Graphics Based Path Planning for Multi-Axis Machine Tools

Joshua A. Tarbutton[1], Thomas R. Kurfess[2] and Tommy M. Tucker[3]

[1]Clemson University, trbttn@clemson.edu
[2]Clemson University, kurfess@clemson.edu
[3]Tucker Innovations, Inc., tommy@tuckerinnovations.com

ABSTRACT

This paper discusses a graphics-based approach to the tool path and trajectory planning problem found in machining and robotics applications. Ray casting is used to determine part and blank surfaces from voxel model representations. Tool paths are generated from the ray intersections with the voxels and uncut material is used to determine optimal re-orientation. The algorithm presented is a first step towards automatic path generation for 5-Axis machine tools. The groundwork is laid to demonstrate the power of graphics hardware on manufacturing problems. The resulting tool path is validated by experimental results carried out by multi-axis milling of a free-form surface with undercut regions.

Keywords: path planning, voxelization, GPU, 5-Axis, machine tool.
DOI: 10.3722/cadaps.2010.835-845

1 INTRODUCTION

Path planning has been the subject of over 30 years of research and yet it still consumes tremendous amounts of resources. To date, there is not an automatic solution to tool path generation. However, significant advances can be made towards a solution with modern graphics hardware. Dedicated graphics functionality can be exploited to provide process planners with faster and better performing tools to determine tool paths for complex part surfaces such as those found in the die and mold industry. Tool path planning is the art of transforming a surface representation from a part model into a series of commands that can be given to an arbitrary machine tool to produce a real part. Tool path planning plays a central role in the manufacturing sector and affects nearly every part found in the automobile, aviation, and shipping industries. CAD packages have continually been increasing core functionality to represent and join multiple complex surfaces into single part models. CAD designers are taking advantage of this to design parts with higher surface complexity than ever before. CAM packages attempt to come up with methods to give process planners the ability to generate tool paths based on these part models. The majority of tool path planning algorithms are based on these well-established methods: iso-parametric [1-3], iso-planar [4-6], iso-scallop [7, 8], and polyhedral. Iso-parametric and Iso-planar were the most widely used methods in CAM systems [9] but the most recent commercial CAM systems are using polyhedral machining [10].

The iso-parametric methods are the mapping of a parametric surface, $\mathbf{s}(u,v)$, onto Euclidean space. This mapped parametric path yields the cutter contact point, or CC. A main drawback of this method is

that the mapping of surfaces with large surface gradients results in large gaps between successive paths in Euclidian space. Adaptive methods have been introduced to correct this phenomenon but at the cost of increased algorithm complexity [2]. Iso-parametric methods are useful when there is only one surface but multiple surfaces often result in gouging [7] and compound surfaces are commonly converted to triangular meshes [11]. When the underlying free-form surfaces are extend and combined, as is normally the case in complex surface modeling, the tool paths generated by the iso-parametric tool paths often are no longer boundary confined [11] and surface repair is necessary to ensure error-free tool paths [9].

Iso-planar methods (also referred to as drive surfaces, Cartesian method, constant z-level method, contour method, direction-parallel method, one-way tool path, or zigzag tool path [12]-[13]) generate parametric curves by intersecting the part surface by infinite planes. The plane-surface intersections are used to create parametric curves which are used to generate CC points and cutter location (CL) points. According to Kim [13], determining the plane-surface intersections is very computationally demanding; but, because the algorithm is very reliable when used for complex surfaces, it is still widely used in die and mold manufacturing. A disadvantage of this approach parts with large surface gradients will produce uneven paths if the intersecting planes are spaced evenly and adaptive methods must be used to account for these regions [14].

Iso-scallop methods are extensions of the above two approaches to ensure even scallop height. For the iso-planar method, the interval of the iso-plane adjusted so that accounts for steep surfaces and results in uniforms scallop height when used. When the iso-parametric method is used, the parametric surface is manipulated so that the resulting tool paths are mapped uniformly to Euclidian space.

