平成 28 年度

筑波大学情報学群情報科学類

卒業研究論文

題目

タッチ検出を可能としたスペースキー上での親指スライドを用いたコマンド入力手法

主専攻　　ソフトウェアサイエンス主専攻

著者　　村田 主磨

指導教員　高橋伸 嵯峨智 志築文太郎

要　旨

本研究では，GUI メニューコマンドをスペースキー上での親指スライド操作によって選択する手法を提案する．アプリケーション内の「保存」や「取り消し」などの GUI メニュー上のコマンドを，親指スライド操作によって選択することで，ショートカットキーを覚えることなく，ホームポジションでのコマンドの選択を可能とする．本研究では，静電容量タッチセンサによるスペースキーの拡張を行い，テキストエディタなどのテキスト入力を主とするアプリケーションで本手法を利用できるよう，アプリケーションの GUI メニューを本手法によって操作できるシステムを作成した．また，本手法の評価のために，既存のコマンド選択方式との比較実験を行った．実験の結果，既存の手法より入力時間は多くかかったが，正解率ではショートカットキーの入力より有用な結果が得られた．

目 次

図目次

表 目 次

第1章　はじめに

本章では，本研究の背景，目的とアプローチ，貢献，および構成を述べる．

1.1　背景

テキスト入力や動画編集のアプリケーション，ブラウザなど，多くのアプリケーションでは，GUI メニューベースのコマンド入力 (図 1.1) が用いられる．コマンドを入力するには，マウスやタッチパッド等のポインティングデバイスの操作による，プルダウンメニューからの選択や，キーボードによるショートカットキー入力を利用する方法がある．一般的には，ポインティングデバイスによるメニュー選択が，その利用が容易であることから広く使われている．しかし，Microsoft Word[1]やテキストエディタのようなテキスト入力が主であるアプリケーションでは，キーボードの利用が多くなるが，メニューコマンド選択時には，キーボードからポインティングデバイス，およびポインティングデバイスからキーボードへと手を移動する，持ち替え時間がかかってしまう．また，ポインティングデバイスを利用する際，マウスカーソルを GUI メニューの位置まで移動させる，カーソル移動時間がかかってしまう．

中でも，持ち替え時間のかからないポインティングデバイスとして，ポインティングスティックというデバイスがある．ThinkPad には「トラックポイント」(図 1.2) として，キーボードのホームポジションから手を動かさずに，ポインティングを行うことができるポインティングスティックが搭載されている．しかし，ポインティングスティックでの操作は，カーソルが目的地にたどり着くまでポインティングスティックを傾け続ける動作を必要とし，マウスやタッチパッドに比べ，カーソル移動時間が多くかかってしまう．

一方，ショートカットキーによるコマンドの入力は，テキスト入力のようなアプリケーションの場合，キーボードからポインティングデバイスへの持ち替え時間や，カーソル移動時間を必要としない．さらに，1 回キーをタイプすることでコマンドの入力を実現できるという点で，ポインティングデバイスでの入力方法よりも速いインタラクションを行うことが可能である．しかし，コマンドが多くなるほど，対応するショートカットキーを覚えることが難しいため，ポインティングデバイスでのコマンド入力を行うユーザが多く，パフォーマンスの低下が示されている [1].

これらの問題に対して，持ち替え時間を必要としないデバイス，ショートカットキーの利用を支援する手法の研究などがされている．本研究では，持ち替え時間を必要とせず，ショートカットキーを覚えていない状態でもキーボード上に手を置いたまま，GUI メニューコマンドの入力を行うことができる手法を提案する．

[1] https://products.office.com/ja-jp/word

図 1.1: MacOS に組み込まれている GUI メニューの例

図 1.2: キーボード中心にポインティングスティックを搭載した ThinkPad

1.2 目的とアプローチ

1.1 節で述べた課題に対し，本研究ではスペースキー上での親指スライドを用いたコマンド入力手法を提案する．本手法はスペースキー上で親指を動かす操作であり，手をホームポジションに置いた状態での操作を行うことができ，かつキーボードとポインティングデバイス間の持ち替え時間を必要としない操作を可能とする．今回，本手法を実現させるために，スペースキー上がタッチされた時の静電容量によって親指の動作の検出を行うセンサを作成し，タッチ検出を可能とすることでスペースキーの拡張を行った．また，それをデバイスとして利用して操作を行うため，アプリケーション上の GUI メニューの動きと，スペースキー上のセンサの動きを紐付けるアプリケーションを作成した．

1.3 貢献

本研究の貢献を以下に示す．

- デバイス間の持ち替え時間を必要としない，スペースキー上での親指スライドによるメニューコマンド入力手法を提案した．
- スペースキーを拡張することにより，スペースキー上での 1 次元タッチ検出，およびスライド操作を可能とした．

- 本手法を用いて操作を行うため，スペースキー上での操作とアプリケーションのGUIメニューを紐付けるアプリケーションを作成した．
- 本手法と既存の手法の比較実験を行い，その考察を行なった．

1.4 本論文の構成

1章では，本研究の背景，目的とアプローチ，および貢献を示した．2章では，本研究の関連研究について述べる．3章では，本研究の提案する手法を述べる．4章では，本手法のセンサの実装について述べ，5章ではソフトウェア実装について述べる．6章で，実装したシステムを用いた評価実験について述べ，7章で本研究のまとめと今後の課題について述べる．

第2章　関連研究

本研究では，スペースキー上での親指スライド操作を用いたGUIメニューコマンドの入力手法を提案する．本章では，キーボードの操作の拡張を行った研究，メニューコマンドのパフォーマンスに関する研究，持ち替える動作を必要としないデバイスの研究，および静電容量センサに関する研究を示し，本研究との関係性を述べる．

2.1　センサの利用によるキーボードの操作の拡張

センサを用いて，キーボード上での操作を拡張する手法がいくつか提案されている．Dietzら [2] は，キーボードの各キーの下に圧力センサを埋め込み，ユーザがキーを押した強さによって，1つのキーで異なった入力を行うことを可能としている．Katoら [3] は，内臓マイクでキーボード上の手のスライドの音を取得することによって，手が左右どちらにスライドされたかを認識し，それをジェスチャの操作として用いる手法を提案している．FlickBoard[4] は，タッチセンサを組み込んだシリコンカバーをキーボード上に搭載し，キーボード表面上でのユーザのジェスチャ操作認識を行うことで，新たなジェスチャでの入力を可能としている．Taylorら [5] は，キーボードの各キーの隙間にフォトリフレクタを埋め込み，キーボード上での空中ジェスチャの認識を行うことで，その操作を可能としている．

本研究では銅箔テープを用い，横軸のタッチ位置の検出を可能とするセンサを作成し，それを利用することで，スペースキー上での入力を拡張することを可能とした．

2.2　メニューコマンドのパフォーマンスに関する研究

Jacobら [6] は，動画編集者は頻繁にキーボードショートカットを使うことを述べており，Knuthら [7] は，彼が頻繁にEmacsのショートカットを利用することを述べている．これらより，キーボードショートカットの利用も多くされているとわかる．キーボードショートカットは覚えることは難しい [1] が，一度覚えてしまえば良いパフォーマンスを発揮すると示されている [1][8]．しかし，ショートカットキーの入力はマウスよりも高速でない場合もある [9]．多くの場合にショートカットキーは高速な入力を可能とするが，多くのショートカットキーがあり，覚えることが難しいため，ポインタベースのショートカットキー入力を行うユーザが多い．この問題を解決するために，Sylvainら [9] は，コマンドキーなどの特定のキーを押した時に，メニューバーの内容を全て表示させる「Hotkey」として採用している．これにより，ユーザにどのショートカットキーがどのキーで使用することができるかを視覚的なフィードバックとして与えることで，ユーザの

ショートカットキーの利用を支援している．しかし，一度に目に入る情報量が多くなってしまう問題や，階層構造のような横幅を必要とするメニューへの拡張が困難である．また，Zheng ら [10] は，キーを押す指によって使用するコマンドを変える，ショートカットキーのコマンド数を増やす手法を提案している．

本手法は，特定のキーを押すことで，GUI メニューを開く機能を使用し，かつ GUI メニューをそのまま利用するため，ショートカットキーに割り当てられていないコマンドも使用できるようにする．

2.3 持ち替え時間を必要としないデバイス

キーボードとポインティングデバイス間の持ち替える動作を必要としないデバイスに関する手法が提案されている. 西村ら [11] は，タッチパネルに表示されたキーボード上で，タッチによるポインティング操作を行うことができるように拡張することで，持ち替える動作を必要としないデバイスを提案している．Rekimoto[12] は，キーボードに手を乗せた状態で親指をタッチパッド上で動かし，キーボードとタッチパッドを組み合わせたインタラクションについての研究を行なった．また，製品化されているデバイスとして，トラックポイントがある．これは，キーボード中心にポインタがあり，そのポインタを指で動かすことにより操作を行う．

本研究では，新たにセンサを取り付けることで，ポインティングデバイスとの持ち替える動作の問題を解決する．

2.4 静電容量タッチセンサに関する研究

静電容量を用いたタッチ認識に関する研究がある．Tsuruta ら [13] は，銀ナノインクで印刷された複数のオブジェクトの中から，どのオブジェクトがタッチされたかを，一本の導線で認識することを可能としている．Tobias ら [14] は，静電容量センサを複数用いて，手の位置，およびマルチタッチを認識する研究を行った．Tomita ら [15] は，静電容量センサを用いて，机の上でのタッチジェスチャを行うインタフェースの研究を行なった．本研究では複数の銅箔テープをタッチセンサとして用いることで，スペースキー上でのタッチ認識を行う．

第3章　スペースキー上での親指スライドを用いたコマンド入力手法

本手法は，スペースキー上で親指を横にスライドさせることにより，GUI メニューのコマンド (図 3.1) を選択する手法である．ショートカットキーを覚えることを必要とせず，ポインティングデバイスへの持ち替え時間がかからない特徴を有す．

本章では，本手法を用いたメニュー項目の 2 つの選択方式について述べ，階層化されたメニューの操作について述べる．その後，本手法の利用イメージについて述べる．

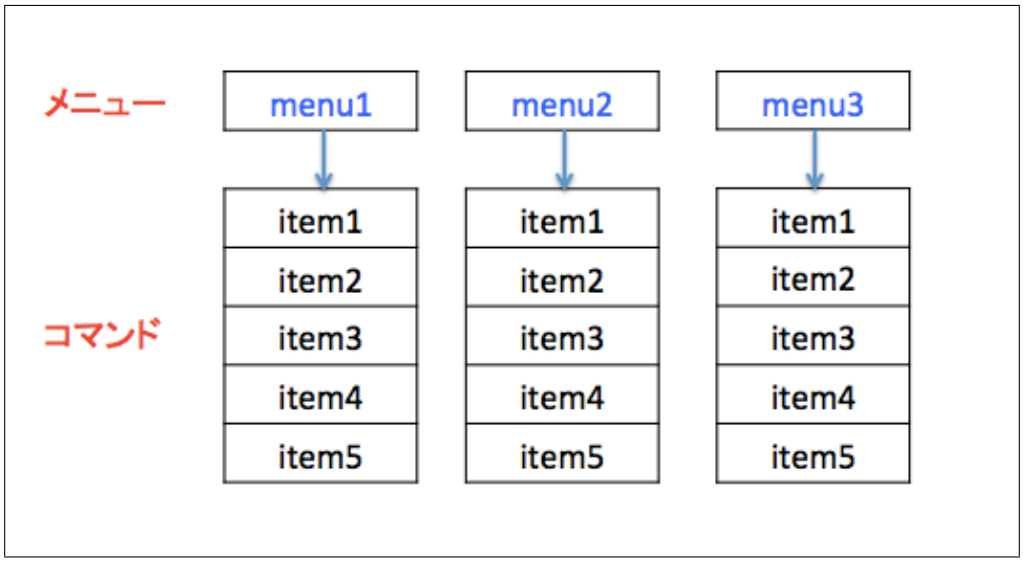

図 3.1: 一般的な GUI メニューの構造

3.1 選択方式 1: 1 段階選択

選択方式 1(図 3.2) は，1 段階の選択である．全てのコマンドを 1 次元の列の並びとみなし，コマンドの選択を行う．

3.1.1 操作の流れ

操作は以下の 3 ステップから成る．

1. 特定のキー（以後 HOLDKEY とする）を押すと，最左のメニューが開き，最上のコマンドが選択状態になる．
2. 親指スライドにより，メニューアイテムの選択を行う．
3. HOLDKEY を離すと，選択されているメニューアイテムに決定される．

3.1.2 メニュー項目の遷移

親指スライドによるメニューアイテムの選択は以下のように成る．

- HOLDKEY が押され，最左のメニューが開かれると，その最上の項目に選択カーソルがある．
- 親指を右にスライドさせると，選択カーソルは下に移動する．
- 親指を左にスライドさせると，選択カーソルは上に移動する．
- 選択カーソルがメニューの最下にあり，さらに次に移動する場合は，右隣のメニューが開き，その最上に選択カーソルが移動する．
- 選択カーソルがメニューの最上にあり，さらにその前に移動する場合は，左隣のメニューが開き，その最下に選択カーソルが移動する．
- 選択カーソルが最右のメニュー上の最下にあり，さらにその次に移動する場合は，最左のメニューが開き，その最上に選択カーソルが移動する．
- 選択カーソルが最左のメニュー上の最上にあり，さらにその前に移動する場合は，最右のメニューが開き，その最下に選択カーソルが移動する．

親指スライドによる選択は 1 次元の列を相対的に移動するため，一度の親指スライドで目的のメニューに辿り着かない場合は，何度か親指をスライドさせることで，その項目への移動を行う．

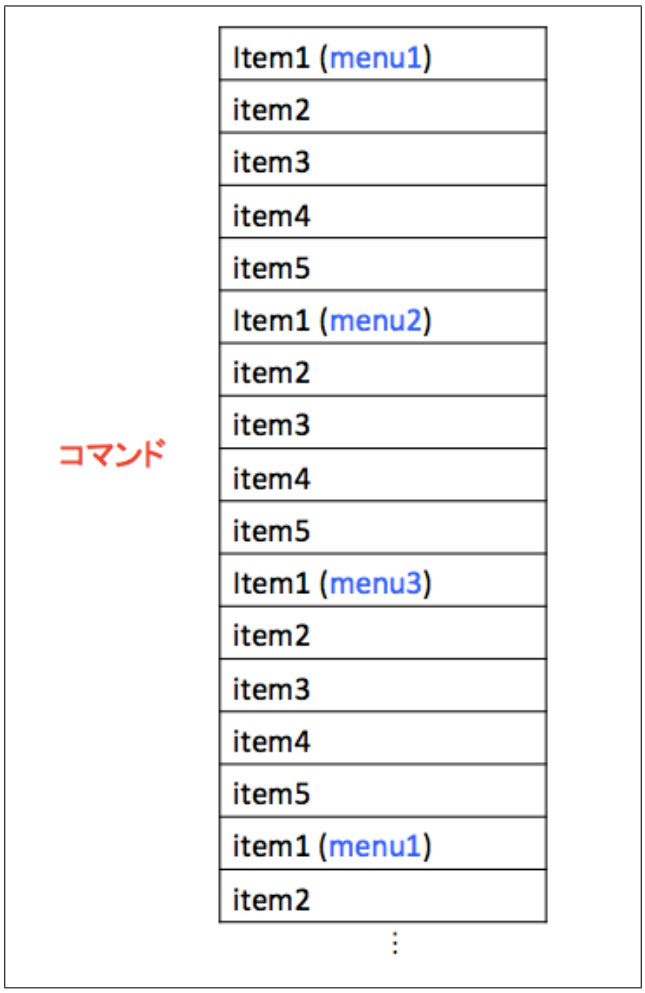

図 3.2: 選択方式 1: メニューコマンドを 1 次元の列とみなした状態

3.2 選択方式 2: 2 段階選択

選択方式 2(図 3.3) は，2 段階の選択である．全てのコマンドを 2 次元の行列とみなし選択を行う．列がメニューに対応しており，行がコマンドの並びに対応しており，列を選んだ後に行を選ぶという流れでの入力を行う．

