Boolean operations for 3D simulation of CNC machining

Dani Tost
Anna Puig
Lluís Pérez Vidal

LSI-02-49-R

Boolean operations for 3D Simulation of CNC Machining

Dani Tost and Anna Puig and Lluís Pérez-Vidal
dani@lsi.upc.es, anna@lsi.upc.es, lpv@lsi.upc.es
Software Department.
Polytechnical University of Catalonia

July 4, 2002

Abstract

This paper addresses the simulation of drilling tools CNC machining. It describes a novel approach for the computation of the boundary representation of the machined tools. Machining consists of a sequence of boolean operations of difference between the tool and the grinding wheels through time. The proposed method performs the dynamic boolean operations on cross sections of the tool and it reconstructs the 3D model by tiling between the cross sections. The method is based on classical computational geometry algorithms such as intersection tests, hull computations, 2D boolean operations and surface tiling. This approach is efficient and it provides user control on the resolution of the operations.

CNC simulations, Bores machining, Computational geometry, Boolean operations, Surface tiling.

1 Introduction

Most of the research on CNC in CAD is centered on the automatic computation of tool paths [12] [4]. Given a final tool design, the optimal trajectories of the tool and the grinding wheels must be computed yielding as final result the CNC code. Machining simulation and verification has exactly the opposite goal: to calculate the tool starting from the CNC code and from a geometrical model of the machine, the wheels and the tool before machining. This simulation has three main applications [5]. First, it detects eventual collisions between the tool or any of the grinding wheels and the rest of the machine. It is important to avoid collisions because serious damages to the machines can follow. Next, simulation provides a means of visually verifying the efficiency of the trajectories, which may result in faster and cheaper processes. Finally, the simulation allows users to check if the surface of the resulting tool is effectively the desired one. In the routine practice of machining, experienced operators have enough skills to imagine the tool final shape by only reading the CNC code. However, they are generally not able to do so with new or non-standard designs. Therefore, the use of 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.
This paper addresses a particular type of CNC machining simulation: the grinding of bores and cutters (see Color Plate 1). Conventional CAD systems do not provide a

means of realizing this type of simulations and specific applications are needed. Until recently, most of the simulation applications dealt only with the machining of 2D cross-sections of the tools and they were restricted to the main fluting operation [2]. Three-dimensional applications are rather recent [3] [22]. They provide a machining simulation for specific 5-axes machines and they are not applicable to general movements. This paper presents a novel approach for the computation of the external shape of the tools through a sequence of coordinated movements of the tool and the wheels on machines of up to 6-axes. The proposed method reduces the 3D problem to 2D dynamic boolean operations followed by a surface tiling. The 2D solution involves different techniques of planar computational geometry: from intersections to hull computations.
The paper is structured as follows. In Section 2 we review previous approaches on machining simulations. Section 3 describes briefly the contour conditions of the simulation. Finally, Section 4 describes the computation of boolean operations and the results of the implementation are shown in Section 5.

2 Previous work

Machining can be considered a dynamic boolean operation of difference between the grinding wheel and the tool. It is dynamic, because both the tool and the wheels move along time through rotations and translations.
The Vector Cut [7], [9] is probably the most referenced numerical control simulation method. 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 suggested by the presence of abnormally high or small cut vectors. Besides, except for the extension of [15], it does not yield directly a model of the bit to be machined.
An alternative strategy for machining simulation consists of realizing a sequence of 3D static boolean operations through time. The main drawback of this strategy is its high computational cost. According to [10], 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, [20], to perform all the sequence of boolean operations on the discrete model and then reconstruct the machined surface, at the end. This approach benefits 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, it does not hold for the general case of 6-axes machines.
Finally, another option taken into account is that of the computation of the volume swept by the tool and the grinding wheel in their motions. A geometric representation of this volume would allow performing only one boolean difference operation between the two volumes. The main difficulty of this option is the computation of swept vol-

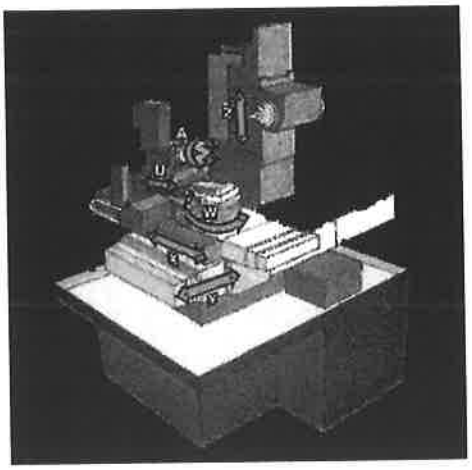

Figure 1: 6-axes machine tool.

umes. There are several references [1], [19] 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.
The strategy proposed herein overcomes the disadvantages of these methods. It consists of a double discretization of four dimensional space (3D+ time) that reduces the general problem to a sequence of 2D Boolean operations and 3D geometric reconstructions. This algorithm is fast and it provides user-control on simulation accuracy.

