Automated Motion LoD with Rigid Constraints

Junghyun Ahn
VRLab, EPFL
IC ISIM, Station 14
1015, Lausanne
Switzerland
junghyun.ahn@epfl.ch

Byung-Cheol Kim
EECS, KAIST
335 Gwahangno, Yusung
305-701, Daejeon
South Korea
ciel@vr.kaist.ac.kr

Daniel Thalmann
VRLab, EPFL
IC ISIM, Station 14
1015, Lausanne
Switzerland
daniel.thalmann@epfl.ch

Kwangyun Wohn
GSCT, KAIST
335 Gwahangno, Yusung
305-701, Daejeon
South Korea
wohn@kaist.ac.kr

ABSTRACT

Motion LoD (Level of Detail) is a preprocessing technique that generates multiple details of captured motion by eliminating joints. This LoD technique is applied to movie, game or VR environments for the purpose of improving speed of crowd animation. So far, replacement techniques such as 'impostor' and 'rigid body motion' are widely used on real-time crowd, since they dramatically improve speed of animation. However, our experiment shows that the number of joints has a greater effect on the animation speed than anticipated. To exploit this, we propose a new motion LoD technique that not only improves the speed but also preserves the quality of motion. Our approach lies in between impostor and skeletal animation, offering seamless motion details at run time. Joint-elimination priority of each captured motion is derived from joint importance, which is generated by the proposed posture error equation. Considering hierarchical depth and rotational variation of joint, our error equation measures posture difference successfully and allows finding key posture of the entire motion. This 'motion analysis' process contributes error reduction to the next 'motion simplification' stage, where multiple details of motion are regenerated by the proposed motion optimization. In order to reduce the burden of optimization, all the terms of the objective function - distance, string, and angle error - are defined by joint-position vectors. In this aspect, a constrained optimization problem is formulated in a quadratic form. Thus, a sequential quadratic programming (SQP), a nonlinear optimization method, is suitable for resolving this problem. As the result of our experiment, the proposed motion LoD technique improves the animation speed and visual quality of simplified motion. Moreover, our approach reduces the preprocessing time and automates the whole process of LoD generation.

Keywords
Level of detail, Crowd animation, Real-time animation, Motion analysis, Motion optimization

1. INTRODUCTION

In recent animated products, the number of joints and polygons of articulated bodies has been increased substantially. Furthermore, the number of bodies that can be rendered in crowd scenes is also increasing. According to these trends, many studies on crowd animation have been presented to date. These works can be classified into two major categories: (1) realism enhancement - behavior manipulation [TT94] [MT01][ST05][YMP+09], collision detection [Rey87] [TCP06][PAB07], and (2) speed improvement of crowd animation. In the past, enhancement of realism was the major area of research with regard to crowd animation. At present, however, due to GPU evolution, the demands of real-time crowd animation have increased and, for this reason, speed of animation is becoming another important research topic. Skinning vertices are transformed in the GPU rather than in CPU. Therefore, at each frame, the CPU sends to GPU only the initial pose of mesh and the transformation matrix of joints [Dom01]. This animation mechanism reduces the calculation time of the dynamic mesh and relatively increases the burden of joint transformation. In order to demonstrate this, we conducted an experiment on the effect of joints. Joints and polygons are both simplified into 8 levels and finally 64 levels of detail are created. For each detail, 10,000 articulated bodies are cloned and animated. Our experiment shows that the number of joints remarkably affects the overall speed of crowd animation. In comparison with polygon reduction (see Fig. 1), joint elimination appears to be even more effective. Although this skeletal simplification approach cannot surpass the animation speed of impostor technique [ABT00][TLC02][DHO+05], our experiment shows that the speed of the lowest detail (8% detail) is about four times faster than the original animation (100% detail), without any other physical simulation or interpolation, *i.e.*, just for playback.

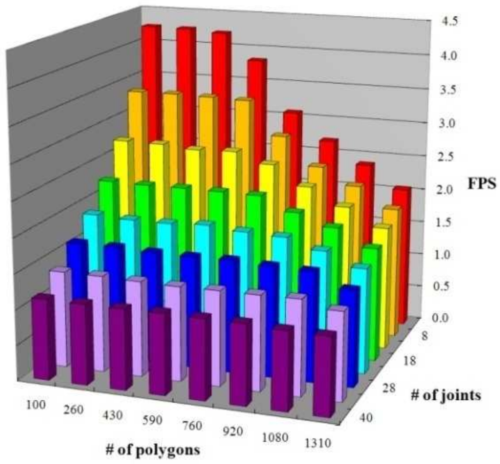

Figure 1. Experiment on 10,000 walking virtual humans (VHs). A VH consists of 40 joints and 1,310 polygons. Joints and polygons are simplified at a rate of 100, 82, 70, 58, 45, 33, 20, and 8%. Each bar represents animation speed of 10,000 VHs by frame per second (FPS) at GeForce 7800 GTX 512 MB and Pentium D 3.0 Ghz.

Our major contributions are as follows:

Automatic level generation: Each joint has different hierarchical depth and rotational variance per motion. If a joint has fast moves, it can be considered that it contains more information than a slow joint. The priority of elimination of joint will be decided by this importance measure. In this paper we present an equation that automatically generates this importance.

