

## RPV-II: A Stream-Based Real-Time Parallel Vision System and Its Application to Real-Time Volume Reconstruction

Arita, Daisaku  
Department of Intelligent Systems, Kyushu University

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

<https://hdl.handle.net/2324/5667>

---

出版情報 : Proceedings of Second International Workshop on Computer Vision System, pp.174-189,  
2001-07

バージョン :

権利関係 :

# RPV-II: A Stream-Based Real-Time Parallel Vision System and Its Application to Real-Time Volume Reconstruction

Daisaku Arita and Rin-ichiro Taniguchi

Department of Intelligent Systems, Kyushu University  
6-1 Kasuga-koen, Kasuga, Fukuoka 816-8580 Japan  
{arita,rin}@limu.is.kyushu-u.ac.jp  
<http://limu.is.kyushu-u.ac.jp/>

**Abstract.** In this paper, we present RPV-II, a stream-based real-time parallel image processing environment on distributed parallel computers, or PC-cluster, and its performance evaluation using a realistic application. The system is based on our previous PC-cluster system for real-time image processing and computer vision, and is designed to overcome the problems of our previous system, one of which is long latency when we use pipelined structures. This becomes a serious problem when we apply the system to interactive applications. To make the latency shorter, we have introduced stream data transfer, or fine grained data transfer, to RPV-II. One frame data is divided into small elements such as pixels, lines and voxels, and we have developed efficient real-time data transfer mechanism of those. Using RPV-II we have developed a real-time volume reconstruction system by visual volume intersection method, and we have measured the system performance. Experimental results show better performance than that of our previous system, RPV.

## 1 Introduction

Recently, especially in computer vision community, image analysis using multiple cameras has been extensively researched[1,2]. When we use multiple cameras, distributed systems are indispensable because a single or centralized system can not handle a large amount of image data[3,4,5,6,7]. When we require real-time analysis of those images, the problem becomes much more serious because of their large bandwidth and their huge computation demand. To solve the problems, we have developed a parallel/distributed real-time image processing system, RPV (Real-time Parallel Vision), on a PC-cluster, which consists of multiple off-the-shelf PCs connected via very high speed network[3,4]. The PC-cluster strategy has an important merit of scalability, which means that putting additional PCs into the network we can easily acquire larger number of sensors, or cameras, and larger computation power with low cost. The key issues of such distributed real-time image processing systems are synchronization among distributed PCs, the performance of network, the end-to-end latency at user level, the programming framework and the cost of the system.

The biggest problem of RPV is its long latency when we employ pipelined structures to acquire higher throughput. This is because, for real-time data transfer in RPV, we have developed synchronization mechanism based on the timing of image frame in an video sequence, and, as a result, the latency becomes  $2N \times \text{one frame period}$ , when we use  $N$  PCs arranged in a pipeline structure. This becomes a serious problem when we apply the system to interactive applications.

To make the latency smaller, we have introduced stream data transfer, or fine grained data transfer, to RPV-II, the next version of RPV. One frame data is divided into small elements such as pixels, lines and voxels. When each PC finishes processing an element, it starts sending it to the succeeding PC and then starts processing the next element. This mechanism can make the latency  $1/M$ , where  $M$  is the number of elements in one frame period.

Using RPV-II, as a realistic application, we have developed real-time volume reconstruction system by visual volume intersection method[8], and we have measured the system performance. The main part of this system consists of three steps: silhouette extraction, visual cone generation and volume construction by intersecting the visual cones. Each of three steps are executed by multiple PCs and voxel data representing the visual cones and the object shape are transferred in stream among the PCs. In this paper, at first, we overview RPV and outline RPV-II emphasizing on its stream data transfer. Then we describe real-time volume reconstruction using RPV-II, and, finally, the performance evaluation based on the developed application.

## 2 Real-Time Parallel Vision

### 2.1 Overview

Our PC-cluster system consists of 14 PCs, each of which has Pentium-III $\times 2$ (see Fig 1 and Fig. 2). All the PCs are connected via Myrinet, a crossbar-switch based gigabit network, and six of them have IEEE1394-based digital cameras[9], each of which can capture an uncompressed image sequence in real-time. On our PC-cluster, we support the following parallel processing schemes and their combinations. From the viewpoint of program structure, each PC corresponds to a component of a structured program of image processing.

