EAI Endorsed Transactions
on Internet of Things
Research Article EAI.EU

Gaming using different hand gestures using artificial neural network

Prema S[1*], G. Deena[2], Hemalatha D[1], Aruna K. B[3], Hashini S[1]

[1]Vel Tech Rangarajan Dr. Sagunthala R&D Institute of Science and Technology, Chennai
[2]SRMIST, Ramapuram Campus, Chennai
[3]S.A. Engineering College, Chennai

Abstract

INTRODUCTION: Gaming has evolved over the years, and one of the exciting developments in the industry is the integration of hand gesture recognition.
OBJECTIVES: This paper proposes gaming using different hand gestures using Artificial Neural Networks which allows players to interact with games using natural hand movements, providing a more immersive and intuitive gaming experience.
METHODS: Introduces two modules: recognition and analysis of gestures. The gesture recognition module identifies the gestures, and the analysis module assesses them to execute game controls based on the calculated analysis.
RESULTS: The main results obtained in this paper are enhanced accessibility, higher accuracy and improved performance.
CONCLUSION: To communicate with any of the traditional systems, physical contact is necessary. In the hand gesture recognition system, the same functionality can be interpreted by gestures without requiring physical contact with the interfaced devices.

Keywords: Hand gestures; Physical controller; Gesture recognition; Gaming controls; Analysis of gestures

Received on 22 November 2023, accepted on 15 February 2024, published on 21 February 2024

doi: 10.4108/eetiot.5169

*Corresponding author. Email: premas89@gmail.com

1. Introduction

In the past decade, computer technology has made remarkable advancements, becoming an essential part of our daily lives. While keyboards have traditionally played a central role in Human-Computer Interaction (HCI), there are situations in the real world where they may not be the most suitable means of facilitating this interaction. Hand gestures, as a highly natural and intuitive HCI technique, have emerged as a promising alternative to replace traditional keyboards. Leveraging camera technology for vision-based computer control can eliminate the need for physical keyboards altogether.

An example of hand gestures in HCI is evident in the Nintendo Wii, which sold over 50 million units within its first year of release. This gaming console demonstrated how motion controls have not only shaped the future of the gaming industry but also significantly boosted video game sales. When it comes to one-on-one computer interaction, hand gestures used in HCI offer a Natural User Interface (NUI) that is highly intuitive and effective.

New devices and techniques for using hand gestures to control the cursor have been the subject of numerous investigations. The technique is made much more crucial by the usage of hand gesture recognition in HCI and sign language recognition [1]. The gestures can be recognized using the gesture recognition module. The analysis module will then use the derived analysis to perform game controls after analysing a human hand motion. This method is also beneficial for athletes who may have injuries or physical impairments. Through analysis, users might be able to utilise identical gestures to accomplish different functions.

2. Motivation

Gamers have long grappled with the tedious and monotonous challenge of operating a keyboard, mouse, joystick, and various other computer peripherals for extended periods. As players grow tired of this routine, it can ultimately lead to losses for game companies. It is crucial to innovate and introduce a fresh approach for gaming to create a more dynamic, aesthetically pleasing, and enjoyable gaming environment. Our motivation stems from the desire to develop a system that allows gamers to fully immerse themselves in their gaming experience without the frustrations of cumbersome equipment. This drive is fueled by the shortcomings observed in the gaming industry and the shared vision of enhancing player interaction. For new advances to be made and for players to have new opportunities to play games in novel ways, a revolution must first take place prior to all of this.

The user experience is elevated significantly by this. Our reliance on dated peripheral devices would decrease thanks to gesture control technology, which would also result in a reduction in the system's overall complexity. Initially, this technology was primarily envisioned for gaming purposes, as seen with the Xbox Kinect. Nevertheless, by expanding the application of motion and gesture control technology to various devices like personal computers, televisions, and beyond, it can cater to a wider range of uses, including everyday tasks such as scrolling, selecting, and clicking [2,3]

This device serves as a means to virtually control a computer by detecting hand gestures, essentially providing us with a virtual interface to interact with the computer. However, due to limitations in budget and schedule, producing the required hardware for a device of this nature was not feasible. The main goal in crafting this system was to design a tool that drew inspiration from Leap Motion [4,5].

