Vol.2015-CVIM-197 No.2
2015/5/18

RGBD画像を用いた人物姿勢追跡の高速化と人体モデルパラメータ推定への適用

長谷 康平[1,a)]　溝部 諒[1]　右田 剛史[1]　尺長 健[1]

概要：我々は，RGBD 画像を用いた人物姿勢追跡において，パーティクルフィルタと Levenberg-Marquardt(LM) 法を併用することで精度の高い追跡を実現しているが，入力インタフェースとして用いるには課題がある．本稿では，まず追跡系の処理速度を向上させるためにパーティクルフィルタと LM 法の GPU 化を行い，実時間処理が可能であることを示す．また，体格の異なる人物に対応するため，姿勢追跡と人体モデルのパラメータ推定を LM 法によって同時に行うことを検討する．

1. はじめに

物体追跡は従来より広く研究が行われており，人体や手指の追跡のような高次元の姿勢追跡に関する研究 [1] も行われている．近年では，Kinect などの高性能な RGBD センサを利用することで，背景と人物の分離処理が容易となり，精度の高い姿勢追跡を行うことができる．加藤ら [2] は人物の上半身をパーティクルフィルタを用いて追跡し，さらに追跡によって得られた姿勢情報を用いてジェスチャ認識を行う手法を提案している．また，筒井ら [3] はパーティクルフィルタと Levenberg-Marquardt 法 [4](以下，LM 法) を併用することで，追跡精度を向上させる手法を提案している．

人物姿勢追跡を入力インタフェースに利用することを考えた場合，実時間追跡を行うことは不可欠である．しかし，従来の追跡では実時間追跡を行えなかった．そこで，本稿では実時間追跡のために，パーティクルフィルタ，及び LM 法の処理を GPU で行うことにより，処理時間の短縮を目指す．また，追跡には人物の 3D モデルを用いるが，人物固有のパラメータである両肩の位置や腕の長さなどを与える必要がある．この人物固有のパラメータの調整を手動で行うには手間がかかるため，自動化は大きな課題であった．一方，LM 法を用いた姿勢パラメータの最適化は，すべてのパラメータを同時に最適化するため，両肩の位置などのパラメータを追加するのが容易である．そこで本稿では，LM 法を用いた姿勢パラメータの推定に両肩の位置を表すパラメータを新たに追加し，同時に推定を行う手法を提案する．以下，2 章では人体関節モデルとその姿勢を表す行列について述べる．3 章ではパーティクルフィルタと LM 法を用いた追跡，及びモデルのパラメータ推定について述べる．4 章ではこれらの追跡手法に対する GPU を用いた高速化について述べる．5 章ではシミュレーション実験を行い，CPU と GPU の処理時間の比較結果を述べる．また，モデルパラメータの推定について，シミュレーションと実動画で実験を行った結果を述べる．6 章では本稿で行ったことをまとめ，今後の課題を述べる．

[1] 岡山大学
Okayama University
[a)] {hase, mizobe, migita, shaku} @chino.cs.okayama-u.ac.jp

2. 人体関節モデル

本稿で追跡対象とするのは，人体の上半身である．上半身は，胴体 (6 自由度)，右上腕，左上腕，右下腕，左下腕 (各 2 自由度) の 5 つの部位が，右肩，左肩，右肘，左肘の 4 つの関節によって連結されているものとする．したがって，人体関節モデルの姿勢 $\mathbf{p}$ は以下のベクトルで表すことができる．

$$\begin{aligned}\mathbf{p} &= \left[\, p_1 \;\; p_2 \;\; p_3 \;\; \cdots \;\; p_{14} \,\right]^\top \\ &= \left[\, \mathbf{p}_O^\top \;\; \mathbf{p}_{RU}^\top \;\; \mathbf{p}_{LU}^\top \;\; \mathbf{p}_{RL}^\top \;\; \mathbf{p}_{LL}^\top \,\right]^\top\end{aligned} \tag{1}$$

ここで，$\mathbf{p}_O = [p_1 \cdots p_6]^\top$，$\mathbf{p}_{RU} = [p_7 \;\; p_8]^\top$，$\mathbf{p}_{LU} = [p_9 \;\; p_{10}]^\top$，$\mathbf{p}_{RL} = [p_{11} \;\; p_{12}]^\top$，$\mathbf{p}_{LL} = [p_{13} \;\; p_{14}]^\top$ はそれぞれ，胴体，右上腕，左上腕，右下腕，左下腕の姿勢パラメータを表す．

2.1 座標系の定義

一般に，関節物体の姿勢は，主となる物体の姿勢とそれぞれの関節位置，関節角度により定義される．本稿では上半身の関節モデルの姿勢推定問題を取り扱うため，3 つの (同次) 座標系を用いる．ここで，上腕と下腕については左

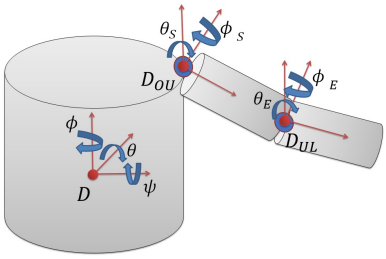

図 1 各座標系の関係図

右別々の座標系を用いる必要があるが，左右の腕は独立に考えることができるため，以下では 3 つの座標系について述べる．

(1) $\widetilde{\mathbf{M}}_O$：胴体座標系で表される座標
(2) $\widetilde{\mathbf{M}}_U$：上腕座標系で表される座標
(3) $\widetilde{\mathbf{M}}_L$：下腕座標系で表される座標

上腕座標系の点 $\widetilde{\mathbf{M}}_U$ と下腕座標系の点 $\widetilde{\mathbf{M}}_L$ は以下の式で胴体座標系の点に変換できる．

$$\widetilde{\mathbf{M}}_O = \mathbf{D}_{OU}\widetilde{\mathbf{M}}_U \tag{2}$$
$$\widetilde{\mathbf{M}}_O = \mathbf{D}_{OU}\mathbf{D}_{UL}\widetilde{\mathbf{M}}_L \tag{3}$$

$\mathbf{D}_{OU}$， $\mathbf{D}_{UL}$ はそれぞれ上腕座標系から胴体座標系，下腕座標系から上腕座標系への座標変換行列である．それぞれの座標系の関係を図 1 に示す．詳細については次節 2.2 で述べる．

任意の姿勢に対して関節モデルの各点の画像座標を計算するためには，胴体座標系からカメラ座標系 $\widetilde{\mathbf{M}}_C = [X_C\ Y_C\ Z_C\ 1]^\top$ 及び，画像座標系 $\widetilde{\mathbf{M}}_I = [X_I\ Y_I\ 1]^\top$ に変換する必要がある．この変換は姿勢行列 $\mathbf{D}$ と内部パラメータ行列 $\mathbf{K}$ を用いて次式で表される．

$$\widetilde{\mathbf{M}}_C = \mathbf{D}\ \widetilde{\mathbf{M}}_O \tag{4}$$
$$\widetilde{\mathbf{M}}_I = \widehat{\mathbf{K}}\ \mathbf{D}\ \widetilde{\mathbf{M}}_O\ ,\ \text{ただし}\ \widehat{\mathbf{K}} = [\mathbf{K} \mid \mathbf{0}] \tag{5}$$

2.2 各剛体の表現と座標変換行列

胴体は，6 自由度（3 次元並進 +3 次元回転）の姿勢行列で取り扱う．胴体座標系からカメラ座標系への変換 $\mathbf{D}$ を胴体の姿勢行列と考えることができる．

$$\mathbf{D} = \begin{bmatrix} \mathbf{R}_z^\theta \mathbf{R}_y^\phi \mathbf{R}_x^\psi & \mathbf{t}_O \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{6}$$

ここで式 (6) における $\mathbf{t}_O$ は胴体の並進を表すベクトル，$\mathbf{R}_z^\theta \mathbf{R}_y^\phi \mathbf{R}_x^\psi$ は胴体の姿勢を表す回転行列であり，それぞれの項については以下の式で表される．

$$\mathbf{R}_x^\psi = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\psi & -\sin\psi \\ 0 & \sin\psi & \cos\psi \end{bmatrix} \tag{7}$$

$$\mathbf{R}_y^\phi = \begin{bmatrix} \cos\phi & 0 & \sin\phi \\ 0 & 1 & 0 \\ -\sin\phi & 0 & \cos\phi \end{bmatrix} \tag{8}$$

$$\mathbf{R}_z^\theta = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{9}$$

上腕の姿勢は胴体の姿勢 $\mathbf{D}$ と肩を中心とした回転の合成により得られる．胴体座標系における肩関節の位置を $\mathbf{t}_S$ とする．これは追跡する人物によって異なる既知パラメータである．上腕の主軸を x 軸とし，肩を中心に y 軸，z 軸回転を考える．これらにより，胴体座標系への座標変換行列 $\mathbf{D}_{OU}$ は以下の式 (10) で表される．

$$\mathbf{D}_{OU} = \begin{bmatrix} \mathbf{R}_z^{\theta_S}\mathbf{R}_y^{\phi_S} & \mathbf{t}_S \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{10}$$

下腕の姿勢は上腕から胴体への座標変換行列 $\mathbf{D}_{OU}$ と，肘を中心とした回転の合成により得られる．上腕座標系における肘の位置を $\mathbf{t}_E$ とする．これは追跡する人物によって異なる既知パラメータである．下腕座標系の原点は肘関節にあり，下腕の主軸を x 軸とし，肘を中心に y 軸，z 軸回転を考える．これらにより，上腕座標系への座標変換行列 $\mathbf{D}_{UL}$ は以下の式 (11) で表される．

$$\mathbf{D}_{UL} = \begin{bmatrix} \mathbf{R}_z^{\theta_E}\mathbf{R}_y^{\phi_E} & \mathbf{t}_E \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{11}$$

2.3 3 次元疎テンプレート

3 次元疎テンプレートによる姿勢追跡 [5] では，現時刻 t の姿勢を推定する時，まず，前時刻 $t-1$ の姿勢において各部位に対応したテクスチャ付き関節モデルを投影する．このとき隠面処理を行うことにより，モデルの表側の画素値と距離値を取得することができる．局所的に輝度値が最大・最小となる点を選び，各点の 3 次元座標 $(X_i\ Y_i\ Z_i)$ と画素値 T_i の組の集合を作成する．これを 3 次元疎テンプレートと呼び，$\{(\widetilde{\mathbf{M}}_i; T_i)\} = \{(X_i\ Y_i\ Z_i; T_i)\}$ で表す．同様に隠面処理に用いた Z バッファにおいて，局所的に距離値が最大・最小となる点を選び，$\{(\widetilde{\mathbf{M}}'_i)\} = \{(X'_i\ Y'_i; Z'_i)\}$ を作成する．

3. 人物姿勢追跡

本稿では，RGB 画像と距離画像を併用して人物姿勢追跡を行う．距離画像を用いることで，背景と人物の分離処理が容易となり，高精度な追跡が期待できる．なお，本稿では RGB 画像と距離画像という名称を用いるが，実際には RGB 画像をグレースケールに変換して計算を行っている．

3.1 パーティクルフィルタ

パーティクルフィルタ [6] は，時刻 $t-1$ における少数の

候補姿勢 (例えば 10 個) からそれぞれに対し乱数ベクトルを加算することで時刻 t における多数のサンプル姿勢を生成し，評価値の小さいサンプル姿勢を選択する．ここで，候補姿勢を $\mathbf{p}_c$，乱数ベクトルを $\delta\mathbf{p}$ とすると，サンプル姿勢は $\mathbf{p}_c + \delta\mathbf{p}$ で表される．なお，$\mathbf{p}_c$ は 10 個の候補姿勢から 1 つが選ばれる．$\delta\mathbf{p}$ は次の性質を持つ正規乱数である．

$$\begin{cases} \mathcal{E}[\delta\mathbf{p}\delta\mathbf{p}^\top] = \mathrm{diag}(\sigma_1^2,\ \cdots,\ \sigma_{14}^2) \\ \mathcal{E}[\delta\mathbf{p}] = \mathbf{0} \end{cases} \tag{12}$$

ただし，$\mathcal{E}[\cdot]$ は期待値であり，σ_n は各パラメータに対応する標準偏差である．

一般に，パーティクルフィルタによる追跡において追跡性能を改善するためには極めて多くのサンプルが必要である．そのため，本稿では粗密探索法 [7] を用いることで，サンプル数の削減を図る．この手法では，σ_n を半減させながらパーティクルフィルタを数回繰り返すことで，効率よく姿勢空間を探索する．

3.2 カスケード姿勢推定

人物上半身の姿勢推定では，姿勢空間の次元が大きいため，高精度な追跡を行うためには膨大な量のサンプルが必要である．それを避けるために胴体 (6 自由度)，右上腕・下腕 (2 + 2 自由度)，左上腕・下腕 (2 + 2 自由度) の順に姿勢の推定を行うカスケード姿勢推定 [2] を用いる．この手法では，まず胴体の姿勢を推定し，次に推定した胴体の姿勢を固定して右腕，左腕の姿勢の推定を行う．このとき，k 番目のサンプル姿勢 $\mathcal{D}_k$ は胴体，左右上腕，左右下腕の順で以下の式 (13) により計算される．

$$\mathcal{D}_k = \begin{cases} (\delta\mathbf{D}_k \circ \hat{\mathbf{D}}_{CO}) \\ (\delta\mathbf{D}_k \circ \hat{\mathbf{D}}_{CO})(\delta\mathbf{D}'_k \circ \hat{\mathbf{D}}_{OU}) \\ (\delta\mathbf{D}_k \circ \hat{\mathbf{D}}_{CO})(\delta\mathbf{D}'_k \circ \hat{\mathbf{D}}_{OU})(\delta\mathbf{D}''_k \circ \hat{\mathbf{D}}_{UL}) \end{cases} \tag{13}$$

ここで，

$$\delta\mathbf{D}_k \circ \hat{\mathbf{D}}_{CO} = \begin{bmatrix} \mathbf{R}_x^{p_1^+}\mathbf{R}_y^{p_2^+}\mathbf{R}_z^{p_3^+} & \mathbf{t}_O + \delta\mathbf{t}_k \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{14}$$

$$\delta\mathbf{D}'_k \circ \hat{\mathbf{D}}_{OU} = \begin{bmatrix} \mathbf{R}_y^{p_j^+}\mathbf{R}_z^{p_{j+1}^+} & \mathbf{t}_S \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{15}$$

$$\delta\mathbf{D}''_k \circ \hat{\mathbf{D}}_{UL} = \begin{bmatrix} \mathbf{R}_y^{p_{j+2}^+}\mathbf{R}_z^{p_{j+3}^+} & \mathbf{t}_E \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{16}$$

ただし，$\mathbf{t}_O = [p_4\ p_5\ p_6]^\top$，$\delta\mathbf{t}_k = [\delta p_4\ \delta p_5\ \delta p_6]^\top$ であり，式 (15), (16) 中において, 右手系では $j = 7$, 左手系では $j = 11$ である．なお，$p_i^+ = p_i + \delta p_i (i = 1, 2, \cdots, 14)$ である．

RGB 画像では，テンプレートの i 番目の点 $\widetilde{\mathbf{M}}_i$ に対応する入力画素値を $\hat{I}(\widetilde{\mathbf{M}}_i; \mathcal{D}_k)$ と表し, 距離画像ではテンプレートの i 番目の点 $\widetilde{\mathbf{M}}'_i$ に対応する入力距離値を $\hat{Z}(\widetilde{\mathbf{M}}'_i; \mathcal{D}_k)$ と表す．

3.3 RGBD 画像を用いた追跡

入力 RGB 画像に対する k 番目のサンプル姿勢の評価値 ϵ_k^C は，次式で定義される．

$$\epsilon_k^C = \sum_{i=1}^{N} \rho_c(d_i),\ \ \text{ただし}\ \ \rho_c(x) = \frac{x^2}{c^2 + x^2} \tag{17}$$

$$d_i = \beta\hat{I}(\widetilde{\mathbf{M}}_i; \mathcal{D}_k) - \alpha T_i$$

$$\alpha = \frac{1}{\sum_{i=1}^{N} T_i},\quad \beta = \frac{\alpha\sum_{i=1}^{N} T_i^2}{\sum_{i=1}^{N} T_i\hat{I}(\widetilde{\mathbf{M}}_i; \mathcal{D}_k)}$$

なお，α はテンプレート画像の明度の正規化係数，β は入力画像の明度の正規化係数，N はテンプレート点数である．また ρ_c は外れ値に対応するための関数，c は定数であり, ここでは，$c = 0.5/N$ とする．

一方，入力距離画像に対する k 番目のサンプル姿勢の評価値 ϵ_k^D は, 次式で定義される．

$$\epsilon_k^D = \sum_{i=1}^{N'} \rho_{c'}(d'_i),\ \ \text{ただし}\ \ d'_i = \hat{Z}(\widetilde{\mathbf{M}}'_i; \mathcal{D}_k) - Z_i \tag{18}$$

なお，$c' = 5.0$ とする．

k 番目のサンプル姿勢に対する評価値 ϵ_k を式 (17)，(18) により計算した ϵ_k^C，ϵ_k^D の重み付き和によって (19) で定義する．

$$\epsilon_k = (1 - \lambda)\epsilon_k^C + \lambda\epsilon_k^D \tag{19}$$

ここで，λ は定数である．

3.4 Levenberg-Marquardt 法

本節では，Levenberg-Marquardt 法 (以下 LM 法) を用いた追跡について述べる [3]．パーティクルフィルタによる追跡では 14 次元の姿勢パラメータを同時に推定するのが困難であるため，カスケード姿勢推定による追跡を行っていた．しかし，LM 法による最適化を行うことによって，14 パラメータの同時推定を行うことができる．さらに，追跡対象のパラメータを増やす拡張が容易となる．

LM 法により式 (19) を最小化するために $\mathbf{r}(\mathbf{p})^\top\mathbf{r}(\mathbf{p})$ の形に変形する．

$$\mathbf{r}(\mathbf{p}) = \begin{bmatrix} \mathbf{r}_O^\top & \mathbf{r}_{RU}^\top & \mathbf{r}_{LU}^\top & \mathbf{r}_{RL}^\top & \mathbf{r}_{LL}^\top \end{bmatrix}^\top \tag{20}$$

ここで，

$$\mathbf{r}_* = \begin{bmatrix} r_1 & r_2 & \cdots & r_N & r'_1 & \cdots & r'_{N'} \end{bmatrix}^\top \tag{21}$$

であり，$*$ はそれぞれの部位を表している．ただし，$r_i = \sqrt{1-\lambda}d_i/\sqrt{c^2 + d_i^2}$，$r'_i = \sqrt{\lambda}d'_i/\sqrt{c'^2 + d'^2_i}$ である．式 (21) の前半は ϵ_k^C に対応し，後半が ϵ_k^D に対応する．

LM 法では，次式により候補姿勢 $\mathbf{p}^{(\tau)}$ を $\mathbf{p}^{(\tau+1)}$ に修正する．

$$\mathbf{p}^{(\tau+1)} = \mathbf{p}^{(\tau)} - (\mathbf{J}^\top\mathbf{J} + C\mathbf{I})^{-1}\mathbf{J}^\top\mathbf{r} \tag{22}$$

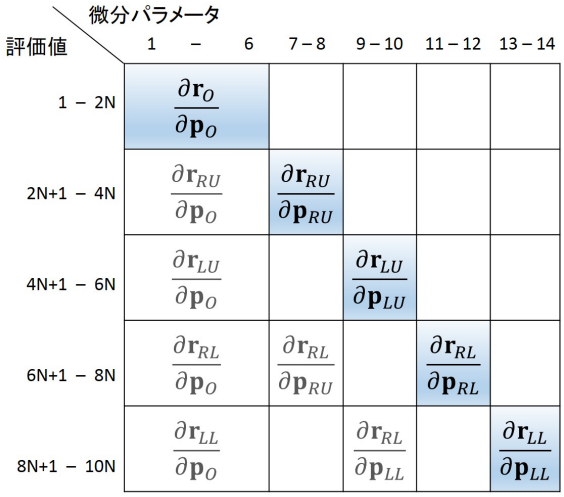

図 2 ヤコビ行列の近似

ここで，

$$\mathbf{J} = \left[\begin{array}{cccc} \frac{\partial \mathbf{r}}{\partial p_1} & \frac{\partial \mathbf{r}}{\partial p_2} & \frac{\partial \mathbf{r}}{\partial p_3} & \cdots & \frac{\partial \mathbf{r}}{\partial p_{14}} \end{array} \right] \tag{23}$$

であり，$\frac{\partial \mathbf{r}}{\partial p_n}$ については以下の式で近似を行う．

$$\frac{\partial \mathbf{r}}{\partial p_n} \approx \frac{\mathbf{r}(\mathbf{p} + \varepsilon \mathbf{e}_n) - \mathbf{r}(\mathbf{p} - \varepsilon \mathbf{e}_n)}{2\varepsilon} \tag{24}$$

ただし，$\mathbf{e}_n$ は 14 次元のベクトルで，n 番目の要素が 1，その他の要素が 0 のベクトルである．ε は各パラメータに応じた微小定数である．C はある正数（例えば 10^{-5}），$\mathbf{I}$ は単位行列である．候補姿勢 $\mathbf{p}^{(\tau)}$ が最適解から離れている場合は，C を大きくとることで発散しにくくなり，最適解に近づくにつれて C を小さくとることで高速に収束させることができる．なお，本稿ではヤコビ行列 $\mathbf{J}$ を図 2 に示す形に近似を行っている．図 2 において，色がついている要素だけを計算している．本来はそれ以外の要素も計算する必要があるが，この近似により計算量を削減することができ，1 回の反復にかかる時間を短縮することができる．図 3 にこの近似を用いて 10 個の候補姿勢を LM 法で最適化した場合の評価値の変化を示す．図 3 から近似ヤコビ行列で計算を行った場合においても評価値が減少することを確認できる．

3.5 パーティクルフィルタと LM 法の併用

人物姿勢追跡において，腕はフレーム間の動きが大きいため，LM 法のみの追跡では安定した追跡が困難である．そこで本稿では，パーティクルフィルタと LM 法を併用して姿勢の最適化を行う．パーティクルフィルタの複数の候補姿勢をそれぞれ最適化することで，安定した追跡を行うことが期待できる．

なお，本稿では浮動小数点数演算に関し，パーティクルフィルタの計算では float 型を用いている．一方，LM 法の計算では式 (24) で表される差分近似を計算するため，桁落ちを防ぐために double 型を用いる．また，LM 法では式 (19) で定義される評価値の微分を用いるため，LM 法の反復毎に評価値は滑らかに変化する必要がある．そこで，式 (17)，(18) を計算する際に，入力画素値 $\hat{I}(\widetilde{\mathbf{M}}_i; \mathcal{D}_k)$ と入力距離値 $\hat{Z}(\widetilde{\mathbf{M}}_i'; \mathcal{D}_k)$ に対し Bilinear 補間を行い，座標が実数の場合に画素値，距離値が滑らかに変化するようにする．

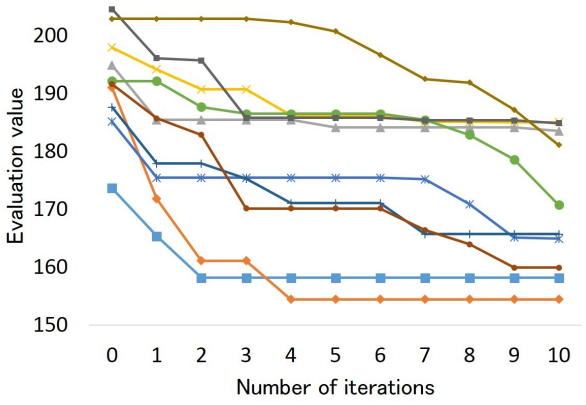

図 3 各候補姿勢の評価値の変化

3.6 モデルのパラメータ推定

従来の人物姿勢追跡では，人体モデルのパラメータ (肩，肘関節の位置) の調整を手動で行っていた．しかし，追跡対象の人物毎にモデルのパラメータは異なるので，パラメータ調整に労力がかかり，パラメータ調整の自動化は大きな課題となっていた．そこで，本節では LM 法を用いた追跡とパラメータの同時推定法を検討する．

3.6.1 パラメータ推定の方法

本稿では，式 (10) の $\mathbf{t}_S$ で表される肩関節のパラメータを推定する．$\mathbf{t}_S$ は並進 3 パラメータなので，式 (1) を式 (25) に拡張し，合計 20 パラメータを LM 法により同時に推定する．

$$\begin{aligned} \mathbf{p} &= \left[\begin{array}{cccccc} p_1 & p_2 & p_3 & \cdots & p_{20} \end{array} \right]^\top \\ &= \left[\begin{array}{ccccccc} \mathbf{p}_O^\top & \mathbf{p}_{RU}^\top & \mathbf{p}_{LU}^\top & \mathbf{p}_{RL}^\top & \mathbf{p}_{LL}^\top & \mathbf{t}_{RS}^\top & \mathbf{t}_{LS}^\top \end{array} \right]^\top \end{aligned} \tag{25}$$

ここで，$\mathbf{t}_{RS} = [p_{15} \ p_{16} \ p_{17}]^\top$ は右肩の位置，$\mathbf{t}_{LS} = [p_{18} \ p_{19} \ p_{20}]^\top$ は左肩の位置を表す．初期値として一般的な肩関節の位置を与え，パラメータを推定しながら動作の姿勢追跡を行う．1 フレームのみでの推定は難しいため，あるフレームまで関節の位置を追跡し，その関節位置を最終的な推定結果とする．

3.6.2 モデルパラメータの推定を含めた姿勢追跡

姿勢追跡と肩関節のパラメータ推定を同時に行うため，式 (23) を次式のように拡張する．

$$\mathbf{J} = \left[\begin{array}{cccc} \frac{\partial \mathbf{r}}{\partial p_1} & \frac{\partial \mathbf{r}}{\partial p_2} & \frac{\partial \mathbf{r}}{\partial p_3} & \cdots & \frac{\partial \mathbf{r}}{\partial p_{20}} \end{array} \right]^\top \tag{26}$$

式 (24) の差分近似を用いて式 (22) の反復公式でパラメー

Algorithm 1 パーティクルフィルタ
<<<block,thread>>> は GPU カーネルの呼び出し

RGB 画像の転送 ($H \rightarrow D$)
距離画像の転送 ($H \rightarrow D$)
3 次元疎テンプレートの転送 ($H \rightarrow D$)
候補姿勢 $\{\mathbf{p}_c\}$ の転送 ($H \rightarrow D$)
乱数ベクトル $\delta\mathbf{p}_k$ の生成 <<< $(K+31)/32, 32$ >>>
サンプル姿勢 $\mathbf{p}_k$ の生成 <<< $(K+31)/32, 32$ >>>
姿勢行列 $\mathcal{D}_k$ の計算 <<< $(K+31)/32, 32$ >>>
ϵ_k^C の計算 <<< K, N >>>
ϵ_k^D の計算 <<< K, N' >>>
評価値 ϵ_k の計算 <<< $(K+31)/32, 32$ >>>
すべてのパーティクルの評価値の転送 ($D \rightarrow H$)

表 1 パーティクルフィルタに関する処理についての CPU と GPU の処理時間の比較 [ms]

項目	CPU	GPU
サンプル姿勢 $\mathbf{p}_k$ の生成	12.48	1.90
姿勢行列 $\mathcal{D}_k$ の計算	20.58	1.10
ϵ_k^C の計算	318.22	2.04(3.31)
ϵ_k^D の計算	275.04	2.02(2.49)

タを最適化する．

肩関節を追加した姿勢の推定は，式 (13) を用いて同様に計算できる．このとき，式 (15) は以下の式のように拡張される．

$$\delta\mathbf{D}'_k \circ \hat{\mathbf{D}}_{OU} = \begin{bmatrix} \mathbf{R}_y^{p_j^+}\mathbf{R}_z^{p_{j+1}^+} & \mathbf{t}_S + \delta\mathbf{t}'_k \\ \mathbf{0}^\top & 1 \end{bmatrix} \tag{27}$$

ただし，$\delta\mathbf{t}'_k = [\delta p_l\ \delta p_{l+1}\ \delta p_{l+2}]^\top$，$\mathbf{t}_S = [p_l\ p_{l+1}\ p_{l+2}]^\top$ であり，式 (27) 中において, 右肩は $j=7$，$l=15$, 左肩では $j=11$，$l=18$ である．

4. GPU による高速化

本節では，3 章で述べたパーティクルフィルタ及び LM 法による姿勢追跡を高速化する方法について述べる．本稿では，CUDA を用いて GPU で計算を行い追跡を高速化する．一般に，CPU メモリと GPU メモリ間のデータ転送はオーバヘッドになるので，GPU で計算を始めると，関連する処理もすべて GPU で行い，結果のみを転送する必要がある．本稿では，その点に留意して GPU 化を行う．

4.1 パーティクルフィルタ

パーティクルフィルタでは，すべてのパーティクルの計算を並列に行うことができる．まず，複数 (ここでは 10 個とする) の候補姿勢 $\{\mathbf{p}_c\}$ から K 個 (例えば 1500) のサンプル姿勢 $\mathbf{p}_k$ を生成する．大量の正規乱数を高速に生成するために cuRAND ライブラリを利用する．次に，$\mathbf{p}_k$ から姿勢行列 $\mathcal{D}_k$ を計算する．その後，RGB 画像と距離画像に対する評価値を求める．この際，ε_k^C，ε_k^D を計算するカーネルは K ブロック N スレッドで実行する．これにより，N 点の合計をリダクション (スレッド間通信) により

Algorithm 2 Levenberg-Marquardt 法
<<<block,thread>>> は GPU カーネルの呼び出し

候補姿勢 $\{\mathbf{p}_c\}$ の転送 ($H \rightarrow D$)
$\mathbf{p}^{(\tau)}$ の評価値の計算 <<< $50, N+N'$ >>>
for (int i=1; i<=m; i++) {
  $\mathbf{p} + e_n\varepsilon$，$\mathbf{p} - e_n\varepsilon$ の生成 <<< $(280\times 14+31)/32, 32$ >>>
  $\mathbf{r}(\mathbf{p})$ の計算 (RGB 画像)<<< $280, N$ >>>
  $\mathbf{r}(\mathbf{p})$ の計算 (距離画像)<<< $280, N$ >>>
  ヤコビ行列 $\mathbf{J}$ の計算 <<< $(10,14), N+N'$ >>>
  $\mathbf{J}^\top\mathbf{r}$ の計算 <<< $(10,14), N+N'$ >>>
  $\mathbf{J}^\top\mathbf{J}$ の計算 <<< $10, (16,16)$ >>>
  $(\mathbf{J}^\top\mathbf{J}+C\mathbf{I})^{-1}\mathbf{J}^\top\mathbf{r}$ の計算 <<< $10, 14$ >>>
  $(\mathbf{J}^\top\mathbf{J}+C\mathbf{I})^{-1}\mathbf{J}^\top\mathbf{r}$ の転送 ($D \rightarrow H$)
  $\mathbf{p}^{(\tau+i)}$ の評価値の計算 <<< $50, N+N'$ >>>
}

計算する必要があるが，各スレッドで評価値を計算するよりも高速であった (表 1 の括弧内の値が各スレッドで評価値を計算した場合の処理時間を示す)．最後に，各サンプル姿勢の評価値 ϵ_k を求め，それをホストメモリに転送する．Algorithm1 にパーティクルフィルタのアルゴリズムを示す．

また，CPU と GPU の処理時間の比較を表 1 に示す．表 1 の時間はパーティクルフィルタが 3 段，姿勢サンプル数 K が 1500，3 次元疎テンプレートの点数 N=128，N'=128 で追跡を行った場合の 1 フレーム当りの処理時間である．なお，比較に用いた CPU，GPU の詳細は 5 章で述べる．表 1 より，特に ϵ_k^C，ϵ_k^D の計算にかかる時間が大きく短縮され，GPU 化の効果が大きいことが確認できる．これは，カーネルが $K\times N$，$K\times N'$ と高い並列度で実行されるためである．一方，姿勢サンプル生成の処理時間がそれほど短縮されていないのは，ϵ_k^C，ϵ_k^D の計算に比べ，並列度が低いからである．

4.2 LM 法

LM 法では，10 個の候補姿勢に対し並列で計算を行い，これを m 回反復する．図 2 に示す近似ヤコビ行列 $\mathbf{J}$ を計算するために，1 つの候補姿勢につき，$\mathbf{p}+e_n\varepsilon$，$\mathbf{p}-e_n\varepsilon(n=1,\cdots,14)$ で 28 個の姿勢を生成し，$\mathbf{J}$ を求める．また，各姿勢に対し $\mathbf{r}(\mathbf{p})$ を計算する．次に，$(\mathbf{J}^\top\mathbf{J}+C\mathbf{I})^{-1}\mathbf{J}^\top\mathbf{r}$ の行列演算に対し，cuBLAS，CULA といったライブラリを利用することが考えられるが，本稿で取り扱う $\mathbf{J}$ はサイズがあまり大きくない上に，10 個の行列の計算を同時に行うため，上述のライブラリを利用するよりも，独自のカーネルで処理を行う方が計算時間が短かった．Algorithm2 に LM 法のアルゴリズムを示す．

また，表 2 に LM 法の反復を 10 回行った場合の 1 フレーム当りの処理時間の比較を示す．表 2 から $\mathbf{r}(\mathbf{p})$ の計算において，GPU では CPU に比べ処理時間が大きく短縮されており，GPU 化の効果が大きいことが確認できる．これはパーティクルフィルタの場合と同様に，カーネルが高い

表 2 LM 法に関する処理についての CPU と GPU の処理時間の比較 [ms]

項目	CPU	GPU
$\mathbf{p}+e_n\varepsilon$，$\mathbf{p}-e_n\varepsilon$ の生成	1.66	0.99
$\mathbf{r}(\mathbf{p})$ の計算 (RGB)	36.06	1.29
$\mathbf{r}(\mathbf{p})$ の計算 (距離)	35.55	0.99
ヤコビ行列 $\mathbf{J}$ の計算	0.62	0.62
$\mathbf{J}^\top\mathbf{r}$ の計算	13.49	1.92
$\mathbf{J}^\top\mathbf{J}$ の計算	2.02	0.90
$(\mathbf{J}^\top\mathbf{J}+C\mathbf{I})^{-1}\mathbf{J}^\top\mathbf{r}$ の計算	3.93	1.17
$\mathbf{p}^{(\tau)}$，$\mathbf{p}^{(\tau+i)}$ の評価値の計算	12.50	1.50


右上腕　胴体　左上腕

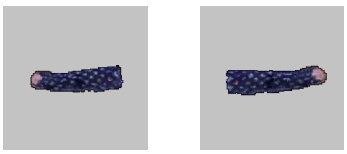
右下腕　左下腕

図 4 上半身モデル

並列度で実行されるからである．一方，一部に関しては最適化が 不十分であるが，さらなる処理時間の短縮が可能であると考えられる．

5. 実験

本手法の効果を確認するために，シミュレーションと実動画を用いて人物姿勢追跡，及び肩関節パラメータの推定を含めた姿勢追跡の実験を行った．実験に使用した CPU は Intel Core i7-4790K であり, 動作周波数は 4.0GHz, コア数は 4 である．また GPU は NVIDIA GeForce GTX 980 であり，GPU クロックは 1253MHz，コア数は 2048 である．実験には Visual Studio Express 2013 for Windows Desktop と CUDA 7.0 を用いた．また，本稿では RGBD センサとして Kinect を用いた．距離情報は mm 単位で表され，500mm 以上離れた対象について距離情報がリアルタイムで得られる．また RGB 画像，距離画像ともに解像度は 640×480，フレームレートは 30fps である．

人物モデルは KinectFusion[8] を用いて作成した人物モデルを用いる．図 4 に実験に使用した人物モデルの例を示す．

5.1 人物姿勢追跡実験

人物姿勢追跡について，7 種類のシミュレーション動画を用いてパーティクルフィルタによる追跡，パーティクルフィルタ+LM 法による追跡を CPU と GPU で行い，追跡精度と処理速度の比較を行った．シミュレーション動画は Kinect の距離分解能の粗さに合わせて，入力距離値を 5[mm] 単位としている．

表 3 実験条件

項目	設定値
姿勢サンプル数	1500
粗密探索の段数	2∼3
初段のサンプル生成範囲 (σ)	表 4 参照
式 (17) の N，及び式 (18) の N' の値	128
式 (19) における距離情報の比率 (λ)	0.90

表 4 初段の乱数の生成範囲 (σ_n)

胴体 [mm]						肩 [deg]		肘 [deg]	
t_x	t_y	t_z	θ_O	ϕ_O	ψ_O	ϕ_S	θ_S	ϕ_E	θ_E
5.0	5.0	5.0	1.0	3.0	1.0	8.0	8.0	15.0	15.0

詳細な実験条件を表 3，4 に示す．精度評価の尺度として，推定姿勢の左右上腕，下腕のそれぞれの主軸 (x 軸) とシミュレーションにおける正解姿勢の主軸がなす角度を用いる．

5.1.1 パーティクルフィルタによる追跡実験

CPU と GPU で各動画に対し乱数系列を変えて 10 回試行した．パーティクルフィルタの段数は 3 段とする．各部位の平均誤差と 1 フレーム当りの処理時間の比較を表 5 に示す．表 5 から，GPU を用いることで，追跡精度を保ったまま処理時間が大きく短縮されていることがわかる．パーティクルフィルタでは，大量のパーティクルの計算を並列に行うことができるため，GPU 化の効果が非常に大きいことが確認できた．

5.1.2 LM 法

次に，パーティクルフィルタと LM 法を併用した追跡について，乱数系列を変えて 10 回試行した．パーティクルフィルタの段数は 3 段とし，LM 法の反復回数は 10 回とする．LM 法を CPU と GPU で行った場合の各部位の平均誤差と 1 フレーム当りの処理時間の比較を表 6 に示す．表 6 から LM 法を GPU で行うことにより，追跡精度を保ったまま CPU に比べ処理時間が短縮されていることがわかる．

また，図 5 にパーティクルフィルタと LM 法による追跡の各部位についての平均をとったグラフを示す．図 5 の PFx は x 段のパーティクルフィルタを表し，LMy は LM 法の反復回数が y 回であることを表す．パーティクルフィルタは 2 段，3 段，LM 法の反復回数は 0，10，30 回でそれぞれ行った．図 5 から PF2 の場合，LM10 では，LM0 に比べ誤差が大きく減少しているのがわかる．一方，LM10 に比べ LM30 では誤差の減少速度が低下した．これは，ヤコビ行列 $\mathbf{J}$ を図 2 に示した形に近似したことで，LM 法による最適化が小さな姿勢の変動に対し，うまく動作していないためであると考えられる．また，PF3 では LM0，LM10，

表 5 パーティクルフィルタによる追跡における誤差平均 [deg] と処理時間 [ms]

	動画	右上腕	左上腕	右下腕	左下腕	処理時間
CPU	no1	0.826	0.790	0.675	0.723	581
	no2	0.735	0.693	0.670	0.674	587
	no3	0.743	0.763	0.599	0.694	604
	no4	0.800	0.856	0.715	0.753	578
	no5	0.559	0.775	0.708	0.725	577
	no6	0.622	0.767	0.691	0.701	573
	no7	0.593	0.627	0.603	0.561	591
GPU	no1	0.831	0.772	0.678	0.705	20.6
	no2	0.738	0.707	0.673	0.680	20.6
	no3	0.720	0.724	0.582	0.668	20.5
	no4	0.856	0.836	0.737	0.718	20.6
	no5	0.582	0.778	0.731	0.729	20.4
	no6	0.609	0.771	0.636	0.692	20.6
	no7	0.589	0.607	0.635	0.549	20.6

表 6 パーティクルフィルタ+LM 法による追跡における誤差平均 [deg] と処理時間 [ms]

	動画	右上腕	左上腕	右下腕	左下腕	処理時間
CPU	no1	0.782	0.742	0.558	0.647	119
	no2	0.675	0.692	0.557	0.610	120
	no3	0.670	0.677	0.508	0.580	124
	no4	0.754	0.821	0.596	0.632	122
	no5	0.547	0.725	0.647	0.629	129
	no6	0.578	0.719	0.533	0.583	124
	no7	0.569	0.607	0.529	0.511	126
GPU	no1	0.758	0.776	0.523	0.691	29.0
	no2	0.685	0.691	0.530	0.602	28.8
	no3	0.652	0.699	0.464	0.593	29.2
	no4	0.785	0.810	0.577	0.620	29.2
	no5	0.538	0.713	0.633	0.633	29.4
	no6	0.569	0.713	0.549	0.588	28.9
	no7	0.577	0.636	0.512	0.531	28.8

LM30 のいずれも誤差の減少が遅かった．これも，PF2 と同様の理由であると考えられる．

5.2 モデルパラメータ推定実験

5.2.1 シミュレーション実験

シミュレーション動画において，肩関節パラメータの推定実験を行った．シミュレーションにおける正解の肩関節パラメータに乱数を用いて 30[mm] の範囲でノイズを加え，LM 法を用いて肩関節の推定を行い，正解の肩関節パラメータとの誤差を調べた．なお，オクルージョンのある動作では肩関節パラメータの推定は困難であるため，本稿では肩関節パラメータを推定しやすい動作を用いる．この動作は，胴体の横で両腕を広げて，肩を中心として下へ動かすものである．あるフレーム (腕をおろす) まで肩関節の位置を追跡し，最後のフレームの関節位置を推定結果とする．シミュレーション実験では，パーティクルフィルタの段数を 1 段，LM 法の反復回数を 50 回で肩関節パラメータの推定を行った．その他の実験条件は表 3，4 と同じである．乱数系列を変えて 30 回試行し，推定した肩関節パラメータと正解の肩関節パラメータとの誤差の平均を表 7 に示す．表 7 の結果より，高精度な推定が行えることが確認できる．このようなモデルパラメータの推定はパーティクルフィルタでは不可能であったが，LM 法を適用することにより，追跡と同時に関節パラメータの推定が行えた．なお，この実験では，追跡開始から 1 フレーム後には正解のパラメータに近い値に推定が行われた．

