(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
30 September 2010 (30.09.2010)

PCT

(10) International Publication Number
WO 2010/111258 A1

(51) International Patent Classification:
G06T 15/00 (2006.01)

(21) International Application Number:
PCT/US2010/028301

(22) International Filing Date:
23 March 2010 (23.03.2010)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
12/409,592 24 March 2009 (24.03.2009) US

(71) Applicant *(for all designated States except US)*: **ADVANCED MICRO DEVICES, INC.** [US/US]; One AMD Place, Sunnyvale, California 94085 (US).

(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: **IOURCHA, Konstantine** [RU/US]; 7186 Wooded Lake Dr., San Jose, California 95120 (US). **DOGGETT, Michael** [AU/AU]; 33 Murrills Cres, Baulkham Hills, New South Wales 2153 (AU).

(74) Agent: **HALT, Gerald, B., Jr.**; Volpe and Koenig, P.C., 30 South 17th Street United Plaza, Suite 1600, Philadelphia, Pennsylvania 19103 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PE, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report (Art. 21(3))*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments (Rule 48.2(h))*

(54) **Title**: METHOD AND APPARATUS FOR ANGULAR INVARIANT TEXTURE LEVEL OF DETAIL GENERATION

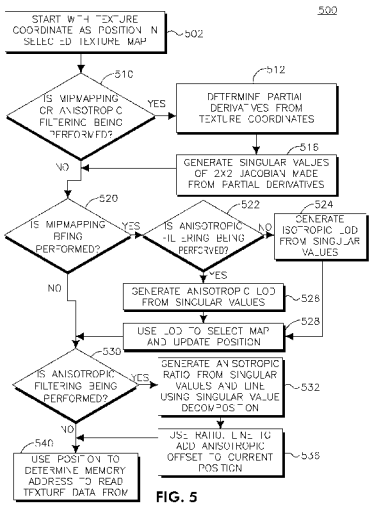


FIG. 5

(57) **Abstract**: A method and apparatus for angular invariant texture level of detail calculation is disclosed. The method includes a determination for a LOD that determines angular invariant LODs that result in efficient ASIC hardware implementation.

WO 2010/111258 A1

[0001] METHOD AND APPARATUS FOR ANGULAR INVARIANT TEXTURE LEVEL OF DETAIL GENERATION

[0002] CROSS REFERENCE TO RELATED APPLICATIONS

[0003] This application claims the benefit of U.S. Patent Application No. 12/409,592 filed March 24, 2009, which is incorporated by reference as if fully set forth herein.

[0004] FIELD OF INVENTION

[0005] The present invention is directed to texture mapping.

[0006] BACKGROUND

[0007] Synthetic image generation is the process of generating two-dimensional (2D) images from digitally created three-dimensional (3D) scenes. The image generation process includes 3D models of objects which are rendered into a 2D image.

[0008] To improve the appearance of the 3D models, textures, also known as color maps, are applied to the 3D objects. These textures can have one, two three or more dimensions and are mapped onto the surface of 3D objects using a mapping function. This mapping of textures onto an object is called texture mapping.

[0009] The 3D space is mapped to the 2D image using a perspective transformation. This transformation results in close objects being shown as bigger and objects which are farther away being shown as smaller. For closer objects, a higher level of detail is observed on the surface of the object. As the object moves further away from the view point (viewer's perspective), the object gets smaller and the level of detail (LOD) on the object is accordingly reduced. The LOD of the 3D object can be computed via a variety of methods.

[0010] As the surface onto which a texture is mapped is reduced in size in the 2D image the original texture map is reduced. This process is called minification. Minification requires that all of the individual colors within the

texture (known as texels) are sampled and filtered together to compute the final color value for applying to the 3D object. The area of texels that is sampled from the texture map is based on the projection of the image pixels onto the texture map. For large texture map minification, this filtering would require processing a large number of texels, thus increasing the computation time. This computation time can be significantly reduced by precomputing the filtered results using a number of different methods, including but not limited to mipmapping, summed area tables, and elliptical weighted average.

[0011] Mipmapping takes the texture map at its highest level of detail and generates several lower levels of detail texture maps in a series. For example, if the highest level of detail texture map is a color image of 512x512 resolution, mipmapping may generate a series of textures at lower resolutions such as 256x256, 128x128, 64x64, 32x32, 16x16, 8x8, 4x4, 2x2, and 1x1. Each texel in the new image can be calculated by segmenting the higher resolution image into 2x2 pixel segments and computing the average texture values for each of these segments. Other filters to produce the lower resolution images may also be used. For example, a summed area table computes the sum of all texels from the origin of the texture to the current location in the texture. The newly created lower resolution textures represent the original texture in a new image with a different (i.e., reduced) level of detail (LOD). When the 3D object is rendered, a final color value is generated based upon a selection, or selections, from these LOD images.

[0012] For the 2D case, the LOD is computed from the size of the projection of the 2D image pixel into the 2D texture space. For the equations below, if pixel coordinates are referred to as x, y then the texture space coordinates are referred to as u, v.

[0013] The projection can be represented by the partial derivatives which are estimated from the change in the texture coordinates across a quad. These partial derivatives are also known as the texture coordinate derivative (or gradient). 2D texture mapping has four components du/dx, du/dy, dv/dx, dv/dy. One conventional method for computing LOD is as follows:

$$lod = \log_2\left(\max\left\{\sqrt{\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2}, \sqrt{\left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2}\right\}\right) \qquad \text{Equation (1)}$$

[0014] As the texture map and 3D object rotate and the angle changes with respect to the screen coordinates, the above algorithm computes a changing LOD value, though the value should stay constant under rotational change.

[0015] The LOD value is used to select one of the LOD images. For objects that are at oblique viewing angles and the projection of the texture is at a large angle to the camera, the above LOD computation will result in selecting over blurred LOD images. Anisotropic filtering is a method of improving the texture filtering by selecting a less blurry LOD image and filtering that level according to the pixel projection into the texture space.

[0016] Modern graphics processing units (GPUs) are designed to render 3D scenes (images). A high level diagram of a conventional GPU 100 is shown in Figure 1. A GPU 100 can include multiple components such as a shader controller and input unit 110, a programmable shader unit 120, a render backend 130 and a memory 140. The programmable shader 120 runs programs that perform rendering algorithms to transform the 3D objects into a 2D image. The programmable shader 120 comprises hundreds of arithmetic logic units (ALUs). Older GPUs used custom hardware logic for the transformation and color computation stages instead of the programmable shader 120.

[0017] A shader controller and input unit 110 reads the input for a variety of computer graphics programs that are performed on the programmable shader 120 in a sequence of stages that comprise the graphics pipeline. The shader controller and input unit 110 can contain several fixed function, custom designed hardware stages of the graphics pipeline including, but not limited to: tessellation, triangle setup, and rasterization. These graphics pipeline stages can also be implemented in software on some graphics devices. The shader controller and input unit 110 also controls the execution of the programs on the programmable shader unit 120 by switching between a large number of programs that are waiting to run.

[0018] When texture data is accessed via a hardware texture unit or a program that issues a texture instruction running on the programmable shader 120, texture data is read from memory 140 into the texture unit 105 and texels are selected from the texture unit 105 and returned to the programmable shader 120.

[0019] Once a final color value is calculated in the programmable shader 120, the result is sent to the render backend unit 130 which tests the visibility of the color and blends the color with existing colors (color and Z operations) in the frame buffer in memory 140 and writes the final result into the memory 140. The color result from the shader can also use Z and color operation units that are implemented in software on programmable devices.

