Improving the Extended Z-Buffer Z-Trace, Time-Trace and Animation

I. Blasquez and J.-F. Poiraudeau
LICN–IUT, Université de Limoges
Allée André Maurois, 87065 LIMOGES Cedex, FRANCE
E-mail : poiraudeau@unilim.fr

Keywords: extended z-buffer, geometric modelling, animation, performance study, NC milling simulation.

Abstract

In this paper, we propose an improvement of the extended z-buffer. This object representation is well adapted to the NC machining simulation as practiced in the workshops. However, it does not keep in memory the history of the construction of the workpiece. Indeed, to go back in the visualization process, it is necessary to replay the simulation from the first stage of the machining. We propose an algorithm to obtain an interactive simulation by using two new data structures called traces : one trace in a specified viewing direction, the other in time. The Z-trace or complete trace allows the part to be modelized again at any given moment. The time-trace or fast trace only displays a specific moment of the simulation but the setting up of an animation is possible because this trace can be consulted quickly. The experimental evaluation shows our method is compatible with the memory capacity and the processing speed of standard PC hardware, thus to workshops use.

1 Introduction

NC Milling Simulation is practiced in the workshops with the standard PC graphics hardware. With the simulation, the machine operator tries to find out the errors which may appear during a real machining. He needs to know where, when and how these errors appear so as to measure their impact and thus find adequate solutions. The best way of achieving this is to perform an interactive simulation where the operator can visualize not only the finished part but also the different phases of the machining. In order to follow the actual material removal and to visualize a specific moment of the machining, two possibilities are available. It is first possible to execute again the simulation from the beginning to the chosen moment, however, this is rather time consuming. The second solution consists in gluing together the virtual part and chips obtained and memorized during the simulation. To visualize faster a specific moment of the machining, we choose to focus on the second solution which allows the different machining phases to be stored in two data structures, and we introduce the notion of trace in the extended z-buffer. An animation is done thanks to a time-trace or fast trace which makes it possible to go back very quickly to a specific moment of the simulation. To obtain an interactive simulation and to enable the operator to correct the errors of the program, we propose a complete trace which allows the part to be modelized at any given moment, the simulation can then continue from a modified part. The originality of our work lies in this dual point of view : one trace in a specified viewing direction, the other in time.

2 Machining simulation with the extended z-buffer technique

The complete machining of a part involves a sequence of toolpaths. Each toolpath is characterized by the way the tool center moves (linear or circular interpolation) and by the positioning in space (starting point and end point) The simulation consists in following the movement of the tool by determining the volume of material removed from the part by the tool during a toolpath. To obtain a realistic simulation, we choose a "continuous" removal of material. The toolpath must be discretized into elementary positions representing the different positions of the tool center. *The different "stages" of the machining simulation are defined by the successive elementary positions of the tool center.* These elementary positions are obtained by generalizing the Bresenham's algorithm in a 3D space (3D DDA). The bigger the chosen resolution, the more numerous the positions.

To evaluate the volume removal during the simulation, a calculation of volume intersection of the part with each position of the tool center is obtained by using boolean operations, such as subtraction when milling. Complete machining consists of many thousand tool positions. Intersection calculations are more and more numerous, so the computational cost is intensive and time-consuming. During the simulation, a graphical representation is displayed. Several methods have already been developed in the field of machining simulation ; two main

