COMPUTER ANIMATION AND VIRTUAL WORLDS
Comp. Anim. Virtual Worlds 2013; **24**:445–456
Published online 21 September 2012 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cav.1477

RESEARCH ARTICLE

A system for automatic animation of piano performances†

Yuanfeng Zhu*, Ajay Sundar Ramakrishnan, Bernd Hamann and Michael Neff

Department of Computer Science, University of California, Davis, 1 Shields Ave., Davis, CA, USA, 95616

ABSTRACT

Playing the piano requires one to precisely position one's hand in order to strike particular combinations of keys at specific moments in time. This paper presents the first system for automatically generating three-dimensional animations of piano performance, given an input midi music file. A graph theory-based motion planning method is used to decide which set of fingers should strike the piano keys for each chord. As the progression of the music is anticipated, the positions of unused fingers are calculated to make possible efficient fingering of future notes. Initial key poses of the hands, including those for complex piano techniques such as crossovers and arpeggio, are determined on the basis of the finger positions and piano theory. An optimization method is used to refine these poses, producing a natural and minimal energy pose sequence. Motion transitions between poses are generated using a combination of sampled piano playing motion and music features, allowing the system to support different playing styles. Our approach is validated through direct comparison with actual piano playing and simulation of a complete music piece requiring various playing skills. Extensions of our system are discussed. Copyright © 2012 John Wiley & Sons, Ltd.

KEYWORDS

piano animation; fingering generation; optimization method

***Correspondence**

Yuanfeng Zhu, Department of Computer Science, University of California, Davis, 1 Shields Ave., Davis, CA, USA, 95616.
E-mail: yuazhu@ucdavis.edu

1. INTRODUCTION

The piano is a complex musical instrument to play as one performs musical theme and chords simultaneously. It is also a very complex task to correctly animate piano playing, given the tight constraints on the timing of the motion, necessary to generate the correct notes, and the high number of degrees of freedom (DOFs) of the hand and fingers. While specialized, piano playing is not an uncommon activity and will occur in movies, games, and virtual environments. Furthermore, in the field of piano tuition, given any piece of music, amateur players find it difficult to determine the most effective fingering and to render it exactly the way it is supposed to be performed. Therefore, a system capable of generating high quality piano animation may also be useful in piano tutoring.

Piano playing is a challenging task for multiple reasons. First, piano animation requires high dimensional poses for both hands involving many joints (16 joints with 27 DOFs for each hand in our hand model). Second, exact musical timing constraints require precisely aligning the hand motion with music in order to strike, hold, and release the instructed piano keys. Third, occlusions often occur in playing, such as when the hand crosses over the thumb, making optical motion capture of limited use. Even if motion capture were available, in order to play a new piece of music, the motion would need to be adapted to the volume, velocity, pitch, note structure, and timing of that piece—problems that are still unexplored to the best of our knowledge. We instead develop a kinematic system that is very flexible in the range of music it can play, including unanticipated pieces, and uses limited motion capture data to improve the output quality.

Our system automatically generates animations of piano performances as follows:

Given any piano midi file as input, a novel algorithm combining geometry constraints, graph theory, and piano theory is used to determine the most efficient piano fingering to play the whole piece. For this, a rule-based method based on geometric constraints for the fingertips is first used to find the most comfortable fingering choices for playing each chord (the "instructed fingers"). Then, an optimal motion planning method based on graph theory is used to determine the sequence of fingering choices for

†Supporting information may be found in the online version of this article.

Copyright © 2012 John Wiley & Sons, Ltd.

the sequence of chords that make up the musical piece, expending minimum energy. Finally, geometric constraints are used to determine where the unused fingers for each chord (the "non-instructed fingers") should be placed in order to most efficiently play subsequent notes such that all the fingers are placed in comfortable and natural positions. Then, using geometric constraints on the fingers, the wrist, and the keyboard contact surface, the algorithm determines initial natural hand poses for each chord.

Given the generated set of poses, a novel optimization-based method is proposed to generate detailed motion curves under the requirements of different performance styles: (1) playing scales, where a special case of notes having complex motion sequences such as finger crossover is handled; (2) playing chords, which is more difficult to handle because several fingers are used to play one chord and the wrist must have reasonable translation and rotation to maintain the naturalness of the pose while the instructed fingers are pressing piano keys; and (3) the special case of arpeggio, in which notes in a chord are played in sequence rather than simultaneously. We consider additional important factors to enhance the realism of the animation, such as how the music volume influences the motion and how instructed fingers influence non-instructed fingers, and add wrist compensation to simulate the reaction of the piano key strike.

Our system is implemented as a plug-in component to Maya, following the procedure outlined in Figure 1. After reviewing relevant previous work, the remainder of the paper explains these steps in detail and presents the results obtained with the system. The applications of our work is discussed in the last section.

2. RELATED WORK

There are two problems that we deal with in our paper: fingering generation and hand animation. Many researchers have worked on the problem of generating the right fingering for various musical instruments. The fingering problem for string instruments has been extensively surveyed by [1]. Heijink and Meulenbroek [2] use statistical approaches to analyze the factors that influence left-hand movements in classical guitar playing and demonstrate that guitar players usually keep their finger joints in the middle of their range and control the sound variation by regulating the timing and placement of the left fingers. Viana and Junior [3], Lin and Liu [4], and Parncutt *et al.* [5] use rule-based expert systems to generate piano fingering, but there are cases of conflicts between applicable rules and cases where no rules were applicable. Yonebayashi *et al.* [6] use hidden Markov models for piano fingering generation but cannot generate fingering for chords. Radisavljevic and Driessen [7] propose a path difference learning method that evaluates cost function weights to adapt guitar fingering generation for a given guitar playing style, but the generated fingering does not perform well when not enough training data are available. Tuohy and Potter [8] use a genetic algorithm to find playable guitar fingering, but it usually only generates playable fingering rather than fingering like that published in guitar books, which means the generated fingering might not be elegant, smooth, and/or energy-saving to play. Tuohy [9] employs genetic algorithms and artificial neural networks for music arrangement and tablature generation for guitar. This namely takes a music piece originally written for another instrument as an input, and then the proposed method translates this music to a new version that guitar can play. Rather than finding optimal fingering solutions for a certain instrument, Handelman *et al.* [2] report their development of an interactive program that models various performance possibilities for the same music being player by violin, viola, or cello. Hart *et al.* [10], Kasimi *et al.* [11], Radicioni *et al.* [12], Radicioni and Lombardo [13], and Radisavljevic and Driessen [7] use a greedy algorithm approach of traversing a trellis graph to find optimal fingering for piano or guitar, on which we base our approach.