[0020] The texture unit 105 computes a LOD value as well as the memory address of the texture before reading it from the memory 140. The programmable shader 120 sends the current pixel position (x,y) and the current texture coordinates for that pixel (u,v) to the texture unit 105. The programmable shader 120 sends a 2x2 group of neighboring pixels to the texture unit 105. Using the 2x2 group, the texture unit 105 can estimate a rate of change of the texture coordinates (called derivatives) with respect to the pixel coordinates using partial differences as follows :

$$u_x = u_{Pixel1} - u_{Pixel0} \; ; \qquad \text{Equation (2)}$$

$$v_x = v_{Pixel1-} v_{Pixel0} \; ; \qquad \text{Equation (3)}$$

$$u_y = u_{Pixel2} - u_{Pixel0} ; \text{ and} \qquad \text{Equation (4)}$$

$$v_y = v_{Pixel2} - v_{Pixel0} . \qquad \text{Equation (5)}$$

These estimates can be arranged as a Jacobian matrix as follows:

$$J = \begin{pmatrix} u_x & u_y \\ v_x & v_y \end{pmatrix}. \qquad \text{Equation (6)}$$

[0021] Using the following equations:

$$A = v_x^{\,2} + v_y^{\,2} ; \qquad \text{Equation (7)}$$

$$B = -2(u_x v_x + u_y v_y) ; \qquad \text{Equation (8)}$$

$$C = u_x^2 + u_y^2 \text{ ; and} \qquad \text{Equation (9)}$$

$$F = (u_x v_y - u_y v_x)^2 \text{ ;} \qquad \text{Equation (10)}$$

new values au_x, au_y, av_x, av_y can be calculated as follows:

$$au_x = \sqrt{\frac{F(t+r)}{t(s+t)}} \text{ ;} \qquad \text{Equation (11)}$$

$$av_x = sign(Br)\sqrt{\frac{F(t-r)}{t(s+t)}} \text{ ;} \qquad \text{Equation (12)}$$

$$au_y = -sign(Br)\sqrt{\frac{F(t-r)}{t(s-t)}} \text{ ; and} \qquad \text{Equation (13)}$$

$$av_y = \sqrt{\frac{F(t+r)}{t(s-t)}} \text{ ;} \qquad \text{Equation (14)}$$

where:

$$r = A - C,\ s = A + C \text{ and } t = sign(r)\sqrt{r^2 + B^2}.$$

[0022] The new values au_x, au_y, av_x, av_y can be used instead of the texture coordinate partial derivative estimates and substituted into Equation (1) to compute an LOD value. This method improves the LOD determination by ensuring that it is angularly invariant. The method still requires the LOD calculation of Equation (1) and adds a complex solution requiring a large and slow application specific integrated circuit (ASIC) implementation. A new method for determining the LOD value used to select from among a series of texture maps, which is computationally efficient and reduces the size of the corresponding ASIC implementation, is highly desirable.

[0023] SUMMARY

[0024] A method and apparatus for efficient angular invariant texture level of detail determination is disclosed. The method includes a determination for a LOD that determines angular invariant LODs that result in efficient ASIC hardware implementation.

[0025] BRIEF DESCRIPTION OF THE DRAWINGS

[0026] Figure 1 shows a conventional GPU;

[0027] Figure 2 is a block diagram of a texture unit using an example implementation for angular invariant LOD calculation; and

[0028] Figure 3 is shows the components a singular value generator 210;

[0029] Figure 4 shows the components of anisotropic line generator 230;

[0030] Figure 5 is flow diagram of a method for LOD computation using singular value decomposition;

[0031] Figure 6 is a detailed flow diagram for generate singular values 516; and

[0032] Figure 7 is a detailed flow diagram for generating an anisotropic line using singular value decomposition 532.

[0033] DETAILED DESCRIPTION

[0034] When referred to hereafter, the term "map" refers to a "texture map." A method and apparatus for efficient angular invariant texture level of detail generation is disclosed. The method includes a new way to determine a LOD that generates angular invariant LODs that result in efficient ASIC hardware implementation. Texture unit changes described herein may be applied to either older or newer GPUs.

[0035] Figure 2 is a block diagram of one example of a texture unit 105 that performs angular invariant LOD generation. The texture unit 105 includes a partial derivative generator 202, a singular value generator 210, an anisotropic ratio (AR) generator 220, a LOD generator 222, an anisotropic line (AL) generator 230, a texel fetcher 240, a texel filter 250 and a memory 260. In

general, as shown in Figure 2, coordinates x, y, u and v are input at 201, and the partial derivative generator 202 calculates the partial derivative estimates. The singular value generator 210 calculates singular values using the calculated partial derivatives. The LOD is determined by the LOD generator at 222 using a singular value. The AR is determined by the AR generator at 220 and the AL is calculated by the AL generator at 230, using the singular values. The LOD value, AR and the AL are used if they (any one or any combination) exist to determine an updated position for the texels, otherwise the current texture map position is used, the appropriate texels (based upon the position) are fetched by a texel fetcher at 240 from memory 260. The texture unit 105 may also share memory with the GPU 100. The resulting texels are passed through a texel filter at 250 and the resulting color texels are output at 262.

[0036] In one embodiment, 3D graphics are rendered using mipmapped anisotropic texture filtering. In this embodiment, referred to as LOD generation using singular value decomposition (SVD), the texturing will not vary regardless of the level of detail displayed as the object is rotated in a three dimensional scene.

[0037] To generate the correct LOD, the singular values are generated. Using the texture coordinate derivatives in the Jacobian Form:

$$\mathbf{J} = \begin{pmatrix} u_x & u_y \\ v_x & v_y \end{pmatrix}. \qquad \text{Equation (15)}$$

The matrix J can be factorized using SVD:

M and m represent the singular values of the matrix J, where:

. $$\mathbf{J} = \mathbf{USV}; \qquad \text{Equation (16)}$$

where:

$$\mathbf{S} = \begin{pmatrix} z_1 & 0 \\ 0 & z_2 \end{pmatrix}; \qquad \text{Equation (17)}$$

and the matrix **S** is comprised of the singular values, z_1 and z_2.

The singular values are computed as follows:

First compute the intermediate values a, b, c, d, k, and l.

$$a = u_x + v_y; \quad \text{Equation (18)}$$

$$b = u_y - v_x; \quad \text{Equation (19)}$$

$$c = u_x - v_y; \quad \text{Equation (20)}$$

$$d = u_y + v_x; \quad \text{Equation (21)}$$

$$k = \sqrt{a^2+b^2}; \text{ and} \quad \text{Equation (22)}$$

$$l = \sqrt{c^2+d^2}. \quad \text{Equation (23)}$$

The singular value z_1 is the Isotropic LOD value which is computed as:

$$z_1 = \frac{k+l}{2}; \quad \text{Equation (24)}$$

and the singular value z_2 is the Anisotropic LOD which is computed as:

$$z_2 = \frac{|k-l|}{2}. \quad \text{Equation (25)}$$

[0038] For Anisotropic filtering, the ratio of anisotropy and the line or direction of anisotropy is calculated.

