IVECS, Interactively Correcting .STL Files in a Virtual Environment

Stéphane M. Morvan
smorvan@eng.clemson.edu

Georges M. Fadel
gfadel@eng.clemson.edu

Design Methodology Group
College of Engineering and Science
Clemson University,
Clemson, SC 29634–0921

ABSTRACT

Free Form Fabrication (FFF) machines transform objects merely existing as 0s or 1s in a computer into a tangible object. FFF machines shift the paradigm of standard 2 Dimensional printers/paper printouts to 3 Dimensional printers/volumetric printouts (or 3D hardcopies). Currently, this technology is weakened by the link between computers and FFF machines: the .STL file, which contains a series of triangles representing the skin of the object to be prototyped. A prototype, reflecting precisely the evolution of a concept within a design cycle and allowing a systematic inspection/verification, is essential. A system responding to this need was designed at Clemson University for the inspection and the correction of such a file. IVECS, the Interactive Virtual Environment for the Correction of .STL files, is a tool that allows minute surgery to be performed on faulty tessellated models. IVECS allows STL files to be imported, tessellation errors to be detected and automatically or manually fixed. This paper expands on the use of IVECS for the inspection and the correction of .STL files. It extends the usefulness of the STL format by allowing designers to virtually prototype before actually building a physical model, thus contributing to a shorter design cycle.

INTRODUCTION

Bad .STL files have caused RP users to develop tools to correct errors within .STL files. These errors result from a poor tessellation algorithm that generates bad models not respecting the .STL format. The .STL file format [1] defines an .STL model as a set of triangles forming the material boundary. It specifies that an edge shall be shared by two and only two triangles and that each triangle obeys the right–handed rule (the vertices are ordered such that the cross product of the two edges should give an outward normal). Clearly, the triangles should intersect only at their edges and have exactly three adjacent triangles, one along each edge. Furthermore, an .STL model should be "water–tight": if water is poured inside the model, it should not leak. This is a necessary condition to generate a physically valid part.

Within an .STL file, several types of errors can be uncovered: those due to a weak tessellation algorithm in the CAD system which result in misoriented facets (e.g. outward normal not consistent throughout the model) and non–closed shells (presence of gaps or holes) and others due to the CAD user's ignorance of .STL files requirements (internal walls, self intersecting geometry). The internal walls problem does not really hinder the building of the part, but it may cause problems to the finished prototype. In the case of the stereolithography process in which a laser beam hardens the resin, the areas having internal wall problems are polymerized twice, resulting in overcure and possible warpage of surfaces.

PREVIOUS WORK

Most tessellation algorithms find their origins in the algorithms used by the CAD system to compute solid views (e.g. smooth shaded views). CAD systems rely at some point on an unevaluated, surface–based representation of solids: roughly all solids are assembled in surfaces (The B–Rep, or Boundary Representation). Each surface is mathematically defined by a function of two variables. These two variables are contained within a finite interval named the parameter space. When a surface is partially used within a CAD model (e.g. it is trimmed), the number of constrains on the parameters increases, leading to a more complicated parameter space [2]. A subdivision algorithm transforms surfaces into triangles by recursively dividing surfaces. When the surfaces obtained are flat enough, they are replaced by triangles. The process of dividing surfaces is a trivial problem [3]. In practice, surfaces are replaced by sculpted models, and the ordinary parameter space becomes increasingly complicated as surfaces get trimmed across most of their edges to form the desired geometry. Furthermore, the difficulty of correctly tessellating a model stems from boundary curves, which result from the intersection of two surfaces. A boundary curve belongs to two adjacent surfaces, and as such, is evaluated/sampled twice (once for each surface) during the tessellation. If the parameter space used to sample the curve differs slightly from one surface to the other, non–coincident vertices may emerge at the junction of the two surfaces.

Rock and Wozny[4] introduced the preliminary steps in correcting .STL files. The principle is that of inferring topology information from a "Bucket of Facets" (e.g. the unordered list of triangles) which is achieved using efficient detection techniques and intuitive data structures. The paper presents its "Topology Reconstruction Concepts" as based on the merger of vertices, the creation of faces and edges, and by determining face & edge relations. The merger of vertices consists of unambiguously identifying vertices listed throughout the .STL file, and assigning them a unique entry pointing to a list of vertices. The unambiguous identification of all equivalent vertices is difficult to achieve because of the round off error mentioned earlier; hence the requirement for a robust search/add algorithm to be used with a convenient data structure. The face and edge creation is straightforward from this point: a triangle and an edge are identified as a combination of respectively three and two integers that point to the list of vertices. Then, the face and edge relationships are created. Determining the face relationships is done by searching faces that share an edge, and cross referencing them; similarly the edge relationship is created (edge relationships are useful while slicing the part.)

