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

FootSee: an Interactive Animation System

KangKang Yin[1] and Dinesh K. Pai[1,2]

[1] Department of Computer Science, The University of British Columbia, Vancouver, BC, Canada
[2] Department of Computer Science, Rutgers, the State University of New Jersey, Piscataway, NJ

Abstract

We present an intuitive animation interface that uses a foot pressure sensor pad to interactively control avatars for video games, virtual reality, and low-cost performance-driven animation. During an offline training phase, we capture full body motions with a motion capture system, as well as the corresponding foot-ground pressure distributions with a pressure sensor pad, into a database. At run time, the user acts out the animation desired on the pressure sensor pad. The system then tries to "see" the motion only through the foot-ground interactions measured, and the most appropriate motions from the database are selected, and edited online to drive the avatar. We describe our motion recognition, motion blending, and inverse kinematics algorithms in detail. They are easy to implement, and cheap to compute. FootSee can control a virtual avatar in a fixed latency of 1 second with reasonable accuracy. Our system thus makes it possible to create interactive animations without the cost or inconveniences of a full body motion capture system.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1. Introduction

Avatar control in video games, virtual reality, and human-like character animation has been widely studied (see [8] for a survey). There are several reasons why this problem is challenging. Many animation packages have complicated mouse-based interfaces, and are only mastered by a few experts. More exotic interfaces[25] are more intuitive, but still not very efficient. To make matters worse, humans are really good at human motion perception; some recent studies of human motion perception include [13, 28].

For these reasons, motion capture based performance-driven character animation has become popular and widely used[18, 9, 5, 4]. However, full body motion capture is expensive and delicate. The capturing process involves a lot of work including careful preparation, calibration, special clothing, and tedious post-processing. Applications like interactive video games can hardly afford these.

Our system, named FootSee, uses a non-intrusive easy-to-use pressure sensor mat as the animation front end. A high quality motion database serves as the animation back end. Our hypothesis is that many full body motions have their own distinctive footwork. In addition, postural control is an essential part of all full body motions, and is constantly regulated by foot-ground interactions. Our work provides a viable alternative to applications that need an intuitive, interactive, robust, and high quality animation interface, but do not need high accuracy reconstruction (i.e., require only plausible animations and can afford occasional mistakes), and do not involve subtle motions of the upper body.

2. Related Work

There is rich information encoded in the foot-ground interaction forces. Force platforms (often called force plates) have been widely used to measure ground reaction forces (GRF) for clinical gait analysis, athletic performance analysis, rehabilitation, kinetic and biomechanics research [31, 34, 1, 24, 21]. In these applications, one typically measures the total force and moment. In contrast, we measure the normal pressure distribution to extract information about locomotion and postural control. To our knowledge, there has been no application of the foot-ground pressure measurement in computer animation. A method of using footprints as a basis for generating animated locomotion was presented in [32, 30] . As the authors point out, appropriate timing and footprint positions are key to the success of their algorithms, but hard to obtain.

A pressure sensor pad makes a good candidate as a front-end tool, replacing tedious and non-intuitive manual input. In the robotics community, however, there has been some work using bed pressure for monitoring patient activity[11]. They target posture estimation while we aim at full body animation.

Although dynamic simulation and/or machine learning techniques have been used to animate humans[14, 10], reusing motion capture data is still much more realistic. The idea of motion synthesis based on pre-captured motion databases has been extensively explored recently[16, 2, 20, 18, 26]. State-of-the-art methods require a training or learning phase that typically lasts for several hours, after which, motions of a certain kind (locomotion, disco dance, or climbing) can be synthesized interactively, sometimes even in real time. In particular, [18] focused on interactive avatar control, and reported an intuitive vision-based interface that can control an avatar to step around a stool with a 3-second lag. Motion recognition was done by comparing visual features extracted from online images of the user and the pre-rendered images of the motions in the database. Vision based interfaces have been studied in the computer vision community for quite some time for various tasks such as motion tracking and motion recognition[22, 6]. Our work follows the same track of [18] but uses a novel front end: a foot pressure sensor pad. The advantage of the foot pressure sensor pad is that it is viewpoint free (i.e., always under your feet) and occlusion free. The determination of foot-ground contacts, which has been proved very important to full body animation in previous work, is fully automatic. The pressure sensor pad does have its own limitations, which we will elaborate on later.

3. System Overview

FootSee consists of an off-line data capture and database construction stage (Section 4), an online motion recognition stage (Section 5), and a motion editing stage (Section 6). Figure 1 illustrates the main idea.

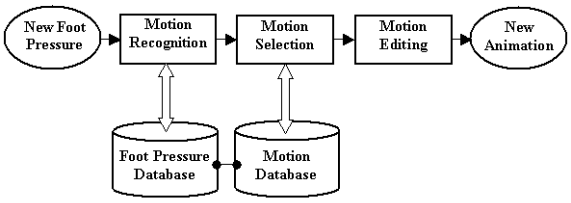


Figure 1: *System illustration. The data in the foot pressure database and the motion database are properly synchronized during motion capture.*

The motion database and the foot pressure database are properly synchronized during motion capture, so that the motion selection module just fetches the motion corresponding to the matching footwork. However, we can also deliberately change the association rules and map footwork to fake motions. For example, we could map a marching-in-place motion to walking. This would allow the user to explore a virtual environment in a limited real environment.

4. Data Capture and Processing

4.1. Data Capture

The database consists of synchronized full body motion data and foot pressure data. Full body motions are captured by a Vicon 6[33] motion capture system at 60 Hz. Foot-ground pressure data is captured by an XSensor[35] pressure pad at 12 Hz. This gives us 5 motion frames for every foot pressure frame. Our motion capture volume is roughly 1.5m long, 2m wide, and 2m tall. The pressure sensor pad is about 0.8m long and 2m wide. Given the constraints of the motion capture volume and the pressure pad dimension, we selected a number of behaviors to capture: quiet standing (with natural sway), and interesting motions (kicking, punching, stepping, weight transfer and balancing). The users were requested to return to quiet standing between interesting motions, and there was no specification for the duration of quiet standing.

