Manufacturability Analysis for 5-Axis Sculptured Surface Machining

Wenyu Yang  Han Ding  Youlun Xiong

School of Mechanical Science & Engineering
Huazhong University of Science and Technology,
Wuhan, Hubei, 430074, P.R.CHINA

Abstract

The general algorithm for manufacturability analysis of 5-axis sculptured surface machining is presented in this paper. The visibility cone which represents the aggregate of all visible directions is constructed to describe geometric constraints. Combined with convex hull computation algorithm, a detailed procedure for computing the visibility cone has been developed. Through the geometric constraint analysis, the manufacturability for sculptured surface machining is evaluated so that geometric defects and inefficiencies can be inspected and corrected in the earlier design stages. Hence reworks in product development can be reduced or even avoided and the concurrence of sculptured surface design and manufacturing will be enhanced.

1 Introduction

Sculptured surfaces have been commonly applied in the automotive or aircraft industry, die and mold manufacturing, ship hull manufacturing, consumer appliances, etc. The generation of machining instructions is an important and time consuming step in multi-axis machining. To make machining technology more accessible in today’s demanding industrial environment, it is necessary to explore a new paradigm for generating interference-free, multi-axis machining paths directly from the CAD representation of the geometric object. Some methods of machined surface error analysis and tool collision detection by machining strip evaluation methods for 5-axis machining have been developed and presented in [1,2,4]. A method of using offset surfaces and differential geometry to evaluate feasible machining strip width for 5-axis die/mold surface machining has been reported in (Lee 1997). A method of non-isoparametric path planning has been developed to improve the machining efficiency of complex surface machining (Lee 1998). Due to the non-analytic contours of a sculptured surface, the method has been considered as an efficient way only for a small product related feature set [5] so far. A framework for a new, kinematics-based method is presented in [6]. It provides a basis for concurrent geometric shape design and machining kinematics analysis. Although it has demonstrated a compatible mathematical method of geometry modeling and machining, the method is limited in the kinematics application area. Vafaeeesa and EIMaraghy [7] presented an efficient approach for analyzing the global feasibility of the tool orientation in 5-axis sculptured surface machining. In their approach, tool geometry, local shape of the surface and other obstacles can be taken into consideration.

Evaluating manufacturability involves finding a way to manufacture a proposed design, and to estimate the corresponding production cost and quality. As described by Jones et al [5], from a manufacturing point of view a sculptured surface is almost featureless. It is normally produced by a multi-axis CNC milling machining according to the desired design specifications. The manufacturability analysis module is used to evaluate the feasibility of CNC machining. According to Chang [8], we assume that machining resource constraint, condition constraint and precedence constraint, for an operating plan to machine a sculptured surface, can be satisfied. Because a sculptured surface has a variety of curvatures and a complicated topological geometry, geometric constraint for CNC machining.

In our research, we are interested in whether a proposed design is manufacturable with a given tool approaching direction. The proposed method can identify admissible tool positions and directions during the CNC machining process. If a tool is in a feasible direction, there will be no collision between the non-cutting area of a tool and the work piece.
2 Visibility direction

For a surface \( S \in \mathbb{R}^3 \), \( S \neq \emptyset \), and \( x_v \in S \), \( x_v \) is a viewing point.

1. A ray \( \gamma \) from \( x_v \) to \( x_0 \)
   \[ \gamma = \lambda (x_v - x_0) \quad \forall \lambda > 0, \ x_v \neq x_0 \]

2. \( \gamma \) is a visible direction, when
   \[ \gamma = \{ \gamma : \gamma \cap S = \emptyset \} \]

3. \( \mathcal{E}(S, x_0) \) can be defined as visibility cone,
   \[ \mathcal{E}_o = \{ x_v : \gamma \cap S = \emptyset \} \]
   \[ \mathcal{E} = \text{cl} \mathcal{E}_o \]

where \( \text{cl} \mathcal{E}_o \) represents the closed hull of \( \mathcal{E}_o \). If the ray starting from a viewing point \( x_v \) to point \( x_0 \) in the direction \( \gamma \) does not intersect the interior of \( S \), then \( x_0 \) is visible in a viewing direction \( \gamma \). The visibility cone represents the aggregate of all visible directions. A point on a surface is machinable by (or visible to) the cutter if the cutter can come in contact with the point without gouging the neighborhood of the point. Thus, given a surface, how to compute visibility cone at a given point should be solved for a set of viewing directions.

3 Visible Direction Constrained by Local Domain and Neighboring Domain

For a given surface \( S \) and a point \( x_0 \) at surface \( S \), \( P \) is a normal vector at \( x_0 \). \( H \) is the hyperplane at \( x_0 \) which is defined as

\[ H(x_0) = \{ x \in \mathbb{R}^3 | (x - x_0) \cdot P = 0 \} \]

where \( \cdot \) is an inner product between vector \( (x - x_0) \) and vector \( P \).

