Shoe last machining using Virtual Digitising

Antonio Manuel Jimeno Morenilla
Assistant professor of the University of Alicante (Spain)
Assistant researcher of the Spanish Footwear Research Institute (INESCOP)

Juan Manuel García Chamizo.
Director of the Computer Science Technology and Computation department. University of Alicante (Spain).

Faustino Salas Pérez.
Director of the CAD/CAM department the Spanish Footwear Research Institute (INESCOP).

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

Abstract
Shoe lasts are the moulds used in the footwear industries in order to mount the shoe. Most of the machines used in the sector to make lasts are simply mechanical copiers. CAD/CAM systems have just arrived to the shoe last market but its accuracy and efficiency is not better than traditional machines, for this reason new systems have difficulty to implant. Presented in the paper there is a tool path generation algorithm that takes the advantages of traditional copier systems that do not fulfil the CNC standards. The tool path is computed from a "virtually digitised" model of the last surface. The algorithm is then analysed in terms of computing cost and accuracy and refined by applying a series of optimisations. Some computer architectures are proposed in order to reduce the computation time. The proposed algorithm has been successfully implemented in a commercial CAD/CAM system specialised in shoe last making. Finally, some illustrative examples are shown.

Introduction

Traditional shoe last machining

In the first half of this century a series of new machines appeared in the shoe market. Those machines were able to produce a couple (right and left sides) of shoe last pairs just in 5 minutes. Their precision was ± 0.1 mm. The way of work of those machines was quite simple: two turning lathes, one for the original model and the other for the cutting wheels. The copying lathe consisted of a metallic torus with the same dimension and relative position as the cutting wheels; the original last model was put in the copying lathe by holding his extremes. In the cutting lathe a rough model was locked, when the machine started, the copying wheel was touching the last surface and a group of arms transmitted the spiral movement to the cutting wheels which perform the same movement and finished the copied model.

The process was simple and robust, original models holders avoid the collision with the rough model's holders, mechanical movement guarantied model precision and the only restriction was the motor cinematic. After this process, the operator unlocked the copied model and removed the holders by hand (the heel and the toe part). Using these machines surface corners are well defined and copied surface appears smooth. Last makers used this kind of machines for more than 50 years due to the fact they were accurate and simple to manage.

State-of-the-art in shoe last machining

In the recent years new numerical control (NC) machines have appeared in the last making world. They are managed by computer and the most are similar to the traditional ones (they are also turning lathes). The basic principle is very similar to the ancient machines: there is a mechanical digitiser that touches the original surface and stores the tool centre points (tool path) in the computer. Finally the NC reads these points that make the tool path.

Shoe lasts can be considered as free-form surfaces; non-uniform rational b-splines (or NURBS) usually model these kinds of surfaces. There are a lot of commercial CAM applications that can generate tool paths for NURBS. Examples of such computer aided manufacturing (CAM) systems include Tebis[1] from Germany, Work-NC[2] from France, Clicks[3] from Japan, Z-Master[4] from Korea, and Shoemaster[5] from England. Most of them are prepared to detect tool collisions and so on, however they are not used for NC shoe last machines. Presented in the paper is a straightforward tool path generation algorithm, which is as accurate and simple as the ancient shoe last machines. It has been implemented in a commercial CAM system Forma3D[6] (INESCOP, Spain) as well.

Problem definition and overall procedure

It is assumed that the part surface is represented by a collection of parametric surfaces. In the following, we will use the term "surface" for a parametric surface *r(u,v)*, such a Bezier or NURB surface. In the case of a shoe last, a part surface consists of 3 different surfaces one for the last and the other for the machine holders.

Tool path generation problem: Obtain a trajectory of tool centres that defines the part surface with a given precision. Figure 1 shows the trajectory of a circle centre point in order to define a rectangle. In this case, the problem is presented in 2D. For 3D surfaces the problem becomes more complex. This problem can be related to the dilation process from the mathematics morphology 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.

