Multi-Layer Level of Detail for Character Animation

Yann Savoye, Alexandre Meyer

Workshop in Virtual Reality Interactions and Physical Simulation VRIPHYS (2008)
F. Faure, M. Teschner (Editors)

Multi-Layer Level of Detail for Character Animation

Yann Savoye † and Alexandre Meyer ‡

Lyon Research Center for Images and Intelligent Information Systems (LIRIS), France.
UMR 5205 CNRS, INSA-Lyon, University Claude Bernard Lyon 1, Ecole Centrale of Lyon and University Lumière Lyon 2

Abstract

Real-time animation of human-like characters has been an active research area in computer graphics. Nowadays, more and more applications need to render various realistic scenes with human motion in crowds for interactive virtual environments. Animation and level of detail are well explored fields but little has been done to generate level of detail automatically for dynamic articulated meshes. Our approach is based on the combination of three interesting layers for run-time level of detail in character crowd animation: the skeleton, the mesh and the motion. We build a Multiresolution Skeletal Graph to simplify the skeleton topology progressively. In contrast with previous works, we use a Dual-Graph Based Simplification for articulated meshes, where the triangle decimation is driven by triangle compactness, to build a dynamic, continuous, progressive and selective mesh level of detail. We also present Power Skinning to ensure the stability of Linear Smooth Skinning, during the simplification, with an efficient multi-weight update rule.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Three-Dimensional Graphics and Realism]: Animation, Virtual reality; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling

1. Introduction

Real-time animation of human-like characters has been an active area of research in computer graphics for several years. In addition to crowd behavior simulation, which is not the topic of this paper, the independent animation of each character inside of a large crowd in real-time poses a major problem in terms of performance and realism. Motion capture data sets and animation techniques are used widely in a variety of applications, such as medicine, sports, serious games, video games, and film production, where synthetic crowd generation and processing techniques are needed. In this paper, we present an overview of the related works in a number of different areas in order to identify the weaknesses of these methods and to present our contributions. We introduce a new three-layer level of detail hierarchy model specific to bone-controlled articulated meshes. A critical analysis of work performed and results achieved allow us to think about new perspectives for future work. This work has been realised in the context of the *PlayAll* project.

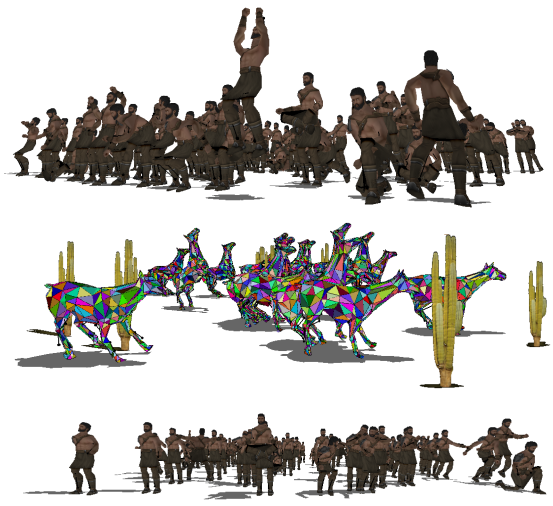

Figure 1: *Automatic generation of crowd scenes.*

† research@yannsavoye.com
‡ alexandre.meyer@liris.cnrs.fr

Computing the mesh deformation of every character is CPU time consuming. Moreover, avoiding repetitive animation is hard to do by hand for an artist. The principle objective of this paper is to propose an automatic method for generating relevant level for animated characters by adapting, at run-time, the complexity of the models without resorting to pre-computation. Indeed, it is not necessary to display or animate all details of the geometry of a character when it is far from the observer. Real-time display is possible if the character animations and models are set at different levels of detail: a precise level for characters close to the camera and more coarse levels for more distant characters. Incontestably, the generation of animation on a large scale needs to take level of detail into account inside the graphics pipeline.

2. Background and Related Works

The basic idea is to view character animation as an articulated mesh or skeleton-driven animation problem. We propose an overview of fundamental tools used in character animation and level of detail that are useful for crowd animation.

2.1. Skeletal Animation

The realistic rendering of human models uses a simple anatomical character layered model with four layers: skeleton layer, muscle layer, skin layer and cloth layer. In our work, we define the character model using a deformable, articulated layered model with two layers: skeleton and skin. Most real-time characters are modeled as an arbitrary 2-manifold triangle mesh that provides volume to the skeleton and simulates the skin. Every mesh is defined by its geometry and its connectivity. A skeleton of animation $\mathcal{S} = (\mathcal{J}, \mathcal{F})$ is made of a set $\mathcal{J}$ of several joints assembled in a hierarchy and a set $\mathcal{F}$ of frames called the motion data. A skeleton may be thought of as a collection of rotational joints. Motion data consists of a bundle of motion signals sampled in a sequence of frames.

The mesh $\mathcal{M}$ can be attached or bound to an underlying skeleton through a process called *skin binding*. Each vertex of a polygonal mesh is attached to one joint or more joints. A blend weight w_{j_k,v_i} can be defined to measure the influence of a joint j_k on a vertex v_i. We assume that $w_{j_k,v_i} = 0$ if the vertex v_i is not influenced by the joint j_k. We define the complete joint-vertex influence set by a list of tuples to represent the joint-vertex attachments or assignments:

$$\mathcal{A} = \left\{ \left(j_k; v_i; w_{j_k,v_i} \right), \forall j_k \in \mathcal{J}, \forall v_i \in \mathcal{M} \right\}$$

The full joint-vertex influence could be estimated and adjusted by computing normalized vertex weight values for each association between a vertex and a joint. The *Point-to-line Mapping*, *Containment Binding* and *Nearest Joint Assignment* algorithms can be used to adjust weights when each vertex is attached to a joint.

