BHARAT AI-SOC STUDENT CHALLENGE

PROBLEM STATEMENT 2 -TECHNICAL REPORT

TOUCHLESS HCI FOR MEDIA CONTROL USING HAND GESTURES ON NVIDIA JETSON NANO

PLATFORM: NVIDIA JETSON NANO DEVELOPER KIT

SOC: NVIDIA TEGRA X1 SYSTEM-ON-CHIP (QUAD-CORE ARM CORTEX-A57 CPU WITH 128-CORE NVIDIA MAXWELL GPU)

TOOLS: NVIDIA JETPACK SDK | UBUNTU LINUX (ARM64) | PYTHON 3 | OPENCV | MEDIAPIPE | CUDA TOOLKIT | GSTREAMER
DATE: 2025

TEAM DETAILS

RAJAKAVI S
(2ND YEAR | B.E.ELECTRICAL & ELECTRONICS ENGINEERING)

MIDHUNAN D J
(2ND YEAR | B.E.ELECTRICAL & ELECTRONICS ENGINEERING)

SHIVANI B
(3RD YEAR| B.E. ELECTRONICS & COMMUNICATION ENGINEERING)

CHENNAI INSTITUTE OF TECHNOLOGY, CHENNAI

ABSTRACT

The rapid advancement of embedded artificial intelligence and edge computing has created a growing demand for efficient, low-power human–computer interaction systems. Traditional media control mechanisms rely on physical input devices such as keyboards, remotes, or touch interfaces, which limit usability in hands-free environments. This project presents a touchless Human–Computer Interaction (HCI) system that enables real-time media control using hand gesture recognition implemented on an ARM-based embedded platform. The system is developed using the NVIDIA Jetson Nano, powered by an ARM Cortex-A57 processor, demonstrating the capability of ARM architecture in executing AI workloads efficiently at the edge.

The proposed system utilizes a USB camera to capture live video frames, which are processed using OpenCV and MediaPipe frameworks for hand landmark detection and gesture classification. Recognized gestures are mapped to media control commands such as play, pause, volume adjustment, and navigation. The implementation leverages the energy-efficient ARM CPU architecture along with GPU acceleration to achieve real-time performance while maintaining low power consumption. System optimization techniques, including performance mode configuration and efficient video decoding pipelines, were applied to improve frame rate and reduce latency.

Experimental results demonstrate stable gesture recognition with real-time responsiveness suitable for embedded applications. The system highlights the advantages of ARM-based platforms in edge AI deployment, offering a balance between computational performance and power efficiency. This work showcases the feasibility of implementing intelligent touchless interfaces on ARM processors, making it applicable to smart homes, healthcare environments, public kiosks, and accessibility systems. The project emphasizes the role of ARM architecture in enabling scalable, energy-efficient AI solutions for next-generation embedded computing applications.

-TABLE OF CONTENT-

INTRODUCTION

The rapid evolution of embedded computing and artificial intelligence has significantly transformed the way humans interact with digital systems. Traditional human–computer interaction methods rely primarily on physical input devices such as keyboards, mice, remote controllers, and touchscreens. Although effective, these interaction methods require direct physical contact, which limits usability in hands-free environments and may not be suitable for applications requiring hygiene, accessibility, or natural interaction. With the increasing demand for intelligent and contactless interfaces, gesture-based interaction systems have emerged as an important area of research in modern computing.

Recent advancements in edge AI and computer vision have enabled real-time processing of visual data directly on embedded platforms without relying on cloud infrastructure. Edge computing reduces latency, improves privacy, and enhances system responsiveness by performing computation locally. ARM-based processors play a crucial role in enabling such systems due to their high performance-per-watt efficiency, low power consumption, and scalability across embedded and mobile devices. Platforms powered by ARM architecture are widely used in modern smart devices, robotics, and AI-enabled systems because they provide an optimal balance between computational capability and energy efficiency.