Polyhedral machining was introduced by Duncan in 1983 [15] and is superior to the computationally intensive surface checking algorithms found in the above methods [10]. However, polyhedral machining was not widely used simply because of hardware cost and memory limitations; which are mostly solved with today's hardware. Polyhedral machining creates a tool path based on surfaces represented by polyhedrals. Polyhedral surface models can easily be generated by tessellating a part's surfaces. Nearly all CAD packages have tessellation algorithms that produce tessellated surfaces with the desired surface accuracy. Although, the tessellated surface is an approximation of the true surface, the surface accuracy can be constrained to whatever tolerance is acceptable at the cost of slightly more memory. The tessellation algorithms are extremely robust and capable of combining any number of surfaces into a single triangular mesh [11]. Tessellated surfaces have been widely used for rapid prototype systems where the surface accuracy of the final part is much less important than the rapid creation of an initial part. In addition, the de-facto standard file format for rapid prototyping systems (as well as CAD system file sharing) is the STL file format; which is little more than a header with triangle vertices and their normal. Because hardware limitations are no longer a limiting issue, polyhedral machining has made its way into commercial CAM packages [10]. Determining the CC and CL from a polyhedral model has been accomplished by two main approaches which are: the point/curve-based approach [6, 10, 15-17] and the inverse tool offset surface approach [18-20].

This paper discusses a new tool path planning approach based on ray-casting and voxel models, both of which are concepts found ubiquitously in computer graphics. Voxel models have been used extensively in the area of "Volume Rendering" which is concerned with rendering, in a very realistic way, a scene with volumetric elements. Applications of volume rendering are predominantly concerned with the rendering of objects with various densities such as clouds and smoke for the movie and gaming industries or for rendering the bone and organ data obtained from an MRI [21]. Ray-casting is a cornerstone of graphics research [22, 23] and is used in CAD systems to render the modeled object to the GUI or "scene" [24]. Ray-casting is used in photo realistic rendering as a means to render a virtual scene as close to reality as possible by physically modeling rays of light [25]. These concepts are used to create the state of the art games and movies enjoyed by many.

The concept of ray casting has previously been used in tool path planning to find the highest patch surface in a work area [26] or specific intersection with a surface [27]. Voxel models have been used in virtual machining simulation [28] but have not been used to generate tool paths. In this research, a STL file is used as the basis for voxelization and rays are cast at the iso-surface of the voxel model for fast surface detection. Tool paths are generated from these intersections in a non-traditional nearest neighbor approach suitable for rapid prototyping. The tool paths are generated

based on an optimal plane orientation similar to 5-axis machining where two of the axes are used as indexing axes.

2 PATH PLANNING AND THE GPU

Polyhedral tool path planning has become a reality due to the technological advances in the CPU and memory hardware but advances in the Graphics Processor Unit or GPU may have an even more significant impact on polyhedral tool path planning. The insatiable appetite of the multi-billion dollar gaming industry has radically transformed the GPU hardware technology resulting in GPU's that are capable of hardware implementations of various graphics functions and highly parallel multiprocessing. Every year the CPU is able to run significantly faster than previous years but its speed is simply not enough to meet gaming demands. In spite of faster multi-core processors provided by CPU manufacturers such as Intel, AMD and etc. the GPU industry is booming. This is due to the fundamental differences between a CPU and GPU.

The CPU executes *sequential* code on general hardware suitable for a variety of computing tasks to support a wide range of software whereas the GPU executes *parallel* code on hardware dedicated to operating on millions of floating point numbers to support code specifically written to exploit the GPU. To support these operations, GPUs have what are called Streaming Multiprocessor Cores. These are similar to CPU processors except in their density. Current high-end CPUs have 2 – 4 cores (e.g. Intel Duo, Intel Quad) running at approximately 3GHz whereas current GPUs have 64 – 240 processor cores operating at 1.3GHz. There is a tremendous difference in the computational horsepower of the CPU vs. GPU. Computational problems that have been ported from the CPU to the GPU have resulted in a nominal acceleration of 10-100x's [29] which can result in significant implications for manufacturing. Implementation of this technology is essential to maintaining a competitive manufacturing edge and it is particularly well suited for polyhedral tool path planning.

Historically GPU's were limited to fixed functionality but roughly seven years ago, GPU manufacturers opened up the "graphics pipeline" to computer programmers. The programmer was granted the access and ability to manipulate raw vertex and pixel information stored in graphics memory. With this new capability, research accelerated in what is commonly known as General Purpose Graphics Processing Unit computing or GPGPU computing. A tremendous amount of research in non-graphics fields such as physics, economics, medicine, and engineering soon followed [30]. These new tools have enabled game designers and movie producers to create environments and special effects that are indistinguishable from real life events. But this technology is not limited to the entertainment industry and manufacturing can be significantly impacted by exploiting this new hardware functionality. The approach presented in this paper depends on portions of the hardware implementation of standard GPU functionality. In addition to exploiting native functions, the parallel programming capabilities of the GPU can also be used to significantly enhance the approach presented here.

