the Visual Computer manuscript No.
(will be inserted by the editor)

Efficient View-Dependent Refinement of 3D Meshes using $\sqrt{3}$-Subdivision

Pierre Alliez[1,2], **Nathalie Laurent**[1], **Henri Sanson**[1], **Francis Schmitt**[3]

[1] France Télécom R&D DIH HDM
4, rue du Clos Courtel BP 59
35512 Cesson-Sévigné Cedex, France
e-mail: {nathalie.laurent,henri.sanson}@rd.francetelecom.fr
[2] University of Southern California
e-mail: alliez@usc.edu
[3] Ecole Nationale Supérieure des Télécommunications
46, rue Barrault
75634 Paris Cedex 13, France
e-mail: francis.schmitt@enst.fr

Received: date / Revised version: date

Abstract In this paper we introduce an efficient view dependent refinement technique well-suited to adaptive visualization of 3D triangle meshes on a graphic terminal. Our main goal is the design of fast and robust smooth surface reconstruction from coarse meshes. We demonstrate that the $\sqrt{3}$ subdivision operator recently introduced by Kobbelt offers many benefits, including view-dependent refinement, removal of polygonal aspect and highly tunable level of detail adaptation. In particular, we propose a new data structure that requires neither edges nor hierarchies for efficient and reversible view-dependent refinement. Results on various 3D meshes illustrate the relevance of the technique.

Key words Surface reconstruction – Adaptive visualization – View dependent refinement – Subdivision surfaces – Device driven refinement

1 Introduction

Subdivision surfaces define smooth surfaces as the limit of a sequence of successive refinement steps applied to a base mesh. This approach offers a number of benefits, and has been used in the context of geometry compression, animation, editing, scalability and adaptive rendering [1,2]. Subdivision is a key issue for the reconstruction of surfaces, and brings enough robustness to reach adaptive rendering at an interactive rate. The recursive structure of the subdivision paradigm naturally addresses the scalability feature required for the adaptation to graphic capabilities of a terminal. Furthermore, the distribution of the levels of details can be tuned by considering the additional adaptation to the user viewpoint.

While much effort has been spent on the design of geometric and topological laws that drive the subdivision process [3–7], less effort has been spent for real-time rendering of highly detailed surfaces generated through adaptive refinement. This paper represents a step towards realtime performance, and introduces a data structure that allows fast and robust reversibility of the refinement process and that requires only a small additional amount of memory. We carefully take the camera viewpoint into account in order to define precisely the areas of interest on the terminal and to generate on them well-suited levels of details.

Two limiting factors seriously impede the perceptual quality at the rendering stage:

- The polygonal aspect is obviously the main defect when a smooth surface is poorly described for some memory and transmission constraints. This defect is highly noticeable along the silhouette areas as defined in Fig. 1;
- The bi-linear Gouraud or Phong shading interpolation generates a piecewise rendering effect which is noticeable for large triangles in the image space.

These two limiting factors lead us to elaborate a refinement strategy through the subdivision paradigm. The main goal of subdivision being the smoothness of the limit surface, a subdivision approach appears then clearly well-suited to remove the visual polygonal aspect of silhouette areas and to solve the shading interpolation problem. For a given viewpoint, we first only consider the areas located in the vision frustum, i.e. in the field of vision of the camera, and then refine the subset of the triangles belonging to visible areas facing the camera. We also pay a special attention to the silhouette areas in order

Send offprint requests to: Nathalie Laurent
Correspondence to: nathalie.laurent@rd.francetelecom.fr

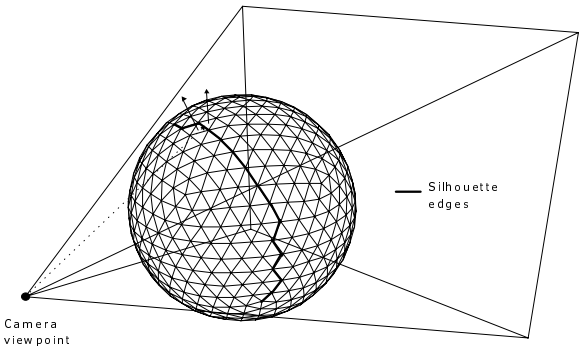

Fig. 1 Silhouette definition : for a given viewpoint and a 3D 2-manifold mesh, the silhouette is the subset of edges having two opposite adjacent faces, one frontfacing and the other one backfacing.

to remove their polygonal aspect. Therefore we separate the problems into two tasks: the *extraction* of the areas of interest, and the *adaptive refinement* process.

Section 2 reviews related work, while section 3 overviews the general principle of the proposed method. The extraction and subdivision processes are described in sections 4 and 5 respectively. An efficient implementation is proposed in section 6. Section 7 illustrates the numerous benefits of the scheme with various mesh samples. In section 8, concluding remarks and future work are presented.

2 Related works

In the context of visualization and telecommunication, subdivision surfaces turn out to be very relevant since they address the numerous challenges of ever more complex 3D-based contents having to be stored, transmitted and rendered on heterogeneous networks and terminals. These challenges are addressed with geometry compression, be it progressive [8,9] or not [10], eventually combined with adaptive visualization [11]. Although a progressive compression scheme is well-suited to an adaptation to network constraints, the presented method also allows us to increase the mesh complexity in a scalable way until the graphics capabilities of the terminal are reached. The complexity of the mesh is locally adapted depending on the camera viewpoint this time, where it really matters. It is useful to recall that a telecom operator is mostly interested in the *perceived quality* of the data for the final user. For him the global bit count of a given 3D content is often less important than the delay before displaying the first image.

We now review in the next two subsections existing work related to the extraction of the regions of interest and to subdivision surfaces.

2.1 Extraction

As already mentioned, we are mainly concerned with the efficient extraction of the silhouette. Indeed, the silhouette is a very important geometrical feature of an object since it plays a key part in the visual perception process [12], describes the global shape of an object, and often corresponds to the first pencil strokes of a draft. It is thus used in numerous research fields:

- Computer vision: the silhouette is often used for shape recognition applications or for 3D reconstruction from multiviews;
- Line drawing: the silhouette separating hidden areas from visible ones, it is thus naturally used for hidden line removal tasks;
- Hybrid rendering: fast silhouette display is crucial in order to reduce the rendering calculation costs. A complex geometry is replaced by a coarse mesh on which a precomputed normal map is mapped. The illusion is effective, except on the silhouette since the poorly described geometry is highly noticeable on these areas. Gu *et al.* [13] thus define by preprocessing a set of silhouette edges in order to clip the geometry. A recent work by Sander *et al.* [14] defines two progressive convex envelopes for the coarse geometry, a normal map for the geometry illusion and an edge tree for silhouette clipping. Results are eloquent, especially when the silhouette edges are anti-aliased during the clipping stage;
- Artistic rendering: silhouette is extracted and superimposed on a shaded mesh in order to obtain attractive visual effects [15–19];
- Computational geometry: authors propose a very efficient method that allows output sensitive [20] and perspective accurate [21] extraction of silhouette edges. Benichou and Elber [20] build a preprocessed data structure containing a set of geodesic lines deduced from the mesh edges, then convert a camera request into an intersection problem between a plane and a set of segments.

The silhouette extraction is not sufficient, since we also have to extract frontfacing areas to rebuild smooth surfaces in a view-dependent manner. In order to address this issue, Zhang and Hoff [22] propose a fast backface culling method using normal masks.

2.2 Subdivision surfaces

In this paper, we assume that a mesh is transmitted over the network using a given compression scheme [23,10,24,8,25,9]. We thus only consider visualization of the received data, and we assume that the data is a triangle mesh. This stage being independent of the transmission process, we thus only focus on the rendering part of the *adaptation* challenge. Such a technique is also attractive for an Internet-based service since the user is able to see an appealing image very early in the connection when the initial mesh has been transmitted. We thus review the reconstruction methods acting on general triangular meshes, and especially focus on those which allow powerful and adaptive smooth surface reconstruction for a *visualization* purpose.

The first basic principle for subdivision was developed by Chaikin [26] in 1974: a smooth curve is generated through successive refinement of a polygonal curve defined with a set of control nodes, by inserting new nodes in between the old ones. Catmull and Clark [3] and Doo and Sabin [4] extended this principle to surfaces in 1978. A smooth surface is defined by infinite subdivision of an initial control mesh, subdivision still consisting in inserting new elements (i.e. nodes, faces and associated edges). Much effort has been made to ensure that subdivision processes converge towards a smooth limit surface. The main advantage of subdivision comes from its ability to rebuild a geometry in a very robust way, while only requiring an initial coarse mesh used as a control model, and encoded with a low bit count [10]. This last feature is very appealing for telecommunication, since networks capabilities are highly variable and may encounter some bottlenecks. In addition, subdivision surfaces are intrinsically scalable, allowing us to address the *Quality of Service* issue.

