Vol.2019-HCI-183 No.2
Vol.2019-EC-52 No.2
2019/6/10

RNNを用いたVR空間を操作する為のジェスチャ認識の研究

市川 ひまわり[1,a)]　新田 善久[1,b)]

概要：VR 空間をコントローラで操作するインターフェースは必ずしも直感的ではなく，没入感や操作性が損なわれることがあるため，ナチュラル・ユーザ・インタフェース (NUI) が効果的であると考えられる．そこで我々はジェスチャで VR 空間を操作する NUI システムを開発してきた．そのシステムでは，機械学習で作成した識別器を用いて姿勢を認識し，特定のジェスチャ中に含まれる特徴ある姿勢のシーケンスとマッチングさせることでジェスチャを認識していた．また，ジェスチャ中から等間隔で取り出した姿勢を，特徴ある姿勢として扱っていた．今回我々はシステムを改良し，特徴ある姿勢を k-means 法で決定し，さらに，姿勢の認識とシーケンスの認識を RNN(リカレントニューラルネットワーク) を用いてまとめて行うこととした．この方式の有用性を検証するため評価実験を行なったので報告する．

Natural User Interface using Gesture on VR Space

Ichikawa Himawari[1,a)]　Nitta Yoshihisa[1,b)]

1. はじめに

近年，Virtual Reality（VR）市場がますます拡大している．家庭用の安価なものや，スマートフォンを用いて VR 体験をできるものもある．VR 空間を操作するには Xbox コントローラのようなゲームパッド式のコントローラや，Oculus Touch[1] のようなモーションコントローラを使うのが主流である．しかし，Head Mount Display（HMD）によって手元が見えない状態でのコントローラの操作は必ずしも直感的ではなく，VR における没入感を損なわせることがある．そこで我々はデバイスを身体に装着することなく，全身を用いたジェスチャで VR 空間を操作できるナチュラル・ユーザ・インターフェースが必要であると考え，システムを開発してきた．

我々は，ジェスチャをポーズが時系列に並んだものとしてとらえ，ジェスチャ中の「特徴あるポーズ」の出現を判定してジェスチャ認識を行っている．我々の以前の研究 [2] においては，人間のポーズを骨格データ同士の相対的位置座標で表現し，ジェスチャ中のポーズを等時間隔で取り出して特徴あるポーズとみなした．そして，その特徴あるポーズそれぞれについて機械学習で SVM を用いて識別器を作成した．実際のユーザの動作をジェスチャ認識する際には，特徴あるポーズの識別器が正しい時系列で反応するかどうかを exact matching して判定を行った．

しかし，相対的位置座標ではユーザの身体的特徴によっては識別機が正しく反応しないという問題点があった．また，毎回同じポーズが等間隔で出現しないため，この手法だと不適切なポーズを特徴あるポーズとして選択している可能性があった．そこで今回，データを極座標で表現することにし，k-means 法を用いて特徴あるポーズを選定することとした．また，特徴あるポーズの時系列順の出現を柔軟に判定できるように，Recurrent Neural Network（RNN）を用いた．改善点に関しては，詳しくは 4 章で述べる．

[1] 津田塾大学大学院
Tsuda University, 2–1–1, Tsuda, Kodaira, Tokyo 187–8577, Japan
[a)] m18hichi@gm.tsuda.ac.jp
[b)] nitta@tsuda.ac.jp

