Animation Model Simplifications

Fu-Chung Huang* Bing-Yu Chen† Yung-Yu Chuang‡ Ming Ouhyoung‡
National Taiwan University
*jonash@cmlab.csie.ntu.edu.tw †robin@ntu.edu.tw ‡{cyy,ming}@csie.ntu.edu.tw

ABSTRACT

In this paper, we propose a new framework for the representation of deforming meshes by only updating necessary changes of the connectivity. The deforming meshes, which is also known as time-varying surfaces, are often constructed with static connectivity. To progressively represent the deforming meshes with level-of-details, people can simplify the meshes independently to obtain good simplification meshes but total different connectivity for each frame. On the other hand, people can also simplify the meshes while keeping their static connectivity, but this constraint makes the simplified meshes distorted. Hence, we present a feature-adaptive simplification scheme, which takes all time-dependent information into consideration. Through the application of one-dimensional Haar wavelet, we successfully quantize the costs of all edges of the deforming meshes. Then, we apply a global optimization to minimize the meshes' distortion while maximizing the temporal coherence. The optimization is hard in terms of computational complexity, and we therefore exploit the advantages of genetic algorithm together with dynamic programming to solve this problem. The result makes the progressive deforming meshes represented without updating the connectivity rapidly and have good simplification meshes for each frame.

1. INTRODUCTION

To date, more high resolution 3D animated models are required to present important details and fine structures, however, sometimes such high resolution models are un-necessary and undesired. Mesh simplification is a process to remove such un-necessary or redundant data from a high resolution 3D model. Hence people can get a sequence of 3D models with continuous level-of-details (LOD) through this process by removing the vertices, edges, or faces from the original 3D model. However, to remove the vertices, edges, or faces from the 3D model usually makes the simplified model distorted. Various metrics were then proposed to measure the deviation from the original mesh and the simplified one, so that the distortion of the simplified model can be minimized.

In the past years, many mesh simplification methods for simplifying a static 3D model have been proposed. However, the 3D model with motion data, or so-called deforming meshes or time-varying surfaces, is more widely used in many fields, like on-line games, animations, etc. Therefore, how to provide a method similar with these mesh simplification methods but working for the deforming meshes is necessary.

To simplify the deforming meshes, some previous methods focused on preserving static connectivity, i.e., the connectivity of the deforming meshes in all frames remains unchanged in all resolutions. The static connectivity, like the metamesh used in 3D metamorphosis, aggregates the features of all frames in one model, where subsequent simplification is applied. However, such adaptation is inadequate and the results are often not satisfactory. Figures 1 (c) and (d) show the example, which is a 3D morphing sequence from a horse to a man originally. Since the connectivity of the meshes in all frames in Figure 1 (c) is the same, but the features of a horse may not be the features of a man, the simplified model shown in Figure 1 (c) has distortion in the mouth area. On the other hand, Figure 1 (d) is our result which does not keep the connectivity the same and has no such problem.

In contrast to early work using static connectivity, recent methods start to change the connectivity adaptively and dynamically to improve the quality of the resulting meshes. These methods start with the model in the first frame, and incrementally update the connectivity so that the model in the next frame is well-approximated. In order to provide an appealing simplified mesh, great amount of updates is inevitable, and so does the popping effect. If great constraints on the *temporal coherence*, which measures the consistency of connectivity of meshes from frame-to-frame, are imposed, errors may propagate accordingly, and the result will behave like a first-frame-connectivity approximation. The fundamental problem is that the model in either the first or arbitrary frame does not represent a good compromise between meshes distortion and connectivity updates.

