Istituto di Matematica Applicata e Tecnologie Informatiche

Consiglio Nazionale delle Ricerche

Genova - ITALY

ReMESH

Version 1.2

An interactive and user-friendly environment for remeshing surface triangulations

MARCO ATTENE

USER MANUAL

Index

Research and software development involving geometry processing are often slowed down by the absence of suitable models for testing and benchmark purposes. In particular, when dealing with triangle meshes, a researcher may need to check the behavior of a new algorithm on several particular cases. In most situations, the test model is easily conceivable in mind but, at actual design time, its formalization turns out to be a much harder task than expected. Also, simple modifications over an existing triangle mesh may become a tedious work without a suitable interactive environment.

In order to simplify the remeshing of existing models, we have developed ReMESH to interactively edit manifold triangle meshes, mostly through user friendly actions such as mouse clicks and drags.

1. ReMESH

ReMESH incorporates a wrapper for loading several file formats, including the web standard VRML 2.0, the older VRML 1.0, the OpenInventor file format (.iv), the Object File Format (.off), Stanford's PLY, Wavefront's OBJ, Stereolithography (.stl), and the SWM, which is the compressed format produced by the SwingWrapper encoder [3]. In all of the cases, however, the parser assumes that the file has **one** section describing the vertex coordinates, and **one** section describing the **faces** (only simply connected faces are allowed, and are automatically triangulated); further information (transformations, colors, …) is ignored. While loading, a data structure such as the one described in appendix B is initialized. Some file formats, however, may represent sets of polygons which are non-manifold and/or non-orientable. In this case the loader runs the Cutting&Stitching algorithm [11] to make the mesh manifold. If the resulting manifold surface is not oriented, ReMESH assigns an orientation to one triangle for each connected component, and propagates the orientation to neighboring triangles; once all of the triangles have been visited, the mesh is cut along edges having non-consistently oriented incident triangles. Possible isolated vertices are removed automatically.

Once the loader is settled with the topology and the structure is properly filled, the tool performs some geometrical checks and corrections (removal of degenerate triangles, identification of edges having coincident end-points, …). These corrections can be prevented by passing the additional parameter '--nocheck' to the command line.

Finally, the graphical user interface (GUI) starts up and shows the model encoded in the data structure, some information about the model (number of vertices, handles, boundaries, …), and a set of controls and parameters to be used for the editing (see Figure 1).

If the user wants to operate on more models at the same time, the *Append* button may be used to load a triangle mesh from file and to insert it in the current scene without removing the existing mesh(es). Notice that no geometrical check is performed on the newly loaded mesh that may consequently have degenerate elements. To automatically remove all the degeneracies of the current meshes use the buttons provided in the *Check & Repair* window.

To load multiple models at once and perform a global geometric correction just pass all their filenames to the command line.

1.1 Point-and-Click Help

To obtain information about a widget of the toolbox, first click on 'Help' and, while the cursor looks like a hand, click on the desired widget to pop up a help dialog. If no dialog is displayed, it means that no help is available for that widget.

