3D Simulation of tool machining

Anna Puig, Lluís Pérez-Vidal and Dani Tost

Software Department.
Departamento de Lenguajes y Sistemas Informáticos
Universitat Politècnica de Catalunya

Abstract

This paper describes a three-dimensional tool machining simulation system. The initial tool and the grinding wheels are integrated with the machine-tool. The application reads and interprets the CNC program code that controls the machine, it computes the positions and the motion of components and it translates the sequence of machining operations into boolean operations. The machining is computed for 2D sections and, later, a 3D model of the tool is reconstructed. The application is ready to yield tool visualization, it gives measurements on sections and it can show an interactive animation of the whole process. A novel aspect of the simulation is that it is able to deal with 6-axes machines, whereas most of previous work is limited to 3 and 4 axes machines. In addition, it allows to interrupt the machining process and to show partially machined tools. A major contribution is the fact that the boolean operations are performed in 2D and the 3D model is reconstructed from the cross sections, which provides user control on the resolution of the operations at a low cost.

Key words:
Application packages, Industrial Machining, Simulation, Boolean operations, Graphics data structures and data types

1 Introduction

In mechanical industry CNC (numerical control) machines have been in use for more than twenty years. Metal-cutting machines move according to the instructions of programs that command step-by-step linear or rotating servomotors and they yield better precision than that obtained with the old manual analogical machines.

Email address: `anna@lsi.upc.es, lpv@lsi.upc.es, dani@lsi.upc.es` (Anna Puig, Lluís Pérez-Vidal and Dani Tost).

There is a wide variety of literature on CNC in CAD, mainly centered on the automatic computation of tool paths, [7]. An integrated CAD/CAM system starts from a final tool design and yields the optimal trajectories of the tool and the grinding wheel, giving the CNC code as a final result. Machining simulation and verification wants to do exactly the opposite: to calculate the tool starting from the path with a triple goal, [3]: (i) To detect possible collisions between the tool or any of the grinding wheels and the rest of the machine tool. (ii) To check visually the efficiency of the trajectories; (iii) To verify that the surface of the resulting tool is effectively the desired surface. Using a simulation system decreases considerably the tool production cost because it avoids the trial and error process on the real machine with costly materials that is otherwise necessary.

Conventional CAD systems don't offer the function of simulating the mechanization of tools, for which specific applications are needed. This paper deals with a particular type of machine tools: bore and cutter grinders. Most of the existing simulation aplications for these types of machines deal only with the machinning of 2D cross-sections of the tools and they are restricted to the main fluting operation [1]. Three-dimensional applications are rather recent [2] [13]. They provide a machining simulation for specific 5-axes machines and they are not applicable to general movements. The goal of this paper is to describe a novel application able to deal with any combination of movements up to 6-axes. A major feature of the application is that, by opposition to the existing systems that construct one final tool, it is able to construct intermediate tools corresponding to the different steps of the machining. This feature provides better user control on the machining process and it facilitates updating locally the trajectories. In order to provide this flexibility, the system is based on a novel approach of the boolean operations based on a double discretization of four dimensional space (3D+ time). This algorithm is fast and it provides user-control on simulation accuracy.

The paper is structured as follows. First, the requirements of the application are discussed and a general structure for the simulation process is proposed, that fulfills these requiremts. In Section 3, each step of the process is described, previous to the definition in Section 4 of the geometric model for the workshop objects. The results of the simulation are discussed in Section 5.

2 General structure of the application

The workshop is based on machine tools which have up to 6 degrees of freedom: four in translation and two in rotation. It handles the motion of the bit to be machined, a tubular object made of sintered steel, called "the tool", and the motion of the set of grinding wheels which will bite on it. The tool is placed on a spindle (the tool-holder), that allows translation on three axes (X, Y and U) and rotation on two axes (W and A). The wheel spindle is another moving capstan that allows motion of the grinding wheels set as a solid unit along a translation axis (Z). After several movements of the two capstans, the initial tool is transformed into a bore.

