Volume: 6 Issue: XII Month of publication: December 2018
DOI:

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue XII, Dec 2018- Available at www.ijraset.com

Gesture Controlled Home Automation using Leap Motion

Anurag Sethuram[1], Abhilash Somasamudramath[2], Anupama H S[3], Ganne Sai Gokul[4], Ashwith Atluri[5]
[1, 2, 3, 4, 5]Dept. of Computer Science and Engineering, R.V. College of Engineering, Bangalore – 560059, India

Abstract: The purpose of this project is to develop an end to end home automation system that is designed to respond to specific gestures for turning on or off a set of designated IoT enabled home appliances each controlled with a specific gesture. The Leap Motion controller is one of the primary device around which this project has been designed and this device has the capabilities to model a person's palm in a conical three-dimensional space above the controller. The reason this entire project was conceived was to help those who cannot physically move from place to place to control their home appliances through conventional switch boards and also for those who are mute and hence cannot use the existing voice-controlled home automation devices.

The set of 4 distinct gestures were decided upon to control four different light bulbs. These gestures were the waving hello gesture, hold gesture, swipe or move gesture and the upside-down fist clench or grab gesture and had to be trained using Support Vector Machine (SVM) after which the model that was trained was used in real time gesture recognition to detect the specific gestures. A vast multitude of features were used in training the model which included different relative distances and angles amongst the fingers, the finger joints and so on. In order to demonstrate the working, the Infineon XMC 1100 was used alongside a Particle Photon. The Leap Motion was setup with a laptop, and an AWS instance was used in order to wirelessly connect the Photon with the Leap Motion gesture recognition system. A simple LED strip, which was connected to the XMC 1100 was used to show the application of this system as a whole.

Each gesture was successful in controlling the switching on and off of a device that was assigned to it with an accuracy of 92.3% using SVMs with RBF (Radial Basis Function) kernel when used in real time gesture recognition and an accuracy of 100% when the testing is done while reading the gesture feature data from a file. The reason for the difference of accuracy is because of the movement factors added during real time gesture check and the transition that happens between the gestures

Keywords: Home Automation, Gesture Recognition, Support Vector Machine, RBF Kernel

I. INTRODUCTION

With the advent of Artificial Intelligence, conventional Computer Vision based methods for home automation are being rendered obsolete, and various AI based algorithms are taking over. The primary goal of gesture recognition is to identify specific human gestures [1]. Typical gesture recognition methods use cameras, which capture just the surface data of any subject in question. However, this limits the performance video based gesture recognition systems. The advent of better technology and corporate research has introduced devices such as the MSFT Kinect Sensor and the Leap Motion Controller, which provide three dimensional scene data [2]. In this paper, we will be using the Leap Motion Controller in order to train a gesture recognition model.

As stated before, the Leap Motion Controller, or just Leap Motion in short, is one of the few available devices that provide three dimensional scene depth data. The operating hardware consists of two cameras and three infrared light emitting diodes. The diodes operate at a very specific frequency of 850 nanometers, so that reflected rays can be sensed even in the presence of visible light [3]. The controller has an operating region 150 degrees above the controller, and a depth of 2 feet, or 120 degrees on each side. Its operation is further enhanced by wide angle lenses and inbuilt resolution adjustment algorithms, which make sure that the image data obtained is of high accuracy and precision.

The Leap Motion Controller's software reconstructs a three dimensional representation of the scene in question. A tracking layer tracks information like finger tips and the palm centers. Various filtering techniques are applied for data coherence. Finally, the controller produces a series of frames and encapsulates them in a transport protocol [3]. Through the protocol, the processed image data is available to use on the Leap Motion SDKs.

In order to access the tracking data, the Leap Motion official SDKs are made use of. The SDK is based on C, and has built-in integrations for engines such as Unity and Unreal. The SDK library in meant to create bindings for direct usage in high level languages. The latest builds of the software (V4 software or Orion Generation provide enhanced support on VR platforms.

