Istituto di Matematica Applicata e Tecnologie Informatiche

Consiglio Nazionale delle Ricerche

Genova - ITALY

ReMESH

Version 2.1

An interactive and user-friendly environment for remeshing surface triangulations

MARCO ATTENE

USER MANUAL

Index

1. Basics

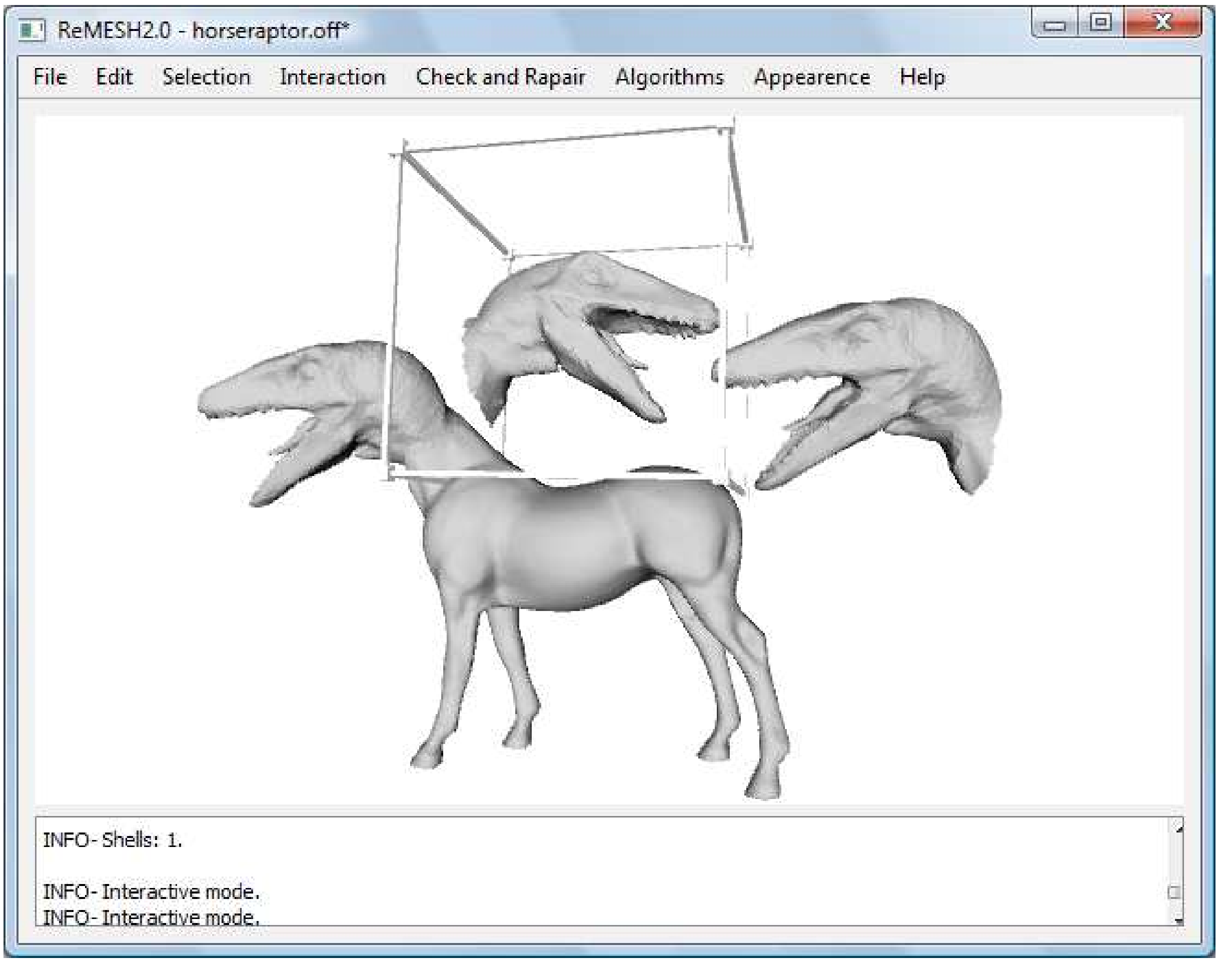
ReMESH is an editor for manifold triangle meshes. It has been developed to finely post-process polygon meshes coming from digitization sessions, but it can also be used to edit meshes produced differently.

The mesh being edited is **manifold**, **oriented** and made of **triangles** only: if the input file does not satisfy these requisites, the loader performs automatic conversion algorithms, that is, it triangulates non-triangular faces, duplicates singularities and re-orients the triangles in a consistent manner (non orientable meshes are cut).

