

A Memory Efficient Slicing Algorithm for Large STL Files

S. H. Choi and K. T. Kwok

Department of Industrial and Manufacturing Systems Engineering
The University of Hong Kong, Pokfulam Road, Hong Kong
Email: shchoi@hkucc.hku.hk and ktkwok@hkusua.hku.hk

ABSTRACT

This paper proposes a memory efficient slicing algorithm for Rapid Prototyping (RP) processes. The algorithm is aimed to overcome the constraints of computer memory inherent in the conventional slicing methodologies. It extracts from the Stereolithography (STL) file the facets that intersect with the cutting plane to process the slice data and the topological information. Reading only the facets of the current layer greatly reduces the amount of computer memory required and involves less computationally intensive searching operations. Thus, large STL files of virtually unlimited sizes can be sliced to facilitate the RP process. The algorithm is also relatively fault-tolerant in that inconsistent contour due to defects of the STL file may be more effectively repaired.

The topological information of the layer contours can be subsequently processed by further operations, such as hatching, physical fabrication or virtual simulation. To cater for the variations of RP processes, the Common Layer Interface (CLI) format is adopted as the output interface.

1. INTRODUCTION

Rapid Prototyping (RP) is an additive fabrication process. Contrary to the traditional subtractive machining processes such as milling, RP processes use liquid, powder or sheet materials to form a part layer by layer. These parts are used in various stages of the product development. The role of RP has become more important since it was introduced about 10 years ago.

Fig.1 shows the flow of a typical RP process. The first phase is to validate the 3D CAD model, which is then orientated with the optimal orientation with respect to the build time and the surface quality. Support structures are then generated based on the process requirements. The final model is then sliced with a set of horizontal planes. Each horizontal plane gives a piecewise linear contour, which is then crosshatched to determine the laser paths to control the sintering or solidification process. The process is repeated for the next layer until the model increment to its final shape.

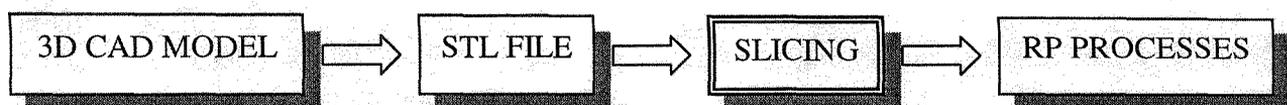


Fig. 1 STL as an interface between CAD and RP

2. SLICING STRATEGIES

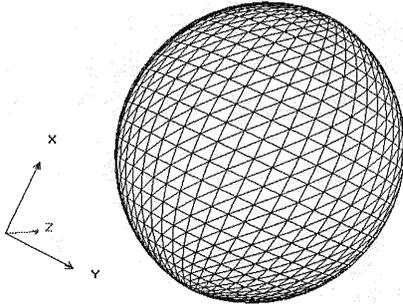


Fig. 2 Sphere in STL format

When creating parts layer by layer, a CAD model of the part is required [1]. Currently, STL format is the *de facto* standard representation of CAD data input. Tessellation is the process that converts a CAD model to a STL file. We consider now only the CAD model as the input, despite there are lots more data input methods such as reverse engineering data and layered data from mathematical programs.

A STL file consists of an unordered list of triangular facets covering the outer surface of the model. Each facet is described by a set of X, Y and Z coordinates for each of the three vertices and a unit normal pointing outside of the solid. An example of STL sphere is shown in Fig. 2. As STL is just an approximation of the surface model, the user needs to define the chordal value, which is the maximum distance from the surface to the vector representing the surface of a facet. STL format offers good geometry approximation of the CAD model if small layer thickness is used. However, the smaller the tolerance, the larger the number of facets required to represent the solid, and hence the bigger the file size. Other cases would also result large STL file sizes, such as complex medical scan data which require high resolution for specific vital purposes (Fig. 6), life-sized models of large objects to be fabricated by RP processes (Figs. 7 and 8) or virtual models developed for different applications (Fig. 9).