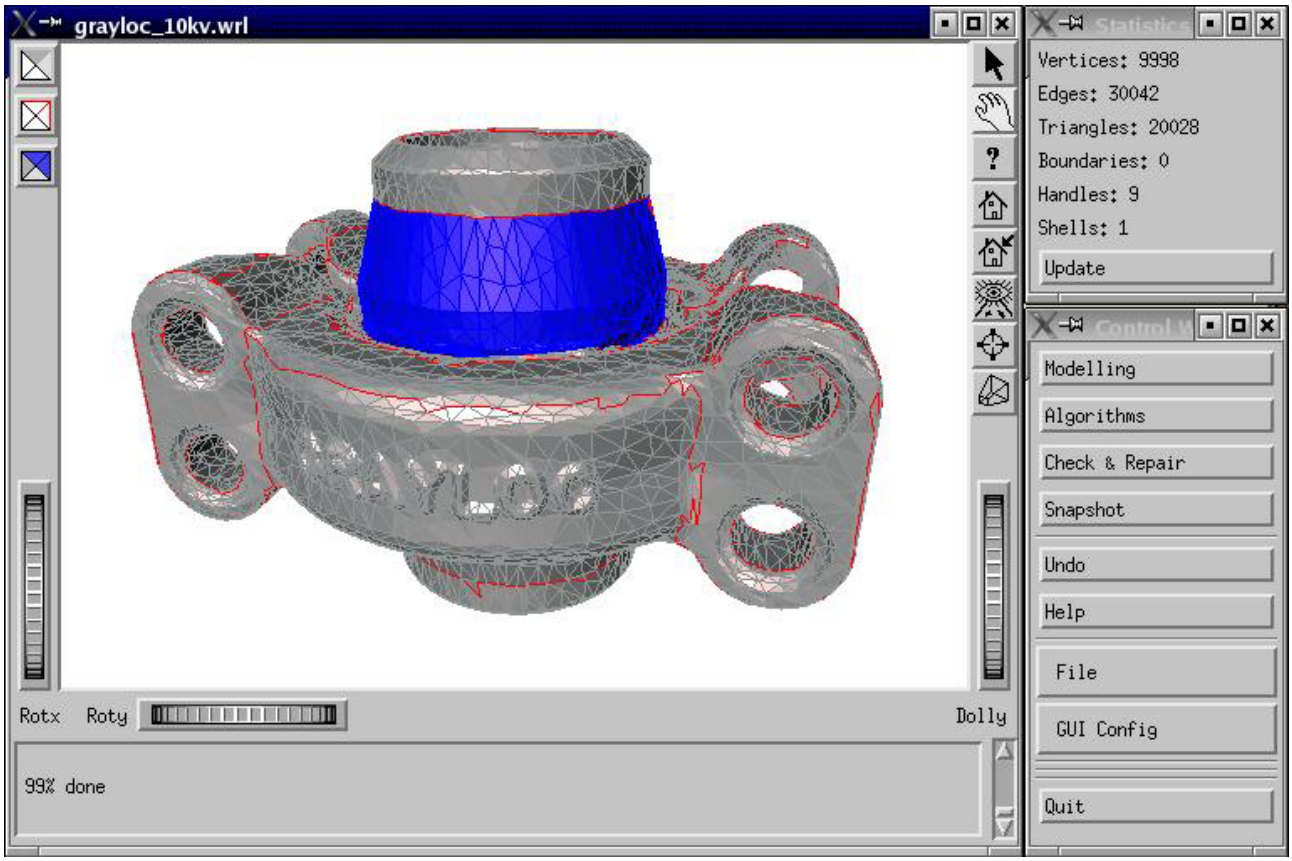

Figure 1: Screenshot of the graphical user interface of the remeshing tool.

1.2 Navigating the scene

The graphical part of the toolbox is built upon SGI's Open Inventor toolkit, and most of the facilities for navigating the scene have been inherited from it. For self-containedness, however, we briefly sketch them here.

The canvas and all of the controls in its window are part of an Examiner viewer component of Open Inventor. It allows the user to rotate the view around a point of interest using a virtual trackball. This viewer also allows the user to translate the camera in the viewer plane, as well as dolly (move forward and backward) to get closer to or further away from the point of interest. The viewer also supports seek to quickly move the camera to a desired point.

With reference to Figure 1, the three wheels labeled *Rotx*, *Roty* and *Dolly* control respectively the rotation around the X axis, the one around the Y axis and the dolly. These actions, however, can also be carried out through direct interaction of the mouse in the canvas.

Further navigation parameters can be controlled through the buttons on the right of the window. Some aspects of the scene, such as the background color and pictorial properties of the surface (material, shading, …), can be selected by clicking the button labeled *GUI Config -> settings*.

2. Overview of the features

Besides navigation and visualization tasks, the user can perform a number of editing operations, that we conveniently subdivide into two classes.

2.1 Interactive operations

This kind of actions can be performed through mouse-clicks directly on the surface. Notice that clicks and drags can also be used to navigate the scene. In order to establish the behavior of mouse events in the canvas without ambiguity, two buttons are provided (the little arrow and the hand on the top-right corner of the window) for switching between *visualization* and *interactive* modes. At start up the mode is set to *visualization*, the pointer has the shape of a hand and mouse events are

used for navigating the scene. The user can switch to the *interactive* mode by clicking on the *arrow* button and can go back to *visualization* by clicking on the little hand. Interactive operations include triangle removal, vertex insertion, edge swapping, shell selection, and many others that can be selected from a list of *Interaction* buttons in the *Modeling* window.

2.2 Selections and cut ‘n paste operations

Clicking using the middle mouse button sets the center of a spherical selection that can be used for processing sub-regions of the mesh. By holding the middle-nutton and dragging the mouse, it is possible to specify the radius of the selection. All the triangles having all the three vertices within the sphere are selected and shown using a different color.

Holding the **shift** key before clicking adds the new selection to the existing one. Holding the **ctrl** key before clicking removes the new selection from the existing one. Holding both the **shift** and **ctrl** keys before clicking intersects the new selection with the existing one. Clicking without dragging selects a new region with the same radius of the last one. Selected triangles can be removed by pressing the "Delete" key while in *interactive* mode.

The usual shortcuts **CTRL+C**, **CTRL-X** and **CTRL-V** can be used to perform **copy**, **cut** and **paste** operations respectively. The clipboard consists of the selected and copied/cut triangles.

2.3 Other operations

Besides interactive operations, there are several other actions available for the user in which some parameters are required. In such a case, a new dialog-style window containing controls on the parameters is shown. For example, if the user wants to simplify the mesh, the corresponding dialog shows, among the others, a slider for the selection of the target number of vertices. When settled with the parameters, the user clicks on the *OK* button to actually perform the simplification. Also, non-interactive operations include all of the actions which do not require the specification of a particular element of the mesh, such as global modifications, or operations which rely on a prior specification of a selection.

3. Local and Global Editing

In this section we describe the set of operations that can be performed interactively on the triangle mesh and produce a modification. Sometimes it is easier to interact with the mesh if its edges are explicitly rendered; to switch to such a kind of rendering, the *GUI Config -> settings* window includes a toggle button labeled *hidden-line*. Local editing operations are accessible through two windows: the *Modelling* and the *Algorithms*.

`Modelling -> Transform -> Normalize Scene`

This operation computes the bounding box of the scene and normalizes it. Specifically, it performs a uniform scaling and a translation of the mesh, so that it fits into the cube [0,1],[0,1],[0,1]. This operation is particularly useful to avoid numerical overflows due to extreme values of the original coordinates.

`Modelling -> Transform -> Transform Scene`

This button pops up a dialog containing a number of controls for setting affine transformations to the whole geometry. Notice that while the visualization of the scene being transformed is interactive, vertex coordinates are actually transformed only when the dialog is closed.

`Modelling -> Transform -> Transform Shells`

This button pops up a dialog in which each connected component (shell) is bounded by a "sensitive" box. Each such box can be interactively manipulated through clicks and drags, and the corresponding shell is transformed accordingly. Dragging a face of the box corresponds to 'translate'

the shell. Dragging an edge of a box makes the corresponding shell rotate. Dragging a vertex of the box causes a uniform scale of the shell.