3 Scene model

There are different types of machine tools for the fabrication of bores. They share the same general structure but they differ in the number of degrees of freedom. The method proposed herein deals with machines up to 6 degrees of freedom. These machines have a static vertical axis (Z in picture 1) on which the grinding wheel set can move up and down. One tool is placed on a spindle (the tool-holder), that may translate on three axes (X, Y and U) and rotate on two axes (W in relation to the wheel axis and A relative to its own axis). At the beginning of the process, a tool has a piecewise cylindrical or conical shape. Its final shape is the result of a sequence of machining operations consisting of simultaneous movements of the tool and the wheels. The wheel shape is also piecewise cylindrical or conical. It remains unchanged during the process.
The machining process is divided into a set of operations, each one with a specific name in CNC jargon. Each operation is performed using a specific wheel. This information is written in the CNC file.
Specifically, the main operations are (in their usual order):

- Fluting: performing the lateral helicoidal o straight grooves
- Gashing: cuts in the tool head
- Outer Diameter Sharpening: edge sharpening of the lateral grooves
- End Face Sharpening: edge sharpening of the tool head cuts

Figure 2: A set of real tool data.

- Notching: direct cut in the tool head.

Figure 2 shows a real bore and it indicates the operations that have gave it shape.
Each operation performs several symmetrical cuts in the tool shape. The tool shown in Figure 2, for instance, has three lateral grooves realized during the "Fluting" operation. Each cut is performed through a sequence of movements. In the CNC code, each movement corresponds to a line instruction specifying the motion axes (X, Y, U, A, or W for the tool and Z for the wheel) along with the amount of rotation or translation to be performed for each edge.

4 Machining simulation

4.1 Overview

Our approach uses the fact that the tools have a tubular shape. It consists of discretizing the tool in axial sections, performing the machining operations on these cross-sections and finally, reconstructing the surface of the tool by tiling between cross- sections. Before machining, the cross-sections are circles. Afterwards, they have a complex shape that may even have been split into separate connected shells at the tool end.
The movements are divided into blocks, each one corresponding to an CNC operation or even to one cut within an operation. The machining process is performed sequentially for each block. Therefore, as many intermediate models are created as instruction blocks exist. The initial tool is taken as input of the first machining process. The resulting tool is used in the second block processing and so on. The surface reconstruction step can be performed on any of these intermediate models or, alternatively only on the last one.
Therefore, the simulation process of each instructions block is composed of two steps:

- A 2D boolean operation process, that receives as input: (i) the tool representation, (ii) the machining wheel representation, (iii) a list of movements and that gives as output a new representation of the tool cross sections.
- A tiling process that completes the tool representation with the triangulation between contours.

The second step, surface tiling, is a classical subject in computer graphics [13]. It consists of two related problems: (i) establishing correspondences between contours

(branching problem) and (ii) searching correspondent vertices to form tiles (correspondence problem). Several solutions have been published to solve both problems based on minimizing the distance between successive contours [6] [16] and interpolating in-between contours [11]. The method used herein in an extension of these algorithms that adds to these criteria the constraint of tiling between segments of the contour corresponding to the same machining operation. This extension is described in depth in [21].

4.2 Machining of the tool cross-sections

The computation of the new shape of tool cross section consists of three steps:

- Computation through time of the intersections of the wheel cross sections and the external contour of the tool section. Both sections are circular and, due to their relative orientation, their intersection is a segment. Therefore, the result of this step is a set of segments.
- Calculation of the hulls of the segments set. These hulls are polygonal approximations of wheel cuts on the tool section.
- Reconstruction of the tool cross section contour given its original shape and the hull curves.

The pseudo-code algorithm below illustrates this process. Let st be the tool cross section at the beginning of the process, wh the wheel and ml the movements list. The wheel is discretized into a set of circular cross sections sw (procedures $FirtSectWheel$ and $NextSectWheel$). The movement of sw and st is decomposed into a a set of successive positions (inner loop). For each position, the intersection between sw and st is computed in the procedure $InterSect$. If there is intersection, then the corresponding segment $segm$ is stored in the segments list $seglist$. Then, the geometry of st, sw and $seglist$ is updated to next positions in the procedure $UpdateGeom$. The position of st is reset at its initial location for each new wheel section. After all the wheel sections have been processed, the hulls of the segment list are computed in $CompHulls$ and then clipped against the initial contour of st with the procedure $Reconstruct$.

procedure $CrossSectionMachining(st : tSection, wh : tWheel, ml : tMovList)$
  var
    $sw : tSection$
    $segm : tSegment$
    $seglist : tSegmentList$
    $hulls : tHullList$
  fvar
  $InitSegList(seglist)$
  $sw := FirstSectWheel(wh)$
  while $ValidSection(sw)$ **do**
      $endofmov := FALSE$
      while $\neg endofmov$ **do**
          $InterSect(st, sw, \&segm, \&status)$
          if $status \rightarrow InsertSegment(segm, seglist)$ **endif**
          $UpdateGeom(ml, \&st, \&sw, \&seglist, \&endofmov)$
      endwhile