There has been much work on human hand modeling. Yasumuro *et al.* [14] used an anatomical approach for building a hand model. Lee and Kroemer [15] proposed a kinematic model of the human hand. Pollard and Zordan [16] proposed an effective physics-based approach

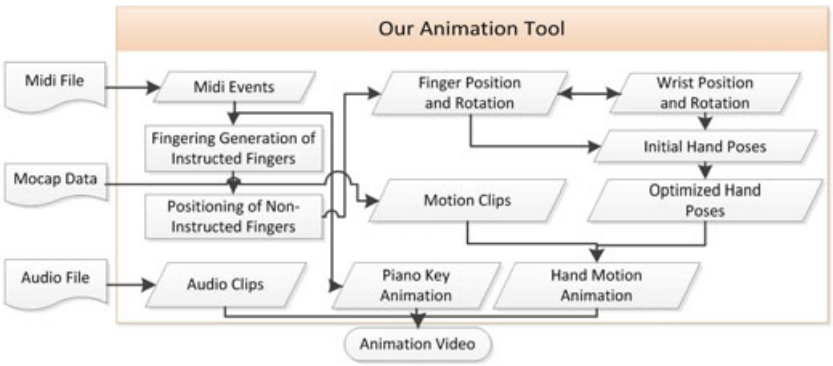

Figure 1. Flowchart of system implementation.

DOI: 10.1002/cav

for grasping control and hand interaction with a small set of parameters. Much recent work on hand animation has focused on physical models but has not considered complex finger tasks with exact timing constrains. By contrast, our work uses kinematic models but addresses the challenge of calculating a coordinated, rapidly changing set of hand poses to interact with a keyboard and satisfy musical timing constraints.

Research has also been carried out on animating the motions associated with playing a musical instrument. The Handrix system proposed in [17] generates the motion of the fretting hand when playing a guitar using a procedural algorithm. Kim *et al.* [18] presents an approach to control a violinist's hand movement by using neural networks. Viana and Junior [3] also implemented a 2D piano key animation given any input music.

We improve the existing work in the following two directions: (1) No previous methods consider the position of non-instructed fingers, which is important for generating comfortable non-instructed finger positions while minimizing hand motion for the whole music piece. A novel fingering approach accomplishing such a goal is proposed on the basis of hand motion distance and ease for the whole music piece and also on the basis of the principle of piano theory that the best fingering is the one that involves least effort for the player. (2) None of previous systems are capable of automatically generating three-dimensional piano playing animation, while our approach automatically generates accurate hand motion at all points of time even for complex piano performance such as finger crossover or arpeggio skill.

3. FINGERING DETERMINATION

We need to determine the best possible fingering, out of the many possibilities, for the whole musical piece, such that it satisfies the principle of piano theory that a player will play most effectively by making choices that relax his body and save energy. For each chord, we determine which finger should press on the corresponding key for each note. Then, we calculate the best positions for the fingers that do not press on a key (non-instructed fingers) to make it easier to play the succeeding notes. Note that the fingers are numbered 1 to 5 with 1 denoting the thumb and 5 denoting the little finger.

3.1. Placement of Instructed Fingers

Instructed fingers are those that press on a key to play a given note while playing a chord. The first step in our approach is to determine the position of instructed fingers to play every note of every chord for a given piece of music.

3.1.1. Representation as a Trellis Graph.

We first represent the instructed fingering problem as a trellis graph (Figure 2) where each node in the same time-slice represents a finger combination to play that chord and the edges connecting them correspond to the hand motion between neighboring chords.

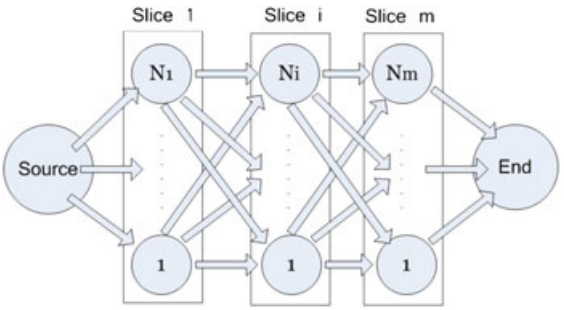

Figure 2. A trellis graph where consecutive time-slices correspond to consecutive chords in the piece. The weighted nodes represent the cost of hand poses for a fingering choice, and weighted edges represent the cost of hand motion between the hand poses.

Depending on the type of chord and the number of notes in it, we can generate different fingering choices for it: a chord with one or four notes has five fingering choices, a chord with two or three notes has 10 choices, and so on.

3.1.2. Determining Cost of Hand Pose.

The hand pose cost, corresponding to each node weight, quantifies the effort required to pose the fingers in a particular configuration to play a given chord, and these fingers are called instructed fingers in our system.

$$C_{N_i} = \sum C(a,b) \tag{1}$$

$C(a,b)$ is the energy cost for any adjacent instructed fingers a and b for maintaining a pose. The function value is obtained considering the distance and ease of two neighboring fingers pressing on the piano keys. For example, we expect $C(1,4)$ to be minimum when $d(a,b) \approx$ 3–5, where $d(a,b)$ denotes the distance between the two fingers pressing the piano keys in units proportional to the breadth of a white key in the piano, because the thumb and ring fingers are three fingers (corresponds to three keys on the piano) apart by default, and so this corresponds to the most relaxed arrangement. The cost value $C(a,b)$, which is a segmented function evaluated on the basis of the performance experience of piano players, increases for larger or smaller separations, as illustrated in Figure 3.

3.1.3. Determining Cost of Hand Motion.

The hand motion cost from current fingering choice i to the next j, corresponding to the edge $E_{i,j}$ connecting from node i N_i to node j N_j, arises from three individual costs:

$$C_{E_{i,j}} = C_f + C_c + C_r \tag{2}$$

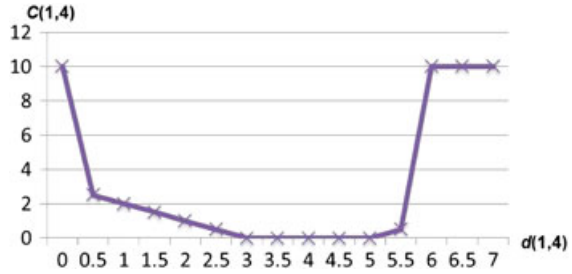

Figure 3. Hand pose cost values for poses corresponding to different separation of the thumb and ring fingers.

