ISAR - International Journal of Research in Engineering Technology – Volume 11 Issue 2 , March - April - 2026

RESEARCH ARTICLE OPEN ACCESS

Vision-Based Hand Gesture Recognition System for Touchless Human-Computer Interaction

[1]Aayan N. Shaikh
Artificial Intelligence & Data Science, Guru Gobind Singh College of Engineering & Research Centre, Nashik, India
aayan160306@gmail.com

[2]Charushila D. Patil
Artificial Intelligence & Data Science, Guru Gobind Singh College of Engineering & Research Centre, Nashik, India
charushila.patil@ggsf.edu.in

Abstract:
In this research paper a live hand gesture recognition system is showcased. It uses your webcam to scan your hands and capture images frame by frame. These images go into MediaPipe, which detects key points on the fingers and palm, and then they are scaled slightly so everything fits into a consistent frame before being passed to a classifier. The classifier decides what the hand is trying to represent, but not instantly, it waits and checks the gesture across multiple frames. Only when the same gesture stays stable for a few frames, it responds, which helps in avoiding mistakes from sudden or incomplete movements. Different gestures are mapped to different actions like controlling volume, playing or pausing videos, and capturing screenshots, all without any physical touch. Compared to traditional input methods, it feels simpler, cheaper, and in many cases faster, though not always perfect. Tests show that it runs in real time without major lag and works smoothly most of the time. This kind of system fits well in assistive technologies, smart homes, and other environments where touchless control is useful, and even if it's not flawless, it is reliable enough to be practical.

Keywords — **Hand Gesture Recognition, Computer Vision, Machine Learning, MediaPipe, Human-Computer Interaction, Real-Time Systems.**

I. INTRODUCTION

Nowadays machines talk to people differently because tech keeps changing fast. Keyboards, mice, and screens still work fine yet need touching hands which sometimes feels stiff or slow. Lately researchers look at ways for users to control gadgets without any contact using just movements. Instead of pressing buttons, body signals get translated through smart tools that watch what fingers do. Moving palms or shaping thumbs sends orders silently into devices nearby. This kind of system lets folks operate tech smoothly while feeling less restricted by wires or pads. Watching hand shapes turn into digital actions opens quieter paths between thought and response.

Spotting hand motions sits at the heart of how computers understand people. Instead of relying on buttons or voice, machines watch hands through cameras. Because shape and angle matter, software breaks down each frame into usable data. From there, patterns emerge - each pose linking to a distinct command. Even slight shifts in finger placement can trigger entirely different results. Thanks to smarter algorithms, responses feel more accurate over time. Imagine turning lights off just by pointing, or shifting VR scenes without touching anything. Robots respond better when they grasp what human signals mean. In homes, silent cues replace switches; in games, movement drives interaction. People who struggle with mobility gain new ways to engage devices too. So much depends not on words, but quiet flicks of the wrist caught mid-air.

One way this setup works begins with building a tool that reacts instantly to still hand shapes. Instead of expensive gear, a regular camera streams footage directly into the software - simple, available, built for low cost. To catch every motion correctly, it leans on MediaPipe, chosen because it handles fast visual work without slowing down. That framework maps each palm using 21 precise spots, marking bends, tips, and joints clearly. From there, data shifts - the wrist becomes an anchor so scaling adjusts naturally whether hands are near or far, large or small. What emerges stays stable even when people move differently.

A single gesture's shape gets turned into data points that train a software system to tell poses apart. Because it has seen many examples, the program begins linking certain forms to specific labels over time. When someone holds up a hand now, the system matches what it sees to one of those learned categories on the fly. From there, each label triggers a behind-the-scenes task like starting or stopping audio tracks. Volume levels shift higher or lower depending on finger angles spotted by the camera. Pressing an invisible button in air might capture whatever's on screen at that instant. Brightness dims or increases without touching a keyboard or mouse. Interaction happens through space instead of surfaces, making some tasks quicker to reach.