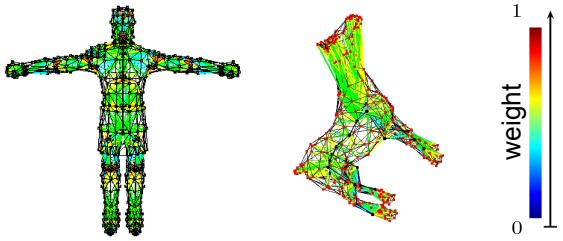

Figure 2: *Binding.*

Skinning Algorithms move each vertex according to the set of attached joints, taking the blend weight influences into account. Kavan and Zara introduce in [KZ03] a method called bone blending which has the capacity to overcome the artifacts of vertex blending. Mohr and Gleicher show in [MG03a] a very fast algorithm called *Linear Smooth Skinning* or *Linear Blend Skinning* where each vertex is assigned to a set of influencing joints. Kavan and co-workers introduce in [KCZO07] the *Dual Quaternion Skinning* for rigid transformation blending. finally a multi-weight formulation for skinning is used in [WP02].

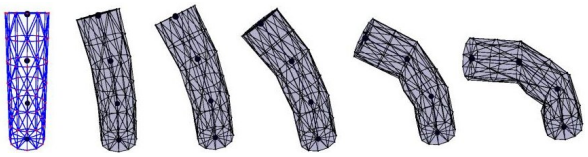

Figure 3: *Linear Smooth Skinning.*

Linear Smooth Skinning (LSB) is a widely used and important traditional skinning algorithm known under many names, such as *Smooth Skinning* and *Skeleton Subspace Deformation* (SSD). This technique works by attaching a mesh vertex v_i with a set of joints. The sum of all weights for a given vertex v_i must be normalized.

$$\sum_{k=1}^{n} w_{j_k,v_i} = 1 \quad ; \quad 0 \leq w_{j_k,v_i} \leq 1$$

A blend vertex is obtained by determining the result transformation of the weight average of all attached joint transformations. We obtain the follow formula:

$$\left(v'_{i,f}\right)_{global} = \sum_{k=1}^{n} \left(w_{j_k,v_i} \cdot (v_i)_{local_{j_k,f}} \cdot W_{j_k,f} \right)$$

where:

- $(v_i)_{local_{j_k,f}}$ is the default position of the vertex v_i in the j_k joint local coordinate system deformed by the frame f;
- $W_{j_k,f}$ represents the *world space matrix* associated with the joint j_k for the frame f;
- w_{j_k,v_i} represents the normalized weight for the joint j_k associated with the vertex v_i.

2.2. Level of Detail Overview

The computation and storage requirements for scenes such as those used in virtual reality applications by far exceeds the capacity of modern graphical hardware. Traditionally, pre-computed static level of detail of a model is a common technique for improving rendering performance in real-time animation systems. The main idea is to not expend rendering resources on model detail when the details cannot be perceived.

Mesh Level of Detail: Mesh simplification is a key research area in scientific visualization and virtual reality. By representing distant objects with lower LoD and nearby objects with higher LoD, rendering is accelerated and interactivity is increased. No algorithm today excels at simplifying all animated models with a high visual fidelity. In order to control the simplification and preserve topology, the approximation error should be measured locally. A surface simplification using the *Quadric Error Metric* is introduced by Garland and Heckert in [GH97]. *Progressive Meshes* is one of the most popular real-time triangle reduction algorithms for general polygonal manifolds, introduced by Hoppe in [Hop96].

Animation Level of Detail: Simplification techniques work very well for static meshes but do not handle the case of deforming objects. Mohr and Gleicher present in [MG03b] a *Deformation Sensitive Decimation* method based on the ideas of Garland and Heckbert for measuring the contraction cost of edges. Pilgrim *et al.* present in [PAMS06] a method called *Progressive Skinning*. A view and pose independent method for the automatic simplification of skeletally articulated meshes is offered in [DR05]. The method uses an edge collapsed contraction with *Quadric Error Metric* and a linear skinning weight update rule. Kavan *et al.* introduce in [KDC*08] a novel representation of virtual characters called *Polypostors* with 2D polygonal impostors. Mukai and Kuriyama introduce the idea of motion level of detail in [MK07] using an LoD control method of motion synthesis with a multilinear model. Ahn and Wohn present in [AW04] a motion level of detail framework.

2.3. Weaknesses in current methods

Weaknesses in current methods often appear in video games that use a reduced collection of pre-computed static meshes for the same character at various levels of detail. LOD techniques based on mesh simplification like [Hop96] can produce low detail approximations for a single pose but do not provide any guarantee of fidelity for other poses leading to animated characters that appear non-continuous and unrealistic. Techniques, like the *Quadric Error Metric*, that work well on a single character, are effectively impossible to apply in the context of a crowd scene where characters are designed to be unique. Ensuring the best quality of a simplification requires a huge computational cost for no significant visual improvement when a character is far away, masked partially by other characters or elements of the decor.

3. Contributions

We introduce a multi-layer LoD hierarchy model specific to articulated meshes by identifying three layers for level of detail in character animation. Our approach called *Multi-layer Animation Level of Detail* adaptively changes the accuracy level of each character in a crowd in real time, handling bottlenecks through several primary contributions including:

- **Skeleton Labeling Algorithm:** we suggest a new algorithm to label joints and to cluster them by their importance in the topology in order to be able to sort them in a decimation order according to the joint energy.
- **Multiresolution Skeletal Graph:** we present a dynamic skeleton simplification algorithm, based on a statistical estimation of the energy, which respects the homotopy of the original skeletal structure. A hierarchical skeleton at different resolutions is generated for the multiresolution skeletal graph. The correctness of the result is guaranteed by graph theory.
- **Dual-Graph Based Simplification for Articulated-Meshes:** in contrast with previous works, our dynamic skeletal-controlled simplification of the mesh uses a new structure called *BindGraphMesh* embedding the dual graph of the mesh to pilot its simplification and embedding the vertex-joint influence set to update the binding information at simplification run-time with a multi-weight update rule. In addition, we implement a simple, selective, progressive, dynamic, view-dependant and pose-dependent mesh simplification method to enhance the performance.
- **Power Skinning Algorithm:** we present a new advanced method called *Power Skinning* for skeleton-driven control of the level of detail based on a multi-weight update rule for *Linear Smooth Skinning* to ensure and maintain the skinning performance during the dynamic mesh simplification.