In this paper, we propose a similar approach using dynamically changing connectivity, and achieve a better result than previous work. Under this framework, the models in each frame are simplified separately using Quadric Error Metrics (QEM) [9], which contracts a pair of vertices and collapses an edge as a basic operation to reduce the mesh primitives. Then, the popping effect problem is eliminated by solving

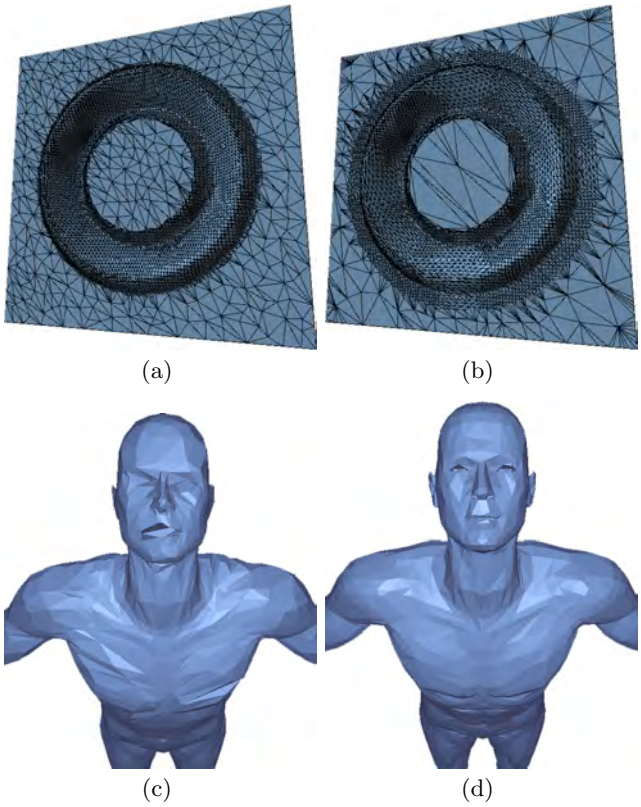

Figure 1: (a) and (b) are snapshots of an animation of a sine wave, which spread from the center to the outer of the rectangle. (a) shows the result of Direct-QEM and (b) shows our result; both with the same number of polygons and under the same condition of simplification. In our method, additional connectivity is preserved for the next few frames just around the wave, whereas Direct-QEM does not. (c) and (d) show one of the simplified frame of a 3D morphing sequence with (c) static connectivity and (d) dynamic connectivity, where (d) outperforms in the quality of the simplified mesh in this frame.

an optimization problem. Hence, our problem is defined as: *minimize meshes distortion while maximize temporal coherence*. In simple words, our goal is to find a sequence of meshes which approximate good compact version of the original meshes and the amount of work spent on updating frame-to-frame connectivity is minimized.

A trivial solution to this problem may consist of N^m possibilities, where N is the number of vertices and m is the number of frames. Randomized algorithm can assist the search. In order to solve the optimization problem, we divide it into two sub-problems. The first finds a feasible set of solutions and the second search the best solution within the set found by the first sub-problem. To find a feasible set of solutions, we use the concepts of one-dimensional Haar wavelet decomposition and vertex tree which is widely used in terrain rendering. The edge costs in temporal dimension are treated as a signal, where we apply wavelet decomposition. By removing high frequency wavelets, the rest provides a coherent quantization. The errors are dispersed to nearby frames so that adaptation to a feature can be prepared in advance. The process of edge-collapse simplification in each frame implicitly constructs vertex trees. Each branch in the tree represents an edge-collapse and then we search an optimal combination using these branches.

For each iterating simplification, we choose a branch of the tree, or an edge-collapse, in each frame and these branches together form a solution to our problem. Solution found in this way does not represent a good one if the selections are not taken carefully and optimal solution is also hard to be found. As for the second sub-problem, we exploit the genetic algorithm (GA). Implicit parallelism, an important property of GA, searches the solution simultaneously by guessing and improving several feasible solutions at the same time. A series of recombination produce potentially better solutions, and mutations reinforce the search. Convergence is accelerated by integrating dynamic programming, where the feasible solutions reach the local minimum faster.

In this paper, our major contributions are:

- Propose a framework that simplifies deforming meshes dynamically with improvement on the temporal coherence. One-dimensional Haar wavelet decomposition is used to assist the quantization.
- Develop an energy function that captures the importance of both meshes distortion and temporal coherence at the same time.
- Solve the energy function with the assist of genetic algorithm and accelerate the convergence using dynamic programming.