Now that we have our data ready, the next part of the problem was to select a suitable learning algorithm to train our gesture recognition model. In our project, we chose to use Support Vector Machines for classifying different gesture Classification is the process to be a group of models or function that distinguish class data or concepts that aims to be used as a class prediction model of an object as unknown class. [4]. In order to carry this out, SVMs are an excellent tool. An SVM works by building an optimal hyper-plane, and is proven to have more superiority in generalization, robustness and efficiency [5].

SVM algorithms use a set of mathematical functions that are defined as the kernel. The function of kernel is to take data as input and transform it into the required form. Different SVM algorithms use different types of kernel functions. In our project, we will be using the RBF kernel. The reason for choosing this kernel is because data present in images is rarely linearly separable in nature, and hence, we resort to using a non-linear kernel like the RBF [6].

The training time for an SVM increases with the number of training samples. To remedy this, a local SVM is used to reduce the number of training samples. This reduction improves classification efficiency.

The kernel is basically a mapping function that transforms a given space into some other very high dimensional space. For this reason, it is usually preferred to have classifications involving a large number of features to be performed with SVMs. While actually solving the SVM, we only need to know the inner products of the coordinate vectors. IF we choose a kernel Q and A and B are points in the actual space, then the kernel would map these points as Q(A) and Q(B). We define a function F such that:

$$F(A, B) = < Q(A), Q(B)>$$

This is where the widely used Kernel Trick comes into play. We express F in terms of the original space without ever needing to know what the function F is. This greatly speeds up SVM solving computations [7].

Having obtained a solution for mining the data from our Leap Motion Controller, and also choosing to go with SVMs, the next step was to show a working implementation of the automation system. In order to achieve this, an LED was taken and paired with an Infineon XMC 1100 microcontroller. In order to toggle it, the microcontroller was also paired with a Particle Photon module. Using Amazon IoT, a connection was established between the Photon and a laptop. Finally, the Leap Motion controller was connected to the laptop. A C++ program was used to train the SVM, by making use of the Leap Orion API calls, and was also used to establish a wireless (cloud-based) connection with the Particle. Gesture pairs were used to switch on and off the LED strip.

In order to show the successful classification, LEDs on the strip lit up in combinations. Each combination was mapped to a certain gesture. During real time gesture recognition, a particular combination would light up only if the triggering gesture mapped to that combination was identified amongst the classified gestures.

II. METHODOLOGY

This chapter describes the methodology adopted for the construction of the feature space, training and testing the model.

A. Construction Of Feature Space And Training

The leap motion controller returns a confidence value for the accuracy with which is assumes that the hand object has been constructed. For this project, a confidence value of > 0.2 is criteria for using hand object which seems to give the right balance for this use case. The hand object and finger object returned by the controller are used to construct the feature space primarily.

For each fingertip, the x ,y, z coordinates are used to calculate distance from the center of the palm (x, y, z) coordinates considering relative distance to make sure the hand can be anywhere in the conical space above the controller. The angle made between each finger vector to the center of the palm vector is used (in degrees). Figure 1 shows the typical representation, as rendered on a display unit by the Orion SDK.

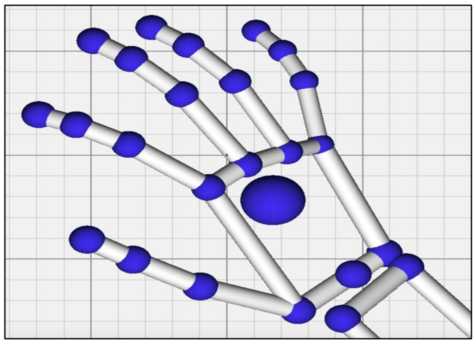
Figure 1. Leap Motion Controller Gesture View

ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 6.887
Volume 6 Issue XII, Dec 2018- Available at www.ijraset.com

Also, for each joint, the distance between each joint and the hand center is found out and likewise the angle made by each joint vector to the center of the palm vector is used. The joints considered are the meta-carpo-phalangeal joint, or knuckle, of the finger, the proximal inter-phalangeal joint of the finger or the middle joint of a finger, the distal inter-phalangeal joint of the finger or the joint is closest to the tip and the tip of the finger. There are four custom gestures defined and used in this project in total (Hold, Grab, Move, Hello-Wave).

