Abstract
Gesture Prediction Model for the Guitar Fingering Problem
by Arman Samavatian
December, 2014
Director: Dr. Nasseh Tabrizi
DEPARTMENT OF Computer Science

In this thesis we provide a method for finding the fingering of a music piece on any type of guitar using a hand model. Adapting to the real world conditions by deploying a model of the user's hand, and considering the constraints of the guitar and the music notes is what makes our method more realistic. We have modeled the movements of the user's hand in such a way that the thumb does not play any role, and the movements of the other four fingers are modeled using a set of kinematics equations. We use two sets of constraints derived from the guitar and the music notes in order to include the playing techniques, which are required by the music piece and the guitar. The guitar is considered to be a separate entity in our model having its own properties, resulting in a method independent of the type and tuning of the instrument. Since we are using the hand model for generating the fingering of the music piece, the results of the method are gestures generated for the notes, and the final outcome will be an animation for the entire sheet of music.

Gesture Prediction Model for the Guitar Fingering Problem

A Dissertation

Presented To

The Faculty of the Department of Computer Science

East Carolina University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science in Software Engineering

by

Arman Samavatian

December, 2014

© Copyright 2014
Arman Samavatian

Gesture Prediction Model for the Guitar Fingering Problem

by

Arman Samavatian

APPROVED BY:

DIRECTOR OF DISSERTATION:_____
M. H. N. Tabrizi, PhD

COMMITTEE MEMBER:_____
Junhua Ding, PhD

COMMITTEE MEMBER:_____
Sergiy Vilkomir, PhD

CHAIR OF THE DEPARTMENT OF COMPUTER SCIENCE:

_____
Karl Abrahamson, PhD

DEAN OF THE GRADUATE SCHOOL:

_____
Paul J. Gemperline, PhD

Table of Contents

List of Tables

List of Figures

CHAPTER 1: INTRODUCTION

As computers become a larger part of our lives, the need for better and easier communication with them becomes more sensible. Human Computer Interaction (HCI) is the field which studies how people design, implement, and interact with computer systems, and how the interaction is developed between those systems and their users [1]. The goal is to have efficient and easy to use User Interfaces (UIs), which provide adequate support for the defined tasks, have a more powerful form of communication, and better access to information [2]. Many applications exist today with the goal of improving the user experience in a particular area and tutoring applications are a good example of them. This study proposes a method with the ultimate goal of being used in a music tutoring application. More specifically, the target application can be used to teach how to play the guitar, assuming the teacher's hand is similar to that of the user. To achieve this, the application needs to play the music piece in a way that the user can play it. We will refer to "the way of playing the music piece" as fingering.

In this thesis, we present a new method for finding the fingering of a music piece for the guitar and other instruments in the guitar family. Our method uses two models to find the fingering of the music sheet, namely i. a constraint based model (a similar approach is reported in [3]); and ii. a hand model for identifying appropriate gestures.

A particular note can be played on up to 6 positions on the guitar neck (for the guitars with 6 strings). So, for *n* notes we can have up to 6^n playable positions. For any of the playable positions, it is possible to use up to 4 fingers; so the number grows to a maximum of 24^n possible fingered positions for *n* notes. This is the essential reason that turns fingering into a challenging problem that needs to be solved.

As it is often the case, multiple acceptable fingering scenarios may exist for the same music sheet. If that is the case, we consider the fingering which is closest to what the user would prefer (if no user preferences are reported, any of the acceptable scenarios can be considered).

The fingering for any music note on the guitar can be specified using a triplet consisting of three variables <*string, fret, finger*> [4]. A combination of <*string, fret*> represents a unique position on the guitar fretboard and, by assigning a *finger* to the position, a gesture is produced. The triplet is all we need to know for each note to play the entire music piece. The goal in identifying the fingering is to ultimately have only one triplet for every note of the music score.

There are three entities involved when playing the guitar, namely the hand, the music notes, and the instrument itself. These entities enforce numerous constraints to the fingering. Identifying these constraints and applying them in an orderly manner, will lead us to the successful outcome.

One of the motivations of this study was to develop a model, which can adapt to human hands when playing a musical instrument. Guitarists focus on the movements of their hands as they play the notes; they do not refer to an external source for identifying the possible movements/positions of their fingers [4, 5, 3, 6, 7]. The Object Oriented Design (OOD) approach can help to design an entity with the specific task of thinking about the movements of the hand, and can helps us to not need to statically define the possible movements of the fingers. Examples of static movement definitions in some of the most recent studies include: defining comfortable spans [4], defining impossible finger states [3], and using cost functions [5].

In our method, we have used a hand model to be the decision maker for the movements of the fingers, instead of the traditionally used cost functions. The hand model generates appropriate gestures for the possible combinations of triplets, and chooses the most relaxed ones

to be parts of the fingering. This requires us to create a model of the human hand with adequate accuracy. The hand consists of a skeleton, muscles and tendons, and a layer of skin; each enforcing multiple constraints to the hand movements. The skeleton in our model is a mesh of hierarchical joints with 19 Degrees of Freedom (DOF); one for the Distal Interphalangeal (DIP) joint, one for Proximal Interphalangeal (PIP) joint, 2 for Metacarpophalangeal (MCP) joint, and three for the wrist. Figure 1 shows the finger joints. The constraints of the skeleton are derived from the freedoms of the joints. Since the position of the hand is also important to us, we add the position (x, y, and z) to the number of DOFs resulting in 22 in total. We are not considering the thumb in our model as its movements are independent of the other four fingers, especially since it will be always placed behind the neck of the guitar.

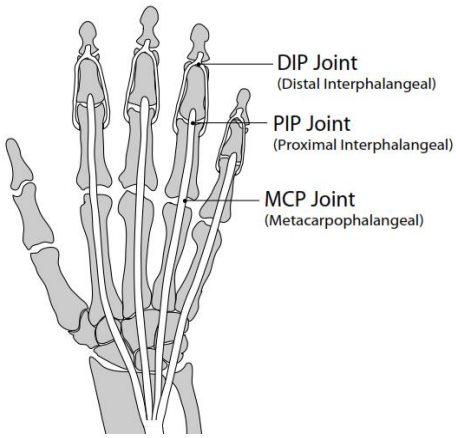

Figure 1 - Finger Joints (Adapted from **[5]**)

According to numerous hand model studies [8, 2, 9, 10], the DIP joint can be approximated from the PIP joint using Eq. 1

$$\Theta_{DIP}=2/3\Theta_{PIP} \quad (1)$$

This helps to reduce the total DOF from 22 to 18. We have used this approximation to reduce the amount of time, which takes to identify a gesture. Our hand model only contains the

bones and the joint hierarchy. Due to the limited time we had for this study, we haven't included muscles, tendons, and the skin. In developing our hand model, we have assumed the following:

- The movements of the fingers are not influenced by each other's
- Motion frequencies and angular velocities are not influential on the finger movements
- The thumb is not present in the model, since its movements are not of interest

1.1 Thesis Contributions

The contributions of this thesis are as follows: Development of a new model for the guitar fingering using the hand model, in which we have used no statically defined cost model or finger movements. Using our method, the system is able to find gestures for any hand, and on any guitar (bass, guitar, banjo, ukulele, etc.) with any particular tuning. The hand model can produce gestures close to the actual user's hand which can follow the path of the user's fingers, with the ability to account for possible disabilities or personal playing styles[1].

Inclusion of new guitar playing techniques such as sliding, bending, and limited two handed tapping in the fingering.

1.2 Thesis Overview

The remainder of this document is organized as follows: in the next chapter we have reviewed previous literature work, Chapter 3 talks about the constraints of the music notes and the instrument, Chapter 4 describes the hand model in detail, Chapter 5 will focus on the design and implementation and will show the results, and Chapter 6 will propose the possible future extensions to improve this study and discusses the limitations. Finally, Appendix A gives some insight about the guitar and MIDI files (the music notes input).

[1] A playing style is defined by choosing the probabilities of finger usages.

CHAPTER 2: RELATED WORK

This chapter reviews the previous literature work in the following two sections:

- Literature is reviewed for the musical performance problem and the limitations and valuable ideas behind those methods are discussed.
- A brief survey on the studies of human hand modeling is presented.

