Vol.2014-HCI-156 No.17
2014/1/16

大画面を備える携帯情報端末における片手ポインティング手法

大西 主紗[1,a)]　志築 文太郎[2]　田中 二郎[2]

概要：大型のタッチ画面を備える携帯情報端末上における片手操作手法を示す．本システムでは，ユーザはタッチ画面を持つ端末を片手にて把持し，その親指にてすべての領域に対するポインティングを行う．画面の全領域に対する操作を実現するために，本手法では画面端において反射を行うカーソル，またはドラッグ操作にて画面の指定した領域にタッチ点を転送する手法のどちらかを用いる．これらの手法は画面に強くタッチを行う Forcetouch によって起動するため，タッチ画面上における既存の操作手法と共存することが可能である．ポインティング性能を検証するため，既存手法との比較実験を行った結果，提案手法の性能が発揮される環境が発見された．

1. はじめに

大型のタッチ画面をもつ携帯情報端末（以降，大端末）が市場に登場し，ユーザに使用されるようになった．Karlson ら [2] によると，ユーザの大部分は片手操作，すなわち端末を把持した手の親指のみを用いた操作による携帯情報端末の操作を望んでいる．しかし，片手操作によって大端末を操作する場合，ユーザの指が届かない画面領域に対する操作が困難であるという問題が存在する [1].

この問題を解決するため本研究では 2 つのポインティング手法を実装した．それぞれのポインティング手法は Forcetouch と呼ばれる起動手法によって起動され，それに続けて行われる以下の操作手法からなる．

- 操作手法 1：ドラッグによる操作が可能な，画面の端において反射するカーソルを用いる操作手法（以降，Reflection）
- 操作手法 2：タッチイベントを指が届かない画面領域に転送させる操作手法（以降，TouchOver）

これらによりユーザは大端末を片手にて把持し，その親指にてすべての画面領域に対する選択操作を行うことができる．なお，Forcetouch は，Forcetap[3] を応用した操作であり，タッチ画面を強くタッチする操作である．この Forcetouch を起動手法とすることによって，提案ポインティング手法は，タッチ画面を備える端末にて従来から使用されてきたダブルタップやピンチなどのジェスチャ操作と共存することが可能である．さらに，提案手法の実装に要する入力デバイスは，シングルタッチの検出が可能なタッチ画面，および加速度センサである．このため，提案手法は多くの携帯情報端末にて動作することが可能である．

我々は，提案手法のプロトタイプを Android 端末上にて動作するアプリケーションとして実装し，本手法の精度と使用感を検証する実験を行った．本稿では，これらについて報告する．

[1] 筑波大学情報学群情報情報科学類
College of Information Science, School of Informatics, University of Tsukuba
[2] 筑波大学システム情報系
Faculty of Engineering, Information and Systems, University of Tsukuba
[a)] onishi@iplab.cs.tsukuba.ac.jp

2. 関連研究

本研究において提案するポインティング手法は，大端末を把持した手の親指が届かない画面領域のターゲットを間接的に選択する手法である．そこで本節では，携帯情報端末上におけるターゲットの直接選択及び間接選択に関する手法を述べる．

2.1 直接選択

指の届かない画面領域に存在するターゲットを指の届く範囲に移動させることにより直接選択を可能とする手法がこれまでに研究されてきた．LoopTouch[4] では，ユーザは，大端末の表面に人差し指を，裏面に親指をタッチした状態で，両指の相対位置が近づく方向にスワイプする．この操作により画面領域全体がループするため，ユーザはターゲットを指の届く範囲に移動させることが可能となる．この操作を検出するために，LoopTouch では裏面にタッチパネルを装着した大端末を用いる．Sliding Screen[5] では，

画面を広い面積でタッチ，もしくはベゼルスワイプ [6] を行った後のドラッグ操作によって，ユーザは画面領域全体を指の方向に移動させ指の届く位置にターゲットを移動させることが可能である．永田らは，ベゼルスワイプを行うことにより指の届かない画面領域にあるオブジェクトを指の届く領域に移動させる手法を提案している [7]．一方，本研究はこれらと異なり間接選択手法を用いており，また [7] とは異なりオブジェクト以外の任意の箇所をポイントすることも可能とする．

2.2 間接選択

ThumbSpace[8] は，ユーザのドラッグ操作によって画面領域の縮小画像が表示されたタッチパッドを任意の位置に生成する．ユーザはこのタッチパッドを操作することによって指の届かない位置のオブジェクトを選択することが可能である．ユーザは，タッチパッドをタッチすることによりオブジェクトにフォーカスを当て，指をドラッグすることによりフォーカスを別のオブジェクトに移し，指を離すことによりフォーカスされたオブジェクトを選択する．一方，我々の手法はオブジェクト以外の任意の箇所をポイントすることも可能である．

また，カーソルを用いて選択する手法も存在する．Shift[10] や MagStick[11] は“fat fingers”[9] を解決するためにカーソルを用いているのに対して，本研究の Reflection では指の届かない画面領域に存在するターゲットの選択を行うためにカーソルを用いる．CornerSpace[2] は，カーソルの出現位置を決定するためのウィジェットを使用するポインティング手法である．ベゼルスワイプによって画面を 4 領域に区切るウィジェットが生成され，そのいずれかの領域のタッチにより，その領域に対応した画面隅にカーソルを表示する．ユーザはこのカーソルをドラッグ操作によって移動させ，ターゲットを選択する．この手法ではターゲットの選択に 2 段階の操作を要するのに対して，本研究の 2 手法ではどちらも 1 段階の操作のみ要する．Extendible Cursor[5] は，広い面積でのタッチ，もしくはベゼルスワイプが行われたとき，ドラッグ操作によって移動させることのできるカーソルを画面に表示する．ユーザはこれを用いてターゲットの選択を行う．一方，本手法に用いるカーソルは CornerSpace や Extendible Cursor と異なり，画面端にて反射するためカーソルがターゲットを通り越した際も指を引き戻す必要がない．