Subdivision surface schemes can be classified [1] according to whether they approximate or interpolate [27] their control nodes. They turn out to be very efficient in the case of surface reconstruction [5, 6, 28–32]. Interpolation schemes leave original control nodes unchanged, while approximation ones apply successive filtering onto the node positions. As the preservation of the positions is not a strong constraint for our targeted applications, we are thus free to choose any of these methods. Nevertheless, we choose the approximating schemes because of their simplicity and their resulting visual smoothness.

The simplest scheme suited to irregular triangular meshes is the Loop subdivision scheme, designed by Charles Loop [5]. Like most subdivision schemes, this one uses the uniform 1-to-4 subdivision that inserts a new vertex on each edge of the control mesh. It is based on a three-directional quartic box spline basis function, and has been derived by Hoppe *et al.* [6] in order to rebuild piecewise smooth surfaces. This last method is well-suited to a global reconstruction, and would suppress the polygonal aspect of silhouette areas while preserving (and even rebuilding) the geometric singularities such as corners, creases and dart vertices. In addition, and since 1997, Hoppe has strongly advocated the relevance of silhouette areas for a visualization purpose, and extended his *progressive meshes* in order to adapt geometric complexity to current viewpoint [11]. This last method, using the edge collapse for the transitions between levels of details, thus results in a forest of edge collapses which appear to be CPU-consuming to get through. We prefer a refinement through subdivision which is more efficient for the targeted applications.

Let us recall our main challenge: we aim to rebuild a smooth surface from an irregular triangular base mesh, while keeping the mesh complexity below the graphic capabilities of a terminal. Required features are:

- Efficiency: the refinement speed must coincide with an interactive rendering rate;
- Local refinement: triangle count being the main limiting factor of current graphic terminals, additional information has to be generated only where it is relevant (i.e. on visible frontfacing regions and on silhouette areas), while keeping a low triangle count anywhere else in a consistent manner;
- A robust and reversible subdivision operator, the viewpoint being highly variable during visualization.

If we refine a mesh in the highest localized manner (i.e. in a single face) using the conventional 1-to-4 subdivision operator, while maintaining the mesh consistency, a rapid increase in the number of mesh faces is observed. By using the 1-to-3 uniform subdivision operator a third times slower increase is observed (see Fig. 3). However, as each new vertex inserted by the 1-to-3 subdivision operator has a valence of 3, this would rapidly generate a valence imbalance and a highly degenerated mesh as shown in Fig. 2. But the $\sqrt{3}$ subdivision operator recently introduced by Kobbelt [7] judiciously integrates the edge swapping operator in order to balance the valences (Fig. 5), while providing a powerful relaxation scheme that guarantees a C^2-differentiability everywhere on the limit surface, except on a finite number of points where only C^1-differentiability is guaranteed. We thus choose this scheme in order to derive an efficient view-dependent refinement scheme, also well-suited to piecewise smooth surfaces.

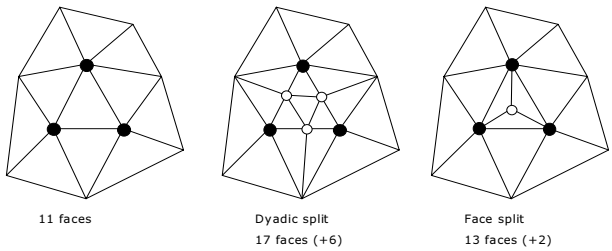

Fig. 3 Left: the original mesh we aim to refine in a very localized manner. Center: the well-known dyadic split (also named edge bisection) needs supplementary edges to maintain the mesh consistency, thus increasing by 6 the number of mesh faces. Right: the 1-to-3 uniform face subdivision allows us to build a slower refinement scheme, increasing only by two the number of mesh faces.

3 Proposed method layout

Our challenge consists in maximizing the ratio *perceptual quality / number of faces*, while keeping the number of faces below the capabilities of the terminal. The proposed technique iteratively applies an adaptive refinement through the $\sqrt{3}$ subdivision operator. One can distinguish two processes which will be detailed in the next two sections: extraction of areas of interest, and local refinement by an iterative surface subdivision. Each subdivision iteration is composed of the following three successive steps which will be detailed in section 6:

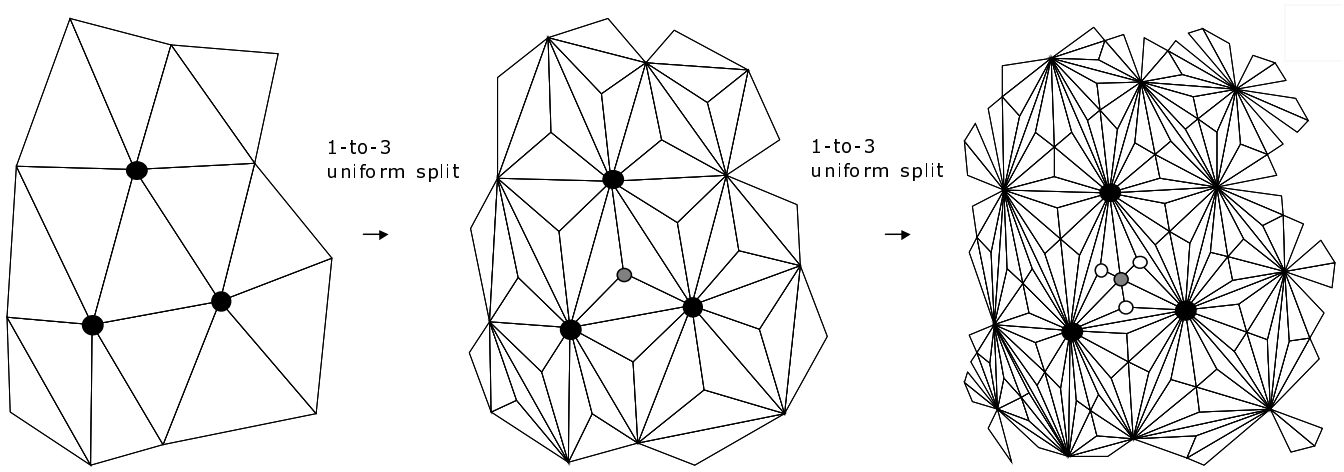

Fig. 2 A sequence of 1-to-3 uniform splits without edge swapping would imbalance the valence of the vertices and degrade the triangle aspect ratio, while keeping the initial edges unchanged.

– Mesh subdivision;
– Geometrical filtering of the vertex positions in order to obtain the desired surface smoothness;
– Edge swapping in order to increase the triangle aspect ratio and to obtain smooth connections between refined areas and their surrounding.

We thus obtain an efficient surface reconstruction, while an optimized data structure presented in section 6.3 allows the reversibility of the refinement process and provides the robustness required for interactive visualization. A pseudo-code of the full refinement sequence will be given at the end of section 6.

4 Extraction of areas of interest

The aim of the extraction process is to select visually relevant faces. We first define the viewpoint of the user by the optical center and the vision frustum formed by this center and the corners of the display window. The areas of interest thus concern:

– The set of faces located inside the vision frustum;
– The frontfacing triangles, i.e. triangles whose outward half space pointed by their normal contains the camera viewpoint;
– The silhouette areas derived from the silhouette edges definition shown in Fig. 1. Silhouette areas are formed by the set of triangles adjacent to the vertices belonging to the silhouette edges. The triangle strips built from extracted silhouette edges are shown in Fig. 4.

5 Local refinement by surface subdivision

Our aim is to locally refine the mesh on the areas of interest, while preserving a low triangle count elsewhere in order to stay below the manageable graphics budget. As explained in section 2, we have chosen the $\sqrt{3}$ subdivision operator (Fig. 5) in order to ensure a highly progressive refinement and harmonious transitions between different levels of detail (Fig. 6). The silhouette is limited to triangle strips (Fig. 4), therefore the refinement process is very localized since the operator is applied alternately with the extraction of the silhouette areas, which gradually narrow during the adaptive refinement (see Fig. 11 in the results section).