The major disadvantage of tessellated or facet models used in most current RP systems is that they are poorly suited for representing highly curved objects. A simple solution to improve the overall surface quality is to increase the number of facets and decrease the slice thickness that results in large STL file. Hitherto, commercially available slicing algorithms usually read the whole STL file into the computer memory to reconstruct the topological relationship of the surface facets. However, a complex or accurate CAD model is often too large to read into the memory. Consequently, slicing cannot take place and thus becomes a bottleneck in the RP process. This bottleneck limits the use of RP technology in certain areas where highly detailed or complex models are required.

2.1 Direct Slicing

Direct slicing has been proposed to alleviate the problem of STL by skipping the facet representation and slice the CAD model directly [2]. It can therefore, solve the big STL file problem if implemented between the CAD system and RP machine. Implementation of direct slicing based on binary representation (B-rep) solid model and constructive solid geometry (CSG) had been proposed. However, similar to sloping surface approach [3], the major disadvantage of direct slicing is that it is machine/CAD dependent. Solid model representation such as B-rep and CSG are fundamentally different, and a generic model format has yet to be devised.

2.2 Adaptive Slicing

Adaptive slicing was proposed to maximize the geometry without affecting the accuracy. In contrast to uniform slicing, adaptive slicing increases the slice density in highly convoluted regions, and reduces it in other regions wherever possible. Adaptive slicing mainly addresses the geometry issues and need a specific RP system to achieve the desired results. However, there are still not RP machines that fully support or able to take full advantages of adaptive slicing. The build thickness for a single layer is usually fixed to a small range between certain upper and lower limits. In case of large model size adaptive slicing offers no compensation. The current RP systems such as Selective Laser Sintering (SLS) and Fused Deposition Modeling (FDM) only fuse a layer of building material having a thickness of less than 0.5 mm in order to have a higher accuracy along the build direction.

Most researchers tend to propose another file format to replace the STL file standard. Jacob [4] proposed a new rapid prototyping interface, which includes facets topological information. However, similar to other proposed formats [5], they failed to provide imperious advantages over STL file as a new standard. Others focus on process parameter optimization [7,8] but less effort has been given to solve the big STL file problem that provides a practicable and economical solution to the existing RP users. Due to the nature of STL file, no sorting or reconstructing algorithm is able to give satisfactory results. Once if the input STL file is too large to fit in the system memory or has minor defects, it will stop the RP process or even 'hang' the machine. It is the purpose of this paper to solve this memory limitation problem.

It can be concluded that STL representation of CAD models is still the most versatile interface for commercially available RP systems. STL format has been a unified input to all downstream processes that are performed in rapid prototyping as well as other layered manufacturing. STL export option has been implemented in most CAD systems that ensures a unified neutral interface between CAD and RP systems. Solving the memory bottleneck provides an economical and practicable option for RP users. A memory efficient slicing algorithm that eliminates the bottleneck has been developed and is discussed in the following sections.

3. THE MEMORY EFFICIENT SLICING ALGORITHM

The memory efficient slicing algorithm adopts a different approach. Instead of storing the whole model into the computer memory, this algorithm reads only the facets that intersect with the cutting plane as shown in Fig. 4. Hence although the number of facets in a STL file depends on the model density, the number of facets read in the computer memory is greatly reduced. The difference is even more observable as the facet density becomes higher.

Fig. 5 shows the overall flowchart of the slicing program. The facet model will be scanned to obtain properties such as maximum and minimum vertices, model name and number of facets. Once the model has been scanned, all the properties will be displayed on the screen. Since a valid STL model should be in the positive Cartesian space, a Model Manipulator module has been developed to handle the case of negative vertices, which may be produced from the CAD to STL conversion or part orientation optimization. The simple Model Manipulator uses the translation

matrix for the calculations of model rotation, model transition and model scaling. With the Model Manipulator, the STL model can be properly orientated before the slicing process.

In the slicing stage, the algorithm mainly involves manipulation of the facet data and comparison of z-coordinates of the facets, and then calculates the line segment that defines the contour for the layer. The line segments are vectors following the anti-clockwise rule, which will be stored in the computer memory for contour construction. The sliced line segments are sorted and joined to form a closed contour by simple head-to-tail searching mechanism with a given tolerance. The tolerance is the maximum distance between the head and the tail of two joining line segments, as shown in Fig. 3. The choice of the tolerance value is critically related to the chordal value of the STL file [10]. In this algorithm, the maximum tolerance is set to equal to the shortest edge in the STL file.

