International Journal of Artificial Intelligence Research and Development (IJAIRD)
Volume 2, Issue 1, January-June 2024, pp. 158-175. Article ID: IJAIRD_02_01_014
Available online at https://iaeme.com/Home/issue/IJAIRD?Volume=2&Issue=1
Impact Factor (2024): 6.98 (Based on Google Scholar Citation)
Journal ID: 234A-56Z1

WINDOWS CONTROLLING USING HAND GESTURE

Pratik Pramod Alkutkar
Electronics and Computer Engineering, P.E.S's Modern College of Engineering Pune, India

Anurag Chandan Angal
Electronics and Computer Engineering, P.E.S's Modern College of Engineering Pune, India

Ameya Ajit Kabir
Electronics and Computer Engineering, P.E.S's Modern College of Engineering Pune, India

Prof. Ashwini A. Kokate
Electronics and Computer Engineering, P. E. S's Modern College of Engineering, Pune, India

ABSTRACT

This document gives information about the implementation of the project Windows Controlling using Hand Gestures. The proposed system uses Computer Vision techniques to recognize hand gestures captured by a webcam in real time. The system's architecture comprises three main stages: hand detection, gesture recognition, and Windows application control. The proposed system makes use of Computer Vision techniques to recognize and understand hand gestures captured by a webcam in real time. The system's architecture comprises three main stages: hand detection, gesture recognition, and Windows application control.

Keywords: Hand Gesture Recognition, Computer Vision, Real-Time Processing, Webcam Input, Windows Application Control

Cite this Article: Pratik Pramod Alkutkar, Anurag Chandan Angal, Ameya Ajit Kabir and Prof. Ashwini A. Kokate, Windows Controlling Using Hand Gesture, International Journal of Artificial Intelligence Research and Development (IJAIRD), 2(1), 2024, pp. 158-175.
https://iaeme.com/Home/issue/IJAIRD?Volume=2&Issue=1

I. INTRODUCTION

This project introduces an AI-driven virtual mouse system utilizing computer vision for hand gesture recognition and fingertip detection, enabling intuitive computer interaction. In a landscape where compactness and wireless technologies are prevalent, this system offers a streamlined approach to computing. Gesture recognition systems have emerged as a prominent technology, superseding conventional mechanical communication methods. This paper delineates the segmentation of the domain market based on various factors such as technology, type, application, product, usage, and geographic location. The proliferation of gesture recognition systems spans diverse applications including virtual controllers, virtual mice, smart TVs, immersive gaming technologies, assistive robotics, and sign language recognition. Notably, while a plethora of solutions exist, only a minority directly leverage webcams for gesture recognition; the majority rely on Arduino and sensor-based approaches. Nevertheless, challenges persist, particularly in scenarios where the background environment contains components resembling human skin, potentially leading to misidentifications of motions. Additionally, ensuring the hand remains within the permitted range poses a significant constraint. This paper offers a comprehensive overview of gesture recognition systems, highlighting advancements, applications, and pertinent challenges, thereby providing insights for future research and development endeavors in this domain.

The usage of a hardware computer mouse in conjunction with manual mouse inputs and mouse positioning is projected to be replaced by the Virtual Mouse program. With the use of gestures, every job can be completed with this program, making computer use easier. Furthermore, by simply displaying the right combination of colors to the webcam, the Virtual Mouse program enables persons with motor impairments to interact with the computer.

OBJECTIVE OF PROJECT

The main goal of this project, is to provide an alternative to a routine physical mouse so that there will be less physical contact with the mouse. We can perform all the mouse operations and a few keyboard operations by just recognizing different hand- gestures through web-cam.

Outline of Project

The outline of this project is as follows, In today’s world, there is

In today's rapidly evolving technological landscape, Artificial Intelligence (AI) plays a pivotal role in driving innovation. This project focuses on a specific aspect of AI, namely finger movement gesture detection, to enhance computer interaction. By utilizing a camera, users can control their computer's interface solely through finger movements, eliminating the need for physical input devices like a mouse. This approach not only simplifies user interaction but also promotes accessibility through intuitive finger detection methods. AI virtual mouse and keyboard are developed using Python and OpenCV, a computer vision library. The proposed model makes use of, the MediaPipe which is the package for recognizing the hands and the tip of the fingers, as well as PyAutoGUI and Autopy packages for controlling the system by performing mouse operations like left click, right click, scroll up/ down, and keyboard operations like escape, volume up, volume down. The outcome of this model demonstrates a high-level accuracy that can function extremely well in real-time applications using only a CPU and no GPU. This system also helps in controlling robots. In the virtual keyboard, the movement of finger tapping will be captured on the virtual keyboard which will be displayed on the screen while in the mouse finger movement will be captured with the assistance of the camera.

Significance of Problem

In the field of human-computer interaction technology, gesture recognition is a widely used and sought-after analytical technique. It can be used in a variety of fields, such as sign language translation, medical applications, music production, robot control, virtual environment administration, and home automation.

Recent years have seen a particular emphasis on HCI research. Because of its skill, the hand is the most useful tool for communication in a variety of body regions. Just a few of the situations when human motion—especially that of the hands, arms, and face—is described with the term "gesture" is instructive.II. Theoretical Aspects

Human-Computer Interaction Technology

Human Computer Interaction is a multidisciplinary field concerned with designing computer technology to facilitate interaction between Computers and humans (Users). Initially focused solely on computers, HCI has evolved to encompass the design of various forms of information technology. It has garnered significant attention within academic circles, with scholars and practitioners recognizing its importance in establishing computer-user interaction akin to human-to-human dialogue.

