Support Generation for Fused Deposition Modeling

Kumar Chalasani, Stratasys Inc., Minneapolis, USA.
Larry Jones, SDRC, Cincinnati, USA.
Larry Roscoe, Stratasys Inc., Minneapolis, USA.

Support Generation for Fused Deposition Modeling

1. Introduction

With the growth of the use of Rapid Prototyping (RP) systems, there has been a corresponding growth in the complexity of parts expected from the various RP processes. To meet the demand for increasingly more intricate and detailed prototypes, the technology has matured and improved, allowing RP users to build these parts faster, better and with a variety of materials.

Stratasys Inc.'s Fused Deposition Modeling (FDM ®) system is one of the leading prototyping systems available in the market today. Recent enhancements in electro-mechanical control of the machine head have improved the viability of FDM as a system for manufacturing complex prototypes. The FDM-1600 system can now also create detailed parts using ABS, a widely used plastic in the automotive and other industries. In addition, we have further reduced the lead time (preprocessing or operator time) in the product design cycle with the inclusion of an automatic support generation module in Stratasys' proprietary QuickSlice® software product. QuickSlice processes STL data to generate machine code to drive the FDM system.

2. Problem Statement

Some of the RP processes use supporting structures (supports) to hold or anchor areas of the model as it is being built. FDM's support requirements are different from other RP technologies because of the nature of the process - additive from bottom-up, with material being extruded from above. With the StereoLithography Apparatus (SLA), for instance, the viscous resin is sufficient support in many cases, but it also may require other supports such as anchors to prevent drifting of portions of the part, gussets or webs to prevent sag on overhangs and long bridge-spans. The amount and type of supports also depend on the geometry and, to a smaller extent, on the material used. On the FDM-1600, using our new BASS ™ (Break-Away Support System), we are able to use an alternate material as a 'release layer' at the part-support interface. An inherently weaker bond is created between the release and primary modeling materials material, thus facilitating the removal of supports from the part without damaging part surface or features.

This paper describes three computational techniques used in the generation of support geometry. Primary requirements are automation, speed of support generation, ease-of-use, speed of build (optimal quantity of supports), and support removability.

3. History

Any support generation scheme must be based on part geometry. In the RP industry at this time, the STL format is widely used to represent part geometry. STL files are generated by many of the available and popular CAD software packages. An STL is a tessellated (triangulated) surface model, and the quality of an STL file depends on several factors, including:

1. operator expertise and care in creating a 3D CAD model;
2. the triangulation algorithm used; and
3. topology and geometry of the part.

Most STL generators produce certain flaws which become evident upon *slicing* (the process of generating layers - common to all the RP processes), and require operator time for correction. Stratasys's file format for these layers is the SLC file[1], which is essentially a list of points for all the curves on each level. For support generation on the FDM, thus, we have for input a choice between STL and SLC files. Using the STL as input, other methods have generated supports based on facet orientation and size. The motivation for generating supports from STLs is that STL is currently the most widely used format in the industry, and supports so generated could be used in any system that uses STLs. However, as noted above, support requirements vary from one system to another, as well as from one material to another. Another problem with STL based support generators is that the supports are also usually generated as STLs, which is more data for the slicing module to handle. These methods often generate discrete supports that are stilt-like, tall, thin columns that usually fall over as they are getting built. It is hard with this data to merge two or more support stilts to make them 'fatter' and more stable. There is also the problem of multiplication of data as well as error (from flawed STLs) from generating supports so early in the process. STL flaws are common and varied, including overlapping facets, gaps between facets and long, thin facets.

The preferred method in this case is to proceed from slice data, which has already been processed to eliminate errors coming from flawed STL files; and after the *road-widths* (a *road* is a track or bead of extruded material) and other parameters have been selected, including material to be used for the build.

4. Description of Support Generation Methods in QuickSlice

Intuitively, a part requires support wherever there are substantial overhanging areas, or if the part is unstable in the orientation of build. A sub-optimal but obvious generation scheme would be to engulf the part wholly inside a bounding box using the cubic volume around the part. Thus, every *voxel* (a pixel in three dimensions) in the volume that is not part is support. In what follows, we improve this generalized and blind support generator to create more optimal supports. The term 'curve' is used to mean 'polyline'. Three strategies are discussed:

1. Containment - enclosing the entire part in a shape-following container
2. Region - supporting a specified region
3. Direct - automatic generation of optimal supports.

---

[1]The Stratasys SLC format was made public in 1993 as a simple method of data exchange with scanners and other pre-sliced data. It is not to be confused with 3D Systems' SLC format.

5. Supports by Containment

With SLC as input, finding the minimal shape-following enclosure around the part is the easiest option, computationally. Conceding that a 'containment' approach will by definition generate more supports than strictly necessary (*all around*, rather than just *under*, which would be sufficient), we nevertheless provide this option in QuickSlice, allowing the user to use it whenever deemed necessary, considering time-to-build and quality expectations. One of the premises in this approach is the fact that any solid is eventually connected in a single mass, even if on any given level there may be several disjoint curves.

