I-054

反復更新の制御と動的負荷分散による並列三次元形状復元法の高速化

Parallel Multi-stereo 3D Shape Reconstruction Based on Flexible Iteration Control and Dynamic Load Balancing

関口 直紀 † 福士 将 † 阿部 亨 ‡ 堀口 進 †
Naoki Sekiguchi Masaru Fukushi Toru Abe Susumu Horiguchi

1 はじめに

異なる視点から撮影された 3 枚以上の画像を用いて撮影対象の三次元形状を復元するアプローチとして，多眼ステレオ法がある．多眼ステレオ法では，特殊な計測機器を必要とせず，カメラで撮影された画像のみを用いて対象の三次元形状復元を行う．そのため，撮影対象，撮影環境への制限が少ないという利点があり，物体形状認識や任意視点画像の生成，バーチャルリアリティなど，多くの分野でその応用が期待されている．

多眼ステレオ法の一つに，ボクセルベースの手法がある [1]．この手法は，(1) 計測範囲である三次元空間をボクセルに分割し，(2) ボクセルを各画像へ投影した箇所の画素の一致度（photo-consistency）を求め，(3) 画素の一致度に基づき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の復元を行うものである．ボクセルベースの手法は，複数の画像の情報を効果的に統合した三次元形状復元が可能となるものの，復元精度・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る．そこで，各ボクセルにおいて，他のボクセルの状態に基づき，物体境界に対応する確からしさを更新し，これを全ボクセルで反復する手法が提案されている．この手法では，他の多眼ステレオ法よりも高精度・安定した三次元形状復元が可能となる反面，膨大な計算が必要になるという問題点がある．

ボクセルベースの手法の計算量を削減するアプローチとして，Projection Grid[3][4] や八分木データ構造 [5][6] などを導入して，不要な計算を削減するアプローチや，計算処理の並列化 [7] により計算時間を短縮するアプローチがある．本稿では，大規模なボクセル空間に対しても高速化を実現できる点で，並列化によるアプローチに注目する．並列化の方法としては，ボクセル空間を複数に分割し，各々を別々のプロセッサ (Processing Element: PE) に割り当て，複数の PE により三次元形状復元を実行することで並列計算を実現する手法が提案されている．しかし，他のボクセルの情報を参照しながら確からしさを反復更新する手法に対し並列処理を適応させる場合，データ参照の複雑さにより効率のよい並列化手法の実現が難しいものとなっている．ボクセル空間の分割方法を工夫し，PE 間の通信量を少なく抑える手法などが提案されているが，負荷の不均衡を解消するまでには至っていない．

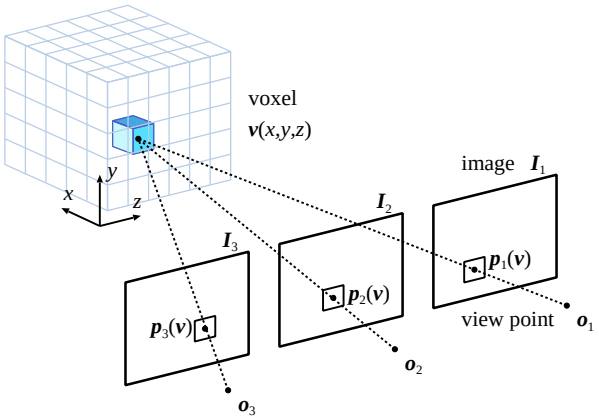


図 1 ボクセルベースの手法

本稿では，並列化手法のさらなる高速化を実現するために，以下の 2 点を提案する．(1) 確からしさの反復更新において，値が収束したボクセルの処理を打ち切ることで更新を制御し，計算量を削減する．(2) 反復更新の制御により広がる PE 間の計算負荷の不均衡に対処するために，反復更新に必要なデータ量に応じてボクセル空間の再分割を行い，動的に負荷分散を行う．上記 2 点の手法により，三次元形状復元の精度を保ちながら，高速化を実現することを図る．

本論文の構成は以下の通りである．2 節では，他のボクセルの情報を参照しながら確からしさを反復更新する手法と，その並列化手法について述べ，その問題点を挙げる．3 節では，反復更新の制御と動的負荷分散による高速な並列三次元形状復元手法を提案する．4 節では，提案手法の有効性を示すための実験を行い，結果に対する考察を述べる．最後に 5 節において結論と今後の課題を述べる．

