7

Modeling, Animation, and Rendering of Human Figures

Uğur Güdükbay, Bülent Özgüç, Aydemir Memişoğlu
and Mehmet Şahin Yeşil

Department of Computer Eng., Bilkent University, 06800, Bilkent, Ankara, Turkey

Human body modeling and animation has long been an important and challenging area in computer graphics. The reason for this is two-fold. First, the human body is so complex that no current model comes even close to its true nature. The second reason is that our eyes are so sensitive to human figures that we can easily identify unrealistic body shapes (or body motions).

Today many fields use 3D virtual humans in action: video games, films, television, virtual reality, ergonomics, medicine, biomechanics, etc. We can classify all these applications into three categories: *film production for arts and entertainment*, *real-time applications*, such as robotics, video games and virtual environments, and *simulations*, such as computer-aided ergonomics for the automotive industry, virtual actors, biomedical research, and military simulations. The type of application determines the complexity of the models. For example video games or virtual reality applications require the lowest possible ratio between the computation cost and capabilities of the model. However, for biomedical research, realism is essential and the animated model should obey physical laws. Hence, the models are designed and animated according to the specific area in which they are applied.

Humans are an indispensable part of dynamic 3D scenes. Therefore, human face and body specific representations and animation techniques should be heavily used in a 3DTV framework to achieve the goals of real-time implementation and realism. Techniques of 3D motion data collection, such as motion capture, can be incorporated in human model animation. Continuous video and motion recording at high sampling rates produce huge amounts of data. Keyframe transmission that can be regenerated into continuous motion using interpolation techniques will reduce the size of the transmitted data significantly.

To study human modeling and animation, many techniques based on *kinematics*, *dynamics*, *biomechanics*, and *robotics* have been developed by researchers. In order to produce realistic animations, rendering is also an inseparable part. Furthermore, hair, garment, interaction of multiple avatars, expression of feelings, behavior under extreme conditions (such as accidents,

deep sea diving, etc.) and many more real life experiences make the problem as complicated as one's imagination.

The human body has a rigid skeleton. This is not the case with some other living, artificial or imaginary objects. If the animation aims at a particular instance of bone fracture, maybe for an orthopedical simulation, then the rules all of a sudden change. As long as the subject excludes these non-articulated body behaviors, there is a reasonable starting point, a skeleton that is an articulated object with joints and rigid elements. It is natural, then, to assume that if a proper motion is given to the skeleton, one can build up the rest of the body on top of this. Layers include muscles, skin, hair and garments that can somehow be realistically rendered based on skeleton motion, plus some external forces, such as wind and gravity, to add more realism, at least to hair and garment. This obviously is a reverse way of looking at things; it is the muscles that expand or contract to give motion to the skeleton, but if the ultimate aim is to generate a realistic animation visually, and if the muscles can be accurately modeled, the order in which the forces are originated can be reversed. This makes the skeletal motion to be the starting source of animation.

It is very difficult to fit all the aspects of human modeling and animation into a limited scope of a book chapter. Thus, this chapter discusses some aspects of human modeling, animation, and rendering, with an emphasis on multi-layered human body models and motion control techniques for walking behavior.

7.1 Articulated Body Models

Since the 1970s, researchers have proposed several different approaches for the realistic modeling of the human body and its movements. Human body modeling first consists of the basic structural modeling. This includes the definition of joints and segments, their positions and orientations, and the hierarchy between these components. It also includes defining the volume of the body which is composed of muscles, fat, and skin. The second part of the problem, simulating human motion, is a complex task. It is very difficult to take into account all the interactions with the environment involved in a simple movement. A realistic human model should provide accurate positioning of the limbs during motion, realistic skin deformations based on muscles and other tissues, realistic facial expressions, realistic hair modeling, etc.

In the early stages, humans were represented as stick figures, simple articulated bodies made of segments and joints. These articulated bodies were simulated using methods based on kinematics. More recently, *dynamic methods* have been used to improve the realism of the movement. However, since the human body is a collection of rigid and non-rigid components that are very difficult to model, dynamic and kinematics models did not meet the need. Consequently, researchers began to use human anatomy to produce human

models with more realistic behaviors. The models proposed can be divided into four categories: *stick figure models*, *surface models*, *volume models*, and *multi-layered models* (see Fig. 7.1).

7.1.1 Stick Figure Models

Early studies on human body modeling and animation date back to the seventies. Badler and Smoliar [1] and Herbison-Evans [2] proposed 3D human models based on volumetric primitives, such as spheres, prisms, or ellipsoids. The technological limitations allowed stick figures with only a few joints and segments and simple geometric primitives. These models are built by using a hierarchical set of rigid segments connected at joints. The complexity of these articulated figures depends on the number of joints and segments used. The motions were usually specified as a set of hierarchical transformations, controlled by the joint constraints so that the members will not break from each other. Studies on directed motions of articulated figures by Korein [3] and the stick figure model by Thalmanns [4] are representative of this category.

7.1.2 Surface Models

Surface models were proposed as an improvement on the stick models. A new layer, representing human skin, was introduced in addition to the skeleton layer [5]. Therefore, this model is based on two layers: a skeleton, which is the backbone of the character animation and a skin, which is a geometric envelop of the skeleton layer. Deformations in the skin layer are governed by the motion of the skeleton layer. The skin layer can be modeled by using points and lines, polygons (used in Rendezvous a Montreal [6]), and curved surface patches (e.g., Bézier, Hermite, and B-splines).

In the case of polygons, the skin layer is deformed by attaching each mesh vertex to a specific joint. In this way, the motion of the skin layer follows the

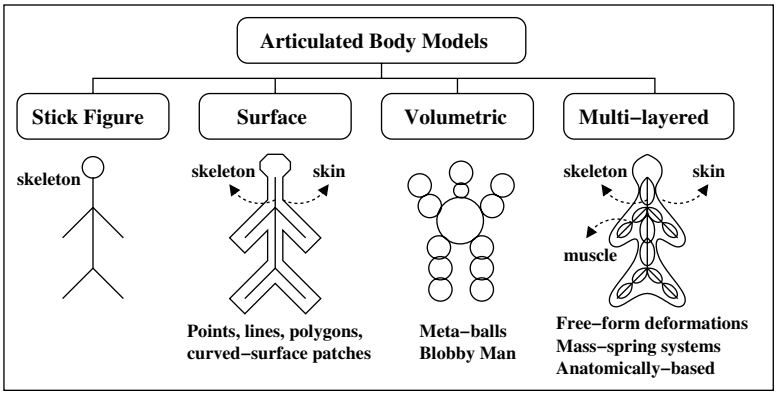


Fig. 7.1. Taxonomy of articulated body models

motion of the articulated structure. However muscle behavior is not modeled in this approach and body parts may disconnect from each other during some motions. In spite of these deficiencies, these models are still very common in Web applications.

A solution to the skin deformation problem is to use a continuous deformation function based on joints angles. This method is first used by Komatsu [7] to deform the control points of biquartic Bézier patches. Thalmanns introduced the concept of Joint-dependent Local Deformation (JLD) [8]. In both approaches, the skin is deformed algorithmically. First, the skin vertices are mapped to the corresponding skeleton segments in order to limit the influence of the joint connecting the segments. Next, a function of the joint angles is used to deform the vertices. These studies showed that specialized algorithms may help to achieve more realistic skin deformations but there are two limitations. First, an algorithmic deformation is basically a mathematical approximation, mostly far away from the physical behavior of a model under various forces; thus algorithmic deformations cannot always accurately describe complex skin deformations of a real human body. Second, a graphic designer cannot easily direct the deformations because they are specified via an algorithm.

Stitching is the process of attaching a continuous mesh to a bone structure. In rigid body animation, polygons are attached to the bones. The polygons are transformed by changing the matrix representing the corresponding bone. In stitching, each vertex of a polygon can be attached to a different bone. Therefore, each vertex can be transformed by a different matrix representing the bone to which the vertex is attached. Breaking up skin in this way so that the vertices are in the local space of the bone to which they are attached simplifies the process of stitching. This type of attachment enables us to create a single polygon that "stitches" multiple bones by simultaneously attaching different vertices to different bones. A polygon must fill the gap formed as a result of a manipulation of the bones [9].

Although stitching is a useful technique, it has some problems. Unnatural geometries appear during extreme joint rotations. For example, rotating a forearm by 120 degrees using the stitching technique results in a shear effect at the elbow. A solution is to allow a vertex to be affected by more than one bone. This is called *full skinning* and is compatible with the behavior of the human body. In a real human, the skin on the elbow is not affected by a single bone but by both the upper and lower arms. In order to implement this, we must know the bones effecting each skin vertex and the weight for each bone specifying the amount of the effect. The position of each vertex is calculated using (7.1) [9].

$$new\_vertex\_position = \sum_{i=0}^{N-1} weight_i \times matrix_i \times vertex\_position, \qquad (7.1)$$

where $weight_i$ is the weight for bone i and $matrix_i$ is the matrix used to transform the vertices attached to bone i. In the case of linear skinning, the sum of all weights is 1.0.

Skinning is a simple technique that has a very low computational cost. For this reason, it is used in video games. Current research focuses on improving the skinning procedure by increasing the speed of computations. Sun et al. [10] use the concept of normal-volume, i.e., they reduce the computational cost by mapping a high-resolution mesh onto a lower-resolution control mesh. In this way, the high-resolution object can be deformed by skinning the low-resolution surface control structure. Singh and Kokkevis choose surface-based Free-Form Deformations (FFDs) to deform skin [11]. A very useful property of the surface-oriented control structures is that they bear a strong resemblance to the geometry they deform. In addition, they can be automatically constructed from deformable objects.

Assigning weights is a semi-automatic process that requires a huge amount of human intervention; this significantly limits the skinning technique. In addition, a combination of weights for highly mobile portions of the skeleton may be very appropriate for one position of the skeleton but the same weights may not be acceptable for another position. Therefore, there is no single combination of weights that provides an acceptable result for all parts of the body. In spite of these deficiencies, the skinning method remains one of the most popular techniques for skin deformation because of its simplicity.

Using predefined keyshapes is another approach for skin deformation [12, 13]. *Keyshapes* are triangular meshes in some skeletal positions. They are obtained via a digitization procedure [14]. The idea behind this technique is that new shapes are created by interpolation or extrapolation. The deformation-by-keyshapes technique differs from 3D morphing algorithms in that it is limited to smooth interpolation problems. However this approach does not have the deficiencies of the skinning techniques and it performs better than multi-layer deformation models. There is no limitation on the number of keyshapes, making the technique quite flexible.

7.1.3 Volumetric Models

Controlling the surface deformation across joints is the major problem of surface models. In *volume models*, simple volumetric primitives like ellipsoids, spheres and cylinders are used to construct the body shape. A good example of volume models is metaballs. Metaballs are volumes that can join each other based on a function of nearness. They can do a better job than surface models but it is really hard to control a large number of primitives during animation. In the very early stages of computer graphics, volumetric models were built from geometric primitives such as ellipsoids and spheres to approximate the body shape. These models were constrained by the limited computer hardware available at the time. Along with the advances in computer

hardware technology, implicit surfaces are used as an improvement on these early models. Today, volumetric models are often able to handle collisions.

An *implicit surface* is also known as an iso-surface. It is defined by a function that assigns a scalar value to each 3D point in space. Then an iso-surface is extracted from the level set of points that are mapped to the same scalar value. Skeletons, which are constructed from points and lines, are the source of the scalar field. Each skeleton produces a potential field whose distribution is determined by the field function. For the field function it is common to use a high-order polynomial of distance to the skeleton (generally higher than 4^{th} order). This approach is known as *metaball formulation*.

Because they are naturally smooth, implicit surfaces are often used in the representation of organic forms. One of the first examples of this method is a "blobby man" created by Blinn [15]. It is generated from an implicit surface that is constructed using an exponentially decreasing field function. In one of the later studies, Yoshomito shows that a complete, realistic-looking, virtual human body can be created with metaballs at a low storage cost [16]. A more complicated implicit formulation is introduced by Bloomenthal [17].

