Real-time Hand Motion Synthesis for Playing a Virtual Guitar

Ryan Canales
University of Florida
Gainesville, FL, USA
Clemson University
Clemson, SC, USA
r.canales@ufl.edu

Sophie Jörg
University of Bamberg
Bamberg, Germany
sophie.joerg@uni-bamberg.de

Figure 1: Four different poses generated by our hand motion synthesis pipeline.

Abstract

Concerts and performances in virtual reality are becoming increasingly popular. Accurately visualizing the detailed hand motions of musicians playing instruments is challenging. In this work, we present a real-time motion synthesis method that generates the detailed fretting hand motion for playing guitar. Our approach first involves capturing and post-processing hand motion data from guitar performances to create a data set for training. The post-processing ensures that the fingertip positions are placed as accurately as possible regarding distance from the fretboard and location between the frets. We then train a neural network that learns to predict hand and finger poses based on guitar tabs and previous poses. We found that our method produces reasonably stable motion and evaluate our results using accuracy measures and visual evaluation.

CCS Concepts

• **Computing methodologies** → **Animation**.

Keywords

character animation, motion synthesis, virtual humans, hand motions

ACM Reference Format:
Ryan Canales and Sophie Jörg. 2025. Real-time Hand Motion Synthesis for Playing a Virtual Guitar. In *The 18th ACM SIGGRAPH Conference on Motion, Interaction, and Games (MIG '25), December 03–05, 2025, Zurich, Switzerland.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3769047.3769060

MIG '25, Zurich, Switzerland

ACM ISBN 979-8-4007-2236-3/25/12
https://doi.org/10.1145/3769047.3769060

1 Introduction

Accurately tracking and animating hands remains a challenging task, particularly for real-time applications, including virtual reality (VR). Animating hands for playing instruments is particularly challenging, as the generated movement needs to be precise in both movement and timing. While methods have been suggested for capturing music performances [Jin et al. 2024; Perez-Carrillo et al. 2016] and synthesizing motion for playing virtual instruments [Chen et al. 2023; ElKoura and Singh 2003; Hirata et al. 2021; Kyriakou et al. 2024; Luo et al. 2024; Zakka et al. 2023; Zhu et al. 2013], there is a lack of research for synthesizing the motion for real-time, interactive scenarios. As VR has become an established platform for virtual performances [Onderdijk et al. 2023], the demand for high-quality, engaging concert viewing experiences is expected to grow. Hence, methods for generating realistic motion for interactive instrument playing should be investigated.

We therefore present a method for real-time synthesis of fretting hand motions for playing the guitar (see Figure 1). This type of motion requires a very high level of detail, speed, and precision that cannot be captured by consumer VR devices. Even capturing these motions with a high-quality optical motion capture system optimized for hand motion capture is challenging. We therefore first present a method to post-process motion captured guitar motions. Then, we created a data set of guitar playing motions, and we designed and trained a neural network using high-quality motion captured guitar performances to synthesize the full motion of the fretting hand for guitar playing. The model uses annotated guitar music, represented as sequences of note locations on the guitar, and previously generated motion to generate the hand pose at each frame at real-time inference speeds. We evaluated our fretting hand motion synthesis method by comparing the model output to ground truth motion captured data and by measuring the note playing accuracy on a dataset of musical scales. Potential applications range from live virtual music performances, such as VR concerts, to the production of music videos or films featuring virtual characters playing instruments. In addition to enhancing experiences such

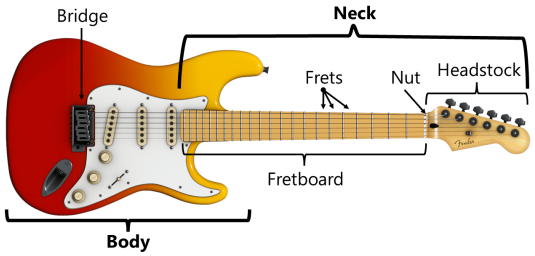

Figure 2: Parts of an electric guitar.

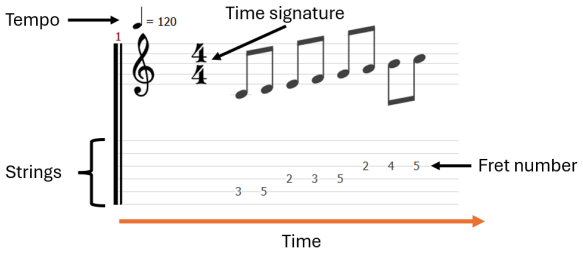

Figure 3: Tab showing one measure from a major scale shape. We refer to string numbers as 1 through 6 (top to bottom).

as live virtual concerts, an interactive motion synthesis method for instrument playing can be used as a tool to generate real-time reference poses for music education [Skreinig et al. 2023].

Our key contributions are as follows:

- We present a high quality hand motion dataset of left and right hand guitar playing motion.
- We detail our motion processing and labeling pipeline for tasks involving fingertip goal positions.
- We describe our algorithm for the real-time synthesis of hand animation with guitar tabs and previous pose outputs and evaluate our approach.

In the next section, we briefly introduce basic terminology related to playing the guitar before presenting research related to our method.

2 Guitar Terminology

The two major parts of a guitar are the body, which serves as a resonating chamber for acoustic guitars, and the neck (see Figure 2). There are typically six strings on a guitar, stretched between the bridge on the guitar body and the nut at the end of the neck such that they each produce a unique pitch. The neck is divided by a series of thin metal strips, called *frets*. When playing the guitar, one or multiple strings are plucked with one hand (the right hand in our examples), while the other presses the string(s) on the guitar neck with one or more fingers, which is called *fretting*. Pressing between frets changes the pitch, or note, of the string by effectively changing its length. Basic fretting hand technique for guitar involves minimizing the amount of movement of the wrist and fingers and generally using the fingertips to fret a note. When guitarists learn a piece of music, they commonly read guitar tablature, or *tabs*, which is written music specifically notated for guitar. Tabs display six lines that are read from left to right, each representing a string (see Figure 3). Numbers indicate the frets to press. The lines the numbers appear on indicate the strings to pluck.

3 Related Work

Hand and finger motions are crucial in our daily lives, for communication, manipulation, and for performing everyday tasks, so it follows that enabling these capabilities for avatars and virtual humans is a significant goal towards making them lifelike and interactive. Finger motions are subtle, yet we are adept at perceiving their motions to such a degree that minor differences can alter the interpretation of a scene [Jörg et al. 2010] or the perceived personality of avatars [Adkins et al. 2023; Wang et al. 2016]. It seems that hand motions can even be used to identify individuals [Normoyle and Jörg 2025]. Although capturing detailed finger motions is possible, there are many restrictions, such as a limited capture area and the need to capture additional motions for new situations [Wheatland et al. 2015]. Therefore, many methods have been developed for capturing hand motions [Han et al. 2018; Jörg et al. 2020; Lee et al. 2019; Wheatland et al. 2013] and synthesizing finger motions for grasping and manipulation [Liu 2008; Rajeswaran et al. 2017] or conversations [Jörg et al. 2012; Mousas and Anagnostopoulos 2017; Mousas et al. 2015; Naert et al. 2020].

