# Smooth Geometry Generation in Additive Manufacturing File Format: Problem Study and New Formulation

Kai-Ming YU* Yu WANG
Department of Industrial and System Engineering
The Hong Kong Polytechniq University
E-mail: mfkmyu@inet.polyu.edu.hk; Carolyn_yuwang@hotmail.com
(*Corresponding Author)

Charlie C. L. Wang
Department of Mechanical and Automation Engineering
The Chinese University of Hong Kong
E-mail: cwang@mae.cuhk.edu.hk

## *Abstract*

**Purpose:** In the newly released ASTM standard specification for *Additive Manufacturing File* (AMF) format – version 1.1, Hermite curve based interpolation is employed to refine input triangles to generate denser mesh with smoother geometry. This paper studies the problems of constructing smooth geometry based on Hermite interpolation on curves and proposes a solution to overcome these problems.

**Design/methodology/approach:** A formulation using triangular Bézier patch is proposed to generate smooth geometry from input polygonal models. Different configurations on the boundary curves are analysed to further enrich this formulation.

**Findings:** The formulation given in the AMF format (version 1.1) can lead to the problems of inconsistent normals and undefined end-tangents.

**Research limitation/implications:** The proposed scheme has requirements for the input normals of a model, only $C^0$ interpolation can be generated on those cases with less-proper input.

**Originality/value:** To overcome the problems of smooth geometry generation in the AMF format, an enriched scheme is proposed for computing smooth geometry by using triangular Bézier patch. For the configurations with less-proper input, the Boolean sum and the Nielson's point-opposite edge interpolation for triangular Coons patch are used to generate smooth geometry as a $C^0$ interpolant.

***Keywords***: curved patch, triangular Bézier, smooth geometry, file format, additive manufacturing.

## 1. Introduction

The processes of *Additive Manufacturing* (AM) have been more and more used in the production of precise parts in complex geometry. According to its nature of accumulating materials layer upon layer, geometries that cannot be fabricated by conventional subtractive manufacturing can now be produced by this technique, which has mainly been used for rapid prototyping but is now widely employed in mass production. In the last two decades, the STL file format has become a standard for describing the geometry of a part to be fabricated by AM. In an STL file, the surface geometry of a three-dimensional object is presented as a collection of triangular planar facets. As a result, very dense triangles have to be used in an STL file to reduce the shape approximate error between the tessellated shape presented in the STL file and the smooth geometry represented by B-rep in a CAD system. The major drawback of using a dense tessellation in representing a smooth geometry in an STL file comes from the cost of storage and communication when sending the file from one machine to another. Moreover, the computation time of slicing is also exponentially increased while refining a mesh surface by subdividing each triangle into four sub-triangles recursively. Although adaptive tessellation can help improve the problem (e.g., [1-3]), more industrial demands are found to rebuild a smooth geometry based on local construction using triangular facets with the addition of some additional geometric information – such as normals and tangents.

### 1.1 Motivation

Recently, to reflect this industrial demand, an open standard on *Additive Manufacturing File Format* (AMF) has been released [4]. This official ISO/ASTM 52915:2013 standard is an XML-based format designed to allow any computer-aided design software to describe the shape and composition of any 3D object to be fabricated on any 3D printer. Differing from the STL file format, non-geometric attributes including colour, materials, lattices, and constellations are fully described. More importantly, smooth geometry can be constructed on each triangle according to the normal vectors defined on vertices or tangent vectors specified on edges of triangles in this AMF format. However, our study shows that the formulation given in this AMF format can lead to the problems of inconsistent normals and undefined end-tangent (details will be presented in Section 2). A scheme of recursive refinement is defined in the AMF format to compute the smooth geometry. As a result, very dense triangles need to be generated (although only occurring in a local region). To overcome these problems, an enriched scheme is proposed in this paper for computing smooth geometry by using triangular Bézier patch.

Benefitting from the continuous representation of a triangular Bézier patch $\vec{P}(u,v,w)$ , any surface point on $\vec{P}(u,v,w)$ can be directly evaluated with given parameters ($u_0$, $v_0$, $w_0$). The construction method is direct and does not need any iterative or recursive steps of computation.

Furthermore, different configurations on the boundary curves of the constructed patch are analysed in Section 4 to further enrich this formulation.

### 1.2 Literature review

The modification of a shape presented by a collection of triangular plane facets into a smooth geometry is a research area that is relatively unexplored. Only a few related papers can be found in the relevant literature, which are mainly derived from the computer graphics community for improving the rendering quality of a model with a limited number of facets.

As a pioneering work in this area, three-sided cubic Bézier patches are employed in [5] to generate smooth shape by using the vertices on each triangle and the normal vectors assigned on these vertices. Although only $C^0$-continuity can be preserved across the boundary of triangular Bézier patches, it can generate satisfactory shading results when using specially processed normal vectors. The quality of rendering is further improved in [6] by assigning to each mesh vertex a set of three scalar tags that act as shape controllers. The approach proposed in [7] is a highly parallel algorithm for efficient construction of smooth surface consisting of low-degree polynomial pieces. To cover all major sampling and modelling scenarios, this method can be applied to polyhedral meshes consisting of triangles, quads, and pentagons and polar configurations and the implementation is developed for the DirectX 10 of Microsoft. They further extend the work into [8] by using the Gregory patches for hardware tessellation running on Direct3D 11. Leung et al. presented a local construction of smooth geometry using Gregory patches in [9]. Their framework provides more flexibility on the shape control – curved sharp features can be produced. Hierarchical triangular surface is presented in [10] for modelling surface of arbitrary topology, where the subdivision surface with $G^1$-continuity at the poles can be generated. Another construction can be found in [11]. A comparison on different Bezier patch interpolation schemes can be found in [12]. Besides, the technique for estimating the curvature tensor of a triangular mesh presented in [13] is also related to this work. In [14], a local refinement scheme is developed to generate smooth geometry for cloth simulation. Triangular Bezier patch is also employed to render giga-scale CAD models in [15]. However, these approaches mainly focus on the result of rendering. The real smooth geometry should be carefully constructed in the area of additive manufacturing.

The work of Navangul et al. [3, 16] developed a method called *Vertex Translation Algorithm* (VTA) to selectively modify the STL faces based on the chordal errors presented in a STL file. However, smooth geometry is not constructed in VTA. Smooth geometry is produced in the step of slicing in [17] through a variational smoothing and simplification process. Santosh et al. [18] proposed a curved shape using bi-quadratic Bezier surfaces with linear and curved edges for approximating the smooth surface of a CAD model in the piecewise manner. It is also worth noting that a Steiner patch based method is recently introduced by Paul and Anand in [19] for constructing

smooth geometry in additive manufacturing. However, unlike the AMF standard and the extension proposed in this paper which are general to all two-manifold models, their method has a limitation on processing trimmed surfaces and parts with holes. In a different manner, we reformulate the construction scheme under the framework of AMF format, which is a standard specification recently released by ASTM.

## 2. Problem Statement

In this section, the formulation of smooth geometry generation in the AMF format [4] will be introduced first. After that, problems of Hermite interpolation in this standard specification are analysed and a continuous formulation based on triangular Bézier patch is proposed to overcome these problems.

### 2.1 Formulation in AMF format

In the AMF format, each triangle facet is recursively refined into four sub-triangles by inserting new vertices at the middle of the polynomial curves interpolating the endpoints of triangular edges and also the tangents given on the endpoints. Without loss of the generality, for interpolating the endpoints $\vec{v}_0$ and $\vec{v}_1$ as well as their tangent vectors $\vec{t}_0$ and $\vec{t}_1$, the Hermite curve below (specified in Eq.(A3.2) from ASTM F2915 [4]) is employed.

$$\vec{h}(u) = (2u^3 - 3u^2 + 1)\vec{v}_0 + (u^3 - 2u^2 + u)\vec{t}_0 + (-2u^3 + 3u^2)\vec{v}_1 + (u^3 - u^2)\vec{t}_1$$

The newly inserted vertex on this edge $\vec{v}_0\vec{v}_1$ is positioned at $\vec{h}(0.5)$. For such cases where the tangent vectors are not explicitly specified in the AMF file, they are computed from the normal vectors $\hat{n}_0$ and $\hat{n}_1$ specified on $\vec{v}_0$ and $\vec{v}_1$ (according to Eq.(A3.1) in [4]):

$$\vec{t}_0 = \|\vec{d}\| \frac{-(\hat{n}_0 \times \vec{d}) \times \hat{n}_0}{\|(\hat{n}_0 \times \vec{d}) \times \hat{n}_0\|}$$

$$\vec{t}_1 = \|\vec{d}\| \frac{(\hat{n}_1 \times \vec{d}) \times \hat{n}_1}{\|(\hat{n}_1 \times \vec{d}) \times \hat{n}_1\|}$$

$$\vec{d} = \vec{v}_1 - \vec{v}_0$$

As a result, the newly computed $\vec{t}_0$ is in the plane formed by $\vec{d}$ and $\hat{n}_0$, and it is orthogonal to $\hat{n}_0$. Similarly, $\vec{t}_1$ is in the plane of $\vec{d}$ and $\hat{n}_1$ and perpendicular to $\hat{n}_1$. An illustration can be found in Figure 1 for this formulation.