Once the model is sliced in each layer, the topological relationship needs to be sorted. The anti-clockwise nature of the line segments can be used to identify the inner loop and the outer loop. Other topological information such as number of holes and number of surrounding contours should be calculated also. This is done by the ray-crossing or ray-shooting containment test [11], which is implemented in the Contour Sorter Module. The Sorting Module outputs the sorted layer contours for subsequent processes. The Layer Viewer Module is used to view the contour output. The user can view the layer contour to check its validity.

Finally, the sliced data are used to generate a data file in the Common Layer Interface (CLI) format [12]. The CLI data format containing the contour information is independent of the RP machines. CLI is a simple and efficient format for data input to all Layer Manufacturing Technology systems based on a 2.5D layer representation. Moreover, medical scan data is already accommodated in CLI format [13].

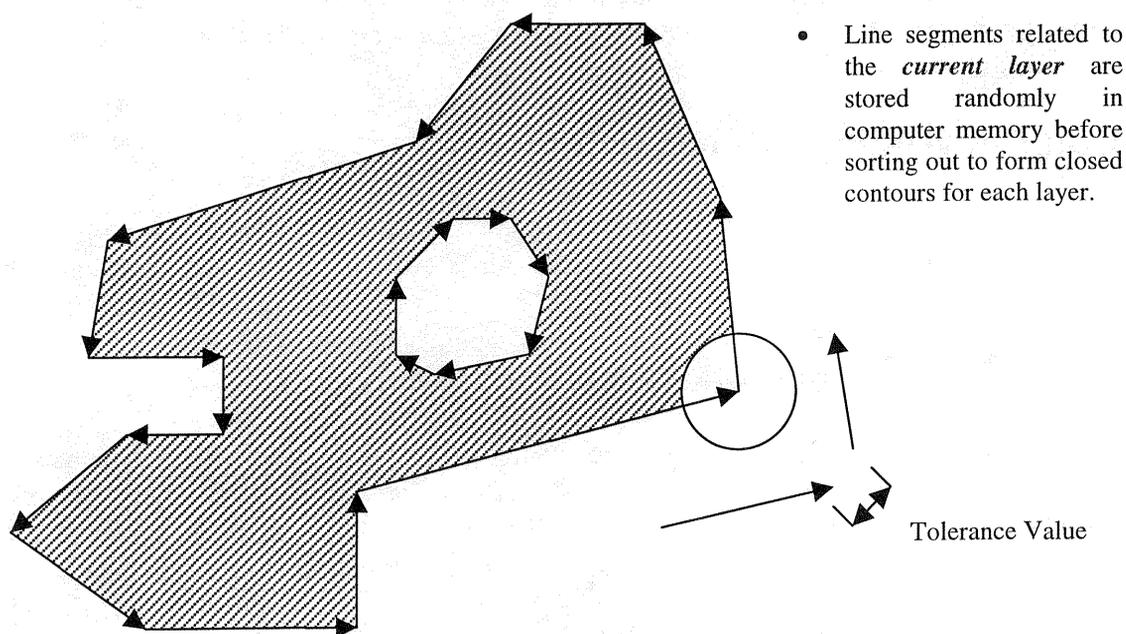


Fig. 3 Sample layer contour following the anti-clockwise rule

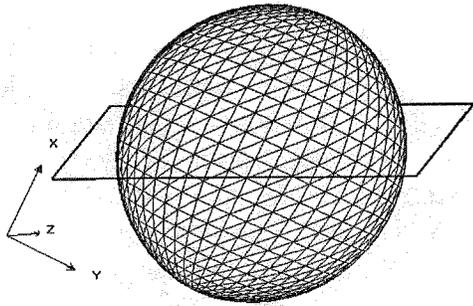


Fig. 4a: Conventional slicing algorithm stores *all facets* in computer memory, which limits the size and complexity of the design and thus leads to breakdown of the rapid prototyping process.