The smoothness properties of the surface reconstructed with a given subdivision operator is dependent on the associated filtering process described in Fig. 7. According to the last analysis from Kobbelt [7], the $\sqrt{3}$ subdivision operator guarantees a C^2-differentiability everywhere, except at extraordinary vertices with valence $\neq 6$ where it is C^1. The $\sqrt{3}$

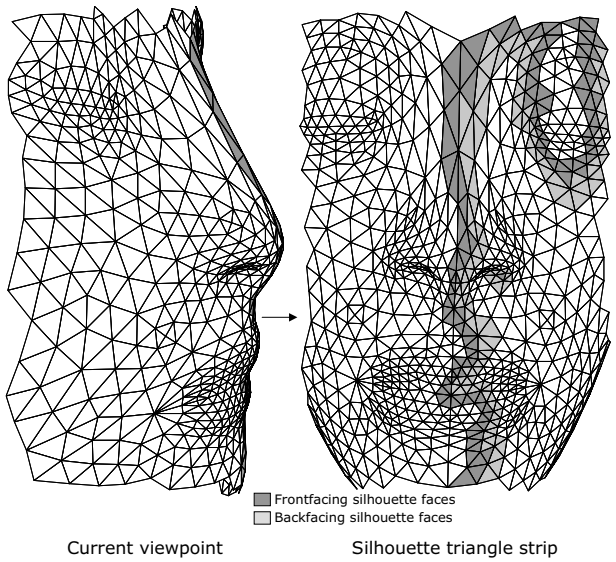

Fig. 4 The silhouette areas form triangle strips on which the $\sqrt{3}$ subdivision scheme is applied. Frontfacing and backfacing areas belonging to the silhouette triangle strips (dark and light grey respectively) are separated by the silhouette edges.

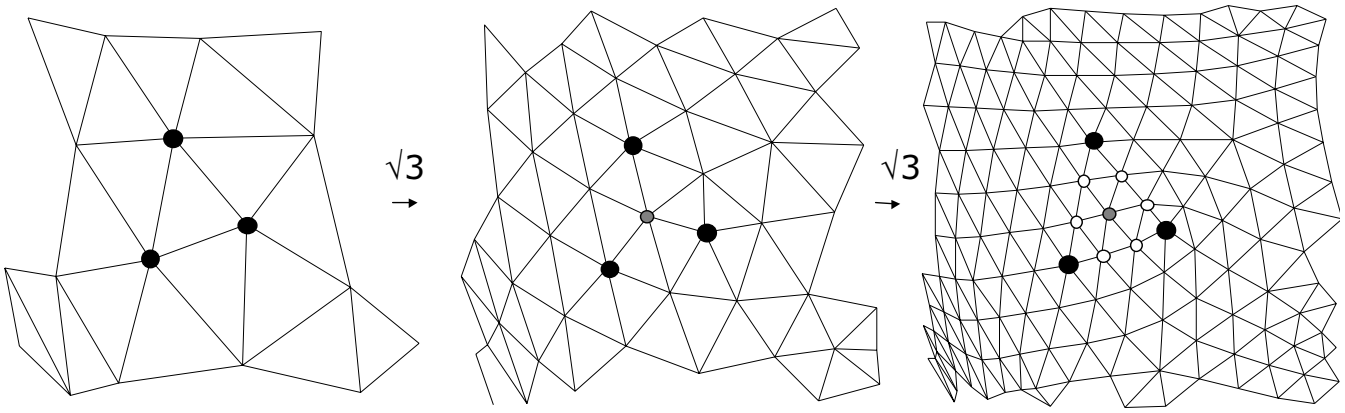

Fig. 5 Applying the $\sqrt{3}$ subdivision operator twice generates a uniform 1-to-9 refinement with trisection of every original edge [7]. Note that the well-known 1-to-4 operator [5], when applied twice, increases the mesh complexity by a factor 16, instead of 9, which is a drawback for the fine tuning of the levels of detail.

operator inserts a new vertex v_b at the barycenter of each triangle defined by the initial vertices $\{v_i, v_j, v_k\}$, then relaxes each initial vertex v (i.e. $v_i, v_j, v_k, \ldots$) according to the following law:

$$v := (1 - \alpha_n)v + \alpha_n \frac{1}{n} \sum_{a=1}^{n} v_a, \tag{1}$$

where n denotes the valence of vertex v in the previous mesh before subdivision, $v_1, \ldots, v_n$ denote the corresponding initial vertices adjacent to v, and α_n is given by the following formula:

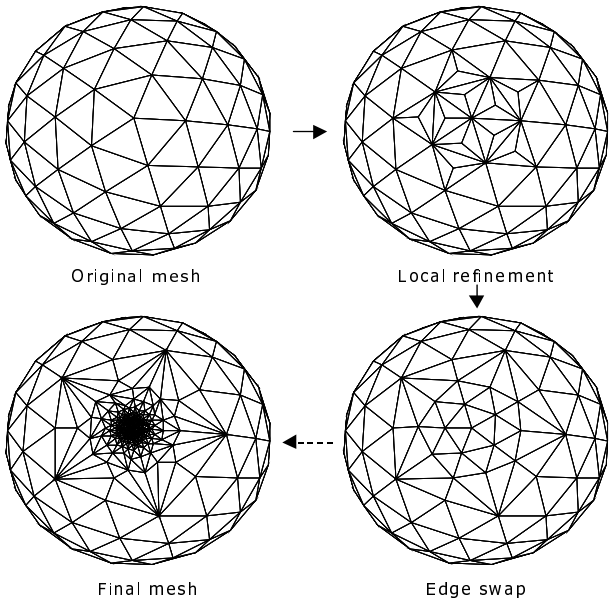

Fig. 6 The $\sqrt{3}$ subdivision operator is chosen because of its capability to locally increase the geometric density from a coarse region to a highly detailed one. The automatic preservation of consistency, associated with an amazingly simple implementation, makes this operator an excellent choice for our local surface refinement scheme.

$$\alpha_n := \frac{4 - 2\cos(\frac{2\pi}{n})}{9}. \tag{2}$$

Notice that the α_n weighting parameter only depends on n, the valence of vertex v. The choice of α_n according to this formula guarantees the convergence to a smooth limit surface [7].

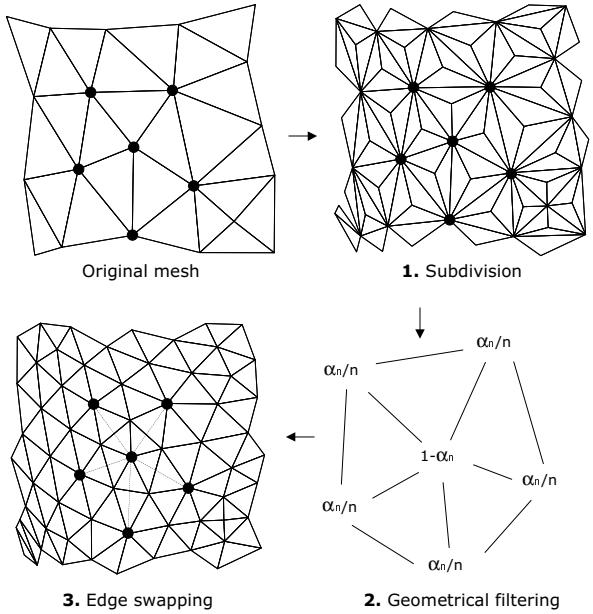

Fig. 7 The $\sqrt{3}$ operator consists of three steps: subdivision, geometrical filtering and edge swapping. The geometrical filtering step is only applied on the initial vertices before subdivision (the black dots on this example): each vertex position is updated by averaging it with the crown of its initial neighboring vertices according to the symmetric convolution mask shown.

6 Efficiency of the implementation

The efficiency of the view-dependent refinement process is closely linked with execution speed and memory cost, while the data structure influences its performances and its robustness. Our goal consisting in the building of a reconstruction engine that generates several thousands of faces by second during the visualization process, robustness is a key feature. Therefore we limit the numbers of data types and required predicates. The following sections detail how we optimized the extraction stage and the corresponding data structure.

