Vol.2018-HCI-178 No.2
Vol.2018-EC-48 No.2
2018/6/14

VR空間におけるジェスチャを用いたNatural User Interfaceの研究

市川 ひまわり[1,a)]　飯島 沙織[1]　新田 善久[1,b)]

概要：VR 市場の拡大に伴って，VR 空間の操作に使用する様々な入力装置が開発されている．しかし，そのようなコントローラの操作は必ずしも直感的ではなく，没入感や操作性を損なうことがあった．我々は，ジェスチャ認識によるナチュラル・ユーザ・インタフェースが効果的であると考え，ユーザのジェスチャで VR 空間を操作するシステムを開発した．ジェスチャを，姿勢データが時系列に並んだものであるととらえ，非接触モーションセンサによって取得した姿勢データから，様々なジェスチャを構成する特徴的な姿勢の分類器を機械学習によって複数作成した．これらの分類器を同時に動かしてジェスチャを認識することで，自然な操作を実現した．

Natural User Interface using Gesture on VR Space

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

1. はじめに

今日，AR(Augmented Reality) や VR(Virtual Reality) におけるハードウェアとソフトウェアの普及は目覚ましいものがある．極めて高い没入感と実在感は，娯楽などのゲーム利用だけに留まらず，仮想観光や組み立て構造の展示，建設，教育，医療，社内研修やプロセス製造トレーニング，自動車の運転のシミュレーションなどあらゆるところで脚光を浴び実用化されている．実際に IDC(International Data Corporation) の Worldwide Semiannual Augmented and Virtual Reality Spending Guide による AR と VR のハードウェアとソフトウェア等の関連サービスの合計支出額は 2017 年 114 億ドルから 2021 年には 2150 億ドルにまで市場が拡大される見通しがあり，その成長率は 113.2%と見込まれる [1]．とりわけ VR に関してはハードウェアやゲーム等の販売により 2017 年，2018 年において AR の支出関連を上回ると予測している．実際に現在 VR のハードウェアにおいても，代表的なものでは PlayStation VR[2] や HTC Vive[3]，Oculus Rift[4] 等がある．

並行して，これら VR を体験できるデバイスの操作に使用できる様々な入力装置が開発されている．中でも，一般的に利用されているのが，Xbox コントローラ [5] や Oculus Touch[6] である．しかしコントローラによるコマンド入力は必ずしも直感的ではなく，VR の没入感や操作性を損なうことがある．また視界を全て覆うヘッドマウントディスプレイの装着により視認範囲が制限されるので，手に持ったコントローラを直接見ることができず，操作に慣れにくい面がある．

そこで我々は，VR における没入感や操作性を損なわせない，ユーザが全身を利用した自然な動作でコンピュータに指示し，VR 空間を操作することのできるインターフェースが必要であると考え，システムの構築を試みた．また，「自然な動作」はユーザの持つ文化や生活背景によって差異があるため，システムが予め定義した動作だけではなく，ユーザ自身が定義した動作も組み込むことが可能なシステムの構築を目指す．

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

2. 関連技術・研究

NUI（Natural User Interface）とはコンピュータとのUI(User Interface) において人間が直感的な動作で操作が可能なインタフェースのことである．[7] では NUI を人間の五感や人間が自然で行う動作によって機械を操作する方法と定義している．具体的な例としては，スマートフォンやタブレット端末を操作する際のタッチパネル操作や，Microsoft 社の提供する Kinect for Windows V2（これ以降”Kinect V2”と記す）[8] を用いたジェスチャ操作，Apple 社の手がける Siri[9] といった音声対話システム等がこれに該当する．文字情報を入力とする CUI(Character User Interface) やグラフィカルであることを特徴とするGUI(Graphical User Interface) とは異なり，人間の身体の一部を操作に用いることを基に直感的な操作を要する為，特別な操作方法の習得が不必要であるというメリットがある．

ジェスチャ認識に関する研究としては，手のジェスチャ認識により決められた動作に反応させ広視野電子作業空間を操作を行うものがある [10]．しかし，我々は全身を使った，ユーザの動作によってシステムを操作することが人間にとってより自然であり，重要であると考える．全身を利用することにより，自由度が高まり，足や体勢に関係する動作にも対応することができる．

全身の動作を認識するシステムとしては，Microsoft 標準の Kinect V2 向けジェスチャ認識用の分類器作成ツールである，Visual Gesture Builder[11] が存在するが，2.4 で述べる．

