Selecting Independent Chains on a Triangulated 2-Manifold*

Siu-Wing Cheng† Sheung-Hung Poon†

Abstract

We prove that there exists linearly many independent edge chains of length k in a connected orientable triangulated 2-manifold (possibly with boundary) of genus zero, for any positive integer parameter k. Such an independent set of chains can be identified by a simple linear-time algorithm. Our motivation is to simplify a surface by repeatedly contracting a large set of independent chains. Preliminary experimental verification of the effectiveness of this approach in surface simplification has been conducted.

1 Introduction

Many graphics softwares, animation applications, computer games and virtual reality applications nowadays use 3D object models extensively. Realistic models often have high complexity, which slow down computation and rendering in applications. Sometimes, not all details are needed, for example, when the object is far away. Therefore, providing levels of details of a model will speed up its manipulation when the resolution needed is low.

Our motivation is to combine and extend two previous approaches in surface simplification, namely, decimation of independent vertices [1] and edge contraction [2, 3, 4, 5, 6]. First, we contract edge chains of length $k \geq 1$ instead of individual edges. Second, we identify a large set of *independent* edge chains for contraction. A set of chains are independent if for every pair of chains, they are disjoint and do not contain the two endpoints of the same edge. This guarantees that the chain contractions and hence chain recoveries do not interfere with each other. See Figure 1. It is thus possible to exploit the inherent parallelism to speed up the simplification and recovery.

In this paper, we prove that there exists linearly many independent edge chains of length k in a connected triangulated 2-manifold of genus zero, for any positive integer parameter k. Moreover, such an independent set of chains can be identified by a simple linear-time algorithm. This theoretical result suggests the possibility of significant reduction in model complexity in a single round of contractions of independent chains. There are other issues to be examined before this can be applied repeatedly to simplify a surface, and we will investigate this in our future work.

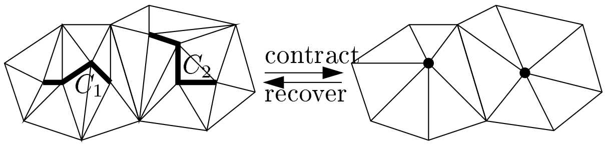

Figure 1: chain contractions and chain recoveries of independent chains C_1 and C_2 do not interfere with each other.

The rest of this paper is organized as follows. Section 2 briefly surveys previous work on surface simplification. Section 3 discusses chain contraction and recovery. We describe and analyze our algorithm for finding the independent chains in Section 4. In Section 5, we provide some implementation details and present some experimentation results. And finally in Section 6, we discuss related issues about contraction algorithms for simplification and propose some directions for future work.

2 Previous work

Some common techniques on surface simplification [7] are vertex clustering, vertex decimation, edge contraction, and patch decimation.

Vertex clustering [8, 9] means that some close vertices are clustered and represented by a single vertex. This technique is fast, but with low quality on the simplified result. *Vertex decimation* [10, 11] is a technique by destroying a vertex, and retriangulating the hole. And *edge contraction* [2, 3, 4, 5, 6] is by contracting an edge to a vertex recursively. The most well-known algorithm is the quadric error metric edge contraction proposed by Garland and Heckbert [2], which is efficient and have high-quality output. *Patch decimation* [12, 13] is a technique to de-

*Research partly supported by Research Grant Council CERG HKUST 6088/99E.

†Department of Computer Science, HKUST, Hong Kong. Email: {scheng, hung}@cs.ust.hk

stroy a set of triangular faces and retriangulate or remesh the hole(s) incurred.

Junger and Snoeyink [1] investigated a method of selecting an independent set of vertices to be deleted, while trying to preserve terrain . Different heuristics on the DFS traversal schemes on the TIN were proposed. Their results support progressive display of TIN terrain models in GIS under a scenario in which a server transmits the data for a client to reconstruct the terrain model. Choosing independent sets makes the client's task easier and has other advantages. Our approach extends their idea by selecting independent edge chains over a 2-manifold of genus zero for contraction instead of by selecting an independent vertices for decimation and retriangulation.

Hoppe [3] proposed a scheme called progressive meshes so that a sequence of edge contractions can be performed back and forth interactively. Our approach also somehow extends his idea to contract or recover chains (instead of edges) interactively and in parallel (instead of sequentially).

3 Chains

A *chain* is a simple path of edges. For any positive integer k, we call a chain a k-*chain* if there are k edges in it. Our simplification algorithm is based on contractions of chains, which is a natural extension of edge contraction.