**Pipeline parallel processing.** As shown in Fig. 3(a), the whole procedure is divided into sub-functions, and each sub-function is executed on a different PC sequentially.

**Data parallel processing.** As shown in Fig. 3(b), a set of data is divided into sub-data, and each sub-data is processed on a different PC in parallel. There are two approaches in data parallel processing:

- Image is divided into sub-images, and each sub-image is processed on a different PC in parallel.

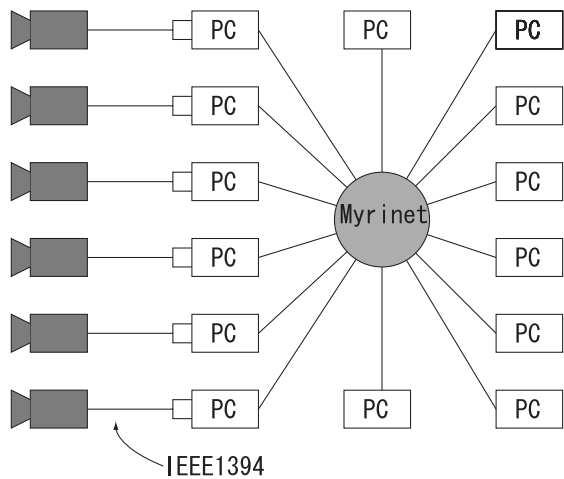
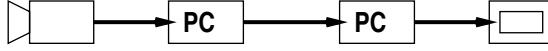


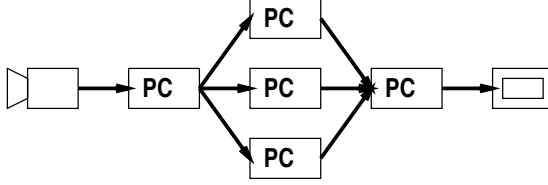
Fig. 1. Configuration of PC-cluster



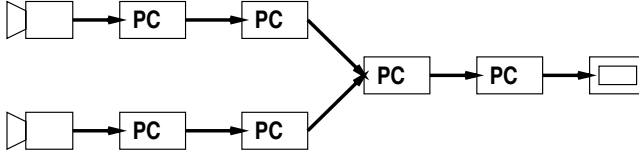
Fig. 2. PC-cluster



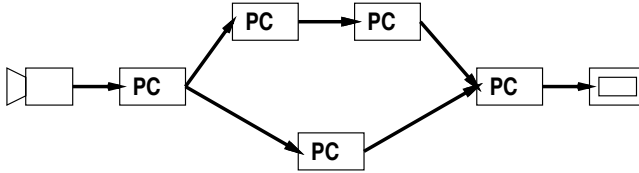
(a) Pipeline parallel processing



(b) Data parallel processing



(c) Data gathering



(d) Function parallel processing

**Fig. 3.** Parallel processing schemes

- Different frames are processed on different PCs. The longer<sup>1</sup> processing time can be secured, but the latency also becomes longer.

**Data gathering.** As shown in Fig. 3(c), images captured by multiple cameras are processed by PCs and integrated on the succeeding processing stage.

<sup>1</sup> It is proportion to the number of pipelined PCs. When  $N$  PCs are used the secured processing time is up to  $N \times \text{one frame period}$ .

**Function parallel processing.** As shown in Fig. 3(d), images are multicast to multiple PCs, on which different procedures are executed in parallel, and their results are unified in the succeeding processing stage.

## 2.2 Mechanisms for Real-Time Parallel Processing

For supporting the real-time parallel processing schemes described before, the system must have such mechanism as follows[3].

**Real-time data transfer.** We have used PM library[10] as communication primitives on Myrinet, and its actual network bandwidth is enough high to transfer image data in real-time. Since PM library supports DMA transfer between a main memory and a buffer on Myrinet board, it is able to reduce waste of CPU power for data transfer.

**Synchronization.** To inform all PCs when they start processing a frame data, we introduce Frame Synchronization Signal (FSS), which notifies the time to start processing, and which is received by all the PCs.

**Error recovery.** If data processing has not been finished or data receiving has not been finished when FSS is received by a PC, it is necessary to invoke an error recovery function. According to the image processing algorithm in each PC, programmers can select an error recovery mode from data missing mode, incomplete data mode or complete queuing mode. In data missing mode, only complete data are sent to the following PCs but some data are lost. In incomplete data mode, no data are lost, but incomplete data, which are intermediate result data, previous frame data or default data, are sometimes sent. In complete queuing mode, no data are lost and all data are completely processed, but the latency of data increases and a certain size of queue is required.