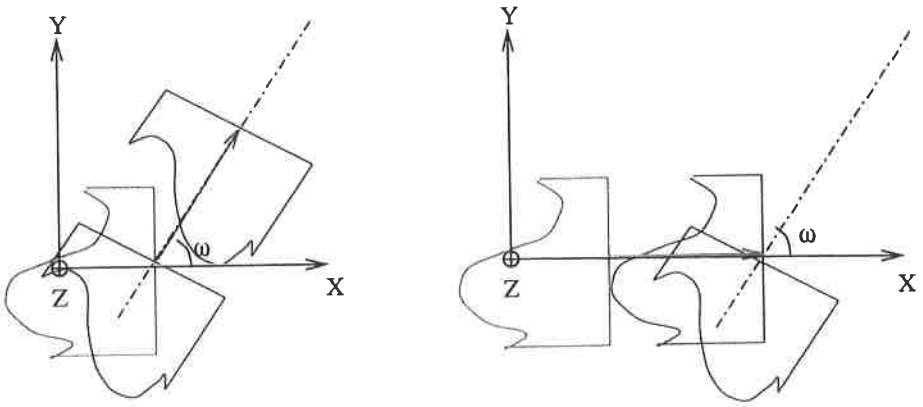

Figure 3: Non equivalent transformations.

$sw := NextSectWheel(wh, sw)$
$ResetToolPosition(\&st)$
endwhile
$CompHulls(slist, \&hulls)$
$Reconstruct(hulls, \&st)$
fprocedure

4.2.1 Updating geometry

Each movement instruction is realized at constant speed. Therefore, a movement can be decomposed into n constant intervals of translation in X, Y, Z and U along with rotation in W and A: $\delta A = \Delta A/n$, $\delta W = \Delta W/n$, $\delta X = \Delta X/n$, $\delta Y = \Delta Y/n$, $\delta U = \Delta U/n$ and $\delta Z = \Delta Z/n$.
As mentioned in the previous section, a line movement can be composed of several simultaneous instructions. Most of tool movements are composed of translations and axial rotations, which are independent. Therefore, the order in which the update of each movement is done is irrelevant. However for conical tools with a round end called "ball nose", simultaneous axial translations and column angle rotations are necessary. These two movements are obviously not independent. This can be a source of error (see Figure 3) because the real machine rotates the tool column angle at the same time as it translates it along its axis, while in the simulation, for each time interval, the tool is first rotated and next translated along its axis. However, in these cases the original CNC is already decomposed as a set of very small movements with a resolution very similar to the one needed in the machining. Therefore, these movements are not further decomposed in the machining.
The global coordinate system in which the geometry is expressed along time is sketched in Figure 4. The axis coincide with the machine axis X, Y and Z at the tool home position at the beginning of machining. Let $ct_k(xt_k, yt_k, 0.0)$ be the coordinates of the tool section center at instant k. The components of the normal vector of the section are $nt_k(nxt_k, nyt_k, 0.0)$. It should be noted that $nxt_k = cos(\omega_k)$ and $nyt_k = sin(\omega_k)$, being ω_k the column angle of the tool at instant k. The updated values of these coordinates at $k + 1$ are:

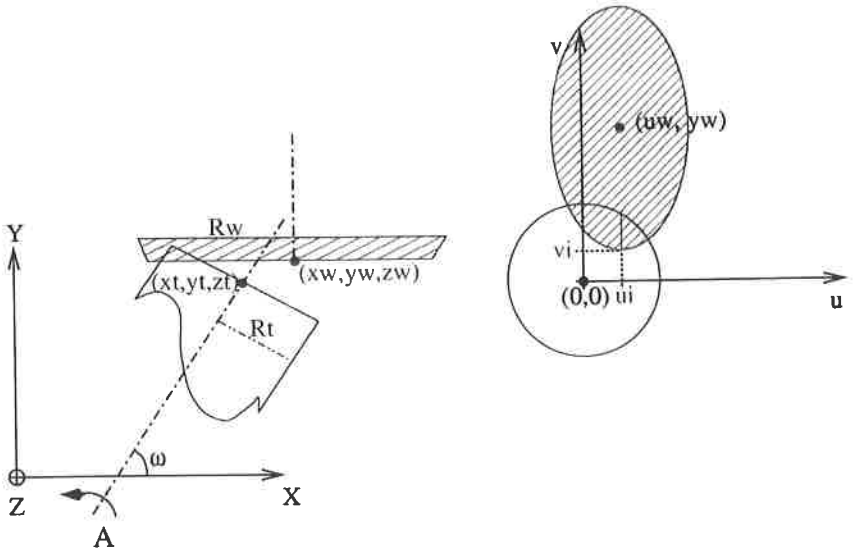

Figure 4: Coordinate system, axes and motion.

$$ct_{k+1}(xt_k + \delta U * nxt_k + \delta X, yt_k + \delta U * nyt_k + \delta Y, 0.0) \quad (1)$$

$$nt_{k+1}(nxt_k * cos(\delta W) - nyt_k * sin(\delta W), nxt_k * sin(\delta W) + nyt_k * cos(\delta W), 0.0) \quad (2)$$

The wheel normal vector is $nw = (0.0, -1.0, 0.0)$ and it remains unchanged during machining since the wheel only moves up and down in their axis. Being $cw_t(xw_t, yw_t, zw_t)$ the coordinates of the wheel center at instant k, their new values at $k+1$ are:

$$cw_{k+1}(xw_k, yw_k, zw_k + \delta Z) \quad (3)$$

Finally, being U and V the axes of the 2D local coordinate system on the tool section plane, and being $p_k = (u_k, v_k)$ a vertex of a segment list from the segment list at instant k, its updated coordinates are simply:

$$p_{k+1}(u_k * cos(\delta A) - v_k * sin(\delta A), u_k * sin(\delta A) + v_k * cos(\delta A)) \quad (4)$$

because local coordinates are only affected by the axial rotation A.

