修士論文

ヘッドマウントディスプレイ向けの直観的な文字入力インターフェースの提案

Intuitive Text Entry Interface for HMDs

平成29年02月03日提出

指導教員
坂井 修一 教授
入江 英嗣 准教授

東京大学大学院 情報理工学系研究科
電子情報学専攻

48-146429 土井 秀信

概要

近年，様々なウェアラブルデバイスの登場により，人とコンピュータの関わり方が変化してきている．例えば，ウェアラブルデバイス環境では，最も代表的な文字入力デバイスであるキーボードを使用することが難しい．そのため，文字入力には小型のコントローラを把持することが多い．しかし，手に何も持たずに文字入力を行うことができれば，ウェアラブルデバイスの利便性が高まる．本論文では，ウェアラブルデバイスの一つであるヘッドマウントディスプレイ (HMD) に着目し，HMD 向けに特化した直観的な文字入力インターフェースの提案および評価を行う．提案システムでは，深度カメラによって指先位置が認識され，空中でタップやフリックジェスチャをすることで文字入力ができる．ポインティングと入力ジェスチャの両方を一本の指で行うため，ジェスチャ動作時にカーソルが不必要に動いてしまう問題がある．本論文ではこれを解決するため，指の運動状態を 3 つのステートに分類することで，必要に応じてカーソルを一時的に固定するロックカーソル方式を提案する．QWERTY 配列とテンキー配列の 2 種類のインターフェースを実装し，従来方式であるコントローラを用いた文字入力と比較検討を行った．また，アンケート調査を実施し，ユーザビリティの面からも評価を行った．

目 次

図 目 次

表 目 次

第1章　はじめに

近年，集積回路の小型化・省電力化等といった技術革新により，コンピュータを「身に着ける」ことが可能になってきた．これらはウェアラブルデバイスと呼ばれ，人々がコンピュータ資源を利用する際の選択肢が大きく広がった．ウェアラブルデバイスに搭載されたセンサ類によってユーザの位置や動作，視界，体温といった情報が取得され，小型化されたインターフェースによりこれらの情報をその場で活用することが可能となった．

代表的なウェアラブルデバイスのひとつに，ヘッドマウントディスプレイ (HMD) がある．HMD の登場により，現実世界とバーチャルな世界の境界は曖昧になろうとしている．HMD は大きく 2 種類に分けることができる．一つは光学シースルー型で，もう一つは密閉型の HMD である．光学シースルー型 HMD では，実視野が確保されており，その上に情報が重畳表示される．このような特性から，拡張現実 (AR) 分野を得意としており [1]，医療分野にも利用されている [2]．また，IoT 機器との連携も期待されている [3]．一方，密閉型 HMD では，視野を覆うように画面が広がり，ユーザを外界から完全に遮断するような形になる．このような特性から，仮想現実 (VR) 分野を得意としている [4]

多くのデジタルデバイスにおいて，文字入力システムは非常に大きな役割を担っている．Web サイトの検索や文書作成などの際に，文字入力システムは不可欠である．しかし，HMD を装着した状態で一般的なキーボードを操作することは，特に密閉型 HMD において困難となっており，親和性が低いと考えられる．このような理由から，HMD 環境での文字入力の多くはコントローラを把持するものになっているが，片手あるいは両手がふさがってしまい，利用の仕方が制限されてしまう．また，これらはスマートフォン向けのソフトウェアキーボードを流用したものがほとんどであるので，HMD 向けに最適化された文字入力インターフェースの開発が必要である [5]．こうした文字入力システムの欠陥は，HMD の普及を妨げる一因となってしまっている．

HMD を装着した状態で，コントローラ等の外部入力デバイスを用いなくてもポインティングやクリック，そして文字入力が可能になれば，コントローラ類は不要となる．そうすれば，両手に何も持つ必要がなくなり，各種入力の際にコントローラを持ったり置いたり，あるいは身体に身に着けておくこともなくなる．これを実現するためには，人間の身体の中で比較的器用な動作が可能となっている手，特に指の動きをセンサ類で捉え，ジェスチャ認識を行うことが有効である考えられる．

本研究では，HMD 装着時のハンズフリーな文字入力を可能にするため，HMD 向けに特化した文字入力インターフェースを提案する．手に何も持たなくても画面が見えるという HMD の特性を最大限生かすため，マーカレスのジェスチャ認識を用いる．また，雑音の多い環境や音の出しにくい環境での使用を考慮し，マイク等は用いない．HMD 上に表示されたキーボードには，深度カメラによって捉えられた指先位置に応じて，カーソルが重畳表示される．QWERTY 配列とテンキー配列の 2 種類のキーボードを開発し，前者に対してはタップ，後者に対してはフリックジェスチャを入力ジェスチャとした．

以下，第 2 章では，本研究に関連する，HMD における表示形式やジェスチャ入力，そしてウェアラブルデバイスにおける文字入力手法を紹介する．第 3 章では，HMD 向け文字入力インターフェースの提案を行い，実装方法を示す．第 4 章では，提案手法の比較実験を行い，考察を述べる．

第2章　関連研究

2.1　HMDにおける表示形式やジェスチャ入力

多くのHMDでは，カーソルキーやジョイスティック，あるいはタッチパネルを搭載したコントローラが付随しており，これらを用いた操作が基本となっている．しかし，これらを用いないジェスチャ入力の試みも存在する．同時に，コントローラを用いないような操作に適した画面表示方法も考案されている．この項では，このような手法を紹介する．

2.1.1　円筒表示 [7]

Billinghurst[7] らは，複合現実感 (Mixed Reality; MR) 環境での情報空間を次の3つに分類している．

- **a.** Head-stabilized: 情報はユーザの視点に固定され，ユーザ自身が動いたり視線を移動させたりしても変化しない．
- **b.** Body-stabilized: 情報はユーザの体と相対的に固定され，視線移動とともに変化するが，ユーザ自身が動いても変化しない．
- **c.** World-stabilized: 情報は実世界の位置に固定され，ユーザの視線移動やユーザの移動に応じて変化する．

このうち，彼らの提案するBody-Stabilized方式では，図2.1に示すように，疑似的に円筒状のディスプレイを周囲に展開する．情報は体に対して相対的な位置に固定され，首の動きを感知して情報がスクロールする．この方式を用いることで，光学シースルー型HMDの弱点である視野角の狭さを克服することができる．

2.1.2　3D Finger CAPE[8]

コントローラを持たなくても，手の位置を推定することで，素手で操作を行うことが考えられる．Jang ら [8] は，深度カメラで指先や手の関節位置を推定し，これを元にバーチャルな手をHMD上に表示し，仮想オブジェクトに触れたり，クリックしたりする，3D Finger CAPE を提案している (図2.2)．指を曲げたり交差

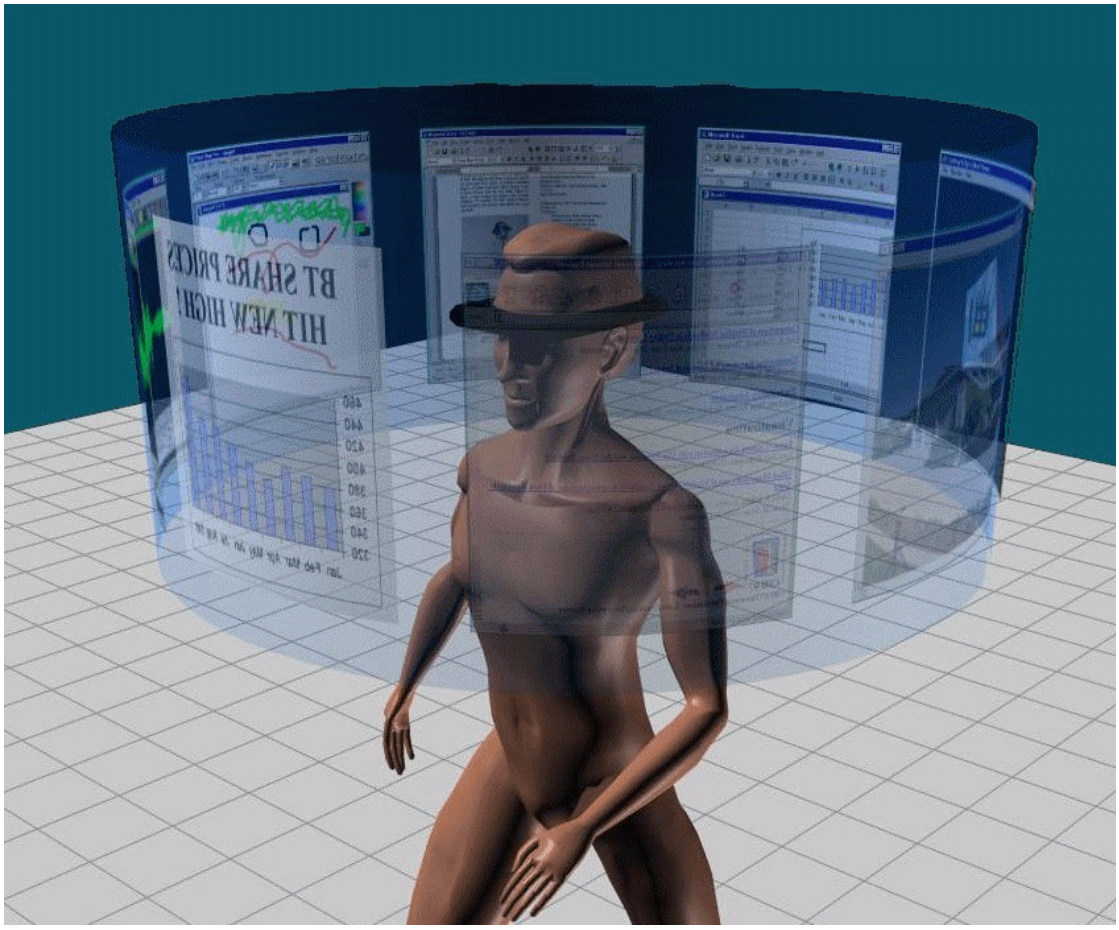

図 2.1: Body-Stabilized display [7]

させると，一人称視点の深度カメラには写らない手の部分が出てきてしまう．このような隠れてしまっている部分の関節位置も取得するために，深度カメラベースの指先認識と，3 次元の手姿勢推定を組み合わせて使用し，高いクリック認識率 (96.90%) となっている．

2.1.3 AirTarget[9]

入江ら [9] は，指で直観的なポインティングを行うことができる UI，AirTarget を提案している (図 2.3)．指先位置がカメラによってマーカレスで認識され，周囲のオブジェクトを矩形選択して検索クエリとすることができる．照明条件の変化に強い独自の肌色検出アルゴリズムによって肌色領域を抽出し，指先位置を検出する．また，指先位置にカーソルを正確に重畳表示するため，画面四隅を指さすキャリブレーションをはじめに行い，線形変換によって表示位置を調節する．18fps 以上のリアルタイム性を確保し，習熟の必要のない簡単なジェスチャで操作が可能

