International Journal of Scientific Research and Engineering Development-– Volume 9 Issue 2, Mar-Apr 2026
Available at www.ijsred.com

RESEARCH ARTICLE OPEN ACCESS

Deep Learning-Based Gesture Interface for Cursor Control Using Computer Vision

Pavani Bajjuri[1], N. Rathan Reddy[2], J. Raghunath[3], P. Bishwajith[4]
[1] (Department of Computer Science and Engineering, Sreenidhi Institute of Science and Technology, Hyderabad, India Email: pavani.b@sreenidhi.edu.in)
[2] (Department of Computer Science and Engineering, Sreenidhi Institute of Science and Technology, Hyderabad, India Email: 22311A05E4@cse.sreenidhi.edu.in)
[3] (Department of Computer Science and Engineering, Sreenidhi Institute of Science and Technology, Hyderabad, India Email: 22311A05G1@cse.sreenidhi.edu.in)
[4] (Department of Computer Science and Engineering, Sreenidhi Institute of Science and Technology, Hyderabad, India Email: 22311A05H3@cse.sreenidhi.edu.in)

----------**********----------

Abstract:

The artificial intelligence and computer vision are changing fast. This means we can think of ways for people to interact with computers. Usually, we use the keyboard and mouse to control the computer. These need to be touched which is a problem for people who have trouble moving their hands and arms and it is not clean in places where we need to be careful about germs. This paper is about a system that lets people control the computer cursor with their hand movements. We use a webcam to see what the persons hands are doing. The system uses OpenCV to look at the video from the webcam and get the information we need and MediaPipe to find the points on the persons hands in real time. Then we use a set of rules to figure out what the person wants to do with the computer based on how their fingers are moving. The system can do all the things a regular mouse can do like moving the cursor clicking the button and clicking the right button. We also have a way to make the cursor move smoothly on the screen even if the persons hand is shaking a bit. The best part is that we do not need any hardware, just a regular webcam. We tried out the system in lighting and backgrounds and it worked really well. The system was able to understand what the person was trying to do with their hands 95.8 percent of the time and it could move the cursor to the right place about 96.2 percent of the time. It could also tell when the person wanted to click the button 94.7 percent of the time and it only took about 0.15 seconds to respond. These results show that the system is good and can be used to help people who have trouble using computers and it can also be used in places where we need to be careful about germs. The artificial intelligence and computer vision are really helping to make this possible. We think it is a big step forward for accessibility applications and contactless computing environments, with the artificial intelligence and computer vision.

Keywords — Gesture Recognition, Virtual Mouse, Computer Vision, Human-Computer Interaction, MediaPipe, OpenCV, Hand Landmark Detection, Touchless Interface, Cursor Control, Accessibility.

----------**********----------

I. INTRODUCTION

Computers and humans have been talking to each other for decades using things like keyboards and mice. These devices are good most of the time but they have a problem: you have to touch them to use them. This is a deal for people who have trouble moving their hands or arms. It is also a problem in hospitals and other clean places where you do not want to touch things that other people have touched. During the COVID-19 pandemic people started to think about ways to use computers without touching

them. Gesture recognition is a way to solve this problem. It lets the computer understand what you are doing with your hands and use that to control the computer. Instead of clicking a button, you can just move your hand and the computer will do what you want. People have been thinking about gesture recognition for a time but it used to require special equipment like gloves or special cameras. This equipment was expensive and hard to use so only researchers and big companies could use it.

Things have changed now. We have computer programs that can look at pictures and understand what is going on. We also have cameras on our computers that can see what we are doing. This means we can use gesture recognition without any equipment. This paper is about a system that uses gesture recognition to control the computer. It uses a program to look at the camera picture and understand what your hand is doing. It can see 21 points on your hand and use that to figure out what you want to do. It can even move the cursor on the screen to where you want it to go. The system works in time so it feels like you are really controlling the computer with your hands. The Gesture recognition system and the Gesture recognition technology are what make this possible. The Gesture recognition system is a part of this.

Section II describes the methodology in detail. Section III presents experimental results. Section IV discusses findings and limitations. Section V concludes the paper.