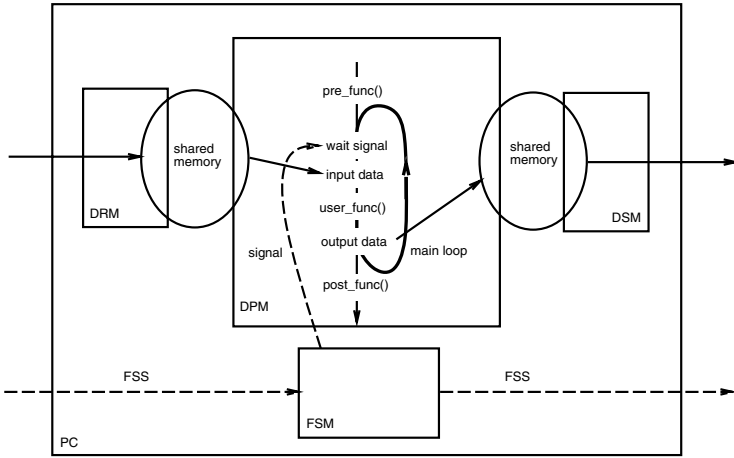
## 2.3 Implementation

In each PC, the following four modules are running concurrently to realize the mechanisms described before(see Fig. 4). Each of the modules is implemented as a UNIX process.

**Data Processing Module(DPM).** This module is the main part of the image processing algorithms, and is to process data input to the PC. It receives data from a DRM and sends data to a DSM via UNIX shared memory.

In DPM, any programs should consist of the following three elements:

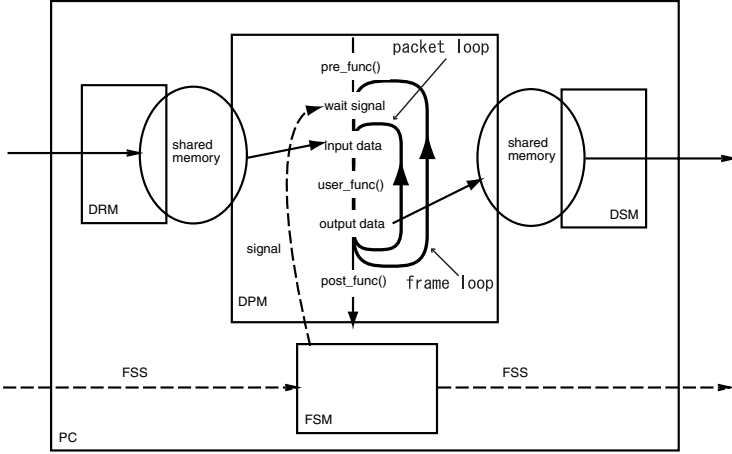
1. A main loop is to process image sequence, in which one iteration is executed per one frame time. The main loop is executed according to the following procedure to process image sequence continuously.
  - a) Wait for a signal from FSM to start processing. If a signal arrives before processing of previous frame is not finished, an error recovery function is invoked.



**Fig. 4.** Modules and functions in RPV

- b) Get input data. Actually pointers to input data are transferred in order to avoid copying data. If input data has not been received, an error recovery function is invoked.
  - c) Execute a user-defined function representing one iteration of the main loop, which is named **user\_func** here. Function **user\_func** receives synchronous input data *I* and asynchronous input data *A* and sends output data *O*. Synchronous input data *I* are main streams of data, which originates from image capture cards and are transferred between PCs synchronously. They are synchronized at the beginning of function **user\_func** (described at previous step). Asynchronous input data *A* can be used for feedback data and irregular data in cooperative processing. They are not synchronized at the beginning of function **user\_func** and are reloaded with the newest ones at any timing a programmer indicates.
  - d) Put output data. Because output data are directly written to shared memory in order to avoid copying data, only a notification of write-done is sent to DSM.
2. Before entering the main loop, a pre-processing function is executed. It is named **pre\_func** here. Function **pre\_func** is a user-defined function, which is used, for example, to load data necessary for the main loop such as a background image and calibration parameters.
  3. After exiting the main loop, a post-processing function is executed. It is named **post\_func** here. Function **post\_func** is a user-defined function, which is used, for example, to save results.

