AirStrum: A virtual guitar using real-time hand gesture recognition and strumming technique

Beulah ARUL*, Shashank PANDA, Tushar NAIR

Department of Computer Science and Engineering,
Sri Sivasubramaniya Nadar College of Engineering, Kalavakkam, Chennai, Tamil Nadu, India
beulaharul@ssn.edu.in*, shashank19102@cse.ssn.edu.in, tushar19116@cse.ssn.edu.in

***Corresponding author:** Beulah ARUL
beulaharul@ssn.edu.in*

Abstract: The efforts in the field of virtualizing the guitar into well-modelled software systems have faced a lot of practical limitations. The existing guitar simulation programs require additional devices such as Electromyography (EMG) controllers, or Musical Instrument Digital Interface (MIDI)-based recording devices. The EMG-based device is still a work in progress, the device is expensive and it was not very well received by the instrumentalists. There exists a gap in the bridge that joins physical instruments to their software counterparts. In this context, this paper aims to significantly remove the inaccuracies and drawbacks related to the existing solutions by accounting for the individual roles that each hand plays in the act of guitar strumming and consolidating them into a single system. The design of the proposed AirStrum system involves a multi-step process. Initially, a dataset is created by recording images of hand gestures corresponding to the playing of various chords on a guitar. The palm is detected, and its related skeleton image is generated using MediaPipe. Subsequently, a model based on a Convolutional Neural Network (CNN) is trained and validated using the employed dataset to adeptly recognize and classify guitar chords. Additionally, this model incorporates a velocity detection function for the strumming hand. Finally, the proposed system can play different sounds by inferring both the played chord and the strumming velocity from human actions. This comprehensive approach enables a sophisticated virtual guitar experience based on a system that responds dynamically to the users' gestures and strumming techniques. The conducted experiments demonstrate that AirStrum achieves an accuracy of 95.92%, and a brief preliminary survey related to its perceived utility and usability received a positive feedback rate of 58% from eight guitar players.

Keywords: Convolutional Neural Network, Chord Classification, Hand Gesture, MediaPipe, Sound Bank, Virtual Guitar, Velocity Detection.

1. Introduction

The classical guitar is a 6-stringed instrument with a hollow wooden body. It has been one of the oldest and most popular musical instruments for many centuries. There are more than 10 different types of guitars today, which are differentiable by the method of playing or by appearance (Roberts, 2023). Depending on the scope and position of the guitarist, there is a wide array of accessories that would be necessary to aid in the playing of this instrument.

A big problem faced by musicians is the unavailability of their instruments when inspiration strikes. Guitars, in particular, aren't frequently carried around too often as they are generally heavy, space-consuming and prone to damage. As it is impossible to know when inspiration will strike, any potential creative ideas could be lost. An accessible and easy-to-use software system, that doesn't require any physical instrument, and encompasses all elements of guitar playing removes this hurdle.

However, there exists a gap in the bridge that joins physical instruments to their software counterparts. The simulation of the natural muscle movements that trigger sounds based on an instrument is not effectively created through software. Pianos, organs and synthesizers can be accurately represented on computer keyboards to an extent, but larger, more complicated instruments such as guitars, flutes and violins are yet to be modeled in such a concise way (Sun, 2023).

There have been numerous attempts to virtualize musical instruments such as guitars, drums, xylophones, and keyboards. Different techniques have been employed to make this possible. While some instruments were modeled based on the natural method of playing, in other cases a perfectly working system for the processed instrument was created which involved a new method of playing, which required a certain amount of additional knowledge specific to that system.

https://doi.org/10.33436/v34i4y202410

In the case of drums, the individual components would have to be clicked or tapped on to produce sounds. In the system proposed by Bering & Famador (2016), the aim is to use computer vision and OpenCV libraries to create a virtual drum simulator. To do this, image processing techniques such as thresholding and applying Gaussian filters were used to extract the contours of the hand from a bounding rectangle. The hands were tracked successfully while striking different areas on the screen to play the corresponding sound. The limitation of this system lies in the fact that the face gets detected as a hand and users have to keep their head outside the frame of the camera.

The work of Tolentino et al. (2019) focuses on providing users with an affordable alternative to purchasing entire drum sets in order to learn to play drums. They adopted the methodology of Computer Vision to do so. Only the webcam of a computer is put in use to drive the system. The use of color-coded drum sticks is adopted, so that the device can identify the behaviour of an air drum. A yellow slip of paper is stuck to the knee of the player to trigger the kick drum, a piece of equipment that is triggered by the stomping of a pedal in a real drum kit. The authors thus used color-coded blobs on sticks and also added foot functionalities such as bass drum and hi-hat control. But one downside of this system is that objects of the same/similar colour as the apparatus indicators must not be present in the frame, else way the result will be undesirable. Blob detection is implemented on the colour key points, and their movements are approximated to trigger the intended sound.

