修士論文

# スノーボード初心者のためにスノーボードシミュレーターを用いた重心転換トレーニングシステムの開発

2210184　LI LIANG

主指導教員 金井 秀明

北陸先端科学技術大学院大学
先端科学技術研究科
（知識科学）

令和 05 年 3 月

# Abstract

Snowboarding is a winter sport that is widely watched and loved around the world. For beginners, the process of learning to snowboard can be both challenging and dangerous. In particular, beginners often face high learning costs when it comes to mastering basic skills such as pushing slopes (straight-line skating) and falling leaf drifts (left-right lateral skating). The aim of this study is to develop a snowboarding simulation system, especially for snowboarding beginners to practice the push slope and lateral leaf float gliding, through the training of the center of gravity and foot pressure, to help beginners to practice the center of gravity to turn off the practice, to fill the gaps of the simulator system in the practice of snowboarding skills for beginners.

We improved the ski simulator system and developed a training system that meets the needs of beginner snowboarders' center of gravity transfer exercises. An experiment was designed to evaluate the effectiveness of the system. In addition to collecting qualitative data, we will be able to obtain results on the participants' performance during the test. The experiment will include the control of the visual feedback of the interface and the complexity of the skiing course to test the effectiveness of the system in supporting the learning of the snowboarding skill of shifting the center of gravity.

The final result is that for different beginners, the effectiveness of visual feedback on skill acquisition varies from person to person. One group preferred System B with the complex heat map and found it useful for future skill improvement. Another group felt that System C, without visual feedback, would allow them to focus more on improving their performance and thus gain skill. By analyzing the data from each group, it was concluded that the system could be useful for beginner snowboarders to learn the skill of center of gravity transformation. It can help beginners to learn snowboarding skills.

We conclude with recommendations for improving the system. The current system is only suitable for beginner snowboarders, and it is expected that based on the above suggestions, we can conduct research on visual feedback, auditory feedback, and tactile feedback in the system to provide a more realistic sports simulation experience in the future.

# 目次

# 図目次

# 表目次

# 第1章 はじめに

## 1.1 研究背景

スノーボードは、世界的に広く注目され、人気を博しているウィンタースポーツである。中国生まれの私は、2022 年冬季オリンピックの正式開催もあり、ウィンタースポーツとしてスノーボードに興味を持つ若者が増えていると見ている。しかし、初心者にとってスノーボードを習得する過程は困難であり、危険でもある[10]。特に、ストレートグライディング（斜面を押し出す）やリーフフォール（横から滑る）といった基本的なテクニックをマスターする際、初心者はしばしば大きな学習コストに直面する[1]。

スノーボードはボードの上でバランスをとるスポーツである。このスポーツの本質は、ボードをコントロールして傾斜のある雪面で連続的なターンを描く動きを行うことにある。これらのターンを描く動きは、角度、体重のかけ方、移行という 3 つの基本要素に基づいている[1]。滑走の安定性は、スノーボードの初心者と上級者を見分ける重要な基準である。上級スノーボーダーは、ボードをよりうまくコントロールし、自由に移動するために、ステップの力をよく使う。スノーボード初心者の場合、最初の段階ではボードを自由にコントロールすることはできない。まずは、サイドスライディングやリーフフォールなどのテクニックを使って、足がボードに与える力をコントロールする方法から学ぶ必要がある。

現在、スノーボードの学習は主に現場での指導と個人練習に頼っている。この方法の主な限界は、第一に自然のスキー場に依存すること、第二に安全上のリスクがあることである。現地での指導は実体験を提供する反面、無雪地帯に住む人々にとっては練習の機会が制限される。さらに、初心者は練習中に怪我をするリスクもある。スノーボードの傷害統計では、初心者の傷害率は 40%から 60%を占めている[2] [10] [11]。このようなリスクは、学習者に恐怖心を植え付け、スノーボードに対する意欲を失わせる可能性がある。

最も理想的なトレーニング状況で、インストラクターがいても姿勢に正確なフィードバックを得ることは困難であり、初心者に適切なアドバイスを定量的に行うことができない。スノーボードのトレーニングでは、重心転換（ローテ

ーション）、身体の動かし方、足のパウーポイントなど標準定量[9]的に教えるシステムを実現することで、練習者ひとりでもスノーボードを訓練することができると考える。スノーボードの技術を上達に支援するため、スノーボードシュミレーター装置でトレーニングシステムの開発が必要だと考えられる.

バーチャルリアリティ（VR）とシミュレーション技術の急速な発展により、非常にリアルなスノーボードシミュレーションシステムが実現する可能性がある。これにより、安全かつ操作性に優れ、初心者でも簡単にアクセスできる学習環境が提供されるようになる。このシステムは、特にスノーボードの基本技術を身につけたい初心者に最適なトレーニングの場を提供し、効果的かつ効率的な学習プロセスを支援する。[3]

本研究では、スノーボード初心者を対象に、足の力のかけ方と重心の移動[12]に着目したスポーツシミュレーションシステムの開発を目指す。その目的は、初心者がスノーボード学習の旅路をより良くスタートできるように支援することである。

## 1.2 目的

本研究では、スノーボード初心者を対象に、斜面からの突き落としや横滑りの葉っぱの落下を練習するスノーボードシミュレータトレーニングシステムを開発することを目的とする。

スノーボードへの参入障壁を下げるために、重心と足圧のトレーニングに焦点を当てる。このシステムは、様々なゲレンデや滑走ルートを含む実際のスキー環境をシミュレートし、可能な限り本物の体験を提供する。さらに、このシステムには、初心者が正しい姿勢と体重移動の技術を通じてスノーボードの基本的な技術を習得できるように導くための指導モジュールとフィードバックモジュールが統合される[1][2][3][12]。 現在、スキーシミュレータ上の練習システムが開発され、実際の指導や練習に利用されている。

## 1.3 本論文の構成

本章では、スノーボード初心者中心に、研究の背景と目的を紹介する。

次では、研究と関連の既存研究をレビューし、研究の文脈を明確にし、本研究の位置付けを理解しやすくする。

第 3 章では、システム設計や開発について詳しく紹介し、システム改良と機能開発、実験デザイン、データ収集の方法、および分析手法を含める。

第 4 章では、実験や調査の結果に分析を行う。

第 5 章では、実験で得る結果について議論し、研究結果が何を意味するのかを考察する。

本論文の主要な結果をまとめ、それらの結果が将来の研究や実践にどのように貢献するかについて考え、今後の研究方向についても提案する。

# 第2章　関連研究

本章では、運動支援システムに関する研究について述べる。まず、既存のスノーボードトレーニング支援システムについて概説し、続いて同じスポーツであるアルペンスキートレーニング支援システムについて紹介する。次に、これらのスポーツシステムにおけるフィードバックに関する支援機能について述べる。最後に、本研究の位置づけについて考察し、スノーボードの学習支援方向を考えた。

## 2.1 スノーボード学習支援に関する研究

Blake ら[3]は、初めてスノーボードに関連するリスクを軽減しつつ、初心者にリアルなトレーニング体験を提供するために設計されたスノーボードシミュレーションシステムを提案した。このシステムは、ユーザーが力を加えないときにボードにバランスを回復させるテンションバンドを備えたパッシブモーションボードが含まれている。（図 2-1）システム図に示すように、この機能により、ユーザーは「葉っぱが落ちる」などの基本的な初心者スノーボードのテクニックを練習することができ、将来、よりリアルで高度なスノーボードシミュレーションシステムを発展するための基礎を築くことができる。

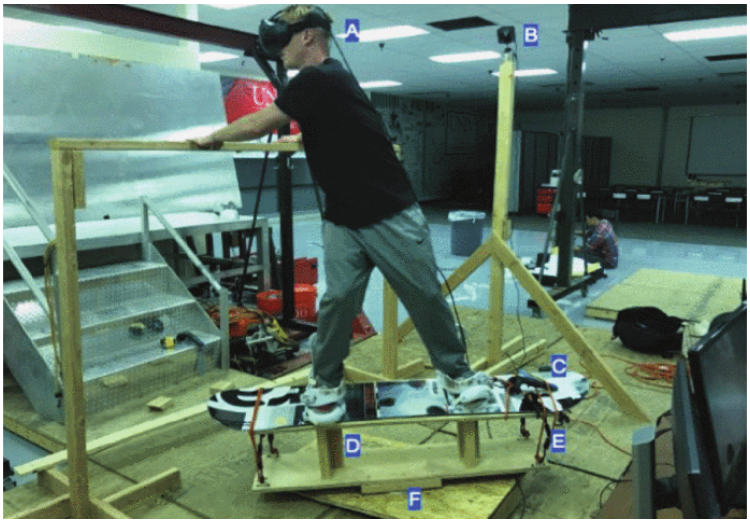



図 2-1：Blake らのシステム

Takashi ら[13]のシステム(図 2-2)は、視覚と聴覚のフィードバックを活用し、初心者が適切な体重移動を習得し、認識した動きと実際の身体の動きの違いを認識できるように支援する。このシステム(図 2-3)は、スノーボードにおける適切な体重移動の重要性を強調し、視覚と聴覚の要素を組み合わせた体重移動に関するリアルタイムフィードバックを提供することで、スノーボード学習を強化することを目的としている。図 2-4 観察によると、短期間の学習シナリオでは、視覚的フィードバックと聴覚的フィードバックの組み合わせ、または視覚的フィードバックのみが、運動学習を強化するために最も効果的な方法であることが証明されていた。

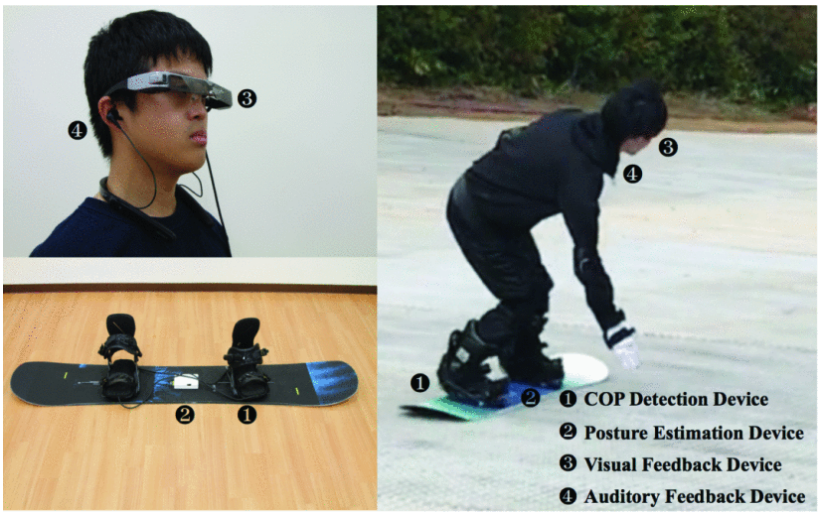



図 2-2：Takashi らが開発されたシステム

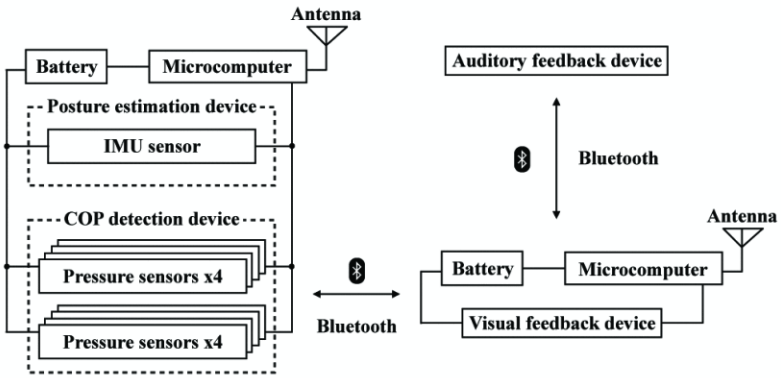



図 2-3：Takashi らのシステム流れ図

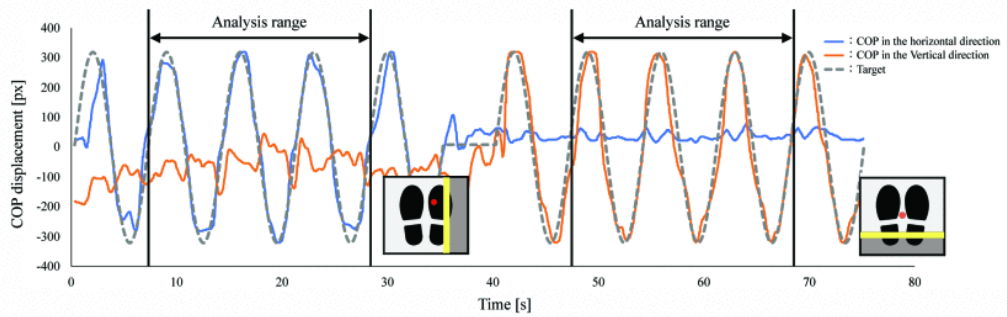



図 2-4：測定 COP 波形と画像データ

Daniel と彼のチームの研究[14]は、リアルタイムのスノーボードトレーニングのためのワイヤレスプロトタイプシステムを開発した。このシステムは、滑走中の一般的なエラーを検出し、学習者に即座に修正フィードバックを提供することができる。彼らはスノーボード（図 2 -5）のスタンスを理解や実行するのに役立つことを発見し、ユーザー受容性と有効性が高いことを示した。従来のコーチング手法では限界がある種目では、触覚フィードバックをスポーツトレーニングに取り入れることで、ユーザーがより効果的にスキルの可能性を学び、向上させることができる。

