Chapter 13
Software Issues for Additive Manufacturing

13.1 Introduction

It is clear that Additive Manufacturing would not exist without computers and would not have developed so far if it were not for the development of 3D solid modeling CAD. The quality, reliability, and ease of use of 3D CAD have meant that virtually any geometry can be modeled, and it has enhanced our ability to design. Some of the most impressive models made using AM are those that demonstrate the capacity to fabricate complex forms in a single stage without the need to assemble or to use secondary tooling. As mentioned in Chap. 1, the WYSIWYB (What You See Is What You Build) capability allows users to consider the design with fewer concerns over how it can be built.

Virtually every commercial solid modeling CAD system has the ability to output to an AM machine. This is because the only information that an AM machine requires from the CAD system is the external geometric form. There is no requirement for the machine to know how the part was modelled, any of the features or any functional elements. So long as the external geometry can be defined, the part can be built.

This chapter will describe the fundamentals for creating output files for AM. It will discuss the most common technique, which is to create the STL file format, explaining how it works and typical problems associated with it. There are numerous software tools for use with AM and there will follow a description of some of these and finally there will be a discussion on how new concepts in AM may affect the development of associated software tools in the future.

13.2 Preparation of CAD Models – the STL File

The STL file is derived from the word STereoLithography, which was the first commercial AM process, produced by the US company 3D Systems in the early 1980s [1], although some have suggested that STL should stand for Stereolithography

I. Gibson, D.W. Rosen, and B. Stucker, *Additive Manufacturing Technologies*,
DOI 10.1007/978-1-4419-1120-9_13,

Tessellation Language. STL files are generated from 3D CAD data within the CAD system. The output is a boundary representation that is approximated by a mesh of triangles.

13.2.1 STL File Format, Binary/ASCII

STL files can be output as either binary or ASCII (text) format. The ASCII format is less common but easier to understand and is generally used for illustration and teaching. Most AM systems run on PCs using Windows. The STL file is normally labeled with a ".STL" extension that is case insensitive, although some AM systems may require a different or more specific file definition. These files only show approximations of the surface or solid entities and so any information concerning the color, material, build layers, or history is ignored during the conversion process. Furthermore, any points, lines, or curves used during the construction of the surface or solid, and not explicitly used in that solid or surface, will also be ignored.

An STL file consists of lists of triangular facets. Each triangular facet is uniquely identified by a unit normal vector and three vertices or corners. The unit normal vector is a line that is perpendicular to the triangle and has a length equal to 1.0. This unit length could be in mm or inches and is stored using 3 numbers. The STL file itself holds no dimensions, so the AM machine operator must know whether the dimensions are mm, inches, or some other unit. Since each vertex also has 3 numbers, there are a total of 12 numbers to describe each triangle. The following file shows a simple ASCII STL file that describes a right angled, triangular pyramid structure, as shown in Fig. 13.1.

Note that the file begins with an object name delimited as a solid. Triangles can be in any order, each delimited as a facet. The facet line also includes the normal for that triangle. Note that this normal is calculated from any convenient location on the triangle and may be from one of the vertices or from the center of the triangle. It is defined that the normal is perpendicular to the triangle and is of unit length. In most systems, the normal is used to define the outside of the surface of the solid, essentially pointing to the outside. The group of three vertices defining the triangle is delimited by the terms "outer loop" and "endloop." The outside of the triangle is best defined using a right-hand rule approach. As we look at a triangle from the outside, the vertices should be listed in a counter-clockwise order. Using the right hand with the thumb pointing upwards, the other fingers curling in the direction of the order of the vertices, the starting vertex being arbitrary. This approach is becoming more popular since it avoids having to do any calculations with an additional number (i.e., the facet normal) and therefore STL files may not even require the normal to prevent ambiguity.

A binary STL file can be described in the following way:

- An 80 byte ASCII header that can be used to describe the part
- A 4 byte unsigned long integer that indicates the number of facets in the object

```
solid triangular_pyramid
          facet normal 0.0 −1.0 0.0
            outer loop
              vertex 0.0 0.0 0.0
              vertex 1.0 0.0 0.0
              vertex 0.0 0.0 1.0
            endloop
          endfacet
          facet normal 0.0 0.0 −1.0
            outer loop
              vertex 0.0 0.0 0.0
              vertex 0.0 1.0 0.0
              vertex 1.0 0.0 0.0
            endloop
          endfacet
          facet normal 0.0 0.0 −1.0
            outer loop
              vertex 0.0 0.0 0.0
              vertex 0.0 0.0 1.0
              vertex 0.0 1.0 0.0
            endloop
          endfacet
          facet normal 0.577 0.577 0.577
            outer loop
              vertex 1.0 0.0 0.0
              vertex 0.0 1.0 0.0
              vertex 0.0 0.0 1.0
            endloop
          endfacet
        endsolid
```

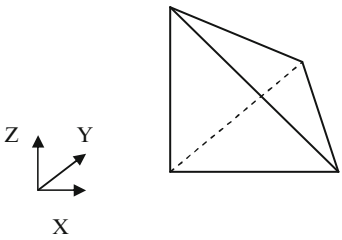


Fig. 13.1 A *right-angled triangular pyramid* as described by the sample STL file. Note that the *bottom left-hand* corner coincides with the origin and that every vertex coming out of the origin is of unit length

– A list of facet records, each 50 bytes long

The facet record will be presented in the following way:

– 3 floating values of 4 bytes each to describe the normal vector
– 3 floating values of 4 bytes each to describe the first vertex
– 3 floating values of 4 bytes each to describe the second vertex
– 3 floating values of 4 bytes each to describe the third vertex
– One unsigned integer of 2 bytes, that should be zero, used for checking

13.2.2 Creating STL Files from a CAD System

Nearly all geometric solid modeling CAD systems can generate STL files from a valid, fully enclosed solid model. Most CAD systems can quickly tell the user if a model is not a solid. This test is particularly necessary for systems that use surface modeling techniques, where it can be possible to create an object that is not fully

closed off. Such systems would be used for graphics applications where there is a need for powerful manipulation of surface detail (like with Autodesk AliasStudio software [2]) rather than for engineering detailing. Solid modeling systems, like Solidworks, may use surface modeling as part of the construction process, but the final result is always a solid that would not require such a test.

