コンピュータ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利用した CG キャラクタの動作制御に関する研究

学生証番号　744456

氏名　小林健将

提出日　令和 3 年 1 月 22 日

指導教員　蚊野　浩

京都産業大学
コンピュータ理工学部

要約

AR(Augmented Reality，拡張現実)技術は，カメラを通して目に見えている景色などに，コンピュータを使って情報を付加する技術である．Apple 社が公開している ARkit は，AR 技術を利用した iOS 端末用のアプリケーションを開発するためのフレームワークである．本研究では ARkit の機能の一つであるモーションキャプチャ機能に関する調査を行い，この機能を用いたデモアプリケーションを開発した．

ARkit のモーションキャプチャはカメラで人体を撮影し，機械学習を使った AI 技術が人体画像を処理し，主要な関節の位置を測定する．そして，疑似的なスケルトンを生成することで動きを認識している．配布されているサンプルプログラムでは，取得した動きのデータを，予め準備されている標準的な人体スケルトンの 3D モデルに同期させることで，カメラで認識した動きと同様の動きをする 3D モデルを，AR オブジェクトとして描写している．標準的な人体スケルトンが，アプリケーションにおいて重要な役割を担っており，このスケルトンと同様のスケルトンを持つ 3D モデルならば，RealityKit を用いることで容易に制御・描画することができる．しかしこの場合，計算が RealityKit 内部で完結しており，モデルを変更するなどの自由度が低い．そこで，モーションキャプチャした結果のデータである，人体スケルトンのジョイントの位置・姿勢を読み取り，それを用いて自作の 3D モデルを制御した．ある程度の制御に成功したが，一部に不具合が残った．

結論として，ARkit におけるモーションキャプチャ機能は，iOS 端末のみで行える上に，少ないソースコードでアプリケーションの開発を行える為 AR やモーションキャプチャの初学には最適なツールである事が理解できた．

目次

1章 序論

AR は「Augmented Reality（拡張現実）」の略称である．現実を拡張するという意味の言葉ではあるが，研究・実用化されているものの多くは，カメラで撮影した風景や室内の画像に対して，その内容にふさわしい情報を追加するものである．さらに限定すれば，写真画像に対して，違和感なく 3DCG 画像を重畳するようなものが多い．

AR 技術は様々な場面で利用されている．身近な例を挙げると，家具量販店 IKEA が Apple 社 と共同で開発した「IKEA Place」では，IKEA の商品の 3 次元 CG 画像を，自宅やオフィスをスマホで撮影した映像に，実寸大で重畳表示させる事ができ，その部屋に商品を置いた時の様子を確認できる(図 1.1)．オンラインでの購入の際に発生する，サイズ感がわからない，写真だけでは見えない部分がある等の問題を解決してくれる為，最近のステイホームという生活様式にも合っており，うまく活用されている．

図 1.1 IKEA Place の使用例

他にも，「Pokémon Go」(図 1.2)は，AR 技術が浸透していなかった時代に一世を風靡した AR ゲームである．「Pokémon Go」では，我々が生活している世界にポケモンが存在しているように感じられることに，多くの人が驚かされた．「Minecraft Earth」は，世界一売れているゲーム Minecraft をもとにしており，自分自身や世界中の人々が作成したサンドボックス型の作成物（多くは建築物）を現実世界の上に投影

