Gesture Detection Using Accelerometer and Gyroscope

Raghav Gupta, Shashank Chaudhary, Akshat Vedant, Niladri Paul Choudhury, and Vandana Ladwani

1 Introduction

Humans and machines, including computers, may now communicate more quickly because of recent electronics and sensor technology improvements. For IoT and universal computing, this human–machine interface (HMI) system will become increasingly important [1]. In most circumstances, communication begins when a machine (or an object) receives and interprets a human's purpose (or the user). As a result, the HMI requires an input device to record the user's intent.

Human gestures provide for a more natural approach to HMI input. Human body language is an intuitive communication technique for conveying, exchanging, interpreting, and understanding people's thoughts, intentions, and emotions. As a result, physical language emphasizes or complements spoken language. It is a language in and of itself. Thus, human emotions, such as hand gestures, should be included for HMI input [2]. Gesture-based interactions are one of the most comfortable and straightforward ways to communicate. On the other hand, gesture recognition has various challenges before becoming widely recognized as an HMI input.

R. Gupta (✉) · S. Chaudhary · A. Vedant · N. P. Choudhury · V. Ladwani
PES University, Bangalore, India
e-mail: raghavjpr@gmail.com

S. Chaudhary
e-mail: shashankchdhry@gmail.com

A. Vedant
e-mail: akshat.shanky@gmail.com

N. P. Choudhury
e-mail: nilpc06@gmail.com

V. Ladwani
e-mail: vandanamd@pes.edu

N. R. Shetty et al. (eds.), *Emerging Research in Computing, Information, Communication and Applications*, Lecture Notes in Electrical Engineering 928,
https://doi.org/10.1007/978-981-19-5482-5_9

Human hand motions are substantially less diversified than the tasks required by the HMI, which poses a considerable challenge. The functions of an (HMI) are more varied and complex. In the case of smartphones, this diversification tendency may be seen. Only a decade ago, a variety of handheld electronic devices, such as mp3 players, cell phones, and calculators, coexisted to meet various human needs. On the other hand, almost all these tasks have now converged into a single mobile device: the smartphone. On the other hand, all human intentions are only conveyed by swiping or tapping fingers on a smartphone's touch screen.

When it comes to HMI inputs, one prevalent approach is gesture-based interaction [3]. There are two hand gesture recognition techniques: vision-based recognition (VBR) and sensor-based recognition (SBR). There have been studies in gesture recognition, but most rely on computer vision. The efficiency of vision-based approaches or the operation of such devices is highly dependent on lighting conditions and camera-facing angles. It is inconvenient, and such limitations often limit the technology's usage in specific environments or for certain users.

Sensors include electromyography, touch, strain gages, flex, inertial, and ultrasonic sensors [4]. The most often utilized sensors are inertial sensors [5, 6]. Sensors with an accelerometer, gyroscopes, and magnetometers are inertial sensors.

Sensor-fusion algorithms frequently combine many sensors. For example, a glove with several wearable sensors has been claimed to monitor hand motions [7]. A 3D printer was used to create the glove housing, which includes flex sensors (on fingers), pressure sensors (at fingertips), and an inertial sensor (on the back of one's hand).

Inertial sensors are used to track hand motions in numerous sensor-fusion algorithms. Additional hand data, such as finger snapping, hand grabbing, or finger-spelling, is detected by other sensors, such as EMG. [8, 9]. Inertial and EMG sensors are a popular combination. [8–13]. The inertial sensor determines the hand location, while the EMG sensors offer additional information to comprehend complex finger or hand gestures fully. Instead of EMG sensors, strain gages, tilt, and even vision sensors can be used.

As a result, the amount of sensor data generated by these advanced gesture detection systems increases. Machine learning is being used to deal with the increasing data. Sensors are introduced to a variety of machine learning approaches. A sensor device processes a linear discriminant analysis or a support vector machine classifier. [9, 14]. In another study, a feedforward neural network (FNN) is used for digitizing, coding, and interpreting signals from a MEMS accelerometer [15].

In the meantime, inertial sensor-only techniques have been developed. This inertial-sensor-only technique may improve portability and mobility while minimizing processing needs in cases involving numerous sensors or complicated algorithms. The handwriting was rebuilt using the phone's gyroscope and accelerometer after users used a smartphone as a pen to write words. [16]. English and Chinese characters, as well as emojis, were written in handwriting. Kinematics based on inertial sensor inputs were employed in other studies to track the movement of hands and arms. [17–19]. Recognizing head or foot motions has also been described [20, 21], but they have not been modified for hand gesture identification.

Inertial sensors are unquestionably accurate and fast as HMI input devices. However, these two objectives are incompatible because increased accuracy typically increases computing load, resulting in sluggish speed. Furthermore, user movements should be uncomplicated. Inertial-sensor-based gesture recognition systems, yet again, have substantial drawbacks. One limitation is the accumulation of inertial sensor noise, which creates bias or drift in the system output [22]. The second disadvantage is that MEMS gyroscope and accelerometer signals can be jumbled [23].

From simple constructions (such as moving average filters) to the recently created outcomes, signal processing of inertial sensor outputs has been intensively researched to overcome these challenges (such as machine learning). Two recent approaches are digitizing sensor data to form codes and generating statistical measurements of the signs to describe their patterns. One method identified seven hand motions using a three-axis MEMS accelerometer. [24]. Hopfield network labels positive and negative symptoms on accelerometer signals, digitized, and restored.

These accelerometer-only systems are good at capturing linear gestures (such as up/down or left/proper patterns) but not so good at capturing circular motions (e.g., clockwise rotation or hand waving). Recognizing linear and rotational gestures has been suggested using accelerometers and gyroscopes. Using accelerometer and gyroscope sensors mounted on the forearms, the researchers used the Markov chain method to track the movement of the arms. [25]. Continuous hand gestures (CHGs), a real-time gesture identification method, were disclosed in another recent work paper. [26]. The approach begins by defining six basic gestures, determining their statistical measurements, such as means and standard deviations (STDs), then generating a database for each motion's actions.

