Journal of Fiber Bioengineering and Informatics 8:2 (2015) 301–308
doi:10.3993/jfbim00128

A Parallel Interpolation Subdivision Scheme for Curve and Surface Design ⋆

Aihua Mao [a], Jie Luo [b,*], Jun Chen [a]

[a] *School of Computer Science & Engineering, South China University of Technology Guangzhou 510006, China*

[b] *School of Fine Art and Artistic Design, Guangzhou University, Guangzhou 510006, China*

Abstract

This paper proposes a new scheme of interpolation subdivision curve and surface using Bezier curve to achieve more realistic surface design. Different from the previous interpolation subdivision schemes, the normal vectors are used to generate a circle from a triangle which results in only three vertices and three edges. This improvement not only makes a curve smooth, but also produces sharp angles during subdivision. Examples of subdivision surface by this scheme are illustrated and compared to show the advantages of this scheme.

Keywords: Interpolation Subdivision; Normal Vectors; Bezier Curve; Parallel Computation

1 Introduction

In the field of 3D computer graphics, subdivision surface is used to represent a smooth surface via the specification of a coarser piecewise linear polygon mesh [1,2]. It can be broadly classified into two categories: approximating and interpolating [3,4]. Approximating schemes adjust the original position of vertices as needed. In general, approximating schemes have greater smoothness. The first interpolating scheme was presented by Dyn, Levin and Gregory (1990), which was also called butterfly scheme [5,6]. They extended the four-point interpolation subdivision scheme for curves to a subdivision scheme for surface. In general, interpolating subdivision schemes are sensitive to the presence of sharp features and may produce low quality surfaces for some input meshes [7]. In recently years, some new interpolating schemes have been proposed, such as the variational subdivision scheme [8] and normal-based subdivision scheme [9]. Based on the CatmullCClark subdivision scheme, Zheng and Cai proposed a two-phase subdivision scheme to interpolate arbitrary topology meshes [10]. A similar method was also applied to the DooCSabin subdivision

⋆This paper is financially supported by the Science and Technology Project of Guangdong Province (No. 2013B021600011), the Science and Technology Project of Guangzhou City (No. 2014J4100158) and the MOE (Ministry of Education in China) Project of Humanities and Social Sciences (No. 13YJC890027), and the Fundamental Research Funds for the Central Universities.

*Corresponding author.
Email address: jieluo@gzhu.edu.cn (Jie Luo).

1940–8676 / Copyright © 2015 Binary Information Press & Textile Bioengineering and Informatics Society
June 2015

scheme [11]. More recently, surface interpolation based on similarity [12] and an efficient constructive algorithm [13] are presented for interpolating meshes by Catmull-Clark surfaces. Chen and Luo [14] have used normal vector and a linear system equations to subdivide surface scheme.

Though NURBS can provide a smooth surface with some control points, it can't be used for an arbitrary mesh, such as a triangle mesh. We utilize the ideas from the NURBS that using the Bezier curve to generate point. Since the 4-point subdivision curve and deducing interpolating subdivision scheme in [15] are both difficult to generate a circle from a square or a triangle, we refresh the mind to propose a scheme to make a curve smooth and circinal, then extend this scheme to the subdivision surface scheme. Meanwhile, it is difficult for the proposed interpolation schemes in [5,10,15] to generate a sphere from a tetrahedron, which has only 4 vertices and 4 faces. In the existing schemes, many have much computational cost of solving a large system of linear equations [15], or change the topology of the mesh [4] or modify the data of the vertices [1–3]. They need extra space to store data or spend much time during computation.

In order to improve these disadvantages, this paper proposes a parallel normal-based subdivision scheme in the simple way for interpolation subdivision curve, and further expands it for surface design. Firstly, we propose an interpolation subdivision scheme, in which the input parameters are very simple with only the position and the normal vector of the two endpoints of the edge. Then, we use the normal vector to construct the Bezier controls, and subdivide the sub-edges recursively to get the limit subdivision curve. The examples of subdivision results of the well-know schemes and our scheme are presented to show the advantages of our method.

