ADMI IJST International Journal Science and Technology

OPEN ACCESS

IJST Vol 4 No. 2 | July 2025 | ISSN: 2828-7223 (print), ISSN: 2828-7045 (online), Page 168-176

# Smart Presentation Control Using Hand Gesture Recognition

**Lilis Setyowati[1*], Hans Naufal Granito[2]**
[1,2]Informatics, Industrial Technology, Universitas Gunadarma, Indonesia

**Article History**
Received : July 08th, 2025
Revised : July 19th, 2025
Accepted : July 21st, 2025
Published : July 31st, 2025

**Corresponding author*:**
lisetyo@staff.gunadarma.ac.id

**Cite This Article:**
Setyowati, L., & Hans Naufal Granito. (2025). Smart Presentation Control Using Hand Gesture Recognition. *International Journal Science and Technology*, *4*(2), 168–176.

**DOI:**
https://doi.org/10.56127/ijst.v4i2.2244

**Abstract:** *Presentation is an activity conducted to express opinions in public, aimed at decision-making or providing new insights on a particular matter. During presentations, audience engagement often suffers due to several disruptions faced by the speaker, such as having to pause to change slides, mismatched transition timing by the moderator when changing slides, and other issues. In order to avoid these distractions, Smart Presentation Feature is designed to provide an effective presentation experience. This feature runs on Windows operating system and is developed using MediaPipe, OpenCV, PyAutoGUI, win32gui module, PyInstaller, and a virtual environment using Miniconda. This feature works by reading the user's hand gesture to determine the control actions for the presentation. According to a user satisfaction questionnaire consisting of 11 questions, each with 5 criteria ranging from "Strongly Disagree" to "Strongly Agree", The final score was 89.26% in total, which translates to "Strongly Agree". This signifies that users are satisfied with the Smart Presentation Feature.*

**Keywords**: *Artificial Intelligence, Hand Gesture Recognition, Mediapipe, OpenCV, Presentation*

## INTRODUCTION

Artificial Intelligence (AI) is a technology that simulates human thinking and applies it to machines, enabling them to process information and make decisions similar to how humans do (Saini, 2023; Khan, 2021). One of the emerging applications of AI is hand gesture recognition, which is part of the field of computer vision (IBM, n.d.; Afni, Silmina, & Pangestu, 2021).

Hand gesture recognition can be utilized in presentation activities to help presenters effectively deliver ideas, viewpoints, or information to the audience. This can ultimately lead to conclusions or new perspectives on a given topic (Budiman, Lestanti, Evvandri, & Putri, 2022).

To enhance the quality of presentations, they need to be made more interactive and engaging — not only in terms of content and delivery but also from a technical perspective, by minimizing potential disruptions. Common issues include the need for the presenter to pause briefly to change slides, mismatched timing between the presenter and the moderator when switching slides, and other technical disturbances.

One potential solution to address these problems is the development of a Smart Presentation Feature that utilizes hand gesture recognition technology (Sruthi & Swetha, 2023). This feature can be built using the Python programming language (Donaldson, n.d.), supported by libraries such as OpenCV and MediaPipe (MediaPipe, n.d.; Mahesh, Reddy, Reddy, & Reddy, 2022), PyAutoGUI, PyInstaller, and the win32gui module (Wuxc, n.d.). With this feature, the presenter can switch slides simply by performing specific hand gestures, eliminating the need to physically interact with a keyboard or mouse. The system detects the gesture and translates it into presentation control commands.

## RESEARCH METHOD

The research method for developing the Smart Presentation Feature is divided into several stages: planning, analysis, design, implementation, and testing. The planning stage involves collecting relevant information to support the development of the feature, sourced from various types of literature such as books, journals, and websites (Humaira, 2023).

The analysis stage consists of two parts: hardware requirements analysis and software requirements analysis. The design stage involves the creation of a flowchart, which serves as a reference for determining the program's workflow. The implementation stage is the process of translating the design into actual code to build the intended feature.

The Smart Presentation Feature utilizes several libraries such as OpenCV and MediaPipe for hand gesture detection (Budiman et al., 2022; Sruthi & Swetha, 2023). These libraries enable accurate landmark recognition to interpret hand movements and gestures (MediaPipe, n.d.).

### Hardware Requirements Analysis

The Smart Presentation Feature was developed using hardware with the following specifications: the system model used is an **Asus TUF Gaming F15 FX506LH**, equipped with an **Intel(R) Core (TM) i7-10860H** processor. The system also includes a **512 GB hard drive**, **16 GB of RAM**, and operates on the **Windows 11** operating system.

