Automatic Rigging for Skeletal Animation

Kayra Hopkins
Michigan State University
hopkin95@msu.edu

Charles B. Owen
Michigan State University
cbowen@msu.edu

Introduction

A common method for figure animation in computer graphics is skeletal animation using the smooth skin algorithm. In this method, vertices of the character mesh are associated with joints of an embedded armature commonly referred to as the skeleton. The vertices are displaced based on a weighted sum of spatial transformations due to movement of the bones that make up the skeleton. The position of vertex v_i is determined using Equation 1, where T_j is a 4x4 transformation matrix representing the transformation of bone b_j from a *bind pose*, where a bind pose is defined as a configuration of the skeleton joints that matches the mesh as designed, prior to any deformation.

$$v_i' = \sum_{j=1}^{B} w_{i,j} T_j v_i \qquad (1)$$

A vertex i on the mesh that is rigidly moved by a bone j would have a weight $w_{i,j}=1$ with all other weights zero. A vertex that is subject to motion by more than one bone, such as a point on the elbow, will have non-zero weights for all bones that influence that vertex. Equation 1 is commonly implemented in graphics cards for computer games and other animation application. In most cases the number of bones that can influence a single vertex is hardware limited to four (4).

Not all of the bones in the armature influence the movement of every vertex, so for best results it is important that, for each vertex, we select only the bones that influence and determine the weights only for that subset of bones. Vertex weights are a measure of the influence of each bone on the flexible mesh, mimicking numerically how the flexibility of flesh determines the movement of the skin when a human joint is flexed. The accuracy of these weights is a major influence on how natural the deformations appear. Assigning vertex weights is traditionally a tedious process requiring an expert animator, a variety of complex and non-intuitive algorithms, and the bone influence is usually determined heuristically. This abstract describes a method for selecting the most influential bones for the movement of each vertex and for automatically computing smooth skin weights based on example models captured using surface scanning. Rather than requiring an animator to determine weights and influential bones by hand, iteratively adjusting them until a desired appearance is achieved, we assume an example model is available for several poses that shows exactly how the deformation should appear. The problem is then treated as an inverse problem, wherein the example poses are used to determine influential bones and compute weights that are optimum in a least-squared sense. The methods used in this paper result in a greatly simplified and more automatic rigging technique for skeletal animation.

Related Work

In order to create a more automated approach to skeletal animation two processes need to be considered. The first is the determination of bone influence, i.e. which bones actually influence the vertex. The second process is determines the values of the weights for each vertex. Various techniques have been employed for both the smooth skin weight-solving and bone choice processes.

Wang and Phillips [4] incorporate an additional set of weights for each element of the bone transformation matrix, a process referred to as weight enveloping. This allows for more flexibility in armature movement. While this technique utilizes inverse matrix approaches to solve for the weights, the process does not lead to intuitive weight values.

James and Twigg [2] present a modified approach to the over-constrained system of equations in which the vertices in the base position and the deformed vertices are subject to a scaling parameter. Rhee et al. [4] employ a similar technique based on the work of James and Twigg in their skinning algorithm. This approach is similar to the approach presented in this research.

Baran and Popovic [1] present a method for bone influence based on heat transfer. Merry et al. [2] manually select bone influence. James and Twigg estimate bone influence by choosing based on an error metric, the best B bones, where B is a relatively small number that minimizes error.

Our methodology

First we present the model for bone influence. When deforming an armature, naïve assumptions presume that every vertex is influence by every bone in the armature. While weights can account for lesser influence of some bones in comparison with others, in reality, in most applications only a small subset of bones have any influence on the movement of the vertex. A variety of

algorithms have been employed in address this concern, though the most common approach is to simply assume the closest *n* bones will influence the vertex. An improved model involves choosing the appropriate subset of bones that influence each vertex by movement data, after which the weights are computed for this subset of bones. We present an algorithm to determine which set of joints influence the movement of a particular vertex.