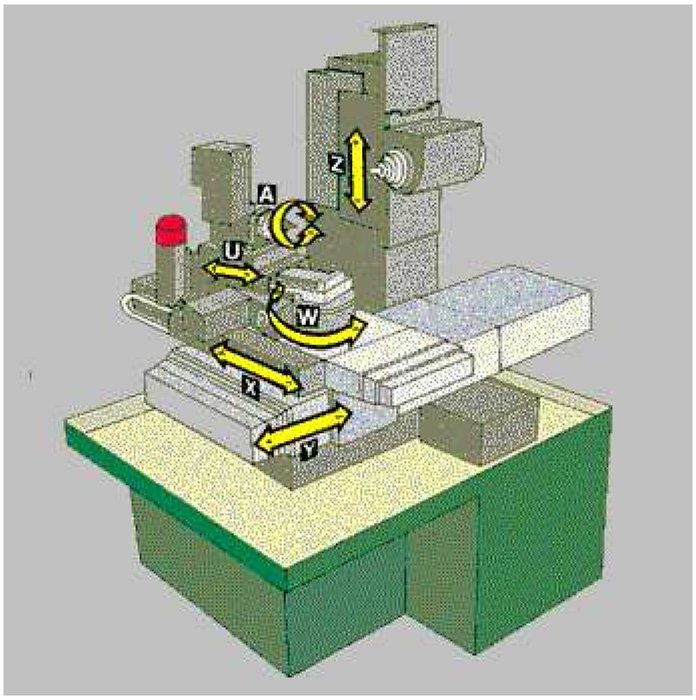

Fig. 1. 6-axes Machine tool.

The simulation of the workshop is performed in several stages, from the input data gathering to the final model construction and the interaction with the results of that simulation. Those stages have been defined from the analysis of input data and the specific requirements of the application. Input data are:

- The base tool, of a cylindrical or conical shape, or even a combined shape with cylindrical and conical slices joined by radial and linear segments.
- The set of grinding wheels: defined initially on the Y axis. Each one of the wheels is basically a trunk of a cone, although other shapes can be dealt with.
- The sequence of numerically-controlled operations. They are a set of instructions that are directly interpreted and executed by the machine-tool. The syntax of this language (CNC) is fairly standardized in Numerical Control applications [6].

On the other hand, the specific requirements of the machining application can be summarized as follows:

- Efficient and exact simulation of tool machining.
- Interactive visualization of the sequence of tools generated in the succession of different machining operations.
- Partial and total animation of the machining process.
- Queries on distances and angles on the planes for different cuts of the tool.

Simulating a practical machining process means, first of all, translating efficiently the sequence of numerical control operations into movements. The CNC code contains many useless (for this matter) operations: switching on and off lubricating oil, and such. Most important, the numerical values of angles and lengths are not explicit, and are built with parameterized expressions. We need a minimal subset of those instructions that can be treated in a homogeneous and fast way. And this without changing neither the initial syntax of the control code nor the semantics of the machining operations. This entails the need of a filtering and simplification subprocess, that must retain the equivalence of geometric operations between the tool and the grinding wheels of the resulting minimal subset and the initial instruction set.

Also related to an efficient machining operation, we need to define the number of instructions that must be simultaneously processed during machining and the number of tools to be generated along the simulation process. The tools so generated will be the only candidates for visualization and query.

Finally, the fundamental points to optimize in order to ensure the efficiency and accuracy of the simulation process are the strategy used for boolean operations and the definition of the representation model for both tools and grinding wheels.

Interactive visualizations of the tools are based on a representation model which can be efficiently queried for the geometry of the surface of objects. The complexity of these surfaces is so high that, not only must we optimize the model, but also adopt optimization strategies for the visualization using the resources offered by state-of-the-art graphics cards.

To produce partial or total animations of the simulation process, a list of motions of all the solids in the workshop must be stored and the ability to query and visualize at any time the corresponding positions and geometry of the objects must be provided.

The steps in the process are illustrated in Figure 2.

