A Study on the Algorithms for 2D Texture Mapping and Its Employment in Hardware Design

Wang Donghui
Department of computer science, XUPT,
Xi'an 710121, China

Dong Liang
Department of computer science, XUPT,
Xi'an 710121, China

Du Huimin
Department of electronic engineering, XUPT,
Xi'an 710121, China

Abstract— **Based on a solid analysis of the working principle of graphic processors, this paper puts forward the dual-rail handshake protocol of the data pipeline which is to be proved to improve the architecture of graphics rendering pipeline and thus make the system much closer to that of a graphic processor in real time. Specific algorithms and hardware design methods are as well developed in this work with their focus dwelling on the computation-consuming texture mappings, including two critical techniques: texture coordinate calculation and texture filtering. FPGA result shows that circuits based on textual algorithms can define accurately the textual coordinates and yield precise textual images, bringing significant improvements to the performance of graphics rendering.**

Keywords-GPU, organization, pipeline, texture mapping

I. INTRODUCTION

With an in-depth study and several necessary improvements on typical working principles of graphics processor [1], we develop a graphics processing pipeline organization tree [2] as is shown in Figure.1. A data pipeline organization of this kind using the dual-rail handshake protocol [3] makes our simulation designing platform much closer to that of a real time graphics processor [4].

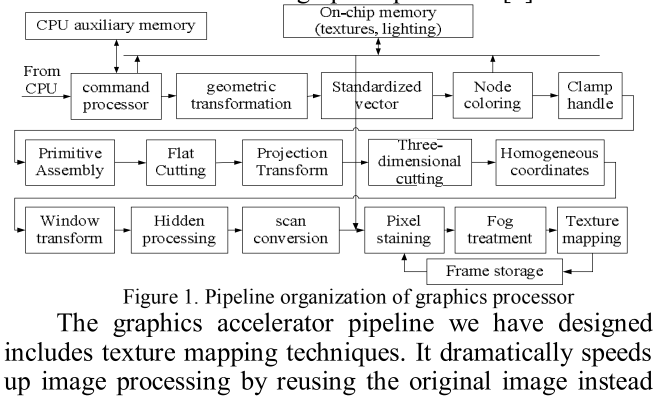

Figure 1. Pipeline organization of graphics processor

The graphics accelerator pipeline we have designed includes texture mapping techniques. It dramatically speeds up image processing by reusing the original image instead of reweaving a new one [5]. The texture mapping process is very suitable for such designs as hardware circuits[6], since it demands simple repetitive calculations for each screen pixel and the calculation of texture coordinates and filtering is so complex that only functionally dedicated hardware can help overcome the practical time limits[7][8]. Having conducted an overall study on the cases of orthographic and perspective projection, this paper is going to offer in terms of these two projections a solution of texture coordinate calculation and two filtering methods which are known as the nearest point sampling filtering and the bilinear filtering. We set the research emphasis on the design of 2-D texture mapping.

II. CALCULATION AND DESIGN OF THE TEXTURE COORDINATES

First of all, the relation between coordinate and mapping needs to be established. In the object coordinate system, when pixel vertex has experienced such coordinate transformations as geometric transformation, projective transformation and window transformation, the vertex would be in the window coordinate system. Therefore, while texture mapping is being implemented, it is also necessary to realize the mapping among the window coordinate (such as (Xw, Yw) in Figure 2, namely the window coordinate), the texture coordinate and the texture image coordinate [9]. The coordinate axes of texture coordinate are marked as s and t with a default range of [0.0, 1.0], as is shown in Figure 2. The coordinate axes of texture image are marked as u and v which correspond respectively with the width and height of the image in Figure 2, where the width and height are both set at 64 in this design.

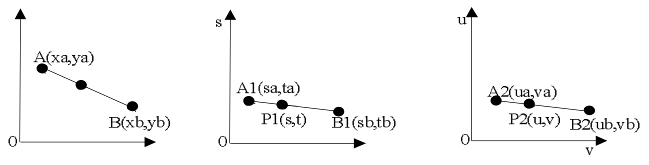

Figure 2. Line segments in texture mapping

After the pixels are assembled, the window coordinates and the texture coordinates of the points, the line and the vertexs of triangles are defined, and when achieving texture mapping in scan conversion and pixel colouration in level one, we need to figure out the relations between the texture coordinates and the window coordinates of all the pixels inside the beeline and the triangle by means of bilinear or hyperbolic insert value, so as to obtain the texture coordinates of the point.