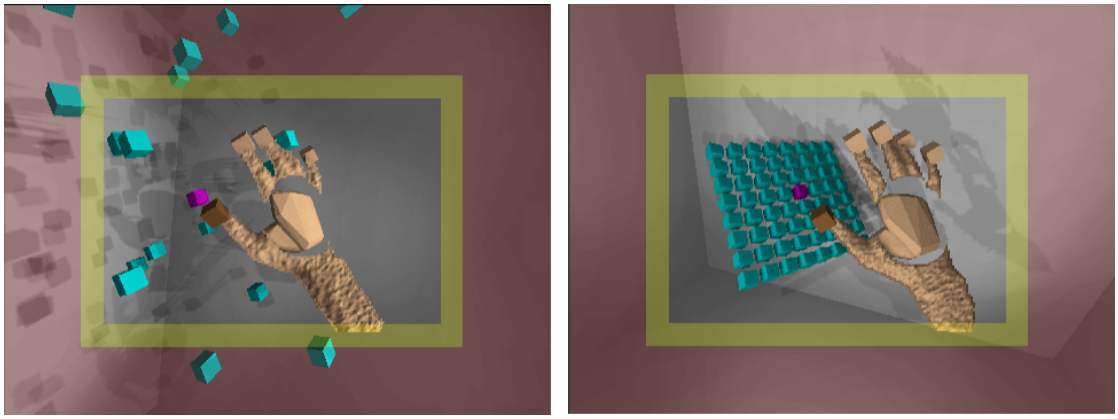

図 2.2: 3D Finger CAPE [8]

となっている．

2.1.4 Air tap[10]

Microsoft の HoloLens では，Air tap というジェスチャが主に用いられる [10]．Air tap は指を一本伸ばした状態から曲げて元に戻すというジェスチャで，クリックに相当する役割を担っている (図 2.4)．正面の深度センサによって指が伸びているか曲げられているかを監視し，それに応じてカーソルの形が変化する (中抜きの円 ↔ 点)．画面中央に固定されたカーソルを World-stabilized 表示のブラウザやキーボードの方へ首の動きで合わせて，Air tap をすることで操作を行う．また，tap 状態のまま手を動かすことで，ウィンドウをスクロールしたり好きな場所に移動させることができる．

2.2 ウェアラブルデバイスにおける文字入力

キーボードや十分な大きさのタッチパネルディスプレイをもたないウェアラブルデバイスにおいて，文字入力をするには様々な工夫が必要となる．この項では，各種ウェアラブルデバイス向けにデザインされた文字入力インターフェースを紹介する．

2.2.1 Swipeboard[11]

スマートウォッチに搭載されるタッチスクリーンは手首程度の大きさであり，そこに全てのキーを独立してセットするのは，指の太さを考慮すると現実的ではない．Chen らの提案する Swipeboard[11] は，図 2.5 のように，となりあう 3,4 キー

図 2.3: AirTarget [9]

を 1 つのキーにまとめ，二段階のスワイプ動作に分けることで，この問題を解決している．一段階目のスワイプでキー群を選択し，二段階目のスワイプでその中からさらに選択する．入力速度は 19.58WPM，エラー率が 13%で，高速な文字入力が可能となっている．この手法の特徴は，タッチ位置の情報を利用せずに，スワイプ方向のみを検出して文字入力を行うため，理論上かなり小型のタッチパネルでも入力操作が可能という点である．

2.2.2 DualKey[12]

上の項でも述べたように，スマートウォッチ上に全てのキーを独立して配置するのは難しい．Gupta ら [12] は，図 2.6 のように，人差し指と中指を区別することでキーを二重化する DualKey を提案している．右手人差し指に装着された光学センサの情報が Bluetooth によって左手のスマートウォッチに送信されることで，指の種類の情報を含んだタップを可能としている．また，人差し指のほうが中指よりも素早い入力が可能という検証結果から，人差し指が中指よりも多く使われるような SWEQTY 配列 (図 2.7) を考案した．SWEQTY 配列を使うことで入力速度

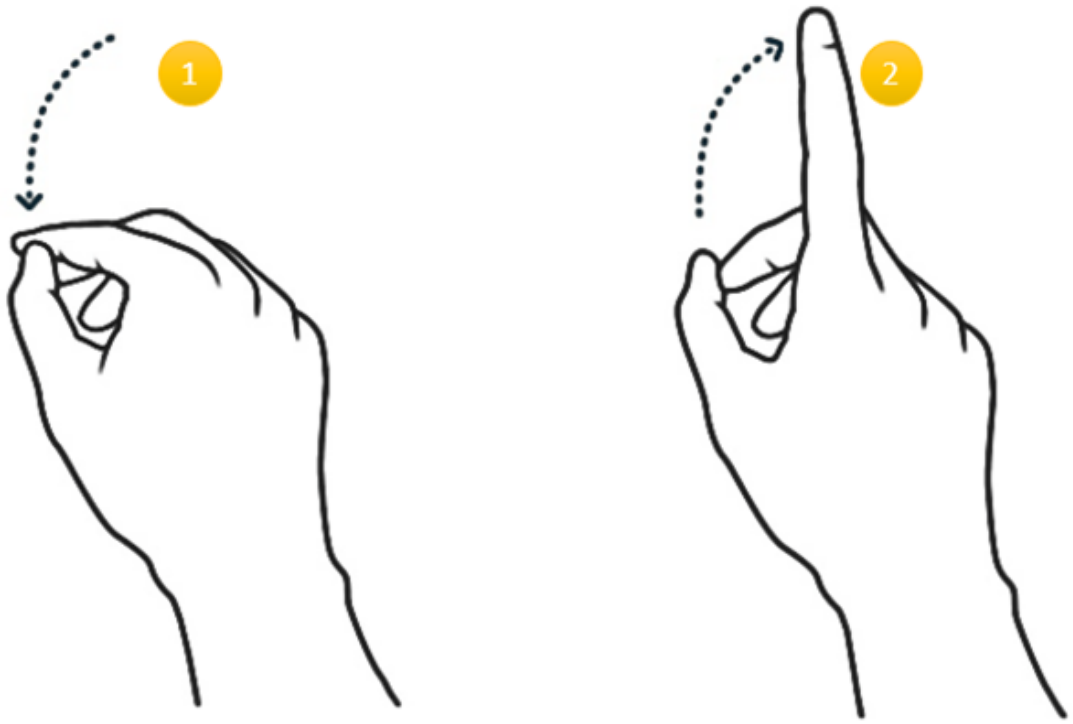

図 2.4: Air tap [10]

が向上し，上級者平均で入力速度が 21.59WPM，エラー率が 3.27%となっている．シングルタップ操作のみで 1 文字入力することができるため，非常に高速な手法となっているが，指先にセンサを装着することが必要なため，日常用途にはあまり向かないと考えられる．

2.2.3 DriftBoard[13]

スマートウォッチのような小さなタッチパネル上で文字入力を行う場合，タッチしたい場所が指で隠れてしまう Occlusion 問題がしばしば発生する．Shibata らの提案する DriftBoard[13] では，カーソルは一定の位置に固定されており，キーボードをスライド動作で移動させる（図 2.8）．スライド動作によって入力したいキーと固定カーソルが重なった時に指を離すと文字が入力される．この方式をとることで，指が入力したいキー上にくることがなくなり，Occlusion 問題は発生しなくなる．入力速度は 8.77WPM となっており，前述のものに比べると遅くなっている．エラー率は 1.13%となっている．.

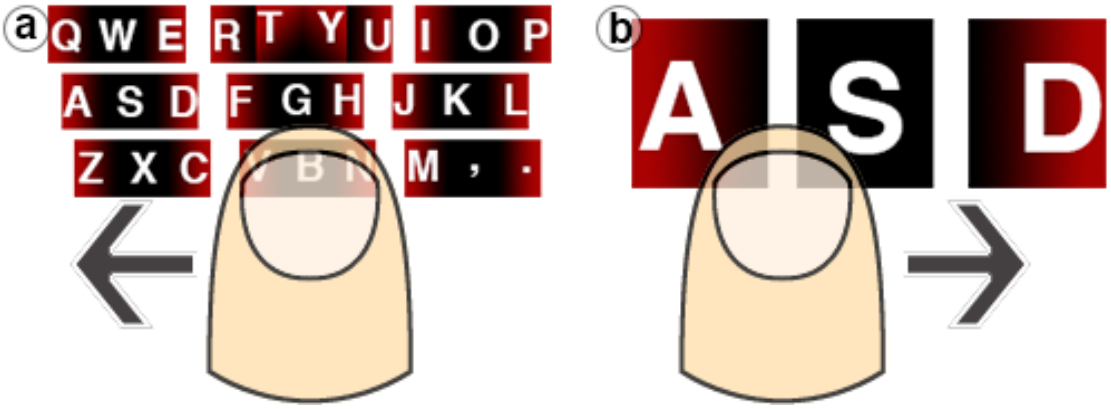

図 2.5: Swipeboard [11]

2.2.4 Kinect を用いた文字入力 [14]

人体の動きを感知するセンサのひとつに，Kinect がある．Shin ら [14] は，Kinect に映った全身画像から指先位置を追跡し，フリックジェスチャによる文字入力インターフェースを提案している．片手フリックによる文字入力がメインで，英字の大文字や日本語の小さい文字を入力する際には補助的に逆側の手を用いるのが特徴的である．しかし，Kinect の正面にいないと利用できないこと，大振りなジェスチャが必要なことから，本研究とは少し方向性が異なる．

2.2.5 EyeBoard[15]

ここまで紹介したものを含めて，多くの文字入力インターフェースは手や指が自由に動かせることが前提になっている．しかし，手が不自由な人でも使えるインターフェースがあれば，こういった人々の助けになる．Panwar ら [15] は，視線カメラを利用し,「見る」だけで文字入力ができる EyeBoard を提案している．図 2.10 のように，9 点でキャリブレーションを行うことで，視線の先にちょうどカーソルが表示され，入力したい画面上のキーを 1 秒ほど見続けると入力される．しかし，キャリブレーションの手間がかかること，視線カメラを追加する必要があること等といった問題がある．また，入力速度は 5.02WPM，エラー率は 1.88%となっている．

2.2.6 脳波による文字入力 [16]

体を動かさなくても操作できるインターフェースの分野に，Brain Computer Interface(BCI) というものがある．Koizumi ら [16] は，頭部に電極を装着し，Event-Related Potential(ERP) と呼ばれる脳の反応を計測することで文字入力を行うイ

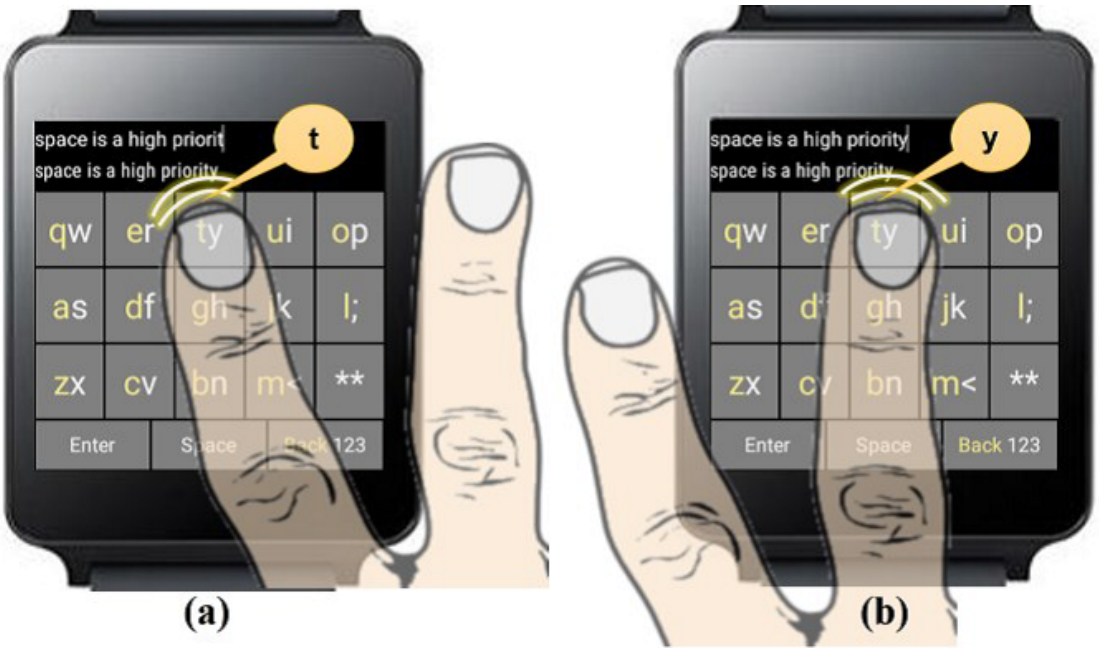

図 2.6: DualKey [12]