We take a first pass of all the curves in the part, looking for curves contributing to the extreme (min and max) x and y, and the curve with the largest area. We then attempt to merge these curves, leaving out those that not merged, to be considered later. We now take a second pass, querying each curve for containment in the merged result (*container*). Beginning with simple bounding box tests to eliminate disjoint curves, we follow with an edge-crossing test for determining whether or not those curves intersect with the container. This eliminates completely contained curves, thus avoiding having to find intersections, which is computationally expensive. We then merge the remaining curves that intersecting the container. In a third pass, we attempt to re-merge the extremes that we left out earlier. Notice that they now do merge, validating the above-stated premise (Figs 1a through 1d). This approach is fast because of collecting extreme curves at the outset (since the data structure we use already carries bounding box information). In parts with evenly distributed masses, the first pass generates a containment curve that includes almost all the other curves in the part. The container is offset further out, and all the part curves are offset (in and out, alternatively) so that the *nesting* (the hierarchical relationship between curves when some are completely contained in others, which determines which regions are part, air and support) of curves is not affected.

The above method immerses the part in a volume that is slightly larger than an exact shadow of the part; looking at the part from above. This method compromises build time for computational speed in support generation.

To reduce the quantity of supports (to save build time and reduce surface scar), the user may edit out portions that are not necessary supports. Even if build time and material cost were not an issue, the feature and surface damage resulting from part/support interfacial contact should be minimized by creating only strictly necessary supports.

6. Supporting a Region

We also provide, in QuickSlice, a semi-automatic method of generating supports. The user indicates regions needing supports, by drawing a region while looking at the part in the top view. This region is then automatically copied between levels as indicated by the user. When the drawn curve intersects a part curve, the region curve trims itself off (Figs 2a and 2b). This method is intuitive and simple for overhanging regions and better than immersing an entire feature inside a support. However, while this method is powerful to support flat overhangs and other

features obviously needing supports, in many parts there are far too many regions for this method to be usable.

Neither of the above methods consider optimization to avoid creating self-supporting inclines. To optimize build time, automatically mark interfacial curves, and to minimize user intervention in the support generation process, we implemented an automated support generation scheme that takes into account process parameters and part geometry, attempting to generate only necessary supports.

7. Automatic Support Generation

The QuickSlice automatic support generator has some built-in knowledge about the kinds of features that require supports on the FDM system.

7.1 Overall Part-Stability

To begin with, the user must find the *best-base*: determine an orientation of the part that will maximize stability and minimize fragility as it is being built. Often, a user may decide to compromise these two factors to avoid damage to large, visible surfaces.

Once the orientation is chosen; if the center of gravity (CG) of the part falls outside the part's *best-base* (such as an inverted 'L'), a first class of supports is needed merely to hold the part up (in other words, to fatten the base to allow the CG to fall within the base). Geometries whose CG falls inside the *best-base* may also require support for overall part stability if the geometry is top-heavy with only a small base-area (such as a 'V').

7.2 Self-Supporting Features.

In the FDM process, many geometries do not need supports: *Cantilevers* (when at most 50% of a *road* is overhanging) (Fig 3a) on gradual inclines; *links* (based on lateral bond between adjacent beads of road) (Fig 3b); and *bridging* (spanning over gaps in material underneath) (Fig 3c). Cantilever length, link length and bridge span are different for different materials. These techniques cannot be indefinitely extended, because of physical limitations: it is not possible to lay down a very long bead of *road* without also making it thick. Typical maximum width for a slice of 0.01 inches (.254 mm) thick is a road 0.06 inches (1.524 mm) wide .

7.3 Features Requiring Supports

Figs 4a through 4d show a partial set of features that are not self-supporting. Fig 4a and Fig 4b show two general types of features: sharp inclines and sudden flats. In general, we classify as sharp inclines any overhang that is at less than 45 degrees to the horizontal (in a front/side view). Sudden flats or near flats are large overhangs that appear suddenly at a level, with nothing underneath. Other frequently occurring features are hooks (Fig. 4c) , domes and arches with large curvatures (Fig. 4d).

7.4 The Algorithm

Consider two levels such as in Fig 5a and 5b., from an SLC file. The human eye can see easily enough whether or not the level above is sufficiently different from the level below to warrant supports. The human eye cannot, without measurement, discern whether or not the difference is small enough to avoid creating supports where it may be self-supporting. We call the difference between level *n* and *n-1* the *shadow* of level *n* on level *n-1* (Figs 5a -c). If the width of this *shadow* is smaller than the *road-width*, level *n* is self-supporting with respect to level *n-1*. This is the fundamental step upon which the automatic support generator is built.

Starting with the top level, we proceed to determine the shadow of every level on adjacent level below it, calculating the *stale shadow* at every level. This results in a *composite shadow* that is all the support required for the part. *Stale shadow* is shadow that has been unchanged for at least the last two levels. *Fresh shadow* is the new shadow generated on the current level by the previous level up. The stale shadow forms the bulk of the support, and can be built in either the primary material or the alternate material. By accumulating only the stale shadow, and keeping the fresh shadow unmerged at the interface of the part and the composite stale shadow, we are able to build the fresh shadow in the alternate material if desired. Figures 6a and 6b show the part and the supports generated by this method.

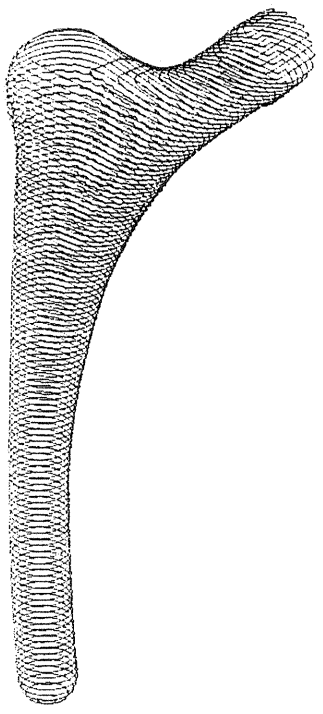
All of the computation is carried out on curves offset from original part curves so as to separate the resulting shadow from part curves by an amount input by the user and dependent on the material.