Vol.2019-HCI-183 No.2
Vol.2019-EC-52 No.2
2019/6/10

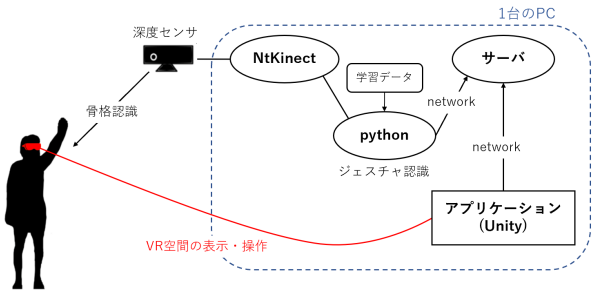

図 1　VR 空間への応用時のシステム

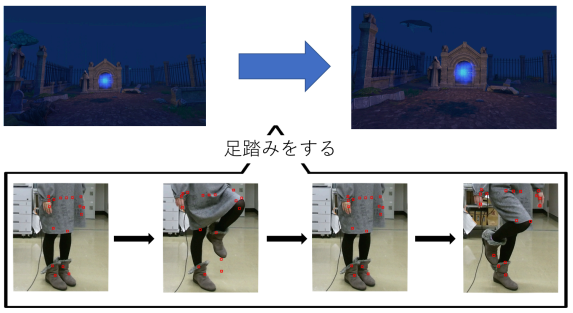

図 2　「足踏み」ジェスチャの例

2.　関連技術・研究

2.1　骨格データの取得

我々のシステムはでは，骨格データを取得するデバイスとして Microsoft 社の Kinect for Windows V2 (これ以降”Kinect V2”と記す)[3] を用いている．Kinect V2 は RGB カメラ，深度センサ，マルチアレイマイクロフォン等のセンサにより，カメラの前に立つユーザの位置や動作，音声などを認識することができる．また，骨格認識を行うことができ，同時に 6 人，1 名につき 25 点の関節の三次元位置を取得することができる．Kinect V2 は 2018 年 10 月に販売終了しているが，システムのプロトタイプの作成には問題ないと考えている．

他にも，深度センサとしてはインテル社の RealSense が存在する．公式の SDK（Intel RealSense SDK2.0）を用いれば，手や指の 3 次元座標を取得することができる．全身の骨格認識を行うには Nuitrack[4] 等と併用する．

また，RGB-D カメラを用いずに骨格検出する手法としては，カーネギーメロン大学の Cao らが発表した OpenPose[5] が存在する．リアルタイムでの使用には構成野の GPU を必要とするが，複数人の人体の関節の位置を取得することができる．

我々のシステムをこれらのデバイスや手法に対応させることは今後の課題である．

2.2　骨格認識を用いた研究

[6] は，RGBD カメラで取得した人間の骨格座標を用いて行動を認識する研究である．RGBD カメラで取得した骨格座標を k-means により選出し，サポートベクタマシン（SVM）を用いて学習させることで認識を実現している．

[7] は RGBD カメラを用いた行動認識において，データの整形を行う部分に工夫を凝らしている研究である．カメラに対するユーザの向きを正面にするように RGBD カメラで取得した座標を回転させ，ユーザの身体的特徴による影響を無視する為に極座標を用いている．整形したデータを隠れマルコフモデルで学習させ，行動認識を行う．

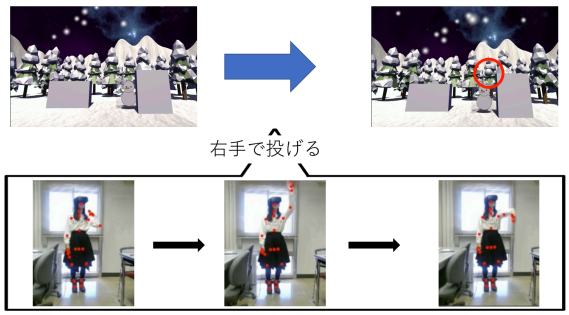

図 3　「投げる」ジェスチャの例

2.3　ジェスチャ認識に関する研究

VR 空間をジェスチャで操作するものとしては，FirstVR[8] が存在する．これは，ジェスチャ入力が可能なセンサを搭載したコントローラと HMD のセットデバイスである．コントローラには筋変位を測定するセンサが搭載してあり，手指の動きを認識することができる．我々の従来の研究 [2] とはデバイスを取り付けずに全身のジェスチャを認識することができることを目的としている点で異なる．

Visual Gesture Builder[9] は Kinect V2 を用いたジェスチャ認識である．Microsoft 標準の Kinect V2 向けジェスチャ認識用の識別器作成ツールであり，Kinect V2 による骨格認識と深度センサ情報を用い，ランダムフォレストで時系列ジェスチャを学習し，ジェスチャの識別器のデータベースファイルを生成する．我々の従来の研究とは，ジェスチャの識別方法が異なる．

3.　研究内容

3.1　本研究の概要

本研究の目的は，デバイスを装着せずにジェスチャで VR 空間を操作することである．そのために，特殊なマーカーをユーザに装着することなく，骨格を認識し，時系列のポーズからジェスチャを認識する．

