Eurographics/SIGGRAPH Symposium on Computer Animation (2003)
D. Breen, M. Lin (Editors)

Handrix: Animating the Human Hand

George ElKoura[1,2] and Karan Singh[1]

[1] Department of Computer Science, University of Toronto, Toronto, Canada
[2] Side Effects Software, Inc., Toronto, Canada

Abstract

The human hand is a complex organ capable of both gross grasp and fine motor skills. Despite many successful high-level skeletal control techniques, animating realistic hand motion remains tedious and challenging. This paper presents research motivated by the complex finger positioning required to play musical instruments, such as the guitar. We first describe a data driven algorithm to add sympathetic finger motion to arbitrarily animated hands. We then present a procedural algorithm to generate the motion of the fretting hand playing a given musical passage on a guitar. The work here is aimed as a tool for music education and analysis. The contributions of this paper are a general architecture for the skeletal control of interdependent articulations performing multiple concurrent reaching tasks, and a procedural tool for musicians and animators that captures the motion complexity of guitar fingering.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation

1. Introduction

The human hand is an essential part of human form, function, and communication, capable of complex, expressive articulation. Its complicated neuro-physiology makes it a formidable challenge for animators to emulate. Most computer graphics research on hand motion has focused on grasping and gestures [27, 12] with application to areas of robot planning, prosthetics, human computer interaction and sign language.

There are also a number of pervasive applications such as typing, the playing of musical instruments and many sports, where fine motor control of the hand is used to accomplish multiple concurrent reaching tasks. The motor skills for these applications are learned over time and the development of form and technique is an important area of study for educators [21]. These applications are also a major source of repetitive stress injuries and thus of great importance to medical research [5, 20]. From a computer graphics standpoint, these applications are difficult to animate realistically and are relatively unexplored.

Kinematics, dynamics and recent motion-capture based techniques provide character animators with high-level skeletal control. These approaches by themselves fail to capture the subtleties of hand motion performing concurrent reaching tasks for two main reasons. The first is that typical inverse kinematics (IK) algorithms are designed to deal with constraints along a single chain, whereas a multi-appendage limb like the human hand has constraints between joints of different chains. The second shortcoming is that IK solutions typically map a single end-effector to a single reaching goal [28, 33]. A sequence of reaching tasks must, therefore, be performed in order by the same end-effector. There is an exponential increase, however, in the number of ways by which a multi-appendage limb may satisfy a sequence of multiple concurrent reaching tasks, such as playing a sequence of chords on a guitar (Figure 10). This is further complicated by the fact that sometimes an entire finger may be used as a large end-effector to satisfy multiple reaching goals simultaneously (shown by the index finger playing a bar chord in Figure 10(c)). Our research is motivated by music education in general and the guitar in particular, where not only must the player decide on a choice of fingering but also do so with economy of effort and as little sympathetic finger motion as possible.

An architecture that maps multiple-appendages to multiple goals must make decisions on the choice of end-effectors with which to satisfy a set of reaching goals. These decisions are based on various appendage parameters and constraints, the geometry and current position of the articulations with

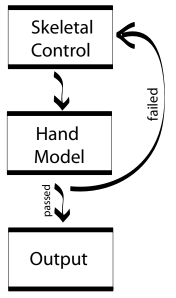

Figure 1: *System architecture overview.*

respect to each other, and the specified reaching goals. Once end-effectors have been mapped to the given goals, the problem reduces to the traditional single chain scenario, where we can apply a wealth of well-established kinematic, dynamic and even motion-capture based algorithms. The results of the single chain solutions then need to be integrated to account for joint interdependencies across articulations. This architecture is shown in Figure 1. Such a modular abstraction of joint interdependencies only works well on the assumption that the incremental sympathetic joint motions are small in comparison to overall reaching motion of the multi-appendage limb. In section 3, we show that this can be applied in the context of the human hand. The modular design allows us to improve the quality of arbitrary hand animations by adding any missing joint interdependencies.

The contributions of this work are broken up into two main parts, the first is described in section 3 and the second in section 5. We start by first showing a data-driven approach for modeling the sympathetic motion between fingers of the hand. We use a *k*-Nearest Neighbor search to map arbitrarily specified hand configurations to realistic hand configurations. The fretting hand playing guitar (Figure 9) is one of the most complex applications in which the hand performs a sequence of multiple concurrent reaching tasks. Section 5 develops a procedural algorithm to control the fretting-hand for a given piece of music. The algorithm homogeneously addresses complex parameters such as the size of the hand relative to the guitar, the number, geometry and relative strength of different fingers and the music reading skill of the player. The result is a system that can be used as a tool for music education, comparison and analysis of various playing styles. While the algorithm is designed and described for guitar, the cost minimization approach generalizes to a number of similar problems, such as typing and complex grasps, where fingers choose multiple points of support. In general many aspects of this work have a broader impact on the skeletal control of structures with interdependent appendages performing multiple concurrent reaching tasks.

An overview of the rest of the paper is as follows: section 2 discusses related work. Section 3 describes the design of our hand model and the data-driven approach to modeling sympathetic motion between fingers. Section 4 then provides basic guitar terminology and a definition of the fretting-hand problem, a procedural solution to which is presented in section 5, followed by implementation details in section 6 and a discussion of results in section 7. Section 8 concludes with the scope for future work.

2. Previous Work

The human hand is a fertile area of research in many disciplines. We classify relevant work here in terms of broad area of research.

