Gesture based Media Control

Domain: Computer Vision

Presented by Batch-6

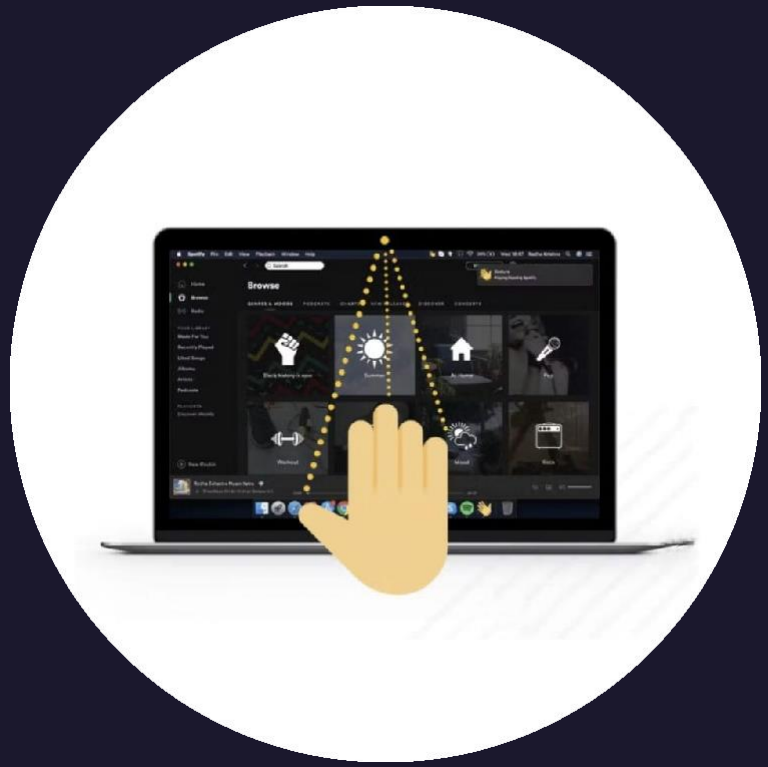
K. Rohit Kumar	2451-20-733-133
K. Santhosh Kumar	2451-20-733-135
J. Jayanth	2451-20-733-137