The information gathered here (the indexed representation and the face relationships) clearly identifies errors in the file, and these serve as a basis for many related papers.

Böhn and Wozny[5] presented a simplistic, efficient approach to the correction of errors within a faulty .STL file. An intuitive algorithm to correct triangle orientation is presented, which is based on an indexed representation of the facets. The indexed representation is the first step for correcting .STL files, it consists of extracting and listing all the vertices forming the corners of the facets in a separate table. Triangles are now specified using pointers to the appropriate vertices. Using this, checking the orientation of triangles is straightforward:the neighbors of a triangle made from vertices #u, #v and #w must contain an indirect permutation of either #u and #v, #v and #w or #w and #u in the list of vertices. The iterative patching of triangles over holes (or punctures) with a "lid–surface" is outlined. Once a puncture has been detected, a closed curve made from the free edges is identified and is filled with facets of harmonious proportions:if a small triangle is near a big triangle, their common diagonal is flipped to create two equally sized triangles.

Mäkelä and Dolenc[6] described a set of procedures for correcting errors in .STL files, based on an adaptive space subdivision data structure. Several methods to automatically correct errors are presented: triangles are oriented using graph theory and small gaps are filled. Filling gaps attempts to preserve the local smoothness by considering the relative orientation of triangles rather than their size (as did the previous algorithm). Finally, the authors conclude by raising a philosophical issue while correcting faulty files: the protection of the original intents of the designer. More importantly, the authors mention the difficulty of algorithmic based approaches. They suggest what benefits could be drawn from an interaction between the user and the system.

More recently, Martin [7] proposed a public domain implementation of a correction suite for .STL files, ADMESH. The system performs surprisingly well on simple problems (misoriented normals and disjoined edges), but the patching of a hole is a more problematic task that works only in some cases. Also, ADMESH allows parts to be scaled, translated, mirrored and grouped, helping RP users to manipulate .STL models.

IVECS, THE INTERACTIVE VIRTUAL ENVIRONMENT FOR CORRECTION OF .STL FILES

The primary goal for IVECS was to allow users to virtually have their "hands on" .STL models by proposing the visualization and the correction of .STL files. IVECS detects errors and points them out to the user, who may choose to correct them. IVECS' ability to automatically solve errors is limited to normal orientation problems and round–off errors. These two are performed automatically to reduce the number of errors present throughout the file, so that the user is solicited solely for the patching of holes or the removal of extraneous facets (e.g. the internal walls problems). Mainly, IVECS supports the addition, the removal, and the reversal of facets. Additionally, the offset of facet is also proposed. IVECS supports

both GUI (Graphical User Interface) and a VR based user interface, although the latter does not offer a comparable level of versatility.

IVECS uses OpenInventor [8] as a rendering engine. OpenInventor is available from Silicon Graphics as a toolkit designed to ease the manipulation of 3D objects. The rendering can be done in a variety of ways: with hidden lines, to show the tessellation, flat shaded, to show the product as it will built on an SLA apparatus, and finally smoothed, which shows the model with rounded edges, comparable to a sanded prototype. The initial goals for this system were to support the reversal of a facet, its removal or the patching of a crack in both the virtual environment and the conventional desktop based user interface. From this, and from geometries complicated to visualize, several functions emerged. These functions can be separated in two groups:

- Mode functions: these set the system in a particular type of mode such as "transparent mode" or "chrome plated mode." Mode functions are not exclusive: a part can be chrome plated and transparent at the same time.

- Facet functions: these are the functions acting on facets. Some of these are:reversal of the normals, removal of a facet or creation of a set of contours.