### 2.2 Problem of AMF formulation

Now we further study the normals at endpoints of $\vec{h}(u)$ using differential geometry [20]. Differentiating once gives

$$\frac{d}{du}\vec{h}(u) = \vec{t}(u) = (6u^2 - 6u)\vec{v}_0 + (3u^2 - 4u + 1)\vec{t}_0 + (-6u^2 + 6u)\vec{v}_1 + (3u^2 - 2u)\vec{t}_1,$$

$$\frac{d}{du}\vec{h}(0) = \vec{t}_0 \qquad \frac{d}{du}\vec{h}(1) = \vec{t}_1.$$

Then, the second differentiation is applied to obtain

$$\frac{d^2}{du^2}\vec{h}(u) = 6(2u-1)\vec{v}_0 + (6u-4)\vec{t}_0 + (-12u+6)\vec{v}_1 + (6u-2)\vec{t}_1,$$

$$\frac{d^2}{du^2}\vec{h}(0) = -6\vec{v}_0 - 4\vec{t}_0 + 6\vec{v}_1 - 2\vec{t}_1 = 2\left(3\vec{d} - 2\vec{t}_0 - \vec{t}_1\right),$$

$$\frac{d^2}{du^2}\vec{h}(1) = 6\vec{v}_0 + 2\vec{t}_0 - 6\vec{v}_1 + 4\vec{t}_1 = 2\left(-3\vec{d} + \vec{t}_0 + 2\vec{t}_1\right).$$

From the differential geometry of curves, we know that the principal normals at endpoints can be obtained from the differentiation of a parametric curve as

$$\left(\frac{d}{du}\vec{h}(0) \times \frac{d^2}{du^2}\vec{h}(0)\right) \times \frac{d}{du}\vec{h}(0) = \left(\vec{t}_0 \times \left(2\left(3\vec{d} - 2\vec{t}_0 - \vec{t}_1\right)\right)\right) \times \vec{t}_0$$

$$= 2\left(3\vec{t}_0 \times \vec{d} - \vec{t}_0 \times \vec{t}_1\right) \times \vec{t}_0,$$

$$\left(\frac{d}{du}\vec{h}(1) \times \frac{d^2}{du^2}\vec{h}(1)\right) \times \frac{d}{du}\vec{h}(1) = \left(\vec{t}_1 \times \left(2\left(-3\vec{d} + \vec{t}_0 + 2\vec{t}_1\right)\right)\right) \times \vec{t}_1$$

$$= 2\left(-3\vec{t}_1 \times \vec{d} + \vec{t}_1 \times \vec{t}_0\right) \times \vec{t}_1.$$

The first terms are parallel (or anti-parallel) to inputs $\hat{n}_0$ and $\hat{n}_1$ respectively but not the second terms. That is,

$$\left(\left(\vec{t}_0 \times \vec{d}\right) \times \vec{t}_0\right) \times \hat{n}_0 = \vec{0} \text{ and } \left(\left(\vec{t}_1 \times \vec{d}\right) \times \vec{t}_1\right) \times \hat{n}_1 = \vec{0},$$

$$\left(\left(\vec{t}_0 \times \vec{t}_1\right) \times \vec{t}_0\right) \times \hat{n}_0 \neq \vec{0} \text{ and } \left(\left(\vec{t}_1 \times \vec{t}_0\right) \times \vec{t}_1\right) \times \hat{n}_1 \neq \vec{0}.$$

In short, the principal normals at the endpoints are generally not parallel (or anti-parallel) to the input normals $\hat{n}_0$ and $\hat{n}_1$ respectively.

- A vertex incidenting to *n* different vertices can have (*n*+1) different normals – *n* edges lead to *n* Hermite curves with n principal normals plus the normal defined on the vertex in the AMF file.
- When normals are parallel to chord (say $\hat{n}_0 \times \vec{d} = \vec{0}$), $\vec{t}_0$ determined by Eq.(A3.1) in [4] (with sign corrected above) is not uniquely defined any more.

Moreover, only a scheme of recursive refinement is defined in the AMF format, with which a very dense set of triangles needs to be generated when computing the smooth geometry. Industrial applications in additive manufacturing actually demand a continuous representation of the smooth geometry. To overcome all these problems, we propose an enriched scheme for computing smooth geometry in the AMF format by using triangular Bézier patch.

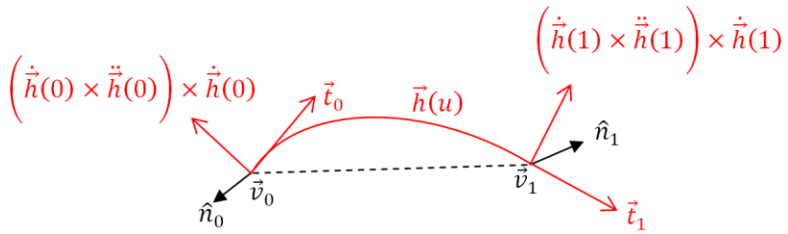


**Figure 1**: An illustration for the Hermite curve based interpolation employed in AMF format [4].

### 2.3 Smooth geometry by triangular Bézier patch

Continuous formulation by using a triangular Bézier patch will be proposed in this paper to solve the aforementioned problems of smooth geometry construction in the AMF format. Specifically, the problem to be solved is defined as follows.

**Input:** A triangle on the input model with three vertices $\vec{v}_i$, $\vec{v}_{i+1}$ and $\vec{v}_{i+2}$, and their associated normal directions $\hat{n}_i$, $\hat{n}_{i+1}$ and $\hat{n}_{i+2}$.

**Output:** A curved parametric patch $\vec{P}(u,v,w)$ interpolating the vertices and their associated normals (with the parameters $u,v,w\in[0,1]$ and $u+v+w=1$).

A deterministic scheme is proposed in Section 3 for constructing such a parametric patch $\vec{P}(u,v,w)$ by using the formulation of triangular Bézier patch. The three curve boundaries are then obtained as $\vec{P}(0,v,w)$, $\vec{P}(u,0,w)$ and $\vec{P}(u,v,0)$ (see the illustration shown in Figure 2).

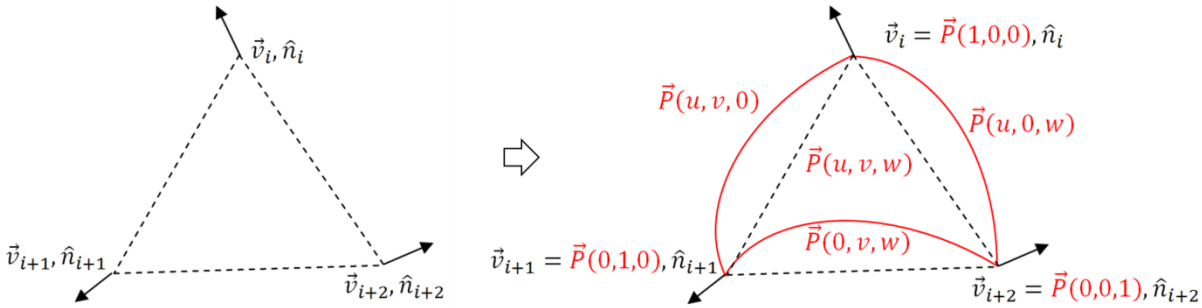


**Figure 2:** Smooth geometry to be constructed by triangular Bézier patch.

## 3. Methodology and Formulation

A triangular Bézier patch is defined as (ref. [21])

$$\vec{P}(u,v,w)=\sum_{i+j+k=n}\binom{n}{i,j,k}u^iv^jw^k\vec{b}_{ijk}$$

with the $n$-degree Bernstein polynomial $\binom{n}{i,j,k}u^iv^jw^k$ $(n=i+j+k;i,j,k\in\{0,1,\dots,n\})$, where the trinomial coefficient is $\binom{n}{i,j,k}=\frac{n!}{i!j!k!}$. The control points of this Bézier patch are $\{\vec{b}_{ijk}\}$. Specifically, the cubic triangular Bézier patch is in fact

$$\vec{P}(u,v,w) = u^3\vec{b}_{300} + v^3\vec{b}_{030} + w^3\vec{b}_{003} + 3u^2v\vec{b}_{210} + 3uv^2\vec{b}_{120} + 3v^2w\vec{b}_{021} + 3vw^2\vec{b}_{012} + 3uw^2\vec{b}_{102} + 3u^2w\vec{b}_{201} + 6uvw\vec{b}_{111},$$

as illustrated in Figure 3. Now we introduce the method to determine the control points according to the input discussed in Section 2.3.

