Motion Captured Data Driven Skinning and Animation of 3D Virtual Human

Chen Xiaoman

Xiamen Huaxia University, Fujian, Xiamen, 361024, China

Keywords: 3d virtual human, Skin deformation algorithm, Dual quaternions blending skinning, Motion captured data

Abstract: The motion simulation of 3D virtual human is becoming a hot research topic in recent years in both academia and industry. However, the current mainstream skin deformation algorithm, that is, linear blending skinning (LBS) algorithm will cause some distinct distortion problems such as skin collapsion or contortion. To void these distortions, this paper proposed a dual quaternions blending skinning (DQBS) algorithm for skin deformation that is driven by the motion captured data. Firstly, the skeleton model of the 3D virtual human is extract from the motion captured data files, and further binding to the skin model built by the 3D modeling software. Then the DQBS algorithm is introduced to deform the skin and its algorithm flow chart is proposed. Finally, a simulation software system is developed and both LBS and DQBS algorithms are simulated and compared in the system. The result shows that the proposed DQBS algorithm can improve the distortion caused by LBS.

1. Introduction

Nowadays, 3D virtual human motion simulation technology has become a hot topic in academic and entertainment circles, and a variety of virtual reality and augmented reality applications emerge in endlessly, such as 3D movies and games. In order to enable 3D virtual human can achieve highly humanized body language and movements, the mainstream way is to use motion data capture equipment to obtain real-time data of key parts of the human body in reality, that is, motion capture data, and mapped to the corresponding joints of the established 3D virtual human model, so as to drive the 3D virtual human to achieve the same action [1]. Motion capture data is easy to obtain and has a strong sense of reality.

The key of 3D human body modeling is how to establish an efficient 3D virtual human skeleton model and skin model. When the motion capture data drives the skeleton movement, it can drive the skin point to deform at the same time, and then present the realistic virtual human action. At present, many three-dimensional virtual human models adopt skeleton and skin models, which mainly include two types: 1) double-layer model: only contains bone layer and skin layer, the inner skeletal layer is the human skeleton structure, and the outer skin layer is the polygonal mesh skin that surrounds the skeleton [2, 3]. 2) Multi-layer model: based on the consideration of human anatomy, muscle layer and fat layer were added between the skeletal layer and skin layer of the double-layer model [4, 5]. Compared with the multi-layer model, the double-layer model is simpler in structure, more convenient in motion control, and faster in computer rendering, so it is more suitable for scenes with high real-time requirements.

Since the skin layer of the three-dimensional virtual human is on the surface of the whole human body model, its deformation effect will directly affect the reality degree of movement of the three-dimensional virtual human. Skin deformation is dynamic and requires high real-time performance. The SDD (Skeleton Driven Deformation) algorithm proposed by Thalmann et al. in 1987 is a classic skinning algorithm, which affects skin deformation through the weight of bones on the surrounding skin vertices [6]. SDD is a linear blending skinning algorithm (LBS), which assumes that skin vertices are affected by multiple bones at the same time, and the position of skin vertices can be updated by simple weighting calculation. Subsequently, many scholars have proposed many new skinning algorithms on the basis of SDD, including spherical blending skinning algorithm (SBS), dual quaternions blending skinning algorithm (DQBS) and so on [6-8].

  DOI: 10.25236/mmmce.2020.025

Based on this, this paper proposes a 3D virtual human motion simulation method driven by motion capture data, which establishes a typical two-layer 3D virtual human geometry model, and proposes a dual quaternion skinning algorithm driven by motion capture data to realize the motion simulation of 3D virtual human. The experimental results show that the method used in this paper is simple and convenient, and can obtain the realistic movement of three-dimensional virtual human.

2. Mapping Relation between Motion Capture Data and 3d Virtual Human Model

2.1 Motion Capture Data

In this paper, the ASF (Acclaim Skeleton File) standard human skeleton model provided by Carnegie Mellon University is used. The model has 31 joints and 62 degrees of freedom. And AMC (Acclaim Motion Capture) motion sequence stores a set of human motion and posture frames with timing characteristics.

