PAPER *Special Issue on IEICE Style File*

High Speed 3D Reconstruction by Spatio-Temporal Division of Video Image Processing

Yoshinari KAMEDA[†], **Takeo TAODA**[††], *Nonmembers, and* **Michihiko MINOH**[†], *Member*

SUMMARY A high speed 3D shape reconstruction method with multiple video cameras and multiple computers on LAN is presented. The video cameras are set to surround the real 3D space where people exist. Reconstructed 3D space is displayed in voxel format and users can see the space from any viewpoint with a VR viewer. We implemented a prototype system that can work out the 3D reconstruction with the speed of $10.55fps$ in $313ms$ delay.

key words: 3D shape reconstruction, Voxel, Real-time image processing, Cooperative distributed vision

1. Introduction

With improvement in processing speed of computers and with increase of network bandwidth, it may come true to synchronize a virtual space in computers with a real 3D space[1]. Our final goal is to construct a real-time virtual space which displays human activities in a certain real space by using multiple computers distributed on LAN. Once the virtual space is constructed, anyone outside the real space can observe the human activities in the real space from any viewpoint. To synchronize the virtual space with the real space, shape of the real space should be reconstructed in realtime at first.

Slit light projection methods and structured light projection methods achieve real-time 3D shape reconstruction, but these methods require active sensing[2] which may limit human activities in the real space. On the contrary, passive vision based approaches[3] do not affect the activities. Stereo vision methods achieve real-time 3D reconstruction. However, they cannot reconstruct backside shapes that cannot be seen by stereo cameras. So it is not available to provide VR data, which are viewed from various virtual viewpoints. In this sense, the cameras have to be placed so as to surround the real space. Realistic 3D reconstruction methods[4][5] based on the stereo vision have been proposed which use over ten cameras, but that need certain period to reconstruct one scene and are not suitable for real-time applications.

Multi-camera based 3D reconstruction methods have been proposed with voxel representation. The fundamental idea was shown in [6] with orthogonal projection. Then perspective projection based methods were proposed[7][8]. They determine the shape of one object by intersecting conic volumes which are defined by the focal point of the cameras and silhouettes of the object on their image planes.

The main problem of 3D reconstruction with such camera surrounding layout is a large amount of calculation time because of many images for taking one scene. Relating to 3D reconstruction of solid objects, an octree based approach[9] succeeded in reducing the calculation by realizing fast inclusion testing of octree nodes on the silhouette. A computational geometric approach[10] reduced it by considering the stacked planes on intersecting the conic volumes. They mentioned the applicability of parallel processing, but did not show the methods to do it.

In this paper, we proposed the fast 3D reconstruction method which is suitable to use distributed computers and discuss the performance of our prototype system. Our 3D reconstruction method called VFM is basically same as that proposed by [7][8], but our method eliminates the existence of static objects and achieves less computation.

We reconstruct the shape of the real space by preparing one computer for each camera to execute image processing, and other computers to calculate 3D reconstruction. All the computers are connected with 100baseT Ethernet and 155Mbps ATM LAN.

In our method, we decrease latency by dividing a real 3D space into some subspaces and reconstructing the subspaces simultaneously at several distributed computers and we improve throughput by dividing video processing into some stages. We can also control throughput and latency by changing the process arrangement in the system and satisfy the requirements of the applications.

In the following sections, Section 2 describes how to reconstruct 3D scene in this method, and in Section 3 we explain the prototype system named SCRAPER and show experimental results. We conclude this paper in Section 4.

2. 3D Reconstruction Method

The reconstruction algorithm has to be suitable for distributed computing. Therefore, the algorithm should

Manuscript received January 1, 2000.
Manuscript revised January 1, 2001.