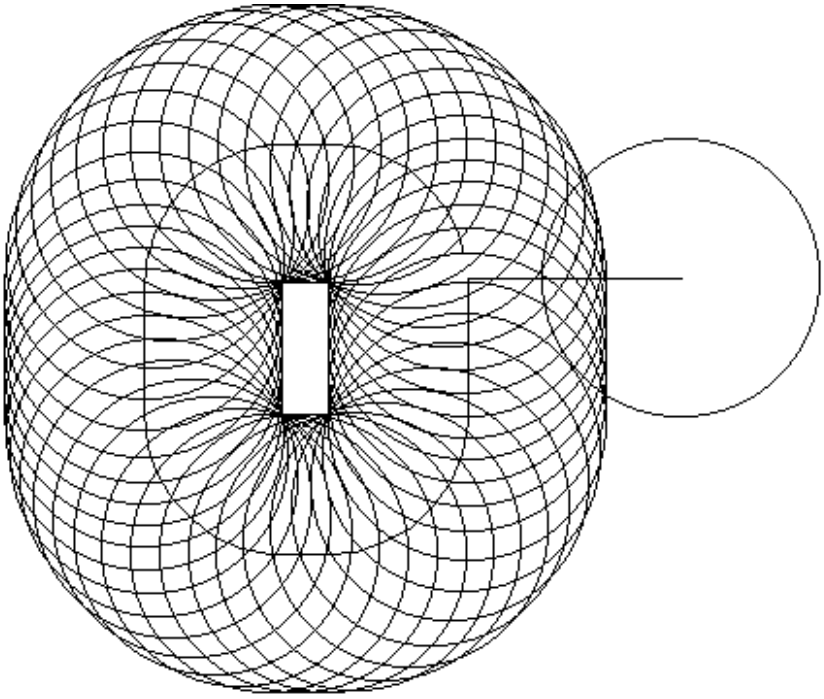

Figure 1 Circle trajectory around a rectangle

There are different techniques to solve the problem:

Offset solution: consists of computing the offset of the part surface. All the offset-surface intersection curves[7] as well as self-intersection curves are computed, and then they are linked to perform pencil curves. These pencil curves are usually needed to finish the concave parts on the part surface using a little-end ball tool. Offsetting must be done twice for torus mill tool paths (two radiuses implied).

Normal vector compensation: Each part surface point to be mechanised is compensated in two directions: first, normal to the surface for the torus minor radius, and then compensated in the tool attack plane for the torus major radius. Again it is necessary to obtain intersection between tool path lines in order to avoid collision with the part surface.

Virtual digitising: Centre tool points are obtained by virtually touching the object to mechanise. This algorithm, typically used to compute pencil curve tracing[9], is used here for imitating the way of work of traditional shoe last 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

Most of these methods are currently implemented by general-purpose CAD/CAM successfully[1,2,3,4,5,6]. All of them are able to generate trajectories for computer numerally controlled (CNC) machines. The problem gets bigger when it is necessary to use a particular machine that does not fulfil the CNC standards. This is the case of traditional copiers for shoe lasts.

Virtual digitising for shoe last turning lathe machines

Definition of the tool motion

One of the main concepts to be considered in virtual digitising, consists of defining the motion of the tool in order to mechanise the object. For a turning lathe machine, this movement is well defined. There are three axes that produce a spiral movement. Let call *X* to the translation axis, *Y* to the tool attack axis and *Z* to the rotation axis. The rotation speed for translation and rotation axis is constant, as a result, the tool arm trajectory

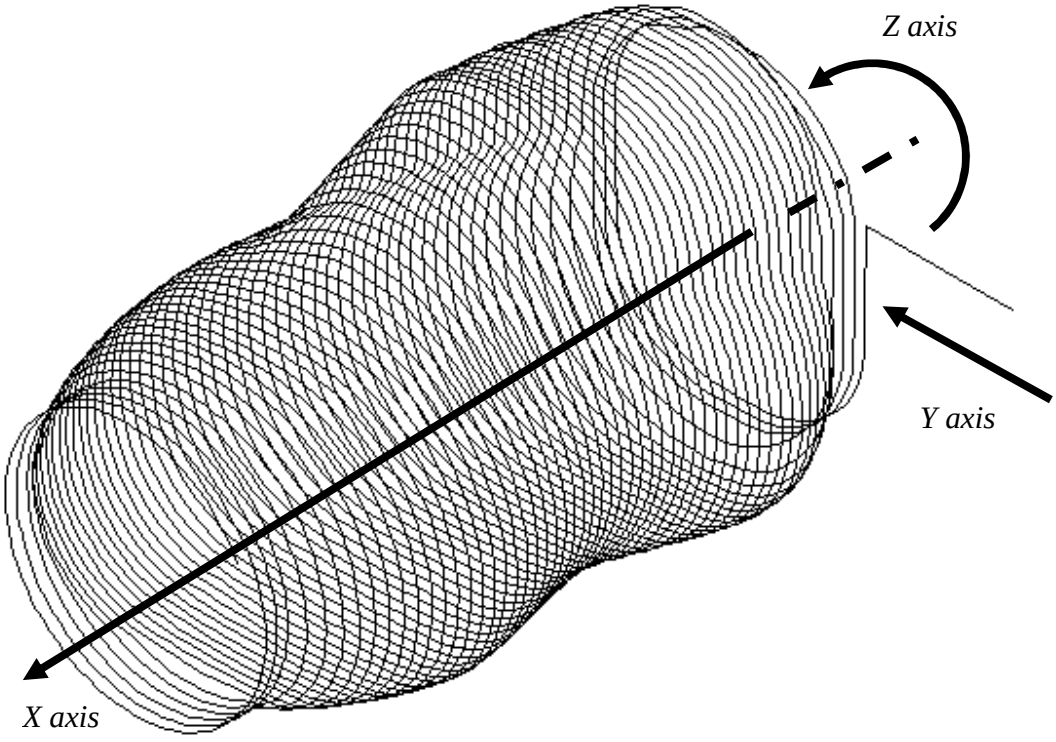

Figure 2 Torus mill tool path for a woman shoe last

makes a spiral of a determined step. For each point of this spiral the tool attack axis is moved in order to reach the surface point to mechanise. Figure 2 shows a typical tool path for a shoe last copier machine:

Obtaining a discrete model

In order to obtain a discrete model of the part surface, it is necessary to obtain a grid for each free-form surface of the part surface. Let suppose that every surface is normalised for each parametric direction *u* and *v*.

Let *PS* the part surface defined as a set of *n* free-from surfaces (NURBS):

$$PS_n = \{r_i(u,v) : i \in [1..n], u \in [0..1], v \in [0..1]\} \qquad 1)$$

Let D*S* the domain of a grid for a surface defined as follows:

$$\begin{aligned} &DS = \{DS_i : i \in [1..n]\} \\ &DS_i = \{(x_j, y_k) : j \in [0..m_i - 1], k \in [0..s_i - 1]\} \\ &x_j = j.\Delta_j : \Delta_j = \frac{1}{m_i - 1}, y_k = k.\Delta_k : \Delta_k = \frac{1}{s_i - 1} \\ &m_i, s_i \in \mathrm{N}^+ \end{aligned} \qquad 2)$$

Finally, let define a grid, *GS* as:

$$\begin{aligned} &GS = \{GS_i : i \in [1..n]\} \\ &GS_i = \{r_i(u,v) : (u,v) \in DS_i\} \end{aligned} \qquad 3)$$

Obtaining a discrete model for computing the tool path suppose precision loose, in next section the method accuracy will be analysed. On the other hand, the algorithm becomes simpler and faster when a discrete model is used. As show in Equation 2, a discrete model of more or less definition can represent every surface. The proper m_i and s_i values will depend on the surface complexity.

Simulating the tool motion

Next step consists of making a virtual model for the toroidal tool. For simplicity, the cutting wheel is considered as a infinite stack of circles. The 3D disk is defined as follows:

Let *R* and *r* the major and minor tool radius respectively, C the tool centre point and *v* a 3D orientation vector. For simplicity let assume *v* as the Cartesian *X*-axis and *C* the origin. (It is always possible to find a 3D-transformation matrix that translates any director vector *v* to the *X*-axis and any centre point to the Cartesian origin).

Then:

$$\begin{aligned} &T(x,y,z) \equiv z^2 + y^2 = m(x)^2 \\ &m(x) = R + \sqrt{r^2 - x^2} \\ &\forall x \in [-r, r], \forall y, z \in [-m(x), m(x)] \end{aligned} \qquad 4)$$

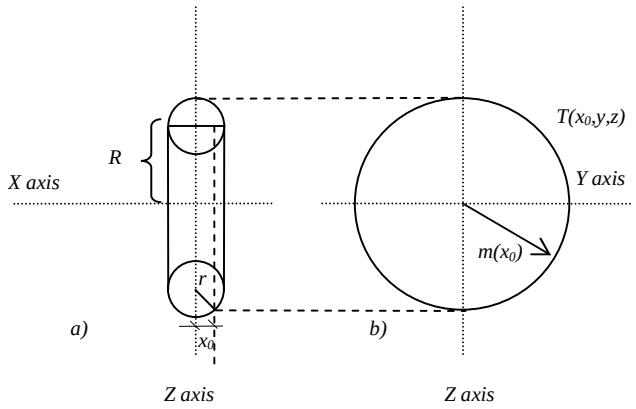

Figure 3 a) Front and b) side view of a modelled toroidal tool

For every simulation step the tool is not moved, the surface is transformed to simulate the milling process. In this case the surface is moved along the rotation axis and rotated in order to simulate a spiral movement along the rotation axis.