However, only a relatively small body of work exists on synthesizing detailed hand motion for virtual performances, specifically instrument playing. Creating animations for playing instruments and guitar in particular comes with many challenges. There are requirements related to the precision and timing of the hand movement. The fingers must contact the fretboard at a precise location within a small margin of error, and should not significantly intersect the fretboard nor each other. The motion of the hands must also have some temporal dependency such that the poses across multiple frames are smooth and natural. This is commonly addressed by other motion synthesis methods with sequence models, such as RNNs and TCNs. Though sequence models can learn temporal dependencies, synthesizing finger motion for guitar playing requires significant movement timing precision. For example, the fretting fingers cannot just briefly touch a goal fret within the timeframe of a note. Instead, they need to press the fret the moment before the note is to be played, and remain pressed until the moment before the note ends. The notes and chords in a melody can have differing lengths. This possible variation of note lengths needs to be supported by a model that has temporal dependencies. As there are a vast number of possible note sequences, and thus, a vast number of fretting hand movements, generalizability is a significant challenge for a supervised model with limited data. Finally, the motion synthesis must be responsive because we are targeting real-time interactive applications.

Recognizing the limitations of previous work, which synthesized finger motion offline, Mousas and Anagnostopoulos [2017] developed a finger motion synthesis method that can be employed in real-time scenarios. This was achieved by training a Hierarchical Hidden Markov Model (HMM) to estimate the current gesture state

and predict the following finger motions. Lee et al. [2019] present a large motion-captured database consisting of conversational gestures and demonstrate the use of recurrent neural networks (RNN's) trained on the data to synthesize finger motion in real-time.

In addition to hand motion synthesis for communication, hand motion synthesis methods for scenarios where the virtual hand interacts with virtual objects is another prominent subject of research, due to potential applications in both computer animation and robotics. Consequently, a number of techniques have been proposed to automate the motion during such interactions. Physics based approaches have been explored for synthesizing grasping animations, since the pose of the simulated hand can be altered based on physical properties of the virtual object [Borst and Indugula 2006; Liu 2008; Pollard and Zordan 2005].

Yet, in VR, we often do not have tactile feedback while interacting with virtual objects, leading to limited interactions methods, such as pinch to grasp, and unrealistic interactions between the virtual hand and object. Hence, methods for animating the hands as they interact with virtual objects in real-time include methods based on physics, heuristics, or deep learning [Borst and Indugula 2006; Jiang et al. 2023; Prachyabrued and Borst 2012; Verschoor et al. 2018; Zhang et al. 2021].

When it comes to synthesizing hand motions specifically for musical instruments, ElKoura and Singh [2003] describe a system for synthesizing the left hand ("fretting" hand) motion for playing guitar with guitar tablature as input. They use an energy optimization approach to generate a set of IK goals for each finger on the fretboard, pose the left hand, then update the pose based on the most similar pose found in a database of captured hand poses (from major scale). Zhu et al. [2013] developed a system for animating both hands for piano playing. Similarly, they use a cost based approach to determining the placement of the fingers for each hand, and a motion database to help pose of fingers not used for playing.

Recently, deep learning methods have been used for synthesizing the motion of instrument playing. Chen et al. [2023] used a generative adversarial network (GAN) trained on a database of reference motions to animate the upper body and hand motion of a character playing Guzheng. Violin hand and upper body animation based on audio input and bowing direction [Hirata et al. 2021] and audio input alone [Hirata et al. 2022] has also been attempted using sequence models and using a diffusion-based framework to animate the whole body [Qiu et al. 2025]. Zakka et al. [2023] introduce piano playing as a benchmark for dexterous hand control, and compare multiple deep reinforcement learning (DRL) control algorithms for training robot hands to play pieces of piano music. Luo et al. [2024] also use reinforcement learning to train a robot to play guitar, with promising results in terms of playing accuracy. Finally, in recent work, Xu and Wang [2024] present a method to generate dexterous hand motions for a guitar player with a focus on synchronizing the left and right hands through cooperative learning. Their work includes a data set which, however, is not post-processed. Another very recent data set was presented by Kyriakou et al. [2025].

The research presented hereafter is towards addressing the spatial and temporal challenges for synthesizing fretting hand animation in real-time settings, focusing on generating motion where playing notes using only the fingertips is desired.

Figure 4: The motion capture setup. Small markers are placed on the musicians hands and around the guitar and are tracked with 15 OptiTrack cameras at 120 frames per second.

4 Motion Data Collection and Processing

Since there was no existing publicly available motion capture data for guitar playing when this work started, we collected a new data set to train our model. We will make the data set public so that it can be used by other researchers.

In early tests of our algorithm, we realized that it was difficult to synthesize compelling results with a limited amount of data as one would expect. We therefore pre-processed our data set to make the fingertip positions hit the frets very precisely when it comes to the height of the fingertip and the location between the frets.

4.1 Data Collection

We used 15 Optitrack Prime optical motion capture cameras, placed around the guitar player at different heights optimized for capturing hand motions, and recorded the motions at 120fps. We used OptiTrack's Motive motion capture software for capturing raw marker data and rigidbody data. Nineteen markers were placed on each hand, three on each digit and four on the palm. The movement of the guitar was tracked as a rigidbody in Motive. There were 9 markers strategically placed on the guitar body to facilitate positioning the virtual guitar model in the virtual scene. The guitar sound was recorded using a Vox Tonelab ST, and videos of each session were also captured. Figure 4 demonstrates the motion capture setup.

4.2 Data Preprocessing

The hand motion data was tracked in real-time using the marker labeling algorithm by Han et al. [2018] and streamed to Unity, where a hand skeleton is constructed and animated based on the labeled markers. The hand motion files and the guitar animation data were then synchronized in Maya.

Because the motion capture sessions took place over several days, replacement of markers onto the hands was required, and therefore hand marker calibration had to be repeated each session. Hence, we retargeted all of the hand data to a common skeleton. The target hand skeleton is also simplified from the source skeleton, with limited degrees of freedom of rotation for the PIP and DIP joints (shown in Figure 5). The retargeting was performed using IK for each finger such that the fingertips of the target hand always reach where the source hand reaches. Further adjustments using pole vector constraints are done to correct any unnatural poses resulting from the IK solver.

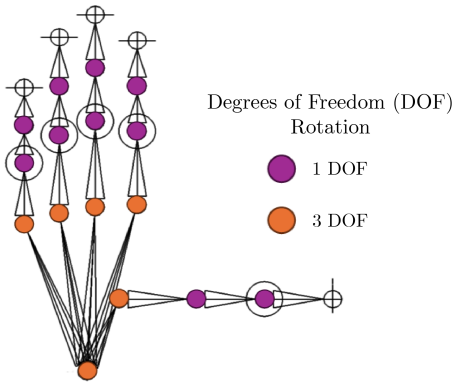

Figure 5: Motion captured data is retargeted to a common hand skeleton. The MCP and wrist joints (orange) have three degrees of freedom (DOF) in rotation, whereas the PIP and DIP joints (purple) have one DOF. Our motion synthesis model predicts the wrist position and rotation as well as the rotation for the remaining color indicated joints.