The visible direction for \( x_0 \) is constrained by the local geometric feature from \( S_j \) and the neighboring geometric feature from \( S_i \), as indicated in Figure 1.

The constraint from neighboring domain \( IS \) to \( x_0 \) can be expressed as \( D(IS, x_0) \).

\[ D(IS, x_0) = D(S_1, x_0) \cup D(S_2, x_0) \cup \ldots \cup D(S_n, x_0) \]  (3)

If the local constraint is not considered, then the visibility cone at \( X_0 \) of surface \( S_j \) can be expressed as a positive half space \( H^+ \) along a normal direction \( P \).

\[ c(S_j, x_0) = H^+(x_0) = \{ x \in \mathbb{R}^3 | (x - x_0) \cdot P \geq 0 \} \]  (4)

The visibility cone at \( X_0 \) can be generally expressed as

\[ \mathcal{E}(S, x_0) = \mathcal{E}(S_j, x_0) \cap D(IS, x_0) \]  (5)

In a special case, if \( S_j \) is a convex surface or a flat surface, then local geometric constraints at \( x_0 \) can be neglected and the visibility cone at \( x_0 \) can be expressed as

\[ \mathcal{E}(S, x_0) = H^+(x_0) \cap D(IS, x_0) \]  (6)

where \( S = S_j \cup IS \), \( S_j \) is the local domain, and \( IS = S_1 \cup S_2 \ldots S_n \) is the neighbouring domain. According to the convexity of a Bezier surface, surface patches in \( S_j \) are located in convex hull \( CH(S_i) \) spanned by \( S_i \) generators. So, if a ray does not intersect the convex hull of a surface, it will not intersect with a surface. The convex hull of \( S_i \) is signed as \( CH(S_i) \).

The \( CH(S_i) \) is a convex hull at an extreme point \( x_0 \) which is shown in Figure 2. The surface \( S_j \), convex hull \( CH(S_i) \) and \( CH(S_i, x_0) \) are shown as in Figure 3. It is concluded that all rays at a common extreme point \( X_0 \) will not intersect surface \( S_j \), if they have no intersection with the convex hull \( CH(S_i, x_0) \).

4 Neighboring Geometric Constraints

Because each face in a convex hull is a supporting face, all supporting faces of the convex hull \( CH(S_i) \) at the extreme point \( X_0 \) construct a convex cone which is called as the supporting cone of \( CH(S_j) \). By extracting adjacent triangle plane patches at an extreme point \( X_0 \), a supporting cone \( \Lambda \) can be formed as indicated in Figure 4. The points in plane patches except \( X_0 \) can be recorded by searching neighbouring triangle
plane patches. Obviously $x_i, i = 1, 2, \cdots, m$ can be obtained as a result, $m$ is the number of triangular plane patches. Point $X_i$ and point $X_0$ can constitute the vector $v_i = x_i - x_0, (i = 1, 2, \cdots, m)$. The vector set $V = \{v_i \mid i = 1, 2, \cdots, m\}$ can form a convex cone $\Lambda$.

$$\Lambda = \{ \sum_{i=1}^{m} a_i v_i \mid a_i \in \mathbb{R}, a_i > 0, i = 1, 2, \cdots, m \}$$  (7)

As we know that $\Lambda$ is a convex cone, if and only if $(x+y) \in \Lambda, \lambda x \in \Lambda$ whenever $x, y \in \Lambda, \lambda \geq 0$. Because every vector $V_i$ in $V$ is an edge with an extreme point $x_0$, the generated supporting cone $\Lambda$ by $V_i$ will be certainly convex. Let every vector $V_i$ normalized, the unit vector $v_i' = x_i' - x_0$ can be obtained. Every unit vector $v_i'$ and supporting cone are shown in Figure 5.

The closed hull of supporting cone $\Lambda$ represents visible direction constraints $D(S_j, x_0)$ for a given point $X_0$.

$$D(S_j, x_0) = c \Lambda$$  (8)

The object-oriented programming method is adopted for the calculation of neighboring constraints. A few basic objects are defined as follows:

(1). Edge: double connected edge in convex hull.

(2). Face: triangle plane patches.

(3). Basepoint: a point position in Cartesian space.

(4). ConeHull: convex hull of generators in surface $S$.

