

3次元ストロークを用いた文字入力手法

柳原 直貴^{†1,a)} 志築 文太郎^{†2,b)} 高橋 伸^{†2,c)}

概要: 片手持持可能なコントローラの3次元移動を用いた文字入力手法を示す。本手法において用いられるキーボードは、中央を除く縦、横、奥に $3 \times 3 \times 3$ に配置された合計26個のキーにより構成された立方体状のキーボード（キューブキーボード）である。キューブキーボードは、コントローラの位置に同期して動き、コントローラのトリガが押下されている間その場で静止する。キューブキーボードが静止している間、カーソルはコントローラの位置に同期して動く。ユーザは縦り順にカーソルをキーに当てて行くこと（3次元ストローク）により、単語単位での文字入力ができる。我々は、仮想現実環境向けに本手法のプロトタイプシステムを実装したので報告する。

1. はじめに

仮想現実（VR）システムに標準搭載されている仮想キーボードは横幅が長い平面状であり、またその表示位置は固定されている。これに伴い、仮想キーボードの視野占有率が大きくなるため、他のコンテンツを見ながら文字入力を行うことを妨げる。

我々は、この問題を解決するため、片手持持可能なコントローラの3次元移動を用いた文字入力手法を開発している（図1）。本手法において用いられるキーボードは、中央を除く縦、横、奥に $3 \times 3 \times 3$ に配置された合計26個のキーにより構成された立方体状のキーボード（以降、キューブキーボード）である。キューブキーボードは、コントローラの位置に同期して動き、コントローラのトリガが押下されている間その場で静止する。キューブキーボードが静止している間、カーソルはコントローラの位置に同期して動く。ユーザは縦り順にカーソルをキーに当てて行くこと（3次元ストローク）により、単語単位での文字入力ができる。このように、キューブキーボードでは縦と横の他に奥にもキーが配置されるため、キーボードの視野占有率を抑えられ、また、ユーザは自由な位置で文字入力ができる。

なお、3次元ストロークによりカーソルが単語の縦りに

含まれないキーに接触するため、我々は2つの単語入力手法を試みている。一つは、カーソルの移動速度を基にキーを選択する手法 [14]、もう一つはカーソルの移動軌跡に基づいて単語予測を行う手法である。

本稿においては2つの単語入力手法、およびそれらの入力性能を調査するために行った比較実験を述べる。

2. 関連研究

これまでに、VR環境における文字入力手法、ジェスチャーキーボードに関する研究が多数行われている。

2.1 VR環境における文字入力手法

Rajannaら [9] は、視線を認識できるヘッドマウントディスプレイ（以降、HMD）を用いて凝視に基づく文字入力手法を開発した。ユーザは前方に表示されるキーボードの中のキーを凝視した状態においてコントローラのボタンを押すことにより凝視されたキーを入力できる。QWERTY配列を採用したVR環境向けの文字入力手法 [4,12] も提案されている。特に、Speicherら [12] は、VR環境上にて6種類の文字入力手法の性能比較を実施している。結果として、6手法において最も性能の良い手法は、コントローラによりキーをレーザーポインタのように指して選択する手法であると述べている。

これらのキーを一つずつ選択する文字単位の入力とは異なり、我々の手法は、3次元ストロークを用いた単語単位の入力である。また、キーボードの位置をユーザはコントローラを使って自由に決められる。

^{†1} 筑波大学コンピュータサイエンス専攻

Department of Computer Science, University of Tsukuba

^{†2} 筑波大学システム情報系

Faculty of Engineering, Information and Systems, University of Tsukuba

a) yanagihara@iplab.cs.tsukuba.ac.jp

b) shizuki@cs.tsukuba.ac.jp

c) shin@cs.tsukuba.ac.jp

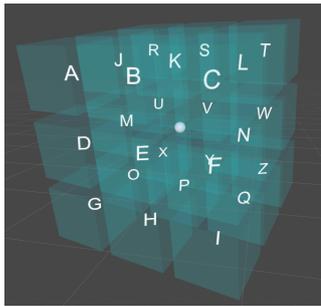


図 1 本手法において使用するキューブキーボード.