1. To texture map a point is to locate it in the texture image so that its color value can be obtained. The mapping of this kind is known as a process of "posting a texture element" and it works as follows:

After obtaining the point's window coordinate (x, y, w) and its texture coordinate (s, t, q), we need to calculate according to the following equations:

$$u = s(width - 1), \quad v = t(height - 1) \qquad (1)$$

2. For the texture mapping of a line segment, the relation between window coordinates and texture coordinates is obtained with linear interpolation. The principle is shown in Figure 2.

Inferred from linear interpolation, |AP|/|AB|=|A1P1|/|A1B1|is applicable for any point P on the line in Figure.2. The similar triangles principle is used here and we replace the ratio of abscissas with that of lengths during the hardware implementation:

$$s = \frac{x - xa}{xb - xa} sb + \frac{xb - x}{xb - xa} sa \text{ , } t = \frac{x - xa}{xb - xa} tb + \frac{xb - x}{xb - xa} ta \qquad (2)$$

Then we transform the value of *s* and *t* obtained in formula (1-1) to *u* and *v* in the texture image. The texture image coordinates (u, v) is the pixel point mapped onto the texture image.

For the vertical line segment, we replace *y*, *ya* and *yb* with *x*, *xa* and *xb*. The method is the same.

3. The triangle texture mapping is achieved in two steps. The first step is to rearrange the place of the triangle window coordinates according to the value of *y*, where the node order in Figure.3 is obtained with P0 in the above, P1 in the lower left corner and P2 in the lower right corner. The second step is to calculate the pixel values of the texture coordinates of the sample points.

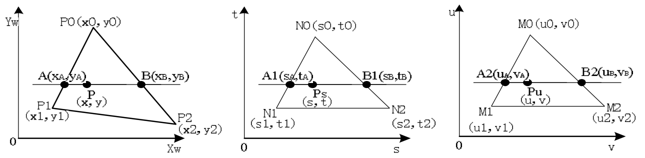

Figure 3. Texture mapping a triangle

As is shown in Figure 3., given that the texture coordinates of the three vertices P0, P1 and P2 of the triangle to be drawn are N0, N1 and N2, and that P is any point on segment AB (y = yA = yB), point A (xA, yA) and point B (xB, yB) being the intersection points of a scan line and the triangle, then the texture coordinates of points A and B, according to their positions on line P0P1 and P1P2, can be calculated with linear interpolation, and the texture coordinate Ps (s, t) of point P can as well be calculated with linear interpolation of the line AB. The specific expressions are as follows.

$$s_{A1} = \frac{|AP_1|}{|P_0P_1|} s_0 + \frac{|AP_0|}{|P_0P_1|} s_1$$

$$s_{B1} = \frac{|BP_2|}{|P_0P_2|} s_0 + \frac{|BP_0|}{|P_0P_2|} s_2 \text{, } s = \frac{x_B - x}{x_B - x_A} s_{A1} + \frac{x - x_A}{x_B - x_A} s_{B1} \qquad (3)$$

This algorithm for the projection transformation where the *w* value is always 1 is correct. The value of *w* may vary if perspective projection transformation is adoped here. In this case, it should be amended by the hyperbolic interpolation. The calculation expressions are as follows(1-4).

$$s_{A1} = \frac{lerp(\frac{s_0}{w_0}, \frac{s_1}{w_1}, f_A)}{lerp(\frac{1}{w_0}, \frac{1}{w_1}, f_A)}$$

$$s_{B1} = \frac{lerp(\frac{s_0}{w_0}, \frac{s_2}{w_2}, f_B)}{lerp(\frac{1}{w_0}, \frac{1}{w_2}, f_B)} \qquad (4)$$

Among them, $lerp(a,b,f) = af + b(1-f)$, and $f_A = |AP_1| / |P_0P_1|$, $f_B = |BP_2| / |P_0P_2|$ In the expression(1-4), w_0,w_1 and w_2 are respectively the *w* values of transformed vertices. Therefore, the texture coordinates of Ps can be obtained with:

$$s = \frac{x_B - x}{x_B - x_A} s_{A1} + \frac{x - x_A}{x_B - x_A} s_{B1} \quad ,$$

$$t = \frac{x_B - x}{x_B - x_A} t_{A1} + \frac{x - x_A}{x_B - x_A} t_{B1} \qquad (5)$$

We put the values of s and t into expression (1-1) and transform them into the relavent texture image u and v so that we can calculate the texture image coordinates of point PU in the uov texture image coordinate system which corresponds with any point P on line segment A1B1 in sot window coordinates .