We use Filmbox[15] to map the Vicon marker position data onto the skeleton shown in Figure 3(b). This skeleton has 21 3-DOF (degrees of freedom) joints. The joint coordinates are represented by three axes (X,Y,Z) (coloured red, green, and blue in the figure). The kinematic root (the black dot in Figure 3(b)) is located at the intersection of the Lumbosacral angle of the spine (the base of the spine) and the pelvic girdle. Mapping the animation onto the skeleton converts the motion data from marker positions to root positions and joint angles. The root planar coordinates (the (x,z) coordinates) are further converted from absolute positions into relative positions (the difference between consecutive frames), so transitions can be made easily to similar poses at different planar locations during later processing.

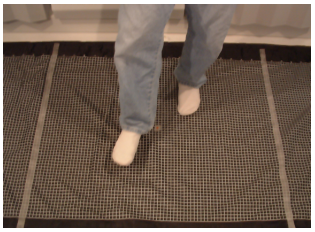


Figure 2: *Custom XSensor pressure sensor pad.*

Figure 2 shows our XSensor pressure sensor pad. It is made of a 160 by 64 grid of pressure sensors. It was originally designed for measuring the pressures of a person lying on a bed, and was capable of sampling the whole pad at 6 Hz. Our pad is specially constructed to measure the larger pressures due to standing and running. Using custom software developed to our specifications, we can sample a smaller region of interest at a higher rate.

In our experiments we only sample the central 80 by 64 region to double the sampling rate to 12 Hz. The spatial separation of the pressure sensors is 0.5 inches. Each pressure pixel is a byte value representing 255 different pressure levels. The upper part of Figure 3(a) is a sample pressure image when the subject is standing still. For easy visualization, we discard 0 pressure values; we map minimum pressure 1 to dark green (R,G,B)=(0,127,0), and maximum pressure 255 to dark red (R,G,B)=(127,0,0). For quiet standing, the pressure in all contact areas are generally low, so we only see green in Figure 3(a). High pressure values usually show up in the heel area when there is only one foot in contact with the ground, as the red heel shown in Figure 3(c).

4.2. Feature Extraction

For motion recognition (Section 5), a 10-component feature vector $\mathbf{x} = (x_1, x_2, ...x_{10})$ is extracted from every pressure image. The feature vector consists of the velocity of the center of pressure (COP) (2 components), contact area of both feet (2 components), and the first 6 Hu moments (6 components). The COP velocity helps differentiate the motion directions (stepping direction, weight shifting direction etc.). The contact areas help differentiate motion types (one-foot-on-ground motion, e.g., kicking or stepping; or two-foot-on-ground motion) and bilateral motions (left-foot kicking or right-foot kicking). Hu moments are a set of algebraic invariants that combine regular moments[3]. They are invariant under change of size, translation, and rotation. Hu moments have been widely used in pattern recognition and proved successful in various applications. Other measures, such as separation of the feet, may easily come into mind as feature candidates. However, they are not used because they contain singularities; for instance, when one foot is in the air, the foot separation is not defined just from the pressure data.

We will use the Mahalanobis distance[12] of feature vectors as the distance metric for motion recognition later. Assuming the component variables of the feature vector are uncorrelated, the Manalanobis distance of $\mathbf{x}_i$ and $\mathbf{x}_j$ is defined as

$$d_{i,j} = (\mathbf{x}_i - \mathbf{x}_j)^T C^{-1} (\mathbf{x}_i - \mathbf{x}_j) \qquad (1)$$

where $C = diag\{\sigma_1^2, ..., \sigma_{10}^2\}$ is the diagonal covariance matrix computed from the database feature vectors.

To compute the contact areas of the left foot and the right foot, we first need to know the foot to which a pressure pixel belongs. In our setup, the Vicon cameras are all placed in the front of the subject and all the motions we captured are facing in roughly the same direction. With the foot orientation known, foot tracking and recognition is very simple. The peak pressure point and all its neighborhood pixels within a bounding box (a box little bigger than the subject's foot) are classified as one foot. The peak pressure point of the remaining pixels and its neighborhood pixels are classified as the other foot. It is easy to determine which foot is left and which is right since the orientation of the subject is relatively fixed. In case there is only one foot in contact with the ground, e.g., Figure 3(c), we determine left and right by checking which foot it is closer to just before one of the feet disappears. For visualization, the left foot is bordered yellow, and the right foot is bordered red. The black dot is the center of pressure. The coloured dots are the center of pressure of the corresponding foot.