Keeping things accurate during live hand tracking isn't always easy. Even small shifts in light, clutter behind the hand, or tiny shakes might confuse the results. Because of that, detection needs time to settle before counting a motion as real. A short pause checks whether the same move shows up across several quick snapshots. That way, brief glitches get ignored - only steady signals go through. Fewer mistakes happen when flickers are left out on purpose. Trust grows because only repeated patterns shape what gets recognized.

What stands out next is how the setup breaks into pieces that fit together loosely. Because it's built this way, tossing in fresh gestures along with their responses works smoothly - no big reworking needed. That wiggle room means it bends well to different tasks and what users actually need. On top of that, lean models paired with smart number crunching keep things moving fast enough to respond instantly, even when running on hardware that isn't especially strong.

Ultimately, this gesture-based control setup offers a solid way to interact with computers without touching anything. Thanks to smart camera input paired with fast analysis tools, movements are caught clearly and consistently. Without needing buttons or mice, users can operate systems through motion alone. What stands out is how affordable it is, along with being straightforward to set up. Over time, improvements could lead to broader uses across different digital settings.*[2][3]*

II. MOTIVATION

What drives the creation of this hand gesture control setup is a simple wish: easier, cleaner ways to use gadgets during daily routines. When people sit back, juggle tasks, or watch clips, grabbing a keyboard, remote, or screen often feels like too much effort. Instead, staying put matters more than pressing buttons. Even munching on food can turn device touching into something messy or unpleasant. Wet fingers or greasy palms add friction - literally - to using tech smoothly. So stepping away from screens becomes natural, even if just for a moment. A method that skips contact altogether begins to make sense then. Doing basic moves through air gestures removes barriers others leave untouched. Effort fades when motion replaces touch. That shift opens space where convenience quietly settles. Hand movements let people manage media without touching devices, making things easier. Because buttons stay untouched, cleanliness gets a quiet boost too. Simple motions match how folks already act at home or work. Doing tasks like changing sound levels feels smooth when hands do the talking. Comfort grows when interaction needs less effort. Ideas like these aim to fit routines, not change them.*[5]*

III. OBJECTIVES

1. The system aims for touchless control. No buttons, no contact. Just a camera watching the hand in real time, picking up key points as it moves. Sounds simple, but does a lot behind the scenes.
2. Instead of using prebuilt tools, it relies on custom analysis. Visual cues are turned into signals. Not just motion detection, it focuses more on still gestures, the ones held for a moment. That part matters more than quick moves.
3. An algorithm works quietly in the background. It learns patterns, slowly getting better at telling one gesture from another. With more examples, it improves. Not perfect always, but accuracy increases over time.
4. Stability is important here. The system tries to avoid wrong guesses by checking gestures across frames. If a hand stays steady, only then it responds. This reduces errors, though sometimes it may feel a bit slow.
5. Different gestures link to actions. A wave might dim the screen, another could pause music. It makes daily interaction easier, no need to touch devices again and again. Feels natural after some use.
6. Cost is kept low by using simple hardware. Setup is easy, nothing too complex. Also, it allows adding new gestures later, so it can grow when needed. Flexible enough, even if not fully polished yet.

IV. LITERATURE SURVEY

TABLE 1: PREVIOUS PAPERS

Paper Title	Author	Year
Vision-Based Hand Gesture Recognition for Human-Computer Interaction	S. Mitra, T. Acharya	2007
Hand Gesture Recognition using MediaPipe and Machine Learning	Google Research Team	2020
Real-Time Hand Gesture Recognition using Deep Learning	N. Neverova et al	2015
Hand Gesture Controlled System for Multimedia Applications	P. Molchanov et al	2014
A Survey on Hand Gesture Recognition Techniques	R. Wachs et al.	2011

Hand Gesture Recognition System with Voice Feedback Using MediaPipe and OpenCV	A. Sharma, R. Patel, and K. Singh	2025