2. RELATED WORK

2.1 Multiresolution Mesh

One common technique in reducing data is multiresolution. There are a plethora of papers have been published to solve this problem, notably using re-meshing and simplification. Various approaches regarding re-meshing are meant to optimize a good re-sampling [8][5] over the original mesh surface; this branch is, however, not yet applied to time-varying surfaces.

As for simplification, vertex-removal [18], vertex-clustering [4], and edge-collapsing [9][10][16] fall into this category. These techniques typically choose a primitive causing the least distortion measured as proposed, and conduct primitive elimination operation. Traditional mesh simplification algorithm works fine on a single static model, yet the adaptation, when directly applied to deforming model, can lead to catastrophe.

2.2 Static Connectivity on Deforming Models

Traditional animation simplifications use a single connectivity to approximate all frame models. [17] adapts QEM algorithm by constructing a meta-mesh which sums all frames quadric error metrics. The meta-mesh uses the aggregated errors as a guide to simplify itself, and the same process is

applied to all frames models. The resulted animation works quite well at medium resolution, yet manifest distortion can be easily seen at low resolution. [7] makes an extension by weighing possible configuration of poses with probabilities. With articulated meshes, skeleton transformation is incorporated into standard QEM algorithm, and user must specified probability distribution on joint for weighing. [20] also uses QEM as base simplification module. By extracting high frequency transformation, simplification is applied to base mesh and results are acquired by inverse transformation on simplified base. Applying single connectivity to all frames produce moderate approximation. Using this adaptation, very simple data structure is used for the representation and the quality is satisfactory in general. A potential limitation is posed in the simplified connectivity, where unnecessary information at unimportant area is used in some frames, while real important features at certain frames are described by fewer indispensable surface descriptors. The phenomena can be easily shown when extreme simplification is applied, as shown in Figure 2.

(a) (b) (c)

Figure 2: An comparison of Direct-QEM, our method, and DSD. The Direct-QEM approximate the best shape of the original dog model. The tail and front leg in our method behave similar to Direct-QEM but has superior temporal coherence. DSD, on the other hand, shows inferior approximation. The jaw and the forelimb are missing, and the tail and the back are simplified poorly.

2.3 Dynamic Connectivity on Deforming Models

[19] designs a scheme for changing connectivity meshes simplification. Time-dependent Directed Acyclic Graph (TDAG) is introduced by merging individual simplification on each frame into a unified graph. TDAG is capable of handling geometry deformation and topology modification. Major drawback inherits from the non-incremental construction and time-inefficient traversal at retrieving queries of updates. Frame to frame coherence is preserved by taking history of decimation into account, subsequent frame can be affected by inadequate previous frame approximate, and such phenomena may propagate throughout the entire sequence.

The work most representing ours is [14], where it uses edge-swap as the major update operation. Next frame simplification can be obtained by a sequence of edge-swaps from the simplified model of previous frame. Inadequate propagation is overcome by applying only valid and beneficial swap operation. Visually appealing animation is obtained by their hierarchy improvement and updating procedure. Huge amount of update operation will, however, be present at extreme deforming models, and can not be overcome by further propagate updates to more advance frames. Lack of control over the meshes quality and temporal coherence is a problem.

2.4 Meshes Data Compression

Another way to compact the sequence of deforming model is through data compression. General compressing tools have poor performance; some researchers consequently develop advanced techniques to compress meshes. [15] proposes compression of 3D animation by decomposing time-varying geometries to SVG matrices. Matrix $\mathbf{V}$ is further decomposed into multiplication of transformation matrices. Also the coherence in the rows and columns are exploited for decomposition. Using information decomposed previously, prediction together with quantized residuals restores the original meshes. [1] use Singular Value Decomposition to find principle components of an animation sequence. Lossless or lossy compression is controlled by either preserving all components or discarding some less important information. [11] shows how to use predictors, ELP and Replica, to compress meshes. Perfect prediction can be achieved when deforming model follows either simple translation or rigid body motion respectively. In extension to geometry image, [2] use RGB as XYZ coordinates to compress time-varying deforming model as video sequence. After parameterization, original topology information is transformed into grids, where multi-resolution is done by scaling the resolution of the video. Advanced techniques on video compression can also be applied. Recent publication [12] automatically finds bones and vertex weights, by using transformation identified, their work enable efficient hardware rendering. Meshes data compression performs excellent on compacting data than general compression tools, yet data size is not reduced at rendering time and varying amount of additional computation is spent.