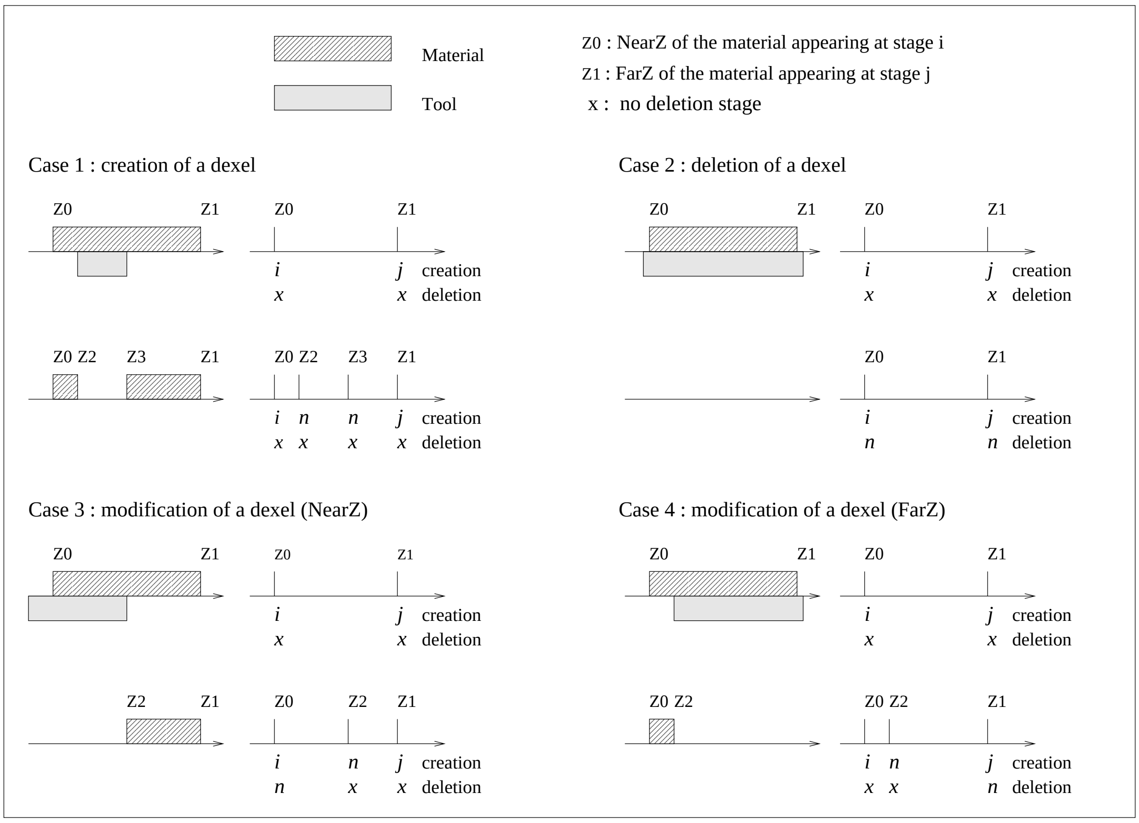
Material
Tool
Z0 : NearZ of the material appearing at stage i
Z1 : FarZ of the material appearing at stage j
x : no deletion stage
Case 1 : creation of a dexel
Case 2 : deletion of a dexel
Case 3 : modification of a dexel (NearZ)
Case 4 : modification of a dexel (FarZ)
Z0
Z1
Z2
Z3
creation
deletion

Figure 1: Boolean operations between two dexels at stage n.

categories can be distinguished : the methods based on 3 D models like the CSG method and the B-Rep which both require at least an $O(n^3)$ execution time [1], and the methods based on image space such as the extended z-buffer [2] and the Ray-Representation [3]. In this paper, we focus on the extended z-buffer technique. Van Hook was the first scientist to fix a view point in the image space and to define an extended z-buffer which contains depth elements, called dexels. A sorted list of dexels is associated with each pixel. A dexel represents a rectangular solid and corresponds to a part of the block behind a pixel for a fixed view point. With the extended z-buffer, an object can be displayed in a specified viewing direction [4]. The part and the tool are also represented by extended z-buffers. When machining, we work on the extended z-buffer data structure of the part, because only the tool removes material from the part. The boolean operations require two extended z-buffers [5]. At each elementary position the extended z-buffer of the part and the extended z-buffer of the tool have to be compared dexel by dexel. As a lists of dexels is associated with each pixel, a dexel can be interpreted geometrically as a segment of a given ray in the specified viewing direction. With the extended z-buffer technique, boolean subtraction operations can be simply performed on one-dimensional line segment intersections. The extended z-buffer takes an expected time of $O(n)$ [1]. A dexel contains graphic, spatial and modeling information about its near depth value (NearZ), its far depth value (FarZ), its color and a pointer to the following dexel. The material is bounded by the NearZ and the FarZ. Algorithmically, the comparison between the different extended z-buffers is performed by operations on sorted lists. We focus on two dimensions : the depth in the specified viewing direction and the time.

