タブレット上でブラインドタッチを可能にするフレキシブルキーボードの提案

田坂 友杉　秋岡 明香

タブレット端末が普及する現代において，タブレット端末上でブラインドタッチを行うと文字入力の煩わしさを感じることがある．その煩わしさは仕事効率の低下を引き起こす原因となる．文字入力の煩わしさの一つとしてミスタイプが挙げられる．本研究では，ミスタイプの原因を解決することで煩わしさを改善させる．本手法では，タブレット端末上でユーザ一人一人に合わせたソフトウェアキーボードを提案する．まず，ユーザがミスタイプする原因の一つとして実際に入力しているキーとユーザが認識しているキーとが異なっている問題がある．そのため，ブラインドタッチが困難である．そこで，ユーザが認識しているキー配置にすることで問題の解決を行う．

In the modern day when tablet-type devices are popular, when touch-type is performed on tablet-type devices, they will be troublesome to input characters. The troubles cause of lowering work efficiency. One of the troubles of inputting characters is typing mistake. In this research, troubles improved by solving the cause of typing mistake. In this method, there is a problem that the actually input key is different from the key recognizing the user. For that reason, touch-type is difficult. Therefore, we solve the problem by using the method of setting the key arrangement recognizing the user.

1　はじめに

タブレット端末が普及する現代において，タブレット端末上での文字入力は必要不可欠である．タブレット端末にはパソコンと同様なタスクを行うことが可能である．例えば，メールやブラウジングである．これらは全て文字入力が必要なタスクである．しかし，ユーザがタブレット端末上でそれらのタスクをブラインドタッチで行おうとすると，文字入力の煩わしさを感じることがある．

文字入力の煩わしさの一つとして，ミスタイプが挙げられる．入力している文字が目的の文字と異なった場合，ユーザはその文字を消さなければならない．文字を消す動作の有無がユーザに煩わしさを与えることがある．そこで本研究では，ミスタイプの原因を解決させる．そして，ブラインドタッチ中にユーザが感じる文字入力の煩わしさを改善させる．ブラインドタッチ中にミスタイプする原因の主として，ユーザが実際に入力しているキー配置とユーザが認識するキー配置とが異なっている問題が挙げられる．本手法では，ユーザが認識しているキー配置にすることで問題の解決を行う．したがって，ユーザが認識している位置にキーを配置するユーザ一人一人に合わせたキーボードを提案する．

本稿ではこのキーボードを実際のタブレット端末上に実装し，従来のソフトウェアキーボードとの比較実験を行った．具体的には，ユーザがブラインドタッチを行なう場合のミスタイプ数を，提案したキーボードと従来のソフトウェアキーボードで比較し，タブレット端末においてユーザが快適にブラインドタッチを行なうための要件を考察する．

2　関連研究

2.1　キーを指の設置位置とその周囲に配置するソ

A proposal of a Flexible Keyboard that enables touch-type on a tablet-type device.

Tomosugi Tasaka, 明治大学大学院先端数理科学研究科ネットワークデザイン専攻, Advanced Mathematical Sceinces based on Netword Design, Meiji University.

フトウェアキーボード

久野ら [2] は,“タッチパネルにおいて，タッチタイピングが容易なソフトウェアキーボード” の提案を行なっている．タッチタイピングが容易でない原因として，久野らは “キーの位置がユーザの手の形状に適応” していないからだと指摘している．

久野らの手法は，キーボードを “手の形状” に合わせる際，ユーザの視覚を頼りに行なっている．そして，ユーザが任意でキーの位置を変化させることが可能である．

タイピングを容易にさせるには，手の形状に合わせることが重要である．しかし，視覚を頼りにした形状変化は，視覚を頼りにしないタッチタイピングを容易になるとは言えないと考える．そこで，本研究では視覚を頼りにしないキーボードの形状を変化させる手法を用いる．

2.2 タッチ画面におけるバブルカーソル状適応型キーボードの提案

遠藤ら [1] の研究は，ソフトウェアキーボードで入力を行う時，“視線の往復” により，“入力速度低下” を問題としている．そのため，遠藤らは入力速度を向上させるために，キーの形状と視覚的補助の両方から，問題解決のアプローチを行なっている．また，遠藤らの手法は，キーの形状をユーザに合わせる際，文字の入力位置と入力回数を参照している．

キーボードの作成において，入力位置は必須な情報と言える．しかし，それぞれのキーの打ちやすさ考えると，入力回数を使ったキーボードの作成は打ちやすさを損ねてしまうと考える．

