Smart Home Automation Using Hand Gestures

Dr Mary Jansi Rani G
Head of the Department
Department of Electronics and Communication Engineering
Kgisl Institute of technology hodece@kgkite.ac.in

Lakshmi Prabha M
Department of Electronics and Communication Engineering
Kgisl Institute of technology
Coimbatore, India
lakshmiprabha724@gmail.com

Abinaya M
Department of Electronics and Communication Engineering
Kgisl Institute of technology
Coimbatore, India
abinaya.meenatchisundharam@gmail.com

Jothi N
Department of Electronics and Communication Engineering
Kgisl Institute of technology
Coimbatore, India jothinanjundan@gmail.com

Kaviya M
Department of Electronics and Communication Engineering
Kgisl Institute of technology
Coimbatore, India
mkaviya2404@gmail.com

Abstract*— A natural way to accomplish Human-Computer Interaction (HCI) is through the visual interpretation of hand gestures. In this paper, we describe a method for configuring a smart home in which a hand gesture recognition system can be used to control the appliances. More precisely, this recognition system effectively differentiates between pre-trained gestures and correctly identifies them to operate the appliances by using transfer learning, a machine learning technique. In order to activate the appliances, the gestures are sequentially recognized as commands. An Arduino Uno microcontroller, which is connected to the PC where the gesture recognition is actually implemented, is used to control a set of LEDs that represent the appliances in order to demonstrate the proof of concept.

.Keywords — Human-Computer Interaction (HCI),Hand Gesture Recognition, Visual Interpretation, Machine Learning, Arduino Uno, Appliance Control

I. INTRODUCTION

What was once thought to be impossible is now only a few clicks away thanks to the development of highperformance computers and contemporary, superior data processing tools. Automation, object recognition, and computer vision are not just lofty concepts; they are realities of the modern world. Innovative research in the domains of artificial intelligence, machine learning, and other related fields has sparked this revolution. This means that with the aid of these fantastic tools, there are now more recent opportunities to address fascinating global issues and use cases.

The study of how people interact with computers is known as human-computer interaction [1][2]. The aim of this research is to develop an idealistic approach to interaction in which human-computer communication perfectly replicates human-to-human communication. It ought to feel equally instinctive and natural. Numerous hypotheses, research projects, demonstrations, and projects have been offered to help us better understand this area of study.

Our concept is a hybrid of computer vision and humancomputer interaction, with machine learning and artificial intelligence being used to implement computer vision. Through the use of a personal computer's webcam, we have created a system that can recognize and distinguish between various gestures. Depending on which gesture is detected, we can then automate specific actions.

Our idea can be divided into three main components:

Using the system's input to train the machine learning model, recognizing and correctly classifying incoming gestures, as well as automating the action sequence unique to each legitimate gesture.

Together, these parts make up a computer vision-based gesture recognition system, which we believed to be a very intuitive and natural method of computer interaction. This prototype is a smart home enabler since we are putting it into practice to act as a control unit for automating the appliances in our house.

An Arduino Uno microcontroller with an ATmega 328 microprocessor on board that is connected to a personal computer was used to illustrate how the system operates . Additionally, a set of LEDs is connected to the Arduino to symbolize the appliances found in a home. We are using gesture inputs that are connected to the PC's webcam to control the LEDs. This is accomplished through the definition and training of gestures for the commands "START," "STOP," and a few gestures for appliance identification. Thus, to make the first LED glow, do the following: gesture sequence has to be provided - “START” “LED1”.

This example can be expanded to support numerous appliances and a wide range of additional commands. We chose these fundamental commands because we believed they were the most frequently used and adequately illustrated how our concept operated. A thorough explanation of the work completed is provided below.

II. LITERATURE SURVEY

A1. A glove-based hand gesture recognition system was developed by Maria Eugenia Cabrera, Juan Manuel Bogado, Leonardo Fermin, Raul Acuna, and Dimitar Ralev . They have presented a novel glove with accelerometer sensors attached in this paper. These sensors determine the hand's orientation along the three axes and the degree of flexion on each finger. In order to recognize the gestures, they

subsequently feed the sensor-generated data into neural networks.

A2. Another glove-based gesture recognition system was created by Yangsheng Xu and Christopher Lee [6]. The orientation data is recorded by sensors that are embedded in the glove.

With just one or two examples of each letter, they can use neural networks to create a gesture recognition system that can classify up to 14 letters from the hand-motioned alphabets.

The following research papers made note of the following points:

1. It is very laborious and not at all natural to wear this mechanical clothing, which is the glove.

2. Only a limited number of different gestures can be recognized by the mechanical glove.

A3. A method for estimating hand pose was proposed by Etsuko Ueda and Yoshio Matsumoto [7]. This method, which is a subset of the Vision-based Gesture Recognition System, involves taking several pictures of the hand area with a camera system that can capture multiple viewpoints before creating a "Voxel."

The Hand and Voxel models are then used to fit a threedimensional model.

The gesture information is then extracted by analyzing this three-dimensional model.

The research paper made note of the following points:

1. Using inputs supplied as an image through a camera setup, a new model is created.

2. A very intricate fitting process is used to create the hand model.

III. SYSTEM ARCHITECTURE

The proposed system consists of four major layers:

1. Input Layer (Image Acquisition)
2. Processing Layer (Gesture Recognition Engine)
3. Control Logic Layer (Command Mapping)
4. Output Layer (Device Control via GPIO)

Each layer is designed to ensure real-time responsiveness and modularity, enabling scalability and adaptability across different smart environments

3.1 Input Layer – Image Acquisition

This layer is responsible for capturing live video feed using a camera device. A standard USB webcam or the RaspberryPi Camera Module is used, offering resolutions between 640x480 to 1280x720 pixels. The camera is mounted in a fixed position to capture a frontal view of the user's hand. Key components include:

1.Frame Grabber: Continuously captures video frames at 2030 fps.

2.Resolution Normalizer**:** Downscales the frame to a fixed input size (e.g., 64x64) for consistent processing.

3.Preprocessor: Converts RGB to grayscale, applies Gaussian blur, and segments the hand region using contour detection and background subtraction techniques.

3.2 Processing Layer – Gesture Recognition Engine

At this stage, the preprocessed image is sent through a Convolutional Neural Network (CNN) trained on a dataset of labeled gestures.

CNN Model Architecture**:**

1.Input: 64x64 grayscale image

2.Conv Layer 1: 32 filters, 3x3 kernel, ReLU

3.MaxPooling **1**: 2x2 pool size

4.Conv Layer 2**:** 64 filters, 3x3 kernel, ReLU

5.MaxPooling 2

6.Flatten Layer

7.Dense Layer**:** 128 neurons, ReLU + Dropout (0.5)

8.Output Layer: Softmax classifier for 5 gesture classes

The model is trained using a custom hand gesture dataset with diverse background, lighting, and hand orientation conditions. Training uses `Adam` optimizer and categorical cross-entropy loss.

Model Performance**:**

Training accuracy: ~98%

Validation accuracy: ~94% Inference

time: ~0.05 seconds per frame

Noise Handling**:**

Temporal smoothing techniques are optionally applied to avoid accidental gesture misclassification.

3.3 Control Logic Layer – Command Mapping

Once a gesture is recognized, the control logic layer translates it into a specific command to control a smart device.

This mapping is implemented using Python data structures (like dictionaries) for fast, readable, and flexible command associations.

Each gesture is linked to a unique home automation action, allowing real-time device control through simple hand signs.

The logic is handled using event-driven programming where gestures trigger functions that control devices via GPIO pins.

The system is designed to execute only one command at a time to avoid conflicts or accidental multiple device activations.

Debounce logic or temporal filters can be applied to prevent gesture noise or repetitive command firing.

The modular design allows new gestures or device actions to be added easily, making the system scalable and customizable.

3.4 Output Layer – Device Control Interface

- This layer is responsible for physically executing the commands generated by the control logic layer by interacting with smart home appliances.

- The system uses the GPIO (General Purpose Input/Output) pins of a Raspberry Pi (or Arduino in alternative setups) to control electrical devices.

- Electrical appliances (e.g., lights, fans, TV) are connected to the Raspberry Pi via relay driver circuits that safely handle high voltage.

3.5 Optional Modules

To enhance the functionality and flexibility of the smart home automation system, several optional modules can be integrated. One such module is a Flask-based web dashboard, which provides a browser-accessible interface displaying real-time system status, logs of recent actions, and manual control buttons for each connected device.

This can be particularly useful for remote access or administrative control. Another extension is the inclusion of speech and gesture fusion, where voice commands can complement gesture recognition to make the system more inclusive and adaptive.

For instance, users can use voice input in situations where gestures are inconvenient, such as while cooking or carrying items. Additionally, the system can be configured for Wi-Fienabled control through communication protocols like MQTT or HTTP, enabling integration with mobile applications or cloud-based automation platforms like Google Home or Alexa. These modules collectively make the system more scalable, user-friendly, and suitable for a broader range of smart home scenarios.

IV. SYSTEM OVERVIEW

As previously mentioned, the concept work can be divided into clear steps, which include: training the model using userprovided input; identifying inputs during runtime; and, lastly, automating the chain of events based on the input that the system has identified.

A. MACHINE LEARNING MODEL TRAINING:

We made the decision to develop our own machine learning model in order to truly recognize the input that was supplied.

A wide variety of models have been created with the intention of identifying and detecting images as well as other gestures. However, since employing a generic machine learning model would require adapting and scaling our own algorithm theory to match that of the generic model, we have chosen to train our own model instead.

We have the freedom to design the system to our specifications by using our own model, which allows us to shape and architect the flow to perfectly fit the system's intended use.

This project's integration of computer vision, machine learning, and embedded systems highlights the possibility of developing effective and easily accessible smart home solutions. Additionally, the system is an affordable and scalable way to improve home automation because it relies on easily accessible hardware and open-source software frameworks. This is especially advantageous for people who have mobility issues or are looking for more hygienic ways to interact with others. "

MobileNet employs depth-wise separable convolutions, which essentially entails that there are several convolutional layers with each layer having a neuron that is only connected to a subset of the input image and all of the neurons having the same connection weights.

Rather than having to write all the code and build a machine learning model from scratch, MobileNet gives us the freedom to customize our own machine learning model. By sharing the model's parameters, fewer computations are required to recognize an image in this situation.

MobileNet is what we use. This is because, thanks to TensorFlow.js, machine learning algorithms can now be run and executed directly within a computer's browser rather than requiring the download of additional software.

The TensorFlow engine powers Keras in the background, making it incredibly user-friendly and cross-platform compatible.

Thus, this is the background configuration of the image recognition system. When the server is started, this phase starts automatically when the index file is run. There are several options available on the webpage after it has successfully loaded in the browser.

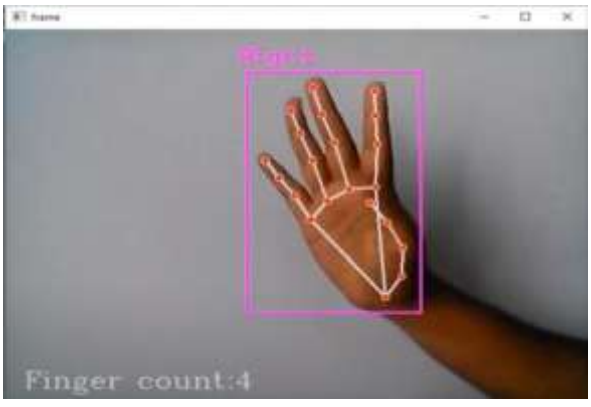

Figure 1

The left side of the home screen has options for fine tuning.

RATE OF LEARNING: In deep neural network training, optimization algorithms typically compute the error in the state of the current model and the examples provided as training data. The algorithm then uses backpropagation to update the

weights. The learning rate, also referred to as step size, is the rate at which the weights are updated.

Typically, this value falls between 0.0 and 1.0.

BATCH SIZE: This phrase, which basically describes how many training examples the model uses in each iteration, is used in machine learning.

EPOCHS: This machine learning term describes the number of times the entire training dataset is supplied to the algorithm in its entirety.

UNITS HIDDEN: Between the input and output layers of a machine learning model, there are several layers. These layers contain neurons known as Hidden Units, which are concealed from the external system.

There are options to add images at the bottom of the page.

This is an extremely significant and vital component of the system. In order to train our own model, the MobileNet model receives gesture inputs in this section. By clicking the Train Model button on the right side of the page, we can begin training our model as soon as the input gesture images are ready.

The images are sent for training when that button is pressed, and our model is produced shortly after. Following model creation, the cost function is computed and the model's resulting loss is shown appropriately. The model can then be downloaded or tested.

The following step, which determines the input provided to the webcam, uses this method that has been trained and tuned by MobileNet.

B. RECOGNIZING INPUTS WHILE RUNNING

We can begin making gestures to the webcam during runtime once the model has been trained. The model will receive this new gesture and attempt to categorize it into one of the training inputs, also known as classes.

The training class that the input matches or corresponds to will be highlighted after the input has been identified.

Following the identification of a class, a list is created in the background. The ClassId that identifies the chosen class is contained in this list. Next, a pattern that mimics a command is looked for in this list, which now includes a list of several ClassIds. Additionally, it is sent for automation if the pattern matches.

It operates as follows:

The ClassId is added to the list once a class has been identified.

We have determined that "START" and "STOP" are the two primary commands in the suggested system. Along with gestures for the devices during training, gestures for these commands are already available.

It pushes the matching ClassId to the zeroth index position after clearing the list when the class "START" or "STOP" is encountered. Additionally, the corresponding ClassId is only added to the list whenever an Appliance is encountered if the list has a START or STOP ClassId at the top.

The list is then sent as a POST method to the server, which has been configured using NodeJS's Express Framework[10], once the ClassId of any appliance has been added.

Essentially a Node JS web application framework, Express.js was created especially for single-page, multi-page, or hybrid web applications.

Figure 2

Since it is very difficult to send any data during runtime from a web browser to another physical device without the use of a local server, this POST method of sending the list has been used.

C. THE GESTURE SEQUENCE AUTOMATION

The Arduino Board attached to the computer system must now receive the list after it has been transmitted to the server.

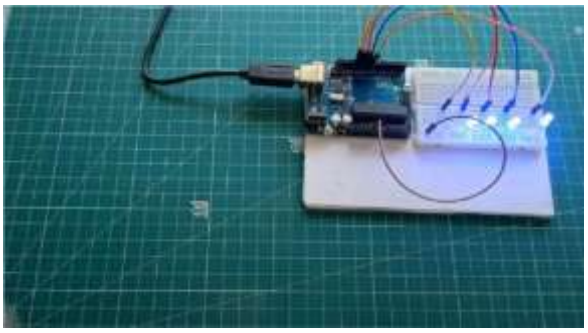

Figure 3

The home environment system has been demonstrated and replicated using an Arduino Uno. The microcontroller has LEDs attached to it that simulate the appliances found in a home system.

We have developed a JavaScript code file that includes the code to control the LEDs based on the input supplied by the list on the server in order to accomplish the automation, or controlling the lights in the current scenario.

The server is also home to this JS file. We used a framework known as JOHNNY FIVE[11] to transmit the JS code to the Arduino. With the help of this framework, an Arduino can be controlled using JavaScript. Additionally, a

local server must be set up and operational, which we have already done with the Express Framework.

The Arduino Board can be controlled by Johnny Five using classes and methods. These techniques involve first defining the Arduino Uno as an object, after which we can control its toggling by defining the pins as LEDs using prebuilt functions. This entire process is carried out within the JavaScript file. However, we must upload the "Standard Firmata"[12] code to the microcontroller in order to initialize the Arduino. The Arduino IDE's example section contains this code. To control the Arduino Uno after the code has been

uploaded, all we need to do is plug it into the computer and launch the JavaScript file on the server.

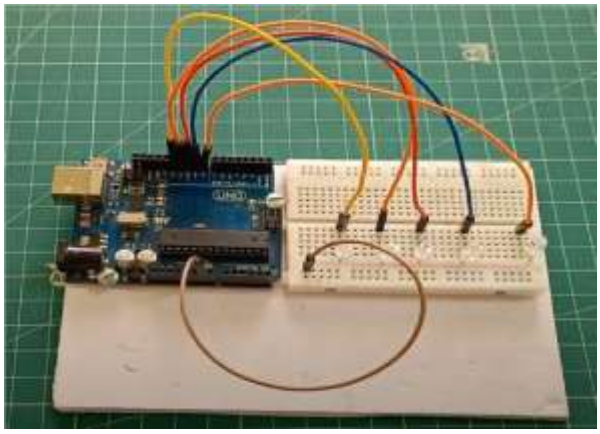

Figure 4

We must set up the server locally and execute all of these files on it because we have used JavaScript, a scripting language, to run continuously throughout the runtime. Therefore, after the server is started, the system's Index file—which is essentially the most crucial file and also the file that contains the home page—must be run. The JS file that governs the Arduino will receive the recognized gesture sequence from this page, which will also load the model and train the new inputs.

After parsing the list, this JS file—which makes use of the Johnny Five framework—will automatically toggle the LEDs in accordance with the results.

V. CURRENT LIMITATIONS AND FUTURE SCOPE

Replicating the same situation at a live product stage is still not possible, despite the fact that this proof of concept operated with more than adequate accuracy. In the real world, the input must be captured with a high resolution and separate camera setup, but we used the webcam built into the personal computer to capture an image.

Although the suggested system allows us to download the fully trained model, using it again in the web browser—where our project is carried out—is quite difficult. Therefore, before actual recognition occurs, we must always provide a fresh training sample from the user as input whenever the concept needs to be demonstrated.

However, this can be fixed in the future when the system as a whole is switched from a personal computer to a standalone electronic device.

Despite the fact that this system is intended to be a closedloop, constantly-running system in the future, the current proof of concept is unable to incorporate that capability because of web browser load limits. In reality, this was fixed by manually clearing the request list buffer and configuring the server.

Lastly, it is extremely difficult to replicate the same gestures during training under various conditions, despite the accuracy that was attained with the aid of precisely aligned training inputs. Because the machine learning portion occurs through the browser, there is a possibility that extremely high latency and errors will infiltrate the system, leading to runtime gesture misidentification. After the gesture recognition system has been moved to a separate system, this can be fixed in the future with the use of high resolution cameras and strong processors.

VI. CONCLUSION

Hand gestures are a natural, intuitive, and increasingly popular form of human-computer interaction, with numerous applications in diverse fields such as virtual reality, gaming, sign language recognition, and mobile device navigation. Unlike conventional input methods like keyboards or touchscreens, gesture-based systems offer users a more immersive and hands-free experience, allowing for seamless interaction with technology. Vision-based hand gesture recognition systems, in particular, hold significant promise, providing a more accessible and ergonomic alternative for controlling devices and environments.

However, despite the promising potential of hand gesture recognition, it remains a challenging task due to various obstacles such as hand shape variability, lighting conditions, background noise, and camera quality. The system presented in this paper is an attempt to tackle these challenges by using computer vision to recognize static hand gestures for controlling home automation devices. By leveraging a convolutional neural network (CNN) for gesture classification, the system demonstrates the ability to control devices such as lights and fans with simple hand gestures, all through the webcam on a personal computer.

While the system has shown positive results in controlled environments, its current performance is not without limitations. Factors such as the resolution of the webcam, the speed of the processing server, and the quality of training examples may contribute to slower response times and occasional misclassifications. Furthermore, the system's effectiveness could be impacted by external conditions like lighting or user positioning. To address these issues, future research should focus on improving the accuracy of the gesture recognition model by incorporating a larger and more diverse dataset, implementing advanced techniques like data augmentation, and optimizing the processing capabilities through edge computing solutions such as the Raspberry Pi or specialized hardware accelerators.

One of the most promising directions for future work is the integration of this gesture-based control system into more robust and portable devices. Hosting the system on a standalone platform or directly within a web browser can offer significant advantages in terms of accessibility and scalability. For example, smart home environments can be easily controlled using gestures via affordable, widely available devices like Raspberry Pi or even smartphones. This flexibility could significantly lower the entry barrier for smart home technology, making it more accessible to a broader audience.

In conclusion, this work demonstrates the viability of using hand gestures for controlling home automation systems, presenting a promising solution for hands-free interaction. While there are still challenges to overcome, the idea is technically sound and opens the door for further research and innovation in the field of gesture-based human-computer interaction. By improving accuracy, reducing latency, and scaling the system for broader applications, future advancements could transform the way we interact with our

smart environments, making them more intuitive, responsive, and user-friendly.

REFERENCES

[1] Donald A. Norman, "Design Principles for Human Computer Interfaces", 1983.

[2] Bergman, Johnson, "Towards accessible Human-Computer Interaction", 1997.

[3] Rajesh Singh, Anita Gehlot, Bhupendra Singh, "Introduction to Arduino and Arduino IDE and toolbox_arduino_v3", 2019.

[4] P. Voštinár, N. Klimová, J. Škrinárová, "Before We Start Arduino", 2019.

[5] Eugenia Cabrera, Maria & Manuel Bogado, Juan & Fermín, Leonardo & Acuña, Raul & Ralev, Dimitar, "Glove-Based Gesture Recognition System", 2012.

[6] Christopher Lee and Yangsheng Xu, "Online, interactive learning of gestures for human robot interfaces" Carnegie Mellon University, The Robotics Institute, Pittsburgh, Pennsylvania, USA, 1996

[7] Etsuko Ueda, Yoshio Matsumoto, Masakazu Imai, Tsukasa Ogasawara. "Hand Pose Estimation for Vision-based Human Interface", 2003

[8] Howard et al, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications", 2017.

[9] Tao Sheng, Chen Feng, "A Quantization-Friendly Separable Convolution for MobileNets", 2018

[10] S. L. Bangare, S. Gupta, M. Dalal, A. Inamdar "Using Node.Js to Build High Speed and Scalable Backend Database Server", 2016 [11] Manoj Kumar, Kailasa Akhi, Sai Kumar Gunti, Sai Prathap Reddy,
"Implementing Smart Home Using Firebase", 2016.