2 Circle Generation from Cubic Bezier Curve

Bezier curves are widely used in computer graphics to model smooth curves. As the curve is completely contained in the convex hull of its control vertices, the vertices can be graphically displayed and used to manipulate the curve intuitively. Any series of any 4 distinct vertices can be converted to a cubic Bezier curve that goes through all 4 vertices in order. Let $P_i = (x_i, y_i, z_i), (0 \le i \le 3)$ are the four control vertices, then the explicit matrix form is

$$P = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} P_0 \\ P_1 \\ P_2 \\ P_3 \end{bmatrix}; \begin{cases} P_0 = (\sin(-\theta), \cos(-\theta)) \\ P_1 - P_0 = (\cos(\theta), \sin(\theta))L \\ P_3 = (\sin(\theta), \cos(\theta)) \\ P_3 - P_2 = (\cos(\theta), -\sin(\theta))L' \end{cases} \tag{1}$$

When $t = 0$, $P = P_0$, and $T = 3(P_1 - P_0)$, which is the beginning point P_0, and going toward P_1; When $t = 1$, $P = P_3$, and $T = 3(P_3 - P_2)$, which is the destination point P_3, and coming from the direction of P_2; When $t = 0.5$, $P = \frac{1}{8}(P_0 + 3P_1 + 3P_2 + P_3)$, and $T = \frac{3}{4}(P_2 + P_3 - P_0 - P_1)$, which is the middle point of this curve, and it is important to generate the new interpolate point and this point can split a edge into 2 sub-edges. Assume we are going to make the Bezier curve to be an arc of a unit circle and its parametric form is $P = (\sin(u), \cos(u)), u \in [-\theta, \theta], \theta \in (0, \frac{\pi}{2}]$, then the direction of its tangent is $T = \frac{dP}{du} = (\cos(u), -\sin(u))$. The endpoint P_i can be calculated with the form of θ.

Since this curve is symmetric to y-axis, thus $L' = L$, and the curve will be in the form of:

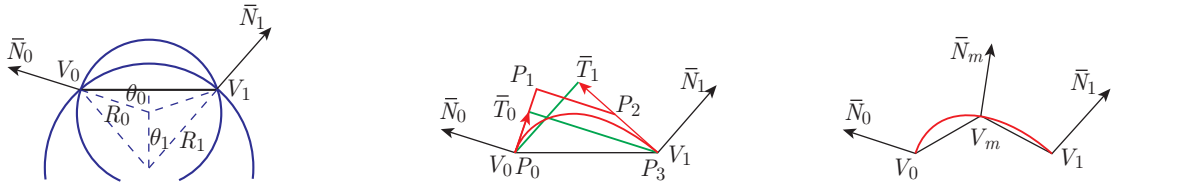
$$P = P(\theta, L, t) = \begin{bmatrix} t^3 & t^2 & t & 1 \end{bmatrix} \left(\begin{bmatrix} -4\sin\theta & 0 \\ 6\sin\theta & 0 \\ 0 & 0 \\ -\sin\theta & \cos\theta \end{bmatrix} + L \begin{bmatrix} 6\cos\theta & 0 \\ -9\cos\theta & -3\sin\theta \\ 3\cos\theta & 3\sin\theta \\ 0 & 0 \end{bmatrix} \right) \quad (2)$$

If $(P_x^2(\theta, L, t) + P_y^2(\theta, L, t) - 1)^2$ is smaller, $P(\theta, L, t)$ is closer to the unit circle. Thus, it is able to get the most approximate sphere when minimizing the value of $I(\theta, L) = \int_0^1 (P_x^2(\theta, L, t) + P_y^2(\theta, L, t) - 1)^2 dt$.

3 Plane Curve Subdivision

This section is going to propose an interpolation subdivision curve scheme. It will be easy to find that this scheme is simple and can be realize in parallel program. Before subdivision, it should check whether initial normal vectors are given. If not, the normal vectors can be generated automatically for each vertex. It is possible to define any arbitrary vectors as the normal vectors, and different normal vectors will lead to different shape.

With the normal vectors, it is possible to calculate and draw each part of the curve (shown in Fig. 1). Algorithm 1 is going to calculate the middle points of the edges, the partition of the 2 sub-edges.