4. Multi-layer Animation Level of Detail

For large virtual environments, such as a virtual city populated by human-like characters, only a relatively small proportion of characters are relevant to a viewer's interactions at any given moment. The rendering bottleneck of processing a highly detail geometric model is well-known in the computer graphics field. We identify three important layers of level of detail in a skeletal-driven character with skinned mesh vertices, where we can perform an automatic, selective, progressive, dynamic, view-dependent and pose-dependent simplification: the skeleton, the mesh and the motion. The first layer simplifies the skeletal animation by removing joints based on a joint energy metric. The second layer simplifies the mesh geometry by collapsing triangles. The third layer merges key frames to simplify animations. Finally, a *Power Skinning* algorithm is employed to ensure skinning performance during animation is stable.

4.1. Dynamic Skeletal Level of Detail

The idea of simplifying the skeleton is the very foundation of the use of skeletal animation. It was therefore decided to combine some bones and articulation, to get a simplified version that keeps enough degrees of freedom to represent the main movements. Skeletal simplification plays an important role in crowd animation; by simplifying the skeleton, we can reduce the cost of transforming joints. Previous research on skeletal simplification has focused on the improvement of performance of physically based simulation. We propose to exploit joint energy to control the decimation of joints for the progressive simplification of the skeleton.

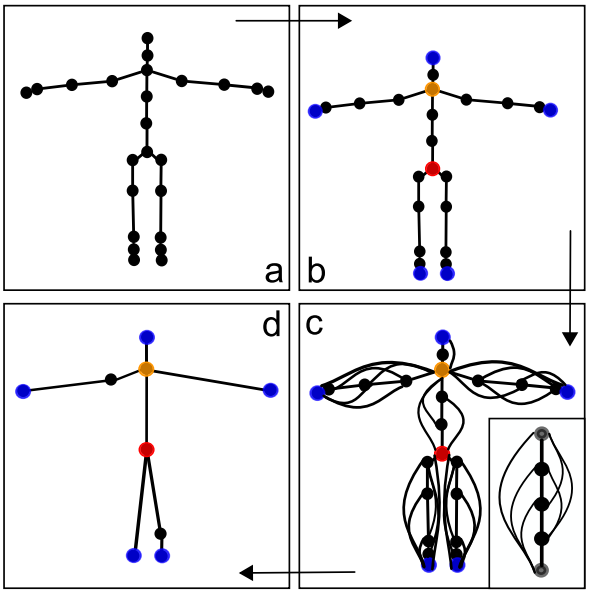

Figure 4: *Multi-resolution Skeleton simplification.*

The principle of our method is to isolate sub-kinematics chains (for instance legs) by identifying key joints in the skeleton topology to decimate progressively intermediate joints according to the joint energy. We propose an overview of our *Dynamic Skeletal Simplification Algorithm* as follows:

- Step 1: loading the animation skeleton and calculation of all relative and absolute joints matrices (Figure 4, a).
- Step 2: computing skeleton statistics including the joint energy (Figure 4, b).
- Step 3: application of the *Skeleton Labeling Algorithm* to cluster joints into categories (Figure 4, b).
- Step 4: building the *Multiresolution Skeletal Graph*. A transitive closure is computed on each kinematic sub-graph of each kinematic sub-chain and all edges are weighted proportionally to their length (Figure 4, c).
- Step 5: extracting a LoD skeleton from the *Multiresolution Skeletal Graph* according to a certain energy threshold (Figure 4, d).

4.1.1. Skeleton Labeling Algorithm

The importance of a joint in the topology can be evaluated by identifying its type. A joint cluster represents a specific kind of joint according to its impact and its role in the topology and the motion. The *Skeleton Labeling Algorithm* is based on joint statistical study of skeleton topology with a computation of average values. We can define many categories in order to represent groups of joints with the same topological properties. In particular:

- χ: joints belonging to the end-effector joints group,
- ζ: joints belonging to the root joints group,
- ψ: joints belonging to the major joints group,
- κ: joints belonging to the minor joints group (unique removable joint group),
- μ: joints belonging to the extreme joints group.

The joint statistical measured values for all joints j_k in a given skeleton topology $\mathcal{J}$ is based on the information and statistical criterion that we detail here:

- $\rho(j_k)$ is the geodesic distance between j_k to the root joint. It corresponds to the depth of the joint in the tree hierarchy of joints,
- $neighbor(j_k)$ is the total number of joints directly linked with it,
- $E(j_k, m_i)$ is the joint energy according to a given motion that we will discuss in details in the next subsection.

We also define functions to compute the average for all statistical criteria values. For example we define a function $averagengb(\mathcal{J})$ which takes a skeleton topology and computes the average of joints in the direct neighborhood for all joints and a function $averagerootdist(\mathcal{J})$ which takes a skeleton topology and computes the average geodesic distance between joints and the root joint:

$$averagengb(\mathcal{J}) = \sum_{\forall j_k \in \mathcal{J}} \frac{neighbor(j_k)}{|\mathcal{J}|}$$

$$averagerootdist(\mathcal{J}) = \sum_{\forall j_k \in \mathcal{J}} \frac{\rho(j_k)}{|\mathcal{J}|}$$

The Skeleton Labeling Algorithm is defined as a heuristic based-rule mechanism on joint properties. This algorithm allows us to identify kinematics sub-chains by finding their extremity represented by two non-minor joints. For all joints $j_k \in \mathcal{J}$, we compute a label $L(j_k) \in \{\kappa, \mu, \psi, \chi, \zeta\}$ using the following identification sort-rules:

1. If $(\rho(j_k) = 0) \Rightarrow (L(j_k) = \zeta)$
2. Else if $(childrennum(j_k) = 0) \Rightarrow (L(j_k) = \chi)$
3. Else if $(neighbor(j_k) > 1.9 * averagengb(\mathcal{J}))$ and $(\rho(j_k) < averagerootdist(\mathcal{J})) \Rightarrow (L(j_k) = \psi)$
4. Else if $(neighbor(j_k) > 1.9 * averagengb(\mathcal{J}))$ and $(\rho(j_k) > averagerootdist(\mathcal{J})) \Rightarrow (L(j_k) = \mu)$
5. Else $L(j_k) = \kappa$