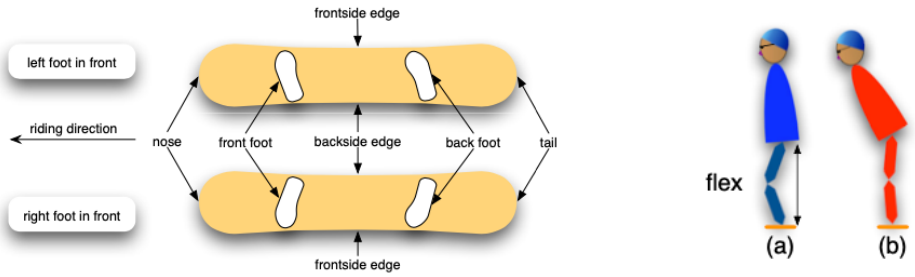



図 2-5：スノーボードのスタンス

特に慣性センサー、力測定装置、表面筋電図などのウェアラブルセンサーの使用、特にランニング、スキー、ラケットスポーツなどのスポーツにおける広範な応用に焦点を当てています。これらのセンサーや力計測ツールは、パフォーマンスやテクニックの評価に役立っている。Daniel らは、運動能力を向上させ、怪我を予防し、バイオメカニクスと動きの力学に関する深い洞察を提供する上で、これらの技術が重要であることを強調する。

## 2.2 スキーの学習支援に関する研究

スキー学習支援システムの研究に関しては、オンライン学習支援技術が近年急速に発展しており、多くの肯定的な評価を得ている。

例えば、東京工業大学小池秀樹研究室の WU らは、シミュレータ「POWER SKI SIMULATOR」[5]を用いて、VR 環境でのアルペンスキートレーニングに適したシステム[VizSki]を開発した。本システムは、図 2-7 に示すように、シミュレータ上で横方向の滑走動作が可能であるとともに、熟練スキーヤーの軌跡を追い、その動きを視覚的にフィードバックして記録することをサポートし、滑走のトレーニング内容を充実させるものである。利用者は視覚的フィードバックにより滑走過程の不備を認識し、適宜修正することができる。

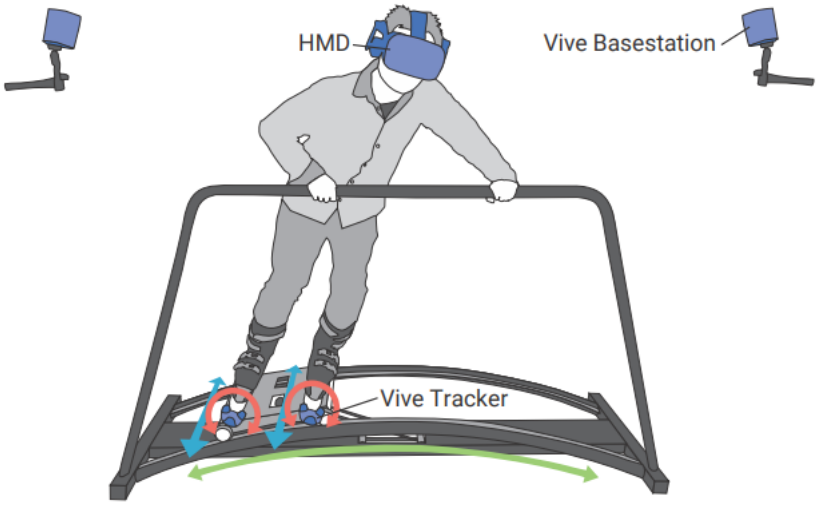



図 2-6：WU らの「VizSki」システム[4]

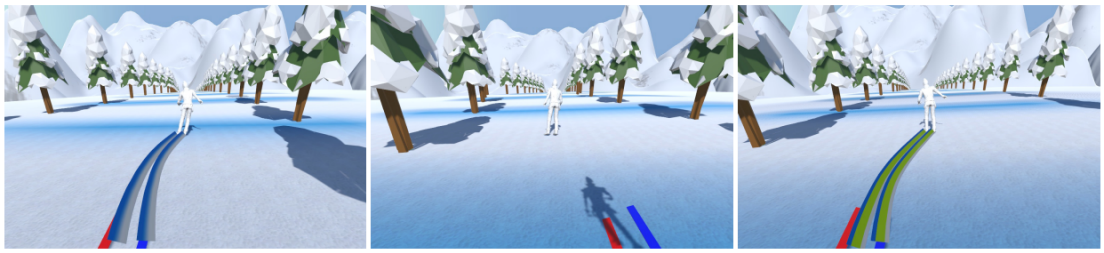


図 2-7：視覚化：軌跡（左）、影（中）、フィードバックの軌跡（右）

亮右らは[5]、Wu らが開発した[Vizski]システムをベースに、図 2-8 に示すように、深度学習を用いてスキーシミュレータ使用時の初心者と熟練者の動きの特徴を抽出した。このアプローチでは、足圧と姿勢によるデータ抽出、熟練者と初心者の VR スキー画像の表示、姿勢検出、体重取得、特徴抽出を行った。このシステムでは、モーションキャプチャに Kinect v2 を利用して動作データ深度学習を用いてデータから（図 2-9）特徴量を抽出し、Grad-CAM で可視化する。その結果、姿勢データと体重データの組み合わせが効果的であることが検証され、熟練者が独自の体重分布パターンを示すことが明らかになり、初心者に的を絞ったトレーニング戦略が提案された。

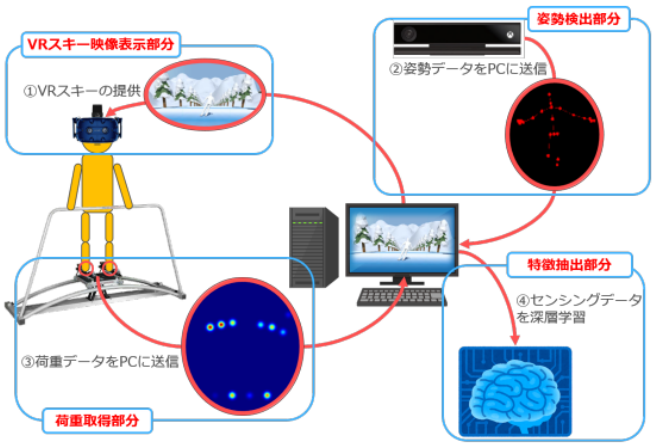



図 2-8 ：亮右らのシステム図[5]

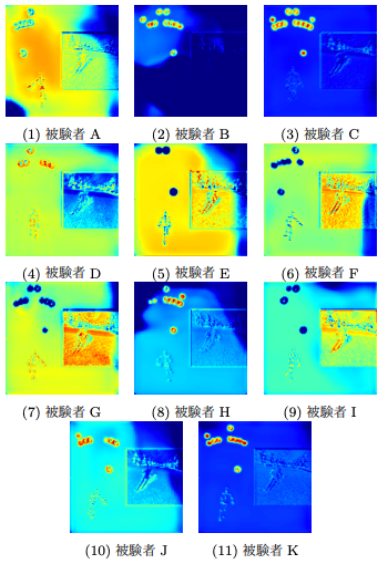



図 2-9：データ特徴抽出

また、小野先輩を中心とする金井研究室では、[Vizski]システムをベースに、体重移動の技術を重視し、視覚的フィードバック(図 2-9）を活用することで、学習者がスキー技術を内面化できるような初心者向け学習支援システムを開発した[6]。図 2-10 に示すように、未来の VR スキーシステムを統合し、フィードバック方法の改善可能性を提案するとともに、トレーニングに人工知能を用いることで、スキーのスキルレベル分類の精度が向上し、初心者の体重移動技術習得を支援する効果が高まったことを強調している。

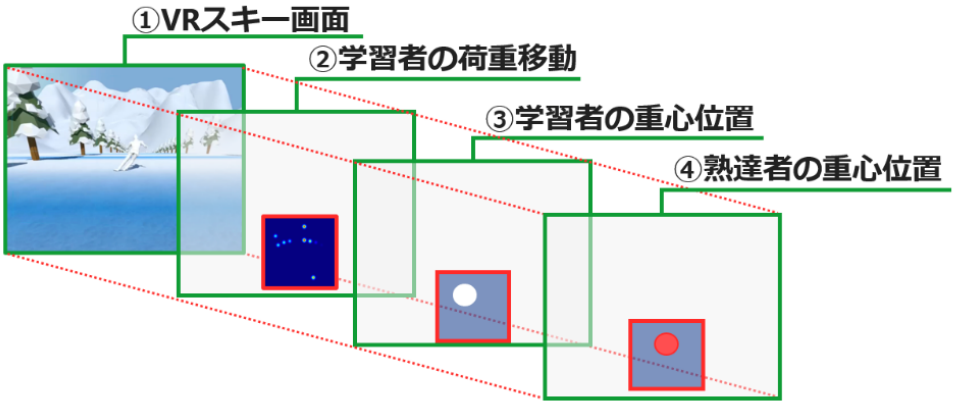
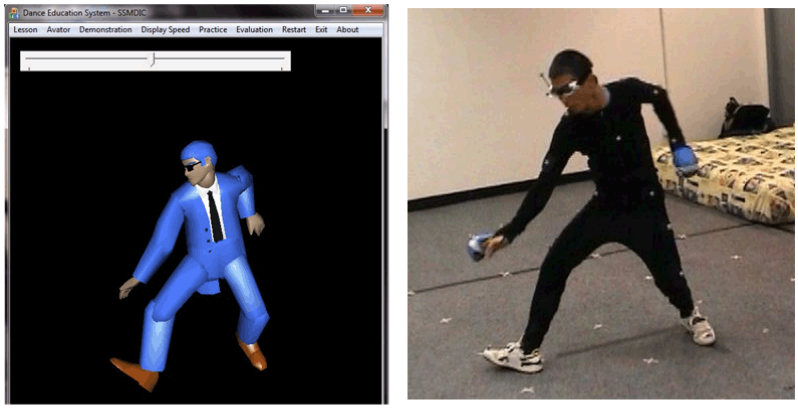



図 2-10：荷重移動の視覚フィードバック構造


図 2-11：小野らの荷重移動システム画面[6]

## 2.3 他スポーツの学習支援に関する研究

Chen らの[15]研究は、モーションキャプチャーとバーチャルリアリティ VR 技術を組み合わせたダンス・トレーニング・システムである。特に、生徒の参加意欲とモチベーションを高めることに注意をした。（図 2-11）このシステムで学習する際、生徒はバーチャル教師が実演する動きを真似し、システムは生徒の動きをキャプチャして成績を分析する。モーションキャプチャからのフィードバックは、スキルを向上させ、学習への関与と意欲を高め、運動スキル学習支援に有意さを示したと考えている。





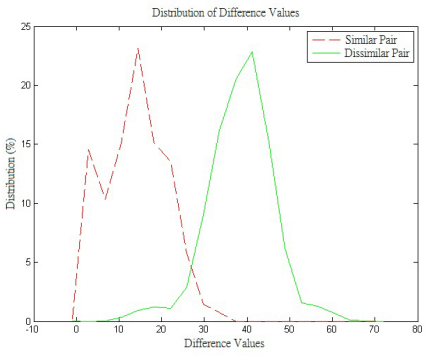



図 2-12：Chen らのシステム図[6]

## 2.4 本研究の位置づけ

スキーやスノーボードなどスポーツ学習支援に関する先行研究から、シミュレータシステムが学習支援に役立つ可能性があることが推測される。初心者にとって、スキーシミュレータを直接使い始めることは非常に困難である[7]。そして、現在スノーボードシミュレータシステム上で初心者の技術練習をサポートする研究は非常に少ない。

本研究では、スノーボードシミュレータ使用時の制御に足圧センサ[6]を用いることで、足圧および筋力技術トレーニングにおいてより良い結果が得られる可能性を探る。アルペンスキーシミュレータをスノーボード用に改造し、圧力センサーを用いてアバターの動きを駆動することで、スノーボードの重心移動技術の学習と自立訓練を支援することを目的とする。

# 第3章　スノーボード支援システム

本章では、実際にスノーボード学習支援システムを導入し、先行研究[6]に基づいてシステムの機能改善と運動ロジックの改良を行った。その後、改善・開発の方法論とプロセスについて詳述し、システムの実用性を検討する。

## 3.1 概要

本研究では、スノーボード初心者が直面する課題である、基本技術の習得の難しさ、練習の季節的制約の克服を目指す。スノーボードシミュレータを用い、主に初心者の横滑り滑降フェーズに焦点を当て、初心者に優しい重心移動トレーニングシステムの開発を提案する。

モーショントラッキング技術を活用することで、練習者はシステム内で自身の動作姿勢を視覚的に観察することができる。視覚フィードバックを利用した学習支援手法と、足圧センサーの応用、初心者に適したスキールートの設計を組み合わせることで、スノーボードシミュレーター上で滑降の過程をシミュレートする。

最終目標は、重心移動技術の習得をサポートするスノーボード・トレーニング・システムを開発したい。このシステムにより、初心者はスノーボードにおける身体姿勢、重心移動の制御、滑走原理を理解することができ、スノーボードの学習効率を高めるようにする。さらに、シミュレータを用いた初心者向け重心移動トレーニングシステムの有意さを検証することを目的とする。

## 3.2 使った装置について

本研究で使用した機器の一部は、過去の研究[5][6]に由来するものである。採用した機器には、Pro Snowboard シミュレータ[図 3-1]、Microsoft Azure Kinect DK[図 3-2]、足圧センサー[図 3-3] [6]が使用した。

図 3-1：POWER SNOWBOARD SIMULATOR

図 3-2：Microsoft Azure Kinect DK

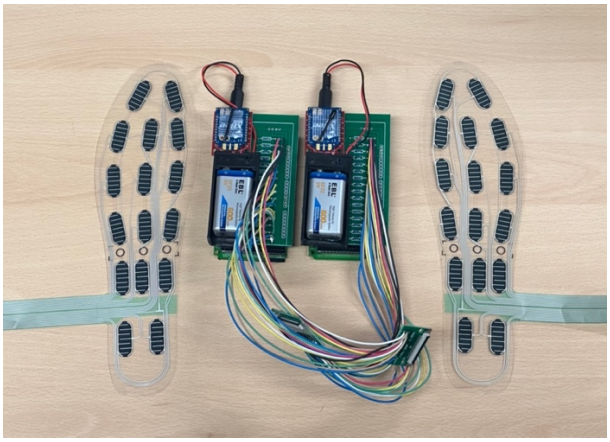


図 3-3：足圧センサー[6]

## 3.3 既存システムの改良

小野らによって開発されたスキー学習支援システム[6]では、モーショントラッキングや動作ロジックなどに改良が加えられ、圧力視覚フィードバックや練習制御方法などの機能が開発された。