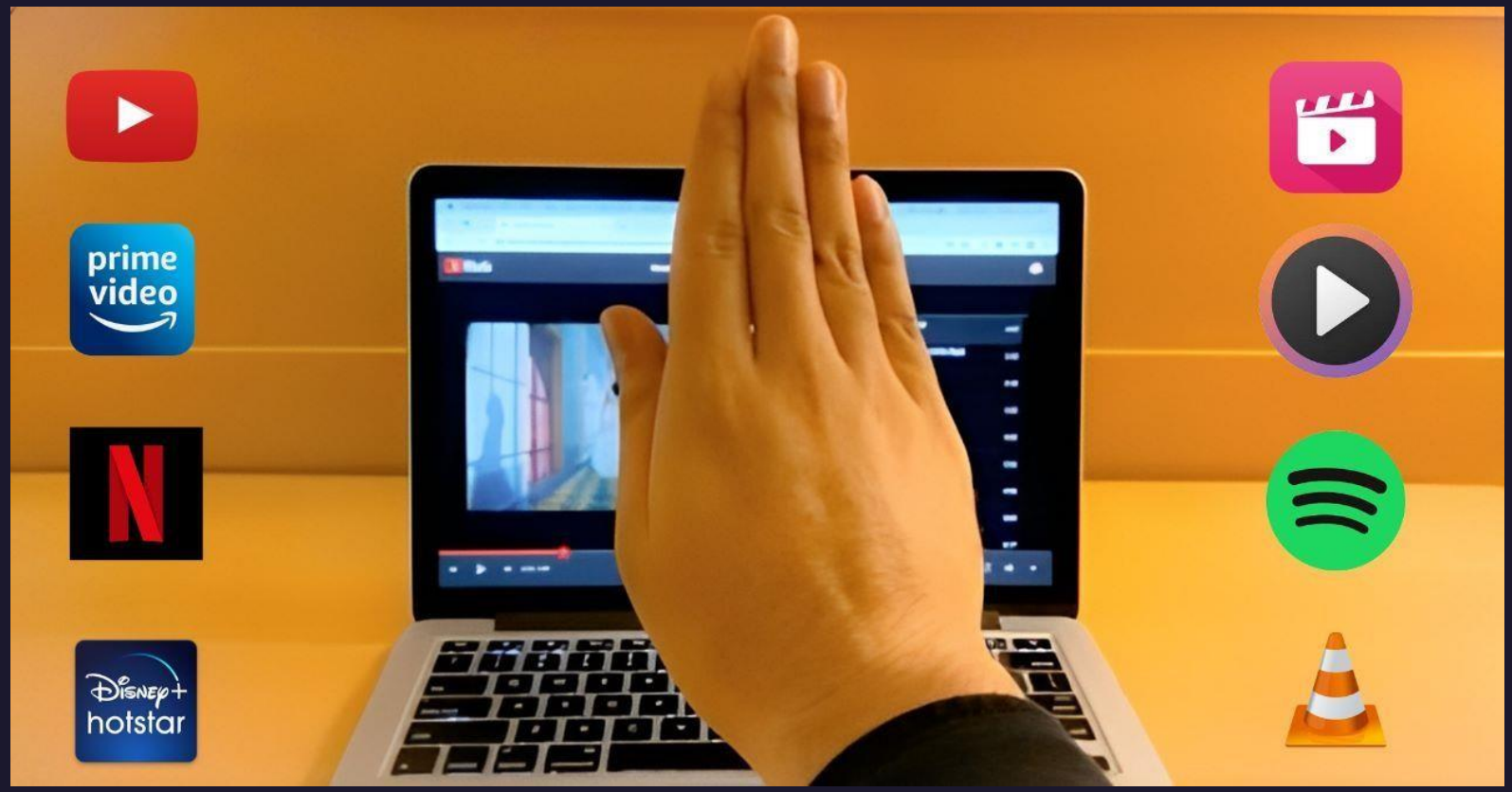
Mentor

V. Sathish
Assistant Professor,
CSE Dept.

Agenda

- Abstract
- Introduction
- Problem description
- System Requirements
- Flow Chart
- Functionalities
- Test Results
- Future scope of work
- Conclusion





Abstract

The Gesture-based media control intuitive way of interacting with multimedia devices, such as TVs, laptops. Utilizes computer vision to enable users to control media playback such as play, pause, volume up, down etc.. through hand gestures. A concise overview of the highlighting the impact of gesture-based control on enhancing user experience in media interaction and accessibility in media consumption.

INTRODUCTION

- In today's world, there is a growing demand for intuitive and hands-free interaction methods. Users crave seamless and natural ways to interact with technology without the constraints of physical input devices

- By leveraging computer vision, this project enhances user experience, offering a seamless and intuitive way of interacting with multimedia content.

Problem Description

The traditional mouse and keyboard interfaces for media control lack intuitiveness. This project addresses the challenge by utilizing computer vision to enable users to control media playback through hand gestures.

System Requirements

Hardware Requirements

- Minimum of 8GB RAM
- i5/ryzen5 or higher CPU
- 2GB dedicated GPU or higher

Software Requirements

- PC running windows 7 or higher OS
- Python(version 3.5 or above)
- IDE (VS Code,Google colab, jupyter.)

Packages

- OpenCV
- MediaPipe
- PyautoGUI

Languages

- Python

System Requirements

Hardware Requirements

- Minimum of 8GB RAM
- i5/ryzen5 or higher CPU
- 2GB dedicated GPU or higher

Software Requirements

- PC running windows 7 or higher OS
- Python(version 3.5 or above)
- IDE (VS Code,Google colab, jupyter.)

Packages

- OpenCV
- MediaPipe
- PyautoGUI

Languages

- Python

System Requirements

Hardware Requirements

- Minimum of 8GB RAM
- i5/ryzen5 or higher CPU
- 2GB dedicated GPU or higher

Software Requirements

- PC running windows 7 or higher OS
- Python(version 3.5 or above)
- IDE (VS Code,Google colab, jupyter.)

