Int J Adv Manuf Technol (2006) 29: 714–721
DOI 10.1007/s00170-005-2570-8

ORIGINAL ARTICLE

J.-L. Shih · S.-H. Frank Chuang

NURBS output based tool path generation for freeform pockets

Received: 22 September 2004 / Accepted: 17 January 2005 / Published online: 10 August 2005

Abstract A robust method is proposed to generate tool paths for NURBS-based machining of arbitrarily shaped freeform pockets with islands. Although the input and output are all of higher-degree NURBS curves, only one simple category of geometric entities, i.e., line segments, is required for initial offsetting and for detecting and removing self-intersecting loops. Furthermore, using those linear non-self-intersecting offsets as the legs of NURBS control polygons, NURBS-format tool paths can be smoothly reconstructed with G^1-continuity, no overcutting, no cusps, and global error control. Since all operations involved in computing tool path curves are linear geometric calculations, the method is robust and simple. Examples with integrated rough and finish cutting tool paths of pockets demonstrate the usefulness and effectiveness of this method.

Keywords Freeform pocket · NC machining · NURBS-format tool paths · Offsetting · Self-intersecting loops

1 Introduction

Pocket machining is a useful milling operation for carving 2.5D die-cavities. A pocket is bounded by several closed loops composed of a peripheral profile and an arbitrary number of island profiles. In a general form, the entities outlining the boundaries of pockets can be types of freeform curves, arcs or straight lines. However, in this study, only the pockets with freeform contours, freeform pockets as shown in Fig. 1, are considered in NURBS-based machining. Pockets can be machined using different patterns of tool paths, such as the contour-parallel offset type and the direction-parallel type. Although using the contour-parallel offset type to construct tool paths is more difficult, which requires successive offsetting and complicated intersection calculations, it is still the most popular method for machining 2.5D pocket because only one consistent milling effect, either up or down milling, occurs through the entire cutting process [1]. Therefore, the contour-parallel tool paths for pocket machining will be focused on in this article.

The tool paths of freeform pocketing must be approximated and represented by straight lines or arcs before NC machining to start running because conventional CNC machines are only capable of linear and circular arc interpolations [2–4]. With the advent of commercial CNC controllers that support NURBS output formats, a more popular cutting strategy is naturally turned to directly use interpolator's NURBS function to carry out a NURBS-based machining. As smoother, more accurate, and higher-speed cutting is concerned, undoubtedly, NURBS-based machining would be a better choice. Although the use of NURBS commands for cutting freeform objects has attracted a lot of attention in the modern manufacturing industry, it is surprising we have found no reported investigation about a total solution to freeform pocketing tool path generation.

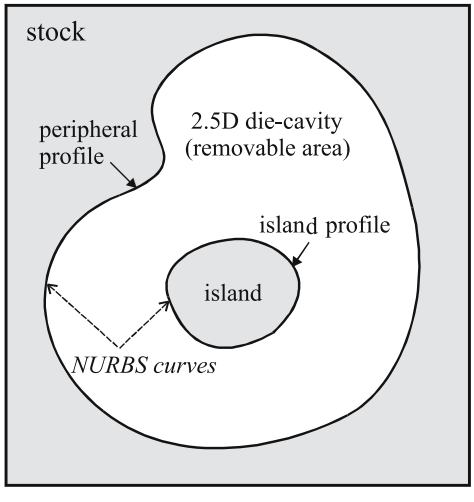

Fig. 1. Freeform pocket

J.-L. Shih · S.-H. Frank Chuang (✉)
Department of Mechanical Engineering,
National Chung-Hsing University,
250 Kuo-kuang Road, Taichung 402, Taiwan
E-mail: fzngf@nchu.edu.tw
Tel.: +886-4-22851348
Fax: +886-4-22877170

To generate tool paths for conventional pocketing, computing 2D offset curves and eliminating self-intersecting loops are generally regarded as two key issues needed to be solved [1, 5]. With NURBS-format tool paths involved, the handling of these problems becomes even more difficult. The exact offset of freeform curves in general can not be obtained analytically, using approximation methods to find the offset curves with NURBS form is inevitable [6]. Also more complicated computation of higher degree curve/curve intersections are required to detect and remove self-intersecting loops.

Some approaches have been proposed for offsetting freeform curves in the literature [7–13]. These methods to some extent can achieve the objective of offsetting, however, some problems arise when they are applied to tool path generation for freeform pocketing. The major shortcomings are described as follows. (1) Most of the reported methods cannot provide approximating offset curves with global controlled error, because they use finite sample points to estimate approximation error [7–11]. (2) Only few approaches can assure the containing maximal global error, but they encounter computation difficulty for the need of extreme point searching of higher degree rational curves or resulting poor tool paths of offset curves with very high degrees [12, 13]. (3) The numerical iteration methods, which are inefficient and prone to numerical errors, must be used to detect and remove self-intersecting loops in general [8, 13]. (4) G^0-continuous cusps, which reduce the smoothness of output tool paths, form at positions of self-intersection points when invalid loops have been removed (Fig. 2) [8].

In this research, an approach is proposed to solve the problems currently existing in NURBS-based tool path generation for freeform pockets. Based on the convex hull property of integral (or rational) Bézier curves and the idea of cumulative errors, the method can generate NURBS-format offset curves guaranteed to be within a given tolerance, that is, the offset approximation error can be globally controlled. Also, to avoid the complicated calculations of self-intersections for higher-degree curves, the method handles the invalid loop problems as early as when the initially linear approximation offsets are obtained. Using the non-self-intersecting polygonal offsets as the legs of NURBS control polygons, non-overcutting and at least G^1-continuous NURBS-format cutter center paths, which are essentially cusp-free, can be reconstructed easily. Although the input and output curves are all in higher-degree NURBS form, all the underlying entities used for geometric computations are only line segments, the proposed method is robust and efficient for NURBS-based automatic tool path generation.