**Data Receiving Module(DRM).** This module is to receive data from other PCs via messages, and has buffers for queuing data. When a data request demand arrives from its succeeding DPM, it returns pointers to data.



**Fig. 5.** Modules and functions in RPV-II

**Data Sending Module(DSM).** This module is to send data to other PCs via messages, and has buffers for queuing data. When processed data arrives from its preceding DPM, it sends the data to the succeeding PCs.

**Frame Synchronization Module(FSM).** To realize synchronization among PCs, FSS is introduced, which is a time quantum signal of 1/30 second cycle. FSS, which originates in an image capturing component, is transferred via Myrinet network from upper PCs to lower PCs of the data flow. FSM sends FSSs to the succeeding FSM, and/or receives FSSs from the preceding FSM. FSM also sends start signals to activate the DPM in the PC. This framework makes executions of different DPMs synchronize with one another.

### 3 Stream Data Transfer on RPV-II

RPV has been designed to achieve quite high throughput of real-time image processing. However, its biggest problem is the long latency when we employ pipelined structures to acquire higher throughput. This is because, for real-time data transfer in RPV, we have developed synchronization mechanism based on the timing of image frame in an video sequence. For each pipeline step, one frame period  $\tau$ , which is 33msec in case of NTSC-based camera signal, is allocated to process one image frame and one frame period is allocated to transfer one image frame to the succeeding PC (see Fig. 6 (a)). Therefore, when we use  $N$  PCs arranged in a pipeline structure, the latency becomes  $2N\tau$  ( $\tau$  is a frame period and is 33msec for NTSC-based camera signal). For example, 3 stage pipeline causes about 0.2 sec. Thus, the latency problem becomes a serious problem when we apply the system to interactive applications. However, in general it is not easy to solve the tradeoff between high throughput and low latency.



To solve this problem, i.e., to make the latency smaller keeping high throughput, we have introduced stream data transfer, or fine grained data transfer, to the next version of RPV, RPV-II. 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. Therefore, when the packet arrives at the succeeding PC, the processing of the packet can be started immediately. Fig. 5 shows the processing procedure implemented in RPV-II. The key difference is that, in RPV-II, function `user_func` is executed for every packet. This mechanism can greatly reduce the latency (see Fig. 6 (b)). Of course, the tradeoff between throughput and latency still remains. When the size of the packet becomes too small, the communication overhead exhibits apparently. Therefore, we have to decide the adequate size of the packet, with which reduction of the throughput is small.

Compared with the previous framework, stream data transfer mechanism has the following features.

- Keeping high throughput, in general, the latency can be drastically reduced, i.e.,  $1/M$ . It is  $2N\tau/M$ , instead of  $2N\tau$ , where  $M$  is the number of packets in one frame period.<sup>2</sup>
- In the succeeding PC, the packets should be processed in the order that the packets are received. When the data should be reordered or reorganized for processing, the latency for the reorganization is added.

## 4 Real-Time Volume Reconstruction

For evaluation of effectiveness of stream data transfer on RPV-II, we have developed a real-time volume reconstruction system on our PC-cluster. The volume of an object is reconstructed via visual cone intersection using multi-perspective view of the object. A visual cone is defined as a cone whose vertex is the view point and whose cross section coincide with the silhouette of the object (see Fig. 7). Since the object is included in the visual cone, intersecting visual cones constructed from multiple view points makes the volumetric data of the object.

### 4.1 System Configuration

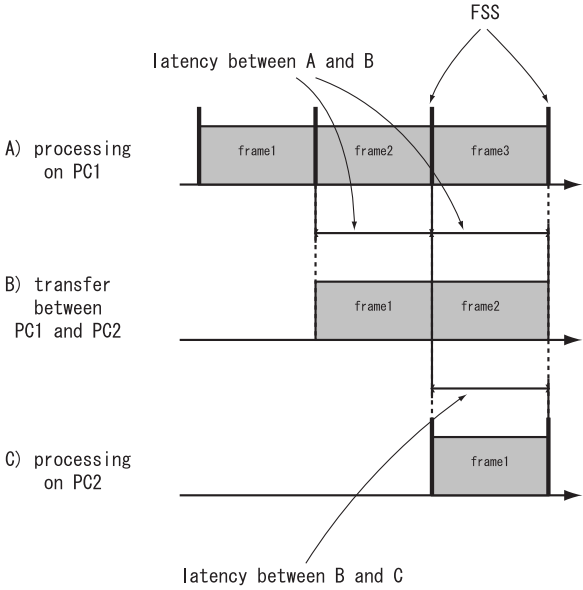
This method consists of four stages as follows (see Fig. 8).