システム構造には、モーションキャプチャ装置、足圧センサー装置、PC・システム、スノーボードシミュレータの 4 つの主要コンポーネントが含まれる。図 3-4 を参照。Xbee Wi-Fi[6]で足圧データに無線通信できる。

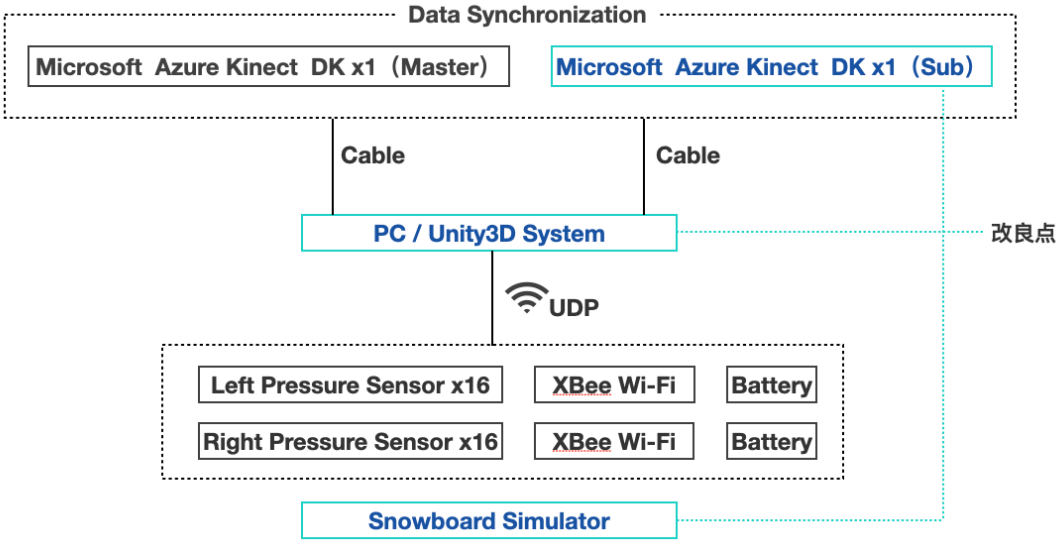



図 3-4：改良したシステム構造図[6]

小野らのスキー向けのシステム[6]にスノーボード向けに改良し、モーショントラッキングカメラを追加し、スノーボードシミュレータと組み合わせることで、より初心者の技術トレーニングに適したシステムを開発した。

### 3.3.1 深度カメラ増加

従来のシステムは、スキーヤーが下を向くアルペンスキー向けで、実験者の正面にカメラを設置することでトラッキング精度が優秀である。しかし、スノーボードの初心者も横滑りでは下を向くが、上達するにつれて、体を横にしてスノーボードを水平にする高度な滑走に移行する。

今回のアップデートでは、より安定したスノーボーダーの身体のトラッキングを目指すため、Microsoft Azure Kinect DK を使い、人体をトラッキング

し、さらに Kinect カメラをもう 1 台横に追加した。この追加は、スノーボーダーが横向きに立った状態でのモーショントラッキングの安定性を高めることを目的とした。図 3-5 によって、 2 つカメラに連接した。

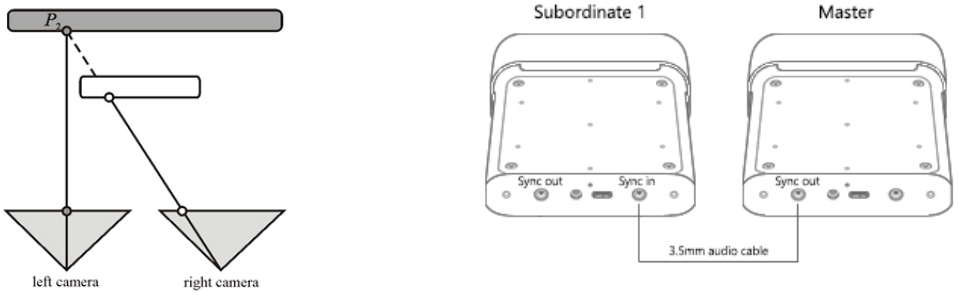



図 3-5：Kinect DK 2 個同期（左）、連接方（右）

データ同期とモーションデータの補足に Microsoft Azure Kinect デプスカメラを利用した。2 台のカメラを互いに 90 度の角度で配置した。図 3-6 のように、複数のカメラを同期させるコードを開発した。なお、動作データ同期に関して、Microsoft Azure Kinect 開発ドキュメントに参考した。

```
...
// Kinect
private Device kinect_master;
private Device kinect_sub;
private Tracker main_tracker;
private Tracker side_tracker;
private Animator animator;
private Capture[] capture;
private Frame main_frame;
private Frame side_frame;
private Skeleton main_skeleton, side_skeleton;
private JointId jointId;
private Microsoft.Azure.Kinect.BodyTracking.Joint main_joint, side_joint;
private Quaternion quaternion, main_quaternion, side_quaternion, average_quaternion, footLeftRot,
        footRightRot;
private System.Numerics.Vector3 main_jointPos, main_footLeftPos, main_footRightPos;
private System.Numerics.Vector3 side_jointPos, side_footLeftPos, side_footRightPos;
private System.Numerics.Vector3 jointPos, footLeftPos, footRightPos;
private Transform tfAnimator;

// for camera configurations
private DeviceConfiguration device_config;
private WiredSyncMode sync_mode;
private TimeSpan delay_off_master_usec;

// for defining devices
private bool master_found;
private Device master_device;
private List<Device> subordinate_devices;
```

```
...
void StartSyncCameras(DeviceConfiguration master_config, DeviceConfiguration sub_config) {
  foreach(Device device in subordinate_devices){
    device.StartCameras(sub_config);
  }
  master_device.StartCameras(master_config);
}
  // prepare a tracker
  main_tracker = Tracker.Create(
  master_device.GetCalibration(),
  new TrackerConfiguration() {
    ProcessingMode = TrackerProcessingMode.Cuda,
    SensorOrientation = SensorOrientation.Default });
  side_tracker = Tracker.Create(
  subordinate_devices[0].GetCalibration(),
  new TrackerConfiguration() {
    ProcessingMode = TrackerProcessingMode.Cuda,
    SensorOrientation = SensorOrientation.Default });
...
void Update() {
  // get elapsed time
  time = (int)Math.Round(Time.time*1000);
  // get a frame from a tracker
  if(main_frame != null){
    if (main_frame.NumberOfBodies > 0) {
      main_skeleton = main_frame.GetBodySkeleton(0);
  foreach (KeyValuePair<HumanBodyBones, JointId> pair in boneJointMap) {
    jointId = pair.Value;
    main_joint = main_skeleton.GetJoint(jointId);
    quaternion.Set(main_joint.Quaternion.X, main_joint.Quaternion.Y, main_joint.Quaternion.Z,
      main_joint.Quaternion.W);
  // Sync data
    tfAnimator = animator.GetBoneTransform(pair.Key);
     if(tfAnimator){
      tfAnimator.rotation = quaternion.normalized * this.GetQuaternionOffset(jointId);
    }
      jointPos = main_skeleton.GetJoint(JointId.Pelvis).Position;
      footLeftPos = main_skeleton.GetJoint(JointId.FootLeft).Position;
      footRightPos = main_skeleton.GetJoint(JointId.FootRight).Position;
      main_jointPos = main_skeleton.GetJoint(JointId.Pelvis).Position;
      main_footLeftPos = main_skeleton.GetJoint(JointId.FootLeft).Position;
      main_footRightPos = main_skeleton.GetJoint(JointId.FootRight).Position;
      side_jointPos = side_skeleton.GetJoint(JointId.Pelvis).Position;
      side_footLeftPos = side_skeleton.GetJoint(JointId.FootLeft).Position;
      side_footRightPos = side_skeleton.GetJoint(JointId.FootRight).Position;

      jointPos = (main_jointPos + side_jointPos) / 2;
      footLeftPos = (main_footLeftPos + side_footLeftPos) / 2;
      footRightPos = (main_footRightPos + side_footRightPos) / 2;

      tfAvatar = avatarRoot.transform;
      vector3.Set(0, -(jointPos.Y - Math.Max(footLeftPos.Y, footRightPos.Y)) / 10000f, 0);
      tfAvatar.localPosition = vector3;}
...
```

図 3-6：複数 Kinect DK 同期コード

## 3.4 学習支援システムの開発

先行研究[6]のシステムの圧力センサ部分機能を使用し、初心者に向けスノーボードシミュレータートレーニングシステムを開発した。

### 3.4.1 圧力データの遠隔制御

足圧センサーを使い、各足に 16×16 個のセンサーからデータを収集した。このデータを小野らの方法[6]に従って処理した後、重心（Gx）を計算した。そして、重心データを（-10～10）の範囲に標準化し、アバターの X 軸方向にマッピングしてリアルタイムに制御することで、「見たままが得られる」学習効果を実現していた。

重心計算いついて、16x 2 （32 個）左右足のデータより、$i$ 個目の圧力センサの位置を（$xi$ , $yi$ ），I {(x1,y1),(x2,y2),(x3,x3)…(x32,y32)}，質量$mi$、 訓練者の足の重心（$Gx$，$Gy$）[6]として計算する。

$$Gx = \frac{\sum_{i \in I} x_i m_i}{\sum_{i \in I} m_i} \qquad Gy = \frac{\sum_{i \in I} x_i m_i}{\sum_{i \in I} m_i}$$

以上の方式を用いて、重心（$Gx$）は平均正規化され（$x'$）、平均値$u$、アバターの3D位置（$x'$）にマッピングした。図3-7のように、圧力センサーでアバターにコントロール機能を実現する。正規化処理においては、x'の位置データについて、最大値max($x$)と最小値min ($x$) を用いて計算される、

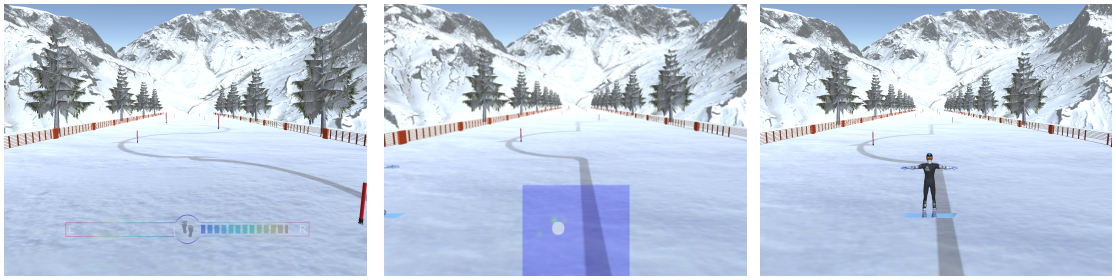
$$x' = \frac{G_x - u}{\max(x) - \min(x)}$$

```
...
// avatar control
tfAvatar = avatarRoot.transform;
float sx1 = (-10)+(20/255f)*(center[0]+10);
float sx2 = 50*(center[0]-127)/255f;
vector3.Set(sx1, 0, 0);
tfAvatar.localPosition = vector3;
...
```

図 3-7：アバターにコントロールコード

### 3.4.2 視覚的フィードバック

視覚化という点では、足圧の変化を区別したい。両足の足圧データを視覚化する手法を採用し、足圧分布の範囲と強度に関するより詳細かつ精密な情報を提供する。このアプローチにより、足圧データの分析と解釈が容易になり、より深い洞察を得ることが可能となる。このアプローチにより、ユーザーはトレーニング中の自分の足圧をより明確に理解しやすくなる。これを実現するために、圧力の視覚的フィードバック機能に 3 つの異なる複雑さのレベルを開発した。



A.視覚圧力バー機能あり　B.荷重移動機能あり[6]　C.視覚フィーック機能なし

図 3-8：三つ視覚フィードバック機能

図 3-8 のように、三つ視覚フィードバック機能を設計した。

A、　簡潔な視覚的圧力バー機能： 黒い線に沿って、コースに完走するため、圧力バー内の色の範囲は足圧の強度を示す。被験者の足圧を簡潔な圧力バーの形で表示し、複雑な画面が練習過程に与える干渉を減少させることを目的としている。

B、　荷重移動機能あり： 小野ら[6]の研究に基づき、白い点は被験者の足底の重心移動と足の圧力熱図を表示する。これにより、被験者が足底の圧力分布を全面的に理解するためのタイムリーな動作調整が可能となる。

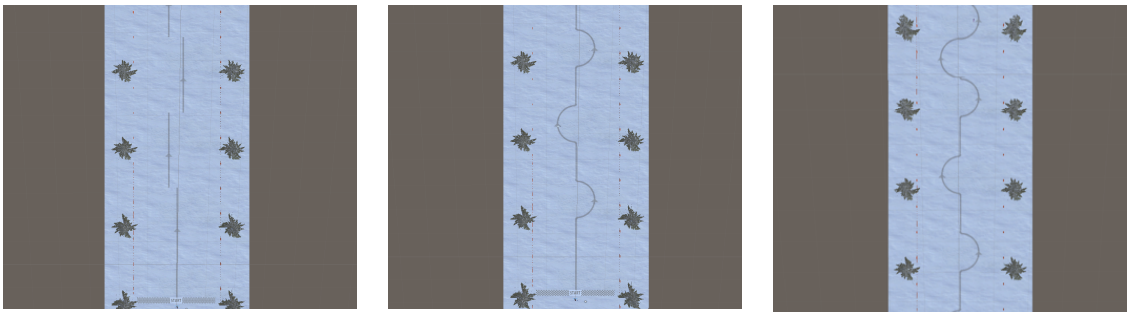
C、　視覚フィードバック機能なし： 黒い線に沿って、画面には干渉要素が含まれず、被験者はコースだけに集中できる。

初心者スノーボーダーにとって、どの視覚フィードバックが最も効果的かを、インストラクターとの練習やテストを通じて評価することを目的に、3 つの異なる視覚フィードバック機能が設計された。

