Webology (ISSN: 1735-188X)
Volume 18, Number 6, 2021

CONTROLLING POWER POINT USING HAND GESTURES IN PYTHON

Muhammad Idrees[1] , Ashfaq Ahmad[2] , Muhammad Arif Butt[3] , and Hafiz Muhammad Danish[4]

[1]Department of Data Science, University of the Punjab, Lahore, Pakistan.

[2]Department of Computer Science, College of Computer Science and Information Technology, Jazan University, Saudi Arabia.

[3]Department of Data Science, University of the Punjab, Lahore, Pakistan.

[4]Department of Computer Science, University of Lahore, Lahore, Pakistan.

ABSTRACT

Presentations have a significant role in different fields of life. Whether you are a student, an entrepreneur, businessman, or a corporate worker, you must have had given presentations at some point in your life. PowerPoint presentations sometimes become less lively because either you have to use the keyboard to change and operate the slides or use a dedicated gadget to perform these tasks. We aimed to enable people to control the slideshow with the gestures of hands. The applications of gestures in human-computer interaction have massively risen in the past few years. The research has tried to control different operations of the PowerPoint slideshow through gestures. This research has used Machine Learning to detect gestures with subtle differences and tried to map them with some fundamental PowerPoint slideshow controlling functions using Python.

KEYWORDS Hand gesture recognition, PowerPoint, gesture control Power Point.

1. INTRODUCTION

The presentation is a source of communication between a speaker and an audience. It conveys information, delivers a lecture, demonstrates new ideas, and exhibits your thoughts to a group of people. It illuminates the content of a topic to a learner, either in academia or for business purposes. PowerPoint is popular software that improves your presentation skills, by providing a visual illustration of your content. It allows presenting text, diagrams, audios, videos, statistical graphs, animations, etc. A presenter can move PowerPoint slides forward or backward by using a mouse, a keyboard, or remote control, but these Human-Computer Interactions (HCI) gadgets are inconvenient and incommodious. However, a hand gesture recognition system has been proposed that will control PowerPoint slides without any disturbance during the presentation.

A. What is a gesture?

A gesture represents a non-verbal and non-vocal mode of communication. It involves body movements to deliver a particular message to the receiver. It comprises of moving a person's hands, facial features, or other components of the body. The concept of gestures is fascinating and moderately easy to learn.

B. Why Gestures?

The gesture recognition systems have been used for numerous applications, including games, communicating deaf people, emotion detection, and robotics. Hand gestures, facial gestures, and other gestures can control these applications. Gestures are of two types: dynamic and static hand gestures.

C. Static Hand Gesture

A static hand gesture represents a single pose or a configuration of the hand. When the position and orientation of the hand do not change for some time, it is categorized as Static hand gestures.

D. Dynamic Hand Gesture

A dynamic hand gesture is a gesture that intends to change over time.

Dynamic hand gestures are the movement of hand like waving of hand such as 'goodbye' sign whereas a static gesture includes a sign like 'Stop Sign' or 'Ok' sign. One of the main problems with static gestures is that we can perform only limited functions.

In our system, static as well as dynamic gestures have been used to perform PowerPoint functionalities. There are two main techniques for gesture recognition: vision and non-vision-based. The latter technique uses a pair of wired gloves to detect the motion of fingers and hands. Glove Based systems require the user to wear custom gloves for their purposes. This technique uses sensors to convert the data from hands to multiparametric data. The greater the number of sensors, the easier it is to collect such data, and very helpful for recording hand movements. However, these sorts of devices are expensive. Therefore, interaction based on visions appears to be a more inexpensive and natural mode of human-machine interaction. Vision-based techniques mainly require a camera that is provided by every laptop manufacturer nowadays.

Vision has more potential of carrying a piece of precious information at a low cost. Vision-based approaches have many applications like robotics, surveillance, monitoring, tracking, security systems, and augmented reality. Vision-based gesture detection can also be improved using deep learning. We have used deep learning in our system to make it more useable, natural, and reliable. Our aim for developing this project was to ease out the presentations and to finding the best methodology to develop a low-cost gesture detection system to control the PowerPoint application. We have to go through different datasets and methods to find the best way to do our desired task. We want to provide a solution to control the slideshow of PowerPoint so that people may use hand gestures to control the slideshow of PowerPoint by using their laptop camera. Now, let's talk about what gestures we are using and what functions these gestures in particulate have. We make categories for our project shown in Table 1.