2 従来手法

2.1 ボクセルベースの三次元形状復元手法

Kutulakos らの提案したスペースカービング法 [1] は，複数の画像の情報を効果的に統合した三次元形状復元が可能な手法である．スペースカービング法は，図 1 に示すように，復元対象の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基

†東北大学 大学院情報科学研究科, Graduate School of Information Sciences, Tohoku University
‡東北大学 サイバーサイエンスセンター, Cyberscience Center, Tohoku University

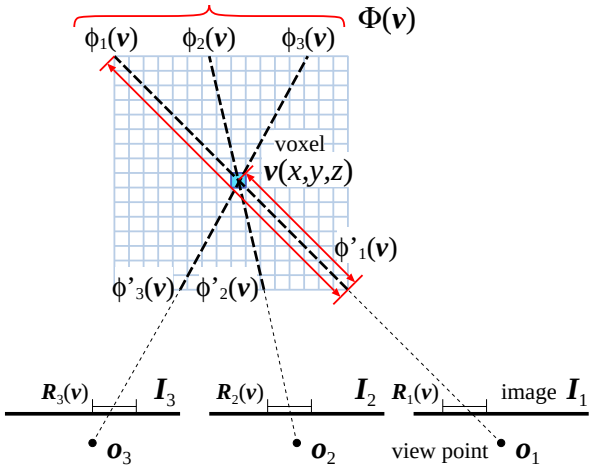


図 2　視点とボクセルの関係

づき，物体境界に対応するか否かを順次判定していくことで三次元形状復元を行う．しかし，復元対象の反射特性や，画像中の雑音の影響により，画素の一致度だけでは安定した復元が困難な場合が生じる．

そこで，三次元形状の整合性を考慮しながら，周囲のボクセルの情報を参照し確からしさの反復更新を行うことで問題の解決を図る手法がいくつか提案されている．以下では，[2] で提案された手法について説明する．

ここでは，三次元形状の復元に N 台のカメラを用いるものとし，$\boldsymbol{o}_n$ $(n = 1, 2, 3, \ldots, N)$ から撮影された画像を $\boldsymbol{I}_n$ で表す．また，計測範囲である三次元空間を分割したボクセルを $v(x, y, z)$，全 v の集合を v で表し，各 v には，物体境界に対応する確からしさ $e(v)$（$[0, 1]$ であり，値が大きいほど確からしい）を与える．図 2 に，視点とボクセルの関係を示す．以下では，v と各 $\boldsymbol{o}_n$ 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 $\boldsymbol{o}_n$ とを結ぶ線分上に位置するボクセルの集合を $\boldsymbol{\phi'}_n(v)$，各 v に対する $\cup_{n=1}^{N}\Phi_n(v)$ を $\Phi(v)$ で表す．

この手法では，次のステップにより形状復元を行う．

Step 0　各 v において $e(v)$ を 0 に設定．

Step 1　各 v において以下の Step 1 a,b,c により，v が $\boldsymbol{I}_n$ で可視である確からしさ $w_n(v)$（$[0, 1]$ であり，値が大きいほど確からしい），画素の一致度 $S(v)$,$e(v)$ を計算．

• **Step 1 a**　e が大きいボクセルが $\boldsymbol{\phi'}_n(v)$ 内に存在する場合，v は，$\boldsymbol{I}_n$ において，そのボクセルにより遮蔽される可能性が高いと考えられる．そこで，提案手法では，式 (1) により $w_n(v)$ を決定する．

$$w_n(v) = 1 - \frac{A_n(v) - \mathrm{avg}_{\Phi_n} e(v)}{\max_{\Phi_n} e(v) - \mathrm{avg}_{\Phi_n} e(v)} \qquad (1)$$

ここで，$\mathrm{avg}_{\Phi_n} e(v)$,$\max_{\Phi_n} e(v)$ は，$\Phi_n(v)$ 内での e の平均値と最大値を各々表す．$A_n(v)$ は，$\boldsymbol{o}_n$ から v まで順に $e(v)$ を探索した際，$e(v)$ がとる最大の極大値とする．極大値が存在しない場合は $e(v) = 0$ とする．

• **Step 1 b**　全ての画像対 $\boldsymbol{I}_m, \boldsymbol{I}_n$ $(1 \le m < n \le N)$ に対し，v を各 $\boldsymbol{I}_n$，$\boldsymbol{I}_m$ へ投影した箇所近傍 $\boldsymbol{R}_m(v), \boldsymbol{R}_n(v)$ 間の画素値の差の 2 乗和（Sum of Squared Difference）$\mathrm{SSD}_{m,n}(v)$ を求め，次式を用いて $S(v)$ を計算する．