### Software Requirements Analysis

The development of the Smart Presentation Feature utilizes several software tools, including **Visual Studio Code**, **Miniconda**, **Python version 3.10.14**, **PIP**, and several Python libraries:

- opencv-python version 4.9.0 (Mahesh et al., 2022),
- numpy version 1.26.4,
- PyAutoGUI version 0.9.54,
- mediapipe version 0.10.11 (MediaPipe, n.d.),
- pyinstaller version 6.6.0 (Cortesi, n.d.),
- and the win32gui module (Wuxc, n.d.).

## Flowchart Design

A **flowchart** is a visual representation of a process or algorithm, illustrating the steps required to complete a specific task. It serves as a universal tool that can be easily understood by various stakeholders, including programmers, managers, and end-users. The flowchart of the Smart Presentation Feature program is shown in **Figure 1**.

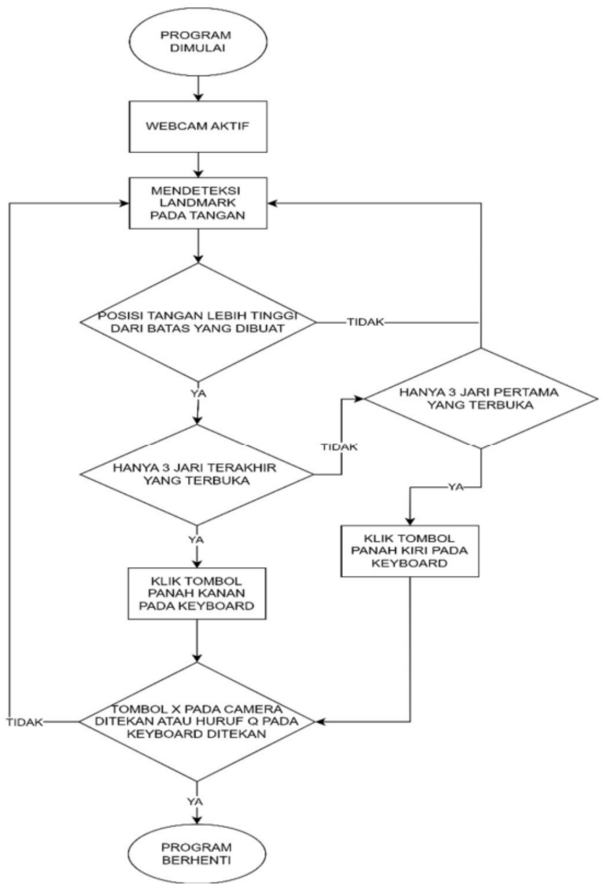

PROGRAM DIMULAI
WEBCAM AKTIF
MENDETEKSI LANDMARK PADA TANGAN
POSISI TANGAN LEBIH TINGGI DARI BATAS YANG DIBUAT
TIDAK
YA
HANYA 3 JARI PERTAMA YANG TERBUKA
HANYA 3 JARI TERAKHIR YANG TERBUKA
TIDAK
YA
YA
KLIK TOMBOL PANAH KIRI PADA KEYBOARD
KLIK TOMBOL PANAH KANAN PADA KEYBOARD
TOMBOL X PADA CAMERA DITEKAN ATAU HURUF Q PADA KEYBOARD DITEKAN
TIDAK
YA
PROGRAM BERHENTI

**Figure 1.** *Program Flowchart*

## RESULT AND DISCUSSION

The result of the Smart Presentation Feature program is an executable file that can be directly run on Windows 10 and 11 operating systems without the need to open the source

code or run it through a terminal. The executable file for the Smart Presentation Feature is available for download via the following link:
https://github.com/hng011/smart-presentation/releases/download/v1.0.1/smart_presentation_feature.zip

## Open Finger Identification Logic

The hand gesture recognition process conducted by the program consists of several stages, beginning with the detection of 21 hand landmarks, as illustrated in the figure 2.