Implicit surfaces have many properties that promote successful body modeling. Their most useful property is continuity, which is the main requirement for obtaining realistic shapes. There are two more advantages worth mentioning: first, due to the compact formulation of the field functions, little memory is required and second, they are simple to edit since they are defined by point or polygon skeletons. However, undesired blending may be observed during animations. Hybrid techniques that are mixtures of surface deformation models and implicit surfaces are proposed as a solution to this problem [18, 19].

Some volumetric models are adept at handling collisions between different models or different parts of the same model and at generating deformed surfaces in parallel. Elastic properties are included in the formulation of distance-based implicit surfaces by Cani-Gascuel [20]. In this work, a correspondence between radial deformation and the reaction force is established. A non-linear finite element model of a human leg derived from the Visible Human Database [21] is recently proposed by Hirota et al. [22]. It also achieves a high level of realism in the deformation.

7.1.4 Multi-layered Models

Lasseter emphasized that computers provide the advantage of building up an animation in layers to create complex movements [23]. The animator specifies different kinds of constraint relationships between different layers. Then, s/he can control the global motion from a higher level.

The most common approach is to start with the skeleton and then add the muscles, skin, hair and other components. A skeleton layer, a muscle layer, and a skin layer are the most common. This *layered modeling technique* is heavily used in human modeling [24]. In this approach, motion control techniques are applied to the skeleton layer and the other layers move accordingly.

The layered approach has been accepted both in the construction and animation of computer generated characters. There are two types of models. The first relies on a combination of ordinary computer-graphics techniques like skinning and implicit surfaces, and tends to produce a single layer from several anatomical layers [24]. The other group, inspired by the actual biology of the human body, tries to represent and deform every major anatomical layer and model their dynamic interplay [25, 26].

The *skeleton layer* is an articulated structure that provides the foundation for controlling movement [19]. Sometimes, the articulated structure is covered by material bones approximated by simple geometric primitives [22, 27, 28]. Studies on the skeleton layer focus on the accurate characterization of the range of motion of specific joints. Different models of joint limits have been suggested in the literature: Korein uses spherical polygons as boundaries for the spherical joints like the shoulder [3] whereas Maurel and Thalmann use joint sinus cones for the shoulder and scapula joints [29].

In the early muscle models, the foundations for the muscles and deformations used in a muscle layer are based on *free form deformations (FFD)* [30]. Muscles construct a relationship between the control points of the FFDs and the joint angles. This method has two limitations: the possibility that the FFD box does not closely approximate the muscle shape and the fact that FFD control points have no physical meaning.

Moccozet models the behavior of the hand muscles using Dirichlet free form deformation; this is a generalization of FFD that provides a more local control of the deformation and removes the strong limitation imposed on the shape of the control box [31]. Since implicit surfaces provide the most appropriate method for modeling organic forms, they are widely used to model the muscle layer. In [32], implicit primitives like spheres and super-quadrics are used to approximate muscles. On the other hand, in [19], the gross behavior of bones, muscles, and fat are approximated by grouped ellipsoidal metaballs with a simplified quadratic field function. This technique does not produce realistic results for the highly mobile parts of the human body, in which each ellipsoidal primitive is simultaneously influenced by several joints.

The muscles that significantly effect the appearance of the human models are the *fusiform muscles* in the upper and lower parts of the legs and arms. These muscles have a fleshy belly, tapering at both extremities. Since an ellipsoid gives a good approximation of the appearance of a fusiform muscle, muscle models tend to use the ellipsoid as the basic building block when the deformation is purely geometric. Moreover, the analytic formulation of an ellipsoid provides scalability. When the primitive is scaled along one of its axes, the volume of the primitive can easily be preserved by adjusting the scaling parameters of the two remaining axes. Following a similar approach, the height-to-width ratio can be kept constant. This is the reason why a volume-preserving ellipsoid for representing a fusiform muscle is used in [25, 26].

A polyline called an "action line" is introduced by Nedel and Thalmann [33]. The action line is used for abstracting muscles. It represents the force produced

by the muscle on the bones, and on a surface mesh deformed by an equivalent mass-spring network. A noteworthy feature of this mass-spring system is the introduction of angular springs that smooth out the surface and control the volume variation of the muscle.

B-spline solids can also be used for modeling muscles, as described by Ng-Thow-Hing and Fiume [34]. These have the ability to capture multiple muscle shapes (fusiform, triangular, etc.). They use three curves to characterize the attachment of musculotendon onto a skeleton: *origin*, *insertion*, and *axial*. The origin and insertion curves represent the attachment areas of the tendons to the bones. The axial curve represents the line of action of the muscle.

Polygonal [22, 24, 25], parametric [19, 26, 35], subdivision [18, 36], and implicit [17, 37] surfaces have been used for modeling the skin. Polygonal surfaces are processed by the graphics unit and are the best choice when speed and/or interactivity are needed. However, some surface discontinuities, which need to be smoothed out, may arise when polygonal surfaces are used.

Parametric surfaces yield very smooth shapes. This makes them very attractive for modeling the skin. Shen and Thalmann [19] derive a lower degree polynomial field function for the inner layers. Implicit surfaces are very good at representing organic forms. The main limitation of implicit surfaces is that it is difficult or impossible to apply texture maps for realistic rendering. Therefore, they are very seldom used to directly extract a skin but are used frequently for invisible anatomical layers.

Subdivision surfaces best represent the skin layer. They have several advantages: i) smoothness can be guaranteed by recursively subdividing the surface, ii) a polygonal version suitable for rendering is automatically derived without further computations, and iii) interpolating schemes can be used [36].

There are three ways of deforming the skin in multi-layered models [38]:

- First, surface deformation models are applied to the skin. Then, the skin is projected back onto the inner anatomical layers.
- A mechanical model is used to deform the skin while keeping the skin a certain distance away from the material beneath.
- The skin is defined as the surface of a volume-finite element (mass-spring model of the body).

7.2 Exchangeable Articulated Models

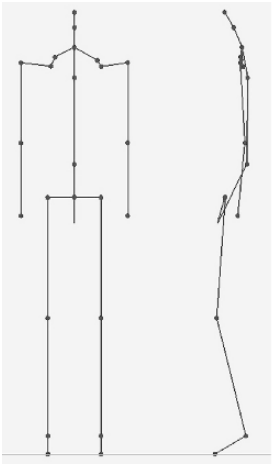
Although modeling the basic structure of the articulated figure is a trivial part of human modeling, it becomes a challenging task when there is no standard for it. The H-Anim 1.1 Specification is the usual standard for human body modeling; it defines the geometry and the hierarchical structure of the human body [39]. The *Humanoid Animation (H-Anim)* standard was developed by the Humanoid Animation Working Group of Web3D Consortium to define interchangeable human models. The H-Anim standard specifies how to define

humanoid forms and behaviors in standard Extensible 3D Graphics/Virtual Reality Modeling Language (X3D/VRML). This group had flexibility as a goal so no assumptions are made about the types of applications that will use humanoids. They also had a goal of simplicity which led them to focus specifically on humanoids, instead of trying to deal with arbitrary articulated figures.

In the H-Anim 1.1 Specification, the human body is represented by a number of *segments* (such as forearm, hand, foot) that are connected to each other by *joints* (such as the elbow, wrist and ankle). The H-Anim structure contains a set of nodes to represent the human body. The nodes are `Joint, Segment, Site, Displacer,` and `Humanoid`. `Joint` nodes represent the joints of the body and they are arranged in a strictly defined hierarchy. They may contain other `Joint` and `Segment` nodes. `Segment` nodes represent a portion of the body connected to a joint. They may contain `Site` and `Displacer` nodes. `Site` nodes are placements for cloth and jewelry; they can also be used as end-effectors for inverse kinematics applications (see Subsection 7.3.1). `Displacer` nodes are simply grouping nodes, allowing the programmer to identify a collection of vertices as belonging to a functional group for ease of manipulation. The `Humanoid` node stores information about the model. It acts as a root node for the body hierarchy and stores all the references to `Joint, Segment,` and `Site` nodes. An H-Anim compliant human body is in the "at rest" position; all the joint angles are zero and the humanoid faces the +z direction, with +y being up and +x to the left of the humanoid according to the right-handed coordinate system.

A simple XML data format to represent the human skeleton can be defined by conforming to the hierarchy of joints and segments of the body as named in the H-Anim 1.1 Specification. XML's structured and self-descriptive format provides an excellent method for describing the skeleton. Front and side views of the skeleton are given in Fig. 7.2(a) and a portion of the XML representation for the skeleton is given in Fig. 7.2(b).

Although the data format can be used for representing the full skeleton specified in the H-Anim standard, the complete hierarchy is too complex for most applications. In real-time applications such as games, a simplified hierarchy will be more suitable. H-Anim 1.1 Specification proposes four "Levels of Articulation" that contain the subsets of the joints. The body dimensions and levels of articulation are suggested for information only and are not specified by the H-Anim standard. Levels of articulation are suggested both for simplicity and compatibility. Animators can share their animations and humanoids if they conform to the same level of articulation. "Level of Articulation Zero" has just a root joint. "Level of Articulation One" represents a typical low-end real-time 3D hierarchy that does not contain information about the spine and has a shoulder complex with insufficient detail. "Level of Articulation Two" contains a lot of necessary information about spine, skull and hands. "Level of Articulation Three" represents the full H-Anim hierarchy.



(a)

```
<Joint name="HumanoidRoot" center="0.1296 0.0000 -3.8563">
   <Joint name="sacroiliac" center="0.0075 0.0000 -4.2920">
      <Joint name="l_hip" center="-0.0005 -0.4446 -4.2920"
      limitx="-1.00 2.00" limity="-1.40 2.90" limitz="-1.40 1.30">
         <Joint name="l_knee" center="0.1441 -0.4446 -2.2778"
         limity="-3.00 0.00">
            <Joint name="l_ankle" center="-0.3444 -0.4446
            -0.3070">
               <Joint name="l_midtarsal" center="0.1722 -0.4446
               -0.0005" />
            </Joint>
         </Joint>
      </Joint>
      <Joint name="r_hip" center="0.0136 0.4446 -4.2920"
      limitx="-1.00 2.00" limity="-1.40 2.90" limitz="-1.40 1.30">
         <Joint name="r_knee" center="0.1488 0.4446 -2.2993"
         limity="-3.00 0.00">
            <Joint name="r_ankle" center="-0.3585 0.4446
            -0.3332">
               <Joint name="r_midtarsal" center="0.1722 0.4446
               0.0000" />
            </Joint>
         </Joint>
      </Joint>
</Joint>
```

(b)

Fig. 7.2. (**a**) Front and side views of the skeleton and (**b**) a portion of the XML representation for the skeleton

7.3 Motion Control Techniques

The different approaches of biomechanics, robotics, animation, ergonomics, and psychology are integrated to produce realistic motion-control techniques. Motion control techniques can be classified into two groups according to level of abstraction that specifies the motion; *low-level* and *high-level*. In low-level motion control, the user manually specifies the motion parameters such as position, angles, forces, and torques. In high-level motion control, the motion is specified in abstract terms such as "walk", "run", "grab that object", "walk happily", etc. [40]. In animation systems using high-level motion control, the low-level motion-planning and control tasks are performed by the machine. The animator simply changes some parameters to obtain different kinds of solutions. To generate realistic animations, both kinds of motion control techniques should be used in an integrated manner [41]. Human motion control techniques can be classified as *kinematics*, *dynamics*, and *motion capture*.

7.3.1 Kinematics

Kinematics, which originates from the field of robotics, is one of the approaches used in motion specification and control. Kinematics is the study of motion by considering position, velocity, and acceleration. It does not consider the underlying forces that produce the motion. Kinematics-based techniques animate

the articulated structures by changing the orientations of joints over time. Motion is controlled by the management of joint transformations over time.

