Eurographics Italian Chapter Conference (2007)
Raffaele De Amicis and Giuseppe Conti (Editors)

# Rewriting rules for the dual graph of a stripified CLOD mesh

Massimiliano B. Porcu, and Riccardo Scateni

Dipartimento di Matematica e Informatica, Università di Cagliari, Cagliari, Italy

**Abstract**

*A* triangular mesh *is the piecewise linear approximation of a sampled or analytical surface, when each patch is a triangle. The connectivity of the mesh can be easily represented using its* dual graph. *Each node of such a graph has at most three incident edges; if the surface is homeomorphic to a sphere, each node has exactly three incident edges.*
*Several triangular meshes, representing the same surface, with an increasing number of triangles are a representation of the surface at different* levels of detail *(LOD). When the number of triangles from one LOD to another varies continuously we call such a structure a* continuous level of detail *(CLOD) approximation of the surface.*
*Given a CLOD data structure we can extract, at each level, the mesh representing the surface and derive its dual graph. If we group the triangles forming each mesh in strips, to accelerate their rendering, we should use two colors for the dual graph's edges to distinguish between the edges linking nodes belonging to the same strip or not.*
*The main goal of this paper is to present a set of rules to recolor the dual graph of the mesh when passing from one LOD to the next and back. The operations used to change the mesh are a Vertex Split (VS) when the resolution increases, and an Edge Collapse (EC) when the resolution decreases. We can, then, use a local topological analysis to derive the rules allowing to recolor the graph, and to show that, under certain conditions, the recoloring is optimal. This allows to keep effectively an optimal triangle strip structure over the mesh, while changing its resolution.*

## 1. Introduction

In computer graphics applications there are two different requirements to comply with: to generate images of the synthetical models as close as possible to the model itself, and to keep a given pace (frames per second) when visualizing them.

In recent years, two trends evolved in parallel: on one side the graphics boards are more and more performant, making it possible to visualize larger models, on the other side the evolution in three-dimensional data acquisition still keeps the largest models available impossible to visualize at reasonable frame rates (usually no less than twenty per second).

This keeps the field of investigation on reducing the data needed to represent a given object very hot.

In this work we show the results we obtained in controlling the topology of the meshes representing a given object while changing their resolution, using the dual graphs of the meshes. We will show how can be simple to *rewrite the graph* on the basis of a collection of cases, that we studied and classified, and that are written in a look-up table. The rest of the paper is organized as follows: in section 2 we will briefly recall the characteristics of the triangle meshes used in computer graphics, in section 3 we will review how it is possible to use meshes to efficiently visualize synthetic objects, in section 4 we will show our proposal for rewriting the dual graphs of the meshes when changing their resolution, in section 5 we will analyze some results obtained on different meshes, and in section 6 we will draw our conclusions.

## 2. Meshes in Computer Graphics

The meshes typically used in computer graphics can heavily vary in size (the number of triangles they are composed of), from few thousands to several hundred of thousands. Recent advances in data acquisition, especially the broadening diffusion of tridimensional scanners, make even huge (over 1 Mtri's) if not gigantic (order of tens Mtri's) triangle meshes commonly available.

We show several examples of meshes of different dimension in Figure 1. For all these meshes, we can see that the ratio between the number of faces and the number of vertices is more or less in accordance with the Euler formula $t = 2v$ that holds exactly only when the represented object is homeomorphic to a sphere and the mesh is perfectly defined. When we use the meshes for visualization purposes we can allow to have small defects as isolated points, triangles appearing twice and such that do not compromise the overall appearance.

**Figure 1:** *From left to right three meshes representing, respectively: a teapot (4,255 vertices, 8,480 triangles); a bunny (35,947 vertices, 69,451 triangles); the "Dea madre" statue (290,449 vertices 571,806 triangles).*

From a topological point of view, manifold triangles meshes have the property that no more than two faces can insist on the same edge. In this picture, *internal* triangles will have three neighbours and *border* triangles two or one neighbours.

### 2.1. Dual Graph of a Mesh

Each triangle mesh can be represented by its *dual graph*. It is a graph in which each node is associated to a triangle of the original mesh and an edge represents an adjacency relation.

Topological properties of the dual graph reflect features of the original mesh. Especially, an important property of such a graph is that each node has, at most, three incident arcs. In case the original mesh is homeomorphic to a sphere, each node has exactly three incident arcs. An example is given in Figure 2.