VR 空間におけるユーザーインターフェースの代表例としては Oculus 社が開発販売している Oculus Touch 等が存在する．Oculus Touch はに関しては 2.1 で述べる．

2.1 Oculus Touch

Oculus Touch は Oculus 社が開発販売している Oculus Rift の標準の入力装置である．従来のコントローラとは異なり，左右両手に片手用コントローラを一つずつ持つことで，仮想空間内での細やかな手の動きを表す．また様々なセンサが搭載されており，ジャイロスコープや加速度計により角度や速度も検知できる．手の位置や向きはボディ周囲に配置される赤外線（IR）LED の光を利用してトラッキングされ，物を握ったり，掴んだり，摘んだり，撫でたり，指を指したりという手の動作がを検知できる．検出は手の動きとリアルタイムでリンクした手の映像が映像上に出力されるので，状態を認識しやすい．しかし，一方で一般的なコントローラと同じく入力ボタンも配置してあり，こういったコントローラに触れた経験が少ないユーザはヘッドマウントディスプレイの装着により手元が見えないことから操作に慣れるまで時間を要する．手の位置や向き，指の状態を検知するデバイスであるが，コントローラを保持しながら指の操作を行うため，指の操作が直感的とはいえない

2.2 Leap Motion

Leap Motion[12] は手にデバイスを持たず手の位置や指の動きをトラッキングするモーションキャプチャツールである．非常にコンパクトで，ヘッドマウントディスプレイ自体に搭載することができる．コントローラを持たず，指や手の細やかな動きまで取得可能な Leap Motion は，デバイスを持たず操作ができるが，指や手の動きしか取得できないため，認識できる動作が限定的であるため，全身を利用するという今回の要件を満たさない．

2.3 Kinect V2

Kinect V2 は 2014 年に米国 Microsoft 社が発売したジェスチャや音声認識によって操作ができるデバイスである．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プレイヤーの位置，動作，顔の表情，音声を認識することができる．骨格認識によって，同時に 6 人，1 名につき 25 点の関節の位置を取得できる．

2.4 Visual Gesture Builder

Visual Gesture Builder とは Microsoft 標準の Kinect V2 向けジェスチャ認識用の分類器作成ツールである．Kinect V2 の骨格認識と深度センサ情報を用いて，ランダムフォレスト [13] という機械学習の方式により continuous ジェスチャを学習し，ジェスチャの分類器のデータベースファイルを生成する．また一つのデータベースファイル内に複数ジェスチャ分類器を入れることができる．つまり分類させたいジェスチャの種類が増えたとしてもプログラムコードが増やすことがないというメリットが存在する．

しかし，我々の実験の範囲では，時系列データを扱うジェスチャ識別精度が低かったため，我々はジェスチャの認識に Visual Gesture Builder を用いることはせず，独自のジェスチャ分類器を開発することにした．

3. 機械学習の方式

機械学習には様々な方式が存在する．その中でも代表的な教師あり学習の方式としては以下のようなものがある．

(1) ロジスティック回帰：2 つの目的変数に対して学習を行う．シグモイド関数を用いてニューロンが発火するか否かを学習していく．

(2) SVM（Support Vector Machine）：トレーニングデータを n 次元空間にプロットし，超平面 (データを分離するような平面もしくは直線) を見つけ出すことで学習を行う．

