Rapid and Flexible Prototyping Through Direct Machining

Frank Wei Li[1], Tyler Davis[2], C. G. Jensen[3], and W. E. Red[4]

[1]Brigham Young University - Provo, frank1li@et.byu.edu
[2] Brigham Young University - Provo, tad29@et.byu.edu
[3] Brigham Young University - Provo, cjensen@byu.edu
[4] Brigham Young University - Provo, ered@et.byu.edu

ABSTRACT

The rapid physical realization of a product from a CAD model prior to full scale manufacturing presents a challenging task for industrial companies, as well as academic researchers. This paper presents a rapid and flexible prototyping system designed to directly connect a CAD system (Alias) with a software based DMAC (Direct Machining And Control) controller. The overall architecture is presented and the tool path planning, animation, and direct machining under Alias are explained. The implementation of the proposed prototyping system is realized on a three-axis mill. An example of milling a real surface through this direct CNC system is presented and conclusions are drawn.

Keywords: free-form surface, rapid prototyping, direct machining, path planning, animation.

1. INTRODUCTION

In automotive, aerospace, shipbuilding, die and mold industries, an original design concept usually begins as a 2D sketch created by industrial designers. This 2D sketch is then transformed into a 3D CAD model through one of two processes. Ideally, the designer will work with an engineer to create a 3D CAD model directly from the 2D sketches. An alternative is to create a physical model, often sculptured from clay by a skilled artisan or sculptor, from which a 3D CAD model is obtained by some form of 3D scanning such as white light imaging. The 3D CAD surfaces are usually modeled as parametric forms that use the well defined mathematics of B-spline, Bezier, and NURBS entities.

To mill out parametric surface, a 3D CAD model is usually translated into some intermediate file format, such as IGES, and is then read into a CAM system. After the CAD model is translated into the CAM system, tool paths can be generated and post processed into CNC codes (M&G codes). A physical model can then be milled out on a CNC machine using M&G codes. This traditional design-to-manufacturing process is time consuming and error prone.

A major limitation with current CNC manufacturing methods is the form of data transfer between the driving CAD model and the machine tool controller—M&G code [3]. Traditionally, M&G code does not support the use of free-form curves such as NURBS [2]. In addition, there is no association between the tool path data and the master CAD model. This means that any modification to the original CAD model requires a regeneration of all subsequent intermediate files as well as the M&G codes. A modification of M&G codes on the shop floor can never be reflected back to the original CAD model. This flow of data makes the task of keeping all relevant files up to date and deleting obsolete files very problematic. Moreover, because of the extra steps needed to generate these intermediate files, and the iterations required to produce a production ready M&G code file, this traditional design-to-manufacturing process increases the time and costs of creating a physical model from a conceptual design.

With the increasing use of these complex surfaces to describe CAD models, it is difficult, if not impossible, to remain in the traditional M&G point/line/arc paradigm of past decades. Traditional approaches to machining curved surfaces have always involved increasing the number of M&G code points (line segments) that the cutter moves through, or fitting the point data with an increasing number of circular arcs. These forms of tool path data representations are inefficient and inaccurate. We, along with other researchers and a few control manufacturers, believe it is desirable to utilize derived NURBS curves for tool path description. Several techniques for Bezier curve-based tool path generation exist currently [4], [8] and new techniques are constantly being developed [1], [17]. To accommodate these needs, the capability to accept free-form curves has recently been added to the M&G standard. However, the lack of associativity between ASCII M&G code tool paths and associative geometric tool path in the CAD model remains a barrier to design creativity.

This paper presents a revolutionary new paradigm for rapid prototyping NC and CNC control by directly connecting a CAD system (Alias) with a software based DMAC (Direct Machining And Control) controller. DMAC technology completely eliminates all intermediate files between the CAD model and the CNC controller by directly processing the geometric tool path entities that are sent from the CAD/CAM system to the motion buffer of the controller [5], [12]. For example, NURBS control points and knot vectors defining the tool paths are sent, not a tessellated approximation of the curve.

Direct machining maintains the efficiency and accuracy improvements seen in NURBS tool path representation, and also retains associativity between the CAD model and the tool paths driving the CNC machine tool. Only one file exists, the master CAD model, which contains the embedded associative process-planned tool paths. This model file is opened by the controller, parsed for the tool path, and then closed when the milling is complete.

2. BACKGROUND

2.1 Related Research

In recent years, efforts have been made, by both industrial and academic researchers, to invent a better design-to-manufacturing process. The goal in machining has been to overcome the limitations existing in current two-, three-, and multi-axis milling processes.

Related research efforts can be found in the principles of open architecture control (OAC) and manufacturing. Three active industrial consortiums, the OSE (Open System Environment for controller) of Japan, the OSACA (Open System Architecture for Controls within Automation systems) of Europe, and the OMAC (Open Modular Architecture Controllers) consortium of the U.S., define and promote the use of open architecture controllers to replace the old closed CNC systems. In academia, several research projects have been undertaken in an attempt to open CNC control. Wright et al. [14, 15] proposed the MOSAIC (Machine Tool Open System Advanced Intelligent Controller) architecture in 1988. Koren et al. [7] in Engineering Research Center for Reconfigurable Machining System at the University of Michigan proposed an open CNC system, named UMOAC. Yellowley et al. [10] at the University of British Columbia proposed and developed a UBC open-architecture controller.