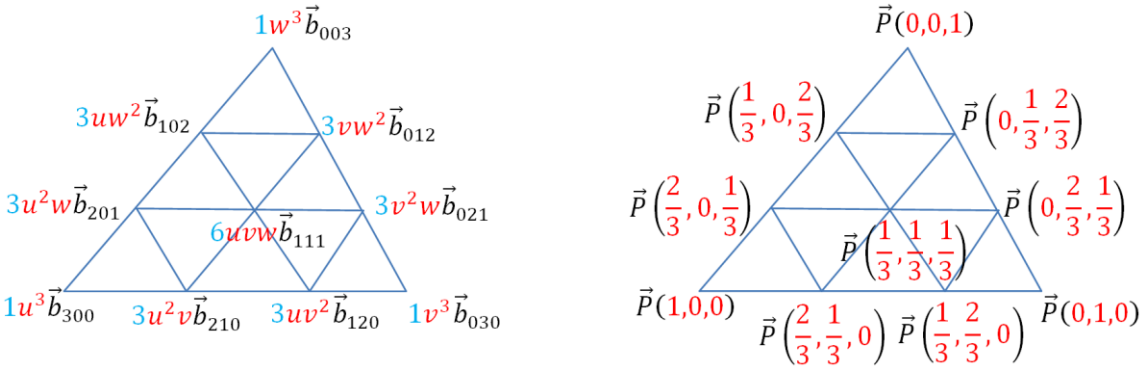


**Figure 3:** Definition of a cubic triangular Bézier patch

### 3.1 Construction of triangular Bézier patch

The construction scheme consists of two steps – the construction of quadratic triangular Bézier patch and the degree elevation from quadratic to cubic patch.

**Step 1) Construction of quadratic patch**

Three control points $\vec{b}_{300}$, $\vec{b}_{030}$ and $\vec{b}_{003}$ are actually coincident with the input vertices $\vec{v}_i$, $\vec{v}_{i+1}$ and $\vec{v}_{i+2}$ (see the red triangle in Figure 4). To obtain a quadratic Bézier patch, another three control points $\vec{b}_{011}$, $\vec{b}_{101}$ and $\vec{b}_{110}$ should be determined. To ensure the directions of normal vectors, $\hat{n}_i$, $\hat{n}_{i+1}$ and $\hat{n}_{i+2}$, are interpolated, these control points are computed using the following method.

Three planes are first constructed by the Hermite data on three vertices – that are $\pi_i(\vec{v}_i, \hat{n}_i)$, $\pi_{i+1}(\vec{v}_{i+1}, \hat{n}_{i+1})$ and $\pi_{i+2}(\vec{v}_{i+2}, \hat{n}_{i+2})$, where each plane passes through the vertex and using the specified normal at the vertex (i.e., $\hat{n}_i$, $\hat{n}_{i+1}$ and $\hat{n}_{i+2}$) as the plane's normal. To determine the position of $\vec{b}_{110}$, we compute the line AB from the intersection of $\pi_i(\vec{v}_i, \hat{n}_i)$ and $\pi_{i+1}(\vec{v}_{i+1}, \hat{n}_{i+1})$ (see Figure 4 for an illustration and the details on computation in Appendix I). The position of $\vec{b}_{110}$ is then determined by computing a point on AB that is closest to the line $\vec{b}_{300}\vec{b}_{030}$ (details of the computation can be found in Appendix II). Similarly, $\vec{b}_{011}$ is obtained on the intersection line AD of $\pi_{i+1}(\vec{v}_{i+1}, \hat{n}_{i+1})$ and $\pi_{i+2}(\vec{v}_{i+2}, \hat{n}_{i+2})$ as a point closest to the line $\vec{b}_{003}\vec{b}_{030}$. And $\vec{b}_{101}$ is a point on the intersection line AC of $\pi_{i+2}(\vec{v}_{i+2}, \hat{n}_{i+2})$ and $\pi_i(\vec{v}_i, \hat{n}_i)$ closest to the line $\vec{b}_{003}\vec{b}_{300}$. Figure 4 gives a clear illustration of the above computation.

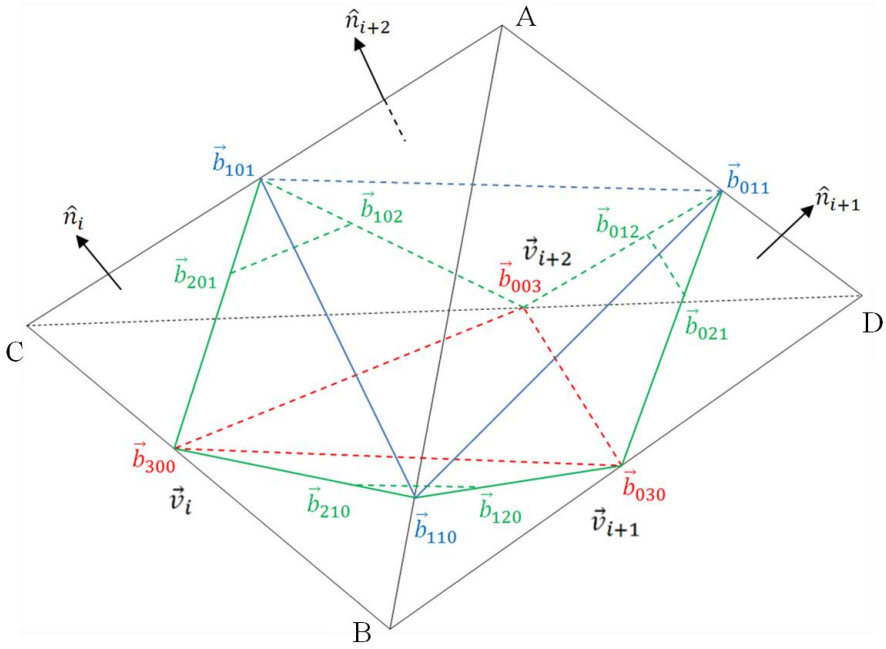


**Figure 4:** The construction polygons of triangular Bézier patch

**Step 2) Degree elevation**

The degree elevation is taken on the quadratic Bézier patch to make a cubic triangular patch. By the method proposed in [21], we can elevate quadratic Bézier triangle to cubic as follows.