Most CAD systems use a "Save as" function to convert the native format into an STL file. There is typically some control over the size of the triangles to be used in the model. Since STL uses planar surfaces to approximate curved surfaces, then obviously the larger the triangles, the looser that approximation becomes. Most CAD systems do not directly limit the size of the triangles since it is also obvious that the smaller the triangle, the larger the resulting file for a given object. An effective approach would be to minimize the offset between the triangle and the surface that it is supposed to represent. A perfect cube with perfectly sharp edges and points can be represented by 12 triangles, all with an offset of 0 between the STL file and the original CAD model. However, few designs would be that convenient and it is important to ensure a good balance between surface approximation and excessively large file. Figure 13.2 shows the effect of changing the triangle offset parameter on an STL file. The exact value of the required offset would largely depend on the accuracy of the AM process to be used. If the offset is smaller than the basic resolution of the process, then making it smaller will have no effect on the precision of the resulting model. Since many AM processes operate around the 0.1 mm layer resolution, then a triangle offset of 0.05 mm or slightly lower will be acceptable for most AM technologies.

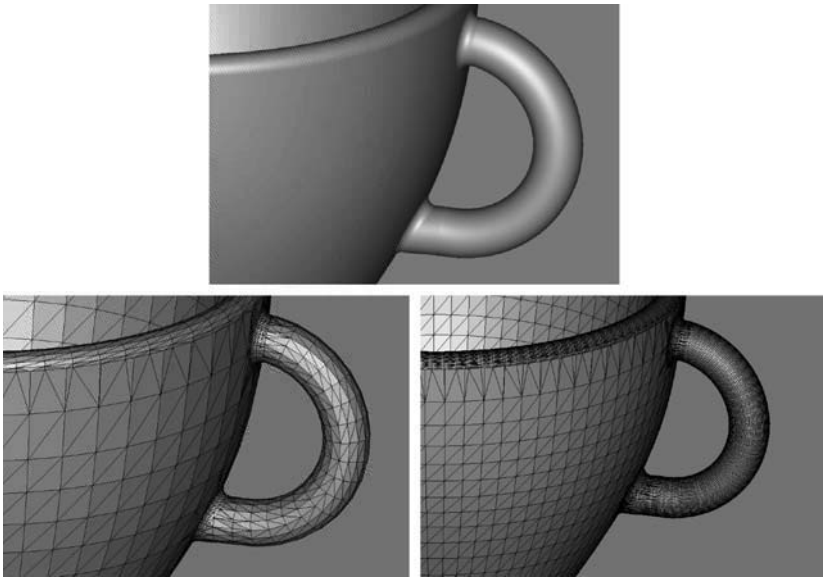
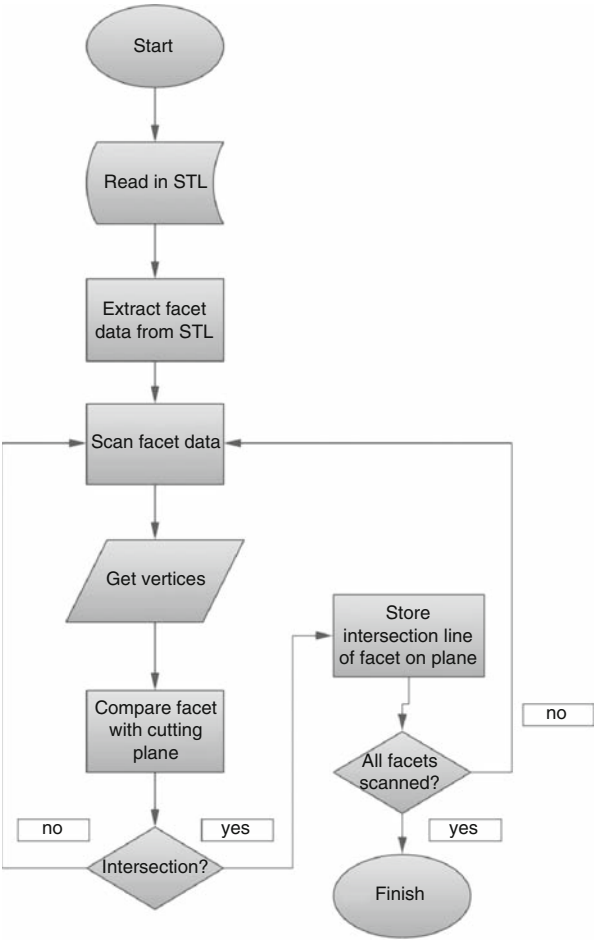


Fig. 13.2 An original CAD model converted into an STL file using different offset height (cusp) values, showing how the model accuracy will change according to the triangle offset

13.2.3 Calculation of Each Slice Profile

Virtually every AM system will be able to read both binary and ASCII STL files. Since most AM works by adding layers of material of a prescribed thickness, starting at the bottom of the part and working upwards, the part file description must therefore be processed to extract the profile of each layer. Each layer can be considered a plane in a nominal XY Cartesian frame. Incremental movement for each layer can then be along the orthogonal *Z* axis.

The *XY* plane, positioned along the *Z* axis, can be considered as a cutting plane. Any triangle intersecting this plane can be considered to contribute to the slice profile. An algorithm like the one in Flowchart 13.1 can be used to extract all the profile segments for a given STL file.



Flowchart 13.1 Algorithm for testing triangles and generating line intersections. The result will be an unordered matrix of intersecting lines (adapted from [3])

The resultant of this algorithm is a set of intersecting lines that are ordered according to the set of intersecting planes. A program that is written according to this algorithm would have a number of additional components, including a way of defining the start and end of each file and each plane. Furthermore, there would be no order to each line segment, which would be defined in terms of the *XY* components and indexed by the plane that corresponds to each *Z* value. Also, the assumption is that the STL file has an arbitrary set of triangles that are randomly distributed. It may be possible to preprocess each file so that searches can be carried out in a more efficient manner. One way to optimize the search would be to order the triangles according to the minimum *Z* value. A simple check for intersection of a triangle with a plane would be to check the *Z* value for each vertex. If the *Z* value of any vertex in the triangle is less than or equal to the *Z* value of the plane, then that triangle may intersect for one or more vertices. Using the above test, once it has been established that a triangle does not intersect with the cutting plane, then every other triangle is known to be above that triangle and therefore does not require checking. A similar check could be done with the maximum *Z* value of a triangle.

