Real-time rendering algorithm based on a hybrid rendering scheme*

ZHENG Wenting (郑文庭), BAO Hujun (鲍虎军), PENG Qunsheng (彭群生)

(State Key Laboratory of CAD&CG, Zhejiang University, Hangzhou, 310027, China)

and SUN Hanqiu(孙汉秋)

(The Chinese University of HongKong, HongKong, China)

Received Jannary 25, 1999; revised April 23, 1999

Abstract A real-time rendering algorithm based on a hybrid rendering scheme is presented. The key frames of the virtual scene are generated with a geometry-based approach, then a non-linear transformation to all visible points of each frame near the current viewpoint is performed to reveal their coplanar characteristics and a hierarchical polygon approximation of the local scene is set up. Any intermediate image between the two key frames could be obtained by reprojecting these polygons onto the view plane related to the current viewpoint. The gaps that might appear on the intermediate image are filled through bi-directional backward image warping. Experimental results show that our approach is insensitive to the complexity of the scene and has a satisfactory performance.

Keywords: virtual reality, image-based rendering, level of detail, texture mapping.

In recent years, many researches have been focused on developing real-time rendering techniques for the complex scenes. Three kinds of effective methods were developed, i.e. the visibility preprocessing[1—5], level of detail algorithm (LOD)[6—7] and the image-based rendering (IBR)[8—12]. Since image-based rendering is a technique whose performance is independent of the complexity of the scene, it can achieve real-time rendering of the complex scene on a low-end computer. Chen and Williams[8] proposed a view interpolation scheme to get smooth transition of two adjacent sample images. As the algorithm performs a linear interpolation between the projection locations of the same visible points on the two images, it cannot simulate the generic perspective transformation precisely. Moreover, gaps appear frequently on the intermediate image due to local image expanding and the changes of visibility of the scene. Other researchers adopted some simple geometry such as quadrangle, polygon meshes to approximate the complex geometry to generate the picture of the current scene[13,14,16—18]. As these simple geometric objects keep the same depth order as that of the visible objects in the current scene, this approach generates correct occlusion among objects. Apparently the simplified geometry approximation of the scene is view dependent, it must be updated as soon as the resultant image deviates from the accurate view by a given tolerance.

A real-time rendering algorithm based on a hybrid rendering scheme is presented in this paper to take the advantages of both the geometry-based and image-based approaches. The proposed method

* Project supported jointly by Huo Yingdong Young Teacher Foundation and the National Natural Science Foundation of China (Grant No. 69673027).

generates the key frames of the virtual scene first with a geometry-based approach, then establishes real-time smooth image transition between two adjacent key frames by backward image warping.

1 Reconstruction of the local 3D scene

To reflect the perspective transformation, the 3D coordinates of the visible point through each pixel on the key frame must be recovered. A sampling of the visible portion of the local scene was established without consideration of the corresponding adjacent key frames.

1.1 Geometric reconstruction of local scene

Let $I_{M\times N}$ be a given image of resolution $M \times N$, $z_{M\times N}$ an associated depth buffer, $\boldsymbol{EUVW}$ the coordinate system of the camera producing image I, where $\boldsymbol{E}$ is the viewpoint (fig.1), $\boldsymbol{U}(U_x, U_y, U_z)$、$\boldsymbol{V}(V_x, V_y, V_z)$ and $\boldsymbol{W}(W_x, W_y, W_z)$ are the three unit-vectors of the axes in the world coordinate. For every pixel P, the corresponding 3D visible point V can be obtained by $V = \boldsymbol{H}_E P$. Note that $V(x_p, y_p, z_p, 1)$ and $P(x, y, 1, 1)$ are in homogeneous coordinates. $\boldsymbol{H}_E$ is the 4×4 perspective matrix at the viewpoint $\boldsymbol{E}$, namely,

$$\boldsymbol{H}_E = \begin{bmatrix} zU_x & 0 & 0 & E_x + zyV_x + zW_x \\ 0 & zV_y & 0 & E_t + zxU_y + zW_y \\ 0 & 0 & z & E_z + zxU_z + zyV_z \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{1}$$