When analysing the systems proposed for playing the drums (Bering & Famador, 2016; Tolentino et. al., 2019), while they do a good job in playing the right sounds when the hitbox is triggered, they do not model the real-life playing of that instrument. This allows for improvement in the area that would make it no longer necessary to look up at the screen to hit right inside a specific boundary. A drummer would not be able to pick up the system and play without spending some time to learn the nuances inherent to this system. Thus, in developing this application, the aim is to simulate the natural experience of playing a virtual guitar, allowing musicians to perform intuitively without needing to learn complex system-specific nuances.

Similarly, to drums, there have been a few attempts at building models for virtualizing a guitar. The work of Santiago et. al. (2015) is the closest to the proposed system. The user is required to wear a pair of red gloves and is able to pluck at different parts of a line that is drawn on the screen where each part of the line, coloured differently, aims to represent the fretboard of a guitar. Every time a point denoting the right hand passes the string, the sound is to be triggered. The notes are generated using the Karplus-Strong Algorithm to simulate a certain sound. A virtual-reality based virtual drum set was developed by Gomes et al. (2023) to simulate drum percussion through gesture recognition on smartwatches.

The work of Tamani et al. (2018) introduces the forearm electromyography to be used for the forearm of the guitar player. The introduction of a physical device directly linking to the muscles of the human hand produced more accurate results and made this experience feel relatively more natural than playing on a simulator screen. The EMG model of a guitar, as a prototype is able to detect the different types of strumming and the chord identification by the other hand. This system is an expensive proposal, posing potential challenges for widespread acceptance among the general learners. Based on a survey conducted by the research team, the apparatus was not very well received by instrumentalists and they would rather play the traditional instrument.

Azuma et al. (2022) developed a virtual guitar learning system for beginners using augmented reality. They have incorporated a MIDI guitar controller detection capable of capturing information about the pressed strings. Furthermore, their system offers corrective animations to address mistakes. PeiQiao & Fujimoto (2022) suggested a prototype for a smartphone application to simulate a virtual slide guitar. The authors claim this is a potential alternative to an actual guitar. However, it needs an extra device to simulate an actual analog guitar.

Currently, there are a lot of software applications that enable the playing of an instrument by employing some hardware or sensors. In the case of drums, the individual components would have to be clicked or tapped on to produce the sound using VR equipment (Gomes et al, 2023). Smith (2023) presents existing virtual air guitar using an infrared slide tube to track the hand gestures. The next breakthrough involved the use of an additional physical device such as Electromyography (EMG) controllers (Tamani et al., 2018), or of the Musical Instrument Digital Interface (MIDI)-

based recording devices to recreate the feeling of playing a guitar. With the existing guitar simulation technology, strumming velocity can be identified based on EMG and chord shapes can be recognized individually by Computer Vision, but there are no solutions that provide a consolidated software that identifies the chord shape based on a hand gesture and strumming action simultaneously.

To mitigate these limitations, this paper proposes a software system using the concept of computer vision, which accurately models the real-life experience of playing a guitar, accounting for the nuances of muscle movements without the need for any additional sensory apparatus. The contributions of this paper are listed below:

- Dataset creation by recording the images of hand gestures corresponding to the chords of the guitar;
- Training and validating a CNN model to recognize chords using the created dataset;
- Classification of chords using the developed model based on hand gestures;
- Velocity detection for the strumming hand (right hand);
- Recognizing the chord and strumming velocity from human actions to play the appropriate corresponding sound from the sound bank with .wav files of previously recorded major chords A to G played on a real guitar at two different strumming velocities.

The proposed AirStrum system seamlessly gathers real-time inputs from a user's video feed by differentiating the left hand from the right one, and thus prescribing the correct gesture for each hand. The left hand must focus on the guitar's chord construction using one's fingers, and the right hand`s motion must trigger the strumming sound of a guitar, once it passes the designated string area.

The remainder of this section is organized as follows: Section 2 covers the methodology used to develop the Airstum virtual guitar simulation system, Section 3 presents the results and its related discussion, Section 4 concludes the paper. In Section 5 we have given the links to access sample programs and the created dataset.

2. Methodology

2.1. AirStrum: virtual guitar simulation system

Without the use of any external apparatus and only based on the camera feed along with concepts from the domain of computer vision, cloud storage and web programming, a seamless guitar playing experience is simulated in this paper. The proposed system broadly consists of two modules that handle gestures from the left and right hands respectively.

- Left Hand: The position and gestures are captured and determine what chord sound is to be played;
- Right Hand: The movement speed dictates the intensity of the sound that is played on strumming of an actual guitar.

The chord dataset is created for six basic chords. The MediaPipe framework is used to recognize the hand and its gestures. MediaPipe's palm detector module detects the presence of the hand or palm. Using the hand landmark co-ordinates, the corresponding chords are trained for the left hand. When the player shows strum action, the velocity is calculated using a novel velocity calculation method. The chord and velocity strum played by both the hands are processed remotely using a server hosted on the Google Cloud Platform (GCP) and the output is presented as the respective guitar sound on the interface. The detailed architecture of the virtual guitar simulator is shown in Figure 1.

