Vol.2017-HCI-172 No.20
2017/3/7

Ubi-Space: スペースキー上での親指スライドを用いたメニューコマンド入力手法

村田 主磨[1,a)]　富田 洋文[2]　嵯峨 智[3]　志築 文太郎[3]　高橋 伸[3]

概要：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である Ubi-Space を提案する．GUI メニューコマンドのカーソルをスペースキー上で親指のスライド操作によって操作することで，コマンドに対応するショートカットキーを覚えるような記憶する作業を必要とせず，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を必要としない，ホームポジションでのコマンドの選択を行うことを可能とする．本研究では，銅箔テープを用いた静電容量タッチセンサによってスペースキーを拡張し，Ubi-Space によってメニューの選択を行うためのソフトウェアを実装した．また，実装したシステムを用いて，本手法の有用性の評価を行った．

1. はじめに

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が用いられる．その入力には，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，ショートカットキーによる入力方法がある．テキスト入力を行うような，キーボードを使用するアプリケーションにおいて，ポインティングデバイスの場合，コマンド選択時にキーボードからポインティングデバイスへと手を移動するための持ち替え時間が発生してしまう．また，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発生してしまう．

一方，ショートカットキーによるコマンドの入力は，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行えるという点で，多くの場合ポインティングデバイスでの入力よりも短い時間でのインタラクションを行うことが可能である [1]．しかし，コマンドの数が多いことや，複数の修飾キーを使うことからその記憶が困難である [2], [3]．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の研究 [4] や，ショートカットキーの利用を支援する手法の研究がされている [3]．

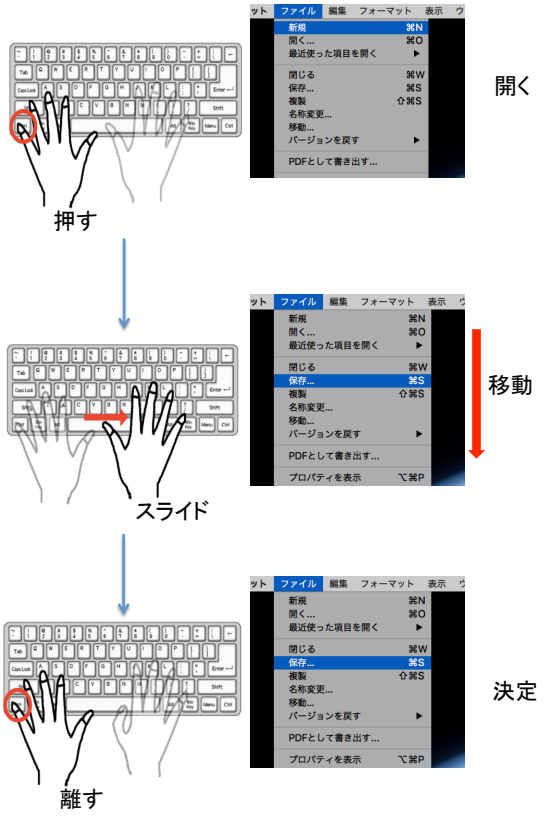

図 1　Ubi-Space の利用イメージ図

1　筑波大学情報学群情報科学類
College of Information Science, School of Informatics, University of Tsukuba
2　筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻
Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba
3　筑波大学システム情報系
Faculty of Engineering, Information and Systems, University of Tsukuba
a)　kmurata@iplab.cs.tsukuba.ac.jp

本研究では，キーボード使用時に，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ホームポジションに手を置いたまま，GUI メニューコマンドの入力を行う手法である Ubi-Space（**図 1**）を提案する．Ubi-Space では，開始キー（HOLDKEY）を押すと，GUI メニューが展開され，コマンドを選ぶ状態になる．その状態で，スペースキー上で親指をスライドさせることにより，コマンドの選択を行う．最後に，押した状態の HOLDKEY を離すとコマンドの決定を行う．これらの操作により，ユーザはポインティングデバイスとの持ち替え時間を必要としない，コマンドの入力を行うことができる．

