Efficient tool path computing in CAD/CAM. Distributed computation for trajectory generation

Antonio Jimeno-Morenilla, Francisco Maciá-Pérez, Juan Manuel García-Chamizo.

Computer Science Technology and Computation Department (DTIC). University of Alicante.
San Vicente del Raspeig s/n. 03080 Alicante. (Spain).

Abstract

Tool path generation is one of the most complex problems in Computer Aided Manufacturing. Although some efficient strategies have been developed to solve it, most of them are only useful for 3 and 5 axis standard machining. The algorithm called Virtual Digitising computes the tool path by means of a “virtually digitized” model of the surface and a geometry specification of the tool and its motion, so can be used even in non-standard machining (retrofitting). This algorithm is simple, robust and avoids the problem of tool-surface collision by its own definition. However, its computing cost is high. Presented in the paper there is a Virtual Digitising optimisation that takes the advantages of distributed computing (using different computer architectures) in order to improve the algorithm speed. A comparative study will show the gain and precision achieved. The proposed algorithm has been successfully implemented in a commercial CAD/CAM system specialized in shoe last making. Finally, some illustrative examples are shown.

Keywords: virtual digitizing, shoe last machining, CAD/CAM, tool path generation, parallel architectures.

Introduction

Tool path generation problem consists in obtaining a trajectory of tool centres that defines the surface to be machined with a given precision [1,2]. Fig. 1.a shows the trajectory of a circle centre point in order to define a rectangle; the problem is presented in 2D, for 3D surfaces the problem becomes more complex.

There are different techniques to solve the problem [3,4,5]; however, they are defined only for spherical tools or three axis machines. The virtual digitizing algorithm [6] offers a generic solution which is independent of tool geometry or mechanical machine restrictions. The algorithm is based on the mathematical morphology paradigm, so that every machining function is defined by means of a morphological operation. For instance, tool path computation is performed for a basic three dimensional dilation where the object to mechanise is the shape to dilate and the tool is the structuring element; however, 3D versions of morphology operations are not efficient and techniques are still in development.

The virtual digitising algorithm

Centre tool points are obtained by virtually touching the object to mechanise. This algorithm, typically used to compute pencil curve tracing [7], is used here for imitating the way of work of traditional copier machines, the process can be divided into four phases:

1. Definition of the tool motion
2. Obtain a discrete model of the part surface
3. Simulate the tool motion
4. Virtual digitisation process
Fig. 1. a) Circle trajectory around a rectangle, b) virtual digitalisation process example for 2D shapes.

The digitalization algorithm becomes simple once the surface and tool motion is well defined. Basically, the behaviour can be described as follows: For each point of the trajectory the part surface is transformed in order to face the cutting tool. Then, the minimum distance from every grid point to the tool is computed in the direction of tool attack axis. This minimum distance determines the tool centre point for this step in the virtual digitalisation process. Physically, we select the point that touches the tool surface in first place when the tool is moved along the attack axis. The process is similar to that of is used for obtaining z-maps of the tool envelope surface, typically used for 3-axis CNC machining: the inverse offsetting method [8] and the direct cutting simulation [9].

Obtaining a discrete model for computing the tool path suppose precision loose. On the other hand, the algorithm becomes simpler and faster when a discrete model is used.

As shown in Fig. 1.b (a simple example in 2D in order to obtain a single trajectory point for a circular tool), the minimum distance represents the tool centre distance in order to reach the grid point without collision with the shape.

1: t=0
2: while t<=1 do
4: for each surface,∈Object do
5: for each point,∈ surface, do
6: point,=point,•Γ(t)
7: d =d,(point,‘,EE)
8: if d<MinDistance then MinDistancie=d endif
9: endfor
10: endfor
11: EECenter=ComputeCenter(MinDistance,Γ(t))
12: AddTrajectory(EECenter)
13: Increment(t)
14: endwhile

Table 1. Basic virtual digitising algorithm

Algorithm cost

Computational cost is analysed in this section in terms of the problem size for the algorithm introduced in table 1. The operator used is $O$ to determine an upper limit of the computation cost.

