Applied Mathematics & Information Sciences
An International Journal

Realistic and Detail Rendering of Village Virtual Scene Based on Pixel Offset

Chunjiang Zhao, Huarui Wu and Ronghua Gao

National Engineering Research Center for Information Technology in Agriculture, Beijing 100097, China
Key Laboratory for Information Technologies in Agriculture, Ministry of Agriculture, Beijing 100097, China
Email: zhaocj@nercita.org.cn

Received: 21 Feb. 2012; Revised 16 May 2012; Accepted 03 Jul. 2012

Abstract: Many complex detail objects in the village virtual scene are important to the photo-realistic rendering technology. Bump mapping, normal mapping and parallax mapping have a capability to represent the realistic rendering by disturbing normal vector or shifting the texture coordinates along the view direction. However, an incorrect offset will lead to the sliding phenomenon in the detail rendering. In this paper, a corresponding relation between the offset and the current viewpoint is proposed by establishing mathematics function of the detail image pixel coordinates and the height values. Furthermore, the new pixel offset is calculated by using GPU technology. With GPU technology, sliding phenomenon does not appear when the viewpoint changes. Thus, the detail objects have the realistic effect in village virtual scene. The experimental results demonstrate that this proposed method has achieved better performance in representing the realistic and detail rendering than traditional methods of parallax mapping and texture coordinate offset correction with GPU technology.

Keywords: Realistic, bump mapping, parallax mapping, pixel offset, virtual scene

1 Introduction

It is an important research branch in virtual reality to make the virtual objects photo-realistic. There are two main approaches to render details of a scene: geometry-based rendering and image-based rendering. The former can generate details images from different viewpoints with 3-D data sets, such as polygon meshes and their surface properties. However a complex scene may involve large data sets that can be prohibitive to process in real-time. The latter renders details from different viewpoints by interpolating between photographic images, because rendering time does not depend on the scene's complexity, so it is easy to generate photorealistic output. However this approach tends the viewpoint and scene structure. Further, its calculations tend to be slow because of the lack of graphic hardware acceleration.

Texture mapping technology[1-2] can describe object details and reduce the complexity of the virtual scene when the object structure is complex. However, texture mapping technology just considers surface color, thus it can describe various decorative pattern designs only on a smooth surface, but can not present rough detail information brought about by bumpy micro geometry shape. When the viewpoint is close to the observation object, the demand for the model accuracy is higher, and the object detail information described by texture mapping technology cannot satisfy user's demand of visual experience. Of particular note, an image mapped onto object surfaces does not have coarseness detail effect within a surface. This problem is magnified when inspecting a texture-mapped surface with stereoscopic vision, because lack of binocular disparity between each mapped polygon is a depth cue that the surface it represents is flat. To solve this problem, the technology of bump mapping is proposed. It can simulate surface bumpy detail information without any overload of increasing extra polygons, and make the virtual scene with more realistic. But it lacks the motion parallax effect-an apparent effect of relative movement of the surface detail due to a viewpoint.

Parallax mapping[3] is a technology that can realize the 3D rendering on dynamic viewpoint of image details, and it can realize 3D parallax effect of image surface details through the texture coordinate offset. However, incorrect texture coordinate offset will lead to a sliding effect in the result, then reduction in realism. In order to cover this shortage of the parallax mapping technology, Donnelly[4] describes a ray tracer that uses 3D textures and a sphere tracing data structure, which

avoids the applications of parallax mapping, and realizes the 3D parallax effect of image surface details. But its efficiency is not high due to the ray tracing calculation. Tatarchuk[5], Policarpo[6] and some others describe an improved method based on traditional normal vector. It avoids the sliding effect of parallax mapping, but the change of parallax is not so big and the 3D effect is not obvious.

Along with the development of computer hardware, 3D rendering of image details under dynamic viewpoint develops further. Steep parallax mapping technology[7]can calculate out the self-projection of image detail information in the Pixel Shader and enhance the 3D display effect of the image details without the use of geometric data. Nevertheless, Steep parallax mapping technology costs much. It needs to use 20 or more layers to present the high quality display effect. This will waste a lot of time. It is more important that it takes up a lot of execution time of Pixel Shader, and this kind of Shader is 20 times slower than Normal Mapping. Instead of traditional vertex transformation and lighting equation to correct the deviation of texture coordinates[8], the custom program can use parameters to adjust and fix offset size, and draw it as near as possible to the correct value quickly, in order to alleviate texture sliding effect.