where $x = \left[\frac{i}{M} - \frac{1}{2}\right] \mathrm{tg}\frac{\alpha}{2}$ and $y = \left[-\frac{j}{M} + \frac{N}{2M}\right] \mathrm{tg}\frac{\alpha}{2}$, α is the field of view of the camera, (i,j) the screen coordinate of pixel P, and $z = z(i,j)$ the depth value stored in the depth buffer. Let $P(i,j)$, $P(i+1,j)$, $P(i+1,j+1)$ and $P(i,j+1)$ be the four adjacent pixels on image I, and $\{V(i,j)$, $V(i+1,j)$, $V(i+1,j+1)$ and $V(i,j+1)$ the corresponding visible points which form a space quadrangle. All space quadrangles of adjacent pixels on the image form a mesh which can be looked as an approximation of the local scene, and image I is just the projection of this mesh onto the current view plane. When a new viewpoint is located nearby, we can simply render this space quadrangle mesh, and map the color of image I to the projections of respective quadrangles. Obviously, the above mesh consists $M \times N$ quadrangles, and it is difficult for ordinary computers to perform image warping with so many quadrangles. An idea is to merge the quadrangles that come from the same polygon. The details inside the polygons can be recovered by texture mapping.

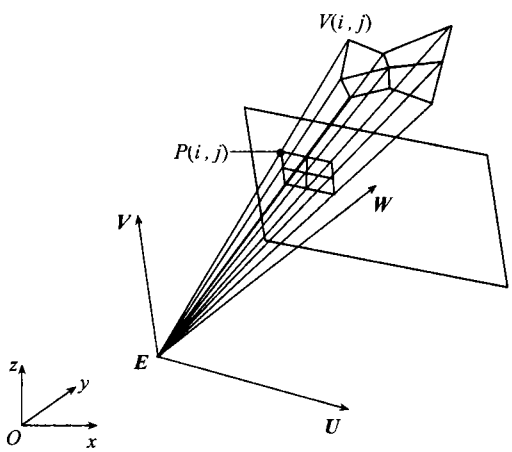


Fig. 1. 3D mesh reconstruction of the local scene based on the depth information.

1.2 LOD model of the local scene

If image I is synthesized by computer with

hardware Z-buffer algorithm, then a depth buffer associated with image I is derived automatically. The hardware Z-buffer algorithm performs a perspective transformation to all objects in the scene. Let (i, j) be a pixel on the image plane, $z(i,j)$ the depth of the visible point through pixel (i,j) in the eye coordinate system, then the depth information $z'(i,j)$ stored in the depth buffer can be expressed as:

$$z'(i,j) = 2^{24} \frac{z_{far}}{z_{far} - z_{near}} \left[1 - \frac{z_{near}}{z(i,j)}\right], \tag{2}$$

here, $z = z_{near}$ and $z = z_{far}$ are the near clip plane and far clip plane of the current view frustum respectively. It is worth mentioning that, all points on the same plane will remain coplanar after perspective transformation. Therefore, if the visible points of adjacent pixels of (i,j) come from the same planar polygon, the depth value $z'(i,j)$ will satisfy $\Delta^2 z'(i,j) = 0$, here Δ^2 is the second order difference operator and

$$\Delta^2 z'(i,j) = 4z'(i,j) - z'(i,j-1) - z'(i,j+1) - z'(i-1,j) - z'(i+1,j). \tag{3}$$

Thus, a new buffer can be established that contains a lot of elements of zero values, and is called Δ buffer. Obviously, the connected region made up by the zero elements on the Δ buffer is a projection of a planar polygon. We can then retrieve the 3D information of the visible polygons of the local scene based on the Δ buffer and the z-buffer.

For images not rendered by hardware z-buffer algorithm, e.g. those produced by range camera or by other algorithm, the depth value of each visible point on the images should undergo a non-linear transformation as described by equation (2).