6.1 The extraction sequence

In order to reduce the processing time of the extraction, the items belonging to the areas of interest are sought through a sequence of three successive steps:

1. The vision frustum is defined by the display window corners and the camera center. We only retain for the refinement process the set of faces adjacent to the vertices located within the vision frustum;
2. Faces orientation: we only retain the frontfacing areas which are deduced from a straight orientation test with the camera center. Such a search process is limited to the set of faces located within the vision frustum;
3. The silhouette area search process is limited on the set of faces selected from the two previous steps. We then retain all the faces adjacent to the vertices belonging to the silhouette edges. They form a set of triangle strips along the silhouette edges (Fig. 4).

The first iterations of refinement are applied on the frontfacing and silhouette items located within the vision frustum, while the following ones concern a tiny subset of the triangles belonging to the refined silhouette strips. During the visualization stage, the camera and the objects may be transformed by scalings, translations, and rotations, which can be the result of a combination of several transformations. For efficiency, we avoid the calculation of each vertex transformation in the camera reference model and prefer to compute an inverse transformation expressed only for the camera. A mesh rotation around its own center is converted into an inverse camera rotation around the same center, a mesh scaling is converted into a camera forwards to or backwards from the mesh center, and so on. As the camera vision frustum is defined by only five points (the optical center and the four window corners), the calculating time is thus strongly reduced.

6.2 Propagation by inheritance

The faces located in the vision frustum are searched in an exhaustive way only at the first extraction. Indeed, the new faces created through the subdivision paradigm generally lie within the vision frustum by inheritance of their parent properties. The same inheritance is applied for frontfacing areas. The level of refinement may be controlled by i) a maximum number of refinement iterations, ii) a geometrical magnitude measured in pixels on the image plane, thus resulting in a device-driven refinement, iii) a surface curvature criterion in order to avoid refinement of nearly planar areas, or iv) by any combination of these criteria.

The first silhouette extraction only applies to the set of seed faces located within the vision frustum and oriented towards the camera. This heuristic runs fine in practice because, as one of the two faces adjacent to a silhouette edge is frontfacing, this face is then a good candidate seed for the search process when it is in the vision frustum. As the silhouette areas are defined by the set of faces adjacent to vertices belonging to the silhouette edges, some of them may be neither located in the vision frustum, nor front-oriented. These faces are indeed necessary in order to apply the refinement process and obtain a final smooth silhouette curve. For the next silhouette extraction, only the subdivided triangles of the last silhouette triangle strips are analyzed. Figure 11 in the results section illustrates the progressive silhouette tracking during subdivision. Such an inherited behavior makes the assumption that no silhouette areas on the limit surface will exist on regions where no silhouette areas have been detected at the first extraction. This is a reasonable assumption because the process converges to a smooth limit surface.

Let us now examine the well-suited data structure used for silhouette extraction and adaptive visualization.

6.3 Data structure

At a given moment in time, the areas of interest are iteratively refined by subdivision for a current viewpoint. At the next instant, the viewpoint and the objects in the scene may be modified. The areas of interest must then be updated before being refined again. This thus necessitates restoring the initial mesh. Three different solutions for the data structure may then be chosen to restore this information:

1. A permanent copy of the original mesh to be restored by direct copy. Such a solution is memory expensive and time consuming because it requires the entire data traversal, instead of incrementally update their modified parts. Furthermore, it does not allow the building of a progressive reversible refinement;
2. A precomputed mesh hierarchy to quickly run through the refinement levels. This solution cannot be retained in the highly adaptive context of this paper. Since the $\sqrt{3}$ operator combines a face subdivision and an edge swapping, this last topological operator makes the number of combinations explode. Furthermore, the geometry we target on silhouette areas being highly detailed, such a hierarchy would take up too much memory space;
3. An adaptive data structure keeping a history of elementary actions generated on-the-fly during the adaptive refinement process. Such a history allows us to rewind the

sequence of actions associated with a view-dependent refinement process in a progressive way. This solution has been selected since it only requires restoration by simplification of the faces belonging to the areas of interest. This simplification is very straightforward, invoking only i) direct face or vertex elimination, ii) direct face updating or, iii) vertex displacement. This solution has the advantages of running fairly fast and requiring only low memory cost.

The proposed data structure implemented in C++ uses only face and vertex items, their adjacency and their history lists of actions as follows:

```
class Face
{
   Vertex *v[3];
   Face *f[3];               // adjacent faces
   list<Action> HistoryList; // history
};
class Vertex
{
   float x,y,z;
   list<Face> f;             // adjacent faces
   list<Vertex> v;           // adjacent vertices
   list<Action> HistoryList; // history
};
class Action
{
   unsigned short label;     // time index
   unsigned char type;       // four distinct actions
   void *parms;
};
```

Each face or vertex carries its own refinement history, made of a set of sequential labeled actions. The label $t := 3i + (s-1)$ refers to an action which has been done during the s_{th} step of the i_{th} iteration of the refinement sequence, where $s := \{1, 2, 3\}$ refers to the face subdivision step, the geometrical filtering step and the edge swapping step respectively. An action labeled with t does not relate to the forward t_{th} refinement step, but rather stores the minimal information required to restore the item in its position at the previous $(t-1)_{th}$ step. As any useless information has to be released during the rewind of a refinement history, the rewind of an elementary action named for example *face_created* simply deletes this face from memory. The following list enumerates the four elementary actions associated with the faces and the vertices required for a complete reversibility of the refinement history:

- Vertex creation during the 1-to-3 face subdivision: a vertex is inserted at the face barycenter (Fig. 8);
- Face creation during the 1-to-3 face subdivision operation: two faces are created while the split one is updated in order to save memory space (Fig. 8);
- Vertex displacement during the filtering process associated with the $\sqrt{3}$ subdivision operator (Fig. 8);

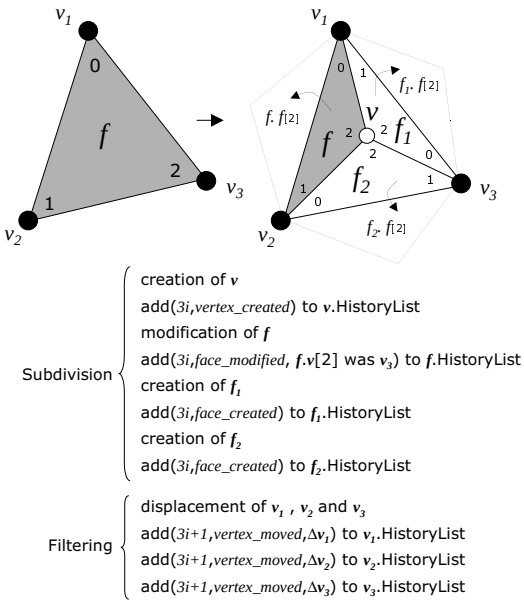

Fig. 8 Face subdivision and vertex filtering at the i_{th} iteration of the refinement sequence. We denote $f.v[n]$ the vertex of index n belonging to the face f, and $f.f[n]$ the face adjacent to f and opposite to vertex $f.v[n]$, $n = \{0, 1, 2\}$. The face subdivision creates one vertex v and two faces f_1 and f_2, modifies the face f, and adds to their history lists four associated elementary actions. The first action denoted *face_modified* associated with f and labeled $3i$ means that $f.v[2]$ was referring to v_3 before subdivision. We thus will be able to restore f to its original position. Each created item f_1, f_2 and v owns an initial action named *face_created* or *vertex_created*. Notice that each face of index 2 adjacent to f, f_1 and f_2 (i.e. $f.f[2]$, $f_1.f[2]$ and $f_2.f[2]$) is located outside the initial triangle face f. The filtering process applies a displacement Δv_n on the position of each initial vertices v_1, v_2 and v_3 and generates for each an action named *vertex_moved*, where the corresponding displacement Δv_n is stored.

- Edge swapping, equivalent to a vertex exchange between two adjacent faces. Our data structure does not use any edge item to enforce robustness and to ensure memory saving. The edge swapping is thus associated with only two elementary actions described in Fig. 9.

To improve the efficiency of the proposed process, we first precompute and store in a finite-length table the weighting coefficients $1 - \alpha_n$ and $\frac{\alpha_n}{n}$ (see Eq. 2), the number of different valences being limited on a triangular mesh. We also estimate in a first pass the memory allocation that would be necessary for the refinement process, in order to allocate memory per large blocks and avoid memory fragmentation.

The following pseudo-code summarizes the global extraction and subdivision sequence. Various refinement criteria may be expressed separately for frontfacing and silhouette areas by:

- A minimum edge length expressed in pixels in the image space;
- A predefined number of iterations;
- A threshold on an estimation of the surface curvature.