Table 1: Gestures used for the project

Gesture	Working	Gesture	Working
Stop Sign	Close Program, End the presentation/ pen/ highlighter	Drumming Fingers	To start pen/ highlighter
Swiping Left	Move to the previous slide	Zooming in with two fingers	Zoom in a picture/ slide
Swiping Right	Move to the next slide	Zooming out with two fingers	Zoom out a picture/slide
Thumb Up	Play and pause video on slide / Start Presentation	No gesture / do other things	No function

2. RELATED WORK

Numerous methods for detecting hand gestures like (AlSaedi & Al Asadi, 2020) proposed a system in which, five phases were used; image acquisition, preprocessing the image, detection & separation of the hand's area, extracting the details, and finally to count the fingers of the segmented hand and performed operations on finger count. (A. & A., 2017) built a mobile robot and used Raspberry pi for hand gesture detection.(Fourney et al., 2010) introduced Maestro for dynamic presentations. (M, 2018) used fixed gestures for Microsoft PP control using a method called thinning. She used four modules inclusive of the above-stated steps from hand detection to recognition and slide control. (Garg et al., 2009) used such methods of gesture detection. (Chen et al., 2015) used a Kinect sensor to detect the movement of hands in real-time. (Ma & Peng, 2018) used Microsoft Kinect to identify gestures over a long distance and the tips of the fingers with detailed information of depth. (Li, 2012) used Kinect for Hand gesture recognition using Kinect. (Haria et al., 2017) used dynamic and static gestures to do many actions like launching VLC and PowerPoint applications. (Nancy & Singh Sekhon, 2012) put forward the color marker technique for gesture recognition. (Phi et al., 2015) simulate a gesture recognition system to recognize Signs of the Vietnamese Language using Glove-based hand gestures. (Pinto et al., 2019) proposed a system and used Convolutional Neural Networks for static hand gestures. (Dubey et al., 2019) trained Neural Networks for Hand Gesture Recognition.

Many people used different approaches for hand gesture detection, which we can find from the above-stated works like vision-based approach, data-glove approach, use of Kinect, and machine learning approach. Our strategy based on using visionary-based machine learning for detecting hand gestures is the most suitable approach. The only vision-based tactic is superb for static gestures and gesture detection by counting contours but not very efficient for dynamic gestures. The glove-based strategy and use of Kinect are expensive as we have to buy sensors or Microsoft Kinect for these approaches. Due to the factors mentioned earlier, we used a machine learning approach for gesture detection.

3. DATA SET

A massive dataset of mini clips of different gestures, distinct backgrounds, lighting conditions, and skin tones, is required to accomplish this project. The Sheffield Kinect Gesture (SKIG) dataset consists of 2160 videos of gestures inclusive of 1080 sequences for both RGB and depth separately. A Kinect Sensor was used by six individuals to acquire the data set. It consists of gestures that are classified into ten categories inclusive of the circle, triangle, up-down, left-right, pat, come here, turn-around, wave, zigzag (Z), and cross. These sequences are based on three hand movements and included various backgrounds and illumination conditions to add diversity (Liu & Shao, 2013). Ego Gesture dataset provided by (Zhang et al., 2018), has 83 categories of hand gestures of both types. The dataset consists of 24161 gestures, both static and dynamic, 2953224 frames, and 2081 RGB-D videos recorded by fifty individuals. These gestures were collected from indoor and outdoor scenes (Zhang et al., 2018). Sebastian Marcel Hand Posture and Gesture datasets comprised of five datasets, three of which are static, and the remaining two are dynamic (Marcel et al., 2000). Microsoft Kinect and leap motion dataset possesses many distinct gestures adequate by both the Leap Motion sensor and Kinect devices. These datasets can help in the establishment of a hybrid gesture recognition system. It holds ten gestures recorded by fourteen distinct individuals distinctively for 1400 videos. Each video was rerecorded ten times, and the data was obtained from both Kinect and Leap motion sensor (Marin et al., 2014, 2015). Creative Senz3D is a dataset possessing static gestures captured by the Senz3D camera. The camera was to check the accuracy of a multi-class classifier to check SVM gesture after being trained on synthetic data produced with Hand-Pose-Generator. This dataset contains gestures performed by four individuals who performed 1320 total gestures comprised of repeating eleven signs thirty times (Memo et al., 2015; Memo & Zanuttigh, 2016). Those were static gestures, so we didn't use them. The NATOPS (The Naval Air Training and Operating Procedures Standardization) is a standard document used for the US Naval Air procedures standardization. NATOPS is a manual in which they mentioned the signals related to aircraft. They picked 24 NATOPS aircraft handling signals. The recordings were of 320x240 resolution from a stereo camera in the database. The frame rate of these videos is 20 frames per second. Twenty individuals recorded 24 gestures, 20 times repeat for everyone, and got 400 videos from each classroom. Each video had a separate run time, with the average of them being 2.34 seconds. The samples were captured in a room with illuminated conditions with fixed camera positions (Song et al., 2011). This dataset was purely for aircraft signals, so that was not suitable for our project.

DVS218 dataset provided by IBM collected using a DVS218 camera consists of eleven gestures performed by 29 individuals, with three illuminated conditions. The gesture categories used in the dataset are not suitable for our project (Amir et al., 2017). Cha Learn Gesture Challenge provided frequently used gesture datasets "Cha Learn LAP IsoGD and Con GD datasets" (CLIC) (Wan et al., 2016), and "Multi-modal Gesture Dataset" (MMGD) (Escalera et al., 2013). The gesture classes in CLIC datasets found from different domain types are 9 in total, are from Italian language signs. Multi-modal Gesture Dataset (MMGD) has 20 gestures categories. RWTH-BOSTON-50 (Zahedi et al., 2006) has been designed for isolated gesture recognition experiments, and RWTHBOSTON-400 comprises 633 sequences recorded for four different speakers.