Many objects with detail information in the villages' virtual scene, such as art relief of custom building, scraggy wall, and so on. In order to solve the details rendering under the premise of not taking up too much time of system, a mapping function of image pixels coordinates and height values is defined in this paper, which can get a correct virtual pixel offset related to viewpoint through the pixel to pixel shift for details image. The experimental results demonstrate that the detail rendering method based on pixels offset does not increase algorithm complexity, and when the viewpoint draws near to the object surface gradually, surface details of object can present a 3D effect with no sliding phenomenon.

2 Method of Detail Rendering

2.1 Bump mapping

Bump mapping as a technique that makes a surface appear rough or wrinkled. This affect is achieved by perturbing the normal used in the illumination computation. Hence, it only affects the shading, not the underlying geometry. The bump map can contain height values that will affect the normal of each pixel. However, it can as an alternative contain normal that have been computed from such a bump map and this map is then called a normal map[9]. Especially moving frame bump mapping[10]is closely related to shading since it includes vector interpolation, which is the essence of Phong shading. Bump mapping is also closely related to texture mapping[11]. It will also suffer from similar aliasing problems when bumps are magnified or minified.

Blinn used an approximation for the perturbed normal, which depends on the surface normal and the height information in the bump map. The perturbed normal is calculated as

$$n' = n + \frac{F_u(n \times P_v) - F_v(n \times P_u)}{\|n\|}$$

Where n is the surface normal, P_u and P_v are the partial derivatives of the surface in the u and v directions respectively. F_u and F_v are the gradients of the bump map.

Bump mapping combines per-fragment lighting with surface normal perturbations supplied by an image, in order to simulate detail information on object surfaces. This effect is achieved without requiring excessive geometric tessellation of the surface. With bump mapping, the detailed surface features that influence an object's lit appearance can be captured in an image instead of actually increasing the object's geometric complexity. A bump-mapped scene has more geometry and surface richness than is actually present.

2.2 Displacement mapping

Although standard bump mapping offers a relatively inexpensive way to add surface detail, there are several downsides to this technique. Bump mapping approaches lack the ability to represent view-dependent unevenness of detailed surfaces, and therefore fail to represent motion parallax—the apparent displacement of the object due to viewpoint change, and today's some graphics hardware can compute it in real-time[12]. In recent years, new approaches for simulating displacement on surfaces have been introduced.

Displacement mapping was first mentioned by Cook[13] as a technique for adding surface detail to objects in a similar manner to texture mapping. It is

a powerful technique for adding detail to three dimensional objects and scenes. Although it can modify the silhouette and realize parallax effects, it addresses the issues above by actually modifying the underlying surface geometry. So it needs many polygons to generate a transformation through height map data. It is a kind of modeling technique that is unsuitable for real-time rendering.

To apply a detail image as a displacement map on a subdivision surface, the geometric elements are mapped the image domain. There are several mapping techniques performing such a task, the simplest planar mapping technique is adopted, which is a function

$$F : R^3 \to R^2$$

defined as follows.

$$F(v_i) = N \cdot p_i$$

N is a 3×3 matrix project the vertex p_i to the image plane. The typical choices of N are as follows.

$$N_{xy} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, N_{xz} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}, N_{yz} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Strictly speaking the function F does produce a vector in R^3. The image plane coincident to one of the three principle plane (xy, xz, yz), in which the projected vertices reside. The projection of vertices in R^2 will be normalize to $e(1,1)$ square domain, since detail image is aligned with the square domain.

The designer is free to choose in which principle plane the image should reside. The region of interest is also interactively selected by the designer. Therefore, appropriate image pixel coordinates can be interactively assigned to each triangle.The generation of the subdivision surface is achieved by iteratively applying a set of refinement rules, each iteration new vertices are inserted to the surface.