Contraction. The *contraction* of an edge, say (v_0, v_1), moves v_0 and v_1 to a new position v, and connects all incident edges of v_0 and v_1 to the new vertex v. Any edge and face incident to v_0 and v_1 that have become degenerate is removed. Obviously, the edge (v_0, v_1) is removed. If (v_0, v_1) is a boundary edge, then one face will be removed; otherwise, two faces will be removed.

Let $(v_0, v_1 \ldots, v_k)$ be a k-chain. *Contraction* of the chain to the new vertex v' moves all vertices v_i, $1 \leq i \leq k$, to the new position v', connects all incident edges of v_i to vertex v', and subsequently remove all degenerate edges and faces. For each chain contraction, the number of degenerate edges is k, and that of degenerate faces is between k and $2k$.

The actual mechanism for determining the new vertex v' for a chain contraction is not needed for developing the result in this paper. We plan to adopt the quadric error metric scheme [2]. It defines for each face f of the input model a quadric $Q_f(x)$ equal to the squared distance of a point x to the plane containing f. Each vertex x of the input model is assigned the sum of quadrics for its incident faces. After the chain contraction, the new vertex v' is assigned a new quadric $Q_{v'}(x) = Q_{v_0}(x) + Q_{v_1}(x) + \ldots + Q_{v_k}(x)$. The position of v' is set to the value of x that minimizes $Q_{v'}(x)$.

Recovery. Basically, chain contraction is a successive edge contraction of the edges along the chain. We suppose here the contraction order of the edges is from the starting to the end of the chain consecutively. After the contraction of a k-chain, say $(0, 1, \ldots, k)$, k vertices are removed and one vertex is moved from its old position to a new position. We stick to the convention that each edge $(i, i+1)$ will contract to the destination vertex $(i+1)$. So when the chain contraction is done, the final destination vertex will be k. Now we let the old position for vertex i just before the chain contraction be $f(i)$. To encode the chain contraction, we need to encode all the edge contractions incurred. For instance, in the example of the Figure 2, we can encode the chain contraction as the

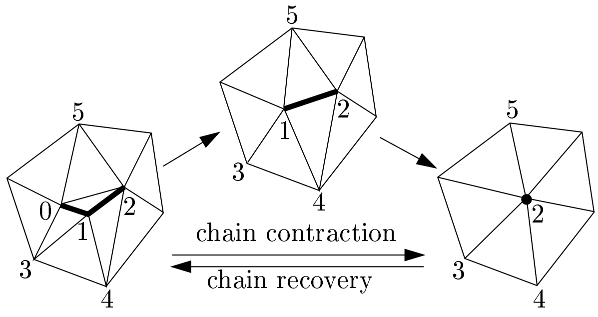

Figure 2: Chain contraction and recovery.

sequence $([0, f(0)], 3, 2, [1, f(1)], 4, 5, [2, f(2)])$, where $([0, f(0)], 3, 2, [1, f(1)])$ represents the first edge contraction, and $([1, f(1)], 4, 5, [2, f(2)])$ the second. The sequence is associated with the new destination vertex $(k+1)$, which is 3 in this example, so that by traversing the sequence reversely will recover the whole sequence of edge contractions and hence the chain contraction.

As we can see from Figure 2, if each time we only perform one edge contraction instead of one chain contraction, the data pair for vertex 1, which is $[1, f(1)]$, will be needed twice with different $f(1)$ values. So for a general k-chain, the encoding scheme for chain contractions instead of only edge contractions saves $k-1$ pair of data, $[i, f(i)]$, for each interior vertex i of the chain.

4 Selecting disjoint and independent chains

We define the degree of a chain C, $deg(C)$, to be the sum of the degrees of all vertices in C with respect to the input 2-manifold T.

Given a k-chain C in an arbitrary triangulated 2-manifold K, we define the *neighborhood* of C, $nb_K(C)$, to be the union of the triangular faces incident to C. See Figure 3 for an illustration. The patch enclosed by the closed chain $(1, 2, 3, 4, 5, 6, 7, 8)$ is the neighborhood of the chain $C = (a, b, c, d, e)$, and the degree of the chain is $deg(a) + deg(b) + deg(c) + deg(d) + deg(e) = 4 + 6 + 5 + 5 + 4 = 24$. Given any triangulation K, we define the boundary size of K, $bdsize(K)$, to be the number of the boundary edges of K, and the size of K, $size(K)$, to be the number of edges in K. The boundary size and size of the triangulation in Figure 3 are 8 and 28, respectively.