Fast preprocessing: From the motion capture system, a number of motions are accumulated. Animators who have access to these captured motions spend lots of time to eliminate joints with low importance and to re-create simplified motion. Our simplification method optimizes simplified motion by excluding mesh parameters in the objective function and by using a fast nonlinear optimization solver.

Preserved motion quality: If the bone length is not consistent during motion, the quality of motion will decrease. Previous works on skeletal simplification [JT05][AOW06] mentioned about this bone length problem. In this approach, we succeed to avoid bone length variation by defining constraints of rigid bone.

From these contributions, our approach not only reduces preprocessing time of optimization, but also considers the quality of simplified motion. Moreover, automatic generation of joint importance shows the practicality and flexibility of this method.

2. RELATED WORKS

There are two approaches on crowd animation for improving the speed: (1) the 'impostor' [ABT00] [TLC02][DHO+05], wherein an image sequence is applied instead of articulated motion, and (2) the 'skeletal simplification' for simulation LoD [CH97] [CIF99][PW99][BK04][RGL05][KRK08] or motion LoD [GJG+03][JT05][AOW06]. Among these works the impostor is more efficient in terms of increasing the speed of crowd. However, this image technique is compromised by the following disadvantages:

Reality: If the camera approaches an animated virtual human (VH) or moves rapidly around the crowd, the realism of the motion is deteriorated.

Memory: To animate a long take or multiple motions, a great deal of memory space is necessary for saving entire image sequences.

Interactivity: In a real-time environment, interaction between the user and VH would not be easy.

Skeletal simplification, meanwhile, can gradually decrease detail of motion and at run time it can adjust motion details by controlling the number of joints. Moreover, the skeletal simplification can overcome these reality, memory, and interactivity problems.

Over the last decade, many researches on skeletal simplification have been presented. [CH97] and [PW99] simplified manually the hierarchy of an articulated figure in order to improve the speed of physical simulation and facilitate convergence. However, these skeletal simplifications are unable to apply in the crowd motion using motion capture data. [GJG+03] proposed a manually constructed level of articulation to improve speed of the real-time networked environment. [BK04] described a method for simulating motion of complex plants. They used a preprocessing method to generate different plant structure, along with a set of simulation LoD. [AW04] proposed a joint posture clustering (JPC) method in order to reduce the number of transformation and improve the speed of animation. However, reducing transformation is not as fast as eliminating joint of motion. [RGL05] presented an adaptive algorithm for computing forward dynamics of articulated bodies using motion error metrics. Their approach simplifies the dynamics of a multi-body system, based on the desired number of degrees of freedom and forces. [JT05] generated different levels of motion based on given animating mesh sequences. The joint structure is reconstructed by clustering rotation of triangles. [AOW06] proposed an optimized motion LoD for real-time crowd animation. In order to generate a simplified motion, they minimized error between the original and the simplified motion by designing a linear system that optimizes skinning matrices by least square approximation (LSA).

In the early period of skeletal simplification, joints were eliminated for the purpose of improving speed of physical simulation (simulation LoD). Recently, however, eliminating joint from a captured motion (motion LoD) became another important issue.

3. MOTION LOD FRAMEWORK

The overall animating pipeline of our approach is depicted in Fig. 2. First, at the preprocessing stage, motion and mesh are divided into a number of details through our motion and geometric LoD [Hop96]. The number of details is given by user and a discrete level of mesh is generated by edge collapsing. Simplified motion is connected to the corresponding mesh level by 'motion mapping'. The background scene is also analyzed to populate simplified bodies into the scene. A depth map is generated from the top orthographic view of the VH's movable region. This map is later used for calculating height and limiting boundary of VH's movement. At the run-time stage, preprocessed results are gathered into the 'simulation' module. During simulation, VH's root position, view frustum culling, and projected size of VH from the camera coordinates are generated and sent to the 'scene generation' via 'LoD control' or directly. Finally, the whole scene is rendered for each frame and camera attributes are modified for the next simulation loop.

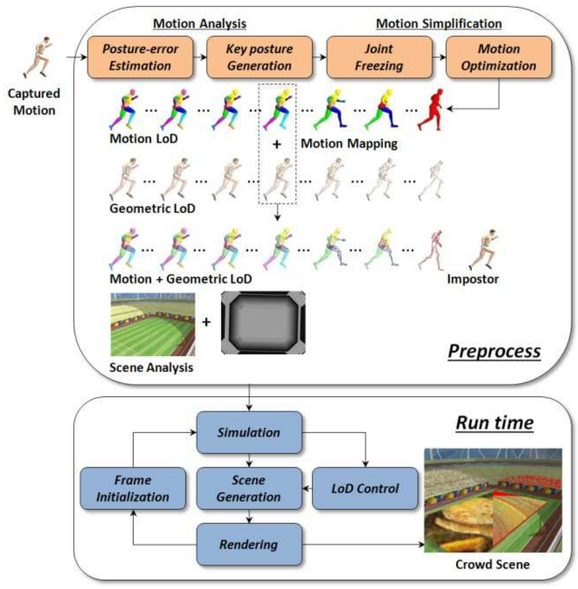

Figure 2. The overall pipeline of motion LoD.

