Implementing a Real-time Free-Viewpoint Video System on a PC-Cluster

Ueda, Megumu
Department of Intelligent Systems, Kyushu University

Arita, Daisaku
Department of Intelligent Systems, Kyushu University

Taniguchi, Rin-ichiro
Department of Intelligent Systems, Kyushu University

出版情報：Proc. of the 7th International Workshop on Computer Architecture for Machine Perception, pp.167-171, 2005-07
バージョン：
権利関係：

Implementing a Real-time Free-viewpoint Video System on a PC-cluster

Megumu Ueda, Daisaku Arita, Rin-ichiro Taniguchi
Department of Intelligent Systems
Kyushu University
6-1, Kasuga-koen, Kasuga, Fukuoka 816-8580 Japan
URL: http://limu.is.kyushu-u.ac.jp/
Email: {ueda, arita, rin}@limu.is.kyushu-u.ac.jp

***Abstract*— In this paper, we present a system generating free-viewpoint video in real-time using multiple cameras and a PC-cluster. Our system firstly reconstructs a shape model of objects by the visual cone intersection method, secondly transforms the shape model represented in terms of a voxel form into a triangular patch form, thirdly colors vertexes of triangular patches, and finally displays the shape-color model from the virtual viewpoint directed by a user. Here, we describe implementation details of our system and show some experimental results.**

I. Introduction

Currently, televisions are used for real-time, or live, distribution of scenes of the world. In television, however, a video captured by a camera is displayed on a screen and the viewpoint is chosen only among camera positions specified by the program director, not among arbitrary positions. On the other hand, computer graphics techniques can generate a free-viewpoint video, in which a viewer can changed the viewpoint to arbitrary positions. However, computer graphics requires a structure and motion model of objects and it is time consuming to construct such a model in advance. Then, we aim to construct a computer graphics model by computer vision techniques in real-time for generating live free-viewpoint videos.

Several researches have been done for generating free-viewpoint videos using multiple cameras since Kanade et al.[1] had proposed the concept of "Virtualized Reality". We can classify the researches into two approaches. The first approach reconstructs 3D shapes of objects and the second one does not reconstruct them. We select the first approach because it can generate free-viewpoint videos with less cameras and less memories. As the first approach, Matsuyama et al.[2], Carranza et al.[3] and Davis et al.[4] have developed systems which generate a computer graphics model from multiple camera videos. Although they achieve relatively precise shape reconstruction and model coloring, it requires a lot of computation time, and, then, real-time processing can not be achieved. In contrast, our system can generate free-viewpoint videos in real-time based on a new model coloring method presented here, sacrificing precision of 3D shape reconstruction. In other words, our system can be used for live videos from arbitrary viewpoints.

II. Parallel Processing Model on a PC-cluster

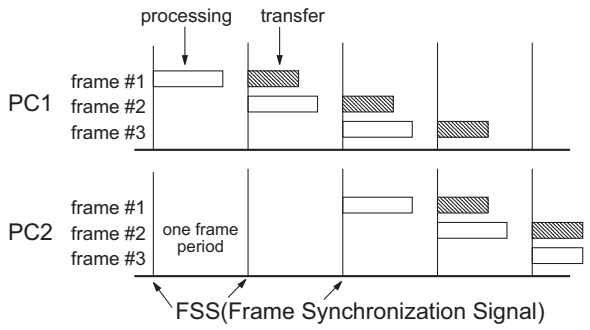
It is quite time consuming to generate free-viewpoint videos, and, therefore, its online, or real-time, processing requires a high-performance computing system. We have employed a PC-cluster to implement a real-time free-viewpoint video system, because it provides quite a high cost performance based on parallel processing techniques. The key issue is programming methodology, especially for real-time algorithms, on a PC cluster. Here, we have implemented our system using RPV[5] on a PC-cluster, which is a programming environment for real-time image processing on a distributed parallel computer such as a PC-cluster. RPV supports several schemes of parallel processing, such as data parallel (space division and time division), function parallel, pipeline parallel, and their combination. Process execution model employed by RPV is periodical data processing and communication, which is synchronous to real-time image sequence (see Fig. 1). Basically, RPV supports frame-synchronous data execution, which means that one frame period, 33msec in case of 30fps camera, is allocated to process one image frame and another frame period is allocated to transfer one image frame to the succeeding PC (see Fig. 1 (a)). This causes a large latency when we have a long pipeline structure.