3. ALGORITHM

Our intuition is that if the models in consecutive frames have similar edge-collapse costs, their simplification process may be similar, which can result in near identical vertex trees. The near identical vertex trees guide us easier to find solutions which can minimize meshes distortion while maximize temporal coherence. Our algorithm is divided into two phases consequently:

1. Construct the near identical vertex trees for models in each frame.

2. In each vertex tree, iteratively select a valid vertices pair for contraction from the full resolution. The selections are made to complete the two objectives described previously.

The rest of this section is organized as follows. In Section 3.1, we first introduce how the improvement on vertex trees construction will be done by using one-dimensional Haar wavelet decomposition. In Section 3.2, a formulation of an energy function incorporating the meshes distortion and temporal coherence will be given, and a trivial solution using dynamic programming is presented. In Section 3.3, a better solution to the above formulation is enhanced by using genetic algorithm (GA). The power of implicit parallelism finds multiple feasible solutions at the same time. Hybridization with dynamic programming accelerates the convergence to a local minimum.

3.1 Vertex Tree Construction

In [9], QEM models the distortion cost of a vertex as the quadric distances from its neighboring planes. For a deforming model, the moving vertices cause the orientation of planes changed consequently. Several optimal vertex positions can be determined at each time stamp for an edge-collapse operation. The quadric metrics of a vertex v at time stamp i is identified as $\mathbf{Q}_{v,i}$.

In [17] and some other similar papers, the edge-collapse cost of an edge (v_1, v_2) during the iterative contraction is equal to the sum of the contraction errors in all time stamps:

$$\mathbf{Q}_{v'} = \sum_t v_t'^{\mathrm{T}} (\mathbf{Q}_{v_{1,t}} + \mathbf{Q}_{v_{2,t}}) v_t',$$

where v_t means the newly generated vertex in time stamp t due to QEM calculation. This formulation treats multi-valued pair contraction as single-valued, and thus a single decimation sequence is applied to the deforming meshes of all frames, as if they are approximated by one feature-aggregated model.

Instead of applying only one decimation sequence to all frames, we simplify the deforming meshes of each frame separately, and apply additional quantization to the quadric error metrics. Because the difference between the quadric error metrics of the consecutive frames $Q_{v,i}$ and $Q_{v,i+1}$ may be slight or excessive as the model deformed, the structure of the vertex trees may vary differently, if the mesh simplification was applied to the deforming meshes in a straightforward manner. Such observation is also described in [15] and the variation of the quadric error metrics is shown as the blue line in Figure 3.

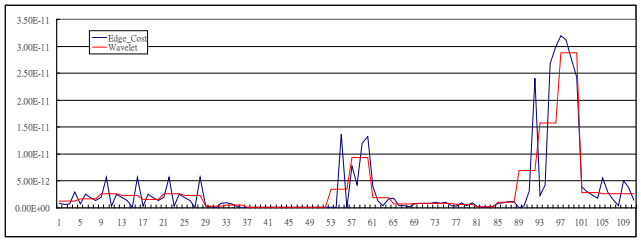

Figure 3: The blue line indicates how the cost of an edge-collapse changing in each frame. The red line shows the result after applying wavelet quantization. Noises are removed by discarding high frequency wavelets.

Incoherent structure can lead to huge amount of updates and obvious popping artifacts at rendering. To solve this problem, we provide an optimization solution to smooth the incoherent structure based on an one-dimensional Haar function and genetic algorithm with dynamic programming. The one-dimensional Haar function is compounded of two functions, which is a mother scaling function $\Phi(x)$ and a mother wavelet function $\Psi(x)$, together with a series of coefficients. The original one dimensional function or image in a series of coefficient times scaling basis function can be transformed into a single scaling basis function plus a series of coefficient times wavelet basis function, which describe high frequency information of the original image.