2.1 Music Performance

Fingering is the act of mapping a music note to a position on an instrument, such as the guitar and associating a finger to it. There are many research articles discussing this problem from different points of views. The study reported in [6] presents one of the oldest methods for the fingering of string instruments using an approach called the optimum path paradigm, which was updated later [7] using a learning algorithm called path difference learning. In this approach, they have used expert opinion to adjust the movement cost factors in order to find solutions closer to the desired path (the path that is chosen by the expert). To achieve this they have applied stochastic models such as simulated annealing or genetic search [11, 12] in the cost function weight space.

Radicioni et al. [4] presents an algorithm with a constant, empirically derived cost function to find the optimum path in the graph of notes using Dijkstra's shortest path algorithm. In another article [3] the authors have used the constraint satisfaction problem technique to find the fingering of the chords, in order to complete their previous study. We have implemented these two methods as a part of studying the previous work.

One of the most popular methods for the guitar fingering problem is reported in [5], where the authors have used a statically defined cost function for different finger movements on

the guitar fretboard, and an Inverse Kinematics hand model only for the purpose of showing the playing as animation. Using a number of variables, they have accounted for the preferences of the user in using their fingers, and can include user's disabilities. Static user-defined frames are used as the choice of how long the algorithm reads into the music notes.

Although others have considered static cost functions and finger movements, we are using a purely dynamic model. In this model, the hand itself decides on the movements among the possible options its fingers have and then selects the best and most relaxed gestures to play the music piece.

Generally speaking, the process of finding a fingering for a music piece is an optimization problem and the cost model heavily depends on the target instrument [13]. To find the optimum fingering, scientists have used dynamic programming [14, 4, 6], hidden Markov model [15], neural networks [16, 17], and genetic algorithm [18] for generating guitar tablature from music notes. The main drawbacks of these methods are as follows. First, most of them either solve the fingering of the entire score or use statically defined note frames. Second, defining static and hardcoded cost models is arguably an unnatural decision, especially since the lengths of the frets become shorter as they become closer to the body of the guitar; this can be handled by providing a better decision maker, such as a model of the human hand. Finally, some of the methods do not consider the timings (durations) of the notes and consider equal timings for them; a decision which will change the input significantly.

2.2 Modeling the Human Hand

Modeling the hand has become a significant part of several research topics as the importance of mimicking the correct motions of the hand increases. Researchers have used multiple methods to model the human hand to use in their studies. In the field of Computer

Graphics and for character animation, one of the most frequently used techniques is motion capture, using cameras, sensors, image processing, or data gloves. A good survey of the topic is available in [19]. However, the generated motions are often dedicated for a specific usage, and they are hard and expensive to regenerate [13]. By constructing the underlying Markov model, some researchers have tried to reuse the motion capture data [20, 21].

Another useful method to produce hand gestures is Inverse Kinematics (IK) for which there is a good instruction in [22]. Originally taken from the field of robotics, IK is extensively used in many researches in computer graphics [23, 24, 5]. Including more details of the hand (muscles, tendons, deformations of the palm, the interdependence of the joints, etc.) would result in a more natural hand model, and such a model would generate better gestures due to its ability to produce more complex effects [25]. But, more complexity means less speed. There have been some efforts to improve the efficiency of IK solutions from various viewpoints. A linear programming approach is introduced in [26] to solve IK problems. This method helped the IK problem to grow linearly with respect to the number of DOF. Other approaches focused on solving the IK problem for special cases. For example, the authors in [27] present a method to solve IK for skeletal manipulation by switching between three IK algorithms based on the situation of the skeleton. Despite the fact that IK is an extensively used technique, applying it to get the desired animation can be time consuming, and the animator needs to carefully set the constraints of the rigid body [2].

Modeling the dynamics of the hand is another method, which includes adding layers of muscles and tendons to the skeleton of the hand. It is a method widely used in robotics, biomechanics, and computer animation [28, 29, 30]. The motions of the hand in this approach are calculated by a set of differential equations over a discrete time period. A model of the

human hand and forearm using forward dynamics simulation to produce biomechanically accurate gestures is proposed in [31]. Some researchers have attempted to model the dynamics by using prediction methods. Particularly, they try to produce continuous hand gestures using the captured hand motions [32]. Some of these methods include extended Kalman Filtering, Finite State Machines, Hidden Markov Models, Bayesian Networks, and Neural Networks [2, 32].

Additionally, researchers have modeled the hand using other techniques such as geometrical modeling and statistical modeling. Geometrical modeling is performed by using the geometry of the surface of the hand [2]. Wu and Huang [32] present a method using splines, geometric shapes, and free-form models to build a geometric surface to model the hand. In statistical modeling, the hand movements are learned from a set of previously existing gestures, such as images or motion capture data [33].

In this thesis we have used a Forward Kinematics hand model (searching the domain of the joint angles in order to find appropriate gestures). The rationale for this decision is presented in Chapter 4.

CHAPTER 3: MODELING THE CONSTRAINTS OF THE GUITAR AND MUSIC NOTES

Our method starts by first modeling the constraints enforced to the playing by the guitar and the music notes without considering the hand object. This step can be thought of as a preprocessing stage with mid-level results, which will be fed into the next step (the hand model, Chapter 4). The expectation from this step should not be high, since the set of constraints is rather small and thus, cannot eliminate many possibilities. However, it can help to reduce the size of the input for the main processing step. The constraints are explained in detail in section 3.2.

Before discussing the constraints, we need to first explain how the music notes are structured in the memory once they are read from the input file. The following section discusses the creation of the graph of notes.

3.1 Constructing the Graph of Notes

As described in [4], the notes are first organized in a layered graph. For each music note from the input, there is a group of vertices in the graph representing every possible fingering triplet for the note. Every triplet of a note is connected to every triplet of the next note of the input file. Figure 2 shows a graph of three notes (F2, D4, E3).

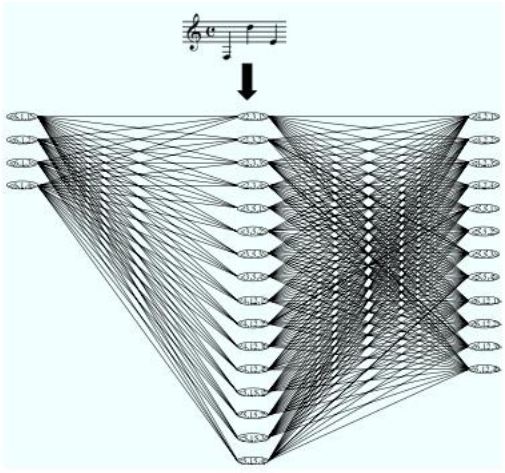

Figure 2 - Graph of three notes (Adapted from [4]).

Each edge will be weighted by the cost of movement from one triplet to the next. The costs of movements are generated by the hand model, based on a number of factors described in Chapter 4. The final solution will be the shortest path in the graph. Since the frames of notes can be played separately, the graph is created and solved separately for every frame of notes (the creation of the frames is discussed later in this chapter). For the chords, since we have more than one note, their vertices will include more than one triplet; in a way that the set of triplets satisfies the constraints of the chord (see section 3.2.1).

3.2 Constraint Types

3.2.1 Simultaneous Notes

These are the notes that share their playing time, either entirely (chords) or partially (a note starts playing before the previous one stops). It is obvious that no two simultaneous notes can be played on the same string of the guitar; so, in order to play the group of the simultaneous notes, we need to use as many strings in the fingering as the number of simultaneous notes. Without considering the hand, all we need to know is that we have to use at least two strings. However, with the use of the hand model, more constraints will be included later; for example, higher frets should be played with higher fingers (e.g., we cannot have <1,2,3> and <2,3,2> for two <*string, fret, finger*> triplets). This constraint (as we will see) is not specifically stated in the hand model, but the model behaves in such a way that it includes this constraint (collision detection).