This project focuses on the design and implementation of a touchless human–computer interaction system that enables media control using hand gestures. The system is developed using the NVIDIA Jetson Nano Developer Kit, which integrates an ARM Cortex-A57 processor and GPU acceleration for edge AI applications. A USB camera captures real-time video input, and computer vision techniques are applied using OpenCV and MediaPipe frameworks to detect hand landmarks and recognize gestures. The identified gestures are mapped to multimedia control commands such as play, pause, volume adjustment, and navigation, allowing intuitive interaction without physical contact.
The proposed system demonstrates how ARM-based embedded platforms can efficiently execute real-time AI workloads while maintaining low power consumption. By combining computer vision algorithms with embedded edge computing, the project highlights the practical implementation of intelligent touchless interfaces suitable for smart homes, healthcare environments, public kiosks, and assistive technologies. This work emphasizes the growing importance of ARM architecture in enabling scalable, energy-efficient, and real-time human–machine interaction systems.

PROBLEM STATEMENT DESCRIPTION

The need for efficient and intuitive human–computer interaction systems has increased with the advancement of smart technologies and embedded artificial intelligence. The major challenges addressed in this project are described below:

Dependence on Physical Input Devices

- Most multimedia systems rely on keyboards, remote controls, touchscreens, or mouse-based interaction.
- These devices require direct physical contact, which limits flexibility and convenience.
- Continuous physical interaction may not be suitable for modern smart environments where seamless and natural interaction is expected.
- Physical controllers also introduce wear and maintenance issues over time

Lack of Touchless Interaction

- Traditional systems do not support natural human gestures as an input method.
- In scenarios such as presentations, smart homes, or interactive displays, users often require hands-free control.
- Touch-based interfaces restrict usability when users are at a distance from the device or when physical access is limited.
- A touchless interface improves accessibility for elderly and physically challenged users.

High Latency in Cloud-Based Solutions

- Many existing gesture recognition systems depend on cloud processing.
- Sending video data to remote servers introduces communication delay.
- Network dependency affects real-time responsiveness.
- Internet connectivity issues may cause system failure or reduced performance.

High Power Consumption of Conventional Systems

- Desktop-based AI systems require powerful processors and high energy consumption.
- Real-time gesture recognition requires fast local processing.
- Edge computing reduces latency by processing data directly on the device.
- Embedded platforms must handle computer vision workloads efficiently without cloud assistance.

Requirement for ARM-Based Efficient Processing

- ARM architecture provides high performance per watt, making it ideal for embedded AI systems.
- An ARM-based platform enables continuous real-time operation with low power consumption.
- Utilizing ARM processors allows scalable deployment in smart and portable devices.

BACKGROUND (CONVERTED FROM EDGE AI)

1. Computer Vision

Computer vision is a field of artificial intelligence that enables computers to interpret and analyse visual information from images and video streams. It combines image processing and machine learning techniques to detect objects, track motion, and recognise human actions. Computer vision is widely used in applications such as surveillance, robotics, healthcare, and intelligent user interfaces.
In this project, computer vision enables touchless media control using hand gesture recognition. A USB camera captures real-time video frames for processing. OpenCV is used for image acquisition and preprocessing operations. MediaPipe is employed to detect hand landmarks and track finger positions accurately. The system analyzes spatial relationships between detected landmarks to identify gestures. Recognized gestures are mapped to media control commands such as play, pause, and volume adjustment.

2. Role of ARM Architecture

ARM architecture is widely used in embedded systems due to its low power consumption and efficient performance. Unlike traditional processors designed for high power computing, ARM processors are optimized to deliver high performance per watt, making them suitable for portable and edge computing devices.
The proposed system is implemented on the NVIDIA Jetson Nano, which uses a quad-core ARM Cortex-A57 processor. The ARM CPU manages operating system tasks, video processing, and gesture recognition simultaneously, enabling real-time performance on a compact embedded platform.
By processing data locally, ARM-based systems reduce latency and dependency on cloud services. In this project, the ARM architecture provides efficient execution of computer vision algorithms while maintaining energy efficiency, demonstrating its suitability for embedded AI applications.

3. Edge AI in Embedded Systems

Edge AI refers to performing artificial intelligence processing directly on local devices instead of cloud servers. This approach reduces latency, improves data privacy, and enables faster response times for real-time applications. Edge AI is particularly important for systems that require immediate interaction and continuous operation.
In this project, gesture recognition is performed locally on the Jetson Nano using real-time camera input.
The combination of edge AI and ARM-based processing allows efficient execution of AI tasks with low power consumption. This demonstrates how embedded platforms can support intelligent and responsive human–computer interaction systems suitable for smart environments and future edge computing applications.