There are a number of discrete scenarios describing the intersection of each triangle with the cutting plane:

1. All the vertices of a triangle lie above or below the intersecting plane. This triangle will not contribute to the profile on this plane.
2. A single vertex directly lies on the plane. In this case, there is one intersecting point, which can be ignored but the same vertex will be included in other triangles satisfying another condition below.
3. Two vertices lie on the plane. Here one of the edges of the corresponding triangle lies on that plane and that edge contributes fully to the profile.
4. Three vertices lie on the plane. In this case, the whole triangle contributes wholly to the profile, unless there are one or more triangles also lying on the plane, in which case the included edges can be ignored.
5. One vertex lies above or below the intersecting plane and the other two vertices lie on the opposite side of the plane. In this case, an intersecting vector must be calculated from the edges of the triangle.

Most triangles will conform to scenario 1 or 5. Scenarios 2–4 may be considered special cases and require special treatment. Assuming that we have performed appropriate checks and that a triangle corresponds to scenario 5, then we must take action and generate a corresponding intersecting profile vector. In this case, there will be two vectors defined by the triangle vertices and these vectors will intersect with the cutting plane. The line connecting these two intersection points will form part of the outline for that plane.

The problem to be solved is a classical line intersection with a plane problem. In this case, the line is defined using Cartesian coordinates in (x, y, z). The plane is defined in (x, y) for a specific constant height, z. In a general case, we can therefore project the line and plane onto the $x = 0$ and $y = 0$ planes. For the $y = 0$ plane, we can obtain something similar to Fig. 13.3. Points P_1 and P_2 correspond to two points

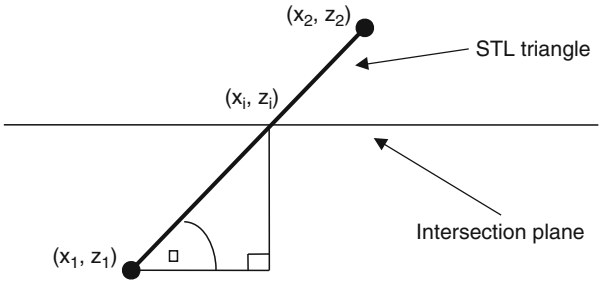


Fig. 13.3 A vertex taken from an STL triangle projected onto the $y = 0$ plane. Since the height z_i is known, we can derive the intersection point x_i. A similar case can be done for y_i in the $x = 0$ plane

of the intersecting triangle. P_p is the projected point onto the $y = 0$ plane to form a unique right-angled triangle. The angle θ can be calculated from

$$\tan\theta = \frac{(z_2 - z_1)}{(x_2 - x_1)} \tag{13.1}$$

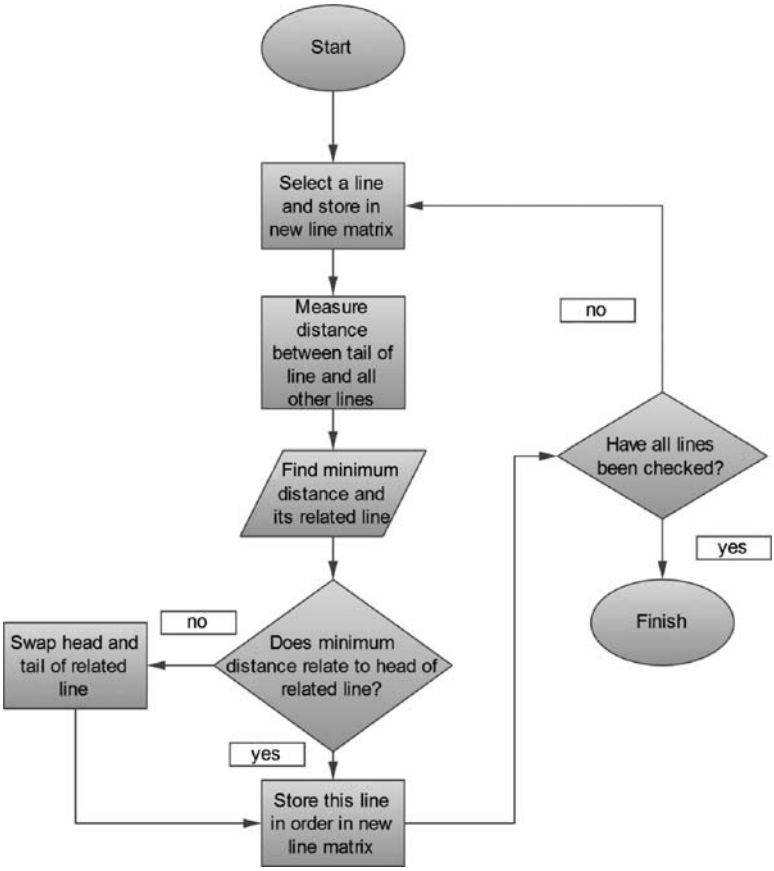
since we know the z height of the plane, we can use the following equation:

$$\tan\theta = \frac{(z_i - z_1)}{(x_i - x_1)} \tag{13.2}$$

and solve for x_i

A point y_i can also be found after projecting the same line on to the $x = 0$ plane to fully define the intersecting point P_i. A second intersecting point can be determined using another line of the triangle that intersects the plane. These two points will make up a line on the plane that forms part of the outline of the model. It is possible to determine directionality of this line by correct use of the right-hand rule, thus turning this line segment into a vector. This may be useful for determining whether a completed curve forms part of an enclosing outline or corresponds to a hole.

