable functions. It is widely utilized in engineering and research applications, including face mask detection using deep learning techniques [10].

2.4 Raspberry Pi 4

The Raspberry Pi 4 Model B, shown in Figure 2, is a compact, cost-effective computer for education and IoT applications. Powered by a 1.5 GHz Quad-core Cortex-A72 CPU and VideoCore VI GPU, it supports 4K video and comes with 1GB, 2GB, or 4GB of RAM.

It offers connectivity via USB-C, dual micro-HDMI, Gigabit Ethernet, 40 GPIO pins, and Wi-Fi/Bluetooth 5.0 [11]. Its efficiency and versatility make it ideal for embedded systems and technology-assisted learning research [12].

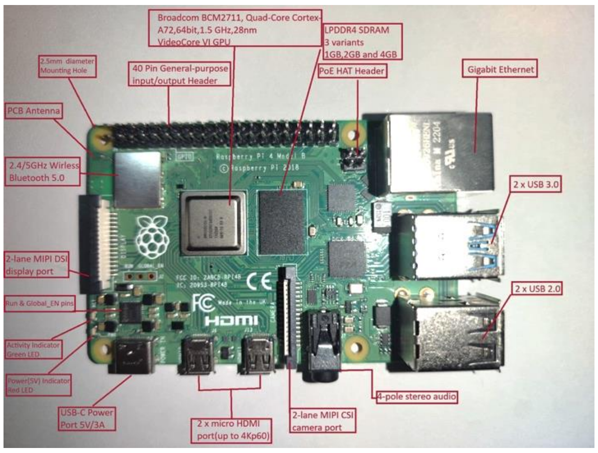

Broadcom BCM2711, Quad-Core Cortex-A72,64bit,1.5 GHz,28nm VideoCore VI GPU
LPDDR4 SDRAM 3 variants 1GB,2GB and 4GB
2.5mm diameter Mounting Hole
40 Pin General-purpose input/output Header
PoE HAT Header
Gigabit Ethernet
PCB Antenna
2.4/5GHz Wirless Bluetooth 5.0
2 x USB 3.0
2-lane MIPI DSI display port
2 x USB 2.0
Run & Global_EN pins
Activity Indicator Green LED
Power(5V) Indicator Red LED
4-pole stereo audio
USB-C Power Port 5V/3A
2 x micro HDMI port(up to 4Kp60)
2-lane MIPI CSI camera port

Figure 2 Raspberry Pi 4 Model B

2.5 Related Work

Numerous research projects have proposed methods to recognize users' faces and hand gestures. These methods have been applied to verify user identities and are widely used in smart and electronic devices.

Table 1 Comparative of related work

Research	Authentication	Accuracy	F1-Score	Cost
YSAF [2]	Face detection	92%	90%	High
Deep Learning [3]	Hand gesture recognition	88%	85%	High
Multimodal [13]	Password, Face, & Audio	N/A	99%	N/A

Table 1 compares existing authentication methods based on accuracy, F1-score, and cost. The YSAF (Yolo-Spatial Attention-FFT) [2] and hand gesture recognition using deep learning [3] approaches use face detection and hand gesture recognition, achieving 92% or 90% and 88% or 85% for accuracy and F1-score respectively, though both incur high costs. The Multimodal authentication [13], combining password, face, and audio, achieved a 99% F1-score, but its accuracy and cost were not reported. This highlights the trade-off between performance and implementation cost across different techniques.

Based on the literature review of the aforementioned research, the study was designed using two types of authentication techniques: face recognition and hand gesture recognition. The system was developed to achieve higher Accuracy and F1-Score values while maintaining low cost.

3. Research Methodology

This section presents the research methodology used in the applied research. It is divided into two main topics: hardware design and software design. The details are as follows.

3.1 Hardware Design

The system uses a Raspberry Pi 4 board with a database in a hard disk drive as the main controller, called the "Process" as shown in Figure 3. The "Input" is a 640x480 pixel camera OKER model177 webcam mounted for face and hand movement recognition and connected to a main controller via the USB port. The "Output" consists of an LCD display and a security door.