今回我々は，スペースキーを静電容量タッチセンサによって拡張し，本手法である Ubi-Space によってメニュー選択を行うソフトウェアを実装した．また，Ubi-Space の性能を評価するため，マウス，ショートカットキーによるコマンド入力との比較実験を行い，コマンドの入力速度，正解率，及び主観的評価の検証を行なった．本稿においては，これらについて報告する．

2. 関連研究

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研究がされている．西村ら [4] は，タッチパネルに表示されたキーボードを，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，デバイスを持ち替える動作を必要としない手法を提案している．この手法は，センサを新たに必要としないという利点があるが，タッチパネルにしか対応していないという問題がある．Rekimoto[5] 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操作を行うインタラクションについての研究を行なった．この手法は，既にあるタッチパッドとキーボードを組み合わせることでインタラクションを増やすことを可能としているが，親指をタッチパッドに伸ばし通常の状態よりも大きく手を広げる必要があるため，手が不自然な形になってしまうという問題がある．本研究では，PC のキーボードに新たにセンサを取り付け，手の自然な形を保ったままホームポジションでの入力を行うことで，ポインティングデバイスとキーボードの持ち替え時間の問題の解決を行った．

また，キーボード上でスライドジェスチャを行う手法に関する研究は，Kato ら [6] の研究や，FlickBoard[7] のように，キーボード上全体をインタラクションに用いているが，単体のキー上でのスライドジェスチャの認識は行なっていない．本研究では，スペースキーを拡張することによって，スペースキー上での指のスライド操作の認識を行う．

3. スペースキー上での親指スライド操作によるコマンド入力手法

本稿に置いて示す Ubi-Space は，スペースキー上でなぞるように親指をスライドさせることによって，GUI メニュー内のコマンド入力を行う手法である．スペースキーが横長であるという特徴を利用し，横方向の親指スライドによって，スペースキーが「押されている」か「押されていない」かの 2 値の入力だけでなく，「どのくらい親指が動いたか」という連続的な 1 次元入力もできるように拡張することができる．この操作により，キーボードを使用する際，ホームポジションに手を置いた状態で，ポインティングデバイスとキーボードの間の持ち替え時間をかけずにインタラクションを行うことが可能になる．

Ubi-Space によるコマンド入力は特定のキーに割り当てた HOLDKEY と親指スライド操作を使用する．HOLDKEY は，メニューを開く操作や，コマンドの決定を行う操作に用いる．本稿において，Ubi-Space を用いたコマンドの選択方法として，2 つの選択方式を提案する．

3.1 選択方式 1: 1 段階のコマンド選択

選択方式 1 は，メニュー内の項目（コマンド）の選択を行う，1 段階の選択方法である．全てのコマンドを 1 次元の列の並びとみなし選択を行う．

コマンド入力の流れは以下の 3 ステップからなる．

(1) HOLDKEY を押すと最左のメニューが展開される．
(2) HOLDKEY を押したまま親指スライドを行うことによって，選択するコマンドを順に移動する．
(3) HOLDKEY を離すと選択項目のコマンドが決定される．

親指スライドでコマンドを選択する際，右へのスライドで項目を下に移動，左へのスライドで項目を上に移動させることができる．また，特定のメニュー上の最下部まで項目が進むと右隣のメニューが開き，その最上部の項目へと移動する．同様に，特定のメニュー上の最上部まで項目が進むと左隣のメニューが開き，その最下部の項目へと移動する．さらに，最右メニューの最下にあるコマンドから右スライドで最左の最上にあるコマンドに移動し，最左メニューの最上にあるコマンドから左スライドで最右の最下にあるコマンドに移動することもできる．

3.2 選択方式 2: 2 段階の選択

選択方式 2 は，メニューバー上の項目（メニュー）を選択し，次にそのメニュー内の項目（コマンド）の選択を行う 2 段階の選択方式である．選択方式 1 の場合は，1 次元の列の並びの遠くのコマンドにカーソルを移動する際，時間がかかってしまうことが考えられる．しかし選択方式 2 では，2 次元の行列上での選択のため，選択手法 1 で選択