Despite all these efforts by industry and research institutes, to this day there is no standard interface that is agreed on by all machine manufacturers, control vendors, software developers, and machine tool end users. In addition, the lack of associativity between CAD model, CAM system, and CNC machine still remains as the greatest limitation in these architectures.

With open-architecture control and manufacturing as a basis, a more recent technology called STEP-NC has been proposed and is being developed for solving the ongoing design-to-manufacturing CNC problems. STEP-NC is a new process description language that provides an improved interface between the CAD model, CAM system, and CNC controller [9, 13]. This technology attempts to create associativity between a CAD model and its tool paths. However, this technology is only a link between CAM and CNC—not a link between CAD and CNC. STEP-NC requires that a CAD part be translated from its native format (e.g. UG part, Alias wire, etc.) to the STEP format. While this format is purported to be universal and independent, it lacks the ability to retain the original design intent and method within the part file. As a result, modifying the part requires a new STEP file to be created, along with any tool path operations and other STEP-NC related work.

2.2 Part Printer Paradigm

The method we are proposing avoids this problem by connecting any CAD/CAM application directly to a mill through a software layer that essentially acts as a "driver". This direct connection eliminates the need for any files outside of the CAD/CAM package's native format since the data can be transmitted directly to whatever machine tool is currently connected. This concept is similar to how printers work, all word processing software connects to a standard printer interface, and all printers provide that standard interface. The direct machining concept acts in the same manner. All CAD/CAM packages connect to a standard software interface, and all machine tools accept that interface through the DMAC controller.

The speed and ease of use provided by the direct machining concept allows machine tools to be used as rapid prototyping machines. Many conventional rapid prototyping machines are limited by what materials they can use, what size of part they can produce, and their high cost. Directly controlled machine tools have a much greater range of sizes, can shape more material types, and cost significantly less [6].

A proof of concept for this system was implemented at GM's design center in Warren, MI to test its compatibility with Alias|AutoStudio surface development software. The method and results of this proof-of-concept are described in the remainder of this article.

2.3 Direct Machining and Control System

Prior to this project, research work has [2], [3], [5], [12] connected the DMAC controller directly with off-the-shelf CAD/CAM software. The general architecture to connect the DMAC controller with a commercial CAD/CAM package is shown in Fig. 1. The idea of this general architecture is to take full advantage of the 3D modeling and tool path planning capabilities of CAD/CAM packages and to utilize DMAC open-architecture controller to process the derived NURBS tool path directly. For example, Unigraphics has advanced 3D modeling and tool path planning capabilities. Any process plan developed in UG can be processed directly by the DMAC controller.

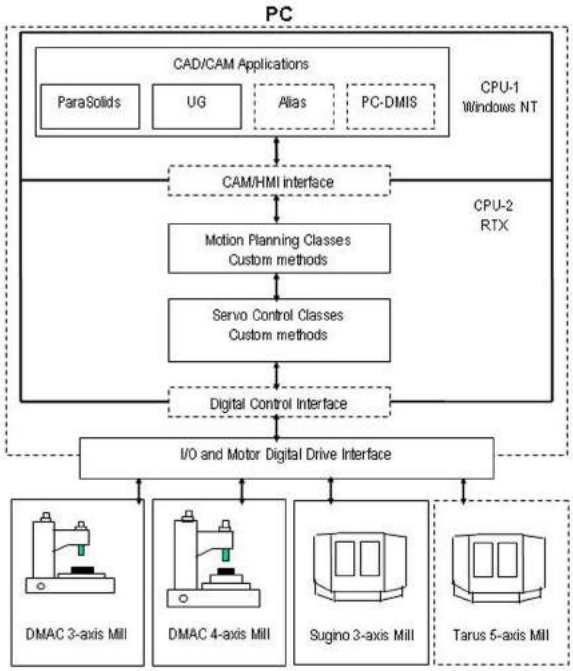


Fig. 1. Illustration of the flexible DMAC architecture.

This paper addresses the beginning of a project aimed at implementing generative tool path planning and direct machining capabilities within a 3D sculpting package, such as Alias|AutoStudio.

3. PROPOSED ALIAS/DMAC ARCHITECTURE

A diagram of the control structure used for a three-axis rapid prototyping CNC mill is shown in Fig. 2. The hardware and software systems are designed in a hierarchical and modular form. This design strategy makes the whole system highly flexible and easy to upgrade.

The hardware system consists of a PC with dual Pentium processors. The non real-time Windows CAD/CAM application runs on one processor and all real-time applications, such as motion control and servo loop control, run on the second processor.