3. 提案手法

本節では，提案操作手法である Reflection，TouchOver，及び，起動手法である Forcetouch について述べる．

3.1 Reflection

Reflection では画面端において反射するカーソルを表示する．このカーソルはユーザのドラッグ操作によって移動させることができる．カーソルが画面端において反射するため，カーソルがターゲットを通り越してしまった際もカーソルを引き戻さず，同じ方向の操作のみを用いて再度ターゲットを選択することが可能である．また，このカーソルの control-display（C-D）比については，大まかにターゲットの方向にカーソルを移動させたのちに細かい操作を用いてポインティングを行うことを可能とするために，反射後に減少させることとした．なお，現実装では反射前の C-D 比を CornerSpace[2] にならい 2 とし，反射後の C-D 比の値は実験的に 1 とした．

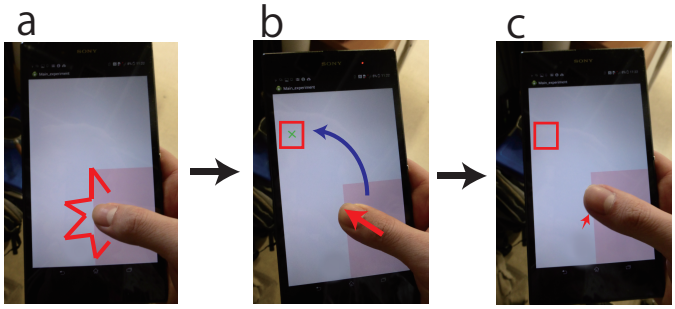

図 1 a : Forcetouch による Reflection の起動．b : ドラッグ操作によるカーソルの移動．c : 画面端におけるカーソルの反射

Reflection の使用方法を図 1 に示す．ユーザはまず，a のように画面に対して強いタッチを行う．その後画面をタッチした指をドラッグすることによって，b のようにカーソルを移動させることができる．また，カーソルが画面外へ移動するような操作を行った場合，カーソルは c のように画面端において反射する．

カーソルの位置の算出法を詳述する．ここで，図 2 に示すように，タッチ画面に最初に触れた時のタッチ点を $S(x_S, y_S)$，ドラッグ操作後のタッチ点を $S'(x_{S'}, y_{S'})$ とする．まず反射前の位置は点 $P(2 \times (x_{S'}\text{-}x_S) + x_S, 2 \times (y_{S'}\text{-}y_S) + y_S)$ である．また，画面の 4 辺 V_1〜V_4 と線分 SP が交わった場合，位置は点 P ではなく点 P' とする．このとき V と SP の交点を $G(x_G, y_G)$ とすると，P' $(x_{P'}, y_{P'})$ は以下となる．

V_1 または V_3 と SP が接触した場合

$(x_{P'}, y_{P'}) = (1 \times (2 \times (x_{S'}\text{-}x_S)\text{-}(x_G\text{-}x_S)) / 2 + x_G,$
$-1 \times (2 \times (y_{S'}\text{-}y_S)\text{-}(y_G\text{-}y_S)) / 2 + y_G)$

V_2 または V_4 と SP が接触した場合

$(x_{P'}, y_{P'}) = (-1 \times (2 \times (x_{S'}\text{-}x_S)\text{-}(x_G\text{-}x_S)) / 2 + x_G,$
$1 \times (2 \times (y_{S'}\text{-}y_S)\text{-}(y_G\text{-}y_S)) / y_G)$