3. Literature Review

Akula et al [1] have Proposed a distinctive application that integrates with the game. Input and interaction devices play a crucial role in gaming, enabling human interaction. In recent times, various technologies have been employed to control computer-based systems. Human interaction can take various forms, including gestures, speech, and text. Among these, gesture interaction is particularly well-suited for gaming. Additionally, the stochastic gradient descent approach to hand gesture recognition offers an efficient method to track both static and dynamic hand gestures.

Tanay et al [2] have developed a system for gesture recognition application that leverages a connected camera or webcam to detect and interpret human hand gestures. The application's primary function is to analyse these gestures and convert them into corresponding actions, utilising the default controls of a game. It also comes equipped with a comprehensive set of guidelines for identifying and interpreting hand gestures performed on the palm of a person's hand. It's important to note that the development of a fully robust hand gesture recognition system is an ongoing area of research and development. This system's work represents a preliminary contribution towards establishing a flexible foundation for future endeavours in this field.

Ahmed et al [3] have proposed a straightforward game system that relies on hand gestures to evaluate performance, cognitive load, comfort, and player satisfaction when utilising these gesture-based devices. The research findings indicated that participants exhibited a strong preference for and performed notably better when using Leap Motion and Kinect in comparison to RealSense. Leap Motion, in particular, displayed superior performance, or at the very least, matched the performance of Kinect. These findings were further supported by player feedback, which highlighted their positive experiences with these gesture-based devices. In light of these conclusions, delve into the potential applications of such devices for game designers and provide a series of design considerations to offer valuable insights for the development of gesture-based games.

Kanishk et al [4] have suggested that everyone, particularly individuals with disabilities, should have the opportunity to live independently. In recent decades, technology has been increasingly focused on empowering individuals with disabilities to take control of their lives to the greatest extent possible. An assistive system designed for the visually impaired, enabling them to gain awareness of their surroundings. It achieves this by utilising YOLO, a deep neural network-based object detection method, to quickly identify objects within images and video streams. The system is implemented using OpenCV in Python on a Raspberry Pi 3. The outcomes of this endeavour demonstrate the effectiveness of the proposed model in providing blind users with the ability to navigate unfamiliar indoor and outdoor environments. This is accomplished through a user-friendly device that incorporates person and object identification capabilities.

D. Perez et al [5] proposed object detection, a widely recognized field within computer technology, closely associated with computer vision and image processing. The main goal is to detect and recognize objects or instances belonging to particular categories (e.g., humans, flowers, animals) within digital images and videos. This technology has been extensively researched and applied in various domains, encompassing tasks like face detection, character recognition, and vehicle counting. Object detection finds application in diverse areas, including image retrieval and surveillance. This research delves into the fundamental principles of object detection, employing the Python 2.7-based OpenCV library as a vital resource for investigation and practical application.

Swiechowski et al [6] suggested the object detection system described here is designed to identify real-world objects within digital images or videos. These objects can belong to various classes, including humans, cars, and more. For effective object detection in images or videos, the system depends on several essential elements: a model database, a feature detector, a hypothesis generator, and a hypothesis

verifier. This document provides a comprehensive overview of the various methods employed for object detection, localization, categorization, as well as feature and appearance information extraction in both images and videos.