2.3 Parallax mapping

Parallax mapping is implemented by displacing the pixel coordinates at a point on the rendered polygon by a function of the view angle in tangent space and the value of the height map at that point. At steeper view-angles, the pixel coordinates are displaced more, giving the illusion of depth due to parallax effects as the view changes. Without changing the geometry property of the object surface, the image 3D illusion of object surface can be drawn by the translation of the pixel coordinates along view direction[14].Now, assuming that a model with geometrical details is on the upper part of the object. Its section can be described with a curve. The observation vector is $\vec{v}$, if the object surface is observed along the observation vector $\boldsymbol{V = (V_x, V_y, V_z)}$, the pixel coordinates (u,v) would be seen on the detail surface, while the point corresponding to the practical geometric detail model is (u',v'). To get the correct pixel coordinates, it is needed to shift the coordinates of each pixel of the image a certain distance along view direction in order to realize parallax effect when the viewpoint shifts on the object surface. That is just shifting point (u,v) to (u',v') along the view direction.

$$(u',v') = (u,v) + offset$$

(u,v) is the actual position of the viewpoint, and (u',v'), the corrected pixel coordinate of the actual pixel coordinate, is corresponding to the geometric detail model of the viewpoint, and *offset* is the offset of pixel coordinate. Retrieve some images with the corrected coordinates, and the image will show the 3D parallax effect when the sight moves relative to the uneven surface.

3 Detail Rendering Based on Pixel Offset

3.1 Pixel offset calculation

To calculate the pixel offset *offset* of point P_i on the current viewpoint, the first step is to normalize the eye vector V. Height value of the image pixel coordinate (u,v) can be read out from the input height map, and then assume that the height value of point (u,v) is set as h_i. As the offset along the view direction, so offset direction is determined by the sight direction, and the size of the offset is determined by the height value corresponding to (u,v), so the pixel offset of point P_i is defined as.

$$offset = h_i \cdot V_{x,y} \Big/ V_z$$

Where V_z is the projection of the eye vector V on to the axis n on the cutting plane space n,t,b, namely the n-component of V, and vector $V_{x,y}$ is the composite vector of the t-component and the b-component of the eye vector V, as shown in Fig.3.1.

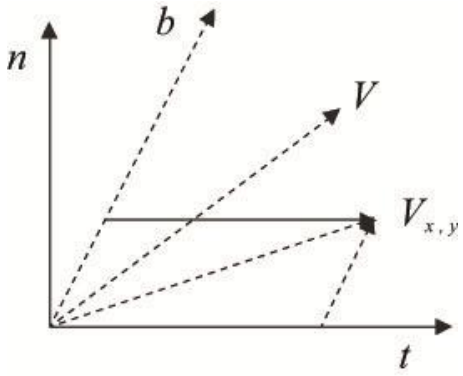


Figure. 3.1. Eye vector composition

And then get the new image pixel coordinate (u',v') as:

$$(u',v') = (u,v) + h_i \cdot V_{x,y} / V_z$$

Pixel offset calculation above is processed under the premise of assuming that point (u',v') and point (u,v) have the same height value, but this case is not usually seen in reality. When the object surface height values are not completely approximately equal, image pixel coordinate offset from the calculation will have a little difference with the actual situation, which will lead to a distortion of the details of the rendered object. Aiming at the deficiency of parallax mapping, some improvement method are proposed: as to the local computer common principle, when the distance between two points on the surface of an object were close, the height values of the two in the height map would not be too different with each other, so the heights can be considered approximately equal if limited in a certain distance range. Therefore, for the calculation of the image pixel coordinate offset, if *offset* is limited in a certain range, the height values of the points in this range would be approximately equal.

To calculate the appropriate limits, the first step is to halve the value of *offset* , namely.

$$offset = \frac{1}{2} h_i \cdot V_{x,y} / V_z$$

Although the drawing effect has been improved with this method, it is still unsatisfactory. Moreover, multiplying $1/2$ also increases a certain amount of calculation. Then the expression was simplified as follows:

$$offset = h_i \cdot V_{x,y}$$

However, when the viewing angle is sharp, the offset distance is small, which means that the heights of point (u',v') and point (u,v) are very close. When the viewing angle becomes very large, the pixel offset will tend to infinity, which will lead to that the offset is bigger than the actual offset value, then the final detail rendering results will not be correct.