8. Examples

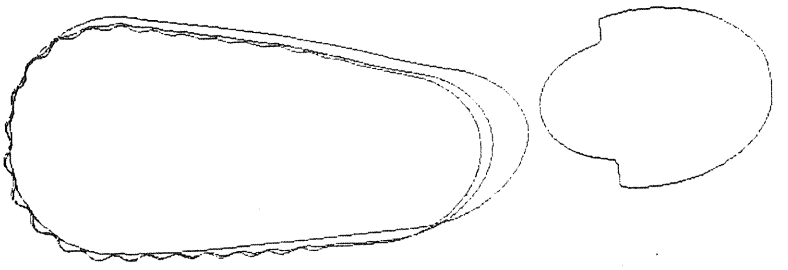
Figs 7a and 7b show SLC files of two parts with and without supports. For the same part, Figs 8a and 8b depict supports generated with a road-width of 0.01 inches and a road-width of 0.08 inches, respectively. These demonstrate that our algorithm is able to determine self-porting features, based on road-width, and avoids creating supports there.

9. Enhancements in Progress

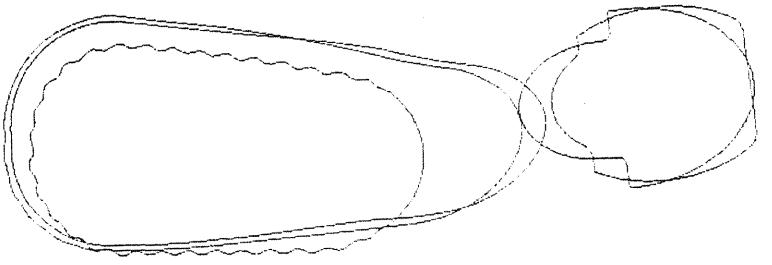
We are studying other materials that can be used as the release material. We are also attempting to determine empirically the largest length of material that can be self-supporting between columns of material underneath (bridge-span), to further optimize the quality and quantity of supports. For stilt-like supports, we are determining a height/base ratio that is stable. If a tall column of support is greater than this ratio, we try to increase its base area while avoiding part geometry (trim off the portion that runs into the part). We also expect to improve the performance (speed) of the automatic support generator.



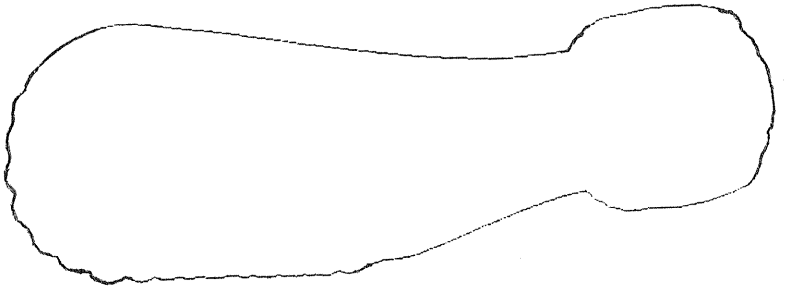
a.

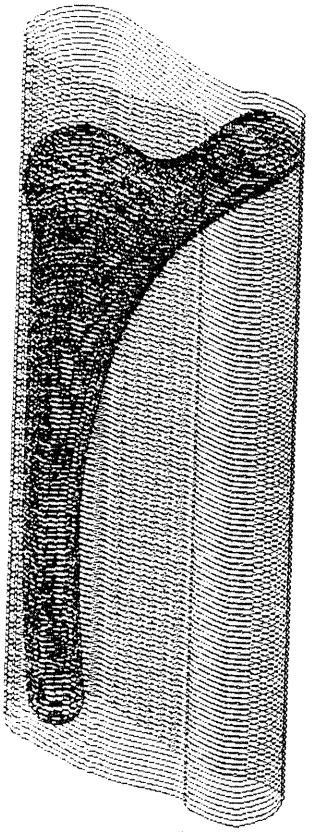


b.



c.





e.

Figure 1. Supports by Containment.

a. SLC File of a part needing supports.

b & c. Curves contributing to part extremes.

d. The container - result of merging all extremes.

e. SLC file of part, with supports by containment.