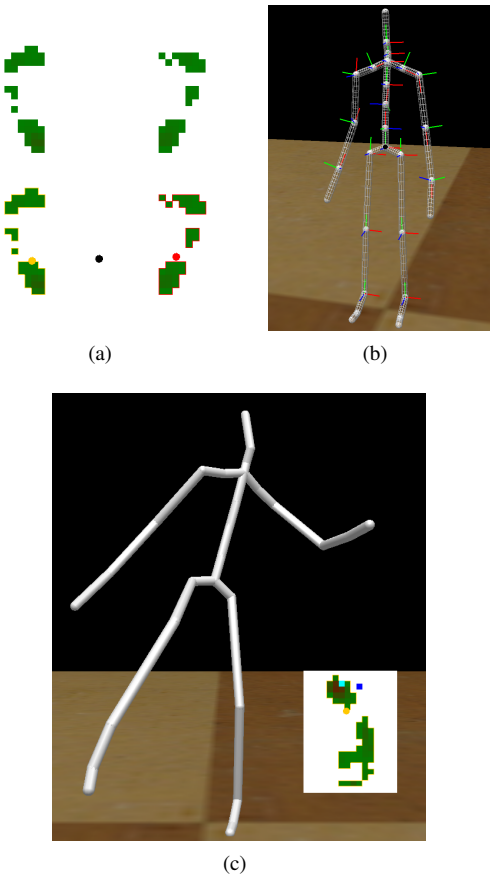


Figure 3: *(a) A raw (top) and a labelled (bottom) foot pressure distribution image. Pressure level 1 is mapped to dark green and pressure level 255 is mapped to dark red. The black dot is the center of pressure. The coloured dots are the center of pressure of the same color bordered foot. The yellow foot is the left foot. The red foot is the right foot. (b) The skeleton model. The black dot is the kinematic root. The (X,Y,Z) axes of joint coordinates are represented by the red, green and blue axes. (c) A foot pressure image and its corresponding motion frame. The blue dot is the ankle position estimated from the avatar. The cyan dot is the ankle position estimated from the pressure data.*

To perform inverse kinematics later (Section 6.1), we also need to estimate the ankle position of the user from the pres-

sure images. When the foot is in full contact with the ground, i.e., when its bounding box is sufficiently large, the rounded centroid of the last three rows of the foot is taken as the ankle position. The ankle position of the avatar is estimated by projecting the ankle joint position onto the horizontal plane, and then transforming it into the pressure pad coordinates. The transformation between the Vicon coordinate frame and the pressure pad coordinate frame is known at the capture and calibration stage. Usually we align the axes of the pad and Vicon calibration tools to reduce this transformation to a simple translation, determined by where we put the Vicon calibration tool. In Figure 3(c), the cyan dot is the estimated user ankle position, the blue dot is the estimated avatar ankle position.

5. Motion Recognition

As illustrated in Figure 1, the first step of the online motion synthesis is to recognize the user's motion by comparing the newly input foot-ground pressure images with the images contained in the foot pressure database. There is no camera involved in the whole avatar control process once the database is constructed. The system "sees" the users through their feet. Our online motion recognition is based on activity detection and template matching. An offline supervised learning process was done after the data capture. We describe it here rather than in the last section because it is more relevant to the motion recognition task.

After the data capture and processing described in the last section, we manually segment the motion database into interleaved quiet standing and action segments. Basically this involves a human watching the motions in the database and marking out roughly where each interesting motion starts and ends. After the motion sequence is segmented, the proper segmentation of the pressure image sequence is also known because the two databases are synchronized and we know their correspondences. We then compute the mean μ and variance σ^2 of the features of the quiet standing pressure images, i.e., we represent the quiet standing as a random variable with a multidimensional Gaussian distribution $N(\mu, \sigma^2)$. For new input foot pressure data, we extract the features as before (Section 4.2), online. We define an activity score s_i for frame i as the Mahalanobis distance from the feature vector $\mathbf{x}$ to the quiet standing cluster center μ. When s_i is large enough (above a chosen threshold), we conclude there is an interesting motion possibly going on.

If a possible activity is detected, the feature vector of the current pressure frame i along with those of k look-ahead frames and l look-back frames are grouped together as a feature window X for the current motion: $X_i = (\mathbf{x}_{i-l}, ...\mathbf{x}_i, ..., \mathbf{x}_{i+k})$. The feature windows of the onsets of all the interesting motions in the database (denoted as $X_{i'}$) are candidates that we will compare with the current feature window. We then define the distance of two feature windows as

$$D_{i,i'} = \sum_{j=-l}^{k} w_j d_{i+j,i'+j} \tag{2}$$

where $d_{i,i'}$ is defined in Equation 1, and w_j is the weight computed from a simple linear hat function centered at $j = 0$. The motion $X_{i'}$ with the minimum motion distance to the current feature window X_i is recognized as the matching motion, and is selected for motion editing in the next step. To reduce false positives, i.e., a quiet standing recognized as an interesting motion, a special quiet standing motion is made up by repeating μ $(l+k+1)$ frames, i.e., $X_o = (\mu, ...\mu, ..., \mu)$. If it turns out this quiet standing motion is the best matching motion, the current frame is just a quiet standing with a bigger sway, and no transition will be made.

During an interesting motion, we still monitor the motion distance between the current input feature window and that of the recognized motion, rather than just playing back the recognized motion blindly. The reason is two fold. First, similar behaviors may have different time course. This is especially true for weight transferring motions in our experiments: the user may shift their weight to the desired location (say left leg), then stay in that pose for a while (the duration will be different each time), then shift the weight back. If we find strong evidence that the selected motion is too slow or too fast compared to the user input, we do a timewarping (Section 6.3) to fast forward or slow down the motion selected from the database. The second reason that we monitor the motion distance is that there is always the possibility of a mismatch. In this case we need to perform a new search and find a better motion. We always try to find a good warp before we give up on the current motion and jump to a new motion. One point of importance is that we cannot judge the quality of the current best match by $D_{i,i'}$, but rather we normalize it and use $D_{i,i'}/(s_i + s_{i'})$ as an indication of how good the best match is. This is because for highly dynamic motions (motions with high activity scores) even a very good match may have large distance to the user motion, while a relatively subtle motion may have small distance to a bad match.

In our experiments, we use $k = 10$ look-ahead frames and $l = 6$ look-back frames. We initialize a motion buffer with 5 motion frames (the duration of 1 pressure frame). Subsequent synthesized motions are put into the buffer for rendering at 60 Hz. All the transition decisions plus the synthesis of the first motion frame after a transition can be made before the motion buffer underflows. So the total latency of the system is 12 pressure frames (1 frame of buffering + 10 look ahead frames + the current frame), i.e., 1 second at our pressure sampling rate of 12 Hz.