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

$$w_{m,n}(v) = \begin{cases} w_m(v), & w_m(v) < w_n(v) \\ w_n(v), & \text{otherwise} \end{cases} \qquad (3)$$

式 (2) では，SSD に対する重みに w を用いることで，v が可視となる確からしさが低い画像に基づく SSD へ小さな重みを与え，遮蔽が生じている画像の影響により $S(v)$ が増加することを防いでいる．

• **Step 1 c**　$S(v)$ が小さい v は，複数の画像で同じように見えていることになり，物体境界に対応する可能性が高いと考えられる．そこで，v が物体境界に対応する確からしさとして，式 (4) で得られる $e(v)$ を用いる．

$$e(v) = 1 - \frac{S(v) - \min_v S(v)}{\max_v S(v) - \min_v S(v)} \qquad (4)$$

ここで，$\min_v S(v), \max_v S(v)$ は，全ボクセル v 内での S の最小値と最大値を各々表す．

Step 2　各 v に対し，Step 1 c で求めた $e(v)$ を，v 全体で三次元情報の整合性がとれるように次式を用いて更新する．

$$e_t(v) = e_0(v) \times \left(\frac{e_{t-1}(v) - \mathrm{avg}_\Phi e_{t-1}(v)}{\max_\Phi e_{t-1}(v) - \mathrm{avg}_\Phi e_{t-1}(v)} \right)^{\alpha} \qquad (5)$$

ただし，$e_{t-1}(v) < \mathrm{avg}_\Phi e_{t-1}(v)$ である場合には，

$$\frac{e_{t-1}(v) - \mathrm{avg}_\Phi e_{t-1}(v)}{\max_\Phi e_{t-1}(v) - \mathrm{avg}_\Phi e_{t-1}(v)} = \beta \qquad (6)$$

となるように設定する．

ここで，$e_0(v)$ は，更新される初期値を表し，Step 1 c で求めた $e(v)$ を用いる．また，$e_t(v)$ は t 回目の更新後の $e(v)$ の値，$\mathrm{avg}_\Phi e_t(v)$，$\max_\Phi e_t(v)$ は t 回目の更新後の $\Phi(v)$ 内での e の平均値と最大値を各々表す．α は 1 より大きい定数，β は $[0, 1]$ の定数である．

Step 2 では，この更新を v 内の各 v で行い，それを T_{2a} 回反復する．さらに，ここで更新された e を用い，Step 1, 2 を $T_{1,2}$ 回反復する．これにより，v 全体で三次元情報の整合性がとれるよう，各 v における $e(v)$ の更新を図る．

Step 3　Step 1，Step 2 の反復更新が全て終了すると，物体境界の判定処理を行う．物体境界か否かの判定には，Step 1，Step 2 の反復処理において計算された $e(v)$ がある閾値以上であった場合に，そのボクセルを物体境界と決定する．今回，物体境界と判定されるボクセル個数と復元精度のバランスを考え，以下の条件を満たしたボクセル v を物体境界とした．

$$e(v) > 0.9 \qquad (7)$$

2.2 並列三次元形状復元法

本節では，三次元形状の整合性を考慮した多眼ステレオ法 [2] に対して並列化を行った並列三次元形状復元法 [7] について説明を行う．

この手法では，ボクセル空間を $P(P \geq 1)$ 分割して，P 個のボクセル領域 $v_l(l = 0, 1, ..., P-1)$ を構成し，それぞれの領域の処理を各 PE が担当し，P 台の PE で並列処理を行っている．あるボクセル v の処理に必要な主な情報は，$\phi_n(v)$，$\phi'_n(v)$，$\Phi(v)$ 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) 上のボクセルが持つデータである．このことから，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を行っている．以下に，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 の座標を求める．

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．

$$g(v) = \frac{y_k(v)}{\sqrt{x_k(v)^2 + z_k(v)^2}} \tag{8}$$

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 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．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．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，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．

以上述べてきた手法で精度の高い三次元形状復元と，その処理の並列化が可能となる．しかし，Step 2 の確からしさの反復更新において，値が早期に収束し，その後の更新で e の値がほとんど変化しないようなボクセルについても，事前に設定した T_{2a} 回の更新を行っており，不必要な計算を行っている．また，ボクセルごとに，必要な処理量が異なっているにも関わらず，ボクセル空間の分割において，ボクセル個数が均等になるように分割を行っているため，並列化した際の各 PE における負荷が不均一になっている．

