Real-Time Image-Based Rendering System for Virtual City Based on Image Compression Technique and Eigen Texture Method

Ryo Sato* Shintaro Ono** Hiroshi Kawasaki* Katsushi Ikeuchi**
*Saitama University
{satou,kawasaki}@cgv.ics.saitama-u.ac.jp
**The University of Tokyo
{onoshin,ki}@cvl.iis.u-tokyo.ac.jp

Abstract

Computer modeling of a large-scale scene such as a city becomes an important topic for computer vision and computer graphics research areas etc. Image-based rendering (IBR) is an effective method for expressing realistic scene, and can construct any arbitrary viewpoint by using the captured real images. However, the large size of the image database in IBR causes serious problems in actual applications, leading to the use of compression techniques. We propose a compression technique based on eigen space combined with a block matching technique to get better result. We also propose a technique to restore the compressed data on Graphic Processing Unit (GPU), allowing us to perform high-speed rendering without raising the load on the CPU.

1. Introduction

In recent years, researches on the computer modeling of large scale scenes such as a city has been intensively conducted in many fields. For constructing and expressing such large scale scenes, model-based rendering (MBR) and image-based rendering (IBR) are mainly proposed.

MBR is a classical method, where a virtual view from any arbitrary viewpoint can be easily reconstructed, by using explicit 3-D geometric model and texture information of the scene. However, reconstruction of models with texture information of the scene requires huge human cost and time. Especially, it is generally difficult to express intricately shaped objects, such as tree with leaves, realistically by MBR.

On the other hand, IBR is a method that reconstructs a virtual view by using a number of images captured beforehand. IBR can easily express the scene with high photo-reality even for intricately shaped objects, since real images are used. One of demerits of IBR is that it requires a large quantity of image data to render large scenes.

Efficiently compressing the image data allows a realistic construction of a large scale scene while keeping the data size manageable.

In our case, we use omni-directional image sequence for the image database. We noticed that the sequence of omni-directional images captured along the road has redundancy, because the same object is captured from multiple view directions. Exploiting this, we propose an effective compression method for the data by using an eigen texture method — applying principal component analysis (PCA), or KL expansion to particular image sequences[1]. It is known this method can give high compression ratio if the similarity among the image sequence is high.

To improve the compression ratio, we propose accurate tracking by using block matching, based on EPI (Epipolar Plane Image) analysis[2]. We also propose a method to restore the compressed data using the GPU, which can achieve high-speed rendering without a high CPU load. By using the proposed methods, we can realize real-time rendering with IBR on a standard PC.

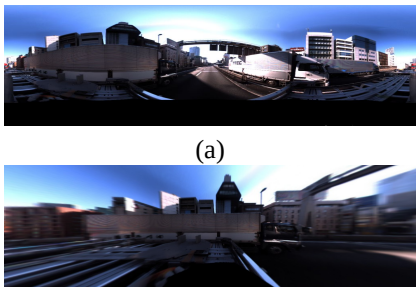
2. Outline of city modeling system

Our system is composed of three processes:
(1) Capturing omni-directional images
(2) Compressing the images
-Image rectification
-Tracking
-Eigen texture method (PCA)
(3) Rendering from the compressed data

3. Compression Method

Many of the applications of IBR make use of omni-directional images (Fig.1(a)) because a wide field of view is available [3, 4]. To capture these images, we used a multiple camera based system, Ladybug2.

Prior to the compression, omni-directional images are projected onto a perpendicular plane along the side of the capturing course, as shown in Figure 1(b). While appearance transition of buildings in an original omni-directional image sequence is non-linear and distorted, the resulting project appearance transition becomes linear and non-distorted. By doing this, the tracking described in 3.1 can be performed easily because it appears as a straight line on an EPI.



(a)

(b)

Figure 1. (a) Omni-directional image, (b) Rectified image

3.1.Tracking

Tracking is preformed in order to find similar part in the image sequence, and two steps: an approximate global tracking and a precise local tracking.

Global tracking by space-time image analysis

We take advantage of space-time image analysis for global tracking. By accumulating the image sequence in the temporal direction, so-called space-time image volume can be constructed. An EPI is an image appears on a planar cross-section of the space-time volume parallel to an epipolar plane among the sequential camera positions. The trajectories of edge lines on the EPI are determined by camera's moving speed and the distance between the camera and the object. If the movement of the camera can be assumed as uniform, or can be roughly normalized as uniform by some external devices, the tracking process is equivalent to determining straight lines on the EPI. In this paper, we assume that the movement of the camera is uniform.

The tracking can be simplified in the case such that the objects face on an approximately constant plane in the real world. A series of building facades in urban scene is a typical example. In such case, the global tracking can be expressed by one parameter — choosing a typical inclination among the straight lines in the EPI. Straight lines in the EPI can be detected by applying edge detection followed by a Hough transform, which gives ρ and θ in Figure 2. The global tracking parameter is the amount of movement of the object in one frame, which is equal to $-\tan\theta$.