The proposed motion LoD consists of two sub-parts - motion analysis and simplification. Every motion has different properties, thus the way of simplifying motion must be distinguished among each motion. The proposed 'motion analysis' enables us to distinguish motion by generating key posture of motion and priority list of joint elimination. In the 'motion simplification' stage, each motion level is generated. Joints that are selected from the priority list become frozen joints (see Section 5). At the 'joint freezing' stage, frozen joints are applied to the simplified structure of motion. Finally, at the 'motion optimization', our nonlinear optimization algorithm calculates a new simplified motion by minimizing the error between the original and the simplified motion.

4. MOTION ANALYSIS

Before simplifying a skeletal structure of motion, we need to know the elimination priority of joints. This priority list is created by sorting joint importance, which are measured by the sum of posture error. In this section, we propose a basic equation on posture error and show how we extract key posture and derive joint importance from the posture error.

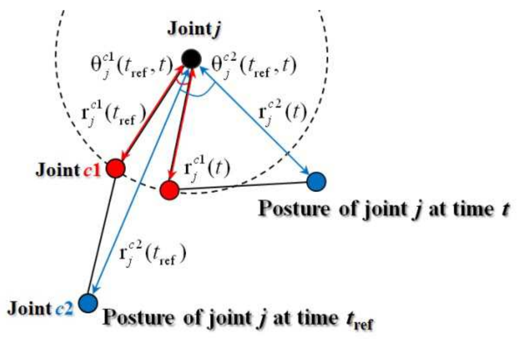
Joint j
$\theta_j^{c1}(t_{\mathrm{ref}}, t)$
$\theta_j^{c2}(t_{\mathrm{ref}}, t)$
$\mathrm{r}_j^{c1}(t_{\mathrm{ref}})$
$\mathrm{r}_j^{c2}(t)$
Joint $c1$
$\mathrm{r}_j^{c1}(t)$
Posture of joint j at time t
$\mathrm{r}_j^{c2}(t_{\mathrm{ref}})$
Joint $c2$
Posture of joint j at time t_{ref}

Figure 3. The evaluation terms of posture error $\mathrm{E}_j(t_{\mathrm{ref}}, t)$. The length $\mathrm{r}_j^c(t)$ is the distance between joint j and its child c at frame t. The radian $\theta_j^c(t_{\mathrm{ref}}, t)$ is the angle of $\mathrm{p}_c(t_{\mathrm{ref}})\mathrm{p}_j\mathrm{p}_c(t)$, where $\mathrm{p}_c(t)$ is the position of joint c at frame t.

4.1. Posture Error of Joint

Our posture error equation considers two important factors - hierarchical depth and rotational variance of joint. For example, if a joint lies near the root of hierarchy, its rotation will propagate to descendents. Therefore, a higher level joint has a higher posture error than a lower level joint. The rotational variance is a more intuitive factor. If a joint rotates in a wide range during motion then the difference will increase. As described in Eq. (1), the posture error $\mathrm{E}_j(t_{\mathrm{ref}}, t)$ is a posture difference of joint j from frame t_{ref} to t.

$$\mathrm{E}_j(t_{\mathrm{ref}}, t) = \frac{\sum_{c=1}^{m}\left\|\left[\mathrm{r}_j^c(t_{\mathrm{ref}})+\mathrm{r}_j^c(t)\right]\theta_j^c(t_{\mathrm{ref}}, t)\right\|}{2\pi \mathrm{r}_{\max}} \tag{1}$$

Basically, the posture error equation is a normalized summation of joint-trajectory distance between t_{ref} and t. Each specific term of equation is depicted in Fig. 3. The constant m is the total number of children of joint j and $2\pi\mathrm{r}_{\max}$ is the normalization value of posture error equation. In our experiment, we set the $r_{\max}$ value as the summation of r^{c}_{root} (c=1, …, m). In this equation, the hierarchical factor is covered by the number of descendants m and the average length of $r^{c}_{j}(t_{\text{ref}})$ and $r^{c}_{j}(t)$. Meanwhile, the rotational factor is covered by $\theta^{c}_{j}(t_{\text{ref}}, t)$. Some previous works proposed a different way to evaluate posture error [LCR+02] [KPS03]. Our posture error equation automatically generates a normalized and weighted error.

4.2. Key Posture of Motion

From the Eq. (1), we create a 2D array of posture errors for each joint, where the row parameter is t_{ref}, column is t and array value is $E_j(t_{\text{ref}}, t)$. The key posture is generated from the minimum sum of the row *i.e.* for each joint j, we select a reference frame t_{ref} and calculate the sum of posture errors on entire frame t ($1 \le t \le n$). The constant n is the number of motion frames. This error summation proceeds for all t_{ref} ($1 \le t_{\text{ref}} \le n$). As defined by Eq. (2), the reference frame t_{ref} with the minimum row sum is set to t^{key}_{j}, which is the key frame of joint j.

$$t_j^{\text{key}} = \arg\min_{t_{\text{ref}}(1 \le t_{\text{ref}} \le n)} \left[\sum_{t=1}^{n} \mathrm{E}_j(t_{\text{ref}}, t)\right] \qquad (2)$$

As the result, each joint contains different key frame t^{key}_{j}. Key posture of motion is generated by applying matrix or quaternion that is defined at frame t^{key}_{j}. Since the key frame of each joint is extracted from the existing motion frames, the key posture doesn't violate human constraint. As depicted in Fig. 4, we applied our key posture algorithm to several motions.