These accelerometer-gyroscope combos are highly accurate, but they are neither portable nor inexpensive instruments. If we want to reduce the system's size and use and give numerous functions with a limited amount of hand motions, we need to find a solution. This research aimed to create a small gesture detection device and a modal HMI input device that could respond accurately and quickly to the user's intention to solve these issues.

The accelerometer, gyroscope, accelerometer-gyroscope fusion, ultrasonic, and combination accelerometer-gyroscope with electromyography approaches are used in reporting recent activities using sensor-based gestures as the HMI input. Rotational motions cannot be detected with merely an accelerometer. As a result, this paper opts for an accelerometer-gyroscope fusion system in the hopes of superior rotation sensing (than the accelerometer-only systems). We believe that the originality of this project is critical for portable HMI input devices, and it is demonstrated using an Arduino Nano 33 BLE board, a very light embedded device. It is one of the most suitable embedded devices for the project, with a weight of 5 g, a length of 45 mm, and a width of 18 mm. Although the Arduino Nano is one of the most portable and lightweight, it also poses a challenge, i.e., memory constraints. Due to its small size, it has only 1 MB of flash memory and 256 KB of static RAM, making working on it very difficult.

2 Design of the Gesture Recognition System

Our proposed system is set up to implement several essential features. First, we use a collection of simple hand motions, each with predefined functions for different application applications. Our system will be aware of the program that is now running. As a result, the procedure carried out by each motion can vary depending on the application, allowing for multifunction capabilities while reducing gesture complexity, resulting in a highly diverse HMI input device.

The second feature is that the complete hardware used is an Arduino Nano 33 BLE which is lightweight and quickly fitted on a stick. The Arduino Nano 33 BLE consists of an inbuilt IMU with one miniature three-axis gyroscope and one miniature three-axis accelerometer. The fusion of these two sensors provides us with a large amount of data about the object's acceleration and rotational motion, i.e., the Arduino Nano 33 BLE.

The third feature is hand gesture recognition in real time. To lessen the delay caused by computing load, we will train our model on a machine with a lot of processing capacity and then lower the model's size using quantization, which reduces the model's height to the extent that we can handle it.

The last feature is system accuracy. Even though the complexity grows and numerous sensors are employed as input sensors to produce a single gesture, sufficient accuracy should be ensured. We strive to apply a pre-processing approach of rasterization that turns the data from the accelerometer and gyroscope into a rasterized image. We train our model, which gives us a very high accuracy, to eliminate errors caused by hand tremors or inadvertent hand gestures.

Our input device can be used in a variety of ways. This technology can benefit input devices such as computers, laptops, portable multimedia players, wireless remote controllers for presentation applications, and virtual reality modules. For example, a user might connect the input device to a laptop and give a presentation to an audience. Pause, play, or turn up the volume if he wants to view a video.

Even if we want to interact with the computer in such situations, many input devices, such as a keyboard or mouse, may be required. However, all these can be replaced by a single input device, which is portable, accurate, and our approach's primary target.

An overview of the system design is shown in Fig. 1. The IMU containing accelerometer and gyroscope generates acceleration and angular velocity data from hand gestures. It feeds it to a process of rasterization, which converts this data feed to a rasterized image which is then given to a CNN machine learning model for training. This pre-processing of rasterization ensures that the model is free of sensor noise, limitations, or unwanted gestures. In addition, initially, while training, the machine "learns" the preferences and habits of users. The pattern is fitted according to the user's gestures needs through the data of a single motion multiple times (Fig. 2).

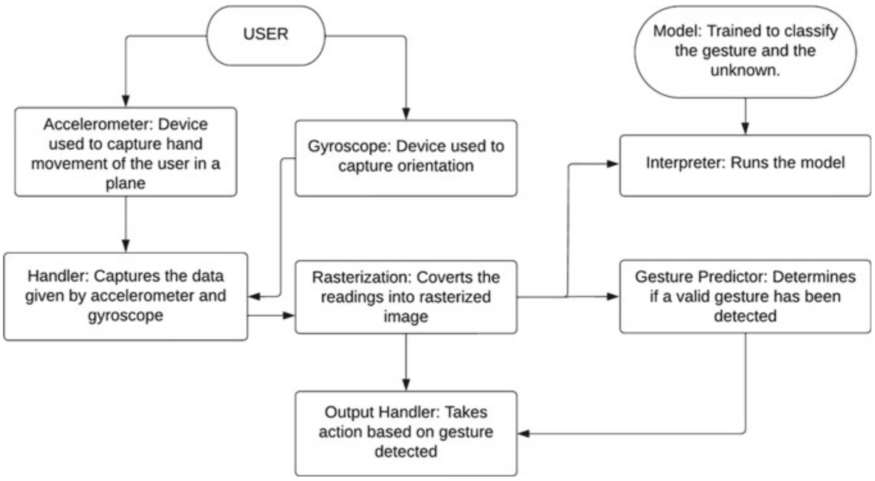
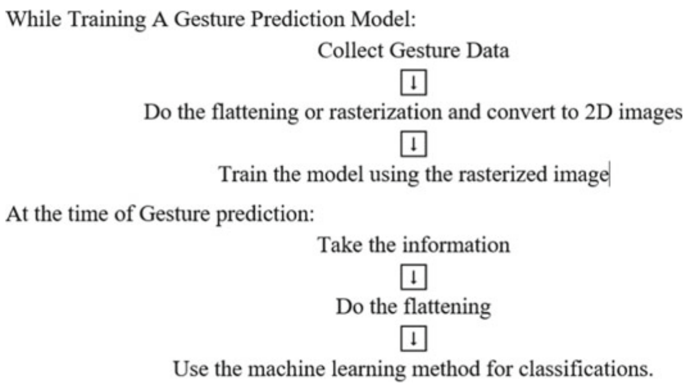

USER
Model: Trained to classify the gesture and the unknown.
Accelerometer: Device used to capture hand movement of the user in a plane
Gyroscope: Device used to capture orientation
Interpreter: Runs the model
Handler: Captures the data given by accelerometer and gyroscope
Rasterization: Coverts the readings into rasterized image
Gesture Predictor: Determines if a valid gesture has been detected
Output Handler: Takes action based on gesture detected

Fig. 1 Flowchart of the system design and data


While Training A Gesture Prediction Model:

Collect Gesture Data

↓

Do the flattening or rasterization and convert to 2D images

↓

Train the model using the rasterized image

At the time of Gesture prediction:

Take the information

↓

Do the flattening

↓

Use the machine learning method for classifications.