3 VOXELIZATION

Voxelization is the process of transforming an object into a volume element. A voxel is a volume element much like a pixel is a picture element. A voxel can be thought of as stacked photographs or layers of pixels in space. The medical industry utilizes voxel models to store the data from CT and MRI scans of bones and internal organs in medical imaging. Graphic artists use voxels in games to simulate a variety of effects including semi-transparent media such as smoke, fog, and clouds. In this research, a voxel model is used to represent a part and blank. The part is voxelized directly from the polyhedral model and the resulting surface of the part is defined by the exterior voxels; but, this is not a requirement as there exists methods to voxelize part models directly from CSG representations [31]. The voxelization is carried out by exploiting native GPU functionality (such as depth testing) and native memory architecture (such as depth buffers and texture memory).

The part is voxelized for the twofold purpose of fast intersection detection with the rays and fast calculations on the voxel volume. Analytically solving for the surface intersections takes significant computation time when using the iso-parametric, iso-planar, iso-scallop, or polyhedral methods.

Surface intersection is drastically simplified by using a voxel model. However, this speed comes at the expense of large memory consumption.

The voxelization process is similar to the one described in [32]. Sub-sampling is used here to increase the accuracy of the resulting voxelized surface and a diagram of this process is shown in Fig. 1. The voxel model is created by clipping the model at a certain location at eight times the desired resolution of the output model and rendering the rear-facing and front-facing polygons onto texture memory. The depth resolution is increased by a factor of eight by rendering intermediate textures between slices. Note that Fig. 1 only represents a resolution that is only four times the desired resolution for clarity. The voxelization algorithm which makes use of a computer Graphics Processing Unit or GPU and the generalized process is as follows:

1. Place the near plane of the camera at the desired slice location and the far plane beyond the extents of the part.
2. Allocate texture memory on the GPU at 8 times the desired resolution of the voxel model.
3. Draw all the background pixels as zero in the high resolution GPU texture memory.
4. Draw all back faces of the polyhedral model as 255 in the GPU texture.
5. Draw all front faces as 0 in the GPU texture if the depth test is passed indicating that the front faces are in front of the back faces.
6. Make 8 textures in depth and average each 8x8 grid to determine the voxel value for the final voxel volume and repeat for the next slice.

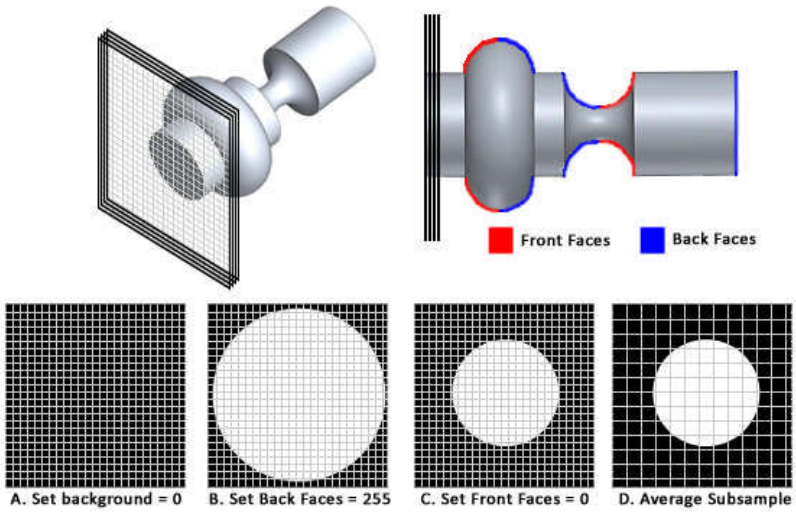

Fig. 1: Rendering slices into memory.

This algorithm was implemented in C++ and DirectX with an nVIDIA GTX8800 graphics card. Screen shots of the voxelization process are shown in
Fig. 2. The part to be tessellated is shown on the left and consists of a raised freeform surface in the form of a tiger paw and a sunken paw. The tessellated part is shown in red next to the gray part in the top row. The clipping plane is shown at the right side of the part of the tessellated part and what is saved to GPU memory is shown in the box below the part. The clipping plane starts at the right side of the part and performs the operations described above then moves to the next plane. The right image shows the output of the algorithm where the tessellated part is now a volumetric part, or voxel model.