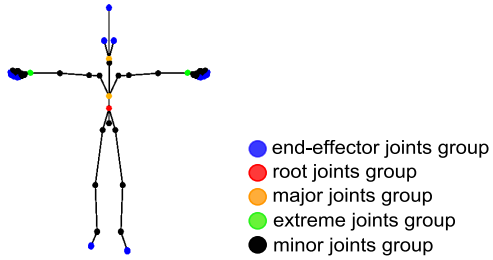

Figure 5: *Joint Labeling.*

4.1.2. Joint Energy

The importance of a joint in a motion is dependent on its level in the hierarchy and can be evaluated by computing a metric we call the joint energy. Moreover the rotation of a joint belonging to the minor group affects the motion very little. We must have a good grasp of the data structure that represents the structure of the human body in order to classify joints for a given motion. We use the following notation:

- h: the depth of the tree representation of the hierarchy of joints,
- $d(j_k)$ the depth of the joint $j_k \in \mathcal{J}$ the hierarchy of joints.

We define the weight factor of the joint $j_k \in \mathcal{J}$ by:

$$\rho(j_k) = \frac{h - d(j_k)}{h}$$

A joint closer to the root will have a bigger impact on the rotation of its children than a joint closer to the leaves. We define the energy of a joint for a given motion m_i with f frames. Each joint j_k has three Euler angles $r_{j_k}(t) = \left(x_{j_k}(t), y_{j_k}(t), z_{j_k}(t)\right)$ which represent the relative orientations of the joint j_k to its parent joint. We estimate the joint energy $E(j_k)$ of a joint j_k for a given motion m_i as follows:

$$E(j_k, m_i) = \rho(j_k) \cdot \sum_{a=1}^{3} \left| \sum_{b=0}^{f-1} \frac{R^a_{j_k}(b+1) - R^a_{j_k}(b)}{Max\left(R^a_{j_k}\right)} \right|$$

where $R^1_j(t), R^2_j(t), R^3_j(t)$ are the three Euler angles of joint j_k at the frame t. $Max\left(R^a_{j_k}\right)$ is the maximum Euler angle of the joint j_k that can be rotated about the a^{th} axis.

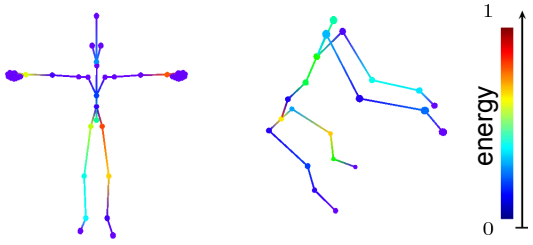

Figure 6: *Joint Energy.*

4.1.3. Multiresolution Skeletal Graph

We propose to build a *Multiresolution Skeletal Graph* to produce a level of detail articulation (Figure 4, d). We build a continuous skeletal level of detail method with heuristics and rules. We build a level of articulation by studying the joint energy that represents the importance of each joint, reflecting how much the animation would be damaged by removing it. We sort joints such that joints with higher importance come first. The *Skeleton Labeling Algorithm* is key to allowing us to identify kinematics sub-chains inside the skeletal graph. A kinematics sub-chain corresponds exactly to the shortest joint chain delimited on each extremity by a non-minor joint.

Multiresolution Skeletal Graph Construction: Skeleton topology is a graph. We convert the tree of bone to a multiresolution graph to produce the skeletal simplification with the conservation of skeleton homotopy. The simplification is based on the *Skeleton Labeling Algorithm* to preserve a homotopic simplification. The *Multiresolution Skeletal Graph* is constructed as a collection of multiresolution kinematics sub-chains. To ensure a topologically valid method, we build a transitive closure on each sub-graph of each kinematic sub-chain.

Joint Decimation: In the next step, joint decimation implies a skeleton simplification. We allow simplification only inside kinematic sub-chains. Each joint, considered as a graph node in the multiresolution structure, could be enabled or disabled, except for joints belonging to the end-effector group, the root group and the major joint group which are always enabled. In the multiresolution graph, joints that do not meet an energy threshold are disabled as well as edges having at least one disabled joint.

Skeletal Topology Simplification and Extraction: We are able to extract a skeleton tree for a given LoD threshold by a graph pruning process based on the joint decimation and shortest edges in the graph. The extraction of a LoD skeleton at a given resolution is based on the extraction of a minimal spanning tree with the minimal edge cost in the graph. We notice that only minor joints could be decimated in the topology during the skeletal simplification to preserve important hub joint in the topology.

4.2. Skinned Mesh Level of Detail

A common polygonal mesh modeling is a shared list of vertices and a list of faces storing pointers for its vertices. This representation using an *Indexed Data Structure* is both convenient and efficient for many purposes, however in some domains it proves ineffective. A mesh simplification algorithm is presented to reduce the number of vertices in a dense mesh of triangles during the animation. Simplifying deforming meshes using a standard algorithm is error prone. We present a run-time mesh simplification algorithm to reduce the number of vertices in a dense mesh of triangles.