Packages

- OpenCV
- MediaPipe
- PyautoGUI

Languages

- Python

System Requirements

Hardware Requirements

- Minimum of 8GB RAM
- i5/ryzen5 or higher CPU
- 2GB dedicated GPU or higher

Software Requirements

- PC running windows 7 or higher OS
- Python(version 3.5 or above)
- IDE (VS Code,Google colab, jupyter.)

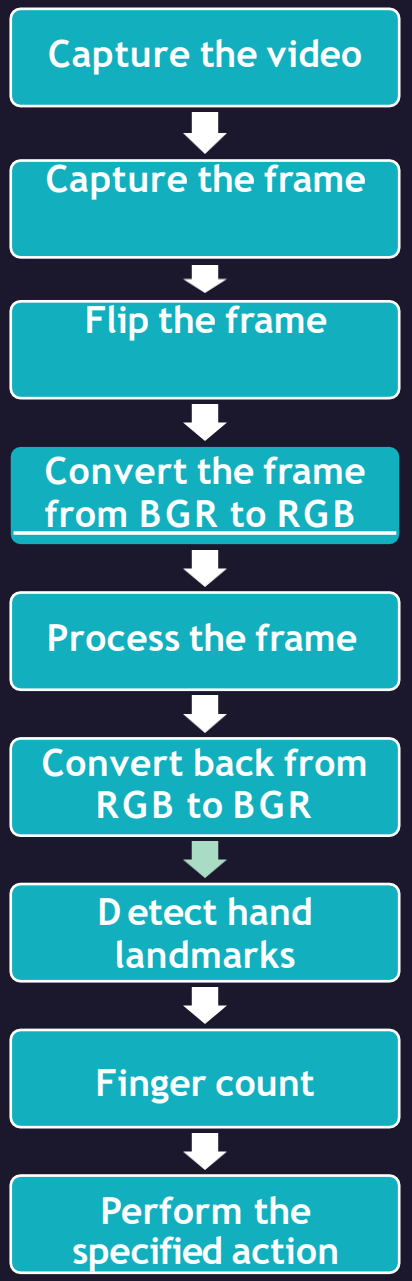
Packages

- OpenCV
- MediaPipe
- PyautoGUI

Languages

- Python

FLOW CHART


Capture the video
Capture the frame
Flip the frame
Convert the frame from BGR to RGB
Process the frame
Convert back from RGB to BGR
D etect hand landmarks
Finger count
Perform the specified action

FUNCTIONALITIES

VIDEO CAPTURE AND PREPROCESSING

- cv2.Videocapture(0) – intializes the webcam with index 0
- cap.read() – retrieve the current frame
- cv2.flip() – ensures that the video feed appears as the mirror image
- cv2.cvtColor() – converts the color space of an image for mediapipe

FUNCTIONALITIES

HAND DETECTION AND TRACKING

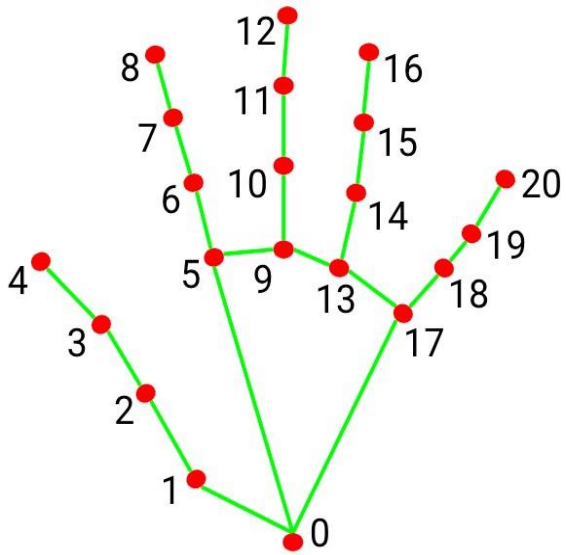
- The **Mediapipe** library is used for hand detection and tracking. Computer vision algorithms are to identify and track the user's hand in real-time
- **Hands** – used to hand detection and tracking

