Real-Time Free-viewpoint Video Generation Based on 3D Shape Reconstruction: Toward High-Fidelity Video Generation

Ueda, Megumu
Department of Intelligent Systems, Kyushu University

Nabeshima, Rui
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：Proceedings of the 12th Korea-Japan Joint Workshop on Frontiers of Computer Vision, pp.71-75, 2006-02
バージョン：
権利関係：

Real-Time Free-viewpoint Video Generation Based on 3D Shape Reconstruction: Toward High-Fidelity Video Generation

Megumu Ueda, Rui Nabeshima, Daisaku Arita and Rin-ichiro Taniguchi

Department of Intelligent Systems, Kyushu University,
{ueda, nabeshima, arita, rin}@limu.is.kyushu-u.ac.jp

Abstract In this paper, we present a system generating free-viewpoint image effectively in real-time using multiple cameras and a PC-cluster. Here, we describe implementation details of our system and show some experimental results.

1 Introduction

Currently, televisions are used for real-time, or live, distribution of scenes of the world. In television, however, an image captured by a camera is displayed on a screen and the viewpoint is chosen by the program director, who selects a view from possible camera positions. The next generation technique of the television is expected to be free-viewpoint system, where a viewer can change the viewpoint to arbitrary position. The basis of the free-viewpoint system is computer graphics techniques, which can generate free-viewpoint images. However, computer graphics requires 3D structure and motion information of objects and it is time consuming to construct such information in advance. Then, we aim to construct a computer graphics model by computer vision techniques in real-time for generating live free-viewpoint images.

Several researches have been done for generating free-viewpoint images using multiple cameras since Kanade et al.[1] 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 does not. We have selected the first approach because it can generate free-viewpoint images with relatively a small number of cameras and memories. As the first approach, Matsuyama et al.[2] and Carranza et al.[3] developed systems which generate a computer graphics model from multiple camera images. Although they achieved relatively precise shape reconstruction and model coloring, it requires a lot of computation time, and, then, real-time processing can not be achieved. Gross et al.[4] and Grau et al.[5] developed systems which generate a computer graphics model from multiple camera images in real-time, sacrificing precision of 3D shape reconstruction. Since they colored the generated 3D shape model without judging a visibility from cameras, the model coloring often makes errors. In contrast, we have proposed a system which can not only generate 3D shape of objects but also color the 3D shape model in real-time with a new model coloring method. However, there still remains a room to improve the quality, and, in this paper, to generate more photo-realistic free-viewpoint images in real-time, we introduce the idea of deformed voxel space, which makes it possible to represent 3D object shapes more accurately without much increase of processing time and memory space.

2 Free-viewpoint Image Generation System

2.1 System Configuration

It is quite time consuming to generate free-viewpoint images, and, therefore, its online, or real-time, processing requires not only efficient algorithms but also a high-performance computing system. We have employed a PC-cluster to implement a real-time free-viewpoint image system, because it provides quite a high cost performance based on parallel processing techniques.

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 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].

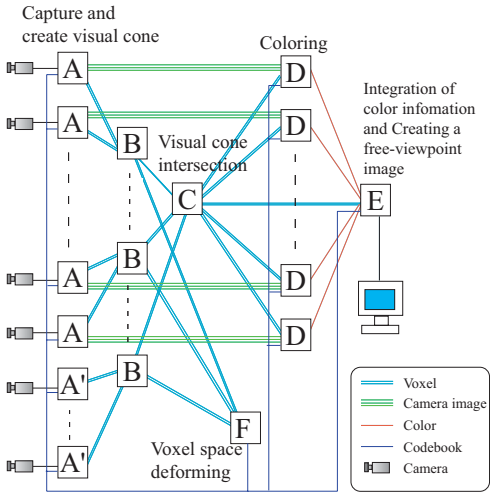

Fig. 1: System configuration

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.