Analysing algorithm, it is possible to observe up to three nested loops. One of them, the most internal one, is used to access to every grid point in the selected surface, that is, it consists into two loops, one for rows and the other for columns in fact.

Every loop iterates on different data. The most external one goes through every trajectory position. In order to obtain a good quality finishing it is necessary produce, at least, as many trajectory points as grid points the surfaces have. The finishing quality also depends on the material to be milled, for example, metallic ones produce best results but need more trajectory points than organic ones (plastics) also more grid points (surface definition).
Let \( n = \max(m \cdot s, \forall m, s \in DS) \)  
\[ (1) \]

Let assume \( n \) as the maximum number of grid points (\( m \) for \( u \) direction, and \( s \) for \( v \)) of all the surfaces.

The cost of the first loop, and consequently the algorithm cost, is:
\[ \lim_{n \to \infty} \left( n \cdot \left( c_1 + c_2 \cdot O(n) + c_3 + c_4 \right) \right) = O(n^2) \]
\[ (2) \]

where \( c_i \) are constant values. As a guide, a usual value for \( n \) in shoe last machining is about thirty thousand, that is, a high computer cost; in next sections we will show some quantitative examples with time measures.

**Virtual digitising parallelization**

In this section we propose an alternative to the algorithm introduced in order to improve the average response time. We will show that using this kind of optimisations should reduce the average response time for the machining.

The independence between different trajectory values makes possible the parallel computation for the minimum distance function as shown in table 2.

| 1: Send(Object, p0, p1, ..., pn-1) |
| 2: In parallel \( t=idproc \times n \) |
| 3: while \( t < (idproc + 1)/n \) do |
| 4: \( \text{MinDistance} = \infty \) |
| 5: for each \( \text{surface}_i \in \text{Object} \) do |
| 6: for each \( \text{point}_{j,k} \in \text{surface} \) do |
| 7: \( \text{point}_{j,k}' = \text{point}_{j,k} \cdot \Gamma(t) \) |
| 8: \( d = d_\text{EE}(\text{point}_{j,k}', \text{EE}) \) |
| 9: if \( d < \text{MinDistance} \) then \( \text{MinDistance} = d \); endif |
| 10: endfor |
| 11: endfor |
| 12: \( \text{EECenter} = \text{ComputeCenter}(\text{MinDistance}, \Gamma(t)) \) |
| 13: AddTrajectory(EECenter) |
| 14: Increment(\( t \)) |
| 15: endwhile |
| 16: endparallel |
| 17: In parallel if \( idproc <> 0 \) then \( \text{Send}(\text{Trajectory}, p_0) \) Endparallel |
| 18: Collect(\text{Trajectory}) |

**Table 2.** Parallel version of virtual digitizing. The \( idproc \) variable contains the local processor number and \( n \) the whole number of processes.  

Theoretically computation cost becomes from 2) expression to:
\[ m = n / p + o \]
\[ \lim_{n \to \infty} \left( n \cdot \left( c_1 + c_2 \cdot O(m) + c_3 + c_4 \right) \right) = O(m^2) \]
\[ (3) \]

Where \( p \) is the number of processors and \( o \) is the cost associated to the communication time.

The computational cost, without considering communication time, is expressed in 4). In the best case, the total computing time could be reduced by the number of processors available. The results of experiments will show the reality of this optimistic prediction.
\[ \lim_{n \to \infty} \left( \frac{nGS_{\text{proc}}}{nproc} \cdot c_1 + c_2 \cdot nGS_{\text{proc}} + c_3 + c_4 \right) = O\left( \left( \frac{nGS_{\text{proc}}}{nproc} \right)^2 \right) \]
\[ (4) \]
Computer architectures for virtual digitising optimization

It is possible to use parallel computer architectures to make the computation in an efficient way. Let suppose \( n \) processors (e.g.: transputers, digital signal processors DSPs, and so on) and a master processor that controls the data exchange. The basic idea consists of distribute all the grid points of the discrete part surface between the processors, so every processor only computes the tool distance formula for its own surface extent. Finally, every processor sends back the tool path for its extent to the master processor which joins the data and obtains the solution.