On the left side of the rendered view is the toolbar. The upper half of the toolbar is dedicated to mode functions while the lower half is dedicated to facet functions. Most functions offer a "power user" mode activated by holding the shift key on the keyboard while selecting the button. This "power user" alternative offers additional control over each function. The upper and the lower half of the toolbar are separated by the "selection of facets" tool, that does not really belong to any of the previous groups. The selection of the facet tool allows facets to be selected. When this function is activated, the cursor changes from an arrow to a cross hair; and a facet is highlighted (e.g. selected) whenever it is clicked. Also, the multiple selection of facets is allowed when the shift key is held at the time the selection occurs, which results in several facets highlighted in red. A facet gets deselected if it is selected twice, and all facets are deselected if the mouse is positioned and activated on the empty space surrounding the artifact. The action of selecting the facets prepares the input for the facet functions. A set of facets can be selected and the mode functions can be accessed without interfering with the current selection.

MODE FUNCTIONS

The mode functions consist of: transparent, backface culling, errors, chrome, virtual reality, vertex and slicing plane. Toggling one mode on/off usually results in visible changes in the display area (the artifact gets rendered differently) and in the update of the icon placed on the button which reflects the state (on/off) of the current mode.

Transparent Mode 

The part is rendered as if it is translucent to light, revealing internal details. The rendering of transparent parts uses an additive blending process, resulting in a greater opacity where many facets are aligned with the direction of sight.

Backface Culling Mode 

Rendering can be quickened if one side of a facet is drawn on the screen. To achieve this, an assumption must be made on the side to render: the system computes the outward normal of each facet and determines if the facet is visible from the current point of view. When backface culling is on, the misoriented facets are not rendered correctly (visually behaving like holes in the skin of the artifact). By contrast, all facets are rendered when backface culling is off. Thus, by successively turning backface culling on and off, misoriented facets clearly appear in the rendered area.

Errors Mode 

The Errors mode displays the errors found in the .STL file. The errors are displayed by coloring problematic facets with a shade differentiable from the overall color of the part. For instance, facets missing one neighbor are colored in pink, and those having an extraneous facet are colored in yellow.

Chrome Plated Mode 

This mode makes the part look chrome plated. A texture is placed inside a sphere, and the reflection of that texture on the part is computed. Thus whenever the part is moved, the reflection is recomputed. Though this dynamic texturing scheme is computationally intensive, it results in an increased understanding of the geometry presented to the user. The principles which make this technique so efficient at grasping most of the geometric features in a single glance are based on surfaces interrogation techniques. Surfaces interrogation techniques [9] were used by car makers to match a stylist's freehand sketch with the machined prototype that emanated from it. Briefly, the technique consisted of observing the reflection of neon lights on the polished surface. The increased amount of information received from the reflection of a known pattern on a polished surface resulted in an intuitive, natural way of assessing the curvature of a surface. The chrome effect promotes the same type of natural experience with the user, and unexpected waviness resulting from a poor surface management at the CAD level is easily flagged. Several type of textures can be used: a scenic view consisting of flowers, sand and a blue sky and red stripes on a white background have been used to date.

Virtual Reality Mode 

This mode allows the user to use Virtual Reality (VR) based techniques to interact with the system. After toggling it on, the user puts on the Head Mounted Display (HMD) and step in the Virtual Environment (VE.) Initially, the artifact is floating in the VE. Several interaction/navigation means have been established:the user can either "grab" the part, enlarge it to view details or navigate around it. [10] details the VE interaction techniques used in IVECS.

Clipping plane 

The slicing plane is a recent addition to the system. It features a moving plane that prevents facets and vertices lying in its negative semi space to be drawn. The positive semi space is defined by the normal of this plane, the negative semi space being the other half. This plane can be conveniently oriented, allowing artifacts to exhibit their interiors. Also, it intuitively simulates the building process: its visual reconstruction from bottom to top helps to assess the best built direction and eventually confirm the result obtained later from a preprocessor of .STL files. Additionally, an auxiliary editor can be displayed to rotate the clipping plane or to key in the normal of the clipping plane.

Facet functions 

The facet functions are the heart of this system:they allow the modification of the artifact in several ways. Facets can be added, deleted, offset, reversed, and color–coded contours scheme can be produced. Any of these functions affects only the selection, resulting in changes in the internal data structures. No undo function is provided, since an error can be easily recovered by either recreating a facet (case of an accidental deletion), removing it (case of the erroneous creation of a facet) or "re–reversing" it (case of the accidental reversion of a correctly oriented facet). They often result in minute changes in the geometry, as opposed to the immediately observable changes produced by the mode functions.