3 提案手法

3.1 反復更新の制御による高速化

従来法では，Step 2 の処理において，e の値の反復更新の回数 T_{2a} は事前に設定する必要があり，全てのボクセルに対して T_{2a} 回の e の反復更新を行う．しかし，ボクセルによっては，早期に e の値が収束し，その後の更新で e の値がほとんど変化しないようなものも存在する．例えば，物体境界に対応していないボクセルでは，その箇所を投影した画像には異なるものが投影されるため，画像間の $S(v)$ の値が大きくなり，式 (4) より，Step 2 の更新処理における e の初期値は小さな値となる．式 (5) に示されるように，そのようなボクセルでは，反復更新において急激に e の値が減少し，$e \to 0$ に向かって収束する．実際，式 (5)，(6) より，$e_{t-1}(v)$ が $\Phi(v)$ 中の平均値以下となる場合には，$e_t(v)$ は更新の初期値 $e_0(v)$ の β^α 倍で一定となる．このようなボクセルの $e(v)$ が Step 2 の反復更新内で再び上昇するには，式 (5) 中において，$e_{t-1}(v) > \mathrm{avg}_\Phi e_{t-1}(v)$ となる必要がある．すなわち，$\Phi(v)$ 中の他のボクセルの e の値が全体的に減少し，その平均値が $e_{t-1}(v)$ を下回ることが最低条件となる．しかし，実際にはそのようなことが起こる可能性は低い．そのため，e の値が早期に収束したボクセルに対しては，T_{2a} 回の反復更新を繰り返す必要はなく，計算を途中で打ち切ることで，復元精度に与える影響を小さく抑えたまま，計算量の削減をすることが可能になると思われる．

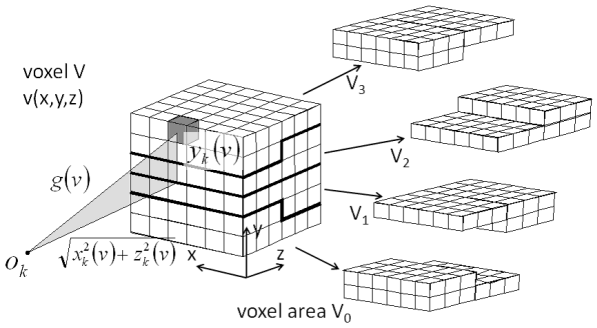


図 3 平均からの傾きによる分割

上記の考えに基づき，並列三次元形状復元法 [7] の Step 2 を以下のように修正する手法を提案する．

Step 2.

a. 各 v において，$\Phi(v)$ 内のボクセルの e に基づき $e(v)$ を更新する．

b. 以下の条件を満たすボクセル v に対しては，$e(v)$ の反復更新を打ち切る．

$$e_t(v) < e_0(v) \text{ かつ } e_t(v) < \gamma \tag{9}$$

ここで，γ は $[0, 1]$ の定数とする．全てのボクセルにおいて e の更新が打ち切られるか，または，更新回数 $t = T_{2a}$ となった場合は Step 2 c へ．そうでない場合は，Step 2 a へ．

c. Step 1, 2 の反復を続ける場合は Step 1 へ，反復を終了する場合は Step 3 へ．

更新打ち切りの条件に関して，上記の方法の他に，$e_t(v)$ の変化量が一定値以下になった場合に更新を打ち切るという方法も考えられる．しかし，式 (5)，(6) に示されるように，$e_t(v)$ の値は Step 2 開始前の初期値 $e_0(v)$ を基に計算されており，すでに述べたように，$e_{t-1}(v)$ が $\Phi(v)$ 中の平均値以下となる場合には，$e_t(v)$ は更新の初

期値 $e_0(v)$ の β^{α} 倍で一定となるため，単純に閾値による条件で問題ないといえる．また，$e_t(v) < \gamma$ を満たしていても $e_t(v) = e_0(v)$ となる場合，$e_t(v)$ は $\Phi(v)$ 中の最大値であることを示し，今後の更新で値が上昇する可能性があるため，$e_t(v) < e_0(v)$ の条件を加えた．

Step 2 c において Step 1 に分岐し，Step 1，2 の反復を続ける場合には，Step 2 b で更新を打ち切られたボクセルに対しても Step 1 の計算を行うこととする．これは，式 (1)〜(4) に示される通り，Step 1 では，Step 2 で更新された $e(v)$ を用いて，$w_n(v)$ を計算し直し，その $w_n(v)$ により重みづけを行った $S(v)$ から $e(v)$ を計算し直すため，これまで $e(v)$ の値が低かったボクセルであっても，$e(v)$ が高い値となりうるためである．

3.2 動的負荷分散による高速化

並列三次元形状復元法 [7] により並列計算を行う場合，提案する Step 2 の処理により，各 PE が担当するボクセルの個数が変化し，計算負荷が不均一になりうる．そこで，反復更新の制御による計算の打ち切りを行った後に，ボクセル空間を再分割し各 PE の負荷の均一化を図る．

計算負荷を均一にするために，単純に各 PE の担当するボクセルの個数が均等になるようにボクセル空間を分割する方法が考えられる．しかし，一つのボクセルの計算を行う際には，式 (5) に示される通り，$\Phi(v)$ 中の全てのボクセルの情報を用いるため，ボクセルごとに必要な処理量は均等ではなく，$\Phi(v)$ 中のボクセルの個数に比例する．特に，視点 $\boldsymbol{o}_n$ からボクセル v を結ぶ直線の傾きが大きい場合には，$\Phi(v)$ 中のボクセルの個数が少なくなることが多く，そのようなボクセルを多く含む，対象物体の端などの空間を担当する PE は処理時間が短くなり，その分他の PE の負荷が大きくなるため，計算負荷が均一にはならない．

以上の考えに基づき，各 PE の担当するボクセルの $\Phi(v)$ 中のボクセルの和が均等になるようにボクセル空間を分割する手法を提案する．ボクセル空間の再分割は，Step 2 b において更新打ち切りを実行した直後に毎回行う．更新打ち切りの判定後，更新を続けるボクセルに関してのみ，視点 $\boldsymbol{o}_n$ からボクセル v を結ぶ直線の傾きの大きさの順にソートして，各 PE で担当するボクセルの $\Phi(v)$ の個数の和が等しくなるようにボクセル空間の再分割を行う．

4 評価

4.1 実験環境

対象物体を撮影した実画像を用いて，クラスタマシンで三次元形状復元を行う．その計算時間と復元精度の結果から評価を行う．実験には 4 台のカメラ（視点を $\boldsymbol{o}_1, \boldsymbol{o}_2, \boldsymbol{o}_3, \boldsymbol{o}_4$ とする）と 2 個の物体 Object 1, 2 を用い，図 4 に示すように各々配置した．各カメラで撮影し実験に用いた画像 $\boldsymbol{I}_1, \boldsymbol{I}_2, \boldsymbol{I}_3, \boldsymbol{I}_4$ を図 5 に示す（画像サイズは 696×480 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 (5),(6),(9) 中のパラメータを $\alpha = 1.5, \beta = 0.01, \gamma = 0.01$ とし，Step 2 a の反復回数を $T_{2a} = 10$，

表 1　計算機性能

CPU	AMD Opteron DP Model 246 2.0GHz
Memory	PC3200 ECC Registered DDR-SDRAM 2GB
Number of Nodes	16
OS	SuSE Linux Enterprise Server 8
Compiler	PGI Compiler
MPI Library	MPICH-SCore

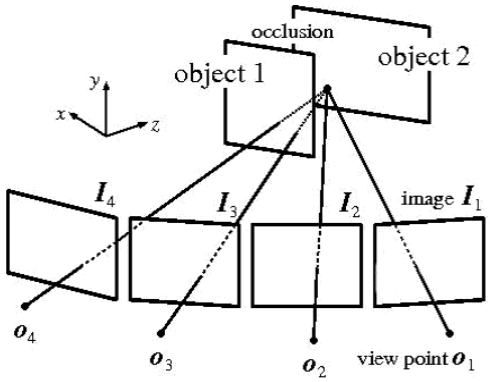


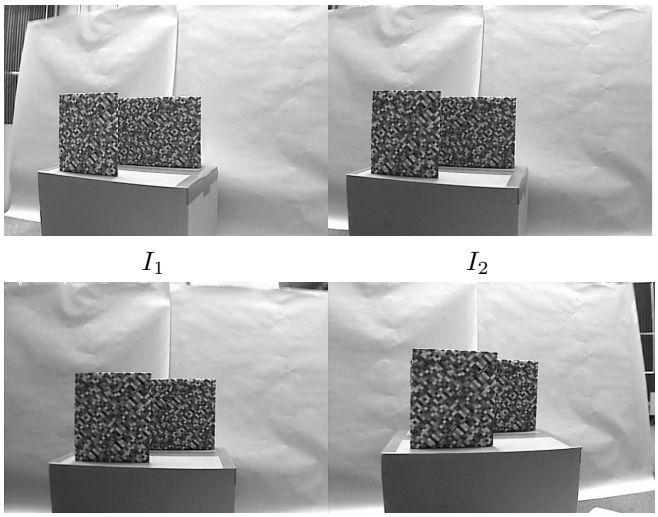
図 4　視点（$\boldsymbol{o}_1, \boldsymbol{o}_2, \boldsymbol{o}_3, \boldsymbol{o}_4$）と物体（object1, 2）の配置

Step 1, 2 の反復回数を $T_{1,2} = 2$ とした．また，実験で用いたクラスタマシンの性能を表 1 に示す．

4.2 計算時間の評価

各手法において，三次元形状の復元に要した計算時間を図 6 に示す．横軸は計算に使用した PE 数，縦軸は計算時間を表している．図 6 より，Step 2 の処理に反復更新を打ち切る処理を加えることで，1〜16 のどの PE 数の場合においても，50〜60%の計算時間の削減という大幅な高速化が実現できていることがわかる．これは，Step 2 の処理において，e の値が収束したボクセルの更新を打ち切ることで，必要以上に行っていた計算を省くことができたためである．

図 7 は，反復更新の制御を行った場合と，反復更新の制御を行った後に動的負荷分散の処理を加えた場合の二通りの手法について，分割数が 2 以上のときの計算時間を表したものである．グラフの下の数値は，更新制御及び動的負荷分散を行った際の計算時間に占めるオーバーヘッドの割合を示す．図 6 と同様，横軸は計算に使用した PE 数，縦軸は計算時間を表している．また，その二通りの手法において，PE 数を 8 とした場合の各 PE がそれぞれ要した計算時間を図 8，図 9 に示す．横軸は PE の番号を表し，縦軸は計算時間を表している．図 7 より，ボクセル空間の再分割の処理を加えることによって，さらなる計算時間の削減が実現できていることがわかる．これは，$\Phi(v)$ の個数を基にしたボクセル空間の再分割を行うことによって，反復更新の制御により差が生じた各 PE 間の負荷を均等にならしているためである．実際に，図 8，図 9 を比較してみると，負荷の不均衡が解消されていることが確認できる．図 8 では，PE_0,PE_6,PE_7 の計算時間が短くなっており，他の PE の計算時間が長くなっている．これは，PE_0,PE_6,PE_7 は復元対象のボクセル空間の上端および下端を担当している PE で，割



I_1 I_2 I_3 I_4

図 5　実験に用いた入力画像

表 2　PE 数 8 の場合における各 PE の計算時間の分散

	分散
ボクセル再分割なし	45.7
ボクセル再分割あり	3.44

り当てられているボクセルの内，$\Phi(v)$ に含まれるボクセルの個数が少なく負荷が小さいボクセルの割合が相対的に大きいためである．一方の，図 9 では，各 PE の計算時間にほとんど差はみられない．表 2 は，ボクセル再分割の処理を加えない場合と加えた場合のそれぞれにおいて，各 PE の計算時間の分散を求めたものである．表 2 の値からも，各 PE の負荷の均一化が行われていることがわかる．以上のことから，$\Phi(v)$ の個数に基づくボクセル空間の再分割によって，各 PE の負荷の均一化を実現できていることがわかる．また，更新制御及び動的負荷分散を行った際のオーバーヘッドも計算時間の削減量に対して少なく，これらの手法は並列三次元形状復元法の高速化を行うために有効な手段であると言える．

4.3　復元精度の評価

復元精度の評価を行うことで，更新打ち切りによる復元結果への影響を調べる．

並列三次元形状復元法及び，提案手法によって得られた復元結果に対し，実際の物体表面に対応するボクセルの集合を A，各手法により物体境界と判定されたボクセルの集合を B とし，以下の値により評価を行う．

(1)A に対し物体表面を判定できた割合 R

$$R = \frac{N(B \text{ と一致する } A)}{N(A)} \tag{10}$$

(2)B に対し物体表面を判定できた割合 P

$$P = \frac{N(A \text{ と一致する } B)}{N(B)} \tag{11}$$

ただし，$N(Z)$ は集合 Z 中のボクセル数とする．

なお，ここでは，更新の制御方法について，式 (9) 中の γ の値を，$\gamma = 0.001$，$\gamma = 0.01$，$\gamma = 0.1$ とした場合の三通りについて評価を行う．図 10 は，それぞれの更新の制御方法における計算時間を示す．横軸は，計算に使用した PE 数を表し，縦軸は計算時間を表している．図 10 より，更新打ち切り条件の閾値を高くした方が，計算時間は短くなっていることがわかる．これは，閾値が高い方が，更新を打ち切られるボクセルが多くなり，計算量の減少が多くなるためである．

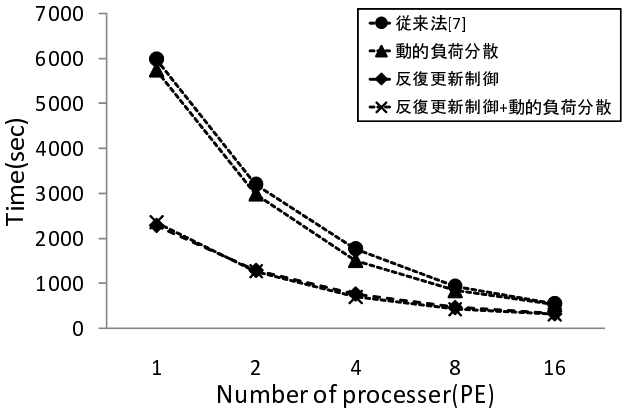


図 6　各手法における計算時間と PE 数の関係

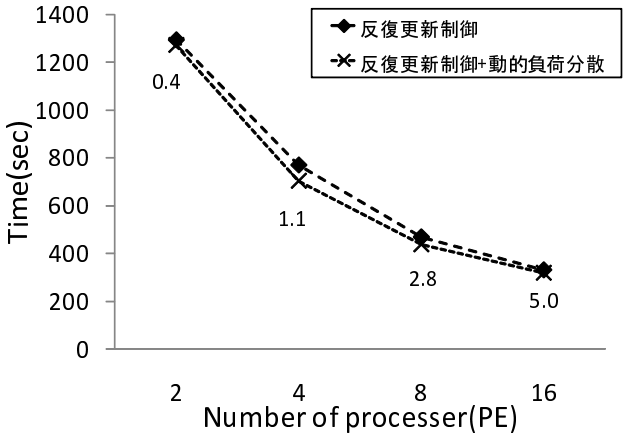


図 7　動的負荷分散の有無による計算時間の変化

表 3 は，並列三次元形状復元法と，反復更新の制御を加えた手法でそれぞれ復元を行った場合の復元精度を示している．R の値が総じて低いのは，形状復元に用いた画像の枚数が少ないため，どの画像にも映っていない Object 2 の端を復元できなかったことによる．表 3 をみると，並列三次元復元法と提案手法の復元精度には，R，P いずれの値にも大きな差はみられない．このことから，e の値が収束したボクセルの反復更新を打ち切る手法は，復元精度を保ったまま高速化を実現できるということが示せる．更新を打ち切る際の閾値の変化による復元精度にも大差はみられなかった．

図 11，図 12 は，それぞれ更新制御を行わない場合と，$\gamma = 0.01$ として更新制御を行った場合での三次元形状復元結果を表す．図 11，図 12 を比較してみても，更新制御の有無によって三次元形状復元結果には大きな影響を及ぼさないということが確認できる．

