Easy Foot Plant

Li, Jun; Hao, Pengwei

ISSN 1470-5559

Easy Foot Plant

Jun Li & Pengwei Hao

Queen Mary
University of London
Department of Computer Science

RR-06-10

November 2006

Easy Foot Plant

Jun Li
Department of Computer Science
Queen Mary University of London
London E1 4NS, UK
junjy@dcs.qmul.ac.uk

Pengwei Hao
Department of Computer Science
Queen Mary University of London
London E1 4NS, UK
phao@dcs.qmul.ac.uk

Abstract

In generation of character animation, a common problem is that the character's feet move when they should keep fixed. Efficient algorithm for planting feet is useful for both pre-process of raw motion data and post-process for synthetic animation. There are plenty of methods on motion editing, including kinematics approaches and dynamics ones. Some of them are applicable for the foot-plant problem.

In this paper, we present a simple scheme especially for foot plant. We introduce an new inverse kinematics solver for a popular character leg model. We have also developed the analytical solution for it and address the problem of finding optimal root displacement in a simple but efficient way. Our algorithm can be used online as well.

1 Introduction

A character is in the *foot-plants* when one or both of its feet keeps still during the animation. Foot-plants often appear in real human motion, especially in human locomotion, and people are sensitive to this common phenomenon. Therefore it is important for realistic animation to deal with foot-plant correctly.

Imperfect motion data is quite usual. For example, the motion data may be mixed with noise. In such a case the data is still useful, but needs to be preprocessed to fix the feet when necessary. Motion retargetting[1], motion editing, including warping[2] and signal processing algorithms[3], can also damage the foot-plant. Furthermore, the results of many motion synthesis algorithms[4] need to be adjusted so that the foot-plants are implemented.

In this paper, we introduce a simple scheme for foot-plant for motion data. Our scheme resembles the algorithm in [5] conceptually. However, we present a new inverse kinematics(IK) solver which uses one less degree of freedom to pose the foot at desired position and orientation. Thus it leaves one DOF for adapting the character's feet to uneven ground or other adjustment. Because of the lack of the heel joint in the skeleton we used, we plant the feet by determining and adjusting their ankle and ball joints. We solve the IK problem analytically. It is as fast as the analytical algorithm in [5], but we introduce it in a more intuitive and simpler way. We also detect when the foot-plant occur automatically.

In the remainder of this paper, after a brief review of the related work, we present our scheme in detail: the detection of the need for foot-plant, the decision of stance, the displacement of the pevis position, and the computation of the inverse kinmatics solution for the adjustment. They are followed by a demonstration of the result and some further discussion.

2 Related Work

Inverse kinematics (IK) means to determine the values for every *degree of freedom* (DOF) of a skeleton, such that the constraints on the end-effectors of the skeleton are satisfied.

Due to complexity of human skeleton, the IK problems are normally highly non-linear and ill-conditioned [6]. Thus the existence of the analytical solution should not be expected for general cases. There is rich literature on solving IK problem numerically [7] [8](and references therein).

However, it has been observed that in a useful special case, *human-arm-like (HAL) link*, the analytical solution exists[9]. And the details of the analytical solution were presented in [10], which is adopted by [11], [12] and [5]. Grochow et al.[13] proposed an IK solver which can choose results from the solution space based on learning.

Approaches that are more related to ours are [5] and [14]. Kovar et al.[5] presented a simple on-line method for footskate removal. Gao et al.[14] adapted it for retargetting purposes. We shall discuss them later.

Many researchers have also proposed various methods on modifying the motion signal for different applications[3] [2] [15] [16]. Gleicher gave a method to apply motion signals from one character to another similar one without disturbing the constraints on the character [1][17].

A foot plant algorithm will find its applications as the pre- and post-process steps more easily while methods for animation synthesis powered by motion capture database are becoming prevailing [4] [18] [19] [20].

3 The Problem

We use the character skeleton and the motion data from [21]. For the foot plant problem, we are interested in the lower part of the body. Our skeleton model is shown in Fig.1. We ignore the toes, because the toe bones are very short compared to other bones and they can be dealt with similar scheme when needed.

The motion data is a multidimensional function of time

$$M(t) = (\vec{P}^R(t), R_X^{J_1}(t), R_Y^{J_1}(t), R_Z^{J_1}(t), \ldots, R_X^{J_N}(t), R_Z^{J_N}(t)) \tag{1}$$

where $J_i,\ i = 1, \ldots, N$ are the joints, such as left hip, left knee, etc. The DOFs at each joints may not be the same. For example, there are R_X^{Hip}, R_Y^{Hip} and R_Z^{Hip} for hip, but only R_X^{Ankle} and R_Z^{Hip} for ankle.

Conceptually, our scheme to plant a foot is similar to [5], but for the kernel IK solver, we develop it for a different type of kinematics link in [21]. Gao et al. [14] also used that database, but they didn't address the structural difference between their character skeleton model and Kovar's. We also solve the IK problem in a more intuitive way from [10] which is adopted in [5].

