International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:7, No:1, 2013

A GPU Based Texture Mapping Technique for 3D Models using Multi-View Images

In Lee, Kyung-Kyu Kang, Jaewoon Lee, and Dongho Kim

***Abstract*—**Previous the 3D model texture generation from multi-view images and mapping algorithms has issues in the texture chart generation which are the self-intersection and the concentration of the texture in texture space. Also we may suffer from some problems due to the occluded areas, such as inside parts of thighs. In this paper we propose a texture mapping technique for 3D models using multi-view images on the GPU. We do texture mapping directly on the GPU fragment shader per pixel without generation of the texture map. And we solve for the occluded area using the 3D model depth information. Our method needs more calculation on the GPU than previous works, but it has shown real-time performance and previously mentioned problems do not occur.

***Keywords*—**Texture Mapping, Multi-view Images, Camera Calibration, GPU Shader.

I. Introduction

THE texture of the 3D model which is made from voxel-carving [1] algorithm is created by storing the texture information from acquired multi-view images to the texture map. The 3D model is segmented for generation of the texture chart [2] using parameterization [3] algorithm which will be used for making texture map on the image space.

The texture information is acquired from multi-view images using camera parameters for each image and the polygon vertices (x, y, z) of the 3D model. Once the texture information is stored in the texture map it is used for texture mapping on the3D model per each frame.

But if the occluded area like inside of the thighs is not considered when the generation of texture, the texture mapping result is not clear because the occluded area will be mapped for using outside texture information from the selected image. And some of the texture chart suffers from the concentration of the texture space of the texture chart. It will be shown that the texture mapping result seems to be blurred. Also after the parameterization process, because it is angle-based algorithm, the some of the texture charts have a self-intersection problem. In this paper we propose a texture mapping technique for 3D models using multi-view images on the GPU. For solving a mentioned problem, we do the texture mapping directly on GPU shader without generation of the texture map. Also, we use the depth map for calculation of the occluded area in the texture mapping process. Our algorithm solves the previous mentioned problems fundamentally and performs in real-time frame rate, per pixel.

In Lee is with the Soongsil University, Dongjak-gu, Seoul, Korea (phone: +82-2-821-2889; fax: +82-2-821-2889; e-mail: einable@magiclab.kr).

Kyung-Kyu Kang is with the Soongsil University, Dongjak-gu, Seoul, Korea (phone: +82-2-821-2889; fax: +82-2-821-2889; e-mail: rcrookie@ magiclab.kr).

Jae-Woon Lee is with the Soongsil University, Dongjak-gu, Seoul, Korea (phone: +82-2-821-2889; fax: +82-2-821-2889; e-mail: woori9000@ magiclab.kr).

Dongho Kim is with the Soongsil University, Dongjak-gu, Seoul, Korea (phone: +82-2-820-0721; fax: +82-2-821-2889; e-mail: cg@su.ac.kr).

II. Related Works

The texture map is generated for packing the texture chart [2] on the image space. It is made that the 3D model is segmented by using the segmentation area decision algorithm like the SOD (Second Order Difference) [4]. The segmented mesh is converted from 3D to 2D space using parameterization [4] algorithm such as LSCM [2] or ABF [5]. But it has a possibility that the texture chart has a concentration of the texture space because it conducts angle-based. Also it has the self-intersection problem in some of the texture chart that the texture information corresponding to intersected area will be duplicated. It means that the some part of the texture in the texture chart is not represented.

The multi-view images are used for generation of the texture on the texture map. The camera calibration information needs for acquiring the texture from the multi-view images. It is the process of the finding camera information that consists of intrinsic and extrinsic parameters and represents 3x3, 3x4 matrix form respectively. For finding the suitable image for texture acquisition, find the image by calculating the smallest angle between a normal vector of the polygon in the 3D model with the camera direction vector corresponding to each image. Next, project 3D model vertex(x, y, z) to the selected image using corresponding camera projection matrix that is calculated by multiplying between intrinsic and extrinsic parameters.

For finding the camera parameters, Tsai [6] proposed using calibration board such as black-white chessboard. He is used two perpendicular calibration board for measurement of the camera parameter, but Zhang [7] proposed the new approach that uses many pictures of one calibration board.

The some of the previous research consider the texture mapping of the occluded area [8]. Proposes the texture mapping technique for the 3D human model made from the modeling tool. It is used for the depth map, normal vector of the vertex and calculated before texture mapping [9]. Proposes for texture mapping method of the occluded area for the visual hull [10]. Also it proposes the soft visibility map method for detection of the occluded area and lerp texture mapping. It calculates before the texture mapping on the GPU.