C_f, which is a constant set based on the piano player's experience, is used to penalize the reuse of a finger in the subsequent chord if it plays a different note than in the current chord. It is easier for a non-instructed finger from the first chord to play a new note in the second chord, as we do not have to worry about re-positioning the fingers after playing the first chord. This cost encourages consecutive chords to be played with different fingers.

C_c represents the extra energy required for a finger to cross over other fingers. While playing a melody, the best fingering may involve the fingers crossing over the thumb to play the next note or the thumb passing under the fingers, but this should be avoided when unnecessary. This value is linearly proportional to the number of fingers between the moving finger and the finger being crossed-over.

C_r penalizes the extra local movement of the fingers required to strike from one note to another on the basis of the fact that larger note changes for each finger will cause the wrist to move more. This value is linearly proportional to the sum of the distance all fingers move from the current piano keys to the next ones.

3.1.4. Finding Shortest Paths by Dijkstra's Algorithm.

Now that we have modeled the problem of fingering choice on a trellis graph and computed the costs of all the nodes and all the edges connecting them, we use Dijkstra's algorithm to find the shortest path. We cannot use Dijkstra's algorithm directly to compute the shortest path, because the nodes have non-zero costs. Therefore, we update the graph such that the node costs are also incorporated into the edge costs as follows:

(1) Update the edge cost $C_{E_{i,j}}$ of $E_{i,j}$ as

$$C'_{E_{i,j}} = C_{E_{i,j}} + (C_{N_i} + C_{N_j})/2, \forall \text{edges}(i,j) \tag{3}$$

(2) Update the node weights to zero as

$$C_{N_i} = 0, \forall i \tag{4}$$

Now that all nodes have zero weights, we can use Dijkstra's algorithm to obtain the shortest path in the updated trellis graph. Each node selected at each level of the graph gives the fingering choice of instructed fingers for that chord.

3.2. Placement of Non-instructed Fingers Depending on Future Notes

After calculating the fingering for instructed fingers, we need to determine how to pre-position the non-instructed fingers to minimize the overall effort, which is required by the piano performance for smooth hand motion. This has not been addressed by any previous work as it is essential only while generating three-dimensional animation output as our system does. The position for non-instructed fingers for the current chord are chosen so that they are easier to re-position to play the next chord where they will actually be used and therefore to minimize the energy cost of hand motion. The algorithm operates as follows:

(1) We wish to determine the position of the non-instructed fingers for chord i and assume that the instructed fingers for all chords have already been positioned. We consider the next k ($1 \leq k \leq 4$) chords in positioning the non-instructed fingers at i because of a maximum of four non-instructed fingers to pre-position. Note that the ability to pre-read the notes varies on the basis of the level of music, player, and familiarity with the music piece, and our method provides the solution assuming the player is familiar with the music and can generate most reasonable fingering for the current non-instructed fingers.
(2) If j is an instructed finger in chord $i+k$, then j has to be positioned in chord i, so that it does not have to move much when we play chord $i+k$.
(3) Let the instructed finger in chord i closest to the jth finger be a. There could be two such fingers, one on either side. For all adjacent fingers that exist, the finger j's position should satisfy that the distance $d(j,a)$ between jth and an adjacent finger should be in the range of comfort playing.
(4) When the preceding conditions are satisfied, the position of finger j has been determined for chord i on the basis of chord $i+k$. Do this until all the positions for non-instructed fingers for a chord have been determined.

When a non-instructed finger is the thumb, ring, or little finger, we update the finger position by -0.5 along the Z axis (move it to the left). In case the new position is occupied by another finger, we increase it instead by 1 (move it to the right). When the non-instructed finger is the index or middle finger, we update the finger position by 0.5 (move it to the right). When this new position is occupied by another finger, we decrease it by 1 (move it to the left). We prefer to move some fingers to the left first and some to the right first in the aim of generating a hand pose where fingers have minimum influence of each other as indicated

DOI: 10.1002/cav

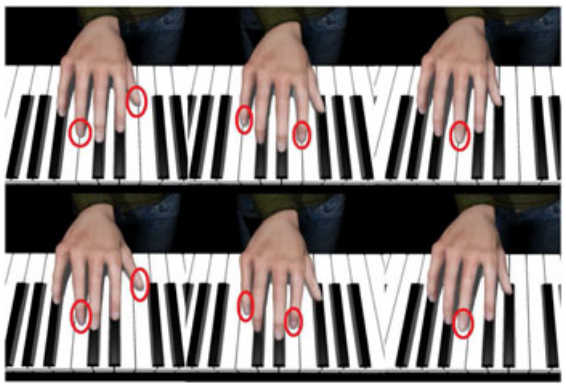

Figure 4. Repositioning of non-instructed fingers, which are indicated by the red circles around them.

by [19]. Row 1 of Figure 4 shows that the default positioning of the non-instructed fingers is not always meaningful. Row 2 shows the hand pose after the correction has been carried out.

4. FINGER AND HAND POSE CALCULATION FOR CHORD

For any given chord, there are four steps in simulating the hand motion as Figure 5: (1) calculate the hand pose, which includes the position of the fingertips and the position and orientation of the wrist; (2) press down the piano keys; (3) hold the piano keys; and (4) release keys back to the original position. Because the last three steps can be simulated using methods similar to the first one, we just focus on the first step. Also, an algorithm to handle the complex performance such as finger crossovers and arpeggio is described.

Before further discussion of this section, we first describe our hand model, as shown in Figure 6. The five fingers in the hand model are labeled from Finger 1 for the thumb to 5 for the little finger. There are 16 total joints with 27 DOFs in this hand model: 6 DOFs for wrist joint labeled with black circle, 1 DOF for extension and flexion of each distal interphalangeal (DIP) joint and proximal interphalangeal (PIP) joint, 2 DOFs for extension and flexion, adduction, and abduction of each metacarpophalangeal (MCP) joint for Fingers 2 to 5, and 3 DOFs for thumb's finger base rotation. Each finger is assigned an IK (Inverse kinematics) handler starting from the finger base and going to the fingertip. IK is used when the fingers strike and release the keys, and FK (Forward kinematics) is used for the in-between movement for different chords, with a blend used to smoothly switch between the FK and the IK.