Anatomy. Studies on the limitations and constraints of the various joints of the hand are well understood. The interdependency of the various joints, however, is largely based on empirical observation [9]. It is that interdependency that we strive to capture in the hand model presented in this paper. The currently accepted theory for the cause of sympathetic movement in the hand is due to a combination of biomechanical and neurological constraints. Biomechanical restrictions are partially due to the muscle and tendon configuration. Muscles, such as the Extensor Digitorum Communis in the forearm have insertions in multiple joints (Figure 2). The activation of such muscles thus results in the excursion of multiple tendons. The tendons also restrict each others motion due to their configuration and close proximity. Neurological constraints are also believed to contribute to the sympathetic motion [30, 7]. Understanding the neuro-physiology of the human hand is still an area of active research and not yet mature enough to construct an anatomically complete model for sympathetic hand motion.

Robotics. Robotics researchers have studied the hand in the context of grasping and manipulation planning [24]. Koga et al. [12] use pre-configured hand postures and choose from among them to select the proper grasp. Much of the work in this area treats the hand as a mitt that can grasp and manipulate objects, but generally does not deal with the fine motor capabilities of the fingers.

Animation Industry. Animators today use a combination of IK and expressions to provide high-level control over kinematics. The spread of the palm (adduction/abduction), the clenching of the fist, and the curling of each finger are examples of high-level forward kinematic control that are usually blended together with inverse kinematics under user control.

Graphics and Vision. Although others have considered branched kinematic chains [32], we address the specific problem reaching of multiple goals by multiple interdependent articulated chains. Vision researchers solve the inverse problem and have employed simplified hand models for image based gesture recognition. Early work on grasping [27] recognized the importance of the interdependence of the hand joints introducing the commonly used joint angle constraint $\theta_{DIP} = \frac{2}{3}\theta_{PIP}$ (see Figure 2). We used the data measured

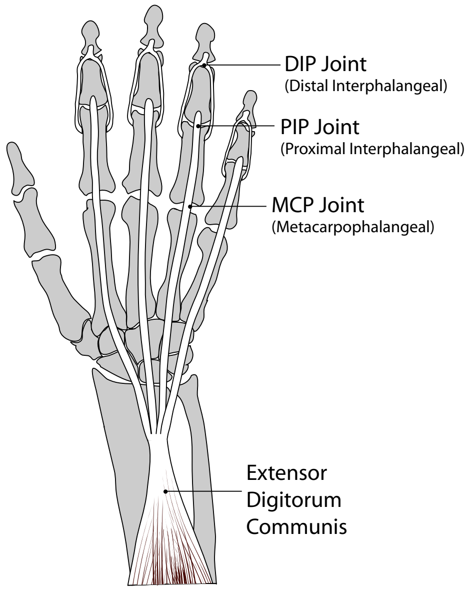

Figure 2: *Bones and joints of the hand and the Extensor Digitorum Communis.*

	Index	Middle	Ring	Little
Average	2.91°	4.24°	0.80°	4.60°
Stddev	19.82°	16.28°	10.80°	15.03°

Table 1: *Average and stddev of the measured value of* $\theta_{DIP} - \frac{2}{3}\theta_{PIP}$ *in degrees over approximately 20,000 samples.*

from a real subject and found that the constraint is too rough an approximation for intricate control of the hand. Further, for applications such as the playing of musical instruments, there is a marked difference in the finger interdependence of players of different skill levels. The average difference between DIP and PIP and standard deviation are summarized in Table 1 and an example of where the assumption falls short is shown in Figure 3.

The hand posture reconstruction work of Lee and Kunii [15] proposed a model that included dependence constraints between the DIP and PIP joints of each finger and among MCP joints of the rest of the fingers. This work, however, does not capture the interdependencies that exist among the DIP and PIP joints of different fingers.

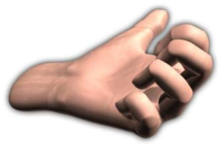

Figure 3: *Example of hand posture where the assumption* $\theta_{DIP} = \frac{2}{3}\theta_{PIP}$ *is inadequate.*

Other work focuses on the skin deformations necessary for realistic hand modeling [22, 11] built upon any arbitrary skeletal control approach.

A lot of interest has recently been given to motion capture techniques [16, 13, 2, 14, 25, 17]. These works show how motion capture data can be segmented and used to effectively synthesize new motion. We also adopt a data-driven approach to generate realistic hand motion. Pullen and Bregler [25] describe a method for using motion capture data to add realism to existing keyframed animation. As in their work we do not explicitly play back complete motion clips but rather use the motion data to correct arbitrarily generated hand animation, shown in section 3.

Music. Our procedural guitar player is a unique computer-simulated aid to music education and analysis of playing style. The marriage of music and computer graphics however is not new and has been applied effectively by many, including Lytle [19, 18], Bargar [4] and Hänninen [10].

Sayegh [29] presents a dynamic programming algorithm for a variant of the fingering problem for string instruments. Unlike Sayegh's solution to a sequence of notes, ours solves a polyphonic problem and handles chords. Our approach also employs various parameters that provide an animator or educator with control over the resulting solution.

3. Hand Model

The human hand has 27 degrees of freedom: 4 in each finger, 3 for extension and flexion and one for abduction and adduction; the thumb is more complicated and has 5 DOF, leaving 6 DOF for the rotation and translation of the wrist [1].