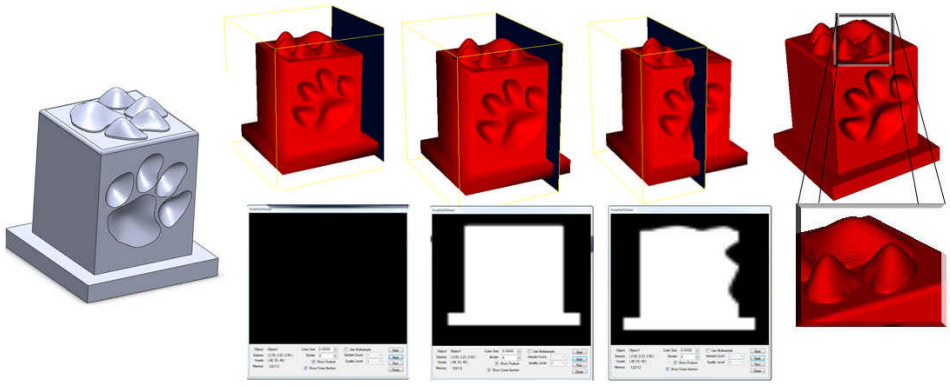

Fig. 2: Voxelization Process.

Once the part has been voxelized, it represents a digital discrete volume. The tool path planning algorithm described below relies on the definition of a blank as well as a part. The blank is also represented by a voxel model. The voxel model of the blank is created based on dimensions of a part volume. The blank is created by copying the voxel volume and assigning all the voxels to their "on" value then taking the part volume values and assigning those "off" in the blank volume. The blank and part volumes are show in
Fig. 3. The blank material is removed by successive tool paths based on the depth of cut until only the part remains. Defining a voxel model of the blank also allows simulation of the material removed per pass.

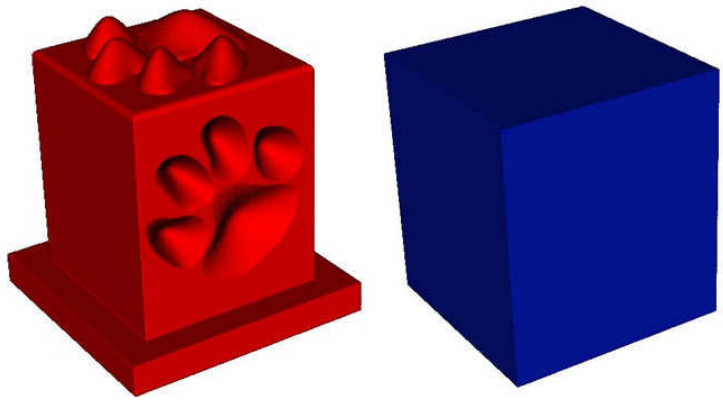

Fig. 3: Voxel part model and voxel blank model.

4 RAY-CASTING INTERSECTION

The concept of ray casting is used to determine the cutter contact (CC) points for the tool path. Rays are cast at the voxel model of the blank and part from a bounded plane in an arbitrary location inside the working volume of the machine tool. The plane is created by translating and rotating two orthogonal vectors initially aligned along two of the tool axis which is important for machines that are 4-axis and higher. For 3-axis machines, the created plane is parallel to the xy-plane. A grid is generated on the plane according to the voxel size, in the simplest case this is a 1:1 ratio; but, can be extended to any arbitrary interval for roughing and finishing paths. A ray consists of a point plus a normal multiplied a distance,

$$r = p_o + \mathbf{n} \cdot d \tag{1}$$

where, r is the ray location, p_o is the origin of the ray, $\mathbf{n}$ is the direction of the ray and also the machining plane's normal, and d is the distance along the ray. Rays are cast from the machining plane by increasing d until the voxel surface is "hit" as shown in Fig. 4. Tri-linear interpolation is used within the voxel volume based on the ray's current location to ensure that the ray hits the voxel model's iso-surface. Each ray's intersection with the voxelized part is stored. The intersections between the ray and blank and the ray and part are used to build a list of surface intersections. This approach allows the rays to determine all the exterior surfaces of the part and blank along that ray. These surfaces intersections are used to determine the cutter contact points and the maximum depth of cut at that location as well as undercut locations.

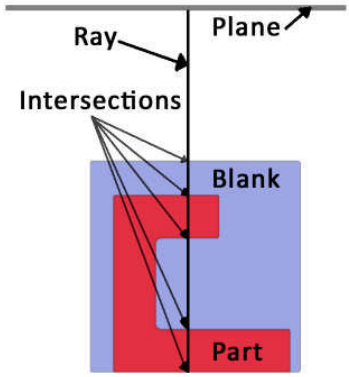

Fig. 4: Ray casting into part and blank voxel volumes.

