The recognition of the signature of a polyline by the Gauss map

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

Abstract

In this paper we propose a new method : the V_k *vector method* is a necessary step to evaluate the required resources during a simulation, and then to adapt these resources to the performances expected by the users. This approach is based on a data simulation pretreatment, especially on the recognition of the contour of a polyline thanks to the Gauss map [1].

Keywords : contour recognition, Gauss map, computer graphics, NC machining simulation.

[1]This research was partly funded by the transfer of the rights of the NC simulation software called LI-CN (Limoges University registered trademark).

1 Introduction

Nowadays, in the impatient world of the workshop (as in that of video games), it becomes necessary in an interactive graphics software to estimate a maximum execution time in order to adapt the resolution and the point of view according to the performances expected and acceptable by the users.

A simulation is more and more required for the verification of the Numerical Control programs for CNC machining. These programs must also be validated and optimized off line, "close enough" to the machine on standard computers to be able to profit from the competences of the operators. This simulation should be as simple and user-friendly as possible to allow an easy intervention of the skilled operator, that is why the execution time must be fast. Thanks to the simulation, we do not want to design a part, but indeed to obtain a realistic view of the effect of several thousand tool paths. The study of the data-processing constraints is essential in order to take into account the requirements of the workshop. In computer graphics, several solid modeling approaches of physical objects have already been reported. The direct solid modeling is typically implemented by using CSG or Brep. This approach is theoretically capable of providing accurate milling simulation. However, it is computationally expensive (at least an $O(n^3)$ execution time [1]), especially for representing the parts with sculptured surfaces. We also chose one of the simple decomposition modeling methods : the extended z-buffer technique wherein a solid is decomposed in image space into dexels [2]. The different stages of a machining simulation can be visualized step by step by this modeling technique which is robust and little time-consuming ($O(n)$). Even if the user-defined accuracy of decomposition depends on the chosen point of view and on the size of the dexel, this technique supports efficient and fast model updating operations for a great number of tool paths, but it is memory expensive[2]. To improve the interactivity, we already introduced in [4] new functionalities called *traces* used to "glue" together the virtual part and the virtual chips memorized and obtained during the simulation. It is also possible to come back to any specific moment of the simulation.

With an NC zigzag tool paths program of more than 17,000 trajectories simulated on an 800 by 600 pixels image, the used memory capacity is about 6 Mo for the dexels of the extended z-buffer and about 51 Mo for the elements of the traces. For an image of a given size, the number of elements in image space varies according to several parameters : some are easily known like the chosen point of view and the number of tools paths, the others must be identified like the used machining strategy. Evaluating the resources used during the simulation (memory capacity and execution time) can only be done if the tool paths strategy chosen in Design Department[3] is identified from data contained in NC program files. The recognition of the tool paths strategy must be thus considered as a necessary stage in the estimation of the number of dexels and elements of traces. Thanks to the latter, it will be possible to check that the simulation is quite compatible with the constraints of the user : on the one hand, by evaluating the optimal resolution for a given point of view and, on the other hand, by evaluating the maximum execution time to visualize the simulation under these conditions.

2 The problem presentation

2.1 Hypotheses

Estimating the length of a curve γ from the expected number of intersections between any straight line and γ can be calculated with a well known result of integral geometry [6]. Let us indicate by $M(\epsilon, \theta)$ the number of

[2]Research is still led in the field of machining simulation [1] and computer graphics [3].

[3]Two major strategies being the zigzag and contour-parallel; a recent survey about different tool paths strategies is presented in [5].

points of intersection between γ and some direction parallel straight lines uniformly spaced from ϵ with $\theta + \frac{\pi}{2}$ as orientation; then $\epsilon M(\epsilon, \theta)$ gives an estimate of the measurement of the projection of γ on one straight line of direction θ where each point is taken into account with its multiplicity. Computing the number of points of intersection between parallel rays and tool paths and applying the following formula : $M(\epsilon, \theta) = \frac{Projected\ length\ of\ \gamma}{\epsilon}$ is not appropriate to estimate the number of dexels obtained during a machining simulation . Indeed, the sequence of the trajectories must be considered to determine if the new calculated points of intersection (corresponding to dexels) must be taken into account. Thus, the recognition of the machining simulation is an essential stage.