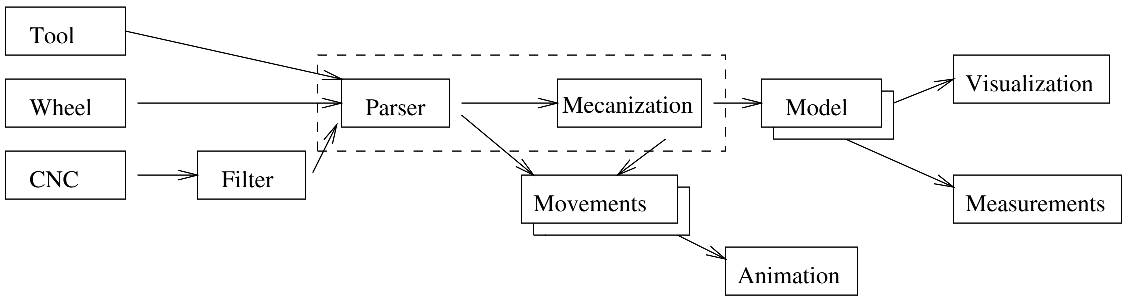

Fig. 2. Architecture of the industrial tools simulation application.

3 Steps

3.1 Filter

The filtering module receives as input a tool machining program written following the CNC rules and regulations, and produces a sequence of lines that simplifies maximally the input sequence. The input CNC sequence is divided into blocks, that correspond to each one of the logical machining operations (fluting, gashing, etc.). In each operation a single wheel and the tool are involved. Each instrucion can be either a command controlling parameters of the machine, (rotation speed, rebalancing, etc.) or the coding of a translating or rotating motion, relative to the previous motion or absolute, according to the initial positions of the tool and the wheel. Instructions are grouped into sequential or iterative compositions and movements are specified in terms of register (variable) values.

The output sequence contains only movement instruction lines with numerical values, code lines indicating if the movement values are relative or absolute and control lines indicating that a set of instructions should be repeated a given number of times applying to the tool an axis rotation. These later instructions are very useful to replicate the result of boolean operations for the different flutes of the tools.

Therefore, the filter consists mainly of removing from the sequence all the registers, subtituting them by actual values, and supressing the loops and alternative structures. Removing registers is feasible because in CNC, registers do not act as input variables but play the role of parametering design values. In order to remove them, they must be evaluated, which implies the treatment of parenthesis (square brackets, really) nested to any depth in the arithmetic expressions. The removal of loops is a recursive procedure, since loops can be nested one inside another. The treatment of a simple loop must handle the following cases:

- Loops that do not contain any machining operation and are used only to perform computations on the contents of registers. In this case, the computations will be done and, no new lines will be added on the final result.
- Loops that repeat exactly the same operations, because the real machine needs several passes to actually remove all the material. Those ones are also suppressed.
- Loops corresponding to several flutes. The set of machining operations that are the same are grouped into a block, the number of loops is counted and the angle between flutes is computed, and these two numbers are written at the beginning of the block in the output sequence.
- Complex loops that depend one from the previous one. Those ones are evaluated and written as a full list of instructions. resulting sequence.

After considering the posibility of implementing this part of the application in C, or in C++, the Perl language has beeen chosen. The biggest advantage of Perl, in this case, is that it lends itself very well to the treatment of lists of CNC command lines. Besides, it caters for the treatment of regular expressions, together with recursive calls to procedures. Finally, a Perl program can be easily embedded in a C program.

3.2 Compilation

At the compilation stage, all the tool and the grinding wheel set motions are processed to build a list of motions for the animation process. This structure lets the program determine, at any step, the tool undergoing machining, its position and the orientation and position of the grinding wheels. The spindle moves together with the grinding wheel set, and the tool support and the collet move together with the tool. The calls to machining that will compute the new tool are made at this stage. The compilation module has to define the blocks to group the numerical code instructions that will give way to a new tool. The overall efficiency of the process and the number of intermediate tools that will be candidates for visualization and query will depend upon this grouping into blocks. Several alternatives have to be taken into account. At one end, we can group all the motions of the process into a single block. This means, that the machining process would perform all checks of possible interferences between the objects at each boolean operation and the creation of a single tool at the end of the process and, therefore, only the first and last tools could be visualized. Sometimes the validity of a single machining operation has to be checked, and this would not be possible with this option.