4.2.2 Intersection between cross sections

Figure 4 illustrates the computation of the intersection segment between the wheel cross section and the tool cross section. This calculus is performed in the tool section local coordinate system (U, V, T). The intersection between the two planes is a vertical line whose equation is:

$$u_i = (nxt_k * u_w + nyt_k * t_w)/nxt_k$$

being (u_w, v_w, t_w) the local coordinates of the wheel center projection onto the tool section:

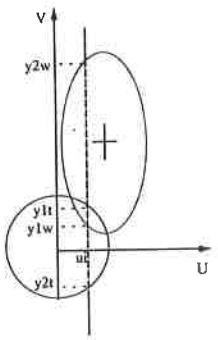

Figure 5: Intersection between the cross sections of the tool and the wheel.

$$\begin{cases} u_w = -(xw - xt) * nyt + (yw - yt) * nxt \\ v_w = zw \\ t_w = (xw - xt) * nxt + (yw - yt) * nyt \end{cases}$$

Note that, in order to simplify the notation, the subscript k indicating the instant in the intersection in the previous equations, has been removed.
The intersection line cuts the wheel section elliptic projection onto the tool plane [14] giving as result the segment $sw = ((u_i, v1_w), (u_i, v2_w))$ and it cuts the tool circular contour giving as result the segment $st = ((u_i, v1_t), (u_i, v2_t))$ (see Figure 5).
$v1_w$ and $v2_w$ and $v1_t$ and $v2_t$ are computed by solving the following equations systems, where r_w and r_t are the wheel and the tool radii respectively:

$$\begin{cases} u = u_i \\ \frac{u-u_w}{r_w * ny_t}^2 + \frac{v-v_w}{r_w}^2 = 1 \end{cases}$$

$$\begin{cases} u = u_i \\ u^2 + v^2 = r_t^2 \end{cases}$$

Finally, the searched segment is: $s = sw \cap st$.

4.2.3 Hull Computation

Depending on how the movement blocks have been computed, the segments list can represent more than a cut. However, in the hull computation step, each cut is treated separately. It is easy to identify which segments correspond to a specific cut because, in the previous step, the segments have been labeled according to the movement that has created them. Between the different cuts there are non-cutting movements in which the wheel goes up and the tool rotates. Therefore, each cut can be associated to a unique sequence of chained cutting movements and so, each segment can be labeled with a unique cut identifier. In the rest of the section, for clarity, we explain the computation of the hull of a single cut.
Let s be the segment list associated to a given cut. We define the *hull* of the set s as $hull(s)$ the closed polygon that encloses all segments of s and such that all its vertices belong to s. The hull is composed of two connected pieces: an arc of the circular

tool contour and a polygonal bit inside the circle. The internal polygonal piece is an approximation of the cut curve produced by the wheel on the tool section along the movement list. The denser is the segment list, the better is the approximation. The computation of the hull is centered at this cut polygonal. In the description below, the term *hull* will designate it directly omitting the arc.

The computation of the hull is a problem similar to the determination of the convex hull of a set of planar points [17]. However, the cut shape is not necessarily convex: the fluting cut shape for instance is concave. Therefore, well-known computational algorithms as the classical Graham scan [8] are not applicable in this case.

Besides, the segments in the list are naturally sorted according to a double criterion: (i) the wheel cross section that has produced them and, (ii) along time. Figure 6 shows different distributions of the segments depending on the type of movements of the tool. The uppermost distribution is a flute. As the tool has a simultaneous axial rotation and translation, the segments produced by a wheel section (pictured in green) rotate giving to the cut curve of one wheel section the final shape pictured in red in the figure. The hull is the blue curve. Observe that the hull of the segments is as well the hull of the wheel sections cut curves. The central distribution shows the end of a flute. The shape is similar to the previous one, but as the wheel goes up, it stops intersecting the tool, finishing the cut curve of a wheel section with a discontinuity. Finally, the rightmost picture shows a gash distribution. The wheel and the tool are perpendicular and there is no rotation. Therefore, the segments of a wheel section are overlaid. In this case, the cut curve of a wheel section is the longest segment and the hull curve is the set of extreme vertices of these segments.