3.2.2 Bends and Slides

These are the notes that include pitch changes while they are alive (not stopped). There are three possible pitch changes in a MIDI input: bends, vibratos, and slides. Bending is performed by stretching a string upward or down, and will result in a continuous pitch change. Vibrato refers to the act of bending the string up and down slightly and very fast (vibrations). Finally, slides are performed by moving a finger from one fret to another on the same string of the guitar without releasing the string, and will result in discrete pitch changes (sliding through frets). This constraint will force us to choose one finger for the action and only one string.

3.2.3 Timings of the Notes

Based on the timings of the notes, we can put them into separately playable frames. If we see a group of fast-playing notes, we can be sure that for playing these notes, the hand should maintain its position as much as possible, thus, the locality of the solution must be as high as possible. This will ultimately reflect in the fingering by having close frets on (possibly) close strings.

3.2.4 Tapping Technique

In the tapping technique, the right hand will also be involved in the act of fretting, and the notes are played by tapping a string against the fretboard and holding down on a fret (hammer-on) and pulling off the string with an already holding finger (pull-off). This can be done with both hands, as opposed to the right hand usually picking the strings and the left hand only holding down the strings. The process of identifying this technique from the music notes is fairly complex, but with some assumptions we can make it easier. Tapping usually happens when the

notes have short timings, and they can be divided into two groups playable (with high locality) on different positions on the fretboard.

3.3 Applying the Constraints

We have implemented these constraints to be applied to the graph of notes as a preprocessing step. The set of constraints are applied to every transition of the graph, and will filter non-desirable edges. Considering all the above, we can conclude that the results of this step would only have to be triplets of *<string, fret, don't_care>*, meaning no finger information can be included, otherwise we are enforcing the hand to choose between positions that are generated with invalid assumptions (any assumption we make about fingers without using the hand model is invalid). In another words, if we assume constant distances for the finger pairs, we will limit the input for the hand model.

Note that the solution resulted from this step is not near complete, since it will not include any *finger* information. All we can expect from this step are guidelines towards which transitions to or not to choose.

3.4 Verification of the Implementation

After executing this step, the graph of notes should include some extra information derived from each constraint; such as which notes are supposed to be (or not to be) played on the same string, which set of notes are considered to require tapping, and the frame definitions for the notes.

In order to test the preprocessing stage, we have composed some music notes (special cases) which contain the above mentioned constraints. Table 1 shows the test cases.

Type of Constraint	*Identified*	*Passed*	*Comments*
Sliding one note	YES	YES	
Sliding two notes simultaneously	YES	YES	
Bending and Sliding simultaneously	NO	YES	MIDI processing issue
Tapping (notes with short timings)	YES	YES	
Tapping (notes with long timings)	NO	NO	Considered normal notes
Chord (F#, Gm, G7, A, Am)	YES	YES	
Putting the notes into frames	N/A	YES	

Table 1 – Tested scenarios and their results

Two of the test cases had problems. The first one was a very odd situation, where a note is supposed to be bent, while two other are sliding. Though it is an extreme situation, and the bend and slides are not identified correctly, the test case is marked as passed, because the requirement was to identify the simultaneous notes and mark the suitable extra information (regarding the pitch bend) for them. The error lies in the implementation of the MIDI file processor. Pitch bends in MIDI files are all shown with a short message (code: 224) following other information which can help to realize whether it is a slide or a bend; the problem is that for simultaneous notes, there is no way to identify which pitch-bend message corresponds to which note, since the note-on messages (code: 144) happen at the same time. After those the pitch-bend messages are listed for them.

The failed test case was related to tapping, for which we had the assumption that all the tapping notes will have short timings; but for this test case, we tested a tapping situation with tapping notes having longer durations then the other non-tapping notes. This is due to the assumption we made, and to the best of my knowledge, there is no specific way to identify the tapping notes directly from the MIDI files.

In conclusion, this chapter described the set of constraints extracted from the instrument and the music notes. The goal of this preprocessing step is to reduce the size of the input for the next step. So, for future extensions we can imagine more constraints in order to eliminate even more undesired transitions (moving from a triplet to the next one). Implementing a better and stronger tapping-recognizing algorithm is a very good example of these constraints. For other techniques, we can mention artificial harmonies, swipes, and recognizing the necessary barrét positions (where a finger is used for holding more than one string, on a particular fret).

CHAPTER 4: GENERATING THE GESTURES USING THE HAND MODEL

The hand model is the entity with three specific tasks. i) It should be able to create gestures close to the user's hand. So, we need to develop a mathematical model, which can map the angles of the finger joints and the wrist, to positions of the fingertips in the 3D Cartesian space. This process is known as Forward Kinematics. ii) It should be able to do the reverse; mapping fingertip positions in the Cartesian space, to finger joint and wrist angles. Usually, this process is done with the use of Inverse Kinematics models; but in this thesis, we have used Forward Kinematics, which is described in section *4.2.2* in more detail. iii) The hand model should be able to choose appropriate gestures for different situations. Since the reverse process (second task in this paragraph) does not have unique answers, we need to include a decision-making mechanism for the hand entity, which is described in section *4.3* in details.

The reasons for choosing kinematics equations for the hand model are listed here. Since the ultimate goal of this study was to use the method in a guitar tutoring application, we have modeled the hand using kinematics equations. Kinematics formulas help us to define a mathematical hand model, which is ready to perform its tasks as soon as the lengths of the user's fingers are measured and inserted into the model. Kinematics equations are also easier to understand and use, compared to modeling the dynamics of the hand; and, they require no peripheral devices as motion capture techniques do. Finally, due to the time constraints of this study, and their simplicity, kinematics equations were the best option.

For the second task stated earlier in the beginning of the chapter, we are using a Forward Kinematics solution. Precisely speaking, we search the domain of joint angles and reposition the hand, until the fingertip reaches to an *epsilon* distance from the target end-point. We made this

decision due to two main concerns: First, due to the time constraints of this study, and because of IK's high complexity, we couldn't implement a suitable Inverse Kinematics solution. Second, since the hand model has the task of choosing the best gestures for the action, we need to have multiple gestures covering each combination of notes in the decision making process. This can be done easier using Forward Kinematics instead of Inverse.

4.1 The Hand Model

Each finger in the hand model has three joints: Metacarpophalangeal (MCP) with two degrees of freedom (DOF), Proximal Interphalangeal (PIP) with one DOF, and Distal Interphalangeal (DIP) having one DOF. The axes of the 3D space are defined as shown in Figure 3. Note that Θ_{MCP}, Θ_{PIP}, and Θ_{DIP} are for extension/flexion, and Θ_{MCP2} is for adduction/abduction (side to side movements) of the fingers (see Figure 4).

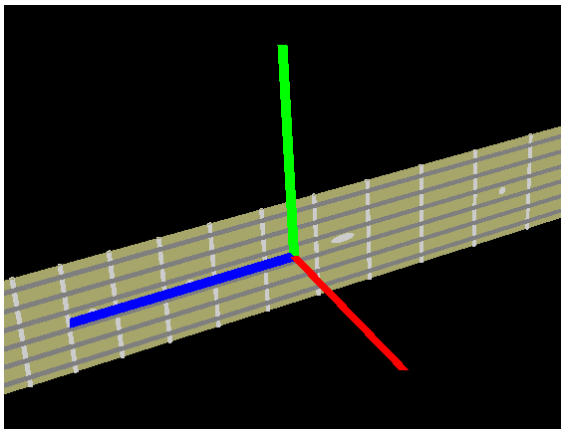

Figure 3 - The axes (X, Y, Z) are shown in red, blue, and green respectively

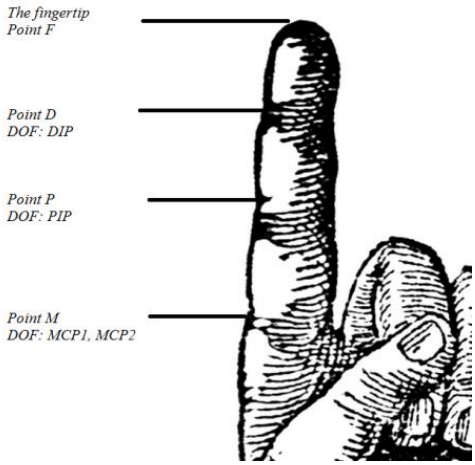

Figure 4 - The four defined points on a finger

4.1.1 Mapping the Joint Angles to Fingertip Positions

To calculate the positions of the points defined on the fingers (shown in Figure 4), we use *Eq. 2-14* (*Base* is a point in space, representing the position of the hand):

$$l = Length*cos(\Theta_{MCP2}) \qquad (2)$$

Where *Length* is the measured value of the length of the user's finger, and *l* is its adjustment after adduction/abduction of the finger, to be used in calculations of X and Z.