表 1 キューブキーボードのキーの配置.

x	-1	0	1	-1	0	1	-1	0	1
$y = 1$	A	B	C	J	K	L	R	S	T
$y = 0$	D	E	F	M		N	U	V	W
$y = -1$	G	H	I	O	P	Q	X	Y	Z
z		-1			0			1	

2.2 ジェスチャキーボード

タッチディスプレイにおける文字入力を高速化する手法の一つとしてジェスチャキーボードが研究開発されている [5, 7, 15]. ユーザは単語の綴り順に指をキーに当てて行くという 2 次元のジェスチャにより文字入力を行える. この入力手法は, 単語単位で文字入力ができるため効率が良い. SHARK² [7] は, ATOMIK キーボードにジェスチャキーボードを適用した研究である. SHARK² は, ジェスチャの入力位置や形状および言語要素を基に文字予測することにより, 高速かつ高精度な文字入力手法を実現している. Rick [10] は, 様々なキー配列のキーボードの入力性能を比較することによって, ジェスチャキーボードにおいて最適なキー配列を調査している. また, WatchWriter [5] は, スマートウォッチにジェスチャキーボードを適用したシステムである.

以上の手法は 2 次元のジェスチャを用いた入力手法である. 一方, 提案手法は, キーを縦と横の他に奥にも配置した上で 3 次元のジェスチャを用いることにより入力効率の向上を図る.

3. キューブキーボード

本手法において使用するキューブキーボードを図 1 に示す. キューブキーボードは, 中央を除く縦, 横, 奥に $3 \times 3 \times 3$ に配置された合計 26 個のキーにより構成された立方体状のキーボードである. キューブキーボードの各キーは表 1 に従って配置されている. なお, キューブキーボードは, コントローラの位置に同期して動く. ただし, コントローラのトリガが押下されている間, 同期が解除されその場で静止する. 一方, この間, キューブキーボードの中心に配置されたカーソルがコントローラの位置に同期して動く.

3.1 文字入力手法

ユーザは, 片手持持可能なコントローラを用いて文字入力を行う. まず, ユーザはコントローラを使用して文字入力したい位置にキューブキーボードを移動させる. コントローラのトリガを押下するとキューブキーボードはコントローラとの同期が解除され, カーソルのみがコントローラに同期して動くようになる. この間に, ユーザは 3 次元ストロークにより, 単語の入力を行う. 最後に, ユーザはトリガを離して 1 単語の文字入力を終える. この後, ユーザは再びトリガを押下することにより, 次の文字入力へと遷移させることができる. このとき, 以前に入力した単語と次に入力する単語の間の空白文字は自動的に入力される.

なお, 3 次元ストロークが行われる際, カーソルが入力したい単語の綴りに含まれないキーに触れることがある. したがって 3 次元ストロークを用いて単語単位の文字入力を行うためには, 入力したい単語の綴りに含まれないキーに触れても正しい単語を入力できるような機能が必要である. 我々は, カーソルの移動速度に基づいてキーを選択する手法およびカーソルの移動軌跡に基づいて単語を予測する手法を試みた.

3.1.1 カーソルの移動速度に基づくキーの選択手法

カーソルの移動速度に基づくキーを選択する手法 (speed-based SB 手法) は, カーソルの移動速度が大きいときに接触したキーを無視し, 小さい時に接触したキーを単語の綴りに含まれるキーとする. カーソルの移動速度の大小は, 式 (1) を用いて判別される. 式中の, P_n はカーソルがキーに接触した際のコントローラの位置, P_p はカーソルがキーに接触する 1 フレーム前のコントローラの位置, V_{th} は移動速度の閾値である (V_{th} は 5 mm と設定した). 式 (1) において $f(P_p, P_n) = 1$ の時, 触れたキーは単語の綴りとして選択されたキーであると判定され, $f(P_p, P_n) = 0$ の時, 単語の綴りとして選択されなかったキーであると判定される.

$$f(P_p, P_n) = \begin{cases} 1 & (|P_p - P_n| < V_{th}) \\ 0 & (otherwise) \end{cases} \quad (1)$$

$|a - b|$: a-b 間のユークリッド距離

3.1.2 カーソルの移動軌跡に基づく単語予測手法

カーソルの移動軌跡の曲がる位置を検出し単語予測する手法 (gesture-based GB 手法) は, 次の手順により単語予測を行う.

