第 29 回日本バーチャルリアリティ学会大会論文集（2024 年 9 月）

ソーシャル VR プラットフォームにおいてデバイスの入力変調を実現するドライバソフトウェアの提案

西本和貴 [1,2]，倉井 龍太郎 [3]，鳴海拓志 [1]，平木剛史 [2]
Kazuki NISHIMOTO, Ryutaro KURAI, Takuji NARUMI and Takefumi HIRAKI

1) 東京大学（〒 113-0033 東京都文京区本郷 7-3-1, {nishimoto, narumi}@cyber.t.u-tokyo.ac.jp）
2) クラスター メタバース研究所（〒 141-0031 東京都品川区西五反田 8-9-5, t.hiraki@cluster.mu）
3) クラスター株式会社（〒 141-0031 東京都品川区西五反田 8-9-5, r.kurai@cluster.mu）

概要: VR 体験でユーザの操作量をそのまま反映しないケースが増えている。ソーシャル VR プラットフォームで操作量を編集するには、クライアントソフトを変更せずに操作量を取得し変調する必要がある。本稿では、OpenVR 対応機器向けに、SteamVR 上のデバイスポーズを変調するドライバソフトと外部アクセス用インタフェースライブラリを開発した。SteamVR Home や cluster 上でデバイス座標の固定や操作量の C/D 比変調をリアルタイムに実行できることを確認した。

キーワード： ユーザインタフェース, ソーシャル VR, ハードウェアデバイス

1. はじめに

アバターをつかった体験は現実の体を使うことでは不可能な体験や効果を生み出すことが可能であり、現実での操作量をそのまま反映しない体験が多数提案されてきている。例えば、VR 環境の広い空間を移動しているように感じさせる場合、実空間を実際に歩こうとすると移動可能な範囲には物理的な空間の制約が生まれる。この問題を解決するために、ユーザに気づかれないように VR 環境を回転させることで実際の歩行方向を変化させ広大な空間を歩いているように感じさせる Redirected Walking(RDW) と呼ばれる手法が提案されている [1]。ほかにも実際のコントローラの操作と視覚的に表示するコントローラの移動量をずらすことで、錯覚的な触覚感覚を作り出す、Pseudo-Haptics と呼ばれる技術が提案されている。ここでインタフェースから入力された操作量（Control）と実際の画面に表示される操作量（Display）の比率のことを Human-Computer-Interaction の分野では Control Display Ratio（C/D 比）と呼ぶ。Samad らは手に持った箱を上下させるタスクにおいて、この C/D 比を変化させることで重量知覚が変化することを示し、その変化量について g 単位で評価を行った [2]。このように VR 環境において、デバイスから入力される操作量を変調して表示することで効果的な VR 体験を作り出すことができる。

一方でこれらの特殊な操作量の変調は効果的であるにもかかわらず、VRChat や cluster などのソーシャル VR プラットフォームにおいて、有効活用されている例は少ない。この原因として、プラットフォーム側は操作量変調という特殊な技術の実装を行なっていないことや、ユーザが独自に実装を行おうとしてもプラットフォームが提供するクライアントソフトウェア上で操作している動作を直接編集するのは難しいといったことが挙げられる。例えば実験室実験やゲーム内の演出の開発ではアプリケーションごとに自分で 1 から実装を行うことが可能であるが、ソーシャル VR プラットフォームのようなプラットフォーマーが提供するクライアントソフトウェア上で特殊な操作量変調を行おうとするとクライアントソフトウェアに手を加える必要があり、これは通常困難である。

図 1: cluster 上における操作変調の例
左が変調なし、右が C/D 比の変調をしたもの。現実ではほぼ同じ移動をしているにもかかわらずオブジェクトの移動量が異なる

そこでユーザ（クリエイタ）側がクライアントソフトウェアそのものに手を付けずに、操作量をリアルタイムに編集できるソフトウェアを実現することができれば、これまでに分かっている効果的な操作量変調技術を使った体験づくりを気軽に行うことができるようになったり、研究者が自身が開発した新しい操作量変調技術を既存のプラットフォーム上でテストしたりするといったことが可能になる。これまでにクライアントソフトウェアに手を付けずに、VR 環境においてデバイスのキャリブレーションなどを行うための