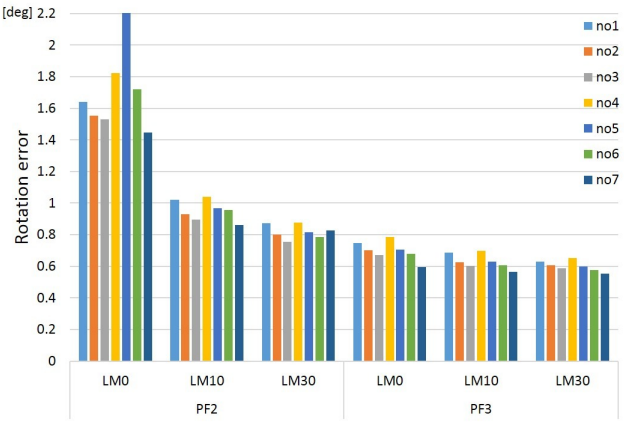

図 5 全関節の平均回転誤差

表 7 肩関節推定実験結果 (シミュレーション)[mm]

右肩			左肩		
t_x	t_y	t_z	t_x	t_y	t_z
0.237	0.311	0.267	0.234	0.164	0.241

5.2.2 実動画実験

次に実動画において，肩関節パラメータの推定実験を行った．ここでは精度を重視し，パーティクルフィルタを 3 段，LM 法の反復回数を 50 回として推定を行った．まず，実動画における肩関節パラメータの推定性能を調べるために，手動で調整を行った肩関節パラメータに乱数でノイズを加えた値を初期値として推定を行った．ノイズは 5.2.1 の実験における精度の高さを考慮して 50[mm] とした．

人物 3 人に対して推定を行い，その様子の例を図 6 に示す．図の緑色の点がノイズを加えた肩関節の初期位置を示し，黄色の点が推定される肩関節の位置を示す．図 6 の 1 番下の図が最終推定結果である．50[mm] という大きなノイズを加えたにも関わらず，初期位置から正しい位置に修正されていることが確認できる．

次に，この 3 人の関節パラメータを平均したものを一般均なパラメータとし，3 人とは別の人物に対して人物固有の肩関節パラメータの推定を行った．推定によって得られた固有のパラメータと平均のパラメータとの差は，それぞれ 10[mm] 以内であった．そして，その推定によって得ら

図 6 肩関節推定実験-入力画像 (左) と推定の様子 (右)

れた固有の関節パラメータを用いて，7 種類の動画に対して追跡実験を行った．この実験により，固有のパラメータを用いた場合，安定した追跡が行われることを確認した．また，一般の平均パラメータを用いた追跡も行ったが，全ての動画で安定した追跡を行うことはできなかった．この結果から，肩関節のモデルパラメータの調整を推定に統合することで，追跡精度が向上することを確認できた．

なお，今回のパラメータ推定では，一般の平均を 3 人の平均として行ったが，実際にはより多人数のデータを取り平均を計算する必要があると考えられる．また，より実用的にするには更に精度を高める必要がある．

6. まとめ

本稿では，人物姿勢追跡の高速化と人体モデルパラメータ推定を検討し，実画像及びシミュレーションで効果を確認した．まず，GPU を利用することにより，パーティクルフィルタ 3 段の後に LM 法を 10 回反復することが実時間で可能となることを示した．この処理時間は従来の 1/20 以下に相当するが，追跡精度を低下させる近似を行ってはいない．また，LM 法では探索空間の次元を増やすことが容易であることに着目し，姿勢追跡と同時に人体モデルの肩関節のパラメータ推定を行う方法を検討した．これにより，追跡時に人体モデルを自動的に調整できることになり，前もって人物毎に人体モデルを調整する必要がなくなった．

今後の課題として，実時間インタフェースの実現，LM 法の更なる高速化、肩関節のパラメータ推定を含んだ追跡の実時間化などが挙げられる．

参考文献

[1] Bray, M., Loller-Meier, E. and Gool, L. V.: Smart Particle for High-Dimensional Tracking, *Computer Vision and Image Understanding*, pp. 116–129 (2007).
[2] 加藤秀章，小林宏章，山根　亮，尺長　健：カスケード姿勢推定に基づく距離 · 画像センサを用いたジェスチャ認識，情報処理学会研究報告 2014-PRMU-113，pp. 81–88 (2014).
[3] 筒井健斗，溝部　諒，右田剛史，尺長　健：パーティクルフィルタと Levenberg-Marquardt 法による顔 · 人物姿勢追跡の検討，情報処理学会研究報告 2015-CVIM-195，pp. 1–8 (2015).
[4] Press, W. H., Teukolsky, S. A., Vetterling, W. T. and Flannery, B. P.: *Numerical Recipes 3rd Edition: The Art of Scientific Computing*, Cambridge University Press (2007).
[5] 四宮洋平，加藤秀章，尺長　健：Kinect を用いた 3 次元疎テンプレートによる人物姿勢追跡，電子情報通信学会技術研究報告 PRMU2011-111，pp. 153–158 (2012).
[6] 松原康晴，尺長　健：疎テンプレートマッチングとその実時間物体追跡への応用，情報処理学会論文誌，Vol. 46, No. 1, pp. 60–71 (2005).
[7] Oka, Y., Kuroda, T., Migita, T. and Shakunaga, T.: Tracking 3d pose of rigid object by 3d sparse template matching, *In Proceedings of International Conference on Image and Graphics (ICIG2009)*, pp. 390–397 (2009).
[8] Newcombe, R. A., Izadi, S., Hilliges, O., Molyneaux, D., Kim, D., Davison, A. J., Kohli, P., Shotton, J., Hodges, S. and Fitzgibbon, A.: KinectFusion: Real-Time Dense Surface Mapping and Tracking, *Proc. International Symposium on Mixed and Augmented Reality (ISMAR2011)*, pp. 127–136 (2011).