舞踊の簡易なモーションデータ化手法の提案

広瀬研究室
C1180896 小松幸太郎

概要

少子高齢化の影響により伝統舞踊の現場では継承が途絶えてしまうことが警鐘されている。この問題にモーションキャプチャによる舞踊のデジタルアーカイブによって解決しようとする取り組みがなされている。しかしモーションキャプチャシステムは非常に高額であり、手法も専門的であることからモーションキャプチャを使ったデジタルアーカイブは容易にはできない。本研究ではモーションキャプチャの費用の低減と簡易化を目的に動画像のみの入力で動作取得を行えるモーションキャプチャシステムを構築した。結果として既存のシステムと比べて精度は落ちるが既存のソフトウェアで利用できるようなモーションデータを作成することができた。またモーションキャプチャにおける費用の低減や操作の簡易化を行うことができた。しかし一定の条件では動作取得が失敗する例があった。今後の展望としては動作取得の失敗する原因の究明と改善、またモーションデータの活用を検討している。(416 文字)

目次

第 1 章

はじめに

本章では研究の背景と目的を説明する。

1.1 背景

現在多くの民俗芸能が少子高齢化の影響により担い手不足が問題視され、様々な解決手段が用いられている [1]。その解決手段の一つとしてモーションキャプチャを使ったデジタルアーカイブが利用されてきた [2]。東北公益文科大学 (以下「本学」という) でもモーションキャプチャを使った伝統舞踊のデジタルアーカイブ事業が行われている。しかしながらモーションキャプチャ装置は非常に高額であり、地域団体が利用できる金額ではない。例としては本学のデジタルアーカイブ事業のモーションキャプチャでは Xsens を利用しており、2020 年時点でハードウェア、ソフトウェア合わせて約 600 万円になる。また現状のデジタルアーカイブで用いられるモーションキャプチャを行うには専門的な知識や作業が必要とされモーションキャプチャの知識がない人が行うのは困難である。一方で近年機械学習の発展により、ビデオからモーションキャプチャデータを作成することが可能になった [3]。

1.2 目的

本研究では伝統舞踊のデジタルアーカイブにおいて既存の装置を使用するモーションキャプチャシステムから Web カメラを用いたモーションキャプチャシステムに置き換えることで、伝統舞踊のモーションキャプチャの費用の低減と簡易化を目的とする。

第 2 章

提案システム関連技術の説明

本章では提案するシステム関連の技術解説を行う。

2.1 モーションキャプチャ

モーションキャプチャとは時系列的に対象の関節の角度や位置を計測して動作を取得するシステムである。動作の取得する方法の違いによっていくつかの方式がある。

2.1.1 光学式

光学式モーションキャプチャとは体の各部位にマーカを取り付け、複数のカメラからマーカの位置を読み取ることで対象の動作を計測する手法である。読み取ったマーカの位置情報を人体のモデルなどに対応付けることで、各部位の位置や関節の角度を取得する。光学式では計測前に後述するキャリブレーションという作業が必要となる。

2.1.2 慣性式

慣性式モーションキャプチャとは演者にセンサを取り付け、そこから角度、加速度、地磁気などを取得してその情報から動作を計測する手法である。慣性式も光学式と同様に計測前にキャリブレーションが必要となる。本学のデジタルアーカイブ事業で用いられた Xsens も慣性式モーションキャプチャである。

2.1.3 ビデオ式

ビデオ式モーションキャプチャとは動画像から色の分割、輪郭の抽出などを行うことで動作を計測する手法である。システムの構成として動画像を取得するカメラと画像を解析するコンピュータのみとなっている。

2.2 BVH

標準的なモーションキャプチャデータ形式であり、骨格モデルの各関節の情報と各関節同士の関係を階層状に表現し記述している HIERARCHY 部と 1 フレームごとの動作を記述した MOTION 部で構成されている。図 2.1 は HIERARCHY 部の例、図 2.2 は MOTION 部の例である。HIERARCHY 部で最初に記述され、基準となる関節 ROOT のみ座標情報と角度情報を持つ。他の関節は角度情報のみ持つ。本研究で扱うシステム

```
HIERARCHY
ROOT J_Bip_C_Hips
{
    OFFSET  0.00    0.00    0.00
    CHANNELS 6 Xposition Yposition Zposition Zrotation Xrotation
    JOINT J_Bip_C_Spine
    {
        OFFSET   0.000000    0.140529    0.000739
        CHANNELS 3 Zrotation Xrotation Yrotation
        JOINT J_Bip_C_Chest
        {
            OFFSET   0.000000    0.078983   -0.004420
            CHANNELS 3 Zrotation Xrotation Yrotation
            JOINT J_Bip_C_UpperChest
            {
                OFFSET  -0.000000    0.077361   -0.003995
                CHANNELS 3 Zrotation Xrotation Yrotation
                JOINT J_Bip_C_Neck
```