ンターフェースを提案している．文字配列の中から不規則にひとつの行または列が光っては消えるという画面をユーザは眺め，脳の反応の大きい行と列から入力文字を決定する (図 2.11)．

図 2.7: DualKey(SWEQTY 配列) [12]

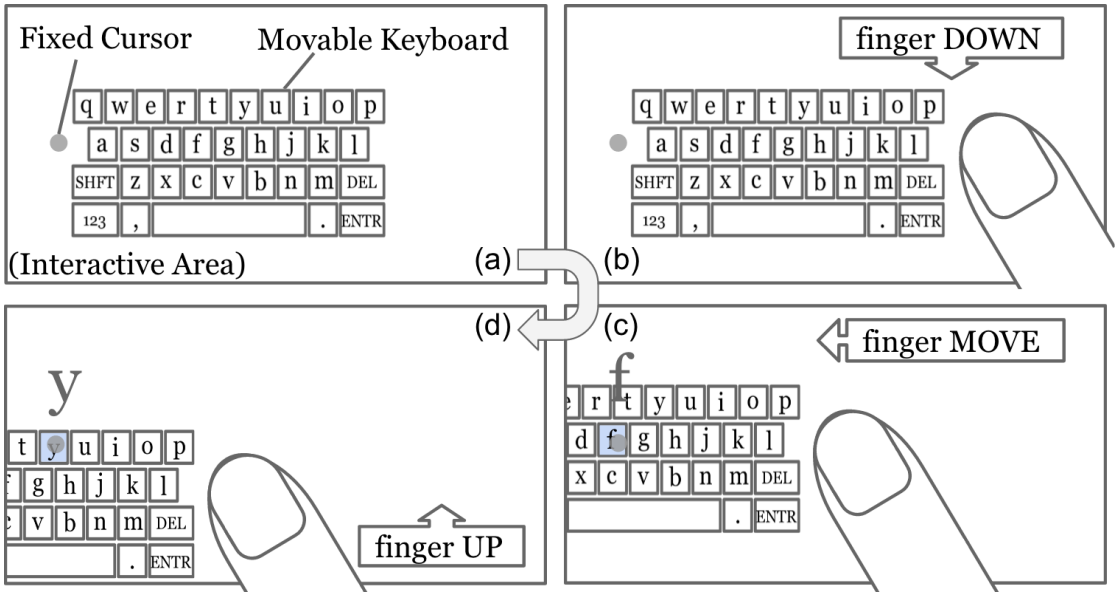

図 2.8: DriftBoard [13]

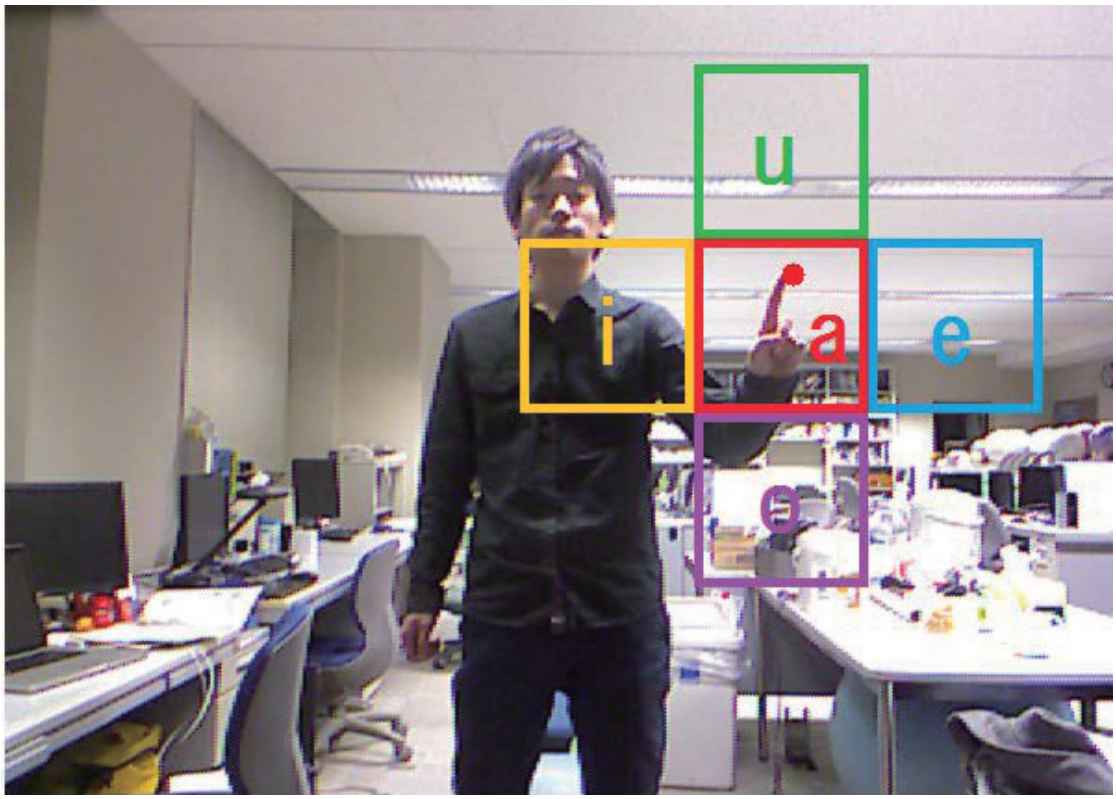

図 2.9: Kinect を用いた文字入力 [14]

図 2.10: EyeBoard [15]

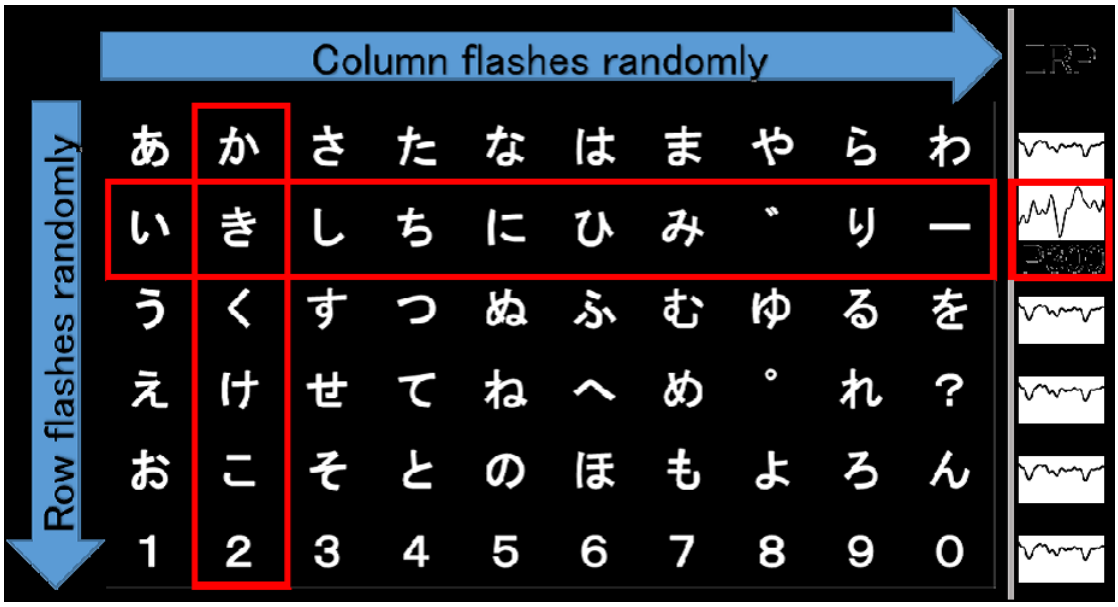

図 2.11: 脳波による文字入力 [16]

第3章　HMD向けの文字入力インターフェースの提案及び実装

3.1　HMD向けの文字入力インターフェースの提案

2章で紹介したように，HMDを装着した状態でジェスチャを利用する試みや，キーボードやマイクを使わない様々な環境で文字入力をする試みは数多く存在する．しかし，HMD向けに作られた文字入力システムの試みは非常に少ない．現行のHMDに標準で搭載されているキーボードは，スマートフォン向けのソフトウェアキーボードをHMD付属のコントローラで操作できるようにしたものになっており，HMD向けに最適化されているとは言えない．手にコントローラを把持する必要があり，他の手を使う作業と並列して操作を行うことが難しい．しかし，HMD環境における指認識やジェスチャ解析を用いて文字入力ができるようになれば，手に入力用デバイスを把持する必要はなくなり，様々な場面で文字入力を用いた支援が得られる．本研究では，このような文字入力システムを提案する．

提案システムでは，ユーザは手にコントローラを持つことなく，指先を使った直観的なジェスチャによって文字入力が可能となる (図3.1)．深度カメラによって指先位置が認識され，タップとフリックの2つのジェスチャによって入力を行う．提案システムを使用することで，日常生活や娯楽，さらには工場作業時など，HMDを装着するあらゆる場面で文字入力ができ，両手を拘束されることがない．こうした文字入力システムによって，HMDの弱点ともいえる文字入力のしづらさを解消する．提案システムを実現するために課題になるのが，入力ジェスチャ時にカーソル位置が移動してしまう問題である．ユーザの感じる座標系と深度画像の座標系が異なっているため，ユーザはd(深度)方向に指を移動したつもりでも，x成分やy成分が混ざってしまうことが多い．これが原因となって，意図した文字の隣の文字を入力してしまうことがある．あるいはジェスチャアルゴリズムを工夫し，カーソルがズレても入力文字はズレてしまわないようにすることが考えられるが，ユーザの意図に反するようなカーソルのぶれはユーザに違和感を与える要因となってしまう．

この問題を解決するため，本論文では指の運動状態をステート管理し，タップジェスチャの最中にはカーソルがキー状に一時的にロックされるロックカーソル

図 3.1: HMD 向けの文字入力インターフェース

方式を提案する．指先位置の移動度に応じて，MOVE，STOP，TAP の 3 つの状態を遷移し，カーソル位置の制御を行う．この 3 つの状態は，人が指を目的位置に移動させ，狙いを定めて静止し，入力するという一連の動作をモデル化したものである．

本研究では，QWERTY 配列とテンキー配列の 2 つのバージョンを実装し，前者はタップ，後者はフリックジェスチャを入力ジェスチャとした．以下，詳細な実装について述べていく．

3.2 指先認識

深度カメラで取得した深度画像 (図 3.2) をもとに，指先認識を大きく二段階に分けて行う．得られた深度画像からまず手領域の推定を行い，得られた輪郭を元に指先を推定する．

