Vol.2017-CVIM-206 No.10
2017/3/10

Rao-Blackwellized 粒子フィルタを用いた手の3次元姿勢追跡手法

梶原 星平[1,a)]　田中 利幸[1,b)]

概要：単一の深度センサ Kinect を用い，手の姿勢を追跡する新たな手法を提案する．提案手法は，手の姿勢を表現する状態変数が時間発展し，各時刻の状態変数により定まる手の 3 次元形状モデルから深度画像が観測されるという状態空間モデルに基づき，粒子フィルタを用いた状態推定により手の姿勢追跡を行う．尤度関数は，手の姿勢ベクトルに対してレンダリングされる深度画像とセンサから得た深度画像との誤差に基づいて定義される．モデル化をより現実に即したものにするためには，手の姿勢ベクトルだけでなくその時間微分を状態変数に含めるのが望ましいが，計算コストが増加してしまうという問題がある．提案手法では，時間微分の時間発展に関する部分構造が線形で記述されることに着目して Rao-Blackwellized 粒子フィルタを用いる，これにより，僅かな計算コストの増加で追跡の精度を向上させることができる．実験の結果，追跡精度の向上を支持する根拠は得られなかったが，追加の計算コストがほとんど無視できるほど小さいものであることが示された．

1. はじめに

本稿では，単一の深度センサを用いて人間の手の姿勢を推定する新たな手法を提案する．結果として得られる姿勢の列は，ゲームやコンピュータの操作といった多くの応用領域においてヒューマン・コンピュータ・インタラクションの向上に役立てることができる．この手法は高価な機器や複雑なセットアップを要しないため，実時間での処理を達成できれば，一般の消費者にも利用可能となる．

安価な機器を用いた手の姿勢推定が実現できれば，日々の生活の質を向上させることができる．例えば，ビデオゲームはより自然で刺激的な体験を提供し，コンピュータはお年寄りにも使いやすいものになる．これらの応用によって体験の質を向上させ情報格差を減らすためには，手の姿勢追跡の手法は実時間処理である必要があり，機器も安価でなければならない．

手の姿勢を連続的に認識することは本質的に困難であるが，深度センサを使えばこの問題はより扱いやすいものになる．深度画像は 2 次元の色画像と比較して 3 次元の手の構造についての情報を多く有しているため，手の候補と現実の手との一致度合いをより正確に評価することができる．したがって，多くの候補を生成し，現実の手とそれらとの一致度合いを評価し，最良のものを選ぶ，あるいは得点に基づいた重み付き平均を取ることで姿勢を推定することができる．この単純な戦略は上手く動作し，提案手法の基礎を成している．

本稿では，手の姿勢の列を明示的にベクトルの時系列と見なし，粒子フィルタを用いて推定する．時系列における連続的な推定はしばしば状態空間モデルにおけるフィルタリング手法で行われる．任意の確率分布を持ったモデルを扱える粒子フィルタ [9] とその変種は，手の姿勢推定に使われている [2], [6]．これらの手法において状態空間モデルは手の姿勢だけで構成された状態変数を用いて定義されているが，提案手法では，これに加えて姿勢の時間微分を状態変数に追加する．この方法は一見より多くの計算量を必要とするように見えるが，姿勢の時間微分の空間を線形構造として構築し，Rao-Blackwellized 粒子フィルタ [10] を適用することで，速度の追跡に要する追加の計算量を小さくできる．したがって，この手法は現実の手の姿勢を表現するモデルの正確さを高め，結果として，わずかな追加コストで追跡の精度を高めることが期待できる．

2. 関連研究

データグローブを用いた手法 [3] や，通常のカメラとカラーグローブを用いた手法 [12] は，前節で述べた目的には適さない．このようなグローブはユーザーにとって装着が

[1] 京都大学情報学研究科システム科学専攻
Department of Systems Science, Graduate School of Informatics, Kyoto University
[a)] kajihara@sys.i.kyoto-u.ac.jp
[b)] tt@sys.i.kyoto-u.ac.jp

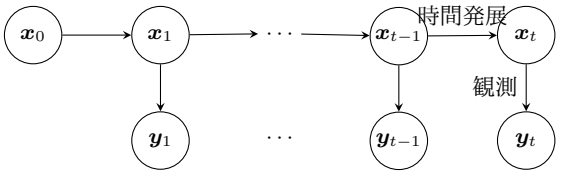

図 1 手の姿勢追跡のための状態空間モデル．上段のノード $\boldsymbol{x}_t,\ t = 0, 1, 2, \ldots$ は手の姿勢とその時間微分を表す．下段のノード $\boldsymbol{y}_t,\ t = 1, 2, \ldots$ は深度センサで観測される深度画像を表す．

面倒であり，没入感を阻害する要因となる．カメラだけを用いた手法 [11] は，指の色が一様であるため手の姿勢追跡の問題が過度に難しくなり，実時間の目的にはそぐわない．

一方で，深度センサを用いた手法は最初の較正を除けば設定の必要がなく，それでいて 3 次元空間についての豊富な情報が得られる．既存の深度センサは高価であったが，Kinect はその価格の低さによって手の姿勢追跡を含めたコンピュータビジョンの領域を加速させた．例えば [7] は，粒子群最適化を用いてほぼ実時間（15Hz）の処理で正確かつ頑健な追跡を行った．

しかしながら，[7] のアルゴリズムは時間発展の情報をほとんど利用していないため，手の姿勢が急激には変わらない状況でのみ効果的に働く．素早く動く手を追跡するためには，手の姿勢の時間発展を明示的に扱う必要がある．

3. 問題定義

提案手法では，姿勢の速度が線形部分構造を成すような状態空間モデにおいて手の姿勢追跡を行う．詳細に入る前に，本節で問題と変数の定義を行う．

$\boldsymbol{x}_t, \boldsymbol{y}_t$ をそれぞれ時刻 t における状態変数，観測変数とする．$\boldsymbol{x}_t$ は手の姿勢とその時間微分，$\boldsymbol{y}_t$ は深度画像である．これらの変数が状態空間を構成する，すなわち，状態変数が確率 $p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ で時間発展し，$\boldsymbol{y}_t$ が $\boldsymbol{x}_t$ から確率 $p(\boldsymbol{y}|\boldsymbol{x}_t)$ で生成されるものと仮定する．これらの関係性を**図 1** に示す．時刻 t における手の姿勢と速度 $\boldsymbol{x}_t$ を，深度画像 $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_t$ が与えられたもとで推定することが目的となる．

