Implementation of Hand Gesture Recognition as Smart Home Devices Controller

Stanley A. Dewangga[1], Mochamad Subianto[1], and Windra Swastika[1]
[1]Informatics Department, Faculty of Technology and Design, Ma Chung University, Malang, Indonesia

Corresponding author: Stanley A. Dewangga (e-mail: stanleyadidewangga@gmail.com).

ABSTRACT Some current virtual assistant products such as Alexa, Siri and Google Home facilitate features to control smart home devices through voice input, which has become increasingly popular in recent years. In addition to voice input, smart home devices can also be monitored and controlled through smartphones or computers using applications that provide users with flexibility. However, both control systems are less efficient, as they consume time and voice input utilization may sometimes not be recognized in crowded conditions. Therefore, this research introduces an application to recognize real-time hand gestures and utilize them for a new control system that provides time and energy efficiency. This application processes images using the Mediapipe framework, generating hand landmark outputs. These landmark outputs are utilized to determine the combination of raised or lowered fingers, which is then used to control smart home devices. The application is developed with ESP32 and ESP01s modules as data receivers from gesture recognition, and ESP32-CAM for image acquisition. Meanwhile, the gesture recognition computation process is executed on a Raspberry Pi 3 Model B. The gesture recognition application achieves good accuracy at 88%, but may experience occasional failures for certain gestures. However, the response time generated by the smart home control system is still relatively long, averaging 7.88 seconds.

KEYWORDS Hand Gesture Recognition, Hand Landmark, Mediapipe, Smart Home

I. INTRODUCTION

This Smart home is a system designed to control and automate household electronic devices intelligently using integrated sensors. There are numerous benefits offered by smart home devices in providing solutions for both routine and specific human needs. Research on the control system of smart home devices for individuals with physical disabilities [1], [2] states that smart home technology facilitates physical disabilities in managing their household devices.

Control over smart home devices can be achieved through the use of current virtual assistant products [3], [4]. However, according to Mtshali & Khubisa (2019) controlling smart home devices using virtual assistants is a complex, complicated, and expensive solution for individuals with physical disabilities or the elderly. These virtual assistants are also vulnerable to spoofing attacks [5]. In a study on the design of a smart home as an IoT application based on Voice Recognition and Arduino [6], failures were also found in recognizing voice input instructions as controllers for smart home devices. Voice recognition failures can be time-consuming, making it less efficient.

In noisy environmental conditions, various noises can interfere the voice recognition capabilities of virtual assistants. These noises typically originate from sounds not intended to control smart home devices. This issue can hinder the smooth operation of control systems and also result in time consuming.

Research on hand gesture recognition to control household electronic devices was conducted [7]. The research design utilized IMU sensors to obtain accelerometer data, which was processed and classified into 4 hand gestures: up, down, left, and right. The results of gesture recognition were then used to trigger the control of electronic devices in the home.

Another study on smart home control devices using hand gesture recognition was also carried out [8]. With depth camera acquisition and the application of the Hidden Markov Model (HMM) method for gesture recognition, an average recognition rate of 98.50% was achieved. However, the control device was limited to only 4 types of hand gestures.

In previous studies [9], the gesture recognition system that utilized input from the ADPS-9960 sensor was limited to basic swipe gestures: left, right, up, and down. Instructions were executed using upward and downward hand

movements to navigate menus on an LCD. This research provides a solution to the time consuming and labor-intensive process of menu selection. By employing camera-based gesture recognition, this approach overcomes the limitations in the variety of gestures that can be recognized compared to those with the ADPS-9960 sensor.

Therefore, this research introduces a smart home control device utilizing hand gesture recognition as its trigger. The utilization of hand gesture recognition is expected to address various issues found in previous research. This approach is also giving more advantage as it does not need any sensors or attachments on our hands, offering more seamless and convenient user experiences [10], [11]. Instead of relying on traditional manual switches or remote controls, users can execute simple gestures to operate lighting, temperature settings, and other devices, allowing for quicker and more intuitive control over home automation systems [12]. This method reduces the need for physical contact and movement, saving energy typically used by conventional methods and diminishing the time taken to perform routine tasks.