に時間がかかってしまったコマンドでも短い時間での入力が期待される．

コマンド入力の流れは以下の 5 ステップからなる．

(1) HOLDKEY を押すと，メニューの選択状態になる．
(2) HOLDKEY を押したまま親指スライドを行い，選択するメニューを順に移動する．
(3) HOLDKEY を押し直すと，選択されているメニューの中の最上のコマンドにカーソルが移動し，コマンドの選択状態になる．
(4) HOLDKEY を押したまま親指スライドを行い，選択するコマンドを順に移動する．
(5) HOLDKEY を離すと，選択項目のコマンドが決定される．

なお，選択方式 2 では選択方式 1 と異なり，メニューを超えてのカーソルの移動は行われない．例えば，メニュー内の最下のコマンドからさらに親指を右にスライドさせてもカーソルの移動は行われない．

3.3 サブメニューのコマンドの選択

特定のメニュー内にはサブメニューを持つメニューのコマンドが存在する．そのコマンドの選択時は，階層化されたメニューコマンド上で HOLDKEY を押し直すとサブメニュー上にカーソルが移動し，その中のコマンドの選択状態に移動する．その後の操作は通常のコマンドの選択と同様に，HOLDKEY を押した状態で親指スライドを行うことによってコマンドの選択を行い，HOLDKEY を離すことによってコマンドの決定を行う．

4. 実装

本研究において，我々は Ubi-Space でのコマンド入力を行うシステムを実装した．システムとしてハードウェア側であるセンサと，ソフトウェア側であるメニューの制御を行うシステムを実装した．

4.1 使用した機器及びライブラリ

使用した機器は Apple 社製の MacBook Air（13 インチ），および Dell 社製のキーボードである．センサの実装に，銅箔テープを用いた．銅箔テープから得られた静電容量を制御するマイコンとして Arduino MEGA を用いた．静電容量を取得するために Arduino のライブラリである CapacitiveSensor[*1]を使用した．ソフトウェア側の実装には AppleScript を使用し，シリアル通信のライブラリとして SerialPort X[*2]を用いた．

[*1] http://playground.arduino.cc/Code/CapacitiveSensor
[*2] http://mac.softpedia.com/get/Communications/SerialPort-X.shtml

4.2 ハードウェア実装

本研究では，ハードウェア実装としてスペースキー上に 5 枚の銅箔テープを貼り付けた（**図 2**）．この 5 枚の銅箔テープの並びをタッチセンサとして扱う．また，回路を**図 4** に示す．回路図の sensor1〜sensor5 は各銅箔テープを示す．スペースキー上をタッチした時の静電容量を取得し，タッチ位置およびフレームごとのタッチ位置の差分を計算することで，親指スライドの認識を行なった．この時の銅箔テープはそれぞれ三角形または平行四辺形の形状である．このような形状とすることで，タッチした部分によって各銅箔テープと指の間の静電容量が変化するため，タッチ位置の取得が可能となる．また，この時に取得するタッチ位置は，横軸一次元のタッチ位置である．以下に 5 枚の銅箔テープからタッチ位置を取得するアルゴリズムを示す．

(1) それぞれの銅箔テープの要素番号 i 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 $Min[i], Max[i]$ とする．
(3) 取得した静電容量を $C[i]$ 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 $C'[i]$ を得る．

$$C'[i] = \frac{C[i] - Min[i]}{Max[i] - Min[i]} \tag{1}$$

この正規化された $C'[i]$ 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．

$$x = \frac{\sum_{i=1}^{5} C'[i] * 10i}{\sum_{j=1}^{5} C'[j]} \tag{2}$$

x は double 型で得られるため，int 型にキャストし，整数値部分のみを取り出し，x' とする．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．