SYSTEM ARCHITECTURE

Overall System Overview

The proposed system architecture is designed to enable touchless media control through real-time hand gesture recognition using an ARM-based embedded platform. The system integrates hardware components such as a USB camera and the NVIDIA Jetson Nano with software frameworks including OpenCV and MediaPipe. The camera captures live video input, which is processed locally on the Jetson Nano to detect gestures and generate control commands. The architecture ensures efficient interaction between input acquisition, gesture processing, and output execution, enabling seamless human–computer interaction without physical contact.

Input Acquisition Module

The input acquisition module is responsible for capturing real-time video data from the user environment. A USB camera connected to the Jetson Nano continuously records video frames and transfers them to the processing unit. The captured frames serve as the primary input for gesture recognition. This module ensures consistent frame capture with minimal delay, allowing the system to respond quickly to user hand movements.

Image Processing Module

The image processing module prepares captured video frames for gesture analysis. OpenCV is used to perform operations such as frame resizing, color conversion, and noise reduction to improve detection accuracy. Preprocessing ensures that frames are optimized for efficient computation while maintaining real-time performance. This stage reduces unnecessary data complexity and improves the reliability of subsequent hand detection processes.

Hand Landmark Detection

Hand landmark detection is performed using the MediaPipe framework, which identifies key points on the hand, including finger joints and palm positions. The framework analyses each frame and generates landmark coordinates that represent the structure of the hand. These landmarks provide spatial information required to understand finger orientation and hand posture. Accurate landmark detection enables precise gesture recognition in real time.

Gesture Recognition Module

The gesture recognition module analyzes the detected hand landmarks to identify predefined gestures. The system compares finger positions and relative distances between landmarks to classify gestures such as open palm, closed fist, or directional movement. Logical conditions are applied to determine the intended command based on hand posture. This module converts visual input into meaningful interaction commands.

Command Processing Module

Once a gesture is recognized, it is translated into a corresponding media control command. The command processing module maps gestures to actions such as play, pause, volume control, and navigation. These commands are generated programmatically and prepared for execution by the media player application. This module acts as the bridge between gesture recognition and system response.

RM-Based Processing Unit

The NVIDIA Jetson Nano serves as the central processing unit powered by an ARM Cortex-A57 processor. The ARM architecture manages operating system tasks, video processing, and gesture recognition simultaneously while maintaining energy efficiency. Local processing enables real-time performance without cloud dependency, demonstrating the effectiveness of ARM-based platforms for edge AI applications.

Output Execution Module

The output execution module delivers the final system response by sending control commands to the media player application. Based on recognized gestures, the system performs actions such as starting or pausing playback and adjusting volume levels. The immediate execution of commands ensures smooth and interactive user experience.

Edge Processing Workflow

The entire system operates using edge computing principles, where all processing occurs locally on the Jetson Nano. This eliminates the need for internet connectivity and reduces latency associated with cloud processing. Edge processing improves data privacy, system reliability, and power efficiency, making the solution suitable for real-time embedded applications.

HARDWARE DESIGN

The hardware design of the proposed system is centered around the NVIDIA Jetson Nano Developer Kit, an ARM-based embedded computing platform designed for edge artificial intelligence applications. Unlike FPGA-based acceleration used in some embedded AI systems, this project utilizes heterogeneous computing through an ARM Cortex-A57 CPU combined with an integrated NVIDIA Maxwell GPU. This architecture enables efficient execution of computer vision and gesture recognition tasks while maintaining low power consumption suitable for embedded environments.

The Jetson Nano is powered by the NVIDIA Tegra X1 System-on-Chip, which integrates a quad-core ARM Cortex-A57 processor responsible for operating system management, data handling, and gesture processing logic. The ARM processor efficiently manages multiple system operations such as camera input acquisition, frame preprocessing, and command execution. Its energy-efficient architecture allows continuous real-time processing without excessive thermal generation, making it ideal for edge computing applications.

In addition to the ARM CPU, the platform includes a 128-core NVIDIA Maxwell GPU that accelerates parallel computation required for image processing and AI inference. GPU acceleration enables faster processing of video frames and improves gesture recognition performance. CUDA-based acceleration is utilized to optimize computational tasks, allowing real-time execution of computer vision algorithms. The Jetson Nano hardware handles video decoding, AI inference, and gesture mapping locally, eliminating dependency on external cloud processing and ensuring low latency operation.