図 2.1　BVH データの HIERARCHY 部の例

```
MOTION
Frames:    449
Frame Time: 0.01666667
 0.001996    0.961243    0.008008   -0.000003   -0.000002   -0.000000    0.000000
 0.879439    0.964339   -1.472245   -2.278127   -10.412860   9.733822   -3.258128
 0.852916    0.960391   -1.469311   -1.778310   -10.720890   7.723927   -2.918087
 0.846592    0.959489   -1.468555   -1.678045   -10.786240   7.332691   -2.822491
 0.844231    0.959155   -1.468269   -1.641623   -10.810320   7.190276   -2.786514
 0.842237    0.958875   -1.468027   -1.611010   -10.830700   7.070432   -2.756290
 0.839901    0.958549   -1.467736   -1.578421   -10.853860   6.940292   -2.718186
 0.835233    0.957903   -1.467144   -1.516526   -10.899540   6.690174   -2.640154
 0.832281    0.957498   -1.466765   -1.477841   -10.928400   6.533203   -2.591156
 0.830146    0.957206   -1.466488   -1.451586   -10.949300   6.423050   -2.554659
 0.828187    0.956939   -1.466227   -1.430760   -10.968540   6.328084   -2.518716
```

図 2.2　BVH データの MOTION 部の例

で出力されるデータも BVH 形式で出力される。

2.3　キャリブレーション

モーションキャプチャを始める前に原点の位置や各骨の構造、各装置の設定値などを設定することである。モーションキャプチャの方式によってキャリブレーションの方法は異なる。

2.4　オープンソースソフトウェア

ソースコードが無償で公開されており、利用、改造、再配布が許可されているソフトウェアである。代表的なものに OS である Linux、各種プログラミング言語、後述する Blender などがある。

2.5 Unity

Unity Technologies が開発しているゲームエンジンである。PC、Web、スマートフォンなど様々なプラットフォームに対応した開発環境が用意されている。コンピュータゲームの他にも CG 映像作品の制作や、製品のデザイン設計などにも用いられている [4]。

2.6 Blender

3DCG アニメーションを製作するための統合開発環境である。オープンソースであり複数の OS 環境上で動作する。3DCG だけでなく 3D モデル、2D アニメーションの製作も行うことができる [5]。

2.7 ThreeDPoseUnityBarracuda

Unity 上で使用するソフトウェアであり、動画像から姿勢推定を行い、取得した動作を 3D モデルに付与するシステムである [6, 7]。複数のオブジェクトから構成されており、使用用途によって設定を変更する必要があるオブジェクトは BarracudaRunner 、 Main Texrure がある。また予め用意した動画像を使用する場合は Video Player の設定も変更する必要がある。設定が必要となる設定値を表 2.1 に示す。

2.8 BVHtools

Unity 上で使用するソフトウェアであり、3D モデルやスケルトンから BVH ファイルを作成するものである。図 2.2 は BVHtools の設定画面である。設定を必要とする設定値を表 2.3 に示す。

上記の要素を設定することで BVH ファイルの作成をすることができる。Unity で Scene を再生すると同時に 3D モデルの動作を BVH 形式で取得していく。Frame Number の数値が増加していけば動作取得が成功していることを確認できる。その後 Save motion to BVH file をクリックすることで設定したディレクトリ、ファイル名でファイルが保存される [8]。

表 2.1 ThreeDPoseUnityBarracuda の主要設定値

BarracudaRunner の設定値	
NN Model	機械学習に用いるニューラルネットワークモデル
V Nect Model	動作を付与する 3D モデル
Main Texrure の設定値	
Use Web Cam	動画像の取得方法の選択
Video Player の設定値	
Video Clip	使用する既存の動画像
Play Back Speed	動画像の再生速度

表 2.2　BVHtools の主要設定値

設定値名	内容
Frame Rate	一秒間に取得する動作数
Directory	保存するディレクトリパス
Filename	作成する BVH ファイルに付けるファイル名
Target Avatar	取得したい 3D モデル
Root Bone	取得対象である 3D モデルの位置情報を決める骨