Let us now consider the current vision frustum, $\{F\}$ the faces belonging to the vision frustum, $\{F_f\}$ the frontfacing faces and $\{F_s\}$ the faces belonging to the silhouette areas:

Adaptive refinement sequence
Rewind last refinement
// Extraction stage
  $flag_{\rightarrow F}$ the faces located in the vision frustum
  $flag_{\rightarrow F_f}$ the frontfacing faces $\in \{F\}$
  $flag_{\rightarrow F_s}$ the silhouette faces from seeds $\in \{F_f\}$
$i := 1$
While($\{F_s\} \bigcup \{F_f\}$ is not empty) // i_{th} subdivision iteration
  If($f.flag_{\rightarrow F_f}$ and frontfacing refinement criterion is true)
    remove f from $\{F_f\}$
  If($f.flag_{\rightarrow F_s}$ and silhouette refinement criterion is true)
    remove f from $\{F_s\}$
  subdivide $\{F_s\} \bigcup \{F_f\}$ (i_{th} iteration)
  $flag_{\rightarrow F_f}$ sub-faces of $\{F_f\}$
  $flag_{\rightarrow F_s}$ silhouette faces from seeds $\in$ sub-faces of $\{F_s\}$
  $i++$

The following pseudo-code details the three subdivision steps presented in section 6.1, and describes the history generation:

1. **Face subdivision** ($3i_{th}$ step), see Fig. 8
For each face $f = \{v_1, v_2, v_3\}$ to subdivide
  add vertex v at $f.barycenter$
  add($3i$,$vertex\_created$) to v.HistoryList
  f := $\{v_1, v_2, v\}$
  add($3i$,$face\_modified$,$f.v[2]$ was v_3) to f.HistoryList
  add face f_1 := $\{v_3, v_1, v\}$
  add($3i$,$face\_created$) to f_1.HistoryList
  add face f_2 := $\{v_2, v_3, v\}$
  add($3i$,$face\_created$) to f_2.HistoryList
  $flag_{\rightarrow to\_filter}$ v_1, v_2 and v_3
  update local vertex and face adjacency

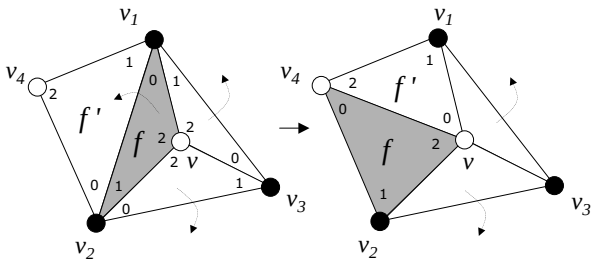

Fig. 9 Edge swapping at the i_{th} iteration of the refinement sequence. Let us consider the face f, we swap the edge shared with $f' := f.f[2]$. This operation is memorized using only two actions labeled $3i + 2$ and associated with f and f'. In this example, the vertex $f.v[0]$ was referring to v_1 before swapping, while $f'.v[0]$ was referring to v_2. The two curved arrows on the right, highlight the fact that two pairs of adjacent faces around vertex v remain to be processed.

  $flag_{\rightarrow to\_swap}$ f, f_1 and f_2

2. **Geometrical filtering** ($(3i + 1)_{th}$ step)
For each vertex v with $flag_{\rightarrow to\_filter}$
  read convolution mask associated to $v.valence$
  compute and store displacement Δv
For each vertex v with $flag_{\rightarrow to\_filter}$
  move vertex according to Δv
  add($3i + 1$,$vertex\_moved$,Δv) to v.HistoryList
  $flag_{\rightarrow filtered}$ v

3. **Edge swapping** ($(3i + 2)_{th}$ step), see Fig. 9
For each face f with $flag_{\rightarrow to\_swap}$
  f' := $f.f[2]$ // neighbor of index 2 only
  // let $f = \{v_1, v_2, v\}$ and $f' = \{v_2, v_1, v_4\}$
  f := $\{v_4, v_2, v\}$
  add($3i + 2$,$face\_modified$,$f.v[0]$ was v_1) to f.HistoryList
  f' := $\{v, v_1, v_4\}$
  add($3i+2$,$face\_modified$,$f'.v[0]$ was v_2) to f'.HistoryList
  update local face and vertex adjacency
  $flag_{\rightarrow swapped}$ f
  $flag_{\rightarrow swapped}$ f'

7 Results

We have implemented the proposed view-dependent refinement technique for 3D meshes in C++ and by using the OpenGL graphics library. The results obtained on many different examples have highlighted strong benefits of this approach compared to a uniform refinement technique. In the general case, and for a given user viewpoint, only a fraction of the global surface is seen. It is then judicious to refine only these areas of interest. One of the most obvious benefits is in the visual quality which is obtained by the selective smoothing of the silhouettes. This smoothing is updated according to user viewpoint with efficiency and complete robustness. The process times given in this section have been measured on a PII 266 MHz, using 256 Mbytes memory, including extraction, but excluding the building of the corresponding OpenGL display lists and the rendering.

Figure 10 highlights the benefits in terms of savings in calculation when taking the viewpoint into account. In this example, the silhouette areas have been refined until the projected edge length does not exceed the screen resolution, this heuristic eliminating clearly the polygonal aspect along the silhouette. The non-visible backfacing areas are not refined and simply removed from the display list, while frontfacing areas are refined according to a maximum number of four subdivision iterations. We could also use, for the frontfacing areas, a device-driven heuristic such as the minimum edge length in the image plane, or a surface geometry criterion such as a curvature estimation. In this example, 6.4 s have been required to increase the original 838 triangle mesh up to a $43\,294$ triangle mesh. A global uniform refinement equivalent to the silhouette area would have led to a $600\,000$ triangle

mesh, with the same visual quality for the given viewpoint as this $43\,294$ triangle mesh.

Figure 11 illustrates how the silhouette areas progressively narrow during four iterations of adaptive refinement only applied on the silhouette areas. The exhaustive silhouette search only occurs during the first extraction (top line, column 2), the following ones being restricted to the last silhouette triangle strips. A time of 0.9 s has been required to increase the triangle count from 562 to $3\,736$, the silhouette extraction representing only 2% of it.

Figure 12 illustrates an adaptive refinement according to the following number of subdivision iterations: three on frontfacing areas and six on silhouette areas. A time of 4.2 s has been required to increase the triangle count from $1\,396$ to $18\,034$. This figure also highlights the cropping effect of the local refinement restricted to the vision frustum.

We have also implemented the uniform 1-to-4 Loop subdivision scheme and the uniform $\sqrt{3}$ subdivision scheme. Figure 13 compares them with the proposed view-dependent refinement technique. A sequence of uniform subdivision highlights the fact that the $\sqrt{3}$ subdivision operator (line 1) increases the geometric density more slowly than the Loop subdivision operator (line 2). The proposed method allows us to further optimize the ratio perceptual quality / number of faces since the refinement is localized on visually relevant areas such as frontfacing and silhouette areas (line 3). In this example, three subdivision iterations on frontfacing areas make it possible to obtain the same quality with fewer faces than the uniform $\sqrt{3}$ scheme. Furthermore, five subdivision iterations on silhouette areas completely remove the polygonal aspect of the silhouette with a $28\,170$ triangle count mesh, while the uniform Loop and the uniform $\sqrt{3}$ scheme reach with only three iterations $88\,840$ and $37\,692$ faces respectively. Figure 14 illustrates the increase of the number of mesh faces associated with the refinement sequences shown in Fig. 13.

In particular, these results demonstrate the gain obtained in memory for the adaptive refinement, and the benefit of not having to render so many redundant triangles. Notice that such a gain is less obvious in the context of a continuous navigation since we do not exploit the frame-to-frame coherence.

Interestingly, we have also noticed a shorter time required for the mesh generation through subdivision than for a straight reading of a raw file describing the refined mesh, or even for a bitstream decoding of this refined mesh [10].