(a) Step 1: Find the radius of the circles (b) Step 2: Get the 4 control points (c) Step 3: Calculate the middle point

Fig. 1: The steps of calculating the middle point

Table 1: Algorithm 1 for division of the line V_0V_1 with their normal vectors

Input: two vertices and their normal vectors $(V_0, \overrightarrow{N_0}, V_1, \overrightarrow{N_1})$, and a counter for level
Step 1: if Level < 1, then draw the line V_0V_1 directly and return. Otherwise, go to Step 2;
Step 2: get the 4 control vertices of the Bezier curve;
Step 3: get the middle vertices of the Bezier curve and its tangent vector $(V_{01}, \overrightarrow{N_{01}})$;
Step 4: get the normal vector $(\overrightarrow{N_m})$ at the middle point (V_m);
Step 5: Level = Level $-$ 1, and divide the line V_0V_{01}, and $V_{10}V_1$ respectively

4 Surface Subdivision

With the curve subdivision scheme, it can be generalized for surface interpolation of mesh directly. The meshes may be composed of triangle meshes or quadrilateral meshes. We donate each triangle

or quadrilateral on the mesh as a surface element. During the surface subdivision, only the curve subdivision scheme is utilized to generate the middle points for each edge of the surface element. Further, the middle points can subdivide a surface element into 4 sub-surface elements again. In this way, we recursively subdivide the sub-surface elements to generate the limit surface.

Algorithm 2 is the basis for the subdivision of a triangle surface and a quadrilateral surface. This scheme is an extension from the subdivision of a plane curve scheme. In this algorithm, Step 1 and Step 2 are the same as the Step 2 and Step 3 in Alogrithm1 for the subdivision of a plane curve. However, Step 3 here will be quite different, in which the detail will be given below. Get the normal plane of $V_0, \overrightarrow{N_0}$, whose normal vector is named as $\overrightarrow{F_0}$. Then get the normal plane of $V_1, \overrightarrow{N_1}$, whose normal vector is named as $\overrightarrow{F_1}$, we have $\overrightarrow{F_0} = \frac{\overrightarrow{N_0} \times \overrightarrow{V}}{|\overrightarrow{N_0} \times \overrightarrow{V}|}, \overrightarrow{F_1} = \frac{\overrightarrow{N_1} \times \overrightarrow{V}}{|\overrightarrow{N_1} \times \overrightarrow{V}|}$. Get the normal vector of the normal plane of V_m as $\overrightarrow{F_m}$, thus $\overrightarrow{F_m} = \frac{\overrightarrow{F_0} + \overrightarrow{F_1}}{|\overrightarrow{F_0} + \overrightarrow{F_1}|}$. Finally, the normal $(\overrightarrow{N_m})$ at the middle point (V_m) is $\overrightarrow{N_m} = \frac{\overrightarrow{T_m} \times \overrightarrow{F_m}}{|\overrightarrow{T_m} \times \overrightarrow{F_m}|}$.

Table 2: Algorithm 2 for getting new interpolation point from the line V_0V_1 with their normal vectors

Input: two vertices and their normal vectors $(V_0, \overrightarrow{N_0}, V_1, \overrightarrow{N_1})$
Step 1: get the 4 control vertices of the Bezier curve;
Step 2: get the middle vertices of the Bezier curve and its tangent vector$(V_m, \overrightarrow{T_m})$;
Step 3: get the normal vector $(\overrightarrow{N_m})$ at the middle point (V_m).

It is easy to know that the middle point and its vector only have the relationship with the two vertices of the edge. In another way, any edge always generates the same middle point and its normal vector. The following parts describe the detail algorithms for the subdivision of triangle surface and quadrilateral surface.

4.1 Subdivision of Triangles

Algorithm 3 is designed for subdivision of a triangle which is mostly based on Algorithm 2. The difference is that we render some local triangles first before the calculation of the final subdivision surface. This property allows using parallel calculation.

Table 3: Algorithm 3 for subdivision of a triangle

Input: The vertices of the triangle V_0, V_1, V_2, and their normal vectors $\overrightarrow{N_0}, \overrightarrow{N_1}, \overrightarrow{N_2}$, and a level counter
Step 1: if Level < 1, then draw triangle directly and stop subdivision, otherwise go to Step 2;
Step 2: call algorithm 1 to get the middle points of V_0V_1, V_1V_2, V_2V_0, and name them V_{01}, V_{12}, V_{20}, and get their normal vectors;
Step 3: Level = Level $-$ 1, Call the algorithm 3 recursively and draw triangles $V_0V_{01}V_{20}$, $V_1V_{01}V_{12}$, $V_2V_{12}V_{20}$, $V_{01}V_{12}V_{20}$.