In *forward kinematics*, the global position and orientation of the root of the hierarchy and the joint angles are directly specified to obtain different postures of an articulated body. The motion of the end-effector (e.g., the hand in the case of the arm) is determined by the joint transformations from the root of the skeleton to the end-effector. Mathematically, forward kinematics is expressed as

$$x = f(\Theta), \tag{7.2}$$

where Θ is the set of joint angles in the chain and x is the end-effector position. After the joint transformations are calculated, the final position of the end-effector is found by multiplying the transformation matrices in the hierarchy. For example, in the case of the leg, the position of the foot is calculated by using the joint angles of hip and knee.

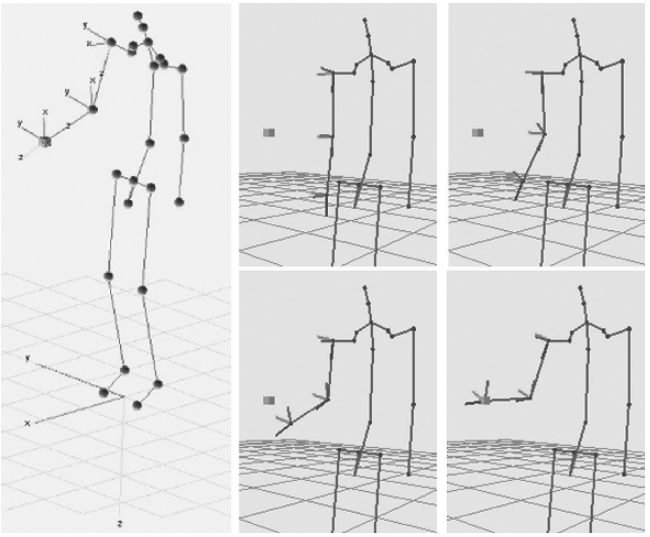
In order to work on articulated figures, Denavit and Hartenberg developed a matrix notation, called *DH notation*, to represent the kinematics of articulated chains [42]. DH notation is a link-based notation where each link is represented by four parameters; Θ, d, a, and α. For a link, Θ is the joint angle, d is the distance from the origin, a is the offset distance, and α is the offset angle. The relations between the links are represented by 4×4 matrices. Sims and Zeltzer [43] proposed a more intuitive method, called *axis-position (AP) representation*. In this approach, the position and orientation of the joint and the pointers to the joint's child nodes are used to represent the articulated figure.

Inverse kinematics is a higher-level approach. It is sometimes called "goal-directed motion." Given the positions of end-effectors only, inverse kinematics solves the position and orientation of all the joints in the hierarchy. Mathematically, it is expressed as

$$\Theta = f^{-1}(x). \tag{7.3}$$

Figure 7.3(a) shows the joint orientations for the end-effector positioning of the right arm and Fig. 7.3(b) shows goal-directed motion of the arm.

Inverse kinematics is mostly used in robotics. Contrary to forward kinematics, inverse kinematics provides direct control over the movement of the end-effector. On the other hand, inverse kinematics problems are difficult to solve as compared to forward kinematics where the solution is found easily by multiplying the local transformation matrices of joints in a hierarchical manner. Inverse kinematics problems are non-linear and for a given position x there may be more than one solution for Θ. There are two approaches to solve inverse kinematics problem: *numerical* and *analytical*.



(a) (b)

Fig. 7.3. (**a**) The joint orientations for the end-effector positioning of the right arm; (**b**) goal directed motion of the arm

The most common solution method for the non-linear problem stated in (7.3) is to linearize it [44]. When the problem is linearized, the joint velocities and the end-effector velocity will be related by:

$$\dot{x} = J(\Theta)\dot{\Theta}, \tag{7.4}$$

where

$$J_{ij} = \frac{\delta f_i}{\delta x_j}. \tag{7.5}$$

The Jacobian J relates the changes in the joint variables to the changes in the position of the end-effector. Jacobian J is an $m \times n$ matrix where m is the number of joint variables and n is the dimensions of the end-effector vector. If we invert (7.4), we obtain the equation:

$$\dot{\Theta} = J^{-1}(\Theta)\dot{x}. \tag{7.6}$$

Given the inverse of the Jacobian, computing the changes in the joint variables due to changes in the end-effector position can be achieved by using an *iterative algorithm*. Each iteration computes the $\dot{x}$ value by using the actual and goal positions of the end-effector. The iterations continue until the end-effector reaches the goal. To compute the joint velocities ($\dot{\Theta}$), we must find the $J(\Theta)$ value for each iteration. Each column of the Jacobian matrix

corresponds to a single joint. The changes in the end-effector position ($\mathrm{P}(\Theta)$) and orientation ($\mathrm{O}(\Theta)$) determine the Jacobian column entry for the i^{th} joint according to

$$J_i = \begin{bmatrix} \delta P_x/\delta\Theta_i \\ \delta P_y/\delta\Theta_i \\ \delta P_z/\delta\Theta_i \\ \delta O_x/\delta\Theta_i \\ \delta O_y/\delta\Theta_i \\ \delta O_z/\delta\Theta_i \end{bmatrix}. \tag{7.7}$$

These entries can be calculated as follows: every joint i in the system translates along or rotates around a local axis u_i. If we denote the transformation matrix between local frames and the world frame as M_i, the normalized transformation of the local joint axis will be:

$$axis_i = u_i M_i. \tag{7.8}$$

We can calculate the Jacobian entry for a translating joint using (7.8):

$$J_i = \begin{bmatrix} [axis_i]^T \\ 0 \\ 0 \\ 0 \end{bmatrix}, \tag{7.9}$$

and for a rotating joint by:

$$J_i = \begin{bmatrix} [(p - j_i) \times axis_i]^T \\ (axis_i)^T \end{bmatrix}. \tag{7.10}$$

The linearization approach makes an assumption that the Jacobian matrix is invertible (both square and non-singular), but this is not generally so. In the case of redundancy and singularities of the manipulator, the problem is more difficult to solve and new approaches are needed.

Unlike the numerical methods, analytical methods in most cases find solutions. We can classify the analytical methods into two groups; *closed-form* and *algebraic elimination methods* [45]. Closed-form methods specify the joint variables by a set of closed-form equations; they are usually applied to six degrees-of-freedom (DOF) systems with a specific kinematic structure. On the other hand, in the algebraic elimination methods, the joint variables are denoted by a system of multivariable polynomial equations. Generally, the degrees of these polynomials are greater than four. That is why algebraic elimination methods still require numerical solutions. In general, analytical methods are more common than numerical ones because analytical methods find all solutions and are faster and more reliable.

The main advantages of kinematics-based approaches are as follows: first, the motion quality is based on the model and the animator's capability; second, the cost of computations is low. However, the animator must still spend a lot of time producing the animations. These approaches cannot produce physically-realistic animations since the dynamics of the movements are not considered and the interpolation process leads to loss of realism.

When we review the literature we observe that Chadwick et al. use inverse kinematics in creating keyframes [24]. Badler et al. also propose an inverse kinematics algorithm to constrain the positions of the body parts during animation [46]. In addition, Girard and Maciejewski [47] and Sims and Zeltzer [43] generate leg motion by means of inverse kinematics. Their systems are composed of two stages: in the first stage, foot trajectories are specified; in the second stage, the inverse kinematic algorithm computes the leg-joint angles during movement of the feet. Welman investigates the inverse kinematics in detail and describes the constrained inverse kinematic figure manipulation techniques [48]. Baerlocher investigates inverse kinematics techniques for the interactive posture control of articulated figures [49]. Greeff et al. propose constraints to fix and pin the position and orientation of joints for inverse kinematics [50].

7.3.2 Dynamics

Since physical laws heavily affect the realism of a motion, dynamics approaches can be used for animation. However, these approaches require more physical parameters such as center of mass, total mass and inertia. The dynamics techniques can be classified as *forward dynamics* and *inverse dynamics*.

Forward dynamics considers applying forces on the objects. These forces can be applied automatically or by the animator. The motion of the object is then computed by solving the equations of the motion for the object as a result of these forces. Wilhelm gives a survey of rigid-body animation techniques by using forward dynamics [51]. Although the method works well with a rigid body, the simulation of articulated figures by forward dynamics is more complicated because the equations of motion for articulated bodies must also handle the interaction between the body parts. This extension of the equation makes control difficult. In addition, forward dynamics does not provide an accurate control mechanism. This makes the method useful for the tasks in which initial values are known. Nevertheless, there are some examples of articulated-figure animation using forward dynamics [52, 53].

Inverse dynamics however is a goal-oriented method in which the forces needed for a motion are computed automatically. Although the inverse dynamics applications are rarely used in computer animation, Barzel and Barr [54] are the first users of the method. They generate a model composed of objects that are geometrically related. These relationships are represented by constraints that are used to denote forces on an object. These forces animate the figure in such a way that all the constraints are satisfied. The Manikin

system proposed by Forsey and Wilhelm [55] also animates articulated models by means of inverse dynamics. The system computes the forces needed when a new goal position for the figure is specified. Another system, called Dynamo, is introduced by Isaac and Cohen [56]; it is based on keyframed kinematics and inverse dynamics. Their approach is an example of combining dynamic simulation and kinematic control. Ko also developed a real-time algorithm for animating human locomotion using inverse dynamics, balance and comfort control [57].

All the studies outlined above describe the motion of a figure by considering geometric constraints. On the other hand, some researchers develop some inverse dynamics solutions based on non-geometric constraints. Brotman and Netravali [58], Girard [59] and Lee et al. [60] are examples.

The combination of anatomical knowledge with the inverse dynamics approach generates more realistic motion. This composite system can also handle the interaction of the model with the environment. That is why this method is useful for virtual reality applications. However, dynamic techniques are computationally more costly than kinematic techniques and are rarely used as interactive animation tools.

There are difficulties in using a purely forward dynamics system or an inverse dynamics system. To produce the desired movement with high level of control requires hybrid solutions. The need to combine forward and inverse dynamics is discussed in [61].

7.3.3 Motion Capture

Since dynamics simulation could not solve all animation problems, new approaches were introduced. One of these methods animates virtual models by using human motion data generated by motion capture techniques. The 3D positions and orientations of the points located on the human body are captured and this data is then used to animate 3D human models. It is mainly used in the film and computer games industries.

Human motion capture systems can be classified as *non-vision based*, *vision-based with markers*, and *vision-based without markers* [62]. In non-vision based systems, the movement of a real actor can be captured by using magnetic or optical markers attached to the human body. In vision-based systems, the motion of the human body is tracked with the help of cameras either with markers attached to the human body [63] or without markers [64, 65]. The main advantage of motion capture techniques is their realistic generation of motion quickly and with a high level of detail. Moreover, with additional computations, the 3D motion data can be adapted to new morphology. *Motion blending* and *motion warping* are the two techniques for obtaining different kind of motions. There are also studies to generate smooth human motions interactively by combining a set of clips obtained from motion capture data [66].

Motion blending needs motions with different characteristics. It interpolates between the parameters of motions and has the advantage of low

computation cost. Motion warping techniques change the motion by modifying the motion trajectories of different limbs interactively. But this method suffers from the same problem as kinematic or procedural animation techniques; since they cannot handle dynamic effects, it is impossible to ensure that resulting motions are realistic.

In their studies, Bruderlin and Williams [67] work on signal processing techniques to alter existing motions. Unuma et al. [68] generate human figure animations by using Fourier expansions on the available motions. In contrast with the procedural and kinematic techniques, motion modification techniques provide for using real-life motion data to animate the figures. This has the advantage of producing natural and realistic looking motion at an enhanced speed. On the other hand, this method is not a convenient method to modify the movement captured in the data. Realism can be lost while applying large changes to the captured data.

With the advance of motion capture techniques, vast amount of of motion capture data is produced. Since the storage and processing of this data becomes very difficult, keyframe extraction from motion capture data [69] and compression techniques [70] become more and more important.

7.3.4 Interpolation Techniques for Motion Control

The parameters of body parts (limbs and joints) should be determined at each frame when a character is animated using kinematic methods. However, determining the parameters explicitly at each frame, even for a simple motion, is not trivial. The solution is to specify a series of keyframe poses at different frames. Following this approach, an animator needs only specify the parameters at the keyframes. Parameter values for the intermediate frames, called in-betweens, are obtained by interpolating the parameters between these keyframes.