II. APPLICATION DESIGN

As shown in Figure 1, the application flow begins with the image acquisition process. In real-time image processing, the term "image" refers to the frame unit acquired using the camera. This image is then used as input for the hand gesture recognition model. The output from hand gesture recognition is subsequently employed to trigger the control of smart home devices.

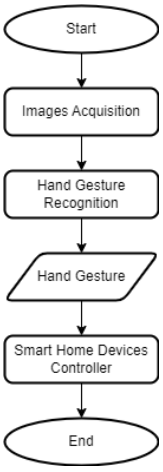

Figure 1. Application Flowchart

A. IMAGE ACQUISITION

Image acquisition is performed on the ESP32-CAM using the OV2640 camera module. Upon activation, the ESP32-CAM will automatically create a web server and send real-time image data to the web server. The process of sending image data to the web server and creating the web server itself utilizes the built-in code or sketch from the ESP32 add-on in the Arduino IDE called "cameraWebServer". On this web server, the resolution and image quality can be adjusted, and there are also features to apply effects to the image, such as Grayscale, Binary, and others.

B. HAND GESTURE RECOGNITION

The hand gesture recognition process is executed by inferencing each frame captured by the camera in real-time into the Mediapipe model. Mediapipe is a versatile framework developed by Google, provides a suite of image processing functions that facilitate the accurate detection and recognition of hand gestures as they occur. This real-time processing allows the system to provide instantaneous feedback based on the gestures it recognizes.

Before hand gestures can be recognized, the input of the hand image is initially detected using the Palm Detection model from Mediapipe as shown in Figure 2. This model is created using the Single Shot Detector (SSD) architecture and Convolutional Neural Network (CNN) method [13]. The output of the detection is a bounding box stored in the values of x-minimum, y-minimum, x-maximum, and y-maximum, representing the coordinate values of the detection box on the image.

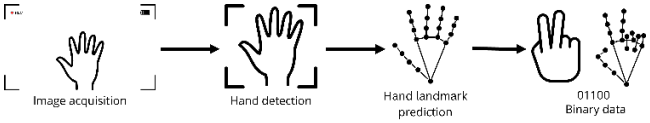

Figure 2. Hand Gesture Recognition Flowchart

The image is then cropped based on the bounding box values and enters the hand landmark prediction model. Hand landmark refers to the points of joints or the framework of the palm, as shown in Figure 3. For example, INDEX_FINGER_MCP represents the coordinate point of the metacarpophalangeal (MCP) joint on the index finger. The output of the hand landmark prediction utilized for the subsequent process includes the x and y coordinates of each landmark on the image.

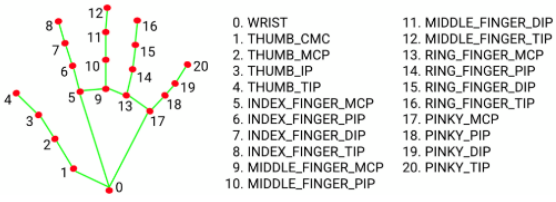

Figure 3. Hand Landmarks

The landmark coordinates in Figure 3 are utilized to determine whether each finger is raised or lowered. For example, if the y-coordinate of the finger's TIP in the image is smaller or positioned above the MCP, the finger is interpreted as raised. Each raised finger is assigned a value of 1, while the lowered ones are assigned a value of 0. These values of 0 and 1 represent the raised and lowered states of the 5 fingers on both hands, as illustrated in Figure 4. By utilizing these values for the 5 fingers, a total of 2^5 combinations or 32 types of static hand gestures can be

formed. In this study, only 5 of the 32 hand gestures were tested to measure the response time of the system.