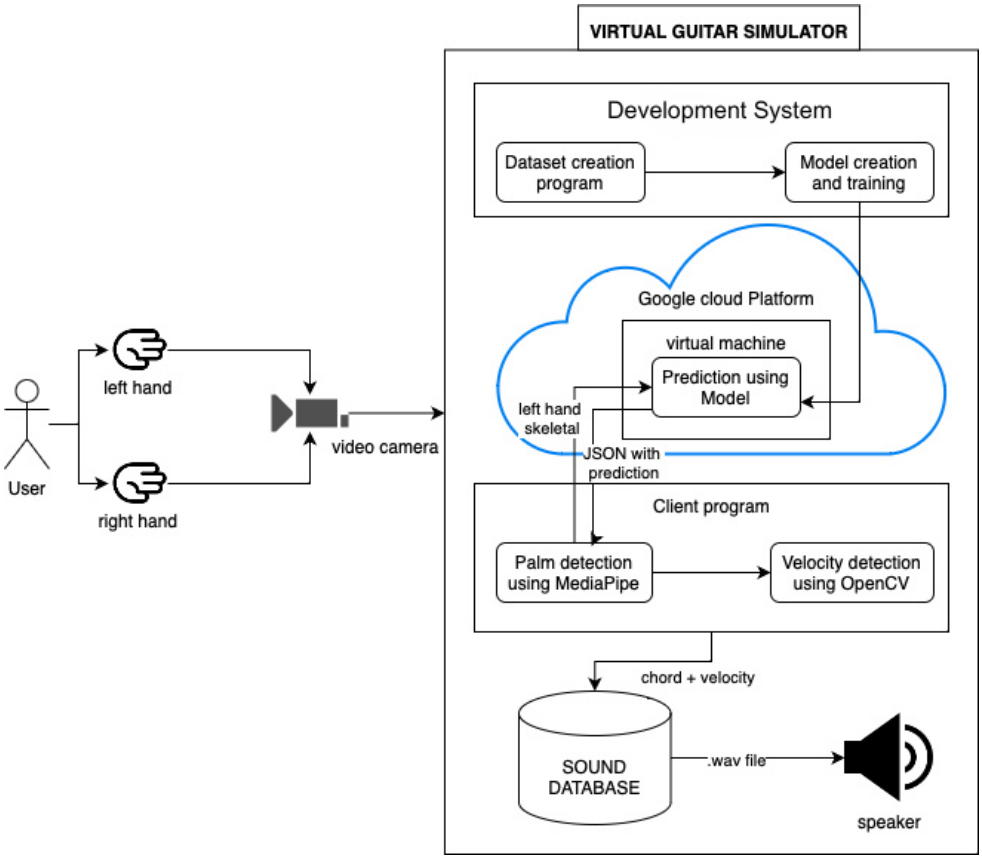

Figure 1. The architecture of the virtual guitar simulator

2.2. MediaPipe

MediaPipe is an open-source framework developed by Google that provides tools for building various perceptual computing applications, including solutions for face detection, hand tracking, pose estimation, and other computer vision tasks. These programs are written in the Python language and can be implemented and customized so as to suit the problem statement. For hand gesture and position recognition, the MediaPipe Hands solution stands out as the most effective (Zhang et al., 2020). This involves using two separate models, a Palm Detection model and a Hand Landmark model. The MediaPipe Hands model would undergo extensive training with various datasets to ensure an accurate performance. These datasets are diverse in terms of how and where the data was collected, helping to improve the overall model accuracy. The inclusion of multiple datasets increases the robustness of this model.

2.2.1. Palm detection model

A single-shot detection mode called BlazePalm Detector is used to identify the presence of hands on the screen. Unlike face detection, the detection of hands is very complex as it has to deal with varied hand sizes and problems with occlusion and self-occlusion. Trying to pinpoint the landmarks directly on the fingers of the hand is difficult due to the large differences that are present. So, this model generates the bounding boxes for the entire palm or fist of a hand as an initial step. This ensures a lower number of anchors and a good scene context-awareness even for small objects. The architecture of the palm detector model is shown in Figure 2.

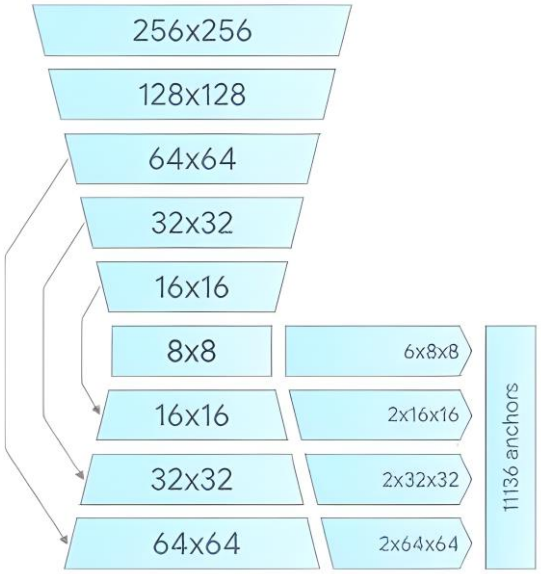

Figure 2. Palm detector model architecture (Zhang et al., 2020)

When the presence of the hand is first detected on the screen, the palm detector model gets triggered and executed. It is not run at every frame of the video unless the hand in focus was removed from the frame or a new hand was added into the scene. This saves a lot of computational power by avoiding the running of an unnecessary detection function. There exists a probability score to check if a hand is present in the frame or not. When this score falls below the acceptable threshold, then it is called a tracking failure and the Palm detection model is triggered to reset the tracking.

