ユーザ適応型ソフトウェアキーボードに関する研究

２０１９年　３月

久野 祐輝

ユーザ適応型ソフトウェアキーボードに関する研究

久野 祐輝

システム情報工学研究科

筑波大学

２０１９年　３月

概要

本論文では、ソフトウェアキーボードにおけるタッチタイピングを実現するアプローチである、ユーザ適応型ソフトウェアキーボードに関する研究について述べる。

タッチタイピングとは元来タイプライターの物理キーボードにおける手元を注視しない入力手法であり、手元注視が不要なことにより、効率の良い入力を可能とした。

ソフトウェアキーボードにおいてタッチタイピングを実現することができれば、タッチパネル搭載端末においても効率的な文字入力を実現できる。特に、物理キーボードと同等の大きさのタブレット端末におけるタッチタイピングの実現が強く期待される。しかし、現状のソフトウェアキーボードではタッチタイピングが困難である。ソフトウェアキーボードは物理キーボードと異なり、入力した際の物理的なフィードバックが存在しない。そのため、ユーザは物理キーボードのように手元注視を行わずにキーに指を合わせることが困難になる。

本研究の目的はタブレット端末向けのソフトウェアキーボードにおけるタッチタイピング、すなわち手元を注視しない入力手法の実現である。そのアプローチは、ソフトウェアキーボードのキーの位置をユーザの指の位置に合わせることである。本研究ではこのようなソフトウェアキーボードのことを適応型ソフトウェアキーボードと呼ぶこととした。

本研究では最初の適応型ソフトウェアキーボードのプロトタイプとして “Leyboard” を示した。Leyboard はソフトウェアキーボードのキーをユーザの指の位置とその周辺に配置することによって入力を容易にする。Leyboard の長期入力実験は既存のソフトウェアキーボードに対して有意に高速な入力が可能であることを示した。しかし、手元注視をなくしてタッチタイピングを実現する目的は達成されなかった。

Leyboard において手元注視が必要となった要因として、キーボードの上段、中段、下段としてユーザが認識する領域、すなわちタッチ入力の範囲が重なり合っていることが判明した。そこで、重なりを抑えるために QWERTY 配列キーボードの中段のキーを省略した Meyboard を設計、実装した。本研究ではこのようにキーの省略を行ったキーボードを減数キーボードと呼ぶ。Meyboard はキーの幅をユーザの指の位置に合わせる、適応型減数ソフトウェアキーボードであった。Meyboard では上下段のキーのいずれかにおけるフリック入力を中段のキーの入力と解釈することによって、省略された中段のキーの入力を補っている。実験により、Meyboard において手元注視の可不可は入力率に有意差を与えないことが判明した。これはタッチタイピング実現につながるが、一方で手元注視を行わないときにエラー率が高くなることも判明した。

エラー率を抑える方策として、ソフトウェアキーボードのキーの位置をユーザの指の位置に合わせる機能を入力時に逐次自動的に行う自動適応型減数ソフトウェアキーボードとして、Meyboard Ⅱ を示した。Meyboard Ⅱ では Space、Enter、Shift など文字入力において頻繁に入力することとなるキーの入力時にキーの位置をユーザの指の位置に合わせている。実験の結果、Meyboard Ⅱ は Meyboard よりもエラー数が抑えられており、習熟によってタッチタイピングを実現可能であることを示した。

目 次

図 目 次

表 目 次

第1章　序論

1.1　本研究の背景

タイプライターの発明以降、物理キーボードを用いた文字入力手法が様々な機器において採用され、人々の文字入力にかける時間とコストの削減に貢献してきた。物理キーボードは文字、記号等が印字されたキーと呼ばれるボタンが多数設置された入力機器である。物理キーボードを用いた文字入力においてはタッチタイピングという入力手法が考案され、主流となっている。

物理キーボードを用いた世界初の民生タイプライターは “Sholes-Glidden” であるとされる [Yam80]。この物理キーボードの配列を基とした、今日に至るまで様々な機器の物理キーボードにおいて使用される配列が QWERTY 配列である。QWERTY 配列は文字入力用途に最適化されたキー配列ではなく、アルファベット順が基となっている。その後、QWERTY 配列よりも高速な入力を意図して様々な配列が考案されたが [Yam80]、現在のデファクトスタンダードは QWERTY 配列となっている。

その結果、QWERTY 配列はハードウェアとしての物理キーボード以外に、ソフトウェアの UI としても用いられることがある。この時、ユーザは画面に表示された QWERTY 配列のキーの並びの中から、入力する文字を選択する。選択する方法はマウス等のポインティングデバイス、または上下左右の入力が可能な機器（複数個のボタン、ジョイスティック等）が考えられる。近年はタッチパネルを搭載した端末が存在し、ユーザが画面に直接触れることにより入力が可能となっている。タッチパネル搭載端末では、ユーザが UI のボタンとしてのキーに触れることにより入力を行う、ソフトウェアキーボードが用いられることが多い。本研究ではこのソフトウェアキーボードに着目する。

1.1.1　タッチタイピング

タッチタイピングは Yamada の定義 [Yam80] によると、10 本の指全てを用いてキーボードを基本的に見ずにタイプライターを操作することを指す。留意すべき点は、基本的に見ないという表現である。Yamada の定義ではタッチタイピングは完全アイズフリー入力のように手元を全く見ないで入力を行うものではなく、手元を多少見ることを許容している。これは手元を注視せずに入力を行うことを意味する。手元を注視しないことにより、ユーザは入力する原稿あるいは入力結果が表示される媒体（紙、ディスプレイ）に視線を集中することが可能となる。よって、手元への視線移動が低減され、高速な入力を実現できる。

タッチタイピングでは、ユーザはホームキーと呼ばれる特定のキー（QWERTY 配列の場合は a、s、d、f、j、k、l、セミコロン）に指を置いた状態を基本として、ホームキーの位置から指を上下（人差し指に関しては左右、斜め方向も含む）に動かすことによりキーに指を移動させ、入力を行う。ホームキーの位置を触って確認し、その位置に指を合わせることができれば、ユーザは記憶した筋肉の動き（motor memory）に基づいて各キーの入力が可能である [ZK07]。

1.1.2 QWERTY 配列のソフトウェアキーボード

近年はタッチパネルを搭載した端末において、ユーザが画面に触れることによって文字入力を行うソフトウェアキーボードが存在する。これらの端末では、ソフトウェアキーボードが物理キーボードの代わりに用いられている。QWERTY 配列が物理キーボードのデファクトスタンダードであるため、ユーザの慣れからソフトウェアキーボードにおいてもアルファベットの入力に関しては QWERTY 配列を採用したものが多い。

特に、画面の大きさが物理キーボード以上であるタブレット型のコンピュータ（以降タブレット端末とする）ならばキーの大きさを物理キーボードと同等にすることによって、物理キーボードと同じような入力が可能となるように思われる。ソフトウェアキーボードに物理キーボードの優れた入力手法であるタッチタイピングを適用できれば、優れた入力性能を発揮すると考えられる。

しかしながら、タブレット端末におけるタッチタイピングは困難である。 ソフトウェアキーボードのキーはタッチパネルの平面上に表示された画像であるため、それらの位置をユーザが把握するためにはキーボードの注視が必要となる。仮に注視なしにて入力を行った場合、ユーザはキーの位置を把握できていないため誤入力を頻発させることとなる。

1.2 本研究の目的とアプローチ

Yamada によれば、タイプライターは発明当初、左右の手の指それぞれ 1 本から 3 本程を用いて入力を行うことが主流であった [Yam80]。そして、タッチタイピングは上記よりも高速かつ疲れにくい入力を実現するために考案された。 Yamada によるタッチタイピングの定義には、手元を注視しないことのほかに、タイプライターを対象とすることと、両手 10 本の指を用いることが含まれていた。しかし、タッチタイピングの手法は文字入力の機器がタイプライターからコンピュータに変わった現代においても使われており、タッチタイピングをタイプライターに限定する定義は現実と乖離している。また、両手 10 本の指を利用することは QWERTY 配列の物理キーボードにおいて手元を注視しないための手段であり、タッチタイピングの本質とは考えにくい。よって、本研究ではタッチタイピングを単に手元を注視しない文字入力手法と定める。先に述べたようにタブレット端末のソフトウェアキーボードにおいてタッチタイピングの実現が期待されるため、本研究の目的はタブレット端末向けのソフト

ウェアキーボードにおけるタッチタイピングの実現とする。本研究にて想定しているタブレット端末の大きさは横幅 250mm 以上である。

なお、タッチタイピングによる入力を可能とする文字としてアルファベットを対象とし、数字、記号は対象とはしない。一般的に文章の入力においてはアルファベットの出現頻度が数字、記号よりも高いためタッチタイピングによるアルファベットの入力が可能となれば入力性能の向上が期待されるためである。

ソフトウェアキーボードのキーの配列に関しては、もっともよく知られた配列であるという観点から本研究では QWERTY 配列を基とする。本研究の目的はソフトウェアキーボードにおけるタッチタイピングの実現であり、キー配列が QWERTY 配列であることはタッチタイピング実現の要件ではない。一方で、QWERTY 配列とは全く異なった配列を採用すると、それを覚えるための負担がユーザに生じる。キー配列を覚える負担が本研究の実験結果に影響することを抑えるためには、多くのユーザが理解している配列を基とすることが望ましい。QWERTY 配列はこれを満たす配列であると考えられる。よって、本研究において想定されるユーザは QWERTY 配列の物理キーボードのタッチタイピングが可能であり、QWERTY 配列のキーの並びを理解しているものとする。

タッチタイピングを実現するソフトウェアキーボードのキー配列は 2 通り考えられる。1 つは既存のソフトウェアキーボードの形状およびキーの配列を維持したもの、もう 1 つはユーザが意図したキーを入力しやすいように形状およびキーの配列を変更したものである。 前者はそのままではユーザの入力誤りを頻発させるため、入力誤りへの対処を必要とする。これはソフトウェアキーボード側がユーザの入力を予測し、それに合わせてユーザの入力内容を修正することによって実現する。修正は辞書と入力内容を照らし合わせることによって行われることが多いが、ゆえに辞書に存在しない文字列の入力が困難になる。本研究では、ユーザが任意の文字列を入力可能とすることを要件とし、後者のアプローチを行うこととする。

本研究ではユーザが意図したキーを入力しやすくするため、キーの位置を個々のユーザの指の位置に合わせている。このようなソフトウェアキーボードを、本研究ではユーザ適応型ソフトウェアキーボード、あるいは適応型ソフトウェアキーボードと呼ぶこととする。

1.3 本研究の成果

本研究の成果は以下の通りである。

適応型ソフトウェアキーボード Leyboard

キーの位置をユーザの指の位置に合わせる適応型ソフトウェアキーボードとして Leyboard を実装し、これが既存のソフトウェアキーボードを超える入力性能を発揮する可能性を示した。

適応型減数ソフトウェアキーボード Meyboard

Leyboard では手元注視が必要であったためにタッチタイピングが困難であることが判明した。解決策として、キーボードのキーを段単位にて省略し、縦方向のキーの数を減らして指の移動量を抑えることを考えた。モバイル機器向けに物理キーボードを小型化するため、QWERTY 配列のキーの一部を省略した reduced keyboard（本研究では和名として減数キーボードと呼ぶこととする）が存在する [GKFS04, GBAT99]。本研究ではタッチタイピングを実現するために減数キーボードに着目し、適応型減数ソフトウェアキーボード Meyboard を実装した。そして、Meyboard が手元を隠したアイズフリーの状態にて入力できる可能性を示した。

自動適応型減数ソフトウェアキーボード Meyboard II

Meyboard はアイズフリー入力時にエラー数が多くなることが判明した。その原因は入力が進むにつれて生じる、ユーザの指の当初の位置からのずれであった。ずれに対応するため、適応型ソフトウェアキーボードにおけるキーの位置をユーザの指の位置に合わせる機能を逐次用いるようにした。すなわち適応機能の自動化を行った。このような自動適応型減数ソフトウェアキーボードとして Meyboard II を実装した。Meyboard II は Meyboard 同様にアイズフリーの状態にて入力が可能であり、かつ Meyboard よりもエラー数を抑えられており、自動適応機能の有用性を示した。

1.4 本論文の構成

第 2 章では、既存の文字入力に関する研究を挙げ、Leyboard 及び Meyboard とそれらを比較する。第 3 章では、最初に実装した適応型ソフトウェアキーボードのプロトタイプである Leyboard に関して、その機能と実装を説明し、入力性能を調査するために行った実験の内容と結果を記述する。実験の結果、Leyboard は既存のソフトウェアキーボードを超える入力性能を発揮したことにより適応型ソフトウェアキーボードの一定の効果を示した。しかしながら、Leyboard では手元注視が必要であったためにタッチタイピングが困難であることが判明した。その理由に関する議論と改善策に関しても本章にて議論する。第 4 章では第 3 章における議論を受けて実装した、適応型減数ソフトウェアキーボード Meyboard の機能と実装を説明し、タッチタイピングよりも厳しい条件であるアイズフリーの入力性能を調査した実験の内容と結果を記述する。第 5 章では Meyboard のアイズフリー入力においてエラーが頻発したことから、エラー数を抑えるために自動適応型とした自動適応型減数ソフトウェアキーボードである Meyboard II を実装し、その機能と実装を説明する。また、第 4 章同様にアイズフリーの入力性能を調査した実験の内容と結果を記述し、自動適応型としたことの効果に関して議論する。最後に第 6 章にて本研究をまとめ、結びとする。

第2章　関連研究

2.1　ユーザ適応型ソフトウェアキーボード

適応型ソフトウェアキーボードの先行研究として、LiquidKeyboard [SLL11]、CATKey [GE07]、Personalized input [FW12]、Gunawardana らのキーボード [GPM10]、Go らのキーボード [GT10] が挙げられる。これらのキーボードではアルファベットを入力可能であるが、小文字及び一部の記号のみが入力可能であった。本研究では上記に加えて大文字を入力可能にするために Shift キーの入力が可能なようにキーボードを設計した。また、これらのキーボードでは、キーの位置と大きさを一度ユーザに合わせると、その配置を固定していた。配置の変更が可能なもの（[SLL11]）も存在するが、ユーザが自発的に変更しない限り変更は行われない。本研究では最終的にキーの位置と大きさがユーザに逐次適応し続ける自動適応型ソフトウェアキーボードを作成している。

2.2　ユーザ適応型物理キーボード

キーの位置をユーザに合わせた物理キーボードに TRON キーボード [坂村 86] が存在する。よって、TRON キーボードはユーザ適応型物理キーボードといえる。そのキー配置は直線状ではなく、人間の手の形に合わせて湾曲している。TRON キーボードの英字の配列は QWERTY ではなく Dvorak に従っている。日本語入力に関しては日本語の特性に応じて独自に論理的に定めた配列が適用される。Leyboard もキャリブレーションにより、ユーザの手に合わせてキーが湾曲する配列となる。本研究ではユーザの習熟のしやすさの観点から配列は QWERTY を基にした。

2.3　減数キーボード

減数キーボードをソフトウェアキーボードに適用し、キーの一部を省略したキーボードとして 1Line keyboard [LGYT11] と Gestyboard 2.0 [CWA+13] が挙げられる。Space などを無視してアルファベットに限定すれば、これらは共に 1 行のソフトウェアキーボードであり、QWERTY 配列の各列の文字（人差し指が入力する文字に限り 2 列分）をまとめて 1 つのキーに統合した配列である。但し、キーの数に対して足りない文字数を入力する手法はそれぞれ異なる。

1Line keyboard では、キーの入力時点では文字を区別せず、ユーザは文字がまとまったまま入力を行う。ユーザの入力後に辞書を用いて、まとまりから単語を推測し、ユーザが単語を確定することにより、単語が入力される。

Gestyboard 2.0 ではキーの入力方法によって文字を区別する。これはタップまたはフリックであり、後者に関してはその方向も文字の区別に利用する。 具体的には、Gestyboard 2.0 は 8 個のキーを持ち、タップにて 8 種類のホームキーの文字を、フリックにて 22 種類の非ホームキーの文字を入力する。

本研究とこれらのキーボードはキーの数に加えて、1Line keyboard は辞書を用いる点が、Gestyboard は文字に対する入力方法が異なる。

なお、1Line keyboard の目的はタブレット端末におけるキーボードの表示領域を小さくし、情報をユーザに表示する領域を大きくすることであり、Gestyboard 2.0 の目的は本研究同様にタッチタイピングを実現することである。

2.4 ジェスチャに基づく入力手法

SHARK [ZK03] を起源としたジェスチャ入力はタッチタイピングを実現し得る手法として有用な手法の一つであると考えられる。ジェスチャ入力は、キーボード上のペンまたは指の動きを単語に変換する入力手法であり、変換に統計処理を用いている。ユーザの入力した 1 ストローク分の動きをユーザが入力を意図した単語としてもっともらしいものに変換する際に、一つの候補に定まらないことがあり、単語を最終的に確定するための選択 UI とともに、単語の候補群をユーザに提示している。

Gordon らは WatchWriter というスマートウォッチ上におけるジェスチャ入力を可能にしたソフトウェアキーボードにおいてユーザが 22–24wpm かつエラー率がほぼゼロの入力を実現したことを報告している [GOZ16]。スマートウォッチは画面の小ささからユーザの指によって画面の大部分がふさがれるためにキーボードが見えにくい。よって、ユーザはタッチタイピングに近い状態において、高速かつ正確な入力を行っている。

単語の選択 UI の表示位置と入力した文字の位置が同じであり、WatchWriter のようにユーザがキーボードを見ずに入力が可能ならば、ジェスチャ入力においてタッチタイピングが成立する。Fleksy [Thi] と BlindType [LYY+17] は単語の選択 UI の位置を文字の位置と同じにして、キーボードを非表示にすることによりタッチタイピングを実現した。これらのキーボードでは、ユーザが水平方向にフリックをした際に、ユーザが入力中の文章上にて単語への変換が行われ、フリックを繰り返すと他の候補の単語に表示が切り替わっていく。よって単語の選択 UI の位置を文字の位置と同じにすることが可能であり、非表示のキーボードと合わせてタッチタイピングを実現している。

但し、ジェスチャ入力は変換に辞書を用いるため、辞書に登録されていない単語の入力が困難であるという限界が存在する。本研究では言語モデルに依存しないため、任意の単語を入力可能である。

2.5 一筆書きによる入力手法

先に挙げたジェスチャ入力とは異なり、1 つの文字を単一のストローク、すなわち一筆書きにて表現する、元来はスタイラス向けの文字入力手法として、Graffiti [MZ97]、Unistrokes [GR93]、EdgeWrite [WMK03] が挙げられる。ストロークはアルファベットを模した単純なものであるため、ユーザは容易にその形状を学習し、タッチタイピング同様に入力部分を見ずにその形状を描くことが可能である。一方で、これらの研究は単一のストロークにて一文字を表現するため、入力に時間を要する。例えば、[MZ97] では、Graffiti の入力速度が 11.4wpm、Unistrokes が 15.8wpm であった。本研究では 20 から 30 種類の文字をタップにて入力するため、すべての文字がストロークによる入力であるこれらの研究よりも入力に時間がかからない。

2.6 視覚障害者向けの入力手法

視覚障害者向けにソフトウェアキーボードにおけるアイズフリー入力を実現した研究として、BrailleType [OGN+11b]、BrailleTouch [SCF+12]、Perkinput [AWPL12] が存在する。これらのキーボードでは文字に応じた点字のパターンを入力することにより、アイズフリー入力を実現している。これらのキーボードは点字の知識を持つユーザにとっては有用であるが、本研究では知識に乏しいユーザに対して他の入力手法も必要と考え、ユーザが慣れていると思われる QWERTY 配列に基づいたキー配置とした。

2.7 ウェアラブル機器向けの入力手法

近年は、ヘッドマウントディスプレイのようなウェアラブル機器使用時にタッチタイピングを実現する研究が行われている。その中には我々と同様に QWERTY 配列に基づいたキーボードを用いた研究が存在する。Digitouch [WJJ+17] はデータグローブを利用して、ユーザの両手の人差し指から小指をキーボードのキーとして用いている。Swipezone [GCF15] は Google Glass のタッチパッドを利用してキーの選択を可能にしている。この手法はスマートウォッチ向けソフトウェアキーボードである Swipeboard [CGF14] に基づいている。これらのキーボードではユーザがタッチタイピングできるように、指を入力面（データグローブまたはタッチパッド）にて滑らせることにより入力するキーを選ぶことが可能である。本研究の対象はタブレット向けのソフトウェアキーボードであり、キーの位置を指の位置に合わせることにより、タッチタイピングが可能である。

2.8 手元注視を必要としない文字入力に関する研究

タッチパネル上における文字入力に関するものではないが、ソフトウェアキーボードと同様に物理キーボードが存在しない入力対象において文字入力を行う研究が存在する。これら

の研究では手元にキーが表示されることはないので、ユーザは手元注視に頼らない入力を求められる。

Mujibiya らの研究 [MMR10] ではタッチパネルではない机等の平面上における指の動きをカメラで計測する。入力面はプロジェクタによって出力される赤、緑、青の縞模様によって構成される。ユーザはこの入力面上において入力を行う。カメラを用いて奥行きを計測することによって、入力面上における指先の動きと入力面への接触の有無を把握する。このようにして、入力するキーを決定している。Goldstein らの研究 [GBAT99] では、データグローブを用いて文字入力を行っている。データグローブは指に固定された圧力センサによってユーザが入力した指を判断するものである。Guesso [藤田 09] では、ユーザの指の先端にスイッチを取り付けて文字入力を行う。上記 2 つの研究において、それぞれのシステムが知ることができるのはユーザが入力を行った指に関しての情報のみである。QWERTY 配列の文字入力において、各指は担当するキーが決まっていることを利用して、システムは入力された指の順番から字句解析を行い、入力された単語を推定する。ユーザはシステムが推測によって作った候補群の中から入力を確定する。

上記の研究ではユーザは QWERTY 配列に従った指の動かし方によって単語の入力を行う。本研究においても、ユーザは QWERTY 配列に沿った文字入力を行う。但し、本研究では字句解析を行わず、ユーザがタッチパネルの画面上に指を置いた位置から入力するキーを決定する。そのため、アルファベットにて表現可能な範囲においては、言語に依存しない入力が可能である。

FingeRing [FS94] もユーザの指の動きを検知し、入力を行う。FingeRing はユーザの各指の付け根に指輪型のセンサを装着して指の入力の検知を行う。但し、FingeRing は QWERTY 配列に沿った文字入力手法ではなく、ユーザの各指の置き方のパターンを文字に割り当てている。

2.9 QWERTY 配列のキーボード以外の文字入力に関する研究

QWERTY 配列の物理キーボードの他にも、様々な文字入力手法が考案されている。本研究では QWERTY ベースのソフトウェアキーボードを対象としたが、ここで挙げる入力手法をソフトウェアキーボード上において再現することも手元注視を減らすための手段として考えられる。

Proschowsky [PSJ06] らや、Slay ら [SFM08] はホイールの操作による文字入力を考案した。但し、前者はプロトタイプの実装をタッチパッド上にて行っている。TwoStick [KIG07] はビデオゲームに用いられるコントローラに搭載された 2 つのジョイスティックを利用して文字入力を行う。CLURD [JK09] はハングル文字と英字の文字入力の研究である。CLURD は 5-way key という中央ボタン付きの矢印キーを利用し、ボタンを押す回数、順番、時間の組み合わせにより入力文字を決定する。ユーザはこの入力方法を覚えなければならないため、学習難易度は高いと思われる。これらの研究はボタンを押す順番や指の動きのパターンを文字に割り当てることによって入力を行う。この点はスワイプやフリック入力を用いて入力を行う本研究と類似する。

Hiraga らはキーボードの漢字無連想変換方式として T-CODE を考案した [HOY80]。日本語入力において現在一般的に用いられている変換は連想式であり、「トキ」と「時」のようにキーの組み合わせと文字の間に連想がある。一方、無連想式はこの連想関係がない。端的に言えば数ストローク (T-CODE では 2) 分のキー入力を 1 つの文字に割り当てることにより無連想を実現している。連想式は習熟が容易であるが、変換に思考を割くために一定以上入力速度が上がらなくなる。無連想方式は変換の際の思考が不要なので人間やハードウェアの限界を無視すれば入力速度は無限に向上する。但し全ての文字 (かな漢字数字) に対して入力方法を覚えなければならないので学習難易度は高い。本研究にて対象としたソフトウェアキーボードは QWERTY ベースであり、入力速度よりもユーザの習熟の容易さを重視している。

2.10　タップによって入力を行うソフトウェアキーボード

本研究では大半のキーの入力をタップにて行う。TapBoard [KSL+13] は本研究と同様にタップにて入力を行うソフトウェアキーボードである。TapBoard は物理キーボードと同様に、ユーザがキーの上に指を置いた状態からの入力を行えるようにした。その方法はキーの入力をタップによって行うこととして、指を置いた状態と入力を区別することであった。そのため、両者を区別するためのタッチ時間の閾値を設けている。本研究においてキーの入力をタップによって行うのは、キャリブレーションを行うために指を置いた状態とキーの入力を区別するためである。

2.11　フリックによって入力を行うソフトウェアキーボード

フリック入力を QWERTY 配列に取り入れた研究としては Gestyboard [CAP+12] が挙げられる。Gestyboard はタップの場合中段のキー、フリックの場合該当方向のキーの入力となる。このため、入力するキーに応じてフリック方向を使い分ける必要がある。Meyboard は Gestyboard のように入力キーによってフリック方向を使い分ける必要はないので、フリック方向に特に制約は設けない。そのため、習熟はより簡単であると思われる。また、Blossom [桜井 13] では QWERTY 配列のソフトウェアキーボードにおいてフリック入力を利用して日本語入力を行っている。ユーザは子音となるアルファベットのキーの上から、母音を決定する 5 方向のフリック入力を使い分けることにより日本語入力を行う。Meyboard ではフリック入力は中段のキーと大文字の入力に用いられ、ユーザが使い分けるフリック方向は少ない。

2.12　bimanual な操作を行うソフトウェアキーボード

Don らは bimanual な操作をソフトウェアキーボードに取り入れた [DS10]。ユーザはまずキーボード上の大まかな範囲を片手で選択する。するとユーザのもう片方の手の位置にその範囲がズーム表示されるのでユーザはそこから入力するキーを確定する。Bimanual Gesture Keyboard [BCO+12] も bimanual な操作によって入力を行うソフトウェアキーボードである。

そのキー配列は QWERTY 配列を格子状に並べ直し、さらに左右の 2 つに分割したものになっている。ユーザの左親指の位置に、分割されたキーボードの左部分が、右親指の位置に、分割されたキーボードの右部分が表示される。ユーザは入力する単語に含まれるアルファベットのキーを通るように、親指をスワイプさせることによって入力を行う。本研究の Leyboard では、キーセットを切り替えてから入力を行う際に bimanual な操作となることがあり、類似性が存在する。また、Meyboard は Bimanual Gesture Keyboard のように QWERTY 配列を格子状に並べ直したキー配列を持つ。

2.13 文字入力手法の入力性能を比較した研究

Ko らは巨大なタッチパネル端末であるテーブルトップにおける種々の文字入力手法の入力性能について調査した [KKKE11]。結果、エラー率にはさほど差はないが、QWERTY 配列が最も速度に優れていた。これは QWERTY 配列を基にする本研究のデザインの妥当性を補強する。

Oliveira らは盲人の QWERTY、MultiTap、NavTouch、BrailleType それぞれのソフトウェアキーボードにおける入力を評価した [OGN+11a]。但し、ここにおける QWERTY は split-tapping [KBW08] または double tap を用いている。split-tapping は盲人がソフトウェアキーボードにおいて入力を行う手法である。ユーザは人差し指のスワイプによって入力対象を選択し、中指のタップによって入力対象を確定する。double tap は人差し指のスワイプによって入力対象を選択し、そのタップを 2 回行うによって入力対象を確定する手法である。MultiTap は 12 のキーにアルファベットを複数ずつ割り当て、その入力回数から入力文字を確定する。すなわち、MultiTap はテンキーを搭載した携帯電話における入力を再現している。NavTouch はアルファベットをアルファベット順に格子状に並べ、縦方向と横方向のスワイプ入力を組み合わせることによって入力対象を選択、決定する手法である。Oliveira らが行った評価の結果、入力速度が速い順に QWERTY、MultiTap、NavTouch、BrailleType となり、エラー率の低い順に BrailleType、NavTouch、QWERTY、MultiTap となった。ここから QWERTY 配列は入力速度に優れるが、画面注視が不可能である場合、他のアイズフリー入力手法と比較してエラー率が高くなることが分かる。よって、QWERTY ベースのソフトウェアキーボードの手元注視を減らすためには何かしらの工夫が必要である。Oliveira らの評価結果は手元注視を減らすための工夫を行う本研究の妥当性を補強する。

第3章　適応型ソフトウェアキーボード

キーを指の設置位置とその周囲に配置する、適応型ソフトウェアキーボードのプロトタイプとして Leyboard を開発した [KST12, KST13]。本章ではこのプロトタイプの設計と機能に関して述べる。

Leyboard のキーはタッチパネル上のユーザの指の設置位置と、QWERTY 配列を基に配置される。ユーザの各指の位置には a、s、d、f、Space、Enter、j、k、l、セミコロン（;）キー（ホームキー）が配置される。非ホームキーはホームキーの周囲に配置される。

各キーの形状はボロノイ図を描画することにより決定される。

Leyboard を開発する言語として C#を用いた。Leyboard は.NET Framework 4/WPF4 の API を利用して、マルチタッチ対応 WPF アプリケーションとして実装した。

3.1　キャリブレーション

本研究では Leyboard のキー配置を決定する行程をキャリブレーションと呼ぶ。

3.1.1　キャリブレーションの方法

Leyboard は、ユーザが両手全ての指（10 本の指）をタッチパネル面に置いた時に、各指の位置にホームキーを配置し、非ホームキーをその周囲に配置する。ユーザが画面上にすべての指を置いている限り、キーの位置は確定しない。この時ユーザが指を動かすとそれに合わせてキーの位置も移動する。ユーザがいずれかの指をタッチパネル面から離し、ユーザが置いた指の数が 10 本未満になるとキーの位置が確定し、キャリブレーションが終了する。

キャリブレーションを行うことによって、キーの位置がユーザの指の位置に合うようなキーの配置が作られる。すると意図したキーが入力しやすくなると考えられる。

3.1.2　キャリブレーションとキー入力の衝突回避

Leyboard においてはユーザが 10 本の指を置いた際に必ずキャリブレーションを行うとして、このとき入力をできなくした。入力をできなくしたことによって、キャリブレーション時に誤ってキー入力が行われないようにしている。但し、入力をできなくしたことによって、関連研究の TapBoard のような指を置いた状態からの入力も不可能になった。ゆえに、Leyboard では少なくともいずれかの指が浮いた状態においてユーザは入力を行う。厳密にはユーザが

画面に置く指の数が 8 点を超えた場合に入力が行えなくなるようになっている。そして、ユーザが 10 本の指を置く度にキャリブレーションが行われる。

3.1.3 各タッチと指の対応

Leyboard は、ユーザが両手全ての指（10 本の指）をタッチパネル面に置いた時に、各タッチと指の対応をとる。各タッチ点がどの指によるものであるかは以下のように判定する。まず画面下部に最も近いタッチ点 2 つは親指によるものとみなす。それ以外の 8 つのタッチ点を幅方向の座標値によってソートする。ソートされたタッチ点に対して、座標値の昇順に左手小指、左手薬指、左手中指、左手人差し指、右手人差し指、右手中指、右手薬指、右手小指を割り当てていく。

3.1.4 キーの配置規則

各キーの具体的な配置規則を以下と図 3.1 に示す。キャリブレーション時の人差し指と薬指の位置を結ぶ線分を考える。次にホームキーの座標を通りかつ先の線分に平行な直線を各指において考える。その直線においてホームキーの位置から右（左手ならば薬指から人差し指への、右手ならばその逆方向）に 17mm 離れた位置を基準とする。親指の場合は担当するキーの数が多いので、誤入力を避けるために距離を他の指よりも大きめにとって 23mm とした。17mm、23mm という数値は共に経験的に求めた。非ホームキーはホームキーの位置を中心にその位置から一定の角度だけ回転した位置に配置される。以下に具体的な角度を述べる。中指、薬指によって入力されるキーについては 90（w、e、i、o）、270（x、c、コンマ、ピリオド）度である。人差し指については 0（g）、60（t、u）、120（r、y）、180（h）、240（v、n）、300（b、m）度である。小指については 90（q、p）、225（スラッシュ）、315（z）度である。親指については 0（全角半角切り替えキー、右矢印）、45（Delete、右 Shift）、90（左 Alt、上矢印）、135（左 Ctrl、Back Space）、180（Num、左矢印）、225（左 Shift、Alt）、270（Fn、下矢印）、315（Tab、右 Ctrl）度である。ここでユーザの各指によって入力されるキーをそれぞれグループにまとめたとすると、各グループ（例: q、a、z）において非ホームキー（例: q、z）はホームキー（例: a）の位置を基準として同一円周上に配置されることになる。よって、各指で入力されるキーのグループにおいてホームキーから非ホームキーまでの距離は全て等しくなる。一方、従来のソフトウェアキーボードではこの距離は等しくなかった。このように距離を等しくすることにより、指の移動量が抑えられるので非ホームキーの入力はより容易になると考えられる。