To accurately model the hand, a complete model of the muscles, tendons, bones and a neurological control structure is necessary. The dynamics of such a complex model are still poorly understood, forcing the use of simplified models. Current models [15, 23, 31] are too simplified for our purpose so we turn to recent work from the medical community to motivate assumptions used in a new model that we propose here. We use a 27 DOF model of the hand with the following simplifying assumptions:

1. The thumb is independent of the other fingers.
2. Adduction/abduction of the finger joints are independent.
3. Motion frequency does not affect joint interdependence.
4. Both hands have the same interdependence model.
5. The posture of the wrist and the rest of the arm do not affect the underlying interdependence structure.

Assumption 1 can be verified by casual observation, but is also quantifiably justified by Häger-Ross and Schieber [9].

We thus do not include the 5 DOF thumb in the interdependence model. Abduction and adduction is the side-to-side movement of fingers. That these motions are not affected by interdependence constraints is a simplifying assumption that seems reasonable but is not supported by research known to us. Assumption 3 is known to be false. The independence of the fingers is inversely correlated to their movement frequency. The degree to which independence is affected is further explored by Häger-Ross and Schieber, and may not have a large practical impact for our purposes, however, a more complete model would certainly include this factor. We leave the removal of this assumption as future work. Assumption 4, supported by Häger-Ross and Schieber [9], allows us to talk freely of the hand without having to specify whether we mean the left hand or the right hand. Häger-Ross and Schieber quote findings that finger dexterity is maximized when the wrist is extended 10-20 degrees but in general the effect of the arm and wrist posture on the interdependence of fingers is not clearly understood. We make assumption 5 to reduce the dimensionality of our interdependent DOFs. We observe, though, that our data-driven model can be easily extended to include data for the wrist, arm or any other joints as contributing to the interdependent DOFs.

The above five assumptions leave us with the task of modeling the 12 most strongly interdependent joints of the hand (MCP, PIP and DIP of the 4 fingers).

3.1. Posture Reconstruction

Motion capture of real hands was used to extract the space of possible hand configurations. Two NDI Optotrak 3020 cameras were connected and calibrated to capture the position of 24 IRED markers. Four markers were placed along each finger and three along the thumb, leaving five that were placed along the wrist. The 3D position data of the markers was filtered and converted to joint-angle configurations [6].

Our sample data is a general capture of hand motion drills and finger wiggling, which is used to correct partially specified hand configurations. We specifically did not choose to only include task specific motions so that we may be able to apply the data to arbitrary hand animations. The difference between IK by example approaches (where complete postures are strictly determined by sample data) [28] and our posture correction is subtle but important for problem spaces with requirements such as ours: (a) a complex motion space with singularities; (b) many different ways of realizing the same goals; (c) precise IK requirements with multiple hit points on the same finger like with bar chords; (d) the ability to isolate motion resulting from joint interdependencies. This model is applicable to any animation involving hands and can be turned on/off seamlessly in current animation workflows.

The joint angles resulting from the motion capture process are points, Θ_i's, in a 12 dimensional vector space. Together they represent the physically possible finger joint configuration space. Our problem is as follows: Given a desired (sometimes only partially specified) hand configuration, find a proximal posture in the space of physically attainable postures. Mathematically, we can model the problem as a function that maps a joint angle vector to another joint angle vector. The input to the function is two 12D vectors: a joint angle vector Φ and a weight vector ω; the output is a joint angle vector Ψ. The joint angle vector is a desired hand posture and the weights capture the relative importance of corresponding joint angles. The weight vector for Figure 8, for example, would have the weight values for the joint angles of the index finger (controlled by IK), higher than those for the remaining unconstrained fingers. The output joint angle vector Ψ is the resulting realistic configuration.

The hand model is designed to be a modular and interactive post-process to an arbitrary skeletal control system. The weight vector provides the skeletal control system the capability to specify a partial posture.

The model is implemented using a k-Nearest Neighbors search in a 12D space to retrieve the plausible joint angles. Firstly a choice for k must be made. If sufficient data exists, a choice of $k = 1$ returns the nearest neighbor, which is guaranteed to be a valid configuration since it was captured from a real hand. In the absence of sufficient data, we choose a larger k and interpolate between the k nearest neighbors. In practice, a value of $k = 3$ or $k = 4$ gives good results for a database of $\approx 20,000$ samples.

We use a weighted Euclidean distance metric given by

$$\delta_i = \sqrt{\sum_{j=1}^{12} \omega_j((\Theta_i)_j - \Phi_j)^2} \tag{1}$$

The interpolation weights for the k nearest neighbors are defined by

$$w_i = \frac{e^{-\beta\delta_i}}{\sum_{j=1}^{k} e^{-\beta\delta_j}} \tag{2}$$

where δ_i is the distance of the ith nearest neighbor and β is a decay parameter.

The output joint vector Ψ is an average of the joint vectors of the k nearest neighbors, proportional to their interpolation weight. It is worthwhile noting that unless $k = 1$, there is no guarantee that the interpolated configuration is physically possible. However, unless the data is extremely sparse, blending between a small number of k nearest neighbors does not produce undesirable results. We also note that since the hands configuration space is not necessarily convex, it is possible, even with dense sampling, for the k nearest neighbors of an input posture to have very different configurations. This is especially true of partially specified input postures. We should thus take care to pick a single cluster of spatially proximal neighbors for blending to be valid. The algorithm bottleneck is the k-Nearest Neighbors search. We found a k-D tree based scheme to give us interactive results for 20,000 data points in a 12 dimensional space. For higher dimensionality, approaches such as the approximate nearest neighbor [3] may provide better results.