$$\vec{b}_{300} = \vec{b}_{300}^{(1)} = \frac{1}{3}\left(3\vec{b}_{300-100} + 0\vec{b}_{300-010} + 0\vec{b}_{300-001}\right) = \vec{b}_{200}$$
$$\vec{b}_{030} = \vec{b}_{030}^{(1)} = \frac{1}{3}\left(0\vec{b}_{030-100} + 3\vec{b}_{030-010} + 0\vec{b}_{030-001}\right) = \vec{b}_{020}$$
$$\vec{b}_{003} = \vec{b}_{003}^{(1)} = \frac{1}{3}\left(0\vec{b}_{003-100} + 0\vec{b}_{003-010} + 3\vec{b}_{003-001}\right) = \vec{b}_{002}$$
$$\vec{b}_{210} = \vec{b}_{210}^{(1)} = \frac{1}{3}\left(2\vec{b}_{210-100} + 1\vec{b}_{210-010} + 0\vec{b}_{210-001}\right) = \frac{1}{3}\left(2\vec{b}_{110} + \vec{b}_{200}\right)$$
$$\vec{b}_{120} = \vec{b}_{120}^{(1)} = \frac{1}{3}\left(1\vec{b}_{120-100} + 2\vec{b}_{120-010} + 0\vec{b}_{120-001}\right) = \frac{1}{3}\left(\vec{b}_{020} + 2\vec{b}_{110}\right)$$
$$\vec{b}_{021} = \vec{b}_{021}^{(1)} = \frac{1}{3}\left(0\vec{b}_{021-100} + 2\vec{b}_{021-010} + 1\vec{b}_{021-001}\right) = \frac{1}{3}\left(2\vec{b}_{011} + \vec{b}_{020}\right)$$
$$\vec{b}_{012} = \vec{b}_{012}^{(1)} = \frac{1}{3}\left(0\vec{b}_{012-100} + 1\vec{b}_{012-010} + 2\vec{b}_{012-001}\right) = \frac{1}{3}\left(\vec{b}_{002} + 2\vec{b}_{011}\right)$$
$$\vec{b}_{102} = \vec{b}_{102}^{(1)} = \frac{1}{3}\left(1\vec{b}_{102-100} + 0\vec{b}_{102-010} + 2\vec{b}_{102-001}\right) = \frac{1}{3}\left(\vec{b}_{002} + 2\vec{b}_{101}\right)$$
$$\vec{b}_{201} = \vec{b}_{201}^{(1)} = \frac{1}{3}\left(2\vec{b}_{201-100} + 0\vec{b}_{201-010} + 1\vec{b}_{201-001}\right) = \frac{1}{3}\left(2\vec{b}_{101} + \vec{b}_{200}\right)$$
$$\vec{b}_{111} = \vec{b}_{111}^{(1)} = \frac{1}{3}\left(1\vec{b}_{111-100} + 1\vec{b}_{111-010} + 1\vec{b}_{111-001}\right) = \frac{1}{3}\left(\vec{b}_{011} + \vec{b}_{101} + \vec{b}_{110}\right)$$

The tenth coefficient, $\vec{b}_{111}$, has not been determined yet. It is independent of the prescribed data and can be assigned any arbitrary value. A reasonable choice is to select a position such that quadratic precision of the interpolant is maintained - i.e., if the nine prescribed data (nine-parameter interpolant)

were read off from a quadratic, the interpolant would reproduce this quadratic. If a quadratic is degree elevated to cubic, the control coefficient $\vec{b}_{111}$ can be expressed by (as suggested in [22, 23])

$$\vec{b}_{111} = \frac{1}{4}\left(\vec{b}_{210} + \vec{b}_{120} + \vec{b}_{021} + \vec{b}_{012} + \vec{b}_{102} + \vec{b}_{201}\right) - \frac{1}{6}\left(\vec{b}_{300} + \vec{b}_{030} + \vec{b}_{003}\right).$$

The cubic triangular Bézier patch constructed in this way will preserve the normal interpolation at the three vertices and $C^0$ continuity across the boundary of neighbouring patches. These nice properties are proved below.

### 3.2 Properties of constructed patch

We now analyse the properties of triangular Bézier patch constructed by our scheme.

**Remark 1** *The normal interpolation at corners is preserved by our construction scheme*.

***Proof:*** According to the theorem of triangular Bezier patch [21-23], it is known that the normal of surface $\vec{P}(u, v, w)$ at a corner is along the direction of the control triangle's normal at that corner. Specifically, from section 17.4 (pp.318) of [21], we know that

- three vertices $\vec{b}_{200}\vec{b}_{210}\vec{b}_{201}$ span the tangent plane at $\vec{b}_{200}$ on the quadratic triangular Bézier patch – here, $\vec{b}_{200} = \vec{b}_{300} = \vec{v}_i$=$\vec{P}$(1,0,0);
- three vertices $\vec{b}_{020}\vec{b}_{021}\vec{b}_{012}$ span the tangent plane at $\vec{b}_{020}$ with $\vec{b}_{020} = \vec{b}_{030} = \vec{v}_{i+1}$=$\vec{P}$(0,1,0);
- similarly, the normal at $\vec{v}_{i+2}$=$\vec{P}$(0,0,1) is along the plane normal of $\vec{b}_{003}\vec{b}_{102}\vec{b}_{012}$.

The control points $\vec{b}_{300}$, $\vec{b}_{210}$ and $\vec{b}_{201}$ are located on the plane $\pi_i(\vec{v}_i, \hat{n}_i)$ in our construction scheme (see also the illustration in Figure 4). As a result, the normal at $\vec{P}(1,0,0)$ is along the direction of $\hat{n}_i$. Similarly, the normals at $\vec{P}(0,1,0)$ and $\vec{P}(0,0,1)$ are along the directions of $\hat{n}_{i+1}$ and $\hat{n}_{i+2}$ respectively.

**Q.E.D.**

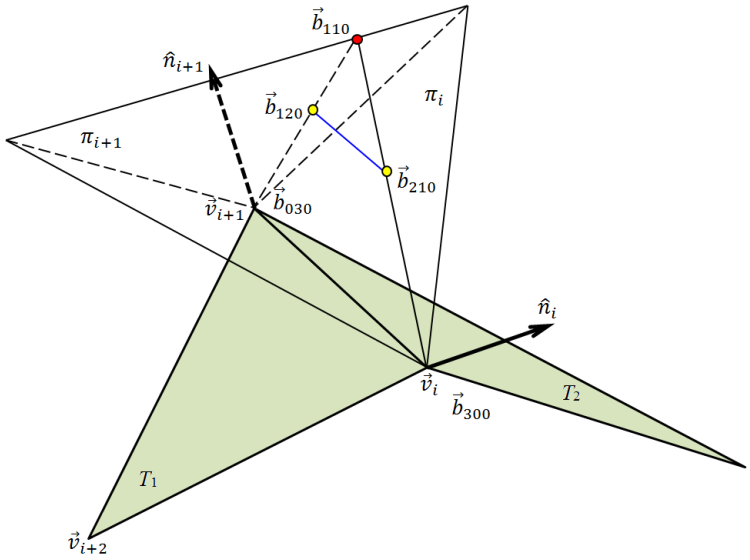


**Figure 5:** An illustration for proving the continuity between neighbouring Bézier patches.

**Remark 2** *$C^0$ continuity is preserved on the boundary curve of two neighbouring triangular Bezier patches constructed by our scheme separately*.

***Proof:*** Without loss of the generality, the boundary curve along the edge $\vec{v}_i\vec{v}_{i+1}$ is considered here. The triangle $T_1$=($\vec{v}_i\vec{v}_{i+1}\vec{v}_{i+2}$) and another triangle $T_2$ on the other side of $\vec{v}_i\vec{v}_{i+1}$ share the same edge. According to the construction scheme introduced above (see also Figure 5), the control point $\vec{b}_{110}$ in the control polygons on both $T_1$ and $T_2$ are determined on the intersection of the same two planes: $\pi_i(\vec{v}_i, \hat{n}_i)$ and $\pi_{i+1}(\vec{v}_{i+1}, \hat{n}_{i+1})$. And $\vec{b}_{110}$ is the closest point on the intersection to the edge $\vec{b}_{300}$ - $\vec{b}_{030}$, which is unique. The quadratic Bezier patches constructed on both $T_1$ and $T_2$ sharing the same control polygon on this boundary – that is $\vec{b}_{300}$-$\vec{b}_{110}$-$\vec{b}_{030}$.

Furthermore, by the formulas of degree elevation given above, it is easy to prove that the control polygon on the cubic patch – i.e., $\vec{b}_{300}$-$\vec{b}_{210}$-$\vec{b}_{120}$-$\vec{b}_{030}$ is only determined by $\vec{b}_{300}$, $\vec{b}_{110}$ and $\vec{b}_{030}$. As a result, these two neighbouring cubic Bezier triangular patches also share the same set of boundary control points. Therefore, the $C^0$ continuity can be preserved.

**Q.E.D.**

With the help of these properties guaranteed by our construction scheme of smooth geometry, the curved shape can be generated separately, patch by patch, meanwhile preserving the normal vectors specified on the vertices of an input triangular mesh. This locality allows an out-of-core and also parallel computation taken on the generation of smooth geometry. In the next section, geometric properties on boundary curves are further analysed, which helps to achieve a better understanding of the shape constructed by this scheme.