In the ASF file, the root node defines the overall position and orientation of the 3D virtual human model in the world coordinate system, while the location information of all other related nodes in the hierarchical structure of the skeleton model is obtained based on location and orientation of the root node. Each frame of the AMC file contains a description of the rotation and translation information of the root node, as well as the rotation information of other bones or related joint points. The formal description of AMC motion data frame is as follows:

$$f_i ::= \left\langle t_i^0, r_i^0, r_i^1, r_i^2, \cdots, r_i^n \right\rangle \tag{1}$$

$$t_i^0 = (t_{i,x}^0, t_{i,y}^0, t_{i,z}^0) \tag{2}$$

$$r_i^j = (r_{i,x}^j, r_{i,y}^j, r_{i,z}^j) \tag{3}$$

Where, f_i represents the motion data of the i-th frame, t_i^0 represents the translation information of the root node in the i-th frame, r_i^0 represents the rotation information of the root node, and r_i^j represents the rotation information of the j-th segment bone in the i-th frame. $t_{i,x}^0, t_{i,y}^0, t_{i,z}^0$ represent the translation component of the root node along the X, Y, Z axis in the world coordinate system in the i-th frame, $r_{i,x}^j, r_{i,y}^j, r_{i,z}^j$ represent the rotation component of the j segment bone around the X, Y, Z axis in the world coordinate system.

Assuming that the motion data of the AMC file is made up of a sequence of frames, the motion data can also be formally described in the following form:

$$MD ::= \left\langle f_1, f_2, \cdots, f_m \right\rangle \tag{4}$$

After analyzing the data in ASF and AMC, the human skeleton model established in this paper is shown in Figure 1:

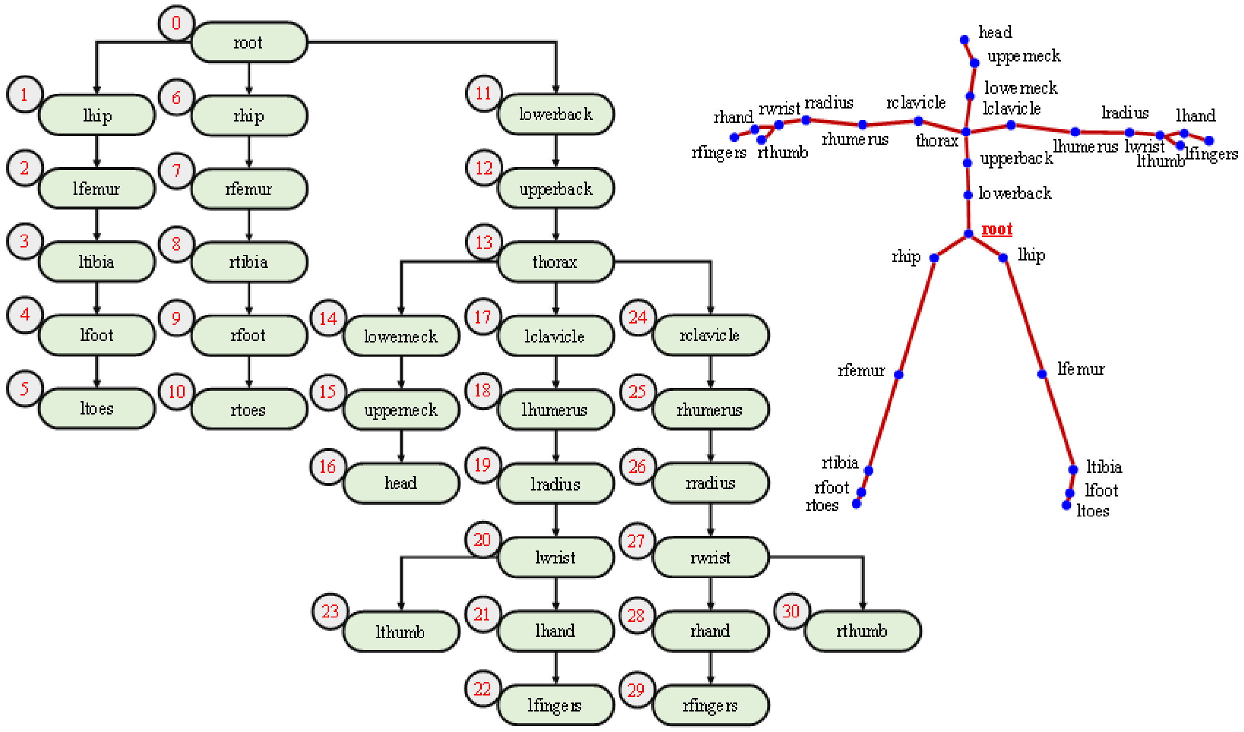


Fig.1 3d Virtual Human Skeleton Model

2.2 3D Human Body Model

In this paper, 3D modeling software is used to create 3D human body model. A complete 3D model includes model and texture. The model is actually a polygon composed of points, lines and surfaces. The 3D human body model created in this paper is shown in figure 2.

Fig.2 3d Human Body Model Rendering

The three-dimensional human body model is made up of a certain number of patches. “Vertex Position” represents the three-dimensional coordinates of the geometry vertices, “Face” represents the connection order of the index of four vertices of each quadrilateral patch, and “Vertex Normal” represents the vertex normal. After loading the mapping file, the mapping coordinate “Texture Vertex” is used to complete the mapping between the model and the map. Three-dimensional human body model can be represented by the following data structures, as shown in Table 1.

Table 1 Data Structure of Skin Layer of 3d Human Body Model

Data name	Data structural	Remarks

VertexPosition	class VertexPosition{
	float vx;
	float vy;
	float vz;
	float *weight;
	};
		“VertexPosition” stands for the coordinates of the skin vertexs.
		“weight” is the weight that the vertex is affected by the bone.

TextureVertex	class TextureVertex{	“TextureVertex” is the coordinates of the texture.

	float vtx; float vty; float vtz; };	

VertexNormal	class VertexNormal{
	float vnx;
	float vny;
	float vnz;
	};
		"VertexNormal" is the normal of the vertex.

Vertex	class Vertex{
	public:
	VertexPosition *pVp;
	TextureVertex *pTv;
	VertexNormal *pVn;
	Vertex *pNext;
	};
		"Vertex" is the vertex information, containing vertex coordinate, texture coordinate and vertex normal.

Face	class Face{
	int vertex_num;
	Vertex *pv;
	};
		"Face" means the polygonal faces that make up the model surfaces.
		"vertex_num" is the numbers of the vertices
		"pv" represents the index of the vertex

2.3 Binding of Skeleton and 3d Human Body Model

The key of 3D virtual human animation is to use motion capture data to drive the deformation of human skin mesh model, and the motion capture data is carried by bones. Therefore, it is necessary to establish a certain connection to associate the vertices on the skin mesh with one or more nearby joint nodes, so that the skin vertices are affected by the relevant nodes, and this influence coefficient is the weight. In general, there are no more than 4 nodes that affect a skin vertex, and the sum of their weights is 1.

In order to achieve a good deformation effect of the virtual human skin model, it is necessary to smooth the transition of the weights of the two adjacent vertices on the surface of the model. In this paper, the method of "Linear Gradient of Bone Projection Direction" is used to calculate the weight value of skin vertices affected by bones. First of all, we need to match the position of the bone and the skin model in the initial state; secondly, we use the bounding box expanded by the bone to delineate the influence area of the bone; finally, we use the linear gradient of the projection length of the distance from the model vertex to the joint in the bone direction to determine the weight of the vertex, and use the nearest distance to assign the missing vertices outside the bounding box.