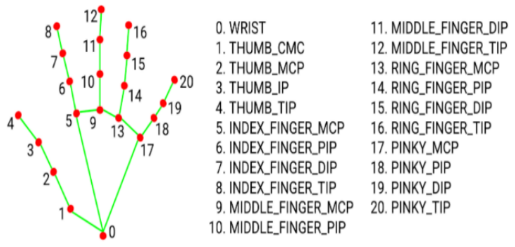


**Figure 2.** Hand Landmarks

These 21 landmarks are obtained using MediaPipe Hands, a solution provided by MediaPipe. MediaPipe Hands consists of several models working in tandem, beginning with a Single-Shot Detector (SSD) that detects the location of the hand. Once the hand is detected, the Hand Landmark Model identifies the 21 specific hand landmarks using a regression technique.

After detecting the hand landmarks, the next step is to determine which fingers are open. Figure 3 shows a pseudocode used to identify the open fingers. An array [4, 8, 12, 16, 20], corresponding to the fingertips (as shown in Figure 2), is used, along with an empty array to store the status of each finger (open or closed). The identification is performed by comparing the fingertip coordinates with their corresponding lower joint coordinates [1, 6, 10, 14, 18]. If the fingertip position is higher than its reference joint, a value of 1 is stored in the array, indicating the finger is open; otherwise, a value of 0 is stored, indicating it is closed.

```
opened_finger = []
finger_points = [4, 8, 12, 16, 20]
hand = [0..20]
for point in finger_points:
        if hand[point][2] < hand[point-2][2]:
                opened_finger.append(1)
        else:
                opened_finger.append(0)
```

Figure 3. Pseudocode for Open Finger Identification

The value 2 in [...,][2] refers to the horizontal coordinate, while using 1 would refer to the vertical position. After processing, the opened_finger array will contain five values representing the thumb to the little finger, respectively. This array only contains binary values: 1 for open and 0 for closed.

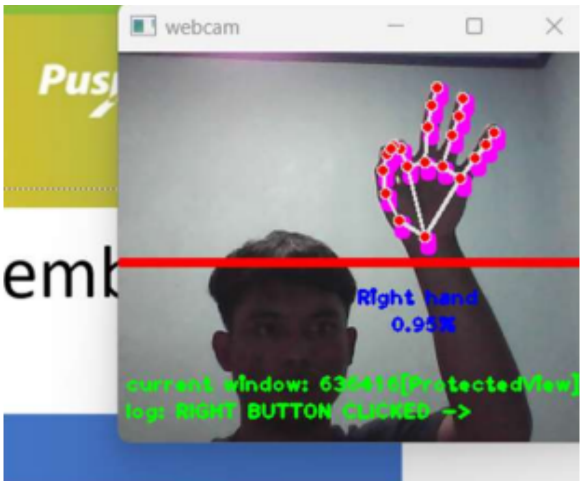
## Presentation Slide Navigation Logic

The logic for navigating presentation slides—such as in Microsoft PowerPoint, Google Slides, Canva, and similar platforms—relies first on identifying which fingers are open, as previously explained. Once a final array containing five binary values (1s and 0s) is produced, two additional arrays are used to define gesture-based slide navigation commands.

- The first array [0, 0, 1, 1, 1] represents the gesture to move forward to the next slide.
- The second array [1, 1, 1, 0, 0] represents the gesture to move backward to the previous slide.

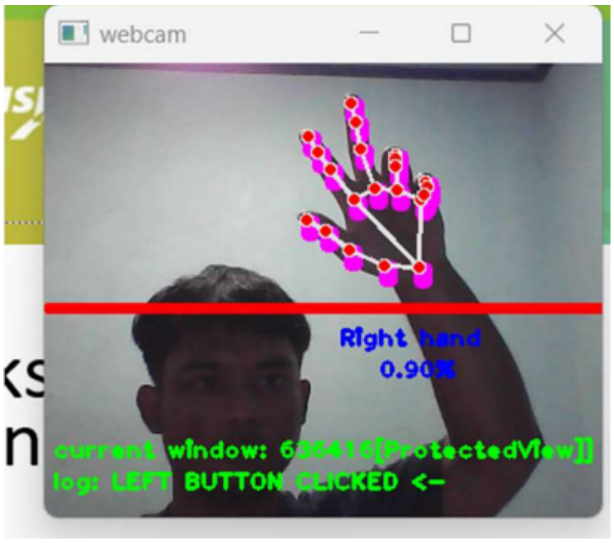
These gestures are illustrated in Figure 4 and Figure 5, respectively. When the detected finger pattern matches the first array, the program sends a command to move to the next slide. When it matches the second array, it sends a command to move to the previous slide. This functionality is made possible by the win32gui module.