Sitting between the software based DMAC controller and the physical digital drives and I/O is a software layer called the Digital Control Interface (DCI). Evans et al. discuss the DCI [5]. Currently an ISA-bus card that connects to proprietary fiber optic lines forms a network that is used to communicate between the software controller and the digital drives. A commercial version uses an IEEE 1394 network. The modular structure of the system makes it easy to replace the existing ISA card and fiber optic lines with any new PC communication cards and protocols, such as PCI based fiber-optics, IEEE Fire wire 1394, or USB2.

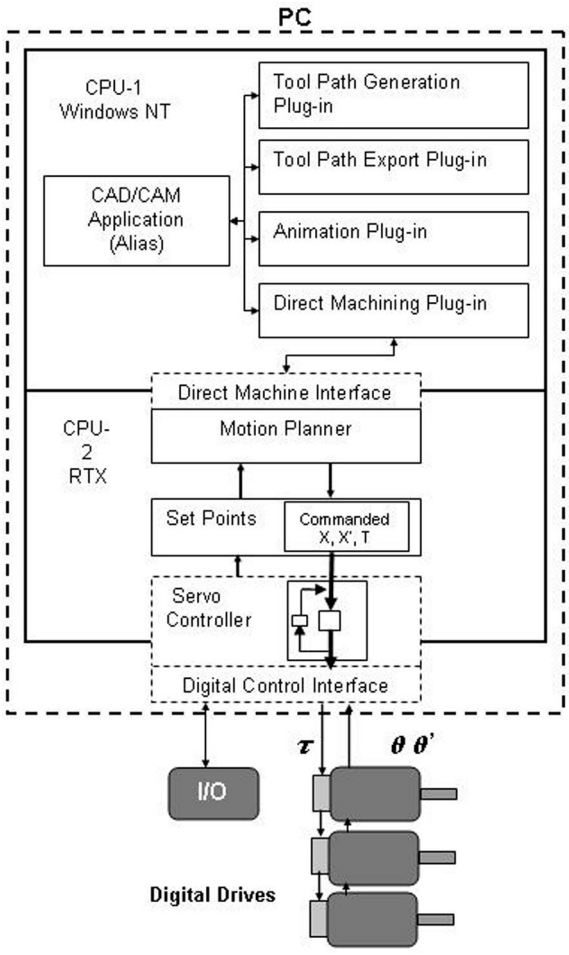


Fig. 2. Overall architecture.

Each distributed networked drive consists of a digital motor interface, an amplifier, and a motor. Each drive is connected to the ISA-bus card through two fiber optics lines. The digital motor interface receives torque set points from the software controller through one fiber optic line. It converts the torque set point into an analog signal, which is then amplified to drive the motor. Actual torque, position, velocity, and acceleration signals are sensed and put into digital form to be relayed back to the software controller through the other fiber optic line.

All of the control software is written in object oriented C++ code. The software system is divided into three layers - CAD/CAM application, Motion Planner, and Servo Controller.

The user interface or top layer of our DMAC system is a customized commercial CAD/CAM application. In this example extensions are made to Alias, a popular system used by aesthetic designers, though the authors have also successfully implemented DMAC in Unigraphics and ParaSolids. All customized milling functionalities were written as plug-ins to Alias. As shown in Fig. 2, the plug-ins consisted primarily of Tool Path Generation functionality, Tool Path Export, Animation, and Direct Machining. Since each Plug-in is a separate software module, the software developers or the end users can replace any of the Plug-ins without changing other software modules. All modifications of Alias were planned and implemented so that it would run independently, on a single Pentium processor.

With tool path data correctly prepared in Alias, by the Direct Machining Plug-in, the real-time system (or second processor) gains access via the shared memory queue of the Direct Machine Interface layer. The shared memory queue acts as a bi-directional link between the two CPUs. It also allows a near real-time simulation of the actual milling process back on the Windows (or Alias) CPU.

Motion commands and settings (coolant on/off, feed rate, spindle speed, etc.) are generated in Direct Machining Plug-in and are packed into motion or I/O data. This data will then be passed to the Motion Planner through Direct Machine Interface. Red et al. discuss the DMAC Motion Planner architecture [12]. The Motion Planner is composed of a trajectory generator and a kinematics object. An adaptive optimal trajectory generator [11] is used in our architecture to generate position, speed, acceleration, and jerk values for each trajectory step, based on a distance parameter (such as total distance along a path). All joints position, speed, and acceleration can be found by calling an inverse kinematics routine. To drive a motor, each joint position, speed, and acceleration value must first be mapped into actuator space and then converted into a torque set point. This torque set point is fed into the Servo Controller.

The Servo Controller [5] receives the torque set point from the Motion Planner and performs the servo control functions for each actuator in the system. Currently, a Proportional-Integral-Derivative (PID) control law is implemented on our Servo Controller. Because of the flexibility and modularity of our system architecture, any new servo control laws can be easily implemented to replace the existing PID control.

4. TOOL PATH GENERATION

Many possible methods exist for implementing tool path generation within Alias|AutoStudio. However, determining which method would be most suitable in a particular business case is beyond the scope of this project. It was determined that basic tool path generation capabilities would be implemented using custom code, developed within an Alias plug-in specifically for this project.

