社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2007－ＡＶＭ－59 (22)
2007／12／14

多眼ステレオ三次元形状計測の並列計算法

高橋 一樹† 福士 将† 阿部 亨†† 堀口 進†

† 東北大学 大学院情報科学研究科 〒980-8579 宮城県仙台市青葉区荒巻字青葉 6-3-09
†† 東北大学 情報シナジー機構 〒980-8577 宮城県仙台市青葉区片平 2-1-1
E-mail: †{kazuki,mfukushi,susumu}@ecei.tohoku.ac.jp, ††beto@isc.tohoku.ac.jp

あらまし 異なるカメラで撮影された複数の画像を用いて三次元形状計測を行う多眼ステレオ法の一つに，三次元空間をボクセルに分割し，各ボクセルが物体境界に対応するか否かを判定することで三次元形状計測を行う手法がある．この手法では，複数の画像の情報を効果的に統合した計測が可能となるものの，計測精度・安定性の向上を図るためには，ボクセル相互の遮蔽関係を各ボクセルの判定処理に反映させる必要があり，これには膨大な計算が必要となる．この問題に対処するために，本稿では，ボクセル空間分割をすることによる並列計算法を提案する．提案手法では，カメラの仮想視点と各ボクセルを通る直線の傾きを考慮したボクセル空間分割を行うことでプロセッサ間の通信量を大幅に削減し，また，プロセッサ間で並列通信を行うことで通信のオーバーヘッドを削減する．実画像を用いた三次元形状計測の性能評価により，16 台のプロセッサを用いた場合に約 8.2 倍の速度向上が得られることを確認した．
キーワード 三次元形状計測，多眼ステレオ法，ボクセル，並列計算

Parallel Calculation Method for Multicamera Stereo 3D Shape Reconstruction

Kazuki TAKAHASHI†, Masaru FUKUSHI†, Toru ABE††, and Susumu HORIGUCHI†

† Graduate School of Information Sciences, Tohoku University Aoba 6-3-09, Aramaki, Aoba-ku, Sendai-shi, Miyagi, 980-8579 Japan
†† Information Synergy Center, Tohoku University Katahira 2-1-1, Aoba-ku, Sendai-shi, Miyagi, 980-8577 Japan
E-mail: †{kazuki,mfukushi,susumu}@ecei.tohoku.ac.jp, ††beto@isc.tohoku.ac.jp

Abstract For reconstructing 3D shape of a scene from more than two images, voxel based multicamera methods have been proposed. In these methods, the 3D space is divided into voxels, and the 3D shape of a scene is reconstructed by determining the voxels corresponded to object surfaces. To improve the accuracy and stability of the reconstruction, these methods require that the states of other voxels are reflected on the determination for each voxel. However, this process necessitates huge computation time. To overcome this problem, this article proposes a parallel computation method based on voxel space division. The proposed method reduces the amount of communication between processors by voxel space division which takes into account the slope of the line through a virtual viewpoint and each voxel, and also reduces the communication latency by a parallel communication. The experimental results shows that the proposed method achieves up to 8.2 times faster performance using 16 processors.
Key words 3D Shape Reconstruction, Multicamera Stereo Method, voxel, Parallel Calculation

1. はじめに

計測範囲の拡大や計測精度の向上を図るために，異なるカメラで撮影された複数の画像を用いて三次元形状計測を行う手法（多眼ステレオ）が提案されている．その一つに，三次元空間をボクセルに分割し，異なるカメラから撮影された複数の画像を基に，各ボクセルが物体境界に対応するか否かを判定することで，対象の三次元形状計測を行うアプローチが提案されている．スペースカービング法 [1] に代表されるこのアプローチでは，複数の画像の情報を効果的に統合した三次元形状計測が可能となるものの，計測精度・安定性の向上を図るためには，ボクセル相互の遮蔽関係を各ボクセルの判定処理に反映させる必

