Fix and Slide: 文字列全体を操作することによるキャレットの操作手法

鈴木 健司 * 岡部 和昌 * 坂本 竜基 * 坂本 大介 †

概要. スマートフォン等のタッチ操作可能な小型携帯端末においては，指が画面を隠してしまい操作が難しくなってしまうという問題がある．我々は，この問題が「画面をタッチする指で，操作対象そのものを操作する」ことで生じていることに注目し，操作対象そのものではなく，画面全体を操作することでこの問題を解決することを試みた．提案するインタフェースでは，まずユーザは画面をタップすることで，文字列上の任意の場所にキャレットを置く．ここでユーザはキャレットそのものを操作するのではなく，キャレットを置いた場所を固定し，文字列全体を移動させることにより，目的の位置にキャレットを移動させる．本稿では提案手法を利用した文字列選択機能を実装し，評価実験を行った．この結果，はじめて提案手法を使うユーザであっても従来手法と変わらない速度で作業が完了できたか，もしくは早い場合があった．さらにユーザビリティ調査の結果から従来手法よりも使いやすいインタフェースであると評価された．

1 はじめに

計算機の画面上をポインティングしたり，文字選択することは計算機を利用する上で最も基本的な操作である．この操作について，我々は一般的にマウスやトラックボール，最近ではタッチパッドを使うことが多い．これらの非直接操作手法はデスクトップ型の計算機を利用する場合に一般的である．一方で，近年スマートフォンに代表されるタッチ操作式の小型携帯端末が普及し「操作したい対象を，画面上で直接触る（タッチする）」ことで操作する直接操作手法が一般的になってきた．本操作手法は操作対象に直接触れることができるという点で直感的な操作が可能であるが，一方で「指が操作対象を隠してしまう」「指が操作対象よりも大きく，正確な操作が難しい」という新たな問題が生じた．これがいわゆる *“Fat Finger Problem”* である [10].

この問題は文字操作を行う場合にも生じる．すなわち，画面に表示された文字列中の文字を選択する場合において，ユーザはまずキャレットを文字選択を開始する位置に移動させ，文字選択モードにしたのち，文字選択の終了地点にキャレットを移動させる（図 1 左）．ただし，一般的に前述した *Fat Finger Problem* により，一度のタップでは正確にキャレットを配置することが難しいため，何度かタップとドラッグを繰り返す必要がある．同様に，画面上をドラッグするなどして文字選択の終了地点までキャレットを移動させる場合でも正確な操作が難しいため，何度かタップとドラッグを繰り返す必要がある．まとめると，一般的には小型携帯端末においては，デスクトップ型の計算機と比較してポインティングや文字選択が難しいと言える．

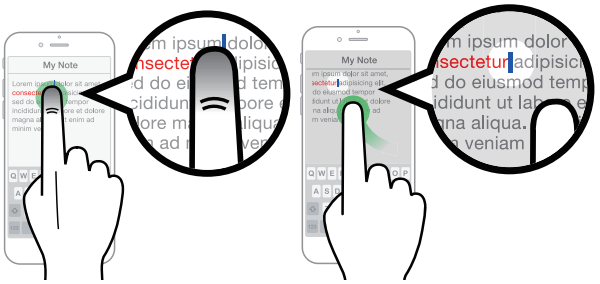

図 1. 一般的なキャレット操作手法（左）と提案手法（右）：ユーザは背景となる文字列をドラッグすることで任意の場所にキャレットを移動することができる.

本稿では，小型携帯端末における文字選択操作について，直接対象を触って操作する直接操作手法ではなく，その背景となる文字列全体を操作することによるキャレット操作および文字選択手法の提案を行う（図 1 右）．本手法においては，まずユーザは画面をタップすることでキャレットを文字列中に置く．このキャレットは画面上で固定され，ドラッグ操作はできない（ただし，タップによる再配置は可能）．その後，ユーザは背景となる文字列全体をドラッグすることで，文字列全体を移動させることができる．文字列全体を移動させた後，指を画面から離すと，その時点でキャレットが存在する文字列中の位置にキャレットが移動する．これによりキャレットを直接操作するのではなく，間接的に操作することが可能となる．我々は提案手法を実装した後，提案手法のパフォーマンス測定やユーザビリティ調査のための実験を実施した．実験においては現在一般的に利用されている iOS の文字列選択手法と比較を行った．本稿ではこれらについて詳述する．