Iterative design stands as a cornerstone principle within HCI. Following an initial comprehension of the audience targeted, their tasks, and measurements related to interaction, designers undertake several iterative steps: designing the user interface, conducting user testing, analyzing test results, and iterating the design process. This iterative cycle continues until a user friendly interface is achieved.

The interaction between humans and machines can be facilitated through various means, utilizing different human senses to create user interfaces (UI). These include tactile UI (touch), visual UI (sight), and auditory UI (sound). HCI (Human-Computer Interaction) practitioners play a crucial role in determining the optimal combination of these interfaces based on the product's purpose and user needs. For instance, a mobile app might benefit from a combination of visual and auditory UI elements to enhance user experience and accessibility. Mouse and keyboard are one of the brilliant developments of HCI.

Fig 2.1: HCI & Related Research Fields

For instance, interactive input devices encompass speech recognition, keyboards, and touch-sensitive screens, while sensory perception extends to output devices like printers and visual displays. Wireless internet applications and virtual reality devices also contribute to this spectrum of human-machine interaction.

Image Processing

Image processing refers to the digital manipulation of images to achieve various objectives such as enhancement or extraction of useful information. It falls under the broader umbrella of signal processing, where the input typically consists of photographs or video frames, and the output may be an image or a set of parameters related to the image's characteristics.

Within image processing, images are often segmented into regions of interest, reflecting the presence of distinct objects or areas within the image.

In the realm of image science, image processing encompasses techniques applied to input images, resulting in outputs that could be images themselves or parameters associated with the input image. While digital image processing is the most common form, optical and analog methods are also feasible. The process of obtaining input images is termed imaging.

Image processing treats images as two-dimensional signals, allowing for the application of standard signal processing techniques. Each image can be analyzed for sub-regions, reflecting the presence of distinct objects or features. Consequently, image processing finds application in various domains, including defect identification on surfaces such as ceramic tiles, where the defective areas constitute regions of interest.

Image Processing stands as a swiftly evolving technology, finding applications across diverse sectors of business. It constitutes a significant research domain in the Engineering & Computer Science disciplines.

Essence of the Image Processing encompasses three primary stages:

1. Importing: This initial step involves acquiring the image data, typically achieved through optical scanners or digital photography.
2. Analysis and Manipulation: Following importation, the image undergoes analysis and manipulation procedures, which may include tasks such as data compression and enhancement techniques. These processes aim to improve the quality of the image or extract valuable information from it.
3. Output: The final stage involves generating the output, which could be an altered image reflecting the effects of the analysis and manipulation performed, or a comprehensive report based on the findings of image analysis.

This systematic approach to Image Processing underscores its pivotal role in various fields, offering solutions to enhance image quality, extract useful insights, and derive actionable information from visual data.

Image processing encompasses 2 main types:

1. Analog Image Processing:

Analog image processing involves manipulating two dimensional analog signals through analog means, rather than digitally. It is often utilized for processing hard copies such as printouts and photographs. Image analysts rely on visual techniques and fundamental interpretations to analyze images. This approach extends beyond the specific area under study, leveraging the analyst's knowledge and association capabilities

2. Digital Image Processing:

Digital image processing involves manipulating digital images using computers. The raw data which is obtained from image sensors, such as those on platforms like satellites, often contain imperfections. Digital processing techniques are applied to overcome these deficiencies and extract accurate information.

The digital processing workflow typically comprises three phases:

i. Pre-processing: It Involves preparing the raw data for subsequent processing stages by noise removal, correction of distortions, and standardizing the format.
ii. Enhancement and Display: Focuses on improving visual quality of the image for human perception, enhancing specific features, and adjusting contrast and brightness.
iii. Information Extraction: This involves extracting relevant information or features from the processed image for analysis or further utilization.

In our case, digital computers are employed to execute image-processing tasks. The process involves converting the image into a digital format using a scanner or digitizer, followed by subsequent manipulation. Image processing is defined as subjecting numerical representations of objects to a sequence of operations to achieve a desired outcome. It entails taking an initial image and generating a modified version thereof, effectively transforming one image into another.

The term "image processing" commonly refers to the manipulation of two-dimensional images by a digital computer. However, in a broader sense, it encompasses the digital processing of two-dimensional data. A digital image is essentially an array of real numbers represented by a finite number of bits. One notable advantage of digital image processing lies in its versatility, repeatability, and ability to preserve the precision of the original data throughout the processing pipeline.

Machine Learning

Machine learning (ML) is a branch of artificial intelligence (AI) that enables software applications to enhance their accuracy in predicting outcomes without explicit programming. ML algorithms leverage historical data to forecast new output values. It encompasses the study of computer algorithms capable of autonomous improvement through experience and data utilization, representing a crucial aspect of AI.

ML algorithms construct models based on sample data, known as training data, facilitating predictions or decisions without explicit programming. This technology finds widespread application across various domains, including medicine, email filtering, speech recognition, and computer vision, addressing tasks where conventional algorithms may be impractical.

A subset of ML closely intersects with computational statistics, focusing on prediction using computers. However, not all ML falls within the realm of statistical learning. Mathematical optimization contributes methods, theories, and application domains to the ML field. Data mining, an associated discipline, emphasizes exploratory data analysis through unsupervised learning.

Certain ML implementations employ data and neural networks to emulate biological brain functionality. In business contexts, ML is often referred to as predictive analytics, aiding in decision-making processes. This interdisciplinary field continues to advance, driving innovation and automation across various industries.