手領域の推定には，まず閾値による 2 値化を行う．今回は，36.4cm よりも遠い領域を背景とみなした．こうして 2 値化された画像をもとに，OpenCV の関数を用いて，手領域の候補となる輪郭がいくつか得られる．そして，得られた候補のうち，面積が最大の部分を手領域として推定する．また，このようにして得られた手領域を元に，指先座標の推定を行う．まず，手領域のうち，y 座標が最小の点

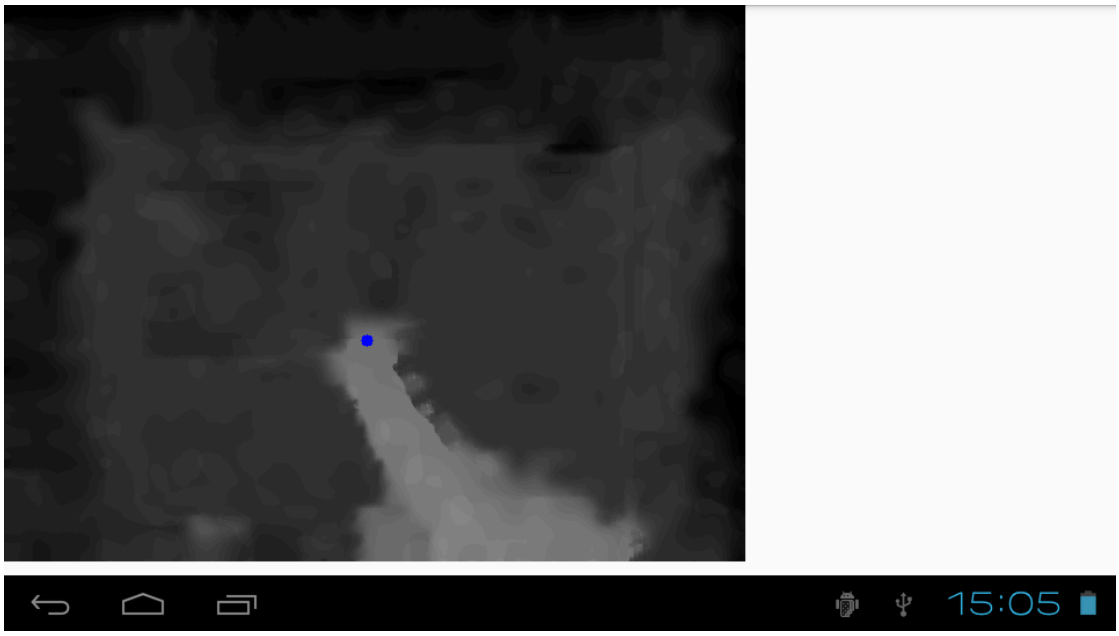

図 3.2: 指先の深度画像

T を取得する．しかしこのままだと，深度画像の特性上，実際の指先よりも少し上に外れた座標をとってしまい，実際の指先の深度を取得することができない．そこで，指の先端ではなく，少し手の甲に近い部分の座標を取得することを考える．先ほど取得した指の先端 T から，手の輪郭をもとに中心線 l を下に向かって求めていく．中心線 l を 50 点 (深度画像の縦幅の 10%ほど) 求めたのち，l の平均座標を指先座標 P と推定する (図 3.3)．また，中心線 l の平均深度を指先 P の深度とみなす．用いた指先認識アルゴリズムをアルゴリズム 1 に示す．

この方法による指認識の問題として，画面端の部分，特に右端と下端付近の領域を指示しようとした場合に，カメラに映る手の面積が著しく小さくなるため，そもそも手が検出されない，背景が指先と認識されてしまうなど，動作が著しく不安定になる．図 3.4 では，ユーザは画面右上を指示しているが，深度カメラの視野角限界によって人差し指とそれ以外の親指を含む領域が分離し，結果として面積の小さい人差し指領域が無視されてしまっている．そのため，なるべく画面端を指示しなくてもいいようにキー配置を工夫する必要がある．また，画面上部はお題となる単語や入力文字を表示するスペースとするため，指で指示する必要がない．そこで，実際に認識された指先位置 P を，y 軸正方向 (下方向) に定数シフトして利用する．

こうして実際に得られる指先位置は，ユーザは静止しているつもりでも様々な要因でゆらいでしまう．手や頭部の微小な動きであったり，深度画像生成アルゴリズムに起因するゆらぎといった要因が考えられる．しかし，このようなユーザの意図しないゆらぎは，様々な誤入力の原因になりうる．そのため，以下の式で

図 3.3: 指先認識

表される簡単な平滑フィルタによってこの不自然なゆらぎを緩和する．

$$fingertipSmoothed[t] = \frac{2 \times P_{shift} + fingertipSmoothed[t-1]}{3} \tag{3.1}$$

ここで，fingertipSmoothed[t] は最新の補正済み指先位置，P_{shift} は認識された指先位置 P を y 軸方向に定数シフトしたもの，fingertipSmoothed[t-1] は 1 フレーム前の補正済み指先位置を表す．抽出された指先座標と平滑化を行った後の座標を比較したものを図 3.5 に示す．図 3.5 では，3 フレーム目から押し込み動作が始まり，8 フレーム目から右方向へスライド動作を行っている．10 フレーム目にフリック判定がなされて s の文字が入力され，それを受けて 13 フレーム目以降はほぼ静止状態となっている．平滑化フィルタによって，静止状態の時のゆらぎを除去しつつ，押し込みやスライドといった意図的な動作に対してはほとんど遅延なく検知が可能となっていることが確認できる．

3.3 ジェスチャ解析

本研究では，主にタップ動作とフリック動作の 2 つのジェスチャを用いる．本項では，前項に記した指先認識によって (x, y, d) の 3 次元指先座標が既に取得されているものとする．また，バッファの id = t の場所に最新のカーソル位置が格納されており，毎フレームこれを更新しているものとする．原点は画面左上にあり，x, y は画素数スケールで，d は深度カメラからの距離 (単位:cm) となっている．

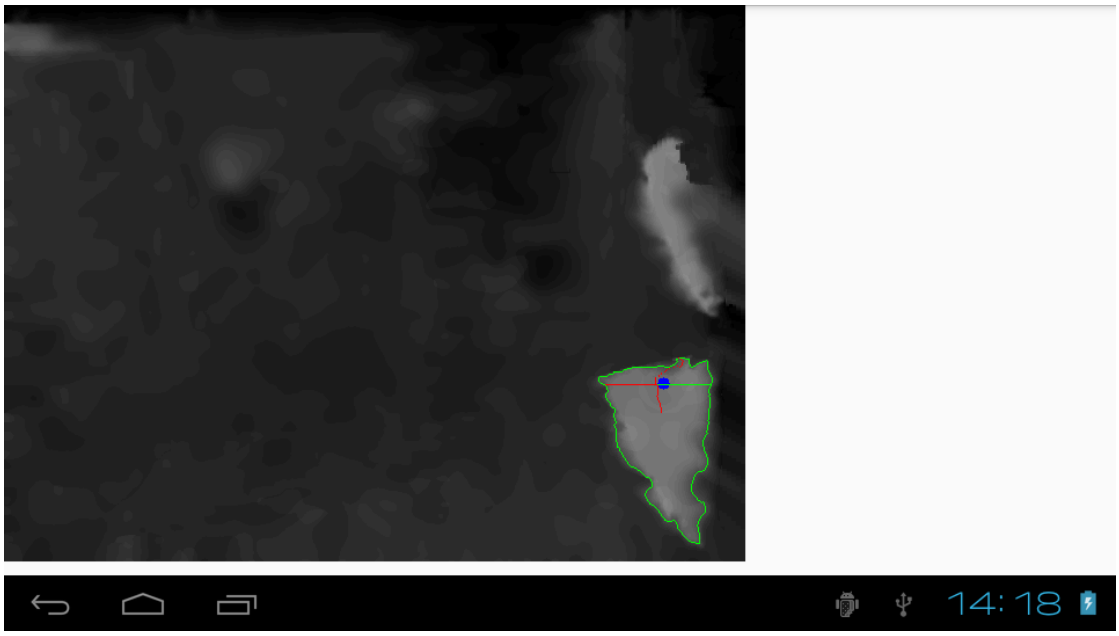

図 3.4: 指先認識（失敗例）

3.3.1 ステート管理

ユーザは前後に指を動かしているつもりでも，深度画像上の指先位置は縦や横にも少し移動してしまう．これは，ユーザと深度カメラとの間で座標系が異なっているなどといった理由が考えられる．タップ中は，ユーザにカーソルを移動させる意図はなく，それに反してカーソルが移動してしまうのは，ユーザに違和感を与える原因となってしまう．そこで本研究では，タップ動作やフリック動作中にはなるべくカーソルが移動してしまわないように，カーソルをキー上に一時的にロックすることでこの問題を解消する．カーソルをロックするため，指先位置の変化度をもとに，指の運動状態を監視することを考える．指のステートを以下に示す 3 つに分類し，状態遷移を行う．

- MOVE: 指先の移動度が大きく，カーソルは指先位置にそのまま追従する．
- STOP: 指先の移動度が小さく，指先位置にキーがあれば，そこにカーソルが移動する．
- TAP: 止まった状態から深度方向の移動があり，既にカーソルがキー上にあればカーソル位置がそのままロックされる．

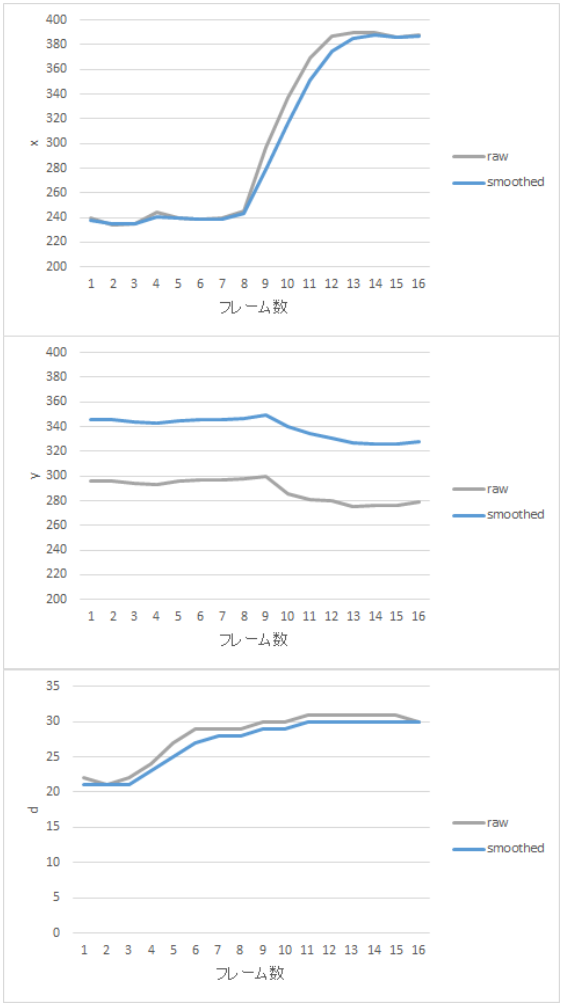

図 3.5: 平滑化前後の指先座標

状態遷移に用いる指先の移動度は，t フレーム目のとき，指先座標を格納しているバッファtipbuf[] を元に以下のように求める．

$$v_x = tipbuf[t].x - tipbuf[t-1].x \tag{3.2}$$

$$v_y = tipbuf[t].y - tipbuf[t-1].y \tag{3.3}$$

$$v_d = tipbuf[t].d - tipbuf[t-1].d \tag{3.4}$$

また，TAP 状態の時のみ，t-1 フレーム目の移動度 v'_x, v'_y, v'_d も用いる． こうして求めた移動度をもとに，図 3.6 のように状態を遷移させる．