III. GPU based Texture Mapping

In this section we will present our GPU based texture mapping technique for 3D models using multi-view images.

Our algorithm has two parts that is depth map creation before texture mapping and detection of the occlusion for selecting the texture acquired image on the GPU fragment shader.

A. Depth Map Creation

Some of the previous texture mapping algorithms based on the multi-view images is based on the angle between camera direction vector and polygon normal vector for selection of the texture acquisition image. But it has a mismatch problem on the occlusion area's texture mapping like Fig. 1 because the camera just only available to take a 2D image. If the occlusion is not considered when performing the texture mapping, its result will be mismatched.

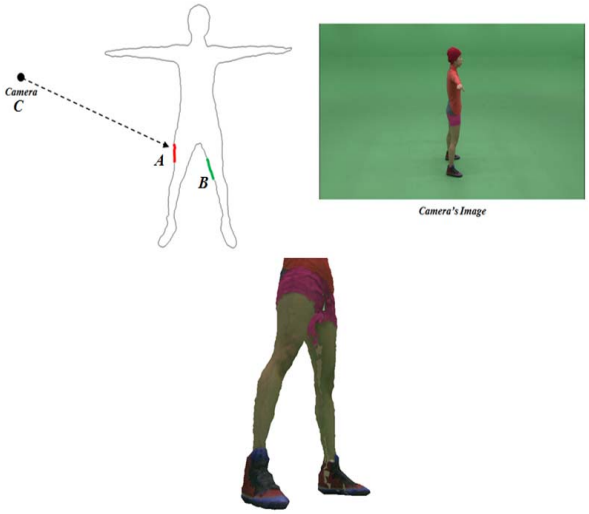

Fig. 1 The case of the occlusion problem. For finding of the B's texture (top left), the image taken from camera C is selected (top right) and the mapping result using C's image on the inside of the thigh.

In Fig. 1, the surface information of the B is not existence in the top-right image. But some of the multi-view images have information of surface B. For this situation we create and use the depth maps for detection of the occluded area like [8] before do the texture mapping on the GPU fragment shader.

The depth map for the 3D model is made using for each camera parameters. The conventional transformation from 3D to 2D in the computer graphics needs model-view, perspective projection and viewport transformation matrix. We made the each transform matrix using camera parameters.

The model-view transformation matrix, convert world to camera coordinates, is consist of the rotation information and transformation in the camera coordinates. The extrinsic parameter is already represented it, we just convert 3x4 to 4x4 matrix form. But (y,z) coordinates of the camera space is converted for world coordinates system, we reverse the extrinsic matrix (y,z) elements sign for creation of model-view matrix equally. The perspective projection transformation matrix needs FOV(field of view), aspect ratio, near distance and far distance of z coordinates value. The FOV is calculated using principle point $O_c(O_x, O_y)$, the coordinates of 0 of the y in $O_c'(O_x, 0)$ and focal length like an equation below:

$$focalLength = -f / S_x \tag{1}$$

$$fov = \tan^{-1}\left(\frac{\sqrt{(O_c - O_c')^2}}{focalLength}\right) \times 2 \tag{2}$$

The aspect ratio we use principle point Ox /Oy division result. And near and far distance value we set the 3000, 7000 respectively. The last is viewport transformation value we set the image size like (0, 0, $O_x \times 2, O_y \times 2$).

All of the values for creation of the depth map are done, we create the depth map corresponding for each image. Fig. 2 is shown overall process of creation of the depth maps.

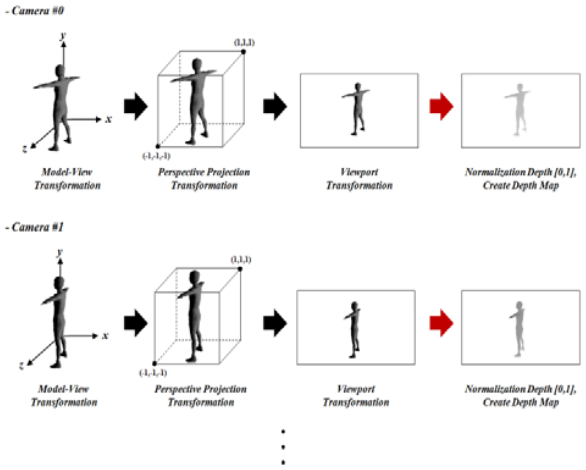

Fig. 2 The depth map generation process for each image

B. GPU Based Texture Mapping Technique

The key concept of the proposing texture mapping algorithm is the process of the occlusion areas before do the texture mapping using depth maps. For solving it, our texture mapping consists of the creation of the texture acquisition candidate array, comparison of the occlusion for selection of the texture acquisition image and fetching the texture and mapping. It is performed on the GPU fragment shader per pixel in real-time per each frame.

