박 사 학 위 논 문

실시간 동작 변형 및 대입을 위한 중요도 기반 접근

신 현 준 (申 鉉 俊)

전자전산학과 전산학전공

한국과학기술원

2002

실시간 동작 변형 및 대입을 위한 중요도 기반 접근

An importance-Based Approach for Computer Puppetry

An importance-Based Approach for Computer Puppetry

Advisor : Professor Sung Young Shin

by

Hyun Joon Shin

Department of Electrical Engineering & Computer Science

Division of Computer Science

Korea Advanced Institute of Science and Technology

A thesis submitted to the faculty of the Korea Advanced Institute of Science and Technology in partial fulfillment of the requirements for the degree of Doctor of Philosophy in the Department of Electrical Engineering & Computer Science Division of Computer Science

Taejon, Korea

2002. 12. 5

Approved by

Professor Sung Young Shin

Major Advisor

실시간 동작 변형 및 대입을 위한 중요도 기반 접근

신 현 준

위 논문은 한국과학기술원 박사 학위논문으로 학위논문심사위원회에서 심사 통과하였음.

2002년 12월 5일

심사위원장 신 성 용 (인)

심사위원 김 홍 오 (인)

심사위원 오 영 환 (인)

심사위원 원 광 연 (인)

심사위원 좌 경 룡 (인)

DCS 975194

신 현 준. Hyun Joon Shin. An importance-Based Approach for Computer Puppetry . 실시간 동작 변형 및 대입을 위한 중요도 기반 접근. Department of Electrical Engineering & Computer Science Division of Computer Science. 2002. 69p. Advisor: Prof. Sung Young Shin. Text in English.

Computer puppetry maps the movements of a performer to an animated character in real-time. In this thesis, we provide a comprehensive solution to the problem of transferring the observations of the motion capture sensors to an animated character whose size and shape may be different from the performer's. Our goal is to map as many of the *important* aspects of the motion to the target character as possible, while meeting the online, real-time demands of computer puppetry. We adopt a Kalman filter scheme that addresses motion capture noise issues in this setting. We provide the notion of importances that allow determining which aspects of the performance must be kept in the resulting motion based on interaction of the performer with its environment and self-interaction among the performer's segments. We introduce a novel inverse kinematics solver that realizes these important aspects formulated with geometric constraints within tight real-time restriction. Our approach is demonstrated by its application to broadcast television performances.

Contents

Chapter 1

Introduction

Computer puppetry [26] transforms the movements of a performer to an animated character in real-time. The immediacy of computer puppetry makes it useful for providing live performances and as a visualization tool for traditional cinematic animation. However, this immediacy creates a number of challenges, as solutions to animation issues must be handled in an online real-time manner. A computer puppetry system must capture the movements of the performer interpret the important aspects of this motion, and determine the movements required to make the character reproduce these important aspects of the performance.

The challenges of mapping a motion from the performer to the target character become more difficult when the target character is of a different size and shape than the performer [3, 5, 7, 12]. In such cases, the resulting motion of the character cannot exactly duplicate the original performer's. For example, we cannot simultaneously match the original joint angles and end-effector positions. Generally, to preserve the important aspects of the original motion we must alter the unimportant aspects of the motion. This process of adapting a motion for a new character is called retargeting [12, 19]. To date, solutions to computer puppetry issues have been

limiting: either restricting the range of puppets that can be used, or providing restrictive notions of what is important in motions. The latter implicitly limits the range of puppets since artifacts are introduced as the differences of the puppet from the performer are increased. In this thesis, we provide techniques that address the challenges of computer puppetry when the target character is different from the performer. The following major animation issues are addressed in a manner that fits within the online real-time nature of computer puppetry.

1. The sensors used to capture the performer's motion are often noisy. Therefore, we provide a filtering technique that operates in an online manner with the efficiency required to process whole body motions in real-time. We apply a Kalman filter to rotation vectors, providing an orientation smoothing technique that is more efficient than previous methods.

2. The important aspects of the original performance must be determined such that these details can be reproduced in the resulting motion. We provide the notion of importance measures that allow us to account for changing situations even when the future is unknown. To determine which aspects are to be preserved while sacrificing the others, we present importance criteria based on interactivity of the performer. We also account for the self-interactivity among the performer's segments in importance computation to obtain more realistic results.

3. The resulting pose of the target character must be computed in a way that recreates the important aspects of the original. To realize those aspects which are formulated with geometric constraints, we provide a fast inverse kinematics solver that provides the necessary real-time performance and predictability.

The proposed solutions have been used to realize a computer puppetry system that has been used successfully to create animated television broadcasts. We begin the discussion of computer puppetry by providing an overview of our approach. We examine previous solutions with respect to the issues raised in the overview. The components of the suggested approach are then detailed in Chapters 3 through 5. An analysis in Chapter 6 reviews why this approach avoids introducing unwanted artifacts such as temporal discontinuities. Some experimental results are provided to support this approach in Chapter 7. We conclude with a summary and discussion of future research directions.

Chapter 2

Overview

Computer puppetry requires the captured movements of the performer to be mapped to a target character in real-time. As shown in Figure 1, the proposed approach for online motion retargeting divides the task into phases. First, the filtering phase "cleans" the sensor data to remove artifacts of the motion capture device. The second phase examines this filtered motion and determines the importance of each of those body parts such as hands, feet, elbows, and knees which are highly likely to interact with its environment and the other body segments. The final phase computes a pose for the target character that achieves as many of the important aspects as possible while properly interacting with the environment and the segments. In this chapter, we provide an overview of these components and survey their relation ship to previous work.

2.1 Online Filtering of Orientations

In general, captured motion data are noisy. The real-time sensors required for computer puppetry are particularly problematic in this regard. However because of the dense sampling rates and signal characteristics of motion cap-

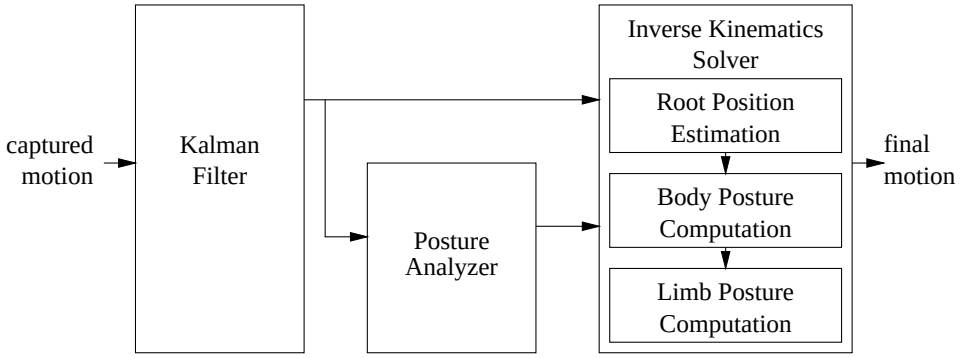


Figure 2.1: Overall structure

ture data, low-pass filtering is an effective tool to suppress noise in the captured data. This is challenging for several reasons.

1. Because computer puppetry is an online application, standard offline filters cannot be employed.

2. Because the orientation space is highly nonlinear, standard signal processing methods cannot be applied directly.

3. Because of the real-time demands, filtering should be performed on the entire body very efficiently.

A Kalman filter predicts the future values and corrects them in accordance with the actual input data to create a filtering scheme that operates in an online manner. The technique is common in online applications, and was first introduced to the graphics community by Friedmann et al. [9]. A version of Kalman filter scheme, called hierarchical Kalman filter, was also adopted to track and estimate motion of articulated figures by Jung and Wohn [15].

However, a standard Kalman filter cannot be directly applied to rotation data without accounting for nonlinearity of the orientation space. To address this problem, Welch and Bishop [28] linearized the orientation space by locally parameterizing the incremental orientation change with Euler angles, based on the result in Azarbayejani and Pentland [1] and Broida and Chellappa [6]. Because they were interested only in tracking the head motion, they were less concerned with efficiency than we are and therefore addressed only issues 1 and 2 above. In Chapter 3, we provide a modified Kalman filter. To achieve real-time performance, we locally parameterize the incremental orientation changes with rotation vectors instead of the Euler angles used in Welch and Bishop [28].

2.2 Importance Determination

The goal of computer puppetry is to create the movements of a target character based on the performer's movements. If the target character is quite different from the performer, there may not be a direct mapping. Indirect mappings are common in traditional puppetry; for example, a marionette is controlled by strings that pull on its end-effectors. Computer equivalents may create arbitrary mappings from sensor input to character parameters. For example, the Alive system from Protozoa [24] allows arbitrary Scheme functions [14] to be written to perform mapping.

Our interest is in recreating characters with human form, so the target character has equivalent degrees of freedom as a simplified model of a hu-

man performer. In this thesis, we consider characters that are articulated figures with identical connectivity, so that it is possible to transfer the captured joint angles directly to the target character. Despite this structural equivalence, the resulting motion will not match the performer's unless the character has an identical size and shape. There will be some level of mismatching even for characters that have the same size and shape as the performer, since we simplify the real human by a hierarchy of rigid bodies. One approach to performance animation, described by Molet et al. [20, 21], models the character to be as similar to the performer as possible. Bodenheimer et al. [5] presented a way to determine the segment lengths of a character that best fit the captured motion data while discarding outliers in these data by a robust estimation technique. If the segment proportions of the character are kept the same as those of the performer a motion adaptation can often be achieved by scaling the position data according to the size difference and then by translating the character globally. Restricting the proportions of the character precludes the use of stylized cartoon characters, unless we can find similarly proportioned performers.

When the virtual character and performer have different sizes and proportions, not all aspects of the motions can be preserved during mapping. At the lowest level, it is simply not possible to mimic both the locations of me end-effectors and the joint angles. A system must make choices as to which aspects of the motion should be preserved and which should be allowed to change. We call an approach to motion retargeting that makes this choice

explicitly an *importance-based* approach. Nonimportance-based approaches make implicit choices as to what should be preserved during retargeting. For example, the most naive implementation of retargeting simply transfers the parameter (joint angles and root position) values from the performer to the target character. Such a scheme implicitly selects the values of the parameters to be important and, therefore, the positions of the end-effectors to be unimportant. This is a poor choice when the character must interact with other objects in the world such as the floor.

A common approach to motion retargeting matches the end-effector positions of the character to those of the performer. Such an approach has the advantage that it preserves the interactions between the character and its environment. Badler et al. [3] used only the position data of hands and feet to adopt them to a virtual character with an inverse kinematics technique. Residual degrees of freedom are fixed by exploiting biomechanical knowledge. Choi et al. [7] adopted the idea of inverse rate control [29] to compute the changes in joint angles corresponding to those in end-effector positions while imitating the captured joint angles by exploiting the kinematic redundancy.

Implicit in the schemes that try to preserve the captured end-effector positions, is the notion that end-effector positions are more important than joint angles; that is, joint angles should be changed to achieve end-effector positioning goals. While this prioritization is often preferable to the reverse, it is not without its flaws. Consider the example of Figure 2.2. In this example,

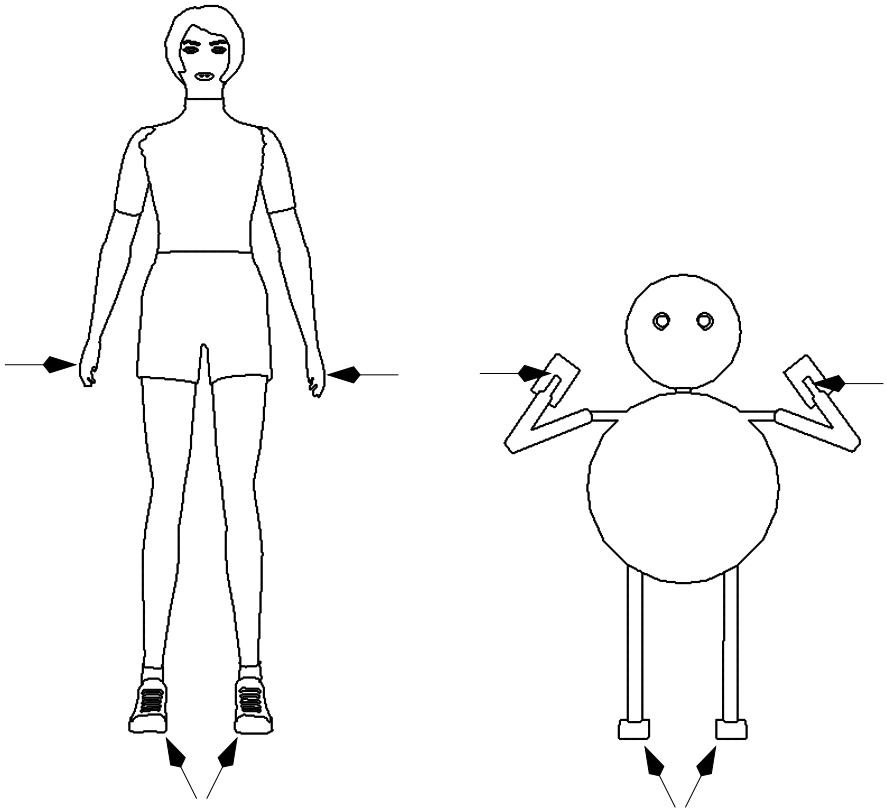


Figure 2.2: Artifacts of position-based approach

the importance of the foot position is properly reflected, while that of the hand positions is overstated.

The central observation of an importance-based approach is that what is important can only be determined by the context of the motion. At each instant, a system must somehow select among the many possible things that are important, so it can change the aspects that are not important. Constraint-based approaches to motion adaptation explicitly represent details of the motion that are importance as geometric constraints. The space-time motion

editing and retargeting system of Gleicher [11, 12] proposed the notion of preserving the importance quantities of the motion by changing unimportance ones, where the important qualities were defined by constraints. Lee and Shin's [19] hierarchical motion editing provided similar results using a different underlying implementation. Popovic and Witkin [23] demonstrated results that made the kinetic aspects of the original motion important to preserve.

The methods mentioned in the previous paragraph are all offline in that they examine the entire motion simultaneously in processing. This offline nature is also implicit in the problem formulation, as well as in the solution method. All of the methods require the constraints to be identified before the motion can be processed. The decisions as to what is important in a motion must be known before processing can occur in these previous constraint-based approaches. This is infeasible in online applications. Bindiganavale and Badler [4] introduced a scheme to generate constraints automatically. However, their motion adaptation is done in an offline manner.