The transformation has one advantage that the transformed function still lies in the original domain, unlike Fourier transformation in frequency domain, and frequencies are expressed in an increasing order. Different treatments can be made on the new coefficients for different purposes. Hence, we use an one-dimensional Haar wavelet decomposition due to its base functions exhibit box-like orthogonality, on which our quantization is dependent. Another advantage is that the function values are disseminated at a low-frequency support, and feature areas in latter frames can be prepared in advance. The result of quantization is exemplified as the red line in Figure 3.

By discarding high-frequency wavelet coefficients with a specified level, the quantization is completed, and based on the quantization the coherent structures of vertex trees within the support can be built. Our simplification scheme is competitive in that the features are preserved at the needed frames, where the static-connectivity scheme suffers, as shown in Figure 4.

(a) (b) (c)

Figure 4: The comparison of feature adaptive schemes of (a) First-Frame-Static, (b) Deformation Sensitive Decimation, and (c) ours. The toes are the last frame of the horse to man morphing sequence, and our method adaptively preserves more details then other methods.

Our algorithm is as follows:

1. Compute the groups for every edge.
2. Initialize quadric error metrics for every frame.
3. Apply one-dimensional Haar wavelet quantization to every edge according to the groups in Step 1.
4. Contract edges iteratively in each frame. Errors of contracted and degenerated edges are set to zero, and apply the Haar wavelet quantization again to the edges whose errors in other frames have been changed.

Note that re-sampling is required if the number of frames does not match 2^n, edges contracted are preferred to that of degenerated in other frames. One thing worth to note is that if all wavelets are discarded, our algorithm perform exactly the same as [17].

3.2 Energy Function Formulation

After the construction of the vertex trees, we will reorder the decimation sequence of each frame. Some previous methods consider only two consecutive frames at a time; we instead incorporate distortion and coherence as an energy function of all frames as the following function:

$$E = \alpha \sum_t \mathbf{Q}_{v,t} + (1-\alpha) * \xi(\text{Level}) * \sum_t \text{Update}(vf_t, vf_{t+1}, v_t, v_{t+1}) \quad (1)$$

This energy function states that errors are the summation of distortion cost of v_t at each frame plus the changes of quadric error metrics in updates cost of vertex fronts vf_t from frame to frame, which can be positive or negative indicating that the decimation of v_t and v_{t+1} in time t and $t+1$ causing the change in the updates is beneficial or hazardous. The vertex fronts vf_t here means a set of currently visible vertices in time t. The quadratic term $\mathbf{Q}_{v,t}$ penalizes the peak of connectivity updates accumulated in certain frames. ξ(Level) is an exponential adjusting function, which adapts the unit of update cost to distortion error, and also reflects how distortion cost grows. We use the averaged costs in decimation sequences of all frames generated in the initial phase to fit the function. The result is an averaged distortion costs to level of details curve. Serious impulse noise is present in the error curve fitted and median filter can be used to reduce noises. Update$(vf_t, vf_{t+1}, v_t, v_{t+1})$ is a function that measures how the two edge-collapses v_t and v_{t+1} in time slots t and $t+1$ can cause the connectivity updates from vertex-front vf_t in t to vertex-front vf_{t+1} in $t+1$. The graphical illustration is shown in Figure 5.

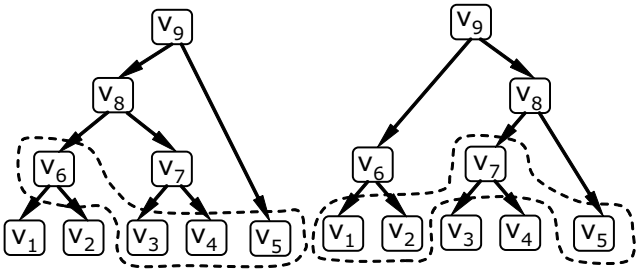

Figure 5: Vertex trees of consecutive frames t and $t+1$. They share the same full resolution vertices, yet the vertex fronts are slightly different. If an edge-collapse selection on t is V_7 on time t and V_6 on $t+1$, the evaluation of Updates () return -2 indicating benefits. If V_7 is selected on time t, a benefit is gained, but may be counterbalanced by the selection of V_8 on time $t+1$. Original updates on connectivity are two, one on refinement of V_6 and one on coarsening of V_7.