なぜならば，遠藤らの手法ではユーザに合わせる際，入力回数に応じたキーの重心を移動させることでキーの形状を変化させている．たしかに，キーの重心を移動させることで，対象のキーを打ちやすい形状に変化させることができる．ところが，この手法は入力回数が少ないキーと多いキーでは，打ちやすさに差が生じてしまう．そこで，本研究では入力回数が異なるキーでも，打ちやすさが異なりにくい手法を用いる．

2.3 ソフトウェアキーボードの打鍵パフォーマンス向上のサポートツールの提案

辛島ら [3] の研究は，“QWERTY 配列のレイアウト” を変更させずに，“打鍵パフォーマンス” の向上を目的としている．辛島らは，キーボードの形状を変化させずに，入力の判定位置を変化させるというアプローチを行なっている．

辛島らの手法は，ブラインドタッチができるユーザを対象に，実際の入力座標のデータを参照している．入力座標を元に，実際の判定位置を作成し，キーボードの適応させている．

キーボードの判定位置を作成する際，ブラインドタッチができる自分以外の入力座標も元にしている．つまり，一般的に打ちやすい判定位置である．しかし，一般的な手の形状に合わない，例えば手小さい人や大きい人たちは打ちやすいとは言えないと考える．

キーボードは一般的な人たちが入力しやすいことは重要である．しかし，物理キーボードではないソフトウェアキーボードは，手の大きさが特徴的な人も使いやすくすることが可能だと考える．なぜならば，ソフトウェアを利用して表示しているため，ソフトウェア側から調整すればどのような人でも打ちやすいキーボードが再現可能だからである．よって，本研究ではユーザ一人一人に合わせることが可能なソフトウェアキーボードを提案する．

3 提案手法

本手法ではブラインドタッチを行う際の，ユーザが認識しているキー配置にすることで問題解決を行う．そして，ユーザが認識している位置にキー配置することで，ユーザ一人一人に合わせたキーボードの作成を行う．

本章ではユーザ一人一人に合わせたキーボードを作成するための手順を論ずる．そして，提案した手法を用いたソフトウェアキーボードを，以降では Flexible キーボードと呼称する．

まず，ユーザが認識しているキー配置を調査しなければならない．そして，ユーザが認識しているキー配置に適応させることでキーボードの作成を行う．

3.1　ユーザの入力

本手法ではブラインドタッチする時に，ユーザが認識しているキー配置を調査しなければならない．

そこでユーザはブラインドタッチで文字入力を行う．そして，ユーザが入力した入力座標と入力文字を取得することで，ユーザが認識しているキー配置を調査する．

入力座標においては，タッチパネル上の座標を直接取得する場合，文字の出現頻度が疎なデータと密なデータとが混在する．出現頻度が異なるデータでは，出現頻度に依存した偏ったキーボードが作成される．

解決策として，タッチパネル上の座標を直接取得するのではなく，ユーザの入力を格子状に区分した平面にマッピングする．そして文字の入力頻度が密なデータを疎なデータに近づけさせる．

入力が可能な座標を減少させるために，タッチパネルを仮想的に格子状に区分する．格子状に区分したエリアを入力が可能な座標とする．

図 1 は同一位置に入力した場合，格子状に区分したエリアでの入力とタッチパネル上での入力座標を示している．格子状に区分したエリアでは入力が 2 になり，タッチパネル上では入力が 4 となる．タッチパネルを仮想的に格子状に区分することで，近い入力を一つにまとめることで，文字の入力頻度に関わらず疎なデータに近づけることが期待できる．

よって，格子状に区分したエリアを入力座標として，A から Z のキー分布図を作成する．図 2 はユーザが認識するキー分布図の例である．

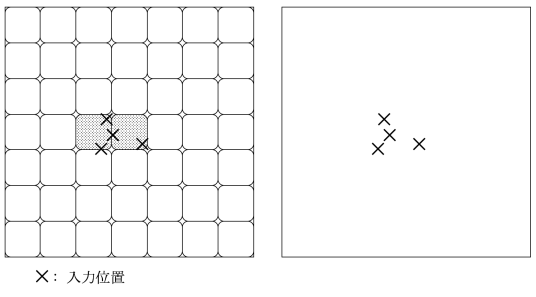


図 1　格子状に分割されたエリアでの入力座標 (左) とタッチパネル上での入力座標 (右)