Another problem arises in searching for a suitable interpolation method. Linear interpolation is the simplest method to generate intermediate poses, but it gives unsatisfactory motion. Due to the discontinuities in the first derivatives of interpolated joint angles, this method generates a robotic motion. Obtaining more continuous velocity and acceleration requires higher order interpolation methods like piecewise splines.

Intermediate values produced by interpolation generally do not satisfy the animator. Therefore, the interpolation process should be kept under control. For just a single DOF, the intermediate values constitute a trajectory curve that passes through the keyframe values. Interpolating a spline along with the keyframe values at both ends determines the shape of the trajectory. An interactive tool that shows the shape of a trajectory and enables an animator to change the shape could be useful. After a trajectory is defined, traversing it at a varying rate can improve the quality of the movement. Parameterized interpolation methods control the shape of a trajectory and the rate at which the trajectory is traversed.

Kochanek and Bartels describe an interpolation technique that relies on a generalized form of piecewise cubic Hermite splines [71]. At the keyframes, the magnitude and direction of tangent vectors (tangent to the trajectory) are controlled by adjusting *continuity*, *tension* and *bias*. Changing the direction of the tangent vectors locally controls the shape of the curve when it passes through a keyframe. On the other hand, the rate of change of the interpolated value around the keyframe is controlled by changing the magnitude of the tangent vector. Some animation effects such as action follow-through and exaggeration [23] can be obtained by setting the parameters. This method does not have the ability to adjust the speed along the trajectory without changing the trajectory itself because the three parameters used in the spline formulation influence the shape of the curve.

Steketee and Badler [72] offered a double interpolant method in which timing control is separated from the trajectory itself. Similar to the previous method, a trajectory is a piecewise cubic spline that passes through the keyframed values. In addition, the trajectory curve is sampled by a second spline curve. This controls the parametric speed at which the trajectory curve is traversed. Unfortunately, there is no one-to-one relation between actual speed in the geometric sense and the parametric speed. Therefore, the desired velocity characteristic is obtained by a trial-and-error process.

A way to obtain more intuitive control over the speed is to reparameterize the trajectory curve by arc length. This approach provides a direct relation between parametric speed and geometric speed. An animator can be provided with an intuitive mechanism to vary speed along the trajectory by allowing him to sketch a curve that represents distance over time [73].

In the traditional animation, keyframes are drawn by experienced animators and intermediate frames are completed by less experienced animators. In this manner, the keyframe-based approach and traditional animation are analogous. The problem with keyframe-based animation is that it is not good at skeleton animation. The number of DOF is one of the main problems. When the number of DOF is high, an animator has to specify too many parameters for even a single key pose. Obviously controlling the motion by changing lots of trajectory curves is a difficult process. The intervention of the animator should ideally happen at low levels, perhaps at the level of joint control.

Another problem arises from the hierarchical structure of the skeleton. Since the positions of all other components depend on the position of the root joint, the animator cannot easily determine the positional constraints in each keyframe pose creation. The problem can be solved by specifying a new root joint and reorganizing the hierarchy but this is rarely useful. The interpolation process also suffers from the hierarchical structure of the skeleton. It is impossible to calculate the correct foot positions in the intermediate frames by only interpolating joint rotations.

7.4 Simulating Walking Behavior

Most of the work in motion control has been aimed at producing complex motions like walking. Kinematics and dynamic approaches for human locomotion have been described by many researchers [59, 74, 75, 76, 77, 78, 79]. A survey of human walking animation techniques is given in [80].

7.4.1 Low-level Motion Control

Spline-driven techniques are very popular as the low-level control mechanism to specify the characteristics of the movements. In spline-driven motion control, the trajectories of limbs, e.g., the paths of pelvis and ankles, are specified using spline curves. Using a conventional keyframing technique, the joint angles over time are determined by splines. Since splines are smooth curves, they can be used for the interpolation of motion parameters.

Cubic splines are easy to implement, computationally efficient, and their memory requirement is low. Local control over the shape of the curve, interpolation of the control points, and continuity control are desirable properties of cubic curves. Another advantage is that they are more flexible than lower-order polynomials in modeling arbitrary curved shapes [81].

A class of cubic splines, Cardinal splines, can be used to generate the position curves because Cardinal splines require less calculation and memory, yet can exert local control over shape. In addition, there is a velocity curve or distance curve for these body parts enabling us to change the nature of the movement. For a set of control points, a piecewise-continuous curve that interpolates these control points can be generated. If the set of control points is given by:

$$p_k = (x_k, y_k, z_k), \quad k = 0, 1, 2, \ldots, n. \tag{7.11}$$

The piecewise-continuous curve generated from these control points is given in Fig. 7.4(a). The parametric cubic polynomials to generate the curves between each pair of control points is represented by the following equations:

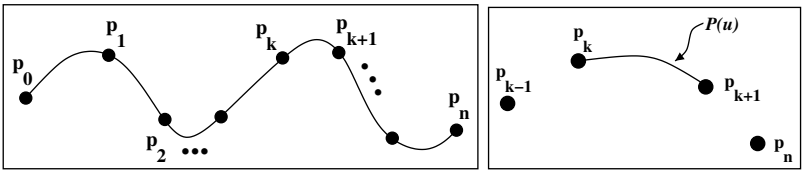


Fig. 7.4. Cubic splines: (**a**) a piecewise-continuous cubic-spline interpolation of $n+1$ control points; (**b**) parametric point function $P(u)$ for a Cardinal spline section

$$\begin{bmatrix} x_u \\ y_u \\ z_u \end{bmatrix} = \begin{bmatrix} a_x u^3 + b_x u^2 + c_x u + d_x \\ a_y u^3 + b_y u^2 + c_y u + d_y \\ a_z u^3 + b_z u^2 + c_z u + d_z \end{bmatrix}, \quad (0 \le u \le 1). \tag{7.12}$$

Cardinal splines interpolate piecewise cubics with specified endpoint tangents at the boundary of each curve section but they do not require the values for the endpoint tangents. Instead, the value of the slope of a control point is calculated from the coordinates of the two adjacent points. A Cardinal spline section is specified by four consecutive control points. The middle two points are the endpoints and the other two are used to calculate the slope of the endpoints. If $P(u)$ represents the parametric cubic curve function for the section between control points P_k and P_{k+1} (Fig. 7.4(b)), the boundary conditions for the Cardinal spline section are formulated by the following equations:

$$\begin{aligned} P(0) &= p_k, \\ P(1) &= p_{k+1}, \\ P'(0) &= \tfrac{1}{2}(1-t)(p_{k+1} - p_{k-1}), \\ P'(1) &= \tfrac{1}{2}(1-t)(p_{k+2} - p_k), \end{aligned} \tag{7.13}$$

where the tension parameter, t, controls how loosely or tightly the Cardinal spline fits the control points. We can generate the boundary conditions as:

$$P_u = \begin{bmatrix} u^3 & u^2 & u & 1 \end{bmatrix} \cdot M_C \cdot \begin{bmatrix} p_{k-1} \\ p_k \\ p_{k+1} \\ p_{k+2} \end{bmatrix}, \tag{7.14}$$

where the Cardinal basis matrix is:

$$M_C = \begin{bmatrix} -s & 2-s & s-2 & s \\ 2s & s-3 & 3-2s & -s \\ -s & 0 & s & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}, \quad s = \frac{(1-t)}{2}. \tag{7.15}$$

The paths for pelvis, ankle and wrist motions are specified using Cardinal spline curves. These are position curves. Another curve for velocity is specified independently for each body part. Thus, by making changes in the velocity curve, the characteristics of the motion can be changed. Steketee and Badler are the first to recognize this powerful method [72]. They call this velocity curve the "kinetic spline". The method is called "double interpolant" method. Kinetic spline may also be interpreted as a distance curve. Distance or velocity curves can be easily calculated from each other.

This application makes use of the "double interpolant" by enabling the user to specify a position spline and a kinetic spline. The kinetic spline, $V(t)$, is commonly used as the motion curve. However, $V(t)$ can be integrated to determine the distance curve, $S(t)$. These curves are represented in two-dimensional space for easy manipulation. The position curve is a three-dimensional curve

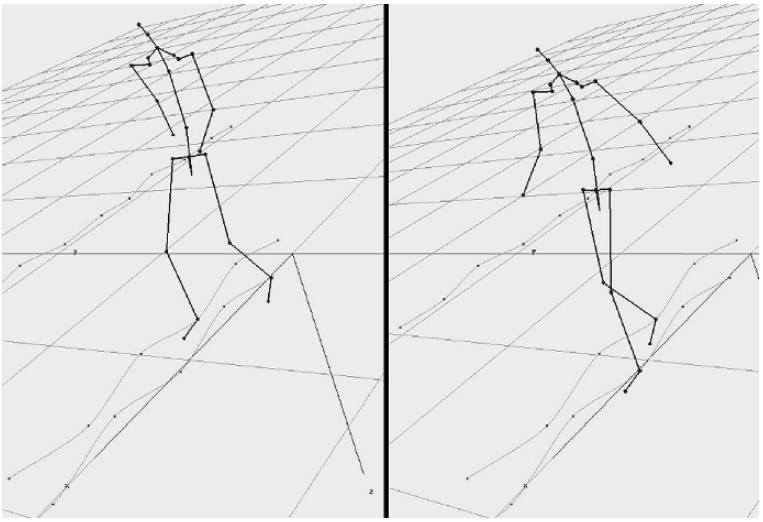


Fig. 7.5. The articulated figure and the Cardinal spline curves

in space, through which the object moves. Control of the motion involves editing the position and kinetic splines. In this application, velocity curves are straight lines and position curves are Cardinal splines. Figure 7.5 shows the position curves for a human figure.

However moving an object along a given position spline presents a problem because of the parametric nature of the cubic splines. Suppose we have a velocity curve and a position curve to control the motion. We can find the distance traveled at a given time by taking the integral of the velocity curve with respect to time. Now we must find a point along the position spline, where the computed distance is mapped.

Assume that we have a path specified by a spline $Q(u)$, $(0 \leq u \leq 1)$ and we are looking for a set of points along the spline such that the distance traveled along the curve between consecutive frames is constant. Basically, these points can be computed by evaluating $Q(u)$ at equal values of u. But this requires the parameter u to be proportional to the arclength, the distance traveled along the curve. Unfortunately, this is not usually the case. In the special cases where the parameter is proportional to arclength, the spline is said to be parameterized by arclength. An object can hardly move with a uniform speed along a spline without arclength parameterization [82].

The animator interactively shapes the curves and views the resulting animation in real time. The double interpolant method enables the animator to change the characteristics of the motion independently. This can be done by independently editing different curves that correspond to the position in 3D space, distance, and velocity. However, to produce the desired movement, the change in the kinetic spline curve should be reflected to the position curve.

After constructing the position curves for end-effectors, such as wrists and ankles, a goal-directed motion control technique is used to determine, over time, the joint angles of shoulder, hip, elbow and knee. The animator only moves the end-effector and the orientations of other links in the hierarchy are computed by inverse kinematics. This system also enables the user to define the joint angles that are not computed by inverse kinematics. Using a conventional keyframing technique, the joint angles over time can be specified by the user. Cubic splines are then used to interpolate joint angles.

7.4.2 High-level Motion Control

Walking motion can be controlled by using high-level kinematic approaches that allow the user to specify the *locomotion parameters*. Specifying a straight traveling path on a flat ground without any obstacles and the speed of locomotion, walking can be generated automatically by computing the 3D path information and the low-level kinematics parameters. The user can adjusts such parameters as the size of step, the time elapsed during double-support, rotation, tilt, and lateral displacement of the pelvis to produce different walking motions.

Walking is a smooth, symmetric motion in which the body, feet, and hands move rhythmically in the required direction at a given speed. Basically, it can be characterized as a succession of phases separated by different states of the feet because the feet drive the main part of the animation. During walking, the foot has contact with the ground (*footstrikes*) and lifts off the ground (*takeoffs*). A *stride* is defined as the walking cycle in which four footstrikes and takeoffs occur. The part of this cycle, which is between the takeoffs of the two feet, is called a *step*. The phases for each leg can be classified into two: the *stance* and the *swing* phase. The stance phase is the period of support. The swing phase is the non-support phase. In the locomotion cycle, each leg passes through both the stance and the swing phases. In the cycle, there is also a period of time when both of the legs are in contact with the ground; this phase is called *double-support*. The phases of the walking cycle are shown in Fig. 7.6.