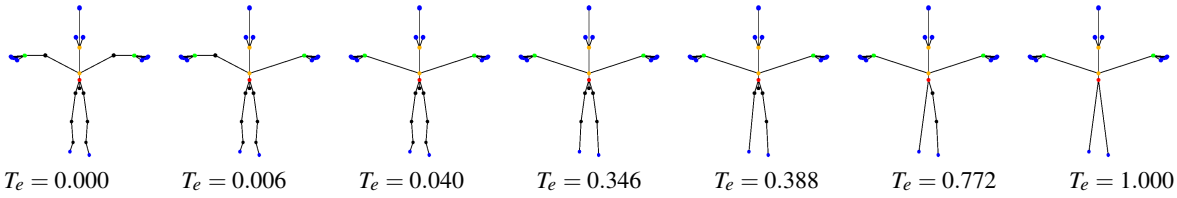

Figure 7: *LoD Skeleton extracted.*

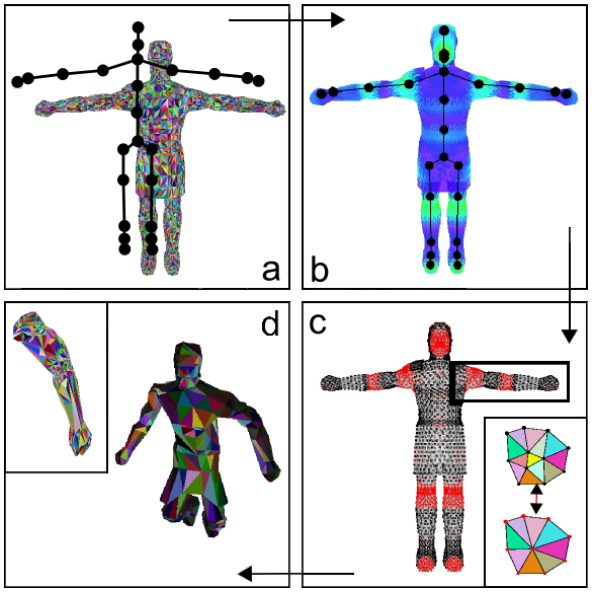

Figure 8: *Multi-resolution mesh.*

We propose an overview of our *Skinned Mesh Level of Detail* as follows:

- Step 1: a highly detailed arbitrary triangulated mesh is loaded inside the *BindGraphMesh* structure, the embedded dual graph of the mesh is built and embedded in the BGM structure; the vertex-joint influences of each vertex are also encapsulated in nodes. The BGM structure elements (faces, nodes) are automatically enriched by relationship information (Figure 8, a).
- Step 2: nodes which are near a joint are marked as ineligible when the corresponding influence is measured by a strong blend weight above a given threshold (Figure 8, b).
- Step 3: at the same time, we compute the triangle compactness or fairness and build a sorted sequence of original faces, according to triangle compactness, which are eligible for the collapse (Figure 8, c).
- Step 4: given a LoD threshold, we do a triangle-collapse operation on triangles that meet the threshold in the sequence of collapsible triangles, and we undo the triangle-collapse operation on others. The decimation process removes and re-textures identified *Mesh Patches*. The sequence of collapsible triangles is enriched and updated with the appearance of new triangles created during the simplification iterations (Figure 8, d).
- Step 5: *Power Skinning* is applied during the re-meshing to ensure skinning stability by calculating blend weights with the update multi-weight rule for the new single vertex of integration according to the sets of vertex-joint influences of the decimated triangle.

4.2.1. Dual-Graph Based Simplification for Articulated-Meshes

We present a new structure called *BindGraphMesh (BGM)* corresponding to an augmented mesh structure with an embedded mesh dual graph and joint-vertex influences in the mesh nodes. The dual graph of a mesh is a key point for a connectivity-oriented mesh simplification. A *BindGraphMesh* consists of a dynamic multiresolution mesh built at run-time and made of an original base mesh at full resolution and a set of collapsible triangles at a given character configuration.

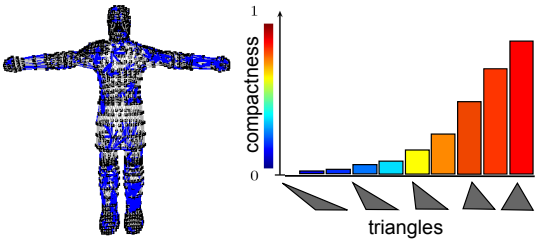

Figure 9: *Dual-Graph Simplification (left) and Triangle Compactness (right).*

4.2.2. Skeletal-Controlled Mesh Simplification

We introduce a simplification method driven by the skeleton joints. We build an automatic method to find specific parts of the model. We identify ineligible nodes corresponding with areas away from the joints that have been drastically simplified. We mark as ineligible nodes according to a threshold based on the higher subset skin weight of each joint. Our

technique allows a skeleton-driven selective LoD and a controllable decimation during the simplification where ineligible vertices are conserved in the mesh. Areas away from the joints will be dramatically simplified, nevertheless we preserve high-detail regions around joints which are significantly deformed during the simplification.

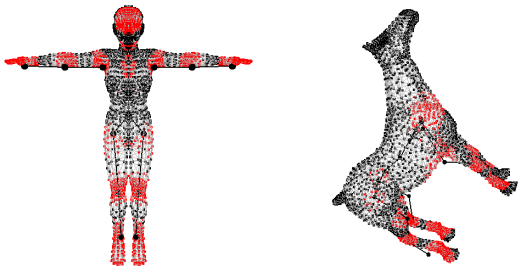

Figure 10: *Ineligible Vertices (in red).*

For a given character $\mathcal{C} = (\mathcal{S}, \mathcal{M}, \mathcal{A})$, we denote by $v_i \in \mathcal{M}$ a vertex of the articulated mesh of the character $\mathcal{C}$. The blend weight average for a given articulation $j_k \in \mathcal{J}$ of $\mathcal{C}$ is defined by:

$$\eta(\alpha) = \frac{\sum w_{j\alpha,v_i}}{|\{w_{j\alpha,v_i}\}|}$$

We build the set of eligible vertices for collapsing operations, denoted by $\tilde{\mathcal{V}}$, as follows:

$$\forall v_i \in \mathcal{M}, v_i \in \tilde{\mathcal{V}} \Longleftrightarrow w_{j_k,v_i} < (\tau \cdot \eta(k), \forall j_k \in \mathcal{J})$$

where τ is a parameterizable coefficient. We use $\tau = 1$ by default in our implementation. If $v_i \notin \tilde{\mathcal{V}}$, v_i cannot disappear during the mesh simplification.