3. Dual Quaternion Blending Skinning Algorithm

3.1 Dual Quaternion Represents Rigid Transformation

Suppose there is a vector $\vec{r} = (r_0, r_1, r_2)$ in three-dimensional space, and the corresponding unit dual quaternion can be constructed as:

$$\hat{\boldsymbol{r}} = 1 + \varepsilon(r_0 \boldsymbol{i} + r_1 \boldsymbol{j} + r_2 \boldsymbol{k}) \tag{5}$$

When the dual part q_ε of the unit dual quaternion $\hat{\boldsymbol{q}} = \boldsymbol{q} + \varepsilon \boldsymbol{q}_\varepsilon$ is equal to 0, $\hat{\boldsymbol{q}}\hat{\boldsymbol{r}}\overline{\hat{\boldsymbol{q}}^*}$ is used to represent the rotation transformation of the rigid body represented by the vector $\vec{r}$ in three-dimensional space, that is:

$$\begin{aligned} \hat{\boldsymbol{r}}' &= \hat{\boldsymbol{q}}\hat{\boldsymbol{r}}\overline{\hat{\boldsymbol{q}}^*} = \boldsymbol{q}[1 + \varepsilon(r_0 \boldsymbol{i} + r_1 \boldsymbol{j} + r_2 \boldsymbol{k})]\boldsymbol{q}^* \\ &= \boldsymbol{q}\boldsymbol{q}^* + \varepsilon \boldsymbol{q}(r_0 \boldsymbol{i} + r_1 \boldsymbol{j} + r_2 \boldsymbol{k})\boldsymbol{q}^* \\ &= 1 + \varepsilon \boldsymbol{q}(r_0 \boldsymbol{i} + r_1 \boldsymbol{j} + r_2 \boldsymbol{k})\boldsymbol{q}^* \end{aligned} \tag{6}$$

where $\boldsymbol{q}(r_0\boldsymbol{i}+r_1\boldsymbol{j}+r_2\boldsymbol{k})\boldsymbol{q}^*$ is the rotation transformation of common quaternion $\boldsymbol{q}$ on vector $\vec{r}$.

Suppose that the translation transformation of a rigid body is represented by a vector $\vec{t}=(t_0,t_1,t_2)$, and the unit dual quaternion constructed by the translation vector is:

$$\hat{\boldsymbol{t}}=1+\frac{\varepsilon}{2}(t_0\boldsymbol{i}+t_1\boldsymbol{j}+t_2\boldsymbol{k}) \tag{7}$$

When the dual quaternion $\hat{\boldsymbol{r}}=1+\varepsilon(r_0\boldsymbol{i}+r_1\boldsymbol{j}+r_2\boldsymbol{k})$ is used to represent the translation transformation of vector $\vec{r}$, the transformed dual quaternion can be expressed as:

$$\begin{aligned}\hat{\boldsymbol{r}}'' &= \hat{\boldsymbol{t}}\hat{\boldsymbol{r}}\overline{\hat{\boldsymbol{t}}^*} = [1+\frac{\varepsilon}{2}(t_0\boldsymbol{i}+t_1\boldsymbol{j}+t_2\boldsymbol{k})][1+\varepsilon(r_0\boldsymbol{i}+r_1\boldsymbol{j}+r_2\boldsymbol{k})][1+\frac{\varepsilon}{2}(t_0\boldsymbol{i}+t_1\boldsymbol{j}+t_2\boldsymbol{k})] \\ &= 1+\varepsilon[(r_0+t_0)\boldsymbol{i}+(r_1+t_1)\boldsymbol{j}+(r_2+t_2)\boldsymbol{k}]\end{aligned} \tag{8}$$

It can be seen that the unit dual quaternion $\hat{\boldsymbol{t}}=1+\frac{\varepsilon}{2}(t_0\boldsymbol{i}+t_1\boldsymbol{j}+t_2\boldsymbol{k})$ can realize translation transformation of a rigid body, and the translation amount is determined by the dual part of the unit dual quaternion, that is, $\vec{t}=(t_0,t_1,t_2)$.