We assume the studied machining sequence as a monotonous area (with only one strategy) and we suppose that all the trajectories are linearized and defined by the Cartesian co-ordinates of their starting point and of their point of destination. On the one hand, a zigzag machining is based on the following ideal pattern (figure 2.1) : two big trajectories opposite π, where length depends on the outline of the part, and two small trajectories which follow the outline of the part and have a length lower than $2 \times (tool\ radius)$. They allow the tool to move from a forward to a backward path. On the other hand, the repetitive contour-parallel tool path pattern (figure 2.2) cannot be planned in advance because it is derived from the boundary of the concerned machining region. In such a machining, the trajectories follow the outline of the part by offset segments which are parallel and spaced equally between trajectories.

The most simple surfaces encountered today on the mechanical part (such as prismatic shapes) are machined by moving the center of the tool on a plane (a $2D\frac{1}{2}$ machining). For sculptured surfaces, at least interpolated 3-axes must be simultaneous used by the tool. The geometrical representation of these surfaces is based on a mesh composed of a set of squares which consists in Bézier or NURBS functions. The tool paths are then defined by isoparametric curves which are approached by small linear segments. For a zigzag machining for example, the ideal pattern is preserved even if it is composed of a multitude of trajectories.

To have a robust method both for $2D\frac{1}{2}$ and for 3D machining, we decided to recognize the *signature* of the machining (zigzag or contour-parallel) by studying the curvature of the polyline defined by the sequence of trajectories. We also introduced a method based on *Gauss map* which is able to analyze the traversal direction of the successive trajectories. This method, which we called the $\overrightarrow{V_k}$ vector method, combines robustness and speed ($O(n)$ execution time) and it is independent of the length of trajectories.

2.2 The Gauss map

To visualize the *signature* of the machining program, we use the *Gauss map* [7]. It can be defined by associating with each oriented curve point the tip of its unit normal on the unit circle. Let us pick an orientation for the curve γ (figure 1), and associate with every point P on γ, the point Q on the unit circle S^1 where the tip of the associated normal vector meets the circle. The corresponding mapping from γ to S^1 is the Gauss map associated with γ. As P' approaches P on the γ curve, so does the Gaussian image Q' of P' approach the Gaussian image Q of P. The Gauss map could have been defined just as well by associating with each curve point P the tip of its unit tangent S^1. The two representations are only equivalent in the case of planar curves, the situation will be different with twisted curves and surfaces.

The Gauss map can also provide a study of the curvature. A conventional sign can be chosen for the curvature at every point of a plane curve : the curvature will be positive when the center of curvature lies on the same side of γ as the tip of the oriented normal vector, and negative when these points lie on opposite sides of γ. Thus, reversing the orientation of a curve also reverses the sign of its curvature. Let us pick a point moving along the γ curve,

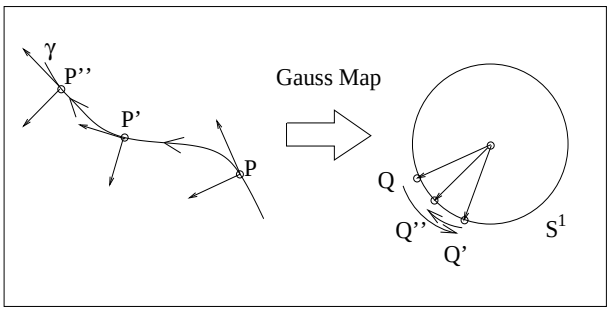

FIG. 1 – The Gauss map

the sense of the rotation given by the motion of the Gaussian image Q on S^1 is reversed when the curvature is changed (figure 1). The curvature of a curve can also be identified by comparing the sense of the oriented curve arc obtained from successive Gaussian images corresponding to the different points of the initial curve. So, when a curve is transformed in a polyline, there is a correlation between the curve and the traversal direction of the trajectories. The curvature is defined as positive for a anti-clockwise-oriented polyline and negative for an clockwise-oriented polyline.