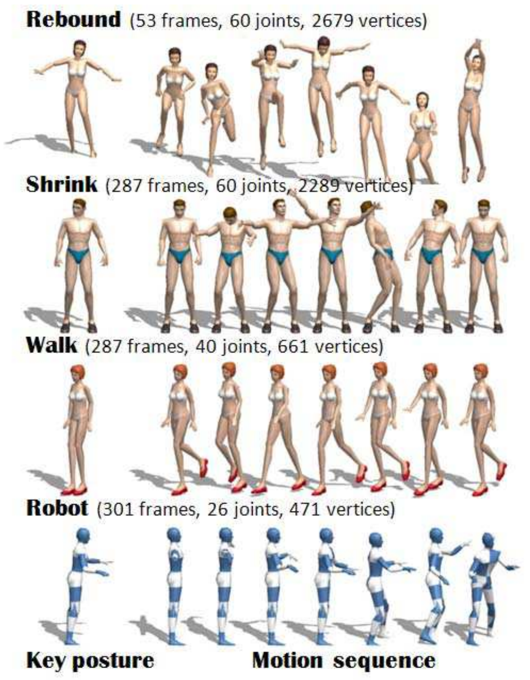
Rebound (53 frames, 60 joints, 2679 vertices)
Shrink (287 frames, 60 joints, 2289 vertices)
Walk (287 frames, 40 joints, 661 vertices)
Robot (301 frames, 26 joints, 471 vertices)
Key posture
Motion sequence

Figure 4. The key posture generation results.

4.3. Joint Importance

Joint importance is a measure of average variance of a joint on the entire motion. From this value, we can generate the priority list for joint elimination. The joint importance ε_j can be obtained from Eq. (2). As defined by Eq. (3), the sum of the row t^{key}_{j} is normalized by the number of frames. A joint with low importance has a higher elimination priority. The value ε_j is used on the motion simplification stage as for creating the priority list for joint elimination.

$$\varepsilon_j = (1/n)\sum_{t=1}^{n} \mathrm{E}_j(t_j^{\text{key}}, t) \qquad (3)$$

5. MOTION SIMPLIFICATION

In this section, we describe how to minimize error between original and simplified motion. The goal is achieved by two stages - joint freezing and motion optimization. In order to construct a priority list of joint elimination and to generate frozen joints, the joint importance ε_j is applied. The basic terms of the proposed objective function are acquired from joint position vectors and frozen joints.

5.1. Frozen Joint

Before removing joint from a motion, we freeze joint (make it rigid) in order to keep useful parameters such as rigid bone length and angle. Frozen joints are selected by ε_j and by the number of joints that will be eliminated during motion simplification. The local transformation of frame t^{key}_{j} which has the minimum error sum over all frames is applied as a frozen joint. For each frozen joint, the bone length and cosine angle constraints can be defined as in Eq. (4).

$$\mathrm{b}_{c,p}(t) = \|\mathbf{x}_c(t) - \mathbf{x}_p(t)\|^2 \qquad (4)$$
$$\theta_j(t) = \frac{[\mathbf{x}_c(t) - \mathbf{x}_j(t)] \cdot [\mathbf{x}_p(t) - \mathbf{x}_j(t)]}{\mathrm{b}_{c,j}(t)\mathrm{b}_{j,p}(t)}$$

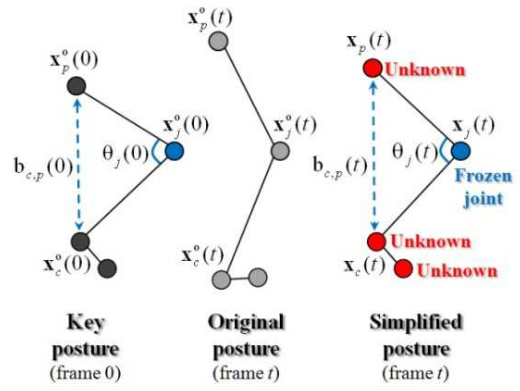

Figure 5. The frozen joint and motion attributes for optimization. A simplified posture is a posture of which we want to minimize the error.

As depicted in Fig. 5, the unknown motion consists of position vectors of joint $\mathbf{x}_j(t)$, functions of bone length $b_{c,p}(t)$, and cosine angle $\theta_j(t)$, where p is parent and c is child of joint j. Here the cosine angle $\theta_j(t)$ is defined from 0 to π. For every frozen joint j, known values $b_{c,p}(0)$ and $\theta_j(0)$ of the key posture (keys are saved into frame zero) are pre-calculated. The rest pose of a VH's mesh is modified by the key posture, including skinning weights re-arrangement on the frozen area. If a vertex is related to a frozen joint, the skinning weights move to its parent joint.