### 3.4.3 コース設計

本研究では、異なる難易度を持つ三つのコースを開発した。このシステムは、利用者が自己の技能レベルに適したコースを選択できるように細心の注意を払って設計されている。目的は、技能の向上過程をより効率的かつ楽しい体験に変えることにある。個々の利用者が自己の能力に合わせた最適な学習経路を見つけ、技能向上を促進できるよう努めている。[9]

また、スノーボード技術の向上において、ターンの滑走は不可欠な技術要素である[1]。初心者が重心移動を習得を目指すことができるように、初心者が練習目標に集中し、技術の完成度を高めることを目的とした。



a.コース 1　　b.コース 2　　c.コース 3

図 3-9：コース設計

図 3-9 のように、コースを設計した。

a. 区間直線コース（簡単）：黒い線に沿って、練習初期段階で、初心者が迅速に慣れ、トレーニングシステムに適応するために、簡単な直線コースを設定した。この段階の目的は、被験者が練習前後の達成度の変化を観察することである。
b. 半ターンコース（普通）：訓練中期に、コースの難易度を上げ、滑走半ターンコースを採用した。重心移動技術の学習目標をより良く達成することを目指す。

c. 混合コース（稍困難）：直線、半ターン、連続ターンを混合し、難易度を高め、被験者の挑戦意欲を刺激し、トレーニングの面白さを増加させると目指している。同時に、実験を通じて、被験者のデータ表現上の変化を観察したいと考えている。

### 3.4.4 訓練方法

図 3-10 のように、訓練者は専用のスノーボード靴と圧力センサーを備えた装置を着用し、（a）シミュレーター上で横滑りの（b）模擬実験を行う[8]。実際のスキー場での横滑り動作（c）を模倣し、システムがスノーボード技術トレーニングに与える効果をより評価することを目的としている。

a、シミュレータ　　b、実際の練習方　　c、システム内の様子

図 3-10：トレーニング方法

## 3.5 予備実験

スノーボード初心者 3 名を対象に、革新的なアプローチを用いて実験を行い、改良したトレーニングシステムを検証した。このシステムは、実験者から重心データ、足圧データ、動きデータ、アバターの空間位置データなどを収集することができた。実験者が操作するアバターの空間位置データと、コースのベースライン位置データを比較することで、スノーボードのシミュレーション・トレーニングにおける各実験者のパフォーマンスと進捗状況を評価することを目指した。

表 3-1：予備実験の被験者

| 被験者 | スノーボード経験 | 性別 |
|---|---|---|
| 01 | なし | 男 |
| 02 | なし | 男 |
| 03 | 1 日 | 女 |

実験者は、足の圧力センサー付きの特殊なスノーシューを着用した状態で POWER SNOWBOARD SIMULATOR を使用した。このシステムの中で、被験者は 3 つの異なるコースでトレーニングをし、それぞれ 40 秒間の練習を 3 回行った（合計 9 回）。練習の最後に、被験者は 3 つのコースそれぞれでテストを行い、アンケートに答えた。最初の練習では、40 秒間の間に各実験者が生成した 2596 点の有効なリアルタイムの重心データを収集した。実験を通して、3 人の実験者によって合計 89,456 点の足圧データが得られた。

運動前後の被験者 1 の足圧と重心データを比較・分析したところ、大きな変化が観察された。図 3-11 に示されている通り、青い線は練習前の足圧データを表し、オレンジ色の線は練習後の足圧データを示す。運動前後の足底重心の差が大きくなっており、運動中に足底圧をより効果的にかけることができるようになったことがわかる。この結果は、本システムが人間の重心移動スキルのトレーニング学習に有効であることを示唆している。

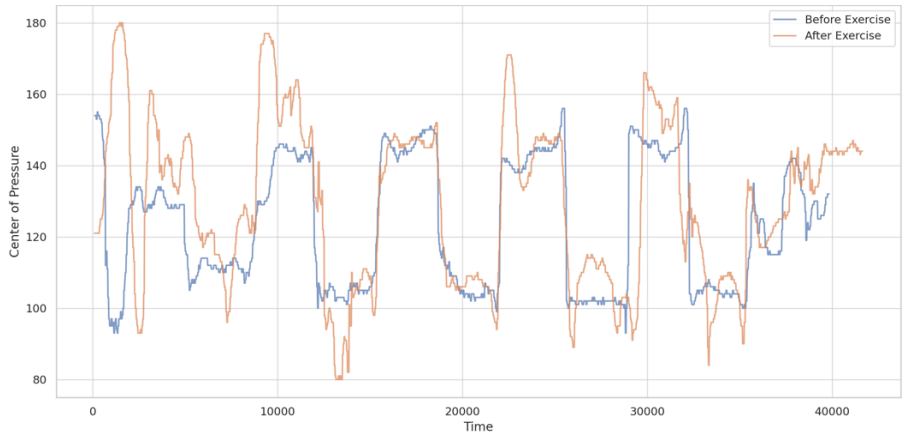



図 3-11：被験者 0 1 練習前後重心データ変化

実験の結果は、システムの改善提案にもつながった。実験者は、自分のパフォーマンスについてより明確なフィードバックを得たいと考えており、初心者向けの簡単な指導ステップや使用ガイドラインを追加するなど、システムによるガイダンスの充実を期待している。また、予備実験では、システムが圧力データによってアバターの動きを制御し、収集したデータを効果的な実験分析に利用できることを検証した。これは本実験の内容を改善させる上で重要である。

## 3.6 まとめ

システムの改良点としては、横滑り時のモーショントラッキングの安定性を向上させるため、モーショントラッキングカメラを追加した。この改善により、システムの精度と応答性が向上し、よりスムーズでリアルなユーザー体験が検証された。さらに、足底の圧力データを使ってバーチャル・キャラクターをコントロールする機能を実装した。この技術革新は、ユーザーとのインタラクションを強化するだけでなく、トレーニングの効果も向上させる。また、3種類の視覚フィードバックと3種類の雪上ルートを開発し、技能学習の楽しさと目的意識を高めることで、ユーザーが楽しみながら効果的に技能を向上できるようにした。

最後に、実験プロセス、データ収集、データ分析機能を検証するための予備実験を行った。収集したデータを分析することで、初心者が初めてスキーシミュレータを使用した際の実体験を得るとともに、彼らの感想やフィードバックを記録した。実験者の提案に基づき、トレーニングプロセスをより安全するために、ユーザーインストラクションなどの機能を追加する必要性を実感した。このアドバイスは、システムのさらなる改善と改良のために重要である。

# 第4章 実験・評価

## 4.1 本実験

### 4.1.1 実験の目的

システムの有効性を評価するため、定性データの収集に加えて、参加者がテスト中に達成した成果も得たいと考えている。そのため、インターフェースの視覚フィードバックとスキーのコースの複雑さをコントロールする変数制御実験を行うことにした。初心者が重心移動技術を学ぶ過程でシステムがどのように助けとなるかを評価するために、以下の三つの研究課題を設定した。これらの課題を通じて、システムの効果を体系的に分析し、初心者が重心移動技術を習得する際の支援効果を定量的および質的に評価することを目指す。

#### 4.2.1.1 研究問題

RQ1：スノーボードにおける重心移動技術の習得を支援するシステムは、どの程度効果的か？
RQ2：初心者にとって、どのような視覚的フィードバックが学習成果を向上させるのに最も効果的か？
RQ3：異なるコースの設計が学習やトレーニングの効果にどのような影響を与えるのか？

### 4.1.2 実験方法

今回、（表 4-1）合計 15 名の参加者（その中に 8 名が女性、7 名が男性）を募集した。これらの参加者の中で、8 人はスノーボード経験が全くなく、5 人は 2〜3 回のスノーボード経験があるが頻繁には行っていない。残りの 2 人は約 10 回のスノーボード経験があるが、まだ中級レベルには達していない。

表 4-1：本実験の被験者

| 被験者 | スノーボード経験 | 過去 5 年間のスポーツ経験 | 性別 |
|---|---|---|---|
| 1 | ～5 日 | 3 年以上 | 男性 |
| 3 | 0 日 | 1 年～3 年 | 女性 |
| 4 | 0 日 | 0 | 女性 |
| 11 | 0 日 | 1 年未満 | 女性 |
| 15 | 0 日 | 1 年未満 | 女性 |
| 2 | 5 日~10 日 | 3 年以上 | 男性 |
| 6 | ～5 日 | 3 年以上 | 女性 |
| 7 | 0 日 | 3 年以上 | 男性 |
| 9 | 0 日 | 1 年未満 | 男性 |
| 10 | 5 日~10 日 | 1 年～3 年 | 女性 |
| 5 | ～5 日 | 1 年未満 | 女性 |
| 8 | 0 日 | 1 年未満 | 男性 |
| 12 | ～5 日 | 0 | 男性 |
| 13 | 0 日 | 1 年未満 | 男性 |
| 14 | ～5 日 | 3 年以上 | 女性 |

実験条件と目標を詳細に説明した後、実験参加者はスノーボードの靴と圧力センサーを装着し、シミュレーター上で実験を行う必要がある。実験開始前に、参加者は地面上で装置を装着し、数回の予備練習を行い、実験装置と実験プロセスに慣れる。　その後、参加者を A、B、C グループにランダムに割り当てられ、以下の流れ（図 4 -1）に従って実験を行う：

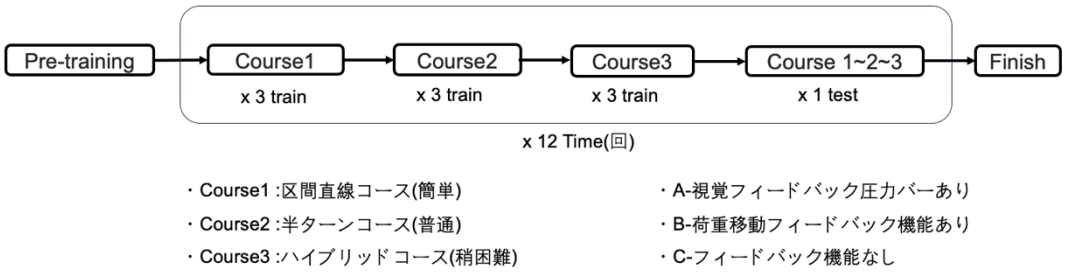



図 4-1：本実験の流れ

参加者は練習とテスト時間はそれぞれ 45 秒で各コースを 3 回練習し、合計 9 回行う。 全ての練習が終了した後、各コース 1 回の正式なテストを行う。全体の実験プロセス：事前準備（10 分）、実際の実験（30 分）、アンケート調査とインタビュー（10 分）

### 4.1.3 本実験の分析方法

本実験では、参加者から足の裏の圧力のリアルタイムデータ、身体動作データ、重心移動データ、システム内キャラクターの 3D 移動位置など多くのデータを収集した。達成度を計算するために、システム内アバターの位置データを使った。

データ処理：各人がシステムを起動する際の異なる遅延によるデータの誤差を減らすため、参加者が各練習とテストで 3 秒目から 35 秒目までの足の裏の圧力データを分析することにした。この方法で、合計 264,240 個の 3D 空間位置データポイントを収集し、後続の分析に使用した。

基準データの設定：トレーニングシステムのルート位置データが固定されているため、達成度の理論上の最大値は 100%に達することは不可能である。この基準に基づき、参加者が足の裏の圧力制御で得たシステム内アバターの位置データに対して、 2 乗平均平方根誤差（Root Mean Squared Error） [7]分析を行った。その後、これらの誤差データを 1 から 100 の範囲に標準化し、各参加者が各トレーニングで基準データに対して達成度$Pi$得点を計算した。
達成度$Pi$：

$$Pi = \frac{\sum_{t=1}^{T} 100(1 - \frac{|D_{it} - S_t|}{\max(x) - \min(x)})}{T}$$

*$D_{it}$被験者データ
*$S_t$基準データ
*$T$全時間:35649 ミリ秒

### 4.1.4 本実験の結果

被験者のデータに分散分析を通じて、異なる実験者のデータ間に顕著な差異があるかを検出した。(F-statistic :42.114179 P-value <0.001).

実験終了後、調査アンケートのデータに統計分析を行い、評価基準としてリッカート尺度（Likert Scale）[17]を使用した。アンケートを評価するために、クロンバックの α（Cronbach‘s Alpha）[18]値を計算した。全体的な分析結果は、平均 α 値が 0.681 であることを示している。注目すべきは、アンケートから第三の質問（Q3）を除外した後、 α 値が 0.746 に上昇したことである。 これらの結果は、調査アンケートのデータ全体の α 値が 0.7 に近く、全体的に尺度が比較的合理的な一貫性を示すことを示唆している。

RQ 1 ：

練習前後の足圧データの変化を観察するために、収集したデータに対して分散分析（ANOVA）を行った。参加者全員の練習前後の足の裏の圧力データの統計検定結果は、F 値=8.760027229、P 値=0.006204263 を示した。P 値が 0.05 の有意水準より小さいため、練習前後のデータには統計学的に顕著な差異が存在する。練習前後の被験者全員の平均 RMSE（誤差）値は 23.586%減少し、図