2.2.2. Hand landmark model

In the second stage, the hand landmark detection model is run and it will identify and plot 21 distinct landmark coordinate points on the hand in 2.5D. This model is trained on the various hand poses and provides an accurate landmark localization even when the hands are partially visible or occluded via means of regression. The architecture of the hand landmark model is shown in Figure 3.

The model has three outputs, namely:

1. There are 21 hand landmarks, each providing the x, y, and z coordinates to represent the relative depth.
2. A flag is present to indicate if a hand is detected in the frame.
3. The classification of handedness, determining whether it is the right or the left hand, is included.

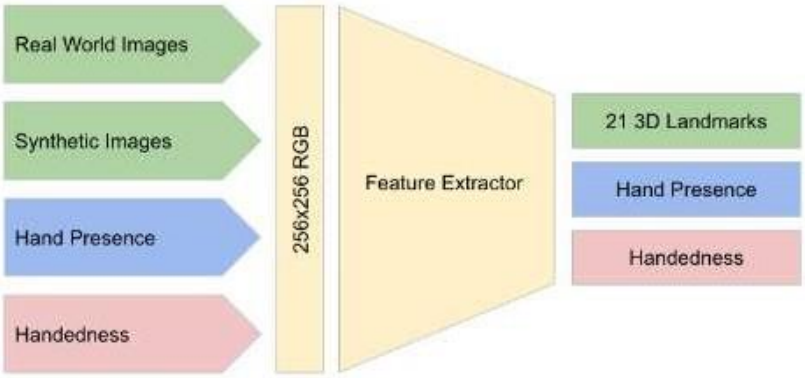

Figure 3. Hand landmark model architecture (Zhang et al., 2020)

Figure 4. The 21 Landmark coordinates (Zhang et al., 2020)

The 21 points corresponding to the hand landmarks are set accurately as learnt from the multiple datasets. This can be seen in Figure 4. The introduction of relative depth exponentially increases the possibility that the employed model detects and understands gestures, while evading the issues caused by occlusion. The threshold score or the hand flag that triggers the tracking reset is included in this model. When two hands are located in the frame, a certain feature would determine which is the left and which the right hand.

This model has different versions that can be used based on the hardware availability at the end-user side. GPU intensive versions as well as a lighter version that would rely on the existing CPU power in a more real-time sense while not compromising on the accuracy of the model also exist. In the highest-level perspective, the entire system can be understood as a directed graph with multiple subgraphs. These graphs are made up of modular components called Calculators. Calculators perform tasks that involve manipulation, cropping, rendering and computations on data. The MediaPipe graphs consists of two subgraphs, related to palm detection and hand landmark plotting. Palm detection is run only as and when needed, thereby saving GPU power. The confidence metric in the hand landmark module will determine whether to run the previous model or not.

2.3. Left-hand module

The left-hand module involves creating the dataset of hand gestures that correspond to each of the different chords that can be played on a guitar. The scope of the program limits it to only the major chords. Then a CNN model is trained on this dataset to be able to accurately predict which chord gesture is made by the left hand.

2.3.1. Dataset creation

The dataset is recorded and created by running a program that creates folders on the local system for each of the 7 major chords A, B, C, D, E, F and G. In order to do this, a bounding box is first defined around the hand in focus and the contour lines are drawn to identify each of the landmark coordinates and their connections using the MediaPipe framework. A rectangle box with the hand fully inside is then captured from the video feed and processing is done on it. The colors are stripped from the image to make it grayscale with the lines on the hands in white making it seem like a skeletal representation of certain gestures. The sample images created for all cords are shown in Figure 5. 1200 images were captured for each chord and resized to a predefined dimension. Therefore, the dataset consists of 8400 images.

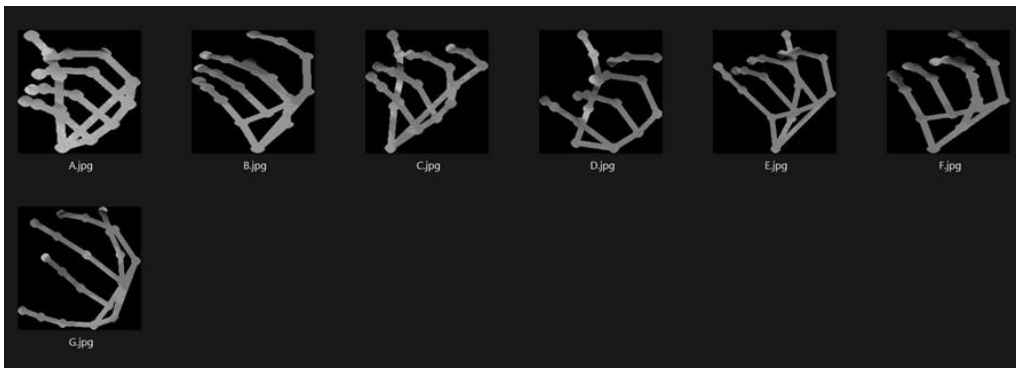

Figure 5. Sample images created for all chord

Heterogeneity was incorporated in the images for each chord by recording them in two different ways:

1. While holding an actual guitar in the hand and articulating the fingers in the position that indicates the gesture for a particular chord.
2. Making the chord-playing gestures in the most accurate way that cords could be played without the aid of a guitar.

