View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Infoscience - École polytechnique fédérale de Lausanne

Motion Level-of-Detail: A Simplification Method on Crowd Scene

Junghyun Ahn
VR lab, EECS, KAIST
ChocChoggi@vr.kaist.ac.kr
http://vr.kaist.ac.kr/~zhaoyue

Kwangyun Wohn
VR lab, EECS, KAIST
wohn@vr.kaist.ac.kr
http://vr.kaist.ac.kr

Abstract

Recent technological improvement in character animation has increased the number of characters that can appear in a virtual scene. Besides, skeletal and mesh structures are expected to be more complex in the future. Therefore, simulating massive characters' joints in a real-time crowd environment without any preprocessing is unaffordable. We propose a preprocessing method called 'motion level-of-detail' to overcome this limitation. Our 'motion level-of-detail' framework not only minimizes the simulation cost of the joints, but also maintains the similarity between the original and the simplified motion. 'Joint posture clustering (JPC)', which is the skeletal simplification method of our framework, reduces skeletal node by the clusters of similar postures. A cluster is a set of continuous frames, where each frame has similar posture. Because our approach depends on motion trajectory, simplified result preserves the quality of the motion. We also applied a geometric simplification on deformable character mesh, to increase performance. Our approach was particularly useful for the complex skeletal motions that have a monotonous trajectory.

Keywords: motion level-of-detail, clustering, crowd animation, character animation, mesh simplification

1. Introduction

In the game and entertainment industry, a number of researchers are now trying to animate huge crowd scenes. As a result, the simplification of a character's motion is now a challenge for character animation. Non real-time application such as movie or animation, computing time is not critical. However, a character in the crowd scene does not take much space on a rendered image. Therefore, simulating detail motion on each character is a waste of computing resource. Besides, in a real-time application such as game or NVR environment, adjusting computation vs. animation quality is as much important as well known computation vs. image quality.

In this paper, we present the 'motion level-of-detail' framework which is an effective preprocessing method in the real-time crowd environment. We are subjected to motion captured data, because game and movie industry also prefer to utilize motion capture rather than physical simulation. The 'motion level-of-detail' framework was designed similarly to the common real-time virtual environment. As shown in figure 1, selected motion from the motion database is mapped to the selected character from the character database. After the mapping process, character is transformed to the virtual world. However, a lot of joint transformation is required on each process and generating motion without preprocessing is inefficient. Therefore, to enhance the performance of our framework, we resolved the problem from the perspective of simulation (skeletal simplification) and rendering (geometric simplification).

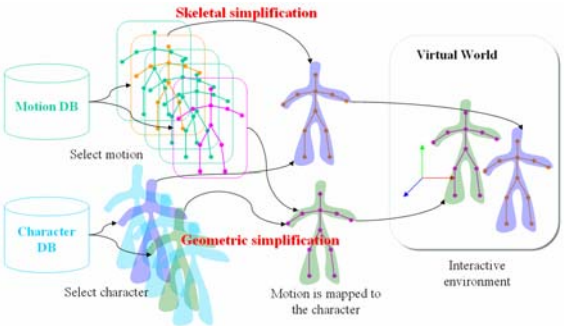


Figure 1 : Motion level-of-detail framework

Skeletal simplification: Every motion includes the hierarchy of articulated figure. This skeletal structure can be defined as a tree, whose node's (joint) attributes (position, rotation)

change every frame. By simplifying the skeleton, we can reduce the cost of transforming joints, which is serious on crowd scenes. Moreover, in the skinning process we can reduce the overheads for vertex transformation. Previous work has introduced a simplification method of articulated figure [1][2][20]. But these works need manual preprocess and cannot be applied to crowd motion. We describe these works in detail in the next section.

Geometric simplification: Prior researchers on geometric simplification focused on a rigid-body mesh structure [3][4][5] to reduce rendering cost. However, character mesh dynamically deforms every frame. Because of that, character mesh must be localized by each body parts. To improve the rendering performance, previous research has substituted character geometry for an image [6][7][21]. We expect a tremendous improvement on performance, but the motion is not realistic and has many limitations on the camera work.