3.2 Pixel offset translation

Incorrect pixel offset will lead to that the 3D effect obtained dose not meet the actual geometric detail model, and could even cause sliding phenomenon. In order to get the correct pixel coordinate offset in different viewing angles, as shown in Fig.3.2, the method of pixel offset translation is proposed.

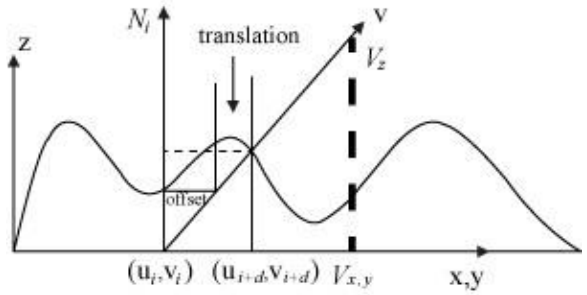


Figure.3.2 Pixel offset translation

In the graph V is the eye vector, and (u_i,v_i) is the actual image pixel coordinate, and N_i is the normal vector of the smooth surface mapping the detailed sense, and (u_{i+d},v_{i+d}) is a new pixel coordinates after translation. When the surface heights of the actual geometric detail model are not equal, in Fig.3.2 the distance between normal vector N_i and the intersection point of the eye vector V and the geometric detail model is the correct offset.

3.2.1 Mapping between pixel coordinate and height value

To get a true offset translation, mapping relationship should be established first, which is between pixel coordinates and the height values. Assume that the sizes of the image map and the height map are $M \times N$, both stored in chain tables. The data structure is as Fig.3.3.

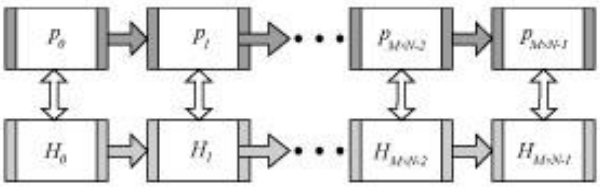


Figure.3.3 Data structure of mapping

Where, $P_0, P_1 \cdots P_{M\times N-2}, P_{M\times N-1}$ is the coordinates chain table,

$$P_i = (u_i, v_i), i = 0,1,\cdots M \times N - 1$$

and $H_0, H_1 \cdots H_{M\times N-2}, H_{M\times N-1}$ is the height chain table,where $H_i = h_i \quad i = 0,1,\cdots M \times N - 1$,and the mapping function is f , which defines the mapping relationship of the pixel coordinate P_i and the height value H_i ,

$$f(u_i, v_i) = h_i \quad i = 0,1,\cdots M \times N - 1$$

Each pixel coordinate is corresponding to a height value, as shown in Fig.3.4, which shows the mapping function of coordinates and the height values, and new pixel coordinates

$$P_i = (u_i, v_i), i = 0,1,\cdots M \times N - 1$$

are corresponding to the height values

$$H_i = h_i \quad i = 0,1,\cdots M \times N - 1 \quad .$$

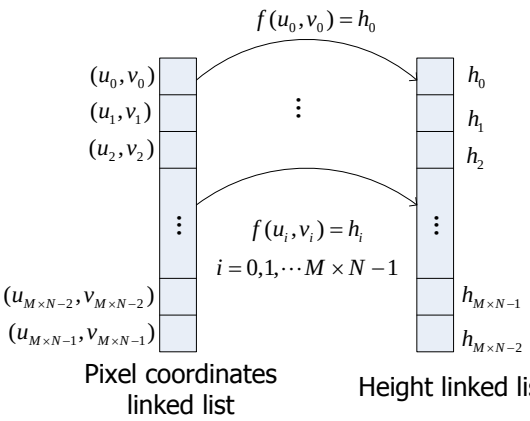


Figure3.4. Pixel coordinates and height linked list

3.2.2 Translation value calculation

Before calculating translation, f mapping function of the pixel coordinates and the height values should be defined first. Now assuming that the eye vector is V and its angle with the object surface is θ , as shown in Fig.3.5, and the cotangent value of θ is equal to the ratio of Vector $V_{x,y}$ and Vector $V_{x,y}$, which are the component vectors of the eye vector on the plane x, y and the axis z . The expression is as follows.