3 The z-trace or complete trace

For a given stage, it is possible to recreate a scene identical to the scene performed during the machining simulation thanks to the z-trace. The data of the trace are defined when the extended z-buffer is updated during the simulation. It is then necessary to define an algorithm using these data.

3.1 Characteristics of the z-trace.

The basic element of the z-trace, called Z-element, is derived from the dexel structure.

However, it contains two supplementary pieces of data which define the moment when a Z appears in the trace (creation stage) and when it disappears (disappearing stage). For each Z of the extended z-buffer (both for a NearZ or for a FarZ), a Z-element is defined in the trace. The Z-element contains the following information : the value, the color, the creation stage, the deletion stage and a pointer to the following element. A list of Z-elements is associated with each pixel on the image. As the extended z-buffer does not have any memory, the dexels are modified, even deleted, when material is removed. To show that a Z-element is no longer present in the buffer, it is necessary to use in the trace a piece of data which can memorize the deletion of the extended z-buffer. This is the deletion stage.

3.2 Setting-up of the z-trace.

3.2.1 Initialization

The initial part is described by the z-trace created during the initialization (stage 0).

3.2.2 Updating

The trace can be updated by studying the different boolean operations between the two extended z-buffers presented in figure 1. For a stage n of the simulation, a dexel of the part is compared to a dexel of the tool. The material is bounded by a NearZ ($Z0$) which appears at a stage i of the simulation and by a FarZ ($Z1$) which is created at a stage j of the simulation. Stages i and j take place at any time before the present stage n. The z-trace is built while the extended z-buffer is modified.

Creation of a dexel (case 1 in figure 1). When a new dexel is created in the extended z-buffer, two new Z-values appear. This appearance has repercussions on the z-trace : two new Z-elements are created. They are sorted in a list according to the increasing Z-values. Information such as Z-value and the creation stage must be updated.

Deletion of a dexel (case 2 in figure 1). When a dexel is deleted from the extended z-buffer, the information about this dexel is lost for the extended z-buffer. The z-trace must keep in memory all the operations performed on the extended z-buffer. The deletion of the dexel involves the updating of the deletion stage in the trace.

Modification of a dexel (cases 3 and 4 in figure 1). The front of a dexel can be reduced (case 3), then the value of its NearZ is modified. The back of a dexel can be reduced (case 4), then the value of its FarZ is modified. In both cases the trace undergoes two modifications : the updating of the Z-element already present in the trace (the deletion stage is updated), and the creation of a new Z-element whose value, color and creation stage are known.

Thanks to this analysis, it is possible to infer the following rules :

Rule 1. In the z-trace, all the modifications of the extended z-buffer are listed.

Rule 2. If a dexel (i.e. two new values of Z) is put into the extended z-buffer, then two Z-elements are created in the trace.

Rule 3. If a dexel (i.e. two new values of Z) is deleted in the extended z-buffer, then two Z-elements are updated in the trace.

Rule 4. If and only if a NearZ is modified in the extended z-buffer, then only one Z-element is added and only one other is updated in the trace.

Rule 5. If and only if a FarZ is modified in the extended z-buffer, then only one Z-element is added and only one other is updated in the trace.

3.3 Reconstruction of the scene at stage n

The extended z-buffer is going to be reconstructed as it was at stage n of the simulation, starting from the z-trace (figure 2a).