These images were recorded using Python and computer vision functions in order to render them as a set of connector points along with the lines joining them. Inconsistencies in recording the dataset were observed when using different lighting and introducing noise in the background. This brings about the idea of further increasing entropy by using different hand types along with a dynamic environment, thereby making the system more robust. A great computational power would be required in order to make this come to fruition.

2.3.2. Chord detection model training

A Convolutional Neural Network (CNN) (Li et al., 2022) model is trained on the 'chords' directory housing around 8400 images of all the different hand gestures. The first layer of the CNN model is a Conv2D layer with 32 filters, each of the size 5x5. This layer learns spatial features from the input image, which has a shape of (200, 200, 1), indicating a 200x200 grayscale image. The activation function used is Rectified Linear Unit (ReLU), which introduces non-linearity and helps the model learn complex patterns. Following this is a MaxPooling2D layer, which applies max pooling to reduce the spatial dimensions of the feature map. It uses a 2x2 pooling window with a stride of 2, halving the height and width of the feature map. Pooling reduces computational complexity and helps mitigate overfitting.

The second Conv2D layer has 64 filters, also with a kernel size of 5x5. This layer extracts higher-level, more complex features compared to the initial convolutional layer. After this, another MaxPooling2D layer is implemented, but with a larger pooling window of the size 5x5. This significantly reduces the spatial dimensions, making the feature maps more abstract. Next, a Flatten layer converts the 2D feature maps into an 1D vector. This transformation prepares the data for the fully connected layers.

The first Dense layer, which is fully connected, consists of 1024 neurons and uses the ReLU activation function. This layer learns high-level representations of the features extracted by the previous convolutional and pooling layers. A Dropout layer is applied after this with a dropout rate of 0.6, meaning 60% of the neurons are randomly turned off during training. This helps regularize the model and prevents overfitting by reducing the co-adaptation of neurons. Finally, the output layer is a Dense layer with 7 neurons, corresponding to the 7 chord classes (representing different guitar chords). It uses the softmax activation function, which outputs a probability distribution over the 7 classes, ensuring that the sum of all outputs equals 1. The model is compiled using the categorical cross-entropy loss function, which is suitable for multi-class classification, and the Adam optimizer, known for its efficiency in training neural networks.

2.3.3. Chord identification

The live video feed annotates the left hand with the connectors and hand landmark points and then sends each frame as an image after performing the same processing that was done on the images during dataset creation to the server hosted on the cloud where the trained model performs a prediction with regard to the probability that a given hand gesture belongs to a certain chord class.

2.4. Right-hand module

The right-hand module involves velocity detection when playing a strumming pattern on the strings. A novel velocity calculation technique is presented in this paper.

2.4.1. Velocity calculation technique

The hitboxes for the velocity calculation method are shown in Figure 6. The horizontal dotted line shows the center of the screen horizontally. The vertical dotted line shows the center of the screen vertically. The left portion of the screen is used for identifying the chords played by using the left hand. The right portion of the screen shows the hand positions while strumming. By tracking the 12th hand landmark, which corresponds to the tip of the middle finger, we determine when the finger crosses a pair of horizontal hitboxes positioned near the midpoint from the bottom of the screen. The timestamp of the event is obtained when the point crosses the top horizontal line and similarly the timestamp of the event is recorded when it crosses the bottom horizontal line. The time it took the hand to pass between these two hitboxes is inversely proportional to the speed of the strumming. Using this time value, one can decide to play a strum sound with a higher distortion that gives a buzzing effect or a softer strumming sound as when a guitar is strummed gently.