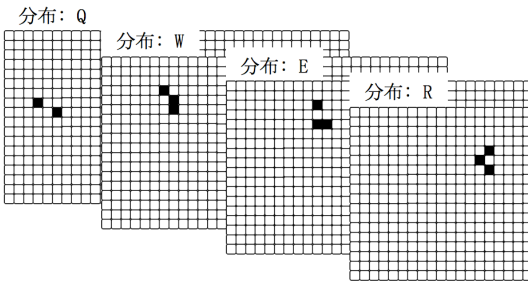


図 2　ユーザが認識するキー分布の例

3.2　入力データの加工

本手法では 3.1 章で述べた，ユーザが認識する配置（図 2）を元にキーボードの作成を行う．しかし，ユーザの入力データをそのまま適応した場合，歪なキーボードが作成される．図 3 の例では，QWERTY キーボードを前提とした配列を元に，Q から R までの入力データを反映させたものである．しかし，図 3 のキーボードは歪な形をしているため，ユーザは打ちづらいと感じる．

本手法では，ユーザの入力データを加工することで，歪さを改善させる．ユーザの入力位置を中心とした，一定の距離を一つの入力とすることで，キーボードの歪さを改善させる．図 4 は，一定の距離を一つの入力とし，入力データを反映させたものである．ユーザの入力データをそのまま適応させずに，加工した入力データを適応させる．入力範囲が広がることで，ユーザの入力に多少のブレが発生してもミスタイプ

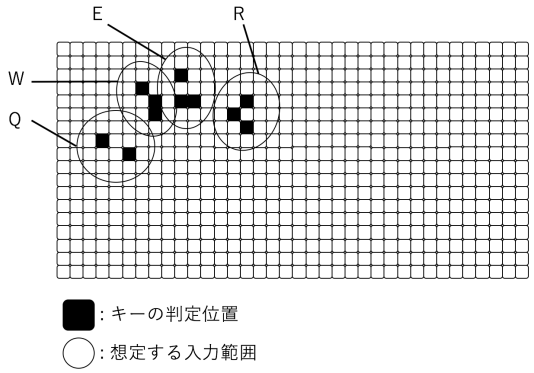


図 3　歪なキーボード

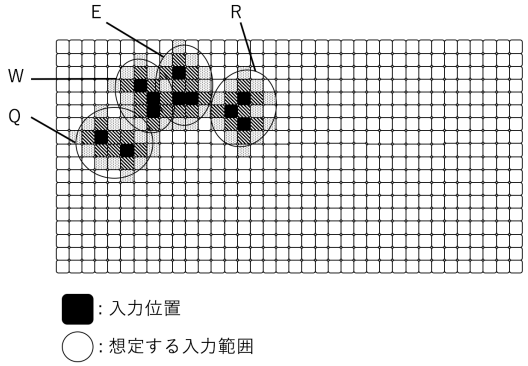


図 4　データ加工後の入力範囲

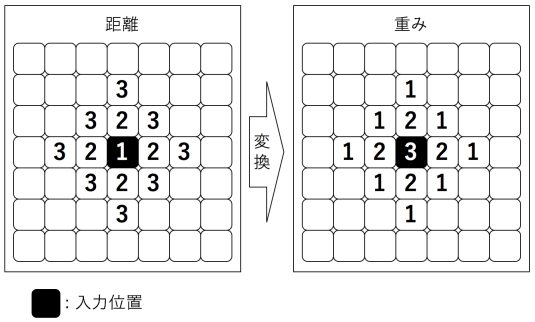


図 5　距離値から重みへの変換の例

ではなく正しい入力と認識できるようになる．したがってユーザは入力しやすいと感じる．

3.3　キー配置の決定

ユーザに合わせたキーボードを作成するために，格子状に分割されたエリアにどのキーが配置するかを決めなければならない．

本手法では，3.2 章で述べた距離の値を利用する．図 5 のように，距離値を重みの値へと変換させる．格子状に分割されたエリア毎に重みを比較し，A から Z キーとして割り当てる．

まず，それぞれのキーの入力データを加工し，キーボードに反映させる．その後，格子状に分割されたキーボード上に，入力データを反映させる（図 6）．本手法では，図 7 を用いてキーの重みを比較する．したがって，同座標上における，最大の重みを持つキーがその座標でのキーとなる．加工後の入力データを反映させ，重みを比較したキーボードが図 8 である．