して楽しむことができる．このように，AR 技術を用いて，現実世界とコンピュータが生成するコンテンツをリンクさせながら楽しむゲームが登場し，人気を博している．

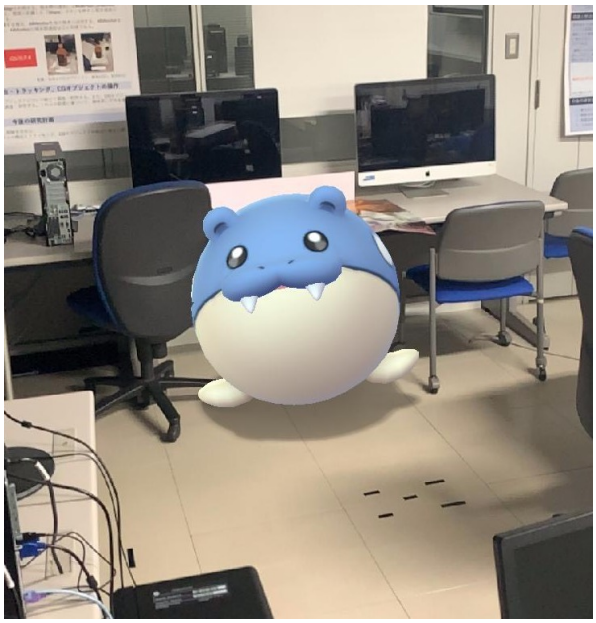

図 1.2 Pokémon Go の利用例

例にあげたアプリケーションは，スマートフォンやタブレット端末上で動作するものであるが，これら以外の AR デバイスも登場している．Google 社が開発した「Google Glass」やその後継機の「Glass Enterprise Edition 2」(図 1.3)は，メガネのレンズ上に資料等を表示させることで，仕事をアシストする事を目的としたスマートグラス型の AR デバイスである．マイクロソフト社が開発した「HoloLens」のようなヘッドマウントディスプレイ型のデバイスもある，他にも様々な形式で AR 技術は活用されている．

図 1.3 Glass Enterprise Edition2 の全体像
(https://www.google.com/glass/start/)[1]

AR 技術では，現実世界の情報をカメラなどのセンサデバイスを用いて取得し，コンピュータに認識させている．コンピュータに現実世界を認識させる方法がいくつかある．GPS や加速度センサを用いて自身とデバイスの位置を推定し，それに応じて情報を提示させるものを位置情報型と呼ぶ．「Pokémon Go」が代表的な位置情報型 AR である．次に，カメラで取得した画像情報から，そこに写っているものや空間を認識させ，情報を追加するものをビジョンベース型と呼ぶ．ビジョンベース型はマーカー型とマーカーレス型の二つの型がある．マーカー型は特定の形や柄を持つマーカー画像を予め登録しておき，カメラから入力した画像からそのマーカーを識別するという方式である．マーカーレス型は，特定のマーカー画像を用いることなく認識を行う方式である．マーカー型は明示的に認識対象を指示することができ，また，認識の精度が高いので AR システムの構築が用意である．一方，マーカーレス型は環境認識が難しくなるが，目障りなマーカーを使わないので，デザイン性が高くなる．

ARKit は，iOS デバイス用に Apple 社が提供しているアプリケーション開発フレームワークの一つである．これを用いることで AR アプリケーションの開発が容易になる．ARKit を用いて作成されたアプリケーションは，特別な場合を除き，iOS 端末だけで動作させることが可能である．また，Apple の公式ドキュメントで，ARKit に関する詳細な説明がされており，機能を体感できるサンプルプログラムも配布されている．これらは，AR 技術の初学者でも理解しやすい内容である，誰でも開発作業に手早く着手することが可能である．本研究では，ARKit 内の機能の一つであるモーションキャプチャに関して調査を行い，これを用いた 3DCG キャラクタの制御について研究した．

2 章 ARKit におけるモーションキャプチャ

2.1 モーションキャプチャ技術

モーションキャプチャ技術はスポーツ選手の動きを解析する場合や，CG キャラクタにアニメーションを付与する場合など，対象物の動作データの収集や解析に利用されている．

モーションキャプチャの方法は様々であるが，大きく分けて 3 つに分類する事ができる．光学式は人体などの対象物に反射マーカーを取り付け，赤外線照明を照射する．そして，複数のカメラで撮影した画像内のマーカー位置をもとにして，その三次元座標を計算する手法である．計測された位置の精度が高い反面，対象物の周囲を照明とカメラで囲うため広い空間が必要であり，計測できる箇所に制約が生じる事もある．また，赤外線照明や同期して動作するカメラといった装置が必要である．慣性式は，対象物に慣性センサを取り付け，そのセンサから得る加速度や角速度から，その動きを推定する手法である．この方法は光学式よりも精度の点で劣るが，対象物の動きに対する制約が少なく，比較的容易に行う事が可能である．ビデオ式はカメラで対象物を撮影し，対象物の画像からその立体形状を推定する方法である．対象物にマーカーやセンサも装着する必要がなく，場所に関する制約も少ないため，実行が容易である．ただし，通常のビデオカメラだけを用いる場合には，それほど精度が高くない．本研究で用いる ARkit のモーションキャプチャは一台のビデオカメラを使ったビデオ式である．