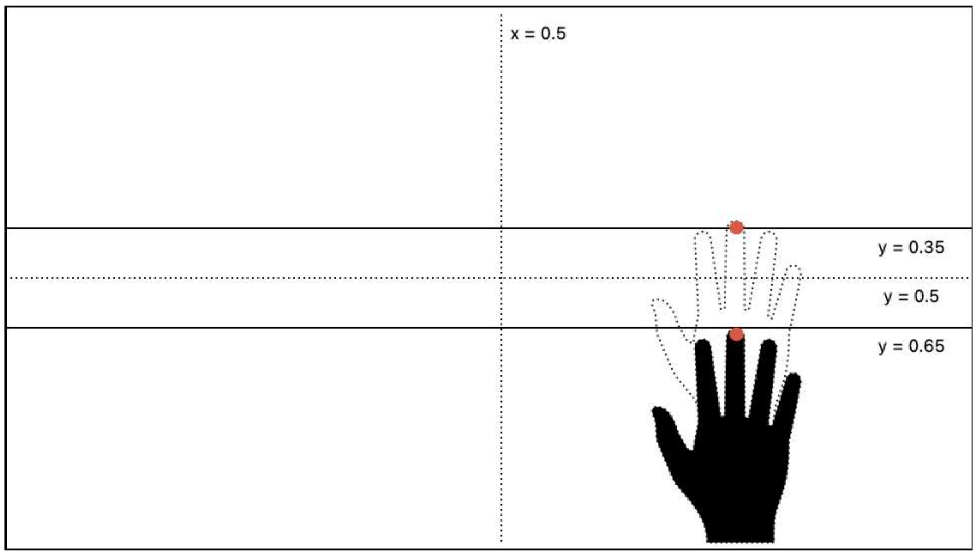

Figure 6. The placement of hitboxes for sensing the strumming

2.5. Hosting the model on the Cloud

In order to abstract the prediction using the model from the user's system, a Virtual Machine (VM) on GCP was set up (Beulah et al., 2022). This virtual machine supports 2 concurrent threads simultaneously (this can be upgraded with an upgrade in infrastructure) and has a memory of 4 GB. The Secure Shell protocol (SSH) was used for connecting with the VM with a pre-configured default key and the virtual machine's terminal was accessed via a window from the employed browser. All the VM operations were carried out through this command line interface. The model was mounted onto a Cloud Storage bucket on GCP. The model is a custom one which was explained in subsection 2.3.2, it was uploaded as a "H5" file and works as a regular Keras model would. The Cloud Storage bucket was referenced by the employed VM to run prediction tasks on

the received skeletal images. The firewall rule was added to accommodate HTTP POST requests from the client program running on the user's system.

The client sends the skeletal image of the left hand's gesture when the strum is triggered, after processing it to the desired format, same as for the images used in the training/ validation dataset. This image once received by the cloud server is passed through the model to receive a prediction. This prediction is sent as a response to the HTTP POST and it is in the JSON format, which is unpacked in the client program. It also allows one to monitor what chords the clients are playing more often for analytical purposes.

2.6. Guitar sound bank

In the final step of the system, the classified chord and strum speed are used to map and play the corresponding sound. To maintain the authenticity of the sound, the sounds were not downloaded from an external source but instead recorded from a guitar with a cherub clip acoustic mic. Reaper is a Digital Audio Workstation (DAW) application that offers recording, editing, mixing and processing functionalities. After setting up the environment for recording with a DAW - Reaper and an Audio interface - iRig that connects the guitar to the Macintosh computer, the sounds were processed and stored in a .wav format within a directory named 'chord_sounds'.

This sound bank has been expanded to accommodate two different levels of strumming based on strumming velocity. For example, A_fast.wav and A_slow.wav were recorded for the A chord to represent both the fast and slow speed, respectively. To further improve the sound bank, it would be beneficial to include a variety of minor chord sounds, along with individual chord plucking sounds in addition to strumming.

3. Results and discussion

This section presents the findings with regard to the specific modules of the whole system and then those related to the entire application.

3.1. The conducted experiments

The conducted experiments can be understood by observing them step-by-step in a modular manner. The whole system has been broken down into units and each unit has been tested diligently and finally the whole system too, in order to ensure the seamless working of the application.

The first stage involves the left-hand chord prediction by the trained machine learning model Tolentino et. al. (2019). On the left half of the screen the position and gesture of the player`s hand is tracked. While running the trained model, the gesture is recognized and the chord class corresponding to that gesture is classified by the system and further mapped to the respective chord value. The output of this module for the sample chords played by left hand is shown in Table 1.

It can be seen that the prediction is made accurately at each step, even changing the displayed chord class and chord value when the left-hand gesture changes. The next stage involves localizing the landmark point 12 as defined through the MediaPipe Hands solution on the right side of the screen. On the right half of the screen, as the strumming point moves from the top half of the screen to the bottom half, the program identifies the regions where the landmark is above or below the strumming region and also the instant(s) of motion indicating the strumming.

Table 1. Left-hand chord detection

The Played Chord	The Predicted Chord	Time taken
F	F	91ms
G	G	92ms
B	B	96ms
A	A	90ms
C	C	92ms