The program design is outlined as follows:

```cpp
// -- Define the class for describing double connected edge list --/
class Face: public Object
{
  basepoint Vertex1, Vertex2, Vertex 3; /* define triangle plane patches */
  ...
};

class Edge: public Object
{
  public:
    int V1, V2, F1, F2, P1, P2; /* define pointer for point, plane and edge with double connected edge list */
    Boolean adjust; /* identify attribute of operation procedure */
    Boolean Inside; /* identify attribute of operation procedure */
    ...
};

//-- Define the class for describing points in Cartesian space --/
class basepoint : public Object
{
  private:
    float _xpos, _ypos, _zpos; /* define points in Cartesian space */
    ...
};

//-- Define the convex hull --/
class ConeHull
{
  private:
    basepoint *VPtr; /* Pointer for a point */
    Face *FPtr; /* Pointer for a plane */
    Edge *EPtr; /* Pointer for an edge */
    Boolean Status; /* Status attribute */
    ...

  public:
    void Hull(basepoint *VPtr); /* Convex hull calculation */
    void Cone(basepoint X0); /* Convex cone calculation */
    ...
};
```

5 Local Geometric Constraints

This section will deal with local constraints to visible direction at point $X_0$. For surface patches which do not include the point $X_0$, the above method can be directly applied. For a surface patch including $X_0$, it must be further subdivided until eq.(4) is satisfied. To subdivide $S_j$, the oriented refinement subdivision algorithm is adopted according to the position of the point $X_0$. Compared with the accuracy-based subdivision algorithm, the oriented refinement subdivision algorithm is only to subdivide a surface patch including the point $X_0$; the computational efficiency can be greatly enhanced. Figure 6 shows only 10 sub patches are needed when the approaching accuracy $\varepsilon = 5.00$ mm based on the oriented refinement subdivision algorithm. However, 169 sub patches are needed by using...
accuracy-based subdivision algorithm, as shown in Figure 7. Oriented refinement subdivision algorithm can be concluded as follows:

(1). Initialize the stack A and quad tree object B and let \( S_j \) to be root nodes of B.
(2). Subdivide surface \( S_j \) and get corresponding sub nodes to be recorded.
(3). Locate a surface patch including point \( x_0 \).
(4). Judge approaching accuracy. If not satisfied, go to Step 2.
(5). Push all nodes in B into the stack A, except root nodes and a node including the point \( x_0 \).

6 Visibility Cone Computation

For given local and neighboring constraints, the procedure for computing visibility cone \( C(S, x_0) \) at a point \( x_0 \) can be summarized as follows:

(1). Initialize the stack A and push all surface patches \( S_i \) to the stack.
(2). Initialize the visibility cone \( C(S, x_0) = H^+(x_0) \).
(3). Pop each surface patch \( S_k, k = 1, 2, \ldots, n \).
(4). Calculate \( D(S_k, x_0) \).
(5). Calculate \( C(S, x_0) = H^+(x_0) \cap D(S_k, x_0) \).
(6). Repeat (3), (4) and (5) until stack A is empty.
(7). End.

7 Interactively Manufacturability Analysis

A framework for sculptured surface manufacturability analysis is shown as Figure 8. The geometry information extracting and converting module consists of an interactive interface sub-module and a data modelling sub-module. The interactive interface sub-module is used to access the local database of a CAD/CAM system and to retrieve the surface data from the geometric model. The data modelling sub-module converts the initial geometric model to a machining oriented intermediate data model which is called the Inter-Data Model. A manufacturability analysis module is designed to calculate the machining feasibility. The module will allow the designer to consider manufacturing problems during the design stage, thereby producing designs that will be easier to manufacture.

8 Conclusion

This paper presents a computer aided manufacturability analysis for the concurrent design and manufacture of the sculptured surface based on visible direction and visibility cone. Using the visibility cone, feasible tool approaching direction information at a given point can be easily obtained. Combined with convex hull computation algorithm, a detailed procedure for computing the visibility cone has been developed. Therefore, the visibility cone can be used as an effective tool for manufacturability analysis of the sculptured surface machining. After the visibility cone is acquired, tool orientation can be adjusted to adapt itself to the local geometry of the sculptured surface at each cut contact position. The proposed methodology can be applied to computer-aided planning and programming of cutter path generation for 5-axis sculptured surface machining.

References

Figure 1 Visibility direction constrained by neighboring and local geometric feature

Figure 2 Convex hull \( CH(S_i, x_0) \)

Figure 3 (a) Surface \( S_i \) and convex hull \( CH(S_i) \)
(b) Convex hull \( CH(S_i, x_0) \)

Figure 4 Supporting cone
Figure 5 Supporting cone after normalisation

Figure 6 Oriented refinement subdivision algorithm

Figure 7 Accuracy-based subdivision algorithm

Figure 8 A framework for sculptured surface manufacturability analysis