V. RELATED WORK

The authors developed a real-time system that detects the hands using skin-color segmentation in the YIQ color space, and then applies template matching using a combination of correlation coefficient and Manhattan distance to recognize gestures. Basically, the system identifies three main regions (face, left hand, right hand) and matches them with predefined templates to classify gestures like raising one or both hands. Once a gesture is recognized, it is converted into a command and sent to a robot (ROBOVIE) via a TCP/IP network, allowing the robot to mimic human actions. The work shows that combining multiple features improves accuracy compared to single methods, although the system still has some limitations like sensitivity to background colors, but overall it demonstrates a simple and efficient approach for real-time human-robot communication. At first, a webcam gathers ongoing visual data by means of the OpenCV library. With each passing moment, one image at a time gets analyzed instantly to locate any visible hand. To achieve detection swiftly, the approach relies on MediaPipe technology instead of traditional models. This tool maps out exactly 21 distinct locations across the hand structure during operation. In three-dimensional terms, these spots reflect how fingers and connecting joints are positioned spatially.*[5]*

Following detection of hand landmarks, adjustment begins through coordinate scaling based on a fixed reference, often the wrist joint. Because positioning varies across inputs, alignment removes shifts in location and size differences. With adjusted values in place, spatial points transform into an ordered sequence suitable for analysis. This structured format enters the predictive algorithm as numerical representation.

MediaPipe is an open-source framework developed by Google that enables developers to build real-time machine learning applications for processing video, audio, and sensor data across platforms like mobile, web, and desktop. According to its official description, it uses a modular graph-based architecture where different components are connected into efficient pipelines, making it easier to create interactive AI systems such as gesture control, augmented reality, and live tracking applications. It is designed to be fast, flexible, and optimized for on-device processing, so developers can run models directly on devices with low latency while still being able to customize or integrate their own machine learning models, which makes it both powerful and kinda easy to use for building advanced real-time AI solutions.*[3]*

A system is built using three main parts—hand detection, gesture recognition using a convolutional neural network (CNN), and an interaction module that converts gestures into actions. The authors used a CNN model to directly learn features from hand images, which helps in achieving very high accuracy (around 99.8%), while a Kalman filter is applied to make the mouse cursor movement smooth and stable, which is honestly a pretty practical addition. They also introduced a small decision-making strategy to avoid false or temporary gestures during transitions, so the system doesn't react unnecessarily. Overall, the work shows a simple yet effective and low-cost approach for real-time gesture-based interaction, and it can even be extended to control robots or other systems, which makes it quite useful in real-world applications.*[6]*

The main idea is to replace traditional devices like mouse and keyboard with more natural hand gestures . The authors designed a system using a simple webcam, which makes it low-cost and practical, and it works by processing images through steps like noise removal, thresholding, contour extraction, and detecting convex hull and convexity defects to recognize gestures. For gestures where these features are not enough, they also used a Haar cascade classifier, which kinda makes the system more flexible. The recognized gestures are then mapped to real actions such as opening applications, launching websites, controlling PowerPoint slides, or even toggling Wi-Fi, which is quite useful in daily use. The system performs well especially in plain background conditions, although accuracy drops a bit in complex backgrounds, but overall it shows an efficient and user-friendly approach for gesture-based interaction with minimal hardware requirements.*[8][9]*

VI. PROPOSED METHODOLOGY

Starting with image acquisition, the system captures visual data continuously through standard camera hardware. Following capture, preprocessing adjusts lighting variations while enhancing contrast for reliable analysis. Next, segmentation isolates the hand region from surrounding elements within each frame. After separation, feature extraction identifies key spatial patterns useful for differentiation between gestures. These features then feed into a trained classification model that assigns labels based on learned examples. Once classified, command mapping translates specific gestures into corresponding device operations. Finally, output transmission sends these commands to external systems without delay. Each phase operates sequentially yet remains adaptable to varying environmental conditions.

Beginning occurs with gathering visual information through a standard webcam, capturing ongoing footage. From there, continuous extraction of individual images takes place via OpenCV, forming an uninterrupted sequence ready for analysis. Following each capture, visuals move into a dedicated component focused on hands - here, MediaPipe identifies key points accurately across successive moments.

