DOI: 10.1111/cgf.15166

ACM SIGGRAPH / Eurographics Symposium on Computer Animation 2024
M. Skouras and H. Wang
(Guest Editors)

COMPUTER GRAPHICS *forum*
Volume 43 (2024), Number 8

Learning to Play Guitar with Robotic Hands

Chaoyi Luo[1], Pengbin Tang[2], Yuqi Ma[1] and Dongjin Huang[†][1]

[1]Shanghai Film Academy, Shanghai University, Shanghai, China
[2]ETH Zürich, Switzerland

Abstract

Playing the guitar is a dexterous human skill that poses significant challenges in computer graphics and robotics due to the precision required in finger positioning and coordination between hands. Current methods often rely on motion capture data to replicate specific guitar playing segments, which restricts the range of performances and demands intricate post-processing. In this paper, we introduce a novel reinforcement learning model that can play the guitar using robotic hands, without the need for motion capture datasets, from input tablatures. To achieve this, we divide the simulation task for playing guitar into three stages. (a): for an input tablature, we first generate corresponding fingerings that align with human habits. (b): based on the generated fingerings as the guidance, we train a neural network for controlling the fingers of the left hand using deep reinforcement learning, and (c): we generate plucking movements for the right hand based on inverse kinematics according to the tablature. We evaluate our method by employing precision, recall, and F1 scores as quantitative metrics to thoroughly assess its performance in playing musical notes. In addition, we conduct qualitative analysis through user studies to evaluate the visual and auditory effects of guitar performance. The results demonstrate that our model excels in playing most moderately difficult and easier musical pieces, accurately playing nearly all notes.

CCS Concepts
• ***Computing methodologies*** → ***Animation; Reinforcement learning;***

1. Introduction

Human dexterous multi-fingered hands exhibit remarkable precision and agility across a myriad of tasks. However, emulating this level of finesse and dexterity in robotic hands presents multifaceted challenges, spanning from task comprehension to decision-making, and ultimately, precise task execution. Extensive research endeavors have been dedicated to various capabilities of robotic hands, encompassing in-hand manipulation for regular shapes [ABC*20, HAM*23] and its extension to diverse geometries [QKC*23, LFL*23], finger motions for object grasping [CKA*22, WGL22, CWL23], and the generation of finger movements for piano playing [ZSG*23]. Nonetheless, despite these advancements, existing methods still encounter significant hurdles in achieving parity with human capabilities. In this work, we aim to emulate human-like dexterous guitar-playing capabilities through precise control and motion planning for hands and fingers using both kinematic and physics-based methods.

For simulating the guitar playing process, there is a limited number of research [ES03] due to its complexity and precision, in stark contrast to the many studies on simulating guitar sounds [KCN24, DA20, WK23]. Traditional methods often rely on motion capture technology to convert the movements of real guitarists into data, which are then mapped onto virtual characters to simulate authentic playing actions. This approach not only heavily relies on motion capture data with limited generalization but also requires tedious post-processing to convert it into usable animations. To address this limitation, we introduce a new model for playing the guitar with robotic hands based on reinforcement learning and inverse kinematics [DVS01]. To the best of our knowledge, our work is the first to control robotics hands for playing guitar from an input of tablature, and thus, does not rely on any external data for motion control.

To create a realistic live performance using robotic hands, we ask the left hand to press strings on the guitar neck to change pitch and the right hand to pluck strings to produce sound. However, there are several challenges need to be addressed. For a given input tablature, correct finger allocations for each note are essential. From the assigned fingerings, the left-hand fingers must press their designated frets, mirroring human-like precision. Meanwhile, the right-hand fingers must synchronize their plucking actions precisely with the tablature to produce sound. Achieving this demands meticulous control and seamless coordination between the two hands.

In this work, we present a model for controlling guitar playing with robotic hands based on deep reinforcement learning and inverse kinematics. To arrange finger positions, we use a brute-force

[†] Corresponding author. E-mail: djhuang@shu.edu.cn

© 2024 Eurographics - The European Association
for Computer Graphics and John Wiley & Sons Ltd.

delivered by EUROGRAPHICS DIGITAL LIBRARY
www.eg.org diglib.eg.org

search algorithm to select the closest finger to the target fret. We also ensure that when multiple fingers are required, those on the left side handle higher strings and those on the right side handle lower strings. After generating fingerings for all notes, we develop a reinforcement learning algorithm and employ the *Dropout Q-Functions (DroQ)* [HIH*22] algorithm to learn control policies for controlling the fingers of the left hand. We then plan the trajectory for the plucking motions of right-hand fingers and leverage the inverse kinematics to compute their joint angles. Based on the frets pressing and plucking configurations, we generate guitar tones using the *Karplus-Strong* [Str83] algorithm. We quantitatively assess our method using precision, recall, and F1 scores, while qualitatively evaluating it through a user study focusing on the visual and auditory aspects of guitar performance. The results demonstrate that our model excels in playing most moderately difficult and easier musical pieces, accurately playing nearly all notes. The source code of our method is publicly available at `https://github.com/MRXuanL/GPS-GuitarPlaySimulation`.

2. Related Work

In the realm of robotics and computer graphics, simulating human dexterity skills presents a formidable challenge. While some models focus on directly generating outcomes [RBL*21, LXJ*23, WWY23], the core of simulating these skills lies in crafting controllers capable of guiding characters through diverse activities and decision-making in various scenarios. In this section, we explore the primary methods employed to generate such controllers. One significant subdivision within this field is instruments playing simulation. Due to the high level of difficulty inherent in this domain, only a handful of researchers have ventured into it.

2.1. Human Dexterity Skill Simulation

2.1.1. Traditional Methods.

Human dexterity skills are very complex and challenging to simulate accurately. Since the advent of motion capture technology, researchers have begun utilizing this technique to capture the real-world skill processes of humans and map them onto controllers to reproduce human skill activities. However, animations generated using this method often remain confined to specific segments, making it difficult to flexibly apply them to other scenarios. Therefore, trajectory optimization-based methods have emerged, offering new avenues to address this issue [KLXvdP21, MWTK13, WPP14]. However, when facing high-dimensional spaces, this approach often requires significant computational resources and time, making it unsuitable for real-time scenarios. By employing inverse kinematics [DVS01, ALCS18], the method offers strong real-time capabilities but lacks natural fluidity.

2.1.2. Motion Capture-Based Methods.

With the continuous advancement of machine learning techniques, reinforcement learning has opened up new avenues in the field of skill simulation, such as imitation learning [BCHF19, CMM*18]. Zhang et al. [ZYM*23] utilized tennis match videos to master diverse tennis skills, while Xie et al. [XSLVDP22] enabled characters to perform complex football juggling skills. Hassan et al. [HGW*23] trained a controller through imitation learning on motion capture data from the *SAMP* dataset [HCV*21], enabling characters to interact with the scene, lift heavy objects, move around, and assume different poses on furniture. However, this method had limitations as it was confined to skills in the dataset, lacking generalization to broader contexts. A recent method [PGH*22] utilized imitation learning to pre-train a strategy on a large-scale dataset of simple human actions and applied these strategies to more complex tasks such as running, hitting, turning, and kicking. Won et al. [WGH22] proposed an algorithm that utilizes conditional variational autoencoders (VAEs) [VK20] to learn various behaviors similar to those in the training dataset, enabling the execution of downstream tasks such as maze navigation and path following.

2.1.3. Data-Independent Methods.

The primary drawback of motion capture-based methods lies in the scarcity of datasets. Without a suitable dataset as a foundation, it becomes challenging to simulate similar strategies. Hence, skill simulation methods independent of motion capture data exhibit a distinct advantage. These approaches leverage reinforcement learning principles, rewarding actions aligning with predefined standards while penalizing those that deviate, thereby incentivizing the agent to perform desired skill actions, such as locomotion [PVDP17] and using chopsticks [YYL22]. However, mastering highly complex actions using this method may pose challenges. To address this issue, a curriculum-based learning approach has emerged. This method typically designs multi-stage learning curricula based on the characteristics of the skill [CEG*20, XLKvdP20], aiming to systematically accomplish task learning. Tao et al. [TWGvdP22] proposed a three-stage reinforcement learning framework, which meticulously shapes the reward mechanism for characters to get up from the ground, ultimately enabling the characters to display diverse postures of standing up after continuous learning. Given the current scarcity of guitar-playing datasets, we employ a reinforcement learning method independent of data and divide the learning task into multiple stages for sequential learning and optimization.

2.2. Instrument Playing Simulation

Instrument playing simulation mainly comprises sound simulation and motion simulation. Among them, sound simulation has garnered significant attention. In the early stages, sound simulation primarily relied on physics-based methods, including additive, subtractive, and phase modulation synthesis [All80, Cho73]. However, in recent years, deep learning techniques have gradually been employed in this field to achieve more realistic sound simulation [KNK*22, SY16, DZBKM22]. We employ the *Karplus-Strong* [Str83] algorithm to simulate the sound of the guitar, providing adequate sound support for our guitar-playing simulation. Motion simulation, on the other hand, focuses on simulating the process of playing musical instruments. However, due to its difficulty and complexity, few people have ventured into or mentioned motion simulation of playing instruments. In the early days, people attempted to simulate playing musical instruments using both humanoid and non-humanoid hardware machines [BW16]. These machines typically depended on specific control programs, lacking versatility. While some non-humanoid machines could play instruments, their style differed greatly from human playing, emphasizing

© 2024 Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd.