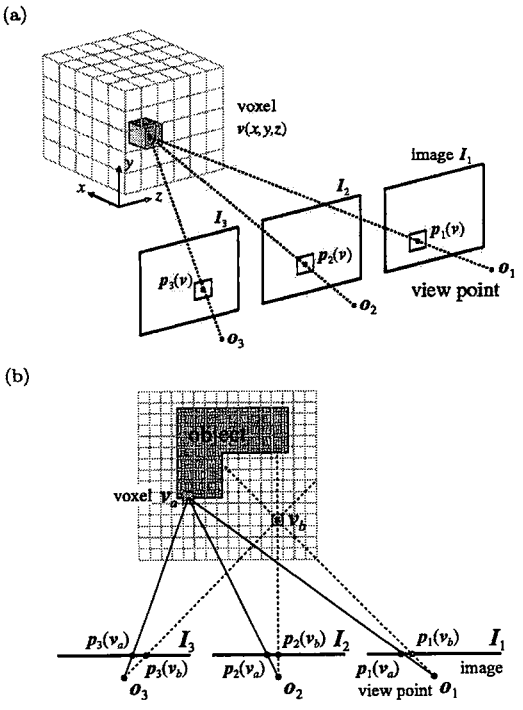

図 1　三次元空間の分割に基づく多眼ステレオ法

要があり，これを行う際には膨大な計算が必要となる．

スペースカービング法自体については，各ボクセルに対する判定処理を並列化し，GPU (Graphics Processing Unit) の機能を用いて高速に処理する手法が提案されている [2], [3]．しかし，遮蔽関係を反映した判定処理を行う場合には，各ボクセルで，他のボクセルの状態を参照した計算を反復する必要があり，処理の効率的な並列化を図るためには，他のボクセルの状態の参照に要する通信量を低減する並列化手法が必要となる．

この問題に対処するために，本稿では，ボクセル空間分割をすることによる並列計算法を提案する．提案手法では，カメラの仮想視点と各ボクセルを通る直線の傾きを考慮したボクセル空間分割を行うことでプロセッサ間の通信量を大幅に削減し，また，プロセッサ間で並列通信を行うことで通信のオーバーヘッドを削減する．実画像を用いた三次元形状計測の性能評価により，16 台のプロセッサを用いた場合に約 8.2 倍の速度向上が得られることを確認した．

2.　三次元空間の分割に基づく多眼ステレオ法

スペースカービング法に代表される多眼ステレオ法では，計測範囲である三次元空間をボクセル $v(x,y,z)$ に分割し，v を各画像へ投影した箇所の画素の一致度により v が物体境界に対応するか否かを判定することで，三次元形状計測を行っている．例えば，図 1 の v_a の場合，各画像へ投影した箇所 $p_1, p_2, p_3(v_a)$ には三次元空間内の同じ箇所が映るため，これらの画素の一致度は高く，物体境界に対応すると判定される．一方，v_b の場合，$p_1, p_2, p_3(v_b)$ には三次元空間内の異なる箇所が映り画素の一致

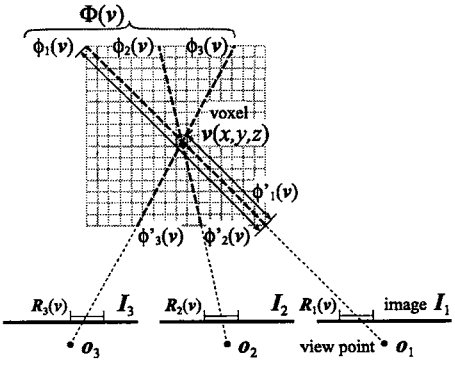

図 2　視点とボクセルの関係（y 軸方向から見た図）

度が低いため，物体境界に対応しないと判定される．

しかし，画素の一致度は，対象の遮蔽や対象の反射特性等によっても変化するため，これらの影響により，各 v の判定を誤る場合がある．この問題に対処するために，物体境界に対応する確からしさを各 v で求める際，他のボクセルの状態を参照することで判定精度の向上を図る手法が提案されている [4]〜[7]．

例えば，小田倉らの手法 [7] では，各 v ついて，物体境界に対応する確からしさ $e(v)$，画素の一致度 $S(v)$，画像 I_n で可視となる（遮蔽されない）確からしさ $w_n(v)$ を求め，これらの値を次に述べる手順で修正することにより，v に対する判定精度の向上を図っている．なお，以下では，計測範囲である三次元空間内の全 v の集合を V，各視点 o_n $(n = 1, 2, 3, \ldots, N)$ から撮影された画像を I_n とし，v を I_n へ投影した箇所を中心とする矩形領域を $R_n(v)$ で表す．また，図 2 に示すように，v と各 o_n とを通る直線上に位置するボクセルの集合を $\phi_n(v)$，v と各 o_n とを結ぶ線分上に位置するボクセルの集合を $\phi'_n(v)$ とし，各 v に対する $\cup_{n=1}^{N}\phi_n(v)$ を $\Phi(v)$ で表す．

Step 1.　各 v において以下の値を計算．

- **a.**　$\phi_n(v)$ 内のボクセルの e から $w_n(v)$ を計算．
- **b.**　$R_n(v)$ と $w_n(v)$ とを用い $S(v)$ を計算．
- **c.**　$S(v)$ に基づき $e(v)$ を計算．

Step 2.

