I.J. Information Technology and Computer Science, **2025, 5, 12-24**
Published Online on October 8, 2025 by MECS Press (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2025.05.02

Hand Gesture-controlled 2D Virtual Piano with Volume Control

Vijayan R.
School of Computer Science Engineering and Information Systems (SCORE), Vellore Institute of Technology (VIT), Vellore, 632 014, India
E-mail: rvijayan@vit.ac.in
ORCID iD: https://orcid.org/0000-0002-1164-7569

Mareeswari V.*
School of Computer Science Engineering and Information Systems (SCORE), Vellore Institute of Technology (VIT), Vellore, 632 014, India
E-mail: vmareeswari@vit.ac.in
ORCID iD: https://orcid.org/0000-0002-2768-943X
*Corresponding author

Sarathi G.
School of Computer Science Engineering and Information Systems (SCORE), Vellore Institute of Technology (VIT), Vellore, 632 014, India
E-mail: sarathi.g2021@vitalum.ac.in
ORCID iD: https://orcid.org/0009-0002-9159-9305

Sathya Nikethan R. V.
School of Computer Science Engineering and Information Systems (SCORE), Vellore Institute of Technology (VIT), Vellore, 632 014, India
E-mail: sathyanikethan.rv2021@vitalum.ac.in
ORCID iD: https://orcid.org/0009-0006-6026-2362

Received: 09 May 2025; Revised: 07 July 2025; Accepted: 21 August 2025; Published: 08 October 2025

Abstract: The rise of virtual instruments has revolutionized music production, providing new avenues for creating music without the need for physical instruments. However, these systems rely on costly hardware, such as MIDI controllers, limiting accessibility. As an alternative, 3D gesture-based virtual instruments have been explored to emulate the immersive experience of MIDI controllers. Yet, these approaches introduce accessibility challenges by requiring specialized hardware, such as depth-sensing cameras and motion sensors. In contrast, 2D gesture systems using RGB cameras are more affordable but often lack extended functionalities. To address these challenges, this study presents a 2D virtual piano system that utilizes hand gesture recognition. The system enables accurate gesture-based control, real-time volume adjustments, control over multiple octaves and instruments, and automatic sheet music generation. OpenCV, an open-source computer vision library, and Google's MediaPipe are employed for real-time hand tracking. The extracted hand landmark coordinates are normalized based on the wrist and scaled for consistent performance across various RGB camera setups. A bidirectional long short-term memory (Bi-LSTM) network is used to evaluate the approach. Experimental results show 95% accuracy on a public Kaggle dynamic gesture dataset and 97% on a custom-designed dataset for virtual piano gestures. Future work will focus on integrating the system with Digital Audio Workstations (DAWs), adding advanced musical features, and improving scalability for multiple-player use.

Index Terms: 2D Virtual Piano, Hand Gesture Recognition, Mediapipe, OpenCV, Coordinate Normalization, Bi-LSTM.

1. Introduction

1.1. Virtual Instruments

Virtual instruments [1] are software applications designed to generate sound on a computer, enabling users to

compose music without relying on physical instruments. These tools can replicate the sounds of real instruments, such as classical pianos and orchestral instruments, or digitally emulate analog synthesizers to produce similar tones. Depending on the user's needs, virtual instruments can operate either as plugins within Digital Audio Workstations (DAWs) or as standalone software.

Virtual instruments are typically categorized into two types: software synthesizers and software instruments. Software synthesizers generate sound through algorithms and mathematical computations, starting with a digitally created fundamental waveform that serves as the basis for sound design. Users can then shape and manipulate the sound further through an interface. In contrast, software instruments, such as virtual pianos, rely on pre-recorded samples of real instruments stored in sound libraries and are controlled using hardware components like keyboards, mice, or MIDI controllers.

1.2. Hand Gesture Recognition using Computer Vision

The Industrial Revolution [2] catalyzed the rise of machines, fundamentally reshaping labor and manufacturing processes. Despite these changes, human operators remained necessary to control machines using input devices like levers, pedals, and hydraulic systems. With the development of computers, human-machine interaction (HMI) evolved to rely on keyboards and mice as primary interfaces. As computers became smaller and more portable, other input methods emerged.

One such method is hand gesture recognition (HGR) [3], which uses gestures as a natural means of communication. Over time, HGR has been applied in areas like virtual reality, gaming, healthcare, and education. While the accuracy of HGR systems depends on the acquisition methods and algorithms used, they offer a promising alternative to traditional input devices. As the demand for more intuitive human-computer interactions increases, researchers have turned to computer vision techniques to enhance HGR technology.

A vision-based HGR system typically consists of three phases: detection, tracking, and recognition [4]. During the detection and tracking phases, the system isolates the hand from its surroundings and records its movements. The information is then processed and analyzed using machine learning algorithms to classify gestures. The precision of these systems largely depends on the tools used to collect the data [5]. RGB cameras are widely used in HGR systems to capture hand movements through video frames or images. These cameras are readily available on devices like smartphones and laptops. For greater accuracy, some systems use RGB-D (RGB-Depth) cameras or motion sensors, such as Kinect and Leap Motion, which can detect depth and track movements more precisely. These tools enhance the reliability of HGR systems and their ability to interpret gestures accurately.

1.3. Current Challenges in Gesture-based Music Control

Virtual instruments often require intermediary hardware like a mouse, keyboard, or MIDI controller for interaction. Professional musicians generally prefer MIDI controllers for their responsiveness and functionality. However, quality MIDI controllers are expensive, require physical space, and demand technical proficiency, which may discourage novices. As virtual instruments increasingly shift toward more intuitive interaction methods, gesture recognition has emerged as a promising alternative. Over the past decade, researchers have explored hand-gesture control to make music production more accessible and user-friendly.

Despite advancements in gesture recognition, external hardware like motion tracking sensors remains a common requirement. Devices such as Kinect and Leap Motion provide precise hand tracking but are often cost-prohibitive for hobbyists and aspiring musicians, limiting their adoption in home studios. In contrast, standard RGB cameras, widely available in laptops and smartphones, present a cost-effective alternative for beginners. While these systems may lack the precision of specialized sensors, recent advancements in computer vision and deep learning—particularly tools like Google's MediaPipe—have significantly improved their accuracy, making them viable options for gesture-driven music interaction in everyday environments.

RGB cameras, however, are not without limitations. Poor lighting conditions and occlusion can affect their performance. Yet, touchless systems offer several advantages, including enhanced convenience and hygiene—an increasingly important consideration in light of global health concerns. These benefits make gesture-based systems appealing not only for their cost-effectiveness but also for their practicality in diverse scenarios.