Kalyanam et al [7] suggested that Object detection and tracking are critical, primarily because they deal with the persistent challenges posed by changes in object movement, fluctuations in scene size, occlusions, modifications in appearance, shifts in ego-motion, and variations in illumination. In particular, the process of feature selection holds immense significance within the context of object tracking. This area of research remains significant for a wide range of real-time applications, including vehicle perception and video surveillance, among others. To tackle the obstacles associated with object detection and tracking, numerous algorithms focus on improving the continuity and consistency of video sequences. Conversely, a subset of methods leverages pre-existing information regarding object attributes like shape, colour, and texture. This research focuses on the analysis and discussion of tracking algorithms that amalgamate the aforementioned object parameters, aiming to provide a comprehensive solution to the complex task of object tracking in dynamic environments.

4. Implementation of Proposed System

Capturing Images: At this stage, the device's video camera is employed to capture an image of the user's hand, ensuring that the hand is clearly visible on the screen as per the user's requirement. The computer then identifies that the user is presenting their hand during this phase. To accomplish this, the code makes use of the MediaPipe Python package. Data is received from the video camera with the aid of MediaPipe 33 3D landmarks are inferred by MediaPipe using machine learning from a single image. Four separate segments of the hand image are created for improved processing. Moreover, grid lines are utilised to more accurately depict the hands and actions.

Image Manipulation: Unwanted noise and disturbances are eliminated at this stage. If this phase is skipped, the undesired information may interfere with the detection's accuracy. Filtering and smoothening are done here.

Feature Selection: In this study, the issue of feature extraction was addressed through the application of two distinct methods: hand contour and complex moments. These two extraction techniques were selected due to their utilisation of different approaches in feature extraction. Hand contour relies on a boundary-oriented approach, whereas complex moments employ a region-based approach. These feature extraction algorithms were utilised to resolve common challenges encountered in hand gesture recognition, such as scaling, translation, and rotation [6,7,8].

Categorization: In the categorization phase, neural networks are employed to recognize the gesture image based on the extracted features. In this stage, concerns related to the performance are thoroughly examined and convergence of the neural network algorithm. Artificial Neural Networks (ANN) have found extensive use as a classification method, particularly in real-world applications. This is due to their capability to operate in parallel and perform online training.

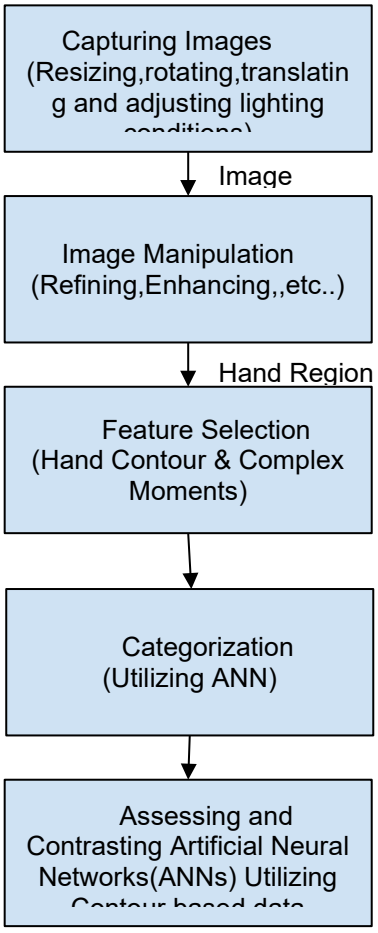

Figure: 1 Architecture of Proposed System

4.1 Issues & Challenges in Existing System

Using a webcam to accurately detect the hand is the first difficult task. For this, a computer vision library is required. In this system OpenCV is used because it is the most widely translated into many languages and is supported by a wide range of operating systems, including Windows and Android. It has a decent selection of common image processing features. Next, OpenCV for our IDE (Python 3x Idle) is set up. Also, fundamentals of using OpenCV must be understood. Skin recognition methods must be comprehended after understanding OpenCV, as well as background subtraction, image smoothing, noise reduction, and noise removal methods. The foundational application

based on the hand after accurately identifying it and mapping the movements must be developed. Given the high cost associated with top-tier cameras and sensors, this system opted to utilise a simple webcam instead [4,9,10].