To establish a hierarchical visible polygon approximation of the local scene, a binary subdivision of the Δ buffer is performed to partition the region. The current region of the buffer is recursively cut into two sub-regions in x and y directions alternatively. The cut position is dynamically determined as follows. First, the current region is scanned from one extreme along the cut direction to the other until the $\Delta^2 z'(i,j)$ value of an element on the current scan-line is greater than a given threshold. If the scanned area is large enough (e. g. 1/4 of the whole region), then the last scan-line is selected as the cut position, otherwise a similar operation is implemented from the other extreme. If the cut position cannot be determined through the above two steps, the middle scan-line is used to partition the region. This procedure is recursively implemented until either the maximum value of $\Delta^2 z'(i,j)$ in the current region is less than the given threshold or the region cannot be divided any further. The 3D information of each spatial polygon can then be retrieved from the four vertices of the respective rectangle on the Δ buffer. Plate I-A illustrates the results of the reconstructed polygonal mesh of the local scene based on a depth image at different thresholds. Apparently, by specifying different thresholds, we get a LOD model of the current local scene.

It should be pointed out that the spatial quadrangles generated by the above algorithm may be discontinuous topologically in the 3D space, although their projections on the key frame do. To avoid gaps which may appear on the intermediate image during image warping, the boundary of the projection of each element of the mesh is expanded by half a pixel. While this method reduces the texture distortion in the following warping process, it might yield new gaps on other view planes.

2 Rendering

In this section, we will discuss how to quickly synthesize the intermediate image between two key frames. In contrast with conventional methods, we perform backward image warping according to the position of the intermediate viewpoint.

2.1 Backward warping

Let E' be an intermediate viewpoint. As it lies near the current viewpoint, the intermediate view keeps a great coherence with the current view. The intermediate view can be generated by re-projecting the visible polygon approximation of the local scene onto the intermediate image plane. To keep the coherence, image I in fig.2 is attached to the quadrangle mesh as texture, and each spatial quadrangle is rendered by hardware texture mapping. So in principle, the image on the screen needs to be projected onto the spatial quadrangles, which is in fact a process of establishing the textures. However, the inverse projection of textures is not a linear mapping, it cannot be accomplished with the current texture mapping hardware. Rather, in our method, the spatial quadrangles are first projected onto the intermediate image plane and a number of planar quadrangles are obtained, then each projected quadrangle is associated with a corresponding region on the key frame which is taken as a texture. The vertices of the projected quadrangle on the intermediate image plane can be obtained by

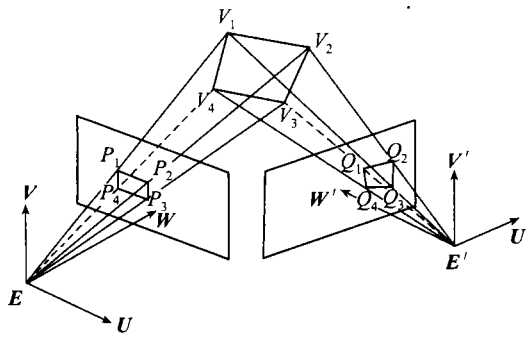


Fig. 2. Backward warping procedure.

$$\boldsymbol{Q}_i = \boldsymbol{H}_{E'}^{-1} V_i = \boldsymbol{H}_{E'}^{-1} \boldsymbol{H}_E P_i, \qquad (i = 1,2,3,4),$$

where $\boldsymbol{Q}_i$ is a vertex of the projected quadrangle, V_i the corresponding vertex of the spatial quadrangle.

In fact, a backward mapping is performed in the above procedure. The projected area of a spatial quadrangle on the new image plane can be looked as a warping of rectangle image on the original image. As shown in fig.2, the pixel region $P_1P_2P_3P_4$ on image I corresponds to the spatial quadrangle $V_1V_2V_3V_4$, the quadrangle $Q_1Q_2Q_3Q_4$ on the new image plane is the projection of spatial quadrangle $V_1V_2V_3V_4$, and $P_1P_2P_3P_4$ provides the texture of rendering the planar quadrangle $Q_1Q_2Q_3Q_4$.

When the area of the retrieved spatial polygon is large and near the new viewpoint, the above backward warping process may cause serious distortion due to the great depth variety of the visible points on the spatial quadrangle. To keep the rendering quality, such kind of quadrangles should be subdivided adaptively. At present, however, a uniform subdivision is performed in our algorithm.