The creation of the texture acquisition candidate array, the first step of our texture mapping algorithm, consists of the (camera number, angle) set which is used for comparison of the occlusion on the next step. The angle is calculated by the dot product between the GPU fragment shader's normal vector and for each camera direction vector. The camera direction vector is calculated using extrinsic matrix using (3) equation. The set for each camera is calculated, we sort the smallest angle value first and store to the array. This process is important for enhancement of the hit ratio for real-time performance on the

$$ModelView = \begin{bmatrix} r_{11} & r_{12} & r_{13} & -R_1^T T \\ -r_{21} & -r_{22} & -r_{23} & R_2^T T \\ -r_{31} & -r_{32} & -r_{33} & R_3^T T \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

GPU fragment shader that only some areas of the 3D model are shown the occlusion from the cameras. Fig. 3 is shown the process of the creation of the texture acquisition candidate array.

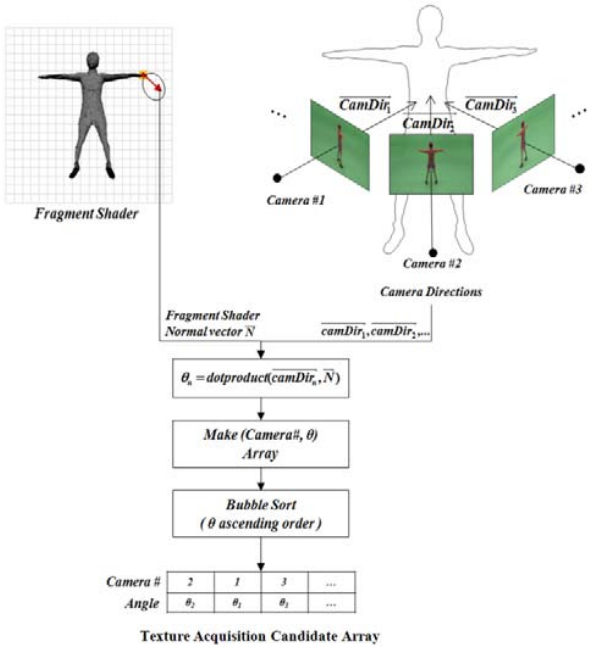

Fig. 3 The process of creation of the texture acquisiton candidate array

The comparison of the occlusion is performed for comparison of the occlusion area. The GPU fragment shader performs the pixel value calculation. So this process is the decision of the texture acquired image for the decision of the pixel color. It uses camera number in the texture acquisition candidate array sequentially, transformation matrices generated above and depth maps. The GPU fragment shader provides the vertex information of the output pixel. It is transformed using each transformation matrices for calculation of the position of the depth map and depth value. After the perspective projection transformation, the depth value is calculated and after the viewport transformation, the position of the depth map is calculated. We compare the depth value and value fetched from the depth map. If it is equal, the area is not occluded. Else this area is an occlusion area using its camera number. In other words, the image corresponding camera number is not acquired the texture information and performs previous process using next camera number in the texture acquisition candidate array.

The image number for acquisition of the texture is selected, fetch the texture information of the image and return the texture value by multiplying between the GPU fragment shader's vertex and the camera projection matrix corresponding to selected image number. Fig. 4 is shown the process of the selection of the image.