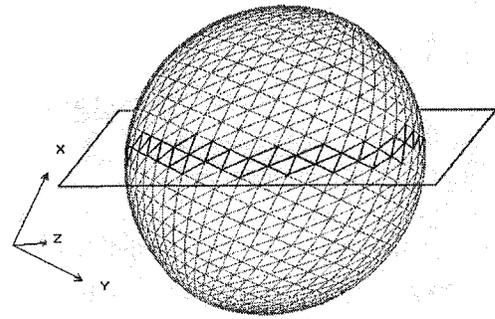


Fig. 4b: The memory efficient slicing algorithm stores only the *facets intersect with the cutting plane*, thus alleviates bottlenecks due to memory limitations and thereby facilitates rapid prototyping of large and complex models.

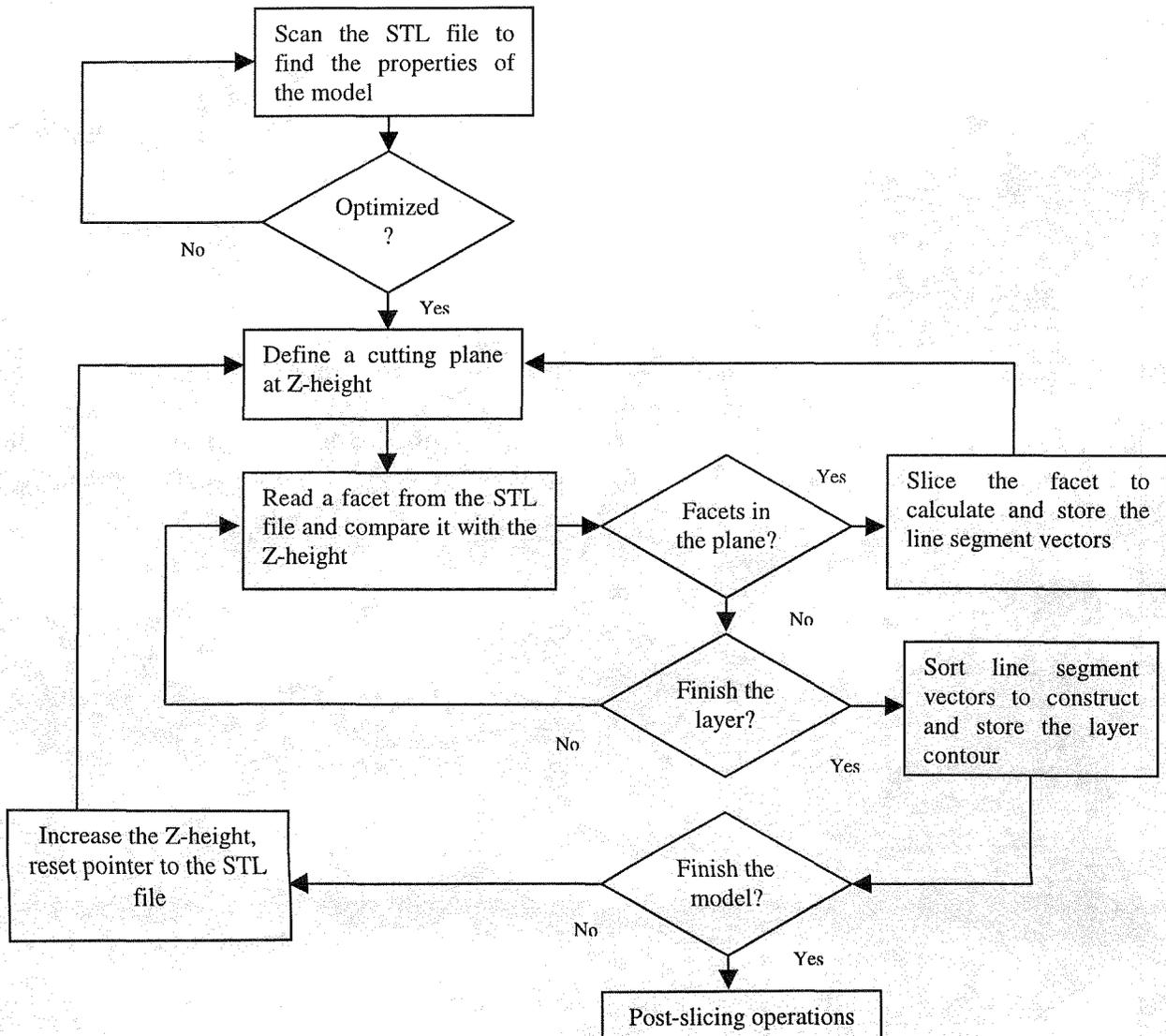


Fig. 5: Flowchart of the memory efficient slicing algorithm

4. EXAMPLES

The algorithm has been developed in Visual C++ and is now being enhanced. Samples STL files of different sizes and shapes have been sliced to test the functionality and stability of the program. The configuration of the computer used is as follows: Intel Pentium 166MHz processor, 64MB Ram, a 4.2GB hard disk with SCSI controller and Microsoft Windows 98. Fig. 6 shows a human skull model, which consists of 767,256 facets. A bell and a car of 426,572 and 151,350 facets are given in Figs. 7 and 8 respectively, while