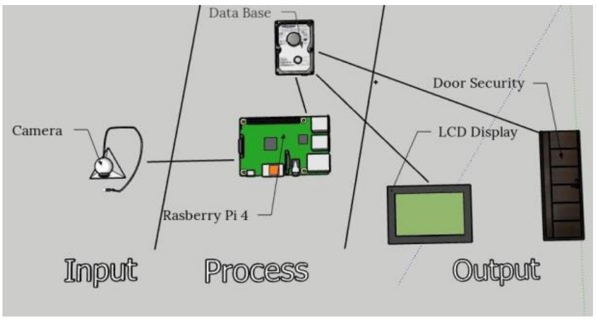


Figure 3 Hardware system overview

3.2 Software Design

This system, designed for real-time operations, uses Python integrated with OpenCV, MediaPipe, and SciPy libraries. OpenCV handles image detection and face processing, MediaPipe facilitates hand gesture detection, and SciPy computes the similarity between detected gestures and database data.

The system software design consists of three algorithms: face detection algorithm, hand gesture detection algorithm, and face and hand gesture recognition algorithm. The details are as follows.

บทความวิจัย (Research Article)

3.2.1. Face detection algorithm

The steps of the face detection algorithm are illustrated in Figure 4 and consist of ten sequential stages:

- Folder Verification: The algorithm begins by verifying the existence of a folder named *saved_faces*, which is designated for storing face images.

- Camera Initialization: The camera is enabled using the *cv2.VideoCapture* function.

- Camera Status Check: The status of the camera is checked using the *video_capture.isOpened* function to ensure proper functionality.

- Image Capture: An image is captured from the camera feed utilizing the *video_capture.read* function.

- Face Detection: The *face_recognition. face_locations* function is employed to detect faces in the image, identifying their coordinates (top, right, bottom, left).

- Border Extension: A border around the detected face is created by extending the face area and adjusting the margin variable.