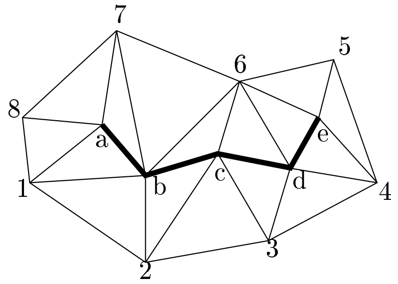

Figure 3: Neighborhood of the chain (a, b, c, d, e).

Finding disjoint chains. We initialize empty sets $\mathcal{S}$ and $\mathcal{F}$ and initialize a queue Q that contains T only. Then our algorithm iterates until Q is empty. In each iteration, we dequeue a 2-manifold K from Q and trace a simple path C, from an arbitrary vertex, of length k in a depth-first manner. If such a path cannot be found, then we dump K into $\mathcal{F}$. Otherwise, we add C to the set $\mathcal{S}$. We denote $nb_K(C)$ by $reg(C)$. We remove C and edges incident to C, but we do not remove the boundary of $reg(C)$. This may produce vertex whose link is no longer connected. We split all such vertices. This may generate several 2-manifolds and we append each to Q.

After the main loop ends, we run a supplementary step: For each chain C, repeatedly add to $\mathcal{S}$ k-chains from the boundary of $reg(C)$ that are vertex-disjoint from the existing ones in $\mathcal{S}$.

Clearly, the whole process takes $O(n)$ time.

Lemma 4.1 *The size of each 2-manifold in $\mathcal{F}$ is bounded by a constant depending on k.*

Proof: Consider the rooted DFS tree traversed in any 2-manifold H in $\mathcal{F}$. The tree height is less than k as we failed to trace a k-chain. Since there is no cross edge in a rooted DFS tree, each internal node has at most two children so as not to violate the property of a 2-manifold. So the number of vertices of the DFS tree is at most $1 + 2 + 2^2 + \ldots + 2^{k-1} = 2^k - 1$. It follows that the size of H is bounded by a constant depending on k as H is planar. $\square$

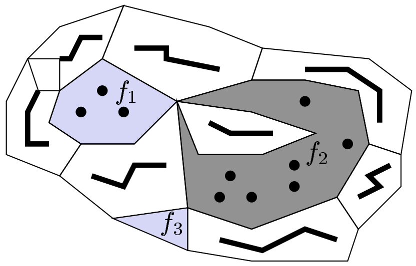

Figure 4: f_1, f_2, and f_3 are subgraphs dumped into $\mathcal{F}$.

Lemma 4.2 $\sum_{C\in\mathcal{S}} deg(C) = \Theta(\sum_{v\in V} deg(v)) = \Theta(n)$*, where n is the number of vertices in T.*

Proof: Clearly, $\sum_{C\in\mathcal{S}} deg(C) \le \sum_{v\in V} deg(v)$ as the chains in $\mathcal{S}$ are vertex-disjoint. For each subgraph H in $\mathcal{F}$ of at least two vertices, we attach H to $reg(C)$ for some chain C in $\mathcal{S}$ such that H and $reg(C)$ share a common edge. Since T is a connected triangulated 2-manifold, we can always find such a chain C if H has at least two vertices. After this step, the total number of edges in subgraphs attached to $reg(C)$ for a chain C is bounded by $\max_{f\in\mathcal{F}} size(f) \cdot bdsize(reg(C))$. Hence,

$$\sum_{f\in\mathcal{F}} size(f) \le \max_{f\in\mathcal{F}} size(f) \cdot \sum_{C\in\mathcal{S}} bdsize(reg(C)).$$

Then $\frac{1}{2}\sum_{v\in V} deg(v)$

$$\le \sum_{C\in\mathcal{S}} size(reg(C)) + \sum_{f\in\mathcal{F}} size(f)$$
$$\le \sum_{C\in\mathcal{S}} size(reg(C)) + \max_{f\in\mathcal{F}} size(f) \cdot \sum_{C\in\mathcal{S}} bdsize(reg(C)).$$

Clearly, $bdsize(reg(C)) \le deg(C)$ and $size(reg(C)) \le 2deg(C)$. Hence,

$$\frac{1}{2}\sum_{v\in V} deg(v) \le (2 + \max_{f\in\mathcal{F}} size(f)) \sum_{C\in\mathcal{S}} deg(C).$$

By Lemma 4.1, $\max_{f\in\mathcal{F}} size(f))$ is bounded by a constant, which implies that

$$\sum_{C\in\mathcal{S}} deg(C) = \Omega(\sum_{v\in V} deg(v)).$$

$\square$

Given two chains C and C' in $\mathcal{S}$, we call C and C' *adjacent* if C' has a vertex on the boundary of $reg(C)$.