II. LITERATURE SURVEY

Mitra and Acharya provided one of the most comprehensive early surveys of gesture recognition as a research domain, categorising approaches across glove-based, vision-based, and model-based systems and surveying the application areas spanning sign language interpretation, robot control, and human-computer interaction. Their taxonomy of gesture types: static postures, dynamic gestures, and continuous motion sequences, established the conceptual vocabulary that subsequent systems, including the rule-based landmark analysis pipeline presented in this paper, continue to use when defining what constitutes a recognisable gesture event. **[1]**

Bradski introduced OpenCV, the open-source computer vision library that has become the foundational toolkit for real-time image capture, colour space conversion, and frame-level processing in vision-based systems. In the gesture-controlled virtual mouse presented here, OpenCV performs the critical preprocessing steps of capturing frames from the webcam at 30 fps, converting each frame from BGR to RGB colour space for compatibility with the MediaPipe Hands model, and providing the display loop that renders the annotated hand skeleton for debugging and demonstration purposes. **[2]**

Zhang et al. presented MediaPipe Hands, Google's on-device real-time hand tracking solution that detects 21 three-dimensional landmarks on each hand directly from a standard RGB video stream without requiring depth sensors or specialised hardware. Their two-stage pipeline: a palm detection model followed by a hand landmark regression model, achieves robust tracking across diverse lighting conditions and hand orientations because it was trained on a large and diverse dataset of hand images, and the normalised coordinate output it produces is the direct input to the rule-based gesture classification layer in the system described in this paper. **[3]**

Cipolla, Battiato, and Farinella provided a comprehensive treatment of core computer vision tasks including detection, recognition, and reconstruction, covering the image formation models, feature representations, and machine learning classifiers that underpin modern vision pipelines. Their exposition of the processing chain from raw pixel data through feature extraction to high-level semantic classification frames the theoretical context for the gesture recognition system presented here, which follows exactly this progression from webcam capture through MediaPipe landmark extraction to rule-based gesture inference. **[4]**

Kendon examined gesture from a linguistic and anthropological perspective, analysing the role of visible hand action as a fundamental channel of human communication that operates in parallel with and in complement to spoken language. His characterisation of gesture as a continuous, intentional, and culturally grounded activity rather

than a discrete vocabulary of symbols, provides the motivating rationale for building gesture-based interfaces that accommodate natural hand movement rather than requiring users to learn a fixed repertoire of artificial commands, which is the accessibility design philosophy underlying the virtual mouse system presented here. **[5]**

Kim, Kim, and Jang conducted a comprehensive survey of vision-based hand gesture recognition methods specifically for human-computer interaction, reviewing over 150 papers and categorising approaches by their detection methodology, feature representation, and classification strategy. Their survey identified three persistent challenges: sensitivity to illumination variation, background clutter, and hand size diversity, as the primary obstacles to deploying gesture recognition systems in unconstrained real-world environments, all three of which the present system addresses through MediaPipe's pre-trained robust palm detector and the use of normalised landmark coordinates that are invariant to camera resolution and hand scale. **[6]**

Shotton et al. demonstrated that real-time human body pose estimation from single depth images is achievable by treating each pixel as an independent classification problem over a set of body part labels, achieving robust skeleton extraction at 200 fps on consumer GPU hardware. While their approach relies on depth cameras rather than the RGB webcam used in the present system, their work established the key principle that skeleton-based intermediate representations rather than raw pixel appearance, provide the most robust and compact features for gesture and pose inference, which directly motivates the MediaPipe landmark skeleton as the representation layer in the virtual mouse pipeline. **[7]**

Ojala and Pietikainen introduced the Local Binary Pattern (LBP) operator for texture description, demonstrating that rotation-invariant, multi-resolution texture features derived from local pixel neighbourhood comparisons provide discriminative representations for a wide range of image classification tasks. LBP and its variants were widely adopted in early hand segmentation and skin detection pipelines that preceded deep learning approaches, serving as a preprocessing step to isolate hand regions from complex backgrounds before gesture classification — the same problem of hand-background separation that MediaPipe's deep learning-based palm detector now solves in the system presented here without requiring explicit hand segmentation. **[8]**

