

Blender を用いた モーショキャプチャーデータの可視化

250304 三原領太郎

背景

モーショキャプチャーより得られたデータを分析、活用する際にはそのデータを出力し、外部で Python や解析ソフトを用いることが一般的である

しかし、Python のみで解析を行う場合、過程と結果を柔軟に三次元で可視化することは容易なことではない。解析ソフトを利用する場合、そのソフトウェアに関する学習コストと導入費用に悩むケースが多く、行いたい解析をするまでのハードルが高い。

そこで、Blender を用いることでその諸問題を解決することを試みた。

Blender とは



Fig1.Blender ロゴ

無料のオープンソース 3D モデリングソフトウェアである。スクリプトとして Python を実行すること、Python による処理をアドオンとして機能拡張を行うことが可能。ただし、解析としての機能はデフォルトで持っていない。

目的

背景として

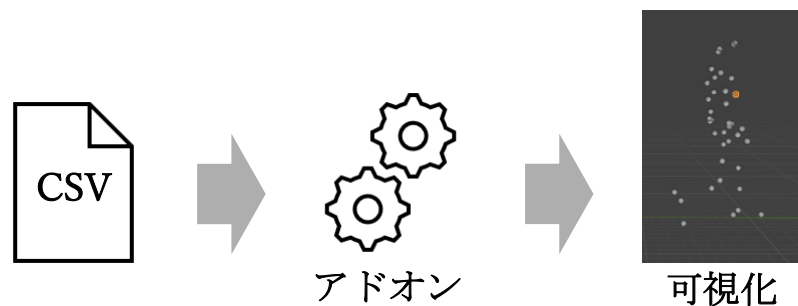
1. 可視化を行う手間
2. 解析ソフトの学習コストと導入費用

という 2 つの問題点がある。

そこで、無料の Blender を用いて可視化を行い、可視化した情報を UI として活用して、Python で解析を行うことを目的としている。

方法

1. C3D フォーマットでの CSV データを点群として Blender に入力し、それを解析する。
2. 入力となる CSV ファイルから必要な情報を抽出し提供する部分をアドオンとして実装。
3. CSV データが点群のアニメーションして置換されることで、その点群から解析を行う。



