MVA2011 IAPR Conference on Machine Vision Applications, June 13-15, 2011, Nara, JAPAN

A Parallelization Method for Multiview Stereo

Masaru Fukushi † Naoki Sekiguchi † Toru Abe ‡
† Graduate School of Information Sciences, Tohoku University Aoba, Sendai, 980-8579 Japan
‡ Cyberscience Center, Tohoku University Aoba, Sendai, 980-8577 Japan
mfukushi@ecei.tohoku.ac.jp naoki_s@ecei.tohoku.ac.jp beto@isc.tohoku.ac.jp

Abstract

This paper proposes a parallelization method for 3D shape reconstruction with voxel-based multiview stereo. In our approach, a target voxel space is divided into several domains, each of which is assigned to a different PE (processing element) to be processed in parallel. To reduce the amount of inter-PE communication, our proposed method divides the voxel space by planes passing through a virtual viewpoint computed from a configuration of actual viewpoints. Moreover, to reduce the idle time of PEs, our method performs dynamic load balancing flexibly, based on the estimated effect of it. Experimental results indicate that our method can reduce the processing time by 35% at 32 PEs as compared to a naive parallelization method.

1 Introduction

Multiview stereo is an approach for reconstructing 3D shape from a set of images taken at different viewpoints. Because of a wide range of applications, various methods have been proposed for this approach [1, 2]. A voxel-based method is a promising technique for the multiview stereo. In this method, a subject 3D space is divided into voxels, and the 3D shape is reconstructed by determining whether or not each voxel is on an object boundary, based on the photo-consistency of the voxel. The voxel-based method offers high accuracy to the 3D shape reconstruction by integrating the information in multiple images. However, as the numbers of images and voxels increase, it requires huge processing time for determining object boundary voxels.

Several techniques have been employed for reducing the processing time of the multiview stereo. Visual hulls [3] and hierarchical voxel spaces [4] are applied to pruning unnecessary voxel processing; however, their effectiveness is affected by object shapes. Multiprocessor systems have been used for processing individual images in parallel and/or the steps of 3D shape reconstruction in a pipeline fashion [5]. Their effectiveness barely depends on object shapes; however, it is difficult to proportion their effectiveness to the number of PEs (processing elements). The approaches using GPU (graphic processing unit) have been proposed for processing voxels in parallel [6]. Although they are highly effective, their efficiency decreases for large scale subjects (a large number of images and voxels) due to an increase in the amount of communication between GPU and CPU. To accomplish further speedup for large scale subjects, a method should be devised such that it achieves efficient parallel processing.

In this paper, we investigate an approach of voxel space division and propose a parallelization method to speed up the voxel-based multiview stereo. Our approach divides a target voxel space into several domains and assigns these domains to individual PEs (e.g., CPUs, GPUs, etc.). To reduce the amount of inter-PE communication, our method divides the voxel space by planes passing through a virtual viewpoint computed from a configuration of actual viewpoints. Moreover, to reduce the idle time of PEs, our method performs dynamic load balancing flexibly, based on the estimated effect of it. Our method can proportion its effectiveness to the number of PEs, and consequently we can expect that this method works well for the 3D shape reconstruction on large scale subjects by increasing the number of PEs.

2 Voxel-Based Multiview Stereo

2.1 Determining object boundary voxels

Suppose that N images $I_n (n = 1, 2, \dots, N)$ are used for 3D shape reconstruction, and each I_n is taken at a viewpoint o_n. As shown in Figure 1, the subject 3D space is divided into voxels $v(j, k, l)$. A whole set of v is represented by the voxel space V. By projecting a voxel v onto each I_n, a region $r_n(v)$ is obtained.

The voxel-based multiview stereo reconstructs the 3D shape by determining whether or not each v is on an object boundary, based on the photo-consistency $pc(v)$, i.e., the similarity in the color appearances of v among all $r_n(v)$. When v on an object boundary is projected onto I_n, the same object surface appears in all $r_n(v)$, and then it gives a high $pc(v)$. On the other hand, if v isn't on an object boundary, a different object surface (or the background) appears in each $r_n(v)$, and it gives a low $pc(v)$. Consequently, information in the images is integrated into the photo-consistency, from which object boundary voxels can be determined.