4.2 Problem Statement

Hand gestures have a wide range of potential applications for interacting with systems, including video games, UAV control, medical equipment operation, and more. These gestures can also provide a means of interaction for individuals with disabilities, offering an alternative to traditional input devices like keyboards, mice, and touchscreens that require physical contact with the interface. However, a challenge in utilising hand gestures lies in interpreting them accurately. The same gesture performed by different individuals may appear differently while serving the same function. Deep learning techniques, notably Convolutional Neural Networks (CNNs), have risen as valuable tools for addressing the challenges of gesture recognition. Nonetheless, one limitation of deep learning approaches is their potential performance issues in real-world recognition scenarios, often demanding significant computational power for gesture processing [11,12].

4.3 Algorithm & Pseudo Code

Algorithm

- Open Visual Studio Code.
- Install the necessary libraries: OpenCV, Mediapipe and Python.
- Start writing the Code in the Python file.
- Import the required libraries.
- Create a video capture object.
- Initialise the hand tracking module from Mediapipe.
- Create a drawing utility for the landmarks detected by Mediapipe.
- Start the main loop to read frames from the video capture object.
- Perform hand gesture recognition using the landmarks obtained from step 8.
- Use a machine learning algorithm (Artificial Neural Networks) for this task.
- Display the output on the screen.
- Release the video capture object and destroy any open windows.

Pseudo Code

- Import the necessary libraries.
- Create a video capture object.
- Initialise the hand tracking module from Media pipe.
- Create a drawing utility for the landmarks detected by Media pipe.
- Start the main loop to read frames from the video capture object.
- Read a frame from the video capture object. While true:
- Convert the frame to RGB format for processing by Media pipe.
- Process the frame using Media pipe.
- Check if any hand is detected in the frame and get the landmarks.
- Perform hand gesture recognition using the landmarks obtained from above. Use a machine learning algorithm or a rule-based system for this task.
- Display the output on the screen.
- Release the video capture object and destroy any open windows.

4.4 Module Description

Pose Estimation

Posture estimation is a crucial component of hand gesture recognition since it aids in identifying and following the major points or landmarks of the hand. For recognizing hand gestures, MediaPipe offers a posture estimation package for Python. Human pose estimation from video feeds or real-time sources is a critical aspect in various domains, including full body gesture control, quantifying physical activity, and sign language recognition. MediaPipe Pose stands out as a framework for highly accurate body pose tracking. It operates by taking input from RGB (Red Green Blue) video frames and deducing the positions of 21 3D landmarks across the entire human body.

Current state-of-the-art methods often depend on high-performance desktop computing setups for inference, and this approach excels in terms of performance, delivering impressive real- time results. Leveraging machine learning algorithms, it becomes possible to recognize hand gestures by analysing the landmarks obtained from MediaPipe's pose estimation module.

In Figure 2, 21 3D landmarks of Hand show the landmarks of the hand marked on the entire hand. The numbering of the landmark starts from 0 and ends with 20 and hence the complete human hand has 21 3D landmarks on the hand to play games using hand gestures. According to the landmarks identified and then the shape of the hand shown to the camera is finalised and edges of the hand are recognized, and action is performed.

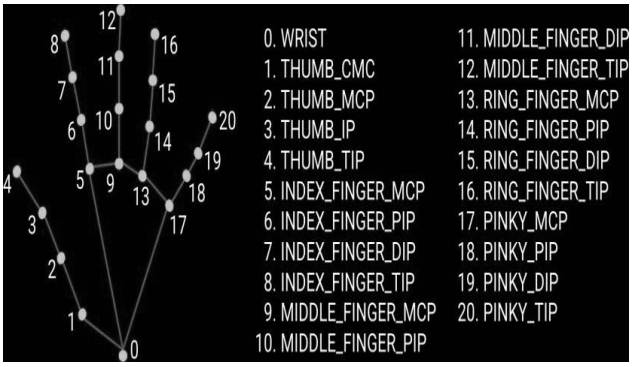

Figure:2 21 3D landmarks of Hand

Preprocessing

Preprocessing is a crucial module in the recognition of hand gestures since it helps to improve the quality of the input data, reduce noise, and increase the precision of the recognition algorithm. For the purpose of recognizing hand gestures, Python's OpenCV library offers a variety of image preprocessing tools. In this Phase the BGR (Blue Green Red) image is converted to RGB image. The system's computational complexity can be decreased by converting the image, which can also increase recognition precision. In this Preprocessing to improve the process performance, prepare a model or image as not writable. As completed or trained the model then made a model or image as writable and covert back the RGB image to BGR image [13,14].

Feature Extraction

The process of feature extraction is a pivotal component in hand gesture recognition. OpenCV in Python offers a range of functions for performing feature extraction in the context of hand gesture recognition.

In the field of image processing and pattern recognition, feature extraction serves as a specific type of dimensionality reduction. When dealing with large input data that may contain considerable redundancy, this process involves transforming the data into a more compact set of features, commonly known as a feature vector. This transformation of input data into a reduced feature set is termed "feature extraction." This module focuses on analysing features such as the height, width, finger counts, and image coordinates of the hand visible to the camera.

Segmentation

In the process of recognition, the goal is to distinguish the hand from the background and extract information suitable for classification. OpenCV for Python offers several segmentation techniques that can be applied in hand gesture recognition. This segmented hand can then be used for classification purposes, particularly in identifying characters as letters and converting the image into text.

Once the image is cleaned and transformed into a binary format containing only the text, it is saved, and memory resources are released. This step is crucial for enhancing system speed. Following segmentation, the classification phase comes into play. The gesture image is identified using neural networks, leveraging the extracted features for recognition. In this phase, various challenges related to neural network algorithm recognition and convergence are analyzed. ANN serve as a prevalent classification method, especially in real-world applications, owing to their capacity for parallel processing and online training.

Import Libraries

Import necessary libraries, in our system only three libraries are required.

```
import mediapipe
as mp import cv2
import pydirectinput
```

Initializing model

Initialization of the Holistic model and the utilisation of drawing utilities for the purpose of detecting and illustrating landmarks on the image.

```
mp_draw=mp.solutions.drawing_utils
mp_holistic=mp.solutions.holistic
mp_pose=mp.solutions.pose
```

Figure 3: Initialising Holistic model

Implementation

The procedure includes the identification of facial and hand landmarks within an image by employing the Holistic model. This model analyses the image, identifying landmarks for the face, left hand, and right hand, while also detecting the pose of the hand movement.

To accomplish this, a continuous stream of frames is captured from the camera using OpenCV. To obtain a live webcam feed, the initial line in the main function is modified to "cap = cv2.VideoCapture(0)." The captured BGR image is subsequently converted into an RGB image to facilitate

predictions using a pre-initialized holistic model.
The predictions generated by the holistic model are then stored in the "results" variable.The landmarks for the face, right hand, and left hand can be accessed using
"results.facelandmarks,"
"results.righthandlandmarks,"and
"results.lefthandlandmarks" respectively.

To visualise these landmarks, the "drawlandmarks" function from drawing utilities is employed, and the detected landmarks are illustrated on the image. The resulting image is used to portray pose movements according to the axis, enhancing the image window's interactivity.

5. Implementation & Testing

Input Design of Hand Gesture System

Figure 4: Input Image for Hand Gesture System

In Figure 4, Input Image for Hand Gesture System shows that the button present in hill climbing race (Brake) is operated through the gesture of closed palm. The registered gesture for performing the action of the brake button is closed palm. So,When the closed palm is shown to the camera the palm is recognized and the action of the brake button is performed. There are different buttons present in the game like brake, gas. Where both the buttons are replaced by the hand gestures.

Output Design of Hand Gesture System

Output Image for Hand Gesture System in Figure 5, describes that the hand gesture to operate the brake button is captured and in this figure the action is performed through operating the brake button. The registered gesture for performing the action of the brake button is closed palm. So, when the closed palm is shown to the camera the palm is recognized and the action of the brake button is performed. Hence the game can be played using different hand gestures. The game is played here with the hand gestures.

Figure 5: Output Image for Hand Gesture System

Testing

System Testing-Segmentation

Input

main.py

```
import cv2
import mediapipe as mp
import time
from directkeys import
rightpressed,leftpressed,uppressed,downpresse
d from directkeys import PressKey ,
ReleaseKey brakekeypressed=leftpressed
acceleratekeypressed=rightpressed
jumpkeypressed=uppressed
slidekeypressed=downpressed
time.sleep(2.0) #suspends execution for a no.of
sec
currentkeypressed=set( ) # build an unordered
collection of unique elements assigns to a
variable
mp draw=mp.solutions.drawingutils #solution
for drawingutils mp hand=mp.solutions.hands
#solution for handpose tipIds=[4,8,12,16,20] #
List of landmarks of tip of fingers
video=cv2.VideoCapture(0) #Start capturing
video from webcam,0 refers to computer or
default camera
with mphand.Hands(max numhands =1
,mindetection confidence=0.5, #max numhands
= no.of.hands to be detected at a time
mintracking confidence=0.5)as hands:
# mindetection confidence=min
confidence value for
detectedperson-model to be
```

```
considered as successful .
whileTrue:
#mintrackingconfidence=min
confidence value with which the
detection from the landmar k –
tracking model must be
```
considered as successful keyPressed=False brakepressed=False acceleratorpressed=False jumppressed=False slidepressed=False keycount=0 keypressed=0 ret,image=video.read( ) #Readvideoframe by frame & rechecks whether camera is working(1) or not(0) image=cv2.cvtColor(image,cv2.COLOR BGR2RGB) #Convert BGR image to RGB image as opencv gives us BGR image image.flags.writeable=False directkeys.py import ctypes import time SendInput=ctypes.windll.user32.SendInputright pressed=0x4D uppressed=0x48 downpressed=0x50 leftpressed=0x4B PULS=ctypes.POINTER(ctypes.culong) Class KeyBdInput(ctypes.Structure): fields=[("wVk",ctypes.cushort),("wScan ",ctypes.cushort) ,("dwFlags " ,ctypes.culong),("time",ctypes.culong),("dwExtr aInfo",PULS)] class HardwareInput(ctypes.Structure) : fields=[("uMsg",ctypes.culong),("wParamL",ct ypes.cshort),("wParamH",ctypes.cushort)] class MouseInput(ctypes.Structure): fields=[("dx",ctypes.clong),("dy",ctypes.clong), ("mouseData",ctypes.culong),("dwFlags",ctyp es.culong),("time",ctypes.culong),("dwExtraInf o", PULS)]