3.2. Joint Blending

A problem with modifying an IK solution is that the guarantee of reaching a given target is lost. A similar problem exists when modifying a dynamic solution due to collision response or constraint satisfaction. While the weight vector can strongly bias the hand model to preserve the angles on joints controlled by the input skeletal controller (see Figure 1), a precise solution may be desirable. This can be solved by an iterative skeletal control and hand model loop, converging to a desired accuracy. As there is no guarantee of convergence, the loop can be made more efficient in practice by simply blending in the allowable corrections the hand model can make to any given joint angle. This blend is based on another weight vector provided by the skeletal controller, and brings the computed hand posture closer to the desired solution.

3.3. Joint Retargeting

The hand-data is stored as joint-angle vectors; no other concept of space is required for our purposes. Retargeting the data model to a hand of different size is thus straightforward, so long as we are not concerned with space-constraints external to the hand. Detecting object grasps, and other specific fingertip constraints can be done in the way described by Gleicher [8]. Whether the sympathetic motion between fingers is affected by the size of the hand itself remains to be explored.

4. The Guitar

The guitar is among the most complex instruments to play with respect to overall hand motion. The fretting-hand problem is the task of providing the skeletal control of the fretting-hand (see Figure 9) needed to play a given piece of tablature (see Figure 8). This problem, of great importance to guitar players and teachers, inspires this research and illustrates the complexity of hand motion required to perform multiple concurrent reaching tasks. In this section we provide sufficient background and guitar terminology.

A guitar produces sound through the vibration of strings (typically 6) that are stretched across its neck as shown in Figure 4. The strings are numbered from 1 (highest pitch, lightest gauge) to 6 (lowest pitch, heaviest gauge). The musician can play different notes by pressing down on the string at calibrated locations along the neck, called frets. The frets are numbered in increasing order from the nut (fret 0).

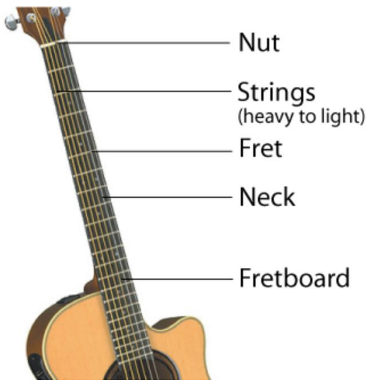

Figure 4: *The anatomy of a guitar.*

The two hands play different but equally important roles. The dominant hand typically strums or plucks the strings. The fretting hand controls the notes played by pressing the string at a fret.

Tablature is a commonly used form of guitar notation (Figure 7). Six horizontal lines each represent a string (1 to 6 from top to bottom) on a guitar. Notes are played in time from left to right. Numbers are placed on lines to indicate which fret on the corresponding string must be held down. We use an augmented form of tab notation in which the spacing between notes indicates the relative timing of the notes. Note that tablature does not tell the player what finger to use but rather provides a sequence of reaching goals that must be met by some combination of fingers.

5. Procedural Guitar Player

This section contains a description of an algorithm that solves the fretting-hand problem. The use of such an algorithm as a music education tool drives our design objectives.

5.1. Design Objectives

Playing the guitar is an art that offers limitless expressive capabilities. We would, therefore, like to provide a wide range of control over the guitarists fretting hand, while adhering to basic playing principles and maintaining a level of realism throughout. We would also like a system that can produce the fretting hand of a novice as readily as that of an expert player, allowing players to learn a given piece by adjusting style and skill level. The following objectives help us meet these requirements.

1. Minimize the amount of motion produced while playing. The “economy of effort” [26] is recommended for the fretting-hand.
2. Control over how far ahead a piece of music is read. This is a skill level objective that helps model novice players by restricting the ability to read ahead in the music.

3. Maintain the natural posture of the fingers as much as possible, keeping each finger over a different fret. Finger motion should favor the natural arc of the fingers.
4. Capture the difference in dexterity and strength of different fingers. For example, shorter and weaker fingers, like the little finger, have a harder time playing the heavier strings.
5. The fingertip must touch the guitar fretboard as close to the desired fret as possible in the region behind it.

A cost-minimization approach allows us to frame these objectives as cost functions. Economy of effort is achieved by assigning a cost to the motion of the wrist and fingers. Look-ahead in a passage is controlled by performing the optimization localized over a window of a size specified by the user. Beginners will typically have a smaller window than expert players. Controlling skill level is also achieved by altering cost functions to produce higher costs for fingers with which the player is unskilled. The cost functions are also used to embody the structure of the hand and the relative position, strength and dexterity of the fingers. The last objective is met by first assuming the ideal position behind a fret followed by a simple geometric post-process to adjust the fingers into a collision free configuration behind the fret.

5.2. Choosing the Cost Functions

The final fingering of the given musical score is largely determined by the chosen cost functions. As an example, a low cost for the wrist and index finger and high cost for the other fingers would result in notes being played by the index finger alone.

Multiple cost functions can be associated with each finger to reflect various attributes. The input to a geometric cost function is the displacement between a target position and the current position of the finger. This cost also captures the reachability of a finger. The input to a dexterity function would include the time given to reach the target, which would influence the cost due to acceleration constraints of the finger. We use a constant dexterity function leaving the extension to a more complete cost function model as future work. The strength cost function, in our context, is controlled by the gauge of the string to be played and the strength of each finger. Other stylistic attributes such as finger preference are also modeled as individual cost functions.

The geometric cost function is particularly important in a kinematic setting. The functions are typically smooth, convex and have a unique minimum for zero displacement. Smoothness and convexity assure a reasonable placement and that fingers do not teleport. The unique minimum achieves economy of motion.