[†]Center for Information and Multimedia Studies, Kyoto University
[††]Graduate School of Engineering, Kyoto University

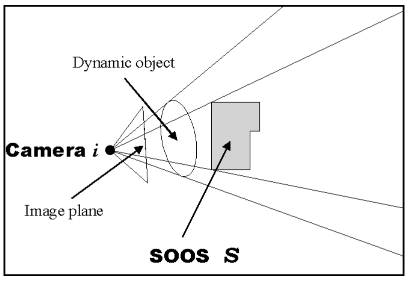

Fig. 1 SOOS $\mathcal{S}$

have the following two characteristics.

- It is possible to equalize processing time of each process by dividing calculation procedure and data.
- Amount of communication among processes is as little as possible.

The method of intersecting conic volumes proposed by [7][8] can be extended to satisfy the two characteristics. Moreover, the space occupied by the static objects should be eliminated to reduce the calculation amount in the 3D reconstruction algorithm.

In this section we propose the improved method called Viewing Frustum Method (VFM) and explain its detail.

We decompose the procedure of the VFM so that it satisfies these two characteristics. We will explain the way of the decomposition in this section.

With VFM, we reconstruct the real space in real-time by generating voxel data from several images taken at the same time. We call the part of the real space which can be imaged by the cameras *target space.*

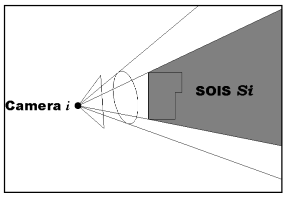

Fig. 2 SOIS $\mathcal{S}_i$

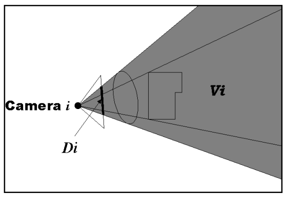

Fig. 3 Viewing frustum $\mathcal{V}_i$

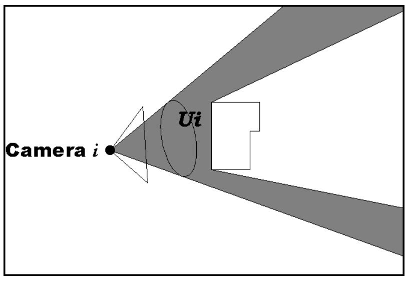

Fig. 4 ESS $\mathcal{U}_i$

2.1 Static Object Occupation Subspace (SOOS)

Since our objective is to reconstruct changing shape in the target space where people work, it is reasonable to have knowledge of static objects in the space in advance. As the static objects do not change their locations and shapes during human activities, we can exclude the subspace where the static objects occupy. We call the subspace *Static Object Occupation Subspace (SOOS)* denoted by $\mathcal{S}$.

When we see the target space from the viewpoint of camera i, some subspaces cannot be seen because $\mathcal{S}$ occludes them. We merge $\mathcal{S}$ and these occluded subspaces together and call it *Static Object Influence Subspace (SOIS)* $\mathcal{S}_i$. Generally $\mathcal{S}_i$ is different from $\mathcal{S}_j$ $(i \neq j)$ because the locations of the cameras are different. Fig. 1 and Fig. 2 show the relation between $\mathcal{S}$ and $\mathcal{S}_i$ at camera i. The figures are drawn in 2D for simplicity. Consider the grey region as a static object in Fig. 1.

From now on, we concentrate on reconstructing the voxels which represent dynamic objects observed in the target space. The dynamic object is defined as object whose shape and location information is not known in advance.

2.2 3D Reconstruction

When the dynamic objects are imaged by a camera i, they exist within frustums that circumscribe their projected regions on the image and whose apexes are the focus location of the camera i. We call all the projected regions, on the image of the camera i together, *dynamic region* D_i and denote subspace including corresponding viewing frustums by $\mathcal{V}_i$. D_i is described in the binary image format. In Fig. 3, the grey region represents $\mathcal{V}_i$.

As the dynamic objects can exist only outside $\mathcal{S}$ and can be seen outside $\mathcal{S}_i$, we only care a subspace named *Existence Shadow Subspace (ESS)* $\mathcal{U}_i$ defined by Equation (1). The grey region in Fig. 4 represents

$\mathcal{U}_i$.

$$\mathcal{U}_i = \mathcal{V}_i \cap \overline{\mathcal{S}_i} \qquad (1)$$

The dynamic objects ought to be observed somewhere inside $\mathcal{U}_i$.