- **a.**　各 v において，$\Phi(v)$ 内のボクセルの e に基づき $e(v)$ を修正し，この処理を V 全体に対し T_{2a} 回繰り返す．
- **b.**　Step 1, 2 の反復を $T_{1,2}$ 回繰り返す．反復を終了する場合は Step 3 へ．

Step 3.　各 v における $e(v)$ と $w_n(v)$ とに基づき，v が物体境界に対応するか否かを判定．

上記のステップ中，$w_n(v), S(v), e(v)$ の計算・更新，及び，v に対する判定は，以下のように行われている．

- $w_n(v)$ の計算（**Step 1a**）

大きな e を持つボクセルが $\phi'_n(v)$ 内に存在する場合，v は，I_n において，そのボクセルにより遮蔽される可能性が高い．そこ

で，式 (1) により，可視となる確からしさ $w_n(v)$ を決定する．

$$w_n(v) = 1 - \frac{\max_{\phi'_n} e - \mathrm{avg}_{\phi_n} e}{\max_{\phi_n} e - \mathrm{avg}_{\phi_n} e} \quad (1)$$

ここで，$\max_{\phi_n} e$ と $\mathrm{avg}_{\phi_n} e$ は，$\phi_n(v)$ 内での e の最大値と平均値を，$\max_{\phi'_n} e$ は，$\phi'_n(v)$ 内での e の最大値を各々表す．

- $S(v)$ の計算（**Step 1b**）

全ての画像対について，$R_m(v), R_n(v)$ 間の画素値の差の 2 乗和（sum of squared difference）$\mathrm{SSD}_{m,n}(v)$ を求め，式 (2) により，画素の一致度 $S(v)$ を計算する．

$$S(v) = \frac{\sum_{1 \leqq m < n \leqq N} w_{m,n}(v) \times \mathrm{SSD}_{m,n}(v)}{\sum_{1 \leqq m < n \leqq N} w_{m,n}(v)} \quad (2)$$

式 (2) では，v が可視となる確からしさが低い画像を用いた SSD へ小さな $w_{m,n}$ を与えることで，遮蔽の影響により $S(v)$ が増加することを防いでいる（$w_{m,n}$ には，2 つの重み w_m, w_n の内，値が小さいものを用いる）．

- $e(v)$ の計算（**Step 1c**）

$S(v)$ が小さい v は，各 I_n で同じように見えており，物体境界に対応する可能性が高いと考えられる．そこで，式 (3) により，物体境界に対応する確からしさ $e(v)$ を求める．

$$e(v) = 1 - \frac{S(v) - \min_V S}{\max_V S - \min_V S} \quad (3)$$

ここで，$\max_V S$ と $\min_V S$ は，全ボクセル V 内での S の最大値と最小値を各々表す．

- $e(v)$ の更新（**Step 2a**）

注目する v の $e(v)$ が，$\Phi(v)$ 内で相対的に小さな値であるならば，その v が物体境界に対応する可能性は低いと考えられる．そこで，$\Phi(v)$ 内での相対的な大きさに基づき，式 (4) により，$e(v)$ の値を更新する．

$$e_{t+1}(v) = e_0(v) \times \left(\frac{e_t(v) - \mathrm{avg}_\Phi e_t}{\max_\Phi e_t - \mathrm{avg}_\Phi e_t} \right)^\alpha \quad (4)$$

ここで，$e_0(v)$ は，更新される初期値を表し，Step 1c で求めた $e(v)$ の値を用いる．また，α は 1 より大きい定数である．$\max_\Phi e_t$ と $\mathrm{avg}_\Phi e_t$ は，t 回目の更新時における $\Phi(v)$ 内での e の最大値と平均値を各々表す．

- $e(v), w_n(v)$ に基づく v の判定（**Step 3**）

注目する v の $e(v)$ が各 $\phi_n(v)$ 上で極大となるかを調べ，極大となる $\phi_n(v)$ における $w_n(v)$ の総和が閾値 γ 以上であれば，v が物体境界に対応するものと判定する．例えば，図 2 の例で，$e(v)$ が $\phi_1(v), \phi_3(v)$ 上で極大となる場合，$w_1(v) + w_3(v) > \gamma$ ならば，v が物体境界に対応すると判断する．これにより，物体境界に対応する可能性が高く，かつ，複数の画像で可視となっている可能性も高いボクセルを物体境界として選択する．

以上述べた例からもわかるように，他のボクセルを参照することで判定精度を向上させる手法では，各 v において，他のボクセルの状態の参照と自身の状態の更新を反復するため，膨大な計算時間が必要となる．また，各 v は，$\Phi(v)$ 中の全ボクセルの状態を参照する必要があるため，ボクセル相互の参照関係は非常に複雑なものとなる．

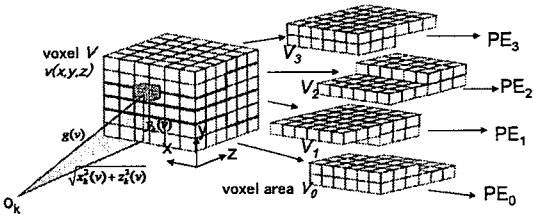

図 3　ボクセルを仮想視点からの傾きにより分割し各 PE に割り当て

3. 並列化手法

3.1 ボクセル空間分割

前述した小田倉ら [7] の手法では，$v \in V$ である一つ一つのボクセル v に対して各ステップの計算が行われるため，ボクセル空間を $P(P \geqq 1)$ 分割して P 台のプロセッサ（$PE : Processing Element$）で並列処理をする手法が考えられる．ボクセル空間の分割方法としては，空間全体を単純に水平方向に P 分割する手法や，垂直方向に P 分割する手法がある．しかし，あるボクセル v の計算に使用するデータは，$\phi_n(v)$，$\phi'_n(v)$，$\Phi(v)$ で示されているように，カメラの視点 o_n と v を通る直線（または線分）上のボクセル V' が持つデータである．そのため，単純に水平，垂直方向に分割した場合，計算に必要なボクセルデータが他の PE に存在する割合が増加し，PE 間の通信量が増加してしまう可能性がある．

このことから，本研究では，仮想視点と各ボクセルを通る直線の傾きを用いてボクセルを分割することで，通信量の少ない並列計算法を提案する．以下に，本稿で提案する並列計算法の手順を示す．

1. 各視点 o_n に配置されるカメラの座標の平均から仮想視点 o_k の座標を次式により求める．

$$o_k(x,y,z) = \frac{\Sigma_{n=1}^{N} o_n(x,y,z)}{N} \quad (5)$$

ここで $o_k(x,y,z)$ と $o_n(x,y,z)$ は，仮想視点 o_k の座標と各視点 o_n の座標を表す．

2. 求めた仮想視点 o_k から $v \in V$ に対して引いた各直線の傾き $g(v)$ を次式により求める．

$$g(v) = \frac{y_k(v)}{\sqrt{x_k(v)^2 + z_k(v)^2}} \quad (6)$$

ここで，$x_k(v), y_k(v), z_k(v)$ はそれぞれ仮想視点 o_k から各ボクセル v までの x 軸，y 軸，z 軸方向の距離である．

3. 求めた傾き $g(v)$ の大きさの順にソートしてボクセル全体を P 分割し，P 個のボクセル領域 $V_l (l = 0, 1, \ldots, P-1)$ を構成する．この時，$V = V_0 \cup V_1 \cup \ldots \cup V_{P-1}$ であり，各ボクセル領域 V_l は傾き $g(v)$ の小さい順に V_0 から V_{P-1} まで $\| V \| / P$ 個のボクセルで構成される．ここで，$\| V \|$ は全ボクセル V の要素数である．

4. 各ボクセル領域 V_l を PE_l に割り当てて，PE_l は割り当てられたボクセル領域 V_l の v についてのみ計算を行うことで並列的に処理を行う．

ボクセルを $P=4$ として分割する場合の概略図を図3に示す．図に示すように，仮想視点 o_k から求めた傾き $g(v)$ を用いてボクセル空間分割し，各 PE_l がボクセル領域 V_l の v について計算をすることで並列計算を行う．この手法により，同一直線上のボクセルデータが同一 PE 内に比較的多く配置されるため，単純にボクセル全体を水平，垂直平面で P 分割する場合に比べて，各 PE 間のデータ通信量を少なく抑えることが可能になる．

上記の並列化手法で小田倉ら[7]の手法を並列化する際に発生する通信は，次の3種類である．

1. Step 1b で各ボクセル v について $S(v)$ を計算できたかできなかったか(初回のみ)．
2. Step 1c で $max_V S, min_V S$ の計算に必要な $max_{V_l} S, min_{V_l} S$ の値．
3. Step 1a，Step 2a，Step 3 で計算に必要な $e_n(v)$ の値

1.については，あるボクセル v から視点 o_n へ投影した時に，画像 I_n 中に矩形領域 $R_n(v)$ を作れない場合が生じる．この時の v はSSDを求められないため，$S(v)$ が求められない．$S(v)$ の計算の可否は，直線上のボクセルを数え上げる時に計算から省くために必要になる．そのため，各 PE が担当する v について $S(v)$ の計算の可否を求め，それを他の全ての PE へと通信をする．2.については，$max_V S, min_V S$ の計算のために，各 PE_l に割り当てられたボクセル領域 V_l 内での $max_{V_l} S, min_{V_l} S$ を各 PE それぞれで求め，その値を代表として PE_0 が集めて，PE_0 が $max_V S, min_V S$ の値を求めて各 PE へと返す．

そのため，傾き $g(v)$ によるボクセル空間分割により通信量を削減できるボクセルデータは，3.の $e_n(v)$ の値を通信する時である．

3.2 並列通信

ボクセル空間を P 分割した時，あるプロセッサ PE_l で計算に必要になるボクセルデータは，PE_l 自身と，接するプロセッサ PE_{l+1} と PE_{l-1} に集中して存在することになる．そのため，PE_l が担当する計算を実行するには隣接する PE 間の通信が発生する．P 台の PE で並列処理を行う場合，以下に示す通り $2P-2$ 回の逐次通信を行う必要がある．ここで矢印はボクセルデータの通信の流れを示す．

1. $PE_0 \rightarrow PE_1$
2. $PE_1 \rightarrow PE_0$
3. $PE_1 \rightarrow PE_2$
4. $PE_2 \rightarrow PE_1$

$\vdots$

$(2P-3)$. $PE_{P-2} \rightarrow PE_{P-1}$

$(2P-2)$. $PE_{P-1} \rightarrow PE_{P-2}$

図4に，$P=8$ の場合に逐次通信を行った時のタイムチャートを示す．縦軸は通信をする時の送信元 PE であり，四角の中の PE は送信先 PE を表している．図4に示されるように，PE_0 と PE_1 で通信を行った後に PE_1 と PE_2 で通信を行う，という順番で通信を繰り返す．しかし，このような通信方法だと，例えば，PE_0と PE_1 との間で通信を行っている時は PE_2

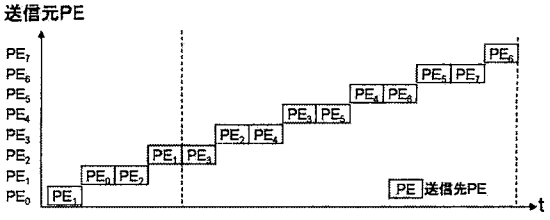

図4 逐次通信する場合 ($P=8$)

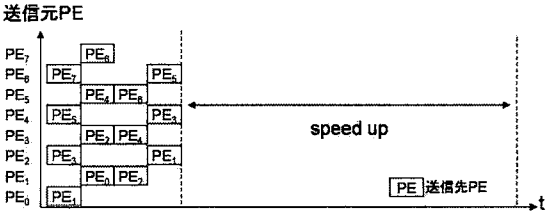

図5 並列通信する場合 ($P=8$)

～PE_7 がアイドル状態となっているため，通信待ち時間の無駄が生じる．そのため，本研究では以下のような順番で並列通信を行う．

1. $PE_0 \rightarrow PE_1$ $PE_2 \rightarrow PE_3$ $\cdots$ $PE_{P-2} \rightarrow PE_{P-1}$
2. $PE_0 \leftarrow PE_1$ $PE_2 \leftarrow PE_3$ $\cdots$ $PE_{P-2} \leftarrow PE_{P-1}$
3. $PE_1 \rightarrow PE_2$ $PE_3 \rightarrow PE_4$ $\cdots$ $PE_{P-3} \rightarrow PE_{P-2}$
4. $PE_1 \leftarrow PE_2$ $PE_3 \leftarrow PE_4$ $\cdots$ $PE_{P-3} \leftarrow PE_{P-2}$

図5に，$P=8$ の場合に並列通信を行った時のタイムチャートを示す．各 PE がなるべくアイドル状態にならないように並列通信を行うことで通信に必要な時間を削減することができる．特に，$P \geq 3$ の時には，P の値に関わらず1.～4.までの4段階で隣り合った PE 間の通信を完了させることができるため，P の値が大きいほど通信時間の短縮効果が大きくなる．

なお，P の値を増やしていくにつれて，PE_l の計算に必要なボクセルデータが PE_{l+2} や PE_{l-2}，もしくはそれ以上離れた PE に存在するようになる．しかし，PE_l で必要な大半のボクセルデータが PE_{l+1} と PE_{l-1} に存在するため，PE_{l+2} や PE_{l-2} などとの通信量は非常に少ない．今回は，2つ以上離れた PE に必要なボクセルデータが存在した場合には，隣接 PE 間での通信が終了した後に，適宜，通信を行うものとした．

4. 評価

4.1 実験環境

ボクセル空間分割による並列化の有効性を検証するために，実画像を対象として三次元形状を計測し，計算時間による評価を行った．実験には4台のカメラと2個の物体 Object1,2 を用い，図6に示すように各々を配置した．各カメラで撮影し実験に用いた画像 I_1～I_4 を図7に示す(画像サイズは 680×480 画素)．計測範囲である三次元空間は，x, y, z 方向に各々 $100 \times 100 \times 100$ 個のボクセルに分割している(1ボクセルのサイズは $8 \times 2 \times 8$mm)．

なお，式(4)中のパラメータを $\alpha = 1.5$ とし，物体境界に対

表 1　計算機性能

CPU	AMD Opteron DP Model 246 Processor 2.0GHz
Memory	PC3200 ECC Registered DDR-SDRAM 2GB
Number of Nodes	23
OS	SuSE Linux Enterprise Server 8
Compiler	PGI Compiler
MPI Library	MPICH-SCore

応するボクセルを判定する際の w に対する閾値は $\gamma = 1.3$ とし，Step 2a の反復回数を $T_{2a} = 10$，Step 1, 2 の反復回数を $T_{1,2} = 2$ とした時の計算時間を測定した．

また，実験で用いたクラスタマシンの性能を表 1 に示す．

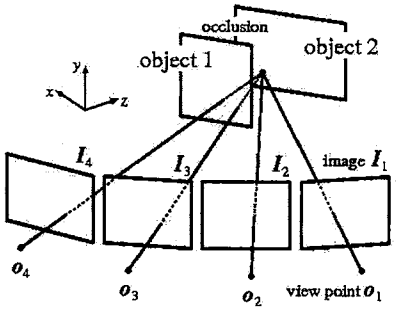

図 6　視点と物体 (object1,2) の配置

4.2　並列計算の評価結果

提案する並列計算法により三次元形状計測を行った際の計算時間の内訳を図 8 に示す．ここで，横軸は計算に使用したプロセッサ (PE) 数，縦軸は計算に要した時間である．PE 数が 1 の場合は，グラフの下から，画像の分散計算などにかかる計算時間，Step 1，Step 2，Step 3 の各計算時間が積み上げられている．また，PE 数が 2 以上の場合には，Step 1 と Step 2 の計算時間の間に Step 1 の通信時間が，Step 2 と Step 3 の計算時間の間に Step 2 の通信時間がさらに積み上げられる．なお，ここでの通信時間は前章 3.2 で提案した，並列通信を行った場合の通信時間である．

図 8 より，PE 数を増加させるにつれて，各 Step とも計算時間が削減されていることがわかる．これは，提案する並列化手法では，各 PE がそれぞれの担当ボクセル領域内に存在するボクセルの計算だけを行うためである．特に，$e(v)$ の値を繰り返し更新する Step 2 の計算時間の削減効果が大きいことがわかる．なお，画像の分散計算などの処理は，今回の並列化手法では並列化ができない処理であり，PE 数が増加しても計算時間に変化が生じない．

図 9 に，ボクセル V を水平方向に等分割して三次元形状計測を行った際の計算時間の内訳を示す．この時，提案する並列計算法と同様に，各 PE がそれぞれの担当ボクセル領域内に存在するボクセルの計算だけを行うため，各 Step とも計算時間が削減されていることがわかる．しかし，各 PE の計算に必要

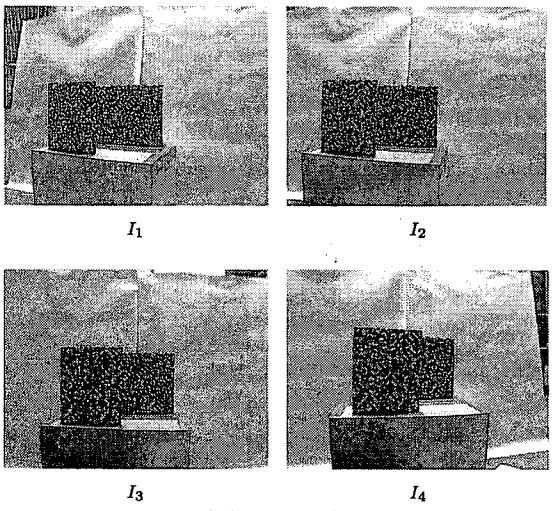

図 7　実験に用いた入力画像

なボクセルデータが他の PE に存在する割合が増加し，PE 間の通信量が増加してしまう．そのため，PE 数が 16 の時には，ボクセルデータの通信に多くの時間を費やしてしまい性能が低下してしまっている．

図 10 に，傾きによる分割を行い，並列通信をする場合としない場合の速度向上比と，水平分割をした場合の速度向上比を示す．なお，速度向上比は次式で定義される指標である．

$$\text{速度向上比} = \frac{P = 1 \text{ の時の計算時間}}{P = \{1, 2, 4, 8, 16\} \text{ の時の計算時間}} \qquad (7)$$

図 10 より，傾きによる分割を行い，さらに，並列通信をする並列計算法において，PE 数が 16 の時に約 8.2 倍の速度向上を達成することがわかる．ボクセル空間を P 分割して P 台の PE で並列計算した場合，速度向上比は P となるのが理想的であるが，実際は，速度向上比 P を実現するのは困難である．その原因として以下のようなことが挙げられる．

- PE 数が増加するにつれて，通信量が増加する．
- PE 数が増加するにつれて，並列計算ができない部分の計算時間が相対的に高くなる．
- 各 PE 間の計算負荷の大小に偏りがある．

このような理由により速度向上比が実測値で示す値に抑えられてしまっている．

4.3　並列通信の評価結果

並列通信の効果を検証するために，並列通信を行う場合と行わない場合で通信時間の測定を行った．結果を図 11 に示す．PE 数が増加するにつれて，PE 間でのボクセルデータの通信量が増加するために，通信時間が増加する傾向がある．また，並列通信を行うことで，行わない場合に比べて，通信時間を削減できていることがわかる．これにより，通信待ち時間の無駄が省かれるため計算速度が向上する (図 10 参照)．

5. ま と め

本稿では，他のボクセルの状態を参照することで判定精度の

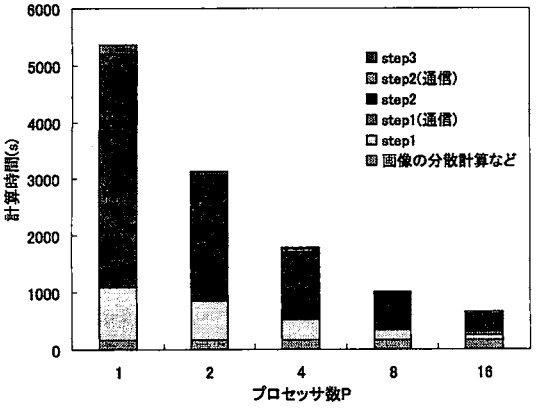

図 8　全体の計算時間 (傾きによる分割 & 並列通信)