我々の NUI システムを VR に応用したときの構成ををを図 1 に示す．深度センサで骨格認識を行い，そのデータを

Vol.2019-HCI-183 No.2
Vol.2019-EC-52 No.2
2019/6/10

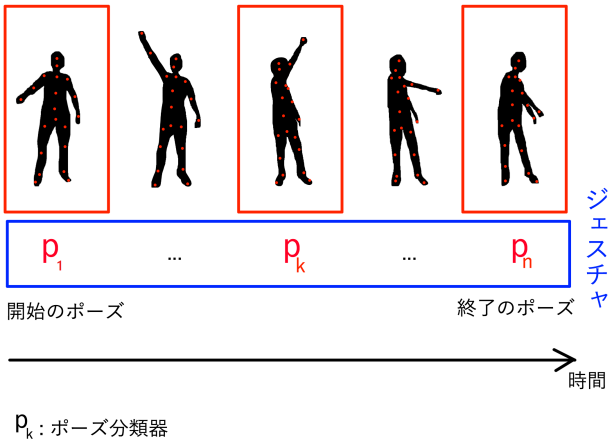

図 4　骨格認識による時系列ポーズデータの取得

もとにジェスチャを認識し，VR 空間を操作する．python 環境で Kinect V2 で取得した骨格データを利用するのに NtKinect[10] ライブラリを利用している．python 環境で作成した識別器を用いてジェスチャを判定し，その結果を Unity[11] で作ったアプリケーションに送り，VR 空間を操作する．VR 空間はユーザの装着する HMD に表示される．

図 2 は「足踏み」のジェスチャの利用例である．ユーザが足踏みを行うと VR 空間で一歩前進することができる．**図 3** は「投げる」ジェスチャの利用例である．ユーザがオーバースローで投げる動作をすると雪玉が発射される．これらはジェスチャの認識がトリガーとなり，操作が行われる．

我々が重視するのは，ユーザによってシステムが拡張可能なことである．ユーザーによる拡張性が必要なのは，ジェスチャに対するイメージがユーザにより異なるためである．例えば，「投げる」というジェスチャであっても，「アンダースロー」や「オーバースロー」など様々な投げ方が考えられる．

3.2　ジェスチャ認識方法について

ジェスチャの認識を実装する方法として，３つ挙げられる．

(1) コーディング ： 姿勢を表す条件を時系列で複数判定するプログラムを人間が書く
(2) 機械学習 ： RNN(deep learning) 以外の機械学習を用いる
(3) 深層学習 ： RNN を用いて時系列の認識を行う

（1）は骨格認識で得られたデータから，三次元座標の大小を比べることで，ジェスチャの認識を行うプログラムを組む手法である．認識率は高いが，コーディングコストがかかり，各ジェスチャごとに莫大な量のコードを書く必要があり，そのため拡張性が低い．本システムでは拡張可能なシステムの開発を目指しているため，この手法はとらない．

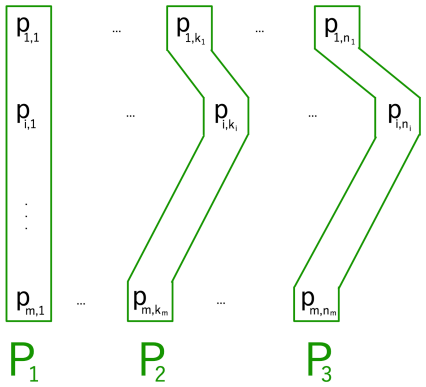

図 5　ジェスチャ中のポーズの識別器生成

（2）は，時系列データを学習できる，RNN 以外の機械学習を利用する手法である．Visual Gesture Builder はジェスチャ認識のための時系列処理まで機械学習で行う，また，我々の従来の研究 [2] では，ポーズ認識を機械学習で行い，マッチングでジェスチャを認識する方法はこれに属する．骨格データさえあればユーザが認識するジェスチャを追加することができ，拡張性がある．

（3）は，深層学習を用いてジェスチャ認識を行う手法である．ユーザがジェスチャ識別器を拡張（カスタマイズ）することができる．今回はこの手法を採用する．

3.3　我々の従来のシステムにおけるジェスチャ認識方式

ジェスチャ認識をする際に，深度センサ搭載のカメラからの距離による差を無くし，少ないデータでも機械学習を行えるようにするために，関節同士の相対的位置関係を入力データとして扱った．相対的位置関係とは，ある関節の座標に対する他の座標の位置をプラス，ゼロ，マイナスの３種類で定義したものである．