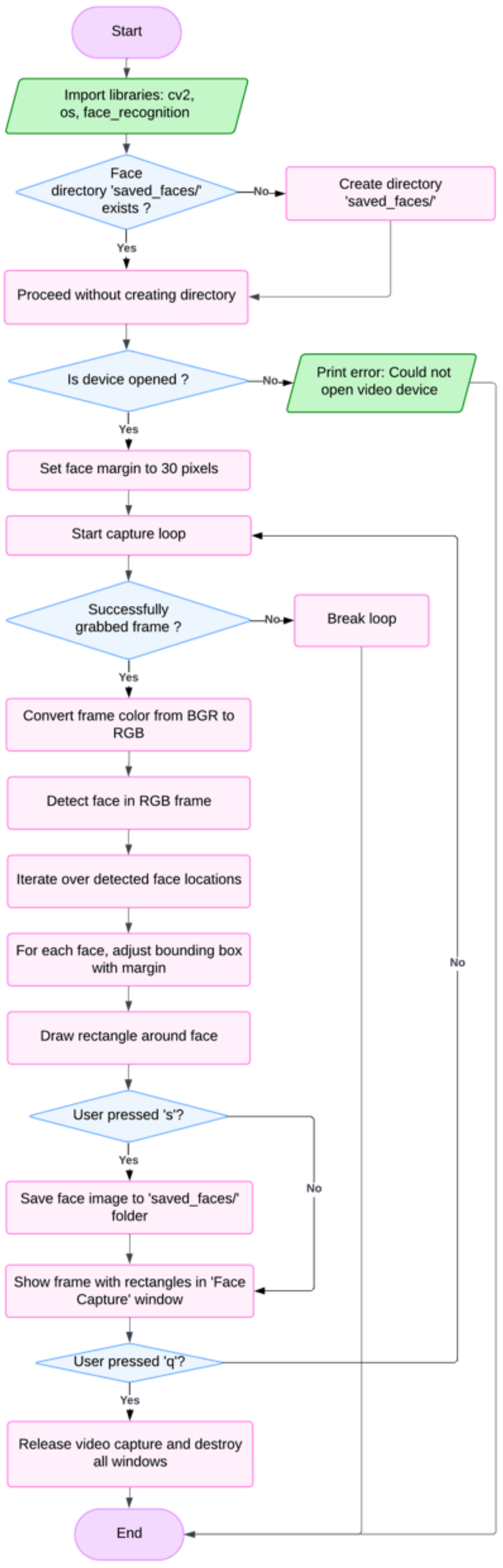

Start
Import libraries: cv2, os, face_recognition
Face directory 'saved_faces/' exists ?
No
Create directory 'saved_faces/'
Yes
Proceed without creating directory
Is device opened ?
No
Print error: Could not open video device
Yes
Set face margin to 30 pixels
Start capture loop
Successfully grabbed frame ?
No
Break loop
Yes
Convert frame color from BGR to RGB
Detect face in RGB frame
Iterate over detected face locations
For each face, adjust bounding box with margin
Draw rectangle around face
User pressed 's'?
Yes
No
Save face image to 'saved_faces/' folder
Show frame with rectangles in 'Face Capture' window
User pressed 'q'?
No
Yes
Release video capture and destroy all windows
End

Figure 4 Face detection algorithm

- Frame Drawing: A visual frame is drawn around the face using the *cv2.rectangle* function.

- Image Saving: The captured face image is saved as a JPG file within the *saved_faces* folder using the *cv2.imwrite* function. This operation is executed only when the user presses the 's' key on the keyboard.

- Image Display: The captured image is displayed on the screen using the *cv2.imshow* function.

- Termination of Process: The camera is released using *video_capture.release*, and all open windows are closed with the *cv2.destroyAllWindows* function. To terminate the operation at any point, the user can press the 'q' key on the keyboard.

3.2.2. Hand gesture detection algorithm

The steps of the hand gesture detection algorithm are illustrated in Figure 5 and consist of ten consecutive steps, as described below:

- Folder Verification: The algorithm begins by verifying the existence of a folder named *saved_faces*, which is designated for storing face images.

- Import Libraries: Import the required libraries, OpenCV and MediaPipe.

- Initialize Hand Detection: Set up hand detection using the MediaPipe Hands function. This involves configuring real-time hand detection with the *static_image_mode* function, enabling detection of both hands using the *max_num_hands* function, and setting the minimum detection accuracy to 0.7 with the *min_detection_confidence* function.

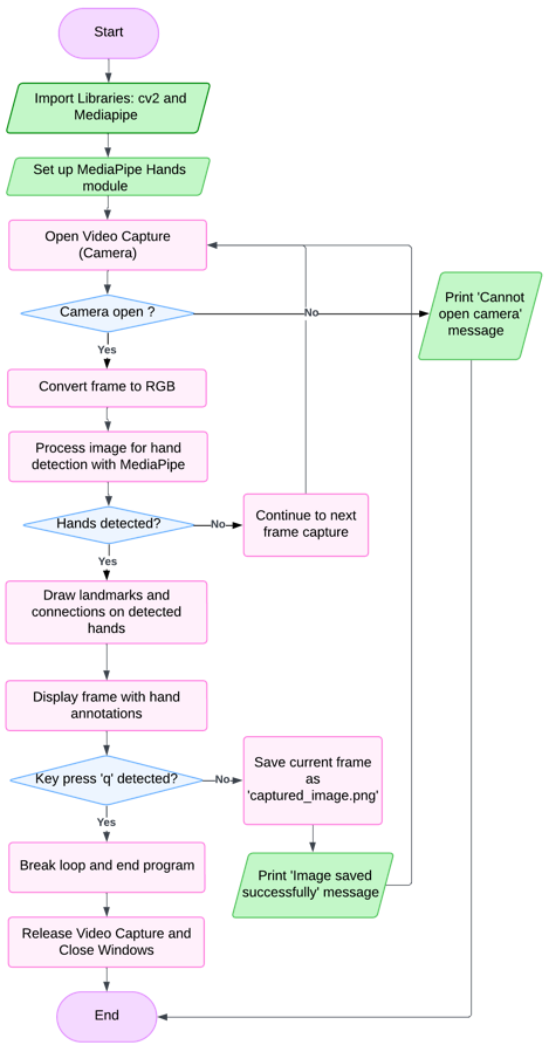

Start
Import Libraries: cv2 and Mediapipe
Set up MediaPipe Hands module
Open Video Capture (Camera)
Camera open ?
No
Print 'Cannot open camera' message
Yes
Convert frame to RGB
Process image for hand detection with MediaPipe
Hands detected?
No
Continue to next frame capture
Yes
Draw landmarks and connections on detected hands
Display frame with hand annotations
Key press 'q' detected?
No
Save current frame as 'captured_image.png'
Print 'Image saved successfully' message
Yes
Break loop and end program
Release Video Capture and Close Windows
End

Figure 5 Hand gesture detection algorithm

- Activate Camera: Open the camera using the *cv2.VideoCapture* function.