Our 'motion level-of-detail' framework applied these skeletal (JPC) and geometric simplification. Motion dependent method is required to automate skeletal simplification in any kind of motions. Therefore, we first analyzed the trajectories of each joint. By plotting each trajectory on the unit sphere, we can separate the clusters of postures. Correct clusters of postures are evaluated by an error metric between two postures. The error values are applied to our clustering table. Optimal posture clusters are classified by the error threshold and we applied the level-of-detail at the run-time. A joint is dynamically simplified using clustered information. Every clustered area inside a joint trajectory keeps the key posture. At the run-time, our system transforms the key posture and current joint's information is updated to his parent, until current clustered area of joint is finished.

JPC method automatically generates multi-resolution motion and increase performance without loss of animation quality. However, variable motion, which postures of joints change quickly are not much effective, because of hierarchy reconstruction on every cluster. We also implemented a simple mesh simplification method to enhance performance. Our method is useful in a real-time interactive crowd environment.

After a reviewing related works in section 2, we analyze bottleneck of motion generation in section 3. In section 4, we describe the joint posture clustering (JPC) method. In section 5, we show how we applied geometric simplification to a character mesh structure. Finally, in section 6, we show the results of our implementation and experiments.

2. Related works

In this section we describe the research on crowd animation. Crowd animation falls into three major subjects: First for an ideal system of crowd animation, the reality of the generated scene must be satisfied; second, interactivity is important because the user or animator of a crowd system must be able to access the individual characters of a scene; and finally, enhancing the overall performance of the system is important for a massive crowd scene.

Scene reality: To represent a realistic crowd scene in the virtual world, crowd modeling is required to control an individual character's motion [8][9]. Analyzing a human-like crowd flow is also important. Some researchers analyzed the crowd flow in an emergency situation [10][11]. Another major factor for maintaining reality is the set of behavioral rules for each individual. Behavioral rules are well defined for the fishes in an aquarium [12]. The behavior of the fish is simulated by the current state of the virtual environment. The behavior of human crowd has been similarly defined [13][14]. Collision detection of individuals in a crowd [15] and motion transition [16] are also important behavioral factors of the scene realism.

Interactivity: The behavioral rules defined for each individual are inadequate for an animator who designs crowd scene. The animator sometimes needs interaction to convey the intended behavior of each character. For this type of situation, some of the early research on the interaction and scripting of crowds is a good reference [17] [18].

Performance: Simulating complex and massive crowds every frame is time consuming. The problem is more serious when simulating many characters at the same time. The research on improving performance of a crowd scene covers two major topics.

The first major topic is the simplification of the skeletal structure for a rapid physical simulation. A manually simplified hierarchy

for motion blending has presented to enhance performance of physical simulation and facilitate convergence of the optimization [1]. Three basic principles DOF removal, node sub-tree removal and symmetric movement removal are introduced in this work. Another work has introduced 'full physics – kinematics – center of mass' model of a simple articulated object to simplify physical simulation [2]. However, these skeletal simplifications are unable to apply in the crowd motion using motion capture data. We are also able to reference recent work that has focused on real-time network environment [20].

The other major topic is the application of an image-based technique to each character [6][7][21]. They sampled image sequence of character's motion from each direction and used an imposter or billboard technique to enhance performance. Although we can estimate a good performance using image-based techniques, the result of the sampled motion is not realistic and the camera work has many limitations. To overcome these limitations, we propose a new simplification framework in a real-time crowd environment.

3. Motion bottleneck

Rendering bottleneck by processing a highly detailed geometric model is well known in the computer graphics field. However, a bottleneck that occurs by joint processing is not a common knowledge. In this section, we define the process of motion generation and analyze the performance by simulating joints.

3.1 Motion process

The process of generating a motion to a character is shown in figure 2. During the preprocessing, current animation system aligns the selected motion / character structure and calculates difference matrix between them. At the run-time, the system transforms the joint posture from the motion database and applies the difference matrix. In the skinning process, the vertices and normals of the character mesh are deformed by the current generated joint posture. Finally, the system transforms entire posture result to the virtual scene. This sequence can be defined as the 'scene generation' of character animation. A motion bottleneck occurs in that process.