The system also incorporates a USB camera as the primary input device for capturing live video frames. A regulated power supply through the barrel jack ensures stable system performance during continuous processing. The combination of ARM-based processing, GPU acceleration, and optimized hardware integration enables the implementation of a responsive and energy-efficient touchless media control system suitable for real-world embedded applications.

SOFTWARE INTEGRATION

The software integration of the proposed system combines multiple software frameworks and development tools to enable real-time gesture recognition and touchless media control on an ARM-based embedded platform. The system operates on Ubuntu Linux provided through the NVIDIA JetPack Software Development Kit, which includes optimized drivers, libraries, and GPU acceleration support specifically designed for the Jetson Nano architecture. Python is used as the primary programming language due to its simplicity, flexibility, and strong ecosystem for computer vision and artificial intelligence development.

The computer vision pipeline is implemented using OpenCV and MediaPipe frameworks. OpenCV is responsible for camera interfacing, frame acquisition, preprocessing operations, and visualization of processed frames. Captured video frames are resized and formatted to ensure efficient processing while maintaining real-time performance. MediaPipe is employed for hand tracking and landmark detection, where multiple key points representing finger joints and palm structure are extracted from each frame. These landmarks provide spatial information that enables accurate analysis of hand posture and movement.
The gesture recognition module processes the detected landmark coordinates using predefined logical conditions to classify user gestures. Each recognized gesture is mapped to a specific media control command such as play, pause, forward navigation, backward navigation, or volume adjustment. The gesture-to-command mapping is implemented using Python-based control logic to ensure reliable and responsive interaction.

For multimedia playback, the system utilizes the mpv media player running on Ubuntu . MPV is selected due to its lightweight architecture, better compatibility with ARM64 systems, and efficient command-line controllability, making it highly suitable for embedded platforms like the Jetson Nano. Gesture-generated commands are transmitted to mpv using system-level control interfaces, allowing real-time execution of playback operations. The integration of mpv ensures smooth video playback with minimal latency and improved stability during continuous operation.
Overall, the integration of Ubuntu, Python, OpenCV, MediaPipe, and mpv creates a unified software environment capable of performing real-time edge AI processing. The system operates entirely on-device without cloud dependency, reducing latency, improving privacy, and optimizing resource utilization. This software architecture demonstrates the capability of ARM-based platforms to support intelligent human–computer interaction systems efficiently in embedded environments.

OPTIMIZATION TECHNIQUES

Efficient system performance is essential for real-time gesture recognition on embedded platforms where computational resources and power availability are limited. Several optimization techniques were implemented to improve frame rate, reduce latency, and ensure stable operation of the gesture-based media control system on the ARM-based NVIDIA Jetson Nano. These optimizations focus on hardware utilization, software efficiency, and resource management to achieve reliable real-time performance.

Performance Mode Optimization

The Jetson Nano supports multiple power modes that control CPU and GPU frequency levels. To achieve maximum computational performance, the system was configured to operate in MAXN performance mode using the nvpmodel utility. This mode enables the ARM Cortex-A57 processor and GPU to run at higher clock frequencies, improving processing speed during gesture recognition. Additionally, the jetson_clocks command was used to lock system clocks at maximum frequency, preventing dynamic throttling and ensuring consistent performance during continuous execution.

GPU Acceleration Utilization

Although gesture logic runs on the ARM CPU, computationally intensive image processing operations benefit from GPU acceleration. The NVIDIA Maxwell GPU integrated within the Tegra X1 SoC assists in parallel processing tasks and optimized multimedia handling. CUDA-enabled libraries and JetPack optimizations allow faster frame handling and efficient execution of computer vision pipelines, reducing CPU workload and improving overall system responsiveness.

Frame Resolution Optimization

Processing high-resolution video frames significantly increases computational load and reduces frame rate on embedded devices. To balance performance and accuracy, the input video resolution was reduced to an optimal size during preprocessing. This reduction minimizes unnecessary pixel processing while preserving sufficient detail for accurate hand landmark detection. As a result, gesture recognition achieves improved frame rates with minimal impact on detection accuracy.

Efficient Thread Management

The system was designed to separate video capture and gesture processing tasks to improve execution efficiency. Efficient threading allows frame acquisition and gesture recognition to operate concurrently rather than sequentially. This reduces processing delay and prevents frame drops during real-time operation. Proper task distribution across ARM CPU cores ensures better utilization of available processing resources.