A second strategy consists of defining as a machining block the whole set of instructions of a machining operation (all those of gashing, for instance). The advantage of this strategy is that one tool per operation is built and this can be independently queried and visualized, improving the realism of the simulation. A possible refinement of this strategy would be the definition as a block of the group of instructions of one machining operation from the beginning of a collision to the end of the collision, that is, from the moment the grinding wheels touches the tool to the moment that contact is lost. Thus the computational cost of the machining process is reduced, avoiding boolean operations where there is no real substraction of materials. This means that an approximate control of possible interferences has to be implemented inside the compilation process and that the tool has to be generated when a solid with a consistent machining is obtained.

There are several defined cases where that control is not enough. In some machining operations, like *flute*, a grinding wheel machines the tool by translating and rotating the tool, and returns to the initial position with the opposite movements. In this case, the procedure will detect interferences both in the real machining motions and in the oposite (“reverse”) motions. Consequently, the end of a block is determined by a motion that does not produce interference and also by the beginning of a reverse motion- This is the alternative actually implemented with the aim of performing efficient boolean operations without decreasing the number and quality of possible queries and visualizations in the middle of the machining process. Finally, the last situation that may determine the end of a block is the end of an iteration block.

3.3 Machining

Machining strategies.
Machining is the operation that creates a new tool resulting from the grinding of an initial tool with a grinding wheel with a sequence of operations. It is a dynamic boolean operation of difference between the grinding wheel and the tool along time. The most referenced numerical control simulation method is the so called Vector Cut [4], [9]. It is an approximate solution that represents the frontier as a set of points and normal vectors that will be cut along the path of the grinding wheel. This method is effective for the simulation of sculptured surface polishing, but it is not extensible to complex motions of the tool and/or the grinding wheels. It is mainly useful to detect mistakes in the path that mean a cut of the vectors too big or too small, but, apart from the extension of [8], it does not yield directly a model of the bit to machine.

Another option consists of discretizing the time dimension and implement a sequence of boolean static operations between the solids at every time slice. This operation shows its biggest drawback in its very high computational cost. According to [6], this is proportional to the number of discrete positions to the fourth. This puts it out of question, in practical terms. Another problem it shows is the granularity of the temporal discretization: it must be very fine if precision in the final tool is required. This means that very little material is cut off in each boolean operation, and that may entail robustness problems in the computations. A possible method to avoid both problems is to discretize the initial tool model into a voxel or an octree model, [11], to perform all the succession of boolean operations on the discrete model and then reconstruct the machined surface, at the end. Like this, we benefit from the fact that the cost of discrete boolean operations is much lower and the reconstruction phase at the end of the process is done as late as possible. This option requires the sequence of movements to be specified in terms of relative motion of the grinding wheel, while the tool and its discretization remain fixed. This prerequisite is not always valid and, in particular, does not hold for the general case of 6-axes machines.

The third option taken into account is that of the computation of the volume swept by the tool and the grinding wheel in their motions to performor a single boolean difference operation between the two volumes. The main difficulty of this option is the computation of swept volumes. There are several references [5], [10] on this subject, that contain methods generally applied in CAD for extrusions, collision detection, and other problems but none of them can be applied to the non-trivial case of simultaneous motion of the two solids in play.

2D sections machining

The novel proposal we develop here is sketched in figure 3. It consists of discretizing the tool in axial sections, performing the machining operations on these sections and finally reconstructing the surface of the tool stitching the sections. This solution has the main advantage of reducing the computational complexity and thus the computational cost of boolean operations from 3D to 2D and, as the second option examined [11], to postpone the reconstruction to the end of the sequence. It also allows for the adaptive adjustment of both the resolution of the machining in one single section and the thickness between successive cuts. It also enables the fulfillment of the requirements of the query of axial cuts described in Section 2. The main difficulty of this operation is the choice of the resolution for the reconstruction phase. However, we will show in Section 4 that satisfactory solutions can be given to this problem, and it has been selected because it seems the best compromise overall.

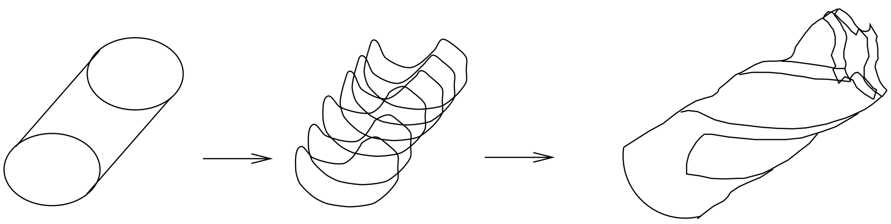

Fig. 3. Machining process scheme.