Fig. 2 Flowchart when the system is the indifferent process: **a** while training; **b** while predicting gesture

3 Hand Gesture Recognition Algorithm

The gesture recognition application accomplishes a reasonably complicated task by carefully crafting a 2D image from 3D IMU data. The dataflow is as follows:

1. The accelerometer data is read and passed to the EstimateGravityDirection (), which is used to determine the orientation of the Arduino concerning the ground.
2. The accelerometer data is passed to UpdateVelocity (), which is used to calculate the velocity of the Arduino.
3. The direction of gravity is passed to UpdateVelocity () and is used to cancel out the acceleration due to gravity from the accelerometer data.

4. The velocity is then passed to EstimateGyroscopeDrift (), determining if the Arduino is stationary or moving.
5. The gyroscope data is passed to EstimateGyroscopeDrift (), which calculates the gyroscope's sensor drift if the Arduino is not moving (velocity is 0).
6. The gyroscope data is passed to UpdateOrientation (), where it is integrated to determine the angular orientation of the Arduino.
7. The gyroscope drift is also passed to UpdateOrientation () and subtracted from the gyroscope reading to cancel the essence.
8. The 3D angular orientation is passed to UpdateStroke () and transformed into 2D positional coordinates. UpdateStroke () also handles whether the current gesture has ended, or a new motion has been started by analyzing the length of the gesture and testing whether the orientation data is still changing.
9. The direction of gravity is also passed to UpdateStroke () to determine the roll orientation of the Arduino.
10. The 2D positional coordinates are passed to RasterizeStroke (), which takes the 2D coordinates and draws lines between them on a 2D image. The color of the lines shifts from red to green to blue to indicate the direction of motion during the gesture.
11. The 2D image of the gesture is converted to ASCII art and printed on the serial monitor.
12. The 2D image of the gesture is passed to the model.
13. The model predicts the label of the gesture, and the title is printed on the serial monitor. Figure 3 depicts the above algorithm workflow.

4 Implementation

There are two primary components: the initialization phase and the main loop (Fig. 4).

4.1 Initialization

The initialization phase's job is primarily to set up the IMU, and all the resources needed to run the TensorFlow lite macro-model (Fig. 5).

The first step of the initialization phase is the IMU initialization, which is done using this **setup IMU** routine. When you go into the setup IMU routine, you will find device-specific calls that tap into the IMU functions that the library provides (Fig. 6).

The second component is setting up all the resources needed to run the model. This might be the model's pointer, the interpreter's initialization using the Tensor arena, the model, the observer, etc.

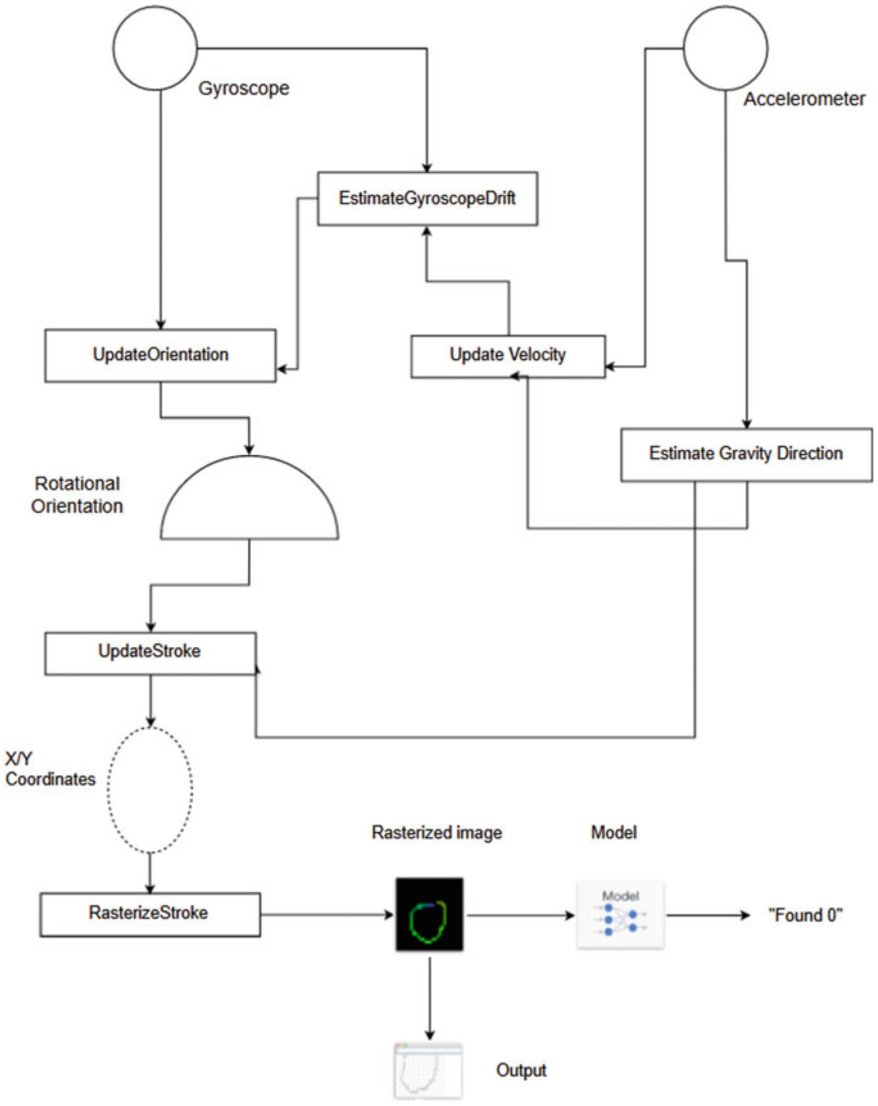


Fig. 3 Hand gesture recognition algorithm workflow

4.2 IMU Provider—Pre-processing

Its job is to get data from the gyroscope and the accelerometer and then process it. Function calls readily available will allow us to read the data from the gyroscope and the accelerometer. So, if data is available from the IMU, we will process that data (Fig. 7).