Fig. 9 shows a computer generated virtual model that consists of 74,634 facets. Screen captures of Layer Viewer Module are also shown in Figs. 6 to Fig. 9. Layer Viewer displays contours in different colours for better visualisation and easier error detection. Indeed, most commercially available slicing algorithms will not endure even one fault facet [6], and slicing will not proceed unless all the triangles are perfectly linked together in the STL file. In contrast, the proposed algorithm will slice the STL file even if some edges are missing. The slice can be viewed in Layer Viewer module, and hence the user can manually edit the slice in case of unbound contours.

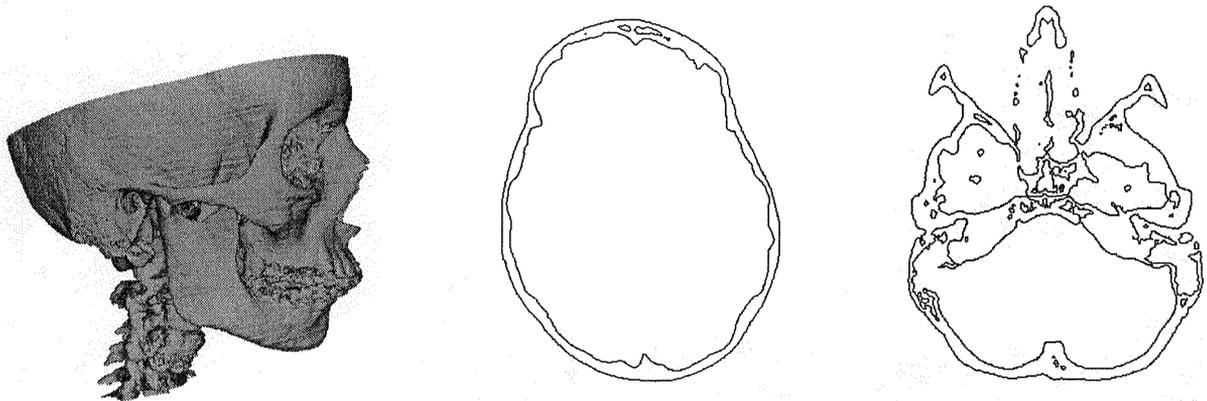


Fig. 6 A Human Skull Model (36MB binary STL file, with two slices at different Z-height generated by the algorithm. Mean processing time per slice is approximately 1 minute)