- Capture Images: Loop through the camera feed and capture images using the *video_capture.read* function.

- Convert Image Format: Convert the captured image from BGR to RGB format for compatibility with the MediaPipe library, using the *cv2.cvtColor* function.

- Detect Hands: Perform hand detection using the *hands.process(rgb_frame)* function from the MediaPipe library.

- Render Hand Landmarks: Draw landmarks and connecting lines on the detected hand using the *results.multi_hand_landmarks* and *mp_drawing.draw_landmarks* functions from the MediaPipe library.

- Display Output: Manually display the processed image with the *cv2.imshow* function.

- Handle Keyboard Commands: Use the *cv2.waitKey* function to check for keyboard input. If the user presses the 'q' key, exit the loop and terminate the program using the *video_capture.release* and *cv2.destroyAll Windows* functions. If any other key is pressed, save the current image as a PNG file.

3.2.3. Face and hand gesture recognition algorithm

Face recognition algorithms compare facial data from the camera with the database using encoding techniques. Hand gesture recognition relies on landmark point locations and cosine similarity “CosSin” to confirm correct gestures.

$$CosSin = 1 - \frac{A \cdot B}{\|A\|\|B\|} \qquad (1)$$

where A and B are vectors of the relative positions of the landmarks on the hand gesture and the sign. If the dot product of the two vectors is close to 1, then the hand gestures are very similar.

This algorithm uses a database of finger pattern detection criteria, which can be verified by comparing different finger patterns. The finger pattern consists of '1' which is any finger up pattern and '0' which is no finger up pattern. These algorithmic conditions are used to determine if the detected hand gesture matches a gesture recorded in the database, which will result in authorization to unlock the security door system.

Figure 6 illustrates the eight steps of the face and hand gesture recognition algorithm for unlocking the security door.

- Open the camera and load pre-stored face and hand gesture data for recognition, including results from detection algorithms.

- Camera Frame Processing: The algorithm captures real-time images to begin face and hand gesture recognition.

- Face Detection: Using the face recognition library, the system searches for a face. If a face is detected, it compares the data with the face database.

- Face Comparison: If the detected face does not match the data in the database, the system displays an “Unlock failed” message on the screen. If the face matches, the system proceeds to hand gesture detection.