Vol.2018-HCI-178 No.2
Vol.2018-EC-48 No.2
2018/6/14

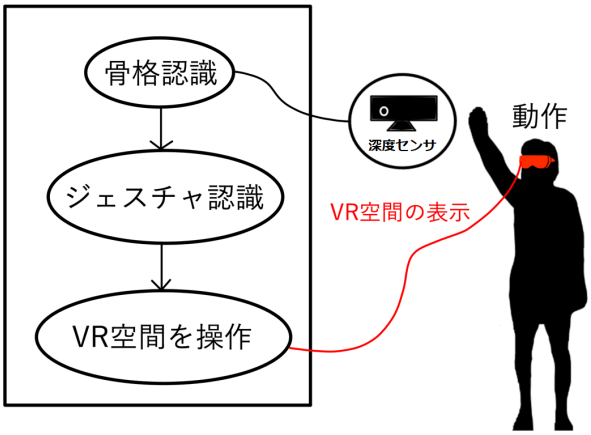

図 1　NUI システム

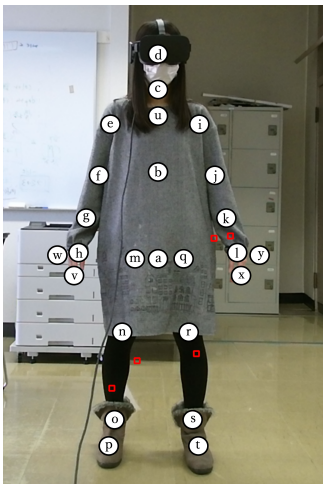

図 2　Kinect V2 による骨格認識時の関節点

(3) ランダムフォレスト：集団学習アルゴリズムの一つであり，学習によって決定木を複数集めて分類を行う．
(4) `TensorFlow`：2015 年に Google によって開発された機械学習用ソフトウェアライブラリ [14]．ニューラルネットワークを構築することができ、ディープラーニングに用いることができる．テンソルを用いて処理を行う．
(5) `Caffe`：ディープラーニングのフレームワークである [15]．計算処理を高速に行うことができ，また GPU にも対応している．
(6) `scikit-learn`：Python の機械学習ライブラリである [16]．前述したロジスティック回帰，SVM，ランダムフォレスト等のクラスタリングアルゴリズムが含まれている．

ジェスチャを認識する場合，ジェスチャ毎にプログラムを組むことも可能だが，非常に難しく，効率も悪いため機械学習を用いることとした．また，機械学習で用いる入力データを増やすことで，多くのユーザに対応したジェスチャでの操作を定義できる．

4. 研究の内容

4.1 本研究の概要

本研究の目的は，VR 空間の操作を現実空間でのジェスチャに対応付け，それにより VR 空間を操作できる NUI システムの構築である．例えば，現実世界で投げる動作を行うと，VR 空間で実際にボールを投げることができるインタフェースを実現する．ただし，「投げる」動作といっても，オーバースローであったり，アンダースローであったり，自然な投げ方はユーザのによって様々に異なる．そのため，ユーザが考える動作をシステムが認識できるように学習できる必要がある．

本研究で作成を目指した NUI システム概要を**図 1** に示す．

表 1　Kinect V2 の関節点と種類
Table 1　Joint Type of Kinect V2.

Position	Joint Type
a	SpineBase
b	SpineMid
c	Neck
d	Head
e	ShoulderLeft
f	ElbowLeft
g	WristLeft
h	HandRight
i	ShoulderRight
j	ElbowRight
k	WristRight
l	HandRight
m	HipLeft
n	KneeLeft
o	AnkleLeft
p	FootLeft
q	HipRight
r	KneeRight
s	AnkleRight
t	FootRight
u	SpineShoulder
v	HandTipLeft
w	ThumbLeft
x	HandTipRight
y	ThumbRight

本研究は以下の 2 つの構成要素から成る．
(1) 機械学習により生成したポーズ分類器を用いたジェスチャ認識
(2) ジェスチャ認識を利用した `NUI` の `VR` 空間への応用