- (1) カーソルが通過したキー列を取得
 - (2) カーソルの移動軌跡が曲がる位置にてキー列を分割
 - (3) 分割した各文字列の中から少なくとも一つ以上の文字を含む正規表現を作成
 - (4) 正規表現にマッチした単語を予測単語リストに登録
- カーソルの移動軌跡が曲がる位置を基に触れたキー列 S_{input} を分割した例を表 2 に示す. 例として, 3 次元ス

表 2 カーソルの移動軌跡の曲がる位置を基に S_{input} を分割した例.

単語	触れたキー列 (S_{input})	S_{input} の分割結果
available	akvkabefiflbabble	<u>akv</u> , <u>ka</u> , <u>befi</u> , <u>fl</u> , <u>ba</u> , <u>bbl</u> , <u>e</u>
japanese	jadmpdabnekske	<u>jad</u> , <u>mp</u> , <u>da</u> , <u>bn</u> , <u>e</u> , <u>ks</u> , <u>ke</u>
water	watklekr	<u>wa</u> , <u>tk</u> , <u>le</u> , <u>kr</u>

表 3 辞書の単語とマッチした単語および検索時間.

S_{input}	マッチした単語	検索時間 (秒)
akvkabefiflbabble	available	0.0060
jadmpdabnekske	japanese	0.0113
watklekr	water	0.0070

トロークを用いて「available」を入力した際、 S_{input} は「akvkabefiflbabble」となり、この例は S_{input} 中の文字の順にカーソルがキーに触れたことを示している。ここで、 S_{input} をカーソルの移動軌跡が曲がる位置にて分割すると「akv, ka, befi, fl, ba, bbl, e」と分割できる。このとき、表 2 の S_{input} を分割した結果から、下線部に示すように分割した各文字列の中に「available」の綴りが 1 つ以上含まれていることが確認できる。この特徴を利用して、分割した各文字列の中から少なくとも 1 文字以上の文字を含むような正規表現を作成する。

分割した各文字列に用いる正規表現

「available」の例において S_{input} を分割した文字列の一番最初である「akv」から作成した、キー列中の少なくとも 1 文字以上の文字を含む正規表現を以下に示す。

$$(a|k|v|ak|kv|av|akv)$$

$$(a_1|a_2): a_1 \text{ か } a_2 \text{ を含む}$$

以上の正規表現を分割した各文字列全てについて同様に作成し、1 単語分の正規表現を生成する。その後、英単語辞書の全ての単語に対して 1 単語分の正規表現がマッチするか検索する。GB 手法においては、Google の Trillion Word Corpus の N グラム頻出度解析により得られた上位 10000 単語を収録した辞書 [1] を用いる。表 2 の S_{input} に対して検索をかけた結果を表 3 に示す。表 3 の結果においてマッチした単語はそれぞれ 1 単語であった。マッチした単語は表 2 の入力したい単語と一致している。探索時間についても、1/60 秒以下であるため、提案手法はリアルタイムに予測単語リストを更新できる。

カーソルの移動軌跡が曲がる位置の検出方法

カーソルの移動軌跡が曲がる位置を検出する方法として、X, Y, および Z 軸の各移動ベクトルを分析し、各移動ベクトルの正負が反転した位置をジェスチャの曲がる位置として検出する方法を導入する。 n 番目に触れたキーの位置および $n-1$ 番目に触れたキーの位置との差のベクトル V_n を式 (2) により定義する。

$$V_n = \begin{cases} (0, 0, 0) & (n = 0) \\ P_n - P_{n-1} & (\text{otherwise}) \end{cases} \quad (2)$$

ここで、判定ベクトル V_{judge} を用いて V_{judge} および V_n の各要素の正負が全て同じの場合、位置 n においてカーソルの移動軌跡は曲がっていないと判断し、いずれか 1 組の正負が異なる場合、移動軌跡は曲がっていると判断する。 V_{judge} の初期値は V_0 とし、この判定手法においては、0 は正負の両方に含まれると定義する。