The current algorithm for organizing and searching data is sufficient for our test database. However, as the database grows larger, better algorithms may be needed. For instance,

Motion graph techniques[16, 18, 2] can be used. Hierachical classification and organization[18] of the motion database can also boost search speed. We can also use feature vectors partially to quickly rule out bad matches, say we use the velocity of COP $\hat{\mathbf{x}} = (x_1, x_2)$ to quickly discard motions to different directions. Multiple matching hypotheses can also be maintained whenever we search the database.

6. Motion Editing

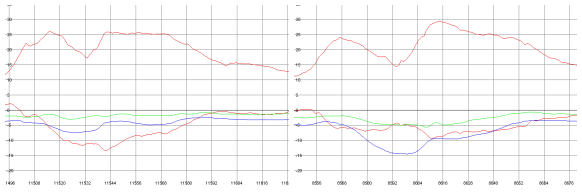
The motion recognition module outputs the closest motion in the database, for each input pressure image. When there is a transition, the current motion has to be transformed into the new motion smoothly. Simple spherical linear interpolation and displacement mapping[7] are currently used. For a couple of situations, such as inverse kinematics, foot-ground contact constraint satisfaction and timewarping, simple blending has to be combined with special manipulations to maintain time and spatial constraints. Since we want motion editing without incurring extra latency, we choose algorithms that are as simple as possible.

6.1. Inverse Kinematics for Stepping

It is unlikely that the stepping length and direction a user takes at runtime will be exactly the same as those of the steps stored in the database. When the stepping error accumulates over time, the avatar could drift further and further away from the location of the real user. We developed an analytical inverse kinematics (IK) algorithm to modify the steps selected from the database to match the user's input steps when large position error is detected. This IK has the following assumptions, and we will give the rationale shortly:

1. The root orientation and planar (x,z) position are unchanged.
2. The hip joint angles are modified to meet stepping changes.
3. The ankle and knee joint angles are only modified to keep constraints satisfied, i.e, compensating for rotation of the foot during stance, and ground penetration during swing.

The stepping length, direction and height are mainly controlled by the orientations of the pelvic girdle, the left and the right hips, and the left and the right knees. In our experiments, the user is always facing the same direction so the orientation of the pelvic girdle (the root orientation) does not change much. So we left out the pelvic orientation from our IK. Next to the shoulder joint, the hip joint is the most movable of all joints. It is a ball-and-socket type of joint (3 DOF). The knee joint is primarily a hinge type of joint, combined with a small amount of gliding and rolling[23]. It is usually treated in computer graphics as a 1-DOF joint allowing only flexion and extension. Although our skeleton model treats all joints as 3-DOF joints, the 2 extra joint angles of the knees remain small all the time. Because we consider only flat terrain, the knee flexion-extension characteristics



(a) right forward stepping (b) right side stepping

Figure 4: *The joint angles of the right leg for two steps. The horizontal coordinate is the frame number. The vertical coordinate is the joint angle. In both figures, the top red curve is the knee joint angle. The bottom red curve is the hip flexion-extension. The blue curve is the hip abduction-adduction. The green curve is the hip axial rotation. Please refer to Figure 3(b) for the joint angles' corresponding rotation axes which are coloured accordingly.*

for different steps do not vary greatly in our experiments. Figure 4 shows the joint angles of two steps we captured. One is a right forward stepping, the other is a right side stepping. The patterns for knee angles are quite similar: a flexion in toe-off, followed by an extension in strike, followed by a flexion in opposite toe-off, followed by an extension in opposite strike[29]. So basically there are two peaks in the knee angle curves. The hip angles, however, vary with the stepping length and direction more directly. In side stepping, there is more abduction-adduction; while in forward stepping there is more flexion-extension.

Based on the above observations, we designed the fast IK algorithm shown in Figure 5. In Figure 5(a), we know the initial joint positions of the leg, thus we can compute the hip-ankle vector **a**, and the projected hip-ankle vector **b** (**a** projected onto the horizontal plane through the ankle). Suppose we want to shift the ankle position by **c** to get to a new location estimated from the new pressure data, keeping the knee joint angle unchanged and allowing the hip position to only move vertically (which is equivalent to the first assumption). The new hip-ankle vector **e** can be computed, because **d** can be computed from **b** and **c**, and the length of **e** equals the length of **a**. With **a** and **e** available, we compute the quaternion **q** that rotates **a** to **e**.

$$\mathbf{q} = (\mathbf{k}\sin(\theta/2), \cos(\theta/2)) \tag{3}$$

where

$$\theta = \arccos\left(\frac{\mathbf{a}\cdot\mathbf{e}}{\|\mathbf{a}\|\|\mathbf{e}\|}\right) \tag{4}$$

and

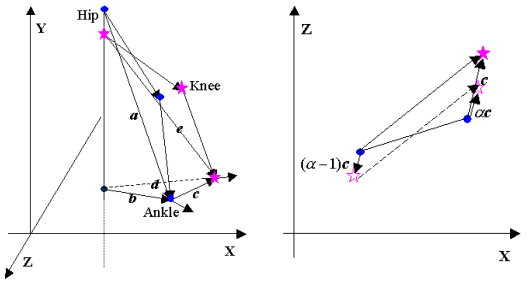
$$\mathbf{k} = \frac{\mathbf{a}\times\mathbf{e}}{\|\mathbf{a}\times\mathbf{e}\|} \tag{5}$$

q is then concatenated with the quaternion computed from the original hip joint angles, so the ankle can be transformed to the new location. This completes the IK for a single leg.