2.2 ARKit で使われるモーションキャプチャデバイス

ARKit は iOS 端末上で動作する AR アプリケーション開発用のフレームワークである．2017 年に ARKit1 が公開されて以降，毎年様々な機能が追加されてきた．そして，2019 年の WWDC(Worldwide Developers Conference)において，iOS 端末のカメラの進化と共に，モーションキャプチャの機能が追加された．この機能は A12 チップ以降が搭載されている iOS デバイスでのみ利用可能である。

2018 年型の iPad Pro は背面のビデオカメラで撮影した映像を解析して，モーションキャプチャを行なっている．この iPad の正面側には奥行き計測が可能な TrueDepth カメラが装着されているが，モーションキャプチャに使うことはできない．ビデオカメラで撮影した映像から人体部分を画像処理で認識し，その人体部分の画像に人体モデルをフィッティングさせることでモーションキャプチャを実現している．映像を使

ったモーションキャプチャであるから，ディスプレイに表示された人体に対しても動きをキャプチャすることが可能である．

2020 年型の iPad Pro は背面に LiDAR スキャナを装着しており（図 2.1），これを使ってモーションキャプチャを行う．従って，2020 年型の iPad Pro のモーションキャプチャは，2018 年型の iPad Pro によるモーションキャプチャよりも精度が高いと考えられる．本研究では 2018 年型の iPad Pro を利用した．

図 2.1 2020 年型 iPad pro の背面カメラ部分

2.3 ARKit におけるモーションキャプチャのプログラム

図 2.2 に Apple が配布しているモーションキャプチャを行うサンプルプログラム「Capturing Body Motion in 3D」の一画面を示す．画面右がキャプチャした人間の姿勢，左がその姿勢を表現した 3DCG モデルである．

図 2.2 公開されている配布プログラムの実行例
(https://developer.apple.com/videos/play/wwdc2019/607/)[3]

姿勢や動作を制御可能な 3DCG モデルは，姿勢を表現するスケルトンと，スケルトンを覆うメッシュから構成されている．これを使ってアニメーションを行う場合，スケルトンだけを変形させる．スケルトンは棒がジョイント（関節）で接続した構造になっている．個々のジョイントの位置と角度を変化させることによってスケルトンを変形させる．スケルトンの変形に対応して自動的にメッシュが変形し，画面上に 3D モデルが描画される．

モーションキャプチャを行うためには，図 2.3 のように ARBodyTrackingConfigration を初期化し，それを AR セッションに渡して run する．

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

図 2.3 モーションキャプチャを行うコンフィギュレーションの設定と開始

モーションキャプチャを実行して人体が検出されると，ARBodyAnchor オブジェクトがアンカーのリストに追加される．ARBodyAnchor は ARSkeleton3D 型の skeleton プロパティを持ち，ここからトラッキング中の 3 次元骨格情報を取得できる．このスケルトンは図 2.4 のような形のもので，合計 91 個のジョイント(関節)を持つ．モーションキャプチャの結果，ジョイントの位置と角度が設定され，全体の姿勢が決まる．

各ジョイントには図 2.5 の名前がつけられている．これらの名前を使って，ジョイントの位置と角度を読み出すことができる．これらの値は読み出すことだけができ，ユーザプログラムから設定することはできない．