Some important notation is defined as follows: B_i refers to the base of finger i and T_i refers to the fingertip of finger i. P denotes joint position, and Θ denotes the joint orientation. World axes and wrist local axes are defined as shown in Figure 7.

4.1. Initiate Hand Pose

The generated fingering of instructed and non-instructed fingers in Section 3 is used to decide the position of the fingertips along the axis Z, which are then used as the basic parameters to evaluate other position components of fingertips and the position and orientation of the wrist. The finger base position P_B is decided by the wrist position P_W and orientation Θ_W because the finger bases are fixed on the palm, but the fingertip positions P_T, wrist position P_W, and orientation Θ_W have to be calculated.

4.1.1. Initiate Finger Positions.

$P_{T_i}(t)_z$ represents which key is pressed and is hence determined by the fingering method. $P_{T_i}(t)_x$ is determined by wrist position and piano key range occupied by fingers. $P_{T_i}(t)_y$ is the height of a black or white key being pressed.

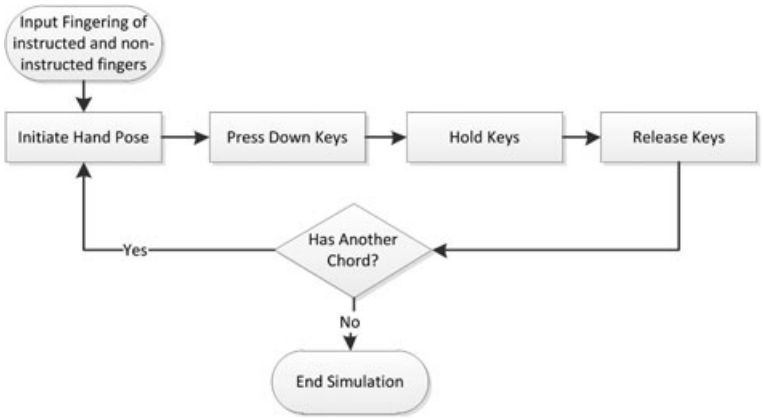

Figure 5. Data flow of simulation of piano performance.

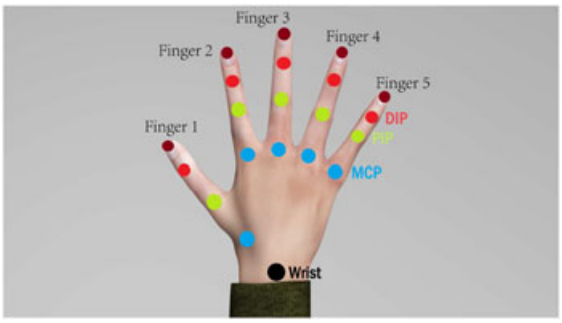

Figure 6. Important joints in our hand model.

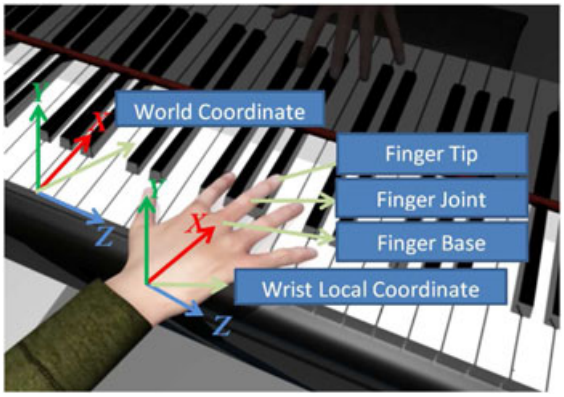

Figure 7. The axes in our model.

4.1.2. Initiate Wrist Position.

The wrist position is determined on the basis of the fact that when the fingers are more spread, the wrist will move forward along the X axis and has a lower position value along the Y axis; along the Z axis, the wrist moves relatively closer to the little finger and farther from the thumb. Therefore, we compute the wrist position as follows:

$P_W(t)_z$ is the weighted sum of the Z components of all fingertip positions. The thumb and the little finger generally have much more influence in determining the wrist position along the Z axis, and therefore, they have much larger weight than the other fingers.

$P_W(t)_x$ is determined by the instructed fingers' relative positions along the X axis in a standard pose, the influence of the finger (prioritized 1, 5, 2, 4, and 3 in decreasing order), and the allowable contact range on the keys pressed by these fingers. For example, if the instructed thumb has to move to a black key from a white key, then this will fully dictate the wrist movement, as the wrist will have to move less to accommodate the position of any other finger.

$P_W(t)_y$ is determined in a similar way as $P_W(t)_x$.

4.1.3. Initiate Wrist Orientation.

The wrist orientation along the Y axis, $\Theta_W(t)_y$, is computed as

$$\Theta_W(t)_y = \sum_{i=1}^{5} w_i * \Theta(P_{T_i}, P_W, t)_y \qquad (5)$$

where $\Theta(P_{T_i}, P_W, t)_y$ is the orientation around the Y axis of the ray from the wrist to T_i at time t. w_i for five fingers are pre-computed weights, which reflect the dependence of rotation of Joint i and the wrist.

In order to obtain five w_i, we set up an equation set that consists of five equations based on Equation (5) for five different chords. For each equation, take w_i for $i = 1, \ldots, 5$ as five unknown variables, and obtain values of $\Theta(P_{T_i}, P_W, t)_y$ for $i = 1, \ldots, 5$ on the basis of the motion capture data with information of wrist and fingertip position for each chord. Solve the equation set to obtain the five weights.

The result of this process is that w_1 has the smallest weight while w_3 has the largest weight, which means the wrist orientation mainly depends on the position of non-thumb fingers on the palm and the middle finger position on the X and Z axes, because hand poses usually satisfy the condition that the middle fingertip, finger base, and wrist are almost colinear.