With some modifications on the proposed method of G^1-continuous NURBS-format tool path generation, the continuity of tool path curves can also be extended to C^2 [14]. However, concerning the problems of computational efficiency and output data size, G^1 continuity would be more suitable than C^2 continuity for freeform pocketing. Particularly, when the pocket has islands and the curvature of boundary curves varies severely, such phenomenon would become more obvious. Hence, only the procedure for constructing G^1-continuous tool paths is introduced here.

In this paper, an additional strategy to improve the machining efficiency and to reduce the output data size for pocketing is to assign different tolerances for rough cutting and finish cutting tool paths. That is, a normal (tighter) allowance is used in the finish cutting to get the geometric detailed features of the pocket profiles, and a looser allowance is used in rough cutting to provide smoother tool paths with smaller data size output. Such that the overall efficiency for pocketing can be improved. More explanations about this issue can be reached in Sect. 5.

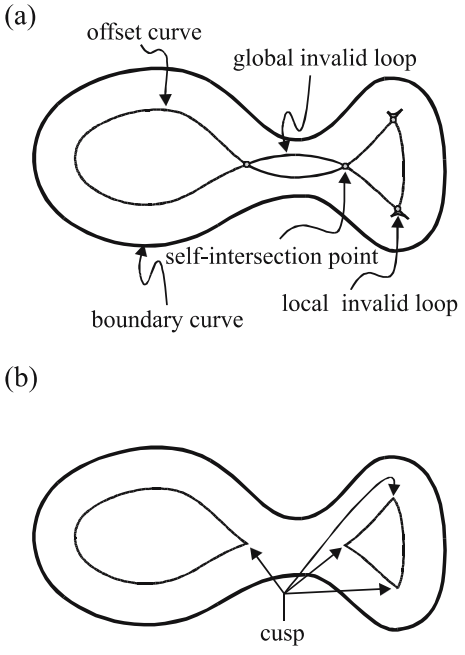

Fig. 2a,b. Offsetting freeform curve. **a** Before removing invalid loops. **b** After removing invalid loops

2 Reviewing NURBS curves

A general non-uniform rational B-spline (*NURBS*) curve of degree d with knot vector $[t_0, \ldots, t_m]$ is the curve defined on the interval $[a, b] = [t_d, t_{m-d}]$ given by

$$\mathbf{C}(t) = \frac{\sum_{i=0}^{n} w_i \mathbf{P}_i N_{i,d}(t)}{\sum_{i=0}^{n} w_i N_{i,d}(t)}, \tag{1}$$

where $\mathbf{P}_i$ are the control points, w_i are the weights, and $N_{i,d}(t)$ are the B-spline basis functions defined on the specified knot vector [15, 16]. Using homogeneous coordinates, an equivalent representation of NURBS curve is

$$\mathbf{C}^w(t) = \sum_{i=0}^{n} N_{i,d}(t) \mathbf{P}_i^w, \tag{2}$$

where $\mathbf{P}_i^w = (w_i x_i, w_i y_i, w_i z_i, w_i)$ are the weighted control points and $\mathbf{C}^w(t)$ is a polynomial B-spline curve in homogeneous space.

Appropriate choices of knot vector and control points give rise to the concepts of *open* or (closed) *periodic* NURBS. The details of mathematical description for different types of NURBS curves can be found in the literature [16]. Here, as an example, an open knot vector which yields a NURBS curve with endpoint interpolating is given as follows:

$$T = \left\{ \underbrace{t_0 = \cdots = t_d}_{d+1}, t_{d+1}, \cdots, t_n, \underbrace{t_{m-d} = \cdots = t_m}_{d+1} \right\}. \tag{3}$$

In fact, if there is no interior knot in the open knot vector, a NURBS curve can be regarded as a rational Bézier curve. That is, a NURBS representation with a knot vector of the form

$$T = \left\{ \underbrace{0, \cdots, 0}_{d+1}, \underbrace{1, \cdots, 1}_{d+1} \right\} \tag{4}$$

is a generalization of a rational Bézier representation.

Since NURBS is a subject of extensive research, a few of the fundamental geometric algorithms for operating NURBS curves have been reported in the literature [15]. One of these basic tools used in this research is the knot insertion algorithm [17]. Based on this algorithm, a NURBS curve can be decomposed into piecewise rational Bézier segments. It is worthwhile to do this, because operation on Bézier pieces tends to be simpler than on NURBS itself.

Another important tool, which can be used to evaluate the point on a Bézier curve and to subdivide a Bézier curve into two curve segments, is the de Casteljau algorithm [18]. For given control points $\mathbf{P}_0, \mathbf{P}_1, \ldots, \mathbf{P}_n$, and for a specified paramter value $t \in [0, 1]$, the de Casteljau algorithm is expressed by the recursive formula

$$\begin{cases} \mathbf{P}_{0,i} = \mathbf{P}_i, \\ \mathbf{P}_{j,i} = (1-t)\mathbf{P}_{j-1,i} + t\mathbf{P}_{j-1,i+1}, \end{cases} \tag{5}$$

for $j = 1, \ldots, n$ and $i = 0, \ldots, n-j$. Figure 3 shows the application of this algorithm on subdividing a cubic Bézier curve at the middle of the parameter domain. The result of such subdividing will generate two curve segments with smaller convex hulls than the original one. In other words, the new control polygons will approximate the Bézier curve more closely than the old control polygon.