The description below is in a reference frame with the origin at the guitar nut, the x-direction along the guitar neck, the y-direction parallel to the frets and the z-direction perpendicular to the plane of the neck. We use the following function

$$f_i(\mathbf{c}) = \begin{cases} \lambda_i \alpha_i ||\mathbf{d}||^{m_i} & \text{if } \mathbf{d}_x \geq 0, \\ \lambda_i \beta_i ||\mathbf{d}||^{m_i} & \text{if } \mathbf{d}_x < 0 \end{cases} \tag{3}$$

to calculate the cost of the moving the finger i by the displacement vector $\mathbf{d} = \mathbf{c} - \mathbf{p}_i$, where $\mathbf{c}$ is the target position and $\mathbf{p}_i$ is the current position of the finger and $\mathbf{c}, \mathbf{p}_i, \mathbf{d} \in \Re^3$.

The functions are split into two parts to capture the different ability of fingers to move forwards and backwards a fret. The natural arc of the hand makes it such that the index and middle fingers can reach forwards easier than can the ring and little finger. This property is captured by the constants α_i and β_i. Setting $\alpha_1 < \beta_1$ would thus indicate that the index can move forwards easier than it can move backwards along the neck of the guitar. The choice for α_i's and β_i's is usually fixed by the system designer, and we recommend $\alpha_1 < \beta_1$, $\alpha_2 < \beta_2$, $\alpha_3 > \beta_3$ and $\alpha_4 > \beta_4$.

The parameter λ_i can be set by the user to customize the relative cost of moving particular fingers, biasing the algorithm to produce different results. Setting $\lambda_1 < \lambda_2$ would indicate that the index finger should, in general, be preferred to the middle finger, for example.

The exponents m_i simply adjust how fast the cost grows with respect to $||\mathbf{d}||$ and thus control the cost incurred by finger i to play at a fret distant from its natural position.

The cost function for the wrist, f_{wrist}, is similar to that of the fingers and reflects that moving the wrist incurs a cost. Once the cost functions have been designed, we feed them into the cost-minimizing algorithm described below.

5.3. Cost-Minimizing Algorithm

The algorithm described here is a greedy solution to a global minimization problem. Instead of finding the overall minimum cost fingering for an entire piece of music, we find the minimum cost fingering within a moving window of a user definable size.

The position, $\mathbf{c}$, that minimizes:

$$\chi(\mathbf{c}) = f_{wrist}(\mathbf{c}) + \sum_{t=T}^{T+W} resolveFingers(\mathbf{c}, t) \tag{4}$$

tells us where to move the wrist. We solve for this minimum at each timestep T. For our application, we solve a simple discrete minimization problem since we need only test at integral fret positions, a maximum of twenty-four on a typical guitar. Only integral steps are considered because guitar fretting technique requires that the fingertip be placed as close as possible behind the fret for the best sound quality [26]. We thus choose the best location for each fret. The possibility of finger collision may modify this position as described below.

In the above Equation 4, W is the size of the window and *resolveFingers* is the function defined by Algorithm 1, which uses the cost functions as described earlier but with the modification shown here:

$$g_i = \begin{cases} L & \text{if finger } i \text{ is used,} \\ f_i & \text{otherwise} \end{cases} \tag{5}$$

where L is a prohibitively large cost. This modification forces the algorithm to pick a finger that may not be optimal if the optimal finger is already used, as is often found when playing chords.

Algorithm 1 *resolveFingers*($\mathbf{c}$, t)

```
Initialize finger positions
$totalcost \leftarrow 0$
for each string $s$ from high to low do
    $n \leftarrow$ fret played on string $s$ at time $t$
    if $n > 0$ then
        $cost \leftarrow min(g_i(\mathbf{c}))$
        $i \leftarrow argmin(g_i(\mathbf{c}))$
        if $i$ is not used then
            play note at $s$ with finger $i$
            mark $i$ as used
            $totalcost = totalcost + cost$
        end if
    end if
end for
return $totalcost$
```

The first step, initializing the finger positions, is done by placing fingers in line and flexed at about one third of their range. This, although user-definable, seems to be the most natural and ready position for the fingers [21].

Finally, we address two details that affect the believability of the result: wrist posture and finger collisions. These issues are handled after the finger positions are found.

Wrist extension and flexion are calculated using a heuristic that the wrist will extend to play high strings and flex to play low strings. The highest and lowest strings played at each timestep, s_{hi} and s_{lo} are bookmarked when running Algorithm 1. The minimum and maximum extension/flexion angles of the wrist, θ_{min} and θ_{max}, are given by the user and a blend factor, $b = \frac{s_{hi}+s_{lo}}{12}$, between them is computed. Wrist pronation/supination can be handled in a similar way.

To handle finger collision we assume each finger is a circle on the fretboard. Once the fingers have been placed, circle-circle intersection tests are performed and if two circles are found to intersect the finger on the lower string is pushed back along the fret in order to avoid the collision (Figure 5).

Finally, the procedural guitar control algorithm supplies the hand model the appropriate IK blend factors as discussed in section 3.2.