Finally, in a triangle mesh, if two triangles share the same edge and the endpoints of the edge shares the same normal vectors, they will share the same middle point of the same edge, as well as the same normal vector at the same middle point. Thus it has no need to consider the meshes adjacent to the target triangle. Moreover, we will not change the topology of the meshes

or update any data of the vertices. That means when calculating the triangle, it does not need concern whether other sub-triangles have effect on this triangle. The only connection between these triangles is the common vertices and normal vectors. Therefore, this scheme can be realized in parallel by programming, which can achieve great faster speed of calculation.

4.2 Subdivision of Quadrilaterals

Similar to Algorithm 3, Algorithm 4 is also based on Algorithm 2. We render some local quadrilaterals at first before calculating the final subdivision surface. This feature also allows using parallel calculation.

Table 4: Algorithm 4 for subdivision of a quadrilateral

Input: The vertices of the quadrilateral V_0, V_1, V_2, V_3, and their normal vectors $\overrightarrow{N_0}, \overrightarrow{N_1}, \overrightarrow{N_2}, \overrightarrow{N_3}$, and a level counter
Step 1: if Level < 1, then draw triangle directly and stop subdivision, otherwise go to Step 2;
Step 2: call algorithm 2 to get the middle points of $V_0V_1, V_1V_2, V_2V_3, V_3V_0$, and name them $V_{01}, V_{12}, V_{23}, V_{30}$, and get their normal vectors;
Step 3: Get the middle point of V_{01}, V_{23}, naming it M_0, and get the tangent vector $\overrightarrow{T_0}$, Get the middle point of V_{12}, V_{30}, naming it M_1, and get the tangent vector $\overrightarrow{T_1}$, Set the center point $O = \frac{1}{2}(M_0 + M_1)$, and its normal vector $\overrightarrow{N_0} = \frac{\overrightarrow{T_0} \times \overrightarrow{T_1}}{\lvert\overrightarrow{T_0} \times \overrightarrow{T_1}\rvert}$;
Step 4: Level = Level $-$ 1, Call the algorithm 4 recursively, and draw quadrilaterals $OV_{30}V_0V_{01}$, $OV_{01}V_1V_{12}$, $OV_{12}V_2V_{23}$, $OV_{23}V_3V_{30}$.

Similar to the subdivision of a triangle mesh, if two quadrilaterals share the same edge, and the endpoints of the edge shares the same normal vectors, they will share the same middle point at the same edge as well. The final subdivision surface of $OV_{01}V_1V_{12}$ has no relationships with the others quadrilaterals. Therefore this algorithm for subdivision scheme can also be realized in parallel program.

5 Examples and Comparisons

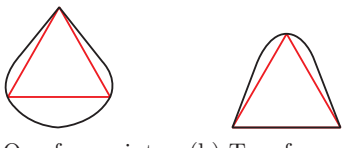
5.1 Plane Curve Subdivision

Algorithm 1 is able to generate a circle from a regular triangle and square, and the results are demonstrated in Fig. 4. In Fig. 2 (a), the red line is the initial triangle, which is in level 0; the green line is a right hexagon, which is in level 1; and the black line is a regular polygon with 48



(a) Triangle to circle (b) Square to circle

Fig. 2: Generation of a circle



(a) One free point (b) Two free point

Fig. 3: Subdivision curves with shape feature

edges, which is in level 4. The black line is quite similar to a circle. In Fig. 2 (b), red line is a square in level 0 initially; the green line is a regular octagon in level 1; while the black line is a regular polygon with 64 edges, which is in level 4. The black line is quite similar to a circle too. The final subdivision curves of these two shapes are both precise circles. In Fig. 3, it shows two sharp feature shapes from two same triangles. The difference between them is raised by their different normal vectors. Fig. 3 (a) has a free point on the top while Fig. 3 (b) has two free points on the bottom.