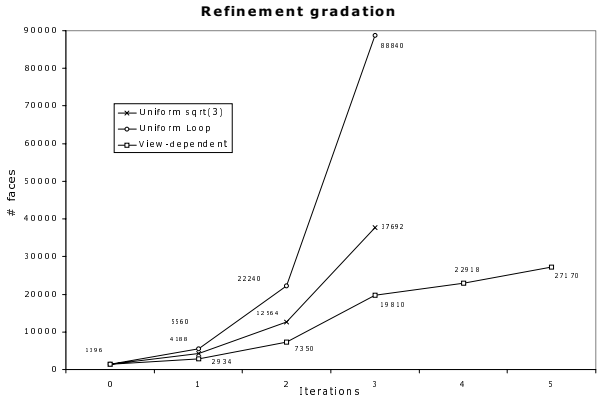

Fig. 14 Increase of the number of faces (mesh *venus*, see Fig. 13) with the uniform Loop refinement scheme, the uniform $\sqrt{3}$ refinement scheme and the proposed view-dependent technique. Note that the fourth and fifth iterations correspond to a view-dependent refinement only applied on the silhouette areas.

8 Conclusion and future work

We have derived an efficient view-dependent refinement technique of 3D meshes which provides visually smooth surfaces from the subdivision scheme defined by Kobbelt [7]. Our contribution mainly concerns the view-dependent behavior, and an efficient implementation associated with a simple data structure that contains neither edges, nor hierarchy which appears unsuitable to highly adaptive methods.

We have pointed out a practical application of the $\sqrt{3}$ subdivision operator: a complete adaptation both to the capabilities of a graphic terminal and to the user viewpoint. For a telecom operator, the proposed technique can be seen as a few steps towards scalable tuning of the *Quality of Service* for 3D-based applications, well-suited to low and high end terminals.

Despite the efficiency of the proposed scheme, we note two remaining limitations. The first one lies within the refined data which have only a short life. Indeed, for a given viewpoint, a set of visually relevant data is generated through subdivision, and completely removed from memory through the rewind of its refinement history, before processing again the original mesh for the next user viewpoint. This drawback may be reduced in the reasonable case of continuous navigation in the virtual scene by stretching the silhouette triangle strips in order to spread the refinement cost over several images. The second limitation is that frontfacing areas which are hidden by other parts are nevertheless refined. Again, in the hypothesis of continuous navigation, the OpenGL z-buffer built for the previous viewpoint could be used to avoid useless processing.

Other future work includes: i) a continuous geomorph between levels of details, ii) the definition of semi-sharp edges as in [33] that may allow a more sophisticated surface reconstruction technique, iii) the enhancement of the extraction efficiency by using the output-dependent silhouette extraction method from Benichou and Elber [20]. However, for current mesh complexity as those in our examples, the extraction process represents only an average of 2% of the entire subdivision process duration, iv) the fast backface culling using normal masks from Zhang and Hoff [22] which could also further reduce the cost of the extraction process.

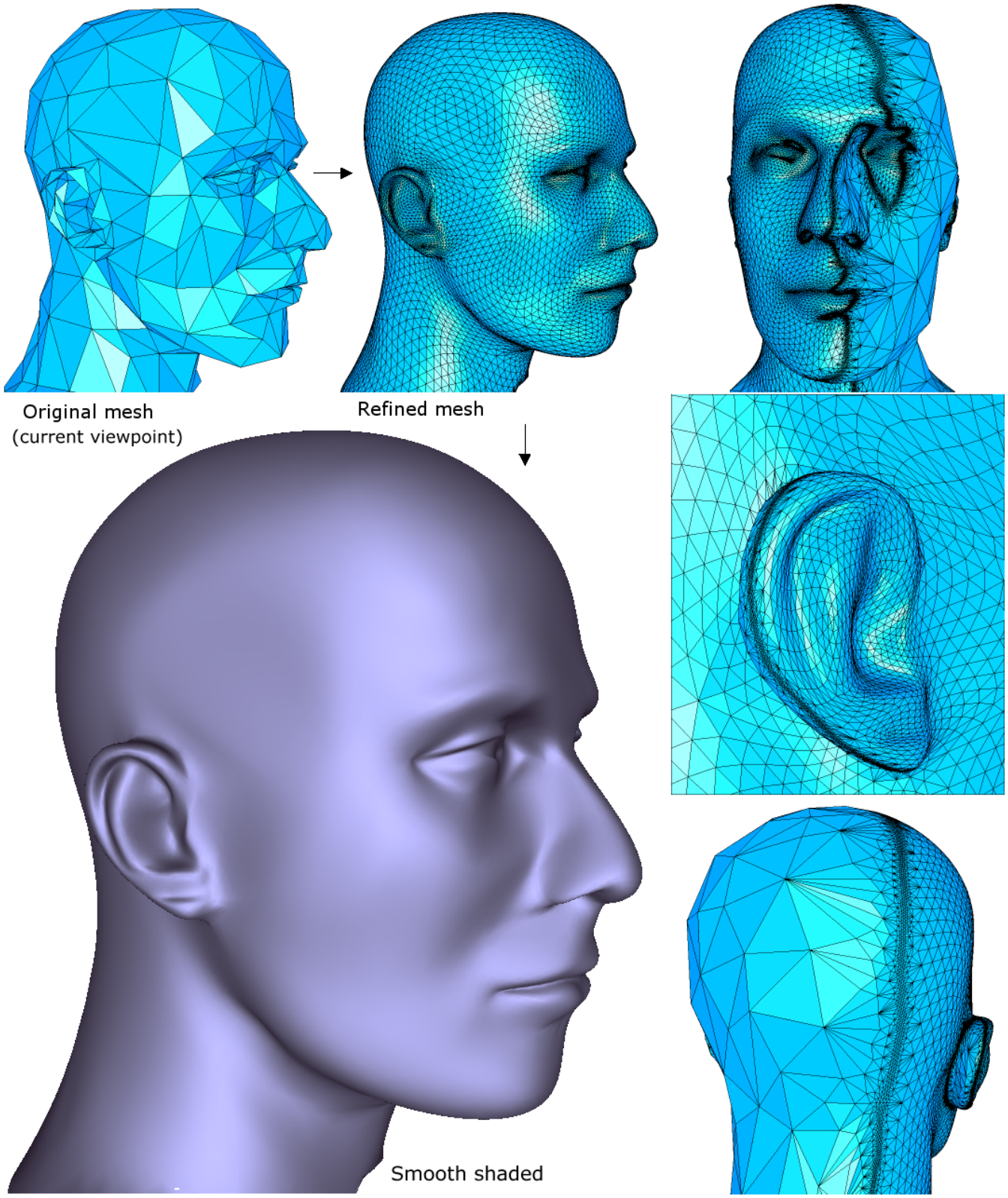

Fig. 10 *Mannequin* mesh . Top line, and from left to right: the original mesh, the refined mesh, and the refined model displayed under a different viewpoint in order to highlight the geometrical density obtained on silhouette areas. Notice how backfacing areas are left unrefined in order to optimize the ratio: perceptual quality / number of faces. Bottom left: the polygonal aspect from the silhouettes has been removed since the mesh is refined on corresponding areas until the edge length measured in the image space is lower than one pixel. Middle right: the silhouette details on the right ear lobe. Bottom right: the refined mesh seen from the back.