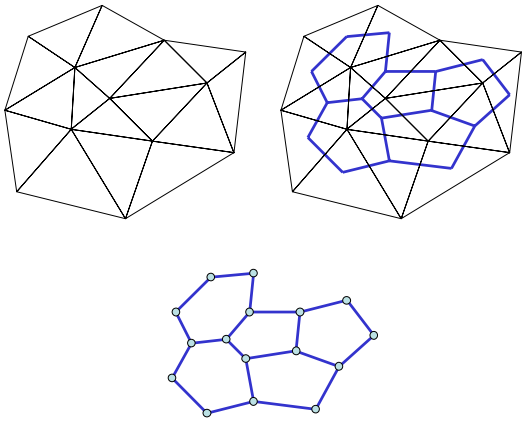

**Figure 2:** *A triangle mesh and its dual graph.*

## 3. Rendering of Tridimensional Meshes

As we anticipated in section 1, using triangular meshes to represent synthetic objects, we tipically face two contrasting requirements: on one side we want to visualize *realistic* scenes and need precise models, with a high number of faces; on the other side, we want to achieve a *fast* visualization, and meshes with big dimension are an obstacle.

Several techniques are used to make these two opposite goals converge. Two of the most widely spread are the use of *level of detail* representations [CMS98] and the construction of *strips of triangles* [AHMS96, SS97, Hop99, ESEK*00, Ste01, PS03].

### 3.1. Level of Details of a Mesh (LOD)

In order to simplify the mesh, we follow the approach first introduced by Hoppe[Hop96], introducing two transformations to modify the simplicial complex $K$ in $K'$ without changing its topology; they are called *Edge Collapse* (EC) and *Vertex Split* (VS).

An edge collapse transformation $ecol(\{v_s, v_t\})$, removes an edge from the mesh, unifying two adjacent vertices $v_s$ and $v_t$ into a single vertex $v_h$. It reduces the size of the mesh removing from it the two triangles sharing the collapsed edge. An example is shown in Figure 3.

Using an iterative mechanism, an initial mesh $M = M^n$ can be simplified into a coarser mesh $M^0$ by applying a sequence of n successive edge collapse transformations:

$$\left(M = M^n\right) \rightarrow^{ecol_{n-1}} \ldots \rightarrow^{ecol_1} M^1 \rightarrow^{ecol_0} M^0$$

Vertex split is the inverse of the edge collapse transformation: it splits a previously identified vertex restoring the two triangles sharing that vertex. (see Figure 3)

Mesh simplification process can be reversed using

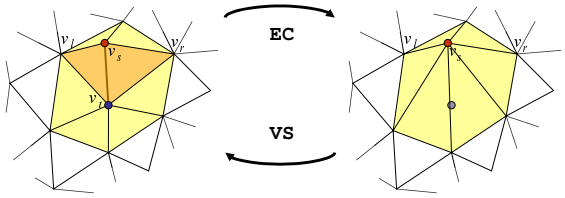


**Figure 3:** *Edge collapse and vertex split over a mesh. The vertices $v_t$, $v_s$ and the two adjacent faces $\{v_s, v_t, v_l\}$ and $\{v_t, v_s, v_r\}$ are affected by the process.*

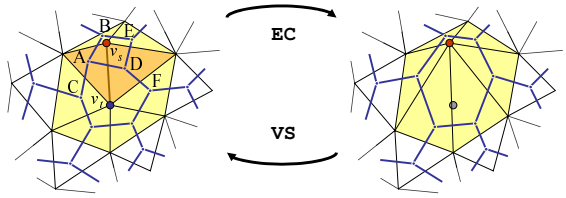


**Figure 4:** *Edge collapse and vertex split over the dual graph.*

vertex split transformation; in this way an arbitrary mesh $M$ may be represented as a simple mesh $M^0$ together with a sequence of n VS records:

$$M^0 \rightarrow^{vsplit_0} M^1 \rightarrow^{vsplit_1} \ldots \rightarrow^{vsplit_{n-1}} \left(M^n = M\right)$$

where each record is parameterized as $vsplit_i(h_i, l_i, r_i, A)$. With $A$ we indicate the position $v_s$ and $v_t$ of the affected vertices.

We call $(M^0; \{vsplit_0, \ldots, vsplit_{n-1}\})$ a *progressive mesh (PM)* [Hop96, Hop97, PH97, Hop98, PR00].