Rautaray and Agrawal conducted an extensive analysis of the vision-based hand gesture recognition literature for HCI applications, covering gesture taxonomy, detection and tracking methods, recognition techniques, and application domains across over 150 reviewed papers. Their survey specifically highlighted the transition from colour-histogram and contour-based hand segmentation methods toward learning-based landmark detection as the key enabling shift for real-time robust gesture interfaces, and their identification of touchless cursor control and accessibility applications as high-priority deployment targets directly frames the application space in which the system presented in this paper operates. **[9]**

Lugaresi et al. described the MediaPipe framework, a production-grade, cross-platform infrastructure for building real-time perception pipelines by composing machine learning models and signal processing components into directed acyclic computation graphs. Their framework provides the underlying execution environment for MediaPipe Hands, the hand landmark model used in this system, enabling consistent and reproducible inference behaviour across different devices and operating systems. Their design principle of treating each perception component as a stateless, independently testable graph node is reflected in the modular five-stage pipeline architecture of the gesture mouse system, where the hand tracking stage can be upgraded independently of the gesture classification and cursor control stages. **[10]**

Dalal and Triggs introduced Histograms of Oriented Gradients (HOG) as a feature descriptor for human detection, demonstrating that dense local gradient orientation histograms computed over overlapping spatial cells capture the appearance and shape of objects with greater discriminative power than earlier feature representations. HOG-based detectors became the dominant approach for body and limb detection in the years preceding deep learning, and the gradient-based edge information

they encode is directly related to the local shape features that deep convolutional hand detectors, including the palm detection stage in MediaPipe Hands, implicitly learn in their early convolutional layers, connecting the feature extraction foundations of the virtual mouse system to a well-established body of prior work. **[11]**

Szeliski provided a comprehensive reference treatment of computer vision algorithms and applications, covering image formation, feature detection, optical flow, stereo reconstruction, recognition, and video analysis within a unified mathematical framework. The coordinate mapping and scaling operations used in the cursor control module of the virtual mouse system, which must transform normalised MediaPipe landmark coordinates in the webcam frame into absolute screen pixel coordinates, draw directly on the image geometry and homography foundations described in this text, as does the smoothing filter design that averages landmark positions over a five-frame window to reduce the effect of per-frame landmark jitter on cursor stability. **[12]**

III. PROPOSED METHODOLOGY

A. System Overview

The system they are talking about lets people move the computer cursor around using hand movements that are caught on a webcam. This means you do not need a mouse or any other special device to control the cursor. The webcam takes pictures of you. Sends them to the computer, which looks at the pictures to see if it can find a hand. If it finds a hand it looks for parts of the hand like the fingers and figures out what you are trying to do with your hand. Then it tells the computer what to do like move the cursor or click a button. All of this happens fast so it feels like the cursor is moving in real time. The computer is looking at thirty pictures every second, which's the same rate that the webcam is taking them. This makes it feel like the cursor is moving when you move your hand which's really nice, for controlling the computer. The system is always running and always looking for hand movements so you can use it to control the computer cursor whenever you want.