2D sections machining is based on a triple discretization: the tool into 2D sections, the grinding wheel into infinitesimal thickness discs, and the motion in successive positions. Thus, the general geometric problem is reduced to that of the intersection between two discs. The algorithm simulates the path of a circular section of the tool and the disc of the grinding wheel. For every instant, it computes the segment of intersection between those two and it, applies to the preceding segments the axial rotation motion of the tool. At the end of the path, the ends of the segments inside the section define a polygonal approximation of the dynamic boolean operation. This process is repeated for each differential section of the wheel. Then, the envolvent of the intersection polygonals is computed, and a new profile of the section is computed. And finally, if the section was already machined, the new profile is cut over the initial profile.

This process is optimized with several acceleration techniques:

- The intervals of probability of intersection between discs are computed to reduce the number of tests between discs.

- The intersection segments are computed incrementally using temporal coherence of motion.
- Finally, in operations like “flute”, in which the shape of the section is the same for a good stretch of the tool, a profile of cut in one maximum diameter section is first computed, next replicated applying an axial rotation and clipped against the actual cross sections.

4 Geometric model

The tools have a tubular shape which makes them suitable to be represented with a generalized cylinder based model. Each geometric model is defined as a set of planar contour curves centered in a sweeping axis. The contour curves are originally circular but they may acquire an irregular shape after machinning. Evenmore, during machinning a contour can be broken into different connected contours, called shells. The representation of circular contours is parametric. They are stored at points where the radius derivative changes. Irregular shaped cross sections are sampled at the highest resolution during machinning, so that no interpolation between explicit curves should be done during rendering. These sections are represented explicitly as sets of labeled shells composed of labeled connected segments of 2D polar contour points. The segments correspond to the machinning operations that have generated the cut.

During the rendering, the tool surface is tiled into triangles between successive contours. A detailed description of this process can be found in [12]. It handles three tiling configurations: two circular cross-sections, an irregular one with a circular one and two irregular cross-sections. The former case is straightforward. The second case is solved by sticking a circular cross-section to the irregular one. This strategy removes eventual folds of the surface and the union between the two sticked sections is almost unnoticeable. Finally, the tiling between irregular adjacent contours is based on the establishment of a doble correspondence: between shells and within shells, between contour points. The correspondences between shells is based on their label: when a connected shell is cut into pieces during a machining operation, the new shells are tagged with a label that indicates that they were formerly pieces of the initial one. Only correspondences between shells coming from the same previous shell are allowed. The labeled segments of the shells indicate the connectivity between them. At the branching, the contour with less shells is sticked to the other and plannar tiles between branches are build (See Figure 4). The correspondences between vertices of adjacent irregular contours are established taking into account the segment to which they belong. This ensures that, except at the segment extremes, points of a segment can only form triangles with points of the same segment. In addition triangles belonging to different segments can be painted with different colors.

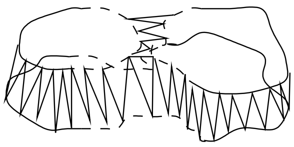

Fig. 4. Construction of a branching.

In order to speed up the visualization, the topological information generated during tiling can be stored in the model by associating to each explicit contour point its list of adjacent vertices in the previous and the next contour along with its averaged normal vector.

This model has low memory requirements, its rendering is fast and, above all, it is well adapted to the tool machinning process. In addition, as the other objects of the workshop , wheels and holders, have also a tubular shape, they are represented with the same model.