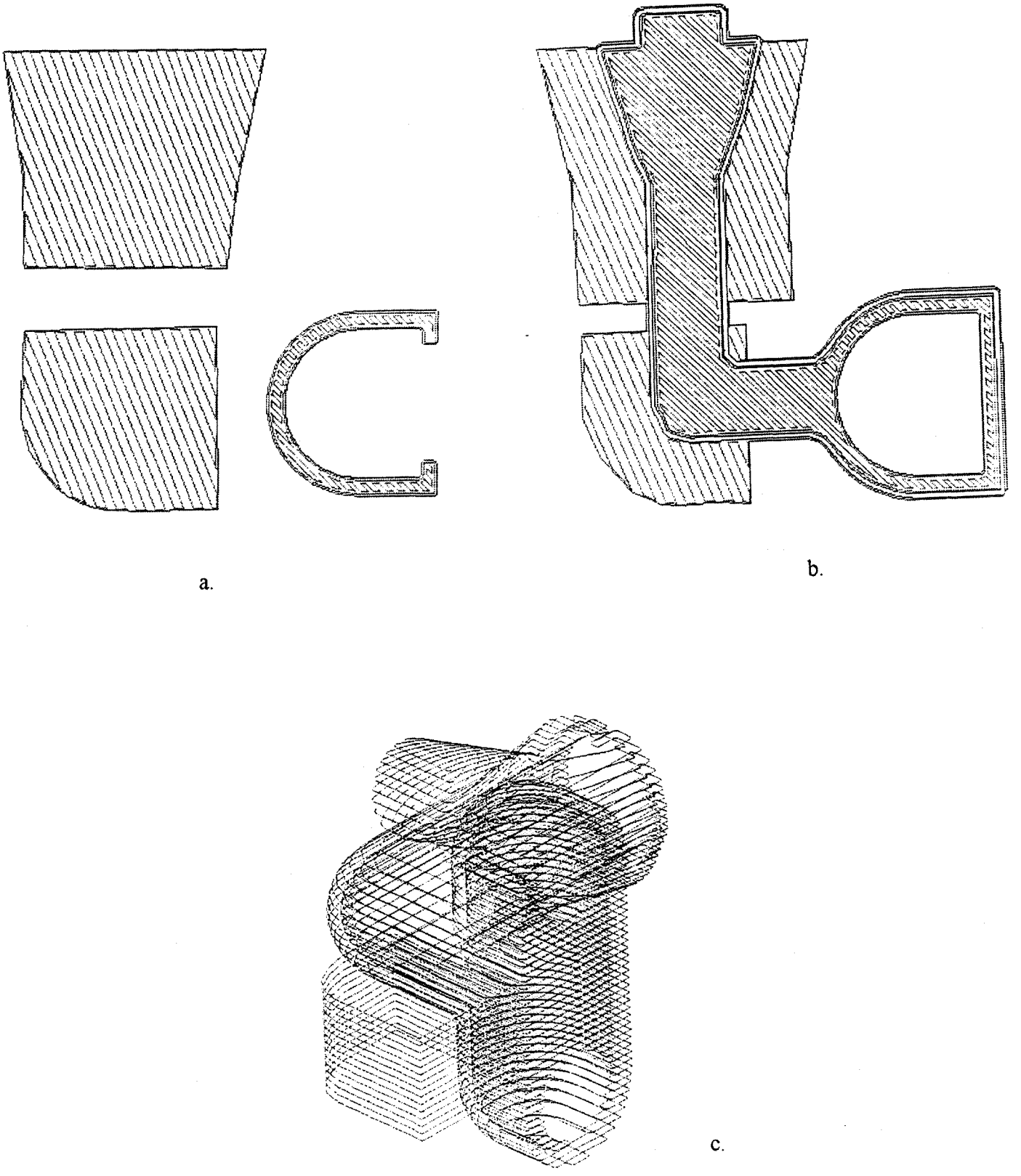


Figure 2. Supporting a Region
a. Region curve not interfering with the part curve, on a certain level.
b. Region curve automatically trimmed where they run into the part.
c. Supports generated by marking a region.