4.2 ジェスチャとその認識

本研究は動作全身を利用するため，Kinect V2 を用いることにする．**表 1** は，Kinect V2 で取得できる関節 25 点

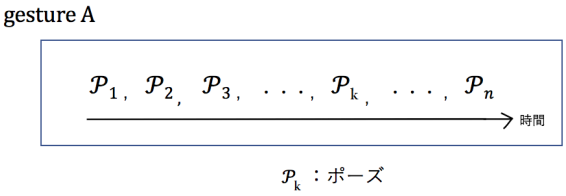

図 3　ジェスチャ A を構成する時系列ポーズ

の種類で，図 2 はその位置を示している．

我々は図 3 で示すように，ジェスチャを時系列状に並べたポーズ集合データとして考える．ここで，あるジェスチャ A にを行ったときに，骨格認識によって各瞬間のポーズデータが図 4 のように取得できたとする．ジェスチャ A が n 個のポーズから構成されるとき，ジェスチャの開始のポーズは p_1 で，終了のポーズは p_n となる．この時，p_1 と p_n の間にそのジェスチャを構成する特徴的なポーズである p_k が含まれているとする・すると，「ジェスチャ A が行われた事を検知する」ということは，「ある限られた時間の間にポーズ p_1，p_k，p_n をこの順番で検知する」ことであるといえる．すなわち，ジェスチャ A を認識するためには，3 つの p_1，p_k，p_n の分類器を用いてジェスチャに含まれるポーズを検出することが必要となる．ジェスチャ A を m 回行ったとき，各ジェスチャを行うのに要する時間は毎回変わることが普通である．図 5 にジェスチャを m 回行った m 個のデータセットを用いて，分類器を学習する方法を示す．ジェスチャの開始時と終了時のポーズは，そのジェスチャを構成する特徴あるポーズとみなすことができる．これらを用いて，m 個のデータセットから開始のポーズの分類器 P_1，終了時のポーズの分類器 P_3 をそれぞれ生成する．問題となるのは，「ジェスチャの途中」の特徴あるポーズをどう選ぶかである．統計的処理を行って特徴あるポーズを検出し，選択することも考えられるが，今回はジェスチャ動作の時間を等間隔に区切った瞬間のポーズを特徴あるポーズとして扱うことにする．図 5 はジェスチャの中心のポーズを特徴あるポーズに選んだ例となっている．この場合は，各データセットを等分割した中心のポーズを分類器 P_2 の学習に使うこととなる．各データセット毎に開始のポーズである $p_{i,1}$ が出現してから $p_{i,n}$ が出現するまでの時間が毎回変わり，それに伴って $p_{i,k}$ の出現する時間も変わる $(1 \leq i \leq m)$.

5. 機械学習による分類器の作成

ユーザがあるジェスチャを行ったとき，その開始のポーズと終了のポーズは必ず特徴的なポーズである．この時，開始のポーズが出現してから終了のポーズが出現するまでの時間を分割し，現れたポーズを特徴的なものと仮定する．ジェスチャを構成する特徴的なポーズが時系列に従って出現するか判定するために，ユーザの動作から特徴的なポーズを判定するための分類器を機械学習で作成する．特徴的なポーズの分類器を作ることができたら，ジェスチャを認識することが可能となる．

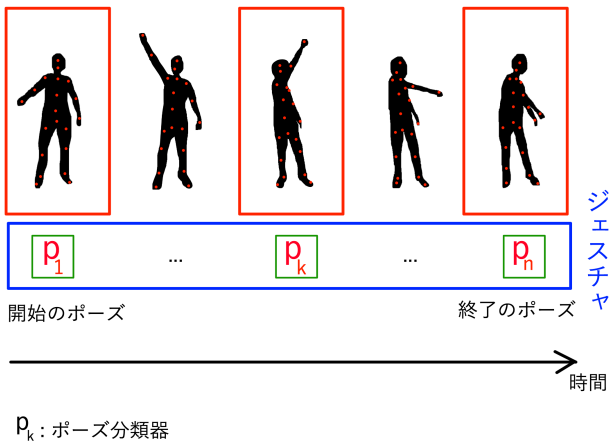

図 4　骨格認識による時系列ポーズデータの取得

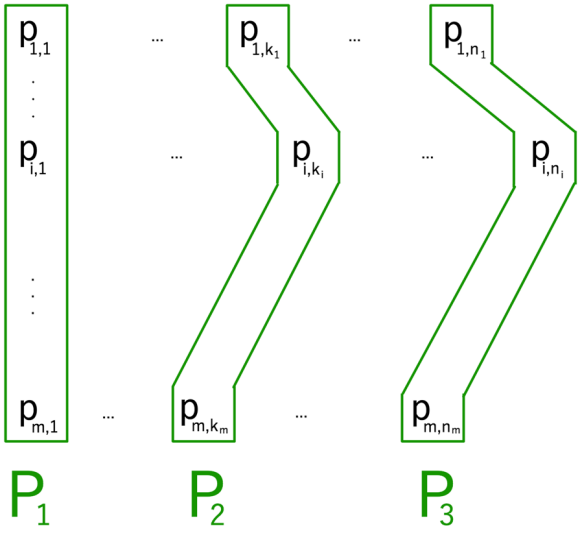

図 5　ジェスチャ中のポーズの分類器生成

今回は非接触モーションセンサである Kinect v 2 を用いて，骨格情報として人体の 25 関節を認識し，その関節同士の相対位置関係を利用してジェスチャ内の特徴的なポーズを機械学習させることとした．

今回は特徴的なポーズであるか否かを学習させたいこともあり， 3 章（ 2 ）で述べた SVM を利用した．

5.1 ポーズ分類器の作成

ジェスチャを構成する特徴的なポーズの分類器の作成には，Python のライブラリである scikit-learn に含まれている SVM による教師あり学習の SVC[17] を用いている．SVC のカーネルとして、「線形カーネル」、「多項式カーネル」、「シグモイドカーネル」、「RBF カーネル」が存在し、他にもユーザが独自で作ることができる。今回利用したの