4-2 から、被験者の誤差が顕著に減少していることが分かった。このシステムがスノーボードの重心移動技術の学習に積極的な助けを提供していることを示した。

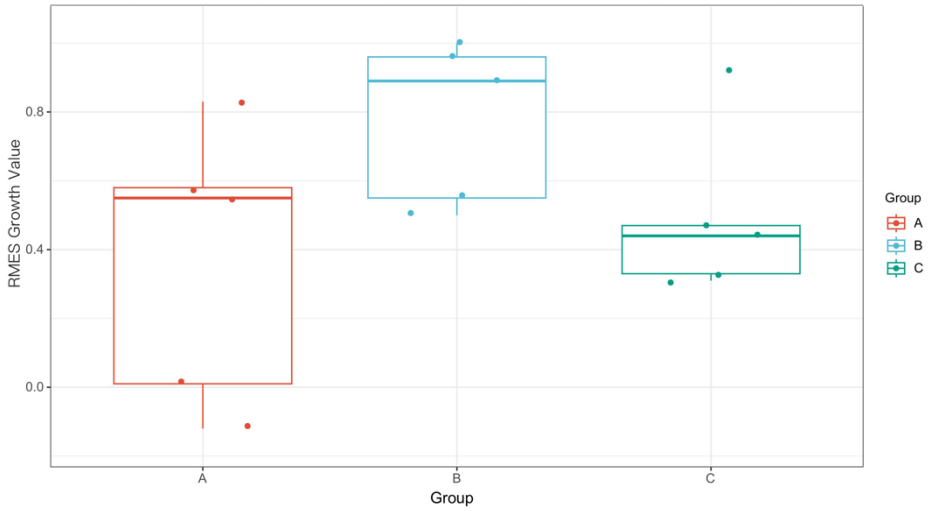



図 4-2：グループ ABC 練習前後 RMSE の差

RQ 2 ：

練習前後の成績を中央値で分析した：

図 4-3 の赤色と空色の部分の A グループの練習前後の成績中央値は 6.25 増加し、オレンジ色と灰色の部分の C グループの練習前後の成績中央値は 6.83 増加した。この向上は非常に顕著ではないが、A グループと C グループのメンバーのスキルが進歩したことを示している。特に C グループのデータは練習前後に一定の一貫性を示した。 緑色と濃紺色の部分の B グループでは、練習後の中央値の向上が 13.06 とより顕著だった。この結果は、B グループのメンバーが練習からより多くの利益を得ており、彼らの能力の向上が統計上より顕著であることを示した。実験者にとって、重心移動機能が彼らの学習成果をより効果的に向上させるようだ。

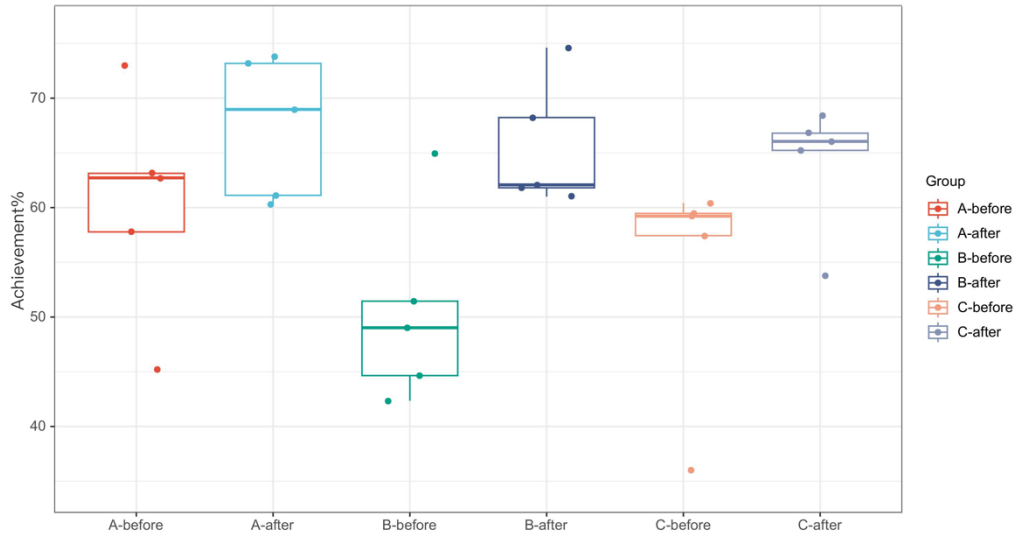



図 4-3：グループ ABC 練習前後コース達成度

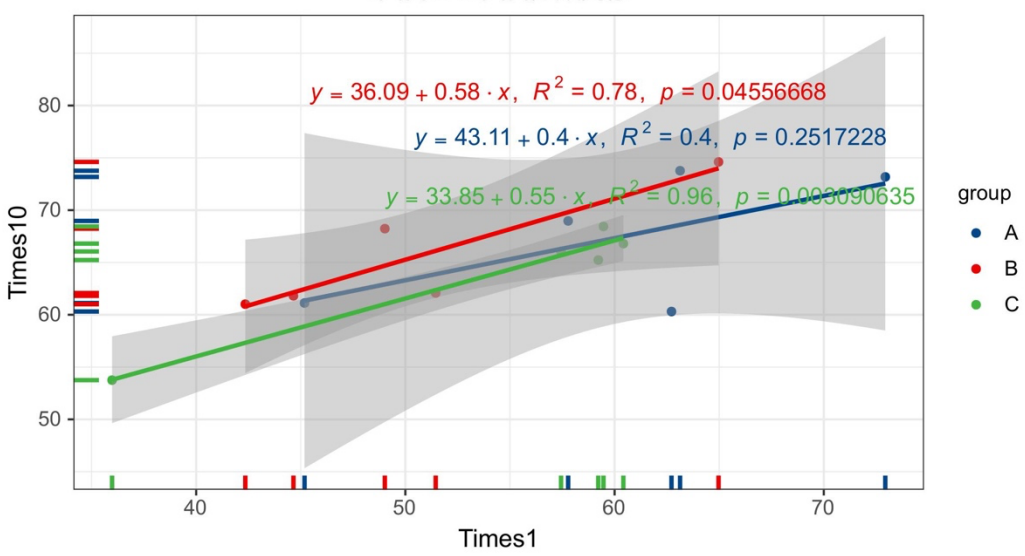



図 4-4：個々グループ分類

三つのグループのデータをそれぞれ分散分析し、観察した結果：

図 4-4 青色線のグループ A の決定係数 $R^2$は 0.4、P 値は 0.2517 である。A グループのデータはある程度の相関がある。赤色線のグループ B の $R^2$は 0.78、P 値は 0.0456。緑色線のグループ C の $R^2$は 0.96 で、P 値は 0.0003909635。この非常に高い $R^2$値は、グループ C のデータ間に非常に顕著な正の相関関係が存在することを示した。

図 4-4 による個々グループの被験者の成績を標準偏差で観察した結果：

B グループの標準偏差は 6.538、C グループの標準偏差は 3.995。これに基づき、C グループと B グループに比較することで、初心者の達成度の助けにおいてより一貫性があると考えられた。これらのデータの観察に基づき、実験者にとって、グループ C のトレーニング方法が成績向上に最も効果的であると推測できた。しかし、異なる初心者にとって、最適なフィードバックのタイプは、彼らの学習能力、初期のスキルレベル、技術への適応性にも依存する可能性がある。

### 4.1.5 RQ 2 質的検証

RQ 2 ：そこで、どの視覚フィードバックが初心者にとってより効果的かをより判断するために、ABC の三つのグループから完全に経験がない参加者を選び、三種類の異なるフィードバック機能を持つシステムでそれぞれ 2 回ずつ練習に参加し、練習終了後に各被験者に 20 分間の半構造インタビューを行った。3 人のインタビュー内容を録音し、テキスト化して、イベント分析を深く行うためにコード化した。インタビューデータを分析するために、オープンコーディング、アクシャルコーディング、セレクティブコーディング[16]技術を使用した。

視覚的フィードバック（A）： 参加者の中には、このモードでは、キャラクターや自分の軌跡だけに集中し、追加の視覚的情報がないと指摘する人もいた。このモードは、十分なガイダンスやフィードバックがないため、一部の人にとっては単純すぎるかもしれない。
視覚的フィードバック(B)： UI 要素やプレッシャーマップのような複雑なビジュアルフィードバックは、一部の参加者にとってより有用であった。これらの要素は、力配分や重心移動をより明確に表現していると感じたようだ。
しかし、視覚フィードバックを処理しながら動作や経路を完了させることは困難であることがわかった。
視覚的フィードバック機能なし(C)： 参加者は、視覚的な手がかりがないと、自分のパフォーマンスや改善点を評価することが難しいことを示した。しか

し、初心者にとっては、完成度と結果に集中することが非常に有益であり、注意散漫を減らし、初期スキルの習得を助けることがわかった。

全体的に、ほとんどの参加者は、練習の指針となるより詳細で直感的な情報を提供する複雑な視覚的フィードバック（B）を学習に好む傾向があった。しかし、一部の参加者は、フィードバックに対する学習者のニーズや好みの違いを反映し、視覚的フィードバックなしによって提供されるフォーカスを好んだ。

RQ３：

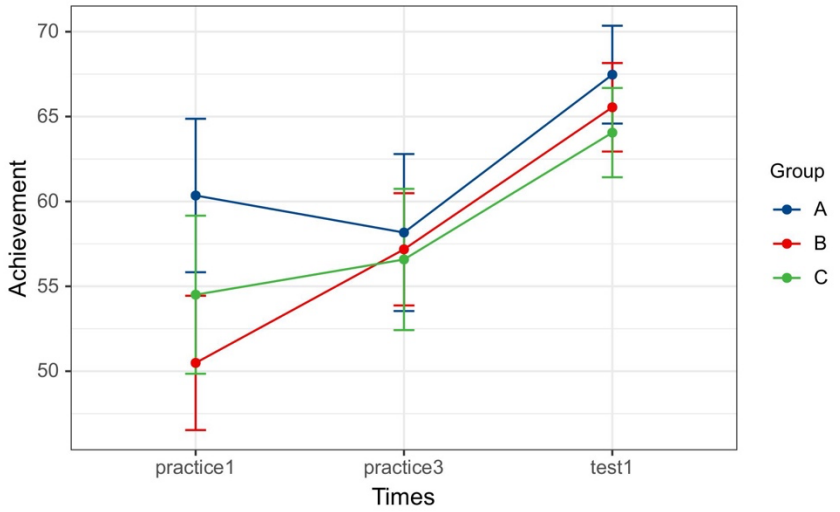



図 4-5：グループ ABC 区間直線コース達成度誤差

コース１の分析：図 4-5 を見ると、練習の回数が増えるにつれて、コース１の直線区間での成績が向上していることがわかる。特に、B グループの成績向上が最も顕著で、平均成績向上は 15.06 に達した。この観察から、初心者が自信をつけるのに大いに役立つ、簡単なコースでの練習者の上達ぶりがよくわかる。

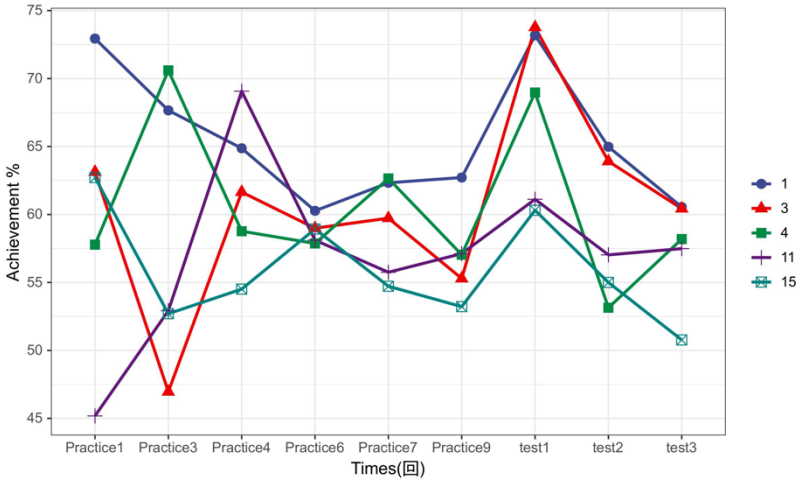



図 4-6：グループ A 個々被験者達成度

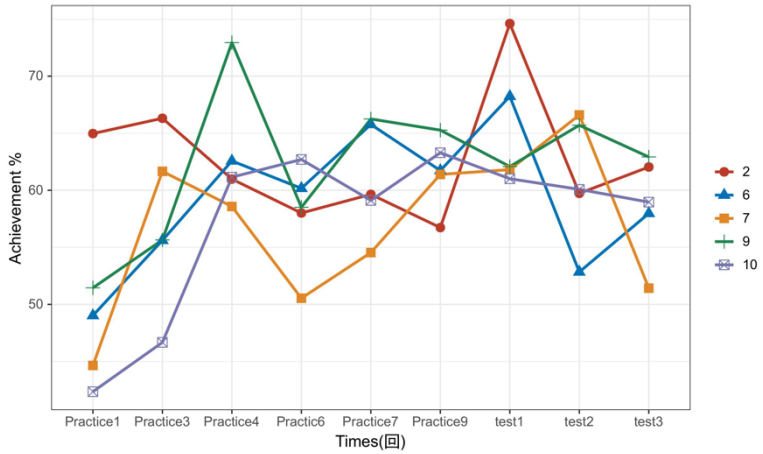



図 4-7：グループ B 個々被験者達成度

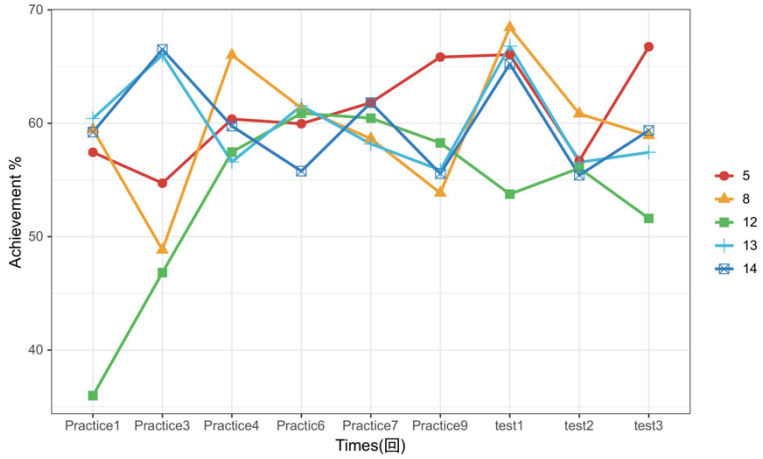
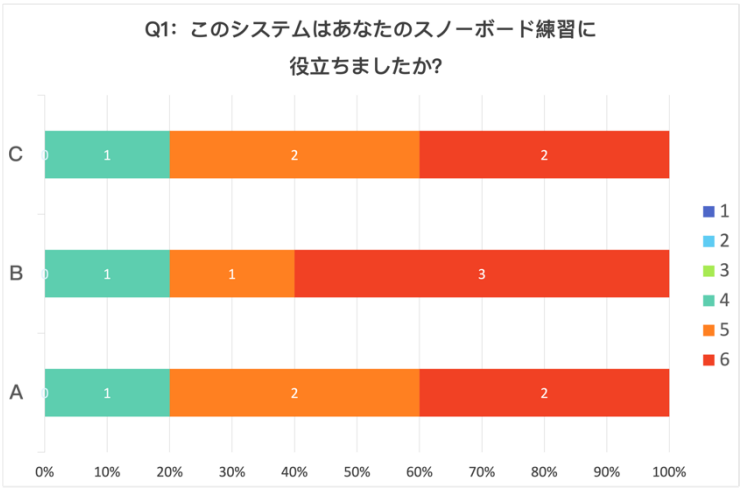