Modelling -> Transform -> Redistribute Shells

All the connected components are normalized and placed on a virtual sphere. Particularly useful to work with several models having very different sizes and positions is space.

Modelling -> Selection -> Grow

Adds to the current selection all the triangles having at least one selected vertex.

Modelling -> Selection -> Shrink

Removes from the current selection all the triangles having at least one vertex on the border of the selection.

Modelling -> Selection -> Invert

Toggles the selection status of the triangles

Modelling -> Selection -> Re-triangulate

Performs a re-triangulation of the currently selected region using a Delaunay-like approach. Specifically, a common plane is computed as the average of the planes of the triangles selected; then, the vertices of the region are projected on the plane and edges are iteratively swapped up to convergence (which is guaranteed on planar and simple domains). Finally, the vertices are moved back to their original positions. This operation is particularly useful to improve the quality of nearly flat regions. The selection must be simple and its Gauss map must be entirely contained in a semi-sphere.

Modelling -> Selection -> Remove

Remove all the triangles belonging to the selected region(s). This is equivalent to pressing the 'Delete' button on the keyboard.

Modelling -> Flip Normals

When the user selects this button, all the triangles of the mesh are inverted.

Modelling -> Fill Holes

When selected, this button pops up a dialog where the user can select a threshold number 'n'. Then, all the boundary loops made of at most 'n' edges are patched using the same strategy as in the interactive operation *Triangulate Holes*. If the 'refine' toggle is set, after triangulation new vertices are inserted in the patched holes so as to reproduce the sampling density of neighboring regions. If the 'smooth' button is set too, the newly inserted vertices are moved so as to produce a tangentially continuous patch with respect to its boundary. If a selection exists, this button limits its action to the selected region. Only completely selected boundaries are filled.

Modelling -> Sharp edge tagging

This feature allows the user to automatically tag as sharp (see the interactive operation *Tag Sharp Edges*) all of the edges in which the normals of the two incident triangles form an excessive angle. The threshold value is selected through a dialog and defaults to PI/4 (= 45 degrees).

Modelling -> Join Boundaries

This feature requires the user to click on two vertices belonging to two different boundary loops. Starting from such a pair, the algorithm joins the two loops with new triangles. Such newly inserted

triangles remain selected for possible further processing. The two boundaries can be “just connected”, that is, only two triangles are added to chenge the topology. If the boundaries are to be filled completely (as opposed to being just connected), the user may choose whether to refine the patch by inserting additional vertices, or to smooth it, by moving additional vertices towards positions that make the patch fairly smooth with respect to the original geometry.

Modelling -> Interaction -> Select triangles

When this toggle is set, each triangle the user clicks on is marked as "selected" and subsequent operations requiring a selection will act only on these triangles. If the user clicks on a previously selected triangle, it changes its state to "unselected". In order to display selected triangles using a different color it is possible to switch to *per-face color* mode from the *GUI config -> settings* window.

Modelling -> Interaction -> Remove triangles

Each triangle the user clicks on is removed from the mesh. The process takes care of maintaining a coherent data-structure, possibly by duplicating non-manifold vertices that can be created due to the removal.

Modelling -> Interaction -> Swap edges

When the user clicks on (or nearby) an edge, that edge is swapped. Also in this case, the toolbox takes care of not performing illegal swaps (i.e. boundary edges or resulting incoherent topology).

Modelling -> Interaction -> Insert vertices

When this toggle is set, a new vertex is inserted in the point of the surface the user clicks on. If such a point is inside a triangle, the insertion is performed through a triangle-split operation. If it is on (or nearby) an edge, an edge-split operation subdivides that edge. If it is on (or nearby) a vertex, that vertex is snapped to the position of the point. In all of the cases, the term "nearby" is quantified by the "Acos tolerance" chosen by the user in the Check&Repair window. Such a threshold prevents the creation of nearly degenerate triangles that would be produced by splitting a triangle with a point too close to one of its edges.

Modelling -> Interaction -> Triangulate Holes

When this toggle is set, the user can click on a boundary edge to start a triangulation routine that tries to fill the corresponding hole. The triangulation approach is based on a number of heuristics inspired from [20][4] which try to minimize bad behaviors such as extreme dihedral angles, degenerate triangles, and so on. No new vertex is inserted.

Modelling -> Interaction -> Fill Holes

This is as 'Interaction->Triangulate Holes', but refines and smooths the resulting patch so as to reproduce the sampling density of neighboring regions and to have a tangential continuity with the surrounding mesh [17].

Modelling -> Interaction -> Flip shell

When the user clicks on a triangle, that triangle along with all the other triangles in the same connected component are reversed, that is, their orientations (and therefore their normals) are inverted.

Modelling -> Interaction -> Selects shell

When this toggle is set, the connected component containing the triangle clicked is selected.

Modelling -> Interaction -> Select SB region

Selects all the triangles belonging to a “sharply bounded” region. Specifically, the surface point the user clicks on is used as a seed to grow the selected region up to edges tagged as *sharp*.

Modelling -> Interaction -> Tag Sharp Features

When this toggle is set, the behavior of the clicks is similar to the case of select triangles, except for the fact that here the user selects edges instead of triangles. To visualize the selected edges it is necessary to turn on the per-tag mode from the *GUI config -> settings* window. These edges will receive particular treatment during some other processes (e.g., *Algorithms -> Bender*).

