Low Complexity Subdivision Algorithm to Approximate Phong Shading Using Forward Difference

Teng-Yao Sheu, Lan-Da Van, Tzung-Rung Jung, Cheng-Wei Lin, Ting-Wei Chang
Department of Computer Science, National Chiao Tung University, Hsinchu, Taiwan
E-mail: ldvan@cs.nctu.edu.tw

***Abstract*—In this paper, we proposed a low complexity subdivision algorithm to approximate Phong shading. It is a combination of a subdivision scheme using forward difference and a recovery scheme to prevent rasterization anomaly. Dual space subdivision, triangle filtering and variable sharing schemes are also proposed to reduce the computation. Compared with the conventional recursive subdivision shading algorithm, the proposed algorithm reduces 46.6% memory access, 80% projective transformations and perspective division, 93.75% clipping/culling tests and 60% 3x3 matrix multiplications for setup operation.**

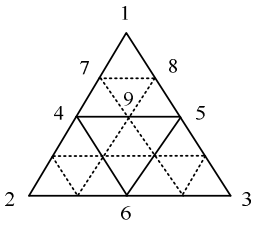
I. INTRODUCTION

Gouraud shading [2], per-vertex lighting, is commonly used in real-time application because it only applies reflection model [1] on vertices of polygons and linearly interpolates the intensities for internal pixels. Phong shading [3], per-pixel lighting, linearly interpolates face normal across a polygon for internal pixels and applies reflection model on each pixels. Phong shading can render smooth and realistic highlight but demand huge computation to light every pixel. Therefore, it's not suitable for real-time application without hardware acceleration.

In order to reduce complexity of Phong shading, many researchers concentrate on approximating scheme with reasonable tradeoff between cost and quality. The Taylor expansion [4] is used to approximate Phong reflection model. The average cost is high in the scenes with small polygons or multi-light sources. Spherical interpolation algorithms [6][7] aim to avoid renormalization. But, the setup cost is expensive for small polygons. The mix shading [11] combines two shading methods. When the hightlight covers the polygon, it is rendered using Phong shading. Otherwise, Gouraud shading is employed. The problem is that per-pixel lighting is involved in the highlight region. To completely eliminate pre-pixel lighting, quadratic interpolation [8][9] uses quadratic function to interpolate light intensities between six points in a triangle. The quadratic scheme may introduce notable gap on the edge if the triangle is too large. Subdivision scheme [5][12] is another way to approximate Phong shading. Subdivision scheme subdivides a triangle into smaller ones and renders these small triangles individually with Gouraud shading. Compared to per-pixel lighting, only vertices are lit. It is manifest that the computational complexity is greatly reduced.

The attractive feature of subdivision scheme is that it can control rendering quality dynamically. If high quality shading is demanded, more small triangles are generated. Otherwise, fewer triangles are generated to reduce the processing time and power consumption. Previous works use recursive subdivision algorithm to subdivide triangles in software. Subdivision in software has unnecessary bus traffic from CPU to graphic hardware because the generated vertices must be transferred to graphic hardware for rendering. To save bandwidth, the subdivision algorithm should be integrated into graphic hardware. Recursive subdivision algorithm is not efficient for hardware implementation because it uses stack to buffer the intermediate vertices. As illustrated in Fig. 1, the memory accesses consume large amount of power and the required memory size depends on the recursive depth. Another problem of previous algorithms is the overhead introduced by the generated triangles. They decrease performance greatly if they were not correctly handled. Little attention has been given to these points above.

In this paper, we proposed a low complexity subdivision algorithm to approximate Phong shading using forward difference technique. With least memory access and constant memory size requirement, it's suitable for hardware implementation. A recovery scheme is proposed to prevent rasterization anomaly. We also present three different schemes to reduce the overhead of the subdivision schemes.