In the case where the dynamic objects are imaged by n cameras, the shape of the objects can be given by the intersection of all of $\mathcal{U}_i (i = 1, \cdots, n)$. We denote this intersected subspace as $\mathcal{U}$ and define it by

$$\mathcal{U} = \bigcap_{i=1}^{n} \mathcal{U}_i \qquad (2)$$

The reconstructed shape of $\mathcal{U}$ is approximation of that of the dynamic objects in the real space. The approximation becomes better as we increase the number of cameras.

With the voxel representation, each voxel which is inside the target space but not included by $\mathcal{S}_i$ is projected onto the image plane of camera i for inclusion test. If the voxel is projected inside D_i for all the cameras, it forms the shape of the dynamic objects.

2.3 Spatial Division

Let us decompose the method in spatial sense so as to improve the latency of the 3D reconstruction calculation. Suppose there are n cameras in the real space and the focus locations of the cameras are given in advance. The cameras capture images of the space simultaneously. We call this set of images a *frame set*. 3D reconstruction process can calculate $\mathcal{U}$ at each frame set just by receiving $D_1, D_2, \cdots$, and D_n.

In order to calculate whether 3D point p is included by $\mathcal{U}$ or not, information of $\mathcal{S}$, the focus locations of the cameras, and $D_1, D_2, \cdots, D_n$ are needed. Since all these values can be given before the calculation starts and each calculation for 3D point p_k does not affect other calculations of 3D point $p_l (l \neq k)$, we can execute the calculations for different 3D points simultaneously. This is the calculation locality feature of our method.

Therefore the 3D reconstruction calculation in Equation (2) is easily decomposed for parallel distributed computing. Thus we achieve spatio-division of the 3D reconstruction process based on the calculation locality of 3D reconstruction.

2.4 Temporal Division

We have presented the way of decomposing the VFM spatially in the previous section. In addition, we can split the 3D reconstruction calculation along time axis so as to process as many frame sets as possible per second and hence improve the throughput of the 3D reconstruction system. The temporal division is achieved by splitting the VFM into three stages. Let us call this sequence of stages a *path*.

1. Image capture
2. Extraction of dynamic region
3. ESS calculation

We prepare three kinds of processes for each stage: *image captor*, *extractor*, and *3D composer*.

As the most time consuming stage is the ESS calculation done by 3D composers, multiple 3D composers should be prepared to use multiple paths. 3D composers in one path receive the data and produce $\mathcal{U}$ in a certain time span while other 3D composers in different path receive the data and start calculation.

2.5 Process Arrangement

The number of 3D composers can be changed independently with the number of the cameras because the calculation on the 3D composer is completely spatio-temporally localized. Therefore, we can improve throughput and latency of the system by preparing the 3D composers on different workstations distributed on LAN.

As the temporal division method and the spatial division method do not affect each other, both methods can be adopted in the same system. We can obtain desired latency and throughput by changing the process arrangement.

Fig. 5 shows the process timing chart when the system has four cameras and four 3D composers are prepared. We assign two 3D composers at two paths. In the figure, dark grey cells indicate image captors, and light grey ones indicate extractors. Long shape cells are 3D composers where process A and B are used for odd-number frame set, and, C and D for even-number frame set. A *cycle* consists of two periods, one is execution time T_{exe} and the other is transmission time T_{tr}. In the execution time T_{exe}, all the processes are executing calculation without data transmission among them. On the other hand, while it is in the transmission time T_{tr}, the processes exchange their data via LAN except for the 3D composers which are under calculation and do not need to exchange the data.

The lines between the processes in Fig. 5 represent data connections during the transmission time T_{tr}.

We introduce a process named *scheduler* to synchronize the processes.

3. Experimental Results and Discussion

We implemented a prototype system named SCRAPER according to the method proposed in this paper. We experimentally reconstructed a part of a lecture room in the graduate school of informatics in Kyoto University.