Modelling -> Region radius

Its functionality is deprecated. Currently, its only scope is to show a numerical feedback of the radius selected interactively by the user.

Algorithms -> Split all edges

This button splits all the edges at their middle points. The geometry of the mesh does not change but the number of triangles is quadruplicated. All the edges are split unless a selection is active; in this latter case, only the edges belonging to the selected area(s) are split.

Algorithms -> Loop subdivide

This button performs a subdivision step using Loop's scheme. Boundary treatment is provided, while no special rule is implemented to manage sharp edges. If a selection is active, the subdivision works on it only.

Algorithms -> Bender

This button performs a subdivision step using Bender, a modified Butterfly scheme in which both boundaries and sharp edges are treated properly [2]. If a selection is active, the subdivision works on it only.

Algorithms -> Sqrt(3) Subdivide

This button performs a subdivision step based on Kobbelt's approximant sqrt(3) scheme. No support is provided for boundaries and sharp edges. If a selection is active, the subdivision works on it only.

Algorithms -> Laplacian Smooth

This button pops up a dialog where the user can select the number of Laplacian smoothing iterations to be performed on the mesh. At each iteration, each vertex is moved towards the center of mass of its neighbors. The amount of movement is controlled by the parameter "Neighbors' weight" (100% means that in one iteration each vertex is placed exactly in the center of mass of its neighbors) Tagged sharp edges are smoothed using a one-dimensional support. If a selection is active, the subdivision works on it only.

Algorithms -> Uniform Remesh

This button strives to uniformize the vertex distribution on the model. The number of vertices in the new mesh can be specified in the dialog along with the number of relaxation steps. Sharp edges and boundaries are handled properly. Although a selection may be active, this feature works on the entire mesh in any case.

Algorithms -> Open to Disk

This operation performs a topological cut along edges in order to make the mesh homeomorphic to

a disk. Edges and vertices belonging to the cuts are properly duplicated.

Algorithms -> Spherize

Iterative flattening of regions with high gaussian curvature. Tips and sharp concavities are greedily smoothed. Works on the entire mesh.

Algorithms -> Feature Recover

When selecting this button, an Edge-Sharpener filter is run [1]. A dialog is popped up from which the user may select threshold values (default values are provided). Note that a zero value of the parameter 'threshold normal angle' implies that the algorithm automatically computes this value based on an averaging of the angles throughout the whole surface. Default values are safe for rough meshes obtained through dense and non-adaptive samplings. Works on the entire mesh.

Algorithms -> Simplify

This button pops up a dialog containing a set of parameters for the simplification. The user can set the desired number of vertices and choose whether to “prevent mesh inversion” (taking into account that in this case the processing time grows lightly). As a rule of thumb, the “prevent invertion” should be used when the surface is densely sampled on large flat regions. It is important to consider that, due to topological constraints, it cannot be guaranteed that the target number of vertices is reached. The simplification strategy implemented uses the quadric error metrics introduced by Garland. Setting the 'multiple-choice' parameter to a value greater than zero makes the method use a faster, but less precise, algortihm (the higher is this value, the slower and more precise is the simplification). Works on the entire mesh.

Algorithms -> Marching Cubes

This operation remeshes the model using a marching-cubes like pattern. The user must select the size of the grid, and the process is performed according to the Marching Intersections paradigm. This functionality is particularly useful to find a single component approximation of a set of nearly adjacent meshes. Vertices of the new mesh can be forced to coincide with vertices of the grid. In their turn, vertices of the grid can be forced to be integer values. Works on the entire mesh.

Algorithms -> Normal noise

This operation distributes Gaussian noise over the vertices in the normal direction. The amount of noise can be selected by the user in the pop-up dialog as a percentage of the model's bounding ball radius. Works on the entire mesh.

Algorithms -> Refine

This feature increases the number of vertices up to a user-selected value. Vertices are inserted one by one. Each of them is inserted in the mid-point of the currently longest edge of the mesh. If a selection is active, only edges belonging to the selection are split.

Algorithms -> Optimize

This feature iteratively swaps edges to locally maximize the minimum angle. If the mesh is flat and non-overlapping, the method converges to a 2D Delaunay triangulation of the vertices. Otherwise convergence is not guaranteed, and the method quits after 10 iterations. If a selection is active, only edges belonging to the selection are swapped.

4. Checking and Repairing a mesh

The problem of robustness of geometric algorithms has received a lot of attention from more than 15 years [8][9], and it remains a vibrant area of research. When designing a geometric algorithm, in

fact, a researcher uses the theory of real numbers and their **exact** arithmetic. Such a framework is often referred to as the Real Arithmetic Model or, more compactly, the RAM. At implementation time, however, real numbers have to be approximated with finite precision representations and the error introduced cannot always be neglected. In the context of our remeshing toolbox, such an error may perturb the geometry and thus cause the creation of degenerate elements, self-intersecting surfaces, and a number of other flaws. Although a slight error in the geometry may be simply interpreted as *noise*, when performing computations such a slight modification may cause a wrong branching in the process pipeline and, as a consequence, it may cause a topological inconsistence or a failure of the algorithm.

In order to deal with robustness, we chose to implement a strategy based on the Epsilon Geometry introduced in [12]. The toolbox provides the possibility for the user to choose a threshold angle ε. In all of the internal computations, including the loading step, the toolbox prevents the creation of triangles with angles smaller than ε or bigger than π-ε. Such a prevention is carried out through swapping and contraction of short edges, inspired from ideas of [5]. The default value of ε is $arcsin(10^{-5})$; by experiment, this value proved to be a good compromise between precision and robustness.