This work addresses the challenge of making gesture-controlled virtual instruments accessible without relying on costly hardware like MIDI controllers or motion tracking sensors. Using a standard laptop webcam with Python's OpenCV library and Google's MediaPipe, our system enables real-time hand gesture recognition. Key features include piano keys, volume control, octave and instrument changes, and sheet music generation. By normalizing hand landmarks, the system ensures consistent performance across varying camera resolutions and hand sizes, detecting finger-tapping gestures to trigger corresponding notes or adjust volume with precision. The proposed system eliminates the need for specialized hardware, leveraging computer vision techniques to offer an affordable, user-friendly solution for real-time hand gesture recognition in virtual music environments.

The remainder of this paper is structured as follows: Section II reviews related literature on virtual piano and volume control systems. Section III details the operation of the virtual piano, volume control, and other features of the system. Section IV presents results, discussing datasets, the BiLSTM neural network, and the use of wrist-normalized landmark coordinates. Section V concludes the paper and outlines future research directions.

2. Related Works

2.1. Virtual Piano

Several approaches have been developed to create virtual piano systems. Ref. [6] proposed a method utilizing a sheet of paper imprinted with the seven major notes of a piano octave. The position of each key is determined by selecting its coordinates on the OpenCV window, and the user's input is reflected in a Pygame window. Using a laptop webcam, the authors employed contour detection to identify fingers accurately. The finger recognition algorithm achieved an accuracy of 98.15% when the finger was in the central portion of the key.

A performance analysis conducted by [7] revealed that utilizing the Mediapipe Hands for playing on a printed piano sheet significantly outperforms both leap motion and deep learning approaches. In the deep learning approach, two datasets comprising 25,000 RGB images and 5,000 RGB-masked images were used for training and validating the models. The Convolutional Neural Network + U-net model architecture achieved a categorical accuracy of 99.3% on test data. However, due to dataset limitations, users could only play one note at a time. The Leap Motion Controller was found unsuitable for the printed piano due to inconsistencies in identifying the paper's vertices. During the calibration of the 2D and 3D vertices of the printed piano paper, the Mediapipe library had a lower mean distance error than the Leap Motion Controller.

An unconventional approach to virtual piano systems involves the use of an inverted monocular camera setup [8] to detect fingertips positioned on a piano. The camera is placed on the floor, beneath a transparent plate on which the piano keys are outlined. Color threshold segmentation is applied to detect the fingertips. Additionally, the area covered by the fingertips is calculated in real-time to determine the timbre and volume of the keys. Experimental results conducted by the authors demonstrate that the fingertip detection algorithm performs successfully even when multiple fingers or hands are involved.

To provide an immersive experience, [9] proposed an augmented reality-driven piano designed for beginners. By leveraging Google's Mediapipe Hands and OpenCV's ARUco (Augmented Reality University of Cordoba) markers, the authors achieve a keypoint accuracy of 81%. Utilizing projective transformation, a piano image is augmented onto the paper. To distinguish hands from the augmented reality piano, Otsu's method for binary segmentation is employed. A key triggers sound upon detecting an increase in the vertical coordinate of the piano texture space. Notably, the AR piano generates only a single note at a time. A leap motion-based 3D piano, integrated with electromyography (EMG) sensors, provided velocity-sensitive note playing [10]. Using Blender, the authors created a two-octave visual model of the piano, which was then imported into Unity3D to serve as its virtual environment. A Leap Motion Controller was integrated to track the player's hands. The notes' volume was adjusted based on the EMG readings. Testing the piano with a mix of experienced and novice players, the model achieved a mean accuracy of 89.09% with acrylic panel support and 85.45% without it.

The researchers [11] achieved a real-time accuracy of 90.8% by utilizing long short-term memory (LSTM) networks. RexNetV1 and MobileNetV1 networks were used to detect skeletal key points of the player's hands and body. A 3x3 matrix-shaped square was displayed on the player's right-hand side for switching octaves and adding accidentals to a note. Left-hand gestures were employed for changing instruments, controlling pitch bend, adjusting note duration, and amplifying frequencies. However, the system restricts chord playing, which could pose a limitation for upper-beginner players.

Virtual reality (VR) setups have been explored extensively, with systems like "TapID" [12] using inertial sensors to detect finger taps. In this system, a VR headset provides a piano interface for user interaction. TapID's inertial sensors detect finger tap events, while a neural network classifier identifies which finger initiated the tap. To evaluate the system, 18 participants were recruited. The results indicate that tap event detection achieved an F1 score of 0.996, while finger identification yielded a cross-person identification accuracy of 0.87. However, the system has not been optimized for multi-touch events from a single hand, preventing users from playing chords on the piano interface. In [13], the authors developed a virtual piano, facilitating gesture recognition using a Leap Motion sensor fixed on an HTC Vive helmet. The target image's primary color is extracted and smoothed, followed by color space conversion and morphological tone reverse mapping. The transformed image is then matched using K-means clustering algorithms. The experimental results, based on 400 data groups collected for four gesture movements, demonstrate a recognition rate exceeding 88%.

Other methods, such as mmWave radar technology, have been applied for keystroke detection on printed pianos, achieving high accuracy rates. "mmKey" [14] uses a transmitter array with one antenna and a receiver array with 32 antennas, arranged in a 6×6 layout. By processing and analyzing the reflected signals, the authors demonstrated how mmKey detects, distinguishes, and locates keystrokes. In practical tests involving printed piano keyboards, mmKey achieved an overall accuracy rate of 99.12%. It also maintained high accuracy levels of over 92% for double-key keystrokes, although accuracy diminished with proximal key presses or simultaneous keystrokes.

"MuGeVI" [15], a computer vision-based interactive virtual musical instrument, utilized MediaPipe and Open Sound Control for real-time music performance using hand gestures. The authors leverage MSP's visual environment for audio processing and MIDI manipulation. In performance mode, hand gestures trigger note playing, while accompaniment mode allows for real-time chord control. Control mode enables transposition and volume adjustment, and Audio Effects mode provides real-time audio effects like wah-wah. User feedback highlights MuGeVI's stability, functionality, novelty, and convenience, with suggestions for improved rhythm flexibility and mobile device compatibility.

2.2. Volume Control

The system outlined in [16] simulates a volume slider using index finger and thumb position data. By processing live camera feeds with OpenCV and Mediapipe's Hand Tracking model, the system achieved an average of 26 fps. Volume adjustments are made in real-time by analyzing the distance between the thumb and index finger, utilizing "PyCaw" for audio control. The authors found that the frames per second may fluctuate due to various factors, such as lighting conditions and camera distortions.

In [17], the authors propose a gesture recognition system based on the Gaussian Mixture Model (GMM) for background reduction and VGG16 (Visual Geometry Group 16), for gesture classification. Background subtraction with GMM isolates hand gesture features, which are then pre-processed, resized, and fed into the VGG16 architecture. The trained model is applied to real-time webcam images, classifying gestures into ten categories corresponding to specific actions, such as volume control, navigation, and system commands. Two Convolutional Neural Networks (CNN) architectures are used: a simple CNN with max-pooling achieves 93% validation accuracy, while VGG16 improves accuracy to 98%. Specifically, F1 scores for volume-down and volume-up gesture recognition are 0.96 and 1.00, respectively.