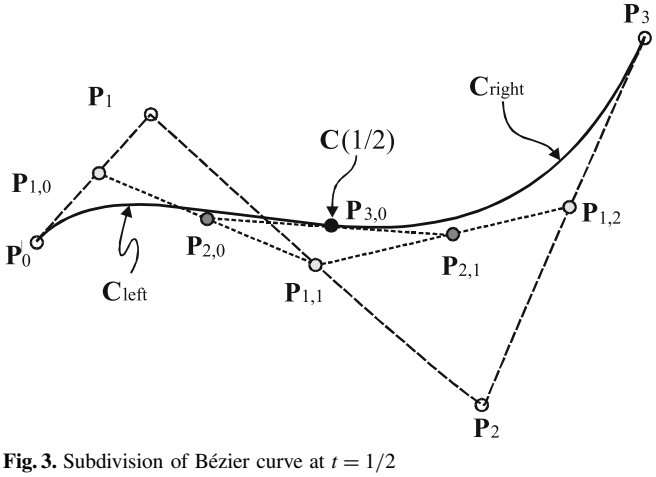

Fig. 3. Subdivision of Bézier curve at $t = 1/2$

3 Generating linear approximation tool paths

Before constructing the intended NURBS tool paths, linear approximation and self-intersection-free tool paths must be obtained first. The reasons why NURBS curves are initially approximated by line segments for tool paths generation are explained as follows.

1. Using line segments to approximate NURBS curves within prescribed tolerance is a fundamental geometric operation in CAD/CAM, which is essentially based on NURBS decomposition and subdivision of Bézier curves [16, 19, 20]. The error generated for this linear approximating can be calculated easily and reliably by considering the convex hull property of Bézier curves [19]. The details are described in Sect. 4.2.
2. Pockets with boundary curves composed of lines and/or arcs have been widely studied in the literature [1, 5, 8, 21, 22], therefore, carefully chosen current algorithms can be directly applied to constructing the linear approximation tool paths for freeform pocketing.
3. The non-self-intersecting linear approximation tool paths will provide a frame structure for building smooth, error-controlled, and non-overcutting NURBS tool paths.

Concerning conventional 2.5D pocket machining, generally, four different approaches for obtaining the offset boundaries of pockets can be found in the literature: (1) pair-wise offset, (2) Voronoi diagram, (3) pixel-based method, and (4) Boolean set operations [5, 23, 24]. As stated previously, a notable feature common to those approaches is that they only can handle pockets with line and/or arc boundary curves. Hence, if freeform curves are used to define pocket boundaries, they must be approximated by line/arc curves to proceed with follow-up operations. On the other hand, as NURBS-format tool paths are demanded for NURBS-based pocketing, the initial transformation between freeform curves and lines/arcs seems to become unnecessary. However, in this research, the process of linear fitting is not redundant since the linear approximation tool paths are able to serve as the "carriers" for constructing final NURBS-format tool paths. And a salient benefit to do so is that the problems of self-intersections for offsetting NURBS curves can be handled directly and easily in those initial linear offsets.

In this research, a modified version of pair-wise offsetting algorithm, which is originally proposed by Hansen and Arbab, is employed for offsetting the linearized boundary curves [5]. However, unlike their method, the entities now used to represent the profiles are only line segments, and the gaps, may exist in their initial offsets, are closed by linear extensions. The main benefit of using Hansen and Arbab's algorithm is that the time com-

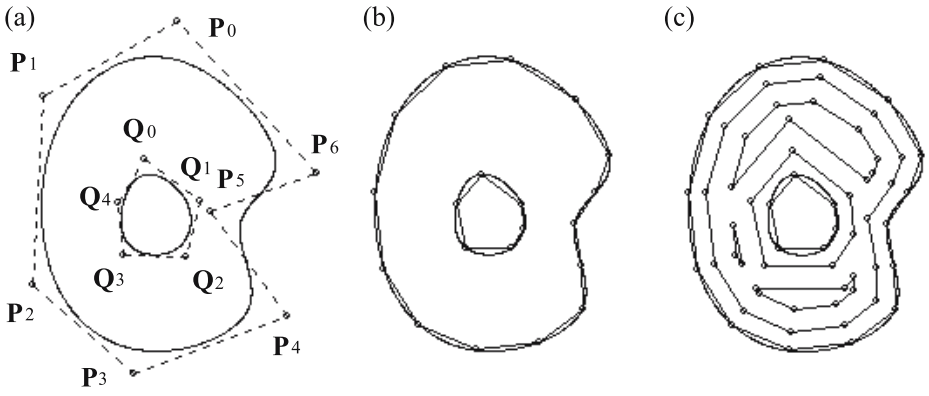

Fig. 4a–c. Generating linear approximation tool paths. **a** Pocket profiles. **b** Linear approximating pocket profiles. **c** Linear approximating tool paths

plexity for finding all intersections is only $K_p O(n \log n)$ and the invalid loops can be removed effectively by applying the interference indexing method.