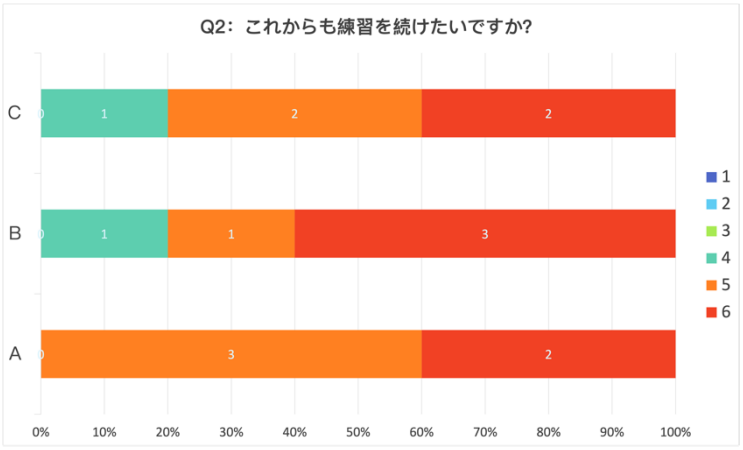

図 4-8：グルグループ C 個々被験者達成度

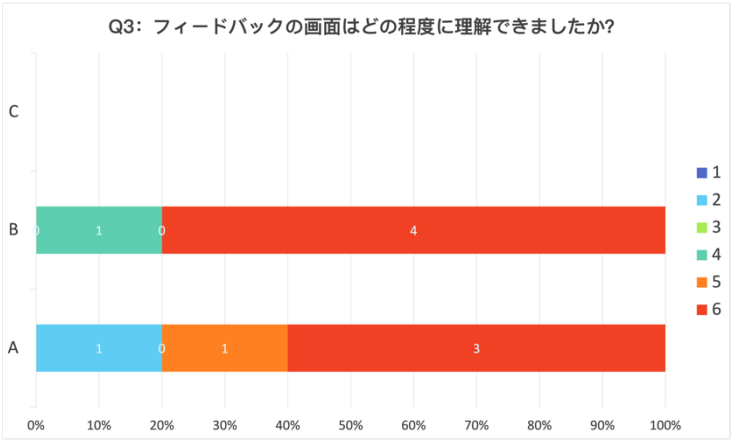
図の点線は、被験者がコース 123 での 6 回の練習と 3 回のテストの達成度の結果をそれぞれ表した。
図 4-8 に示されているように、
コース 1：P1、P3（P 練習）と T10（T テスト）。コース 2：P4、P6、T11。
コース 3：P7、P8、P12。
図 4-8、図 4-7、図 4-8 を観察することで、各グループの被験者の達成度データを見ると、練習の P3 から P4 回目の間に、コースの難易度が上がるときに、7 名の被験者の達成度が向上していることが分かる。また、A グループの（被験者 1、3）、B グループの（6、10）、C グループの（8、12、5）が成績の顕著な向上を示した。完全に経験のない者は 85.7%を占めている。3 回の練習後の P7～P9 の時点で、9 名の被験者の成績が顕著に下降している。この結果は、困難な挑戦に直面したとき、学習進度が常に線形であるとは限らないことを示した。学習者が異なるコースに直面する際、様々な挑戦に遭遇する可能性があり、これらの挑戦が成績の変動を引き起こす可能性がある。

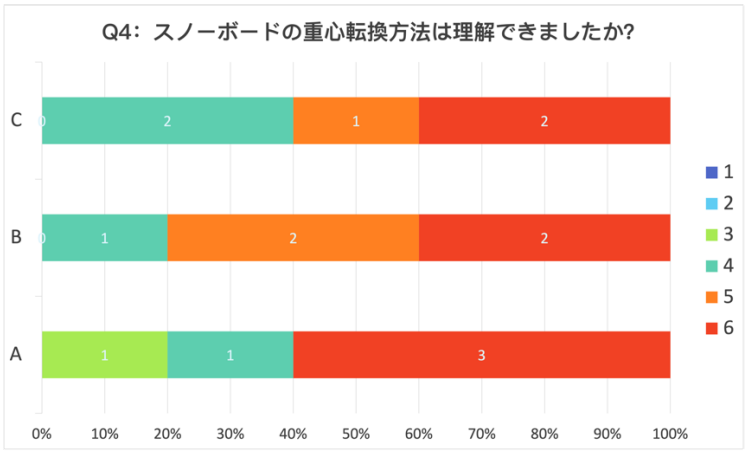
### 4.1.6 アンケート結果

アンケート結果を図 4-9 に示す。




Q2：これからも練習を続けたいですか?
C
B
A
0
1
2
2
0
1
1
3
0
3
2
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
1
2
3
4
5
6


Q3：フィードバックの画面はどの程度に理解できましたか?
C
B
A
0
1
0
4
1
0
1
3
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
1
2
3
4
5
6


Q4：スノーボードの重心転換方法は理解できましたか?
C
B
A
0
2
1
2
0
1
2
2
1
1
3
0%
10%
20%
30%
40%
50%
60%
70%
80%
90%
100%
1
2
3
4
5
6

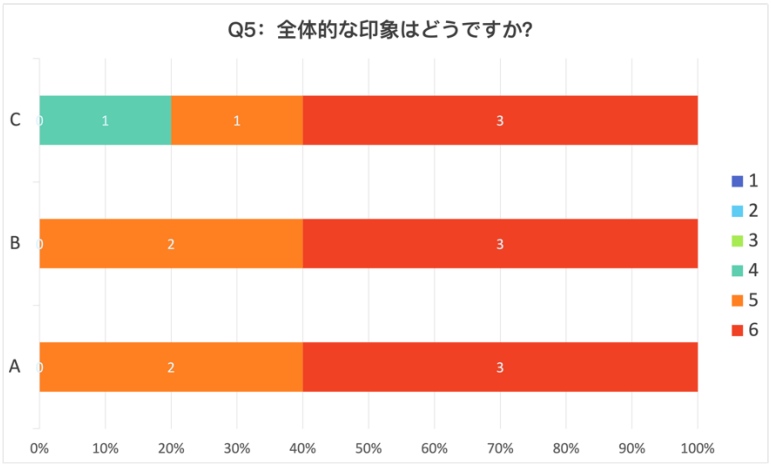



図 4-9：アンケート結果

各グループの事後アンケート結果（図 4-9）を分析する：

「(Q1)このシステムはスノーボードの練習に役立ちましたか？」参加者全員が 4 以上の評価を選んだ。これは、このトレーニングシステムが学習者の総合的なスキル向上に有効であると参加者が概ね感じていることを示した。

「(Q2）今後も練習を続けたいですか？」全グループが 4 以上を選択した。本システムが学習者のモチベーションを高め、練習の継続を促す効果があることを反映した。

「(Q3） フィードバック画面はどの程度理解できたか。」

グループ A では 1 人が評価 2 を選んだが、説明がない場合、視覚的フィードバックの理解度は異なる可能性があり、グループ B では 4 人が評価 6 を選んだ。 グループ AB でフィードバックに差があったということは、各グループでフィードバックのわかりやすさや有効性を感じているということだが、足底受容力では、絵による視覚的フィードバックの方がわかりやすい。

（Q4）スノーボードの重心転換方法は理解できましたか？」

各グループは技術の理解システムと練習内容については理解しているが、その理解度は異なる。これは、異なる視覚フィードバック方法の効果や理解度がさらに改善する必要性がある。

「（Q5）全体的な印象どうですか？」C グループの全員が 4 以上の評価を与え、AB グループの回答者は全員が 5 以上の評価を与えた。全体的に、回答者は重心移動練習システムに対して肯定的な印象を得た。

## 4.2 考察

第 4.1 節では、本研究が行った観察に基づき、スノーボードシミュレーターを使用し、足底データによって駆動されるアバターを操作して重心移動技術の練習を行うシステムの開発について述べている。また、本システムの有意さを検証するために、３つの主要な研究課題に焦点を当て、それらの課題に対する探究を行った。

研究課題１（RQ1）では、圧力データを活用した成績の定性分析を実施し、本システムが重心移動技術の練習に対して高い評価を受けていることを確認した。研究課題２（RQ2）では、質的評価として被験者へのインタビューを実施し、視覚的フィードバックが技能練習に対しては効果的である。しかし、その効果には個人差が存在することが明らかになった。 研究課題３（RQ3）では、被験者の足圧データを分析し、各コースにおける成績を評価した。その結果、簡単なコースにおいては成績が顕著に向上しているものの、複雑なコースでの成績は優れていないことが観察されました。しかし、技能の向上が複雑なコースを経験する際、技能の向上が必ずしも線形でない可能性を示唆している。初心者がこのシステムを利用して学習する場合、より個人に合ったトレーニング計画が必要であると考えている。

最後のアンケート（図 4-9）結果について、研究問題に関する最終的な考察として、本研究では実験結果を導出するために収集されたアンケートデータを詳細に分析した。この分析により、多くの参加者が本システムに対して高い関心と肯定的な評価を示していることが明らかになった。しかし、同時に、システムの改善が必要とされる多くの領域も浮き彫りになった。これらの領域に関する具体的なフィードバックは、自由記述（表 4-2）にて詳細に述べられている。

表 4-2：被験者感想記録

| 被験者 | 感想・自由回答 |
|---|---|
| 回答１ | 雪の道が狭すぎる、少し力を使って、反応が敏感すぎる |
| 回答２ | 音声プロンプトの欠如、および周囲の音楽 |
| 回答３ | 速度を提示してほしい |

| | |
|---|---|
| 回答 4 | 身長に合わせて、足と足の間の距離を調整できるようにしてほしい |
| 回答 5 | VR 機能の追加した方がよい、シミュレーターには実際のスキーの景色やゲレンデの心理的負担がない |
| 回答 6 | ボードが回転するとうれしい |
| 回答 7 | 実際の滑走はもっと広く雪面を見渡せるので、マシンで滑走するときの可動域は実際の滑走より狭くなることが予想される |
| 回答 8 | 乗り降りに介助が必要なこと、マシンの上を滑るときに手すりを持つ必要があることなど、マシンの操作性はあまりよくない |
| 回答 9 | 目新しさがあり、初めて学ぶ人にはとても役に立つ |
| 回答 10 | 振動フィードバック、アンビエントサウンドを追加すれば、より良くなるだろう |
| 回答 11 | 運動の最後には筋肉が緊張し、いいトレーニングになった。 本格的なスノーボードに対する恐怖心を克服する |
| 回答 12 | 動きの合図やより正確なフィードバックがあればもっといいし、運動の結果を知るための採点システムがあればいいと思う |
| 回答 13 | コースデザインは、目標なしに滑るよりいいし、スノーボードを学び始めるのにとても役立つ |
| 回答 14 | 練習中の手すりに頼りすぎる |
| 回答 15 | バランスと重心の練習にはいいが、他の難しいシーンを加えられるともっといい |

## 4.3 まとめ

本章では、スノーボードシミュレータ学習支援システムの評価実験について述べる。被験者はスノーボード初級者 15 名であり、彼らに対する実験とデータ分析により、システムの有効性を評価した。実験の結果、本システム用いて重心移動の運練を行った被験者全員において、技術の向上が見られた。なかでも、視覚フィードバック機能を使わずにシステムを使用した実験グループが、最も顕著な技能向上を示した。一方、実験者は、複雑な荷重移動機能ありシステムをより好感度が上がる傾向を見られた。

半構造化インタビューを通じて、視覚フィードバック機能に対する初心者の好みは人によって異なることがわかった。あるグループは、荷重移動の複雑なヒートマップを持つシステムを好み、将来のスキルアップに役立つと考えていた。もう一方は、視覚的フィードバックのないシステムの方が、よりパフォーマンス向上に集中でき、結果としてスキルアップが達成できると考えていた。

アンケートの結果によると、すべての実験グループが高い評価を与えた。また、スノーボード初心者の重心移動スキルの習得を効果的に促進することができたと考えられる。最後に、システムの改善点についても提案を行った。現時点では、本システムは主にスノーボード初心者に適用可能であるが、将来的には、これらの提案に基づき、本システムにおける視覚、聴覚、触覚フィードバックに関する研究を行うことで、よりリアルなシミュレーション体験を提供し、中・上級レベルのスノーボード学習を支援することが期待される。

# 第5章 おわりに

## 5.1 本研究のまとめ

本研究では、主に初心者を対象としたスノーボードシミュレータ上での重心移動技術トレーニングのための学習支援システムを提案した。

本研究では、スキートレーニングシステムの改良版をスノーボードに初めて適用し、圧力センサを用いてユーザの足圧データを収集し、足圧データを用いてアバターの動きを制御という機能を実現した。圧力の視覚フィードバック機能とコースを 3 つの異なる複雑度で設計・実装し、収集したデータのフィードバックをリアルタイムで同期させた。最後に、15 人の初心者を対象とした実験観察とデータ分析を通じて、継続的な学習に対するシステムの有効性を検証した。インタビュー調査の結果、荷重移動の複雑なヒートマップによる視覚的フィードバックは、概ね今後の学習に役立つと認識されていた。しかし、荷重データを基準データとの比較から分析した結果、視覚的フィードバック機能を持たないシステムが、練習中の初心者の成績向上を最も一貫して示した。そこで、スノーボード初心者が技術を習得するためには、注意散漫を減らし、練習に集中できるシステムがより効果的である可能性を示唆している。

本研究の独創性は、スノーボードの重心移動スキルの学習に足の圧力データ駆動にキャラクターを使用した点にある。このアプローチにより、学習の直感性を向上させるだけでなく、トレーニングの実用性と効果を高める可能性があると考える。この研究は、学習プロセスの直感的な側面と効率性を強化し、より良いトレーニング結果をもたらす方法論の開発に焦点を当てている。