After synchronizing and retargeting the motion data, there were still inconsistencies in the height at which fingertips would contact the fretboard. These inconsistencies were the result of several factors, including noise in the capture process, slight alignment errors between the guitar markers and guitar model, and alignment errors between the hand and guitar motion. Therefore, we implemented a finger motion modification pipeline that slightly adjusts the motion of the index, middle, ring, and pinky fingertips such that the height at which contact with the guitar occurs is consistent across all motions. Algorithm 1 detects when a fingertip is contacting the fretboard by using a combination of the fingertip velocity (approximated using central finite difference) and the fingertip height above the fretboard, and enforces minimum contacting lengths and minimum times between consecutive contacts. The input to the algorithm is an array of the original 3D fingertip positions relative to the guitar fretboard for one finger (F), and the output is an array of adjusted fingertip positions (F') with contacting fingertip heights set to zero.

For each fingertip (excluding the thumb) in a motion capture take, Algorithm 1 adjusts the height of the detected contacting fingertip positions, which is the 3D point of a fingertip position relative to the guitar fretboard at each frame. The algorithm sets the detected contacting fingertip height (Y) to zero and maintains it at zero until the end of the contact. A fingertip position qualifies as the start of a contact if its height is less than H_{start}, the previous S vertical velocities sum to less than zero, the mean of the previous S velocity magnitudes are greater than the overall mean fingertip velocity, and if it has been more than $Min_{unpressed}$ frames since the end of the previous contact. The conditions for ending a contact are that the fingertip height is higher than H_{end}, the sum of the previous S vertical velocities are greater than to zero, the mean of the previous S velocity magnitudes are greater than the overall mean velocity, and the fingertip has been contacting for at least $Min_{pressed}$ frames. The output of the algorithm can be refined with the following parameters:

- S: a smoothing parameter, number of previous frames of the vertical component of fingertip velocities to average over when considering the start and end of a contact.
- $Min_{pressed}$: the minimum number of frames of a detected contact in order for it to be valid.
- $Min_{unpressed}$: the minimum number of frames between the end of a detected contact and the start of a new contact.
- H_{start}: height threshold to detect the start of a contact.
- H_{end}: height threshold to detect the end of a contact.

In our implementation, $H_{start} = \text{Med}(Y) + \alpha * \text{SD}(Y)$, where α is tuned for each finger in each take and Y is an array of the y-values (fingertip heights) for the fingertip position for a take. We set H_{end} to $\text{Med}(Y_{pressed}) + \text{SD}(Y_{pressed})$, where $Y_{pressed}$ is the array of y-values of the pressing fingertip positions.

Algorithm 1 Fingertip contact detection and height adjustment algorithm. List slicing follows the Python programming language convention: $[start : stop]$ refers to list items from index $start$ through index $stop - 1$ and [:] refers to all list items.

Require: List of fingertip positions F of length n ▷ Positions are 3D vectors ($x = 0, y = 1, z = 2$)
Ensure: List of adjusted fingertip positions F' of length n
  $F[:][1] \leftarrow \text{Smooth}(F[:][1])$ ▷ Smoothing of fingertip height (y)
  $Fvel \leftarrow \text{GetVelocity}(F[:])$ ▷ List of fingertip velocities
  $F[:][1] \mathrel{-}= \text{Min}(F[:][1])$ ▷ Subtract minimum height from all heights
  $\mu_v \leftarrow \text{Mean}(||Fvel[:]||)$ ▷ Mean of the vector norms in $Fvel$
  $F' \leftarrow F$ ▷ Copy F to F'
  $L_p \leftarrow 0$ ▷ Index of last pressed frame
  for $i = 0$ to $\text{Length}(F) - 1$ **do**
    if $F[i][1] < H_{start}$ and $\text{Sum}(Fvel[i - S : i][1]) < 0$ and $\text{Mean}(Fvel[i - S : i]) > \mu_v$ and $i - L_p > Min_{unpressed}$ **then**
      $P \leftarrow [F[i]]$ ▷ Start a list of adjusted positions with $F[i]$
      for $j = i + 1$ to $\text{Length}(F) - 1$ **do**
        if $F[j][1] > H_{end}$ and $\text{Sum}(Fvel[j - S : j][1] > 0$ and $\text{Mean}(Fvel[j - S : j]) > \mu_v$ **then**
          $P.append(F[j])$ ▷ Add $F[j]$ to end of P
          $P[:][1] \leftarrow 0$
          i = j-1
          break
        end if
      end for
      if $\text{CountZeros}(P[:][1]) \geq Min_{pressed}$ **then**
        $F'[i : i + \text{Length}(P)] \leftarrow P$
      end if
    end if
  end for
  Return F'

In a second step, we adjust the contact positions so that they lie centered on the nearest fretboard position, following Algorithm 2. After all adjustments to the fingertip positions are performed, the fingertip positions are smoothed using Savitsky-Golay filtering with a window size of 5 and a 1 degree polynomial and 1cm is added to the fingertip heights to make sure they lie on top of the fretboard and not inside while contacting. The resulting adjusted fingertip

positions are then used as IK goals for each finger, and the resulting finger joint rotations are used for training.

Algorithm 2 Fingertip contacting position adjustment algorithm. List slicing follows the Python programming language convention: $[start : stop]$ refers to list items from index $start$ through index $stop - 1$ and [:] refers to all list items.

Require: List of fingertip contact windows C ▷ Each list element is a list of 3D fingertip positions
Ensure: New fingertip positions F'
  Instantiate a list of new contacting positions C'
  for $i = 0$ to Length$(C) - 1$ **do**
    $S \leftarrow$ Mean$(C[i][:][0])$ ▷ Mean position along string axis
    $C[i][:][0] \leftarrow S$ ▷ Force same string for contact length
    FretPositions $\leftarrow$ Quantized fret positions for all points in $C[i]$
    $P0 \leftarrow$ Mode$(FretPositions)$ ▷ Most frequent fret position
    $P1 \leftarrow$ NearestFretboardPos$(C[i][c-1])$ ▷ c is length of $C[i]$
    InterpFrames $\leftarrow 1$
    if $P0 \neq P1$ **then**
      InterpFrames $\leftarrow$ # of frames from last instance of $P0$ to first $P1$
    end if
    if $P0$ or $P1 \in$ PreviousContacts **then**
      Increment fret of $P0$ and $P1$ by 1 ▷ Avoid overlapping contacts with other fingers
    end if
    for $j = 0$ to Length$(C[i]) - 1$ **do**
      $I \leftarrow P0$
      if InterpFrames > 0 **then**
        $t \leftarrow j/$InterpFrames$(C[i])$
        $I \leftarrow P0 + t * (P1 - P0)$
      end if
      $C'[i][j] \leftarrow (I_x, 0, I_z)$
    end for
  end for
  $j \leftarrow 0$
  for $i = 0$ to Length$(F') - 1$ **do**
    if $F[i][1] \leq 0$ and $j <$ Length(C') **then**
      $F[i] \leftarrow C'[j]$ ▷ Assign new contacting position $C'[j]$ to fingertip $F[i]$
      j += 1
    end if
  end for
  $F'[:] \leftarrow$ Smooth$(F'[:])$ ▷ Smooth the resulting new fingertip positions
  Return F'

Finally, the data is labeled using the fingertip heights above the fretboard. If the fingertip height is less than a threshold T (set to 0.05cm), a label is created using the nearest fret and string position on the guitar as well as which fingertip is contacting. The labels are created per fingertip. If multiple fingertips are contacting the fretboard on the same string, only the one pressing the highest fret is used for labeling. This rule is added because while playing guitar, a player can press multiple frets on the same string, however, only the highest fret on that string will sound.