Test Result

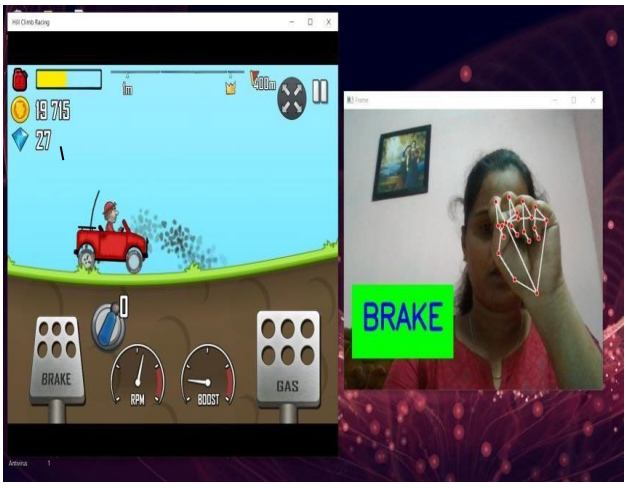

Figure 6: Hand Gesture System-Test result

In Figure 6, System testing results of the Hand Gesture System describes that the system testing is done. Here, it is verified whether the system is working or not and whether the game can be played using hand gestures and whether the webcam is activated, and frames are detected or not. the testing done using the system here. This System is completely working, and System Testing is done successfully. During system testing, the integrated components that have passed integration testing are used as input. The objective of integration testing is to identify any anomalies or inconsistencies that may arise when combining these integrated units.

6. Results and Discussions

Efficiency of the proposed system

The proposed system employs a shape-based methodology encompassing multiple stages for recognizing hand gestures, including palm recognition and finger counting. It represents an effort to consolidate various endeavours in the field of gesture recognition, with a primary focus on gaming applications. Currently, the system's primary domain of application is gaming.

In comparison to prior methods, the system has achieved an approximately 22 percent increase in success rates. To enhance its robustness, several obstacles present in earlier approaches have been addressed and eliminated. The ultimate goal of the system is to empower users to create personalised gestures for triggering specific events or actions across different domains.

However, it's important to note that the system cannot entirely replace the conventional computer keyboard. This limitation arises from the common use of computers in outdoor settings with suboptimal lighting conditions.

Comparison of Existing and Proposed system

Table 1: Comparison Table for Existing and Proposed System

	Classification	Accuracy
Existing Model	K-Means Algorithm & CNN	64.8%
Proposed Model	Artificial Neural Networks	86.3%

Existing System:(K-means algorithm and CNN)

In the current system, hand classification following detection is carried out using the K-means algorithm. Through the combination of the K-means algorithm model and CNN for classification, the system transforms detected gestures into actionable commands, primarily for game control purposes. During testing, this approach is applied to games such as Rock, Paper and Scissors, Hill Climb Racing, and Off-Road Super Racing. The system allows for control using both static and dynamic gestures, with dynamic gestures serving to modify game controls [10,11].

The results obtained indicate that HCI can be achieved with minimal hardware requirements, although the accuracy of the output is lower when compared to the proposed system. The proposed system aims to establish a unique application that integrates with games. Additionally, it introduces a stochastic gradient descent-based hand gesture recognition system, which efficiently tracks both static and dynamic hand gestures.

Proposed System: (ANN Algorithm)

The system under consideration tackles the task of creating a vision-based static hand gesture recognition algorithm with a specific focus on recognizing four commonly used static hand gestures: left, right, up, and down arrows. These gestures were selected due to their frequent use in communication and their potential applications, including creating a virtual keyboard for specific software.

Feature selection techniques are employed to handle typical challenges encountered in hand gesture recognition, including scaling, translation, and rotation. In the classification stage, neural networks are leveraged to identify the gesture image using the extracted features, and this phase includes an examination of issues related to neural network recognition and convergence. ANN are chosen as the classification method, as they are widely adopted, particularly in real-world applications, due to their parallel processing capabilities. The implemented proposed system aims to achieve higher accuracy compared to the existing system, enhancing the recognition of static hand gestures for improved performance.

7. Conclusion and Future Enhancements

Conclusion

A computer vision-based system for controlling the keyboard cursor using hand gestures has been developed using the Python language and the OpenCV library. MediaPipe package is used for highly accurate pose tracking. This system allows users to control the movement of a virtual keyboard key by tracking their hand, primarily for gaming applications. Various hand gestures are employed to activate different keyboard key functions. Although the system holds promise as a viable alternative to the conventional computer keyboard, it is important to note that certain limitations and restrictions currently prevent it from entirely supplanting the keyboard.