後述するタップジェスチャやフリックジェスチャは，動作の性質上，一度のジェスチャで 2 フレーム以上の間陽性が継続することがあり，意図せず一度に同じ文字を複数回打ってしまうといった事態が考えられる．こうした問題を対策するため，文字の入力があった直後の数フレームに対して文字入力を制限することが有

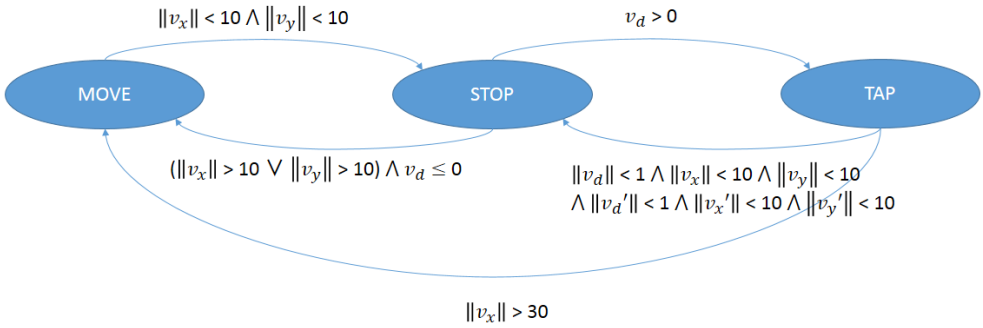

図 3.6: 状態遷移

効である．本研究では，文字の入力が行われた直後数フレームの間は必ず MOVE の状態になり，カーソルはキーにロックされず，文字入力は受け付けない．

3.3.2 タップ動作

指先の移動によってカーソルを所望の位置に移動したのち，タップジェスチャをすることで，文字を入力することを考える．タップジェスチャを行うとき，深度 d[cm] は小→大→小という変化をすると考えられる．そこで，深度 d が最大のフレームと最新のフレームを比較し，深度の変化量を条件に加えるのが有効である．また，タップジェスチャの前後で指先の位置は概ね近い位置に戻ってくると予想される．そこで，最新のカーソル座標に近いものをバッファから探索し，存在した場合にその座標をタップ位置と推定する．タップジェスチャ認識アルゴリズムをアルゴリズム (2) に示す．

タップ時の指先位置の遷移を図 3.7 に示す．x 軸や y 軸方向の動きはほとんどなく，d 値のみ目立った変化が現れている．d が最大となるのが 9 フレーム目で，12 フレーム目の時にタップ判定がなされ，同じ深度と近い xy 座標をもつ 6 フレーム目のカーソル座標がタップ位置として認識される．このように最新フレームではなく過去フレームの値をタップ座標として返すのは，タップの前後で位置が意図した場所とずれてしまうことが多く，タップ前の座標の方がタップ位置として信頼度が高いと考えられるからである．

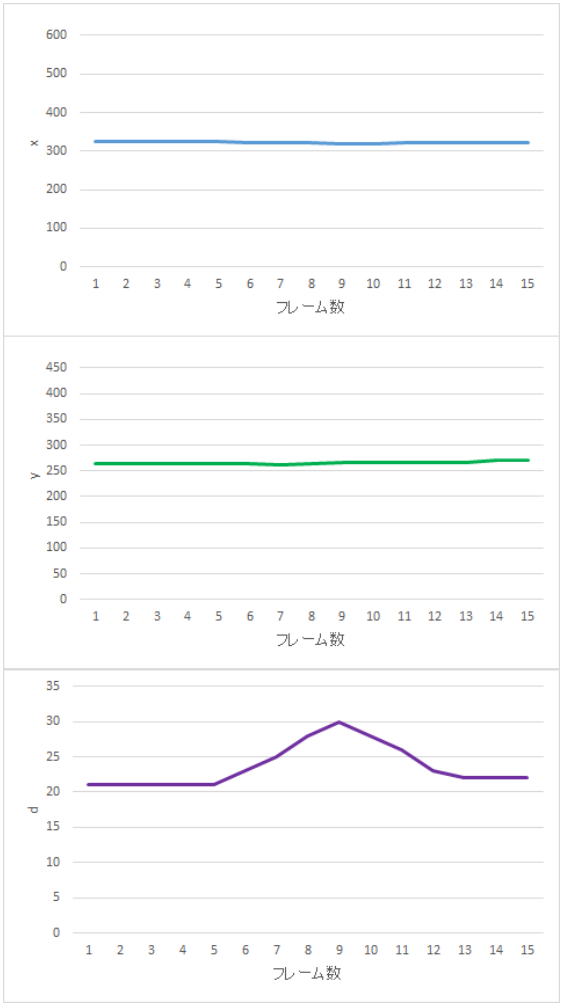

図 3.7: 指先位置の遷移 (タップ時)

3.3.3 フリック動作

指先の移動によってカーソルをキー上に移動させ，フリック動作をすることで，文字を入力することを考える．フリック動作は，押し込みと，スライド移動の 2 つのフェーズにわけることができる．まず，指先座標が 2 フレーム連続で d 方向に移動していた場合に押し込み動作を行ったと判定する．直近 8 フレームをチェックし，連続する 2 フレームの移動度 $v[t-i], v[t-i-1]$ が次の条件を満たすときに，押し込み状態と判定する．

$$0 < v_d[t-i] < 5 \quad \wedge \quad v_x[t-i] < 20 \quad \wedge \quad v_y[t-i] < 20 \tag{3.5}$$

$$0 < v_d[t-i-1] < 5 \quad \wedge \quad v_x[t-i-1] < 20 \quad \wedge \quad v_y[t-i-1] < 20 \tag{3.6}$$

また，キーが押し込まれた状態で，指先座標が 2 フレーム連続で上下左右のいずれかにスライドしたとき，フリックジェスチャを完了する．例えば，右向きの

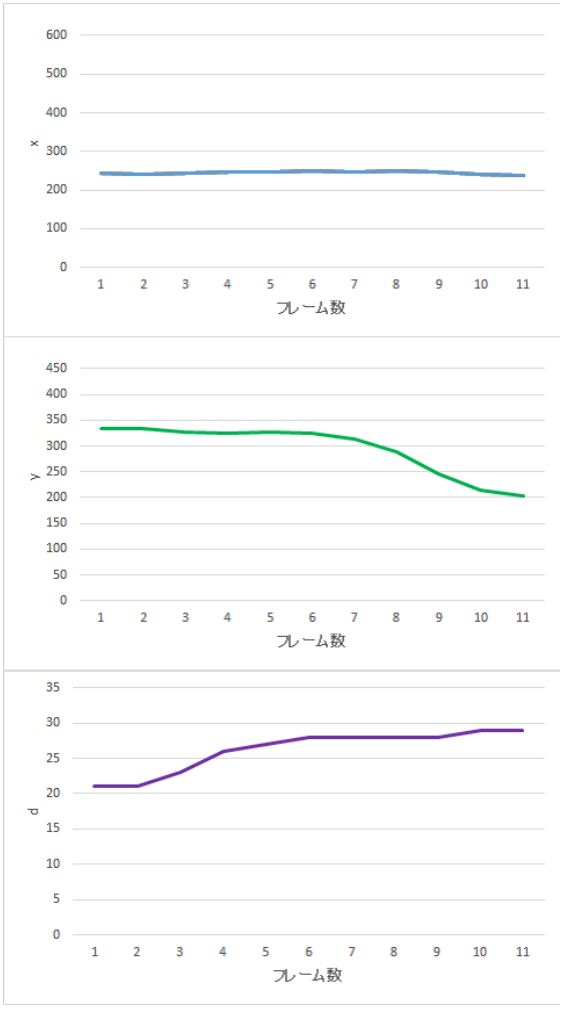

図 3.8: 指先位置の遷移 (上方向フリック時)

スライド動作は，最新のフレームの指先の移動度 $v[t]$ および一フレーム前の移動度 $v[t-1]$ を用いて次の式を満たすときとする．

$$10 < v_x[t] < 50 \quad \wedge \quad v_y[t] < v_x[t] \tag{3.7}$$

$$10 < v_x[t-1] < 50 \quad \wedge \quad v_y[t-1] < v_x[t-1] \tag{3.8}$$

移動度が 10 以下の時にはおそらく静止状態であり，また，50 以上の時には指の動きではない不連続な指先位置の変化である可能性が考えられるので，このような条件式となっている．

上方向フリック時の指先位置の遷移を図 3.8 に示す．2 フレーム目から押し込み動作が始まり，6 フレーム目から上向きのスライド動作が始まっていることがわかる．指の動きは 11 フレーム目まで上向きフリックの動作をしているが，実際には 8 フレーム目の時点で上向きフリックの判定が下され，r の文字が入力された．

図 3.9: QWERTY 表示

3.4 画面表示

画面上部にお題となる文字と入力文字を表示し，下部にユーザの操作するインターフェースを配置する．下部にに表示する文字配列には，QWERTY 配列とテンキー配列の 2 種類の配列を使用する．この二つは，多くの人にとって馴染みのある配列であり，目的の文字を探すことに時間がかからないことを重視した．

3.4.1 QWERTY 配列

QWERTY キーボードでは，指先の移動によってカーソル位置を操作し，タップジェスチャによって指定した位置の文字を入力する．QWERTY キーボードには，以下の利点が考えられる．

- 全ての文字が見える
- 多くの人にとって馴染み深く，文字位置の探索が容易
- 一段階のジェスチャによって入力が完了する

しかし，キーひとつが占める領域の面積が小さく，すぐ隣のキーを入力してしまうような誤入力が起きやすいといったデメリットが存在する．そこで，次項に示す文字予測機能によって，次に登場しやすい文字を通常よりも大きく表示することで，これを緩和する．予測されるキーの判定が大きくなることで，ユーザは指先によるポインティングがしやすくなる．

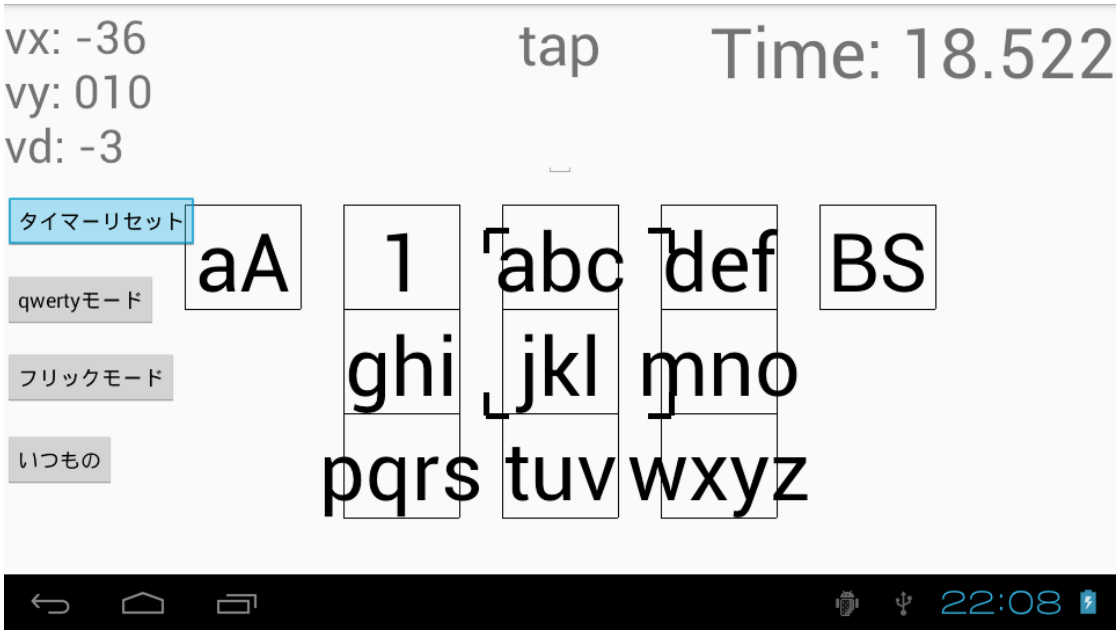

図 3.10: テンキー表示

3.4.2 テンキー配列

テンキー入力では，ひとつのキーに 3〜5 文字程度を圧縮して配置し，フリックジェスチャによって文字入力を行う．各キーの 1 文字目はタップ，2 文字目は左フリック，3 文字目は上フリック，4 文字目は右フリック，5 文字目は下フリックが割り当てられている．例えば”b”を入力する場合，abc と書いてあるキーの位置で左フリックを行うことで入力が完了する．テンキー入力には，以下の利点が考えられる．

- ひとつのキーが大きく，隣のキーを押してしまうミスが少ない
- かな入力にも同じ表示形式が使える
- スマホなどで見慣れている

テンキー表示の外観を図 3.10 に示す．テンキー表示に使用するフリックジェスチャは，押し込みとスライドの 2 段階に分かれるため，途中の段階である押し込み時にユーザに何らかのフィードバックがあるとユーザに優しい．そこで，フリック入力の途中の段階として，キーを押し込んでいるときは，小窓表示によりキーの 2 文字目以降が上下左右に展開され，図 3.11 のように表示される．押し込み後，指を元の位置に引くことで各キー 1 文字目 (a, d, g, ..) が入力される．それ以外の文字は，対応する方向へフリックすることで入力される．

また，文字の割り当てられていない方向へのフリックがあった場合，例えば abc キーの位置で右フリックまたは下フリックがあった時には，小窓が閉じられ，何

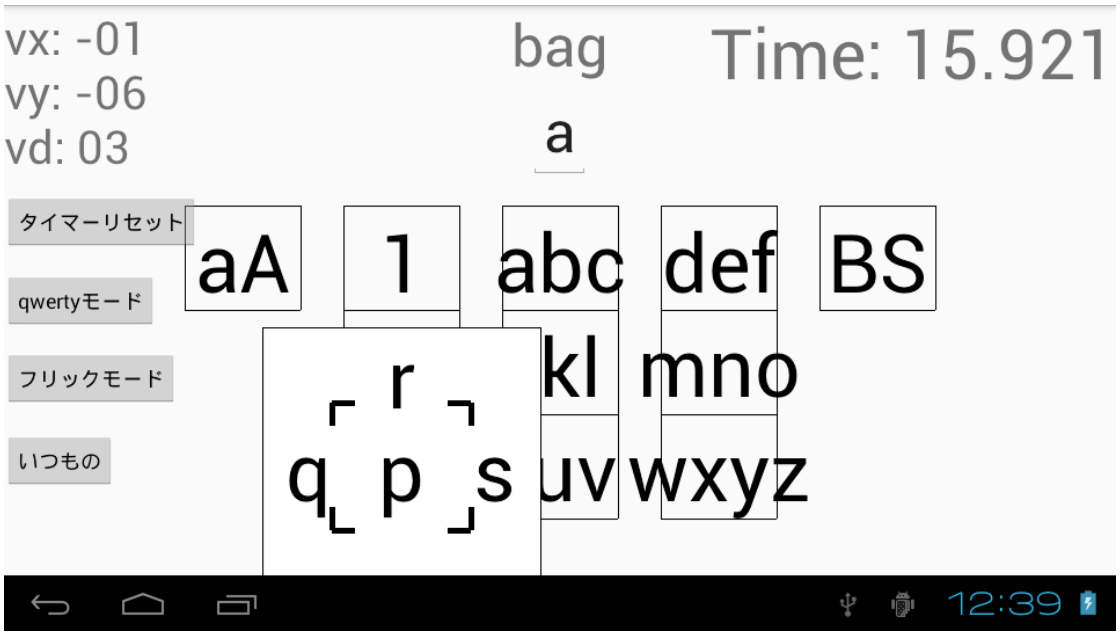

図 3.11: テンキー表示 (押し込み時)

も入力されずに元の表示に戻る．この空フリックの機能は，誤ったキーを押し込んでしまった場合に入力回避を行うために用いることも可能となっている．

3.5 文字の予測

HMD 環境で文字を入力する際に，ランダムな文字列を入力することは比較的少ない．よって，次に入力される文字を予測し，入力のアシストを行うことが有効であると考えられる．連続する 2 文字の間の相関を求めることで，簡易な予測器を実装し，QWERTY キーボードに適用した．学習用のデータベースには，GSL(General Service List)[17] という基本的な英単語を集めたものを使用した．ある文字 c_1 が与えられたときに，次に c_2 が来る確率 $P(c_2|c_1)$ が式.3.9 の条件を満たすような組を求め，サジェストを行う．

$$P(c_2|c_1) > 0.1 \tag{3.9}$$

このような出現頻度の高い文字の組 (c_1, c_2) を表.3.1 に示す．次にくる可能性の高い文字を予測し，通常よりも大きく表示することで，ユーザは所望のキーをよりスムーズにポインティングできる．各 c_1 に対してそれぞれ 1〜5 文字がこの条件を満たし，キーの大きさが $70 \times 85 \rightarrow 100 \times 100$ に変化する．予測によるキーボードの変化の例を図 3.12 に示す．

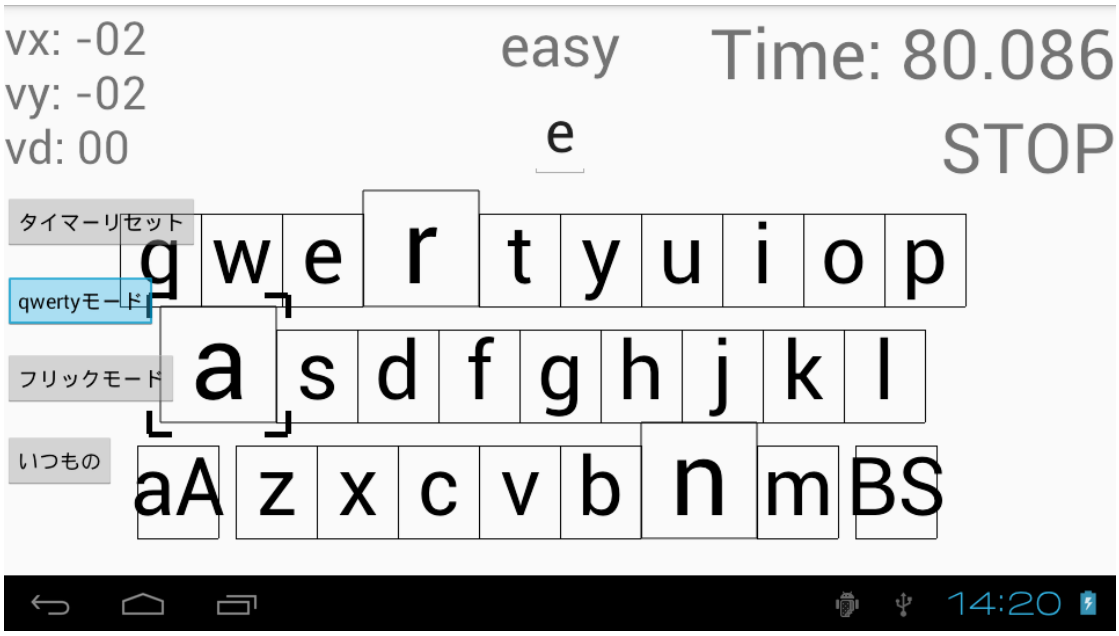

図 3.12: 予測による文字サイズの変化

3.6 カーソル

本研究におけるカーソルは，指の動きに応じて視覚的にフィードバックを行う重要な装置である．このような重要な役割をもつカーソルの位置をどのように決定するか述べる．カーソルを実装するにあたって，次のようなふるまいを考える．

- ユーザがどのキーを選択しているか直観的にわかる
- 入力ジェスチャの最中に不必要に動かないように，位置を一時的に固定する
- 入力したい文字がカーソルによって邪魔されずにしっかり見える

これらを実現するため，本研究では四隅にかぎ型のカーソルを表示させる表示形式を採用する．この形式によって，カーソルがキーにロックされることを直観的に理解できる．入力の最中にキーの文字に被ってしまうことがなく，邪魔にならない．画面上のどこに表示されても見つけやすく，視認性が高い．

カーソルの位置は，指の運動状態によって振る舞いを変える．指の動きが大きいとき，すなわち指が別のキーの位置へ移動しようとしていると推測されるときは，素直に指先位置に従った位置にカーソルが表示される．この際，深度カメラと HMD のディスプレイの画素数が異なっているため，単純な伸縮を行う．深度画像のサイズが $(W_{depth} \times H_{depth})$，ディスプレイが $(W_{disp} \times H_{disp})$ としたとき，伸縮

は式 3.10, 3.10 のように行う．

$$cursor_x = fingertipSmoothed_x[t] \times \frac{W_{disp}}{W_{depth}} \tag{3.10}$$

$$cursor_y = fingertipSmoothed_y[t] \times \frac{H_{disp}}{H_{depth}} \tag{3.11}$$

このカーソル座標はカーソルの中心を示し，実際にカーソルが描画されるのはこの座標の左上，右上，左下，右下の四カ所となっている．

指の動きが遅いとき，すなわち指の移動が終わっていよいよ入力しようという段階，あるいは待機状態のとき，指先位置がキー内部にあればカーソル座標はそのキーの中心座標になり，弱いロック状態となる．このとき，キーの大きさに合わせて少し外側の位置にカーソルが描画される．この弱いロック状態の時には，別のキーへと容易に移動することができるが，後述する強いロック状態のときには別のキーへの移動が難しくなる．

指の深度方向の動きが強いとき，すなわちタップやフリックジェスチャの最中には，カーソル位置はキー上にロックされ，強いロック状態となる．静止状態が２フレーム以上連続する，素早い横移動がある，または文字入力があるのいずれかを満たしたときのみ状態が移行する．強いロック状態の時は，指先位置が他の場所にあってもカーソル位置は変化しない．このようにすることで，ジェスチャの最中に指の xy 座標が動いてしまっても，カーソル位置はジェスチャを行う前に合わせたキー位置に固定されるため，今どのキーに対して入力動作が行われているのか視覚的に判別しやすい．