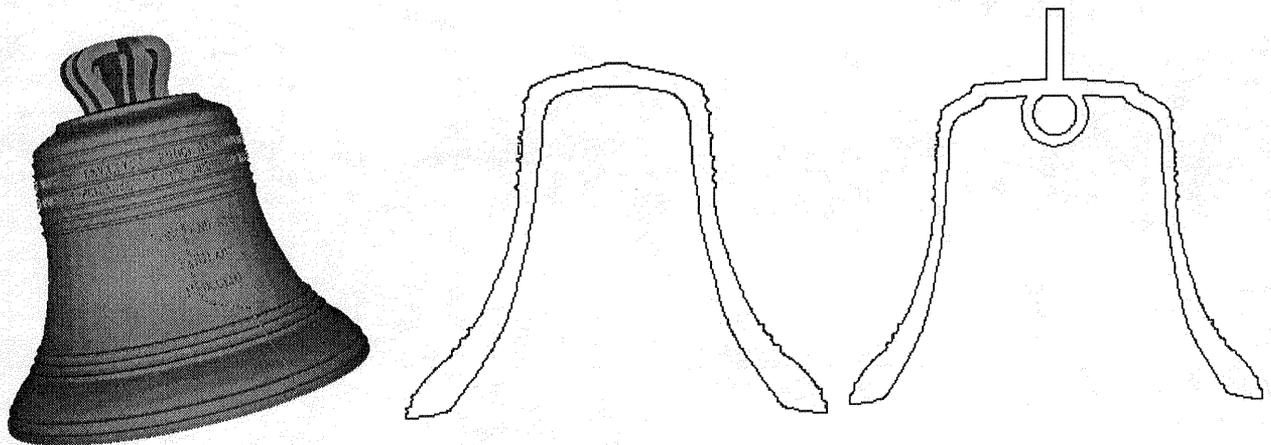


Fig. 7 A Bell Model (21 MB binary STL file, with two slices generated at different Z-height by the algorithm. Mean processing time per slice is approximately 30 seconds)

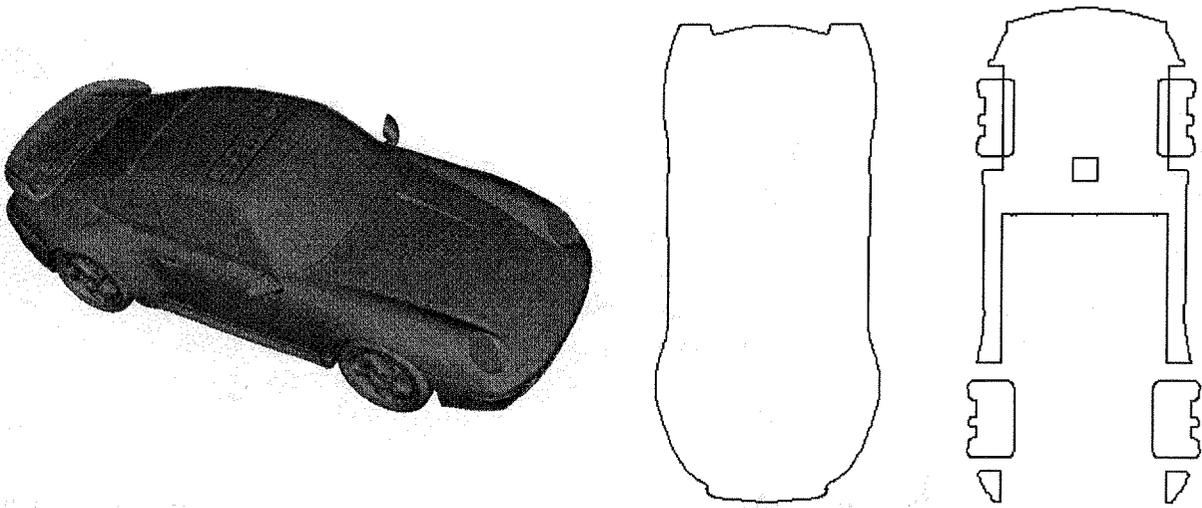


Fig. 8 A Car Model (7.21MB binary STL file, with two slices generated at different Z-height by the algorithm. Mean processing time per slice is approximately 12 seconds)