The target space is imaged by four SONY EVI-G20 video cameras fixed at the corners of the lecture

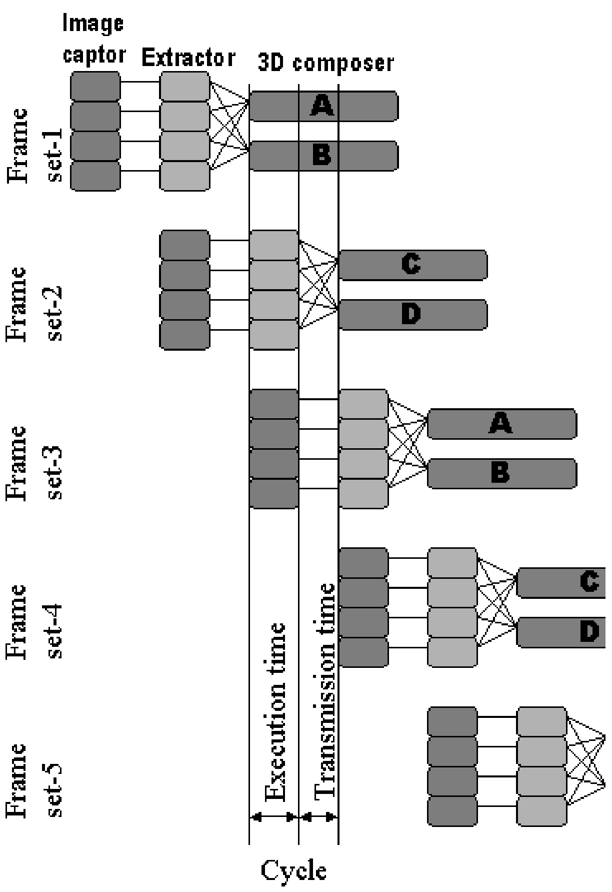

Fig. 5 Timing chart of processes

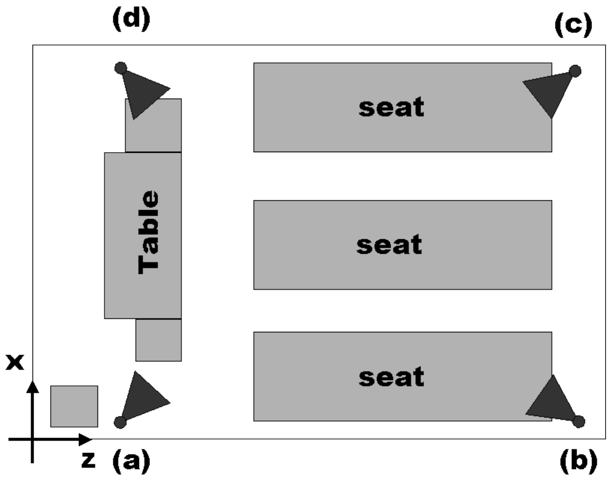

Fig. 6 Camera layout in the lecture room

room (Fig. 6). Table 1 shows the camera location in the room-coordinate system.

In the experiment, we configured four image captors and four extractors, and four 3D composers. Four SUN Ultra2 workstations (296MHz with 2 CPU) are used and each workstation has a video capture card.

In our prototype system, one image captor and one extractor are assigned to each workstation. This is because an image is captured by the video capture card for which CPU power is not necessary and an extractor extracts $\mathcal{D}_i$ by detecting regions where the pixel values differ from the background image taken beforehand for which it needs CPU power. The two processes need only one CPU to work together.