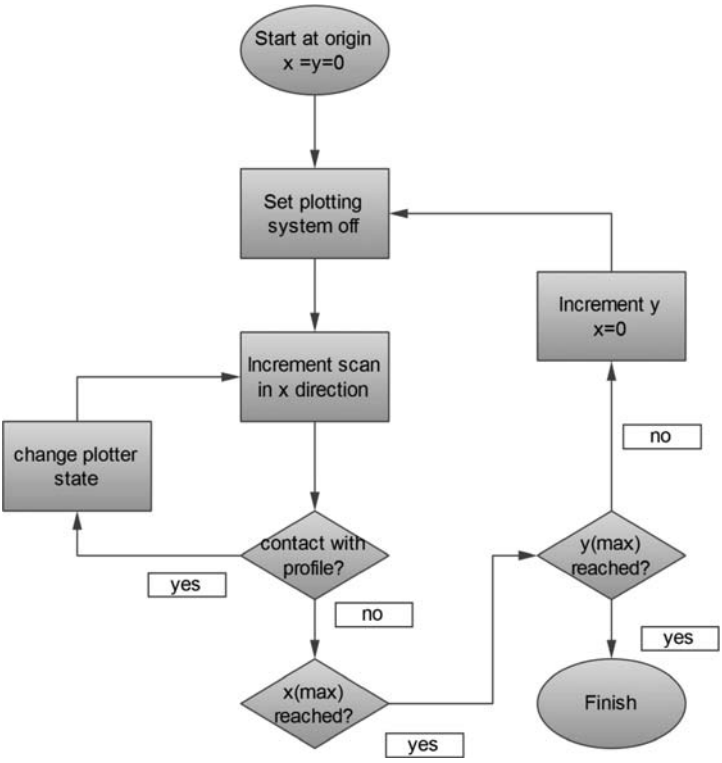
Once all intersecting lines have been determined according to Flowchart 13.1, then these lines must be joined together to form complete curves. This would be done using an algorithm based on that described in Flowchart 13.2. In this case, each line segment is tested to determine which segment is closest. A "closest point" algorithm is necessary since calculations may not exactly locate points together, even though the same line would normally be used to determine the start location of one segment and the end of another. Note that this algorithm should really have further nesting to test whether a curve has been completed. If a curve is complete, then any remaining line segments would correspond to additional curves. These additional curves could form a nest of curves lying inside or outside others, or they could be separate. The two algorithms mentioned here focus on the intersection of



Flowchart 13.2 Algorithm for ordering the line intersections into complete outlines. This assumes there is only a single contour in each plane (adapted from [3])

triangular facets with the cutting plane. A further development of these algorithms could be to use the normal vectors of each triangle. In this way, it would be possible to establish the external direction of a curve. This would be helpful in determining nested curves. The outer-most curve will be pointing outside the part. If a curve set is pointing inwards on itself then it is clear there must be a further curve enveloping this one (see Fig. 13.3). Use of the normals may also be helpful in organizing curve sets that are in very close proximity to each other.

Once this stage has been completed, there will be a file containing an ordered set of vectors that will trace complete outlines corresponding to the intersecting plane. How these outlines are used depend somewhat on which AM technology is to be used. Many machines can use the vectors generated in Flowchart 13.2 to control a plotting process to draw the outlines of each layer. However, most machines would also need to fill in these outlines to make a solid. Flowchart 13.3 uses an inside/outside algorithm to determine when to switch on a filling mechanism to draw



Flowchart 13.3 Algorithm for filling in a 2D profile based on vectors generated using Flowchart 13.2 and a raster scanning approach. Assume the profile fits inside the build volume, the raster scans in the *X* direction and lines increment in the *Y* direction

scanning lines perpendicular to one of the planar axes. The assumption is that the part is fully enclosed inside the build envelope and therefore the default fill is switched off.

13.2.4 Technology Specific Elements

Flowcharts 13.1–13.3 are basic algorithms that are generic in nature. These algorithms need to be refined to prevent errors and to tailor them to suit a particular process. Other refinements may be employed to speed the slicing process up by eliminating redundancy, for example.

As mentioned in previous chapters, many AM systems require parts built using support structures. Supports are normally a loose-woven lattice pattern of material placed below the region to be supported. Such a lattice pattern could be a simple

Fig. 13.4 Supports generated for a part build

square pattern or something more complex like a hexagonal or even a fractal mesh. Furthermore, the lattice could be connected to the part with a tapered region that may be more convenient to remove when compared with thicker connecting edges.

Determination of the regions to be supported can be made by analyzing the angle of the triangle normals. Those normals that are pointing downwards at some previously defined minimum angle would require supports. Those triangles that are sloping above that angle would not require supports. Supports are extended until they intersect either with the base platform or another upward facing surface of the part. Supports connecting with the upward facing surface may also have a taper that enables easy removal. The technique that would normally be used would be to extend supports from the entire build platform and eliminate any supports that do not intersect with the part at the minimum angle or less (see Fig. 13.4).

The support structures would be generated directly as STL models and can be incorporated into the slicing algorithms already mentioned. Some other processing requirements that would be dependent for different AM technologies include:

- *Raster scanning*: While many technologies would use a simple raster scan for each layer, there are alternatives. Some systems use a switchable raster scan, scanning in the *X* direction of an *XY* plane for one layer and then moving to the *Y* direction for alternate layers. As discussed in Chap. 5, some systems subdivide the fill area into smaller square regions and use switchable raster scans between squares.
- *Patterned vector scanning*: FDM technology requires a fill pattern to be generated within an enclosed boundary. This is done using vectors generated using a patterning strategy. For a particular layer, a pattern would be determined by choosing a specific angle for the vectors to travel. The fill is then a zigzag pattern along the direction defined by this angle. Once a zigzag has reached an end there may be a need for further zigzag fills to complete a layer (see Fig. 13.5 for an example of zigzag scan pattern).