3 The V_k vector method

3.1 Introduction to the V_k vector

For the recognition of the machining strategy, we chose to represent on the Gauss map the $\overrightarrow{V_k}$ vector resulting from the vectorial sum of all the previous trajectories. Each i^{th} trajectory is represented by a $\overrightarrow{v_i}$ vector with l_i as length and α_i as orientation (where $0 \leq \alpha_i \leq 2\pi$ is the angle between the trajectory and the X axis). Let us define the $\overrightarrow{V_k}$ vector : the origin of the $\overrightarrow{V_k}$ vector is the starting point of the first trajectory ($\overrightarrow{v_1}$) and the tip of the $\overrightarrow{V_k}$ vector is the point of destination of the k^{th} trajectory ($\overrightarrow{v_k}$). The $\overrightarrow{V_k}$ vector is also defined by : $\overrightarrow{V_k} = \sum_{i=1}^{k} \overrightarrow{v_i}$. Thus, the $\overrightarrow{V_k}$ vector can be identified by its length (L_k) and its orientation angle (θ_k : angle between the $\overrightarrow{V_k}$ vector and the X axis). It is also possible to define the angle and the angular function of each $\overrightarrow{V_k}$ vector according to time. We move along the trajectories of the machining sequence with a constant speed, that is why it could be said that the final point of the i^{th} trajectory is reached in a time t_k, which also corresponds to the generation of a new $\overrightarrow{V_k}$ vector. Thus, we call $\theta(t_k)$ the angle of the $\overrightarrow{V_k}$ vector, $\varphi(t_k)$ the *cumulative angular function* of the $\overrightarrow{V_k}$ vector (defined by Otterloo [8]) and $\alpha(t_{i-1})$ the angle of the $\overrightarrow{v_i}$ trajectory starting at t_{i-1}, the $\overrightarrow{v_i}$ trajectory being defined on the time interval $[t_{i-1}, t_i[$.

To recognize the *signature* of the machining program, the motion of the Gaussian image of the $\overrightarrow{V_k}$ vector must be analyzed (figure 2) :

- In a zigzag sequence, the Gaussian image of the $\overrightarrow{V_k}$ vector does a backward and forward motion only on a part of the unit circle, but never all around the circle. This backward and forward motion corresponds to the change of the traversal direction of the trajectories, i.e the change of the curvature from a forward path to a backward path.
- In a parallel-contour sequence, the Gaussian image of the $\overrightarrow{V_k}$ vector rotates all around the unit circle only in one sense. The sign of the curvature is also constant during the whole sequence of trajectories.

Using the Gauss map, by studying the sense of the circle's arcs described by the tip of the Gaussian image of the $\overrightarrow{V_k}$ vector, amounts to identifying the strategy used for the machining program.
That is why the evolution of θ_k (angle of the