* ヤフー株式会社
† 東京大学

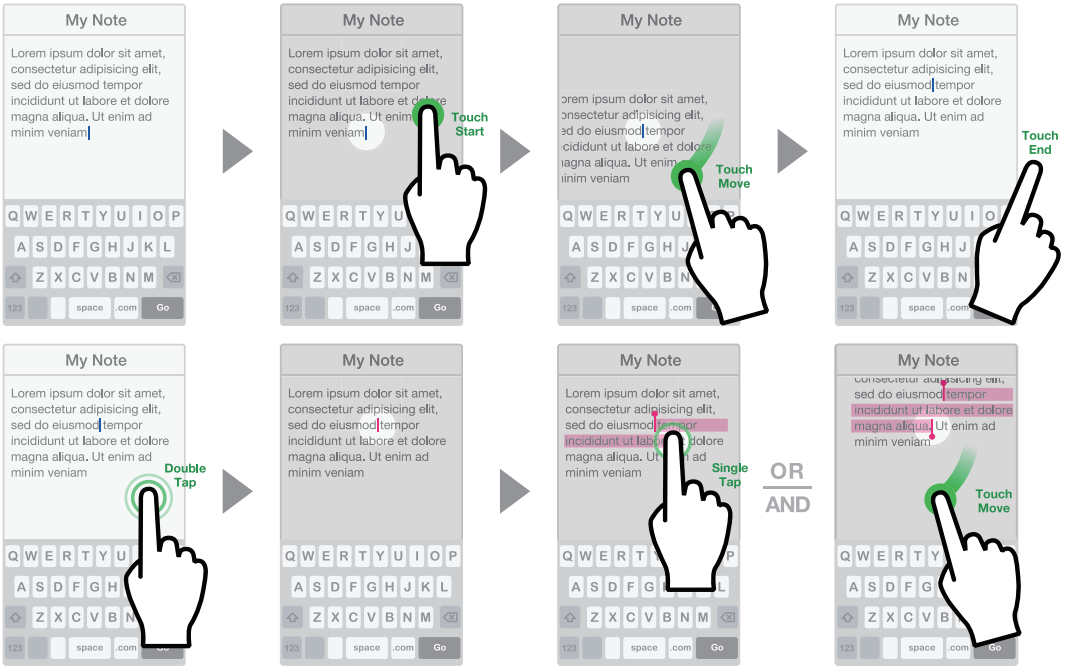

図 2. 提案手法の概要．はじめにキャレットを固定（Fix）し，その後背景文字列を移動（Slide）することで文字列選択ができる．

2 関連研究

Fat Finger Problem は小型携帯端末のインタフェースに関する研究領域において良く知られた問題であり，これまでに様々なアプローチの研究が実施されてきている．Shift はその中でも最も有名な研究であるが，この手法では指の下の領域が「ふきだし」として指の周囲に表示されるものである [10]．本手法は現在の iOS 等に搭載されている．TapTap は画面をタップすることにより，タップされた領域を拡大して表示するポップアップを利用した手法である [8]．他にもタッチ操作可能な機器に対してオフセットカーソルを利用する手法が存在するが，直接操作手法よりも操作時間が遅かった [9]．これまでは小型携帯端末の「表面」のタッチスクリーンを利用するものであったが，反対の「裏面」を活用した研究も存在する [11]．これらの研究はポインティング操作において *Fat Finger Problem* を回避するための手法であり，キャレット操作や文字列選択に注目した研究ではなかった．

正確なポインティングに関する手法についても概観する．これまでに高精度にタッチ操作をセンシングする技術 [6] や，確率モデルを採用してタッチされるポイントを予測する手法 [3]，マルチタッチスクリーン上で両手を利用して正確に対象の選択を行う技術の提案 [2] が行われてきている．最近になって Eady らによって小型携帯端末上のキャレット操作に関する手法について提案があったが [5]，端末の周囲に曲げセンサを取り付ける必要があるなど，端末の拡張が必要であった．また，形態素解析を用いた日本語文字列選択手法についての提案もされてきているが [7]，それぞれ本研究の目的とは大きく異なる．