The location of the intersection of the ray with the part is used to create either a raster path or a nearest neighbor path suitable for rapid prototyping. The surface normals are also calculated for use in higher axis machining as shown in
Fig. 5. As can be seen in the figure, the cutting path can be created for freeform surfaces.

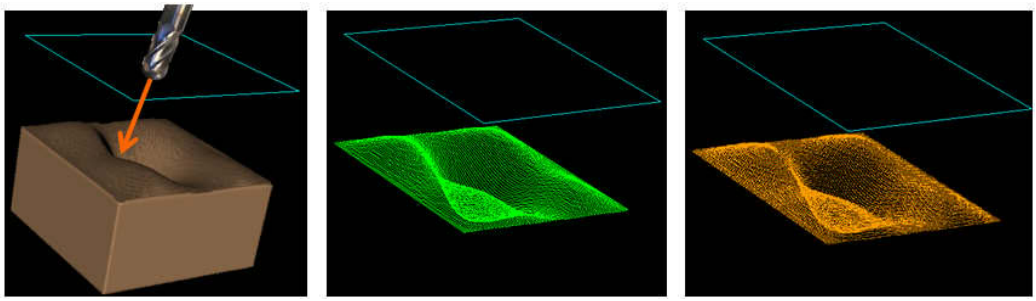

Fig. 5: Ray casting and raster tool path creation.

5 OPTIMAL MACHINING PLANE DETERMINATION

Five axis machining can result in faster machining times and higher quality parts. To maintain the stiffest possible machining conditions, 5-axis machining is often performed by using two of the rotary axes as indexing axis that are fixed while the other three perform the machining. This is also called 2+3 machining and all five axis are employed to re-orient the part. The work presented here closely mimics 2+3 machining because all machining is done from a plane. In addition to stiffer machining

characteristics, 2+3 machining is often employed because it is easier to manually create tool paths using CAM software.

The algorithm for determining the optimal plane orientation is described below. First a series of planes are created and a bounding box is created on them by projecting the bounding box of the part and blank. Then a grid is generated from the bounded plane that represents the part surface and another grid is generated for the blank surface. These grids are generated by casting rays from the plane and recording the surfaces intersections in the grid. The grid is essentially a point cloud for both the part and blank surfaces. However, the part grid points are associated with the blank grid points by the ray cast from that location on the grid. This is used to subsequently create tool points according to a depth of cut from any machining plane orientation.

Once the part and blank grids are generated, the blank voxels in the orthogonal view of the machining plane are removed because they are all reachable and would be removed in the case that that plane is selected as the machining plane. Once the voxels in view of the plane are cleared from the scene, the remaining voxels are counted. This process is repeated until a plane reaches a suitable number of voxels removed or is the minimum in a set of test planes.

For this research, the test planes are created according to the following geometric shape: gyro-elongated square bi-cupola. This shape is show in Fig. 6 which can be constructed by a plane normal to a vector originating from the center of a square and ending at the shortest distance to every face, edge, and vertex.

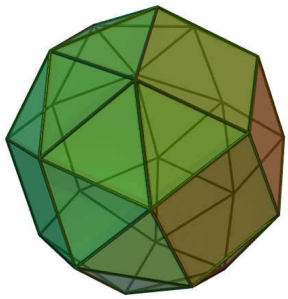

Fig. 6: Test shape for optimal plane.

The gyro-elongated square bi-cupola consists of 26 planes which are the test planes used in this research. The planes are shown below.

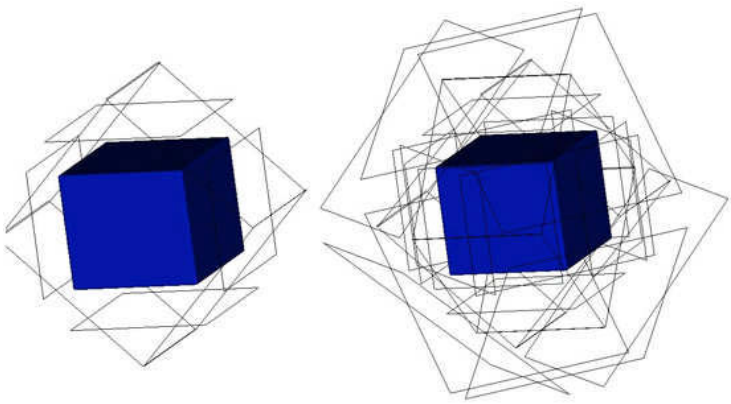

Fig. 7: First 8 planes and all 26 planes.

The voxels removed during a search for the best plane are shown below. This is repeated for the 26 planes and the best plane is used as the machining plane.