## 4. Analysis Based on Boundary-Curves

In this section, different configurations on the boundary curves in above formulation are analysed in configurations with *proper* input. Moreover, formulation is also given to construct curved patch in the configurations with *less-proper* input, which is based on the Nielson's point-opposite edge interpolation for triangular Coons patch [24].

### 4.1 Proper input

For an edge $\vec{v}_i\vec{v}_{i+1}$ with the associated normals $\hat{n}_i$ and $\hat{n}_{i+1}$ at the endpoints, the input is considered as *proper* when the planes $\pi_i(\vec{v}_i, \hat{n}_i)$ and $\pi_{i+1}(\vec{v}_{i+1}, \hat{n}_{i+1})$ have an intersection line. That means the construction scheme of triangular Bézier patch introduced above can successfully work out the quadratic control point $\vec{b}_{110}$, and then the cubic control points $\vec{b}_{210}$ and $\vec{b}_{120}$.

**Definition 3** *The proper input to construct a triangular Bézier patch needs to have* $\hat{n}_i \times \hat{n}_{i+1} \neq \vec{0}$.

There are three possible configurations when the input normals are proper:

- *Configuration 1*: The chord $\vec{v}_i\vec{v}_{i+1}$ intersecting planes $\pi_i(\vec{v}_i, \hat{n}_i)$ and $\pi_{i+1}(\vec{v}_{i+1}, \hat{n}_{i+1})$;
- *Configuration 2*: The chord $\vec{v}_i\vec{v}_{i+1}$ on plane $\pi_i(\vec{v}_i, \hat{n}_i)$;
- *Configuration 3*: The chord $\vec{v}_i\vec{v}_{i+1}$ on plane $\pi_{i+1}(\vec{v}_{i+1}, \hat{n}_{i+1})$.

Configuration 1 is as per the illustration in Figure 4, and other two configurations need to be further discussed. For Configuration 2, $\vec{b}_{110}$ determined by the method in section 3 will be coincident to $\vec{v}_{i+1}$. Similarly, $\vec{b}_{110}$ will be coincident to $\vec{v}_i$ for Configuration 3. For these two cases, we can simply place $\vec{b}_{110}$ at the middle point of $\vec{v}_i\vec{v}_{i+1}$ to avoid generating singularity.

It is also worth analysing what happens when a higher order curve is employed on the edge. Note that the curves are not unique as more than one set of end-tangents can satisfy the same case conditions. From properties of degree-*n* Bézier curve (with control points $\vec{P}_i$), tangent or first derivative depends on first/last two points are $n(\vec{P}_1 - \vec{P}_0)$ and $n(\vec{P}_n - \vec{P}_{n-1})$ respectively. Second derivative depends on first/last three points as

$$n(n-1)(\vec{P}_2 - \vec{P}_1 + \vec{P}_0 - \vec{P}_1) \quad \text{and} \quad n(n-1)(\vec{P}_{n-2} - \vec{P}_{n-1} + \vec{P}_n - \vec{P}_{n-1}).$$

As a result, $\left(\dot{\vec{P}} \times \ddot{\vec{P}}\right) \times \dot{\vec{P}}$ (or normal) also depends on first/last three points. From these, different Bézier control polygons can be drawn for different cases. The simplest possible ones are those we have used in the construction scheme.

### 4.2 Less-proper input

When the normals are given in the cases $\hat{n}_i = \hat{n}_{i+1}$ or $\hat{n}_i = -\hat{n}_{i+1}$, the two planes $\pi_i(\vec{v}_i, \hat{n}_i)$ and $\pi_{i+1}(\vec{v}_{i+1}, \hat{n}_{i+1})$ are either overlapped or parallel thus having no intersection. Such input is called *less-proper*. In such configurations, we adopt the Boolean sum [21] and the Nielson's point-opposite edge interpolation [24] for triangular Coons patch to generate the smooth geometry as a $C^0$ interpolant.

First of all, the parametric curves, $\vec{h}_1(t)$, $\vec{h}_2(t)$ and $\vec{h}_3(t)$, are constructed by using the method of ASTM AMF standard specification. Specifically, $\vec{h}_1(t)$, $\vec{h}_2(t)$ and $\vec{h}_3(t)$, are curves for the chords $\vec{v}_{i+1}\vec{v}_{i+2}$, $\vec{v}_{i+2}\vec{v}_i$ and $\vec{v}_i\vec{v}_{i+1}$ respectively (see also the illustration in Figure 6). We can now start to construct a Coons patch interpolating these boundaries by the triple Boolean sum:

$$\wp = \wp_1 \oplus \wp_2 \oplus \wp_3 = \wp_1 + \wp_2 + \wp_3 - \wp_1\wp_2 - \wp_2\wp_3 - \wp_3\wp_1 + \wp_1\wp_2\wp_3$$

Since the sum of barycentric coordinates needs to be one, i.e., $u$+$v$+$w$=1, the corner linear interpolants can be formed as

$$\wp_1\vec{P}(u,v,w) = u\vec{P}(1,0,0) + (1-u)\vec{P}\left(0, \frac{v}{v+w}, \frac{w}{v+w}\right)$$
$$\wp_2\vec{P}(u,v,w) = v\vec{P}(0,1,0) + (1-v)\vec{P}\left(\frac{u}{w+u}, 0, \frac{w}{w+u}\right)$$
$$\wp_3\vec{P}(u,v,w) = w\vec{P}(0,0,1) + (1-w)\vec{P}\left(\frac{u}{u+v}, \frac{v}{u+v}, 0\right)$$

where $\vec{P}(1,0,0)$, $\vec{P}(0,1,0)$ and $\vec{P}(0,0,1)$ stands for $\vec{v}_i$, $\vec{v}_{i+1}$ and $\vec{v}_{i+2}$ respectively, and

$$\vec{P}\left(0, \tfrac{v}{v+w}, \tfrac{w}{v+w}\right) = \vec{h}_1(t) \qquad \left(\text{with} = \tfrac{v}{v+w}\right)$$

$$\vec{P}\left(\tfrac{u}{w+u}, 0, \tfrac{w}{w+u}\right) = \vec{h}_2(t) \qquad \left(\text{with} = \tfrac{u}{w+u}\right)$$

$$\vec{P}\left(\tfrac{u}{u+v}, \tfrac{v}{u+v}, 0\right) = \vec{h}_3(t) \qquad \left(\text{with} = \tfrac{u}{u+v}\right)$$

Substituting

$$\wp_1\wp_2\vec{P}(u,v,w) = u\wp_2\vec{P}(1,0,0) + (1-u)\wp_2\vec{P}\left(0, \frac{v}{v+w}, \frac{w}{v+w}\right) = u\left[0 + \vec{P}(1,0,0)\right]$$

$$+(1-u)\left[\frac{v}{v+w}\vec{P}(0,1,0) + \frac{w}{v+w}\vec{P}(0,0,1)\right] = u\vec{P}(1,0,0) + v\vec{P}(0,1,0) + w\vec{P}(0,1,0)$$

Further substituting

$$\begin{aligned}\wp_1\wp_2\wp_3\vec{P}(u,v,w) &= u\wp_3\vec{P}(1,0,0) + v\wp_3\vec{P}(0,1,0) + w\wp_3\vec{P}(0,1,0)\\ &= u\left[0 + \vec{P}(1,0,0)\right] + v\left[0 + \vec{P}(0,1,0)\right] + w\left[0 + \vec{P}(0,1,0)\right]\\ &= u\vec{P}(1,0,0) + v\vec{P}(0,1,0) + w\vec{P}(0,1,0)\end{aligned}$$

As a result,

$$\wp_1\wp_2\vec{P}(u,v,w) = u\vec{P}(1,0,0) + v\vec{P}(0,1,0) + w\vec{P}(0,1,0) = \wp_2\wp_3\vec{P}(u,v,w) = \wp_3\wp_1\vec{P}(u,v,w)$$
$$= \wp_1\wp_2\wp_3\vec{P}(u,v,w)$$