In practical hardware designing, when we calculate the texture coordinates in accordance with the principle of similar triangles with the ratio of the distances between f_A and f_B in the expression above replaced by the fact that the ratio of longitudinal length is equal to that of the side length, the calculation of square and square root can be avoided and circuit performance can as well be greatly enhanced.

III. TEXTURE FILTER AND DESIGN

Nearest point sampling and bilinear filtering are included in this design[10].

A. Nearest Point Sampling:

Nearest point sampling is a process where the value of the nearest one of the 4 coordinate points around the texture sample point is taken as the texture sample pixel value.

As is shown in Figure 4., the coordinates of any point P(u, v) is on a minimum pixel region of the texture image. If neighboring pixels form a quadrilateral P0P1P2P3, the texture pixels the point corresponds with are as follows (The integral parts of calculated coordinates of *u* and *v* are taken and denoted as | u | and | v |.):

When u <| u | +0.5& v <| v | +0.5, use the texture pixels of coordinate points P0;

When u <| u | +0.5& v> | v | +0.5, use the texture pixels of coordinate points P1;

When u> | u | +0.5& v <| v | +0.5, use the texture pixels of coordinate points P3;

When u> | u | +0.5& v> | v | +0.5, use the texture pixels of coordinate points P2.

In case of the sampling point P happening to fall on the horizontal or vertical symmetry line of the quadrilateral, the principle of “lower left first” is adoped.

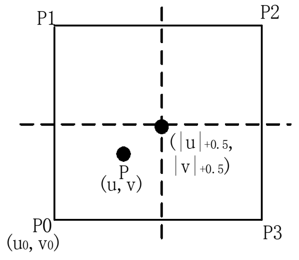

Figure 4. Nearest Point Sampling

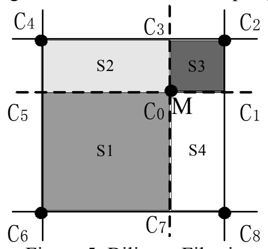

Figure 5. Bilinear Filtering

B. Bilinear Filtering

For bilinear filtering, the texture pixel value of the sample point is defined by the pixel values of the 4 textual pixel points around it. The weights of this 4 point are in direct proportion to the corresponding area they cover.

As is shown in Figure.5, the coordinate of arbitrary point M(u,v) falls on a minimum pixel area of the texture image. If neighboring pixel points form a quadrilateral $C_6C_4C_2C_8$, the texture pixel value of this point M is

$$c=\frac{S4}{S1+S2+S3+S4}\times c_4+\frac{S2}{S1+S2+S3+S4}\times c_8 \\ +\frac{S3}{S1+S2+S3+S4}\times c_6+\frac{S1}{S1+S2+S3+S4}\times c_2 \tag{6}$$

In the equation above, S1~S4 are the 4 areas in Figure.5, c4、c8、c6 and c2 are the color value of C4、C8、C6 and C2.

Therefore, the final computing formula is

$$c=(|u|+1-u)\times(v-|v|)\times c_4+(u-|u|)\times(|v|+1-v)\times c_8 \\ +(|u|+1-u)\times(|v|+1-v)\times c_6+(u-|u|)\times(v-|v|)\times c_2 \tag{7}$$

C. Designing the Nearest Sampling Point and the Bilinear Filtering

Figure.6 is the framework map of the nearest sampling point and bilinear filtering. Each time the latter two are used, they will read the texture color values from RAM (For bilinear filtering, the divider is activated. The divider in Figure 6. below is omitted.):

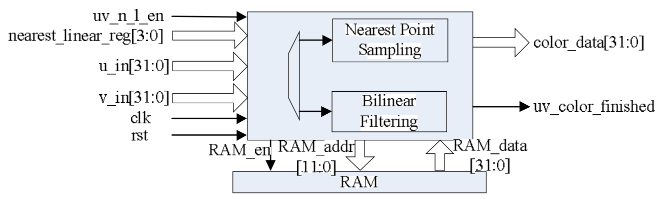

Figure 6. General framework of the nearest sampling point and bilinear filtering

IV. GENERAL DESIGN AND ITS VALIDITY

The general hardware circuit architecture of the texture mapping consists of 4 parts: the master module, nearest_linear_uv_rgba, the RAM module and the divider, as is shown in Figure 7.