Machine learning has given computer systems the ability to automatically learn without being explicitly programmed. So, it can be described using the life cycle of machine learning. The machine learning life cycle is a cyclic process to build an efficient machine learning project. The main purpose of the life cycle is to find a solution to the problem or project.

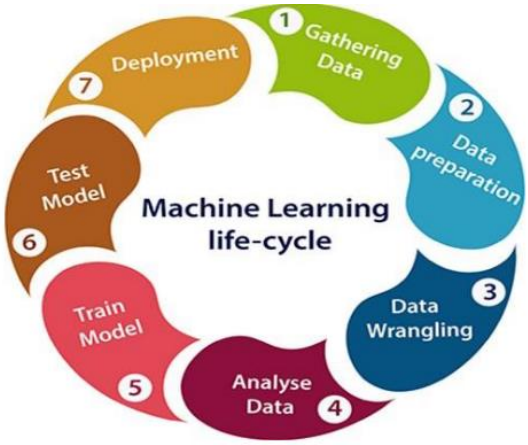


Fig.2.2: Machine Learning Lifecycle

The machine learning life cycle involves seven major steps, which are given below:

1. Data Preparation
2. Data Wrangling
3. Analyze Data
4. Train the model
5. Test the model
6. Deployment

Computer Vision

Computer vision, a captivating form of AI, is likely encountered in various instances without awareness. It mimics aspects of human vision, enabling computers to recognize and analyze objects in images and videos. Recent advancements in artificial intelligence, particularly in deep learning and neural networks, have propelled significant progress in this field, even outperforming humans in certain object detection and labeling tasks. The proliferation of data plays a pivotal role in enhancing computer vision capabilities through training and refinement processes.

III. SYSTEM DESIGN

The primary goal of proposed system is to facilitate computer mouse cursor functionalities utilizing either an integrated camera within the computer, thereby eliminating the need for a conventional mouse device. Human-Computer Interaction (HCI) is achieved through hand gesture and hand tip detection employing computer vision techniques. The system employs an AI virtual mouse system to track fingertip movements captured by the built-in or external camera, enabling cursor operations and scrolling functions without the use of additional hardware such as a wireless or Bluetooth mouse.

In the proposed system, the webcam captures frames, which are subsequently processed to identify various hand gestures and hand tip movements, leading to the execution of corresponding mouse functions. Development of the AI virtual mouse system is conducted using the Python programming language, leveraging the OpenCV library for computer vision tasks.

The system utilizes the MediaPipe package for hand tracking and tracking of hand tip movements. Additionally, the Autopy and PyAutoGUI packages are employed to facilitate

window screen navigation, including left-click, right-click, and scrolling functionalities. Experimental results demonstrate the proposed model's high accuracy and suitability for real-world applications, even when utilizing a CPU without GPU acceleration.

MediaPipe

Recognizing the shape and movements of hands holds immense potential for enhancing user experiences across a spectrum of technological domains and platforms. These applications range from facilitating sign language understanding and hand gesture control to enabling augmented reality experiences by overlaying digital content onto the physical world. Despite being a natural ability for humans, achieving robust real-time hand perception presents formidable challenges in computer vision due to factors such as self-occlusions and the lack of distinct patterns in hand features.

MediaPipe Hand introduces an advanced solution for hand and finger tracking, leveraging machine learning (ML) methodologies to infer the precise 3D coordinates of 21 key landmarks within a hand from a single input frame. While existing state-of-the-art approaches predominantly depend on powerful desktop environments for inference, our method demonstrates real-time performance even on resource-constrained mobile devices. Furthermore, our system seamlessly scales to accommodate the tracking of multiple hands concurrently.

By democratizing access to this sophisticated hand perception functionality within the wider research and development community, we anticipate the emergence of a multitude of innovative applications and novel research avenues. This accessibility is poised to catalyze exploration into creative use cases, thereby fostering advancements in both practical applications and theoretical understanding within the field of computer vision.

MediaPipe hands utilize an ML pipeline consisting of multiple models working together. The proposed system comprises two main components: a palm detection model, which operates on the entire image to identify and delineate the orientation of a hand bounding box, and a hand landmark model, which operates on the cropped image region defined by the palm detector to extract precise 3D hand key points. This approach mirrors the methodology utilized in our MediaPipe Face Mesh solution, which combines a face detector with a face landmark model.

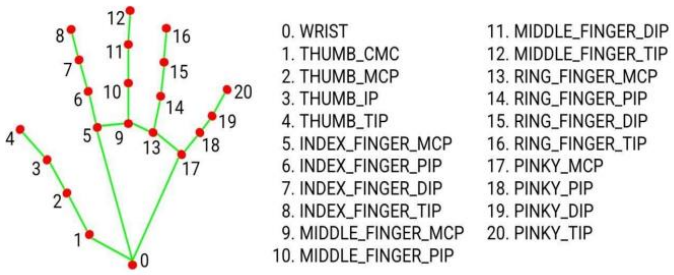


Fig 3.1: Landmarks of Hand

By providing the hand landmark model with accurately cropped hand images, the reliance on data augmentation techniques such as rotations, translations, and scaling is significantly reduced.

The system is implemented as a MediaPipe graph, incorporating a hand landmark tracking subgraph sourced from the hand landmark module, and rendering functionalities facilitated by a dedicated hand renderer subgraph.

Internally, the hand landmark tracking subgraph utilizes a hand landmarks graph from the same module, along with a palm detection subgraph sourced from the palm detection model.

This modular design enhances system flexibility and facilitates seamless integration of additional functionalities within the pipeline.