ドライバなどが公開されている[1]。しかし、これらは元のデバイス座標に回転・並進移動などのオフセットを加えて動作を補正することを目的としており、操作量をリアルタイムに変調したり直接デバイスのポーズを上書きするといった例には対応していない。

そこで本研究では操作量を編集する技術を民主化することを目指し、ソーシャル VR プラットフォームなどに接続されたデバイスの入力を、起動しておくだけで外部から変調することができるドライバソフトウェアを提案する。これにより、商用プラットフォームにおいて、開発者が最低限の操作のみでクライアントソフトウェアに対して非破壊的に VR 環境内での操作量を変調することが可能になる。

本稿では VR 環境においてデバイス座標系を補正するプログラム[1]を参考にして、独自の動作変調機能を持つドライバソフトウェアの設計と実装を行い、SteamVR 上やソーシャル VR プラットフォームである cluster 上で操作量変調の動作テストおよび実行速度の測定を行った。具体的には接続デバイスの座標を上書きする機能を実装し、操作量変調の例としてコントローラの座標の停止と入力デバイスの移動量に関する C/D 比の変更を可能にした。図 1 では本提案手法により cluster 上で操作量変調を行い C/D 比を変化させた時の様子を表している。

2. 提案手法

本稿では、現在 VR 機器の多くが対応する OpenVR および SteamVR を使ったアプリケーション全般に対して操作量の変調が可能なドライバソフトウェアを開発した。

初めに図 2 を用いて、簡単に提案手法の概略について説明する。通常、SteamVR は OpenVR API によってデバイスからデバイスポーズを取得し、クライアントアプリケーションへ渡す（図 2 上部）。本提案手法では、OpenVR API から SteamVR へデバイスポーズが渡される経路をフックし、本来のデバイスポーズに任意の変調を行った上で、SteamVR へあたかもデバイスから直接デバイスポーズが渡されているかのような情報経路を構築する（図 2 下部）。この情報経路を作成するために OpenVR API を使って元のデバイスポーズの取得および SteamVR へデバイスポーズを送信する「OpenVR ドライバ」と、OpenVR ドライバと通信を行うことで、オリジナルのデバイスポーズの取得や変調リクエストを行い、外部プログラムからライブラリとしてアクセスできる「OpenVR ドライバインタフェース」を作成した。

2.1 用語解説

次に本稿で使用する用語について説明を行う。

- **デバイスポーズ**

  OpenVR API によって定められたデバイスのポーズ情報。世界座標系からデバイスに対する平行移動量およびクォータニオン、HMD からデバイスに対する平行移動量およびクォータニオン、デバイス座標系での座標およびクォータニオン、加速度、角加速度などから構成される。

- **入力デバイス**

  今回は VR システムとして、SteamVR(OpenVR) 対応 PC 向け VR デバイスを対象とし、OpenVR API によってデバイスポーズを取得できるデバイスのことを指す。

- **SteamVR**

  Valve 社が運営する VR プラットフォームであり、VR 機器とソフトウェアを統合するシステムを提供する。

- **OpenVR API**

  SteamVR を提供している Valve 社が公式に提供している VR アプリケーションの開発キット（SDK）。特定のハードウェアベンダーの SDK に依存せずヘッドセットやデバイスの位置情報などにアクセスが可能であり、SteamVR のコアテクノロジーとして OpenVR API を使用して SteamVR は VR 機器と VR アプリケーションの通信を管理している。

- **クライアントアプリケーション**

  VRChat や cluster のようなソーシャル VRSNS、VR ゲームなど、VR アプリケーションの開発者が提供している SteamVR に対応したアプリケーションのことを指す。

- **OpenVR ドライバ**

  本稿で提案するドライバソフトウェアを示す。SteamVR 向けの外部ドライバとして機能し、SteamVR が OpenVR API を用いて OpenVR 対応機器からデバイスポーズを取得する経路をフックする。フック時にオリジナルのデバイスポーズを取得したり、必要に応じてオリジナルのデバイスポーズを変調して SteamVR へ情報を渡す機能をもつ。

- **OpenVR ドライバインタフェース**

  本稿で提案する OpenVR ドライバと通信を行うインタフェースライブラリを示す。OpenVR ドライバが取得したデバイスポーズを受け取る関数や、オリジナルのデバイスポーズを変調するリクエストを OpenVR ドライバへ送信する関数などを提供する。

2.2 設計・実装

2.2.1 OpenVR ドライバによる通信のフック

これまでに実装されてきた RDW や Pseudo-Haptics のような操作量変調システムでは、主に Unity 上などクライアントアプリケーションとして作用するレイヤーでデバイスからのポーズデータの取得と描画位置の変更を行ってきた。しかし、今回は内部情報へのアクセスが難しいソーシャル VR プラットフォームが提供するアプリケーションに対して操作量の変調を加える必要がある。そこで、多くの VR アプリケーションで使用されている SteamVR へ渡される

[1] pushrax et al.: OpenVR-SpaceCalibrator, https://github.com/pushrax/OpenVR-SpaceCalibrator.