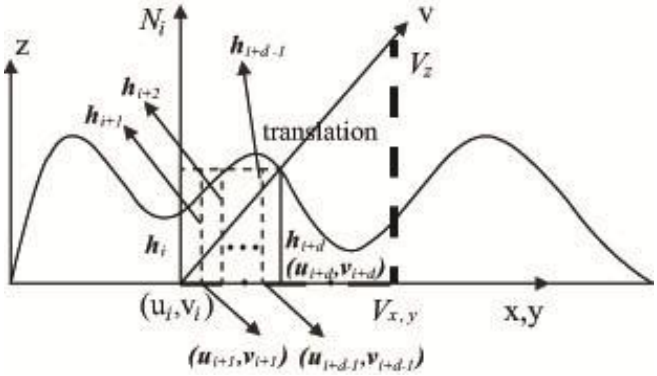


Figure.3.5 Translation value calculation

$$\cot\theta = \frac{V_{x,y}}{V_z}$$

On the current viewpoint, the ratio of the correct coordinate offset and the corresponding height value after the offset is

$$\frac{offset + translation}{H_{i+d}} = \cot\theta \quad .$$

Thus, the relationship between the new pixel coordinate offset and the current viewpoint coordinates is

$$\frac{offset + translation}{H_i} = \frac{V_{x,y}}{V_z} \quad .$$

Therefore, the current pixel coordinates T_i moves along the eye vector pixel to pixel, and according to the function f , the height H_{i+1} of the new pixel coordinate P_{i+1} can be obtained. Move the coordinate pixel to pixel, and then the ratio of the number of the pixels of the coordinate offset and the height of the image coordinates can be obtained, When the equation

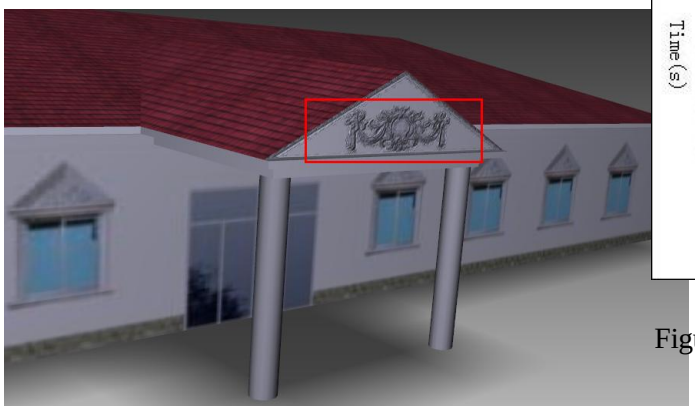
$$\frac{i+d}{H_i} = \frac{V_{x,y}}{V_z}$$

is true, the ordinate will no longer move, and then $d = |offset + translation|$ is the correct pixel offset of detail rendering.

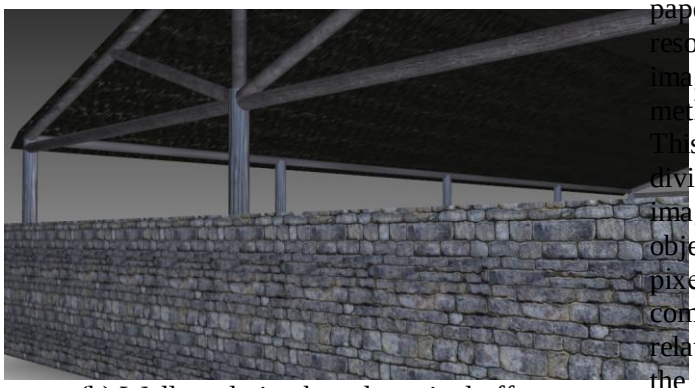
4 Experimental Results

In order to verify the realistic result with this detail rendering method based on pixel offset, in this paper, some flower relief of villages building and wall of civil house are rendering uses CG language, on an ordinary PC, transfers data from

Pixel Shader to Fragment Shader, and three-dimensional realistic rendering of detail objects in the villages are realized. Experimental results as Fig. 4.1.