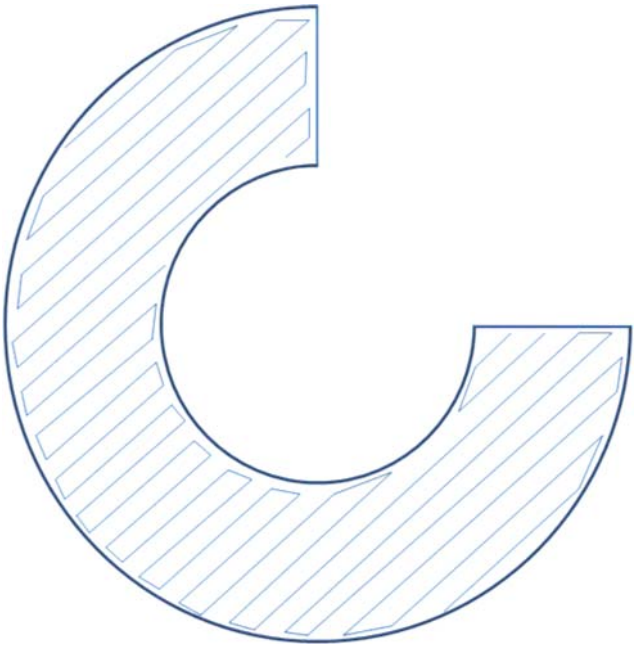


Fig. 13.5 A scan pattern using vector scanning in FDM. Note the outline drawn first followed by a small number of zigzag patterns to fill in the space

– *Hatching patterns*: Laminated object modeling processes like the Helisys LOM and the Solid Centre machine from Kira [4] require material surrounding the part to be hatched with a pattern that allows it to be de-cubed once the part has been completed (see Fig. 13.6).

13.3 Problems with STL Files

Although the STL format is quite simple, there can still be errors in files resulting from CAD conversion. The following are typical problems that can occur in bad STL files:

Unit changing: This is not strictly a result of a bad STL file. Since US machines still commonly use imperial measurements and most of the rest of the world uses metric, some files can appear scaled because there is no explicit mention of the units used in the STL format. If the person building the model is unaware of the purpose of the part then he may build it approximately 25 times too large or too small in one direction. Furthermore, units must correspond to the location of the origin within the machine to be used. This normally means that the physical origin of the machine lies in the bottom left-hand corner and so all triangle coordinates within an STL file must be positive. However, this may not be the

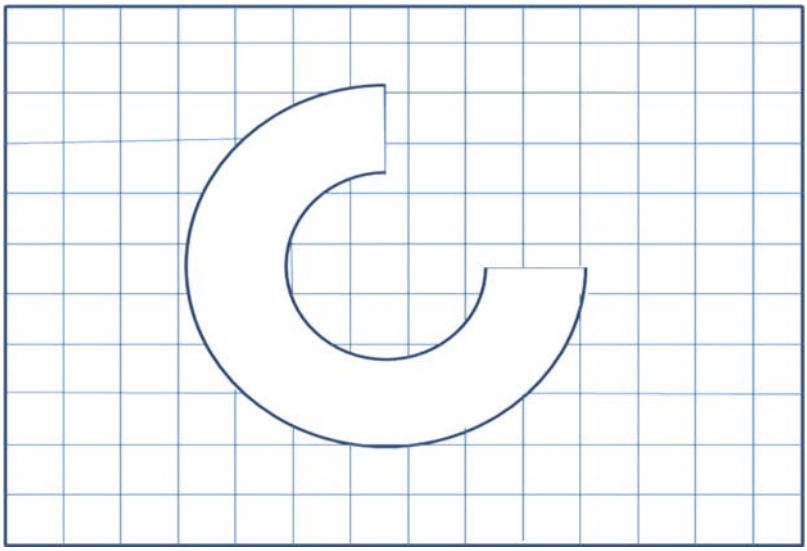


Fig. 13.6 Hatching pattern for LOM-based processes. Note the outside hatch pattern that will result in cubes which will be separated from the solid part during post-processing

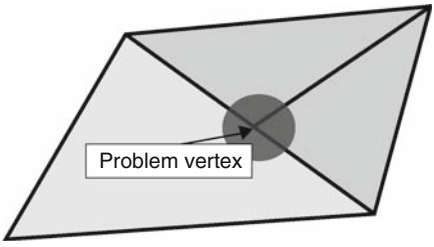


Fig. 13.7 A case that violates the vertex to vertex rule

case for a particular part made in the CAD system and so some adjustment offset of the STL file may be required.

Vertex to vertex rule: Each triangle must share two of its vertices with the triangles adjacent to it. This means that a vertex cannot intersect the side of another, like that shown in Fig. 13.7 (reproduced from [5]). This is not something that is explicitly stated in the STL file description and therefore STL file generation may not adhere to this rule. However, a number of checks can be made on the file to determine whether this rule has been violated. For example, the number of faces of a proper solid defined using STL must be an even number. Furthermore, the number of edges must be divisible by 3 and follow the equation:

$$\frac{\text{No.of faces}}{\text{No. of edges}} = \frac{3}{2} \tag{13.3}$$

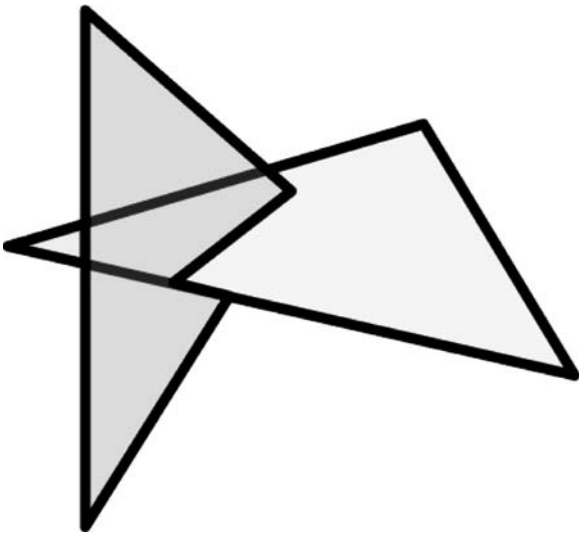


Fig. 13.8 Two triangles intersecting each other in 3D space

Leaking STL files: As mentioned earlier, STL files should describe fully enclosed surfaces that represent the solids generated within the originating CAD system. In other words, STL data files should construct one or more nonmanifold entities according to Euler's Rule for solids:

$$\text{No. of faces} - \text{No. of edges} + \text{No. of vertices} = 2 \times \text{No. of bodies} \qquad (13.4)$$

If this rule does not hold then the STL file is said to be leaking and the file slices will not represent the actual model. There may be too few or too many vectors for a particular slice. Slicing software may add in extra vectors to close the outline or it may just ignore the extra vectors. Small defects can possibly be ignored in this way. Large leaks may result in unacceptable final models.