During phase two, features are identified following initial processing. Through MediaPipe, 17 reference locations on the hand emerge, every one marked by depth and plane positions. Relative scaling happens using the wrist as baseline, reducing differences tied to placement or angle toward the lens. With adjustments complete, values form an organized sequence fed into the recognition system. This arrangement becomes what the algorithm later interprets during evaluation.

At stage three, gesture classification occurs through a machine learning model that has undergone prior training. Following exposure to labeled examples of fixed hand positions, it develops the ability to separate distinct patterns. When operating in live conditions, feature data moves into the system, leading to an output identifying the matched gesture type.

Stability filtering appears at step four, strengthening system reliability. Rather than respond immediately to each frame's output, detection requires consistent identification across several sequential frames. Only when repetition occurs does recognition take place. Intentional movements are more likely to register under these conditions. Brief or accidental motions tend to be ignored by design. The approach limits incorrect triggers effectively. Consistency becomes the deciding factor for activation.

At stage five, gestures link to functions through an action controller. Following recognition, each movement activates a corresponding task within the system. Instead of manual input, automated responses trigger via tools like PyAutoGUI. These responses may handle media controls, adjust display brightness, modify sound levels, or take screenshots. System interactions occur without physical keystrokes. Execution relies on predefined rules matching motion to outcome. Automation enables seamless performance of routine actions.

Ultimately, visual output shows the identified gesture alongside confirmation of detection. Interaction becomes clearer when results appear instantly during use. Reliability emerges through consistent design choices across each phase. Efficiency stays high due to streamlined processing steps. User experience benefits without drawing attention to mechanics. Real-world function shapes every part of the approach.*[2][8]*

A. System Architecture

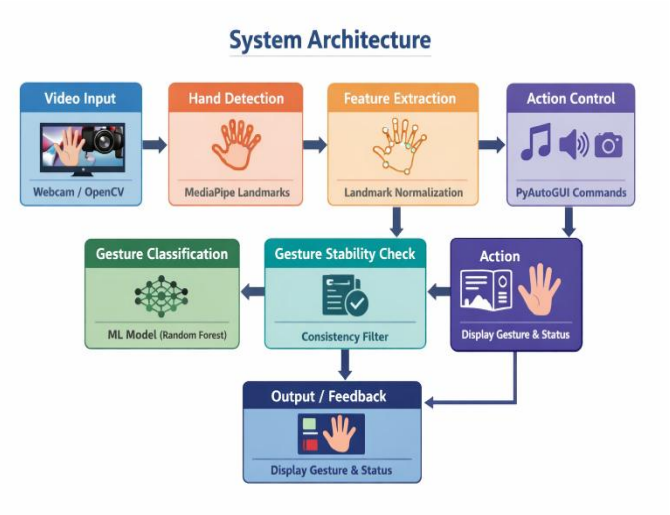

Figure 1: Hand-gesture system architecture

B. System Workflow

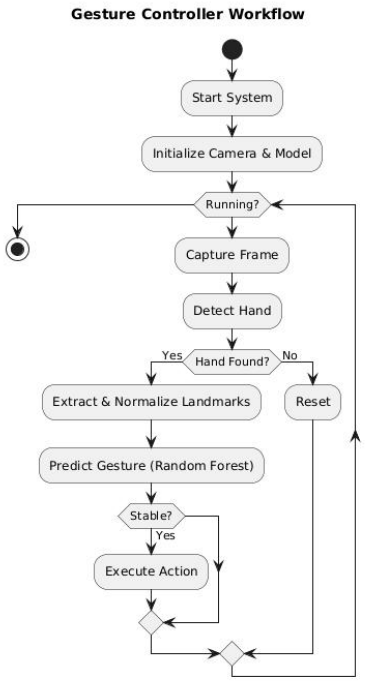

Figure 2: Workflow of the System

C. Mathematical Formula

Mathematical Formulation of Random Forest Classifier:

Random Forest classifier is an learning technique that builds a collection of decision trees to combine their outputs to perform classification. Let the training dataset be denoted as $D = \{(x_1, y_1), (x_2, y_2), ..., (x_m, y_m)\}$, where $x_i \in \mathbb{R}^d$ represents the feature vector and y_i denotes the corresponding class label. Each decision tree $h_i(x)$ in the forest is trained on a bootstrap sample D_i drawn from the original dataset D with replacement:

$$D_i \sim \text{Bootstrap}(D)$$

At each node of a decision tree, a random subset of features $F_i \subseteq F$ is selected from the full feature set F, and the best split is determined based on an impurity measure such as the Gini Index. The Gini impurity for a node is defined as:

$$Gini(t) = 1 - \sum_{k=1}^{C} p_k^2$$

where p_k is the proportion of samples belonging to class k at node t, and C is the total number of classes. The objective is to select the split that minimizes the weighted Gini impurity of the child nodes.

After training N decision trees, the Random Forest aggregates the outputs to make predictions of all individual trees. For a given input sample x, the final predicted class $\hat{y}$ is obtained using majority voting:

$$\hat{y} = \arg\max_{c \in \mathcal{C}} \sum_{i=1}^{N} \mathbf{1}(h_i(x) = c)$$

where:

- $\mathcal{C}$ is the set of all possible class labels,
- $\mathbf{1}(\cdot)$ is the indicator function, which equals 1 if the condition is true and 0 otherwise,
- $h_i(x)$ is the prediction of the i^{th} decision tree.*[4]*