Algorithm 1 Extract Fingertip

given depthdata[$W_{depth} * H_{depth}$]
for i = 0 → $W_{depth} * H_{depth} - 1$
  if data[i] < THRESHOLD then
    bwdata[i] ← 127
  else
    bwdata[i]← 0
  end if
end for
contours ←findContours(bwdata)
contour ← contours[largest]
if contour.area < AREA_MIN then
  return 0
end if
ptcontour ← contour.toArray
ymin ← HEIGHT
for i = 0 → ptcontour.length-1
  if ptcontour[i].y < ymin then
    ymin ← ptcontour[i].y
    ptid ← i
  end if
end for
idleft ← ptid
idright ← ptid
for i = 0 → 49
  center_points[i].y ← ptcontour[idleft]
  center_points[i].x ← (ptcontour[idleft] + ptcontour[idright])/2
  idleft++
  idright–
end for
x_mean ← Mean(center_points[].x)
y_mean ← Mean(center_points[].y)
d_mean ← Mean(getDistance(depthdata, center_points[].x, center_points[].y)
return [x_mean, y_mean, d_mean]

Algorithm 2 Find tap

given cursorBuf[]
for i = t-9 $\rightarrow$ t
  if cursorBuf[i].d > d_{max} then
    $d_{max} \leftarrow$ cursorBuf[i].d
    $id_{max} \leftarrow$ i
  end if
end for
if d_{max} - cursorBuf[t].d > 3 AND d_{max} - cursorBuf[t].d < 10 then
  id $\leftarrow id_{max} - 1$
  while id $\geq$t-9
    if cursorBuf[t].d == cursorBuf[id].d
    AND ‖cursorBuf[t].x - cursorBuf[id].x‖ <40
    AND ‖cursorBuf[t].y - cursorBuf[id].y‖ <40 then
      tapWaitCount $\leftarrow$ 5
      return cursorBuf[id]
    end if
    id $\leftarrow$ id-1
  end while
end if
return [0, 0, 0]

表 3.1: 出現頻度の高い文字組

c1	c2	$P(c_2\|c_1)$
a	r	0.145
a	t	0.142
a	n	0.120
a	l	0.103
b	e	0.240
b	o	0.143
b	a	0.129
b	l	0.129
b	r	0.124
c	e	0.198
c	o	0.184
c	t	0.108
c	h	0.105
d	e	0.385
d	i	0.185
e	r	0.206
e	n	0.173
e	a	0.110
f	e	0.184
f	a	0.154
f	i	0.145
f	o	0.132
f	f	0.110
g	e	0.271
g	h	0.192
g	r	0.154
h	e	0.268
h	a	0.195
h	o	0.188
h	i	0.176
i	n	0.179
i	o	0.114
i	s	0.101
j	o	0.350
j	e	0.300
j	u	0.300
k	e	0.597
k	i	0.194
k	n	0.113
l	e	0.238
l	a	0.152
l	o	0.146
l	i	0.142
l	l	0.109
m	e	0.261
m	a	0.183
m	p	0.146
m	o	0.132
m	i	0.129
n	t	0.197
n	e	0.136
n	d	0.134
n	c	0.113
n	g	0.103
o	n	0.211
o	r	0.139
p	e	0.203
p	r	0.165
p	o	0.137
p	a	0.129
p	l	0.119
q	u	1.000
r	e	0.269
r	a	0.121
r	o	0.106
s	t	0.216
s	e	0.160
s	i	0.106
t	e	0.217
t	i	0.210
t	h	0.139
u	r	0.170
u	s	0.150
u	n	0.133
v	e	0.723
v	i	0.120
w	e	0.217
w	a	0.211
w	h	0.166
w	i	0.160
w	o	0.114
x	p	0.300
x	c	0.225
x	t	0.225
x	i	0.125
y	e	0.195
y	a	0.171
y	o	0.146
z	e	0.647
z	o	0.176

第4章　評価

HMD 向けの文字入力インターフェースを被験者にテストしてもらい，入力速度およびエラー率の測定を行う．また，アンケートによってユーザビリティの評価を行った．本論文では，お題となる英単語が画面上部に表示され，正しく文字列を入力し終えると自動的に次のお題が出題される形式をとる．比較対象として，HMD に付属しているコントローラを持って従来方式のソフトウェアキーボードによる文字入力をしてもらう．

4.1　エラー率

文字入力おけるエラーは，主に次の二つが挙げられる．

- Incorrect Not Fixed(INF): ユーザが気付いていない誤字であり，最終的に訂正されていないもの
- Incorrect Fixed(IF): 一時的に誤入力され，ユーザによって訂正されたもの

この二つのエラー数と正しく入力された文字数 (Correct: C) を合わせて，Total Error Rate(Total ER) は次のように表される [20]．

$$TotalER = \frac{INF + IF}{C + INF + IF} \times 100 \tag{4.1}$$

また，今回の出題方式では全ての文字が正しく入力されるまで計測を行うため，INF の値は常に 0 となっており，実質的に IF 数の割合を表す修正済みエラー率となっている．

4.2　入力速度

付属のコントローラを用いてタイマーリセットのボタンを押すと測定開始となり，規定数のお題が入力し終わるまでの時間 S[sec] を計測する．入力速度 Words Per Minute(WPM) は，次の式で求める．

$$WPM = \frac{|T|}{S}\frac{60}{w} \tag{4.2}$$

なお，$|T|$ は最終的に入力された文字数 w は 1 単語あたりの平均文字数を表す．英単語の平均長は 5 文字程度であることが知られている [15]．そのため，本論文ではそれに合わせて平均 5 文字となるように出題を行う．

4.3 ユーザビリティ

アンケート調査によって，ユーザビリティの評価を行う．ユーザビリティの尺度としてよく用いられている System Usability Scale(SUS) を用いる [21]．SUS は，10 の設問に対して 1(まったくそう思わない)〜5(まったくそう思う) の 5 段階で回答してもらい，0(使えない)〜100（非常に使いやすい）の間でスコアを算出する．設問は以下のようになっている．

- 1. このシステムをしばしば使いたいと思う
- 2. このシステムは不必要なほど複雑であると感じた
- 3. このシステムは容易に使えると思った
- 4. このシステムを使うのに技術専門家のサポートが必要とするかもしれない
- 5. このシステムにあるさまざまな機能がよくまとまっていると感じた
- 6. このシステムでは、一貫性のないところが多くあったとおもった
- 7. たいていのユーザは、このシステムの使用方法について、素早く学べるだろう
- 8. このシステムはとても扱いにくいと思った
- 9. このシステムを使うのに自信があると感じた
- 10. このシステムを使い始める前に多くのことを学ぶ必要があった

なお，奇数番目の設問はポジティブなもの，偶数番目の設問はネガティブなものとなっている．提案システムの二つのレイアウト及び既存のソフトウェアキーボードに対してそれぞれ上記の 10 項目を回答してもらった．n 番目の質問の回答を $a_n(=1$〜$5)$ とすると，SUS のスコアは次の式で算出される．

$$SUSscore = 2.5 \times \sum_{n=1}^{5}(a_{2n-1} - 1) + 2.5 \times \sum_{n=1}^{5}(5 - a_{2n}) \tag{4.3}$$

アンケートの実施には，Web 上で簡単にアンケートを作成，配布，集計できる SurveyMonkey[22] を利用した．回答画面は図 4.1 のようになっている．

4.4 実験条件

実験にはEPSON製のHMD: MOVERIO BT-2000(図4.2)を用いる．本機には，額の位置に両眼方式の深度カメラが搭載されており，上下方向の調整がある程度可能である．今回は，視線と深度カメラがなるべく近くなるように装着してもらった．はじめに10分程度，操作方法の説明を行い，HMDを実際に装着した状態で練習をしてもらう．その後実際に文字入力をしてもらい，測定を行う．GSL[17]から単語を1つずつ画面上部に出題し，入力し終えると次の単語が出題される形式をとる．また，入力文字数に差がないように，各入力方式で4文字の単語→5文字の単語→6文字の単語という一定の順番で出題され，3単語入力し終えた段階で測定を終了する．比較対象として，BT-2000に付属しているコントローラ(図4.3)をもち，Android端末向けの標準的な文字入力ソフトであるiWnn IME[18]で同様に文字入力をしてもらった．動作速度は，約8fpsとなっている．全ての実験は屋内でおこなっている．

4.5 実験

5人の初級ユーザに提案システムのテストをしてもらった．User1は一度に6単語入力してもらい，User2，User3，User4，User5は二回に分けて3単語ずつ入力してもらった．5人とも２０代学生であり，本システムの操作経験はない．入力速度およびエラー率は表4.1，4.2，4.3のような結果になった．また，ユーザビリティに関するアンケート結果は表4.4，4.5，4.6のようになった．アンケート結果から算出されるSUSスコアは，従来のソフトウェアキーボードで78.5，テンキー配列で49.5，QWERTY配列で63.0となった（図4.4）．

	入力速度 (WPM)	エラー率 (%)
User1	2.5	17
User2-1	6.72	0
User2-2	5.42	0
User3-1	5.43	0
User3-2	4.62	6
User4-1	6.45	0
User4-2	7.45	0
User5-1	5.90	6
User5-2	7.07	17
平均	5.41	6

表 4.1: 入力速度とエラー率 (従来ソフトウェアキーボード)

	入力速度 (WPM)	エラー率 (%)
User1	0.32	62
User2-1	0.70	17
User2-2	0.77	29
User3-1	0.51	25
User3-2	0.64	32
User4-1	0.46	42
User4-2	0.74	32
User5-1	0.82	17
User5-2	0.54	21
平均	0.58	34

表 4.2: 入力速度とエラー率 (テンキー配列)

	入力速度 (WPM)	エラー率 (%)
User1	0.35	42
User2-1	1.63	17
User2-2	1.94	0
User3-1	0.67	21
User3-2	1.04	35
User4-1	0.32	66
User4-2	0.23	69
User5-1	0.87	17
User5-2	0.54	21
平均	0.79	33

表 4.3: 入力速度とエラー率 (QWERTY 配列)

4.6 考察

提案法におけるエラー率および入力速度の個人差が非常に大きく，現状では誰でも手軽に使えるインターフェースとは言い難い．特にエラー率はテンキー配列で平均 34%，最大で 62%，QWERTY 配列で平均 33%，最大で 69% となっており，従来のソフトウェアキーボードの平均 6% に比べて 5 倍以上となってしまい，文字入力インターフェースとしては信頼性が低すぎる．入力速度に関しても，従来方式の 5.41WPM に比べてテンキー配列では 11%，QWERTY 配列では 15%となっており，かなり時間のかかるインターフェースとなってしまっている．しかし，指認識に用いるセンサの性能向上，ジェスチャ認識アルゴリズムの改良，ジャイロセンサとの連動など，まだまだ改良の余地は多い．QWERTY 配列では特に中段のキーの入力速度が遅く，四方を別のキーで囲まれているために，狙った位置をポインティングすることが難しかったと考えられる．中段以外にも，p や BS など右の方のキーは，ユーザの思う前後の移動と深度カメラにおける d 軸方向との間の

	まったく そう思わない	あまり そう思わない	どちらとも いえない	どちらかといえば そう思う	まったく そう思う
しばしば使いたい	0	0	1	3	1
不必要なほど複雑	2	2	1	0	0
容易に使える	0	0	0	4	1
専門家のサポートが必要	2	2	1	0	0
機能がまとまっている	0	0	1	3	1
一貫性がない	2	2	1	0	0
素早く学べる	0	0	1	3	1
とても扱いにくい	2	2	0	1	0
使うのに自信がある	0	0	0	3	2
多くのことを学ぶ必要があった	2	2	1	0	0

表 4.4: アンケート結果 (従来ソフトウェアキーボード)