上記のモデルから，以下の手順で線形部分構造を抽出することができる．まず，$\boldsymbol{x}_t$ を非線形な部分 $\boldsymbol{x}_t^n$ と線形な部分 $\boldsymbol{x}_t^l$ に分解する．これらはそれぞれ，手の姿勢 $\boldsymbol{p}_t$ とその時間微分 $\frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{p}_t$ とする．

$$\boldsymbol{x}_t := \begin{bmatrix} \boldsymbol{x}_t^n \\ \boldsymbol{x}_t^l \end{bmatrix} = \begin{bmatrix} \boldsymbol{p}_t \\ \frac{\mathrm{d}}{\mathrm{d}t}\boldsymbol{p}_t \end{bmatrix}. \tag{1}$$

ここに，$\boldsymbol{p}_t$ は，付録 A.1 で定義する時刻 t における手の姿勢を表す 22 次元のベクトルである．これに加えて，変数間の依存性は確率密度関数 $p(\boldsymbol{x}_t^n|\boldsymbol{x}_{t-1}^n, \boldsymbol{x}_{t-1}^l)$, $p(\boldsymbol{x}_t^l|\boldsymbol{x}_{t-1}^l)$, $p(\boldsymbol{y}_t|\boldsymbol{x}_t^n)$ で表されるものとする．これらの関係性を図 2 に示す．姿勢の速度の列 $\boldsymbol{x}_t^l,\ t = 0, 1, \ldots$ が観測に直接の影

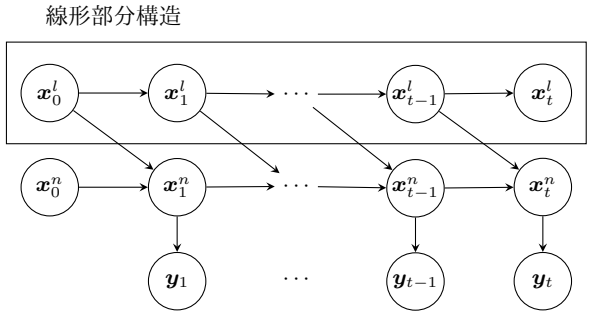

図 2 手の姿勢追跡のための線形部分構造を持った状態空間モデル．上段のノード $\boldsymbol{x}_t^l,\ t = 0, 1, 2, \ldots$ は手の姿勢の速度を表し，中段のノード $\boldsymbol{x}_t^n,\ t = 0, 1, 2, \ldots$ は手の姿勢を，下段のノード $\boldsymbol{y}_t,\ t = 1, 2, \ldots$ は観測される深度画像を表す．

響を与えていないことが分かる．$\boldsymbol{x}_t^n$ の $\boldsymbol{x}_{t-1}^l$ に対する依存性を線形かつガウス的であるすると，線形な部分に対するカルマンフィルタと非線形な部分に対する粒子フィルタの組み合わせて Rao-Blackwellized 粒子フィルタ [10] を適用することができる．

手の姿勢とその速度を追跡するためには，上述の確率分布を定義する必要がある．提案手法では時間発展に単純な線形変換とガウス分布の形式を用い，観測にはセンサの深度画像とレンダリングされた深度画像との差に基づくコスト関数から計算されるギブス分布を用いる．コスト関数とフィルタリング技法はそれぞれ第 4 節と第 5 節で説明する．

4. コスト関数の計算

4.1 手の姿勢に基づく深度画像の生成

本稿では，深度画像 $\boldsymbol{y}$ は手の姿勢 $\boldsymbol{x}$ により決定される深度画像 $\boldsymbol{d}(\boldsymbol{x})$ にノイズが加わったものとして生成されるものとする．

$$\boldsymbol{y} = \boldsymbol{d}(\boldsymbol{x}) + \boldsymbol{e}. \tag{2}$$

この確率モデルに基づいて，各時刻においてノイズを含んだ深度画像 $\boldsymbol{y}$ が深度センサから得られるものとする．本稿では深度センサとして Kinect を用いるため，$\boldsymbol{y}$ は $Y = 512 \times 424 = 217088$ 次元の実ベクトルである．各要素はカメラから最も近い点の奥行きをミリメートル単位で表すピクセルの値である．

深度画像生成の過程を OpenGL を用いて再現することができる．$\boldsymbol{x}$ を手の姿勢を表すベクトルとする．手のメッシュモデルは予め用意してあるものとする．このメッシュモデルを $\boldsymbol{x}$ の値に基づき変形し，仮想的なカメラを通してレンダリングすることで，深度画像 $\boldsymbol{d}(\boldsymbol{x})$ を得る．この深度画像はセンサで観測されたものと同じ次元数を持つ．

$\boldsymbol{e}$ の確率密度関数を明示的に定義することは避け，$p(\boldsymbol{y}|\boldsymbol{x})$ をギブス分布

$$p(\boldsymbol{y}|\boldsymbol{x}) \propto \exp\left[-\frac{D(\boldsymbol{d}(\boldsymbol{x}), \boldsymbol{y})}{\sigma}\right], \tag{3}$$

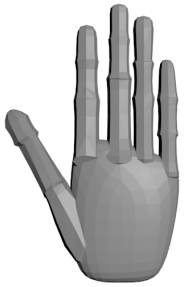

図 3 手のメッシュモデル.

で定義する．ここに，$D(\cdot,\cdot)$ は次節で定義されるコスト関数であり，$\sigma > 0$ は画像に含まれるノイズの大きさを表す定数である．式 (3) は，$\boldsymbol{d}(\boldsymbol{x})$ と $\boldsymbol{y}$ が似ていればコストは小さくなり，尤度が大きくなるということを意味している．ノイズ $\boldsymbol{e}$ は $\boldsymbol{d}(\boldsymbol{x})$ に依存することに注意する.

メッシュモデルの選び方には複数の選択肢がある．正確なモデル [4] は写実的な深度画像を生成し，より正確にコスト関数の評価を行える．しかし，レンダリングに多くの時間を要する．一方で，単純なモデル [8] はインスタンスレンダリングの技法を用いることでより素早くレンダリングできるが，正確さに欠ける．本来，深度画像の正確さを高め，レンダリング時間を削減するようなメッシュモデルを設計する必要があるが，本稿ではこの議論を省き，図 3 に示されるメッシュモデルを用いる．このモデルは，3D モデリングソフトウェアである Blender を用いて設計した.

4.2 一致度合いの計算

2 つの深度画像 $\boldsymbol{d}(\boldsymbol{x}) = [d_1, \ldots, d_Y]^{\mathrm{T}}$, $\boldsymbol{y} = [y_1, \ldots, y_Y]^{\mathrm{T}}$ が与えられとき，コスト関数 $D(\boldsymbol{d}(\boldsymbol{x}), \boldsymbol{y})$ をピクセル毎の深度の差異に基づいて以下のように定義する.

$$D(\boldsymbol{d}(\boldsymbol{x}), \boldsymbol{y}) = \frac{1}{|R|} \sum_{i \in R} \min\{|d_i - y_i|,\ c_{\max}\}. \tag{4}$$

ここに，R は計算に用いるピクセルの添字集合であり，$c_{\max} > 0$ は一致しないピクセルが全体のコストに影響を与えすぎないようにするための閾値である．R は，前時刻の手の位置の推定値を中心とした矩形領域のうち，レンダリングされた画像 $\boldsymbol{d}(\boldsymbol{x})$ の中で有効なピクセルとする．式 (4) は，対象ピクセルにおける深度の差の平均が小さければ 2 つの深度画像が似ているということを意味する.