## 5.2 今後の展望と課題

スノーボードシミュレータについては、クロスキーの滑降における重心移動スキルのトレーニングに重点を置いた POWER SNOWBOARD SIMULATOR を使用した。このトレーニング方法は、初心者が左右の足の力配分をバランスよく行い、体の感知と重心位置の認識する目的としている。ただし、このシステムは初心者に最も適している。円弧滑走など、より高度なトレーニングをシ

ミュレーター上で行いたいユーザーにとっては、現在のシミュレーターでは一定のリスクや困難がある。将来的には、ボードの回転や傾きなどの動きをシミュレーターに加えることで、シミュレーションの臨場感を向上させることも考えられる。

実装システムでは、足底圧センサーを用いてユーザーの圧力データを収集し、バーチャルキャラクターを駆動させる。しかし、人それぞれ足の大きさが異なるため、足底の各位置の圧力データを完全に正確に収集できない場合がある。両足間の重心を計算することで調整を行い、より正確にシステムキャラクタを制御しているが、将来的には、ユーザーの足のサイズに合わせて適切な圧力センサーを選択し、データ収集の精度を向上させる必要がある。

視覚フィードバックに関しては、人の身体認識と実際の動作との間に乖離があることはよくある問題である。視覚フィードバックがないことで、ユーザーはより運動の完遂に集中することができるが、スキルの向上という点では、学習プロセスの再現としての視覚フィードバックは、自分の身体の動きやストレス状態を再確認するために重要である。将来的には、視覚フィードバックに加えて、音、振動、触覚フィードバックもスキー技術の学習に良い影響を与える可能性がある。

実験方法について、実験は 2 週間にわたって行われ、その間に 15 人の初心者が 12 回のトレーニングとテストを受けた。シミュレーターでのトレーニングは 1 回 30 分であった。そのため、技術習得のレベルが不十分であった可能性がある。したがって、システムの改善と技能習得の成果をより十分に評価するためには、今後より多くの参加者を募る必要がある。

さらに、参加者全体のうち、スノーボード経験者がある人はわずか 6 名であった。参加者は実験終了時に実際の雪原でスノーボードをしなかったため、シミュレータでの練習が実際のスキー学習にどの程度影響したかを判断することはできなかった。したがって、スノーボード・シミュレーター・システムの実際の有効性を評価し、さらに最適化・改善するためには、すでにスキー場で滑った実験者とインタビューや実験を繰り返すことが不可欠であると考える。

# 謝辞

本研究の遂行にあたり、東京工業大学情報理工学院の小池研究室で開発された[Vzski]と北陸先端科学技術大学院大学金井研究室で小野らの研究[6]に基づく圧力センサー装置を使用しました。

この場を借りて、主指導教員である金井秀明教授に心からの感謝を申し上げます。会議中には辛抱強く指導をしていただき、研究に対する新たな視点を提供してくださいました。研究過程で学んだこと、特にスノーボードシミュレータ装置、システム開発、実験設計、データ分析などの分野において、金井教授から多大な助言とサポートをいただきました。研究の目的と方法を深く理解することができ、心から感謝しています。

また、研究室の先輩である小野重遥さんには、システム設計、データ取得、コードの記述、データ分析及びグラフ作成などの面で大変お世話になりました。先輩の無私の支援により、私はこの研究を完成させることができました。

JAIST の田中研究室の王驰野先輩にも深く感謝します。深夜にもかかわらず、深度カメラを用いた動作追跡データの同期作業で支援していただき、C#の学習やコードの修正、食事を共にし、プログラミングに関する多くの貴重なことを教えてくれました。王さんに深く感謝いたします。

最後に、本実験に積極的かつ熱心に参加してくれた学生の皆さんに心からの感謝を申し上げます。忙しい学業の中でスノーボードへの情熱を持ち続けてくれた皆さんの熱意は、非常に高く評価されるべきものです。皆さんの参加があったからこそ、このシステムは継続的に改善され、完成していくことができるのです。

# 付録

## Microsoft Azure Kinect DK カメラ同期プログラム全

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Text;
using System.Threading.Tasks;
using Microsoft.Azure.Kinect.BodyTracking;
using Microsoft.Azure.Kinect.Sensor;
using UnityEngine;


public class AvatarTracker : MonoBehaviour {
    // GameObjects
    #pragma warning disable 0649

    // [SerializeField]
    [SerializeField]
    private GameObject poleLeft;
    [SerializeField]
    private GameObject poleRight;
    [SerializeField]
    private GameObject footLeft;
    [SerializeField]
    private GameObject footRight;

    #pragma warning restore 0649

    // Kinect
    private Device kinect_master;
    private Device kinect_sub;
    private Tracker main_tracker;
    private Tracker side_tracker;
    private Animator animator;
    private Capture[] capture;
    private Frame main_frame;
    private Frame side_frame;
    private Skeleton main_skeleton, side_skeleton;
    private JointId jointId;
    private Microsoft.Azure.Kinect.BodyTracking.Joint main_joint, side_joint;
    private Quaternion quaternion, main_quaternion, side_quaternion, average_quaternion, footLeftRot,
footRightRot;
    private System.Numerics.Vector3 main_jointPos, main_footLeftPos, main_footRightPos;
    private System.Numerics.Vector3 side_jointPos, side_footLeftPos, side_footRightPos;
    private System.Numerics.Vector3 jointPos, footLeftPos, footRightPos;
    // private Vector3 vector3;
    // private Transform tfAnimator, tfAvatar;
    private Transform tfAnimator;

    // Folder
    private StreamWriter writer;

    // TimeStump
    private int time;

    // for writing sensor values
    private StringBuilder sb = new StringBuilder();

    // for indexing cameras
    private int device_num;
    private List<Device> devices_list;

    // for camera configurations
    private DeviceConfiguration device_config;
    private WiredSyncMode sync_mode;
    private TimeSpan delay_off_master_usec;

    // for defining devices
    private bool master_found;
    private Device master_device;
    private List<Device> subordinate_devices;

    void Start() {
      Application.targetFrameRate = 30;
```

```
  try
  {
    StartDevices();
  }
  catch (Exception e)
  {
    Debug.Log($"Unable to start the device.");
    Debug.Log(e);
    throw e;
  }

  try {
    StartSyncCameras(master_config, sub_config);
  }

  catch(Exception e) {
    Debug.Log($"Unable to start the camera.");
    throw e;
  }

  // prepare a tracker
  main_tracker = Tracker.Create(
  master_device.GetCalibration(),
  new TrackerConfiguration() {
    ProcessingMode = TrackerProcessingMode.Cuda,
    SensorOrientation = SensorOrientation.Default
  });

  side_tracker = Tracker.Create(
  subordinate_devices[0].GetCalibration(),
  new TrackerConfiguration() {
    ProcessingMode = TrackerProcessingMode.Cuda,
    SensorOrientation = SensorOrientation.Default
  });

  animator = GetComponent<Animator>();
  quaternion = new Quaternion(0,0,0,0);
  footLeftRot = footLeft.transform.rotation;
  footRightRot = footRight.transform.rotation;

  // create folder and log-file
  writer = new StreamWriter(string.Concat(Initialize.saveLocation, "log_posture.csv"),false);

  // set capacity of the stringbuilder
  sb.Capacity = 1000;
}

private readonly DeviceConfiguration sub_config = new DeviceConfiguration{
    ColorFormat = ImageFormat.ColorBGRA32,
    ColorResolution = ColorResolution.R720p,
    DepthMode = DepthMode.WFOV_2x2Binned,
    CameraFPS = FPS.FPS30,
    SynchronizedImagesOnly = true,
    DepthDelayOffColor = TimeSpan.Zero,
    WiredSyncMode = WiredSyncMode.Subordinate,
    SuboridinateDelayOffMaster = TimeSpan.FromTicks(5000),
    DisableStreamingIndicator = false
};

private readonly DeviceConfiguration master_config = new DeviceConfiguration{
    ColorFormat = ImageFormat.ColorBGRA32,
    ColorResolution = ColorResolution.R720p,
    DepthMode = DepthMode.WFOV_2x2Binned,
    CameraFPS = FPS.FPS30,
    SynchronizedImagesOnly = true,
    DepthDelayOffColor = TimeSpan.Zero,
    WiredSyncMode = WiredSyncMode.Master,
    SuboridinateDelayOffMaster = TimeSpan.Zero,
    DisableStreamingIndicator = false
};

void StartSyncCameras(DeviceConfiguration master_config, DeviceConfiguration sub_config) {
  foreach(Device device in subordinate_devices){
    device.StartCameras(sub_config);
  }

  master_device.StartCameras(master_config);
}

void StartDevices() {
  master_found = false;

  device_num = Device.GetInstalledCount();
  Debug.Log(device_num + " devices detected.");
```

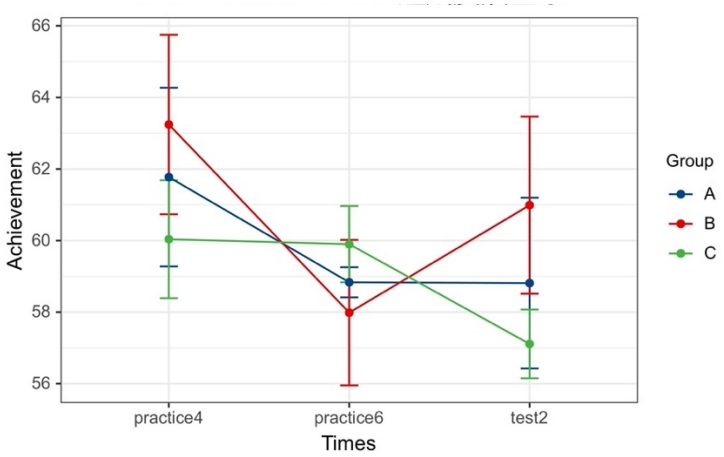
```
    devices_list = new List<Device>();

    for (int i = 0; i < device_num; i++){
      try
      {
        devices_list.Add(Device.Open(i));
      }
      catch (AzureKinectOpenDeviceException e)
      {
        Debug.Log($"Failed to open device {i}!");
        throw e;
      }
    }

    subordinate_devices = new List<Device>();

    foreach (Device device in devices_list)
    {
      device.SetColorControl(ColorControlCommand.ExposureTimeAbsolute, ColorControlMode.Manual, 30);

      if (device.SyncOutJackConnected & !device.SyncInJackConnected & !master_found) {
        master_found = true;
        master_device = device;
      }
      else if (!device.SyncInJackConnected & !device.SyncOutJackConnected) {
        Debug.Log("Each device must have sync in or sync out connected for synchronization!");
      }
      else if (!device.SyncInJackConnected) {
        Debug.Log("Non-master camera found that doesn't have the sync in port connected!");
      }
      else {
        subordinate_devices.Add(device);
      }
    }

    if (!master_found) {
      Debug.Log("No device with sync out connected found!");
    }
  }

  private Capture[] capture_list;
  private int current_index;
  private const double WAIT_FOR_SYNCHRONIZED_CAPTURE_TIMEOUT = 60000;
  private const int MAX_ALLOWABLE_TIME_OFFSET_ERROR_FOR_IMAGE_TIMESTAMP = 100;
  Capture[] GetSynchronizedCaptures(DeviceConfiguration sub_config){
    capture_list = new Capture[subordinate_devices.Count + 1];

    current_index = 0;
    capture_list[current_index] = master_device.GetCapture();
    ++current_index;

    foreach (Device device in subordinate_devices)
    {
      capture_list[current_index] = device.GetCapture();
      ++current_index;
    }

    if (subordinate_devices.Count == 0)
    {
      return capture_list;
    }

    bool have_synced_images = false;

    DateTime start = DateTime.Now;
    while (!have_synced_images)
    {
      TimeSpan duration = DateTime.Now - start;
      double duration_ms = duration.TotalMilliseconds;
      if (duration_ms > WAIT_FOR_SYNCHRONIZED_CAPTURE_TIMEOUT)
      {
        Debug.Log("ERROR: Timeout waiting for synchronized captures.");
      }

      Image master_color_image = capture_list[0].Color;
      double master_color_image_time = master_color_image.DeviceTimestamp.TotalMilliseconds;

      for (int i = 0; i < subordinate_devices.Count; ++i) {
        Image sub_image;

        sub_image = capture_list[i+1].Color;

        if (master_color_image != null && sub_image != null)
        {
          double sub_image_time = sub_image.DeviceTimestamp.TotalMilliseconds;
          double expected_sub_image_time = master_color_image_time
```