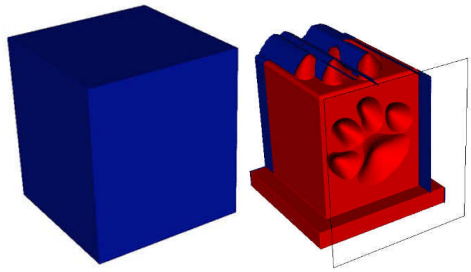

Fig. 8: Before and after voxels are removed during optimal plane search.

Once the machining plane has been determined the tool paths are generated from this plane as discussed below. The material is removed during the machining process and the search process is repeated; only this time with the machined voxels no longer contributing.

6 PATH PLANNING

Path planning is carried out from the CC points generated by the ray intersections with the voxel volume. Each ray is cast from a point on a "grid" located on the machining plane. The resolution of the grid corresponds to the voxel size in the simplest case but this is by no means a restriction. The data structure used to store the ray intersections with the voxel closely resembles the grid. The location of the intersections with the blank volume and part volume are used to determine the final CC point. The final cutter contact point for that location in the grid is equal to the blank surface location plus the depth-of-cut-distance along the ray that penetrates that point in the surface. The CC point is updated with the parts iso-surface if the ray hits the part before the depth of cut is reached the. This yields a grid of tool points. Two types of tool paths considered in this approach which are: a raster path and a nearest neighbor path.

A raster or lawnmower path starts at a tool point in a corner of the grid and increments along one dimension the increments the other dimension then decrements the original dimension and repeats. For this approach only the z-height is varied. The z-height is given by the CC point determined by the ray casting approach. A simple raster path is shown in
Fig. 9.

Fig. 9: Raster path.

The raster path works well for simple shapes where there are large planes of similar height in the part to be machined. However, for more complicated surfaces where a part has large variations in the z-value such as in freeform surfaces, the raster path approach becomes inefficient because it traverses the same area multiple times to reach areas of varying height.

An alternative approach to the raster path is to use a nearest neighbor approach. The nearest neighbor approach is a concept borrowed from image processing. The first tool point begins at the corner of the grid at the specified CC point and tests all the neighbors in a flexible fashion. The default is a clockwise approach where from the start point the neighbors are tested in a clockwise fashion.

Each surrounding point is tested to determine if it is able to be machined (not part of the part) and if it has already been visited. The tool point is added to the tool path if it passes these two tests. If any of the points are able to be machined then they are candidates to be added to the tool path. A decision is made at this point as the whether the tool path will continue to points that have already been visited or retract and engage the blank material at a new point. This is entirely flexible and governed by a past history buffer. The output paths from this approach are shown in
Fig. 10.

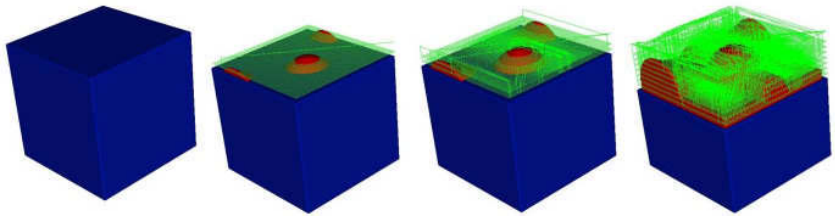

Fig. 10: Nearest neighbor paths.

7 POST-PROCESSING AND 3-AXIS MACHINIG RESULTS

The tool path must be converted from points in voxel space to commands that can be processed by NC machine tools to machine the part in real dimensions. This is accomplished by post-processing the tool path and translating voxel space points into G and M-codes which are the de-facto standard used on NC machine tools. The post-processor is able to scale the output commands during post-processing so that a single tool path generated from the ray casting voxel approach presented here is able to make a variety of sized parts based on the same path. Once the tool path has been post-processed it can be immediately used in a machine tool to produce a part however further optimization will improve its performance. 3-Axis machining based directly on the G-codes has successfully been performed as shown in Fig. 11. Typically once CC points are defined, a method is employed to produce cutter location (CL) data [6, 33]. This has not been implemented yet and the results presented here represent tool paths based on CC data only. The part below was cut with a 1/8” ball end mill on a 3-Axis Okuma 46 MVU and the material is Necuron610 which is suitable for test milling. As can be seen from the figure, the method presented here is able to detect ray-surface intersections and produce cutter contact points. However, the resulting parts suffer from gouging and other artifacts that are the subject of future work.