However, for several reasons (e.g., an object surface is occluded in part of the images, different object surfaces have similar colors, etc.), there are some cases where object boundary voxels cannot be determined correctly from the photo-consistency. To resolve this, many existing methods introduce geometric constraints [7, 8]. As shown in Figure 2, let $\phi_n(v)$ be a set of voxels on the straight line passing through o_n

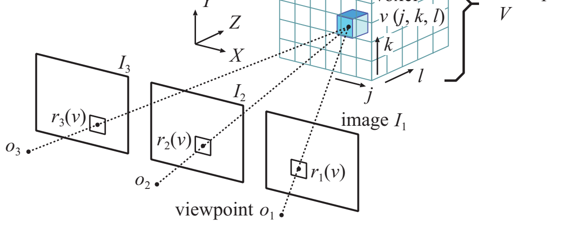

Figure 1. Voxel v and its projection $r_n(v)$.

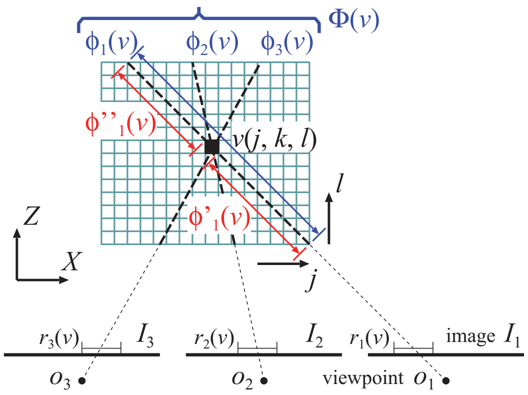

Figure 2. Geometric condition.

and v. The subsets $\phi'_n(v)$ and $\phi''_n(v)$ represent the voxels in $\phi_n(v)$ before v and behind v, respectively. If v is on an object boundary and visible from o_n, the voxels $\phi'_n(v)$ cannot be on object boundaries and $\phi''_n(v)$ cannot be seen from o_n. This condition is imposed as the geometric constraints; therefore, to determine whether v is on an object boundary, not only the photo-consistency of v but also the states of the other voxels in each $\phi_n(v)$ should be referred to, and those for all n need to be integrated (i.e., the states of the voxels $\Phi(v) = \cup_{n=1}^{N} \phi_n(v)$ are needed to determine v). For a large number of images and voxels, imposing this geometric constraints requires huge processing time.

2.2 Introducing geometric constraints

There are several approaches for voxel-based multiview stereo with the geometric constraints. Although our parallelization method is applicable to most of them, we focus on Kawakami's method [8] in the following. Figure 3 shows the algorithm of Kawakami's method. This method computes the certainty $e(v)$ that each v is on an object boundary, and refines it iteratively from the states of other voxels to satisfy the imposed constraints.

At Step 1, the certainty $e(v)$ is computed as a weighted average of $e_n(v)$ for all n by

$$e(v) = \textstyle\sum_n w_n(v) e_n(v) \ / \ \sum_n w_n(v). \qquad (1)$$

In Eq. (1), $e_n(v)$ denotes the certainty that v is on an object boundary which can be seen from o_n, and $w_n(v)$ denotes the weight for o_n. The certainty $e_n(v)$ is computed from the photo-inconsistency $pi_n(v)$ instead of the photo-consistency by

$$e_n(v) = (pi_{\max} - pi_n(v)) \ / \ (pi_{\max} - pi_{\min}), \qquad (2)$$

where $pi_{\max}$ and $pi_{\min}$ are the maximum and minimum of photo-inconsistency $pi_n(v)$ for all v and n. The photo-inconsistency $pi_n(v)$ is computed by

$$pi_n(v) = \frac{\sum_{m \neq n} w_m(v) w_n(v) \times SSD_{m,n}(v)}{\sum_{m \neq n} w_m(v) w_n(v)}, \qquad (3)$$

where $SSD_{m,n}(v)$ represents the sum of squared differences between $r_m(v)$ and $r_n(v)$. The weight $w_n(v)$, which is the certainty that v satisfies the geometric constraints, is computed by

