Distributed virtual reality system based on hybrid rendering*

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)

(Department of Computer Science and Engineering, the Chinese University of Hong Kong, Shatin, NT, Hong Kong, China)

Received March 5 , 1999; revised April 20, 1999

Abstract A distributed virtual reality system is presented based on a hybrid rendering scheme. This system first predicts the camera position of the next key frame in real-time according to the tendency of the current camera movement. The predicted key frame is then generated using a geometry-based algorithm. Finally, a real-time backward image warping algorithm is employed to create a smooth transition between the two adjacent key frames. Since the rendering procedure can be divided into two independent steps, this system can be efficiently implemented in distributed mode.

Keywords: virtual reality, texture mapping, distributed computation, parallel computation.

Walkthrough is a fundamental function provided by every virtual reality (VR) system. During walkthrough, the view of the scene corresponding to the current position and direction of the camera must be updated in real time. Thus rendering the scenes in real time is essential to a VR system.

In recent years, many real-time rendering techniques have been developed for complex scenes. These techniques can be classified into two categories. One is geometry-based approach and the other is image-based approach. While the emphasis of the former is to efficiently exploit the space and image coherence to reduce the scene complexity[1—3], the latter integrates the re-projecting techniques with hardware texture mapping[4—7], taking pre-sampled images as the rendering primitives. It is interesting to note that the merits of both kinds of techniques are complementary. For the geometry-based approach, it is easy to incorporate user interactions in the scenes during walkthrough and the quality of the resultant image is ensured. But it is hard for a geometry-based approach to meet the real-time rendering requirement on the low-end computers. The image-based technique, on the other hand, takes advantage of the coherence of the adjacent frames to avoid the complicated visibility computation. It generates the intermediate images either by view interpolation or by image warping. Due to its sampling features, the technique often causes serious aliasing effects. Moreover, since the technique needs to generate and store the sample images at preprocessing step, it can only be applied to the static scenes[8]. In order to extend the image-based rendering technique to the dynamic scenes, Raskar et al.[9] developed a virtual office system by which the depth information on the sample images is used to reconstruct the intermediate images. Since this system adopts the imperceptible structured

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

light to quickly recover the depth information, its application is limited to the scenes of narrow range.

To take full advantage of the hybrid rendering scheme[10], we developed an interactive desktop virtual reality system which is implemented in a distributed mode.

1 Framework

The image warping algorithm described in ref. [10] can achieve a smooth transition of two adjacent depth images in real-time. If the depth images of key frames on the walkthrough path can be generated interactively, we can conveniently walk through the scene in real time. This process is shown in figure 1.

To support the user interactions with the scene, the geometry-based rendering technique is adopted to generate the key frames, and the image-based rendering technique is adopted to generate the transition images. As the latter is insensitive to the complexity of the scenes, the hybrid rendering scheme can achieve a high refreshing rate. The user interactions can be realized using the geometric information.

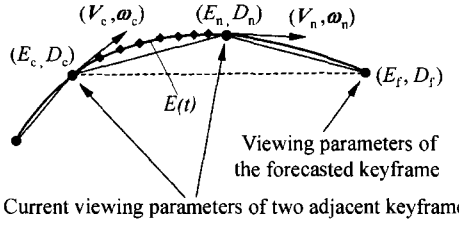

Fig. 1. Relationship between the transition images and the key frames. ● Camera positions of the keyframes; ◆ Camera positions of the intermediate images.

Since the position and viewing direction of the camera are interactively controlled by the user during walkthroughs, it is impossible to generate the key frames at preprocessing step. Our strategy is to use a predictor to forecast the next key frame according to the user interaction in real time. The key frame is then generated based on the geometric models and conformed to the movement of the camera. Fig.2 is the flowchart of our rendering scheme. Note that the implementation order of each step should be followed strictly, otherwise, it may lead to incorrect results.

2 The strategy of real-time distributed rendering

From fig.2, it can be seen that the generation of the next key frame is synchronous with the process of image warping from the last key frame to the current one. To avoid any unnatural pause, the former must be accomplished before the latter, only when the system can step into the next cycle, i.e. warping from the current key frame to the next key frame. This rendering scheme can be naturally implemented in parallel or distributed mode.