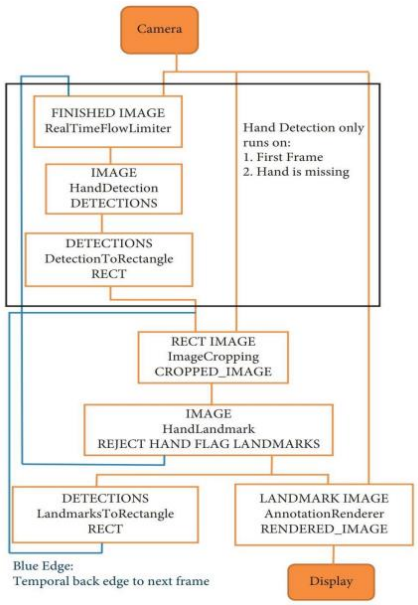


Fig 3.2: Flowchart of MediaPipe

Open CV

OpenCV stands as a robust open-source library pivotal in computer vision, machine learning, and image processing domains, significantly contributing to real-time operations crucial in contemporary systems. Leveraging OpenCV, users can effectively process images and videos to detect various entities such as objects, faces, or human handwriting. Integration with libraries like NumPy enhances Python's capability to analyze OpenCV array structures, facilitating the identification of image patterns and their distinct features through vector space representation and mathematical operations.

Since its inception with the first version, OpenCV 1.0, the library has been released under a BSD license, rendering it freely available for both academic and commercial utilization. Supporting interfaces in C++, C, Python, and Java, and compatibility with Windows, Linux, Mac OS, iOS, and Android platforms, OpenCV offers versatility in application development. Notably, OpenCV's design prioritizes real-time applications, emphasizing computational efficiency. To achieve this, the library is predominantly implemented in optimized C/C++, harnessing the capabilities of multi-core processing architectures.

Python

Python has emerged as a cornerstone in the realm of programming languages, renowned for its versatility, readability, and extensive ecosystem of libraries and frameworks. Its simplicity and expressive syntax make it accessible to both novice and experienced programmers alike, fostering rapid development and prototyping across various domains. Python's ubiquity spans diverse fields including web development, data science, machine learning, artificial intelligence, scientific computing, and automation. Leveraging libraries such as NumPy, Pandas, TensorFlow, and sci-kit-learn, Python facilitates complex data manipulation, statistical analysis, and machine learning tasks with ease. Furthermore, frameworks like Django and Flask empower developers to build scalable web applications efficiently.

The robustness of Python is augmented by its vibrant community, which actively contributes to its development, documentation, and support. As Python continues to evolve and adapt to

emerging technologies, it remains a pivotal tool for tackling contemporary challenges and driving innovation across industries.

Autopy

Autopy is a Python library designed to automate GUI interactions and simulate human input such as keyboard presses, mouse movements, and clicks. It provides a simple and intuitive interface for performing tasks like window management, screen capturing, and pixel-level manipulation. Autopy is particularly useful for automating repetitive tasks, testing graphical applications, and creating interactive simulations. With its cross-platform compatibility and ease of use, Autopy empowers developers to streamline workflows and enhance productivity in a variety of software development and testing scenarios.

PyAutoGUI

PyAutoGUI is a popular Python library that enables the automation of GUI interactions on desktop platforms. It provides functionalities to control the keyboard and mouse, capture screenshots, and perform GUI operations such as clicking, dragging, and typing. PyAutoGUI is platform-independent, making it suitable for automating tasks across various operating systems including Windows, macOS, and Linux. With its intuitive API and extensive documentation, PyAutoGUI simplifies the process of automating repetitive tasks, testing GUI applications, and creating user-friendly automation scripts. Additionally, PyAutoGUI can be integrated with other Python libraries and frameworks to build complex automation workflows and enhance efficiency in software development and testing processes.

IV. LITERATURE REVIEW

Several research studies have been conducted in the field of deep learning, with a specific focus on Convolutional Neural Networks (CNNs) and their applications in hand gesture recognition. One of these studies explores how CNNs have significantly improved the precision and speed of systems designed for recognizing hand gestures. This research highlights the remarkable advancements in this technology and its potential to revolutionize gesture recognition.

Another research effort, involving multiple authors, delves into the comprehensive understanding of CNN architectures and their broad range of applications. It traces the evolutionary progression of CNNs and underscores their significance in tasks related to image and pattern recognition. This study sheds light on the foundational aspects of CNNs, offering insights into their adaptability and versatility in various domains.

In a different study, the emphasis shifts to the real-time capabilities of CNNs in the context of computer vision applications, particularly in the domain of hand gesture recognition. The critical role of CNNs in achieving low-latency recognition is highlighted, underscoring its significance for real-time systems dedicated to recognizing hand gestures. This research acknowledges the practical implications of low-latency recognition, which is vital for applications requiring quick and responsive interaction.

The introduction of the Windows Gesture Recognition Framework by Microsoft is a noteworthy development in the realm of gesture recognition. This framework offers software developers the means to seamlessly integrate gesture-based control into applications designed for the Windows operating system. It provides valuable insights into Microsoft's approach to gesture recognition, indicating a growing interest in making gesture-based interaction more accessible and widespread.

Another study explores the various interfaces within the Windows environment that can be controlled through hand gestures. It scrutinizes different methods and technologies employed for implementing gesture-based control and provides a comparative analysis of their usability

and accuracy. This research aids in understanding the diverse possibilities and challenges involved in integrating gesture recognition into mainstream computing systems.