To improve the rendering performance, we can define several versions of a model at various levels of detail. With the PM representation we can capture a continuous sequence of meshes $M^0, M^1, \ldots M^n$ of increasing accuracy. From the PM structure it is possible to efficiently retrieve the LOD approximations at any desired complexity. A detailed mesh can be used when the object is close to the viewer, and coarser approximations are substituted as the object recedes. Such a kind of structure is called a continuous level of detail mesh (CLOD).

#### 3.1.1. CLOD on the dual graph

On the dual graph the EC and VS operations have always the same consequences: an EC will cancel a *loop* from the graph decreasing the total number of edges (by three) and nodes (by two), while a VS will perform the inverse modifications: two more nodes, three more edges and one more loop.

An example is shown in Figure 4. If the edge $(v_s, v_t)$ is collapsed, we eliminate the node $A$ and $D$, and we join with a direct connection nodes $C - B$ and $E - F$; the connection $A - D$ disappears as well. Again, the VS will perform the inverse modifications.

### 3.2. Strips of Triangles

A strip is a consecutive set of neighbours triangles in which each triangles share an edge with the next.

By partitioning a mesh in strips, it is possible to obtain an acceleration of the visualization process. This is due to the fact that the algorithms used by the current graphics boards are especially optimized to represent triangles packed in strips. In this context, is a very important task the study of efficient algorithms, able to partition a mesh in an optimal number of strips and thus ensure maximum efficiency.

Several papers illustrate geometrical and topological properties of a stripification [AHMS96] and many variations of algorithms to partition a triangle mesh in strips [ESV96, Cho97, SS97, Ise01, IA02, EMX02]. Unfortunately it has been proven [GJT76, AHMS96] that a problem equivalent to searching the optimal single strip (finding a Hamiltonian path on the dual graph) is an NP-complete problem, thus the stripification process is always based on local heuristics.

Two approaches for finding a stripification on the mesh's dual graph have been proposed: one is to compute a spanning tree on the dual graph, partition it into triangle strips, and then concatenate these strips into larger ones [XHM99]; the second one is the so-called *tunnelling* algorithm and it uses a topological operator on dual graph of a mesh seen as a bipartite graph [Ste01, PS06].

To represent in a proper way the triangle strips in the dual graph of a mesh, we need to *color* the graph edges in two different ways:

**solid edges** linking nodes associated to triangles in the same strip;

**dashed edges** linking nodes associated to adjacent triangles not belonging to the same strip.

In every node there are, at most, two incident solid edges. Figure 5 shows a stripified mesh and its dual graph colored accordingly.

### 3.3. Strips and CLOD

Both stripification and CLOD improve the rendering efficiency of triangle meshes, but usually do not work well together when applied to the same triangle mesh. Updating the level of detail after a stripification has been performed, in fact, it makes the EC and VS operations *break* the strips, destroying the optimality obtained. Moreover, a local reparation is not usually possible using standard stripification algorithms, and the

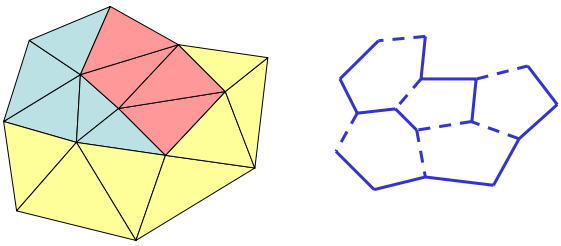

**Figure 5:** *A stripified mesh and its dual graph colored accordingly.*

idea to perform a new stripification from scratch for each level of detail is not practically feasible because too much time expensive.

## 4. Rewriting rules

Our goal is to keep a stripification composed by the lowest possible number of strips, when inserting new triangles in a mesh, performing a VS operation.

We define a $k$-vertex as a vertex belonging to $k$ triangles. In the dual graph, the $k$ triangles form a $k$ loop, that is a path of length $k$ with no nodes inside. We can classify the VS operation according to the cardinality of the vertex, so we can say that we have a $k$-VS.

We can see in Figure 6 all the possible changes in the graph when performing a 3-VS, 4-VS, a 5-VS, a 6-VS, a 7-VS or a 8-VS. All in all the cases are limited, as you can see we enumerated only nineteen different cases, and as you will see in section 5 these cases will cover almost any possible configurations in real meshes. More formally eight is the upper bound imposed by the local curvature when we deal with *well conformed* meshes. In any case we can reduce the order of a vertex performing a local operation, different from VS and EC, called *edge flip*.