Even if the strategy implemented does not guarantee robustness in all of the cases, we have found that it avoids nearly all of the most common problems when dealing with remeshing. Furthermore, it is worth to be considered that if the input model has many degenerate faces (according to the value of ε) the corrected mesh may be strongly distorted. To cope with this cases, the toolbox provides a command-line option to set the value of ε to zero, but subsequent processing must take into account possible failures.

All of the check and repairing tasks can be performed by the user after selecting a new value for ε. This can be useful when the model being edited is known to become the input for a less robust system. At run time, the *Check&Repair* window provides an interactive slider to set the value for ε. The same window provides a number of buttons for further operations dealing with robustness issues; in particular the following controls are provided:

1. *Check Connectivity.* This action performs a sequence of consistency tests on the connectivity graph of the mesh. Notice that, in normal conditions, it should always pass all of the tests. However, it has been incorporated to help the developer of a new plug-in to check the status of the structure at any time.
2. *Check Geometry.* This action looks for coincident vertices, coincident edges, degenerate triangles and overlapping adjacent triangles, according to the current value of ε. If one of these situations is verified, the camera is automatically moved to a viewpoint suitable for showing the flaw, so that the user can analyze what went wrong, and choose to perform some manual editing for a possible correction. Also, the user can rely on the automatic correction approaches provided.
3. *Glue Boundaries.* This action looks for coincident edges and, if possible, merges them. This is particularly useful to topologically join several connected components that are usually created by geometric modeling tools.
4. *Remove Smallest Components.* This action scans the data structure and removes all of the connected components but the one with the largest number of triangles. This is useful for eliminating typical tiny disconnected sheets from models coming from marching-cubes or other sorts of polygonization algorithms.
5. *Orient Normals.* This action scans the data structure and, for each connected component orients the triangle normals consistently. If a connected component is closed, triangle normals are oriented so that the outer surface is visible (i.e. normals point outwards the enclosed solid).
6. *Duplicate non-manifold vertices*. This action checks for non-manifold vertices and, for each one

of them, creates a copy of the vertex for each connected component in its original VT. The incidence relations of the neighboring edges are then updated with the new vertices so as to have a single component VT for each copy. Notice that this operation is purely topological, that is, the geometry of the surface is not modified. As for the *Check Connectivity* button, however, this should have no effect in normal conditions.

7. *Remove Degenerate Triangles*. This button performs a removal of possible degenerate triangles. According to the current value of ε, all of the degenerate triangles are removed from the mesh, unless their removal would corrupt the topology; in this case, the user must manually remove these elements (see *Check Geometry* above). Note that in extreme cases (very large ε or wire-like cylinders connecting two bodies) this action may also cause topological modifications.
8. *Remove Overlapping Triangles*. This action removes all of the triangles which form an excessive dihedral angle with one of the neighbors. A dihedral angle is excessive if it is less than ε of more than π-ε.
9. *Remove Tiny Handles*. This action looks for small handles and tunnels, referred to as *topological noise* in [13], and selects them. The threshold size for handles to be removed is controlled through the spherical selection tool, which helps the user in having a visual idea of that size.
10. *Select Boundary Triangles.* This features selects all the triangles having at least one vertex on the boundary. This may be useful, for example, to "erode" range maps, in which the boundary typically incorporates most of the measurement distortion.
11. *Select Intersecting Triangles.* This selects all the intersecting triangles, and works both on self-intersections and on intersections of disconnected components. If a selection was alredy present, the test is performed for selected triangles only, and results in a sub-selection.
12. *Acos tolerance.* Controls the value of ε. Notice that a zero value makes the tool non-robust. Allowed values for ε range from 0 to arcsin(0.00499).

5. GUI Configuration

ReMesh provides some facilities for interactively setting visualization parameters of the scene being rendered. After setting these parameters, each user can store their configuration to a file for future use. Current settings are saved in a hidden file placed in the user's home dir, and are automatically applied when ReMESH will be launched in the future.

These settings include:

- windows position, size and visibility

- backgroud color of the render-area

- material settings of the rendered mesh

`Save Current`

This button saves the current configuration to a file in the user's home directory. The next time ReMESH is launched by the same user, it will start with the configuration saved.

`Load Last`

This button restores the last-saved configuration.

`Load Default`

Restores the default configuration of the GUI

Settings -> Material Editor

The material editor is a widget through which the user can control a number of settings about the material properties of the mesh(es) being displayed. According to OpenGL's definition of materials, the user may control material reaction to ambient, diffuse, specular and emissive lightening, as well as surface shininess and transparency. Through the menu button 'Edit' a material list is provided from which the user may choose a pre-defined set of material properties. While chaning parameters, the effects are interactively shown in the main scene, unless 'manual' was chosen from the 'Edit' menu.

Background Color

This is another set of controls to define the color of the background of the scene being rendered. These controls are not interactive, that is, the user is required to select OK to show the result.

Per-face color

This button toggles the status 'per-face color' which defines wether the mesh must be rendered with a single material/color throughout the whole surface, or with different colors for different triangles. When this toggle is set ReMesh shows selected triangles using a brighter color with respect to the remaining surface. Notice that this feature may slightly impact on the rendering performances.

Per-tag color

As ReMesh supports edge tagging, this button allows the user to display edges currently tagged as 'sharp' using a different color (red). Notice that this feature may slightly impact on the rendering performances.

Show Edges

This button toggles between the default flat-shading and the hidden-line + flat shading. More visualization options may be selected through the proper pull-down menu by right-clicking into the rendering area. This menu includes the old “hidden-line” visualization mode.