3.2 TouchOver

TouchOver はタッチイベントを，入力された画面領域

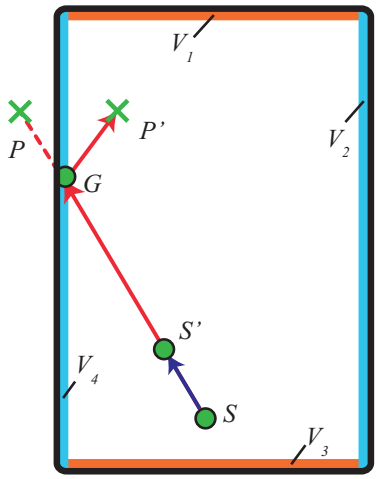

図 2 Reflection におけるタッチ点及びドラッグ操作後のカーソルの位置の関係

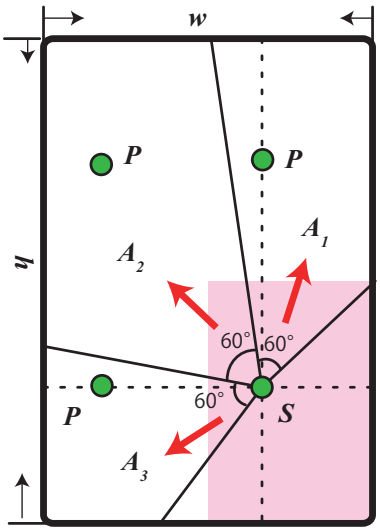

図 4 TouchOver におけるタッチ点及び転送後のタッチ点の位置の関係

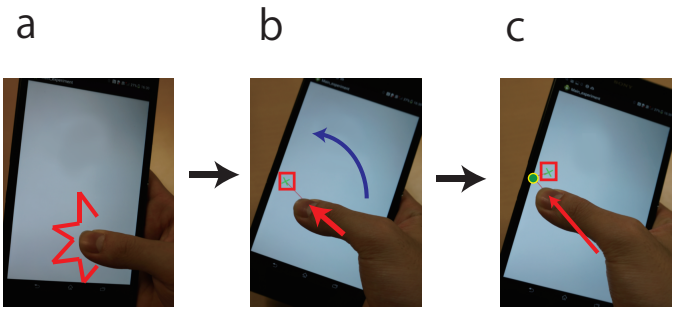

図 3 a : Forcetouch による起動．b : ドラッグ操作による転送先の決定．c : タッチアップによる TouchOver の解除

とは異なる画面領域に転送する．この工夫によって，指が届く画面領域における操作のみを用いて，離れた画面領域に存在するターゲットを選択することをユーザに可能とする．転送対象となる画面領域の選択にはドラッグ操作を用いる．ただし，このドラッグ操作には画面領域の選択に必要な最小限の指の移動のみを要する．結果として，本ポインティング手法に要する指の移動量はわずかであり，結果としてカーソルを用いる手法に比べて高速なポインティングが可能となり得る．

TouchOver の使用方法を図 3 に示す．ユーザはまず，a のように画面に対して強いタッチを行う．その後画面をタッチした指を左上方向にドラッグすることによって，b のようにタッチイベントを左上領域に転送する．このとき，転送先のフィードバックとしてカーソルが表示される．c : 目的の操作を終えたとき，指をタッチ画面から離す事によって TouchOver を終了させる．

転送されタッチイベントの位置の算出法を詳述する．本手法により転送されたタッチ点の座標 $P(x_P, y_P)$ は，ユーザのタッチ点を S (x_S, y_S)，画面の横方向の長さを w，縦方向の長さを h とした場合，図 4 に示すようにユーザのドラッグ方向に応じて以下のようになる．

A_1 方向にドラッグ操作を行った場合

$$(x_P, y_P) = (x_S, y_S - \alpha \times h)$$

A_2 方向にドラッグ操作を行った場合

$$(x_P, y_P) = (x_S - \alpha \times w, y_S - \alpha \times h)$$

A_3 方向にドラッグ操作を行った場合

$$(x_P, y_P) = (x_S - \alpha \times w, y_S)$$

α の値は，ユーザの直接タッチが必要な領域が最も狭くなるように設定する．上記の移動量にしたがった場合，A_2 方向に転送後のタッチ点の x_P 座標は $x_S - \alpha \times w$ となる．したがって，タッチ画面左端の領域をポインティングするためには，$x_S = \alpha \times w$ を満たす点をタッチする必要がある．この点が，タッチ点を移動させたときにポインティング可能な最も右端の x 座標となるとき，すなわち $x_S = w$ を満たす座標 $x_P = (1- \alpha) \times w = \alpha \times w$ を満たすとき，直接タッチの必要な領域の幅が最も小さくなる．また，高さについても同様である．したがって α= 0.5 と定めることとした．また，直接タッチが必要な領域を薄く表示しユーザにヒントを与えることとした．この領域を手元領域と呼ぶ．

3.3 Forcetouch

Forcetouch はタッチ画面を強くタッチする操作である．その検出には端末に組み込まれた加速度センサの値を取得し，ユーザが行ったタッチの強弱を識別する．同様に画面に対するタップの強弱を識別する手法である Forcetap[3] では，10 ms 間隔にて加速度センサの z 軸方向の絶対値を取得し，ユーザの指が端末の画面に接触している間に取得された値すべての合計が閾値を超えていた場合，そのタッ

プを強いタップであるとみなした．一方本手法においては，ユーザのタッチが行われた瞬間にそのタッチの強弱を識別する必要がある．先行研究は取得された加速度の推移とタッチイベントの関係の調査を行い，以下の知見を得た．

- 加速度のピークがタッチイベントの発生前に起こっている．
- ユーザがタップを行う際，指が画面に接触している時間が平均約 96 ms である．

そこで本手法においては，タッチイベントが発生する直前の加速度の値の合計を用いた．これはタッチイベントの直前に加速度のピークがあるため，Forcetap と同様にタッチ時の加速度の指標となる値が得られると考えたためである．また，タッチイベント開始直前 96 ms に取得された z 軸方向の加速度の絶対値の合計を判定に用いることとした．なお本手法における加速度の取得間隔を 5 ms とした．

[3] では，画面領域ごとにユーザのタッチによる加速度に差があると述べ，端末画面を縦 6 領域　横 4 領域の計 24 領域に分割し，強くタップされたことを識別するための加速度の閾値をそれぞれの領域に定めていた．本手法もそれにならい，画面を 24 領域に分割しそれぞれの領域に閾値を設定することとした．ただし，我々が実装を進めるにあたって，タッチ時の加速度にユーザごとの個人差が見られ，定数の閾値を用いた場合の識別制度が低かった．したがって本手法においては，予めキャリブレーションを行い，ユーザごとに各領域に対して閾値を設定することとした．