4 Footplant decision

The first step is to detect the need for the foot-plants. We use the simple scheme in [14], therein [22] is cited. Ikemoto et al. presented a training based algorithm to tell whether the foot should be kept fixed automatically. However, because the feet keep fixed in a foot-plant, the velocity threshold is a simple and efficient way to decide a joint's *contact flag* at every frame, if the noise is properly dealt with. In [22], the minimum duration of contact is also considered. However, because of the existence

of the noise, frequently flipping of the *contact flag* may occur. It is needed to consider not only the minimum duration of contact but also that of non-contact. Thus we filter the *contact flag*s generated by thresholding. For joint J (J is ankle or ball), the flags of the neighboring frames are checked to decide whether J is fixed. We take L_0 frames on both sides, and let those $2L_0 + 1$ frames vote whether the current frame is fixed.

In practice, we have a simple and on-line implementation for the voting. After thresholding, we can convolute the *contact flag* signal with a mask $1/(2L_0 + 1)\underbrace{[1, \ldots, 1]}_{2L_0+1}$ If the response is greater than 0.5 at frame F_i, the joint J is fixed at F_i.

Fig.2 shows the determination of the foot-plants for an ankle in a motion segment. Note that at the frame F_{853}, the noise makes the speed increase suddenly. The 'filter' keeps the judgment from unnecessary flipping. (Note the height of that joint, at the 'faked' speed peak, the height does not change much.)

5 Constrained Foot Position

For those foot-plant frames, it is needed to find the desired foot configuration. For our model, the foot is fully configured by the positions of the ankle and the ball joints.

If a joint J (J is A for ankle and B for ball) should be fixed at frame F_i, our algorithm decides the desired position of J according to its status and positions at F_i and F_{i-1} and those of the other foot joint.

When J is constrained at F_{i-1}, it simply takes the same position as at F_{i-1}. Care must be taken when a constraint is being applied to J, say, J is free at F_{i-1} and constrained at F_i, because we now need the constrained position of J not only for the current frame, but also for those following

constrained frames in a sequence. There are two requirement of the position of J to be decided:

- The change made to every affected frame should be small.
- The length of foot should be kept.

To address the first requirement, we average the positions of J over F_i and the following L_1 frames. L_1 is the maximum frames we take. We stop averaging after encountering the first frame at which J is free. For the second requirement, we check the the other foot joint J'. If J' is also constrained, it is required that the constrained position of J and that of J' must keep the correct foot length. Instead of averaging the positions of J over the following frames, we compute the average *foot direction*($P_{Bj} - P_{Aj}$) on a sphere over those frames j. If J' is fixed and J is free at F_{i-1}, let its position be kept, and compute the position for J using the average foot direction and that of $P_{J'}$. If both joints are free at F_{i-1}, we firstly fix the ball position by averaging the positions over the following frames, then figure out the position of ankle by finding the average foot direction. See Algorithm 1 for details.

Discontinuity may be introduced into the motion at the switch of constraints. Therefore a post-process of smoothing is necessary. We adopt the terms in [5]: A frame is called *single constrained*, if either ankle or ball is constrained; a frame is called *double constrained*, if both of them are constrained. For each single constrained frame, we look forward and backward for a double constrained frame. If found, we smooth the foot direction vector according to the frame distance between the current frame and the found one. The details are given in Algorithm 2. There are still discontinuity at the constrained(single or double) and free frames. However, because we can adjust the DOFs of the free frames directly to ensure the result motion is smooth, it is not necessary to smooth our the discontinuity of the desired foot joint positions here.

Fig.3 shows the procedure to find the desired positions of an ankle. The dashdot line represents the

original height of the ankle. The dashed one is the desired position for ankle after running Algorithm 1. And the solid line is the final choice. The dotted line shows the contact flag.

6 Root Position Displacement

Having obtained the desired positions of the foot joints decided, we are to find proper values for the leg DOFs to meet the requirements. However, the requirements are not necessarily achievable. Having fixed the hip position, the reachable positions of the ankle and the ball are limited. If any required position is out of range, we must change the position of the corresponding hip. According to [12], the quality of reality of a motion affected less by adjusting the position of root(pelvis for most skeleton) than by changing that of foot joints. Thus they adjusted the root position by projecting the root onto a sphere on which the required position can be reached. However, the projection to different spheres (one for left leg and the other for right) at consequent frames can cause discontinuity. In [5](therein [12] cited), an expedient scheme is adopted, which extends the length of femur and tibia instead of finding more suitable positions for the root. This increases both the size and the structural complexity of the motion data. Moreover, the varying length of a bone is not friendly to many skinning and rendering algorithms.

Our scheme for replacing the root is also simple and ensures the smoothness of its path. There are four requirements for the root path:

- Root should be placed so that both hips are near from their *easy positions*. *Easy position* means that if the hip is put there, the ankle will be at its desired position. Say, if the current positions of the hip and the ankle are P^H and P^A respectively, and the desired position of the ankle is $\tilde{P}^A$, the easy position for the hip is $\tilde{P}^H = P^H + \tilde{P}^A - P^A$.

- Root should be within both *reachable spheres* of the left and right. A *reachable sphere* is a sphere centered at the desired position of ankle with the radius of maximum leg length. The efficiency of adjustment of the knee angle to lengthen the leg vector ($P^A - P^H$) decreases dramatically when the length exceeds 95% of the length[5]. Thus we need to control the length of the leg to avoid knee-popping.