Lemma 4.3 *Given a chain C in $\mathcal{S}$, either $deg(C) < 12k(k+1)^2$ or C is adjacent to $\frac{deg(C)}{12k(k+1)^2}$ chains in $\mathcal{S}$.*

Proof: Let T be the input connected triangulated 2-manifold. Consider the faces in $nb_T(C)$ that are incident to a vertex v in C, but do not share an edge with C. Since T is a 2-manifold, these faces form at most 3 disjoint regions and one of them contains at least $(deg(v)-4)/3$ faces. Denote this region by R_v. Removing from R_v the edges incident to v on C yields a path P_v.

P_v does not touch itself. Otherwise, P_v would contain a vertex w which forms a pinched edge with v, contradicting the property of a 2-manifold. If P_v contains $2k$ or more edges, then running DFS from any vertex in P_v should return a k-chain disjoint from C, and so P_v must contain a vertex on another k-chain in $\mathcal{S}$.

If C is not adjacent to any other chain in $\mathcal{S}$, then P_v contributes at most $2k$ to the degree of C. This implies that $deg(C)$ is at most $(k+1)\cdot(6k+2) < 12k(k+1)^2$ for $k \geq 1$.

Suppose that $deg(C) \geq 12k(k+1)^2$. Let v be the vertex such that P_v is longest among all vertices in C. Every sequence of $2k$ edges in P_v must contain a vertex in another k-chain in $\mathcal{S}$. Since each k-chain can share at most $k+1$ vertices with P_v, the number of k-chains adjacent to C is at least $|P_v|/2k(k+1)$. Since $|P_v| \geq (\frac{deg(C)}{k+1} - 4)/3$, the number of k-chains adjacent to C is at least $deg(C)/6k(k+1)^2 - 4/6k(k+1)$. This is greater than $deg(C)/12k(k+1)^2$ as $deg(C) \geq 12k(k+1)^2$. $\square$

Lemma 4.4 *There are $\Theta(n)$ disjoint k-chains in $\mathcal{S}$.*

Proof: Clearly, the number of chains in $\mathcal{S}$ is bounded by the number of edges in T which is $O(n)$. We prove that $|\mathcal{S}| = \Omega(n)$.

Each chain C in $\mathcal{S}$ may be adjacent to zero or more chains in $\mathcal{S}$. Since T is a triangulated 2-manifold, we can always select distinct edges of T to connect C to its adjacent chains in $\mathcal{S}$. Then we form a planar graph G with vertices corresponding to chains in $\mathcal{S}$ and the selected edges of T as edges of G.

Let S_0 be the subset of vertices in G with degree at most $12k(k+1)^2$. Let $\mathcal{S}_0$ be the corresponding set of chains in $\mathcal{S}$. Let S_1 be the set of remaining vertices in G. Let $\mathcal{S}_1$ be the corresponding set of chains in $\mathcal{S}$. Thus,

$$|\mathcal{S}_0| = |S_0| \geq \frac{1}{12k(k+1)^2} \sum_{C\in\mathcal{S}_0} deg(C) \qquad (1)$$

Lemma 4.3 implies that each vertex in S_1 has degree at least $\frac{deg(C)}{12k(k+1)^2}$. Thus, the number of edges in the subgraph induced by S_1 is

$$\frac{1}{2}\sum_{v\in S_1} deg(v) \geq \frac{1}{24k(k+1)^2} \sum_{C\in\mathcal{S}_1} deg(C).$$

Since the subgraph of G on S_1 is a planar graph, the above implies that there are $\Omega(\sum_{C\in\mathcal{S}_1} deg(C))$ vertices in S_1, or in other words, chains in $\mathcal{S}_1$.