3.2.1 操作の流れ

操作は以下の 4 ステップから成る．

1. HOLDKEY を押すと，メニューを選択する状態になる．
2. 親指スライドにより，メニュー (列) の選択を行う．この時，選択状態にあるメニューが開いている状態になる．
3. HOLDKEY を押し直すと，選択されているメニューの最上のコマンドが選択状態になり，コマンドの選択に移る．
4. 親指スライドにより，メニューコマンド (行) の選択を行う．
5. HOLDKEY を離すと，選択されているコマンドに決定される．

3.2.2 メニュー項目の遷移

親指スライドによるメニューアイテムの選択は以下のようになる．

- メニュー選択時，親指を右にスライドさせると，選択カーソルは右に移動する．
- メニュー選択時，親指を左にスライドさせると，選択カーソルは左に移動する．
- メニュー選択時，選択カーソルが最左の状態でさらに左に，または選択カーソルが最右の状態でさらに右に移動させようとした場合，選択カーソルは動かない．
- コマンド選択時，親指を右にスライドさせると，選択カーソルは下に移動する．
- コマンド選択時，親指を左にスライドさせると，選択カーソルは上に移動する．
- コマンド選択時，選択カーソルが最上でさらに上に，または選択カーソルが最下でさらに下に移動させようとした場合，選択カーソルは動かない．

選択方式 1 と同様に，メニューおよび，コマンド選択時には行または列を相対的に移動するため，一度の親指スライドで目的のメニューに辿り着かない場合は，何度か親指をスライドさせることで，その項目への移動を行うことができる．

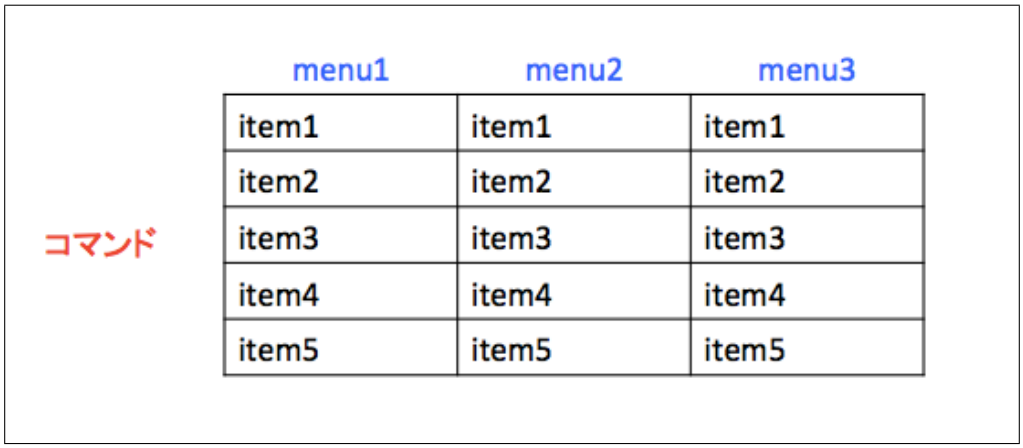

図 3.3: 選択方式 2: メニューコマンドを 2 次元の行列とみなした状態

3.3 階層化されたメニューの遷移

多くのアプリケーションには階層化されたメニュー (図 3.4) が備わっているため，その選択方法を以下に示す．

- 階層化されたメニューアイテム上で HOLDKEY を離すと，その子階層のメニューが開き，再度 HOLDKEY を押すことでその最上のメニューアイテムが選択状態となる．
- 階層化されていないメニューアイテム，および子階層の存在しないメニューアイテム上で HOLDKEY を離した場合は，そのアイテムに決定される．
- 子階層のメニュー上で親指を右にスライドさせると，選択カーソルは下に移動する．
- 子階層のメニュー上で親指を左にスライドさせると，選択カーソルは上に移動する．
- 子階層のメニュー上で選択カーソルが最上よりさらに前に，または最下よりさらに次に移動することはない．

以上が階層化されたメニューの遷移の流れとなる．

3.4 利用イメージ

本手法の利用イメージを図 3.5，図 3.6 に示す．アプリケーションの利用者が任意のタイミングで HOLDKEY を押すと，メニューが開き，親指スライドでのコマンドの選択を行うことができる．

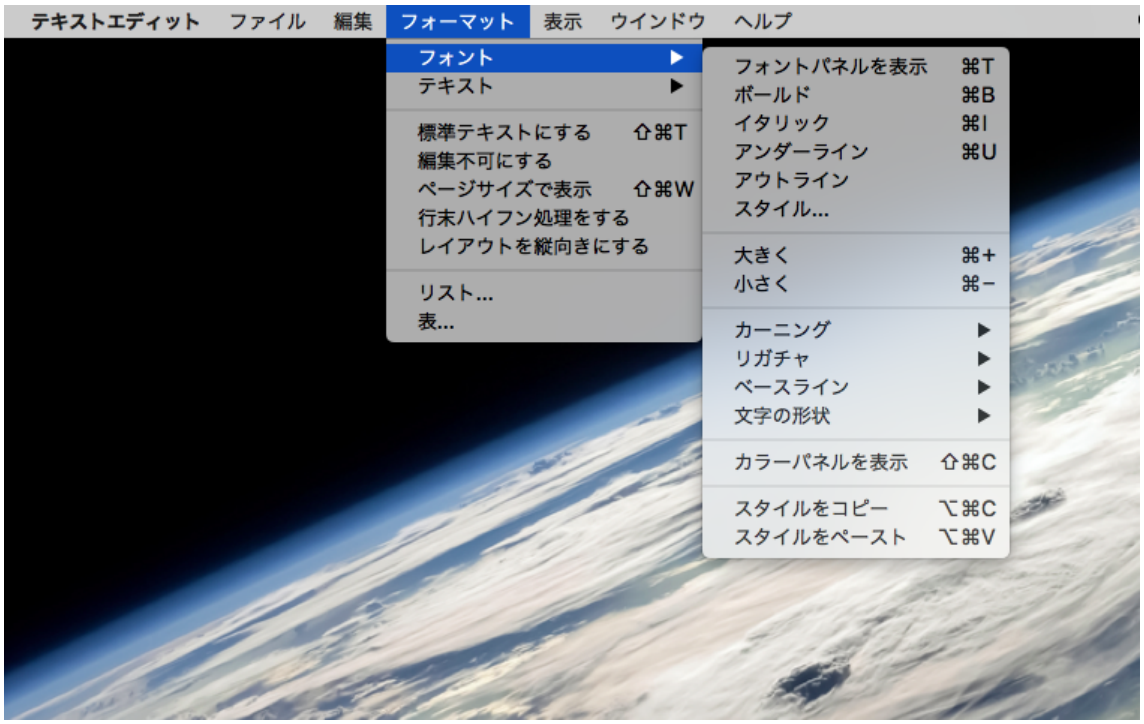

図 3.4: 階層化されたメニュー

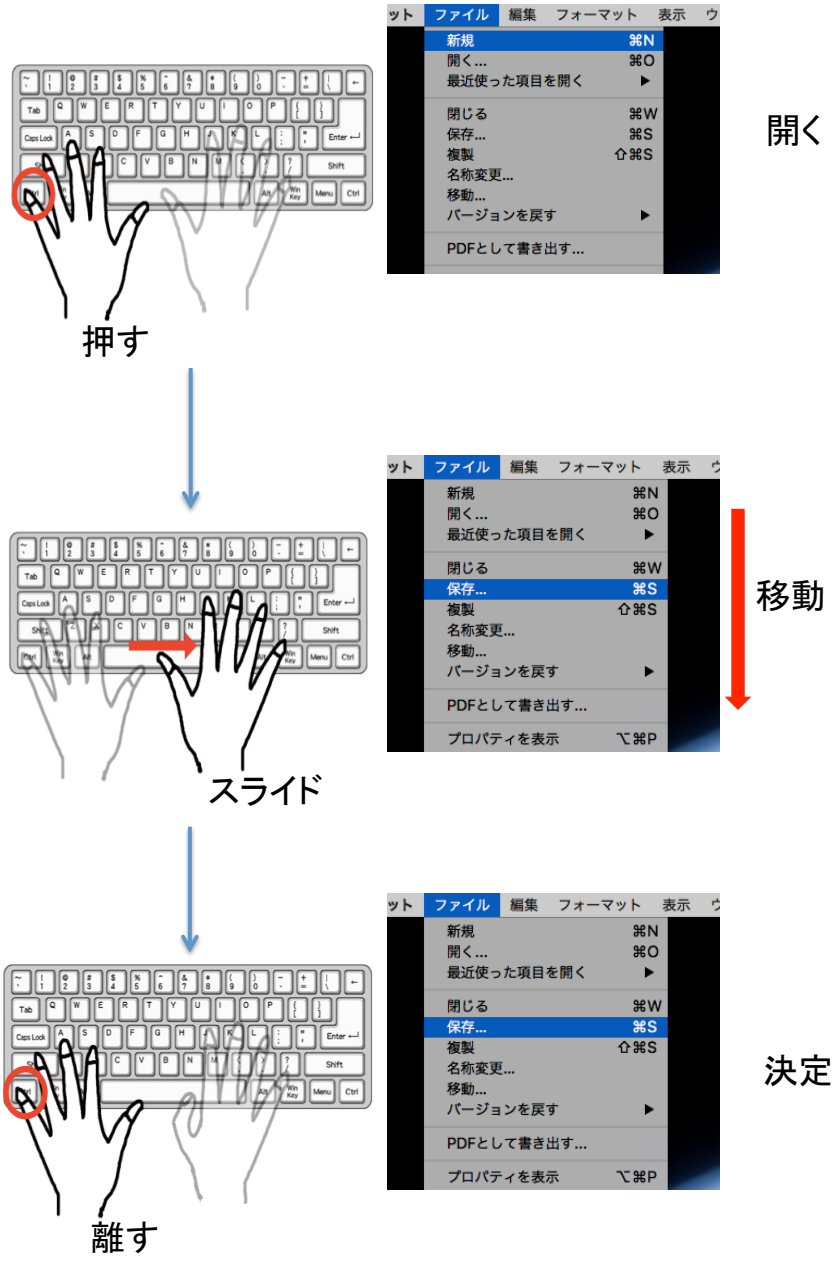

図 3.5: 本手法の利用イメージ図

図 3.6: 本手法の利用イメージ写真

第4章　親指の動きを認識するセンサの実装

スペースキー上での親指のタッチ位置の検出，および動きを検出するためのセンサを作成した．本章では，静電容量を用いたタッチ検出の仕組み，実装するセンサのインタフェース，センサの回路および設計，1 次元タッチ位置の認識アルゴリズム，実装に用いた装置およびライブラリ，センサから得られる値と各パラメータの決定について述べ，2 つの予備実験とその結果を示す．その後，実装した視覚的にタッチ位置を確認する描画アプリケーション，2 点タッチの実装について述べる．

4.1　静電容量によるタッチ検出の仕組み

本研究では静電容量センサに銅箔テープを用いる．銅箔テープに指が触れると，指と銅箔テープの間に微弱な電流が生じる．この電流が静電容量であり，指と銅箔テープの間にコンデンサが生成されるため，静電容量が発生する．発生した静電容量の変化によりタッチされたかどうかを検出する．銅箔テープに指を近づけた時，およびタッチした時には静電容量が増加し，銅箔テープから指を遠ざけると静電容量は減少する．静電容量 $C[F]$ は，触れる指の面積 $S[m^2]$，指と銅箔テープの距離 $d[m]$ とし，真空の誘電率を ε とする時，

$$C = \varepsilon \frac{S}{d} \tag{4.1}$$

と表され，静電容量の大きさはタッチしている指の面積に比例する．

4.2　実装するセンサのインタフェース

センサの実装，アプリケーションの実装にあたり，試作機として擬似スペースキー (図 4.1) を作成した．擬似スペースキーは，3D プリンタを用いてスペースキーに見立てた直方体を印刷し，その上に 5 枚の銅箔テープを貼り付けた．それぞれの銅箔テープの形は，三角形および平行四辺形である．触れる位置によって指と銅箔テープの接触する面積が異なるため,，タッチされた位置を取得することが可能となる．試作機を用いて予備実験等を行った後，完成版としてキーボードのスペースキー上に 5 枚の銅箔テープを付けた (図 4.2).

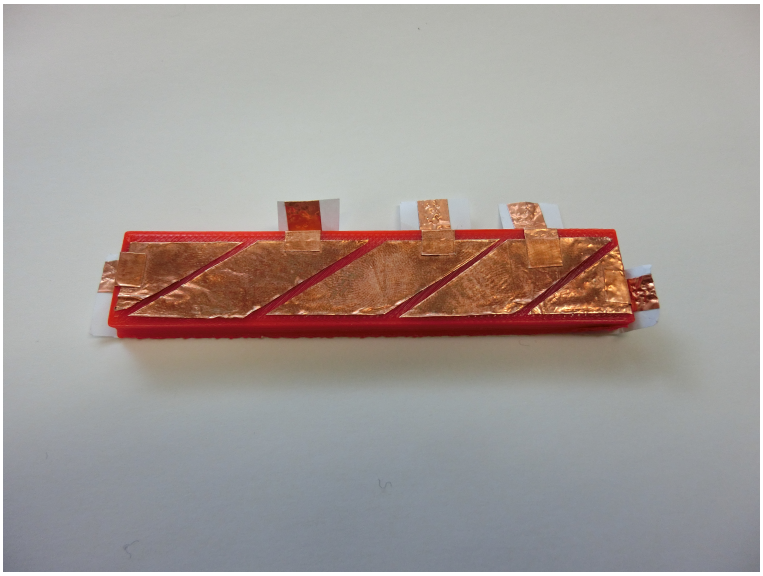

図 4.1: 試作機である擬似スペースキー

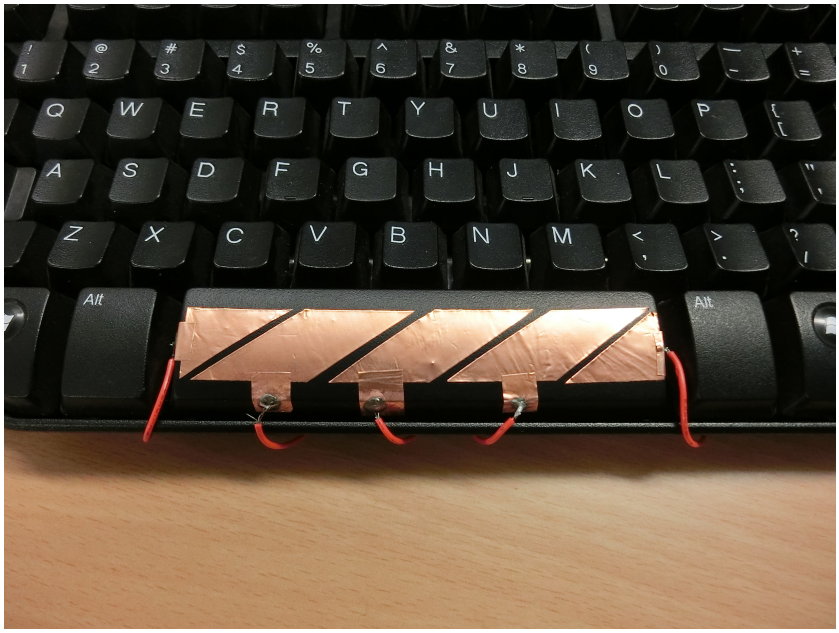

図 4.2: スペースキー上にタッチセンサを組み込んだキーボード

4.3 回路および設計

作成した回路の回路図を図 4.3 に示す．銅箔テープとマイコンの間の抵抗には $1M\Omega$ の抵抗を用い，マイコンからは 10 ミリ秒毎に値を読み取った．