- The change should be as small as possible.

- The change of the root velocity should be as small as possible.

We represent the first two requirements by two objective functions on the new positions of the root P^R_{new}

$$O_H(P^R_{new}) = \sum_{i=1}^{2} w_{H_i} \|\tilde{P}^{H_i} - (P^{H_i} + P^R_{new} - P^R)\|^2 \tag{2}$$

$$O_L(P^R_{new}) = \sum_{i=1}^{2} w_{L_i} \|P^H_N - \tilde{P}^{A_i}\|^2 \tag{3}$$

It is easy to see that Eq.(2) is for the easy positions of the hips and Eq.(3) is for the lengths of the legs.

The final objective function is the sum

$$O(P^R_{new}) = O_H(P^R_{new}) + O_L(P^R_{new}) \tag{4}$$

Eq.(4) is quite easy to optimize analytically

$$\tilde{P}^R_{new} = P^R + \delta^R \tag{5}$$

where

$$\delta^R = \sum_{i=1}^{2} \frac{1}{W}(w_{H_i}(\tilde{P}^{H_i} - P^{H_i}) + w_{L_i}(\tilde{P}^{A_i} - P^{H_i})) \tag{6}$$

$$W = \sum_{i=1}^{2}(w_{H_i} + w_{L_i})$$

In practice, the first two objectives work well. However, it is fairly easy to add the other two.

The ratio between the weights for leg length (w_L) and easy positions (w_H) are determined by the user. The ratio between left and right weights are determined by the constraint status. A leg is called constrained if either its ankle or its ball is constrained. If both legs are constrained, the weights are distributed to both sides equally, $w_1 = w_2 = 1$. (Here we use the symbol of w_1 and w_2 to describe the relationship between both (w_{H_1}, w_{H_2}) and (w_{L_1}, w_{L_2}).) If one leg is free and the other is constrained, say, left leg is constrained, and right one is free, we give all the weight to the constrained leg, say, $w_1 = 2$ and $w_2 = 0$. However, as the case of determining the desired position of foot, for those single constrained frames, we try to find a frame in which the other leg is constrained by searching forward and backward for L_3 frames. If one frame is found in either direction, we assign interpolated weights for the frames in-between, such that the weights are smooth and δ^R is determined by them in Eq.(6).

7 Inverse Kinematics Solver

Having obtained the desired positions for all related joints at every frame, we compute the values of the DOFs to meet the position requirements. A common model of a human leg often consists of three joints, the hip joint and the ankle joint have 3 DOFs, and the knee joint has one DOF. This kind of links are widely used, and called *human-arm-like* (HAL) link (hip to shoulder, knee to elbow, and ankle to wrist, hereby we use the names of the leg joints). The typical inverse kinematics problem for such kind

of links is to find the 7 rotational DOFs (3 for hip, 3 for ankle and 1 for knee), such that the ankle position and the foot orientation satisfy the given requirements (6 constraints). There exists analytical solution to the inverse kinematics problem for an HAL link [10][11].

In practice, a varied "arm-like" model is common [21]. And for the simplified leg model, the foot configuration is determined only by the direction of the ankle-to-ball vector, which needs two DOFs rather than three at the ankle joint. Thus our new version of the inverse kinematics problem is to determine 6 rotational DOFs to satisfy 5 constraints. Compared to the original link, our inverse kinematics solver leaves one DOF for the character to fit uneven ground, or to adjust the body posture. It determines the required rotational parameters for hip, knee, and ankle such that the ankle position and the ankle-to-ball vector are configured to the desired position and direction.

7.1 Human-Arm-Like Link Model

A typical HAL link consists of a 3-DOF hip joint, a 1-DOF knee joint, and a 3-DOF ankle joint. In our model, the ankle is a 2-DOF joint: pitch and yaw. The knee rotational axis is intuitive. The HAL link model is shown in Fig.4a. The desired position of the ankle and direction of the foot (ankle-to-ball) vector are also shown in Fig.5.

7.2 Find knee angle

As some authors have addressed the HAL inverse kinematics problem ([10] [11] [5]), we compute the *knee angle* firstly, because it determines the length between the hip and the ankle independently.

Those previous approaches need project the bones of the leg onto the plane perpendicular to the vector $\tilde{P}^A - P^H$ linking the hip P^H and the desired ankle $\tilde{P}^A$. However, we determine the knee angle ϕ_K in a more intuitive way, the algorithm need only the desired length of the leg, $\|P^H - \tilde{P}^A\|$, without

the specific position requirement of the ankle. This makes the solver more useful while being applied to certain cases. For example, when the animator needs the character to shorten its legs to anticipate a collision.