An additional research project focuses on the integration of CNNs with real-time systems for recognizing hand gestures. It discusses the technical challenges encountered in developing systems capable of understanding and responding to dynamic hand movements. This integration of deep learning with real-time applications represents a significant step forward in the field of human-computer interaction.

Furthermore, a study presents a practical application of CNN-based hand gesture recognition for controlling the Windows operating system. It offers insights into the implementation of a real-time system that responds to hand gestures to perform a variety of tasks within the Windows environment. This research demonstrates the tangible benefits of incorporating CNNs into everyday computing tasks.

Lastly, another work outlines the challenges faced in achieving real-time gesture recognition and proposes potential solutions. It particularly emphasizes the need for enhanced accuracy and user-friendliness in these systems, recognizing the importance of making gesture-based control both reliable and user-accessible.

In summary, these research endeavors collectively contribute to the evolving landscape of hand gesture recognition, where CNNs and real-time capabilities are at the forefront of technological advancements, offering a promising outlook for the integration of gestures into various applications and interfaces, especially within the Windows environment.

V. METHODOLOGY

Various functions and conditions we used in the system are explained in the flowchart of the system shown in Fig 3.3.

1. The AI virtual mouse system relies on webcam frames captured by a laptop or PC. Utilizing the Python computer vision library OpenCV, a video capture object is initialized to begin video capture from the webcam. These captured frames are then passed to the AI virtual system for processing and interaction.
2. In the AI virtual mouse system, the webcam continuously captures video frames throughout the program's runtime. These frames undergo processing, transitioning from BGR to RGB color space, facilitating the identification of hands within each frame of the video, done frame by frame.
3. In the AI virtual mouse system, a transformational algorithm is employed to convert fingertip coordinates from the webcam screen to the full computer window screen, enabling mouse control. Upon hand detection and identification of the specific finger used for mouse functions, the webcam captures the relevant frame, initiating further processing operations.
4. During this phase, the system identifies the lifted finger by utilizing the tip ID obtained through MediaPipe, along with the corresponding fingertip coordinates. Based on this information, the system executes the appropriate mouse function corresponding to the detected finger position.

Tip ID	Respective finger
0	Thumb Finger
1	Index Finger
2	Middle Finger
3	Ring Finger
4	Little Finger

Table 5.1 Finger Assignment Table

5. Mouse and keyboard Functions Depending on the Hand Gestures and Hand Tip Detection Using Computer Vision
   a. If the index and middle fingers with tip id 1 and 2 are up then the mouse is moved around the window of the computer by using the AutoPy package.
   b. If the index finger is depressed and the middle finger is up, the Left-click operation is performed by using the PyautoGUI package.
   c. If the index finger with tip id 1 is up and the middle finger with tip id 2 is depressed, then the Right-click operation is performed by using the PyautoGUI package.
   d. If both the thumb finger with tip id 0 and index finger with tip id 1 are up then the volume-up operation is performed by using the PyautoGUI package.
   e. If no finger is up then the grab operation is performed by using the PyautoGUI package.
   f. if all fingers are up then the mouse is put into the neutral position.

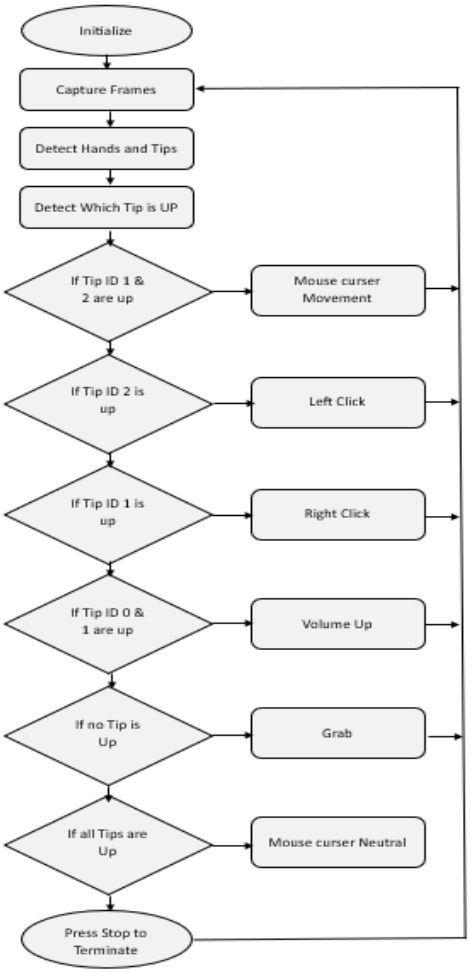


Fig 5.1: Flow-chart of the system

Model Creation

A model for a hand gesture recognition system for Windows control has been developed, necessitating the creation of a proprietary dataset due to encountered overfitting issues with available datasets. The initiative involved the collection of data comprising six distinct hand gestures aimed at executing various activities such as cursor movement, left click, right-click, no action, drag, and drop functionalities. This bespoke dataset was meticulously curated to

encompass diverse hand gestures essential for effective Windows control operations. The inclusion of these specific gestures facilitates comprehensive training of the model, ensuring robust performance in accurately recognizing and interpreting user hand gestures for controlling Windows functionalities. By leveraging this custom dataset, the developed system endeavors to mitigate overfitting challenges and enhance the accuracy and reliability of hand gesture recognition for Windows control applications.

Our Dataset Includes various sets of images for various purposes:

For the testing phase, we took a total of 3k images 300 for each gesture.

For training the model, we considered 2k images 200 for each gesture, and for the validation dataset, 500 images total images were generated.