5 Model For Fretting Hand Motion Synthesis

Table 1: The motion capture takes consisting of six keys of the major scale. The note length statistics (seconds) are based on the length of the detected contacts from the motion data.

Take Name	# Frames	# Notes	Mean/SD Note Length
Ab Maj.	3656	368	0.166 / 0.12
A Maj.	3941	336	0.195 / 0.115
Bb Maj.	3954	341	0.193 / 0.124
C Maj.	4001	345	0.193 / 0.117
Db Maj.	3407	309	0.184 / 0.124
D Maj.	3421	293	0.195 / 0.12
F Maj.	3981	347	0.191 / 0.117
Exercise 1a	5053	535	0.157 / 0.09
Exercise 1b	5155	568	0.151 / 0.08
Random 1	6962	703	0.165 / 0.139
Random 2	1883	353	0.089 / 0.064
Random 3	5227	683	0.128 / 0.092
Exercise IR-MP	3615	328	0.184 / 0.12
2 Chords	1195	160	0.124 / 0.167

5.1 Dataset for Training

A total of fourteen motion capture takes, shown in Table 1, are used: ten were used as the training set, two as the validation set, and two as the test set. The training set consists of performances of five keys of the major scale (Ab maj., Bb maj., C maj., D maj., and F maj.), two structured motions (Exercise 1a and 1b) that use all four fretting fingers on every string (similar to the "spider exercise") up and down the fretboard, a randomly improvised performance of individual notes (Random 2), a take with common two-note chords shapes (2 Chords), and a take with combinations of index-ring, middle-pinky movements (Exercise IR-MP). The validation set includes a performance of the A major scale (A maj.) and an improvised performance of random notes (Random 1), and the test set includes a performance of the Db major scale (Db Maj.) and another improvised performance of random notes (Random 3). The motion data was downsampled from 120fps to 60fps before being used for training and evaluation.

The hand motion data was transformed before being used for training such that the movement of the guitar is negated. To do this, the wrist position is computed relative to the guitar for each frame: $R_{guitar}^{-1} * (P_{wrist} - P_{guitar})$, where P_{wrist} and P_{guitar} are the 3D world locations of the wrist and guitar, and R_{guitar} is the world rotation of the guitar, represented as a quaternion. The MCP joint and fingertip joint locations are also converted to be relative to the guitar. The rotations (as quaternions) for the PIP and DIP joints are computed relative to their parent joint rotations. Before the motion is used as input to the model, the rotations are converted from quaternions to Euler angles in radians, which preserves network capacity by reducing the size of the inputs and outputs.

To increase the number of motion samples used for training, dataset augmentation was performed. Two techniques were used for augmenting each take. One was random speed scaling (*speed*),

which altered the lengths of notes (and their corresponding motion) by a random factor. The same scaling factor was applied to groups of consecutive notes. For scaling factors less than 1, notes were grouped to have 3 to 8 notes (randomly determined), and for scaling factors greater than 1, they were grouped to have 8 to 30 notes. The scaling factor for each group of notes was randomly sampled from an evenly spaced set of 20 values between 0.1 (1/10th speed) and 2 (2x speed). If the scaling factor was less than 1 (slower), the pose of the hand was linearly interpolated between frames to maintain the same framerate (60fps). If the scaling factor was more than 1 (faster), the frames of the motion were subsampled. The second augmentation technique was clip reversal (*reverse*), which reversed the order of the notes and their corresponding motion for an entire take. Each augmentation technique was applied to each take in our dataset, resulting in 4 variations of each take (*original*, *speed*, *reverse*, *speed+reverse*). We included the augmented versions of the clips in the validation and test sets because the augmentations reflect realistic variations in tempo and motion speed that are expected in real-world scenarios.

The entire data set including augmentations totals 248,096 frames (about 68.9 minutes). The training set contained 160,006 frames (about 44.4 minutes), the validation set contained 47,815 frames (about 13.3 minutes), and the test set contained 40,275 frames (about 11.2 minutes). Additional details about the dataset (without augmentations) are shown in Table 1.

5.2 Network Input and Output

Our model uses the input sequence of future notes and previous wrist and finger motion to predict the hand pose at each frame. There are two sets of features for training and prediction. The first set of features are the "goal features", which consist of the fretboard positions of the notes from the following N_g frames, specifically from frame t through frame $t + N_g$. The fretboard positions for a note or chord consists of the 2D positions of the fret (or frets for chords) on the guitar fretboard. There are six strings and up to six frets being pressed at once, so the fretboard position vector for one frame consists of 12 elements (2D position x 6 strings). If there are no frets being pressed on a string, the fretboard position for that string is a 12 element zero vector. With the currently used dataset, the goal feature vectors are often sparse, meaning most of the time the goal is to play a single note.

The second set of features are based on the pose of the fretting hand ("pose features"). The wrist motion feature is the sequence of the preceding N_m frames of 3D wrist positions and 3D orientations, relative to the guitar. The finger motion feature is the sequence of the preceding N_m frames of the local finger joint orientations. The finger joint orientation feature vector for one frame is 25 elements, corresponding to the sum of the degrees of freedom for each joint (see Figure 5). Since the fingertip positions are a function of both the wrist and finger poses, we also include the fingertip positions from the preceding frame $(t-1)$, relative to the guitar and excluding the thumb. All position features are in meters and all orientation features are in radians. Based on these features, the model predicts the wrist pose (position and orientation) and the finger pose (orientations) at frame t for the fretting hand. The resulting input sizes for each feature are:

- $x1 = N_g * 12$: the N_g length sequence of goal features, flattened.
- $x2 = 12$: the vector of the 3D fingertip positions for the index, middle, ring, and pinky.
- $x3 = (N_m \times 6)$: the N_m length sequence of previous wrist positions and rotations.
- $x4 = (N_m \times 25)$: the N_m length sequence of previous finger joint rotations.

5.3 Network Architecture

Each set of features is first transformed with fully-connected (FC) layers, with the goal and fingertip features further encoded with one more FC layer each. The wrist and finger motion features are each independently encoded with a 3-layer RNN, then the flattened encoded sequences are transformed again with FC layers. The resulting feature encodings are then concatenated and used as input to two residual dense blocks, which allow information to be carried from the input to the output [Zhang et al. 2021, 2018], and a final decoding FC layer, which outputs the predicted hand pose for the current frame. A diagram of the architecture is shown in Figure 6.

The loss function for the base model is composed of four terms: the finger rotation loss ($L1$), the wrist rotation loss ($L2$), the wrist position loss ($L3$), the inverse kinematics (IK) loss ($L4$), and the goal loss ($L5$). The finger rotation loss and the wrist rotation loss are the Mean Squared Errors (MSE) between the predicted and ground truth finger joint and wrist rotations (in radians). The wrist position loss is the MSE between the predicted wrist position and ground truth wrist position, in meters. Since playing the guitar requires precise fingertip placement on the guitar fretboard, using just these pose-based losses is likely to be insufficient. To help ensure that the resulting fretting fingertip positions (index, middle, ring, and pinky) factor into our model, we include the IK loss, similar to what was done by Lee et al. [2019]. To compute the IK loss, the first step is solving for the predicted fretting fingertip locations relative to the guitar fretboard using the predicted wrist and finger joint rotations. Likewise, the ground truth fretting fingertip locations are computed relative to the guitar fretboard. The IK loss is the MSE between the predicted fretting fingertip positions and ground truth fretting fingertip positions. The IK computation is only performed during training. Finally, a goal loss is included to help the model learn the relationship between the fretting fingertips and the goal positions. The goal loss is calculated by computing the absolute differences between each fretting fingertip position and each fretboard goal position, then averaging the minimum fingertip distance to each valid (nonzero) goal position. Both the IK loss and goal loss are used to encourage the model to learn to press at least one fingertip on the goal fret (goal loss) while simultaneously encouraging it to learn which fingertip to use (IK loss). The full loss term is: $\mathcal{L} = L1 + L2 + L3 + L4 + L5$.