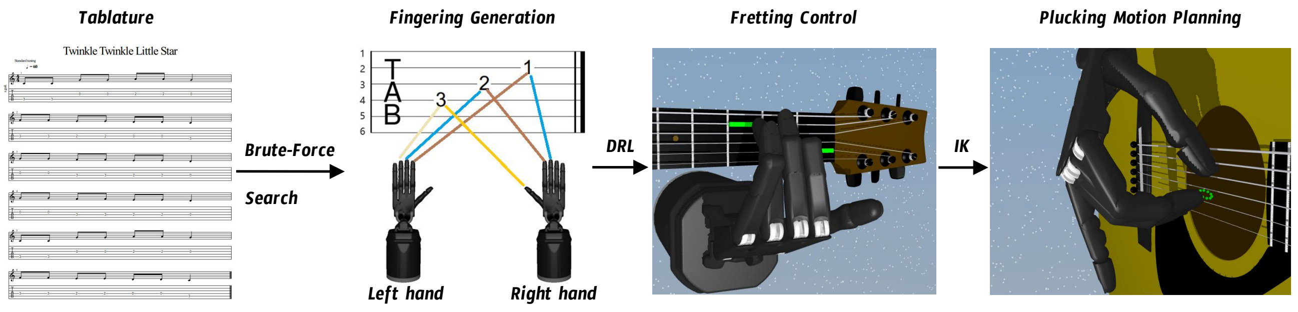


Figure 1: *System overview. For a given input tablature, our system utilizes a brute-force algorithm to generate the fingering for playing the guitar. Subsequently, it employs Deep Reinforcement Learning (DRL) to train a policy to control the left hand pressing the strings, while using Inverse Kinematics (IK) to control the right hand plucking the strings.*

auditory output rather than authentic performance. We recommend a review paper [KdlCCP*24] which provides an extensive analysis of both sound and motion simulation techniques.

In the field of instrument playing simulation, significant attention is directed towards simulating pianos [LC13, ZMM11, XLW*22]. Researchers such as [Yeo21] utilized inverse kinematics to plan the finger trajectories of a robotic hand for piano playing, while others, like [ZWS*23], developed a reinforcement learning environment specifically for piano playing. However, in the field of guitar playing simulation, current research primarily relies on mechanical means for guitar playing [KK19] or employs traditional methods such as motion capture technology or inverse kinematics to simulate guitar performance. The prior art of guitar-playing simulations like ElKoura and Singh [ES03] built a hand motion dataset and employed the K-nearest neighbors algorithm to reconstruct hand postures from the dataset, avoiding the unnatural gestures of inverse kinematics (IK). This method relies on the captured dataset to drive finger motions through interpolation, which can result in unnatural movements and poses challenges when required motions fall outside the dataset. Additionally, their approach models the fretting motions of fingers by kinematically guiding them to specified positions without accounting for real-world physical dynamics.

In contrast, we present a guitar-playing simulation model with reinforcement learning for the first time, to the best of our knowledge, capable of reading tablature and, after training, guiding robotic hands to play the guitar with more realistic and natural motions without the need for pre-existing datasets.

3. System Overview

Our goal is to generate an automated control strategy for playing guitar with robotic hands from input tablatures. To this end, we develop a system that contains three main components as shown in Figure 1. For readers who may be unfamiliar with terminology in playing guitar, a brief explanation is provided in Appendix A.

Fingering generation. Our aim is to ensure that each note in the guitar tablature is assigned to a finger for playing while avoiding conflicts between fingers. For instance, no finger should handle two notes simultaneously at any given moment (excluding barre chords). Additionally, we strive to generate fingerings that are both simple and in line with human playing habits.

Fretting control. After generating fingerings, we determine the fret for each note along with its corresponding fingering. Consequently, when a note needs to be played, we must ensure that the robotic hand swiftly approaches and accurately presses the corresponding fret with its fingertips. To achieve this goal, we employ reinforcement learning techniques to control the left hand's fretting actions, ensuring the accuracy and fluidity of the performance.

Plucking motion planning. In order to generate plucking motion for the right-hand fingers, we first design the trajectory of plucking motion for the tip of the fingers. Based on the trajectory, we then employ inverse kinematics to compute the joint angles for the fingers of the right hand, enabling precise control during plucking.

4. Fingering Generation

After inputting a tablature, we must ensure that each note is played with coordination between the fingers of both hands. There are three issues to be addressed: (1) no single finger can play two notes simultaneously, (2) finger assignments should minimize energy expenditure, meaning that the distance the fingers need to travel to reach the designated fret should be minimized, and (3) finger assignment should align with human playing habits.

To address the first issue, we sort the notes according to their start times and then iteratively assign fingers to each note. Each finger has a last usage time, and if the start time of a note is earlier than the last usage time of a finger, it indicates that the finger is still assigned to another note and cannot be used. In this case, we must choose another finger. If a finger can be assigned to the current note, its last usage time is updated to the start time of the note plus the duration of the note to avoid conflicts.

To minimize energy expenditure for the current given note, we seek to assign the closest finger to the target fret position. To this end, we keep recording the assumed hold position in real time as P. The hold position refers to the location on the fingerboard where the index finger of the left hand is placed. Based on the proximity of fingers to this holding position, it is natural to set the fret positions

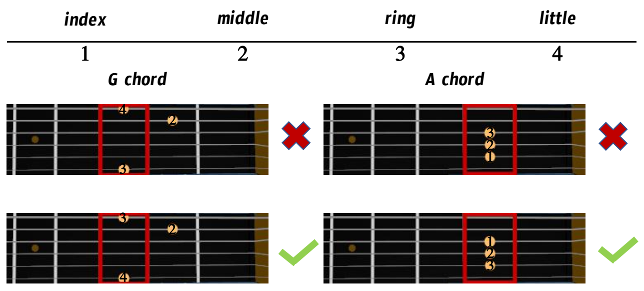


Figure 2: *Fingers are numbered from 1 to 4. When playing the G chord, it is typical to use the little finger to hold down the first string and the ring finger to hold down the sixth string, rather than the other way around. Similar to the case of the A chord, the index, middle, and ring fingers are used to hold down the fourth, third, and second strings, respectively.*

P_i with $i \in 1,2,3,4$ for the index finger, middle finger, ring finger, and little finger as P, $P+1$, $P+2$ and $P+3$, respectively. Given the target fret position P_t for the current note, we find the closest finger to the target as $\arg\min_i(|P_i - P_t|)$. This can be efficiently computed by leveraging the brute-force search among four fingers.

While the above criteria assign correct fingers to the target fret positions according to the given note, they ignore human playing habits. When multiple notes share the same fret, considering the G and A chords as illustrated in Figure 2, it is typical to designate fingers on the right side of the hand to play the lower-positioned notes, and fingers on the left side to play the higher-positioned notes. Therefore, in such cases, we sort currently available fingers and follow the above preference for the assignment of fingers.

Based on the above operations, we can effectively allocate fingers for each note. This allows us to guide the fingers of robotic hands to press the correct fret for a note. We then use this as the training data for controlling the robotic hands with reinforcement learning.

5. Fretting Control

After generating fingering, we know the exact position of the fingers to be placed. However, the orchestration of finger movement, encompassing rotation of joints and translation of the left hand, remains a challenge. To address this, we train a neural network with reinforcement learning to develop a precise fretting strategy for robotic hands within a virtual simulation environment. In this environment, we attach touch sensors to all frets of every string to precisely detect the pressing of frets during simulation as shown in Figure 3. These frets are considered as the press status when their force exceeds the given threshold. With this virtual environment setting, it is ready for us to simulate robotics for training.

In reinforcement learning, the design of effective reward functions stands as a pivotal element, shaping the decision-making process for robotics hands and influencing the attainment of desired outcomes. Our designed reward function contains three key components: r_{finger}, r_{key}, and r_{energy}. We describe each component as follows.

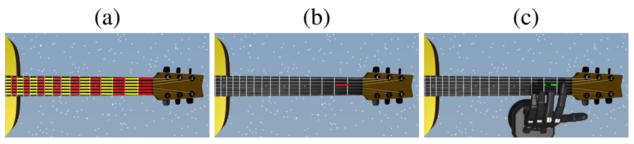


Figure 3: *(a) displays the locations and sizes of all the added touch sensors, (b) indicates the target fret (red) that needs to be pressed with a finger, and (c) reflects which fret is currently being pressed (green) by the finger in real-time.*

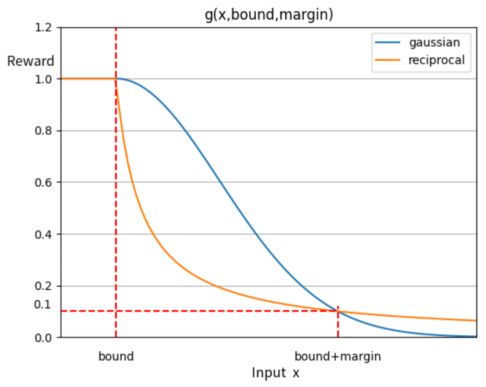


Figure 4: *$g(x, bound, margin)$ is a mapping function in dm_control. The reward reaches its maximum when the value of x is in the range of 0 to bound, and decreases to 0.1 when x is equals bound + margin. We will use the reciprocal function as the mapping function and compare its effect with the Gaussian function in the training process.*

Finger reward. This term, r_{finger}, is intuitively designed to incentivize movements of the fingers toward the target fret. Since each guitar fret has a designated pressing area, any finger that falls within this area is considered to have successfully pressed the string. Therefore, instead of using Euclidean distance as in [ZWS*23], we define the reward for the finger in three components of the fingertip position, allowing the robotic hand to locate the target fret more efficiently, i.e.,

$$\begin{aligned} r_{finger} &= \frac{r_x + r_y + r_z}{3} \quad \text{with} \\ r_x &= g(|f_x - k_x|, 0.5L_{key}, 0.1)\ , \\ r_y &= g(|f_y - k_y|, 0.001, 0.1)\ , \\ r_z &= g(|f_z - k_z|, 0.001, 0.1)\ , \end{aligned} \tag{1}$$

where g is a mapping function that converts reward values to the range of 0 to 1, as shown in Figure 4. (f_x, f_y, f_z) and (k_x, k_y, k_z) represent the position of the fingertip and the center position of the fret, respectively. L_{key} stands for the length of the target fret.