Leaks can be generated by facets crossing each other in 3D space as shown in Fig. 13.8. This can result from poorly generated CAD models, particularly those that use Boolean operations when generating solids.

A CAD model may also be generated using a method which stitches together surface patches. If the triangulated edges of two surface patches do not match up with each other then holes, like in Fig. 13.9, may occur.

Degenerated facets: These facets normally result from numerical truncation. A triangle may be so small that all three points virtually coincide with each other. After truncation, these points lay on top of each other causing a triangle with no area. This can also occur when a truncated triangle returns no height and all three vertices of the triangle lie on a single straight line. While the resulting slicing algorithm will not cause incorrect slices, there may be some difficulties with any checking algorithms and so such triangles should really be removed from the STL file.

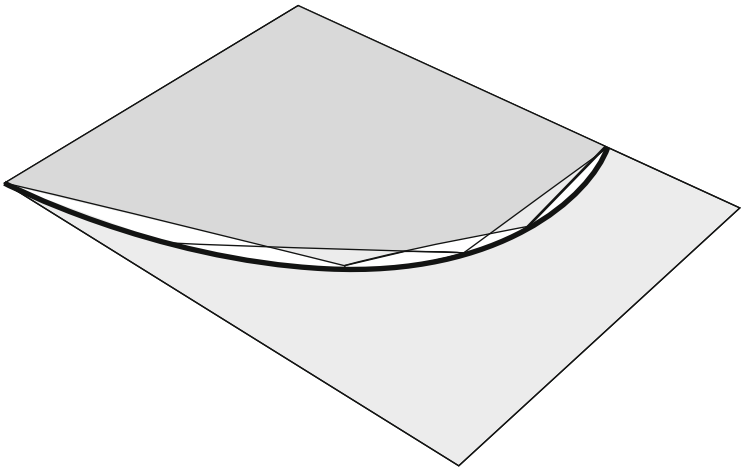


Fig. 13.9 Two surfaces patches that do not match up with each other, resulting in holes

It is worth mentioning that, while a few errors may creep into some STL files, most 3D CAD systems today produce high quality and error-free results. In the past, problems more commonly occurred from surface modeling systems, which are now becoming scarcer, even in fields outside of engineering CAD like computer graphics and 3D gaming software. Also, in earlier systems, STL generation was not properly checked and faults were not detected within the CAD system. Nowadays, potential problems are better understood and there are well known algorithms for detecting and correcting such problems.

13.4 STL File Manipulation

Once a part has been converted into STL there are only a few operations that can be performed. This is because the triangle-based definition does not permit radical changes to the data. Associations between individual triangles are through the shared points and vertices only. A point or vertex can be moved, which will affect the connected triangles, but creating a regional affect on larger groups of points would be more difficult. Consider the modeling of a simple geometry, like the cut cylinder in Fig. 13.10a. Making a minor change in one of the measurements may result in a very radical change in distribution of the triangles. While it is possible to simplify the model by reducing the number of triangles, it is quite easy to see that defining boundaries in most models cannot be easily done. The addition of a fillet in Fig. 13.10b shows an even more radical change in the STL file. Furthermore, if one were to attempt to move the oval that represents the cut surface, the triangles representing the filet would no longer show a constant-radius curve.

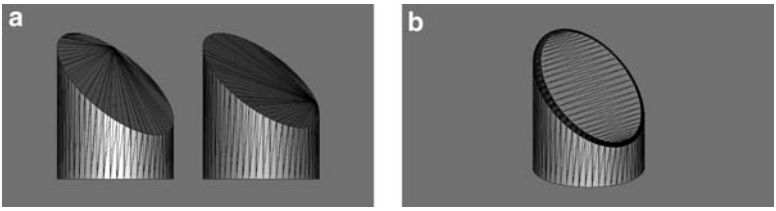


Fig. 13.10 STL files of a cut cylinder. Note that although the two models in (**a**) are very similar, the location of the triangles is very different. Addition of a simple filet in (**b**) shows even greater change in the STL file

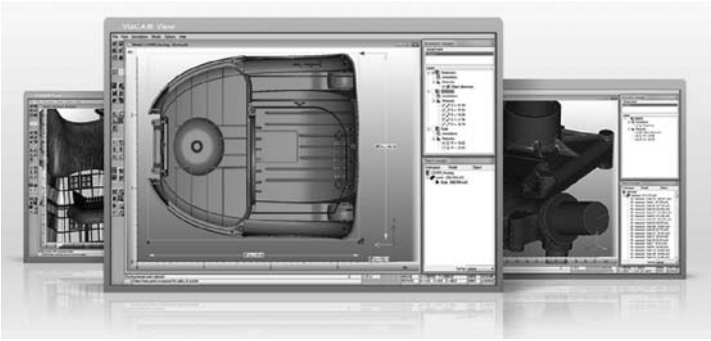


Fig. 13.11 The VisCAM viewer from Marcam that can be used to inspect STL models

Building models using AM is often done by people working in departments or companies that are separate from the original designers. It may be that whoever is building the model may not have direct access to the original CAD data. There may therefore be a need to modify the STL data before the part is to be built. The following sections discuss STL tools that are commonly used.

13.4.1 Viewers

There are a number of STL viewers available, often as a free download. An example is STLview from Marcam [6] (see Fig. 13.11). Like many other systems, this software allows limited access to the STL file, making it possible to view the triangles, apply shading, show sections, etc. By purchasing the full software version, other tools are possible, for example allowing the user to measure the part at various locations, annotate the part, display slice information, and detect potential problems with the data. Often the free tools allow passive viewing of the STL data, while the more advanced tools permit modification of the data, either by rewriting the STL or supplying additional information with the STL data (like measurement information, for example). Often these viewers are connected to part

building services and provided by the company as an incentive to use these services and to help reduce errors in data transfer, either from incorrect STL conversion or from wrong interpretation of the design intent.