3 Fix and Slide

本稿で提案する Fix and Slide の主たるコンセプトはポインティングや文字選択方法として文字列を含む背景を操作するということである（図 2）．現在の標準的なキャレット操作手法はスクリーンに表示された文字上でユーザの指をドラッグさせるような方法で移動させるようになっているが，本提案手法はまずキャレットを固定し，その周囲の文字を含む背景を動かすことによってキャレットを移動させようとするものである．本手法によりユーザが小型携帯端末を操作する際に *Fat Finger Problem* のように指の下の領域を隠すことなく，ポイントしたい場所や，文字選択したい場所を確認しながら操作できるようになることが期待される．

3.1 キャレット操作手法

小型携帯端末上でキャレット操作を行う際に問題となるのが，ユーザがスクリーン上をタップしても，文字列中の目的の位置に正確にキャレットを一度のタップで置けないことである．この場合，多くは目的の位置のすぐ側に置かれてしまうため，何度かタップするか少しずつドラッグするかのどちらか，もしくは両方の操作を行うことが求められることが

多い．目的の位置が指の下に来てしまうようなインタフェースの場合には，さらに操作が難しくなる．前述した Shift [10] を適用した方法の場合には，指の周囲の適切な場所に「ふきだし」が表示され，その中に指の下に位置する情報が提示されるが，操作の難しさを解決することには直結しない．我々はこれは直接操作手法の問題点であると考えている．

これに対して本稿で提案する Fix and Slide においては間接操作手法を提案する．ユーザはまず直接操作手法と同様にスクリーンをタップしてキャレットを配置する（図 2 上段）．その後，スクリーン上の任意の場所にタッチしてドラッグすることで，背景を動かすことができる．このとき，キャレットは最初にタップされた場所に固定されるため，キャレットは固定されたまま，背景が移動することになる．ユーザはキャレットを文字列中で移動させるのではなく，文字列（背景）を移動させることによってキャレット操作を行うことが本提案手法の特徴である．

3.2 文字列選択手法

ユーザが最初にキャレット操作を行った後，文字列選択操作に移行する．画面をダブルタップ（2 度画面を素早く触る）することで，文字列選択モードに変更することができる（図 2 下段）．これ以降はキャレット操作手法と同様で，最初にスクリーンをタップし，スクリーンを動かすことで背景および文字列全体を移動させる．これにより，文字列選択の終点を決めることができる．この操作はダブルタップを挟んでキャレット操作手法を 2 度実行していることと同義である．

4 既存手法との比較実験

提案手法の有効性を検証するために，現在一般的に利用されている既存のインタフェース，具体的には iOS 上で実装されているインタフェースとの比較実験を行う．ここでは文字列選択をタスクとして扱う．すなわち，実験参加者は Fix and Slide 手法と iOS 上での文字列選択手法を使用して文字列選択を行うよう指示される．以降は iOS 上での文字列選択手法を Default 手法と呼ぶ [1]．我々はこの文字列選択操作を行う際の作業実行時間（パフォーマンス）およびユーザビリティに関するフィードバックを得ることを目的として実験を行う．

4.1 タスク：文字列選択

本稿で実施する実験では，文字列選択タスクを採用する．使用する言語は英語とする．英語は単語間

図 3. 実験で使用するフォントサイズ 3 種：12 ポイント（上段），14 ポイント（中段），16 ポイント（下段）．中段の図は文字列の部分選択条件の例を示し，下段は外側選択条件の例を示している．

がスペースで区切られていることが特徴である．一方で，日本語は単語間はスペースで区切られておらず，句読点で文章の終わり，もしくは文章の切れ目を表現することが特徴である．今回の実験では「日本語の文字列選択は，英語における長い単語の途中を切り出すこと．」と見做すことができることに注目し，英語を使用することとした．

実験で使用する文字列はランダムに生成される [2]．本実験では生成された文字列のうち，最初の 2 から 4 段落を使用する．ただし，後述するように特殊なケース，具体的には URL については手動で挿入する．実験において我々は 5 つの文章を用意した．