The real enumeration problem arises when we want to consider a bi-colored graph, as the one we use to represent a stripified mesh. For a general $k$-vertex we have $k$ edges forming the loop and $k$ edges incident on the nodes on the loop apart the ones in the loop. We have, thus, to take in account, when performing a $k$-VS, $2k$ edges each of them assuming two possible states. For a $k$-VS we have $2^{2k}$ different possible configurations to examine for each of the subcases listed in Figure 6.

Currently we have exhaustively examined the $256+256$ configurations occurring in a 4-VS (the 3-VS is trivial). We can reduce them to eighteen $(9+9)$ applying rotational and specular simmetry rules. All the configuration are listed in Figure 7.

We are seeking a systematic approach, other from analyzing directly all the cases, to reduce and classify all the other $k$-VS up to 8-VS. We are already sure

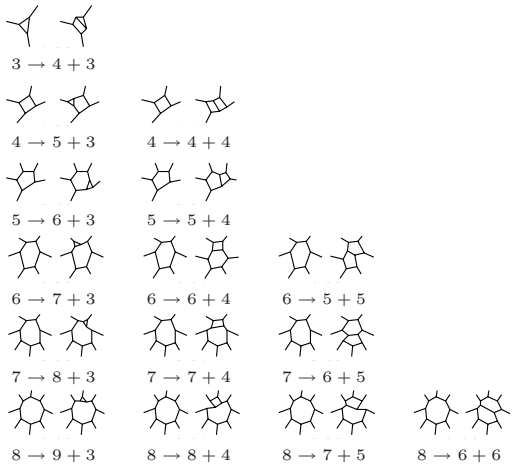


**Figure 6:** *All the possible graph rewriting cases when splitting, from the topmost row, a 3-vertex (1 subcase), a 4-vertex (2 subcases), a 5-vertex (2 subcases), a 6-vertex (3 subcases), a 7-vertex (3 subcases), and a 8-vertex (4 subcases).*

that not all the configurations are rewritable without introducing isolated triangles. In Figure 8 we can see an example of a 6-VS introducing an isolated triangle. Theoretically this is acceptable, since it can be thought as a strip composed by a single element. Practically, introducing too many isolated triangles would disrupt the advantages of using triangle strips.

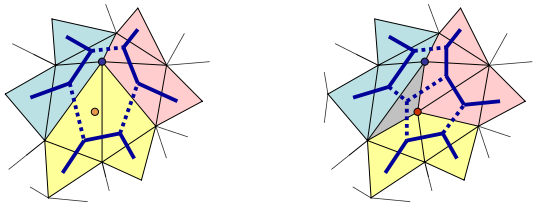

**Figure 8:** *A configuration where the graph recoloring after a VS leaves an isolated triangle (marked grey).*

To solve this problem we elaborated a strategy involving the invocation of the tunneling algorithm on the finer mesh when the number of isolated triangles exceed a given threshold.

## 5. Results and discussion

We first examined the cardinality of the vertices of a set of sample meshes. Some of them are typical benchmarks meshes, while other are meshes we reconstructed from three-dimensional scans of local cultural heritage manufacts.