Another dataset is the 20BN-jester V1 dataset (Materzynska et al., 2019). Table 2 represents the 20BN Jester V1 dataset class's overview.

Table 2: 20BN Jester V1 Dataset Classes Overview

Gesture	**Videos**	**Gesture**	**Videos**	**Gesture**	**Videos**
Doing Other Things	12416	Zooming Out with Two Fingures	5379	Pulling Two Fingers In	5315
No Gesture	5344	Zooming In with Two Fingures	5355	Pushing Two Fingers Away	5358
Stop Sign	5413	Zooming Out with Full Hand	5330	Sliding Two Fingers Up	5262
Thumb Down	5460	Zooming In with Full Hand	5307	Sliding Two Fingers Down	5410
Thumb Up	5457	Turning Hand Counterclockwise	4181	Sliding Two Fingers Right	5244
Swiping Up	5140	Turning Hand Clockwise	3980	Sliding Two Fingers Left	5345
Swiping Down	5303	Rolling Hand Backward	5031	Pulling Hand In	5379
Swiping Right	5066	Rolling Hand Forward	5165	Pushing Hand Away	5434
Swiping Left	5160	Drumming Fingers	5444	Shaking Hand	5314

We went through a lot of gesture datasets, but none of them was suitable. The two main reasons for selecting the 20BN-jester Dataset V1 are the volume of the dataset. It provides many gestures where each consists of a sufficient number of videos suitable for training data. The second main reason is that the signs are fundamental. It is one of the massive datasets containing

real-world gestures. The dataset includes 148092 small-sized sequences. Almost all the videos are 3 seconds in length, and if we do the math, it takes around 5 million frames. The videos consist of a person performing a gesture. The dataset has 1,378 different people that perform the same 27 actions. And since there are so many actors, variation in the lighting, background condition, color tones, and hand shapes is perfect for our project. The specifications of the dataset are given in table 3.

Table 3: Dataset Specifications

Dataset Specifications	**Count**	**Dataset Specifications**	**Count**
Videos Count	148,092	Average length	3 sec
Frames Count	5,331,312	The average count of videos in each class	4391
Classes Count	27	The average count of videos per individual.	43
Number of actors	1376		

These gestures demand understanding of both facial and spatial features, such as their involvement in pinching in and out fingers to zoom in and out, similarly pushing of the hand inwards or outwards. Gestures that we used for our project are demonstrated in Figure 1 through Figure 7.

Figure 1: Stop Sign

Figure 2 a, b, c, d: Swiping Left

Figure 3: Thumb Up

Figure 4 a, b, c, d: Swiping Right

Figure 5 a, b, c, d: Drumming Fingers

Figure 6 a, b, c, d: Zooming out with two fingers

Figure 7 a, b, c, d: Zooming in with two fingers

4. MODEL

The baseline model is mentioned in 20BN Jester Dataset V1 (Materzynska et al., 2019), in which Spatio-temporal filters are utilized as their essential component. The first layer of the model is a Conv3d convolutional block. It contains a convolutional layer, Re LU non-linearity, and batch normalization layer. The model is comprised 3 of the above-cited blocks, each succeeded by a pooling layer by strides operational on the spatial dimensions. Then three additional convolution blocks (3D) are applied, and then another max-pooling global layer is employed in the end. Therefore, the output of the max-pooling global layer is an x-feature temporal step that maps channel dimensions vector. It is stored in a recurring layer through LSTM cell, and then it is passed through a fully connected layer.

The model was trained by using SGD with a learning rate of 0.001 for 100 epochs and achieved an accuracy of 93.87%. We have used a seven-layered model shown in Figure 8, for training the dataset provided by https://github.com/eleow/Gesture-Recognition-and-Control/blob/master/model.py.