Unlike the traditional distributed and parallel algorithms[11], this rendering algorithm must be executed in a specified order. We call the generation of the next key frame the precursor process, and the image warping the successor process. In order to synchronize the two processes, the precursor process must wait for the accomplishment of the successor process in each cycle. These two processes are indepen-

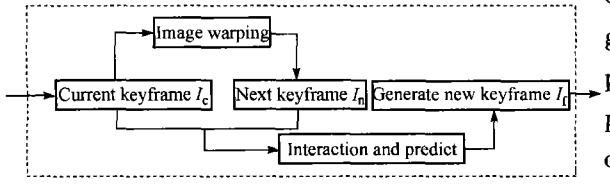

Fig. 2. Flowchart of the rendering scheme.

dently implemented in separate spaces, and their communication is realized by sharing the memory. The distributed mode is described here and the parallel mode can be implemented in a similar manner.

The distributed rendering strategy uses more than two computers to implement the rendering process[12]. In our system, the geometry-based rendering process and the image-based rendering process are performed on multiple computers. The tasks of the master computer include forecasting the viewing parameters of the next key frame, sending the rendering information to each slave computer, receiving the key frames from the slaves and generating the transition images between the key frames. The tasks of the slave computer are to receive the rendering information to generate the key frames and send them to the master. The user interaction interface is provided by the master. If the master is powerful enough to simultaneously warp multiple key frames in real time, the multiple slaves are preferred and the resultant intermediate image will be more accurate.

Figure 3 shows the flowchart of our distributed rendering strategy in which only one slave is used for the clarity. The rendering information means the necessary information for generating an image using geometry-based rendering algorithms, including the camera information and the geometry information of the scene.

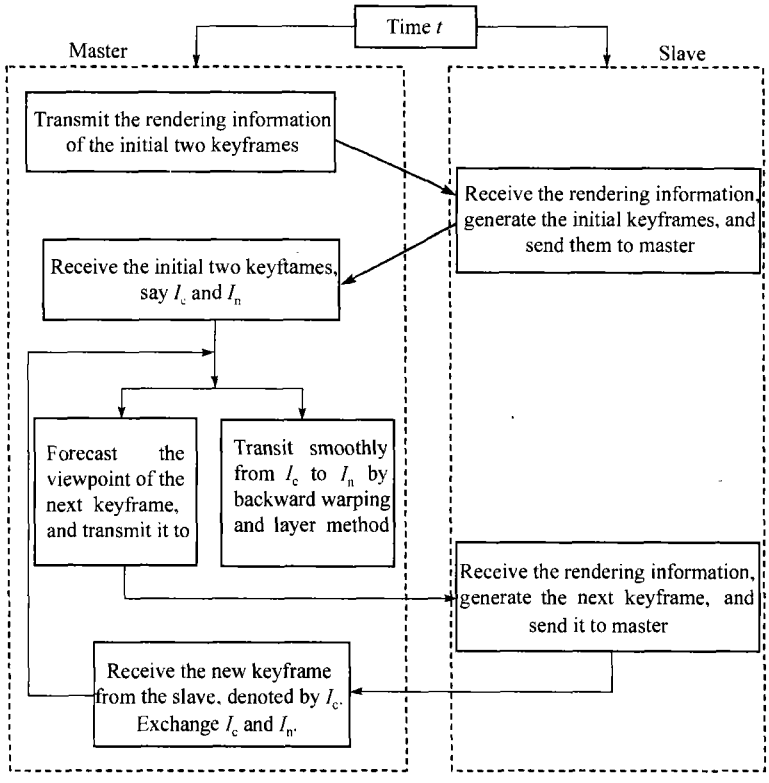

Fig. 3. Flowchart of the distributed rendering strategy.