Creation of colored contours 

To help visualize problems, sets of contiguous triangles are displayed in different colors: the neighbors (sharing one vertex) of a specific triangle are colored in red, the neighbors ("sub–neighbors") of these neighbors are colored in green, the next one in blue, and so on, repeating red, green and blue patterns concentrically around the selected triangle. Whenever a triangle is flagged as problematic, its insertion in the current set of sub–neighbors is delayed to the next set of sub–neighbors. Thus, problematic facets break the color sequence:

instead of say, being colored in green they are colored in blue and the skipped color immediately flags the error and its effect on the arrangement of vertices. Also, non–intersecting solids can be flagged easily: since these isolated artifacts do not share common vertices, the contours do not spread from one artifact to the other, resulting in the coloring of one of the artifact while the other one is left untouched. Additionally, the colored contours can be transformed in an active selection, equivalent to individually selecting all the colored facets.

Patching of a hole 

When filling holes, two approaches were considered: one trivial approach was vertex based while the other was triangle based. In both cases, the creation of the patch was subject to a test ensuring that the triangle was actually filling a hole. In the first case, once three appropriate vertices were selected (that is, vertices near an error), a triangle was computed, regardless of the arrangement of the vertices in space and allowing erroneous triangles (triangles not matching the geometry) to be created. It appeared that the adjacencies of the selected vertices had to be considered when the user attempted to map a patch.

An alternate approach was proposed: instead of selecting vertices, the user selects a pair of problematic triangles sharing one vertex and the system builds a triangular patch over the hole between the two triangles. If one (or both) of the triangles selected has more than one missing neighbors, several patches could be computed. In that case, the user browses through the set of possible solutions by repeatedly clicking on the mouse. The process is repeated until the hole is filled. If the Errors mode is on, the set of errors will be updated each time a new facet is created:pink facets are "moving" around the remaining holes, and when the surface finally gets closed, they (the pink triangles) totally disappear. This dynamic update guides the user towards achieving shell closure by drawing attention on remaining problems or newly created ones.

CONCLUSION

An environment to deal with .STL files and interactively correct them has been presented. It consists of both algorithmic tools to deal with easily correctable errors, and interactive tools that allow minute surgery. A short overview of some of the functions available in IVECS expands upon some of the techniques used. IVECS offers the opportunity to correct erroneous facets in a relatively short amount of time, but more importantly, provides the user with a tool to virtually prototype before building a physical artifact. After using such a tool, Rapid Prototyping facilities will increase their throughput since build problems will be eliminated.. Additionally the ability or visualizing a part in a VE complements ideally RP machines by offering a "print preview" function comparable to those found on popular word processors. It even helps the designer identify errors in the design before investing in a physical prototype.

REFERENCES

[1]. 3D Systems Inc., StereoLithography Interface Specifications. Valencia, CA: 3D Systems Publications, 1989.

[2]. H. Toriya and H. Chiyokura, "Chapter 8: Boolean Operations," in 3D CAD: Principle and Applications: Springer Verlag, 1991, pp. 182–195.

[3]. X. Sheng and U. Tucholke, "On Triangulating Surface Model for SLA," presented at The Second International Conference On Rapid Prototyping, Dayton, OH, 1991.

[4]. S. J. Rock and M. J. Wozny, "Generating Topological Information from a "Bucket of Facets"," presented at Solid Freefrom Fabrication Symposium, Austin, TX, 1992.

[5]. J. H. Böhn and M. J. Wozny, "Automatic CAD–model Repair: Shell–Closure," presented at Solid Freefrom Fabrication Symposium, Austin, TX, 1992.

[6]. I. Mäkelä and A. Dolenc, "Some Efficient Procedures for Correcting Triangulated Models," presented at Solid Freeform Fabrication Symposium, Austin, TX, 1993.

[7]. A. D. Martin, "ADMESH 0.93," .

[8]. J. Wernecke, The Inventor Mentor: Addison–Wesley, 1994.

[9]. A. F. Lennings, J. C. Peters, and J. S. M. Vergeest, "An Efficient Integration of Algorithm to Evaluate the Quality of Freeform Surfaces," Computer & Graphics, vol. 19, pp. 861–872, 1995.

[10]. S. Morvan and G. Fadel, "IVECS: An Interactive Virtual Environment for the Correction of .STL Files," presented at the ASME Design Technical Conferences, U. of California at Irvine, Irvine, CA, 1996.