$$w_n(v) = \frac{\max_{u \in \phi_n(v)} e_n(u) - \max_{u' \in \phi'_n(v)} e_n(u')}{\max_{u \in \phi_n(v)} e_n(u) - \min_{u \in \phi_n(v)} e_n(u)} \times \frac{\max_{u \in \phi_n(v)} e_n(u) - \max_{u'' \in \phi''_n(v)} e_n(u'')}{\max_{u \in \phi_n(v)} e_n(u) - \min_{u \in \phi_n(v)} e_n(u)}. \qquad (4)$$

```
// $T_1$: iteration bound for Step 1
// $T_2$: iteration bound for Step 2
for $t_1 = 0$ to $T_1 - 1$ do
    // Step 1
    for all $v \in V$ do
        compute $e(v)$ by Eqs. (1)~(4)
        push all $v \in V$ in set $V_{\text{proc}}$
        for $t_2 = 0$ to $T_2 - 1$ do
            // Step 2
            for all $v \in V_{\text{proc}}$ do
                compute $e_n^{(t_2+1)}(v)$ by Eqs. (5)~(7)
                if $\forall n \ \ e_n^{(t_2+1)}(v) = e_n^{(t_2)}(v) = \gamma$ then
                    pop $v$ off set $V_{\text{proc}}$
                end if
            end for
        end for
    end for
end for
// Step 3
determine object boundary voxels $v$ from $e(v)$
```

Figure 3. Algorithm of Kawakami's method.

At Step 2, each certainty $e_n(v)$ is refined iteratively according to the states of the voxels in $\Phi(v)$. Let V_{proc} be the set of all voxels which need the refining process. The certainty $e_n^{(t_2+1)}(v)$ at the $t_2 + 1$ th iteration is computed from the state at the t_2 th iteration by

$$e_n^{(t_2+1)}(v) = e^{(0)}(v) \times \left(A_n^{(t_2)}(v)\right)^{\alpha} \times \left(B^{(t_2)}(v)\right)^{\beta}, \qquad (5)$$

where $A_n^{(t_2)}(v)$ is defined as a relative value of $e_n^{(t_2)}(v)$ in each $\phi_n(v)$, and $B^{(t_2)}(v)$ is defined as a weighted average of $A_n^{(t_2)}(v)$ for all n. In Eq. (5), α and β are given constants. The values of $A_n^{(t_2)}(v)$ and $B^{(t_2)}(v)$ are computed by

$$A_n^{(t_2)}(v) = \frac{e_n^{(t_2)}(v) - \min_{u \in \phi_n(v)} e_n^{(t_2)}(u)}{\max_{u \in \phi_n(v)} e_n^{(t_2)}(u) - \min_{u \in \phi_n(v)} e_n^{(t_2)}(u)}, \qquad (6)$$

$$B^{(t_2)}(v) = \textstyle\sum_n w_n(v) A_n^{(t_2)}(v) \ / \ \sum_n w_n(v). \qquad (7)$$

In the iteration of Step 2, if $e_n^{(t_2+1)}(v)$ is less than a given threshold γ, its value is set to γ, and if $e_n^{(t_2+1)}(v)$ is greater than $1 - \gamma$, its value is set to $1 - \gamma$. When $e_n^{(t_2+1)}(v) = e_n^{(t_2)}(v) = \gamma$ for all n, the iteration for such v is terminated and v is removed from V_{proc}. At Step 3, in each v, the sum of $w_n(v)$ is calculated for n where $e_n(v)$ is a local maximum value. If the sum of $w_n(v)$ is greater than a given threshold δ, such v is determined to be an object boundary voxel.

As is described above, Kawakami's method refines the certainty $e(v)$ at each voxel by using the geometric constraints; hence, it allows to improve the accuracy in determining object boundary voxels.

