# マウスドライバの改良によるポインティング精度改善について

齊藤明紀[†] 西田知博[†]
辻野嘉宏[††] 都倉信樹[††]

グラフィカルユーザインタフェース（GUI）システムは現在では広く使われ，一般の利用者が使用するOSやアプリケーションはほぼすべてGUI化されている．そのため，マウスを長時間連続して使用するような利用形態が増えてきている．マウスなどでのポイントエラーは操作効率の低下や快適性の低下の大きな要因である．我々は，ボタン操作の副作用として発生するマウスの移動と，ボタン操作を行う指と移動操作を行う手首および腕の動きの不一致との2つのエラー原因に注目した．我々は，マウスが発生する信号の履歴を解析することで，これらのエラーが発生した際に利用者が意図した目標座標を推測するアルゴリズムを考案した．またXウィンドウシステムのマウスデバイスドライバ内にマウスの動きを補正する機能を追加し，効果を確認した．

# Refinement of Mouse Device-driver to Improve Pointing Accuracy

Akinori Saitoh,[†] Tomohiro Nishida,[†] Yoshihiro Tsujino[††]
and Nobuki Tokura[††]

In Graphical User Interface (GUI) System, a pointing error is one of the primary factor of inefficiency and unpleasantness. We focus on two types of pointing errors. One is an unintentional mouse motion affected by button operation side effect. The other is a synchronization error between a button operation and a move operation, which comes from a time lag between arm–wrist and finger movement. We propose an algorithm to guess the coordinates that the user want to point. It analyzes mouse motion and button operation information history. We also added some compensatory functions into the mouse device driver of X window system.

## 1. はじめに

現在広く使われているGUI（Graphical User Interface）システムでは，マウスに代表されるポインティグデバイスがユーザインタフェース上重要な要素となっている．ポインティングデバイスを用いた入力に関してはさまざまな研究が行われている[1),2)]．マウスは訓練なしですぐ使えるポインティング装置であるが意図した位置と入力されたポイント位置がずれることがある．特に，ポイント操作を連続して行う場合や，高精度のポインティングが要求される場合にポイントエラーが起こりやすい．

本研究ではマウス装置からの入力の処理方式によって，ポイントエラーを低減することを目指す．

すでに，ジッタの除去処理[3)]や，クリックやダブルクリックの際にマウスが少し移動してもそれを短距離ドラッグと誤認しないための処理は，多くのウィンドウシステムで一般的に実装されている．マウスのポイントエラーを軽減するソフトウェア手法としては，文献4), 5)で非リニアマウス（タイプ1, 2, 3）やブレーキマウスが提案されている．

本研究で解決を目指す問題点は，以下の2つである．

(1) 領域選択のドラッグ終了後ただちにマウスカーソルを他の位置（特に遠方）に動かすような場合，選択された領域が意図と異なることがある．

(2) マウスボタンを押すあるいは離す反動でマウスが動いてしまうことがある．ピクセル単位の高精度のポインティングの際には問題である．

本研究では，UNIXとXウィンドウシステム（以降**X**）を対象として実験を行った．

## 2. マウスの性質

まず，マウスの構造と通信方式，およびウィンドウシステム側での処理形態について調査した．

† 大阪大学情報処理教育センター
Education Center for Information Processing, Osaka University
†† 大阪大学大学院基礎工学研究科情報数理系専攻
Graduate School of Engineering Science, Osaka University

現在の主流はシリアルマウスで，同期あるいは非同期（調歩同期式）によるビット直列でマウスから計算機に信号が送られる．シリアルマウスの発生するデータは3バイトあるいは4バイトの長さのデータパケットで構成される．1つのパケットには位置差分情報 $(\Delta X, \Delta Y)$ とボタン状態 (押されている/いない) が含まれる．位置変化やボタンの状態の変化があるとパケットが発生される．また，変化が引き続いて起こっている間はパケットを発生し続ける．しかし，その頻度は通信速度の限界の数分の1程度のある一定時間（サンプリング間隔）ごとである．位置変化もボタンの状態変化もないときには何も送らない．

このサンプリング間隔はウィンドウシステムとマウスの機種の組合せごとに異なるが，よく用いられるのは10 ms～数10 msの範囲である．

ボタン操作と移動がサンプリング間隔以下の時間幅内に起きた場合，1つのデータパケットで両方が報告される．その結果，どちらが先に起こったのか，あるいは本当に移動しながらマウスボタンを操作したのかの区別ができない．

**X** で用いられるマウス操作を表すイベントは，以下の3種である．

- Motion Notify（以下 MN）：マウスの移動を示す．
- Button Press（以下 BP）：マウスボタンが押されたことを示す．
- Button Release（以下 BR）：マウスボタンが放されたことを示す．

**X** では，ボタンとポインタ移動が同時に起こったことに対応するイベントがなく，BR/BP と MN イベントが順に生成される．たとえばPC-UNIX用に移植されたX window system（X11R6）であるXFree86[6]のXサーバではMNイベントが先になる．

クリックに付随する意図しないマウス移動は，短距離のドラッグと紛らわしい．ソフトウェア手法で素早く短いドラッグをクリックと見なすようにすることは，すでに一般的に行われている．しかし，クリックの際のポイント位置は特に補正されていない．多くの場合クリックの対象物は十分に大きく，その必要性がないからである．

なお，本論文ではマウス装置が計算機に送る信号における距離単位として「カウント」を用いる．また，ウィンドウシステム側での距離単位は，画面のピクセル（ドット）数を用いる．