キャリブレーションの手順を以下に示す．キャリブレーションを行う際，画面を押し込む程度の強さ（strong 条件），通常のタッチを行う程度の強さ（normal 条件），画面に触れるか触れないか程度の強さ（weak 条件）の 3 種類のタッチ条件を設けた．ユーザはそれぞれのタッチ条件にて，画面に表示されたターゲットに対して 6 回ずつタッチを行う．このとき用いたターゲットを図 6 に示す．本手法は，ユーザがタッチする際に求められるタッチ条件を，ターゲットの大きさ及び色を用いて示した，すなわち図 6 のターゲットは左から，strong 条件のタッチを行う際のターゲット，normal 条件のタッチを行う際のターゲット，weak 条件のタッチを行う際のターゲットである．ターゲットを表示する領域の順番はランダムとしたが，タッチ条件はすべて strong，normal，weak の順番とした．

閾値を決定する際には strong 条件のタッチによる加速度及び weak 条件のタッチによる加速度の値のみを用いるが，ユーザが strong 条件のタッチ及び weak 条件のタッチのみを行った場合，2 種類のタッチにおける加速度の差が小さかった．そこでユーザに両条件を明確に使い分けてもらうため，タッチ条件に normal 条件を加えキャリブレーションを設計した，

こうして得られた strong 条件のタッチの加速度および weak 条件のタッチの加速度の分布がそれぞれ正規分布であるとし，図 5 に示される両分布の交点 F を算出した．しかし，この交点をそのまま閾値とした場合誤認識が見られたため，両分布から得られた標準偏差 σ_i，σ_i を用いて閾値 F_S，F_W を再設定し，誤認識を減らした．すなわち，あるタッチの加速度を f とした場合，そのタッチは以下のように識別される．

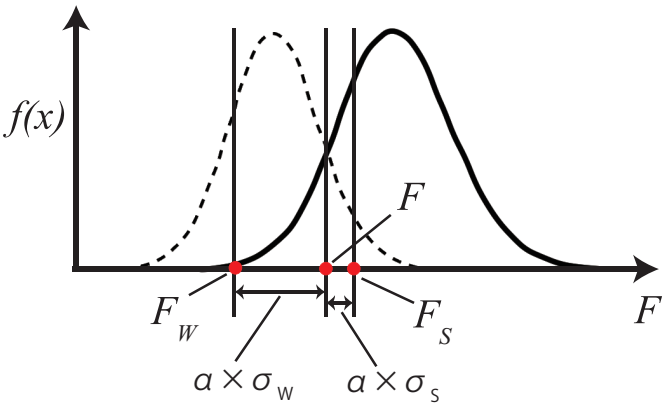

図 5　閾値 F_S 及び F_W の関係

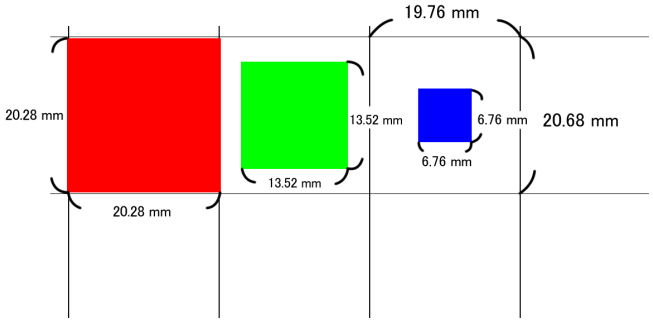

図 6　キャリブレーション時に使用したターゲットと分割した領域の大きさの比較

Forcetouch：$F_S = F + \beta \times \sigma_W < f$

通常のタッチ：$f < F$ - $\beta \times \sigma_S = F_W$

このとき上記のどちらにも当てはまらない場合には，そのタッチによる処理を行わないようにした．なお β の値は実験から 0.5 とした．

4. Forcetouch 予備実験

Forcetouch のキャリブレーション機能の性能評価を行うため，予備実験を行った．

4.1 実験設計

提案手法のプロトタイプを，Android 4.2.2 上で動作するアプリケーションとして実装した．また実行端末として Sony Xperia Z Ultra（Android 4.2.2，サイズ 179.4 × 92.2 × 6.5 mm，解像度 1080 × 1920 px）を用いた．被験者は 22 歳から 24 歳の大学生及び大学院生 3 名であり，全員右利きであった．被験者には机に利き腕の肘を付けた状

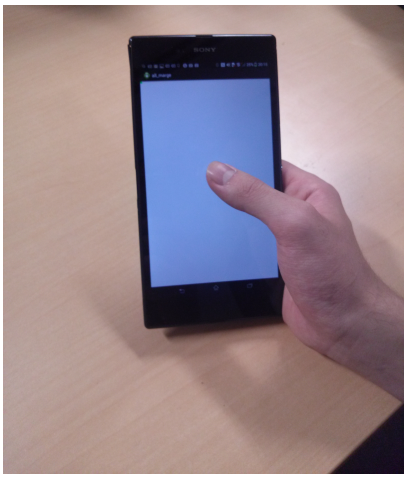

図 7　実験時の把持姿勢

0	1	2	3
4	5	6	7
8	9	10	11
12	13	14	15
16	17	18	19
20	21	22	23

図 8　閾値設定のための分割領域

態にて利き手にて図 7 のように大端末を把持し，その親指のみを用いて操作を行ってもらった．この時指の届かないターゲットを選択する際には，利き手のみを用いて端末を回転させ，ターゲットをタッチすることを許可した．この間ユーザは非利き手を膝の上におき続けた．被験者にはまず，図 8 に示す 24 領域に対してキャリブレーションを行ってもらった．