5.2. Motion Optimization

The basic idea of motion optimization is to minimize the sum of difference between the original motion $\mathbf{x}^o_j(t)$ and simplified motion $\mathbf{x}_j(t)$. Due to the frozen joints, the objective function must consider additional hard constraints such as the bone length and joint angle. As was defined by Eq. (4), these functions can be replaced by unknown vectors $\mathbf{x}_j(t)$. Therefore, we formulate the objective function E as a summation of distance E_d, string E_s, and angle E_a error as in Eq. (5). Sequential quadratic programming (SQP) [GMW81][Fle87][Gle98], is applied to solve this problem. Each error term is multiplied by the weighting constants α, β, and γ. In our experiment, we applied 0.2, 0.4 and 0.4 respectively.

$$E = \alpha E_d + \beta E_s + \gamma E_a \qquad (5)$$

Distance error: The square sum of the positional difference between the original joint position $\mathbf{x}^o_j(t)$ and an unknown simplified joint position $\mathbf{x}_j(t)$ is the distance error E_d as in Eq. (6). Constants f and n are the total number of frames and joints, respectively. Root joint (j =1) is excluded in the evaluation, since the root position of the simplified motion is constrained to be the same as the original position.

$$E_d = \sum_{t=1}^{f} \sum_{j=2}^{n} \left\| \mathbf{x}_j(t) - \mathbf{x}_j^o(t) \right\|^2 \qquad (6)$$

String error: The square sum of joint-to-joint length difference between $b_{c,p}(t)$ and $b_{c,p}(0)$ is the string error E_s as in Eq. (7). Function $b_{c,p}(t)$ is the length between joint c and p of the unknown simplified motion at frame t, where p is the first parent of c among joints not to be removed (non-frozen joint). Constant $b_{c,p}(0)$ is the same length at the key posture. Constant m is the total number of non-frozen joints. Root joint (c = 1) is excluded in the evaluation, since its parent joint does not exist.

$$E_s = \sum_{t=1}^{f} \sum_{j=2}^{m} \left\| b_{c,p}(t) - b_{c,p}(0) \right\|^2 \qquad (7)$$

Angle error: The square sum of approximated cosine difference between $\theta_r(t)$ and $\theta_r(0)$ is the angle error E_a, as in Eq. (8). Function $\theta_r(t)$ is the cosine value of joint r at frame t, where r is one of the frozen joints. Constant $\theta_r(0)$ is the same cosine value of the angle of joint r at the key posture. Constant n-m is the total number of frozen joints. In the case of non-frozen joints, the cosine value varies through the frame, and therefore is not considered here.

$$E_a = \sum_{t=1}^{f} \sum_{r=1}^{n-m} \| \theta_r(t) - \theta_r(0) \|^2 \qquad (8)$$

For the initial values, the original motion's joint position is used. The local transformation value of each joint is recovered from the optimization result $\mathbf{x}_j(t)$ and the hierarchical information of each joint. Our approach optimizes simplified motion from the motion data itself. The frozen joints are removed after fitting motion data into the meshed structure. Fig. 6 shows each level of simplified motion.

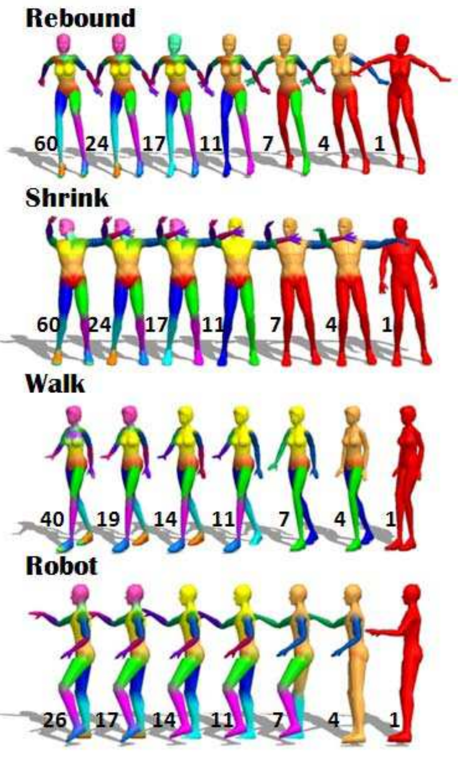

Figure 6. Motion LoD results. The number of joints is indicated on the left, for each LoD posture. Each color corresponds to a joint.

6. EXPERIMENTAL RESULTS

In this section, we describe our experiments that address the advantage of our proposed method. We have conducted four experiments - preprocessing time, motion quality, memory, and animation speed.

6.1. Preprocessing Time

The main advantage of our approach is that we use simple parameters in the optimization process. Since the previous works [JT05][RGL05][AOW06] include complex parameters such as velocity, acceleration, position, and normal of mesh, they give a burden of preprocessing time. As was described in Table 1, we analyzed the preprocessing time of four different methods. The number of joints, frames, and vertices (N_j, N_f, N_v) are described for each experiment. Each preprocessing time was measured by minutes (t_m). Motion analysis contributes to a fast convergence.