A computer vision method discussed in [18] operates on template matching, enabling video control through hand gestures. Hand gesture recognition is particularly emphasized, utilizing OpenCV to capture frames from video files or a live stream. The authors use "PyAutoGUI", to enable the translation of recognized gestures into automated actions. Validation procedures demonstrate a 94% success rate in good lighting conditions with full luminous intensity and 71.1% efficiency in poor lighting conditions with reduced luminous intensity. This work [19] presents a system for hand gesture volume control, where the input device records hand gestures using a USB camera, followed by processing to identify hand contours and marks. Subsequently, the system computes the volume control percentage based on the recognized hand gestures. To implement the hand gesture volume control system, the authors develop an application on a Raspberry Pi 4 platform using the Python programming language. The Raspberry Pi 4 features general-purpose input/output (GPIO) pins for electronic component control and USB ports for device connectivity, including a USB camera. The authors observed that greater distances between hand gestures and the camera result in reduced accuracy, especially with lower resolutions.

While significant advancements have been made in gesture recognition accuracy and system responsiveness, challenges remain in ensuring accessibility and reducing hardware dependency. This work aims to improve the user experience of virtual instruments by integrating an effective hand gesture recognition system utilizing a simple RGB camera. To detect and process the user's palms, MediaPipe's Hands framework is used. The hand landmark coordinates are then normalized based on the wrist landmark and scaled to a uniform range, allowing for real-time gesture control of a 2D virtual piano. This approach eliminates the need for costly depth-sensing cameras or motion sensors.

3. Methodology

3.1. Graphical User Interface

Python's Tkinter is a standard GUI (Graphical User Interface) toolkit used for creating desktop applications with components such as buttons, menus, and text fields. The user interface for the virtual piano system is designed using the Tkinter toolkit. The main interface includes clear instructions on how to operate the virtual piano. Upon clicking the "PLAY HERE" button, a new window opens, displaying a graphical representation of the virtual piano with interactive piano keys. In addition to the piano display, this window includes controls for volume adjustment, octave selection, instrument sounds, and sheet music generation. Users can also adjust the threshold levels for each finger to optimize the recognition of finger-tapping gestures. This feature ensures smooth interaction despite varying hand sizes or camera resolutions.

3.2. Mediapipe Hands

A standard laptop webcam is used to capture video frames, which are subsequently processed for hand detection. MediaPipe's [20] palm detection model is employed to identify hands within the frame. For each detected hand, 21 key landmarks are extracted, as illustrated in Figure 1. Each landmark consists of three coordinates: x, y, and z, where x and y are normalized to the range [0.0, 1.0] based on the image's width and height, while z indicates the depth relative to the wrist. However, only the x and y Coordinates are utilized in further processing, as the z value does not provide accurate depth information relative to the camera.

To ensure consistency, the coordinates of each landmark are adjusted relative to the wrist landmark, which is treated as the origin. This adjustment repositions the coordinate system so that all hand positions are referenced uniformly. Mathematically, this transformation is expressed as Eq. 1 and Eq. 2:

$$x_i' = x_i - x_0 \tag{1}$$

$$y_i' = y_i - y_0 \tag{2}$$

where (x_0, y_0) are the coordinates of the wrist landmark, and (x_i, y_i) are the coordinates of the i-th landmark. To preserve the aspect ratio during scaling, the maximum absolute values of the adjusted x' and y' coordinates are computed. The larger of these values, denoted as "max_value", is used as the scaling factor. The adjusted coordinates are then normalized using Eq. 3 and Eq. 4:

$$\hat{x}_i = x_i' \div \max\_value \tag{3}$$

$$\hat{y}_i = y_i' \div \max\_value \tag{4}$$

This normalization process scales the coordinates into the range [-1,1], ensuring consistency across varying hand sizes and positions within the frame.

Finally, the normalized coordinates of all 21 landmarks are flattened into a single list, which serves as the input for subsequent processing steps. This procedure [21] standardizes the representation of hand landmarks, eliminating dependencies on the hand's absolute position, orientation, or the camera's resolution. By focusing solely on the relative geometry of the hand, the model achieves robust performance without requiring depth-sensing capabilities.

3.3. Virtual Piano

The piano interface, illustrated in Figure 2, consists of a single octave containing seven major notes (C, D, E, F, G, A, B) and five minor notes (C#, D#, F#, G#, A#). Each key corresponds to a specific region on the screen, enabling interaction via designated hand movements. The system employs a live camera feed to track the user's left hand, with hand landmarks detected via MediaPipe's hand-tracking algorithm.

To approximate the distance between the user's hand and the camera, the Euclidean distance, d, between Landmark 5 (Index finger metacarpophalangeal joint) and Landmark 17 (Pinky finger metacarpophalangeal joint) is calculated. This distance, measured in pixels, is converted to a real-world measurement (in centimeters) using a predefined quadratic model. This conversion is expressed as Eq. 5:

$$D_{CM} = A \,.\, d^2 + B \,.\, d + C \tag{5}$$

where A, B, and C are calibration coefficients tailored to the camera's specific configuration and resolution. These coefficients are determined through a calibration process to ensure accurate distance measurement. To contextualize the hand's position within the camera frame, a bounding box is constructed by analyzing the minimum and maximum x and y coordinates of the detected hand landmarks. This bounding box defines the spatial boundaries of the hand, enabling precise localization of its movements on the virtual piano interface [22].

3.4. Tap Gesture Recognition Using Threshold System

The system employs a combination of normalized and raw landmark lists along with predefined sound mappings to detect finger taps on virtual piano keys. Interactions with the piano keys are determined by analyzing the y-axis coordinates of the normalized fingertip landmarks, as shown in Figure 3. Fingertip landmarks are identified within the normalized landmark list, enabling the system to extract their y-axis coordinates and spatial positions relative to the piano interface.

When a fingertip's y-axis coordinate drops below a predefined threshold—indicating a downward tapping motion—and its x-axis coordinate falls within the boundaries of a piano key, the system triggers the corresponding sound. Conversely, if the fingertip's x-axis coordinate moves outside the key boundaries and the y-axis coordinate rises above the threshold, the system updates an internal state variable. This state tracking ensures the finger has returned to its original position, preventing repeated or accidental detections of the same tap.

The system employs adjustable thresholds to accommodate variations in hand-to-camera distances, ensuring accurate detection across diverse setups. For white keys, the threshold is calibrated to identify downward finger movements over the larger, rectangular key areas. For black keys, a different threshold is applied, tailored to the smaller surface areas of these keys.

To further enhance accuracy, the system verifies the fingertip's x-axis position to confirm it lies within the spatial boundaries of the intended piano key. This combination of positional checks and threshold-based analysis minimizes false positives and improves the precision of key activations [23].