To make the latency as small as possible keeping high throughput, we have introduced stream data transfer, or fine grained data transfer, to RPV. One frame data is divided into small packets consisting of pixels, lines, voxels, or other image features, and data processing and transfer are applied to each packet. When each PC finishes processing a packet, it immediately starts sending it to the succeeding PC and then starts processing the next packet (see Fig.1 (b)). Therefore, when the packet arrives at the succeeding PC, the processing of the packet can be started immediately. This mechanism can greatly reduce the latency.

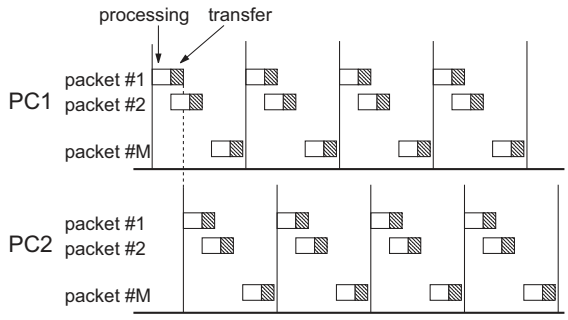
III. Free-viewpoint Video Generation

If it is possible to divide a process into several packet-based processings, the process is divided and allotted to PCs, and the process is processed in parallel. Then we must consider process time and communication. We must make process time of each PC be almost the same and we must not make too many PCs since too many PCs make a large latency and the

Proceedings of the Seventh International Workshop on Computer Architecture for Machine Perception (CAMP'05)
0-7695-2255-6/05 $20.00 © 2005 IEEE



(a) Frame-based data execution



(b) Packet-based data execution

Fig. 1. Pipeline processing model on a PC-cluster

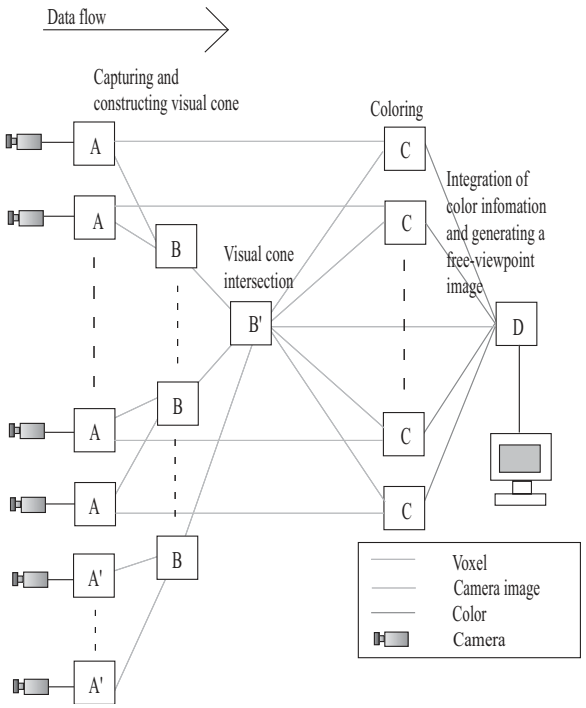


Fig. 2. System configuration

number of PCs is limited. So we have thought that following processes are best balance.

1) Reconstructing a shape model of objects by the visual cone intersection method[6].
2) Transforming the shape model represented in terms of a voxel form into a triangular patch form by the discrete marching cubes method[7].
3) Coloring vertexes of triangular patches varying with the position relation between the virtual viewpoint directed by a user and the viewpoints of cameras.
4) Displaying the shape-color model from the virtual viewpoint with painting triangular patches by interpolating among vertexes.

These processes are distributed to PCs shown in Fig. 2 and executed in pipeline parallel.