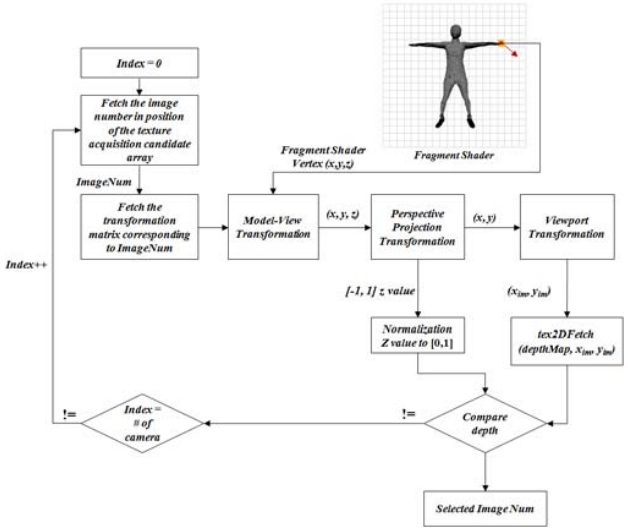

Fig. 4 Image selection process for texture acquisition

IV. Experiments

We implement our algorithm in OpenGL and Cg Shader and was run on an Intel i7 @ 2.93GHz, 8GB RAM, NVIDIA GeForce GTX 480. We use the 3D model, twenty 1280 x 720 multi-view images and the camera parameters for each image. Fig.7 shows the 3D model vertex and normal information made from Fig. 8's multi-view images.

Before do the real-time texture mapping on the GPU fragment shader, we make the Fig. 7 depth map for comparison of occlusion area using the camera parameter information for each image.

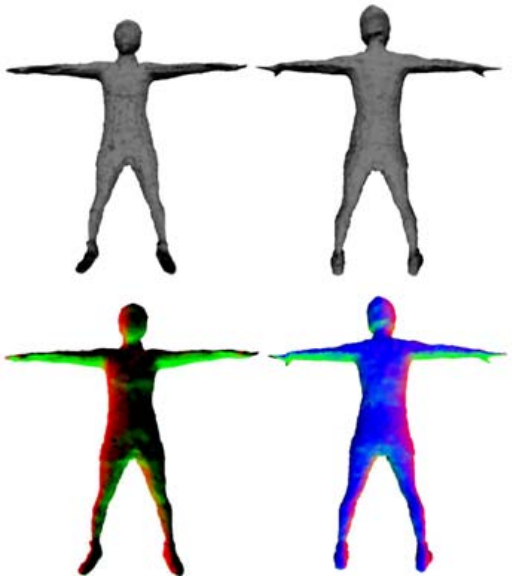

Fig. 5 The 3D model vertex (Top) and normal vector (Bottom)

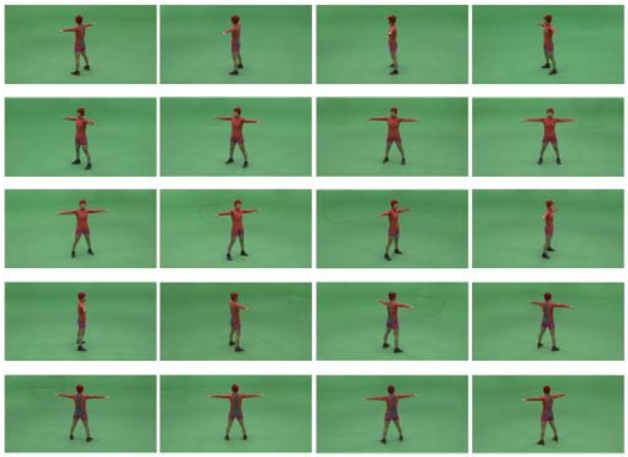

Fig. 6 Twenty multi-view images for using our experiment

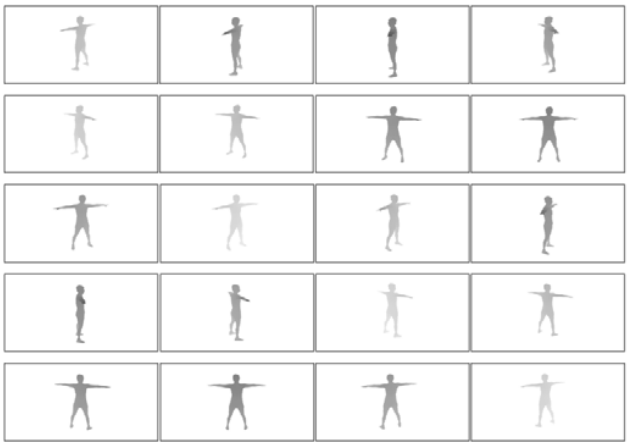

Fig. 7 The depth map generation results using corresponding camera information and the 3D model

The Fig. 8 is our texture mapping result. It is shown that our algorithm is well performed and the occluded area inside of the thighs is possible.

But, some parts of the occlusion area (crotch areas in Fig. 8) have not performed because the texture information of this area has not in images. Our test model is made by automatic voxel-carving algorithm, so some areas are unnecessary. Fig. 9 is shown the comparison of crotch areas between an image that is suitable case for fetching texture information and our mapping result.