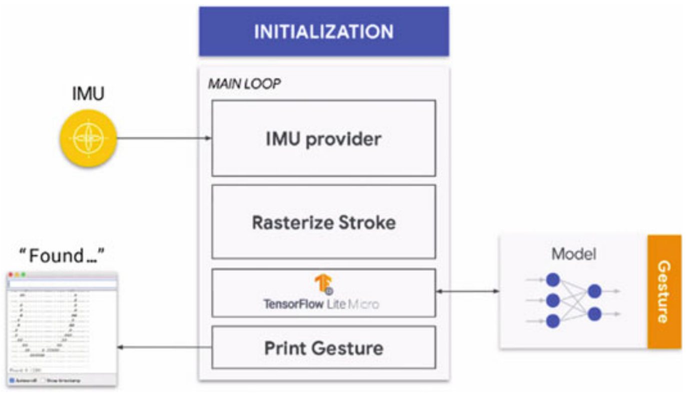


Fig. 4 Implementation of Arduino application

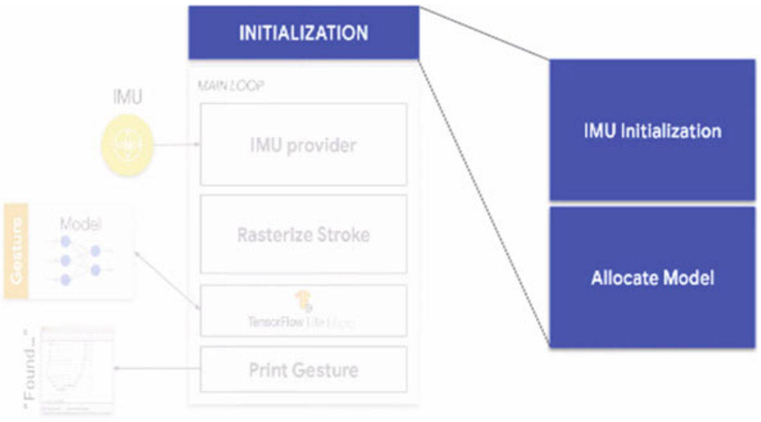


Fig. 5 Initialization of IMU

The gyroscope ends up having a little bit of drift, so we must compensate for that drift. And that is what this function **estimating gyroscope drift** is doing. When we determine that the IMU is not moving using the accelerometer, we can calculate the gyroscope's importance and then account for it. Next, we want to integrate the gyroscope's incremental angular changes that are coming in overtime because that will give us a part of the gesture in the spherical coordinate system, and that is how this function **updates orientation**. It is trying to capture that part that is coming in continuously. Next, we effectively want to project it into a two-dimensional plane inside this physical system. Well, that is because it is much easier to understand a 2D gesture than a complex 3D gesture. And therefore, **update stroke** is going to do that flat mapping.

Then comes the process of processing the accelerometer data. We want to estimate the gravity's direction to control the sensor's role in the gyroscope readings. Everyone

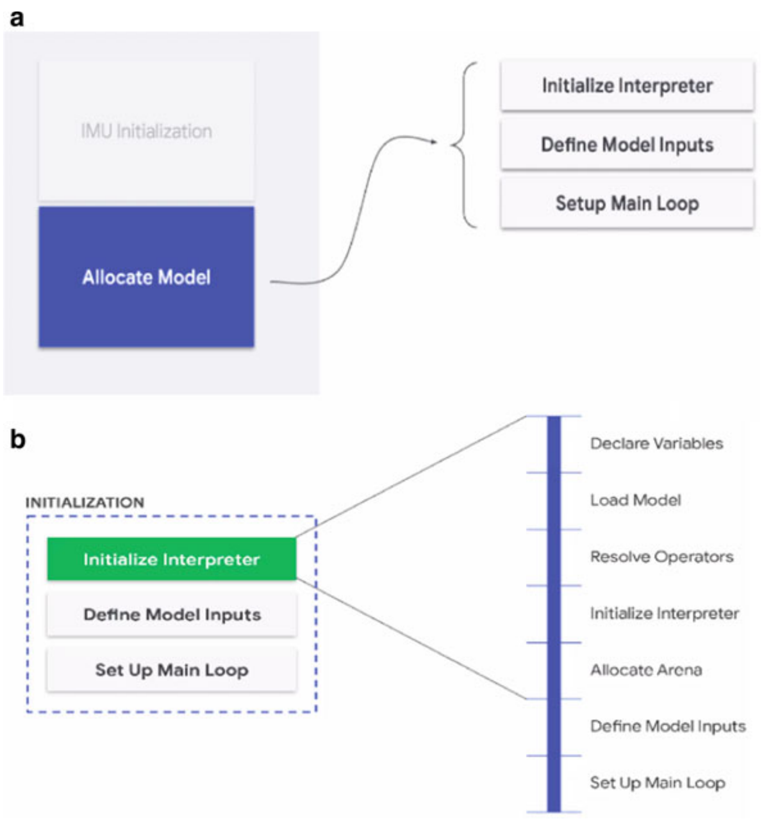


Fig. 6 The model initialization. **a** Model variable and space allocation, **b** model interpreter variable allocation

is going to be holding the stick at a specific angular momentum. So, this means that you must neutralize or normalize for that effect. For example, you might have the bar with your right hand or hold the post with your left hand. However, the gesture that you are performing is the same thing. Either way, we have got the same number written, so we get to compensate for that. And the way we do that is by effectively trying to figure out the role of the gyroscopes reading. And then, we update the velocity to know when the sensor is still, and we can correct the sensor drift.

4.3 *Rasterize Stroke—Pre-processing*

After that, the step of effectively capturing the data is to rasterize that stroke. We pre-process this because it is easy to feed an image into a convolutional neural network. And there is a function that helps us do that: rasterized stroke (Figs. 8 and 9).

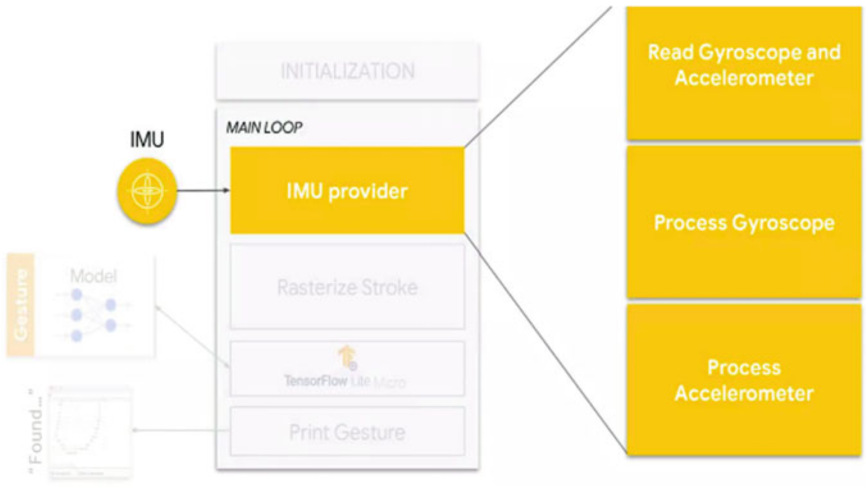


Fig. 7 Read the data of the gyroscope and accelerometer and estimate the drift of the gyroscope

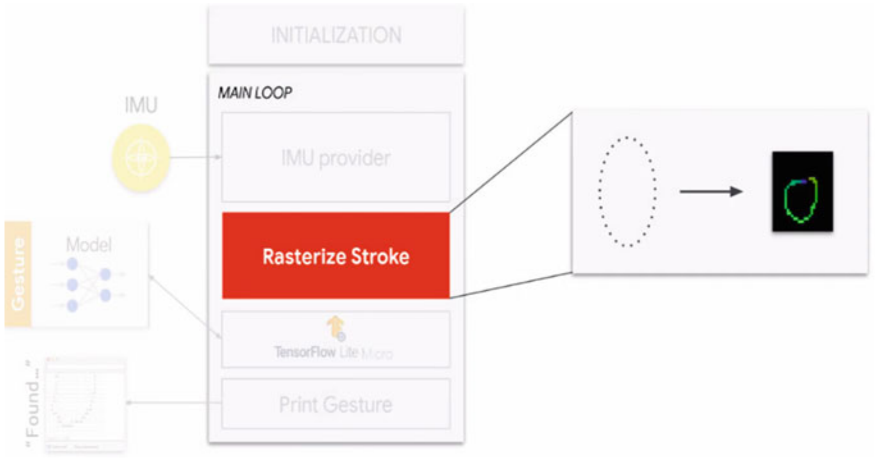


Fig. 8 Flatten the three-dimensional coordinates to two-dimensional coordinates and then rasterize that into an image

4.4 Model

After pre-processing, the next part is to hand that rasterized image directly to our convolutional neural net. In this case, we will pass in an RGB image, a red, green, and blue image. So, there are three challenges to the idea that we are giving into the net. And that input will then be run using a convolutional neural network, predicting the gesture. To invoke the model, we must set up the input buffers. Also, due to having

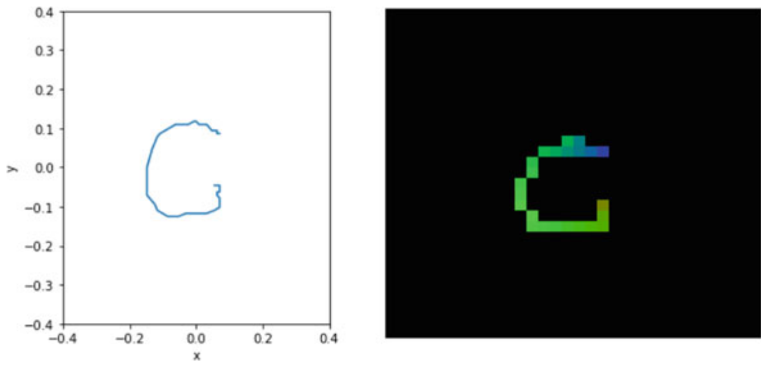


Fig. 9 Rasterization process: flatten the three-dimensional coordinates to two-dimensional and convert an image into an RGB image for building a CNN model for classification

Table 1 Reduction in size of the model to fit our need for Arduino Nano 33 BLE

Model	Size	
TensorFlow	683,299 bytes	
TensorFlow lite	98,812 bytes	(Reduced by 584,487 bytes)
TensorFlow lite quantized	30,576 bytes	(Reduced by 68,236 bytes)

memory constraints, we must quantize our model. Table 1 shows how quantization reduces the size of the model (Figs. 10, 11 and 12).