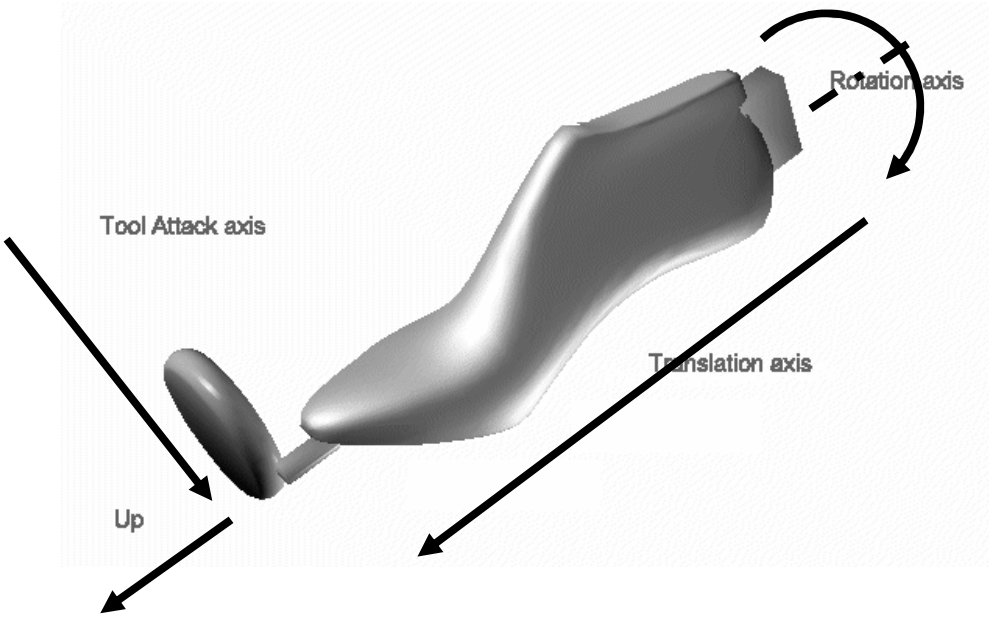

Figure 4 Axis implied on a shoe last turning lathe machine

Digitalisation process

The digitalisation 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 (*Y*). 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 offseting* method[10] and the *direct cutting* simulation[11,12].

$$[x', y', z', 1] = [x, y, z, 1]\ TR_{4x4}$$

$$D(x', y', z') = \begin{cases} |y'| & \forall x' \in [-r, r], \forall y', z' \in [-m(x'), m(x')] \\ +\infty & other \end{cases} \qquad 5)$$

A pseudo code algorithm is presented below:

```
For every trajectory position trpos do
 Min_distance=∞
 For every surface PSi in PSn do
  For every grid point pgj,k in GSi do
    pg'j,k= pgj,k * TR4x4
    Current_distance=D(pg'j,k)
    If Current_distance<Min_distance
    then Min_distance=Current_distance
    Endif
  Endfor
 Endfor
 Tool_centre=Get_centre_point(MinDistance,trpos,TR4x4)
 Add_trajectory(Tool_centre)
EndFor
```

Algorithm 1 Simple virtual digitising algorithm

In order to find the nearest point from the grid to the tool in the attack axis direction, it is necessary to use the Equation *5*. This distance is computed by projection of the grid point on the tool, in the tool attack direction. The distance between the given point and the projected one will be used to compute the tool centre point for that machining position.

Next figure shows a simple example in 2D in order to obtain a single trajectory point for a circular tool.

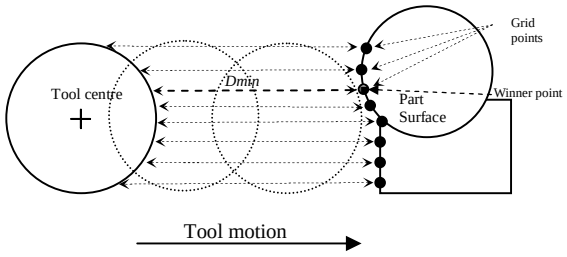

Figure 5 Virtual digitalisation process example for 2D shapes

As shown above, the minimum distance represents the tool centre distance in order to reach the grid point without collision with the shape.

Error analysis

The accuracy of the virtual digitised method is analysed in this section in terms of maximum distance between the part surface and the mechanised one.

Due to the fact a discrete model is being used for modelling the part surface, there exits an implicit error joined to the election of the grid points. There are several methods in literature[12] used to obtain a grid from a NURB given a maximum error parameter. For this reason this kind of error is not going to be analysed in this section although is going to be taken into account.

Every point in the grid is associated with four neighbours in the grid, one pair for each surface direction. When the digitalisation process is carried out, the virtual tool can enter into the gaps existing between associated neighbours. The error will depend on both the tool radius and the gap length, that is, the distance between neighbours. Figure 6 shows the maximum error produced between a couple of neighbours.