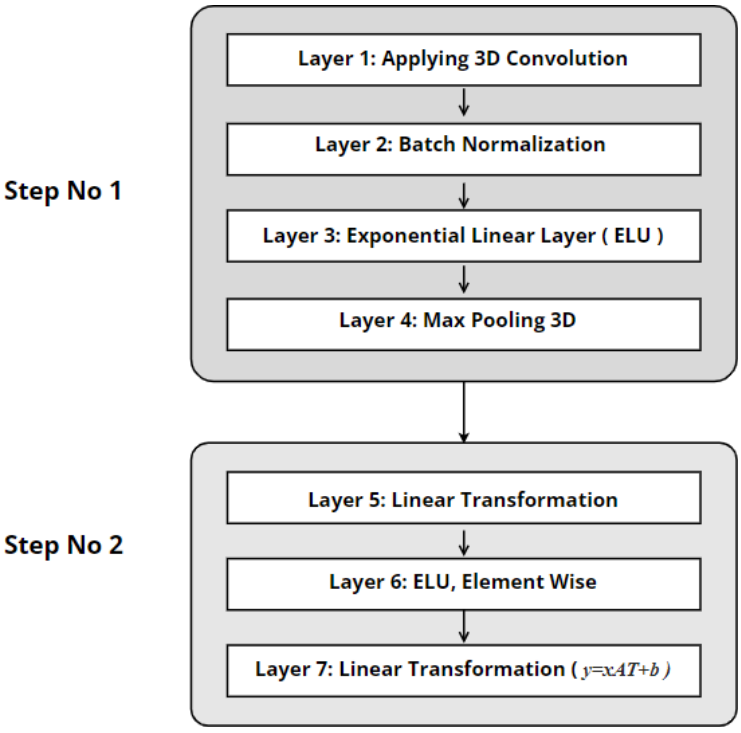

Figure 8 a, b, c, d: Zooming in with two fingers

A. Step No 1

In first step, four layers are used:

a) Layer 1

The first layer is responsible for applying a three-dimensional convolution, made up of numerous input planes. The output result of the layer with input size (Ni, C_out, D_out, H_out, W_out) is represented by:

$$OUT(Ni, C\_outj) = BIAS(C\_outj) + k=0 \sum Cin - 1\ WGT(C\_outj, k) * INP(Ni, k)$$

- **In_channels** (integer) = Count of channels of input image.
- **Out_channels** (integer) = Count of channels after Convolution
- **kernel_size** (tuple or integer) = kernel Size for Convolution.
- **stride** (optional, integer or tuple,) = Convolution Tread: Default value: 1
- **padding** (optional, integer or tuple) = Zero padding along all sides of the input.

b) Layer 2

The second layer is BatchNorm3d, in which Batch Normalization has been used over a 5D input. Dropping Internal Covariate Shift is used in order to accelerate network training.

$$y = x - E[x]\ Var[x] + \epsilon * \gamma + \beta$$

Over the mini-batches, standard deviation and mean are calculated. Gamma (γ) and Beta (β) are vectors, where default value of nts of γ is one and the value of β element is zero.

c) Layer 3

In third layer, Exponential Linear Unit (ELU) is used and its formulation is given below:

$$ELU(x) = maximum(0, x) + minimum(0, \alpha * (e(x) - 1))$$

- **alpha** = the default value of α is 1.0 for ELU
- **inplace** = inplace is optional operation, by default it Zero.

d) Layer 4

The fourth layer 3D Max Pooling function applied over the input signal collected from multiple input planes.

B. Step No 2

In second step, three layers are used:

a) Layer 5

The fifth layer is a Linear (12800, 512) transformation that has been applied to incoming data.

$$Y = xAT + b$$

b) Layer 6

The sixth layer is nn. ELU, used to calculate element wise exponential linear function:

$$ELU(x) = maximum(0, x) + minimum(0, \alpha * (e(x) - 1))$$

- **alpha** – the default value of α is 1.0 for ELU

- **inplace** – inplace is optional operation, by default it Zero.

c) Layer 7

The seventh layer is Linear (512, num_classes) where the number of classes are 9 to apply the linear transformation.

It applies a linear transformation to the input data: y=xAT+b

5. TRAINING

We have used 29999 videos of the above-cited categories in total for training and 5374 validation videos. We trained our model for 20 epochs. We picked Py Torch as our preferred library for training purposes. Py Torch is an open-source library for machine learning and is implemented upon the Torch library. The development of this library is mainly attributed to Facebook's AI Research lab and is preferred over its competitors Keras and TensorFlow after careful consideration.

Deep learning depends on training that uses an immense size of datasets. So, the system driving the training demands to be equipped with enough computational power to deal with such datasets and produce results quickly. The use of a GPU significantly reduces the training time and provides results much quicker. The greater the computational power of the GPU, the more beneficial it is to the ordinary CPU. The system used for the training process had the following specifications:

- Hp pavilion 15-cx0058wm
- Intel® Core™ i5-8300H (2.3 GHz base frequency, up to 4 GHz with Intel® Turbo Boost Technology, 8 MB cache, 4 cores) 8GB DDR4 System memory
- 8 GB DDR4-2666 SDRAM (1 x 8 GB)1TB 7200rpm hard drive storage
- 16 GB Flash cache
- NVIDIA® GeForce® GTX 1050 (4 GB GDDR5 dedicated)

6. METHODOLOGY

The methodology that we used for our project consists of different phases.

The first and second phases include selecting and opening a PowerPoint file for presentation on PowerPoint Windows. The user selects the PowerPoint file to open, and our system will open the file for presentation. The user can select .ppt, pptx, .pptm files for presentation. After selecting, our program will automatically open the file.

In the third phase, the system starts a live video stream for detecting and recognizing the live gestures. A built-in or an external webcam will record this live stream. The gesturing will be recorded as an image array of size 20 in the fourth phase. This array will help detect a specific gesture. This array can be an entire performed gesture or action recorded frame by frame and fed to the network for detection. This image array is an array of continuous frames where every frame is processed 20 times for gesture or action recognition. A transform function transforms

this array before predicting it for a specific action. The fifth phase transforms this array. The transform function used in our project is as follows

Torch vision. transforms. Compose ([To PIL Image(), Center Crop (84), To Tensor (), Normalize(mean=[0.485,0.456,0.406], std=[0.229,0.224,0.225])])

Torch vision. transforms are common image transformations, and they can be chained together using Compose functions.