2.2 Gap filling

Although the backward warping technique removes a great number of gaps due to local image ex-

panding, it cannot eliminate the gaps caused by visibility changes of the local scene. The essential problem is that no visible information through these gaps is provided by original image I. Various geometry-based approaches can be employed to recover the visible information within these gaps[3,8], but they failed to meet both the requirements of real-time rendering and image quality. In our algorithm, the following strategies are used to solve this problem.

(i) Using two adjacent key frames to fill the gaps. When two spatial quadrangle meshes derived from adjacent key frames are projected onto the same intermediate image plane, the two projections are mostly complemental for gap filling. One projection is chosen as the foreground of the intermediate image according to the distance from the intermediate viewpoint to the viewpoints of the key frames. The projection of the mesh with less distance is regarded as the foreground while the other as background. This method can reduce the alias of rendering to some extent.

(ii) For the remaining gaps whose contents are visible to the intermediate viewpoint, but invisible to the viewpoints of both the two key frames, we make use of the last frame image to fill them. This simple technique takes advantage of the image coherence and avoids image flashing.

The experimental results show that the above gap filling method is fast and efficient. The quality of the intermediate image is much better than that generated by view interpolation. The process is illustrated by Plate I-B. Note that this method can also be applied to the real world images. To do this, we need to adopt computer vision techniques to recover the 3D positions of viewpoints and the depth information of each visible point.

3 Implementations and discussions

We implemented the proposed algorithm on a SGI Octane workstation. The key frame depth images are generated by Iris Performer software. Scene models are generated with Multigen software. Two scenes are adopted for testing the performance of the method, one is an outdoor scene, the other is an indoor scene. The resolution of all test images is 512×512 pixels.

The outdoor scene (Plate I-A(a)) consists of more than 100,000 polygons. The excessive details are represented by textures. Plate I-B(a) and B(b) are the two key frame images, B(c) and B(d) are generated by backward warping key frame B(a) and B(b) onto the same image plane with respect to the intermediate viewpoint E_M, $E_M = (E_a + E_b)/2$, (E_a, E_b are the viewpoints corresponding to B(a) and B(b)). The red regions in the picture represent the gaps. B(e) is the compositional result of B(c) overlapping B(d). As B(c) and B(d) are complementary, the gaps in B(e) become much less. The remaining gaps in B(e) are filled by adopting the previous frame i.e. B(a), as the background. B(f) is the final result of this process. It takes 0.048 seconds to accomplish the whole procedure. The cost for the key frame generation is not accounted here.

It can be observed that the rendering time of an intermediate image is almost independent of the complexity of the scene except for the generation of the key frames which is relevant to the resolution of the key frame image. According to the performance of the hardware, a suitable key frame resolution and a Δ buffer region partition threshold can be determined to achieve a satisfactory refresh rate.

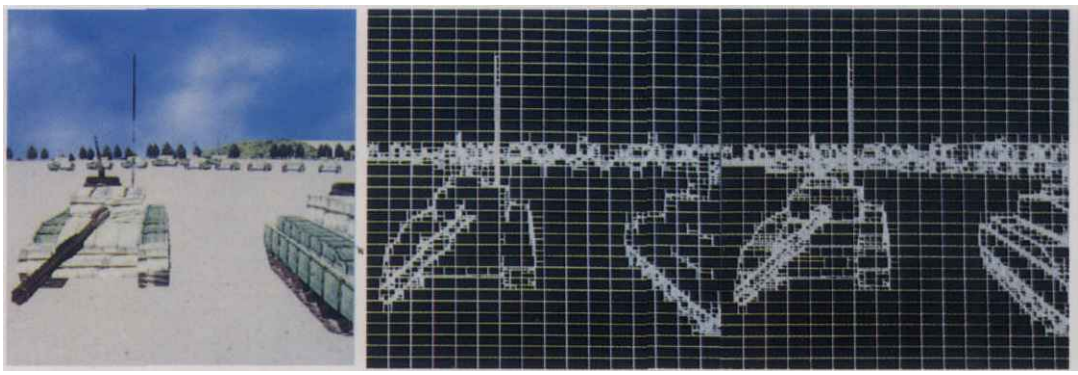
From the experiments, we found that objects near the viewpoint of the key frame may cause serious texture floating and alias on the intermediate images and our above gap-filling scheme cannot properly recover the information within the gaps. To solve this problem, one can remove the objects near the current viewpoint from the key frames, and then render these geometric objects directly to the intermediate picture.