For computer puppetry, we must decide what is important in a given motion in an online manner. We observe that artifacts of applying only the captured joint angles are apparent when the performer is interacting with itself or the external world. Such interactions include the interactions of the performer with the environment and its self-interactions among the body segments. Those interactions are mostly done by limb segments such as hands, feet, elbows and knees, since they are the most active parts of human

body. Therefore we analyze the importance of each end-effector position to preserve the interaction with the environment. Moreover the importance of the relative position of each limb segment to the other segments is also measured for reproducing the self-interactions.

Those importance values are calculated based on several factors discussed in Chapter 4. For example, the proximity of an end-effector to its surrounding environment can be used as a predictor of the importance of the end-effector position. The importance of the end-effector position is inversely proportional to its distance to the nearest object in the environment. Similarly the proximity of a limb segment to the others is also a good criterion for measuring the importance of its relative position with respect to the other body segments. A key notion of this work is that the power of an importance-based approach, already demonstrated in offline constraint-based systems, can be brought to the online domain of computer puppetry.

2.3 Inverse Kinematics

We employ an inverse kinematics (IK) solver to compute the pose of the target character. IK has become a standard technique in animation systems to control the pose of a character based on the positions of its end-effectors.

IK solvers can be divided into two categories: analytic and numerical solvers. Most industrial robot manipulators are designed to have analytic solutions for efficient and robust control. Paden [22] divided an IK problem into a series of simpler subproblems each of which has closed-form solu-

tions. Korein and Badler [18] showed that the IK problem of a human limb allows an analytic solution, and Tolani et al. [27] derived their actual solutions. A numerical method relies on an iterative process to obtain a so1ution. Girard and Maciejewski [10] generated the locomotion of a legged figure using a pseudo inverse of a Jacobian matrix. Based on neurophysiology, Koga et al. [17] produced an experimentally good initial guess for a numerical procedure. Gullapalli et al. [13] reduced the dimensionality of the redundant control system using synergies as a basis control set. Zhao and Badler [30] formulated the IK problem as a constrained nonlinear optimization problem. Rose et al. [25] extended this formulation to cover constraints that hold over an interval. To prevent the figure from making unnatural motions and reduce the redundancy of the IK problem, Badler et al. [3] incorporated biomechanical information.

For computer puppetry, we make a number of demands on IK that require the development of a novel solver. First, we must achieve real-time performance on the entire body of the character. Second, we need the solver to provide predictably consistent solutions: small changes to the problems should provide similar answers. Finally the solver must be able to account for the importance of each feature to preserve that is determined dynamically in the analysis phase. The proposed IK solver is discussed in Chapter 5. To solve an IK problem in real-time, we divide it into three subproblems: root position estimation, body posture computation, and limb posture computation. First, the root position of a virtual character is computed to provide

a good initial guess for the body posture computation. If needed, we then adopt numerical optimization to refine the body posture, which consists of the root position, the root orientation and the posture of the upper body. Finally, we use an analytic IK solver to compute the limb postures and blend them with the captured limb postures.

The solution for each of these subproblems is designed to incorporate the importance values measured in the analysis phase. Solving the first and second subproblems, our IK solver tries to preserve the captured end-effector positions when their importance values are high. Otherwise, the captured joint angles are preserved. Here we need not to account for the relative positions of the limb segments, since those two steps adjust the global posture of the character and unlikely affect the spatial relation between the segments. In contrast, at the final limb posture computation we adjust the limb to preserve the relative position of the limb segments as well as the end-effector positions in accordance with their importance values. That is, we reproduce the more important features in the resulting motion while sacrificing less important ones. To generate realistic motions, our IK solver tries to keep the captured joint angles of the limb, when none of those features are important enough to preserve the interaction with the other body parts or the environment.

Chapter 3

Motion Filtering

In general, motion capture devices capable of providing real-time performance are particularly susceptible to noise. Magnetic motion capture systems, which are widely used for real-time motion capture, suffer from the interference of low-frequency current-generating devices such as a CRT-type display. Thus, there always exists some level of jitter, that is, rapid random changes in reported positions and orientations that do not correspond to actual movements [8]. Since computer puppetry requires a high quality input motion as the reference of an output motion, filtering is an essential part. In the context of computer puppetry filtering must be real-time, online, and performed on orientations as well as positions.

For online filtering, Kalman filters [2, 9, 28] are often employed because of their capability of prediction and correction, that is, predicting future input data from their history and correcting them by incorporating actual input data. In a standard(extended) Kalman filter, its state would completely describe the positions of a sensor and its velocity. However, because of the nonlinearity of the orientation space, this scheme can hardly be applied directly to orientation data. Adopting the results in Azarbayejani and Pentland

[1] and Broida and Chellappa [6], Welch and Bishop [28] parameterize an incremental orientation change with Euler angles which are regarded as a three-vector to filter. The filtered Euler angles are transformed back to an incremental orientation change in the nonlinear space to update the target orientation at each time step. However, the conversion between an incremental orientation change and its equivalent Euler angles is inefficient. Moreover, recent motion capture devices measure orientations directly in unit quaternions. Therefore, differently from Welch and Bishop, we parameterize incremental orientation changes with rotation vectors.

To facilitate this scheme, the target orientation $\mathbf{q}_e$ is maintained externally to the Kalman filter together with the internal state vector $\mathbf{x}$. In particular, $\mathbf{q}_e$ is represented by an unit quaternion:

$$\mathbf{q}_e = (w\ (x\ y\ z)),$$

where $w^2 + x^2 + y^2 + z^2 = 1$. The internal state $\mathbf{x}$ consists of the position $\mathbf{p}$, the rotation vector $\mathbf{r}$, and their derivatives $\dot{\mathbf{p}}$ and $\dot{\mathbf{r}}$:

$$\mathbf{x} = (\mathbf{p}^T\ \dot{\mathbf{p}}^T\ \mathbf{r}^T\ \dot{\mathbf{r}}^T)^T. \tag{3.1}$$

Here the rotation vector $\mathbf{r}$ parameterizes the incremental orientation change of the actual sensor input $\mathbf{q}(t)$ at the current frame with respect to the target orientation $\mathbf{q}_e(t-\Delta t)$ at its previous frame. Therefore, $\mathbf{r}(t)$ can be measured

through the logarithm map [16]:

$$\mathbf{r}(t) = \ln(\mathbf{q}_e^{-1}(t - \Delta t)\mathbf{q}(t)). \tag{3.2}$$

At each filter update step, $\mathbf{r}(t)$ in the state is converted into its incremental orientation change equivalent $e^{\mathbf{r}(t)}$ through the exponential map to update the external target orientation $\mathbf{q}_e$ and then reset to be zero. Therefore, incremental orientations are linearized for the (extended) Kalman filter, centered about zero.

The dynamic model predicts the current position and the rotation by first-order approximations. Therefore, the prediction $\hat{\mathbf{x}}^-(t)$ of the state through the *state transition matrix* $\mathbf{A}$ can be described :

$$\hat{\mathbf{x}}^-(t) = \mathbf{A}\hat{\mathbf{x}}(t - \Delta t) = \begin{bmatrix} \mathbf{I}_3 & \Delta t\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \Delta t\mathbf{I}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 \end{bmatrix} \hat{\mathbf{x}}(t - \Delta t), \tag{3.3}$$

where $\mathbf{I}_3$ and $\mathbf{0}_3$ are, respectively, 3×3 identity and zero matrices. Similarly, the *error covariance matrix* $\mathbf{P}(t)$ is predicted:

$$\mathbf{P}^-(t) = \mathbf{A}\mathbf{P}(t - \Delta t)\mathbf{A}^T + \mathbf{Q}. \tag{3.4}$$

Here, $\mathbf{P}(t) = E\left[(\hat{\mathbf{x}}^-(t) - \mathbf{x}(t))\,(\hat{\mathbf{x}}^-(t) - \mathbf{x}(t))^T\right]$, which models estimation uncertainty. The *process noise covariance matrix* $\mathbf{Q}$ characterizes the

accuracy of the dynamic model. In our implementation, we simplify $\mathbf{Q}$ as:

$$\mathbf{Q} = \begin{bmatrix} q_1\mathbf{I}_3 & q_2\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ q_3\mathbf{I}_3 & q_4\mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & q_5\mathbf{I}_3 & q_6\mathbf{I}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & q_7\mathbf{I}_3 & q_8\mathbf{I}_3 \end{bmatrix}. \tag{3.5}$$

When the values of q_i's are small, the filter tends to suppress the detail of the captured motion. On the other hand, if they are large, it tends to preserve the captured motion. Therefore, q_i's should be tuned interactively for a good filter response.

In practice, we sample motion signals at a higher frame rate (120 Hz) than that actually required for animation to avoid the overshooting which occasionally occurs in constant velocity models, especially when the velocity changes suddenly. Our measurement consists of the position of a sensor and its incremental orientation represented by a rotation vector; that is, $=(\mathbf{p}^T\ \mathbf{r}^T)^T$ which can be obtained from of the state vector directly. Therefore, our measurement can be estimated from the predicted state:

$$\hat{\mathbf{z}}(t) = \mathbf{H}\hat{\mathbf{x}}^-(t) = \begin{bmatrix} \mathbf{I}_3 & \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{0}_3 \\ \mathbf{0}_3 & \mathbf{0}_3 & \mathbf{I}_3 & \mathbf{0}_3 \end{bmatrix} \hat{\mathbf{x}}^-(t). \tag{3.6}$$

Now, we are ready to compute the Kalman gain $\mathbf{K}(t)$:

$$\mathbf{K}(t) = \mathbf{P}^-(t)\mathbf{H}^T(\mathbf{H}\mathbf{P}^-(t)\mathbf{H}^T + \mathbf{R})^{-1}, \tag{3.7}$$

where $\mathbf{R}$ is the *measurement noise covariance matrix*. That matrix is either given from the device manufacturer or acquired by offline measurement. In practice, we measure the noise while holding the sensor stationary to compute its noise covariance matrix $\mathbf{R}$.

The residual between the actual sensor measurement $\mathbf{z}(t)$ and the predicted measurement $\hat{\mathbf{z}}(t)$ from Equation (3.6) is:

$$\Delta\mathbf{z}(t) = \mathbf{z}(t) - \hat{\mathbf{z}}(t). \tag{3.8}$$

Then, the predicted state and the error covariance matrix are corrected as follows,

$$\begin{aligned} \hat{\mathbf{x}}(t) &= \hat{\mathbf{x}}^{-}(t) + \mathbf{K}(t)\Delta\mathbf{z}(t), \quad \text{and} \\ \mathbf{P}(t) &= (\mathbf{I} - \mathbf{K}(t)\mathbf{H})\mathbf{P}^{-}(t). \end{aligned} \tag{3.9}$$

We finish filtering at each frame by updating the external target orientation using the rotation vector $\hat{\mathbf{r}}(t)$. Taking the exponential map of the rotation vector and postmultiplying it with the external target orientation $\hat{\mathbf{q}}_e(t - \Delta t)$ at the previous frame, we can find the final target orientation $\hat{\mathbf{q}}_e(t)$ at the current frame:

$$\hat{\mathbf{q}}_e(t) = \hat{\mathbf{q}}_e(t - \Delta t)e^{\hat{\mathbf{r}}(t)}. \tag{3.10}$$

The rotation vector $\hat{\mathbf{r}}(t)$ is reset to zero for filtering at the next frame.

Chapter 4

Motion Analysis

When the performer and the target character do not have the same size and shape, not all aspects of the original motion can be preserved. A system must determine what aspects of the motion are important to preserve, so that other less important aspects may be changed to preserve them.

For an articulated figure, differing segment lengths means that the joint angles and end-effector position cannot be recreated simultaneously. Moreover, the difference in shape prevents the character from preserve the relative position of segments to the other while keeping the joint angles. There are five obvious choices of motion aspects to preserve:

1. the position of the root of the character,
2. the joint angles
3. the positions of the end-effectors,
4. the distance from the end-effector to its closest segments, and
5. the distance from the elbow/knee to its closest segments.

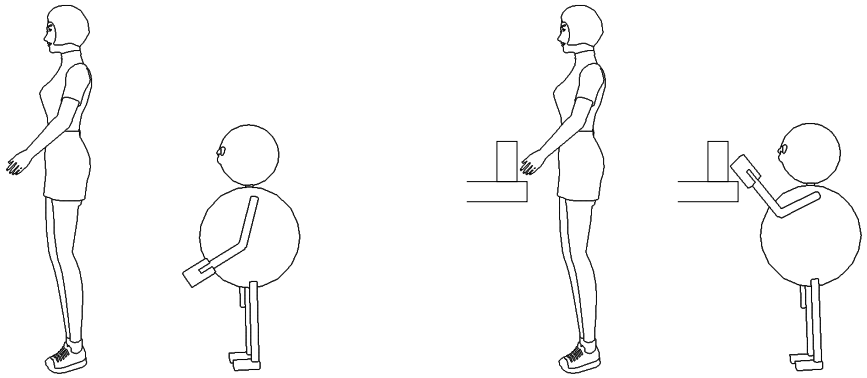
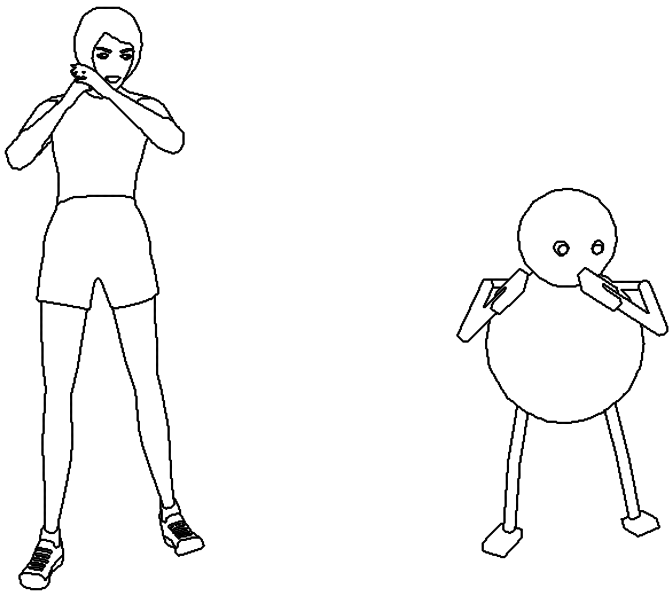


Figure 4.1: Two different situations

There exist situation under which any of these five might be most important. For example, observe the arm postures in Figure 4.1. Figure 4.1(a) shows a captured arm posture from the performer that does not touch any object. Retargeting this motion to a virtual character, we prefer the posture in the Figure 4.1(b) that preserves the joint angles. However, the position of a hand needs to be preserved when it touches an object as shown in Figure 4.1(c) and (d). When the character is clapping, as shown in Figure 4.2(a), the distance between the hands are the most important feature to preserve to prevent the unnatural posture given in Figure 4.2(b). To avoid unwanted penetration, the distance between the elbow and the body segment must be kept especially when the character is fat.