  hands.process() – processes the image and detects hand landmarks
- **mp.solutions.drawing_utils:**

  draw_landmarks() – used to draw hand landmarks and connections

FUNCTIONALITIES

HAND LANDMARK EXTRACTION



0. WRIST
1. THUMB_CMC
2. THUMB_MCP
3. THUMB_IP
4. THUMB_TIP
5. INDEX_FINGER_MCP
6. INDEX_FINGER_PIP
7. INDEX_FINGER_DIP
8. INDEX_FINGER_TIP
9. MIDDLE_FINGER_MCP
10. MIDDLE_FINGER_PIP
11. MIDDLE_FINGER_DIP
12. MIDDLE_FINGER_TIP
13. RING_FINGER_MCP
14. RING_FINGER_PIP
15. RING_FINGER_DIP
16. RING_FINGER_TIP
17. PINKY_MCP
18. PINKY_PIP
19. PINKY_DIP
20. PINKY_TIP

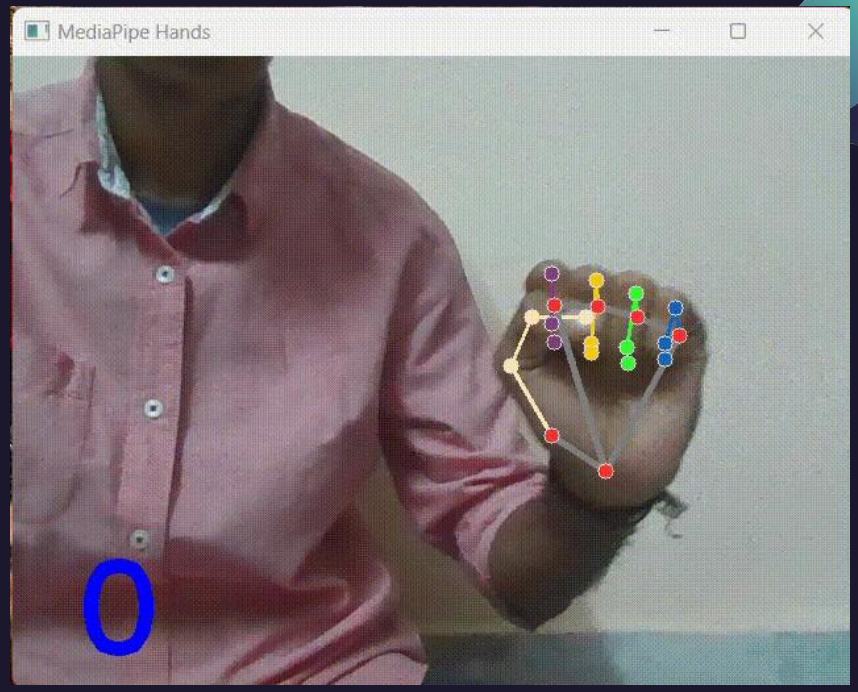
These landmarks represent key points on the hand that aid in recognizing different hand gestures. By analyzing the positions and movements of these landmarks, the system can accurately determine the finger count.

The landmark attribute of the hand_landmarks object contains the detected hand landmarks info.

FUNCTIONALITIES

FINGER COUNT

- By comparing the relative positions of these landmarks, the system can accurately identify the number of fingers raised by the user.
- By utilizing the positions of the hand landmarks to perform finger counting based on specific condition
- **hand_landmarks.landmark[index]** – access landmark positions
- **cv2.imshow()** – display image with hand landmarks and finger count