ReMESH works on a single mesh, possibly made of several connected components (the *shells*).

Besides launching the operations selectable from the menus, the user may interact with the mesh in either **inspection** (the default) or **interactive** mode.

In *inspection* mode, mouse clicks and drags within the *canvas* (i.e. the area where the 3D scene is depicted) are used to control the visualization of the mesh, namely to translate or rotate the scene, to zoom in and out, to choose among flat shading or wireframe, and so on.



```
INFO- Shells: 1.

INFO- Interactive mode.
INFO- Interactive mode.
```

Figure 1: A snapshot of the tool. Besides standard window decorations and menus, the GUI is made of a canvas (the big white rectangle containing the shapes being edited) and an information box (the smaller white rectangle in the bottom containing information about the processes performed).

In *interactive* mode, mouse clicks and drags are used to perform actions on specific portions of the mesh, such as selecting triangles or shells, swapping edges, and so on.

The user may switch between inspection and interaction modes by hitting the space bar on the keyboard.

As for any standard editor, a **selection mechanism** is provided. In ReMESH, a selection is a collection of triangles, that is, the triangle is the atomic entity treated (just as characters for a word processor, or pixels for an image processor).

1.1 Terminology

ReMESH is a Graphical User Interface (GUI) composed of:

- A menu bar;
- A **canvas**, which is the area in which the 3D scene is depicted;
- An **information box**, which contains textual messages about the processes performed.

The **3D scene** depicted in the canvas is constituted by the mesh along with possible manipulators, selection tools, or any other graphical entity. The **mesh** is the part of the scene being edited. The mesh is a unique entity, but it can be made of several connected components called **shells**.

Each shell may be closed or it may have one or several boundary loops.

2. Loading stage

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 and Stereolithography (.stl). 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 asks the user if he/she wishes to perform some geometrical checks and corrections (removal of degenerate triangles, identification of edges having coincident end-points, …).

If the user wants to operate on more models at the same time, the "Append" function may be used to load a triangle mesh from file and to insert it as a new shell(s) in the current mesh without removing the existing shell(s). Notice that geometrical check is not proposed for the newly loaded shell that may consequently have degenerate elements. To automatically remove all the degeneracies of the current shells use the functions provided in the Check & Repair menu.

3. Inspection Mode

The graphical part of the toolbox is built upon Coin3D [22] (an extended version of 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 is provided by an Inventor Examiner viewer component. It allows the user to rotate the view around a point of interest using a virtual trackball (left click and drag). This viewer also allows the user to translate the camera in the viewer plane (middle-click and drag), as well as dolly (move forward and backward, through left+middle click and drag or through the mouse wheel) 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 (right click and select “functions → seek” from the popup menu).

Further interaction parameters can be controlled through the popup menu.

4. Interactive Mode

In interactive mode, the user can define and work with selections, and can pick specific surface points to work with. Normally, operations work on the entire mesh unless a selection is specified. If a selection has been specified, operations work on it exclusively. The menu bar is highlighted when a selection is active to make the user aware of the state even if selections are not visible.

4.1 Selections

Clicking on a point of the mesh surface using the right mouse button sets the center of a **spherical selection tool**. By holding the right-button and dragging the mouse, it is possible to modify the radius of the selection. Each triangle having all the three vertices within the sphere is selected. The selection can be expanded or contracted by rotating the mouse wheel, or through the “Grow” and “Shrink” entries of the “Selection” menu.

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.

The usual shortcuts **CTRL+C**, **CTRL+X** and **CTRL+V** can be used to perform **copy**, **cut** and **paste** operations respectively, while the **DEL** key deletes all the selected triangles from the mesh. In ReMESH, after a *paste* operation the newly inserted shells are attached to a manipulator for possible transformations until the user hits the **ESC** key (see “Transform Shells” in the “Interaction” menu for details).

4.2 Interactive operations

The behavior of the left mouse button in interactive mode is user-defined through the choice of an entry in the “Interaction” menu. The default behavior is “Select → Triangles”, which means that a left click on the mesh surface causes the selection of the triangle picked (or its de-selection if already selected).

5. Undo

ReMESH provides an “Undo” functionality that can be called through the standard **CTRL+Z** shortcut. Triangle meshes may be extremely complex, and it is not unusual to work with millions of triangles in a single session; in these cases, recording the previous mesh version for the undo might become an unnecessary waste of time and memory. For these cases, ReMESH provides the possibility to disable the undo functionality (see “Disable Undo” in the “Edit” menu).