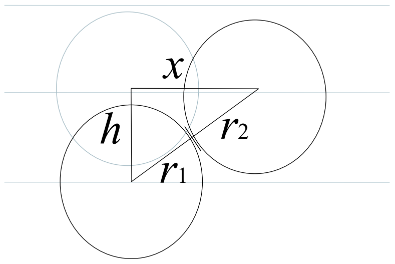

Figure 5: *Finger collision on the fretboard: r_1 and r_2 are the radii of two colliding fingers, h is the distance between strings. We solve for x to determine how far back to move finger 2.*

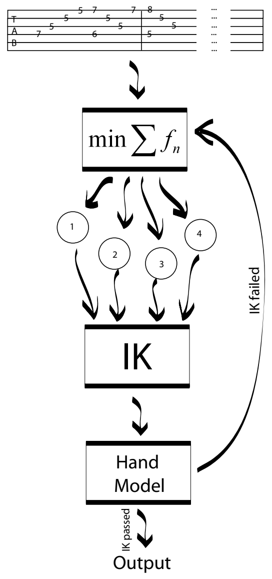

Figure 6: *Detailed system architecture.*

5.4. Controller Failure Feedback Loop

It is possible that the given cost functions and window size cause the IK solver to fail reaching a desired target. A feedback loop is therefore desirable in order to modify the parameters so that the targets are reached. The hand model would report back to the skeletal controller if it was unable to achieve its targets. The controller would then adjust some parameters and try again (see Figure 6. A straightforward approach is to shrink the window size upon failure until only one note or chord is considered (shown in the video on a chromatic scale). We assume the cost functions and hand geometry is designed so that any note can be hit with at least one finger by moving the wrist.

6. Implementation

The tablature parser, procedural fretting algorithm and hand model, as shown in Figure 6, are all implemented in Side Effects Software's Houdini animation platform.

The tablature parser feeds the procedural fretting algorithm, which is implemented as a CHOP (channel-operator) filter. The algorithm controls the motion of the end-effectors or fingertips of the hand based on the input tablature. These feed into the standard Houdini IK solver (a CHOP). The solver produces finger configurations that are constrained by predefined joint constraints. These finger configurations are then fed along with weights that reflect which fingers are currently playing a note, to the hand model (also a CHOP). The hand model filter then corrects the hand configurations to conform to the captured hand motion data. The hand model can be bypassed with a switch, to compare the results of the hand model with that of the unmodified IK solution.

Part of a musician's training is to minimize the sympathetic motion between fingers [21] allowing maximum dexterity and independence of the fingers. The modularity of our architecture allows one to monitor progress by seeing the magnitude and nature of the sympathetic motion of a students hand. Such a separation also allows the hand model to be used independently of the procedural guitar player.

7. Results

Figures 8, 9, 10 and the accompanying video show our results. We describe a few test cases below.

7.1. Hand Model

Figure 8 shows the difference between using a general IK solution and using the augmented hand model. Both illustrations are the results of using the same reaching targets. Figure 8 compares real hand data with a traditional IK solution and the IK solution augmented with the hand model. The subject was asked to flex only his index finger. The IK solution and the hand model use the same target end-effector positions. The IK solution ignores sympathetic motion between fingers producing unrealistic results. The hand model augmented solution produces results close to the real hand data, falling a little short due to the non-zero weights of the fully extended fingers that are output by the IK solution.

7.2. C-Major Scale

Figure 9 shows eight frames from an animation playing a C-Major scale on a guitar. Note that the first note is played using the middle finger in order to avoid having to move the wrist to play the entire scale. The accompanying video shows how adding in the sympathetic motion from the hand model results in a less mechanical appearance of the hand.

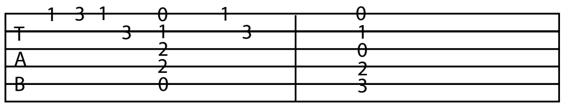

Figure 7: *Passage showcasing fingering for note continuity.*

7.3. Chords

Unlike a musical scale, which can be played naively with one finger, chords require multiple notes to be played simultaneously. Figure 10 shows a chord progression C-G-F#-G. Note that the algorithm fingers the G chord following the C differently from the way it fingers the same chord after the F#. The reason for this is that the minimal movement needed to change from a C to a G requires the use of the little finger. Following the F# there is sufficient motion needed to change to a G, that the general finger preference favors the index, middle and ring finger over the little finger. Context dependent fingerings like this are commonly used by guitarists. Also noteworthy is the index finger forming a bar at the second fret to play the F# chord. The chord requires all 6 strings to be played with 4 fingers. The only way this is possible is for fingers to reach multiple strings. Given the pervasiveness of bar chords to guitar playing we implemented them as a special case. The algorithm attempts to play more than four concurrent notes by using the index finger at the lowest given fret and using the other three fingers to resolve the remaining notes.

7.4. Complex Passage

The passage in Figure 7 shows a mix of single notes and chords. The C chord at the end is fingered with the index, middle and ring fingers. A novice is likely to use the ring finger to play the penultimate note on the third fret. As the ring finger is moved to play the subsequent chord, an expert would play the note with the unused little finger allowing the note to linger for a better sound [21]. Our procedural algorithm captures this with a cost function for the movement of fingers across strings.

8. Conclusion and Future Work

This paper presents an exploration in capturing the intricate nature in which the fingers of the hand work together. Our hand model is a simple interactive module that enhances the realism of arbitrary hand animations. It also provides insight into the complexity of joint interdependencies. We set up the groundwork for a more complete anatomically based hand model that can be fitted to and validated by human motion data. The consideration of temporal coherence is important to remove problems that may result in jerkiness of the animation, if the algorithm chooses particularly differing postures between frames. Our requirement that postures be chosen based on geometric proximity does not guarantee temporal coherence and is an important issue to address in the future.

Other improvements to the hand model include the addition of joint velocity and wrist and arm posture to the configuration space. Joint velocities would allow better handling of dexterity constraints and provide better motion dynamics as the fingers move from note to note.