$$X(M) = X(Base) \quad (3)$$

$$X(P) = X(M) + \sin(\Theta_{MCP}) * l_{MtoP} \quad (4)$$

$$X(D) = X(P) + \sin(\Theta_{MCP} + \Theta_{PIP}) * l_{PtoD} \quad (5)$$

$$X(F) = X(D) + \sin(\Theta_{MCP} + \Theta_{PIP} + \Theta_{DIP}) * l_{DtoF} \quad (6)$$

Where M, P, D, and F correspond to *MCP* joint, *PIP* joint, *DIP* joint, and the *Fingertip*.

$$Y(M) = Y(Base) \quad (7)$$

$$Y(P) = Y(M) + \sin(\Theta_{MCP2}) * Length_{MtoP} \quad (8)$$

$$Y(D) = Y(P) + \sin(\Theta_{MCP2}) * Length_{PtoD} \quad (9)$$

$$Y(F) = Y(D) + \sin(\Theta_{MCP2}) * Length_{DtoF} \quad (10)$$

Note that for calculating the Y coordinates, we are using *Length* instead of *l*.

$$Z(M) = Z(Base) \quad (11)$$

$$Z(P) = Z(M) + \cos(\Theta_{MCP}) * l_{MtoP} \quad (12)$$

$$Z(D) = Z(P) + \cos(\Theta_{MCP} + \Theta_{PIP}) * l_{PtoD} \quad (13)$$

$$Z(F) = Z(D) + \cos(\Theta_{MCP} + \Theta_{PIP} + \Theta_{DIP}) * l_{DtoF} \quad (14)$$

And Eq.11-14 are used to calculate Z coordinates.

Based on the values of all the joint angles, we will be able to calculate the coordinates of each joint on the finger. The next step will be applying the rotations of the wrist to these coordinates.

The wrist has three DOF and, for each one, we need to apply a rotation matrix to the calculated X, Y, and Z. The rotation matrix is presented in Eq. 15.

$$R_x R_y R_z = \begin{bmatrix} \cos\beta\cos\gamma & -\cos\beta\sin\gamma & \sin\beta \\ \cos\alpha\sin\gamma + \sin\alpha\sin\beta\cos\gamma & \cos\alpha\cos\gamma - \sin\alpha\sin\beta\sin\gamma & -\sin\alpha\cos\beta \\ \sin\alpha\sin - \cos\alpha\sin\beta\cos\gamma & \sin\alpha\cos + \cos\alpha\sin\beta\sin\gamma & \cos\alpha\cos\beta \end{bmatrix} \quad (15)$$

Where α, β, and γ are the rotation around x, y, and z axis respectively. The rotation matrix (Eq. 15) will be multiplied to the previous coordinates (calculated by Eq. 2 - 14) as a matrix to get the rotated coordinates.

To test this step, we have implemented an application using Java and OpenGL, and measured real user hand data to be plugged into the model. No OpenGL built-in methods have been used for rotations of the finger joints or the wrist angles, and only the calculated coordinates for *X*, *Y*, and *Z* were taken into account. Figure 5 shows examples of what the hand looks like (the palm is not drawn, only the four fingers are being shown).

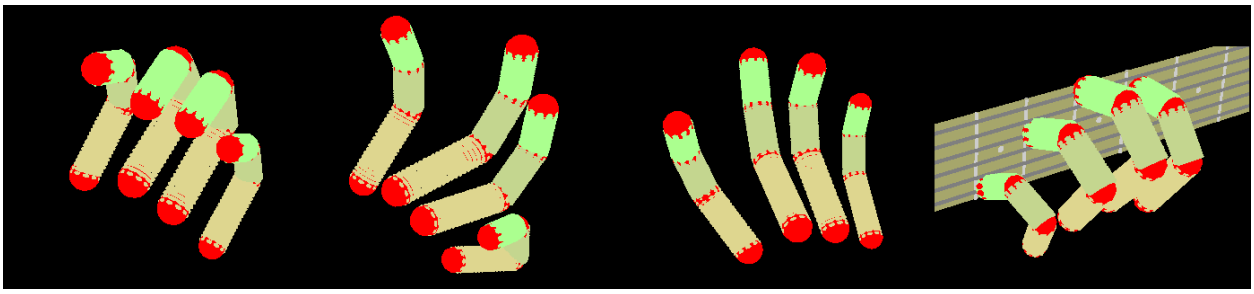

Figure 5 - Showing example gestures from the implemented hand model

The red spheres in Figure 5 are representing the four defined points (joints and the fingertips) on each finger. The bodies of the phalanges are drawn as parallel circles; this is done for the purpose of using their coordinates to achieve a collision detection mechanism.

Relying on the visual outcome of the kinematics equations seems to be safe enough, since the domain of the joint angles is not so vast. Usually Θ_{DIP} is changing between 0 and 90 degrees, Θ_{PIP} changes between 0 and 120 degrees, and Θ_{MCP} has the domain of -30 to 90 degrees. The wrist angles are also tested in all of the 360 possible values. We have performed an exhaustive test for testing all different states of the hand.

4.2.2 Mapping the Target End-Point Positions to Joint Angles

Mapping the joint angles to fingertip positions was discussed in the previous section, but that is the first step. In fact, what we actually need is the reverse: we need to extract finger joint and wrist angles, from the given coordinates of the target end-point for the fingertip. While playing the guitar, guitarists know where each note can be on the fretboard; by having the position for the target endpoint (position on the guitar fretboard) they move their fingers and change the joint and wrist angles, so that the fingertips can reach the target positions.

For any given point for the fingertip, there are multiple options for the finger *base,* joint angles, and the wrist angles; each producing a different gesture. Thus, in order to find plausible values, since the angle variables won't have unique answers, and the hand needs multiple options to use for the decision making process, a search procedure has to be done.

The strategy to find gestures from the given positions is, to perform a search on all possible values for every parameter of the hand model. Basically, for a given *Base* point and wrist angles, we first reposition the hand, and then search the finger joint and wrist angle domains to find a state in which the fingertip has an *epsilon* distance from the target point.

Since in this approach we are examining every joint on each finger and the wrist, this is a suitable place to include some limitations to model the disabilities of the user. If any of the joints have limitations, or even if the user prefers not to use a particular finger, those can be marked and, the solution will be changed according to the new limitations.

Performing the described search method for all possible combinations of two triplets (*string, fret, finger*) will give us a database with more than 50,000 gesture samples. Using this method, though it takes a significant initialization time to fill the database, results in a much

faster processing step; otherwise we have no choice other than having a long midpoint preprocessing step every time a new music note input is fed in.

4.2 Using the Hand Model as the Cost Identifier

This is the decision making step; which includes selecting gestures for every two consecutive notes of the music score, in a way that the selection minimizes the cost of movement. Using the database of the gestures, every possible combination of two triplets is accessible via a simple query. In comparing two gestures, finger *base* indicates horizontal and vertical hand displacements on the fretboard; the wrist angles of the two gestures indicate how much the hand needs to rotate to be able to move from one gesture to the next; and the finger angles show the finger displacements. Selecting the cost function from all the possible choices the hand model gives, is the most important factor in the quality of the results; which had to be done empirically for different situations. After the gesture selection process is done, we will assign the cost of the selected gesture to the corresponding edge in the graph.

The most common factor to use for the cost function is the hand displacements (*Base* point). $\Delta Y(Base)$ and $\Delta Z(Base)$ indicate horizontal and vertical hand displacements respectively (for the sake of consistency, only the *Base* point of the index finger is used). Additionally, changing the wrist angles while playing fast notes has drastic effects on the economy of movement, except for minor rotations around the X axis. Also, for playing simultaneous notes, Θ_{MCP2} is a suitable metric for the hand pressure (the less the sum of its absolute value for every finger is, the less pressure the hand bears while maintaining the gesture). Finally, the rest of the hand variables can be used as metrics if more than one gesture has passed the previous filters (looking for the smallest changes in them).