Figure 4. Gesture Illustration

C. SMART HOME DEVICES CONTROLLER

In Figure 5, the ESP32-CAM is connected to the access point created by the Raspberry Pi and sends image data to the web server it establishes. Subsequently, the Raspberry Pi accesses this data through the web server and uses it as input for the gesture recognition model. The Raspberry Pi serves as the main data processing center, handling image processing, inferring to the gesture recognition model, and providing output data in the form of gesture information. Additionally, the Raspberry Pi opens an access point, allowing other components to connect to it.

The Raspberry Pi establishes a web server so that after the image or frame is inferred through the model, the output results from the model are sent to the web server created by the Raspberry Pi. The inference process and web server operation are executed concurrently using the threading library in the Python programming language, while the creation of the web server utilizes the Flask library. Both processes need to run simultaneously for the web server to operate and update data while the Raspberry Pi processes image data. The Gesture Output on the web server is accessed by the ESP-01S and ESP32, serving as triggers to control devices.

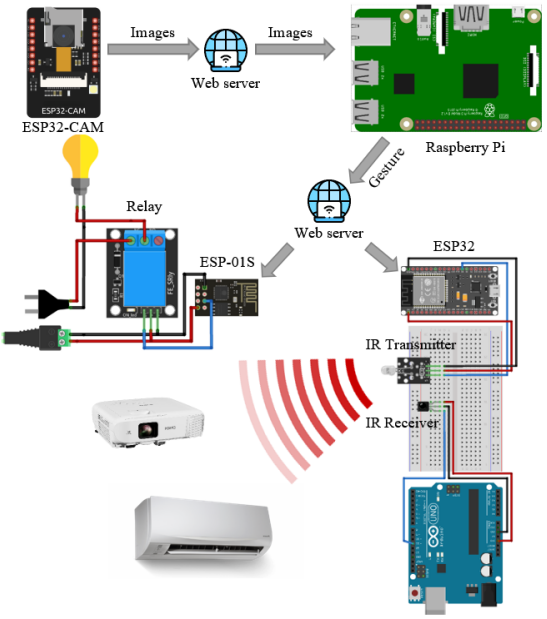

Figure 5. System Illustration

1) CONTROLLER USING RELAY

The ESP-01S is utilized to access gesture data and control the relay based on the received gesture information. The ESP-01S is connected to the Raspberry Pi's access point and retrieves gesture data from the Raspberry Pi's web server. If the gesture data corresponds to instructions to turn on or off the light, the relay will operate accordingly by opening or closing the electrical circuit. Figure 6 illustrates the configuration scheme of the controller using a relay and ESP-01S.

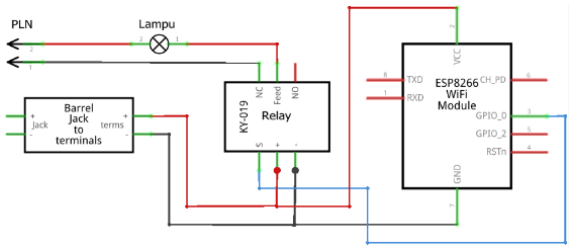

Figure 6. Installation Scheme of ESP01S and Relay

2) CONTROLLER USING IR TRANSMITTER

One limitation of the relay in this application is its capability to only open and close the electrical circuit, allowing instructions for turning on or off only. To address this issue, an infrared (IR) transmitter is employed. Various household electronic devices such as ACs, TVs, and projectors use infrared remote controls to execute additional instructions like adjusting temperature, changing channels, and controlling volume. By replicating these functions, the given instructions can be diverse and not limited to just on or off. However, the use of the infrared transmitter is constrained by its limited range, directional emission that must be in line, and its inability to penetrate objects.