3 System Controls

Note that the transition images are generated through backward image warping of the key frames, the camera movements can be specified according to the respective camera position and orientation of the previous key frames. To produce a natural motion, the number of the transition images between two key frames is predetermined during the initialization phase. As a result, the distance between adjacent key frames determines the speed of the camera.

3.1 Viewing controls

Three mouse buttons were used to control camera movement. The camera coordinate system is defined by (E, D), where $E(x, y, z)$ is its origin (viewpoint) and $D(p, r, h)$ is its orientation (three Euler angles) with respect to the world coordinate system. The camera coordinate system is initialized as the world coordinate system, and its current position can be conveniently determined by rotating the initial camera coordinate system around z, x and y axes of the world coordinate system by angle h, p and r in turns, and then translating it to E.

As discussed in the previous section, in each rendering cycle, we need to perform image warping between the current key frame I_c and the next key frame I_n to generate a set of transition images and to predict the new key frame I_f. Let the local camera systems of the three key frames be (E_c, D_c), (E_n, D_n) and (E_f, D_f) respectively, and (V_c, ω_c) the velocity vector of the current camera (E_c, D_c), then (E_f, D_f) can be determined by

$$(E_f, D_f) = (E_n, D_n) + (\Delta E_n, \Delta D_n).$$

And,

$$s_n = \| V_c \| + a,$$

$$\Delta E_n = (s_n \sin h_n \cos p_n, s_n \cos h_n \cos p_n, s_n \sin p_n),$$

$$\Delta D_n = (k(m_y - c_y)/R_y, 0, k(m_x - c_x)/R_x),$$

where (m_x, m_y) is the screen coordinate of the mouse, s_n the displacement of viewpoint, and (c_x, c_y) the center of the image; R_x and R_y are the resolutions of the image in vertical and horizontal direction, a and k are the user-specified constants related to the status and the distance of the mouse movement, which control the sensitivity of the mouse regarding the camera movement.

Once the slave computer receives parameters (E_f, D_f), it begins to render the new key frame immediately, at the same time the master computer implements backward image warping from (E_c, D_c) to (E_n, D_n). The speed vector (V_n, ω_n) at (E_n, D_n) can be estimated as

$$V_n = s_n \frac{E_f - E_c}{\| E_f - E_c \|}, \quad \omega_n = \| \Delta D_n \| \frac{D_f - D_c}{\| D_f - D_c \|}.$$

To achieve smooth transition, a kinematic spline is adopted to determine the viewpoints and viewing directions of the intermediate images, which are the interpolations between (E_c, D_c) and (E_n, D_n) with the end conditions (V_c, ω_c) and (V_n, ω_n) (fig.1). According to the Newton's kinematics equation, the interpolation kinematics spline can be described in the component form of

$$\begin{cases} E_\alpha(t) = E_{c_\alpha} + \dfrac{E_{n_\alpha} - E_{c_\alpha}}{V_{c_\alpha} + V_{n_\alpha}}[2V_{c_\alpha} + t(V_{n_\alpha} - V_{c_\alpha})]t, \\ D_\alpha(t) = D_{c_\alpha} + \dfrac{D_{n_\alpha} - D_{c_\alpha}}{\omega_{c_\alpha} + \omega_{n_\alpha}}[2\omega_{c_\alpha} + t(\omega_{n_\alpha} - V_{c_\alpha})]t, \end{cases} \quad t \in [0,1]$$

where subscript α corresponds to one of the three component of the respective vector.

If the number of the transition images between two key frames is known, say N, a uniform sampling strategy is adopted to evaluate the viewing parameters of each intermediate image, namely $t = i/(N+1)(i = 1,2,\cdots,N)$. Thus, the transition images can be easily generated by bi-directionally warping the adjacent key frames.

3.2 Interactive operation on the scene

The traditional image-based rendering techniques capturing the source images at preprocessing step, can only be applied to static scenes. On the contrary, our system can deal with dynamical scene, because it employs a hybrid rendering scheme. The dynamic objects can always be rendered in real time using geometric information.