#### 1. Extraction of object silhouette (EOS)

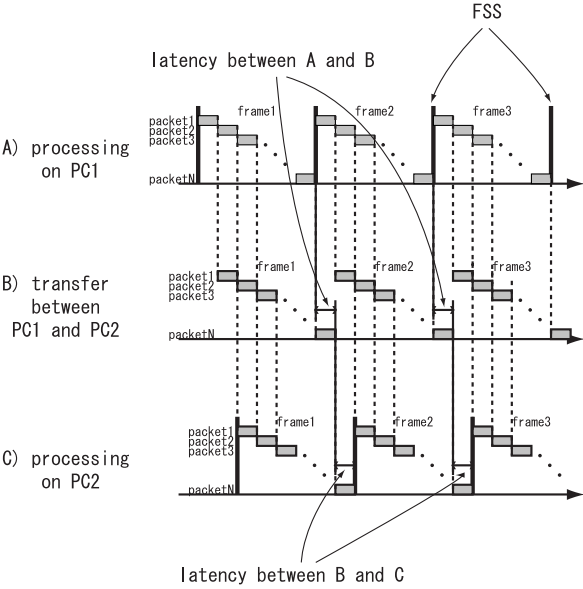
Background subtraction is employed to extract the silhouette of the object in each view. We use a simple method, i.e., thresholding of difference between pixel values of a captured image and those of a pre-acquired background image.

---

<sup>2</sup> Here, we assume that data transfer time is equal to data processing time.



(a) Synchronous (Frame-based) Data Transfer



(b) Stream Data Transfer

**Fig. 6.** Comparison of data transfer mechanism

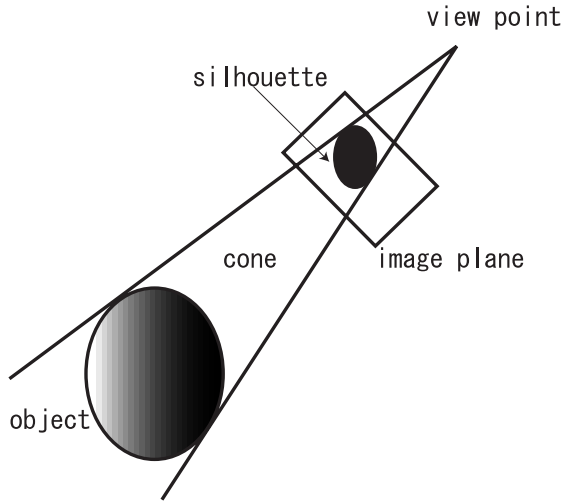


Fig. 7. Visual Cone

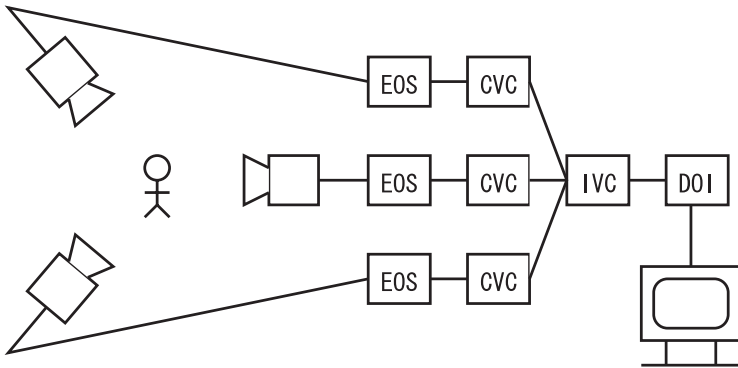


Fig. 8. System Configuration

Essentially a silhouette image can be calculated and sent to the succeeding PC in pixel-wise order, and therefore its latency should be  $\tau/M_p$ , where  $M_p$  is the number of packets of pixels in one frame image. However, in this system, an image captured by a camera is received frame by frame, not pixel by pixel, and the silhouette calculation can not be started after one complete image frame has been received. This inconsistency makes an additional latency  $\tau$ .<sup>3</sup> As a result, the latency caused in this stage becomes  $\tau + \tau/M_p$ .