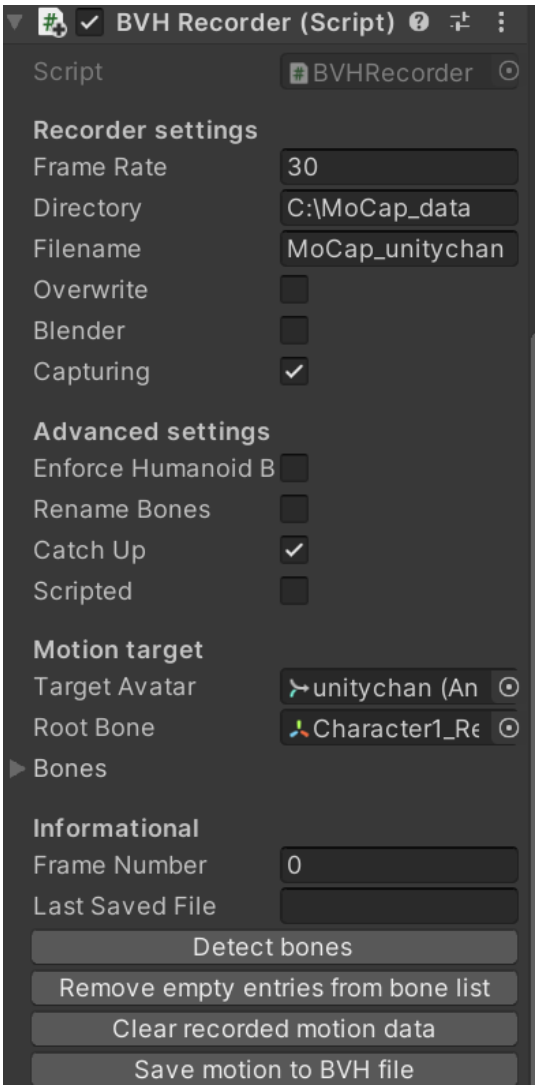

図 2.3　BVHtools の設定画面

第 3 章

先行研究と既存のシステム

本章ではこれまでに行われてきた研究と既存のモーションキャプチャシステムを取り上げている。

3.1　モーションキャプチャを用いた研究

モーションキャプチャに関する先行研究を述べる。

3.1.1　伝統舞踊のデジタルアーカイブの取り組み

この研究では舞踊を光学式モーションキャプチャで計測し、そこから得られるデータから舞踊動作や踊り手の識別、特徴的な部分の抽出を行っている [9]。そこから動作類似性に基づく類似検索に関する研究を行っている。またモーションキャプチャデータから教育用 CG コンテンツの作成、VR 環境での舞踊のコラボレーションを行う取り組みもなされている。

3.1.2　デジタル技術を用いた黒川能の継承の取り組み

伝統舞踊において地方の過疎化、少子高齢化に伴う後継者不足が問題となっており、失伝が避けられない状況になる。この研究ではこういった問題対して黒川能を題材にしデジタルアーカイブという手法で解決しようと試みた [2]。慣性式のモーションキャプチャを利用してデジタルデータとして保存し、MMD を使い CG アニメーションとして再現する方法が提案されている。

3.1.3　カメラとカラーマーカを利用したシステムの提案

この研究では個人利用を想定した取り扱いの容易なモーションキャプチャシステムを提案している [10]。システムは荷台の Web カメラと簡易カラーマーカを装着したユーザの上半身を撮影する。そこから各マーカの三次元座標を復元し、アバターの腕姿勢を設定する。各処理の制度の検証を行い、一般ユーザの利用に耐えうる制度であることを確認している。

3.1.4　深度カメラを利用したシステムの提案

この研究では新規に 3DCG 関連の技術に触れる足掛かりとして比較的安価な構成でモーションキャプチャを行い、リアルタイムで 3D キャラクタを操作できるシステムを開発している [3]。システムの構成は安価な深

度カメラとオープンソースの機械学習ライブラリの TensorFlow を使用することでコストを 15 万円以内に収めながら全身のモーションキャプチャを実現した。

3.2　既存のシステム

既存のシステムについて述べる。

3.2.1　Xsens MVN

Xsens 社が開発している慣性式モーションキャプチャシステムである。ハードウェアには Awinda と Link の二つがあり、 Awinda は小型のセンサを体の規定位置に設置することで、Link は予め装着されたスーツを着ることでそれぞれモーションキャプチャを行う。ハードウェア、ソフトウェアを合わせた価格は 2020 年時点で約 600 万円である [11]。

3.2.2　OptiTrack

アキュイティー社が開発している光学式モーションキャプチャシステムである。カメラ、スーツ、マーカ、ソフトウェア、各種ケーブルなど様々な製品で構成されている。モーションキャプチャを行うために掛かる費用は 2021 年時点で約 250 万円である [12][13]。

第 4 章

課題と改善案

本章では先行研究及び既存のシステムから伝統舞踊のデジタルアーカイブに対する課題とそれに対する改善案を述べる。

4.1 課題

3.1.1、3.1.2 はそれぞれ光学式、慣性式のモーションキャプチャシステムを使用している。これらはモーションキャプチャを行う前にキャリブレーションを行う必要があり、これにはモーションキャプチャ等の専門的な知識が必要とされる。またそれぞれの方式の装置を扱うにも専門的な知識が必要とされる。しかし民族芸能の継承団体がそういった知識を有しているとは考えられない。また 3.2.2 ではモーションキャプチャを行うにあたって構成する装置が多数あることが分かる。そのためシステムは複雑な構成をしていると言える。このことからも専門的な知識が要求されることが分かる。また 3.2.1、3.2.2 より既存のモーションキャプチャシステムを利用するには高額な費用が掛かることが分かる。このことから民族芸能の継承団体がモーションキャプチャを用いようと考えても高額な費用により断念する可能性がある。以上のことから現状のモーションキャプチャシステムを用いた伝統舞踊のデジタルアーカイブの課題点として下記の二点を挙げる。

1. モーションキャプチャを行うための専門的な知識が必要である。
2. モーションキャプチャを行うために高額な費用が掛かる。

以後 1 を「課題 1」、2 を「課題 2」とする。

4.2 改善案

課題 1 の改善案としてビデオ式モーションキャプチャを使用することとする。ビデオ式モーションキャプチャは光学式、慣性式と比べてキャリブレーションを行わないという特徴がある。またカメラと PC のみで動くためシンプルな構成で利用することができる。そのため操作が簡易でありながらもモーションキャプチャをおこなえる。以上のことから課題 1 の改善案としてビデオ式モーションキャプチャを使用することを挙げる。次に課題 2 の改善案としてオープンソースソフトウェアを利用してシステムを構築することとする。オープンソースソフトウェアは無償での使用を行うことができ、それによってモーションキャプチャの費用を抑えることができると考えられる。加えてシステムの構成が他モーションキャプチャ方式より少ないことから費用を抑えることができるため課題 1 の改善案も費用を抑えることができると考えられる。以上のことから課題 2 の改

善案としてオープンソースソフトウェアを利用してシステムを構築すること、ビデオ式モーションキャプチャを使用することを挙げる。

第 5 章

提案システム

本章では前章で取り上げた改善案を踏まえたシステムの提案を行う。

5.1　要件定義

以下に必要となる要件を示す。

- モーションキャプチャ方式はビデオ式とする。
- オープンソフトウェアを使用する。
- モーションデータを出力する。
- モーションデータは BVH 形式とする。

5.2　システム設計

5.1 を元にシステム設計を行う。

5.2.1　システムの基本設計

本システムのイメージ図を 5.1 に示す。まず演者の動作をカメラを用いて撮影し動画像とする。続いて撮影した動画像に対して姿勢推定を行い、モーションキャプチャを行う。最後にモーションデータとして BVH データを出力する。

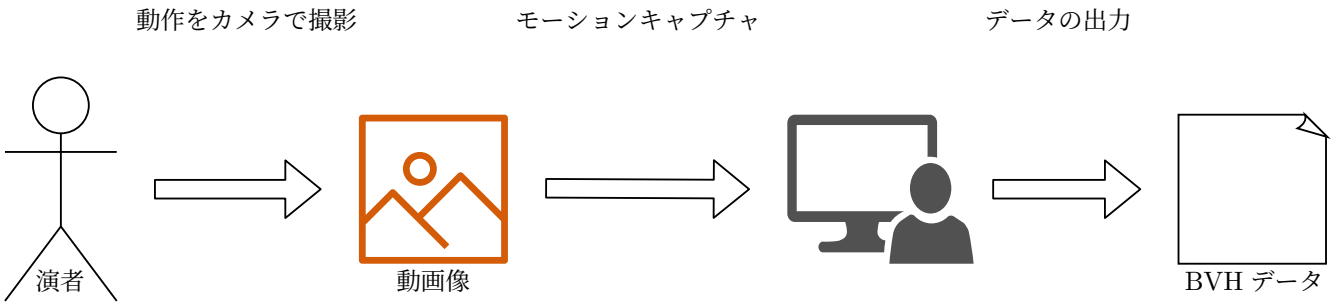

図 5.1　イメージ図

5.3　システム詳細設計

本システムはハードウェアとしてノート PC を使用し、ソフトウェアとして Unity、ThreeDPoseUnityBarracuda、BVHtools、3D モデルを使用する。開発言語は C#を使用する。

5.3.1　動作をカメラで撮影

動作の撮影はノート PC の Web カメラ、外部カメラを用いる。また現存の動画像も入力として使用できるものとする。

5.3.2　モーションキャプチャ

モーションキャプチャは ThreeDPoseUnityBarracuda を用いて行う。この時に取得した動作を随時 3D モデルに付与していく。

5.3.3　データの出力

モーションデータの出力は BVHtools を用いて行う。出力されるモーションデータの形式は BVH 形式とし、フレームレート[*1]は 30fps とする。

5.3.4　その他

システムの簡易化のために機能の追加を行う。

カメラの操作

モーションキャプチャ中のの 3D モデルの動きの確認のためにカメラの操作を行えるようにする。カメラは正面に対して前後左右、上下、水平方向への回転を行えるようにする。操作に割り当てるキーは前後左右に A、S、D、W を、上下に I、M を、水平方向の回転に J、K を用いる。キーの操作の取得には Unity スクリプトの Input クラスの GetKey 関数を用いる。カメラの移動、回転にはそれぞれ Unity スクリプトの transform クラスの Translate 関数、Rotate 関数を用いる。作成したプログラムを以下に示す。

[*1] 単位時間に動画が何枚の画像で構成されているかを示す数値である。この数値が高いほど動画内の動きが滑らかに見えるが、それに伴いデータ量も増える。単位時間を 1 秒としたときのフレームレートの単位を fps と表す。

カメラ操作のプログラム

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class CameraMove : MonoBehaviour
{
    float up = 0.1f;
    float front = 0.1f;
    float right = 0.1f;
    float turn = 2f;
    Dictionary<string, bool> move = new Dictionary<string, bool>
    {
        {"up", false},
        {"down", false},
        {"front", false },
        {"back", false },
        {"right", false },
        {"left", false },
        {"turnR",false},
        {"turnL",false}
    };
    void Update()
    {
        move["up"] = Input.GetKey(KeyCode.I);
        move["down"] = Input.GetKey(KeyCode.M);
        move["front"] = Input.GetKey(KeyCode.W);
        move["back"] = Input.GetKey(KeyCode.S);
        move["right"] = Input.GetKey(KeyCode.D);
        move["left"] = Input.GetKey(KeyCode.A);
        move["turnR"] = Input.GetKey(KeyCode.K);
        move["turnL"] = Input.GetKey(KeyCode.J);
    }
    void FixedUpdate()
    {
        if (move["up"]){transform.Translate(0f, up, 0f);}
        if (move["down"]){transform.Translate(0f, -up, 0);}
        if (move["front"]){transform.Translate(0f, 0f, front);}
        if (move["back"]){transform.Translate(0f, 0f, -front);}
        if (move["right"]){transform.Translate(right, 0f, 0f);}
        if (move["left"]){transform.Translate(-right, 0f, 0f);}
        if (move["turnR"]){transform.Rotate(0f, turn, 0f);}
        if (move["turnL"]){transform.Rotate(0f, -turn, 0f);}
    }
}
```

キーボード入力の追加

データ出力がマウスでのクリック操作のみだったが簡易化のためキーボード入力操作をできるようにする。システムの稼働中にスペースキーを押すことでデータの出力、及びシステムの終了を行う。キーの操作の取得には Unity スクリプトの Input クラスの GetKey 関数を用いる。作成したプログラムを以下に示す。

キーボード操作を追加するプログラム

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEditor;
using UnityEngine.SceneManagement;
using System.IO;
public class save : MonoBehaviour
{
    GameObject ob;

    void Start()
    {
        ob = GameObject.Find("kotaroid");
    }

    void Update()
    {
        if((Input.GetKey(KeyCode.Space))){
            ob.GetComponent<BVHRecorder>().saveBVH();
            UnityEditor.EditorApplication.isPlaying = false;
        }
    }
}
```