3.2 各キーの入力方法

Leyboard のキー入力はタップにて行う仕様になっている。タップとしたのはキーの位置を決定するキャリブレーションと、キーの入力を衝突させないためである。

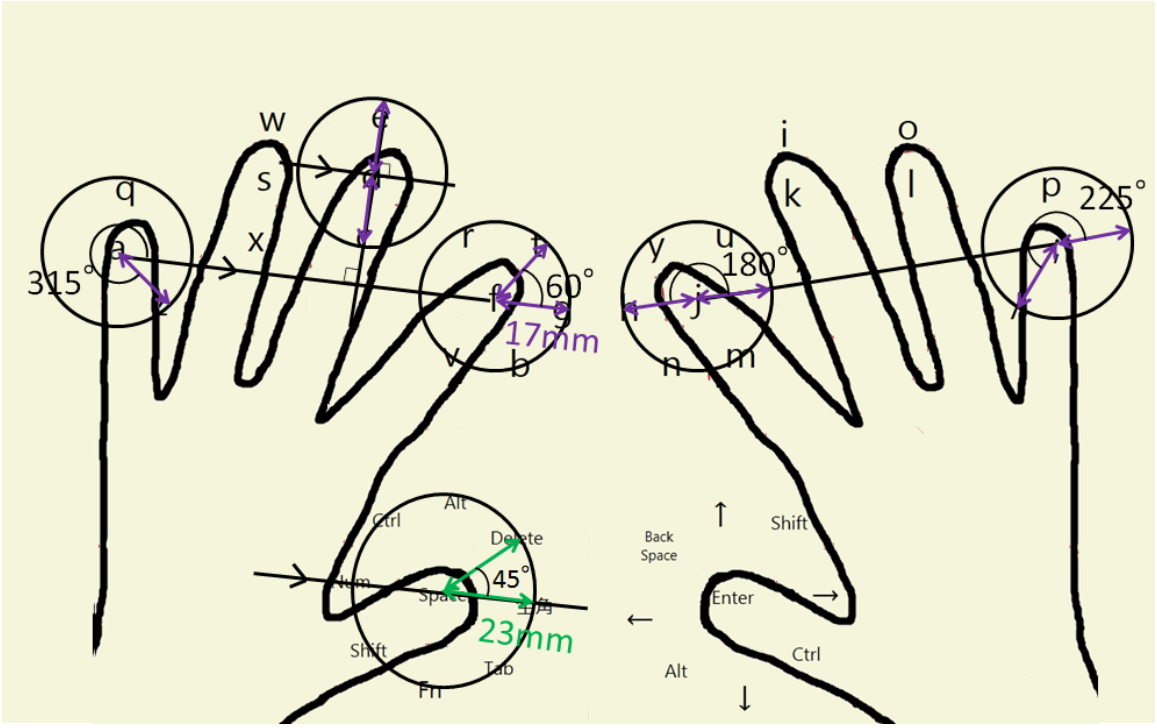

図 3.1: キーの配置規則

ユーザが指をタッチパネル面に接触させてから離すまでの時間で判定を行い、その時間が短かった時をタップ入力とみなす。タップ入力であるかを判定する時間の閾値は経験的に設定した。その時間は平均的には 140 ミリ秒であった。固定値になっていないのは実装上の問題であり、Leyboard の設計上の必然ではない。実際にはユーザが指を画面に置いている間呼び出され続けるメソッドにおいて、カウンターとなる変数の値（初期値 0）を 1 ずつ加算している。その値が 10 を超えるまでの時間がタップ入力であるかを判定する閾値となっている。

タップ入力とならない例外のキーは Shift キー等の修飾キーと後述するキーセット切り替え用のキー、Back Space、Delete であり、これらのキーでは、ユーザがそのキーの上に指を置く間入力が行われる。

3.2.1 キーの位置座標と領域

Leyboard ではキーはそれぞれ位置座標として x、y 座標の値を持つ。これが Leyboard における各キーの位置を決定する。

Leyboard ではユーザの入力の際に、ユーザのタッチ位置に最も位置座標が近いキーを入力の対象とする。これは、ユーザがキーを押しやすいように各キーの面積を最大限大きくするためである。キーとキーの境界がユーザに分かるように、キーの位置座標を基にしたボロノイ図が描画される。ボロノイ図を描画したのは、タッチ位置に最も位置座標が近いキーを入

[1] http://blog.controul.com/2009/05/speedy-voronoi-diagrams-in-as3flash/（参照 2014-01-19）

力の対象とする時にキーとキーの境界はボロノイ図と等しくなるためである。ボロノイ図の描画には Fortune's algorithm [For86] を使用した。このアルゴリズムは計算量が $O(n \log n)$ であるという特徴から、高速な描画が可能である。なお、Fortune's algorithm を実装する際には公開されているコードを移植した。

3.2.2 親指周りへのキーの配置

ユーザが入力する際に、その指の位置がホームキー周辺にとどまれば誤入力が減らせると筆者は考えた。そこで Leyboard では Space キー（物理キーボードにおける親指の担当キー）の他、後述するキーセット切り替え用のキーも含めて複数のキーを親指の周りに配置した。そのため Leyboard の配列は QWERTY 配列を元にしているものの、厳密には QWERTY 配列と異なったものになった。この配列により Leyboard の全てのキーはユーザのいずれかの指の位置あるいはその周囲に存在する。ゆえに、ユーザはいかなる入力においても、従来の QWERTY 配列のキーボードの時のように手を大きく動かす必要がない。

3.2.3 キーセット切り替え

テキスト入力に必要なキーの全てを一つのレイアウトとして配置すると、全てのキーがユーザの指の位置あるいはその周囲に配置するという Leyboard の提案要件を満たさなくなる。そこで Leyboard に 3 つのキーセットを用意し（図 3.2 から図 3.4）、それらを切り替えて用いるようにした。Leyboard ではユーザは入力中に必要に応じてキーセットを切り替えることになる。先述したように、キーセットを切り替えるキーは親指の周りに配置した。図 3.3 や図 3.4 上の円によって示されるキーセット切り替え用のキーをユーザが押している間、Leyboard のキーセットはそれぞれの図において示されるものに変化する。ユーザが指をキーセット切り替え用のキーから離したとき、キーセットは図 3.2 のアルファベットセットに戻る。これら 3 つのキーセットにより、Leyboard は合計 102 種類のキーが入力可能になっている。なお、キーセットを切り替えた状態においてさらに親指による入力を行うために親指スワイプ入力を設けている。

3.2.4 親指基点スライド

筆者は親指基点スライドと名付けた機能を実装した。一部の文字を入力する際に、2 つのキーを押す必要があることがある。例えば、f キーと Shift キーで大文字の 'F' を入力するときである。親指基点スライドは Leyboard のレイアウトにおいてこのような 2 つのキーの同時入力を容易にする。図 3.5 に親指基点スライドの例を示す。本例では左手親指によって Shift キーの入力を行っている。この時左手の人差し指から小指のキーの位置が、キャリブレーション時の親指との位置関係を保つように平行移動する。このように、親指基点スライドにより平行移動するキーは入力した親指と同じ手の人差し指から小指のキーである。この設計はユーザが

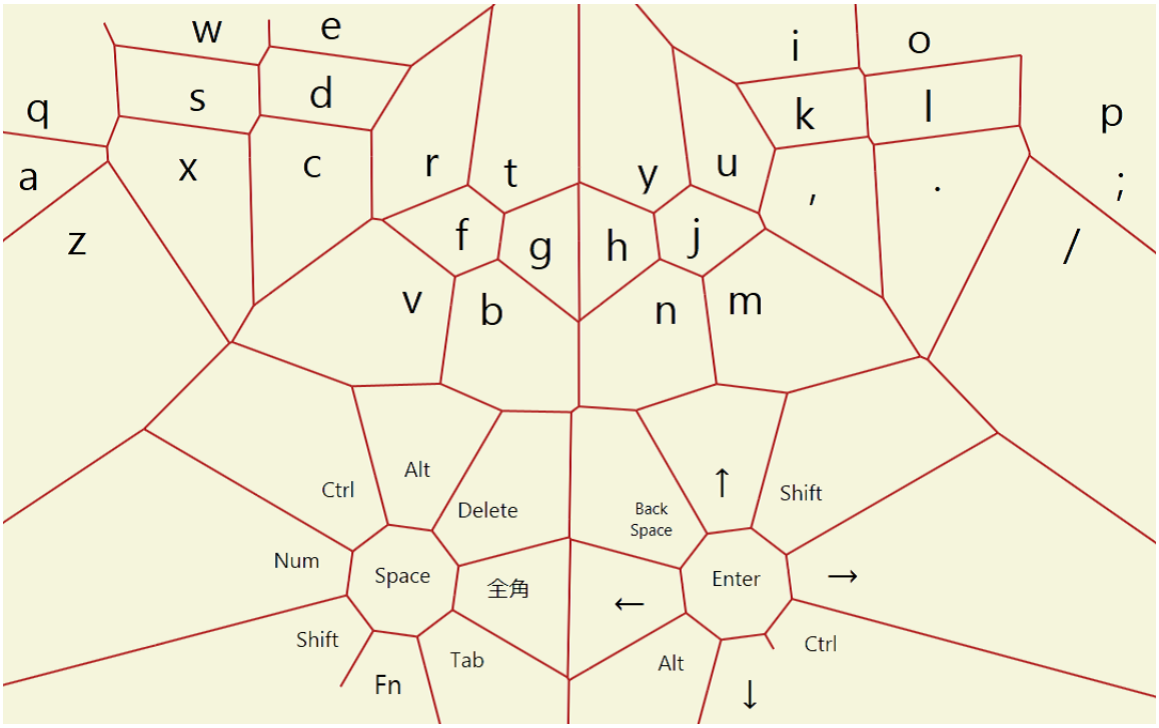

図 3.2: アルファベットセットの配列

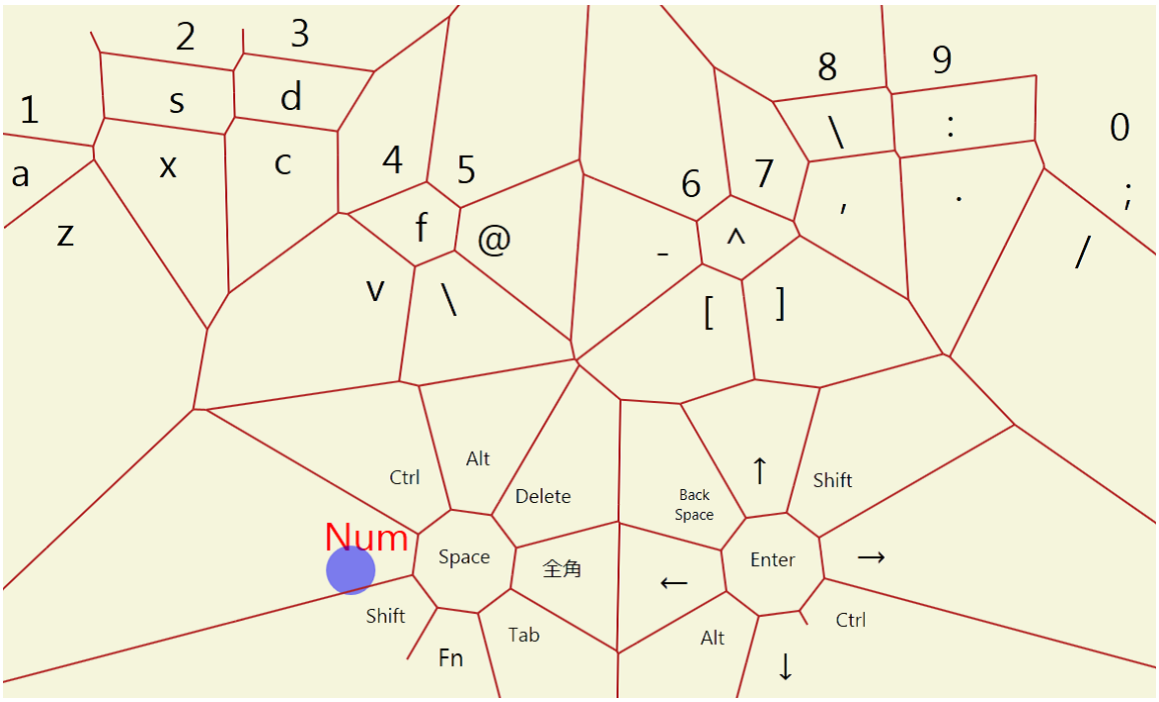

図 3.3: 数字・記号セットの配列

2 つのキーを同時入力する際に、手の姿勢を崩すことなしに入力することを可能にした。よってユーザはスムーズな入力であるがゆえに高速な入力が可能になると考えられる。親指基点スライドによって平行移動するキーの移動量と方向は、親指のホームキー（Space、Enter）の位置座標からユーザの親指が現在押しているキーの位置座標までの距離と方向に等しい。親指基点スライドによるキーの平行移動はユーザが親指のキーが押している間続き、ユーザが指を離したときに平行移動していたキーは元の位置へ戻る。

3.2.5 親指スワイプ入力

キーセット切り替え用のキーは押している間しかキーセットを切り替えない。Shift キーはキーセット切り替え用のキーと同じ左手親指が担当するキーなので、キーセット切り替え用のキーと Shift キーを左手親指にて同時に入力することはそのままでは不可能である。キーセット切り替え用のキーと Shift キーの左手親指による同時入力を可能にするために、親指とその周りのキーに、親指スワイプ入力を設けた。これは、親指をあるキーから別のキーへとスワイプさせることによってスワイプ先に存在するキーの入力を行うものである。親指スワイプ入力によって以下の 2 つのキー入力が可能になる。

- キーセットを数字・記号セットに切り替えるキー（Num）と Shift キーの同時入力

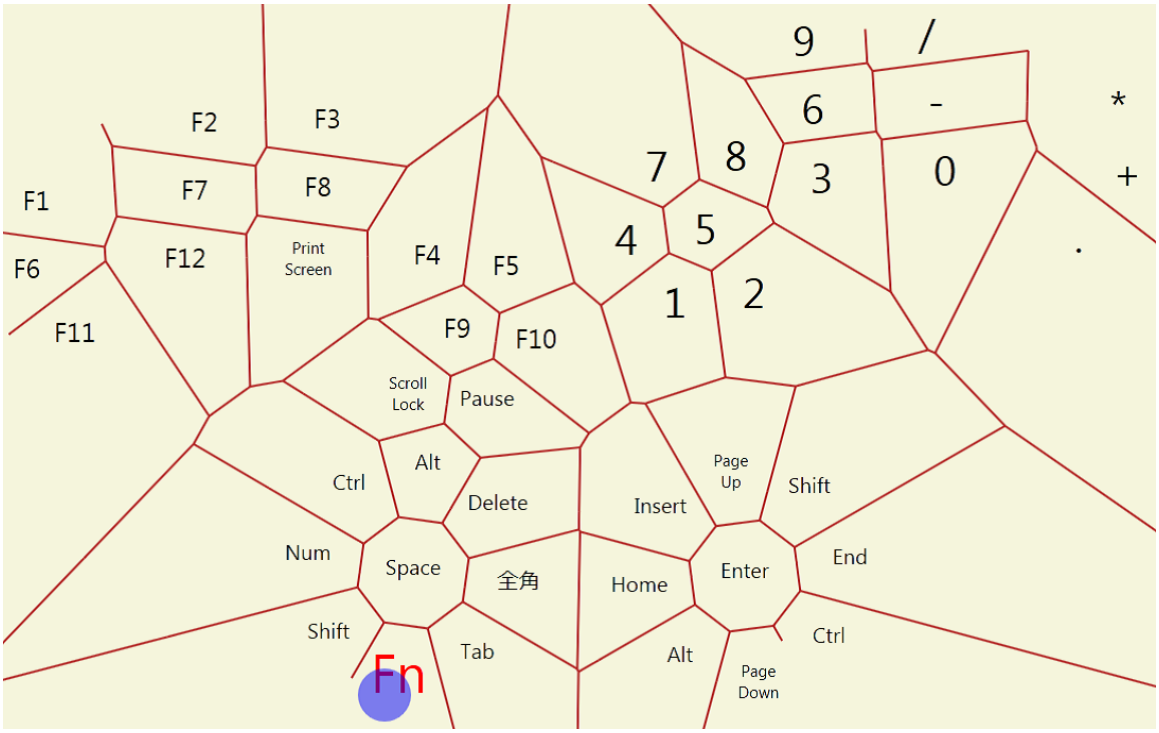

図 3.4: ファンクション・テンキーセットの配列

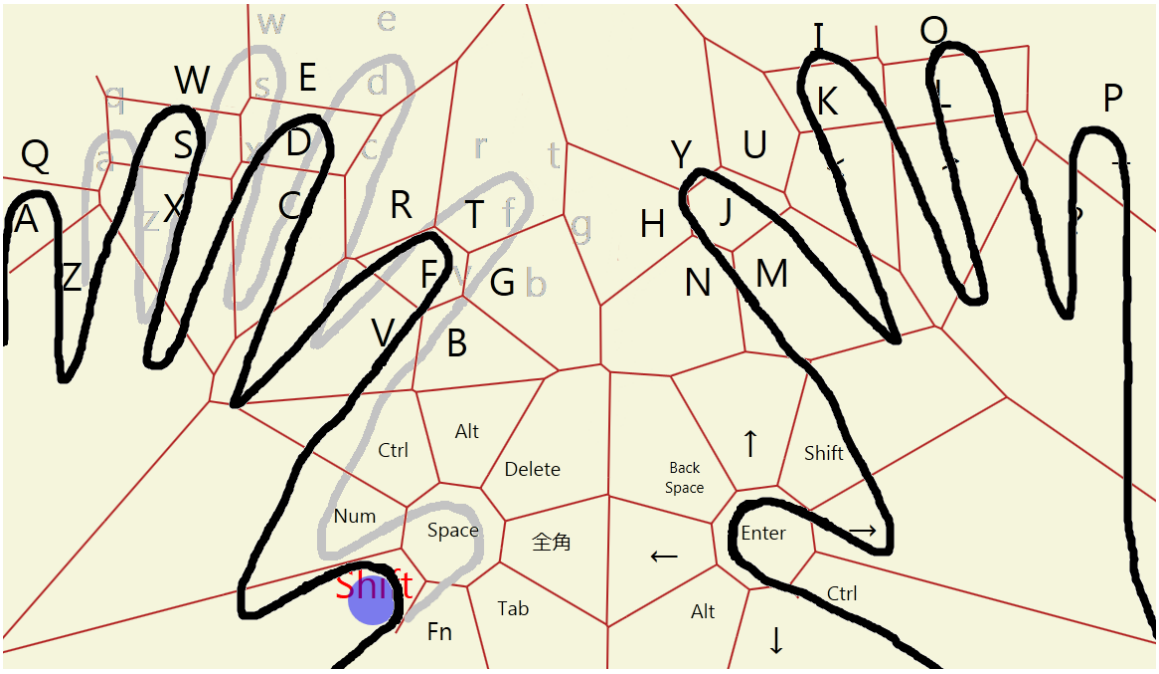

図 3.5: 親指基点スライドの例

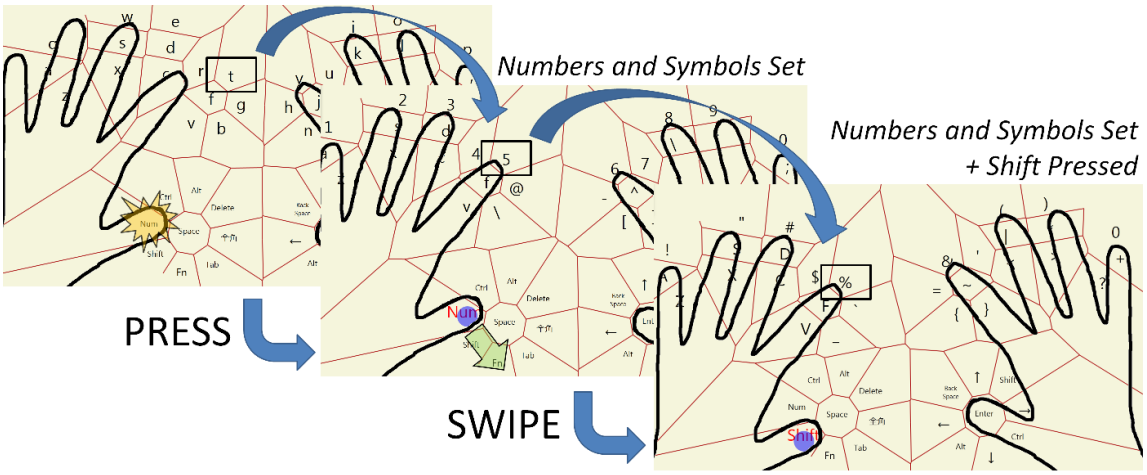

図 3.6: 親指基点スライドと親指スワイプ入力の連携例

- キーセットをファンクション・テンキーセットに切り替えるキー（Fn）と Shift キーの同時入力

Shift キー以外の修飾キー（Ctrl、Alt）の入力に関してはこれらのキーを左手だけでなく右手親指にも配置しているのでキーセット切り替え時にも入力可能である。なお、左手親指にしか存在しないキーも存在する（Tab、半角全角切り替えキー、Delete）が、これらはキーセットを切り替えながらの入力が必要ではないため右手親指には設けられていない。また、本来想定された用途ではないが、片側親指のキーを複数同時に入力することも可能である。この時、親指基点スライドは最後に入力されたキーの位置座標を基点として人差し指から小指のキーを平行移動させる。

3.2.6 親指基点スライドと親指スワイプ入力の連携

Leyboard は親指基点スライドと親指スワイプ入力を組み合わせることによって、ユーザの手の姿勢を崩すことなしに多種類の文字を入力可能にしている。図 3.6 に親指基点スライドと親指スワイプ入力の連携の例を示す。左手人差し指の右上の矩形内のキーに注目すると、左手親指をスワイプするにつれて、数字・記号セットに切り替わった状態、さらに Shift キーが押された状態とキーの表示が遷移している。この時、キーの位置は常にユーザの指の位置に合ったものになっている。

3.3 実験

Leyboard と既存の QWERTY 配列ソフトウェアキーボードの入力性能を比較する長期的な実験を行った。既存のものとして、筆者は Windows 7 に標準搭載されているソフトウェアキー

ボードを選択した。以降、これを Windows 7 キーボードとする。Windows 7 キーボードのアルファベットキーの形状は正方形になっており、1 辺の長さは 17mm であった。各ソフトウェアキーボードの幅（横幅）は共に 31cm であった。各ソフトウェアキーボードの高さ（縦幅）は Windows 7 キーボードが 10.5cm、Leyboard が 17.5cm であった。実験は約 1 年 4 か月という長期にわたって行われた。

3.3.1 実験環境

Leyboard を動作させるデバイスとして、Acer 社の ICONIA-F54E を用いた。ICONIA-F54E を使用した実験環境を図 3.7 に示す。ICONIA-F54E は液晶サイズが 14 インチ、解像度が 1366×768px（WXGA）の 10 点までのタッチ入力に対応するタッチパネルを搭載した端末である。ICONIA-F54E は 2 面のタッチパネルによって構成されるが、その下画面にて 10 本指のタッチを行った場合、ICONIA-F54E に用意された独自のソフトウェアキーボードが無条件に起動するようになっている。よって Leyboard を下画面にて使用することは不可能であった。そこで画面設定にて、それぞれの画面の表示を上下反転することによって上画面を見かけ上の下画面とし、そこにおいて Leyboard を動作させた。

3.3.2 参加者と実験課題及び実験期間

本実験の参加者は筆者一人である。

実験課題として、英文パングラムの入力を選択した。パングラムとはアルファベットの全ての文字を少なくとも 1 回用いた文章である。例えば “A quick brown fox jumps over the lazy dog.”が挙げられる。このパングラムは必ず大文字を含み、また複数の記号を含むものもある。1 つのパングラムは 31 から 63 の文字によって構成されていた。パングラムの入力をタスクとするのは関連研究の CATKey において行われており、本研究ではこれに倣った。パングラムの入力をタスクとすることの利点は、各文章で全てのアルファベットが必ず 1 回は入力されることである。そのため、全てのアルファベットにおいて少なくとも入力文章数と同じ数の出現が保証される。以降、10 の異なるパングラムを入力することを 1 セットとする。この 10 のパングラムは 120 用意したパングラムの中からランダムに選び出す形式によって、セットごとに決定された。

参加者は各ソフトウェアキーボードにおいて毎日 3 セットの入力を行った。本課題では入力を間違えた場合、そこから正しい入力をやり直す必要がある。言い換えれば、参加者が正しい入力を行わない限りタスクは進まない。参加者はこの実験を 2012 年 2 月 15 日から 2013 年 6 月 13 日までの間、計 483 日間行った。これは各ソフトウェアキーボードあたり 1449 セット分に相当する。期間と実験日数が合わない理由は、期間中実験を行わなかった日が 2 日存在するためである。なお、最初の 7 日間は Windows 7 キーボード、Leyboard の順にタスクを行い、次の 7 日間はその順番を逆にするという行程を繰り返している。これは、カウンターバランス法に基づいて、ソフトウェアキーボードを使用する順番が実験結果に影響を与える


How quickly daft jumping zebras vex.
Good :14
Bad :1
Result:CharEnd
Time :122.524

図 3.7: 実験環境

ことを避けるためである。よって、順番こそ異なるが参加者は 1 日のうちに双方のソフトウェアキーボードを用いて入力を行うことになる。ソフトウェアキーボードの入力時に画面を見ることに関して、参加者に特に規定は設けなかった。結果的に、参加者は実験時に双方のソフトウェアキーボードにおいて、画面を見ながらの入力を同程度に行った。なお、Leyboard におけるボロノイ図は常に表示されていた。

3.3.3 結果

入力率

筆者は入力率を 1 分間あたりに入力された単語数（wpm）として算出した。そのようにした理由は、参加者が各セットにおいて入力する文章内容が毎回異なるので、入力時間そのものは各ソフトウェアキーボードの性能を比較する基準にならないからである。wpm は Gentner によって定義された [Gen83]、1 分間あたりに入力された単語数を表す単位である。それは以下のように計算される。

$$\frac{1}{5}\frac{\text{入力誤りを除いたキーストローク数 (回/セット)}}{\text{入力の所要時間 (分/セット)}}$$

各ソフトウェアキーボードにおける入力率の変動及び対数近似に基づいた近似曲線を図 3.8 に示す。近似曲線を見ると、大きな差ではないが、Leyboard が既存のソフトウェアキーボードと比べて入力率に優れていることが分かる。各キーボードの入力率の最大値は Windows 7 キーボードが 59.34wpm、Leyboard が 61.27wpm となった。各キーボードの入力率の平均値は Windows 7 キーボードが 41.65wpm、Leyboard が 43.66wpm となった。各キーボードに十分に慣れた状態の数値として、2013 年 5 月 17 日から 2013 年 6 月 14 日までの 28 日分の各キーボードの入力率の平均値も求めた。その結果は Windows 7 キーボードが 47.09wpm、Leyboard が 49.69wpm となった。いずれの値においても Leyboard の入力率は Windows 7 キーボードよりも高くなった。

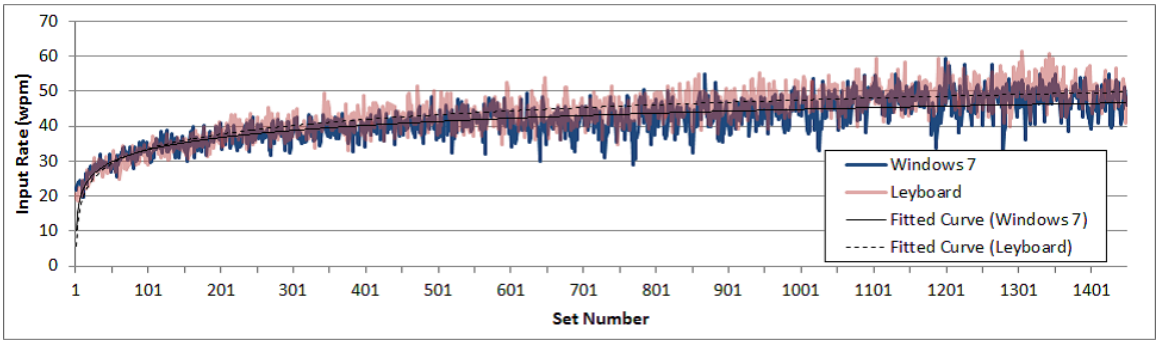

図 3.8: 各ソフトウェアキーボードの入力率（wpm）

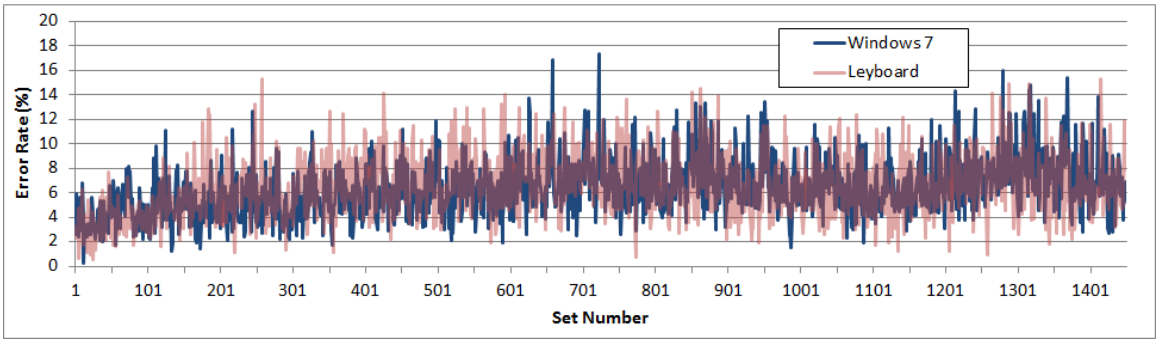

図 3.9: 各ソフトウェアキーボードのエラー率

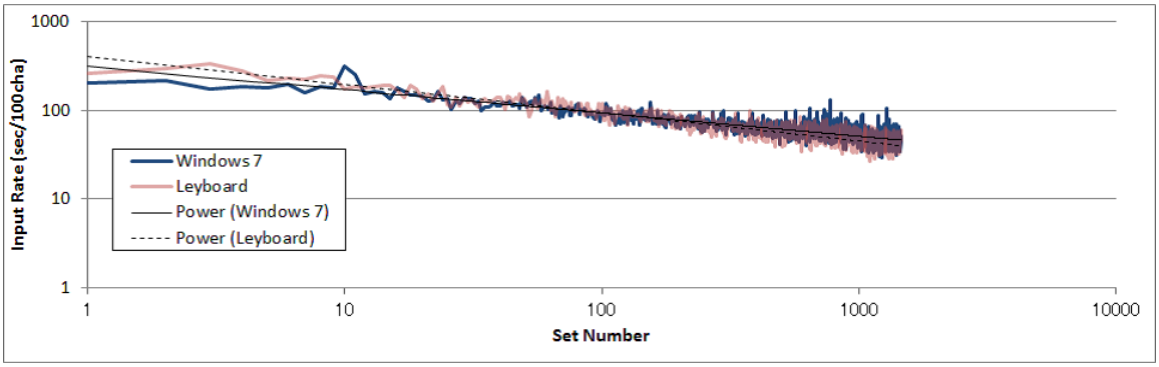

図 3.10: 各ソフトウェアキーボードの入力時間（100 文字あたりの入力時間）

エラー率

各ソフトウェアキーボードにおけるエラー率の変動を図 3.9 に示す。その平均値は Windows 7 キーボードが 6.36%、Leyboard が 6.54%であり、Leyboard の方が若干高い。

なお、参加者の物理キーボードにおける入力能力を調べるため、上記実験タスクを物理キーボードにて 9 セット行い、入力率とエラー率を求めた。その結果、入力率は 52.58wpm であり、エラー率は 5.98%であった。また、入力率の最大値は 55.63wpm であった。

3.3.4 入力速度の推移の妥当性

長期的に実験を行った場合、実験の値はべき乗の法則に従う。この時、対数グラフにおいて累乗近似曲線を出力すると直線に近づくことが知られている [NR81]。図 3.10 に 100 文字を入力するのにかかった時間を軸とした対数グラフを示す。累乗近似曲線が直線に近いため、本実験の経過がべき乗の法則に従っていることが分かる。よって、本実験における値の推移は妥当であると言える。また、今回の実験においては、1 週間おきに入力を行うキーボードの順番を入れ替えた。参加者は長期期間にわたり、毎日双方のキーボードの入力を行っていたた

め、キーボードを入力する順番を交代することによる違和感は特に抱かなかった。実際、交代が行われた日のデータを排除したうえで入力率の平均値を求めたところ、Windows 7 キーボードが 41.63wpm、Leyboard が 43.71wpm となり、全データの 41.65wpm、43.66wpm と比較して特に差はなかった。