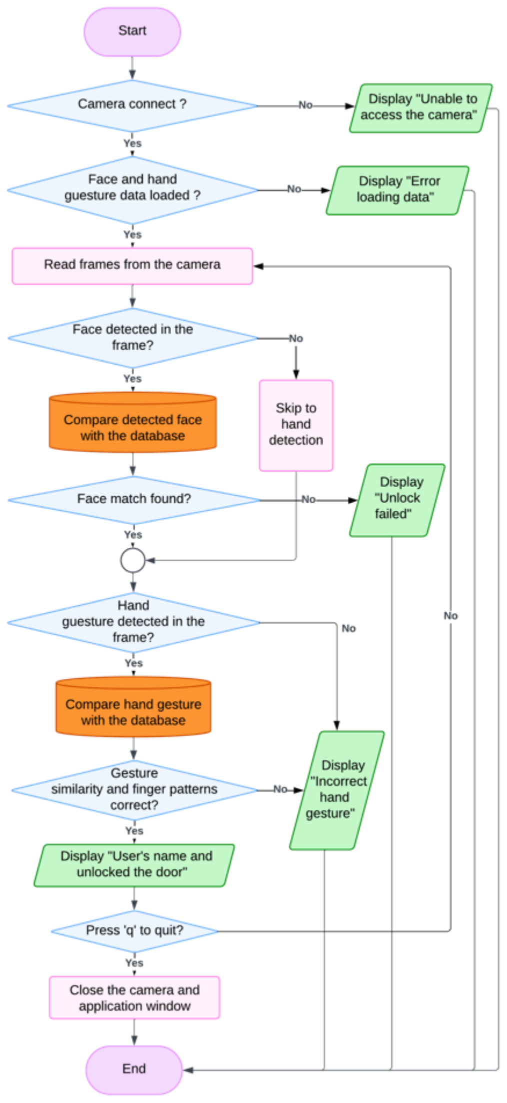

Start
Camera connect ?
No
Display "Unable to access the camera"
Yes
Face and hand guesture data loaded ?
No
Display "Error loading data"
Yes
Read frames from the camera
Face detected in the frame?
No
Yes
Compare detected face with the database
Skip to hand detection
Face match found?
No
Display "Unlock failed"
Yes
Hand guesture detected in the frame?
No
No
Yes
Compare hand gesture with the database
Gesture similarity and finger patterns correct?
No
Display "Incorrect hand gesture"
Yes
Display "User's name and unlocked the door"
Press 'q' to quit?
Yes
Close the camera and application window
End

Figure 6 Face and hand gesture recognition algorithm

- Hand Detection with MediaPipe: During hand gesture detection, if no hand is detected in the image, the system displays an “Incorrect hand gesture” message on the screen. If a hand is detected, it is compared with the data in the database.

- Hand Gesture Comparison: The hand gesture similarity is calculated using the Cosine Similarity function, and the finger pattern is analyzed. If the similarity value is low or the finger pattern is incorrect, the system displays an “Invalid Hand Gesture” message. However, if the similarity value is above 0.93 and the finger pattern is correct, the system allows the security door to be unlocked.

- Unlocking the Security Door: If the user successfully passes both face and hand gesture recognition, the system displays the user’s name, and the message “Unlock” on the screen.

- Stopping the Application: The application closes and turns off the camera when the user presses the 'q' button.

4. Experiments and Setup

This section presents an experiment on face and hand gesture recognition using MediaPipe, as shown in Figure 7. It is divided into three parts: the sample population part, the experimental condition part, and the experimental part. The system is developed on the Raspberry Pi 4 board, which is the main real-time processing device, and the details are as follows:

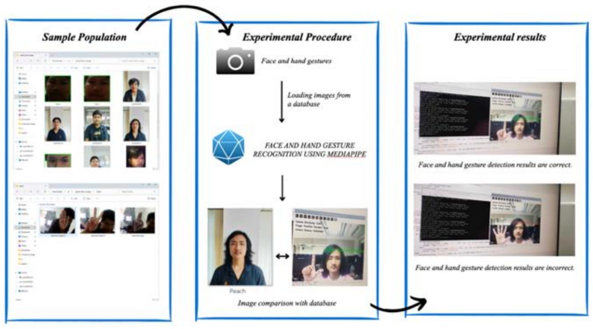

Sample Population
Experimental Procedure
Face and hand gestures
Loading images from a database
FACE AND HAND GESTURE RECOGNITION USING MEDIAPIPE
Peach
Image comparison with database
Experimental results
Face and hand gesture detection results are correct.
Face and hand gesture detection results are incorrect.

Figure 7 Face and hand gesture recognition experiment using MediaPipe

4.1 Data Collection

There were 100 participants from the Department of Computer Engineering at Suan Sunandha Rajabhat University. Data collection is divided into two parts: data types and data grouping.

Two types of data were collected: face images and hand gestures. A total of 500 face images were captured (5 images per participant) under varying illumination conditions and different angles. Additionally, each participant performed 10 dynamic hand gestures (e.g., thumbs-up, peace sign). The data is divided into a training set and a test set. The training set, comprising 70% of the data is used for model training, while the test set, comprising 30% of the data is used for validation.

4.2 Experimental Condition

Experimental conditions include lighting, camera angles, and environmental factors. Light intensity should range from 300 to 500 lux for clear images. The camera should be positioned at face level, with the face 50-100 cm and the hand 30-50 cm from the camera. A plain, patternless background is recommended to minimize detection interference.