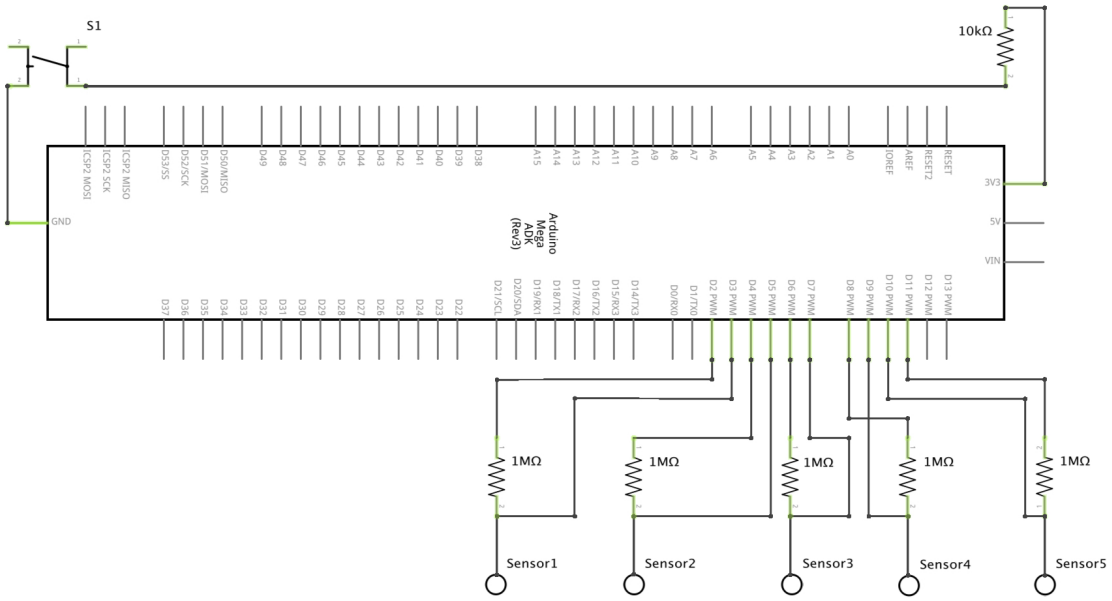

図 4.3: デバイスの回路

4.4 1 次元タッチ位置の認識アルゴリズム

スペースキー上に付けた 5 枚の銅箔テープから，親指のタッチ位置の検出を行う．アルゴリズムは以下のようになる．

1. それぞれの銅箔テープの要素番号 $index$ を左から $[1, 2, 3, 4, 5]$ とする．
2. 銅箔テープのタッチされていない時の静電容量，タッチされた時の最大の静電容量をそれぞれ $Min[index]$, $Max[index]$ とする．
3. 取得した静電容量を $C[index]$ とし，以下の式でそれぞれの銅箔テープの静電容量を正規化した値 $C'[index]$ を得る．

$$C'[index] = \frac{C[index] - Min[index]}{Max[index] - Min[index]} \tag{4.2}$$

この正規化された $C'[index]$ は，銅箔テープがタッチされた時，最大の静電容量に対し，静電容量が何%であるか示す値である．

4. 以下の式で，それぞれの銅箔テープの重心 x を求める．

$$x = \frac{\sum_{i=1}^{5} C'[i] * 10i}{\sum_{j=1}^{5} C'[j]} \tag{4.3}$$

重心 x がタッチされた位置を示す値であり，10〜50 の数値で示される．

4.5 実装に用いた装置およびライブラリ

センサの実装に，4.2 で述べた，5 枚の銅箔テープ用いた．マイコンは Arduino MEGA を用いた．プログラムで銅箔テープからの静電容量の取得を行うために，Arduino のライブラリである CapacitiveSensor[1] を使用した．さらに，$Min[index]$, $Max[index]$ を設定を行うためのキャリブレーションをする必要があり，そのためのタクトスイッチを回路に組み込んだ．コンピュータは，Macbook Air 13 インチ (MacOS Sierra) を使用し，拡張するキーボードは Dell[2] 製の US 配列キーボードを用いた．また，銅箔テープ上を直接指で触れると，静電容量の値がオーバーフローを起こし，正確な値が読み取れないため，銅箔テープ上に紙を敷くことで対応した．使用した回路およびキーボード写真を図 4.4 に示す．

図 4.4: 回路およびキーボード写真

[1] http://playground.arduino.cc/Code/CapacitiveSensor
[2] http://www.dell.co.jp/

4.6　センサから得られる値と各パラメータの決定

実装したセンサから親指のタッチ位置の値を読み取った．得られた値から，ノイズとして値にぶれ幅があることがわかったため，フィルタによる平滑化を行った．また，タッチ位置を視覚的に確認するためのアプリケーションを実装し，タッチ位置の検出ができていることを確認した．

4.6.1　キャリブレーション

銅箔テープ上をタッチした時の静電容量は，環境や扱う人の指によって毎回異なる．また，各銅箔テープによってもタッチしていない時の静電容量，タッチされた時の最大の静電容量は異なる値となる．そのため，システムを使用する前に，センサに対してキャリブレーションを行う必要がある．そのため，キャリブレーションを行うシステムを実装した．

銅箔テープをタッチしていない状態で，ブレッドボード上のタクトスイッチを押すと，その状態での各銅箔テープ上の静電容量が $Min[index]$ として保存され，一度センサ上全体を指で撫でることで各銅箔テープは，触れられた中で最も大きい静電容量の値として $Max[index]$ を設定する．

4.6.2　リアルタイムで得られる値

親指のタッチ位置の結果をグラフ (図 4.5) として得た．(1) 親指を左から右に一往復させた時と，(2) 親指を動かさず同じ部分をタッチし続けた時の結果が表示されている．グラフより，タッチ位置を読み取った時，タッチした状態で指を動かさない状態であっても，読み取った値にノイズによるぶれ幅があることがわかった．これは指を動かしていない状態でも，静電容量の大きさに微妙な変化が現れるためである．タッチしていない状態でも同様である．この問題を解決するために，平滑化によるノイズの除去を行った．

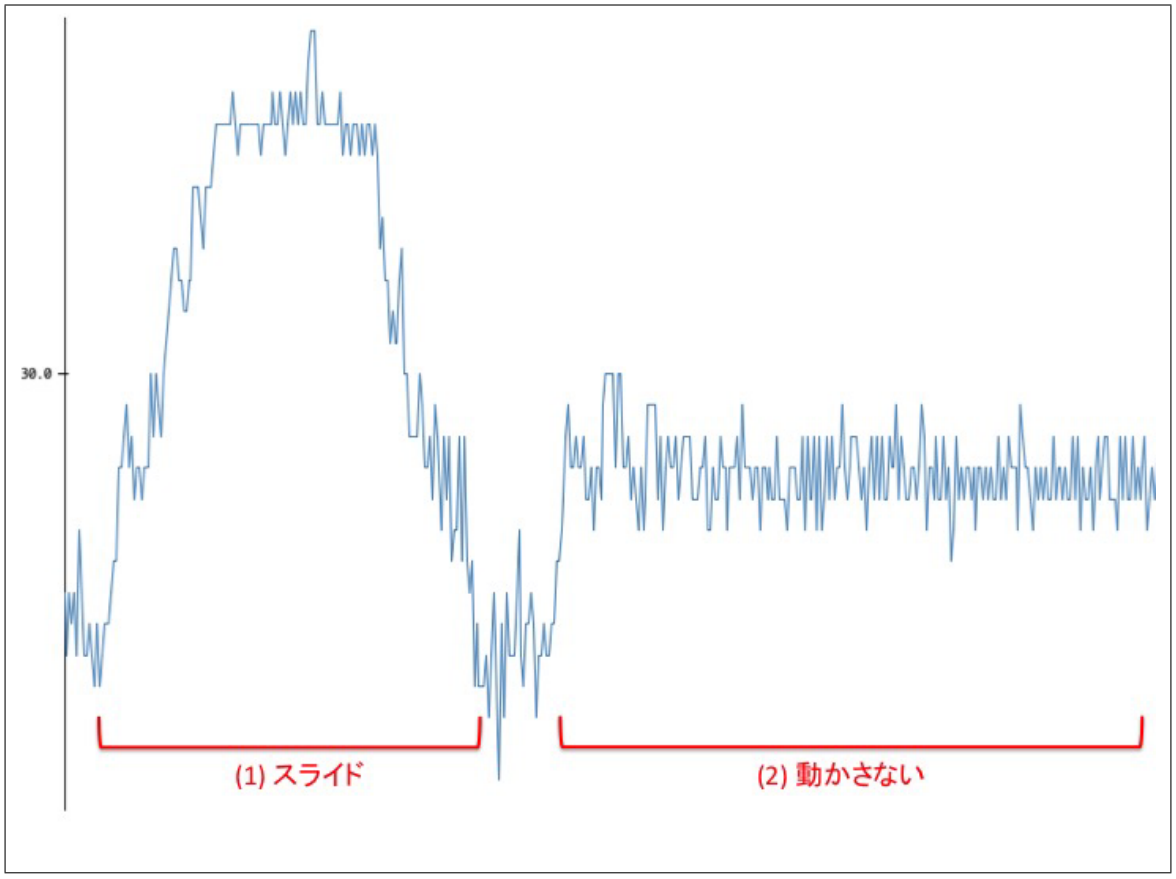

図 4.5: リアルタイムで得られるタッチ位置のグラフ

4.6.3　移動平均フィルタを用いた平滑化によるノイズの軽減

ノイズを除去するための平滑化に，移動平均フィルタを用いた．移動平均フィルタは，連続した N 個の入力の平均を取ることでノイズを軽減するフィルタである．この時の N をフィルタ係数と呼ぶ．今回は，5 枚の銅箔テープそれぞれの静電容量に対して平滑化を行い，移動平均フィルタにはプログラム上で配列 (図 4.6) を用いた．そのため，フィルタ係数 N がフィルタ用配列の配列長となる．静電容量の入力値を配列に格納しておき，新しい入力が得られた時には最も古い値を配列から除去する (図 4.7)．フィルタ係数 N を大きくするほど，ノイズは軽減されるが，入力値が反映されるまでの遅延時間が長くなる．

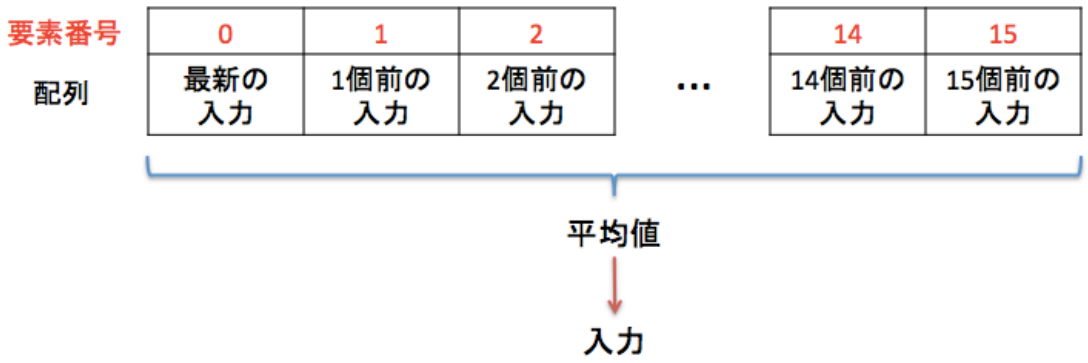

図 4.6: フィルタとして用いる配列

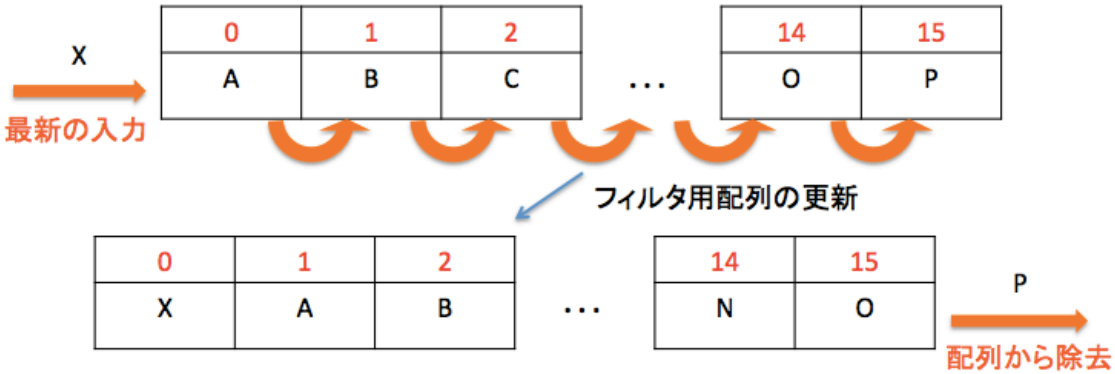

図 4.7: フィルタ用配列の更新のイメージ

4.7　予備実験 1: フィルタ係数の決定のための実験

銅箔テープから得られる静電容量の値に生じるノイズの平滑化を行う．平滑化に用いる移動平均フィルタのフィルタ係数を求める必要がある．今回は，$[5, 10, 15, 20, 25]$ の 5 つのフィルタ係数に対して，シリアルプロッタでのグラフを結果とし，遅延時間を考慮するとともにフィルタ係数の決定を行った．

4.7.1　ループ毎の読み取り時間

マイコンから 1 回のループで値を読み取る時間を測定を行った．Arduino の millis メソッドを用い，ループ毎にそれまでの経過時間を測る．その経過時間から前のループまでの経過時間と，ループ毎の Delay を引くことで求められる．1 回のループは 9 ミリ秒 - 10 ミリ秒で処理されていることがわかった．

4.7.2　遅延時間

移動平均フィルタで N 回の入力の平均を入力とする．そのため，(ループの $Delay$) × (ループ毎の読み取り時間) × (フィルタ係数 N) が，入力が反映されるまでの遅延時間である．今回は 9 ミリ秒 - 10 ミリ秒毎に静電容量を読み取るので，それぞれのフィルタ係数に対する遅延時間は表 4.1 のようになる．

表 4.1: フィルタ係数と遅延時間

フィルタ係数 N	遅延時間 [ミリ秒]
5	95 - 100
10	190 - 200
15	285 - 300
20	380 - 400
25	475 - 500

