GESTURE RECOGNITION

Ritik Gupta, Rishabh Chauhan, Rishav Chaba, Ritish Varshney, Praveen Saini
Department of Computer Science and Engineering
Moradabad Institute of Technology, Moradabad

Abstract:

Gesture Recognition is a type of perceptual computing user interface that allows computers to capture and interpret human gestures as commands. The general definition of gesture recognition is the ability of a computer to understand gestures and execute commands based on those gestures. [1] The main objective is to develop a Machine Learning based system that uses gestures to perform various functions. For this project, we are aiming to build a system which can be trained to recognize the gestures we make and perform the dedicated function we decide for it. Gestures, to the system, will be taken from Arduino with Accelerometer through micro USB cable.

Keyword*:* Gesture Recognition, Machine Learning, Arduino, Accelerometer, micro USB cable, gestures, commands, interpret, system.

I. Introduction

Gesture recognition, Arduino based, from accelerometer data is an emerging technique for gesture-based interaction, which suits well the requirements in ubiquitous computing environments. With the rapid development of the MEMS (Micro Electrical Mechanical System) technology, people can wear/carry one or more accelerometer-equipped devices in daily life. For this project, we aim to build a system which can be trained to recognize the gestures we make and perform the dedicated function we decide for it. We'll be demonstrating this by training the system to recognize letters by the gestures we make in the air. To build this system, we'll be using an Arduino Board interfaced with an accelerometer. The device can be attached to the user's hand. The accelerometer will provide input to the microcontroller about the hand's (which is being used to make the gesture) coordinates. The algorithm will pick up this data and maintain a database to recognize each gesture differently. Once we train the system with the same gesture multiple times it will gather enough data to have an estimate of what the gesture should look like. This gesture can then be assigned to perform a task on the computer.

The accelerometer can collect hand orientation and acceleration, 3-axis acceleration of users' hand motion. Now most accelerometers can capture three-axis acceleration data, i.e. 3D accelerometers, which convey more motion information than 2D accelerometers. The accelerometer is embedded into Arduino. Here, we use Arduino as an input device for experimental set-up and performance evaluation. To recognize a gesture from the captured data, we have applied many machine learning algorithms. The most accurate and efficient algorithm that we found is Support Vector Machine (SVM) to classify the data. Based on the SVM algorithm, new gestures can be classified and the corresponding symbol can be recognized. The classification algorithm SVM is applied using a python script in the project using Arduino's serial monitor. Then, the symbol, which can be uppercase English letter or lower-case English letter or numeric, can be typed at the current focusing mouse pointer in the editor. Other operations can be performed like mouse operations, media operations, browser operations etc. also but this is done by only Arduino Remote itself. [2]

II. Methodology

STEP 1: Connect the accelerometer

In this step, we are connecting the accelerometer with Arduino. To do this, first solder male header pins onto the breakout board, if you haven't done so already. Then you'll wire up the accelerometer to the Arduino. As a shortcut, you can plug the accelerometer breakout directly into the analog input pins of the Arduino (or other Arduino with the same form factor). Then in the Arduino code, you can configure the appropriate pins to provide power and ground to the accelerometer. Alternatively, you can plug your accelerometer into a breadboard and wire it to the Arduino, connecting its power and ground pins to the 5V and GND pins of the Arduino, and its X-, Y-, and Z-axis pins to three analog inputs of the Arduino board.

STEP 2: Upload the Arduino code

An Arduino program is created to read data from the accelerometer and send it over serial (USB) to the computer. First, check that the pins specified in the Arduino program match the way you've wired up your accelerometer (e.g. that xpin corresponds to the analog input pin that's connected to the X-axis pin of your accelerometer). Then select the appropriate board and serial port from the Arduino tools menu and upload the Arduino sketch.

STEP 3: Check for data in the serial monitor

Open the Arduino serial monitor, set it to 38400 baud, and check that we're getting the appropriate accelerometer data from your Arduino or not. On the serial monitor, the data coming from the accelerometer to the Arduino can be seen like that:
START -2556 4296 22696 696 666 -1446 END
START -2904 2448 19832 2687 12345 -4298 END
START -1656 -1520 8952 -9006 2312 -4080 END
START -3552 -464 10844 8037 -6754 -529 END
START -3200 1044 29152 744 -9458 -3292 END
START -856 -4464 24000 -4321 7144 3458 END
First 3 columns are Acx, Acy, Acz, all axis data of the accelerometer for speed. And the next 3 columns are Gx, Gy, Gz, all axis data of gyroscope for orientation. On moving the arduino, the data is changing rapidly which shows that the connection of the accelerometer with the Arduino has been done correctly. Be sure to close the serial monitor before continuing, as otherwise they'll

block the python application from talking to your Arduino.