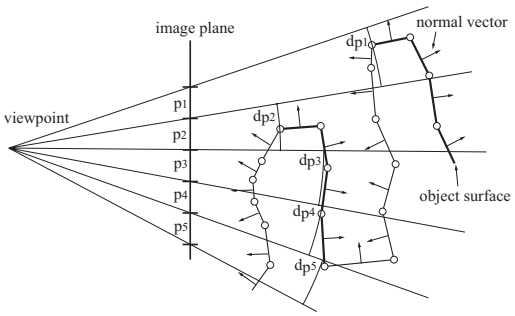
a) **Node-A:** First, each node-A extracts object silhouettes from video frames captured by a camera by background subtraction and noise reduction. Secondly, each node-A constructs visual cones. A visual cone is defined as a cone whose apex is the viewpoint and whose cross section coincides with the silhouette of the object. Visual cones are represented in terms of a voxel space. Finally, each node-A sends the visual cones to a node-B and sends the colored silhouette image to a node-C.[1]

b) **Node-B:** Each node-B gathers and intersects visual cones from multiple viewpoints to construct a shape model of the objects represented in terms of a voxel space. Since this process is time consuming, it is distributed to multiple node-Bs hierarchically. Node-B', the last node of node-Bs, transforms the finale shape model represented in terms of a voxel space into that in terms of triangular patches. However, node-B' sends the voxel space and its corresponding patterns of the discrete marching cubes method instead of triangular patches since the triangular patch form is not efficient from the viewpoint of data size.

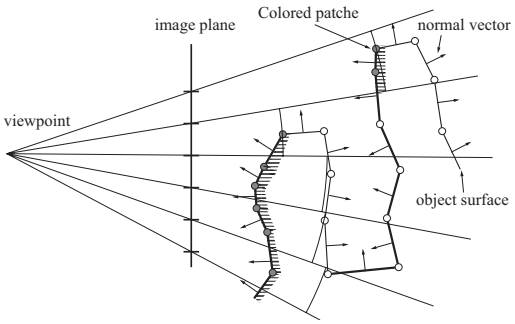
c) **Node-C:** First each node-C transforms the shape model represented in terms of a voxel space into those of triangular patches by the discrete marching cubes method using patterns sent from node-B'. Then, each node-C colors visible vertexes of the shape model based on one camera image. Finally, each node-C sends color information of all vertexes of the shape model.

For coloring vertexes in real-time, it is necessary to quickly judge whether each vertex is visible from the camera or not. Conservative visibility check method has to check whether each vertex is occluded by each triangular patch. That computation amount is $O(N^2)$, where N is the number of vertexes. So we propose a new method based on the Z-buffer method, whose computation amount is $O(N)$. Our method consists of two steps (See Fig. 3). At the first step, node-C searches for

[1]To reduce the computation time, some of node-As, indicated as node-A' in Fig. 2, do not send the colored silhouettes to reduce their redundancy. The selection is done in advance based on the camera arrangement.

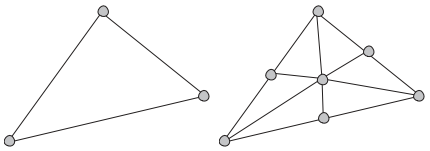


(a) step 1



(b) step 2

Fig. 3. Coloring vertexes



(a) Original triangular patch (b) Divided triangular patch

Fig. 4. Dividing triangular patch

the object surface which faces against the viewpoint and which is nearest to the viewpoint in each pixel p. This step is realize by the Z-buffer method altered to taking account of not all surfaces but only surfaces facing against the viewpoint. Then, node-C lets d_p be the distance between the viewpoint and the nearest surface. At the second step, node-C colors all vertexes which faces toward the viewpoint and which is nearer to the viewpoint than d_p in each pixel p. The color of the vertexes is that of pixel p. At this time, each triangular patch is divided into six triangular patches as shown in Fig. 4 since increasing the number of vertexes makes coloring resolution higher without lengthening processing time for shape reconstruction.

d) **Node-D:** First node-D receives the position of the virtual viewpoint directed by a user.