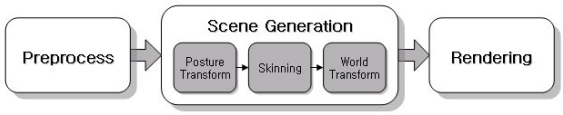


Figure 2: Motion process

If the processing time for multiplying the vertex on the transformation matrix is *p*, the cost of multiplying the matrix is *4p*. The total complexity of a joint can be defined as *(8L+2N)p*, where *L* is the number of joints from the current node to the leaf node, and *N* is the number of mesh vertices connected to the current body segment.

To analyze the motion bottleneck in a crowd scene, we conducted an experiment in which we animated 1000 characters. Each character consists of 946 polygons and 50 rigid joints. The result of the experiment in figure 3 shows that the motion bottleneck is challenging, especially for interactive crowd environment.

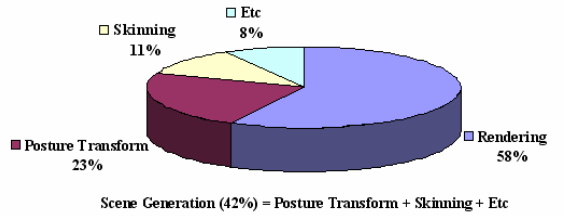


Figure 3: Computing time ratio on crowd scene

3.2 Defining motion

We defined motion as a tree structure in which the head, hands and feet are leaf nodes and the pelvis is the root. Motion is therefore assumed to be a general tree definition such as T(N, E). The joint values such as position and orientation are assumed to be node attributes and the joint link forms a rigid segment. We define motion M on time t as follows:

$$M(t) = M(J(t), S(t))$$

$$J(t) = \{j_0(t), j_1(t), j_2(t), ..., j_j(t), ..., j_n(t)\}$$

$$j_j(t) = \begin{cases} \{p_j(t), q_j(t)\} & \text{if j = root joint} \\ \{NIL, NIL\} & \text{if k = leaf joint} \\ \{NIL, q_j(t)\} & \text{otherwise} \end{cases}$$

$$S(t) = \{s_1(t), s_2(t), ..., s_j(t), ..., s_n(t)\}$$

$$s_j(t) = \{j_j(t),\ j_{jp}(t), l_{j,p}(t)\}$$

Where *J(t)* is the set of joint nodes, *S(t)* is the set of segment edges at time *t*, and $j_j(t)$ is the *j-th* joint at time *t*. Only the root joint has position values, because joint representation is based on a local coordinate. Because the leaf

nodes are end-effectors, the position and orientation values on the leaf are unavailable. Every joint has a parent, except for the root. Therefore, $j=0$ is excluded from $S(t)$. The parameter $s_j(t)$ is the link between the *j-th* joint and the *j-th* parent with its length $l_{j,p}(t)$.

4. Joint posture clustering (JPC)

The rotational effect of a local joint is updated every frame even though the difference is insignificant. By plotting the joint trajectory on a unit sphere, we found some groups of frames that needed no transformation.

We grouped similar postures by analyzing the entire motion trajectory. By this method, n frames of motion were reduced to m groups of posture ($m<n$). To minimize the error cost of the motion difference between the original motion $M(t)$ and the simplified motion $M'(t)$, the original motion became a superset of the simplified motion ($M(t) \supset M'(t)$). The simplified joint maintains the key posture of the current cluster and sends its own information to the parent joint.

4.1 Error evaluation

To generate optimal clusters, we defined the error cost between postures. The difference between two postures at joint j is defined as equation (1):

$$E_j(t,t_{ref}) = E_{pos,j}(t,t_{ref}) + \alpha E_{ori,j}(t,t_{ref}) \quad (1)$$

The error is the sum of the position and the orientation error. The time t is the current frame and t_{ref} is the estimated key frame. The constant α is the weighting value of the orientation difference, and the weighting value is set to zero when joint j has no twist rotation. By using this measurement we clustered each group of postures. The position error was generated by the local joint trajectory and the angle difference between the two postures. The orientation error is defined as the logarithm of the quaternion. Position and orientation error are formulated by measurement values as shown in figure 4.