Table 1 Camera position

Camera ID	X [m]	Y [m]	Z [m]
(a)	0.70	2.80	1.65
(b)	0.53	2.80	10.41
(c)	6.45	2.80	10.42
(d)	6.47	2.77	1.64

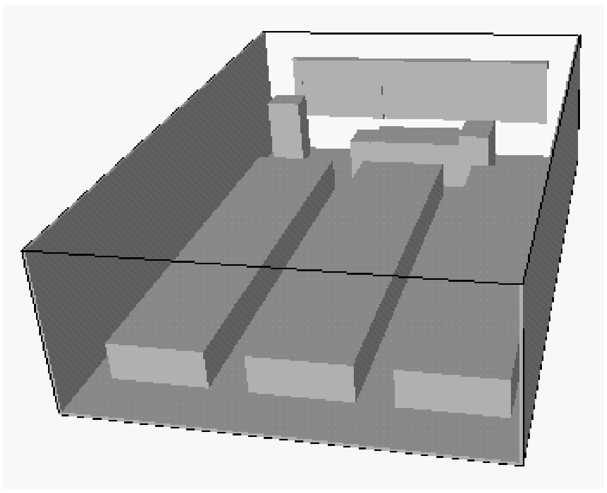

Fig. 7 SOOS

The 3D composers are also assigned to the same four workstations. The workstations have two network interfaces, one is 100base-T and the other is 155Mbps ATM. The dynamic region data from the extractors to the 3D composers are transmitted via ATM LAN whereas the synchronization signals are sent via 100base-T LAN. The scheduler is placed on SUN Ultra30 workstation (296MHz) which is connected to 100base-T LAN. As EVI-G20 does not have a frame synchronization mechanism, the scheduler checks the captured time of each images and allows the extractors and the 3D composers to process only when all the images in the frame set are captured within a certain period.

Fig. 7 shows SOOS defined by the static object database given in advance. A SOIS from the camera (a) in Fig. 6 is shown in Fig. 8 for example. These subspace have been calculated before the SCRAPER system starts the reconstruction. As the camera (a) is located at the corner of the room in Fig. 8, you can see the slanted subspace around the tables occluded by the static objects at the opposite side of the camera (a).

We set the target space which was imaged more than three cameras. Hence, a part of the target space

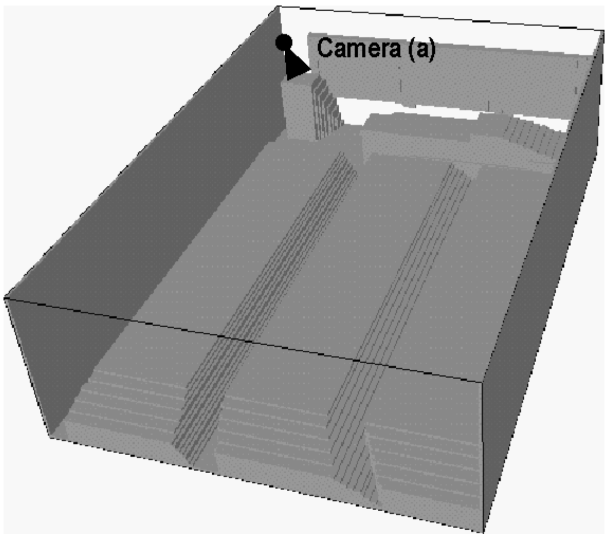

Fig. 8 SOIS of camera (a)

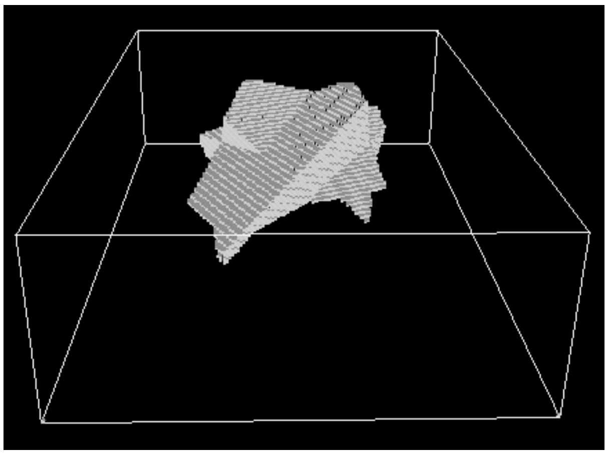

Fig. 9 Target space

is observed by four cameras, and the other part is observed by three cameras. In the subspace where all the four cameras image, n in Equation (2) should be four, and in the subspace which a camera j could not observe, n does not contain j at the calculation of Equation (2). Fig. 9 displays the target space which is visible by at least three cameras in the lecture room.