センサから得られた重心のフレームごとの差分が，親指の相対位置である．ソフトウェア側にはこの相対位置を出力する．

センサからは 10 ミリ秒ごとに静電容量を取得し，タッチ位置の計算を行う．フレームごとに得られる静電容量にはノイズによる値のぶれが発生する．そのため，移動平均フィルタにより値の平滑化を行った．平滑化を行なっていない時，平滑化を行なった時のタッチ位置の変化を**図 3** に示す．図中の (A)，(B)，(C) はそれぞれスペースキー上の最左，真ん中，最右をタッチした時の位置を示す．移動平均フィルタのフィルタ係数は 15 とした．平滑化によりノイズを抑えた値が得られるが，15 回の取得分の値の平均を取るため，実際の値が反映されるまで遅延時間が発生する．

また，取得できる静電容量は，環境や使用する人によっ

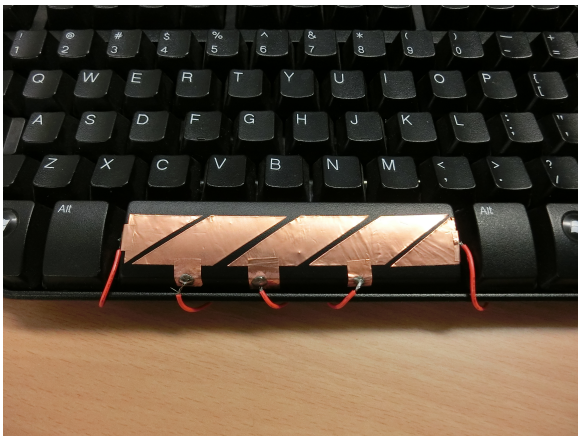

図 2 銅箔テープを用い，タッチ検出を可能としたスペースキー

て変化する．そのためセンサを使う前にキャリブレーションを行う必要がある．したがって，本研究ではキャリブレーションの機能を実装した．キャリブレーションは，回路上のタクトスイッチを押し，親指をスペースキー上で 1 往復スライドさせることによって行われる．キャリブレーションによって設定される値は，タッチされていない時の静電容量 $Min[i]$，およびタッチされた時の最大の静電容量 $Max[i]$ である．

4.3 ソフトウェア実装

メニュー選択制御を行うソフトウェアを実装した．実装したソフトウェアは，使用するアプリケーションの GUI メニュー部分のみを呼び出し，その制御を行う．そのため任意の GUI アプリケーションで使用することが可能である．ソフトウェアはシリアル通信によって親指の相対位置を読み取り，その位置分選択カーソルの移動を行うことによって，コマンドの選択に反映させる．本手法で用いる HOLDKEY は Control キーに設定した．

5. 評価実験

本手法におけるコマンド入力の評価を行うため，既存手法との比較実験を行った．実験は 6 名の被験者（20 歳〜23 歳の大学生および大学院生，男性 4 名，女性 2 名）を対象として行った．全員が日常でコンピュータを使った文書作成やプログラミングを行っていた．

5.1 実験設計

被験者には実験用アプリケーションを用い，指定されたコマンドを既存手法，本研究の提案手法による GUI メニューのコマンド入力を行ってもらう．

実験で用いる手法は既存手法として (1) マウスによる入力，(2) ショートカットキーによる入力，提案手法として (3) 選択方式 1，(4) 選択方式 2 による入力の合計 4 つの手法である．コマンド入力はセッション単位で行ってもらう．1 セッションあたり，10 個の異なるコマンド入力から成り，1 手法あたり 3 セッションで構成される．したがって，被験者 1 名あたりのコマンド入力数は，4 手法× 3 セッション× 10 コマンド=120 コマンドである．コマンド入力の際には，アプリケーション画面に入力すべきコマンドが指示される．コマンドの入力を行った時，指示されたコマンドに対して正解であったか不正解であったかどうかに関わらず，次のコマンドが指示される．

ショートカットキーのセッション内で，コマンドに対応するショートカットキーがわからない場合は command キーを押しながら enter キーを押すことで次のコマンドへとスキップを行うことができる．

実験ではまず，被験者に GUI メニューコマンドの並び，およびショートカットキーを把握するための練習タスクを 10 分間設ける．その後，本番タスクをマウスによる入力から行なってもらう．また本番タスク内で，選択方式 1 の最初のセッションの前，選択方式 2 の最初のセッションの前には，それらの手法の練習時間を設ける．