The procedural algorithm to the fretting-hand problem, though described in the context of guitar generalizes to similar tasks such as typing and playing other instruments. As a music education tool the algorithm allows a student to control and separate various elements of playing style and explore alternate tunings. While most important aspects of the fretting hand are captured, elements of playing such as hammers, pull-offs and bends are left as future work. The "Stairway to Heaven" tablature in Figure 6 and the video further show the importance of generalizing the reaching area of a finger beyond the fingertip, where the arpeggiated notes on the 5th fret in the opening could all have been played with a barred index finger. As an animator's tool, this work greatly simplifies the control of realistic animation of the human hand.

Acknowledgements

We would like to thank: Eugene Fiume, Anne Agur, Nancy McKee, Jeffrey McFadden, Richard Zemel, Aaron Hertzmann, Kyros Kutulakos, Ravin Balakrishnan, Brian Li, Northern Digial Inc., Janet Fraser, Kim Davidson, Paul Salvini, Cristin Barghiel, Mark Elendt, Jeff Lait, Mark Alexander, Edward Lam, Lucas Moore, Ondrej Kos, everyone at Side Effects Software and the Dynamic Graphics Project lab.

References

1. A. M. R. Agur and M. J. Lee. *Grant's Atlas of Anatomy*. Lippincott Williams and Wilkins, 10th Edition, 1999.
2. O. Arikan and D. A. Forsyth. Interactive Motion Generation from Examples. *ACM Computer Graphics (Proc. of SIGGRAPH 2002)*, 483–490, 2002.
3. S. Arya, D. Mount, N. Netanyahu, R. Silverman and Y. Wu. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*, **45**, 891–923, 1998.
4. R. Bargar. Music for Unprepared Piano. *ACM SIGGRAPH 98 Electronic Theatre*, ACM SIGGRAPH Computer Animation Festival, 1998.
5. E. Barr and M. Barbe. Pathophysiological tissue changes associated with repetetive movement: A review of the evidence. *Physical Therapy*, **82**(2):173–187, 2002.
6. B. Bodenheimer, C. Rose, S. Rosenthal and J. Pella. The Process of Motion Capture: Dealing with the Data. *Computer Animation and Simulation, Eurographics Animation Workshop*, 3–18, 1997.
7. A. P. Georgopoulos, G. Pellizzer, A. V. and M. H. Schieber. Neural Coding of Finger and Wrist Movements. *Journal of Computational Neuroscience*, **6**, 279–288, 1999.
8. M. Gleicher. Retargetting Motion to New Characters. *ACM Computer Graphics (Proc. of SIGGRAPH 98)*, 33–42, 1998.
9. C. Häger-Ross and M. H. Schieber. Quantifying the Independence of Hand Finger Movements: Comparisons of Digits, Hands, and Movement Frequencies. *The Journal of Neuroscience*, **20**(22)8542–8550, 2000.
10. R. Hänninen. LibR – An Object Oriented Software Library for Real-Time Virtual Reality. Ph. D. thesis, Helsinki University of Technology, 1999.
11. P. G. Kry, D. L. James and D. K. Pai. EigenSkin: Real Time Large Deformation Character Skinning in Hardware. *Proc. of ACM SIGGRAPH 2002 Symposium on Computer Animation*, 2002.
12. Y. Koga, K. Kondo, J. Kuffner and J. C. Latombe. Planning Motions with Intentions. *ACM Computer Graphics (Proc. of SIGGRAPH 94)*, 395–408, 1994.
13. L. Kovar, M. Gleicher and F. Pighin. Motion Graphs. *ACM Computer Graphics (Proc. of SIGGRAPH 2002)*, 473–482, 2002.
14. J. Lee, J. Chai, P. S. A. Reitsma, J. K. Hodgins, N. S. Pollard. Interactive Control of Avatars Animated with Human Motion Data. *ACM Computer Graphics (Proc. of SIGGRAPH 2002)*, 491–500, 2002.
15. J. Lee and T. Kunii. Model-based Analysis of Hand Posture. *IEEE Computer Graphics and Applications*, **15**(5), 77–86, 1995.
16. Y. Li, T. Wang and H. Y. Shum. Motion Texture: A Two-Level Statistical Model for Character Motion Synthesis. *ACM Computer Graphics (Proc. of SIGGRAPH 2002)*, 465–472, 2002.
17. C. K. Liu and Z. Popović. Synthesis of Complex Dynamic Character Motion from Simple Animations. *ACM Computer Graphics (Proc. of SIGGRAPH 2002)*, 408–416, 2002.
18. W. Lytle. Pipe Dream. *ACM SIGGRAPH 2001 Electronic Theatre*, ACM SIGGRAPH Computer Animation Festival, 2001.
19. W. Lytle. More Bells and Whistles. *ACM SIGGRAPH 90 Electronic Theatre*, ACM SIGGRAPH Computer Animation Festival, 1990.

20. C. L. MacKenzie and T. Iberall. *The Grasping Hand*. North-Holland (Elsevier Science B.V.), 1994.

21. J. McFadden. Discussion. *www.jeffreymcfadden.com*, 2002.

22. L. Moccozet. *Hand Modelling and Animation for Virtual Humans*. Ph. D. thesis, University of Geneva, 1996.

23. C. Nölker and H. Ritter. GREFIT: Visual Recognition of Hand Postures. *Proc. of the International Gesture Workshop*, 61–72, 1999.