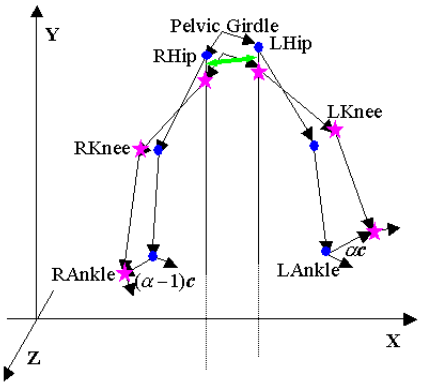
However, stepping involves two legs. Say we want to shift the ankle of the leading leg by **c**. We can not just ask one leg to try to make this shift because it will likely change the root's vertical position. The other leg will be affected by this root shift, and since its configuration is not changed, its ankle will be likely moved up into the air or down under the ground. So we need to distribute the shift **c** to both legs with two requirements:

1. The summed shift of the two ankles equals **c**.
2. The vertical translation of the right hip equals that of the left hip, because both of them are connected to the pelvic girdle (see the green bi-directional arrow in Figure 5(c)).

From requirement 1, we simply let the shift for the leading leg to be $\alpha\mathbf{c}$, and the shift of the stance leg to be $(\alpha-1)\mathbf{c}$, see Figure 5(b). Then by requirement 2, we equate the vertical shifts of the two hips. It happens that α has a simple closed-form solution for this hip constraint (see Appendix A).



(a) One-leg IK (b) Planar illustration



(c) Two-leg IK

Figure 5: *IK algorithm illustration. Dots are original joint locations, stars are new joint locations.*

Although we compute the pose difference by shifting both ankles, i.e., by shifting the dots to the imaginary hollow stars in Figure 5(b), we do not really shift both ankles in the synthesized motion since the stance ankle has to remain fixed. The kinematic root is temporarily switched to the stance ankle after the leading leg takes off, and we perform a displacement mapping from the original hip rotations to the desired hip rotations during the swing phase of the leading leg, until its touchdown. The pose of the lower body is transformed so that we get a step from the left dot to the solid star in Figure 5(b). The imaginary dashed step was used for the IK pose computation, while the parallel solid step is the actual step that satisfies both the stance ankle constraint and the stepping ankle displacement requirement.

During the step transformation, two constraints have to be kept. First, due to the hip rotation shift, the stance ankle rotation has to be shifted in the reverse direction (opposite of **q**) to counteract the hip rotation displacement and remain stable. Second, our IK basically keeps the knee flexion in the original stepping. However, large ankle displacement may need different knee flexion amplitudes to guarantee certain foot clearance for the swing foot (this can be seen from Figure 4). We constantly check how close the swing foot is to the ground. When there is danger of the foot hitting the ground during the swing, the knee is flexed allowing the foot to follow the height of its original swing motion. Figure 6(h) is an example of our IK. The original motion is a step to the right (see pressure data), and is modified to a right backward stepping by the IK to match the input pressure data.

More general IK algorithms can be used[17, 19], although we found ours is sufficient when the ankle shifts are small. When the ankle shift is large (typically that means a matching error occurred), we distribute the correction into several steps.

6.2. Foot-ground Contact Satisfaction

As already mentioned briefly in the last subsection, the foot-ground contact is a hard constraint that needs to be satisfied all the time and treated explicitly whenever an edit may change the foot location and orientation. For example in Figure 5(c), if we want to morph from the dot frame to the star frame, simple blending will result in ankles sliding from the dots to the stars. Foot slipping and excessive rotation are visually very disturbing.† There is a whole spectrum of how people deal with this problem, also known as *footskate*: some work only allows transitions during contact changes[18], while others have addressed this problem more fully[17].

We distinguish two cases according to how many feet remain in contact with the ground during a transition. Suppose we want to jump from motion frame i to frame j. If both feet are in contact with the ground during the transition, we have two constraints that we have to maintain. The displacements of the lower-body configuration (lower-body rotations

† This actually supports our hypothesis that foot-ground interaction is constantly regulating our full body motions, and justifies our interface from another perspective.

and root translation), denoted Δ_{ij} between frame i and frame j, are computed. Then, the configurations of frame j and its subsequent frames are all displaced by Δ_{ij}. This eliminates the footskates while most of the time no visible degradation of the motion quality could be detected. The reason is perhaps that humans are far less sensitive to small displacements of non-constrained body parts than they are to constrained ones (in this case the feet). Δ's of subsequent transitions are composed together with the current displacements should they occur. If during the transition only one foot keeps in contact with the ground, the solution is much simpler, as we already mentioned in Section 6.1. The kinematic root is temporarily switched to the stance ankle. Only the stance ankle needs to be kept fixed and all the other joints are simply morphed to the target configuration.

Over time, the accumulated lower body displacements Δ will be large enough that visible artifacts will appear. In our case, we reset this displacement to zero smoothly whenever the foot-ground contact breaks, such as in kicking or stepping motions. If there are no kicking or stepping in the database, an optimization routine [19] could be invoked regularly to find a configuration as close as possible to the target motions while maintaining the foot contact constraints.

6.3. Timewarping

The duration and target frame for timewarping is decided at the motion recognition phase (Section 5). Say we want to generate n motion frames for m frames fetched from the database to slow down ($n > m$) or speed up ($n < m$) the ongoing motion. We need to resample the original joint angle trajectories. Define $j_i = \lfloor \frac{i}{n} m \rfloor$, and $\alpha_i = \frac{i}{n} m - j_i$. Then the ith synthesized frame should be the $(\frac{i}{n} m)$th frame in the original motion sequence, which we estimate by a linear interpolation of frame j_i and frame $j_i + 1$ with interpolation coefficient α_i.