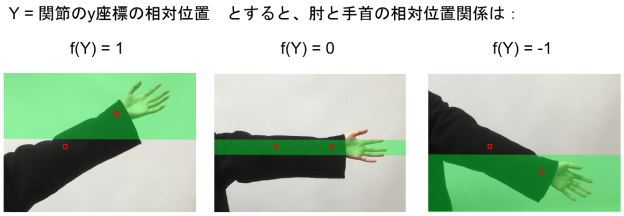

図 6　相対位置関係

は RBF カーネルである。

教師あり学習をするには，ラベル付きのデータをトレーニングデータとテストデータの二つに分けて利用する．トレーニングデータ (訓練データ・教師データ) とは，事前に学習させるデータのことをいう．特徴量を示すデータが含まれるデータ部と，それに対する正例データが含まれるラベル部の二部で構成される．テストデータは，トレーニングデータによる学習を検証するのに用いる．テストデータも，トレーニングデータと同様，データ部とラベル部で構成される．学習後，作成した分類器にテストデータのデータ部を渡し，分類器が予測した結果がテストデータのラベル部と同じであった確率がその分類器の正答率となる．

機械学習に用いる入力データは**図 7** の構造になっている．1 ポーズにつき，Kinect V2 で取得した 25 関節の 3 次元座標のそれぞれの相対位置関係 (25[関節点] * (25-1)[関節点] * 3[次元] = 1800[データ数]) を特徴量としている．ある瞬間の関節の相対位置関係とポーズをクラス分類するための値が一行となっている．これが複数行あり，学習に必要なデータを構成している．**図 6** は肘に対する手首のｙ座標の相対的位置関係を示している．赤い点が関節の位置である．肘のｙ座標よりも手首のｙ座標のほうが大きければ相対位置はプラスの値（例は 1），小さければ相対位置はマイナスの値（例は-1），ほぼ等しければゼロと定義した．緑色の領域は手首の関節がプラス，ゼロ，マイナスになりうる範囲を表している．Python3 環境で Kinect V2 を利用するにあたって，NtKinect[18] を利用した．

5.2　ジェスチャ認識

本システムでは，ジェスチャを「特徴を持つポーズを時系列に並べたもの」として扱った．ポーズ毎の分類器を生成した後，ジェスチャを構成する特徴的なポーズが決められた fps 内で順序通りに存在しているかを検出した．決められた fps とは，ジェスチャの開始のポーズから終了のポーズまでにかかる平均的な長さである．

図 8 には本システムにおけるジェスチャ認識の方法を示す．一定期間中にジェスチャを構成するポーズが正しい順に正になった時点で，ジェスチャが認識される．複数のポーズ分類器が並列に動作し，複数のジェスチャの判定を並行して行っている．