5.4 Training Details

Previous work in synthesizing real-time hand motion in VR focused on predicting only the finger poses during grasps since they use tracked wrist movement as input at inference ([Jiang et al. 2023; Zhang et al. 2021]). Our model, however, predicts the entire hand pose, including the wrist, meaning that all pose-based inputs are

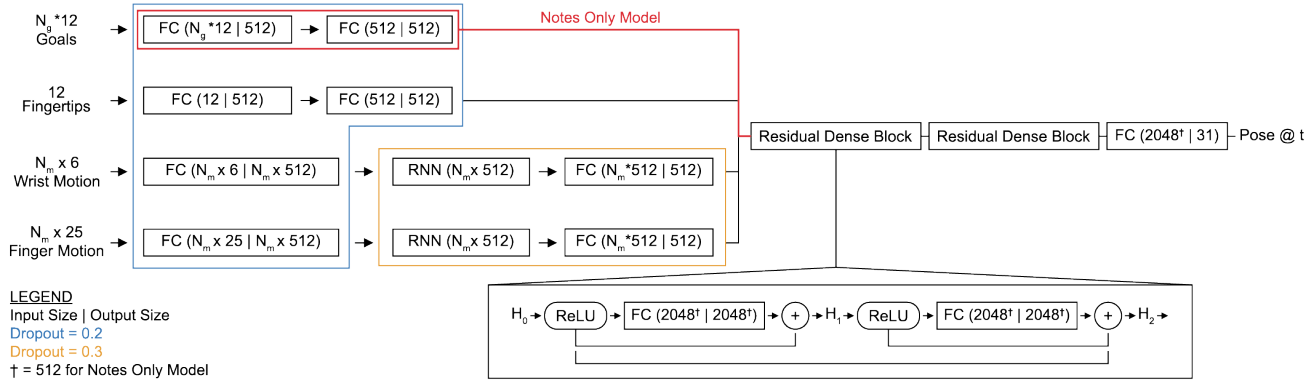

Figure 6: The neural network architecture for the fretting hand. Each feature is encoded independently, then concatenated before being used as inputs to two residual dense blocks. The values in the parentheses indicate the input size (left side) and output size (right side) for each node. The input and output sizes for the RNNs are the same. Dropout is applied during training to the layers within the blue and orange outlines, with probabilities indicated in the legend. Dropout is applied within the RNN block to each layer except the output layer. The architecture for the Notes Only model is outlined in red.

Table 2: The limits for the wrist position and rotation and the finger (Thumb, I, M, R, P) joint rotations. The wrist position units are in meters and the joint rotations are in degrees.

Value	Lower / Upper Limit
Wrist Position	[-0.15, -0.15, -0.2] / [0, 0, 0.2]
Wrist Rotation	[-360, -360, -360] / [360, 360, 360]
Thumb: MCP	[-360, -360, -360] / [360, 360, 360]
Thumb: PIP	-360 / 360 (Y)
Thumb: DIP	-360 / 360 (Y)
I, M, R, P: MCP	[-30, -180, -44] / [19, 180, 89]
I, M, R, P: PIP	0 / 98 (Z)
I, M, R, P: DIP	0 / 79 (Z)

from previous model predictions at inference. This could result in the model outputs drifting over time or regressing to the mean. Therefore, we use a variant of scheduled sampling [Bengio et al. 2015] for training the model. Scheduled sampling allows the model to learn from previous motion features taken from both the ground truth and the previous model outputs. This is done by randomly sampling batches from either the ground truth or previous model outputs. The probability of sampling the a batch of ground truth previous poses is ϵ and the probability of sampling a batch of previous model outputs $1 - \epsilon$. The parameter ϵ decays linearly after each epoch, with a minimum value of 0.05 enforced so the model can still sample ground truth batches. When evaluating the model on the validation set, the previous motion inputs are always from the previous model outputs, giving a more accurate evaluation of model performance on unseen data after each training iteration. The batches are not randomized since the order of the batches matters when training using the previous model outputs. The batches are created based on each motion capture take to avoid discontinuities between model outputs between takes. While training, the gradients are clipped to a maximum value of 10 to stabilize training.

Dropout regularization is used on each linear layer before concatenation and within the RNNs to avoid overfitting the training set. See Figure 6 for details on where dropout is applied. All models were trained with the AdamW optimizer with a learning rate of 0.0001, weight decay of 0.001, and a batch size of 512 samples. The parameters N_g and N_m are both set to 30.

The predicted pose values are clamped to be within a valid range (see Table 2) enabling the model to focus on inputs that are within a valid range during training and prevent the output values from diverging excessively during inference. Importantly, while training, these values are only clamped *after* the loss is computed, meaning the loss value reflects the true prediction error.

The model trained for 500 epochs. Inference takes less than 0.01 seconds per frame on a PC with an AMD Ryzen 7 5700X CPU and Nvidia RTX 3060 GPU, fast enough for the target 60 fps application. For inference, the initial hand pose is chosen by averaging the fretboard positions of the first note or chord in the input tablature and retrieving the database pose with the nearest mean fretboard position. The initial pose is repeated for N_m frames.

5.5 Prediction Postprocessing

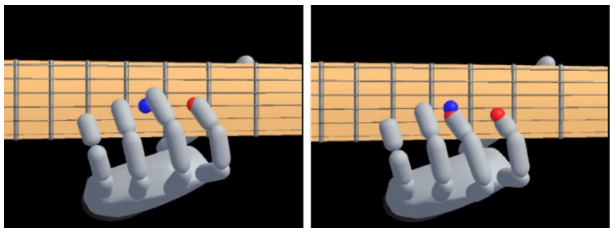

Figure 7: A frame from the test set before IK adjustment (left) and after (right). The blue sphere indicates where a fingertip should press, and the red sphere shows where a fingertip is pressing. In this example, the middle fingertip is adjusted.

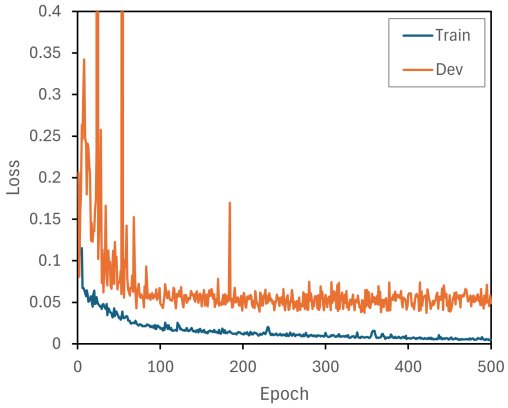

Figure 8: The loss curves for the training (train) and validation (dev) set for the model trained with scheduled sampling.