Therefore, the final interpolant in the form of Coons patch is

$$\wp\vec{P}(u,v,w) = (1-u)\vec{P}\left(0,\frac{v}{v+w},\frac{w}{v+w}\right) + (1-v)\vec{P}\left(\frac{u}{w+u},0,\frac{w}{w+u}\right)$$
$$+ (1-w)\vec{P}\left(\frac{u}{u+v},\frac{v}{u+v},0\right) - u\vec{P}(1,0,0) - v\vec{P}(0,1,0) - w\vec{P}(0,0,1)$$

$$= (1-u)\vec{h}_1\left(\frac{v}{v+w}\right) + (1-v)\vec{h}_2\left(\frac{u}{w+u}\right) + (1-w)\vec{h}_3\left(\frac{u}{u+v}\right) - u\vec{v}_i - v\vec{v}_{i+1} - w\vec{v}_{i+2}$$

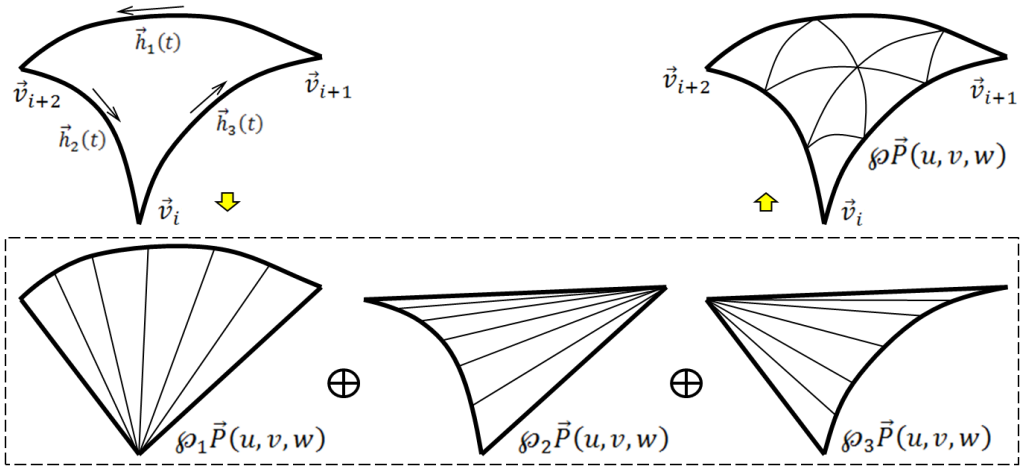


**Figure 6:** A $C^0$ interpolant – Coons patch formed by triple Boolean sum of three linear interpolants.

## 5. Conclusion and Discussion

To enrich the function of smooth geometry construction in ASTM standard specification for the AMF format, a formulation based on Bézier triangular patch is developed in this paper. This new formulation can overcome the problems caused by inconsistent normals, ambiguous tangents and recursive dense mesh generation. Smooth geometry is formulated as a continuous polynomial patch interpolating the given Hermite data (i.e., vertices and their associated normal vectors). Moreover, analysis of the boundary curves has been conducted to classify the properness of input. A $C^0$ interpolant – triangular Coons patch is given for those less-proper inputs. In summary, we consider the work presented in this paper an important enrichment on the current ASTM standard specification of the additive manufacturing file format.

In the practical aspect of implementation, the current file format of ASTM AMF does not need to be modified when using our approach. Instead, only the method for generating smooth geometry needs to be replaced by our approach. As a result, our method does not add any size loading to the current ASTM AMF standard. Also, the evaluation of triangular Bezier patch can be realised by the de Casteljau algorithm using recursive bi-linear interpolations. The computational complexity is similar to the algorithm used in the current AMF specification.

Although our formulation has enriched the current ASTM AMF specification, there are still some limitations to our method.

- First, similar to other smooth geometry construction approaches, self-intersection could be generated on an original input model formed by planar facets. No control is provided to avoid self-intersection. A post-processing may need to be applied to remove the self-intersections.
- Another difficulty on the smooth geometry is the computation of slicing, which is simple on a model only consisting of planar facets. Now, as the smooth geometry represented by polynomial patches has been introduced, more sophisticated planar to surface intersection algorithms must be developed to overcome this difficulty. This is another possible future research direction.
- Last, only $C^0$ continuity can be preserved across the boundary curves of two neighbouring patches by our scheme to determine the control points $\{\vec{b}_{ijk}\}$. As stated in [21-23], to achieve $C^1$ continuity, the neighbouring triangles of control points along a boundary curve must be coplanar and be an affine map of the two domain triangles.

We plan to study these problems in future research.

**Acknowledgements**

The work described in this paper was fully supported by a grant from The Hong Kong Polytechnic University (Project No. G-YL95).

## References

[1] L.A. Piegl, W. Tiller, "Geometry-based triangulation of trimmed NURBS surfaces", Computer-Aided Design, 30(1), pp.11-18, 1998.

[2] L.A. Piegl, A.M. Richard, "Tessellating trimmed NURBS surfaces", Computer-Aided Design, 27(1), pp.16-26, 1995.

[3] G. Navangul, R. Paul, and S. Anand, "A vertex translation algorithm for adaptive modification of STL file in layered manufacturing", *Proceedings of 2011 ASME MSEC Conference*, June 13-17, 2011.

[4] *Standard Specification for Additive Manufacturing File Format (AMF) Version 1.1*, Designation: F2915-12, ASTM International, United States, 2013.

[5] A. Vlachos, J. Peters, C. Boyd, and J.L. Mitchell, "Curved PN triangles", *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, pp.159-166.

[6] T. Boubekeur, P. Reuter, and C. Schlick, "Scalar tagged PN triangles", *Proceedings of EUROGRAPHICS Short Papers*, Sep 2005, Dublin, Ireland.

[7] A. Myles, T. Ni, and J. Peters, "Fast parallel construction of smooth surfaces from meshes with tri/quad/pent Facets", Computer Graphics Forum, 27(5), 2008.

[8] C. Loop, S. Schaefer, T. Ni, and J. Peters, "Approximating subdivision surfaces with Gregory patches for hardware tessellation", ACM Transactions on Graphics, 28(5), Article 151, 9 pages, 2009.

[9] Y.-S. Leung, C.C.L. Wang, and Y. Zhang, "Localized construction of curved surfaces from polygon meshes: a simple and practical approach on GPU", Computer-Aided Design, 43(6), pp.573-585, 2011.

[10] A. Yvart, S. Hahmann, and G.-P. Bonneau, "Hierarchical triangular splines", ACM Trans. Graph., 24(4), pp.1374-1391, 2005.

[11] M. Siqueira, D. Xu, J. Gallier, L.G. Nonato, D.M. Morera, and L. Velho, "A new construction of smooth surfaces from triangle meshes using parametric pseudo-manifolds", Computers & Graphics, 33(3), pp.331-340, 2009.

[12] M. Boschiroli, C. Fünfzig, L. Romani, and G. Albrecht, "A comparison of local parametric C0 Bézier interpolants for triangular meshes", Computers & Graphics, 35(1), p.20-34, 2011.

[13] H. Theisel, C. Rossi, R. Zayer, and H.-P. Seidel, "Normal based estimation of the curvature tensor for triangular meshes", *Proceedings of the 12th Pacific Conference on Computer Graphics and Applications*, pp.288-297, 2004.

[14] M. Müller and N. Chentanez, "Wrinkle meshes", *Proceedings of the 2010 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, pp.85-92.

[15] G. Sun, "A digital mock-up visualization system capable of processing giga-scale CAD models", Computer-Aided Design, 39, pp.133-141, 2007.

[16] G. Navangul, *Stereolithography (STL) file modification by vertex translation algorithm (VTA) for precision layered manufacturing*, MS Thesis, 2011.

[17] P. Huang, C.C.L. Wang, and Y. Chen, "Intersection-free and topologically faithful slicing of implicit solid", ASME Journal of Computing and Information Science in Engineering, 13(2), 021009 (13 pages), June 2013.

[18] A.R.S. Santosh, R. Paul, and S. Anand, "A new additive manufacturing file format using Bezier patches", *Proceedings of the 41st NAMRC*, June 10-14, 2013.

[19] R. Paul, and S. Anand, "A new Steiner patch based file format for Additive Manufacturing processes", Computer-Aided Design, 63, pp.86-100, 2015.

[20] DoCarmo M., *Differential Geometry of Curves and Surfaces*, Prentice-Hall, Englewood Cliffs, NJ, 1976.

[21] G. Farin, *Curves and Surfaces for CAGD: A Practical Guide*, the 5th Edition, Morgan-Kaufmann, 2002.