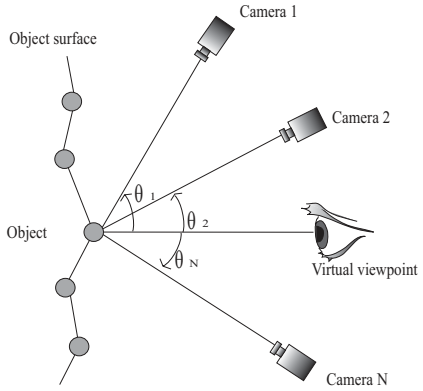


Fig. 5. Angle between camera and virtual viewpoint

Secondly node-D transforms the shape model represented in terms of a voxel space into those of triangular patches by the discrete marching cube method in the same way as node-C. There are two reasons why shape model transformation is made on both node-C and node-D. The first one is because the data size of triangular patches is very large and the time to transport triangular patches is too long. The second one is because processing times of node-B, node-C and node-D are balanced best.

Thirdly, node-D integrates color information of all cameras. The integrated color value for each vertex is weighted mean of color value from node-C. The weight W_n of camera n is calculated by the following expression;

$$W_n = \frac{(cos\theta_n + 1)^\alpha}{\sum_{x=0}^{N}(cos\theta_x + 1)^\alpha} \quad (1)$$

where N is the number of cameras visible the vertex, θ_n is the angle between the vector from the virtual viewpoint to the vertex and that from the camera viewpoint to the vertex (See Fig. 5). α is decided from balance between processing time of node-D and other nodes. In this experiment, we let α be 1. Color value of a vertex visible from no camera is let be same as the mean value of neighbor vertexes.

Finally, node-D generates an image from the directed viewpoint. Each triangular patch is painted by interpolating among vertexes.

IV. Experiments

Using our proposed system, we generate free-viewpoint video in real-time to evaluate the precision of generated images, processing time of each node, latency, and the amount of data transfer. We have used seven IEEE-based cameras with 320×240 pixel resolution (see Fig. 6), and 17 PCs (six node-As, one node-A', two node-Bs, one node-B', six node-Cs, one node-D), each of which has an Intel Pentium4 (3GHz), 1GB memory and NVidia GeForce FX. PCs are connected by Myrinet. All the cameras are calibrated in advance by Tsai's method[8]. Voxel space resolution is $128 \times 128 \times 128$ and the size of a voxel is 2cm.

TABLE I

AMOUNT OF DATA SENDING FROM EACH NODE

Node	Average (Kbyte)
A (Image)	93.5 (variable)
A and B (voxel)	256 (constant)
B'	28.6 (variable)
C	92.0 (variable)

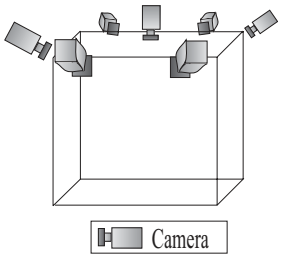
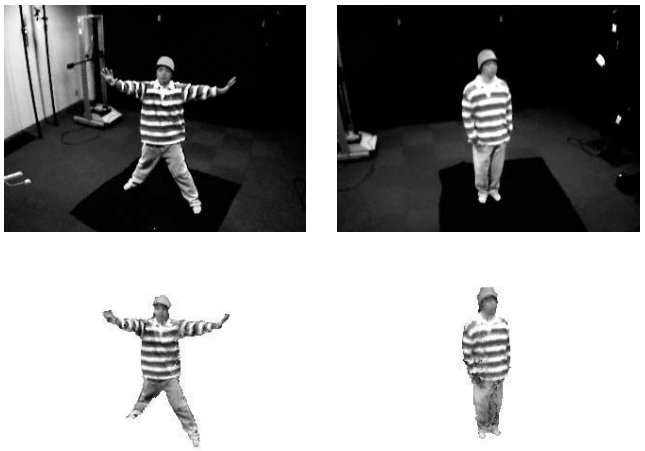


Fig. 6. Camera arrangement



(a) viewpoint 1 (b) viewpoint 2

Fig. 7. Camera images(upper) and generated images(lower)

Fig. 7 shows two pairs of a camera image and a generated image whose viewpoints are same. Fig. 8 shows four generated images from virtual viewpoints. Each image is well-generated. Fig. 9 shows the sum of root mean square errors between a camera image and a generated image with a same viewpoint. This may be caused by

- shape reconstruction error,
- camera calibration error,
- re-sampling error from image pixels to triangular patch vertexes, and
- color integration error.