Because the transformations of the skin vertices and bones involved in the motion of the three-dimensional virtual human body in this paper are both translation and rotation transformations, the dual quaternion can be used to represent the rigid transformation process in which a rigid body rotates first and then translates, as follows:

$$\hat{\boldsymbol{t}}(\boldsymbol{q}\hat{\boldsymbol{v}}\overline{\boldsymbol{q}^*})\overline{\hat{\boldsymbol{t}}^*} = (\hat{\boldsymbol{t}}\boldsymbol{q})\hat{\boldsymbol{v}}\overline{\boldsymbol{q}^*\hat{\boldsymbol{t}}^*} = (\hat{\boldsymbol{t}}\boldsymbol{q})\hat{\boldsymbol{v}}\overline{(\hat{\boldsymbol{t}}\boldsymbol{q})^*} \tag{9}$$

Therefore, a rigid transformation (translation-after-rotation) can be realized directly by dual quaternion $\hat{\boldsymbol{t}}\boldsymbol{q}$:

$$\hat{\boldsymbol{t}}\boldsymbol{q}=\boldsymbol{q}+\frac{\varepsilon}{2}(t_0\boldsymbol{i}+t_1\boldsymbol{j}+t_2\boldsymbol{k})\boldsymbol{q} \tag{10}$$

where $\hat{\boldsymbol{t}}$ is an unit dual quaternion, $\boldsymbol{q}$ is a common unit quaternion, then $\|\hat{\boldsymbol{t}}\boldsymbol{q}\|=\|\hat{\boldsymbol{t}}\|\|\boldsymbol{q}\|=1$, and $\hat{\boldsymbol{t}}\boldsymbol{q}$ is also an unit dual quaternion. Thus, a rigid translation-after-rotation transformation can be expressed as a unit dual quaternion, in which the rotation information is stored in the common unit quaternion $\boldsymbol{q}$, and the translation information is contained in the imaginary part of $\hat{\boldsymbol{t}}\boldsymbol{q}$.

3.2 Dual Quaternions and Transformation Matrix

In order to transform the skin of a three-dimensional virtual human by using the dual quaternion algorithm, it is necessary to convert the transformation matrix used to directly describe the rotation and translation transformations in space into the dual quaternion $\hat{\boldsymbol{q}}=\boldsymbol{q}+\varepsilon\boldsymbol{q}_\varepsilon$. Since the real part $\boldsymbol{q}$ contains the rotation information in the dual quaternion $\hat{\boldsymbol{q}}$. So the rotation information in the matrix needs to be extracted firstly and converted into the common dual quaternion d. Suppose $\boldsymbol{q}=w_0+x_0\boldsymbol{i}+y_0\boldsymbol{j}+z_0\boldsymbol{k}$, then for any matrix $\mathbf{A}$, we have

$$\mathbf{A}=\begin{bmatrix} a_{11} & a_{12} & a_{12} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

As the rotation information of the matrix $\mathbf{A}$ is contained in its submatrix $[a_{ij}]_{3\times3}, i\in[1,3], j\in[1,3]$, the rotation information can be directly transformed into common quaternion $\boldsymbol{q}$ by the following formula

$$\begin{cases} w_0=\dfrac{\sqrt{a_{11}+a_{22}+a_{33}+1}}{2} \\ x_0=\dfrac{a_{23}-a_{32}}{4w} \\ y_0=\dfrac{a_{31}-a_{13}}{4w} \\ z_0=\dfrac{a_{12}-a_{21}}{4w} \end{cases} \tag{12}$$

Conversely, the rotation information in the common unit quaternion can also be extracted and transformed into the matrix, as follows:

$$\mathbf{A}=\begin{bmatrix} 1-2(y_0^2+z_0^2) & 2(x_0y_0-w_0z_0) & 2(x_0z_0+w_0y_0) & a_{14} \\ 2(x_0y_0+w_0z_0) & \mathbf{1}\text{-}2(x_0^2+z_0^2) & 2(y_0z_0-w_0x_0) & a_{24} \\ 2(x_0z_0-w_0y_0) & 2(y_0z_0+w_0x_0) & \mathbf{1}\text{-}2(x_0^2+y_0^2) & a_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13}$$