The suggested system must choose which of the five choices above are important in a dynamic online way. To make this decision, we employ a number of heuristics.



(a) Captured Clapping Motion (b) Applying Joint Angles

Figure 4.2: Clapping motions

1. The position of the root is most likely *not* important. This heuristic comes from the observation that the choice of making the root is arbitrary: we could have just as easily chosen any point as the root. In fact, preserving the root position may change some important parameters that characterize a posture itself. Because of this, the importance of the root position is downplayed in many approaches that consider importance. Like the proposed solver, described in Chapter 5, Gleicher's retargeting system [12] uses a heuristic that attempts to satisfy

the constraints (generally on the end-effectors) as much as possible by moving the root position.

2. If an end-effector is interacting with another object (such as the floor), then its position is likely to be important. Therefore, proximity to objects in the environment should increase the importance of the absolute position of an end-effector.

3. If an end-effector is close to another segment of the character, then its relative displacement is likely to be important to duplicate a possible self-interaction and to prevent self-penetration. Therefore, proximity to the other segments of the body increases the importance of an end-effector's distance from the nearest segment.

4. Similarly, if an elbow/knee is close to the another segment of the character, then their distance is likely to be important. Therefore, proximity to the other segments should increase the importance of the distance from the elbow/knee to the nearest segment.

5. If an end-effector will be interacting with another object in the near future, then its position is important (as it is likely to be getting ready for the interaction). Therefore, we incorporate prediction of proximity of an end-effector to an object in the measure of its importance.

6. If an end-effector has just finished interacting with another object and is moving away from it, its position may not be as important as its

proximity suggests.

7. If the end-effector is not in proximity to another object, it is likely that its position is unimportant.

In order to measure the interactivity of an end-effector with its environment and the self-interactivity of an end-effector or an elbow/knee, we introduce the notion of *importance* of features, which can be determined by analyzing the posture of the performer. In particular, the distance from the end-effector to objects in the environment is a good measure of interaction possibility of the end-effector. That is, the end-effector is more likely to interact with the environment when it is closer to objects in the environment. Therefore, as the end-effector approaches an object, its importance value should be increased to enforce the geometric constraints created by the object. As the end-effector moves away from the object, the importance value should be continuously decreased to preserve the captured posture of the corresponding limb. Moreover, it is desirable to develop the distance measure to reflect the trajectory of an end-effector and its dynamic nature. Similarly, the importance of the distance between an end-effector (or an elbow/knee) its the closest segment can be used to measure self-interactivity as well as the self-penetration possibility of the character. The remainder of this chapter, we describe the way to measure importance values of features based on those observations.

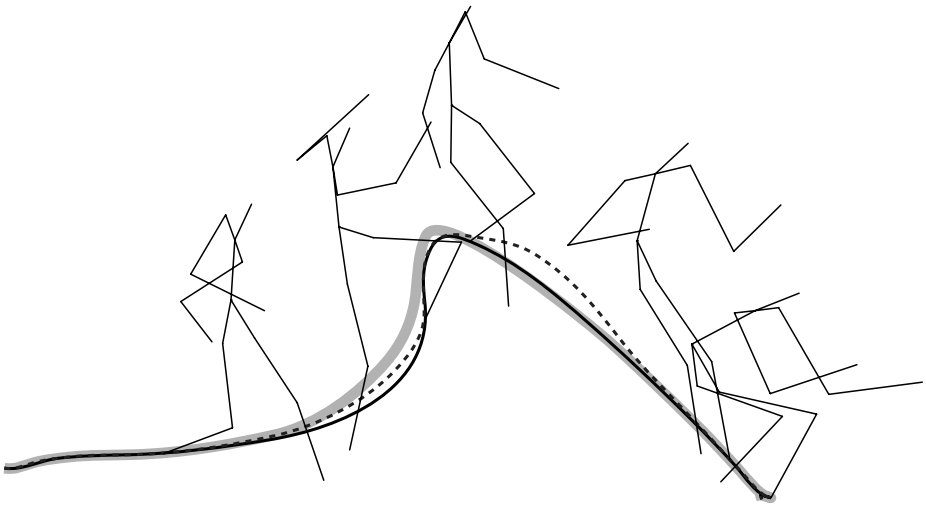


Figure 4.3: Trajectories of the left foot generated by varying importance measure

4.1 Analysis of Interaction with Environment

Given end-effector e_i of the performer and object o_j in the real space, let $d_{ij}(t)$ be Euclidean distance between them at time t. o_j has is its corresponding object in virtual space. The new distance function $d^+_{ij}(t)$ is defined as

$$d^+_{ij}(t) = \frac{d_{ij}(t) + d_{ij}(t + \kappa\Delta t)}{2} \tag{4.1}$$

for small positive κ and Δt. $d^+_{ij}(t)$ represents the average of the current distance and the predicted distance after $\kappa\Delta t$ time. For small Δt, $d^+_{ij}(t)$ can

be approximated as

$$\begin{aligned} d_{ij}^{+}(t) &\approx \frac{d_{ij}(t) + (d_{ij}(t) + \kappa\Delta t\, \dot{d}_{ij}(t))}{2} \\ &= d_{ij}(t) + \frac{\kappa\Delta t}{2}\dot{d}_{ij}(t) = d_{ij}(t) + \lambda\dot{d}_{ij}(t), \end{aligned} \tag{4.2}$$

where $\dot{d}_{ij}(t)$ is the first derivative of $d_{ij}(t)$. $d_{ij}^{+}(t)$ reflects both the distance at t from e_i to o_j and its changing rate $\dot{d}_{ij}(t)$. By varying λ we can control the degree of prediction for $d_{ij}^{+}(t)$.

For example, Figure 4.3 exhibits a jumping motion adapted with $\lambda = 0$ and $\lambda = 0.15$. The legs of the character are shorter than the performer's. For $\lambda = 0$, the left foot trajectory of the character (dashed line) agrees with that of the performer (thicker line) only near the floor. For $\lambda = 0.15$, the former follows the latter while approaching down to the floor (solid line). The foot is moving off the captured trajectory to preserve the captured joint angles, either near the peak ($\lambda = 0$) or approaching to the peak ($\lambda = 0.15$).

Let D_{ij} denote the maximum distance within which e_i is influenced by o_j. Then, the normalized distance $\bar{d}_{ij}$ is defined as

$$\bar{d}_{ij} = \frac{d_{ij}^{+}}{D_{ij}}. \tag{4.3}$$

An animator assigns D_{ij} for the pair of end-effector e_i and object o_j in the environment in accordance with a given animation context. A wider range of D_{ij} shows a sensitive interaction of end-effector e_i with object o_j. On

the other hand, a narrower range exhibits that e_i moves independently of o_j unless e_i is close to o_j.

The importance is zero when the normalized distance $\bar{d}_{ij}$ is greater than or equal to one, that is, e_i is out of the influence of o_j. As the distance decreases to zero, the importance increases to one. Thus, the importance function p of the normalized distance $\bar{d}_{ij}$ can be designed with the condition of $p(1) = 0$ and $p(0) = 1$. In addition, we set its derivatives there to be zero, that is, $p'(0) = 0$ and $p'(1) = 0$, to reduce the rate of change of the function p at both extreme points. Thus, the importance of e_i with respect to o_j is represented by the cubic polynomial function p satisfying those conditions. That is,

$$p(\bar{d}_{ij}) = \begin{cases} 2\bar{d}_{ij}^3 - 3\bar{d}_{ij}^2 + 1, & \text{if } \bar{d}_{ij} < 1, \\ 0, & \text{otherwise.} \end{cases} \tag{4.4}$$

The importance value w_i of end-effector e_i over all external objects can be defined as the maximum of them:

$$w_i = \max_j(p(\bar{d}_{ij})). \tag{4.5}$$

It requires much time to compute the distance $\bar{d}_{ij}$ from each end-effector e_i of a virtual character to every object o_j in the environment, especially for a complex surrounding environment. To achieve a real-time performance, we need to minimize the number of possible objects that interact with each end-effector in accordance with an animation context. An object that is hardly

touched during the animation may be eliminated in importance value computation. Moreover, objects may also be described approximately with simpler geometry for easy distance computation.

4.2 Analysis of Self-interaction

To obtain realistic results, we need to preserve self-interactions among the segments of the performer as well as its interaction with the environment while preventing their self-interpenetrations. Most of self-interactions are done by the limbs, which are the most active segments of a human body. Therefore, we try to reproduce the captured self-interactions of the performer in the resulting motion by analyzing the importance values of the self-interactions done by hands, feet, elbows, and knees and preserving them in accordance with their importance values. For further reference, we call those parts 'general end-effectors', since each of them is the endpoint of the corresponding segment.

The importance of the self-interaction of each general end-effector is measured based on its proximity to the other segments. When the general end-effector is close to another segment of the performer's body, it is likely to either interact with or penetrate into the segment. Therefore, the distance of a general end-effector from the other segments gives a clue to determining how much we need to preserve its relative position with respect to the closest segments. In this section, we first discusses how to compute the proximities of the general end-effectors and then how to analyze the importance values

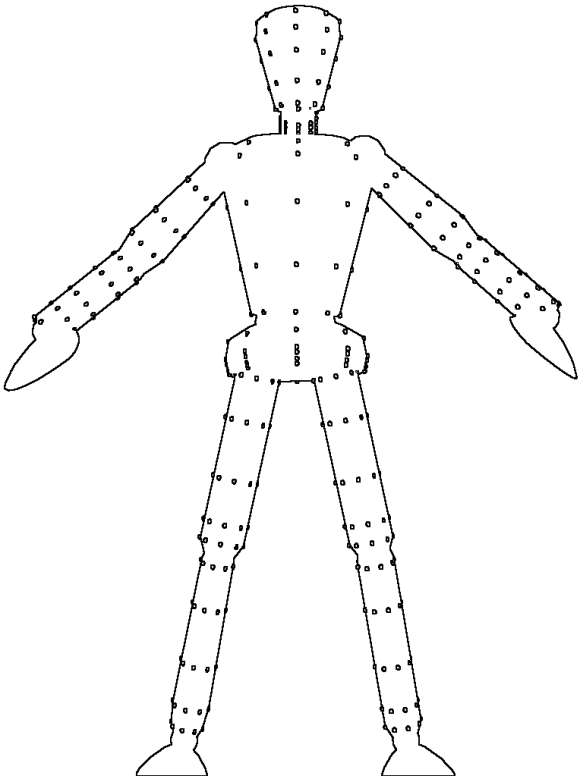


Figure 4.4: Example of body point sampling

based on their proximities.

To measure the distance of a general end-effector from the other segments, we need to know the geometries of the performer. However, it is time-consuming to capture the precise geometry of the performer and to measure the exact distance between a pair of segments. We simplify distance computation by measuring the proximity of a general end-effector to its closest sample point on another segment of the performer. A number of points are sampled on the surface of the performer with motion capture sensors. In practice, we sampled forty points regularly on each segment as il-

lustrated in Figure 4.4 by drawing eight closed curves on a segment aligned with its skeleton and spaced evenly, and sampling five points along each curve regularly. With those sample points, the distance of a general end-effector from a segment can be computed by measuring the proximity of the general end-effector to its closest sample point on the segment.

Similarly to the importance value of the end-effector interaction with the environment, we can measure the importance value of the self-interaction of a general end-effector based on its distances from the other segments. As shown in Section 4.1, the importance value is inversely proportional to the distance from another segment, since the closer the general end-effector is to the segment, the more it is likely interact with the segment. To compute the importance value of the self-interaction with each segment, the first-order prediction and normalization are applied to the distance of the general end-effector from the segment of the performer through Equations (4.2) and (4.3), respectively. Then, each importance value is obtained by the importance function shown in Equation (4.4). Finally, the importance value of each general end-effector over all segments of the performer can be defined as the maximum of them. Those importance values allow us to determine how urgent the self-interaction among the segments is.

Chapter 5

Real-time Inverse Kinematics Solver

The final step of motion retargeting is posing the character so that it preserves the as many important features in the motion as possible. Given the captured end-effector position with its importance, and the bounding 3-D balls of the end-effector and elbow/knee, we adjust the captured joint angle by introducing an inverse kinematics solver that is specialized for the problem.

For computer puppetry, we must position the character such that the important aspects of a captured motion are preserved while providing real-time performance. For the application, this demands computing the character's posture 30 times per second. Therefore, we need an IK solver that not only can incorporate the importance measures of the previous chapter, but also has real-time performance even in the worst case.

As discussed in Section 2.3, previous IK solution methods do not meet the demands of computer puppetry. Analytic methods provide guaranteed performance but cannot incorporate importance measures required for retargeting. Numerical solvers can include the importance metrics, but they hardly guarantee real-time performance. To meet these two conflicting demands, we have developed a hybrid solver.

In this chapter, we present a fast IK algorithm that is specialized for human-like articulated characters. We divide the IK process into three sub-problems: root position estimation, body posture computation, and limb-posture computation. For each step, We give a method that is specialized to achieve high-performance. This leads us to employ inexpensive, closed-form solutions if applicable, and reserve numerical optimization for the case in which it is absolutely required. Notice that we need not to include the bounding 3-D spheres in the root position estimation, and the body posture computation, since those steps manipulate the global position and posture of the character.