Combining with equation 1, we conclude that the total number of chains in $\mathcal{S}$ is $\Omega(\sum_{C\in\mathcal{S}} deg(C))$ which is $\Omega(n)$ by Lemma 4.2. $\square$

Independence. From the set $\mathcal{S}$, we extract an $\Theta(n)$-size independent subset by applying a result due to Kirkpatrick [14]. Kirkpatrick's result states that a planar graph contains a linear-size independent subset of vertices, which can be found in linear time. Applying this to the planar graph G in the proof of Lemma 4.4 proves our claim.

Finally, we summarize our results in the following theorem.

Theorem 4.1 *Given a connected triangulated 2-manifold of genus zero, a linear number of independent k-chains, for any fixed constant k, can be identified in linear time.*

5 Experimentation & results

The algorithm we used for the experimentation in this part is actually different from the one presented above. The algorithm implemented is simpler but our current proof of its correctness is more complicated than the one in Section 4. Since the general idea is the same, we choose the present the simpler proof.

The algorithm implemented works as follows. We first initialize empty set $\mathcal{S}$ and load a queue Q with all vertices (which are unmarked) of T. Then our algorithm iterates until Q is empty. In each iteration, we dequeue a vertex v from Q. If the vertex was marked, then skip that vertex. Otherwise starting from v, we try to trace in a depth-first manner along the unmarked vertices on T for a simple k-chain. Whenever we visit an unmarked vertex, we mark it. If a chain is found, it will be added to $\mathcal{S}$. And then we go on to run next iteration.

On the other hand, in our implementation, instead of just selecting chains arbitrarily without considering the errors after their contraction, we try to grow up the chain by the edge with smallest quadric error at the two endpoints of the current incomplete chain. This heuristic helps to preserve the shape of the original surface.

After each level of contraction of independent chains. As we expect, some vertex/edge pinchings appeared because basically the edge contraction does not guarantee topological preservation. However, in some cases, the vertex/edge pinchings are due to the dangling edges or faces appeared after contraction around tetrahedron-shaped subgraph (See Figure 5). We simply remove these dangling edges and faces and

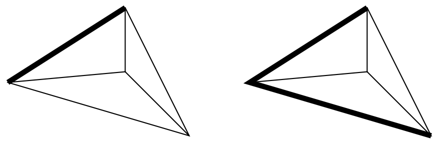

Figure 5:

thus remove the pinchings.

In the following experiments, we use two models, one is the cow model, which has 5804 faces originally, and the other is the bunny model, which has 69451 faces originally.

In tables 1, 2, and 3, we have some results about how many independent chains found with respect to different surfaces. The number of faces on the surface is denoted by #faces, and number of independent chains found on the current surface is denoted by #ichains.

Tables 1, 2, and 3 show that ratio of #faces to #ichains are approximately constant. This suggests that our algorithm can select linear number of independent chains as expected from our theoretical result.

From tables 1 to 3, we also observe that a greater reduction in the complexity of the surface is achieved with a larger k. For example, Table 1 shows that 361 vertices are removed in a surface of 5804 faces, and 94 vertices are removed in a surface of 1426 faces. On the other hand, Table 2 shows that 540 vertices are removed in a surface of 5804 faces, and 147 vertices

#faces	#ichains	#faces/#ichains
5804	361	16.08
5082	335	15.17
4412	292	15.11
3828	255	15.01
3318	218	15.22
2882	183	15.75
2514	159	15.81
2196	151	14.54
1896	126	15.05
1642	108	15.20
1426	94	15.17
1238	83	14.92
1072	70	15.31
932	61	15.28
810	54	15.00
700	45	15.56

Table 1: The original cow model was simplified successively by contracting the collected sets of independent **1-chains** on each level. For last column, mean=15.26, and s.d.=0.38.

#faces	#ichains	#faces/#ichains
5804	180	32.24
4720	151	31.26
3808	137	27.80
2796	99	28.24
2368	85	27.86
1848	64	28.88
1454	49	29.67
1154	37	31.19
922	31	29.74
734	27	27.19
564	19	29.68

Table 2: The original cow model was simplified successively by contracting the collected sets of independent **3-chains** on each level. For last column, mean=29.43, and s.d.=1.63.

#faces	#ichains	#faces/#ichains
5804	141	41.16
4662	124	37.60
3656	105	34.82
2790	74	37.70
2172	55	39.49
1716	41	41.85
1360	33	41.21
1086	31	35.03
818	22	37.18
624	14	44.57

Table 3: The original cow model was simplified successively by contracting the collected sets of independent **4-chains** on each level. For last column, mean=39.06, and s.d.=3.15.

are removed in a surface of 1454 faces.