The translation information contained in matrix $\mathbf{A}$ can be represented by the vector $\vec{\boldsymbol{a}}=(a_{14},a_{24},a_{34})$, and then the dual quaternion $\hat{\boldsymbol{q}}=\boldsymbol{q}+\varepsilon\boldsymbol{q}_\varepsilon$ is transformed into a unit dual quaternion:

$$\hat{\boldsymbol{q}}'=\frac{\hat{\boldsymbol{q}}}{\|\hat{\boldsymbol{q}}\|}=\boldsymbol{q}'+\frac{\varepsilon}{2}(t_0\,\boldsymbol{i}+t_1\,\boldsymbol{j}+t_2\,\boldsymbol{k})\boldsymbol{q}'=\boldsymbol{q}'[1+\frac{\varepsilon}{2}(t_0\,\boldsymbol{i}+t_1\,\boldsymbol{j}+t_2\,\boldsymbol{k})]=\hat{\boldsymbol{q}}'\hat{\boldsymbol{t}}' \tag{14}$$

Transform the translation matrix into a unit dual quaternion, that is:

$$(a_{14},a_{24},a_{34})=(t_0,t_1,t_2) \tag{15}$$

3.3 Dqbs Algorithm Flow

Suppose that a vertex v is affected by $n\in[2,4]$ segments bones, the global transformation matrices $\mathbf{E}_1,\cdots,\mathbf{E}_n$ of these n segments bones are first transformed into the form of dual quaternions $\hat{\boldsymbol{q}}_1,\cdots,\hat{\boldsymbol{q}}_n$, then $\hat{\boldsymbol{q}}_1,\cdots,\hat{\boldsymbol{q}}_n$ are mixed linearly according to the weight value $w_1,\cdots,w_n$ to calculate the mixed dual quaternion $\hat{\boldsymbol{p}}=\sum_{i=1}^{n}w_i\hat{\boldsymbol{q}}_i$. Unitize $\hat{\boldsymbol{p}}$ and transform the unit dual quaternion into a transformation matrix $\mathbf{E}'$, and finally multiply $\mathbf{E}'$ by the skin vertex in the initial pose to update the position of the skin vertex. The DQBS algorithm flow is shown in Fig.3.