In the experiment, the image captor takes images with the size of 320×240 pixels. The camera which locates the furthest position from the target space images a cubic subspace of 5 centimeters on a side in the target space onto one pixel in the captured image. Therefore, we set the voxel size as a cube of 5 centimeters on a side. The target space shown in Fig. 9 corresponds to 96,769 voxels.

We conducted an experiment to measure the throughput and the latency of our prototype system. The target space is shown in Fig. 9, and we put a box as a dynamic object whose size is $60cm \times 55cm \times 30cm$. The result of using four 3D composers are shown in Table 2. A variable t indicates number of paths in the system and so it means number of temporal division of the target space. On the other hand, s indicates number of 3D composers served in each path and so it means number of spatial division of the target space. We also conducted an experiment with only one 3D composer just for comparison and its throughput is 3.92 fps and its latency is 746 $msec$. These results are averaged result of 1,000 frame sets.

Table 2 Throughput and latency

Number of paths : t	1	2	4	1
3D comp. per path : s	4	2	1	1
Latency [msec]	293	313	418	746
Throughput [fps]	8.78	10.55	11.23	3.92

Table 3 shows the time which the processes consumed. We call the time consumed at an extractor T_{ext} and the time at a 3D composer T_{cmp}. The result of T_{cmp} proved that the spatial division of our method could shorten the calculation time in proportion to s.

Table 3 Process time

3D comp. assignment (t/s)	1/4	2/2	4/1	1/1
T_{ext} (Extractor)[msec]	29.3	28.9	28.8	26.5
T_{cmp} (3D comp.)[msec]	64.9	123.2	239.9	236.0

The times the extractor consumes in Table 3 are to be the same ideally, but inevitable light change and video signal noise made difference to some extent.

Table 4 shows both the execution time T_{exe} and the transmission time T_{tr} during the 3D reconstruction calculation in the experiment. Each value of the execution time is longer than that in Table 3 because one execution cycle does not end until all the processes finish.

Table 4 Time per cycle

3D comp. assignment (t/s)	1/4	2/2	4/1	1/1
T_{exe}[msec]	86.3	72.2	69.3	236.1
T_{tr}[msec]	27.5	22.5	19.8	18.8

The size of image data from the image captors to the extractors is $320 \times 240 \times 4 = 307,200$ bytes per image. The size of the dynamic region information is at most $320 \times 240/8 = 9,600$ bytes. Since the size will be shrunk when the size of the detected dynamic region is small, it is smaller in most cases. In our prototype system, the former data is transmitted inside the workstation because the image captors and the extractors are located in the same workstation, while the latter data is transmitted via ATM LAN.

In Table 4, the transmission time becomes shorter as we increase t and decrease s because the mount of data transmission become smaller. Let us consider the number of data connections between the processes which are shown in Table 5. The first figure shows the number of overall connections in the system at a time and a figure in parenthesis indicates the number of connections received by one process.

If you plan to suppress the transmission time, it is better to set s smaller. In this case the latency becomes longer as side effect.

Table 5 Number of data connections

3D comp. assignment (t/s)	1/4	2/2	4/1	1/1
I.C. to Ext.	4 (1)	4 (1)	4 (1)	4 (1)
Ext. to 3D comp.	16 (4)	8 (4)	4 (4)	4 (1)

By dividing 3D reconstruction procedure spatially with larger s, T_{exe} will be worse because the end of the execution time is determined by the latest 3D composer process at that cycle among all the synchronized processes the number of which becomes larger in proportion to s. There are also image captors and extractors which are to be synchronized. As they consume much smaller time than the 3D composers, they can be ignored.