3.5. Volume Controller

The virtual piano includes two buttons adjacent to the interface for volume adjustment. The volume control module leverages a straightforward hovering mechanism, which inherently minimizes the impact of issues like noise or hand jitter. When a fingertip is detected within the circular boundaries of either button, the circle changes color to indicate activation, prompting the application to adjust the volume accordingly. The system determines fingertip positioning over the Volume Up or Volume Down button using the Euclidean distance formula. Depending on the fingertip's detected location, a predefined variable is incremented or decremented. The updated variable is then mapped to a range of 0.0 to 1.0, representing the volume levels. This mapped value is applied as the system's master volume using Python libraries pycaw

and comtypes, enabling precise control. A progress bar visually reflects the volume changes, providing immediate feedback to the user.

Since the interaction requires only a single key point (fingertip) to hover over a relatively large circular button, minor variations in hand position do not significantly affect the system's responsiveness. This simplicity ensures reliable performance without the need for advanced noise-reduction techniques or filtering. Additionally, introducing more complex methods could increase computational overhead, potentially slowing down the system, which is designed to function efficiently in real time.

3.6. Octaves, Sounds, and Sheet Music

The system supports dynamic sound management, with sound files loaded using the “mixer” function from the Pygame library and stored in a dictionary. This construction of file paths is dynamic, utilizing string formatting to incorporate the selected sound and octave. The available instrument sounds include piano, electric guitar, electric bass, and flute, each spanning seven octaves.

For generating sheet music, the system utilizes LilyPond [24], a music engraving program renowned for producing high-quality sheet music that replicates the aesthetic of traditional engraving. LilyPond supports various musical styles, including classical, modern, and complex notations. The sheet music generation process begins by iterating through the list of notes played by the user. Key features of LilyPond’s syntax, such as version information, language settings, and notes, are incorporated into the generated code. As the system processes the notes, it identifies clef changes, denoted by specific symbols like spaces, commas, and quotation marks. When a clef change is detected, the corresponding LilyPond command (e.g., \clef bass or \clef treble) is appended to the code. Once all notes are processed, the resulting LilyPond code string is used to generate a PDF file containing the musical notation, providing users with a polished representation of their performance.

While the virtual piano system performs well under standard conditions, its accuracy depends on the reliability of MediaPipe’s hand-tracking capabilities. Factors such as poor lighting, low camera quality, or suboptimal hand positioning can influence landmark detection and, consequently, the system’s responsiveness. Additionally, the application is designed for single-user interactions, assuming that only one set of hand landmarks will be detected in the camera frame at any given time.