Then, we use dynamic programming to solve this formulation intuitively, by first selecting a set of edge-collapse candidate, and compute a best combination that minimize the energy function for each simplification iteration. The algorithm is as the follows:

1. Collect S, a set of arbitrary edge-collapse candidates.
2. For each frame, a subset S_t is derived from S, by the intersection of S and available candidates in frame t.
3. Run optimization algorithm
4. Best solution can be found by starting from the least value in f[n] and traced backwardly.

In Step 1 of the above algorithm, set S is generated through the least distortion cost collected from each frame without duplication. Performance of the above algorithm is bounded by the Update() function. Hence, we use dynamic programming to enhance the speed of calculation. Dynamic programming provides fast and approximate solution for our energy function only if that S is chosen well. The search space is limited by the candidate set S, where the edge-collapses with least distortion cost in each frame are used as initial guess. We later exploit the power of genetic algorithm for global solution search.

3.3 Genetic Algorithm

In Section 3.2, we utilize dynamic programming to find a feasible solution to our energy function. A limitation is imposed by the candidate set found in Step 1. In this section, we introduce the general genetic algorithm and explain how to modify the general one to fit our specific use. The general genetic algorithm [6] is given by:

1. Initialize a population of chromosome.
2. Evaluate each chromosome in the population.
3. Create new chromosomes by mating parents; apply mutation and recombination as the parent chromosome mate.
4. Delete members of the population to make room for the new chromosomes.
5. Evaluate the new chromosome and insert them to the population.
6. If time is up, stop and return the best chromosome; if not, go to Step 3.

Original genetic algorithm encodes chromosome by bit-string, yet other encodings like alphabetic or real numbers may suffice. We concatenate edge-collapse of each frame in a sequence as the encoding. Evaluation of chromosomes is given by a fitness function, where our energy function is very appropriate for this purpose, and this is why we use genetic algorithm instead of other discrete combinatorial optimization scheme like simulated annealing or tabu search.

Initial population is given by mutating the very first chromosome we created from the one used in the previous subsection. The mutation of a chromosome is randomly replacing an edge-collapse in an arbitrary position by another edge-collapse candidate. The switch randomly selects one of the first n least cost edge-collapses of the model in that time slot, where n is a user specified parameter, and we use 100.

Two-point crossover recombination replaces the original one-point scheme, used in general genetic algorithm, and the

points are selected at the highest gradient point in the chromosome. The gradient captures how the energy function accumulates from each frame. High gradient occurs where the edge-collapse in time stamps t and $t+1$ cause connectivity to increase/decrease or change of distortions dramatically.

The mating of two chromosomes is done by linear normalization of the fitness function and random selection. We sort chromosomes in their fitness function value, assign ranking value to replace the original fitness value, and use roulette wheel for random selection.

Genetic algorithm exhibits an excellent performance, yet we also must note that simply use dynamic programming may work as well, in that temporal coherence is not a great concern because the initial solutions have provided good guesses. However if strong constraint is imposed, dynamic programming perform poorly.

4. RESULTS

In this section, all of the results were running on a notebook PC with an Intel Pentium M 1.6GHz CPU and 2.0GB RAM. The parameter α in our energy function through out this paper is set to 0.5, unless otherwise specified. In Section 3.3, the population size is set to 20, with crossover probability of 0.95 and 0.1 for run-time mutation probability. The initial population is generated using mutated least-distortion sequence, with mutation probability set to 0.2. In order to improve performance, elitism is used. The search, for most of the time, converges within a hundred run, and we set the search to be 100 times.

A comparison of genetic algorithm with random selection is also conducted. In order to make it fare to collate, chromosomes generated in genetic algorithm and random selection are the same, where search-runs multiply population size is total 2000 here. Our experiments show that genetic algorithm outperform random selection.