In the sixth phase, the array of images is classified using the pre-trained weights, loaded earlier by the system. The whole array can be organized into nine categories based on the recorded actions. When an activity is detected, the mapped keys are pressed virtually to perform the desired task. If the identified category is among the gestures, then only the keys are pressed implicitly. We have selected the threshold of 0.8 or 80%, i.e., if the accuracy of the recorded action is equal to or more than 80%, then the operation is performed accordingly. Figure 9 depicts the methodology.

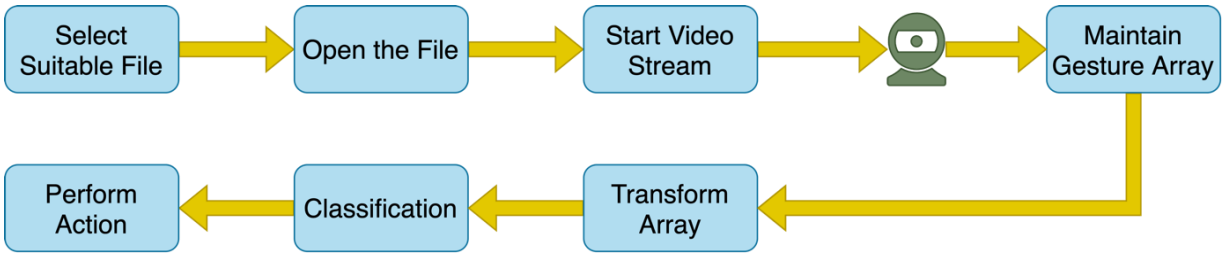

Figure 9: Step Diagram

7. Implementation And Working

The program provides a graphical user interface (GUI) to select a file for presentation. This GUI is build using the Tkinter module in python. Using this GUI, the file is browsed and selected. After the file is selected, it is opened by the system, as discussed in the previous section. PowerPoint Windows application is started with the selected file for presentation using PyWin32.

A wrapper of python language is peculiarly used for communicating with COM objects and controlling Windows applications. This wrapper allows the user to do any required task of a Microsoft application using python. Then the checkpoint of training is loaded using Py Torch. Then the video stream is started using the python-OpenCV library. Then the user will be able to perform gestures. The 'Thumb up' posture starts the slideshow of the presentation in the PowerPoint application. Other gestures would not work until starting the slideshow, so the user must perform the 'Thumb Up' gesture in the first place. ‘Swipe Right’ gesture would be used for the next slide, i.e., ‘Right Arrow Key’ is pressed virtually. ‘Swipe Left’ gesture would be used for the previous slide by virtually pressing the ‘Right Arrow Key’. For the slide zooming-in, perform the ‘Zooming in with two fingers’ gesture that will virtually press the combination of ‘Ctrl’ and ‘+’ keys. ‘Zooming out with two fingers' causes the slide to zoom out. To perform this action ‘Ctrl’ and ‘-’ keys are pressed virtually. If there is a video on the slide, it will be played using the ‘Thumb Up’ gesture. A ‘Thumb Up’ gesture also pauses the playing video.

When the 'Thumb Up' Gesture will be performed then 'ALT' and 'P' keys are pressed virtually. To start the highlighter 'Drumming Fingers' gesture should be performed. If the gesture is performed again, PowerPoint will start the pen, i.e., if you want to use the pen you have to perform drumming gestures twice. Drumming fingers gesture is also used to switch between the highlighter and the pen. The combination of 'CTRL' and 'I' is pressed either or the combination of 'CTRL' and 'P' depending upon the condition. To turn off the pen or highlighter mode, the 'Stop Sign 'gesture will be needed. 'Stop Sign' is also used to end the slideshow and to close the application, i.e., if it is needed to exit the application and the user is currently in slideshow mode, the user should perform this gesture twice, and keys are pressed virtually according to the stage at which the gesture is performed. To start the slideshow again the user has to perform the 'Thumb Up' gesture as mentioned earlier. The keys are pressed virtually by using Py Auto GUI. Py Auto GUI library is used when the user desires to control the mouse and keyboard for any required task. This GUI can operate on multiple platforms and is for automation. The purpose of including this library is to virtually press a single key or a combination of keys. Figure 10 depicts the activity flowchart.