3.3.5 考察

本章では Leyboard の実験の結果を受けて、Leyboard が従来のソフトウェアキーボードに対して有用な点、及び Leyboard の問題点に関しての考察を行う。

キーボードごとの入力率及びエラー率の差

Leyboard と Windows 7 キーボード、それぞれのキーボードの入力率とエラー率の全データ（全 1449 セット分の入力率及びエラー率の数値の集合）に関して、それぞれ平均値に有意な差があるかを確かめるために、対応のある t 検定を行った。ユーザは実験が進むにつれてそれぞれのソフトウェアキーボードの入力に習熟していくので、キーボードの各セットに対応が存在する。検定の結果、Leyboard の入力率は Windows 7 キーボードに対して有意に高いことが判明した（$t = -17.41, p = 2.2e^{-16}$<0.01）。一方、エラー率の有意差はなかった（$t = -2.203, p = 0.02777$>0.01）。

上記の結果は実験全体のデータを比較したものである。十分に慣れた状態において、両者に有意差があるかを確かめるために、2013 年 5 月 17 日から同年 6 月 13 日までの 28 日分のデータのみを抽出し、再度検定を行った。その結果、Leyboard の入力率は Windows 7 キーボードに対して有意に高く（$t = -4.973, p = 3.506e^{-06}$<0.01）、エラー率に有意差はなかった（$t = 0.7552, p = 0.4523$>0.01）。

エラー内容分析

次に、エラーの内容を分析した。筆者は実験途中の全てのキー入力内容に対してログを取った。そこで上記ログを参照し、以下のアルゴリズムに基づいて誤字と脱字の判定を行った。

1. 単語の判定は前方一致の形式で行う。
2. 入力と正解を文字単位によって比較し、正しい入力が行われている個所はスキップする。
3. 入力と正解が異なる場合、正解の次の文字に注目する。それが入力と一致した場合は脱字、それ以外は誤字と判定する。
4. 誤りが生じた場合、次に正しい入力が行われるまでの入力は全て無視する。これは脱字による入力のずれを誤りとしないためである。

例えば、テキストが “puppy”で、それに対する入力が “pupy”であった場合、このアルゴリズムは ‘p’ の脱字があったと見なす。別の例として、テキストが “lazy”で、それに対する入力が “kazy”であった場合は ‘l’ の誤字があったと見なす。

本実験では参加者が正しい文字を入力するまで次の入力を受け付けない仕様になっている。参加者が誤った入力を続ける限り入力は進まず、参加者は最終的には正しい文字を入力することになる。その結果、エラーのある入力は本来のテキストに特定の文字が挿入されたものになる。つまり、最初の例では、実際の入力は “pupypy”（pup（y）py）となり、 次の例では “klazy”（(k）lazy）となる。先のアルゴリズムに基づき、これらの例はそれぞれ 1 回の脱字、1 回の誤字と判定される。

Windows 7 キーボードには 23441 回、Leyboard には 23252 回のエラーが存在した。その内訳は以下の通りである。Windows 7 キーボードには 17376 回の誤字と 6065 回の脱字が存在した。Leyboard には 14521 回の誤字と 8731 回の脱字が存在した。エラーの回数自体は Leyboard の方が若干少ないが、Leyboard は Windows 7 キーボードに比べて誤字が少なく脱字が多いように思われる。そこで、2 つのキーボードのエラー回数を文字別に分けてグラフ化した。誤字に関しては図 3.11 に、脱字に関しては図 3.12 に示す。なお、グラフからは誤り回数が両者ともに一定未満である文字（誤字は 50 回、脱字は 25 回）を除いた。ここで知りたいのはエラーが多くなる文字であったため、エラーの回数が非常に少ない（0.3%未満の比率である）上記の文字に関しては紙面の都合からグラフから排除した。グラフを見ると、エラー回数に極端な差がある文字が存在することが分かる。

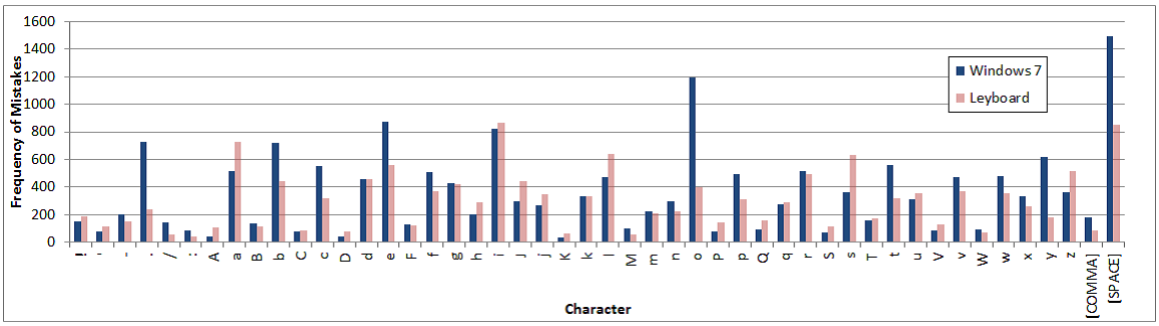

図 3.11: 誤字の分析結果

3.3.6 誤字内容の分析

Windows 7 キーボードと Leyboard の誤字内容に関して、多かったパターン上位 10 位をそれぞれ表 3.1 と表 3.2 に示す。

表 3.1 を見ると、正解と誤字の組み合わせは Space と n 以外全て横に隣り合ったキーによって構成されている。Windows 7 キーボードでは横に隣り合ったキーの誤入力が多いと言える。

表 3.2 を見ると、Leyboard における誤字のパターンは複数に分類される。s と x、l と o、z と a、a と q、a と z は縦に隣り合ったキーである。J と j は Shift キーの押し損ねによって発

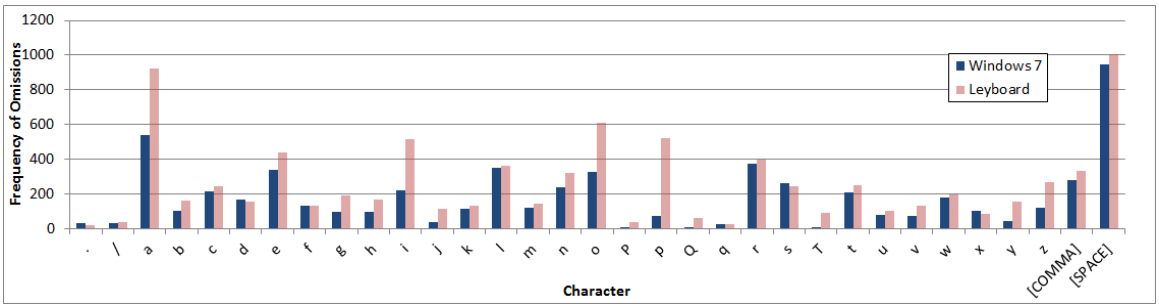

図 3.12: 脱字の分析結果

生している。Space と s は空白を入れるべきところに s を入力したことにより生じる。恐らく複数形にしなくてもよい単語を複数系にしたために生じた誤字である。i と o、u と y、v と b は横に隣り合ったキーである。よって Leyboard では Windows 7 キーボードより縦に隣り合ったキーの誤入力が多いと言える。そのため、縦に隣り合ったキーの誤入力が軽減できれば、Leyboard の入力率はさらに向上すると考えられる。

個々の比率や比率の累計を見ると Leyboard は Windows 7 キーボードより低いので、Leyboard の誤字の内容は Windows 7 キーボードと比べて偏りが小さい。

なお、Space と s の誤字に関して、Windows 7 キーボードでは 123 回生じていた。Leyboard の数が多い（177 回）傾向にあるものの、Space と s の誤字は双方のソフトウェアキーボードに見られる現象である。一方、J と j の誤字は Windows 7 キーボードでは 159 回生じており、Leyboard の 322 回とは大きく異なっていた。Leyboard の方が上記誤字を生じやすいと考えられる。その原因は、Windows 7 キーボードと Leyboard における Shift キーの位置が大きく異なるためと思われる。前者はアルファベットキーの外側にあり、後者は親指周りにある。そのため、Leyboard の方が j キーと Shift キーの距離が近い。距離が近いと手をほぼ動かさずに入力できるので、参加者は高速に入力しやすくなる。また、Shift キーと j キーは異なる手によって入力される。前者は左手親指であり、後者は右手人差し指である。入力する手が異なると、それぞれ別に入力できるので入力は高速になるが、入力する順番が逆になる可能性がある。入力する順番が逆になった時、参加者は左手親指が Shift キーに触れる前に右手人差し指が j キーに触れている。あるいは、右手人差し指が j キーから離れる前に左手親指が Shift キーから離れている。この事態は入力を高速に行うことを試みたときに生じやすい。Shift キーと j キーの距離が近い Leyboard において、参加者が別々の手によって入力可能な Shift キーと j キーを高速に入力しようとした結果、入力する順番が逆になり誤字となったと考えられる。

3.3.7 脱字内容の分析

Windows 7 キーボードと Leyboard の脱字内容に関して、多かったパターン上位 10 位をそれぞれ表 3.3 と表 3.4 に示す。2 つのキーボードに共通して出現しているパターンは “ick”、“fro”、“for” のみである。個々の比率や比率の累計を見ると Leyboard は Windows 7 キーボードより

表 3.1: Windows 7 キーボード誤字内容上位

回数	正解	誤字内容	全体中の比率（%）	比率の累計（%）
823	[SPACE]	n	4.74	4.74
693	o	p	3.99	8.73
460	b	v	2.65	11.38
440	i	o	2.53	13.91
418	.	,	2.41	16.32
367	p	o	2.11	18.43
344	t	r	1.98	20.41
339	y	u	1.95	22.36
318	o	i	1.83	24.19
316	e	w	1.82	26.01

表 3.2: Leyboard 誤字内容上位

回数	正解	誤字内容	全体中の比率（%）	比率の累計（%）
510	i	o	3.51	3.51
322	J	j	2.22	5.73
318	s	x	2.19	7.92
298	l	o	2.05	9.97
241	z	a	1.66	11.63
230	a	q	1.58	13.21
186	a	z	1.28	14.49
177	[SPACE]	s	1.22	15.71
176	u	y	1.21	16.92
169	v	b	1.16	18.08

高い。よって Leyboard の脱字の内容は Windows 7 キーボードと比べて偏りが大きいほか、その特性が異なると言える。特に Leyboard の脱字における “uiz”や “mph”の比率は、Windows 7 キーボードの脱字内容上位の比率と比べると高いといえる。

表 3.3: Windows 7 キーボード脱字内容上位

回数	1 文字前	脱字内容	1 文字後	全体中の比率（%）	比率の累計（%）
60	[SPACE]	l	a	0.99	0.99
57	i	c	k	0.94	1.93
56	f	r	o	0.92	2.85
56	h	,	[SPACE]	0.92	3.77
51	t	c	h	0.84	4.61
47	f	o	r	0.77	5.38
46	e	r	[SPACE]	0.76	6.14
44	e	[SPACE]	m	0.73	6.87
42	e	d	[SPACE]	0.69	7.56
41	a	c	k	0.68	8.24

表 3.4: Leyboard 脱字内容上位

回数	1 文字前	脱字内容	1 文字後	全体中の比率（%）	比率の累計（%）
199	u	i	z	2.28	2.28
121	m	p	h	1.39	3.67
87	f	o	r	1	4.67
85	e	n	[SPACE]	0.97	5.64
84	u	i	c	0.96	6.6
71	[SPACE]	T	V	0.81	7.41
70	f	r	o	0.8	8.21
70	[SPACE]	o	f	0.8	9.01
65	i	c	k	0.74	9.75
59	n	g	[SPACE]	0.68	10.43

Leyboard において脱字が多くなった要因として、タップによって入力を行うという仕様上の問題が考えられる。ソフトウェアキーボードにおいてキーが入力される瞬間はソフトウェアキーボードごとに異なるが、それらを大きく分けると、ユーザが指を画面に置いた時に入力を行うものと指を画面から離した時に入力を行うものの 2 種類に大別することができる。Leyboard はユーザがタップ入力を行ったときのみにキー入力を実行するため、後者に分類される。この場合、キーの入力順番はユーザが指を画面から離す時刻のみによって決定される。

物理キーボードにおいてはユーザがボタンを押した瞬間にキーが入力される。そのため、ソフトウェアキーボードにおいては、ユーザが指を画面上のキーに置いた順にキーが入力されることが自然に感じられる。しかし、ユーザが画面に指を置く順番とユーザが画面から指を離す順番は同じであるとは限らない。ゆえに次のような問題が発生する。ユーザが例えば u、i の順に指を置いても、指を i、u の順番に離していたならば、入力は i、u になる。するとユーザには入力が入れ替わったように感じられる。これを見かけ上の入力入れ替わりと呼ぶことにする。この実験でエラー分析の際に、入力と正解が異なる場合において、正解の次の文字が入力と一致した場合を脱字とみなしている。よって先に挙げた、ユーザが画面に指を置いた順番と離した順番が異なることによって発生する、見かけ上の入力入れ替わりも脱字と解釈される。Leyboard において脱字が多くなった要因は、この見かけ上の入力入れ替わりが多数発生したためと考えられる。

Leyboard においては見かけ上の入力入れ替わりが発生し得る。Leyboard と同様に指を画面から離した時に入力を行うソフトウェアキーボードには、見かけ上の入力入れ替わりを回避するために、ユーザが画面に指を置いた順番を記憶しているものがある。この種のソフトウェアキーボードはユーザが画面に指を置いたときにその順番を記憶している。そして、画面からユーザの指が離れた際に、ユーザが画面に指を置いた順番に沿って、画面から離れた指の分までの入力をまとめて行っている。するとキーの入力順番はユーザが画面に指を置いた順番によって決定されることになるため、見かけ上の入力入れ替わりは発生しない。例えば Windows 8 に標準搭載されているソフトウェアキーボードはこのような仕様になっている。

Leyboard で見かけ上の入力入れ替わりが発生するのは、上記の仕様を取り入れなかったからである。取り入れなかった理由は、Leyboard ではキー入力以外にキャリブレーションが求められるためである。上記の仕様は前提として、ユーザが画面に指を置く場合はキー入力時に限られるという条件を持たなければならない。つまり、ユーザの画面に指を置く行為がキー入力を意図したものであると保証することによって、ユーザが画面に指を置いた順番に沿ってまとめて入力を行うことがソフトウェアキーボードにとって可能になる。これが保証されない場合、ユーザがキー入力を意図せずに画面においた指がキー入力を誘発し、結果として誤入力になる。Leyboard においてはユーザが画面においた指がタップによるキー入力途中なのか、キャリブレーションを意図したものなのかがユーザが画面に指を置いたその時点では確定しない。それが確定するのは、画面からユーザの指が離れ、タップ入力と解釈される、あるいはユーザが指を置いたまま時間が経過し、タップ入力である可能性がなくなる時である。そのためユーザが画面に指を置く場合はキー入力時に限られるという前提が成立しないので、Leyboard では上記の仕様を取り入れていない。

3.3.8 手元注視が必要となった原因

Leyboard では従来のソフトウェアキーボードと同程度の手元注視を必要とした。その原因として考えられるのは、手の位置のずれである。Leyboard は手を浮かせながら入力を行う。よって、手を同じ位置に保ちながら入力を行うことが困難であり、入力中にキャリブレーショ

ンを行った位置から各指の位置がずれていく。結果、そのずれの修正のために手元注視が必要となったと考えられる。

手を同じ位置に保つ方法として、手のひらの下側の位置を机上に置くことによって手の位置を安定させることが考えられる。これは物理キーボードにおけるタッチタイピングにおいて実際使われる手法である。この時、パームレストまたはリストレストと呼ばれる台上に手のひらの下側を置くことがある。

Leyboard において同様に手の位置を安定させれば各指の位置ずれを抑えられるかを検証する実験を行った。内容は実験参加者を 3 人、時給 820 円にて雇用し、手元を不透明な段ボール板によって覆い隠した状態にて 1 セッション分（英文 10 文）の入力を行ってもらうものである。参加者は手のひらの下側を置き、手の位置を安定させた状態において入力を行った。タッチパネルが誤反応を起こすことを防ぐために、参加者には文庫本をパームレストとして用いてもらった。実験時に、参加者のタッチ位置を記録した。 各指のタッチ位置が重ならず、一定の範囲内の位置に留まるのであれば、手の位置を安定させることにより手元注視が不要となると考えられる。

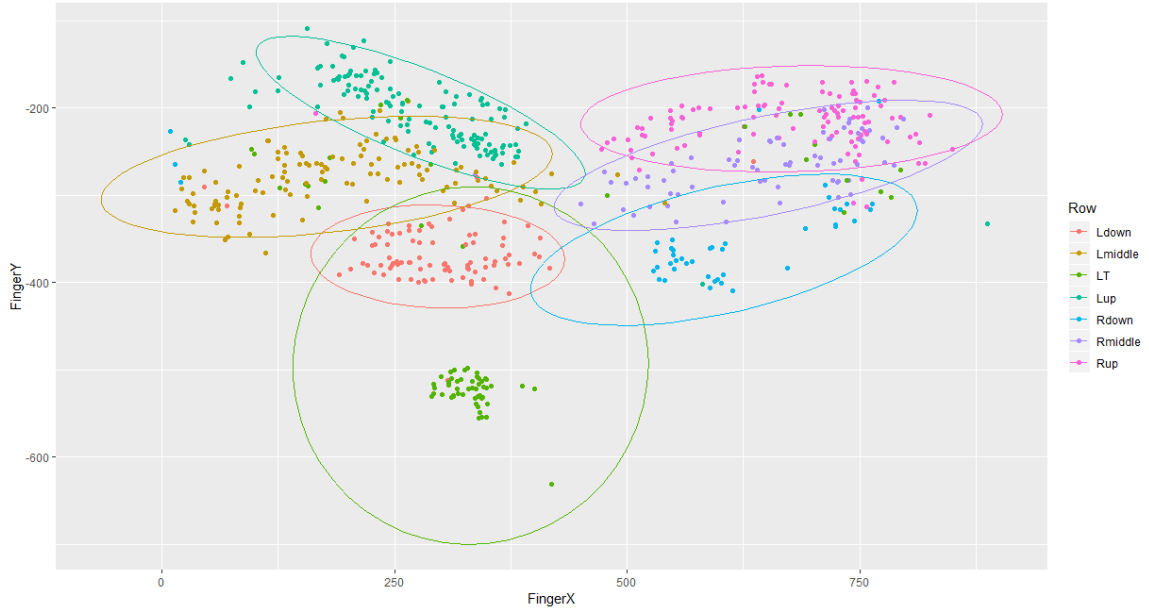

図 3.13: 参加者 A のタッチ位置（行方向）

実験の結果、3 人の参加者の入力率はそれぞれ 3.48wpm、2.11wpm、0.92wpm、エラー率はそれぞれ 69.3%、86.1%、90.5%となった。この結果から、Leyboard では手元注視がないタッチタイピングが困難であると考えられる。

3 人の参加者のタッチ位置を図 3.13 から図 3.18 に示す。なお、Leyboard では垂直方向（Y 軸方向）において下向きを正としているため、ここでは Y 座標の正負の値を反転している。図 3.13 から図 3.15 が Leyboard の各行のタッチ位置を、図 3.16 から　図 3.18 が各列のタッチ

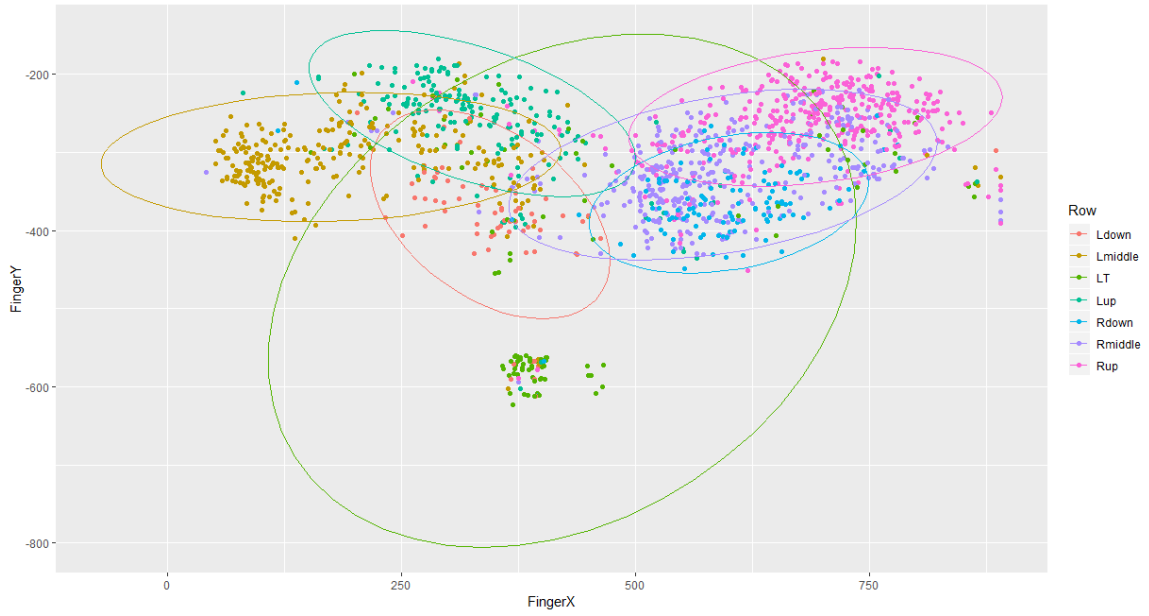

図 3.14: 参加者 B のタッチ位置（行方向）

図 3.15: 参加者 C のタッチ位置（行方向）

図 3.16: 参加者 A のタッチ位置（列方向）

図 3.17: 参加者 B のタッチ位置（列方向）

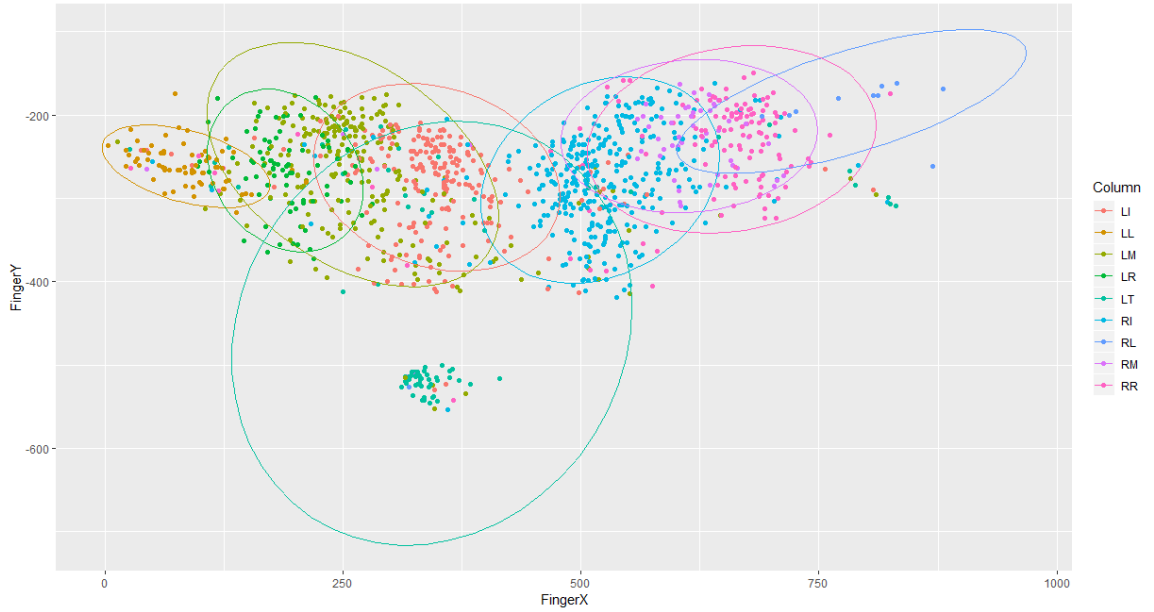

図 3.18: 参加者 C のタッチ位置（列方向）

位置を確率楕円によって表現している。図 3.13 から図 3.15 において、Lup、Lmiddle、Ldown は左手の上段、中段、下段を、Rup、Rmiddle、Rdown は右手の上段、中段、下段を、LT は左手親指を表す。図 3.13 から図 3.15 において、一文字目の L、R は左右の手を、2 文字目の T、I、M、R、L はそれぞれ親指、人差し指、中指、薬指、小指を表す。例えば、LR は左手薬指を表す。 図 3.13 から図 3.18 を見ると、行、列共にタッチ位置が重なっていることが分かる。よって、入力が進むにつれて、手の位置がずれていったと推測される。そして、このずれは手のひらを置いて手の位置を安定させても生じる。

以上より、Leyboard では手の位置の安定を問わずタッチ位置のずれが入力中に生じるため、タッチタイピングが困難となっている。

3.3.9 手元注視を減らすための要件分析

Leyboard の手元注視を減らせなかった理由から、次に実装するキーボードのプロトタイプの仕様や課題を定めた。

キーの数を減らす

Leyboard の手元注視を減らせなかった理由の 1 つとして、1 つの指が担当するキーの数が多いことが考えられる。Leyboard においては表 3.5 に示すキーと指の対応において入力が行われることを想定している。これは物理キーボードと等しく、1 つの指の担当するキーの数は最小 3 つ最大 6 つである。さらに表 3.5 からは省略しているが親指が担当するキーの数は

9 つであり、非常に多い。指が担当するキーが多くなるほど、それらのキーの入力に合わせて指の位置を使い分ける必要がある。例えば担当するキーが 3 つである中指の場合、上段のキーを入力する際の指の位置、中段のキーを入力する際の指の位置、下段のキーを入力する際の指の位置の 3 つの指の位置が存在し、これらを使い分けなければならない。しかし物理的なフィードバックに乏しく、キーの位置に指を合わせにくいソフトウェアキーボードでは指の位置の使い分けが困難である。Leyboard には既存のソフトウェアキーボードと同様に物理的なフィードバック（キーを触って感知すること）は存在せず、フィードバックとして視覚フィードバックとフィードバック音が存在した。前者は手元注視を行わない場合においては効果に乏しく、後者はキーの位置に応じて異なるフィードバック（音）を返すのではなく、常に同じ音を鳴らすためユーザがキーの位置を把握する用途には有用ではない。以上の理由から、Leyboard のフィードバックは指の動かし方を使い分けることに貢献しているとは考えにくい。そのため物理的なフィードバックを設けないのであれば、使い分けるべき指の位置の数を Leyboard よりも少なくするべきである。これは 1 つのキーセットにおけるキーの数を減らすことに等しい。以上より、ソフトウェアキーボードにおいて手元注視を減らすためには、キーの数を減らすことが有効であると考えた。

表 3.5: Leyboard におけるキーと指の対応（親指は多数につき省略）

左手				右手			
小指	薬指	中指	人差し指	人差し指	中指	薬指	小指
q	w	e	r、t	y、u	i	o	p
a	s	d	f、g	h、j	k	l	;
z	x	c	v、b	n、m	,	.	/

Leyboard において誤字の原因の一つは縦に隣り合ったキーの誤入力であった。この誤入力は縦に隣り合ったキーの誤入力は中段のキーと間違えて上下段のキーを入力する、あるいは上下段のキーと間違えて中段のキーを入力することにより生じる。よって、図 3.19 のように中段のキーを除けばこの誤入力は生じ得ないと考えられる。

また、図 3.13 から図 3.15 よりキーボードの行が 3 つ（上段、中段、下段）存在すると、上段と中段、中段と下段のタッチ位置が重なることが分かっている。

そこで次に実装するキーボードは中段のキーを除く。するとキーセットにおけるキーの数が減るので、使い分けるべき指の位置の数が減り、ユーザの手元注視の回数を減らせると考えられる。この機能の課題は、排除した中段のキーの入力方法を決めなければならないことである。中段のキーの入力方法は上下段の入力と区別するために、上下段の入力とは異なっていなければならない。

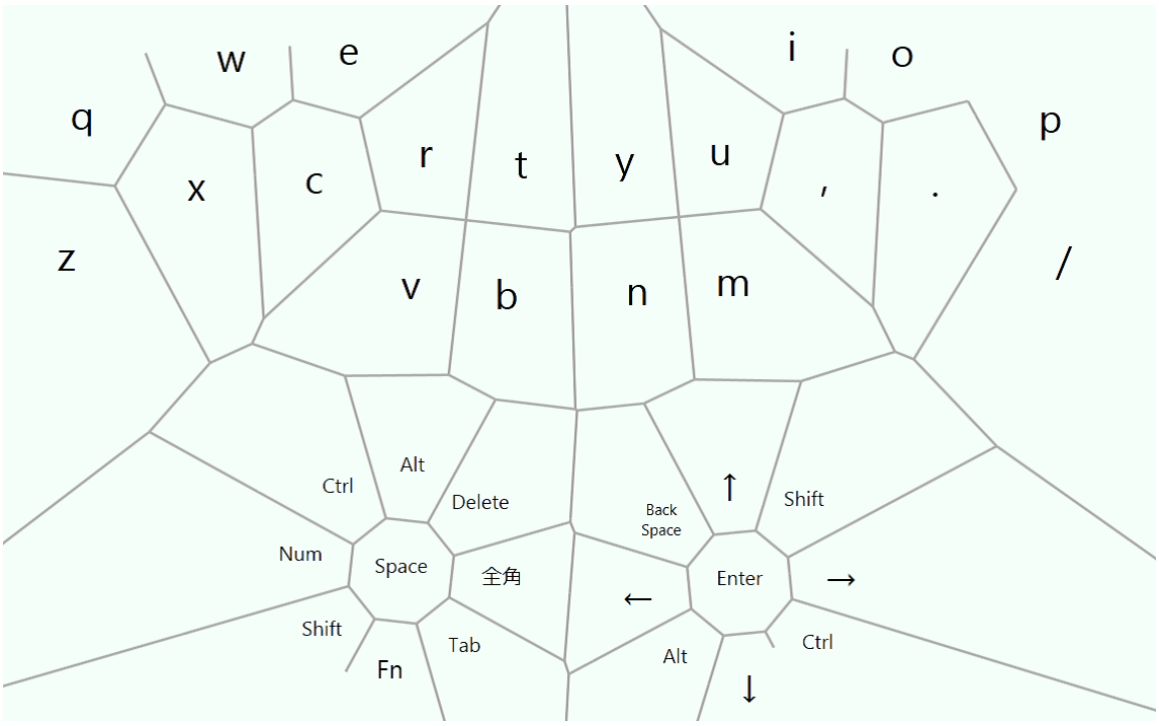

図 3.19: 中段のキーを排除した配列

不要なキーを除外する

キーの数を減らすために、不要なキーを除外することも考える。Leyboard には文字入力において全く用いられない（ゆえに実験においても全く用いられなかった）キーが含まれていた。例えば、Fn・テンキーセットに用意されたキーや、Ctrl、Alt のような修飾キーの一部がこれに該当する。これは Leyboard が物理キーボードで入力可能なキーを入力可能であるようにしたためである。しかし、ソフトウェアキーボードの既存製品は文字入力のみを前提としていると考えて良いと思われる。言い換えると、既存製品では物理キーボードを忠実に模したものを除けば、先に挙げたキーは入力不能であるものが多い。例えば Leyboard の実験に用いた Windows 7 キーボード（図 3.20）では物理キーボードのキー配置が忠実に再現されていたが、Windows 8 ではよりシンプル化したレイアウトのソフトウェアキーボード（図 3.21、以降 Windows 8 キーボードと呼ぶ）が採用された。そこで次に実装するキーボードも文字入力のみを前提とし、それに不要なキーを除く。

手のひらを固定できるようにする

Leyboard ではユーザの手の一部が画面に触れると誤入力を誘発する可能性があるため、ユーザは入力に用いる指のみを適宜画面に置き、手自体は画面上を浮かせながらの入力を行った。そのため、手の位置を固定する方法がなく、入力が進むにつれて手の位置が変わりやすくなっ

図 3.20: Windows 7 キーボード

図 3.21: Windows 8 キーボード

ていた。これはユーザに手の位置を調整するための画面注視を増やす要因になったと考えられる。そこで次に実装するキーボードは、手のひらの下 1/2 程の範囲を机上など（端末の画面外になる）に置き、その位置を固定しながら入力を行うことができるようにする。この時端末は横倒しの状態にて置かれていることを前提とする。

第4章　適応型減数ソフトウェアキーボード

Leyboard の実験結果を受けて、ユーザの手元注視を減らす適応型減数ソフトウェアキーボードとして Meyboard の実装を行った。

図 4.1 から図 4.4 に Meyboard の各種キーセットを示す。その配列は QWERTY 配列を基にキーを格子状に並べ、さらに中段のキーを排除したものになっている。r、v、u、m のキーは、その位置がユーザの左右それぞれの人差し指が来るべきところであることを示すガイドとなるようにキーの色を変更した。物理キーボードにおいては f、j キーに設けた突起を人差し指のガイドにすることによってユーザが指をキーに合わせやすいようにしている。r、v、u、m のキーの色変更はこれに沿っている。基本的に各指が担当するキーの個数は上下段それぞれのキーの 2 つであるが、人差し指は例外的に 4 つ（物理キーボードにおいては 6 つ）となる。そのため、色変更はそれら 4 つのキーの位置や境界を分かりやすくする効果もある。