Local joint trajectory: If the base pose vector from joint j to its child is v_j, and the orientation of the joint at time t is $q_j(t)$, the trajectory $v_j'(t)$ of the local joints at time t is expressed as equation (2):

$$v_j'(t) = (q_j(t) \cdot v_j \cdot q_j(t)^{-1}) / \| l_{j,c}(t) \| \quad (2)$$

The trajectory information $v_j'(t)$ is the vector position plotted on the unit sphere. The scalar value $l_{j,c}(t)$ is the segment length between joint j and its child joint c.

Angle difference: Using the equation (2), we can simply calculate the difference angle between t and t_{ref} as follows:

$$\theta_j(t,t_{ref}) = \arccos\left(\frac{v_j'(t) \cdot v_j'(t_{ref})}{\| l_{j,c}(t) \|^2}\right) \quad (3)$$

Weighting factor: A joint with many children, which has a higher weighting value than a joint near a leaf, propagates a large error cost. The parameter $r_j(t)$ is the weighting factor that is applied to the position difference. It is defined as follows:

$$r_j(t) = \sum_{c}^{leaf} l_{j,c}(t) \quad (4)$$

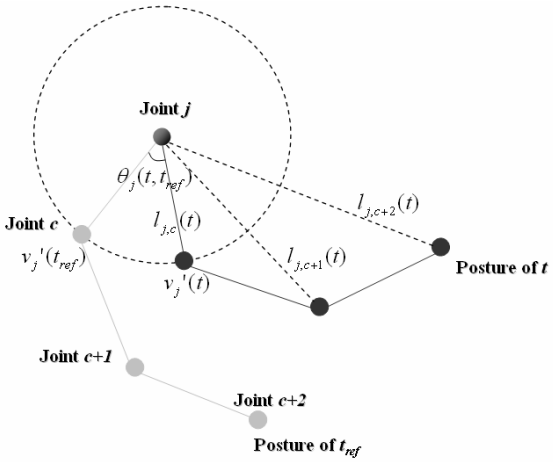
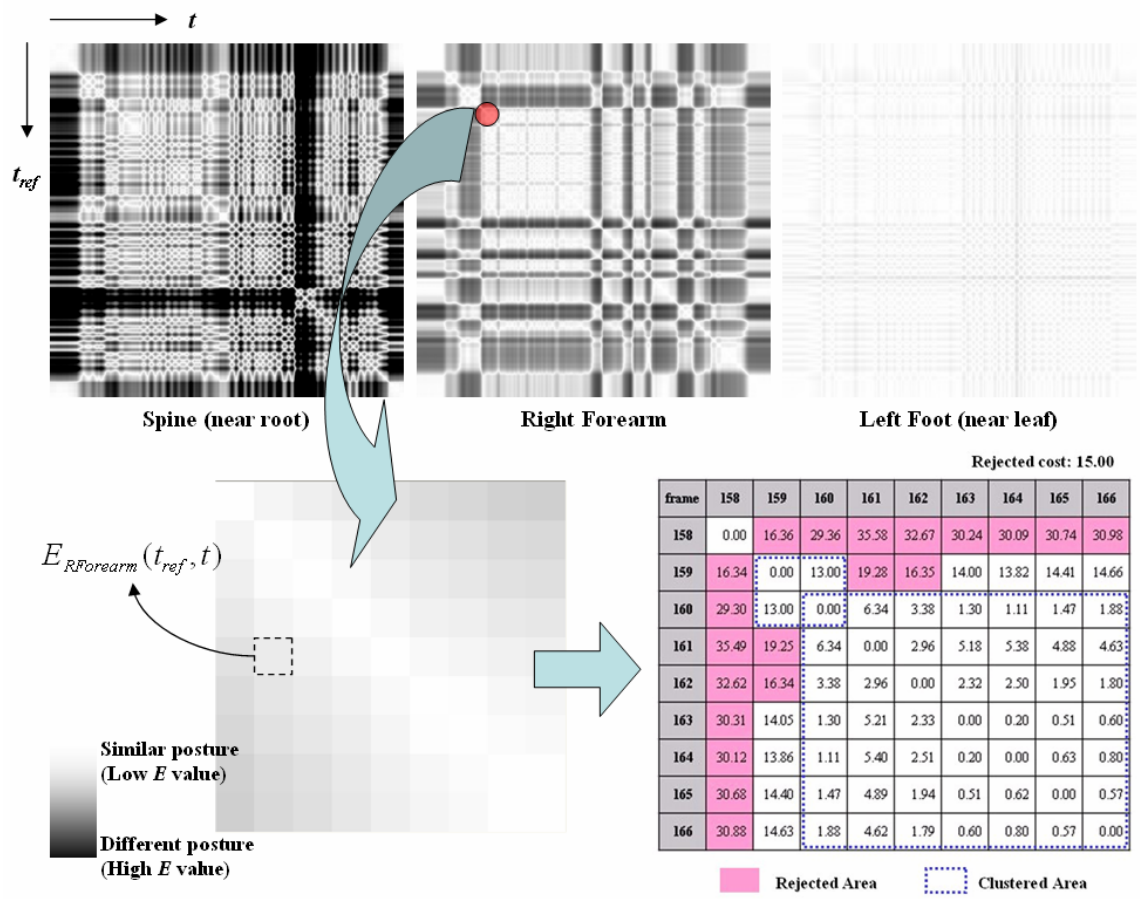