その後被験者には，図 8 の赤枠にて示される領域 5，6，7，9，10，11，13，14，15，17，18，19，21，22，23 に 1 度ずつ表示されるターゲットを strong 条件にて順番にタッチしてもらった．その後，同じ条件のターゲットを weak 条件にてタッチしてもらった．このとき指示した強さにてタッチを行うことができなかった場合，その強さにてタッチに成功するまで同じターゲットに対してタッチを行い続けてもらった．被験者一人当たりの所要時間は約 5 分間であった。

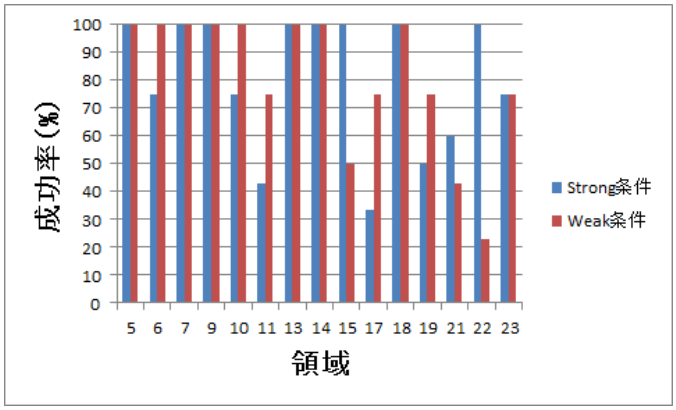

図 9　予備実験結果：識別成功率

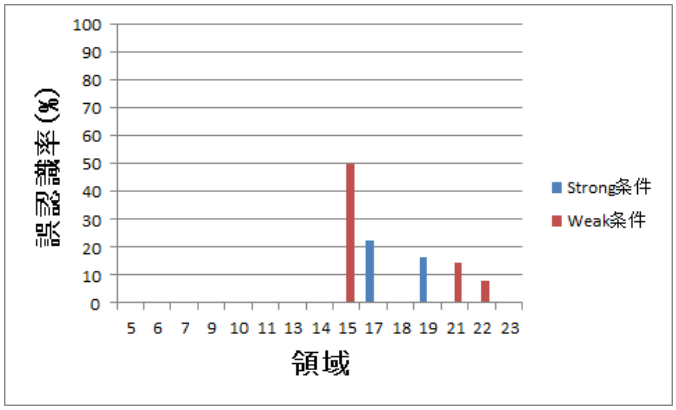

図 10　予備実験結果：誤識別率

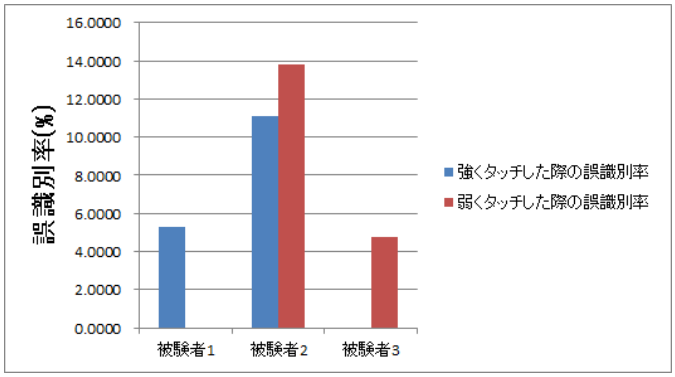

図 11　予備実験の結果：被験者ごとの誤識別率

4.2　結果

実験の結果を図 9，10，11 に示す．ここで示される成功率はターゲット数 / タッチ回数であり，誤識別率は誤認識された回数 / タッチを行った回数である．

強くタッチを行う試行に関しては領域 11，17 の成功率が特に低く，また領域 17 のエラー率が特に高かった．また，非常に弱くタッチを行う試行に関しては領域 22 の成功率が特に低く，また領域 15 の失敗率が非常に高かった．領域 15 に関して，キャリブレーションから得られた閾値を比較したところ，全領域における閾値 F_S の平均値は 14.50 であり，閾値 F_W の平均値は 8.52 であるのに対し，領域 15

における F_S の平均値は 9.39 であり，F_W の平均値は 5.29 であった．F_S と F_W の差は全領域平均が 6.98 であるのに対し領域 15 が 4.10 であることから，領域 15 が被験者が端末を把持する手の親指の付け根に近いため，タッチの際に力の使い分けが困難であるためと考えられる．

4.3 考察

強くタッチする試行に関して平均誤認識率は 5.46 %であった．また非常に弱くタッチする試行に関して平均誤認識率は 6.18 %であった．これらの識別精度と Forcetap の識別精度を比較するため，Forcetap の平均誤識別率 5.83 %を用いて 1 サンプル t 検定を行ったところ，両試行に関して有意差が見られなかった（p=.918 > .10）（p=.938 > .10）．したがって Forcetouch は Forcetap と同程度の識別精度を持つと言えるため，閾値の設定をキャリブレーションによって行うこととした．

5. 被験者実験

提案手法と，既存手法である直接タッチ及び CornerSpace との性能を比較するため被験者実験を行った．

5.1 実験設計

提案手法のプロトタイプを，Android 4.2.2 上で動作するアプリケーションとして実装した。実行端末として Sony Xperia Z Ultra（Android 4.2.2，サイズ 179.4 × 92.2 × 6.5 mm，解像度 1080 × 1920 px）を用いた．被験者は 20 歳から 24 歳の大学生及び大学院生 8 名であり，全員右利きであった．被験者には予備実験と同様の条件にて端末を操作してもらった．また，実験終了後，各被験者に実験についてのアンケートを行った．被験者には実験終了後に謝礼として報酬を渡した．被験者 1 人あたりの実験時間は約 1 時間であった．