The Anisotropic ratio is calculated as:

$$\frac{z_1}{z_2}. \quad \text{Equation (26)}$$

The line of anisotropy can be found by computing the singular vector from the matrix **U**. The components of the singular vector are $\{u_1, u_2\}$.and can be calculated as follows:

$$g = \frac{1}{k}; \quad \text{Equation (27)}$$

$$h = \frac{1}{l}; \quad \text{Equation (28)}$$

$$m = ag + bh; \quad \text{Equation (29)}$$

$$n = dg + ch; \quad \text{Equation (30)}$$

$$o = ag - bh; \quad \text{Equation (31)}$$

$$p = dg - ch; \quad \text{Equation (32)}$$

$$u_1 = \frac{\sqrt{m^2 + n^2}}{2}; \text{ and} \quad \text{Equation (33)}$$

$$u_2' = \frac{\sqrt{o^2 + p^2}}{2}. \quad \text{Equation (34)}$$

[0039] The direction for $\{u_1, u_2\}$. is computed:

```
if ( abs(m) + abs(p) > abs(n) + abs(o)) {
    if (((m > 0) && (p > 0)) | |((m < 0) && (p < 0)))
    u2 = - u2'; }
else {  if (((n > 0) && (o < 0)) | |((n < 0) && (o > 0)))
    u2 = - u2'; }
```

[0040] The calculation may also be used for 1D LOD generation in which case the v_x, v_y, derivatives would be 0. Referring to Figure 3, a more detailed description of singular value generator 210 is provided. The singular value generator 210 comprises a derivative combination generator 302, a vector creator 304, Euclidian norm generators 306 and 308, a small singular value generator 310 and a large singular value generator 312.

[0041] The singular value generator 210 performs a linear combination of the texture partial derivatives 302, these combinations are formed into vectors 304, the Euclidean norm of the vectors is generated in 306 and 308, and the norms are then used in linear combination to generate a small singular value 310 and a large singular value 312. A more detailed description of singular value generation will be described with reference to Figure 6 below.

[0042] Referring to Figure 4, a more detailed description of AL generator 230 is provided. AL generator 230 comprises singular vector norm reciprocal

generator 402, direction intermediate vector creator 404, Euclidian norm generators 406 and 408, and u2 direction generator 416.

[0043] The anisotropic line generator 230 performs a reciprocal of the norms generated by the singular value generator 210. The anisotropic line generator 230 comprises a singular vector norm reciprocal generator 402 that generates singular vestor norm reciprocals, a direction intermediate vector generator that generates vectors using linear combinations of the reciprocals and previous values 404, a Euclidean norm generator that generates Euclidean norm vectors 406 and 408, and a direction generator that generates the direction of the anisotropic line 416.

[0044] A flow diagram of a method 500 for LOD computation using SVD is shown in Figure 5. The method 500 is initiated (502) with accepting at least one texture coordinate (e.g., output from a rasterization process not shown), which is a position (the current texture map position) in a selected texture map (e.g., a user may configure/determine which texture map is to be utilized). It is then determined if mipmapping or anisotropic filtering has been configured to be performed (e.g., by a user) (510).

[0045] If so, texture partial derivatives are computed from texture coordinates (512) and singular values are generated from a 2x2 Jacobian made (constructed) from the texture partial derivatives (516).

[0046] If not, or if the texture partial derivatives and singular values have already been computed at 512 and 516 respectively, it is determined whether mipmapping is configured to be performed (e.g., by a user) (520).

[0047] If mipmapping has been configured to be performed, it is determined if anisotropic filtering has been configured to be performed (522). If anisotropic filtering is not configured, then an isotropic LOD is computed (524) based upon a singular value (e.g., pick one singular value out of two). If anisotropic filtering has been configured then an anisotropic LOD is computed (526) based upon a singular value (e.g., pick one singular value out of two). A texture map is selected (it could be the same or a new texture map) and the current texture map position is updated (528) based upon the computed LOD (resulting from either

condition). The updated current texture map position might be identical to the previous position.

[0048] If mipmapping has not been configured (a negative result at 510 and 520) or the LOD (isotropic or anisotropic) has been computed (at 524 or 526), and it is determined that anisotropic filtering is configured (530), then an AR and an AL are computed based upon the singular value decomposition (532). The AR and AL are used to generate an anisotropic offset (536) (or multiple offsets) which is added to the current texture map position (updating the current texture map position). The updated current texture map position might be identical to the previous position.

[0049] In all cases, the current texture map position (updated or not) is used to calculate a memory location from which to read (retrieved, selected) the texture mapping data (540).

[0050] Referring to Figure 6, a more detailed description of singular value generation 516 is provided. The generation of singular values from the Jacobian matrix is performed by first generating linear combinations of the matrix elements with predefined constant coefficients 602, creating two two-dimensional vectors with the components coming from the previously generated linear combinations 604, generating the Euclidian norms (or an approximation of the Euclidian) of the vectors 606, and generating singular values as a linear combination of the generated norms of the vectors 608.

[0051] Referring to Figure 7, a more detailed description of AL generation using SVD 532 is provided. The generation of the AL from the SVD is performed by taking the reciprocal of the Euclidian norms generated during the singular value generation 702, creating linear combinations of the reciprocals with the linear combinations created during the singular value generation 704, and generating two two-dimensional vectors with the components coming from the previously generated linear combinations and then generating the Euclidean norm of these vectors 706. Also the direction of the anisotropic line is generated by comparing the magnitude and sign of the linear combinations generated previously 706.

[0052] As shown by Figure 5, various combinations of mipmapping, no mipmapping, filtering or no filtering are possible. For example, in one embodiment with no mipmapping or filtering configured, the unmodified current texture map position will be used to calculate a memory location from which to read (retrieve) the texture data (540).

[0053] In another embodiment, any other filtering algorithm, which uses the projection of a pixel into texture space where the projection can be represented by a singular value or singular values, may be substituted in place of mipmapping (e.g., summed area tables and elliptical weighted average).

[0054] Although the features and elements are described in the example embodiments in particular combinations, each feature or element can be used alone without the other features and elements of the example embodiments or in various combinations with or without other features and elements. The present invention may be implemented in a computer program or firmware tangibly embodied in a computer-readable storage medium having machine readable instructions for execution by a machine, a processor, and/or any general purpose computer for use with or by any non-volatile memory device. Suitable processors include, by way of example, both general and special purpose processors.

[0055] Typically, a processor will receive instructions and data from a read only memory (ROM), a RAM, and/or a storage device having stored software or firmware. Storage devices suitable for embodying computer program instructions and data include all forms of non-volatile memory, including by way of example semiconductor memory devices, read only memories (ROMs), magnetic media such as internal hard disks and removable disks, magneto-optical media, and optical media such as CD-ROM disks and digital versatile disks (DVDs). Types of hardware components, processors, or machines which may be used by or in conjunction with the present invention include Application Specific Integrated Circuits (ASICs), Field Programmable Gate Arrays (FPGAs), microprocessors, or any integrated circuit. Such processors may be manufactured by configuring a manufacturing process using the results of processed hardware description language (HDL) instructions (such instructions capable of being stored on a

computer readable media). The results of such processing may be maskworks that are then used in a semiconductor manufacturing process to manufacture a processor which implements aspects of the present invention.

CLAIMS

What is claimed is:

1. A method for texture mapping comprising:

inputting at least one texture coordinate as a current texture map position and if filtering based on a projection of a pixel into a texture space is configured:

determining texture coordinate partial derivatives; and

generating singular values of a Jacobian matrix constructed from the texture coordinate partial derivatives comprising:

generating linear combinations of the matrix elements with predefined constant coefficients;

forming two two-dimensional vectors from the linear combinations;

generating Euclidian norms or approximation of Euclidian norms, from the vectors; and

generating singular values as linear combinations of the norms.

2. The method of claim 1, wherein filtering based on a projection of a pixel into a texture space is mipmapping or anisotropic filtering.