図 4.1: 半角アルファベットセットの配列

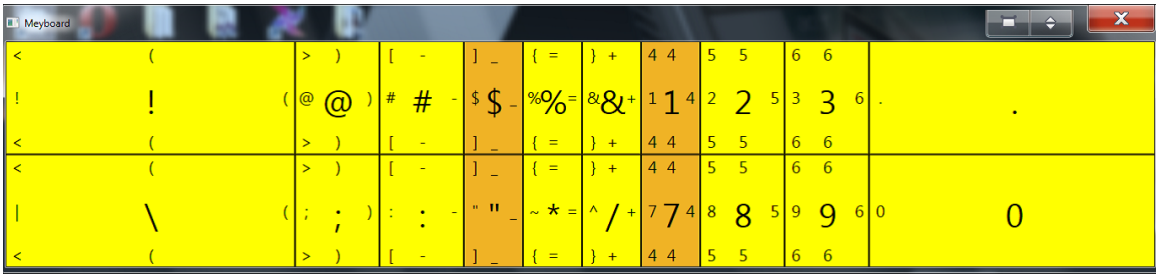

図 4.2: 半角数字・記号セットの配列

Meyboard ではユーザが手のひらを机の上に置き、手の位置を固定しながら入力が可能なようにした。結果、Meyboard の画面を占有する領域は Leyboard に比べて小さくなった。 これ

図 4.3: 全角アルファベットセットの配列

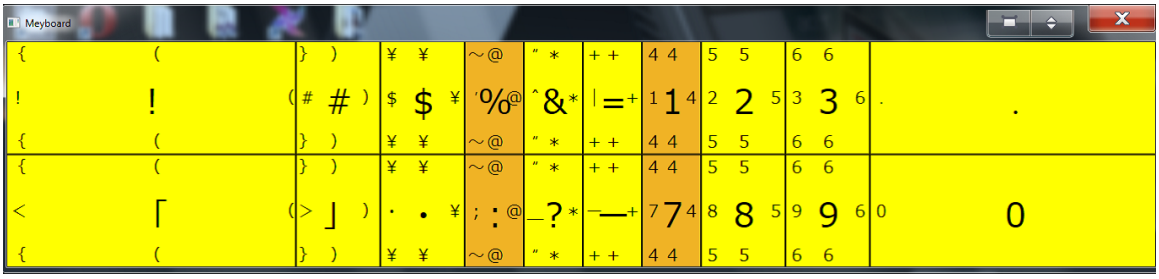

図 4.4: 全角数字・記号セットの配列

により、ユーザの親指が画面に届かなくなったため、Meyboard では左右の 8 本の指によって入力を行うものとして、親指を使わないこととした。なお、タブレット端末が手のひらによる入力と指による入力を区別できれば親指を使うことも可能と考えられるが、今回の実験にて用いた端末ではタッチ点を同時に 10 点までしか認識できず、すべての指と手のひらによる入力を認識させることが困難であったため行わないこととした。

4.1 キャリブレーション

Leyboard のキャリブレーションはユーザの指の位置にホームポジションキーを配置し、その周囲に非ホームポジションキーを配置するものであった。ホームポジションキーは中段のキーであるが、Meyboard は Leyboard と異なり中段のキーが存在しない。そのため、キャリブレーションにおいては Leyboard と異なった方法にてキーの位置と大きさをユーザの指に合わせている。

4.1.1 キャリブレーションの方法

Meyboard ではユーザが片手の人差し指から小指までの 4 本の指を置き、かつ各指の位置を変えないまま、指を置いた時間から 500 ミリ秒が経過するとキャリブレーションが行われる。500 ミリ秒の待ち時間を設けたのは、文字の入力時にキャリブレーションが行われないように

するためである。キャリブレーション時の手の姿勢として望ましいのは、ユーザが自然に指をタッチパネルに置いた時の姿勢であり、文字の入力中の姿勢は指の曲げ伸ばしなどが生じているため、キャリブレーションを行うには不適切であると判断した。

Meyboard では Leyboard と異なり片手ごとにキャリブレーションが可能なようにした。これは利便性のためである。例えば右手にて文字を入力すると同時に左手側のキャリブレーションを行うといった、入力と並行して入力を行っていない側の手のキャリブレーションを行うことができれば入力時間の短縮につながる。

4.1.2 各タッチと指の対応

Meyboard には入力に用いられたユーザの指を推測する機能がある。これを利用して各タッチと指の対応をとる。その手法は 4 本の指のタッチ点の水平方向の座標（X 座標）を比較し、左手の場合は最も左にあるものから小指、薬指、中指、人差し指と解釈し、右手の場合は最も左にあるものから人差し指、中指、薬指、小指と推測する単純なものである。

Meyboard においては、上記の推測手法でも十分な精度がある。ユーザの手の傾きが急であると上記手法は成立しない（例えば右手中指が右手人差し指よりも左側に位置することがある）が、キーボードの入力時にこのような姿勢をユーザが取ることは考えにくい。また、Leyboard のように 10 本指にて入力を行う場合、ユーザの手の傾きによっては親指が人差し指の左側にも右側にも来る可能性があるが、Meyboard では左右の手の人差し指から小指の 8 本のみを使用するため親指位置の考慮は不要である。

なお、タッチ点が左手によるものか、右手によるものかは指が触れたキーによって判断している。例えば、f キーに触れた場合は、左手による入力と解釈する。そのため、左手によって右手のキーを触れていた場合はキャリブレーションが正常に行われないことになるが、ユーザの左右の手が十分に離れていればこれは生じない。仮に
delspan そうなった左右の手が近づきすぎていた場合は、左右の手をより離してもう一度キャリブレーションを行う必要がある。

4.1.3 キーの幅と高さ

Meyboard ではユーザの指と指の中間の位置を通る、垂直線がキー間の高さ方向の境界となる仕様にした。これを満たすようにキーの幅を決定することがキャリブレーションの内容となった。一方、キーの高さは常に一定の値となっている。これはキャリブレーションによってキーの高さが小さくなりすぎることを避けるためである。Meyboard は画面を占有する領域が小さくなるように設計された。その結果、Meyboard は Leyboard に比べてその高さが著しく小さくなった。 Meyboard のキーは Leyboard と異なり、格子状に並べられた配置となっている。これは 4.2.2 節にて述べるが、Meyboard では縦方向のフリック入力があり、その際にキーが格子状に整列しているとフリック時にユーザの指が他のキーにはみ出しにくくなると考えたためである。

キーの幅は隣のキーと重ならない範囲においてなるべく大きくなるように決定される。Meyboard では隣り合う指の水平方向の距離に等しい長さの線分を垂直線にて二等分し、2 つの垂直線間の距離をキーの幅とする。左端または右端のキー（q、a、p、クエスチョン）は Meyboard の左端または右端から垂直線までの距離をキーの幅とする。人差し指のキー（r、v、t、b、y、n、u、m）は r、v、u、m キーのみを幅 16mm の固定値とした。これは、人差し指から遠い側に位置する t、b、y、n キーに人差し指が届くようにするためである。

4.2 各キーの入力方法

Leyboard では修飾キーを除くほぼ全てのキーがタップによって入力可能であったが、中段キー等が省略された Meyboard では入力方法としてタップに加えて、フリック、Shift 入力、二段フリック、複数の指による入力を加えている。

4.2.1 タップによる入力

Meyboard では上段と下段のキーをタップによって入力する。ユーザの指がタッチパネルに置かれてから 450 ミリ秒以内に離れた場合、タップを行ったと解釈する。450 ミリ秒の値は TapBoard におけるタイピングと待機状態を分ける閾値をそのまま採用した。

4.2.2 フリックによる入力

Meyboard では中段のキーの入力にフリックを用いる。ユーザが上段か下段のいずれかのキーにおいてフリック入力を行った場合に、中段のキーが入力される。ユーザの指がタッチパネルに触れ、そのまま 3.4mm 以上動き、かつ移動量が 3.4mm を超えたときから 500 ミリ秒以内に画面から離れたときに、フリックを行ったと解釈する。フリックでは指の移動があるため、タップよりも長い時間を閾値としている。

物理キーボードにおいては、ユーザはキーに指を置いた状態（待機状態）からの入力が可能である。Meyboard でもユーザの手元注視を減らすために待機状態からの入力を可能にした。待機状態からの入力を可能にすると、手の位置が安定するため、手元注視を減らすことができると考えた。中段のキーの入力にフリックを用いるのは待機状態からの入力を実現する手段でもある。

フリックする方向は左方向でなければ任意の方向で良い。左方向のフリックは後述する Shift 入力として扱われる。

4.2.3 Shift 入力

Meyboard では Shift 入力が上段や下段のキーの大文字の入力として用いられる。Shift 入力とは左方向へのフリック入力である。ここで、左方向を以下のように定義する。フリックに

おいて、ユーザの指の移動量が閾値の 3.4mm を超えた瞬間における指の位置から、指が画面から離れた位置を結ぶベクトルと、画面の幅方向に平行な、右端方向へのベクトル（大きさは任意）とのなす角を θ とする。θ の値が 3/4π から π、あるいは-π から-3/4π の時を左方向へのフリックと解釈する。

なお、中段キーの入力を上下方向のフリックによって行うことを想定していたため、左方向へのフリックに Shift 入力を割り当て、上下方向のフリックと区別した。実装上は右方向へのフリックも中段キーの入力と解釈される。

4.2.4 二段フリック

Meyboard では中段のキーの大文字の入力に二段フリックを用いる。二段フリックはフリックと Shift 入力の動きを組み合わせた動きによる入力である。ユーザの指がタッチパネルに触れ、そのまま左方向以外に 3.4mm 以上動き、さらに左方向に 3.4mm 以上動く、あるいは左方向に 3.4mm 以上動き、さらに左方向以外に 3.4mm 以上動く、その後、画面から離れるという動きになる。この二方向の動きの組み合わせにおいて、最初の動き、2 番目の動きのそれぞれを 500 ミリ秒以内に行ったときに二段フリックを行ったと解釈される。

二段フリックは他の入力との誤入力を避けるために採用した。

4.2.5 複数の指による入力

Meyboard では複数の指を同時に用いることにより、入力を行うキーを設けた。

Meyboard にて複数の指を同時に用いる入力を取り入れたのは、Meyboard のキーの数の少なさのためである。Meyboard は中段のキーがない 20 個のキー群によって構成される 2 つのキーセットを用意している。そのため、キーは高々40 個しか存在しない。入力できるキーを増やすためにフリック入力を加えているが、単一の指による入力のみですべてのキー入力を賄うのは困難である。また、そのようなキー配列は QWERTY 配列から著しく離れたものになり、ユーザの学習コストが高くなる問題がある。加えて、Space や Enter のようなアルファベットでも数字でも記号でもないが、使用頻度が非常に高いキーは常に入力できるようにする必要がある。このようなキーがいずれかのキーセットにのみにしか存在しない場合、入力において要求されるキーセット切り替えの回数が著しく増大し、不便である。そこでいずれのキーセットにおいても入力が可能であり、かつその入力が 20 個のキーのいずれかを占有することがない、複数の指を同時に用いる入力を取り入れた。

複数の指を同時に用いるキーの入力方法を表 4.1 に示す。表 4.1 では入力に用いる指と用いない指をそれぞれ黒丸と白丸によって示している。左手の 4 つの丸は左から小指、薬指、中指、人差し指であり、右手の 4 つの丸は左から人差し指、中指、薬指、小指である。入力時に求められるユーザの指の動作は、タップまたはフリックである。

複数の指による入力を行う際に、ユーザは最初の指を画面に触れさせてから 120 ミリ秒以内に入力に用いる指すべてをタッチパネルに触れさせ、620 ミリ秒以内に画面から離す必要が

ある。120 ミリ秒の数値は以下の考えに基づいて設定した。Leyboard の実験において、十分に慣れた参加者は平均 49.69wpm の入力率であった。Meyboard が理想的には同等の 50wpm の入力率となると想定する。すると Gentner による入力率の定義より、単語の入力率を 5 倍すると文字の入力率となるので、1 文字を入力するのにかかる時間は $60 \times 1000 \div (50 \times 5) = 240$ ミリ秒となる。つまり、ユーザの入力率が高々50wpm であるならば、最初の指が画面に触れてから 240 ミリ秒以内の別の指の画面への接触を最初の指との同時入力と解釈しても問題はないと考えられる。但し、Leyboard の入力率の最大値は 61.27wpm であり、ユーザや入力内容によっては瞬間的にさらに高い入力率になった可能性も考えられる。そこで、Meyboard において想定される瞬間最大入力率を 100wpm とすると、$60 \times 1000 \div (100 \times 5) \approx 120$ ミリ秒が 1 文字の入力に要する最短の時間である。よってこの値をすべての指をタッチパネルに触れさせるまでの閾値とした。620 ミリ秒は 120 ミリ秒にフリックの閾値 500 ミリ秒を加算することによって求めている。

表 4.1: 複数の指による入力（黒丸は入力する指）

左手	右手	動作	入力内容
●●○○	○○○○	タップ	全角/半角
○●●○	○○○○	タップ	Tab
○○●●	○○○○	タップ	Space
○○●●	○○○○	フリック	キーセット切り替え
○○○○	○○●●	タップ	Enter
○○○○	○●●○	タップ	Back Space
○○○○	●●●○	タップ	Delete
○○○○	●●○○	フリック	矢印

キーセット切り替え

Meyboard には 2 種類のキーセットが半角、全角文字それぞれにおいて設けられている。1 つはアルファベットセット（図 4.1、図 4.3）であり、もう 1 つは数字・記号セット（図 4.2、図 4.4）である。Meyboard には Leyboard に存在した Fn・テンキーセットが存在しない。Fn・テンキーセットに含まれるキーは文字入力において必ずしも必要ではないため、Meyboard では除外することとした。

Meyboard の各キーセットにおけるキーの配置は Windows 8 キーボード（第 3 章図 3.21 参照）を参考にして決定した。

2 種類のキーセットを切り替えるために、左手人差し指と中指のフリック入力を用いる。これは Meyboard では親指を使わないこととしたためである。Leyboard の親指基点スライドは手の形を保ったままの入力をユーザに可能にしていた。しかし、前提として手全体を動かすことを前提としたデザインになっているため、親指基点スライドが終了し、キーの位置が元

に戻る際に手の位置を調整し直す必要があった。Leyboard と同様にキーセットの切り替えを採用した Meyboard では、手全体を動かすことなしにキーセットの切り替えが行えるように、フリック入力によってキーセットを切り替えられるようにした。なお、キーセット切り替えには Shift 入力に相当する入力がないので、キーセットを切り替える際のフリック方向には一切の制約を設けていない。

4.3 実験

Meyboard の入力性能と手元注視を減らす可能性について検証するため、Meyboard を手元注視に制限を設けない場合と、手元が全く見えない場合において、それぞれの入力性能を調査する実験を行った。Meyboard のキーの高さは全て 30mm であった。Meyboard のキーの幅はキャリブレーションの結果によって変わるため不定であるが、r、v、u、m キーのみ固定されており、その値は 16mm であった。Meyboard 全体の高さは 6cm、幅は 31cm であった。

4.3.1 実験環境

Meyboard を動作させるデバイスとして、Leyboard の実験時と同様に Acer 社の ICONIA-F54E を用いた。ICONIA-F54E を使用した実験環境を図 4.5 と図 4.6 に示す。図 4.5 は手元注視に制限を設けない場合（注視可能条件）であり、図 4.6 は手元が全く見えない場合（注視不可能条件）である。後者においては、手元を布で覆うことにより手元が見えないようにした。参加者の左方には物理キーボードが設置されている。これは、参加者が Meyboard におけるキーとそれを入力する指の対応が分からなくなった時に、物理キーボードを見ることによってそれを思い出せるようにしたためである。

4.3.2 参加者と実験課題及び実験期間

本実験の参加者は男性の 22 歳から 25 歳の合計 3 名からなる。

実験課題は英字の入力であり、練習セッションと本番セッションが存在した。各セッションでは、参加者に英文はまたは意味のない文字及び単語の羅列の入力を 10 回行ってもらった。

参加者はまず注視可能条件において練習セッションを 1 セッション、本番セッションを 2 セッションの入力を行った。続けて、注視不可能条件において同様の入力を行った。以上の内容を 2014 年 1 月 8 日から同年 1 月 10 日までの 3 日間にわたって毎日行った。但し、ユーザが Meyboard に習熟しているとはいいがたい初日はユーザの疲労や実験に要する時間を考慮し、各条件において練習セッション及び本番セッションを 1 セッションずつのみ行う形式にした。結果、参加者は本番として 5 セッション分の入力を行った。

練習タスクにおいては以下の内容の入力を行った。

- A quick brown fox jumps over the lazy dog.
- Cwm fjord veg balks nth pyx quiz.

図 4.5: 実験環境（注視可能条件）

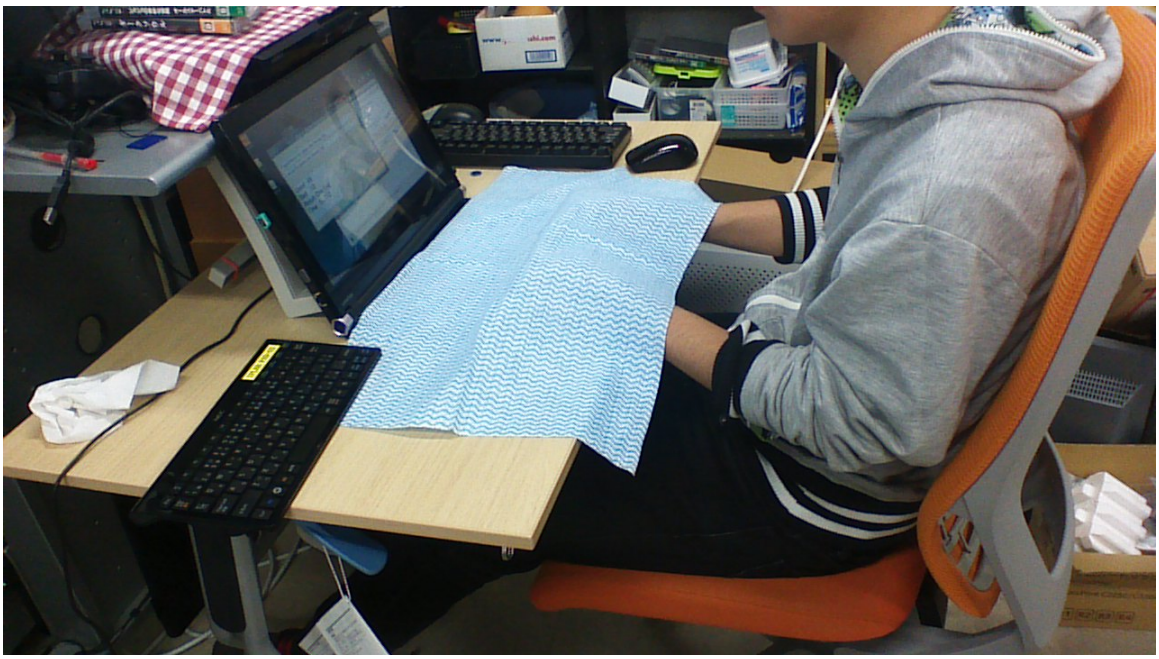

図 4.6: 実験環境（注視不可能条件）

- Pack my box with five dozen liquor jugs.
- Jackdaws love my big sphinx of quartz.
- Jumbling vext frowzy hacks PDQ.
- rtfgvb alks dk yuhjnm jfgh.
- asdf jkl ruty fjgh vmbn.
- RTFGVB ALKS DK YUHJNM JFGH.
- ASDF JKL RUTY FJGH VMBN.
- Daft skull god mat envy flick had burnt duke jar.

上記は、全てのキーの入力を行うようにパングラム 5 つと、Meyboard におけるフリック入力、Shift 入力、二段フリック入力、そして担当するキーの数が多い左右の人差し指による入力が多くなるように設定された意味のない文字や単語の羅列 5 つからなる。なお、これらを入力する順番は練習セッションごとにランダムに設定された。練習セッションは参加者が Meyboard の入力の練習を行う機会として設けられており、その入力結果は実験結果においては考慮されない。

本番セッションにおいては MacKenzie らによる文章セット [MS03] を加工し、最初の文字及び一人称の “I” を大文字にして最後にピリオドを加えたものを入力対象とした。セッションごとに、ここから 10 文がランダムに選択され、ユーザはその入力を行った。

今回の実験においても、Leyboard の実験時と同様に、入力を間違えた場合、参加者はそこから正しい入力をやり直す必要がある。言い換えれば、参加者が正しい入力を行わない限りタスクは進まない。そのため、参加者は最終的には正しい文字を入力することになる。

手元を隠す本実験においては、ユーザの意図しないキーセット切り替えがユーザの誤入力によって発生した場合、課題を完了することが不可能になる可能性がある。何故ならば、参加者はキーセットが切り替わったことを知る手段がないからである。そこで本実験においては空白の入力以外の複数本の指を用いた入力をできないようにした。

ユーザは実験中に何回キャリブレーションを行ってもよいこととした。特に、注視不可能条件においては頻繁にキャリブレーションを行うことが期待された。注視不可能条件時には、自分の指がキーの位置に合わることができているかを参加者が知る手段がないためである。そこで参加者はキャリブレーションを行うことにより、適宜キーの位置を指に合わせる必要があった。そのため、実験中においてキーの位置は頻繁に変動した。また、各セッションを開始する前にも参加者はキャリブレーションを行った。

4.3.3 結果

注視可能条件における入力率の変動を図 4.7 に、注視不可能条件における入力率の変動を図 4.8 に示す。いずれの条件においても入力率は上昇傾向にあった。

注視可能条件における参加者ごとの入力率の平均値は参加者 A が 14.69wpm、参加者 B が 17.95wpm、参加者 C が 14.45wpm となった。注視不可能条件における参加者ごとの入力率の平均値は参加者 A が 14.58wpm、参加者 B が 14.85wpm、参加者 C が 12.42wpm となった。い

ずれの参加者においても、注視不可能条件の入力率は注視可能条件よりも低くなった。

注視可能条件における参加者ごとの入力率の最大値は参加者 A が 19.32wpm、参加者 B が 20.80wpm、参加者 C が 16.40wpm となった。これは全員とも実験 3 日目（最終日）に計測された。

注視不可能条件における参加者ごとの入力率の最大値は参加者 A が 18.10wpm、参加者 B が 20.68wpm、参加者 C が 16.54wpm となった。これは全員とも実験 3 日目（最終日）における本番タスク第 2 セット目（実験全体における最終タスク）に計測された。

注視可能条件における参加者ごとの入力率の最小値は参加者 A が 11.48wpm、参加者 B が 12.53wpm、参加者 C が 11.56wpm となった。これは全員とも実験初日の本番タスク第 1 セット目に計測された。

手元注視不可能条件における参加者ごとの入力率の最小値は参加者 A が 11.24wpm、参加者 B が 9.02wpm、参加者 C が 9.02wpm となった。これは全員とも実験初日の本番タスク第 1 セット目に計測された。

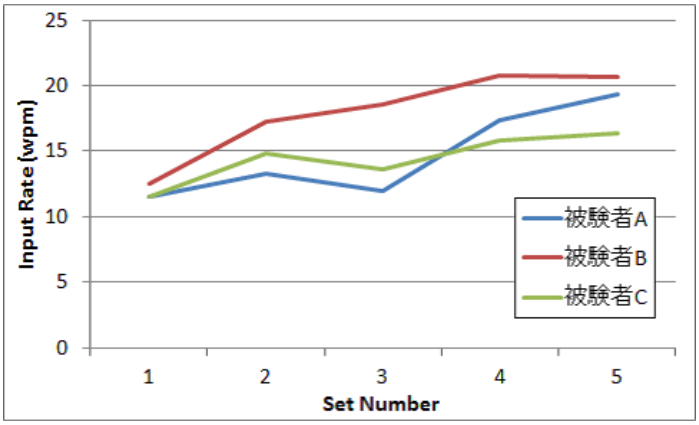

図 4.7: Meyboard の入力率（wpm）（注視可能条件）

注視可能条件におけるエラー率の変動を図 4.9 に、注視不可能条件におけるエラー率の変動を図 4.10 に示す。注視可能条件においては参加者 C を除いてエラー率はやや上昇傾向にあった。注視不可能条件においては参加者 A を除いてエラー率は下降傾向にあった。

注視可能条件における参加者ごとのエラー率の平均値は参加者 A が 5.74%、参加者 B が 7.10%、参加者 C が 5.62%となった。注視不可能条件における参加者ごとのエラー率の平均値は参加者 A が 12.53%、参加者 B が 18.77%、参加者 C が 20.94%となった。いずれの参加者においても、注視不可能条件のエラー率は注視可能条件よりも非常に高くなった。

注視可能条件における参加者ごとのエラー率の最大値は参加者 A が 8.52%、参加者 B が 8.63%、参加者 C が 7.79%となった。これは実験初日または 2 日目に計測された。

注視不可能条件における参加者ごとのエラー率の最大値は参加者 A が 15.14%、参加者 B が 23.15%、参加者 C が 27.55%となった。これは参加者 C を除いて、実験初日の本番タスク第 1 セット目に計測された。

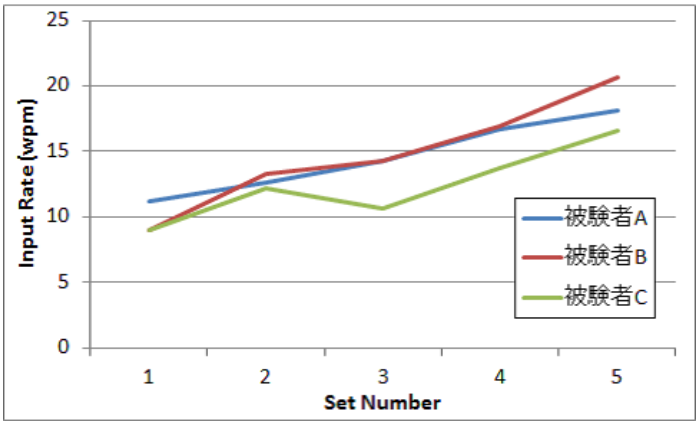

図 4.8: Meyboard の入力率（wpm）（注視不可能条件）

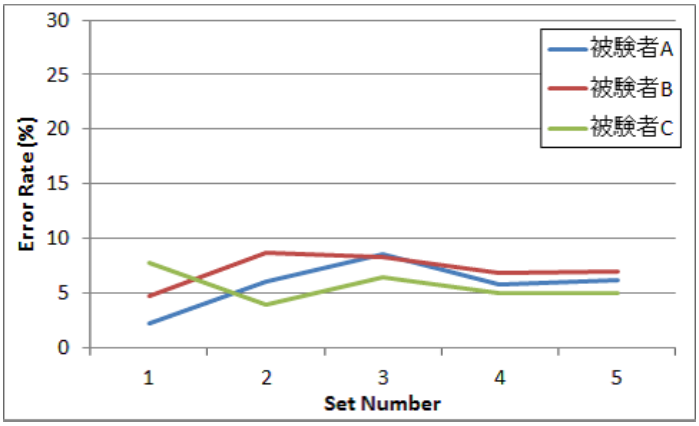

図 4.9: Meyboard のエラー率（注視可能条件）

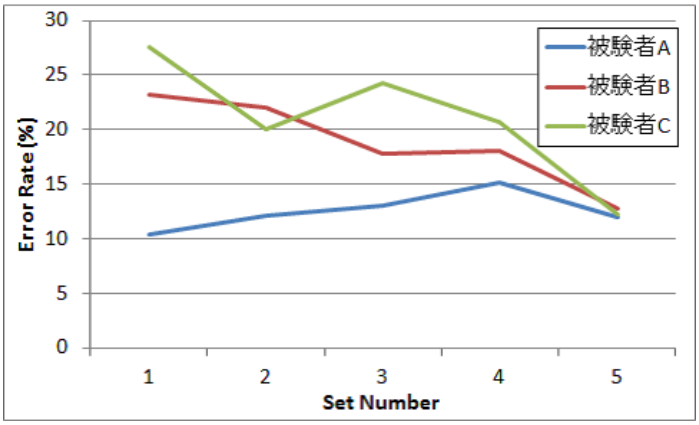

図 4.10: Meyboard のエラー率（注視不可能条件）

注視可能条件における参加者ごとのエラー率の最小値は参加者 A が 2.24%、参加者 B が 4.75%、参加者 C が 3.91%となった。これは参加者 C を除いて、実験初日の本番タスク第 1 セット目に計測された。

注視不可能条件における参加者ごとのエラー率の最小値は参加者 A が 10.41%、参加者 B が 12.74%、参加者 C が 12.20%となった。これは参加者 A を除いて、実験 3 日目（最終日）における本番タスク第 2 セット目（実験全体における最終タスク）に計測された。

4.3.4 考察

本節では、Meyboard の注視可能条件と注視不可能条件のそれぞれにおける入力性能を比較し、Meyboard の設計の妥当性や問題点を検証する。

条件ごとの数値の差

Leyboard と Windows 7、それぞれのキーボードの入力率とエラー率の全データに関して、それぞれ平均値に有意な差があるかを確かめるために、対応のある t 検定を行った。その結果、注視可能条件と注視不可能条件の入力率に有意差はないことが判明した。

- 参加者 A: $t = 0.1828, p = 0.8638 > 0.01$
- 参加者 B: $t = 3.8737, p = 0.01794 > 0.01$
- 参加者 C: $t = 3.6405, p = 0.02196 > 0.01$

つまり、Meyboard は手元注視を行わなくても手元注視可能時と同等の入力率があると言える。一方、エラー率においては注視不可能条件が注視可能条件に対して有意に高いことが判明した。

- 参加者 A: $t = -7.7544, p - value = 0.00149 < 0.01$
- 参加者 B: $t = -5.5762, p - value = 0.00507 < 0.01$
- 参加者 C: $t = -7.0923, p - value = 0.002087 < 0.01$

つまり、Meyboard は手元注視を全く行わなかった場合、手元注視可能時と比べてエラー率が高くなる。

以上の結果から Meyboard は手元注視を行わなくても手元注視可能時と同等の入力率があるが、その際は手元注視可能時と比べて多量のエラーが発生する。本実験ではエラーをもたらした入力に要した時間は入力率に影響を与えた（正しい入力を行わない限りタスクを進めることができない）が、エラーをもたらした入力を消すという行程が存在しなかった。そのため間違えた入力を消すのに要する時間が入力率に考慮されていない。よって、エラー修正にかかる時間を考えると実際の入力率は本実験にて測定された値よりも低くなると思われる。しかしながら、それは注視可能条件と注視不可能条件双方において成立するため、結果として入力率はやはり同等になると考えられる。

Meyboard において、ユーザが Meyboard の手元注視を減らしたとしても、それが入力率に支障をきたすとは言えない。よって、Meyboard におけるタッチタイピングの実現性はあるが、一方で Meyboard は手元注視なしではエラー率が高いので、エラー率を抑えるためにはある程度手元を注視する必要はある。実験が進むにつれて 3 人中 2 人の参加者にエラー率が下がる傾向があり、参加者全員に入力率向上の傾向があったので、ユーザの習熟につれて Meyboard の入力性能は向上する可能性がある。注視不可能条件のエラー率が注視可能条件のエラー率と同等まで下がるのであれば、Meyboard の入力に手元注視は影響しないことになる。

エラー内容分析

Meyboard のエラー率を軽減する方法を考察するためにエラーの内容を分析した。

Leyboard の実験時と同様に、実験途中の全てのキー入力内容に対してログを取った。そこで上記ログを参照し、Leyboard の実験時と同様のアルゴリズムに基づいて条件ごとに誤字と脱字の判定を行った。

参加者のエラーを合計すると、注視可能条件では 223 回、注視不可能条件では 590 回となった。その内訳は以下のとおりである。注視可能条件では 193 回の誤字と 30 回の脱字が存在した。注視不可能条件では 559 回の誤字と 31 回の脱字が存在した。エラーの回数は注視不可能条件が圧倒的に多いが、脱字に関してはほぼ差がない。手元注視の条件別にエラー回数を文字別に分けてグラフ化した。誤字に関しては図 4.11 に、脱字に関しては図 4.12 に示す。

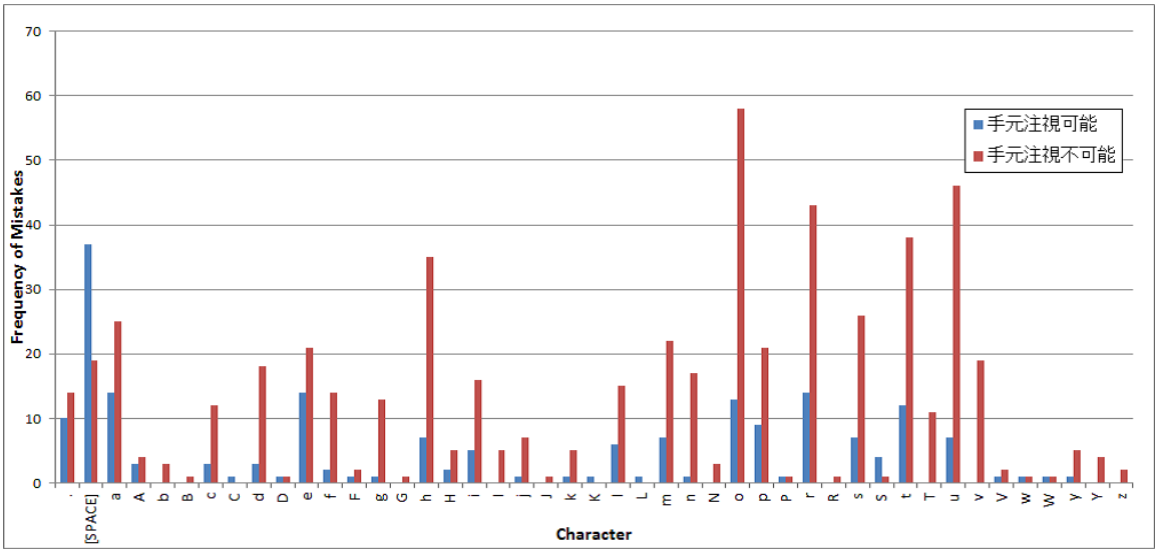

図 4.11: 誤字の分析結果

表 4.2: 注視可能条件時と注視不可能条件時のエラー数と発生率

エラー	注視可能		注視不可能	
	数	発生率	数	発生率
j → h	0	0.00%	5	41.67%
v → b	0	0.00%	16	40.00%
u → y	6	5.41%	44	38.60%
T → R	0	0.00%	8	20.51%
m → ,	4	4.44%	16	18.18%
p → o	7	12.96%	15	17.24%
o → i	8	2.61%	52	16.30%
h → j	0	0.00%	17	12.23%
S → s	4	40.00%	1	11.11%
g → t	0	0.00%	8	10.96%
r → e	6	2.60%	21	9.05%
f → g	1	1.45%	6	8.45%
h → n	0	0.00%	11	7.91%
r → t	1	0.43%	16	6.90%
. → ,	7	4.67%	10	6.67%
d → c	1	0.93%	7	6.54%
s → w	0	0.00%	13	5.63%
d → s	2	1.87%	6	5.61%
c → d	0	0.00%	6	5.45%
n → m	0	0.00%	13	5.42%
l → .	1	0.85%	6	5.04%
p → ?	0	0.00%	4	4.60%
c → v	1	1.05%	5	4.55%
l → k	2	1.71%	5	4.20%
i → u	2	0.79%	10	3.83%
d → e	0	0.00%	4	3.74%
a → q	5	1.92%	9	3.61%
a → z	0	0.00%	7	2.81%
e → w	1	0.24%	11	2.50%
s → a	1	0.48%	5	2.16%
s → x	2	0.95%	5	2.16%
a → s	0	0.00%	5	2.01%
e → r	0	0.00%	5	1.14%
Space → r	24	3.59%	6	0.91%
Space → v	4	0.60%	3	0.45%
a → n	4	1.53%	1	0.40%
e → n	5	1.19%	0	0.00%

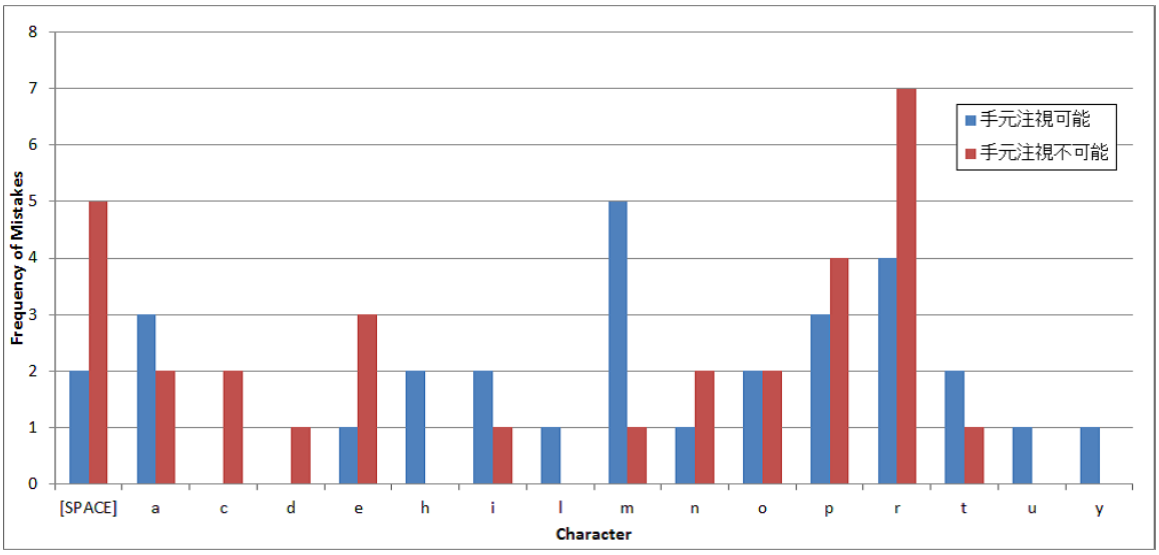

図 4.12: 脱字の分析結果

誤字内容の分析

以降、○を△と間違えるエラーを○→△と表し、両条件におけるエラーの発生率を 表 4.2 に示す。エラーの発生率は

$$\frac{\text{エラー数}}{\text{間違えた文字の出現回数}}$$

として計算される。例えば、w → e の発生率は w → e の数を w の出現回数にて割ったものである。エラー数ではなく、発生率を評価対象とするのは、注視可能条件時と注視不可能条件時それぞれの入力文字数が異なることと、母音（a、e、i、o、u）のように出現頻度が高いゆえにエラー数が多い文字を間違えやすい文字と混同することを避けるためである。なお、 表 4.2 では全体のパフォーマンスに与える影響の小ささから双方のエラー数が 3 回以下のエラーを除いている。

表 4.2 を見ると、全般的に注視不可能条件時のほうがエラーの発生率が高い傾向にある。同じキーボードに対して、手元を見ない方がキーの位置を把握しにくいため、この結果は妥当である。しかしながら、注視不可能条件では発生率が 10%を超えているエラーが多数存在しており、タッチタイピングを行うソフトウェアキーボードとして実用的であるとは考えにくい。

注視不可能条件時においては j → h、v → b、u → y、T → R、m →コンマなど列方向に隣り合うキーと間違えるエラーが多い。次に多いエラーとして、g → t、h → n、d → c、s → w、l →ピリオドなどフリックするべきところをタップしたために生じたエラーが見られる。他には、行方向に隣り合うキーと間違えるエラーである p →クエスチョン、タップすべきところをフリックしたために生じた c → d、Space の入力時に人差し指のみの入力となった Space → r が見られる。

注視不可能条件時において発生率が上位である j → h、v → b、u → y、T → R、m →コンマはすべて人差し指によって入力するキーのエラーである。人差し指が担当するキーのエラーの発生率が特に高くなった理由は二つ考えられる。一つは人差し指の担当するキーが 2 列存在したこと、もう一つは r、v、u、m キーが固定幅であったためである。他の列のキーは指の上下方向の移動により入力が可能であるが、人差し指は 2 列分を入力するために指を左右方向にも動かす必要があった。加えて、r、v、u、m キーは固定幅であったため、キャリブレーション時に必ずしも人差し指の位置にあるとは限らなかった。よって、参加者はキャリブレーションを行っても r、v、u、m キーの位置を把握することができず、人差し指のキーのエラーの発生率が増えることにつながったと考えられる。事実、参加者の全員が人差し指のキーの入力のしにくさに関してコメントを残している。以下に要約する。

- 人差し指によって入力するキーがすべて入力しにくい。
- v が圧倒的に入力しにくい。
- r、f、v、u、j、m が入力しにくい。

元々、t、b、y、n キーに人差し指が確実に届くようにするためにこれらのキーの幅を固定値としていたが、ユーザに合わせてこれらのキーの幅を確保し、キャリブレーション時に人差し指の下に位置するようにすることが必要と考えられる。

Meyboard ではユーザの指の位置に合わせてキーの幅を調節するキャリブレーションを行っているが、人差し指のキーに限らず列方向に隣り合うキーと間違うエラーの発生率が高いことを考えるとキャリブレーションが有効に作用していたとは考えにくい。これは、第 3 章にて見られたように、手のひらの位置を固定した状態においてもタッチ位置のずれが生じるためと思われる。これを解決する方策の一つとしては、キャリブレーションを行う回数を増やすことが考えられる。キャリブレーションにより、手がずれた状態に合わせてキー配列を更新し続けることにより、エラーの発生率を抑えられると考えられる。また、p →クエスチョンのように行方向に隣り合うキーと間違えるエラーも見られるため、キーの幅のみならず高さ方向の大きさもキャリブレーションによって調節することが望ましいと考えられる。

第5章　自動適応型減数ソフトウェアキーボード

Meyboard におけるエラー数を減らすために、キャリブレーションを行う回数を増やすこととした。キャリブレーションの実施をユーザが逐次かつ意図的に行うことはユーザにとって文字入力以外の要素の入力が増えることとなり、煩雑である。そこで、入力時にキャリブレーションが自動的に行われるようにすることを考え、自動適応型減数ソフトウェアキーボードである Meyboard Ⅱ を実装した。

Meyboard Ⅱ では誤入力を減らすために QWERTY 配列における下段のキーを省略しかつユーザの手に適応するようにキーの幅と高さを決定している。下段のキーを省略したのは、これらのキーが QWERTY 配列において最も使用頻度が低いためである。Hiraga らによると QWERTY 配列の物理キーボードにて英文を入力する際、各行の使用率は上段が 51.5%、中段が 31.8%、下段が 17.7%であり [HOY80]（図 5.2）、下段のキーの使用頻度が最も低いことが分かる。また、The Oxford Math Center が公表する英語におけるアルファベットの使用頻度 [The] に基づき、QWERTY 配列の行単位にて使用頻度を求めると（図 5.3）、同様に上段が 51.3%、中段が 34.0%、下段が 14.6%と下段のキーの使用頻度が最も低いことが分かる。Meyboard Ⅱ では使用頻度の高い上段と中段のキーは入力しやすいタップによる入力を、使用頻度の低い下段のキーは省略して、タップの代わりにフリックによる入力を行うこととした。キーを省略したこととキーの幅と高さをユーザの手に合わせることにより、ユーザは手元を見なくても Meyboard Ⅱ による入力が可能となる。

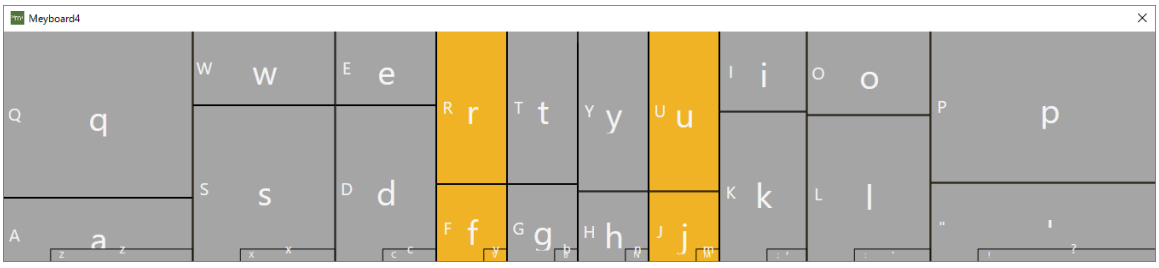

図 5.1: Meyboard Ⅱ のキー配置

5.1　キャリブレーション

Meyboard Ⅱ では起動時のキー並びが、キーの高さがすべて等しい格子状となっているが、ユーザが片方の手の 4 本の指をタッチパネル面に置いた際に、キーボードの片側のキーの位

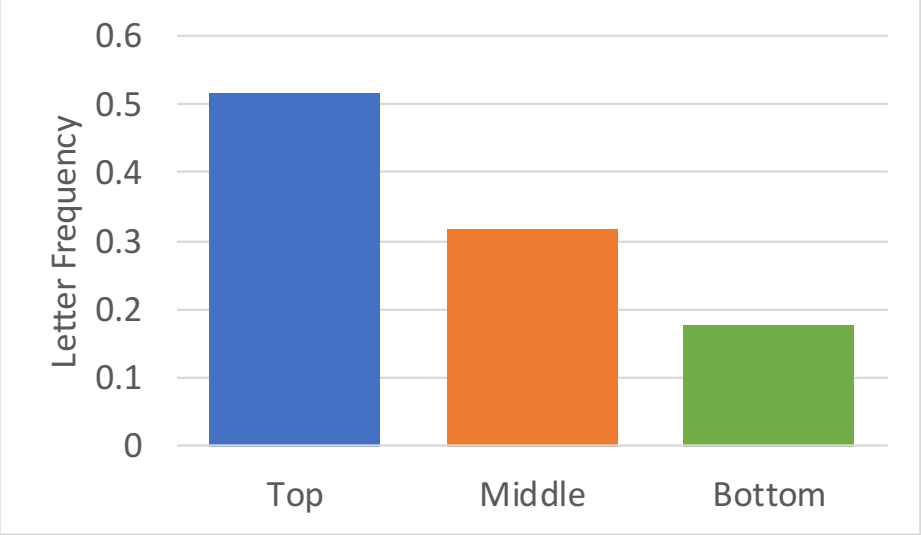

図 5.2: 英語入力時の QWERTY 配列の各行の使用頻度（Hiraga らによる）

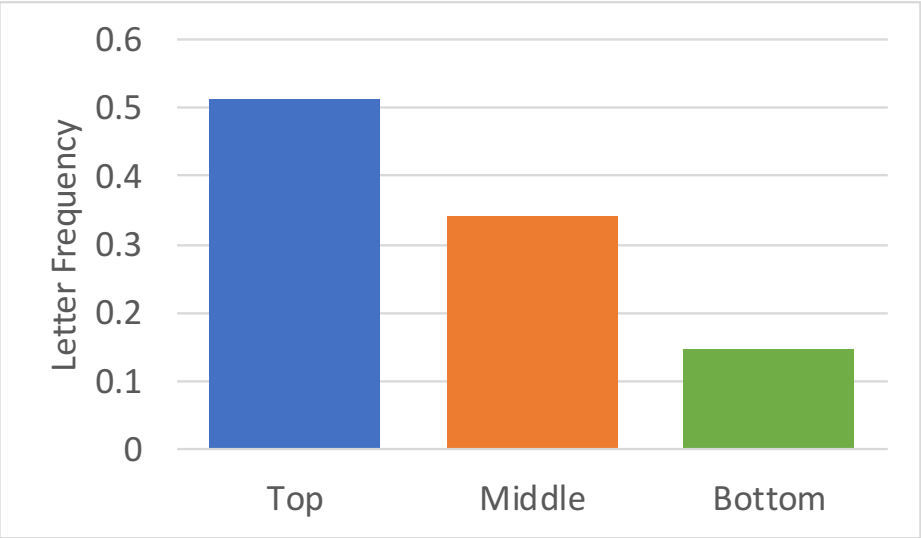

図 5.3: 英語入力時の QWERTY 配列の各行の使用頻度（The Oxford Math Center による）

置と大きさをユーザに適応させる（図 5.4）。この 4 本の指を置いてキーの幅と高さを調節する機能が Meyboard Ⅱ のキャリブレーションとなる。

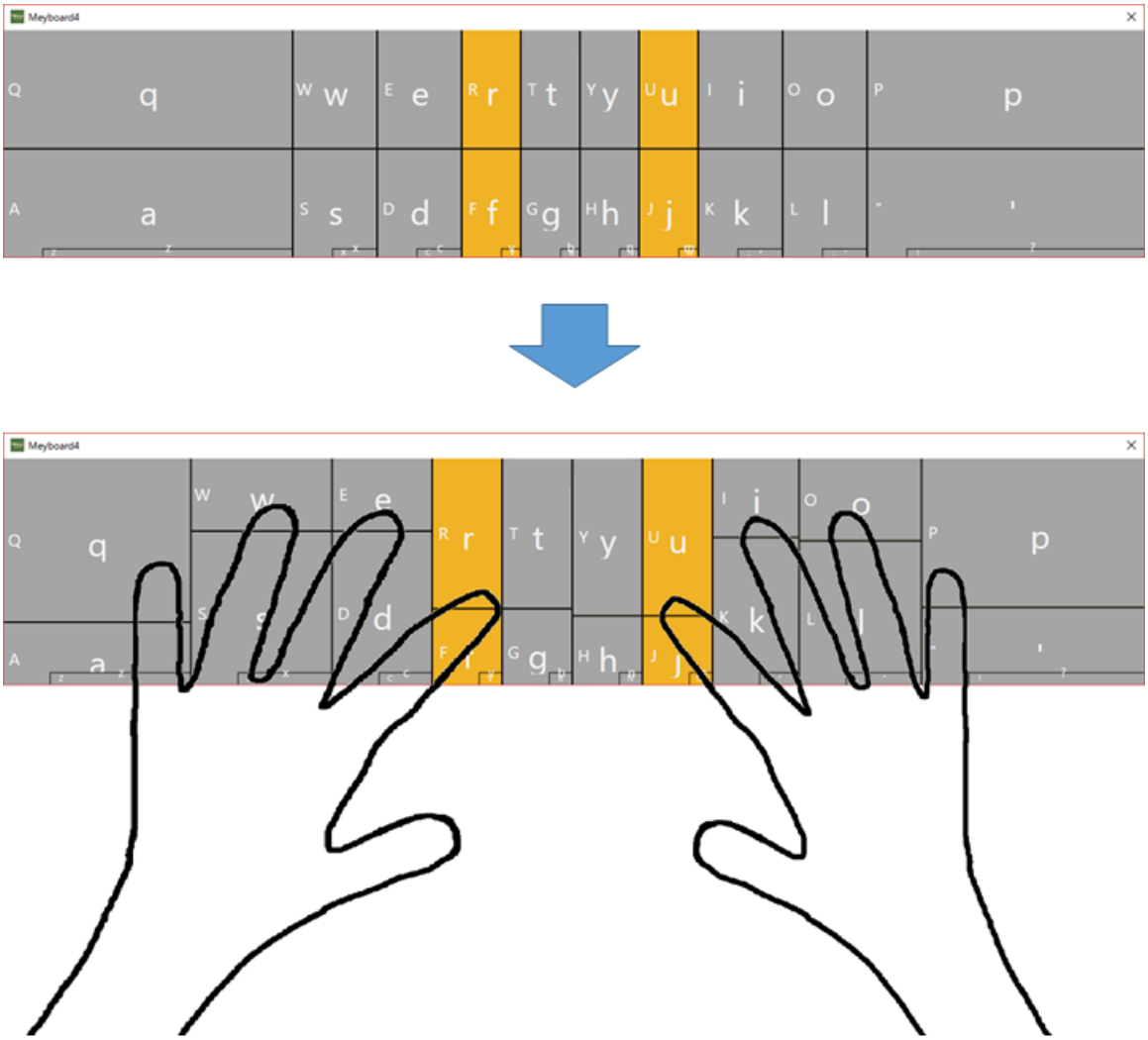

図 5.4: Meyboard Ⅱ のキャリブレーション

ユーザが入力を行う際に、その手の位置は元の位置からずれていくことが予想されるため、キャリブレーションを定期的に行うことが望ましい。ここで重要なのは、入力途中において、ユーザが意識することなくキャリブレーションが行われることである。ユーザが意識的にキャリブレーションを行うキーボードでは、キャリブレーションを行う際に一旦ユーザの入力が中断され、結果としてユーザの入力時間を長くする。

キャリブレーションを自動的に行う方法として、物理キーボードにおいて頻繁に入力するキーの入力を Meyboard Ⅱ のキャリブレーションに割り当てることを考えた。頻繁に入力するキーとして、Shift、Space、Enter、BackSpace を選択した。 何故ならば、Shift は文頭の大

文字の入力、Space は空白の入力のために 1 文を入力する際に必ず入力され、Enter の改行、BackSpace の文字消去は Shift と Space ほど高頻度ではないにしてもしばしば入力されるからである。 Meyboard Ⅱ ではこれらのキーをユーザが入力する際に、キャリブレーションを行う。

キャリブレーションは 4 本指の入力によって行われることとしたので、これらのキーの入力も 4 本または 8 本の指を用いる。左手 4 本指のタップ入力を Space、右手 4 本指のタップ入力を Enter、左手または右手 4 本指をタッチパネル面に置いたままにする入力を Shift、左手 4 本指をタッチパネル面に置いたまま、右手 4 本指をタップする入力（Shift と Enter の入力の組み合わせに等しい）を BackSpace に割り当てた。 Space が左手なのは、QWERTY 配列では左手の方が右手よりも使用頻度が高いため（[HOY80, The]）、左手のキャリブレーションを行う頻度を高くした方が望ましいと考えたからである。Space は単語を入力する度に入力するため、キャリブレーションに割り当てた 4 つのキーの中では最も入力頻度が高い。Shift が左右両方の手によって入力可能なのは、片側だけにするとそちら側のキーの大文字入力ができないためである。また、左手のキーの大文字入力は右手側の Shift、右手のキーの大文字入力は左手側の Shift の入力と決めることにより、右手側のキャリブレーションが生じるようにしている。Enter は Space を左手に割り当てたため、空いている右手に割り当てた。 BackSpace は 8 本の指を用いて入力することとしたため、入力時に Meyboard Ⅱ の全キーのキャリブレーションが行われる。これにより、ユーザが誤入力を行った際に、キーの幅と高さを自動的に修正し、以降のユーザの誤入力の増加を抑える狙いがある。

5.1.1 キーの幅と高さ

図 5.5 にキーの幅の決定方法を示す。Meyboard Ⅱ では Meyboard 同様に隣り合う指の水平方向の距離に等しい長さの線分（①〜③）を垂直線にて二等分し、2 つの垂直線間の距離をキーの幅とする。左端または右端のキー（q、a、p、シングルクォーテーション）を Meyboard Ⅱ の左端または右端から垂直線までの距離をキーの幅とする点も同様である。人差し指のキー（r、f、t、g、y、h、u、j）は Meyboard では固定値であったが、Meyboard Ⅱ では 2 本の人差し指（4 本指の時は、指が置かれていない側の手の指の位置として、前回のキャリブレーション時の値を用いる）の距離（④）に等しい長さの線分を二等分する垂直線から、中指のキーの端までの距離（⑤）を二等分する垂直線までの距離をその幅とする。但し、⑤が大きいと、人差し指の外側のキー（t、g、y、h）に人差し指が届きにくくなるため、⑤が③の 2 倍を超えるときは、内側のキー（r、f、u、j）の幅を③と同じにする。

次に、図 5.6 にキーの高さの決定方法を示す。キーの高さはユーザの指の置き方に適応しつつも、小さくなりすぎないように決定される。各指の位置から、キーの幅を決定した際の垂直線と交わるまで（人差し指では図 5.5 ④を二等分する垂直線まで）水平線（①〜④）を引く。垂直線と水平線に囲まれた範囲からキーの高さを決定すると、キャリブレーション時の位置によってはキーの高さが小さくなりすぎる可能性がある。そこで 4 本の水平線すべてを垂直線の方向に同じ距離だけ動かすことにより、キャリブレーション時の指の置き方を保ったままキーの高さ補正を行う。4 本の水平線のうち、最も高いものと最も低いものの距離に等しい長さの線分を二等分する水平線（⑤）と、Meyboard Ⅱ の高さを二等分する水平線（⑥）が

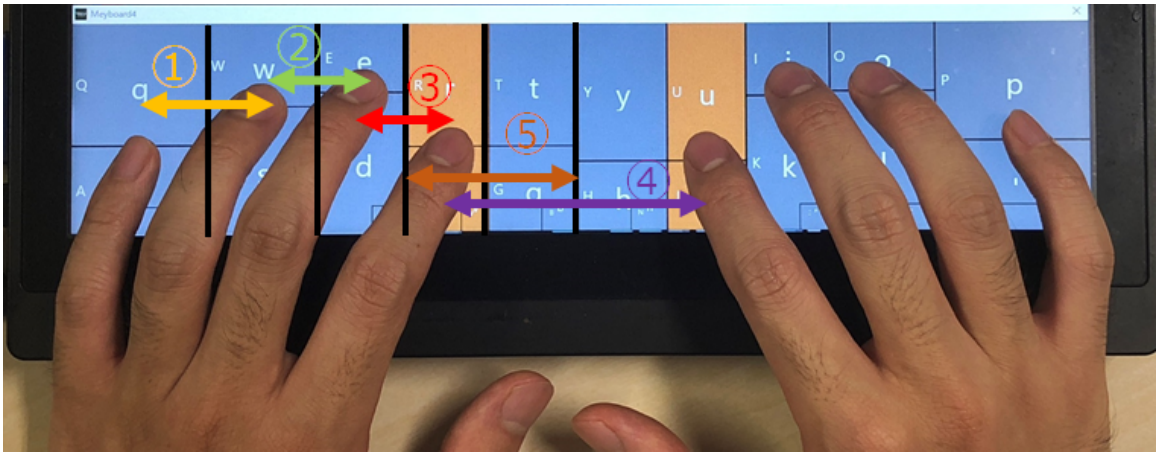

図 5.5: Meyboard Ⅱ のキーの幅の決定

等しくなる（①'〜⑤'）だけの移動量が高さ補正の移動量である。高さ補正により、キャリブレーション時の指の位置が高くても低くても、キーの高さが小さくなりすぎない（図 5.7）。

5.1.2　キャリブレーションにおける各指の推測方法

Meyboard Ⅱ ではキャリブレーション時に各タッチ点がユーザのどの指によってもたらされているのかを推測し、その結果に基づいてキーの幅と高さを決定している。その手法は 4 本の指のタッチ点の水平方向の座標（X 座標）を比較し、左手の場合は最も左にあるものから小指、薬指、中指、人差し指と解釈し、右手の場合は最も左にあるものから人差し指、中指、薬指、小指と推測する単純なものである。

Meyboard Ⅱ においては、上記の推測手法でも精度は良い。この手法が成立しないのは、ユーザの手の傾きが急であった場合である（例えば右手中指が右手人差し指よりも左側に位置することがある）が、キーボードの入力時にこのような姿勢をユーザが取ることは考えにくい。また、10 本指にて入力を行う場合、ユーザの手の傾きによっては親指が人差し指の左側にも右側にも来る可能性があるが、Meyboard Ⅱ では左右の手の人差し指から小指の 8 本のみを使用するため親指位置の考慮は不要である。

なお、タッチ点が左手によるものか、右手によるものかは指が触れたキーによって判断している。例えば、f キーに触れた場合は、左手による入力と解釈する。そのため、左手によって右手のキーを触れていた場合はキャリブレーションが正常に行われないことになるが、ユーザの左右の手が十分に離れていればこれは生じない。仮にそうなった場合は、左右の手をより離してもう一度キャリブレーションを行う必要がある。

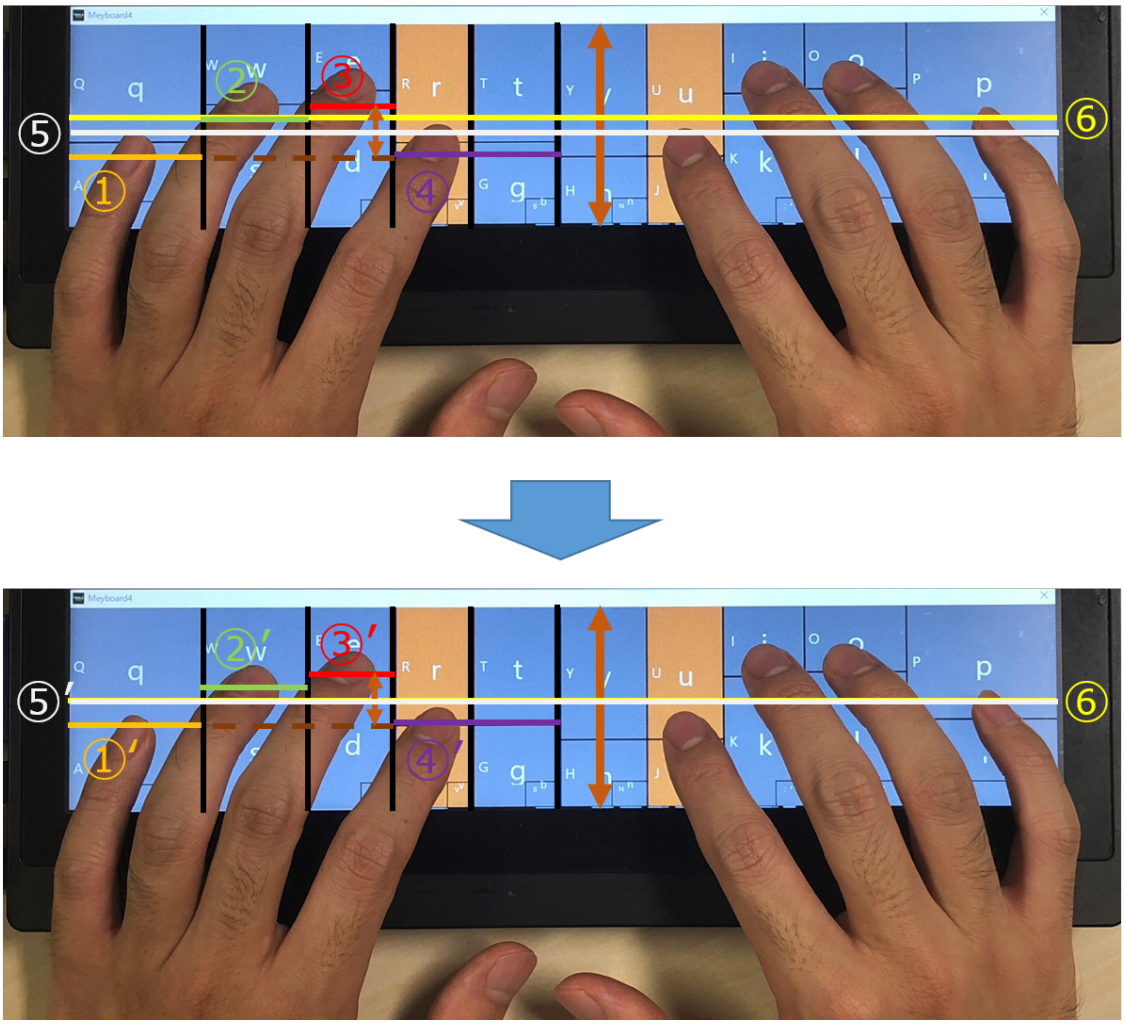

図 5.6: Meyboard II のキーの高さの決定

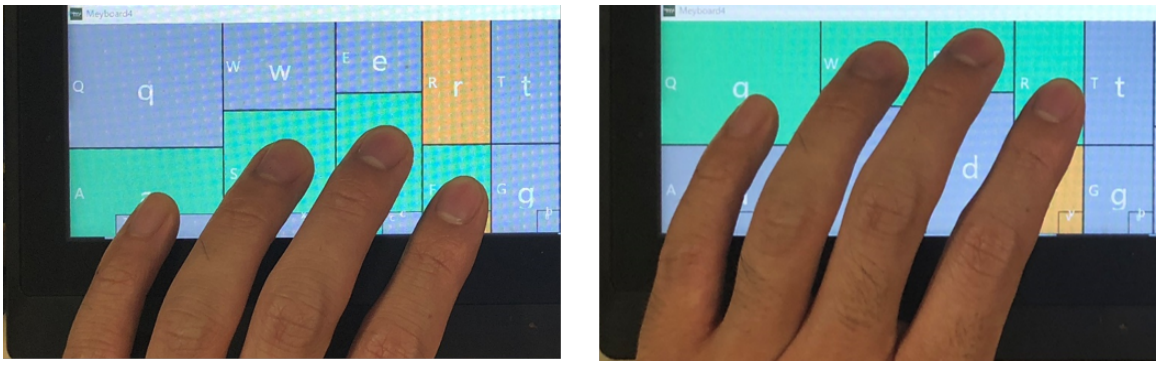

図 5.7: キーの高さ補正の例。キャリブレーション時の指の位置によらず、キーの高さが小さくなりすぎることを防いでいる。

5.2　各キーの入力方法

Meyboard II ではアルファベットと記号による 30 の文字を 20 個のキーにて入力可能にするために、QWERTY 配列におけるキーを複数のエリアに分けて、エリアごとにタップまたはフリックを入力方法として割り当てている（図 5.8）。

Home（a–f、j–l、シングルクオーテーション）、

UpperHome（q–r、u–p）
これらのキーはホームキーとその上に位置するキーである。これらのキーを入力するにはキーをタップすれば良い。タップは Meyboard II において最も入力時間が短い入力である。Home と UpperHome に含まれる文字は最も使用頻度が高いため（図 5.2、図 5.3）、タップを割り当てた。