Figure 4 shows the process of generating linear approximation tool paths for a freeform pocket with one island. As shown in Fig. 4a, the pocket periphery is arbitrarily defined counterclockwise by control points $\mathbf{P}_0, \ldots, \mathbf{P}_6$ and the island is defined clockwise by control points $\mathbf{Q}_0, \ldots, \mathbf{Q}_4$. To get the linear approximation tool paths, the boundary curves are approximated by straight line segments within a user prescribed tolerance (Fig. 4b). Then, based on the redefined boundary line segments, the tool paths without invalid loops can be obtained by using the modified pair-wise offsetting algorithm with interference indexing method (Fig. 4c). With this illustration, contour-parallel tool paths are given for demonstrating this procedure graphically. The approximation error is 6, the cutter radius is 10, and the cutter overlap is 70%.

4 Reconstructing NURBS-format tool paths

This section describes how to efficiently reconstruct G^1-continuous NURBS-format tool paths based on the obtained linear approximation tool paths. In addition, issues of error control and self-intersecting loop removal for generating valid tool paths with practically few control points are also dealt with in this section. The details are further described as follows.

4.1 G^1-continuous NURBS format tool path generation

The ultimate goal of this method is to generate NURBS-format tool paths for NURBS based pocket machining. The linear approximation tool paths will be regarded as the intermediate objects for building valid NURBS-format tool paths. Considering the linear tool paths as a collection of Bézier control polygons, piecewise Bézier curves with G^1 continuity can be constructed. To achieve G^1 continuity for the cutter paths, some points are inserted along the line approximating curves as additional control points of Bézier curves. Choosing Bézier curves of degree three to fit the offset curve, a proposed procedure for the inserting operations is described as follows.

1. If the number of line segments is *even*, see Fig. 5a, the insertion operation starts from the middle of the first line segment and continues to insert control point once on every other line segment down the line approximating curve. The procedure is ended when the first insertion point is met again.
2. If the number of line segments is *odd*, see Fig. 5b, the same manipulating procedure for the even case is applied on the line approximating curve except the last line segment, in which two control points are inserted with equally spaced distance.

As shown in Fig. 5, using the end points of the line segments, i.e., vertices of the polygon, combined with the insertion points marked as gray filled circles, piecewise Bézier curves can be constructed to be at least G^1-*continuous* in integral form.

However, two additional problems emerge when piecewise Bézier curves are used to fit the linear tool paths. (1) Can the

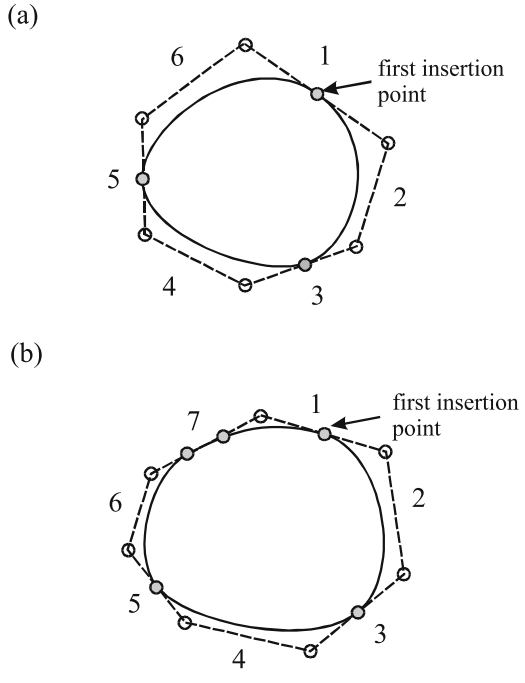

Fig. 5. a Inserting points for even case; **b** Inserting points for odd case

approximation error be measured efficiently? (2) If the user prescribed tolerance is not satisfied, what measures can be taken for improvement? Answers to these questions are described as follows.

1. Approximation error for Bézier-fitting tool paths is measured also by using the convex hull property of Bézier curves. The details are described in Sect. 4.2.
2. If the approximating error for any one of the Bézier pieces is greater than the prescribed tolerance, then, the control polygon, not the Bézier curve itself, is subdivided into two smaller parts by inserting a control point to the middle of every leg of the control polygon. As shown in Fig. 6, the two newly generated Bézier curves will approach the original line segments more closely than the old one and still keep G^1 continuity on the piecewise curve. The refining process is performed repeatedly until the prescribed tolerance is satisfied for every Bézier curve.

Comparing to subdivision on Bézier curves, subdivision on Bézier control polygons is simpler and faster. However, because of different numbers of subdivisions of control polygons for different Bézier entities in achieving the same prescribed tolerance, the path curves generated can only be piecewise Bézier curves with G^1-continuity in uniform parameterization. If more compact output data are requested, the continuity at joints should be further improved to C^1, such that the piecewise Bézier segments can be represented in integral B-spline form with fewer control points [15]. This objective can be easily achieved with a post-process that reassigns the piecewise Bézier curves with proper parameter ranges to make the underlying parametrization of all joints to be C^1-continuous [15].

4.2 Error control

One of the most attractive features of the proposed method lies in its capability of error measurements. As stated previously, all the errors generated in computing NURBS format tool paths can be estimated by using the convex hull property of Bézier curves [16, 19]. As shown in Fig. 7, the base line of a Bézier curve is defined as a straight line passing through the endpoints of a Bézier curve. The approximation error is measured by the distance of the farthest control points on the convex hull deviating from the base line. Since the Bézier curve is completely contained in the convex hull of its defining control polygon, the real error of a Bézier curve to its control polygon or its base line segment will be no more than the approximation error defined by Fig. 7.