5.4　システム構築

上記した設計を元にシステムを構築する。実装したシステムを図 5.2 に示す。システムの構築、実装は Unity 上で行う。Unity のバージョンは 2019.4.22.f1 を使用する。システムは上部の三角マークの再生ボタンをクリックすることで動作する。画面右側の BVH Recorder 内の Save motion to BVH file と書かれたボタンをクリック、またはキーボードのスペースボタンを押すことで動作のデータを出力することができる。

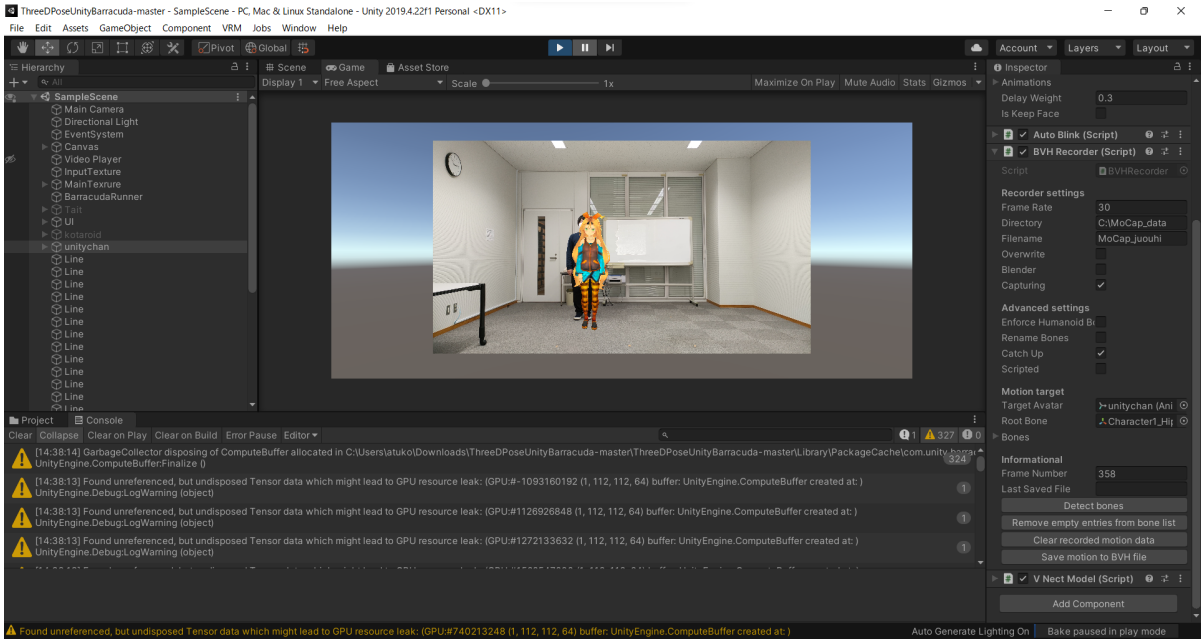
ThreeDPoseUnityBarracuda-master - SampleScene - PC, Mac & Linux Standalone - Unity 2019.4.22f1 Personal <DX11>
File Edit Assets GameObject Component VRM Jobs Window Help
Hierarchy
SampleScene
Main Camera
Directional Light
EventSystem
Canvas
Video Player
InputTexture
MainTexrure
BarracudaRunner
Tait
UI
kotaroid
unitychan
Scene
Game
Asset Store
Inspector
Animations
Delay Weight
Is Keep Face
Auto Blink (Script)
BVH Recorder (Script)
Recorder settings
Frame Rate
Directory
Filename
Overwrite
Blender
Capturing
Advanced settings
Rename Bones
Catch Up
Scripted
Motion target
Target Avatar
Root Bone
Bones
Informational
Frame Number
Last Saved File
Detect bones
Remove empty entries from bone list
Clear recorded motion data
Save motion to BVH file
V Nect Model (Script)
Add Component
Project
Console
Clear Collapse Clear on Play Clear on Build Error Pause Editor
Found unreferenced, but undisposed Tensor data which might lead to GPU resource leak: (GPU:#740213248 (1, 112, 112, 64) buffer: UnityEngine.ComputeBuffer created at:)
Auto Generate Lighting On
Bake paused in play mode

図 5.2　システムの画面

第 6 章

実験

実験として本システムを実際に運用することでどの程度モーションキャプチャシステムとして利用できるか、どのような問題点があるかを評価する。本実験で使用する機器の仕様を表 6.1 と表 6.2 にまとめる。

表 6.1　システム実行機の仕様表

名称	GF65-10SER-257JP
OS	Windows 11 home
CPU	Intel® Core™ i7-10750H
GPU	NVIDIA®GeForce™RTX 2060
主記憶	16GB

表 6.2　撮影機の仕様表

名称	oppo reno5 A
OS	ColorOS 11
CPU	Qualcomm®Snapdragon™765G
主記憶	6GB
最大画素数	約 6,400 万画素

6.1　本システムの運用実験

実際に本システムを運用しモーションデータを作成し、動作を再現できるか実験を行う。今回は外部カメラを使わず、本システムの実行機に付属している Web カメラを使用した。図 6.1 は本システムでモーションデータを作成する様子である。モーションデータの作成までの実験の手順を以下に示す。

1. ThreeDPoseUnityBarracuda、BVHtools の設定値を設定する。
2. 演者はカメラに全身が映る位置に立つ。
3. モーションキャプチャシステムを動かす。
4. 演者をシステムが認識して 3D モデルが演者の姿勢を模倣するまでしばらく待つ。
5. 3D モデルが演者の姿勢を模倣し始めたら計測したい動作を始める。
6. BVHtools 内の Save motion to BVH file ボタンを押す。

7. BVHtools の設定値で定めたディレクトリにモーションデータがあることを確認する。

実験の結果、7 秒のほどしてシステムが演者を視認し、動作に合わせて 3D モデルが動作した。その後 3D モデルの姿勢が乱れることなく安定的に動作取得を行うことができた。しかし動画像に映っている動作でも取得できない場合があった。取得できない場合としてはカメラから前方 300cm 地点で水平方向に左右 77cm 以上動くと動作取得に失敗した。モーションデータは指定のディレクトリに保存されていた。

図 6.2 は作成した動作を再現した様子である。動作を再現するソフトウェアとして Unity を利用した。動作の再現の確認をできるかの実験の手順を以下に示す。

1. Blender を起動し、file ⇒ import ⇒ Motion Capture (bvh) から作成したモーションデータをインポートする。
2. モーションデータをインポートしたのちに file ⇒ Save as … から blender 形式で保存する。
3. Blender 形式にしたモーションデータを Unity 内にインポートする。
4. インポートしたモーションデータを選択し、 rig の情報の animation type を humannoid に変更する。
5. 変更したモーションデータの中から Scene と書かれたファイルを選択し、対象の 3D モデルの animator に設定する。
6. Unity の再生ボタンを押して 3D モデルの動作を再生させて動作を再現されているか確認する。

作成したモーションデータを blender 形式に変換した理由としては本システムで作成されるモーションデータ形式の BVH データは Unity で直接利用することは出来ず、Unity で動かせられる形式に変換する必要があったためである。今回は作成したモーションデータの変換を行うにあたってオープンソフトで誰でも利用することができることから Blender を採用した。実験の結果、3D モデルが地面と定義している位置から少し浮いた位置で動作は再現された。指先などの細かい動作は取得することが出来なかったが、腕、足、首など大きな動作は取得することができた。

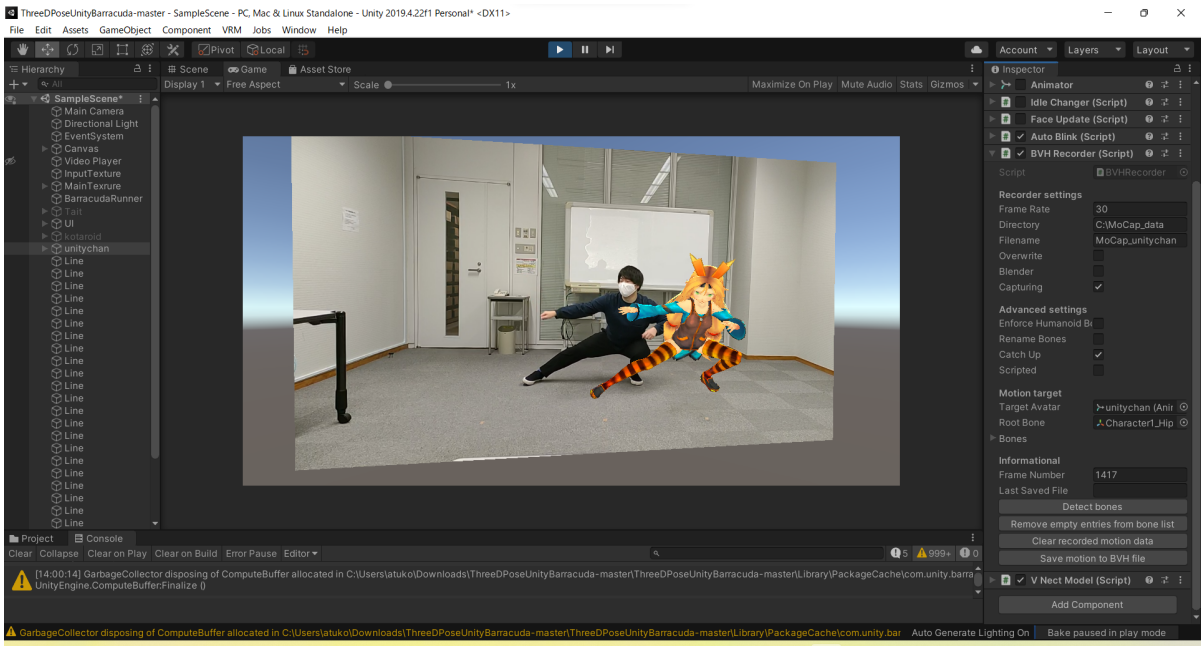

図 6.1　本システムの画面

図 6.2　動作再現

6.2　取得したモーションデータの座標情報の比較

ビデオ式モーションキャプチャは単独のカメラからの情報で動作を計測するため動作取得精度が他システムより不安定である。そこで本システムで取得したデータから垂直方向の移動量と水平方向の移動量を比較して違いはあるのか実験を行う。ここでのカメラ正面に対して垂直方向であることとする。またカメラ正面に対して平行方向であることとする。今回は撮影機を使用した。実験の手順は以下の通りである。

1. 演者が正方形の形に沿いながら地点 A,B,C,D,E(e は A と同地点) の順番でそれらに向けて移動する。
2. このとき演者は各地点で 5 秒停止する。
3. これらの動きを撮影機で撮影する。
4. 本システムで撮影した動画像から動作を計測し、モーションデータを作成する。
5. 作成したモーションデータから各地点の停止時の位置情報を取得する。
6. 各地点ごとの位置情報の差を求める。
7. 数値を比較する。

動作取得対象の移動経路及び距離を図 6.3 に示す。前回の運用実験の結果、本システムが動作を計測できる範囲がカメラ前方 300cm 地点で水平方向に左右に 77cm であると分かったため移動距離を右の計測限界範囲と左の計測限界範囲を足した 154cm に設定した。

各地点間のでの移動量を表 6.3 に示す。A-B 間、C-D 間の移動量を垂直方向の移動量、B-C 間、D-E 間の移動量を平行方向の移動量とする。縦の移動量と横の移動量を比べると本システムで取得された縦と横の移動量には 3.87 倍の差があることが分かる。

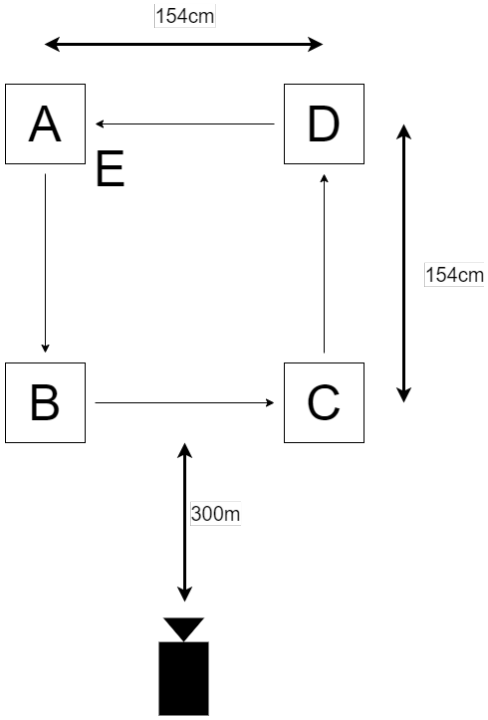

図 6.3　移動経路及び距離

表 6.3　各地点間での移動量の差

地点間名	移動量
A → B	0.325034
B → C	1.259002
C → D	0.344675
D → E	0.882234

第 7 章

実験の考察

本章ではこれまでに行った実験に対する考察について述べる。

7.1 本システムの運用実験の考察

本システムの運用実験では一定の範囲を外れると動作が取得できないことが分かった。これの原因としては ThreeDPoseUnityBarracuda が入力された動画像をトリミングしているためと推測する。それを行う理由としては画像のトリミングを行い画像サイズを縮小することで姿勢推定の時間を削減するため、または動画像の形式に関係なく ThreeDPoseUnityBarracuda で姿勢推定を行えるように動画像の形を均一にするため、と考える。動作取得が可能な水平方向の範囲はカメラ前方 3m 地点で左右に 77cm であることから、カメラの中心から水平に 14° の範囲であることが分かる。垂直方向の取得可能な画角の範囲は現状検証することができなかったため今後検証していくこととする。姿勢推定は完了するまで 7 秒ほどかかった。これが完了することでモーションキャプチャを行うことができるようになるため、この 7 秒を他モーションキャプチャシステムでのキャリブレーション時間と扱う。Xsens でのキャリブレーション時間は 30 分ほど掛かる。これと比べるとキャリブレーションの時間が非常に短縮されていると言える。動作再現では大まかな動作を再現することができることが分かった。胴体のひねりや腕などを前後に動かす動作も再現されていた。しかし体などによってカメラに対して隠れた部位の動作は再現することができなかった。これはビデオ式のモーションキャプチャシステムでは必ず起こる不具合である。複数のカメラを用いるなどの解決方法があるが装置の数を増やせば簡易性が失われることが考えられるため、構成を変えるかは検討が必要と考える。

7.2 取得したモーションデータの座標情報の比較の考察

取得したモーションデータの縦方向と横方向の移動量の比較では実際の移動量は縦方向と横方向で同じであったにも関わらず、取得したデータ上では横方向の移動量が縦方向の移動量に比べて 3.87 倍の差が生まれてしまっていた。こうなった原因としては奥行きのある動作取得がビデオ式では不利であることが考えられる。これを踏まえると本システムは移動を伴う動作よりも演者の移動が少なく腕や足、胴体などを動かす動作に対して適していると考えらえる。