Methods	t_m	N_j	N_f	N_v
SMA	7.2	22	400	3030
LSA	29.4	11	287	2239
SQP	3.5	24	331	2679
ASQP	0.8	24	331	2679

Table 1. A comparison of preprocessing times; SMA [JT05]; LSA [AOW06]; SQP: Our method; ASQP: Our method with motion analysis

6.2. Motion Quality

By formulating constraints of bone length and angle, we succeed to attain reasonable quality of motion. In order to show this, we conducted an experiment on the average errors and variations of the bone length (see Fig. 7). Each motion is simplified into 8 levels of detail and *the sum of bone errors* is calculated for each posture (*i.e.* the sum of bone length differences in a frame). The blue and green bar is the average of *the sum of bone errors* over entire frames and levels of motion. In addition to the average bone error, we generated the variation of bone length as well. For each bone, the standard deviation of bone length is calculated over all frames of motion. For each level, we added standard deviations of all bones. The red and violet bar is the average standard deviation over all levels of motion. As the result, our approach shows better stability compared to the other approach. Moreover, the length variation is less than 3%, which is not perceivable with full attention [HRP04]. The enhancement of our approach is shown in Fig. 8.

6.3. Memory

A previous work on impostor [DHO+05] mentioned that 7 MB of memory are required for sampling a single frame of motion. Considering a 287-frame shrinking motion, more than 2 GB will be required. However, our approach needs only 10.3 MB for ten levels of motion (animation and geometry: 9.94 MB, texture: 0.36 MB). According to the experiment of polypostor [KDC+08], our approach also gives better efficiency. The memory cost will decrease more, if a temporal compression [Ari06] or a geometric LoD [PHB07] are applied into our framework.

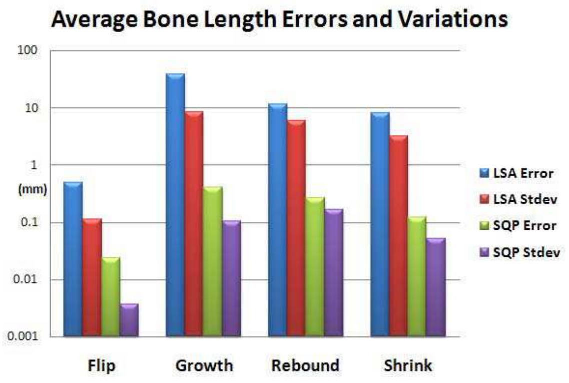

Figure 7. The comparison of average bone length errors and variations in logarithmic scale of mm.

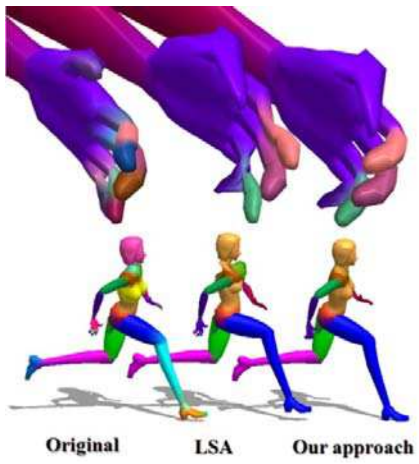

Figure 8. Enhancement in terms of motion quality; Top: Growth motion; Bottom: Rebound motion

6.4. Animation Speed

To conduct an experiment on speed, 5,000 VHs with 200,000 joints are populated in the scene. Motion and mesh are simplified into ten levels. As described in Table 2, average animation speed is measured for four different types of animation with the same navigation path. During the navigation, our approach improves the speed by minimum two to maximum five times faster than the original. Moreover, owing to the GPU skinning technique, motion LoD is even faster than geometric LoD. We used the same simplification rate for both GLoD [Hop96][OZS+03] and MLoD. The hardware environment is GeForce 7800 GTX 512 MB and Pentium D 3.0 Ghz.

	Original	GLoD	MLoD	G+MLoD
Nav1	6.78	12.44	17.83	21.73
Nav2	12.49	20.62	26.44	30.43

Table 2. Average animation speed in FPS for two navigation paths (Nav1, Nav2); GLoD: Navigation with Geometric LoD; MLoD: with Motion LoD; G+MLoD: with Geometric and Motion LoD