ジェスチャを認識するには，ジェスチャに含まれるポーズが時系列順に存在していることを判定すれば良い．**図 4** はあるジェスチャ A を行ったときに取得した骨格データである．ジェスチャ A に赤枠で囲われたポーズが必ず含まれるならば，ジェスチャ A を認識するには，赤枠で囲まれたポーズが時系列に出現することをを判定すれば良い．これらのポーズを特徴あるポーズとする．ジェスチャ A に n 個のポーズが存在するとき，p_1 を開始のポーズ，p_n を終了のポーズとする．開始のポーズと終了のポーズは必ず出現する為，特徴的なポーズだといえる．この時，開始のポーズから終了のポーズの間に特徴的なポーズ p_k が存在するならば，ジェスチャを認識する p_1，p_k，p_n がこの順番で存在することを判定すればよい．我々は p_k を，時間を等分割することで求めた．**図 5** は，ジェスチャを m 回行ったデータから識別器を３個作成する様子である．開始のポーズの識別器 P_1，終了のポーズの識別器 P_3，そして間に存在する特徴的なポーズの識別器 P_2 を作る．これらを並列

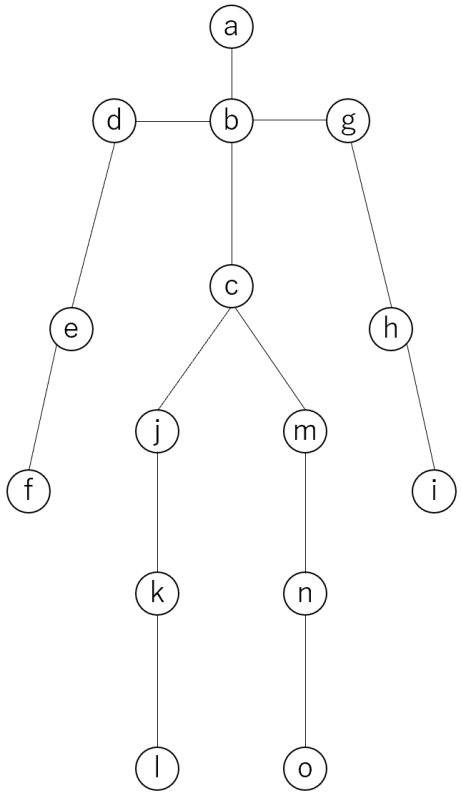

図 6　Florence3D 骨格認識時の関節点

に動かすことでジェスチャを認識する．

3.4　今回の研究における改善点

従来の研究からの改善点は以下のとおりである．

(1) データを極座標系に整形 ：詳しくは 4.2 章で述べる
(2) 特徴あるポーズを k-means 法により選定 ：詳しくは 4.3 章で述べる
(3) RNN による識別器の作成 ：詳しくは 4.4 章で述べる

4.　ジェスチャとその認識

4.1　利用データ

今回，ジェスチャのデータとして，Florence 3D Actions Dataset (Florence3D) [12] を利用した．このデータセットは 9 種類の動作を 10 人の被験者が行ったときの 15 関節の骨格データと動画をまとめたものである．15 点の関節位置を**図 6** に示し，その種類を**表 1** に示す．骨格データは初期型の Kinect を用いて取得されたもので．含まれる動作は，wave, drink from a bottle, answer phone, clap, tight lace, sit down, stand up, read watch, bow である．

4.2　データの正規化

骨格認識で取得した関節の 3 次元座標を以下の 3 点について正規化しなければ，データ間で差が生じてしまう．

(1) 深度センサからユーザへの距離

表 1　関節点と種類

Position	Joint Type
a	Head
b	Neck
c	SpineBase
d	ShoulderLeft
e	ElbowLeft
f	WristLeft
g	ShoulderRight
h	ElbowRight
i	WristRight
j	HipLeft
k	KneeLeft
l	AnkleLeft
m	HipRight
n	KneeRight
o	AnkleRight