4.7.3 読み取り結果

(1) 親指を左から右にスライドさせた時と，(2) 親指を動かさずに同じ部分をタッチし続けた時のタッチ位置の変化を，図 4.8 - 図 4.13 に示す．

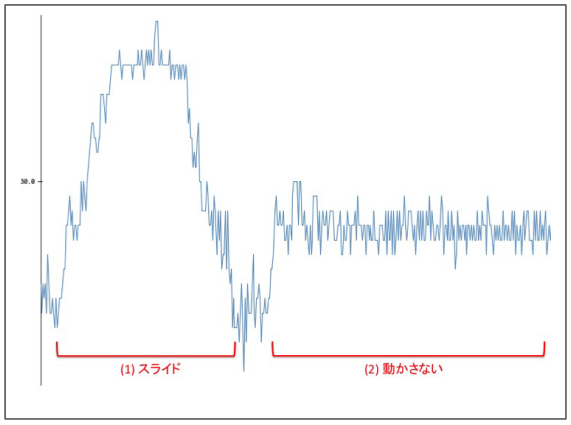

図 4.8: 平滑化無しの時のタッチ位置の変化

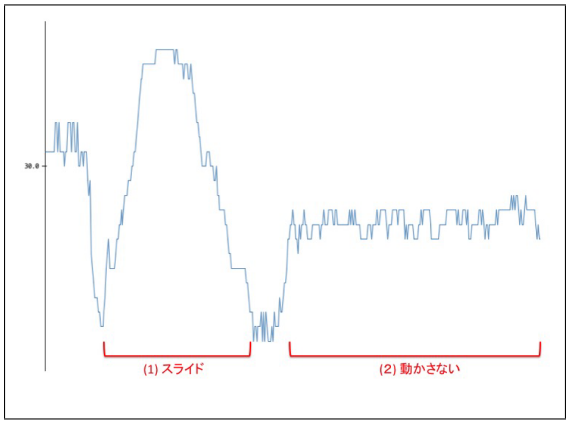

図 4.9: $N = 5$ の時のタッチ位置の変化

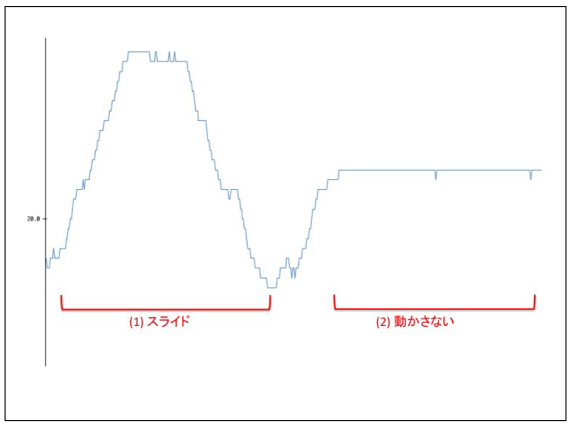

図 4.10: $N = 10$ の時のタッチ位置の変化

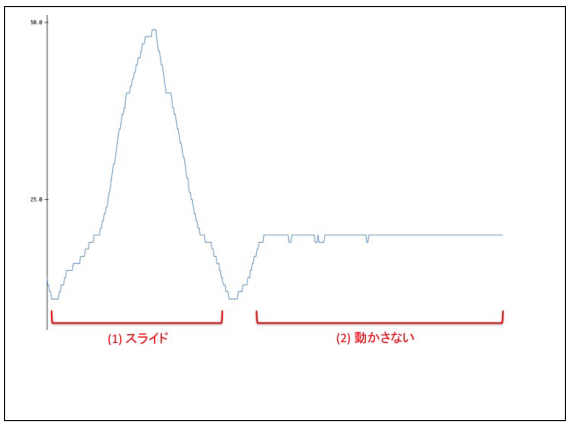

図 4.11: $N = 15$ の時のタッチ位置の変化

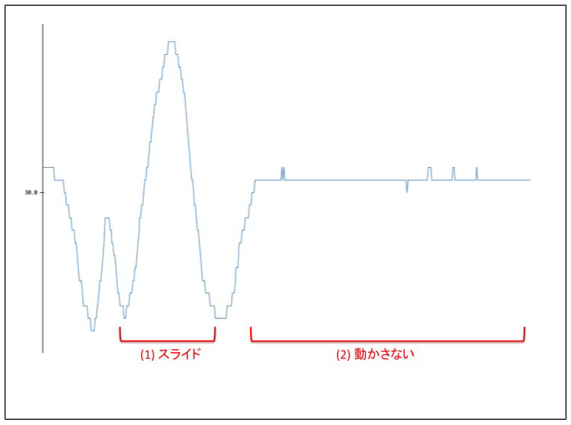

図 4.12: $N = 20$ の時のタッチ位置の変化

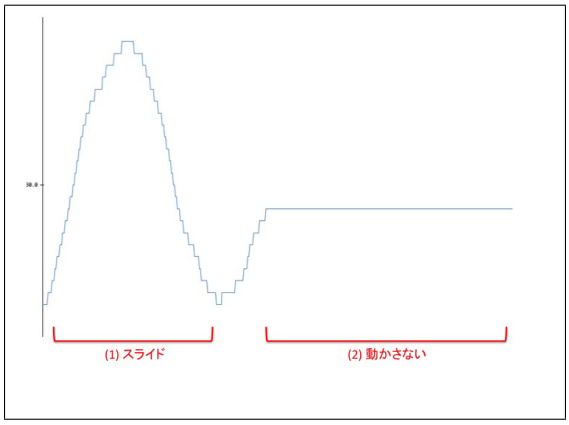

図 4.13: $N = 25$ の時のタッチ位置の変化

4.7.4 フィルタ係数の決定

以上の結果より，$N = 5$ の時にはノイズによるタッチ位置のぶれは頻繁に起こっていることがわかる．$N = 10$ の時にはスライド操作と同じ位置をタッチし続けた時のどちらにも若干のぶれが生じていることがわかった．$N = 15$，$N = 20$ の時には，同じ位置をタッチし続けた時に少々ぶれが生じていた．$N = 25$ の時には殆どぶれは生じていなかった．遅延時間に関して，著者の使用感から，$N = 15$ の時までは指を動かしてからタッチ位置の結果がグラフに反映されるまで，遅いと感じられなかった．しかし，$N = 20, N = 25$ の時に関しては遅延時間によるタッチ位置の反映が遅いと感じられた．以上の結果を踏まえ，フィルタ係数 N を 15 に設定した．

4.8　予備実験 2: タッチ判定の閾値に関する実験

スペースキー上がタッチされていない時には動作として何も起きないことが望ましい．そのため，タッチされているかどうかの判定を行う必要がある．タッチ判定に必要となる閾値を予備実験によって求めた．

4.8.1　タッチ判定に使用する値

4.4 節で述べた，それぞれの銅箔テープの正規化の値の和によってタッチ判定を行う．この時使用する正規化の値は，フィルタによって平滑化した値ではなく，最新の入力の値である．

最新の入力の正規化値の和をシリアルプロッタによって確認したところ，図 4.14 のようなグラフが得られた．このグラフから，タッチ判定の閾値を 60 に設定して，タッチ判定が正確に行われているかどうかを確認する．

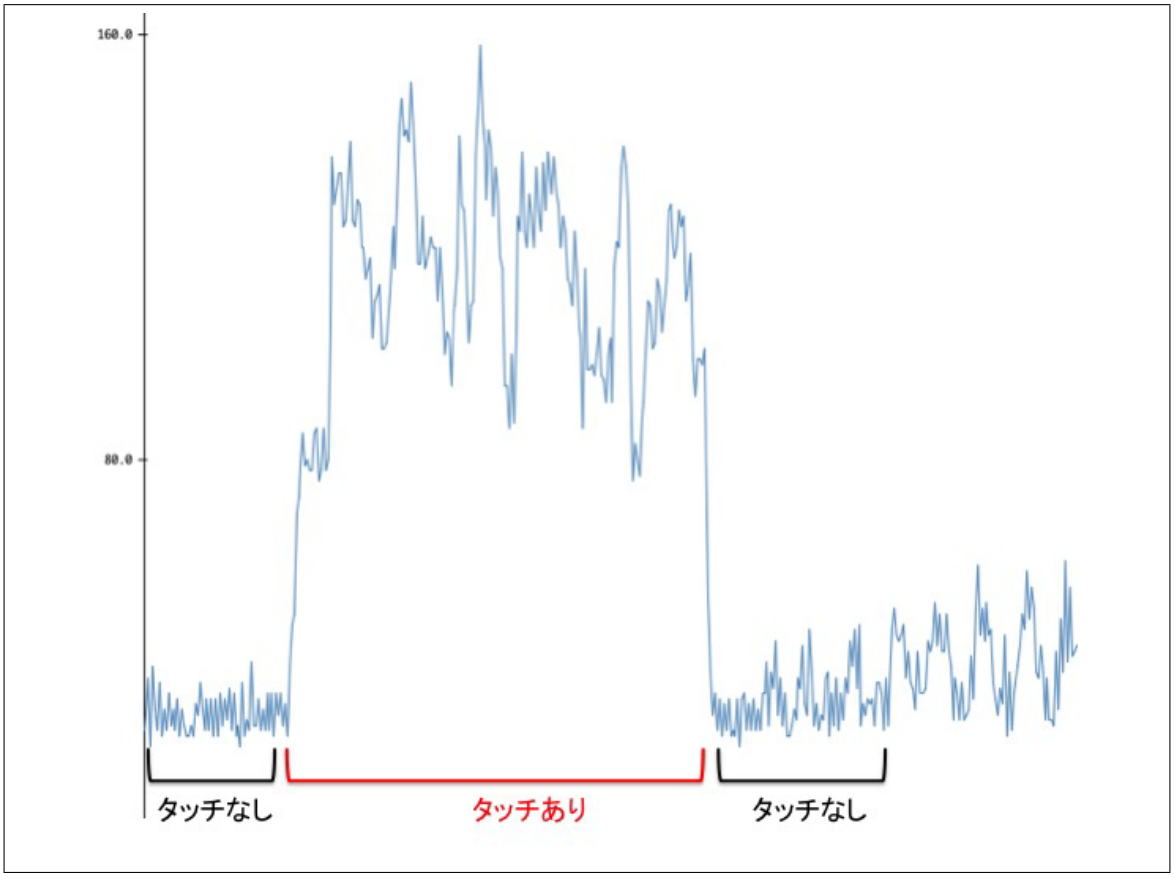

図 4.14: 正規化値の和のグラフ出力

4.8.2　結果

正規化値の和が 60 以上の時に 1，60 未満の時に 0 を出力するプログラムによって，図 4.15 のグラフが結果として得られた．赤く示している区間が実際にタッチした時の出力の部分であり，黒く示している区間が実際にタッチしていない時の出力の部分である．グラフより，閾値 60 でタッチ判定が正確に行えることがわかったため，閾値を 60 として採用した．

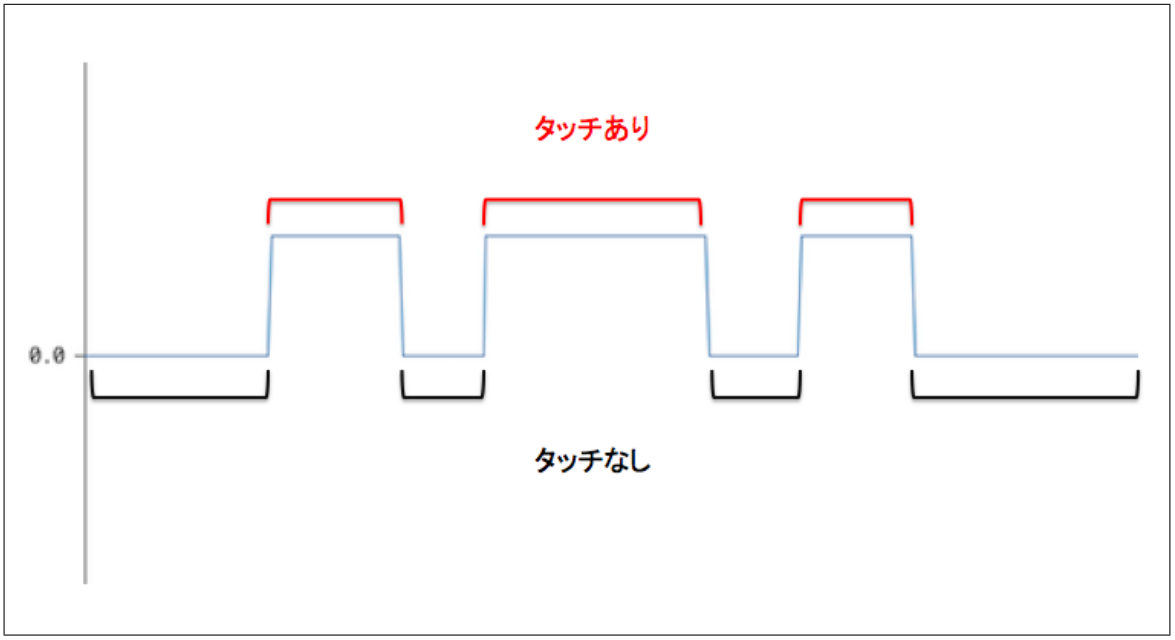

図 4.15: 閾値によるタッチ判定のグラフ出力

4.9　タッチ位置のフィードバック

タッチ位置や，指の動きを視覚的フィードバックとして表示する GUI アプリケーション (図 4.16) を作成した．実装の言語には Python を用い，GUI 描画用のライブラリとして wxPython[3]，シリアル通信用のライブラリとして pyserial[4]を用いた．アプリケーションでは，画面の左から右にかけて，赤い四角で囲まれている部分に，5 つの青い円でそれぞれの銅箔テープの正規化された静電容量を，円の大きさとして表す．また，センサのどの位置がタッチされているかを，下部の緑の円の x 軸方向の位置として表す．作成したアプリケーションによって，正しく親指のタッチ位置，動きが認識されていることが確認できた．