3. The method of claim 2 further comprising:

if mipmapping is configured:

if anisotropic filtering is not configured:

determining an isotropic level of detail (LOD) from a selected singular value from the generated singular values;

selecting a texture map based upon the LOD;

updating the current texture map position; and

selecting texture data based upon the current texture map position.

4. The method of claim 2 further comprising:

if mipmapping is configured:

if anisotropic filtering is configured:

generating an anisotropic level of detail (LOD) from a selected singular value from the generated singular values;

selecting a texture map based upon the LOD; and

updating the current texture map position.

5. The method of claim 4 further comprising:

generating anisotropic ratio and anisotropic line from singular value decomposition comprising:

determining the reciprocal of the Euclidian norms;

creating linear combinations of the reciprocals of the linear combinations and generating two two-dimensional vectors from the linear combinations; and

generating the Euclidean norm of the vectors;

adding anisotropic offset to update the current texture map position; and

selecting texture mapping data based upon the current texture map position.

6. The method of claim 2 further comprising:

if mipmapping is not configured:

generating an anisotropic ratio and an anisotropic line from singular value decomposition comprising:

determining the reciprocal of the Euclidian norms;

creating linear combinations of the reciprocals of the linear combinations and generating two two-dimensional vectors from the linear combinations; and

generating the Euclidean norm of the vectors;

adding at least one anisotropic offset to update the current texture map position; and

selecting at least one texture mapping data based upon the current texture map position.

7. The method of claim 2 further comprising selecting texture mapping data based upon the current texture map position.

8. A method for texture mapping comprising:

inputting at least one texture coordinate as a current texture map position;

if mipmapping or anisotropic filtering is configured:

determining texture coordinate partial derivatives;

generating singular values of a Jacobian matrix constructed from the texture coordinate partial derivatives comprising:

generating linear combinations of the matrix elements with predefined constant coefficients;

forming two two-dimensional vectors from the linear combinations;

generating Euclidian norms or approximation of Euclidian norms, from the vectors; and

generating singular values as linear combinations of the norms;

if mipmapping is configured:

if anisotropic filtering is not configured:

generating an isotropic level of detail (LOD) from a selected singular value from the generated singular values;

selecting a texture map based upon the LOD;

updating the current texture map position;

if anisotropic filtering is configured:

generating an anisotropic level of detail (LOD) from a selected singular value from the generated singular values;

selecting a texture map based upon the LOD;

updating the current texture map position;

if anisotropic filtering is configured:

generating an anisotropic ratio and an anisotropic line from singular value decomposition comprising:

determining the reciprocal of the Euclidian norms;

creating linear combinations of the reciprocals of the linear combinations and generating two two-dimensional vectors from the linear combinations; and

generating the Euclidean norm of the vectors;

adding at least one anisotropic offset to update the current texture map position; and

selecting at least one texture mapping data based upon the current texture map position.

9. The method of claim 1 wherein approximation of Euclidian norms is generated by using a polygonal Minkowski norm.

10. The method of claim 8 wherein approximation of Euclidian norms is generated by using a polygonal Minkowski norm.

11. A texture unit comprising:

a partial derivative generator configured to determine texture coordinate partial derivatives;

a singular value generator configured to generate singular values of a Jacobian matrix constructed from the texture coordinate partial derivatives comprising:

the singular value generator further configured to generate linear combinations of the matrix elements with predefined constant coefficients;

the singular value generator further configured to form two two-dimensional vectors from the linear combinations;

the singular value generator further configured to generate Euclidian norms or approximation of Euclidian norms, from the vectors; and

the singular value generator further configured to generate singular values as linear combinations of the norms

an anisotropic ratio generator configured to generate an anisotropic ratio from singular values;

an anisotropic line generator configured to generate an anisotropic line from singular value decomposition comprising:

the anisotropic line generator further configured to determine the reciprocal of the Euclidian norms;

the anisotropic line generator further configured to create linear combinations of the reciprocals of the linear combinations and generate two two-dimensional vectors from the linear combinations; and

the anisotropic line generator further configured to generate the Euclidean norm of the vectors;

a level of detail (LOD) generator configured to generate anisotropic LOD or isotropic LOD;

a texel fetcher configured to select texels;

a texel filter configured to filter texels; and

a memory configured to store data comprising at least one of a texel, texture data, or a frame buffer.

12. The texture unit of claim 11 wherein approximation of Euclidian norms is generated by using a polygonal Minkowski norm.

13. A computer readable storage medium comprising:

a first set of instructions adapted to provide a processor, wherein the processor is configured to implement a second set of instructions, the second set of instructions comprising:

an input code segment for inputting at least one texture coordinate as a current texture map position and if filtering based on a projection of a pixel into a texture space is configured:

a determination code segment for determining texture coordinate partial derivatives; and

a generation code segment for generating singular values of a Jacobian matrix constructed from the texture coordinate partial derivatives comprising:

a generation code segment for generating linear combinations of the matrix elements with predefined constant coefficients;

a formation code segment for forming two two-dimensional vectors from the linear combinations;

a generation code segment for generating Euclidian norms or approximation of Euclidian norms, from the vectors; and

a generation code segment for generating singular values as linear combinations of the norms.

14. The computer readable storage medium of claim 13, wherein filtering based on a projection of a pixel into a texture space is mipmapping or anisotropic filtering.

15. The computer readable storage medium of claim 14 wherein the second set of instructions further comprises:

if mipmapping is configured:

if anisotropic filtering is not configured:

a determination code segment for determining an isotropic level of detail (LOD) from a selected singular value from the generated singular values;

a selection code segment for selecting a texture map based upon the LOD;

an update code segment for updating the current texture map position; and

a selection code segment for selecting texture data based upon the current texture map position.

16. The computer readable storage medium of claim 14 wherein the second set of instructions further comprises:

if mipmapping is configured:

if anisotropic filtering is configured:

a generation code segment for generating an anisotropic level of detail (LOD) from a selected singular value from the generated singular values;

a selection code segment for selecting a texture map based upon the LOD; and

an update code segment for updating the current texture map position.

17. The computer readable storage medium of claim 16 wherein the second set of instructions further comprises:

a generation code segment for generating anisotropic ratio and anisotropic line from singular value decomposition comprising:

a determination code segment for determining the reciprocal of the Euclidian norms;

a creation code segment for creating linear combinations of the reciprocals of the linear combinations and generating two two-dimensional vectors from the linear combinations; and

a generation code segment for generating the Euclidean norm of the vectors;

an adder code segment for adding at least one anisotropic offset to update the current texture map position; and

a selection code segment for selecting at least one texture mapping data based upon the current texture map position.

18. The computer readable storage medium of claim 14 wherein the second set of instructions further comprises:

if mipmapping is not configured:

a generation code segment for generating an anisotropic ratio and an anisotropic line from singular value decomposition comprising:

a determination code segment for determining the reciprocal of the Euclidian norms;

a creation code segment for creating linear combinations of the reciprocals of the linear combinations and generating two two-dimensional vectors from the linear combinations; and

a generation code segment for generating the Euclidean norm of the vectors;

an adder code segment for adding at least one anisotropic offset to update the current texture map position; and

a selection code segment for selecting at least one texture mapping data based upon the current texture map position.

19. The computer readable storage medium of claim 14 wherein the second set of instructions further comprises:

selecting texture mapping data based upon the current texture map position.