3 Parallelization Method

3.1 Gradient-dependent domain division (GDD)

For the voxel-based multiview stereo, domain division is an effective approach to the parallelization, since the computation for each voxel v is independent of one another as shown in Figure 3. However, each v needs data on $\Phi(v)$ for refining $e(v)$; therefore, a naive domain division, which divides the voxel space V horizontally (or vertically) into P domains $V_p (p = 1, 2, \ldots, P)$

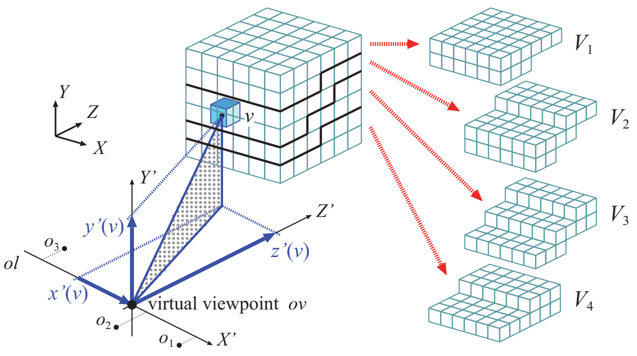

Figure 4. Voxel space dividing by GDD ($P = 4$).

```
// $T_2$: iteration bound for Step 2
push all $v \in V$ in set $V_{\mathrm{proc}}$
for $t_2 = 0$ to $T_2 - 1$ do
    // Step 2
    counting $\|V_*\|_{\max}$, $\|V_*\|_{\mathrm{avg}}$, and $\|V_{\mathrm{proc}}\|$
    if $(\|V_*\|_{\max} - \|V_*\|_{\mathrm{avg}})/\|V_{\mathrm{proc}}\| > C_2/C_1$ then
        //DLB
        reassign $V_p$ to PE$_p$
    end if
    for all $v \in V_{\mathrm{proc}}$ do
        compute $e_n^{(t_2+1)}(v)$ by Eqs. (5)~(7)
        if $\forall n \;\; e_n^{(t_2+1)}(v) = e_n^{(t_2)}(v) = \gamma$ then
            pop $v$ off set $V_{\mathrm{proc}}$
        end if
    end for
end for
```

Figure 5. Algorithm of FDLB.

to be processed in parallel at P PEs PE$_p$, causes huge inter-PE communication for data exchange and increases the total processing time.

We propose a new domain division method named Gradient-dependent Domain Division (GDD). GDD divides V focusing on the gradient of $\Phi(v)$ to reduce the inter-PE communication. The procedure of GDD is described as follows;

1. A virtual viewpoint ov is computed as the average of viewpoints o_n by $ov = (\sum_{n=1}^{N} o_n)/N$, and a row of o_n is approximated by a straight line ol. The coordinate system XYZ is transformed to $X'Y'Z'$, where the origin and X' axis are set at ov and ol.
2. For all v, the gradient $g(v)$ of a plane passing through ov and v is calculated by $g(v) = y'(v)/z'(v)$, where $y'(v)$ and $z'(v)$ denote the distances between ov and v along the Y' and Z' axes.
3. All v are sorted into descending order of $g(v)$, and V is divided into P domains V_p so that each V_p includes the same number of voxels.
4. The voxel in each V_p is assigned to PE$_p$.

Figure 4 illustrates an example of GDD ($P = 4$). Compared to the naive domain division, GDD makes it possible to reduce the amount of data to be exchanged between PEs because the data on each $\Phi(v)$ are mostly to be stored in the same PE. Note that this reducing effect of GDD is obviously larger in cases where viewpoints lie on a straight line.

3.2 Flexible dynamic load balancing (FDLB)

As shown in Figure 3, at the end of every iteration in Step 2, each PE must wait for the other PEs to complete the processes assigned them. Even though the same number of voxels are initially assigned to each PE, it gradually differs as the refinement of $e(v)$ proceeds because the iteration counts to terminate the refinement differ from voxel to voxel. Such imbalanced computation loads result in the increase of processing time due to the wasted waiting time. Therefore, Dynamic Load Balancing (DLB) should be employed.

DLB dynamically balances the computation loads of all PE$_p$. Although the computation load of each PE$_p$ cannot be known beforehand, it can be estimated by the number of voxels assigned to PE$_p$. Let $\|V_{\mathrm{proc}}\|$ be the number of all voxels which need the refinement of $e(v)$ and $\|V_p\|$ be the number of voxels assigned to each PE$_p$ from V_{proc}. When the difference in $\|V_p\|$ increases, DLB reassigns V_{proc} to each PE$_p$ so as to equalize $\|V_p\|$ in all PE$_p$. However, the reassigning procedures incur some overheads.