In tables 4 and 5, we look at how many number of vertex and edge pinchings appeared when we performed successive rounds of contractions to get different levels of details until the complexity of the model was reduced to its 10% or 1%. The current number of faces on the surface is denoted by #faces, the current level of the simplified surface by #level (the original surface is at level 0), and the number of vertex/edge pinchings in the simplified surface by #vp/#ep respectively.

k-chain	#faces	#level	#vp	#ep
1	600	15	0	1
2	638	11	0	2
3	572	10	0	4
4	624	9	2	16

Table 4: The original cow model has #faces=5804. Successive rounds of contraction of independent **chains** were performed until its detail was reduced to its 10%.

For the cow model, when k=1, 2, or 3, the number of pinchings are negligible, but it becomes large when k=4. For the bunny model, when k=3 or 4, the number of pinchings become large. This suggests that we need to address the issue of pinchings (see Section 6).

In Table 5, when k=1, 16 rounds were needed to contract the bunny model to its 10%, while 33 rounds were needed to contract to its 1%. The number of rounds are nearly doubled. For k=2, 3, and 4, comparative results were got.

Apart from looking at chain contraction, we also look at *tree contraction*, which means the contraction of k edges which form a tree instead of a chain. This should give more flexibility and hence better results. So now the algorithm will change accordingly, we grow an incomplete k-tree by the edge with the smallest quadric error from the edges incident to all degree-1 vertices of the current incomplete tree. Some experimentation results are summarized in the table below.

k-chain	#faces	#level	#vp	#ep
1	7207	16	0	0
	677	33	2	0
2	7668	11	0	3
	727	23	5	8
3	7018	10	4	37
	713	23	9	66
4	7828	9	4	103
	1494	23	10	168

Table 5: The original bunny model has #faces=69451. Successive rounds of contraction of independent **chains** were performed until its detail was reduced to its 10% or 1%(if possible).

k-tree	#faces	#level	#vp	#ep
1	7025	16	0	0
	677	33	2	0
2	7400	11	0	2
	676	23	3	8
3	6374	10	2	6
	757	19	4	12
4	6194	9	0	13
	644	18	3	18

Table 6: The original bunny model has #faces=69451. Successive rounds of contraction of independent small **trees** were performed until its detail was reduced to its 10% or 1%(if it could).

In Table 6, when k=4, the number of vertex/edge pinchings have significantly improvement compared to Table 5.

In figures 6, 7, and 8, we present some simplified results of the cow model, wheareas figures 9 and 10 for the foot model. As we can see that chain/tree contraction might create some black faces on the simplified surface especially when k is large, which means it is more prone to self-intersections.

6 Discussion & future work

We remark that like edge contraction, chain/tree contraction may produce self-intersections. For visualization purposes, self-intersections causes no harm as occluded portions of the surface will be hidden.

The handling of vertex/edge pinchings of the current implementation is quite ad-hoc. It can be proved that a linear number of independent edges can be selected whose contraction will not cause any topological change. Still, this issue needs further study.

Instead of selecting chains for contraction, selecting of small trees of k edges for contraction is also possible. It has plus and minus. One plus is that it tends to produce less vertex/edge pinchings after many rounds even though k is large. But perhaps one minus is that the encoding scheme is more complex and so less compact.

Instead of selecting chains, one might wonder whether it is possible to select linear number of independent faces/strips/patchs for contraction. Unfortunately, the answer is negative. Figure 11 shows a counter example. The maximum independent set of

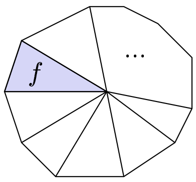

Figure 11: A star-shape surface.

faces in Figure 11 has only one face, say f.

References

[1] B. Junger and J. Snoeyink. Selecting Independent Vertices For Terrain Simplification. In *WSCG '98*, Plzen, Cz.

[2] M. Garland and P.S. Heckbert. Surface Simplification using using quadric error metrics. *SIGGRAPH '97 Proc.*, Aug. 1997. http://www.cs.cmu.edu/~garland/.

[3] H. Hoppe. View-Dependent Refinement of Progressive Meshes. *Computer Graphics (SIGGRAPH 1997 Proceedings)*, pages 189-198.

[4] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Mesh Optimization. *SIGGRAPH '93 Proc.*, pages 19-26, Aug. 1993.