Graph1.システムの流れ

結果

出力された CSV データから各マーカーの可視化を行うことができた。

それに伴い、解析そのものが可視化され、データの前処理の手間を大幅に削減、視覚的に解析の状態を把握することが可能となった。

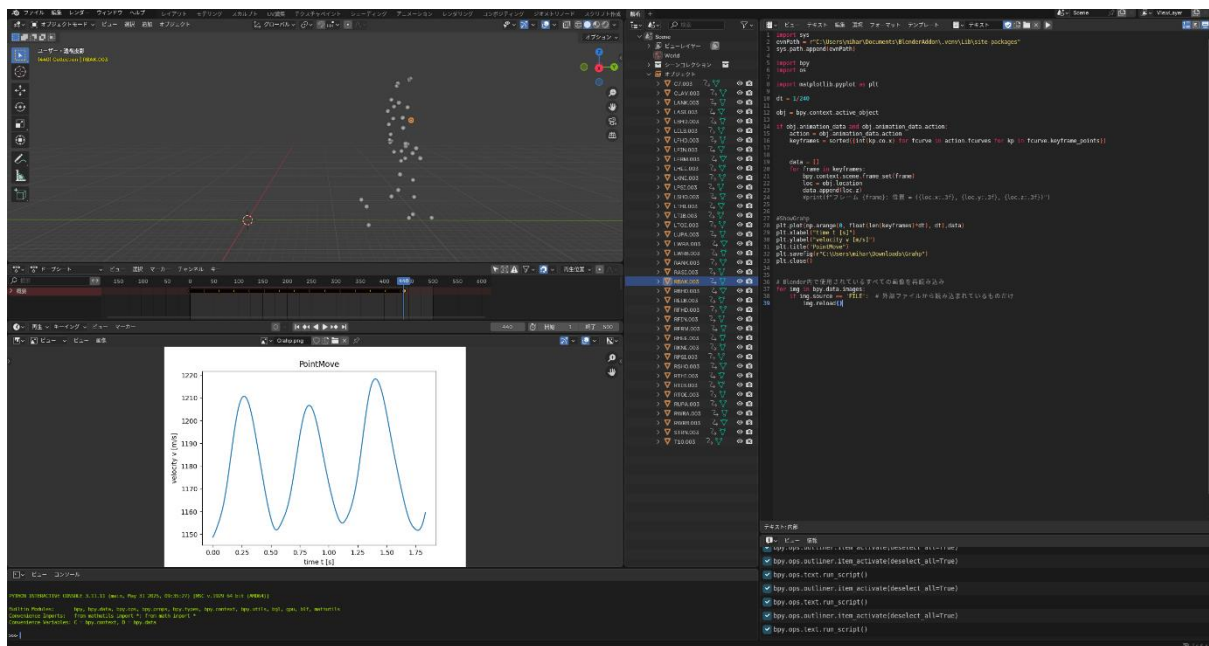


Fig2. 構築した解析環境

Fig2 では選択された点群の Z 軸の動きをグラフによって可視化している。

Code1. 時系列で z 軸の位置をグラフに表示するコード

```
import sys

evnPath = r"C:\Users\mihar\Documents\BlenderAddons\venv\Lib\site-packages"
sys.path.append(evnPath)

import bpy
import os

import matplotlib.pyplot as plt

dt = 1/240

obj = bpy.context.active_object

if obj.animation_data and obj.animation_data.action:
    action = obj.animation_data.action
    keyframes = sorted({int(kp.co.x) for fcurve in action.fcurves for kp in
```

```

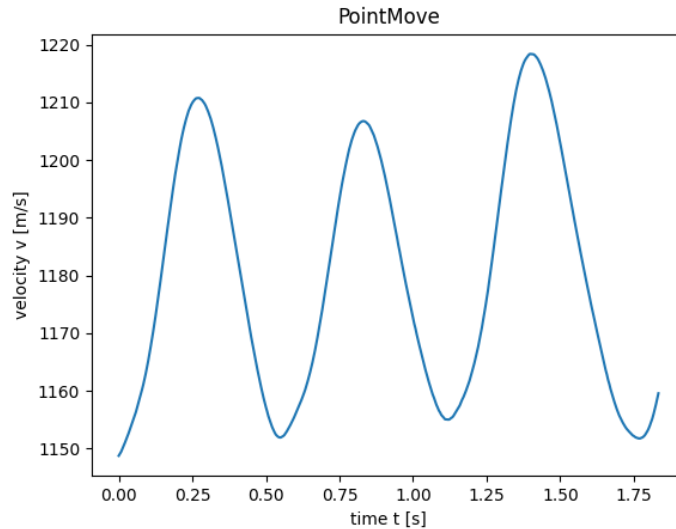
fcurve.keyframe_points})

data = []
for frame in keyframes:
    bpy.context.scene.frame_set(frame)
    loc = obj.location
    data.append(loc.z)
    #print(f" フ レ ー ム   {frame}:   位 置   =   ({loc.x:.3f},   {loc.y:.3f},
{loc.z:.3f})")

#ShowGrahp
plt.plot(np.arange(0, float(len(keyframes)*dt), dt),data)
plt.xlabel("time t [s]")
plt.ylabel("velocity v [m/s]")
plt.title("PointMove")
plt.savefig(r"C:¥Users¥mihar¥Downloads¥Grahp")
plt.close()

#Blender 内で使用されているすべての画像を再読み込み
for img in bpy.data.images:
    if img.source == 'FILE': #外部ファイルから読み込まれているものだけ
        img.reload()

```



Graph2. プログラムにより表示されたグラフ (RBAK)

今後の展望

現状ではモーションキャプチャーデータをオブジェクトとして可視化を行えるようになっており、そのオブジェクトを選択してデータを抽出することが可能になっている。

つまり、可視化から、Blender を UI として Python スクリプトを用いたデータの解析をそのまま行うことができるようになっている。

しかし、現状では3つの問題点が存在する。

1. Python スクリプトで外部ライブラリを導入するためには Blender 専用が存在する Python 実行ファイルにアクセスする必要があるため、環境構築の手間がある。
2. データ形式が Blender の Object のアニメーションフレームであるため、Blender の Python スクリプトを初めて書く者にとって学習コストがある。
3. copilot などのコード生成 AI などのツールも Blender にはデフォルトで備わっていないため、現代のコーディング環境との壁がある。

そこで、簡単なデータ解析を行う Python スクリプトのサンプルコードの提示、一般的に使われる解析手法の関数や UI の実装を Blender 内で行えるアドオンの開発を行うことで、利便性が向上すると考えている。

今回のような全身によるマーカーでは、アーマチュアを作成して骨格を形成することで、より高度な分析や、3D アニメーションとして活用することも期待でき、コンテンツを作成するツールとして、モーションキャプチャーの利用用途拡大を図ることができると考えている。

付録

ソースファイルについて

GitHub より

<https://github.com/ryotaromiharadev-creator/Blender-MotionCapture-Point-Visualizer/tree/1.0>

から

- 解析環境のワークスペース (Sample.blender)
- アドオン (BlenderMoCapPointVisualizer.zip)
- C3D フォーマットの CSV データ (SampleData)