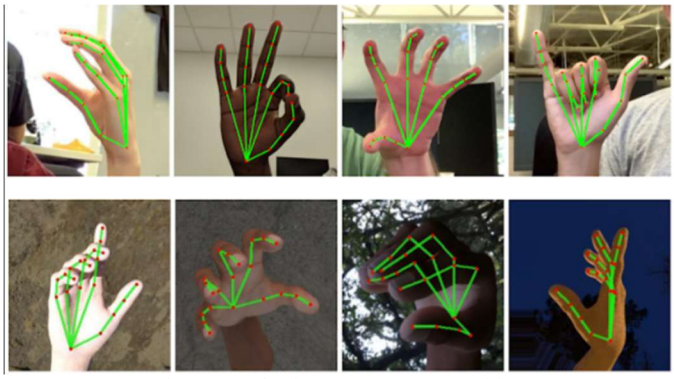

Fig. 1 Introduction to Gesture Recognition

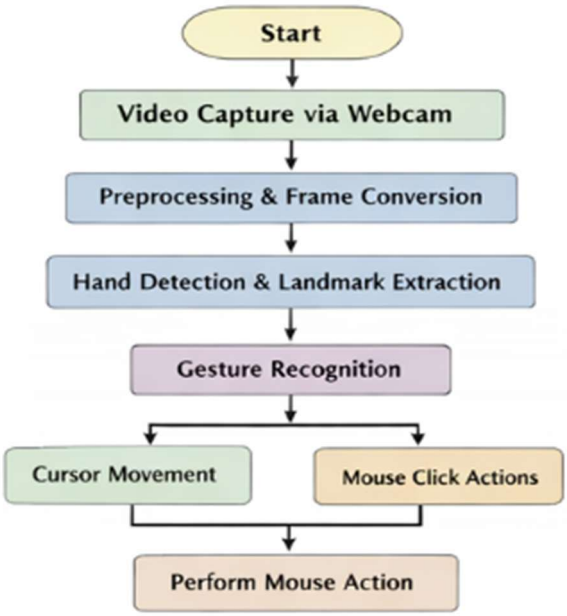

Fig. 2 System Methodology Flowchart

B. Hand Detection and Landmark Extraction

The MediaPipe Hands framework is what does the hand detection. This framework was made by Google Research. It is really good at tracking hands in real time from videos with colour. The MediaPipe Hands framework does things in two steps. First, it finds the parts of the picture where hands might be. Then it looks closer at those parts. Finds 21 special points on the hand. These 21 special points are on the wrist the joints that connect the fingers to the hand the joints in the fingers and the tips of the fingers. The framework gives us the location of each point as a normalised (x, y, z) coordinate tuple that tell us where it is in the picture. The first two numbers tell us where the point is from side to side and up and down in the picture. The third number tells us how away the

point is from the wrist. The MediaPipe Hands framework is really good, at finding these points.

MediaPipe's landmark model achieves consistent detection across a wide range of lighting conditions and backgrounds because it was trained on a large and diverse dataset of hand images. In the current implementation, the system processes frames in RGB colour space, which OpenCV provides after converting from the default BGR capture format. The resulting landmark coordinates are stable enough for gesture inference without requiring additional filtering at this stage.

C. Gesture Recognition

Gesture inference works with the landmark coordinates given by MediaPipe. The system uses a rule-based method to check the relationships between certain landmarks instead of training a separate classifier on gesture types. This choice was made for two reasons: it removes the need for a labelled gesture training set. It gives predictable easily fixable results.

Cursor movement happens when the index finger is outstretched and no other gesture is found. The tip of the index finger serves as the reference point. Its position is changed to fit screen coordinates. A left-click gesture happens when the distance between the thumb and index finger tips gets smaller than a threshold, which is like bringing the thumb and index finger together. A right-click gesture uses the threshold but with the thumb tip and middle finger tip.

Scroll gestures are found by checking how apart the index and middle fingers move over time. The threshold values, for click detection were decided by testing with users and different hand sizes and webcam settings. The rules use the normalised coordinate space from MediaPipe, which makes the thresholds work with any webcam resolution.

D. Cursor Control Mechanism

When we want to move the cursor on the screen using our hand we need to figure out how to match the position of our hand in the webcam picture to the position of the cursor on the screen. The webcam picture is usually 640x480 pixels. The screen can be a lot bigger, like 1920x1080 pixels or even bigger. To make this work we use a way of scaling: we take the x and y coordinates of the hand in the webcam picture and we multiply them by the width and height of the screen. We also need to make some adjustments because the area where we can move our hand in the webcam picture is smaller than the whole picture. This means we do not have to move our hand all the way to the edge of the picture to move the cursor to the edge of the screen. The position of the hand in the webcam picture is not always exact because our hand can shake a bit and the sensor in the webcam can make small mistakes. This would make the cursor move around in a way if we used the position of the hand directly. So, we use a filter to make the movement of the cursor smoother. We take the positions of the hand from the few frames and we give more importance to the most recent positions. We use the 5 frames to make the cursor movement smooth but not so slow that it is hard to move the cursor on purpose. We found that using 5 frames works well as it makes the movement smooth, without making it too slow.