	まったく そう思わない	あまり そう思わない	どちらとも いえない	どちらかといえば そう思う	まったく そう思う
しばしば使いたい	1	4	0	0	0
不必要なほど複雑	1	4	0	0	0
容易に使える	0	5	0	0	0
専門家のサポートが必要	1	0	3	1	0
機能がまとまっている	0	0	3	2	0
一貫性がない	3	0	2	0	0
素早く学べる	0	1	0	3	1
とても扱いにくい	0	0	0	4	1
使うのに自信がある	1	2	2	0	0
多くのことを学ぶ必要があった	0	3	0	2	0

表 4.5: アンケート結果 (テンキー配列)

ギャップが大きく，タップの認識精度が低かった．QWERTY キーボードでは，平均入力速度 0.79WPM(1 キーあたり 15.2sec) に対して，p は 1 キーあたり 92.3sec，l は 23.0sec かかっていた．また，実験では，次のようなミスが観測できた．

- **false-positive**　キーの選択中などに，意図せずタップやフリック判定されてしまう
- **false-negative**　ジェスチャを行っても何らかの原因で認識されず，文字が入力されない
- **wrong entry**　ジェスチャのやカーソル位置のずれによって別の文字が入力されてしまう

これらのうち，false-positive と wrong entry が実際に入力されたエラーとしてエラー率に反映されるが，false-negative に関しては正確な数値を出すことは難しく，入力速度低下という形でしか反映されていない．false-positive 率と false-negative 率はトレードオフの関係にあるが，提案法ではどちらも多く観測できた．

アンケート調査では，従来法が一番 SUS スコアが高く (78.5)，次に提案法の QWERTY 配列 (63.0)，そしてテンキー配列 (49.5) という順になった．10 の質問のうち，従来法と提案法の間に大きな差が見られたのは,「しばしば使いたい」「容易に使える」「とても扱いにくい」「使うのに自信がある」の 4 つの項目である．特に「とても扱いにくい」の項目では，従来法で回答の平均が 2.0（あまりそう思わ

	まったく そう思わない	あまり そう思わない	どちらとも いえない	どちらかといえば そう思う	まったく そう思う
しばしば使いたい	1	2	1	1	0
不必要なほど複雑	1	3	0	1	0
容易に使える	0	1	2	1	1
専門家のサポートが必要	1	2	1	1	0
機能がまとまっている	0	0	2	3	0
一貫性がない	3	2	0	0	0
素早く学べる	0	0	0	3	2
とても扱いにくい	0	1	2	1	1
使うのに自信がある	1	1	0	3	0
多くのことを学ぶ必要があった	1	3	0	1	0

表 4.6: アンケート結果 (QWERTY 配列)

ない），テンキー配列で 4.2（どちらかといえばそう思う），QWERTY 配列で 3.4（どちらともいえない）となり，ユーザの提案法に対するネガティブな感情を象徴している．逆に,「不必要なほど複雑」「専門家のサポートが必要」「一貫性がない」の 3 つの項目に関しては，各手法の間でほとんど差が見られなかった．このようなアンケート結果から推定できる提案法の特徴は,「一見シンプルですぐに操作方法がわかるが，いざやってみるとなぜかうまく操作できず，自信をなくしてしまう」というものである．

図 4.1: アンケート回答画面 [22]

図 4.2: BT-2000

図 4.3: BT-2000 付属コントローラ

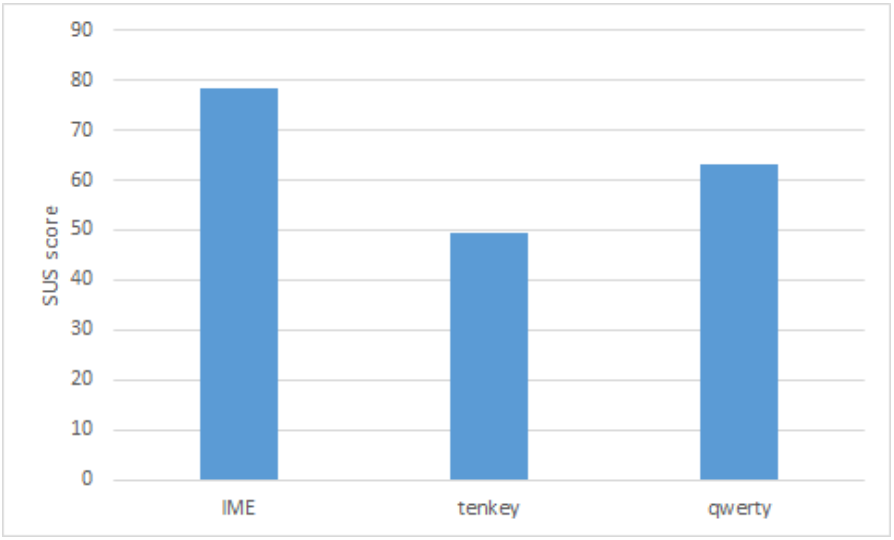

図 4.4: SUS score

第5章　おわりに

本論文では，HMD 環境における文字入力システムの提案を行った．本システムでは，深度カメラによって指先位置を認識し，カーソル位置を対応させる．文字の探しやすさの観点から既存の QWERTY 配列とテンキー配列を実装し，それぞれタップとフリックジェスチャ入力によって文字入力を行う．被験者に実際に HMD を装着してもらい，本システムの 2 つの表示形式と，従来方式のソフトウェアキーボードの比較を行った．入力速度は，従来法で 5.41WPM，テンキー配列で 0.58WPM，QWERTY 配列で 0.79WPM となった．エラー率は，従来法で 6%，テンキー配列で 34%，QWERTY 配列で 33%となった．システムの使いやすさを示す SUS スコアは，従来方式で 78.5，提案法のテンキー配列で 49.5，提案法の QWERTY 配列で 63.0 となった．

参考文献

[1] Hong Hua, Bahram Javidi. A 3D integral imaging optical see-through head-mounted display. Opt. Express 22, 13484-13491, 2014

[2] Muge Goken, A. Nuri Basoglu, Marina Dabic. Exploring Adoption of Smart Glasses: Applications in Medical Industry. In Portland International Conference on Management of Engineering and Technology (PICMET), pp. 3175-3184, 2016

[3] E. De Buyser, E. De Coninck, B. Dhoedt and P. Simoens, "Exploring the potential of combining smart glasses and consumer-grade EEG/EMG headsets for controlling IoT appliances in the smart home," 2nd IET International Conference on Technologies for Active and Assisted Living (TechAAL 2016), London, 2016, pp. 1-6.

[4] J. R. Bruun-Pedersen, S. Serafin and L. B. Kofoed, "Going Outside While Staying Inside — Exercise Motivation with Immersive vs. Nonimmersive Recreational Virtual Environment Augmentation for Older Adult Nursing Home Residents," 2016 IEEE International Conference on Healthcare Informatics (ICHI), Chicago, IL, USA, 2016, pp. 216-226.

[5] R. McCall, B. Martin, A. Popleteev, N. Louveton and T. Engel, "Text entry on smart glasses," 2015 8th International Conference on Human System Interaction (HSI), Warsaw, 2015, pp. 195-200.

[6] Sarah D. Miyahira, Raymond A. Folen, Melba Stetz, Albert Rizzo, Michelle M. Kawasaki. Use of immersive virtual reality for treating anger. In IOS Press, pp. 82-86, 2010

[7] Mark Billinghurst, Hirokazu Kato. Collaborative mixed reality. In First ISMR, pp. 261-284, 1999

[8] Y. Jang, S. T. Noh, H. J. Chang, T. K. Kim and W. Woo, "3D Finger CAPE: Clicking Action and Position Estimation under Self-Occlusions in Egocentric Viewpoint," in IEEE Transactions on Visualization and Computer Graphics, vol. 21, no. 4, pp. 501-510, April 18 2015.

[9] 入江 英嗣, 森田 光貴, 岩崎 央, 千竃 航平, 放地 宏佳, 小木 真人, 樫原 裕大, 芝 星帆, 眞島 一貴, 吉永 努. AirTarget：光学シースルー方式 HMD とマーカレス画像認識による高可搬性実世界志向インタフェース, 情報処理学会論文誌, Vol. 55, pp. 1415-1427. 2014

[10] https://support.microsoft.com/ja-jp/help/12644/hololens-use-gestures

[11] Xiang 'Anthony' Chen, Tovi Grossman, and George Fitzmaurice. 2014. Swipeboard: a text entry technique for ultra-small interfaces that supports novice to expert transitions. In Proceedings of the 27th annual ACM symposium on User interface software and technology (UIST '14). ACM, New York, NY, USA, 615-620.

[12] Aakar Gupta and Ravin Balakrishnan. 2016. DualKey: Miniature Screen Text Entry via Finger Identification. In Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI '16). ACM, New York, NY, USA, 59-70.

[13] Tomoki Shibata, Daniel Afergan, Danielle Kong, Beste F. Yuksel, I. Scott MacKenzie, and Robert J.K. Jacob. 2016. DriftBoard: A Panning-Based Text Entry Technique for Ultra-Small Touchscreens. In Proceedings of the 29th Annual Symposium on User Interface Software and Technology (UIST '16). ACM, New York, NY, USA, 575-582.

[14] Jungpil Shin and Cheol Min Kim. 2016. Character Input System using Fingertip Detection with Kinect Sensor. In Proceedings of the International Conference on Research in Adaptive and Convergent Systems (RACS '16). ACM, New York, NY, USA, 74-79.

[15] P. Panwar, S. Sarcar and D. Samanta, "EyeBoard: A fast and accurate eye gaze-based text entry system," 2012 4th International Conference on Intelligent Human Computer Interaction (IHCI), Kharagpur, 2012, pp. 1-8.

[16] Y. Koizumi et al., "Effective approach to character input for novice BCI users," 2015 10th Asia-Pacific Symposium on Information and Telecommunication Technologies (APSITT), Colombo, 2015, pp. 1-3.

[17] GSL: http://jbauman.com/gsl.html

[18] iWnn IME for Android: http://www.omronsoft.co.jp/product_text/iwnn-ime-for-android/

[19] Yuichiro Abe, Shigenobu Sato, Koji Kato, Takahiko Hyakumachi, Yasushi Yanagibashi, Manabu Ito, Kuniyoshi Abumi. A novel 3D guidance system using augmented reality for percutaneous vertebroplasty. IN JNS 19, pp. 492-501, 2013

[20] A. S. Arif and W. Stuerzlinger, "Analysis of text entry performance metrics," 2009 IEEE Toronto International Conference Science and Technology for Humanity (TIC-STH), Toronto, ON, 2009, pp. 100-105.

[21] J. Brooke, "SUS: A quick and dirty usability scale," in Usability evaluation in industry, P. W. Jordan, B. Thomas, B. A. Weerdmeester, and I.L. McClelland, Eds. London, UK: Taylor and Francis, 1996, pp. 189194.

[22] https://jp.surveymonkey.com/

対外発表

ポスター発表 (査読なし)

1. 土井秀信, 入江英嗣, 坂井修一:
   深度カメラを利用した直感的な文字入力インターフェースの提案,
   電子情報通信学会技術研究報告, Vol.116, No. 240, pp. 3—5 (2016)

謝辞

坂井修一教授には，時に厳しいご指導をいただいたり，ミーティングで意見をいただくなど，大変お世話になりました．

入江英嗣准教授には，原稿のチェックや開発のアドバイスなど，多方面でのご指導をいただきました．

秘書の八木原晴水さん，赤羽彩子さんには，書類や事務手続き等の際に丁寧なサポートをしていただきました．

また，その他の研究室のメンバーには，ちょっとした相談に乗ってもらったり，共通の趣味の話をするなど，著者の心の支えになりました．