実験後に被験者に，本手法についての主観的評価を行うアンケートへの回答を行ってもらった．アンケートの設問を**表 1**，および**表 2** に示す．なお表 1 の設問 7，9，および表 2 の設問 7，11，12 は記述項目である．設問 3 以降の記述項目以外の設問は，1〜5 の 5 段階評価であり，この値が高いほど有用性を示すよう設定した（設問 3 の場合，1：使いにくい，5：使いやすい）．また，設問 9, 10 も 5 段階評価であり，1 を使わない（行わない），5 を使う（行う）に設定した．

表 1 アンケート（選択方式 1）

設問	内容
1	年齢
2	性別
3	本手法は使いやすいと思ったか
4	本手法は誰にでもすぐに使えると思った
5	本手法を使いたいと思った
6	本手法の操作に煩わしさを感じたか
7	どのような操作に煩わしさを感じたか
8	本手法を使うことで素速いコマンド選択を行えたと思ったか
9	他，本手法に対しての感想，意見

表 2 アンケート（選択方式 2）

設問	内容
1	年齢
2	性別
3	本手法は使いやすいと思ったか
4	本手法は誰にでもすぐに使えると思った
5	本手法を使いたいと思った
6	本手法の操作に煩わしさを感じたか
7	どのような操作に煩わしさを感じたか
8	本手法を使うことで素速いコマンド選択を行えたと思ったか
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
10	普段 PC を使う際，ショートカットキーをよく使うか
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
12	他，本手法に対しての感想，意見