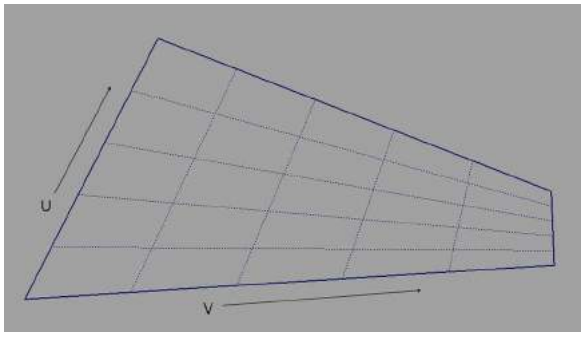


Fig. 3. Light lines indicate iso-parms. Cartesian spacing is much greater on left than on right.

Two routes exist for developing the custom code within Alias. The first option is to utilize as many of the available Alias curve and surface tools as possible, followed by customization within the Alias API to add tool planning and generation. The other option is to pass the Alias surface description to a separate (existing) CAM algorithm that performs the necessary tool path planning and generation, which in turn is displayed within Alias. This project was specific to Alias, so we chose the first option to take advantage of whatever surface tools Alias could offer to help speed development times. The authors are presently investigating the most appropriate third-party CAM tool to integrate.

4.1 Surface Contact Curve Generation

Two techniques for tool path generation were implemented: ISO-Curve Cut and Planar Cut. The ISO-

Curve cutting method develops tool paths by following lines of constant parametric value (iso-parms) along the surface. For example, an ISO-curve could be created on a parametric surface defined in U and V by holding U constant at some value, while allowing V to vary from its minimum to maximum value (see Fig. 3).

A surface S(u, v) in Alias is defined in NURBS form by Eqn. (1).

$$X(u,v)=\frac{\sum_{i=0}^{n}\sum_{j=0}^{m}w_{ij}X_{ij}B_i^n(u)B_j^m(v)}{\sum_{i=0}^{n}\sum_{j=0}^{m}w_{ij}B_i^n(u)B_j^m(v)}$$

$$Y(u,v)=\frac{\sum_{i=0}^{n}\sum_{j=0}^{m}w_{ij}Y_{ij}B_i^n(u)B_j^m(v)}{\sum_{i=0}^{n}\sum_{j=0}^{m}w_{ij}B_i^n(u)B_j^m(v)}$$

$$Z(u,v)=\frac{\sum_{i=0}^{n}\sum_{j=0}^{m}w_{ij}Z_{ij}B_i^n(u)B_j^m(v)}{\sum_{i=0}^{n}\sum_{j=0}^{m}w_{ij}B_i^n(u)B_j^m(v)} \quad (1)$$

$$B_i^n(u)=\binom{n}{i}(1-u)^{n-i}u^i$$

$$B_j^m(v)=\binom{m}{j}(1-v)^{m-j}v^j$$

As a simple method of tool path generation, iso-curves can be created by specifying the number of lines desired across the surface, and the parameter direction in which to travel (e.g. U or V). This technique suffers from over- or under-machining on surfaces where the correlation between parametric value and Cartesian position is not constant. Fig. 3 illustrates a condition where the Cartesian spacing of lines defined by constant parametric values is too large on one end, and too small on another.

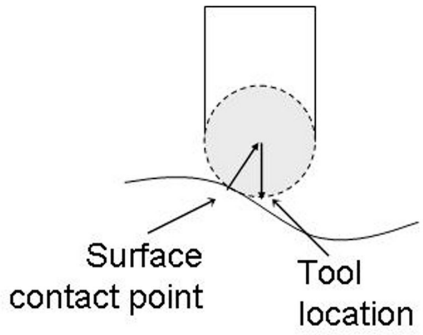


Fig. 4. Relationship of tool tip to surface contact point.

Another tool path generation technique, called the Planar Cut method, avoids this problem by drawing planes through the surface and using the intersection of each plane to calculate a tool path. This method allows the user to specify a constant step distance between consecutive tool paths, but it suffers from scallop heights that increase as the surface normal nears the perpendicular of the tool axis.

4.2 Offset Tool Paths

Tool paths are defined as the locations of the tool tip as the cutter sweeps across the surface. Eqn. (2) is used to calculate the tool tip location of a ball end mill making point contact with the design surface (see Fig. 4):

$$\vec{p}_t = \vec{p}_s + r_t\hat{n}_s - r_t\hat{a}_t \quad (2)$$

where

$\vec{p}_t$ = position of the ball end tool tip,

$\vec{p}_s$ = position of point contact on the surface,

r_t = radius of tool,

$\hat{n}_s$ = surface unit normal at point $\vec{p}_s$,

$\hat{a}_t$ = tool axis unit normal

The entire tool path generation process can be completed by applying Eqn. (2) in three main steps:

1. Create curves on the surface.
2. Offset these curves by a distance equal to the tool radius along the surface normals at these curves locations.
3. Move the offset curves down the tool axis by a distance equal to the tool radius.

The first step for creating curves on a surface is using one of the two methods described in above. These curves define the path of contact that the tool will follow along the surface.