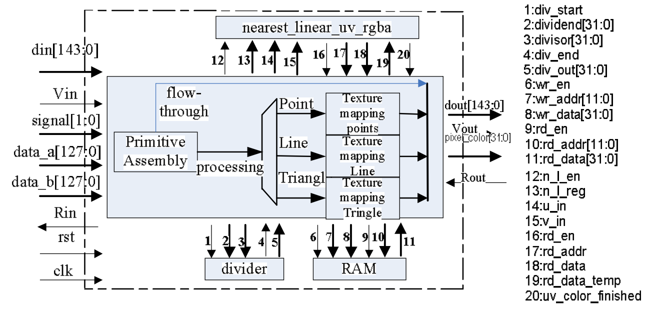

Figure 7. General Framework

Functions of each module:

(1) Master module is the most important module, which is mainly used for data stream processing and controls the computing of other modules.

(2) Filtering module, processes the nearest sampling point and bilinear filtering of the texture mapping. For the previous texture coordinates and according to the input control words, we can choose one of the filtering methods and figure out the texture pixel value of the sampling point.

(3) The main function of the RAM module is to store texture images.

(4) The divider operates 32-bit division (22-bit integers and 10-bit decimals) calculation.

In FPGA circuit authentication, an ALTERA DE2-70 development board is

used and a standard 64*64 checkerboard serves as the texture image. Following

is an analysis of how well the algorithm design is simulated.

The texture image obtained by hardware simulation is as follows in Figure 8:

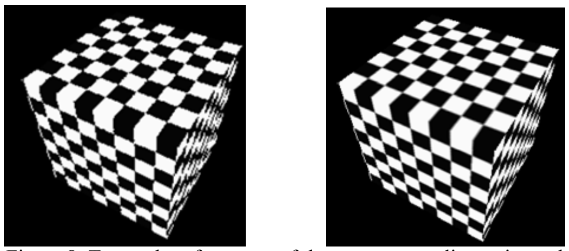
Figure 8. Textural performance of the nearest sampling point and bilinear filtering

Shown in Figure 8 is a hexahedron, each face of which is mapped with a checkerboard texture image and is weaved in the mode of perspective projection. The hexahedron on the left is drawn with the algorithm of the nearest sampling filtering and the right one with that of bilinear filtering. It is observable in Figure. 8 that the latter performs better and has achieved the antialiasing effect.

V. CONCLUSION

In this paper, we have firstly calculated textual coordinates in orthographic and perspective projection by making use of the algorithms for bilinear and hyperbolic interpolation, then realized textural mapping by means of the nearest point sampling and bilinear filtering, and finally offered a hardware circuit design of textural mapping. The authentication result shows that the design of the graphics processor system offered in this paper is correct and the algorithm for bilinear filtering can lead to more accurate images.

VI. REFERENCES

[1] Smelyanskiy, M.; Holmes, D.; Chhugani, J., et al. Mapping High-Fidelity Volume Rendering for Medical Imaging to CPU, GPU and Many-Core Architectures. Visualization and Computer Graphics [J], 2009, PP: 1563-1570.

[2] Fresse, V.; Houzet, D.; Gravier, C. GPU architecture evaluation for multispectral and hyperspectral image analysis. Design and Architectures for Signal and Image Processing [J], 2010, PP: 121-127.

[3] Civit, J.; Escoda, O.D. Robust foreground segmentation for GPU architecture in an immersive 3D videoconferencing system . Multimedia Signal Processing [J], 2010, PP75-80.

[4] Poli, G.; Saito, J.H.; Mari, J.F.; Zorzan, M.R. Processing Neocognitron of Face Recognition on High Performance Environment Based on GPU with CUDA Architecture.Computer Architecture and High Performance Computing [J], 2008, PP81-88.

[5] Dudgeon, J.E.; Srinivasan, An Algorithm for Graphics Texture Mapping. System Theory [J], 1991, PP 613-617.

[6] Demirer, M.; Grimsdale, R.L.; Cavusoglu, A.. Real time performance for texture mapping using the Pim Map technique in hardware. Electrotechnical Conference [J], 1996, PP 149 - 152

[7] Crow. The Aliasing Problem in Computer Generated Shaded Images. CACM [J], 1977, PP: 799- 805.

[8] Gouraud. Continuous Shading of Curved Surfaces. IEEE Trans. Computers [J], 1971.PP:623-629.

[9] Donghyun Kim; Lee-SupKim. Area-efficient pixel rasterization and texture coordition interpolation. Computer&Graphics [J], 2008, PP: 669-681.

[10] Heckbert, P.S. Survey of Texture Mapping. Computer Graphics and Applications, IEEE [J], 1986, PP: 56-67.