The next step in finding the solution will be performing a shortest path algorithm on the graph of notes. We have used Dijkstra's shortest path algorithm (dynamic programming).

4.3 Animating the Hand Model

After selecting the shortest path in the graph of notes, animating the hand is only a matter of playing with the gestures for different timings. The start time and playing duration of each note is available in the input, so we can infer how long a finger can be free before playing each note. We assume a time frame of

$$t_f = max(1\ second,\ idle\ time) \tag{16}$$

for the finger to animate. Applying the mathematical model developed by *Flash and Hogan, 1985* [34] to the finger, a natural movement for the finger starting from t_f to t_s (start time) is generated. The movement has to minimize Eq. 17 (the time integral of the square of the magnitude of jerk)[2].

$$c = \tfrac{1}{2} \int_{t_f}^{t_s} [(d^3x / dt^3)^2 + (d^3y / dt^3)^2 + (d^3z / dt^3)^2]\, dt \tag{17}$$

Where x, y, and z are the coordinates of the time varying positions of the points defined on the finger. For the animating finger, we are only using the varying *X* coordinate of the finger at time *t*, for the sake of simplicity. According to [35] the above optimization problem can be solved, and the *X* coordinate of the moving point can be uniquely determined using Eq. 18.

$$x(t) = x_f + (x_s - x_f)\,[\,-6(t / t_s)^5 + 15(t / t_s)^4 - 10(t / t_s)^3\,] \tag{18}$$

[2] According to the authors, the ½ coefficient has no significance, and it is only used to produce prettier results

We then insert the gesture which includes the returned finger coordinates, for its corresponding timestamp, into the list of gestures to be shown in the animation. To improve the animation, the same equation is applicable for both *Y* and *Z* coordinates. See Eq. 19 and 20.

$$y(t) = y_f + (y_s - y_f) \left[-6(t / t_s)^5 + 15(t / t_s)^4 - 10(t / t_s)^3 \right] \quad (19)$$

$$z(t) = z_f + (z_s - z_f) \left[-6(t / t_s)^5 + 15(t / t_s)^4 - 10(t / t_s)^3 \right] \quad (20)$$

CHAPTER 5: IMPLEMENTATION AND RESULTS

5.1 System Design and Implementation

5.1.1 High-level View of the System Components

Figure 6 shows the high level view of the components in the system.

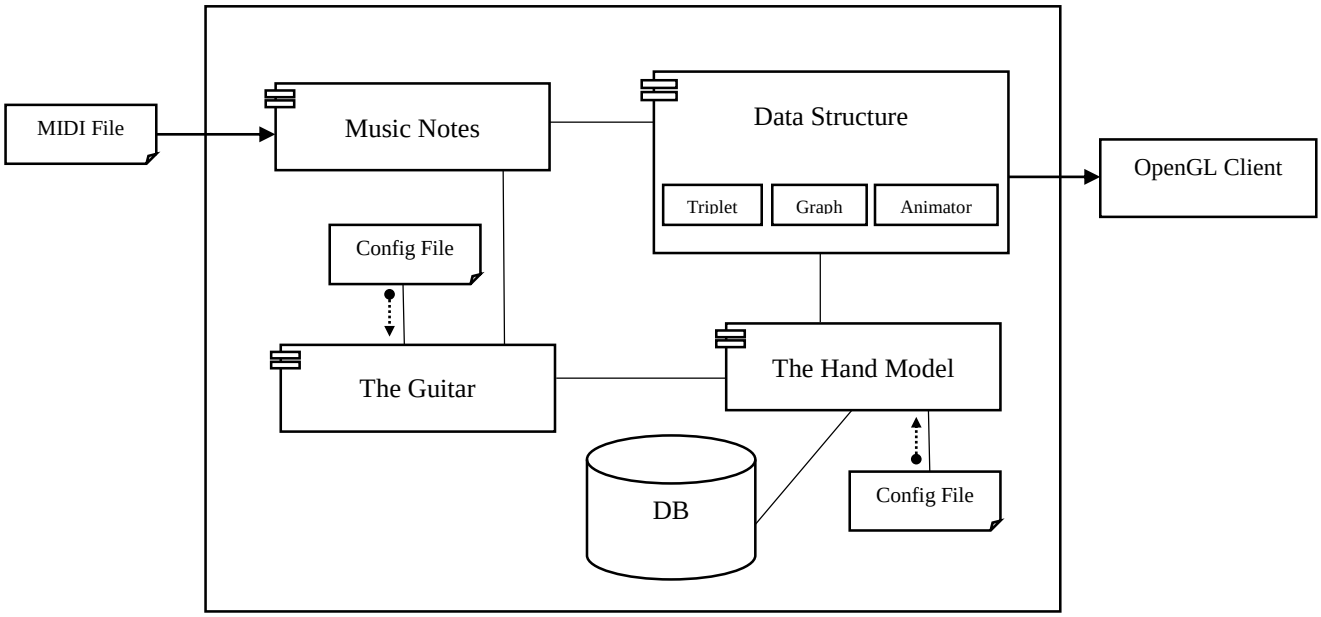

Figure 6 - View of system components

5.1.2 System Implementation

All of the system components are implemented using Java. For reading MIDI files, we have used the package javax.sound.midi, which provides every functionality we need for working with the MIDI files. For the graphical environment, as the diagram shows, we have used OpenGL in Java (jogl); and for the database we have used MySQL. Figure 7 shows the class diagram.

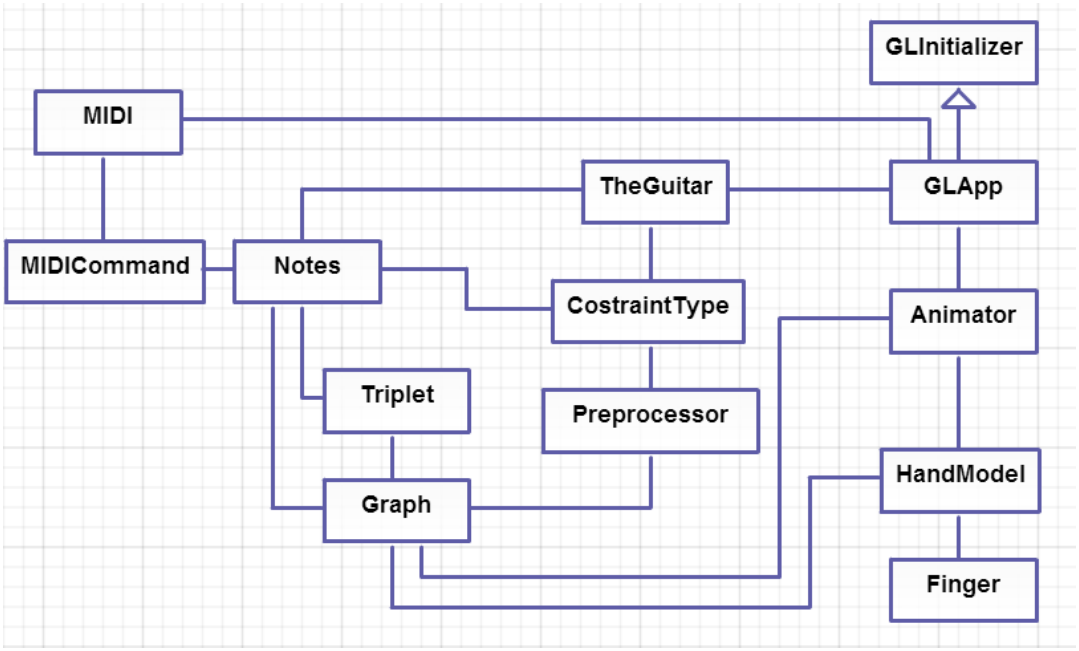

Figure 7 – Class Diagram

5.1.3 Sequence of Events

The following describes the flow of events when a new MIDI file is fed in:

1) The Music Notes entity will read the file, and tells the Guitar entity to tune itself according to the tuning of the guitar in the music score; 2) The Guitar will create the 3D positions of the notes in the Cartesian space, based on its tuning and the configuration file; 3) The graph of notes will be created with all possible triplets for playing each note; 4) The Music Notes and the Guitar entities will then update the graph based on their constraints (explained in Chapter 3); 5) The hand model will then assign weights to the edges of the graph based on the gestures from the DB (the DB is filled/updated after changing the configuration file of the hand model); 6) The shortest path of the graph of notes will be identified; 7) The gestures corresponding to the triplets of the shortest path will be inserted into the queue of the Animator entity at their corresponding times and, as described in Chapter 4, additional gestures will be added to improve the animation (adding more scenes); 8) The OpenGL engine will show the queue of the Animator and will play the MIDI file simultaneously.