Push vertices 1, 2, 3 (3 write)
Pop vertices 1, 2, 3 (3 read)
Gen vertices 4, 5, 6
Push vertices 1, 4, 5 (3 write)
Push vertices 4, 2, 6 (3 write)
Push vertices 4, 6, 5 (3 write)
Push vertices 5, 6, 3 (3 write)
Pop vertices 1, 4, 5 (3 read)
Gen vertices 7, 9, 8
Output triangle 1, 7, 8
Output triangle 7, 4, 9
Output triangle 7, 9, 8
Output triangle 8, 9, 5
……

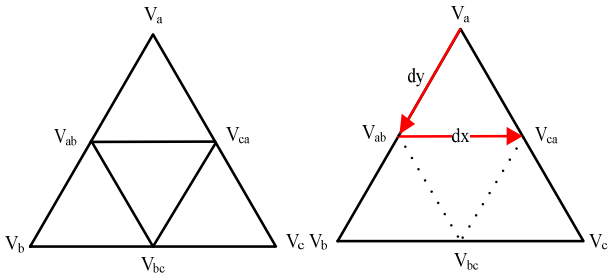
Fig. 1. Illustration of recursive subdivsion scheme.

This work was supported in part by the National Science Council (NSC) Grant NSC-96-2220-E-009-038, NSC-97-2220-E-009-055

This paper is organized as follows: the proposed algorithm is described in Section II. In Section III, we present three schemes to improve the subdivision algorithms. Complexity analysis is given in Section IV. At last, the brief statements conclude the presentation.

II. Proposed Subdivision Algorithm

In this section, we present proposed subdivision algorithm to approximate Phong shading. Our algorithm is the combination of a subdivision scheme using forward difference and a recovery scheme used to prevent rasterization anomaly. Forward difference is a technique used to refine mesh [10]. Herein, we use it to subdivide triangles. An example is illustrated in Fig. 2. First, we compute the differentials vectors, dx and dy in horizontal and vertical direction, respectively:



(a) Subdivided four triangle (b) Incremental subidivision

Fig. 2: Illustration of the proposed subdivision scheme.

$$dx = (V_c - V_b) / level$$
$$dy = (V_b - V_a) / level$$
$$\text{level} = \log_2(\text{number of generated triangles})$$

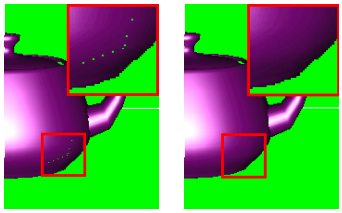
After obtaining the differentials, the middle vertices, V_{ab}, V_{bc} and V_{ca}, are computed by adding the difference vectors:

$$V_{ab} = V_a + d_y$$
$$V_{ca} = V_{ab} + d_x$$
$$V_b = V_{ab} + d_x$$
$$V_{bc} = V_b + d_x$$
$$V_c = V_{bc} + d_x$$

Finally, all vertices: V_a, V_b, V_c, V_{ab}, V_{bc}, and V_{ca} are obtained. They are packed into four triangles and output. This scheme is extended easily to subdivide arbitrary level.

The proposed subdivision scheme is more efficient than recursive ones because generating one vertex only access memory one time. Since only two difference vectors and two accumulation vertices for different direction to be buffered, the required memory size is constant. Therefore, the proposed scheme is very suitable for hardware implementation.

Subdivision schemes using forward difference may result in rasterization anomaly where pixels are lost in the rendered object. As shown in Fig. 3(a), the green pixels on the teapot are the lost pixels. The forward difference technique is numerical instable with small step width [10].

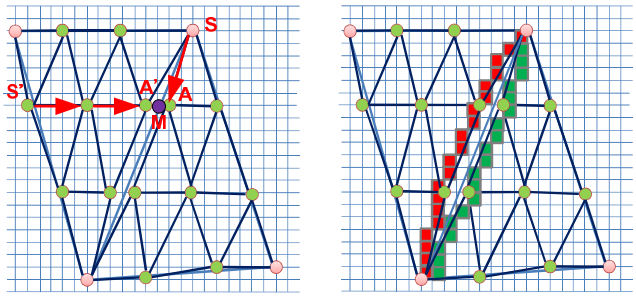


(a) Anomalies occurs (b) No Anomalies

Fig. 3: Example of teapot.

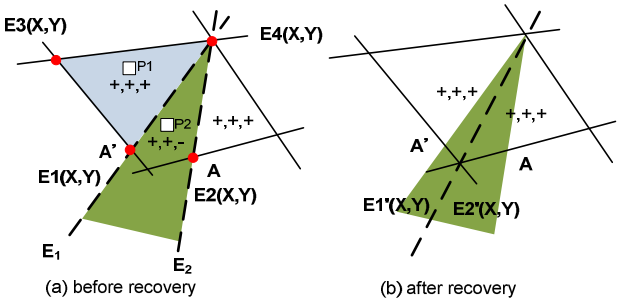
$$E_0(x,y) = a_0x + b_0y + c_0$$
$$= -(y_2 - y_1)(x - x_1) + (x_2 - x_1)(y - y_1)$$
$$= -(y_2 - y_1)*x + (x_2 - x_1)*y + (y_2*x_1 - y_1*x_2)$$
$$a_0 = -(y_2 - y_1)$$
$$b_0 = -(x_2 - x_1)$$
$$c_0 = y_2*x_1 - y_1*x_2$$

Fig. 4: Edge function definition.