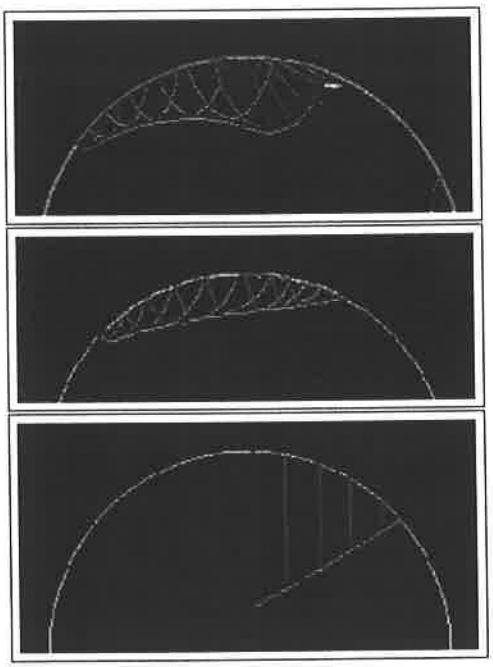

Figure 6: Hull computation.

The method that we have designed to compute the hull takes benefit from this double sorting. Instead of calculating the hull of the segments without taking into account the wheel that has produced them, it computes the hull of the cut curves (or cut seg-

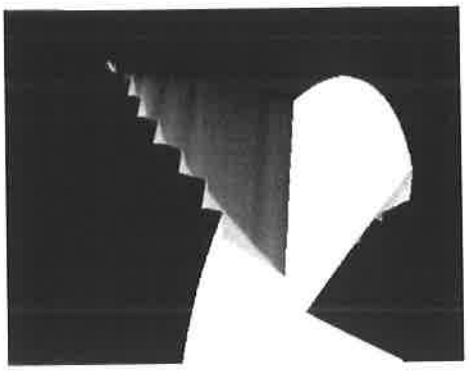

Figure 7: Tiling problem of a gashing operation.

ments) produced by each wheel section. Let s_{wi} be the set of segments produced by the cross-section w_i and let nw be the number of cross sections into which the wheel is discretized. Then, $s = \{s_{wi}, i = 1...nw\}$ and $hull(s) = hull(hull(s_{wi}), i = 1..nw)$. Besides, the segments of any set s_{wi} are naturally sorted along time because they have been computed sequentially thoughout the motion. Thus, any $hull(s_{wi})$ can be approximated as a polygon composed of all the segments vertices that are inside the tool circle.
Therefore, the computation of the hull is similar to the third step of Graham's algorithm. It consists of marching along the hulls of the segments of each wheel section segment list and keeping only those that define a partition of the plane such that all the remaining segments are at the same side of the partition. This algorithm can be optimized by traversing all the hulls (except the first and the last ones) between the intersections with the previous and last hull.
The keypoint for the hull computation is to provide a good approximation of the wheel intersection. Figure 7 shows an example of an error that may occur if the resolution is not good enough. The discretization of the wheel has been done at constant step for all cross sections. The contact point between the tool and the wheel has not been captured by the wheel discretization, therefore the hull does not capture the real cut. This error becomes obvious in the reconstruction of the surface giving a staircase appearance to the edge of cut.
Our approach avoids this error by performing an adaptive refinement of the wheel cross sections discretization in order to match the first and the last contact. Similarly, for a "fluting-like" segments distribution (i.e. whenever an axial rotation and a simultaneous translation occur), the wheel resolution is adapted to force intersections between hulls of successive cross sections. Conversely, in a "gash-like" segments distribution (without A angle), the resolution between the first and last contact sections, the wheel resolution can be rough.

4.2.4 Cross section reconstruction

The cross-section reconstruction consists of computing the boolean operation of difference between the cross-section previous contour shape and the cut curves polygonal approximation. Figure 8 shows an example of this step. It consists of first computing all the intersection points between the contour and the cut curves and next reconstructing the new shape by following the external edges.

Figure 8: Cross section reconstruction.

5 Simulations

The method has been implemented as part of a simulation application [18] that performs the following processes:

- Creation of the scene model starting from a textual description of the wheels and the initial tool.
- Translation of the CNC code into a linear list of movement instructions. This step consists of removing loops and alternative structures, filtering unnecessary expressions as well as evaluating variables.
- Parsing of the movements list with calls to a rough collision test in order to split the movements of a file into a block.
- Machining simulation for each block. This gives as result a temporal representation of the workshop (timeline) that contains a description of all the objects movements including the intermediate tool representations computed.
- Animation of the process, based on the timeline.
- Measurements of the tool on 2D cross-sections

During the simulation, the tool, the wheel and the temporal resolution are adaptively refined to match constructivity criteria. However, the application allows interactively modifying these resolutions to detect eventual errors.
Figure 11 shows an example of a machined tool. The triangles between contours have been colored according to the operation that has created them. Figure 11 shows a timeline representation of the workshop and Figure 10 a set of frames of the animation. The application is written in C and it uses Tcl/Tk for the interface and Perl for the CNC translation. Rendering is performed using Open GL calls. Therefore, the application is hardware independent. Specifically, we have benchmarked it on PC architectures with Windows 98 and Windows-NT as well as Solaris workstations.
A tool is computed in 10 to 30 seconds depending on the number of operations and the resolution of the approximation. The resulting model has between 3000 and 5000 triangles. The rendering time is about 3 to 5 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 the process time has been 47 seconds. The time to visualize each frame is 1

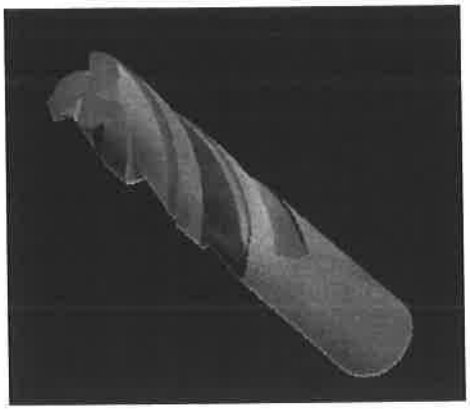

Figure 9: A final tool after fluting and gashing operations.