5.1 Root Position Estimation

In order to position the end-effectors of a character, an IK solver may change the root position of the character or adjust its joint angles. As mentioned in Chapter 4, the root of the character has been arbitrarily chosen as the character's root, which is rarely the most important aspect to preserve. Therefore, the solver first attempts to make the character satisfy the constraints as much as possible by moving the root position. This strategy was demonstrated for retargeting by Gleicher [12].

Beginning with the positional offset has an important advantage: unlike angular changes that cause non-linear equations to compute, positional offset computation is trivial and therefore efficient. Let $\mathbf{p}_i^e$ represent the position of the ith end-effector when the character is posed with the captured joint

angles, and $\mathbf{p}_i^g$ denote the goal position for that end-effector. The displacement vector $\mathbf{d}_i = \mathbf{p}_i^g - \mathbf{p}_i^e$ measures how much the solver must move an end-effector to reach its goal. If there were only one end-effector with a specified goal position, this constraint could be met by simply moving the character's root position by the displacement vector, where the joint angles would not need to be changed.

In the event that multiple end-effectors are to be positioned, we compute the weighted average of the displacements to find an initial offset $\mathbf{d}$ as follows:

$$\mathbf{d} = \frac{\sum_i^n w_i \mathbf{d}_i}{\sum_i^n w_i}, \tag{5.1}$$

where w_i is the importance of the ith end-effector. In the (unlikely) event that all end-effectors require the same displacement, this displacement will solve all of the constraints. More likely, the joint angles will need to be adjusted so that all of the end-effector goals can be met.

While the weighted averaging attempts to position the root to meet all of the goals simultaneously, it does not necessarily guarantee that all goals can be met. Once the root position is fixed, the character can meet its goals by straightening its joints. Therefore, the root position must be chosen such that all end-effector goals are "reachable," that is, close enough that straightening limbs will be sufficient. The root position estimate are further refined such that it guarantees reachability if possible. We relocate the root such that it is within the reachability limits to the goals while being as close to the initial

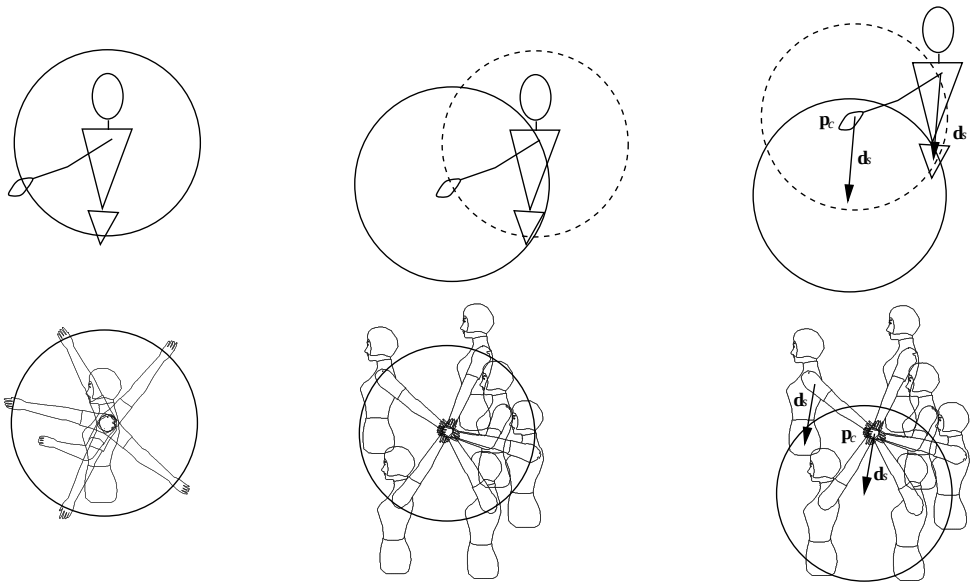


Figure 5.1: Range 3-D balls: range of hand, shoulder, and root position

estimate as possible.

As shown in the left column of Figure 5.1, the reachable range of the hand can be represented as the 3-D ball centered at the shoulder, and its radius is the length of the arm. Here, a 3-D ball consists of a sphere and the set of all points bounded by it. The middle of Figure 5.1 shows that the same 3-D ball centered at the goal position represents the range of the shoulder joint position. Finally, with the orientations of the pelvis and the waist fixed as in the captured posture, the range of the root position is computed as illustrated on the right of Figure 5.1. Let $\mathbf{d}_s$ denote the vector from the shoulder to the root position. The translation of the 3-D ball at the goal position $\mathbf{p}_c$ by the vector $\mathbf{d}_s$ yields the 3-D ball that gives the range of the

root position. If the root is in this 3-D ball, the character can reach the goal position by stretching the limb only.

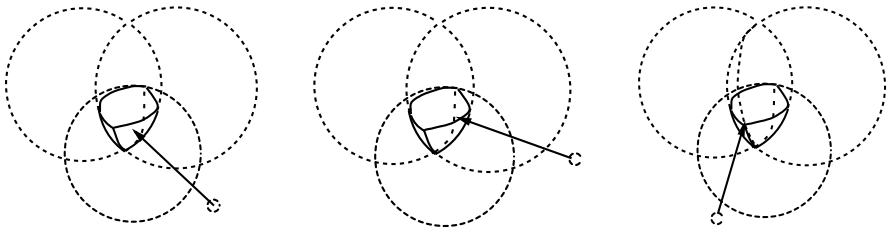
When the importance value of an end-effector is low, the root position does not need to be modified to make this end-effector reachable at its goal. Therefore, the range corresponding to this end-effector may be larger than the actual reachable range. To avoid an unnecessary offset of the root position, we enlarge the size of the 3-D ball, so that its size is inversely proportional to the importance value. The increased radius r_i corresponding to the ith limb is given as

$$r_i(l_i, w_i) = \frac{l_i}{w_i}, \tag{5.2}$$

where l_i is the length of the ith limb and w_i is its importance value.

Since the virtual character has four end-effectors, we have four 3-D balls. The common intersection of these 3-D balls is the range of the root position that makes all of the end-effectors reachable to their goal positions. As an initial guess for the root position, we choose the closest point from the offset root position to this intersection to preserve the posture of the performer as much as possible. Thus, the root position estimation is formulated as the problem of finding the closest point from a given position to the common intersection of four 3-D balls.

The intersection of 3-D balls consists of four surface elements as shown in Figure 5.2: spherical regions, circular edges, and vertices. A spherical region is a part of a sphere bounded by a sequence of spherical arcs. A



(a) On a sphere (b) On a circle (c) At a vertex

Figure 5.2: Closest points

circular edge is a part of a circle that is the intersection of two spheres. A vertex is determined by the common intersection of three spheres.

There are two cases depending on the offset root position with respect to the intersection. If this point is contained in the interior of the intersection, then the point itself is the closest point to the intersection. Suppose that it is not contained in the interior. Then the closest point must lie on the boundary of the intersection. Therefore, we may enumerate all possible surface elements due to the intersection of the four spheres corresponding to the bounding surfaces of the balls, respectively.

Three spheres determine at most two vertices. Since there are four ways of choosing a triple out of four spheres, we have a maximum of eight vertices. Every pair of vertices can possibly admit a spherical edge, and thus we have at most 24 edges. However, these are completely included in a maximum of six circles. Moreover, each spherical face is completely contained in one of four spheres. Instead of enumerating all surfaces elements, we

equivalently check those spheres, circles and vertices.

We first compute the closest point to each sphere from the offset root position. Among these points, if any, we choose the point that is contained in the intersection and the closest to the root position. If such a point does not exist, then we compute the set of points, each of them is the closest from the root position to each circle. Out of them, we choose the one that is closest to the root position and in the intersection. Suppose that there does not exist such a point. Then one of vertices may be the solution. We choose the one closest to the root position among those contained in the intersection. For more details in computing the initial root position, refer to the Appendix. If there does not exist a common intersection of the balls, we discard the spheres that do not intersect the one whose corresponding end-effector has the largest importance value and repeat this process for the remaining balls.

5.2 Body Posture Computation

If the initial root position estimate does not allow all limbs to be reachable to the goal positions, we need to adjust the body posture consisting of the root position, the root orientation, and the posture of the upper body. Since those segments are tightly coupled, a numerical method is adopted to find their configurations. Numerical methods hardly guarantee a real-time response for computing the inverse kinematics of an entire human figure, while it is practical to solve only a small part of the IK problem numerically, and to employ analytic methods for the rest of the task. Such a hybrid solver was

demonstrated in [19].

We formulate a restricted version of the IK problem for determining the posture of the body posture separately from the problem of computing the posture of the limbs. The body posture of a character can be written as $\mathbf{v} = (\mathbf{p}_0, \mathbf{q}_0, \mathbf{q}_1, \cdots, \mathbf{q}_n)$, where $\mathbf{p}_0$ and $\mathbf{q}_0$ are the position and the orientation of the root, respectively. $\mathbf{q}_j, 1 \leq j \leq n$, are the orientations of body segments such as the waist and the upper body. When the character has a rigid torso, $\mathbf{v}$ is simply reduced to $(\mathbf{p}_0, \mathbf{q}_0, \mathbf{q}_1)$, since $n = 1$.

The objective function consists of two terms:

$$E = E_g + \alpha E_p, \tag{5.3}$$

where the first term E_g is for making the end-effectors reachable to their goals and the last term E_p is to preserve the captured posture. We will explain those two terms in detail.

E_g is the sum of E_i's, each of which is a function of the distance from the ith end-effector e_i to its goal position. Provided with the shoulder (or the coxa) position $\mathbf{p}_i^s$ of the ith limb and its goal position $\mathbf{p}_i^g$, E_i is given as follows:

$$E_i = \begin{cases} 0, & \text{if } ||\mathbf{p}_i^s - \mathbf{p}_i^g|| < l_i, \\ \left(||\mathbf{p}_i^s - \mathbf{p}_i^g|| - l_i\right)^2, & \text{otherwise,} \end{cases} \tag{5.4}$$

where l_i is the length of the ith limb when it is maximally stretched. E_i is zero when the end-effector e_i is able to reach its goal position. For this case,

we prefer to lengthen or shorten the corresponding limb than to adjust the body posture. Recall that an end-effector of a low importance value has no need to preserve its captured position. Thus, to relax the constraint on this end-effector we enlarge the range of the shoulder. By substituting the length l_i of each limb with the new radius $r_i = \frac{l_i}{w_i}$ as mentioned in Section 5.1, we have

$$E_i = \begin{cases} 0 & \text{, if } ||\mathbf{p}_i^s - \mathbf{p}_i^g|| < r_i, \\ \left(||\mathbf{p}_i^s - \mathbf{p}_i^g|| - r_i\right)^2 & \text{, otherwise.} \end{cases}$$

Note that with the importance value w_i of one, E_i plays a role of pulling the end-effector to reach the goal position exactly. On the other hand, as importance value w_i approaches zero, the ith end-effector keeps the original posture by preserving the joint angles.

Letting $\mathbf{q}_j^*$ and $\mathbf{p}_0^*$ be the captured orientation of the jth segment and the estimated position of the root, respectively, E_p is a weighted sum of the squared geodesic distances between $\mathbf{q}_j$ and $\mathbf{q}_j^*$ for all $0 \leq j \leq n$, and the squared distance between $\mathbf{p}_0$ and $\mathbf{p}_0^*$:

$$E_p = \sum_{j=0}^{n} \beta_j ||\ln(\mathbf{q}_j^{-1}\mathbf{q}_j^*)||^2 + \gamma ||\mathbf{p}_0 - \mathbf{p}_0^*||^2. \tag{5.5}$$

Minimizing E_p, we preserve the captured motion as much as possible. We find the optimal solution that minimizes the objective function by employing the conjugate gradient method. Here, we use the captured joint angles and the root position computed in Section 5.1 as the initial guess for the

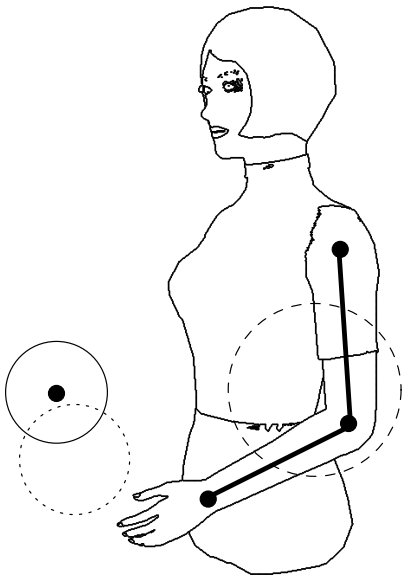


Figure 5.3: Ranges given for a limb

optimization.

5.3 Limb Postures Computation

Given the root position and the body posture, we finally adjusts the limb postures of the target character to preserve the important features in the captured motion to reproduce both the interaction with the environment and the self-interactions among the segments. Here, we have three possibly important features for a limb: the interaction of the end-effector with the environment, and the self-interactions of two kind of general end-effectors such

as hands/feet and elbows/knees. We represent each of those geometric constraints for preserving the interactions by the range 3-D ball as drawn in Figure 5.3. The target character can reproduce the interactions by locating each general end-effector inside of the corresponding ball. To provide an analytic solution for this problem, we first solve the joint angle of the shoulder to locate the elbow position and then calculate the joint angle of the elbow from the fixed elbow position to locate the end-effector. Finally, the joint angle of the wrist is adjusted to satisfy the captured hand orientation. The configuration of a leg is computed similarly by fixing the hip joint angle first, and adjusting the knee and the ankle in order.

5.3.1 Range for Preserving Self-interactions

We first determine the range of a general end-effector of the target character in which the character can preserve its self-interactions with the other segments while avoiding the self-interpenetrations. For illustration, the range of a hand and that of an elbow are drawn with the dotted and the dashed circles in Figure 5.3, respectively. If the importance value of the self-interaction of a general end-effector is low, the corresponding range should be wide enough not to alter the captured joint angle unnecessarily. Otherwise, the position of the general end-effector needs to be bounded tightly by the range 3-D ball to preserve its self-interaction. In both cases, the 3-D ball should effectively prevent the self-interpenetration.