1. It is first necessary to select the Z-elements (figure 2b) that verify the following inequalities :

   $$\textbf{(creation stage} \leq \textbf{stage } n \textbf{) AND}$$
   $$\textbf{(deletion stage} > \textbf{stage } n \textbf{)}$$

   A list ordered according to the increasing Z is obtained in which the NearZ and the FarZ are alternated. According to rule 4 if the deletion stage of a single NearZ is updated in the trace, then a new NearZ is created. Thus if the extended z-buffer is constructed from the trace, a Z ordered list where a NearZ follows a FarZ is always obtained. The same is true with rule 5 and FarZ. Rules 2 and 3 also show that if two elements of the trace are modified or created, the result is necessarily a NearZ or a FarZ, which maintains the alternation between the NearZ and the FarZ.
2. Secondly, it is necessary to pair up Z-elements (figure 2c). They make up a dexel whose NearZ takes the value of the first Z-element, and whose FarZ takes the value of the second Z-element.
3. Finally, the extended z-buffer is obtained by linking the dexels.

4 Time-trace or fast trace

For a given stage, it is possible to display a

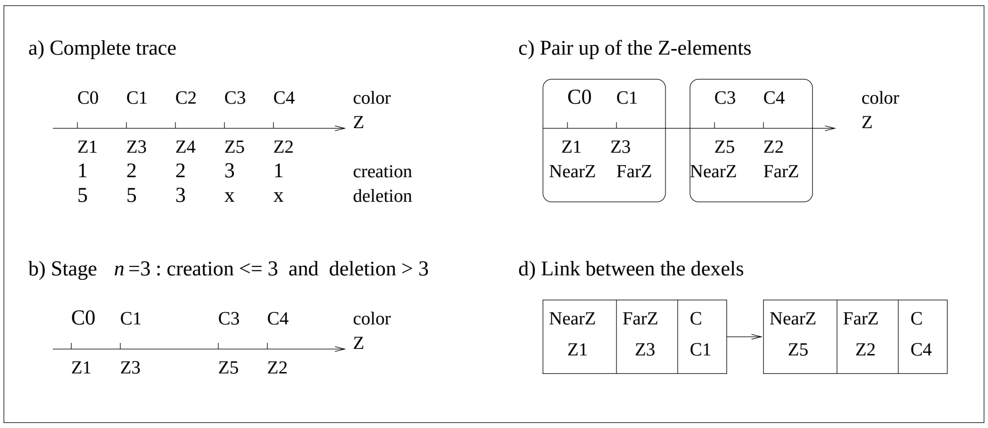

Figure 2: Reconstruction of the scene at stage $n = 3$.

scene thanks to the time-trace. The data of the trace are defined when the extended z-buffer is updated. It is then necessary to define an algorithm using these data. The z-trace and the time-trace are totally independent.

4.1 Characteristics of the time-trace.

With the extended z-buffer, the direction of view is along the increasing Z-values. When a pixel is displayed, it is necessary to consider the last Z of the extended z-buffer (it is therefore a FarZ).

Rule 6. The pixel displayed is the last FarZ of the extended z-buffer.

The time-trace evolves while the last FarZ is modified. As with the extended z-buffer, a list of time-elements is associated with each pixel on the image. The time-element is also the basic element of the time-trace. It contains the following information : the value and the color of the last FarZ, the creation stage which memorizes the time when the last FarZ appears in the trace and a pointer to the following element.

4.2 Setting-up of the time-trace.

4.2.1 Initialization

The initial part is described by the time-trace created during the initialization (stage 0).

4.2.2 Updating

The updating of the time trace is only possible by studying the different boolean operations between the two extended z-buffers on one-dimensional line segment, presented in figure 1. For a stage n of the simulation, a dexel of the part is compared to a dexel of the tool. The material is bounded by a NearZ ($Z0$) which appears at a stage i of the simulation and by a FarZ ($Z1$) which is created at a stage j of the simulation. Stages i and j take place at any time before the present stage n.

Creation of a dexel (case 1 in figure 1). The updating of this trace does not take this operation into consideration because the last FarZ is not modified.

Deletion of a dexel (case 2 in figure 1). If the deleted dexel is the last dexel of the extended z-buffer, then the updating of this trace takes this operation into consideration. Two cases can be examined. In the first case, the last dexel is the single dexel of the extended z-buffer. A new time-element is created : its creation stage is also the present stage n. Its Z-value and its color are the data of the background. In the second case, the last dexel of the extended z-buffer is preceded by another dexel, then a new time-element is created : its creation stage is also the present stage n. Its Z-value and its color corresponds to the Z-value and the color of the previous dexel.