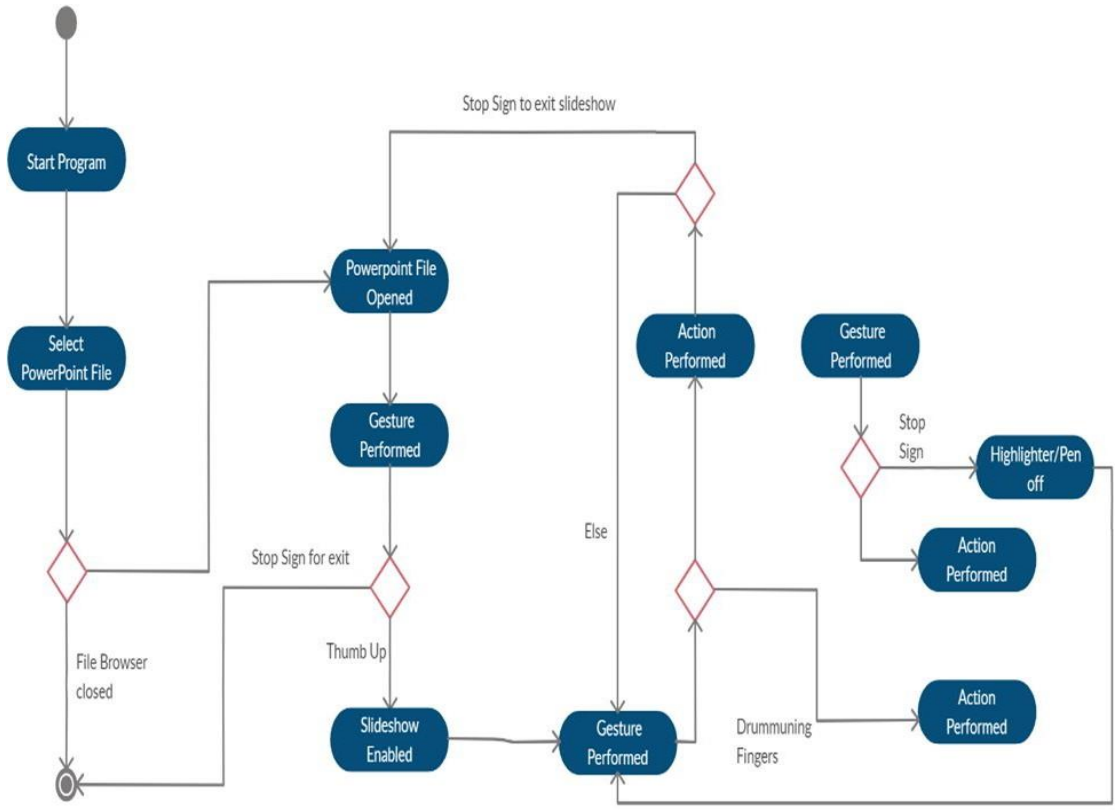

Figure 10: Activity Diagram

The observed range for the person's gestures to be detected correctly is a maximum of 2 feet. The original dataset used for training contained videos that were very close to the webcam, mainly from 1 to 1.5 feet. This constraint made it significantly harder to increase the range from this distance, as the massive dataset trained the model to detect from this distance only.

Therefore, additional steps were taken to detect a person standing a short distance away from the webcam as long as he/she was detected without blurring out. Additional libraries and techniques were used to carve out a person after sensing him/her, and then create a box around the person inside which the gestures are to be recognized. The used techniques are:

A. Harr Cascade

Haar Cascade was used to detect the upper bodies of the person standing in front of the camera and then predicting. The problems faced in its use were that it caused too much flickering, which made it significantly harder to predict accurate gestures as there were many false detections.

B. Mobile Net-SSD model

This model was also used for the same purpose of detecting the upper body and showed similar problems related to flickering and false detections.

To resolve these issues, a box was created on the screen for the presenter to stand in and present for accurate readings. The gestures are to be recognized in this specified area only, which caused problems related to the mobilization and movement of the presenter.

C. Face Recognition API

This Application Programming Interface is used to detect faces. We used this API to segment-out the upper body based on faces. The upper body segmentation using this API is better than the above-cited techniques. The issue is that this API gives CMAKE error when we try to install it in python 3.7. We have used this API in python 3.5.6, but we can't use it in our project because of the torch vision version. Python 3.7 supports torch vision version 0.4.1, which is one of the basic requirements for our project.

D. Use of High-Resolution Camera

As most laptops are not equipped with high-grade cameras, the detection becomes more troublesome due to the image blurring after some distance. The use of HD cameras for this program can help resolve such problems, as they provide more excellent quality and hence increase chances of detection. In conjunction with the above-discussed techniques, the range of detection can be enhanced.

8. RESULTS

We had trained videos for nine categories, and seven among them are gestures that we used to perform some basic PowerPoint tasks. We have set the detection threshold 0.8, i.e., 80% for detecting a hand gesture. The maximum accuracies that we had achieved are given in Table 4 and graphically represented by Figure 11: Accuracy GraphFigure 11.