Key reward. To explicitly encourage the fingers to maintain pressure on the target fret while penalizing incorrect presses, we further introduce a key reward as

$$r_{key} = 0.5\Big(\frac{1}{K}\sum_{i}^{K} g(|ks-1|, 0, 1)\Big) + 0.5(1 - \mathbf{1}_{\{false\ positive\}})\ , \tag{2}$$

where K represents the number of frets that need to be held at the moment. ks denotes the normalized force applied to the frets. $\mathbf{1}_{\{false\,positive\}}$ equals 1 if any prohibited fret is held, otherwise it is 0. It is worth noting that while sound can only be generated on a guitar by plucking strings with the right hand, fingers are allowed to press frets where strings are not plucked.

Energy reward. The above rewards allow fingers to press correct frets, whereas we observed that when the right hand is plucking the strings and no string is required to press for the left hand, the left hand may move unnecessarily. To avoid these scenarios, we introduce an energy reward r_{energy} to restrict the torque and useless translation used by the hand, promoting more stable movements rather than rapid and unpredictable ones. We establish an energy penalty mechanism to ensure that the fingers can press the strings more stably with the form

$$r_{energy} = -k|\tau_{joints}|^T|\mathbf{v}_{joints}| \ , \tag{3}$$

where τ_{joints} is a vector of joint torques and $\mathbf{v}_{joints}$ is a vector of joint velocities, $|*|$ outputs a vector contains the absolute value of each element in the original vector, and $k = 0.0025$ is a coefficient. For cases where no frets need to be held by the left hand, we increase k to 0.005.

With all the defined rewards, we combine them to form the final reward function in the reinforcement learning as

$$r = 0.5r_{key} + 0.5r_{finger} + r_{energy} \ . \tag{4}$$

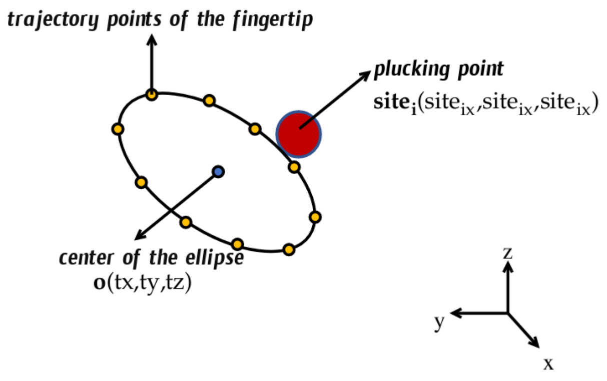

trajectory points of the fingertip
plucking point
site_i(site_ix,site_ix,site_ix)
center of the ellipse
o(tx,ty,tz)
z
y
x

Figure 5: *The generation of the trajectories for plucking the strings 1, 2, and 3.*

6. Plucking Motion Planning

During guitar playing, coordination between the left and right hands is crucial. The left hand presses the strings while the right hand plucks them. Following typical guitar playing technique for plucking, we assign the thumb to handle strings 4, 5, and 6, while the index, middle, and ring fingers are responsible for strings 3, 2, and 1, respectively. Through observing guitarists during their performances, we observe that the motion trajectory of the fingertips of the index, middle, and ring fingers when plucking strings resembles an ellipse, whereas the trajectory of the thumb when plucking strings is closer to a straight line. These indicate that plucking motions are relatively consistent trajectories, allowing us to choose a precise trajectory planning for the plucking motion of the fingers.

For string plucking with the given plucking point, we employ ellipses and lines trajectories as the path for the fingertip. For the plucking motion planning of the thumb finger, it is straightforward to determine a line between the current tip position of the thumb and the target plucking point. For other fingers with an elliptical trajectory, we first define an ellipse trajectory at origin as $\mathbf{p} = (0, a\cos\theta, b\sin\theta)$, where coefficients are $a = 0.005$ and $b = 0.0025$. With this elliptical trajectory at origin, we compute a rigid transformation $\mathbf{p}' = \mathbf{T} + \mathbf{R}\cdot\mathbf{p}$ such that the trajectory passes the given plucking points $site_i$ for a target guitar string with a proper tilt of the hand to avoid colliding with other strings as illustrated in Figure 5, where the translation and rotation are $\mathbf{T} = \text{site}_i + (0, 4.33e^{-3}, -2.5e^{-3})$ and $\mathbf{R} = \mathbf{R}_z(-\pi/2)\mathbf{R}_y(-\pi/9)\mathbf{R}_x(\pi/9)$, respectively.

Upon completing the plucking path planning, we apply inverse kinematics to compute the angles for each finger joint.

7. Experimental Setup and Result

We start with a detailed description of our experimental setup, mainly including the tablature dataset, the parameters specifications for the training process, and the evaluation metrics for our system, after which we demonstrate the performance of our guitar-playing model via monophonic and polyphonic pieces. Additionally, we conduct comparative analyses of the results across different pieces.

7.1. Experimental Setup

We focus on several key aspects in the experimental setup, including the tablature dataset, observation and action space, environment wrapping and training, sound generation, evaluation methods as well as parameter configuration.

Tablature dataset. We have created a class called 'Tablature' specifically for representing guitar tablature, which is built on a list of musical notes containing information such as duration, start time, fret position, and the finger used to play. Using this class, we have compiled eight training pieces to comprehensively assess the model's performance on different types of tablatures. These pieces include 'Chord Transition,' which tests polyphonic playing skills, the monophonic exercise 'Scales Practice,' and moderate-difficulty pieces like 'Twinkle Twinkle Little Star,' 'Für Elise,' and 'Long Long Ago,' which consist of monophonic tablatures. Additionally, we have covered pieces with polyphonic tablatures such as 'Farewell,' 'Happy Birthday,' and 'Red River Valley,' showcasing various types of tablatures that beginners can play. This series of piece designs aims to systematically evaluate the robotic hand's ability to play guitar tablatures.

Observation and action space. State encompasses all information about the environment, while observation perceives the agent as information it can sense. Within the state, we provide data such as whether guitar frets are pressed, joint velocities, positions, etc., all displayed in Table 1. From these, we select a subset as observations of the environment. It is worth noting that "string state" not only

Table 1: *The state, action, and observation space*

State space	Unit	Size
Hand joint positions	rad	21
Hand joint velocities	rad/s	21
Forearm positions	m	3
Forearm velocities	m/s	3
Last action	–	20
Last reward	–	1
Guitar keys(frets) state	discrete	120
Finger state(goal)	m	15
String state(goal)	discrete	120(n+1)
String state(simplified goal)	discrete	6(n+1)

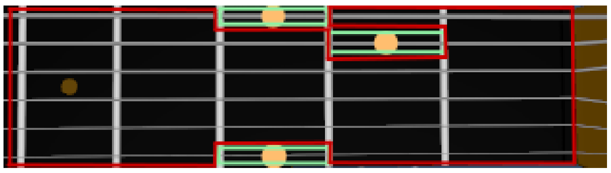
Action space	Unit	Size
Desired hand joint angles	rad	17
Desired forearm positions	m	3

Observation space	Unit	Size
Hand joints positions	rad	21
Forearm positions	m	3
Guitar keys(frets) state	discrete	120
Last action	–	24
Last reward	–	1
Finger state(goal)	m	15
String state(goal)	discrete	120(n+1)
String state(simplified goal)	discrete	6(n+1)

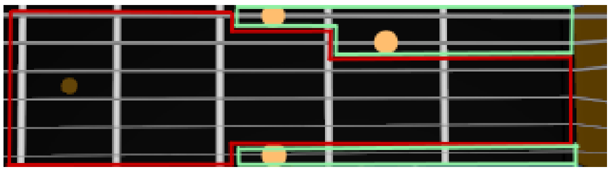
covers the current target string status but also extends to the target string status for the next *n* ticks. The value of *n* determined by the parameter "n_step_lookahead" in Table 2 aims to enable the robotic hand to anticipate in advance. For the target string status in "string state," we provide two options: one representing the target values of each key as 0 and 1, while the other is a simplified version indicating the target frets to be pressed on each string. Another aspect we consider is the "finger state," which involves information about the current availability of the fingers and their distance to the responsible fret positions. For the action space, we use the position servo in *MuJoCo* [TET22] to control the target angles reached by the joints of the robotic hand in real-time; see Appendix B for specific details on the action space.

Environment wrapping and training. Many open-source reinforcement learning algorithms operate on *gymnasium* [TTK*23] environments. To facilitate users in training guitar-playing models, we have specifically wrapped the guitar-playing environment according to gymnasium standards. In terms of algorithm selection, we prioritize the *DroQ* algorithm from the *sbx* [RHG*21] library (an efficient implementation of stable baselines 3 in *jax* [BFH*18]). The *DroQ* algorithm, an improvement on the *SAC* [HZH*19] algorithm, offers higher sampling efficiency. By employing the *DroQ* algorithm, we can more efficiently optimize the guitar-playing model, enabling it to demonstrate superior performance in simulated environments.