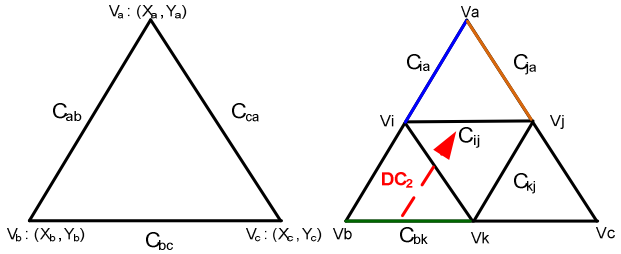
(a) After subdivision (b) Rasterization result

Fig. 5: Example of rasterization anomaly.



(a) before recovery (b) after recovery

Fig. 6: Illustration of edge function.



$$C_{ab} = Y_b*X_a - Y_a*X_b$$
$$C_{bc} = Y_c*X_b - Y_b*X_c$$
$$C_{ca} = Y_a*X_c - Y_c*X_a$$

$$C_{ia} = C_{ab}/2$$
$$C_{bk} = C_{bc}/2$$
$$C_{ja} = C_{ca}/2$$
$$DC_1 = ((X_c - X_b)*(Y_b - Y_a) + (Y_c - Y_b)*(X_a - X_b))/2$$
$$DC_2 = ((X_a - X_b)*(Y_c - Y_b) + (Y_a - Y_b)*(X_b - X_c))/2$$
$$DC_3 = ((X_b - X_a)*(Y_a - Y_c) + (Y_b - Y_a)*(X_c - X_a))/2$$
$$C_{kj} = -C_{ia} - DC_1$$
$$C_{ji} = -C_{bk} - DC_2$$
$$C_{ik} = -C_{ia} - DC_3$$

Fig. 7: Illustration of the recovery scheme.

Moreover, the vertices on the sharing edge are calculated with different starting vertex and differential vectors. Therefore, the subdivision leads to considerable numerical errors. As illustrated in Fig. 5(a), M is a vertex on the

sharing edge. Since the accumulated numerical errors, the A and A' have different coordinates. As a result, small triangles defined by A and A' are not adjacent to each other. Consequently, the rasterization anomaly occurs. Fig. 5(b) shows the result after rasterization.

To eliminate the anomalies, a recovery scheme is proposed. The proposed scheme takes the advantage of the edge function [14]. The edge function is a line equation thought two vertices as shown in Fig. 4. Pixels are regarded as being in the triangle if they are located in the region where the values of the edge functions of the triangle are all positive. For example, in Fig. 6(a), the P1 is in the triangle because it is located the blue region where the values of edge functions E1, E3 and E4 are all positive. Since A and A' have different coordinates, they define different edge functions E1 and E2 on edge E_1 and E_2 in Fig. 6(a), respectively.. The pixels, for example P2, in the green region have negative values of both E1 and E2 and are not regarded as pixels in any triangle. Thus, these pixels will not be rendered which result in rasterization anomaly. With the proposed recovery scheme, the problem can be eliminated.

In Fig. 7, C_{ab}, C_{bc}, C_{ca} are edge function constants of the original triangle. C_{ai}, C_{ij} and C_{ja} are edge function constants of the upper small triangles and can be derived from the following steps. First, by the linearity, C_{ai}, C_{ja}, and C_{bk} are calculated with dividing C_{ab}, C_{ca}, and C_{bc} by two, respectively. Secondly, the forward difference, DC_2, is derived using the formula listed in Fig. 7. Finaly, C_{ij} is derived by adding DC_2 to C_{bk}. Edge function constants of other small triangles can be derived in the similar way. After all constants are obtained, these small triangles with the derived constants in the previous steps are rendered.