When a user uses mouse or other devices to select some objects in the scene, the system immediately informs the master and the slaves that these objects will be interacted with. These objects are then put into a queue and are excluded from the rendering of next key frame. In the sequential rendering cycle, they will be rendered directly with hardware z-buffer algorithm and their image will be inserted into the respective transition image according its depth information.

3.3 Fixed frame rate

As the complexity of the visible portion of a virtual environment may dramatically change during walkthroughs, it is important to keep a stable refresh rate. To synchronize the generation of the key frames and the image warping process, the mutual wait of the precursor and successor process must be considered. Fig.4 illustrates the algorithm flowchart of fixing refresh rate.

4 Implementation and Results

We have developed a distributed desktop virtual reality system on SGI octane and O2 workstations in our laboratory. In the system, the key frames were generated using Iris Performer software, and Multigen software was used for modeling the scenes. The system was tested with four scenes illustrated in Plate I at different hardware configurations. The first is to take Octane as the master and O2 as the slave. The second exchanges their roles. The remaining two were produced by Iris Performer on Octane and O2, respectively. The test results are listed in table 1. The resolution of the images is

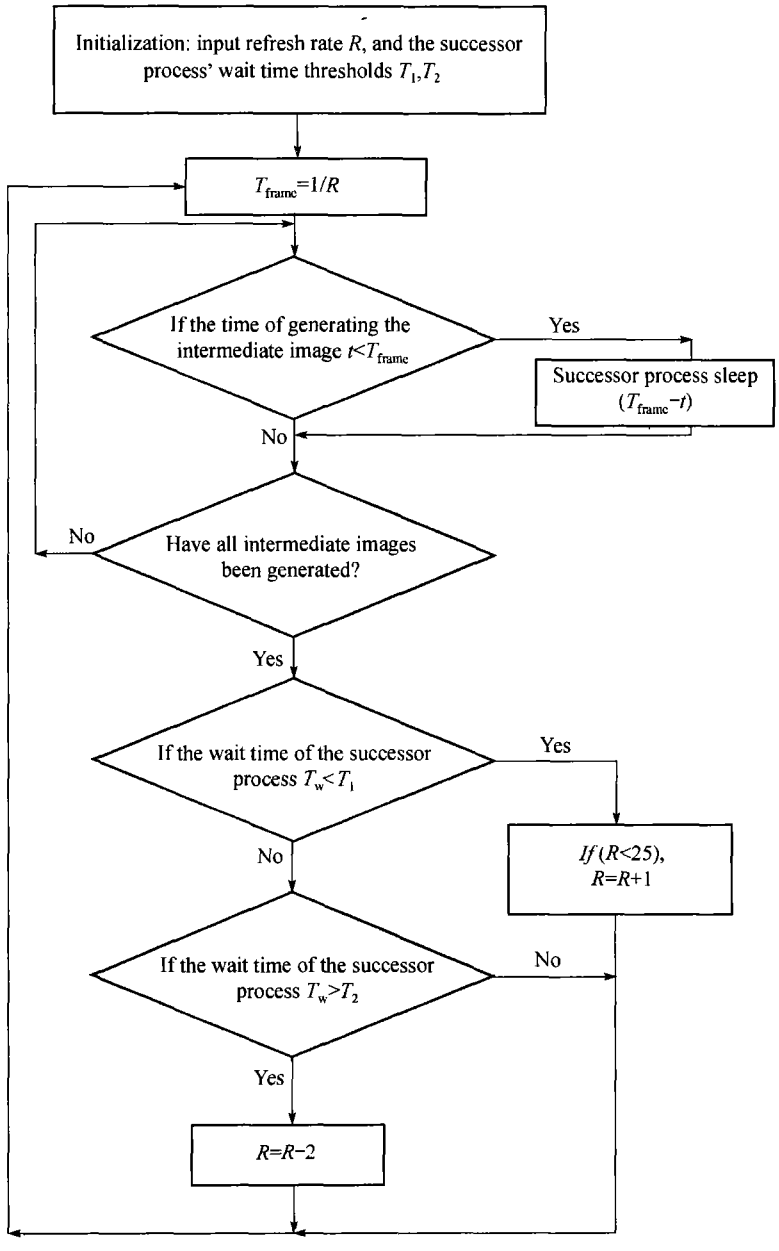

Fig. 4. Flowchart of fixing refresh rate algorithm.