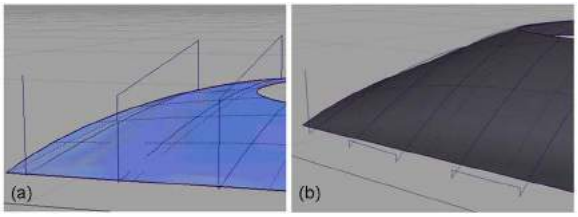


Fig. 5. Tool path connecting types: (a) Box cut, (b) Lace cut.

Once the curves on a surface are created, they must be offset from the surface by a distance equal to the tool radius along the normal to the surface. The normal is a function of position along the surface, so the offset curve must be created by fitting a new NURBS through a series of offset points. The process of offsetting is not exact and suffers a slight loss of accuracy. Determining the optimal

number of interpolation points to use is a separate issue and is not discussed in this article.

These steps will create a series of tool paths and store them within Alias|AutoStudio. The tool paths must be connected to form a continuous series that describe the entire tool operation. This is done by adding a retract move, up from the end of one path, over the start of the next, and then down to begin the next cut. The connected paths can be switched to create lace style tool paths, or box style tool paths (see Fig. 5).

4.3 History

A feature within Alias|AutoStudio called "history" allows users to maintain associativity between an object and its construction elements. This way, if one or more of an object's construction elements are modified, it will update to match the new geometry. Alias|AutoStudio also makes this capability available to plug-in developers. Since our group of tool paths is clearly associated with a surface, the ability to use construction history for the generated tool paths was used for this project. As a result, whenever a surface that had been used to generate tool paths was modified in any way, the associated tool paths were regenerated.

4.4 Layers and Organizing

The completed tool paths are added to the DAG (Directed Acrylic Graph) under a single group node. A DAG node is simply an object within the structured hierarchy defined with Alias|AutoStudio. This allowed all the paths to be handled as a single object in the user interface. They are also created on a new layer that is separate from the rest of the model. This way the user can easily hide the tool paths if they are causing undue visual clutter.

5. ANIMATION

To make the machining environment user friendly, capabilities for checking the validity of a machining sequence was required. An important step toward this capability is animating the generated tool paths.

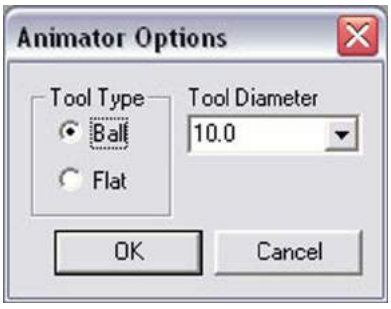


Fig. 7. Dialog used to specify tool properties

Since this project focuses on keeping the master part in its native format, it was appropriate to perform the tool path animation within the native program as well. Fortunately, Alias|AutoStudio contains advanced animation capabilities.

The first step toward animation of the tool path is to create geometry for the actual tool (Fig. 6). The user is asked to enter the tool size, and specify whether the tool is flat or ball ended (Fig. 7). The new geometry is then stored in a DAG node.

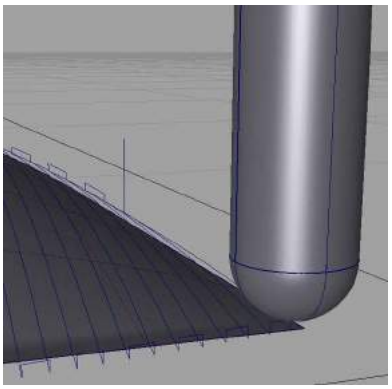


Fig. 6. Animated ball end mill.

In Alias|AutoStudio, animation is accomplished by specifying the location of a DAG node in reference to time. Time is defined by specifying the number of image frames to play back per second. Each frame that contains position information for a particular DAG node is called a key frame. Animation is displayed by defining key frames for position information, such as start, end, and a few intermediate positions along a path. Alias|AutoStudio interpolates the position between each key frame for all other frames. The shape of the parameter value curve as it is interpolated between key frames can be defined by user specified shape characteristics such as step, smooth, linear, etc.

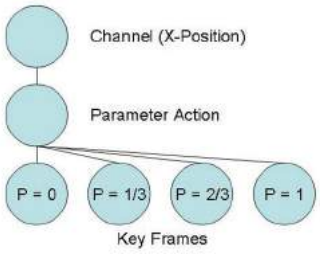


Fig. 8. Key frames define a parameter action. Parameter actions are associated with an animatable parameter such as x-position through a channel.

A series of key frames is called a parameter action, and defines a parameter value as a function of frame number. Each action is not associated with any specific parameter until it is stored in a channel. The channel associates a parameter action with an animation parameter value, such as position (Fig. 8).

For our animation, three parameter actions were defined—one each for the x, y, and z positions of the tool tip. Key frames were defined for all three position parameters by interpolating along each tool path in the selected tool path group. The parameter actions were then stored in channels and associated with the x, y, and z positions of the cutter tool DAG node.

Once these steps are completed, the user is able to view the animation by clicking play on the animation tool bar. The tool then follows the tool paths using the animation engine built into Alias|AutoStudio.