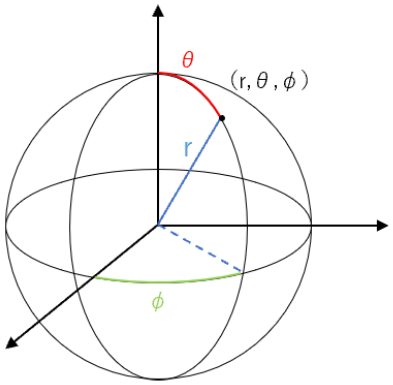

図 7　極座標系

(2) 深度センサに対するユーザの向き
(3) ユーザの身体的な特徴

関節点 SpineBase（図 6 中の c）を基点として，正規化を行うことでセンサからの距離による座標の差による影響を排除する．SpineBase の座標が (x_0,y_0,z_0) だった時，i 番目の関節は (x_i,y_i,z_i) となる．この時，二つの関節の相対的位置は

$$(x', y', z') = (x_i - x_0, y_i - y_0, z_i - z_0)$$

となる．

次に，ユーザによる身体的な差と，身体の回転を無視するために，センサで取得した直交座標系を極座標系（**図 7**）に変換する．極座標系に直すことにより，座標を角度で扱うことができ，身長や手足の長さに関係ないデータを取得できる．センサにより取得したある座標（x，y，z）を極座標（θ，ϕ，r）にするには以下のようにして求める．

$$r = \sqrt{x^2 + y^2 + z^2}$$

$$\Theta = \cos^{-1}(\frac{z}{r})$$

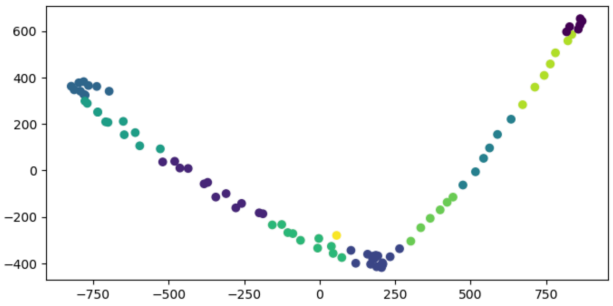

図 8　k-means 法による散布図

$$\Phi = \tan^{-1}(\frac{y}{x})$$

このとき，長さに関係ないデータは必要ないため，r は無視することができる．つまり，（θ，ϕ）の 2 次元にデータを置き換えることができる．

4.3　ポーズの選定

今回我々はジェスチャ中に現れる特徴あるポーズを k-means 法を用いて統計的に選定する方法を採用した．k-means 法は python の機械学習用ライブラリである scikit-learn[13] を用ている．k-means 法では，クラスタリング数 k を指定する必要がある．自動化するには，データのセントロイドを選び出して，それぞれのグループが最も離れているものを見つければよいが，今回は散布図を手動で作成し，最も適切なものを k として手動で選択した．**図 8** は Florence3D の”wave” の骨格データに対して後に k-means 法を適用した散布図である．k=10 の場合のデータであるが，主成分分析を行い，2 次元に次元を削減してある．

4.4　RNN によるジェスチャ認識

本研究では，RNN でジェスチャ認識を行なっている．ニューラルネットワークの構築は Keras[14] と TensorFlow[15] を python 上で利用している．RNN のレイヤーとして，時長期的な依存関係の学習を行うことができる，LSTM（Long Short-Term Memory）を用いている．Florence3D の”wave” ジェスチャのデータをトレーニングデータ 80%とバリデーションデータ 20%に分けた．

図 9 に本研究で用いた Keras によるニューラルネットワークの構成を示す．活性化関数として softmax 関数を，最適化アルゴリズムとして RMSprop を用いている．

5.　評価

図 9 中の LSTM レイヤーでは，最初に入力データに対して行う dropout（以降 $Dropout_1$）と再試行のたびに行う recurrent dropout（以降 $Dropout_2$）をパラメータとして指定できる．また，図 9 中の Dropout レイヤーの dropout の値（以降 $Dropout_3$）も指定できる．これら 3 種類の値を 0.1 から 0.5 まで 0.1 きざみで変化させたときの Training Accuracy と Validation Accuracy の値の変化を調べた．隠れ層の数は 100，エポック数は 100，バッチ数は 64 とした．全体的に Accuracy の高かった $Dropout_3$ が 0.1 の時の結果を**表 5** および**図 10** と**図 11** に示す．計測結果全体は [16] から参照できる．