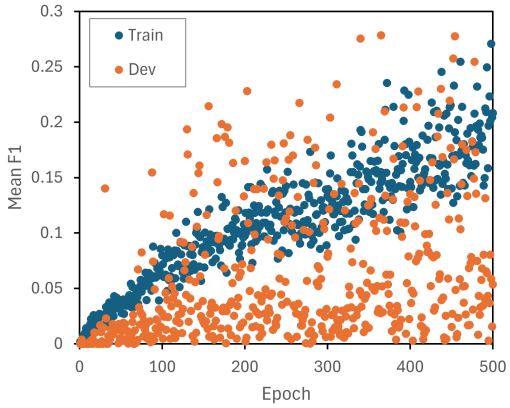

Figure 9: The mean F_1 score per epoch for the training (train) and validation (dev) set for the model trained with scheduled sampling. The F_1 score on the training set trends up over time. This trend is less apparent for the validation set.

To improve the frequency that the generated pose's fingertips contact the fretboard goals, a rudimentary IK-based augmentation is performed as a real-time postprocessing step in Unity. In this phase, we update the predicted finger poses by finding the nearest fingertip to each goal at the current frame, then linearly interpolating the fingertip towards the goal over 5 frames. A fingertip is marked as assigned until either the predicted fingertip has moved further than a given threshold from the goal or once the goal has cleared. If a fingertip is assigned to a goal, the next nearest fingertip to that goal is assigned to it. When a fingertip is not assigned to a goal, it is linearly interpolated back to the predicted fingertip position over 5 frames. Unity's two-bone IK constraint, set up with the MCP, PIP, and fingertip joints, is used to update the finger joint rotations in order to reach the newly adjusted fingertip goal positions. An example of the prediction results before and after postprocessing is shown in Figure 7. While this simple adjustment method can make a fingertip reach a target more frequently, it might result in artifacts, such as part of the finger clipping through the fretboard.

6 Evaluation and Results

We evaluate our model and our results by assessing if the notes are played accurately and by examining rendered examples. We also perform an ablation study on the model input features and compare the fretting accuracy after training with and without scheduled sampling. We consider our model with all inputs ("goal features" + "pose features") and trained with scheduled sampling to be the baseline model (referred to as Baseline).

6.1 Training Results

Inspecting the loss curves in Figure 8, we observe that the model quickly adapts to the data, as indicated by the rapid decrease in the training loss within the first 100 epochs. However, while the training loss continues to decrease steadily, the validation loss plateaus sooner and varies much more at each iteration than the training loss. The plateauing validation loss curve indicates that the training set may not be representative enough to generalize to the validation set in terms of pose-target position pairs.

As expected, the average F_1 score improves (described in Section 6.2) as the model continues training. Although it increases slowly, the trend is clearly noticeable for the training set F_1 score, as shown in Figure 9. However, while the validation set F_1 score trends up, it is much more variable per epoch, and sometimes surpasses the training $F1$. Inspecting the pose related loss terms ($L1$, $L2$, $L3$, and $L4$) during training on the validation set, it appears that the wrist rotation loss and finger rotation loss are the most difficult to generalize to unseen data. However, the goal loss ($L5$) on the validation set is very close to that of the training set.

6.2 Note Playing Accuracy

To quantify how precisely the synthesized motion plays the notes in a sequence, we calculate the fingertip contact precision, recall, and F_1 score at every output frame in the output sequence and average them. The goal at each frame is the set of fretboard positions that should be pressed by a fingertip. A set of contacted fretboard positions is created based on the predicted pose at each frame. Similar to how notes were labeled, if there are multiple fingertip contacts on the same string, only the one associated with the highest fret number is used. If a fingertip is contacting the correct fret and string for a note, this is considered a true positive (TP). On the other hand, when a finger is touching a fret and string that do not correspond to the notes that are supposed to be played, it is considered a false positive (FP). Likewise, if a finger does not contact the corresponding fret and string for a note, it will be considered a false negative (FN). If a finger is not contacting a fret and string and there is no note to be played, it is a true negative (TN). The precision is $P = TP/(TP + FP)$, the recall is $R = TP/(TP + FN)$. The F_1 score is the harmonic mean of the precision and recall $2 * (R * P)/(R + P)$.

The models we selected for evaluation were the Baseline model and the Notes Only model (described later in Section 6.3). For both models, the checkpoint weights that resulted in the highest $F1$ score were used for inference. Table 3 shows the mean F_1, precision, and recall values for the generated motion resulting from the input set

Table 3: The mean F_1, precision (P), and recall (R) scores for all takes in the test set for the Baseline and Notes Only models. The values in the columns labeled "Baseline" and "Notes Only" are computed directly from the model output (before IK adjustments). The columns labeled "Baseline IK" and "Notes Only IK" show the scores after IK adjustments are applied (described in Section 5.5).

Baseline	Baseline + IK	Notes Only	Notes Only + IK
F_1: 0.273	F_1: 0.523	F_1: 0.377	F_1: 0.554
P: 0.281	P: 0.524	P: 0.384	P: 0.562
R: 0.277	R: 0.543	R: 0.381	R: 0.561

Table 4: The F_1, precision (P), and recall (R) values of the synthesized motion from tabs containing additional major scales at three tempos. These values are with IK adjustments.

Scale Tempo	Baseline F_1, P, R	Notes Only F_1, P, R
80BPM	0.532, 0.512, 0.572	0.579, 0.567, 0.603
100BPM	0.525, 0.507, 0.563	0.569, 0.558, 0.59
120BPM	0.501, 0.457, 0.534	0.570, 0.560, 0.591

of notes from the test set both before and after IK adjustments. Our IK postprocessing method improves fretting performance, however, the values are still low compared to the ground truth. This is due to the IK solver being unable to reach the goals because the model positioned the hand too far away from them and also in part due to the limitations of our postprocessing step outlined in Section 5.5.

Since measuring note playing accuracy does not require a ground truth comparison, we also tested the models on major scales not included in the training or validation set as inputs (E, G, B, Eb, Gb, Bb) with 648 eighth notes. We measured the performance on this dataset at 80, 100, and 120 beats per minute (BPM). The accuracy results are similar to the test set, indicating the models' ability to handle sequences of notes at different tempos. Table 4 shows the note playing accuracy on this dataset.

6.3 Ablation Study

We performed an ablation study in order to understand the effect that the input features and scheduled sampling have on the fretting accuracy on the validation set. We trained a version of our model, referred to as Notes Only, that includes only the future notes (N_g = 30) as input (the "goal features") with no pose related features. This required removing the RNN nodes from the neural network architecture, since they only processed previous pose inputs (Figure 6 outlines the Notes Only architecture variant) . The Notes Only model was trained with scheduled sampling. Interestingly, having only the goal features present resulted in higher F1, precision, and recall scores on the validation set than the Baseline model with all input features (see Table 5). However, our observation is that the resulting motion exhibits less pose diversity compared to our baseline model with all input features.

We also trained our model with all features described in Section 5.2 with and without scheduled sampling. We found that training with scheduled sampling (described in Section 5.4) resulted in a higher F1 score on the validation set than when trained without scheduled sampling. The resulting F1, precision, and recall values are shown in Table 5.

Table 5: The mean F_1, precision (P), and recall (R) scores for all takes in the validation set for the Notes Only model (trained with scheduled sampling (SS)), a model trained with all features but without SS, and our baseline model. These results are without IK adjustments.