4.2 条件

実験では 2 つの文字列選択手法（提案手法と Default 手法）に加えて，以下の条件を設定する．

4.2.1 フォントサイズ

文字列選択タスクの作業実行時間はフォントサイズに大きく依存することが考えられる．このため，本実験では 12 ポイント，14 ポイント，16 ポイントの 3 つのフォントサイズを用意する（図 3）．これらのフォントサイズは Apple 社の iOS Human Interface Guidelines[1] に従って選択した．すなわち：

“ユーザが「極小」を選択しても、11 ポイン

[1] 誌面の都合上，Default 手法の文字列選択方法はこちらを参照されたい．端的には iOS 上には文字列選択手法は複数存在し，実験では全て利用可能であった．https://support.apple.com/kb/PH3577?locale=ja_JP

[2] http://randomtextgenerator.com/

ト未満にはしないでください。ちなみに「本文」スタイルでは、「大」のとき 17 ポイント（デフォルト値）で表示するようになっています。”

とされており，12, 14, 16 ポイントは妥当なフォントサイズの選択であると考えている．

4.2.2 文字列選択の位置

フォントサイズと同様に，文字列中のどの部分を選択するのかも重要な要素となることが考えられる．本実験では「文字列の部分選択」と「文字列の外側選択」の 2 条件を設定する．

文字列の部分選択 現代的な OS に搭載されている賢い選択機能は有用であるが，一方でそれは外側選択を行う場合のみである．すなわち，文字列の部分選択を行う際には，あまり役に立たない．例えば，英語の場合には数字を部分的に選択したり，長い URL の場合，ホスト名だけを抜き出したい場合には有効に機能しない．日本語の場合には基本的に部分選択となるため，いわゆる賢い選択の恩恵を受けることはあまりない．さらに悪いことに賢い選択の場合には文字列の最初と最後にスナップ（吸着）する性質があるため，問題が生じることもある．前述した例と同様に，部分選択の例を表 1 に示す．

文字列の外側選択 まず，文字列の外側選択について説明する．現代的な携帯端末用の OS においては，文字列を選択する際にユーザの指の動き，すなわちキャレットの位置を認識し，文字列の最初（文字列間にスペースがある部分）もしくは最後（同様に文字列間にスペースがある部分）に賢く移動（スナッピング）する機能が実装されている．一般的にユーザは文字列の最初から最後までを選択することを考えると，この機能によりユーザは素早く文字列を選択することが可能となっている．ここで，このような文字列選択を「外側選択」と呼ぶ．

実験において参加者は，表 1 の外側選択に示される文字の緑色の背景となっている部分を選択するように指示される．実際の表示では図 3 のように最初と最後に緑色の括弧が表示される．これらの文字列はランダムに生成されたものである．

4.3 実験参加者と手続き

30 名のボランティアが実験に参加した．男性は 19 名，女性は 11 名であり，年齢は 23 歳から 42 歳までであった（$mean = 30.6$, $S.D. = 4.9$）．全ての実験参加者はスマートフォンを所有しており，1 名が Android 端末（3.3%），23 名が iPhone（76.7%）を，両方を所有しているのは 6 名であった（20.0%）．すなわち，1 名を除いて全ての参加者が iPhone を所有していた．また，全ての参加者はスマートフォン上での文字選択を行ったことがあり，20 名については文字列選択に慣れていると回答した．どのような場面で文字列選択を行うかについては，URL をコピーする場合，インターネット検索をする場合，ツイートやブログに書く場合と回答した例が多かった．さらに，多くの参加者が文字列選択する際に指で選択する領域が隠れてしまうことに不満を持っており，同様に iOS での「賢い選択」によって不用意にキャレットがジャンプしてしまうことに不満を持っていた．

表 1. 文字列選択の位置の例

部分選択	Admitted add peculiar get 83,566 joy doubtful.
	http://www.koqiksmkis.test/a.php?q=gao d=dplzk
	http://www.hksllaz.test /test.html ?x=ddsa
外側選択	Maids hoped gay yet bed asked blind dried 2,000,000 point.
	Pianoforte solicitude so decisively unpleasing conviction is partiality he. Or particular 278,061 so diminution entrea- ties oh do.