曲がる位置の検出例として、「available」の S_{input} に着目する。まず、最初の文字の「a」の位置は表 1 から $P_a = (-1, 1, -1)$ であり、この位置を初期位置 P_0 とする。次の文字の「k」の位置は $P_k = (0, 1, 0)$ であり、この位置を P_1 とする。この時、移動ベクトル V_1 は $V_1 = P_1 - P_0 = (1, 0, 1)$ となる。ここで、 V_{judge} および V_1 の各要素の正負は同じであるため、a-k 間の移動軌跡は曲がっていないと判断される。また、判定ベクトル V_{judge} は $V_1 + V_{judge} = (1, 0, 1)$ と更新される。次に、 $P_2 = P_v = (0, 0, 1)$ とすると、移動ベクトル V_2 は $V_2 = P_2 - P_1 = (0, -1, 1)$ となる。ここで、 V_{judge} および V_2 の各要素の正負は同じであるため、k-v 間の移動軌跡も曲がっていないと判断される。また、判定ベクトル V_{judge} を $V_2 + V_{judge} = (1, -1, 2)$ と更新する。次の文字の「k」の位置を $P_3 = P_k = (0, 1, 0)$ と置くと、 $V_3 = P_3 - P_2 = (0, 1, -1)$ となる。ここで、 V_{judge} および V_3 の各要素の正負を確認すると y 座標および z 座標において正負が異なるため、v-k 間の移動軌跡は曲がっていると判断され、「akv」と「kabefiflbabble」は分割される。この処理をキー列全てに行うことにより、 S_{input} は「akv, ka, befi, fl, ba, bbl, e」と分割される。

予測単語の選択方法

予測単語リストに登録された単語は、図 2 に示すようにキューボードの上部に単語キーとして現れる。予測単語リストは最大 30 単語に登録できる。文字入力を行い S_{input} が変化するたび、予測単語リストは更新されるので単語キーも変わる。3 次元ストロークによる文字入力を終えた後、ユーザはカーソルを単語キーに接触させることによりその単語が入力される。一方、この接触が行われない場合、予測単語リストの先頭要素（すなわち正規表現にマッチする単語のうち最頻出の単語）が自動的に入力される。よって、変換された単語が入力したい単語と一致していた場合、単語キーを選択する操作を省略できる。なお、予測単語リストに何も登録されていない場合は、SB 手法により得られた文字列を出力する。この機能により、ユーザは辞書に存在しない単語も SB 手法により入力できる。

4. 2 つの入力手法の比較実験

SB 手法および GB 手法の入力性能を比較するために、評価実験を実施した。入力対象とする文章は Vertanen ら [13] のフレーズセットから、20-28 文字からなる英文 (全 30 文章) を対象とした。

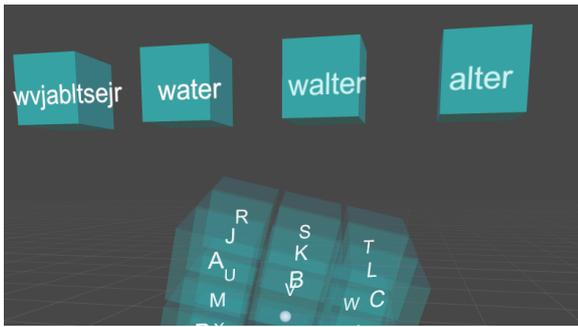


図 2 予測候補キーの表示位置。一番左のキーのみ SB 手法により得られた文字列を表示。



図 3 実験にて使用した VIVE コントローラ。(左) コントローラの正面図。(右) コントローラの側面図。

4.1 実験参加者およびシステム

研究室内の大学生および大学院生を含む 4 人 (22–24 歳, 平均年齢 22.5 歳, 全員男性, P1–P4) を実験参加者として実験を行なった。参加者全員が右利きであり VR の経験がほとんどなかった。

実験は, VR 環境向けに作成した提案手法のプロトタイプシステムを用いて行った。プロトタイプシステムでは, 文字入力をしていない間にコントローラのグリップボタンを押すと単語単位での文字削除が可能である。

実験の様子を図 4 に示す。実験には, VR デバイス (HTC VIVE, 解像度: 両眼 2160 × 1200 ピクセル, 視野角: 110 度, 自由度: 6DoF) を用いた。

4.2 実験内容

まず, 参加者の頭部に HMD を装着し, 右手に VIVE コントローラ (図 3) を渡した。参加者に実験システムの操作方法および入力手法を説明し, その後, 10 分間の練習時間を設けた。練習後, 参加者は HMD を外し, 1 分以上の休憩をとるよう指示した。参加者には, コントローラのグリップボタンを押すと単語単位で削除できること, メニューボタンを押すと測定が開始し再度押すと測定が終了することを説明した。またタスクを行う前に, 参加者には, 前方 (VR 環境の z 軸正の方向) に表示された課題文をなるべく記憶してから入力すること, 正確にかつ素早く入力すること, 3 回程度入力した単語が作れなかった場合次の単語の入力を始めることを指示した。その後, 参加者は各

手法によりタスクを行うことを指示した。

タスク終了後には, ユーザビリティの評価のため System Usability Scale (SUS) [2] を用いたアンケート, 「どちらの手法がより楽にタスクを行えましたか」という問および各手法に対する自由記述欄を含むアンケートを行った。

4.3 タスク

参加者の前方に表示される課題文と同じ文章の入力を行うことを 1 タスクとした。タスク開始時には, 参加者にコントローラのメニューボタンを押してもらい, 終了後にはもう一度メニューボタンを押してもらった。測定終了後, 課題文は自動的に次の課題文へと切り替わる。5 文章の入力を 1 セッションとした。1 セッション終了ごとに, 参加者には HMD を外してもらい, 1 分以上の休憩をとってもらった。3 セッション終了後に, もう片方の手法に対して同様のタスクを行った。2 種類の手法の測定終了後, 1 時間以上の時間を空け, 今度は測定する手法の順番を逆にして, 同様の実験手順の測定を行った。つまり, 1 回目に SB 手法–GB 手法の順に測定を行った場合, 2 回目は GB 手法–SB 手法の順に測定を行った。よって, 合計で 240 回 (= 5 タスク × 3 セッション × 2 手法 × 2 回 × 4 参加者) 測定を行った。

4.4 実験結果および考察

評価対象は, 単語入力速度 (WPM), 提示された文章と入力された文章から算出されるエラー率 (MSD エラー率) [8, 11], SUS の得点 [2] である。MSD エラー率は, 提示された文章を P , 入力された文章を T として式 (3) により算出される。

$$MSD \text{ エラー率} = \frac{MSD(P, T)}{\max(|P|, |T|)} \times 100\% \quad (3)$$

セッションごとの各手法の入力速度を図 5 に示す。また, 合わせて手法ごとの対数近似曲線を示す。両手法とも右上がりのグラフを描き, 全てのセッションにおいて GB 手法の入力速度が高くなった。取得した入力速度のデータに正規性がなかったため, ウィルコクソンの符号順位検定を行った。その結果, 手法間の入力速度には有意な差が存在する事がわかった ($p = 6.573 \times 10^{-6} < .05$)。また, 図 5 の対数近似曲線が右上がりの曲線を描いていることから, 両手法の入力速度はセッション数を重ねるごとに高くなると考えられる。

セッションごとの各手法の MSD エラー率の結果を図 6 に示す。全てのセッションにおいて GB 手法の方が MSD エラー率が低かった。取得した MSD エラー率に正規性がなかったため, ウィルコクソンの符号順位検定を行った。その結果, 手法間の MSD エラー率において有意な差が存在する事がわかった ($p = 2.888 \times 10^{-8} < .05$)。この結果から, GB 手法の方が入力精度は高いことがわかった。



図 4 実験の様子。

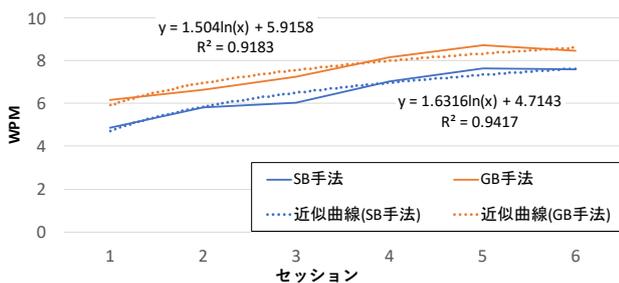


図 5 セッションごとの各手法の WPM。

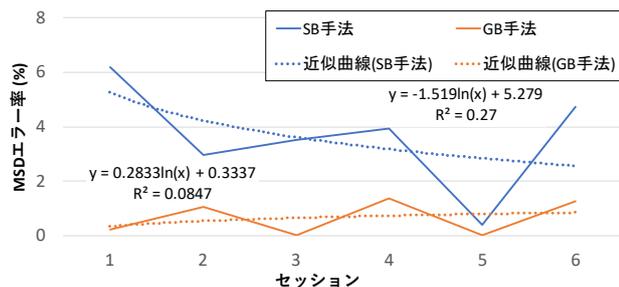


図 6 セッションごとの各手法の MSD エラー率。

SUS の得点は、SB 手法 ($M = 50.0, SD = 5.45$), GB 手法 ($M = 66.3, SD = 3.70$) となった。取得した SUS の得点には、正規性および等分散性があったため Student の t 検定を行った。その結果、手法間の SUS の得点には有意な差が存在することがわかった ($p = .01 < .05$)。

