Babylonian Journal of Machine Learning
Vol.2024, **pp**. 56–62
DOI: https://doi.org/10.58496/BJML/2024/005; ISSN: 3006–5429
https://mesopotamian.press/journals/index.php/BJML

Research Article

A Proposed Method of Gesture-controlled presentation software design

Nadia Mahmood Hussien [1,*], Yasmin Makki Mohialden [1], Wurood A. Jbara [1]

[1] *Computer Science Department, Collage of Science, Mustansiriyah University, Baghdad-Iraq*

ARTICLE INFO

Article History
Received 11 Jan 2024
Revised 21 Feb 2024
Accepted 08 Mar 2024
Published 01 Apr 2024

Keywords
Software engineering
Gesture control
Presentation software
Machine learning
Interactive user interface
Computer vision

ABSTRACT

This paper introduces an innovative method for developing a presentation application that empowers users to seamlessly control slide transitions and other essential actions through intuitive hand gestures. The approach integrates sophisticated computer vision algorithms capable of real-time gesture detection and interpretation from a standard webcam feed. Furthermore, machine learning techniques personalize the system to individual users' unique gestures, enhancing usability and accuracy. The proposed method is a groundbreaking innovation that seamlessly integrates with existing presentation tools. Furthermore, the research delves into cross-device synchronization, enabling a cohesive presentation experience. To ensure optimal usability and performance, we follow established software engineering principles, resulting in a user-friendly interface and an efficiently structured codebase. This paper comprehensively guides the design, implementation, and potential of this gesture-controlled presentation software.

1. INTRODUCTION

In today's dynamic world of technological advancements, our interaction with digital systems continues to evolve. More intuitive and immersive interfaces are complementing and, in some cases, replacing traditional input methods like keyboards and mice. Gesture control technology, allowing users to interact with software and devices through natural body movements, has garnered increasing attention, remarkably bridging the gap between humans and machines [1, 2].

Presentations, fundamental in various domains like business, education, and entertainment, heavily rely on the presenter's ability to engage and interact with the audience. However, traditional presentation software often limits presenters' mobility and engagement potential. Gesture recognition technology lets presenters manipulate slides with hand motions without gadgets or direct touch, improving presenting experiences [3].

This study presents a simpler gesture-controlled presentation software framework that covers its fundamental structure. While not addressing computer vision and machine learning in detail, this example offers a starting point [4].

A software engineering technique is used to construct, test, and deploy computer systems to solve real-world problems while adhering to technical standards and quality [5] and [6].

A formal design blueprint for gesture-controlled presentation software is the main contribution. Existing solutions sometimes entail complex implementations, but this example simplifies the basics. This contribution might inspire interactive presentation innovation and inquiry.

Our main objectives are to :

1. Introduce the fundamental design and essential components of gesture-controlled presentation software.
2. User Interaction: Demonstrating how user motions may influence slide transitions for intuitive and engaging presentations.
3. Code Framework: Providing a basic code framework that shows how software components may be structured and interact for gesture-based control.

*Corresponding author. Email: nadia.cs89@uomustansiriyah.edu.iq

4. Extendibility: Inviting developers to use powerful computer vision and machine learning methods for robust gesture recognition and interpretation.
5. Innovation: inspiring researchers and developers to use computer vision and machine learning to produce increasingly advanced gesture-controlled presentation software.

The structure of the paper is as follows: Section 2 discusses related work, Section 3 outlines the proposed method, and Section 4 offers conclusions and suggestions for future work.

2. RELATED WORK

Gesture recognition technology has emerged as a promising approach for enhancing human-computer interaction, especially in the domain of presentation software. Prior research has explored various techniques and systems to achieve gesture-based control of slides.

Microsoft Kinect has been utilized in several studies to enable hand-gesture navigation for PowerPoint presentations, with a focus on slide transitions and animation triggers [4]. Machine learning techniques have further advanced this field by enabling customized gestures tailored to user preferences, thereby improving system accuracy and usability [7].

Recent implementations leverage OpenCV and Python for practical, real-time gesture control. For instance, a study demonstrated how simple hand motions could effectively navigate slides, emphasizing the ease of use and integration with standard hardware [8]. Moreover, systems incorporating machine learning have shown potential in converting gestures into precise commands for seamless presentation control [9].

While these approaches demonstrate significant progress, challenges such as real-time performance, adaptability, and integration with existing tools remain. This research builds on these insights, proposing a simplified and robust framework that integrates computer vision and machine learning for intuitive and efficient gesture-controlled presentation software.

3. THE PROPOSED METHOD

Figure 1 shows the suggested technique block diagram components:

1. Video Capture: Capturing webcam frames.
2. Hand Gesture Detection: OpenCV is used to process frames and detect hand gestures.
3. Gestural Interpretation: Converts sensed motions into slide transitions.
4. Slide Simulation: Used a keyboard simulation library to replicate slide changer keyboard inputs.
5. Display and Control: Displays video and responds to gestures.

