複数台の Kinect を用いたモーションキャプチャの実現

Development of Motion Capture System using Multiple Kinects

宮武順平 [1*] 大坪正和 [1] 吉田香 [1]
Miyatake Jumpei[1] Ohtubo Masakazu[1] Yoshida Kaori [1]

[1] 九州工業大学 大学院 生命体工学研究科
[1] Graduate School of Life Science and Systems Enginnering, Kyushu Institute of technology

Abstract: Nowadays, user-friendly motion gesture system has been expected according to humanoid robot spread. On the other hand, motion sensing system like Kinect is easy to get and enables to develop motion capture system. However, conventional motion capture system has the following weak points, (i) poor accuracy except front direction, (ii) unable to capture hidden part, and (iii) problem against occlusion. In this paper, we propose the motion capture system using multiple Kinects and show the experimental results.

1 はじめに

近年，ロボットが様々な場所で普及し始めている．総務省の ICT が拓く未来社会 [1] では，ロボットが人間のパートナとしてのニーズに関する調査があり，様々な分野で利用したいという希望があることが分かっている．その中でも，Pepper[2] をはじめとする人型ロボットを見かけることが多くなっている．これらのロボットの普及によって，人型ロボットの動作を作り，動かすというニーズが増加傾向にある．ロボットの動作を作成する方法は，古典的な方法として，各モータに直接数値を入力する方法や，実際にロボットの腕を操作して，その動きを記憶する方法など様々な方法がある．これらの方法は，専門家や技術者の試行錯誤などによって動作の作成が行われるため，時間的コストが高いことが問題として挙げられている．この課題を解決する一つの手法として，モーションキャプチャの利用が挙げられる．モーションキャプチャは，人物の動作を獲得することができるデバイスである．中澤らの研究 [3] ではモーションキャプチャを利用して実際に人の動きをとりロボットにまねさせることができると報告されている．そこで本研究ではモーションキャプチャを利用することでロボットを動作させるためのシステムを提案する．モーションキャプチャは Microsoft 社の Kinect[4] を使用する．Kinect はカラーセンサと奥行きを獲得できる深度センサなどのセンサを持ち，一般的なモーションキャプチャとは違い，マーカを必要とせずにモーションキャプチャが可能 [5] という利点がある．モーションキャプチャはカメラ型デバイスであるため，単一のデバイスだけではオクルージョン (不可視部位) の発生が多々ある．オクルージョンを回避するために複数台のデバイスを用いることでオクルージョンを軽減することが可能である．本論文では，複数台の Kinect を用いてオクルージョンに頑健なモーションキャプチャを実現するためのシステムを構築し，モーションデータの精度を向上させるアルゴリズムを提案する．

*連絡先： 九州工業大学 大学院 生命体工学研究科
〒 808-0196 北九州市若松区ひびきの 2 番 4 号
E-mail: miyatake-jumpei@edu.brain.kyutech.ac.jp

2 関連研究

本研究の関連研究は，Kinect の特性デバイス特徴と複数台の Kinect の情報統合に分類することができる．

2.1 Kinect のデバイス特徴

Kinect のデバイス特徴については，Zennaro らの研究 [6] で詳しく述べられている．Kinect には，初期型の Kinect ver1(KinectV1) と後期型である Kinect ver2 (KinectV2) がある．KinectV1 は Light Coding(LC) 方式によって奥行きを獲得している．LC 方式は，パターンを照射し，そのパターンの見え方によって奥行きを獲得する方法である．そのため，複数台の KinectV1 を使用する場合，パターンに被りが生じると，精度が損なわれることが，Butler らの研究 [7] によって報告されている．KinectV2 では，Time of Flight (ToF) 方式を利用しており，プロジェクタから照射された赤外線が反射して，カメラに帰ってくるまでの時間から奥行きを獲得する方法である．赤外線の光線の反射時間を利用するため，複数台の使用の際に干渉の影響を受けに