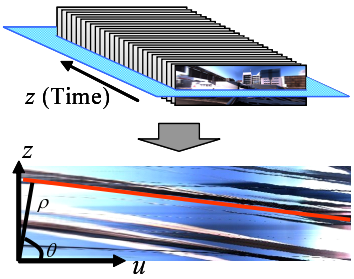


Figure 2. EPI

Local Tracking by Block-Matching

In order to get higher compression ratio, partial image sequences which are subject to be compressed are desired to be similar as much as possible. We perform local tracking to realize this. The process is shown in figure 3. According to the global tracking, a block image drawn in bold at frame f_1 should move to the dotted block in frame f_2 and f_3. In actual case, however, because of vibration and minute change of moving speed of the camera, the corresponding image blocks are shifted from the result of global tracking. Therefore, the block image with the highest correlative value with the block image of previous frame is searched by doing block matching between the frames around the result of the global tracking.

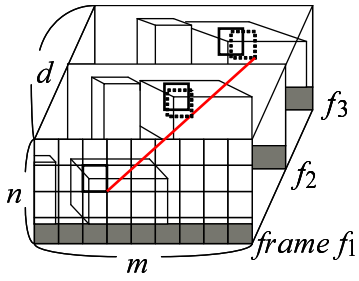


Figure 3. Tracking using block maching

3.2.Compression by eigen texture

First video frame is divided into the unit of the block of $m \times n$ (figure 3). For each block image, the tracking

described in 3.1. is applied up to the d-th frame, then the sets of similar block images are obtained. PCA is applied to each block set, which is compressed into k eigen images. Applying this to all $m \times n$ block sets, the whole image sequence is compressed.

4. Rendering

4.1.Restoration of Compressed Images

Restoration of the image uses the eigen image which is obtained by PCA, and the image of the original i-th frame X_i can be restored by the linear sum as follows.

$$X_i = \sum_{k=1}^{r} (w_{ik} * V_k) \tag{1}$$

where k-th eigen image is V_k, weight coefficient of i-th frame is w_{ik}, and the number of principal component required in order to achieve a certain cumulative proportion is r.

4.2.Rendering Algorithm

Figure 4 shows our method of rendering compressed data in the case of 3×1 block sets in figure 3. Rendering at a certain viewpoint P is to put the texture corresponding to an angle composed of the viewpoint P and the center of wall plane, and the texture is selected from among the image set captured on the capturing route A,B,C [4]. The texture at this time is obtained by restoring the image which corresponds to θ_1, θ_2, θ_3 from the eigen images in each A,B,C.

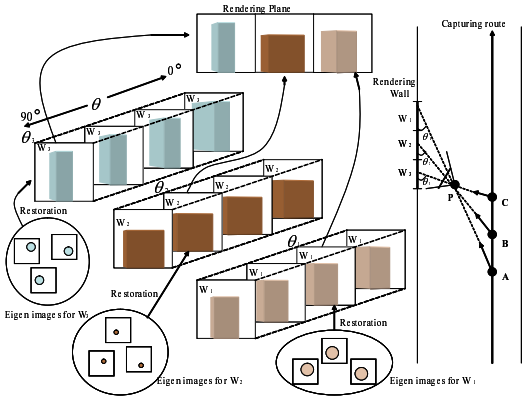
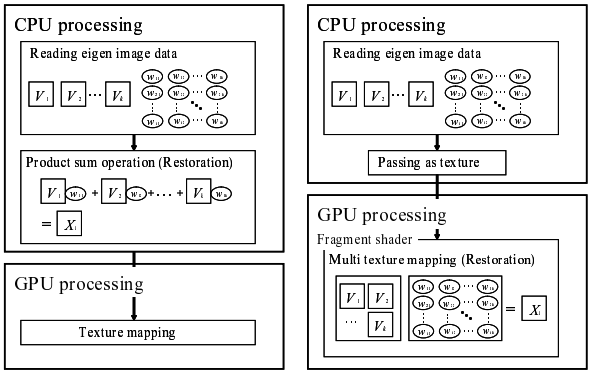


Figure 4. Rendering algorithm

4.3.Fast restoration using GPU

We explain how to restore an original image X_1 using figure 5. Firstly the compressed data (eigen images V_k and weight parameters w_{ik}) are loaded onto the main memory, and product sum operation of V_k and w_{ik} is processed using CPU (Figure 5.(a)). The image obtained is then passed to GPU as a texture, and it is used for texture mapping. This is an ordinary CPU-based restoration process. However, when restoring the original image X_2 secondly, the procedure same as w_{2k} is needed in a weighting coefficient and processing takes time. The complexity of this method is of $O(n^2)$, so the load on CPU increases rapidly if the texture size becomes large. On the other hand, in GPU-based method, after the compressed data has been read using the CPU, it is passed to GPU as textures, and a product sum operation is processed only in the fragment shader of GPU (figure 5(b)). Even when restoring X_2, it can process by the product sum operation of V_k and w_{2k} which have been held beforehand, and it is not necessary to remake a texture.