Figure 11: *Decimation Sensitive Deformation.*

4.2.3. Triangle decimation

The triangle decimation process is based on the triangle-collapse algorithm. Removing a triangle implies removing all triangles sharing an edge with this triangle, so a total of four triangles are removed. To preserve mesh topology, it is better to decimate triangles with small angles first. The decimation technique iteratively removes faces from the mesh, re-triangulating the resulting hole after each step. During a triangle-collapse, all vertices of the triangle merge into a new vertex that is carefully positioned to ensure the simplification. To achieve a good result for low computation cost, we collapse triangles at their barycenter. Triangle compactness allows us to sort triangles for decimation and to ensure the correctness of the simplification for a low computational cost and good enough results compared to the *Quadric Error Metric*. The triangle compactness measure, $Compactness_{face}$, is equivalent to the aspect ratio and is given by:

$$Compactness_{face} = \frac{4\sqrt{3a}}{l_1^2+l_2^2+l_3^2}$$

where a is the area of the triangle $face$ and l_i are the lengths of its edges. Taking into account the aspect ratio of triangles helps to select operations that improve mesh quality. A flat triangle has a compactness near zero and should be decimated in priority (Figure 9).

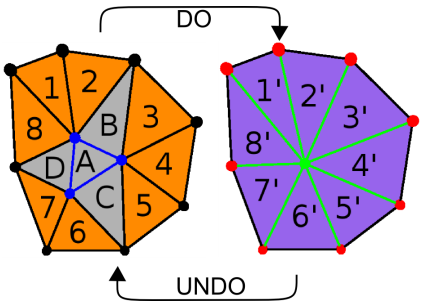

Figure 12: *Dual-Graph Based Triangle Decimation inside of a re-meshing patch.*

To generalize the triangle decimation to n vertices, we define a set of eligible vertices to collapse by a tuple $(v_0, \ldots, v_i, \ldots, v_n) \subset \tilde{\mathcal{V}}$. We denote by $\bar{v}$ the new single vertex resulting of the aggregation of $(v_0, \ldots, v_i, \ldots, v_n)$, and we describe this polygon collapse operation by the contraction as follows:

$$\left(P = \{v_i, \forall i \in [0,n], v_i \in \tilde{\mathcal{V}}\}\right) \rightarrow \bar{v}$$

In the triangle decimation case, three vertices (v_a, v_b, v_c) are contracted in a new representative vertex $\bar{v}$ that must be chosen. This collapse is driven by contracting two edges in the dual graph of the mesh. To ensure maximum of quality for the mesh simplification, we choose the triangle barycenter to be the mesh collapse vertex $\bar{v}$.

For each decimated triangle, we identify its associated *Mesh Patch* (Figure 12), or a sub-part of the mesh, composed of this triangle and all neighboring triangles sharing a node of an edge in the dual graph. A triangle decimation involves the deactivation of all triangles inside the associated patch and the re-meshing of the oldest neighbors (orange triangle) in the interior of the patch with new triangles. We associate its list of old triangles in the patch before the simplification and a list of new triangles in the patch after the simplification with a new contracted position.

4.2.4. Power Skinning Algorithm

One problem remains: the skin. The original skin refers to all the bones. As we do LoD, we would like to have a reduced skin as well at run-time with aggregation of blend weights. Articulated meshes use weighted vertices in order to smooth blending at the joint, so we have to ensure smooth blending during the re-meshing process. The *Power Skinning Algorithm* is a *Multiple Weight Update Rule* that ensures *Linear Smooth Skinning* stability during re-meshing. During the triangle collapsing and re-meshing phase, a node integration is done and it is necessary to update the weight and texture coordinate information at the same time.

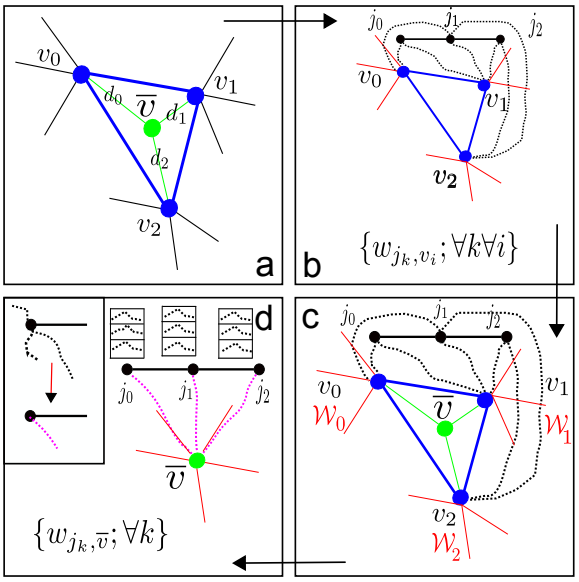

Figure 13: *Power Skinning Algorithm.*

Given a polygon collapse operation (Figure 13.a) $\left(P=\left\{v_i,\forall i\in[0,n],v_i\in\bar{\mathcal{V}}\right\}\right)\rightarrow\overline{v}$ and for $v_\alpha\in P,\alpha\in[0,n]$, we compute the Euclidean distance d_α between v_α and the new common aggregation vertex $\overline{v}$ using the formula:

$$d_\alpha = d_{\text{euclidean}}\left(\overline{v};v_\alpha\right)$$

For the whole configuration (Figure 13.b), we compute a single *Power Distance (D)* as the sum of euclidean distances between $\overline{v}$ and $v_\alpha\in P,\alpha\in[0,n]$:

$$D=\sum_{i=0}^{n}d_i=\sum_{i=0}^{n}d_{\text{euclidean}}\left(\overline{v};v_i\right)$$

The impact of a vertex of the mesh belonging to the triangle to decimate is inversely proportional to the distance, so we compute an Inverse Distance for each vertex $v_\alpha\in P,\alpha\in[0,n]$:

$$dinv_\alpha=\left(\sum_{i=0}^{n}d_{\text{euclidean}}\left(\overline{v};v_i\right)\right)-d_\alpha$$