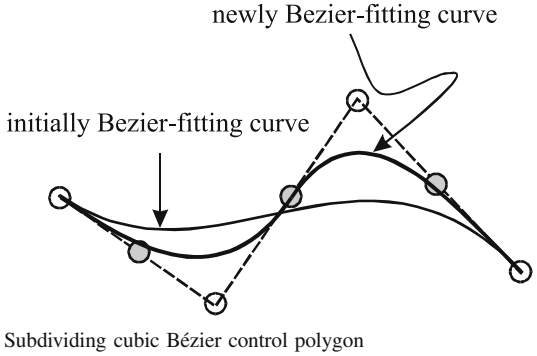

Fig. 6. Subdividing cubic Bézier control polygon

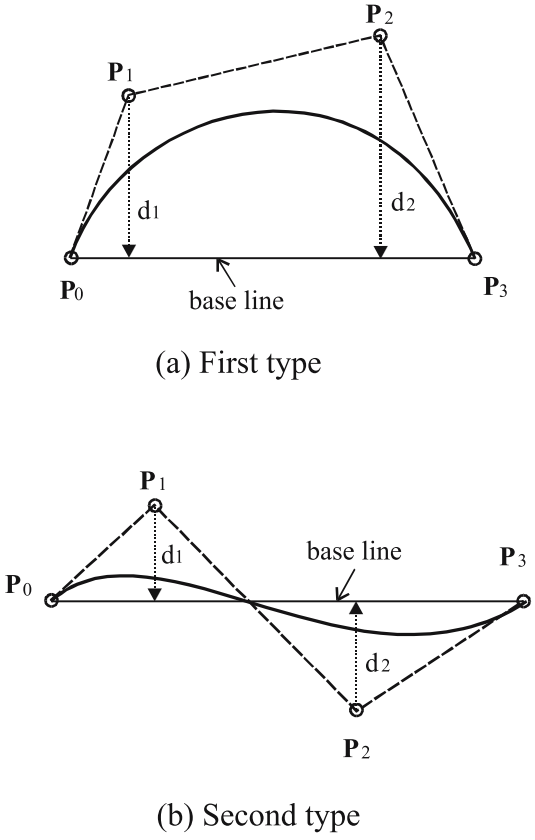

Fig. 7. Estimating line-fitting or Bézier-fitting error (upper bound error = Max(d_1, d_2))

The total error of an approximated NURBS offset curve ε is caused by the fitting error ε_1 of line segments relative to the progenitor curves plus the fitting error ε_2 of piecewise Bézier curves relative to the offset line segments. That is

$$\varepsilon = \varepsilon_1 + \varepsilon_2. \tag{6}$$

For the line-fitting curves, the first part of the prescribed tolerance ε_1 should be satisfied by every constituent Bézier piece. If any Bézier curve deviates from the allowance, it must be recursively subdivided until the allowance is satisfied. As for the Bézier-fitting curves, the second part of the prescribed tolerance ε_2 should be satisfied in each reconstructed Bézier curve. If any Bézier curve deviates from the allowance, its defining control polygon must be recursively subdivided until the allowance is satisfied.

However, how to distribute the total error ε into the two smaller parts, ε_1 and ε_2, is another issue worth further discussing. Obviously, small sizes of ε_1 and/or ε_2 imply that a large number of subdivisions are required. Unfortunately, subdivisions are also the major cause of generating additional control points with this method. Therefore, to seek an optimal error distribution, which

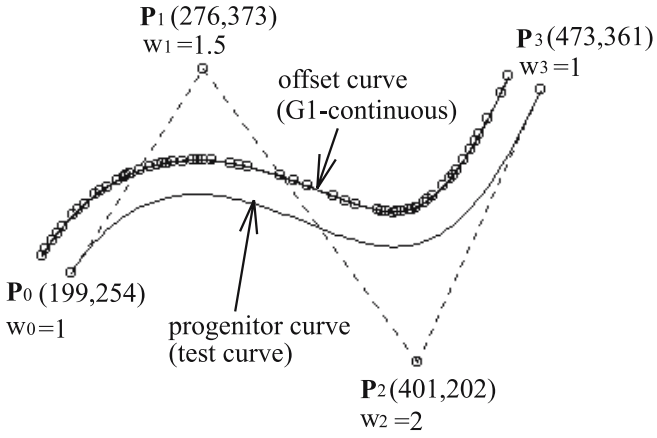

Fig. 8. Offset of a cubic NURBS curve ($\varepsilon = 1$, $\varepsilon_1/\varepsilon_2 = 3/7$, offset distance = 20)

could inhibit the fast growth of the number of control points, is an important task. With extensive tests of practical examples, the results have shown that a ratio of $\varepsilon_1/\varepsilon_2 = 3/7$ is the best value. Table 1 demonstrates this fact by using the test curve shown in Fig. 8 with three different tolerances $\varepsilon = \{1, 0.1, 0.01\}$ and an offset distance of 20.

Why the optimal $\varepsilon_1/\varepsilon_2$ ratio falls in the vicinity of 3/7 is reasoned as follows. On the side of an extremely low $\varepsilon_1/\varepsilon_2$ ratio, the precision of a line-fitting curve is far higher than a Bézier-fitting curve. Subdividing the Bézier curve one time only can increases a control point to the output data because base line segments are used to approximate the Bézier curves in constructing the line-fitting curve. Hence, over-subdividing in constructing the initial line-fitting curve should be the main factor for generating such unnecessarily massive control points. On the other hand, if the $\varepsilon_1/\varepsilon_2$ lies in the range of the higher ratio side, the required subdivisions will be mostly shifted to the Bézier-fitting curve. The control points used for reconstructing NURBS tool paths will be drastically increased because subdividing the control polygon of Bézier curves for the Bézier-fitting curve will generate more control points than subdividing the Bézier curves for the line-fitting curve.