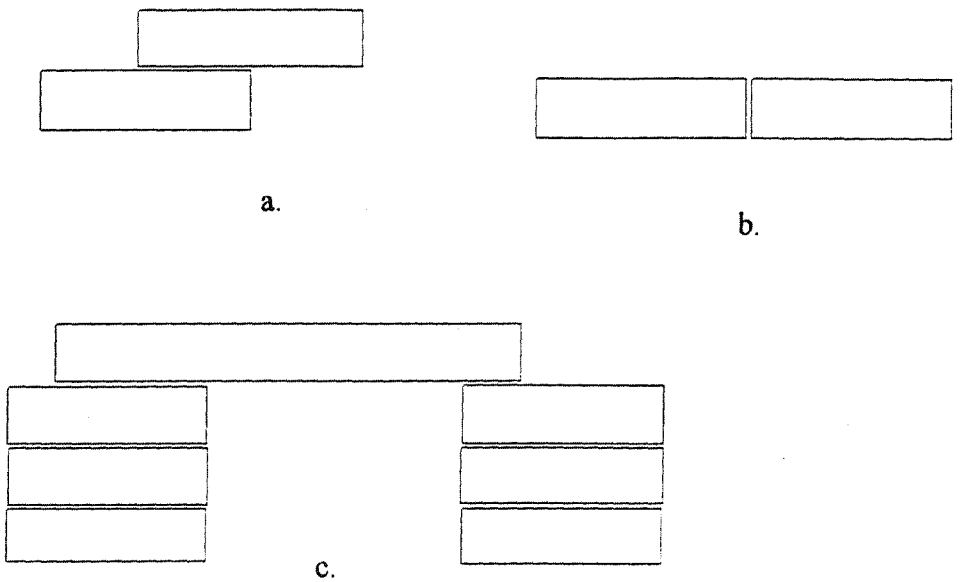


Figure 3. Self Supporting Features a. Cantilevers, b. Links, c. Bridges

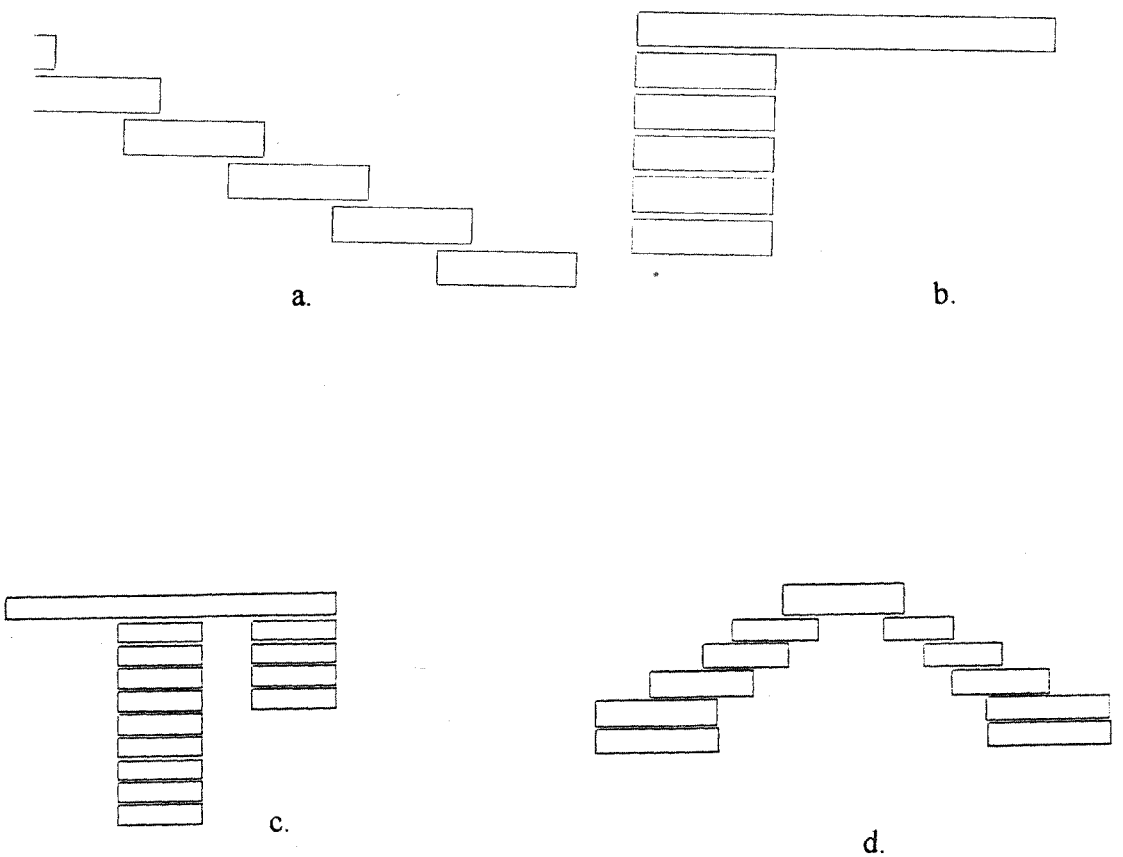
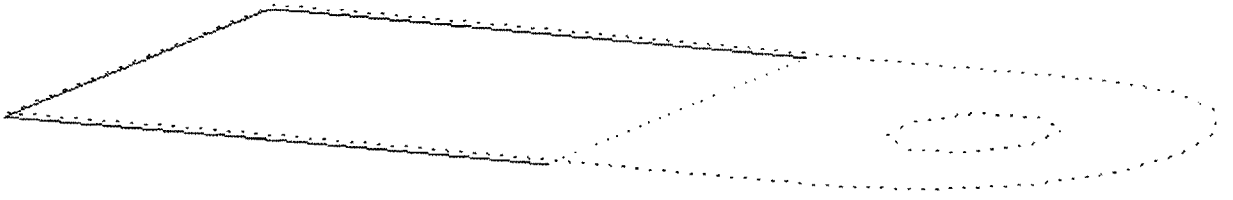
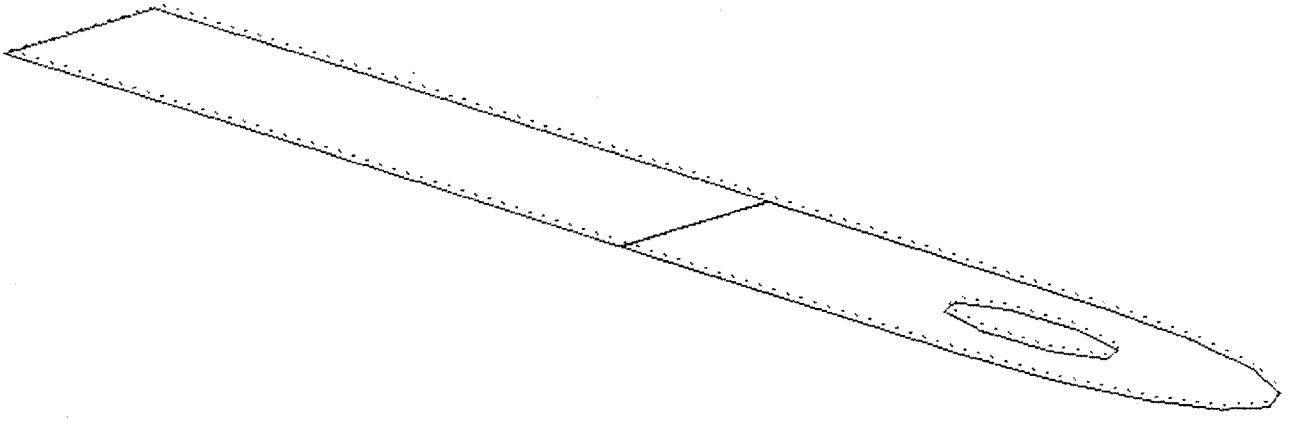


Figure 4. Features Requiring Supports. a. Sharp Inclines. b. Sudden Flats. c. Hooks. d. Arches/domes.

a.