Figure 4: Measurement values

Position error: The positional error cost between the original and the clustered postures at joint j, which means the position difference by simplifying t posture to t_{ref}, is defined as equation (5):

$$E_{pos,j}(t,t_{ref}) = \| r_j(t) \cdot \theta_j(t,t_{ref}) \| \quad (5)$$



frame	158	159	160	161	162	163	164	165	166
158	0.00	16.36	29.36	35.58	32.67	30.24	30.09	30.74	30.98
159	16.34	0.00	13.00	19.28	16.35	14.00	13.82	14.41	14.66
160	29.30	13.00	0.00	6.34	3.38	1.30	1.11	1.47	1.88
161	35.49	19.25	6.34	0.00	2.96	5.18	5.38	4.88	4.63
162	32.62	16.34	3.38	2.96	0.00	2.32	2.50	1.95	1.80
163	30.31	14.05	1.30	5.21	2.33	0.00	0.20	0.51	0.60
164	30.12	13.86	1.11	5.40	2.51	0.20	0.00	0.63	0.80
165	30.68	14.40	1.47	4.89	1.94	0.51	0.62	0.00	0.57
166	30.88	14.63	1.88	4.62	1.79	0.60	0.80	0.57	0.00

Figure 5: Optimal cluster generation by using clustering table (joint difference map)

Orientation error: Minimizing an error with only the trajectory of joint can cause a loss of the twisted rotation on each joint. We applied a rotational measurement of joint j using the logarithm of quaternion [16][19], which means the orientation difference by simplifying t posture to t_{ref}. The equation is expressed as follows:

$$E_{ori,j}(t, t_{ref}) = 2 \| \log(q_j(t)^{-1} \cdot q_j(t_{ref})) \| \qquad (6)$$

4.2 Clustering table

To generate the optimum cluster that does not exceed the error threshold, we used n x n difference map (clustering table) of joint postures, where n is the total number of frames. The table forms a matrix of the error cost between two postures in a motion sequence. The elements of the table are $E_j(t, t_{ref})$, and if $t = t_{ref}$, the value of the element is zero, because the same postures have no difference. A cluster is selected when the top-left coordinate (t_{ref}, t) has the same value and the size of the mask is m x m ($m \leq n$). At least one row inside the mask must be lower than the error threshold. Figure 5 shows the process of generating optimal cluster from the joint difference map.