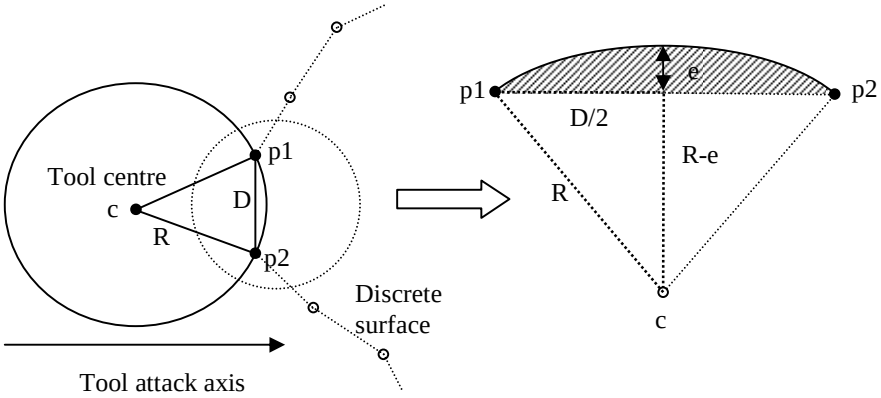

Figure 6 Gouging produced by the torus tool in a discrete model surface

$$R^2 = \left(\frac{D}{2}\right)^2 + (R-e)^2 \rightarrow e = R - \sqrt{R^2 - \left(\frac{D}{2}\right)^2}, R \geq \frac{D}{2} \qquad 6)$$

For example, using a toroidal cutting wheel with major radius of 40mm and a grid of point with a maximum gap of 2 mm, we can obtain a maximum error of 0.0125 mm.

Algorithm cost

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

Problem size

Analysing algorithm, it is possible to observe up to three nested loops. One of them, the most internal one, it 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).

$$n = max(m \cdot s, \forall m, s \in DS) \qquad 7)$$

Let assume *n* as the maximum number of grid points of all the surfaces in *PS (see expression 2).*

The second loop iterates on every surface in the part surface. This number is inappreciable related to *n*, for this reason, is not going to be taken into account to compute an upper limit. In this way *n* will be define the problem size for computational cost analysis in next section.

Computational Cost

We analyse the algorithm from the most internal loops to the most internal ones in order to obtain an upper limit for the computational cost.

The third loop (the most internal) it is repeated for *n* times. Every time, a product *vector x matrix* it is computed and stored in a local variable. We call this cost *ct31*, this cost can be considered constant since it does not depend on the problem size. After the product is made, we apply the formula expressed in *5)* to carry out the digitalisation process. We call *ct32* to the cost to do this operation. As the previous one, we can consider it constant. Finally, a comparison and an assignment is done. In the worst case the assignment is always done. We summarise the comparison and the assignment cost with the constant *ct33*.

Next expression evaluates the cost of the third loop:

$$\lim_{n\to\infty}\left(n \cdot (ct31 + ct32 + ct33)\right) = \mathrm{O}(n) \qquad 8)$$

$$ct2 \cdot \mathrm{O}(n)$$

Next loop, the second one repeats the third one for every surface in PS. Lets call *ct2* as the number of surfaces that belong to the part surface. Its cost is expressed as:

Finally, first loop operates on trajectory positions, as commented above, this number is close to *n* in order to obtain good milling results. In this loop an assignment is done with a constant cost called *ct1*. The function *Get_centre_point* obtains the centre tool point given a tool position and a transformation matrix of 4x4 elements. This function uses a couple of trigonometric operations and a *vector x matrix* multiplying. It can be considered constant related to *n* and called *ct3*. The final *Add_trajectory* function consists of adding result points to a list, its computational cost can be considered constant and called *ct4*.

The cost of the first loop, and consequently the algorithm cost, is:

$$\lim_{n\to\infty}\left(n \cdot \left(ct1 + ct2 \cdot \mathrm{O}(n) + ct3 + ct4\right)\right) = \mathrm{O}(n^2) \qquad 9)$$

As a guide, a usual value for *n* in shoe last machining is about twenty thousand, that is, a high computer cost, in next sections we will show some quantitative examples with time measures.

Optimisations

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 we can not reduce the Omega cost of the algorithm, but we should reduce the average response time for the machining.

The basic idea for the reduction is the *local* principle. As mentioned before, the algorithm proposes to go through the part surface "touching" it and reporting the tool centre point for every nearest point of the trajectory. In every trajectory step of the algorithm, a distance "D" formula is applied for all grid points belonging to the part surface. However, due to the physics of tool motion some points can not to be touched for a specific trajectory point. For this reason, in every trajectory step, we will analyse only the grid points that the tool can touch, i.e. the D formula does not return infinity.

Another related idea to the local principle is does not to compute those points that are not faced to the tool, since we are working with solid surfaces it has no sense touching a point from the back part. We should determine which points are faced off the tool and eliminate them before computing the distance formula (this step will be eliminated approximately the 50% of the points to be analysed).

Going forward the local principle, since surfaces are continuous, we should conclude that for consecutive trajectory points, the nearest touching points are nearly the same. We should define a delta distance (defined in grid co-ordinates) that assures that the nearest point is included in.