The feature space is constructed keeping the above features in mind. The program written to generate the training data, basically run for 100 iterations for the generation of a single training example of any gesture. Iterations use the above-mentioned features, thereby generating a total of 10000 features for a single training instance of any gesture. This is done to make the sample very robust. Also, the movement factors are not included in the feature space because they would become insignificant amongst the relative distance metric that has been used. The translation and clench factors are used in association with the model when it is employed in real time.

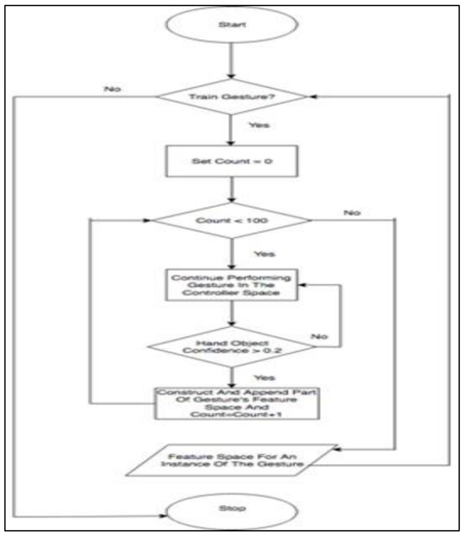
Start
Train Gesture?
No
Yes
Set Count = 0
Count < 100
No
Yes
Continue Performing Gesture In The Controller Space
Hand Object Confidence > 0.2
No
Yes
Construct And Append Part Of Gesture's Feature Space And Count=Count+1
Feature Space For An Instance Of The Gesture
Stop

Figure 2. The Training Process

Figure 2 depicts a graphical representation of the training process followed.

B. Testing And Real Time Gesture Recognition

In testing the model statically, multiple instances of each gesture are enacted over the controller and the resulting feature spaces are recorded in a file and they are sent as input to the model trained prior and the model outputs along with the expected outputs are observed. Figure 3 pictorially depicts the process followed for this.