本実験は，独立変数の 1 つとしてダミー条件の有無を儲けた．これは，ポインティング目標として表示されるターゲットの他に，ポインティングしてはいけないダミーターゲットを表示させるか否かという条件設定であり，実世界の，例えば誤って押してはいけないボタンや選択してはいけないアイコンを模したものである．

5.2 実験手順

被験者にはまず，予備実験同様に Forcetouch の閾値を決めるためのキャリブレーションを行ってもらい，その後画面上に表示されるターゲットのポインティングタスクを行ってもらった．このタスクにおいて被験者は，指示された手法条件に従いタッチ画面上のターゲットをポインティングする．この手法条件は、Touch 条件、CornerSpace 条件、Reflection 条件、TouchOver 条件の 4 種である。Touch 条件においては，被験者はタッチのみを用いてターゲットのポインティングを行う．その他の手法条件においては，被験者は指示された手法及びタッチを任意に使い分け，ターゲットのポインティングを行う．このとき選択するターゲットは Forcetouch の閾値設定に用いた 24 領域それぞれの中心に配置され，ランダムにひとつが目標ターゲットとして青く表示される．ダミーなし条件においてタッチ画面上に表示されるのは目標ターゲットのみであるが，ダミーあり条件においては残りのターゲット 23 個がすべて灰色に表示される．図にダミーなし条件，ダミーあり条件のそれぞれの様子を示す

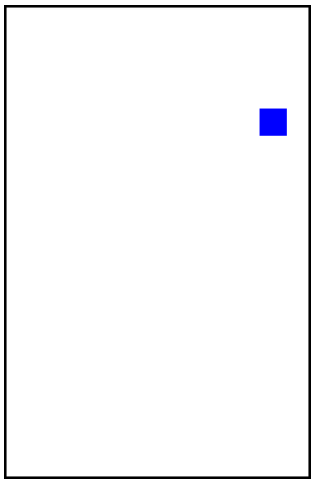

図 12 ダミーなし条件

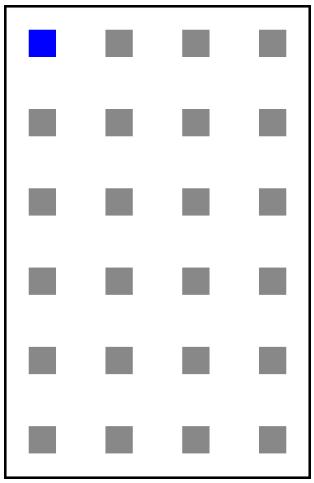

図 13 ダミーあり条件

タスクは被験者が待機画面に対してタッチを行い，その指が画面から離れたときに開始される．被験者はタッチ画面に表示される目標ターゲットに対してポインティングを行い，タスク中にそれぞれ 1 度ずつ表示されるターゲット 24 個をすべてをポインティングする．ただしダミーあり条件において，被験者がダミーターゲットをポインティングした場合，そのポインティングは致命的失敗として扱われ，新たな目標ターゲットが表示される．このタスク間に，被験者は任意に休憩を取ることができた．また 2 度の本番タスクを行う前に，被験者には本番と同じ条件で 1 タスク分の練習タスクを行ってもらった．

以上より本実験の独立変数は，4（手法条件），2（ダミー条件），3（練習タスク 1 回 + 本番タスク 2 回），24（ターゲット数）であり，ポインティング回数は独立変数の積である 4 × 2 × 3 × 24 = 576 回である．手法条件が指示される順番は被験者ごとにランダムであった．また，ダミー条件はすべてダミーあり条件，ダミーなし条件の順にて行った．