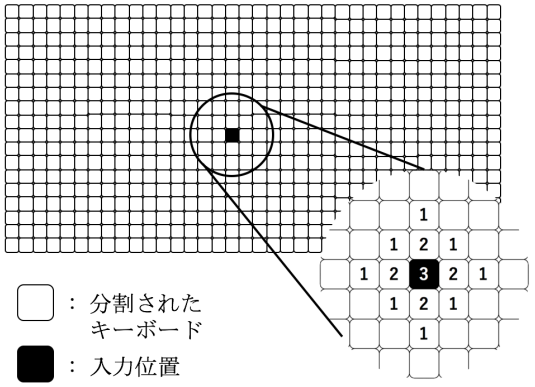


図 6　分割されたキーボードへの入力データ反映の例

$weight_{new}$　: 現在入力されている重みの値
$weight_{max}$　: 過去に入力された重みの最大値
$label_{new}$　: 現在入力されている文字
$label_{max}$　: 過去に入力された重みの最大値を持つ文字

If $weight_{new} \geq weight_{max}$ ***then***
$weight_{max} \Leftarrow weight_{new}$
$label_{max} \Leftarrow label_{new}$
end If

図 7　重みの比較方法

3.4　Flexible キーボードの作成

配置が決定し，A から Z までが反映されたキーボードが作成される．完成されたキーボードの作成例が図 9 で示す．

本アプリケーションは，テキストエリアとキーボードエリアに分かれている．テキストエリアは文字入力する対象の文字の列を表示する．ユーザには，入力した文字の色が変わることで入力されたことを知らせる．

キーボードエリアには，ユーザの入力データから作成されたキーボードを表示する．キーは色で示され，中心にあたられた文字が表記されている．ユーザは色がついた場所を触れることで，文字入力を行うことが可能である．

4 実験

本章では，提案したキーボードと従来のキーボードのミスタイプ数による実験方式と比較実験の結果，考察を述べる．

4.1 実験方式

本実験では，提案した Flexible キーボードと従来のキーボードとのミスタイプ数の比較実験を行う．従来のキーボードとして，Google Inc. が作成したソフトウェアキーボード (Google キーボード) のレイアウトを使用する．

条件として，キーボード配列は QWERTY 配列とし，A から Z のみが入力可能キーとする．A から Z 以外のキーが設置されているエリアは，空白とするが，触れても入力として扱わない．

また，ミスタイプ数のカウント方法は以下の図 10 で示す．図 10 で示す方式は連続してミスタイプした場合でも，ミスタイプ数が 1 であることを示している．なぜならば，本研究はブラインドタッチによる入力を前提としているため，ミスタイプであるとユーザが気づくまで入力することが多い．補正前である図 11 の方式では，ユーザが過去のミスタイプに気づくまで，現在の入力がミスタイプとして扱われるからである．

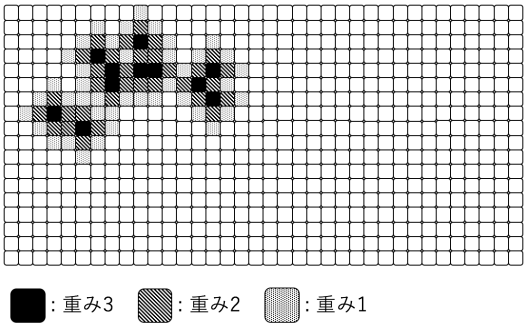


図 8　重みデータを反映したキーボード

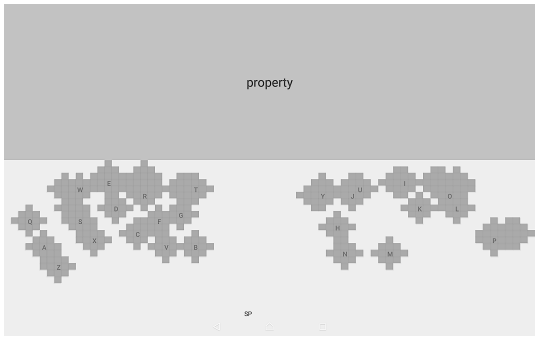


図 9　Flexible キーボードの作成例

表 1　入力平均文字数と入力単語数

入力平均文字数 (文字/回)	約 400
入力単語数 (単語/回)	25

$input_{Char}$:現在の入力文字
$target_{Char}$:意図する文字
$missCnt_{Integer}$:ミスタイプ数
$inputCurrent_{Boolean}$:現在の入力の真偽
$inputPast_{Boolean}$:一つ前の入力の真偽

If $input_{Char} = target_{Char}$ ***then***
　$inputCurrent_{Boolean} \Leftarrow True$
else If
　$inputCurrent_{Boolean} \Leftarrow False$
end If
If not $inputCurrent_{Boolean}$ ***then***
　If $inputPast_{Boolean}$ ***then***
　　$missCnt_{Integer} \Leftarrow missCnt_{Integer} + 1$
　end If
　$inputPast_{Boolean} \Leftarrow False$
else If
　$inputPast_{Boolean} \Leftarrow True$
end If