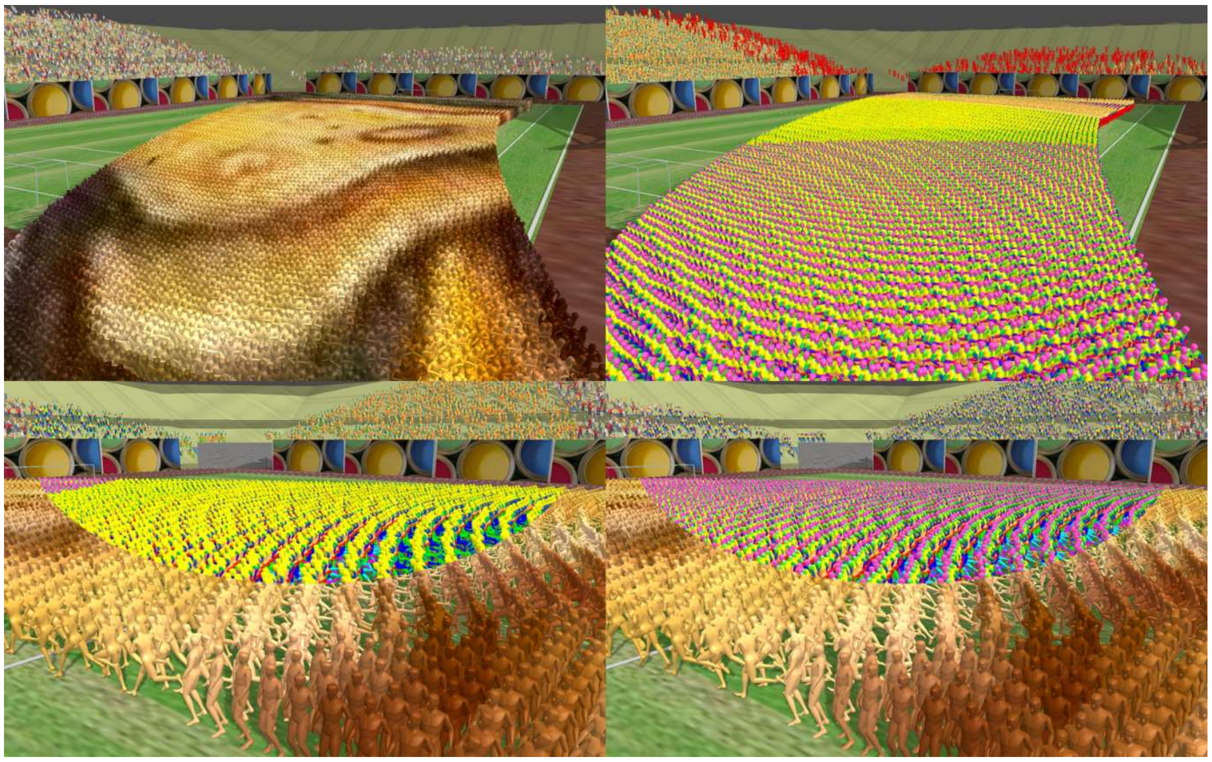
Figure 9. The Stadium: A massive crowd scene; Top left: A scene populated with 15,880 VHs, 805,189 joints, and 56,775,042 polygons; Top right: Top left scene with colored joints; Bottom left: Top right scene shown from the other camera view (motion LoD is applied from the top scene's camera view); Bottom right: Bottom left scene without motion LoD (for comparing visual quality of our motion LoD); The average frame rate is 5.23 FPS (original scene is 1.21 FPS)

Finally, as was depicted in Fig. 9, we have populated 15,880 VHs in a stadium environment to show the efficiency of our motion LoD approach. A total six different VH models and six different motions with about 1,240 frames were preprocessed by geometric LoD and motion LoD, respectively. The size of memory required for simplified VH was 41.7 MB (animation and geometry: 39.88 MB, texture: 1.82 MB). We were able to animate 805,189 joints and 56,775,042 polygons at 5.23 FPS with GeForce 8800 GTX 768 MB and Core2Duo 2.4 Ghz.

7. CONCLUSION

We have presented a new LoD framework that improves the performance of a real-time crowd environment. We automate the whole process and reduce the preprocessing time for the practical use. Moreover, the bone length and angle preservation improves the animation quality of the simplified motion. As the result, we verified that our approach can be adopted in a crowd animation framework. However, the proposed motion LoD doesn't consider the end effector or foot plant of the simplified motion. We assumed that the end effector is not a serious problem, since the detail of low level motion is indistinguishable in the scene. Our approach is not suitable for physics based animation, since the motion analysis and simplification is running on the preprocessing time. The proposed motion LoD is suitable for a real-time environment with captured motion. The error of LoD transition is not considered in our optimization. We tried to minimize the artifact by controlling LoD with projected pixel size of VH.

For improving the quality of optimization, some future works remain. The proposed optimization focuses on a fast preprocessing with affordable accuracy of the simplified result. Therefore, other important issues such as motion smoothness are not considered in the proposed objective function. By resolving the temporal coherence problem, the quality of simplified motion will surely be improved. Another work that should be considered is to derive multiple key postures, since the error of simplified motion strictly depends on the key posture. The proposed motion LoD technique is expected to be more useful in the applications of highly crowded environments such as an urban simulation and games, since a great number of virtual humans are often occluded or appear tiny in the entire scene. It can readily be surmised that the number of virtual humans and the complexity of a skeleton would greatly increase in the near future. Motion LoD will be more challenging than ever before.

8. ACKNOWLEDGMENTS

This research was supported in part by a grant from European Union (EU) FP7 funded IP project, CYBEREMOTIONS (Grant: 231323), from Swiss National Research Foundation (FNRS) funded project, AERIALCROWDS (Grant: CRSI20-122696), and from KAIST Institute for Entertainment Engineering (KIEE). The authors wish to thank Dr. Stéphane Gobron, Quentin Silvestre, Clément Marx, and Olivier Renault for the discussions and supports.

9. REFERENCES

[ABT00] Aubel A., Boulic R., and Thalmann D., *Real-time display of virtual humans: Levels of detail and impostors*, IEEE Transactions on Circuits and Systems for Video Technology, 10(2): 207–217, 2000.

[AOW06] Ahn J., Oh S., and Wohn K., *Optimized motion simplification for crowd animation*, Computer Animation and Virtual Worlds, 17(3–4): 155–165, 2006.