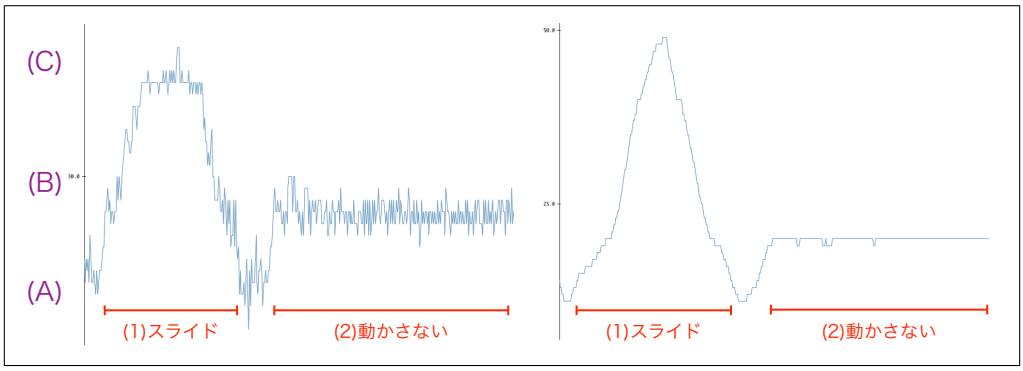

図 3 平滑化を行っていない時（左）と行った時（右）の親指のタッチ位置の変化

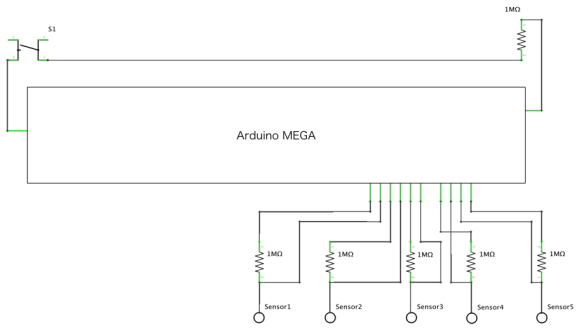

図 4 センサの回路図

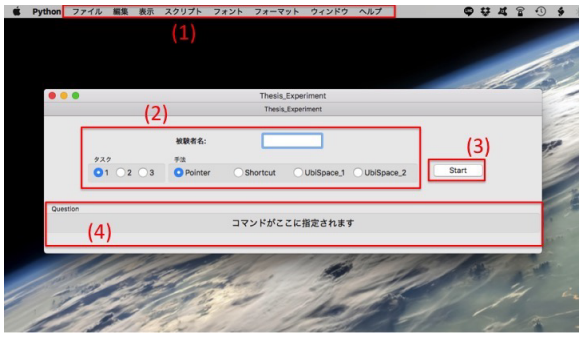

図 5 実験アプリケーション画面

5.2 実験用アプリケーション

実験にて使用するアプリケーション画面を**図 5** に示す．アプリケーションのプログラミング言語に Python を用い，GUI 部品を扱うライブラリとして wxPython[*3]を使用した．アプリケーションの GUI メニューの構成，および GUI メニュー内のコマンドに対応するショートカットキーの構成は MacOS デフォルトのアプリケーションであるスクリプトエディタと同様の構成とした．被験者は (1) の GUI メニューからコマンドの選択を行う．(2) の被験者名，タスク (セッション)，手法の入力は実験者が行い，(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，セッションが開始する．コマンドは (4) に示される部分に指定される．また，アプリケーション内では各コマンドの入力にかかった時間と，指定されたコマンド，実際に入力されたコマンドを csv ファイルに記録する．実験後に，記録された csv ファイルを用いてコマンドの入力時間，および正解率の分析を行った．

5.3 結果

各手法によるコマンド入力時間の平均およびセッションごとの正解率を**図 6** に示す．グラフ内のエラーバーは標準偏差を示す．また，被験者ごとのコマンド入力時間の平均，セッションごとの正解率を**図 7** に示す．コマンドの入力時間の平均は，マウスが 4.18 秒 ($SD = 1.71$)，ショートカットキーが 3.16 秒 ($SD = 2.02$)，選択方式 1 が 13.52 秒 ($SD = 8.62$)，選択方式 2 が 9.28 秒 ($SD = 5.12$) であり，ショートカットキーの入力時間が最も短く，選択方式 1 の入力時間が最も長かった．本研究のどちらの提案手法も，既存手法より長い入力時間となった．

また，各手法によるセッションごとのコマンド入力の正解率の平均は，マウスが 100%($SD = 0.00$)，ショートカットキーが 70.5%($SD = 21.5$)，選択方式 1 が 92.7%($SD = 16.9$)，選択方式 2 が 83.3%($SD = 15.6$) であり，マウスの正解率が最も高く，ショートカットキーの正解率が最も低かった．本研究のどちらの選択方式も，ショートカットキーよりも正解率が高く，マウスより正解率が低い結果となった．ショートカットキーの入力の中で，スキップが行われたコマンドは不正解とみなし，全体，および被験者別の正解率の分析を行った．また，図 6 のショートカットキーの灰色のグラフは，スキップが行われたコマンドを不正解とみなさない場合のグラフである．