To mimic the operation of a remote control, the infrared signals emitted by the remote based on instructions are recorded and initially noted using the IR Receiver HS1838 module. Remote instruction signals are in hexadecimal form. For example, the signal to turn on the AC is '81C08F70 C1AA09F6'. Each hexadecimal character represents 4 bits of digital signal transmitted at a frequency of 38 kHz. For instance, the character 'A' represents the value '0101,' while '9' represents the value '1001' in the digital signal. When the infrared is emitted from the remote, the signal received by the receiver or AC is its inverse. For example, if the transmitted signal value is '11001001,' the signal received by the receiver is '10010011.' Therefore, after recording the code signal for each instruction, the code is reversed or converted. If the initially obtained signal code was '81C08F70 C1AA09F6' during recording, the original signal is '6F905583 0EF10381.' This original signal is what the infrared transmitter will emit when intending to turn on the AC.

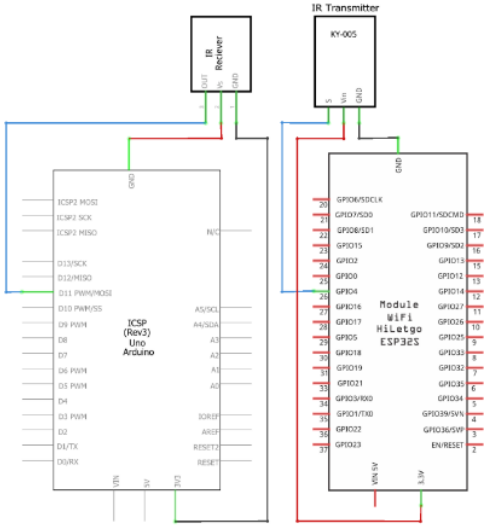

Figure 7. Installation Scheme of IR Transmitter and Receiver

The Infrared transmitter is connected to the ESP32. The ESP32 is linked to the Raspberry Pi's access point connection to access gesture data from the Raspberry Pi's web server. When the gesture data meets the conditions to execute specific instructions, the infrared transmitter will emit an infrared signal to the target device and execute instructions based on the hexadecimal code of the previously recorded original signal. As illustrated in Figure 7, there is an IR receiver connected to Arduino UNO to validate or crosscheck the signal code emitted by the transmitter, ensuring there are no errors in the code.

D. APPLICATION TESTING

The testing phase involves designing a prototype simulation of the application's operation. The simulation is created on a smaller scale compared to a typical smart home system, with the aim of depicting the real system's conditions. Additionally, this simulation is used to test the solutions implemented for previously identified issues and the smooth functioning of additional features. Two aspects are tested in the formed simulation: response time testing and gesture recognition accuracy testing.

Response time testing involves recording the time from when a gesture is recognized on the Raspberry Pi until the instructed device receives a response. The testing is conducted 10 times for each gesture and measured in seconds. Meanwhile, accuracy is tested using a Confusion Matrix. A Confusion Matrix is a method for calculating accuracy, precision, and recall values in a classification system [14].

		Prediction	
		1	0
Actual	1	TP	FN
	0	FP	TN

Figure 8. Confusion Matrix Example

There are four terms to calculate performance in the Confusion Matrix, namely TP, FN, FP, and TN as shown in Figure 8. TP or True Positive is the number of data predicted as class 1, and the actual result is also class 1. FN or False Negative is the number of data predicted as class 0, but the actual result is class 1. FP or False Positive is the number of data predicted as class 1, but the actual result is class 0. Meanwhile, TN or True Negative is the number of data predicted as class 0, and the actual result is also class 0 [15]. Equation (1) is used to measure the number of cases predicted correctly compared to the overall cases. Equation (2) is a measurement that indicates how accurate the prediction is for the positive class. The precision value is obtained from the ratio of True Positive to the total positive cases. Equation (3) is used to measure the ratio of True Positive to the total actual positive cases.

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (1)$$

$$\text{Precision} = \frac{TP}{FP+TP} \quad (2)$$

$$\text{Recall} = \frac{TP}{FN+TP} \quad (3)$$

III. RESULT AND DISCUSSIONS

The camera resolution for image acquisition is set to VGA resolution, which is 640x480 pixels. The resulting frames are obtained in the .JPEG file format with a file size ranging from approximately 8000 to 12,000 bytes or 8-12 KB. Raspberry Pi receives frame data from ESP32-CAM with an average frames per second (fps) of 52.6, using 2 frame buffers. Frame buffer refers to additional memory for storing graphic information that has not been displayed on the screen. The image data is then sent to the web server created by ESP32-CAM. This data can be accessed using the IP address of ESP32-CAM.