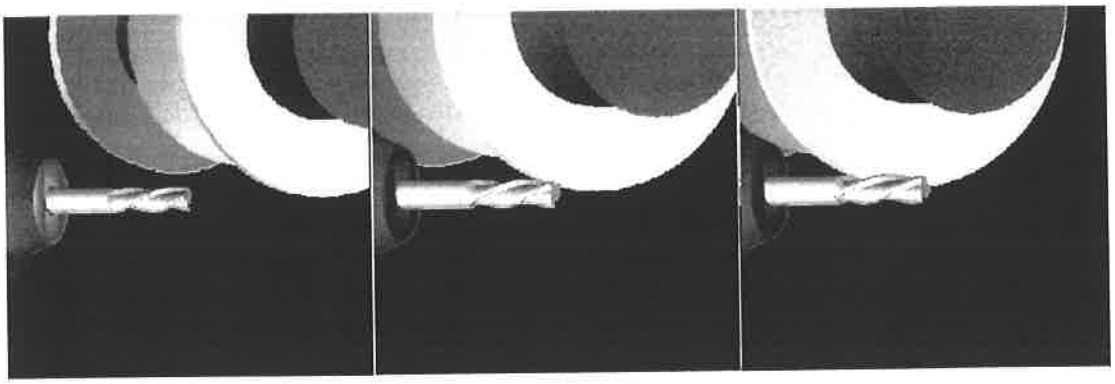

Figure 10: Several animation frames of a machining process.

second. Although the pre-process of all the display-lists before the visualization proper is very time-consuming, the animation process accelerates a 65 %.

6 Conclusions and future work

This paper describes a novel method for the simulation of drilling tools CNC machining. Our approach simplifies the 4D (space+time) boolean operations between the tool and the wheels by reducing them to a sequence of intersections between 2D perpendicular cross-sections along time. Specifically, the method discretizes the tool into cross-sections and simulates machining on the cross sections. Next, the shape of the tool is recomputed by tiling between contours.
The primary advantage of this approach is its simplicity. It addition, it provides user-control on the resolution of the simulation: spacing between cross-sections as well as time interval between consecutive intersections.
Starting from this work, new research and development lines are opened. Specifically, we are working on global pipelines that would put into the same process automatic CNC computation and tool verification. With such pipelines, given a final tool description, the CNC code to create it would be automatically computed, next using the CNC

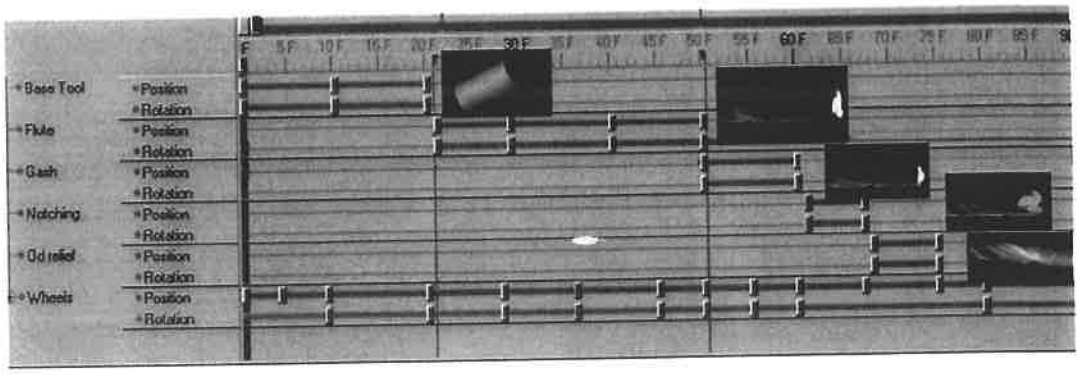

Figure 11: Timeline representation of the animation.

code as input, tool machining would be simulated. Finally, differences between the input and the output model could be computed and shown.

References

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

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

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

[4] E.L. Bohez. Compensating for systematic errors in 5-axis nc machining. *Computer-Aided Design*, (34):391–403, 2002.

[5] 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.

[6] L.W. Chang, H.W. Chen, and J.R. Ho. Reconstruction of 3d medical images: A nonlinear interpolation technique for reconstruction of 3d medical images. *CVGIP: Graphics models and Image Processing*, 53(4):382–391, 1991.

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

[8] R.L. Graham. An efficient algorithm for the determining the convex hull of a finite planar set. *Processing Letters*, 1972.