Notes Only + SS	All Features	Baseline (All Features + SS)
F_1: 0.371	F_1: 0.227	F_1: 0.280
P: 0.381	P: 0.235	P: 0.291
R: 0.374	R: 0.227	R: 0.280

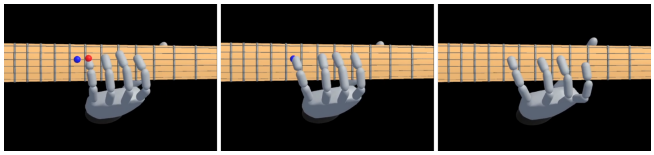

Figure 10: An example of the generated pose from the baseline model (left), the pose after IK adjustment (center), and the ground truth pose (right). The blue sphere indicates where a fingertip should press, and the red sphere indicates where the fingertip is pressing.

6.4 Motion Quality

Example poses generated with our method are shown in Figure 10. Our supplementary video shows side-by-side examples of generated motion from the test set, the generated motion with IK adjustments, and the corresponding ground truth motion. Looking at example sequences, we can see that our method generates reasonable wrist motions with intricate finger animations, reaching many of the required goals. Compared to the ground truth, however, one notices that the wrist sometimes moves too much and with slight jitter, which might be improved by adding an energy loss term.

7 Discussion and Limitations

Our results show a promising direction for synthesizing detailed, dexterous hand motion for guitar playing using only guitar tablature as input. The model learns to synthesize continuous wrist and finger movement for an indefinite amount of time. Our testing shows that the model can handle synthesizing motion for long sequences with notes that vary in length. Despite not explicitly modeling the dynamics of the motion, such as acceleration and velocity, the model learned to moderate the magnitude of the speed and movement of the hand from the motion data alone.

While our method can reasonably synthesize the motion of the fretting hand based on guitar tablature, the pipeline still relies on IK adjustment as a postprocessing step, and the output is limited to scale-like motions without complex chords or expressive fretting hand motions, such as bends and slides. Furthermore, our method was developed with an emphasis on fingertip placement, whereas

in real-life, other parts of the fingers can press against the fretboard. Moreover, as a result of emphasizing fingertip placement, our method does not guarantee that other parts of the finger will not press against the fretboard, which would affect the sound in real-life guitar performances. The fretting hand plays a significant role while playing guitar, and therefore we focused on synthesizing the movement of the fretting hand. However, a full guitar playing motion synthesis algorithm would include animating both hands.

We evaluated our method via visual comparisons to ground truth motion and by measuring fingertip placement accuracy both before and after postprocessing, but further qualitative evaluation of the synthesized motion through a user study would be valuable. Likewise, further evaluation that assesses motion realism in terms of the believability of the fretting hand movement by guitarists would help determine the potential utility as an instructional tool. Finally, a larger problem that this research has not fully addressed is the task of learning from unstructured data. In order to achieve the results presented, structure was added to the data through preprocessing and labeling.

8 Conclusion and Future Work

Synthesizing detailed hand and finger motion remains a significant challenge, especially for real-time applications, such as those designed for virtual reality. With this research, we address some of the many challenges of synthesizing detailed and accurate hand and finger motion by focusing on the challenging, precision hand motion task of guitar playing. Animating hands for playing instruments is particularly challenging, as the generated movement needs to be precise in both movement and timing. Our method can reasonably generate fretting hand motion for individual notes and simple chords using guitar tablature as input. Potential applications of this genre of work range from live virtual music performances, such as VR concerts, to the production of music videos or films featuring virtual characters playing instruments. In addition to enhancing experiences like live virtual concerts, a real-time motion synthesis method for instrument playing could be used as a tool to generate real-time reference poses for music education. Moreover, the scope of applications could extend beyond music to other applications requiring detailed hand animation for performing tasks with high levels of dexterity, speed, and precision.

In future work, we wish to further improve our motion synthesis results. Near term improvements might include enhancing the prediction post-processing pipeline and using the post-processed output as input for prediction while training and during inference. Our method might also benefit from training with more data, with a particular focus on motion diversity. Investigation into techniques that utilize motion capture but rely less on accurately labeled data could also be a promising avenue for future research.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Award No. IIS-1652210.

References

Alex Adkins, Aline Normoyle, Lorraine Lin, Yu Sun, Yuting Ye, Massimiliano Di Luca, and Sophie Jörg. 2023. How Important are Detailed Hand Motions for Communication for a Virtual Character Through the Lens of Charades? *ACM Trans. Graph.* 42, 3, Article 27 (may 2023), 16 pages. doi:10.1145/3578575

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. *Advances in neural information processing systems* 28 (2015).

Christoph W Borst and Arun P Indugula. 2006. A spring model for whole-hand virtual grasping. *Presence* 15, 1 (2006), 47–61.

Jiali Chen, Changjie Fan, Zhimeng Zhang, Gongzheng Li, Zeng Zhao, Zhigang Deng, and Yu Ding. 2023. A Music-Driven Deep Generative Adversarial Model for Guzheng Playing Animation. *IEEE Transactions on Visualization and Computer Graphics* 29, 2 (2023), 1400–1414. doi:10.1109/TVCG.2021.3115902

George ElKoura and Karan Singh. 2003. Handrix: Animating the Human Hand. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*. Eurographics Association, 110–119.

Shangchen Han, Beibei Liu, Robert Wang, Yuting Ye, Christopher D. Twigg, and Kenrick Kin. 2018. Online Optical Marker-based Hand Tracking with Deep Labels. *ACM Trans. Graph.* 37, 4, Article 166 (July 2018), 10 pages. doi:10.1145/3197517.3201399

Asuka Hirata, Keitaro Tanaka, Masatoshi Hamanaka, and Shigeo Morishima. 2022. Audio-Driven Violin Performance Animation with Clear Fingering and Bowing. In *ACM SIGGRAPH 2022 Posters*. 1–2.

Asuka Hirata, Keitaro Tanaka, Ryo Shimamura, and Shigeo Morishima. 2021. BowingNet: Motion Generation for String Instruments Based on Bowing Information. In *ACM SIGGRAPH 2021 Posters*. 1–2.

Haiyan Jiang, Dongdong Weng, Zhen Song, Xiaonuo Dongye, and Zhenliang Zhang. 2023. DexHand: Dexterous hand manipulation motion synthesis for virtual reality. *Virtual Reality* (2023), 1–16.

Yitong Jin, Zhiping Qiu, Yi Shi, Shuangpeng Sun, Chongwu Wang, Donghao Pan, Jiachen Zhao, Zhenghao Liang, Yuan Wang, Xiaobing Li, Feng Yu, Tao Yu, and Qionghai Dai. 2024. Audio Matters Too! Enhancing Markerless Motion Capture with Audio Signals for String Performance Capture. *ACM Trans. Graph.* 43, 4, Article 90 (July 2024), 10 pages. doi:10.1145/3658235

Sophie Jörg, Jessica Hodgins, and Carol O'Sullivan. 2010. The Perception of Finger Motions. In *Proceedings of the 7th Symposium on Applied Perception in Graphics and Visualization (APGV 2010)* (Los Angeles, California). 129–133. doi:10.1145/1836248.1836273

Sophie Jörg, Jessica K. Hodgins, and Alla Safonova. 2012. Data-driven Finger Motion Synthesis for Gesturing Characters. *ACM Transactions on Graphics* 31, 6 (November 2012), 189:1–189:7. doi:10.1145/2366145.2366208

Sophie Jörg, Yuting Ye, Franziska Mueller, Michael Neff, and Victor Zordan. 2020. Virtual hands in VR: Motion capture, synthesis, and perception. In *SIGGRAPH Asia 2020 Courses*. 1–32.