512 × 512 pixels.

It can be found from table 1 that our system achieves much higher frame rates in comparison with the pure geometry-based rendering technique. Since the performance of the O2 workstation is not high enough, when it was used as the master, the frame rate went down. More satisfactory results can be expected with a better hardware configuration.

5 Conclusions and future work

We have described a desktop distributed VR system based on a hybrid rendering algorithm. In the system, a user can use the mouse and keyboard to conveniently specify his viewpoint and viewing direction, and to move, delete or add objects in the scene. We have also presented an efficient algorithm for maintaining a stable refresh rate.

Table 1 Performance of the system with different hardware configurations

Rendering Scheme	Plate I							
	(a)		(b)		(c)		(d)	
	Number	Rate	Number	Rate	Number	Rate	Number	Rate
Performer (O2)	58450	2.5	3000	8.6	24450	4.6	55000	2.6
Performer (Octane)	58450	5.1	3000	18.0	24450	10.0	55000	3.6
Master(O2), Slave(Octane)	2000 ~ 3600	8.2	1500	15.0	1000 ~ 10000	8.5	1800 ~ 2500	10.4
Master(Octane), Slave(O2)	2000 ~ 3600	11.6	1500	20.0	1000 ~ 10000	14.8	1800 ~ 2500	14.3

Note: number means the average number of visible polygons, and rate means the refresh rate using frame/second as unit.

In the current implementations, however, our hybrid geometry- and image-based rendering algorithm, does not account for the variation of scene appearance with respect to the viewing direction. Many researchers have addressed this problem in an attempt to enhance the realism of the produced image sequence. The future work should also focus on developing new algorithms to recover the depth information from video image sequence so that our system can deal with real images.

References

1 Chen, S. E., Williams, L., View interpolation for image synthesis, *Computer Graphics*, 1993, 27(2): 279.

2 McMillan, L., Bishop, G., Plenoptic modeling: an image-based rendering system, *Computer Graphics*, 1995, 29(4): 39.

3 Greene, N., Kass, M., Miller, G., Hierarchical Z-buffer visibility, *Computer Graphics*, 1993, 27(2): 231.

4 Shade, J., Lischinski, D., Salesin, D. H. et al., Hierarchical image caching for accelerated walkthroughs of complex environments, *Computer Graphics*, 1996, 30(3): 75.

5 Sillion, F. X., Drettakis, G., Bodelet, B., Efficient imposter manipulation for real-time visualization of urban scenery, *Computer Graphics Forum*, 1997, 16(3): 207.

6 Torborg, J., Kajiya, J. T., Talisman: commodity real-time graphics for PC, *Computer Graphics*, 1996, 30(3): 353.

7 Carolina, C., Sandin, D. J., Defanti, T. A., Surround-screen projection-based virtual reality: the design and implementation of the CAVE, *Computer Graphics*, 1993, 27(2): 135.

8 Chen, S. E., QuickTime VR: an image-based approach to virtual environment navigation, *Computer Graphics*, 1995, 29(4): 29.

9 Raskar, R., Welch, G., Cutts, M. et al., The office of the future: a unified approach to image-based modeling and spatially immersive displays, *Computer Graphics*, 1998, 32(3): 179.

10 Zheng, W. T., Bao, H. J., Peng, Q. S., Real-time rendering algorithm based on a hybrid rendering scheme, *Progress in Natural Science*, 2000, 10(2): 141.

11 Lihner, V. D., DeFanti, T. A., Distributed virtual reality: supporting remote collaboration in vehicle design, *IEEE Transaction on Computer Graphics and Applications*, 1997, 17(2): 13.

12 Pan, Z. G., Shi, J. Y., He, Z. J., A distributed graphics processing support environment (in Chinese), *Chinese Journal of Advanced Software Research*, 1995, 2(2): 161.

(a)

(b)

(c)

(d)

Four test scenes