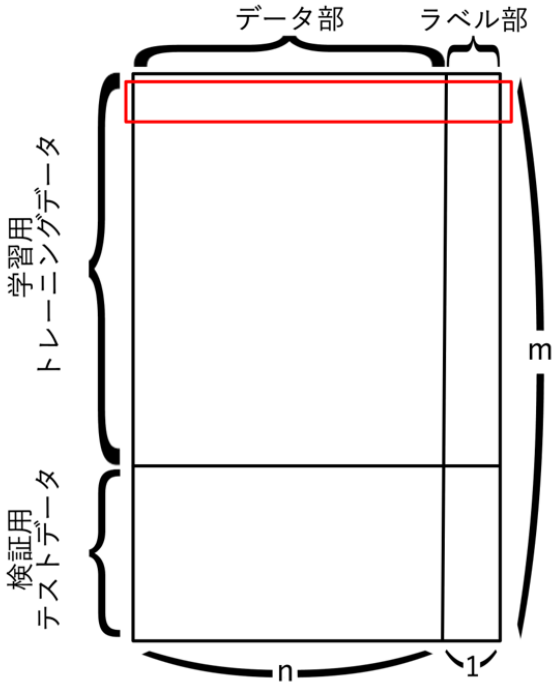

図 7　機械学習データ構成

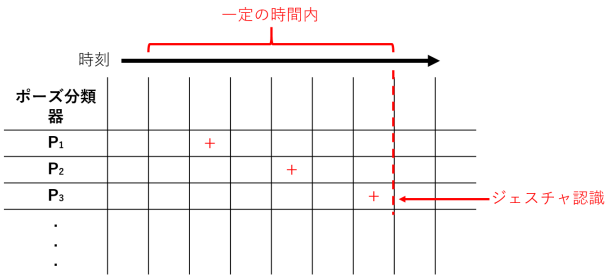

図 8　特徴あるポーズの分類器からジェスチャを認識する

6.　ジェスチャ認識を用いた NUI の VR 空間への適用

「足踏み」,「両手を挙げる」,「手招き」,「追い払う」,「投げる」の 5 つのジェスチャについて，作成したシステムでジェスチャ認識を行い，VR 空間を操作した．VR 空間の構築にはユニティ・テクノロジーズの開発したゲームエンジン Unity[19] を使い，出力には Oculus Rift を使用した．

6.1　VR 利用時のシステムの構成図

図 9 には VR 利用時のシステムの構成図を示す．Kinect V2 でユーザの骨格を認識し，NtKinect を用いて骨格の認識結果を python 環境でジェスチャ認識のために使用する．ジェスチャ認識の結果をネットワークを介して同一 PC 上にあるサーバに受け渡す．その結果を，Unity で作成したアプリケーションがサーバから受け取り，VR 空間への表

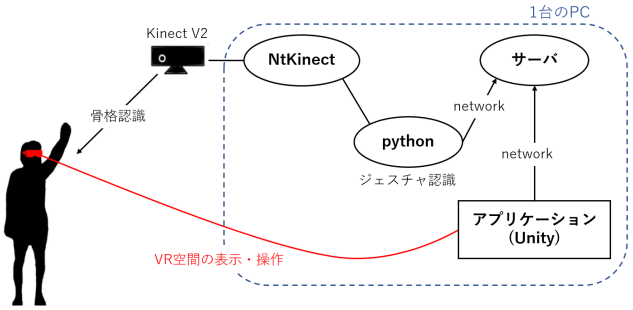

図 9　VR 利用時のシステムの構成図

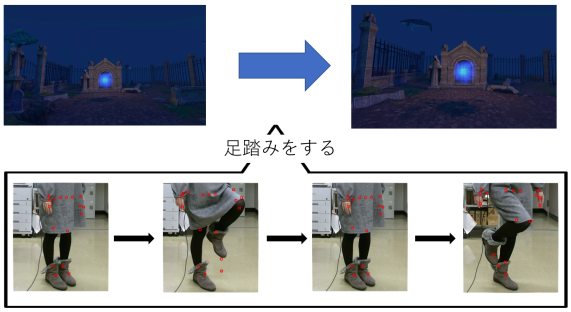

図 10　「足踏み」ジェスチャの例

示・操作を行う．

6.2　足踏み

図 10 に「足踏み」のジェスチャの動作例を示す．「足踏み」のジェスチャを行うと VR 空間で一歩前進するように定義した．足踏みのジェスチャを足をそろえたポーズ，左足を上げたポーズ，足をそろえたポーズ，右足を上げたポーズの時系列とみなし，分類器を作成した．

6.3　両手を挙げる

図 11 に「両手を挙げる」ジェスチャの動作例を示す．「両手を挙げる」ジェスチャを行うと宝箱が開くように定義した．片手では開かず，両手を挙げた時のみ操作として認識される．両手を下したポーズ，両手が肩と水平のポーズ，両手が肩よりも上にあるポーズの 3 ポーズの時系列とみなし，分類器を作成した．

6.4　手招き・追い払う

図 12 に「手招き」の動作例を示し，図 13 に「追い払う」の動作例を示す．「手招き」のジェスチャを行うと対象物が近づき，「追い払う」ジェスチャを行うと対象物が遠ざかるように定義した．今回作成した VR 空間では，画像ファイルに視線上に表示されるカーソルを合わせることで操作する対象物を選択できる．指先が手首より上にあるポーズ，指先が手首に対して水平なポーズ，指先が手首より下にあるポーズの三種類のポーズについて分類器を作成