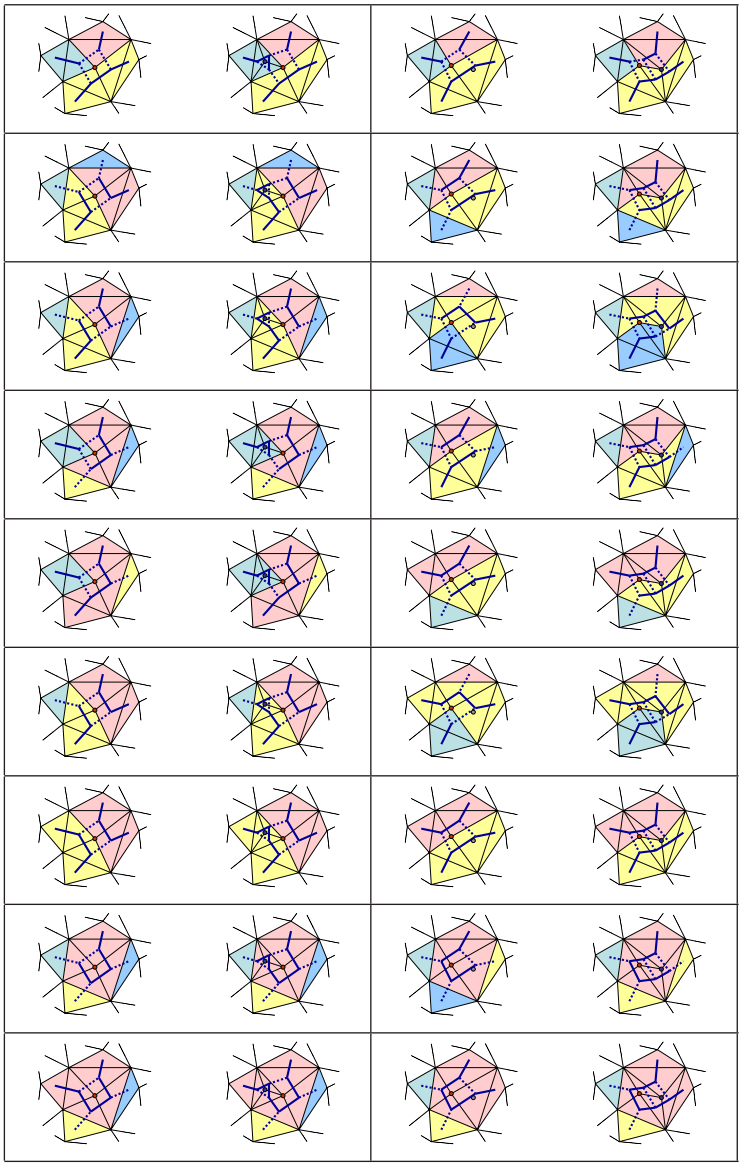

**Figure 7:** *The rewriting rules to apply when performing a 4-VS on a bi-colored graph. In each cell, on the left there is the configuration before the VS (the vertex to be split is marked in red), on the right the configuration after the VS (the new inserted vertex is marked in blue).*

In table 1 we list the values obtained for the three meshes representing cultural heritage manufacts (see in Figure 9 the plot of the percentage values). The number of vertices for the three meshes is comparable (in the order of hundreds of thousands) and so are the results of the analysis.

In table 2 we list instead the values obtained for three meshes typically used for benchmarking purposes (in Figure 10 the plot). In this case we can see how the size and shape of the mesh, increases its *regularity*, that is it concentrates the number of vertices around the canonical value of six. In fact, a completely regular mesh of curvature 0 consists only of equilateral triangles and there would be only 6-vertices in it.

If we compare, instead, the percentage of VS resulting in introducing an isolated triangle they are 52.7% in the Teapot mesh, passing from the lowest to the highest LOD, it is 51.5% in the Bunny mesh and 59.5% in the "*Capotribù*" mesh. This is the consequence of

| Card. | "*Arciere*" 111,613 vertices | | "*Capotribù*" 152,591 vertices | | "*Dea madre*" 500,002 vertices | |
|---|---|---|---|---|---|---|
| | **Num** | **%** | **Num** | **%** | **Num** | **%** |
| 1 | 544 | 0.49 | 435 | 0.29 | 3 | ∼0.00 |
| 2 | 855 | 0.77 | 1,480 | 0.97 | 0 | 0.00 |
| 3 | 1,079 | 0.97 | 2,752 | 1.80 | 2,000 | 0.40 |
| 4 | 3,999 | 3.58 | 12,451 | 8.16 | 34,509 | 6.90 |
| 5 | 26,343 | 23.60 | 38,166 | 25.01 | 129,164 | 25.83 |
| 6 | 52,602 | 47.13 | 50,778 | 33.28 | 181,303 | 36.26 |
| 7 | 20,479 | 18.35 | 32,355 | 21.20 | 109,814 | 21.96 |
| 8 | 5,315 | 4.76 | 11,398 | 7.47 | 36,059 | 7.21 |
| 9 | 377 | 0.34 | 2,440 | 1.60 | 6,418 | 1.28 |
| 10 | 19 | 0.02 | 298 | 0.20 | 680 | 0.14 |
| 11 | 1 | ∼0.00 | 32 | 0.02 | 48 | 0.01 |
| 12 | | | 4 | ∼0.00 | 4 | ∼0.00 |
| 13 | | | 2 | ∼0.00 | | |