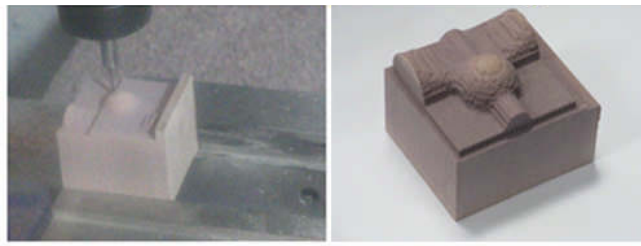

Fig. 11: Machining results for different voxel resolutions.

8 MULTI-AXIS PLANE RE-ORIENTATION SIMULATION RESULTS

The process used to create the part is shown in Fig. 12 and follows the method discussed above. First, a part model is tessellated and saved as an STL file (a polyhedral model). Then the part is voxelized and an optimal machining plane is determined (shown in red in the figure below). Then the rays are cast from the optimal machining plane to detect surface interactions. From the surfaces intersections,

tool paths are planned using a nearest neighbor approach (shown as green below). The algorithm produces G-Code that can be used to machine a part.

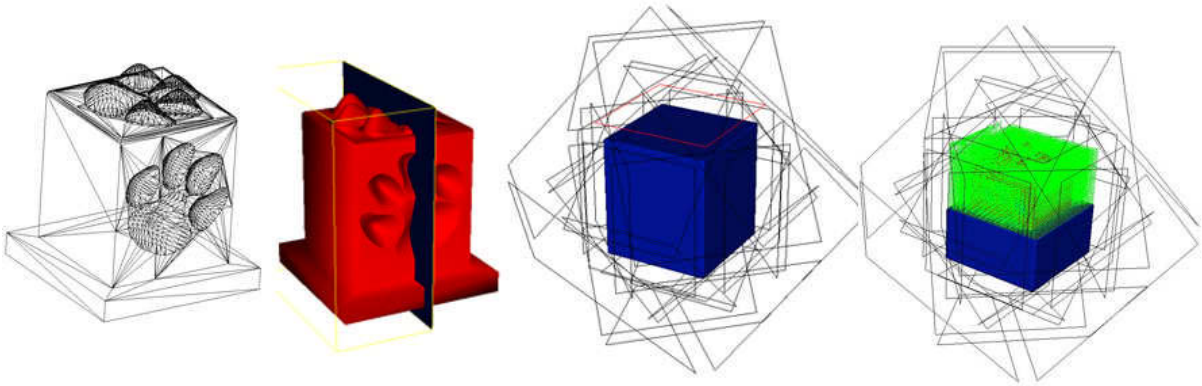

Fig. 12: From CAD model to experimental results.

9 SUMMARY

In this paper a machining algorithm was introduced based on an alternative graphics-based approach to the classical tool path planning procedures. The computer graphics concept of using a voxel model was used to represent the part and blank. The graphics concept of ray tracing was used to generate the cutter contact points based on intersections with the voxelized part. A way to optimally determine a machining plane orientation was introduced that is based on the voxel models. The method presented here can be considered a polyhedral machining method because it is based on a polyhedral model of the part. The polyhedral methods are becoming more widely used because memory limitations are no longer an issue. Experimental and simulation results demonstrated the potential of this alternative path planning approach.

10 ACKNOWLEDGEMENTS

The authors would like to express their gratitude to the National Science Foundation for supporting their work on Graphics Accelerated Manufacturing as well as Okuma for their generous donation of equipment and support resources.

REFERENCES

[1] Broomhead, P.; Edkins, M.: Generating NC data at the machine tool for the manufacture of free-form surfaces, International Journal of Production Research, 24(1), 1986, 1-14.
[2] Elber, G.; Cohen, E.: Tool path generation for freeform surface models, Proceedings on the second ACM symposium on Solid modeling and applications, 1993, 419-428.
[3] Loney, G. C.; Ozsoy, T. M.: NC machining of free form surfaces, Computer-Aided design, 19(2), 1987, 85-90.
[4] Bobrow, J. E.: NC machine tool path generation from CSG part representations, Computer-aided design, 17(2), 1985, 69-76.
[5] Huang, Y.; Oliver, J. H.: Non-constant parameter NC tool path generation on sculptured surfaces, The International Journal of Advanced Manufacturing Technology, 9(5), 1994, 281-290.
[6] Hwang, J. S.: Interference-free tool-path generation in the NC machining of parametric compound surfaces, Computer-aided design, 24(12), 1992, 667-676.
[7] Kim, S.-J.; Yang, M.-Y.: A CL surface deformation approach for constant scallop height tool path generation from triangular mesh, International Journal of Advanced Manufacturing Technology, 28, 2006, 314-320.