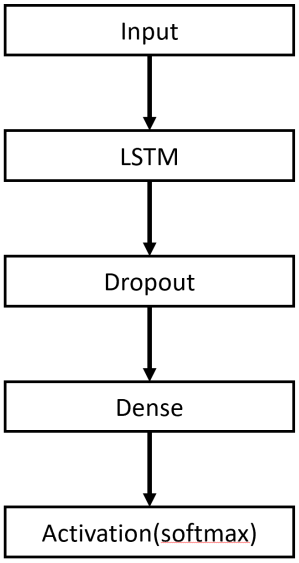

図 9　LSTM を利用したモデル

表 2　正答率

No	$Dropout_1$	$Dropout_2$	Training	Validation
1	0.1	0.1	0.9956	0.7119
2	0.1	0.2	0.9933	0.7386
3	0.1	0.3	0.9914	0.8109
4	0.1	0.4	0.9892	0.7716
5	0.1	0.5	0.9886	0.7792
6	0.2	0.1	0.9911	0.7195
7	0.2	0.2	0.9898	0.7754
8	0.2	0.3	0.9886	0.7373
9	0.2	0.4	0.9822	0.7792
10	0.2	0.5	0.9819	0.7665
11	0.3	0.1	0.9921	0.7678
12	0.3	0.2	0.9879	0.7386
13	0.3	0.3	0.9809	0.8135
14	0.3	0.4	0.9765	0.7931
15	0.3	0.5	0.9800	0.7906
16	0.4	0.1	0.9889	0.7411
17	0.4	0.2	0.9841	0.7297
18	0.4	0.3	0.9794	0.7830
19	0.4	0.4	0.9682	0.7652
20	0.4	0.5	0.9657	0.7614
21	0.5	0.1	0.9867	0.7881
22	0.5	0.2	0.9775	0.7322
23	0.5	0.3	0.9676	0.7919
24	0.5	0.4	0.9505	0.7830
25	0.5	0.5	0.9508	0.7982

図 10 が示すように，Training Accuracy は 90%を超え

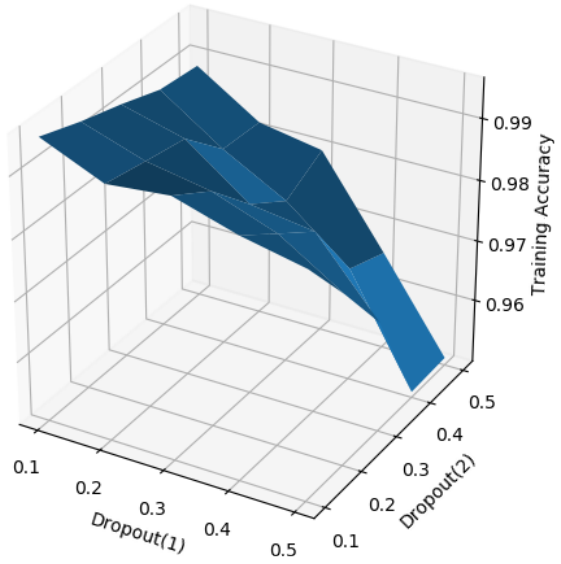

図 10　Training Accuracy

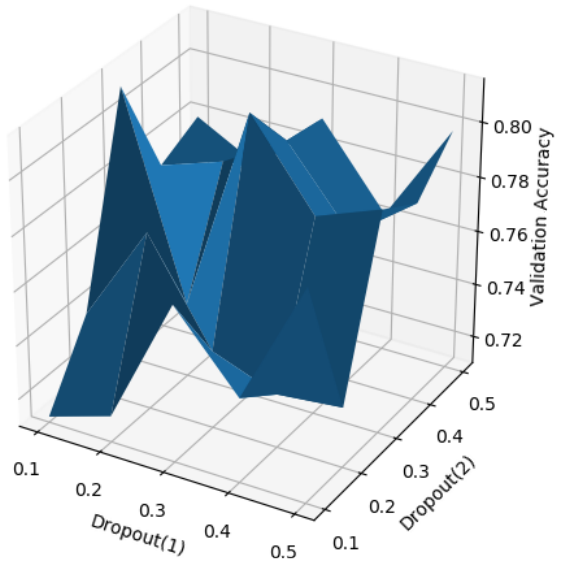

図 11　Validation Accuracy

ている．Training Accuracy は $Dropout_1$ と $Dropout_2$ がどちらも 0.1 の時が 99.56%と最も高く，dropout 数が上がるにつれて，正答率が下がる．また，図 11 が示すように，Validation Accuracy は 70%から 80%の間と低い結果となった．Training Accuracy とは逆で，dropout の数が高いほど正答率が上がる．$Dropout_1$ と $Dropout_2$ がどちらも 0.3 の時が正答率 81.35%と最も高く，どちらも 0.1 の時が正答率 71.19%と最も低い．Training Accuracy が 90%を超えていて，Validation Accuracy が低いことから，過学習が起きており，今回利用した Florence3D に含まれる 215 個の動作では学習データが足りないと考えられる．