Every diagonal value is 0 and the map is almost symmetric. Difference between symmetric pairs are occurred by the weighting factor $r_j(t)$. We used only the upper triangle, since the table is almost symmetric. The lower triangle was used only for the process of retrieving the key posture. The optimum set of the clusters are derived by the following algorithm:

STEP 1: Generate available range on each row
LOOP START
    STEP 2: Search MAX range
    STEP 3: Set range to final cluster list
    STEP 4: Delete current MAX range
    STEP 5: Refresh remaining range
LOOP END
STEP 6: Retrieve key posture on each cluster

In step 1, the range of each row is generated from the diagonal element (upper triangle element) until the element of the table is higher than the error threshold. Every top-left coordinate of the mask must satisfy (t_{ref}, t).

In step 5, we must refresh the remaining range because an intersected area exists between removed range and remaining range. If the size of the refreshed range is 1, the range is also removed. The cluster length must be greater than 2.

4.3 Key posture

After the final cluster list was generated, we extracted the key posture of the clustered area. The entire cluster maintains the key posture in the motion sequence. The key posture is the minimum sum of the elements in the same row. The key posture is generated as equation (7), where *m* is the range of the optimal cluster.

$$\min_{t_{ref}} (\sum_{t}^{m} E_j(t, t_{ref})) \qquad (7)$$

As the result, the minimum reference frame 165 means that every frame from 160 until 166 is maintained to posture 165. Maintaining posture 165 is the minimum cost of error compared with other reference frames.

Figure 6 shows the trajectory of a joint after the JPC process. As the error threshold increases, the number of clusters decreases.

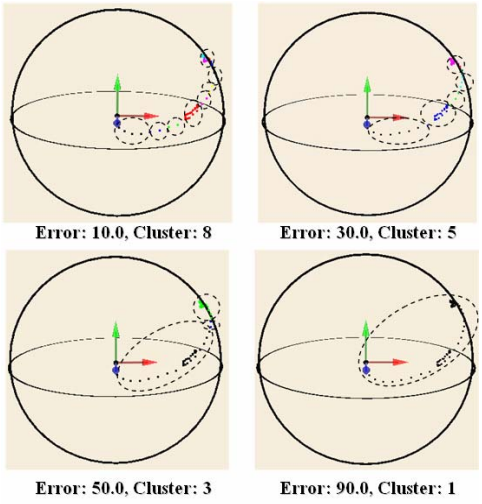


Figure 6: Clustered trajectory

If the motion database consists of long size motions, we need to reduce the problem, because the problem complexity is $O(n^2)$. Before constructing the table, we applied the angle velocity of the motion to separate the range of the table. As we described in equation (6), the position error is in r, Θ form. Therefore the reduction measurement can be defined as $E_{pos,j}(t, t\text{-}1)$.

5. Geometric Simplification

Motion reconstructed by JPC method should be more improved by combining a common geometric simplification. In character animation, however, the mesh deforms dynamically on each frame. Every vertices and weighting values are linked to the joint segment and thirty percent of full-body mesh vertices are shared by more than two joint segments. Shared vertices are deformed from the local coordinate by weighting values.

We divided the vertex list by each joint and applied progressive mesh [3] to each joint segment. Before applying the mesh simplification, we divided the local vertex list into the shared vertex area and the non-shared vertex area. The weighting value of the shared vertex is ($0<w<1$), and of the non-shared vertex, ($w=1$).

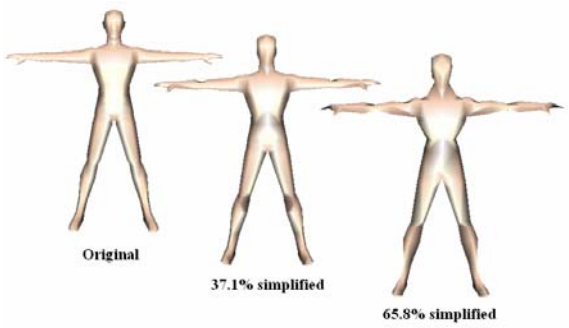


Figure 7: Mesh simplification result

6. Experiments and Results

6.1 Experiment result

Skeletal simplification: JPC method extracts maximum size of the cluster inside error bound. Therefore, the original frame posture and the key posture of the cluster are different. As we shown in figure 8, 9, 10, low error motion is more similar to the original motion. We did experiments about motion with small number of joints (figure 8) vs. large number of joints (figure 9) and variable motion (figure 9) vs. monotonous motion (figure 10).