4.3 Experimental Procedure

The experimental procedure consists of three steps. Step 1: Select an individual to stand in front of the Raspberry Pi 4 camera for real-time face and hand gesture detection. Step 2: The system compares the detected face and hand gestures with database data using CosSin. Step 3: If the data matches, the system unlocks the security door; otherwise, it displays a warning.

5. Experimental Results

From the experiment using a sample group of 100 people, divided into two equal groups, the experimental results can be divided into two parts: the results of the experiment on recognizing 100 consecutive frames of faces and the results of the experiment on recognizing 100 consecutive frames of hand gestures.

Figure 8 shows the experimental results of face recognition on 100 consecutive face frames. The x-axis represents the face frame number, ranging from 1 to 100, while the y-axis denotes the number of faces detected within each frame, spanning from 0 to 5. A blue line with markers shows the number of faces detected in each frame, providing a clear temporal trend.

A horizontal line at the 1 face threshold serves as a benchmark for system performance. Variations above and below this line indicate fluctuating face detections, ranging from zero to five, reflecting challenges such as illumination changes, occlusion, and varying numbers of visible individuals. The average number of detected faces per frame was 0.98 (±0.12 SD), demonstrating high stability.

Figure 9 shows the experimental results of hand gesture recognition in 100 consecutive frames. The x-axis represents the frame number, ranging from 1 to 100, while the y-axis displays the cosine similarity values ranging from 0 to 1 and the binary finger status ‘1’ for correct gestures and ‘0’ for incorrect gestures.

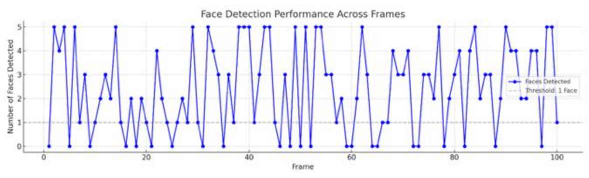


Figure 8 Face detection performance across frames

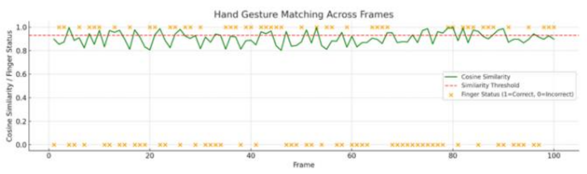


Figure 9 Hand gesture matching across frames

The green line with markers shows the cosine similarity per frame, indicating alignment with the reference gesture. The red line marks the 0.93 similarity threshold for acceptable matches, while orange crosses denote binary correctness: '1' for correct and '0' for incorrect gestures.

The average CosSin score was 0.91 (±0.05 SD) exceeding the 0.93 threshold, indicating generally accurate gesture matching. However, the alternating binary pattern of correct and incorrect classifications suggests that factors beyond similarity scores influence gesture recognition.

Table 2 Summary of experimental results

Metric	Authorized Users	Unauthorized Users	Overall
Success Rate	95%	95%	95%
Mean Detect Time	1.2s (±0.3s SD)	1.5s (±0.4s SD)	1.35s
Accuracy	-	-	95%
Precision	94%	96%	-
Recall	96%	94%	-
F1-Score	95%	95%	-

Table 2 presents the experimental results, showing a consistent 95% success rate for both authorized and unauthorized users. The average detection time was 1.2s (±0.3s SD) for authorized users and 1.5s (±0.4s SD) for unauthorized users, with an overall mean of 1.35s. The system achieved 95% accuracy, with precision and recall values above 94% for both user groups. The F1-score was consistently 95%, indicating balanced and reliable real-time classification performance.

6. Conclusion

This paper introduces a face and hand gesture recognition system built using MediaPipe. A highlight of this research is the implementation of a dual authentication approach using the cosine similarity technique with OpenCV and SciPy for security systems, deployed on a Raspberry Pi 4 board equipped with a webcam. The study involved experiments with a sample of 100 university students. Results indicate that the system achieves a success rate of approximately 95% in unlocking a security door for students whose information is stored in the database, while effectively blocking unauthorized access by individuals without matching data. However, the system faces limitations related to lighting conditions, which in the experiments were set at 300-500 lux and the camera angle positioned at face level. When compared with studies [2–3], this research shows an improvement in Accuracy and F1-Score, reaching 95%.