## 2. Calculation of visual cone (CVC)

A visual cone represented in terms of voxels is constructed from the object silhouette of each view point. For each voxel, the system examines whether

<sup>3</sup> This can be easily reduced by modifying the camera interface with which image data can be acquired line by line.

a pixel corresponding to a voxel is included in the object silhouette or not<sup>4</sup>. This process consists of two subprocesses, voxel projection to a silhouette image and inclusion test on it. The voxel projection can be optimized by using a pre-computed lookup table representing correspondence between voxels and pixels. Using this lookup table improves the speed of the visual cone calculation drastically.

A silhouette image is received in a pixel-wise order, but visual cone construction is executed in a voxel-wise order. Because of this inconsistency the latency between data receiving and data processing becomes  $\tau$ . On the other hand, since voxel space representing visual cone is generated and sent in a voxel-wise order and there is no inconsistency, the latency between data processing and data sending is  $\tau/M_v$ , where  $M_v$  is the number of packets of voxels in one voxel space. Total latency at this stage becomes  $\tau + \tau/M_v$ .

### 3. Intersection of visual cones (IVC)

Visual cones from multiple view points are gathered and intersected to generate a volumetric data of the object represented in terms of voxels. Since receiving a voxel space, intersecting all voxel spaces and sending a generated voxel space are executed in a voxel-wise order, both the latency between data receiving and data processing and the one between data processing and data sending are  $\tau/M_v$ .

### 4. Display of object image (DOI)

Reconstructed volumetric data is projected to an image plane to generate an object image and the image is displayed on a screen.

Since receiving a voxel space and projecting it to an image plane is executed in a voxel-wise order, the latency between receiving and processing is  $\tau/M_v$ .

The total latency becomes  $\tau/M_p + 4\tau/M_v + 2\tau \simeq 2\tau$  (suppose  $M_p$  and  $M_v$  is large enough). It is much reduced compared with the case of frame-based data transfer where the total latency becomes  $7\tau$ . Again, the decline of the throughput is very little.

## 4.2 Performance Evaluation

First, Table 1 shows throughput, the number of frames which the volume reconstruction system process in one second. The amount of its computation is proportional to the number of voxels. In case that the number of packets is small, or the size of the packet is large, the throughput is in inverse proportion to the number of voxels, which is an ideal characteristic. However, in case that the number of packets is large, or the size of the packet is small, the throughput is not in inverse proportion to the number of voxels. This is because the overhead of data transfer is getting large. Since the maximum frame rate of our cameras 30 fps, the maximum throughput of the system becomes 30 fps.

Second, Table 2 shows latency. Throughput in this experiment is lower than that in previous one because of probing overhead. The results are nearly equal

<sup>4</sup> Though multiple pixels are usually corresponding to one voxel, only one pixel is selected for simplification and speed up.

**Table 1.** Throughput(fps)

# voxels	$M_v$ ( $M_p$ is fixed on 10)			
	125	1000	2500	5000
$40 \times 40 \times 40 = 64000$	30.00	30.00	30.00	30.00
$80 \times 80 \times 80 = 512000$	28.57	25.00	11.11	6.67
$100 \times 100 \times 100 = 1000000$	14.27	12.46	10.99	5.71

**Table 2.** Latency(msec)

stage	$M_v$ ( $M_p$ is fixed on 10)			
	125	1000	2500	5000
EOS	10.154	11.163	11.140	11.070
EOS-CVC	0.309	0.337	0.350	0.329
CVC	156.260	182.816	201.501	384.612
CVC-IVC	9.904	0.143	1.644	38.518
IVC	0.187	0.073	1.107	23.135
IVC-DOI	0.315	0.155	0.501	42.939
DOI	6.407	6.326	6.390	7.195
Total( $L$ )	183.536	201.013	222.696	507.793
Frame rate( $1/\tau$ )	12.5	11.1	10.0	4
Frame period( $\tau$ )	80.000	90.000	100.000	250.000
$L/\tau$ (experimental)	2.294	2.233	2.227	2.031
$L/\tau$ (designed)	2.132	2.104	2.102	2.101

to the designed values, which shows that the system works correctly. Assuming that the experimental values ( $L/\tau$ ) are effective in case of no probes, the latency  $L$  may be equal to  $1000/14.27 \times 2.294 = 160.8$  msec.