をダウンロード可能

アドオンの導入

編集 → プリファレンス →

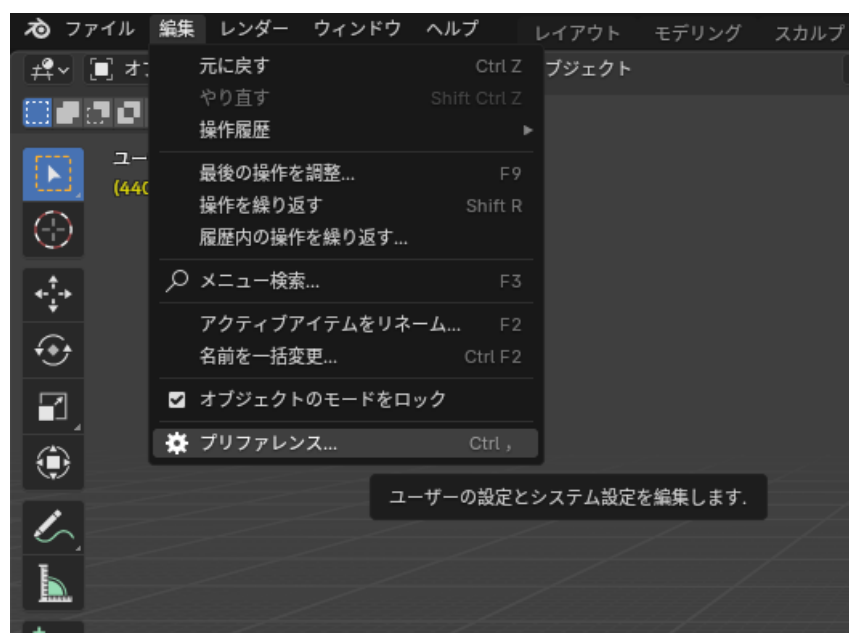


Fig3. プリファレンスを開く

アドオン → ディスクからインストール → 「BlenderMoCapPointVisualizer.zip」を選択

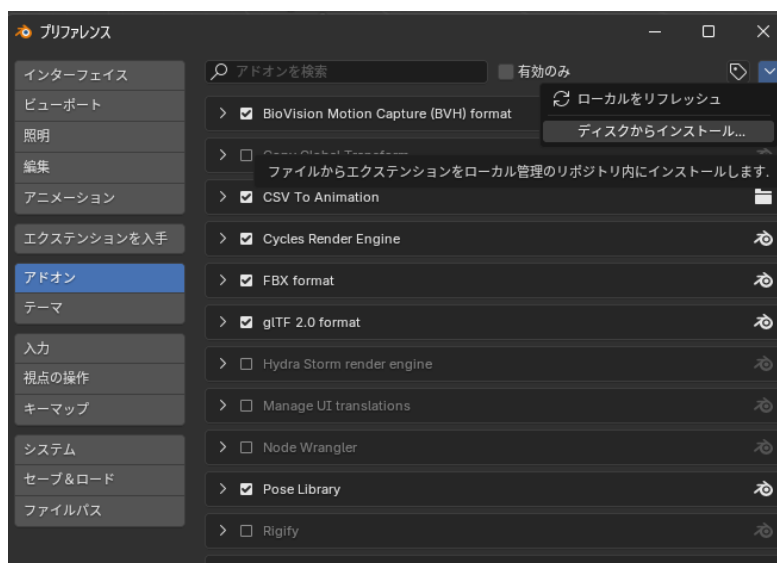


Fig4. アドオンの導入

アドオンの使い方

ファイル → インポート → MotionData.csv をアニメーションに書き出す

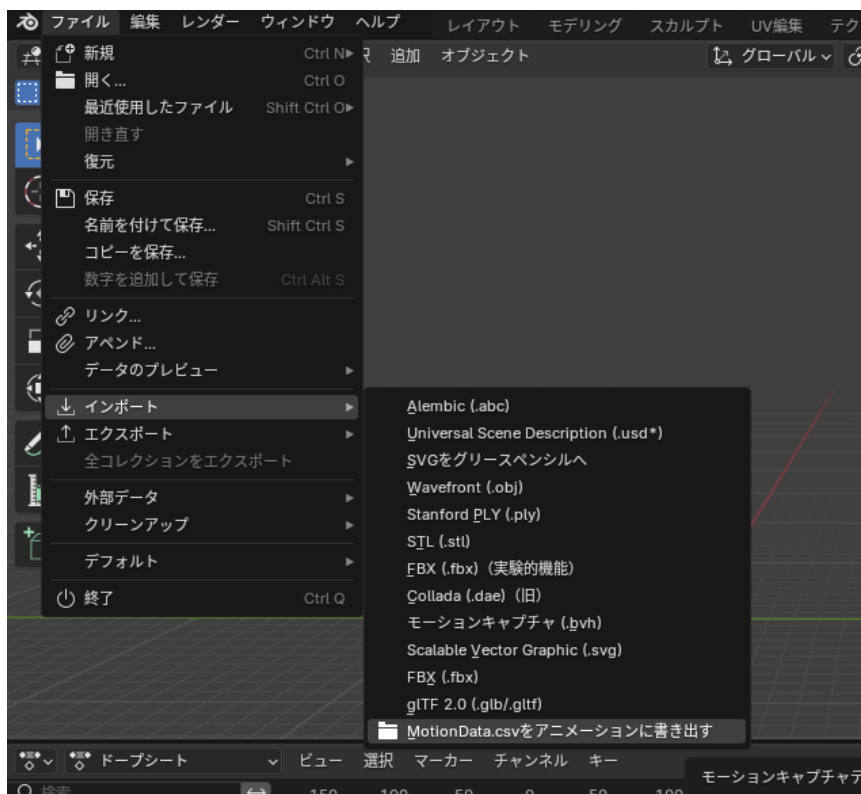


Fig5. アドオンで CSV をインポート

BlenderPython スクリプトに外部ライブラリをインストールする方法

外部ライブラリをインストールしてある仮想環境のファイルパスを指定する。

Code2. 仮想環境をリンクする

```
import sys

#ファイルパスを指定する

evnPath = r"C:¥Users¥Documents¥BlenderAddon¥.venv¥Lib¥site-packages"

sys.path.append(evnPath)
```