Future work will evaluate the system in real-world scenarios, focusing on refining thresholding and adding features to improve accuracy and robustness in face and hand gesture recognition.

7. Acknowledgement

This research was supported by the Department of Computer Engineering, Suan Sunandha Rajabhat University, along with Mr. Peeradet Tanawiwat for their assistance in testing the system developed in this work.

8. References

[1] K. Suphalak, N. Klanpet, N. Sikaressakul, and S. Prongnuch, "Robot Arm Control System via Ethernet with Kinect V2 Camera for use in Hazardous Areas," in *Proc. 1st Int. Conf. Robot., Eng., Sci., and Technol.*, 2024, pp. 175-180. DOI: 10.1109/RESTCON60981.2024.10463582.

[2] R. Jeyaraj, B. Subramanian, K. Yogesh, A. Jin, and H. A Gohel, "YSAF: Yolo with Spatial Attention and FFT to Detect Face Spoofing Attacks," in *Proc. IEEE 3rd Int. Conf. AI in Cybersecurity*, 2024, pp. 1-6. DOI: 10.1109/ICAIC60265.2024.10433802.

[3] N. Herbaz, H. El Idrissi, and A. Badri, "Deep Learning Empowered Hand Gesture Recognition: using YOLO Techniques," in *Proc. 14th Int. Conf. Intell. Syst.: Theor. and Appl.*, 2023, pp. 1-7. DOI: 10.1109/SITA60746.2023.10373734.

[4] Y. Kale, N. Sakhare, M. Varghese, U. Dafe, P. Sawarkar, and A. Bagade, "Deploying a Sliding Gate by Utilizing Computer Vision and Machine Learning Techniques," in *Proc. Int. Conf. Innov. Data Commun. Technol. and Appl.*, 2023, pp. 22-26.

DOI: 10.1109/ICIDCA56705.2023.10099906.

[5] A. K. Raja, C. Sugandhi, G. Nymish, N. Sai Havish, and M. Rashmi, "Face Gesture Based Virtual Mouse Using Mediapipe," in *Proc. IEEE 8th Int. Conf. for Convergence in Technol.*, 2023, pp. 1-6. DOI: 10.1109/I2CT57861.2023.10126453.

[6] Google, "MediaPipe Solutions Guide," Google AI, [online]. Available: https://ai.google.dev/edge/mediapipe/solutions/guide. [Accessed: Nov. 24, 2024].

[7] OpenCV, "About OpenCV," OpenCV, [online]. Available: https://opencv.org/about/. [Accessed: Nov. 22, 2024].

[8] T. Wiangtong and S. Prongnuch, "Computer vision framework for object monitoring," in *Proc. 9th Int. Conf. Elect. Eng./Electron., Comput., Telecommun. and Inf. Technol.*, 2012, pp. 1-4. DOI: 10.1109/ECTICon.2012.6254291.

[9] SciPy Community, "About SciPy," SciPy, [ออนไลน์]. Available: https://scipy.org/about/. [Accessed: Nov. 27, 2024].

[10] J R. V. Jeny, B. Shraddha, B. Ashritha, D. Shiva Sai, and M. Naveen, "Deep Learning Framework for Face Mask Detection," in *Proc. 5th Int. Conf. Trends in Electron. and Inf.*, 2021, pp. 1705-1712. DOI: 10.1109/ICOEI51242.2021.9452930.

[11] Raspberry Pi Foundation, "Raspberry Pi 4 Model B," Raspberry Pi, [online]. Available: https://www.raspberrypi.com/products/raspberry-pi-4-model-b/. [Accessed: Nov. 20, 2024].

[12] S. Prongnuch and S. Sitjongsataporn, "Technology-assisted Learning on Embedded Systems with Multi-single Board," in *Proc. Int. Conf. Power, Energy and Innov.*, 2021, pp. 191-194. DOI: 10.1109/ICPEI52436.2021.9690651.

[13] T. Hild, P. Moore, and M. Powell, "Multimodal Authentication," in *Proc. Annual General Donald R. Keith Memorial Conf.*, May 2, 2024. [online]. Available: