情報理工学特別研究報告書

題目

ARKit のモーションキャプチャ機能を使った モーションデータの生成に関する研究

学生証番号　854544

氏名　政岡　徹

提出日　令和 4 年 1 月 28 日

指導教員　蚊野　浩

京都産業大学
情報理工学部

要約

ビデオゲームや映画などに 3DCG が数多く使われている．3DCG を使ってアニメーションを行うには，3D モデルにあったモーションデータが必要である．本研究の目標は，Apple 社が開発した AR 用フレームワークである ARKit のモーションキャプチャ機能を使って，多目的に使用可能な，人体のモーションデータを作成することである．

ARKit で人体のモーションをキャプチャすると，キャプチャした姿勢の情報を持つ変数を得る．これは関節ごとの位置・姿勢を表現しており，これを，汎用的なモーションファイルとして出力した．本研究で利用するモーションファイルは BVH フォーマットである．このフォーマットでは，姿勢を表現する回転の情報がオイラー角となっているため，ARKit の形式である回転行列をオイラー角に変換する必要があった．その他，いくつかの適切な処理を加えることで，BVH ファイルとして出力することができた．その BVH ファイルを Blender 等のソフトウェアで読み込み，モーションを再生することができた．しかし，出来上がったモーションデータは実際の動きを正確に再現できていない場合が多かったため，モーションキャプチャの精度を，モーションデータを使って調査した．その結果，手首と足首の関節はどれだけ動かしても，出力ファイルに動きが反映されていないことが分かった．それらの関節については BVH ファイルに記述する回転角をすべて 0 にした．また，奥行きのある姿勢や，体の一部が隠れてしまう姿勢などをキャプチャした場合，モーションが乱れやすいことが分かった．また，動きが滑らかでないことも多く実用性に乏しかったため,モーションを平滑化するフィルタについて検討した．時間的に前後の動きとの平均を取るフィルタを実装したが，誤検出の影響を強く受けてしまい，精度が悪化するという結果になった．本研究では，モーションキャプチャの性能調査のための条件やサンプルが少ないため，今後の課題として，より多くの場合で精度を確認することや，誤検出に対応したフィルタを開発することなどが残った．

目次

1 章 序論

ゲームやアニメ，映画などにおいて，3DCG が利用されている．3DCG を使ったアニメーションで 3D モデルを動かすためには，モデルあったモーションが必要である．代表的なモーションの作成方法としてキーフレーム法とモーションキャプチャが存在する．キーフレーム法は人が手作業で要所要所の 3D モデルの姿勢を与え，その間をコンピュータが補間することでモーションを作成する．モーションキャプチャは実際の人や物の動きを読み取り，それを元にしてモーションを作成する．キーフレーム法の作業には特別な機材は基本的に必要無い．しかし人が手作業でモーションを作成するため，高品質なモーションを得ようとすると高度な技術を持った人材と時間が必要になる．一方，モーションキャプチャは実際の動きをそのまま使えるため，高品質なモーションを比較的簡単に得ることができる．また，Vtuber のようにリアルタイムに 3D モデルを操作することも可能である．しかし高性能なモーションキャプチャシステムは高価な機材やスタジオが必要となり，使用するためのハードルが高い．

AR（Augmented Reality，拡張現実）は現実世界に CG などを重ねて表示し，実際の風景をバーチャルに拡張する技術のことである．エンターテイメント分野を中心に様々な活用がされており，今後もますますの発展が期待される．AR はスマートフォンや AR グラスと呼ばれるメガネ状のデバイスを用いることが一般的である．

AR を実現するには複雑な処理を実装する必要がある．そこで米 Apple 社は iOS 用の AR フレームワークとして ARKit を提供している．これは AR を実現するために必要な様々な機能を実装した API であり，デベロッパが簡単に AR アプリを作成することができる．そのような ARKit の機能の一つにモーションキャプチャが存在する．ARKit のモーションキャプチャは，バックカメラに写った人体の画像から姿勢を計算しており，そのアルゴリズムに機械学習をベースとした手法を採用している．

本論文では，ARKit のモーションキャプチャ機能を利用して，様々な用途に使用可能なモーションデータを作成する研究を行った．iPhone や iPad といった広く普及したデバイスで，誰でも簡易にモーション作成を行えるようにすることが目標である．また，この機能を利用し，ARKit のモーションキャプチャの精度に関しても調査し，考察を行った．

2章　モーションキャプチャとモーションデータ

2．1　モーションキャプチャ

モーションキャプチャは，人や物の動きをキャプチャし，その動きに関する情報をデジタル化する技術のことである．ゲームや 3DCG アニメーションで 3D モデルを動かす場合や，スポーツなどでの動きの解析に使用されている．

モーションキャプチャの代表的な方法に光学式，慣性式，磁気式，機械式，ビデオ式などがある[4]．光学式はキャプチャ対象に取り付けたマーカーの位置を複数のカメラで読み取り，動きを得る方法である．マーカーの位置を読み取るため，人に限らず様々な物の動きをキャプチャできる点や，位置精度の高さから最も広く使われている．慣性式は体に取り付けた慣性センサから得た情報を基に姿勢を算出する．カメラなどの外部機材が必要ないため狭いスペースでも使えるが，位置精度はあまり高くない．磁気式は磁界を発生させ，体に装着した複数のセンサで磁場の変化を読み取り，動きのキャプチャを行う方法である．磁場を使うので，遮蔽物によって死角が生まれることは無いが，環境によっては磁場に影響を与えたり，センサから伸びる配線によって動きに制限が生まれたりする可能性がある．機械式は人の関節に角度を測定する機械を取り付け，動きのキャプチャを行う方法である．周辺環境の影響は受けにくいが，装着する機械や配線によって動きが制約される可能性がある．ビデオ式はカメラで撮影した画像をもとにキャプチャを行う．マーカーレスな計測が可能だが，一般に位置精度は低い．ARKit のモーションキャプチャはビデオ式であり，人間を撮影した映像からモーションデータを得るために機械学習に基づくアルゴリズムを使っている．

2．2　ARKit と関連するフレームワーク

ARKit は Apple 社が iOS デバイス向けに提供している AR 開発用のフレームワークである．2017 年の公開以降，着々と機能を追加し続け，2022 年 1 月現在は ARKit5 が公開されている[2]．モーションキャプチャ機能は 2019 年に公開された ARKit3 から実装されている．ARKit は iOS 11.0 以降で，A9 以降のプロセッサを搭載した iOS デバイスが必要である．機能によってはより新しいバージョンの iOS やデバイスが必要となる場合もある．モーションキャプチャ機能は A12 以上のプロセッサが必要である．また，一部の iOS デバイスは周囲の 3 次元的な情報を取得する LiDAR スキャナが搭載されており，より精度の高い AR 体験が期待できる．また，ARKit は 60fps でカメラ画像を取得している．

AR 体験を実現するために 3D オブジェクトを描画する場合，レンダリングフレームワークを別途使用する必要がある．Apple が提供する，AR 用に開発されたレンダリン

グフレームワークとして RealityKit があり，ARKit と連携させて使用する．ARKit と RealityKit を使った AR アプリケーションの基本的なプログラムについて説明する．

ARKit を使用するためには，ARSession というクラスを利用する必要がある．一方，RealityKit を使用する場合，ARView という AR コンテンツを描画するためのビューが ARSession のインスタンスを生成するため，ARSession の起動を自分で記述する必要はない．

次に，ARKit のどの機能を使うのかを決める Configuration を設定し，セッションを run する．図 2.1 にその時のプログラムの例を掲載する．

```
let configuration = ARWorldTrackingConfiguration()
arView.session.run(configuration)
```

図 2.1 AR を開始するのに必要なプログラムの例

この例では Configuration に ARWorldTrackingConfiguration を設定している．これはデバイスの位置や向きを取得するためのものであり，ARConfiguration クラスのサブクラスである．他にも GPS や地図データから位置を取得する ARGeoTrackingConfiguration や，フロントカメラから顔の動きと表情を取得する ARFaceTrackingConfiguration などのサブクラスがあり，目的に応じて使用する．また，各 Configuration 内でも使用する機能を設定できる場合がある．

ARSession で起こるイベントは，ARSessionDelegate で処理する．ARSessionDelegate には，カメラ画像が更新された時や，ARAnchor が追加，更新された時などに呼び出されるイベントが用意されている．ここでは特に ARAnchor に関して説明する．ARAnchor は空間上に配置されるアイテムの位置や姿勢を表すクラスである．位置や姿勢の情報は，ARAnchor のトランスフォーム（transform プロパティ）に保持する．ARKit が平面検出やモーションキャプチャなどを行った時，検出したオブジェクトのトランスフォームが ARAnchor のリストに追加される．

また，ARView 上に実際にオブジェクトを表示するためには，ARView が持つ scene に追加する必要がある．scene はレンダリングする 3D エンティティを格納する変数である．

2. 3 ARKit におけるモーションキャプチャ

ARKit でモーションキャプチャを利用する方法を Apple が提供しているサンプルプログラム「Capturing Body Motion in 3D」[1]を例に説明する．

図 2.2 のように，ARSession の ARConfiguration に ARBodyTrackingConfiguration を設定する．

```
let configuration = ARBodyTrackingConfiguration()
arView.session.run(configuration)
```

図 2.2 モーションキャプチャを利用する際の Configuration 設定

これだけでモーションキャプチャが行われ，カメラで撮影した画像中に人体を認識した時に ARBodyAnchor が ARAnchor のリストに追加される．これはキャプチャした人体の姿勢を保持している．このプログラムでは BodyTrackedEntity 型の character というメンバ変数を持っている．これはキャプチャした姿勢に従って動く 3D モデルのエンティティであり，Apple が用意した 3D モデルをロードしている．これをシーンに追加するとキャプチャした通りの動きをする 3D モデルが図 2.3 の様に表示される．

図 2.3「Capturing Body Motion in 3D」の実行例[1]

2．4　BVH ファイル

BVH は，Biovision 社が開発したモーションデータを記録するためのファイルフォーマットである[3]．図 2.4 に BVH ファイルの例を示す．ファイルはテキストデータであり，2 つの部分に分かれている．HIERARCHY で始まる部分は骨格の階層構造と初期姿勢を定義している．MOTION で始まる部分は各関節の動きを示している．

```
HIERARCHY
ROOT root
{
  OFFSET 0.000000 0.000000 0.000000
  CHANNELS 6 Xposition Yposition Zposition Xrotation Yrotation Zrotation
  JOINT hips_joint
  {
    OFFSET -0.000000 0.000000 0.000000
    CHANNELS 3 Xrotation Yrotation Zrotation
    JOINT left_upLeg_joint
    {
      OFFSET 0.100540 -0.024969 0.001188
      CHANNELS 3 Xrotation Yrotation Zrotation
      JOINT left_leg_joint
      {
        OFFSET -0.000063 -0.418794 0.044015
        CHANNELS 3 Xrotation Yrotation Zrotation
        JOINT left_foot_joint
        {
          OFFSET 0.000056 -0.451676 -0.071544
          CHANNELS 3 Xrotation Yrotation Zrotation
          JOINT left_toes_joint
          {
            OFFSET -0.000148 -0.057876 0.137335
            CHANNELS 3 Xrotation Yrotation Zrotation
            JOINT left_toesEnd_joint
            {
              OFFSET -0.000064 -0.001059 0.060651
              CHANNELS 3 Xrotation Yrotation Zrotation
              End Site
              {
                OFFSET -0.000064 -0.001059 0.060651
              }
            }
          }
        }
      }
    }
 (中略)
}
MOTION
Frames: 2
Frame Time: 0.016666
0.0 0.0 0.0 6.27315 -35.71763 0.95748764 0.5088709 -0.01720456 -0.020046335 -13.6494255 5.068608 2.6307757
19.741205 23.586985 -7.9058814 2.6826084 18.901114 32.10575 2.941923 0.5828457 11.0975895 1.6342e-05
1.8983157e-05 -2.8317287e-05…
```

図 2.4 BVH ファイルの例

HIERARCHY から始まる部分では ROOT から階層的に関節(JOINT)が定義される．ROOT は親を持たない特別な JOINT であり，すべての JOINT の親子関係をたどると ROOT に行き着く．JOINT は必ず親と子を持ち，親 JOINT からのオフセット値（平行移動量）と CHANNELS（関節の自由度）を持つ．末端に位置する JOINT は，オフセットのみを持った“End Site”を子に持つ．CHANNELS は JOINT が持つパラメータであり，移動と回転

の情報を持たせることができる.

また,“CHANNELS 3”のようにパラメータをいくつ持つのかを記述する必要がある.そのうち移動は XYZ 軸に対して“Xposition”,“Yposition”,“Zposition” と記述することで設定できる.回転は同様に“Xrotation”,“Yrotation”,“Zrotation”と記述することで設定できる.各関節にどの情報を持たせるか,どの順番で記述するかはユーザが任意に設定できる.例では ROOT のみが移動量を持ち,他の JOINT は回転のみを持っている.また,回転は XYZ の順で記述されているが,これは実際の回転の順を表しており,記述する順を入れ替えることで回転の順も入れ替えることができる.なお,BVH で使用される回転はオイラー角で単位は度である.

MOTION 部分では,まず Frames:の後に総フレーム数が,Frame Time:の後に 1 フレームあたりの秒数が記述される.例では 0.016666 となっており,これは 60fps を意味し,ARKit と同じ 60fps である.その後 1 行につき 1 フレームで全 JOINT のパラメータが HIERARCHY で定義した順に記述される.

3 章　BVH ファイルのモーションデータへの変換

ARKit のモーションキャプチャを利用して，人体の 3D モデルに適用できるモーションデータファイルを生成する方法について説明する．なお，本研究は以下の開発環境で実施した．

開発環境：

MacBook Air(13-inch, 2017)

MacOS Big Sur version 11.6.2

Xcode version 13.2.1

プログラミング言語：

Swift

使用端末（iOS デバイス）：

iPad Pro （11-inch, 第 3 世代）

iPad OS 15.2.1

3. 1　モーションキャプチャで得るデータ

ARKit のモーションキャプチャでは，人体が検出されると ARBodyAnchor が ARAnchor のリストに追加される．ARBodyAnchor は ARSkeleton3D クラスの変数を含んでいる．これはキャプチャした人体の骨格について，3 次元空間における 91 個の関節（ジョイント）の位置と角度を保持したものである．91 個の関節には図 3.1 のように名前がつけられており，これらを使い特定のジョイントの値を呼び出すことができる．

図 3.1 ジョイント一覧[1]

ARSkeleton3D の 91 個のジョイントは図 3.2 のように接続されている．ジョイントの接続関係は，腰関節に位置する root を最上位として、上位・下位の親子関係を持つ階層的な構造になっている．

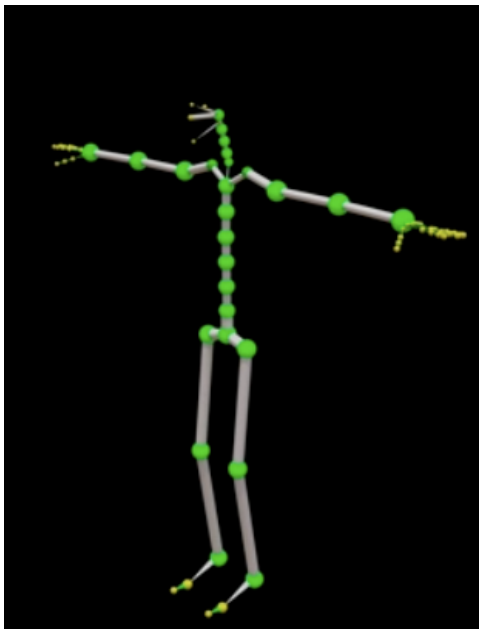

図 3.2 91 個のジョイントを持つスケルトン[1]

各ジョイントの位置と姿勢は，modelTransform と localTransform という２つの方法で得ることができる．いずれのトランスフォームも，ある基準点からの回転と移動の情報を持った 4×4 の変換行列で表現される．modelTransfrom の基準点は root，localTransfrom の基準点は親のジョイントである．左手首の関節を例にすると，図 3.3 のようなイメージである．なお，ARBodyAnchor の原点は root と同じになっている。

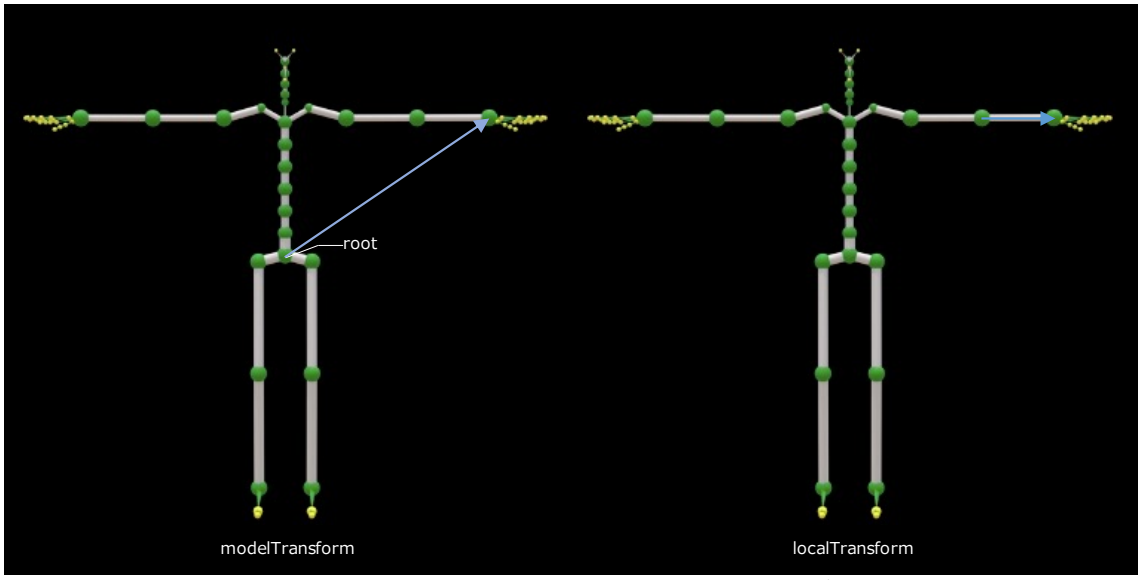

図 3.3 modelTransform と localTransform の違い[1]

3. 2　データの変換

BVH ファイルに記述する回転は，階層構造の親を基準点としたものである．従って，BVH ファイルの回転情報を求める場合，localTransform が適していると考えられる．しかし，localTransform では期待する結果を得ることができなかった．原因は分かっていないが，これに変わる方法として，modelTransform を使って localTransform に相当する変換行列を生成して利用した．

あるジョイントの modelTransform は，その親ジョイントの modelTransform と自身のローカルなトランスフォームの合成と考えることができる．図 3.4 はそのイメージである．よって modelTransform をローカルなトランスフォームに変換するには，変換したいジョイントの modelTransform に，親ジョイント modelTransform の逆行列をかけてやればよい．

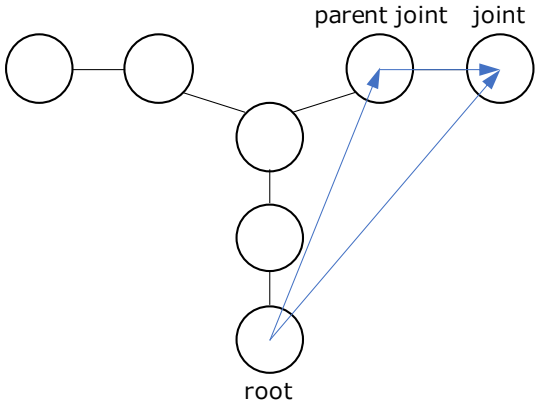

図 3.4 modelTransform と localTransform の関係のイメージ

次に，初期姿勢の違いを考慮する必要があった．BVH ファイルで定義する人体骨格は，すべてのジョイントの回転角が 0 度の時 T ポーズを取る．T ポーズは図 3.5 のような姿勢であり，人型 3D モデルのニュートラルな姿勢として広く利用されている．

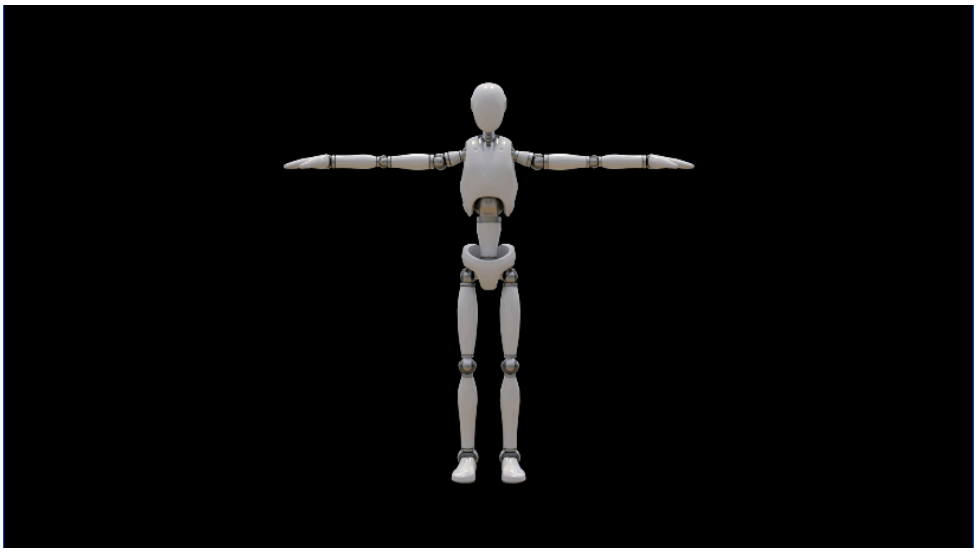

図 3.5 T ポーズ[1]

一方，ARKit のモーションキャプチャで使われている ARSkeleton3D は，すべての回転角を 0 度にしても T ポーズを取らない．従って，図 3.6 で示すように，T ポーズからキャプチャポーズへの変換行列を得るためには，初期姿勢からキャプチャポーズへの変換行列に，初期姿勢から T ポーズへの変換行列の逆行列をかけてやれば良いことが分かる．ARkit における T ポーズの情報は，neutralSkeleton3D として定義されており，そこから T ポーズの modelTansform と localTransform を参照できる．以上の処理をまとめると，各ジョイントの modelTransform に親ジョイントの modelTransform の逆行列をかけ，初期姿勢から T ポーズへの変換行列の逆行列をかけることで目的の変換行列を得ることができる．

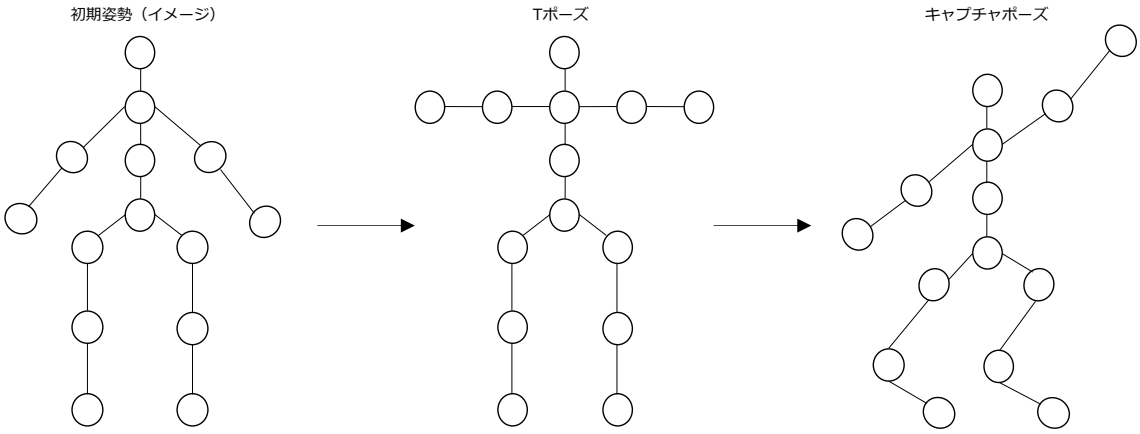

図 3.6 ARSkeleton3D の変換行列のイメージ

ジョイントの最上位に位置する root は親ジョイントを持たないので，平行移動と回転移動の量は 0 である．ジョイントの階層構造とは別に，モデル全体を示す ARSkeleton3D にもトランスフォームが存在し，これはキャプチャした骨格のワールド

座標を表している．そこで，今回生成する BVH ファイルでは，ARSkeleton3D トランスフォームを root の平行・回転移動の量とする．キャプチャ開始時のワールド座標を記録しておき，各フレームでのワールド座標から引くことでキャプチャ開始地点を（0，0，0）とした時の移動量を得た．

次に，得られた変換行列をオイラー角に変換する．トランスフォームは 4×4 の変換行列であるから，その回転部分だけに注目し，回転行列をオイラー角に変換する．XYZ の順に回転させた場合の回転行列は

$$R = \begin{pmatrix} \cos\theta_y \cos\theta_z & -\cos\theta_y \sin\theta_z & \sin\theta_y \\ \sin\theta_x \sin\theta_y \cos\theta_z + \cos\theta_x \sin\theta_z & -\sin\theta_x \sin\theta_y \sin\theta_z + \cos\theta_x \cos\theta_z & -\sin\theta_x \cos\theta_y \\ -\cos\theta_x \sin\theta_y \cos\theta_z + \sin\theta_x \sin\theta_z & \cos\theta_x \sin\theta_y \sin\theta_z + \sin\theta_x \cos\theta_z & \cos\theta_x \cos\theta_y \end{pmatrix}$$

となる．

$$R = \begin{pmatrix} r_{00} & r_{01} & r_{02} \\ r_{10} & r_{11} & r_{12} \\ r_{20} & r_{21} & r_{22} \end{pmatrix}$$

とすると，

$$\theta_y = \arcsin(r_{02})$$

としてθ_yが求められる．次に、$\cos\theta_y \neq 0$の時

$$\theta_x = \arctan\left(-\frac{r_{12}}{r_{22}}\right)$$

$$\theta_z = \arctan\left(-\frac{r_{01}}{r_{00}}\right)$$

$\cos\theta_y = 0$の時，

$$\theta_x = \arctan\left(\frac{r_{21}}{r_{11}}\right)$$

$$\theta_z = 0$$

としてθ_xとθ_zが求められる．こうして求めた数値を BVH ファイルの MOTION 部分に順に書き込むことでモーションキャプチャした動きの BVH ファイルを作成した．

3．3　アプリケーションへの実装

iPad でモーションキャプチャを行い，取得したデータから BVH ファイルを生成するアプリケーションを開発した．アプリケーションが動作している様子を図 3.7 に示す．アプリケーションを起動するとバックカメラが取得した画像が画面に

映し出される．人体を検出すると，人体の主要な関節の上に青い球体が表示される．この球体はモーションを記録している間，赤色に変化し，記録が終了すると青色に戻る．また，リアルタイムにモーションキャプチャの精度を確認することを目的として，キャプチャ対象の動きに追従する 3D モデルを表示した．

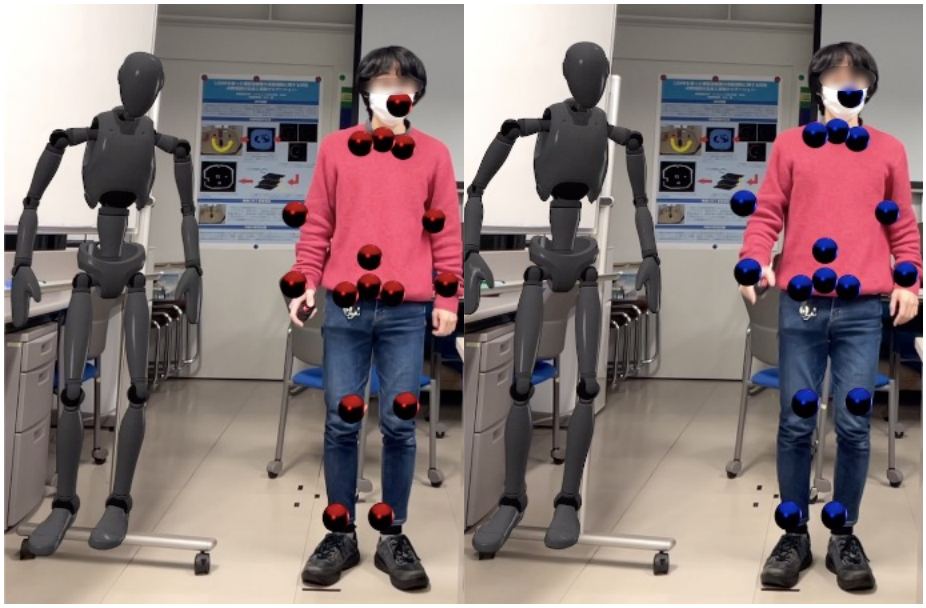

図 3.7 キャプチャ時と非キャプチャ時の画面の様子

プログラムでは，記録を始めると，1 フレーム毎に ARBodyAnchor を配列に収める．キャプチャを終了した後，名前を付けてエクスポートを行うと，配列に収めた Anchor をもとに root の移動量や各関節の回転角を求める．ARSkeleton3D と対応した HIERARCHY を持つ BVH ファイルを用意しておき，その MOTION に求めた数値を記述したものを最終的に出力する．

4 章　動作確認と性能評価

4. 1　生成したモーションデータの確認

生成した BVH ファイルがキャプチャした人体の動きになっていることを，Blender を使用して確認した．Blender は 3DCG 制作に用いるオープンソースソフトウェアであり，ネイティブに BVH ファイルに対応している．図 4.1 にキャプチャした姿勢と Blender で読み込んだモーションの一例を示す．

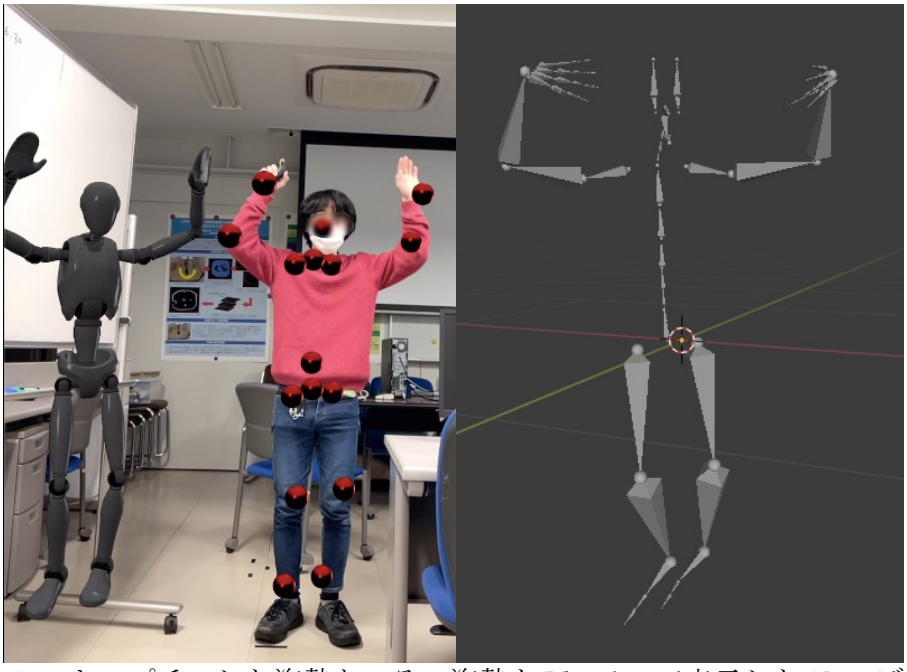

図 4.1 キャプチャした姿勢と，その姿勢を Blender で表示した 3D モデル

この図では，キャプチャ時の姿勢を概ね再現できているように見えるが，手首の向きがおかしい．アニメーション全体でも手の向きは不自然なままであった．また，人体モデルの姿勢が急変するなど，明らかにキャプチャした姿勢が再現できていない場面があった．

4. 2　正常にキャプチャできない部位

生成したモーションデータを見ると，明らかにキャプチャ対象の姿勢とは異なる回転量が記録されている関節があった．手や足の関節について，部位を様々に動かしたモーションを取得し，実際の動きと比較した

最初に，手首から先の関節について検証した．ARKit 内での手のジョイントは，左手の小指を例に図 4.2 のような位置と名称になっている．親指は Thumb，人差し指は Index，中指は Mid，薬指は Ring，小指は Pinky で表され，各指が例と同様の命名規則で名前が付けられている．

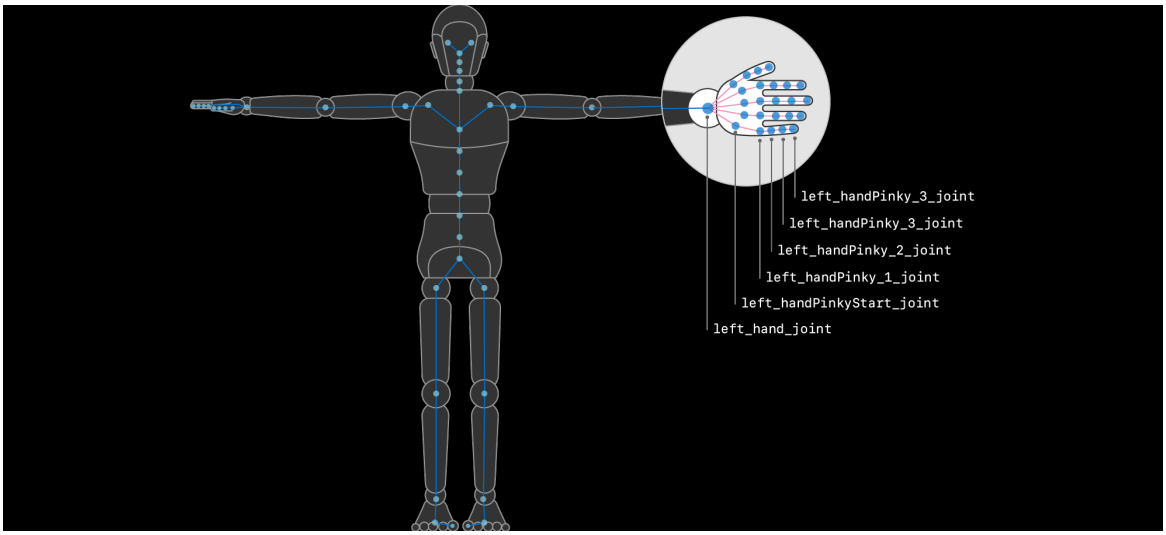

図 4.2 左手の関節の名称と位置関係[5]

左手以外の姿勢を固定し，手首と指だけを動かすモーションをキャプチャした．キャプチャ時の姿勢と出力された BVH ファイルで再生したモーションを図 4.3 に示す．左手首を曲げたり，手の平を開閉させたりしても対応するジョイントは動かず，実際の動きが一切反映されていない．これは右手でも同様であり，手首から先の動きはキャプチャできていなかった．

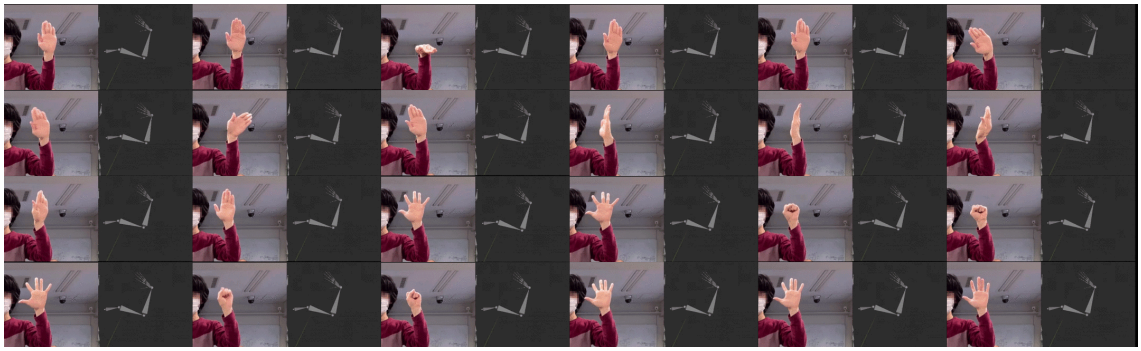
図 4.3 左手の姿勢とモーションデータ

次に，足首から先の関節について検証する．足のジョイントは図 4.4 のような位置と名称になっている．

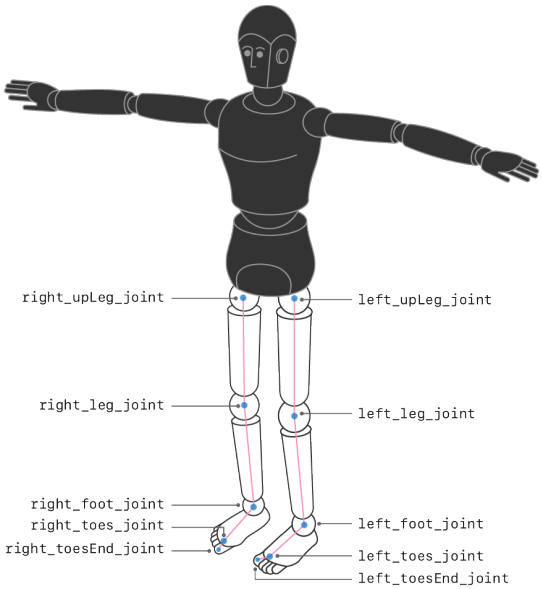

図 4.4 足の関節の名称と位置関係[5]

足の動きがはっきりとカメラに映るように，横を向いて左足を上げた状態でキャプチャを行った．キャプチャ時の姿勢とモーションデータを図 4.5 に示す

図 4.5 左足の姿勢とモーションデータ

足首を動かしたり，足の指を握ったりしても特に対応するジョイントは動かなかった．手と同様に動きをキャプチャできていないことがわかった．

これらの関節はキャプチャ時の姿勢が反映されていないだけではなく，おかしな方向を向いていることもあった．従って，それらの関節については，BVH ファイルに記述する回転量を 0 にした．

4．3　キャプチャーが難しい動き

姿勢を正しくキャプチャできず，全く違った姿勢を取ることがあった．いくつかの動きのパターンをキャプチャした結果から，ARKit が苦手とする動作の条件を明らかにする．

正面を向いた状態で腕を回転させる動きをキャプチャした．その様子と得られたモ

ーションデータを図 4.6 に示す．モーションデータでは腕が前に来たあたりから，正面ではなく右を向いてしまっている．腕の回転運動自体は概ね実際の動きを追従しているように見える．

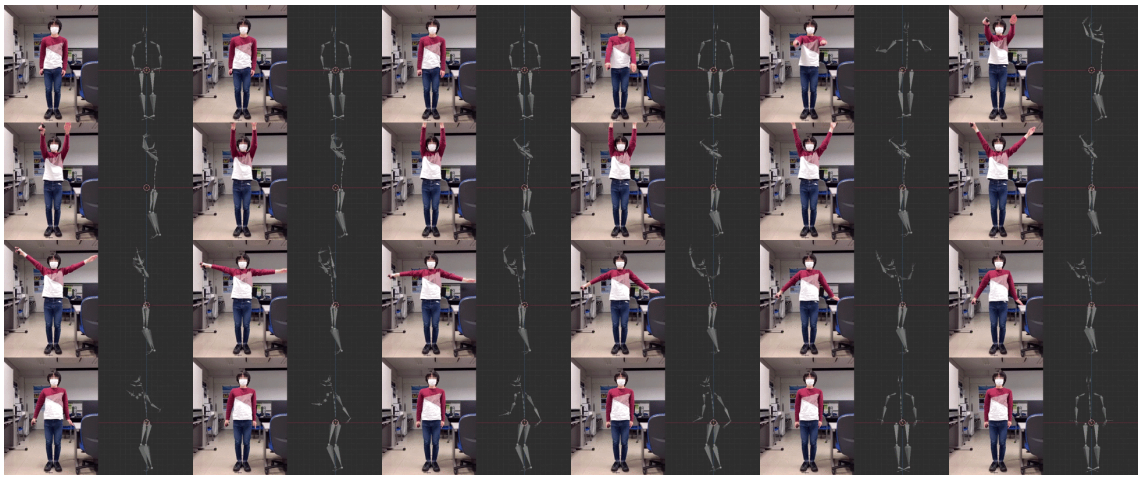

図 4.6 腕を回転させる動きとモーションデータ

T ポーズの状態から腰をひねる動きをキャプチャした．その様子と得たモーションを図 4.7 に示す．片腕が正面に回り，もう片方の腕が体に隠れたあたりが最も乱れている．T ポーズに戻ると再び安定する．

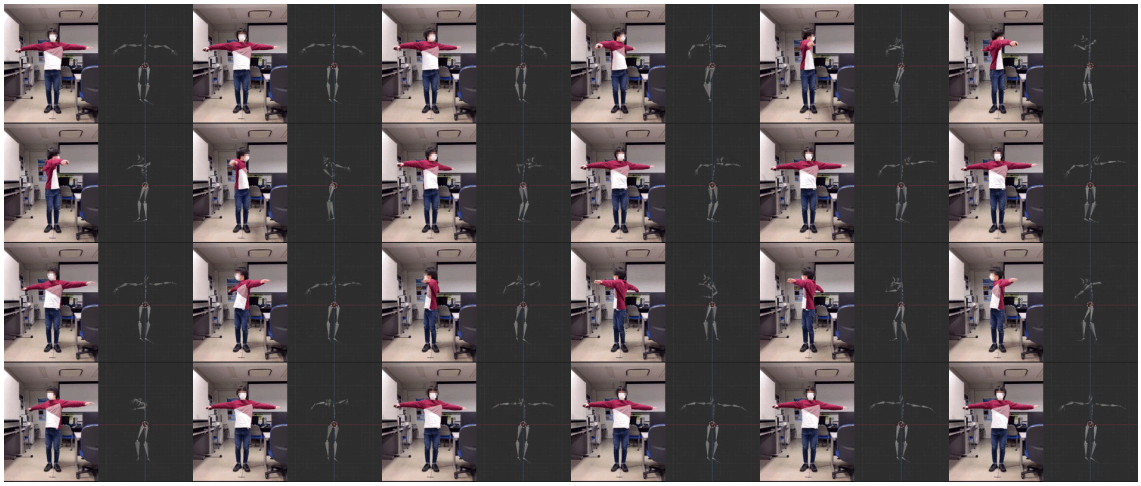

図 4.7 腕を広げて腰を回転させる動きとモーションデータ

立った状態から前後屈運動を行いキャプチャした．その様子と得たモーションを図 4.8 に示す．手が地面に近づいたあたりでは，腰が折りたたまれ，腕は体側に添わせたようになっている．その後，全体の座標がずれ，もとの姿勢に戻ると姿勢は乱れたままだが座標は元の位置に戻っている．後屈を行っている時は，腰を反らさず全体が後ろを向いてしまっている．

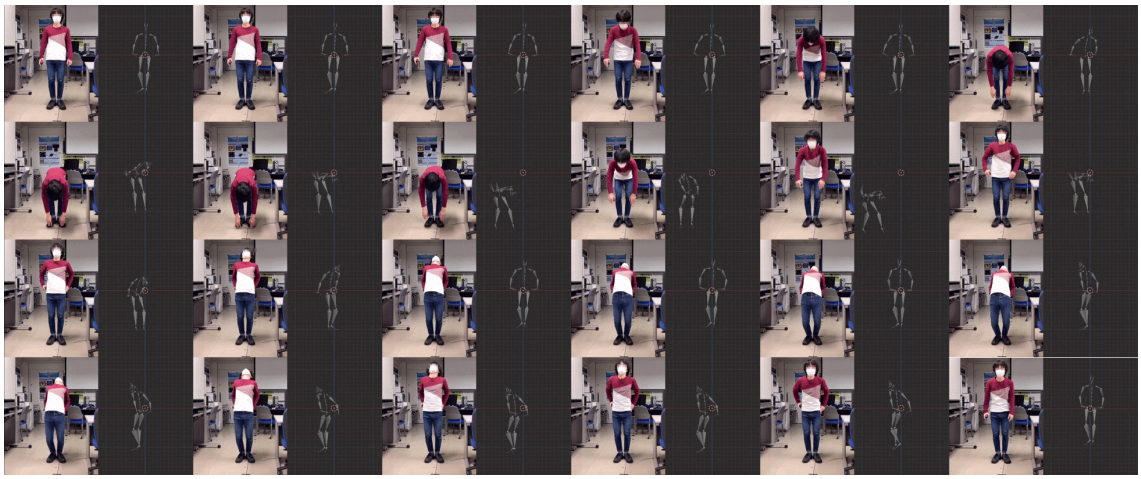

図 4.8 前後屈とモーションデータ

脚を前や横に動かしてキャプチャを行った．その様子と得たモーションを図 4.9 に示す．脚を前に蹴るような動作をした場合，大腿の動きは概ね同じように見えるが，膝は曲げていないにも関わらず曲がってしまっている．横への動作では，体が少し回転してしまっているが，脚の動き自体は実際のものと近い．

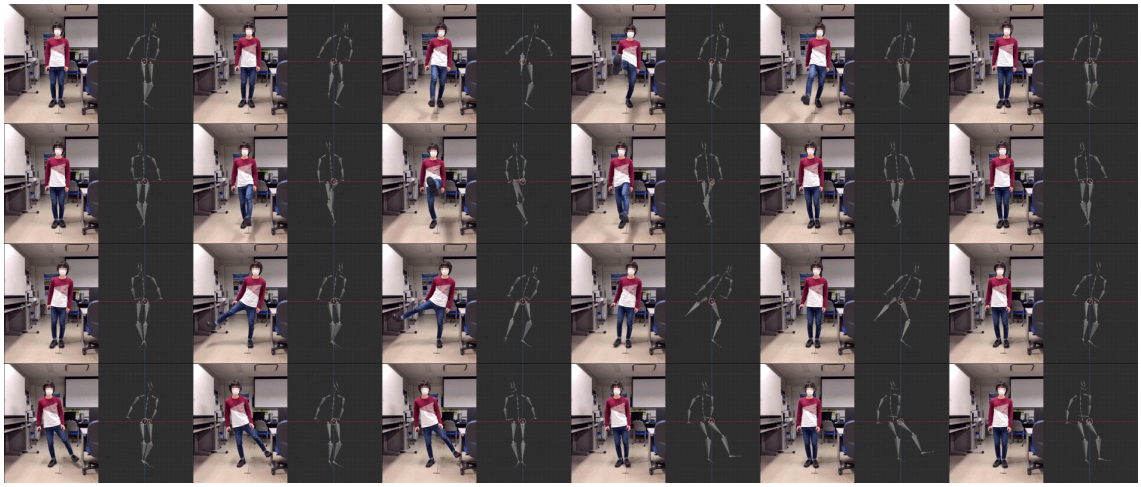

図 4.9 脚の動きとモーションデータ

その場でしゃがみ，ジャンプを行いキャプチャした．その様子と得たモーションを正面から見たものと，真横から見たものを図 4.10 に示す．まず，しゃがんだ時点で位置が大きく後方に移動してしまっている．ジャンプしている場面では，元の位置から鉛直方向に飛び上がり，着地できている．

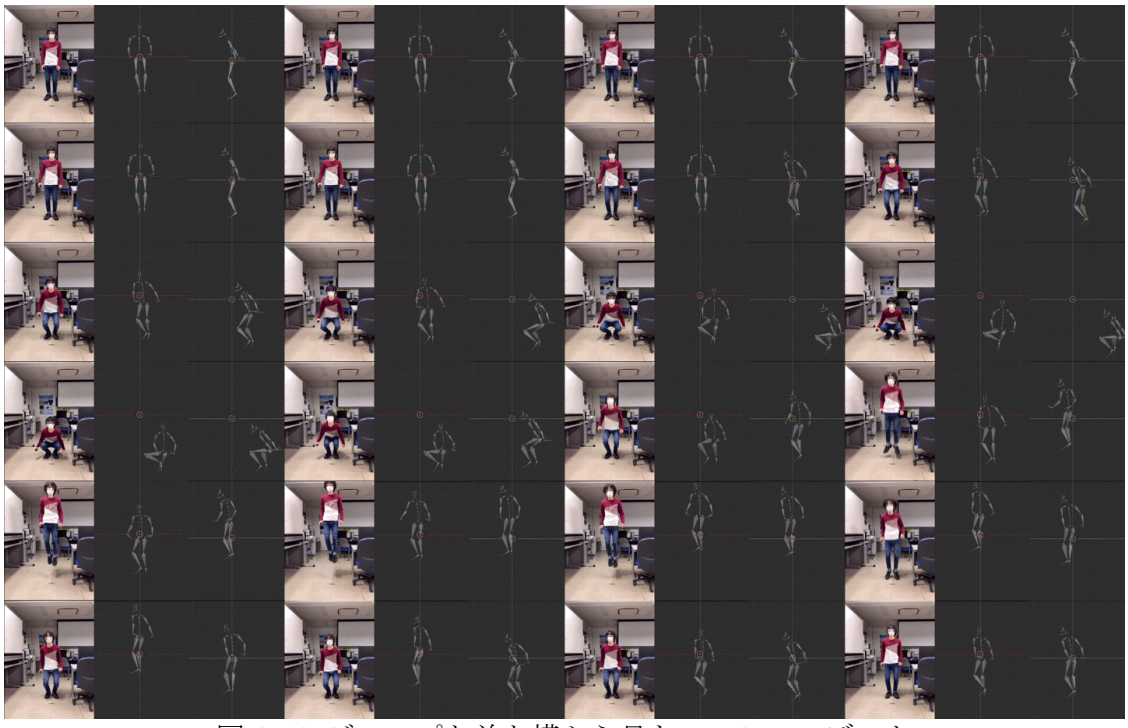

図 4.10 ジャンプと前と横から見たモーションデータ

図 4.6，図 4.7，図 4.9 のように，腕や足を前方に向けるなど奥行きのある姿勢を取った時にキャプチャの乱れが発生している．また，図 4.7 や図 4.8 のように体の一部が隠れた場合にも乱れが発生している． 1 台のカラーカメラでキャプチャを行っているため，死角になる部分の動きを正確に推定できていないと考えられる．また，T ポーズや直立姿勢などで，正面を向いている時は比較的安定して動作が取れている．正面を向き，四肢を含む体全体がはっきり写っている姿勢のキャプチャ精度は高い．

まとめると，ARKit のモーションキャプチャは奥行きのある姿勢や体の一部が隠れてしまっている姿勢をキャプチャした時データが乱れやすく，体全体がはっきりと写るような姿勢をキャプチャした時データは乱れにくいのだと考えられる．しかし，今回の検証では正面を向いた状態から動き始めるようなモーションしかキャプチャしておらず，その数も少ない．上手く取得できなかったモーションも，工夫次第ではそれなりにキャプチャできる可能性もあるため，より多くの条件での検証が必要である．

4．4　動きの補正

このシステムで得たモーションは，実際の動きの変化に対して滑らかに追随できなかったり，急におかしな姿勢を取ったりする場合が多々見受けられた．このままのデータでは実用性は低い．出力後のデータを扱いやすくするため，動きを補正するフィ

ルタをかけることを検討した．今回はモーションを滑らかにする目的で，各フレームについて，その前後の数フレームとの平均を取るフィルタを実装した．

実装したフィルタをかけたモーションとかけていないモーションを比較したが，特に滑らかになることはなくモーションによってはより精度が悪くなる結果となった．このシステムでは 4 章 3 節のデータの様に，誤計測が多く発生する．このようなデータがある状態で平均を取ったため，誤計測が無い部分もその影響を強く受けたためである．より良いモーションデータを得るには，誤計測の影響を除去できるようなフィルタを作成する必要がある．

5章　結論

本研究では，ARKit のモーションキャプチャ機能を使って人型 3D モデルを動かすためのモーションデータ生成するプログラムを作成した．モーションデータを BVH 形式で出力し，Blender などのソフトウェアで使用することができるようにした．

生成したデータを使ってモーションキャプチャの性能について調査を行った．その結果，手や足の関節は実際に動かしてもキャプチャしたデータに反映されていないことが分かった．そのためそれらの関節は実際に出力するデータでは動かないように設定した．さらに，いくつかのパターンの動きをキャプチャし，どのような動作が上手くキャプチャできないのかを検証した．この検証では奥行きのある姿勢や，体の一部が隠れてしまうような動作で誤検出が起こりやすいことが分かった．しかし，条件やサンプル数の少なさから検証内容は不十分だと考えられるため，今後の課題としてより多くの条件での検証が必要である．

また，出力したデータはあまり滑らかでなかったり誤検出が多く含まれたりしており，実用的ではない.そのため,それらを補正するフィルタについて検討した．今回は動きを滑らかにすることを目的とし，前後の動きとの平均を取るフィルタを実装したが効果は得られず，むしろ誤検出の影響を受けモーションデータの精度が下がってしまった．誤検出を補正するようなフィルタを検討する必要があると考えられる．

参考文献

[1] WWDC 2019 session 607: Bringing People into AR,
https://developer.apple.com/videos/play/wwdc2019/607

[2] ARkit - 日本語ドキュメント - Apple Developer,
https://developer.apple.com/jp/documentation/arkit/

[3] M. Meredith, S. Maddock, “Motion Capture File Formats Explained”
https://www.researchgate.net/publication/242308869_Motion_Capture_File_Formats_Explained

[4]【モーションキャプチャとは？】原理や活用シーンを種類別に解説 | モーションキャプチャシステム販売|株式会社スパイス
https://mocap.jp/about-mocap/

[5] Validating a Model for Motion Capture | Apple Develper Documentation
https://developer.apple.com/documentation/arkit/content_anchors/validating_a_model_for_motion_capture

謝辞

本論文を作成にあたり，丁寧な御指導を賜りました蚊野浩教授に感謝いたします.

付録　本研究で開発したプログラム

ViewContrloler.swift

キャプチャを行う画面を制御するプログラム．検出した人体の上に球体を表示する，3D モデルの表示をするなどの処理を行っている．

exportViewController.swift

得られた ARBodyAnchor の配列から BVH ファイルを出力するための処理を行っている．

・calcEuler

modelTransform を BVH に記述するオイラー角まで変換する関数

・bvhExport

calcEuler で求めたオイラー角を BVH ファイルの雛形に記述して出力する関数

・makeBvhMotion

オイラー角を BVH ファイルに記述できるように並べて文字列を作る関数 bvhExport 内で使われる．