For the whole configuration, we compute a single *Inverse Power Distance (Dinv)* as the sum of inverse distance between $\overline{v}$ and $v_\alpha\in P,\alpha\in[0,n]$:

$$Dinv=\sum_{i=0}^{n}\left(D-d_i\right)=(n-1)D$$

We introduce the concept of a *Power Weight*, denoted by $\mathcal{W}$, computed for each vertex $v_\alpha\in P,\alpha\in[0,n]$ in order to control the importance of all vertex-joint influence tuples for a given vertex participating in the aggregation of weights:

$$\mathcal{W}_\alpha=\frac{dinv_\alpha}{Dinv}=\frac{D-d_\alpha}{(n-1)D}$$

We use an accumulator to compute the set of vertex-joint influences for the new vertex $\overline{v}$. The concept is that when a triangle is collapsed in a new vertex, the weights of each of the vertices of the collapsed triangle are transferred to the new vertex in parts inversely proportional to their distance from it. The consistency of $\overline{v}$'s blend weights is kept using a function of the inverse distance weighting (Figure 13.d) according to the vertices participating to the aggregation.

$$v_i\mapsto\left(j_k;w_{j_k,v_i}\right),w_{j_k,v_i}\in[0;1]$$

$$\overline{v}\mapsto\left(\overline{v};j_k;\sum_{l=0}^{n}\mathcal{W}_l*w_{j_k,v_l}\right),\forall j_k\in\mathcal{J}$$

$$\begin{bmatrix}w_{\overline{v},j_1}\\ \cdots\\ w_{\overline{v},j_k}\\ \cdots\\ w_{\overline{v},j_n}\end{bmatrix}=\begin{bmatrix}\sum_{l=0}^{n}\mathcal{W}_l*w_{j_1,v_l}\\ \cdots\\ \sum_{l=0}^{n}\mathcal{W}_l*w_{j_k,v_l}\\ \cdots\\ \sum_{l=0}^{n}\mathcal{W}_l*w_{j_n,v_l}\end{bmatrix}$$

The stability of multi-weights skinning is ensured because $\sum_{k=0}^{n}w_{\overline{v},j_k}=1$. Now, the new vertex $\overline{v}$ can be deformed correctly through *Linear Smooth Skinning*. We demonstrate the stability of the skinning of a multi-weighted articulated figure. Whereas our purpose is real-time simplification, *Power Skinning* may also be used by artists to automatically generate LoD levels using our progressive mesh based geometry simplification support.

4.3. Motion Level of Detail

After skeletal and then mesh simplification, we propose the third layer of our complete LoD scheme: motion simplification. An animation is a sequence of key positions. The motion structure yields two relationships among sampled values, inter-frame and intra-frame relationships. If each frame is considered independently, then there could be an undesirable jerkiness between consecutive frames. Therefore, we take into account the inter-frame relationship as well. A character shown close to the viewer can be very detailed, but

once it moves away from the camera, we cannot see the individual fingers of a character anymore. Our motion LOD system will stop animating the fingers automatically, but there can still be a lot of overhead costs from the finger nodes; they still have to be transformed. Also nodes that add small detailed meshes to the character still exist in the skeleton and waste processor power.

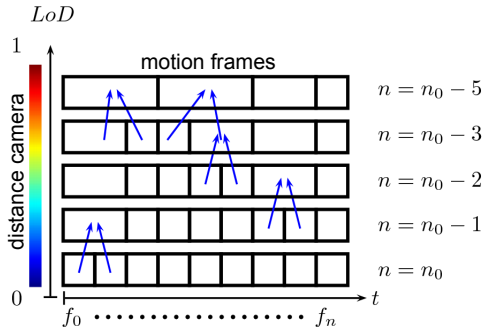

Figure 14: *Motion LoD.*

We merge nearby postures to simplify the number of frames. We build a multiresolution motion data tree where two neighboring frames can be fusionned if the distance of similarity between them is below a given threshold. We achieve an extrapolation by removing a frame from among the two or by merging the two into a single frame, preserving the duration of frames on the result. The distance of similarity of two frames for a given skeleton can be measured by the energy needed to move all joints from one frame to the other. In Figure 14, line 0 is the initial animation with n key positions.

4.4. Controlling the Level of Detail

Our system proposes level of detail control in interactive animation systems to pilot the progressive simplification by eliminating unperceivable details of skeleton, mesh and motion data. The LOD manager picks the best LoD of skeleton, mesh and motion according to the visual importance. Polygons, motion frames and joints removed according to the distance between the character and the camera.

$$d = length(character - eye)$$
$$T = \alpha(t) = \frac{d - dnear}{dfar - dnear} \Longleftrightarrow \alpha(t) = \frac{t_e - t}{t_e - t_s}$$
$$\text{where } t \in [t_s, t_e] \text{ and } \alpha(t) \in [0,1]$$

If the character is currently at a distance d from the camera, we denote the current interpolation value as $\alpha(t)$; we define 3 interpolation values: one for the skeleton ($t_{skeleton}$), another one for the motion (t_{motion}) and another one for the mesh (t_{mesh}). We denote by t_s and t_e the start and the end value of the interpolation respectively. Such an interpolation is easy to use, computationally cheap, guarantees C^0 continuity, and ensures that our system dynamically produces a continuous level of detail.

5. Results & Discussion

We implemented a prototype version of a real-time environment with our multi-layer level of detail. Tests of our implementation were performed and were obtained using an Intel Core Quad CPU workstation with a 2.0 Ghz processor, 2 Go of RAM and a Nvidia Geforce 8800GTS video card. The real time rendering is used to measure the performance of structure construction and run-time processes.

The computational time needed to generate the *Multiresolution Skeletal Graph* for one character is less than a millisecond for all tests. Table 1 gives a performance summary of the overall run-time simulation. The computational time needed to create the *BindGraphMesh* structure ($t_{initBGM}$) and lower mesh level of detail through a series of 10 triangle-collapse operations ($t_{10*decim}$) is provided, according to the number of joints ($joints$) and the number of triangles ($triangles$), for one character.