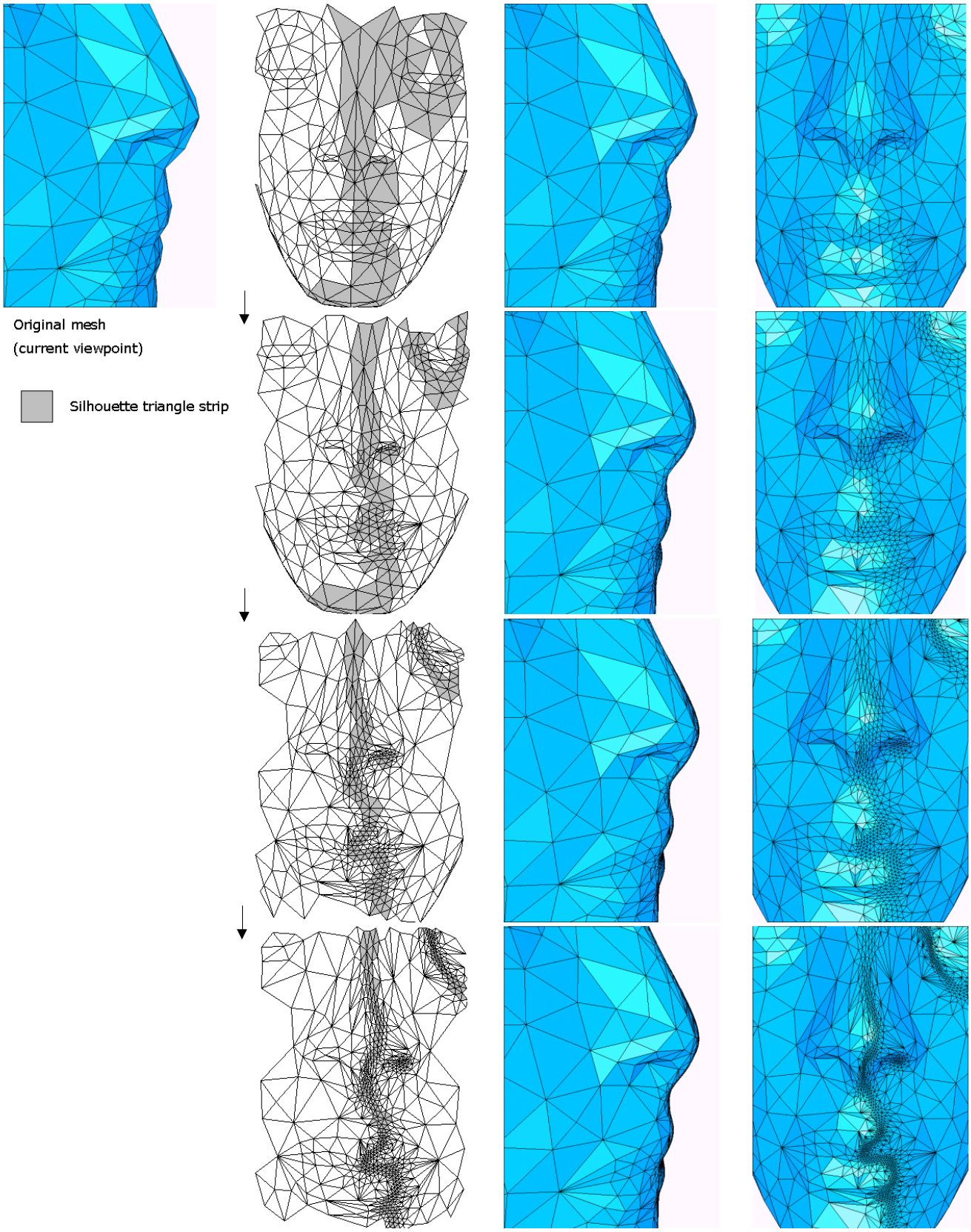

Fig. 11 *Nefertiti* mesh. Top left: the original mesh seen from the current viewpoint. From top to bottom: four successive iterations of the silhouette refinement. Column 2: the silhouette strips extracted before subdivision. Column 3: progressive smoothing of the silhouette after subdivision as seen from the user viewpoint. Column 4: local and progressive increase of the mesh density near the silhouette.

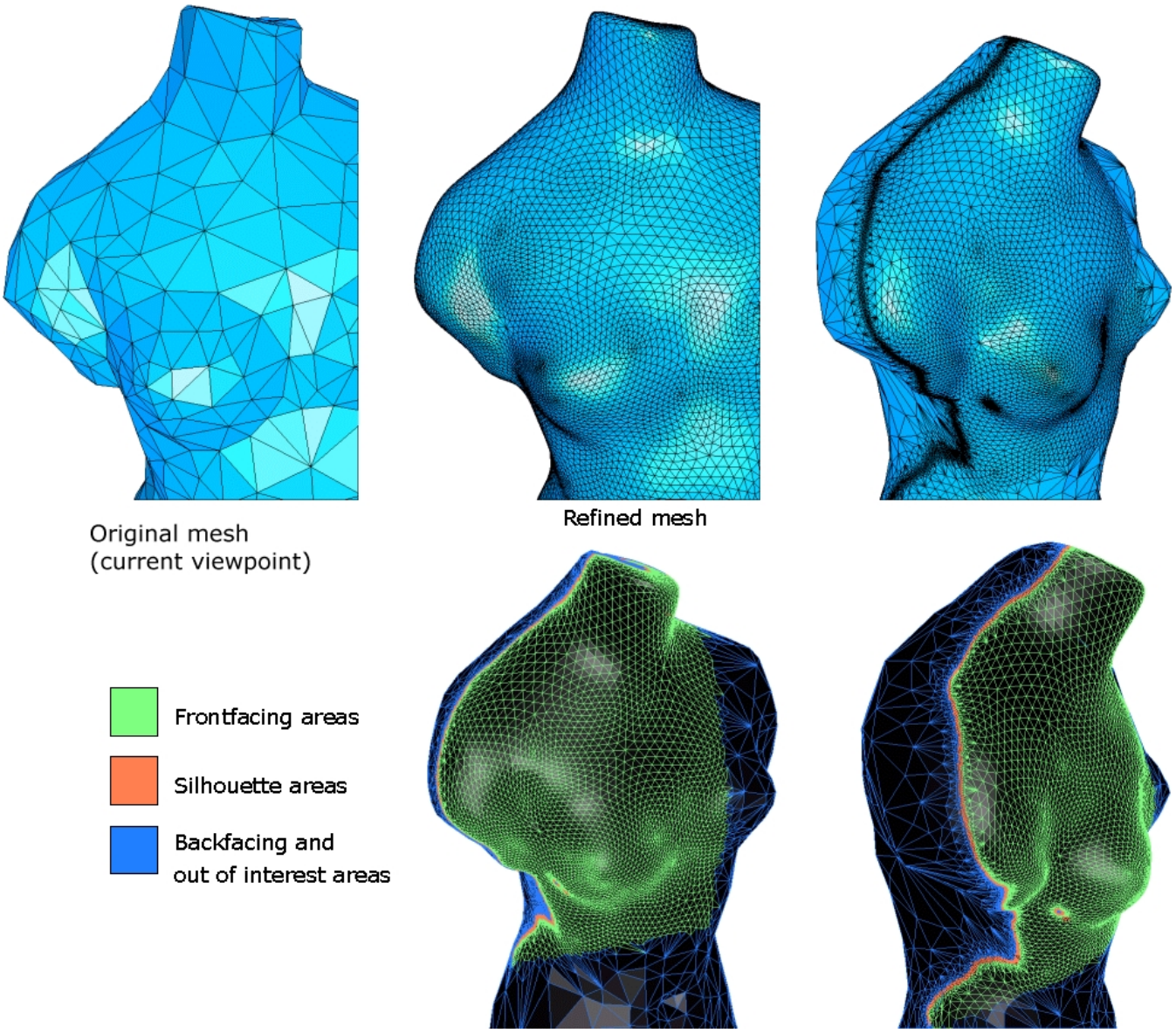

Fig. 12 *Venus* mesh. Top line, from left to right: the original mesh seen from the current viewpoint, the model refined using three iterations on frontfacing areas and six on silhouette areas, and the model seen from a different viewpoint. Bottom images: areas of interest are restricted to the silhouette strips and to the set of frontfacing faces intersecting the vision frustum.

Acknowledgements A very special thanks to Gaspard Breton for helping to implement the camera transform, to Alexandre Buisson for invaluable help in optimization tips, and to Mathieu Desbrun for writing assistance. The mannequin head is courtesy of Hugues Hoppe. This work is part of the 3D project V2NET, backed by a grant from the RNRT (National Telecommunications Research Network), and has been partially supported by the Integrated Media Systems Center, a NSF Engineering Research Center, cooperative agreement number EEC-9529152.

References

1. D. Zorin and P. Schröder. Subdivision for Modeling and Animation. *SIGGRAPH 2000 course notes*, 2000.
2. P. Schröder. Opportunities for subdivision-based multiresolution modeling. In *Pacific Graphics 99 Conference Proceedings*, pages 104–105, 1999.
3. E. Catmull and J. Clark. Recursively generated B-Spline surfaces on arbitrary topological meshes. *Computer Aided Design*, 10(6):350–355, 1978.
4. D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer Aided Design*, 10(6):356–360, 1978.
5. C. Loop. Smooth surface subdivision based on triangles, 1987. Master's thesis.
6. H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. Mc Donald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. In *SIGGRAPH 94 Conference Proceedings*, pages 295–302, 1994.
7. L. Kobbelt. $\sqrt{3}$-Subdivision. In *SIGGRAPH 2000 Conference Proceedings*, pages 103–112, 2000.