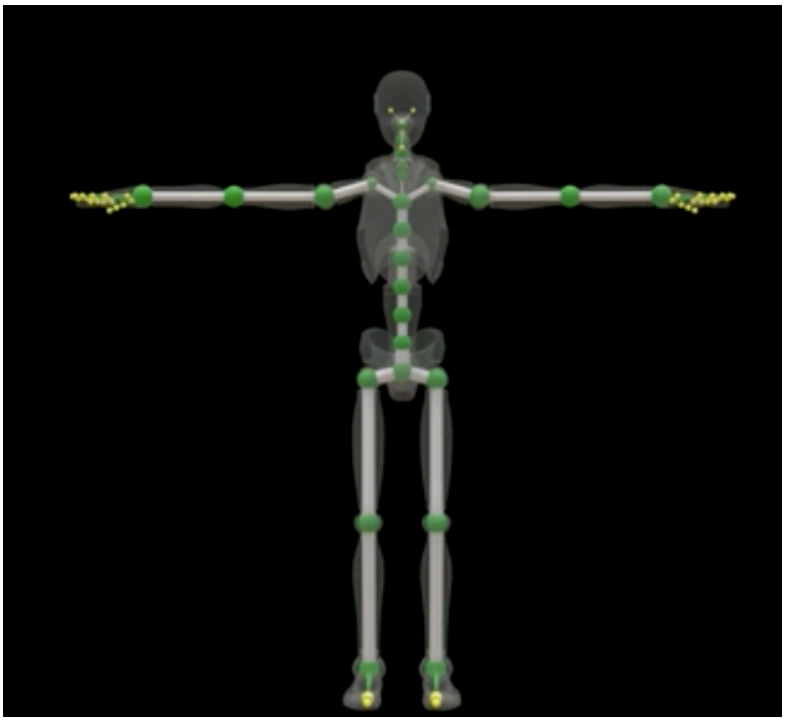

図 2.4 スケルトンの全体図
(https://developer.apple.com/videos/play/wwdc2019/607/)[3]

図 2.5 ジョイント名の一覧
(https://developer.apple.com/videos/play/wwdc2019/607/)[3]

モーションキャプチャの結果を使って 3D モデルにアニメーションを与える方法には，RealityKit を使う方法と，SceneKit を使う方法がある．Apple のサンプルプログ

ラム[6]は RealityKit を使っており，この場合，BodyTrackedEntity オブジェクトに適切な構造の 3D モデルファイル（図 2.4 のような構造で，図 2.5 のネーミング規則に合致するモデルで，usdz 形式のファイル）を読み込ませるだけで良い．一般の 3D モデルをアニメーションさせる場合は，スケルトン付きの 3D モデルファイル（usdz 形式）を読み込み，そのジョイントをモーションキャプチャの結果で制御することで可能になる．

3 章 モーションキャプチャを使った 3D モデルの制御

ここでは，ARKit のモーションキャプチャ機能で直接制御できる 3DCG モデルを構築する．本実験は以下の開発環境で実施した．

開発環境:

MacBook Air (13-inch, Early 2015)

MacOS Catalina version 10.15

Xcode version2.3 beta

Maya 2018

プログラミング言語:

swift

使用端末(iOS デバイス):

iPad pro(第 2 世代) システムバージョン 14.2

3.1 モーションキャプチャで制御可能な 3DCG モデル

ARKit のモーションキャプチャ機能で直接制御できる 3DCG モデルは，以下の性質が必要である[4].

・91 個のジョイントが図 2.5 の名称を持っていること．

・各ジョイントが図 2.4 の階層構造(親子関係)で接続されていること．

・3DCG モデルの初期状態が T ポーズ（図 2.4 のポーズ）であること．

・スケルトンの主要なジョイントがメッシュと適切にバインドされていること．

この条件のいずれかが満たされていない場合，予期せぬ動作を行うことや，全く機能しない可能性がある．

3.2 モデル作成の手順

Apple の公式サイトから「biped_robot.zip」をダウンロード，展開しファイル内に存在する「biped_robot.fbx」を Maya にインポートする．インポートすると図 3.1 の T ポーズで静止する 3D モデルが表示される．

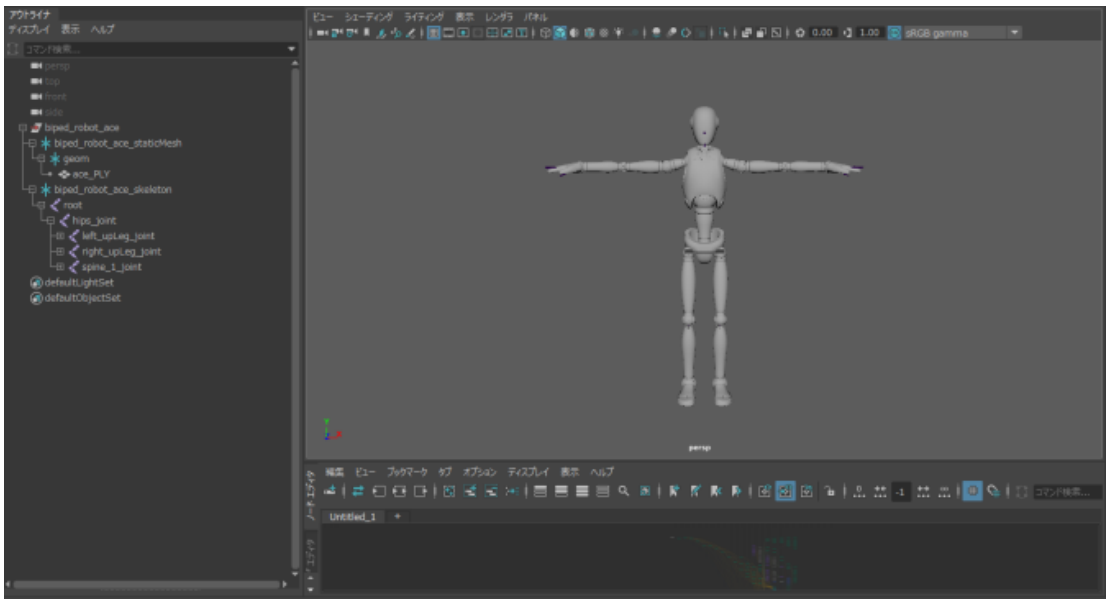

図 3.1　ダウンロードした 3D モデルの全体像

図 3.1 左側のアウトライナ（画面左側の縦長の領域）にこのファイルを構成している要素が表示されており，biped_robot_ace の下にスケルトンとメッシュが存在している事がわかる．このスケルトンを基にして 3D モデルを作る際，メッシュは不必要であるから，スケルトンとのバインドを解除し削除する．そうすると，スケルトンのみが画面に表示された状態になる(図 3.2).

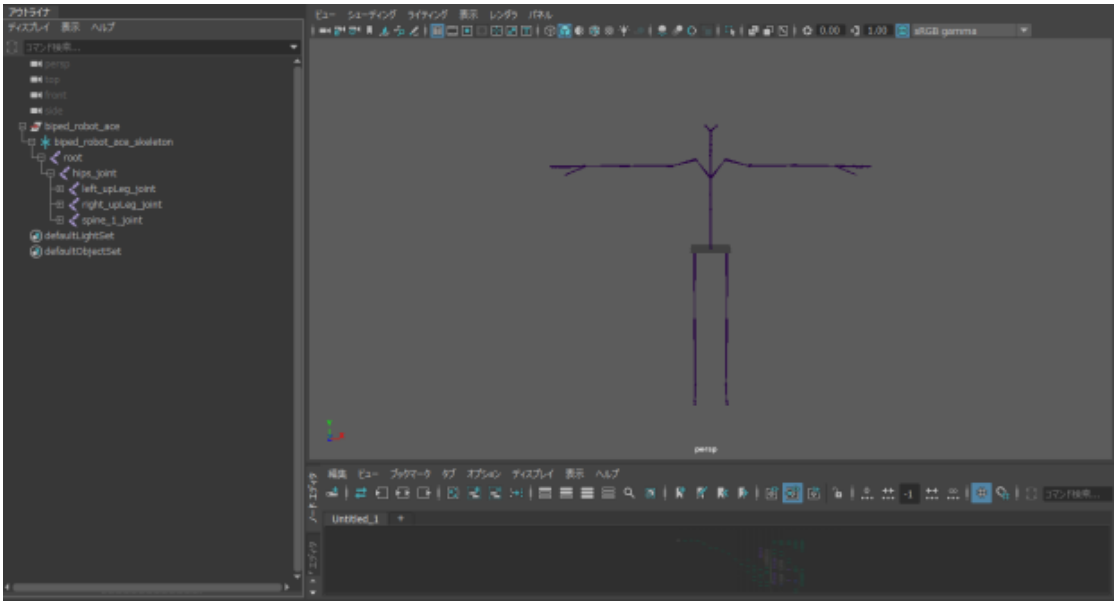

図 3.2　Maya 上でのスケルトンだけが表示された状態

ここに新たなメッシュをインポートする．今回は，図 3.1 の 3D モデルに近い二足歩行の人型メッシュを，Maya のメッシュ・ライブラリからインポートした(図 3.3)．

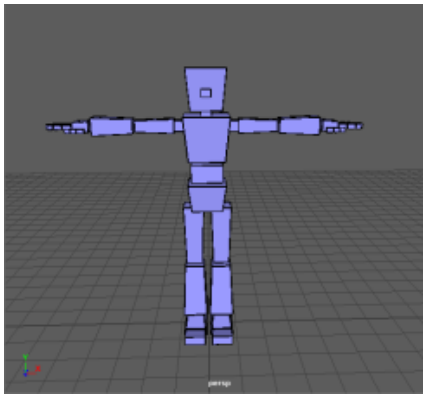

図 3.3 新たにインポートするメッシュ

インポートしたメッシュに対して，biped_robot から取り出したスケルトンをバインドするために，次のように調整した．まずスケルトンと縮尺を合わせ，腰や膝といった主要なジョイントの位置を重ねた．次に，指先や足といった細かな場所の修正をおこなった．最後にメッシュ全体とスケルトンの root を選択し，バインドを行なった．こうして完成した 3D モデルを.fbx 形式でエクスポートし，iOS 端末に送信する．最後に，Apple が提供している，ファイル形式を.usdz に変更するアプリケーション reality converter(図 3.4)を用いて，3D モデルを.usdz 形式に変換する事で，iPad の AR アプリケーションで動作する 3D モデルを作成した[5]．

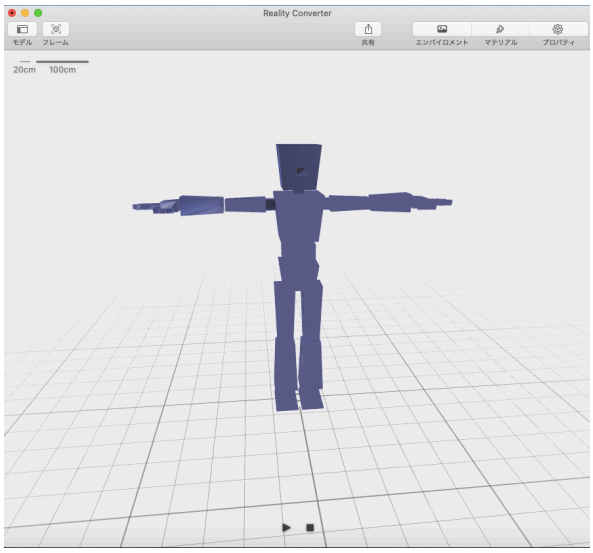

図 3.4 reality converter で usdz 形式に変換した 3D モデル

3.3　サンプルプログラムでの検証

Apple のサンプルプロジェクト「Capturing Body Motion in 3D」[6]をダウンロードし，その内容を確認した(図 3.5).

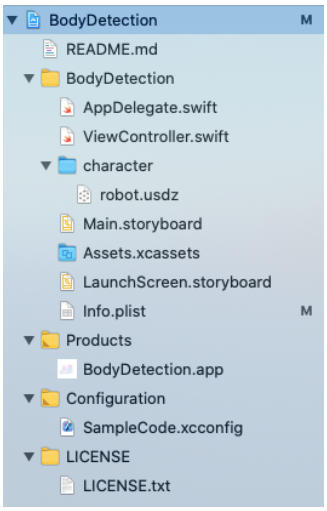

図 3.5 Capturing Body Motion in 3D のプロジェクト内容

この中の ViewController.swift に具体的な処理が書かれている．図 3.6 に，モーションションキャプチャの初期化を行う部分を示す．1 行目の BodyTrackedEntity オブジェクトはモーションキャプチャにしたがって動作する 3D モデルのエンティティであり，このプログラムの先で，モーションキャプチャで制御可能な 3D モデルファイルを読み込ませる．characterOffset は 3D モデルの表示位置を設定する．AnchorEntity はカメラ画像に重畳表示させる CG コンテンツをシーンに加えていく場合のルートになるアンカーエンティティである．ARBodyTrackingConfiguration()で，モーションキャプチャを行うコンフィグレーションを初期化し，arView.session.run で AR セッションを開始する．

```
var character: BodyTrackedEntity?
let characterOffset: SIMD3<Float> = [-1.0, 0, 0]
let characterAnchor = AnchorEntity()
let configuration = ARBodyTrackingConfiguration()
arView.session.run(configuration)
arView.scene.addAnchor(characterAnchor)
```

図 3.6 トラッキングとカプセル化のためのコードセグメント

このサンプルプログラムを iPad Pro で実行し，ディスプレイに表示した人物のモーションをキャプチャした(図 3.7)．このサンプルプログラムでは，実際に映る人物の左隣に 3D モデルが現れる様に設定されていた為，撮影した人物の左側に動きを模倣し

ている 3D モデルが描画された．この 3D モデルのファイルは，サンプルプログラムのプロジェクトに存在した robot.usdz である．

図 3.7　撮影した人体と動きを模倣する 3D モデル

次に，このモデルを 3.2 で作成したモデルに変更して動作を確認した．作成した 3D モデルをプロジェクト内の「character」フォルダに移動し，ソースコード内で指定されている，「named: robot」の robot を使用するモデルファイル名に変更することで，モデルの変更が可能になる。その結果を図 3.8 に示す．図 3.7 と同様の動作が確認できた為，今回作成したモデルは基準を満たしているものであると考えた．

図 3.8　モデルを変更した場合の動作

3.4 モーションデータの取得と生成したモデルへの適用

3.3 で述べたサンプルプログラムの方法は，モーションキャプチャの人体モデルと互換性のある 3D モデルを自動的に制御する方法である．この場合，ジョインの位置・姿勢のデータを直接制御する必要はなかった．次に，取得したモーションデータから，個々のジョイントの位置と姿勢を読み取り，それを自作したモデルに適用する方法を説明する．

図 3.6 のコードセグメントでは，変数 character が BodyTrackedEntity クラスのオブジェクトとして宣言されている．モーションキャプチャの結果の位置・姿勢のデータは character の 91 個のジョイントのデータとして読み取ることができる．例えば姿勢を示す回転の成分は「character?.jointTransforms[].rotation」で取得する事ができる．例えば，jointTransforms[0]は hips_joint（腰）の位置・姿勢を示す．このようにして読み取ったモーションキャプチャの姿勢に関する数値（３次元の回転を表す４×４の行列）を，自作した 3D モデルの対応するジョイントの姿勢に代入した．このようにして制御する自作の 3D モデルをモデル２と呼ぶことにする．3.3 の方法で制御されるモデルをモデル１と呼ぶ．

図４.２で，モデル１を画面左側に，モデル２を右側に表示する．人体は T 型のポーズをとっている．左側のモデル１は，概ね，T 型のポーズである．右側のモデル２は，左腕は T 型になっているが，右腕は垂れ下がっている．人体が動作を続けると，モデル１は正しい動作を続けた．モデル２の右腕は垂れ下がった状態であったが，それ以外の部位は概ね，動作した．モデル２は，サンプルプログラムで動作させた時は，正しく動作したのであるが，動作の制御を自分でプログラムした時には，正しく動作させることができなかった．これは，動作制御の方法に間違いが発生していることが原因であるが，それを解明することはできなかった．