Moreover, we add techniques to refine a generated image, which are mesh model smoothing and voxel space deformation. Shape model smoothing and voxel space deforming are new function for high fidelity image generation. These processes are distributed to PCs shown in Fig. 1 and executed in pipeline parallel.

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

Node-B and Node-C 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 receiving this large data at a time is time consuming, it is distributed to multiple node-Bs and Node-C hierarchically. Node-C transforms the finale shape model represented in terms of a voxel space into that in terms of triangular patches. Node-C sends the voxel space and its corresponding patterns of the discrete marching cubes method.

Node-D First, each Node-D 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-C, and, then, smooths the shape model (See section. 3). Then, each Node-D colors visible vertexes of the shape model based on one camera image (See section. 2.2). Finally, each Node-D sends color information of all vertexes of the shape model.

Node-E First, Node-E receives the position of the virtual viewpoint directed by a user. Secondly Node-E transforms the shape model represented in terms of a voxel space into those of triangular patches by the discrete marching cube method, and, then, smooths the shape model in the same way as Node-D (See section. 3). Thirdly, Node-E integrates color information of all cameras (See section. 2.2). The integrated color value for each vertex is weighted mean of color value from Node-D. Finally, Node-E generates an image from the directed viewpoint. Each triangular patch is painted by interpolating among vertexes.

Node-F First, Node-F gathers and intersects visual cones from multiple viewpoints to construct a shape model of the objects represented in terms of a voxel space. Secondly Node-F calculates a new voxel space depending on the shape model (See section. 3). Thirdly, Node-F sends the information of the new voxel space to all other nodes without Node-B and Node-C.

2.2 Coloring

Node-D obtains the color information of vertex depending on an colored silhouette image and Node-E integrates these color information. The details are described in [8].

2.2.1 Visibility Check

For coloring vertexes in real-time, it is necessary to judge quickly whether vertex is visible from a camera or not. We use two features of invisible patches. First, patches facing toward the viewing direction are invisible. Second, patches occluded by

those patches are also invisible. So Node-D searches for patches facing toward the camera view direction in the first step. Secondly, Node-D searches for patches occluded by other patches which Node-D searched in the first step. This second step is achieved by using the Z-buffer method. Thirdly, Node-D colors remaining patches.

2.2.2 Color Integration

Node-E integrates the color information of all cameras, calculating weighted mean of the color acquired by the cameras. The weight W_n of camera n is calculated by the following expression;

$$W_n = \frac{(cos\theta_n cos\phi_n + 1)^\alpha}{\sum_{k=0}^{N}(cos\theta_k cos\phi_k + 1)^\alpha} \quad (1)$$

where N is the number of cameras which can view 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, ϕ_n is the angle between the vector from the camera viewpoint to the vertex and the normal vector of the patch. By this equation, the smaller an angle between virtual viewpoint and a camera becomes, the larger the camera's weight becomes.

3 Shape Refinement

We have implemented two techniques for high fidelity image generation: the first one is a mesh model smoothing and the second one is a voxel space deforming. These techniques are applicable at the same time, The mesh model smoothing is effective in the discrete marching cubes method and The voxel space deformation is effective in the visual cone intersection. Our system can generate more photorealistic free-viewpoint images in real-time using these techniques.

3.1 Surface Smoothing

Usually, reconstructed 3D shape models are not very accurate because a mesh model which the discrete marching cubes method generates[7] is a bit jaggy. Therefore, generated virtual views based on the reconstructed shape models tend to be not photorealistic, and it is essential for us to smooth reconstructed shape models. We have chosen the laplacian smoothing that is simple and fast. The laplacian smoothing is done by simply averaging the positions of neighboring vertexes iteratively. One step of the process is described by the following equation;

$$v_i^{new} = v_i^{old} + \sum_{j=0}^{n}(\frac{v_j^{old} - v_i^{old}}{n}) \quad (2)$$

where v_i and v_j are vertexes of triangular patches of mesh model, v_j shares a triangular patche with v_i and n is the number of v_j. When the laplacian smoothing is applied iteratively, the effect of smoothing grows. The laplacian smoothing is quite useful for real-time smoothing. However, if the laplacian smoothing is applied too much times, features of objects are lost, and, practically, we should apply it once or twice (Fig. 2).