TABLE I
INFRARED SIGNAL RESULTS

Devices	Command	Signal Code	Converted Signal
Projector	Power On/Off	81C08F70 C1AA09F6	03810EF1 55836F90
	Freeze/ Unfreeze	81C08F70 C1AA49B6	03810EF1 55836D92
AC	Power On	96BFBCDC CEC65E15	FD693B3D 6373A87A
	Power Off	96BFBCDC 9A664E43	FD693B3D 6659C272

Table 1 illustrates the infrared signals successfully recorded using an IR Receiver and emitted using an IR Transmitter. Some signals from different brand AC remotes were unable to be recorded because the hexadecimal codes emitted for one instruction and one device varied. To address this issue, attempts were made to record the pulse width and retransmit it, but the results were still unsuccessful.

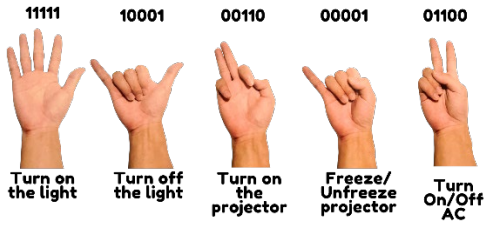

Figure 9. **Hand gestures and its commands**

With the developed application, five recognizable hand gestures have been formed to execute various instructions. Among the executable instructions are turning on and off the lights, projector, AC, and freezing/unfreezing the projector, as illustrated in Figure 9. The number 1 represents raised fingers from right to left on the right hand; for example, the gesture data '01100' signifies that the index and middle fingers are raised. The output of this gesture data is sent through the web server within the local network, making it accessible to both ESP-01s and ESP32. Figure 10 demonstrates an example of using the application for gestures 11111 and 10001. The executed instructions for the performed gestures were successful.

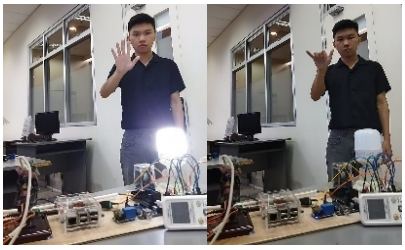

Figure 10. **Example of Application Usage**

Testing was conducted under specified camera resolution, distance, and lighting conditions. The distance between the camera and the hand was set at 1 meter. Meanwhile, the lighting conditions were measured using a lux meter to measure the light intensity. Measurements recorded in the area where hand movements were tested showed a light intensity of 245 lux.

Variations in lighting conditions, either darker or brighter, can significantly affect the detail captured in the hand's image, thereby making gestures more difficult to recognize. The distance between the hand and the camera also plays a crucial role for the system to detect hand gestures. Therefore, further testing on both lighting and distance factors is necessary to optimize the recognition system's performance.

A. ACCURACY TESTING RESULTS

As shown in Figure 11, the hand gesture recognition system achieved a good accuracy of 88%. However, the gesture 00110 frequently experiences prediction failures and is often misclassified as 01100, with a precision value of 0.6. This occurrence may be attributed to instances where the hand is far from the camera and has a relatively low resolution. In such cases, the texture or shape of the hand may not be clearly recognizable, leading to inaccurate landmark outputs. Some similar gestures can be recognized accurately at short distances to handle misclassification, but this may reduce the user experience. Implementing higher-resolution cameras that capture more detailed images could improve the system's ability to recognize differences between similar gestures at long distances.