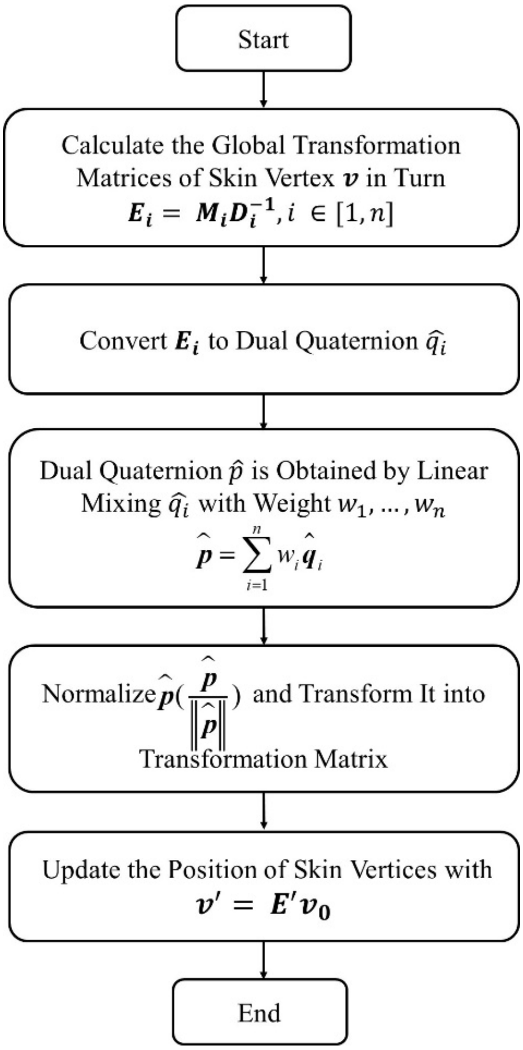


Fig.3 Update the Position of Skin Points with Dqbs Algorithm

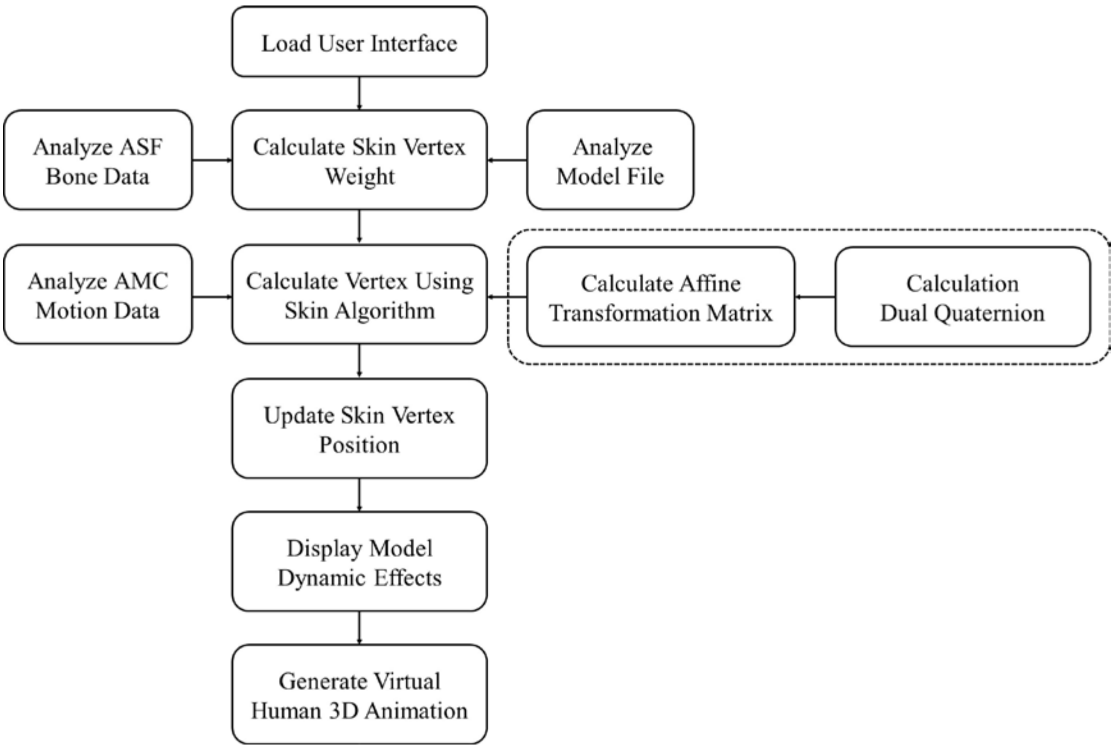


Fig.4 System Flow Chart

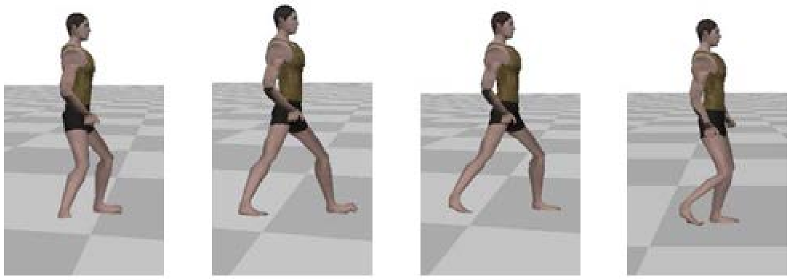
4. Experimental Result

4.1 Dynamic Simulation

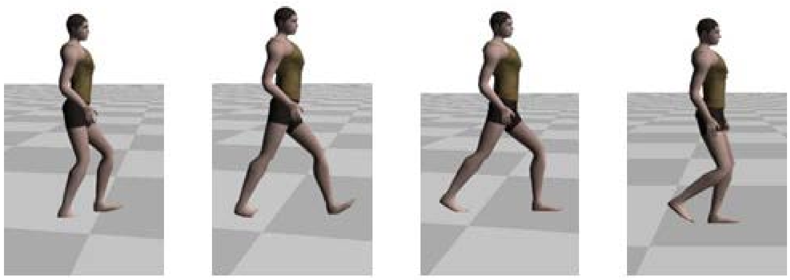
This paper uses Visual Studio 2010 as the development platform, and uses OpenGL (Open Graphics Library) graphics development kit to develop the skin animation visualization system driven by motion capture data. Based on the ASF/AMC file of motion capture data provided by CMU, a virtual human model is made by using 3D modeling software, bind the motion data skeleton to the 3D mesh human skin model to calculate the weight value of the skin vertex affected by the skeleton. The influence of bones on skin is calculated by LBS and DQBS algorithms, as well as the skin animation simulation. The system flow chart is shown in Fig.4.