T. Kyriakou, A. Aristidou, and P. Charalambous. 2025. Multi-Modal Instrument Performances (MMIP): A Musical Database. *Computer Graphics Forum* 44, 2 (2025), e70025. doi:10.1111/cgf.70025

Theodoros Kyriakou, M. Á de la Campa Crespo, Andreas Panayiotou, Yiorgos Chrysanthou, Panayiotis Charalambous, and Andreas Aristidou. 2024. Virtual Instrument Performances (VIP): A Comprehensive Review. In *Computer Graphics Forum*. Wiley Online Library, e15065.

Gilwoo Lee, Zhiwei Deng, Shugao Ma, Takaaki Shiratori, Siddhartha Srinivasa, and Yaser Sheikh. 2019. Talking With Hands 16.2M: A Large-Scale Dataset of Synchronized Body-Finger Motion and Audio for Conversational Motion Analysis and Synthesis. 763–772. doi:10.1109/ICCV.2019.00085

C. Karen Liu. 2008. Synthesis of interactive hand manipulation. In *Proceedings of the 2008 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Dublin, Ireland) *(SCA '08)*. Eurographics Association, Goslar, DEU, 163–171.

Chaoyi Luo, Pengbin Tang, Yuqi Ma, and Dongjin Huang. 2024. Learning to Play Guitar with Robotic Hands. In *Computer Graphics Forum*. Wiley Online Library, e15166.

Christos Mousas and C. Anagnostopoulos. 2017. Real-time performance-driven finger motion synthesis. *Computers and Graphics* 65 (2017), 1–11.

Christos Mousas, Christos-Nikolaos Anagnostopoulos, and Paul Newbury. 2015. Finger Motion Estimation and Synthesis for Gesturing Characters. In *Proceedings of the 31st Spring Conference on Computer Graphics* (Smolenice, Slovakia) *(SCCG '15)*. 97–104. doi:10.1145/2788539.2788552

Lucie Naert, Caroline Larboulette, and Sylvie Gibet. 2020. A survey on the animation of signing avatars: From sign representation to utterance synthesis. *Computers & Graphics* 92 (2020), 76–98.

Aline Normoyle and Sophie Jörg. 2025. User identification based on conversational gestures. In *2025 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*. IEEE Computer Society, Los Alamitos, CA, USA, 291–294. doi:10.1109/VRW66409.2025.00068

Kelsey E Onderdijk, Lies Bouckaert, Edith Van Dyck, and Pieter-Jan Maes. 2023. Concert experiences in virtual reality environments. *Virtual Reality* 27, 3 (2023), 2383–2396.

Alfonso Perez-Carrillo, Josep-Lluis Arcos, and Marcelo Wanderley. 2016. Estimation of guitar fingering and plucking controls based on multimodal analysis of motion, audio and musical score. In *Music, Mind, and Embodiment: 11th International Symposium, CMMR 2015, Plymouth, UK, June 16-19, 2015, Revised Selected Papers 11*. Springer, 71–87.

Nancy S. Pollard and Victor B. Zordan. 2005. Physically Based Grasping Control from Example. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation* (Los Angeles, California) *(SCA '05)*. ACM, 311–318. doi:10.1145/1073368.1073413

Mores Prachyabrued and Christoph W. Borst. 2012. Visual interpenetration tradeoffs in whole-hand virtual grasping. In *2012 IEEE Symposium on 3D User Interfaces (3DUI)*. 39–42. doi:10.1109/3DUI.2012.6184182

Zhiping Qiu, Yitong Jin, Yuan Wang, Yi Shi, Chao Tan, Chongwu Wang, Xiaobing Li, Feng Yu, Tao Yu, and Qionghai Dai. 2025. ELGAR: Expressive Cello Performance Motion Generation for Audio Rendition *(SIGGRAPH Conference Papers '25)*. Article 54, 9 pages. doi:10.1145/3721238.3730756

Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. 2017. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087* (2017).

Lucchas Ribeiro Skreinig, Denis Kalkofen, Ana Stanescu, Peter Mohr, Frank Heyen, Shohei Mori, Michael Sedlmair, Dieter Schmalstieg, and Alexander Plopski. 2023. GuitARhero: Interactive Augmented Reality Guitar Tutorials. *IEEE Transactions on Visualization and Computer Graphics* 29, 11 (2023), 4676–4685. doi:10.1109/TVCG.2023.3320266

Mickeal Verschoor, Daniel Lobo, and Miguel A. Otaduy. 2018. Soft Hand Simulation for Smooth and Robust Natural Interaction. In *2018 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. 183–190. doi:10.1109/VR.2018.8447555

Yingying Wang, Jean E. Fox Tree, Marilyn Walker, and Michael Neff. 2016. Assessing the Impact of Hand Motion on Virtual Character Personality. *ACM Trans. Appl. Percept.* 13, 2, Article 9 (mar 2016), 23 pages. doi:10.1145/2874357

Nkenge Wheatland, Sophie Jörg, and Victor Zordan. 2013. Automatic Hand-Over Animation using Principle Component Analysis. In *Proceedings of Motion on Games* (Dublin 2, Ireland) *(MIG '13)*. 197–202. doi:10.1145/2522628.2522656

Nkenge Wheatland, Yingying Wang, Huaguang Song, Michael Neff, Victor Zordan, and Sophie Jörg. 2015. State of the Art in Hand and Finger Modeling and Animation. *Computer Graphics Forum* (2015).

Pei Xu and Ruocheng Wang. 2024. Synchronize Dual Hands for Physics-Based Dexterous Guitar Playing. In *SIGGRAPH Asia 2024 Conference Papers (SA '24)*. Association for Computing Machinery, New York, NY, USA, Article 143, 11 pages. doi:10.1145/3680528.3687692

Kevin Zakka, Philipp Wu, Laura Smith, Nimrod Gileadi, Taylor Howell, Xue Bin Peng, Sumeet Singh, Yuval Tassa, Pete Florence, Andy Zeng, and Pieter Abbeel. 2023. RoboPianist: Dexterous Piano Playing with Deep Reinforcement Learning. In *Proceedings of The 7th Conference on Robot Learning (Proceedings of Machine Learning Research, Vol. 229)*, Jie Tan, Marc Toussaint, and Kourosh Darvish (Eds.). PMLR, 2975–2994. https://proceedings.mlr.press/v229/zakka23a.html

He Zhang, Yuting Ye, Takaaki Shiratori, and Taku Komura. 2021. ManipNet: Neural manipulation synthesis with a hand-object spatial representation. *ACM Transactions on Graphics (ToG)* 40, 4 (2021), 1–14.

Yulun Zhang, Yapeng Tian, Yu Kong, Bineng Zhong, and Yun Fu. 2018. Residual dense network for image super-resolution. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2472–2481.

Yuanfeng Zhu, Ajay Sundar Ramakrishnan, Bernd Hamann, and Michael Neff. 2013. A system for automatic animation of piano performances. *Computer Animation and Virtual Worlds* 24, 5 (2013), 445–457.