Prediction

Actual	11111	10001	00110	00001	01100	Precision
11111	10	-	-	-	-	1
10001	-	10	-	-	-	1
00110	-	-	6	-	4	0.6
00001	-	2	-	8	-	0.8
01100	-	-	-	-	10	1
Recall	1	0.83	1	1	0.71	
Accuracy	0.88					

Figure 11. **Hand gestures and its commands**

B. RESPONSE TIME TESTING RESULTS

The response speed performance in controlling smart home devices is still below average. The average time required from gesturing to the camera until the smart home device lights up entirely is 7.88 seconds. The average response time between using relay and infrared also does not indicate a significant difference, with a minimal average speed difference (0.28 seconds) and a comparable difference in standard deviation (0.07 seconds).

TABLE II
RESPONSE TIME TESTING RESULTS

Response time in seconds					
No	**11111 (Relay)**	**10001 (Relay)**	**00110 (IR)**	**00001 (IR)**	**01100 (IR)**
1	9.62	10.08	11.03	9.56	11.43
2	10.63	9.89	9.58	13.42	10.24
3	11.28	10.8	9.74	9.09	8.4
4	10.42	9.21	10.27	7.79	7.97
5	7.98	8.09	9.62	11.04	08.87
6	5.2	5.27	7.79	6.62	5.98
7	5.78	6	6.68	5.65	4.82
8	5.95	5.31	5.83	6.47	6.71
9	6.97	5.15	5.87	5.4	6.09
10	5.83	5.35	5.17	6.42	7.18
Mean	7.74		8.02		
	7.88				
Standard Deviation	2.27		2.2		
	2.24				

The generated deviation indicates that the speed performance of the smart home controller is still fluctuating. As observed in Table 2, numbers 1 to 5 and 6 to 10, the

resulting response times show a significant difference with an average gap of 3.86. When the program is running, the number of frames per second captured by the Raspberry Pi drastically decreases from the initial 52.6 fps to around 1 to 3 fps. This is because each frame captured by the camera is processed and inferred into the model before capturing the next frame. Therefore, the computational speed of the Raspberry Pi in processing and inferring images into the model remains inconsistent and inefficient.

IV. CONCLUSION

The smart home device control system developed has an average overall response time of 7.88 seconds from 10 tests for each gesture. This response time is still too long if applied in product form. The prolonged response time is due to the computational limitations of the Raspberry Pi 3 Model B in handling images and inferring them into the gesture recognition model. Therefore, this research still cannot solve the issues of time and energy efficiency identified in the previous study. However, this research has the potential to be improved and can address these issues by changing the central computing unit to another small computer with higher computational speed.

The hand gesture recognition system achieved an overall accuracy of 88%, which is satisfactory. However, some gesture recognition errors still occur, and certain gestures remain challenging to use. The system can recognize up to 32 gestures, but this study only utilized 5, indicating that the limitations in the number of recognized gestures in the previous research can be addressed with some improvements in this study.

Based on the experiments and tests conducted, there are several suggestions that can be applied to enhance and further develop this research, including:

1. Replacing the Raspberry Pi 3 Model B as the main computing unit with a faster computer for image processing. This decision should also consider the amount of input to be processed on that computing resource, as multiple inputs may be required for more than one room.
2. Considering the option of not running gesture recognition in real-time but providing an interface feature for gesture recording mode, making the power consumption more efficient.
3. Implementing a feature to record and store infrared signals from the original remote and being able to emit them when specific instructions are called.

AUTHORS CONTRIBUTION

Stanley Adi Dewangga: Writing original draft, conceptualization, methodology, editing writing, software, validation, and data curation.

Mochamad Subianto: Supervision, analysis, investigation, resources, system validation;

Windra Swastika: Supervision, methodology, conceptualization, system validation, investigation;

COPYRIGHT

REFERENCES

[1] P. Mtshali and F. Khubisa, “A Smart Home Appliance Control System for Physically Disabled People,” in *2019 Conference on Information Communications Technology and Society (ICTAS)*, IEEE, Mar. 2019, pp. 1–5. doi: 10.1109/ICTAS.2019.8703637.