**Table 1:** *Vertices classification for three meshes representing small statues conserved in National Archaeological Museum of Cagliari. The percentage values are rounded to the second digit.*

| Card. | Dragon 437,645 vertices | | Bunny 34,834 vertices | | Teapot 4,255 vertices | |
|---|---|---|---|---|---|---|
| | **Num** | **%** | **Num** | **%** | **Num** | **%** |
| 1 | 775 | 0.18 | 2 | 0.01 | | |
| 2 | 1073 | 0.25 | 31 | 0.09 | 2 | 0.05 |
| 3 | 2244 | 0.51 | 111 | 0.32 | 28 | 0.66 |
| 4 | 39545 | 9.04 | 438 | 1.26 | 16 | 0.38 |
| 5 | 119944 | 27.41 | 3875 | 11.12 | 61 | 1.43 |
| 6 | 141423 | 32.31 | 26165 | 75.11 | 4077 | 95.82 |
| 7 | 83482 | 19.08 | 3888 | 11.16 | 51 | 1.20 |
| 8 | 34087 | 7.79 | 298 | 0.86 | 17 | 0.40 |
| 9 | 10931 | 2.50 | 22 | 0.06 | 2 | 0.05 |
| 10 | 3145 | 0.72 | 3 | 0.01 | 1 | 0.02 |
| 11 | 770 | 0.18 | 1 | ∼0.00 | | |
| 12 | 170 | 0.04 | | | | |
| 13 | 44 | 0.01 | | | | |
| 14 | 10 | ∼0.00 | | | | |
| 15 | 2 | ∼0.00 | | | | |

**Table 2:** *Vertices classification for three meshes typically used for benchmarking.*

not having a complete set of robust rewriting rules even for the 5- and 6-vertices.

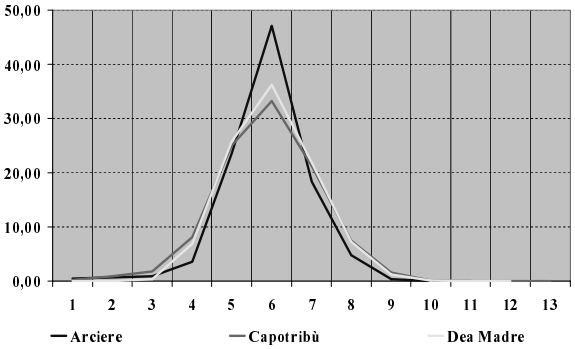


**Figure 9:** *Plot of the percentage values reported in table 1.*

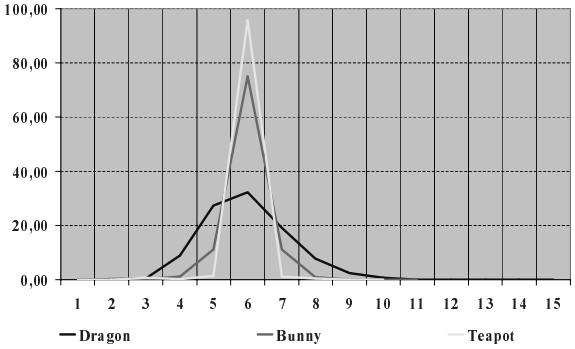


**Figure 10:** *Plot of the percentage values reported in table 2.*

## 6. Conclusions and future work

We showed in this work how a systematic approach to the analysis of the dual graph of a triangle mesh can help in accelerating the time needed to update the stripification of the mesh when changing its resolution using a CLOD structure. The analysis we performed is limited to a subset of the total number of cases that occur when dealing with *real* meshes. This is not anyway a limit to the fast visualization and adaptation of the meshes since we use a complementary strategy to *repair* the stripification. From the results we presented we can see that the analysis on the $k$-VS should arrive to the coverage of, at least, the 8-VS.

To reach this goal we need to better describe the local portions of the graph involved in the VS operation. This will bring to automatically identify the similar patterns among the cases to analyze (referring to specular and rotational simmetry). The further, and much more challenging, step would be to derive a set of rules that would: rewrite the graph to keep the stripification at its best looking only to the $2k$ edges directly affected by a $k$-VS, enlarge the analysis to the edges at distance 2 from the nodes on the loop to try to rewrite the graph without isolated triangles when the local analysis is insufficient.