[22] G. Farin, “Smooth interpolation to scattered 3D data”, In Robert E. Barnhill and Wolfgang Boehm, editors, *Surfaces in Computer-Aided Geometric Design*, pages 43–63. North-Holland, 1983.

[23] G. Farin, “Triangular Bernstein-Bézier patches”, Computer Aided Geometric Design, 3, pp.83-127, 1986.

[24] G.M. Nielson, “The side-vertex method for interpolation in triangles”, Journal of Approximation Theory, 25(4), pp.318-336, 1979.

## Appendix I Straight line of intersection between two planes

For two planes

$\pi_0$: $\left(\vec{P} - \vec{P}_0\right) \cdot \hat{n}_0 = 0$

$\pi_1$: $\left(\vec{P} - \vec{P}_1\right) \cdot \hat{n}_1 = 0$

The straight line of intersection between two planes $\pi_0$ and $\pi_1$ must be perpendicular to both plane normals, and so parallel to their cross product $\hat{n}_0 \times \hat{n}_1$ (this cross product is zero if, and only if, the planes are parallel, and are therefore non-intersecting or entirely coincident).

Since $\{\hat{n}_0, \hat{n}_1, \hat{n}_0 \times \hat{n}_1\}$ is a basis (linearly independent or scalar triple product $(\hat{n}_0 \hat{n}_1 \hat{n}_0 \times \hat{n}_1) = 0$). Any point in space can be written as

$$\vec{P} = c_0 \hat{n}_0 + c_1 \hat{n}_1 + c_2 (\hat{n}_0 \times \hat{n}_1)$$

Substituting into the planes,

$$\vec{P} \cdot \hat{n}_0 = c_0 \hat{n}_0 \cdot \hat{n}_0 + c_1 \hat{n}_1 \cdot \hat{n}_0 + c_2 (\hat{n}_0 \times \hat{n}_1) \cdot \hat{n}_0 = \vec{P}_0 \cdot \hat{n}_0$$

$$\vec{P} \cdot \hat{n}_1 = c_0 \hat{n}_0 \cdot \hat{n}_1 + c_1 \hat{n}_1 \cdot \hat{n}_1 + c_2 (\hat{n}_0 \times \hat{n}_1) \cdot \hat{n}_1 = \vec{P}_1 \cdot \hat{n}_1$$

Since $(\hat{n}_0 \times \hat{n}_1) \cdot \hat{n}_0 = 0 = (\hat{n}_0 \times \hat{n}_1) \cdot \hat{n}_1$, therefore

$$c_0 \hat{n}_0 \cdot \hat{n}_0 + c_1 \hat{n}_1 \cdot \hat{n}_0 = \vec{P}_0 \cdot \hat{n}_0$$

$$c_0 \hat{n}_0 \cdot \hat{n}_1 + c_1 \hat{n}_1 \cdot \hat{n}_1 = \vec{P}_1 \cdot \hat{n}_1$$

Or

$$c_0 + c_1 \hat{n}_1 \cdot \hat{n}_0 = \vec{P}_0 \cdot \hat{n}_0$$

$$c_0 \hat{n}_0 \cdot \hat{n}_1 + c_1 = \vec{P}_1 \cdot \hat{n}_1$$

Or

$$c_0 = \frac{\begin{vmatrix} \vec{P}_0 \cdot \hat{n}_0 & \hat{n}_0 \cdot \hat{n}_1 \\ \vec{P}_1 \cdot \hat{n}_1 & 1 \end{vmatrix}}{\begin{vmatrix} 1 & \hat{n}_0 \cdot \hat{n}_1 \\ \hat{n}_0 \cdot \hat{n}_1 & 1 \end{vmatrix}} \;\&\; c_1 = \frac{\begin{vmatrix} 1 & \vec{P}_0 \cdot \hat{n}_0 \\ \hat{n}_0 \cdot \hat{n}_1 & \vec{P}_1 \cdot \hat{n}_1 \end{vmatrix}}{\begin{vmatrix} 1 & \hat{n}_0 \cdot \hat{n}_1 \\ \hat{n}_0 \cdot \hat{n}_1 & 1 \end{vmatrix}}$$

Therefore, the straight line of intersection between two planes $\pi_0$ and $\pi_1$ is

$$\vec{P}(u) = \frac{\begin{vmatrix} \vec{P}_0 \cdot \hat{n}_0 & \hat{n}_0 \cdot \hat{n}_1 \\ \vec{P}_1 \cdot \hat{n}_1 & 1 \end{vmatrix}}{\begin{vmatrix} 1 & \hat{n}_0 \cdot \hat{n}_1 \\ \hat{n}_0 \cdot \hat{n}_1 & 1 \end{vmatrix}} \hat{n}_0 + \frac{\begin{vmatrix} 1 & \vec{P}_0 \cdot \hat{n}_0 \\ \hat{n}_0 \cdot \hat{n}_1 & \vec{P}_1 \cdot \hat{n}_1 \end{vmatrix}}{\begin{vmatrix} 1 & \hat{n}_0 \cdot \hat{n}_1 \\ \hat{n}_0 \cdot \hat{n}_1 & 1 \end{vmatrix}} \hat{n}_1 + u(\hat{n}_0 \times \hat{n}_1)$$

where parameter $u \in \mathbb{R}$. Or the line can be represented in an alternative form as

$$\vec{P}(u) = \frac{\vec{P}_0 \cdot \hat{n}_0[\hat{n}_1 \times (\hat{n}_0 \times \hat{n}_1)] + \vec{P}_1 \cdot \hat{n}_1[\hat{n}_0 \times (\hat{n}_1 \times \hat{n}_0)]}{(\hat{n}_0 \times \hat{n}_1) \cdot (\hat{n}_0 \times \hat{n}_1)} + u(\hat{n}_0 \times \hat{n}_1).$$

In addition, the following notes can be summarised.

1. *unique* straight line of intersection

2. have solution when point $\vec{P}_1$ lies on plane $\vec{P}_0\hat{n}_0$ or point $\vec{P}_0$ lies on plane $\vec{P}_1\hat{n}_1$, i.e.,

$$\left(\vec{P}_1 - \vec{P}_0\right) \cdot \hat{n}_0 = 0 \Longrightarrow \vec{P}_1 \cdot \hat{n}_0 = \vec{P}_0 \cdot \hat{n}_0$$

or

$$\left(\vec{P}_0 - \vec{P}_1\right) \cdot \hat{n}_1 = 0 \Longrightarrow \vec{P}_1 \cdot \hat{n}_1 = \vec{P}_0 \cdot \hat{n}_1$$

3.

$$\frac{\begin{vmatrix} \vec{P}_0 \cdot \hat{n}_0 & \hat{n}_0 \cdot \hat{n}_1 \\ \vec{P}_1 \cdot \hat{n}_1 & 1 \end{vmatrix}}{\begin{vmatrix} 1 & \hat{n}_0 \cdot \hat{n}_1 \\ \hat{n}_0 \cdot \hat{n}_1 & 1 \end{vmatrix}} \hat{n}_0 + \frac{\begin{vmatrix} 1 & \vec{P}_0 \cdot \hat{n}_0 \\ \hat{n}_0 \cdot \hat{n}_1 & \vec{P}_1 \cdot \hat{n}_1 \end{vmatrix}}{\begin{vmatrix} 1 & \hat{n}_0 \cdot \hat{n}_1 \\ \hat{n}_0 \cdot \hat{n}_1 & 1 \end{vmatrix}} \hat{n}_1 = \frac{\vec{P}_0 \cdot \hat{n}_0[\hat{n}_1 \times (\hat{n}_0 \times \hat{n}_1)] + \vec{P}_1 \cdot \hat{n}_1[\hat{n}_0 \times (\hat{n}_1 \times \hat{n}_0)]}{(\hat{n}_0 \times \hat{n}_1) \cdot (\hat{n}_0 \times \hat{n}_1)}$$

since

$$1 - (\hat{n}_0 \cdot \hat{n}_1)^2 = sin^2\theta = (\hat{n}_0 \times \hat{n}_1) \cdot (\hat{n}_0 \times \hat{n}_1)$$

where θ is the angle subtended between $\hat{n}_0$ & $\hat{n}_1$.