図 10　ミスタイプのカウント方法

入力する単語は，中学生で学ぶ英単語を用いる．100 単語選出し，実験を行うたびにランダムで表示される．被験者ひとりあたりの平均文字数と単語数は表 1 で示す．

キー入力を行うため，比較的大きいタブレット端末を使用しなければならない．今回使用されたタブレット端末は，SONY 製の Xperia Z2 Tablet である．Xperia Z2 tablet は，10 インチの画面を持つ，現状で入手できる最大の画面サイズである．

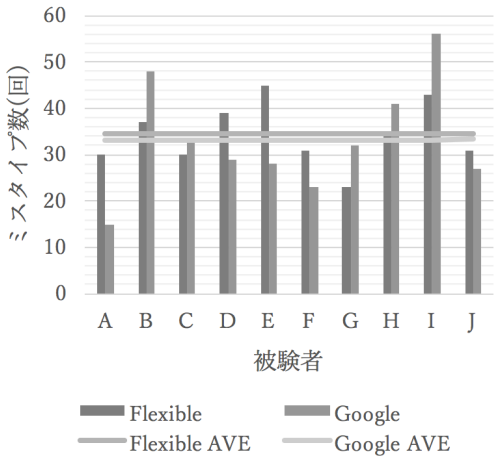
被験者は，理系の学部 4 年生の 10 人である．また，被験者はパソコンのキーボード入力に慣れており，ブラインドタッチが可能である．

input :現在の入力文字
target :意図する文字
missCnt :ミスタイプ数

If not *input* = *target* ***then***
    missCnt += 1
end If

図 11　補正前のミスタイプのカウント方法

表 2　ミスタイプ数の比較



4.2　実験結果

ミスタイプ数の比較結果は表 2 の通りである．提案した Flexible キーボードは，一般的な Google キーボードと比較しミスタイプの平均数は高い．また，ミスタイプ数が多いユーザは，Flexible キーボードと Google キーボードはそれぞれ 5 人であった．

一方，ミスタイプ数を元に，ミスタイプ数の合計と平均，分散を計算した結果が表 3 である．Flexible キーボードは Google キーボードよりミスタイプ数が多いことを示している．しかし，分散で比較した結果は，Flexible キーボードの方が低い値であった．

4.3　考察

実験の結果より，提案した Flexible キーボードは様々な人が使いやすさが等しいことが言えた．また，表 3 より，一般的である Google キーボードは，人によって使いやすさが異なっていることが言える．つまり，本手法によるキーボードの作成は，一人一人に合わせることに成功した．

表 3　キーボードごとのミスタイプ数の合計と平均，分散

	Flexible	Google
合計 (回)	344	332
平均 (回)	34.4	33.2
分散	45.2	146.6

しかし，ミスタイプ数の改善は見られなかった．原因は，新しい配置をしたキーボードに慣れていないからだと考えられる．慣れているキーボードと，慣れきっていないキーボードでは，同じ条件下での実験が不可能である．このような慣れの差が，ミスタイプの改善を阻害している一つの要因と考える．

5　今後の課題

今後の課題の一つとして，実験データの少なさが挙げられる．実験の結果から多角的に考察を行なわなければ，今後の改善点が明確にすることができない．解決するために実験の種類に，アンケートや文字ごとのミスタイプ数，ユーザごとにキーボードの形を保存する機能の実装を行わなければならない．

また，被験者から，キーボードの作成手順のわかりづらさと作成されたキーボードの反応速度の悪さが挙げられた．UI や処理などのソフトウェア面を見直すことが今後の課題である．

参 考 文 献

[1] 遠藤祐貴, 郷健太郎: タッチ画面におけるバブルカーソル状適応型キーボードの提案, 情報処理学会研究会報告ヒューマンコンピュータインタラクション, Vol. 117, No. 3(2006), pp. 24–32.
[2] 久野祐輝, 志築文太郎, 三末和男, 田中二郎: キーを指の設置位置とその周囲に配置するソフトウェアキーボードの開発, 情報処理学会論文誌, Vol. 55, No. 4(2014), pp. 1353–1364.
[3] 辛島光彦, 柳井勇馬, 芳賀崇文, 西口宏美: ソフトウェアキーボードの打鍵パフォーマンス向上のためのサポートツールの提案, Supplement, Vol. 50, Japan Ergonomics Society, 2014.