4.2 Effect Comparison between Lbs and Dqbs

Fig.5 Selects the Motion Posture of Four Key Frames in the Walking Motion to Test and Compare the Deformation Effect of the Skin Model under the Two Skinning Algorithms.



a) Effect of LBS



b) Effect of DQBS

Fig.5 Comparison of Skin Deformation in Four Postures of Walking

a-1 a-2

a) Effect of LBS

b-1 b-2

b) Effect of DQBS

Fig.6 Animation Effect Comparison between Lbs and Dqbs in Large Angle Rotation of Joints

It can be seen that distortions such as skin “collapse” and “sugar wrap” appear in the LBS algorithm. The main reason is that the original rigid transformation becomes non-rigid after linear mixing and cannot keep the skin volume constant. Based on the LBS algorithm, the DQBS algorithm converts the rigid transformation matrix that was originally mixed directly into a dual quaternion, and then transforms the dual quaternion into matrix after linear mixing, which can maintain the rigidity and constant skin volume of the transformed matrix. thus fundamentally eliminating the problem of skin distortion that cannot be compensated by the LBS algorithm. Fig.6 shows the animation effect comparison between LBS algorithm and DQBS algorithm when the joint rotates at a large angle.

5. Conclusion

The traditional linear blending skinning algorithm can show the motion posture correctly, which meet the needs of dynamic simulation basically. However, in the detail part, the skin near the joint is easy to deform when the bending angle of the joint is large. After analyzing the defects of the linear blending skinning algorithm, this paper uses the dual quaternion blending skinning algorithm to improve the linear blending skinning algorithm. The effect comparison graph shows that under the same posture, the elbow and knee parts of the LBS algorithm appears “collapse” during bending, while the model using DQBS algorithm appears smoother in the bending part, and the effect is more realistic than LBS. This paper will study how to improve the efficiency of the algorithm and how to integrate the algorithm into the three-dimensional animation software in the future.

References

[1] Kwon T, Cho Y S, Sang I P, et al. Two-Character Motion Analysis and Synthesis[J]. IEEE Trans Vis Comput Graph, 2008, 14(3):707-720.

[2] Vaillant R, Barthe L, Guennebaud G, et al. Implicit skinning: real-time skin deformation with contact modeling[C]. International conference on computer graphics and interactive techniques, 2013, 32(4).

[3] Zhang G, Zhu D, Qiu X, et al. Skeleton-based control of fluid animation[J]. The Visual Computer, 2011, 27(3): 199-210.

[4] Anglesbaptiste, Tarinimarco, Wyvillbrian, et al. Sketch-based implicit blending[J]. ACM Transactions on Graphics, 2017, 18:1-13.

[5] Luo S, Wang J, Yao X, et al. A novel method for determining skin deformation of lower limb in cycling[J]. Journal of The Textile Institute, 2017, 108(9): 1600-1608.

[6] Magnenat-Thalmann N, Thalmann D. The Direction of Synthetic Actors in the film Rendez-vous à Montréal[J]. Computer Graphics and Applications, IEEE, 1987, 7(12): 9-19.

[7] Pennestrì E, Valentini P. Dual quaternions as a tool for rigid body motion analysis: A tutorial with an application to biomechanics[J]. Archive of Mechanical Engineering, 2010, 57(2): 187-205.

[8] Kavan L, Collins S, O'Sullivan C. Skinning with dual quaternions[C]// Symposium on Interactive 3d Graphics and Games. ACM, 2007:39-46.

[9] Maruyama D, Bailly D, Carrier G, et al. High-Quality Mesh Deformation Using Quaternions for Orthogonality Preservation[J]. AIAA Journal, 2014, 52(12): 2712-2729.