The suggested gesture-controlled presentation software's architecture and capabilities are shown in this block diagram.

Fig .1. The proposed design block diagram

3.1 Requirements Specification

1. Functional Requirements:

- The software should be set up and use the computer's camera to record video.
- The app should analyze video frames and recognize hand motions using OpenCV.
- Hand motions should be identified as left, right, up, and down swipes.
- Detect and transform motions into actions, such as slide transitions or instructions should be identified.
- The software should use a keyboard simulation library (e.g., keyboard) to simulate keyboard inputs for slide changes based on gestures.
- The software should show live camera footage, enabling users to view their motions and presentation.

- The user should be able to perform gestures to control the presentation
- The software must allow the exit of the display.

2. Non -Functional Requirements:

- Performance:
  - Real-time and seamless gesture detection and reaction.
  - Ensure smooth transitions by synchronizing webcam frame rate and presentation speed.
- Reliability:
  - The software must eliminate slide control mistakes by effectively detecting gestures.
  - It must also manage unforeseen events, such as camera disconnects or gesture misinterpretations.
- User Interface:
  - The interface should be straightforward, with visual clues suggesting gestures. • The software should be simple to use.
- Compatibility:
  - The software must be compatible with typical cameras and presenting tools like Microsoft PowerPoint.
- Documentation:
  - Provide clear and comprehensive user documentation, including installation and troubleshooting instructions.

a. Technical Requirements:

- Programming Languages and Libraries
- The software should be developed using a programming language like Python.
- It should utilize libraries such as OpenCV for computer vision [10] and keyboard [11]for simulation.
- **Hardware Requirements:**
- The software should be capable of running on computers with standard webcams and suitable hardware specifications.

3.2 Use Cases Scenario

A use-case diagram is a behavioral UML diagram commonly employed to analyze different systems. It facilitates the examination of various roles in a plan and illustrates how these roles communicate with the system [11], [12], and [13]. This section provides detailed scenarios depicting how users will interact with the software, controlling presentations using gestures. In this use-case scenario, the user commands a display using hand motions identified by the gesture-controlled presentation software's computer vision capabilities. Figure 2 illustrates the general use case of the system. The actors involved are three: the user, the PPT slides, and the system.

A scenario is a concise user story that describes who utilizes the system and what they are attempting to achieve. A scenario comprises a series of simple, discrete stages that are either done by the "system" or the "user.[14][15]. Below are the steps of a general use case diagram.

3.3 Actor: User

Preconditions: Computer users install and configure gesture-controlled presentation software. The webcam and PC are compatible.

- Basic Flow:

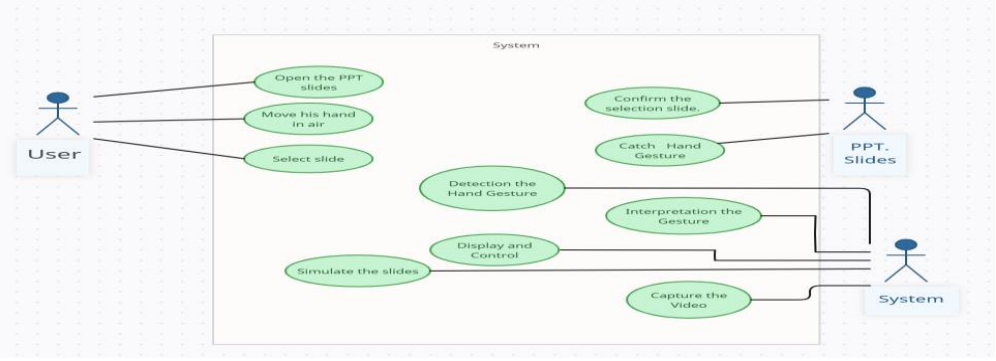

Fig .2. Use case of the proposed method

- The user starts gesture-controlled presentation software.
- The software activates the computer's webcam to record video.
- The OpenCV software continually analyzes video frames to recognize hand motions.
- The software determines a hand gesture's action (e.g., swipe left or right).
- The software simulates a key press using a keyboard simulation library if a slide transition is recognized (e.g., swipe right).
- PowerPoint moves to the next slide after the simulated keystroke.
- The software monitors the camera stream to show the user's motions and presentation.
- Swipe left to switch to the previous slide to manage the presentation.
- Press 'q' or use another technique to terminate the presentation.
- The software releases the camera, quits the presenting tool, and closes windows after the presentation.

- Alternative Flow:
  - If no hand motions are detected or recognized, the software displays the current slide without taking any action.
  - Users may close the software to free up the camera and resources if issues or interruptions occur.
- Postconditions:
  - The gesture-controlled presentation software may be closed after the presentation. Release the webcam and resources, and close PowerPoint.
- Constraints
  - Highlight any limits, such as lighting issues, that may affect gesture recognition.