Here, we consider the condition that DLB should be performed. Let T_{red} and T_{inc} be the processing time reduced and increased by DLB, respectively. Obviously, DLB should be performed only when $T_{\mathrm{red}} > T_{\mathrm{inc}}$. Therefore, to achieve well-timed DLB, we have to estimate T_{red} and T_{inc} beforehand. Let $\|V_*\|_{\max}$ and $\|V_*\|_{\mathrm{avg}}$ be the maximum and average numbers of all $\|V_p\|$. Without DLB, the processing time for an iteration of Step 2 is the same as the maximum processing time of all PE$_p$ and it is proportional to $\|V_*\|_{\max}$. If computation loads of all PE$_p$ are ideally balanced, the processing time is the same as the average processing time of them and it is proportional to $\|V_*\|_{\mathrm{avg}}$. On the other hand, the overhead of DLB is proportional to $\|V_{\mathrm{proc}}\|$ because DLB reassigns the voxels in V_{proc} to each PE$_p$. Then, T_{red} and T_{inc} can be estimated by

$$T_{\mathrm{red}} = C_1\left(\|V_*\|_{\max} - \|V_*\|_{\mathrm{avg}}\right), \qquad (8)$$

$$T_{\mathrm{inc}} = C_2\|V_{\mathrm{proc}}\|, \qquad (9)$$

where C_1 and C_2 correspond to unit time for processing and reassigning a voxel, respectively, and they depend on a used system. Thus, the condition that DLB should be performed is derived as follows:

$$\left(\|V_*\|_{\max} - \|V_*\|_{\mathrm{avg}}\right)/\|V_{\mathrm{proc}}\| > C_2/C_1, \qquad (10)$$

where the left term is obtained by counting $\|V_p\|$ at each PE$_p$, and the right term (i.e., a threshold of the condition) is determined by preliminary experiments on the system. Using Eq. (10), DLB can be performed flexibly. We named it FDLB (Flexible DLB).

Figure 5 shows the modified procedures of Step 2 employing FDLB. Before starting the original procedures of Step 2, each PE$_p$ counts $\|V_p\|$ and sends it to all PEs so that they can decide whether DLB should be performed or not based on Eq. (10).

4 Experimental Results

We conducted the experiment of 3D shape reconstruction to evaluate the effectiveness of our parallelization method. The specification of the cluster machine used in the experiments is listed in Table 1. As inputs for the voxel-based multiview stereo, four images in the well-known data set "Temple" [9] were used.

Table 1. Specification of cluster machine.

CPU	AMD Opteron 2.0GHz
Memory	PC3200 ECC Registered, 2GB
Number of nodes	16 (2 PEs/node)
OS	SuSE Linux Enterprise Server 8

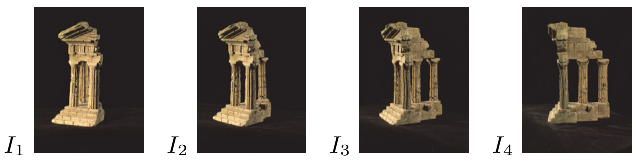

Figure 6. Input images used in the experiments.

These images $I_1 \sim I_4$ were taken at viewpoints placed along an arc (Figure 6). The voxel space consisted of 100^3 voxels. The iteration bounds T_1 and T_2 in Kawakami's method were set to 5 and 10. Other constants were set as $\alpha = 1.0$, $\beta = 1.5$, $\gamma = 0.001$, and $\delta = 1.0$. Based on the preliminary experiments, the threshold C_2/C_1 in Eq. (10) was set to 0.05.

Figure 7 shows the processing (communication and calculation) time and communication time by varying the number of PEs, where the voxel space was divided by HDD (naive Horizontal Domain Division) and GDD (our proposed method). This result indicates that the calculation time of both HDD and GDD decreases as expected with the increase of the number of PEs. It means that the voxel space division is effective for the parallelization of voxel-based multiview stereo. The communication time of GDD is much smaller than that of HDD, and the difference between them increases as the number of PEs increases. This is because GDD saves inter-PE communication by assigning a large part of the data on each $\Phi(v)$ to the same PE.