Additionally, Table 1 also shows that if a large cumulative error ε is assigned, a relatively small number of control points will be generated, meanwhile the continuity of offset curves keeps unchanged. Since most of the generated offset curves are used as rough cutting tool paths for freeform pocketing, see Sect. 5, the output data size can be further reduced by using a looser tolerance. Thus, the proposed method is able to generate compact and smooth tool paths for high-speed NURBS-based machining.

4.3 Eliminating self-intersecting loops

To solve the problem of self-intersecting loop removal for NURBS offset curves is very difficult for the computing of higher degree curve/curve intersections. Moreover, G^0-continuous cusps will form on the tool path curves after invalid loops have been removed. Such adverse effect as a result of self-intersecting loop removal is at the cost of loss of continuity in NURBS based pocketing.

Fortunately, the problem of detecting and removing self-intersecting loops can be reduced to a simpler case of polygon offsetting. Also it will be solved before reconstructing the final NURBS-format tool paths. The loss of continuity, which is usually produced by the self-intersecting loop removal in using other methods, will not appear with this method.

Figure 9 demonstrates the phenomena of eliminating self-intersections and cusps in reconstructing a closed NURBS offset curve. The simple polygon, shown in Fig. 9a, represents an initial linear approximation tool-path. As stated previously, in order to construct G^1-continuous NURBS offset curve, some control points must be inserted on the line segments (Fig. 9b). Figure 9c clearly indicates that all of the Bézier convex hulls of the composite curve do not intersect each other for the corresponding positions of control points. Hence, the reconstructed NURBS curve is of course valid with no self-intersections or cusps (Fig. 9d). In addition, if subdividing control polygons are required for some constituent Bézier pieces shown in Fig. 9c, self-intersections and cusps will not occur either. Figure 9e shows the result with zooming in.

As shown in Fig. 10a, an open Bézier curve can have a self-intersecting loop even though its control polygon has no self-intersection. However, in this research, since all tool paths are closed and the curves represent the cutter paths are at least G^1-continuous, the tool path curves are initially C^1-continuous (Fig. 5) and G^1 continuity can be formed only after some control polygons have been subdivided (Fig. 6), a tool path cannot be constructed by using just an isolated Bézier curve. As demonstrated in Fig. 10b, if another Bézier curve is added to the Bézier curve in Fig. 10a to form a closed piecewise Bézier curve with C^1 continuity, the two control polygons will certainly intersect each other no matter if subdividing control polygons is performed or not. Such a condition is in conflict with the result of the linear approximation tool path, which is a simple polygon without a self-intersecting loop, and should never be used as a pocket boundary. As researched by Tiller and Hanson [8], the same result has also been mentioned from a different direction of view, which says that if the control polygons approximate the curves enough closely, the phenomenon as in Fig. 10a is very unlikely to happen.

Table 1. Numbers of control points generated for the test curve shown in Fig. 8

$\varepsilon_1/\varepsilon_2$	1/9	2/8	3/7	4/6	5/5	6/4	7/3	8/2	9/1
$\varepsilon = 1$	82	67	58	64	85	100	118	157	223
$\varepsilon = 0.1$	277	184	178	190	229	289	406	490	676
$\varepsilon = 0.01$	811	586	544	595	799	1006	1240	1636	2140