During the cyclic stepping motion, one foot is in contact with the ground at all times and for a period both of the feet are in contact with the ground. These characteristics of walking really simplify the control mechanism. The kinematics nature of walking must be dissected further to produce a realistic walking motion. For this purpose, Saunders et al. defined a set of *gait determinants*, which mainly describe pelvis motion [83]. These determinants are *compass gait*, *pelvic rotation*, *pelvic tilt*, *stance leg flexion*, *planar flexion of the stance angle*, and *lateral pelvic displacement*.

In pelvic rotation, the pelvis rotates to the left and right, relative to the walking direction. Saunders et al. quote 3 degrees as the amplitude of pelvic rotation in a normal walking gait. In normal walking, the hip of the swing leg falls slightly below the hip of the stance leg. For the side of swing

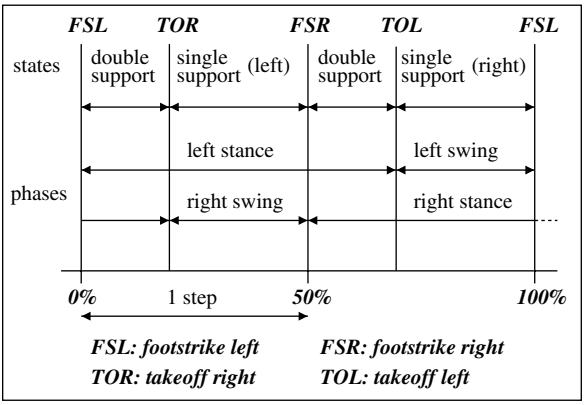


Fig. 7.6. The phases of the walking cycle [76]. © 1989 Association for Computing Machinery, Inc. Reprinted by permission

leg, this happens after the end of the double support phase. The amplitude of pelvic tilt is considered to be 5 degrees. In lateral pelvic displacement, the pelvis moves from side to side. Immediately after double support, the weight is transferred from the center to the stance leg; thus the pelvis moves alternately during normal walking. Moreover, individual gait variations can be achieved by modifying these pelvic parameters.

The main parameters of walking behavior are velocity and step length. However, experimental results show that these parameters are related. Saunders et al. [83] relate the walking speed to walking cycle time and Bruderlin and Calvert [76] state the correct time durations for a locomotion cycle. Based on the results of experiments, the typical walking parameters can be stated as follows:

$$velocity = step\_length \times step\_frequency, \tag{7.16}$$

$$step\_length = 0.004 \times step\_frequency \times body\_height. \tag{7.17}$$

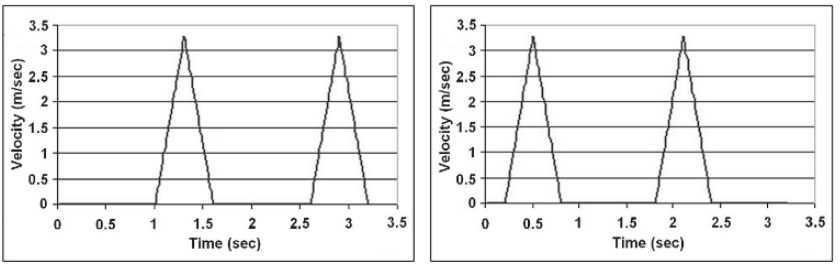
Experimental data shows that maximum value of $step\_frequency$ is 182 steps per minute. The time for a cycle (t_{cycle}) can be calculated from the step frequency:

$$t_{cycle} = 2 \times t_{step} = \frac{2}{step\_frequency}. \tag{7.18}$$

Time for double support (t_{ds}) and t_{step} are related according to the following equations:

$$t_{step} = t_{stance} - t_{ds}, \tag{7.19}$$

$$t_{step} = t_{swing} + t_{ds}. \tag{7.20}$$



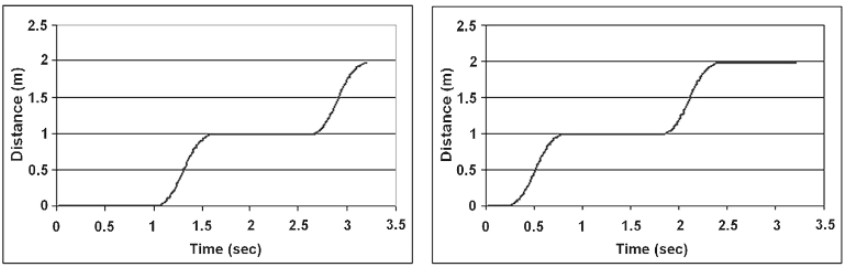
(a) (b)

Fig. 7.7. Velocity curve of (**a**) left ankle and (**b**) right ankle

Based on the experimental results, t_{ds} is calculated as:

$$t_{ds} = (-0.0016 \times step\_frequency + 0.2908) \times t_{cycle}. \tag{7.21}$$

Although t_{ds} can be calculated automatically if $step\_frequency$ is given, it is sometimes convenient to redefine t_{ds} to have walking animations with various characteristics. By utilizing these equations that define the kinematics of walking, a velocity curve is constructed for the left and right ankles. Figure 7.7 illustrates the velocity curves of the left and right ankles. The distance curves shown in Fig. 7.8 are automatically generated using the velocity curves. The ease-in, ease-out effect, which is generated by speed up and slow down of the ankles, can be seen on distance curves.



(a) (b)

Fig. 7.8. Distance curve of (**a**) left ankle and (**b**) right ankle

7.5 Motion Control for a Multi-layered Human Model

In a multi-layered human model, the skin is deformed based on the transformation of the inner layers, namely the skeleton and muscle layers.

7.5.1 Skeleton Layer

The *skeleton layer* is composed of joints and bones and controls the motion of the body by manipulating the angles of the joints. To solve the inverse kinematics problem, there are software packages, such as *Inverse Kinematics using Analytical Methods (IKAN)*. IKAN has the functionality to control the movement of body parts [45]. It is a complete set of inverse kinematics algorithms for an anthropomorphic arm or leg. It uses a combination of analytic and numerical methods to solve generalized inverse kinematics problems including position, orientation, and aiming constraints [45].

For the arm, IKAN computes the joint angles at the shoulder and elbow to put the wrist in the desired location. In the case of the leg, the rotation angles for the hip and knee are calculated. IKAN's methodology is constructed on a 7 DOF fully revolute open kinematic chain with two spherical joints connected by a single revolute joint. Although the primary work is on the arm, the methods are suitable for the leg since the kinematic chain of the leg is similar to the kinematic chain of the arm. In the arm model, the spherical joints with 3 DOFs are the shoulder and wrist; the revolute joint with 1 DOF is the elbow. In the leg model, the spherical joints with 3 DOFs are the hip and ankle; the revolute joint with 1 DOF is the knee.

Since leg and arm are similar to human arm-like (HAL) chains, only the details for the arm are explained here. The elbow is considered to be parallel to the length of the body at rest. The z-axis is from elbow to wrist. The y-axis is perpendicular to z-axis and is the axis of rotation for the elbow. The x-axis is pointing away from the body along the frontal plane of the body. A right-handed coordinate system is assumed.

Similar coordinate systems are assumed at the shoulder and at the wrist. The projection axis is always along the limb and the positive axis points away from the body perpendicular to the frontal plane of the body. The projection axis differs for left and right arm. Wrist to elbow and elbow to shoulder transformations are calculated since they are needed to initialize the inverse kinematics solver in IKAN. The arm is initialized with a Humanoid object and the shoulder, elbow and wrist joints are named. During initialization, the transformation matrices are computed and the inverse kinematics solver is initialized.

The orientations of joints for the positioning of the right arm are seen in Fig. 7.3(a). The joint angles are found automatically according to the end-effector positions. Figure 7.9 illustrates different motions of the skeleton (walking, jumping, squatting, running, and forearm motion).

7.5.2 Muscle Layer

While the skeleton creates the general structure of the body, muscles determine the general shape of the surface mesh. Human muscles account for nearly half of the total mass of the body and fill the gap between the skeleton and the skin [84].

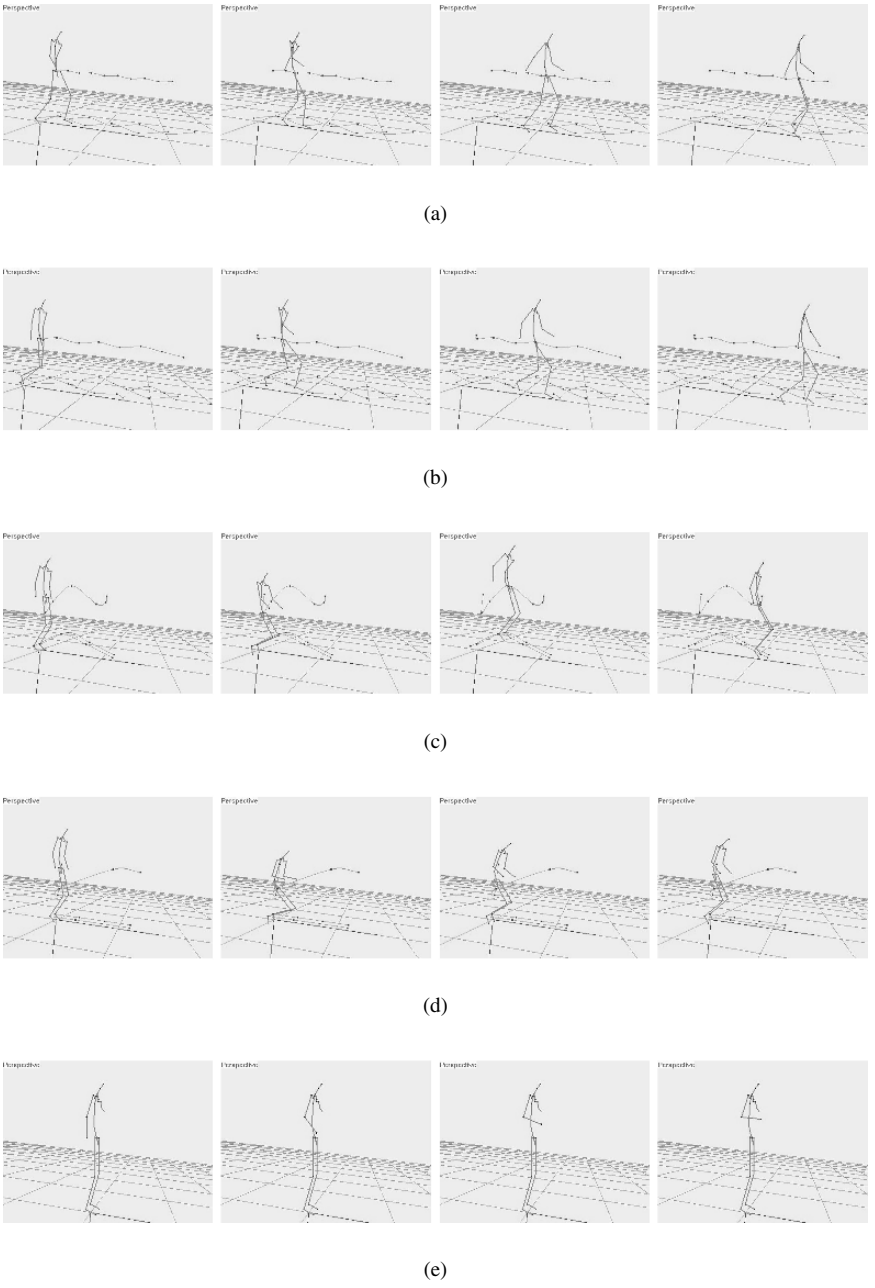


Fig. 7.9. Different motions of the skeleton: (**a**) walking; (**b**) running; (**c**) jumping; (**d**) squatting; (**e**) forearm motion

Human movements require the muscles to perform different tasks; thus the human body includes three types of muscle: *cardiac*, *smooth* and *skeletal* [85]. Cardiac muscles, found only in the heart, perform the pumping of the blood throughout the body. Smooth muscles are part of the internal organs and are found in the stomach, bladder, and blood vessels. Both of these muscle types are involuntary muscles because they cannot be consciously controlled. On the other hand, skeletal muscles control conscious movements. They are attached to bones by tendons and perform various actions by simply contracting and pulling the bones they are attached to towards each other. If only the external appearance of the human body is important, modeling the skeletal muscles serves the purpose.