第 8 章

考察

各実験の結果を踏まえてモーションキャプチャの費用の低減と簡易化がなされているか考察する。

8.1 モーションキャプチャとしての利用

実験からシステムから取得したモーションデータを Unity 上で再生することができたため正常なモーションデータを作成することできることが確認できた。しかし既存のシステムと比べて動作取得精度はかなり劣ったものとなっている。また横方向の移動量に比べて縦方向の移動量の取得が不得手であること、動作を取得可能な範囲が限定されていること、カメラの角度によっては動作が取得できないことを踏まえると取得する動作によっては十分に取得することができないと考えられる。

8.2 費用の低減

今回の運用実験でシステムに掛かった費用は約 13 万円だった。今回は Web カメラを使用したが、外部のカメラを使用することも可能である。その場合もフル HD 画質での画像でモーションキャプチャが行えため、高性能なカメラを使用する必要はないと考える。結果としては既存のシステムよりモーションキャプチャに掛かる費用を低減できたと言えると思われる。

8.3 モーションキャプチャの簡易化

モーションキャプチャを行うまでに掛かった時間を比較して考える。本システムは起動してから 7 秒ほどで動作取得対象の姿勢推定が行われ、そこからモーションキャプチャが行われ始める。従って本システムのモーションキャプチャを行うまでに掛かった時間は 7 秒とする。既存のシステムの例として Xsens では体の指定された各部位にセンサを取り付け、そこからキャリブレーションを行う。ここまでに掛かる時間は技術者の技量によって 30 分から 1 時間かかる。既存のシステムと比べてモーションキャプチャを始めるまでの時間が短縮されて扱い易くなったことがわかる。このことからモーションキャプチャの簡易化はできたと考えられる。

第 9 章

結論と今後の展望

本章では本研究の結論と今後の展望について述べる。

9.1 結論

本研究では Unity,ThreeDPoseUnityBarracuda,BVHtools を使い動画像のみでモーションキャプチャを行うシステムを構築した。本システムを利用することで既存のモーションキャプチャシステムと比べて安価で簡易にモーションデータを作成することができた。しかし動作種類や動画像を取得する環境によっては動作取得に失敗する場合があることが分かった。

9.2 今後の展望

今後の展望としては今回の実験で露呈した本システムの問題点の解明と改善を検討している。具体的には

- 垂直方向の動作取得可能範囲の検証
- 取得される垂直方向の移動量と平行方向の移動量の差の低減

が挙げられる。

また動作取得だけでなく、動作活用も検討している。案として Web 上で立体的な動作の投稿と閲覧を行える Web サービスを考えている。

参考文献

[1] 小川直之.“「民俗芸能」を継承する各地の取り組み — 文化遺産の世界”. https://www.isan-no-sekai.jp/report/7243, (参照 2021-12-8).

[2] 玉本英夫, 唐栄. “黒川能のディジタル化を通した民俗芸能の踊り継承の新手法”. 総合研究論集, p. 87, 2020.

[3] 三浦彰人. “深度カメラと姿勢推定モデルを用いた 3d キャラクターリアルタイムモーションシステムの開発”. 総合研究論集, p. 99, 2020.

[4] Unity Technologies. “unity real-time development platform — 3d, 2d vr & ar engine”. https://unity.com/, (参照 2020-10-12).

[5] Blender Foundation.“blender.org - home of the blender project - free and open 3d creation software”. https://www.blender.org, (参照 2020-10-12).

[6] Unity Technologies. “barracuda - unity マニュアル”. https://docs.unity3D.com/ja/2019.4/Manual/com.unity.barracuda.html, (参照 2021-12-5).

[7] 株式会社デジタルスタンダード. “digital-standard/threedposeunitybarracuda: Unity sample of 3dpose estimation using barracuda”. https://github.com/digital-standard/ThreeDPoseUnityBarracuda, (参照 2020-10-12).

[8] Emiliana. “bvhtools”. https://assetstore.unity.com/packages/tools/animation/bvh-tools-144728, (参照 2020-10-12).

[9] 八村広三郎ほか. “モーションキャプチャによる舞踊のデジタルアーカイブ”. 情報処理学会研究報告コンピュータビジョンとイメージメディア (CVIM), Vol. 2007, No. 1 (2007-CVIM-157), pp. 1–8, 2007.

[10] 川澄裕一, 宮岡伸一郎ほか. “個人利用を想定した簡易モーションキャプチャシステム”. 情報処理学会研究報告コンピュータビジョンとイメージメディア (CVIM), Vol. 2008, No. 82 (2008-CVIM-164), pp. 99–106, 2008.

[11] Xsens. “home - xsens 3d motion tracking”. https://www.Xsens.com/, (参照 2020-10-12).

[12] アキュイティー株式会社. “optitrack モーションキャプチャ — acuity inc.（旧 optitrack japan）”. https://www.optitrack.jp/, (参照 2021-1-11).

[13] 株式会社ソリッドレイ研究所. “optitrack/オプティトラック — ソリッドレイ研究所”. http://www.solidray.co.jp/product/sensor/optitrack.html, (参照 2021-1-11).