We calculate $\Theta_W(t)_z$ as follows:

$$\Theta_W(t)_z = \Theta_{W\_\max_z} - \Theta * w_D(t) \qquad (6)$$

This equation implies that the wider the key range the fingers press on, the lower the wrist orientation around Z. This happens because we obtain a wider range to spread the fingers when the wrist is closer to the keyboard surface than when it is far away. $\Theta_{W\_\max_z}$ is the largest angle of the wrist along the Z axis (also the initial orientation along the Z axis for the standard pose). Θ controls the largest wrist orientation along axis Z; $w_D(t)$ is the rotation weight influenced by the five finger distributions at time t, and a larger finger extension will have larger $w_D(t)$.

$\Theta_W(t)_x$ is computed as

$$\Theta_W(t)_x = \Phi - \arctan \frac{P_{B(\text{ring})}(t)_y - P_{B(\text{index})}(t)_y}{P_{B(\text{ring})}(t)_z - P_{B(\text{index})}(t)_z} \qquad (7)$$

This equation keeps the hand nearly parallel with the piano face. Because the index and ring finger bases are fixed on the palm, they can be used to define a line in three-dimensional space, and therefore, the projection to the plane perpendicular to the X axis can be used to evaluate the orientation of the wrist around axis X. The parameter Φ is evaluated on the basis of the standard pose pressing on five neighboring keys with distance of 4 and distance of 7, respectively, from the little finger to thumb.

4.2. Crossover between Thumb and Other Fingers

Crossover is common while various note sequences are played. This case is handled separately using the algorithm outlined in Figure 8, which illustrates the case where the thumb crosses under finger j.

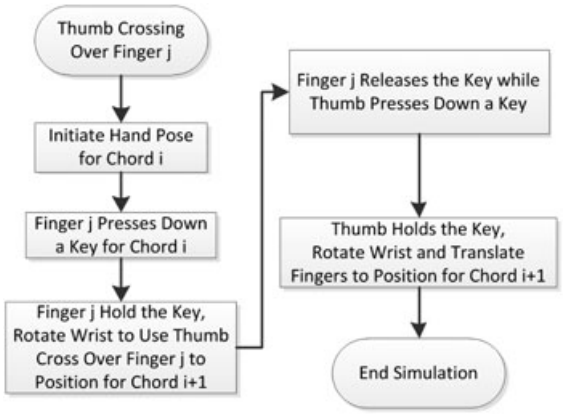

Figure 8. Algorithm to handle finger crossover.

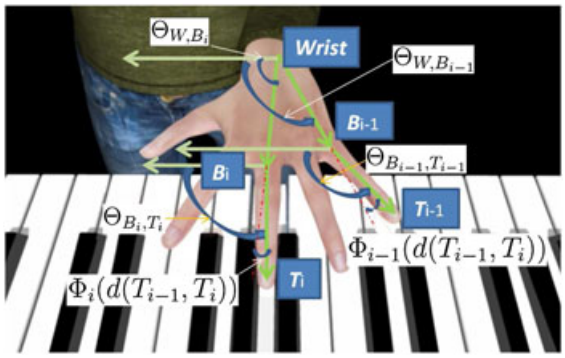

Figure 9. Arpeggio skill.

After a finger j presses down the corresponding key for chord i, the wrist is translated by a distance, which is evaluated on the basis of the corresponding key postures extracted from motion capture data depending on what finger it crosses over: index, middle, or ring. After the translation, the wrist is rotated such that

$$\Theta(P_W, P_{B_j}, t)_y = \arctan \frac{P_{B_j}(t)_x - P_W(t)_x}{P_{B_j}(t)_z - P_W(t)_z} \tag{8}$$

$$\Theta(P_W, P_{T_j}, t)_y = \arctan \frac{P_{T_j}(t)_x - P_W(t)_x}{P_{T_j}(t)_z - P_W(t)_z} \tag{9}$$

$$\Theta(P_W, P_{B_j}, t)_y - \Theta(P_W, P_{T_j}, t)_y \in [\Theta_{\min}(j), \Theta_{\max}(j)] \tag{10}$$

The joint angles of all fingers other than the thumb are kept the same, and the positions are translated by the wrist rotation. After j releases the key and the thumb presses down its key, the wrist and fingertips are translated to the position for Chord $i+1$.

The algorithm for fingers crossing over the thumb is similar to case of the thumb crossing under other fingers.

4.3. Arpeggio Skill

In arpeggios, notes in a chord are played in sequence rather than simultaneously. Sometimes the successive notes might be farther apart from each other, such as playing a typical chord with notes C3-G3-C4-E4 using left fingering 5-3-2-1, which is more difficult to handle than playing common chords because more complex wrist motion is needed to make sure the instructed finger can reach the required piano note in the arpeggio in time while keeping hand pose natural. The method in Section 4.1.2 is first used to compute the initial wrist position for the common chord, which has the same notes as the required arpeggio, and then we shift the wrist position to satisfy the following geometry constraints:

$$\Theta_{B_{i-1},T_{i-1}} - \Phi_{i-1}(d(T_{i-1},T_i)) = \Theta_{B_i,T_i} - \Phi_i(d(T_{i-1},T_i)) + \Theta_{W,B_{i-1}} - \Theta_{W,B_i} \tag{11}$$

Just as shown in Figure 9, T_i is the fingertip used to strike note i, and B_i is the corresponding finger base; Θ_{B_i,T_i} is the orientation along the Y axis from finger base i to fingertip i; Θ_{W,B_i} is the orientation along the Y axis of the vector from wrist to note i's finger base; $\Phi_i(d(T_{i-1},T_i))$ is the rotation offset used for the finger to strike note i on the basis of the distance between two neighboring instructed fingertips. This constraint is to determine the wrist position so that the two instructed fingertips can be positioned on the required neighboring piano keys for smooth hand motion from note $i-1$ to note i.