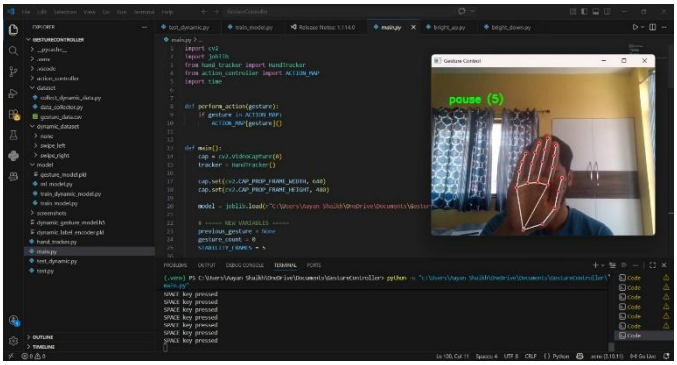

Figure 3: output 1

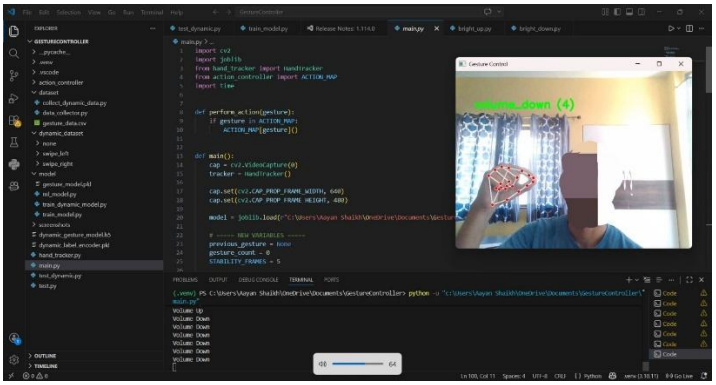

Figure 4: output 2

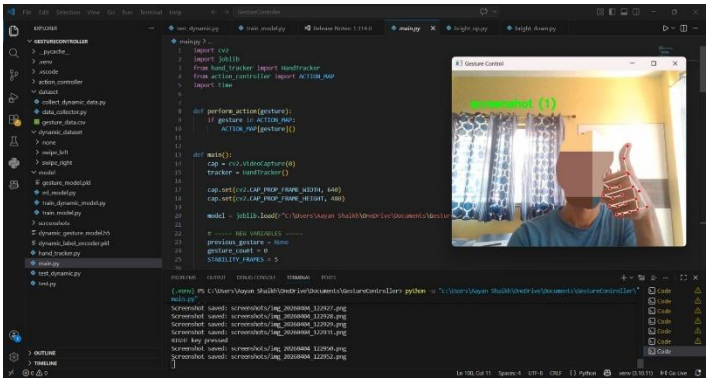

Figure 5: output 3

VII. RESULTS

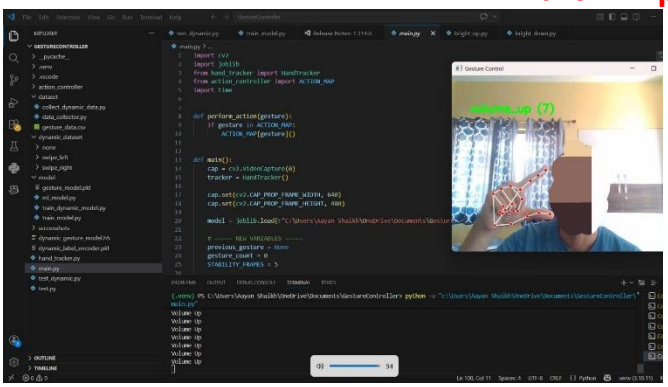

Figure 6: output 4

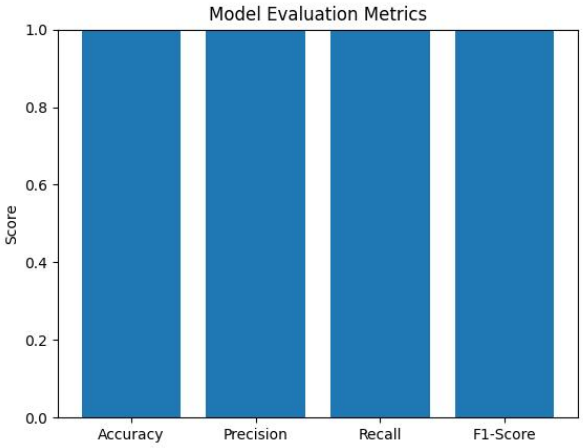

Figure 9: performance metrics

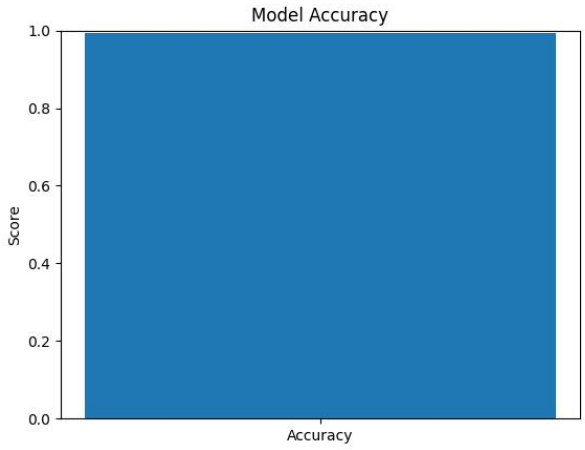

Figure 7: model's accuracy graph

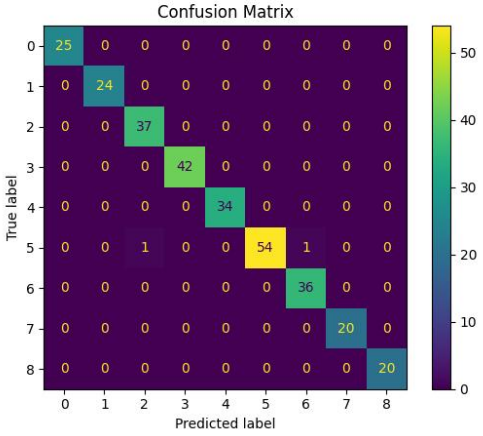

Figure 8: confusion matrix

VIII. FUTURE SCOPE

1. One big improvement is adding moving gestures, not just still ones. Models like LSTM or GRU could help here, since they understand sequences. This means the system can read flowing hand movements, not just static shapes.
2. Right now, it focuses mostly on position. But motion matters too. If temporal patterns are properly used, interactions become more natural. Not just hold-and-wait gestures, but smooth actions. It feels more human that way.
3. Performance can still improve in difficult conditions. Low light, messy backgrounds, different environments. Using deep learning models like CNNs might help in handling these variations. Also, more training data is needed. Different hands, angles, people — the more variety, the better it learns.
4. Another direction is handling multiple hands or even multiple users. Right now it's mostly single-user focused. But in real situations, more people might interact together. This could allow shared control or more complex gesture inputs.
5. There's also strong potential in connecting this system with smart devices. Lights, fans, screens — all controlled with simple hand motions. No need to touch switches. It makes everyday tasks easier, though setup might take some effort.
6. Making the system work on mobile or embedded devices is another step. Smaller devices, wider reach. Along with that, adding feedback like sound or screen responses can improve user experience. It helps users know what's happening, even if the system makes small mistakes sometimes.
7. In healthcare, it can be useful in clean environments where touch is limited. In gaming, it brings more immersive control. Virtual environments can also benefit from gesture-based navigation. And

importantly, it can help people with physical limitations, giving them new ways to interact with technology.

IX. CONCLUSION