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

[10] K. Lee. *Principles of CAD/CAM/CAE Systems*. Addison-Wesley, 2000.

[11] D. Levin. Multidimensional reconstruction by set-valued approximation. *IMA, J. Numer. Anal.*, pages 173–184, 1986.

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

[13] D. Meyers, S. Skinner, and K. Sloan. Surfaces from contours. *ACM Transactions on Graphics*, 11(3):228–258, 1992.

[14] M. Mortenson. *Computer Graphics Handbook. Geometry and Mathematics*. Industrial Press Inc., 1990.

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

[16] J.M. Oliva, M. Perrin, and S. Coquillart. 3d reconstruction of comlex polyhedral shapes from contours using a simplified generalized voronoi diagram. *Computer Grahics Forum*, 15(3):C397–C408, 1996.

[17] F. Preparata and M.I. Shamos. *Computational geometry, an introduction*. Springer Verlag, 1985.

[18] A. Puig, L. Perez-Vidal, and D.Tost. 3d simulation of tool machining. *Research report- LSI-01-58-R, Software Dept. Polytechnical University of Catalonia*, 2001.

[19] 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.

[20] 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.

[21] D. Tost and A. Puig. Visualization of labeled segment cross-contour surfaces. *Volume Graphics 2001 Eds. K. Mueller and A. Kaufman*, pages 211–221, Springer Verlag 2001.

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

Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya

Research Reports - 2002

- LSI-02-1-R : *Anatomía de los antropónimos españoles*, Camps, R.
- LSI-02-2-R : *From Ternary Relationship to Relational Tables: A Case Against Common Beliefs*, Camps, R.
- LSI-02-3-R : *MKtree: Generation and Simulations*, Franquesa-Niubo, M. and Brunet, P.
- LSI-02-4-R : *A characterization of universal stability for directed graphs in the adversarial queueing model*, Alvarez, C. and Blesa, M. and Serna, M.
- LSI-02-5-R : *Transforming N-ary Relationships to Database Schemas: An Old and Forgotten Problem*, Camps, R.
- LSI-02-7-R : *Study on behavioral impedance for route planning techniques from the pedestrian s perspective: Some Findings and Considerations*, Nogueira, D. and Borges de Barros, H. and Pérez, L.
- LSI-02-8-R : *PROMO: detection of transcription regulatory elements using species-tailored searches* , Alba, M.M. and Messeguer, X. and Escudero, R. and Farre, D. and Nuñez, O. and Martinez, J.
- LSI-02-9-R : *Genome-wide analysis of the Emigrant family of MITEs: amplification dynamics and evolution of genes in Arabidopsis thaliana*, Casacuberta, J.M. and Messeguer, X. and Santiago, N. and Herráiz, C. and Goñi, J.R.
- LSI-02-10-R : *Interval Methods for Constructive Geometric Constraint Solving Problems*, Joan-Arinyo, R. and Mata, N. and Soto, A.
- LSI-02-11-R : *On-line Support Vector Machines for Function Approximation*, Martin, M.
- LSI-02-12-R : *Declarative characterization of a general architecture for constructive geometric constraint solvers*, Joan-Arinyo, R. and Soto-Riera, A. and Vila-Marta, S. and Vilaplana-Pasto, J.
- LSI-02-13-R : *Rendering techniques for multimodal data*, Ferre, M. and Puig, A. and Tost, D.
- LSI-02-14-R : *Supporting Process Reuse in PROMENADE*, Ribó , J.M. and Franch, X.
- LSI-02-15-R : *Evolving Partitions in Conceptual Schemas in the UML (Extended Version)*, Gómez, C. and Olivé, A.
- LSI-02-16-R : *Islands, coordination and parasitic gaps*, Morrill, G.
- LSI-02-17-R : *Revisiting Decomposition Analysis of Geometric Constraint Graphs*, Joan-Arinyo, R. and Soto-Riera, A. and Vila-Marta, S. and Vilaplana-Pasto, J.
- LSI-02-18-R : *MALLBA: A library of skeletons for combinatorial optimisation*, E. Alba and F. Almeida and M. Blesa and J. Cabeza and C. Cotta and M. Díaz and I. Dorta and J. Gabarró and C. León and J. Luna and L. Moreno and C. Pablos and J. Petit and A. Rojas and F. Xhafa
- LSI-02-19-R : *An Efficient Representation for Sparse Graphs*, Valiente, G.
- LSI-02-20-R : *Tree Edit Distance and Common Subtrees*, Valiente, G.
- LSI-02-21-R : *Universal stability of undirectd graphs in the adversarial queueing model.*, Alvarez, C. and Serna, M. and Blesa, M.
- LSI-02-22-R : *The complexity of restrictive H-coloring*, Diaz, J. and Serna, M. and Thilikos, D.
- LSI-02-23-R : *Using the Partial Least Squares (PLS) Method to Establish Critical Success Factor Interdependence in ERP Implementation Projects*, Esteves, J. and Pastor, J. and Casanovas, J.
- LSI-02-24-R : *Recent results on Parameterized H-Coloring*, Diaz, J. and Serna, M. and Thilikos, D.
- LSI-02-25-R : *Revisiting variable radius circles in constructive geometric constraint solving*, Ching-Shoei, C. and Joan-Arinyo, R.