Sound generation. To test the performance of our model, we need to simulate the process of the guitar producing sound based



(a) *Freedom: False*



(b) *Freedom: True*

Figure 6: *We use the G chord as an example to demonstrate the finger pressable areas with freedom set to true and false. The area surrounded by red indicates no pressing, while the area surrounded by green allows pressing.*

Table 2: *Parameter for training*

Env	Description
n_step_lookahead	lookahead horizon
n_env	synced training envs
table	tablature to learn
sigmoid	mapping function
normalization	normalize the states
freedom	boost hand freedom
frame rate	frame rate of env

DroQ	Value
batch_size	512
dropout_rate	0.01
start_step	50000
total_step	5000000
use_noise	true
seed	42
buffer_size	1000000
learning_rate	0.0003

on plucking actions. We bind a *PyAudio* [Pha06] audio stream for each string of the guitar, serving as a medium for music playback. When the fingertip of the right hand approaches the plucking point, we will determine the corresponding pitch frequency based on the maximum fret pressed on the current string, following the principle of guitar sound production. We will then utilize the *Karplus-Strong* algorithm to simulate the sound waves of the guitar and play them through the corresponding *PyAudio* audio stream for each string.

Evaluation methods. We use F1 score, recall, and precision as metrics to evaluate guitar playing performance. Specifically, precision reflects the proportion of correct frets pressed by the performer when holding down a fret, while recall indicates the proportion of correct frets pressed among all the correct frets in the total target. The F1 score serves as a balance between precision and recall, de-

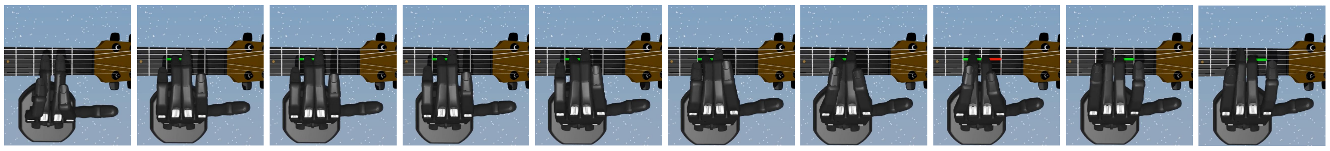


Figure 7: *The left hand transitions from pressing down on the 4th string and the 3rd fret (F) to the 4th string and the 2nd fret (E).*

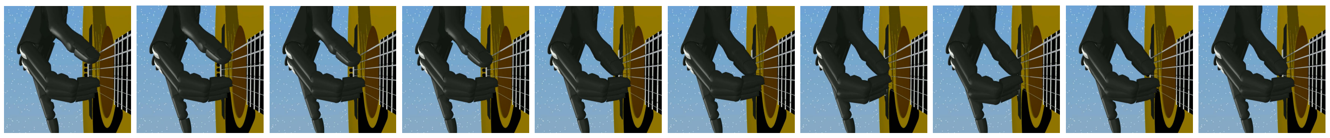


Figure 8: *The process of plucking strings with the thumb and the other fingers.*

fined as $F1 = 2 \times \frac{\text{Precision}\times\text{Recall}}{\text{Precision}+\text{Recall}}$. It is worth noting that pressing a fret incorrectly on a guitar, without plucking, doesn't affect the overall performance, unlike the piano. Therefore, a higher recall often implies a potentially better actual performance of guitar playing.

Parameter configuration. During the training process, we offer various parameters for user selection, all of which are detailed in Table 2. Among them, the "freedom" parameter is a particularly emphasized option. This parameter is based on the acoustic principle of guitars. The sound produced by plucking a string is solely related to the highest fret pressed on the string even if some lower frets have been pressed. When setting it to 'False', the robotic hand can only hold down the designated frets at each moment, similar to the playing strategy of a piano. This approach can enhance precision but may reduce recall. If it sets to 'True', the robotic fingers can hold down frets lower than those required, while prohibiting frets higher than those required. Enabling this parameter will increase the freedom level of the robotic hand, however, it may decrease precision and help improve recall. We implement this by punishing the situation where fingers press forbidden frets in the key reward, thus restricting the left hand to only press allowable frets. Figure 6 depicts the areas where pressing is possible when the freedom parameter is set to 'True' and 'False'.

7.2. Result

We utilize a computer equipped with an Nvidia GeForce RTX 3070 GPU and an AMD Ryzen 5 5600 CPU for model training. To improve data collection efficiency, we employ the *sbx* library with 4 threads enabled for synchronous training. This results in an average training time of 6 hours per song, with 5,000,000 training iterations.

Basic playing ability. As a simple example, we choose the melody 'Long Long Ago' as a test piece to evaluate the performance of our model in left-hand fretting and right-hand plucking. As shown in Figure 7, 8 for fretting and plucking actions, respectively, our model can generate outstanding results, with the left hand accurately and swiftly positioning and fretting according to the target notes, while the right hand synchronously plucked the strings with the corresponding fingers; please also see the supplemental video.

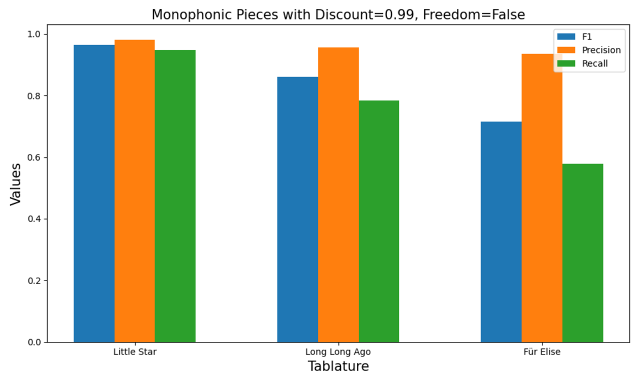


Figure 9: *Performance of monophonic pieces. We found that the precision of monophonic pieces is generally very high, meaning that the model can accurately press down the right notes for the most part. However, for the piece 'Für Elise,' the recall is slightly lower because of its fast speed.*

Performance of monophonic pieces. In the performance of monophonic pieces, players are required to press only one fret with their left hand at each moment. To delve deeper into its playing effects, we select several classic melodies, including 'Long Long Ago', 'Für Elise', and 'Twinkle Twinkle Little Star' as examples, and set the freedom parameter to false for improved precision. Figure 9 illustrates the specific metrics of the monophonic pieces. Through observation and testing, we noticed that the speed at which melodies are played significantly affects the playing effect of the robotic hand. When playing at a slower pace, the robotic hand accurately presses each fret, delivering an exceptional performance. However, as the playing speed increases, the performance of the robotic hand noticeably deteriorates. This is particularly evident in the faster-paced 'Für Elise', where the robotic hand struggles to rapidly and accurately press the frets, resulting in a lower F1 score. These faster-paced pieces require the robotic hand to move swiftly while maintaining a high level of accuracy. However, the physical limitations of the actuators controlling the joints of the robotic hands, as implemented by the position servos in *MuJoCo* [TET22], are constrained by the

© 2024 Eurographics - The European Association
for Computer Graphics and John Wiley & Sons Ltd.

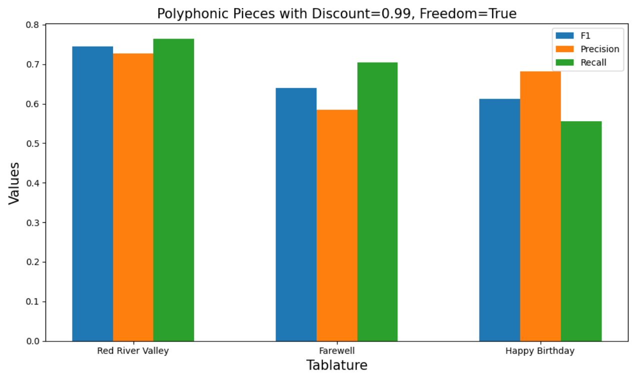


Figure 10: *Performance of polyphonic pieces. We observe that the performance of polyphonic pieces is indeed slightly inferior to that of monophonic pieces, but their F1 scores still hover around 0.6. In terms of recall, the scores for these polyphonic pieces are all above 0.6, so in the overall performance, although there are some flaws, it is still possible to recognize the corresponding pieces with some effort.*

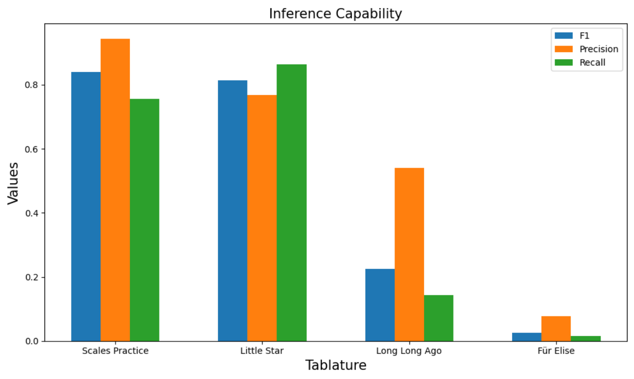


Figure 12: *We utilize a controller trained with 'Scales Practice' as the target to play other monophonic pieces. During testing, we found that 'Little Star' performed well and had a similar tempo to scales practice. However, when attempting to play faster-paced pieces like 'Long Long Ago' and 'Für Elise', the performance appeared less satisfactory.*