Future improvements of this test code will add the ability to detect collisions and surface gouging. Methods will also be developed to synchronize the graphical playback with the actual machining operation. This will allow the screen to mirror what is actually happening in the machine tool's work area.

With these additions, an operator will be able to completely test an operation for validity before transferring the master model and tool path to an actual machine tool for milling.

6. DIRECT MACHINING

Once tool paths are generated natively in Alias, rather than being passed to the controller as thousands of M&G GOTO points stored in a file, they are sent in their native mathematical form to the DMAC controller for direct machining.

6.1 NURBS-Based Tool Path Encapsulation

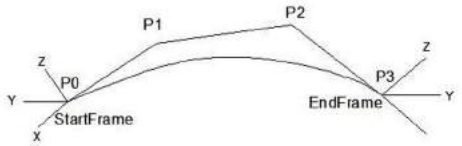
As explained in section 4, a surface S(u, v) in Alias is defined in NURBS form. As a result, tool paths created from these surfaces are also defined in NURBS form. For this project, all the tool paths are extracted from Alias as NURBS and encapsulated into NURBS tool path data structures for direct machining.

Mathematically, a NURBS tool path can be fully defined with the following parameters: NURBS degree, number of knots, number of control points, knot vector, weight and control points. A NURBS tool path is shown in Fig. 9.

The data structure of the NURBS tool path and the real NURBS data extracted from Alias is also shown in Fig. 9.

The displayed NURBS tool path contains a single piece of B-spline curve with the control points P0, P1, P2, and P3 and within the knot interval [0.000 1.000]. For any NURBS tool paths, the NURBS data structure will contain all the necessary parameters for the controller to correctly interpret the derived tool paths. A StartFrame and EndFrame are also defined in the NURBSToolPath data structure. These two frames are used by the DMAC controller to correctly interpolate the NURBS tool paths [16]. For a three-axis mill, as used in this project, the tool will remain in a fixed orientation as it moves through a NURBS tool path. But the DMAC controller has the capability to correctly generate an intermediate orientation frame by interpolating between the start and end frames of the curves as a function of the curve-length. So this research work can easily be modified to fit for a five-axis machining system.

A shared memory queue is created inside the Direct Machine Interface at machine start up. The Alias Direct Machine plug-in will push the NURBS tool paths data structure into this shared memory queue. If the memory queue is full, the Direct Machine plug-in will be blocked until a slot is freed. The DMAC controller will pull these NURBS tool paths data out of the shared memory queue at each trajectory cycle and interpolate the NURBS tool paths correctly with all the necessary parameters extracted from the NURBS data structure.




```
struct NurbsToolPath
{
   int Nurbs_Degree;
   int Number_of_knots;
   int Numbe_of_control_points;
   double knotVector[Number_of_knots];
   double CtrlPts[Number_of_control_points];
   double Weight[Number_of_control_points];
   DMACFrame StartFrame;
   DMACFrame EndFrame;
}
```

```
NurbsToolPath
Nurbs Degree 3
Number of knots 6
Number of control points 4
KnotVector
0.000000
0.000000
0.000000
1.000000
1.000000
1.000000
CtrlPts
0.000000 0.000000 0.000000
0.512325 0.412654 0.157250
0.925820 0.612500 0.201250
1.256000 0.287500 0.250000
StartFrame
0.8700 -0.4300 0.2500 0.0000
0.5000 0.75600 -0.430 0.0000
0.0000 0.50000 0.8700 0.0000
0.0000 0.00000 0.0000 1.0000
EndFrame
0.9330 0.06700 0.3540 1.1200
0.0670 0.93300 -0.354 0.2875
-0.354 0.35400 0.8667 0.2500
0.0000 0.00000 0.0000 1.0000
```

Fig. 9. NURBS-based tool path description.

The ability to pass tool path information directly to the controller from the master part file is an advantageous feature of direct machining that is not found in any file based process description such as STEP-NC. Direct

machining not only eliminates post-processing and its related errors, is also enables the master part paradigm by forcing tool path changes and revisions to be saved within the driving part file. Since all kinematics are handled within the controller, and the interface to the DMAC controller is the same regardless of which machine is being controlled, it is possible to run one process on various machines without necessitating process plan modifications. Direct machining takes advantage of modern computing power to simplify every stage of the machining process.

6.2 Direct Machining Execution

The execution of the direct machining application is handled by customized user interfaces embedded inside Alias. Once the tool paths are generated, the native Alias tool paths can be passed down to DMAC controller for direct machining. All the direct machining applications are activated in Direct Machining Plug-in.

The Direct Machining application software is organized so that a main function will call different sub-functions, which will set all necessary parameters for tool paths data transmission and direct machining execution.

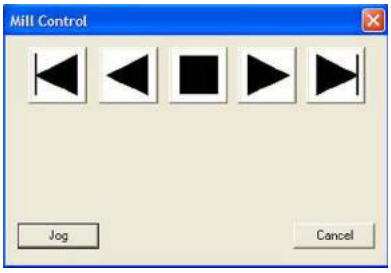


Fig. 10. Direct machining dialog box.