3.4 The Class Diagram

In the Unified Modeling Language (UML), a class diagram is a form of static structural diagram that depicts the structure of a system by displaying the system's classes, their attributes, actions (or methods), and the relationships between objects. [15-19]. Figure 3. Provide a simplified version of the class diagram for your gesture-controlled presentation software:

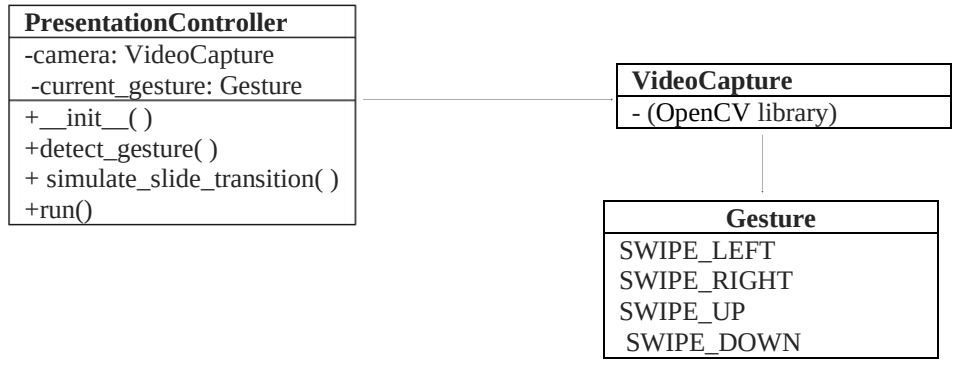

Fig .3. The proposed design class diagram

In this class diagram, The PresentationController class has camera and gesture characteristics. Initialize, detect motions, simulate slide changes, and then execute the presentation loop with it.
The OpenCV library's VideoCapture class handles webcam capturing. It depends on the presentation controller.
The enumeration of gestures includes SWIPE_LEFT, SWIPE_RIGHT, SWIPE_UP, and SWIPE_DOWN.

3.5 Pseudo-Code Algorithm

Gesture-Controlled Presentation Software stages are summarized in this pseudo-code. Figure 4 Initializes characteristics,

detects motions, mimics slide transitions, and controls the presentation loop. For resource cleaning and management, the code execution is in a try-except-finally block.

```
InitializeGestureEnum():
    Define Gesture enumeration values:
    - SWIPE_LEFT
    - SWIPE_RIGHT
    - SWIPE_UP
    - SWIPE_DOWN
Class PresentationController:
    Initialize():
        Initialize class attributes:
        - camera
        - current_gesture
    DetectGesture(frame):
        # TODO: Implement gesture detection logic
        Return detected gesture or None
    SimulateSlideTransition(gesture):
        If gesture is SWIPE_LEFT:
            Simulate pressing "left" arrow key
        Else If gesture is SWIPE_RIGHT:
            Simulate pressing "right" arrow key
    Run():
        Loop while True:
            ret, frame = Read a frame from the camera
            If ret is False:
                Break the loop
            gesture = DetectGesture(frame)  # Detect gesture
            If gesture is not None and gesture is not equal to current_gesture:
                current_gesture = gesture
                SimulateSlideTransition(gesture)  # Simulate slide transition

            Display the video feed
            If key "q" is pressed:
                Break the loop

        Release the camera
        Close all windows
If __name__ == "__main__":
    Try:
        Create an instance of PresentationController
        Call Run() method on the instance
    Except KeyboardInterrupt:
        Pass  # Handle keyboard interrupt
    Finally:
        Close all windows
        Release camera resource
```

Fig .4. Pseudo-code algorithm

4. CONCLUSION

A gesture-controlled presentation software highlights the potential of computer vision and user interactions to facilitate seamless slide transitions through hand gestures. However, to achieve a comprehensive implementation, several key areas require further exploration and enhancement.

The primary focus should be on realizing the gesture detection component, ensuring accurate identification and interpretation of gestures using a robust computer vision library like OpenCV, and implementing advanced algorithms for seamless user interaction.

Integration with PowerPoint is crucial. To create a consistent and intuitive user experience, grasp the tool's APIs and programmatic slide manipulation methods.

More gesture recognition beyond basic instructions might boost the software's adaptability. Learning and adapting to users' movements with machine learning models improves accuracy and expands action range.

Users may easily understand motions and actions with better user interface signals and feedback.

Further development may include cross-device synchronization for multi-device presentations, accessibility needs, speed optimization for real-time interactions, rigorous testing to confirm functionality, and thorough user documentation.

Although the existing program shows a basic notion, gesture-controlled presentation software requires careful study, sophisticated development, and a commitment to improving user experiences with cutting-edge technology.