Firstly, we project the femur and the tibia to the plane π perpendicular to the axis (V_{axis} in Fig.5(the plane on which the tibia rotates). In the figure, the current positions of the two ends of the HAL link are A and B respectively, the projected positions are A_P and B_P(Note that this is the general case. For normal human being, A is on the plane and identical to A_P, so does B.), and the knee(rotational joint) position is O. So the relationship between the length of the leg $\|\vec{AB}\|$ and the knee angle ϕ_K can be described by the following equations

$$\|\vec{AB}\| = \sqrt{\|\vec{A_P B_P}\|^2 + (\|\vec{AA_P}\| + \|\vec{BB_P}\|)^2} \tag{7}$$

$$\|\vec{A_P B_P}\|^2 = \|\vec{OA_P}\|^2 + \|\vec{OB_P}\|^2 - 2\|\vec{OA_P}\|\|\vec{OB_P}\| \cos\phi_K \tag{8}$$

Note that the distances between the two ends of the HAL link and the plane π keep invariant when the knee angle changes. And so do the lengths of the projections of the two segments. Thus we can obtain those constants from the current leg

$$l_{A\perp} = \|\vec{AA_P}\|$$

$$l_{A\|} = \|\vec{OA_P}\|$$

$$l_{B\perp} = \|\vec{BB_P}\|$$

$$l_{B\|} = \|\vec{OB_P}\|$$

If the desired length of the leg is l_D, the knee angle can be solved as

$$\phi_K = \text{arccos}(\frac{l_D^2 - l_{A\perp}^2 - l_{B\perp}^2 - l_{A\|}^2 - l_{B\|}^2}{2l_{A\|}l_{B\|}}) \tag{9}$$

Fig.4b shows the modification of the length of the leg.

7.3 Find hip angles

As in [5], we define $Rot(\vec{a}, \vec{b})$ as the minimum rotation that aligns the vector $\vec{a}$ to $\vec{b}$. Then the hip rotation $R^H = Rot(P^A - P^H, \tilde{P}^A - P^H)$. As in Fig.4, P^H and P^A are the positions of hip and ankle respectively, and $\tilde{P}^A$ is the desired position of the ankle. At last, we convert R^H into Euler angles and choose the set of angles closest to the original one. Fig.4c shows the result of rotating the hip such that the ankle position meets the requirement.

7.4 To find ankle angles

The last step is to decide the two DOFs at the ankle, so that the foot direction makes the ball at the desired position. Without losing generality, let the two DOFs be rotations about X and Z axes consequently. If the foot vector in ankle fixed(local) coordinate system is $\vec{v_{f0}} = (v_1, v_2, v_3)^T$. After applying the rotation about X axis by ϕ_A and then the other one about Z axis by ψ_A, it becomes(in the same coordinate system)

$$\vec{v_{f1}} = \begin{pmatrix} v_1 \cos\psi_A - v_2 \sin\psi_A \\ v_1 \cos\phi_A \sin\psi_A + v_2 \cos\phi_A \cos\psi_A - v_3 \sin\phi_A \\ v_1 \sin\phi_A \sin\psi_A + v_2 \sin\phi_A \cos\psi_A + v_3 \cos\phi_A \end{pmatrix} \tag{10}$$

If the desired position of the ball in the local coordinate system is $P_B = (B_X, B_Y, B_Z)^T$, we choose ϕ_A and ψ_A so that the components of $\vec{v_{f1}}$ equal those of P_B. (Note that the foot vector starts from the origin point in the local coordinate system)

We solve the two angles as

$$
\begin{aligned}
\psi_A &= \arcsin\frac{A_\psi - B_\psi}{C_\psi} \\
&\text{or} \\
&= \arcsin\frac{A_\psi + B_\psi}{C_\psi} \qquad (11) \\
A_\psi &= A(B_X, v_1, v_2) \\
B_\psi &= B(B_X, v_1, v_2) \\
C_\psi &= B(B_X, v_1, v_2)
\end{aligned}
$$

and

$$
\begin{aligned}
\phi_A &= \arcsin\frac{A_\phi - B_\phi}{C_\phi} \\
&\text{or} \\
&= \arcsin\frac{A_\phi + B_\phi}{C_\phi} \qquad (12) \\
A_\phi &= A(B_Z, v_3, -v_1 \sin\psi - v_2 \cos\psi) \\
B_\phi &= B(B_Z, v_3, -v_1 \sin\psi - v_2 \cos\psi) \\
C_\phi &= C(B_Z, v_3, -v_1 \sin\psi - v_2 \cos\psi)
\end{aligned}
$$

where

$$
\begin{aligned}
A(r, s, t) &= -rt \\
B(r, s, t) &= s\sqrt{s^2 + t^2 - r^2} \\
C(r, s, t) &= s^2 + t^2
\end{aligned}
$$

The details are given in Appendix A. Note that, according to Eq.(11) and Eq.(12) there can be four sets of feasible solutions. Furthermore, because the trigonometrical functions are periodic, for each

set of solutions, we can find infinite periodical solutions. When the algorithm is applied in practice, we choose the solution that makes minimum adjustment to the original rotation and within the rotation limits for the ankle. In Fig.4d, we can see all the joints on the leg meeting the requirement.

8 Results

Here we present some results of our algorithm. In our experiment, we let all L_i be 6 frames (approximate one twentieth second for data on [21]), except L_1, which is used to find average contact position for joints of foot. We let L_1 be 60 frames, which covers approximately a half second in our data.

In Fig.6, we illustrate the results of our algorithm. Fig.6a and Fig.6b represent a segment of the motion, during which the character should keep a stance. We can see that the foot plant algorithm removes the noise in the original data (Fig.6a). Fig.6c and Fig.6d represent the process of taking-off of a foot. It shows that the our foot plant eliminates the floating of the foot before its taking-off.