Since the interface between the users and the machine tool is the CAD/CAM system—in this case Alias—some customized dialog boxes must be created in Alias to provide interfaces for the users to control the CNC machine.

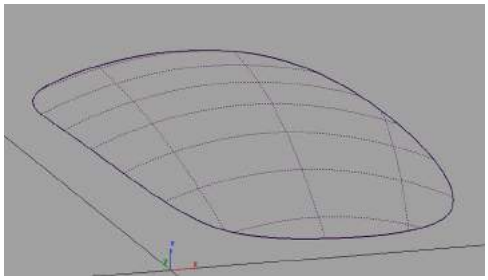


Fig. 11. Example surface used to simulate production data.

Fig. 10 shows the dialog box that is activated from a customized direct machining menu, which is created and inserted inside Alias GUI. This dialog box is the manager of the direct machining application. All dialog boxes that are used to set up the CNC machine and to execute the direct machining commands are launched from this dialog box.

To properly mill a part directly out of Alias, the cutter tool first has to be moved to the position where X, Y, and Z are set to zero in the reference frame defined by Alias. Fig. 12 shows a jog dialog box, where the users may jog the cutter tool to any positions in relation to the coordinate system of a 3-axis mill.

Fig. 12. Jog dialog box.

The slider provided in this dialog box is used to set the distance (in mm) that the cutter tool will be jogged. If the slider has been set to 5 and the "Jog X+" button is pressed once, the cutter tool will be jogged 5 mm in the positive X direction. If the "To Reference Point" button is pressed, the cutter tool will be moved back to the zero position in the Alias reference frame.

The machining process is executed using the dialog box shown in Fig. 10. The direct machining process can be run under two modes. If the "Continuous" button is pressed, a flow of machining process information is passed to DMAC controller without interruption. If the "Single Step" button is pressed, a single move is sent each time.

7. EXPERIMENTAL RESULTS

An experiment was arranged using a 3D CAD data model of a car headlight, similar to data that would typically be used in production at GM (see Fig. 11). The headlight was used because it consists of only one free-form surface, but still demonstrates a fair amount of curvature and shape. Tool paths for the surface were

generated using the test plug-ins and then sent to a 3-axis table mill (see Fig. 13) that was directly connected, through fiber optic lines, to a computer. The computer runs dual Pentium II processors at 400 MHz. The none real-time Alias application runs on one processor and all real-time applications run on the second processor. The headlight surface was machined directly out of Alias as shown in Fig. 13. The process was also completed using a conventional Tarus 3-axis mill currently utilized at GM. The resulting decrease in production time does not come from a reduction in actual machining time, but from a decrease in time required for tool path generation and file handling. A comparison of the time line for the direct process and the typical process is shown in Tab. 1.

The proposed rapid prototyping CNC system shows the following advantages over the current CNC process creation method:

1. Native part file contains tool path geometry, so no disassociation between tool paths and part ever occurs.
2. Multiple file type conversions are eliminated reducing production time and geometry tessellation error.
3. Simplified machine tool control places rapid prototyping capability in the hands of design software users.
4. A single 3D CAD model enables design and manufacturing to be seamlessly integrated into a single database—facilitate "art-to-part" process—and gives the designers a significant power to realize their visions on new products concepts.

Method	**Direct CNC**		**Typical Method**	
	Description	*Time*	*Description*	*Time*
Steps	Create Part	n/a	Create Part	n/a
			Convert to IGES	1 min.
			Send to Tool Path operator	5 min.
	Generate Tool Paths	n/a	Generate Tool Paths	n/a
			Convert to M&G code	2 min.
			Send to machine operator	5 min.
			Load file on CNC machine	5 min.
	Run program	n/a	Run program	n/a
Totals	3	0 min.	8	18 min.

Tab. 1. Comparison of process time for direct CNC method and typical method

8. CONCLUSIONS

This paper describes the use of a CNC mill as a rapid prototyping machine when controlled by a DMAC controller being driven directly from Alias|AutoStudio. The experiment shows that the process of connecting the software packages is relatively simple to accomplish. It also demonstrates that significantly more production power can be given to current designers as well as other users of CAD packages such as Alias|AutoStudio.

Several areas of this work will require further research and development before a useable product is truly available. Foremost among these areas is the generation of tool paths. Further research will be needed to decide whether to use a currently available set of software functions to perform the tool path generation, or to develop a new suite of functions that are perhaps generative and easy to use for this implementation. Another area of further development is the connection to the DMAC controller. Ties to the complete machining and animation functionality available from the DMAC controller have not yet been made. While this project is not yet complete, its promise for future capability is very exciting.

Fig. 13. Three-axis direct CNC.

9. FUTURE DIRECTIONS

Several research projects are currently being undertaken at BYU to invent the next generation of machine tool controllers. The goal of these research projects is to develop a new class of algorithms in order to fully implement direct and dynamic mechanism control by a CAD/CAM application. Among these algorithms, curve fitting of dynamic tool path motion and other control data into n-dimensional-curves is now in development, and will allow the true implementation of curvature-matched machining on five-axis mills.