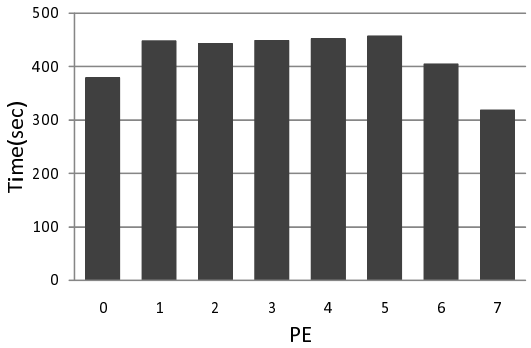


図 8　各 PE の計算時間-更新打ち切りのみ

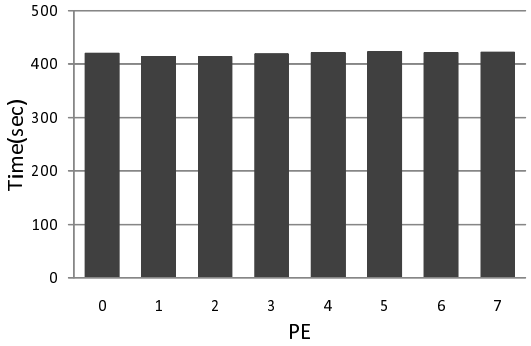


図 9　各 PE の計算時間-ボクセル再分割あり

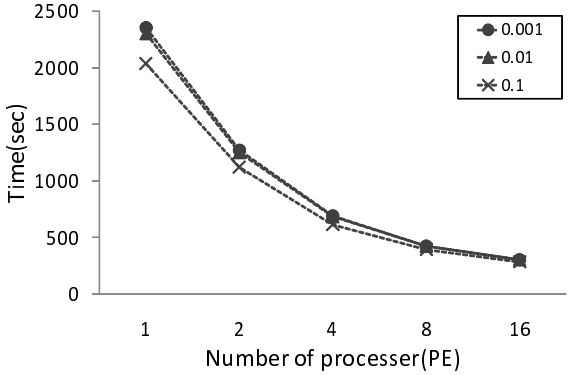


図 10　更新制御方法 (γ) の違いによる計算時間の変化

5　まとめ

本稿では，高橋らの提案した並列三次元形状復元手法 [7] に対し，以下の 2 点の手法を提案することによって，高速化を図った．(1) 物体境界である確からしさの反復更新を行う過程において，値が収束したボクセルの更新を打ち切ることで，計算量を削減する．(2) 反復更新の制御により生じる PE 間の計算負荷の不均衡に対処するために，ボクセルの更新に必要なデータ量を示す $\Phi(v)$ 中のボクセル個数に基づきボクセル空間の分割を行い，動的に負荷分散を行う．(1) の手法においては 50〜60% の計算時間の削減が実現でき，(2) の手法においても負荷の不均衡を解消し，三次元形状復元の精度を保ちながら，高速化を実現することができた．

今後の課題として，今回，カメラの台数は 4 台でかつ水平に並んでいるものとしたが，カメラの台数や配置を変化させた場合に対応できる並列化手法等について検討を進める必要がある．また，復元対象の物体をより複雑なものへと変更した場合の提案手法の効果について評価を行う必要がある．

表 3　更新打ち切りによる復元結果の変化

	R	P	N(B)
並列三次元形状復元法	0.581	0.905	2265
更新制御（$\gamma = 0.001$）	0.579	0.910	2255
更新制御（$\gamma = 0.01$）	0.586	0.913	2257
更新制御（$\gamma = 0.1$）	0.587	0.912	2230

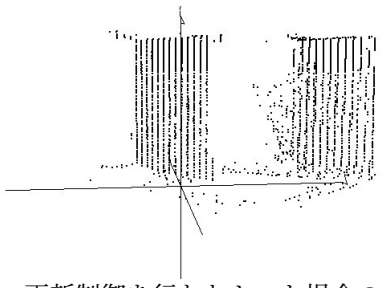


図 11　更新制御を行わなかった場合の復元結果

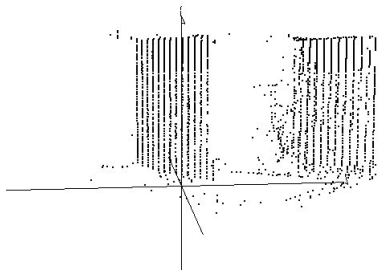


図 12　更新制御 ($\gamma = 0.01$) を行った場合の復元結果

参考文献

[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] 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", MIRU2006 論文集, pp.483–488,2006.

[3] H. Saito and T. Kanade, "Shape reconstruction in projective grid space from large number of images", IEEE Computer Society Conf. Computer Vision and Pattern Recognition, vol.2, pp.49–54, 1999.

[4] J.-S. Franco and E. Boyer, "Fusion of multiview silhouette cues using a space occupancy grid", Tenth IEEE Int. Conf. Computer Vision, vol.2, pp.1747–1753, 2005.

[5] Z. Chen and H.-L. Chou, "New Efficient Octree Construction from Multiple Object Silhouettes with Construction Quality Control", 18th Int. Conf. Pattern Recognition, vol.1, pp.127–130, 2006.

[6] R. Szeliski, "Rapid Octree Construction from Image Sequences", CVGIP: Image Understanding, vol.58, no.1, pp.23–32, 1993.

[7] 高橋一樹, 福士将, 阿部亨, 堀口進, "多眼ステレオ三次元形状計測の並列計算法", 信学技報 IE2007-134,Vol.107, No.379, pp.129–134, 2007.