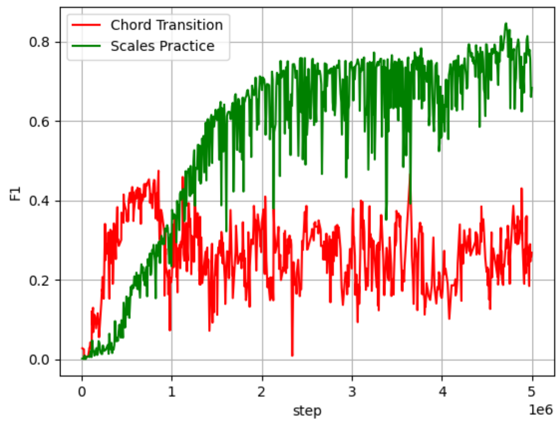


Figure 11: *'Scales Practice' and 'Chord Transition' showcase the model's ability in playing monophonic and polyphonic pieces. Upon observation, it becomes evident that the model performs better when playing monophonic pieces compared to polyphonic ones.*

maximum allowed torque due to the inertia of the robotic hands. In addition, the simulation frame rate is another factor contributing to this phenomenon. In Figure 9, the same frame rate is used to simulate different pieces. For faster passages, the relative frame rate is effectively lower compared to slower segments. This relatively lower frame rate reduces the opportunities for the robotic hand to accurately adjust and press the specified frets. As demonstrated in Figure 16, increasing the frame rate is indeed able to increase the performance of our method.

Performance of polyphonic pieces. Playing polyphonic pieces presents a more challenging task for the robotic hand, as it necessitates the ability to maintain multiple pitches simultaneously. We experiment with songs such as 'Happy Birthday,' 'Farewell,' and 'Red River Valley' to assess the proficiency of robotic hands, and the specific metrics are shown in Figure 10. In an effort to enhance its performance in playing these complex songs, we adjust a key setting—setting "Freedom" to 'True'. This adjustment aimed to grant the robotic hand greater freedom, with the hope of achieving more accurate note reproduction. However, when compared to monophonic pieces, the performance of the robot noticeably declines with polyphonic ones. Most often, its F1 score remains around 0.6, indicating frequent omission of notes and an inability to faithfully replicate the original melody. While the robotic hand may still have inherent limitations, we are actively striving to bridge the gap and improve its performance to approach that of human capability.

Performance of scales practice and chord transition. We evaluate the performance of 'Scales Practice' and 'Chord Transition' to assess overall monophonic and polyphonic playing abilities, respectively. These two skills were chosen to represent the difficulty levels of monophonic and polyphonic scores. The performance of the model on these types of scores largely reflects its ability to play monophonic and polyphonic pieces, as illustrated in Figure 11.

Inference capability. To assess the inference ability of our model, namely its performance after inputting tablature using a pre-trained model, we specifically chose 'Scales Practice' for training piece, while adopting a series of monophonic pieces for testing purposes. It can be seen from Figure 12 that when the tempo of the music sheets remains constant, the model demonstrates commendable inference capability. However, once the tempo of the piece changes, the performance of the model relatively declines, resulting in less accurate playing.

Compare with prior art. We compare the motion quality of our model to the method described in Handrix [ES03], using the same fingering. We select two passages from their results video: the chord transitions and the complex passages. The results show that our method produces more natural motions compared to those generated by the Handrix model. However, for the complex passages, our model exhibits slightly less precision. The Handrix model achieves high precision by using an IK-like method to keep the finger close to the target fret. Since their method reproduces the playing motion by considering only kinematics without dynamics, their results lack the ability to replicate human-like, vivid playing motions. For detailed comparisons, please refer to our supplementary video. Note that to maintain consistent actions, we adjusted the playback speed of some parts of the video, as we do not know the exact tempo in the Handrix video.

User study. We conduct a qualitative assessment of our simulation results with 58 participants, including 27 guitar learners and 31 non-learners, predominantly university students. We designed a detailed questionnaire to evaluate visual and auditory effects, instructional significance for beginners, and potential virtual applications, all rated on a 5-point scale. The statistics of the rating are summarized in Table 3. Most participants expressed positivity toward the simulation, seeing it as beneficial for beginners and applicable in virtual environments. However, participants with guitar learning experience gave slightly lower ratings, possibly due to higher expectations for accuracy.

Table 3: *Rating of the user study.*

Assessment Indicators	Learned		Not Learned	
	Avg	Var	Avg	Var
Fretting Action	4.15	0.57	4.34	0.48
Plucking Action	4.07	0.59	4.19	0.74
Musical Auditory	4.19	0.52	4.16	0.45
Beginner Guidance	4.30	0.50	4.26	0.58
Virtual Potential	4.41	0.32	4.32	0.54

8. Ablation and Comparison

To validate the correctness of the design of our system, we select various parameters for comparative analysis.

Mapping function. We chose "reciprocal" as the mapping function because finger movements in guitar playing require extreme precision. Even slight movements can result in pressing the wrong fret, so we opted for the "reciprocal" mapping function. This function is characterized by increased sensitivity to changes in rewards near the target. Figure 13 illustrates the comparison between the "reciprocal" and "Gaussian" mapping functions during the training of 'Little Star'. This choice helps the model better capture subtle variations in finger movements, thereby improving playing accuracy.

Finger reward. During the training of 'Little Star', we begin by attempting to utilize the Euclidean distance as the metric for the reward, but the results are unsatisfactory, with an F1 score consistently around 0.6. To enhance training effectiveness, we adopt a new strategy introduced in Sec.5. The advantage of this method lies in relaxing the conditions for fingers to receive full rewards, allowing them to obtain full scores at different positions within the same fret, rather than being restricted to the center of the fret. From the comparison shown in Figure 14, we can evidently see that the presented reward leds to a significant improvement in training effectiveness.

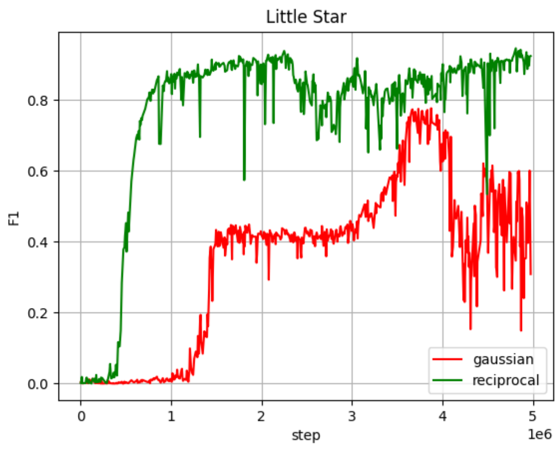


Figure 13: *It's evident that when using "reciprocal" as the mapping function, the convergence speed is notably faster compared to the "Gaussian" mapping function, and the overall performance is also better.*

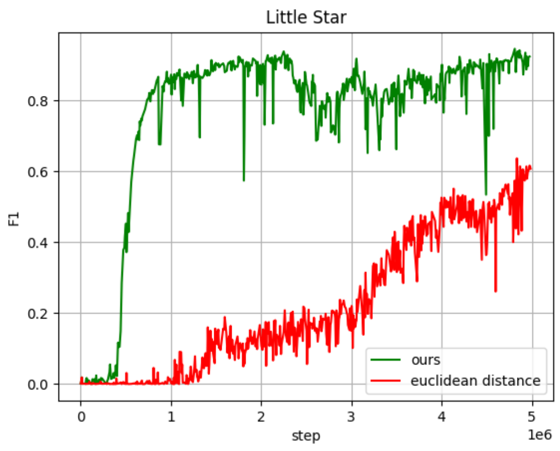


Figure 14: *When using Euclidean distance as the metric for finger reward, the convergence speed is very slow. However, when we adopt our method as the metric for finger reward, namely relaxing the conditions for fingers to achieve full rewards, the convergence speed significantly improves, and the overall performance is better.*

Freedom. The "freedom" parameter plays a crucial role in optimizing robotic guitar performance. In Figure 15, we illustrate this by comparing the performance of 'Für Elise' with the "freedom" parameter set to 'True' and 'False'. We observe a significant increase in recall when "freedom" is set to 'True', thereby enhancing the completeness of the performance. Although this adjustment may result in a slight decrease in precision, its impact on the auditory perception of guitar performance is not significant. Therefore, reasonably adjusting the "freedom" parameter allows for a better representation of the completeness of the piece while maintaining performance quality.

© 2024 Eurographics - The European Association
for Computer Graphics and John Wiley & Sons Ltd.