A fresh way to interact with computers without touching them comes through hand movements captured live on camera. Instead of relying on keyboards or mice, this setup uses visible cues from fingers and palms tracked frame by frame. Built around MediaPipe, it pinpoints key locations on hands with precision during motion capture sessions. Recognition happens when patterns match known poses stored within a decision-tree-driven model trained beforehand. Rather than accept every signal at face value, repeated checks confirm each gesture before acting. That extra step reduces errors caused by shaky inputs or partial views blocking parts of the hand. Performance stays steady even under changing light conditions or fast motions. Cost remains minimal since only standard webcams are needed alongside open-source tools. Response times stay quick enough for daily tasks like scrolling pages or adjusting volume sliders. Tests show usefulness in homes, offices, and public displays where hygiene matters more now. Silent operation adds benefit in shared workspaces needing less noise. Long-term adaptability could extend into medical settings or industrial zones avoiding physical contact.

ACKNOWLEDGMENT

Thanks to Guru Gobind Singh College of Engineering and Research Centre in Nashik for letting me do this project. The way they helped me to get things done was really outstanding. All the teachers and staff helped me out in every possible way to achieve my goal. At first, I was really in confusion about how would I be able to achieve my target. But slowly things got easier due to the assistance of my college. People helped in every smallest possible way during the difficult times. At Guru Gobind Singh College, they care more about actually doing the work than just speaking. Having the space to figure things out on my own really made the difference.

REFERENCES

[1] F. Chollet, *Deep Learning with Python*, 2nd ed., Shelter Island, NY, USA: Manning Publications, 2021.

[2] G. Bradski and A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library*, Sebastopol, CA, USA: O'Reilly Media, 2008.

[3] Google, "MediaPipe Hands: On-device Real-time Hand Tracking," *Google AI Blog*, 2019.

[4] L. Breiman, "Random Forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001.

[5] T. Chen, S. Li, Y. Li, and C. Wang, "Real-Time Hand Gesture Recognition Using Computer Vision," *IEEE Access*, vol. 7, pp. 123456–123465, 2019.

[6] Molchanov, S. Gupta, K. Kim, and K. Pulli, "Short-Range FMCW Radar for Gesture Recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 1–8.

[7] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[8] R. Szeliski, *Computer Vision: Algorithms and Applications*, London, U.K.: Springer, 2010.

[9] A. Sharma, R. Patel, and K. Singh, "Hand Gesture Recognition System with Voice Feedback Using MediaPipe and OpenCV," *International Journal of Innovative Research in Technology (IJIRT)*, vol. 11, no. 1, pp. 1–6, 2025.