Fig.7 shows the height of the left ball before and after the foot plant. The motion data is a segment of consequent jumping. Note that the height of the contact position of the ball is not the same before and after a jump. We accept the height of the *local* ground, which is inferred from the neighbour frames.

9 Discussion

In this paper, we solve the problem of planting foot for imperfect motion data. The algorithm is fast, robust, easy to implement and suitable for online application. We also introduce the analytical IK solver to 6-DOF-for-5-constraints link for human arms or legs. The determination of knee angle in our solver is superior to previous ones. Another important different point between our method and the previous

approaches addressing the similar problem is that we use greater displacement of root position rather than varying leg length to avoid the knee popping while solving the IK problem, because we think it is cumbersome to introduce varying bone length to motion data. We use the similar scheme to ensure the smoothness of our result motion as Kovar's method, but we do smooth interpolation for foot-vector rather than using projection.

There are several problems left open. One is that at the stage of choosing foot-plant, we do not project those positions to the virtual ground. Because we believe the actual ground is also unknown, we *infer* the *local ground* at each foot-plant. This also makes it possible to process motion data in which the two feet are not at the same plane. Another is that our algorithm for root position arrangement cannot guarantee a *reachable* root position. The philosophy behind is that our main purpose is to recover faulty raw or synthesis data. For this application, the ideal global displacement should not be too big, thus the optimization scheme is adequate. However, an objective function with non-linear leg length cost, for example $e^{\frac{L-\alpha L_{max}}{\gamma L_{max}(1-\alpha)}}$, would guarantee the reachable root at the price of complexity of the solution expression or even non-existence of analytical solution. The third problem is that in fact we can find out the heel for the model. Thus in our future application, we can deal with the heel joint as well as the ball. Fig.8 shows us a virtual heel.

A Solution to the trigonometrical equation

We give the solution to Eq.(10) in Eq.(11) and Eq.(12). Because for trigonometrical equation with the form of

$$r = s\cos\psi - t\sin\psi \tag{13}$$

we can rewrite it as

$$r + t \sin \psi = s \cos \psi \tag{14}$$

and replace both sides with their squares. Then we have

$$r^2 + t^2 \sin^2 \psi + 2rt \sin \psi = s^2 (1 - \sin^2 \psi) \tag{15}$$

Solve the normal quadratic equation, we have $\sin \psi$. Read the first component in Eq.(10) in the form of Eq.(13), thus we have Eq.(11).

About Eq.(12), we repeat what has been done for the third component in Eq.(10), and simplify it with care, we find it as well has the form of Eq.(13), with

$$r = B_Z$$

$$s = v_3$$

$$t = -v_1 \sin \psi - v_2 \cos \psi$$

References

[1] Michael Gleicher. Retargetting motion to new characters. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 33–42, New York, NY, USA, 1998. ACM Press.

[2] Andrew Witkin and Zoran Popovic. Motion warping. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 105–108, New York, NY, USA, 1995. ACM Press.

[3] Armin Bruderlin and Lance Williams. Motion signal processing. In *SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, pages 97–104, New York, NY, USA, 1995. ACM Press.

[4] Lucas Kovar, Michael Gleicher, and Fred Pighin. Motion graphs. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 473–482, New York, NY, USA, 2002. ACM Press.

[5] Lucas Kovar, John Schreiner, and Michael Gleicher. Footskate cleanup for motion capture editing. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 97–104, New York, NY, USA, 2002. ACM Press.

[6] Anthony A. Maciejewski. Motion simulation: Dealing with the ill-conditioned equations of motion for articulated figures. *IEEE Comput. Graph. Appl.*, 10(3):63–71, 1990.

[7] C. Weiman. Inverse kinematics and geometric constriants for articulated figure manipulation. Master's thesis, Simon Fraser University, 1989.

[8] Jianmin Zhao and Norman I. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures.

[9] James U. Korein and Norman I. Badler. Techniques for generating the goal-directed motion of articulated structures. *IEEE Comput. Graph. Appl.*, 2(9):71–74, 76–81, November 1982.

[10] Deepak Tolani, Ambarish Goswami, and Norman I. Badler. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graph. Models Image Process.*, 62(5):353–388, 2000.

[11] Jehee Lee and Sung Yong Shin. A hierarchical approach to interactive motion editing for human-like figures. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 39–48, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[12] Hyun Joon Shin, Jehee Lee, Sung Yong Shin, and Michael Gleicher. Computer puppetry: An importance-based approach. *ACM Trans. Graph.*, 20(2):67–94, 2001.

[13] Keith Grochow, Steven L. Martin, Aaron Hertzmann, and Zoran Popović. Style-based inverse kinematics. *ACM Trans. Graph.*, 23(3):522–531, 2004.

[14] Yan Gao, Lizhuang Ma, Zhihua Chen, and Xiaomao Wu. Motion normalization: the preprocess of motion data. In *VRST '05: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 253–256, New York, NY, USA, 2005. ACM Press.

[15] Charles Rose, Michael F. Cohen, and Bobby Bodenheimer. Verbs and adverbs: Multidimensional motion interpolation. *IEEE Comput. Graph. Appl.*, 18(5):32–40, 1998.