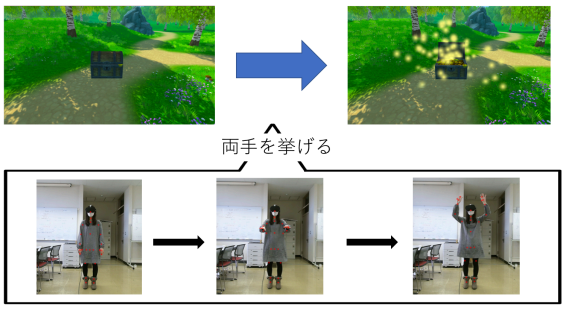

図 11　「両手を挙げる」ジェスチャの例

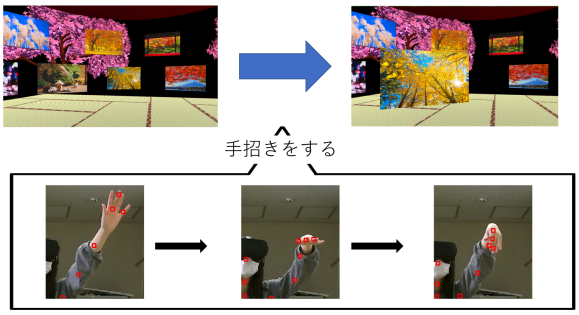

図 12　「手招き」ジェスチャの例

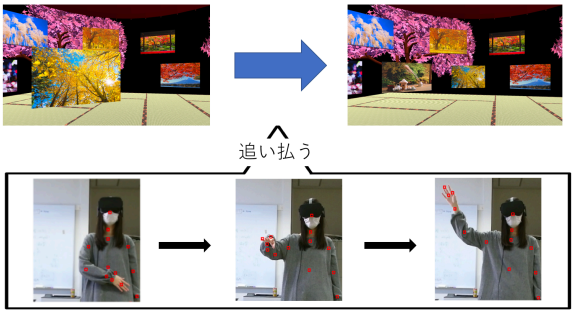

図 13　「追い払う」ジェスチャの例

した．「追い払う」には手が肩より身体の内側にあり低いポーズ，肩と手が大体同じ高さにあるポーズ，手が肩の外側にあり高いポーズの時系列とみなし，分類器を作成した．

6.5　投げる

図 14 に「投げる」ジェスチャの動作例を示す．「投げる」ジェスチャを行うと，雪玉が発射される．手が頭の後ろにあるポーズ，手が頭の上にあるポーズ，手が頭の前にあるポーズの 3 ポーズの時系列とみなし，分類器を作成した．ジェスチャを行ってから VR 空間に操作が加わるため，雪玉が発射されるタイミングが現実のものと異なっている．

7.　評価実験

本研究で作成したシステムの評価を行うため，被験者 5

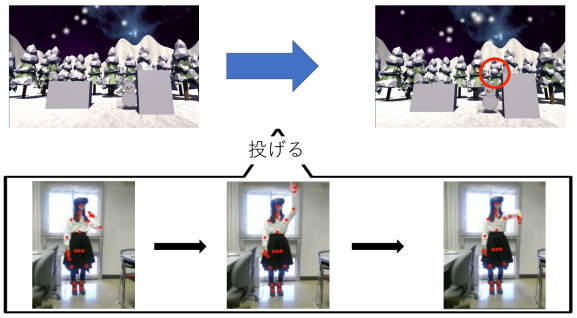

図 14 「投げる」ジェスチャの例

表 2 実験結果

被験者番号	見本無	見本有
1	1	7
2	8	10
3	3	4
4	6	7
5	2	9
合計	20	37

人を対象とした実験を行い，ジェスチャの認識率を調べた．まず，「右手で投げる」ジェスチャについて，名前からイメージするジェスチャを 10 回行う実験を行った．その後，システム構築者側が考えるジェスチャを見本としてみてもらってから，自身のジェスチャを 10 回行ってもらう実験を行った．被験者のジェスチャが認識される確率を調べた．

表 2 に実験の結果を示す．見本を見てもらう前の段階で，自由にジェスチャを行ってもらった場合，認識率は 40%で，見本を見た後のジェスチャの認識率は 74%となった．

7.1 評価実験の考察

評価実験の結果から，見本を見た後のほうが，見本を見せていない状態よりも認識率が高いことが分かった．上手投げや下手投げのように，ジェスチャの種類を増やし，より多くのユーザに対応させることができれば，認識率はより高まると考える．

8. 考察

実験によりジェスチャに対する実際の動きが個人によって相当変化するという当初の予想を裏付ける結果が得られた．ジェスチャ認識をユーザ・インタフェースに用いる場合は個人に合わせて学習・カスタマイズできる環境が必要であろう．

本研究では，データをある関節点と他の関節点のプラス，ゼロ，マイナスの相対位置関係とした．しかし，この相対関係のとり方では，相対関係が変わらない動作（例：突き）に関してはジェスチャとして認識できない．相対位置関係をもっと細かく分類し，機械学習の学習データおよび入力データとして用いることで，より認識精度を上げることができるかもしれない．