The radius of the 3-D ball must be defined, so that it can prevent the

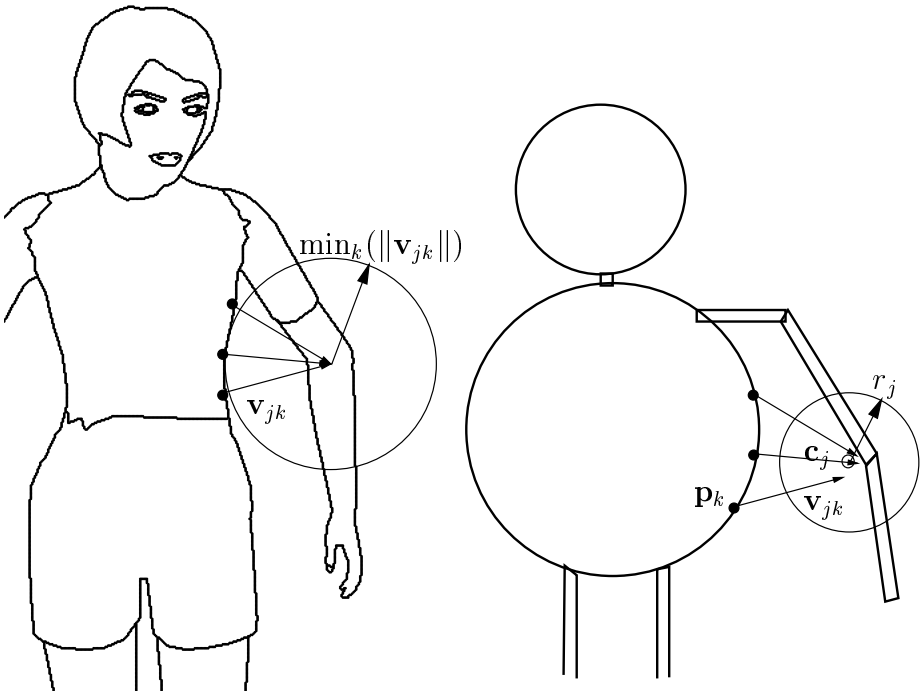


Figure 5.4: Range of an elbow to preserve its relative position

self-interpenetration of the corresponding general end-effector. Therefore, it should be smaller than or equal to the distance from the closest sample point, that is, $\min_k(\|\mathbf{v}_{jk}\|)$, where $\mathbf{v}_{jk}$ is the vector from the kth sample point to the joint position corresponding to the jth 3-D ball as shown in Figure 5.4. Moreover, the 3-D ball needs to gradually tighten up the target position as the importance value increases. Therefore, we define the radius r_j of the jth range 3-D ball as:

$$r_j = w_j \min_k(\|\mathbf{v}_{jk}\|) \tag{5.6}$$

This 3-D ball can limit the position of the general end-effector to preserve the

self-interaction and prevent the self-penetration when it is likely to interact with the other segment, while growing enough to allow the captured joint angles being preserved.

The center of the 3-D ball should be determined so that the distance from the closest segment is preserved. We choose the center of the ball as a weight average of the displacement from the three closest sample points as shown in Figure 5.4. To reduce the influence of farther points, their weight should be smaller than nearer ones. The weight of a point is computed by the importance function, which gives a lower value when the distance gets larger. Therefore, the resulting center position reflects the displacement from the closest point while moving smoothly due to the influence of nearby points. To compute the center position $\mathbf{c}_j$ of the jth end-effector (or elbow/knee), let $\mathbf{v}_{jk}$ be the displace of the jth end-effector to the kth sample point measured from the performer. Then, $\mathbf{c}_j$ is

$$\mathbf{c}_j = \frac{\sum_k \left(w_{jk}(\mathbf{v}_{jk} + \mathbf{p}_k)\right)}{\sum_k w_{jk}}, \tag{5.7}$$

where w_{jk} is the weight of the kth sample point computed through the importance function, and $\mathbf{p}_k$ is the position of kth sample point of the character.

However, when both of the general end-effector and the closest segment are included the limbs, their positions are adjusted by limb posture computation. Therefore, the range ball cannot make the general end-effector preserve the self-interaction. For example, suppose that the hands of the per-

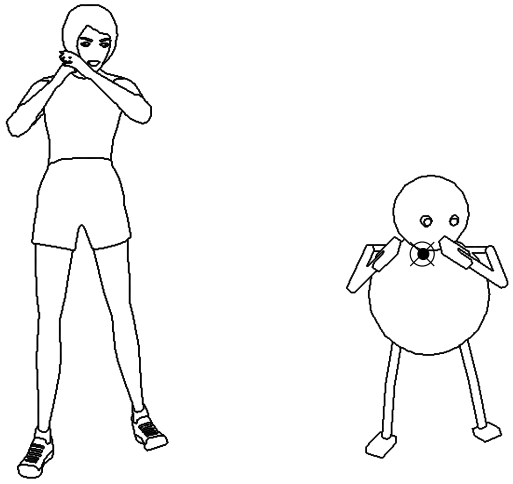


Figure 5.5: Target position when the closest segment is moved.

former are very close to each other in the captured posture as illustrated in the right column of Figure 5.5. Applying the captured joint angle gives the posture of the target character in which the end-effector are far from each other as show in right column of Figure 5.5. In such a case, the the center $\mathbf{c}_j$ of the ball obtained as above is the position of the other hand. However, the more desirable target position is their midpoint as drawn with the crossed circle in the right column of Figure 5.5. In general, when the closest segment to the jth general end-effector is a part of limbs, the center of the corresponding ball should located at the midpoint of the initial position of the general end-effector and the center position $\mathbf{c}_j$ computed as above. To address this situation, the center of the ball needs further refining. We begin

with rewriting Equation (5.7) into an equivalent form:

$$\mathbf{c}_j = \frac{\sum_k \left(w_{jk} (\mathbf{v}_{jk} + \mathbf{p}_k - \mathbf{p}_j^e) \right)}{\sum_k w_{jk}} + \mathbf{p}_j^e, \tag{5.8}$$

where $\mathbf{p}_j^e$ is the initial position of the jth end-effector (or elbow/knee) which is obtained by applying the captured joint angle. The refined center position is

$$\begin{aligned} \mathbf{c}_j &= \frac{\sum_k w_{jk} \mathbf{v}'_{jk}}{\sum_k w_{jk}} + \mathbf{p}_j^e \quad \text{and} \\ \mathbf{v}'_{jk} &= \begin{cases} \frac{\mathbf{v}_{jk} + \mathbf{p}_k - \mathbf{p}_j^e}{2}, & \text{where } k \in \mathcal{S}, \\ \mathbf{v}_{jk} + \mathbf{p}_k - \mathbf{p}_j^e, & \text{otherwise.} \end{cases} \end{aligned} \tag{5.9}$$

Here $\mathcal{S}$ is the set of points sampled on the segments which are to be moved by IK, that is the limb segments.

5.3.2 Range for Preserving Interaction with Environment

As drawn with the solid circle in Figure 5.3, we can similarly determine the range of the end-effector of the target character to preserve the interaction with the environment. When the importance value of the interaction with the environment is low, the end-effector needs not to be exactly located at its captured position. Otherwise, we have to preserve the captured end-effector position. Those requirements can be achieved by bounding the end-effector position in the 3-D ball of which the center is located at the goal position and its radius is inversely proportional to its importance. Therefore, its center is

simply the captured position of the end-effector and the radius must be zero when its importance value is one, and increased as the importance decreases. We define the radius r_i of the 3-D ball of the ith end-effector as

$$r_i = r_i^{max}(1 - w_i), \tag{5.10}$$

where w_i is the importance value of the interaction of the ith end-effector with the environment. r_i^{max} is the maximum radius of the end-effector range, which is defined as the radius of the maximum influence range of the end-effector with respect to its closest object in the environment.

In shoulder joint computation, we have to apply the range 3-D balls of a hand together with the range balls of the corresponding elbow. Otherwise we may be unable to find the solution in the following hand position adjustment. We have two range balls of the hand, one is due to the interaction with the environment and the other is for preserving the self-interaction. Therefore, we need to compute the ranges of the elbow corresponding to those range 3-D balls of the hand.

Suppose that the end-effector is located at the boundary of a 3-D ball, and the lower arm is directed outward. Then as drawn with the larger solid sphere in Figure 5.6, the elbow lies on the sphere of which the radius coincides with the 3-D ball and its radius is $l_i + r_m$, where l_i and r_m is the length of the ith forearm and the radius of the 3-D ball bounding the hand, respectively. In contrast, the lower arm passes through the center of the ball, the position

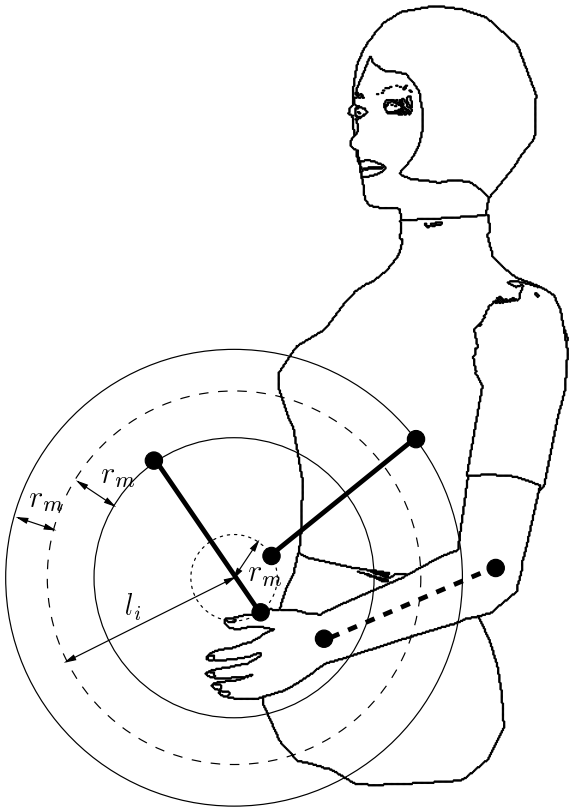


Figure 5.6: Range of elbow corresponding range of hand

of the elbow is on the sphere centered at the same as the ball and of which the radius is $l_i - r_m$ as shown with the smaller solid circle in Figure 5.6. Therefore, if the hand is placed in the 3-D ball, the elbow is located inside of the larger sphere and outside of the smaller one. Moreover, when the elbow is in this range, we are able to make the end-effector reachable to the 3-D ball. With this fact, we can represent the range of the elbow for preserving each interaction of the end-effector by two 3-D balls.

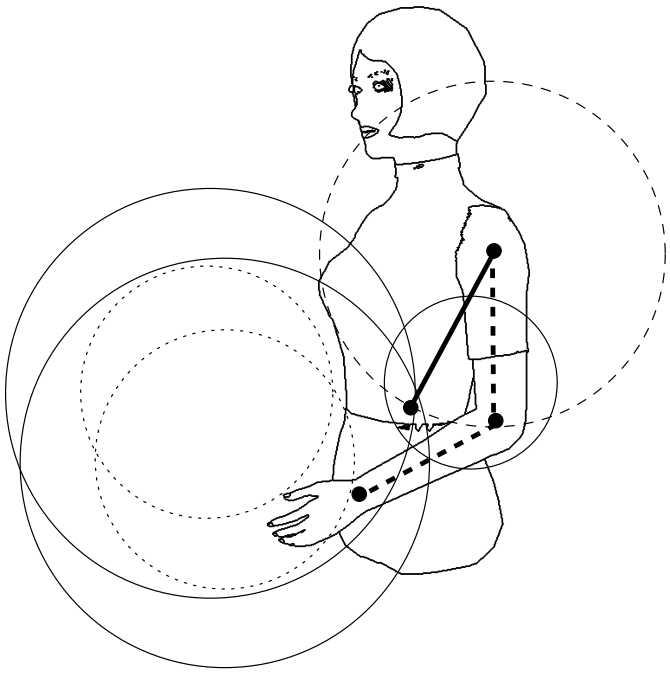


Figure 5.7: Range balls of elbow and its range sphere

5.3.3 Posture Computation

The target elbow position is selected in the common intersection of the range 3-D balls representing its geometric constraints. As drawn with the solid and dotted circle in in Figure 5.7, we have five range balls, one bounding ball from the range for keeping the self-interaction of the elbow, two from the range of the hand for preserving the interaction with the environment, and two for preserving the self-interaction of the hand. Those five balls are drawn with the solid and dotted circle in Figure 5.7. When the elbow is located in the common intersection, the elbow position preserves its distance from

the closest segment in accordance with the corresponding importance value while guaranteeing the reachability of the hand to its bounding range. From the fixed position of the shoulder, the elbow lies on the sphere of which the center is the position of the shoulder and its radius is the length of the upper arm (See the dashed circle in Figure 5.7.) On this sphere, we find the closest point included in the common intersection from the initial elbow position which is obtained by keeping the captured joint angles of the shoulder to preserve the original posture as much as possible. The closest point to the common intersection of the 3-D balls from the sphere can be found with the algorithm described in Section 5.1.