[3]https://wxpython.org/
[4]https://pypi.python.org/pypi/pyserial

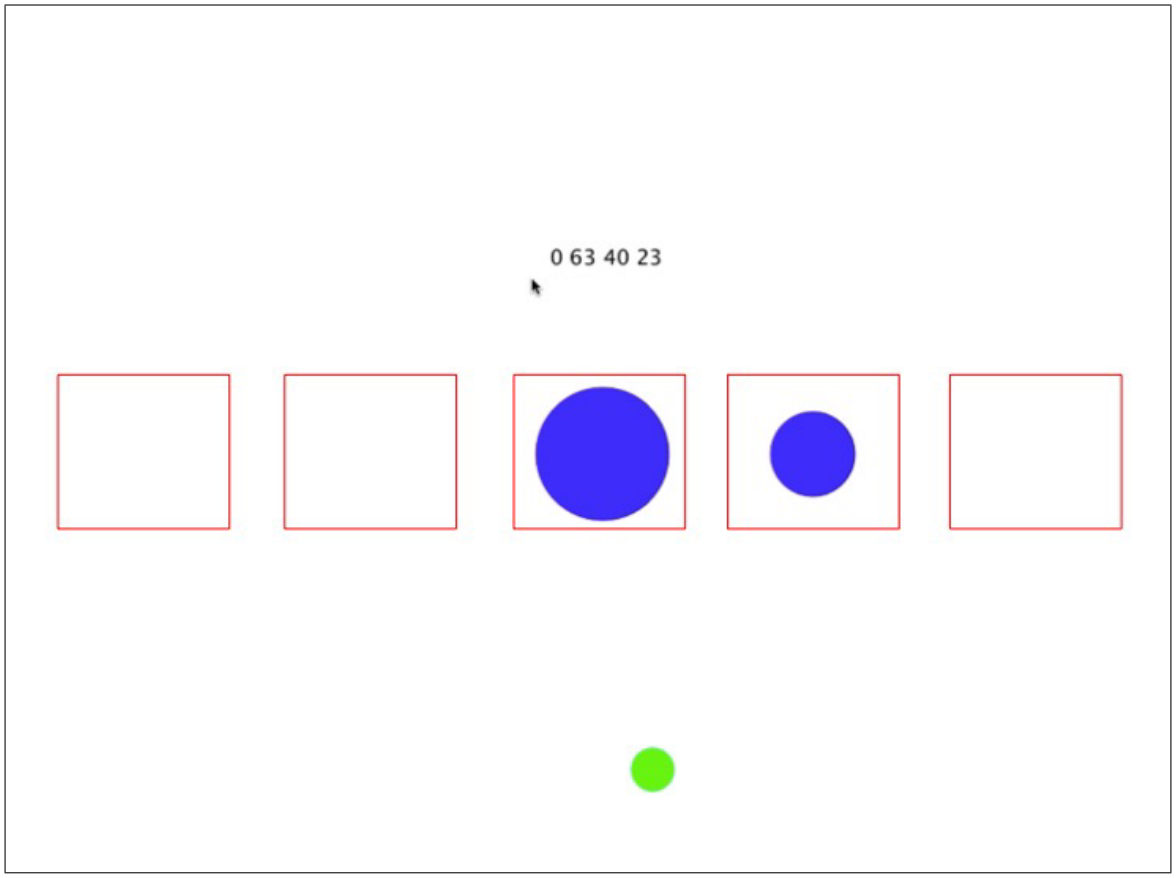

図 4.16: タッチ位置確認用アプリケーション

4.10　2 点タッチ検出

本研究では 2 点タッチを用いるコマンド選択は行わないが，両手の親指でスペースキーに触れるダブルタッチの検出が可能となった．それについて本節で述べる．

4.10.1　予備実験 3: 2 点タッチ検出の閾値決定のための実験

4.8 節と同様に，各銅箔テープの正規化された静電容量の和の値によってダブルタッチを検出した．(1) 親指一本でタッチした時 (1 点タッチ) と (2) 両手の親指でタッチした時 (2 点タッチ) の静電容量の正規化値の値の変化を図 4.17 に示す．グラフは Arduino のシリアルプロッタを用いた．

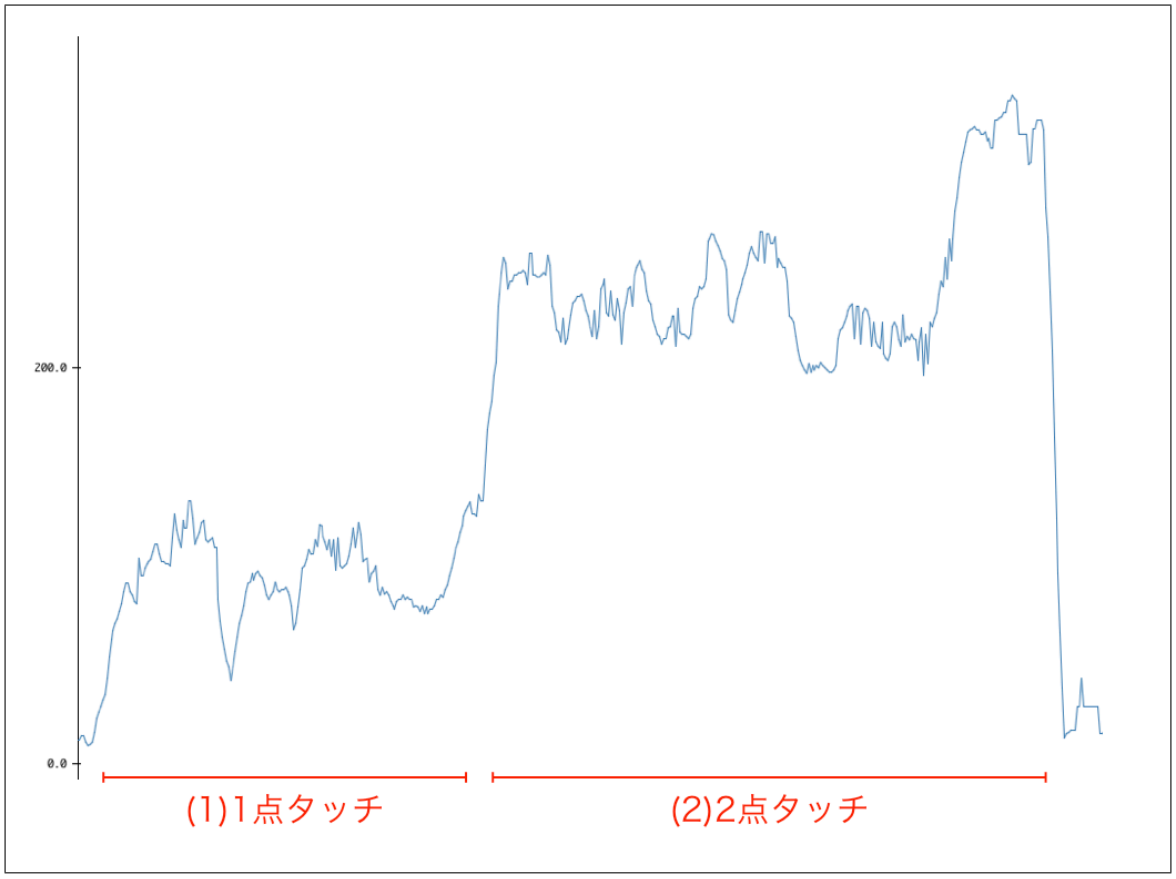

図 4.17: 1 点タッチと 2 点タッチの静電容量の正規化値の変化

図 4.17 の値の変化から，1 点タッチと 2 点タッチの正規化の和の閾値を 180 に設定した．タッチなしで 0，1 点タッチで 1，2 点タッチで 2 を出力するプログラムを作成し，シリアルプロッタ確認を行った．結果を図 4.18 に示す．

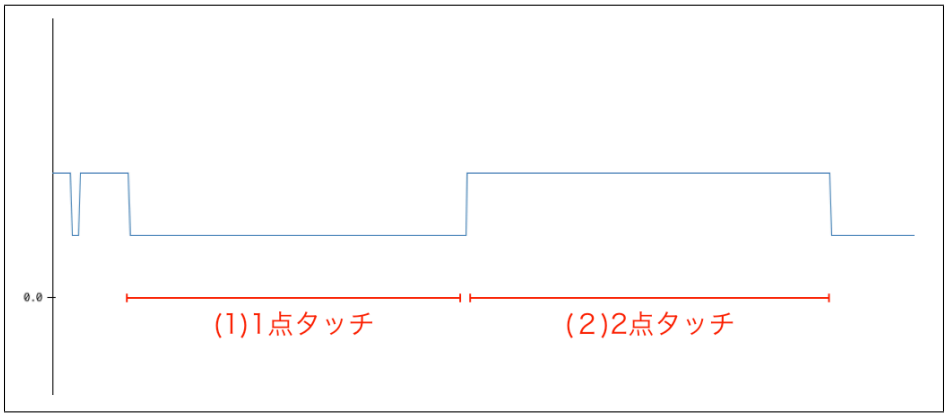

図 4.18: 1 点タッチと 2 点タッチの判定のグラフ出力

グラフのように，1 点タッチと 2 点タッチの判定が行われていることが確認できた．このように閾値による判定を行うことで，3 本指でのタッチ検出も可能となる．本手法で 2 点タッチによる入力は扱わないが，両手の親指による 2 点タッチをトリガに用いるといった操作などに用いることで，より多くのインタラクションが可能となる．

第5章　ソフトウェア実装

作成したセンサの動作と，アプリケーション内の GUI メニューの選択を紐付けるソフトウェアの実装を行った．作成したソフトウェアを起動させることによって，様々なアプリケーションの GUI メニューを本手法によって扱うことが可能となる．本章では実装に用いた装置，および実装に用いたプログラミング言語である AppleScript について紹介し，その後キーの割り当て，センサからの値を用いたメニューコマンド選択の制御および実装について述べる．

5.1　実装に用いた装置，ライブラリ

実装にあたり，OS を MacOS Sierra とする Macbook Air 13 インチを用いた．実装のプログラミング言語は AppleScript，シリアル通信のライブラリとして SerialPort X[1] を用いた．

5.2　AppleScript

AppleScript は，Apple Inc. が開発した MacOS 用のスクリプト言語である．英文のように記述できることが特徴であり，MacOS のシステムやアプリケーションを制御することが可能である．拡張するライブラリは，Library フォルダの ScriptingAdditions というフォルダに対応する osax ファイルを置くことで使用することができる．

スクリプト内でアプリケーションを呼び出すには，tell application 宣言をした後にアプリケーション名を記述する．この宣言により，Finder や Google Chrome など，様々なアプリケーションを呼び出すことができる．また，アプリケーション内の GUI メニューを制御するには，System Events というアプリケーションを呼び出す必要がある．

さらに，ハンドラという機能があり，これは AppleScript に予め備わっている関数を示す．このハンドラの中に，処理を書くことで，ソフトウェアが終了する時などの特定のタイミングでの制御を行うことが可能となる．今回使用したハンドラは以下の 4 つである．

run ハンドラ　処理が実行された時に呼び出される関数

idle ハンドラ　ソフトウェアが起動されている間，設定した時間ごとに呼び出される関数

print ハンドラ　ソフトウェア間通信で値を渡すための関数

quit ハンドラ　ソフトウェアが終了する時に呼び出される関数

[1] http://mac.softpedia.com/get/Communications/SerialPort-X.shtml

また，本手法では，ScriptingAdditions に SerialPort X.osax ファイルを置くことでシリアル通信のライブラリを使用し，AppleScript によってスクリプトエディタの GUI メニューの制御を行うソフトウェアの実装を行った．

5.3 キーの割り当て

入力は，3 章で述べたように，2 つの選択方式を使用する．HOLDKEY とし，Control キーを採用した．キャンセル操作は，Esc キーを押すことで行う．項目の選択には，Control キーを押した状態でスペースキー上で親指のスライドを行う．最後に，Control キーを離すことで項目が決定される．階層化されたメニューの遷移は，子階層のメニューをもつアイテム上で再度 Control キーを押し直すことで可能となる．

5.4 センサからの値を用いたメニューコマンド選択の制御および実装

センサから読み取った値を用いたメニューコマンド選択の実装を行った．シリアル通信で値を読み取る処理と，GUI メニューを扱う処理の制御および実装について述べる．

5.4.1 逐次処理での制御

同一プログラム内で，HOLDKEY で GUI メニューを開き，センサから値を読み取り，項目の選択を行う処理を記述した．しかし，GUI メニューを開いた状態で，シリアル通信の処理を行うことが出来ないことを動作によって確認した．なお，センサから値を読み取る部分で処理が停止されていた．そのため，別の実装のアプローチとして，シリアル通信での値の読み取りの処理と，GUI メニューを扱う処理を並列処理として機能させるための実装を行なった．

5.4.2 並列処理での制御

シリアル通信での値の読み取り処理と GUI メニューを扱う処理を別アプリケーションとして，同時に動かした状態で，ソフトウェア間でセンサから読み取った値を通信することによって並列処理を行った．AppleScript に備わっている print ハンドラによって通信を行うことで，シリアル通信のソフトウェアから GUI メニューを扱うソフトウェアにセンサからの値を渡した．並列処理での実装により，GUI メニューを開いた状態でセンサの値を読み取り，その値によって項目の移動を行うことが可能となった．

5.4.3 ハンドラの実装と処理の流れ

2 つのソフトウェアの各ハンドラに実装した処理を以下に示す.

- シリアル通信側ソフトウェア

  run ハンドラ シリアル通信のためのポートを開く

  idle ハンドラ 10 ミリ秒毎にセンサから値を読み取る

  print ハンドラ 読み取った値を返す

  quit ハンドラ ポートを閉じ，ソフトウェアを終了する

- GUI メニュー遷移側のソフトウェア

  run ハンドラ 10ms 毎のループで，HOLD キーの検出，print からの値の読み取り，メニューの遷移を行う

  quit ハンドラ ソフトウェアを終了する

第6章　評価実験: 既存手法との比較実験

本手法の評価を行うため，既存手法との比較実験を行なった．本章では，被験者，実験内容，実験アプリケーション，実験結果，議論，および実験結果のまとめについて述べる．

6.1　被験者

実験は，20歳から23歳の男女6名に対して行った．そのうち6名全員が普段からMicrosoft Wordやテキストエディタを使って文書を作成したり，プログラミングを行っていた．1名は普段からショートカットキーを使用しており，3名は少しだけ使用しており，2名はあまり使用していなかった．

6.2　実験内容

被験者に実験用アプリケーションを用い，既存手法，本手法によるGUIメニューコマンドの入力を行ってもらう．既存手法として，(1) マウスによる入力，(2) ショートカットキーによる入力，本手法として，(3) 選択方式1，(4) 選択方式2による入力の計4つの手法で入力を行う．それぞれの入力に対し，異なる10個のコマンドを入力するタスクを3回行う．タスク数は，4手法× 3タスク=12タスクであり，コマンド入力数は12タスク× 10回=120回である．コマンド入力の際には，アプリケーション側で入力すべきコマンドが指示される．この時に指示されるコマンドは,「メニュー ＞ コマンド」という形で指定される．例えば「ファイル」メニューの「ページ設定」を指定される場合は,「ファイル ＞ ページ設定」のように表示される．各コマンドを入力した際，入力したコマンドの正解，不正解に関わらず次のコマンドが指定される．ショートカットキーの入力の際に，コマンドが対応するキーがわからない場合は，command + ENTERを押すことで次のコマンドが指定される．

まず被験者に，GUIメニューの使用に慣れ，ショートカットキーを把握するための，アプリケーションを使った練習タスクを10分間行ってもらう．その後，本番のタスクをマウスによる入力から開始する．また，練習タスクとは別に，手法3のタスクの前，手法4のタスクの前には，それらの手法の練習を行ってもらう．

実験後に被験者に，本手法についてのアンケートへの回答を行なってもらった．

6.3 実験用アプリケーション

実験用アプリケーション画面は図 6.1 のようになる．アプリケーションの GUI メニューは MacOS 用アプリケーションである，スクリプトエディタと同じ構成に設定した．コマンドとショートカットキーの対応も本アプリケーションとスクリプトエディタで同様である．被験者は (1) の GUI メニューからコマンドを選択する．(2) の被験者名の入力とタスク，手法の選択は実験者が行い，(3) の start ボタンを被験者がクリックすると最初のコマンドが指定され，タスクが開始する．コマンドは (4) に示される部分に指定される．また，アプリケーション側では，各タスク内でそれぞれのコマンドの入力にかかった時間と，指定コマンド，実際の入力コマンドを csv ファイルに記録する．入力時間，正解率は記録された csv ファイルを用いて分析を行う．