However, for the root (x,z) position (i.e., the root planar coordinates), since we already turned them into relative positions (Section 4.1), we need to accumulate them properly instead of simply resampling from surrounding frames. Denote $\mathbf{p'}_i$ as the relative root (x,z) position for frame i. It is not the interpolation of $\mathbf{p'}_{j_i}$ and $\mathbf{p'}_{j_i+1}$, but

$$\mathbf{p'}_i = \alpha_{i-1}\mathbf{p'}_{j_{i-1}+1} + \sum_{k=j_{i-1}+2}^{j_i-1} \mathbf{p'}_k + (1-\alpha)\mathbf{p'}_{j_i} \qquad (6)$$

when $n > m$. The same idea is used for computing $\mathbf{p'}_i$ when $n < m$.

If the latency requirement can be lifted, for instance, in an off-line performance-driven animation system, we could have a good sense of where an action begins and ends, and thus perform a true dynamic timewarping algorithm[7] to align the database motions with the user's motions.

7. Experimental Results

We experimented with this system on a dual-CPU (1.78 GHz Intel) machine. Currently FootSee can control an avatar with a fixed latency of 1 second, and render it at 60 Hz. The CPU usage during a typical session is very low (approximately 10% on average when measured using the Windows 2000 task manager), leaving the CPU mostly free for other tasks such as dynamic simulation and game AI. The performance comes from the simplicity and efficiency of the algorithms we adopted.

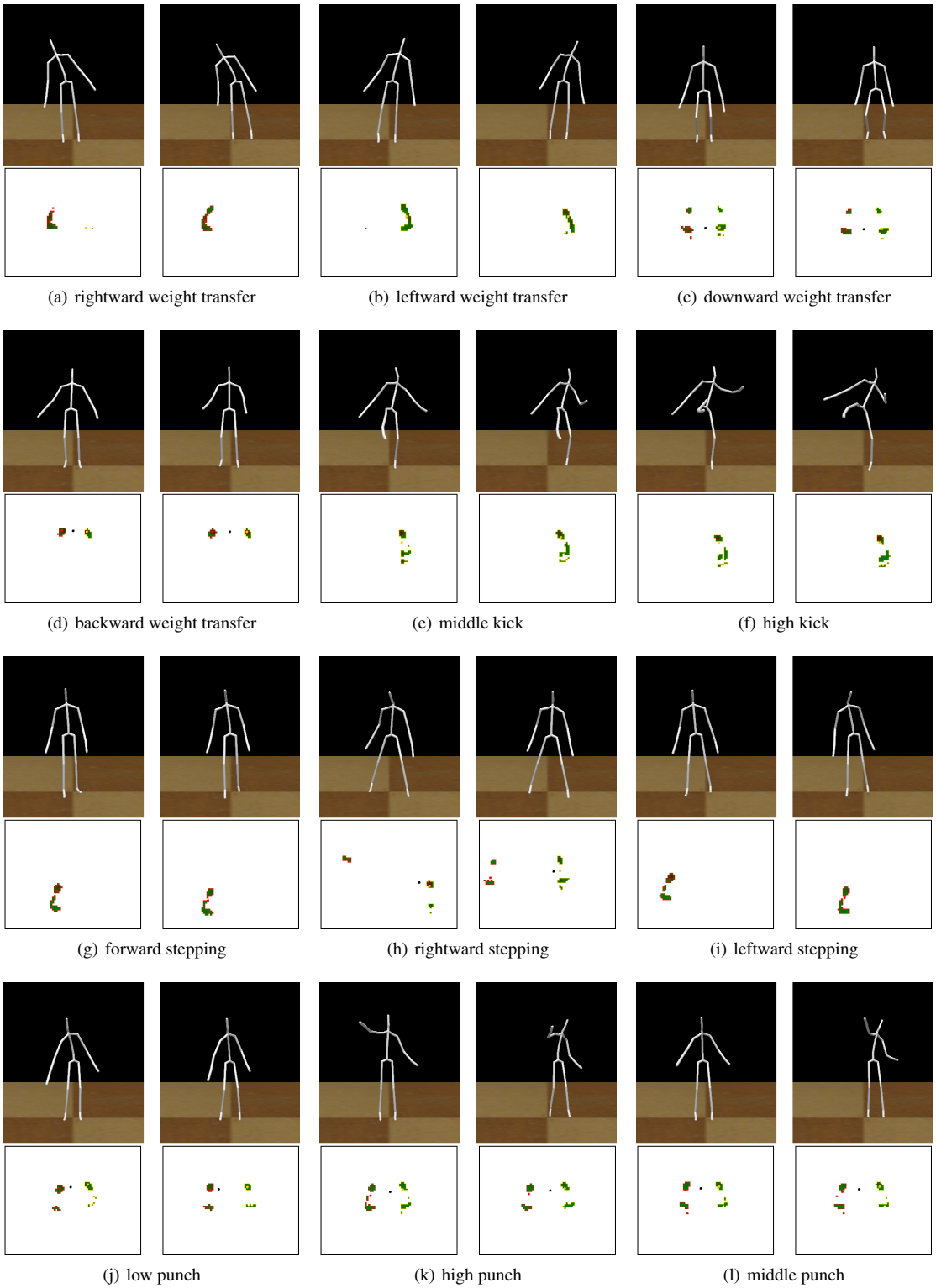
Figure 6 shows some samples of frames generated by FootSee. The lower left part of each image shows the input foot pressure image. The lower right part shows the corresponding pressure image of the best matching motion from the database. The upper right part is the controlled avatar. For comparison purpose, we also captured the motions of the real user. These can be seen in the upper left part of each image. Also for comparison, the motion of the user is rendered with a 1 second lag in sync with the output of FootSee, so that we can compare the poses easily.