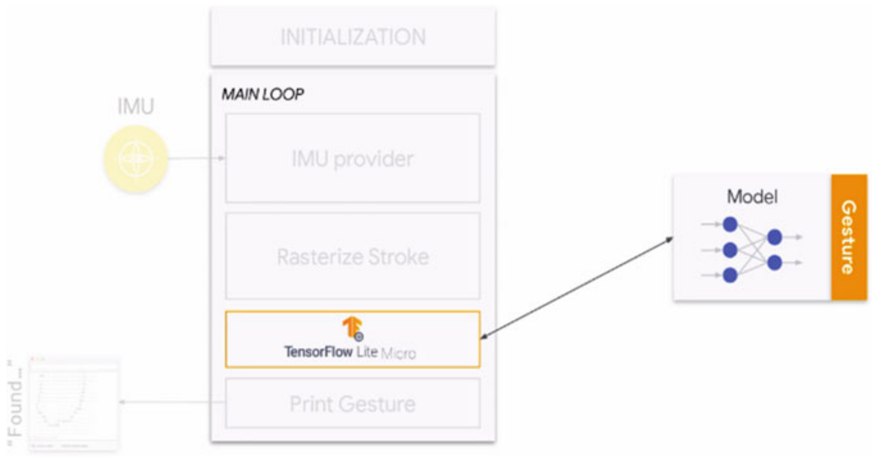


Fig. 10 Calling the TensorFlow lite micro-model for learning and classification

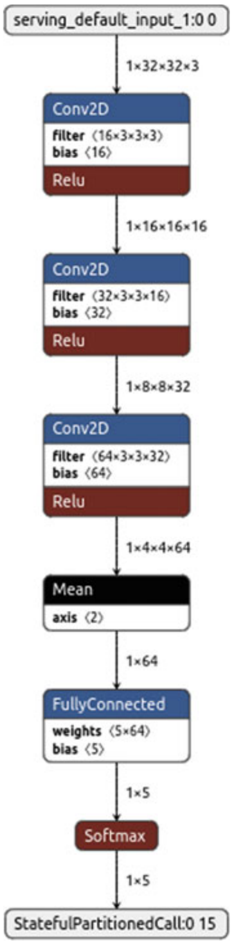


Fig. 11 Model flow

4.5 Output

We get the output from the neural network to see what it has determined as an actual gesture, and in terms of processing the result, we print the work to the screen (Fig. 13).

5 Hand Gestures Recognized

The input device for HMI must perform a wide range of operations, yet it can only recognize a limited amount of hand motions. The five gestures are depicted in Figs. 14, 15, 16, 17 and 18. We move our IMU consisting of an accelerometer and gyroscope in three-dimensional space.

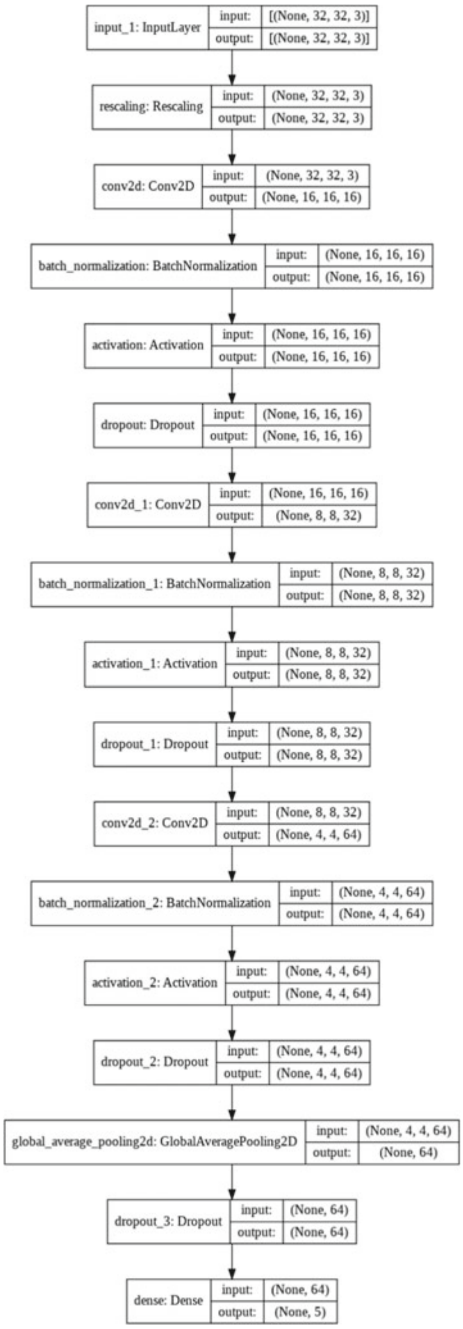


Fig. 12 Complex machine learning model workflow. It uses the CNN machine learning model for classification

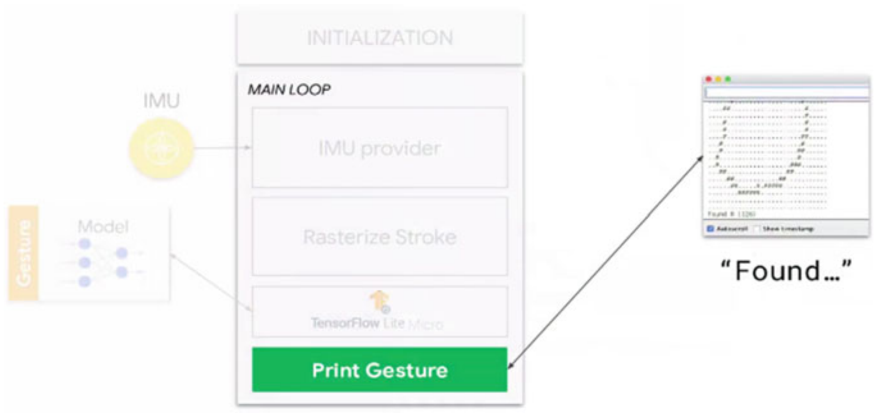


Fig. 13 Output of the gesture recognition

Fig. 14 C Alphabet

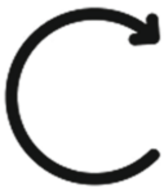


Fig. 15 L Alphabet

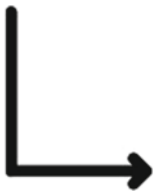


Fig. 16 I Alphabet

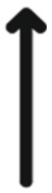
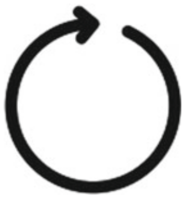


Fig. 17 O Alphabet



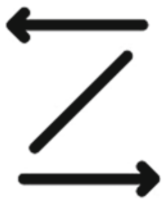


Fig. 18 Z Alphabet

6 Experimental Demonstration Gesture Recognition Device

Several critical approaches were previously discussed, such as rasterization, the model utilized for high-accuracy classification. These strategies created a varied HMI input device with easy, real-time, accurate, user-friendly, and multi-functional capabilities. These benefits were demonstrated in follow-up investigations.