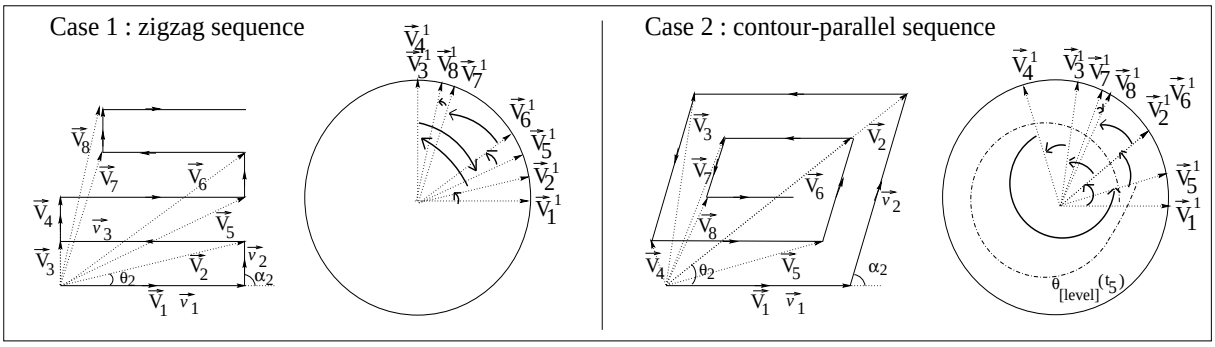

FIG. 2 – Evolution of the V_k vector during two machining sequences using different strategies

$\overrightarrow{V_k}$ vector) should be represented as follows :

$$\theta_{[level]}(t_k) = \theta_{[-\pi;\pi]}(t_k) \ + \ n \times \ 2\pi$$

where n indicates how many times the $\overrightarrow{V_k}$ vector rotates around the unit circle.

- if $n \neq 0$ then $\theta_{max[level]} < 2\pi$ or $\theta_{max[level]} < -2\pi$, the $\overrightarrow{V_k}$ vector rotates at least one time all around the circle : a contour-parallel machining is also identified.
- if $n = 0$ then a zigzag machining is identified.

We call n the *level* of the contour and n is defined by :

$n = \lfloor \frac{\sum_{i=1}^{k-1} \Delta\varphi(t_i)}{2\pi} \rfloor$ where $\lfloor \ X \ \rfloor$ gives the integer part of X.

n is calculated from the cumulative angular function.

3.2 The cumulative angular function

The cumulative angular function is defined by Otterloo [8] from the sum of the angles between two successive trajectories : $\Delta\alpha(t_i) = \alpha(t_i) - \alpha(t_{i-1})$. Each term $\Delta\alpha(t_i)$ is converted into $\Delta\varphi(t_i)$ thanks to the following inequality : $-\pi \ \leq \Delta\varphi(t_i) \ \leq \pi$:

$$\Delta\varphi(t_i) = \begin{cases} \Delta\alpha(t_i) + 2\pi & \text{for} \quad -2\pi \ < \Delta\alpha(t_i) < \ -\pi \\ \Delta\alpha(t_i) & \text{for} \quad -\pi \ \leq \Delta\alpha(t_i) \leq \ \pi \\ \Delta\alpha(t_i) - 2\pi & \text{for} \quad \pi \ < \Delta\alpha(t_i) < \ 2\pi \end{cases}$$

$\Delta\varphi(t_i)$ is positive for convex angles and negative for concave angles. The cumulative angular function $\varphi(t_k)$ is also equivalent to :

$$\varphi(t_k) = \sum_{i=1}^{k-1} \Delta\varphi(t_i) \text{ with } k > 1$$

The cumulative angular function is used by Otterloo to identify a simple closed curve (such as a polygon) by studying its outline. The parametric representation of such a curve is a periodic function :

$$z(t_k) = z(t_k + n \times 2\pi)$$
$$\text{for } t_k \in [0, 2\pi] \text{ and } n \in Z$$

Calculating and following the evolution of the cumulative angular function along a certain curve is possible by studying the constant speed motion of a point on this curve. Thanks to the periodicity of the parametric representation, the cumulative angular function must also become a periodic function given by :

$$\varphi(t_k + 2\pi) = \varphi(t_k) + 2\pi \text{ for } t_k \in [0, 2\pi]$$

The $\varphi(t_k)$ cumulative angular function is equal to 2π at the end of the traversal of a simple closed polygon defined by an anti-clockwise-oriented contour and $\varphi(t_k) = -2\pi$ at the end of a clockwise-oriented contour. During the traversal of a simple closed anti-clockwise oriented contour, we have : $0 \leq \varphi(t_k) < 2\pi$.