Given the target position of the elbow, the joint angle q_{i0} of the shoulder is computed as follows: Let the position of the shoulder is $\mathbf{p}_s^*$ and the initial and the target elbow positions are $\mathbf{p}_e^*$ and $\mathbf{p}_e$, respectively. Then, as shown in Figure 5.8, the vectors $\mathbf{v}_u^*$ and $\mathbf{v}_u$ representing the captured and the target directions of the upperarm are $\mathbf{v}_u^* = \mathbf{p}_e^* - \mathbf{p}_s^*$ and $\mathbf{v}_u = \mathbf{p}_e - \mathbf{p}_s^*$, respectively. The quaternion which rotates $\mathbf{v}_u^*$ to $\mathbf{v}_u$ is

$$\mathbf{q}_u^r = e^{\mathbf{n}\theta/2}, \tag{5.11}$$

where $\mathbf{n}$ is the rotation axis and the angle θ between $\mathbf{v}_u^*$ and $\mathbf{v}_u$ are given as

$$\mathbf{n} = \frac{\mathbf{v}_u \times \mathbf{v}_u^*}{\|\mathbf{v}_u \times \mathbf{v}_u^*\|}$$

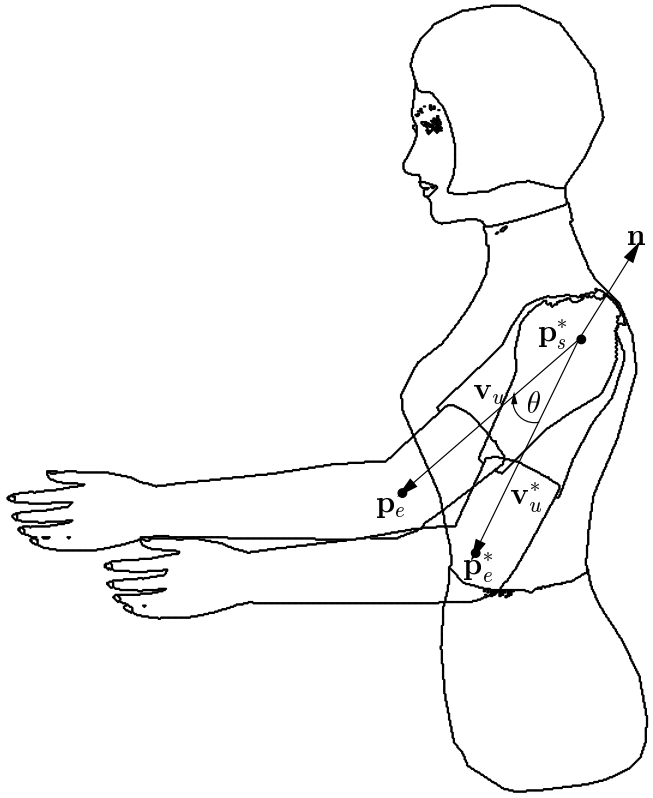


Figure 5.8: Adjusting Shoulder Joint Angle

and

$$\theta = \cos^{-1}(\mathbf{v}_u \cdot \mathbf{v}_u^*),$$

respectively. The shoulder joint angle $\mathbf{q}_{i0}$ can be achieved by premultiplying it to the the captured shoulder joint angle $\mathbf{q}_{i0}^*$:

$$\mathbf{q}_{i0} = \mathbf{q}_u^r \mathbf{q}_{i0}^*. \tag{5.12}$$

Provided with the fixed position of the elbow, its joint angle is computed

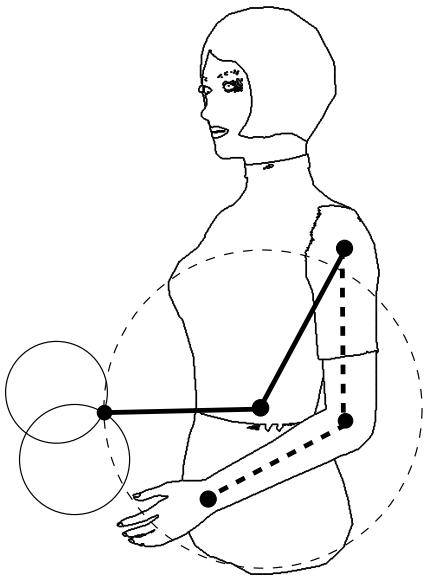


Figure 5.9: Range of hand

similarly to the shoulder joint angle. For the position of the hand, we have two bounding balls (the solid circles in Figure 5.9) for preserving its interaction and the self-interaction, and one sphere representing the reachable range of the hand from the fixed elbow position (the dashed circle in Figure 5.9). We find the closest point on the common intersection of them from the initial hand position which is obtained by applying the captured elbow joint angle to preserve the original posture while reproducing the external interaction of the hand and its self-interaction.

When the target position of the hand is determined, the joint angle of the elbow can also be computed similarly to the shoulder joint angle. We are

given the fixed elbow position $\mathbf{p}_e$, the initial position $\mathbf{p}_h^*$ of the hand and its target position $\mathbf{p}_h$. The vectors $\mathbf{v}_l^*$ and $\mathbf{v}_l$ representing the captured and the target forearm directions can be obtained as $\mathbf{v}_l^* = \mathbf{p}_h^* - \mathbf{p}_e$ and $\mathbf{v}_l = \mathbf{p}_h - \mathbf{p}_e$, respectively. The quaternion $\mathbf{q}_l^r$ representing the rotation from the initial forearm direction to the target direction is

$$\mathbf{q}_l^r = e^{\mathbf{n}'\theta'/2}, \tag{5.13}$$

where $\mathbf{n}'$ is the rotation axis and the angle θ' between $\mathbf{v}_l^*$ and $\mathbf{v}_l$ are given as

$$\mathbf{n}' = \frac{\mathbf{v}_l \times \mathbf{v}_l^*}{\|\mathbf{v}_l \times \mathbf{v}_l^*\|}$$

and

$$\theta' = \cos^{-1}(\mathbf{v}_l \cdot \mathbf{v}_l^*),$$

respectively. Premultiplying $\mathbf{q}_l^r$ with the initial elbow joint angle yield the elbow joint angle which places the hand in the bounding 3-D balls. Finally, the joint angle of the wrist can be calculated so that it preserves the hand orientation.

Now, with both the captured and the computed limb postures available, we blend them together to obtain a realistic motion. For this purpose, we perform spherical linear interpolation between each captured joint orientation of a limb with its corresponding IK solution. Let $\mathbf{q}_{ik}$ and $\mathbf{q}_{ik}^*$ be the orientation of the kth joint in the ith limb obtained from the IK solver and that from

the captured posture. Then the blended joint angle $\mathbf{q}'_{ik}$ can be described by spherical linear interpolation as follows:

$$\begin{aligned} \mathbf{q}'_{ik} &= \text{slerp}(\mathbf{q}^*_{ik}, \mathbf{q}_{ik}, w_{ik}) \\ &= e^{w_{ik} \ln(\mathbf{q}_{ik} \mathbf{q}^{*-1}_{ik})} \mathbf{q}^*_{ik}, \end{aligned} \tag{5.14}$$

where w_{ik} is the maximum importance value of the kth joint of the ith limb. That is, for a shoulder/hip joint, w_{ik} is the maximum among the importance values of the end-effector interaction to the environment, and those of the self-interaction of the general end-effectors. For blending the elbow/knee joint angle, the larger between the importance value of the end-effector interaction and that of its self-interaction is used as the weight w_{ik}. As a result, the limb preserves the captured joint angle when the importance values of the interaction with the environment and the self-interaction are low. Otherwise, the target character is able to reproduce the captured interaction with the environment and the self-interaction in accordance with their importance values.

The non-penetration condition may be violated since the posture is blended regardless of the constraints. Thus the blended posture has to be adjusted explicitly to prevent unwanted penetration. Provided with the predefined external objects for each end-effector, this violation can be detected easily. Before penetrating an object, the end-effector touches the boundary of the object. Thus, the preferable position of the end-effector is the intersection point of the object boundary and the ray from the shoulder to the end-effector during

penetration. This position moves continuously on the object in accordance with the end-effector movement. The penetration problem can be effectively eliminated by adjusting the limb posture using the IK solver for limbs.

Chapter 6

Analysis of Temporal Constraints

In retargeting motions, we must preserve important temporal aspects of the motion along with spatial aspects. Gleicher [12] emphasizes the importance of avoiding the introduction of high-frequencies during adaptation. Both this work and the work of Lee and Shin [19] provide approaches for avoiding the addition of discontinuities during adaptation. Unfortunately, both schemes rely on examining durations of motions and therefore can only be applied in offline applications. In this chapter, we show that the approach presented in this thesis does not introduce unwanted discontinuities into the resulting motion.

To begin, we must assume that the initial motion is free of unwanted discontinuities. This assumption is not restrictive because the movement of the performer is continuous. Discontinuities may be introduced by noise in the capture process, but these are generally removed by the filtering process described in Chapter 3. The continuity of the initial motion applies to the captured joint angles, end-effector positions, distance of the end-effector from other segments, and distance of the elbow/knee.

Provided with smooth trajectories of segments and motion features, the proposed analysis method yields continuous output. For any continuous dis-

tance function, the suggested importance function gives continuous importance values as described in Chapter 4. In other words, the importance values are consistently changed to reflect the temporal proximity of end-effectors to the environment. Therefore, the importance values have inter-frame coherence. For example, as an end-effector is approaching an object in the environment, its distance from the object is monotonically decreasing. Similarly, the distance is monotonically increasing as the end-effector is departing from the object. When the end-effector touches (or passes by) the object, the monotonicity changes but the distance function is still continuous at that instance. The similar analogy can be also applied the importance of features due to the self-interaction. Moreover, the trajectories of the bounding 3-D balls are smooth, since both end-effector (or elbow/knee) and the other segments move smoothly.

For the proposed IK solver, the kinematic constraints are the positions of end-effectors, and the bounding 3-D balls. These constraints are specified at every frame as temporal constraints as shown above. Given continuous paths for the segments of the performer, the IK solver will provide continuous trajectories for the parameters. Achieving this requires the solver to make consistent changes. That is, similar inputs to the solver must provide similar outputs. To guarantee this consistency, the IK solver tries to find the solution in an online manner so that it is close to the filtered input posture, while satisfying the kinematic constraints.

Since the IK solver utilizes as input the reference motion data and the

importance values, we can exclude unexpected motion artifacts such as unwanted jerkiness. That is, enforced to minimize the change from the reference motion, the suggested IK solver tries to find an intended motion. Moreover, guided by the importance values for interaction with the environment, it also predicts the future temporal constraints and continuously pays attention to them for motion coherence.

Chapter 7

Experimental Results

For puppetry performance we use a MotionStar Wireless motion capture device from Ascension Tech, Inc. with 14 sensors and two extended range transmitters. Each of sensors detects the magnetic field emitted by a transmitter to report its position and orientation up to 144 times per second.

The prototype system has been deployed for production and used successfully to create a virtual character for a children's television program as well as a virtual news reporter. Both have been shown on Korean national television, called KBS. The frog-like creature shown in Figure 7.1(a) ('Pang-Pang') who regularly appears in a daily TV show for children to demonstrate his comic performance. Thanks to the capability of the system for synthesizing realistic motion in real-time, Pang-Pang and a real actor can interact with each other. Figure 7.1(b) shows a virtual character ('Aliang') who has performed the role of a news reporter for the election of Korea National Assembly. Even in a time-critical situation such as reporting interim election results, Aliang can accomplish his role successfully.

The skeleton used in the system has fifty-one degrees of freedom including fifteen revolute joints of three degrees of freedom and the position of the

(a) Pang-Pang

(b) Aliang

Figure 7.1: Virtual characters on air controlled by the prototype system

Table 7.1: The number of iterations in numerical solver with and without root position estimation

motion	#frames	the number of iterations			
		without		with	
		Blubby	Sally	Blubby	Sally
Walk	39	47	0	0	0
Throw	157	244	0	0	0
Jump	88	111	0	0	0
Handstand	211	266	38	0	0
Dance	591	1253	0	1	0
Total (61 Clips)	9692	15634	429	8	0

root and its orientation. The floor is modelled as a plane for all of the uses of the system to date.

To test the system's performance, we created two puppets specifically designed to provide challenging retargeting problems. The character named *long tall Sally* has long arms and legs, while a ball-shaped man called *Blubby* with extremely short legs. To perform experiments, 61 prerecorded motion clips were used as the input for motion retargeting.

Table 7.1 shows the number of iterations in numerical optimization with and without initial root position estimation. Statistics for five selected motion clips are given in the first five rows of the table. The total figures for 61 clips are shown in the last row. Since Sally has long arms and legs, she can reach the captured end-effector positions without moving its root position. Thus, the number of iterations for Sally is small even without initial root position estimation. However, with estimated initial root positions, the

Table 7.2: Timing data

motion	#frames	Blubby		Sally	
		elapsed time (msec)	per frame (msec)	elapsed time (msec)	per frame (msec)
Walk	39	61.3	1.613	53.4	1.406
Throw	157	246.9	1.583	218.6	1.401
Jump	88	148.5	1.707	122.3	1.417
Handstand	211	359.4	1.711	296.1	1.410
Dance	591	960.4	1.628	838.1	1.421
Total (61 Clips)	9692	16.286	1.680	1366.6	1.410

number of iterations decreases to zero for the test motion clips. The effect of initial root position estimation is more apparent for Blubby with short legs. In most cases, the suggested estimation algorithm finds the root position that makes the end-effectors reachable to their goal positions without any help of the numerical solver given in Section 5.2.

Table 7.2 gives an overall performance of the proposed online motion retargeting algorithm excluding rendering time. Timing information was obtained on a IBM compatible PC with an Intel® Pentium® 4 1700MHz processor and 512Mb memory. The execution time for each example mainly depends on the number of iterations in numeric optimization. The tables show real-time performance for each examples.

In Figure 7.2, a captured walking motion is applied to a character with various methods. The upper images of Figure 7.2 reveal artifacts due to the geometric inconsistency between the performer and the puppet. Since

the positions of the feet are not incorporated into the motion retargeting, the supporting foot is sliding. In contrast, the middle motion preserves the positions well. However, the motions of the arms look unrealistic, since the joint angles of the arms are overadjusted to preserve the positions of the hands. The bottom figure is generated by the motion retargeting. The supporting foot is fixed at the proper position without sliding, and the joint angles of the arms are preserved as the original ones.

With conventional approaches based on joint angle preservation, there would also exist foot-sliding artifacts when the character has longer limbs, as given in the top of Figure 7.3. The middle image exhibits unintended bending of legs due to position preservation and an ill-selected initial root position. By assigning low importance values to the hands and offsetting the root position, we have a better result in which the legs are not bent as shown in the bottom figure.

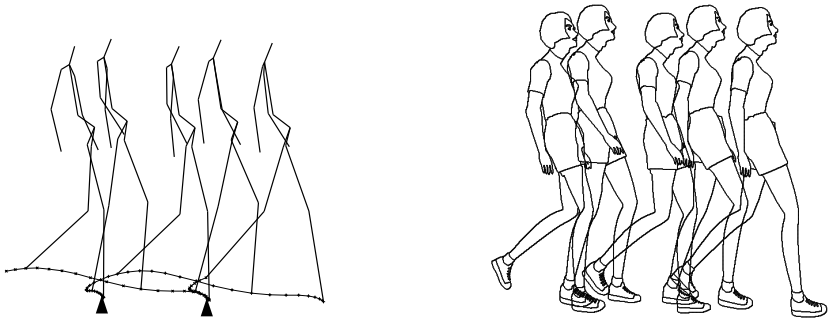
More examples are given in Figures 7.4 through 7.6. In particular, Figure 7.5 shows the motions such as crawling and picking up a which exhibit interaction of hands with objects in addition to that of feet. In Figure 7.6, motions including self-interaction are illustrated. We draw a selected frame from each movie clip to show the resulting posture clearly.