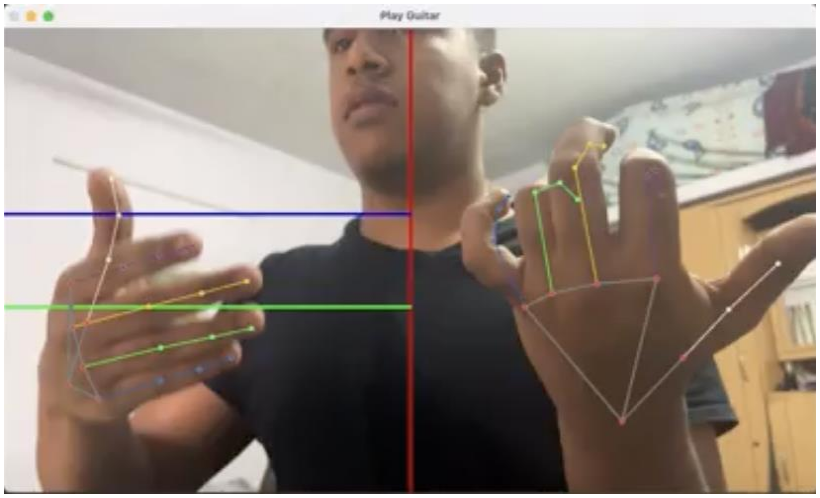

Figure 7. The Virtual Guitar System on the cloud

When the left-hand is held up and the right-hand strumming point reaches the bottom of the horizontal strumming line, the chord value at that instant is predicted by the model and this value is recorded and sent to a cloud server. The working of both the left and right side of asynchronously is observed. When the strumming point is reached, the chord value at that instant is predicted by the model and the time spent in the strumming region is flagged as fast or slow. The server then accesses the created sound bank to play the respective sound.

3.2. Performance analysis

The performance analysis was done by continually checking the chord that is played when a particular chord is shown to the camera. Each chord is tested 25 times and the total number of successful predictions is tallied out of the number 25. The ground truth in this experiment lies in the knowledge of the guitar player conducting the experiment regarding which chord is to be held while noting the observations. The confusion matrix obtained by doing this experiment is included in Table 2. The performance analysis based on the confusion matrix is presented in Table 3.

Table 2. Confusion Matrix

		Predicted Chord Class						
Actual Chord Class		**A**	**B**	**C**	**D**	**E**	**F**	**G**
	A	20	1	0	0	1	1	0
	B	1	21	1	1	1	0	2
	C	0	1	22	0	0	0	0
	D	2	0	0	21	0	1	1
	E	0	1	2	0	23	1	1
	F	1	1	0	1	0	22	0
	G	1	0	0	2	0	0	21

Table 3. Performance metrics based on the confusion matrix

Class	**Accuracy (%)**	**Precision**	**Recall**	**F1-Score**
A	95.43	0.87	0.80	0.83
B	94.29	0.78	0.84	0.81
C	97.71	0.97	0.88	0.92
D	95.43	0.84	0.84	0.84
E	96	0.82	0.92	0.87
F	96.57	0.88	0.88	0.88
G	96	0.88	0.84	0.86

Although the employed model has a great accuracy, the results obtained when running the model are not satisfactory due to the external environment being dynamic in nature. The chord playing gestures being held up, while correct, are not being tracked accurately. The proposed AirStrum system achieved an average accuracy of 95.92%.

3.3. User insight analysis

A brief preliminary survey related to the perception on the utility and usability of the AirStrum system was conducted on a uniform pool of 8 guitarists whose skill levels ranged from the beginner to the professional level. They were asked a series of 6 agree/disagree questions and their insights were recorded and tabulated. The overall reception of this project was satisfactory, and the scope for improvement was shared with us. The survey questions and its results are shown in Table 4.

Table 4. The survey questions and the obtained results

Survey Questions	**Agree (%)**	**Disagree (%)**
I would use this app for regular practice	25	75
It is convenient to use this app in its current state	62.5	37.5
It can effectively replace a guitar	25	75
I have come across similar guitar virtualization tools on the market	75	25
I think this application has market potential	75	25
I would recommend this app to a friend	87.5	12.5
I believe this app provides a realistic guitar-playing experience.	62.5	37.5
Average	**58.9**	**41.1**

4. Conclusion

A responsive and real-time system was developed based on an architecture encompassing the MediaPipe framework, OpenCV library and a Google Cloud platform. The performance analysis showed that the AirStrum system made a step forward towards better equipping the guitar-playing community with easily accessible guitar simulators. The conducted experiments reveal that AirStrum garnered a positive feedback rate of 58% following the survey which was carried out. Additionally, the proposed AirStrum system attained an impressive accuracy of 95.92% when used in a well-lit environment with a stable and high-definition camera. AirStrum also serves as a platform to learn the various chord-playing gestures and the accompanying sounds for the respective chords.

The implementation of the proposed method was carried out on a small scale and on hardware that was restricted in terms of its computational ability. Due to this the stability and consistency of the developed solution was not very strong and would waver easily with small changes in camera positions when practically used on an everyday computer system. Further work in this direction could focus on creating a more user-friendly and sustainable product built on the basis of the work that has been detailed in this paper.

REFERENCES

Azuma, Y., Miyao, H. & Maruyama, M. (2022) A Guitar Training System for Beginners Using a Mixed Reality Device and a MIDI Guitar. In: Stephanidis, C., Antona, M. and Ntoa, S. (eds.) *HCI International 2022 Posters - 24th International Conference on Human-Computer Interaction, HCII 2022, Virtual Event, 26 June – 1 July 2022* (part of the book series *Communications in Computer and Information Science* (*CCIS*, vol. 1582)). Cham, Springer International Publishing. pp. 19-26.