24. N. S. Pollard and J. K. Hodgins. Generalizing Demonstrated Manipulation Tasks. *Workshop on the Algorithmic Foundations of Robotics (WAFR '02)*, Nice, France, 2002.

25. K. Pullen and C. Bregler. Motion Capture Assisted Animation: Texturing and Synthesis. ACM Computer Graphics (Proc. of SIGGRAPH 91), 501–508, 2002.

26. H. Quine. *Guitar Technique*. Oxford University Press, New York, 1990.

27. H. Rijpkema and M. Girard. Computer Animation of Knowledge-Based Human Grasping. *ACM Computer Graphics (Proc. of SIGGRAPH 91)*, 339–347, 1991.

28. C. F. Rose, P. J. Sloan, M. F. Cohen. Artist-Directed Inverse-Kinematics Using Radial Basis Function Inerpolation. *Eurographics*, **20**(3), 2001.

29. S. I. Sayegh. Fingering for String Instruments with the Optimum Path Paradigm. *Computer Music Journal* **13**(3):76–83, 1989.

30. M. H. Schieber. Individuated Finger Movements: Rejecting the Labeled-Line Hypothesis. Editors A. M. Wing, et al. *Hand and Brain: The Neurophysiology and Psychology of Hand Movements*. San Diego: Academic Press, Inc. 1996.

31. Y. Wu and T. Huang. Capturing Articulated Human Hand Motion: A Divide-and-Conquer Approach. *Proc. of the International Conference on Computer Vision*, 1999.

32. K. Yamane and Y. Nakamura. O(N) Forward Dynamics Computation of Open Kinematic Chains Based on the Principal of Virtual Work. *Proc. of the 2001 IEEE International Conference on Robotics and Automation*, 2824–2831, 2001.

33. J. Zhao and N. I. Badler. Inverse Kinematics Positioning Using Non-Linear Programming for Highly Articulated Figures. *ACM Transaction on Graphics*, **13**(4):313–336, 1994.

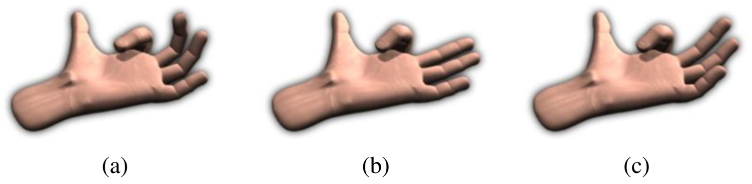

Figure 8: *(a) Joint angle data of a person asked to flex his index finger; (b) the result of an IK only solution with the same target end-effector for the index; (c) the result of our hand-model with the same target end-effector for the index.*

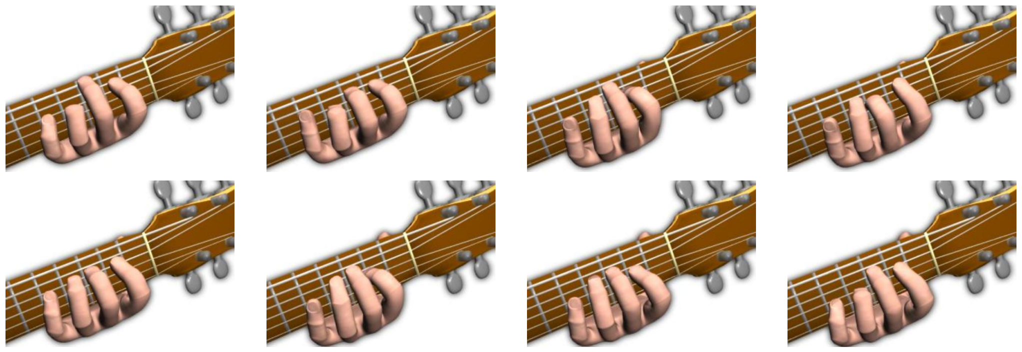

Figure 9: *Result of the algorithm running on a C-Major scale using the hand model.*

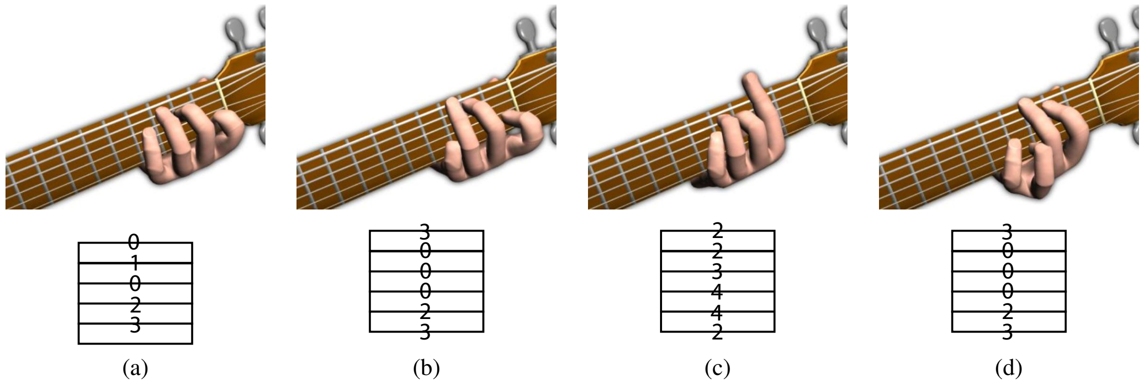

Figure 10: *Here a progression of chords is played: C (a) – G (b) – F# (c) – G (d). The G is played in two different configurations based on the preceding chord in order to minimize the movement of the hand.*