また、事後アンケートの「どちらの手法がより楽にタスクを行えましたか」という問に対して、参加者全員が GB 手法と回答した。

5. 議論

今回の実験にて得られた SUS の得点においては、GB 手法 (66.3) は SB 手法 (50.0) に比べて有意に高かった。しかし、これらの得点は、どちらも SUS の平均得点である 68 [3] を下回っている。すべての参加者は実験後に、腕を上げたまま操作し続ける必要があったため腕がつかれたと述べていた。また、多くの参加者はキー配置が覚えにくかったと述べていた。提案手法は、これらの原因により SUS の得点が低くなったと考えられる。さらに、GB 手法

は、SB 手法に比べて遅い 3 次元ストロークによる入力が可能であった。そのためキーの位置を確認しながら文字を入力するときや、腕が疲労したときの入力が容易である。これが要因となって GB 手法の SUS の得点は SB 手法に比べて有意に高くなったと考えられる。実際に P2 は GB 手法に対して、キー配置を覚えていないときにゆっくり選択できたため扱いやすかったとアンケートの自由記述欄に記述している。

提案手法は、同じ単語を入力する際、キューブキーボードの向きにより 3 次元ストロークが変化する。P4 は、同じ単語を入力するとき、コントローラの向きに依存して 3 次元ストロークが異なる角度になり入力しにくいとアンケートの自由記述欄に記述している。特に、キューブキーボード自体が傾いた状態において 3 次元ストロークを行うと誤入力が多くなることがわかった。

提案手法は、文字単位の削除機能を追加することによって入力性能が向上する可能性がある。今回の実験においては、参加者は単語単位の削除しかできなかった。そのため、単語中の 1 文字の修正を行うためにその単語を入力し直す必要があった。特に、SB 手法においては、1 文字でも選択を誤ると意図した単語の入力ができない。P1 や P4 は、文字単位で削除する機能が欲しいとアンケートの自由記述欄に記述している。

また、課題文の表示位置が測定結果に影響した可能性がある。GB 手法において用いられる予測単語キーはキューブキーボードの上に表示されるため、予測単語キーの文字と課題文が重なってしまうことがある。これにより、予測単語キーの文字が見づらくなってしまい GB 手法の入力性能が悪くなったと考えられる。

6. 今後の課題

本稿におけるキューブキーボードのキー配置はプロトタイプである。次に入力される可能性の高いキーを隣接するキーに配置することにより、カーソルの移動軌跡の長さは短くできる。QWERTY キーボードのような既存のキー配置を基に設計することにより、キーをユーザが記憶しやすい位置に配置できる。我々は、3 次元ストロークが効率的に実行できるような最適なキー配置を検討する。

提案した両手法は、単語の綴りに含まれるキーに必ず触れなければ正しい単語を入力できない課題がある。タッチディスプレイ上におけるジェスチャキーボード [5, 7, 15] は、機械学習を用いて辞書に収録された単語のジェスチャモデルを学習し入力ジェスチャの予測を行うため、単語の綴りに含まれるキーに触れなくても正しい単語を入力できる。この予測手法を 3 次元に拡張し 3 次元のジェスチャを予測できれば、課題が解決され予測性能が向上すると考えられる。

実験において、疲れに関するフィードバックを得た。全ての参加者が提案手法による文字入力を長時間行うと腕が疲れると述べていた。そのため、今後は腕の疲れを定量的に測定するため、Consumed Endurance [6] を使用した実験を予定している。

7. まとめ

本稿にて、片手把持可能なコントローラの3次元的な移動を用いた文字入力手法を示した。提案手法は、キーボードの視野占有率を抑えられ、VRシステムに標準搭載されている仮想キーボードの視野占有率が大きくなる問題を解消した。3次元ストロークによりカーソルが単語の綴りに含まれないキーに接触するため、我々はカーソルの移動速度を基にキーを選択するSB手法およびカーソルの移動軌跡に基に単語予測を行うGB手法を提案した。2つの入力手法の入力性能についての比較実験を通して、GB手法の方が入力速度、入力精度、ユーザビリティが有意に高いことがわかった。