The win32gui module provides an interface to the native Win32 GUI API [14]. It offers a wide range of functions that allow users to interact directly with the Windows GUI system through Python code. For example, it can simulate pressing the right arrow key when the first array is detected or the left arrow key when the second array is identified.



```
current window: 63[illegible][ProtectedView]
log: RIGHT BUTTON CLICKED ->
```

**Figure 4.** Hand Gesture for the Array Value [0, 0, 1, 1, 1]



```
current window: 636416[ProtectedView]]
log: LEFT BUTTON CLICKED <-
```

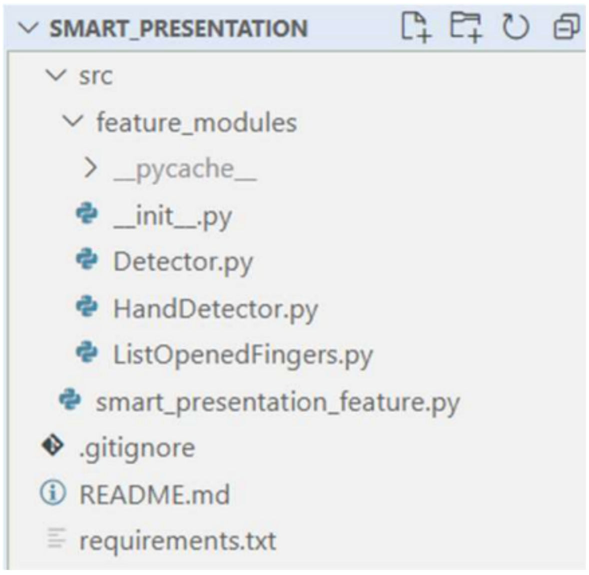
**Figure 5**. Hand Gesture for the Array Value [1, 1, 1, 0, 0]

### Executable File Creation

The program was converted into an executable file using **PyInstaller**, a tool that bundles Python scripts and all required dependencies into a single standalone application (Cortesi, n.d.). This enables end-users to run the application without the need to install Python or any external libraries separately.

The process of creating the executable file for the Smart Presentation Feature involved the following steps:

1. Ensure that all Python code files are located in the same directory, for example, placed inside a folder named src, as illustrated in figure 6.



```
SMART_PRESENTATION
  src
    feature_modules
      __pycache__
      __init__.py
      Detector.py
      HandDetector.py
      ListOpenedFingers.py
    smart_presentation_feature.py
  .gitignore
  README.md
  requirements.txt
```

**Figure 6.** Directory Structure

2. Then, enter the following command in the terminal:
3. After pressing **Enter**, and once the command has been executed, two new folders named **"build"** and **"dist"** will appear in the main directory, along with a new file in **.spec** format.
    - The **build** folder contains temporary files generated during the conversion process.
    - The **dist** folder contains the final result of the conversion, including the executable file and several other supporting files or folders.
    - The **.spec** file stores the configuration used by the **PyInstaller** library during the packaging process.
4. The **.exe file** located in the **dist** folder is the file that can be executed directly by the user.

## Distance-Based Testing

In this study, distance-based testing was conducted to evaluate how far the feature can detect hand gestures for navigating presentation slides. Table 1 presents the results of testing the feature at four different distance intervals:

**Table 1.**System Responsiveness Based on Distance

| **No** | **Distance** | **Result** |
|---|---|---|
| 1 | 0–1 Meter | Responsive |
| 2 | 1–2 Meters | Responsive |
| 3 | 2–3 Meters | Less Responsive |
| 4 | > 3 Meters | Not Responsive |

As shown in Table 1, the system was fully responsive within a range of 0 to 2 meters. However, responsiveness dropped at distances greater than 2 meters, and no response was observed beyond 3 meters.

## Testing with Both Hands Visible

Another test was conducted to assess the system’s performance when both hands are visible to the camera. It was observed that the system sometimes failed to accurately identify the active hand performing the gesture. Figure 7 illustrates a scenario where this issue occurred.

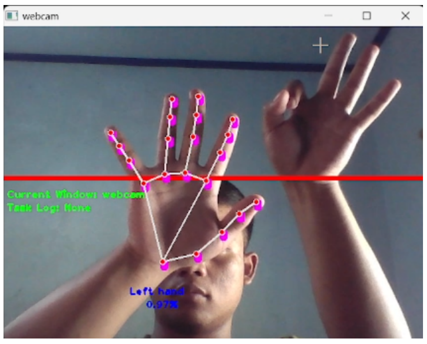


**Figure 7.** *Example of Incorrect Hand Detection During Gesture Recognition*

In the instance shown in Figure 7, the right hand was intended to perform the gesture to trigger a slide change. However, the system mistakenly detected the left hand, which was not performing any gesture. Consequently, the system did not execute any command.