くい．これらの理由から，本研究では KinectV2 を使用する．

2.2 複数台の Kinect の情報統合

Kinect をロボットの動作生成に使用した研究には，占崎らの研究 [8] などがあり，Kinect から獲得された情報を使用して人型ロボットを操作することが可能である．複数台の Kinect を用いた研究として，平藤らの研究 [9] では複数台の KinectV1 を用いてオクルージョンを減らし，3D モデルを生成する手法が提案されている．吉本誠也らの研究 [10] では，複数人の骨格の動きの情報を統合する方法が提案されており，運動に対するフレーム間の調整法や骨格情報の統合方法が提案されている．実際には，両肩を結んだ線と，首・脊椎中央を結んだ線を利用してアフィン変換によって，統合しているが，この方法では手や足などの末端の関節に対し座標のずれが生じる．正確なモーションキャプチャを実現するには，末端のずれを軽減する必要がある．Kitsikidis らの研究 [11] では，複数台の Kinect を使用してダンスのモーションキャプチャを実現している．具体的には，複数台の Kinect の情報に対してモデルを使用して情報統合を行うことでオクルージョンを減らし，トラッキング精度を向上させている．本研究では，モーションキャプチャのデータをロボットの動作に適応させるため，モデルを使用せず，複数台の Kinect の情報を各関節レベルで統合することを目標とする．つまり関節レベルで精確に動作をロボットに伝えることのできる情報統合を目指す．

3 システム概要

複数台の KinectV2 を使用する場合，単一の PC で複数台を扱うことができない．そこで，図 1 のようなシステムを構築することで KinectV2 を複数台扱えるようにする．各 Kinect に PC を接続し，Kinect から獲得した骨格情報をサーバに送信することで複数台の KinectV2 を可能にする．また，PC と KinectV2 間は USB3.0 で接続し，サーバと PC 間はイーサネットを利用して通信する．信頼性が高くはないがリアルタイム性を求める通信に使用されるプロトコルである UDP を用いて通信を行う．Kinect から得られるデータについては，Microsoft 社から公開されている Kinect SDK を利用して獲得する．本提案システムでは，各 Kinect の情報をサーバで受け取り，その情報をサーバで処理する．サーバでは，それぞれの Kinect から送信された情報のキャリブレーションと骨格情報の情報統合を行うことによって複数台の Kinect を用いたモーションキャプチャを実現する．

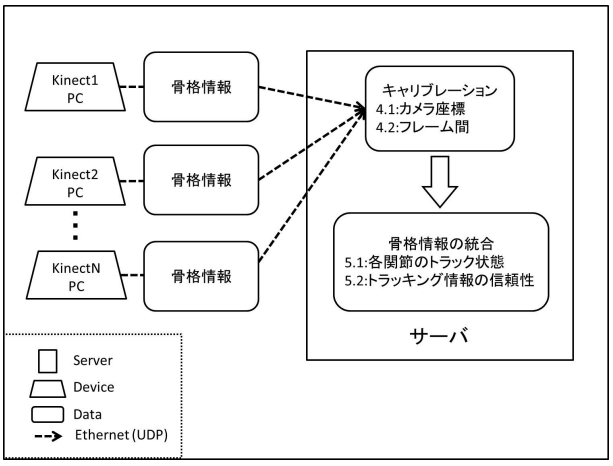

図 1: システム概要

4 キャリブレーション

Kinect から取得した骨格情報を統合ために，各 Kinect の設置位置によって生じるずれを補正する必要がある．この位置によるずれについてはカメラの内部のゆがみとカメラの設置位置の違いから生じる．カメラの内部パラメータの補正では，Zhang らの手法 [12] によってキャリブレーションが可能である．カメラの外部パラメータの取得には，ワールド座標の中心にマーカを置き，そこから取得する 6 点の座標を用いて取得する．これらのパラメータの値が既知であることを前提として，計算法を 4.1 節で説明する．また，システムの特性上，各 Kinect の骨格情報はイーサネットを経由するため，フレーム間で微小な遅れが生じる可能性がある．このフレーム間のずれを補正する方法を 4.2 節で示す．

4.1 座標の位置合わせ

複数台の Kinect の位置を合わせる方法として，一般的なカメラ座標変換 [13] を用いる．Kinect が N 個あるとき，Kinect の 1 つを主として，Kinect のワールド座標に対する姿勢と座標を $\mathbf{K}_{main} = \mathbf{K}_0$ とし，N 個の Kinect のワールド座標に対する姿勢と座標を $\mathbf{K}_n = \{\mathbf{K}_0, \cdots, \mathbf{K}_{N-1}\}$ とする．$\mathbf{K}_n$ を主 Kinect の座標系に変換する変換する変数をとすると，

$$\mathbf{K}_0 = \mathbf{A}_n \mathbf{K}_n \tag{1}$$

変換変数 $\mathbf{A}_n$ は，(2) 式で求めることができる．

$$\mathbf{A}_n = \mathbf{K}_0 \mathbf{K}_n^{-1} \tag{2}$$

変換座標 $\mathbf{A}_n$ は各 Kinect に対して計算する必要がある．n 番目の Kinect から取得したある時刻 t の骨格情報を $\mathbf{S}_n^t = \{\mathbf{j}_1^t, .., \mathbf{j}_I^t\}$ とし，$\mathbf{S}_n^t$ は式 (2) で得られた

変換によってすべて $\mathbf{K}_{main}$ の座標系で表現される．$\mathbf{j}_i^t$ は，ある時刻 t の i 番目の関節の座標を持つことから，$\mathbf{j}_i^t = (x_i^t, y_i^t, z_i^t)$ となり，X 座標，Y 座標，Z 座標の 3 次元の座標を持つ．ただし $i = 1, \cdots, I$ であり，I は，Kinect が 25 の関節の情報を持つことから $I = 25$ とする．

4.2 フレーム間の調整

骨格情報は時刻によって変化するため，イーサネットによる遅延やデバイス間のハードウェア的遅延を考慮する必要がある．ハードウェア的遅延については，ごく微小であり，Kinect の 30 fps という性能を考慮しても誤差の範囲であるため，ここでは無視できる遅延であると考える．フレーム間の修正では，特定のモーションを行い修正行う．フレーム間の修正は，主 Kinect とその他の Kinect 間に対して，前後の数フレームを調べ，各 Kinect が獲得した骨格情報のズレが最小となるようなフレームを求める．ある時刻 t のときの主 Kinect が獲得したフレームを f_0^t とし，ある時刻 t のときの n 番目の Kinect のフレームを f_n^t とすると，

$$f_n^t = \arg\min_f(|\mathbf{S}_n^f - \mathbf{S}_0^{f_0^t}|) \tag{3}$$

ただし，$|f_n^t - f_0^t| \leq F_{max}$ とする．F_{max} は，キャリブレーションのデータから経験的に求めることとする．式 (3) の条件を利用することでイーサネットによる遅延を考慮し，情報統合する際のデータを選択する．

5 骨格情報の統合

キャリブレーションが完了した各 Kinect の情報に対して，骨格情報を統合する．情報統合後の骨格情報を $\mathbf{S}^{t*} = \{\mathbf{j}_1^{t*}, .., \mathbf{j}_I^{t*}\}$ とする．統合後の各関節である $\mathbf{j}_i^{t*}$ は式 (4) のようになる．

$$\mathbf{j}_i^{t*} = \sum_{n=0}^{N-1} {\alpha_{n,i}^t}^* \delta_{n,i}^t \mathbf{j}_{n,i}^t \tag{4}$$

ここで ${\alpha_{n,i}^t}^*$,$\delta_{n,i}^t$ は統合の際の調整パラメータである．それぞれのパラメータについては，$\delta_{n,i}^t$ は 5.1 節でを説明し，${\alpha_{n,i}^t}^*$ は 5.2 節で説明する．

5.1 各関節のトラック状態

Kinect SDK から得られる情報として骨格情報の各関節のトラッキング状態を取得することができる．具体的には，関節ごとに true , false が与えられており，トラッキングできている (Tracked) 状態を true，トラッキングできていない (Lost) 状態を false として獲得できる．この値を $\delta_{n,i}^t$ に当てはめる．true の場合を 1,false 場合を 0 として代入する．

$$\delta_{n,i}^t = \begin{cases} 0 & \text{(Lost)} \\ 1 & \text{(Tracked)} \end{cases} \tag{5}$$

$\delta_{n,i}^t$ は，関節ごとに Tracked 状態と Lost 状態を判別することができるパラメータである．

5.2 トラッキング情報の信頼性

それぞれの Kinect から得られた骨格情報から，ある時刻 t の $\mathbf{j}_{n,i}^t$ と $\delta_{n,i}^t$ を獲得する．トラッキングできている関節に対して信頼度を定義することで，より正確性が高い関節情報を選択するためのアルゴリズムを提案する．1 時刻前のフレームと現在のフレームの各関節の移動距離をユークリッド距離で表現し，ユークリッド距離を用いて信頼度を定義する．1 フレーム間におけるユークリッド距離の最大値をワールド座標系の X,Y,Z 軸を動作するモーションデータで構成されたテストデータから求める．式 (6) に 1 フレーム間のユークリッド距離を示す．$\mathbf{j}_{n,i}^t$ は n 番目の Kinect のある時刻 t における関節 i のデータ, $\mathbf{j}_{n,i}^{t-1}$ は n 番目の Kinect のある時刻 $t-1$ における関節 i のデータである．n 番目の Kinect のある時刻 t における 1 フレーム間の移動距離を $d_{n,i}^t$ とすると，

$$d_{n,i}^t = |\mathbf{J}_{n,i}^t - \mathbf{J}_{n,i}^{t-1}| \tag{6}$$

1 フレーム間の最大移動距離 $d_{n,i}$ は，

$$dmax_{n,i} = \max_t(d_{n,i}^t) \tag{7}$$

となる．(7) 式により各関節の最大移動距離を求める．信頼度 $\alpha_{n,i}^t$ には，最大移動距離を基準とした移動距離に応じてペナルティがかるような関数を用いる．また，ペナルティは関節ごとに設定する．式 (8) にペナルティ $r_{n,i}^t$ に示す．

$$r_{n,i}^t = \frac{\exp(dmax_{n,i} - d_{n,i}^t)}{\exp(dmax_{n,i})} \tag{8}$$

式 (8) の $r_{n,i}^t$ は総和が 1 という条件を満たさないため，式 (9) により $\sum_{n=0}^{N-1} \alpha_{n,i}^t = 1$ の条件を満たすような信頼度 $\alpha_{n,i}^t$ を求める．

$$\alpha_{n,i}^t = \frac{r_{n,i}^t}{\sum_{n'=0}^{N-1} r_{n',i}^t} \tag{9}$$

式 (9) で得られた $\alpha_{n,i}^t$ は，骨格情報の選択で利用した $\delta_{n,i}^t$ の値が 0 の場合も信頼度を計算する．骨格情報の

統合に利用しないデータに対しては，信頼度 $\alpha_{n,i}^t$ を必要としない．そこで，$\delta_{n,i}^t = 0$ とき $\alpha_{n,i}^t = 0$ とする．このとき，0 を代入したことによって $\sum_{n=0}^{N-1} \alpha_{n,i}^t = 1$ の条件を満たさなくなるため，条件を満たすよう修正を行う．

$$\alpha_{n,i}^{t\ *} = \frac{\alpha_{n,i}^t \delta_{n,i}^t}{\sum_{n'=0}^{N-1} \alpha_{n',i}^t \delta_{n',i}^t} \tag{10}$$

$\sum_{n=0}^{N-1} \alpha_{n,i}^{t\ *} = 1$ 条件を満たし，$\alpha_{n,i}^{t\ *}$ を信頼度とし，統合の際に使用する．また，$\alpha_{n,i}^{t\ *}$ は，$\delta_{n,i}^t$ の重みを含んでいるため，最終的な統合アルゴリズムは式 (11) となる．

$$\mathbf{j}_i^{t*} = \sum_{n=0}^{N-1} \alpha_{n,i}^{t\ *} \mathbf{j}_{n,i}^t \tag{11}$$

6　実験

本章では，まず，実験環境を説明したあとに，5.2 章で述べた骨格情報の統合の際に用いる 1 フレームあたりの最大移動距離 $d_{m,n}^t$ と各関節の関係性を説明する．また，複数台の Kinect を用いることで単体の Kinect と比較してオクルージョンが軽減されているかを確認し，提案手法である情報統合のアルゴリズムを用いたときの骨格情報の確認をする．

6.1　実験環境

本実験では，2 台の Kinect($\mathbf{K}_0$，$\mathbf{K}_1$) で実験を行った．図 2 のように Kinect を設置し，それぞれの Kinect のワールド座標と，角度のパラメータを既知となるように設置した後，キャリブレーションを行った．

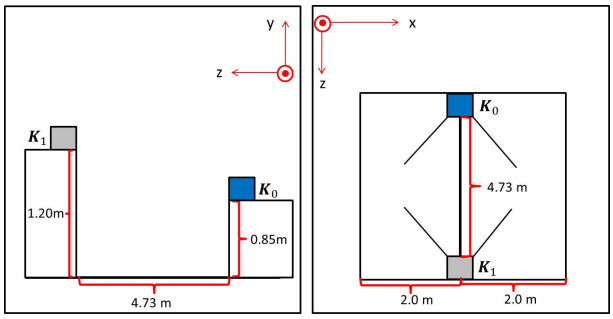

図 2: Kinect の設置図

骨格情報の統合には，フレーム間のキャリブレーションと，各関節の 1 フレーム当たりの最大移動距離を求める必要がある．この 2 つを行うために，テストデータを用いてそれぞれの値を求める．テストデータは，ワールド座標系の X,Y,Z 軸に対して動作するモーションで構成されている．また，データには，ノイズが含まれていないかまたは，ほぼノイズが含まれていないようなデータを使用した．主 Kinect($\mathbf{K}_0$) の骨格情報を中心としたときのその他の Kinect($\mathbf{K}_1$) とのフレーム間の時間誤差をデータから求める．ただし，1 fps は 33 msec. とする．計算した結果，時間的誤差は ±28 msec. であり，Kinect のフレームレートが 30 fps であることから前後 1 フレーム間のズレが生じる可能性が考えられる．骨格情報の統合の際には，前後 1 フレームの骨格情報を参照しフレーム間の調節を行う．次に，1 フレーム間の各関節の最大移動距離を表 1 に示す．表 2

表 1: 1 フレーム間の最大移動距離 $d_{n,i}$　　[m]

j	$\mathbf{K}_0$	$\mathbf{K}_1$
1	0.033	0.146
2	0.039	0.068
3	0.060	0.111
4	0.045	0.102
5	0.038	0.084
6	0.037	0.191
7	0.031	0.133
8	0.047	0.146
9	0.073	0.134
10	0.027	0.195
11	0.045	0.128
12	0.068	0.133
13	0.035	0.152
14	0.021	0.171
15	0.064	0.148
16	0.096	0.197
17	0.047	0.137
18	0.027	0.190
19	0.059	0.171
20	0.068	0.165
21	0.053	0.100
22	0.058	0.160
23	0.050	0.193
24	0.113	0.157
25	0.104	0.192

より，$\mathbf{K}_0$ の各関節の最大移動距離は，$\mathbf{K}_1$ と比較して小さいことがわかる．テストデータは，$\mathbf{K}_0$ が人物を正面にとらえる時間が長いことが要因として考えられる．また $\mathbf{K}_0$ の各関節ごとの最大移動距離は，足や手などの末端の関節や，動きの頻度が高い関節に対して，大きな値をとることが確認できる．しかし，$\mathbf{K}_1$ は $\mathbf{K}_0$ のような特徴がないことが確認できる．原因としてテスト用のデータに大きく影響していると考えられる．

6.2 実験結果

2 台の Kinect($\mathbf{K}_0$，$\mathbf{K}_1$) を使用し，骨格情報を統合した時の各関節のトラッキング数を表 2 に示す．$\{\mathbf{K}_0\}$ は，正面から得た情報を示し，$\{\mathbf{K}_1\}$ は背面から得た情報である．$\{\mathbf{K}_0, \mathbf{K}_1\}$ が 2 台の Kinect の情報統合によって得られた結果を示している．Tracked は，モーションデータを与え，1 フレームごとの各関節を正常にトラッキングできている回数であり，lost は，1 フレームごとの各関節を見失った時の回数である．rate は，モーションデータに対して，どの程度トラッキングできていたかを表す．$\{\mathbf{K}_0\}$ と $\{\mathbf{K}_1\}$ はほぼ同じトラッキング数であり，$\{\mathbf{K}_0, \mathbf{K}_1\}$ が単一の Kinect よりもよくトラッキングできていることが確認できる．この結果から複数台の Kinect の情報を用いたときオクルージョンが軽減できることがわかる．しかし，2 台の Kinect では，オクルージョン問題に対しては限界があり，本結果のように 92.5%のトラッキング率となったと考えられる．より多くの Kinect を用いることでオクルージョンを減らすことは可能であるが，設置位置やフレーム誤差のキャリブレーションなどの課題があり，単純には解決できないと考えられる．

表 2: 各関節のトラッキング数

	Tracked	lost	rate
$\{\mathbf{K}_0\}$	15302	2518	85.6%
$\{\mathbf{K}_1\}$	14812	2728	84.3%
$\{\mathbf{K}_0, \mathbf{K}_1\}$	16244	1306	92.5%

次に，ある時刻の各 Kinect が取得した骨格情報を可視化した結果を図 3，図 4 に示し，統合した骨格情報を図 5 示す．ただし，図中の赤いラインが X 軸，緑のラインが Y 軸，青のラインが Z 軸とする．図 3 は，$\mathbf{K}_0$ の骨格情報であり，右肩から右手と腰右部から右足にかけての関節にオクルージョンが発生している状態である．図 4 に，$\mathbf{K}_0$ の骨格情報を示す．$\mathbf{K}_0$ との骨格情報と対称的であり，左肩から右手と腰右部から右足にかけてオクルージョンが発生している状態である．この二つの骨格情報に対して，提案アルゴリズムによって統合した結果を図 5 に示す．$\mathbf{K}_0$ の座標系に対して骨格情報を統合した結果，$\mathbf{K}_0$ と $\mathbf{K}_1$ の骨格情報を利用して，オクルージョンが発生し不安定な骨格情報を排除し，精度の良い骨格情報を情報統合によって獲得できている．しかし，図 6 では，関節の座標情報がとなり，骨格情報の統合に失敗している．特に，末端の関節の値に大きな影響が確認できる．要因として，Kinect の台数が足りず，完全にオクルージョンをカバーできていないことが考えられる．また，提案したアルゴリズムでの信頼度は，テストデータに左右されやすいという課題があり，これらの要素によって骨格情報の統合が失敗したと考えられる．より精度の高い骨格情報の統合を行うためには，人体のモデルを利用し，図 6 のような関節の点にペナルティを与える方法が考えられる他，単純に Kinect の台数を増やし，精度を向上させることなどが考えられる．

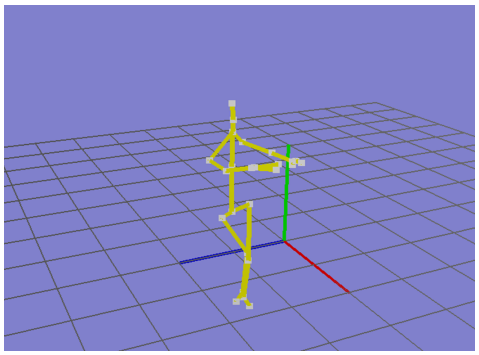

図 3: $\mathbf{K}_0$ の骨格情報

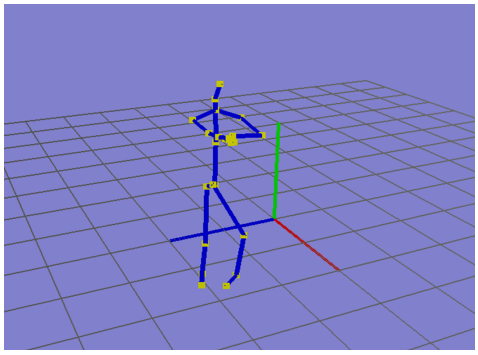

図 4: $\mathbf{K}_1$ の骨格情報

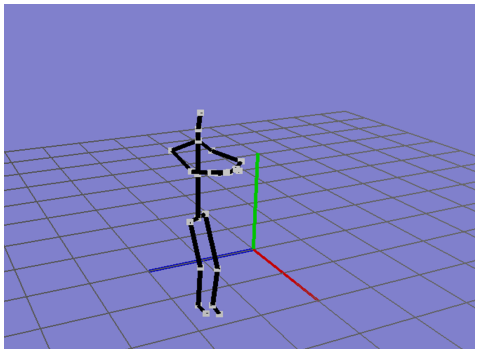

図 5: 骨格情報の統合結果

今後，より精度の高いモーションキャプチャを行うためには，信頼度の見直しだけでなく新たなパラメータの追加が考えられる．提案した信頼度は，テストデータ

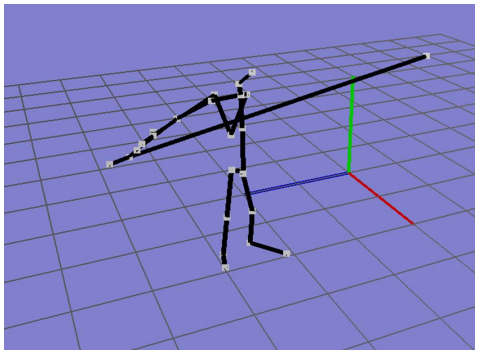

図 6: 骨格情報統合の失敗例

によって値が大きく変動するという欠点がある．新たな信頼度は，テストデータに依存しないように変更する必要がある．また，Kinect の特性を考慮するパラメータの追加や，骨格情報だけでなく，深度情報や RGB カメラの要素を利用することでより高精度化が可能であると考えられる．

7　おわりに

本研究では，ロボットなどの動作を作成するために必要なモーションキャプチャを複数台の Kinect を用いたシステムによって構築し，実現した．複数台の Kinect を用いるために，通常のカメラキャリブレーションに加え，Kinect 間で生じる時間的な誤差を修正する方法を提案した．また，骨格情報を統合するために，新たに信頼度というパラメータを定義し，統合を行うアルゴリズムを提案した．実験では，本システムを利用し，オクルージョンの軽減を確認し，オクルージョンに対応した骨格情報の統合を確認した．

今後は，より高精度なモーションキャプチャを実現するため，信頼度の見直しや，人体のモデルをパラメータとして情報統合に与えるなど，新たなパラメータを追加することでより高精度なモーションキャプチャが可能なシステムを目指す．

参考文献

[1] 総務省，第 2 部　ICT が拓く未来社会, http://www.soumu.go.jp/johotsusintokei/white-paper/ja/h27/html/nc241350.html

[2] Pepper, http://www.softbank.jp/robot/

[3] 中澤篤史, 中岡慎一郎, 白鳥貴亮, 工藤俊亮, 池内克史, モーションキャプチャによる全身運動解析と模倣ロボット―「じょんがら」節を HRP-1S に踊らせる―, 情報処理学会研究報告　*CVIM*, pp 31-39, 2004.

[4] Kinect, https://developer.microsoft.com/en-us/windows/kinect

[5] J. Shotton , A. Fitzgibbon, M. Cook, T. Sharp, M. Finocchio, R. Moore, A. Kipman , A. Blake, Real-time human pose recognition in parts from a single depth image, *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[6] S. Zennaro, M. Munaro, S. Milani, P. Zanuttigh, A. Bernardi, S. Ghidoni, E. Menegatti, PERFORMANCE EVALUATION OF THE 1ST AND 2ND GENERATION KINECT FOR MULTIMEDIA APPLICATIONS, *Proceedings of the 2015 IEEE International Conference on Multimedia and Expo*, ICME ’15, pp. 1-6, 2015.

[7] D. A. Butler, S. Izadi, O. Hilliges, D. Molyneaux, S. Hodges, D. Kim., Shake’n’sense: reducing interference for overlapping structured light depth cameras, *Proc. of CHI2012*, pp. 1933-1936, 2012.

[8] 占崎航, 加賀美聡, Kinect を用いた人型ロボットの全身制御, *Prooeedings of the 2011 JSME Conference on Robotics and Mechatronics*, No.11-5, 2P2-LO5, pp. 1-3, 2011.

[9] 平藤大智, 斎藤剛, 複数台のキネクトを用いた 3 次元モデル生成, 情報処理学会第 *76* 回全国大会, pp. 279-280, 2014.

[10] 吉本誠也, 高野茂, 岡田義広, 複数 Kinect ボーン情報を組み合わせた高精度モーションキャプチャシステムの構築, 情報処理学会第 *75* 回全国大会, pp219-220, 2013.

[11] A. Kitsikidis, K. Dimitropoulos ,S.Douka, N.Grammalids, Dance Analysis using Multiple Kinect Sensor, *VISAPP2014*, 2014.

[12] Z. Zhang, A flexible new technique for camera calibration, *IEEE TPAMI*, 22(11), pp.1330-1334, 2000.

[13] 奥富正敏 他, “ディジタル画像処理”,CG-ARTS 協会, pp. 251-267, 2004.