Modification of a dexel (cases 3 and 4 in figure 1). The front of a dexel can be reduced (case 3), then the value of its NearZ is modified. In this case, the time-trace is unchanged because, according to rule 6, the time-trace is updated when the last FarZ is modified. The back of a dexel can be reduced (case 4), then the value of its FarZ is modified. According to rule 6, the time-trace is updated if and only if the modified FarZ is the last FarZ. A new time-element is created : its creation stage, its new FarZ-value and it colors are also known. The time-trace is ordered in time along the creation stage.

The updating is performed only in the two following cases :

1. when the FarZ of the last dexel is modified (case 4),
2. when the last dexel is deleted.

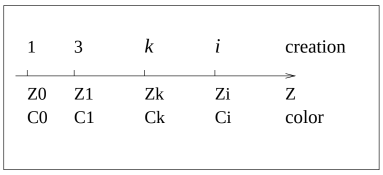

Figure 3: Representation of the time-trace.

4.3 Reconstruction of the scene at stage n.

The color of each pixel has to be the same as it was at stage n of the simulation, starting from the time-trace (figure 2a). It is necessary to examine the time-trace up to stage s which verifies the following inequality :

$$\textbf{(creation stage } s \geq \textbf{stage } n \textbf{)}$$

If creation stage s = stage n, then the Z-value and the color of the pixel at stage n are found. If creation stage s > stage n, then the Z-value and the color of the pixel at stage n are the Z-value and the color of the Z-element which precedes the Z-element at stage s, because the next modification takes place after stage n.

5 Experimental evaluation

Let's take the example of the machining simulation of a small mould. Machining this mould requires 17,108 toolpaths - most of them being 0.5 mm long - as well as a 107×72×33 mm stock and a 3 mm tool. The machining takes 1 h 30 min with a three-axis milling machine. To evaluate the algorithms of the extended z-buffer and of the traces, a 300 Mhz PC is used. It has a level-two 512 Ko cache memory and a 128 Mo RAM. The simulation software uses direct access to a standard graphics board. A 640×480 image (307,200 pixels) [1] is used. The data structures have 307,200 dexels, Z-elements and time-elements when initialized. A dexel and a time-element use 12 octets while a Z-element has 16 octets. For an image of a given size, the number of elements varies according to several parameters : the size and the shape of the stock, the number of toolpaths and the machining strategy used, and the view point.

First, we study the memory and the time required to create traces. We analyse the results related to : the construction of an extended z-buffer, the construction of an extended z-buffer and a z-trace, and the construction of an extended z-buffer and a time-trace. The results related to the memory used are presented in table 1. The creation of the extended z-buffer takes 99.42 s. The creation of the extended z-buffer and the setting-up of the z-trace take 103.04 s while the creation of the extended z-buffer and the setting-up of the time-trace take 101.23 s. These times do not take into account the display time as only the creation of the extended z-buffer and the traces are significant here.

We now study the reconstruction times of scenes without display. Considering the large number of elementary positions present in the simulation (44,084), we list simulation scenes which correspond to arbitrary elementary positions in table 2. For example, the scene in which the part has already undergone a tenth of the simulation is listed in the first column. This corresponds to the position 4,408 when the extended z-buffer is constructed. To simplify matters, let's say that this simulation stage is called "stage $\frac{1}{10}$". In table 2, the time required is listed for any given stage : for the construction of the extended z-buffer, for the reconstruction of the scene with the z-trace, and for the reconstruction of the scene with the time-trace.

These results show how useful and efficient the traces are. The z-trace is constructed in 103.04 s, that is to say only 3.6 % more than the construction of the extended z-buffer itself. However, the z-trace requires 45.64 Mo, that is to say 921.5 % more than the extended z-buffer. The construction of the z-trace requires only slightly more time than the construction of the extended z-buffer while the memory occupation is much bigger for the trace than for the extended z-buffer itself. To reconstruct any given scene, the z-trace uses in the worst case only 5.2 % of the time necessary to construct the same scene with the extended z-buffer. A choice must be made between memory and execution time.