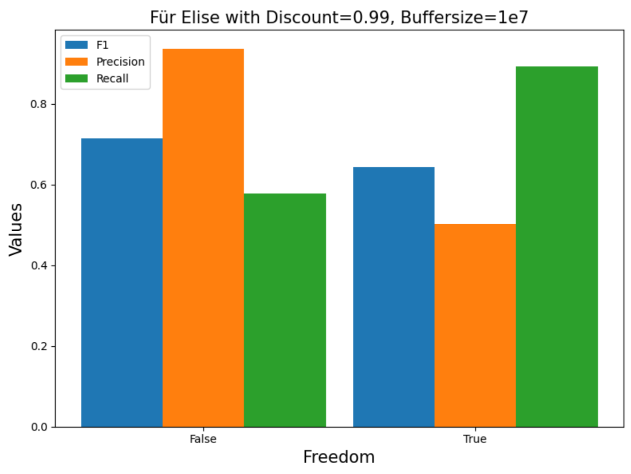


Figure 15: *When we set the freedom parameter to true, the recall index of our model presents a significant improvement, greatly enhancing the integrity of the model's performance. Although the precision rate decreased slightly, it did not have a negative impact on the overall performance of the guitar.*

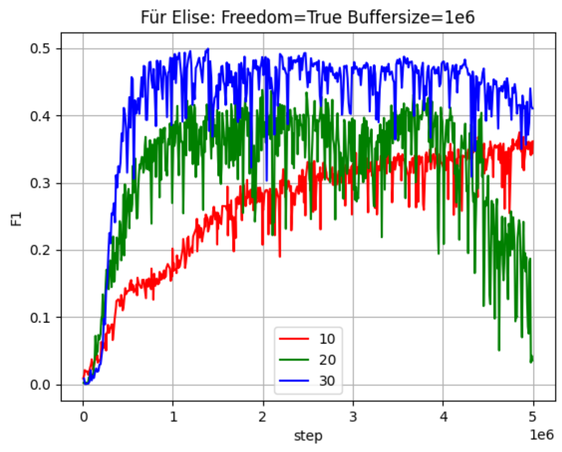


Figure 16: *Testing under different frame rates, i.e., 10, 20, and 30 frames per second. The results show that higher frame rates indeed contribute to improving performance. However, excessively high frame rates may also lead to instability during training.*

Frame rate. A higher frame rate implies a greater frequency at which the model controls the robotic hands, enabling it to perform more precise operations. To validate this hypothesis, we test the performance of the robotic hands playing the guitar at various frame rates. It can be seen from the Figure 16 a high frame rate does indeed lead to better performance, but it may also introduce instability.

Discount. We further explore the impact of the discount parameter in reinforcement learning. Initially, experiments used a discount value of 0.99. However, adjusting it to 0.84 led to significant improvements in outcomes, with almost all previously tested training pieces achieving recall values above 0.85. Figure 17 shows the training outcomes of 'Happy Birthday' with different discount parameters. We conjecture that this improvement is because the movements of robotic hands are primarily affected by the reward of the current note, with lower impact on subsequent notes. Hence, setting a high discount value may be unnecessary in this scenario. To further validate the model, we introduced more challenging pieces for testing, and Figure 18 displays the training outcomes of all pieces, sorted by recall values, using a discount value of 0.84.

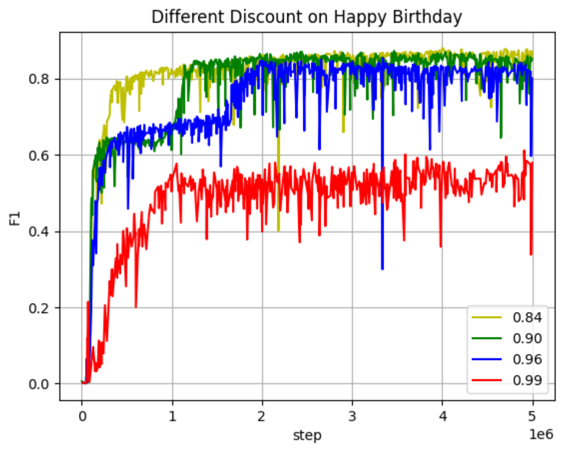


Figure 17: *The results indicate that using a lower discount value can accelerate the convergence process and relatively achieve higher effectiveness.*

9. Conclusion and Discussion

We developed a guitar-playing model for robotic hands, which generates precise controllers through training and can play the guitar directly from a tablature. We achieved this by first introducing a fingering generation algorithm based on a brute-force search with only the input of tablature. Based on generated fingering, we constructed a reinforcement learning algorithm for playing the guitar, effectively motivating the robotic hands to accurately press the target frets by setting up reward mechanisms. Furthermore, we utilized path planning to generate plucking movements for the right-hand fingers, and then used inverse kinematics to accurately calculate the joint angles for each moment.

Our guitar playing model demonstrated excellent performance in both simple monophonic pieces and polyphonic pieces. For more complex compositions, the majority achieved a recall rate of over 0.7, indicating that most machine-played pieces can be recognized by the audience. However, the model has not entirely replicated the strategy of human left-hand fingering. Humans, while playing, anticipate and press the frets in advance according to the tablature to ensure the coherence between notes. Additionally, human performers use their left thumb to anchor the back of the guitar to prevent displacement. However, since our guitar model is fixed in position and does not face displacement issues due to physical factors, the role of the thumb has not been accounted for. To avoid unnecessary interference with model training caused by collisions, we ensured that the thumb

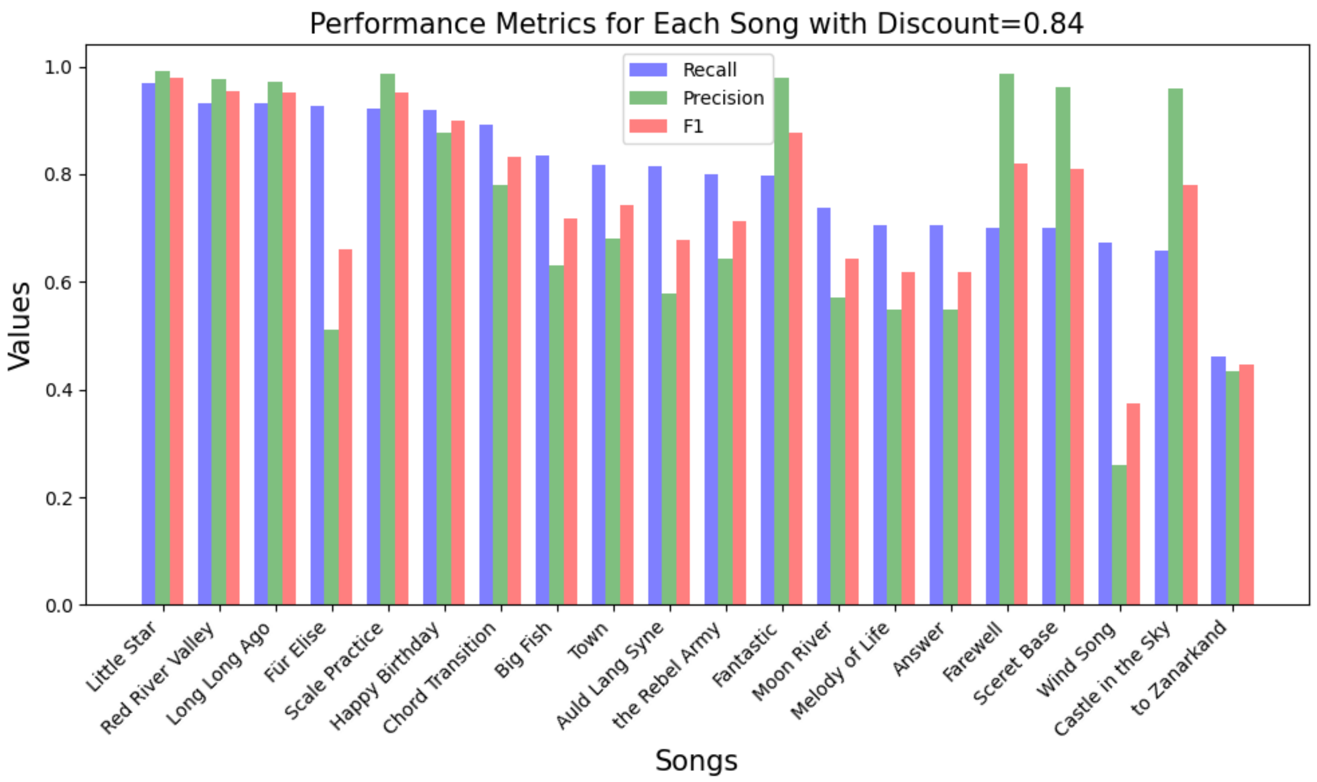


Figure 18: *We collect multiple MusicXML [mus24] pieces online as a training dataset. After testing, it is satisfying that most pieces achieve a recall rate above 0.7, which means the majority of played pieces are accurately identified by the audience.*

remains motionless. In the future, we will further optimize the model to approach human playing strategy more closely.

We discovered that the range of motion of each joint in the robotic hand significantly affects the performance of our model. Through repeated adjustments, we identified the optimal range of motion to achieve effective training results. Although increasing the degrees of freedom of hand joints allows for more diverse hand shapes and covers frets that were previously inaccessible, it also enlarges the action space, increasing the complexity of training. Therefore, finding a proper balance between joint freedom and action space is crucial.

The current model is trained from a single song and demonstrates satisfactory performance with pieces possessing a similar tempo. However, to develop a model with broader applicability across a diverse array of musical compositions, it is imperative to expose the model to a dataset with a more extensive collection of songs. Consequently, future work can construct a comprehensive dataset featuring a wide range of musical genres for the purpose of training and enhancing machine learning models to achieve greater versatility.

Replicating human dexterous multi-fingered hand motions is a challenging task. Our work specifically focuses on the intricate hand movements required for playing guitar, providing key insights into leveraging reinforcement learning to precisely control fingers in a physically aware manner. While our fingering generation and fretting control mechanisms are tailored for guitar playing, the underlying reward framework is generalizable and can be applied to other finger control tasks. Future research can explore and validate the algorithm's adaptability to various scenarios based on our work.