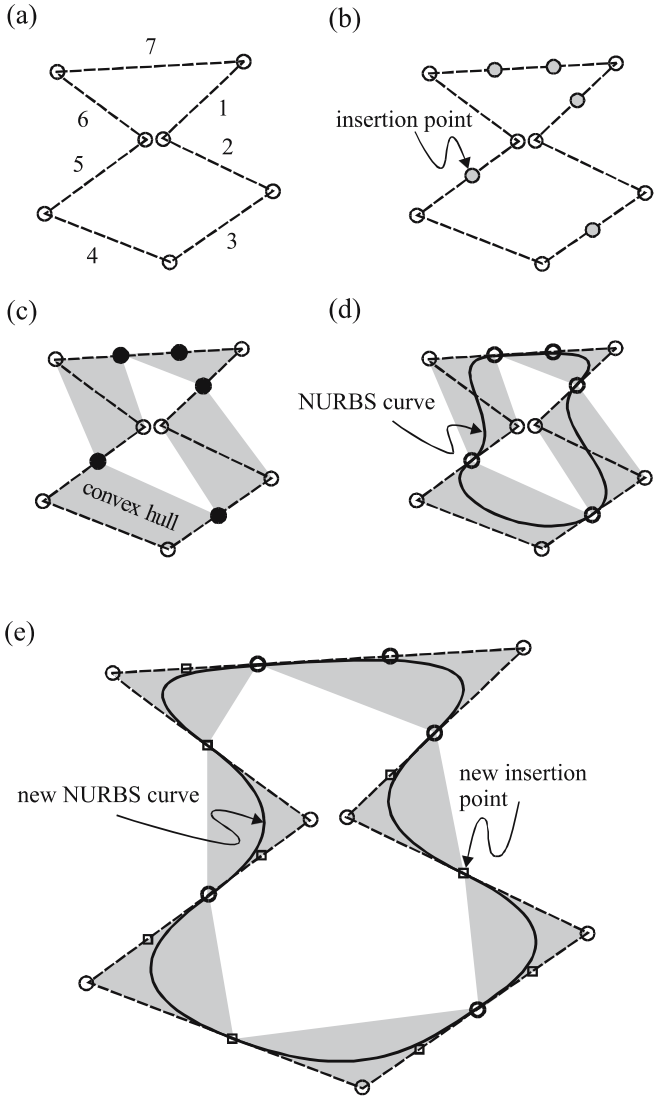

Fig. 9. Constructing non-self-intersecting and cusp-free NURBS offset

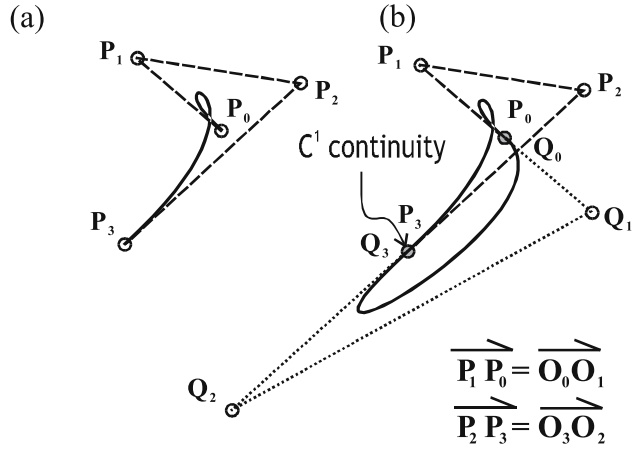

Fig. 10. a A self-intersecting Bézier curve with non-self-intersecting control polygon. **b** A closed and C^1-continuous piecewise Bézier curve with self-intersecting curve and control polygon

5 Examples

The method described earlier has been implemented to generate NURBS-format tool paths for arbitrary freeform pockets with islands. In the following demonstrating examples, the contour-parallel tool paths are obtained by repeatedly offsetting the NURBS boundary curves of pockets. Essentially, two types of tool paths for a pocket can be identified in practice. The curve of the first offsetting is used as a finish cutting tool path, and the other offset curves are regarded as rough cutting tool paths. As shown in Fig. 11, generally, in order to reduce the output data size and to increase the machining efficiency for pocketing, different offset distances (stepovers) are used for finish cutting and rough cutting. And the prescribed tolerances for the two types of tool paths are also assigned differently. The stepover size for finish cutting is equal to the cutter radius, and the stepover size for rough cutting can be in the range from cuter radius to cutter diameter. The allowance for the finish cutting tool path is assigned mainly based on functional or aesthetic requirements of machined parts. Hence it is tighter in order to carve the details of pocket boundary curves. For rough cutting tool path, fast removal of the unnecessary material is its main purpose, the allowance should be assigned more loosely to reduce the output data size and hence to increase machining efficiency. Practically, the range of an enlarged tolerance zone for rough cutting tool path curves is between zero and half of the distance of cutter overlapping.

Two practical examples are presented in Figs. 12 and 13 to demonstrate the effectiveness and usefulness of the proposed method. In Fig. 12, the NURBS tool paths are shown for a fish-like pocket with one island. The following data are used for this case: the cutter radius is 10 mm, the cutter overlap is 100%, and a consistent tolerance 0.1 mm is used for both finish cutting and rough cutting. Figure 13 shows the NURBS-format tool path generation for an arbitrary freeform pocket with two islands. In this case, the

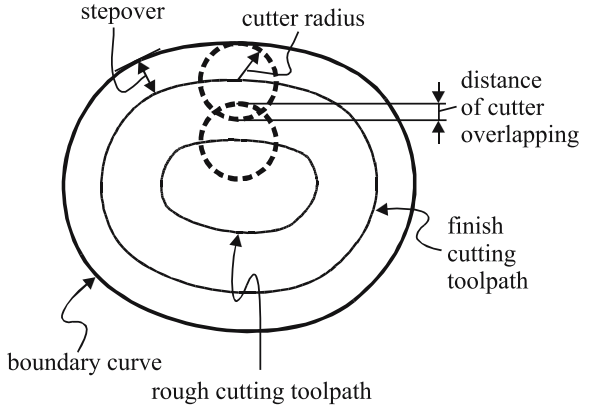

Fig. 11. Terminology of tool paths

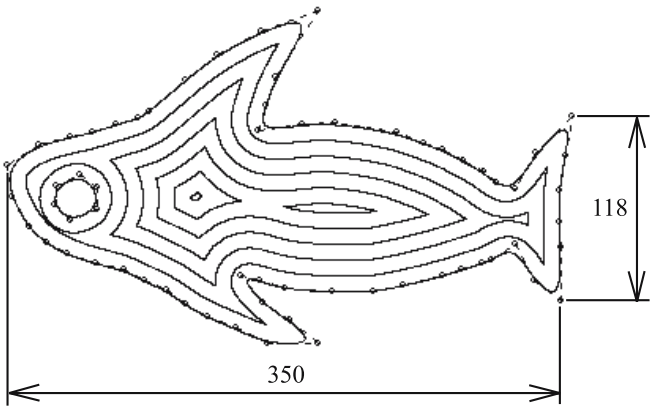

Fig. 12. NURBS tool path generation for a fish-like freeform pocket with one island