Finger count

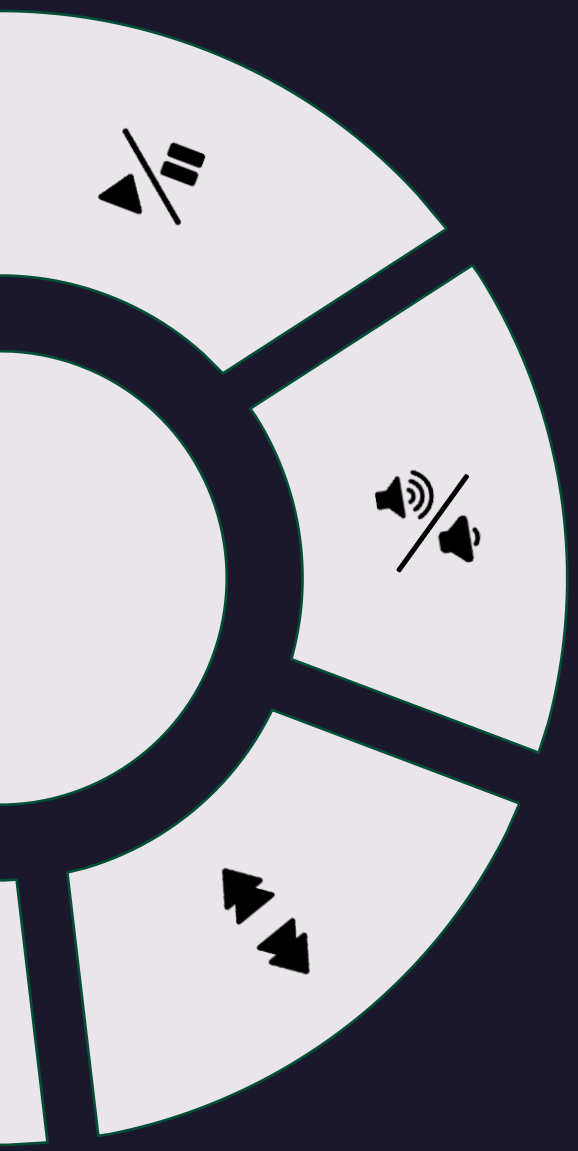
FUNCTIONALITIES

PERFORMING THE SPECIFIED ACTIONS

- pyautoGUI – simulating the keyboard keys
- pyautogui.press(key) – simulates pressing a key
- pyautogui.hotkey(key1,key2,….) – simulates multiple keys

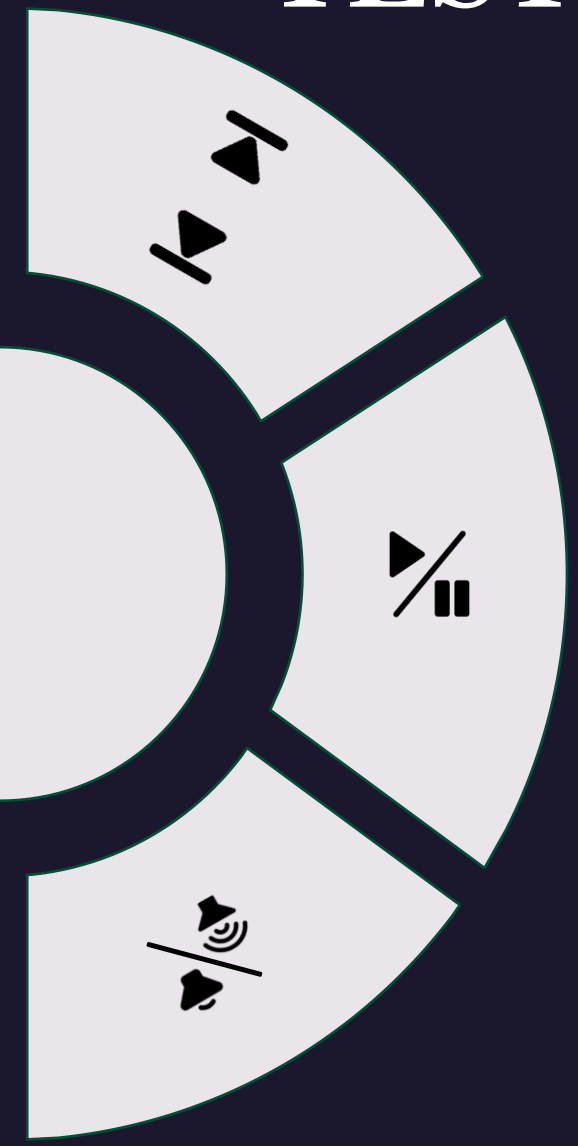
RESOURCE RELEASE

- cap.release() – release the webcam
- cv2.destroyAllWindows() – closes opencv windows