Lets define delta "δ" as the maximum distance between neighbours (vicinity range), and LS_i the subset of neighbours for the grid point u,v:

Then, the algorithm becomes to:

$$LS \subset GS$$
$$LS_{i,u,v} = \left\{ r_i(j,k) : (j,k) \in DS_i, \left| j-u \right| < \delta u, \left| k-v \right| < \delta v \right\} \qquad 10)$$

```
For every trajectory position trpos do
 Min_distance=∞
 For every surface PSi in PSn do
   LSi=ObtainSubGrid(Gsi,LSi-1, δ)
   For every grid point pgj,k in LSi do
     pg'j,k= pgj,k * TR4x4
     Current_distance=D(pg'j,k)
     If Current_distance<Min_distance
     then Min_distance=Current_distance
     Endif
   Endfor
 Endfor
 Tool_centre=Get_centre_point(MinDistance,trpos,TR4x4)
 Add_trajectory(Tool_centre)
EndFor
```

Algorithm 2 Optimised virtual digitising algorithm

Due to the fact LS is much less than GS the average computational time is reduced a lot. Notice that a new function is added to the pseudo-code: "*ObtainSubGrid*". This function chooses a new local region in order to

compute the minimum distance. From the previous local region (taken into account the winner grid points), the new region is computed by adding the δ factor to these winner points.

Choosing the delta factor

Delta factor determines the continuity "strength" for a surface. A big delta factor means the grid is very irregular along every direction, e.g. a tridimensional star, on the opposite, a small factor means that the grid is very regular for every direction, e.g. plane. Given from experience, for traditional turning lathes for shoe last, this factor is about 20 from a range of 120 points in the v direction and 8 from a range of 150 in the u direction. It is possible to use artificial intelligence algorithms to determine this factor for every free surface. The problem is the efficiency, the more complex obtaining the delta factor is, the less computational reduction you get. Usually, using this factor is good for a subset of prototype surfaces (for example if we mechanise always shoe last or heels), and the time reduction you may obtain is about 90%.

Computer architectures for virtual digitising optimisation

Due to the fact of the local principle, it is possible to use a parallel computer architecture to make the computation in an efficient way. Let suppose *n* processors (e.g.: transputers, digital singal 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.

Next figure 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.

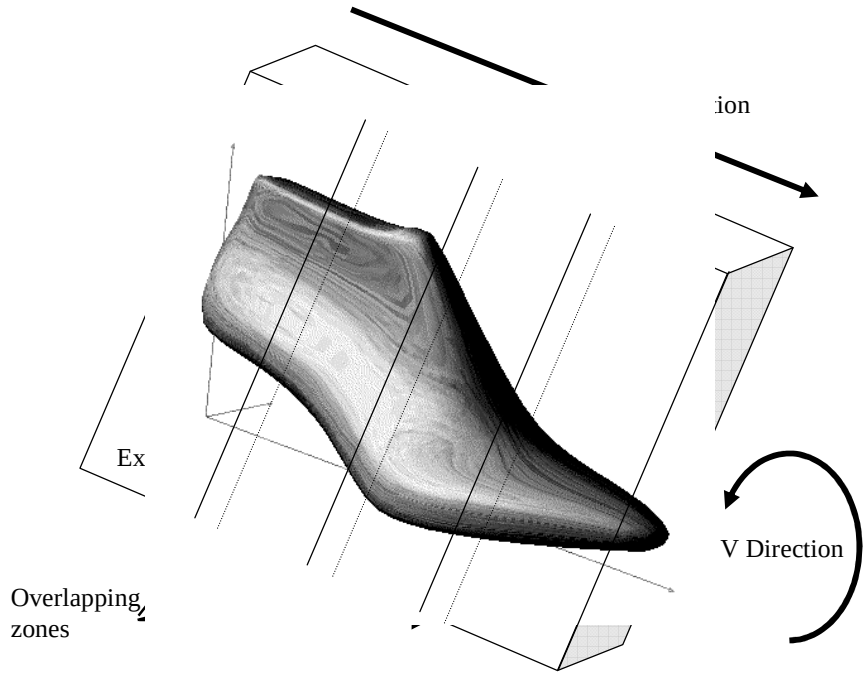

Figure 7 Surface splitting in direction *u* for 4 processors

In this case we have used only the *u* direction for the extent distribution, since the tool geometry makes the overlapping smaller in this direction. For another tool geometry (spherical, cylindrical, conical) it is possible to distribute extents in both directions with a minimum overlapping.

Two different architectures are proposed in figure 8 and figure 9. In the first one there is no communication between neighbour processors, the master processor distribute each overlapped extent to each processor, so overlapped information is twice in neighbours. On the other hand, using architecture shown in figure y the master processor distribute 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, on the contrary, in second one, overlapped information distribution is carried out by every processor in parallel.