4 Conclusions

In this paper, we propose a real-time rendering algorithm based on backward image warping. A nonlinear transformation and a second order difference operation are applied to the depth values of all visible points to reveal their coplanar characteristics. A hierarchical visible quadrangle approximation of the local scene is then established. The intermediate images can be generated by re-projecting the visible quadrangle mesh onto the intermediate image planes, and the gaps that may appear on the intermediate images are efficiently eliminated by bi-directional backward image warping. Experimental results show that our algorithm can achieve real time rendering with fairly good image quality. The rendering time requested by our approach depends on the image resolution but almost independent of the geometric complexity of the scene.

References

1 Greene, N., Kass, M., Miller, G., Hierarchical Z-buffer visibility. *Computer Graphics*, 1993, 27(2): 231.
2 Regan, M., Pose, R., Priority rendering with a virtual address recalculation pipeline. *Computer Graphics*, 1994, 28(3): 155.
3 Zhang, H. S., Manocha, D., Hudson. T. et al., Visibility culling using hierarchical occlusion maps. *Computer Graphics*, 1997, 31(3): 77.
4 Teller, S., Sequin, C., Visibility preprocessing for interactive walkthroughs. *Computer Graphics*, 1991, 25(4): 61.
5 Wang, Y. G., Bao, H. J., Peng, Q. S., Accelerated walkthroughs of virtual environments based on visibility preprocessing and simplification. *Computer Graphics Forum*, 1998, 17(3): 187.
6 Hoppe, H., Progressive meshes. *Computer Graphics*, 1996, 30(3): 99.
7 Luebke, D., Erikson, C., View-dependent simplification of arbitrary polygonal environments. *Computer Graphics*, 1997, 31(3): 199.
8 Chen, S. E., Williams, L., View interpolation for image synthesis. *Computer Graphics*, 1993, 27(2): 279.
9 Chen, S. E., QuickTime VR: an image-based approach to virtual environment navigation. *Computer Graphics*, 1995, 29(4): 29.
10 McMillan, L., Bishop, G., Plenoptic modeling: an image-based rendering system. *Computer Graphics*, 1995, 29(4): 39.
11 Levoy, M., Hanrahan, P., Light field rendering. *Computer Graphics*, 1996, 30(3): 31.
12 Gortler, S. J., Grzeszczuk, R., Szeliski, R. et al., Lumigraph. *Computer Graphics*, 1996, 30(3): 43.
13 Shade, J., Lischinski, D., Salesin, D. H. et al., Hierarchical image caching for accelerated walkthroughs of complex environments. *Computer Graphics*, 1996, 30(3): 75.
14 Sillion, F. X., Drettakis, G., Bodelet, B., Efficient imposter manipulation for real-time visualization of urban scenery. *Computer Graphics Forum*, 1997, 16(3): 207.
15 Torborg, J., Kajiya, J. T., Talisman: commodity real-time graphics for PC. *Computer Graphics*, 1996, 30(3): 353.
16 Mark W. R., McMillan L., Bishop G., Post rendering 3D warping, in: *proc. of Symposium on Interactive 3D Graphics*, 1997, 7.
17 Darsa, L., Silva, B. C., Varshney, A., Navigating static environments using image-space simplification and morphing, in: *proc. of Symposium on Interactive 3D Graphics*, 1997, 25.
18 Darsa, L., Costa, B., Varshney, A., Walkthroughs of complex environments using image-based simplification. *Computer & Grpahics*, 1998, 22(1): 55.



(a) (b) (c)

A

(a) (b) (c)

(d) (e) (f)

B

A. Partition the depth image. (a) The original image; (b) result of partitioning with threshold of 2^{15}; (c) result of partitioning with threshold of 2^{11}. B. Process for generating an intermediate image.