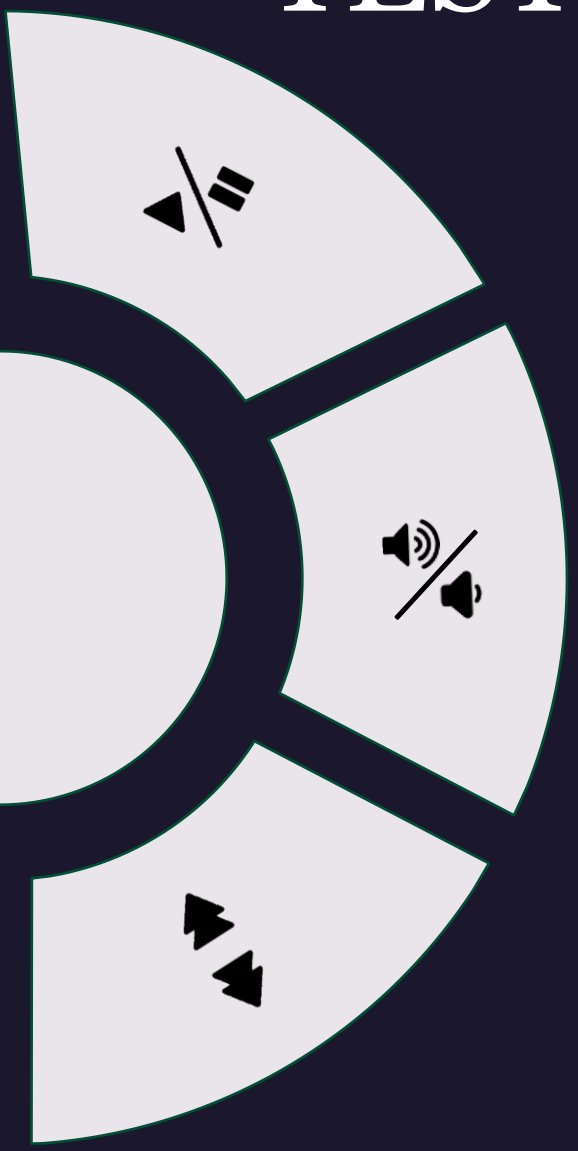
TEST RESULTS

TEST RESULTS



Play/pause

TEST RESULTS



Volume up/down

TEST RESULTS

Forward/Backward

TEST RESULTS

Next/Previous track

Simulating media control actions

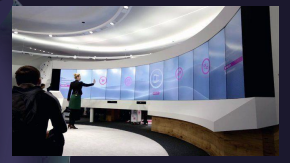
Apps	Play/Pause	Volume Up	Volume Down	Forward	Backward	Next track	Previous track
Youtube	5	3	4	1	2	9	7
Netflix	5	3	4	1	2	-	-
Prime	5	3	4	1	2	-	-
Disney+	5	3	4	1	2	-	-
Jio Cinema	5	3	4	1	2	-	-
VLC	5	3	4	1	2	-	-
Media player	5	-	-	10	-	8	6

APPLICATIONS

Media player

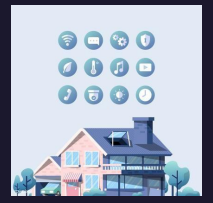
Gaming

Virtual Reality (VR)



Presentations

Accessibility



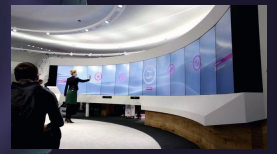
Smart Homes

APPLICATIONS

Media player

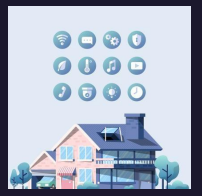
Gaming

Virtual Reality (VR)



