情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2012-CVIM-181 No.19
2012/3/16

物体形状・反射特性・光源の同時推定の実装法と評価

右田剛史[†1]　槌谷洋介[†1]　尺長健[†1]

物体形状・反射特性・光源の同時推定は未知数が十万規模の非線形最適化であり，効率的なアルゴリズムと実装を要する．Levenberg-Marquardt 法による解法で特に実行時間を要するのは，探索方向を決定する十万元規模の線形連立方程式，及び，その係数行列 (ヘッセ行列) の計算である．本稿では，マルチコア CPU や GPU を利用した幾つかの計算法を述べ，実画像で性能を比較する．その結果，最大で数倍の性能が得られることを確認した．

Implementation and Evaluation of Simultaneous Estimation of Shape, Reflectance and Lighting

Tsuyoshi Migita,[†1] Yosuke Tsuchiya[†1]
and Takeshi Shakunaga[†1]

Estimation of shape, reflectance and lighting is a numerical optimization with $O(10^5)$ unknown parameters. A Levenberg-Marquardt approach to this problem requires a careful implementation of a sophisticated algorithm. Several algorithms and their parallel implementation on a multi-core CPU or a GPU are compared to show a substatial performance gain on real image sequences.

1. はじめに

複数の画像を入力として，撮影された物体の形状を推定する様々な手法が研究されている．本稿では図 1 の様な画像列を用いる未校正照度差ステレオ法を扱う．入力画像は，カメラと物体を暗室内に固定し，移動する点光源により照明条件を変えて撮影した画像である．

†1 岡山大学
Okayama University

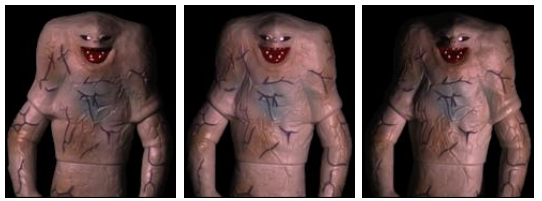

図 1　光源を変化させて撮影した画像列

図 2　復元形状

ここから図 2 の様な形状を推定する．この時，光源の位置も未知であるため推定する必要がある．また，一般に物体表面の反射率は一定ではないため，表面の反射率も推定する．これは，推定パラメータに基づく合成画像と入力画像との差を最小化する非線形最適化問題である[1),2)]．物体表面の反射モデルは Torrance-Sparrow モデルを仮定する．

本稿で想定する問題規模は，画素数 $P = 20000$，画像数 $F = 30$ 程度である．これは，図 1 の画像列の問題規模であるが，一部の実験では $P = 60000$ 程度の問題も扱う．推定すべきパラメータは各画素に対して 5 つ，各画像に対して 3 つ，全体に対して 4 つあるため，10 万元規模の線形連立方程式を繰り返し解く必要がある．本稿では，この問題の効率的な解法について議論する．

なお，文献 1)，2) の手法の改良を目指した文献 3)，4) の手法では，画像の類似度の計算の際に残差の RGB ベクトルのうち光源色に直交する 2 次元成分のみを使って評価する．この目的関数でも計算法の大部分は本来の手法と共通であるので，この手法についても併せて述べる．

まず，2 章で本稿の問題の定式化と解法の概略を示し，3 章以降で解法の詳細について実例を交えて述べる．ここで用いる数値解析アルゴリズムの基礎は付録で述べる．

2. 物体形状・反射特性・光源の同時推定

2.1 定式化

静止物体を静止カメラで撮影した F 枚の画像が与えられている．画像毎に光源位置 $\boldsymbol{l}_f$ が異なる．また，光源色は全画像に共通で $\boldsymbol{s}$ とする．

画素は対象物体と背景に分けられ，対象物体上の推定対象画素数を P とする．物体形状は，各画素に対応する面素までの距離 (depth) λ_p で表す．面素の色 (反射率) を表す RGB ベクトルを $\boldsymbol{d}_p = (w_{0p}, w_{1p}, w_{2p})^T$ とし，鏡面反射の強度を w_{3p} とする．面素の座標 $\boldsymbol{x}_p$ は

距離値とカメラモデルから計算される．

カメラから物体側を z 軸の正の方向とし，焦点距離を ℓ，位置を $\boldsymbol{v} = (0, 0, -\ell)^T$ とする．このカメラモデルでは，ℓ は射影歪効果を調整するパラメータであり，像の大きさは変化しない．このため，$\ell \to \infty$ として平行投影を扱うこともできる (ℓ は常に逆数で用いられるため，実装上は ℓ^{-1} を表す変数を用いる)．画像座標系の原点は画像中心とし，画素の座標を (x_p, y_p) とすると，画素の座標と画素からカメラへの方向 $\boldsymbol{v}'_p$ は次の様になる．

$$\boldsymbol{x}_p = (x_p, y_p, 0)^T + \lambda_p \boldsymbol{u}_p \ , \quad \text{ただし}, \quad \boldsymbol{u}_p = (x_p/\ell, y_p/\ell, 1)^T \tag{1}$$

$$\boldsymbol{v}'_p = (\boldsymbol{v} - \boldsymbol{x}_p)/\ell \ = \ -(1 + \lambda_p/\ell)\boldsymbol{u}_p \tag{2}$$

面素の法線ベクトル $\boldsymbol{n}_p$ は近傍の点の座標から計算される (これは単位長ではない)．

$$\boldsymbol{n}_p = (\boldsymbol{x}_R - \boldsymbol{x}_L) \times (\boldsymbol{x}_T - \boldsymbol{x}_B) \tag{3}$$

ここで，L,R,T,B は着目点 p の左右上下の点を表すが，物体境界外になる場合は代わりに p 自身を表す．例えば，p の右の画素が推定対象外なら $R = p$ である．このため，幅が 2 に満たない領域は推定対象としない．

画像の輝度値 $\boldsymbol{e}_{fp}$ は Torrance-Sparrow モデルによって次の様に表される．

$$\boldsymbol{e}_{fp} = \eta_{fp} \left(\boldsymbol{d}_p D_{fp} + w_{3p} \boldsymbol{s} S_{fp} \right) \ , \tag{4}$$

$$\text{ただし}, \quad D_{fp} = \max\left(0, \frac{\boldsymbol{n}_p^T}{|\boldsymbol{n}_p|} \frac{\boldsymbol{l}'_{fp}}{|\boldsymbol{l}'_{fp}|}\right) \ , \quad S_{fp} = \frac{1}{\frac{\boldsymbol{n}_p^T}{|\boldsymbol{n}_p|} \frac{\boldsymbol{v}'_p}{|\boldsymbol{v}'_p|}} \exp\left(\rho \alpha^2\right) \tag{5}$$

D_{fp}, S_{fp} はそれぞれ拡散反射，鏡面反射の強度を表し，ρ は鏡面反射の広がりを表す全面素に共通のパラメータである．また，以下の定義を用いた．

$$\eta_{fp} = |\boldsymbol{l}'_{fp}|^{-2} \ , \quad \boldsymbol{l}'_{fp} \ = \ \boldsymbol{l}_f - \boldsymbol{x}_p, \tag{6}$$

$$\alpha_{fp} = \arccos\left(\frac{\boldsymbol{n}_p^T}{|\boldsymbol{n}_p|} \frac{\boldsymbol{b}_{fp}}{|\boldsymbol{b}_{fp}|}\right) \ , \quad \boldsymbol{b}_{fp} \ = \ \frac{\boldsymbol{l}'_{fp}}{|\boldsymbol{l}'_{fp}|} + \frac{\boldsymbol{v}'_p}{|\boldsymbol{v}'_p|}. \tag{7}$$