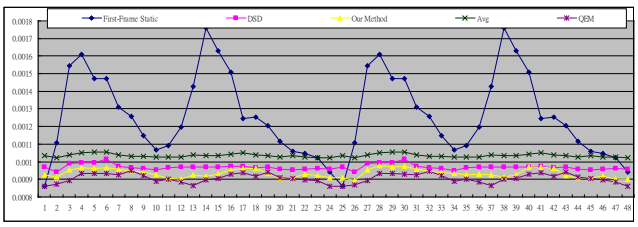

Figure 6: Various methods compared in RMS measuring meshes distortion.

First in Figure 6, a comparison with other schemes is demonstrated. We have implemented the Deformation Sensitive Decimation, First-Frame-Static approximation, and the "Average" approximation. Apparently our result has good result at approximating the original deforming meshes, and the First-Frame-Static is obviously a bad choice for simplifying animating sequence. The result can also be seen in Figure 1. For drastically deforming sequence, the horse collapsing animation exemplifies importance of our method, and comparison for every level of details can be found in Figures 7 and 8. The distortion measuring tool we use is Metro [3], and average the result of two way measurement.

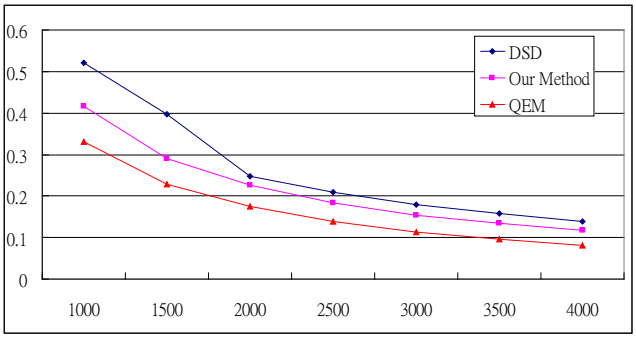

Figure 7: The comparison of our method with static connectivity schemes. As meshes start coarsening, the distortion of static scheme increases faster than ours.

(a) (b)

Figure 8: The result of horse-collapose animation at 26th frame by using (a) Deformation Sensitivity Decimation using static connectivity scheme and (b) our method. Our method not only preserves necessary details in advance as in Figure 1 but also provides good approximation.

In Figure 9, we show an illustration of how control is done on the number of updates. GA is an excellent choice for optimization when great constraint is imposed on temporal coherence.

5. CONCLUSIONS AND FUTURE WORK

In this paper, we presented an algorithm for simplifying the deforming meshes with changing connectivity dynamically. In our method, we use an one-dimensional Haar wavelet function to quantize the quadric error metrics and genetic algorithm with dynamic programming to find the best combination of a set of edge-collapse operations which minimizes distortion while maximizing temporal coherence. Hence, we can obtain a simplified deforming meshes with updating only necessary connectivity.

Some animation techniques, such as key-framing and skinning, usually require the connectivity to be static, one of our future work can addresses this problem by incorporating dynamically adaptive connectivity with key-framing or skinning.

One extension to this work is to re-design the energy function by putting a third term to measure the perceptional

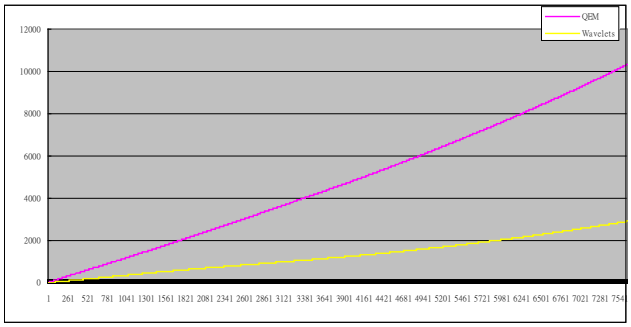

Figure 9: Our method versus Direct-QEM in number of updates. As the level of details decreasing, the time spent on the updating connectivity is very huge, and sometimes even exceed the cost of retransmitting a whole new mesh.

popping effects. Throughout this paper we assume that reducing the numbers of updates can eliminate popping effects implicitly. Imagining that a connectivity update occurs at completely flatten plan and curved surface. The former is undoubtedly less easy to be perceived, yet the two cases evaluated to be equal in our function. Another issue is that popping effects are perceived at the vertex front level, whereas the updating happens throughout the entire vertex tree hierarchy. It is inadequate to simply measure the connectivity change at vertex front level or put the entire simplified structure into consideration.