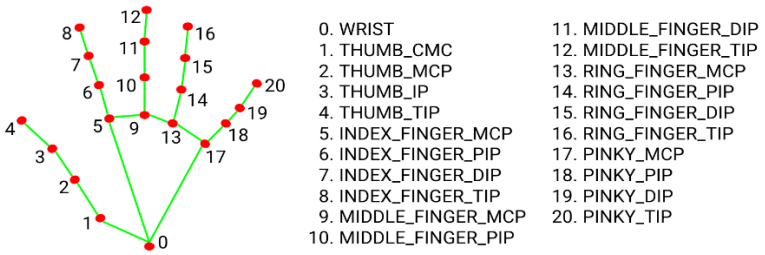

Fig.1. Hand landmarks

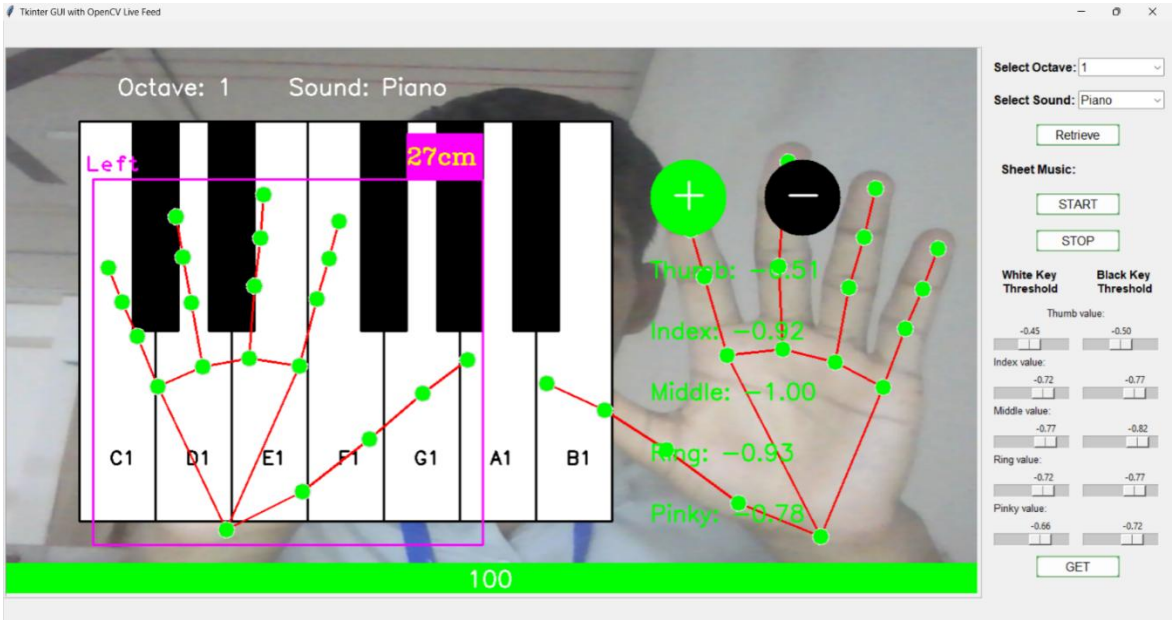

Fig.2. Virtual piano system

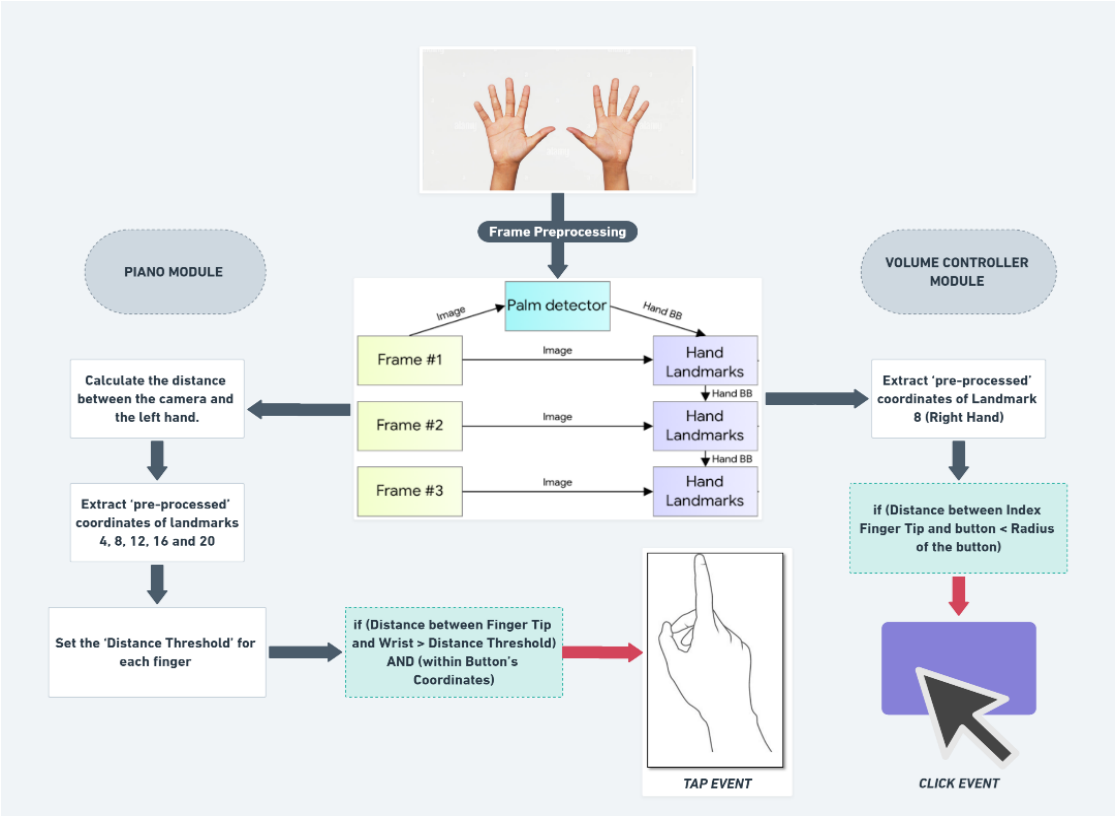

Fig.3. Processes involved in recognizing tap and click gestures

4. Results and Discussion

4.1. Datasets

To evaluate the effectiveness of the wrist-normalized landmark coordinates approach, two datasets were utilized, catering to both general-purpose and domain-specific scenarios. The first dataset, sourced from Kaggle [25], focuses on dynamic hand gestures, making it well-suited for assessing the model's robustness in recognizing diverse motion patterns relevant to virtual instrument interactions. This dataset includes five gestures: left swipe, right swipe, stop, thumbs down, and thumbs up. It is divided into a training set of 578 sequences and a test set of 88 sequences, with each sequence comprising 30 frames. Originally intended for smart TV control applications, this dataset serves as a challenging testbed for evaluating the model's adaptability to real-world scenarios beyond its original domain.

The second dataset is a smaller, custom collection designed specifically to evaluate the virtual piano application. It focuses on five dynamic gestures performed by each finger of the left hand, simulating the downward and upward motion associated with piano taps. Each finger contributes 30 sequences, with each sequence containing 30 frames. This dataset reflects the unique interaction dynamics of the virtual piano system, ensuring the evaluation aligns with the intended application.

No data augmentation was applied to the second dataset due to two key considerations. First, the natural range of motion for piano interactions was preserved to ensure accurate recognition of intended movements without introducing artificial variations. Second, the wrist-normalized coordinates approach eliminates reliance on the piano interface's fixed position on the screen. Consequently, augmentation techniques, which primarily affect pixel-level coordinates, would not significantly enhance the model's accuracy or performance.

By combining a general-purpose dataset from Kaggle with a domain-specific custom dataset, the evaluation provides a comprehensive assessment of the model's ability to handle both broad gesture recognition tasks and the precise requirements of a virtual piano interface.

4.2. BiLSTM Neural Network for Gesture Classification

To classify the hand gestures in the datasets and evaluate the effectiveness of the wrist-normalized landmark coordinates approach, a BiLSTM neural network [26] was constructed due to its ability to capture both forward and backward temporal dependencies in sequential data. LSTM (Long Short-Term Memory) models, in general, excel at processing sequential data, such as in tasks like language modeling and question answering [27]. BiLSTMs, with their bidirectional nature, allow the model to learn these dependencies more effectively than unidirectional models. This is particularly important for dynamic hand gesture recognition, where the sequence of movements is often influenced by both preceding and succeeding gestures.

The model architecture, depicted in Figure 4, comprises multiple layers. It begins with two BiLSTM layers designed to process the sequential data effectively. The first BiLSTM layer includes 64 units, which process the flattened, normalized coordinates of the 21 hand landmarks across 30 frames per gesture in both forward and backward directions. To introduce non-linearity, the ReLU (Rectified Linear Unit) activation function is used, while returning sequences to preserve temporal information. A Dropout layer with a rate of 50% follows, acting as a regularization technique to mitigate overfitting.

The second BiLSTM layer, with 128 units, refines the sequential data further, employing ReLU activation and returning sequences. This is followed by another Dropout layer to enhance regularization. Subsequently, an LSTM layer with 64 units processes the data without returning sequences, capturing higher-level temporal features [28]. Batch Normalization is applied at this stage to normalize activations, ensuring stable and efficient training.

The architecture also includes two Dense layers with 64 and 32 units, respectively, both using ReLU activation to extract essential features from the data. Finally, a Dense output layer with 5 units employs a softmax activation function to classify the gestures into one of the five categories. The model design highlights the importance of capturing temporal dependencies and mitigating overfitting, ensuring robust performance in dynamic hand gesture recognition.

Layer (type)	Output Shape	Parameter #
Bidirectional LSTM	(None, 30, 128)	54784
Dropout	(None, 30, 128)	0
Bidirectional LSTM 1	(None, 30, 256)	263168
Dropout 1	(None, 30, 256)	0
LSTM 2	(None, 64)	82176
Batch Normalization	(None, 64)	256
Dense	(None, 64)	4160
Dense 1	(None, 32)	2080
Dense 2	(None, 5)	165

Total parameters: 406,789

Trainable parameters: 406,661

Non-trainable parameters: 128

Fig.4. BiLSTM neural network model architecture

4.3. Performance Metrics

The training and testing processes were conducted on an Intel i5 processor with 8 GB RAM. As illustrated in Figures 5 and 6, the BiLSTM model, using the wrist-normalized and scaled hand landmark coordinates as input, achieved a test accuracy of 0.95 on the Kaggle dataset, and 0.97 on the custom dataset, respectively.