20. A computer readable storage medium comprising:

a first set of instructions adapted to provide a processor, wherein the processor is configured to implement a second set of instructions, the second set of instructions comprising:

an input code segment for inputting at least one texture coordinate as a current texture map position;

if mipmapping or anisotropic filtering is configured:

a determination code segment for determining texture coordinate partial derivatives;

a generation code segment for generating singular values of a Jacobian matrix constructed from the texture coordinate partial derivatives comprising:

a generation code segment for generating linear combinations of the matrix elements with predefined constant coefficients;

a formation code segment for forming two two-dimensional vectors from the linear combinations;

a generation code segment for generating Euclidian norms or approximation of Euclidian norms, from the vectors; and

a generation code segment for generating singular values as linear combinations of the norms;

if mipmapping is configured:

if anisotropic filtering is not configured:

a generation code segment for generating an isotropic level of detail (LOD) from a selected singular value from the generated singular values;

a selection code segment for selecting a texture map based upon the LOD;

an update code segment for updating the current texture map position;

if anisotropic filtering is configured:

a generation code segment for generating an anisotropic level of detail (LOD) from a selected singular value from the generated singular values;

a selection code segment for selecting a texture map based upon the LOD;

an update code segment for updating the current texture map position;

if anisotropic filtering is configured:

a generation code segment for generating an anisotropic ratio and an anisotropic line from singular value decomposition comprising:

a determination code segment for determining the reciprocal of the Euclidian norms;

a creation code segment for creating linear combinations of the reciprocals of the linear combinations and generating two two-dimensional vectors from the linear combinations; and

a generation code segment for generating the Euclidean norm of the vectors;

an adder code segment for adding at least one anisotropic offset to update the current texture map position; and

a selection code segment for selecting at least one texture mapping data based upon the current texture map position.

21. The computer readable storage medium of claim 13 wherein approximation of Euclidian norms is generated by using a polygonal Minkowski norm.

22. The computer readable storage medium of claim 20 wherein approximation of Euclidian norms is generated by using a polygonal Minkowski norm.

23. The computer readable medium of claim 13 wherein the first set of instructions or the second set of instructions are hardware description language (HDL) instructions.

24. The computer readable medium of claim 20 wherein the first set of instructions or the second set of instructions are hardware description language (HDL) instructions.