Visualization modes (Per-face, Per-tag and Show-edges) may be also toggled through the buttons placed on the top-left of the canvas.

6. File Operations

This menu contains buttons to add meshes to the current scene and to save the current scene to a file.

Append

This operation loads a triangle mesh from file and inserts it in the current scene without removing the existing mesh(es). Notice that no geometrical check is performed on the newly loaded mesh that may consequently have degenerate elements. To automatically remove all the degeneracies of the current meshes use the buttons provided in the Check & Repair window. The same file-formats supported when launching ReMESH are supported here.

Save as "out.wrl"

The file "out.wrl" is the default filename for "work-in-progress" models. If ReMesh was launched from the directory 'MY/PATH', then this button creates the file 'MY/PATH/out.wrl' and saves the current mesh in it. Notice that if the file already exists ReMesh automatically overwrites it without asking confirmation, because, again, this file should be considered a "work-in-progress".

Save as ...

This button pops-up a file selector from which the user may choose the name of an existing file to overwrite, or may specify a new file name for the current model. If the file exists confirmation will be asked before overwriting.

The FORMAT is chosen based on the filename extension according to the following table:

```
".wrl" -> ASCII VRML 1.0
".iv"  -> Open Inventor 2.1
".off" -> Object File Format
".ply" -> Stanford's PLY format
".obj" -> Wavefront's OBJ
".stl" -> Stereolithography
".tri" -> IMATI ver-tri
```

If no extension is provided, ReMESH automatically appends ".wrl" to the filename and saves it as an ASCII VRML 1.0 model.

7. Manifold triangle meshes

Let V be a nonempty set. An *abstract simplicial complex* [14] is a collection Σ of finite nonempty subsets of V such that every $\{v\} \in V$ belongs to Σ and if A is an element of Σ, then so is every nonempty subset of A. Each element of V is called a *vertex* of Σ. An element σ of Σ of cardinality k+1 is called a *k-simplex*, and k is the *order* of σ (hence, a 0-simplex is a vertex). A d-simplex σ is said to be *incident* at a k-simplex τ if $\tau \subseteq \sigma$.

Let V_σ be a set of d+1 affinely independent points in the 3-dimensional Euclidean space R^3, with $d \leq 3$. The subset σ of R^3 formed by the points x that can be expressed as the convex combination of the points v of V_σ:

$$x = \sum_{i=0}^{d} l_i v_i \text{ , with } \sum_{i=0}^{d} l_i = 1, l_i \geq 0$$

is called an *Euclidean d-simplex* generated by V_σ, and the points of V_σ are called the *vertices* of σ. Any Euclidean s-simplex τ generated by a set $V_\tau \subseteq V_\sigma$ of cardinality $s < d$ is called an *s-face* of σ. A finite collection Σ of simplexes is an *Euclidean simplicial complex* iff the following conditions hold:

1. For each simplex $\sigma \in \Sigma$, all faces of σ belong to Σ;
2. For each pair of simplexes σ and τ, either $\sigma \cap \tau = \varnothing$ or $\sigma \cap \tau$ is a face of both σ and τ.

Let V be a nonempty set and let Σ be an abstract simplicial complex on V. An *embedding* of Σ into R^3 is a function f: $V \rightarrow R^3$, such that:

- For every k-simplex σ in Σ, the set σ' generated by the vertices of $f(\sigma)$ is an Euclidean k-simplex;
- The collection Σ' of all the simplexes generated by f, as described above, fulfills condition (2) of the definition of Euclidean Simplicial Complex.

Thus, an embedding of an abstract simplicial complex is fully defined by mapping its vertices into points in the Euclidean space. In R^3, an abstract simplicial complex Σ endowed with an embedding f describes a *triangle mesh* M=(V,E,T) where V, E and T are the sets of 0, 1 and 2-simplexes respectively and each element of V can be mapped to R^3 through f [16].

An element of V is called a *vertex*, an element of E is called an *edge* and an element of T is called a *triangle*. Furthermore, we call *connectivity* of M the combinatorial structure of Σ, while we call *geometry* of M the function f that associates a 3D point to each vertex of V.

The set $|\Sigma| \subseteq R^3$ defined as the union of all the Euclidean k-simplexes generated by the vertices of $f(\sigma)$, for each $\sigma \in \Sigma$, is called the *geometric realization* of Σ. We say that a triangle mesh is *manifold* (possibly with boundary) if $|\Sigma|$ is a two-dimensional manifold (possibly with boundary).

8. A data structure for manifold triangle meshes

The definition of a data structure for boundary representations, such as triangle meshes, requires the coding of topological entities (with the associated geometric information) and of a suitable subset of the topological relationships between such entities. In particular, it is desirable that all of the following requirements are satisfied:

- The structure must be **complete**, that is, it must be possible to extract *all* of the entities and relationships which are not explicitly stored, without ambiguity.
- The structure should be **non-redundant**, that is, if an entity (or a relationship) can be computed in *optimal* time, then it should not be explicitly coded in the data structure.

- Each relationship which is not explicitly stored must be **computable in optimal time**, that is, the number of operations required must be linear in the number of elements of the relationship.

8.1 Scheme of the relationships

A topological relation is essentially a function which associates to each element σ of a given type (a vertex, an edge or a triangle) the set of all the elements of another given type having a topological connection with σ. For example, if V is the set of vertices of a triangle mesh M = (V,E,T), then the set of pairs $VE = \{(v, Y) \mid v \in V, Y \subseteq E \text{ and } \forall \varphi \in Y, v \subset \varphi\}$ is the topological relationship which relates each vertex with the set of all the edges incident at it. Clearly, the set of pairs VE may be viewed as a function $VE: V \rightarrow \wp(E)$ which maps each element of V into a subset of E.