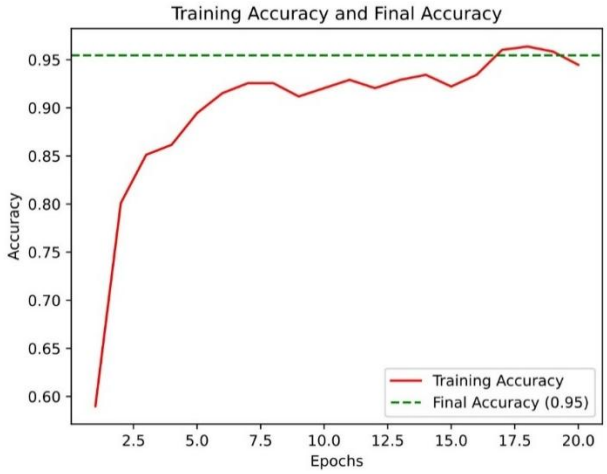

Fig.5. Training and test accuracy – kaggle dataset

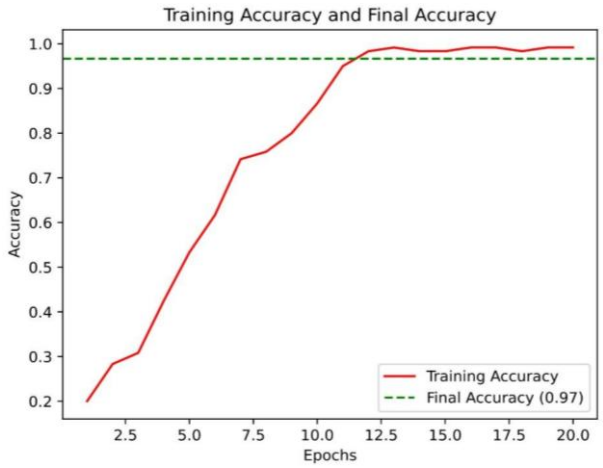

Fig.6. Training and test accuracy – custom dataset

The precision, recall, and F1 scores of using the Kaggele dataset and custom dataset are presented in Table 1 and Table 2, respectively. In both cases, most gestures exhibit high precision, recall, and F1 scores, indicating that the model performs well in recognizing the different gestures. However, certain gestures show room for improvement.

For the Kaggle dataset, the "Left swipe" and "Right swipe" gestures exhibit some confusion due to their similarity in motion. The lower recall and instances of misclassification between these two gestures suggest that their motion patterns overlap significantly, making differentiation more challenging.

In the custom dataset, the "Middle" and "Ring" finger gestures display relatively lower metrics, with the "Ring" finger gestures proving more challenging to differentiate. This is corroborated by the confusion matrix, which shows instances where "Ring" finger gestures are misclassified as "Middle" finger gestures.

Table 1. Precision, recall, and F1 score for gesture classes from the kaggle dataset

Gesture Class	Precision	Recall	F1 Score
Left Swipe	0.86	0.86	0.86
Right Swipe	0.94	0.89	0.91
Stop	1.00	1.00	1.00
Thumbs down	0.95	1.00	0.97
Thumbs up	1.00	1.00	1.00

Table 2. Precision, recall, and F1 score for gesture classes from the custom dataset

Gesture Class	Precision	Recall	F1 Score
Thumb	1.00	1.00	1.00
Index	1.00	1.00	1.00
Middle	0.80	1.00	0.89
Ring	1.00	0.80	0.89
Pinky	1.00	1.00	1.00

Figures 7 and 8 provide a visual representation of the model's performance through confusion matrices for both the Kaggle and custom datasets. The misclassification of "Left swipe" as "Right swipe" (and vice versa) highlights the model's struggle with gestures that are similar in motion or appearance. One likely explanation is the overlap in the hand's starting position when performing these gestures, which might confuse the model.

Similarly, for the custom dataset, the misclassification of "Ring" finger gestures as "Middle" finger gestures could stem from the inherent similarity in movements between adjacent fingers. This issue is particularly pronounced in the context of piano tap motions, where subtle differences in finger movements might be less distinguishable for the model.

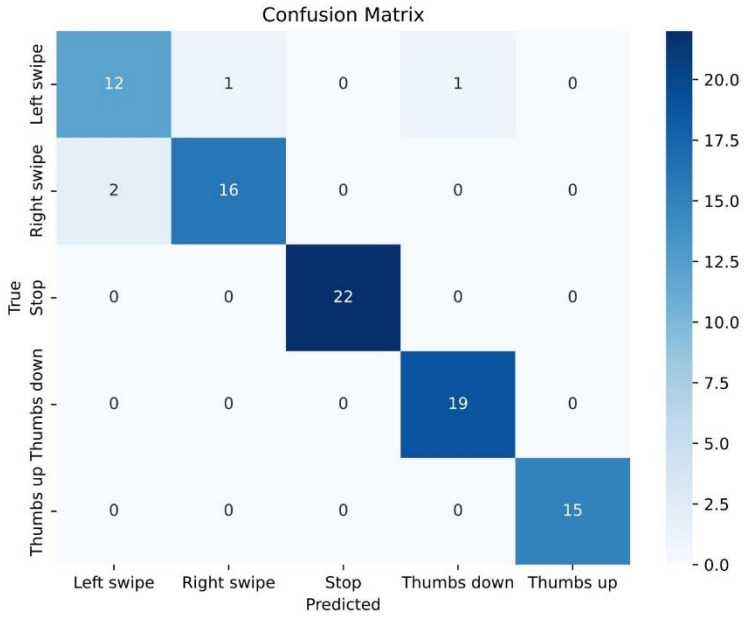

Fig.7. Confusion matrix for test set– kaggle dataset

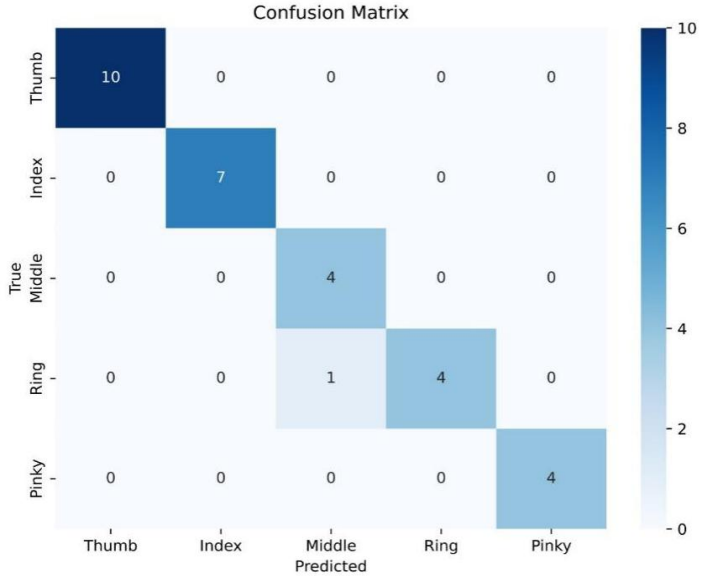

Fig.8. Confusion matrix for test set– custom dataset

4.4. Reliability Assessment of Model Accuracy

Bootstrap sampling was performed with 100 iterations on both the Kaggle and custom datasets to validate the reported accuracies. The mean accuracy, standard deviation, and 95% confidence intervals (CIs) were calculated using the following equation Eq.6:

$$CI = \left(\mu - z \cdot \frac{\sigma}{\sqrt{n}} , \mu + z \cdot \frac{\sigma}{\sqrt{n}} \right) \tag{6}$$

In this equation, μ represents the mean accuracy, z is the Z-score corresponding to the desired confidence level, in this case being 95%, σ is the standard deviation of the accuracies, and n is the number of test samples. These calculations provide a statistical measure of the reliability of the model's reported performance metrics.