Figure 19 depicts our experimental setup. The sensor system consisted of a microcontroller unit, inertial sensor IMU, and an inertial sensor system. An accelerometer, a gyroscope, and a magnetometer were all included in the inertial sensor, but only the three-axis accelerometer and three-axis gyroscope were used in this investigation. The sampling frequency was set to 25 Hz. The sensor system is constructed by mounting the microcontroller on a stick and interacting via USB. Using the BLE (Bluetooth) module, we may increase the usability.

6.1 Verification

The constructed input device is used on multiple application programs controlled by the input device to test the suggested concept—the program aimed to transition between various programs and a media player for playing video and media files.

Each experiment was carried out in a particular order. First, we tested the connectivity and operation of the input and gesture recognition device. We execute the target program and assess the functions once the device is connected correctly. The five hand gestures in Figs. 15–19 and if necessary, simple variations of five movements match the activities. After the initial setup, the first volunteer in the experiment activated the device and performed a scenario involving a series of hand gestures that fulfilled all the fundamental operations.

6.2 Verification of Media Player

The opening of the media player is the initial feature. The current playing file can be played and paused with the second function. The third option is to mute the movie

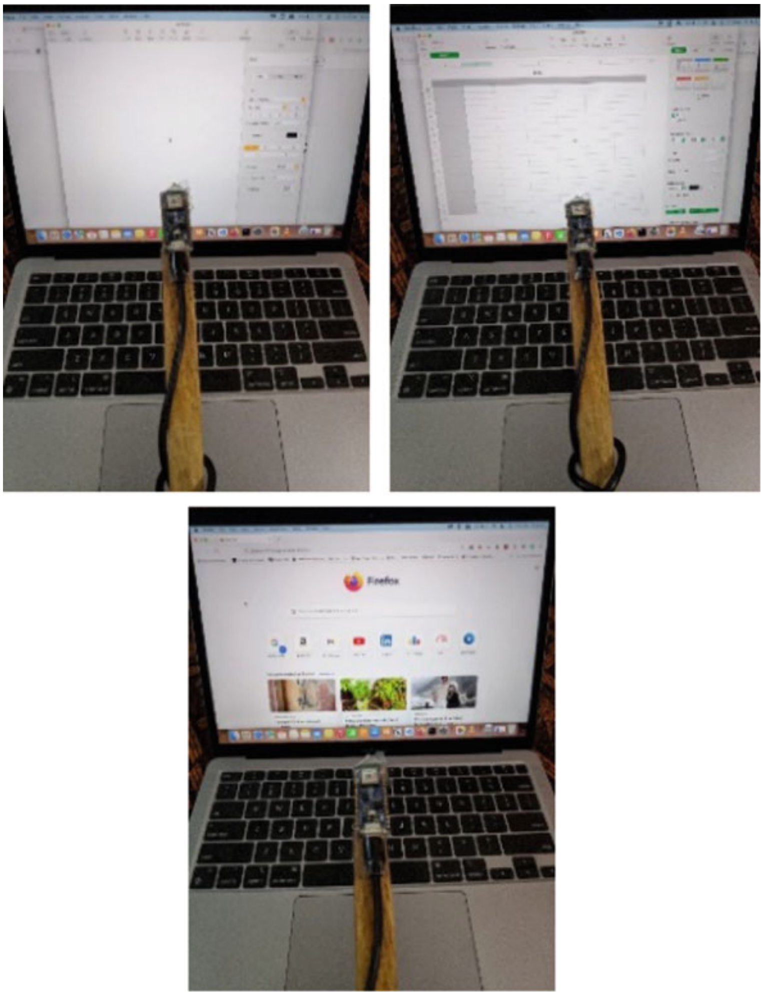


Fig. 19 Input device is used for the application program

and increase or decrease the volume. Figure 20 shows an experimental video file playback sequence. The following are the play, pause, and volume controls.

7 Conclusion

This paper proposes a sensor-based gesture recognition system that can be used as an input device for the HMI system. Five gestures are being used for multiple different applications. The same device behaves differently at some point for other

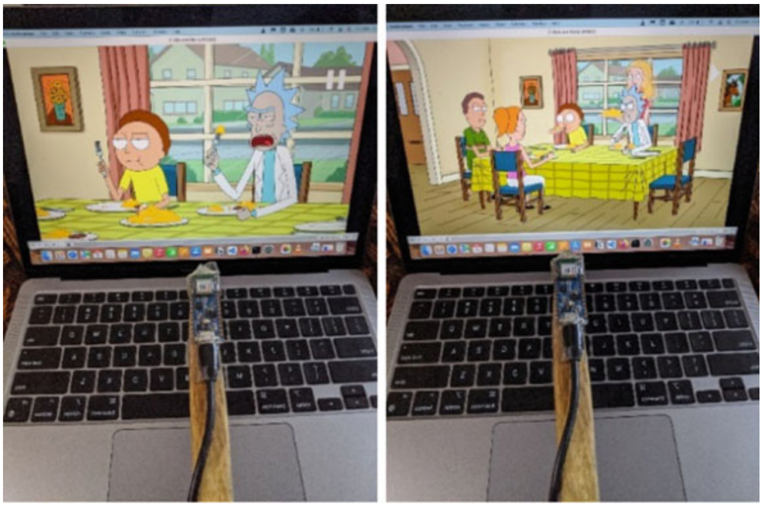


Fig. 20 Sequence of experiments uses gestures as input to a multimedia video player and plays and pauses the video

types of running on the device. If used to open an application for another application, it could mute the device, pause it, or play it. The project's primary emphasis is on the project's portability and fast and reliable recognition. For portability, we have used Arduino Nano 33 BLE. We use the rasterization process for fast and reliable recognition, which converts the three-dimensional spherical coordinates into two-dimensional coordinates and then rasterizes the image. This image is easy to build a highly accurate and robust model, which ensures our gesture recognition is perfect.