Apart from the values of the throughput and the latency, we estimated the CPU working rate while it is assigned to 3D composers because CPU is one of the most limited resource. This can be estimated by

$$R_{cmp}(t) = \frac{T_{cmp}}{t * T_{exe} + (t-1) * T_{tr}} \tag{3}$$

The denominator indicates the averaged overall time in the system on executing calculation of 3D composer. After the calculation is finished in T_{cmp}, the CPUs assigned to the 3D composers are idle for the time $t * T_{exe} + (t-1) * T_{tr} - T_{cmp}$. Hence $R_{cmp}(t)$ should be close to 1.0 in the viewpoint of CPU resource.

We estimated $R_{cmp}(t)$ in the experiment and the result is shown in Table 6. From the result with our prototype system, we can say that smaller t (and larger s) is better.

Table 6 CPU rate

3D comp. assignment (t/s)	1/4	2/2	4/1	1/1
$R_{cmp}(t)$	0.751	0.738	0.713	1.000

Generally speaking, required throughput and latency differ according to applications. One good feature of our method is that we can adopt the process arrangement suitable to the applications by changing t and s. The result indicates that the case of two 3D composers at two paths is good because the throughput is almost same as that of four paths and the latency is as short as that of four 3D composers.

We implemented a virtual space viewer which displays the reconstructed real space as a set of voxels in realtime. This viewer displays not only the dynamic objects but also the static objects given to the system in advance, so a user can walk around the lecture room and observe the reconstructed space from any viewpoint with little delay.

An example of a captured image is shown in Fig. 10. Fig. 11 shows the reconstructed space displayed by the viewer. The voxels displayed in the center corresponds to $\mathcal{U}$, which were transmitted from the SCRAPER system.

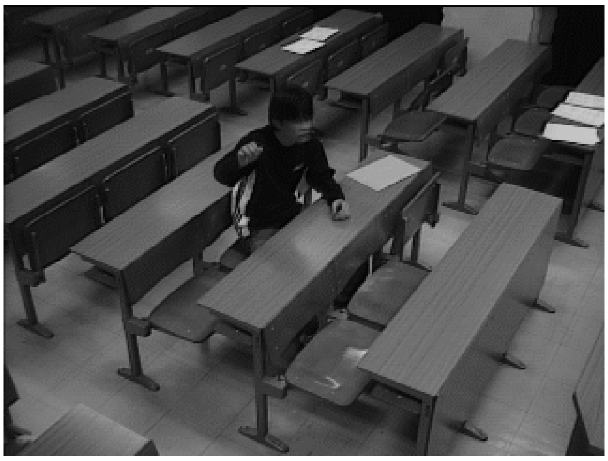

Fig. 10 Video image from camera (d)

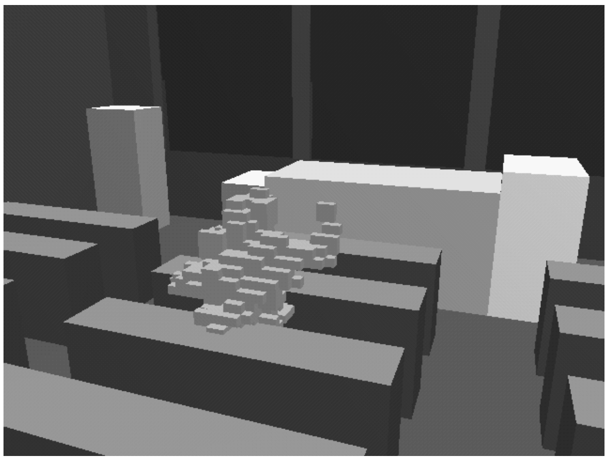

Fig. 11 Reconstructed space from other viewpoint

With this prototype system, it is possible to estimate the pose of students and a lecturer, such as standing, sitting, raising their hands, and so on from the reconstructed voxel data. Although the current resolution of the voxel data is not so good, it is able to locate the persons and their heads in the 3D world. We are planning to apply our method so that active cameras in the scene can shoot the head of the people precisely in our other project[11].