5. 時系列の推定

5.1 状態空間モデル

状態空間モデルは，隠れ変数に対応する観測が条件付き分布で与えられる隠れマルコフモデルであり [1]，しばしば以下の形式で記述される [9].

$$\boldsymbol{x}_0 \sim p(\boldsymbol{x}_0), \tag{5}$$

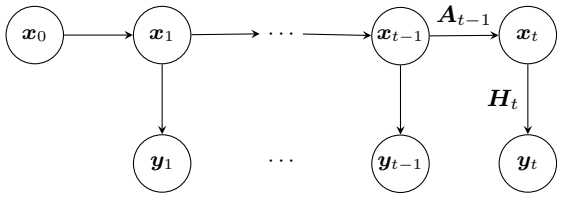

図 4 線形ガウスの状態空間モデル．上段のノード $\boldsymbol{x}_t$, $t = 0, 1, 2, \ldots$ は隠れ変数であり，下段のノード $\boldsymbol{y}_t$, $t = 1, 2, \ldots$ は観測変数である.

$$\boldsymbol{x}_t \sim p(\boldsymbol{x}_t | \boldsymbol{x}_{t-1}),\ t = 1, 2, \ldots \tag{6}$$

$$\boldsymbol{y}_t \sim p(\boldsymbol{x}_t),\ t = 1, 2, \ldots. \tag{7}$$

$\boldsymbol{x}_t$, $t = 1, 2, \ldots$ は時間発展する隠れ変数であり，$\boldsymbol{y}_t$, $t = 1, 2, \ldots$ はそれらに対応する観測である．状態空間モデルにおける典型的な問題の 1 つは，各時刻 t において，$\boldsymbol{y}_{1:t} := \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_t\}$ から $\boldsymbol{x}_t$ を推定することである.

5.2 カルマンフィルタ

カルマンフィルタは，時間発展と観測のモデルが線形ガウスである状態空間

$$\begin{aligned} \boldsymbol{x}_t &= \boldsymbol{A}_{t-1}\boldsymbol{x}_{t-1} + \boldsymbol{w}_{t-1}, \\ \boldsymbol{y}_t &= \boldsymbol{H}_t\boldsymbol{x}_t + \boldsymbol{e}_t \end{aligned} \tag{8}$$

における逐次推定の問題に対し閉形式の解を与える [9]．ここに，事前分布はガウス分布 $p(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{m}_0, \boldsymbol{P}_0)$ であり，$\boldsymbol{A}_t$ は遷移行列，$\boldsymbol{H}_t$ は観測行列，$\boldsymbol{w}_t, \boldsymbol{e}_t$ は以下の分布に従うノイズである.

$$\boldsymbol{w}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_t), \tag{9}$$

$$\boldsymbol{e}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_t). \tag{10}$$

これを図 4 に示す.

カルマンフィルタは $\boldsymbol{x}_t$ の事後分布を閉形式で与える.

$$p(\boldsymbol{x}_t | \boldsymbol{y}_{1:t}) = \mathcal{N}(\boldsymbol{x}_t | \boldsymbol{m}_t, \boldsymbol{P}_t). \tag{11}$$

これは，各時刻 $t = 1, 2, \ldots$ において以下の 2 ステップで計算される.

予測ステップ

$$\begin{aligned} \boldsymbol{m}_t^- &= \boldsymbol{A}_{t-1}\boldsymbol{m}_{t-1}, \\ \boldsymbol{P}_t^- &= \boldsymbol{A}_{t-1}\boldsymbol{P}_{t-1}\boldsymbol{A}_{t-1}^{\mathrm{T}} + \boldsymbol{Q}_{t-1}. \end{aligned} \tag{12}$$

更新ステップ

$$\begin{aligned} \boldsymbol{v}_t &= \boldsymbol{y}_t - \boldsymbol{H}_t\boldsymbol{m}_t^-, \\ \boldsymbol{S}_t &= \boldsymbol{H}_t\boldsymbol{P}_t^-\boldsymbol{H}_t^{\mathrm{T}} + \boldsymbol{R}_t, \\ \boldsymbol{K}_t &= \boldsymbol{P}_t^-\boldsymbol{H}_t^{\mathrm{T}}\boldsymbol{S}_t^{-1}, \\ \boldsymbol{m}_t &= \boldsymbol{m}_t^- + \boldsymbol{K}_t\boldsymbol{v}_t, \\ \boldsymbol{P}_t &= \boldsymbol{P}_t^- - \boldsymbol{K}_t\boldsymbol{S}_t\boldsymbol{K}_t^{\mathrm{T}}. \end{aligned} \tag{13}$$

低次元の行列演算のみで構成されているため，これらの計算にはほとんど時間が掛からない.

5.3 粒子フィルタ

粒子フィルタは非線形または非ガウス的な状態空間モデルにおいても隠れ変数を推定することができる手法である．N 個の粒子を用いた粒子フィルタは，以下のように計算される [9].

- 初期分布 $p(\boldsymbol{x}_0)$ から N 個の初期サンプル $\boldsymbol{x}_0^{(i)},\ i = 1,\ldots,N$ を生成する．

$$\boldsymbol{x}_0^{(i)} \sim p(\boldsymbol{x}_0),\ i = 1,\ldots,N. \tag{14}$$

  重みを $w_0^{(i)} = 1/N,\ i = 1,\ldots,N$ とする．
- 各時刻 $t = 1, 2, \ldots$ において以下を実行する．
  (1) 予め設計した提案分布

$$\boldsymbol{x}_t^{(i)} \sim \pi(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}^{(i)}, \boldsymbol{y}_{1:t}),\ i = 1,\ldots,N. \tag{15}$$

  からサンプル $\boldsymbol{x}_t^{(i)}$ を生成する．
  (2) 重みを更新する．

$$w_t^{(i)} \propto w_{t-1}^{(i)} \frac{p(\boldsymbol{y}_t|\boldsymbol{x}_t^{(i)})p(\boldsymbol{x}_t^{(i)}|\boldsymbol{x}_{t-1}^{(i)})}{\pi(\boldsymbol{x}_t^{(i)}|\boldsymbol{x}_{t-1}^{(i)}, \boldsymbol{y}_{1:t})},\quad i = 1,\ldots,N. \tag{16}$$

  和が 1 になるように重み $w_t^{(i)},\ i = 1,\ldots,N$ を正規化する．
  (3) $\boldsymbol{x}_t$ の推定値をサンプルの重み平均で与える．

$$\hat{\boldsymbol{x}}_t := \sum_{i=1}^{N} w_t^{(i)} \boldsymbol{x}_t^{(i)}. \tag{17}$$

  (4) リサンプリングを行う．
    (a) 重み $w_t^{(i)},\ i = 1, 2, \ldots, N$ を，集合 $\{\boldsymbol{x}_t^{(i)} : i = 1,\ldots,N\}$ からサンプル添字 i を得る確率と解釈し，N 個のサンプルを生成する．古いサンプル集合を新しいものに置き換える．
    (b) 重みを均等に $w_t^{(i)} = 1/N,\ i = 1,\ldots,N$ と設定する．

5.4 線形部分構造を持つ状態空間モデル

状態空間モデルに線形な部分構造を追加する．状態変数を非線形な部分と線形な部分の結合で表す．

$$\boldsymbol{x}_t := \begin{bmatrix} \boldsymbol{x}_t^n \\ \boldsymbol{x}_t^l \end{bmatrix}. \tag{18}$$

文献 [10] で定義されている 4 つのモデルのうち，三角形モデルと呼ばれるモデル 2 を採用する．これは，次の様に定式化される．

$$\begin{aligned} \boldsymbol{x}_t^n &= f_{t-1}^n(\boldsymbol{x}_{t-1}^n) + \boldsymbol{A}_{t-1}^n(\boldsymbol{x}_{t-1}^n)\boldsymbol{x}_{t-1}^l + \boldsymbol{w}_{t-1}^n, \\ \boldsymbol{x}_t^l &= \qquad\qquad\quad \boldsymbol{A}_{t-1}^l(\boldsymbol{x}_{t-1}^n)\boldsymbol{x}_{t-1}^l + \boldsymbol{w}_{t-1}^l, \\ \boldsymbol{y}_t^n &= h_t(\boldsymbol{x}_t^n) + \boldsymbol{C}_t(\boldsymbol{x}_t^n)\boldsymbol{x}_t^l + \boldsymbol{e}_t. \end{aligned} \tag{19}$$

ここに，f_t^n, h_t は任意の非線形な関数であり，$\boldsymbol{A}_t^n, \boldsymbol{A}_t^l, \boldsymbol{C}_t$ は $\boldsymbol{x}_t^n$ に依存する行列，$\boldsymbol{w}_t^n, \boldsymbol{w}_t^l, \boldsymbol{e}_t$ は以下の分布に従うノイズである．

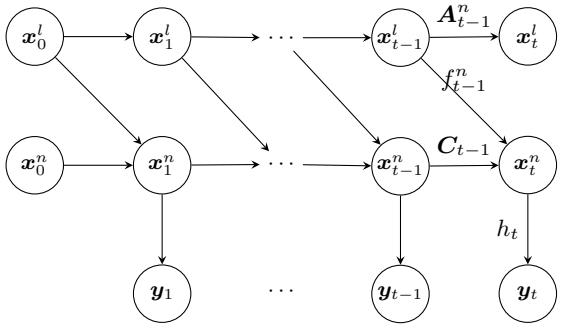

図 5 線形部分構造を持つ状態空間モデル．

$$\boldsymbol{w}_t^n \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_t^n), \tag{20}$$

$$\boldsymbol{w}_t^l \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_t^l), \tag{21}$$

$$\boldsymbol{e}_t \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_t). \tag{22}$$

このモデルを図 5 に示す．

5.5 Rao-Blackwellized 粒子フィルタ

Rao-Blackwellized 粒子フィルタ（周辺化粒子フィルタ）[10] は非線形な部分に対する粒子フィルタと線形な部分に対するカルマンフィルタを組み合わせた手法である．カルマンフィルタを組み込み計算時間を削減することで，粒子フィルタに比べて高次元の状態空間モデルを扱うことができる．

N 個の粒子を用いた Rao-Blackwellized 粒子フィルタは，以下の手順で実行される．

- 非線形な部分を初期化する．N 個のサンプル $\boldsymbol{x}_0^{n,(i)}, i = 1,\ldots,N$ を事前分布 $p(\boldsymbol{x}_0^n)$ から生成し，重みを均等に $w_0^{(i)} = 1/N,\ i = 1,\ldots,N$ とする．
- 線形な部分を初期化する．$\boldsymbol{P}_0$ と $\boldsymbol{x}_0^l$ の初期値を与え，$\boldsymbol{P}_0^{(i)} = \boldsymbol{P}_0,\ \boldsymbol{x}_0^{l,(i)} = \boldsymbol{x}_0^l, i = 1,\ldots,N$ とする．
- 各時刻 $t = 1, 2, \ldots$ において以下を実行する．
  (1) 非線形な部分のサンプルを $\boldsymbol{x}_t^{n,(i)},\ i = 1,\ldots,N$ を提案分布

$$\boldsymbol{x}_t^{n,(i)} \sim \pi(\boldsymbol{x}_t^n|\boldsymbol{x}_{t-1}^{n,(i)}, \boldsymbol{x}_{t-1}^{l,(i)}, \boldsymbol{P}^{(i)}, \boldsymbol{y}_{1:t}),\quad i = 1,\ldots,N \tag{23}$$

  から生成する．
  (2) 重みを更新する．

$$w_t^{(i)} \propto w_{t-1}^{(i)} \frac{p(\boldsymbol{y}_t|\boldsymbol{x}_t^{n,(i)})p(\boldsymbol{x}_t^{n,(i)}|\boldsymbol{x}_{t-1}^{n,(i)}, \boldsymbol{x}_{t-1}^{l,(i)}))}{\pi(\boldsymbol{x}_t^{n,(i)}|\boldsymbol{x}_{t-1}^{n,(i)}, \boldsymbol{x}_{t-1}^{l,(i)}, \boldsymbol{P}^{(i)}, \boldsymbol{y}_{1:t})},\quad i = 1,\ldots,N. \tag{24}$$

  重み $w_t^{(i)},\ i = 1,\ldots,N$ を和が 1 になるように正規化する．
  (3) 線形な部分に対するカルマンフィルタの予測ステップを実行する．

$$\boldsymbol{z}_t = \boldsymbol{x}_t^n - f_{t-1}^n, \tag{25}$$
$$\boldsymbol{N}_{t-1} = \boldsymbol{A}_{t-1}^n \boldsymbol{P}_{t-1} (\boldsymbol{A}_{t-1}^n)^{\mathrm{T}} + \boldsymbol{Q}_{t-1}^n, \tag{26}$$
$$\boldsymbol{L}_{t-1} = \boldsymbol{A}_{t-1}^l \boldsymbol{P}_{t-1} (\boldsymbol{A}_{t-1}^n)^{\mathrm{T}} \boldsymbol{N}_{t-1}^{-1}, \tag{27}$$
$$\boldsymbol{x}_t^{l,-} = \boldsymbol{A}_{t-1}^l \boldsymbol{x}_{t-1}^l + \boldsymbol{L}_{t-1}(\boldsymbol{z}_t - \boldsymbol{A}_{t-1}^n \boldsymbol{x}_{t-1}^l), \tag{28}$$
$$\boldsymbol{P}_t^- = \boldsymbol{A}_{t-1}^l \boldsymbol{P}_{t-1} (\boldsymbol{A}_{t-1}^l)^{\mathrm{T}} + \boldsymbol{Q}_{t-1}^l - \boldsymbol{L}_{t-1} \boldsymbol{N}_{t-1} \boldsymbol{L}_{t-1}^{\mathrm{T}}. \tag{29}$$

ここで，$\boldsymbol{A}_{t-1}^n, \boldsymbol{A}_{t-1}^l$, f_{t-1}^n の $\boldsymbol{x}_{t-1}^n$ に対する依存性と，サンプル添字 i は表記上省略している．

(4) 線形な部分に対するカルマンフィルタの更新ステップを実行する．

$$\boldsymbol{v}_t = \boldsymbol{y}_t - h_t - \boldsymbol{C}_t \boldsymbol{x}_t^{l,-}, \tag{30}$$
$$\boldsymbol{S}_t = \boldsymbol{C}_t \boldsymbol{P}_t^- \boldsymbol{C}_t^{\mathrm{T}} + \boldsymbol{R}_t, \tag{31}$$
$$\boldsymbol{K}_t = \boldsymbol{P}_t^- \boldsymbol{C}_t^{\mathrm{T}} \boldsymbol{S}_t^{-1}, \tag{32}$$
$$\boldsymbol{x}_t = \boldsymbol{x}_t^{l,-} + \boldsymbol{K}_t \boldsymbol{v}_t, \tag{33}$$
$$\boldsymbol{P}_t = \boldsymbol{P}_t^- - \boldsymbol{K}_t \boldsymbol{C}_t \boldsymbol{P}_t^-. \tag{34}$$

ここで，$\boldsymbol{C}_t$,h_t の $\boldsymbol{x}_t^n$ に対する依存性と，サンプル添字 i は表記上省略している．

(5) $\boldsymbol{x}_t^n$ と $\boldsymbol{x}_t^l$ の推定値を，サンプルの重み付き平均で与える．

$$\hat{\boldsymbol{x}}_t^n := \sum_{i=1}^N w_t^{(i)} \boldsymbol{x}_t^{n,(i)}, \tag{35}$$
$$\hat{\boldsymbol{x}}_t^l := \sum_{i=1}^N w_t^{(i)} \boldsymbol{x}_t^{l,(i)}. \tag{36}$$

(6) リサンプリングを行う．

(a) 重み $w_t^{(i)}$, $i = 1, 2, \ldots, N$ を，添字集合 $\{(\boldsymbol{x}_t^{n,(i)}, \boldsymbol{x}_t^{l,(i)}, \boldsymbol{P}^{(i)}) : i = 1, \ldots, N\}$ からサンプル添字 i を得る確率と解釈し，N 個のサンプルを生成する．古いサンプル集合を新しいものに置き換える．

(b) 重みを均等に $w_t^{(i)} = 1/N, i = 1, \ldots, N$ とする．

6. 手の姿勢と速度に対する Rao-Blackwellized 粒子フィルタ

本節で，第 4 節と第 5 節を統合し，手の姿勢推定特有の定式化を行う．種々の関数と行列が，それぞれ

$$f_t^n(\boldsymbol{x}_t^n) = \boldsymbol{x}_t^n, \tag{37}$$
$$h_t^n(\boldsymbol{x}_t^n) = \boldsymbol{d}(\boldsymbol{x}_t^n), \tag{38}$$

および

$$\boldsymbol{A}_t^n(\boldsymbol{x}_t^n) = \boldsymbol{A}^n = \Delta T_t \boldsymbol{I}, \tag{39}$$
$$\boldsymbol{A}_t^l(\boldsymbol{x}_t^n) = \boldsymbol{A}^l = \boldsymbol{I}, \tag{40}$$
$$\boldsymbol{C}_t(\boldsymbol{x}_t^n) = \boldsymbol{O}, \tag{41}$$
$$\boldsymbol{Q}_t^n = \Delta T_t \boldsymbol{Q}^n, \tag{42}$$
$$\boldsymbol{Q}_t^l = \Delta T_t \boldsymbol{Q}^l, \tag{43}$$

で与えられるとする．ここに，$\boldsymbol{Q}^n$ と $\boldsymbol{Q}^l$ は対角行列であるとし，ΔT_t は時刻 t と $t+1$ の間の時間の長さであるとする．加えて，$\boldsymbol{e}_t$ は時刻 t における，第 4 節で定義したノイズ $\boldsymbol{e}$ であるとする．これはガウス分布に従わないが，式 (22) において $\boldsymbol{e}$ がガウス分布に従うという仮定に問題を生じない．というのも，式 (32) における条件 $\boldsymbol{C}_t(\boldsymbol{x}_t^n) = \boldsymbol{O}$ が，式 (31) の $\boldsymbol{R}_t$ の $\boldsymbol{x}_t$ および $\boldsymbol{P}_t$ に対する影響を打ち消すからである．さらに，$\boldsymbol{P}_t$ はもはや $\boldsymbol{x}_t^n$ に依存しないため，$\boldsymbol{P}_t$ のサンプルを生成してカルマンフィルタを N 回適用する必要はない．

これらの仮定は以下のように要約される．

$$\begin{aligned} \boldsymbol{x}_t^n &= \boldsymbol{x}_{t-1}^n + \Delta T_{t-1} \boldsymbol{x}_{t-1}^l + \boldsymbol{w}_{t-1}^n, \\ \boldsymbol{x}_t^l &= \boldsymbol{x}_{t-1}^l + \boldsymbol{w}_{t-1}^l, \\ \boldsymbol{y}_t^n &= \boldsymbol{d}(\boldsymbol{x}_t^n) + \boldsymbol{e}_t. \end{aligned} \tag{44}$$

ここに，

$$\boldsymbol{w}_t^n \sim \mathcal{N}(\boldsymbol{0}, \Delta T_t \boldsymbol{Q}^n), \tag{45}$$
$$\boldsymbol{w}_t^l \sim \mathcal{N}(\boldsymbol{0}, \Delta T_t \boldsymbol{Q}^l), \tag{46}$$
$$p(\boldsymbol{y}|\boldsymbol{x}) \propto \exp\left[-\frac{\{D(\boldsymbol{d}(\boldsymbol{x}), \boldsymbol{y})\}^2}{2\sigma^2} \right], \tag{47}$$
$$D(\boldsymbol{d}(\boldsymbol{x}), \boldsymbol{y}) = \frac{1}{|R|} \sum_{i \in R} \min\{|d_i - y_i'|, c_{\max}\}. \tag{48}$$

これらに加えて，提案分布を

$$\begin{aligned} &\pi(\boldsymbol{x}_t^n | \boldsymbol{x}_{t-1}^{n,(i)}, \boldsymbol{x}_{t-1}^{l,(i)}, \boldsymbol{P}, \boldsymbol{y}_{1:t}) \\ &= \pi(\boldsymbol{x}_t^n | \boldsymbol{x}_{t-1}^{n,(i)}, \boldsymbol{x}_{t-1}^{l,(i)}) \qquad (49) \\ &= p(\boldsymbol{x}_t^n | \boldsymbol{x}_{t-1}^{n,(i)}, \boldsymbol{x}_{t-1}^{l,(i)}) \qquad (50) \\ &= \mathcal{N}(\boldsymbol{x}_{t-1}^{n,(i)} + \Delta T_{t-1} \boldsymbol{x}_{t-1}^{l,(i)}, \Delta T_{t-1} \boldsymbol{Q}^n) \\ &\quad , \ i = 1, \ldots, N \qquad (51) \end{aligned}$$

と定義する．

このモデルに対する，N 個の粒子を用いたアルゴリズムは以下のように計算される．

- 非線形な部分を初期化する．N 個のサンプル $\boldsymbol{x}_0^{n,(i)}, i = 1, \ldots, N$ を初期分布 $p(\boldsymbol{x}_0^n)$ から生成し，重みを $w_0^{(i)} = 1/N$, $i = 1, \ldots, N$ とする．
- 線形な部分を初期化する．$\boldsymbol{P}_0, \boldsymbol{x}_0^l$ の初期値を与え，$\boldsymbol{x}_0^{l,(i)} = \boldsymbol{x}_0^l$, $i = 1, \ldots, N$ とする．
- 各時刻 $t = 1, 2, \ldots$ において以下を実行する．

(1) 非線形な部分のサンプル $\boldsymbol{x}_t^{n,(i)}$, $i = 1, \ldots, N$, を，提案分布

$$\boldsymbol{x}_t^{n,(i)} \sim \pi(\boldsymbol{x}_t^n | \boldsymbol{x}_{t-1}^{n,(i)}, \boldsymbol{x}_{t-1}^{l,(i)}),\ i = 1, \ldots, N. \tag{52}$$

から生成する．

(2) 重みを更新する．

$$\begin{aligned} & w_t^{(i)} \\ & \propto w_{t-1}^{(i)} \frac{p(\boldsymbol{y}_t|\boldsymbol{x}_t^{n,(i)})p(\boldsymbol{x}_t^{n,(i)}|\boldsymbol{x}_{t-1}^{n,(i)},\boldsymbol{x}_{t-1}^{l,(i)})}{\pi(\boldsymbol{x}_t^n|\boldsymbol{x}_{t-1}^{n,(i)},\boldsymbol{x}_{t-1}^{l,(i)})}, \quad (53) \\ & = w_{t-1}^{(i)} p(\boldsymbol{y}_t|\boldsymbol{x}_t^{n,(i)}),\ i=1,\ldots,N. \quad (54) \end{aligned}$$

重みの和が 1 になるよう正規化する．

(3) 線形な部分の行列に対してカルマンフィルタの予測ステップを実行する．

$$\begin{aligned} \boldsymbol{N}_{t-1} &= (\Delta T_{t-1})^2 \boldsymbol{P}_{t-1} + \Delta T_{t-1}\boldsymbol{Q}^n, \quad (55) \\ \boldsymbol{L}_{t-1} &= \boldsymbol{P}_{t-1}\boldsymbol{N}_{t-1}^{-1}, \quad (56) \\ \boldsymbol{P}_t^- &= \boldsymbol{P}_{t-1} + \Delta T_{t-1}\boldsymbol{Q}^l \\ &\quad -\boldsymbol{L}_{t-1}\boldsymbol{N}_{t-1}\boldsymbol{L}_{t-1}^{\mathrm{T}}. \quad (57) \end{aligned}$$

(4) 線形な部分のベクトルに対してカルマンフィルタの予測ステップを実行する．

$$\begin{aligned} \boldsymbol{z}_t^{(i)} &= \boldsymbol{x}_t^{n,(i)} - \boldsymbol{x}_{t-1}^{n,(i)}, \quad (58) \\ \boldsymbol{x}_t^{l,-,(i)} &= \boldsymbol{x}_{t-1}^{l,(i)} + \boldsymbol{L}_{t-1}(\boldsymbol{z}_t^{(i)} - \Delta T_{t-1}\boldsymbol{x}_{t-1}^{l,(i)}), \\ & \quad i=1,\ldots,N. \quad (59) \end{aligned}$$

(5) 線形な部分に対してカルマンフィルタの更新ステップを実行する．

$$\begin{aligned} \boldsymbol{x}_t^{l,(i)} &= \boldsymbol{x}_t^{l,-,(i)},\ i=1,\ldots,N \quad (60) \\ \boldsymbol{P}_t &= \boldsymbol{P}_t^-. \quad (61) \end{aligned}$$

(6) $\boldsymbol{x}_t^n, \boldsymbol{x}_t^l$ の推定値を，サンプルの重み平均で与える．

$$\hat{\boldsymbol{x}}_t^n := \sum_{i=1}^N w_t^{(i)} \boldsymbol{x}_t^{n,(i)}, \quad (62)$$

$$\hat{\boldsymbol{x}}_t^l := \sum_{i=1}^N w_t^{(i)} \boldsymbol{x}_t^{l,(i)}. \quad (63)$$

(7) リサンプリングを行う．

(a) 重み $w_t^{(i)},\ i=1,2,\ldots,N$ を，添字集合 $\{(\boldsymbol{x}_t^{n,(i)},\boldsymbol{x}_t^{l,(i)}) : i=1,\ldots,N\}$ からサンプル添字 i を得る確率と解釈し，サンプルを生成する．古いサンプル集合を新しいものに置き換える．

(b) 重みを $w_t^{(i)} = 1/N,\ i=1,\ldots,N$ とする．

7. 実験結果

7.1 実験の概要

提案手法のコスト関数は，グラフィックスカード上の並列計算システムである CUDA を用いて実装した．加えて，メインメモリ・グラフィックスメモリ間のデータ転送を減らすために，OpenGL でレンダリングした深度画像はグラフィックスカード上で直接 CUDA に渡されるようにした．GPU は NVIDIA GeForce 980 Ti を用いた．

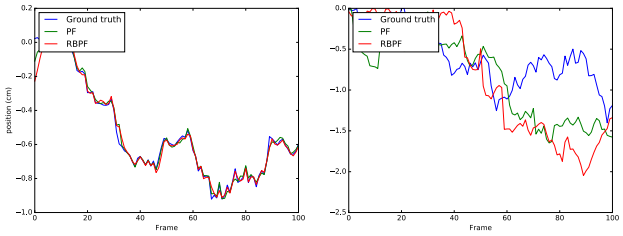

図 6 人工データ A に対する粒子フィルタと Rao-Blackwellized 粒子フィルタの推定結果．(左) 手首の位置の y 座標．(右) 中指の第 2 関節の x 軸についての角度．

手の姿勢追跡手法の性能評価は 2 種類の方法で行うことができる．1 つは，用意した手の姿勢ベクトルの列から生成した深度画像をセンサから得られたものと見なし，推定した結果の姿勢を元の姿勢と比較するという方法である．追跡対象の姿勢の列として，ランダムに姿勢が変化するデータ A と，姿勢に加えて速度もランダムに変化するデータ B を用意した．これらの結果は第 7.2 節で示す．

もう 1 つの方法は，実際に深度センサから得られた実データに対して手法を実行するというものである．追跡制度の評価を厳密に行うことはできないが，より直感的に性能を知ることができる．この結果は第 7.3 節に示す．

両方の実験において，粒子フィルタと Rao-Blackwellized 粒子フィルタを比較した．コスト関数 (4) のパラメータは以下のように与えた．

$$\sigma = 0.02, \quad (64)$$

$$c_{\max} = 100 \text{ mm}. \quad (65)$$

深度画像の解像度は $(512, 424)$ ピクセルであり，深度の比較に用いるピクセル領域 R の大きさは $(128, 128)$ とした．

7.2 人工データ

7.2.1 人工データ A

1 つ目の追跡対象である手の姿勢の列 $\boldsymbol{x}_t^n, t=0,1,\ldots$ は，ガウス分布

$$p(\boldsymbol{x}_t^n|\boldsymbol{x}_{t-1}^n) = \mathcal{N}(\boldsymbol{x}_{t-1}^n, \Delta T_t \boldsymbol{Q}^n),\ t=1,2,\ldots \quad (66)$$

から生成した．これは，線形な要素がないことを除いて式 (44) と同一である．この追跡対象は速度の情報を持たない．

粒子フィルタ (PF) と Rao-Blackwellized 粒子フィルタ (RBPF) による推定結果のうち，一部のベクトル要素の時間変化を図 6 に示す．この推定におけるパラメータは，$\Delta T_t = 1/30,\ t=0,1,\ldots$ および $N=512$ とした．

図 6 左から，速度の情報を持たない小さな変動に対して，一部の部位で両手法ともほとんど正しい推定値が得られていることが分かる．一方で，図 6 右から，別の部位においては両手法とも上手く追跡できていない．さらに，手法間

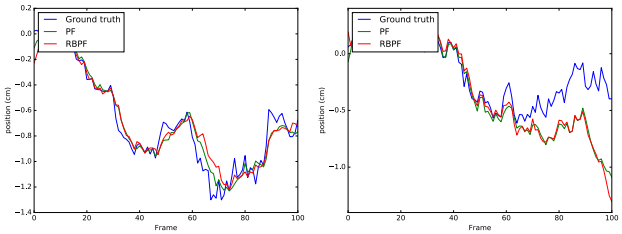

図 7 人工データ B に対する粒子フィルタと Rao-Blackwellized 粒子フィルタの推定結果．(左) 手首の位置の y 座標．(右) 手首の位置の x 座標．

の違いに大きな特徴が見られない．

7.2.2 人工データ B

2 つ目の追跡対象である手の姿勢の列 $\boldsymbol{x}_t^n, t = 0, 1, \ldots$ は，ガウス分布

$$\begin{aligned} p(\boldsymbol{x}_t^n|\boldsymbol{x}_{t-1}^n, \boldsymbol{x}_{t-1}^l) &= \mathcal{N}(\boldsymbol{x}_{t-1}^n + \Delta T_{t-1}\boldsymbol{x}_{t-1}^l, \Delta T_t \boldsymbol{Q}^n), \\ p(\boldsymbol{x}_t^l|\boldsymbol{x}_{t-1}^l) &= \mathcal{N}(\boldsymbol{x}_{t-1}^l, \Delta T_t \boldsymbol{Q}^l), \\ & t = 1, 2, \ldots \end{aligned} \tag{67}$$

から生成した．これは，式 (44) と同一である．この追跡対象は速度の情報を持つ．

粒子フィルタ (PF) と Rao-Blackwellized 粒子フィルタ (RBPF) による推定結果のうち，一部のベクトル要素の時間変化を図 7 に示す．この推定におけるパラメータは，データ A に対する実験と同じものを使用した．

実験結果は人工データ A に対するものとほとんど変わらなかった．図 7 左から，速度の情報を持たない小さな変動に対して，一部の部位で両手法ともほとんど正しい推定値が得られていることが分かる．一方で，図 7 右から，別の部位においては両手法とも上手く追跡できていない．さらに，手法間の違いに大きな特徴は見られなかった．

7.3 Kinect の実データ

深度センサ Kinect から得た深度画像の実データに対して実験を行った．時刻の間隔 ΔT_t, $t = 0, 1, \ldots$ はおよそ 1/30 秒であるが，センサの状態や追跡アルゴリズムの性能によって変動し，非常に長くなる場合がある．

粒子数 $N = 512$ の粒子フィルタおよび Rao-Blackwellized 粒子フィルタで手の姿勢追跡を行っている様子を，図 9 に示す．大域的な手の動作と，指を折り曲げる動作の 2 種類に対して実験を行った．姿勢の初期分布は，センサに対して画面中央で手のひらを向けている状態をガウス分布の平均値とした．姿勢の変化速度の初期分布は平均 0 のガウス分布で与えた．

いずれの動作においても，変化速度が小さければ両手法で追跡ができたが，精度は高くなかった．また，手法間で大きな差異は見られなかった．

両手法とも精度が高くなかった原因として，前処理が不充分である可能性が考えられる．[7] ではセンサから得られた深度画像において手に該当するピクセル領域を事前に判別し，深度情報とは別に，両画像のピクセル領域の一致度合いもコスト関数に組み込んでいる．それに対し，提案手法のコスト関数では，レンダリングした画像の手の領域に関する深度の平均を用いているだけで，領域の一致度合いを考慮していない．その結果，粒子フィルタの種類に依らず精度が不充分となり，差が現れなかったものと考えられる．

図 8 実データに対する粒子フィルタの推定結果．上段はセンサから得た深度画像を白黒表現したものであり，下段は推定結果のレンダリング画像．(左) 大域的な手の動作の追跡結果．(右) 指を折り曲げる動作の追跡結果．

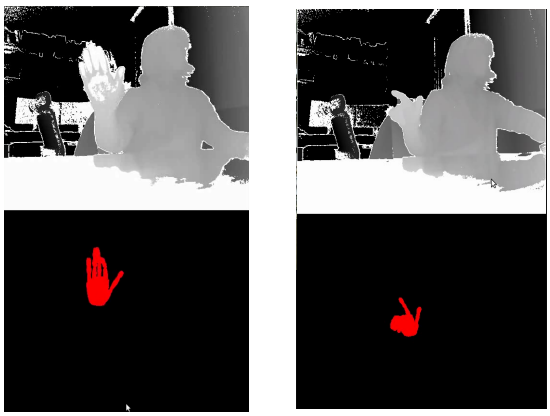

図 9 実データに対する Rao-Blackwellized 粒子フィルタの推定結果．上段はセンサから得た深度画像を白黒表現したものであり，下段は推定結果のレンダリング画像．(左) 大域的な手の動作の追跡結果．(右) 指を折り曲げる動作の追跡結果．

7.4 実行時間

粒子フィルタおよび Rao-Blackwellized 粒子フィルタの，1 フレームあたりの計算時間の平均を図 10 に示す．粒子数以外の実験設定は第 7.2.2 節と同じとした．

結果から，Rao-Blackwellized 粒子フィルタでは状態変数の次元数が 2 倍であるにも関わらず，両手法の実行時間にほとんど差がないことが分かる．このことから，Rao-Blackwellized 粒子フィルタは必要となる計算資源を増やすことなく，より複雑なモデルを扱うことができると言え

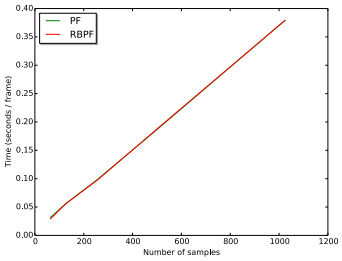

図 10 粒子フィルタと Rao-Blackwellized 粒子フィルタの，1 フレームあたりの計算時間の比較．

る．しかしながら，実装の最適化が不充分であるため計算が遅く，センサがフレームを取得する時間間隔よりも長い計算時間を要し，実時間での利用には限界がある．

8. 結論

本稿では単一の深度センサ Kinect を用いて人間の手の姿勢を追跡する新たな手法を提案した．提案手法では，状態変数が手の姿勢だけでなくその時間微分を含んだ状態空間モデルに対し，Rao-Blackwellized 粒子フィルタを適用した．実験結果から，追跡精度の向上を支持する根拠は得られなかったが，追加の計算コストがほとんど無視できるほど小さいものであることが分かった．追跡精度が向上しなかった原因として，コスト関数の設計が不適切である可能性が挙げられる．実用に足る精度を得るには更なる検討を要するが，計算時間削減において有効性を示した．

参考文献

[1] Bishop, C. M.: *Pattern Recognition and Machine Learning*, Springer-Verlag New York, Inc., Secaucus, NJ, USA (2006).
[2] Brown, J. A. and Capson, D. W.: A framework for 3D model-based visual tracking using a GPU-accelerated particle filter, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 18, No. 1, pp. 68–80 (2012).
[3] Dipietro, L., Sabatini, A. M. and Dario, P.: *A survey of glove-based systems and their applications*, Vol. 38, No. 4, pp. 461–482, IEEE (2008).
[4] Keskin, C., Kıraç, F., Kara, Y. E. and Akarun, L.: Real time hand pose estimation using depth sensors, *Consumer Depth Cameras for Computer Vision*, Springer, pp. 119–137 (2013).
[5] Lin, J., Wu, Y. and Huang, T. S.: Modeling the constraints of human hand motion, *Workshop on Human Motion, 2000. Proceedings.*, IEEE, pp. 121–126 (2000).
[6] Makris, A., Kyriazis, N. and Argyros, A.: Hierarchical particle filtering for 3D hand tracking, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pp. 8–17 (2015).
[7] Oikonomidis, I., Kyriazis, N. and Argyros, A. A.: Efficient model-based 3D tracking of hand articulations using Kinect., Vol. 1, No. 2, p. 3 (2011).
[8] Qian, C., Sun, X., Wei, Y., Tang, X. and Sun, J.: Real-time and robust hand tracking from depth, *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1106–1113 (2014).
[9] Särkkä, S.: *Bayesian Filtering and Smoothing*, Vol. 3, Cambridge University Press (2013).
[10] Schön, T., Gustafsson, F. and Nordlund, P.-J.: Marginalized particle filters for mixed linear/nonlinear state-space models, *IEEE Transactions on Signal Processing*, Vol. 53, No. 7, pp. 2279–2289 (2005).
[11] Sudderth, E. B., Mandel, M. I., Freeman, W. T. and Willsky, A. S.: Visual hand tracking using nonparametric belief propagation, *Proceedings of Conference on Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04.*, IEEE, pp. 189–189 (2004).
[12] Wang, R. Y. and Popović, J.: Real-time hand-tracking with a color glove, *ACM Transactions on Graphics (TOG)*, Vol. 28, No. 3, ACM (2009).

付　　録

A.1 手の姿勢ベクトル

実際の人間の手の姿勢は非常に複雑であるが，本稿では [5] にならい，22 次元の実ベクトル $\boldsymbol{p} \in \mathbb{R}^{22}$ でパラメータ化する．ベクトル要素は大域的な位置 (3 パラメータ)，大域的な回転 (4 パラメータ)，および各指関節の角度 (各指に対して 3 パラメータ) を表現する．簡単のため，手のスケールは考えない．指関節における角度は，予め定義した軸に対するものとする．各指の第 1 関節の角度は，対応する第 2 関節の角度の半分であるとして自動的に決定されるものとする．

深度センサは右手座標系の原点に設置され z 軸方向を向いており，手の平はセンサの方向を向いているものとする．このとき，手の姿勢ベクトル $\boldsymbol{p} = [p_1, \ldots, p_{22}]^{\mathrm{T}}$ を以下のように構成する．まず，(p_1, p_2, p_3) を手首の位置，(p_4, p_5, p_6, p_7) を手首の回転四元数とする．残りの要素を指関節の角度に割り当てる．p_8 を親指第 3 関節の x 軸，p_9, p_{10} をそれぞれ第 2 関節の x, z 軸とする．p_{11}, p_{12} を人差指第 3 関節のそれぞれ x, z 軸とし，p_{13} を第 2 関節の x 軸とする．他の要素は中指，薬指，小指の順に，人差指と同様にして割り当てる．