表 3，**表 4**，**表 5** および**表 6** に実験後のアンケート結果を示す．

表 3 選択方式 1 のアンケート結果（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

[*3] https://wxpython.org/

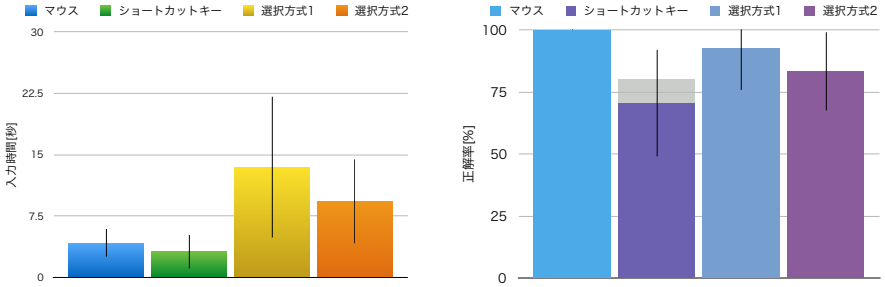

図 6 各手法の入力時間の平均（左）および正解率（右）

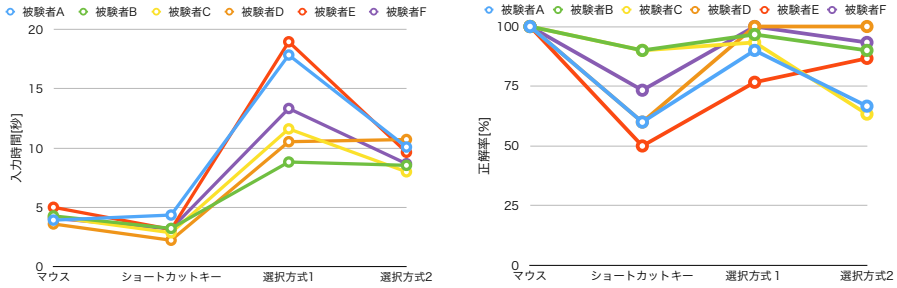

図 7 被験者別の入力時間の平均（左）および正解率（右）

表 4 選択方式 1 のアンケート結果（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.
	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思った.
	D	やはりマウスと同様に素速く動かすと速く移動できるようになると嬉しい.

表 5 選択方式 2 のアンケート結果（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 6 選択方式 2 のアンケート結果（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が，遠くの項目に行く時は便利だと思う.
	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.

5.4 考察

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．親指を素速くスライドさせて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して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速く動くため，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，実験の様子から，カーソルを速く動かそうと，フリック操作のように親指を動かしていた被験者の場合，入力時間が長くかかっていた．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し，より直感的な操作を可能とすることが一つの課題である．

また正解率が 100%ではなかった原因としてノイズの問題がある．被験者は入力の間違いとして，指示されたコマンドに対し，その前後隣のコマンドの選択を行なっていたことが多かったため，ノイズによる選択カーソルのずれによって生じた入力間違い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．したがって，このノイズの問題を解消し，精

度を上げることがもう一つの課題である．

さらにアンケートより，「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，通常の操作の支援を可能とす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の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．

6. 終わりに

本稿において，持ち替え時間を必要とせず，ホームポジションでのコマンド入力を可能とする手法である Ubi-Space を提案した．Ubi-Space はスペースキー上での親指スライド操作をコマンド入力に用いる．本研究では銅箔テープを用いて実装したタッチセンサによってスペースキーを拡張し，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実装した．さらに，アプリケーションの GUI メニュー上のコマンド選択に本手法を使えるよう，メニュー制御を行うソフトウェアを実装した．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とし，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，正解率は提案手法のどちらも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，ノイズによる入力間違いが生じていたことが分かった．これらの問題から，遅延時間を無くし，精度の向上を目指すことが課題である．

また，キーボードを主に使用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利用することを考えている．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．

参考文献

[1] Karat, J., McDonald, J. E. and Anderson, M.: 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).
[2] Lane, D. M., Napier, H. A., Peres, S. C. and Sándor, A.: 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).
[3] Malacria, S., Bailly, G., Harrison, J., Cockburn, A. and Gutwin, C.: Promoting Hotkey Use Through Rehearsal with ExposeHK, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '13, New York, NY, USA, ACM, pp. 573–582 (2013).
[4] 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，Vol. 2011, No. 3, pp. 173–176 (2011).
[5] Rekimoto, J.: ThumbSense: Automatic Input Mode Sensing for Touchpad-based Interactions, *CHI '03 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '03, New York, NY, USA, ACM, pp. 852–853 (2003).
[6] Kato, J., Sakamoto, D. and Igarashi, T.: Surfboard: Keyboard with Microphone As a Low-cost Interactive Surface, *Adjunct Proceedings of the 23Nd Annual ACM Symposium on User Interface Software and Technology*, UIST '10, New York, NY, USA, ACM, pp. 387–388 (2010).
[7] Tung, Y.-C., Cheng, T. Y., Yu, N.-H., Wang, C. and Chen, M. Y.: FlickBoard: Enabling Trackpad Interaction with Automatic Mode Switching on a Capacitive-sensing Keyboard, *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, New York, NY, USA, ACM, pp. 1847–1850 (2015).