実験の手続きについて簡単に述べる．実験参加者は 2 つの文字列選択手法（Fix and Slide と Default）の両方を，それぞれ 6 回ずつ使用する．実験では 3 つのフォントサイズと，2 つの文字列選択位置の条件が存在する．文字列選択手法 2 種類とフォントサイズ 3 種類については，参加者が経験する順番のバランスを取った．一方で，文字列選択位置 2 種類については学習効果がないため，固定とした．参加者は 1 つの試行について，5 つの文字列選択タスクを行う．以上を簡単にまとめると，1 つの文字列選択手法について都合 30 回の文字列選択を実施し，実験参加者は合計で 60 回の文字列選択を行う．文字列選択を行う位置はランダムではなく，統制を取るために予め決められており，被験者間で同じ文字列を選択することとなる．それぞれの文字列選択手法を利用する前には，操作に慣れるためのトレーニングを時間を設けた．この時使われる文字列は本番のものとは別のものであった．

5 実験結果

5.1 作業実行時間

作業実行時間 [秒] の平均について，フォントサイズ毎にまとめたものを図 4 に示す．Fix and Slide 手法および Default 手法の操作手法毎に，部分選択，外

部選択と比較するように示している．これに対して3要因分散分析(被験者内計画;独立変数として，操作手法，フォントサイズ，およびキャレットの位置．従属変数として，作業実行時間)を実施した．この結果，有意な2次の交互作用と（操作手法 × フォントサイズ × キャレットの位置）$[F(2,58)=8.60, p<.01]$，1次の交互作用が確認された（(操作手法 × キャレットの位置) $[F(1,29)=16.07, p<.01]$，および（操作手法 × フォントサイズ）$[F(2,58)=3.41, p<.05]$）．また，単純主効果の操作手法 $[F(1,29)=11.19, p<.01]$ とフォントサイズ $[F(2,58)=9.55, p<.01]$ に有意差が確認された．キャレットの位置については有意差が確認されなかったが，有意傾向が確認された $[F(1,29)=3.13, p<.1]$．

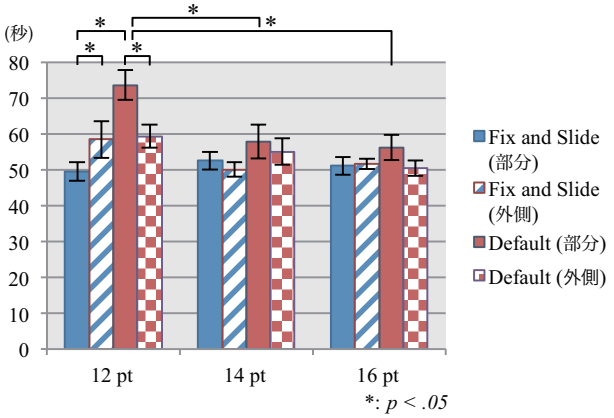

図 4. 3 要因分散分析の結果と標準誤差.

操作手法 × 文字列選択の位置 2次の交互作用について，操作手法と文字列選択の位置に注目して2要因の分散分析を行った（被験者内計画）．この分析は3つのフォントサイズ毎に実施した．この結果，12ポイントの場合に交互作用が有意であった $[F(1,29)=25.75, p<.01]$（図4および図5の12ポイントの部分）．そこで，各要因の単純主効果を分析した結果，操作手法 × 文字列選択の位置（部分選択）$[F(1,29)=51.50, p<.01]$，文字列選択の位置 × 操作手法（Fix and Slide）$[F(1,29)=6.68, p<.05]$，文字列選択の位置 × 操作手法（Default）$[F(1,29)=16.69, p<.01]$ について有意な差が確認された．