13.4.2 STL Manipulation on the AM Machine

STL data for a part consist of a set of points defined in space, based on an arbitrarily selected point of origin. This origin point may not be appropriate to the machine the part is to be built on. Furthermore, even if the part is correctly defined within the machine space, the user may wish to move the part to some other location or to make a duplicate to be built beside the original part. Other tasks, like scaling, changing orientation, and merging with other STL files are all things that are routinely done using the STL manipulation tools on the AM machine.

Creation of the support structures is also something that would normally be expected to be done on the AM machine. This would normally be done automatically and would be an operation applied to downward-facing triangles. Supports would be extended to the base of the AM machine or to any upward facing triangle placed directly below. Triangles that are only just veering away from the vertical (e.g., less than 10°) may be ignored for some AM technologies. Note, for example, the supports generated around the cup handle in Fig. 13.4.

With some AM operating systems there is little or no control over placement of supports or manipulation of the model STL data. Considering Fig. 13.4 again, it may be possible to build the handle feature without so many supports, or even with no supports at all. A small amount of sagging around the handle may be evident, but the user may prefer this to having to clean up the model to remove the support material. If this kind of control is required by the user, it may be necessary to purchase additional third party software, like the MAGICS and 3-matic systems from Materialise [7].

Such third party software may also be used to undertake additional roles. MAGICS, for example, has a number of modules useful to many AM technologies. Other STL file manipulators may have similar modules:

- Checking the integrity of STL files based on the problems described above.
- Incorporating support structures including tapered features on the supports that may make them easier to remove.
- Optimizing the use of AM machines, like ensuring the machine is efficiently filled with parts, the amount of support structures is minimized, etc.
- Adding in features like serial numbers and identifying marks onto the parts to ensure correct identification, easy assembly, etc.
- Remeshing STL files that may have been created using Reverse Engineering software or other non-CAD based systems. Such files may be excessively large and can often be reduced in size without compromising the part accuracy.
- Segmenting large models or combining multiple STL files into a single model data set.

– Performing Boolean tasks like subtracting model data from a tool insert blank model to create a mold.

13.5 Beyond the STL File

The STL definition was created by 3D Systems right at the start of the development history of AM technology and has served the industry well. However, there are other ways in which files can be defined for creation of the slice. Furthermore, the fact that the STL file only represents the surface geometry may cause problems for parts that require some heterogeneous content. This section will discuss some of the issues surrounding this area.

13.5.1 Direct Slicing of the CAD Model

Since generation of STL files can be tedious and error-prone, there may be some benefit from using in-built CAD tools to directly generate slice data for the AM machines. It is a trivial task for most 3D solid modeling CAD systems to calculate the intersection of a plane with a model, thus extracting a slice. This slice data would ordinarily need to be processed to suit the drive system of the AM technology, but this can be handled in most CAD systems with the use of macros. Support structures can be generated using standard geometry specifications and projected onto the part from a virtual representation of the AM machine build platform.

Although this approach has never been a popular method for creating slice data, it has been investigated as a research topic [8] and even developed to suit a commercialized variant of the Stereolithography process by a German company called Fockle & Schwarz. The major barrier to using this approach is that every CAD system must include a suite of different algorithms for direct slicing for a variety of machines or technologies. This would be a cumbersome approach that may require periodic updates of the technology as new machines become available. There may be some benefit in the future in creating an integrated design and manufacturing solution, especially for niche applications or for low-cost solutions. However, at present it is more sensible to separate the development of the design tools from that of the AM technology by using the STL format.

13.5.2 Color Models

Currently, there is one AM technology available on the market that can produce full color output, namely the color 3D Printing technology from ZCorp. This has proven to be a very popular technology, and it is likely that other color AM machines will

make their way to the market. The conventional STL file contains no information pertaining to the color of the part or any features thereon. Coloring of STL files is possible and there are in fact color STL file definitions available [9], but you would be limited by the fact that a single triangle can only be one specific color. It is therefore much better to use the VRML painting options that allow you to assign bitmap images to individual facets [9]. In such a way, it is possible to take advantage of the full color possibilities that the ZCorp machine can give you.

13.5.3 Multiple Materials

Carrying on from the previous section, color is one of the simplest examples of multiple material products that AM is capable of producing. As has been mentioned in other chapters, parts can be made using AM from composite materials, with varying levels of porosity or indeed with regions containing discretely different materials. For many of these new AM technologies, STL is starting to become an impediment. Since the STL definition is for surface data only, the assumption therefore is that the solid material between these surfaces is homogeneous. As we can see from the above this may not be the case. While there has been significant thought applied to the problem of representations for heterogeneous solid modeling [10], there is still much to be considered before we can arrive at a standard to supersede STL for future AM technology, as discussed in Chap. 11.

13.5.4 Use of STL for Machining

STL is used for applications beyond just converting CAD to Additive Manufacturing input. Reverse Engineering packages can also be used to convert point cloud data directly into STL files without the need for CAD. Such technology connected directly to AM could conceivably form the basis for a 3D Fax machine. Another technology that can easily make use of STL files is subtractive manufacturing.

Subtractive manufacturing systems can readily make use of the surface data represented in STL to determine the boundaries for machining. With some additional knowledge concerning the dimensions of the starting block of material, tool, machining center, etc., it is possible to calculate machining strategies for creating a 3D surface model. As mentioned in earlier chapters, the likelihood is that it may not be possible to fully machine complex geometries due to undercutting features, internal features, etc. but there is no reason why STL files cannot be used to create Computer Aided Manufacturing (CAM) profiles for machining centers. Delft Spline [11] has been using STL files to create CAM profiles for a number of years now. Figure 13.12 shows the progression of a model through to a tool to manufacture a final product using their DeskProto software. Another technology that uses a hybrid of subtractive and additive processes to fabricate parts is the SRP