```
                                    + sub_config.SuboridinateDelayOffMaster.TotalMilliseconds
                                    + sub_config.DepthDelayOffColor.TotalMilliseconds;
          double sub_image_time_error = sub_image_time - expected_sub_image_time;

          if(sub_image_time_error < -MAX_ALLOWABLE_TIME_OFFSET_ERROR_FOR_IMAGE_TIMESTAMP)
          {
            capture_list[i + 1] = subordinate_devices[i].GetCapture();
            break;
          }
          else if (sub_image_time_error > MAX_ALLOWABLE_TIME_OFFSET_ERROR_FOR_IMAGE_TIMESTAMP)
          {
            capture_list[0] = master_device.GetCapture();
            break;
          }
          else
          {
            if (i == subordinate_devices.Count - 1)
            {
              have_synced_images = true;
            }
          }
        }
        else if (master_color_image == null)
        {
          Debug.Log("Master image was bad!\n");
          break;
        }
        else if (sub_image == null)
        {
          Debug.Log("Subordinate image was bad!\n");
          capture_list[i + 1] = subordinate_devices[i].GetCapture();
          break;
        }
      }
    }

    return capture_list;
  }

  void FixedUpdate(){
    // get capture from Kinect
    capture = GetSynchronizedCaptures(sub_config);
    main_tracker.EnqueueCapture(capture[0]);
    main_frame = main_tracker.PopResult();

  void Update() {
    // get elapsed time
    time = (int)Math.Round(Time.time*1000);
    if(main_frame != null){
      if (main_frame.NumberOfBodies > 0) {
        main_skeleton = main_frame.GetBodySkeleton(0);

        foreach (KeyValuePair<HumanBodyBones, JointId> pair in boneJointMap) {
          jointId = pair.Value;
          main_joint = main_skeleton.GetJoint(jointId);
          quaternion.Set(main_joint.Quaternion.X, main_joint.Quaternion.Y, main_joint.Quaternion.Z,
main_joint.Quaternion.W);

          tfAnimator = animator.GetBoneTransform(pair.Key);
          if(tfAnimator){
            tfAnimator.rotation = quaternion.normalized * this.GetQuaternionOffset(jointId);
          }

          sb.Length = 0;
          sb.Append(time.ToString());
          sb.Append(',');
          sb.Append(pair.Key);
          sb.Append(',');
          sb.Append((quaternion.x).ToString("f2"));
          sb.Append(',');
          sb.Append((quaternion.y).ToString("f2"));
          sb.Append(',');
          sb.Append((quaternion.z).ToString("f2"));
          sb.Append(',');
          sb.Append((quaternion.w).ToString("f2"));
          writer.WriteLine(sb);
        }

        jointPos = main_skeleton.GetJoint(JointId.Pelvis).Position;
        footLeftPos = main_skeleton.GetJoint(JointId.FootLeft).Position;
        footRightPos = main_skeleton.GetJoint(JointId.FootRight).Position;

        main_jointPos = main_skeleton.GetJoint(JointId.Pelvis).Position;
        main_footLeftPos = main_skeleton.GetJoint(JointId.FootLeft).Position;
        main_footRightPos = main_skeleton.GetJoint(JointId.FootRight).Position;
```

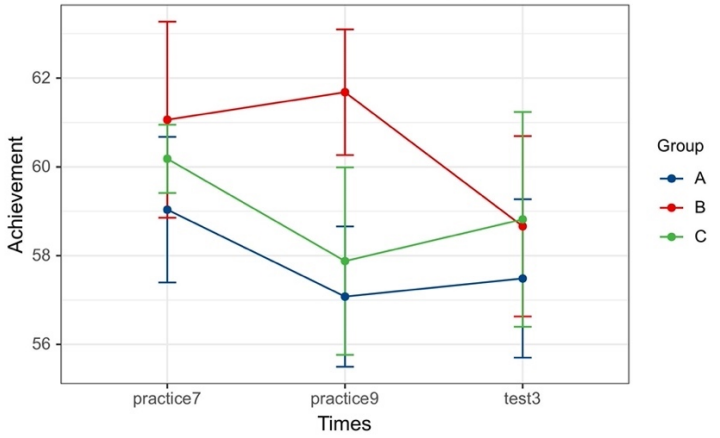
```
        side_jointPos = side_skeleton.GetJoint(JointId.Pelvis).Position;
        side_footLeftPos = side_skeleton.GetJoint(JointId.FootLeft).Position;
        side_footRightPos = side_skeleton.GetJoint(JointId.FootRight).Position;

        jointPos = (main_jointPos + side_jointPos) / 2;
        footLeftPos = (main_footLeftPos + side_footLeftPos) / 2;
        footRightPos = (main_footRightPos + side_footRightPos) / 2;

        tfAvatar = avatarRoot.transform;
        vector3.Set(0, -(jointPos.Y - Math.Max(footLeftPos.Y, footRightPos.Y)) / 10000f, 0);
        tfAvatar.localPosition = vector3;

        sb.Length = 0;
        sb.Append(time.ToString());
        sb.Append(',');
        sb.Append("position");
        sb.Append(',');
        sb.Append((-jointPos.X / 200f).ToString("f2"));
        sb.Append(',');
        sb.Append((-(jointPos.Y - Math.Max(footLeftPos.Y, footRightPos.Y)) / 10000f).ToString("f2"));
        writer.WriteLine(sb);

        footLeft.transform.rotation = footLeftRot;
        footRight.transform.rotation = footRightRot;
      }
    }
  }

  private void OnApplicationQuit() {
    foreach(Device device in devices_list){
      device.StopCameras();
      device.Dispose();
    }
    writer.Flush();
    writer.Close();
    System.GC.Collect();
  }

  private Quaternion GetQuaternionOffset(JointId jointId) {
    switch (jointId)
    {
        case JointId.Pelvis:
        case JointId.SpineChest:
        case JointId.SpineNavel:
        case JointId.Neck:
        case JointId.Head:
            //return Quaternion.Euler(90, 0, 90);
            return Quaternion.Euler(180, 0, 0);
        case JointId.HipLeft:
        case JointId.KneeLeft:
        case JointId.AnkleLeft:
            //return Quaternion.Euler(90, 0, 270);
            return Quaternion.Euler(0, 0, 180);
        case JointId.HipRight:
        case JointId.KneeRight:
        case JointId.AnkleRight:
            //return Quaternion.Euler(270, 0, 90);
            return Quaternion.Euler(0, 0, 0);
        case JointId.ClavicleLeft:
        case JointId.ShoulderLeft:
        case JointId.ElbowLeft:
        case JointId.ThumbLeft:
            //return Quaternion.Euler(180, 0, 90);
            return Quaternion.Euler(90, 0, 0);
        case JointId.ClavicleRight:
        case JointId.ShoulderRight:
        case JointId.ElbowRight:
        case JointId.ThumbRight:
            //return Quaternion.Euler(0, 0, 270);
            return Quaternion.Euler(90, 0, 180);
        case JointId.FootLeft:
            return Quaternion.Euler(180, 90, 0);
        case JointId.FootRight:
            return Quaternion.Euler(180, 270, 180);
        case JointId.WristLeft:
        case JointId.HandLeft:
        case JointId.HandTipLeft:
            //return Quaternion.Euler(0, 0, 90);
            return Quaternion.Euler(270, 0, 0);
        case JointId.WristRight:
        case JointId.HandRight:
        case JointId.HandTipRight:
            //return Quaternion.Euler(270, 0, 270);
            return Quaternion.Euler(0, 0, 180);
    }
```

```
      return Quaternion.Euler(90, 0, -90);
    }

    private readonly Dictionary<HumanBodyBones, JointId> boneJointMap = new Dictionary<HumanBodyBones,
JointId>() {
      // 顔
      {HumanBodyBones.LeftEye, JointId.EyeLeft},
      {HumanBodyBones.RightEye, JointId.EyeRight},

      // 上半身
      {HumanBodyBones.Hips, JointId.Pelvis},
      {HumanBodyBones.Head, JointId.Head},
      {HumanBodyBones.Neck, JointId.Neck},
      {HumanBodyBones.Chest, JointId.SpineChest},
      {HumanBodyBones.Spine, JointId.SpineNavel},

      // 左腕
      {HumanBodyBones.LeftShoulder, JointId.ClavicleLeft},
      {HumanBodyBones.LeftUpperArm, JointId.ShoulderLeft},
      {HumanBodyBones.LeftLowerArm, JointId.ElbowLeft},
      {HumanBodyBones.LeftHand, JointId.WristLeft},

      // 右腕
      {HumanBodyBones.RightShoulder, JointId.ClavicleRight},
      {HumanBodyBones.RightUpperArm, JointId.ShoulderRight},
      {HumanBodyBones.RightLowerArm, JointId.ElbowRight},
      {HumanBodyBones.RightHand, JointId.WristRight},

      // 左脚
      {HumanBodyBones.LeftUpperLeg, JointId.HipLeft},
      {HumanBodyBones.LeftLowerLeg, JointId.KneeLeft},
      {HumanBodyBones.LeftFoot, JointId.AnkleLeft},

      // 右脚
      {HumanBodyBones.RightUpperLeg, JointId.HipRight},
      {HumanBodyBones.RightLowerLeg, JointId.KneeRight},
      {HumanBodyBones.RightFoot, JointId.AnkleRight},

      // 両足
      {HumanBodyBones.LeftToes, JointId.FootLeft},
      {HumanBodyBones.RightToes, JointId.FootRight},

      // 左手
      {HumanBodyBones.LeftIndexProximal, JointId.HandTipLeft},
      {HumanBodyBones.LeftIndexIntermediate, JointId.HandTipLeft},
      {HumanBodyBones.LeftIndexDistal, JointId.HandTipLeft},
      {HumanBodyBones.LeftMiddleProximal, JointId.HandTipLeft},
      {HumanBodyBones.LeftMiddleIntermediate, JointId.HandTipLeft},
      {HumanBodyBones.LeftMiddleDistal, JointId.HandTipLeft},
      {HumanBodyBones.LeftRingProximal, JointId.HandTipLeft},
      {HumanBodyBones.LeftRingIntermediate, JointId.HandTipLeft},
      {HumanBodyBones.LeftRingDistal, JointId.HandTipLeft},
      {HumanBodyBones.LeftLittleProximal, JointId.HandTipLeft},
      {HumanBodyBones.LeftLittleIntermediate, JointId.HandTipLeft},
      {HumanBodyBones.LeftLittleDistal, JointId.HandTipLeft},

      // 右手
      {HumanBodyBones.RightIndexProximal, JointId.HandTipRight},
      {HumanBodyBones.RightIndexIntermediate, JointId.HandTipRight},
      {HumanBodyBones.RightIndexDistal, JointId.HandTipRight},
      {HumanBodyBones.RightMiddleProximal, JointId.HandTipRight},
      {HumanBodyBones.RightMiddleIntermediate, JointId.HandTipRight},
      {HumanBodyBones.RightMiddleDistal, JointId.HandTipRight},
      {HumanBodyBones.RightRingProximal, JointId.HandTipRight},
      {HumanBodyBones.RightRingIntermediate, JointId.HandTipRight},
      {HumanBodyBones.RightRingDistal, JointId.HandTipRight},
      {HumanBodyBones.RightLittleProximal, JointId.HandTipRight},
      {HumanBodyBones.RightLittleIntermediate, JointId.HandTipRight},
      {HumanBodyBones.RightLittleDistal, JointId.HandTipRight},

      // 左指
      {HumanBodyBones.LeftThumbProximal, JointId.HandLeft},
      {HumanBodyBones.LeftThumbIntermediate, JointId.ThumbLeft},
      {HumanBodyBones.LeftThumbDistal, JointId.ThumbLeft},

      // 右指
      {HumanBodyBones.RightThumbProximal, JointId.HandRight},
      {HumanBodyBones.RightThumbIntermediate, JointId.ThumbRight},
      {HumanBodyBones.RightThumbDistal, JointId.ThumbRight},
    };
}
```

## 個々グループの平均達成度得点




グループ ABC コース 2 達成度得点




グループ ABC コース 3 達成度得点

# 参考文献


[1] 書籍：日本 JSBA スノーボード教程　p10 .p64

[2] BLADIN, Christopher; MCCRORY, Paul; POGORZELSKI, Anita. Snowboarding injuries: current trends and future directions. Sports　Medicine, 2004, 34: 133-138

[3] Hament, B., Cater, A., & Oh, P. Y. (2017, June). Coupling virtual reality and motion platforms for snowboard training. In 2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI) (pp. 556-560). IEEE

[4] Wu, E. et al. : How to VizSki: visualizing captured skier motion in a VR ski training simulator. In: The 17th International Conference on Virtual-Reality Continuum and Its Applications in Industry, VRCAI 2019.

[5] 渥美亮祐, & 金井秀明. 深層学習を用いたスキーシミュレータ利用時における初心者と熟練者の運動特徴抽出手法. 研究報告グループウェアとネットワークサービス (GN), 2020(15), 1-7 (2020).

[6] 小野重遥, 金井秀明, 西本一志, & 渥美亮祐. (2021). スキー初学者のための荷重移動の提示によるスキー技能学習支援:『内面化可能 AI』 の実現に向けた初期的試み. 研究報告ヒューマンコンピュータインタラクション (HCI), 2021(4), 1-8.

[7] 小野重遥, 金井秀明, & 小池英樹. (2022). アルペンスキーの遠隔教育システムの提案と初期的検討. 研究報告グループウェアとネットワークサービス (GN), 2022(24), 1-6.

[8] Hämäläinen, P., Marshall, J., Kajastila, R., Byrne, R., & Mueller, F. F. (2015, October). Utilizing gravity in movement-based games and play. In Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play (pp. 67-77).

[9] Lyons, E. J. (2015). Cultivating engagement and enjoyment in exergames using feedback, challenge, and rewards. Games for health journal, 4(1), 12-18.

[10] Fulham O'Neill, D., & McGlone, M. R. (1999). Injury risk in first-time snowboarders versus first-time skiers. The American journal of sports medicine, 27(1), 94-97.

[11] Bladin, C., McCrory, P., & Pogorzelski, A. (2004). Snowboarding injuries: current trends and future directions. Sports Medicine, 34, 133-138.

[12] Taborri, J., Keogh, J., Kos, A., Santuz, A., Umek, A., Urbanczyk, C., ... & Rossi, S. (2020). Sport biomechanics applications using inertial, force, and EMG sensors: A literature overview. Applied bionics and biomechanics, 2020.

[13] Kuwahara, T., Takahashi, I., & Harikae, S. (2020, October). Real-Time Snowboard Training System for a Novice Using Visual and Auditory Feedback. In 2020 IEEE

International Conference on Systems, Man, and Cybernetics (SMC) (pp. 4230-4235). IEEE.

[14] SPELMEZAN, Daniel; BORCHERS, Jan. Real-time snowboard training system. In: CHI'08 Extended Abstracts on Human Factors in Computing Systems. 2008. p. 3327-3332.

[15] CHAN, Jacky CP, et al. A virtual reality dance training system using motion capture technology. IEEE transactions on learning technologies, 2010, 4.2: 187-195.

[16] Modgil, S., Singh, R. K., & Hannibal, C. (2022). Artificial intelligence for supply chain resilience: learning from Covid-19. The International Journal of Logistics Management, 33(4), 1246-1268.

[17] NEMOTO, Tomoko; BEGLAR, David. Likert-scale questionnaires. In: JALT 2013 conference proceedings. 2014. p. 1-8.

[18] Taber, K. S. (2018). The use of Cronbach's alpha when developing and reporting research instruments in science education. Research in science education, 48, 1273-1296.