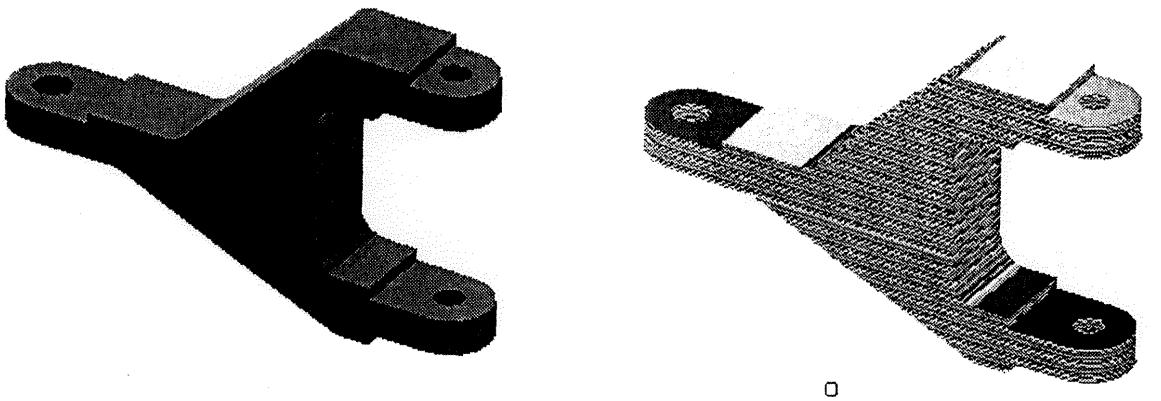


b.

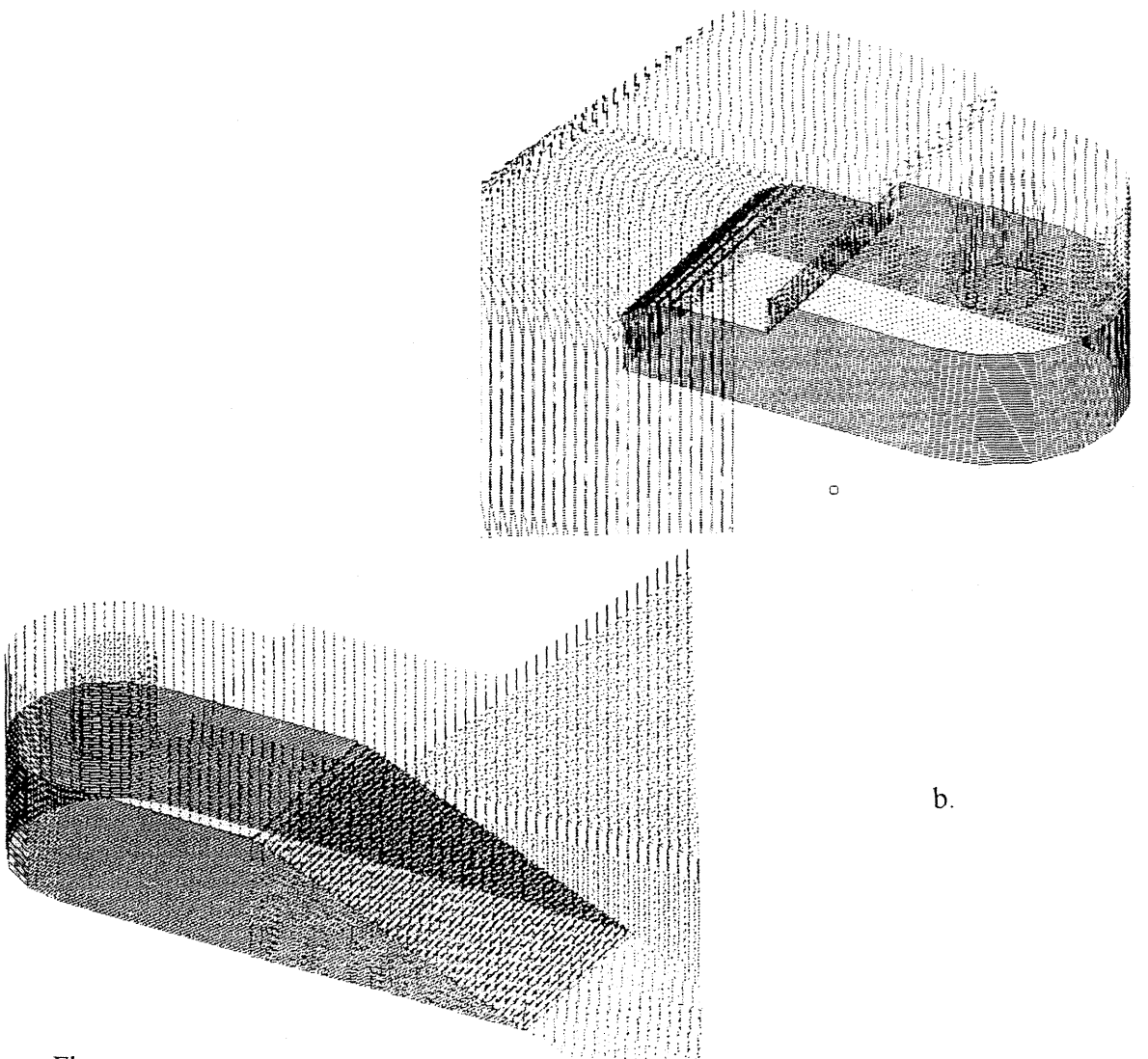


c.