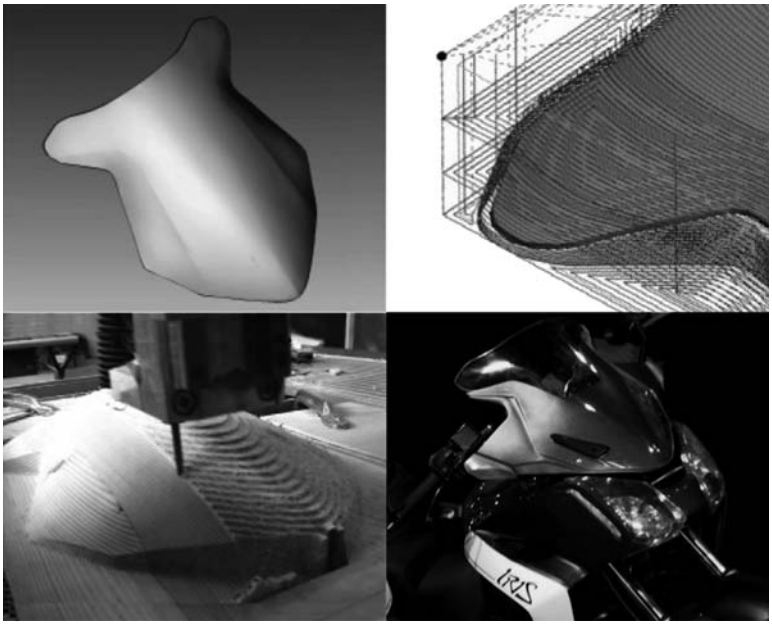


Fig. 13.12 DeskProto software being used to derive machine tool paths from STL file data to form a mold for creating the windscreen of a motorcycle

(Subtractive Rapid Prototyping) technique developed by Roland [12] for their desktop milling machines. The creation of finish machining tool paths for AM parts from STL files is also discussed in Chap. 16.

13.6 Additional Software to Assist AM

As well as directly controlling the manufacturing process, other software systems may be helpful in running an effective and efficient AM-based facility. Such software can include one or more of the following functions:

Simulation: Many operating systems can perform a simulation of the build process, showing how the layers will be formed step by step in accelerated time. This can allow the user to detect obvious errors in the slice files and determine whether critical features can be built. This may be particularly important for processes like FDM where the hatch patterns can have a critical effect on thin wall features, for example. Some work has been carried out to simulate AM systems to get a better impression of the final result, including rendered images to give an understanding of the surface roughness for a given layer thickness, for example [13].

Build-time estimation: AM is a highly automated process and the latest machines are very reliable and can operate unattended for long periods of time. For effective process planning it is very important to know when a build is going to be completed. Knowing this will help in determining when operators will be required to change over jobs. Good estimates will also help to balance builds; adding or subtracting a part from the job batch may ensure that the machine cycle will complete within a day-shift, for example, making it possible to keep machines running unattended at night. Also, if you are running multiple machines, it would be helpful to stagger the builds throughout the shift to optimize the manual work required. Early build-time estimation software was extremely unreliable, performing rolling calculations of the average build time per layer. Since the layer time is dependent on the part geometry, such estimates could be very imprecise and vary wildly, especially at the beginning of a build. Later software versions saw the benefit in having more precise build-time estimations. A simplified build-time model is discussed in Chap. 14.
Machine setup: While every AM machine has an operating system that makes it possible to set up a build, such systems can be very basic, particularly in terms of manipulation of the STL files. Determining build parameters based on a specific material is normally very comprehensive however.
Monitoring: This is a relatively new feature for most AM systems. Even though nearly every AM machine will be connected either directly or indirectly to the Internet, this has traditionally been for uploading of model files for building. Export of information from the machine to the Internet or within an Intranet has not been common except in the larger, more expensive machines. The simplest monitoring systems would provide basic information concerning the status of the build and how much longer before it is complete. However, more complex systems may tell you about how much material is remaining, the current status parameters like temperatures, laser powers, etc., and whether there is any need for manual intervention through an alerting system. Some monitoring systems may also provide video feedback of the build.
Planning: Having a simulation of the AM process running on a separate computer may be helpful to those working in process planning. Process planners may be able to determine what a build could look like, thus allowing the possibility of planning for new jobs, variability analysis, or quoting.

Once again such software may be available from the AM system vendor or from a third party vendor. The advantage of third party software is that it is more likely to be modified to suit the exact requirements of the user.

13.6.1 Exercises

1. How would you adjust Flowchart 13.2 to include multiple contours?
2. Under what circumstances might you want to merge more than one STL file together?

3. Write out an ASCII STL file for a perfect cube, aligned with the Cartesian coordinate frame, starting at (0,0,0) and all dimensions positive. Model the same cube in a CAD system. Does it make the same STL file? What happens when you make slight changes to the CAD design?
4. Why might it be possible that a part could inadvertently be built 25 times too small or too large in any one direction?
5. Is it okay to ignore the vertex of a triangle that lies directly on an intersecting cutting plane?
6. Prove to yourself with some simple examples that the number of faces divided by the number of edges is 2/3.

References

1. 3D Systems Inc. (1989) Stereolithography Interface Specification, October 1989
2. Autodesk CAD software, usa.autodesk.com
3. Vatani M, Rahimi AR et al (2009) An enhanced slicing algorithm using nearest distance analysis for layer manufacturing. Proceedings of World Academy of Science, Engineering and Technology, vol 37, pp 721–726, Jan 2009. ISSN 2070-3740
4. Kira, Solid Center machine. http://www.kiracorp.co.jp/EG/pro/rp/top.html
5. Wai HW, discussion on STL file manipulation rpdrc.ic.polyu.edu.hk/old_files/stl_introduction.htm
6. Marcam, VisCAM software. http://www.marcam.de
7. Materialise, AM software systems. http://www.materialise.com/materialise/view/en/92074-Magics.html
8. Jamieson R, Hacker H (1995) Direct slicing of CAD models for rapid prototyping. Rapid Prototyping J 1(2):4–12. Pub. by Emerald, ISSN 1355-2546
9. Ling WM, Gibson I (2002) Specification of VRML in color rapid prototyping. Int J of CAD/CAM 1(1):1–9
10. Siu YK, Tan ST (2002) Representation and CAD modeling of heterogeneous objects. Rapid Prototyping J 8(2):70–75. ISSN 1355-2546
11. Delft Spline. http://www.spline.nl/dp/deskproto.html
12. Roland, desktop milling and subtractive RP. http://www.rolanddga.com/ASD/
13. Choi SH, Samavedam S (2001) Visualisation of rapid prototyping. Rapid Prototyping J 7(2):99–114