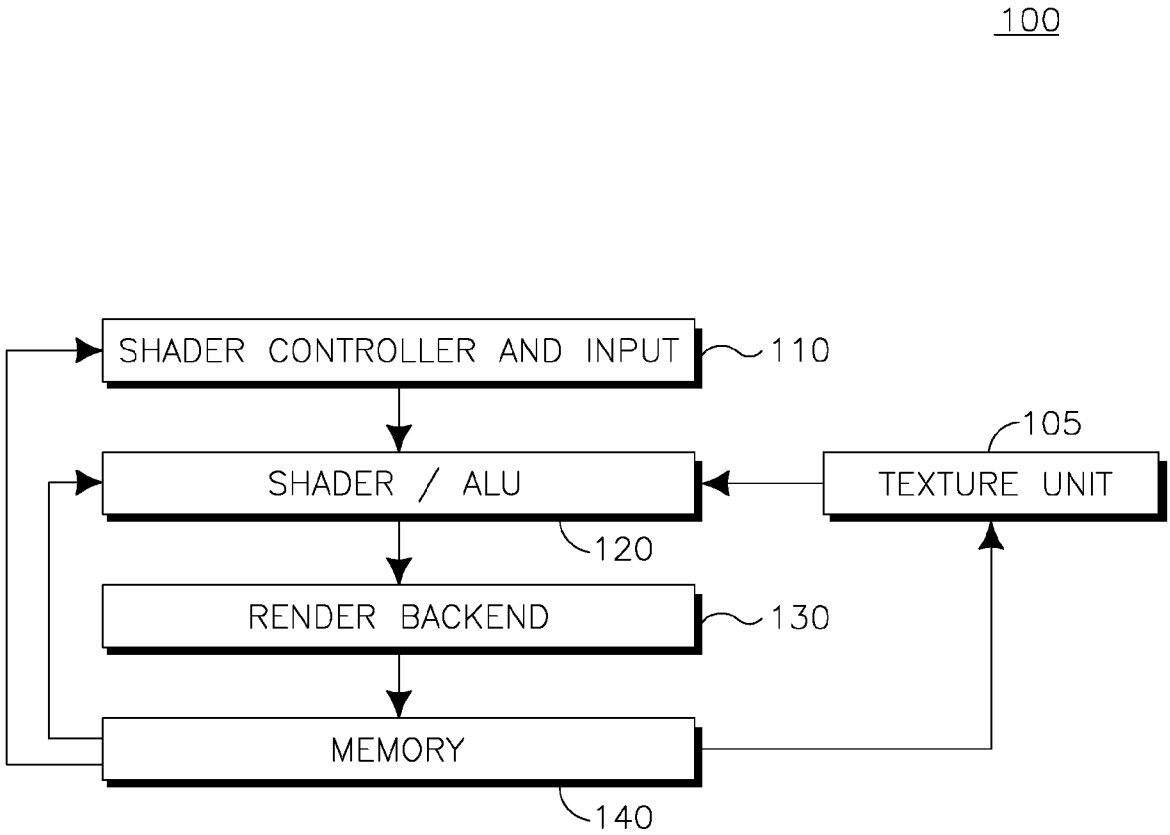

100
SHADER CONTROLLER AND INPUT
110
105
SHADER / ALU
TEXTURE UNIT
120
RENDER BACKEND
130
MEMORY
140

FIG. 1

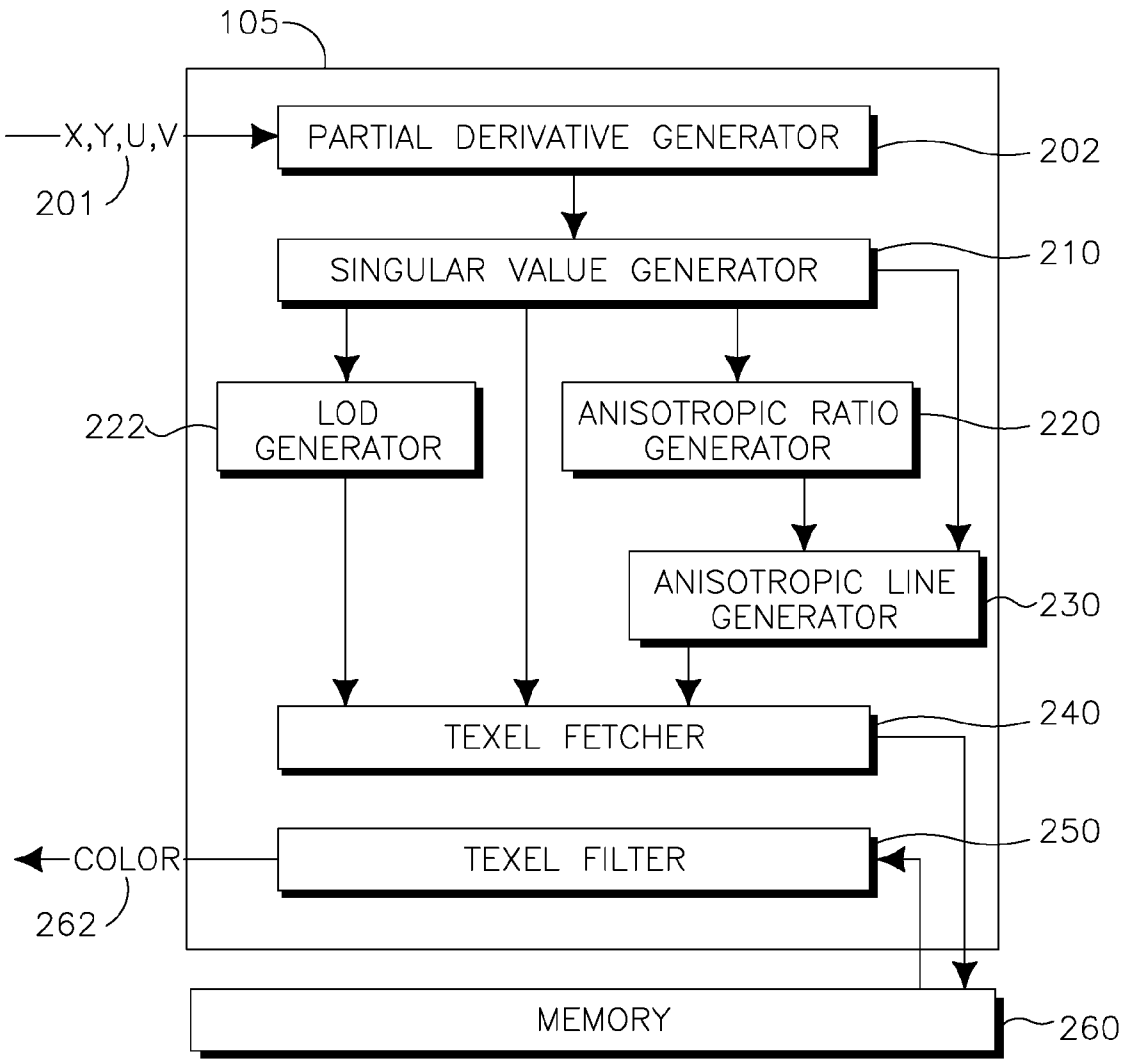

105
X,Y,U,V
201
PARTIAL DERIVATIVE GENERATOR
202
SINGULAR VALUE GENERATOR
210
222
LOD GENERATOR
ANISOTROPIC RATIO GENERATOR
220
ANISOTROPIC LINE GENERATOR
230
TEXEL FETCHER
240
COLOR
262
TEXEL FILTER
250
MEMORY
260

FIG. 2

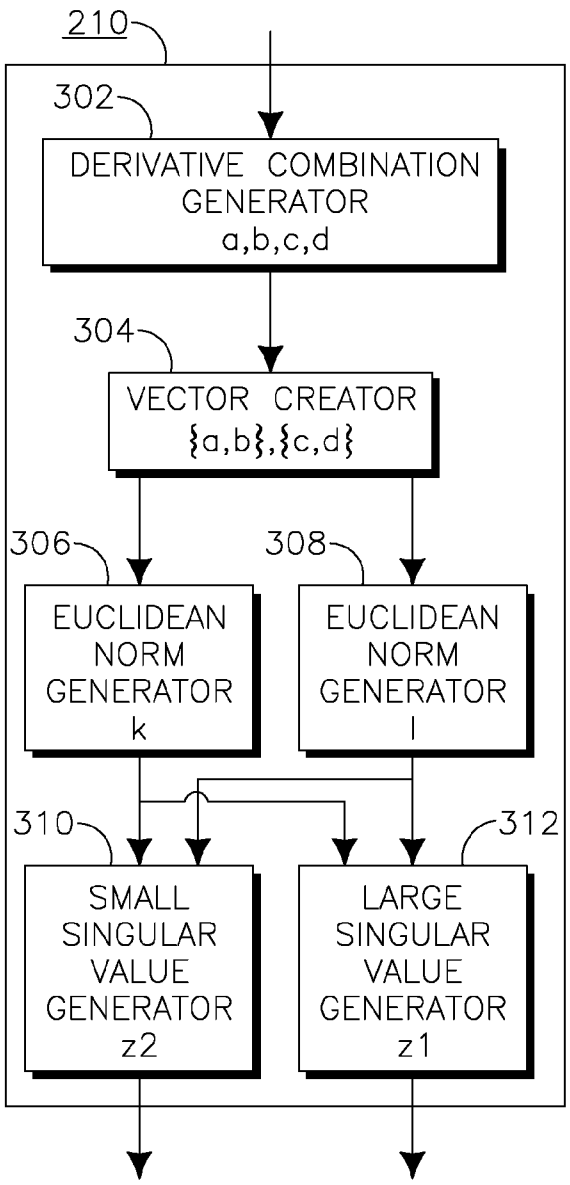

210
302
DERIVATIVE COMBINATION GENERATOR a,b,c,d
304
VECTOR CREATOR {a,b},{c,d}
306
EUCLIDEAN NORM GENERATOR k
308
EUCLIDEAN NORM GENERATOR l
310
SMALL SINGULAR VALUE GENERATOR z2
312
LARGE SINGULAR VALUE GENERATOR z1

FIG. 3

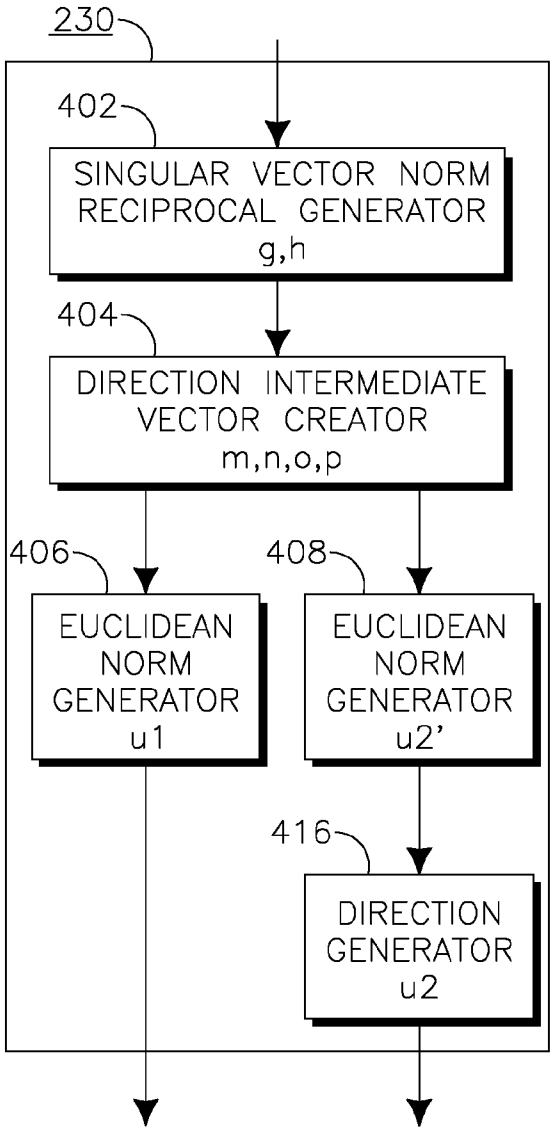

230
402
SINGULAR VECTOR NORM RECIPROCAL GENERATOR g,h
404
DIRECTION INTERMEDIATE VECTOR CREATOR m,n,o,p
406
EUCLIDEAN NORM GENERATOR u1
408
EUCLIDEAN NORM GENERATOR u2'
416
DIRECTION GENERATOR u2