Figure 5. Basic Step in Automatic Support Generation
a. level n from an SLC file b. Level n-1 from the SLC file.
c. Shadow of level n on level n-1.

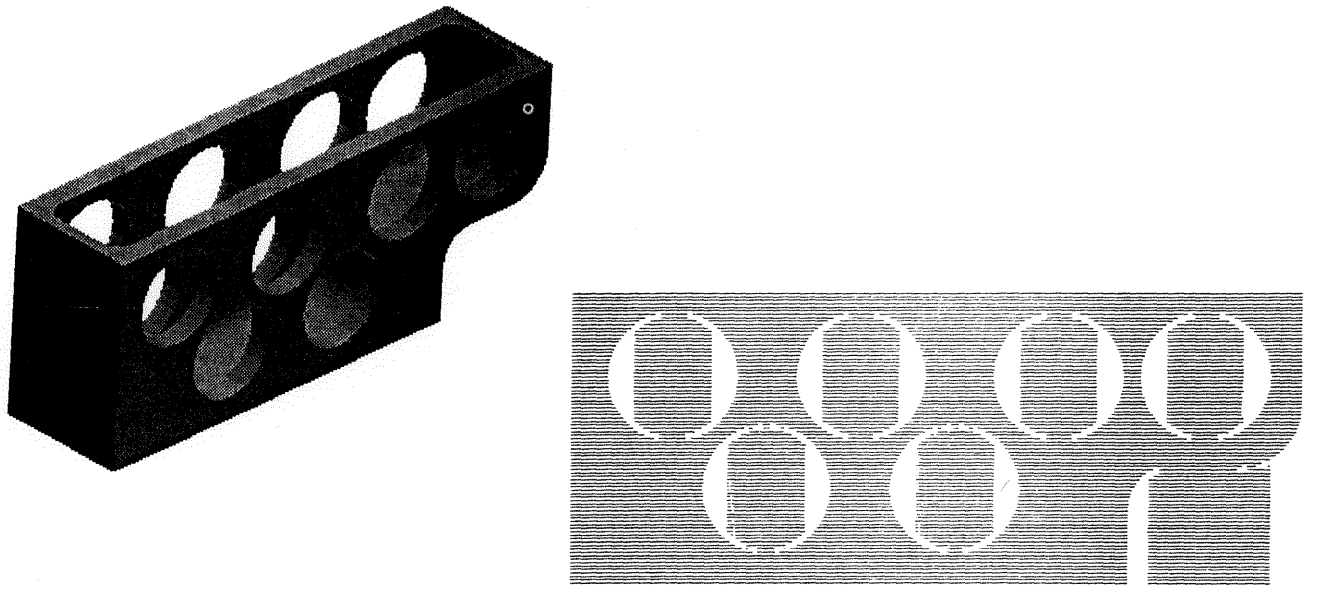


a.