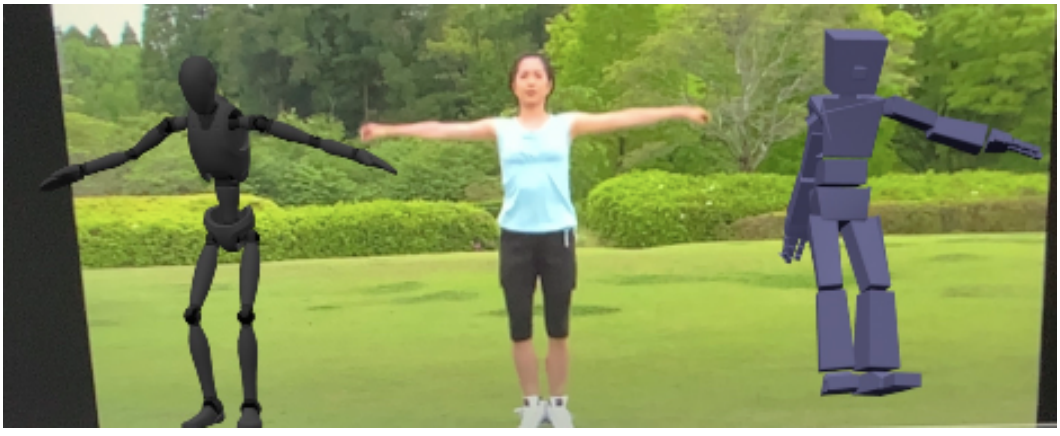

図 4.2 静止状態のモデルの比較

3.5 奥行き方向の認識性能

同じ格好をしているディスプレイ画面に映った人体と，現実の人体を比較した際，その変化は目視できる範囲では確認することができない。手を横に動かした場合や，上にあげた場合は当然の如く，のけぞる様な行動の際にもその変化はほとんど見受けられず，どちらともに正確に捉えることは不可能であった。このことから，今回のモーションキャプチャは映像ベースなので，ディスプレイ画面の人体など，奥行きがないものでも動作する上に，奥行き方向の動きを伴う動作の認識性能が低いことが分かる。その一部として、人体を正面からではなく横側からや、斜めから認識した場合や、体の捻り、腕や足の交差や重なりが発生した場合においても認識のミスが発生している。これはカスタムしたモデルだけでなく、配布されているモデルでも発生している事から、モデルでの問題ではなく現状の ARKit でのモーションキャプチャの限界点を表していると考えられる。

4 章　結論

本論文では，ARkit によるモーションキャプチャ機能，サンプルプロジェクトの調査及び，それらの機能を用いたデモアプリケーションを作成した．その結果，ARkit はモーションキャプチャという面において，その手軽さに見合わぬ有用性と拡張性がある事が理解できた．複雑な動きや，高速な動きを認識する際には誤動作も見られたが，カメラ一つで場所を選ばずに行えるという点と，モーションキャプチャを用いたアプリケーションの開発を，少ないソースコードで行う事ができるという点は他のツールには無い，ARkit の優位性であることが理解できた．しかし，AR やモーションキャプチャに関するアプリケーション作成の第一歩目としては，最適なフレームワークであるといえるが，より深く学ぶ際には，複雑な計算や人体の認識，モデルとの同期等，ARkit 内の関数で完結している場合が多く，知識を深めるといった際には向いていないと考えられる．また，スポーツの競技シーンや，映画の撮影といった詳細な動きの認識が必要な際は，iOS 端末で行うという仕様上，精度の面で難しいとも考えられる．しかし，この様な事柄を持ってしても，ARkit の持つ機能は簡易的なアプリケーションの開発や，短時間での開発に非常に効果的であるといえる．

参考文献

[1] Google Glass, ホームページの商品紹介画面, https://www.google.com/glass/start/

[2] Apple, iPad pro ショップページ, https://www.apple.com/jp/ipad-pro/

[3] WWDC 2019 session 607: Bringing People into AR, session 607: Bringing People into AR.

[4] https://developer.apple.com/documentation/arkit/validating_a_model_for_motion_capture

[5] https://developer.apple.com/documentation/arkit/rigging_a_model_for_motion_capture

[6] https://developer.apple.com/documentation/arkit/capturing_body_motion_in_3d

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致します.