UnderHome（z–v、m、コンマ、ピリオド、クエスチョン）　これらのキーはホームキーの下に位置するキーである。UnderHome は Meyboard II において省略されたキーである。これらのキーを入力するには、同じ行のキー（Home または UpperHome）をフリックする。例えば、ユーザが q または a キーをフリックした場合、z が入力される。フリックはタップと比較すると入力に時間がかかるが、ユーザが慣れている入力方法であると考える。加えて、UnderHome を含む下段は Home 及び UpperHome を含む上段、中段と比べて、使用頻度が低い（図 5.3）。なお、フリック方向は縦方向を想定しており、上向き、下向きの双方にて入力が可能である。

TopCenter（t、y）と MiddleCenter（g、h）
これらのキーは QWERTY 配列の中心に位置する上段または中段のキーである。これらの文字の入力は、QWERTY 配列においてそれらのキーが存在する側の手の人差し指と中指の 2 本指による同時タップによって行う。上段と中段の区別は Meyboard II の上段、

中段のどちらのキーをタップしたかによって行う。例えば、t を入力する際は左手人差し指によって r キーを、左手中指によって e キーを同時にタップする。

従来のソフトウェアキーボードにおいて、ユーザが TopCenter または MiddleCenter の文字を入力する際は、ホームキーから指を横または斜め方向に動かす必要がある。すると、ユーザは文字入力中において指を縦横斜めの三方向に動かすこととなり、結果、ホームキーの位置を見失いやすくなる。一方、Meyboard Ⅱ の方式では指の動きが縦方向に固定されるため、キーの位置を見失いにくい。

BottomCenter（b、n）

これらのキーは QWERTY 配列の中心に位置する下段のキーである。これらの文字の入力は、UnderHome に対するフリックと、TopCenter または MiddleCenter に対する人差し指と中指による入力を組み合わせ、人差し指と中指の 2 本指によるフリックにより行う。UnderHome と同じ行かつ、TopCenter または MiddleCenter と同じ列に属する BottomCenter の文字の入力方法として自然になるように、UnderHome と TopCenter または MiddleCenter 組み合わせた入力方法とした。

なお、Meyboard Ⅱ には t、y、g、h キーが存在しており、TopCenter と MiddleCenter のキーはこれらのキーを 1 本指にてタップしても入力が可能である。また、BottomCenter のキーは t、y、g、h キーを 1 本指にてフリックしても入力が可能である。これらのキーはタッチタイピングを行う際に用いることはなく、2 本指入力にユーザが慣れるまでの代替の入力方法として設けている。

また、Meyboard Ⅱ では QWERTY 配列におけるセミコロンとスラッシュをそれぞれシングルクオーテーションとクエスチョンに置き換えている。これは Windows 10 における標準ソフトウェアキーボードの配列に従っている。

5.2.1　指一本または複数の指によるタップとフリックの判定

Meyboard Ⅱ ではタップとフリックによって入力を行うが、その判定基準は、制限時間以内にユーザが特定の操作を行うことである。タップは 450 ミリ秒以内に指をタッチパネル面に置いて離し、かつタッチパネル面上において指を 1.68mm より動かさないこととした。フリックは 500 ミリ秒以内に指をタッチパネル面に置いて離し、かつタッチパネル面上において指を 1.68mm 以上動かすこととした。また、Meyboard Ⅱ では複数の指を同時に使用する入力（Space、Enter、BackSpace 及び TopCenter、MiddleCenter、BottomCenter のキー）が存在する。同時入力を行うためには、ユーザは最初の指を画面に触れさせてから 200 ミリ秒以内に入力に用いる指すべてを画面に触れさせ、700 ミリ秒以内に画面から離す必要がある。同時入力を行う際、ユーザはタップまたはフリックの制限時間と、同時入力の制限時間の両方を満たす必要がある。但し、タッチパネル面に指を置いた状態を保つ Shift と、BackSpace（Shift+Enter）時の左手に関しては制限時間が存在せず、ユーザが指を置いている限りは入力が保たれる。なお、制限時間と指の移動量に関して、前者は Meyboard よりも長く、後者は Meyboard よりも

短くなっている。これは、前者は Meyboard II には Meyboard にない 4 本指によるタップ入力があるため、入力に時間がかかることを考えて閾値を増やしたこと、そして後者は経験的に指の移動量を半減させても影響がないと考えたためである。

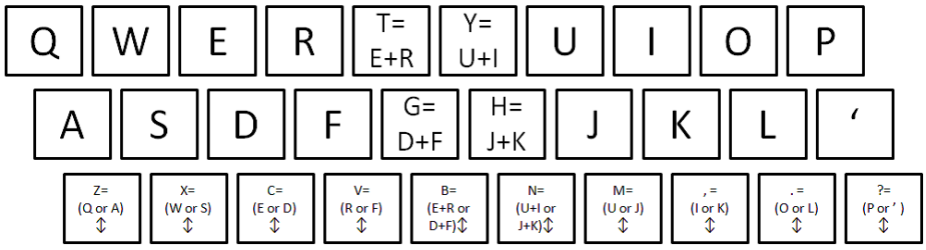

図 5.8: Meyboard II の文字入力方法。縦方向の矢印はフリックを行うことを意味する。これらに加えて、左手 4 本指のタップによって Space を、右手 4 本指のタップによって Enter を、いずれかの手の 4 本指によりタッチパネル面を押し続けることにより Shift を、左手による Shift と Enter の同時入力により BackSpace を入力する。

5.3 比較実験

本節では Meyboard II のタッチタイピング性能を調べるための実験について記述する。

5.3.1 実験環境

図 5.9 から図 5.11 に実験環境を示す。実験には画面の大きさが 20 インチであり、画素数が 1600 x 900 pixel の SONY VAIO Tap 20 を縦向きにして用いている。Meyboard II の入力エリアの大きさは 900 x 248 pixel であり、その対角線の大きさは約 10.2 インチであった。なお、今回の実験では、参加者が入力する文を提示するソフトウェアとして、タイピング練習用ソフトウェアとして無料にて公開されていたものを改造して用いた。また、全ての場合においてパームレストとなる台を用意し、参加者が入力する際にその上に手のひらを置いてもらうように促した。

5.3.2 参加者と実験課題及び実験期間

参加者とそのグループ分け

参加者として、20〜23 歳（平均 21.5 歳）の 6 人の大学生（男性）を時給 820 円にて雇用した。実験前に行ったアンケート調査により、全参加者が QWERTY 配列のキーボードに慣れていることを確認している。具体的には、熟練を「手元を全く見ずに素早い入力が可能であ

[1] C# Tips (Japanese). http://www.geocities.jp/tinqwill/cs.html（参照 2019-02-19）

図 5.9: Windows 8 を使用するグループの実験環境

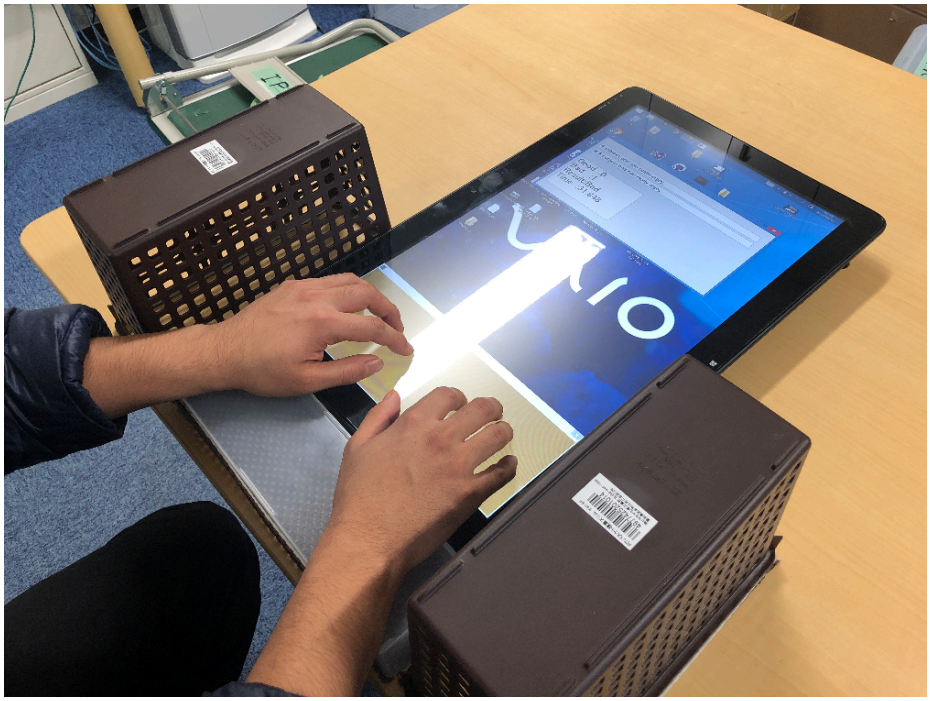

図 5.10: Meyboard II を使用するグループの実験環境（手元を隠す前の状態）

図 5.11: Meyboard II を使用するグループの実験環境（手元を隠した状態）

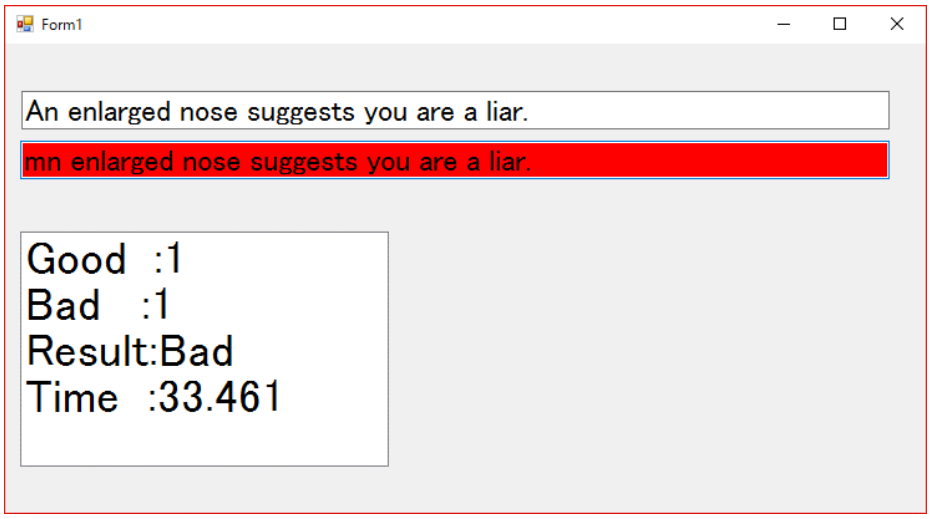

図 5.12: エラーが生じているときの入力ウィンドウ

図 5.13: Windows 8 キーボード

る」として 5 点、未習得を「手元を見ないと入力することができない」状態として 1 点とするリッカート尺度において平均が 4.2 となった。

実験設計は Castellucci ら [CM08] や、Yu ら [YGY+17] が行った複数のキーボードの入力性能を比較する研究に倣ってグループ間実験とした。具体的には、6 人の参加者を 3 人ずつの 2 つのグループに分けた。片方は Windows 8 に標準搭載されているソフトウェアキーボード（図 5.13。以降、“Windows 8”）を用いるグループ（Windows 8 グループ）であり、もう片方は Meyboard Ⅱ を用いるグループ（Meyboard Ⅱ グループ）である。Windows 8 を比較対象としたのは、同じハードウェアにおける同じ幅を持つソフトウェアキーボードとして Meyboard Ⅱ の入力性能を調べる際の基準とするためである。

Windows 8 グループは入力中にキーボードを見てよいこととした（図 5.9）。Meyboard Ⅱ グループにはアイズフリー入力を強制した。このために、画面上において Meyboard Ⅱ が表示されている部分を不透明の板にて覆い隠した（図 5.11）。但し、その板には参加者が文字の入力方法を忘れた際の助けとして図 5.8 のような Meyboard Ⅱ の操作方法が記されている。

なお、この操作方法には QWERTY 配列が描かれているため、間接的にキーの位置を示すことが懸念される。しかし、実際に入力を行う手元は隠されており、かつ Meyboard Ⅱ のキーの位置、大きさ、数は操作方法の記載とは異なる。よって、この操作方法の記載はアイズフリー入力に影響を及ぼさないと考えられる。

実験課題

実験課題は英文の入力である。参加者が 1 文を入力すると、自動的に次の文に切り替わる方式となっている。

参加者は誤入力（エラー）を必ず直さなければならないこととした。入力時に、その文字が正しい場合は入力ウィンドウの入力欄からその文字が消える。文字が正しくない（エラーが生じた）場合は、その文字が入力欄に挿入されて入力欄が赤くなる（図 5.12）。なお、エラーをもたらした入力が文字ではなく、Space の場合は正方形が、Enter の場合はばつ印が代わりに挿入される。エラーが生じている状態においては、参加者は BackSpace を入力することによって文字の消去を行い、エラーが生じる前の状態に戻す必要がある。エラーが生じる前の状態に戻ると入力欄は白に戻り、参加者は入力を続行できる。

各参加者は 1 回につき 6 セッション分の入力（1 ユニットの入力）を行った。各参加者は 1 セッションに 10 文の入力を行った。これらの 10 文は全て異なるものにした。

練習

実験の初日では参加者はユニットの入力の前に練習として 2 セッション分の入力を行った。練習では参加者は QWERTY 配列のある行またはアルファベットの一部を順番に並べた 9〜16 文字の文字列（例えば、“zxcvb nm.”, “ABCDEFG HIJKLMN.”）を文として入力した。これらの文は、参加者に各キーの位置に慣れてもらうため、本番において入力される可能性のあるすべての文字、すなわちアルファベットの大文字または小文字、Space、ピリオドから構成される。セッション中の 10 文の内容は全参加者共通である。但し、提示順はランダムとした。

参加者に各キーの位置に慣れてもらうため、練習においては、Meyboard Ⅱ グループの参加者もキーボードを見てよいことにした。また、参加者に各キーの位置に慣れてもらうため、2 回ある練習セッションのうち 1 回目はキーが表示されるが、2 回目ではキー表示をなくして Meyboard Ⅱ の大きさを示す矩形のみが表示されるようにした。この時の入力は、図 5.10 に示した通りになる。

本番

本番にて入力する英文として、MacKenzie らによって作成された文章セット [MS03] を用いた。本来、この文章セットは文字のすべてが小文字かつ記号が存在しないが、今回の実験では文章の最初の文字と、一人称の “I” を大文字にし、さらに文末にピリオドを追加した。大文字化には文章を 1 文入力するごとに Meyboard Ⅱ の少なくとも片側のキャリブレーションを行うことの、ピリオドの追加には実際の文章入力に近い状況を作る狙いがある。1 セッション中にて入力する 10 文はこの加工された文章セットの中から重複しないようにランダムに選択された。

実験期間

練習本番含め、参加者はセッション間に最低 5 分の休憩をとることとした。1 ユニットの入力に要する時間は休憩も入れて 1〜2 時間であった。ユニット間には 12 時間以上 72 時間以内のインターバルを挟むこととした。この結果、参加者 1 人につき実験開始から終了まで 11〜17 日の期間を要している。

5.3.3 結果

図 5.14 は各グループの入力率の平均を words per minute（wpm） [Gen83] にて表している。入力率に対して二元配置の分散分析を行った結果、キーボード間に有意な差があり（$F_{1,4} = 525.2, p < 0.001$ ）、また、セッション間にも有意な差があった（$F_{59,236} = 6.07, p < 0.001$

)。セッション間の有意差は、実験が進むにつれて入力率が向上したことを示す。キーボード間の有意差に関して、本番の最初のセッションにおいて Windows 8 グループの平均入力率は 21.5wpm であったのに対して、Meyboard Ⅱ グループは 6.98wpm であり、最後の 60 セッション目ではそれぞれ 30.3 と 25.2wpm であった。よって、今回の実験では Windows 8 の方が高速という結果になった。しかしながら、図 5.14 の近似曲線の式から、134 セッション目の入力にて Meyboard Ⅱ の入力率が Windows 8 を上回る計算となり、Meyboard Ⅱ の入力率が最終的に Windows 8 を超えることが期待される [Sei63]。また、Meyboard Ⅱ のグループのある参加者は最終的に 29.0wpm に達しており、Windows 8 のグループの入力率にほぼ等しい入力速度を実現していた。

Meyboard Ⅱ の入力速度と関連研究の比較を図 5.16 に示す。また図 5.17 にその初期段階を示す。関連研究として、Gestyboard 2.0 [CWA$^+$13]、Swipeboard と ZoomBoard（Swipeboard との比較実験） [CGF14]、ZoomBoard [One13]、スマートフォンにおける片手親指入力（Typing）と Shape Writing と SWiM [YPC$^+$17]、ジェスチャ入力（Touch）と Vulture [MJH$^+$14]、iPad に標準搭載されたソフトウェアキーボードと 1Line keyboard [LGYT11] を挙げる。これらのキーボードはすべて QWERTY 配列を基としたキー配列である。関連研究の各実験は入力率に関しては wpm であったが、実験の方式が Meyboard Ⅱ の比較実験同様に一定量の文章を入力するもの（[CWA$^+$13, CGF14, YPC$^+$17, MJH$^+$14]）と、一定時間文章を入力し続けるもの（[One13, LGYT11]）に分かれた。そこで、それぞれ入力した文字数を求めて比較することとした。Meyboard Ⅱ は初期段階では Swipeboard、ZoomBoard と同等の入力率であり、Gestyboard 2.0 を若干上回る結果となった。第 2 章にて述べたように、Gestyboard 2.0 はホームキー以外のキーをフリックによって入力する。一方、Meyboard Ⅱ は QWERTY 配列の下段のキーのみをフリックによって入力する。よって、Gestyboard 2.0 は Meyboard Ⅱ よりも入力に時間がかかると予想された。実際は、Gestyboard 2.0 の実験が完了する 3000 文字入力時点では Meyboard Ⅱ が 15.7wpm、Gestyboard 2.0 が 12.6wpm と Meyboard Ⅱ の方が高く、グラフを見ても Meyboard Ⅱ の方が高い結果となったが、大きな差があるとは言えない。

図 5.15 にエラーの修正量を示す値として correction rate [CM08] を示す（入力率と同様に各グループの平均である）。correction rate は BackSpace の入力回数を用いた指標であり、参加者がエラーをすべて修正する場合に用いられる。この場合、セッション終了時のエラーの数が 0 となり、エラー率も 0 となり評価に用いることができないため、correction rate を代わりに計算する。correction rate は

$$\frac{\text{BackSpace の入力回数（回数/セッション）}}{\text{文章の文字数（文字/セッション）}}$$

として計算される。例えば、参加者が “apple” を入力しようとして、実際には “appple” と入力した場合、参加者は BackSpace を 3 回入力し、入力を “app” まで戻した後に “le” を入力する必要がある。この時、文字列 “apple” の長さ 5 文字に対して BackSpace の入力回数は 3 回なので、correction rate は 0.6（60%）になる。

入力率と同様に correction rate に対して二元配置の分散分析を行った結果、こちらもキーボード間に有意な差があり（$F_{1,4} = 29.9, p < 0.001$ ）、一方で、セッション間には有意差がな

かった（$F_{59,236} = 0.83, p > 0.1$ ）。両キーボードとも実験の進行に対して correction rate に上昇も減少も見られないが、Meyboard Ⅱ は全体的に Windows 8 より低かったことが図 5.15 より分かる。

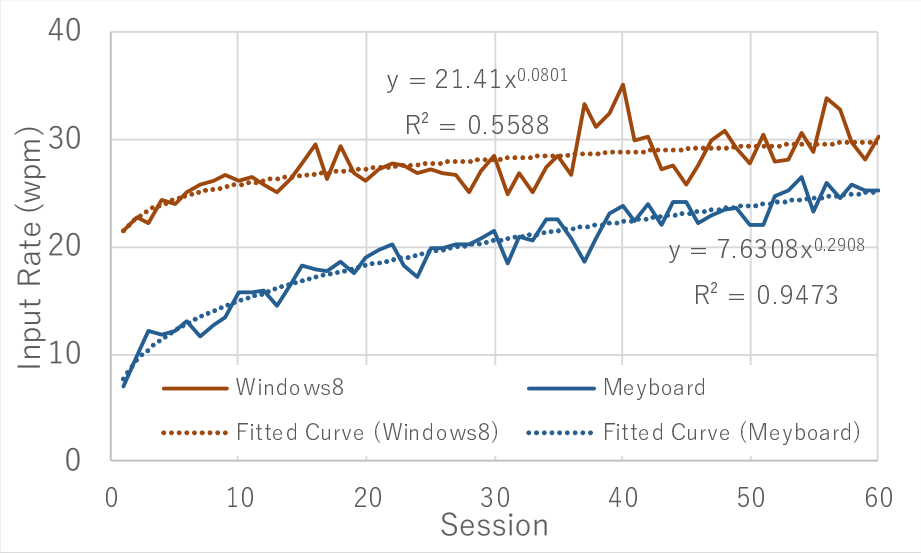

図 5.14: 各ソフトウェアキーボードにおける参加者を平均した入力率

表 5.1: 各ソフトウェアキーボードにおける各種入力回数

入力回数	Windows 8	Meyboard Ⅱ
文字数	53,391	53,350
エラー	3,417	4,091
BackSpace	10,163	7,784

Windows 8 と Meyboard Ⅱ 両グループにおける全セッションの文章における文字数の合計と、エラー数、BackSpace の入力回数を表 5.1 に示す。この時、エラーの発生頻度と、エラーの修正量を区別するために、エラー 1 回を 1 文字の入力ミスが生じてから BackSpace の入力によりそれが修正されるまでとして数えている。

Meyboard Ⅱ はエラー数が多い一方で、BackSpace の入力回数は少ないので、Meyboard Ⅱ は Windows 8 に対してエラーの発生頻度は高いが、エラーの修正量は少ないことになる。考えられることとして、Windows 8 のグループの参加者はキーの位置を入力中に時折確認する必要があり、単語の入力後等、ある程度まとまった入力が完了するまでは入力ウィンドウを見ていなかったと思われる。ゆえに、エラーが生じてからその修正を開始するまでに入力が続いており、結果必要な BackSpace の数も増えたと考えられる。一方、Meyboard Ⅱ のグループの参加者はキーの位置を確認できないアイズフリー入力を行う必要があった。そのため、操

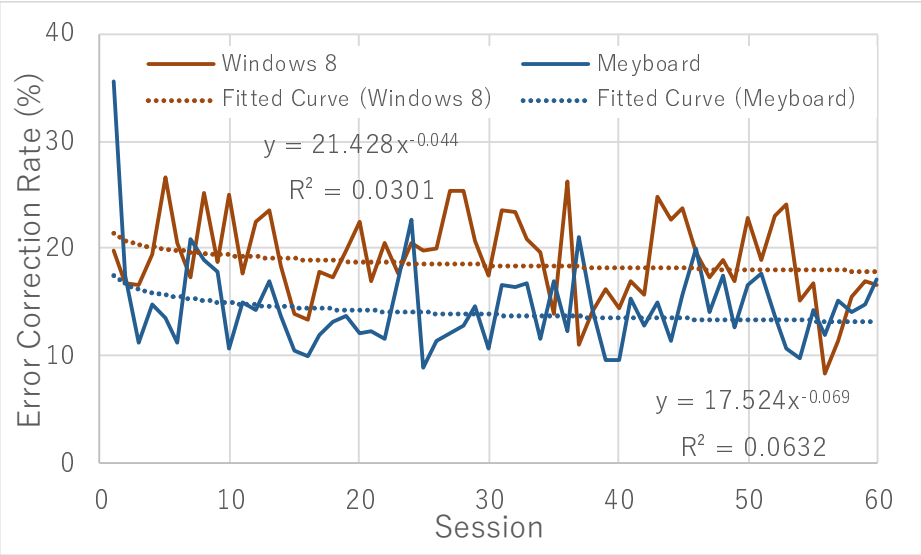

図 5.15: 各ソフトウェアキーボードにおける参加者を平均した correction rate

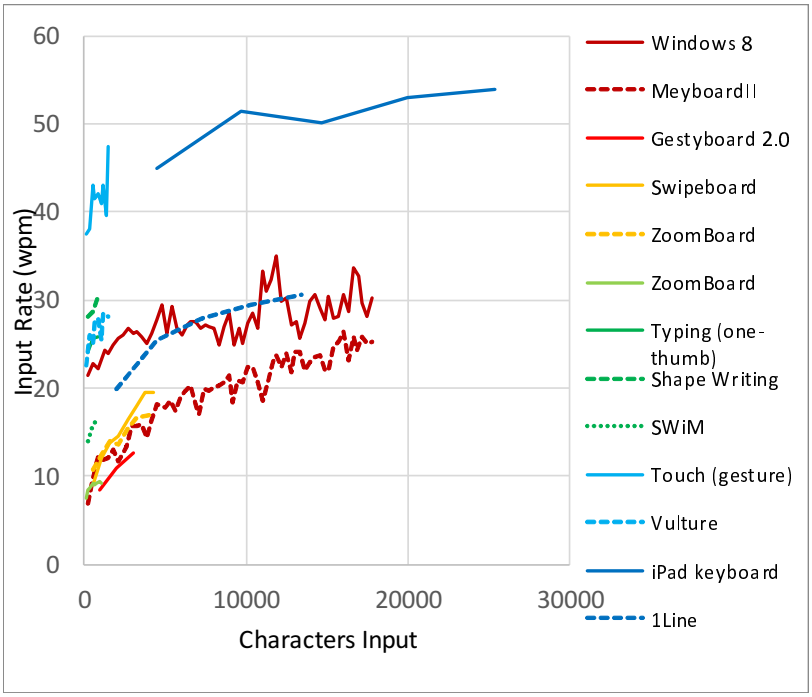

図 5.16: 各ソフトウェアキーボードの実験時の入力文字数と入力率

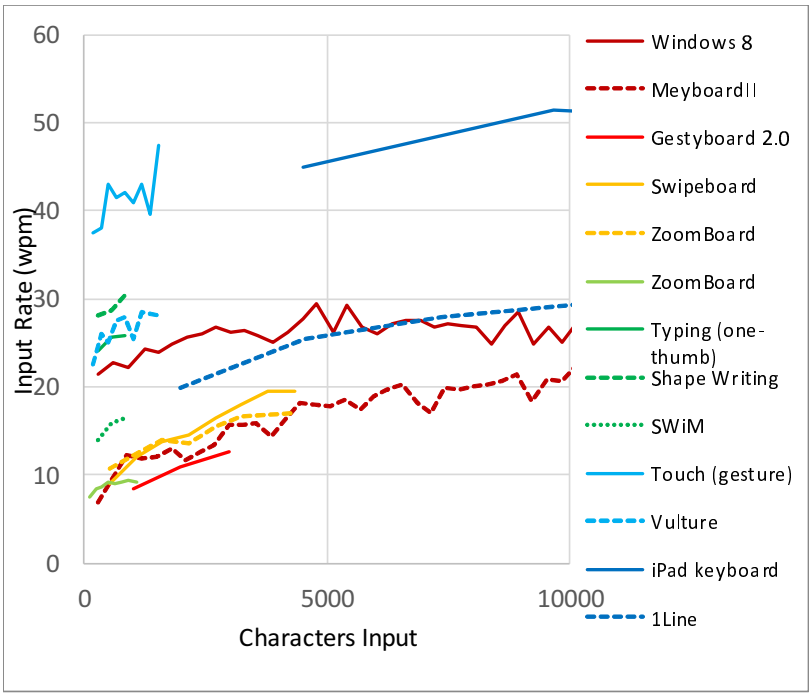

図 5.17: 各ソフトウェアキーボードの実験時の入力文字数と入力率（初期段階）

作方法を確認する以外に入力ウィンドウから視線を外す必要がなく、エラーの発生に直ちに気づくことができるので、1 回のエラー当たりの BackSpace の入力回数が少なかったと考えられる。

5.3.4 考察

図 5.16、図 5.17 を見ると、Meyboard Ⅱ は関連研究及び既存のソフトウェアキーボードと比べて優れた入力速度を直ちに発揮するソフトウェアキーボードであるとはいえない。しかしながら、Meyboard Ⅱ の実験は手元を隠した状態における入力であり、その状態においていくつかの既存研究と同等以上の入力速度を実現したことは、タッチタイピングの実現の観点からは意味のある結果であったと考えられる。

しかしながら Meyboard Ⅱ のエラー数は Windows 8 と比較して多く、これが Meyboard Ⅱ の入力性能を阻害していたと考えられる。Meyboard Ⅱ のエラー数を抑えることができれば、関連研究よりも入力率が高くなる可能性がある。そこで Meyboard Ⅱ グループにおけるエラーの内容を調べ、その原因を考察する。

まず、全てのエラーを以下のアルゴリズムにより誤字と脱字に分類した。

1. セッションにおいて入力を求められた文（提示文）と、参加者の入力した文（入力文）を先頭文字から比較し、両者が一致する場合は次の文字へ移る。

表 5.2: Meyboard Ⅱ のエラーパターンとその発生率

エラー	Windows 8		Meyboard				
				パターン内		パターン外	
	数	発生率	パターン	数	発生率	数	発生率
v → c	20	3.88%	1	54	9.42%	2	0.35%
k → l	18	3.59%	1	32	6.69%	0	0.00%
u → i	82	5.89%	1	57	4.11%	0	0.00%
r → e	45	1.58%	1	113	3.91%	0	0.00%
i → u	2	0.07%	1	95	3.27%	0	0.00%
c → v	18	1.52%	1	33	2.75%	0	0.00%
o → i	8	0.22%	1	75	2.10%	0	0.00%
e → w	64	1.17%	1	112	2.03%	0	0.00%
l → k	3	0.18%	1	31	1.90%	0	0.00%
i → o	210	6.93%	1	53	1.82%	0	0.00%
s → d	39	1.45%	1	35	1.27%	0	0.00%
. → ,	82	4.56%	1	20	1.11%	1	0.06%
d → f	47	3.46%	1	11	0.78%	0	0.00%
. → ?	35	1.94%	1	12	0.67%	0	0.00%
o → p	242	6.65%	1	12	0.34%	0	0.00%
e → r	52	0.95%	1	9	0.16%	0	0.00%
p → ’	0	0.00%	2	42	5.07%	0	0.00%
w → s	7	0.91%	2	32	4.40%	0	0.00%
f → r	1	0.12%	2	34	3.73%	0	0.00%
a → q	9	0.29%	2	92	2.89%	0	0.00%
o → l	36	0.99%	2	102	2.86%	0	0.00%
h → y	0	0.00%	2	51	2.80%	20	1.10%
l → o	2	0.12%	2	43	2.64%	0	0.00%
e → d	28	0.51%	2	141	2.55%	1	0.02%
t → g	58	1.66%	2	49	1.43%	0	0.00%
i → k	52	1.72%	2	28	0.96%	0	0.00%
y → h	64	6.22%	2	8	0.72%	0	0.00%
s → w	35	1.30%	2	7	0.25%	0	0.00%
u → j	51	3.66%	2	1	0.07%	0	0.00%
s → Space	36	1.34%	3	7	0.25%	0	0.00%
g → f	27	2.82%	4	49	5.64%	0	0.00%
g → d	5	0.52%	4	36	4.14%	0	0.00%
y → u	49	4.76%	4	38	3.42%	0	0.00%
t → r	58	1.66%	4	109	3.18%	0	0.00%
n → m	20	0.76%	4	41	1.55%	13	0.49%
h → j	121	6.25%	4	9	0.49%	0	0.00%
m → n	15	1.68%	5	68	7.20%	9	0.95%
r → t	27	0.95%	5	54	1.87%	7	0.24%
u → y	1	0.07%	5	19	1.37%	20	1.44%
c → d	7	0.59%	6	47	3.91%	0	0.00%
m → j	0	0.00%	6	33	3.50%	0	0.00%
. → l	3	0.17%	6	37	2.06%	0	0.00%
n → h	2	0.08%	6	25	0.95%	21	0.80%
h → n	35	1.81%	8	4	0.22%	1	0.05%
n → j	6	0.23%	10	54	2.04%	0	0.00%
r → d	0	0.00%	10	38	1.32%	0	0.00%
T → a	31	6.97%	10	0	0.00%	0	0.00%

2. 両者が一致しなかった場合、提示文の次の文字を参照する。これが入力文と一致した場合はこのエラーを脱字と、一致しなかった場合は誤字と解釈する。

3. エラーが生じた場合、入力文の文字に対して、それが提示文と一致するまで参照を進める。これにより、脱字が生じた場合に以降の入力が誤字と解釈されることを回避している。

例えば、提示文が “force” であり、入力文が “frce” であった場合、アルゴリズムは ‘ o ’ の脱字と解釈する。また、提示文が “force” であり、入力文が “firce” であった場合は ‘ o ’ を ‘ i ’ と間違えた誤字と解釈する。

次に、誤字の内容を調べた結果、いくつかのパターンが見られた。そこで、脱字も含めてエラーを 9 つのパターン（パターン 1〜9）とその他に分けた。以下にパターンを示す。これはその他（パターン 10）を除いて、Meyboard Ⅱ において数が多かった順に番号を振っている。以降、第 4 章同様に、○を△と間違えるエラーを○→△と表す。