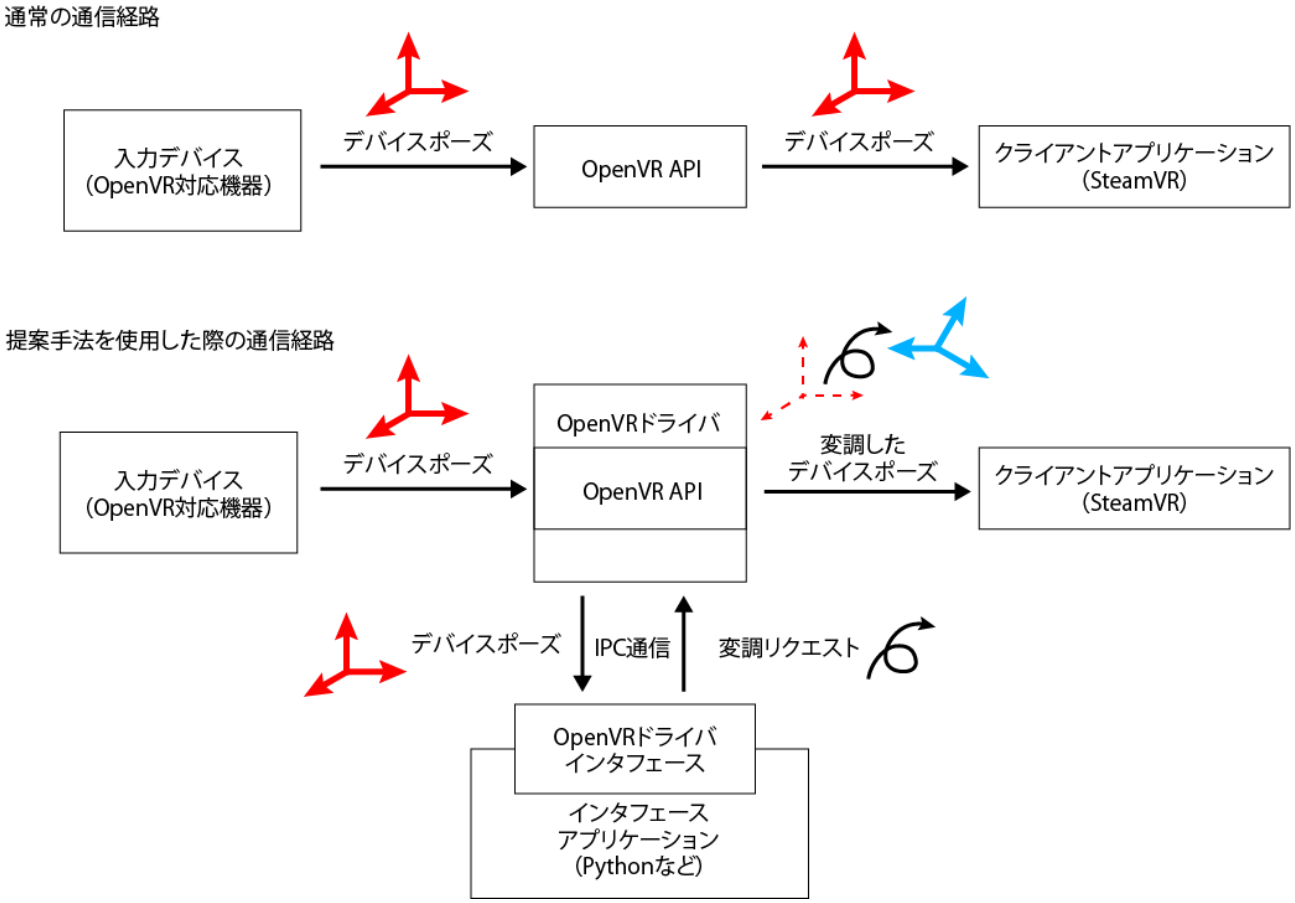

図 2: 提案手法：デバイスの入力変調を行う際の通信の模式図
赤い座標系が本来のデバイスポーズを表し、青い座標系が OpenVR ドライバインタフェースの変調リクエストを受けて本来のデバイスポーズを変調した後のデバイスポーズを表す