この画素値の推定値と入力画像を比較することで形状・反射特性・光源の推定が可能である．推定の目的関数 E は次の様に表される．

$$E(\boldsymbol{p}) = (1/2) \sum_{(f,p)} \boldsymbol{r}_{fp}^T \boldsymbol{r}_{fp} \quad \text{where} \quad \boldsymbol{r}_{fp} = \boldsymbol{e}_{fp} - \boldsymbol{e}'_{fp} \tag{8}$$

ここで，$\boldsymbol{p}$ は全推定パラメータを含むベクトル (以後，その要素数を N とする)，$\boldsymbol{e}'_{fp}$ は入力画像の輝度値 (RGB) である．なお，総和の添字 (f, p) は，必ずしも直積 $[0, 1, \cdots, F-1] \times [0, 1, \cdots, P-1]$ の全要素ではない．モデルに従わない影画素や飽和画素を閾値処理で判定し，幾つかの (f, p) の組み合わせは除外される．

2.2 非線形最小 2 乗法の解法

関数 $E(\boldsymbol{p})$ を最小化する Levenberg-Marquardt 法[8](以下，L-M 法) を考える．この手法は，現在の推定値 $\boldsymbol{p}$ における E の勾配 $\boldsymbol{g}$ とヘッセ行列 $\boldsymbol{H}$ を用いて次式に従って推定値を更新する手法である．矢印 $\leftarrow$ は変数の更新を表す．

$$\boldsymbol{p} \leftarrow \boldsymbol{p} - (\boldsymbol{H} + \mu \boldsymbol{I})^{-1} \boldsymbol{g} \tag{9}$$

ここで，μ は発散を防ぐために用いられる適当なパラメータである．

関数 E が残差 2 乗和の場合，残差ベクトル $\boldsymbol{r}$ を使って $E(\boldsymbol{p}) = (1/2)\boldsymbol{r}^T\boldsymbol{r}$ と書くことができる． $\boldsymbol{r}$ の第 i 要素を $\boldsymbol{p}$ の第 j 要素で偏微分した導関数 $(\partial r_i/\partial p_j)$ を i 行 j 列要素とするヤコビ行列 $\boldsymbol{J}$ を用いて，$\boldsymbol{r}(\boldsymbol{p} + \delta\boldsymbol{p}) \approx \boldsymbol{r}(\boldsymbol{p}) + \boldsymbol{J}\delta\boldsymbol{p}$ と近似できる．これを用いて，関数 E は $E(\boldsymbol{p} + \delta\boldsymbol{p}) \approx (1/2)\boldsymbol{r}^T\boldsymbol{r} + (\delta\boldsymbol{p})^T\boldsymbol{g} + (1/2)(\delta\boldsymbol{p})^T\boldsymbol{H}\delta\boldsymbol{p}$ と近似できる．ここで，ヘッセ行列 $\boldsymbol{H}$ と勾配 $\boldsymbol{g}$ は次のように表される．

$$\boldsymbol{H} = \boldsymbol{J}^T\boldsymbol{J} \ , \quad \boldsymbol{g} = \boldsymbol{J}^T\boldsymbol{r} \tag{10}$$

なお，厳密なヘッセ行列は $\sum_k [(\partial_i r_k)(\partial_j r_k) + r_k(\partial_i \partial_j r_k)]$ であり，$\boldsymbol{J}^T\boldsymbol{J}$ は第 2 項を無視した近似 (Gauss-Newton 近似) である．

なお，式 (9) で発散を防ぐ μ を動的に調節する方法[8] は大規模連立方程式の実行回数が増えるので，本稿では用いない．代わりに，次の様に重み α を導入し，これを推定する直線探索を行う．

$$\boldsymbol{p} \leftarrow \boldsymbol{p} - \alpha(\boldsymbol{H} + \mu \boldsymbol{I})^{-1} \boldsymbol{g} \tag{11}$$

探索方向に沿う 3 点で目的関数と値が一致する 2 次関数を最小にする α を選ぶ．文献 2) では直線探索により目的関数の悪化を防ぎ，共役方向補正によって反復回数の削減を図っているが，今回の実験では共役方向補正は収束速度を低下させたため，直線探索のみを用いる．

2.3 初期値と多段階推定

文献 3) では，初期形状を 2 次曲面とし，初期光源位置は物体正面の 1 点に集中させている．これは極めて粗い初期値であるが，複数の実験例で局所解に陥ることなく機能している．一方，文献 1) では，最初に特異値分解を用いて法線と光源の推定を行い初期値に利用している．但し，分解の不定性の解消は特定の光源設定に特化した方法を用いている．一般の場合では積分制約[6] によって不定性が減るが，3 自由度の GBR 不定性が残るため完全な自動化は難しい．本稿では，前者の単純な初期値を用いる．

推定の安定性や速度のために，文献 1)，2) では推定を複数の段階に分けて行っている．本稿では，この手法を多少変形した次の方法を用いる．まず,「ステップ 1」では鏡面反射項を除く拡散反射のみのモデルで推定を行う．また，大きな μ 値を使うことで局所解を避ける．

表 1 計算時間の比較

手法	$\boldsymbol{H}, \boldsymbol{g}$	探索方向	直線探索
mtd1.CPU.single	290	1470–1490	155-210
mtd1.CPU.multi	60–75	↓	30–50
mtd1.GPU	↓	300	↓
mtd2.CPU.single	290	830	155–206
mtd2.CPU.multi	50–110	730–780	30–50
mtd2.GPU	↓	200	↓

L-M 法の更新一回当たりの処理時間．単位は msec.

表 2 使用計算機と性能

	CPU Core-i7 3930K	GPU GeForce GTX 580
コア数	6	512
周波数 GHz	3.2	1.54
倍精度 Gflops	154	192
メモリ GB/s	21	192

次に,「ステップ 2」では鏡面反射項を含むモデルを使って推定を行う．また，μ を段階的に減少させ速い収束を目指す．

2.4 各部分の計算時間の削減

大規模問題においては，式 (9) の探索方向の計算，式 (10) のヘッセ行列と勾配の計算，式 (11) の直線探索を効率的に計算する必要がある．これらに掛かる実際の時間を表 1 に示す．入力画像は図 1，用いた計算機を表 2 に示す．表 1 の最上行が本稿で最も単純な計算法による結果であり，他の行は以下の章で述べる方法により効率化した結果である．最上行の手法では 1 反復当たり 2 秒で，その 3/4 が探索方向の計算 $(\boldsymbol{H}+\mu\boldsymbol{I})^{-1}\boldsymbol{g}$ に費やされている．4 章で述べる手法によりこの計算時間は 1/5 から 1/8 に削減できる．この時，$\boldsymbol{H}, \boldsymbol{g}$ の計算や直線探索の時間が無視できなくなるため，これらの改善も必要である．これらについては，それぞれ 3 章と 5 章で述べる．表の最下行は，これら全ての改善後の計算時間を示しており，1 反復当たりの 0.35 秒程度に削減されている．

3. ヘッセ行列と勾配の計算

$\boldsymbol{r}_{fp}$ を全てつなげた残差ベクトル $\boldsymbol{r}$ の次元数は最大 $3FP$ であり，図 1 の問題規模では 10^6 程度，$\boldsymbol{J}$ は $10^6 \times 10^5$ 程度の巨大な行列であるが，一度に全要素を必要とすることはない．まず，次の様に 3 行毎に分けて考える．

$$\boldsymbol{H} = \sum_{(f,p)} \boldsymbol{J}_{fp}\boldsymbol{J}_{fp}^T\ ,\quad \boldsymbol{g} = \sum_{(f,p)} \boldsymbol{J}_{fp}\boldsymbol{r}_{fp}\ ,\quad \boldsymbol{J} = (\boldsymbol{J}_{00}^T|\boldsymbol{J}_{01}^T|\cdots|\boldsymbol{J}_{F-1,P-1}^T)^T \qquad (12)$$

この $\boldsymbol{J}_{fp}$ は $3\times N$ 行列であるが，$\boldsymbol{r}_{fp}$ に影響を及ぼすパラメータは 16 個のみであるため，$\boldsymbol{J}_{fp}$ の 16 行のみが非零要素を含んでおり，それらを 3×16 行列に圧縮して格納することができる．この行列を $\hat{\boldsymbol{J}}_{fp}$ と書く．式 (12) の総和では，各 f,p に対してそれぞれ $\hat{\boldsymbol{J}}_{fp}^T\hat{\boldsymbol{J}}_{fp}$ の 136$(=1+2+\cdots+16)$ 個の要素と $\hat{\boldsymbol{J}}_{fp}^T\boldsymbol{r}_{fp}$ の 16 個の要素を算出し，$\boldsymbol{H}$ と $\boldsymbol{g}$ の然るべき場所に加算する．

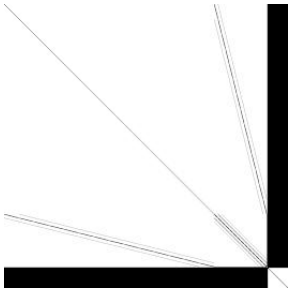

図 3 $\boldsymbol{H}$ の例

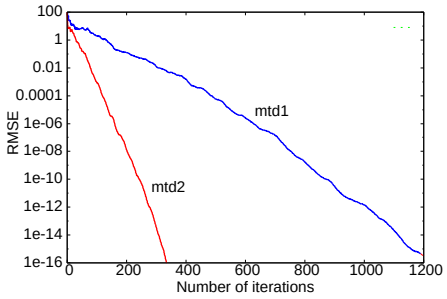

図 4 反復回数と残差

表 3 $\boldsymbol{H}$ のブロック名

$\boldsymbol{H}_{WW}$	$\boldsymbol{H}_{WS}$	$\boldsymbol{H}_{WM}$	$\boldsymbol{H}_{WL}$
$\boldsymbol{H}_{SW}$	$\boldsymbol{H}_{SS}$	$\boldsymbol{H}_{SM}$	$\boldsymbol{H}_{SL}$
$\boldsymbol{H}_{MW}$	$\boldsymbol{H}_{MS}$	$\boldsymbol{H}_{MM}$	$\boldsymbol{H}_{ML}$
$\boldsymbol{H}_{LW}$	$\boldsymbol{H}_{LS}$	$\boldsymbol{H}_{LM}$	$\boldsymbol{H}_{LL}$

表 4 ブロック毎の非零要素数

10P	*	*	*
20P	7 P	*	*
16P	4 P	10	*
12FP	3FP	12F	6F

表 5 スレッド分割

$\mathcal{P}_1$
$\mathcal{P}_2$
⋮
$\mathcal{P}_T$

3.1 $\boldsymbol{H}$ の構造

図 3 に $\boldsymbol{H}$ の例を示す．$P=196$, $F=16$ の極めて小さな例で大きさは 1059×1059 である．白部分が零，黒部分が非零要素を表す．疎な対角ブロックと密な非対角部分があることが分かる．行列 $\boldsymbol{H}$ は 2 階導関数 $\partial^2 E/(\partial p_i \partial p_j)$ から成る行列であり，p_i と p_j が属するグループに従って異なる構造をしている． $\boldsymbol{p}$ のパラメータは下記の 4 グループに分類できる．便宜上，各グループにアルファベット一時文字の符号を割り当てる．これに従い $\boldsymbol{H}$ を 4×4 に分割した部分行列名を表 3 に列挙する．

(W) 反射率: $\boldsymbol{w}_p = (w_{0p}, w_{1p}, w_{2p}, w_{3p})^T$
(S) 形状: λ_p
(M) 共通パラメータ: $\boldsymbol{s}, \rho$
(L) 光源位置: $\boldsymbol{l}_f$

$\boldsymbol{H}$ は疎な対称行列であることを考慮して記憶領域を圧縮する．表 4 に各ブロックの非零要素数を示す (厳密に言えば，これらの数値は $\boldsymbol{H}$ の非零要素の格納場所を単純な規則で計算するために配置される空き領域を含めたメモリ量である)．例えば $\boldsymbol{H}_{LL}$ は 3×3 ブロックが対角に F 個並んだ対称行列で，$6F$ 個の独立要素がある．$\boldsymbol{H}_{MM}$ は 4×4 対称行列で自由度は 10 である， $\boldsymbol{H}_{SS}$ は $P\times P$ 行列で大きいが，第 (i,j) 要素は第 i 画素と第 j 画素

の市街地距離が 2 以下 (表 6) の場合のみ非零であるから，非零要素は一行あたり 13 個以下である (残差 $\boldsymbol{r}_{fp}^T\boldsymbol{r}_{fp}$ は着目点 p と 4 近傍の画素から計算され，市街地距離が 3 以上離れた画素を用いることはないため)．対称であるから $7P$ 個の領域があれば全要素を格納することができる (このうち幾つかの領域は未使用である)．

図 1 の問題規模では $\boldsymbol{H}$ の非零要素数は約 10^7 個，倍精度実数で約 80MBytes である．

3.2 $\hat{\boldsymbol{J}}_{fp}$ の導出

以下，$\bar{\partial}$ は複数のパラメータによる偏微分演算子からなる行ベクトルを表す (パラメータが x_1, x_2 なら $\bar{\partial}$ は $(\partial/\partial x_1\ \partial/\partial x_2)$ で，$\bar{\partial}(x_1, x_2)^T$ は単位行列である)．$\boldsymbol{p}$ の全パラメータに関する $\bar{\partial}$ を用いると，$\boldsymbol{J}_{fp} = \bar{\partial}\boldsymbol{r}_{fp}$ と書くことができる．実際には，$\boldsymbol{p}$ のうち $\boldsymbol{r}_{fp}$ に影響する 16 パラメータ $(\boldsymbol{w}_p, \lambda_L, \lambda_R, \lambda_p, \lambda_T, \lambda_B, \boldsymbol{s}, \rho, \boldsymbol{l}_f)$ に関する $\bar{\partial}$ 演算子を用いて，$\hat{\boldsymbol{J}}_{fp} = \bar{\partial}\boldsymbol{r}_{fp}$ を考えれば良い．まず，定義 (4) から次のように変形する．

$$\begin{aligned}\bar{\partial}\boldsymbol{r}_{fp} = {} & \bar{\partial}\eta_{fp}\left(\boldsymbol{d}_p D_{fp} + w_{3p}\boldsymbol{s}S_{fp}\right) + \\ & \eta_{fp}\left(\bar{\partial}\boldsymbol{d}_p D_{fp} + \boldsymbol{d}_p\bar{\partial}D_{fp} + \bar{\partial}w_{3p}\boldsymbol{s}S_{fp} + w_{3p}\bar{\partial}\boldsymbol{s}S_{fp} + w_{3p}\boldsymbol{s}\bar{\partial}S_{fp}\right)\end{aligned} \tag{13}$$

ここで，

$$\bar{\partial}\eta_{fp} = -2|\boldsymbol{l}'_{fp}|^{-4}(\boldsymbol{l}_f - \boldsymbol{x}_p)^T(\bar{\partial}\boldsymbol{l}_f - \bar{\partial}\boldsymbol{x}_p)\ , \tag{14}$$

$$\bar{\partial}D_{fp} = \begin{cases} \dfrac{\boldsymbol{l}'^T_{fp}}{|\boldsymbol{l}'_{fp}|}\dfrac{\boldsymbol{n}_p^\perp}{|\boldsymbol{n}_p|}\bar{\partial}\boldsymbol{n}_p + \dfrac{\boldsymbol{n}_p^T}{|\boldsymbol{n}_p|}\dfrac{\boldsymbol{l}'^\perp_{fp}}{|\boldsymbol{l}'_{fp}|}(\bar{\partial}\boldsymbol{l}_f - \bar{\partial}\boldsymbol{x}_p) & \text{if}\ \ \dfrac{\boldsymbol{n}_p^T}{|\boldsymbol{n}_p|}\dfrac{\boldsymbol{l}'_{fp}}{|\boldsymbol{l}'_{fp}|} > 0 \\ 0 & \text{otherwise,}\end{cases} \tag{15}$$

$$\begin{aligned}\bar{\partial}S_{fp} = {} & \frac{1}{\dfrac{\boldsymbol{n}_p^T}{|\boldsymbol{n}_p|}\dfrac{\boldsymbol{v}'_p}{|\boldsymbol{v}'_p|}}\exp\left(\rho\alpha^2\right)\left\{\alpha^2\bar{\partial}\rho\right. \\ & - \frac{1}{\dfrac{\boldsymbol{n}_p^T}{|\boldsymbol{n}_p|}\dfrac{\boldsymbol{v}'_p}{|\boldsymbol{v}'_p|}}\left(\frac{\boldsymbol{v}'^T_p}{|\boldsymbol{v}'_p|}\frac{\boldsymbol{n}_p^\perp}{|\boldsymbol{n}_p|}\bar{\partial}\boldsymbol{n}_p + \frac{\boldsymbol{n}_p^T}{|\boldsymbol{n}_p|}\frac{\boldsymbol{v}'^\perp_p}{|\boldsymbol{v}'_p|}(-\bar{\partial}\boldsymbol{x}_p/\ell)\right) \\ & \left. - \frac{2\rho\alpha}{\sin\alpha}\left(\frac{\boldsymbol{b}_{fp}^T}{|\boldsymbol{b}_{fp}|}\frac{\boldsymbol{n}_p^\perp}{|\boldsymbol{n}_p|}\bar{\partial}\boldsymbol{n}_p + \frac{\boldsymbol{n}_p^T}{|\boldsymbol{n}_p|}\frac{\boldsymbol{b}_{fp}^\perp}{|\boldsymbol{b}_{fp}|}\bar{\partial}\boldsymbol{b}_{fp}\right)\right\}\ ,\end{aligned} \tag{16}$$

ただし，$\bar{\partial}\boldsymbol{b}_{fp} = \left(\dfrac{\boldsymbol{l}'^\perp_{fp}}{|\boldsymbol{l}'_{fp}|}(\bar{\partial}\boldsymbol{l}_f - \bar{\partial}\boldsymbol{x}_p) - \dfrac{\boldsymbol{v}'^\perp_p}{|\boldsymbol{v}'_p|}\bar{\partial}\boldsymbol{x}_p/\ell\right)$

また，$\boldsymbol{x}^\perp = I - \boldsymbol{x}\boldsymbol{x}^T/(\boldsymbol{x}^T\boldsymbol{x})$ である．

更に，次式を使って面素の位置 $\boldsymbol{x}_p$ と法線 $\boldsymbol{n}_p$ を形状パラメータ λ_* に帰着させる．

$$\bar{\partial}\boldsymbol{x}_p = \bar{\partial}\lambda_p\boldsymbol{u}_p\ , \tag{17}$$

$$\bar{\partial}\boldsymbol{n}_p = \left(\bar{\partial}\lambda_R\boldsymbol{u}_R - \bar{\partial}\lambda_L\boldsymbol{u}_L\right) \times \left(\bar{\partial}\lambda_T\boldsymbol{u}_T - \bar{\partial}\lambda_B\boldsymbol{u}_B\right) \tag{18}$$

3.3 拡散反射のみの場合

「ステップ 1」で用いられる鏡面反射項を除く簡単な画像生成モデルは次式の通りである．

$$\boldsymbol{e}_{fp} = \eta_{fp}\boldsymbol{d}_p D_{fp} \tag{19}$$

この定義の下では $\boldsymbol{r}_{fp}$ に影響するパラメータは 11 個である．これらに関する導関数は式 (13) から S_{fp} に関する項を除いた 3 項であり，式 (16) は現れない．

3.4 SR 法の場合

文献 3)，4) では，式 (8) による推定を安定化するために鏡面反射項を除去する．この場合の残差ベクトルは次の様に定義される．

$$\boldsymbol{r}_{fp} = \boldsymbol{s}_\times\boldsymbol{r}'_{fp}\ ,\ \ \text{ただし，}\boldsymbol{r}'_{fp} = \frac{\eta_{fp}\boldsymbol{d}_p D_{fp} - \boldsymbol{e}'_{fp}}{|\boldsymbol{s}_\times\boldsymbol{m}_p|}\ ,\quad \boldsymbol{m}_p = \mathcal{N}\left[\sum_f \boldsymbol{e}'_{fp}\right] \tag{20}$$

この場合，次の様になる．

$$\bar{\partial}\boldsymbol{r}_{fp} = \left\{-\left[\boldsymbol{r}'_{fp}\right]_\times + \frac{\boldsymbol{r}_{fp}\boldsymbol{s}^T\boldsymbol{m}^\perp}{|\boldsymbol{s}_\times\boldsymbol{m}_p|^2}\right\}\bar{\partial}\boldsymbol{s} + \frac{\boldsymbol{s}_\times}{|\boldsymbol{s}_\times\boldsymbol{m}_p|}\left\{\bar{\partial}\left(\eta_{fp}\boldsymbol{d}_p D_{fp}\right)\right\} \tag{21}$$

右辺第 2 項の括弧内は，拡散反射のみの場合と同じである．

3.5 効率的な計算方法

$\boldsymbol{J}_{fp}$ の計算は p 毎に独立であるから，マルチコア CPU 上で複数のスレッドを用いて並列計算を行うことができる．このために対象画素集合 $\mathcal{P}$ を $\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_T$ に分ける．本稿では画素数がほぼ均等になるように画像を縦に $T(=12)$ 分割した (表 5)．

この時，$\boldsymbol{H}$ と $\boldsymbol{g}$ を T 組用意して各スレッドで独立に式 (12) の部分和を計算した後に総和を計算すればヘッセ行列と勾配が得られるが，$\boldsymbol{H}$ は 80MBytes 程度の巨大なデータであるため，この様な領域の多重化は必要最小限に留めるべきである．即ち，計算が 1 つの部分和によって完結する部分 ($\boldsymbol{w}_p$ に関する導関数と λ_p に関する導関数の大部分) と，2–3 の部分和にまたがる部分 (表 5 の境界付近の λ_p に関する導関数) と，全部分和が関わる部分があることを考慮し，最小限の記憶容量と計算量に留める．

また，f に依存しないパラメータに関しては，$\boldsymbol{J}$ を 3 行毎に細分化するのではなく，全光源をまとめて計算する方が効率的に計算できる．即ち，

$$\boldsymbol{H} = \sum_p \boldsymbol{J}_p^T\boldsymbol{J}_p\ ,\quad \boldsymbol{g} = \sum_p \boldsymbol{J}_p^T\boldsymbol{r}_p\ ,\quad \boldsymbol{J}_p^T = (\boldsymbol{J}_{0p}^T|\boldsymbol{J}_{1p}^T|\cdots|\boldsymbol{J}_{F-1,p}^T) \tag{22}$$

ここで，$(\boldsymbol{w}_p, \lambda_{\{p,R,L,T,B\}}, \rho, \boldsymbol{s})$ に対応する行に関して $\boldsymbol{J}_p$ は密行列である．

表 1 の 2 行目と 5 行目に示した結果によると，6 コア CPU を使って計算時間が約 1/5 か

ら 1/3 に削減されている．CPU での並列計算は実行時間の変動が大きいが，平均的に 4 倍以上の性能を達成している．なお，GPU による $\boldsymbol{H}$ や $\boldsymbol{g}$ の計算も考えられるが，現時点では未実装である．GPU では CPU よりも粒度の細かい並列化となるため，$\boldsymbol{H}$, $\boldsymbol{g}$ の部分和の計算と統合が CPU の場合よりも複雑になると考えられるためである．

4. 探索方向の計算

連立方程式 $(\boldsymbol{H}+\mu\boldsymbol{I})\boldsymbol{d}=\boldsymbol{g}$ の未知数は 10^5 個規模であり，前処理付き共役勾配法 (PCG 法，付録を参照) によって解を探索する．L-M 法 (あるいは Newton 法) の性質上，探索の初期段階で推定値が解から遠い時は $\boldsymbol{d}$ を低い精度で計算すれば良く，解の近くでは $\boldsymbol{d}$ を高い精度で計算した方が収束が速くなると期待できる．理想的な場合は L-M 法 (Newton 法) では厳密な $\boldsymbol{d}$ を用いると 2 次収束を示し，僅か数回の反復で解が得られるが，実問題では L-M 法は 2 次収束しない (実験を参照)．その理由としては，前述のように $\boldsymbol{H}$ が真のヘッセ行列ではないことや，条件数の大きな連立方程式では反復法が収束しても厳密解との誤差が大きいこと等が考えられる．

以下では 2 種類の解法を検討する．4.1 節の一括解法は，簡単な前処理を用いるため，PCG 法の収束が遅く途中で打ち切ることになる．4.2 節の縮約解法は，比較的速く収束する．図 4 に各手法の反復回数と残差の例を示す．一括解法 (mtd1) よりも縮約解法 (mtd2) の方が収束が速いことが分かる．L-M 法全体の収束性については 6 章の実験で比較する．

4.1 一括解法

全パラメータに対して一括で前処理付き共役勾配法を用いる．このためには，適当な前処理行列が必要である．ここでは，$\boldsymbol{p}$ の前半に反射率と形状を交互に配置し，後半に共通パラメータ・光源位置を配置する．即ち，$\boldsymbol{p}=(\boldsymbol{w}_0^T\ \lambda_0\ \boldsymbol{w}_1^T\ \lambda_1\cdots\rho\ \boldsymbol{s}^T\ \boldsymbol{l}_0^T\cdots)^T$．これを 3 つの部分に分けて考える．$p$ に依存する部分，全体にかかる部分，f に依存する部分である．この時，$\boldsymbol{H}+\mu\boldsymbol{I}$ は 3×3 の部分行列に分けて考えることができ，このうち対角ブロックのみを前処理行列に用いる．

$$\boldsymbol{H}+\mu\boldsymbol{I}\approx\begin{bmatrix}\boldsymbol{H}_{PP}&&\\&\boldsymbol{H}_{MM}&\\&&\boldsymbol{H}_{LL}\end{bmatrix}\tag{23}$$

このうち $\boldsymbol{H}_{MM}$ と $\boldsymbol{H}_{LL}$ の逆行列は容易に得られる．残る $\boldsymbol{H}_{PP}$ は本来はブロック対角ではないが， $\boldsymbol{H}_{PP}$ の逆行列が容易に得られるように，主対角線上の P 個の 5×5 ブロック部分のみを残し，残りを 0 にして近似する．この方法は，$\boldsymbol{H}_{PP}$ の近似精度が低いため，方程式の残差が十分に小さくなるまでに多大な反復回数を要する．

4.2 縮約解法

ここでは，まず反射率を従属変数として消去し未知数の数を約 1/5 に減らす．この縮約の後に反復法を用いる．これは，bundle-adjustment[7] で Schur 補行列を用いて特徴点座標を消去しカメラパラメータのみを残す方法に似ているが，本問題では反射率を消しても形状パラメータが残るので，縮約後の方程式でも未知パラメータは $O(10^4)$ 個あるので，やはり反復法を要する．

ここでは，$\boldsymbol{H}+\mu\boldsymbol{I}$ が図 3 や表 3 と同じ配置とするが，次の様に 4 つの部分に分ける．

$$\boldsymbol{H}+\mu\boldsymbol{I}=\begin{bmatrix}\boldsymbol{A}&\boldsymbol{B}\\\boldsymbol{B}^T&\boldsymbol{D}\end{bmatrix}\tag{24}$$

$\boldsymbol{A}$(図 3 の $\boldsymbol{H}_{WW}$) は 4×4 のブロックが対角に並ぶ行列であるから，容易に逆可能である．また，$\boldsymbol{d}$ と $\boldsymbol{g}$ を u 部分 (upper) と l 部分 (lower) に分割して，Gauss 消去を途中まで行うと，

$$\begin{bmatrix}\boldsymbol{d}_u\\\boldsymbol{d}_l\end{bmatrix}=\begin{bmatrix}\boldsymbol{A}&\boldsymbol{B}\\\boldsymbol{B}^T&\boldsymbol{D}\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{g}_u\\\boldsymbol{g}_l\end{bmatrix}=\begin{bmatrix}\boldsymbol{A}&\boldsymbol{B}\\\boldsymbol{0}&\boldsymbol{\Delta}\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{g}_u\\\boldsymbol{g}_l-\boldsymbol{B}^T\boldsymbol{A}^{-1}\boldsymbol{g}_u\end{bmatrix}\tag{25}$$

ただし，$\boldsymbol{\Delta}=\boldsymbol{D}-\boldsymbol{B}^T\boldsymbol{A}^{-1}\boldsymbol{B}$ である．この式に基づき，以下の手順で計算する: $\boldsymbol{g}'_u=\boldsymbol{A}^{-1}\boldsymbol{g}_u$ を得る．$\boldsymbol{g}'_l=\boldsymbol{g}_l-\boldsymbol{B}^T\boldsymbol{g}'_u$ を得る．$\boldsymbol{d}_l=\boldsymbol{\Delta}^{-1}\boldsymbol{g}'_l$ を反復法で近似する．$\boldsymbol{d}_u=\boldsymbol{g}'_u-\boldsymbol{A}^{-1}(\boldsymbol{B}\boldsymbol{d}_l)$ を得る．この第 3 ステップの反復法について $\boldsymbol{\Delta}$ を 4 つの部分に分けて考える．

$$\begin{bmatrix}\boldsymbol{d}_2\\\boldsymbol{d}_3\end{bmatrix}=\begin{bmatrix}\boldsymbol{S}&\boldsymbol{Y}\\\boldsymbol{Y}^T&\boldsymbol{Z}\end{bmatrix}^{-1}\begin{bmatrix}\boldsymbol{g}_2\\\boldsymbol{g}_3\end{bmatrix},\quad\text{ただし，}\boldsymbol{\Delta}=\begin{bmatrix}\boldsymbol{S}&\boldsymbol{Y}\\\boldsymbol{Y}^T&\boldsymbol{Z}\end{bmatrix}\tag{26}$$

この $\boldsymbol{Y}$, $\boldsymbol{Z}$ は密である．$\boldsymbol{Z}$ は $(3F+4)$ 次の正方行列であり，具体的には 100×100 程度なので $\boldsymbol{Z}^{-1}$ の計算は容易である． $\boldsymbol{S}$ の非零要素の配置は $\boldsymbol{H}_{SS}$ と同じで，1 行あたり 13 要素で P 行の大規模疎行列である．$\boldsymbol{S}$ を係数行列とする方程式は，市街地距離が 2 以下の画素との近接作用を表すため Poisson 方程式に似た構造であり，このような方程式には SSOR 前処理 PCG 法が有効である．但し，本問題では $\boldsymbol{Y}$, $\boldsymbol{Z}$ の部分が付加されているので，SSOR 前処理を拡張する．即ち，2×2 ブロックに対する SSOR 前処理の中で $P\times P$ 行列に対する SSOR 前処理を行う入れ子構造となる．

式 (26) のようにブロック分割した $\boldsymbol{d}_l=\boldsymbol{\Delta}^{-1}\boldsymbol{g}'_l$ の計算に，SOR 法を形式的に適用する

表 6 12 近傍

		2		
	2	1	2	
2	1	0	1	2
	2	1	2	
		2		

数値は中心画素からの市街地距離

表 7 画素のグループ分割

0	1	2	3	4	0	1	2	3	···
3	4	0	1	2	3	4	0	1	···
1	2	3	4	0	1	2	3	4	···
4	0	1	2	3	4	0	1	2	···
2	3	4	0	1	2	3	4	0	···
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	

と， $\boldsymbol{d}_2 \leftarrow \boldsymbol{d}_2 + \omega \boldsymbol{S}^{-1}(\boldsymbol{g}_2 - \boldsymbol{S}\boldsymbol{d}_2 - \boldsymbol{Y}\boldsymbol{d}_3)$ と $\boldsymbol{d}_3 \leftarrow \boldsymbol{d}_3 + \omega \boldsymbol{Z}^{-1}(\boldsymbol{g}_3 - \boldsymbol{Y}^T\boldsymbol{d}_2 - \boldsymbol{Z}\boldsymbol{d}_3)$ の 2 つの更新式が得られる．この $\boldsymbol{d}_2$ の更新に含まれる $\boldsymbol{S}^{-1}$ の乗算操作を SOR 法で近似する．そして，初期値を $\boldsymbol{d}_2 = \boldsymbol{d}_3 = \boldsymbol{0}$ とし，$\boldsymbol{d}_3$ の更新, $\boldsymbol{d}_2$ の順方向 SOR, $\boldsymbol{d}_2$ の逆方向 SOR, $\boldsymbol{d}_3$ の更新を行う．なお，$\boldsymbol{d}_2$ と $\boldsymbol{d}_3$ の更新順序を逆にすると速度が落ちることを実験で確認している．

具体的な計算法を以下に示す: $\boldsymbol{d}_3 = \omega \boldsymbol{Z}^{-1}\boldsymbol{g}_3$ を得る．次に $\boldsymbol{g}_2' = \boldsymbol{g}_2 - \boldsymbol{Y}\boldsymbol{d}_3$ を計算し，$\boldsymbol{d}_2 = \boldsymbol{S}^{-1}\boldsymbol{g}_2'$ を SSOR 法で近似する (順方向 SOR の後で逆方向 SOR を行う)．最後に $\boldsymbol{d}_3 \leftarrow \boldsymbol{d}_3(1-\omega) + \omega \boldsymbol{Z}^{-1}(\boldsymbol{g}_3 - \boldsymbol{Y}^T\boldsymbol{d}_2)$ を計算する．実験では $\omega = 1.7$ とした．

- 並列性

SOR 法において $i = 1$ から N まで順に更新する方法は回帰演算であり並列計算には適していない．一方，SOR 法の性能は更新の順序に依存しないので，4 近傍との相互作用を考える方程式 (例えば，法線から形状を計算する Poisson 方程式等) では未知変数を市松模様の 2 つのグループに分けて交互に更新することで並列計算する赤黒法[10] が知られている．本問題では表 6 の 12 近傍を考える必要があるので，表 7 のように画素 (x, y) の所属グループを $(x + 3y) \bmod 5$ として 5 グループに分割する．これにより 12 近傍内に自身と同じグループに属する画素は存在しないため，各グループに属する全画素の更新は並列計算可能である．この分割に従い，グループ 0,1,2,3,4,4,3,2,1,0 の順で更新することにより， $\boldsymbol{d}_2 = \boldsymbol{S}^{-1}\boldsymbol{g}_2'$ の SSOR 近似を行う．GPU での実装では，この分割を用いることができる．

4.3 PCG 法の実装の課題

GPU の場合，PCG 法全体を GPU で計算する必要がある．一部の行列演算のみを GPU で行い CPU との通信を頻繁に行う実装では性能が得られない．実験では，$\boldsymbol{H}$ の部分行列に使われる様々な圧縮方式に対応させるため十数種類の行列積等の GPU カーネル関数を用意し，次々と呼び出すことで PCG 法の計算を行っている．反復 1 回毎に打ち切り判定のために GPU の計算した実数を 1 つ参照するが，それ以外の計算は GPU 内で閉じている．表 1 の結果では，一括解法は 300msec，縮約解法は 200msec となり，CPU での一括解法と比べて 5–7 倍の速度が得られた．なお，縮約解法のうち 50msec は CPU で行っている $\boldsymbol{\Delta}$ の算出等の変数消去の計算である．この部分も GPU で行うことで更に計算時間を短縮できると考えられる．

一方，CPU での PCG 法の並列化は困難と考えられる．表 1 の結果では，PCG 法の処理時間が一括解法で約 1.5 秒，縮約解法で約 0.83 秒である．この時間の中で数百から数千回のスレッド起動や同期を行うのはオーバーヘッドが大きすぎる．また，一括解法ではキャッシュメモリに入りきらない $\boldsymbol{H}$ を反復毎に主記憶から読み込む必要があり，この転送の総量が延べ 10GB を越える．よって，計算時間の大半は演算よりも $\boldsymbol{H}$ の読み込みに費やされており，並列実行コアを増やしてもデータ供給が追いつかず速度向上の余地は大きくない．

表 2 の数値に基づくと，現在の CPU や GPU では読み込んだデータがキャッシュやレジスタに存在する間に数回から数十回の演算を行うアルゴリズムでなければ，演算性能を活かすことができない．しかし，本問題で用いる PCG 法や前処理の行列-ベクトル積では，1 要素の読み込みに対して 1 回か 1/2 回の積和演算しか行えない場合が多い．即ち，本手法は演算回数の少ない手法であるが，メモリ転送量が多過ぎるため演算性能の数%しか使えない．よって，更なる性能向上の余地は大きいと考えられる．

本問題で GPU により CPU の数倍の計算速度が得られているのは，GPU のメモリ帯域幅によるところが大きい．このため，行列要素を単精度にしてデータ転送量を削減する実験を 6 章で述べる．

4.4 パラメータの圧縮

$20,000,000$ 画素のカメラで撮影した画像を対象とするなら，本稿で仮定した $P = 20000$ に対する解法を直接適用できるかどうか疑わしい．そこで，パラメータベクトル $\boldsymbol{p}$ を基底ベクトルの線型結合 $\sum_i \boldsymbol{b}_i$ で近似することが考えられる．例えば，形状をスプライン基底の線型結合[5] で表したり，画像圧縮のように小さなブロックの DCT 基底で表す等である．形状をピクセル単位で表現するよりも少ないパラメータ数で表現できる可能性があり，高解像度画像を用いるにはこのような表現法を要すると考えられる．

形状基底を列ベクトルとする縦長の行列 $\boldsymbol{Q}$ を用い，前節の縮約後の方程式 $\Delta \boldsymbol{d}_l = \boldsymbol{g}_l'$ の解 $\boldsymbol{d}_l$ を $\boldsymbol{Q}\hat{\boldsymbol{d}}_l$ で近似する．結合重み $\hat{\boldsymbol{d}}_l$ を得るには $(\boldsymbol{Q}^T\Delta\boldsymbol{Q})\hat{\boldsymbol{d}}_l = \boldsymbol{Q}^T\boldsymbol{g}_l'$ を解けば良い．現時点の解像度ではピクセル単位の表現を用いて数分で解けるので，この方法の実装と評価は今後の課題である．

Vol.2012-CVIM-181 No.19
2012/3/16

5. 直線探索の並列化

直線探索の計算の大半は，E を 2 次関数で近似するために 3 点で E を評価することである．選んだ 3 点が望ましくない 3 点であった場合は 4 点目や 5 点目の評価を行うこともあるが，いずれにしろ E の計算を並列化すれば良い．表 1 の結果によると，シングルスレッドで 50msec の E の計算が 6 コアで 10msec となっている．これを GPU で行うことは容易であるが最大で 10msec の削減であるから優先順位の高い課題ではない．

6. 実　　験

図 5 は，図 1 の実画像に対する推定の処理時間 (横軸，単位は msec) と残差 (縦軸) を示した図である．一括解法・縮約解法の比較，及び GPU の有無で比較を行った．図中 g1,g2,c1,c2 は GPU の有無を表す記号と手法を表す数字から成る．g は GPU を用いた場合，c は用いない場合であり，1 は一括解法，2 は縮約解法である．CPU のみで計算した場合は，c2 の縮約解法により c1 の一括解法と比べて約 1.7 倍の高速化が達成されていることが分かる．GPU を利用した g2,g1 の場合 c1 と比べて 3–4 倍程度の高速化が達成されているが，2 つの解法の性能差は殆どない．これは，縮約解法の並列化粒度が小さく $P = 20000$ の規模では充分に効果が得られない等の理由が考えられる．なお，PCG 法の反復回数は 128 回とした．また，縮約解法の方が $(\boldsymbol{H} + \mu\boldsymbol{I})^{-1}\boldsymbol{g}$ を精度良く計算しているにも関わらず，L-M 法の同一反復回数での残差を見ると一括解法の方が若干小さくなっている．なお，この画像列で達成された既知の最小の残差は約 2.86 であり，図の下端よりも下に位置する．ここにたどり着くには更に多くの反復を要する．

図 6 は，3.4 節の目的関数を用いた同様の比較である．g1,g2,c1,c2 の記号の意味は図 5 と同じで，それらの傾向も殆ど同じである．縦軸の残差の定義が図 5 と異なるため単純な比較はできないが，この目的関数を使うと初期の収束は速いが，後半では速度の低下が見られる．

図 7 の g1,g2 は図 5 と同じであり，ここでは単精度での実装 f1,f2 と比較する．一括解法 (f1) は GPU で計算する行列・ベクトルを単精度にして行列のデータ量を半分することにより約 1.6 倍の速度となっている．一方，縮約解法 (f2) ではデータ転送以外に掛かる時間の割合が大きいため，今回の実装ではあまり効果が見られない．単精度による精度の低下はこの画像列では見られず倍精度と同等の残差を達成できている．但し，100 例以上の実画像列による実験により，この方法は時々不安定な挙動をすることが判っているため，今後の検討を要する．

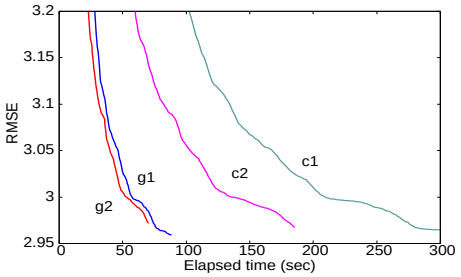

図 5　計算時間と残差 (FR 法)

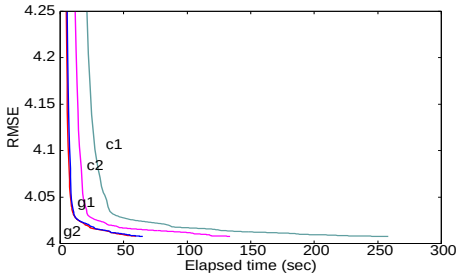

図 6　計算時間と残差 (SR 法)

なお，GPU の実装は前世代の GPU でも動作するように書かれており，今回の実験に用いた GTX580 に最適化されているわけではない．この世代固有の機能 (3 倍に増やされた共有メモリ，書き込み可能な L2 キャッシュ，異種カーネルの同時実行，他) を使うことでの性能向上の余地もあると考えられる．

図 8 は，シミュレーション画像列に対する実験であり，残差が 0 に近づく時の挙動を調べる．入力にノイズは加えていないが，入力画像が単精度実数であるため 10^{-6} 程度の丸め誤差があり，残差 0 にはならない．対象物体は球で，有効点数は 64516，画像数は 25 である．グラフ中の g1,g2,S1,S2 のアルファベットは目的関数 (g:本来の目的関数，S:3.4 節の目的関数)，数字はこれまでと同じく一括解法 (1) と縮約解法 (2) を示す． 0 に近い残差を達成するために，L-M 法の途中から μ を 10^{-6} にし PCG 法の反復回数を 512 に増やした．これにより縮約解法 (g2,S2) は途中から急速に収束している．この実験では S2 が途中で停滞するため g2 の方が速く収束している．一括解法 (g1,S1) の場合，グラフ右端以降はこの付近と同様の傾きで残差が減少するので残差が 10^{-6} 程度になるにはかなりの時間を要する．また，途中から S1 の方が速くなっているのが分かる．図では 150sec までしか示していないが，いずれの手法も推定の後半ではグラフはほぼ直線を描く．これは，1 次収束であることを示しており，2 次収束は達成されていない．

他の実画像列において，図 9 では一貫して縮約解法 (2) の方が残差が小さいが，図 10 のように縮約解法 (2) の方が先に減速する場合も見られた．復元全体の速度は μ や PCG 反復回数の段階的変更等により変化するため，これを自動調節する方法の検討を要する．

7. ま　と　め

本稿では，未校正照度差ステレオ法による形状・反射特性・光源の推定を実現する最適化

Vol.2012-CVIM-181 No.19
2012/3/16

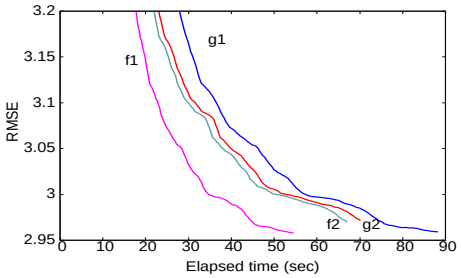

図 7 計算時間と残差 (単精度と倍精度)

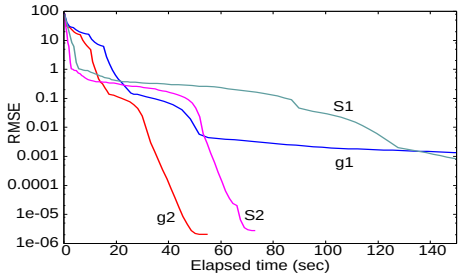

図 8 計算時間と残差 (球のシミュレーション)

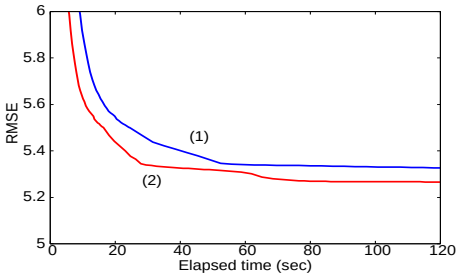

図 9 計算時間と残差 (実物体 1)

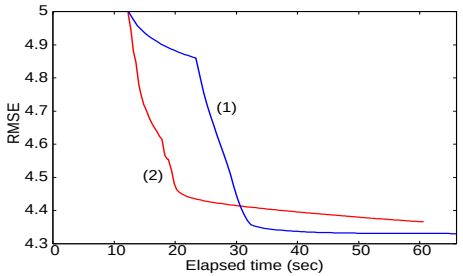

図 10 計算時間と残差 (実物体 2)

アルゴリズムの詳細を述べた．複数の解法や実装法を実画像で比較し，今回最良の手法は単純な手法と比べて最大で数倍の性能が得られることを確認した．また問題点や今後の課題についても述べた．

本研究の一部は文部科学省科学研究費若手研究 (B 22700181) の助成によった．

参 考 文 献

1) T. Migita and S. Ogino and T. Shakunaga: Direct Bundle Estimation for Recovery of Shape, Reflectance Property and Light Position, *Computer Vision – ECCV2008*, Springer, LNCS 5304, pp. 412–425, (2008).
2) 荻野真佑, 右田剛史, 尺長健: Torrance-Sparrow モデルに基づく反射特性・形状・光源位置の同時推定の検討, 情報処理学会研究報告, CVIM-162-87, (2008).
3) T. Migita, K. Sogawa, T. Shakunaga: Specular-free Residual Minimization for Photometric Stereo with Unknown Light Sources, *in Proc. PSIVT2011 PART I (LNCS 7087)*, pp. 178–189, (2011).
4) 祖川和弘，右田剛史，尺長健: 光源色の補空間を用いた物体形状・反射特性・光源の同時推定，情報処理学会研究報告, CVIM-176-23, (2011).
5) 渡辺隼，右田剛史，尺長健: スプライン補間を用いた動画像からのステレオ形状復元, 情報処理学会研究報告, CVIM-180-50, (2012).
6) A.L. Yuille and D. Snow and R. Epstein and P.N. Belhumeur, Determining Generative Models of Objects Under Varying Illumination: Shape and Albedo from Multiple mages Using SVD and Integrability, *Int'l J. of Computer Vision 35(3)*, pp. 203–222, (1999).
7) B. Triggs and P. McLauchlan and R. Hartley and A. Fitzgibbon, Bundle Adjustment — A Modern Synthesis, *Vision Algorithms '99 (LNCS 1883)*, pp. 298–372, (2000).
8) D. W. Marquardt: An Algorithm for Least-squares Estimation of Nonlinear Parameters, *SIAM J. Appl. Math.*, vol. 11, pp. 431–441, (1963).
9) G. ストラング著，山口昌哉監訳，井上昭訳：線形代数とその応用，産業図書 (1978).
10) 金田 康正 編著：並列数値処理—高速化と性能向上のために—，コロナ社 (2010).

付　　録

A.1 線型方程式解法

式 $\boldsymbol{A}^{-1}\boldsymbol{b}$ は，線型連立方程式 $\boldsymbol{A}\boldsymbol{x}=\boldsymbol{b}$ の解 $\boldsymbol{x}$ を計算することを表す． $\boldsymbol{A}$ が大規模疎行列の場合は，反復法で $\boldsymbol{x}$ の近似値を計算する．なお $\boldsymbol{A}$ は対称行列とする．
Poisson 方程式等に対しては，SOR(Successive Over Relaxation) 法[9] が簡便で比較的高速な方法として知られている．SOR 法とは $x_i \leftarrow x_i - \omega(\sum_j a_{ij}x_j - b_i)/a_{ii}$ の更新操作を $i=1,...,N$ の順に逐次行い (他の順序でもよい)，その操作を収束するまで繰り返すものである．なお，ω は適当な加速定数，a_{ij}, x_j, b_i は $\boldsymbol{A}$, $\boldsymbol{x}$, $\boldsymbol{b}$ の要素である．
最も効率的な連立方程式解法は前処理付き共役勾配法 (以下 PCG 法) と考えられる．この手法は次の様に書くことができる．

$$\boldsymbol{x} \leftarrow \boldsymbol{x} - \alpha\boldsymbol{d} \text{ where } \boldsymbol{d}^T(\boldsymbol{A}(\boldsymbol{x}-\alpha\boldsymbol{d})-\boldsymbol{b}) = 0, \tag{27}$$

$$\boldsymbol{d} \leftarrow \boldsymbol{C}(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}) - \beta\boldsymbol{d} \text{ where } \boldsymbol{d}^T\boldsymbol{A}(\boldsymbol{C}(\boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}) - \beta\boldsymbol{d}) = 0 \tag{28}$$

$\boldsymbol{C}$ は前処理行列であり，対称行列で，積 $\boldsymbol{C}\boldsymbol{A}$ が単位行列に近く，積 $\boldsymbol{C}\boldsymbol{x}$ が容易に計算できることが要求される．有力な $\boldsymbol{C}$ の構成法として．前述の SOR 法を前処理に応用する SSOR(Symmetric SOR) 前処理が知られている．$\boldsymbol{d}$ の計算に現れる $\boldsymbol{C}\boldsymbol{b}'$(ただし $\boldsymbol{b}' = \boldsymbol{A}\boldsymbol{x}-\boldsymbol{b}$) の計算において，$\boldsymbol{C} \approx \boldsymbol{A}^{-1}$ なので，方程式 $\boldsymbol{A}\boldsymbol{z}-\boldsymbol{b}'=0$ の解 $\boldsymbol{z}$ を SOR 法の少ない反復で近似することが考えられる (通常，反復回数は 1 とする)．ただし，$\boldsymbol{C}$ は対称行列である必要があるため，$i=1,...,N$ の順に SOR 法を適用した後に $i=N,...,1$ の順に適用することで対称化する．この操作の結果得られる $\boldsymbol{z}$ の近似値を $\boldsymbol{C}\boldsymbol{b}'$ として用いる．$\boldsymbol{C}$ を閉じた形式で書き表すことは可能だが，実際の計算において行列 $\boldsymbol{C}$ を陽に計算することはない．