Since the definition of a selection can be a time-consuming task, operations on selections are considered as ordinary operations, therefore the “undo” makes it possible to restore the previous selection as well.

6. File

- **`Open`**

Loads a polygon mesh from file and replaces the current mesh with the newly loaded mesh. Supported file formats are VRML (1.0 and 2.0), Open Inventor (.iv), Object File Format (.off), Stanford's PLY, Wavefront's OBJ and Stereolithography (.stl). See "Loading stage" for details on partially supported formats and "Import " to know how to convert scene graphs to triangle meshes.

- **`Append`**

Loads a polygon mesh from file and adds it to the current mesh without removing the existing shells.

- **`Reload`**

Loads the lastly saved version of the current mesh.

- **`Save`**

Saves the current mesh to file.

- **`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.

- **`Import → Scene Graph`**

This function allows to load an Open Inventor or VRML file containing a generic scene graph, not necessarily a polygon mesh. The scene graph is virtually rendered by the underlying Coin3D subsystem, and the resulting triangles are used to build the resulting new mesh.

- **`Import → 3D Image`**

This function allows to load a 3D image stored in Analyze™ 7.5 format and to convert it to a triangle mesh by executing a marching-cubes algorithm. Besides selecting the threshold value for iso-surfacing, the user may choose to subsample the input image prior to conversion (3D images can be huge).

- **`Export → 3D Image`**

This function rasterizes the mesh to a binary 3D image and saves it in Analyze™ 7.5 format. The user is asked to select the size of the grid.

- **`Snapshot ...`**

Converts the currently visible canvas to an image file.

- **`Properties ...`**

Shows information about the current mesh (number of vertices, boundary loops, …).

7. Edit

- **`Undo`**

Restores the mesh as it was before the last operation. Only one undo step can be done (double undo is equivalent to no undo).

- **`Disable Undo`**

Disables the undo function to save memory and somehow (actually very little) speed-up the operations. The undo can be re-enabled later on.

- **`Copy / Cut / Paste / Delete`**

The copy and paste mechanism is standard. Selected triangles are copied (as a mesh) to the clipboard, and then pasted as new shells. After a *paste* operation the newly inserted shells are attached to a manipulator for possible transformations until the user hits the **ESC** key (see “Transform Shells” in the “Interaction” menu for details). The *delete* process removes all the triangles currently selected, and takes care of maintaining a coherent data-structure, possibly by duplicating non-manifold vertices that can be created due to the removal.

- **`Normalize mesh`**

The mesh is resized and translated so that its bounding box fits in the cube [0,1]x[0,1]x[0,1].

- **`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.

- **`Flip Normals`**

Invert the orientation of all the mesh triangles.

- **`Tag Sharp Edges ...`**

This feature allows the user to automatically tag as sharp (see the “Tag Sharp Edges” in “Interaction” menu) 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).

- **`Tag Plane Borders ...`**

This function subdivides the mesh in nearly flat connected regions and tags their boundary as sharp. Each single region can be selected through “Interaction→Select→Sharp-bounded region”.

- **`Join closest shells`**

Change the topology of the mesh by connecting the boundary of two different shells with a pair of new triangles. Among all the possible pairs of shells with boundary that can be joined, the pair with closest vertices is chosen and the closest vertices are joined.

8. Selection

- **`Grow`**

Adds to the current selection all the triangles having at least one vertex on the border of the selection. Equivalent to moving the mouse wheel forward.

- **`Shrink`**

Removes from the current selection all the triangles having at least one vertex on the border of the selection. Equivalent to moving the mouse wheel backward.

- **`Invert`**

Toggles the selection status of all the triangles.

- **`Select Boundary`**

This features selects all the triangles having at least one vertex on the boundary of the mesh. This may be useful, for example, to "erode" range maps, in which the boundary typically incorporates most of the measurement distortion.

9. Interaction

- **`Select → Triangles`**

When this toggle is set (the default), each triangle the user clicks on is marked as "selected". If the user clicks on a previously selected triangle, it changes its state to "unselected".

- **`Select → Vertex neighbors`**

When the user clicks on (or nearby) a vertex, all the triangles incident at that vertex are selected. The **SHIFT** and **CTRL** modifiers work as in the case of the spherical selections (see “Selections”).

- **`Select → Shells`**

When this toggle is set, the shell containing the triangle clicked is selected. The **SHIFT** and **CTRL** modifiers work as in the case of the spherical selections (see “Selections”).

- **`Select → Sharp-bounded 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*. The **SHIFT** and **CTRL** modifiers work as in the case of the spherical selections (see “Selections”).