VI. RESULTS AND DISCUSSIONS

While developing a hand gesture recognition system for computer control, we learned important lessons in the process, which will help us in the future. Even though the validation accuracy of 80.40% showed a strong base, we weren't content to accept "good enough." The model's accuracy increased to an impressive 85.90% in a real-world testing setting, indicating its potential for useful implementation. But the real test will be how well the model can interpret the subtleties of hand motions made by people. This is where the intriguing difficulties, or our model's own "demons," as you put it, were exposed by the classification reports.

Several issues arose during our analysis, particularly with movements such as "cross," "scissor," and "up," resulting in consistent model inaccuracies. These errors could stem from various sources. Firstly, these movements may exhibit similar hand configurations, leading to ambiguity in the model's recognition capabilities. This ambiguity becomes pronounced when distinguishing between subtle variations, such as a closed fist and a slightly open hand, particularly in low-quality images. Additionally, deficiencies in the training data might contribute to these inaccuracies. It's possible that the training set lacked an adequate variety of hand postures for these specific movements. Similar to teaching a child to differentiate between dog breeds, the model needs exposure to a broad range of hand gestures within each category. If the training set predominantly featured rigid "cross" gestures, for instance, the model may struggle to identify more relaxed variations of the same gesture. Despite these challenges, our utilization of various techniques yielded an accuracy rate of 85.9%.

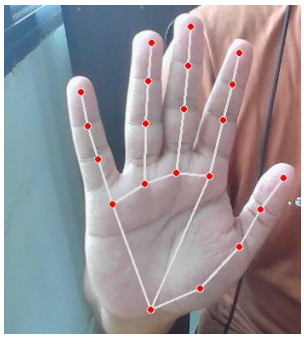


Fig 6.1: Recognition Hand, Mouse Neutral

If all tip IDs are up then the hand is recognized, which can be observed in Fig 6.1., When a hand is in this position mouse is in the neutral position. Meaning the hand can move freely and the mouse cursor won’t move.

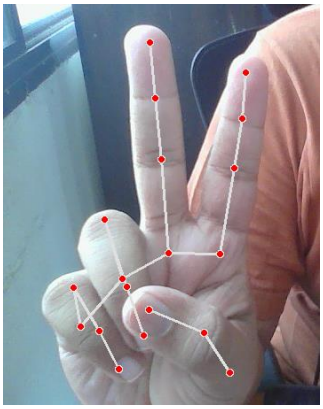


Fig 6.2: Gesture for mouse movement

If both index finger with tip id 1 and middle finger with tip id 2 are up then the Mouse movement operation is performed by using the PyautoGUI package as shown in Fig 6.2

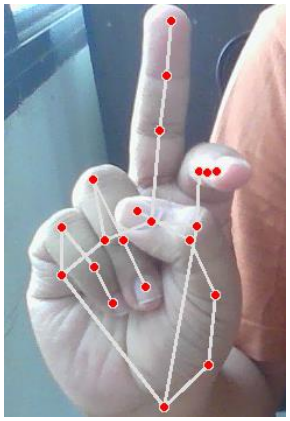


Fig 6.3: Gesture for left click function

If the middle finger with tip ID 2 and is up then the Left Click operation is performed by using the PyautoGUI package as shown in Fig 6.3

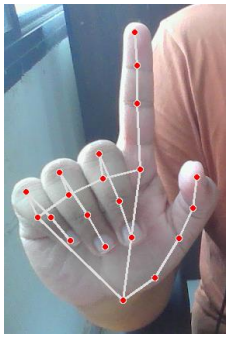


Fig 6.4: Gesture for right click function

If the index finger with tip id 1 is up and all other fingers are depressed, then the right click function is performed using the PyautoGUI package as shown in Fir 6.4

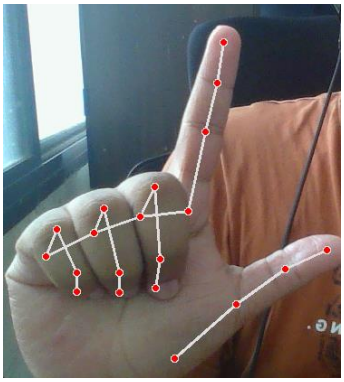


Fig 6.5: Gesture for volume up function

If both the thumb finger with tip id 0 and the index finger with tip id 1 are up then the volume-up operation is performed by using the PyautoGUI package as shown in Fig 6.5

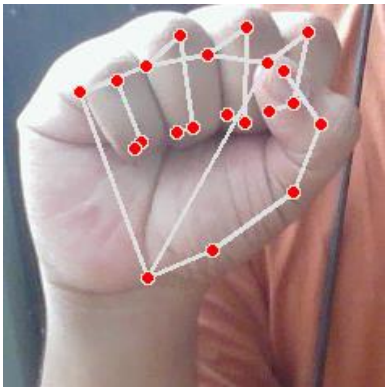


Fig 6.6: Grab function

If no finger is up as shown in Fig. 6.6, then the Grab operation is performed, this operation can be used to drag and drop operations or for selecting multiple items using the grab method by using the PyautoGUI package as shown in Fig 5.9.