[16] Seyoon Tak and Hyeong-Seok Ko. A physically-based motion retargeting filter. *ACM Trans. Graph.*, 24(1):98–117, 2005.

[17] Michael Gleicher. Motion editing with spacetime constraints. In *SI3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics*, pages 139–ff., New York, NY, USA, 1997. ACM Press.

[18] Katherine Pullen and Christoph Bregler. Motion capture assisted animation: texturing and synthesis. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 501–508, New York, NY, USA, 2002. ACM Press.

[19] Jehee Lee, Jinxiang Chai, Paul S. A. Reitsma, Jessica K. Hodgins, and Nancy S. Pollard. Interactive control of avatars animated with human motion data. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 491–500, New York, NY, USA, 2002. ACM Press.

[20] Okan Arikan, David A. Forsyth, and James F. O'Brien. Motion synthesis from annotations. *ACM Trans. Graph.*, 22(3):402–408, 2003.

[21] CMU. Carnegie-mellon motion capture database.

[22] Bodik. Automatic footplant dectection inside flmoview. Technical report, Student Summer Project, 2000.

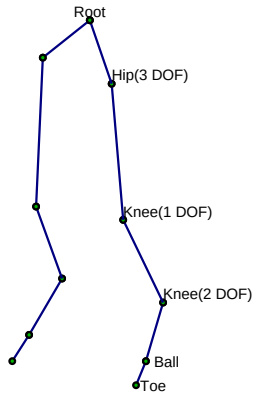


Figure 1: The lower part of the skeleton with DOFs for each joint.

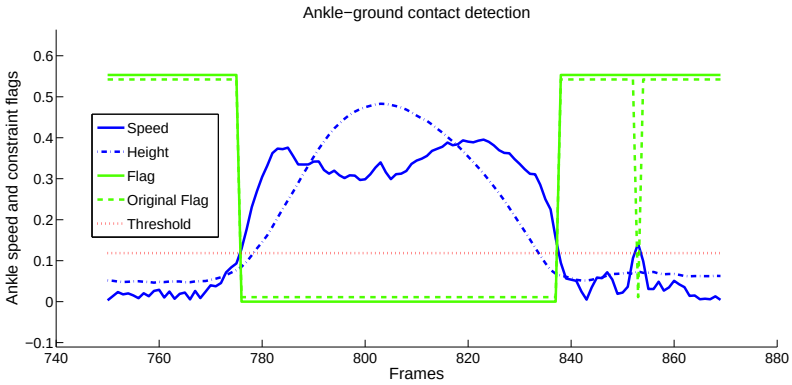


Figure 2: To determine whether the ankle is constrained.

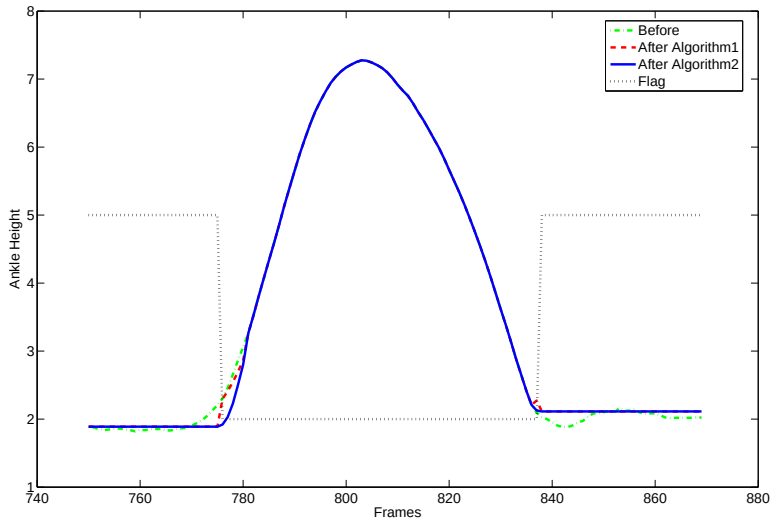
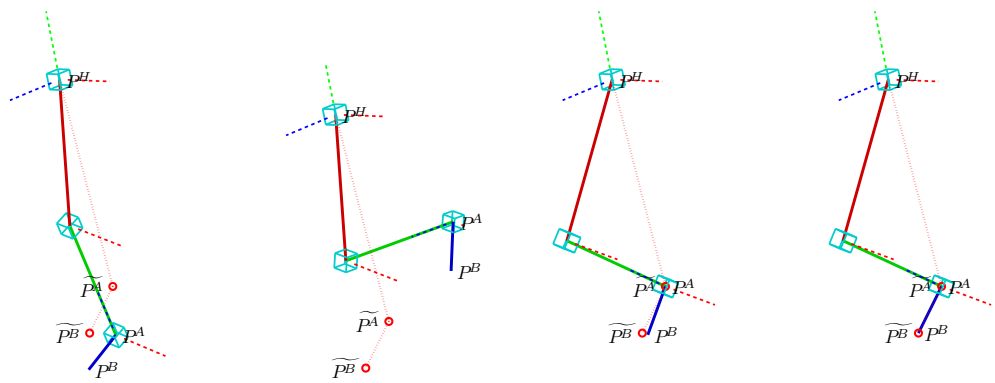


Figure 3: To find the desired positions for constrained joints.