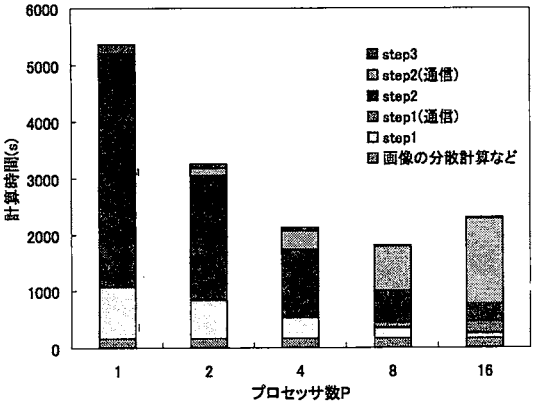

図 9　全体の計算時間 (水平分割)

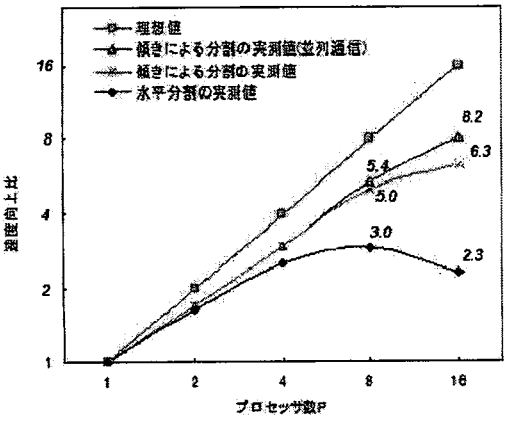

図 10　速度向上比

向上を図る多眼ステレオ法のうちで，小田倉ら [7] によって提案された手法を例に挙げて，計算時間を短縮化するための並列計算法を提案した．提案する手法では，カメラの仮想視点 o_k

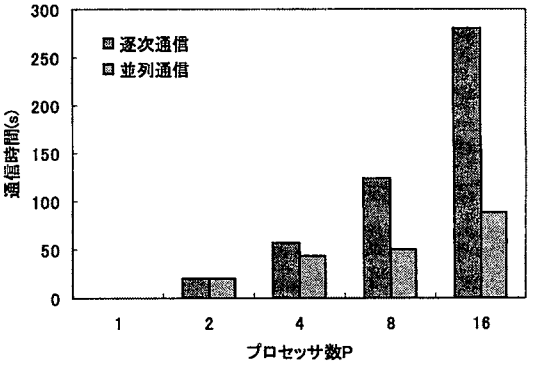

図 11　全体の通信時間

と各ボクセル v を通る直線の傾き $g(v)$ を考慮したボクセル空間分割を行うことで，三次元形状計測の並列計算で必要となる PE 間通信量を大幅に削減し，また，通信待ち PE をできるだけ少なくするような並列通信を行うことで，通信オーバーヘッドを削減する．実画像を用いた三次元形状計測の性能評価により，PE 数 16 で最大約 8.2 倍の速度向上を達成可能なことを示した．

今後は，さらなる速度向上を実現するために，各 PE の計算負荷の不均衡に対処するための負荷分散法を検討する予定である．

文　献

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

[2] C. Nitschke, A. Nakazawa, and H. Takemura, "Real-time space carving using graphics hardware," 画像の認識・理解シンポジウム (MIRU2006) 論文集, pp.928–933, 2006.

[3] 板野 友哉, 森栄 晃彦, 古川 亮, 川崎 洋, "未観測ボクセルのクラス推定を用いた形状の統合及び補間手法と GPU を用いた高速な実装," 画像の認識・理解シンポジウム (MIRU2007) 論文集, pp.365–371, 2007.

[4] M. Agrawal and L.S. Davis, "A probabilistic framework for surface reconstruction from multiple images," Proc. 2001 IEEE Comput. Society Conf. Comput. Vision Pattern Recognit., pp.470–476, 2001.

[5] A. Broadhurst, T.W. Drummond, and R. Cipolla, "A probabilistic framework for space carving," Proc. 8th Int. Conf. Comput. Vision, vol.1, pp.388–393, 2001.

[6] R. Bhotika, D.J. Fleet, and K.N. Kutulakos, "A probabilistic theory of occupancy and emptiness," Proc. 7th European Conf. Comput. Vision, pp.112–130, 2002.

[7] 小田倉 聡司, 阿部 亨, 木下 哲男, "三次元情報の整合性を考慮した多眼ステレオ法," 画像の認識・理解シンポジウム (MIRU2006) 論文集, pp.483–488, 2006.