The guitar playing demonstrated in this paper is still in its infancy, primarily simulating the techniques commonly used by beginners, where the left hand is responsible for fretting while the right thumb plucks 6th, 5th, and 4th strings, and the index, middle, and ring fingers pluck 3rd, 2nd, and 1st strings, respectively. However, true guitar-playing skills go far beyond this. In addition to basic fretting movements, the left hand may involve advanced techniques such as barre chord, hammer-ons, pull-offs and slides during performance, while the right hand's playing technique is more flexible and varied, involving techniques like harmonics, tapping, muting, and strumming to produce a richer and more nuanced performance.

The effects presented in this paper only scratch the surface of guitar-playing techniques. To truly simulate the diverse techniques of both hands, setting reward functions alone is far from sufficient. To achieve visual effects closer to human performance, we may need to utilize motion capture technology to accurately capture data on these special playing techniques and train the model using imitation learning methods. In our future exploration, we will focus on combining imitation learning to further refine the guitar playing effects and bring them closer to the level of human performance.

Acknowledgments

We are grateful to the anonymous reviewers for their insightful feedback, which significantly enhanced the quality of our paper. This work was supported by the Shanghai Talent Development Funding of China (Grant No. 2021016).

Conflicts of Interest

All authors declare that there are no conflicts of interest.

References

[ABC*20] ANDRYCHOWICZ O. M., BAKER B., CHOCIEJ M., JOZEFOWICZ R., MCGREW B., PACHOCKI J., PETRON A., PLAPPERT M., POWELL G., RAY A., ET AL.: Learning dexterous in-hand manipulation. *The International Journal of Robotics Research 39*, 1 (2020), 3–20. 1

[ALCS18] ARISTIDOU A., LASENBY J., CHRYSANTHOU Y., SHAMIR A.: Inverse kinematics techniques in computer graphics: A survey. In *Computer graphics forum* (2018), vol. 37, Wiley Online Library, pp. 35–58. 2

[All80] ALLES H.: Music synthesis using real time digital techniques. 436–449. URL: http://ieeexplore.ieee.org/document/1455942/, doi:10.1109/PROC.1980.11673. 2

[BCHF19] BERGAMIN K., CLAVET S., HOLDEN D., FORBES J. R.: Drecon: data-driven responsive control of physics-based characters. *ACM Transactions On Graphics (TOG) 38*, 6 (2019), 1–11. 2

[BFH*18] BRADBURY J., FROSTIG R., HAWKINS P., JOHNSON M. J., LEARY C., MACLAURIN D., NECULA G., PASZKE A., VANDERPLAS J., WANDERMAN-MILNE S., ZHANG Q.: JAX: composable transformations of Python+NumPy programs, 2018. URL: http://github.com/google/jax. 6

[BW16] BRETAN M., WEINBERG G.: A survey of robotic musicianship. *Communications of the ACM 59*, 5 (2016), 100–109. 2

[CEG*20] CLEGG A., ERICKSON Z., GRADY P., TURK G., KEMP C. C., LIU C. K.: Learning to collaborate from simulation for robot-assisted dressing. *IEEE Robotics and Automation Letters 5*, 2 (2020), 2746–2753. 2

[Cho73] CHOWNING J. M.: The synthesis of complex audio spectra by means of frequency modulation. *Journal of the audio engineering society 21*, 7 (1973), 526–534. 2

[CKA*22] CHRISTEN S., KOCABAS M., AKSAN E., HWANGBO J., SONG J., HILLIGES O.: D-grasp: Physically plausible dynamic grasp synthesis for hand-object interactions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (2022), pp. 20577–20586. 1

[CMM*18] CHENTANEZ N., MÜLLER M., MACKLIN M., MAKOVIYCHUK V., JESCHKE S.: Physics-based motion capture imitation with deep reinforcement learning. In *Proceedings of the 11th ACM SIGGRAPH Conference on Motion, Interaction and Games* (2018), pp. 1–10. 2

[CWL23] CHEN S., WU A., LIU C. K.: Synthesizing dexterous nonprehensile pregrasp for ungraspable objects. In *ACM SIGGRAPH 2023 Conference Proceedings* (2023), pp. 1–10. 1

[DA20] DEBUT V., ANTUNES J.: Physical synthesis of six-string guitar plucks using the udwadia-kalaba modal formulation. *The Journal of the Acoustical Society of America 148*, 2 (2020), 575–587. 1

[DVS01] D'SOUZA A., VIJAYAKUMAR S., SCHAAL S.: Learning inverse kinematics. In *Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180)* (2001), vol. 1, IEEE, pp. 298–303. 1, 2

[DZBKM22] DONG H.-W., ZHOU C., BERG-KIRKPATRICK T., MCAULEY J.: Deep performer: Score-to-audio music performance synthesis. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2022), IEEE, pp. 951–955. URL: https://ieeexplore.ieee.org/document/9747217/, doi:10.1109/ICASSP43922.2022.9747217. 2

[ES03] ELKOURA G., SINGH K.: Handrix: animating the human hand. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation* (2003), pp. 110–119. 1, 3, 8

[HAM*23] HANDA A., ALLSHIRE A., MAKOVIYCHUK V., PETRENKO A., SINGH R., LIU J., MAKOVIICHUK D., VAN WYK K., ZHURKEVICH A., SUNDARALINGAM B., ET AL.: Dextreme: Transfer of agile in-hand manipulation from simulation to reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)* (2023), IEEE, pp. 5977–5984. 1

[HCV*21] HASSAN M., CEYLAN D., VILLEGAS R., SAITO J., YANG J., ZHOU Y., BLACK M.: Stochastic scene-aware motion prediction, 2021. URL: http://arxiv.org/abs/2108.08284, arXiv:2108.08284[cs]. 2

[HGW*23] HASSAN M., GUO Y., WANG T., BLACK M., FIDLER S., PENG X. B.: Synthesizing physical character-scene interactions, 2023. URL: http://arxiv.org/abs/2302.00883, arXiv:2302.00883[cs]. 2

[HIH*22] HIRAOKA T., IMAGAWA T., HASHIMOTO T., ONISHI T., TSURUOKA Y.: Dropout q-functions for doubly efficient reinforcement learning, 2022. URL: http://arxiv.org/abs/2110.02034, arXiv:2110.02034[cs]. 2

[HZH*19] HAARNOJA T., ZHOU A., HARTIKAINEN K., TUCKER G., HA S., TAN J., KUMAR V., ZHU H., GUPTA A., ABBEEL P., LEVINE S.: Soft actor-critic algorithms and applications, 2019. URL: http://arxiv.org/abs/1812.05905, arXiv:1812.05905[cs, stat]. 6

[KCN24] KIM H., CHOI S., NAM J.: Expressive acoustic guitar sound synthesis with an instrument-specific input representation and diffusion outpainting. *arXiv preprint arXiv:2401.13498* (2024). 1

[KdlCCP*24] KYRIAKOU T., DE LA CAMPA CRESPO M. , PANAYIOTOU A., CHRYSANTHOU Y., CHARALAMBOUS P., ARISTIDOU A.: Virtual instrument performances (vip): A comprehensive review. *Computer Graphics Forum 43*, 2 (2024), e15065. 3

[KK19] KODAMA K., KOUTAKI G.: Development of guitar playing robot by pwm control of solenoid. In *2019 IEEE 8th Global Conference on Consumer Electronics (GCCE)* (2019), IEEE, pp. 291–293. 3

[KLXvdP21] KIM N. H., LING H. Y., XIE Z., VAN DE PANNE M.: Flexible motion optimization with modulated assistive forces. *Proceedings of the ACM on Computer Graphics and Interactive Techniques 4*, 3 (2021), 1–25. 2

[KNK*22] KAWAMURA M., NAKAMURA T., KITAMURA D., SARUWATARI H., TAKAHASHI Y., KONDO K.: Differentiable digital signal processing mixture model for synthesis parameter extraction from mixture of harmonic sounds. In *ICASSP 2022 - 2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (2022), IEEE, pp. 941–945. URL: https://ieeexplore.ieee.org/document/9746399/, doi:10.1109/ICASSP43922.2022.9746399. 2

[LC13] LI Y.-F., CHUANG L.-L.: Controller design for music playing robot—applied to the anthropomorphic piano robot. In *2013 IEEE 10th International Conference on Power Electronics and Drive Systems (PEDS)* (2013), IEEE, pp. 968–973. 3

[LFL*23] LAKSHMIPATHY A. S., FENG N., LEE Y. X., MAHLER M., POLLARD N.: Contact edit: Artist tools for intuitive modeling of hand-object interactions. *ACM Transactions on Graphics (TOG) 42*, 4 (2023), 1–20. 1

[LXJ*23] LIU M., XU C., JIN H., CHEN L., T M. V., XU Z., SU H.: One-2-3-45: Any single image to 3d mesh in 45 seconds without per-shape optimization, 2023. URL: http://arxiv.org/abs/2306.16928, arXiv:2306.16928[cs]. 2

[mus24] Musicxml. https://www.musicxml.com/, 2024. Accessed: April 27, 2024. 11

[MWTK13] MORDATCH I., WANG J. M., TODOROV E., KOLTUN V.: Animating human lower limbs using contact-invariant optimization. *ACM Transactions on Graphics (TOG) 32*, 6 (2013), 1–8. 2

[PGH*22] PENG X. B., GUO Y., HALPER L., LEVINE S., FIDLER S.: ASE: Large-scale reusable adversarial skill embeddings for physically simulated characters. 1–17. URL: http://arxiv.org/abs/2205.01906, arXiv:2205.01906[cs], doi:10.1145/3528223.3530110. 2

[Pha06] PHAM H.: Pyaudio documentation. https://people.csail.mit.edu/hubert/pyaudio/, 2006. 6