Hardware-Accelerated Video Playback

To ensure smooth media playback, the lightweight mpv media player was used instead of heavier multimedia applications. MPV provides efficient command-line control and better compatibility with ARM64 architecture. Hardware-supported decoding and optimized playback pipelines reduce system overhead, allowing gesture recognition and media playback to run simultaneously without performance degradation.

Power and Thermal Optimization

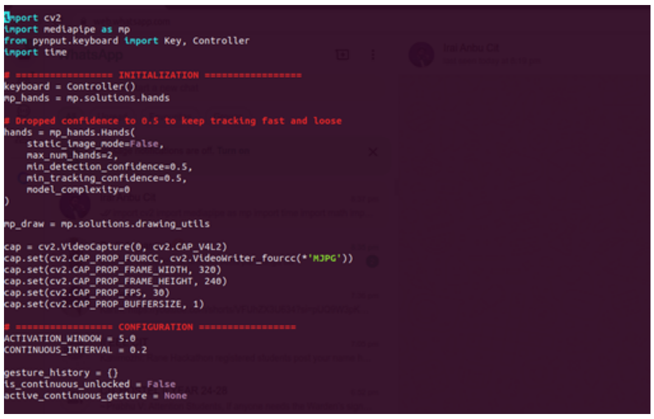
Stable power delivery through the barrel jack power supply was used to prevent system throttling caused by insufficient current. Proper power management ensures that CPU and GPU frequencies remain stable during operation. Maintaining controlled thermal conditions improves reliability and prevents performance drops during prolonged system usage.

METHODOLOGY

The methodology describes the step-by-step implementation process used to develop the touchless media control system using hand gesture recognition on the ARM-based NVIDIA Jetson Nano platform. The system workflow consists of multiple stages, starting from video acquisition and ending with media command execution. Each stage performs a specific function to ensure accurate gesture detection and real-time system response.

9.1 **Camera Initialization**

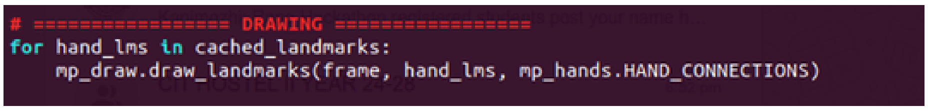
The first stage of the system involves initializing the USB camera connected to the Jetson Nano. The camera is accessed using the OpenCV library, which enables real-time video capture. Parameters such as frame resolution and frame rate are configured to balance performance and processing efficiency. Continuous frame acquisition allows the system to monitor user hand movements without interruption. The captured frames serve as the primary input for gesture recognition processing.



```
import cv2
import mediapipe as mp
from pynput.keyboard import Key, Controller
import time

# ================= INITIALIZATION =================
keyboard = Controller()
mp_hands = mp.solutions.hands

# Dropped confidence to 0.5 to keep tracking fast and loose
hands = mp_hands.Hands(
    static_image_mode=False,
    max_num_hands=2,
    min_detection_confidence=0.5,
    min_tracking_confidence=0.5,
    model_complexity=0
)

mp_draw = mp.solutions.drawing_utils

cap = cv2.VideoCapture(0, cv2.CAP_V4L2)
cap.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*'MJPG'))
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 320)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 240)
cap.set(cv2.CAP_PROP_FPS, 30)
cap.set(cv2.CAP_PROP_BUFFERSIZE, 1)

# ================= CONFIGURATION =================
ACTIVATION_WINDOW = 5.0
CONTINUOUS_INTERVAL = 0.2

gesture_history = {}
is_continuous_unlocked = False
active_continuous_gesture = None
```

Hand Tracking using MediaPipe

After capturing video frames, the system performs hand detection using the MediaPipe framework. MediaPipe identifies the presence of a hand in each frame and tracks it dynamically. The framework uses machine learning models to locate the hand region and maintain stable tracking even during movement. This step ensures that only relevant hand regions are processed, improving detection accuracy and reducing computational overhead.