References

1. Pavlovic VI, Sharma R, Huang TS (1997) Visual interpretation of hand gestures for human-computer interaction: a review. IEEE Trans Pattern Anal Mach Intell 19:677–695
2. Zhang Z (2012) Microsoft Kinect Sensor and Its Effect. IEEE Multimed 19:4–10
3. Cavalieri L, Mengoni M, Ceccacci S, Germani MA (2016) Methodology to introduce gesture-based interaction into existing consumer product. In: Proceedings of the international conference on human-computer interaction, Toronto, ON, Canada, 17–22 July 2016; pp 25–36
4. Yang X, Sun X, Zhou D, Li Y, Liu H (2018) Towards wearable A-mode ultrasound sensing for real-time finger motion recognition. IEEE Trans Neural Syst Rehabil Eng 26:1199–1208
5. King K, Yoon SW, Perkins N, Najafi K (2008) Wireless MEMS inertial sensor system for golf swing dynamics. Sens Actuators A Phys 141:619–630
6. Luo X, Wu X, Chen L, Zhao Y, Zhang L, Li G, Hou W (2019) Synergistic myoelectrical activities of forearm muscles improving robust recognition of multi-fingered gestures. Sensors 19:610
7. Lee BG, Lee SM (2018) Smart wearable hand device for sign language interpretation system with sensors fusion. IEEE Sens J 18:1224–1232

8. Liu X, Sacks J, Zhang M, Richardson AG, Lucas TH, Van der Spiegel J (2017) The virtual trackpad: an electromyography-based, wireless, real-time, low-power, embedded hand-gesture-recognition system using an event-driven artificial neural network. IEEE Trans Circ Syst II Exp Briefs 64:1257–1261
9. Jiang S, Lv B, Guo W, Zhang C, Wang H, Sheng X, Shull PB (2018) Feasibility of wrist-worn, real-time hand, and surface gesture recognition via sEMG and IMU sensing. IEEE Trans Ind Inform 14:3376–3385
10. Pomboza-Junez G, Holgado-Terraza JA, Medina-Medina N (2019) Toward the gestural interface: a comparative analysis between touch user interfaces versus gesture-based user interfaces on mobile devices. Univers Access Inf Soc 18:107–126
11. Lopes J, Simão M, Mendes N, Safeea M, Afonso J, Neto P (2019) Hand/arm gesture segmentation by motion using IMU and EMG sensing. Procedia Manuf 11:107–113; Sensors 19:2562
12. Kartsch V, Benatti S, Mancini M, Magno M, Benini L (2018) Smart wearable wristband for EMG based gesture recognition powered by solar energy harvester. In: Proceedings of the 2018 IEEE international symposium on circuits and systems (ISCAS), Florence, Italy, 27–30 May 2018, pp 1–5
13. Kundu AS, Mazumder O, Lenka PK, Bhaumik S (2017) Hand gesture recognition based omnidirectional wheelchair control using IMU and EMG sensors. J Intell Robot Syst 91:1–13
14. Tavakoli M, Benussi C, Lopes PA, Osorio LB, de Almeida AT (2018) Robust hand gesture recognition with a double channel surface EMG wearable armband and SVM classifier. Biomed Signal Process Control 46:121–130
15. Xie R, Cao J (2016) Accelerometer-based hand gesture recognition by neural network and similarity matching. IEEE Sens J 16:4537–4545
16. Deselaers T, Keysers D, Hosang J, Rowley HA (2015) GyroPen: gyroscopes for pen-input with mobile phones. IEEE Trans Hum-Mach Syst 45:263–271
17. Abbasi-Kesbi R, Nikfarjam A (2018) A miniature sensor system for precise hand position monitoring. IEEE Sens J 18:2577–2584
18. Wu Y, Chen K, Fu C (2016) Natural gesture modeling and recognition approach based on joint movements and arm orientations. IEEE Sens J 16:7753–7761
19. Kortier HG, Sluiter VI, Roetenberg D, Veltink PH (2014) Assessment of hand kinematics using inertial and magnetic sensors. J Neuroeng Rehabil 11:70
20. Jackowski A, Gebhard M, Thietje R (2018) Head motion, and head gesture-based robot control: a usability study. IEEE Trans Neural Syst Rehabil Eng 26:161–170
21. Zhou Q, Zhang H, Lari Z, Liu Z, El-Sheimy N (2016) Design, and implementation of foot-mounted inertial sensor-based wearable electronic device for game play application. Sensors 16:1752
22. Yazdi N, Ayazi F, Najafi K (1998) Micromachined inertial sensors. Proc IEEE 86:1640–1659
23. Yoon SW, Lee S, Najafi K (2012) vibration-induced errors in MEMS tuning fork gyroscopes. Sens Actuators A Phys 180:32–44
24. Xu R, Zhou S, Li WJ (2012) MEMS accelerometer based nonspecific-user hand gesture recognition. IEEE Sens J 12:1166–1173
25. Arsenault D, Whitehead AD (2015) Gesture recognition using Markov Systems and wearable wireless inertial sensors. IEEE Trans Consum Electron 61:429–437
26. Gupta HP, Chudgar HS, Mukherjee S, Dutta T, Sharma K (2016) A continuous hand gestures recognition technique for human-machine interaction using accelerometer and gyroscope sensors. IEEE Sens J 16:6425–6432