[5] A. Gueziec. Surface Simplification with Variable Tolerance. 2nd Annual Intl. Symp. on Mecical Robotics and Computer Assisted Surgery '95, pages 132-139, November 1995.

[6] R. Ronfard and J. Rossignac. Full-Range Approximation of Triangulated Polyhedra. Computer Graphics Forum (Proc Eurographics '96), 15(3), Aug. 1996.

[7] M. Garland and P.S. Heckbert. Survey of Polygonal Surface Simplification Algorithm. *Tech. report, CS Dept.,Carnegie Mellon U.*, May 1997. http://www.cs.cmu.edu/~ph/.

[8] J. Rossignac and P. Borrel. Multi-Resolution 3D Approximation for Rendering Complex Scenes. *Modeling in Computer Graphics: Methods and Applications*, pages 455-465, 1993.

[9] D. Luebke and C. Erikson. View-Dependent Simplification of Arbitrary Polygonal Environments. *SIGGRAPH '97 Proc.*, August 1997.

[10] W.J. Schroeder, J.A. Zarge, and W.E. Lorensen. Decimation of Triangle Meshes. *Computer Graphics(SIGGRAPH '92 Proc.)*, 26(2):65-70, July 1992.

[11] M. Soucy and D. Laurendeau. Multiresolution Surface Modeling Based on Hierarchical Triangulations. *Computer Vision and Image Understanding*, 63(1):1-14, 1996.

[12] A. Varshney. Hierarchical Geometric Approximations. PhD Thesis, Dept. of CS, U. of North Carolina, Chapel Hill 1994. TR-050.

[13] A. Varshney, P.K. Agarwal, F.P. Brooks, W.V. Wright, and H. Weber. Generating Levels of Detail for Large-Scale Polygonal Models. *Technical Report, Dept. of CS, Duke U., Aug. 1995. CS-1995-20.* http://www.cs.duke.edu/department.html#techrept.

[14] D.G. Kirkpatrick. Optimal Search in Planar Subdivisions. *SIAM Journal on Computing*, 12:28-35, 1983.

Figure 6: The left figure is the original cow model, and the right one is the result simplified to its 10% by edge contractions only.

Figure 7: The figures from left to right are the results simplified from the original cow model to its 10% by chain contractions for $k = 1, 2, 3$, and 4 respectively.

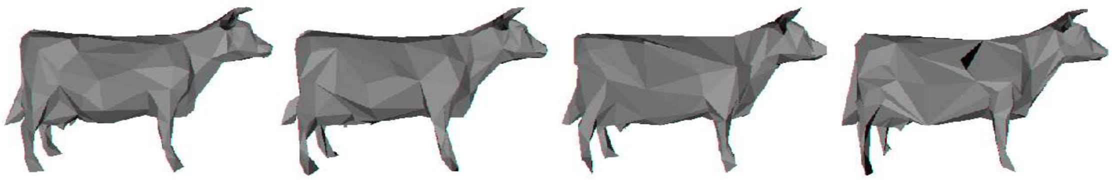

Figure 8: The figures from left to right are the results simplified from the original cow model to its 10% by tree contractions for $k = 1, 2, 3$, and 4 respectively.

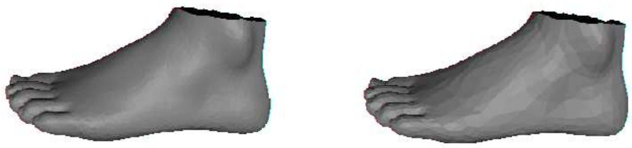

Figure 9: The left figure is the original foot model of 19971 faces, and the right one is the result simplified to its 10% by edge contractions only.

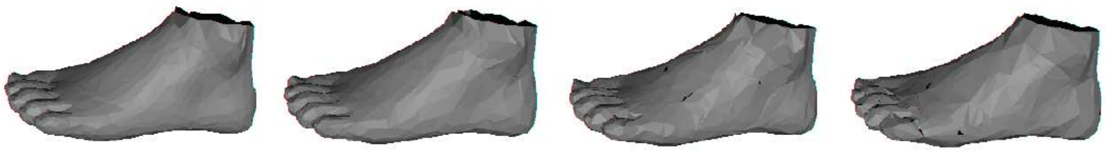

Figure 10: The figures from left to right are the results simplified from the original foot to its 10% by chain contractions for $k = 1, 2, 3$, and 4 respectively.