CMM control and in-cycle-inspection is also currently in development. CMM control can lead to automatic error correction and dynamic process plan updating. New methods for dynamically reconfiguring machine tool

controllers based on the changing process requirements are also under development.

Under this new scheme, machine tools are treated like devices connected directly to the controller. Each machine tool has its own device driver that can be loaded at run-time. If the functionalities of a machine tool need to be enhanced or updated, a new device driver can be developed by the machine tool manufacturer and delivered to the end user. Without changing the existing controller, the end user can obtain the enhanced or updated functionalities provided by the machine tool manufacturer.

In summary, although the DMAC paradigm is still under development, it already demonstrates promising and exciting capabilities. With further development and enhancement, DMAC will provide new opportunities to truly realize the seamless design-to-manufacturing integration by removing many hurdles seen in today's machine tool controllers.

10. ACKNOWLEDGEMENTS

Special thanks for work effort, funding, and technical support go to: The Utah State Centers of Excellence, General Motors, Ford Motor Company, Alias, Unigraphics Solutions, Pratt & Whitney, and Direct Controls, Inc.

Note: Patents are pending on this research. For more information please contact the authors.

11. REFERENCES

[1] Austin, S., Jerard, R., Drysdale, R., Comparison of discretization algorithms for NURBS surfaces with application to numerically controlled machining, *Computer-Aided Design*, Vol. 29, 1997, pp 71-83.

[2] Bassett, C. P., Jensen, C. G., Bosley, J. E., Luo, Y., Red, W. E., Evans, M. S., Direct Machining Architectures Using CAD-CAM Generative Methods, *Proceedings of the IASTED International Conference on Control and Applications*, 2000, pp 287-295.

[3] Bassett, C. P., Jensen, C. G., Red, W. E., Evans, M. S., Direct Machining: a New Paradigm for Machining Data Transfer, *Proceedings of DETC2000/DFM-14298: ASME 5th Design for Manufacturing Conference*, September 10-13, Baltimore, Maryland, 2000.

[4] Ding, S., Mannan, M., Poo, A., Yang, D., Han, Z., Adaptive iso-planar tool path generation for machining of free-form surfaces. *Computer-Aided Design*, Vol. 35, 2003, pp 141-153.

[5] Evans, M. S., Red, W. E., Jensen, C. G., McBride, C., Ghimire, G., Open Architecture for Servo Control using a Digital Control Interface, *Proceedings of the IASTED International Conference on Control and Application*, 2000, pp 339-344.

[6] Huang, H., Lin, G., Rapid and flexible prototyping through a dual-robot workcell, *Robotics and Computer Integrated Manufacturing*, Vol. 19, 2003, pp 263-272.

[7] Koren, Y., Pasek, Z. J., Ulsoy, A. G., Benchetrit, U., Real-Time Open Control Architectures and System Performance, *Annals of the CIRP*, Vol. 45, 1996, pp 377-380.

[8] Lartigue, C., Thiebaut, F., Maekawa, T., CNC tool path in terms of B-spline curves, *Computer-Aided Design*, Vol. 33, 2001, pp 307-319.

[9] Newman, S. T., Allen, R. D., Rosso, R. S. U. Jr., CAD/CAM solutions for STEP Compliant CNC Manufacture, *Proceedings of the 1st CIRP(UK) Seminar on Digital Enterprise Technology*, 2002, pp 123-128.

[10] Oldknow, K. D., Yellowley, I., Design, Implementation, and Validation of a System for the Dynamic Reconfiguration of Open Architecture Machine Tool Controls, *International Journal of Machine Tools & Manufacture*, Vol. 41, 2001, pp 795-808.

[11] Red, E., A dynamic optimal trajectory generator for Cartesian Path following. *Robotica*, Vol. 18, 2000 pp 451-458.

[12] Red, E., Evans, M., Jensen, G., Bosley, J., Luo, Y., Architecture for Motion Planning and Trajectory Control of a Direct Machining Application, *Proceedings of the IASTED International Conference on Control and Applications*, 2000, pp 484-489.

[13] Suh, S., Chung, D., Lee, B., Cho, J., Cheon, S., Hong, H., Lee, H., Developing an Integrated STEP-Compliant CNC Prototype, *Journal of Manufacturing Systems*, Vol. 21, 2002, pp 350-362.

[14] Wright, P., Schofield, S., Open Architecture Controllers for Machine Tools, Part 1: Design Principles, *Journal of Manufacturing Science and Engineering*, Vol. 120, 1998, pp 417-424.

[15] Wright, P., Wang, F. C., Open Architecture Controllers for Machine Tools, Part 2: A Real Time Quintic Spline Interpolator, *Journal of Manufacturing Science and Engineering*, Vol. 120, 1998, pp 425-432.

[16] Yang, Z., Red, E., On-line Cartesian trajectory control of mechanisms along complex curves, *Robotica*, Vol. 15, 1997, pp 263-274.

[17] Yoshimoto, F., Harada, T., Yoshimoto, Y., Data fitting with a spline using a real-coded genetic algorithm, *Computer-Aided Design*, Vol. 35, 2003, pp 751-760.