- LSI-02-26-R : *Fast Connected Component Labeling Algorithm: A non voxel-based approach*, Ayala, D. and Rodríguez, J. and Aguilera, A.
- LSI-02-27-R : *Problems and conjectures on parameterized H-coloring*, Diaz, J. and Serna, M. and Thilikos, D.
- LSI-02-28-R : *Implementation of the methodology "Automatic Characterization and Interpretation of Conceptual Descriptions in ill-Structured Domains using Numerical Variables" (CIADEC) and application to a practical case*, Vázquez, F. and Gibert, K.
- LSI-02-29-R : *Exponential Speedup of Fixed Parameter Algorithms on $K_{3,3}$-minor-free or K_{5}-minor-free Graphs*, Hajiaghayi, M. and Demaine, E. and Thilikos, D.
- LSI-02-30-R : *A Proposal for Wide-Coverage Spanish Named Entity Recognition*, Arévalo, M. and Carreras, X. and Màrquez, L. and Martí, M.A. and Padró L. and Simón M.J.
- LSI-02-31-R : *H-colorings of Large Degree Graphs*, Díaz, J. and Nesetril J. and Serna, M. and Thilikós, D.M.
- LSI-02-32-R : *Extending the Carrel System to mediate in the organ and tissue allocation processes: A first Approach*, Vázquez-Salceda, J. and Cortés, U. and Padget, J. and López-Navidad, A. and Caballero, F.
- LSI-02-33-R : *e-Tools for the disabled and for the new generation of senior citizens. A Position paper*, Cortés, U. and Annicchiarico, R. and Vázquez-Salceda, J. and Sànchez-Marrè, M. and Marini, A. and Caltagirone, C.
- LSI-02-34-R : *The Synonym Management Process in SAREL*, Hernández, A. and Castell, N.
- LSI-02-35-R : *Improving Design and Implementation of OO Container-like Component Libraries*, Marco, J. and Franch, X.
- LSI-02-36-R : *Towards the definition of a quality model for mail servers*, Carvallo, J.P. and Franch, X.
- LSI-02-37-R : *Design of a Multimodal Rendering System*, Puig, A. and Tost, D. and Ferre, M.
- LSI-02-38-R : *Visual Clues in Multimodal Rendering*, Tost, D. and Puig, A. and Ferre, M.
- LSI-02-39-R : *Bisection of Random Cubic Graphs*, Díaz, J. and Do, N. and Serna, M.J. and Wormald, N.C.
- LSI-02-40-R : *1.5-Approximation for Treewidth of Graphs Excluding a Graph with One Crossing as a Minor*, Demaine, E. D. and Hajiaghayi, M. and Thilikos, D. M.
- LSI-02-42-R : *Searching the Solution Space in Constructive Geometric Constraint Solving with Genetic Algorithms*, Joan-Arinyo, R. and Luzón, M.V. and Soto, A.
- LSI-02-43-R : *Creating Agent Platforms to host Agent-Mediated Services that share resources*, Vázquez-Salceda, J. and Mérida-Campos, C. and Pujol, J.M.
- LSI-02-44-R : *Fast approximation schemes for K_{3,3}-minor-free or K_{5}-minor-free graphs*, Hajiaghayi, M. and Nishimura, N. and Ragde, P. and Thilikos, D. M.
- LSI-02-45-R : *Classes of Term Rewrite Systems with Polynomial Confluence Problems*, Godoy, G. and Nieuwenhuis, R.
- LSI-02-46-R : *Aplicación de algoritmos de clustering desarrollados en el entorno FIR a la predicción de la concentración de ozono*, Gómez, P. and Nebot, A. and Mugica, F.
- LSI-02-47-R : *Random scaled sector graphs*, Díaz, J. and Petit, J. and Serna, M.
- LSI-02-48-R : *Using Entropy-Based Local Weighting to Improve Similarity Assessment*, Núñez, H. and Sànchez-Marrè, M. and Cortés, C. U. and Comas, J. and Rodríguez-Roda, I. and Poch, M.
- LSI-02-49-R : *Boolean operations for 3D simulation of CNC machining*, Tost, D. and Puig, A. and Perez-Vidal, Ll.
- LSI-02-50-R : *Automatic Construction of Rules Fuzzy for Modelling and Prediction of the Central Nervous System*, Gómez, P. and Nebot, A. and and Mugica, F.
- LSI-02-51-R : *Caracterización e Interpretación Automática de Descripciones Conceptuales en Dominios Poco Estructurados usando Variables Numéricas*, Vázquez Torres, F.

Hardcopies of reports can be ordered from:

Núria Sánchez
Departament de Llenguatges i Sistemes Informàtics
Universitat Politècnica de Catalunya
Campus Nord, Mòdul C6
Jordi Girona Salgado, 1-3
08034 Barcelona, Spain
nurias@lsi.upc.es

See also the Department WWW pages, http://www.lsi.upc.es/