操作手法 × フォントサイズ 図5に操作手法 × フォントサイズについて，文字列選択位置に注目した結果を示す．部分的な文字列選択の場合，提案手法はフォントサイズに依存せず，作業実行時間は一定だが，Default 手法の場合はフォントサイズが大きくなるにつれて作業実行時間が早くなる傾向が確認された．ここで部分選択に注目して2要因分散分析（被験者内計画）を実施した結果，交互作用が確認された $[F(2,58)=8.89, p<.01]$．Bonferroni 法を用いた多重比較を実施した結果，Default 手法の場合，フォントサイズが14，16ポイントの場合に比べて12ポイントの場合に有意に作業時間が長いことが確認された（$MSe=206.2621, p<.05$）．また，外側選択に注目して2要因分散分析を実施した結果，主効果（フォントサイズ）が有意であった $[F(2,58)=5.82, p<.01]$．ここで Bonferroni 法を用いた多重比較を実施した結果，12ポイントの場合が有意に遅い結果であった（$MSe=179.5455, p<.05$）．

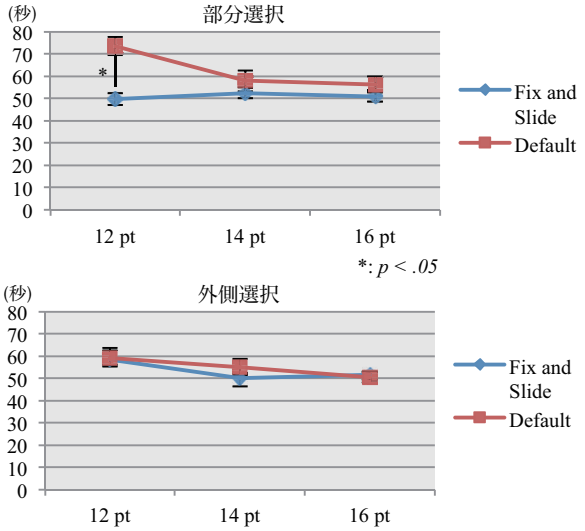

図 5. 操作手法 × 文字列選択の位置の結果と標準誤差

5.2 ユーザビリティ

続いて，各操作手法を使用した後に実験参加者に回答を依頼したアンケートの分析を行う．ここでは System Usability Scale (SUS) [4] をもとに，本実験用に微修正したアンケートを実施した[3]．SUS は5段階のリッカート尺度の質問10項目で構成される，ユーザビリティ評価のためのアンケートであり，最終的に100点満点でスコアが得られるものである．この結果 Fix and Slide 手法は68.67点，Default 手法は49.5点と評価され，これらについて分散分析を実施した結果，提案手法が有意に高い結果であることが確認された $[F(1,29)=19.62, p<.01]$．それぞれの項目についても分散分析を行った結果，Q1:“頻繁に使いたいと思った”$[F(1,29)=50.27, p<.01]$，Q3:“使いやすいと思った”$[F(1,29)=28.29, p<.01]$，Q9:“自信を持って使うことができた”$[F(1,29)=11.15, p<.01]$ で有意な差が確認された（図6）．また，Q5:“この操作方法がスマートフォンの操作方法として良く統合されていると思った” という項目でも有意な差が確認されたことも併せて，Fix and Slide は Default と比較して使いやすく，またユーザはスマートフォンのインタフェースとして Fix and Slide 手法を統合していくことを歓迎していることが示された．

[3] 誌面の都合上，それぞれのアンケート項目は [4] を参照されたい．なお，SUS を日本語にする際に質問が曖昧な表現になる部分を補う修正をした．

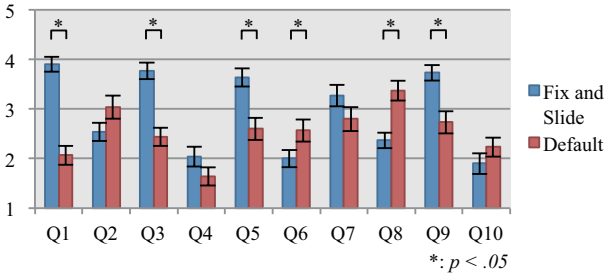

図 6. SUS の結果と標準誤差

6 議論

本研究で実施した実験の結果は以下のようにまとめられる．

- ユーザは提案手法（Fix and Slide）をはじめて使用するにもかかわらず，作業時間は標準的な操作手法と変わらない結果であった．ただし，フォントサイズが小さい場合には提案手法のほうが有意に作業時間が短い例もあった．
- SUS の結果から，ユーザビリティの観点からは提案手法は標準的な操作手法よりも有意に使いやすいインタフェースであると評価された．