[PVDP17] PENG X. B., VAN DE PANNE M.: Learning locomotion skills using DeepRL: does the choice of action space matter? In *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation* (2017), ACM, pp. 1–13. URL: https://dl.acm.org/doi/10.1145/3099564.3099567, doi:10.1145/3099564.3099567. 2

[QKC*23] QI H., KUMAR A., CALANDRA R., MA Y., MALIK J.: In-hand object rotation via rapid motor adaptation. In *Conference on Robot Learning* (2023), PMLR, pp. 1722–1732. 1

[RBL*21] ROMBACH R., BLATTMANN A., LORENZ D., ESSER P., OMMER B.: High-resolution image synthesis with latent diffusion models, 2021. arXiv:2112.10752. 2

[RHG*21] RAFFIN A., HILL A., GLEAVE A., KANERVISTO A., ERNESTUS M., DORMANN N.: Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research 22*, 268 (2021), 1–8. URL: http://jmlr.org/papers/v22/20-1364.html. 6

[Str83] STRONG K. A.: Digital synthesis of plucked-string and drum timbres. *Computer Music Journal 7*, 2 (1983), 43–55. 2

[SY16] SUYUN F., YIBIAO Y.: Improve music synthesis quality by particular harmonic spectrum interpolation based on sinusoidal model. In *2016 IEEE 13th International Conference on Signal Processing (ICSP)* (2016), IEEE, pp. 183–186. URL: http://ieeexplore.ieee.org/document/7877820/, doi:10.1109/ICSP.2016.7877820. 2

[TET22] TODOROV E., EREZ T., TASSA Y.: Mujoco documentation. https://mujoco.readthedocs.io/en/stable/overview.html#mesh, 2022. 6, 7

[TTK*23] TOWERS M., TERRY J. K., KWIATKOWSKI A., BALIS J. U., COLA G. D., DELEU T., GOULÃO M., KALLINTERIS A., KG A., KRIMMEL M., PEREZ-VICENTE R., PIERRÉ A., SCHULHOFF S., TAI J. J., SHEN A. T. J., YOUNIS O. G.: Gymnasium, Mar. 2023. URL: https://zenodo.org/record/8127025, doi:10.5281/zenodo.8127026. 6

[TWGvdP22] TAO T., WILSON M., GOU R., VAN DE PANNE M.: Learning to get up. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings* (2022), ACM, pp. 1–10. URL: https://dl.acm.org/doi/10.1145/3528233.3530697, doi:10.1145/3528233.3530697. 2

[VK20] VAHDAT A., KAUTZ J.: Nvae: A deep hierarchical variational autoencoder. *Advances in neural information processing systems 33* (2020), 19667–19679. 2

[WGH22] WON J., GOPINATH D., HODGINS J.: Physics-based character controllers using conditional VAEs. 1–12. URL: https://dl.acm.org/doi/10.1145/3528223.3530067, doi:10.1145/3528223.3530067. 2

[WGL22] WU A., GUO M., LIU C. K.: Learning diverse and physically feasible dexterous grasps with generative model and bilevel optimization. *arXiv preprint arXiv:2207.00195* (2022). 1

[WK23] WIGGINS A., KIM Y.: A differentiable acoustic guitar model for string-specific polyphonic synthesis. In *2023 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics (WASPAA)* (2023), IEEE, pp. 1–5. 1

[WPP14] WAMPLER K., POPOVIĆ Z., POPOVIĆ J.: Generalizing locomotion style to new animals with inverse optimal regression. *ACM Transactions on Graphics (TOG) 33*, 4 (2014), 1–11. 2

[WWY23] WENG Z., WANG Z., YEUNG S.: ZeroAvatar: Zero-shot 3d avatar generation from a single image, 2023. URL: http://arxiv.org/abs/2305.16411, arXiv:2305.16411[cs]. 2

[XLKvdP20] XIE Z., LING H. Y., KIM N. H., VAN DE PANNE M.: Allsteps: curriculum-driven learning of stepping stone skills. In *Computer Graphics Forum* (2020), vol. 39, Wiley Online Library, pp. 213–224. 2

[XLW*22] XU H., LUO Y., WANG S., DARRELL T., CALANDRA R.: Towards learning to play piano with dexterous hands and touch. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2022), IEEE, pp. 10410–10416. 3

[XSLVDP22] XIE Z., STARKE S., LING H. Y., VAN DE PANNE M.: Learning soccer juggling skills with layer-wise mixture-of-experts. In *Special Interest Group on Computer Graphics and Interactive Techniques Conference Proceedings* (2022), ACM, pp. 1–9. URL: https://dl.acm.org/doi/10.1145/3528233.3530735, doi:10.1145/3528233.3530735. 2

[Yeo21] YEON: Playing piano with a robotic hand | mit 6.843 final project. https://www.youtube.com/watch?v=WzJJ4c6AqnE, 2021. Accessed: December 9, 2021. 3

[YYL22] YANG Z., YIN K., LIU L.: Learning to use chopsticks in diverse gripping styles. 1–17. URL: https://dl.acm.org/doi/10.1145/3528223.3530057, doi:10.1145/3528223.3530057. 2

[ZMM11] ZHANG A., MALHOTRA M., MATSUOKA Y.: Musical piano performance by the act hand. In *2011 IEEE international conference on robotics and automation* (2011), IEEE, pp. 3536–3541. 3

[ZSG*23] ZAKKA K., SMITH L., GILEADI N., HOWELL T., PENG X. B., SINGH S., TASSA Y., FLORENCE P., ZENG A., ABBEEL P.: RoboPianist: A benchmark for high-dimensional robot control, 2023. URL: http://arxiv.org/abs/2304.04150, arXiv:2304.04150[cs]. 1

[ZWS*23] ZAKKA K., WU P., SMITH L., GILEADI N., HOWELL T., PENG X. B., SINGH S., TASSA Y., FLORENCE P., ZENG A., ABBEEL P.: RoboPianist: Dexterous piano playing with deep reinforcement learning, 2023. URL: http://arxiv.org/abs/2304.04150, arXiv:2304.04150[cs]. 3, 4

[ZYM*23] ZHANG H., YUAN Y., MAKOVIYCHUK V., GUO Y., FIDLER S., PENG X. B., FATAHALIAN K.: Learning physically simulated tennis skills from broadcast videos. 2

Appendix A: Guitar-Playing Terminology

Sound Production Mechanism. The guitar produces sound through the vibration of its strings. The left hand presses down on the strings along the fretboard, where the pressed position is called a fret, changing the pitch by shortening the vibrating length of the string. The right hand plucks or strums the strings to initiate the vibrations. Pitch refers to the perceived frequency of a sound, which determines whether it is high or low in tone.

Monophonic Pieces involve playing one note at a time (the left hand only needs to press one fret). This technique is often used for melodies or solo lines.

Polyphonic Pieces involve playing multiple notes simultaneously (the left hand needs to press multiple frets at the same time). This creates harmony and is a fundamental aspect of rhythm guitar playing.

In monophonic pieces, typically only one finger of the left hand is responsible for pressing a specific fret at any given moment. In contrast, in polyphonic pieces, most of the time, multiple fingers of the left hand press different frets simultaneously, making it more challenging than monophonic pieces.

Appendix B: Left Hand Joint Limit

Table 4 shows the joint limits (action space) of the left hand in our system. WRJ, THJ, FFJ, MFJ, LFJ, and forearm represent the joints of the wrist, thumb, index finger, middle finger, ring finger, little finger, and forearm, respectively. The numbers 1, 2, 3, and 4 are IDs for the joints of each finger. We constrain WRJ2 and all joints of the thumb to be immobile. FFJ0, MFJ0, and LFJ0 are tendon elements in *MuJoCo*, simultaneously controlling the two joints 1 and 2 at the tips of each finger. Additionally, we constrain the translation of the forearm as $\text{forearm}_x \in [-0.23, 0.09]$, $\text{forearm}_y \in [-0.01, 0.10]$, and $\text{forearm}_z \in [-0.1, 0.02]$.

Table 4: *Joint Limits / Action Space*

Joint Name	Joint Range Limits	Actual Range Limits
lh_WRJ2	[0.00,0.00]	[-0.52,0.17]
lh_WRJ1	[0.00,0.79]	[-0.70,0.79]
lh_THJ4	[0.00,0.00]	[0.00,1.22]
lh_THJ3	[0.00,0.00]	[-0.21,0.21]
lh_THJ2	[0.00,0.00]	[-0.70,0.70]
lh_FFJ4	[-0.35,0.35]	[-0.35,0.35]
lh_FFJ3	[0.00,1.57]	[-0.26,1.57]
lh_FFJ0	[0.00,2.00]	[0.00,3.14]
lh_MFJ4	[-0.35,0.35]	[-0.35,0.35]
lh_MFJ3	[0.00,1.57]	[-0.26,1.57]
lh_MFJ0	[0.00,2.00]	[0.00,3.14]
lh_RFJ4	[-0.35,0.35]	[-0.35,0.35]
lh_RFJ3	[0.00,1.57]	[-0.26,1.57]
lh_RFJ0	[0.00,2.00]	[0.00,3.14]
lh_LFJ4	[-0.35,0.35]	[-0.35,0.35]
lh_LFJ3	[0.00,1.57]	[-0.26,1.57]
lh_LFJ0	[0.00,2.00]	[0.00,3.14]

© 2024 Eurographics - The European Association for Computer Graphics and John Wiley & Sons Ltd.