5. OPTIMIZATION OF KEY HAND POSES

For most relaxed playing, piano theory requires the player to keep natural and precise poses while decreasing extra motion. Therefore, a novel optimization method with geometry constraints is proposed to smooth the hand motion between the hand key poses for each chord. The following is the objective function to find the wrist pose sequence that minimizes the overall motion cost determined by the wrist translation and rotation for all the given n chords:

$$\min_{C_i}\left(\sum_{i=2}^{n} C_i\right) = \min_{P_i} \times \left(\sum_{i=2}^{n}(||P_i - P_{i-1}|| + w * ||\Theta_i - \Theta_{i-1}||)\right) \tag{12}$$

where C_i is energy cost for Chord i, w is the weight between translation and rotation components, and $||P_i - P_{i-1}||$ and $||\Theta_i - \Theta_{i-1}||$ are the wrist translation and rotation between two neighboring chords, respectively. Sequential quadratic programming is used for optimization solution of the minimum motion cost, considering the

following four constraints for each chord.

$$c_1 : \Theta_{J_{i,j}} \in [\Theta_{J_{i,j}\_\min}, \Theta_{J_{i,j}\_\max}] \quad (13)$$

$$c_2 : d(T_i, B_i) \in [d(T_i, B_i)_{\min}, d(T_i, B_i)_{\max}] \quad (14)$$

$$c_3 : P_i \in [P_i - \zeta_1, P_i + \zeta_2] \quad (15)$$

$$c_4 : \Theta_i \quad (16)$$

c_1 describes a reasonable rotation range constraint for finger i's Joint j, which is used to ensure that the finger has a natural pose.

c_2 describes the distance constraint between the fingertip and base, so that the fingertip can reach the required piano key. $d(T_i, B_i)_{\min}$ and $d(T_i, B_i)_{\max}$ respectively denote the maximal and minimal distance range between finger i's tip and base.

c_3 is the translation constraint used to maintain the local optimized position P_i for Chord i. Because our system can usually generate a good initial pose for each chord, our optimization method uses these good initial poses to generate natural and energy-saving poses with smaller variable range of wrist translation around initial poses.

c_4 is used to generate a natural wrist orientation Θ_i on the basis of the P_i and five finger distributions and is computed in Section 4.1.3.

6. SIMULATION OF MOTION CURVE

After the optimized natural key poses for the given notes are generated, the following set of steps are used to construct the realistic motion curve between these key poses.

6.1. Wrist Motion between Chords

When the hand moves from one chord to another, the hand will move up and down during the motion as shown in Figure 10. Motion capture data show that the wrist is usually raised to its maximum height (along the Y axis) in the middle of two neighboring chords, and this component is calculated by

$$P_W((t_i + t_{i-1})/2)_y = P_W(t_i)_y + P_W(t_{i-1})_y + V(i) * D(i, i-1) * H(P_W(t_i), P_W(t_{i-1})) \quad (17)$$

where V is linearly determined by the volume of Chord i; D is linearly determined by Chord $i-1$ and Chord i's duration; H is the basic height, inversely proportionate to wrist translation between the two chords.

After determining the height in the middle position, motion capture data are used to interpolate the key frames between the middle position and the two key poses for the two chords along axis Y, and the motion capture data along the X and Z axes are used to generate the corresponding curve components in order to generate more realistic hand motion by the following procedure: (1) sample hand motion used for 20 performances of the same chords; (2) segment the motion capture curve manually into up and down motion; (3) normalize the motion capture clips to the same duration and average them to generate a reference curve; and (4) sample this reference curve and use it as the interpolation function when generating output motion.

6.2. Influence of Instructed Fingers on Non-instructed Fingers

While the piano keys are struck, the rotation of instructed fingers induces some rotation in the non-instructed fingers, just as shown in Figure 11. Usually, the less skilled the player is, the more the dependence the instructed fingers will have on the non-instructed fingers. In order to simulate the influence of instructed fingers on non-instructed fingers, we define dependence index $Dep(i)$ for non-instructed finger i's movement influenced by instructed finger j as

$$Dep(i) = \frac{\sum S_{ij}(i, j)}{I} \quad (18)$$

where S_{ij} is the slope of the relative motion of the ith finger during the jth instructed movement [19] and I is the number of instructed fingers. If finger i is an instructed finger, the dependence index will be 0 because the instructed finger should exactly press on the piano key no matter where the other fingers are. If finger i is a non-instructed finger, it will be close to 1 when the neighboring instructed fingers have high influence on this finger and will be close to 0 when the neighboring instructed fingers have little influence. Given a maximum movement range $Y_{\max}(i)$ along the Y axis, the position of the non-instructed finger along the Y axis due to the influence of surrounding fingers is given by

$$P(i)_y = Y_{\max}(i) * Dep(i). \quad (19)$$

Figure 10. Hand motion from one chord to another. The hand will reach its highest position above the piano keyboard in the middle of the motion.

DOI: 10.1002/cav

Figure 11. Rotation of instructed finger(s) influences the rotation of non-instructed fingers. In this example, the rotation of instructed middle finger influences the rotation of non-instructed ring and little fingers.

6.3. Wrist Compensation

The action of pressing a key will tend to induce an upward movement of the wrist due to the reaction force of the strike. While playing low-volume notes (such as only rotate finger base and/or wrist to strike the keys, which will generate low-volume sound), the wrist will do an up–down vertical response, and while playing high-volume notes (such as rotate up arm and/or shoulder to strike the keys, which will generate much larger volume sound), the wrist will do an down–up–down response. Additional key frames based on the feature between extracted motion capture data of wrist and the corresponding sound are inserted to achieve this, and the amount of compensation is therefore scaled on the basis of the note volume.

7. RESULTS AND DISCUSSION

We present piano animations showing a range of different finger placement and motions styles. Please refer to the supporting information for more animation details.

7.1. Scales

In music, a scale is a sequence of notes in ascending or descending order that is used to conveniently represent part or all of a musical work including melody and/or harmony [20]. Finger crossovers always happen during playing scales and are generated realistically in our system. This feature is demonstrated by Figure 12 and the first demo in the accompanying video, which includes a side-by-side comparison between the generated animation and real playing.

7.2. Chords

A chord consists of a set of notes that are heard as simultaneous sound but might not be played simultaneously. Generally, notes are played at the same time, except in the special case of an arpeggio, where the notes are played quickly in sequence. This can be handled by the simple case of individual notes being pressed. As follows, we discuss the common chord that is played simultaneously and show

Figure 12. Key poses of finger crossover while playing scales. The first row shows a key frame of the thumb crossing over the middle finger while playing the C-major scale, and the second row shows a key frame of the thumb crossing over the ring finger while playing the D-major scale, both from three perspectives. Note that the ring/middle finger firmly presses down the keys, the fingers avoid collisions with black keys in C-major, the wrist maintains a natural rotation, and the thumb is positioned well on the key to play it after crossing over.