Skeletal muscles are located on top of the bones and other muscles, and they are structured side by side and in layers. There are approximately 600 skeletal muscles and they make up 40% to 45% of the total body weight. Skeletal muscle is an elastic, contractile material that originates at fixed origin locations on one or more bones and inserts on fixed insertion locations on one or more other bones [84]. The relative positions of these origins and insertions determine the diameter and shape of the muscle. In real life, muscle contraction causes joint motion but in many articulated body models, muscles deform due to joint motion in order to produce realistic skin deformations during animation.

There are two types of contraction: *isometric* (same length) and *isotonic* (same tonicity). In isotonic contraction, when the muscle belly changes shape, the total length of the muscle shortens. As a result, the bones to which the muscle is attached are pulled towards each other. Figure 7.10 illustrates the isotonic contraction. In isometric contraction, the changes in the shape of the muscle belly due to the tension in the muscle do not change the length of the muscle, so no skeletal motion is produced [38]. Although most body movements require both isometric and isotonic contraction, many applications consider only isotonic contraction; isometric contractions have very little influence on the appearance of the body during animation.

In most muscle models, a muscle is represented using two levels: the action line and the muscle shape. Sometimes, the muscle layer is represented only with action lines and the muscle shape is not considered (Fig. 7.11). The key idea behind this is that the deformations of the skin mesh can be calculated based on the underlying action lines and no muscle shape

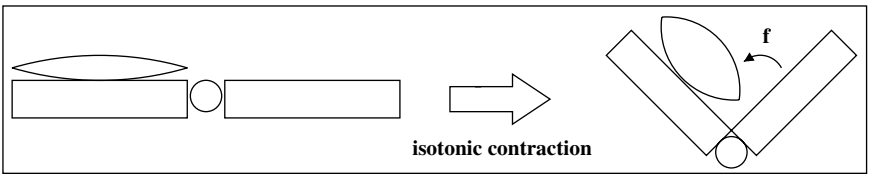


Fig. 7.10. Isotonic contraction

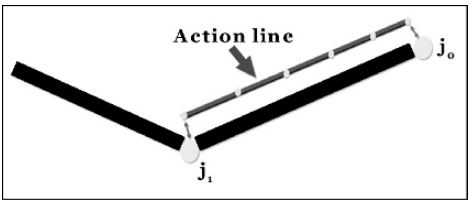


Fig. 7.11. Action line abstraction of a muscle

is needed for most applications. The calculation of the muscle shape and its effect on the skin layer is a complicated and computationally-intensive process.

7.5.2.1 Modeling of Muscles

An *action line* denotes the imaginary line along which the force applied onto the bone is produced. However, the definition of this line is not clear [38]. Many specialists assume the action line to be a straight line, but more commonly, the action line is represented as a series of line segments (or polyline in computer graphics terminology) [86]. These segments and their number are determined through the anatomy of the muscle. A muscle represented by an action line simulates the muscle forces and is basically defined by an origin and an insertion point. The control points on the action line guide the line and incorporate the forces exerted on the skin mesh. These force fields are inversely proportional to the length of the corresponding action line segment. An example of this kind of action line is shown in Fig. 7.12.

7.5.2.2 Animation of Muscles

Animation of muscles is a very complicated process due to the difficulty in determining the position and deformation of a muscle during motion. In computer-generated muscle models, the deformations of the muscles are generally inferred from the motion of the skeleton, the opposite of real life. The deformations of the skin layer are driven by the underlying bones and action lines. This allows the three-dimensional nature of the deformation problem

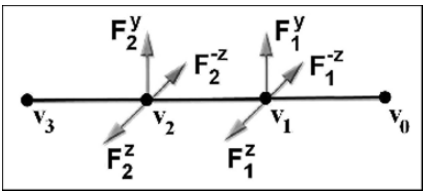


Fig. 7.12. The structure of an action line: control points and forces on these points

to be reduced to one dimension. The control points of the action lines that correspond to the insertion and origin of the muscle are attached to the skeleton joints so that their motion is dictated by the skeleton. The positions of all the remaining control points are obtained through a linear interpolation formulation for each animation frame. We first need to determine the local frame of each action-line control point since the positions of the control points of the action line provide information as to how the surface mesh will expand or shrink over time.

After the local frames are constructed, the action line is ready to animate in correspondence with the underlying skeleton. Since the insertion and origin points of the action line are fixed on the skeleton, movement of the skeleton layer is reflected on the action line as a decrease or increase in length. Parallel to the overall change in action-line length, the lengths of each action line segment also change. Since the force fields at each control point are inversely proportional to the segment length, this variation in length also causes a change in the force fields as demonstrated in Fig. 7.13. The next step in animation is the deformation in the skin mesh due to the changes on the forces that are exerted on skin vertices by the action line control points. This deformation is automatically propagated on the skin layer via the anchors between skin vertices and the action line. If the segment length shortens, the

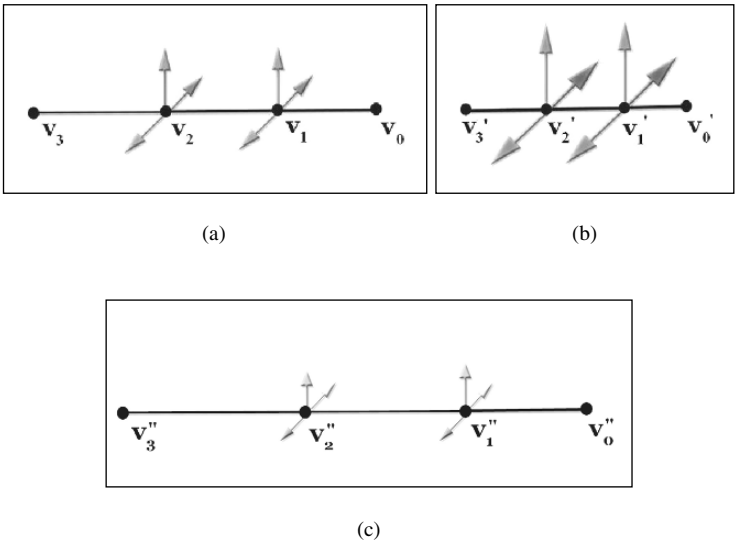


Fig. 7.13. Deformation of an action line and force field changes: (**a**) rest position and initial force fields of an action line, (**b**) the action line shortens and forces increase due to muscle contraction, and (**c**) the action line lengthens and forces decrease due to muscle lengthening

force fields increase and cause the skin mesh to bump. Similarly, the increase in segment length results in a decrease in force fields and relaxation of the skin mesh.

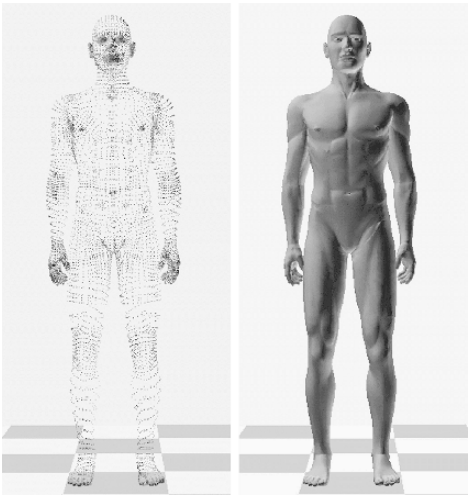
7.5.3 Skin Layer

The *skin* is a continuous external sheet that covers the body. The skin accounts for about 16% of the body weight and has a surface area from 1.5 to 2.0 m^2 in adults [87]. Its thickness varies depending on the location.

7.5.3.1 Modeling the Skin Layer

There are basically three ways to model a skin layer. The first method involves designing from scratch or modifying an existing mesh in a 3D modeler, such as Poser [88]. Another way is to laser scan a real person, thus producing a dense mesh that truly represents a human figure. The last method is to extract the skin layer from underlying components that already exist.

Figure 7.14 shows the shaded points and the solid view of the P3 Nude Man model, one of the standard characters of the Poser software [88]. The model contains 17,953 vertices and 33,234 faces. The whole body is composed of 53 parts such as hip, abdomen, head, right leg, left hand, etc. This structure facilitates the binding of skin vertices to the inner layers.



(a) (b)

Fig. 7.14. The shaded point (**a**) and solid view (**b**) of the skin model

7.5.3.2 Animation of the Skin Layer

In *attaching*, each vertex in the skin is associated with the closest underlying body components (muscle and bone). Basically, the attachment of a particular skin vertex is to the nearest point on its underlying component. Thus shape changes in the underlying component are propagated through these anchors to the corresponding skin vertices [89].

Skin vertices are first bound to the joints of the skeleton layer in a multi-step algorithm. To attach the skin to the joints, skin vertices are transformed into the joint coordinate system. The skin model is decomposed into *parts*, which are basically groups of vertices of the skin mesh corresponding to some part of the body. In the first step of the attachment process, a particular joint is determined for each part of the skin and attach the vertices of this part to this joint, i.e., the right upper arm is bound to the right shoulder joint and the left thigh is anchored to the left hip joint (Fig. 7.15(a)).

However, the first step is not sufficient for realistic deformation of the skin layer. Realism requires that some vertices be bound to more than one joint. In particular, the vertices near the joints need to be attached to two adjacent joints. The second step of the attachment process is focused on binding the necessary vertices to two joints. For this purpose, a distance threshold should be determined for each highly movable joint. If the distance between vertex and joint is smaller than the threshold value, then the vertex is bound to this second joint with a weight (Fig. 7.15(b)). The weights are inversely proportional to the distance between the vertex and the joint.

It is difficult to determine the most appropriate distance threshold for the binding operation. A small threshold value misses some of the vertices that should be attached to the joints. For example, some vertices in the back part of the hip need to be attached to the left hip or right hip joints in order to generate a realistic walking motion. However, since the distances between these vertices and the corresponding joints are larger than the threshold value, there may be holes in the skin mesh because of the unattached vertices. Increasing

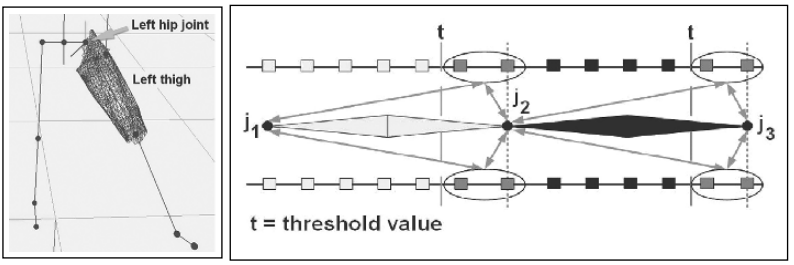


Fig. 7.15. The attachment process: (**a**) the left thigh is bound to the left hip joint; (**b**) binding skin vertices to more than one joint based on a distance threshold. The vertices inside the ellipsoids are bound to more than one joint

the threshold value is a solution to this problem but this attempt may cause unnatural results in other parts of the body during movement.

Therefore, unlike the first two steps, which are fully automatic, a manual correction step is generally applied to overcome these deficiencies. This can be done by selecting some of the unattached vertices one by one using a mouse and binding them to the appropriate joints. In 3D Studio Max, it is performed by manually adjusting the 3D influence envelopes [90]. A 3D influence envelope includes the skin vertices to be effected by the corresponding bone. Since the envelopes may overlap, some vertices are effected by more than one bone. Sometimes the automatically-defined envelopes may be wider or smaller than necessary; thus, a manual size adjustment may be required.