6. 考察

従来の研究ではジェスチャ中のポーズを等時間隔で取り出したものを特徴あるポーズとしていたが，今回の研究では k-means 法を用いて統計的に選定した．このことにより，適切なポーズを優先的に特徴あるポーズとして選び出すことができた．ただし，k-means 法を用いるにあたって，今回は散布図から手動でクラスタリング数 k を決めた．特徴あるポーズとして最適なものを選び出すためにも，k の決定を自動化したい．

実験により，今回構築したネットワークは現時点では認識精度が低い状態であることがわかった．学習データに対する正答率が高く，バリデーションデータに対する正答率が低いことから，過学習が起きていると推測される．学習データを増やすことで，RNN の認識精度を向上させることができるので，今後データを増やしていきたい．また，デバイスの入手しやすさを考慮して，Kinect V2 以外のデバイスや手法にも対応したシステムにしたい．

参考文献

[1] Facebook Technologies, L.: Oculus Rift: VR Headset for VR Ready PCs, Facebook Technologies, LLC. (online), available from ⟨https://www.oculus.com/rift/⟩ (accessed 2019-05-05).
[2] 市川ひまわり，新田善久：VR 空間におけるジェスチャを用いた Natural User Interface の研究，第 178 回ヒューマンコンピュータインタラクション研究会, 情報処理学会 (2018).
[3] Microsoft: Kinect for Windows, Microsoft (online), available from ⟨https://developer.microsoft.com/en-us/windows/kinect⟩ (accessed 2018-01-15).
[4] Inc., D.: Nuitrack, 3DiVi Inc. (online), available from ⟨https://nuitrack.com/⟩ (accessed 2019-04-30).
[5] Cao, Z., Simon, T., Wei, S.-E. and Sheikh, Y.: Real-time Multi-Person 2D Pose Estimation using Part Affinity Fields (2019-04-30).
[6] Cippitelli, E. and Gasparrini, S.: A Human Activity Recognition System Using Skeleton Data from RGBD Sensors.
[7] Taha, A., Zayed, H. H., Khalifa, M. and El-Horbaty, E.-S. M.: Human Activity Recognition for Surveillance Applications.
[8] H2L 社：FirstVR, H2L 社 (online)，available from ⟨https://first-vr.com/⟩ (accessed 2019-04-30).
[9] Microsoft: Visual Gesture Builder, Microsoft (online), available from ⟨https://msdn.microsoft.com/en-us/library/dn785304.aspx⟩ (accessed 2019-05-01).
[10] 新田善久：NTKinect - Kinect V2 C++ Programming with OpenCV on Windows10, Tsuda University (online), available from ⟨http://nw.tsuda.ac.jp/lec/kinect⟩ (accessed 2019-04-30).
[11] Technologies, U.: Unity, Unity Technologies (online), available from ⟨https://unity.com/⟩ (accessed 2019-05-05).
[12] : Florence 3D actions dataset, MICC Media Integration and Communication Center, University of Firenze, Italy (online), available from

⟨https://www.micc.unifi.it/resources/datasets/florence-3d-actions-dataset/⟩ (accessed 2019-04-30).
[13] scikit-learn developers: scikit-learn: machine learning in Python - scikit-learn 0.19.1 documentation, scikit-learn developers (online), available from ⟨http://scikit-learn.org/⟩ (accessed 2019-05-05).
[14] Chollet, F.: Keras Documentation, Francois Chollet (online), available from ⟨https://keras.io/ja/⟩ (accessed 2019-04-30).
[15] Google Brain Team: TensorFlow, Google Brain Team (online), available from ⟨https://www.tensorflow.org/⟩ (accessed 2019-04-30).
[16] 市川ひまわり：Experiment Results of *wave* gesture recognition with RNN, Tsuda University (online), available from ⟨https://github.com/himawari-ichikawa/VRGes.git⟩ (accessed 2019-05-05).