5.1.4 Verification of the Components

In order to verify the correctness of the implementation of components, we mostly have used black box testing, with the following details for different situations:

- Reading the input file: This step in the program includes converting the entire MIDI messages (short messages which are used for describing the notes; see Appendix A, section 2) into "Note" objects in the system. In order to test this step, we recreated MIDI messages from the "Note" objects and then played the sequence of the new MIDI messages, and checked the playback.
- Creation of the Guitar entity: aside from the visual representation (which denotes the correctness of drawing the guitar; hence, the correctness of reading the configuration file), we drew labels on top of the guitar frets for each string, showing the expected note which the fret produces (on the corresponding string). This will test the creation of guitar entity (positions of the notes), and the tuning.
- The constraints of the music notes and the guitar: After applying the constraints (described in Chapter 3), we created logs of the graph of notes, specifically focusing on the proper constraint identification. Table 1 in Chapter 3 shows the results of this logging procedure.

5.2 The Results

These frames are captured from the animation created by our implementation. Figure 8 shows the sequence of movements for fingering of C-Major scale, and Figure 9 shows the fingering of F-Minor scale.

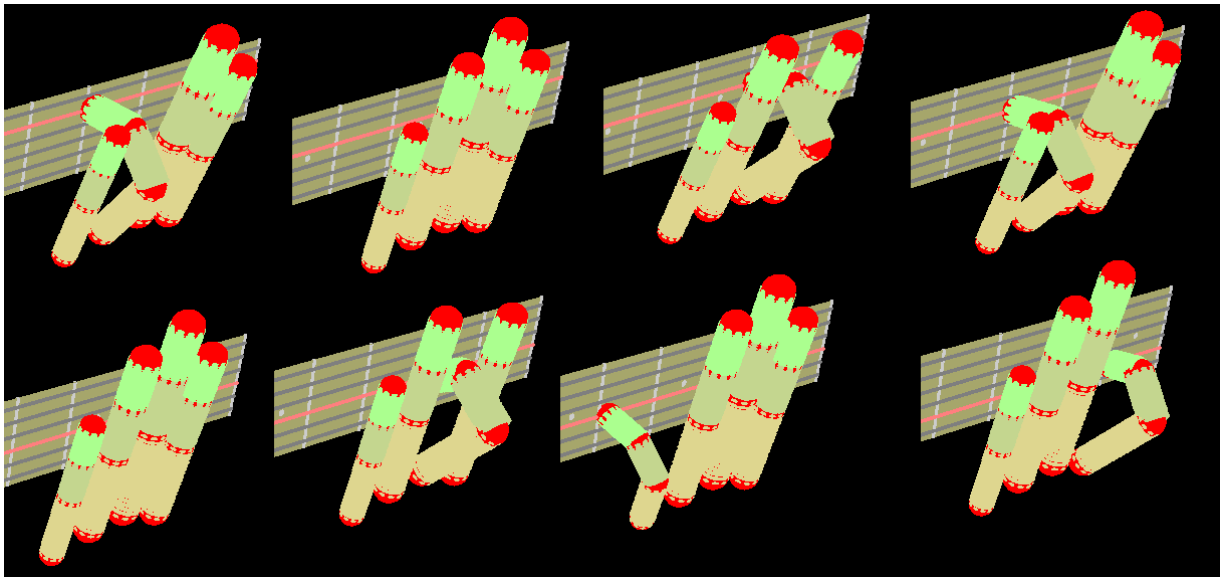

Figure 8 - Showing the scenes of playing C-Major scale

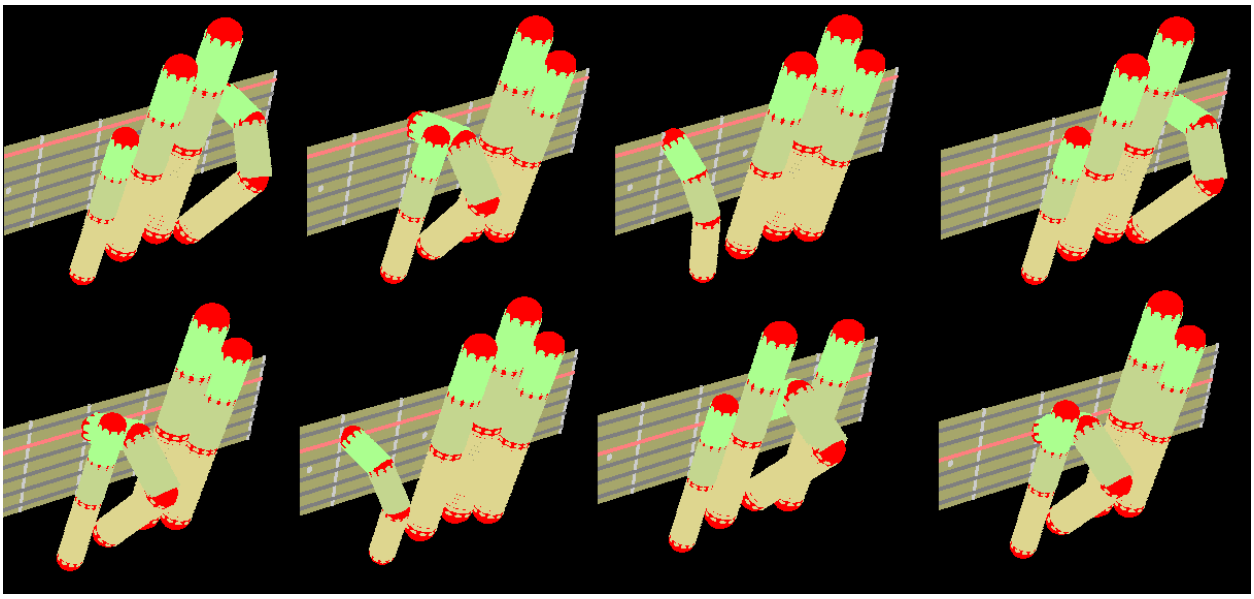

Figure 9 - Showing the scenes of playing F-Minor scale

Note that for the sake of simplicity, and according to our first assumption in Chapter 1, the free fingers are shown at their default state (all joints are set to 0). In fact, since the movements of the fingers are not influenced by one another's, we don't care about the unused fingers. They can be changed in any way that is most comfortable for the user.

Figure 10 shows the fingering of the chord progression C-Am-Dm-G-G7.