The results, summarized in Table 3, demonstrate the model's reliability. For the Kaggle dataset, the mean accuracy was 95%, with a standard deviation of 0.019 across 88 test samples. The calculated 95% confidence interval was (94.6%, 95.4%), reflecting consistent performance across test instances.

For the custom dataset, the mean accuracy was 97%, with a standard deviation of 0.03 across 30 test samples. The 95% confidence interval was calculated as (95.95%, 98.1%), indicating the system's robustness when applied to domain-specific gestures. The narrow confidence intervals for both datasets suggest low variability in performance, highlighting the model's reliability and consistency across different test scenarios.

Table 3. Confidence intervals for accuracy

Dataset	Mean Accuracy (μ)	Standard Deviation (σ)	Number of Samples (n)	Confidence Interval (95%)
Kaggle	0.95	0.019	88	(0.946, 0.954)
Custom	0.97	0.030	30	(0.9595, 0.981)

4.5. Performance Profiling

A performance profiling script was implemented to evaluate the latency and memory usage of the virtual piano application. Python's "tracemalloc" library was used to monitor memory consumption, while the time library tracked execution time.

The application was developed in PyCharm 2023.3.3 (Community Edition) using Python 3.11 and tested on a system featuring an Intel(R) Core(TM) i5-10300H CPU @ 2.50GHz, 8 GB RAM, running Windows 11. The webcam resolution was set to 1280x720 at 45 fps under standard indoor lighting conditions. Table 4 summarizes the time and memory efficiency of the application. The results demonstrate that the system is resource-efficient, capable of real-time user interaction without excessive memory or time consumption.

Table 4. Performance test results

Metric	Value
Total time elapsed (After executing all modules)	50 sec (approx.)
Peak memory usage	7.28 MB

The results indicate that the system performs efficiently in terms of both time and memory utilization, showcasing its potential for seamless real-time operation in practical scenarios.

5. Conclusions

This project successfully demonstrates the potential of MediaPipe's hand landmark model in creating an interactive virtual piano system. By employing wrist-normalized and scaled landmark coordinates as input, the system achieves consistency in detecting hand positions. This normalization effectively mitigates discrepancies caused by variations in hand size, orientation, or camera setup, allowing for more accurate 2D hand gesture recognition.

The project also contributes significantly to the field of virtual piano interfaces, which often focus on replicating traditional piano-playing experiences. By incorporating advanced features such as volume control, pitch alteration, and multiple instrument sounds, this system offers a versatile musical experience that extends its utility beyond traditional applications.

Despite its successes, the system's performance is still dependent on the accuracy of MediaPipe's hand tracking. External factors, such as lighting conditions and background obstructions, can affect landmark detection. Although MediaPipe performs well under typical conditions, these limitations highlight the need for further refinement in real-world scenarios. Future work should focus on rigorous testing under variable lighting, and across different user demographics to assess the system's robustness and practical applicability.

The scalability of the system can be enhanced by expanding the range of supported hand gestures. Currently, the system recognizes basic gestures, but integrating more complex motions, such as multi-finger interactions or finger-specific gestures, would allow for more nuanced control. This could facilitate advanced features, like chord recognition, where multiple notes are played simultaneously, or real-time pitch adjustment.

Moreover, improving the system's scalability to handle multiple users simultaneously would broaden its application. For example, this could allow a group of learners or musicians to interact with the system at the same time, each playing their part on the virtual piano. To make the system even more adaptable to different user preferences, incorporating features such as voice control could provide additional ways to interact with the system. The system could also benefit from adaptive algorithms that learn and adjust to individual users' gestures and performance over time. This would ensure that the system remains responsive to different skill levels, making it more engaging and accessible for beginners while offering enough complexity for advanced users.

Looking ahead, integrating the system with Digital Audio Workstations (DAWs) could allow gesture-based inputs to be converted into MIDI or other compatible formats for real-time music composition. Conducting user studies on usability and ergonomics, coupled with feedback-driven refinement, could help transform the system into a comprehensive tool for both learning and creating music, adaptable to a wide range of users and environments.

References

[1] "What are virtual instruments? How to make digital music," RouteNote Blog, https://routenote.com/blog/what-are-virtual-instruments/, last accessed on 2024/04/25.

[2] Jain, N., Gupta, V., Temperini, V., Meissner, D. and D'angelo, E, "Human machine interactions: from past to future- a systematic literature review", Journal of Management History, Vol. 30 No. 2, pp. 263-302. 2024, doi:10.1108/JMH-12-2022-0085.