(a) Flower relief rendering based on pixel offset

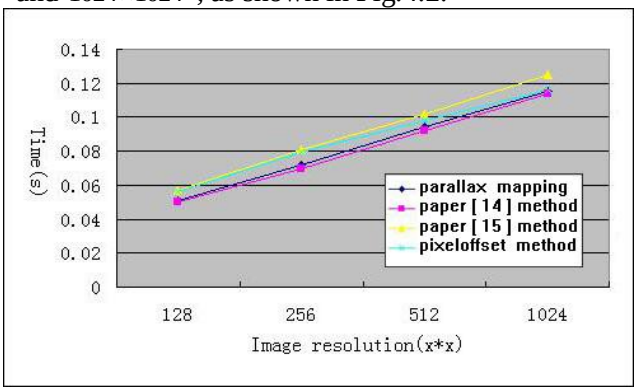


(b) Wall rendering based on pixel offset

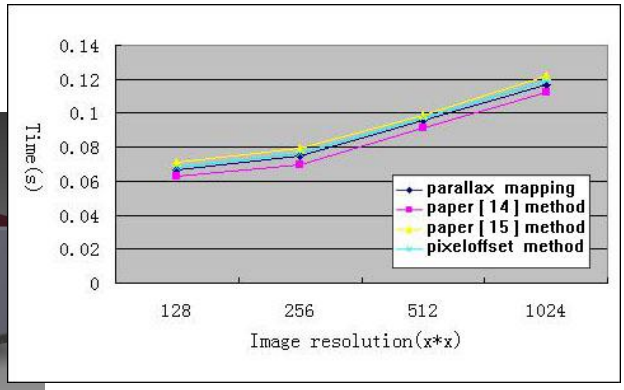
Figure4.1Rendering results based on pixel offset

Can be seen from Fig.4.1, there will be a kind of realistic effect created as the viewpoint moves. And some details will show different effects as the image pixel coordinate offsets are differe.

For the complexity of the algorithm, compare the time costs of different methods rendering brick surface detail image and wrought detail image in different resolution of 128×128 , 256×256 , 512×512 and 1024×1024 , as shown in Fig.4.2.



(a) Brick surface rendering time



(b) Wrought rendering time

Figure.4.2 Rendering time contrastation of different methods

Division in the detail rendering is omitted in paper[14], and it costs the shortest time in the same resolution. Yet, to ensure the correctness of the image pixel coordinate offset, a dot product method[15] is used, so it costs the longest time. This paper concludes a method, which consists of a division and a pixel translation, to obtain the correct image coordinate offset and the realistic detail object, by shifting the image coordinates pixel to pixel. The shifting won't increase the algorithm complexity, so the algorithm complexity only relates to the division. Therefore, the calculation of the pixel coordinates by per-pixel shifting won't increase the whole complexity of algorithm. Meanwhile, the object surface details are formed by some tiny ups and downs changes, and the step of the per-pixel shifting is short, so the improved method of parallax mapping with pixel to pixel shift for image coordinates does not decrease the efficiency of image detail drawing.[16].

6 Conclusion

A lot of detail objects in the virtual village rendering, bump mapping, normal mapping and parallax mapping can produce 3D effect，but offset will cause detail information sliding distortions. When the viewpoint moves to make the angle with plane vary small, the situation will be very sharp, and the effect will be unacceptable. Therefore calculation pixel coordinate offset is a very important problem. In this paper, a one-one mapping function of image coordinate and height is defined, and a detail object rendering method based on pixel offset is proposed. Moving the pixel coordinates to the eye vector in pixels, when the distance and the current viewpoint coordinates meet

the given equation, the right coordinate offset is obtained. This method can get the true coordinate offset, and correct the distortion deformation phenomenon, meantime, pixel shifting won't increase the algorithm complexity. Therefore, it enhances the realistic effect of detail object rendering with no increase in algorithm complexity.