Fig. 2: Smoothing mesh model

The important point is that, in the coloring process, we need correspondence between pixels on the image plane and vertexes of the reconstructed 3D shape model. When we do not apply the smoothing, the correspondence can be calculated in advance, because each vertex is on one of the sample points of the voxel space, and because the correspondence between the sample points and the pixels is fixed. However, after smoothing, positions of the vertexes move arbitrarily, and we can not calculate the correspondence in advance. To save computation time, we calculate the new correspondence referring to the original correspondence. We can get a pixel position corresponding to a new vertex position by calculating weighted mean of pixel positions corresponding to old vertex positions, which are used to calculate the new vertex position.

This process is done at Node-D and Node-E. The reason why this process is not done at Node-C is a balance of processing time and sending time of each node.

3.2 Voxel Space Deformation

High resolution of voxel space is required to obtain a high fidelity shape model using visual cone intersection. However, higher resolution requires higher computation, or the computation cost grows with $O(N^3)$, where N is the number of voxels in one dimension. One solution to this problem is that, without increase of the number of voxels, the voxel space is deformed depending on object shapes so

as that the sampling points represent the objects more accurately. Using the deformed voxel space, the computation amount doesn't grow largely, and it needs only the voxel space deforming calculation.

The idea of voxel space deformation is quite simple: empty sample points adjacent to the object are moved toward the object, and the movements depend on the configurations of neighboring sample points. After moving the empty sample points, the emptiness is checked, and the object shape is reconstructed.(Fig. 3). The new position of a moved sample point should be explained as weighted mean of sample points of neighborhood because, the same as in the section 3.1, we require the correspondence between pixels on the image plane and vertexes of the reconstructed 3D shape model. Here, we should be careful with the following two points: one is that the order of sampling point positions should not be reversed; the other is that sampling points should not move away from the next sampling point. These issues are solved by the voxel space deformation represented in the following equation;

$$s_i^{new} = s_i^{ini} + \lambda \sum_{s_j \in Near(s_i)} (s_j^{ini} - s_i^{ini}), 0 < \lambda < 1 \quad (3)$$

where s is a coordinate of a sampling point and s^{ini} is an initial coordinate value, s_i is an empty sample point, $Near(s_i)$ means that non-empty sample points adjacent to s_i. λ is decided by the number of possible movements of sampling points, which are decided in order that the order of sampling point positions are not reversed. Currently, to maintain the real-time feature of the system, the result of voxel space deforming is applied to the next frame, and 3D shape is reconstructed in the deformed voxel space on the next frame. Thus, objects moving fast can not take a merit of voxel space deforming, but 3D shapes of objests moving slow or not moving can be reconstructed more accurately. Fast recalculation of 3D shape in the deformed voxel space is one of important future work.

Amount of the information should be a little because Node-F sends the information of a new voxel space to many nodes. So Node-F sends only the information about moving sample points.

4 Experiments

Using our proposed system, we generate free-viewpoint image 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 nine IEEE-based cameras with 640×480 pixel resolution (One of cameras is ceiling camera), and 21 PCs (six node-As, three node-A', three node-Bs, one node-C, six Node-Ds, one Node-E, one Nod-F), 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[9]. Voxel space resolution is $128 \times 128 \times 128$ and the size of a voxel is 2cm. Sample points can move 5mm by three turns using voxel space deforming.