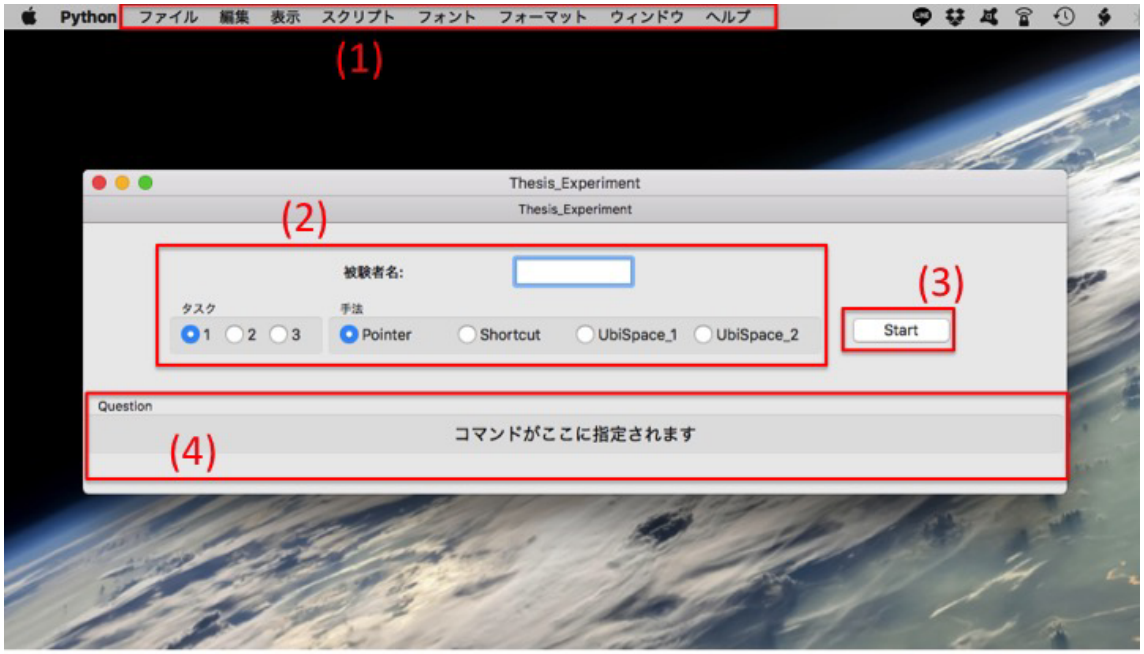

図 6.1: 実験用アプリケーション画面

6.4　実験結果

6.4.1　入力時間

各手法によるコマンド入力の平均時間を，図 6.2 および表 6.1 に示す．図のエラーバーと表の括弧内は標準偏差を示す．ここで用いる結果は，タスク毎の全被験者のコマンド入力時間の平均，標準偏差である．ショートカットキーによる入力時間が最も短く，提案手法の選択方式 1 による入力時間が最も長い結果となった．選択方式 1 では，中央のメニューにあるコマンドへの移動時間を多く必要とするため，中央のメニューにあるコマンドと，端のメニューにあるコマンドで入力時間が大きく異なっていた．また，選択方式 2 は，既存手法であるマウスの入力とショートカットキーの入力より入力時間が長く，選択方式 1 より入力時間が短い結果となった．最初にメニューを選ぶため，最左または最右のメニュー内のコマンドを選択する時と，中央のメニュー内のコマンドを選択する時の入力時間の差は少なかった．被験者によっての選択時間の個人差が大きく，被験者 E は「ファイル」メニューの「閉じる」を 3.7 秒で選択できたが，被験者 C は 11.5 秒かかった．同様に，被験者 B は「表示」メニューの「タブバーを表示」を 3.1 秒で選択できたが，被験者 C は 11.4 秒かかった．

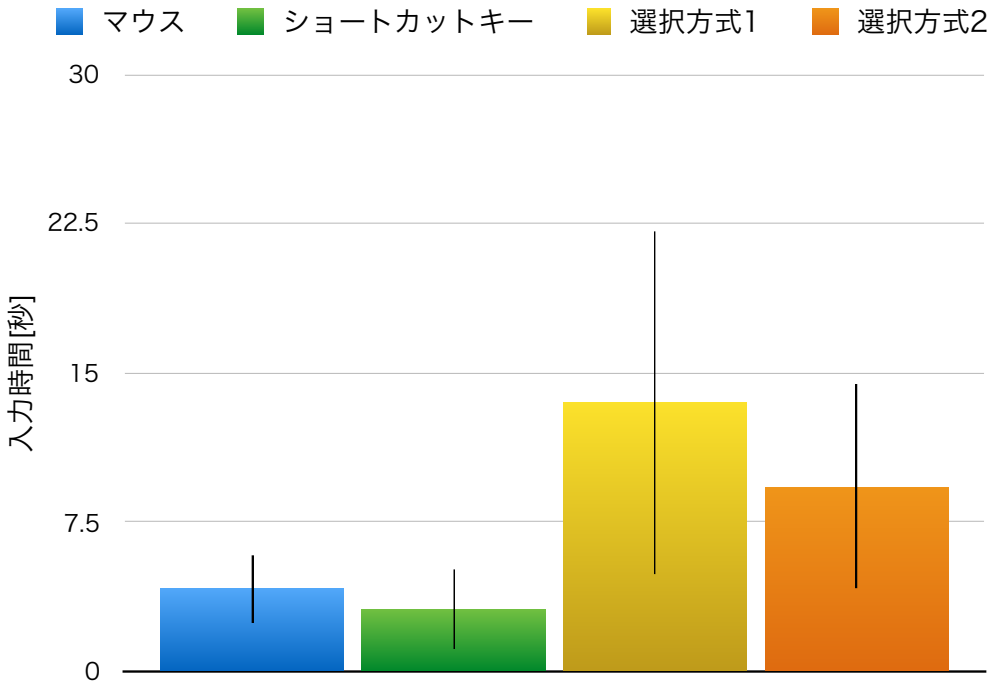

図 6.2: 各手法による入力の平均時間

表 6.1: 各手法による入力の平均時間

タスク・手法	入力時間 [秒] (標準偏差)
1・マウス	3.97 (1.27)
2・マウス	3.60 (1.40)
3・マウス	4.96 (2.50)
1・ショートカットキー	3.67 (2.74)
2・ショートカットキー	2.50 (1.07)
3・ショートカットキー	3.30 (1.70)
1・選択方式 1	13.68 (9.27)
2・選択方式 1	11.94 (7.50)
3・選択方式 1	14.94 (8.66)
1・選択方式 2	9.17 (4.51)
2・選択方式 2	7.54 (5.44)
3・選択方式 2	11.03 (4.73)

6.4.2 正解率

各手法によるコマンド入力の正解率を，図 6.3 および表 6.2 に示す．図のエラーバーと表の括弧内は標準偏差を表す．ここで用いる結果は，各被験者に対するそれぞれのタスクの正解率の平均，標準偏差である．マウスによる入力の正解率が最も高く，ショートカットキーによる入力の正解率が最も低かった．選択方式 1 はどのタスクにおいても，選択方式 1 とショートカットキーの入力による正解率よりも高い数値を示した．選択方式 2 はタスク 2 を除いて，ショートカットキーの入力による正解率よりも高い数値を示した．選択方式 1 のタスク 3 では，全被験者のコマンド入力で，エラーは 1 度のみだった．

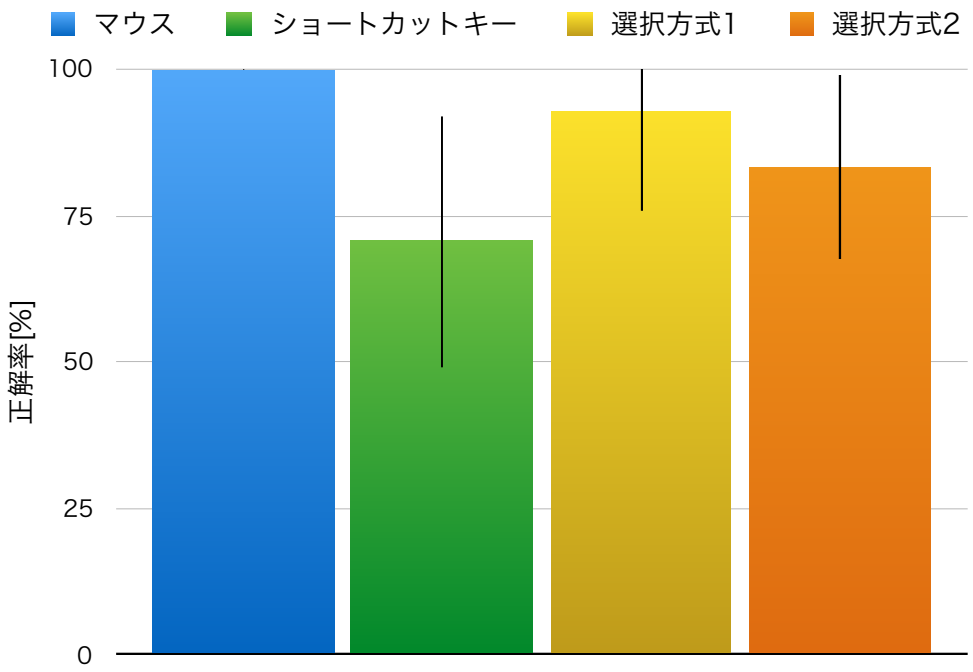

図 6.3: 各手法による入力の正解率

表 6.2: 各手法による入力の正解率

タスク	正解率 [%] (標準偏差)
1・マウス	100.00 (0.0)
2・マウス	100.00 (0.0)
3・マウス	100.00 (0.0)
1・ショートカットキー	71.66 (16.74)
2・ショートカットキー	78.33 (10.67)
3・ショートカットキー	61.66 (29.10)
1・選択方式 1	88.33 (26.08)
2・選択方式 1	91.66 (10.67)
3・選択方式 1	98.33 (3.72)
1・選択方式 2	86.66 (15.98)
2・選択方式 2	76.66 (16.99)
3・選択方式 2	86.66 (11.05)

6.4.3 主観的評価

実験後に被験者にアンケートをとった．アンケート用紙については付録 A.3，その結果については，付録 B.3 を参照されたい．

選択方式 1 に関して，4 名の被験者が誰でもすぐに使えると回答した．1 名の被験者が本手法を使いたいと回答した．3 名の被験者が本手法に煩わしさを感じると回答した．煩わしさを感じる理由として，被験者 A「中央にある項目が選びにくい」，被験者 C「1 コマだけ動かすことが難しい」，被験者 E「コマンドが遠い時に辿っていくのが面倒だった」などの回答があった．その他の意見で，被験者 C「ファイルメニューからヘルプメニューに遷移できるのが良いと思った」，被験者 D「マウスと同様に素早く動かせるようになると嬉しい」などの回答があった．

一方，選択方式 2 に関して，5 名の被験者が誰にでもすぐに使えると回答した．4 名の被験者が本手法を使いたいと回答した．1 名の被験者が本手法に煩わしさを感じると回答した．煩わしさを感じる理由として，被験者 C「選択したいメニューを間違えてしまった時，メニューの選択に戻れないこと」という回答があった．その他の意見で，被験者 A「遠くの項目に行く時に便利だった」，被験者 C「コマンドの移動が縦なので，横よりは縦の親指の移動の方がイメージを掴みやすいと思った」，被験者 D「精度が良くなれば使いやすいと思った」などの回答があった．

また，スペースキー上での親指のスライド操作を使用したい他の場面も尋ねたところ，被験者 A「Excel のセル移動」や，被験者 B「文章の範囲選択，ブラウザのタブ移動」，被験者 C「画像の拡大縮小，スクリーンの移動」，被験者 D「文章を書いている際のカーソル移動」という回答があった．

6.5 議論

本手法の選択方式 1，選択方式 2 の入力時間が長かった原因として，移動平均フィルタによる遅延時間の問題がある．親指を早くスライドさせて親指をスペースキーから離す，フリック操作のように親指を動かす場合，平滑化された指のタッチ位置が正確なタッチ位置と認識される前に指を離してしまうため，メニューの選択カーソルが殆ど動かなかった．それに対して，なぞるようにゆっくり指を動かす方がカーソルが早く動くため，被験者が考えるカーソルの動きと違う動きをしていたと考えられる．被験者によって入力時間に個人差があり，早くカーソルを動かそうと，親指のフリック操作を多くしていた被験者の場合，入力時間が長くかかっていたと考えられる．従って，マウスやタッチパッドのように，遅延時間の生じない入力を可能とすることが一つの課題である．そのために，銅箔テープの枚数を増やし，平滑化のフィルタ係数を減らすことや，平滑化に別のフィルタを用いることがアプローチとして挙げられる．

正解率が 100%ではなく，エラー率が生じていた原因としてノイズによる問題がある．全体のエラーの中で，指示されたコマンドに対し，被験者はその前後隣のコマンドの選択を多く行なっていたため，ノイズによる選択カーソルのずれによって生じたエラーが殆どであったと考えられる．アンケートで「1 コマの移動が難しい」という回答があったことからも，ノイズによるカーソルのずれが生じていたことがわかる．エラーを防ぐためにも，移動平均フィルタよりも強力なフィルタを用いてノイズをより軽減させ，精度を上げる必要がある．

アンケートより,「エクセルのセル移動」や「ブラウザのタブ移動」などの入力にスペースキー上での親指のスライド操作を使いたいとの回答があったことから，テキスト入力のアプリケーションだけでなく，他の場面でも使用することで，通常の操作を支援することができる場面が考えられる．エクセルのセル移動は，親指をスライドさせた分だけセル移動をさせることで，数値の入力後にポインティングデバイスとの持ち替え時間を必要とせず，矢印キーを複数回押さずにセルの移動を行えるような操作が考えられる．ブラウザのタブ移動では，タブは画面の上部にあるため，カーソルを画面上部まで動かさなければならない．この場合，ポインティングデバイスを使用していない方の手で，親指のスライド操作を行うことにより，カーソル移動時間をかけずにタブ移動を行う操作が挙げられる．他にも，3D モデリングソフトで，親指のスライド操作を図形の回転，Control キーを押しながらの親指のスライド操作で図形の拡大縮小のような操作に割り当てることで，ポインティングデバイスと組み合わせたインタラクションが可能となる．

6.6 実験結果のまとめ

実験より，本手法の選択方式 1，選択方式 2 による入力は，既存の手法よりも長い入力時間を必要とするが，正解率はタスク 2 を除いてショートカットキーによる入力より高い結果となった．また，選択方式 2 の方が入力時間が短くなり，選択方式 1 の方が正解率が高くなった．

第7章　まとめと今後の課題

本論文において，持ち替え時間を必要とせず，ホームポジションでの入力を可能とする，スペースキー上での親指のスライド操作を用いたコマンド入力手法を提案した．銅箔テープを用いて作成したタッチセンサによってスペースキーを拡張し，それらの予備実験を行い，スペースキー上でのタッチおよびスライド操作を検出するシステムを作成した．さらに，アプリケーションの GUI メニューの選択に本手法を使えるよう，メニューを動かすアプリケーションを作成した．その後，既存手法との比較実験を行なった．実験の結果から，本手法の選択方式 1，選択方式 2 による入力は，既存手法よりも長い入力時間を必要であり，遅延時間の問題が原因の一つであることが分かった．一方，正解率は選択方式 1 は全てのタスクにおいてショートカットキーによる入力より高い結果となり，選択方式 2 はタスク 2 を除いてショートカットキーによる入力より高い結果となった．しかし，ノイズによるエラーが生じていたことが分かった．これらの問題を解決するために，別のフィルタを用いてタッチ位置のノイズの除去を考えている．遅延時間を生じさせずに，ノイズの除去を行うことができれば，ユーザがタッチした時にリアルタイムで操作が反映され，よりシステムは堅牢になる．

また，テキスト入力を主とするアプリケーションだけでなく，ポインティングデバイスを使うようなアプリケーションでも本手法を使うことを考えている．例えば片手でポインティングデバイスを使用し，もう一方の手で本手法による入力を行うことで，より多くのインタラクションが可能となる．

今後はより高速かつ正確にメニュー選択を行えるよう，実験の結果から考察された改善点を踏まえた実装を行い，本手法のさらなる評価実験を行う．

謝辞

本論文を執筆するにあたり，指導教員である高橋先生，志築先生，嵯峨先生には多大なご助力を賜り，感謝いたします．特に高橋先生には，論文の執筆方法だけでなく，研究生活においても丁寧いかつ熱心なご指導を賜りました．深く感謝いたします．

田中研究室の皆様には，研究に置ける様々なご支援をいただきました．特に UBIQUITOUS チームの皆様には，チームゼミをはじめ，多くのご支援をいただきました．また，嵯峨チームの富田洋文さんには，システムの実装から実験まで様々なアドバイスをいただきました．深く感謝いたします．さらに，実験の被験者としてご協力頂いた皆様に，感謝申し上げます．

最後に，私の生活を支えてくださった家族と友人，大学生活においてお世話になった全ての方々に心より感謝いたします．

参考文献

[1] David M. Lane, H. Albert Napier, S. Camille Peres, and Aniko Sandor. Hidden costs of graphical user interfaces: Failure to make the transition from menus and icon toolbars to keyboard shortcuts. *International Journal of Human-Computer Interaction*, Vol. 18, No. 2, pp. 133–144, 2005.