[3] Oudah M., Al-Naji A. and Chahl J., “Hand Gesture Recognition Based on Computer Vision: A Review of Techniques,” Journal of Imaging 6, no. 8:73, 2020, doi:10.3390/jimaging6080073.
[4] Qi, J., Ma, L., Cui, Z., and Yu, Y., “Computer vision-based hand gesture recognition for human-robot interaction: a review,” Complex Intell. Syst. 10, pp. 1581–1606, 2024, doi:10.1007/s40747-023-01173-6.
[5] L. Guo, Z. Lu, and L. Yao, "Human-Machine Interaction Sensing Technology Based on Hand Gesture Recognition: A Review," in IEEE Transactions on Human-Machine Systems, vol. 51, no. 4, pp. 300-309, August 2021, doi: 10.1109/THMS.2021.3086003.
[6] Thottathil, Isaac Abraham, and S. Thivaharan, "Virtual Musical Instruments with Python and OpenCV," Journal of Ubiquitous Computing and Communication Technologies 5.1: pp. 1-20, 2023.
[7] Hanuˇs, Jiˇr´ı., “Virtual piano using image processing,” Master’s thesis. Czech Technical University in Prague, Faculty of Information Technology, 2021.
[8] Wang, Yajing and Liang Song, “Virtual Piano System Based on Monocular Camera,” In Huang, DS., Jo, KH., Li, J., Gribova, V., Hussain, A. (eds) Intelligent Computing Theories and Application. ICIC 2021. Lecture Notes in Computer Science(), vol 12837. Springer, Cham. 2021, doi:10.1007/978-3-030-84529-2_10.
[9] Hiranaka, A., Grown-Haeberli, E., and Xue, K., “AR Piano Playing Using Real-Time Hand Tracking,” 2022.
[10] C. C. Hui Fen and W. N. Lim, "Virtual Piano Dynamics Enhancement with Electromyography," Innovations in Intelligent Systems and Applications Conference (ASYU), Elazig, Turkey, pp. 1-6, 2021, doi:10.1109/ASYU52992.2021.9598948.
[11] K. Shang and Z. Wang, "A Music Performance Method Based on Visual Gesture Recognition," 2022 China Automation Congress (CAC), Xiamen, China, pp. 2624-2631, 2022, doi:10.1109/CAC57257.2022.10055445.
[12] M. Meier, P. Streli, A. Fender and C. Holz, "TapID: Rapid Touch Interaction in Virtual Reality using Wearable Sensing," 2021 IEEE Virtual Reality and 3D User Interfaces (VR), Lisboa, Portugal, pp. 519-528, 2021, doi:10.1109/VR50410.2021.00076.
[13] Wang, M., “Influence analysis of piano music immersion virtual reality cooperation based on mapping equation,” Applied Mathematics and Nonlinear Sciences, 8(1), pp. 1499-1508, 2022, doi:10.2478/amns.2022.2.0138.
[14] Y. Hu, B. Wang, C. Wu and K. J. R. Liu, "mmKey: Universal Virtual Keyboard Using a Single Millimeter-Wave Radio," in IEEE Internet of Things Journal, vol. 9, no. 1, pp. 510-524, 2022, doi:10.1109/JIOT.2021.3084560.
[15] Yang, Yue, Zhaowen Wang, and Zijin Li, "MuGeVI: A Multi-Functional Gesture-Controlled Virtual Instrument," Proceedings of the International Conference on New Interfaces for Musical Expression, 2023.
[16] Eswaran, K.C.A., Srivastava, A.P., Gayathri, M., “Hand Gesture Recognition for Human-Computer Interaction Using Computer Vision” In: Kottursamy, K., Bashir, A.K., Kose, U., Uthra, A. (eds) Deep Sciences for Computing and Communications. IconDeepCom 2022. Communications in Computer and Information Science, vol 1719. Springer, Cham, 2023, doi:10.1007/978-3-031-27622-4_7.
[17] Devendra Singh, Gaurav Das, Saad Y. Sait, “Hand Gesture Recognition System using Convolutional Neural Network,” Turkish Journal of Computer and Mathematics Education (TURCOMAT). 12, 11, pp. 5609–5616, May 2021, doi:10.17762/turcomat.v12i11.6811.
[18] Mali, Chirag, et al., “Design and Implementation of Hand Gesture Assistant Command Control Video Player Interface for Physically Challenged People,” Proceedings of the 5th International Conference on Communication and Information Processing (ICCIP), June 2023, doi:dx.doi.org/10.2139/ssrn.4626576.
[19] Paniti Netinant, Yannakorn Tuaktao, and Meennapa Rukhiran, “Development of Real-Time Hand Gesture for Volume Control Application using Python on Raspberry Pi,” In Proceedings of the 2022 5th International Conference on Software Engineering and Information Management (ICSIM '22). Association for Computing Machinery, New York, NY, USA, 1–5, 2022, doi:10.1145/3520084.3520085.
[20] “Mediapipe Machine Learning Solutions,” https://developers.google.com/mediapipe, last accessed 2024/04/25
[21] Remiro MÁ, Gil-Martín M, and San-Segundo R., “Improving Hand Pose Recognition Using Localization and Zoom Normalizations over MediaPipe Landmarks,” Engineering Proceedings; 58(1):69. 2023, doi:10.3390/ecsa-10-16215.
[22] Pople, V., Kanaujia, A., Vijayan, R. and Mareeswari, V., “Face Mask Detection for Real Time Video Streams,” ECS Transactions, 107(1), p.8275, 2022, doi:10.1149/10701.8275ecst.
[23] Damkjær, E. L., and Arleth, L., ““But Wait, There’s More!”–A Deeper Look into Gestures on Touch Interfaces,” 2019.
[24] “LilyPond – Music notation for everyone,” https://lilypond.org/, last accessed on 2024/04/25.
[25] https://www.kaggle.com/datasets/imsparsh/gesture-recognition, last accessed on 2024/07/20
[26] Singh, R.P., Singh, L.D., “Dyhand: dynamic hand gesture recognition using BiLSTM and soft attention methods,” Vis Comput, 2024, doi:10.1007/s00371-024-03307-4.
[27] Ramaraj, Vijayan, Mareeswari Venkatachala Appa Swamy, Ephzibah Evan Prince, and Chandan Kumar. "Improving the BERT model for long text sequences in question answering domain," Int J Adv Appl Sci ISSN 2252, no. 8814: 8814, 2023, doi:10.11591/ijaas.v13.i1.pp106-115.
[28] D. R. T. Hax, P. Penava, S. Krodel, L. Razova, and R. Buettner, "A Novel Hybrid Deep Learning Architecture for Dynamic Hand Gesture Recognition," in IEEE Access, vol. 12, pp. 28761-28774, 2024, doi:10.1109/ACCESS.2024.3365274.

Authors’ Profiles

Vijayan R. is a professor and senior faculty member at the School of Computer Science and Information Systems (SCORE], Vellore Institute of Technology (VIT), Vellore, Tamil Nadu, India. He is a life member of the Computer Society of India (CSI) and the Soft Computing and Research Society (SCRS). He has published several national and international research articles in reputed journals and conferences. His research interests include web technologies, wireless networks, ad hoc networks, computer networks, machine learning, the Internet of Things (IoT), Mobile Computing, Cloud Computing, and Data Science.

Dr. Mareeswari V. is working as an Associate Professor at the School of Computer Science Engineering and Information Systems (SCORE), Vellore Institute of Technology (VIT), Vellore, Tamilnadu, India - 632014. She has over 21 years of academic and research experience. She received her Ph.D. in Information Technology and Engineering in the Web Service domain. She has produced several national and international articles in reputed journals, conferences, and book chapters. Her area of interest includes Medical Image processing using Machine and Deep Learning, Data analytics on Blockchain, Prediction systems on Web Service Ranking, Internet of Things (IoT), Routing on Mobile Ad-hoc networks, and E-Commerce systems. She is a life member of the Computer Society of India (CSI) and the Soft Computing and Research Society (SCRS).

Sarathi G. holds a Bachelor's degree in Computer Applications (BCA) from VIT Vellore University, India. Currently pursuing his Master of Computer Applications (MCA) at VIT Vellore, his research interests focus on gesture recognition systems, artificial intelligence in visual technologies, and cloud computing. He completed an internship at CodeClause, where he worked on projects related to Python programming and software development.

Sathya Nikethan R. V. holds a Bachelor's degree in Computer Applications (BCA) from VIT Vellore University, India. Currently pursuing his Master of Computer Applications (MCA) at VIT Vellore, his research interests focus on machine learning, computer vision, and software development.

How to cite this paper: Vijayan R., Mareeswari V., Sarathi G., Sathya Nikethan R. V., "Hand Gesture-controlled 2D Virtual Piano with Volume Control", International Journal of Information Technology and Computer Science(IJITCS), Vol.17, No.5, pp.12-24, 2025. DOI:10.5815/ijitcs.2025.05.02