パターン 1: 同じ行の隣の列の文字と間違える（r → e 等）。

パターン 2: 同じ列の隣の行の文字と間違える（e → d 等）。

パターン 3: 脱字である。

パターン 4: 人差し指と中指の同時入力において片方の指の入力がない（t → r 等）。

パターン 5: 誤って人差し指と中指の同時入力を行う（m → n 等）。

パターン 6: フリックの代わりにタップを行っている （c → d 等）。

パターン 7: 大文字を入力するべきところが小文字になっている（T → t 等）。

パターン 8: タップの代わりにフリックを行っている（s → x 等）。

パターン 9: 小文字を入力するべきところが大文字になっている（i → I 等）

パターン 10: その他

Meyboard Ⅱ のエラーパターンを元に、各ソフトウェアキーボードのエラー内容とその頻度を調べた。その評価指標として、第 4 章と同様に各エラーの発生率を計算した。

一方で、発生率が高くても出現頻度が低い文字は評価対象から外した。出現頻度が低い文字のエラーは、全体のパフォーマンスに与える影響が小さいと考えたためである。

Windows 8 と Meyboard Ⅱ の各エラーのうち、少なくとも片方のソフトウェアキーボードにおいてエラー数 30 回以上のエラーを調べ、 表 5.2 にてパターンごとに Meyboard Ⅱ のエラー発生率の降順に並べた。エラー数 30 回以上のエラーを評価対象としたのは、この時ユニット当たりの平均エラー数が 1 回となるため高い出現頻度とみなすことができるからである。1（ユニット当たりの平均エラー数）×10（実験のユニット数）×3（グループ当たりの参加者

数）=30（全体のエラー数）となることによる。全体的に、Meyboard Ⅱ の方がよりエラー発生率となる傾向にある。一方で、Windows 8 において発生率が高いエラーは Meyboard Ⅱ では発生率が低い傾向にある。 また、表 5.2 を見ると個々のエラーの発生率は Meyboard の実験時（図 4.2）よりも抑えられていることが分かる。よって、自動適応型としてキャリブレーションを逐次行うことは一定の効果があったと考えられる。 なお、Meyboard Ⅱ にはパターン外のエラーが記載されているが、これは 2 本指による入力の失敗（2 本指によるフリックができず、片方がタップになっている等）、不要な入力の混入、t、y、g、h キーの入力（実験時、これらの文字は 2 本指による入力を想定していたが、キーとしても存在していたため誤って入力する可能性があった）によって生じたものである。

パターン 1

Windows 8 の方が発生率が高い傾向にある。

発生率の傾向は等しい部分と異なる部分が存在した。Windows 8 では u → i、i → o、ピリオド→コンマ、d → f、o → p 等の発生率が Meyboard Ⅱ よりも高い。Meyboard Ⅱ では r → e、i → u、c → v 等の発生率が Windows 8 よりも高い。v → c、k → l 、u → i は双方において 4%以上と高い発生率となった。

発生率の傾向が異なる理由として、キーの並びが考えられる。物理キーボードを模した Windows 8 では、下の段になるにつれてキーの位置が右にずれている（図 5.13）。よって、ユーザが上段のキーを入力しようとするとその右隣を、下段のキーを入力しようとするとその左隣を入力するエラーを起こしやすいと考えられる。実際の Windows 8 のエラーからもその傾向が見られ、発生率の高い v → c、ピリオド→コンマは下段のキーの左隣を、u → i、i → o、o → p は上段のキーの右隣を入力するエラーである。

Meyboard Ⅱ ではキャリブレーション時の参加者の手の傾きがエラーに影響している可能性がある。Meyboard Ⅱ ではキーを右にずらさずに並べているため、キーのずれによるエラーは生じない。一方、手の傾きによってはキーの中心が指の位置に対して外側または内側に寄ることがある。キーが片側に寄るのは、Meyboard Ⅱ では隣り合う指の水平方向の距離に等しい線分を垂直線にて 2 分割し、2 つの垂直線に挟まれた範囲をキーの幅とするためである。キャリブレーション時に手が傾いていた場合、手を傾けない時と比べて人差し指と中指の水平方向の距離が短くなり、中指と薬指、薬指と小指の水平方向の距離が長くなるので、人差し指のキーの中心は指に対してより内側寄りに、中指から小指のキーの中心は指に対してより外側寄りになる（図 5.18）。よって、人差し指のキーは外側を、それ以外のキーは内側を入力するエラーを起こしやすいと考えられる。実際に、発生率の高いエラーを見ると、v → c、u → i、r → e は人差し指のキーの外側を、i → u は中指のキーの内側を入力している。逆に、人差し指のキーは内側を、それ以外のキーは外側を入力するエラーを起こしにくくなるため、Windows 8 にて高かった i → o、o → p の発生率は低くなったと考えられる。

一方で、中指の外側を入力する k → l、e → w 等、上記に当てはまらないが発生率の高いエラーも存在する。人差し指、中指に関しては両者の水平方向の距離が短くなることにより、幅が狭くなるため、両側においてパターン 1 のエラーを引き起こしやすいと思われる。キーの

幅を調整し、広くするとエラーは抑えられるが、これは別のキーの幅を狭くすることになるため、そちらのエラーが増えると予想される。

なお、Meyboard Ⅱ では v → c にパターン外のエラーが存在するが、これは誤って 2 本指による入力を行っており（v は 1 本指による入力）、かつその入力が同時（200 ミリ秒以内）ではない、または人差し指の入力がタップとなっていたために b ではなく c が入力された場合を指している。ピリオド→コンマのパターン外のエラーも誤って 2 本指による入力を行い、かつ中指の入力がタップになったために生じている。

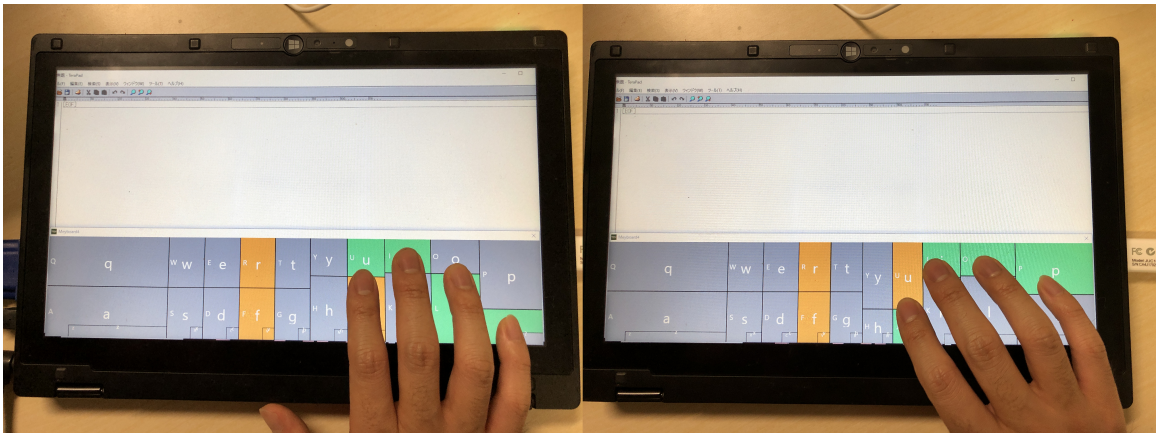

図 5.18: 手の傾きによるキーの大きさの違い（手が傾くと、中指、薬指のキーの右側が広くなる）

パターン 2

Meyboard Ⅱ の方が発生率が高い傾向にあるが、発生率の傾向はそれぞれのキーボードで分かれる傾向にある。y → h、u → j 等は Windows 8 の方が高く、p →シングルクオーテーション、w → s、f → r 等は Meyboard Ⅱ の方が高い。

Windows 8 では右手上段のキーのエラー発生率が高くなる傾向にあり（y → h、u → j、i → k）、これは物理キーボードを模した横方向へのキーのずれが影響している可能性がある。Windows 8 では上段のキーを入力する際にキーのずれのために、参加者が右手ではホームポジションの位置に対して内側の位置を、左手では外側の位置を入力する必要がある。前者の方が参加者にとって負荷の大きい入力であったと考えらえる。特に最も発生率の高い y → h では、ユーザが指を斜め上方と遠くまで動かす必要があることに加えて、キーのずれによりホームキーの j から y までの距離が遠くなっていることも影響していると考えられる。 y → u の発生率が 4.76%と高い（ 表 5.2 のパターン 4 を参照）のも同様の理由と思われる。

Meyboard Ⅱ ではキーの横方向へのずれがないため、左右の手において傾向は変わらないように思われた。しかし、実際は異なる傾向にあった。左手では中指と薬指の上段（w → s、e → d）と、人差し指と小指の中段（f → r、a → q）のエラー発生率が 2%以上と高い。一方右手で

は人差し指と中指あるいはその 2 本の上段（y → h、u → j、i → k）がエラー数 30 未満かつ発生率 1%未満と低く、薬指（o → l）も発生率が 2.86%と数値としては高いが左手薬指（w → s）の 4.40%より低い。よって、右手ではエラー発生率が 5.07%と高い小指を除いて、左手よりも上段のキーのエラーが生じにくいといえる。考えられる理由として、左右の手の傾きと高さの違いが挙げられる。人差し指と小指の高さが等しかった場合の傾きを 0 として、Meyboard Ⅱ グループの最終（第 10）ユニットのキャリブレーション時の平均傾きを求めた結果、左手は 3.63 度、右手は 6.90 度と右手の方が傾きが急であったことが分かった（小指が人差し指よりも高くなる回転方向を正としている）。また、同様に第 10 ユニットのキャリブレーション時の指の平均 Y 座標を求めた結果、左手は 136.2、右手は 103.5 と、右手の方が高い位置にあったことが分かった（Y 座標が 0 の時 Meyboard Ⅱ の上端に位置し、値が大きいほど下側に位置する）。右手の方が傾きが急であるため、指の中において下側に位置する人差し指と中指の上段のキーの高さが高くなると考えられる。また、右手は高い位置にあるため、上段のキーが入力しやすくなっており、さらに図 5.6 に示したキーの高さ補正によって上段のキーは大きくなるように補正される。よって、右手は左手よりも上段のキーのエラーが生じにくくなったと考えられる。但し、手の傾きにより指の中において上側に位置する、薬指と小指の上段のキーは高さが低くなり、キーの高さ補正をもってしてもエラー発生率 2%以上となったと考えられる。

キーの高さ補正には一定の効果があったと考えられるが、Windows 8 と比較した Meyboard Ⅱ のパターン 2 の発生率の高さを鑑みると、改善が必要である。キーの高さ補正がないと、キャリブレーション時の指の位置によってはキーの高さが小さくなりすぎて入力できなくなる可能性がある。一方で、現在のキーの高さ補正はその補正量がキャリブレーションごとに変わる変動値であり、結果として上段のキーまたは中段のキーを入力するための指の移動量がキャリブレーションごとに変化する。そこで改善案として、上段のキーまたは中段のキーのうち指の位置から遠いキーまでの距離が固定値となるようにキーの高さ補正を行うことが考えられる。

なお、右手上段の小指のエラー率がパターン 2 において最も高いエラー発生率となった理由として、小指の長さが短く可動範囲がほかの指と比べて狭いことも考えられる。

なお、Meyboard Ⅱ の h → y のパターン外のエラーは 2 本指の入力ではなく、y キーの入力によって生じたエラーである。y キーがなければ生じないエラーであるため、Meyboard Ⅱ にてアイズフリー入力を行う場合は t、y、g、h のキーをなくすと精度向上が望まれる。e → d のパターン外のエラーは左手の中指に加えて、右手の 4 本指による入力がされていたために生じた。右手 4 本指の入力（Shift）であるため、この時の入力が大文字の D となる可能性もあったが、左手の中指より先に右手の指が画面から離れていたために Shift が解除され、小文字の d となった。

パターン 3

Windows 8 において単語末の s が抜けて Space が入力される場合を除いて、特定のエラーが多くなる傾向はなかった。脱字の種類は多岐にわたり、全体の発生率は Windows 8 が 0.96%、

MeyboardⅡが 0.88%と MeyboardⅡの方が少ない結果となった。これは、MeyboardⅡがアイズフリー入力であったため、参加者が脱字に早く気付くことができたためと思われる。

パターン 4、パターン 5

パターン 4、パターン 5 は 2 本指による入力に関係したエラーである。パターン 4 が 2 本指にて入力すべき際に 1 本指による入力になっていた場合、パターン 5 が 1 本指にて入力すべき際に 2 本指による入力になっていた場合である。

Windows 8 には 2 本指入力が存在しないので、パターン 4、5 のエラー発生率は低いと思われたが、パターン 4 に関しては隣のキーを誤入力するエラー（MeyboardⅡにおけるパターン 1）のために、全体としては MeyboardⅡと同等のエラー発生率となった。

MeyboardⅡではパターン 4 の方がパターン 5 より全体的なエラー発生率が高く、参加者が 2 本指の入力に習熟していないと考えられる結果となった。但し、m の入力（1 本指による）と n の入力（2 本指による）に関しては、パターン 4 の n → m（1.55%）に対してパターン 5 の m → n（7.20%）の発生率が高い結果となった。これは参加者が共に右手人差し指を用いる m（出現頻度 944 回）と n（出現頻度 2641 回）の入力を混同した結果、より出現頻度の高い n の入力を行ったためと考えられる。パターン 4、パターン 5 共に、習熟によってエラー発生率を抑えることが期待される。

なお、n → m のパターン外のエラーは 2 本指による入力ではあるが、中指がタップとなっていたために m が入力されることにより生じた。m → n、r → t、u → y のパターン外のエラーはすべて 2 本指による入力ではなく、t、y、g、h キーの入力によって生じたエラーであった（例えば m → n は y または h キーのフリックにより生じた）。パターン 2 の項目にて述べたように、t、y、g、h キーを除いて人差し指のキーを大きくすることにより精度向上が見込まれる。

パターン 6、パターン 8

パターン 6、パターン 8 はフリックに関係したエラーである。パターン 6 がフリックすべき際にタップになっていた場合、パターン 8 がタップすべき際にフリックになっていた場合である。

パターン 6 では、MeyboardⅡの方が高いエラー発生率となった。一方パターン 8 は Windows 8 においては入力すべきキーの上段、または下段のキーを入力するエラー（MeyboardⅡにおけるパターン 2）となり、MeyboardⅡと比較するとエラー発生率が高くなった。また、MeyboardⅡはパターン 8 のエラー数が低く、最多でも s → x の 12 回であった。

MeyboardⅡにおいてパターン 6 が生じる主要因は、フリックよりもタップの方が負荷の少ない入力であったためと考えられる。パターン 6 における指の移動量を調べた結果、その 73.6%は移動量が 0 ピクセル、8.1%が 1 ピクセルであり、参加者が指を動かしていない場合の方が多いことが分かった。よって、参加者はパターン 6 の大半においてタップを行っていたといえる。フリックは負荷の大きい入力であるが、図 5.2 と図 5.3 にて示したように、フリックを用いる下段の文字は出現頻度が小さいため、全体としてのエラー発生率は抑えられる。

なお、n → h のパターン外のエラーは、2 本指による h の入力ではなく、h キーの入力によって生じたエラーである。

パターン 7、パターン 9

パターン 7 は大文字の代わりに小文字を入力する、パターン 9 は小文字の代わりに大文字を入力するエラーである。大文字の出現頻度が低いこともあり、そのエラー数は双方のキーボードにおいてどちらのパターンも少なく、 表 5.2 には挙げていない。エラー数が多いものを挙げると、Windows 8 における A → a の 18 回（発生率 11.6%）、Meyboard Ⅱ における T → t の 19 回（発生率 4.3%）、S → s の 11 回（発生率 9.7%）、となる。パターン 9 は全てエラー数が 10 未満であった。パターン 7 とパターン 9 を合わせた、全体の発生率は Windows 8 が 0.06%、Meyboard Ⅱ が 0.12% と Meyboard Ⅱ の方が大きい結果となった。Meyboard Ⅱ グループの参加者の入力を見ると、文字を入力する前に Shift の入力を行う 4 本指の少なくとも 1 本が画面から離れている場合が多く見られた。この場合、Shift 入力が解除されるので小文字が入力される。

4 本指の入力があった次の入力を必ず大文字にすれば、この状況は回避可能であるが、4 本指の入力を Space、Enter 等の入力にも割り当てているためこれらの入力との区別が必要となる。新たな入力方法（3 本指等）を割り当てることも考えられるが、キャリブレーションを逐次行うためにも Shift 入力は 4 本指入力にて行うことが望ましい。結論としては、エラー数が少ないため、ユーザの習熟による改善を図ることが望ましいと考える。

パターン 10

Windows 8 では Shift キーの真上にある a キーを入力することにより生じる T → a（Meyboard Ⅱ におけるパターン 2）、Meyboard Ⅱ では 2 本指フリックにより入力すべきところを 1 本指タップにて入力する n → j と、入力するキーの行、列ともに隣を入力する r → d のエラー数が多くなった。このように、Meyboard Ⅱ にて見られたパターン 10 のエラーは、複数のエラーパターンの組み合わせともいえる。ゆえに、パターン 1 から 9 までのエラーを改善することにより、精度向上が期待される。

5.4 長期実験

本節では長期的に Meyboard Ⅱ の入力を行った場合のタッチタイピング性能を調べるための実験について記述する。

5.4.1 実験環境

図 5.19 に実験環境を示す。実験には画面の大きさが 11.6 インチであり、画素数が 1366 x 768 pixel の Panasonic Let’s Note CF-AX2 を用いている。Meyboard Ⅱ の入力エリアの大きさは

1366 x 350 pixel であり、その対角線の大きさは約 10.4 インチであった。Let's Note CF-AX2 における Meyboard Ⅱ の横幅は 5.3 節にて述べた比較実験における SONY VAIO Tap 20 を使用時と等しく 250mm であった。なお、Let's Note CF-AX2 の端末の厚みは 17mm であり、パームレストとなる台がなくとも手のひらを置きながらの入力が可能であった。そこで参加者はパームレスト等は用いずに机の上に直接手のひらを置きながら入力を行った。

今回の実験では比較実験のように手元を隠してはいなかったが、タッチタイピングに近い状態にて入力を行うために Meyboard Ⅱ のすべてのキーを非表示としている。

図 5.19: 長期実験の実験環境

5.4.2 参加者と実験課題及び実験期間

本実験の参加者は筆者一人である。

実験課題は比較実験と同様に MacKenzie らによる英文の文章セットの入力とした。10 文の入力を 1 セッションとして、1 日当たり 3 セッションの入力を行うこととした。但し、最初の 4 日間のみ練習を兼ねて 1 日当たり 6 セッションの入力を行っている。この実験においても、エラーは必ず直すこととした。

実験期間は 2016 年 8 月 20 日から 2019 年 1 月 4 日の 868 日間であり、入力量は 2608 セッションに相当する。但し、2016 年 8 月 25 日から 28 日の 4 日間のみ実験を行っていない。

5.4.3 結果

入力率の推移を図 5.20 に、error correction rate の推移を図 5.21 に示す。図 5.21 には 21 区間移動平均を描画している。これは、直近 1 週間に相当する 21 セッション分の平均値である。

入力率は実験の進行とともに上昇する傾向にあり、error correction rate は 5%前後の値を推移し続けている（全セッションの平均 error correction rate は 5.29%、標準偏差は 3.38 であった）。実験の終盤 1 週間に相当する最終 21 セッション分の平均入力率は 49.5%、平均 error correction rate は 4.76%となった。

比較実験の結果と比較するため、同じセッション数である実験開始より 60 セッション分の入力率を図 5.22 に、error correction rate の推移を図 5.23 に示す。長期実験の初期における入力率の変化は比較実験の Meyboard Ⅱ グループと似た傾向であることが分かる。一方、error correction rate は Meyboard Ⅱ グループよりも低くなった。これは BackSpace の入力回数が少ないことを意味する。

表 5.3 に比較実験時の Windows 8 グループと Meyboard Ⅱ グループ、および長期実験時の Meyboard Ⅱ の入力文字数、エラー数、BackSpace の入力回数を示す。長期実験ではエラー数が Windows 8 グループと 10 回差と近い値となっており、Meyboard Ⅱ グループより低い。BackSpace の入力回数に関しては Meyboard Ⅱ グループと同様 Windows 8 グループよりも低い値となっている。これは比較実験時と同様に、タッチタイピング入力によって手元注視がないためエラーの発生に参加者がすぐに気づいたためと考えられる。

比較実験においては Meyboard Ⅱ の入力率が将来的に Windows 8 と同等以上に上昇することが予想された。図 5.14 の近似曲線の式から、134 セッション目の入力にて Meyboard Ⅱ の入力率が Windows 8 を上回る計算となっており、この時の計算上の入力率は 31.7wpm であった。長期実験ではこの値を超えたのが 87 セッション目（32.6wpm）の時であり、概ねこの値となる、すなわち直近 21 セッションの平均がこの値を超えるのは 94 セッション目（31.8wpm）であった（図 5.24）。比較実験時における計算上の予想を超えており、Meyboard Ⅱ の長期的な入力による性能向上が期待される。

表 5.3: 各実験における実験開始後 60 セッション分の各種入力回数。比較実験の数値は参加者の平均値とした

入力回数	Windows 8	Meyboard Ⅱ（比較実験）	Meyboard Ⅱ（長期実験）
文字数	17,797	17,783	17,743
エラー	1,139	1,364	1,149
BackSpace	3,388	2,595	1,930

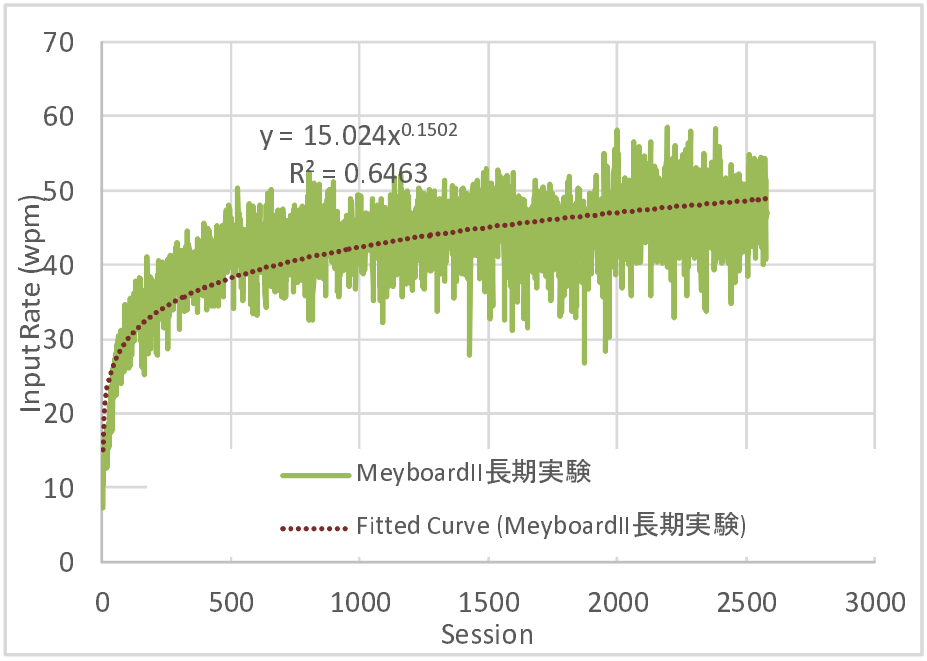

図 5.20: Meyboard II の長期実験における入力率

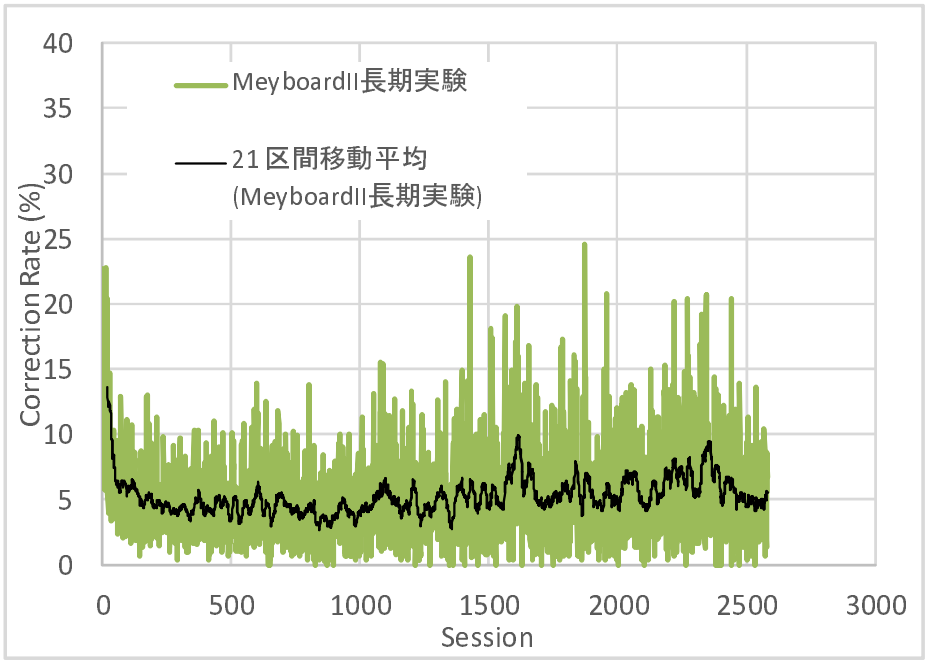

図 5.21: Meyboard II の長期実験における correction rate

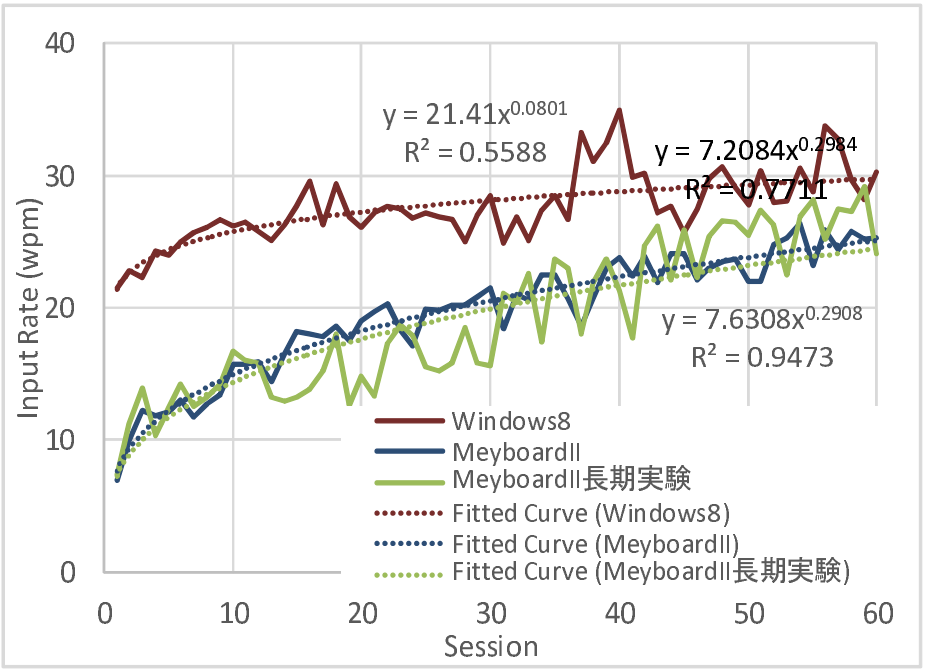

図 5.22: 実験開始後 60 セッション分の入力率の推移

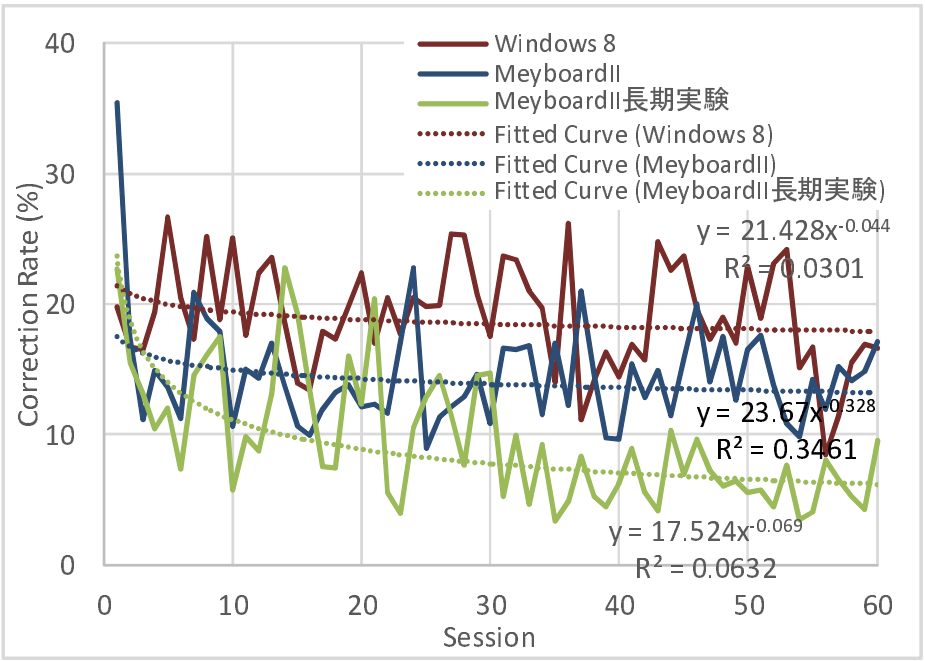

図 5.23: 実験開始後 60 セッション分の error correction rate の推移

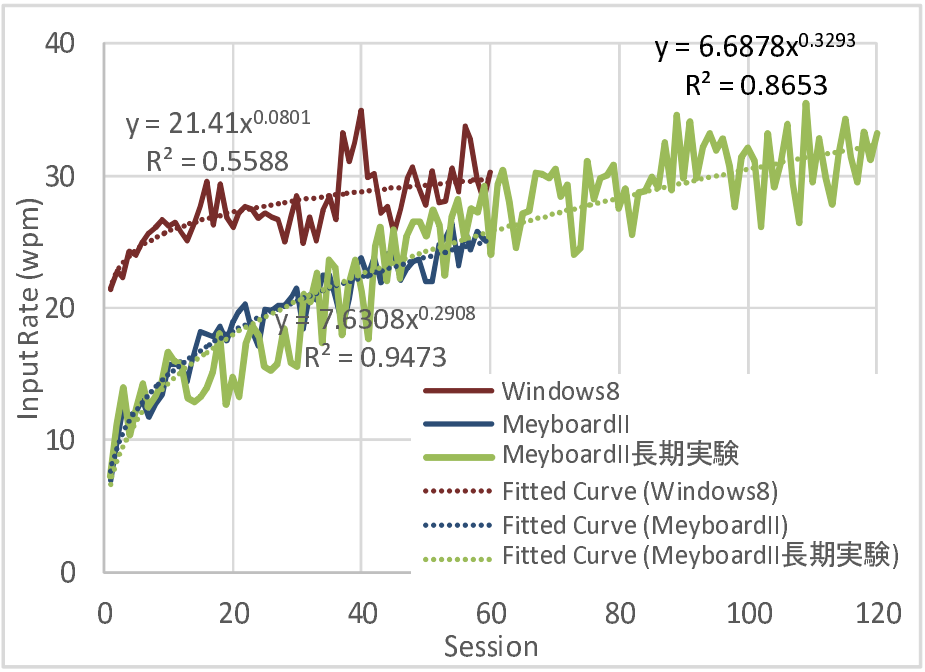

図 5.24: 長期実験開始後 120 セッション分の入力率の推移

5.4.4 考察

同じ入力量である実験開始後 60 セッション分を比較すると、Meyboard Ⅱ の長期実験では、エラー数、BackSpace の入力回数が共に比較実験時よりも少なくなった。長期実験の参加者と、比較実験の参加者の入力方法に何かしらの違いがあると考えられる。それを推測するために、比較実験時と同様にエラーを 9 つのパターン（パターン 1〜9）とその他（パターン 10）に分け、それぞれの発生率を計算した。長期実験の開始 60 セッション分の各エラーのうち、エラー数 10 回以上のエラーを調べ、表 5.4 にてパターンごとにエラー発生率の降順に並べた。表 5.4 には同様のエラーに対して比較実験時の発生率も記述している。

表 5.4 より分かる長期実験と比較実験の違いとして特徴的なのは、長期実験ではパターン 1（横方向のキーと間違える）のエラーの発生率が比較実験よりも低いことである。表 5.2 では、比較実験におけるパターン 1 のエラーとして v → c（9.42%）、k → l（6.69%）、u → i（4.11%）などが見られたが、長期実験ではこれらのエラーの頻度が少なかった（それぞれ 1 回、7 回、2 回。発生率はそれぞれ 0.51%、3.93%、0.45%）。ここから推測できることとして、長期実験の参加者は指をなるべく横方向に動かさず、縦方向のみに動かす運指ができていたと考えられる。Meyboard Ⅱ は縦方向の指の移動のみですべてのキーが入力可能なように設計されており、長期実験の参加者はそのことを理解していたため、横方向に動かさないことを意識して入力を行っていた。比較実験においても、キーの入力方法を説明する際に縦方向の指の移動のみですべてのキーが入力可能であることを説明することにより、参加者に指を縦方向に動かすことを促していた。しかし、実際の運指は傾いた直線であるかあるいは曲線を描いてお