Conflict Of Interest

None.

Funding

None.

Acknowledgment

The Authors would like to thank Mustansiriyah University (https://uomustansiriyah.edu.iq/) in Baghdad, Iraq, for their support in the present work.

References

[1] D. Jessintha, P. P. Kumar, S. Jaisiva, T. A. Kumar, and C. Ananth, "Social Service Robot using Gesture recognition technique," Journal of Physics: Conference Series, vol. 2466, no. 1, p. 012020, 2023. [Online]. Available: https://doi.org/10.1088/1742-6596/2466/1/012020

[2] M. Liu, J. Wei, Y. Liu, and J. Davis, "Do humans and machines have the same eyes? Human-machine perceptual differences on image classification," arXiv preprint, arXiv:2304.08733, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2304.08733

[3] B. Liao, P. E. van den Berg, P. J. van Wesemael, and T. A. Arentze, "Individuals' perception of walkability: Results of a conjoint experiment using videos of virtual environments," Cities, vol. 125, p. 103650, 2022.

[4] J. Lee, R. Rajapakse, and K. Miyata, "An Empirical Study on Intuitive Gesture Manipulation in Virtual Reality," in 2022 8th International Conference on Virtual Reality (ICVR), pp. 216–224, 2022. [Online]. Available: https://doi.org/10.1109/ICVR55215.2022.9848105

[5] S. M. Chang, "Using gesture recognition to control PowerPoint using the Microsoft Kinect," M.S. thesis, Massachusetts Institute of Technology, 2013. [Online]. Available: https://dspace.mit.edu/handle/1721.1/85410

[6] P. Greenspun, E. Andersson, and A. Grumet, Software engineering for Internet applications, 2021.

[7] S. N. AlSaad and N. M. Hussien, "Landmark-based shortest path detection in alarm system," Al-Mustansiriyah Journal of Science, vol. 29, no. 2, pp. 135–140, 2018.

[8] M. Idrees, A. Ahmad, M. A. Butt, and H. Danish, "I am controlling PowerPoint Using Hand Gestures in Python," Webology, vol. 18, pp. 1372–1388, 2021.

[9] "Hand Gesture Controlled Presentation | OpenCV Python," [Online]. Available: https://www.youtube.com/watch?v=CKmAZss-T5Y. [Accessed: Aug. 24, 2023].

[10] B. Pustode, V. Pawar, V. Pawar, T. Pawar, and S. Pokale, "Smart Presentation System Using Hand Gestures [Preprint]," In Review, 2023. [Online]. Available: https://doi.org/10.21203/rs.3.rs-2549833/v1

[11] "Interactive Presentation Software for Audience Engagement," iSpring Blog, Jun. 9, 2023. [Online]. Available: https://www.ispringsolutions.com/blog/10-interactive-presentation-software-grab-and-hold-an-audiences-attention/

[12] "OpenCV: OpenCV modules," [Online]. Available: https://docs.opencv.org/4.x/. [Accessed: Aug. 25, 2023].

[13] H. Boppre, "Keyboard: Hook and simulate keyboard events on Windows and Linux (0.13.5) [Python; MacOS :: MacOS X, Microsoft :: Windows, Unix]," [Online]. Available: https://github.com/boppreh/keyboard. [Accessed: Aug. 25, 2023].

[14] H. H. Ali, J. R. Naif, and W. R. Humood, "A New Smart Home Intruder Detection System Based on Deep Learning," Al-Mustansiriyah Journal of Science, vol. 34, no. 2, pp. 60–69, 2023.

[15] J. Sun, Q. V. Liao, M. Muller, M. Agarwal, S. Houde, K. Talamadupula, and J. D. Weisz, "Investigating explainability of generative AI for code through scenario-based design," in 27th International Conference on Intelligent User Interfaces, pp. 212–228, Mar. 2022.

[16] W. K. Assunção, S. R. Vergilio, and R. E. Lopez-Herrejon, "Automatic extraction of product line architecture and feature models from UML class diagram variants," Information and Software Technology, vol. 117, p. 106198, 2020.

[17] E. Planas and J. Cabot, "How are UML class diagrams built in practice? A usability study of two UML tools: Magicdraw and Papyrus," Computer Standards & Interfaces, vol. 67, p. 103363, 2020.

[18] M. T. Younis, N. M. Hussien, Y. M. Mohialden, K. Raisian, P. Singh, and K. Joshi, "Enhancement of ChatGPT using API Wrappers Techniques," Al-Mustansiriyah Journal of Science, vol. 34, no. 2, pp. 82–86, 2023.

[19] Y. M. Mohialden, H. A. Abdulbaqi, and N. M. Shati, "Developing collaboration tool for virtual team using UML models," Indonesian Journal of Electrical Engineering and Computer Science, vol. 22, pp. 38–44, 2021.