[8] Suresh, K.; Yang, D. C. H.: Constant scallop-height machining of free-form surfaces, Journal of engineering for industry, 116(2), 1994, 253-259.
[9] Yang, D. C. H., et al.: Boundary-conformed toolpath generation for trimmed free-form surfaces via Coons reparametrization, Journal of Materials Processing Tech., 138(1-3), 2003, 138-144.
[10] Jun, C. S.; Kim, D. S.; Park, S.: A new curve-based approach to polyhedral machining, Computer-Aided Design, 34(5), 2002, 379-389.
[11] Yuwen, S., et al.: Iso-parametric tool path generation from triangular meshes for free-form surface machining, The International Journal of Advanced Manufacturing Technology, 28(7), 2006, 721-726.
[12] Park, S.; Choi, B. K.: Tool-path planning for directio-parallel area milling, Computer-Aided Design, 32(1), 2000, 17-25.
[13] Kim, B. H.; Choi, B. K.: Guide surface based tool path generation in 3-axis milling: an extension of the guide plane method, Computer-Aided design, 32(3), 2000, 191-199.
[14] Ding, S., et al.: Adaptive iso-planar tool path generation for machining of free-form surfaces, Computer-Aided Design, 35(2), 2003, 141-153.
[15] Duncan, J. P.; Mair, S. G.: Sculptured Surfaces in Engineering and Medicine. 1983: Cambridge University Press.
[16] Choi, B. K.; Jun, C. S.: Ball-end cutter interference avoidance in NC machining of sculptured surfaces, Computer-Aided Design, 21(6), 1989, 371-378.
[17] Choi, B. K., et al.: Compound surface modeling and machining, Computer-Aided Design, 20(3), 1988, 127-136.
[18] Choi, B. K.; Kim, D. H.; Jerad, R. B.: C-Space approach to tool-path generation for die and mold machining Computer-Aided Design, 29(9), 1997, 657-669.
[19] Takeuchi, Y., et al.: Development of a personal CAD/CAM system for mold manufacture based on solid modeling techniques, Annals of the CIRP, 38(1), 1989, 429-432.
[20] Inui, M.: Fast Inverse offset computation using polygon rendering hardware, Computer-Aided Design, 35(2), 2003.
[21] Stone, S. S., et al.: Accelerating advanced MRI reconstructions on GPUs, Journal of Parallel Distributed Computing, 68(10), 2008.
[22] Appel, A.: Some techniques for shading machine renderings of solids. in Proceedings of the AFIPS Joint Computer Conferences, 1968, Atlantic City, New Jersey.
[23] Cook, R. L.; Porter, T.; Carpenter, L.: Distributed ray tracing, ACM SIGGRAPH Computer Graphics, 18(3), 1984.
[24] Roth, S. D.: Ray casting for modeling solids, Computer Graphics and Image Processing, 18, 1982, 109-144.
[25] Glassner, A.: Principles of Digital Image Synthesis. 1995, San Fransisco: Morgan Kaufman.
[26] Griffiths: Toolpath based on Hilbert's curve 26(11), 1994, 839-844.
[27] Zhang, D.; Bowyer, A.: CSG set-theoretic solid modelling and NC machining of blend surfaces, in Proceedings of the second annual symposium on Computational geometry, 1986, Yorktown Heights, New York.
[28] Jang, D.; Kim, K.; Jung, J.: Voxel-Based Virtual Multi-Axis Machining International Journal of Advanced Manufacturing Technology, 16(10), 2000, 709-713.
[29] Hwu, W.-M., et al., Compute Unified Device Architecture Application Suitability, in Computing in Science & Engineering. 2009. p. 16-26.
[30] nVIDIA, GPU Acclerated Research. 2009 [cited 9/25/2009]; Available from: http://www.nvidia.com/object/cuda_home.html#.
[31] Fang, S.; Liao, D.: Fast CSG voxelization by frame buffer pixel mapping, in Proceedings of the 2000 IEEE symposium on Volume visualization, 2000, Salt Lake City, Utah.
[32] Engle, K., et al., Real-Time Volume Graphics, 2006, Wellesley, MA: A K Peters, Ltd.
[33] Carter, J.A.; Tucker, T. M.; Kurfess, T. R.: 3-Axis CNC Path Planning Using Depth Buffer and Fragment Shader, Computer Aided Design and Applications, 5(5), 2008, 612-621.