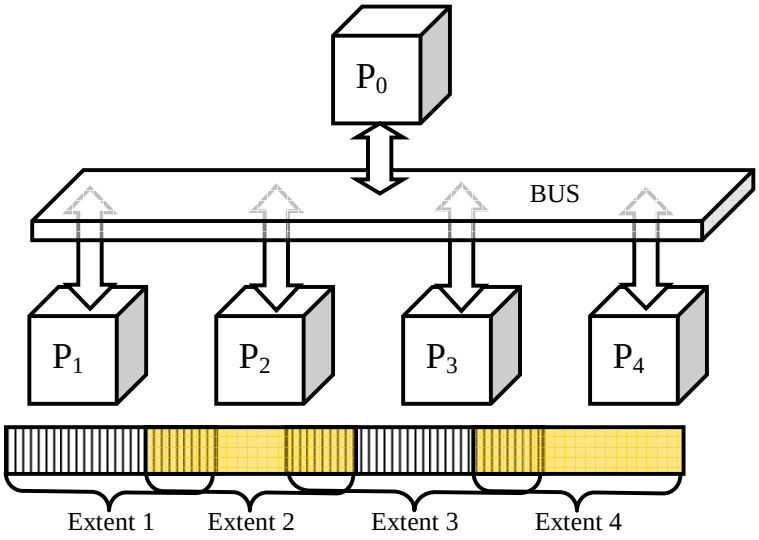

Figure 8 Multiprocessor architecture without communication between neighbours

Theoretically computation cost becomes from *9)* expression to:

$$m = n / p + o$$

$$\lim_{n\to\infty}\left(m \cdot \left(ct1 + ct2 \cdot \mathrm{O}(m) + ct3 + ct4\right)\right) = \mathrm{O}(m^2) \qquad 10)$$

Where *p* is the number of processors and *o* is the cost associated to the communication time.

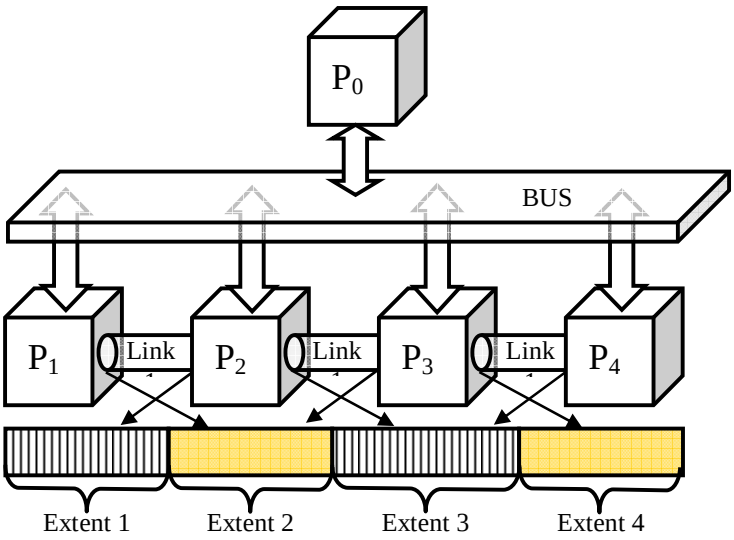

Figure 9 Multiprocessor architecture using communication between neighbour processors

Illustrative examples

In this section a series of experimental results are showed. These tests were realised using the commercial Forma3D® system for design and machining of shoe lasts. This CAD/CAM program uses the algorithm proposed in this article and takes the advantages of the local optimisation we proposed in the previous section.

Next table shows the average response time in seconds for some shoe last machining tests. The trials were carried out using a personal computer (PC), Intel Pentium II® processor at 233MHz under Windows 98® operative system. The maximum grid spacing was 2mm, and the grid density is specified for every direction (Gu and Gv). Chosen delta values were 8 and 20 for u and v direction respectively.

The trajectory is defined by the user, specifying the spiral step (in millimetres) and the number of trajectory points per spiral revolution. Total number of point trajectory is obtained by just multiplying the points per revolution and the number of rounds on the last (total length of the model divided by the spiral step). In these tests we took into account 3 different surfaces: one for the last and two more for the back and fore holders. The parameters for the holders were the same for all the tests.

Experiment 1: Delta factors 8 and 20

Gu	**Gv**	**Points per revolution**	**Spiral Step (mm.)**	**Time (s.)**
159	120	360	2	215
137	90	120	1	100
140	100	90	5	5

Experiment 2: None optimisation

Gu	**Gv**	**Points per revolution**	**Spiral Step (mm.)**	**Time (s.)**
159	120	360	2	2315
137	90	120	1	912
140	100	90	5	42

As a result, it is possible to observe that the average response time of the algorithm depends mainly on the number of trajectory points to be obtained and not too much on the number of grid points (this is the effect of the local optimisation).

Figure 10 shows a step in the machining simulation taken from the Forma3D® simulation system.