情報を事前に変調することで、内部を改変することが難しいアプリケーションにおいても操作量の変調を実現する。

SteamVR は通常、OpenVR API によって提供される特定の関数を呼び出すことで接続された VR デバイスのポーズ情報を取得している。提案手法である OpenVR ドライバは OpenVR API を使って作られた SteamVR の外部ドライバとして動作する。SteamVR がポーズ情報を受け取るために関数を呼び出す処理をフックし、SteamVR に接続された VR デバイスからポーズ情報の取得や必要に応じて変調を行う。変調をどのように行うのかについては OpenVR ドライバインタフェース（後述）と IPC 通信を行い、ホスト側として変調リクエストを受けることで決定する（図 2）。

2.2.2 OpenVR ドライバインタフェース

OpenVR ドライバインタフェースは OpenVR ドライバと IPC 通信を行う際にクライアント側として機能し、オリジナルデバイスのポーズ情報を取得したり変調リクエストを出したりすることが可能な関数を持つ動的ライブラリとして提供される。この OpenVR ドライバインタフェースの関数を Python や Unity のようなインタフェースアプリケーションから呼び出してポーズ変調を行うことを想定している。参考にしたプログラム[1]ではオリジナルのポーズ情報に対して、デバイスキャリブレーションによって得た補正用の並進・回転操作を加えるといった実装が主に行われていた。本提案手法では参考にしたプログラム[1]で実装されていたオリジナルのポーズ情報を指定した分だけ回転・並進方向にオフセットするリクエスト機能など一部を残し、現在までに、オリジナルのポーズ情報を取得するリクエスト、オリジナルのポーズ情報を指定したポーズ情報で上書きするリクエストなどの機能が実装されている。

3. VR 環境における変調例と実行速度評価

SteamVR Home および SteamVR を使って動作しているソーシャル VR プラットフォームである cluster 上で提案システムを使って実際の動作変調を行った。その動作変調の結果や実行速度について測定を行った結果を示す。

3.1 実験環境

今回開発した OpenVR ドライバを導入した SteamVR において、OpenVR ドライバインタフェースを Python から呼び出すことで動作の変調テストをおこなった。OpenVR ドライバの導入の仕方は公式の案内の通り、指定のディレクトリに dll 形式のドライバをおき、SteamVR の設定から有効にすることで行うことができる。SteamVR 対応 Widnows11 に Meta Quest2 を Quest Link で接続し、SteamVR Home の画面にて動作変調の様子をテストした。