Table 4: Accuracy Table

Gesture	Accuracy
Swipe Left	100%
Swipe Right	100%

Zooming In using two fingers	98%
Zooming Out by two fingers	98%
Drumming Fingers	98%
Stop Sign	100%
Thumb Up	100%

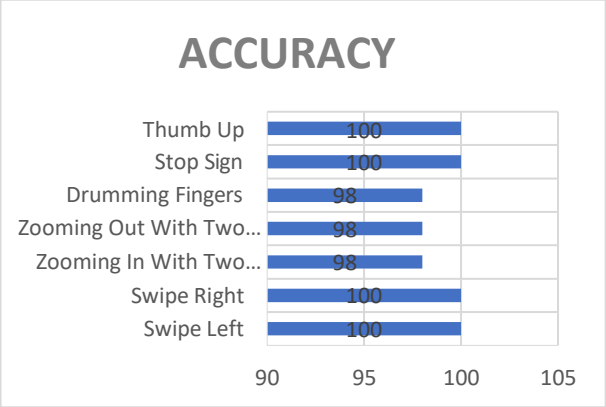

Figure 11: Accuracy Graph

9. Limitations and Future Scope

The significant issue faced in this project is related to the presenter's distance. The dataset contained videos at a specific distance hence the detection can be achieved at that distance only. We have improved this issue to some extent, but it has not been completely fixed. This issue needs some time and attention to resolve by practicing other available works and techniques. We have trained the network on a GPU. The detection is so far so good with the use of GPU, but the detection accuracy is very low on the CPU.

Gestures are being used in many computer science-related fields and are of great importance. Gestures are the future of Human-Machine Interactions due to the need for a more natural way of interacting with computers and machines, as discussed earlier. This project facilitates a presenter with his presentations and helps the students with Human-Computer Interaction (HCI) when gestures come into play. Gestures can be used to control almost everything that we can have in our computer system, and with the blend of speech recognition, we can devise a system, with which one does not have to learn some set of commands or some procedure to work with a machine, but it can all be done by the natural means of communication.

10. Conclusion

While presenting, it is difficult to give the perfect presentation because of focusing on multiple things, including the slides, the keys to change the slides, and the audience, while maintaining composure. This research focused on removing one such distraction by allowing the presenter to manage slides solely by gesturing in front of the camera. We managed to map specific gestures for one action on the slides, including the next slide, previous slide, zoom in and out,

opening a highlighter/pen, and play/pause videos. This project uses the python language, and the library used for machine learning is Py Torch. We used a dataset named 20BN-jester Dataset V1 for the project, which had 148,092 gestures. The results achieved were excellent, which can make hand gestures a righteous mode of presenting.

ACKNOWLEDGEMENT

We would like to thank Muhammad Awais Suhail, Muhammad Hamza bin Shoaib, Danyal Ahmad, and Aqeel Ahmad, the BS students at the Punjab University College of Information Technology, for supporting this research. They were involved in the related development and documentation being part of their final year project. Their contributions are sincerely appreciated and acknowledged.

REFERENCES

A., A., & A., S. (2017). Python-based Raspberry Pi for Hand Gesture Recognition. International Journal of Computer Applications, 173(4), 18–24. https://doi.org/10.5120/IJCA2017915285

Al Saedi, A. K. H., & Al Asadi, A. H. H. (2020). A new hand gestures recognition system. Indonesian Journal of Electrical Engineering and Computer Science, 18(1), 49–55. https://doi.org/10.11591/IJEECS.V18.I1.PP49-55

Amir, A., Taba, B., Berg, D., Melano, T., Mckinstry, J., di Nolfo, C., Nayak, T., Andreopoulos, A., Garreau, G., Mendoza, M., Kusnitz, J., Debole, M., Esser, S., Delbruck, T., Flickner, M., & Modha, D. (2017). A low power, fully event-based gesture recognition system. Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017-January, 7388–7397. https://doi.org/10.1109/CVPR.2017.781

Chen, Y., Luo, B., Chen, Y. L., Liang, G., & Wu, X. (2015). A real-time dynamic hand gesture recognition system using kinect sensor. 2015 IEEE International Conference on Robotics and Biomimetics, IEEE-ROBIO 2015, 2026–2030. https://doi.org/10.1109/ROBIO.2015.7419071

Dubey, K. K., Jha, A., Tiwari, A., & Narmatha, K. (2019). Hand Gesture Movement Recognition System Using Convolution Neural Network Algorithm. IRJCS:: International Research Journal of Computer Science, VI, 154–160. https://doi.org/10.26562/IRJCS.2019.APCS10090

Escalera, S., Gonzàlez, J., Baró, X., Reyes, M., Guyon, I., Athitsos, V., Escalante, H., Sigal, L., Argyros, A., Sminchisescu, C., Bowden, R., & Sclar off, S. (2013). Cha Learn multi-modal gesture recognition 2013: Grand challenge and workshop summary. ICMI 2013 - Proceedings of the 2013 ACM International Conference on Multimodal Interaction, 365–370. https://doi.org/10.1145/2522848.2532597

Fourney, A., Terry, M. J., & Mann, R. A. (2010, September 1). Gesturing in the Wild: Understanding the Effects and Implications of Gesture-Based Interaction for Dynamic

Presentations. Proceedings of the 2010 British Computer Society Conference on Human-Computer Interaction. https://doi.org/10.14236/EWIC/HCI2010.29

Garg, P., Garg, P., Aggarwal, N., & Sofat, S. (2009). Vision Based Hand Gesture Recognition. WORLD ACADEMY OF SCIENCE, ENGINEERING AND TECHNOLOGY, 972--977. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.309.6231