[2] Paul H. Dietz, Benjamin Eidelson, Jonathan Westhues, and Steven Bathiche. A practical pressure sensitive computer keyboard. In *Proceedings of the 22Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '09, pp. 55–58, New York, NY, USA, 2009. ACM.

[3] Jun Kato, Daisuke Sakamoto, and Takeo Igarashi. Surfboard: Keyboard with microphone as a low-cost interactive surface. In *Adjunct Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, pp. 387–388, New York, NY, USA, 2010. ACM.

[4] Ying-Chao Tung, Ta Yang Cheng, Neng-Hao Yu, Chiuan Wang, and Mike Y. Chen. Flickboard: Enabling trackpad interaction with automatic mode switching on a capacitive-sensing keyboard. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, pp. 1847–1850, New York, NY, USA, 2015. ACM.

[5] Stuart Taylor, Cem Keskin, Otmar Hilliges, Shahram Izadi, and John Helmes. Type-hover-swipe in 96 bytes: A motion sensing mechanical keyboard. In *Proceedings of the 32Nd Annual ACM Conference on Human Factors in Computing Systems*, CHI '14, pp. 1695–1704, New York, NY, USA, 2014. ACM.

[6] Robert J.K. Jacob, Audrey Girouard, Leanne M. Hirshfield, Michael S. Horn, Orit Shaer, Erin Treacy Solovey, and Jamie Zigelbaum. Reality-based interaction: A framework for post-wimp interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pp. 201–210, New York, NY, USA, 2008. ACM.

[7] Donald E Knuth and Andrew Binstock. Interview with donald knuth. *InformIT. Retrieved*, pp. 04–01, 2010.

[8] John Karat, James E McDonald, and Matt Anderson. A comparison of menu selection techniques: touch panel, mouse and keyboard. *International Journal of Man-Machine Studies*, Vol. 25, No. 1, pp. 73–88, 1986.

[9] Sylvain Malacria, Gilles Bailly, Joel Harrison, Andy Cockburn, and Carl Gutwin. Promoting hotkey use through rehearsal with exposehk. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pp. 573–582, New York, NY, USA, 2013. ACM.

[10] Jingjie Zheng and Daniel Vogel. Finger-aware shortcuts. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems*, CHI '16, pp. 4274–4285, New York, NY, USA, 2016. ACM.

[11] 西村信哉, 三浦元喜. キーボードとマウスの融合による持ち替える必要のない入力デバイス. 情報処理学会, 2011.

[12] Jun Rekimoto. Thumbsense: Automatic input mode sensing for touchpad-based interactions. In *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '03, pp. 852–853, New York, NY, USA, 2003. ACM.

[13] Masaya Tsuruta, Shuta Nakamae, and Buntarou Shizuki. Rootcap: Touch detection on multi-electrodes using single-line connected capacitive sensing. In *Proceedings of the 2016 ACM on Interactive Surfaces and Spaces*, ISS '16, pp. 23–32, New York, NY, USA, 2016. ACM.

[14] Tobias Grosse-Puppendahl, Andreas Braun, Felix Kamieth, and Arjan Kuijper. Swiss-cheese extended: An object recognition method for ubiquitous interfaces based on capacitive proximity sensing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, pp. 1401–1410, New York, NY, USA, 2013. ACM.

[15] Hirobumi Tomita, Simona Vasilache, and Jiro Tanaka. Using the office desk as a touch interface. *Human Interface and the Management of Information: Information, Design and Interaction*, pp. 585–595, 2016.

付 録A　実験に用いた書類

第 6 章における実験の際に使用した実験同意書，実験手順書，およびアンケート[1]を以下に示す.

[1]Google フォームを使用

A.1　実験同意書

実験対象者へ

年　　月　　日

GUIメニューコマンドの入力を行う実験

実験の目的

　この実験は，PCを使ってのGUIメニューコマンドの入力に関しての比較を行うことで，本研究で提案する手法と既存の手法のパフォーマンスの違いを結果として得ることを目的とします．

実験内容

　PCを使い，画面にて指示されるGUIメニューコマンドを選択するタスクを行っていただきます．実験に関する詳細は本同意書記入後に説明させていただきます．実験後に，簡単なアンケートをご記入いただきます．所要時間は，前後の実験説明も含めて40分程度です．

個人情報とデータの取り扱い

　本実験によって得られたデータは，個人が特定できないように処理いたします．また，これらのデータを学内研究会，専門学会等を通じて発表する場合は，番号付を行うとともに匿名化いたしますので，個人情報は守秘されます．データの保管には万全を期し外部へは漏洩いたしません．

実験対象者の権利について

　実験への参加は，協力者の自由意志によるものであり，実験への参加を随時拒否・撤回することができます．また，それによる不利益はありません．その場合，それまでに得られたデータを破棄し，それ以降の研究には一切使用いたしません．但し，取り消し要求された時点で公表済みの解析結果がある場合は，データを破棄できませんのでご承知おきください．

以上，何かご不明な点がありましたら遠慮なくお尋ねください．
本研究へのご理解とご協力に深く感謝いたします．

(実験者)
筑波大学　情報学群　情報科学類
4年　村田　主磨

参加同意書

メニューコマンドの入力に関する実験

<< 説明を受けた項目 >>

□　研究の目的
□　実験内容
□　個人情報とデータの取り扱い
□　実験対象者の権利について

説明日時　　年　月　日

説明者：＿＿＿＿＿＿＿＿＿＿＿＿

私は，以上の説明を理解し，本実験に参加することに同意します.

年　月　日

所　属：＿＿＿＿＿＿＿＿＿＿＿＿

氏　名：＿＿＿＿＿＿＿＿＿＿＿＿

A.2　実験説明書

実験説明

- メニューコマンドの入力に関する実験 -

概要

- 本実験では，指定された手法でGUIメニューコマンドの選択，及び入力を行っていただきます．画面上にて指定されたコマンドをGUIメニュー上から選択していただきます．
- 実験では入力を何回かのセットに分けて行います．このセットを以後タスクと呼びます．1タスクあたり，10個のコマンドの入力を行っていただき，4つの手法に対して，それぞれ3タスク入力を行っていただきます．3タスク（1手法）ごとに，約2〜3分の休憩時間を設けます．
- また，本番前にアプリケーションを触る時間を時間を10分間設けます．この間に，メニュー項目の把握と，ショートカットキーの記憶をお願いいたします．
- 実験で用いるアプリケーションのGUIメニュー及びショートカットキーの割り当ては，MacOS用アプリケーションである，「スクリプトエディタ」と同じ構成になっています．

画面説明

画面上部にGUIメニューがあり，アプリケーション画面下部にコマンドが指定されます．
コマンドは「ファイル > 保存」といったような，道順を全て表す指定のされ方となっています．
被験者名，タスク，手法等の入力は実験者が行います．

手法1　マウスによる入力

手法1は，マウスによる入力です．マウスカーソルを使い，指定されたコマンドをクリックしていただきます．決定には，左クリックを用います．

手法2　ショートカットキーによる入力

手法2は，ショートカットキーによる入力です．指定されたコマンドに対応するショートカットキーの入力によってコマンドを選択していただきます．以下にショートカットキーに対する説明を記載します．

- 各コマンドに対するショートカットキーは，GUIメニュー上のコマンド名の右に表示されています．
- タスクでは対応するショートカットキーがないコマンドの指定は行われません．つまり，指定されるコマンドはショートカットキーを用いて入力できるコマンドのみです．
- 「⌘」はcommandキー，「⇧」はshiftキー，「⌥」はoptionキーに対応しています．
  （入力例）⌥⌘R　→　option + command + Rキー
- 対応するキーがわからない場合は，「command + enter(return)」を押していただくと，次のコマンド入力へと進みます．

手法3　提案手法による入力(1)

　手法3は，本研究の提案手法による入力です．スペースキー上での親指のスライド操作を用いてコマンドの入力を行っていただきます．本番前に練習時間を設けます．

- ctrlキーを押すと，「ファイル」メニューが開き，その最も上の項目が選択状態となります．
- ctrlキーを押した状態で，親指を右にスライドさせると選択項目が下に，左にスライドさせると選択項目が上に動きます．
- ctrlキーを離すと，選択状態の項目が決定されます．
- 選択項目にサブメニューが存在する場合は，再度ctrlキーを押し直すとサブメニューに移ります．
- メニュー上での選択項目の移動は，メニューコマンド項目を1次元の列とみなした並びの移動になります．

手法4　提案手法による入力(2)

　手法4も，本研究の提案手法による入力です．手法3と同様に，親指のスライド操作を用いてコマンド入力を行いますが，幾つか異なる点があります．本番前に練習時間を設けます．

- ctrlキーを押すと，「ファイル」メニューが開きます．この状態で，親指を右にスライドさせると，右隣のメニューが開きます．左も同様です．

- メニューが開いた状態でctrlキーを押し直すと，そのメニューの最も上の項目が選択されます．

- その他の操作は手法3と同様です．

手法3と4を行う上でのお願い

　手法3と4による操作の際，キャリブレーションを行っていただく必要があります．

- キャリブレーションの流れは，マイコン上にあるタクトスイッチを一度押し，スペースキー上で親指を左から右，右から左へと1往復スライドさせます．

- キャリブレーションは，それぞれのタスクの前に必ず行ってください．(実験者が指示します)

- スペースキー上での親指スライドを行う際，紙の部分以外にはできるだけ触れないようにしてください．

A.3　アンケート

A.3.1　選択方式 1

評価実験 選択手法1

*必須

1. **年齢** *

2. **性別** *

1 つだけマークしてください。

- ◯ 男
- ◯ 女

3. **本手法は使いやすいと思ったか** *

1 つだけマークしてください。

- ◯ 1.使いやすい
- ◯ 2.どちらかと言えば使いやすい
- ◯ 3.どちらとも言えない
- ◯ 4.どちらかと言えば使いにくい
- ◯ 5.使いにくい

4. **本手法は誰でもすぐに使えると思った** *

1 つだけマークしてください。

- ◯ 1.すぐに使える
- ◯ 2.どちらかと言えばすぐに使える
- ◯ 3.どちらとも言えない
- ◯ 4.どちらかと言えばすぐには使えない
- ◯ 5.すぐに使えない（慣れるまでに時間がかかる）

5. **本手法を使いたいと思った** *

1 つだけマークしてください。

- ◯ 1.使いたい
- ◯ 2.どちらかと言えば使いたい
- ◯ 3.どちらとも言えない
- ◯ 4.どちらかと言えば使いたくない
- ◯ 5.使いたくない

6. **本手法の操作に煩わしさを感じたか** *

1 つだけマークしてください。

- 1.感じなかった
- 2.どちらかと言えば感じなかった
- 3.どちらとも言えない
- 4.どちらかと言えば感じた
- 5.感じた

7. **どのような操作に煩わしさを感じたか(※煩わしさを感じた．またはどちらかと言えば感じたにチェックをつけた方のみ)**

---

8. **本手法を使うことで素早いメニュー選択を行えたと思ったか** *

1 つだけマークしてください。

- 1.行えた
- 2.どちらかと言えば行えた
- 3.どちらとも言えない
- 4.どちらかと言えば行えなかった
- 5.行えなかった

9. **他，本手法に対しての感想，意見**

A.3.2 選択方式 2

評価実験 選択手法2

*必須

1. **年齢** *

2. **性別** *

1 つだけマークしてください。

- 男
- 女

3. **本手法は使いやすいと思ったか** *

1 つだけマークしてください。

- 1.使いやすい
- 2.どちらかと言えば使いやすい
- 3.どちらとも言えない
- 4.どちらかと言えば使いにくい
- 5.使いにくい

4. **本手法は誰でもすぐに使えると思った** *

1 つだけマークしてください。

- 1.すぐに使える
- 2.どちらかと言えばすぐに使える
- 3.どちらとも言えない
- 4.どちらかと言えばすぐには使えない
- 5.すぐに使えない（慣れるまでに時間がかかる）

5. **本手法を使いたいと思った** *

1 つだけマークしてください。

- 1.使いたい
- 2.どちらかと言えば使いたい
- 3.どちらとも言えない
- 4.どちらかと言えば使いたくない
- 5.使いたくない

6. **本手法の操作に煩わしさを感じたか** *

1 つだけマークしてください。

- 1.感じなかった
- 2.どちらかと言えば感じなかった
- 3.どちらとも言えない
- 4.どちらかと言えば感じた
- 5.感じた

7. **どのような操作に煩わしさを感じたか(※煩わしさを感じた．またはどちらかと言えば感じたにチェックをつけた方のみ)**

---

8. **本手法を使うことで素早いメニュー選択を行えたと思ったか** *

1 つだけマークしてください。

- 1.行えた
- 2.どちらかと言えば行えた
- 3.どちらとも言えない
- 4.どちらかと言えば行えなかった
- 5.行えなかった

9. **普段Wordやテキストエディタを使って文書を作成する，もしくはプログラミングを行うかどうか** *

1 つだけマークしてください。

- 1.使う(する)
- 2.どちらかと言えば使う(する)方である
- 3.どちらとも言えない
- 4.どちらかと言えば使わない(しない)方である
- 5.使わない(しない)

10. **普段PCを使う際，ショートカットキーをよく使うか** *

1 つだけマークしてください。

- 1.使う
- 2.どちらかと言えば使う方である
- 3.どちらとも言えない
- 4.どちらかと言えば使わない方である
- 5.使わない

11. **スペースキー上でのスライド操作を使用したい他の場面**

    親指を用いたスライド操作に限定しない.

12. **他，本手法に対しての感想，意見**

付 録B　実験結果

被験者毎の入力時間，コマンド入力の正解数，アンケート結果を以下に示す.