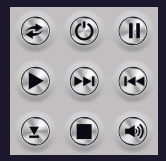
Presentations

Accessibility



Smart Homes

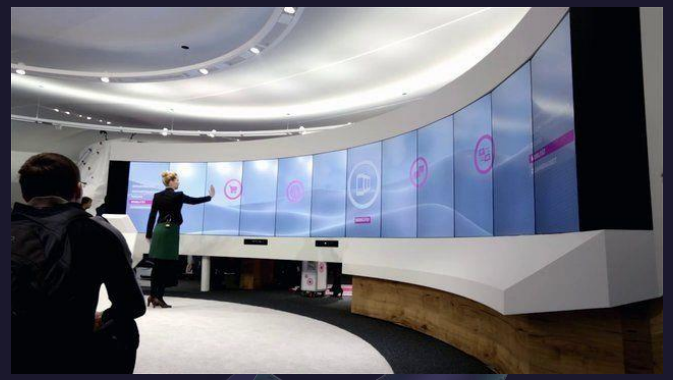
APPLICATIONS



Media player

Gaming

Virtual Reality (VR)



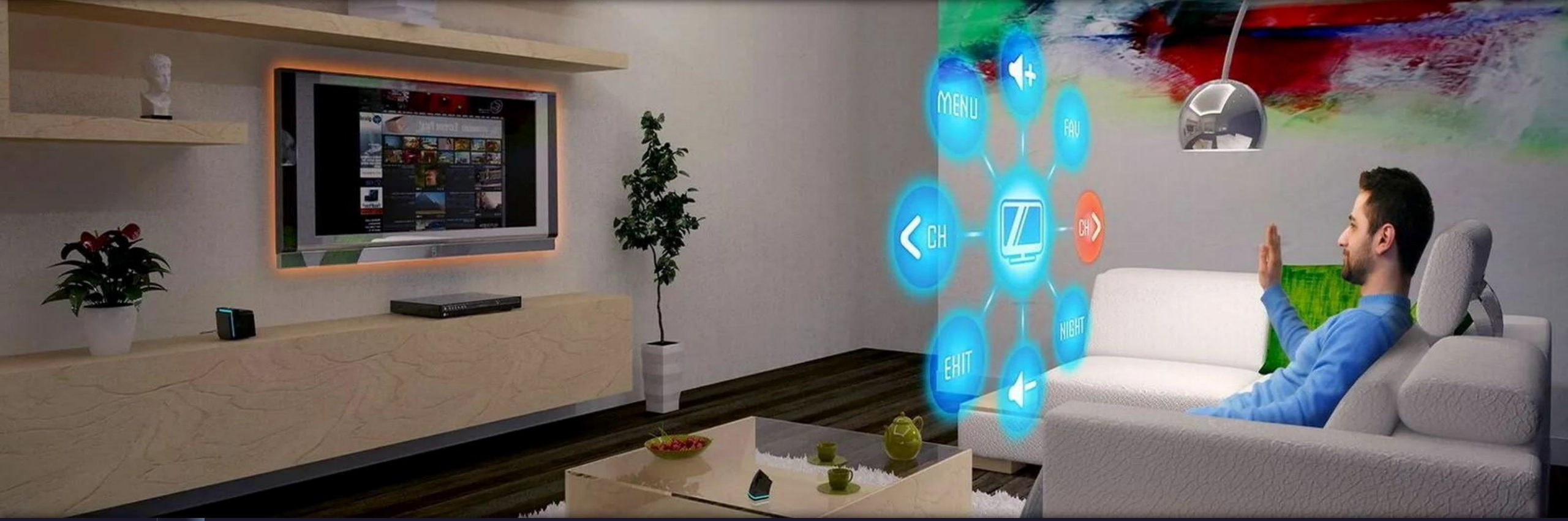
Presentations

Accessibility

Smart Homes

Future scope of the work

The gesture-based media control project has immense potential for expansion and enhancement. Future work could involve integrating the system with virtual reality and augmented reality environments, incorporating more advanced hand gesture recognition algorithms, and expanding its compatibility with a wider range of media platforms and devices, offering users a truly immersive and interactive media control experience.



Conclusion

Gesture-based media control enables intuitive and engaging interaction, enhancing the user experience in media playback. Its successful implementation demonstrates the potential for revolutionizing human-computer interaction and opens avenues for future advancements in immersive media experiences.

Thank You