Table 1 shows average simplified joints rate (%) per frame. The simplification rate of a motion with large number of joints is higher than a motion with small number of joints, because joint density of the former is higher than the latter motion. The joint trajectory of a monotonous motion can extract larger size of cluster than a variable motion, because the

monotonous motion has smaller trajectory area than the variable motion.

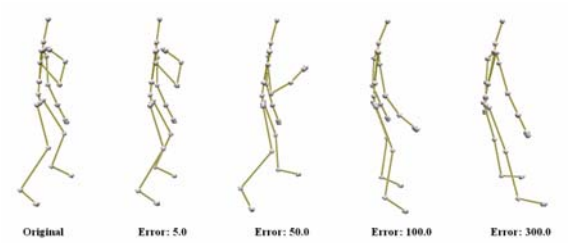


Figure 8: Dance (28 joints, variable motion)

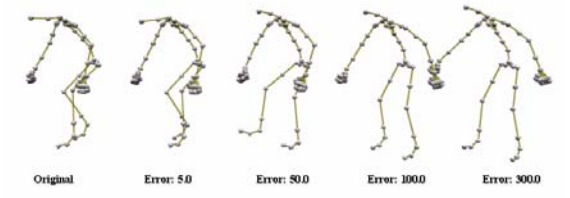


Figure 9: Push (80 joints, variable motion)

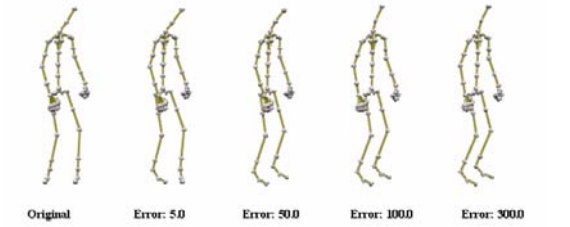


Figure 10: Look (80 joints, monotonous motion)

Motion	Original	E=5.0	E=50.0	E=100.0	E=300.0
Dance	0.0	32.1	70.8	94.1	95.8
Push	0.0	54.9	84.3	94.3	96.4
Look	0.0	73.4	92.6	97.8	98.2

Table 1: Average simplified joints (%)

Overall simulation: We implemented a real-time crowd environment with 'motion level-of-detail' framework to compare the processing time of each experiment result. As we shown in figure 11, we loaded 256 characters with 28 joints dance motion. A character model has 471 vertices and we applied geometric simplification on each character. We also applied skeletal simplification using JPC method. The experiment result is depicted in figure 12. The processing time of a frame using only geometric simplification is a little bit better than the skeletal simplification. However, by applying both mesh and skeletal simplification, we obtained a better result than any other experiments. If the articulated figure is more complex and the number of characters in the crowd scene increase, the processing time of the simplification must be enhanced.

Figure 11: Crowd scene with 256 characters

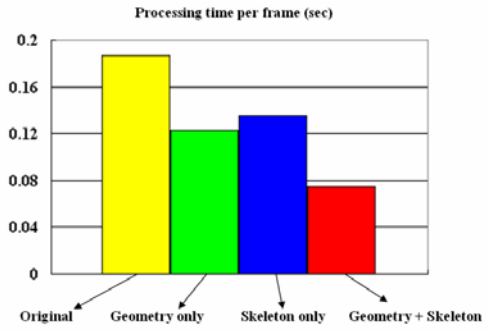


Figure 12: Experiment result of the crowd scene

6.2 Conclusion and future work

In this paper, we presented the 'motion level-of-detail' framework to enhance processing time of the real-time crowd environment. To maintain the visual quality, we applied a motion dependent skeletal simplification method. Moreover, geometric simplification is considered to our framework to improve our system performance. We controlled the detail of the motion and the geometry by the distance between the camera and the character's position. As the result of our experiments, the image and the animation quality was affordable. Our preprocessing method was particularly useful to the monotonous motion and complex articulated figure's motion with many joints. Because of the tree reconstruction time, variable motion was less efficient than the monotonous motion.