FIG. 4

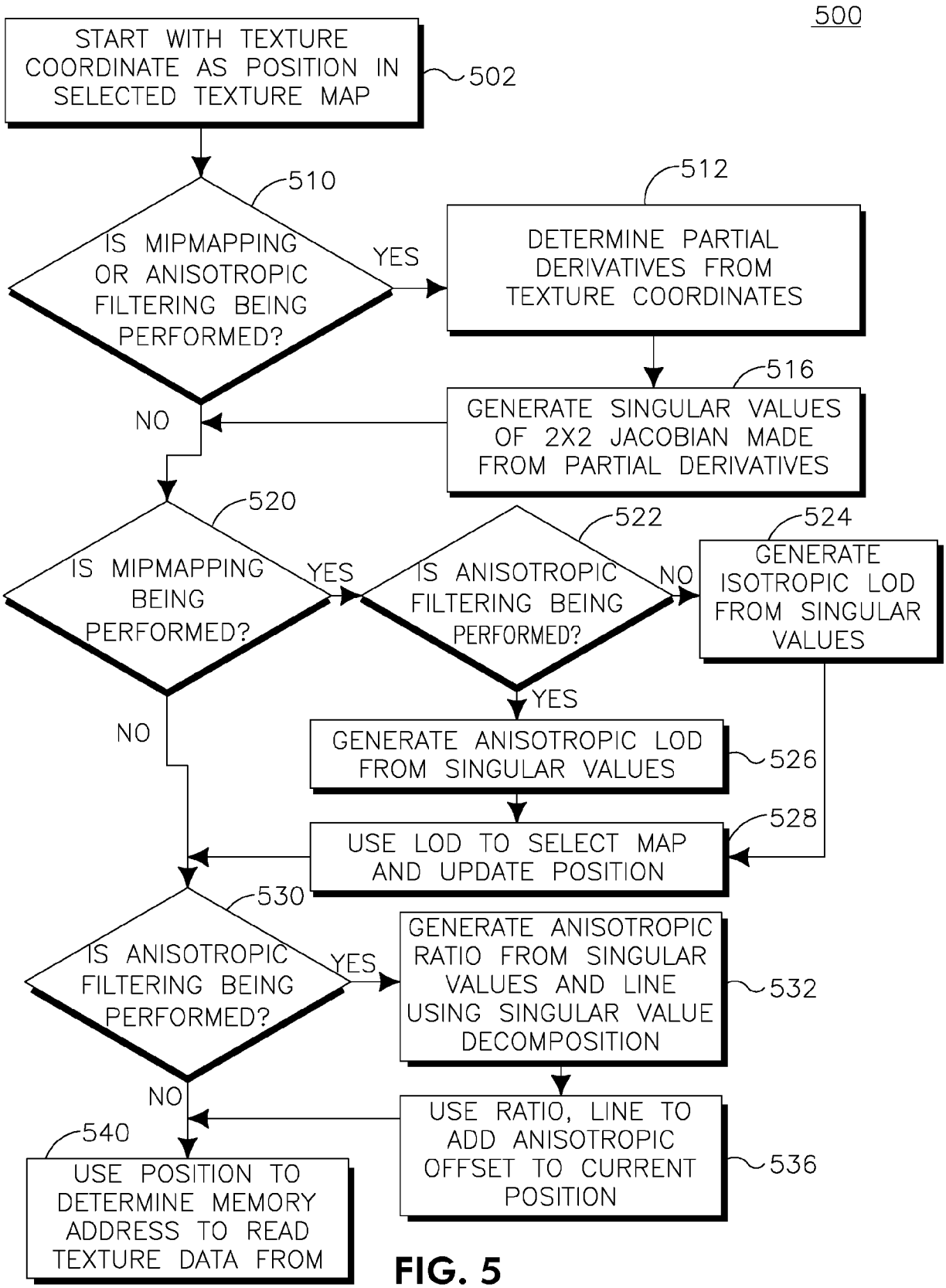

500
START WITH TEXTURE COORDINATE AS POSITION IN SELECTED TEXTURE MAP
502
510
IS MIPMAPPING OR ANISOTROPIC FILTERING BEING PERFORMED?
YES
512
DETERMINE PARTIAL DERIVATIVES FROM TEXTURE COORDINATES
516
GENERATE SINGULAR VALUES OF 2X2 JACOBIAN MADE FROM PARTIAL DERIVATIVES
NO
520
IS MIPMAPPING BEING PERFORMED?
YES
522
IS ANISOTROPIC FILTERING BEING PERFORMED?
NO
524
GENERATE ISOTROPIC LOD FROM SINGULAR VALUES
YES
NO
GENERATE ANISOTROPIC LOD FROM SINGULAR VALUES
526
528
USE LOD TO SELECT MAP AND UPDATE POSITION
530
IS ANISOTROPIC FILTERING BEING PERFORMED?
YES
GENERATE ANISOTROPIC RATIO FROM SINGULAR VALUES AND LINE USING SINGULAR VALUE DECOMPOSITION
532
NO
USE RATIO, LINE TO ADD ANISOTROPIC OFFSET TO CURRENT POSITION
536
540
USE POSITION TO DETERMINE MEMORY ADDRESS TO READ TEXTURE DATA FROM

FIG. 5

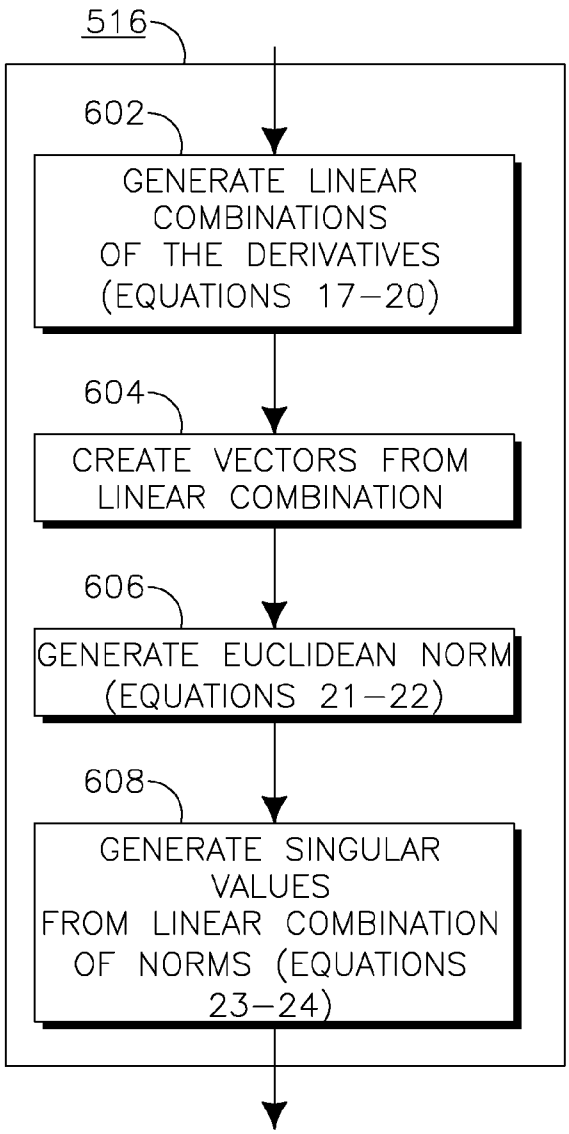

516
602
GENERATE LINEAR COMBINATIONS OF THE DERIVATIVES (EQUATIONS 17–20)
604
CREATE VECTORS FROM LINEAR COMBINATION
606
GENERATE EUCLIDEAN NORM (EQUATIONS 21–22)
608
GENERATE SINGULAR VALUES FROM LINEAR COMBINATION OF NORMS (EQUATIONS 23–24)