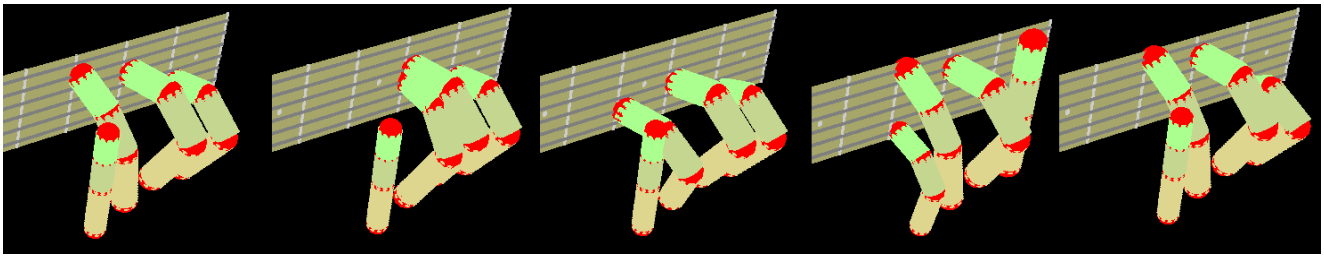

Figure 10 - Showing the scenes of playing chord progression C-Am-Dm-G-G7

5.3 Discussion: Improvements Compared to the Previous Methods

Using our proposed method, we can produce results arguably better and more complete than the previously existing methods. It is fairly hard to compare the results of two methods in studies of the guitar fingering problem, since the evaluation benchmark of none of those methods is published. If we need to compare the results, we first need to implement their method thoroughly, then we need to produce the fingering of their mentioned pieces and compare it to the expert opinion they used for evaluating their results. Thus, comparing the results is often not an option. What we can do though, is comparing the abilities of the two methods. Following is the two main areas that make our proposed method stronger than the previous methods (as we have mentioned in section 1.1):

1) The cost models of every previous method is a static function; for example, they are using Δ(fret), Δ(finger), and Δ(string) as metrics [4]. The problem is, the fret lengths become shorter as their number increases; the space between the strings is not a constant, and has a different value for different types of the guitar. So, relying only on fret numbers and string numbers is not a good decision. As an example, it is hardly possible for many people to hold the gesture defined with the following <*string, fret, finger*> triplets (on a standard electric guitar) : <1, 1, 1> and <2, 4, 2>. Δ(fret) here is 3, and Δ(finger) is 1; but it is fairly easy to hold this

gesture: <1, 19, 1> and <2, 24, 2>. Δ(fret) here is 5, and Δ(finger) is 1. As we can see, it is not a good decision to rely on fret numbers. Same argument is applicable for using string numbers. With the use of our hand model, we are only considering distances and relying on the lengths; so the issue will be resolved.

2) Including new guitar playing techniques (as discussed in Chapter 3) is another major improvement compared to the previous methods [5, 6, 36, 4, 3].

Finally, this study introduces a new approach, and due to the time constraints, it was not possible for us to use a more complete hand model. Expanding ideas for this study is discussed in the following chapter; our method has a great potential for future extensions.

CHAPTER 6: CONCLUSIONS AND FUTURE WORK

This thesis proposes a method for fingering of the music piece on the guitar. In our model we are using the coordinates of the guitar frets as the target end-points for the fingertips. Hence, we can find the fingerings for any type of guitar with any tunings. Additionally, the hand is modeled using a set of kinematics equations. Using Forward Kinematics, we are able to successfully identify the suitable gestures for playing the input. Also, the hand model can take into account the disabilities and preferences of the user for playing the instrument. Using this approach, the fingering of the music piece is determined by the hand, as opposed to using hard coded cost functions and movement definitions by imagining static comfortable spans.

This study sets up the groundwork for using a more complete hand model for the fingering problem, which results in a better animation, and requires less time for initializing the hand model. The main drawback of our method is the time consuming initialization of the hand (filling the database of gestures); so, for future extensions we can imagine a better gesture identification process which does not require the initialization step, or can do it faster. The future extensions can also include the hand dynamics, such as angular velocities of the joints, the skin layer, muscles, and tendons. Additionally, other guitar playing techniques can be accommodated to improve the answers, such as implementation of barrét positions, and an extended tapping mechanism.

Bibliography

[1] "HCI (Wikipedia), Downloaded on November 3rd, 2014," [Online]. Available: http://en.wikipedia.org/wiki/Human%E2%80%93computer_interaction.

[2] P. K. John, "Advanced Human Hand Model with Dynamic Constraints," *The Australian National University,* 2006.

[3] D. P. Radicioni and V. Lombardo, "Computational modeling of chord fingering for string instruments," *strings, vol 40, no 45,* p. 50, 2005.

[4] D. Radicioni and V. Lombardo, "Guitar fingering for music performance," *strings,* p. 50, 2005.

[5] G. ElKoura and K. Singh, "Handrix: animating the human hand," *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation,* pp. 110--119, 2003.

[6] S. I. Sayegh, "Fingering for string instruments with the optimum path paradigm," *Computer Music Journal,* pp. 76-84, 1989.

[7] A. Radisavljevic and P. Driessen, "Path difference learning for guitar fingering problem," *Proceedings of the International Computer Music Conference,* 2004.

[8] H. Rijpkema and M. Girard, "Computer animation of knowledge-based human grasping," *ACM Siggraph Computer Graphics,* pp. 339--348, 1991.

[9] J. Lee and T. L. Kunii, "Model-based analysis of hand posture," *Computer Graphics and Applications, IEEE,* pp. 77--86, 1995.

[10] C. Nolker and H. Ritter, "Visual recognition of continuous hand postures," *Neural Networks, IEEE Transactions on,* pp. 983--994, 2002.

[11] S. Kirkpatrick, G. C. Daniel, M. P. Vecchi and others, "Optimization by simmulated annealing," *science,* pp. 671-680, 1983.

[12] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Machine learning 3, no. 2,* pp. 95-99, 1988.

[13] C.-C. Lin and D. S.-M. Liu, "An intelligent virtual piano tutor," *Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications,* pp. 353-356, 2006.

[14] M. Hart, R. Bosch and E. Tsai, "Finding optimal piano fingerings," *The UMAP Journal 2, no. 21*, pp. 167-177, 2000.

[15] Y. Yonebayashi, H. Kameoka and S. Sagayama, "Automatic Decision of Piano Fingering Based on a Hidden Markov Models," *In IJCAI*, pp. 2915-2921, 2007.

[16] J. Kim, F. Cordier and N. Magnenat-Thalmann, "Neural Network-based Violinist's Hand Animation," *Computer Graphics International, Proceedings*, pp. 37-41, 2000.

[17] R. Parncutt, J. A. Sloboda, E. F. Clarke, M. Raekallio and P. Desain, "An ergonomic model of keyboard fingering for melodic fragments," *Music Perception*, pp. 341-382, 1997.

[18] D. R. Tuohy, "Creating tablature and arranging music for guitar with genetic algorithms and artificial neural networks," *PhD Thesis, University of Georgia*, 2006.

[19] T. B. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Computer Vision and Image Understanding, vol 81, no 3*, pp. 231-268, 2001.

[20] L. J., J. Chai, P. S. Reitsma, J. K. Hodgins and N. S. Pollard, "Interactive control of avatars animated with human motion data," *ACM Transactions on Graphics (TOG), vol. 21, no. 3*, pp. 491-500, 2002.

[21] M. Gleicher, J. S. Hyun, K. Lucas and A. Jepsen, "Snap-together motion: assembling run-time animations.," *ACM SIGGRAPH 2008 classes*, p. 52, 2008.

[22] S. R. Buss, "Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods," *IEEE Journal of Robotics and Automation 17*, pp. 1-19, 2004.

[23] B. Paolo, "Inverse kinematics techniques for the interactive posture control of articu-lated figures," *PhD Thesis, Ecole Polytechnique Federale de Lausanne*, 2001.

[24] K.-J. Choi and H.-S. Ko, "On-line motion retargetting," *In Computer Graphics and Applications. Proceedings. Seventh Pacific Conference on. IEEE*, pp. 32-42, 1999.

[25] M. Bray, E. Koller-Meier, P. Muller, L. Van Gool and N. Schraudolph, "3D hand tracking by rapid stochastic gradient descent using a skinning model.," *1st European Conference on Visual Media Production*, pp. 59-68, 2004.

[26] E. S. Ho, T. Komura and R. W. Lau, "Computing inverse kinematics with linear programming," *Proceedings of the ACM symposium on Virtual reality software and technology,*, pp. 163-166, 2005.

[27] M. Fêdor, "Application of inverse kinematics for skeleton manipulation in real-time," *Proceedings of the 19th spring conference on Computer graphics*, pp. 203-212, 2003.

[28] I. Albercht, H. Jorg and S. Hans-Peter, "Construction and animation of anatomically based human hand models," *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 98-109, 2003.

[29] D. Baraff, "Linear-time dynamics using Lagrange multipliers," *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, pp. 137-146, 1996.

[30] J. L. Sancho-Bru, A. Perez-Gonzalez, M. Vergara-Monedero and D. Giurintano, "A 3-D dynamic model of human finger for studying free movements," *Journal of Biomechanics 34, no. 11*, pp. 1491-1500, 2001.