6. 結果

1 タスクの所要時間の平均値及び平均失敗率のグラフを，図 14，15 に示す．

分散分析の結果 TouchOver は他の手法条件と比べ平均所

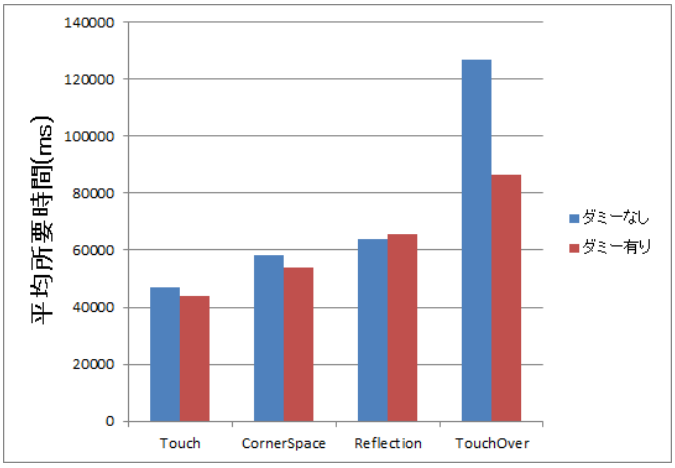

図 14 実験結果：所要時間

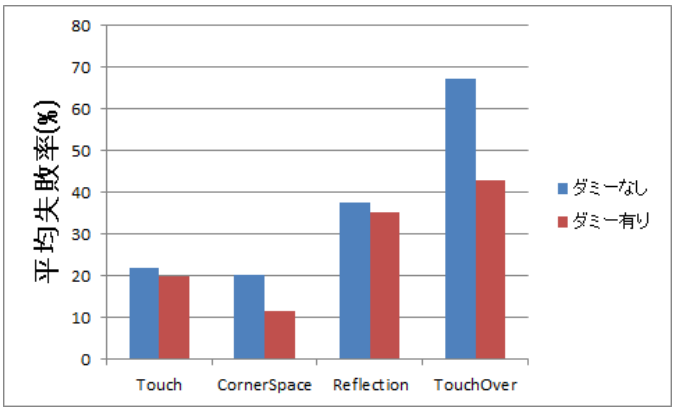

図 15 実験結果：失敗率

要時間が有意に長いことがわかった（$p = .000 < .05$）．また失敗率に関しても，TouchOver は他の手法条件に対し有意に失敗率が高かった（$p = .000 < .05$）．

両結果に対して多重検定を行い交互作用を調べたところダミー条件において交互作用が認められ，ダミーあり条件における TouchOver の所要時間はダミーなし条件に比べて有意に早く（$p = .000 < .05$），失敗率は有意に低かった（$p = .005 < .05$）．また，手法条件について交互作用が認められ，ダミーあり条件においては Reflection 及び TouchOver の失敗率に有意差が認められなかった（$p = .129 > .10$）．

7. 考察

ダミーあり条件にて TouchOver の失敗率が下がり，所要時間が減少している．これは，ダミーとして表示されたターゲットがタッチ点の移動先の指標として機能しているからと考えられる．したがってソフトウェアキーボードなど，十分な指標の存在する環境においては TouchOver は有用であると言える．

成績の悪かった Reflection 及び TouchOver について，実験後のアンケートに「Forcetouch が使いづらかった」とのコメントが多く見られた．そこで，タッチ条件の識別不能率のグラフを図 16 に示す．分散分析の結果，ダミー条件にかかわらない，手法条件間の平均値における有意傾向が見られた（$p = .091 < .10$）．これは，Reflection は任意のタッチ画面領域からタッチ画面全体に対してポインティングを行うことができるため，よりタッチ条件を使い分けやすい領域にて Forcetouch を行うことができたためと考えられる．

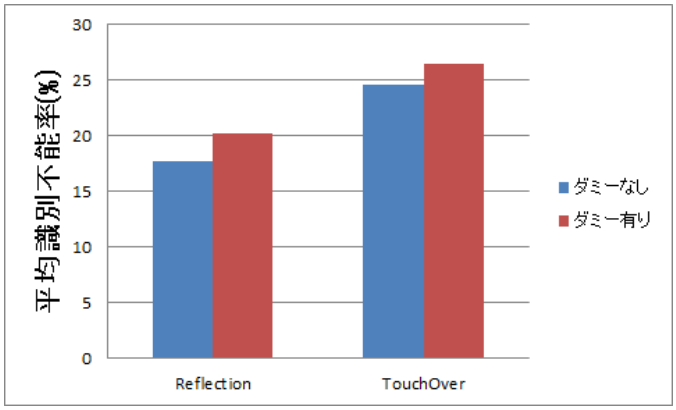

図 16 実験結果：識別不能率

また，識別不能だったタッチが仮に完全に識別できていたと仮定した場合の平均失敗率を図 17 に示す．分散分析を行い多重検定を用いて交互作用を調べたところ，ダミーなし条件において，各手法間に有意差が認められなかった（$p = .152 > .10$）．すなわち，全ての手法の精度がおおよそ等しかったといえる．

しかし図 18 に示すとおり，ダミーあり条件下での TouchOver はダミー選択率が非常に高い値となっている．実験より得られたデータより，このうち約 15%が，strong 条件の誤認識であることがわかった．また実験時の様子から，正しく起動を行ったにも関わらず誤ったターゲットを選択してい場面が多く見られた．これは現在，TouchOver によってタッチ点を移動する方向が，タッチ開始時点の座標及び，画面から指を離した際の座標の関係によって決まるため，指を離す際にタッチ点の移動方向が被験者の意志と関係ない方向に定まってしまうことがあるためである．

8. まとめと今後の課題

本研究では，大型のタッチ画面を備える携帯情報端末における片手操作手法の実現を目的とし，起動手法及び操作手法からなるポインティング手法を 2 種類提案した。この提案手法を Android アプリケーションとして実装を行い，その性能を評価するための被験者実験を行った．具体的には，既存の操作手法である直接タッチ及び先行研究である CornerSpace を比較対象とし，ポインティングタスクを行った。実験の結果，本実験に用いた起動手法に問題があることが分かった。しかし，起動手法が安定した場合既存