Let M = (V,E,T) be a manifold triangle mesh. The following Figure 2 depicts all of the possible relationships between elements of M. Notice that a good data structure should not store them all in order to avoid redundancy.

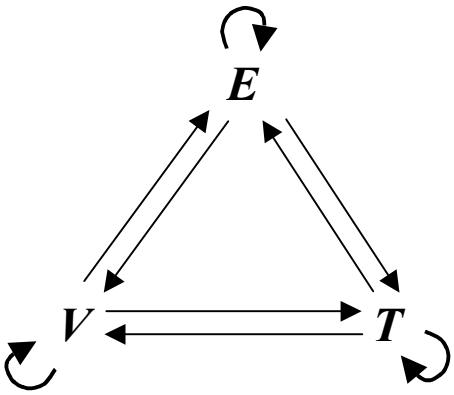

Figure 2: A scheme of all of the possible relationships between topological entities in a triangle mesh.

Let v be a vertex. VV(v) is the set of all the vertices which are connected to v through an edge[1]. VE(v) is the set of all the edges which are incident at v. VT(v) is the set of all the triangles of which v is a vertex.

Let e be an edge. EV(e) is the set containing the two ending vertices of e. ET(e) is the set of the triangles of which e is an edge. Notice that, since we consider only manifold triangle meshes (possibly with boundary), each set ET(e) contains either two elements (when e is an internal edge) or one element (when e is a border edge). The set EE(e) is the set of the edges bounding the triangles of which e is an edge, excluding e itself. Thus, if e is an internal edge, the set EE(e) contains four edges, while if e is on the boundary, |EE(e)| = 2.

Let t be a triangle. The set TV(t) is the set of the three vertices of t. TE(t) is the set of the three edges bounding t and, finally, TT(t) is the set of the triangles sharing an edge with t.

Moreover, a topological relation may map each element of its domain into a set of constant or variable cardinality. Hence, when dealing with closed triangle meshes, one can classify the relationships in:

- **Constant relations.** These relations map edges and triangles into sets of neighboring elements with constant cardinality. Specifically, |EV(e)| = 2, |EE(e)| = 4, |ET(e)| = 2, |TV(t)| = 3, |TE(t)| = 3 and |TT(t)| = 3.
- **Variable relations.** These relations are vertex-based and the cardinality of the set may vary depending on which vertex is being mapped (VV, VE and VT).

In the design of a data-structure, however, it is useful to extend the concept of constant relation to the case of manifold triangle meshes with boundary. In fact, although the cardinality of some

[1] The set VV(v) is also known as the *link* of v.

image-sets is no longer *constant*, it can assume a finite number of values. Specifically the cardinality of the EE may be 2 or 4, the one of the ET may be 1 or 2, and the one of the TT may be 0, 1, 2 or 3. All the others are still *properly* constant.

8.2 A non-redundant data structure

Clearly, a data-structure coding all the relations depicted in Figure 2 is redundant. Conversely, when dealing with manifold triangle meshes with boundary, the scheme depicted in Figure 3 meets all of the requirements discussed above [6].

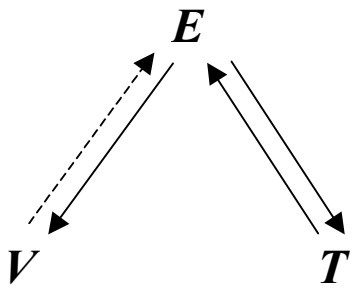

Figure 3: Scheme of relations from which it is possible to derive all of the other (non-stored) relations in optimal time. The dotted line representing the VE indicates that such a relation is only partially stored.

In Figure 3 the VE is indicated with a dotted line, meaning that such a relation is only partially stored. From now on, we denote with VE* such a restricted VE. VE*(v) maps v into **one of its incident edges**. The complete VE(v) can be computed starting from the VE*(v) by "turning around" v through successive applications of the coded relations, keeping track of the already traversed triangles. In particular, the initial VE is initialized as the VE*, then choose one triangle of the ET(VT*(v)), let it be t, and choose the edge e of TE(t) such that $v \in EV(e)$ and $e \neq VE^*$. Now add e to the set VE and repeat the same operations by considering e as the new VE*. If v is not on the boundary, the process terminates when e becomes equal to the original VE*(v). If v is on the boundary, e may become a boundary edge; In this case, the process continues by considering the unconsidered triangle of ET(VE*(v)) and turns in the opposite sense. An example of this process is depicted in the following Figure 4.

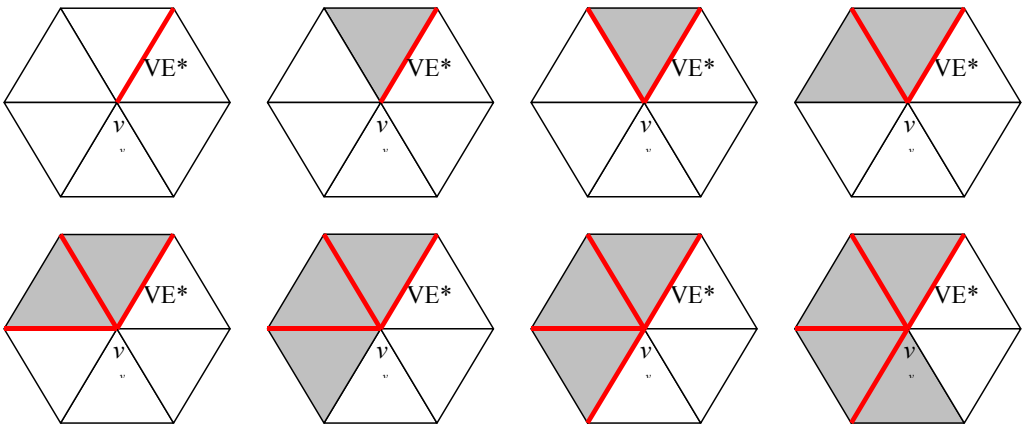

Figure 4: Reconstruction of the VE relation starting from the VE.*

Note that this process requires a number of operations that is linearly proportional to the number of elements of the final VE, therefore it is optimal.