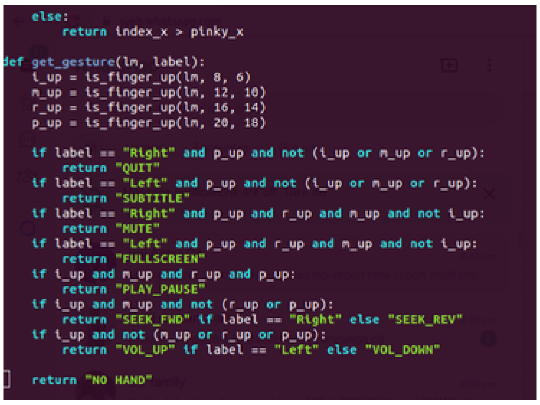
```
# ================= DRAWING =================
for hand_lms in cached_landmarks:
    mp_draw.draw_landmarks(frame, hand_lms, mp_hands.HAND_CONNECTIONS)
```

Landmark Extraction

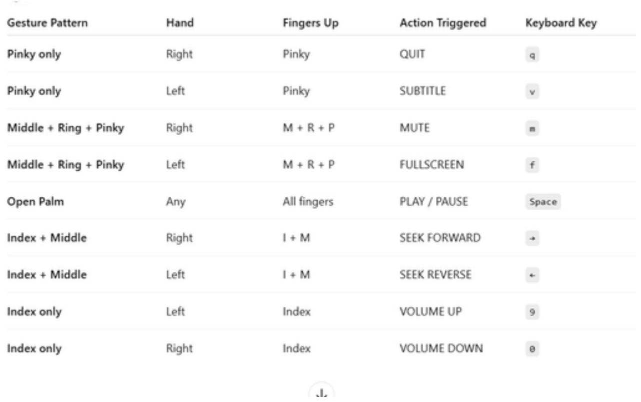
Once the hand is detected, MediaPipe extracts multiple landmark points representing finger joints and palm positions. Each landmark contains spatial coordinates that describe the structure of the hand. These coordinates are used as features for gesture analysis. Landmark extraction provides precise positional data required to differentiate between various hand gestures in real time.

Gesture Classification Logic

The extracted landmark coordinates are analyzed using predefined logical conditions to classify gestures. Finger positions and relative distances between landmarks are compared to determine the intended gesture. This rule-based classification approach enables fast processing suitable for embedded ARM platforms while maintaining reliable gesture recognition performance.



```
    else:
        return index_x > pinky_x

def get_gesture(lm, label):
    i_up = is_finger_up(lm, 8, 6)
    m_up = is_finger_up(lm, 12, 10)
    r_up = is_finger_up(lm, 16, 14)
    p_up = is_finger_up(lm, 20, 18)

    if label == "Right" and p_up and not (i_up or m_up or r_up):
        return "QUIT"
    if label == "Left" and p_up and not (i_up or m_up or r_up):
        return "SUBTITLE"
    if label == "Right" and p_up and r_up and m_up and not i_up:
        return "MUTE"
    if label == "Left" and p_up and r_up and m_up and not i_up:
        return "FULLSCREEN"
    if i_up and m_up and r_up and p_up:
        return "PLAY_PAUSE"
    if i_up and m_up and not (r_up or p_up):
        return "SEEK_FWD" if label == "Right" else "SEEK_REV"
    if i_up and not (m_up or r_up or p_up):
        return "VOL_UP" if label == "Left" else "VOL_DOWN"

    return "NO HAND"
```



Gesture Pattern	Hand	Fingers Up	Action Triggered	Keyboard Key
Pinky only	Right	Pinky	QUIT	q
Pinky only	Left	Pinky	SUBTITLE	v
Middle + Ring + Pinky	Right	M + R + P	MUTE	m
Middle + Ring + Pinky	Left	M + R + P	FULLSCREEN	f
Open Palm	Any	All fingers	PLAY / PAUSE	Space
Index + Middle	Right	I + M	SEEK FORWARD	→
Index + Middle	Left	I + M	SEEK REVERSE	←
Index only	Left	Index	VOLUME UP	9
Index only	Right	Index	VOLUME DOWN	0

Media Control Mapping

After gesture classification, the recognized gesture is mapped to corresponding multimedia control commands. Commands such as play, pause, forward, backward, and volume adjustment are generated programmatically. These commands are sent to the mpv media player through system-level control interfaces, allowing real-time execution of media operations without physical interaction.