Haria, A., Subramanian, A., Asok kumar, N., Poddar, S., & Nayak, J. S. (2017). Hand Gesture Recognition for Human Computer Interaction. Procedia Computer Science, 115, 367–374. https://doi.org/10.1016/J.PROCS.2017.09.092

Li, Y. (2012). Hand gesture recognition using Kinect. ICSESS 2012 - Proceedings of 2012 IEEE 3rd International Conference on Software Engineering and Service Science, 196–199. https://doi.org/10.1109/ICSESS.2012.6269439

Liu, L., & Shao, L. (2013, June). Learning Discriminative Representations from RGB-D Video Data. Wenty-Third International Joint Conference on Artificial Intelligence.

M, S. (2018). Static Hand Gesture Recognition for PowerPoint Presentation Navigation using Thinning Method. International Journal on Recent and Innovation Trends in Computing and Communication, 6(4), 187–189. https://doi.org/10.17762/IJRITCC.V6I4.1541

Ma, X., & Peng, J. (2018). Kinect sensor-based long-distance hand gesture recognition and fingertip detection with depth information. Journal of Sensors, 2018. https://doi.org/10.1155/2018/5809769

Marcel, S., Bernier, O., Viallet, J. E., & Collobert, D. (2000). Hand gesture recognition using input-output hidden Markov models. Proceedings - 4th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2000, 456–461. https://doi.org/10.1109/AFGR.2000.840674

Marin, G., Dominio, F., & Zanuttigh, P. (2014). Hand gesture recognition with leap motion and kinect devices. 2014 IEEE International Conference on Image Processing, ICIP 2014, 1565–1569. https://doi.org/10.1109/ICIP.2014.7025313

Marin, G., Dominio, F., & Zanuttigh, P. (2015). Hand gesture recognition with jointly calibrated Leap Motion and depth sensor. Multimedia Tools and Applications 2015 75:22, 75(22), 14991–15015. https://doi.org/10.1007/S11042-015-2451-6

Materzynska, J., Berger, G., Bax, I., & Memisevic, R. (2019). The jester dataset: A large-scale video dataset of human gestures. Proceedings - 2019 International Conference on Computer Vision Workshop, ICCVW 2019, 2874–2882. https://doi.org/10.1109/ICCVW.2019.00349

Memo, A., Minto, L., & Zanuttigh, P. (2015). Exploiting silhouette descriptors and synthetic data for hand gesture recognition. Italian Chapter Conference 2015 - Smart Tools and Apps in Computer Graphics, STAG 2015, 15–23. https://doi.org/10.2312/STAG.20151288

Memo, A., & Zanuttigh, P. (2016). Head-mounted gesture controlled interface for human-computer interaction. Multimedia Tools and Applications 2016 77:1, 77(1), 27–53. https://doi.org/10.1007/S11042-016-4223-3

Nancy, & Singh Sekhon, G. (2012). An Analysis of Hand Gesture Recognition Technique Using Finger Movement Detection Based on Color Marker. International Journal of Computer Science & Communication, 3(1), 129–133.

Phi, L. T., Nguyen, H. D., Bui, T. T. Q., & Vu, T. T. (2015). A glove-based gesture recognition system for Vietnamese sign language. ICCAS 2015 - 2015 15th International Conference on Control, Automation and Systems, Proceedings, 1555–1559. https://doi.org/10.1109/ICCAS.2015.7364604

Pinto, R. F., Borges, C. D. B., Almeida, A. M. A., & Paula, I. C. (2019). Static Hand Gesture Recognition Based on Convolutional Neural Networks. Journal of Electrical and Computer Engineering, 2019. https://doi.org/10.1155/2019/4167890

Song, Y., Demirdjian, D., & Davis, R. (2011). Tracking body and hands for gesture recognition: NATOPS aircraft handling signals database. 2011 IEEE International Conference on Automatic Face and Gesture Recognition and Workshops, FG 2011, 500–506. https://doi.org/10.1109/FG.2011.5771448

Wan, J., Li, S. Z., Zhao, Y., Zhou, S., Guyon, I., & Escalera, S. (2016). ChaLearn Looking at People RGB-D Isolated and Continuous Datasets for Gesture Recognition. IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops, 761–769. https://doi.org/10.1109/CVPRW.2016.100

Zahedi, M., Dreuw, P., Rybach, D., Deselaers, T., Bungeroth, J., & Ney, H. (2006). Continuous Sign Language Recognition-Approaches from Speech Recognition and Available Data Resources. Second Workshop on the Representation and Processing of Sign Languages: Lexicographic Matters and Didactic Scenarios, 21–24. http://www.let.ru.nl/sign-lang/echo

Zhang, Y., Cao, C., Cheng, J., & Lu, H. (2018). Ego Gesture: A New Dataset and Benchmark for Egocentric Hand Gesture Recognition. IEEE Transactions on Multimedia, 20(5), 1038–1050. https://doi.org/10.1109/TMM.2018.2808769