この仮想環境は BlenderPython スクリプトと同等の Python バージョンを使用する必要があるため、一度 BlenderPython スクリプトのバージョンを調べ、そのバージョンの Python をダウンロードしておく必要がある。

Code3. Python のバージョンを調べる

```
import sys

print(sys.version)
```

これによって Blender のシステムコンソールにバージョンが出力される。

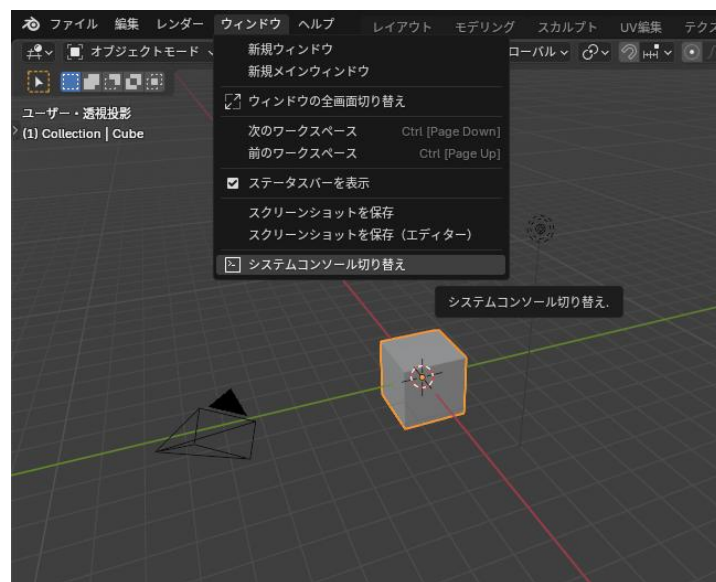


Fig6. Blender のシステムコンソールの開き方

グラフなどを表示する方法

そのまま matplotlib などを使い、グラフを描画しようとする、Window を開くことができず、エラーになってしまう。そのため、一度外部に画像として出力し、確認を行う必要がある。出力された画像データは Blender 内で表示しておくことが可能。

エディタータイプを画像エディターに変更する



Fig7. 画像エディターの設定

開く→保存した画像データを表示する

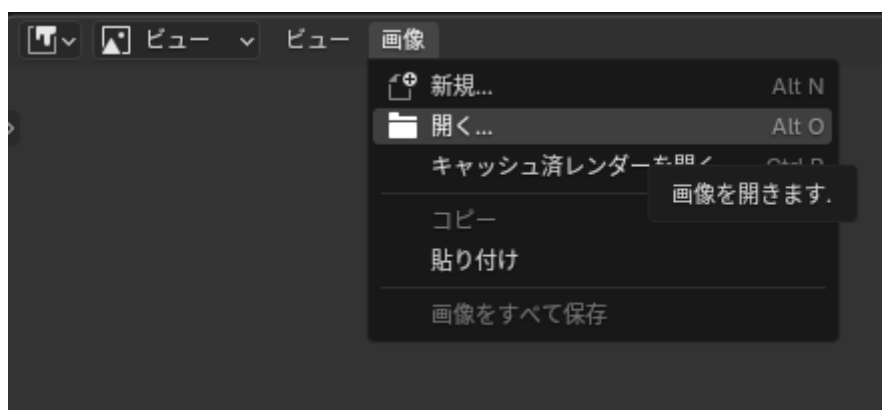


Fig8. 画像データの表示