We suppose that a curve γ is an anti-clockwise oriented curve (or curve oriented in the positive sense) if the "inside" of γ (the area delimited by γ) is always on the left of γ by moving along the curve. So, if two successive trajectories of a polyline are anti clockwise oriented, then $\Delta\varphi(t_k) > 0$ and the $\varphi(t_k)$ cumulative angular function increases. If two

successive trajectories of a polyline are clockwise oriented, then $\Delta\varphi(t_k) < 0$ and consequently $\varphi(t_k)$ decreases.

In a contour-parallel machining sequence, the successive trajectories have always the same traversal direction. Let us suppose that this direction is anti-clockwise, $\varphi(t_k)$ is not limited by 2π. As $\Delta\varphi(t_k) > 0$, we have the following equality for each first trajectory of a new contour : $\varphi(t_k) = n \times 2\pi$ (avec $n \in N$). n can also be expressed according to the cumulative angular function :

$$n = \lfloor \frac{\sum_{i=1}^{k-1} \Delta\varphi(t_i)}{2\pi} \rfloor \text{ where } \lfloor\ X\ \rfloor \text{ gives the integer part of } X.$$

We recognize above the formula given in the previous part with the definition of n as the *level* of a contour.

In a parallel-contour sequence, the *n-level* is positive for anti-clockwise oriented trajectories and negative for the clockwise orientation. The maximum *n-level* is always superior to 1 or inferior to -1. Moreover, it has a more or less high absolute value according to the size of the part and to the uniform space between parallel trajectories. The *0-level* is the first machining contour of the part. The zigzag machining takes place only on this contour and so uses only a 0-level.
The position of the $\overrightarrow{V_k}$vector compared with the X-axis is referenced by the $\theta(t_k)$ angle. Now, we call $\theta_{[-\pi;\pi]}(t_k)$, the $\theta(t_k)$ angle defined from the following inequality : $-\pi \leq \theta(t_k) \leq \pi$. We also introduce the $\theta_{[level]}(t_k)$ angle from the n-level, level of the contour where the k-trajectory (which is associated to the $\overrightarrow{V_k}$ vector) is led :

$$\theta_{[level]}(t_k) = \theta_{[-\pi;\pi]}(t_k) \ + \ n \times \ 2\pi$$

A contour-parallel machining program is recognized if the cumulative angular function verifies one of these inequalities : $\varphi(t_k) > 2\pi$ or $\varphi(t_k) < -2\pi$. The maximum $\theta_{[level]}(t_k)$ angle, called $\theta_{max[level]}$, is defined as : $\theta_{max[level]} > 2\pi$ or $\theta_{max[level]} < -2\pi$.

Let us calculate the cumulative angular function for a zigzag pattern, such as represented on figure 2.1 where $\alpha_4 = \alpha_2$. It is broken down into two big trajectories ($\overrightarrow{v_1}(l_1, \alpha_1)$ and $\overrightarrow{v_3}(l_3, \alpha_1 + \pi)$) and two small trajectories ($\overrightarrow{v_2}(l_1, \alpha_2)$ and $\overrightarrow{v_4}(l_4, \alpha_2)$). For two successive trajectories ($\overrightarrow{v_1}$ and $\overrightarrow{v_2}$), we have the following inequalities : $0 < \alpha_2 - \alpha_1 < \pi$ for anti-clockwise oriented trajectories and $-\pi < \alpha_2 - \alpha_1 < 0$ for clockwise oriented trajectories. Let us focus on the evolution of the cumulative angular fonction along a zigzag pattern where $0 < \alpha_2 - \alpha_1 < \pi$:

$$\Delta\alpha(t_2) = \alpha_2 - \alpha_1 :$$
$$\Delta\varphi(t_2) = \alpha_2 - \alpha_1 > 0 \rightarrow \varphi(t_2) = \alpha_2 - \alpha_1$$
$$\Delta\alpha(t_3) = (\alpha_1 + \pi) - \alpha_2 :$$
$$\Delta\varphi(t_3) = \alpha_1 - \alpha_2 + \pi > 0 \rightarrow \varphi(t_3) = \pi$$
$$\Delta\alpha(t_4) = \alpha_2 - (\alpha_1 + \pi) :$$
$$\Delta\varphi(t_4) = \alpha_2 - \alpha_1 - \pi < 0 \rightarrow \varphi(t_4) = \alpha_2 - \alpha_1$$
$$\Delta\alpha(t_5) = \alpha_1 - \alpha_2 :$$
$$\Delta\varphi(t_5) = \alpha_1 - \alpha_2 < 0 \rightarrow \varphi(t_5) = 0$$

In a zigzag anti-clockwise oriented machining, $0 \leq \varphi(t_k) \leq \pi$. As $\Delta\varphi(t_k)$ is alternately positive or negative (the sign changes by moving from a forward to a backward path), the upper limit of $\varphi(t_k)$ will be 2π.
A zigzag machining is also identified if the cumulative angular fonction is expressed by : $-\pi \leq \varphi(t_k) \leq \pi$. As in such a machining sequence, the trajectories are only led on the *0-level*, the $\theta_{[level]}(t_k)$ angle of the $\overrightarrow{V_k}$ vector is defined for any trajectory of the sequence as : $-\pi \leq \theta_{[level]}(t_k) \leq \pi$.

4 Conclusion

The $\overrightarrow{V_k}$ vector method has been introduced to recognize the machining strategy used in NC program files. It is a robust method because the length of trajectories is not taken into account and only the traversal direction of the trajectories is studied. It is also an $O(n)$ method which is little time-consuming. For the machining program composed of more than 17,000 trajectories, the execution time is about less than one milli-second. The recognition of the machining strategy takes place in the performances prediction context in order

to ensure the feasibility of a fast and user-friendly simulation. Henceforth, it is possible to estimate the memory capacity, to adapt the size of the image and to evaluate the maximum execution time for the complete simulation.

Generally, the $\overrightarrow{V_k}$ vector method is based on the Gauss map and is able to describe the contour of a polyline. This method could be applied in many other fields to evaluate the complexity of a curve by studying the features of its contour. For example, such practice could be used with the vacuum buffer method recently proposed by Popescu [9].

Références

[1] B.K. Fussell, R.B. Jerard, and J.G. Hemmet. Modeling of cutting geometry and forces for 5-axis sculptured surface machining. *Computer-Aided Design*, In Press, 2002.

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

[3] J.W. Shade, S.J. Gortler, L. He, and R. Szeliski. Layered Depth Images. *Computer Graphics*, 32 :231–242, 1998.

[4] I. Blasquez and J.-F. Poiraudeau. The Interval Treap a Complete Data Structure for the Extended Z-Buffer. *Proceedings of Spring Conference on Computer Graphics and its Applications*, pages 247–255, Mai 3-6, 2000.

[5] H.S. Choy and K.W. Chan. A corner-looping based tool path for pocket milling. *Computer-Aided Design*, In Press, 2002.

[6] C. Tricot. *Curves and fractal dimension*. Springer-Verlag, New York, 1995.

[7] D.A. Forsyth and J. Ponce. *Computer Vision : a Modern Approach*. Prentice Hall, 2002.

[8] P.J. Van Otterloo. *A Contour-Oriented Approach to Shape Analysis*. Prentice-Hall International, UK, 1991.

[9] V. Popescu and A. Lastra. The Vacuum Buffer. *Proceedings of 2001 ACM Symposium on Interactive 3D Graphics*, 2001.