STEP 4: Collecting the sample data for training SVM

To collect the sample data, we use a button, connected with Arduino, for capturing the data from accelerometer to python script. On click and hold the button, the sample data starts collecting from the accelerometer and saved in the specific folder using python script. We use classes for collecting sample data for training SVM, because SVM needs classes for each training data, and we use 4-5 classes and for each class we record 10-15 sample data. The recorded sample data is saved under a .txt file, named as:
a_sample_b_c.txt, where
a = symbol,
b = class no
c = sample data no for that class
Be sure to record the example gestures with the accelerometer in the same configuration as it will be later, when you want the system to recognize the gestures. For instance, you might hold the accelerometer in your hand with a particular orientation, or attach it to an object that you'll hold with a particular orientation.
A good sample contains the data corresponding to the whole gesture, but without much additional baseline data at either the start or the end. That is, the sample should start and end with a short period of relatively flat lines, neither too long nor missing altogether. Each additional example you record is another sample that the machine learning algorithm can match against when it's recognizing gestures. That means that if you want the system to recognize different variations of a gesture (e.g. the different ways in which it is made by different people), it may help to record samples of each variation. On the other hand, if you have bad samples, they may confuse the system; more samples aren't necessarily better. In general, we've had good luck recording somewhere around 5 to 10 samples for each gesture, although again, the quality of the individual samples is more important than their quantity. If you don't like a sample (e.g. because you pressed the key at the wrong time and missed the data corresponding to part of the gesture), you can delete it only by going to the appropriate file location in the computer system and delete the file manually. So, try to record the sample data carefully. If you recorded a sample in the wrong class, you can re-label the file name.

STEP 5: Train Machine learning model

Once you've recorded a few example gestures, you can train the machine learning model, Support vector machine, to recognize those gestures from your examples. The training can be done by executing the python script manually in the background. The message "training successful" appears at the bottom of the console window in which the python script is executing. The console window also shows the accuracy score (in percentage) of the SVM model.
The system may not work well the first time you train it. It's helpful to train and test the system often as you record your example gestures, so you can get a sense of how it's behaving. In particular, if the system isn't recognizing gestures you think it should, you may want to record additional examples of that gesture. If the system is recognizing gestures when it shouldn't, you may want to delete or trim examples that look different than the others.

STEP 6: Making Prediction

Once the Machine learning model, SVM, is trained successfully, then it's time to recognize the new gesture symbol with the help of trained SVM model. For this, we have created a simple website which acts as a GUI for many operations performed by the project. After selecting the appropriate mode from the website, the mode is started to predict the new gesture symbol. To predict, the button embedded with the Arduino is clicked and held till the gesture recording by the Arduino. After leaving the button, the python script collects the new gesture data and then SVM model takes that data for the prediction. After successful prediction done by SVM, the corresponding symbol for the gesture is displayed at the current mouse cursor. [3] [4]

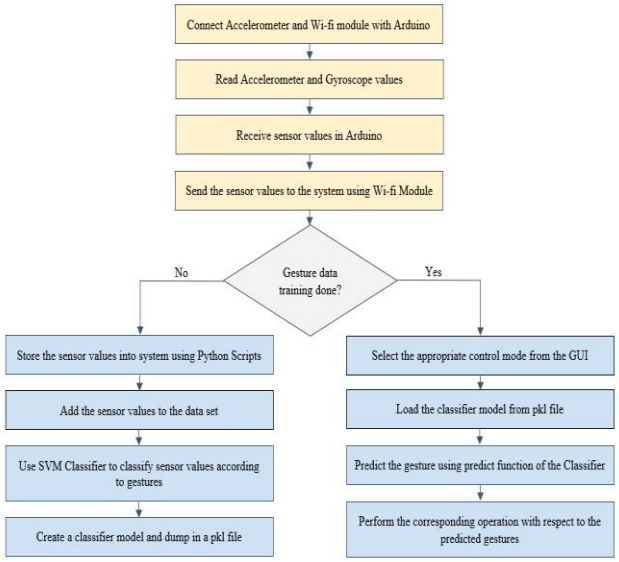

Fig 1: Flow Chart of Methodology

III. MODULES

There are some Modules which can be performed with the help of this proposed system:

1. English Typing A-Z
2. English Typing a-z
3. Numeric Typing 0-9
4. Mouse Operation
5. Media Controlling
6. Web Browser Controlling
7. Presentation Controlling
8. Window Task Switching

IV. **Support Vector Machine (SVM)**

Support Vector Machine (SVM) is a supervised machine learning algorithm capable of performing

classification, regression and even outlier detection. The linear SVM classifier works by drawing a straight line between two classes. All the data points that fall on one side of the line will be labelled as one class and all the points that fall on the other side will be labelled as the second. Sounds simple enough, but there's an infinite amount of lines to choose from. How do we know which line will do the best job of classifying the data? This is where the LSVM algorithm comes into play. The LSVM algorithm will select a line that not only separates the two classes but stays as far away from the closest samples as possible. In fact, the "support vector" in "support vector machine" refers to two position vectors drawn from the origin to the points which dictate the decision boundary.