- **`Select → Connected selection`**

When this toggle is set, the triangle clicked is used to define the only component of the current selection that remains selected.

- **`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.

- **`Flip Shells`**

The normals of all the triangles in the shell clicked are inverted.

- **`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).

- **`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 “Angular tolerance” chosen by the user in the “Check & Repair” menu. 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.

- **`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] and [4] which try to minimize bad behaviors such as extreme dihedral angles, degenerate triangles, and so on. No new vertex is inserted.

- **`Fill Holes`**

This works as “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, as described in [17].

- **`Tag Sharp Edges`**

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. These edges will receive particular treatment during some other processes (e.g., “Subdivision → Bender”) and can be used to define the boundary of regions to be selected through “Select → Sharp-bounded region”.

- **`Transform Shells`**

When this toggle is set, a click on a shell results in the attachment of a manipulator to that shell. The manipulator might be used to translate, rotate and scale the shell independently of the remaining mesh. To remove the manipulator hit the **ESC** key on the keyboard.

- **`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 change 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.

- **`Restore interaction`**

When ReMESH is waiting for specific interactions (e.g. selection of the second point for “Join Boundaries”, removal of a transformation manipulator) the user may tell ReMESH not to wait any more by using this function or, equivalently, by hitting the **ESC** key.

- **`Toggle Viewing`**

Switches between inspection and interactive modes (see “Inspection Mode” and “Interactive Mode”).

10. Check & Repair

- **Check Geometry**

This action looks for coincident vertices, coincident edges, degenerate triangles and overlapping adjacent triangles, according to the current value of ε set through "Angular tolerance". 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 the mesh, and choose to perform some manual editing for a possible correction. Also, the user can rely on the automatic correction approaches provided.

- **Remove Degenerate Triangles**

This button performs a removal of possible degenerate triangles. According to the current value of ε (see "Angular tolerance"), 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 through interactive operations (see "Interaction"). Note that in extreme cases (very large ε or wire-like cylinders connecting two bodies) this action may also cause topological modifications.

- **Remove Overlapping Triangles**

This action removes all of the triangles whose normals form an excessive angle with the normals of the neighbors. Such an angle is "excessive" if it is less than ε (see "Angular tolerance").

- **Select Intersecting Triangles**

This selects all the intersecting triangles, and works both on self-intersections and on intersections of different shells. If a selection was already present, the test is performed for selected triangles only, and results in a sub-selection. This operation works with exact arithmetic described in [21].

- **Glue Boundaries**

This action looks for **exactly** 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.

- **Remove Smallest Shells**

This action scans the mesh and removes all of the shells 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.

- **Select Tiny Handles**

This action looks for small handles and tunnels, referred to as *topological noise* in [13], and selects them. After clicking on this menu entry, ReMESH asks the user to perform a spherical selection through the spherical selection tool (see "Selections"); such an operation, however, does not result in an actual selection, while it is used to define a threshold size for handles to be selected.

- **Angular tolerance**

In order to deal with robustness, ReMESH implements 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 subsequent computations, ReMESH 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.

Notice that a zero value makes the geometric computations non-robust. Allowed values for this parameter range from 0 (no threshold, ε=0) to 100 (ε = *arcsin*(0.01)).

11. Algorithms

- **`Subdivision → Mid-point`**

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.

- **`Subdivision → Loop`**

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.

- **`Subdivision → 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.

- **`Subdivision → Sqrt(3)`**

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.

- **`Resampling → Uniform Remesh`**

This button strives to make the vertex distribution uniform 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. Note that vertices are moved on their tangent plane, which does not guarantee that they lie on the initial surface.

- **`Resampling → 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 slightly). As a rule of thumb, the “prevent inversion” 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 matrices (QEMs) introduced by Garland. If the 'priority on edge length' is checked, then edge length is used instead of QEMs to drive the simplification (shortest edges are collapsed first). Works on the entire mesh.

- **`Resampling → Marching Cubes`**

This operation remeshes the model using a marching-cubes like pattern. The user must select the size of the grid (i.e. the number of voxels per side), and the process is performed according to the Marching Intersections paradigm [19]. This functionality is particularly useful to find a single component approximation of a set of nearly adjacent shells. Works on the entire mesh.

- **`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.

- **`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.

- **`Delaunize`**

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.

- **`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. Tagged sharp edges are smoothed using a one-dimensional support. If a selection is active, the smoothing works on it only.