Fig. 10 shows the mean of processing time of each node in case that there is one person in the experimental space and the latency of the system is 200ms. We think balance of allotting processes is good from Fig.6. Table. I shows the amount of data sending from each node. Node-D receives the largest amount of data, 600KB/frame, of all nodes. This data size requires 4.8ms for receiving via Myrinet. And the actual throughput of the system is about 20fps and 13fps in case of one person and two persons respectively. This means that the actual throughput is lower than the theoretical one calculated by adding the longest processing time (node-D) and the longest data-sending time (to node-D) owing to the overhead of OS such as process switching. However enough performance is realized by using a PC-cluster and communication dose not influence the throughput so much, 20fps and less than.

V. CONCLUSION

In this paper, we propose a system generating free-viewpoint videos using multiple cameras and a PC-cluster in real-time. And we make some experiments to show the performance and the precision of our system.

Major future works especially from the view point of parallel/distributed processing are as follows.

Reduction of latency

The latency of the current system is about 200ms. That value is not small. We think that packet-based processing and data compression is effective for reduction of latency. Currently, the packet-based processing is introduced only from node-A to node-B' on the current system. Therefore, we will introduce it to all nodes.

Invariable throughput

The throughput of our system varies depending on the scenes since the number of voxels and the number of triangular patches depend on the size and the shape of objects. By introducing variable resolution of the voxel space we suppose we can make the throughput relatively unvaried.

REFERENCES

[1] P. J. Narayanan, T. Kanade, and P. W. Rander, "Concepts and early results," in *Proc. IEEE Workshop on the Representation of Visual Scenes*, June 1995, pp. 69–76.

[2] T. Matsuyama, T. T. Xiaojun Wu, and S. Nobuhara, "Real-time generation and high fidelity visualization of 3d video," in *Proc. of MIRAGE2003*, Mar. 2003, pp. 1–10.

[3] J. Carranza, C. Theobalt, M. A. Magnor, and H.-P. Seidel, "Free-viewpoint video of human actors," in *ACM Trans. on Graphics*, vol. 22, no. 3, July 2003, pp. 569–577.

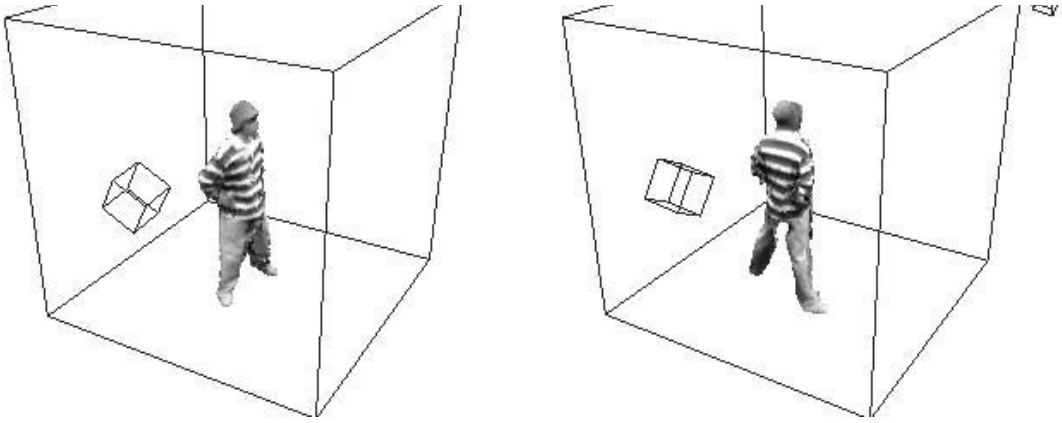
[4] E.Borovikov and L.Davis, "A distributed system for real-time volume reconstruction," in *Proc. 5th Int. Workshop on Computer Architecture for Machine Perception (CAMP2000)*, 2000, pp. 183–189.