(a) HAL link and the desired position of ankle $\tilde{P}^A$ and ball $\tilde{P}^B$

(b) After adjustment of the knee angle

(c) The result of changing parameters for hip to align the position of ankle

(d) The final adjusted link.

Figure 4: A Human-Arm-Like Link

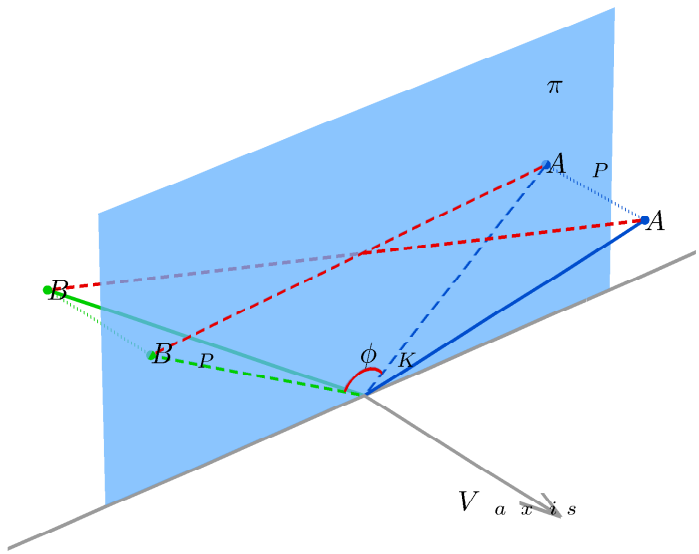
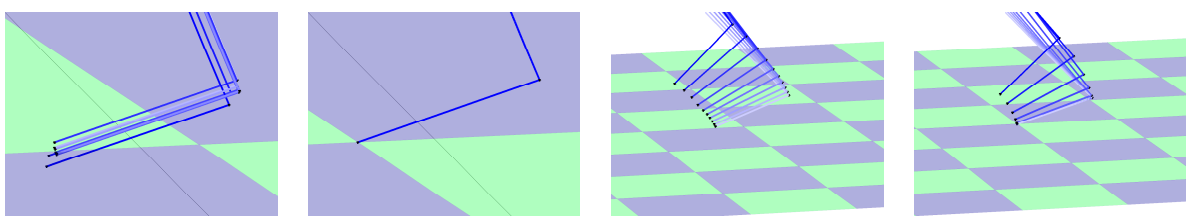


Figure 5: Computation of knee angle



(a) (b) (c) (d)

Figure 6: Result of foot plant

(a) and (c) are frames rendered from original data, while (b) and (d) are results from processed data.

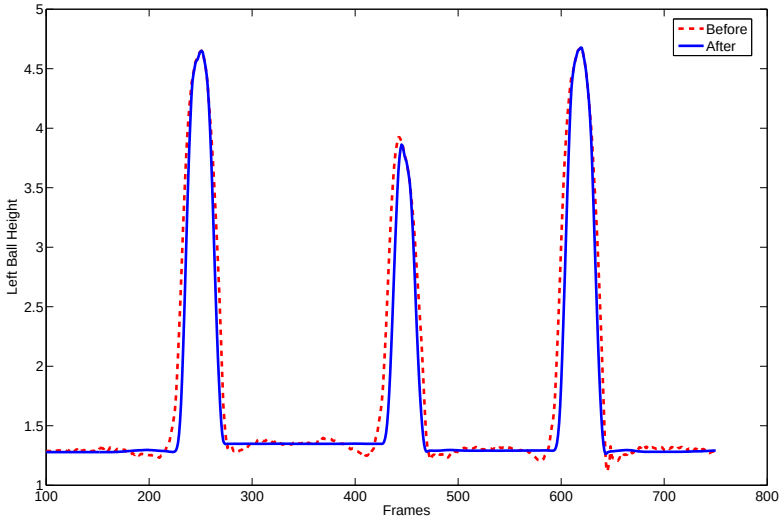


Figure 7: Height of left ball before and after foot plant.

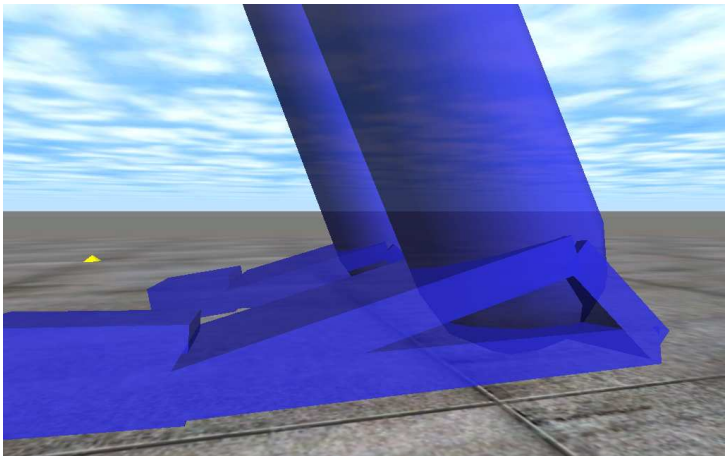


Figure 8: The inferred heel.