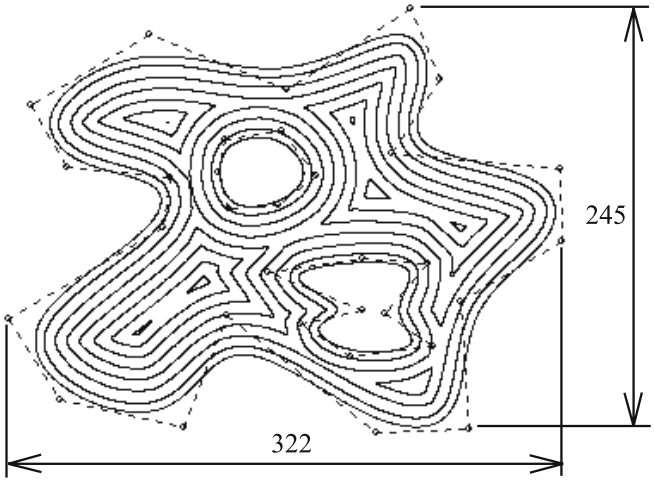

Fig. 13. NURBS tool path generation for an arbitrary freeform pocket with two islands

tool paths are generated with a 5 mm cutter radius and a 70% cutter overlap. The tolerances assigned for calculating finish cutting and rough cutting tool paths are 0.1 mm and 3 mm, respectively.

As illustrated by these examples, the tool paths for NURBS-based pocketing can be constructed validly with globally controllable error and no self-intersections by using the proposed method. The method also guarantees that all of the generated tool paths are at least G^1-continuous and cusp-free. Hence, it is most appropriate for freeform pocketing application with CNC machines that are capable of NURBS interpolating.

6 Conclusions and future work

A method is proposed in this paper to generate NURBS-format tool paths for freeform pockets with islands. The generated cutter path curves are guaranteed to be G^1-continuous with errors within the required tolerances. The advantages of the proposed method are: (1) the problem of detecting and removing self-intersecting loops is degenerated into a simpler case with entities only of line segments; (2) the obtained tool paths are kept cusp-free after invalid loops have been removed; and (3) an integrated strategy to increase the machining efficiency of freeform pocketing is developed by using different stepover sizes and tolerances for rough cutting and finish cutting, respectively.

Since all operations involved in this study are linear geometric calculations, the proposed method for NURBS-based tool path generation is very efficient and robust. In the next step, it is worth investigating to further reduce the number of tool path curve segments and hence the control points generated. Moreover, applying this method on 3D sculptured surface machining is another future direction.

References

1. Held M (1991) On the computational geometry of pocket machining. Springer, Berlin Heidelberg New York
2. Kim K, Jeong J (1995) Tool path generation for machining free-form pockets with islands. Comput Ind Eng 28(2):399–407
3. Chuang SH, Kao CZ (1999) One-sided arc approximation of B-spline curves for interference-free offsetting. Comput Aided Des 31(2):111–118
4. Tseng YG, Chen YD, Liu CC (2001) Numerically controlled machining of freeform curves using biarc approximation. Int J Adv Manuf Tech 17(11):783–790
5. Hansen A, Arbab F (1992) An algorithm for generating NC tool paths for arbitrary shaped pockets with islands. ACM Trans Graph 12(2):152–182
6. Farouki RT, Neff C (1990) Algebraic properties of plane offset curves. Comput Aided Geom Des 7:101–127
7. Klass K (1983) An offset spline approximation for plane cubic splines. Comput Aided Des 15(5):297–299
8. Tiller W, Hanson EG (1984) Offsets of two-dimensional profiles. IEEE Comput Graph Appl 4(9):61–69
9. Coquillart S (1987) Computing offsets of B-splines curves. Comput Aided Des 19(6):305–309
10. Hoschek J (1988) Spline approximation of offset curves. Comput Aided Geom Des 5(1):33–40
11. Pham B (1988) Offset approximation of uniform B-splines. Comput Aided Des 20(8):471-474
12. Elber G, Cohen E (1991) Error bounded variable distance offset operator for free form curves and surfaces. Int J Comput Geom Appl 1(1):67–78
13. Lee IK, Kim Ms, Elber G (1996) Plane curve offset based on circle approximation. Comput Aided Des 28(8):617–630
14. Chuang SH, Shih JL (in press) A novel approach for computing C^2-continuous offset of NURBS curves. Int J Adv Manuf Tech
15. Piegl L, Tiller W (1997) The NURBS book, 2nd ed. Springer, Berlin Heidelberg New York
16. Marsh D (1999) Applied geometry for computer graphics and CAD. Springer, Berlin Heidelberg New York
17. Boehm W, Prautzsch H (1985) The insertion algorithm. Comput Aided Des 17(2):58–59
18. Farin G (1993) Curves and surfaces for computer aided geometry design, a practical guide, 3rd ed. Academic, San Diego, CA
19. Chuang SH, Lin WS (1997) Tool-path generation for pockets with freeform curves using Bézier convex hulls. Int J Adv Manuf Tech 13:109–115
20. Piegl L, Tiller W (2002) Biarc approximation of NURBS curves. Comput Aided Des 34:807–814
21. You CF, Sheen BT, Lin TK (2001) Robust spiral tool-path generation for arbitrary pockets. Int J Adv Manuf Tech 17(3):181–188
22. Park SC, Choi BK (2001) Uncut free pocketing tool-paths generation using pair-wise offset algorithm. Comput Aided Des 33(10):739–746
23. Persson H (1978) NC machining of arbitrarily shaped pockets. Comput Aided Des 10(3):169–174
24. Choi BK, Kim BH (1997) Die-cavity pocketing via cutting simulation. Comput Aided Des 29(12):837–846