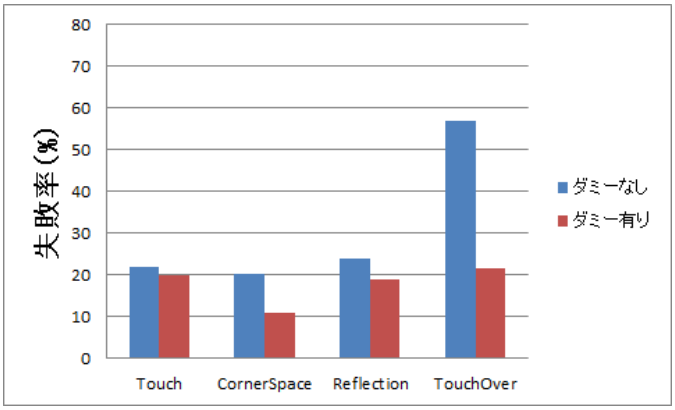

図 17 実験結果：理想起動状態での失敗率

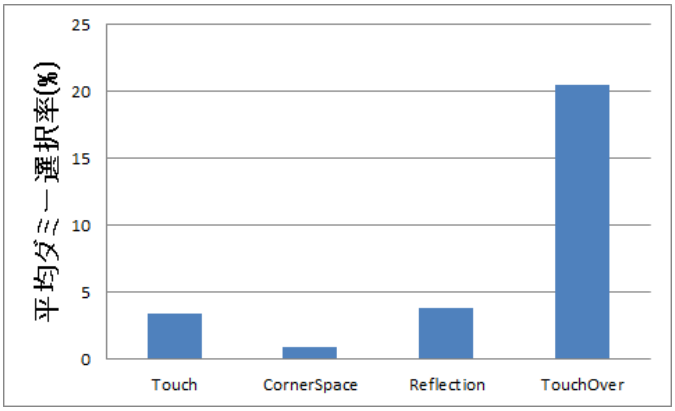

図 18 実験結果：ダミー選択率

手法と同程度の性能が出る可能性が示された。

操作手法の性能を評価するため，以下に Reflection，TouchOver，Forcetouch についてそれぞれの課題を示す。

8.1 Reflection

今後の課題は，より適した C-D 比の設定である。アンケートの中に，カーソルの C-D 比が足りないという意見が見られた。また本実験において，反射を用いた被験者はいなかったが，これはカーソルの移動量が少ないため，カーソルが画面外へ移動することがなかったためであると考えられる。したがって現在の実装よりも大きい C-D 比を設定することにより，より少ない指の移動でのポインティングを実現し，反射による新たなポインティングの提案を行う。

8.2 TouchOver

タッチ点の転送後の操作の実装を行う。現在はタッチイベントを転送した後，ユーザの指と連動したタッチアップイベントを発生させるのみであり，この仕様がポインティングタスクにおける失敗率の原因となっていた。したがって今後は，タッチ点の転送方向を決定する最低限のドラッグ操作を行った後の操作はすべて転送先に対するイベントとして処理するよう実装を変更する。また，新たな実装の性能を評価するため，ソフトウェアキーボードやウェブブラウザに TouchOver を実装し，より実世界に近い環境における性能の評価を行う。

8.3 Forcetouch

キャリブレーション機能の再設計を行う，本研究においては誤認識によって起こる致命的失敗を避けるため，タッチ条件が曖昧だと判断される加速度の値域を過度に大きく設定していた。その結果，Forcetouch による致命的失敗は少なかったが，失敗率は既存手法であるベゼルスワイプを大きく上回る結果となった。今後は閾値設定を見直すことにより，起動の安定化をはかる。

参考文献

[1] Karlson, A. K. and Bederson, B. B.: Understanding Single-Handed Mobile Device Interaction, Technical report (2006).

[2] Yu, N.-H., Huang, D.-Y., Hsu, J.-J. and Hung, Y.-P.: Rapid Selection of Hard-to-access Targets by Thumb on Mobile Touch-screens, in *Proceedings of the 15th International Conference on Human-computer Interaction with Mobile Devices and Services*, MobileHCI '13, pp. 400–403, New York, NY, USA (2013), ACM.

[3] Heo, S. and Lee, G.: Forcetap: Extending the Input Vocabulary of Mobile Touch Screens by Adding Tap Gestures, in *Proceedings of the 13th International Conference on Human Computer Interaction with Mobile Devices and Services*, MobileHCI '11, pp. 113–122, New York, NY, USA (2011), ACM.

[4] 土佐伸一郎，田中二郎：LoopTouch：画面ループを用いたモバイル端末片手操作手法，インタラクション 2013, 8 pages.

[5] Kim, S., Yu, J. and Lee, G.: Interaction Techniques for Unreachable Objects on the Touchscreen, in *Proceedings of the 24th Australian Computer-Human Interaction Conference*, OzCHI '12, pp. 295–298, New York, NY, USA (2012), ACM.

[6] Roth, V. and Turner, T.: Bezel Swipe: Conflict-free Scrolling and Multiple Selection on Mobile Touch Screen Devices, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pp. 1523–1526, New York, NY, USA (2009), ACM.

[7] 永田和信，村田和義，渋谷雄：モバイル機器における片手操作時の選択可能範囲外にあるオブジェクトの引き寄せ手法，シンポジウム モバイル'12, pp. 35–40, 6 pages.

[8] Karlson, A. K. and Bederson, B. B.: One-handed Touchscreen Input for Legacy Applications, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pp. 1399–1408, New York, NY, USA (2008), ACM.

[9] Siek, K. A., Rogers, Y. and Connelly, K. H.: Fat finger worries: How older and younger users physically interact with PDAs, in *Proceedings of the Tenth IFIP TC13 International Conference on Human-Computer Interaction*, pp. 267–280, Springer (2005).

[10] Vogel, D. and Baudisch, P.: Shift: A Technique for Operating Pen-based Interfaces Using Touch, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '07, pp. 657–666, New York, NY, USA (2007), ACM.

[11] Roudaut, A., Huot, S. and Lecolinet, E.: TapTap and

MagStick: Improving One-handed Target Acquisition on Small Touch-screens, in *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '08, pp. 146–153, New York, NY, USA (2008), ACM.