Fig. 6(b) depicts an illustration of the recovery scheme. The dash line represents the edge functions E1' and E2' derived by our scheme. Edge function constants of E1' and E2' have the same magnitude and opposite sign. The pixels in the green region must have positive value of one of the two edge functions and are regarded as being in one of the two triangles. Therefore, they are rendered correctly. In Fig. 3(b), the teapot is rendered without anomalies using the recovery scheme. The proposed recovery scheme only involves one addition for evaluating one edge function constant.

III. PROPOSED SCHEMES TO IMPROVE SUBDIVSION ALGORITHM

In this section, we present three schemes to improve the proposed subdivision algorithm.

A. Dual space subdivision

Many graphic libraries support three kinds of light sources. Point and spot lights require coordinates of vertices in the eye space for light directions. Therefore, subdivision in the eye space is instinctive. After lighting, coordinates in the eye space are projected to the screen space for the rasterization. Since the projection involves the projection transform and perspective division which are expensive operations, we introduce a dual space subdivision scheme depicted in Fig 8.

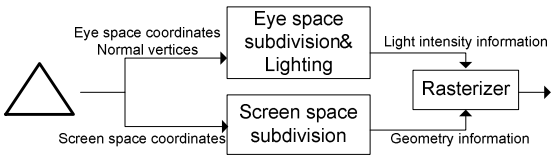


Fig. 8: Data flow of dual space subdivision.

First, the original triangle is transformed to the screen space but the eye space coordinate are maintained. Secondly, the triangle is subdivided in both spaces. Finally, for each small triangle, the eye space coordinates and normal vectors are used for lighting. The screen space coordinates are provided as the geometry information for rasterization. With this scheme, the proposed subdivision scheme is more efficient. The light intensity of the rendered image, of course, is not perspective correct. This problem is neglectable because human eye is not sensitive to the light intensity difference and perspective correctness is more important to texture coordinates, not light intensity.

B. Triangle filtering:Pre-Culling/ Clipping/Specular test

The conventional subdivision based approximating Phong shading algorithms process small triangle individually. They are not efficient because each small triangle is processed with individual complicated culling and clipping operations. Thus, a pre-culling scheme is proposed to reduce the computation. This scheme is based on the fact that the small triangles subdivided from the original triangle are lying on the same plane in the space. Therefore, they have the same face normal and the same culling test result. If the original triangle is culled, subdivision is unnecessary. Otherwise, all small triangles are rendered without culling test. The proposed pre-clipping scheme uses the same property. Once a big triangle is clipped, all small triangles from the original clipped triangle are guaranteed to be inside the viewing frustum. Therefore, no additional clipping operations were needed. The H test [11] is also adopted in the triangle filtering scheme to eliminate unnecessary subdivision operations. As illustrated in Fig. 9(a), the red region indicates the triangles do not pass the H test and will not be subdivided. Fig. 9(b) depicts the data flow of the proposed filtering mechanism. Moreover, we deliver the $H \cdot V$ terms to the Gouraud shader to eliminate the recomputation for these terms.

C. Vertex attributes sharing

The conventional subdivision based approximating Phong shading algorithms subdivide all vertex attributes including depth, fog factor, and texture coordinates. These attributes are interpolated linearly in rasterization stage using planar equation [13]. Computing the planar equation coefficients for one attribute involves a 3x3 matrix

multiplication. A sharing scheme is proposed to reduce the computation. Since the original triangle and the small triangles are lying on the same plane, they define the same plan equation. Therefore, the coefficients of the planar equation derived from the small triangles are the same.

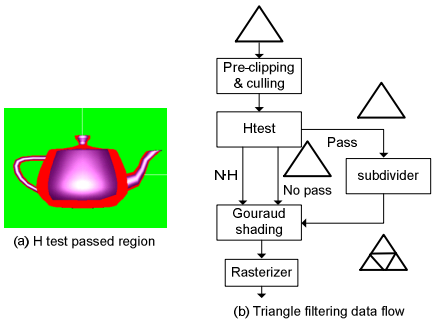


Fig. 9: Illustration of triangle filtering scheme.

Reusing these coefficients eliminates the subdividing and the setting up vertex attributes for the small triangles. However, these coefficients have to re-setup for small triangles for rasterization. The cost depends on the rastization algorithm. It takes about three multiplications to re-setup for each small triangle with tiled traversal scheme [14].