り、入力が進むにつれて指の位置がキーから外れ、パターン 1 のエラーを生じさせたと考えられる。ここから、Meyboard Ⅱ の格子状のキー配列は指を縦方向に動かす運指を習得したユーザにとっては正確な入力を可能とするものであるが、そうでないユーザにはエラーを生じさせやすいものであると考えられる。エラーを抑えるためには、Meyboard Ⅱ における指を縦方向に動かす運指をユーザに重点的に練習してもらうか、あるいは事前にユーザの運指をソフトウェアキーボードに学習させることにより、ユーザの運指に合わせたキャリブレーションを行う仕様とする必要があると思われる。

次に、習熟した状態におけるエラーの内容を調べるため、同じセッション数かつ習熟した状態における入力である最終 60 セッション分を対象として、エラーの分類と発生率の計算を行った。 長期実験の各エラーのうち、エラー数 3 回以上のエラーを調べ、 表 5.5 にてパターンごとにエラー発生率の降順に並べた。

結果として、習熟した状態ではエラー発生率が 1%未満のエラーがほとんどであり、習熟によってエラーが抑えられることが確認できる。しかしながら、一部のエラーには 1%を超えた高い発生率のものがあり、それらは k → j（1.88%）、p →シングルクオーテーション（2.96%）であった。

k → j はパターン 1 のエラーであり、右手中指による入力である。比較実験の考察において述べたように、手の傾きによって右手中指のキーは外側（右手の場合は右側）寄りになる（図 5.18）。これが発生率が高くなった要因と思われる。但し、k → j は発生率の高さに対してエラー数が 3 回と少なく、k の出現頻度が低いことがわかる。よって、入力全体に対して与える影響は小さいと考えられる。k → j の回数を減らすためにはキャリブレーション時に k の幅がより大きくなるように、キーの幅の決定方法を変えることが考えられる。この場合、k がある列の左側に位置する u、j の幅は小さくなるのでこれらのエラーが増える可能性がある。Hiraga らによると、QWERTY 配列では u、j のある右手人差し指の方が k のある右手中指よりも使用頻度が高い（右手人差し指が 19.4%、右手中指が 9.2% [HOY80]）ため、より頻度の低いキーのエラーを抑えるためにキーの幅を変えるかは疑問である。

p →シングルクオーテーションはパターン 2（縦方向のキーと間違える）のエラーであり、右手小指による入力である。エラー数も多いため、入力全体に対して与える影響も大きいといえる。

反対側の左手小指においては、a → q の発生率が 0%（エラーがない。a の入力回数は 1,022 回）、q → a の発生率が 3.7%（但しエラー数は 1 回。q の入力回数は 27 回しかない）であった。Meyboard Ⅱ では下側に存在する中段側の a キーにおいて a → q がなかったことを考えると、小指は上段側のキーに指が届きにくいといえる。小指は指の中では短いため、Meyboard Ⅱ のキーの高さ補正をもってしても上段側に小指を届かせるには不十分であると考えられる。

QWERTY 配列においては小指のキーの入力回数に偏りがあるため、入力回数が多い側のキーを敢えて大きくなるように補正を行うことによって、入力全体に対して与える影響を抑えられると考えられる。入力回数の偏りに関して、左手小指においては上段の q が 27 回、中段の a が 1,022 回、下段の z が 11 回と中段の a に偏っており、右手小指においてはアルファベットが p のみである。

仮に Meyboard II を QWERTY 配列ではない他の配列を用いて実装する場合、同様に小指のキーの使用頻度に偏りが生じていて、上段または中段のいずれかの使用頻度が低くなっていることが望ましい。例えば Dvorak 配列では左手小指に関してはアルファベットが a のみであるため偏りができているが、右手小指に関しては上中下段がそれぞれ l、s、z となっており上段と中段の使用頻度の偏りが乏しい（長期実験最終 60 セッションでは l の入力回数が 579 回、s が 889 回であった）。この場合、l → s または s → l のエラー数が多くなると予想される。例えば l を使用頻度の低い x（長期実験最終 60 セッションでは入力回数が 43 回）と入れ替えるといった配列上の工夫を行うことが考えられる。但し、元の Dvorak 配列から離れていくので、そのユーザにとっては使いにくくなる可能性がある。

あるいは、小指のキーに関しては一段にまで省略し、人差し指外側のキー（t、y、g、h）のように複数本の指による入力を行うキーを設けるか、下段のようにフリックによって入力するキーを設けるなど、他の入力方法を設けることが考えられる。

表 5.4: 長期実験時の Meyboard II のエラーパターンとその発生率（開始 60 セッション分）

エラー	パターン	数	発生率	比較実験時の発生率
r → e	1	21	2.24%	3.91%
i → o	1	13	1.30%	1.82%
e → w	1	18	1.01%	2.03%
w → s	2	18	7.53%	4.40%
h → y	2	31	4.93%	2.80%
f → r	2	11	3.82%	3.73%
a → q	2	34	3.14%	2.89%
i → k	2	21	2.10%	0.96%
e → d	2	13	0.73%	2.55%
t → r	4	50	4.27%	3.18%
y → u	4	12	3.31%	3.42%
n → ,	4	24	2.92%	0.34%
h → j	4	17	2.70%	0.49%
n → m	4	10	1.22%	1.55%
r → t	5	24	2.56%	1.87%
e → t	5	16	0.90%	0.45%
c → d	6	76	19.84%	3.91%
v → f	6	14	7.22%	4.01%
b → g	6	13	6.47%	3.19%
n → h	6	36	4.37%	0.95%
. → l	6	21	3.50%	2.06%
T → t	7	17	10.69%	8.14%
d → c	8	10	2.30%	0.21%
l → .	8	13	2.25%	0.61%
u → k	10	10	2.27%	0.36%
n → j	10	16	1.94%	2.04%
r → d	10	11	1.18%	1.32%
Space → e	10	10	0.37%	0.19%

表 5.5: 長期実験時の Meyboard Ⅱ のエラーパターンとその発生率（最終 60 セッション分）

エラー	パターン	数	発生率	比較実験時の発生率
k → j	1	3	1.88%	1.05%
i → u	1	9	0.90%	3.27%
o → p	1	4	0.33%	0.34%
e → w	1	3	0.16%	2.03%
p → ’	2	9	2.96%	5.07%
o → l	2	9	0.75%	2.86%
h → y	2	3	0.49%	2.80%
r → f	2	3	0.31%	0.21%
i → k	2	3	0.30%	0.96%
d → i	3	3	0.64%	0.00%
r → e	3	4	0.42%	0.62%
t → Space	3	4	0.35%	0.00%
r → o	3	3	0.31%	0.07%
i → t	3	3	0.30%	0.07%
t → a	3	3	0.26%	0.06%
t → i	3	3	0.26%	0.15%
Space → d	3	6	0.23%	0.00%
e → d	3	4	0.22%	0.16%
t → r	4	7	0.61%	3.18%
n → ,	4	3	0.35%	0.34%
u → y	5	3	0.65%	1.37%
e → p	10	3	0.16%	0.04%
Space → e	10	4	0.15%	0.00%

第6章　結論

本研究では、通常のソフトウェアキーボードでは困難なタッチタイピングを実現する手法として、適応型ソフトウェアキーボードを提案した。 結論としては QWERTY 配列を基としたソフトウェアキーボードにおいて適応型とするのみでは手の位置のずれとタッチ位置の重なりによりタッチタイピングは困難であった。本研究では減数キーボードとすることと、自動適応型とすることによってこれらの問題に対応し、タッチタイピング実現の可能性を示した。

最初の適応型ソフトウェアキーボードのプロトタイプである Leyboard はソフトウェアキーボードのキーをユーザの指の位置とその周辺に配置することによって入力を容易にすることを狙ったものであった。Leyboard の長期入力実験は既存のソフトウェアキーボードに対して有意に高速な入力が可能となり得ることを示した。しかし、手元注視をなくしてタッチタイピングを実現する目的は達成されなかった。

Leyboard において手元注視が必要となった要因は、入力が進むにつれて生じる手、及び指の位置のずれと考えられた。手のひらの下側を机上に置き、手の位置を安定させることによってずれを抑えることを検討したが、その際のタッチ位置を見ると、ユーザが認識する領域、すなわちタッチ入力の領域が行方向においても列方向においても互いに重なり合っており、ずれの発生は避けられないことが判明した。

タッチ位置の重なりを抑えるために、タッチ入力の領域の数を減らすことを考えた。そこで、QWERTY 配列キーボードの中段のキーという領域を省略した適応型減数ソフトウェアキーボードである Meyboard を設計、実装した。Meyboard では上下段のキーのいずれかにおけるフリック入力を中段のキーの入力と解釈することによって、省略された中段のキーの入力を補った。実験の結果では、Meyboard における手元注視の可不可は入力率に有意差を与えていなかった。よってタッチタイピング実現の可能性は見受けられたが、一方で手元注視を行わないときにエラー率が高くなることも判明した。その原因は、キー 2 列分の入力を担当する人差し指において、t、y 等の外側に位置するキーを入力しやすくするために r、u 等の内側のキーが入力しにくくなっていたこと、そして入力が進むにつれて生じる指の位置のずれによって入力当初のキャリブレーションが意味をなさなくなるためと考えられた。

キャリブレーションの頻度を上げることにより、指の位置のずれにソフトウェアキーボードのキーの位置を逐次合わせることによって上記問題に対処することとした。キャリブレーションを逐次自動的に行う自動適応型減数ソフトウェアキーボードとして、Meyboard Ⅱ を設計、実装した。Meyboard Ⅱ では Space、Enter、Shift など文字入力において頻繁に入力することとなるキーの入力時にキャリブレーションを行うことにより、キャリブレーションの自動化を実現した。また、人差し指の担当するキーに関しては t、y 等の外側に位置するキーを 2

本指による入力としたことにより、他の指同様に縦方向の移動のみによって入力を可能とした。これは、Meyboard において頻発していた内側のキーと間違えて外側のキーを入力するエラーを抑えることを意図していた。

実験の結果、エラーの数と発生率から Meyboard Ⅱ は Meyboard において頻出した横方向のキーと間違えるエラーの数が抑えられていると考えられ、キャリブレーションの自動化に効果があることを示した。しかしながらエラー数に関しては既存のソフトウェアキーボードよりもまだ多い状態であった。長期実験を行った結果、ほとんどのエラーの発生率を 1%未満とすることができ、習熟によってエラーの発生を抑えられる可能性を示した。

6.1 本研究の貢献

本研究の貢献は、QWERTY 配列を基にしたソフトウェアキーボードにおいて、ユーザがアルファベットによって構成される任意の文字列をタッチタイピングにて入力することを可能にする手法を示したことである。その際に得た知見は、ソフトウェアキーボードのデザインを考える際に有用であると考える。

第 3 章にて示したように、適応型ソフトウェアキーボードは指の位置がキーの位置に合っている限りにおいては既存のソフトウェアキーボードを上回る入力性能を発揮可能と考えられる。 しかしながら、手の位置の安定の有無を問わず入力に応じて指の位置がキャリブレーション位置とずれていくこと、タッチ位置が重なることにより適応型ソフトウェアキーボード単体ではタッチタイピングの実現が困難である。これらの現象は少なくとも上段、中段、下段のキーの配置によって構成される物理キーボードを模したソフトウェアキーボードにおいて生じるため、これに該当するソフトウェアキーボードの設計者はこのことに留意することが望ましい。

本研究において示したこれらの問題への対処策は、第 5 章にて示したキャリブレーションを自動化する自動適応型としたこと、第 4 章にて示したキーを省略することによりタッチ位置の領域の数を減らし、タッチ位置の重なりを抑えた減数キーボードとしたことである。実験結果により、これらの対処策に一定の効果があることを示したことも本研究の貢献であると考える。

6.2 今後の展望

本研究の今後の展望としては、以下の二つが挙げられる。

6.2.1 タッチタイピング可能なソフトウェアキーボードとしての展望

本研究における自動適応と減数キーボードはそれぞれ指の位置のずれとタッチ位置の重なりという問題に対する一解決策であり、他にも様々な解決策が存在すると考えられる。例えば、本研究においては採用しなかった辞書を用いた補正はこれらの問題に対する別方面から

のアプローチといえる。これらの問題に対する対処という観点からタッチタイピング可能なソフトウェアキーボードの諸入力手法を横断し、入力速度、快適性などのユーザの実用上において最適な解決策を求める研究が今後望まれる。

また、本研究におけるキャリブレーション手法も指の位置に合わせたキーの位置と大きさの決定手法に関する一案であり、同様に様々な決定手法が存在すると考えられる。特に、本研究のキャリブレーション手法は各キーの入力エラー数を抑えるという側面が強いが、人間の指の可動範囲等も考慮したユーザの快適性の観点におけるキャリブレーションに関する研究が今後望まれる。

なお、本研究の手法は複数本の指による接触を検知できれば実装可能であるため、タッチパネル以外に例えばテーブルに投影するタイプのソフトウェアキーボード等にも応用可能である。

6.2.2 入力インタフェースとしての展望

本研究における実装はタッチパネルを搭載したタブレット端末を想定しており、その機能上の制約を受けている。すなわち、1 本または複数本の指が触れたことと、その位置、動きを検知する機能のみにおいて入力するキーまたはキャリブレーション時の各キーの位置と大きさを判断している。これは既存のソフトウェアキーボードの大多数が同様の機能のみを利用して実装されているからであるが、近年はこれらの端末に指のホバー検知、タッチ感圧の検知など新たなインタラクション手法を追加することが検討されている。本研究の展望の一つは、これらのインタラクション手法により、タッチタイピング可能かつ Meyboard Ⅱ よりもさらにエラー数を抑えた入力インタフェースを実現することである。例えば、ユーザの各指の位置をセンシングする機能が端末に実装されれば、指によって入力されるキーの候補が縦方向に限定されるため、Meyboard において頻発した横方向のキーと間違えるエラーの数が 0 となる。

また、本研究にて採用した減数キーボードはキーボードの表示領域を小さくする効果もあるため、スマートフォン、スマートウォッチ等本研究の対象としたタブレット端末よりも画面の領域が小さい端末において、情報を表示する領域を多く確保できる。このことがこれらの端末を利用する際の作業効率を向上させることができるのか、既存のソフトウェアキーボードと比較した研究が望まれる。

6.2.3 本研究にて用いた実験手法の応用に関する展望

本研究ではタッチタイピングによる入力が可能であるかを調べるために、実験を不透明の板を用いてユーザの手元を覆い隠した状態にて行っている。この手法は、文字入力だけでなくポインティング等の何かしらの項目の選択、決定をアイズフリーにて行う際の性能評価に応用可能と考えられる。

謝辞

本研究を行うに当たっては、たくさんの方々による時間、金銭、精神的援助をいただきました。

指導教員の志築文太郎先生には、研究指導を行って頂き、また多くの相談に乗っていただきました。ご多忙の中、社会人博士課程に在籍する筆者に合わせて休日または平日の夜の遅い時間に研究指導、相談に臨機応変にご対応いただきました。博士論文を締めるところまで筆者が到達できたのは志築先生の御助力あってのことです。深く感謝申し上げます。

主査の三末和男先生、副査の葛岡英明先生、田中二郎先生、古川宏先生には本研究の予備審査の際に様々なご意見と知見を頂戴いたしました。特に田中二郎先生には筆者の学士課程、修士課程時代よりお世話になっており、キーボードにおける使用頻度の高いキーは上段に集中しているなど、Meyboard 以降のソフトウェアキーボードの設計を考えるうえで参考となる知見も提示していただきました。深く感謝申し上げます。

インタラクティブプログラミング研究室の皆様にも、研究活動において様々な意見を頂きました。また実験を行った際にも企業勤めのため行動に制約のあった自分の代わりとして、自身が忙しい身の上でありながらも、数日に渡り、長時間の実験に辛抱強く付きあってくださいましたことをお礼申し上げます。誠にありがとうございます。

埼玉大学の久野義徳先生には研究をしていく上で、パラメータの決定法などいろいろと意見をいただきました。久野先生は筆者の父でもあり、父としての立場からも生活面において筆者を大いに支えていただきました。家族として同様に私を支えてくれた母も含めて、両親には常に感謝しています。

最後に、私の会社と学生の二足の草鞋生活においてお世話になった様々な人々に感謝を告げて、謝辞を終わらせていただきます。本当にありがとうございました。

参考文献

[AWPL12] Shiri Azenkot, Jacob O. Wobbrock, Sanjana Prasain, and Richard E. Ladner. Input finger detection for nonvisual touch screen text entry in Perkinput. In *Proceedings of Graphics Interface 2012*, GI '12, pp. 121–129, 2012.

[BCO+12] Xiaojun Bi, Ciprian Chelba, Tom Ouyang, Kurt Partridge, and Shumin Zhai. Bimanual gesture keyboard. In *Proceedings of the 25th annual ACM symposium on User interface software and technology*, UIST '12, pp. 137–146, 2012.

[CAP+12] Tayfur Coskun, Eva Artinger, Lorenzo Pirrilli, Daniela Korhammer, Amal Benzina, Claudia Grill, Andreas Dippon, and Gudrun Klinker. Gestyboard: A 10-finger-system and gesture based text input system for multi-touchscreens with no need for tactile feedback. In *Proceedings of the 10th Asia-Pacific Conference on Computer-Human Interaction*, APCHI '12, pp. 701–702, 2012.

[CGF14] Xiang 'Anthony' Chen, Tovi Grossman, and George Fitzmaurice. 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, pp. 615–620, New York, NY, USA, 2014. ACM.

[CM08] Steven J. Castellucci and I. Scott MacKenzie. Graffiti vs. Unistrokes: An empirical comparison. In *Proceedings of the 26th international conference on Human factors in computing systems*, CHI '08, pp. 305–308, New York, NY, USA, 2008. ACM.

[CWA+13] Tayfur Coskun, Christian Wiesner, Eva Artinger, Amal Benzina, Patrick Maier, Manuel Huber, Claudia Grill, Philip Schmitt, and Gudrun Klinker. Gestyboard 2.0: A gesture-based text entry concept for high performance ten-finger touch-typing and blind typing on touchscreens. In *Proceedings of the 2nd International Conference on Human Factors in Computing and Informatics*, SouthCHI '13, pp. 680–691, 2013.

[DS10] Liam Don and Shamus P. Smith. Applying bimanual interaction principles to text input on multi-touch surfaces and tabletops. In *ACM International Conference on Interactive Tabletops and Surfaces*, ITS '10, pp. 253–254, 2010.

[For86] Steven Fortune. A sweepline algorithm for Voronoi diagrams. In *Proceedings of the second annual symposium on Computational geometry*, SCG '86, pp. 313–322. ACM New York, NY, USA, 1986.

[FS94] Masaaki Fukumoto and Yasuhito Suenaga. "FingeRing": A full-time wearable interface. In *Proceedings of the 12th international conference on Human factors in computing systems*, CHI '94, pp. 81–82, 1994.

[FW12] Leah Findlater and Jacob Wobbrock. Personalized input: Improving ten-finger touchscreen typing through automatic adaptation. In *Proceedings of the 30th international conference on Human factors in computing systems*, CHI '12, pp. 815–824, 2012.

[GBAT99] Mikael Goldstein, Robert Book, Gunilla Alsiö, and Silvia Tessa. Non-keyboard QWERTY touch typing: A portable input interface for the mobile user. In *Proceedings of the 17th international conference on Human factors in computing systems*, CHI '99, pp. 32–39, 1999.

[GCF15] Tovi Grossman, Xiang 'Anthony' Chen, and George Fitzmaurice. Typing on glasses: Adapting text entry to smart eyewear. In *Proceedings of the 17th International Conference on Human-Computer Interaction with Mobile Devices and Services*, MobileHCI '15, pp. 144–152, New York, NY, USA, 2015. ACM.

[GE07] Kentaro Go and Yuki Endo. CATKey: Customizable and adaptable touchscreen keyboard with bubble cursor-like visual feedback. In *Proceedings of the 11th IFIP TC 13 International Conference on Human-computer Interaction*, INTERACT '07, pp. 493–496, 2007.

[Gen83] Donald R. Gentner. Keystroke timing in transcription typing. In William E. Cooper, editor, *Cognitive Aspects of Skilled Typewriting*, pp. 95–120. Springer-Verlag, 1983.

[GKFS04] Nathan Green, Jan Kruger, Chirag Faldu, and Robert St. Amant. A reduced QWERTY keyboard for mobile text entry. In *Extended abstracts of the 2004 conference on Human factors and computing systems - CHI '04*, p. 1429, New York, New York, USA, 2004. ACM Press.

[GOZ16] Mitchell Gordon, Tom Ouyang, and Shumin Zhai. WatchWriter: Tap and gesture typing on a smartwatch miniature keyboard with statistical decoding. In *Proceedings of the 34th international conference on Human factors in computing systems*, CHI '16, pp. 3817–3821, New York, NY, USA, 2016. ACM.

[GPM10] Asela Gunawardana, Tim Paek, and Christopher Meek. Usability guided key-target resizing for soft keyboards. In *Proceedings of the 15th International Conference on Intelligent User Interfaces*, IUI '10, pp. 111–118. ACM New York, NY, USA, 2010.

[GR93] David Goldberg and Cate Richardson. Touch-typing with a stylus. In *Proceedings of the INTERACT '93 and CHI '93 Conference on Human Factors in Computing Systems*, CHI '93, pp. 80–87, New York, NY, USA, 1993. ACM.

[GT10] Kentaro Go and Leo Tsurumi. Arranging touch screen software keyboard split-keys based on contact surface. In *CHI '10 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '10, pp. 3805–3810, 2010.

[HOY80] Yuzuru Hiraga, Yoshihiko Ono, and Hisao Yamada. An assignment of key-codes for a Japanese character keyboard. In *Proceedings of the 8th Conference on Computational Linguistics*, COLING '80, pp. 249–256, 1980.

[JK09] Hyunjin Ji and Taeyong Kim. CLURD: A new character-inputting system using one 5-way key module. In *Proceedings of the 13th International Conference on Human-Computer Interaction*, HCI International 2009, Part III, pp. 39–47, 2009.

[KBW08] Shaun K. Kane, Jeffrey P. Bigham, and Jacob O. Wobbrock. Slide Rule: Making mobile touch screens accessible to blind people using multi-touch interaction techniques. In *Proceedings of the 10th International ACM SIGACCESS Conference on Computers and Accessibility*, ASSETS '08, pp. 73–80, 2008.

[KIG07] Thomas Költringer, Poika Isokoski, and Thomas Grechenig. TwoStick: Writing with a game controller. In *Proceedings of Graphics Interface 2007*, GI '07, pp. 103–110, 2007.

[KKKE11] Sungahn Ko, KyungTae Kim, Tejas Kulkarni, and Niklas Elmqvist. Applying mobile device soft keyboards to collaborative multitouch tabletop displays: Design and evaluation. In *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*, ITS '11, pp. 130–139, 2011.

[KSL+13] Sunjun Kim, Jeongmin Son, Geehyuk Lee, Hwan Kim, and Woohun Lee. TapBoard: Making a touch screen keyboard more touchable. In *Proceedings of the 31st international conference on Human factors in computing systems*, CHI '13, pp. 553–562, 2013.

[KST12] Yuki Kuno, Buntarou Shizuki, and Jiro Tanaka. Leyboard: A software keyboard that places keys at positions of fingers and their surroundings. In *Proceedings of the 10th Asia-Pacific Conference on Computer-Human Interaction*, APCHI '12, pp. 723–724, 2012.

[KST13] Yuki Kuno, Buntarou Shizuki, and Jiro Tanaka. Long-term study of a software keyboard that places keys at positions of fingers and their surroundings. In *Proceedings*

of the 15th International Conference on Human-Computer Interaction, HCI International 2013, Part V, pp. 72–81, 2013.

[LGYT11] Frank Chun Yat Li, Richard T. Guy, Koji Yatani, and Khai N. Truong. The 1Line keyboard: A QWERTY layout in a single line. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*, UIST '11, pp. 461–470, 2011.

[LYY+17] Yiqin Lu, Chun Yu, Xin Yi, Yuanchun Shi, and Shengdong Zhao. BlindType: Eyes-free text entry on handheld touchpad by leveraging thumb's muscle memory. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, Vol. 1, No. 2, pp. 18:1–18:24, June 2017.

[MJH+14] Anders Markussen, Mikkel Rønne Jakobsen, Kasper Hornbæk, Anders Markussen, Mikkel Rønne Jakobsen, and Kasper Hornbæk. Vulture: A mid-air word-gesture keyboard. In *Proceedings of the 32nd international conference on Human factors in computing systems*, CHI '14, pp. 1073–1082, New York, New York, USA, 2014. ACM Press.

[MMR10] Adiyan Mujibiya, Takashi Miyaki, and Jun Rekimoto. Anywhere touchtyping: Text input on arbitrary surface using depth sensing. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pp. 443–444, 2010.

[MS03] I. Scott MacKenzie and R. William Soukoreff. Phrase sets for evaluating text entry techniques. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '03, pp. 754–755, 2003.

[MZ97] I. Scott MacKenzie and Shawn X. Zhang. The immediate usability of Graffiti. In *Proceedings of the Conference on Graphics Interface '97*, pp. 129–137, Toronto, Ont., Canada, Canada, 1997. Canadian Information Processing Society.

[NR81] Allen Newell and Paul S. Rosenbloom. Mechanisms of skill acquisition and the law of practice. pp. 1–55. Lawrence Erlbaum Associates, 1981.

[OGN+11a] João Oliveira, Tiago Guerreiro, Hugo Nicolau, Joaquim Jorge, and Daniel Gonçalves. Blind people and mobile touch-based text-entry: Acknowledging the need for different flavors. In *The proceedings of the 13th international ACM SIGACCESS conference on Computers and accessibility*, ASSETS '11, pp. 179–186, 2011.

[OGN+11b] João Oliveira, Tiago Guerreiro, Hugo Nicolau, Joaquim Jorge, and Daniel Gonçalves. BrailleType: Unleashing braille over touch screen mobile phones. In *Proceedings*

of the 13th IFIP TC 13 international conference on Human-computer interaction - Volume Part I, INTERACT '11, pp. 100–107, 2011.

[One13] ZoomBoard: A diminutive QWERTY soft keyboard using iterative zooming for ultra-small devices. No. C, pp. 2799–2802, 2013.

[PSJ06] Morten Proschowsky, Nette Schultz, and Niels Ebbe Jacobsen. An intuitive text input method for touch wheels. In *Proceedings of the 24th international conference on Human factors in computing systems*, CHI '06, pp. 467–470, 2006.

[SCF+12] Caleb Southern, James Clawson, Brian Frey, Gregory Abowd, and Mario Romero. An evaluation of BrailleTouch: Mobile touchscreen text entry for the visually impaired. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, MobileHCI '12, pp. 317–326, 2012.

[Sei63] Robert Seibel. Discrimination reaction time for a 1,023-alternative task. *Journal of Experimental Psychology*, Vol. 66, No. 3, pp. 215–226, 1963.

[SFM08] Hannah Slay, Greg Foster, and Edison Mukadah. Investigating the viability of scroll-wheel interfaced mobile phones for text entry. In *Proceedings of the 2008 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on IT Research in Developing Countries: Riding the Wave of Technology*, SAICSIT '08, pp. 220–228, 2008.

[SLL11] Christian Sax, Hannes Lau, and Elaine Lawrence. LiquidKeyboard: An ergonomic, adaptive QWERTY keyboard for touchscreens and surfaces. In *Proceedings of the Fifth International Conference on Digital Society*, ICDS '11, pp. 117–122. XPS, 2011.

[The] The Oxford Math Center. Letter frequencies in english. `http://www.oxfordmathcenter.com/drupal7/node/353` (2018 年 8 月 1 日最終閲覧).

[Thi] Thingthing Ltd. Keyboard Fleksy: Customizable, fast, with gifs and colourful themes. `http://www.fleksy.com` (2018 年 9 月 22 日最終閲覧).

[WJJ+17] Eric Whitmire, Mohit Jain, Divye Jain, Greg Nelson, Ravi Karkar, Shwetak Patel, and Mayank Goel. DigiTouch: Reconfigurable thumb-to-finger input and text entry on head-mounted displays. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, Vol. 1, No. 3, pp. 113:1–113:21, September 2017.

[WMK03] Jacob O. Wobbrock, Brad A. Myers, and John A. Kembel. EdgeWrite: A stylus-based text entry method designed for high accuracy and stability of motion. In *Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*, UIST '03, pp. 61–70, New York, NY, USA, 2003. ACM.

[Yam80] Hisao Yamada. A historical study of typewriters and typing methods: From the position of planning japanese parallels. *Journal of Information Processing*, Vol. 2, No. 4, pp. 175–202, 1980.

[YGY+17] Chun Yu, Yizheng Gu, Zhican Yang, Xin Yi, Hengliang Luo, and Yuanchun Shi. Tap, dwell or gesture?: Exploring head-based text entry techniques for HMDs. In *Proceedings of the 35th international conference on Human factors in computing systems*, CHI '17, pp. 4479–4488, New York, NY, USA, 2017. ACM.

[YPC+17] Hui-Shyong Yeo, Xiao-Shen Phang, Steven J. Castellucci, Per Ola Kristensson, and Aaron Quigley. Investigating tilt-based gesture keyboard entry for single-handed text entry on large devices. In *Proceedings of the 35th international conference on Human factors in computing systems*, CHI '17, pp. 4194–4202, New York, New York, USA, 2017. ACM Press.

[ZK03] Shumin Zhai and Per-Ola Kristensson. Shorthand writing on stylus keyboard. In *Proceedings of the 21st international conference on Human factors in computing systems*, CHI '03, pp. 97–104, New York, NY, USA, 2003. ACM.

[ZK07] Shumin Zhai and Per-Ola Kristensson. Introduction to shape writing. In I. Scott MacKenzie and Kumiko Tanaka-Ishii, editors, *Text Entry Systems: Mobility, Accessibility, Universality*, pp. 139–158. Morgan Kaufmann, 2007.

[坂村 86] 坂村健. BTRON における入力方式 - TRON キーボードの設計 - . 情報処理学会研究報告ヒューマンコンピュータインタラクション（HCI）, Vol. 1986, No. 41, pp. 1–8, 1986.

[桜井 13] 桜井雄介, 増井俊之. QWERTY ソフトキーボード上のフリック日本語入力システム. 情報処理学会研究報告. HCI, ヒューマンコンピュータインタラクション研究会報告, Vol. 2013, No. 5, pp. 1–4, 2013.

[藤田 09] 藤田晋也, 赤池英夫, 角田博保. 運指情報と統計的手法を用いたウェアラブル機器向けの日本文入力手法の提案と評価. 情報処理学会研究報告. HCI, ヒューマンコンピュータインタラクション研究会報告, Vol. 2009, No. 28, pp. 9–16, 2009.

著者論文リスト

本研究に関連する論文

論文誌

[1] 久野祐輝, 志築文太郎, 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, 情報処理学会論文誌, Vol. 55, No. 4, 2014, pp. 1353-1364.

査読付き国際会議

[1] Yuki Kuno and Buntarou Shizuki. Meyboard: A QWERTY-Based Soft Keyboard for Touch-Typing on Tablets, Proceedings of 19th International Conference on Human-Computer Interaction (HCI International 2017), Vancouver, Canada, July 9-14, 2017, pp. 193-207.

[2] Yuki Kuno, Buntarou Shizuki, and Jiro Tanaka. Long-Term Study of a Software Keyboard that Places Keys at Positions of Fingers and their Surroundings, Proceedings of 15th International Conference on Human-Computer Interaction (HCI International 2013), Las Vegas, Nevada, USA, July 23-26, 2013, pp. 72-81.

その他の論文

[1] Yuki Kuno, Buntarou Shizuki, and Jiro Tanaka. Leyboard: A Software Keyboard that Places Keys at Positions of Fingers and their Surroundings, Proceedings of the 10th Asia-Pacific Conference on Computer-Human Interaction (APCHI2012), Matsue, Shimane, Japan, August 28-31, 2012, pp. 723-724.

本研究に関連しない論文

その他の論文

[1] 黒澤敏文, 久野祐輝, 小森谷大介, 志築文太郎, 田中二郎. タッチ UI におけるボタンの余白の大きさが操作に与える影響, 情報処理学会研究報告（156 回ヒューマンコンピュータインタラクション研究会）, Vol. 2014-HCI-156, No. 16, 2014, pp. 1-7.

[2] 久野祐輝, 大江龍人, 深津佳智, 志築文太郎, 田中二郎. 背面に触感を付与した携帯情報端末におけるタッチ精度の評価, 情報処理学会研究報告（150 回ヒューマンコンピュータインタラクション研究会）, Vol. 2012-HCI-150, No. 6, 2014, pp. 1-7.