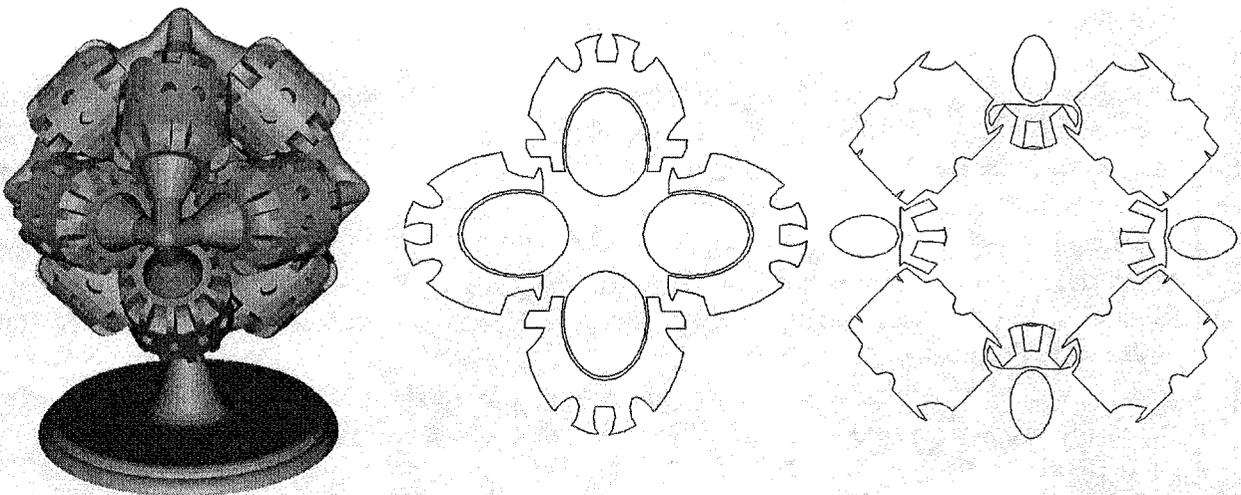


Fig. 9 A Virtual Model (3.55MB binary STL file, with two slices generated at different Z-height by the algorithm. Mean processing time per slice is approximately 5.2 seconds)

5. CONCLUSION

The productivity of layered manufacturing can be improved by using a versatile slicing algorithm, which should be able to slice STL models of any sizes and complexities. Traditional slicing algorithms can only handle relatively small STL models because of the limitation of computer memory. A new memory efficient slicing algorithm has been developed to solve this problem. The advantage of this algorithm is its ability to slice STL models of virtually unlimited sizes. It processes only the facets of a single layer to free up memory. This feature would be essential to make large objects such as a life-sized vehicle, or complex and detailed models used in medical applications.

ACKNOWLEDGEMENT

The authors would like to thank the Research Grant Council of the Hong Kong Government and the CRCG of the University of Hong Kong for their financial support for this project.

REFERENCES

1. Vinod Kumar and Debasish Dutta, "An assessment of data formats for layered manufacturing", *Advances in Engineering Software* vol. 28 (1997), pp. 151-164
2. P. J. de Jager, "A comparison between zero and first order approximation algorithms for layered manufacturing", *Rapid Prototyping Journal*, vol. 3 number 4 1997, pp.144-149
3. Zhiwen Zhao and Luc Laperriere, "Adaptive Direct Slicing of the Solid Model for Rapid Prototyping", <http://ecoleing.uqtr.quebec.ca>
4. Gan G.K. Jacob, "Development of a new rapid prototyping interface", *Computers in industry* vol. 39 (1999), pp. 61-70
5. Anna Kochan, "Rapid prototyping trends", *Rapid Prototyping Journal* vol. 3 number 4, 1997, pp. 150-152
6. K. F. Leong, C. K. Chau and Y. M. Ng, "A study of Stereolithography File Errors and Repair", *The International Journal of Advanced manufacturing Technology* (1996) 12:407-414
7. Zhu Wei-Ming and Yu Kai-Ming, "Pre-processing Technologies for Rapid Prototyping and Rapid Tooling – a Survey", *CIRT International Symposium – Advanced Design and Manufacture in the Global Manufacturing Era*. August 21-22, 1997, Hong Kong, pp. 802-808
8. Mino Bablani and Amit Bagchi, "Quantification of errors in Rapid Prototyping Processes, and Determination of Preferred Orientation of Parts", *Transaction of the North American Manufacturing Research Institute of the SME*, vol. xxiii, May 1995, pp. 319-324
9. Michael J. Laszlo, "Computational Geometry and Computer Graphics in C++", Prentice Hall, 1996, pp.116-130
10. Joel E. McClurkin and David W. Rosen, "Computer-aided build style decision support for stereolithography", *Rapid Prototyping Journal* vol. 4 number 1, 1998, pp. 4-13
11. R. L. Hope, P.N. and P.A. Jacobs, "Adaptive slicing with sloping layer surfaces", *Rapid Prototyping Journal* vol. 3 number 3, 1997, pp. 89-98
12. <http://www.cranfield.ac.uk>, Cranfield University EARP
13. Common Layer Interface, Version 2.0, pp.19-21