This figure shows the heel we infer from the motion data. We lift the character a little to show the virtual heel is approximately parallel to the ground.

Algorithm 1 To find desired positions of foot joints at frame F_i

IsFixed(J) tells whether joint J is fixed at both this frame and the previous one; BeingFixed(J) tells whether joint J is fixed at current frame and free at the previouse one. J_0 and J_1 represent two foot joints, one is ankle and the other is ball or vice versa.

$\vec{v_C} \leftarrow P_i^B - P_i^A$

if IsFixed(Ankle) And IsFixed(Ball) **then**

  $\tilde{P}_i^A \leftarrow \tilde{P}_{i-1}^A$, $\tilde{P}_i^B \leftarrow \tilde{P}_{i-1}^B$

else if IsFree(Ankle) And IsFree(Ball) **then**

  $\tilde{P}_i^A \leftarrow \tilde{P}_i^A$, $\tilde{P}_i^B \leftarrow \tilde{P}_i^B$

else if IsFixed(J_0) And IsFree(J_1) **then**

  $\tilde{P}_i^{J_0} \leftarrow \tilde{P}_{i-1}^{J_0}$, and find $\tilde{P}_i^{J_1}$ according to $\vec{v_C}$ and $\tilde{P}_i^{J_0}$

else if BeingFixed(J_0) And IsFree(J_1) **then**

  $\tilde{P}_i^{J_0} \leftarrow \frac{1}{N_F+1}\sum_{j=i}^{i+N_F} P_j^{J_0}$, where N_F equals to L_1 or let F_{i+N_F} be the last frame in a row at which J_0 is constrained.

  Find $\tilde{P}_i^{J_1}$ according to $\vec{v_C}$ and $\tilde{P}_i^{J_0}$

else if BeingFixed(J_0) And IsFixed(J_1) **then**

  $\tilde{P}_i^{J_1} \leftarrow \tilde{P}_{i-1}^{J_1}$

  $\vec{v_i} \leftarrow AverageOnSphere(P_i^B - P_i^A, \ldots, P_{i+N_F}^B - P_{i+N_F}^A)$, where N_F equals to L_1 or let F_{i+N_F} be the last frame in a row at which both ankle and ball are constrained.

  Find $\tilde{P}_i^{J_0}$ according to $\vec{v_i}$ and $\tilde{P}_i^{J_1}$

else if BeingFixed(Ankle) And BeingFixed(Ball) **then**

  $\tilde{P}_i^B \leftarrow \frac{1}{N_F^B+1}\sum_{j=i}^{i+N_F^B} P_j^B$, where N_F^B equals to L_1 or let $F_{i+N_F^B}$ be the last frame in a row at which ball is constrained.

  $\vec{v_i} \leftarrow AverageOnSphere(P_i^B - P_i^A, \ldots, P_{i+N_F^A}^B - P_{i+N_F^A}^A)$, where N_F^A equals to L_1 or let $F_{i+N_F^A}$ be the last frame in a row at which both ankle and ball are constrained.

  Find $\tilde{P}_i^A$ according to $\vec{v_i}$ and $\tilde{P}_i^B$

end if

Algorithm 2 To adjust positions of foot joints for single constrained frame F_i

$SphereInterp(t, v_0, v_1)$ is a C^2 smooth interpolation function between v_0 and v_1.

Search forward L_2 frames for double constrained frame F_{i+N_F}. If fails, $N_F \leftarrow 0$

Search backward L_2 frames for double constrained frame F_{i-N_B}. If fails, $N_B \leftarrow 0$

if $N_F > 0$ And $N_B = 0$ **then**

  $\vec{v_F} \leftarrow P^A_{i+N_F} - P^B_{i+N_F}$

  $\vec{v_C} \leftarrow P^A_i - P^B_i$

  $\vec{v} \leftarrow SphereInterp(\frac{N_F}{L_2}, \vec{v_F}, \vec{v_C})$

else if $N_F = 0$ And $N_B > 0$ **then**

  $\vec{v_B} \leftarrow P^A_{i-N_B} - P^B_{i-N_B}$

  $\vec{v_C} \leftarrow P^A_i - P^B_i$

  $\vec{v} \leftarrow SphereInterp(\frac{N_B}{L_2}, \vec{v_B}, \vec{v_C})$

else if $N_F > 0$ And $N_B > 0$ **then**

  $\vec{v_F} \leftarrow P^A_{i+N_F} - P^B_{i+N_F}$

  $\vec{v_B} \leftarrow P^A_{i-N_B} - P^B_{i-N_B}$

  $\vec{v_C} \leftarrow P^A_i - P^B_i$

  $\vec{v_1} \leftarrow SphereInterp(\frac{N_F}{L_2}, \vec{v_F}, \vec{v_C})$

  $\vec{v_2} \leftarrow SphereInterp(\frac{N_B}{L_2}, \vec{v_B}, \vec{v_C})$

  $\vec{v} \leftarrow SphereInterp(\frac{N_B}{N_B+N_F}, \vec{v_2}, \vec{v_1})$

else if $N_F = 0$ And $N_B = 0$ **then**

  No change will be made.

end if

Adjust the position of the free joint according to that of the constrained one and $\vec{v}$