**Acknowledgements**

The "*Capotribù*", "*Arciere*", and "*Dea madre*" datasets were obtained from tridimensional scans of manufacts exposed at the Museo Archeologico Nazionale in Cagliari. We are indebted to its director, Carlo Tronchetti, for letting us use these digital

data and to the VCG of the ISTI-CNR in Pisa for the hardware and software used in the acquisition and reconstruction.

**References**

[AHMS96] Arkin E. M., Held M., Mitchell J. S. B., Skiena S. S.: Hamiltonian triangulations for fast rendering. *The Visual Computer 12*, 9 (1996), 429–444.

[Cho97] Chow M. M.: Optimized geometry compression for real-time rendering. In *IEEE Visualization '97* (Nov. 1997), pp. 346–354.

[CMS98] Cignoni P., Montani C., Scopigno R.: A comparison of mesh simplification algorithms. *Computers & Graphics 22*, 1 (Feb. 1998), 37–54.

[EMX02] Estkowski R., Mitchell J. S. B., Xiang X.: Optimal decomposition of polygonal models into triangle strips. In *Proceedings of the eighteenth annual symposium on Computational geometry* (2002), ACM Press, pp. 254–263.

[ESEK*00] El-Sana J., Evans F., Kalaiah A., Varshney A., Skiena S., Azanli E.: Efficiently computing and updating triangle strips for real-time rendering. *Computer-Aided Design 32*, 13 (Oct. 2000), 753–772.

[ESV96] Evans F., Skiena S. S., Varshney A.: Optimizing triangle strips for fast rendering. In *IEEE Visualization '96* (Oct. 1996), pp. 319–326.

[GJT76] Garey M. R., Johnson D. S., Tarjan R. E.: The planar hamiltonian circuit problem is np-complete. *SIAM Journal of Computing 5*, 4 (Dec 1976), 704–714.

[Hop96] Hoppe H.: Progressive meshes. In *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques* (1996), ACM Press, pp. 99–108.

[Hop97] Hoppe H.: View-dependent refinement of progressive meshes. In *Proceedings of SIGGRAPH 97* (Aug. 1997), Computer Graphics Proceedings, Annual Conference Series, pp. 189–198.

[Hop98] Hoppe H.: Efficient implementation of progressive meshes. *Computers & Graphics 22*, 1 (Feb. 1998), 27–36.

[Hop99] Hoppe H.: Optimization of mesh locality for transparent vertex caching. In *Proceedings of SIGGRAPH 99* (Aug. 1999), Computer Graphics Proceedings, Annual Conference Series, pp. 269–276.

[IA02] Isenburg M., Alliez P.: Compressing polygon mesh geometry with parallelogram prediction. In *Proceedings of the conference on Visualization '02* (2002), IEEE Press, pp. 141–146.

[Ise01] Isenburg M.: Triangle strip compression. *Computer Graphics Forum 20*, 2 (2001), 91–101.

[PH97] Popović J., Hoppe H.: Progressive simplicial complexes. In *Proceedings of SIGGRAPH 97* (Aug. 1997), Computer Graphics Proceedings, Annual Conference Series, pp. 217–224.

[PR00] Pajarola R., Rossignac J.: Compressed progressive meshes. *IEEE Transactions on Visualization and Computer Graphics 6*, 1 (Jan–Mar 2000), 79–93.

[PS03] Porcu M. B., Scateni R.: An iterative stripification algorithm based on dual graph operations. In *Proceedings of the EuroGraphics conference 2003 (short presentations)* (2003), The Eurographics Association, pp. 69–75.

[PS06] Porcu M. B., Scateni R.: Partitioning meshes into strips using the enhanced tunnelling algorithm (ETA). In *Proceedings of VriPhys 2006* (2006), EuroGraphics, pp. 61–70.

[SS97] Speckmann B., Snoeyink. J.: Easy triangle strips for TIN terrain models. In *Canadian Conference on Computational Geometry* (1997), pp. 239–244.

[Ste01] Stewart A. J.: Tunneling for triangle strips in continuous level-of-detail meshes. In *Graphics Interface 2001* (June 2001), pp. 91–100.

[XHM99] Xiang X., Held M., Mitchell J. S. B.: Fast and effective stripification of polygonal surface models. In *Proceedings of the 1999 symposium on Interactive 3D graphics* (1999), ACM Press, pp. 71–78.