Acknowledgements

This work was supported by the by the National Natural Science Foundation of China (60871042, 61102126), Beijing Municipal Natural Science Foundation (54122034), National Technology R&D Program(2010ZX01045-001-004,2011BAD21B02).

References

[1] H. X. Ren, Y. Y. Yin, Y. C. Jin, *Realistic rendering of large-scale ocean wave scene. Journal of Computer-aided Design & Computer Graphics,* **20**, 1617-1622 (2008).

[2] G.Francois, S.Pattanaik, K.Bouatouch, et al. *Subsurface texture mapping. Computer graphics and applications,* **28**, 34-42(2008).

[3] T. Kaneko, T. Takahei, M. Inami, et al. *Detailed shape representation with parallax mapping. Proceedings of the 11th International Conference on Artificial Reality and Telexistence, Tokyo, Japan,* 205-208(2001).

[4] P. Matt, F.Randima. *GPU Gems 2: Programming techniques for high performance graphics and general-purpose computation. Boston: Addison-Wesley,* 123-136,(2005).

[5] N.Tatarchuk. *Dynamic Parallax occlusion mapping with approximate soft shadows. Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, Redwood* , 63-69 (2006).

[6] F.Policarpo, M.M.Oliveira,J.L.D.Comba.*Real-Time relief mapping on arbitrary polygonal surfaces. Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, Washington,* 155-162 (2005).

[7] M.Morgan, M.Max. *Steep parallax mapping. Computer Graphics Proceedings, Annual Conference Series, ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games, Washington,* Poster (2005).

[8] M.J.Kilgard. *A Practical and Robust Bump-mapping Technique for Todays GPUs Game Developers Conference, Advanced OpenGL Game Development.* (2000).

[9] M. Peercy, A. Airey, B.Cabral. *Efficient Bump Mapping Hardware. Proceedings of SIGGRAPH,* 303-306 (1997).

[10] P. S. Heckbert. *Survey of Texture Mapping. IEEE Computer Graphics and Applications,* 56-67 (1986).

[11] M. Bóo et al. Hardware Support for Adaptive Subdivision Surface Rendering. Proceedings of Siggraph/Eurographics Workshop on Graphics Hardware, ACM Press, 33-40 (2001).

[12] A. Schilling. *Towards Real-Time Photorealistic Rendering: Challenges and Solutions. Proceedings of Siggraph/Eurographics Workshop on Graphics Hardware, ACM Press,* 7-15 (1997).

[13] R. L. Cook. *Shade Trees. Proceedings of Siggraph, ACM Press,* 223-231 (1984).

[14] T. Welsh. *Parallax mapping with offset limiting: a per-pixel approximation of uneven surfaces. Technical report, Infuscate Corporation,* (2004).

[15] R. H. Gao, B. C. Yin, D. H. Kong, et al. *An improved method of parallax mapping. Proceedings of IEEE 8th International Conference on Computer and Information Technology, Sydney,* **7**,30-34 (2008).

Chunjiang Zhao is a professor at National Engineering Research Center for Information Technology in Agriculture. He received the Ph.D. degree in agronomy from China Agricultural University, Beijing, China, in 1991. He is currently a Senior Scientist with and the Director of the National Engineering Research Center for Information Technology in Agriculture, Beijing. He is also a Member of the Science and Technology Commission, Ministry of Agriculture, China. His research interests include precision farming, intelligent information technology in agriculture, and crop decision support system.Dr.Zhao received the Excellent Scientist Award by Ministry of Science and Technology of China in 2001.

Huarui Wu is a professor at National Engineering Research Center for Information Technology in Agriculture . He is interested in studying Artificial Intelligence. In recent years, he has participated in 18 national and provincial key scientific research projects, and published over 20 academic papers. He got the first prize of Beijing Science and Technology in 2005, and third prize of agricultural technology promotion in 2003.

Ronghua Gao is an Assistant Researcher at National Engineering Research Center for Information Technology in Agriculture. She is interested in developing simple and practical algorithms for computer graphics. She has a PhD in computer science from the Beijing Technology University.