- **`Spherize`**

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

- **`Edge Sharpener`**

When selecting this button, the Edge-Sharpener algorithm is run [1]. A dialog is popped up from which the user may select threshold values (default values are provided). Default values are safe for rough meshes obtained through dense and non-adaptive samplings. Works on the entire mesh.

- **`Add 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.

- **`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” toggle is set too, the newly inserted vertices are moved so as to produce a tangentially continuous patch with respect to its boundary [17]. If a selection exists, this button limits its action to the selected region. Only completely selected boundaries are filled.

A.1 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 $\sigma \cap \tau = \emptyset$ 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).

A.2 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.

A.2.1 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.

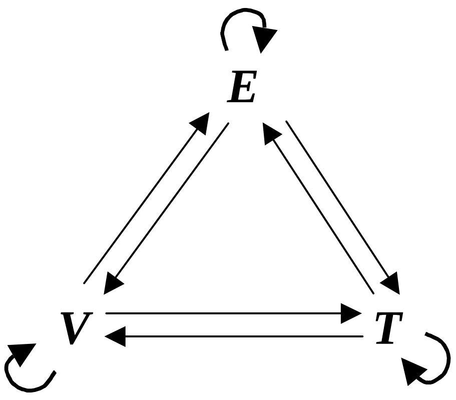


Figure 2: A scheme of all of the possible relationships between topological entities in a triangle mesh.

Let *v* be a vertex. VV(*v*) is the set of all the vertices which are connected to *v* through an edge. 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 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.

A.2.2 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].

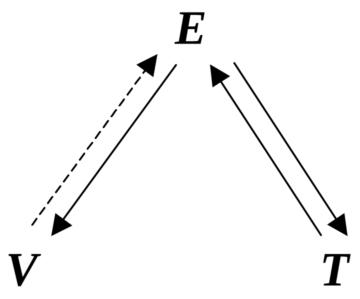


Figure 3: Scheme of relations from which it is possible to derive all of the other (non-stored) relations in optimal time. The dotted line representing the VE indicates that such a relation is only partially stored.

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 ∈ EV(*e*) and e ≠ 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.

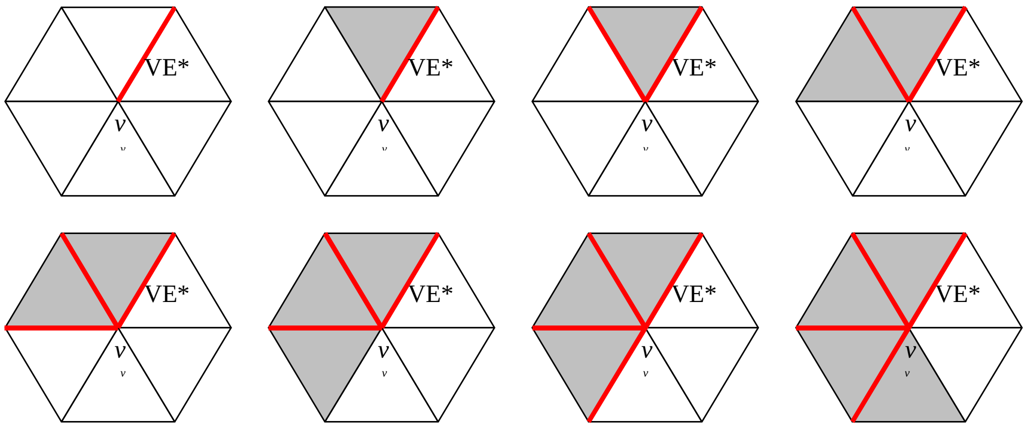


Figure 4: Reconstruction of the VE relation starting from the VE.*

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) = $\{w \in EV(e) \mid e \in VE(v)$ **and** $w \neq v\}$
- VT(v) = $\{t \in ET(e) \mid e \in VE(v)\}$
- EE(e) = $\{f \in TE(t) \mid t \in ET(e)$ **and** $f \neq e\}$
- TV(t) = $\{v \in EV(e) \mid e \in TE(t)\}$
- TT(t) = $\{y \in ET(e) \mid e \in TE(t)$ **and** $y \neq t\}$

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] Shewchuk, J. R. 1997. *Adaptive Precision Floating-Point Arithmetic and Fast Robust Geometric Predicates*. Discrete & Computational Geometry, 18, 305-363.
[22] Systems in Motion, “Coin3D”. http://www.coin3d.org.
[23] Zorin, D., Schröder, P. and Sweldens, W. 1996. *Interpolating subdivision for meshes with arbitrary topology*. In Proceedings of ACM SIGGRAPH '96, 189-192.