(a) the captured joint angles only

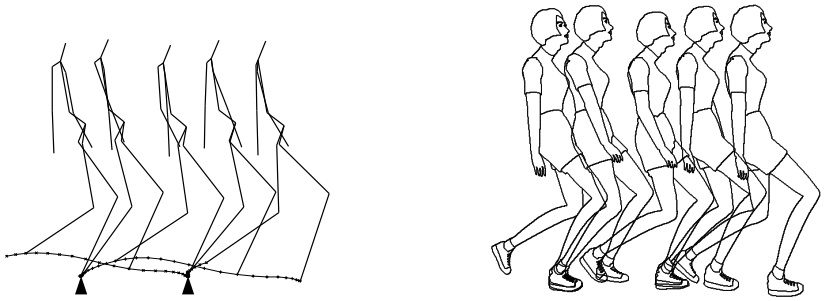
(b) a conventional IK solution with kinematic constraints on end-effectors

(c) Proposed algorithm combining the captured joint angles and the IK solution

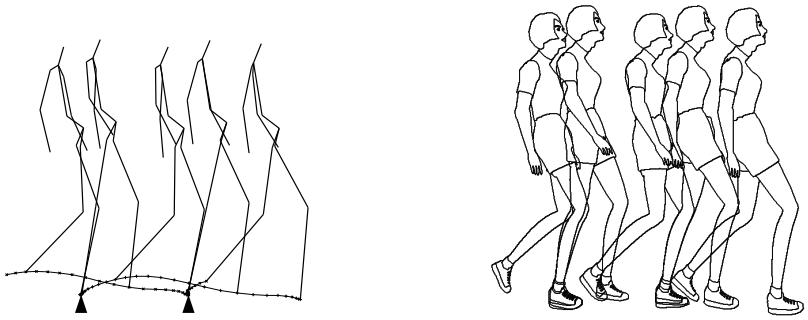
Figure 7.2: Walking motion of *Blubby*



(a) the captured joint angles only

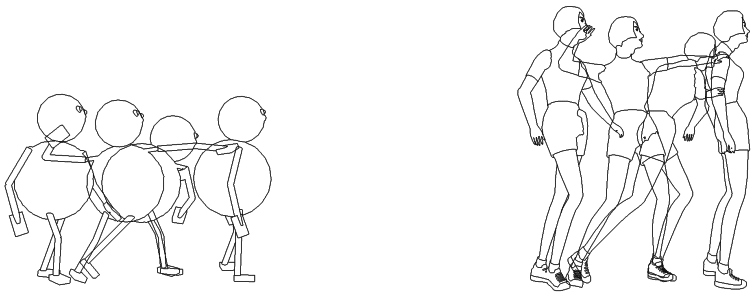


(b) a conventional IK solution with kinematic constraints on end-effectors

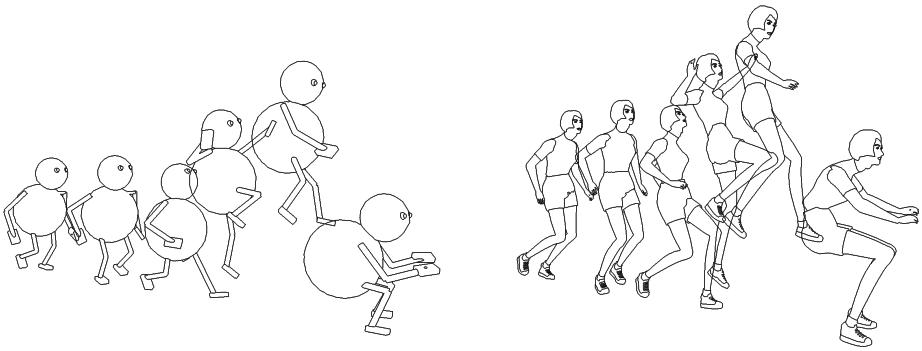


(c) Proposed algorithm combining the captured joint angles and the IK solution

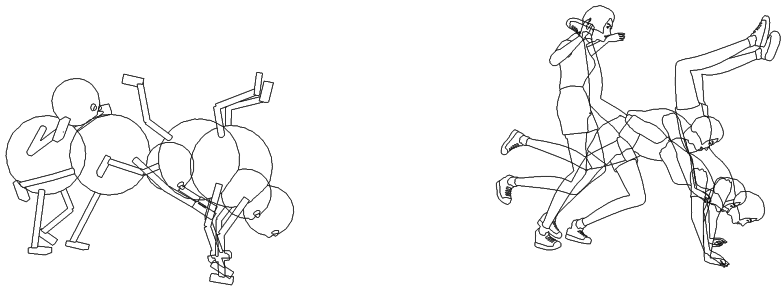
Figure 7.3: Walking motion of *Sally*



(a) Throwing

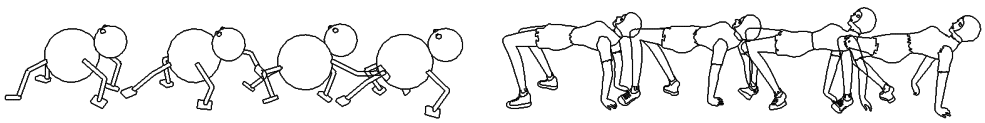


(b) Jumping

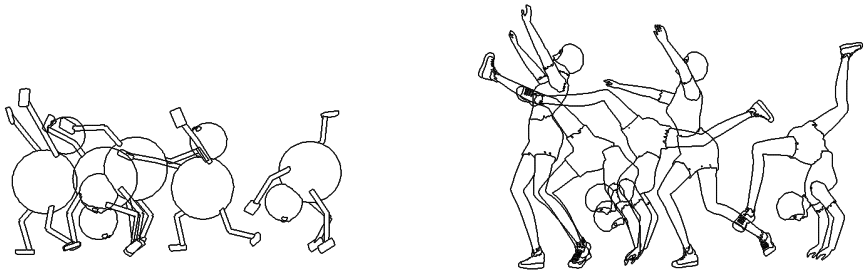


(c) Handstand

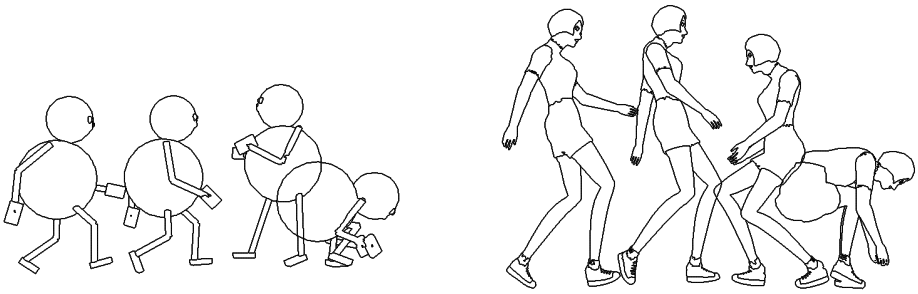
Figure 7.4: Example motions of *Blubby* and *Sally*



(a) Crawling



(b) Back-flipping



(c) Picking a box up

Figure 7.5: Example motions with interaction of hands

(a) Smoking

(b) Hands on waist

(c) Clapping hands

Figure 7.6: Example motions with self-interaction

Chapter 8

Conclusion

We have presented a new approach for online motion retargeting that transforms motions of a performer to a virtual character of a different size and shape. Introducing the notion of the importances to determine which aspects of the motion must be kept, we have been able to generate realistic motion for a character in real-time while preserving the characteristics of captured motions as much as possible.

The proposed motion analysis automatically computes the importance values of the captured end-effector positions and the relative positions of the limb segments with respect to the other segments, to preserve more important ones among them while altering the others. Based on a number of heuristics, we measure the interactivity of the character with its surrounding environment and the self-interactivity among its segments to compute the importance values of the captured end-effector positions and the relative positions of the limb segments, respectively. By including such interactions and transferring important features, we can dramatically broaden the types of characters as well as the repertoire of the motions which can be reproduced realistically.

Moreover, the notion of importance gives reasonable lookahead capability useful for avoiding jerkiness in motion, even when we do not know the future data. However, unlike full-scale space-time optimization [12] which observe the whole motion clip at once and adopt time consuming optimization techniques, the proposed approach has limited lookback capability implicitly achieved by the Kalman filter, and allows only a limited repertoire of constraints.

In this thesis, we proposed a novel inverse kinematics solver which solves a number of geometric constraints simultaneously. By dividing the problem into subsproblems, we can guarantee a real-time performance of the solver. However, the suggested inverse kinematics solver is specialized for human-like characters to insure real-time performance, although it can be easily adapted to other types of creatures with limbs.

KBS (Korean Broadcasting System), the largest public television broadcasting company in Korea, has been adopting a part of the suggested computer puppetry algorithm to control the virtual character, Pang Pang in a daily TV show for children. This show has become one of the favorites among children partly due to Pang Pang's successful performance. KBS also successfully showed the performance of a virtual reporter, Aliang for the real election using this algorithm.

In addition to such onair broadcasting, the computer puppetry algorithm is useful to the applications which require real-time motion adaptation. Those applications include the previewing motion capture data, controlling avatars

in virtual reality applications, and animating the characters in video games. Furthermore, this algorithm can be adopted as a real-time motion refining tool for the emerging researches on motion synthesis from prescribed motion.

In this approach, we focus on handling only the geometric discrepancy between a performer and a puppet. To generate more realistic motions, computer puppetry should also incorporate the characteristics of the puppet. Anthropomorphized animals such as cartoon-like birds and monkeys have their unique characteristics of motions. Those motions can hardly be captured directly from a human performer, and thus give an additional barrier to overcome.

Appendix A

Finding the Closest Point on the Intersection of Spheres

As given in Section 5.1, there are three types of surface elements: spheres, circles, and vertices. We describe how we find the closest point on each type of element to a given point $\mathbf{p}$. It is trivial to find the closest point on a sphere to the given point. Therefore, we proceed directly to the other cases.

Now, consider the closest point on a circle to $\mathbf{p}$. We start with how to construct the circle C, which is the common intersection of the two spheres S_1 and S_2. The radius r_c of C can be computed with Pythagorean theorem. Let $\mathbf{c}_{s_i}$ and r_{s_i} for $i = 1, 2, 3$ be the center of the sphere S_i and its radius, respectively. The radius r_c of C satisfies the following equations:

$$r_c^2 + x^2 = r_{s_1}^2, \text{ and} \tag{A.1}$$

$$r_c^2 + (||\mathbf{d}|| - x)^2 = r_{s_2}^2, \tag{A.2}$$

where x is the distance between the center $\mathbf{c}_c$ of C and that of S_1, and $\mathbf{d}$ is

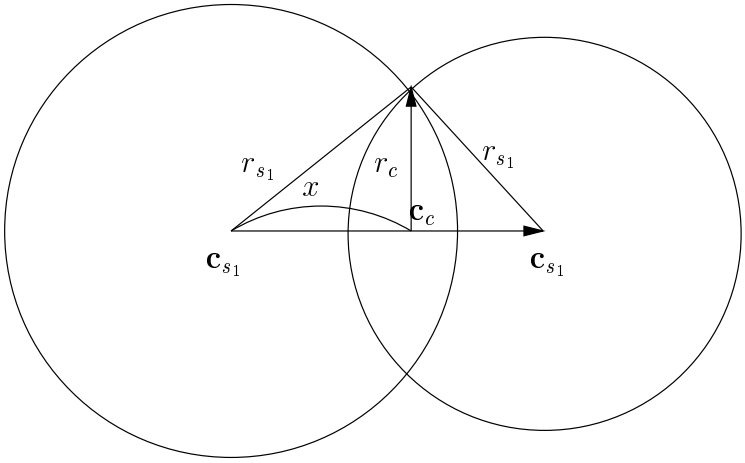


Figure A.1: Intersection of two spheres

the vector from $\mathbf{s}_1$ to $\mathbf{s}_2$. Solving those equations, we get

$$r_c^2 = r_{s_1}^2 - \frac{(r_{s_1}^2 - r_{s_2}^2 + ||\mathbf{d}||^2)^2}{4||\mathbf{d}||^2}. \tag{A.3}$$

Here S_1 and S_2 intersect unless r_c^2 is negative. From Equations (A.1) and (A.2),

$$x = \frac{r_{s_1}^2 - r_{s_2}^2 + ||\mathbf{d}||^2}{2||\mathbf{d}||}. \tag{A.4}$$

Thus,

$$\mathbf{c}_c = \frac{r_{s_1}^2 - r_{s_2}^2 + ||\mathbf{d}||^2}{2||\mathbf{d}||} \cdot \frac{\mathbf{d}}{||\mathbf{d}||} + \mathbf{c}_{s_1}. \tag{A.5}$$

Let $\mathbf{n}$ be the normal vector of the plane where the circle lies. Then,

$$\mathbf{n} = \frac{\mathbf{d}}{||\mathbf{d}||}. \tag{A.6}$$

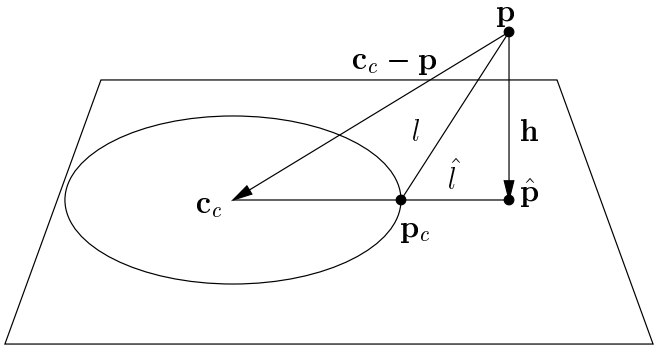


Figure A.2: Closest point from a point to a circle

We are ready to find the closest point on the circle C to the given point $\mathbf{p}$. Let $\mathbf{h}$ be the projection of the vector $\mathbf{c}_c - \mathbf{p}$ onto the normal vector $\mathbf{n}$ of the plane, that is, $\mathbf{h} = [\mathbf{n} \cdot (\mathbf{c}_c - \mathbf{p})]\mathbf{n}$. Then, the closest point $\mathbf{p}_c$ on C to $\mathbf{p}$ is

$$\mathbf{p}_c = \mathbf{c}_c + \frac{\hat{\mathbf{p}} - \mathbf{c}_c}{||\hat{\mathbf{p}} - \mathbf{c}_c||} r_c. \tag{A.7}$$

where $\hat{\mathbf{p}} = \mathbf{p} + \mathbf{h}$, that is, $\hat{\mathbf{p}}$ is the projection of $\mathbf{p}$ onto the plane containing C. As shown in Figure A.2, the distance l from $\mathbf{p}$ to $\mathbf{p}_c$ is $\sqrt{||\mathbf{h}||^2 + \hat{l}^2}$, where $\hat{l}$ is the distance from $\hat{\mathbf{p}}$ to $\mathbf{p}_c$, that is, $\hat{l} = ||\hat{\mathbf{p}} - \mathbf{c}_c|| - r_c$.