(a) CPU-based (b) GPU-based

Figure 5. Image restoration process

5. Experiments

5.1.Result of Tracking and Compression

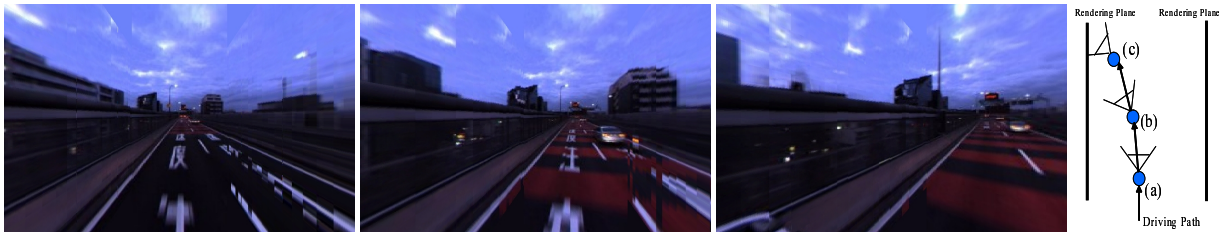
Each of the rectified omni-directional image is divided into the block size 16 × 16 pixel, and the data of 70 frames obtained by the tracking is compressed. Table 1 is final result of compression. It turns out that more similar block image set can be obtained by improvement of the tracking accuracy, and the compression ratio is improved.

5.2.Comparison in Processing Time for Rendering

The processing time for rendering by GPU-based method was compared with CPU-based method. The comparison is performed by measuring the time of processing that repeats process of restoring original 70 images sequentially 100 times. The spec of used PC is Intel Core 2 (2.66GHz), and GPU is Quadro FX 550. The result is the graph of figure 6. It turned out that the processing time in the CPU-based method is increasing

Table 1. Result of compression

	EPI	EPI+Block maching
Average of Correlative value	0.756491	0.936779
Cumulative propotion 50% (Compression ratio of data)	7.3 (28.1/280KB)	4.3 (16.6/280KB)
Cumulative propotion 70% (Compression ratio of data)	14.0 (53.6/280KB)	10.6 (40.9/280KB)
Cumulative propotion 90% (Compression ratio of data)	28.6 (109/280KB)	28.3 (108/280KB)



(a) (b) (c) (d)

Figure 7. Result of rendering.

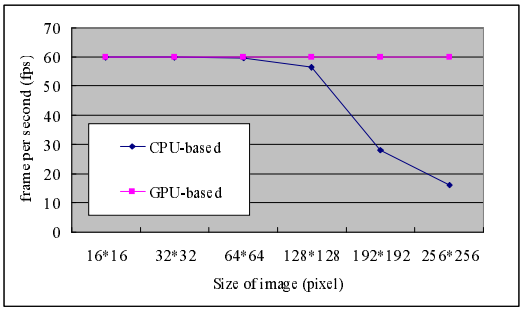


Figure 6. Comparison of processing time

rapidly as the image size becomes larger(Fig.6). On the other hand, the GPU-based method keeps constant processing time regardless of the image size.

5.3.Result of rendering

Figure 7 (a), (b), (c) are the results of rendering, which are the scenes from viewpoints (a), (b), (c) respectively in the Figure 7 (d). These scenes are rendered by sum of eigen images under 75% cumulative proportion.

6. Conclusion

In this paper, we proposed two technique for construction of large scale scene. One is the novel compression method for omni-directional images by using eigen textures, and the other is a high-speed rendering algorithm using the GPU. Since compression ratio becomes low when there is translational difference among a series of image blocks subject to be compressed, we proposed a tracking method using block matching based on EPI analysis, and improved the compression ratio. The compression data can be easily restored by product sum operation of the eigen images and the weight coefficient. Taking advantage that this operation is totally linear, we implemented the operation using fragment shader in a graphic hardware, and achieved more high-speed and stable rendering without raising the load of CPU.

References

[1] K. Nishino, Y. Sato, K. Ikeuchi, “ Eigen-texture method: Appearance Compression based on 3D Model ”, IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), pp. 618-624, Jun. 1999.

[2] R. Bolles, H. Baker and D. Marimont: “ Epipolar plane image analysis: an approach to determining structure from motion ”, Int. J. of Computer Vision, Vol. 1, pp. 7-55, 1987.

[3] C. J. Taylor, “ Video plus ”, IEEE Workshop on Omnidirectional Vision, pp. 3-11, 2000.

[4] H. Kawasaki, K. Ikeuchi, and M. Sakauchi,“ Light field rendering for image-scale scenes ”, Proc. International Conference on Computer Vision and Pattern Recognition (CVPR), Vol. 2, pp. 64-71, Kauai, Hawaii, US, 2001.