IV. COMPLEXITY ANALYSIS

In this section, we present a brief analysis and comparison for the proposed algorithm and the conventional subdivision algorithm. To simplify the comparison, we assume that one triangle is subdivided into 16 smaller ones. Each vertex has five attributes. In this case, 15 vertices where generated and they define 16 small triangles. The conventional recursive scheme requires five stack push and five stack pop operations. Each stack operation involves three writes or reads, in Fig. 1, the memory were accessed 30 times. In proposed scheme, generating one vertex only accesses memory once and the total number of memory accesses is 16.

The conventional scheme takes 15 projection transforms and perspective divisions to project the generated vertices to the screen space. With dual space subdivision scheme, it only needs three of them. The triangle filtering scheme also reduces the culling and clipping computation from 16 times to once. Setting up five attributes, in this case, requires 80 3x3 matrix multiplications. With variable sharing scheme, the expensive 3x3 matrix multiplication can be replaced with inexpensive 1x3 matrix multiplication for re-setting up each small triangle. Thus, the setup cost of the subdivided triangles is equivalent to $80*(1/3) + 5 \cong 32$ 3x3 matrix multiplications where 5 is the initial setup cost. The summary of the complexity analysis is listed in Table. 1.

Table. 1: Complexity Comparison

	Conventional subdivision algorithm	**Proposed subdivision algorithm**	**Complexity reduction in percent**
Number of memory accesses	30	16	46.6%
Number of the projective transforms and perspective divisions	15	3	80%
Number of the clipping/culling test operations	16	1	93.75%
Number of the 3x3 matrix multiplications for setup	80	32	60%

V. CONCLUSION

In this work, a subdivision algorithm for approximating Phong shading is presented. The proposed algorithm is a combination of a subdivision scheme using forward difference and a recovery scheme to prevent rasterization anomaly. Three schemes are proposed to lower the computation of subdivision-based approximating Phong shading. Compared with the conventional recursive subdivision, the proposed algorithm is more efficient and suitable for hardware implementation.

References

[1] A. Watt, "3D computer graphics," 3rd Edition, *Addison Wesley*, 2000.

[2] H. Gouraud, "Continuous shading of curved surfaces," *IEEE Trans. Comp.*, pp.623-628, June, 1971.

[3] A.T. Phong, "Illumination for computer generated pictures," *Communications of the ACM*, Vol. 18, No. 6, June, pp.311-317, 1975.

[4] G. Bishop, and D. M. Weimer., "Fast Phong Shading," *Proc. Computer Graphics and interactive Technique*, 1986, pp.103-106.

[5] J. Pöpsel, and Ch. Homung, "Highlight shading lighting and shading in a PHIGS+PEX environment," *EUROGRAPHICS*, 1989, pp.317-332.

[6] A. A. Mohamed, L. S. Kalos, and T. Horváth., "Hardware implementation of Phong shading using spherical interpolation," *Periodica Polytechnica* Vol. 44, Nos 3-4, 2000.

[7] T. Barrera, A. Hast, and E. Bengtsson., "Faster shading by equal angle interpolation of vectors," *IEEE Trans. Visualization and Computer Graphics*, pp.217-223, March, 2004.

[8] A. A. Mohamed, L. S. Kalos, G. Szijártó, T. Horváth, and T. Fóris., "Quadratic interpolation in hardware Phong shading and texture mapping," *SCCG'01*, April, 2001, pp.181-188.

[9] T. Barrera, A. Hast, and E. Bengtsson., "Fast near Phong-quality software shading," *WSCG'06*, January, 2006, pp.109-115.

[10] S. Bischoff, L.P. Kobbelt, and H.P. Seidel, "Toward hardware implementation of Loop subdivision". *Proc. SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware*, 2000, pp.41-50.

[11] K. Harrison, D. A. P. Mitchell, and A. H. Watt., "The H-test: a method of high speed interpolative shading," *Proc. New Trends in CG.*, 1988, pp.106-166.

[12] Y. Cho, U. Neumann, and J. Woo., "Improved specular highlights with adaptive shading," *Proc. of CG. International,* June, 1996, pp.38-46.

[13] M. Olano, and T. Greer., "Triangle scan conversion using 2D homogeneous coordinates," *Proc. SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, August, 1997, pp.89-95.

[14] J. McCormack and R. McNamara., "Tiled polygon traversal using half-plane edge functions," *Proc. SIGGRAPH/EUROGRAPHICS Workshop on Graphics Hardware,* 2000, pp.15-21