Also

$$\begin{vmatrix} \vec{P}_0 \cdot \hat{n}_0 & \hat{n}_0 \cdot \hat{n}_1 \\ \vec{P}_1 \cdot \hat{n}_1 & 1 \end{vmatrix} \hat{n}_0 + \begin{vmatrix} 1 & \vec{P}_0 \cdot \hat{n}_0 \\ \hat{n}_0 \cdot \hat{n}_1 & \vec{P}_1 \cdot \hat{n}_1 \end{vmatrix} \hat{n}_1 = \vec{P}_0 \cdot \hat{n}_0[\hat{n}_1 \times (\hat{n}_0 \times \hat{n}_1)] + \vec{P}_1 \cdot \hat{n}_1[\hat{n}_0 \times (\hat{n}_1 \times \hat{n}_0)]$$

## Appendix II Closest/nearest points & shortest distance between two skew straight lines

To find the shortest distance between two skew lines (lines which do not meet) you first require the direction of the common perpendicular. Express this vector as a unit vector and then find the component in this direction of a line joining ANY two points on the two lines. This component will be the shortest distance between the lines.

*Line 1*: $\vec{P} = \vec{P}_0 + u\hat{t}_0$

*Line 2*: $\vec{P} = \vec{P}_1 + u\hat{t}_1$

where parameter u ∈ ℝ.

Conditions for skew:

- Intersecting (i.e., not parallel or anti-parallel)

$$\hat{t}_1 \nparallel \hat{t}_0 \vee \hat{t}_1 \nparallel -\hat{t}_0 \iff \hat{t}_0 \cdot \pm\hat{t}_1 \neq \pm 1$$

$$\hat{t}_1 \nparallel \hat{t}_0 \vee \hat{t}_1 \nparallel -\hat{t}_0 \iff \hat{t}_1 \times \hat{t}_0 \neq \vec{0} \iff \|\hat{t}_1 \times \hat{t}_0\| \neq 0$$

- not coplanar

$$\left(\vec{P}_1 - \vec{P}_0\right) \cdot \hat{t}_1 \times \hat{t}_0 \neq 0$$

To find the points with minimum distance is equivalent to finding $\vec{P}(u_0) = \vec{Q}_0$ and $\vec{P}(u_1) = \vec{Q}_1$.

$$\vec{Q}_0 = \vec{P}_0 + u_0\hat{t}_0$$

$$\vec{Q}_1 = \vec{P}_1 + u_1\hat{t}_1$$

From Figure, $\vec{Q}_1 - \vec{Q}_0 \parallel \hat{t}_1 \times \hat{t}_0$

$$\vec{Q}_1 - \vec{Q}_0 = \lambda\hat{t}_1 \times \hat{t}_0 = \left(\vec{P}_1 - \vec{P}_0\right) + u_1\hat{t}_1 - u_0\hat{t}_0$$

$$\lambda \in \mathbb{R}$$

Dot product multiplication by the tangents,

$$\lambda\hat{t}_1 \times \hat{t}_0 \cdot \hat{t}_0 = \left(\vec{P}_1 - \vec{P}_0\right) \cdot \hat{t}_0 + u_1\hat{t}_1 \cdot \hat{t}_0 - u_0\hat{t}_0 \cdot \hat{t}_0$$

$$\lambda\hat{t}_1 \times \hat{t}_0 \cdot \hat{t}_1 = \left(\vec{P}_1 - \vec{P}_0\right) \cdot \hat{t}_1 + u_1\hat{t}_1 \cdot \hat{t}_1 - u_0\hat{t}_0 \cdot \hat{t}_1$$

Using scalar triple product property,

$$0 = \left(\vec{P}_1 - \vec{P}_0\right) \cdot \hat{t}_0 + u_1\hat{t}_1 \cdot \hat{t}_0 - u_0$$

$$0 = \left(\vec{P}_1 - \vec{P}_0\right) \cdot \hat{t}_1 + u_1 - u_0\hat{t}_0 \cdot \hat{t}_1$$

Using Cramer's rule, we have

$$u_0 = \frac{\begin{vmatrix} -\left(\vec{P}_1 - \vec{P}_0\right) \cdot \hat{t}_0 & \hat{t}_0 \cdot \hat{t}_1 \\ -\left(\vec{P}_1 - \vec{P}_0\right) \cdot \hat{t}_1 & 1 \end{vmatrix}}{\begin{vmatrix} -1 & \hat{t}_0 \cdot \hat{t}_1 \\ -\hat{t}_0 \cdot \hat{t}_1 & 1 \end{vmatrix}} = \frac{\left(\vec{P}_1 - \vec{P}_0\right) \cdot \hat{t}_1\hat{t}_0 \cdot \hat{t}_1 - \left(\vec{P}_1 - \vec{P}_0\right) \cdot \hat{t}_0}{(\hat{t}_0 \cdot \hat{t}_1)^2 - 1}$$

$$u_1 = \frac{\begin{vmatrix} -1 & -\left(\vec{P}_1 - \vec{P}_0\right) \cdot \hat{t}_0 \\ -\hat{t}_0 \cdot \hat{t}_1 & -\left(\vec{P}_1 - \vec{P}_0\right) \cdot \hat{t}_1 \end{vmatrix}}{\begin{vmatrix} -1 & \hat{t}_0 \cdot \hat{t}_1 \\ -\hat{t}_0 \cdot \hat{t}_1 & 1 \end{vmatrix}} = \frac{\left(\vec{P}_1 - \vec{P}_0\right) \cdot \hat{t}_1 - \left(\vec{P}_1 - \vec{P}_0\right) \cdot \hat{t}_0\hat{t}_0 \cdot \hat{t}_1}{(\hat{t}_0 \cdot \hat{t}_1)^2 - 1}$$

Or

$$\vec{Q}_0 = \vec{P}_0 + \frac{\left(\vec{P}_1 - \vec{P}_0\right) \cdot \hat{t}_1\hat{t}_0 \cdot \hat{t}_1 - \left(\vec{P}_1 - \vec{P}_0\right) \cdot \hat{t}_0}{(\hat{t}_0 \cdot \hat{t}_1)^2 - 1}\hat{t}_0$$

$$\vec{Q}_1 = \vec{P}_1 + \frac{\left(\vec{P}_1 - \vec{P}_0\right) \cdot \hat{t}_1 - \left(\vec{P}_1 - \vec{P}_0\right) \cdot \hat{t}_0\hat{t}_0 \cdot \hat{t}_1}{(\hat{t}_0 \cdot \hat{t}_1)^2 - 1}\hat{t}_1$$

Note that there is a unique closest point. And the minimum distance is

$$d = \left| (\vec{P}_1 - \vec{P}_0) \cdot \frac{\hat{t}_1 \times \hat{t}_0}{\|\hat{t}_1 \times \hat{t}_0\|} \right|$$

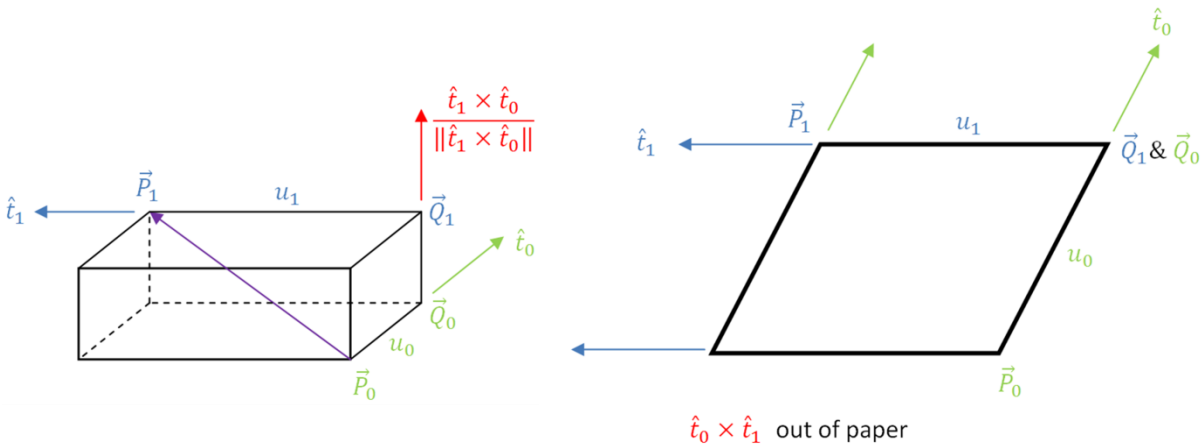


**Figure A:** An illustration for computing the nearest points and shortest distance between two skew straight lines.