The database for this example is about 5.5 minutes. The motions are captured from a subject with no formal martial arts training. Each action typically lasts from 1 to 3 seconds. The interesting motions were performed randomly and repeatedly with 49 occurrences in total. We count a match as a correct match if both the motion type and the motion extent of the user action and the avatar action match, i.e., a low punch matched to a high punch or a right forward step matched to a right side step is counted as a wrong match. With IK disabled (since it modifies steps), the recognition rate for four sessions with more than 10 minutes of motions in total is about 80%. There are three typical types of errors: 1) Upper body movement variations. For example, in Figure 6(l) a middle punch is matched to a high punch. 2) Confusion of low kicks and small steps. We found that the pressure images of the onsets of small kicks and steps are actually very similar. 3) Missing slow vertical weight transfers. The pressure images of slow downward weight transfer are sometimes very similar to those of quiet standing.

8. Discussion and Future Work

In summary, we have developed a new interface for interactive avatar control and low-cost performance-driven animation, using a foot pressure sensor pad and pre-captured pressure and motion databases. It is intuitive, non-intrusive and reasonably robust.

Obviously, one can not expect to completely reconstruct all full body motions only from foot pressure measurements, but we can get very plausible motions from a well chosen database. The performance we have demonstrated may be sufficient for many important applications such as virtual reality, video games, and performance-driven animation.



(a) rightward weight transfer (b) leftward weight transfer (c) downward weight transfer

(d) backward weight transfer (e) middle kick (f) high kick

(g) forward stepping (h) rightward stepping (i) leftward stepping

(j) low punch (k) high punch (l) middle punch

Figure 6: *Sample frames generated by FootSee. For each image, the lower left part shows the input foot pressure image. The lower right part shows the corresponding pressure image of the best matching motion. The upper right part is the controlled avatar. For comparison purpose, we render the motion of the user at the upper left part in sync with FootSee.*

Also there will always be a trade-off between responsiveness and accuracy. Some recognition mistakes can be easily resolved if we allow longer latency, and make decisions using more information than onsets. Motion editing faces the same trade-off as well. If we know exactly where we are going before we step, the IK result will be better. In our situation, the leg may already be half way in the air before we see the stepping error.

Where possible, we selected simple, fast, and easy to implement algorithms, despite the availability of more sophisticated ones (as we have already pointed out when describing our solutions). For instance, our motion recognition algorithm needs negligible training time and a very small amount of training data. It is interesting that the meta-analysis of [27] suggests that more novel and complex recognition algorithms do not necessarily work better. Our goal was to quickly develop a practical system to test this animating approach, and to add complexity only when simple techniques failed. As a result, our technique can be implemented by anyone with a similar foot pressure sensor pad.

We would like to improve some of the current limitations in future work:

- The current foot tracking and recognition (Section 4.2) probably only works for fixed foot direction. If the user changes his facing direction during database capture and online control, we may need to estimate the orientation of the feet from the pressure data as well. By carefully exploiting the foot spatial and temporal coherence, this may or may not be a hard problem.
- Although currently FootSee is neither truly real-time nor highly accurate, it provides a solid point of departure for both types of applications. We would like to try higher pressure sensor density, higher pressure range and resolution, and higher capture rate, which should improve both latency and accuracy. We would also like to incorporate better (but not much slower) inference algorithms, and modelling of behavior dynamics to recognize motions and arbitrate ambiguities, for applications that can afford longer latency.
- We would like to combine other non-intrusive sensing techniques with FootSee. For example, a camera can probably see the arm motions better than FootSee, while FootSee can resolve many ambiguities that a camera cannot.

Despite these current limitations, we believe FootSee is a promising technique that provides an intuitive and easy-to-use interactive interface for many types of applications, including interactive video games, avatar control, sports training, and performance-driven animation.

Acknowledgements

The authors would like to thank the anonymous reviewers for their suggestions; Alias/Wavefront for software donations; Jesse Hoey for the discussion on recognition problems; Etienne Lyard for being our motion capture subject; Shinjiro Sueda for supporting the hardware rendering package; Paul Kry for proofreading and other help he provided. This work was supported in part by NSF grant EIA-0215887 and a UBC University Graduate Fellowship.

Appendix A: Derivation of α

Denote the hip position as $\mathbf{h}$, the ankle position as $\mathbf{a}$. As illustrated in Figure 5(a), the hip vertical translation t is calculated as follows

$$t = \|\mathbf{h}_y - \mathbf{a}_y\| - \sqrt{\|\mathbf{e}\|^2 - \|\mathbf{d}\|^2}$$

Equate the left hip translation with the right hip translation, we can get an equation of this form

$$\sqrt{a\alpha^2 + b_1\alpha + c_1} - \sqrt{a\alpha^2 + b_2\alpha + c_2} + d = 0$$

The above equation can be deduced to a quadratic equation

$$A\alpha^2 + B\alpha + C = 0$$

We pick the solution in $[0,1]$. In case the two solutions are both in this range, we pick the one closer to 0.5. There are cases when both solutions are out of the range $[0,1]$, and the resulting animation is often weird. This is because the needed correction is too large (this often means a wrong match just happened), while the kinematics is satisfied, other constraints such as center of mass should stay in the feet support polygon is violated. In this case, we reduce the correction to be made in this step recursively until we get a reasonable solution for α, and the stepping error is distributed into IKs for the following steps as well, rather than trying to correct the error in one unrealistic step.