[Ari06] Arikan O., *Compression of motion capture databases*, ACM Transactions on Graphics, 25(3): 890–897, 2006.

[AW04] Ahn J., and Wohn K., *Motion level-of-detail: A simplification method on crowd scene*, Computer Animation and Social Agents, pp. 129–137, 2004.

[BK04] Beaudoin J., and Keyser J., *Simulation levels of detail for plant motion*, Symposium on Computer Animation, pp. 297–304, 2004.

[CH97] Carlson D. A., and Hodgins J. K., *Simulation levels of detail for real-time animation*, Graphics Interface, pp. 1–8, 1997.

[CIF99] Chenney S., Ichnowski J., and Forsyth D., *Dynamics modeling and culling*, IEEE Computer Graphics and Applications, 19(2): 79–87, 1999.

[DHO+05] Dobbyn S., Hamill J., O'Conor K., and O'Sullivan C., *Geopostors: a real-time geometry / impostor crowd rendering system*, Symposium on Interactive 3D graphics and games, pp. 95–102, 2005.

[Dom01] Dominé S., *Mesh skinning*, NVIDIA, 2001. (http://developer.nvidia.com/object/skinning.html)

[Fle87] Fletcher R., *Practical methods of optimization (2nd edition)*, Wiley-Interscience, 1987.

[GJG+03] Giacomo T. D., Joslin C., Garchery S., and Magnenat-Thalmann N., *Adaptation of facial and body animation for mpeg-based architectures*, Cyberworlds, p. 221, 2003.

[Gle98] Gleicher M., *Retargetting motion to new characters*, SIGGRAPH, pp. 33–42, 1998.

[GMW81] Gill P., Murray W., and Wright M., *Practical optimization*, Academic Press, 1981.

[Hop96] Hoppe H., *Progressive meshes*, SIGGRAPH, pp. 99–108, 1996.

[HRP04] Harrison J., Rensink R. A., and van de Panne M., *Obscuring length changes during animated motion*, ACM Transactions on Graphics, 23(3): 569–573, 2004.

[JT05] James D. L., and Twigg C. D., *Skinning mesh animations*, ACM Transactions on Graphics, 24(3): 399–407, 2005.

[KDC+08] Kavan L., Dobbyn S., Collins S., Zara J., and O'Sullivan C., *Polypostors: 2D polygonal impostors for 3D crowds*, Symposium on Interactive 3D graphics and games, pp. 149–155, 2008.

[KPS03] hoon Kim T., Park S. I., and Shin S. Y., *Rhythmic motion synthesis based on motion-beat analysis*, ACM Transactions on Graphics, 22(3): 392–401, 2003.

[KRK08] Kim S., Redon S., and Kim Y. J., *View-dependent dynamics of articulated bodies*, Computer Animation and Virtual Worlds, 19(3–4): 223–233, 2008.

[LCR+02] Lee J., Chai J., Reitsma P. S. A., Hodgins J. K., and Pollard N. S., *Interactive control of avatars animated with human motion data*, SIGGRAPH, pp. 491–500, 2002.

[MT01] Musse S. R., and Thalmann D., *Hierarchical model for real time simulation of virtual human crowds*, IEEE Transactions on Visualization and Computer Graphics, 7(2): 152–164, 2001.

[OZS+03] Oliveira J., Zhang D., Spanlang B., and Buxton B., *Animating Scanned Human Models*, Journal of WSCG, 11(2): 362–369, 2003.

[PAB07] Pelechano N., Allbeck J. M., and Badler N. I., *Controlling individual agents in high-density crowd simulation*, Symposium on Computer Animation, pp. 99–108, 2007.

[PHB07] Payan F., Hahmann S., and Bonneau G.-P., *Deforming surface simplification based on dynamic geometry sampling*, IEEE International Conference on Shape Modeling and Applications, pp.71–80, 2007.

[PW99] Popović Z., and Witkin A., *Physically based motion transformation*, SIGGRAPH, pp. 11–20, 1999.

[Rey87] Reynolds C. W., *Flocks, herds and schools: A distributed behavioral model*, SIGGRAPH, pp. 25–34, 1987.

[RGL05] Redon S., Galoppo N., and Lin M. C., *Adaptive dynamics of articulated bodies*, ACM Transactions on Graphics, 24(3): 936–945, 2005.

[ST05] Shao W., Terzopoulos D., *Autonomous pedestrians*, Symposium on Computer Animation, pp. 19–28, 2005.

[TCP06] Treuille A., Cooper S., and Popović Z., *Continuum crowds*, ACM Transactions on Graphics, 25(3): 1160–1168, 2006.

[TLC02] Tecchia F., Loscos C., and Chrysanthou Y., *Image based crowd rendering*, IEEE Computer Graphics and Applications, 22(2): 36–43, 2002.

[TT94] Tu X., and Terzopoulos D., *Artificial fishes: physics, locomotion, perception, behavior*, SIGGRAPH, pp.43–50, 1994.

[YMP+09] Yersin B., Maïm J., Pettré J., and Thalmann D., *Crowd patches: populating large-scale virtual environments for real-time applications*, Symposium on Interactive 3D graphics and games, pp. 207–214, 2009.