Bering, S. R. F. & Famador, S. M. W. (2016) Virtual Drum Simulator Using Computer Vision. In: *The 4th IIAE International Conference on Intelligent Systems and Image Processing (ICISIP2016), 8-12 September 2016, Kyoto, Japan*. Fukuoka, The Institute of Industrial Applications Engineers. pp. 370-375.

Beulah, A., Sree Sharmila, T. & Pramod, V. K. (2022) Degenerative disc disease diagnosis from lumbar MR images using hybrid features. *The Visual Computer: International Journal of Computer Graphics.* 38(8), 2771-2783. doi: 10.1007/s00371-021-02154-x.

Gomes, P., Castro, M., Fernandes, D., Soares, F., Vieira, G., Felix, J. & Nascimento, T. H. (2023) DrumsVR: Simulating Drum Percussion in a Virtual Environment Using Gesture Recognition on Smartwatches. In: *Proceedings of the 28th International ACM Conference on 3D Web Technology, 9-11 October 2023, San Sebastian, Spain*. New York, Association for Computing Machinery. Art. 17.

Li, Z., Liu, F., Yang, W., Peng, S. & Zhou, J. (2021) A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on Neural Networks and Learning Systems.* 33(12), pp. 6999-7019.

PeiQiao, L. & Fujimoto, T. (2022) XI Guitar: Development of a Slide-Type Guitar Application that Reproduces the Playing Sensation of Analog Guitar. In: Selvaraj, H. and Fujimoto, T. (eds.) *Applied Systemic Studies - International Conference on Systems Engineering* (*ICSEng 2022*) (part of the book series *Lecture Notes in Networks and Systems* (*LNNS*, vol. 611)). Cham, Springer International Publishing. pp. 64-76.

Roberts, M. (2023) *Different Types Of Guitar – An In-Depth Guide*. https://www.gear4music.com/blog/different-types-of-guitar/ [Accessed 26th June 2024].

Santiago, J. R., Samuel, S. J. & Sawn, R. (2015) Modified Virtual Air Guitar: A Concept Realized using Image Processing Techniques. *EAI Endorsed Transactions on Context-aware Systems and Applications.* 2(3), e2. doi: 10.4108/casa.2.3.e2.

Smith, D. (2023) *Measurement and Design of a Virtual Slide Guitar*. Master`s thesis. McGill University.

Sun, Y. (2023) Timbre-Based Portable Musical Instrument Recognition Using LVQ Learning Algorithm. *Mobile Networks and Applications*. 28, 2171-2181. doi: 10.1007/s11036-023-02174-y.

Tamani, J. E., Christian, J., Cruz, B., Cruzada, J. R., Valenzuela, J., Chan, K. G. & Deja, J. A. (2018) Building guitar strum models for an interactive air guitar prototype. In: *Proceedings of the 4th International Conference on Human-Computer Interaction and User Experience in Indonesia, CHIuXiD'18, March 23-29 2018, Yogyakarta, Indonesia*. New York, Association for Computing Machinery. pp. 18-22.

Tolentino, C. T., Uy, A. & Naval, P. (2019) Air drums: Playing drums using computer vision. In: *International Symposium on Multimedia and Communication Technology (ISMAC), 19-21 August 2019, Quezon City, Philippines*. IEEE. pp. 1-6.

Zhang, F., Bazarevsky, V., Vakunov, A., Tkachenka, A., Sung, G., Chang, C.-L. & Grundmann, M. (2020) MediaPipe hands: On-device real-time hand tracking. In: CVPR Workshop on Computer Vision for Augmented and Virtual Reality, 15 June 2020, Seattle, WA, USA. pp. 4321-4325.

Beulah ARUL is an Assistant Professor at the Department of Computer Science and Engineering of Sri Sivasubramaniya Nadar College of Engineering in Chennai, India with over 19 years of experience in teaching. Dr. Beulah's research interests include Image Processing, Computer Vision, Machine Learning, and Deep Learning Techniques. She has been successful in guiding several internally funded projects and her work has been recognized through numerous publications in reputed international journals and conference proceedings.

Shashank PANDA obtained his Bachelor`s Degree in Computer Science and Engineering from Sri Sivasubramaniya Nadar College of Engineering in Chennai, India where he was part of several technical clubs such as the Google Developer student club, the ACM student chapter and the SSN Coding club, playing the role of Technical Head in all three clubs. He is currently working as a Full stack Engineer at Fidelity Investments in Chennai. He is always thinking of ways to use technology for solving everyday problems in people's lives and he enjoys crafting solutions for these problems along with other like-minded innovative thinkers.

Tushar NAIR is an Associate SW Systems Engineer at Extreme Networks based in Bangalore, India. He earned his Bachelor`s degree in Computer Science and Engineering from Sri Sivasubramaniya Nadar College of Engineering where he found his calling in the tech space by attending various workshops, seminars and building projects. He has a keen interest in technology, software and dance. He is a creative person and loves to combine his artistic interests and his skills as a developer to discover new ways to combine the two.