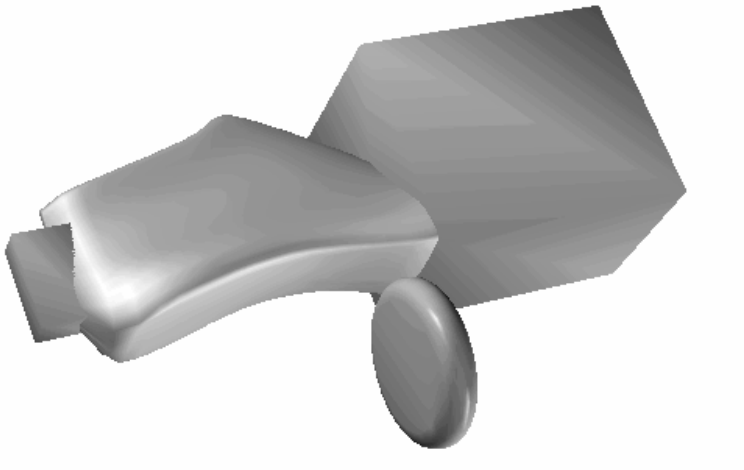

Figure 10 Shoe last machining simulation by Forma3D®

Conclusions and Discussions

Traditional processes used in shoe last manufacturing are simple, robust and efficient. The algorithm presented in the paper copies those processes and takes its advantages using virtual digitising. The results are good and allow introducing CAD/CAM systems in the shoe last sector. The procedure of virtual digitising it is simple to implement, offers good results and avoid the problem of tool collision by its own definition. On the other hand, this algorithm becomes efficient when we can assure a good vicinity factor in the surface to machining and/or we have a subset of prototypes to mechanise. For this reason, the algorithm is not suitable for general purpose machining algorithms since it is too slow versus another types of tool path generation algorithms. The algorithm needs to know the generic tool motion around the surface (spiral, zigzag, steps) in order to compute the tool path, so that is not suitable to compute automatically tools strategies (this kind of problems may be solved in a previous phase by another algorithms and the passed to the virtual digitising algorithm).

Another advantage of the algorithm presented is the paralellisation. Due to fact, we dispose of a grid of points along a surface and a set of trajectory positions, it is possible to distribute the computation among a series of processors reducing the computing time of the algorithm. Every surface can be subdivided in regions and each processor may be able to compute the distance ‘D’ formula for a region of the surface, finally join all sub-trajectories we obtain the resultant tool path. Paralellisation is not tested in the architectures proposed in the article but it is an interesting start point for future studies.

The weak point of the algorithm presented is the election of delta factor, with a high factor we loose efficiency, on the other hand, we a very low delta factor we can fail the election of the winner grid point. At the present time, we use factors given from experience for a reduced set of prototype surfaces. It would be interesting to obtain automatically the optimum factor for every surface. These kinds of optimisation problems are solved usually via the Artificial Intelligence algorithms and will allow the generalisation of the algorithm.

Acknowledgements

We would thank to the shoe last Spanish manufacturers 'Hormas Idella', 'Hormas Beneit' and 'Hormas Maestre' for their advice and attention.

References

1. *TEBIS- The Complete Solution for Tool, Die, Mold, and Pattern Manufacturing*, Tebis Technishe Informationssysteme, Germany (1994)
2. *Work-NC User Guide*, Sescoi, France (1993)
3. *Clicks User Manual*, ARGO Tech., Japan (1990)
4. *Z-Master Reference Manual*, Cubic Tek, Korea (1992)
5. Ian Paris. *"Shoemaster – A CADCAM System for the Design and Development of Pedorthics"*. International Symposium of CAD/CAM Systems in Pedorthics, Prosthetics & Orthotics, Nuremberg, May 1997.
6. Jimeno A, García J., Salas F. *"Shoe lasts and computer systems"*. (Spanish) Tecnología del Calzado. November 1999.
7. Wang, Y 'Intersection of offsets of parametric surfaces' *Computer Aided Geometric Design*, Vol 13 (1996), pp.453-465
9. Jung W Park *et al* **"**Pencil Curve Tracing via Virtual Digitizing" *Proc. of IFIP CAPE Conference*, (1991), pp.97-104
10. Takeuchi, Y *et al.* 'Development of a personal CAD/CAM system for mold manufacturing' *Annals of CIRP*, Vol 38 No 1 (1989), pp.429-432
11. Jerard, R B, Drysdale, R L and Hauck, K 'Geometric simulation of numerically controlled machining' *Proc. of ASME Int'l Conf. on Computers in Engineering*, ASME, New York (1988), pp.129-136
12. Farin, G., "*Curves and surfaces for CAGD*", Academic Press. 1993
13. Faux, I D and Pratt, M J *Computational Geometry for Design and Manufacture*, Ellis Horwood (1980)