The algorithm proceeds by utilizing a set of surface scan data for a human who has been asked to physically pose in specified ways. Given a rigged armature, for each bone *b* in an armature, we deform the armature from the base pose so that only the position of *b* changes in both the synthetic system and the scan. This results in a new pose *p*. Using this newly deformed armature, for every vertex we compute the expected position of the vertex v_i, if only bone *b* were to influence the vertex. We also compute v_{min}, the threshold below which we consider a vertex not to have moved. v_{min} is calculated by determining all edges incident on a vertex and taking half of the largest of these edges. Comparing v_i, to the actual position of the vertex after the deformation, v_i, if $| v_i' - v_i | <= v_{min}$ then the movement of vertex v can be completely described by bone *b*. All other bones can be removed from the set of influence bones for vertex v. If $| v_i' - v_i | > v_{min}$ then bone *b* in combination with another/other bone(s) creates the set of influence bones. This process is repeated until all influences for each vertex are found.

Next we address the task of determining weights. Given a set of vertices, their positions in a base pose, and a set of reference poses, the weights can be computed using a linear system of equations. Given a vertex v_i on a deformable articulated model in some base pose, a set of reference models in *R* distinct reference poses P_r and the joint transformations needed for each joint to be transformed from the base pose to the reference pose T_{jr}, it is possible to calculate the joint weights for vertex v_i. These values can be represented as a matrix composed of the joint transformation in each reference pose applied to the current vertex (each row represents a reference pose, and within the rows each column represent each joint in the reference model), and a 1-dimesional matrix representing the closest point in each reference pose to the vertex as shown in Equation 2.

$$\begin{bmatrix} T_{11}v & T_{12}v & \cdots & T_{1j}v \\ T_{21}v & \ddots & & \vdots \\ \vdots & & \ddots & T_{r-1\,j}v \\ T_{r1}v & \cdots & T_{rj-1}v & T_{rj}v \end{bmatrix} \begin{bmatrix} w_{v1} \\ w_{v2} \\ \vdots \\ w_{vr} \end{bmatrix} = \begin{bmatrix} v'_{v1} \\ v'_{v2} \\ \vdots \\ v'_{vr} \end{bmatrix} \quad (2)$$

Given these structures our problem takes the form of a system of linear equations $Ax=b$, where A is the matrix of transformations applied to the vertex and b is the vector of closest points. Solving for x will yield a vector containing the joint weights for a particular vertex that are optimum in a least-squared sense. This set of calculations is computed for each vertex in the model. This is repeated for each vertex in the deformable pose to get a complete set of weights for the model.

Using these techniques we present an algorithm that both selects the appropriate influence bones and automatically computes the weight as an inverse solution based on examples of a properly deformed mesh captured using surface scanning techniques.

References

[1] Baran, I. and Popovic, J. 2007. "Automatic rigging and animation of 3D characters", In *ACM Trans. Graph.* 26, 3 (Jul. 2007), 72. DOI = http://doi.acm.org/10.1145/1276377.1276467

[2] Merry, B., Marais, P., and Gain, J. 2006. "Animation space: A truly linear framework for character animation", *ACM Trans. Graph*. 25, 4 (Oct. 2006), 1400-1423. DOI= http://doi.acm.org/10.1145/1183287.1183294

[3] James, D. L. and Twigg, C. D. 2005. "Skinning mesh animations", In *ACM SIGGRAPH 2005 Papers* (Los Angeles, California, July 31 - August 04, 2005). J. Marks, Ed. SIGGRAPH '05. ACM, New York, NY, 399-407. DOI= http://doi.acm.org/10.1145/1186822.1073206

[4] Rhee, T., Lewis, J., and Neumann, U. 2006. "Real-time weighted pose-space deformation on the GPU", *Computer Graphics Forum 25*, 3 (Sept.), 439--448.

[5] Wang, X. C. and Phillips, C. 2002. "Multi-weight enveloping: least-squares approximation techniques for skin animation", In *Proceedings of the 2002 ACM Siggraph/Eurographics Symposium on Computer Animation* (San Antonio, Texas, July 21 - 22, 2002). SCA '02. ACM, New York, NY, 129-138. DOI= http://doi.acm.org/10.1145/545261.545283