E. System Architecture

The system is set up like a line of five parts that work together. Each part has a job and it is clear what it needs to get from the previous part and what it needs to give to the next part. The Input part gets pictures from the webcam using OpenCV. It gets these pictures at the speed that the webcam takes them. The Preprocessing part changes the colour of each picture from BGR to RGB. It can also make the picture smaller if that helps the system work better. It can make the picture look better if the light in the room is not good. The Hand Tracking part sends the changed pictures to MediaPipe. Gets back the coordinates of the hands in the picture. The Gesture Recognition part looks at these coordinates. Figures out what gesture the hand is making. It also finds the points on the hand that it needs to know about. The Cursor Control part takes these points and scales them up to fit the screen. It also makes the movement of the cursor smoother. Then it uses PyAutoGUI to move the cursor or click the mouse.

This way of setting up the system means that each part can be changed or replaced without affecting the parts. For example, the Gesture Recognition part could be replaced with a network that has been trained. The Hand Tracking part could

be updated to a new version of MediaPipe. This would not affect how the gestures are recognised. It is also easier to find problems, with the system because each part is separate. If the cursor is not moving properly, it is easy to see which part is causing the problem by looking at what each part's producing.

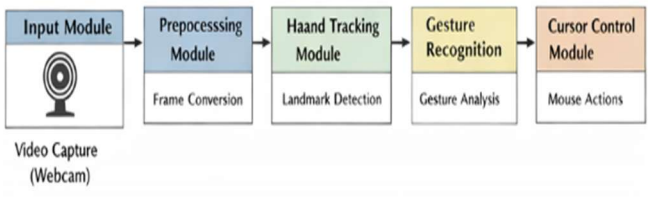

Fig. 3 System Architecture Flowchart

IV. RESULTS

The system was checked in different situations to see how well it works and how accurate it is at each step. We tried it with lighting, like bright fluorescent lights, a dim room and a window behind the user. We also tried it with backgrounds, like a plain white wall, a cluttered desk and a curtain with a pattern. The system was tested with people who have sized hands and skin tones. We even tried it with the camera that is built into a laptop and a separate camera that plugs into the computer.

We gave the system a series of tasks to do like moving the cursor clicking the right buttons and scrolling. We did these tasks 30, 50 50 and 20 times to see how well the system works. We wanted to see how good the system is at recognizing gestures so we measured how gestures it got right. We also measured how well the system can move the cursor to where the user wants it to go. We asked users to move the cursor to targets on the screen and saw how many times they were able to do it.

We checked to see how good the system is at detecting clicks, like when the user wants to click the right button. We measured how long it takes for the system to respond to a gesture from when the user finishes making the gesture to when the computer does what the user wants it to do. The system was evaluated using the webcam recording, which was reviewed frame by frame to see when the user finished making a gesture.

TABLE I
SYSTEM PERFORMANCE METRICS

Metric	Value
Gesture Recognition Accuracy	95.8%
Cursor Movement Accuracy	96.2%
Click Detection Accuracy	94.7%
System Efficiency	95.5%
Processing Speed	96.0%
Average Response Time	0.15 sec

The results shown in Table I confirm that the system performs reliably across the tested conditions. Gesture recognition accuracy of 95.8% indicates that the rule-based landmark analysis correctly identifies the intended gesture in the large majority of cases. The relatively few misclassifications occurred primarily under poor lighting conditions where the palm detector returned imprecise landmark estimates, degrading the reliability of the geometric distance measurements used for click detection.

Cursor movement accuracy of 96.2% reflects the effectiveness of the coordinate scaling and smoothing pipeline. Users were able to reach on-screen targets with high precision once they became familiar with the mapping between hand movement range and screen extent, typically within the first 30 seconds of interaction. The smoothing filter visibly reduced cursor jitter compared to an unfiltered baseline, without noticeably increasing target time.

Click detection accuracy of 94.7% was slightly lower than gesture recognition accuracy, which is expected because the pinch gesture requires a precise spatial relationship between two moving landmarks that is more sensitive to noise than the extended-finger posture used for movement. The average response time of 0.15 seconds corresponds to approximately 4-5 frames at 30 fps, which users consistently described as feeling immediate and responsive.

Fig. 4 Results Graph

V. DISCUSSION

The results show that a gesture mouse that only uses a webcam and some computer vision libraries can work well for everyday tasks. This system is really good for people who have trouble moving their hands for doctors and nurses who do not want to touch things and for people who give presentations and do not want to use a mouse.