VII. FLOW CHART

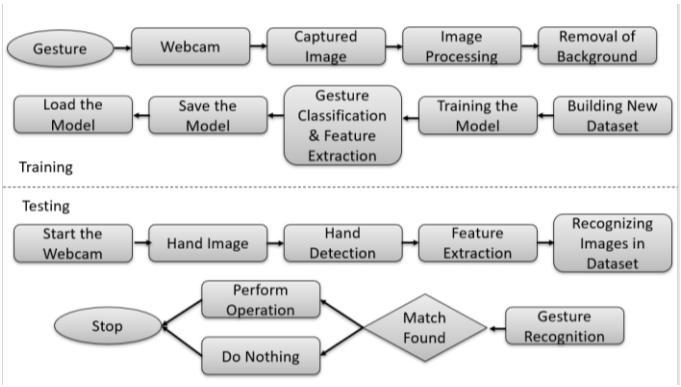


Fig 7: Flowchart of the Model

VIII. SYSTEM EVALUATION

When training a machine learning model, particularly those using deep learning architectures like Convolutional Neural Networks (CNNs), evaluating its performance involves looking at two sets of metrics: training accuracy/loss and validation accuracy/loss. Both accuracy and loss provide insights into how well the model is learning from the data.

Training Accuracy and Loss: These metrics focus on the model's performance on the training data. Training accuracy simply represents the percentage of training examples the model predicts correctly. The lower the training accuracy, the more mistakes the model makes on the data it's specifically trained on. Train loss, on the other hand, measures the average error between the model's predictions and the actual labels for the training data. Ideally, as the model learns from the training examples, the training accuracy should increase (meaning it's making fewer mistakes) and the training loss should decrease (indicating the errors are becoming smaller).

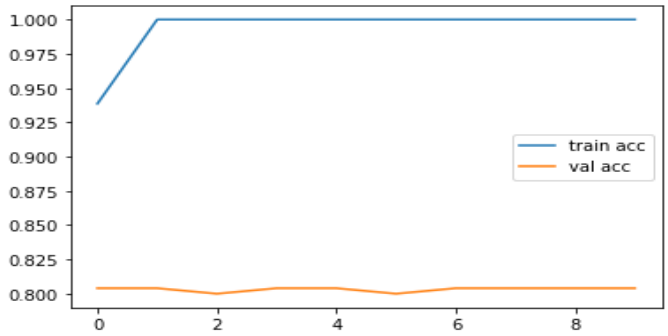


Fig. 8.1 Train accuracy vs Valid accuracy

Validation Accuracy and Loss: These metrics shift the focus to a separate dataset called the validation data. This data is crucial because it's not used to train the model. Instead, it serves as a real-world test to see how well the model generalizes to unseen data. Just like train accuracy and loss, validation accuracy reflects the percentage of predictions made correctly on the validation data, and validation loss measures the average error. In an ideal scenario, both validation accuracy and validation loss should show improvement as the model is trained. However, this isn't always guaranteed.

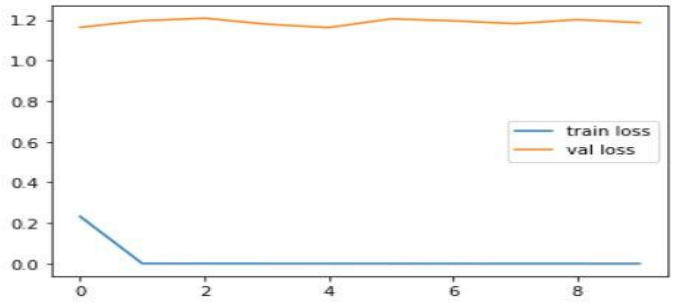


Fig 8.2 Train Loss vs Valid Loss

Analyzing these four values together (train/valid accuracy and loss) is key to understanding the training process. Here's where it gets interesting:

A crucial aspect is discerning between overfitting and underfitting scenarios. Overfitting is indicated by exceedingly high training accuracy coupled with markedly low training loss, while validation accuracy and loss remain stagnant or deteriorate.

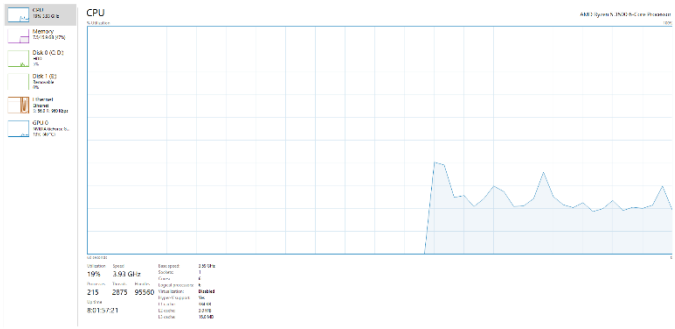
This phenomenon signifies the model's tendency to excessively memorize the training data, thereby failing to generalize effectively to unseen data. Analogously, overfitting resembles exhaustive preparation solely on practice test questions, resulting in exemplary performance on practice tests but poor outcomes on the actual exam. Conversely, underfitting is characterized by minimal improvements in both training and validation accuracy and loss metrics. This indicates the model's incapacity to grasp intricate data patterns, akin to attending lectures without comprehending the material, resulting in subpar performance on both practice problems and actual exams. Striking a balance between these extremes is paramount, aiming for a model that exhibits robust performance on both training and validation datasets. Achieving this equilibrium signifies the model's ability to extract salient patterns from training examples while retaining the capacity to generalize to unseen data, thereby making accurate predictions on novel instances. Vigilant monitoring of these metrics throughout the training process enables the derivation of valuable insights and facilitates adjustments to hyperparameters or model architecture, thereby guiding the training toward optimal performance. This holistic approach ensures the model's efficacy in practical applications while mitigating the risks of overfitting or underfitting.