Fig. 4 compares the results by [14] and our scheme. Through the comparison on the same limit curve by subdivision, we can find that Fig. 4 (b) has a better smoothness and more circular results. Meanwhile, Fig. 4 (b) has less calculation complexity than Fig. 4 (a). The compared figures show that our interpolating subdivision curve scheme can generate a sufficiently smooth and circular limit curve.



(a) Results by the interpolating subdivision scheme in [14] (b) Results by our subdivision scheme

Fig. 4: Interpolating subdivision comparison

5.2 Surface Subdivision

In Fig. 5, it shows the examples of two kinds of surfaces, namely, triangle mesh on the tetrahedron and quadrilateral mesh on the cube. As the level increasing, the subdivision surfaces are more and more close to a smooth sphere, and hence the surfaces match all of the original vertices. From these results, it can see that we can change a cube or a regular tetrahedron to a sphere by using the interpolation subdivision scheme proposed in this paper. When subdividing the tetrahedron, our method just renders the 4 triangles in a fixed level, and then the algorithm will get the sub-triangles and go on rendering the sub-triangles recursively.

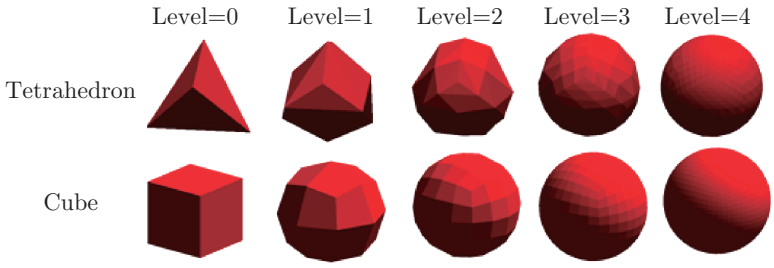
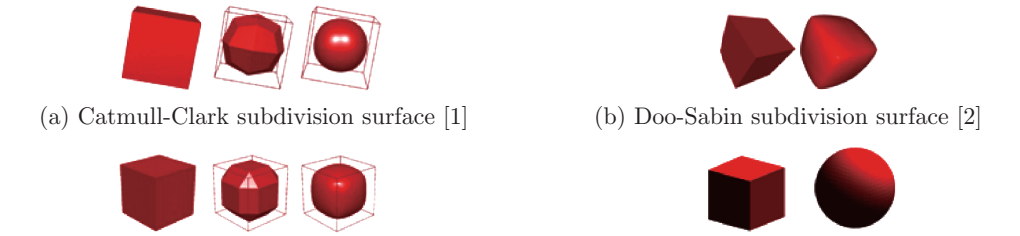


Fig. 5: Subdivision of a tetrahedron and a cube

Fig. 6 shows the results of some examples realized by our method and the algorithms reported by others works in the subdivision of a cube. Fig. 6 (a)-(b) are the results using approximating schemes, in which it can achieve smooth surfaces but the surfaces can not match the vertices on the cube. Fig. 6 (c) is the results of an interpolation subdivision scheme from Doo-Sabin subdivision interpolating schemes [15]. Though the surface is smooth and achieved by complex calculation, it is not able to generate a sphere. Fig. 6 (d) is the results by our subdivision scheme. As shown in the picture, the results are much better than that in Fig. 6 (b)-(c) apparently. Furthermore,

different from these schemes, our subdivision scheme need not deal with the topology of the complex meshes or update any data of the vertices, which enable us to complete the subdivision in a simple and fast way.



(a) Catmull-Clark subdivision surface [1]

(b) Doo-Sabin subdivision surface [2]

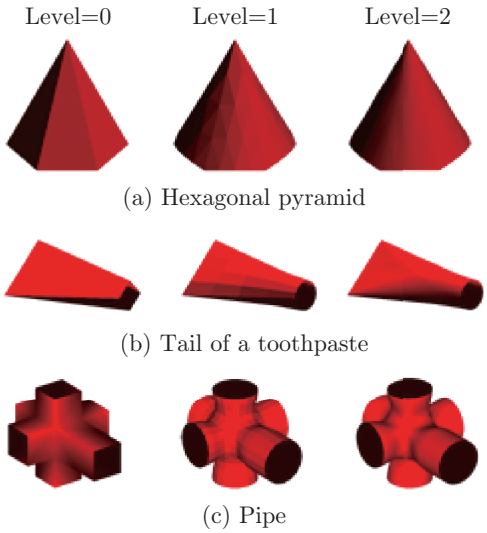
(c) An interpolation Doo-Sabin subdivision [15]