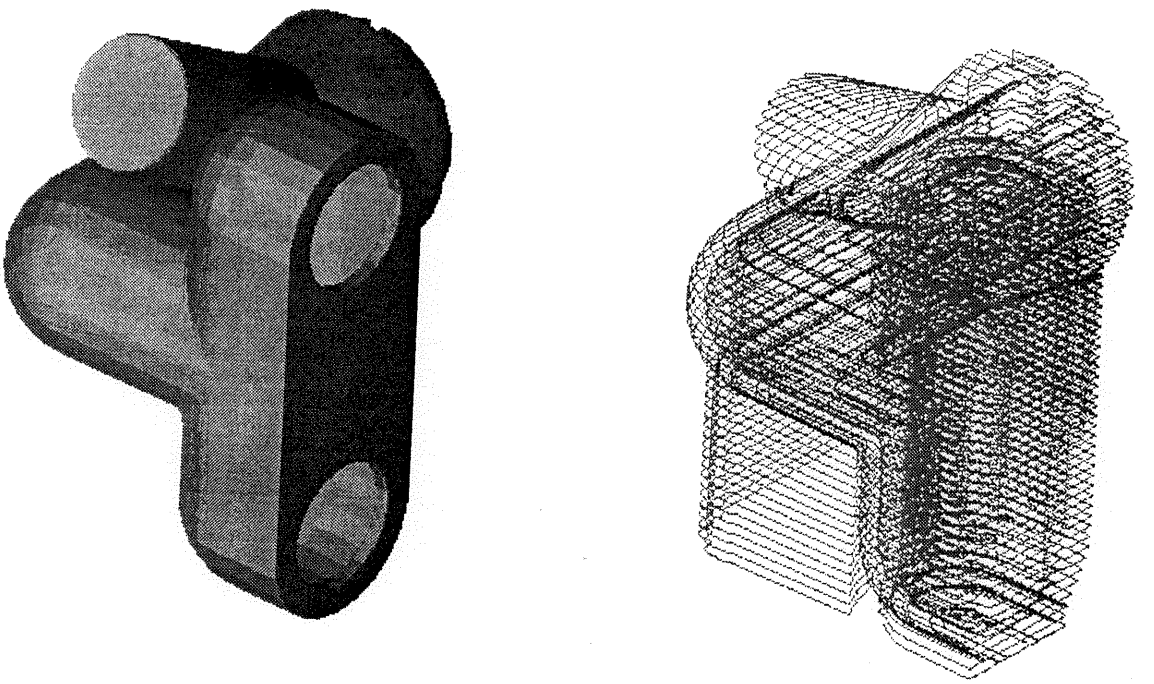


b.

Figure 6. Result of Automatic Support Generation
a. The part and its SLC file. b. Supports generated , showing details.

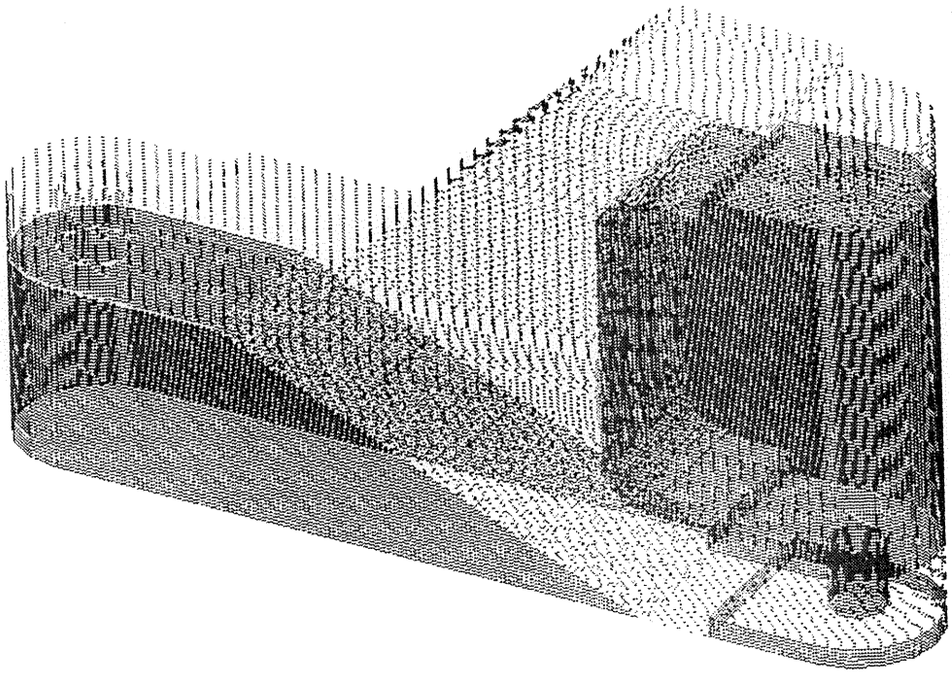


a.

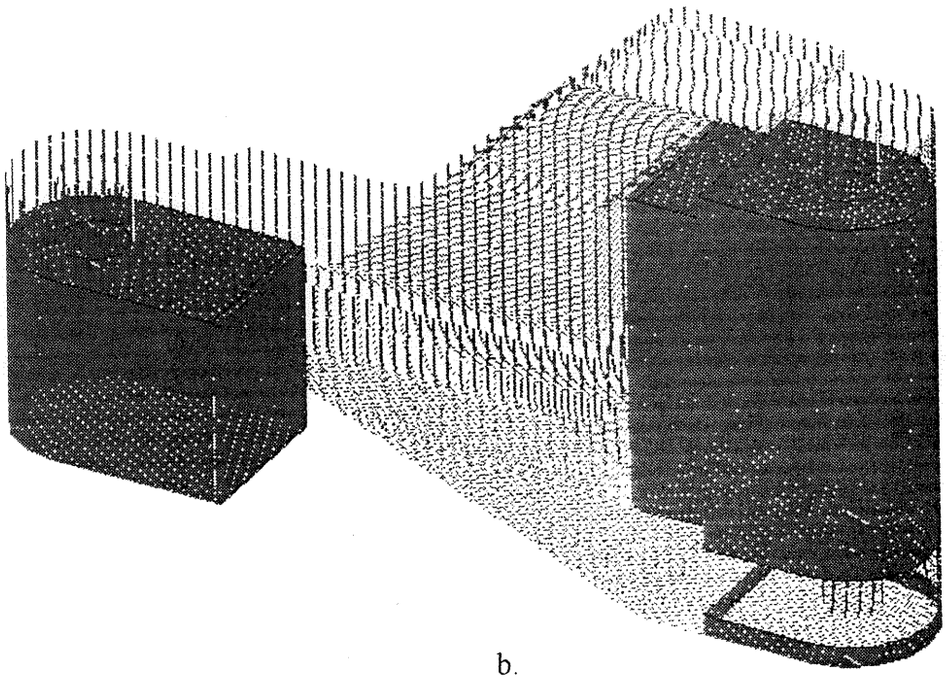


b.

Figure 7. More Results from Automatic Support Generation



a.



b.

Figure 8. Supports Generated with different road-widths. a. 0.01 inches. b. 0.08 inches.