また，今回構築したシステムではジェスチャを認識してから操作が行われるため，現実での自然な動きと必ずしも一致しなかった．例えば，「投げる」という動作は動作の最中に握っていた物を飛ばしたいが，「投げる」動作が終わってから物が飛ぶので，自然な動作に感じられない．操作を行うタイミングを調整する必要がある．また，「投げる」といっても，遠くへ物を投げる手の振り方もあれば，近くへ物を投げる手の振り方も存在する．そういったものも区別したいと考える．

更に，開始のポーズと終了のポーズの間に現れる特徴的なポーズを時間分割したときに得られたものとした．しかし，統計的処理を行うことでより精度の高い分類器を作成することができた可能性もある．更に，本研究ではジェスチャ認識を時系列データのマッチングにより行ったが，ジェスチャ認識自体を機械学習で分類したものとの比較をしたい．このことにより，今回採用した特徴のあるポーズとはまた違った特徴を見つけることができるかもしれない．

参考文献

[1] IDC : *Worldwide Spending on Augmented and Virtual Reality Expected to Double or More Every Year Through 2021, According to IDC*, 入手先 ⟨https://www.idc.com/getdoc.jsp?containerId=prUS42959717⟩ (2018.01).
[2] Sony Interactive Entertainment Inc. : *PlayStation VR*, 入手先 ⟨http://www.jp.playstation.com/psvr/⟩ (2018.01).
[3] HTC Corporation : *VIVE* 日本，入手先 ⟨https://www.vive.com/jp/product/⟩ (2018.01).
[4] Oculus VR : *Oculus Rift*, 入手先 ⟨https://www.oculus.com/rift⟩ (2018.01).
[5] Microsoft : *Xbox Elite*, 入手先 ⟨https://www.xbox.com/ja-JP/xbox-one/accessories/controllers/elite-wireless-controller⟩ (2018.01).
[6] Oculus VR : *Avatar SDK Getting Started Guide*, 入手先 ⟨https://developer.oculus.com/documentation/avatarsdk/latest/concepts/gsg-intro/⟩ (2018.01).
[7] 東京工芸大学：調査結果ニュースリリース，入手先 ⟨https://www.t-kougei.ac.jp/static/file/nui.pdf⟩ (2018.01).
[8] Microsoft : *Kinect for Windows*, 入手先 ⟨https://developer.microsoft.com/en-us/windows/kinect⟩ (2018.01).
[9] Apple : *iOS - Siri*, 入手先 ⟨https://www.apple.com/jp/ios/siri/⟩ (2018.01).
[10] 木村朝子，柴田忠久，鶴田剛史，酒井理生，鬼柳牧子，田村秀行：ジェスチャ操作を活用する広視野電子作業空間の設計と実装，情報処理学会論文誌，Vol.47，No.4，pp1327-1339(2006).
[11] Microsoft : *Visual Gesture Builder (VGB)*, 入手先 ⟨https://msdn.microsoft.com/en-us/library/dn785304.aspx⟩ (2018.01).
[12] LEAP MOTION : *Leap Motion*, 入手先 ⟨https://www.leapmotion.com/⟩ (2018.01).
[13] Microsoft : *RFRProgress*, 入手先 ⟨https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn785524(v=ieb.10)⟩

(2018.01).
[14] Tensorflow：*Tensorflow*，入手先 ⟨https://www.tensorflow.org/⟩ (2018.05).
[15] Yangqing Jia：*Caffe — Deep Learning Framework*，入手先 ⟨http://caffe.berkeleyvision.org/⟩ (2018.05).
[16] scikit-learn developers：*scikit-learn: machine learning in Python — scikit-learn 0.19.1 documentation*，入手先 ⟨http://scikit-learn.org/⟩ (2018.05).
[17] scikit-learn developers：*sklearn.svm.SVC — scikit-learn 0.19.1 documentation*，入手先 ⟨http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html⟩ (2018.05).
[18] 新田善久：*NtKinectDLL (DLL) and NtKinect_py (Python Wrapper) for NtKinect*，入手先 ⟨http://nw.tsuda.ac.jp/lec/NtKinectDLL/NtKinect_py01/⟩ (2018.05).
[19] Unity Technologies：*Unity*，入手先 ⟨https://unity3d.com/jp⟩ (2018.01).