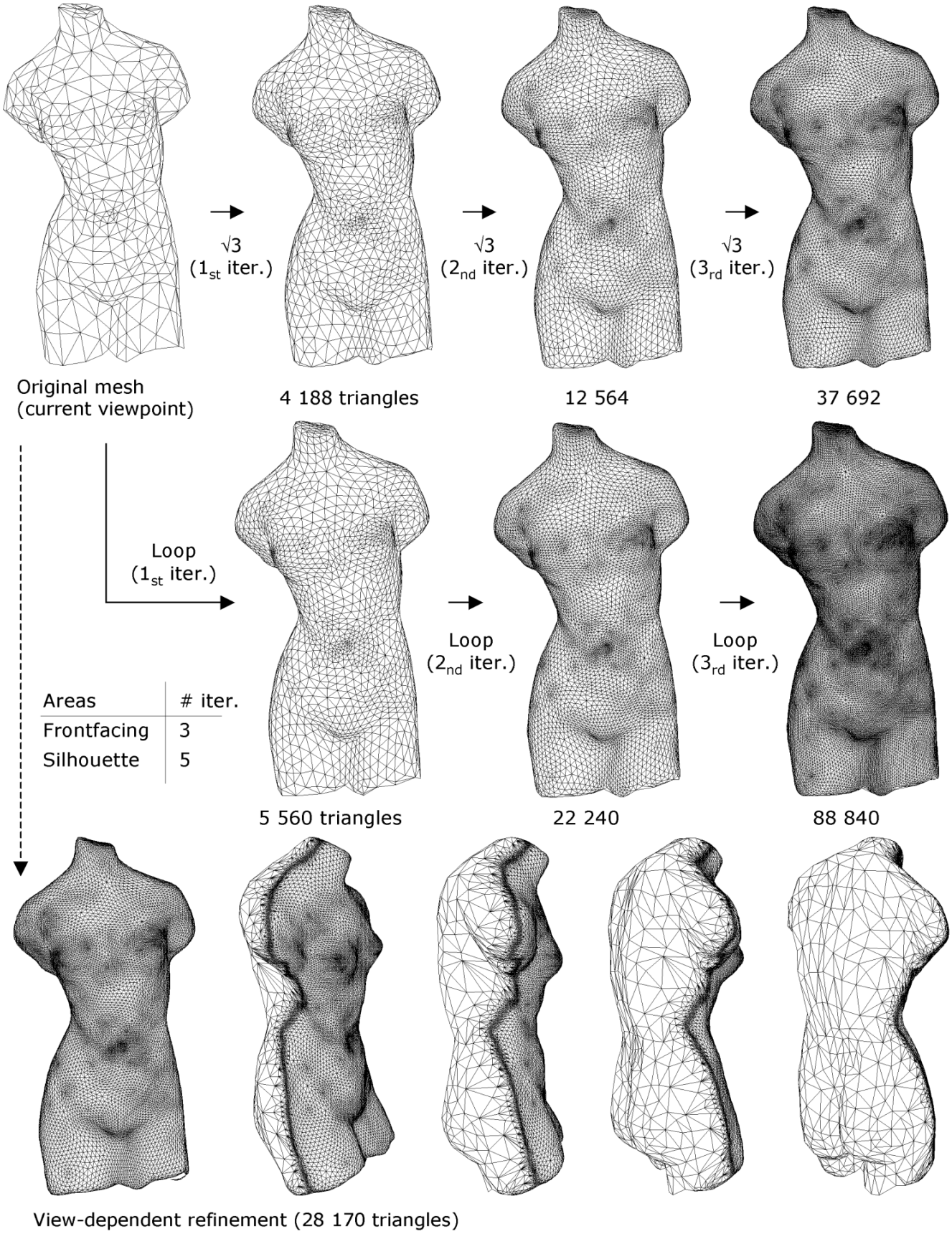

Fig. 13 *Venus* mesh. Top left: the original mesh seen from the current viewpoint. Line 1: a sequence of three uniform $\sqrt{3}$ subdivision iterations. Line 2: a sequence of three uniform Loop subdivision iterations. Line 3: the resulting view-dependent refined mesh after three and five iterations on frontfacing areas and on silhouette areas respectively.

8. A. Khodakovsky, P. Schröder, and W. Sweldens. Progressive geometry compression. In *SIGGRAPH 2000 Conference proceedings*, pages 271–278, 2000.
9. P. Alliez and M. Desbrun. Progressive encoding for lossless transmission of 3D meshes. In *SIGGRAPH 2001 Conference Proceedings*, pages 195–202, 2001.
10. P. Alliez and M. Desbrun. Valence-driven connectivity encoding of 3D meshes. In *Eurographics 2001 Conference Proceedings*, pages 480–489, 2001.
11. H. Hoppe. View-dependant refinement of progressive meshes. In *SIGGRAPH 97 Conference Proceedings*, pages 189–198, 1997.
12. J.J. Koenderink. What does the occluding countour tell us about solid shape. *Perception*, 13:321–30, 1984.
13. X. Gu, S.J. Gortler, H. Hoppe, L. McMillan, B. J. Brown, and A.D. Stone. Silhouette mapping. Technical Report TR-1-99, Harvard University, 1999.
14. P.V. Sander, X. Gu, S.J. Gortier, H. Hoppe, and J. Snyder. Silhouette clipping. In *SIGGRAPH 2000 Conference proceedings*, pages 327–334, 2000.
15. D. Dooley and M. Cohen. Automatic illustration of 3d geometric models: Lines. In *Proceedings of the Symposium on Interactive 3D Graphics*, pages 77–82, 1990.
16. L. Markosian, M.A. Kowalski, S.J. Trychin, J.F. Hugues, and L.D. Bourdev. Real time non photorealistic rendering. In *SIGGRAPH 97 Conference Proceedings*, pages 415–420, 1997.
17. A. Gooch, B. Gooch, P. Shirley, and E. Cohen. A non-photorealistic lighting model for automatic technical illustration. In *SIGGRAPH 98 Conference Proceedings*, pages 447–452, 1998.
18. B. Gooch, P-P.J. Sloan, A. Gooch, P. Shirley, and R. Riesenfeld. Interactive technical illustration. *ACM Symposium on Interactive 3D Graphics*, pages 31–38, 1999.
19. R. Raskar and M. Cohen. Image precision silhouette edges. *Symposium on Interactive 3D Graphics 1999, Atlanta*, pages 135–140, 1999.
20. F. Benichou and G. Elber. Output sensitive extraction of silhouette from polygonal geometry. In *Pacific Graphics 99 Conference Proceedings*, pages 60–69, 1999.
21. G. Barequet, C.A. Duncan, M.T. Goodrich, S. Kumar, and M. Pop. Efficient perspective-accurate silhouette computation. *ACM SCG 99*, pages 417–418, 1999.
22. H. Zhang and K.E. Hoff. Fast backface culling using normal masks. *Symposium on Interactive 3D Graphics*, pages 103–106, 1997.
23. C. Touma and C. Gotsman. Triangle mesh compression. In *Graphics Interface 98 Conference Proceedings*, pages 26–34, 1998.
24. H. Hoppe. Progressive meshes. In *SIGGRAPH 96 Conference Proceedings*, pages 99–108, 1996.
25. O. Devillers and P-M. Gandoin. Geometric compression for interactive transmission. In *Visualization 2000 Conference Proceedings*, pages 319–326, 2000.
26. G. Chaikin. An algorithm for high speed curve generation. *Computer Graphics and Image processing*, 3:346–349, 1974.
27. M. Halstead, M. Kass, and T. DeRose. Efficient, fair interpolation using catmull-clark surfaces. In *SIGGRAPH 93 Conference Proceedings*, pages 35–44, 1993.
28. J. Warren. Subdivision methods for geometric design. Technical report, Rice University - Department of Computer Science, 1995.
29. M. Eck and H. Hoppe. Automatic reconstruction of b-spline surfaces of arbitrary topological type. In *SIGGRAPH 96 Conference Proceedings*, pages 325–334, 1996.
30. T.W. Sedelberg, J. Zheng, D. Sewell, and M. Sabin. Non-uniform recursive subdivision surfaces. In *SIGGRAPH 98 Conference Proceedings*, pages 387–394, 1998.
31. H. Suzuki, S. Takeuchi, and T. Kanai. Subdivision surface fitting to a range of points. In *Pacific Graphics 99 Conference Porceedings*, pages 158–167, 1999.
32. A. Levin. Interpolating nets of curves by smooth subdivision surfaces. In *SIGGRAPH 99 Conference Proceedings*, pages 57–64, 1999.
33. T. DeRose, M. Kass, and T. Trunong. Subdivision surfaces in character animation. In *SIGGRAPH 98 Conference Proceedings*, pages 85–94, 1998.