4. Conclusion

We have presented the method of high speed 3D reconstruction in the situation multiple cameras surround a certain real space. We showed that our approach introducing the spatial and temporal division of the VFM improves the throughput and the latency of the 3D reconstruction calculation. With four 3D composers, SCRAPER achieved $10.55fps$ with $313ms$ delay by using two paths and assigning two 3D composers at each path.

We would like to extract the motion and the pose information from the reconstructed human shape in the future research.

Acknowledgement

This work has been partly supported by "Cooperative Distributed Vision for Dynamic Three Dimensional Scene Understanding (CDV)" project (JSPS-RFTF96P00501, Research for the Future Program, the Japan Society for the Promotion of Science).

And we appreciated Hirofumi Aoki in our laboratory for his cooperation on conducting the experiment.

References

[1] R.Raskar, G.Welch, M.Cutts,A.Lake,L.Stesin, and H.Fuchs, "The Office of the Future: A Unified Approach to Image Based Modeling and Spatially Immersive Displays," SIGGRAPH98 Annual Proc., pp.179-188, 1998.

[2] K.Sato, A.Yokoyama, and S.Inokuchi, "Silicon range finder- a realtime range finding VLSI sensor," Proc. IEEE 1994 CIC, pp.339-342, 1994.

[3] T.Kanade, A.Yoshida, K.Oda, H.Kano, and M.Tanaka, "A Stereo Machine for Video-rate Dense Depth Mapping And Its New Applications," Proc. CVPR, pp.196-202, 1996.

[4] P.J.Narayanan, P.W.Rander, and T.Kanade, "Constructing Virtual Worlds Using Dense Stereo," Proc. Sixth IEEE ICCV, pp.3-10, 1998.

[5] J.E.Boyd, E.Hunter, P.H.Kelly, T.Li-Cheng, C.B.Phillips, and R.C.Jain, "MPI-Video infrastructure for dynamic environments," Proc. IEEE ICMCS, pp.249-254, 1998.

[6] W.N.Martin and J.K.Aggarwal, "Volumetric Descriptions of Objects from Multiple Views," Proc. IEEE PAMI, Vol.5, No.2, pp.150–158, 1983.

[7] M.Potmesil, "Generating Octree Models of 3D Objects from Their Silhouettes in a Sequence of Images," CVGIP, Vol.40, pp.1-29, 1987.

[8] S.K.Srivastava and N.Ahuja, "Octree Generation from Object Silhouettes in Perspective Views," CVGIP, Vol.49, pp.68-84, 1990.

[9] R.Szeliski, "Rapid Octree Construction from Image Sequences," CVGIP, Vol.58, No.1, pp.23-32, 1993.

[10] P.Srinivasan, P.Liang, and S.Hackwood, "Computational Geometric Methods in Volumetric Intersection for 3D Reconstruction," PR, Vol.23, No.8, pp.842-857, 1990.

[11] Y.Kameda, H.Miyazaki, and M.Minoh, "A Live Video Imaging for Multiple Users," IEEE ICMCS 1999, Vol.2, pp.897–902, 1999.

Kameda, Yoshinari He received the B.E., M.E., and D.E. from the Kyoto University in 1992, 1994 and 1999. He is now an assistant professor of Center for Information and Multimedia Studies, Kyoto University. His main research interests are model-based vision for human body, cooperative distributed vision, and multimedia processing. He is a member of IPSJ and IEEE.

Taoda, Takeo He received the B.E. and M.E. from Kyoto university in 1995 and 1997. He is now at NEC Software Chugoku. His main research interests were Cooperative Distributed Vision.

Minoh, Michihiko Michihiko Minoh was born in 1956 in Kyoto, Japan. He received the B.E., M.E. and D.E.degrees from Kyoto University, in 1978, 1980 and 1983, respectively. He has engaged in research in a variety of Image Processing, Artificial Intelligence and Multimedia Applications. He is currently a Professor of Center for Information and Multimedia Studies, Kyoto University. He is a member of IPSJ, IEICE, IEEE, and ACM.