References

1. R.M. Alexander. *The Human Machine*. Columbia University Press, 1992. 1

2. O. Arikan and D.A. Forsyth. Interactive motion generation from examples. In *SIGGRAPH 2002 Conference Proceedings*, pages 483–490, 2002. 2, 5

3. S.O. Belkasim, M. Shridhar, and M. Ahmadi. Pattern recognition with moment invariants: a comparative study. *Pattern Recognition*, 24(12):1117–1138, 1991. 3

4. C. Belland, J.W. Davis, B. Helfer, S. Varadarajan, M. Gleicher, and S. King. *Motion Capture: Pipeline, Applications, and Use*. 2002. 1

5. B. Bodenheimer, C. Rose, S. Rosenthal, and J. Pella. The process of motion capture: Dealing with the data. In *Computer Animation and Simulation '97*, pages 3–18. Eurographics, 1997. 1

6. A. Bottino and A. Laurentini. Experimenting with non-intrusive motion capture in a virtual environment. *The Visual Computer*, 17(1):14–29, 2001. 2

7. A. Bruderlin and L. Williams. Motion signal processing. In *SIGGRAPH 95 Conference Proceedings*, pages 97–104, 1995. 5, 7

8. M. Cavazza, R. Earnshaw, N. Magnenat-Thalmann, and D. Thalmann. Survey: Motion control of virtual humans. *IEEE Computer Graphics and Applications*, 18(5):24–31, 1998. 1

9. P.T. Chua, R. Crivella, B. Daly, N. Hu, R. Schaaf, D. Ventura, T. Camill, J.K. Hodgins, and R. Pausch. Training for physical tasks in virtual environments: Tai chi. In *Proceedings of IEEE Virtual Reality 2003*, pages 87–96. IEEE, 2003. 1

10. P. Faloutsos, M. van de Panne, and D. Terzopoulos. Composable controllers for physics-based character animation. In *Proceedings of SIGGRAPH 2001*, pages 251–260, 2001. 2

11. T. Harada, T. Sato, and T. Mori. Pressure distribution image based human motion tracking using skelton and surface integration model. *Proceedings of the 2001 IEEE International Conference on Robotics and Automation*, 5:3201–3207, 2001. 2

12. T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning:Data Mining, Inference, and Prediction*. Springer-Verlag, 2001. 3

13. J.K. Hodgins, J.F. O'Brien, and J. Tumblin. Perception of human motion with different geometric models. *IEEE Transactions on Visualization and Computer Graphics*, 4(4):307–316, 1998. 1

14. J.K. Hodgins, W.L. Wooten, D.C. Brogan, and J.F. O'Brien. Animating human athletics. In *Proceedings of SIGGRAPH 1995*, pages 71–78, 1995. 2

15. Kaydara. *http://www.kaydara.com/*. Kaydara Inc., 2002. 2

16. L. Kovar, M. Gleicher, and F. Pighin. Motion graphs. In *SIGGRAPH 2002 Conference Proceedings*, pages 473–482, 2002. 2, 5

17. L. Kovar, M. Gleicher, and J. Schreiner. Footskate cleanup for motion capture editing. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 97–104, 2002. 6

18. J. Lee, J. Chai, P.S.A. Reitsma, J.K. Hodgins, and N.S. Pollard. Interactive control of avatars animated with human motion data. *ACM Transactions on Graphics*, 21(3):491–500, 2002. 1, 2, 5, 6

19. J. Lee and S.Y. Shin. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of SIGGRAPH 1999*, pages 39–48, 1999. 6, 7

20. Y. Li, T. Wang, and H.Y. Shum. Motion texture: A two-level statistical model for character motion synthesis. In *SIGGRAPH 2002 Conference Proceedings*, pages 465–471, 2002. 2

21. V. Medved. *Measurement of Human Locomotion*. CRC Press, 2000. 1

22. T.B. Moeslund and E. Granum. A survey of computer vision-based human motion capture. *Computer Vision and Image Understanding: CVIU*, 81(3):231–268, 2001. 2

23. K.L. Moore and A.F. Dalley. *Clinically Oriented Anatomy*. Lippincott Williams & Wilkins, fourth edition, 1999. 5

24. B.M. Nigg and W. Herzog. *Biomechanics of the Musculo-Skeletal System*. John Wiley & Sons Ltd, 1994. 1

25. S. Oore, D. Terzopoulos, and G. Hinton. A desktop input device and interface for interactive 3d character animation. In *Graphics Interface 2002 Conference Proceedings*, pages 133–140, 2002. 1

26. M.J. Park, M.G. Choi, and S.Y. Shin. Human motion reconstruction from inter-frame feature correspondences of a single video stream using a motion library. In *ACM SIGGRAPH Symposium on Computer Animation*, pages 113–120, 2002. 2

27. P.J. Phillips and E.M. Newton. Meta-analysis of face recognition algorithms. In *Proceedings of IEEE Automatic Face and Gesture Recognition*, pages 235–241, 2002. 9

28. P.S.A. Reitsma and N.S. Pollard. Perceptual metrics for character animation: Sensitivity to errors in ballistic motion. In *SIGGRAPH 2003 Conference Proceedings*, 2003. 1

29. J. Rose and J.G. Gamble. *Human Walking*. Williams & Wilkins, second edition, 1994. 5

30. N. Torkos and M. van de Panne. Footprint–based quadruped motion synthesis. In *Graphics Interface*, pages 151–160, 1998. 1

31. M.S. Tyler-Whittle. *Gait analysis: an introduction*. Oxford ; Boston : Butterworth-Heinemann, third edition, 2002. 1

32. M. van de Panne. From footprints to animation. *Computer Graphics Forum*, 16(4):211–224, 1997. 1

33. Vicon. *http://www.vicon.com/*. Vicon Motion Systems Ltd, 2002. 2

34. J.M. Winters and P.E. Crago, editors. *Biomechanics and Neural Control of Posture and Movement*. Springer-Verlag, 2000. 1

35. XSensor. *http://www.xsensor.com/*. XSensor Technology Corporation, 2002. 2