FIG. 6

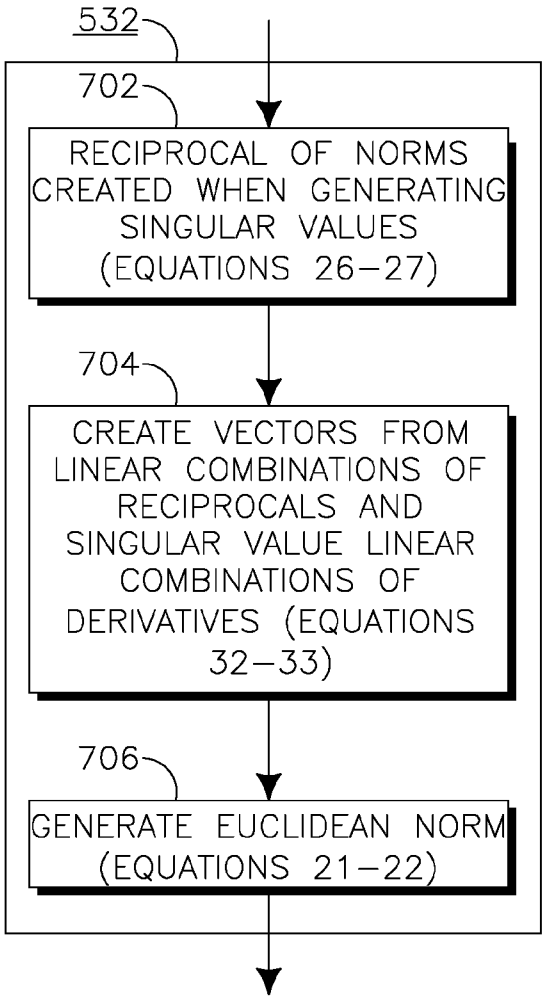

532
702
RECIPROCAL OF NORMS CREATED WHEN GENERATING SINGULAR VALUES (EQUATIONS 26–27)
704
CREATE VECTORS FROM LINEAR COMBINATIONS OF RECIPROCALS AND SINGULAR VALUE LINEAR COMBINATIONS OF DERIVATIVES (EQUATIONS 32–33)
706
GENERATE EUCLIDEAN NORM (EQUATIONS 21–22)

FIG. 7

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2010/028301

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06T15/00
ADD.

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06T

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, COMPENDEX, INSPEC, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	WO 2006/031213 A1 (SILICON OPTIX INC [US]; LACHINE VLADIMIR [CA]; SMITH GREGORY L [CA]; L) 23 March 2006 (2006-03-23) page 4, line 13 - page 5, line 12 page 8, line 26 - page 11, line 15 figures 1,2,3,5A ----- -/--	1-24

[X] Further documents are listed in the continuation of Box C.

[X] See patent family annex.

* Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
24 August 2010	31/08/2010

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Gauthier, J

2

Form PCT/ISA/210 (second sheet) (April 2005)

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 1 434 171 A2 (MICROSOFT CORP [US]) 30 June 2004 (2004-06-30) page 4, paragraph 0026 - paragraph 0027 page 5, paragraph 0031 - paragraph 0032 page 16, paragraph 0152 page 17, paragraph 0171 - page 18, paragraph 0172 page 34, paragraph 0319 - page 35 page 38, paragraph 0348 page 62, paragraph 0566 - page 3, paragraph 0573 page 63, paragraph 0577 - page 64, paragraph 0579 page 64, paragraph 0586 - page 65, paragraph 0592 page 65, paragraph 0594 page 66, paragraph 0596 page 66, paragraph 0602 - page 67, paragraph 0608 page 68, paragraph 0614 page 68, paragraph 0617 - paragraph 0619 page 69, paragraph 0621 - paragraph 0622 figures 9b,10,11,37,38,39,40	1-24
X	US 2004/257376 A1 (LIAO QUN FENG FRED [US] ET AL) 23 December 2004 (2004-12-23) figures 8,11 page 1, paragraph 0005 - page 2, paragraph 0015	1-24
X	US 2007/182753 A1 (ISIDORO JOHN R [US] ET AL) 9 August 2007 (2007-08-09) figures 2,3,4,5,6 page 1, paragraph 0006 page 1, paragraph 0008 - page 2, paragraph 0012 page 4, paragraphs 0032,0034,0036 page 5, paragraph 0037 - paragraph 0039 page 6, paragraph 0046 - page 7, paragraph 0048 page 7, paragraph 0051 page 8, paragraph 0058 - paragraph 0060	1-24
X	US 7 369 136 B1 (HECKBERT PAUL S [US] ET AL) 6 May 2008 (2008-05-06) figures 1,2,3,4 column 1, line 20 - column 2, line 67 column 3, line 56 - column 4, line 17 column 6, line 19 - line 56 column 7, line 23 - column 9, line 13	1-24

-/--

2

INTERNATIONAL SEARCH REPORT

International application No
PCT/US2010/028301

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 6 987 517 B1 (DONOVAN WALTER E [US] ET AL) 17 January 2006 (2006-01-17) figures 6,7,8 page 1, line 17 - line 30 column 8, line 40 - line 55 column 9, line 47 - column 11, line 57	1-24
A	EWINS J P ET AL: "Implementing an anisotropic texture filter" COMPUTERS AND GRAPHICS, ELSEVIER, GB LNKD-DOI:10.1016/S0097-8493(99)00159-4, vol. 24, no. 2, 1 April 2000 (2000-04-01), pages 253-267, XP004236371 ISSN: 0097-8493 page 255, left-hand column, paragraph 2.1 - page 256, right-hand column, paragraph 2.2 page 258, left-hand column, paragraph 3.2.2 - page 260, left-hand column, last line, paragraph 3.2.4 page 261, left-hand column, paragraph 3.2.7 - right-hand column, last line figures 1,2,5,6,7,9,12	1-24
A	EWINS J P ET AL: "MIP-MAP LEVEL SELECTION FOR TEXTURE MAPPING" IEEE TRANSACTIONS ON VISUALIZATION AND COMPUTER GRAPHICS, IEEE SERVICE CENTER, LOS ALAMITOS, CA, US LNKD-DOI:10.1109/2945.765326, vol. 4, no. 4, 1 October 1998 (1998-10-01), pages 317-329, XP000831199 ISSN: 1077-2626 page 318, left-hand column, paragraph 1.1 - page 319, right-hand column page 325, left-hand column, paragraph 3 - page 327, left-hand column, last line	1-24
A	SHIRMAN L ET AL: "A new look at mipmap level estimation techniques" COMPUTERS AND GRAPHICS, ELSEVIER, GB LNKD-DOI:10.1016/S0097-8493(99)00032-1, vol. 23, no. 2, 1 April 1999 (1999-04-01), pages 223-231, XP004165786 ISSN: 0097-8493 page 224, right-hand column, paragraph 2.2 - page 225, left-hand column, last line page 228, right-hand column, paragraph 5 - page 231, left-hand column, last line	1-24

2

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/US2010/028301

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2006031213 A1	23-03-2006	CN 101142614 A	12-03-2008
		EP 1800289 A1	27-06-2007
		JP 2008512766 T	24-04-2008
EP 1434171 A2	30-06-2004	EP 1434172 A2	30-06-2004
US 2004257376 A1	23-12-2004	NONE	
US 2007182753 A1	09-08-2007	NONE	
US 7369136 B1	06-05-2008	NONE	
US 6987517 B1	17-01-2006	NONE	