It is also possible to study the time-trace. It is constructed in 101.23 s, that is only 1.8 % more than the construction of the extended z-buffer itself. The time-trace requires 24.88 Mo, that is to say 456.9 % more than the extended z-buffer. To reconstruct any given scene, the time-trace uses in the worst case only 0.5 % of the time necessary to construct the same scene with the extended z-buffer.

[1] For this specified view, one pixel corresponds to 0.26 mm.

Toolpaths (Positions)	Pixels of the part	z-buffer			z-trace	time-trace
		Created dexels	Deleted dexels	All the dexels	Z-elements	Time-elements
17,108 (44,084)	121,067	83,217 *0.96Mo*	52,427 *0.60Mo*	390,417 *4.46Mo*	2,991,055 *45.64Mo*	2,174,530 *24.88Mo*

Table 1: Memory occupation.

	Stage $\frac{1}{10}$	Stage $\frac{1}{4}$	Stage $\frac{1}{2}$	Stage $\frac{3}{4}$	Last stage
z-buffer[2]	10.88	26.03	50.14	74.20	99.42
z-trace[2]	0.56	0.57	0.57	0.58	0.57
t-trace[3]	0.06	0.09	0.17	0.25	0.31

Table 2: Reconstruction of the scene (time in seconds).

Although the z-trace and the time-trace are totally separate, the time-trace can be set up faster than the z-trace. Indeed, the former is only updated in certain cases (cf. rule 6) whereas the latter follows all the updating stages of the extended z-buffer. Furthermore, in the time-trace the time-elements are already put in order when created while in the time-trace the Z-elements have to be put in order in the list. By using the time-trace the scene is reconstructed faster. Thus it is not necessary to search the wanted value (the color) in this trace, contrary to the z-trace where all the Z-elements must be examined to check if the inequalities allowing for the reconstruction of the scene are respected, whatever the chosen stage. The time-trace takes up less memory than the z-trace. It seems better adapted to reconstruct a scene rapidly but the z-trace still has the advantage of recreating the extended z-buffer. In the latter case it is possible to go beyond mere observation and to perform again the simulation with different toolpaths. Again, it is necessary to choose here between time and the number of possible actions. The previous results show the speed of the simulation : the machining with a milling machine takes about 1 h 30 min whereas the visualization of the final part with the extended z-buffer takes less than 100 s. The results also show how efficient the data structures are : reconstructing one given stage of a time or z-trace is always faster than constructing the extended z-buffer up to the same stage. This is why we choose the faster time-trace to perform an interactive simulation and to revisualize the simulation of the part machining between any two stages chosen by the user. The animation is studied by showing all the stages successively on the screen. It is done in two ways : first, the simulation is run by using the extended z-buffer, then the time-trace is used.

The animation is done between two stages called S_1 and S_2. It corresponds in table 3 to the interval $[S_1,S_2]$. For instance, in the column which contains the interval $[\frac{1}{10},\frac{1}{4}]$, the animation starts with an image corresponding to a tenth of the toolpaths ($S_1{=}\frac{1}{10}$) and it ends when a fourth of the toolpaths are processed ($S_2{=}\frac{1}{4}$). The simulation of the machining is replayed from stage S_1 to stage S_2. When $S_1{\neq}0$, a preliminary phase is necessary. The first stage of the simulation (S_1) is also located in the data structure. Now the "actual" animation (what is seen by the user) can begin and take place from stage S_1 to stage S_2. That is why we listed execution times in the two following phases :

- from 0 to S_1 (if $S_1{\neq}0$) (column (a)). For the first line, the execution time includes the construction of the extended z-buffer up to stage S_1 and the diplay of the image corresponding to stage S_1. For the second line, the execution time includes the search in the trace for the Time-element corresponding to stage S_1 and the display of the color of this element for each pixel.

[2]For one simulation.
[3]Average for 100 simulations.