Anchoring points to the action lines of muscles is achieved by a similar process. An action line can be thought of as a skeleton; control points denote joints and line segments correspond to bones. This allows us to reuse (with some extensions) algorithms originally developed for mapping skin vertices to skeleton joints. In the skin-skeleton mapping algorithm, each skin vertex is bound to a particular joint. In the skin-action line mapping process, each skin vertex is again attached to a particular action line. An action line is composed of a set of control points and each control point has a different force field on the skin mesh. Thus, each skin vertex is bound to a number of control points on the action line (see Fig. 7.16).

Before applying muscle-induced deformations to the skin layer, each skin vertex is moved based on the skeletal position. This step has two goals: to generate a smooth skin appearance and to simulate the adherence of the skin layer to the skeleton layer. Skin vertices are attached to the joints of the skeleton layer rather than the limbs. But since the limbs are connected to the joints, the skin vertices move with the limbs in the underlying skeleton [89].

Figure 7.17 shows different motions with the rendered skin. The position curves seen in Figs. 7.9 and 7.17 are the trajectories of left ankle, right ankle and pelvis. The position and velocity curves of each limb are generated automatically.

Figure 7.18 demonstrates muscular and skeletal deformations. In the series of still frames, the right arm is raised forward and then flexed to demonstrate muscular deformation on the skin mesh.

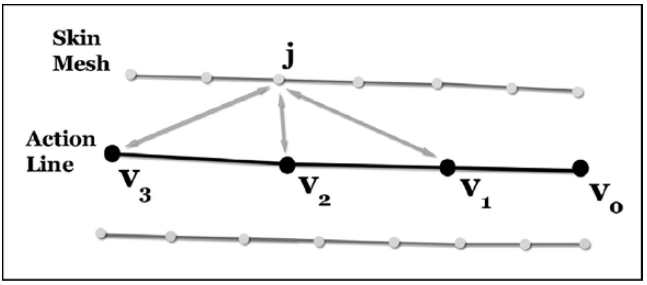


Fig. 7.16. Binding skin vertices with action lines. Point j is attached to its nearest three control points, v_1, v_2 and v_3

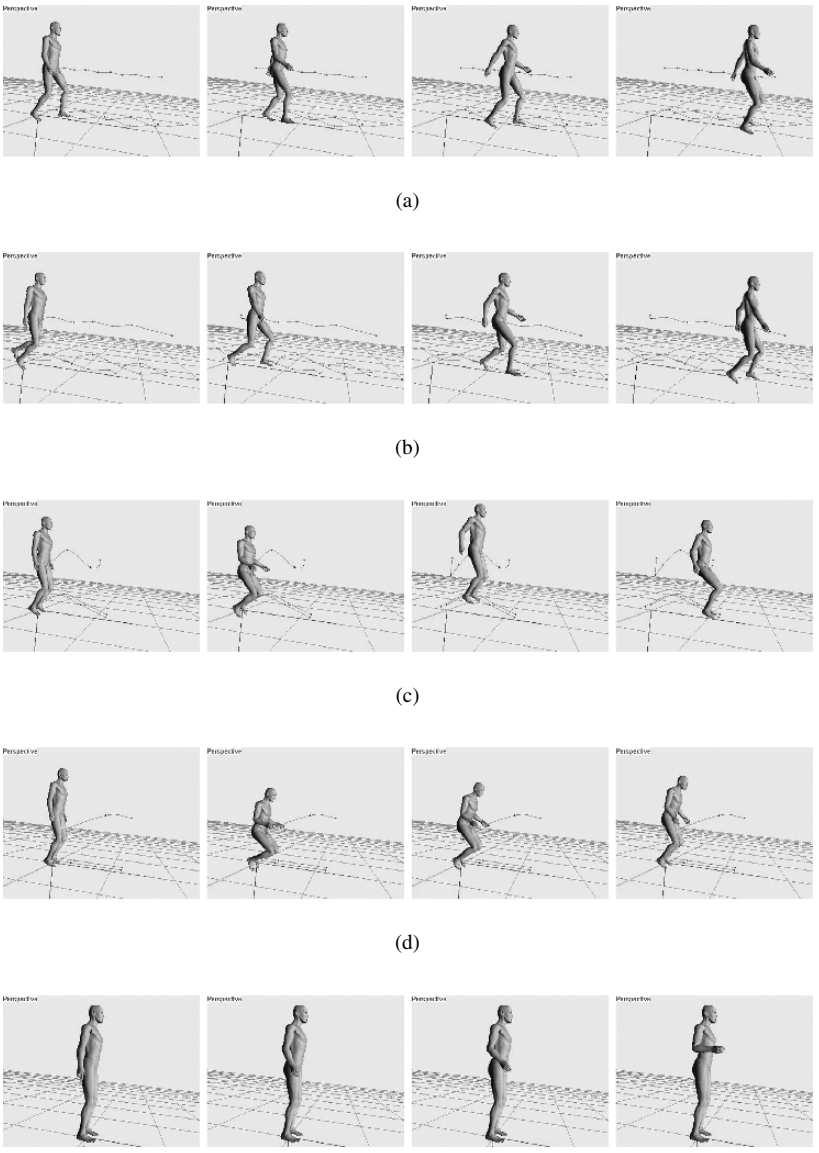


Fig. 7.17. Different motions of the human figure with the skin: (**a**) walking; (**b**) running; (**c**) jumping; (**d**) squatting; (**e**) forearm motion

The most tedious and time consuming operation is binding the skin layer to the underlying layers. However, this does not affect the frame rate if it is done as a preprocessing step before the animations. Depending on the application, different skin meshes ranging from low to high resolution can be used.

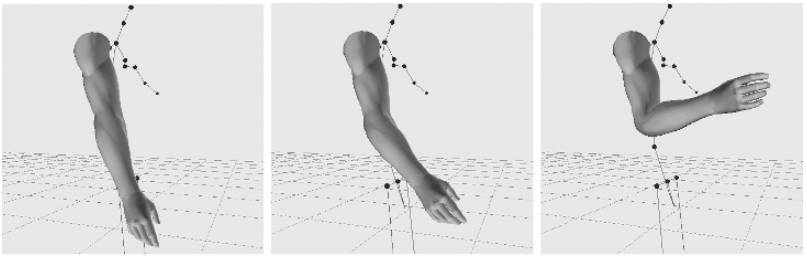


Fig. 7.18. Muscular deformation on the upper arm

7.6 Conclusions

Multi layered modeling of human motion based on anatomical approach yields realistic and real time results for generating avatars in motion for three dimensional computer animations. Films produced by these techniques have become very popular and wide spread, many with major box office success. The motion, up till recently, was specified by the animators. The current developments in motion capture now can also provide data for animation thus making human model animation a readily integral part of three dimensional television systems.

Building up motion data bases and using various motion definitions from such data bases, as well as modifying parts of this data whenever needed, provides a very powerful tool for the entertainment industry for generating three dimensional realistic humans in motion. Many other applications ranging from medical to flight simulators need near correct human models. Research outlined in this chapter will continue to this end and will increasingly become an important source for providing tools for three dimensional display systems.

Acknowledgements

We are grateful to Kirsten Ward for careful editing of the manuscript. This work is supported by the EC within FP6 under Grant 511568 with the acronym 3DTV.

References

1. N. Badler and S. Smoliar, "Digital representations of human movement," *ACM Computing Surveys*, Vol. 11, No. 1, pp. 19–38, 1979.
2. D. Herbison-Evans, "Nudes 2: A numeric utility displaying ellipsoid solids," *ACM Computer Graphics (Proc. of SIGGRAPH'78)*, pp. 354–356, 1978.
3. J. Korein, *A Geometric Investigation of Reach*. Cambridge, MA: The MIT Press, 1985.

4. N. Magnenat-Thalmann and D. Thalmann, *Computer Animation: Theory and Practice*. Berlin: Springer-Verlag, 1985.
5. N. Badler and S. Smoliar, "Graphical behavior and animated agents," in *ACM Siggraph Course Notes # 17, Advanced Techniques in Human Modeling, Animation and Rendering*, pp. 19–38, 1992.
6. N. Magnenat-Thalmann and D. Thalmann, *Synthetic Actors in 3D Computer-Generated Films*. New York: Springer-Verlag, 1990.
7. K. Komatsu, "Human skin model capable of natural shape variation," *The Visual Computer*, Vol. 3, No. 5, pp. 265–271, 1988.
8. N. Magnenat-Thalmann and D. Thalmann, "The direction of synthetic actors in the film rendez-vous à montreàl," *IEEE Computer Graphics and Applications*, Vol. 7, No. 12, pp. 9–19, 1987.
9. M. DeLoura, *Game Programming Gems*. Rockland, MA: Charles River Media Inc., 2000.
10. W. Sun, A. Hilton, R. Smith, and J. Illingworth, "Layered animation of captured data," *The Visual Computer*, Vol. 17, No. 8, pp. 457–474, 2001.
11. K. Singh and E. Kokkevis, "Skinning characters using surface-oriented free form deformations," in *Proceedings of Graphics Interface*, pp. 35–42, 2000.
12. K. Lander, "Skin them bones," *Game Developer*, pp. 11–16, 1998.
13. P. P. Sloan, C. F. Rose, and M. Cohen, "Shape by example," in *Proceedings of the Symposium on Interactive 3D Graphics*, pp. 135–143, 2001.
14. D. Talbot, "Accurate characterization of skin deformations using range data," Master's thesis, University of Toronto, 1998.
15. J. Blinn, "A generalization of algebraic surface drawing," *ACM Transactions on Graphics*, Vol. 1, pp. 235–256, 1982.
16. S. Yoshimoto, "Ballerinas generated by a personal computer," *The Journal of Visualization and Computer Animation*, Vol. 3, No. 1, pp. 85–90, 1992.
17. J. Bloomenthal and K. Shoemake, "Convolution surfaces," *ACM Computer Graphics (Proc. of SIGGRAPH'91)*, Vol. 25, No. 4, pp. 251–256, 1991.
18. A. Leclercq, S. Akkouche, and E. Galin, "Mixing triangle meshes and implicit surfaces in character animation," in *Proceedings of Animation and Simulation*, pp. 37–47, 2001.
19. D. Thalmann, J. Shen, and E. Chauvineau, "Fast human body deformations for animation and vr applications," in *Proceedings of Computer Graphics International*, pp. 166–174, 1996.
20. M. P. Gascuel, "An implicit formulation for precise contact modeling between flexible solids," *ACM Computer Graphics (Proc. of SIGGRAPH'93)*, pp. 313–320, 1993.
21. The National Library of Medicine, "The visible human project," available at `http://www.nlm.nih.gov/research/visible/visible_human.html`.
22. G. Hirota, S. Fisher, A. State, C. Lee, and H. Fuchs, "An implicit finite element method for elastic solids in contact," in *Proceedings of Computer Animation*, pp. 136–146, 2001.
23. J. Lasseter, "Principles of traditional animation applied to 3D computer animation," *ACM Computer Graphics (Proc. of SIGGRAPH'87)*, Vol. 21, No. 4, pp. 35–44, 1987.
24. J. E. Chadwick, D. R. Haumann, and R. E. Parent, "Layered construction for deformable animated characters," *ACM Computer Graphics (Proc. of SIGGRAPH'89)*, Vol. 23, No. 3, pp. 243–252, 1989.