上記の結果は概念実証（Proof of concept）には十分な結果であると考えているが，一方で提案手法についての制限や将来課題は残っている．

まず，本研究で実施した実験は提案手法を iOS で実装されている文字列選択手法と比較しただけであり，他の OS，例えば Android 上で実装されている手法とは比較していない．今回の実験は概念実証には十分な実験ではあったと考えているが，一方で他の手法との比較は必要であると考えている．次に，実験においては長い文章を選択する場面を想定しておらず，この場合の実験についても必要であると考えている．インタフェースの実装としては，例えば一本指のドラッグ操作で選択，二本指のドラッグ操作でスクロール操作を行い，長い文章の選択をすることは可能であるが，実用化に向けては事前検証が必要であると考えている．

最後に，学習に関するコストについて触れる．本研究で提案したインタフェースはこれまでの概念とは全く異なり，キャレット操作および文字列選択に背景を操作するというものであった．このため，学習コストが大きいのではないかという批判が考えられるが，しかし実験には 1 名を除いて全員が iPhone を所有し日常的に使用しているユーザが参加しており，この条件においてもはじめて触る Fix and Slide の操作速度のほうが統計的に有意な差で早く，その他については統計的な差が確認されない程度の速度であったことを考えると，学習コストは高くなく，ユーザビリティ評価の結果と合わせて考えると，十分に受け入れられやすいものであると考えられる．

7 むすび

本稿では，小型携帯端末における文字選択操作について，従来の直接対象を触って操作する直接操作手法ではなく，その背景となる文字列全体を操作することによるキャレット操作および文字選択手法の提案を行った．現在標準的に利用されているインタフェースとの比較実験を行った結果，ユーザは提案手法をはじめて使うにも関わらず，フォントサイズが小さい場合は提案手法のほうが有意に作業時間が早く，またその他の場合では標準のインタフェースとほぼ同等のパフォーマンスであった．さらに，ユーザビリティの観点では提案手法のほうが高い評価を得た．

参考文献

[1] Apple. iOS Human Interface Guidelines: 色とタイポグラフィ. `https://developer.apple.com/library/ios/documentation/UserExperience/Conceptual/MobileHIG/ColorImagesText.html`.

[2] H. Benko, A. D. Wilson, and P. Baudisch. Precise Selection Techniques for Multi-touch Screens. In *Proc. CHI '06*, pp. 1263–1272. ACM, 2006.

[3] X. Bi and S. Zhai. Bayesian Touch: A Statistical Criterion of Target Selection with Finger Touch. In *Proc. UIST '13*, pp. 51–60. ACM, 2013.

[4] J. Brooke. SUS-A quick and dirty usability scale. *Usability evaluation in industry*, pp. 189–194, 1996.

[5] A. K. Eady and A. Girouard. Caret Manipulation Using Deformable Input in Mobile Devices. In *Proc. TEI '15*, pp. 587–591. ACM, 2015.

[6] C. Holz and P. Baudisch. The Generalized Perceived Input Point Model and How to Double Touch Accuracy by Extracting Fingerprints. In *Proc. CHI '10*, pp. 581–590. ACM, 2010.

[7] 三浦 元喜, 最所 賢至, 清弘 祥太. タブレット端末における日本語形態素を考慮した文字列範囲選択手法. In *Proc. WISS '14*, pp. 121–122, 2014.

[8] A. Roudaut, S. Huot, and E. Lecolinet. TapTap and MagStick: Improving One-handed Target Acquisition on Small Touch-screens. In *Proc. AVI '08*, pp. 146–153. ACM, 2008.

[9] A. Sears and B. Shneiderman. High Precision Touchscreens: Design Strategies and Comparisons with a Mouse. *Int. J. Man-Mach. Stud.*, 34(4):593–613, 1991.

[10] D. Vogel and P. Baudisch. Shift: A Technique for Operating Pen-based Interfaces Using Touch. In *Proc. CHI '07*, pp. 657–666. ACM, 2007.

[11] D. Wigdor, C. Forlines, P. Baudisch, J. Barnwell, and C. Shen. Lucid Touch: A See-through Mobile Device. In *Proc. UIST '07*, pp. 269–278. ACM, 2007.