These experiments are preliminary ones and, because of the limitation of physical space and camera calibration, we have used three cameras. According to the experiments, for real-time volume reconstruction, the number of voxels should be around  $100 \times 100 \times 100$ . In this case, the throughput is about 14.27 fps and the latency is 184 msec with an optimal packet size. In other words, the number of packets in one frame is 125, or the size of the packet is 8000 bytes. The low latency exhibits the effectiveness of our method.

Fig. 9 shows input image sequences of three views and Fig. 10 shows re-constructed volumetric representation. Since we have only three cameras in this experiment, the reconstructed 3D shape is not very accurate. We are making large scaled experiments with more cameras, which can produce more precise 3D shape.



(a) Viewpoint 1

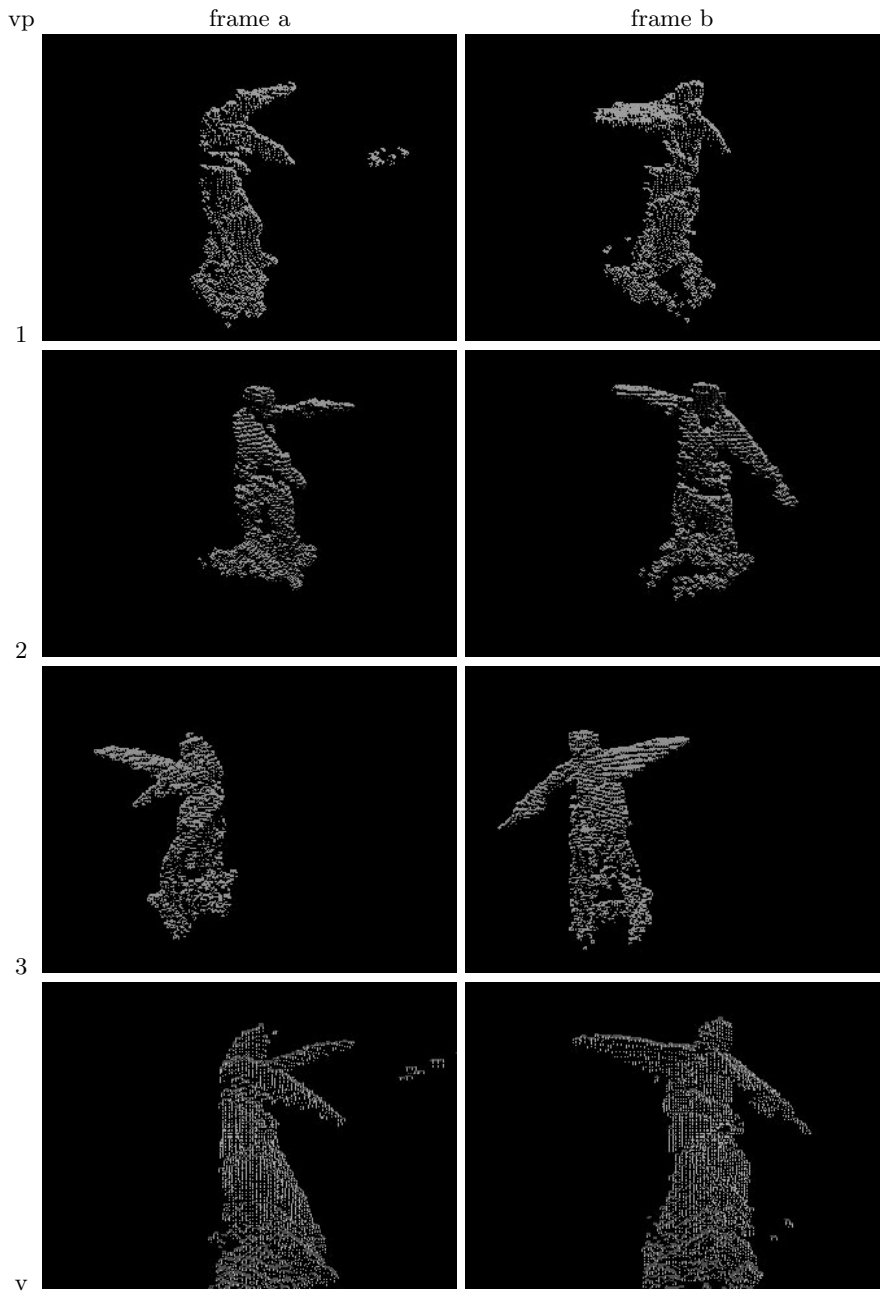


(b) Viewpoint 2



(c) Viewpoint 3

**Fig. 9.** Input Images



**Fig. 10.** Output Images: The first column is viewpoint. Viewpoint v means a virtual viewpoint. The second and the third columns are output images on different frames