	precision	recall	f1-score	support
0	0.30	0.73	0.43	84
1	1.00	1.00	1.00	200
2	1.00	0.87	0.93	229
3	0.99	0.96	0.98	206
4	1.00	1.00	1.00	200
5	0.86	1.00	0.93	173
6	1.00	1.00	1.00	200
7	1.00	0.47	0.64	422
8	0.47	1.00	0.64	94
9	0.96	1.00	0.98	192
accuracy			0.86	2000
macro avg	0.86	0.90	0.85	2000
weighted avg	0.93	0.86	0.87	2000

Fig 8.3 Confusion Matrix

CPU Performance

Recorded CPU Utilization while our project was working was 19% on average PC. The base speed was 3.93 GHz. and the average recorded memory usage was very low



VIII. CONCLUSION

This project investigated the feasibility of utilizing a deep learning model for hand gesture recognition in a human-computer interaction (HCI) context. The model, trained using Python and OpenCV, aimed to enable computer control through a set of ten predefined hand gestures. While the initial validation accuracy of 80.4% demonstrated a promising foundation, further testing revealed areas for improvement.

The evaluation metrics, particularly the classification reports, highlighted specific gestures that posed challenges for the model. Gestures like "cross," "scissor," and "up" were identified

as stumbling blocks, potentially due to their inherent visual similarities or a lack of sufficient variation within the training data.

These findings underscore the importance of meticulous training data curation and potentially refining the model architecture for enhanced performance. Addressing the identified shortcomings could involve enriching the training data with additional variations of the problematic gestures to capture their subtle nuances. Alternatively, exploring modifications to the model architecture might improve its capability to differentiate between visually similar postures.

Overall, this project successfully established the groundwork for a hand gesture recognition system using a deep learning model. The achieved accuracy of 85.9% during real-world testing demonstrates the model's potential for practical applications. However, the encountered challenges highlight the need for further development to achieve robust performance across a wider range of hand gestures. Future work will focus on refining the training data and potentially modifying the model architecture to enhance its ability to differentiate between visually similar gestures. This continuous improvement process will ultimately lead to a more user-friendly and intuitive HCI system based on hand gesture recognition.

REFERENCES

[1] S. Ahmad, "A usable real-time 3D hand tracker," proceedings of 1994 28th Asilomar Conference on Signals, Systems and Computers, Pacific Grove, CA, USA, 1994, pp-1257-1261 vol.2, doi: :10.1109/ACSSC.1994.471660.

[2] C. Patlolla, S. Mahotra and N. Kehtarnavaz "Real-time hand-pair gesture recognition using stereo webcam," 2012 IEEE International Conference on Emerging Signal Processing Applications Las Vegas, NV, USA, 2012, pp. 135-138, doi: 10.1109/ESPA.2012.6152464

[3] C. Jeon, O. -J. Kwon, D. Shin and D. Shin, "Hand-Mouse Interface Using Virtual Monitor Concept for Natural Interaction," in IEEE Access, vol. 5, pp. 25181-25188, 2017, doi: 10.1109/ACCESS.2017.2768405.

[4] C. Jeon, O. -J. Kwon, D. Shin and D. Shin, "Hand-Mouse Interface Using Virtual Monitor Concept for Natural Interaction," in IEEE Access, vol. 5, pp. 25181-25188, 2017, doi: 10.1109/ACCESS.2017.2768405.

[5] S. -H. Yang, W. -R. Chen, W. -J. Huang and Y. -P. Chen, "DDaNet: DualPath DepthAware Attention Network for Fingerspelling Recognition Using RGB-D Images," in IEEE Access, vol. 9, pp. 7306-7322, 2021, doi: 10.1109/ACCESS.2020.3046667.

[6] J. W. Smith, S. Thiagarajan, R. Willis, Y. Makris and M. Torlak, "Improved Static Hand Gesture Classification on Deep Convolutional Neural Networks Using Novel Sterile Training Technique," in IEEE Access, vol. 9, pp. 10893- 10902, 2021, doi: 10.1109/ACCESS.2021.3051454.

[7] M. N. Islam et al., "Developing a Novel Hands-Free Interaction Technique Based on Nose and Teeth Movements for Using Mobile Devices," in IEEE Access, vol. 9, pp. 58127-58141, 2021, doi: 10.1109/ACCESS.2021.3072195.

[8] L. Jiashan and L. Zhonghua, "Dynamic gesture recognition algorithm Combining Global Gesture Motion and Local Finger Motion for interactive teaching," in IEEE Access, doi: 10.1109/ACCESS.2021.3065849.

[9] J. Xu, H. Wang, J. Zhang and L. Cai, “Robust Hand Gesture Recognition Based on RGBD Data for Natural Human–Computer Interaction,” in IEEE Access, vol. 10, pp. 54549-54562, 2022, doi: 10.1109/ACCESS.2022.3176717.

[10] G. Park, V. K. Chandrasegar and J. Koh, “Accuracy Enhancement of Hand Gesture Recognition Using CNN,” in IEEE Access, vol. 11, pp. 26496-26501, 2023, doi: 10.1109/ACCESS.2023.3254537.

Citation: Pratik Pramod Alkutkar, Anurag Chandan Angal, Ameya Ajit Kabir and Prof. Ashwini A. Kokate, Windows Controlling Using Hand Gesture, International Journal of Artificial Intelligence Research and Development (IJAIRD), 2(1), 2024, pp. 158-175

Abstract Link: https://iaeme.com/Home/article_id/IJAIRD_02_01_014

Article Link:
https://iaeme.com/MasterAdmin/Journal_uploads/IJAIRD/VOLUME_2_ISSUE_1/IJAIRD_02_01_014.pdf

✉ **editor@iaeme.com**