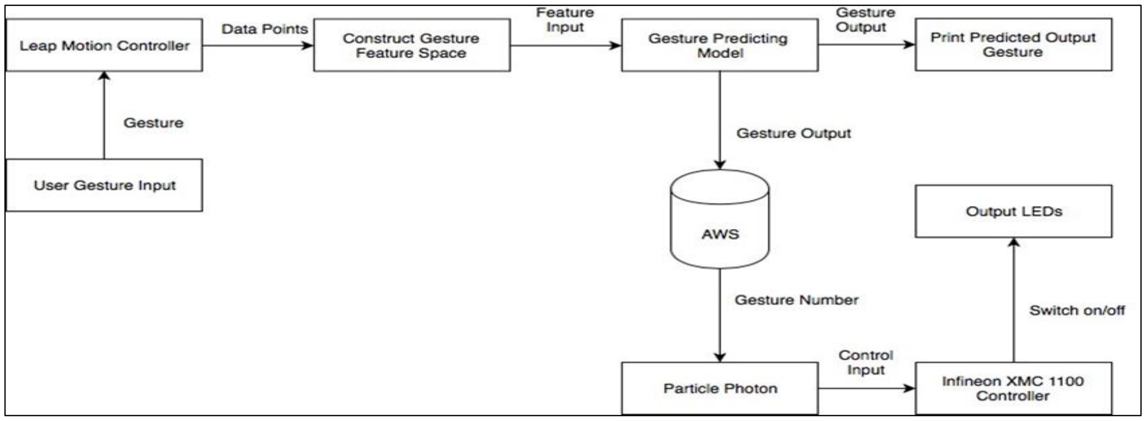

Figure3. Real Time Gesture Recognition

In the real time test case scenario, there are a few differences that need to be taken care of for the model to work as expected. The gesture input is converted along with the data points obtained by the leap motion controller and the program written into a defined feature space. The trained model along with translation_intent_factor is used to check the translational movement of the palm. The hand.grab_strength feature is checked to be a value of 1 at least once during the gesture motion to check closure of palm and also in the same gesture movement is subsequently checked to have a value less than 0.55 to make sure the palm has opened during the gesture.

The model output with the movement factors gives the final gesture output. The output is displayed on the terminal and also sent to the AWS instance running which then send the gesture output number to the particle photon which send the control input to the Infineon XMC 1100 controller which is responsible for controlling the switching on/off of specific LEDs based on the gesture.

III. RESULTS AND ANALYSIS

There were two ways of testing the model that was developed. One of them was reading from a test file which had gesture feature data for multiple gestures and the other was performing gestures in real time with the controller and evaluating accuracy of the model's performance. In real time, each gesture was successful in controlling the switching on and off of a device that was assigned to it with an accuracy of 92.3% using SVMs with RBF (Radial Basis Function) with an accuracy of 100% when the testing is done while reading the gesture feature data from a file.

The reason for the difference of accuracy is because of the movement factors added during real time gesture check and the transition that happens between the gestures.

IV. CONCLUSION

The model developed to recognize gestures was successful in recognizing the gestures defined with a fairly high accuracy and the result of this was the robust switching on/off of the devices each of which was assigned to respond to a specific gesture.

V. FUTURE WORK

The future work includes adding more gestures to control more devices and using the model developed along with the leap motion devices to control actual home appliances and not just LEDs. The gesture recognition concept along with the leap motion controller can also be extended to a multitude of other use cases not just limited to home automation which can possibly include detection of signs in sign language and converting it to text or speech.

REFERENCES

[1] M. B. Khan, K. Mishra and M. A. Qadeer, "Gesture recognition using Open-CV," 2017 7th International Conference on Communication Systems and Network Technologies (CSNT), Nagpur, India, 2017, pp. 167-171. doi: 10.1109/CSNT.2017.8418531

[2] W. Lu, Z. Tong and J. Chu, "Dynamic Hand Gesture Recognition With Leap Motion Controller," in IEEE Signal Processing Letters, vol. 23, no. 9, pp. 1188-1192, Sept. 2016. doi: 10.1109/LSP.2016.2590470

[3] Colgan, A. and Colgan, A. (2018). How Does the Leap Motion Controller Work?. [online] Leap Motion Blog. Available at: http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/.

[4] D. A. Maharani, H. Fakhrurroja, Riyanto and C. Machbub, "Hand gesture recognition using K-means clustering and Support Vector Machine," 2018 IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE), Penang, 2018, pp. 1-6. doi: 10.1109/ISCAIE.2018.8405435

[5] L. Liu, M. Chu, R. Gong and D. Xu, "Classification method based on global and local support vector machine," 2018 Chinese Control And Decision Conference (CCDC), Shenyang, 2018, pp. 1025-1030. doi: 10.1109/CCDC.2018.8407280

[6] Y. Liu and K. K. Parhi, "Computing RBF Kernel for SVM Classification Using Stochastic Logic," 2016 IEEE International Workshop on Signal Processing Systems (SiPS), Dallas, TX, 2016, pp. 327-332. doi: 10.1109/SiPS.2016.64

[7] Y. Rizk, N. Mitri and M. Awad, "An ordinal kernel trick for a computationally efficient support vector machine," 2014 International Joint Conference on Neural Networks (IJCNN), Beijing, 2014, pp. 3930-3937. doi: 10.1109/IJCNN.2014.6889884