B.1　入力時間

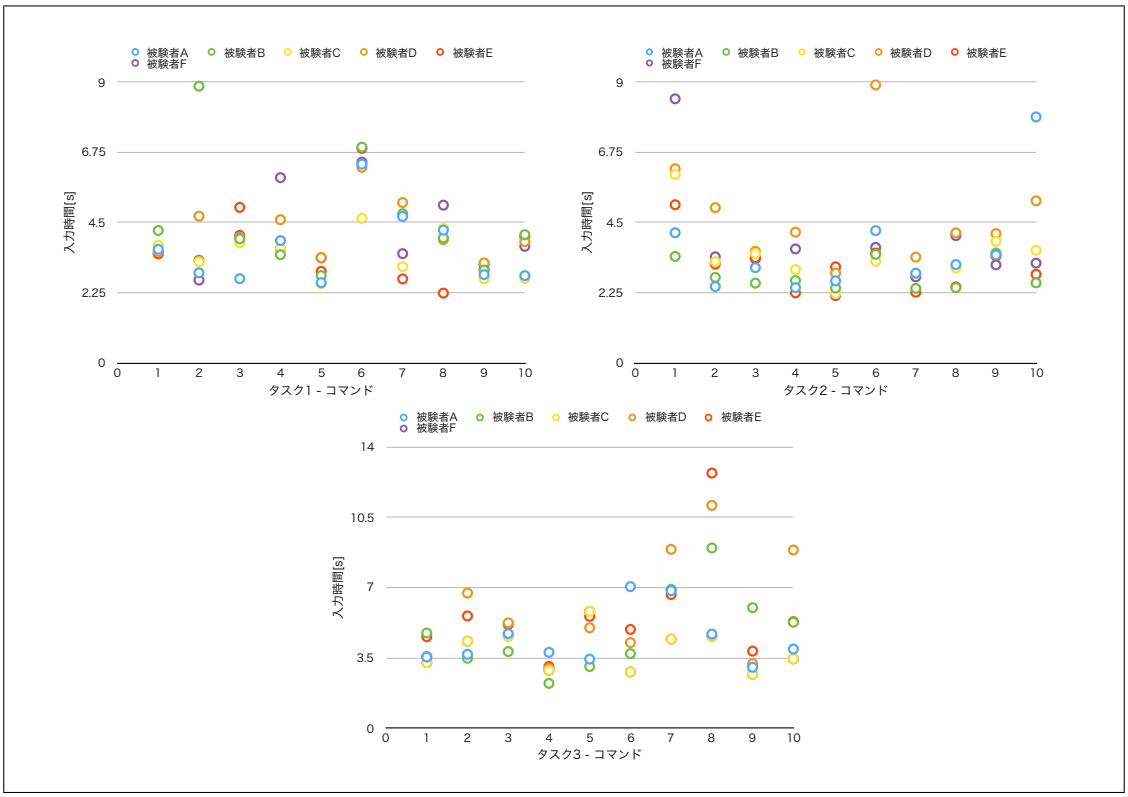

図 B.1: マウスによる入力

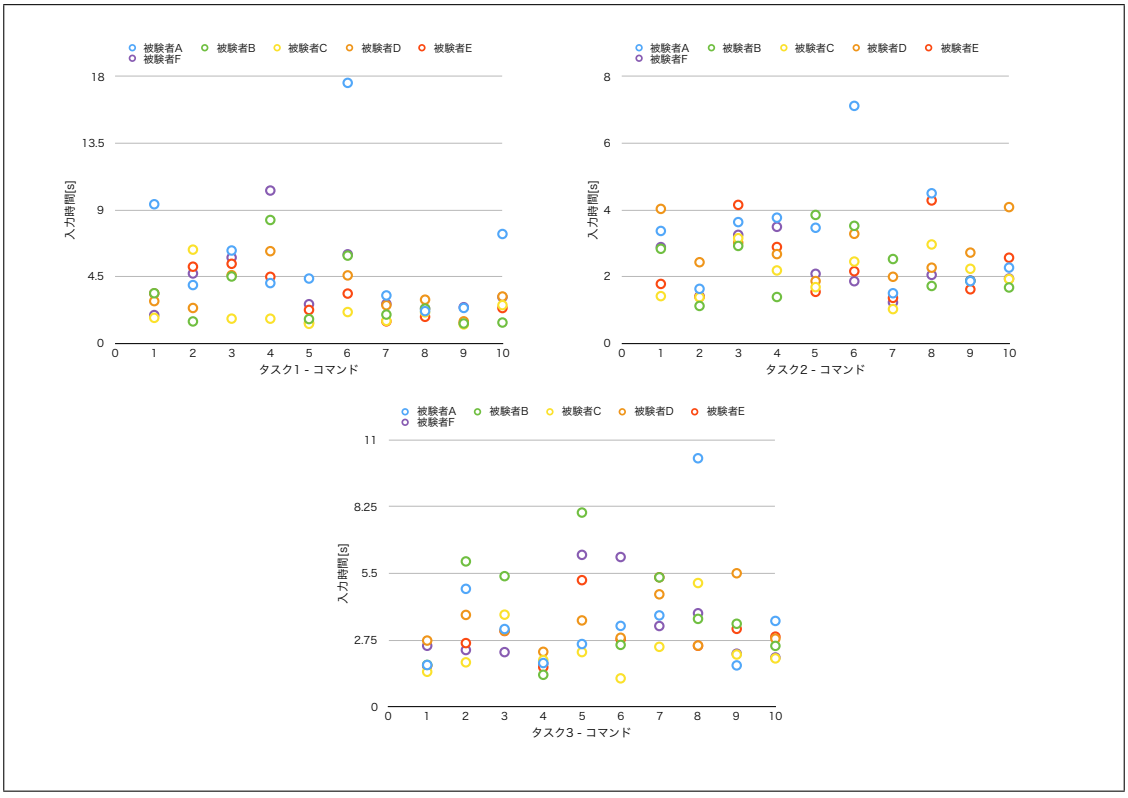

図 B.2: ショートカットキーによる入力

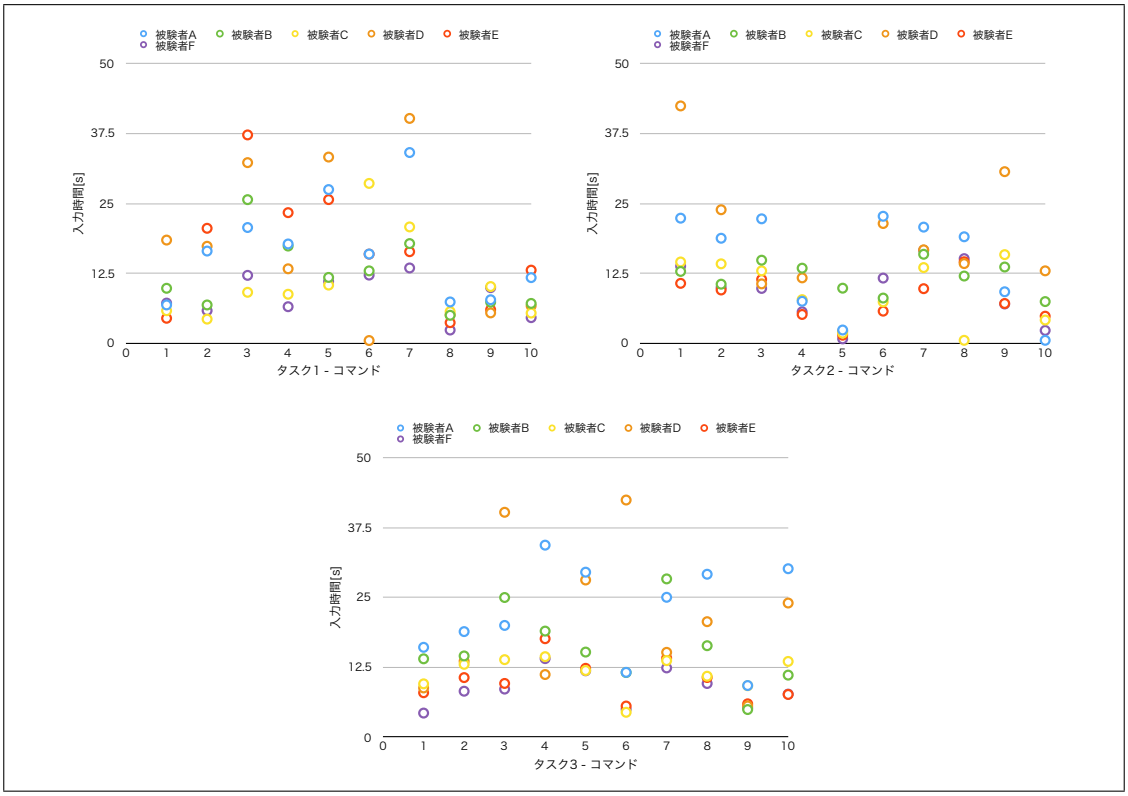

図 B.3: 選択方式 1 による入力

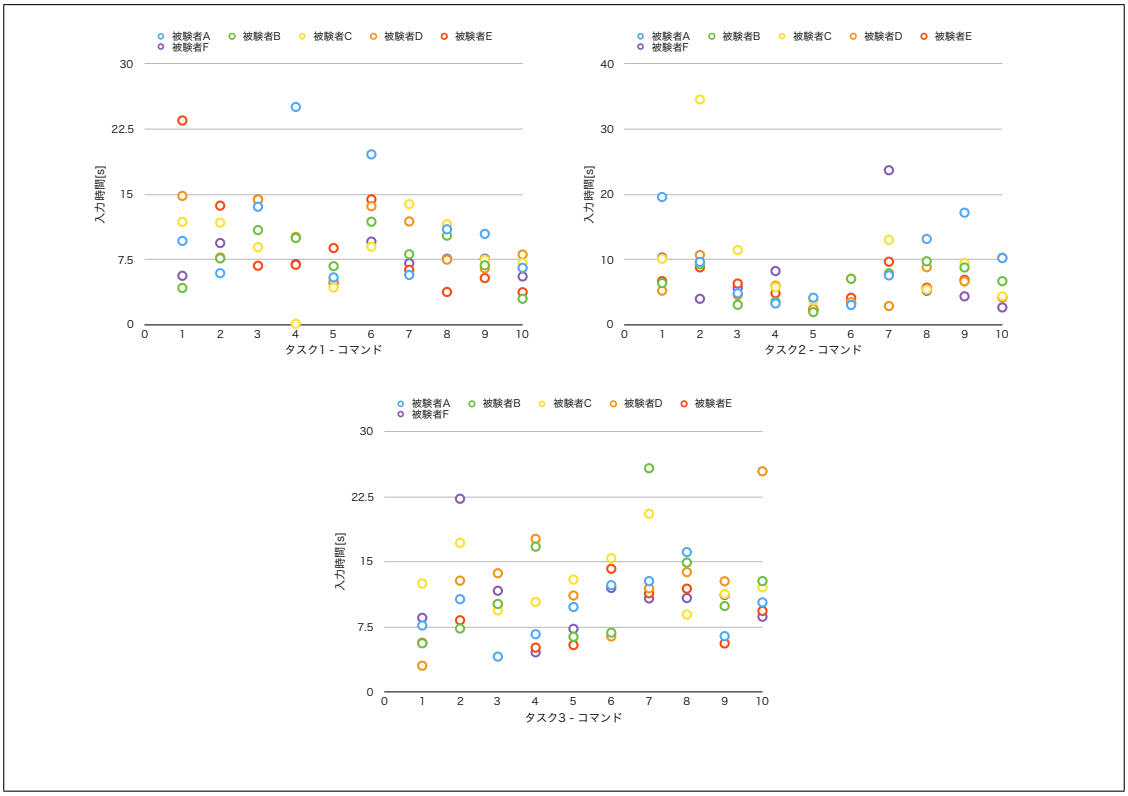

図 B.4: 選択方式 2 による入力

B.2 コマンド入力の正解数

表 B.1: マウスによる入力のコマンド入力正解数

タスク	被験者 A	被験者 B	被験者 C	被験者 D	被験者 E	被験者 F
1	10	10	10	10	10	10
2	10	10	10	10	10	10
3	10	10	10	10	10	10

表 B.2: ショートカットキーによる入力のコマンド入力正解数

タスク	被験者 A	被験者 B	被験者 C	被験者 D	被験者 E	被験者 F
1	5	7	9	5	9	8
2	7	8	8	7	7	10
3	6	7	10	3	2	9

表 B.3: 選択方式 1 による入力のコマンド入力正解数

タスク	被験者 A	被験者 B	被験者 C	被験者 D	被験者 E	被験者 F
1	10	10	10	3	10	10
2	7	10	9	10	10	9
3	10	10	9	10	10	10

表 B.4: 選択方式 2 による入力のコマンド入力正解数

タスク	被験者 A	被験者 B	被験者 C	被験者 D	被験者 E	被験者 F
1	7	10	6	9	10	10
2	5	8	6	9	10	8
3	8	10	7	8	10	9

B.3 アンケート結果

第 6 章における実験の際にとったアンケート結果を以下に示す.

B.3.1 選択方式 1

表 B.5: 年齢・性別・選択項目

質問	被験者 A	被験者 B	被験者 C	被験者 D	被験者 E	被験者 F
1	22	22	20	22	23	21
2	女	男	女	男	男	男
3	4	4	4	4	3	4
4	2	4	1	4	1	2
5	3	4	5	3	4	2
6	4	5	5	4	4	3
8	4	4	5	4	4	3

表 B.6: 記述項目

質問	被験者	回答
7	A	真ん中のほうにある項目が選びにくかったです.
	B	コントロールキーを押しっぱなしにしないといけないところ.
	C	一コマだけ動かすのが難しい.
	D	一個一個メニュー項目を辿らなければならないのが面倒に思った.
	E	コマンドが遠いときに辿っていくのが面倒だった.
9	C	ファイルからヘルプに巻き戻せるのが良いと思った，誰にでもわかりやすいとおもった.
	D	やはりマウスと同様に素早く動かすと早く移動できるようになると嬉しい.

B.3.2 選択方式 2

表 B.7: 年齢・性別・選択項目

質問	被験者 A	被験者 B	被験者 C	被験者 D	被験者 E	被験者 F
1	22	22	20	22	23	21
2	女	男	女	男	男	男
3	2	3	5	3	2	2
4	2	2	1	3	1	1
5	2	3	5	2	1	2
6	2	2	5	2	2	3
8	2	3	5	4	2	3
9	1	1	1	1	1	1
10	3	3	2	1	2	2

表 B.8: 記述項目

質問	被験者	回答
7	C	編集を選択したかったのにファイルを選択してしまった時，ファイルの中の選択に移ってしまった.
11	A	エクセルのセル移動.
	B	文章の範囲選択，ブラウザなどでのタブの移動.
	C	画像の拡大縮小，スクリーンの移動.
	D	文章を書いている際のカーソルの移動に使えれば便利そう.
12	A	一個の移動が難しかったですが，遠くの項目に行く時は便利だと思います.
	B	手法 3 のように，各タブ内で上下が繋がっていても良いと思った.
	C	キャリブレーションがうまくいけば良いと思った．iPhone などのタッチして操作するものは上下に指を動かすので横よりは縦の方がイメージを掴みやすいと思った.
	D	ctrl キーで決定するのが少し煩わしいと思ったが，精度が良くなれば使いたいと思った.
	E	手法 1 と比べてかなり使いやすかった.