Finally, we show how to find the closest among vertices, if any, to the given point $\mathbf{p}$. Given those vertices, it is trivial to find the closest. Thus, we focus on explaining how to compute the vertices lying at the corners of the common intersection of three spheres, S_1, S_2 and S_3. We first calculate the intersection circle C_1 of two spheres S_1 and S_2. Cutting the sphere S_3 with the plane containing C_1, we have the circle C_2. Provided with the center

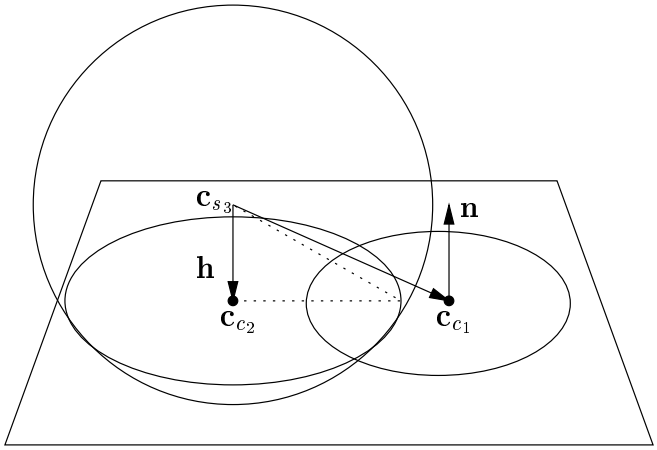


Figure A.3: Intersection of a sphere and a plane

point $\mathbf{c}_{c_1}$ of C_1 and the normal vector $\mathbf{n}$ of the plain containing the circle C_1, the center point $\mathbf{c}_{c_2}$ of C_2 is the projection of the center point $\mathbf{c}_{s_3}$ of the sphere S_3 onto the plane. Thus,

$$\mathbf{c}_{c_2} = \mathbf{c}_{s_3} + \mathbf{h}, \tag{A.8}$$

where $\mathbf{h}$ is the vector from $\mathbf{c}_{s_3}$ to $\mathbf{c}_{c_2}$ on the plane, that is, $\mathbf{h} = [\mathbf{n} \cdot (\mathbf{c}_{c_1} - \mathbf{c}_{s_3})]\,\mathbf{n}$. The radius r_{c_2} of C_2 is given as follows:

$$r_{c_2}^2 = r_{s_3}^2 - ||\mathbf{h}||^2, \tag{A.9}$$

where r_{s_3} is the radius of the sphere S_3. The sphere S_3 does not touch the plane if $r_{c_2}^2$ has a negative value. Two vertices determined by three spheres are the intersection of the circles C_1 and C_2. To compute the intersection

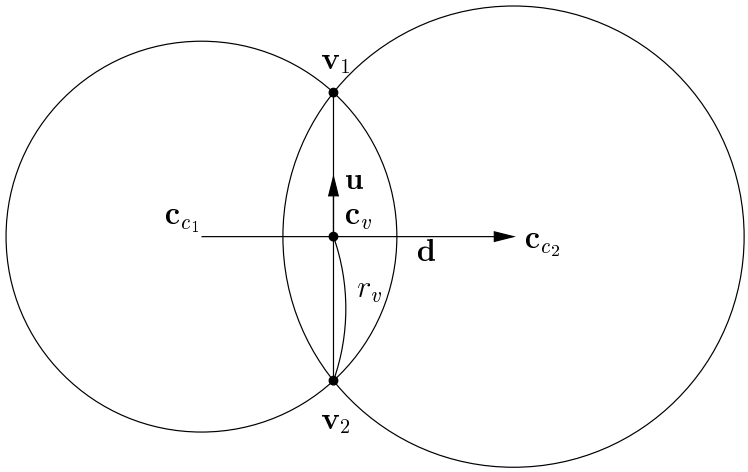


Figure A.4: Intersection of two circles

of C_1 and C_2, we evaluate the mid-point $\mathbf{c}_v$ of the vertices $\mathbf{v}_1$ and $\mathbf{v}_2$ (see Figure A.4.) Similarly to the sphere-sphere intersection, the mid-point $\mathbf{c}_v$ and the distance r_v from each of vertices to $\mathbf{c}$ are given as follows:

$$r_v^2 = r_{c_1}^2 - \frac{(r_{c_1}^2 - r_{c_2}^2 + ||\mathbf{d}||^2)^2}{4||\mathbf{d}||^2}, \text{ and} \tag{A.10}$$

$$\mathbf{c}_v = \frac{r_{c_1}^2 - r_{c_2}^2 + ||\mathbf{d}||^2}{2||\mathbf{d}||} \cdot \frac{\mathbf{d}}{||\mathbf{d}||} + \mathbf{c}_{c_1}, \tag{A.11}$$

where the $\mathbf{d}$ is the vector from the $\mathbf{c}_{c_1}$ to $\mathbf{c}_{c_2}$. The normalized direction vector $\mathbf{u}$ from $\mathbf{c}_v$ to $\mathbf{v}_1$ is obtained from the cross product of $\mathbf{n}$ and $\mathbf{d}$, that is, $\mathbf{u} = \frac{\mathbf{n}\times\mathbf{d}}{||\mathbf{n}\times\mathbf{d}||}$. Hence, we have the vertices $\mathbf{v}_1 = \mathbf{c}_v + r_v\mathbf{u}$ and $\mathbf{v}_2 = \mathbf{c}_v - r_v\mathbf{u}$.

요 약 문

실시간 동작 변형 및 대입을 위한 중요도 기반 접근

본 논문에서는 실제 연기자의 동작을 실시간으로 가상 캐릭터에 대입하기 위한 기법을 다룬다. 특히, 가상 캐릭터의 크기나 형태가 실제 연기자와 다른 경우에 동착 포착 장비를 통해 얻어진 실제 연기자의 동작 정보를 가상 캐릭터에 적합하도록 적절히 변형하여 자연스러운 동작을 얻기 위한 해결책을 제시한다. 매 순간 입력되는 실제 연기자의 동작을 분석하여 동작에서 중요한 특징을 판단하고 이를 최대한 유지하게 함으로써 자연스러운 동작을 실시간으로 생성한다. 칼만 필터를 적용하여 동작 포착 과정에서 나타나는 잡음을 제거하고 실제 사람의 외부 물체와의 상호작용이나 자가 상호작용에 기반하여 최종적인 가상 캐릭터의 동작에 반드시 반영되어야 하는 중요한 특징을 분석하는 기법을 제안한다. 그리고 중요한 특징을 보존하는 최종 동작을 실시간으로 생성하기 위해서 효율적인 역운동학 기법을 소개한다. 본 논문에서 제안하는 실시간 동작 변형 및 대입 기법은 실제 방송에 사용되어 성능을 입증한 바 있다.

References

[1] Ali Azarbayejani and Alex P. Pentland. Recursive estimation of motion structure, and focal length. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(6):562 – 575, 1995.

[2] Ronald Azuma and Gary Bishop. Improving static and dynamic registration in an optical see-through hmd. In *Proceedings of SIGGRAPH 94*, pages 197–204, 1994.

[3] N. Badler, M. J. Hollick, and J. P. Granieri. Real-time control of a virtual human using mininal sensors. *PRESENCE*, 2(1):82–86, 1993.

[4] Rama Bindiganavale and Normal I. Badler. Motion abstraction and mapping with spatial constraints. In *Proceedings of International Workshop, CAPTECH'98*, pages 70–82, 1998.

[5] Boddy Bodenheimer, Charles Rose, Seth Rosenthal, and John Pella. The process of motion capture: Dealing with the data. In *Proceedings of the Eurographics Workshop on Computer Animation and Simulation '97*, 1997.

[6] Ted J. Broida and Rama Chellappa. Estimation of object motion parameters from noisy images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(1):90–99, 1986.

[7] Kwang-Jin Choi and Hyeong-Seok Ko. On-line motion retargetting. *Journal of Visualization and Computer Animation*, 11:223–243, 2000.

[8] Ascension Technology Corporation. *Motion Star Installation and Operation Guide*. Ascension Technology Corporation, 1996.

[9] Martin Friedmann, Thad Starner, and Alex Pentland. Synchronization in virtual realities. *PRESENCE*, 1(1):139–144, 1991.

[10] Michael Girard and Anthony A. Maciejewski. Computational modeling for the computer animation of legged figures. In *Proceedings of SIGGRAPH 85*, pages 263–270, 1985.

[11] Michael Gleicher. Motion editing with spacetime constraints. In *Proceedings of 1997 Symposium on Interactive 3D Graphics*, pages 139–148, 1997.

[12] Michael Gleicher. Retargeting motion to new characters. In *Proceedings of SIGGRAPH 98*, pages 33–42, 1998.

[13] Vijaykumar Gullapalli, Jack J. Gelfand, Stephen H. Lane, and Wade W. Wilson. Synergy-based learning of hybrid position/force control for redundant manipulators. In *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, 1996.

[14] Chris Hanson. MIT scheme reference manual. Technical Report AITR-1281, 1991.

[15] Soonki Jung and Kwangyun Wohn. Tracking and motion estimation of the articulated object: a hierarchical kalman filter approach. *Journal of Real-Time Imaging*, 3(6), 1997.

[16] Myoung-Jun Kim, Sung Yong Shin, and Myung-Soo Kim. A general construction scheme for unit quaternion curves with simple high order derivatives. In *Proceedings of SIGGRAPH 95*, pages 369–376, 1995.

[17] Y. Koga, K. Kondo, J. Kuffer, and J. Latombe. Planning motions with intentions. In *Proceedings of SIGGRAPH 94*, pages 395–408, 1994.

[18] J. U. Korein and N. I. Badler. Techniques for generating the goal-directed motion of articulated structures. *IEEE Computer Graphics & Application*, 2:71–81, 1982.

[19] Jehee Lee and Sung Yong Shin. A hierarchical approach to interactive motion editing for human-like figures. In *Proceedings of SIGGRAPH 99*, pages 39–48, 1999.

[20] T. Molet, R. Boulic, and D. Thalmann. A real-time anotomical converter for human motion capture. In *Proceedings of 7th Eurographics Workshop on Animation and Simulation*, 1996.

[21] T. Molet, R. Boulic, and D. Thalmann. Human motion capture driven by orientation measurements. *PRESENCE*, 8(2):187–203, 1999.

[22] B. Paden. *Kinematics and Control Robot Manipulators*. PhD thesis, University of California, Berkeley, 1986.

[23] Zoran Popovic and Andrew Witkin. Physically based motion transformation. In *Proceedings of SIGGRAPH 99*, pages 11–20, 1999.

[24] Protozoa. Technology information. *http://www.protozoa.com/ Page_2 /info_index.html*, 1999.

[25] Charles F. Rose, Brian Guenter, Bobby Bodenheimer, and Michael F. Cohen. Efficient generation of motion transitions using spacetime constraints. In *Proceedings of SIGGRAPH 96*, pages 147–154, 1996.

[26] D.J. Sturman. Computer puppetry. *IEEE Computer Graphics & Applications*, 18(1):38–45, 1998.

[27] D. Tolani, A. Goswami, and N. Badler. Real-time inverse kinematics techniques for anthropomorphic limbs. *Graphical Models*, 62(5), 2000.

[28] Greg Welch and Gary Bishop. Scaat: Incremental tracking with incomplete information. In *Proceedings of SIGGRAPH 97*, pages 333–344, 1997.

[29] D. J. Whitney. Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine System*, pages 47–53, 1969.

[30] J. Zhao and N. I. Badler. Inverse kinematics positioning using nonlinear programming for highly articulated figures. *ACM Transactions on Graphics*, 13(4):313–336, 1994.

감사의 말

논문이 완성되기까지 물심 양면으로 도와주신 여러분들께 감사를 드립니다. 7년간이나 부족한 저를 지도해 주시고 논문의 마지막 한자 까지 꼼꼼히 지도해 주신 신성용 교수님께 감사드립니다. 학부 과정 부터 항상 저를 지켜봐 주시고 부족한 점을 깨우쳐 주신 좌경룡 교수님, 원광연 교수님께 감사드립니다. 바쁘신 일정 중에도 저의 논문을 심사해 주시고 지도와 충고를 아끼지 않으신 오영환, 김홍오 교수님께 감사드립니다. 또한 저의 연구를 도와주시고 논문 작성과 수정에 도움을 주신 Michael Gleicher 교수와 이제희 선배에게 감사드립니다. 그리고 최근 수년 간 정신적으로 많은 도움을 아끼지 않으신 정유정님께도 깊은 감사드립니다. 따뜻한 동료애로 저의 연구실 생활을 도와주신 많은 선후배님들께도 심심한 감사를 드립니다. 마지막으로 그간 저를 키워주시고 한없는 사랑으로 보살펴주신 어머님과 멀리서 지켜봐주실 아버님께 감사드립니다.

이 력 서

성　　명 : 신 현 준

생 년 월 일 : 1973년 8월 27일

출 생 지 : 서울특별시

본　　적 : 서울특별시

학 력

1991.3–1995.2　한국과학기술원 전산학과 (B.S.)

1995.3–1997.2　한국과학기술원 전산학과 (M.S.)

1997.3–2002.2　한국과학기술원 전산학과 (Ph.D.)