To minimize detection errors, users are advised to ensure that only one hand remains visible to the camera during presentation control.

## CONCLUSION

The hand gesture feature was developed to enable slide transitions in presentation applications such as Microsoft PowerPoint, Google Slides, Canva, and other platforms that support navigation using the right and left arrow keys on a keyboard. The feature was successfully implemented using the Python programming language and several supporting libraries and modules, including MediaPipe, OpenCV, PyAutoGUI, win32gui, PyInstaller, and a virtual environment via Miniconda. This feature is deployable through an executable file, compatible with Windows 10 and 11 operating systems.

Based on the experimental results, several conclusions can be drawn. First, the feature responds effectively at a distance of 0 to 2 meters. Between 2 and 3 meters, system responsiveness is reduced. Beyond 3 meters, the feature no longer responds. Furthermore, a user satisfaction questionnaire utilizing a Likert scale yielded a score of 89.26%, which falls under the “Strongly Agree” category. This indicates a high level of user satisfaction with the Smart Presentation Feature.

The feature functions optimally when the distance between the user's hand and the webcam is no greater than 2 meters. It is also recommended that only one hand be visible to the webcam to avoid detection errors caused by multiple hands being interpreted simultaneously.

Future enhancements to the Smart Presentation Feature could include:

- Displaying an icon indicating the currently active application,
- Offering alternative methods for slide navigation besides hand gestures to improve accessibility, and
- Expanding compatibility to multiple platforms to reach a broader user base.

## REFERENCES

Afni, S. V., Silmina, E. P., & Pangestu, I. B. (2021). *Computer vision digunakan untuk memantau pemuda di masa pandemi Covid-19*. Procedia of Engineering and Life Science, 1(2).

Budiman, S. N., Lestanti, S., Evvandri, S. M., & Putri, R. K. (2022). Pengenalan gestur gerakan jari untuk mengontrol volume di komputer menggunakan library OpenCV dan MediaPipe. *Jurnal Ilmiah Teknik Informatika*, 16(2), 223–232.

Cortesi, D. (n.d.). *PyInstaller manual*. PyInstaller. https://pyinstaller.org/en/stable/ (Accessed: June 16, 2024)

Donaldson, T. (n.d.). *Python as a first programming language for everyone*. https://www.cs.ubc.ca/wccce/Program03/papers/Toby.html (Accessed: April 24, 2024)

Humaira, D. Z. (2023). *Aplikasi Kost Mamih berbasis Android* [Unpublished undergraduate thesis].

International Business Machine (IBM). (n.d.). *What's computer vision?* https://www.ibm.com/topics/computer-vision (Accessed: April 25, 2024)

Khan, H. (2021). *Types of AI: Different types of artificial intelligence systems*. FossGuru. https://fossguru.com/types-of-ai-different-types-of-artificial-intelligence-systems (Accessed: April 9, 2024)

Mahesh, M., Reddy, V., Reddy, A., & Reddy, C. (2022). Object detection and dimensioning using OpenCV. *International Journal of Creative Research Thoughts*, 10(6), 60–68.

MediaPipe. (n.d.). *MediaPipe hands*. https://github.com/google-ai-edge/mediapipe/blob/master/docs/solutions/hands.md (Accessed: April 25, 2024)

Liljenberg, P. (n.d.). *The Python X Library*. https://github.com/python-xlib/python-xlib (Accessed: April 26, 2024)

Saini, N. (2023). Research paper on artificial intelligence & its applications. *International Journal for Research Trends and Innovation*, 4(8), 356–360.

Sruthi, S., & Swetha, S. (2023). Hand gesture controlled presentation using OpenCV and MediaPipe. *International Journal of Engineering Technology and Management Sciences*, 4(7), 338–342.

Wuxc. (n.d.). *win32gui*. https://github.com/wuxc/pywin32doc/blob/master/md/win32gui.md (Accessed: April 26, 2024)

Zahid, P., & Fatima, N. (2016). Performance comparison of most common high level programming languages. *International Journal of Computing Academic Research*, 5(5), 246–258.