[2] A. Mujahid *et al.*, “Real-Time Hand Gesture Recognition Based on Deep Learning YOLOv3 Model,” *Applied Sciences*, vol. 11, no. 9, p. 4164, May 2021, doi: 10.3390/app11094164.

[3] V. Kĕpuska and G. Bohouta, “Next-generation of virtual personal assistants (Microsoft Cortana, Apple Siri, Amazon Alexa and Google Home),” in *2018 IEEE 8th Annual Computing and Communication Workshop and Conference (CCWC)*, 2018, pp. 99–103. doi: 10.1109/CCWC.2018.8301638.

[4] S. P. Yadav, A. Gupta, C. Dos Santos Nascimento, V. de Albuquerque, M. S. Naruka, and S. Singh Chauhan, “Voice-Based Virtual-Controlled Intelligent Personal Assistants,” in *2023 International Conference on Computational Intelligence, Communication Technology and Networking (CICTN)*, 2023, pp. 563–568. doi: 10.1109/CICTN57981.2023.10141447.

[5] Z. Wu, N. Evans, T. Kinnunen, J. Yamagishi, F. Alegre, and H. Li, “Spoofing and countermeasures for speaker verification: A survey,” *Speech Commun*, vol. 66, pp. 130–153, 2015, doi: https://doi.org/10.1016/j.specom.2014.10.005.

[6] N. R. Adhinugroho and H. P. Uranus, *Perancangan Rumah Cerdas sebagai Aplikasi IoT Berbasis Voice Recognition dan Arduino [Voice Recognition and Arduino Based Smart Home Design as Application of IoT]*. 2019.

[7] H. Basanta, Y.-P. Huang, and T.-T. Lee, “Assistive design for elderly living ambient using voice and gesture recognition system,” in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, Oct. 2017, pp. 840–845. doi: 10.1109/SMC.2017.8122714.

[8] T.-S. Dinh Dong-Luong and Kim, “Smart Home Appliance Control via Hand Gesture Recognition Using a Depth Camera,” in *Smart Energy Control Systems for Sustainable Buildings*, C. and H. R. J. and J. L. C. Littlewood John and Spataru, Ed., Cham: Springer International Publishing, 2017, pp. 159–172.

[9] R.-J. Wang, S.-C. Lai, J.-Y. Jhuang, M.-C. Ho, and Y.-C. Shiau, “Development of Smart Home Gesture-based Control System,” *Sensors and Materials*, vol. 33, no. 10, p. 3459, Oct. 2021, doi: 10.18494/SAM.2021.3522.

[10] J. Dai, “Gesture Recognition Based Smart Home Control System,” 2020.

[11] P. Vogiatzidakis and P. Koutsabasis, “Mid-Air Gesture Control of Multiple Home Devices in Spatial Augmented Reality Prototype,” *Multimodal Technologies and Interaction*, vol. 4, p. 61, Aug. 2020, doi: 10.3390/mti4030061.

[12] A. S. Bankar, A. D. Harale, and K. J. Karande, “Gestures Controlled Home Automation using Deep Learning: A Review,” *International Journal of Current Engineering and Technology*, vol. 11, no. 06, pp. 617–621, Dec. 2021, doi: 10.14741/ijcet/v.11.6.4.

[13] F. Zhang *et al.*, “MediaPipe Hands: On-device Real-time Hand Tracking,” *CoRR*, vol. abs/2006.1, 2020, [Online]. Available: https://arxiv.org/abs/2006.10214

[14] B. P. Pratiwi, A. S. Handayani, and S. Sarjana, “Pengukuran Kinerja Sistem Kualitas Udara dengan Teknologi WSN menggunakan Confusion Matrix,” *Jurnal Informatika Upgris*, vol. 6, no. 2, Jan. 2021, doi: 10.26877/jiu.v6i2.6552.

[15] R. B. Widodo, “Confusion matrix,” in *Metode k-Nearest Neighbors Klasifikasi Angka Bahasa Isyarat*, Malang: Media Nusa Creative, 2022, ch. 3, pp. 21–23.