$joints$	$triangles$	$t_{initBGM}$	$t_{10*decim}$
25	982	142 ms	1 ms
4	169	4 ms	$<$ 1 ms
21	7186	7.725 s	9 ms
24	832	105 ms	2 ms
25	3928	2.230 s	4 ms
21	7186	8.848 s	9 ms
24	6656	11.6 s	12 ms

Table 1: *BindGraphMesh and decimation performances.*

Figure 15 shows the gain of using level of detail on a skeleton used in a crowd made of one to 250 skeletons structures, comparing the frame rate at full resolution and the frame rate obtained with a decimation of 39% of the total number of joints. Notice that for 50 skeletons, our system is two times faster than without LoD.

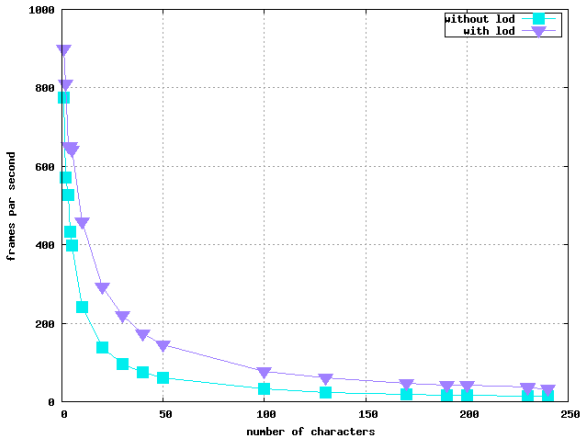

Figure 15: *Benefit of skeletal Lod for crowds.*

As seen in Figure 16, by eliminating 64% of the triangles of the mesh of a character with 3928 triangles at full resolution, we can improve the frame rate by a factor of 1.4. Moreover, errors caused by the simplification of the mesh have no perceptual impact when the character is relatively distant from the camera.

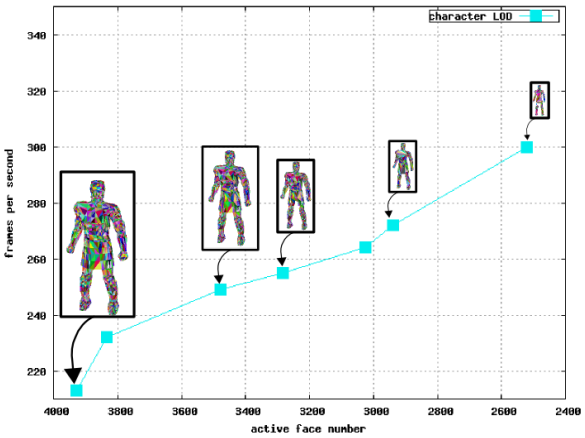

Figure 16: *Benefit of mesh Lod for a character.*

The multiresolution skeletal representation allows us to produce a good skeleton LoD and huge crowds of characters that are extremely detailed. By progressively simplifying the skeleton we can reduce the cost of transforming joints automatically. Our *Dual-Graph Based Simplification for Articulated-Meshes*, using an iterative triangle decimation allows us to compute simplified meshes for a cheaper cost. Desired real-time continuous level of detail can be dynamically extracted at run-time.

6. Conclusion & Future work

Real-time animation of deformable objects is always a trade-off between visual fidelity and computational complexity. A simplification method for crowd scenes is a more challenging issue. In this paper, we present various layers for the application of level of detail specifically designed to work with bone-controlled articulated meshes and their synergistic usage to enhance the overall performance as shown by the results. We describe how to exhibit sensitive key layers for the level of detail through useful associated tools for real-time interactive crowd environments.

Because of a significant amount of programming required, there are some limitations to our prototype system that could be addressed in future works. We select several obvious areas for future works. Swapping among skinning methods could be one other method for performing a skinning level of detail. Indeed, the number of bone influences per vertex may be a fourth layer to our system. Additional work needs to be done to perform a continuous Skeletal LoD and Mesh LoD interlaced. Moreover adding detailed motion clips, with hands and fingers, to an existing skeleton and mesh can increase the realism and should better show the capacity of our three layer LoD technique.

References

[AW04] AHN J., WOHN K.: Motion level of detail: A simplification method on crowd scene. *Computer Animation and Social Agents 2004* (2004), 129–137.

[DR05] DECORO C., RUSINKIEWICZ S.: Pose-independent simplification of articulated meshes. In *Symposium on Interactive 3D Graphics* (Apr. 2005).

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. *Computer Graphics 31*, Annual Conference Series (1997), 209–216.

[Hop96] HOPPE H.: Progressive meshes. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1996), ACM, pp. 99–108.

[KCZO07] KAVAN L., COLLINS S., ZARA J., O'SULLIVAN C.: Skinning with dual quaternions. In *2007 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (April/May 2007), ACM Press, pp. 39–46.

[KDC*08] KAVAN L., DOBBYN S., COLLINS S., ZARA J., O'SULLIVAN C.: Polypostors: 2d polygonal impostors for 3d crowds. In *2008 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games* (February 2008), ACM Press.

[KZ03] KAVAN L., ZARA J.: Real-time skin deformation with bones blending. In *WSCG Short Papers Proceedings* (2003).

[MG03a] MOHR A., GLEICHER M.: Building efficient, accurate character skins from examples. *ACM Trans. Graph. 22*, 3 (2003), 562–568.

[MG03b] MOHR A., GLEICHER M.: Deformation sensitive decimation. *Tech. rep., University of Wisconsin* (2003).

[MK07] MUKAI T., KURIYAMA S.: Multilinear motion synthesis with level-of-detail controls. In *PG '07: Proceedings of the 15th Pacific Conference on Computer Graphics and Applications* (Washington, DC, USA, 2007), IEEE Computer Society, pp. 9–17.

[PAMS06] PILGRIM S. J., AGUADO A., MITCHELL K., STEED A.: Progressive skinning for video game character animations. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Sketches* (New York, NY, USA, 2006), ACM, p. 114.

[WP02] WANG X. C., PHILLIPS C.: Multi-weight enveloping: least-squares approximation techniques for skin animation. In *SCA '02: Proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation* (New York, NY, USA, 2002), ACM, pp. 129–138.