(d) The subdivision surface by our method

Fig. 6: Comparison between different subdivision surfaces

5.3 Subdivision of Complex Surfaces

Fig. 7 shows the objects with complex surfaces subdivided recursively by our method. There are 3 objects including (a) a regular hexagonal pyramid, which is the form of triangle meshes and have a closed angle on the top; (b) the tail of a toothpaste, which combines of triangle and quadrilateral meshes, and could generate sharp features since the tail has two free points; (c) a pipe usually in industrial applications, which is formed by quadrilateral meshes. From the table, it can see that the initial shape in level 0 is quite simple and rough, and when in level 2, the embryonic forms of the objects can be observed, after repeated subdivision and in level 4, we already get the objects with quite smooth surfaces.



(a) Hexagonal pyramid

(b) Tail of a toothpaste

(c) Pipe

Fig. 7: Subdivision for objects with complex surfaces

6 Conclusion

In this paper, we have proposed a new scheme of interpolation subdivision surface to achieve more realistic curve and surface design. Firstly, a curve subdivision scheme is proposed based on cubic

Bezier curve and normal vectors. This algorithm can be further used to generate a circle from a triangle, which has only 3 vertices and 3 edges. Then, we extend this method to the scheme of interpolation subdivision surface of triangle and quadrilateral. The advantage of this scheme is that it is easy for computation, since it avoids the computational cost of solving a large system of linear equations, and has an simple structure in the algorithm. Thus, this scheme can be realized in parallel implementation, which makes the computation much faster. Through the comparison between the results of ours and others work, it shows the scheme proposed in this paper makes an important improvement in smoothness and efficiency.

References

[1] D. Doo, M. Sabin, Analysis of the behavior of recursive division surfaces near extraordinary points, Computer-Aided Design, 10(6), 1978, 356-360

[2] E. Catmull, J. Clark, Recursively generated B-spline surfaces on arbitrary topological meshes, Computer-Aided Design, 10(6), 1978, 350-355

[3] C. Loop, Smooth Subdivision Surfaces Based on Triangles, Master Thesis, USA: Utah University, 1987

[4] L. Kobbelt, sqrt3-subdivision, Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH, 2000, 103-112

[5] N. Dyn, D. Levin, J. A. Gregory, A butterfly subdivision scheme for surface interpolation with tension control, ACM Transactions on Graphics, 9, 1990, 160-169

[6] L. Kobbelt, P. Schroder, A multiresolution framework for variational subdivision, ACM Transactions on Graphics, 17(4), 1998, 209-237

[7] X. Yang, Surface interpolation of meshes by geometric subdivision, Computer-Aided Design, 37(5), 2005, 497-508

[8] J. Zheng, Y. Cai, Making Doo-Sabin surface interpolation always work over irregular meshes, The Visual Computer, 21, 2005, 242-251

[9] J. Zheng, Y. Cai, Interpolation over arbitrary topology meshes using a two-phase subdivision scheme, IEEE Transactions on Visualization and Computer Graphics, 12, 2006, 301-310

[10] S. Lai, C. F. Cheng, Similarity based interpolation using Catmull-Clark subdivision surfaces, The Visual Computer, 22, 2006, 865-873

[11] C. Deng, X. Yang, A simple method for interpolating meshes of arbitrary topology by Catmull-Clark surfaces, The Visual Computer, 26, 2010, 137-146

[12] N. Dyn, D. Levin, J. A. Gregory, A 4-point interpolatory subdivision scheme for curve design, Computer Aided Geometric Design, 4, 1987, 257-268

[13] M. F. Hassan, I. P. Ivrissimitzis, N. A. Dodgson, M. A. Sabin, An interpolating 4-point C2 ternary stationary subdivision scheme, Computer Aided Geometric Design, 19(1), 2002, 1-18

[14] B. Jüttler, U. Schwanecke, Analysis and design of Hermite subdivision schemes, Visual Computer, 18(5), 2002, 326-342

[15] A. W. Zhang, C. M. Zhang, Computer-aided industrial design and conceptual design, CAID and CD 2009, IEEE 10th International Conference on Digital Object Identifier