[31] W. Tsang, K. Singh and E. Fiume, "Helping hand: an anatomically accurate inverse dynamics solution for unconstrained hand motion," *Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pp. 319-328, 2005.

[32] Y. Wu and T. S. Huang, "Human hand modeling, analysis and animation in the context of HCI," *Image Processing, 1999. ICIP 99. Proceedings. 1999 International Conference on, vol. 3*, pp. 6-10, 1999.

[33] T. Heap and D. Hogg, "Towards 3D hand tracking using a deformable model.," *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*, pp. 140-145, 1996.

[34] T. Flash and N. Hogan, "The coordination of arm movements: an experimentally confirmed mathematical model," *The journal of Neuroscience*, pp. 1688--1703, 1985.

[35] E. Lindo Secco and G. Magenes, "Bio-Mimetic Finger: Human Like Morphology, Control & Motion Planning for Intelligent Robot & Prosthesis," *Mobile Robotics, Moving Intelligence, Jonas Buchli (Ed.), ISBN: 3-86611-284-X, InTech, DOI: 10.5772*, 2006.

[36] S. I. a. T. M. F. Sayegh, "Inverse Viterbi algorithm as learning procedure and application to optimization in the string instrument fingering problem," *Neural Networks, 1988., IEEE International Conference*, p. 1988, 491-497.

[37] "MIDI (Wikipedia), Downloaded on October 25th, 2014," [Online]. Available: http://en.wikipedia.org/wiki/MIDI.

APPENDIX A: BACKGROUND

About the Guitar

The guitar is a musical instrument consisting of three major parts: the body, the neck, and the nut. The body in different types of guitars plays the role of producing the tone. In acoustic guitars, the body has a hollow form which acts as a resonating chamber. In electric guitars, the body holds the pick-ups, and they can pick up the signal from the strings and pass it to the amplifier in order to produce the sound. Figure 11 shows the major three parts.

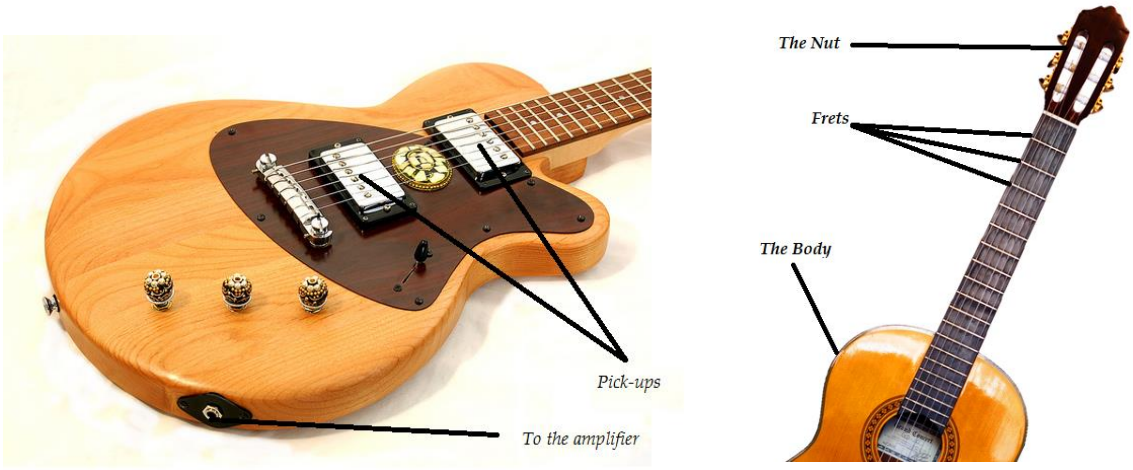

Figure 11 - Major guitar parts, acoustic (right) and electric (left)

The neck of the guitar is where the frets are placed. The frets are indicators of the spaces between the notes on the strings. A position on the guitar neck is defined with two variables: String, and Fret. Placing a finger on any position would result in sounding a specific note. The neck of the guitar is also called the fretboard.

The guitar usually has six strings, and can have up to 24 frets. Depending on the tunings of the strings, each fret on each string produces a specific note. The nut has a knob for each string, and is where we can tune the strings.

These are the parameters of the guitar which determine the playing of the notes: number of strings, the space between the strings (it is different for different types of guitars), the length of the neck (which will determine the space between the frets), and most importantly, the tuning of the strings.

The fretting hand (for right-handed people it is their left hand) is a term referring to the hand which performs the act of fretting. Fretting is done by pressing down a string on a fret as close as possible to the iron indicating the fret. The picking hand (the right hand for right-handed people) refers to the hand which performs the picking. Picking is the term referring to the act of striking the strings with a small plastic object to vibrate the string, to ultimately produce a sound (other techniques for the right hand are also imaginable, like playing with the finger nails).

What has been done in this study helps to have a separate entity for the guitar, in a way that all of those parameters are present in the guitar entity, and it can tell the hand object where each note is in the Cartesian space. So, any music sheet written for any type of guitar can be solved using our method.

Depending on the tunings of the strings, a particular note can be present on up to 6 positions on the guitar neck (because the guitar usually has 6 strings). When using the standard tuning (E3 A3 D4 G4 B4 E5) only E5 is repeated 6 times, as shown in Figure 12.

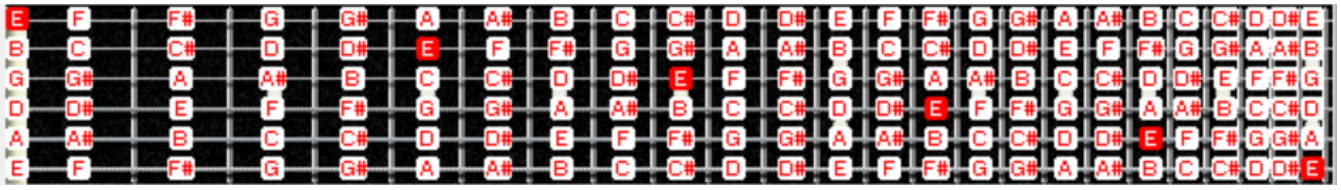

Figure 12 - Showing the positions of the notes for the standard tuning

The difficulty of finding a solution for the music piece on the guitar comes from this fact. We need to decide between all these possible options for every note, in a way that the movements of the hand are minimized (economy of movement).

About MIDI files

According to [37] "MIDI (short for Musical Instrument Digital Interface) is a technical standard that describes a protocol, digital interface and connectors and allows a wide variety of electronic musical instruments, computers and other related devices to connect and communicate with one another. A single MIDI link can carry up to sixteen channels of information, each of which can be routed to a separate device". MIDI files are a MIDI sequence saved into a file in order to be played later. A MIDI sequence is a sequence of MIDI messages coming from a MIDI device or sequencer software, each message containing certain information for the piece of music. MIDI has lots of applications and technical details. For more information see the Wikipedia entry for MIDI [37].

A MIDI message can be one of three types: short message, system exclusive message, and control message. In this study, we have only used short messages.

Each message has a tick number, indicating where the message happens in the sequence. To convert the tick to actual timings, we must infer the tempo of the piece first. Java MIDI class (javax.sound.midi) provides a very good toolset to use MIDI sequences.

MIDI short messages that are of interest to this study are: NOTE_ON, NOTE_OFF, and PITCH_BEND. Each short message consists of three bytes: status byte, data1, and data2. Table 2 shows the meaning of these bytes for these three types of messages.

Message	Status (in hex)	Data1	Data2
NOTE_ON	9x	Note number	Velocity
NOTE_OFF	8x	Note number	
PITCH_BEND	Ex	MSB	LSB

Table 2 - MIDI Messages

A NOTE_ON message with the velocity of 0 can also mean NOTE_OFF.

For PITCH_BEND messages, the LSB does not play a role, and the MSB is the controller of the pitch. The MSB determines how much a note (for which the NOTE_ON message has been sent already) should be bent. The PITCH_BEND messages in the MIDI sequence are listed after the NOTE_ON messages, and finding out which PITCH_BEND corresponds to which NOTE_ON can be confusing in certain scenarios.