4.1 Visualization and animations

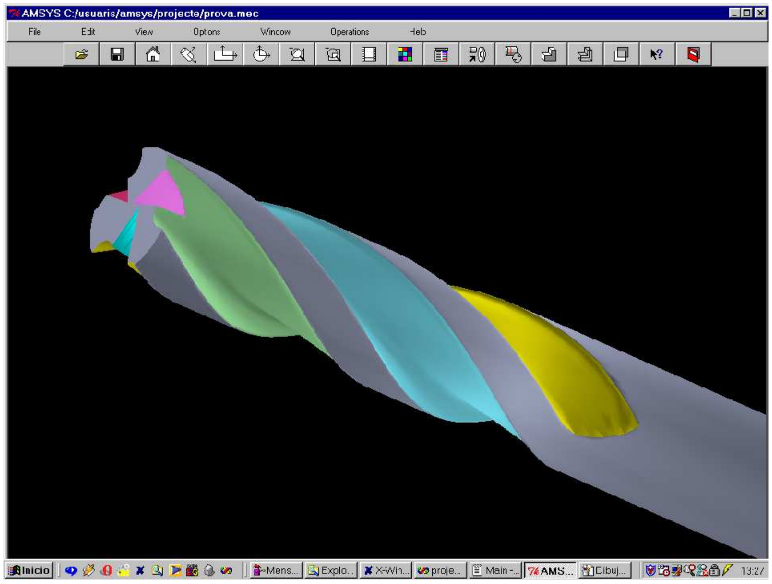

Fig. 5. The tool after a flute and a gashing operations.

The visualization of several parts implies the reconstruction of the boundary of the model, exposed in the following section. It requires information on the geometry and the lighting properties of the tool material along with the eventual colours associated to the different operations (see Figure 5).

OpenGL graphical library primitive calls have been used to exploit the possibilities of the projective Z-Buffer and shading techniques it offers. A blank bit machined with a flute and a gash, consisting of 4724 triangles is visualized in 3-4 seconds, with a Pentium III at 700MHz having a Diamond Viper graphics card. We have used OpenGL display-lists to optimize the visualization delay for the whole workshop. To post 30 frames that include the whole workshop (see figure 6), the process time has been 47 seconds. The time to visualize each frame is 1 second. Although the pre-process of all the display-lists before the visualization proper is very time-consuming, the animation process accelerates a 65 %.

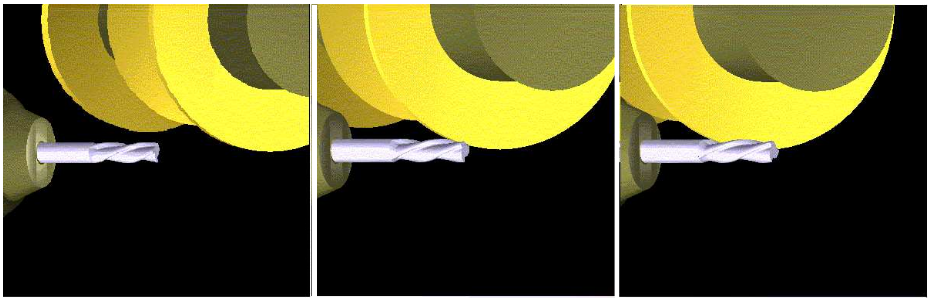

Fig. 6. Several animation frames of a machining process.

The graphical interface of the application is aimed at a user who is a machining expert. It is a single-user and single-process application. It is based on a multi-window control automat with independent and interruptible interactions, implemented is Tcl/Tk and C. Here is an example of the interface where an angle in a section of the machined tool is examined (see Figure 7).