As mentioned above, previous researches had shown the texture chart concentration, self-intersection problems. Our algorithm works per each pixel so it is possible to solve fundamentally. But, it needs much calculation because out texture mapping algorithm was run on the fragment shader and performs per each frame. Nevertheless, our algorithm in our test environment is shown about 60 FPS in real-time.

Fig. 8 The result of our GPU based texture mapping

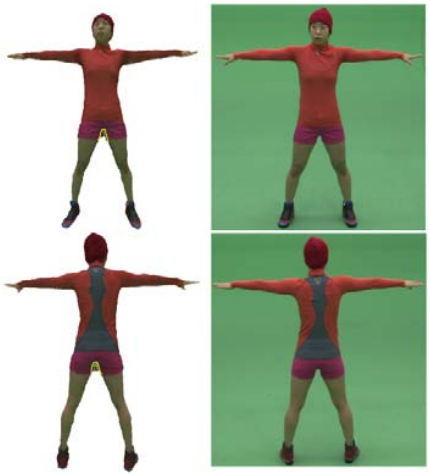

Fig. 9 The crotch area comparison between our texture mapping results(left top and bottom) and related images(right top and bottom)

V. Conclusion and Future Work

We have proposed a texture mapping technique for 3D models using multi-view images on the GPU fragment shader. Previous problems of the proposed algorithm mentioned above are shown in the texture map generation process. But we conduct the texture mapping on the GPU directly without generation of the texture map, the problems have not occurred fundamentally.

However, because our algorithm operates on the GPU fragment shader, it needs a calculation for texture mapping per each frame. Despite of the much calculation, it has shown 60 frames per second in real-time performance on our experimental environment.

But it still has a problem that some part of the texture mapped seam area is unmatched because our algorithm is performed using multi-view images. Our future work will solve the unmatched mapping problem on the GPU fragment shader. [11] proposes the solving method of this problems, but it just applies

using texture map. So we need another approach for our case.

And the resolution of our texture mapping result is low because the 3D model size is big. Also we try to enhance the texture mapping quality, called superresolution [12]. But previously proposed algorithm performs on the image space, we will try directly to it on the GPU in real-time.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No.2011-0015620).

This paper was supported by BK 21 Program in Soongsil University.

References

[1] K. N. Kutulakos, S. M. Seitz, “A Theory of shape by space carving”, International Journal of Computer Vision, Vol.38, No.3, pp.199-218, 2000.

[2] B. Lévy, S. Petitjean, N. Ray, J. Maillot, "Least Square Conformal Maps for Automatic Texture Atlas Generation", ACM Transactions on Graphics, Vol.21, No.3, pp.362-371, 2002.

[3] A. Sheffer, E. de. Sturler, "Parameterization of Faceted Surface for Meshing using Angle-Based Flattening", Engineering with Computers, Vol.17, No.3, pp.326-337, 2001.

[4] A. Hubeli, M. Gross, “Multiresolution features extraction from unstructured meshes”, In Proc. of IEEE Visualization Conf., 2001.

[5] A. Sheffer, B. Lévy, M. Mogilnitsky, A. Bogomyakov, "ABF++: Fast and Robust Angle Based Flattening", ACM Transactions on Graphics, Vol.24, No.2, pp.311-330, 2005.

[6] R. Tsai, "A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology using Off-the-shelf TV Camers and Lenses", IEEE Journal of Robotics and Automation, Vol. RA-3, No.4, 1987.

[7] Z. Zhang, "A Flexible New Technique for Camera Calibration", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.22, No.11, pp.1330-1334, 2000.

[8] J. Carranza, C. Theobalt, M. A. Magnor, H. P. Seidel, "Free-Viewpoint Video of Human Actors", ACM Transactions on Graphics, Vol.22, No.3, pp.569-577, 2003.

[9] M. Eisemann, B. D. Decker, M. Magnor, P. Bekaert, E. de. Aguiar, N. Ahmed, C. Theobalt, A. Sellent, "Floating Textures", Proceedings of the Eurographics, Vol.27, No.2, pp.409-418, 2008.

[10] A. Laurentini, “The Visual Hull Concept for Silhouette-Based Image Understanding”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol.16, No.2, pp.150-162, 1994.

[11] R. Gal, Y. Wexler, E. Ofek, H. Hoppe, Seamless Montage for Texturing Models, Computer Grpahics Forum(Eurographics), Vol.23, No2, pp.479-486, 2010.

[12] G. Bastian, C. Daniel, "Superresolution Texture Maps for Multiview Reconstruction", IEEE International Conference on Computer Vision, pp.1677-1684, 2009.