Figure 8 shows the processing time with FDLB. This result indicates that FDLB enables to reduce processing time in both HDD and GDD. However, the efficiency in HDD declines rapidly with the increase in the number of PEs because FDLB raises the amount of inter-PE communication. In contrast, the efficiency in GDD doesn't decrease as in HDD, since GDD can reduce the amount of inter-PE communication.

These experimental results, GDD with FDLB reduces the processing time by 35% at 32 PEs as compared to HDD without FDLB, show that our method offers efficient parallel processing for 3D shape reconstruction with the voxel-based multiview stereo. As mentioned above, the viewpoints where $I_1 \sim I_4$ were taken didn't lie on a straight line exactly. However, GDD worked well for these viewpoints. It means that GDD can permit the deviations of viewpoints from a straight line at least to some degree.

5 Conclusions

In this paper, we have proposed an efficient parallelization method for the voxel-based multiview stereo. Our approach divides the voxel space into several domains, each of which is assigned to a different PE to be processed in parallel. Our proposed method divides the voxel space in order to reduce the amount of inter-PE communication, and performs DLB flexibly to reduce the idle time of PEs. Through the experiments, we have confirmed the effectiveness of our method. In future work, we plan on conducting experiments for larger scale subjects and extending our method to adapt arbitrary viewpoint configurations.

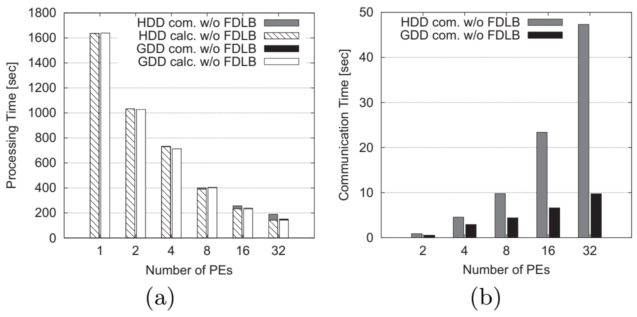

Figure 7. (a) Processing (communication and calculation) time, (b) Communication time.

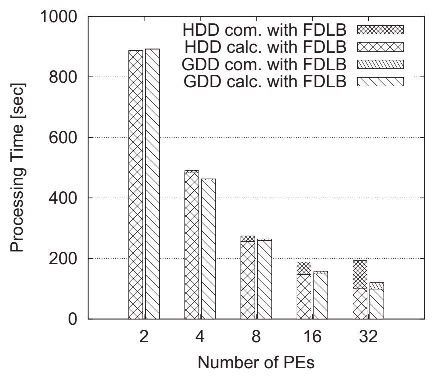

Figure 8. Processing time with FDLB.

References

[1] S.M. Seitz, et al.: "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *Proc. CVPR 2006*, pp.519–528, 2006.

[2] Y. Furukawa and J. Ponce: "Accurate, dense, and robust multiview stereopsis," *IEEE Trans. Pattern Anal. Machine Intell.*, vol.32, no.8, pp.1362–1376, 2010.

[3] K.N. Kutulakos and S.M. Seitz: "A theory of shape by space carving," *Int. J. Comput. Vis.*, vol.38, no.2, pp.199–218, 2000.

[4] A. Hornung and L. Kobbelt: "Hierarchical volumetric multi-view stereo reconstruction of manifold surfaces based on dual graph embedding," in *Proc. CVPR 2006*, pp.503–510, 2006.

[5] R. Nabeshima, et al.: "Frame rate stabilization by variable resolution shape reconstruction for on-line free-viewpoint video generation," in *Proc. ACCV 2006*, pp.81–90, 2006.

[6] C. Nitschke, et al.: "Real-time space carving using graphics hardware," *IEICE Trans.*, vol.E90-D, no.8, pp.1175–1184, 2007.

[7] G.G. Slabaugh, et al.: "Methods for volumetric reconstruction of visual scenes," *Int. J. Comput. Vis.*, vol.57, no.3, pp.179–199, 2004.

[8] K. Kawakami: "3D shape reconstruction method using multiple input images for modification," *Master's thesis*, Tohoku Univ., 2008.

[9] http://vision.middlebury.edu/mview (Nov, 2010).