RESULTS AND PERFORMANCE ANALYSIS

The performance of the proposed touchless media control system was evaluated based on real-time responsiveness, gesture recognition accuracy, system stability, and resource utilization on the ARM-based NVIDIA Jetson Nano platform. The objective of the evaluation was to verify whether the system could perform gesture recognition efficiently while maintaining smooth multimedia playback under embedded hardware constraints.

The system successfully achieved real-time gesture recognition using live video input from a USB camera. After applying optimization techniques such as performance mode configuration and resolution adjustment, the system maintained an average frame rate between 13–18 frames per second (FPS). This frame rate provided smooth interaction and acceptable responsiveness for media control applications. Gesture recognition latency was significantly reduced, allowing commands to be executed almost instantly after gesture detection.

CPU and GPU utilization were monitored during execution to analyze system efficiency. The ARM Cortex-A57 processor handled video capture, gesture logic, and command execution, while GPU acceleration supported multimedia processing and optimized frame handling. Balanced resource utilization ensured that the system operated without excessive overheating or performance throttling. Stable operation was achieved after switching to barrel jack power supply, which prevented power-related performance drops.

The integration of the mpv media player further improved playback stability compared to earlier implementations. MPV enabled lightweight media execution with minimal system overhead, allowing simultaneous gesture recognition and video playback without noticeable lag. The system demonstrated reliable command execution for operations such as play, pause, forward navigation, and volume adjustment.

Overall, the experimental results confirm that ARM-based embedded platforms are capable of supporting real-time computer vision applications when proper optimization techniques are applied. The proposed system achieved efficient edge AI processing with low latency and stable performance, demonstrating the feasibility of implementing touchless human–computer interaction systems on resource-constrained embedded hardware.

DESIGN TRADE-OFFS

During the implementation of the touchless media control system on the ARM-based NVIDIA Jetson Nano platform, several design trade-offs were considered to balance performance, accuracy, power efficiency, and system stability. Due to the limited computational resources of embedded systems, optimization decisions were necessary to achieve reliable real-time operation.

Resolution vs Processing Performance

Higher video resolution improves gesture detection accuracy by providing more visual detail; however, it increases computational load and reduces frame rate. To maintain smooth real-time interaction, an optimal resolution was selected that balances detection accuracy with processing speed.

Performance vs Power Consumption

Enabling MAXN performance mode increased CPU and GPU frequencies, improving gesture recognition responsiveness. However, this also increased power consumption and thermal output. A stable power supply and controlled performance configuration were used to maintain efficient system operation.

Media Player Selection (VLC vs MPV)

Initially, VLC media player was used but showed compatibility and performance issues on the ARM64 platform. The system was transitioned to the lightweight mpv media player, representing a trade-off between advanced features and improved stability and efficiency.

Edge Processing vs Cloud Processing

Cloud-based processing offers higher computational capability but introduces latency and network dependency. Local edge processing on the Jetson Nano reduced response time and improved privacy, though it required optimization to work within hardware limitations.

Accuracy vs Real-Time Responsiveness

Complex gesture models can increase accuracy but also increase processing delay. A rule-based gesture recognition approach was adopted to ensure faster execution suitable for real-time embedded applications.

CONCLUSION

The proposed project successfully demonstrates the implementation of a touchless media control system using real-time hand gesture recognition on an ARM-based embedded platform. By leveraging the NVIDIA Jetson Nano powered by an ARM Cortex-A57 processor, the system achieves efficient edge AI processing capable of interpreting human gestures and converting them into multimedia control commands without physical interaction. The integration of computer vision techniques using OpenCV and MediaPipe enabled accurate hand landmark detection and reliable gesture classification in real time.

Through careful optimization techniques, including performance mode configuration, resolution adjustment, and lightweight media playback using mpv, the system achieved stable performance with improved frame rate and reduced latency. The implementation highlights how ARM architecture supports energy-efficient computing while maintaining sufficient processing capability for real-time artificial intelligence applications. Operating entirely on-device eliminated dependency on cloud processing, resulting in faster response time, improved privacy, and enhanced system reliability.

Overall, the project validates the feasibility of deploying intelligent human–computer interaction systems on resource-constrained embedded platforms. The developed solution demonstrates the practical advantages of ARM-based edge computing for modern touchless interfaces and smart environments. This work emphasizes the growing role of ARM architecture in enabling scalable, low-power, and real-time AI solutions for next-generation embedded systems.