[5] D. Arita and R. Taniguchi, "RPV-II: A stream-based real-time parallel vision system and its application to real-time volume reconstruction," in *Proc. of Second International Workshop on Computer Vision System*, July 2001, pp. 174–189.

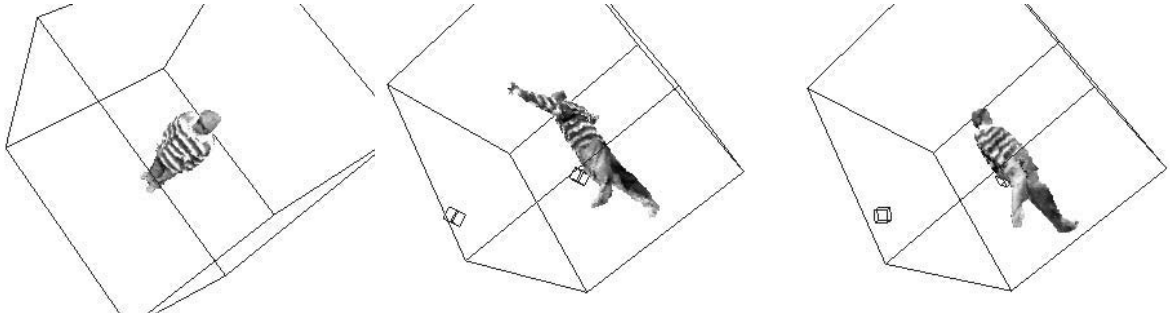
[6] W. N. Martin and J. K. Aggarwal, "Volumetric description of objects from multiple views," in *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 5, no. 2, 1983, pp. 150–158.

[7] Y. Kenmochi, K. Kotani, and A. Imiya, "Marching cubes method with connectivity," in *Proc. on International Conference on Image Processing*, vol. 4, Oct. 1999, pp. 361–365.

[8] R. Y. Tsai, "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses," in *IEEE Trans. on Robotics and Automation*, vol. 3, no. 4, 1987, pp. 323–344.



(a) virtual viewpoint A (b) virtual viewpoint B



(c) virtual viewpoint C (d) virtual viewpoint D (e) virtual viewpoint E

Fig. 8. Generated virtual-viewpoint images

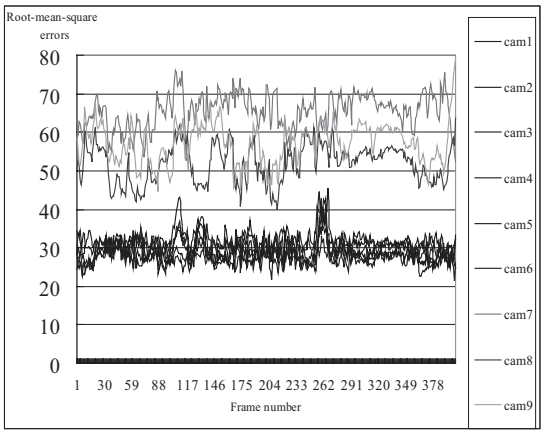


Fig. 9. Error: Cam 7 is ceiling camera unused for model coloring. Cam 8 and cam 9 are cameras unused for shape reconstruction and model coloring

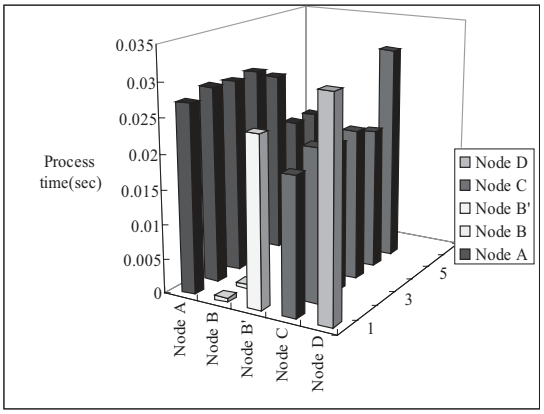


Fig. 10. Processing time from each node

Proceedings of the Seventh International Workshop on Computer Architecture for Machine Perception (CAMP'05)
0-7695-2255-6/05 $20.00 © 2005 IEEE