## 5 Conclusion

In this paper, we have presented a stream-based real-time parallel vision system on PC-cluster and its application to a real-time volume reconstruction system. The motivation of this research is to develop a large-scaled real-time computer vision system with high performance, i.e., high throughput and low latency. The low latency is quite important especially for interactive application such as human interface systems. The key idea is that real-time data processing and transfer is established not on image frames but on finer grained data packets consisting of pixels, lines, voxels, or other image features. This enables the systems to have much less latency with little decrease of throughput, compared with ordinary frame-based real-time image processing and transfer. The effectiveness of our idea has been also shown by a realistic application of real-time volume reconstruction from multiple views. The throughput of the experimental system is about 14 fps for a  $100 \times 100 \times 100$  voxel space, and the latency is only about 160 msec, which can be improved by further refinement.

The future works remaining are as follows.

- Evaluation with larger scaled applications: we are constructing a 20 camera systems to acquire more precise volumetric representation.
- System construction with cheaper cost: we are developing a PC-cluster whose network is IEEE1394 bus, not Myrinet[11]. Evaluation of this system should be done, too.

**Acknowledgment.** This work has been supported by R&D activities in the info-communication area, Telecommunication Advancement Organization (TAO) of Japan(No.10080).

## References

1. T. Kanade, R. T. Collins, A. J. Lipton, P. Burt, and L. Wixson, "Cooperative multi-sensor video surveillance," in Proc. of Image Understanding Workshop, pp.3–10, 1997.
2. T. Matsuyama, "Cooperative Distributed Vision – Integration of Visual Perception, Action, and Communication –, " in Proc. of Image Understanding Workshop, pp.1–39, 1998.
3. D. Arita, Y. Hamada, and R. Taniguchi, "A Real-time Distributed Video Image Processing System on PC-cluster," in Proc. of Int. Conf. of the Austrian Center for Parallel Computation(ACPC), pp.296–305, 1999.
4. D. Arita, S. Yonemoto, and R. Taniguchi, "Real-time Computer Vision on PC-cluster and Its Application to Real-time Motion Capture," in Proc. of IEEE Workshop on Computer Architectures for Machine Perception, pp.205–214, 2000.
5. T. Kanade, H. Saito, and S. Vedula, "The 3D room: Digitizing time-varying 3D events by synchronized multiple video streams," in Technical Report, CMU-RI-TR-98-34, Carnegie Mellon University, Dec. 1998.



6. T. Wada, X. Wu, S. Tokai, and T. Matsuyama, "Homography Based Parallel Volume Intersection: Toward Real-time Volume Reconstruction Using Active Cameras," in Proc. of IEEE Workshop on Computer Architectures for Machine Perception, pp.331–339, 2000.
7. E. Borovikov, and L. Davis, "A Distributed System for Real-time Volume Reconstruction," in Proc. of IEEE Workshop on Computer Architectures for Machine Perception, pp.183–189, 2000.
8. W. N. Martin and J. K. Aggarwal, "Volumetric Description of Objects from Multiple Views," in IEEE Trans. PAMI, Vol.5, No.2, pp.150–158, 1987.
9. 1394 Trade Association, in 1394-based Digital Camera Specification Version 1.20.
10. H. Tezuka, A. Hori, Y. Ishikawa, and M. Sato, "PM: an operating system coordinated high performance communication library," High-Performance Computing and Networking, P. Sloot and B. Hertzberger, in Springer-Verlag, pp.708–717, 1997.
11. H. Yoshimoto, D. Arita and R. Taniguchi, "Real-Time Image Processing on IEEE1394-based PC Cluster," in CD-ROM Proc. of International Parallel & Distributed Processing Symposium, 2001.