Future Enhancements

The application proposed in this paper can be seen as an initial stride in the domain of HCI applications, and there exists substantial potential for considerable improvements and advancements. It has the capacity for extension to include mouse cursor control through the incorporation of additional HCI concepts and OpenCV algorithms. To further enhance accuracy, the integration of Neural Networks-based logic can be explored. Tracking performance can be improved to ensure better results. For enhanced precision in hand gesture recognition, adopting a template matching approach along with a machine learning classifier is an option. Although this implementation may require a longer development timeline, it is expected to result in improved accuracy in gesture recognition.

Acknowledgements.

I thank my co-authors Dr. G. Deena, Ms. Hemalatha. D, Ms. Aruna K. B, Ms. Hashini S for their expertise and assistance throughout all aspects of our study and for their help in writing the manuscript.

References

[1] Akula G, Shitanshu R, Aditya D. Playing Games Using Hand Gesture Recognition. International Research Journal of Modernization in Engineering Technology and Science.2022; Vol. 04, pp.662-668.

[2] Tanay T, Vidya B. Hand Gesture Controlled Gaming Application. International Research Journal of Engineering and Technology (IRJET). 2021; Vol. 8, No. 4, pp. 3654.

[3] Ahmed S, Ali A. A Comparative Study of Hand-Gesture Recognition Devices for Games. National Science Foundation government Journal.2020; pp. 397-402.

[4] Kanishk C, Khushboo S, Mahak S, Mayank S. Gesture Recognition using OpenCV. International Journal of Advanced Networking Applications (IJANA).2018; Vol. 5, No. 4, pp. 3528.

[5] Perez D, Samothrakis S, Togelius J, Schaul T, Lucas S. The 2014 general video game playing competition. IEEE Transactions on Computational Intelligence and AI in Games.2016; Vol. 8, No. 3, pp. 229.

[6] Nhat V, Majed Q, Yu Z, Haoren Z, Cungang Y. Hand Gesture Recognition System for Games. IEEE Asia-Pacific Conference on Computer Science and Data Engineering (CSDE),2022; pp. 1-6.

[7] Świechowski M, Mandziuk J. Specialisation of a UCT- based General Game Playing Program to Single-Player Games. IEEE Transactions on Computational Intelligence and AI in Games.2015; vol. 8, no. 3, pp. 218-228.

[8] Richa D, Pooja L, Nongmeikapam T. Different Categories of Feature Extraction and Machine Learning Classification Models Used for Hand Gesture Recognition Systems. A Review, IEEE 8th International Conference for Convergence in Technology (I2CT). 2023; pp. 1-7.

[9] Urvil P, Sourabh R, Vipin S, Xing T. Gesture Recognition Using MediaPipe for Online Real Time Gameplay. IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology (WI-IAT).2022; pp. 223-229.

[10] Wu J, Kalyanam R, Givan R. Stochastic enforced hill-climbing. Journal of Artificial Intelligence Research.2011; Vol. 42, pp. 815–850.

[11] Rafi A, Rezki Y, Agus K. Hand Gesture Recognition for Controlling Game Objects Using Two- Stream Faster Region Convolutional Neural Networks Methods. International Conference on Information Technology Research and Innovation (ICITRI).2022; pp. 59-64.

[12] Perez D, Samothrakis S, Lucas S. Knowledge-based fast evolutionary MCTS for general video game playing. IEEE Conference on Computational Intelligence and Games (CIG'14).2014; pp. 1–8.

[13] Manoj, G, Manohar, V, Banoth, R, Prasad, S, Sreeja, P. Game Controlling using Hand Gestures. IEEE Conference on Advancements in Smart, Secure and Intelligent Computing (ASSIC), 2022; pp. 1-5.

[14] Elisa C, Tiemi S, Luciana Z. Playing a Computational Thinking Game using Hand Gestures. IEEE International Conference on Advanced Learning Technologies (ICALT),2019; pp. 105-109.