All the other relations which are not explicitly stored in the data structure may be derived in optimal time as follows:

- VE(v) = construction described above;
- VV(v) = {w∈EV(e) | e ∈ VE(v) **and** w ≠ v}
- VT(v) = {t ∈ ET(e) | e ∈ VE(v)}
- EE(e) = {f ∈ TE(t) | t ∈ ET(e) **and** f ≠ e}
- TV(t) = {v ∈ EV(e) | e ∈ TE(t)}
- TT(t) = {y ∈ ET(e) | e ∈ TE(t) **and** y ≠ t}

Bibliography

[1] Attene, M., Falcidieno, B., Rossignac, J. and Spagnuolo, M. 2003. *Edge-Sharpener: Recovering sharp features in triangulations of non-adaptively re-meshed surfaces*. Proceedings of the 1st Eurographics Symposium on Geometry Processing, 63-72.

[2] Attene, M., Falcidieno, B., Rossignac, J. and Spagnuolo, M. 2005. *Sharpen&Bend: Recovering curved sharp edges in triangle meshes produced by feature-insensitive sampling*, IEEE Transactions on Visualization and Computer Graphics, 11, 2, 181-192.

[3] Attene, M., Falcidieno, B., Spagnuolo, M. and Rossignac, J. 2003. *SwingWrapper: Retiling triangle meshes for better EdgeBreaker compression*. ACM Transactions on Graphics, 22, 4, 982-996.

[4] Barequet, G. and Sharir, M. 1995. *Filling Gaps in the Boundary of a Polyhedron*. Computer-Aided Geometric Design, 12, 2, 207-229.

[5] Botsch, M. and Kobbelt, L. P. 2001. *A Robust Procedure to Eliminate Degenerate Faces from Triangle Meshes*. In Proceedings of Vision, Modeling and Visualization.

[6] Bruzzone, E. and De Floriani, L. 1990. *Two Data Structures for Constructing Tetrahedralizations*. The Visual Computer, 6, 5, 266-283.

[7] De Floriani, L., Falcidieno, B. and Pienovi, C. 1985. *Delaunay-based Representation of Surfaces defined over Arbitrarily Shaped Domains*. Computer Vision, Graphics and Image Processing, 32, 127-140.

[8] Forrest, A. R. 1987. *Computational geometry and software engineering: Towards a geometric computing environment*. Techniques for Computer Graphics, 23-37.

[9] Fortune, S. 1996. *Robustness issues in geometric algorithms*. In Proceedings of the 1st Workshop on Applied Computational Geometry (WACG '96), 9-14.

[10] Garland M. and Heckbert, P.S. 1997. *Surface simplification using quadric error metrics*. In Proceedings of ACM SIGGRAPH ‘97, 209-216.

[11] Guéziec, A., Taubin, G., Lazarus, F. and Horn, B. 2001. *Cutting and stitching: Converting sets of polygons to manifold surfaces*. IEEE Transactions on Visualization and Computer Graphics, 7, 2, 136–151.

[12] Guibas, L., Salesin, D. and Stolfi. J. 1989. *Epsilon geometry: building robust algorithms from imprecise computations*. ACM Symposium on Computational Geometry, 5, 208-217.

[13] Guskov, I. and Wood, Z. 2001. *Topological noise removal*. In Proceedings of Graphics Interface ‘01, 19-26.

[14] Hatcher, A. 2002. *Algebraic Topology*. Cambridge University Press, UK.

[15] Kobbelt, L. 2000. *Sqrt(3)-subdivision*. In Proceedings of ACM SIGGRAPH '00, 103-112.

[16] Lee, A. W. F., Sweldens, W., Schröder, P., Cowsar, L. and Dobkin, D. 1998. *MAPS: Multiresolution adaptive parameterization of surfaces*. In Proceedings of ACM SIGGRAPH ‘98, 95-104.

[17] Liepa, P. 2003. *Filling Holes in Meshes*. In Proceedings of the Eurographic Symposium on Geometry Processing, 200-206.

[18] Loop, C. 1987. *Smooth subdivision surfaces based on triangles*. Master’s thesis, University of Utah (USA), Department of Mathematics.

[19] Rocchini, C., Cignoni, P., Ganovelli, F., Montani, C., Pingi, P. and Scopigno, R. 2001. *Marching Intersections: an efficient resampling algorithm for surface management*. In Proceedings of Shape Modeling International (SMI ’01), 296-305.

[20] Schroeder, W., Zarge, J. and Lorensen, W.E. 1992. *Decimation of triangle meshes.* In Proceedings of ACM SIGGRAPH ‘92, 65-70.

[21] Zorin, D., Schröder, P. and Sweldens, W. 1996. *Interpolating subdivision for meshes with arbitrary topology*. In Proceedings of ACM SIGGRAPH '96, 189-192.