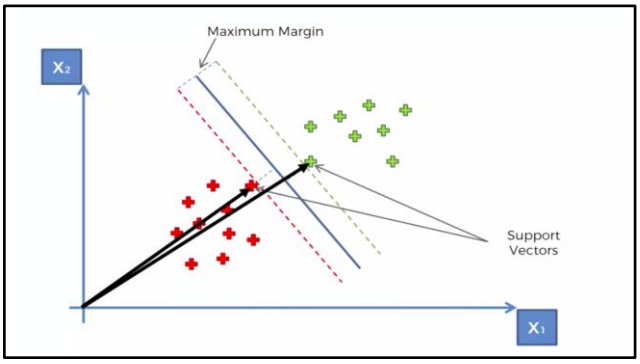

Fig 2: Working of Support Vector Machine (SVM)

In this, the dataset is split into training and testing set, the training set is used to train the system (i.e. learn the system) to recognize different patterns of categories, the testing set is used to evaluate the system, the process of categorization depends on the algorithm used. The choice of which specific learning algorithm we should use is a critical step. Once we complete the preliminary testing and if achieved as to be satisfactory, the classifier that maps the unlabelled instances into classes is available for the routine to use. The classifier's evaluation is most often based on prediction accuracy (the percentage of correct prediction divided by the total number of predictions). There are three techniques used to calculate a classifier's accuracy which is essential. One technique is to split all the training set by using two-thirds of it for training and the other third of it for estimating their performance. In cross-validation, the training set is divided into mutually exclusive, equal- sized subsets and for each subset the classifier is individually trained on the U of other subsets. [5]

V. Conclusion

This paper portrays a framework that has the ability to identify the character drawn through a gesture in air with the assistance of an input device. The strategy proposed here effectively made an acknowledgment framework, that can perceive which motion is performed by the user and precisely play out the usefulness related to it. Also, we conducted evaluation comparing the performance of machine learning algorithms on classifying gestures. This study uses accelerometer-based datasets as the source of signals and also sci-kit learning. The variables from calculating feature distances we used as machine learning input for the learning process. The result of discriminant analysis of the data shows accuracy of classifying the data, while the machine learning algorithms perform well. The peak accuracy of SVM gets into 90.69%, for KNN it reaches 88.89%. The device is capable of successfully reading gestures in air and accurately performing functions as described throughout.

VI. Future Scope

The present framework gives best outcomes in a plain foundation and henceforth puts certain imperatives on the user for effective working. The future work will incorporate usage of extra signals which will empower the client to perform more capacities easily. Moreover, foundation subtraction calculation can be utilized for a more compelling execution. The proposed framework utilizes just the correct hand to perform signals. Henceforth, upgrade of the procedure proposed, is conceivable utilizing the two hands for performing diverse PC activities. Examinations should be done on a bigger scale with the goal that outcomes can be more exact.

There are two most important future work which can be done to improve this system:

1. ESP 8266 Wi-Fi Module
2. Hybrid Machine Learning Model

1. ESP 8266 Wi-Fi Module:

The Module can be used to transfer the Gesture data from Arduino to the system wirelessly. This improves the system range i.e. User can use this proposed system from the large distance also within the given range of W-Fi module.

2. Hybrid Machine Learning Model:

In this, till now, Only Support Vector Machine (SVM) Machine Learning Model is used to classify the Gesture data with an accuracy % of 91.23. But here, an improvement can be done in selecting Machine learning model. The hybrid Model based on the combination of k-nearest neighbors (kNN) and Support Vector Machine (SVM). This combined Model can give the accuracy of 94% which is 3% more than that of previous model.

REFERENCES

[1] S. Schechter, "What is gesture recognition? Gesture recognition defined," 24 March 2014. [Online]. Available: https://www.marxentlabs.com/what-is-gesture-recognition-defined/#:~:text=Gesture%20recognition%20is%20a%20type,commands%20based%20on%20those%20gestures.. [Accessed 07 June 2020].

[2] wikipedia, "Gesture recognition," 06 June 2020. [Online]. Available:https://en.wikipedia.org/wiki/Gesture_recognition. [Accessed 07 June 2020].

[3] J. Wu, G. Pan, D. Zhang, G. Qi and S. Li, "Gesture Recognition with a 3-D Accelerometer," in Department of Computer Science Zhejiang University, Hangzhou, China, 2009.

[4] Mellis, "Gesture Recognition Using Accelerometer and ESP," 20 May 2016. [Online]. Available:

https://www.hackster.io/mellis/gesture-recognition-using-accelerometer-and-esp-71faa1. [Accessed 07 June 2020].

[5] javaTpoint, "Support Vector Machine Algorithm," JavaTPoint, 16 June 2018. [Online]. Available: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm. [Accessed 07 June 2020].