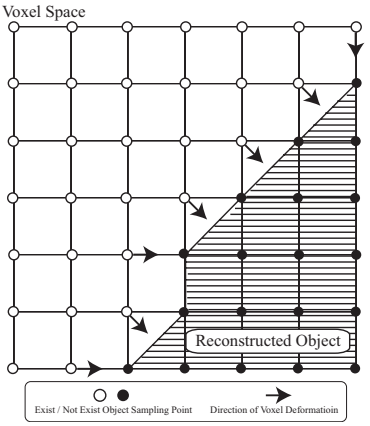

Fig. 3: Voxel Space Deformation

Table 1: Processing time and amount of data sending

Node	Time (msec)	Data (KByte)
A	25	350
B	0.5	256
C	12	35.4
D	48	205
E	44	None
F	65	10.8

Fig. 4 shows three generated images from virtual viewpoints. Each image is well-generated. Fig. 5 shows comparison with normal voxel space and deformed voxel space. Deformed voxel space can reconstruct a thin shape, such as a finger, more than normal voxel space. Table. 1 shows the average processing time of each node in case that there is one person in the experimental space and the amount of data sending from each node. The average of the throughput is about 10fps in case that there is one person. The average latency of the system is about 350ms in case that there is one person. Maximum processing time of each node is processing time of Node-F, about 65ms, but throughput is about 10 fps. The reason is that Node-F sends feedback to Node-A or Node-A'. This is because Node-A or Node-A' can not start the process until it receives feedback from Node-F in the next frame.

(a) Viewpoint 1

(b) Viewpoint 2

(c) Viewpoint 3

Fig. 4: Generated images

(a) Normal voxel space

(b) Deformed voxel space

Fig. 5: Comparison with normal and deformed voxel space

5 Conclusion

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

Major future works are as follows. First, we must make processing time of Node-F which deforms voxel space shorter. Secondly, more powerful voxel space deformation is useful for fidelity image generation. For example, in this paper, voxel space deformation is applied to the next frame, but if we can apply voxel space deformation to the intra-frame, object shapes are reconstructed more accurately.

References

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

[2] Takashi Matsuyama and Xiaojun Wu and Takeshi Takai and Shohei Nobuhara: "Real-time generation and high fidelity visualization of 3d video", Proc. of MIRAGE2003, pp. 1–10, March, 2003.

[3] Joel Carranza and Christian Theobalt and Marcus A. Magnor and Hans-Peter Seidel: "Free-viewpoint video of human actors", ACM Transaction on Graphics, Vol. 22, No. 3, pp. 569–577, July, 2003.

[4] M. Gross and S. Wurmlin and M. Naef and E. Lamboray and C.Spagno and A.Kunz and E. Koller-Meier and T. Svoboda Luc Van Gool and S. Lang and K. Strehlke and A. Vande Moere and O, Staad: "blue-c: A Spatially Immersive Display and 3D Video Portal for Telepresence", Proc. of ACM SIGGRAPH2003, pp. 819–827, 2003.

[5] O. Grau and T. Pullen and G. A. Thomas: "A Combined Studio Production System for 3D Capturing of Live Action and Immersive Actor Feedback", IEEE Transactions on Circuits and Systems for Video Technology, Vol. 14, No. 3, pp. 370–380, March, 2004.

[6] W. N. Martin and J. K. Aggarwal: "Volumetric description of objects from multiple views", IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 5, No. 2, pp. 150–158, 1983.

[7] Yukiko Kenmochi and Kazunori Kotani and Atsushi Imiya: "Marching cubes method with connectivity", Proc. on International Conference on Image Processing, Vol. 4, pp. 361–365, Oct, 1999.

[8] Megumu Ueda and Daisaku Arita and Rin-ichiro Taniguchi: "Real-Time Free-Viewpoint Video Generation Using Multiple Cameras and a PC-Cluster", Proc. of PCM2004, pp. 418–425, Dec, 2004.

[9] Roger Y. Tsai: "A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf tv cameras and lenses", IEEE Transaction on Robotics and Automation, Vol. 3, No. 4, pp. 323–344, 1987.