	$[0,\frac{1}{10}]$	$[\frac{1}{10},\frac{1}{4}]$		$[\frac{1}{4},\frac{1}{2}]$		$[\frac{1}{2},\frac{3}{4}]$		$[\frac{3}{4},\text{end}]$		[0,end]
	$[0,\frac{1}{10}]$	$[0,\frac{1}{10}]$	$]\frac{1}{10},\frac{1}{4}]$	$[0,\frac{1}{4}]$	$]\frac{1}{4},\frac{1}{2}]$	$[0,\frac{1}{2}]$	$]\frac{1}{2},\frac{3}{4}]$	$[0,\frac{3}{4}]$	$]\frac{3}{4},\text{end}]$	[0,end]
	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(a)	(b)	(b)
z-buffer	11.21	10.71	16.31	25.81	27.36	50.81	27.24	76.02	26.86	108.86
t-trace	1.49	0.55	2.03	0.61	7.19	0.66	7.19	0.71	7.53	24.44

Table 3: Animation (time in seconds).

- from S_1 to S_2 (column (b)). For stages S_i such as $S_1 < S_i \leq S_2$, the execution time of the first line includes the evolution of the extended z-buffer from stage S_1 to stage S_2 and the updating of the image for all the stages S_i. In the second line, the execution time includes the search in the trace for the Time-element corresponding to stage S_i and the display of the updated image. If S_1=0, then the image corresponding to stage 0 (the part) is displayed at the beginning of the simulation.

It is interesting to note that if all the columns labelled (b) in table 3 are added excluding the last one, the result obtained is coherent with that of the last column which corresponds to the entire simulation.

The previous results show how useful a time-trace is to set up an animation. The animation obtained with the time-trace for the first machining stages is indeed ten times as fast as the one obtained with the extended z-buffer. In the case of the entire machining simulation, the time-trace is 4.4 times as fast as with the extended z-buffer. The extended z-buffer is then obtained with 17,108 toolpaths in 108.86 s, that is to say 6.36 ms for each toolpath. The data structure of the time-trace keeps the data necessary for the simulation of 17,108 toolpaths, which is obtained in 24.44 s, that is to say 1.43 ms for each toolpath. With the time-trace, it is possible to get a faster, more efficient and friendly-user animation.

In table 3, the scenes are replayed chronologically but the animation can also "go back in time" and "rewind" the trace. It starts with the final machining scenes, and ends with the first machining scenes. The algorithmic solution consists in turning the linked ordered lists of time-elements associated with each pixel into doubly-linked ordered lists.

6 Conclusion

To recreate a given scene in NC milling simulation using extended z-buffer, we have presented two methods : the first one is based on the depth according to a specified viewing direction, the second is based on time. The z-trace is a complete trace which can entirely reconstruct a scene. On the other hand, the time-trace is faster but only displays the scene. The setting up of the traces is all the more useful as the extra time necessary to elaborate the traces is very limited compared with the reconstruction of the scene. Thanks to the time-trace, the animation is made possible, and the user can replay interactively any given phase of the machining. Although this method is memory consuming, it is compatible with the memory capacity and the processing speed of PCs assembled today.

References

[1] R.B. Jerard, S.Z. Hussaini, and R.L. Drysale. Approximate methods for simulation and verification of numerically controlled machining programs. *The Visual Computer*, 5:329–348, 1989.

[2] T. Van Hook. Real–Time Shaded NC Milling Display. *Computer Graphics (Proc. SIGGRAPH'86)*, 20(4):15–20, 1986.

[3] J.P. Menon and H.B. Voelcker. On the Completeness and Conversion of Ray Representations of Arbitrary Solids. Technical report, IBM RC 19935, T.J. Watson Research Center, Yorktown Heights, 1995.

[4] Y. Huang and J.H. Oliver. NC milling error assessment and tool path correction. *Computer Graphics (Proc. SIGGRAPH'94)*, pages 287–294, 1994.

[5] K.C. Hui. Solid sweeping in image space – Application in NC simulation. *The Visual Computer*, 10:306–316, 1994.