Using our method to the complex crowd scene like urban or game environment, we expect a more enhanced performance, since in that kind of environment a lot of characters are occluded during the motion sequence. We estimate that the number of characters and the

complexity of joint and mesh structure in the crowd scene will increase in the near future. The simplification method in the crowd scene will surely be a more challenging issue.

Future improvement is needed to our framework. First, we didn't consider about view-dependent simplification. By applying view-dependent method, we should be able to recover more enhanced result. In this paper, we applied traditional geometric simplification, and we simplified motion and mesh independently. Another work we need to improve is embedding the local geometric part to the joint segment. By embedding the geometry to the joint segment, we should be able to control the motion and the mesh detail simultaneously and expect a more affordable simplification result.

Acknowledgements

This research has been funded by 'Seoul Institutes of the Arts'. We would like to thank 'KAIST VR Lab', 'Kaydara Motion Builder™' and 'HiWin' for providing character motions and models.

References

[1] Z. Popovic and A. Witkin. Physically based motion transformation. ACM SIGGRAPH 99, pages 11-20, 1999.

[2] D. Carlson and J. Hodgins. Simulation levels of detail for real-time animation. Graphics Interface 97, 1997.

[3] H. Hoppe. Progressive mesh. ACM SIGGRAPH 96, pages 99-108, 1996.

[4] M. Garland and P. Heckbert. Mesh simplification with quadric error metrics. ACM SIGGRAPH 97, pages 209-216, 1997.

[5] H. Kim and K. Wohn. Multiresolution model generation with geometry and texture. VSMM 2001, 2001.

[6] F. Tecchia and Y. Chrysanthou. Real-time rendering of densely populated urban environments. 10th Eurographics Workshop on Rendering, pages 45–56, 2000.

[7] A. Aubel, R. Boulic and D. Thalmann. Real-time display of virtual humans: Level of details and impostors. IEEE Transactions on Circuits and Systems for Video Technology, Special Issue on 3D Video Technology, 10(2): 207–217, 2000.

[8] E. Bouvier, E. Cohen and L. Najman. From crowd simulation to airbag deployment: particle systems, a new paradigm of simulation. Journal of Electronic Imaging, 6(1): 94-107, 1997.

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

[10] D. Helbing. A fluid-dynamic model for the movement of pedestrians. Complex Systems, pages 391–415, 1992.

[11] S. Shekhar and Q. Lu. Evacuation planning algorithms: A capacity constrained routing approach. 2002.

[12] X. Tu and D. Terzopoulos. Artificial fishes: physics, locomotion, perception, behavior. ACM SIGGRAPH 94, 1994.

[13] D. Brogan and J. Hodgins. Group behaviors for systems with significant dynamics. Autonomous Robots, 4: 137-153, 1997.

[14] S. Musse and D. Thalmann. A model of human crowd behavior. Workshop of Computer Animation and Simulation of Eurographics 97, 1997.

[15] C. Reynolds. Flocks, herds and schools: A distributed behavioral model. ACM SIGGRAPH 87, 1987.

[16] T. Kim, S. Park and S. Shin. Rythmic motion synthesis based on motion beat analysis. ACM SIGGRAPH 03, 2003.

[17] B. Blumberg and T. Galyean. Multi-level direction of autonomous creatures for real-time virtual environments. ACM SIGGRAPH 95, pages 47–54, 1995.

[18] K. Perlin and A. Goldberg. Improv: A system for scripting interactive actors in virtual worlds. ACM SIGGRAPH 96, pages 205–216, 1996.

[19] J. Lee, J. Chai, P. Reitsma, J. Hodgins and N. Pollard. Interactive control of avatars animated with human motion data. ACM SIGGRAPH 02, 2002.

[20] T. Di Giacomo, C.Joslin, S. Garchery, N. Magnenat-Thalmann. Adaptation of Facial and Body Animation for MPEG-based Architectures. IEEE Cyberworlds, pages 221, 2003.

[21] F. Tecchia, C. Loscos and Y. Chrysanthou. Image Based Crowd Rendering. IEEE CG&A March/April, pages 36-43, 2002.