A very tentative future work relies on the segmentation of spatial-temporal domain. By explicitly segmenting the surface into drastically deforming parts and less deforming parts, the simplification can treats these areas differently, and thus better coherent structure may be built and easier control on optimization can be done. This extension can be incorporated with the existing work [13].

6. ACKNOWLEDGEMENTS

We wish to thank Robert W. Sumner and Jovan Popović for providing the galloping horse and collapsing horse animations. Thanks also give to Alla Sheffer for the horse-to-man morphing data. We thank Scott Kircher and Michael Garland for generously sharing the BigCape deforming model. This work was partially supported by the National Science Council of Taiwan under the numbers: 92-2218-E-002-056, 93-2213-E-002-084, and 94-2213-E-002-097.

7. REFERENCES

[1] M. Alexa and W. Müller. Representing animations by principal components. *Computer Graphics Forum (Proceedings of Eurographics 2000)*, 19(3):411–418, 2000.

[2] H. M. Briceño, P. V. Sander, L. McMillan, S. Gortler, and H. Hoppe. Geometry videos: a new representation for 3d animations. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 136–146, 2003.

[3] P. Cignoni, C. Rocchini, and R. Scopigno. Metro: measuring error on simplified surfaces. *Computer Graphics Forum*, 17(2):167–174, 1998.

[4] J. Cohen, A. Varshney, D. Manocha, G. Turk, H. Weber, P. Agarwal, F. Brooks, and W. Wright. Simplification envelopes. In *ACM SIGGRAPH 1996 Conference Proceedings*, pages 119–128, 1996.

[5] D. Cohen-Steiner, P. Alliez, and M. Desbrun. Variational shape approximation. *ACM Transactions on Graphics (SIGGRAPH 2004 Conference Proceedings)*, 23(3):905–914, 2004.

[6] L. Davis. *Handbook of Genetic Algorithms*. van Nostrand Reinhold, 1991.

[7] C. DeCoro and S. Rusinkiewicz. Pose-independent simplification of articulated meshes. In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games*, pages 17–24, 2005.

[8] M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle. Multiresolution analysis of arbitrary meshes. In *ACM SIGGRAPH 1995 Conference Proceedings*, pages 173–182, 1995.

[9] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *ACM SIGGRAPH 1997 Conference Proceedings*, pages 209–216, 1997.

[10] H. Hoppe. Progressive meshes. In *ACM SIGGRAPH 1996 Conference Proceedings*, pages 99–108, 1996.

[11] L. Ibarria and J. Rossignac. Dynapack: space-time compression of the 3d animations of triangle meshes with fixed connectivity. In *Proceedings of the 2003 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 126–135, 2003.

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

[13] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics (SIGGRAPH 2003 Conference Proceedings)*, 22(3):954–961, 2003.

[14] S. Kircher and M. Garland. Progressive multiresolution meshes for deforming surfaces. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pages 191–200, 2005.

[15] J. E. Lengyel. Compression of time-dependent geometry. In *Proceedings of the 1999 Symposium on Interactive 3D Graphics*, pages 89–95, 1999.

[16] P. Lindstrom and G. Turk. Fast and memory efficient polygonal simplification. In *IEEE Visualization 1998 Conference Proceedings*, pages 279–286, 1998.

[17] A. Mohr and M. Gleicher. Deformation sensitive decimation. Technical report, University of Wisconsin, 2003.

[18] W. J. Schroeder, J. A. Zarge, and W. E. Lorensen. Decimation of triangle meshes. *ACM Computer Graphics (SIGGRAPH 1992 Conference Proceedings)*, 26(2):65–70, 1992.

[19] A. Shamir, C. Bajaj, and V. Pascucci. Multi-resolution dynamic meshes with arbitrary deformations. In *IEEE Visualization 2000 Conference Proceedings*, pages 423–430, 2000.

[20] A. Shamir and V. Pascucci. Temporal and spatial level of details for dynamic meshes. In *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*, pages 77–84, 2001.