参考文献

- [1] google-10000-english, <https://github.com/first20hours/google-10000-english>. 2018年12月20日に閲覧.
- [2] Brooke, J.: SUS: A Quick and Dirty Usability Scale, *Usability Evaluation in Industry*, pp. 189–194 (online), available from (<https://ci.nii.ac.jp/naid/10024892921/>) (1996).
- [3] Brooke, J.: SUS: A Retrospective, *Journal of Usability Studies*, Vol. 8, No. 2, pp. 29–40 (online), available from (<http://dl.acm.org/citation.cfm?id=2817912.2817913>) (2013).
- [4] Carter, L. and Potter, L. E.: Spatial Virtual Keyboard for Wand Based Virtual Reality, *Proceedings of the 5th Symposium on Spatial User Interaction*, SUI '17, New York, NY, USA, ACM, pp. 161–161 (online), DOI: 10.1145/3131277.3134357 (2017).
- [5] Gordon, M., Ouyang, T. and Zhai, S.: WatchWriter: Tap and Gesture Typing on a Smartwatch Miniature Keyboard with Statistical Decoding, *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, New York, NY, USA, ACM, pp. 3817–3821 (online), DOI: 10.1145/2858036.2858242 (2016).
- [6] Hincapié-Ramos, J. D., Guo, X., Moghadasian, P. and Irani, P.: Consumed Endurance: A Metric to Quantify Arm Fatigue of Mid-air Interactions, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '14, New York, NY, USA, ACM, pp. 1063–1072 (online), DOI: 10.1145/2556288.2557130 (2014).
- [7] Kristensson, P.-O. and Zhai, S.: SHARK²: A Large Vocabulary Shorthand Writing System for Pen-based Computers, *Proceedings of the 17th Annual ACM Symposium on User Interface Software and Technology*, UIST '04, New York, NY, USA, ACM, pp. 43–52 (online), DOI: 10.1145/1029632.1029640 (2004).
- [8] MacKenzie, I. S. and Soukoreff, R. W.: A Character-level Error Analysis Technique for Evaluating Text Entry Methods, *Proceedings of the Second Nordic Conference on Human-computer Interaction*, NordiCHI '02, New York, NY, USA, ACM, pp. 243–246 (online), DOI: 10.1145/572020.572056 (2002).
- [9] Rajanna, V. and Hansen, J. P.: Gaze Typing in Virtual Reality: Impact of Keyboard Design, Selection Method, and Motion, *Proceedings of the 2018 ACM Symposium on Eye Tracking Research & Applications*, ETRA '18, New York, NY, USA, ACM, pp. 15:1–15:10 (online), DOI: 10.1145/3204493.3204541 (2018).
- [10] Rick, J.: Performance Optimizations of Virtual Keyboards for Stroke-based Text Entry on a Touch-based Tabletop, *Proceedings of the 23rd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, New York, NY, USA, ACM, pp. 77–86 (online), DOI: 10.1145/1866029.1866043 (2010).
- [11] Soukoreff, R. W. and MacKenzie, I. S.: Measuring Errors in Text Entry Tasks: An Application of the Levenshtein String Distance Statistic, *CHI '01 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '01, New York, NY, USA, ACM, pp. 319–320 (online), DOI: 10.1145/634067.634256 (2001).
- [12] Speicher, M., Feit, A. M., Ziegler, P. and Krüger, A.: Selection-based Text Entry in Virtual Reality, *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, CHI '18, New York, NY, USA, ACM, pp. 647:1–647:13 (online), DOI: 10.1145/3173574.3174221 (2018).
- [13] Vertanen, K. and Kristensson, P. O.: A Versatile Dataset for Text Entry Evaluations Based on Genuine Mobile Emails, *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, New York, NY, USA, ACM, pp. 295–298 (online), DOI: 10.1145/2037373.2037418 (2011).
- [14] Yanagihara, N. and Shizuki, B.: Cubic Keyboard for Virtual Reality, *Proceedings of the Symposium on Spatial User Interaction*, SUI '18, New York, NY, USA, ACM, pp. 170–170 (online), DOI: 10.1145/3267782.3274687 (2018).
- [15] Zhai, S., Kristensson, P. O., Gong, P., Greiner, M., Peng, S. A., Liu, L. M. and Dunningan, A.: Shapewriter on the iPhone: From the Laboratory to the Real World, *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '09, New York, NY, USA, ACM, pp. 2667–2670 (online), DOI: 10.1145/1520340.1520380 (2009).