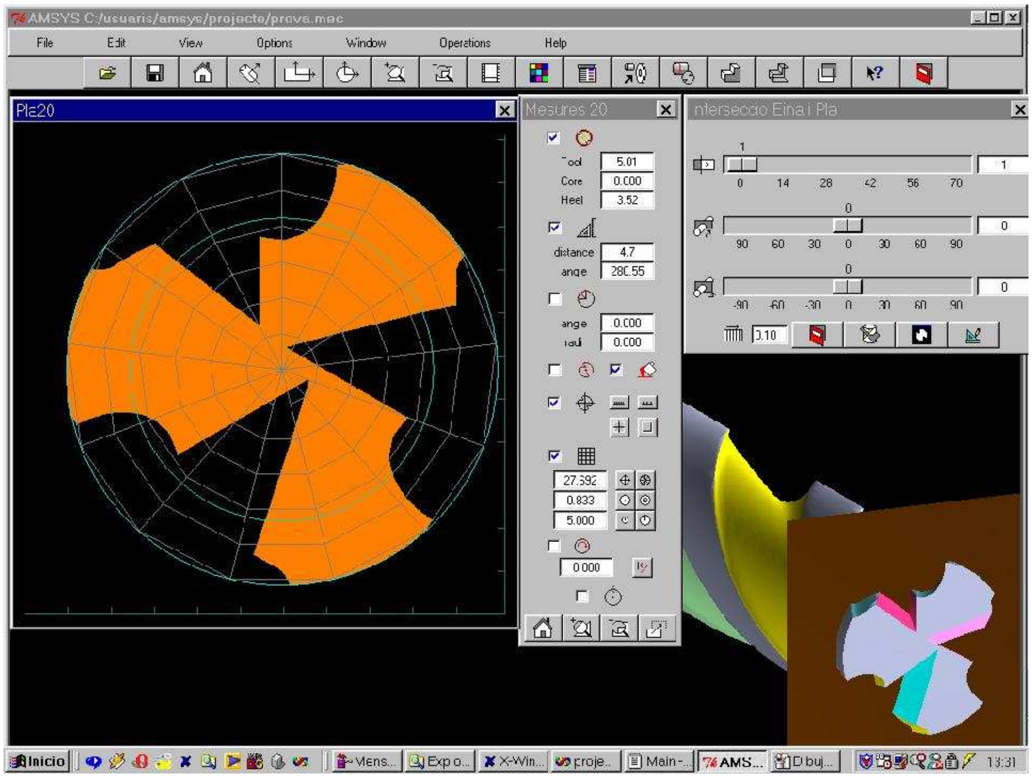

Fig. 7. Measures on an angle in a section of the tool.

5 Conclusions and future work

This paper describes an application that simulates tools machining, a recently developed field of great industrial interest. The main contributions of the package are:

- A flexible and versatile architecture that allows simulation directly from CNC code, through a filtering and compilation preprocess, that minimizes geometric computations.
- A geometric model of the workshop that evolves with time to facilitate real-time animations and high-quality visualizations.
- A novel and efficient method to perform boolean operations.

The package is now in the trials prior to actual installation in an industrial environment. The results obtained so far are satisfactory and validate the design decisions adopted.

Acknowledgements
We gratefully acknowledge the decisive help of Albert Mercade in the implementation of software for the application.

References

[1] Amsys. *WinEMC CNJ2-30*. Advanced Machining Systems, 1998.

[2] ANCA. *Cimulator 3D User Manual*. ANCA, 2000.

[3] Z. Han C.C.H. Yang. Interference detection and optimal tool selection in 3-axis nc machining of free-form surfaces. *Computer-Aided Design*, 31:303–315, 1999.

[4] I. Chappel. The use of vectors to simulte material removed by numerically controlled milling. *Computer-Aided Design*, 15:156–158, 1983.

[5] Abdel-Malek K., Yeh H., and Othman S. Swept volumes: void and boundary identification. *Computer-Aided Design*, 30:1009–1018, 1999.

[6] K. Lee. *Principles of CAD/CAM/CAE Systems*. Addison-Wesley, 1999.

[7] Y. Lin and T.S. Lee. An adaptive tool path generation algorithm for precision surface machining. *Computer-Aided Design*, 31:237–247, 1999.

[8] J. M. O'Connell and A. Jablokow. *Construction of solid models from NC machining programs*. Proceedings of ASME on Manufacturing Science and Engineering, 1993.

[9] Jerard R., Drysdale R., Hauck K., and Schaudt B. Methods for detecting errors in numerically controlled machining of sculptured surfaces. *IEEE Computer Graphics and Applications*, 1:29–39, 1989.

[10] D. Roth, S. Bedi, F. Ismail, and D. Mann. Surface swept by a toroidal cutter during 5-axis machining. *Computer Aided Design*, 33:57–63, 2001.

[11] U. Roy and Y. Xu. Computation of a geometric model of a machined part from its nc machining programs. *Computer Aided Design*, 31:401–411, 1999.

[12] D. Tost and A. Puig. Visualization of labeled segment cross-contour surfaces. *Volum Graphics 2001, New York*, June 2001.

[13] Walter-AG. *Cyber Grinding Software*. Walter AG, Tübingen, 2000.