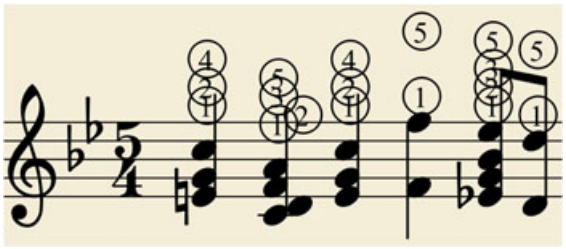

Figure 13. Correct fingering generated for the first part of "Bilder einer Ausstellung".

Figure 14. Pose for (a) E4-G4-C5, (b) F4-F5, and (c) #D4-G4-#A4-#D5.

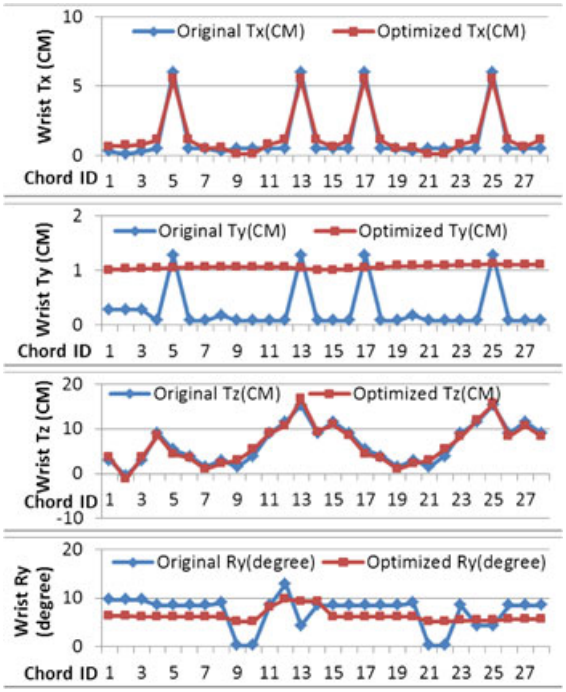

Figure 15. Motion comparison before and after optimization. The first three sub-figures show that the motion curves along three axis components are smoothed after optimization as the red curves show. The fourth sub-figure shows that the rotation of the wrist along the vertical axis(the *Y* axis) is also smoothed. The four figures together demonstrate that the hand moves with less distance and rotation for the same music clip after optimization, and therefore, the hand motion is more smooth after the optimization routine.

that our algorithm generates realistic piano performance animation for complex chords.

Figure 13 shows that our system generates correct fingering of instructed fingers for the first part (six chords) of the musical notation of *Bilder einer Ausstellung* composed by Modest Mussorgsky.

The generated fingering choice is found to be very feasible for the hand, and Figure 14 shows snapshots of hand poses for some complex chords in the chord demo, which are again feasible to play and natural.

We analyze the realism of our animation as follows after the optimization step. In this example, our optimization method improves the total motion cost (150.6) by about 22%, translation improvement(111.3 cm) by 10%, and rotation improvement (78.5°) by 45%. Note that the default weight between translation and rotation is 1.

Figure 15 visualizes the three translation components and the one rotation component along the *Y* axis (the other components are based on the same parameters and are the same) before and after optimization. These graphs illustrate that the optimization method can yield a smoother key pose sequence with less wrist translation and rotation and therefore can minimize motion cost. Note that the 28 nodes in each curve correspond to the key poses for the 28 chords; the line between nodes is used to better trace how the wrist component changes as the chord music progresses.

Figure 16 shows the music with 28 chords; the animated wrist motion agrees well with the ground truth data.

7.3. A Music Piece

Finally, a music piece, *Childhood Memory*, composed by Modest Mussorgsky, is used to generate a comprehensive demo to show all the features supported by our current system, including the simulation of relative finger rotation, wrist compensation, finger crossover, and arpeggio skill. Some key snapshots are shown in the following images.

The first image in Figure 17 shows the instructed index finger for the next chord causing the relative rotation of the non-instructed ring finger; the second shows the index finger fully pressing down the key while the ring finger returns back to the key surface; the third shows the wrist moving up because of wrist compensation after the index finger fully presses down the key (the wrist motion causes the joints of other fingers to rotate a little while keeping contact with the piano keys).

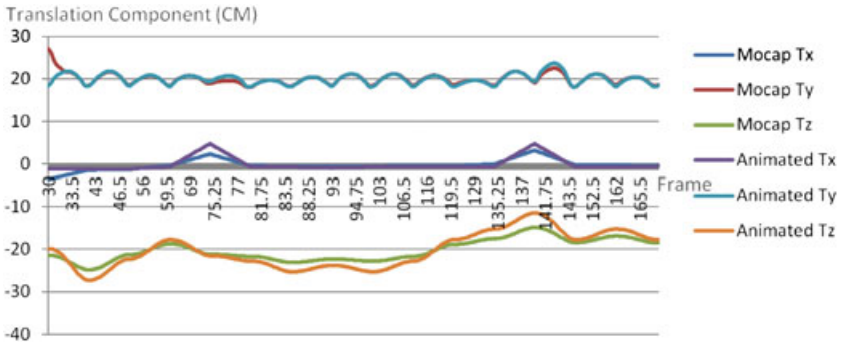

Figure 16. Motion comparison between ground truth data and optimization result. The motion curves along three translation components follows tightly with the those of corresponding motion capture data for the same music clip.

DOI: 10.1002/cav

Figure 17. Some key-poses while playing "Childhood Memory".

8. CONCLUSION AND FUTURE WORK

We have described a system that automatically generates three-dimensional animation of piano playing, given an input piece of music. The graph-based approach determines fingering, which ensures that the character plays the piece in as relaxed a manner as possible, which is one of the fundamental principles of piano theory. A novel rule-based approach is proposed to pre-position non-instructed fingers such that it is convenient and easy to use them for playing succeeding notes. Initial key hand poses are determined on the basis of generated fingering and piano theory, and the complex but often encountered cases such as finger crossovers and arpeggio are also handled. An optimization-based method operating on geometry constraints is proposed to generate smooth and natural key pose sequences for the hand. Motion capture data are then employed to further smooth the transition between poses. We believe the resulting motion is realistic enough to be used directly as a tool for piano self-study. The comparison between the generated motion curves and the curves of the raw motion capture data of a real piano playing shows a good level of realism.