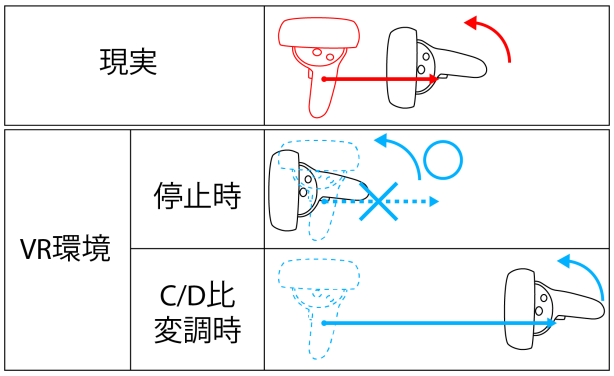

図 3: 現実での操作量と移動量に対して
停止・C/D 比変調を行なった際の動作イメージ
破線：移動前のデバイス位置 実線：移動後のデバイスの位置

3.2 動作の停止

現在操作しているデバイスの座標を VR 空間上に固定することを目的として、前フレームのポーズ情報で現在のポーズ情報を上書きすることで動作の停止ができるかテストをおこなった。その結果、OpenVR API で定義されるポーズ情報のうちデバイスの座標、速度、加速度を上書きすることで座標をその場に固定することに成功した。図 3 では現実での動作（オリジナルのデバイスポーズの変化）と、SteamVR アプリケーション上でテストを行った際に得られた VR 環境における動作の様子を表している。今回は変調の例として動作の停止や C/D 比の変調を行なった。停止の場合は、現実でデバイスを赤い矢印のように移動させても、位置は移動しない。また、この時座標のみ固定し速度、加速度を上書きしない場合は固定された座標上で振り動作のみ描画されるという結果になった。このことからポーズ情報を選択的に更新することが可能であることを確認した。一方で回転情報を上書きしたところ直前のデバイス姿勢と異なる姿勢で座標が固定されてしまった。これは更新が適切でないことが考えられる。回転情報は上書きせず、座標位置のみを上書きすると現実の回転情報を反映しつつも座標を固定することが可能になり、場所は固定されるがポインティングは自然にできるといった状況を作ることが可能になった。

3.3 C/D 比の変化

次に現実で動かしているデバイスの操作量と描画されるデバイスの座標を変化させることが可能か検証した。この機能は Python 上でデバイスの本来の座標変位を追跡し、その変位量に指定した C/D 比を乗算することで C/D 比変化後の座標を計算し、計算した座標を使ってオリジナルのポーズ情報を OpenVR ドライバインタフェースライブラリの関数を用いて上書きすることで実現した。この結果、C/D 比に応じて移動量の倍率を変更することに成功した。例えば C/D 比が 2.0 の時は本来の移動量の 2 倍座標が変位し、C/D 比が 0.5 の時は本来の移動量の半分だけ座標が変位しているように描画された。さらに C/D 比をマイナスに設定することで、本来の移動とは逆方向に座標が変位するといったことを確認した。図 3 において右下の C/D 比変調時と書かれている部分が実際の動作イメージである。現実では赤い矢印分の移動のみしかしていないが、VR 環境で表示されるデバイスの座標変化は青い矢印のように指定した倍率だけ大きくなった。また、この時も回転動作を固定しないことでオリジナルの回転姿勢を反映できる。

3.4 ドライバソフトウェアの実行速度

デバイスポーズの取得と動作の変調リクエスト送信を行うメインループにおいて、開始時刻と終了時刻の差から 1 ループにおける実行時間を計算しこれを 100,000 回実行して処理時間の平均値を求めた。その結果、単にデバイスポーズを取得するだけであれば実行時間は約 15 µs、動作の停止・C/D 比の変化のようなデバイスポーズを上書きする処理を入れた場合は実行時間が約 56 µs になることが分かった。これは仮に 240 Hz の描画システムにこの 56 µs の処理を追加しても、フレームタイムの増加は約 1.34%（4,170 µs + 56 µs）なので、パフォーマンスに大きな影響はないと考えられ、十分にリアルタイム性があるといえる。また、SteamVR Home と cluster 上の両方で実行速度の測定を行ったがほとんど変化はなかった。以上より、SteamVR に対応したアプリケーションであれば、本手法による操作量変調を問題なく適応できると考えられる。

4. むすび

本稿ではソーシャル VR プラットフォームのような内部の変更が困難なクライアントアプリケーションにおいて、入力デバイスの移動量などを変調するドライバソフトウェアの実装を行なった。その結果、SteamVR を利用しているアプリケーションにおいて、作成したインタフェースからの指令をもとに、デバイスポーズの要素を選択的に上書きし、座標停止や本来のインタフェースから入力された操作量と実際に描画される操作量の比率を変化させることに成功した。本提案手法により、SteamVR を利用した多くの VR アプリケーションで、接続されたデバイスの入力変調をリアルタイムに行うことが可能になったと考えられる。しかし、現状回転座標の更新が適切ではなかったり、OpenVR に対応した機器以外では使用できないといった課題を抱える。そこで今後はドライバの調整を進めるとともに、自作のデバイスなどであっても OpenVR のデバイスとして認識させるといった形で本入力ドライバの機能拡張を行なっていきたい。

謝辞 本研究の一部は、JST ムーンショット型研究開発事業 (JPMJMS2013) の支援を受けて実施された。

参考文献

[1] S. Razzaque et al.: Redirected walking, Proc. of EURO GRAPHICS, Vol.9, pp. 105-106, 2001.

[2] M. Samad et al.: Pseudo-Haptic Weight: Changing the Perceived Weight of Virtual Objects By Manipulating Control-Display Ratio, Proc. of the 2019 CHI Conference on Human Factors in Computing Systems, pp. 1-13, 2019.