25. J. Wilhelms and A. Van Gelder, "Anatomically based modeling," *ACM Computer Graphics (Proc. of SIGGRAPH'97)*, pp. 173–180, 1997.
26. F. Scheepers, R. Parent, W. Carlson, and S. May, "Anatomy based modeling of the human musculature," *ACM Computer Graphics (Proc. of SIGGRAPH'97)*, pp. 163–172, 1997.
27. L. Porcher-Nedel, "Anatomic modeling of human bodies using physically-based muscle simulation," Ph.D. dissertation, Swiss Federal Institute of Technology, 1998.
28. F. Scheepers, R. Parent, F. May, and W. Carlson, "Procedural approach to modeling and animating the skeletal support of the upper limb," Department of Computer and Information Science, The Ohio State University, Tech. Rep. OSU-ACCAD-1/96/TR1, 1996.
29. W. Maurel and D. Thalmann, "Human shoulder modeling including scapulothoracic constraint and joint sinus cones," *Computers & Graphics*, Vol. 24, No. 2, pp. 203–218, 2000.
30. T. Sederberg and S. Parry, "Free-from deformation of solid geometric models," *ACM Computer Graphics (Proc. of SIGGRAPH'86)*, Vol. 20, No. 4, pp. 151–160, 1986.
31. L. Moccozet, "Hand modeling and animation for virtual humans," Ph.D. dissertation, University of Geneva, 1996.
32. R. Turner and D. Thalmann, "The elastic surface layer model for animated character construction," in *Proceedings of Computer Graphics International*, pp. 399–412, 1993.
33. L. Nedel and D. Thalmann, "Real time muscle deformations using mass-spring systems," in *Proceedings of Computer Graphics International*, pp. 156–165, 1998.
34. V. Ng-Thow-Hing and E. Fiume, "Application-specific muscle representations," in *Proceedings of Graphics Interface*, pp. 107–115, 2002.
35. M. P. Gascuel, A. Verroust, and C. Puech, "A modeling system for complex deformable bodies suited to animation and collision processing," *The Journal of Visualization and Computer Animation*, Vol. 2, No. 3, pp. 82–91, 1991.
36. T. DeRose, M. Kass, and T. Truong, "Subdivision surfaces in character animation," *ACM Computer Graphics (Proc. of SIGGRAPH'98)*, Vol. 32, pp. 85–94, 1998.
37. M.-P. Cani-Gascuel and D. M., "Animation of deformable models using implicit surfaces," *IEEE Transactions on Visualization and Computer Graphics*, Vol. 3, No. 1, pp. 39–50, 1997.
38. A. Aubel, "Anatomically-based human body deformations," Ph.D. dissertation, Swiss Federal Institute of Technology, 2002.
39. Humanoid Animation Working Group of Web3D Consortium, "H-Anim 1.1: specification for a standard humanoid," available at `http://h-anim.org/`.
40. O. Arikan, D. Forsyth, and D. O'Brien, "Motion synthesis from annotations," *ACM Transactions on Graphics (Proc. of SIGGRAPH'02)*, Vol. 22, No. 3, pp. 402–408, 2003.
41. R. Boulic, P. Bécheiraz, L. Emering, and D. Thalmann, "Integration of motion control techniques for virtual human and avatar real-time animation," in *Proceedings of ACM Symposium on Virtual Reality Software and Technology (VRST'97)*, pp. 111–118, 1997.

42. J. Denavit and R. Hartenberg, "A kinematics notation for lower-pair mechanisms based on matrices," *Journal of Applied Mechanics (ASME)*, Vol. 22, No. 2, pp. 215–221, 1955.
43. K. Sims and D. Zeltzer, "A figure editor and gait controller for task level animation," in *ACM Siggraph Course Notes # 4, Synthetic Actors: The Impact of Artificial Intelligence and Robotics on Animation*, pp. 164–181, 1988.
44. J. Zhao and N. Badler, "Inverse kinematics positioning using nonlinear programming for highly articulated figures," *ACM Transactions on Graphics*, Vol. 13, No. 4, pp. 313–336, 1994.
45. T. Deepak, A. Goswami, and N. Badler, "Real-time inverse kinematics techniques for anthropomorphic limbs," *Graphical Models*, Vol. 62, No. 5, pp. 353–388, 2000.
46. N. Badler, K. Manoochehri, and G. Walters, "The dynamics of articulated rigid bodies for purpose of animation," *The Visual Computer*, Vol. 7, No. 6, pp. 28–38, 1987.
47. M. Girard and A. Maciejewski, "Computational modeling for computer generation of legged figures," *ACM Computer Graphics (Proc. of SIGGRAPH'85)*, Vol. 19, No. 3, pp. 263–270, 1985.
48. C. Welman, "Inverse kinematics and geometric constraints for articulated figure manipulation," Master's thesis, School of Computing Science, Simon Fraser University, 1993.
49. P. Baerlocher, "Inverse kinematics techniques of the interactive posture control of articulated figures," Ph.D. dissertation, Departement D'Informatique, Ecole Polytechnique Fédérale de Lausanne, 2001.
50. M. Greeff, J. Haber, and H.-P. Seidel, "Nailing and pinning: Adding constraints to inverse kinematics," in *Proceedings of the 13th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG'2005), Short Papers*, 2005.
51. J. Wilhelms, *Dynamic Experiences*. San Mateo, CA: Morgan-Kaufmann Publishers, pp. 265–280, 1991.
52. W. Armstrong and M. Green, "The dynamics of articulated rigid bodies for purpose of animation," *The Visual Computer*, Vol. 1, pp. 231–240, 1985.
53. J. Wilhelms, "Virya – a motion control editor for kinematic and dynamic animation," in *Proceedings of Graphics Interface*, pp. 141–146, 1986.
54. R. Barzel and A. H. Barr, "A modeling system based on dynamics," *ACM Computer Graphics (Proc. of SIGGRAPH'88)*, Vol. 22, No. 4, pp. 179–188, 1988.
55. D. Forsey and J. Wilhelms, "Techniques for interactive manipulation of articulated bodies using dynamic analysis," in *Proceedings of Graphics Interface*, pp. 8–15, 1988.
56. P. M. Isaacs and M. F. Cohen, "Controlling dynamic simulation with kinematic constraints, behavior functions and inverse dynamics," *ACM Computer Graphics (Proc. of SIGGRAPH'87)*, Vol. 21, No. 4, pp. 215–224, 1987.
57. H. Ko and N. Badler, "Animating human locomotion in real-time using inverse dynamics," *IEEE Computer Graphics and Applications*, Vol. 16, No. 2, pp. 50–59, 1996.
58. N. Brotman and A. Netravali, "Motion interpolation by optimal control," *ACM Computer Graphics (Proc. of SIGGRAPH'88)*, Vol. 22, No. 4, pp. 309–315, 1988.

59. M. Girard, *Constrained Optimization of Articular Animal Movement in Computer Animation*. San Mateo, CA: Morgan-Kaufmann Publishers, pp. 209–229, 1991.
60. P. Lee, S. Wei, J. Zhao, and N. Badler, "Strength guided motion," *ACM Computer Graphics (Proc. of SIGGRAPH'90)*, Vol. 24, No. 4, pp. 253–262, 1990.
61. S. Loizidou and J. Clapworthy, *Legged locomotion using HIDDS*. Tokyo: Springer-Verlag, 1993.
62. T. Moeslund and E. Granum, "A survey of computer vision-based human motion capture," *Computer Vision and Image Understanding*, Vol. 81, No. 3, pp. 231–268, 2001.
63. G. Johansson, "Visual motion perception," *Scientific American*, Vol. 232, No. 6, pp. 75-80, 1975.
64. J. Carranza, C. Theobalt, A. Magnor, and H.-P. Seidel, "Free-viewpoint video of human actors," *ACM Transactions on Graphics (Proc. of SIGGRAPH'03)*, Vol. 22, No. 3, pp. 569-577, 2003.
65. A. Sundaresan and R. Chellappa, "Markerless motion capture using multiple cameras," in *Proceedings of the Computer Vision for Interactive and Intelligent Environment (CVIIE05)*, 2005.
66. O. Arikan and D. Forsyth, "Interactive motion generation from examples," *ACM Transactions on Graphics (Proc. of SIGGRAPH'02)*, Vol. 21, No. 3, pp. 483–490, 2002.
67. A. Bruderlin and L. Williams, "Motion signal processing," *ACM Computer Graphics (Proc. of SIGGRAPH'95)*, pp. 97–104, 1995.
68. M. Unuma, K. Anjyo, and R. Tekeuchi, "Fourier principles for emotion-based human figure animation," *ACM Computer Graphics (Proc. of SIGGRAPH'95)*, Vol. 29, No. 4, pp. 91–96, 1995.
69. K. Huang, C. Chang, Y. Hsu, and S. Yang, "Key probe: A technique for animation keyframe extraction," *The Visual Computer*, Vol. 21, No. 8–10, pp. 532–541, 2005.
70. O. Arikan, "Compression of motion capture databases," *ACM Transactions on Graphics (Proc. of SIGGRAPH'06)*, Vol. 25, No. 3, pp. 890–897, 2006.
71. D. Kochanek and R. Bartels, "Interpolating splines with local tension, continuity, and bias control," *ACM Computer Graphics (Proc. of SIGGRAPH'84)*, Vol. 18, No. 3, pp. 33–41, 1984.
72. J. Steketee and N. Badler, "Parametric keyframe interpolation incorporating kinetic adjustment and phrasing control," *ACM Computer Graphics (Proc. of SIGGRAPH'85)*, Vol. 19, No. 3, pp. 255–262, 1985.
73. R. Bartels and I. Hardke, "Speed adjustment for keyframe interpolation," in *Proceedings of Graphics Interface*, pp. 14–19, 1989.
74. N. Badler, C. B. Phillips, and B. L. Webber, *Simulating Humans: Computer Graphics, Animation, and Control*. Oxford University Press, 1999.
75. L. Bezault, R. Boulic, N. Magnenat-Thalmann, and D. Thalmann, "An interactive tool for the design of human free-walking trajectories," in *Proceedings of Computer Animation*, pp. 87–104, 1992.
76. A. Bruderlin and T. W. Calvert, "Goal-directed dynamic animation of human walking," *ACM Computer Graphics (Proc. of SIGGRAPH'89)*, Vol. 23, No. 3, pp. 233–242, 1989.
77. ——, "Interactive animation of personalized human locomotion," *ACM Computer Graphics (Proc. of SIGGRAPH'93)*, Vol. 29, No. 4, pp. 17–23, 1993.

78. H. Ko, "Kinematic and dynamic techniques for analyzing, predicting, and animating human locomotion," Ph.D. dissertation, Department of Computer and Information Science, University of Pennsylvania, 1994.
79. D. Zeltzer, "Motor control techniques for figure animation," *IEEE Computer Graphics and Applications*, Vol. 2, No. 9, pp. 53–59, 1982.
80. F. Multon, L. France, P. Cani-Gasguel, and G. Debunne, "Computer animation of human walking: A survey," *Journal of Visualization and Computer Animation*, Vol. 10, pp. 39–54, 1999.
81. R. Bartels, J. C. Beatty, and B. A. Barsky, *An Introduction to Splines for Use in Computer Graphics and Geometric Modeling*. Los Altos, CA: Morgan Kaufmann, 1987.
82. B. Guenter and R. Parent, "Computing the arclength of parametric curves," *IEEE Computer Graphics and Applications*, Vol. 10, No. 3, pp. 72–78, 1990.
83. J. Saunders, V. T. Inman, and H. D. Eberhart, "The major determinants in normal and pathological gait," *Journal of Visualization and Computer Animation*, Vol. 35-A, No. 3, pp. 543–558, 1953.
84. P. Richer, *Artistic Anatomy*. Watson-Gutpill Publications, 1981.
85. W. Maurel, Y. Wu, N. Magnenat-Thalmann, and D. Thalmann, *Biomechanical Models for Soft Tissue Simulation*. Berlin/Heidelberg: Springer-Verlag, 1998.
86. S. Delp and J. Loan, "A computational framework for simulating and analyzing human and animal movement," *IEEE Computing in Science & Engineering*, Vol. 2, No. 5, pp. 46–55, 2000.
87. Y. Lanir, *Skin Mechanics*. New York: McGraw-Hill, pp. 11–16, 1987.
88. MetaCreations Software, Inc., Poser 4, available at `http://www.metacreations.com/products/poser` (Poser 7 is available at `http://www.e-frontier.com/go/poser`).
89. L. Kavan and J. Zara, "Real time skin deformation with bones blending," in *Proceedings of WSCG'2003 (Short Papers)*, pp. 69–74, 2003.
90. Autodesk, "3ds Max", http://usa.autodesk.com.