Two different architectures are proposed in Fig. 2. In the first one, there is no communication between neighbour processors, the master processor distributes each overlapped extent to each processor, so overlapped information is twice in neighbours. In the second one, the master processor distributes extents without overlapping, however, neighbour processors are communicated by links, so a processor receives the overlapped points from the neighbour one. In first architecture, main processor makes the main effort in extent distribution, in second one, overlapped information distribution is carried out by every processor in parallel.

Experiments for shoe last machining

Shoe lasts are a key part of shoe manufacturing. They are moulds on which every component of shoe has to be fitted. For the standpoint of view of geometry, they can be considered as free-form surfaces; non-uniform rational b-splines (or NURBS) usually model these kinds of surfaces. Due to shoe last precision requirements (±0.01 millimetres), the high definition of the foot (about 30,000 points per surface) becomes into a bottleneck for most of common CAD/CAM systems. The parallel algorithm presented in the paper has been implemented in a commercial CAM system Forma3D [10] (Spanish Footwear Research Institute, INESCOP).

Fig. 3 shows the extent distribution among 4 processors for a shoe last. Notice that every extent is overlapped with its neighbours. Overlapping is necessary because of the tool geometry and the minimum distance computation, since for the last tool path point of each processor, we need the rest points affected by the tool geometry. For this example we have used only the \( u \) direction for the extent distribution, since the tool geometry makes the overlapping smaller in this direction. For other tool geometry (spherical, cylindrical, conical) it is possible to distribute extents in both directions with a minimum overlapping.

Distributed computing on a local network

Shoe last making represents a traditional sector in most of Mediterranean countries. It means that every shoe last manufacturer lacks of resources for high performance computer architectures. Most of them own a small local network for management purposes. The distributed algorithm presented in the paper has been tested on such kind of platform: a small local network (up to four Pentium II®-based PC’s).
Fig. 3. Shoe last distribution among four processors

Fig. 4 presents results on a Client/Server based architecture that uses the NetBEUI interface resident in most of Microsoft Windows® systems. Computing times are reduced up to 66% of total sequential time in graphic 4.a. In Fig 4.b we observe the effect of the overlapping zone (that depends on the helicoidal step), the bigger the overlapping zone is the smaller maximum efficiency is obtained.

Fig. 4. Virtual digitising parallelization on a local PC network: a) Effect of points per round (overlapping) vs number of processors (fixed step of 1 mm), b) Virtual digitising parallelization on a local PC network. Effect of helicoidal step vs number of processors (720 points per round).

Fig. 5. a) Virtual digitising parallelization on the IBM® SP2 system. Precision vs. number of processors, b) Overlapping architecture comparative: with overlapping (CS) and without overlapping (SS) for 180 and 360 points per millimeter of trajectory.
The experiment illustrated in Fig. 5.a shows a better parallel fitting of the algorithm. This improvement it is due to the use of better bandwidth of the communication links and the use of a dedicated architecture. Times are reduced up to 80% of total sequential time.

Finally, we probe the use of the two architectures proposed for parallel computing: with and without overlapping distribution. The second alternative presents the best results due to communication synchronization times, this effect proportional to the number of processors (see Fig. 5.b).

Fig. 6. a) Tool trajectory computed in parallel experiments. b) Lady shoe last used in experiments.

Conclusions and future work

The procedure of virtual digitizing it is simple to implement, offers good results and avoid the problem of tool collision by its own definition. On the other hand, the algorithm is not suitable for general purpose machining algorithms since it is too slow versus other types of tool path generation algorithms. The use of parallel computing may be useful to solve high cost problems coming from the CAD/CAM world. In particular, this algorithm has been applied in different parallel architectures (from local PC-networks to specialized MIMD ones) achieving good results in tool path generation where other solutions have not been implemented yet. However, in the algorithm parallelization, the master processor is the responsible for low scalability because it introduces a strong sequential portion of code (Amdahl’s Law). Future studies will try to improve the algorithm efficiency by reducing the bottle neck in the master processor, distributing either the tool path computation or data distribution on processors and achieving better computing times.

References