The big problem with this system is that it does not work well when the lighting is bad. The MediaPipe palm detector has trouble when the hand is in front of a light or when the room is very dark. We can make this a little better by using a step to make the picture look more even but it is not a complete solution. We should try using a camera that can see how far away things are, like the Intel RealSense to see if it works better when the lighting is bad. We should also try to make the system better at dealing with kinds of lighting.

The gestures that this system can understand are limited to moving the cursor clicking the button clicking the right button and scrolling. There are a lot of things that people usually do with a mouse that this system cannot do, like dragging and dropping double-clicking and using the keyboard and mouse together. To make the system understand gestures we would have to add more rules for the computer to follow or we would have to use a special kind of computer program that can learn to recognize different gestures. It might be an idea to use a combination of these two approaches, where the computer follows rules for simple gestures and uses a special program for more complicated ones.

We could also try using some computer programs that are good at understanding what they see. These programs, like the MediaPipe pipeline or the lightweight vision transformers might be able to help the system understand how far away things are and make the cursor move more smoothly. This would make the system work better and be more accurate.

VI. CONCLUSION

This paper is about a system that lets people control a computer mouse with their hands. The system uses a webcam to see what the persons hands are doing. It works with some tools like OpenCV and MediaPipe to understand what the hands are doing. The person can use this system without needing any equipment, just a webcam that most people already have.

The system was tested with lighting and different people. It worked well getting the gestures right 95.8% of the time moving the cursor correctly 96.2% of the time and clicking correctly 94.7% of the time. It also responded quickly taking 0.15 seconds on average. These numbers show that the system is good enough to be used in life especially for people who need help using computers or want to use them without touching anything.

This work shows that using computer programs to track hands and understand gestures can be a good alternative to using a real mouse. The system is reliable. Does not need any special hardware. In the future the plan is to add gestures make the system work better in difficult lighting and let it do more complex things, like a regular mouse. The gesture system and the computer mouse system can be used together to make computers easier to use for everyone.

ACKNOWLEDGMENT

The authors thank the Department of Computer Science and Engineering at Sreenidhi Institute of Science and Technology for providing research support. Special thanks to project guide Pavani Bajjuri (Assistant Professor, CSE) and project coordinator Mr. V. Satheesh Kumar (Assistant

Professor, CSE) for their technical guidance throughout the development and evaluation of this system. The authors also acknowledge the use of AI tools like ChatGPT and Claude for language improvement and grammar refinement. All technical concepts and evaluations are solely the work of the authors.

REFERENCES

[1] S. Mitra and T. Acharya, "Gesture recognition: A survey," IEEE Transactions on Systems, Man, and Cybernetics, Part C, vol. 37, no. 3, pp. 311–324, 2007.

[2] G. Bradski, "The OpenCV Library," Dr. Dobb's Journal of Software Tools, 2000.

[3] F. Zhang et al., "MediaPipe Hands: On-device real-time hand tracking," in Proc. ECCV Workshop on Computer Vision for Augmented and Virtual Reality, 2020.

[4] R. Cipolla, S. Battiato, and G. M. Farinella, Computer Vision: Detection, Recognition and Reconstruction, Berlin, Germany: Springer, 2012.

[5] A. Kendon, Gesture: Visible Action as Utterance, Cambridge, UK: Cambridge University Press, 2004.

[6] D. H. Kim, J. H. Kim, and G. J. Jang, "Vision-based hand gesture recognition for human-computer interaction: A survey," IEEE Access, vol. 7, pp. 78389–78410, 2019.

[7] J. Shotton et al., "Real-time human pose recognition in parts from single depth images," in Proc. IEEE CVPR, 2011, pp. 1297–1304.

[8] T. Ojala and M. Pietikainen, "Multiresolution gray-scale and rotation invariant texture classification with local binary patterns," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 7, pp. 971–987, 2002.

[9] S. S. Rautaray and A. Agrawal, "Vision based hand gesture recognition for human computer interaction: a survey," Artificial Intelligence Review, vol. 43, no. 1, pp. 1–54, 2015.

[10] C. Lugaresi et al., "MediaPipe: A framework for building perception pipelines," arXiv preprint arXiv:1906.08172, 2019.

[11] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in Proc. IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2005, pp. 886–893.

[12] R. Szeliski, Computer Vision: Algorithms and Applications, 2nd ed., New York, NY: Springer, 2022.