Our hand touch model, which allows goal locations for each finger and maintains exact timing constraints, could be extended to perform animation of other instruments with keys, such as woodwinds, brass, and string instruments. Also, our approach may be beneficial for generating natural grasping with a more beautiful hand pose.

The first limitation of our system is that although it solves the collision between the fingertip and the piano surface, it does not handle interpenetration between fingers and collisions with the sides of the black keys.

Secondly, although our system can generate plausible and reasonably realistic piano playing for standard music, it is not capable of generating emotional piano playing that reflects a personal understanding of the music and player's performance background. This is the main future work we will be pursuing.

Thirdly, it might be beneficial to apply principles of machine learning to our system to learn the parameters for determining standard fingering more accurately in cases where multiple fingering sequences have the same optimal cost during instructed fingering generation.

Fourthly, we will enhance our system to generate animation for various size hand models, to meet the requirements of piano students with different hand shapes, so that our system can be used as a good piano teaching tool.

Fifthly, sometimes a melody must be played continuously by two hands. Because our method generates fingering for each hand separately, our work cannot generate fingering for this performance that requires planning fingering simultaneously for two hands. Note that this problem is also unsolved in all of the previous works.

Finally, our solution does not consider the interdependent rotation of the joints within a finger and does not directly simulate the influence from interdependence between fingers. Future work on this point would be helpful to improve the finger motion during striking and releasing the piano keys.

ACKNOWLEDGEMENTS

We would like to thank 3D artist and programmer Paris Maroidis for his quite important contribution to the development of our system, Jonathan Graham for his modeling of the piano character, Yingying Wang and Pengcheng Luo for their help during motion capture process, Professor Hong Zhu for her suggestion related to piano theory, and Binglin Li's piano performance for motion capture work.

REFERENCES

1. Sayegh SI. Fingering for string instruments with the optimum path paradigm. *Computer Music Journal* 1989; **13**(3): 76–84.
2. Heijink H, Meulenbroek RGJ. On the complexity of classical guitar playing: functional adaptations to task constraints. *Journal of Motor Behaviour* 2002; **34**(4): 339–351.
3. Viana AB, de M. Junior AC. Technological improvements in the SIEDP. In *IX Brazilian Symposium on Computer Music*, Campinas, Spain, 2003.
4. Lin C-C, Liu DS-M. An intelligent virtual piano tutor. In *Proceedings of the 2006 ACM International Conference on Virtual Reality Continuum and its Applications*, Hong Kong, China, 2006; 353–356.
5. Clarke JA, Raekallio EF, Parncutt M, Sloboda R, Desain P. An ergonomic model of keyboard fingering for melodic fragments. *Music Perception: An Interdisciplinary Journal* 1997; **14**: 341–382.
6. Yonebayashi H, Kameoka Y, Sagayama S. Automatic decision of piano fingering based on hidden Markov models. In *Proceedings of the 20th International Joint Conference on Artificial Intelligence*, Hyderabad, India, 2007; 2915–2921.
7. Radisavljevic A, Driessen P. Path difference learning for guitar fingering problem. In *Proceedings of the International Computer Music Conference*, Miami, Florida, 2004.

8. Tuohy DR, Potter WD. A genetic algorithm for the automatic generation of playable guitar tablature. In *Proceedings of the International Computer Music Conference*, Barcelona, Spain, 2005; 499–502.
9. Tuohy DR. Creating tablature and arranging music for guitar with genetic algorithms and artificial neural networks, *A Thesis Submitted to the Graduate Faculty of The University of Georgia in Partial Fulfillment of the Requirements for the Degree Master of Science, The University of Georgia*, 2006.
10. Hart R, Bosch M, Tsai E. Finding optimal piano fingerings. *Undergraduate Mathematics and Its Applications* 2000; **21**(2): 67–177.
11. Kasimi E, Nichols AA, Raphael C. Automatic fingering system. The International Society for Music Information Retrieval poster presentation, 2005.
12. Radicioni L, Anselma D, Lombardo V. A segmentation-based prototype to compute string instruments fingering, In *Proceedings of the Conference on Interdisciplinary Musicology*, Graz, Austria, 2004.
13. Radicioni D, Lombardo V. Guitar fingering for music performance. In *Proceedings of the International Computer Music Conference*, Seattle, USA, 2005; 527–530.
14. Yasumuro Q, Chen Y, Chihara K. Three-dimensional modeling of the human hand with motion constraints. *Proceedings of Image and Vision Computing* 1999; **17**(2): 149–156.
15. Lee K-H, Kroemer KH. A finger model with constant tendon moment arms, In *Proceedings of Human Factors and Ergonomics Society 37th Annual Meeting*, Vol. 37, Santa Monica, USA, 1993; 710–714.
16. Pollard NS, Zordan VB. Physically based grasping control from example. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, Los Angeles, USA, 2005; 311–318.
17. ElKoura G, Singh K. Handrix: animating the human hand. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, San Diego, USA, 2003; 110–119.
18. Kim J, Cordier F, Magnenat-Thalmann N. Neural network-based violinist's hand animation. In *Proceedings of Computer Graphics International 2000*, Geneva, Schweiz, 2000; 37–41.
19. Hager-Ross C, Schieber MH. Quantifying the independence of human finger movements: comparisons of digits, hands, and movement frequencies. *The Journal of Neuroscience* 2000; **20**(22): 8542–8550.
20. Benward B, Saker M. *Music in Theory and Practice*, (7th edn), Vol. 1. Mcgraw-Hill College, New York, USA, 1997.

AUTHORS' BIOGRAPHIES

Yuanfeng Zhu is a PhD student in the Department of Computer Science at the University of California, Davis. His main research is on music-driven hand motion and physics-based character motion.

Ajay Sundar Ramakrishnan graduated with an MS degree from the Department of Computer Science at the University of California, Davis, and now working in Intel Company, in Folsom, USA. His previous research was on gesture recognition and hand motion.

Bernd Hamann is a full professor in the Department of Computer Science at the University of California, Davis. His main research and teaching interests are visualization, geometric design and modeling, and computer graphics.

Michael Neff is an Associate Professor of Computer Science and Cinema and Technocultural Studies at the University of California, Davis. His research focuses on character animation, and gesture and interactive techniques for computer graphics.

DOI: 10.1002/cav