通常マウスの空間分解能は画面の解像度と一致する（1カウント = 1ピクセル）．しかし，**X** を含む多くのウィンドウシステムでは非線形（加速機能つき）のマウス・デバイスドライバが実装されている．この場合，マウスがある一定速度以上で動いている間は，カウントとピクセル比を変え（たとえば，1カウント = 3ピクセル），狭い作業領域でも全画面にアクセスできるようにしている．

## 3. 状況調査と実験

1章で述べた問題点がどのように起きているかを分析するため，観察と実験を行った．

### 3.1 選択エラー

マウスのドラッグによって領域選択を行うアプリケーションは多い．**図1** は **X** でのターミナルエミュレータである xterm で文字列の選択を行った例である．このように，反転表示によって選択範囲が示される．**図2** は，選択に失敗した場合の例である．ドラッグ終了地点が1行分上にずれている．

このような失敗は単なる利用者の操作ミスの場合もある．しかし，正しく図1の状況になっていることを目で確認してからマウスボタンを放した（つもり）にもかかわらず選択エラーが起きることがある．それも，**図3** のように，マウスの手ブレでは説明できないほど大きくずれることがある．


**図1** 文字列選択
Fig. 1 String copy & paste.


**図2** 文字列選択の失敗例1
Fig. 2 String selection failure 1.


**図3** 文字列選択の失敗例2
Fig. 3 String selection failure 2.

ここで，利用者がマウスボタンを離してからマウスを他の位置に向けて動かし始めたときに，実際には移動イベントが先に xterm に与えられているのではないかという仮説を立てた．

xterm（kterm☆）や emacs など **X** 上の多くのアプリケーションでは，表示された文字列の copy & paste を行うためにキーボード操作は必要ない．ドラッグで領域を選択すると自動的にその情報がペーストバッファにおかれる．そのため，領域選択が完了した時点ですぐさまペースト先へマウスを動かし始めてよい．そのため，前記の現象が起きやすいと考えられる．

### 3.2 マウス操作の実地調査

前節での考察を確認するため，マウスボタンの操作（プレスおよびリリース）の際のマウスの動きについて本学情報科学科の演習室での学生の **X** の操作記録[7]の分析を行った．

分析に際しては，目的が補正アルゴリズムの開発であるので，ボタン操作起因の動きであるかどうかが疑わしいものは排除し，確実性の高いものを抽出することを目的とした．すなわち，マウスを動かしながらのボタン操作を意図して行うこともあるので，十分長いマウス静止の後のボタン操作のみを抽出した．

分析したものは，**X** のサーバがウィンドウマネージャに対して送ったイベントである．マウスに関連する X イベントは，マウスカーソルの座標とタイムスタンプ（ms 単位）を持つ．

**X** ではウィンドウ管理機能が内蔵されておらず，ウィンドウマネージャと呼ばれる外部の特殊なクライアントアプリケーションが担っている．ウィンドウマネージャでは利用者のマウスやキーボードの操作のほか，マウスカーソルのウィンドウ境界の横断やウィンドウの生成/移動/消滅など多くの情報を X イベントとして受け取っている．

今回分析したデータを取得した環境（SONY NEWS/NEWS OS4.2.1a+）では，マウスのサンプリング間隔は 30 ms であった．そこで，MN イベントが 100 ms 以上発生しないということはマウスがいったん静止したと見なしてよいと考えた．

また，マウスを移動させた後，それに対するマウスカーソルの停止した位置を目で確認して意図した位置であると判断し，さらに指を動かしてマウスボタンを押す（あるいは放す）には，おおむね 300 ms 程度を要すると考えられる[8]．

そこで 3 倍の安全率を見込んで，静止した状態から 100 ms 以内に BP あるいは BR イベントが発生した場合は，マウスが静止した状態でボタン操作を行おうと利用者が意図した場合であると判断することにした．このような場合のうち，静止したときの座標とボタンイベントの発生した座標が異なるものをポイントエラーと推測した．

表 1 操作記録の分析結果
Table 1 Result of mouse motion analysis.

| イベント種別 | 総数 | 位置ずれ発生回数 | 位置ずれ平均距離 |
|---|---|---|---|
| Button Press | 5208 | 584 | 2.5 |
| Button Relase | 2760 | 332 | 3.3 |

分析対象はのべ 179 人の学生がログインしてからログアウトするまでの操作記録であり，イベント総数は 4464863 である．

分析の結果，ボタン操作にともなうポインタの不審な移動は少なくとも 5～10%程度の割合で発生していることが分かった．分析結果は**表 1** のとおりである．

この基準で抽出したポイントエラーが疑われる記録を調べたところ，イベント列としての発生形態は以下の 2 種であることが分かった．

- BP あるいは BR の後にも MN イベントが引き続き発生する．
- BP あるいは BR の直後しばらく（100～数 100 ms）はマウス移動が起こらない．

後者の場合，MN イベントで報告されるマウスの移動量（$\sqrt{\Delta x^2 + \Delta y^2}$）はほとんどの場合（87%）は 1 である．

このことから，前者は「ボタン操作終了後マウスを次の目標位置に動かす」という意図の動作において，ボタン操作（指の動き）がマウス移動（手首や腕の動き）に対して遅れたものと推測できる．

一方後者は，マウスを静止させた状態でボタン操作だけを行う意図の動作において，ボタン操作の振動等でマウスボールに動きが生じた「手ぶれ」であると考えられる．

### 3.3 実　　験

仮説をより詳細に検証するために実験を行った（**図 4**）．

実験のタスクは，以下のようなものである．

- 画面に課題ウィンドウと目的ウィンドウの 2 つのウィンドウが表示される．
- 課題ウィンドウに表示された文字列をマウスドラッグで選択し，目的ウィンドウでペースト操作を行う．

目的ウィンドウを課題ウィンドウを基準にして上下

☆ 漢字対応の xterm.

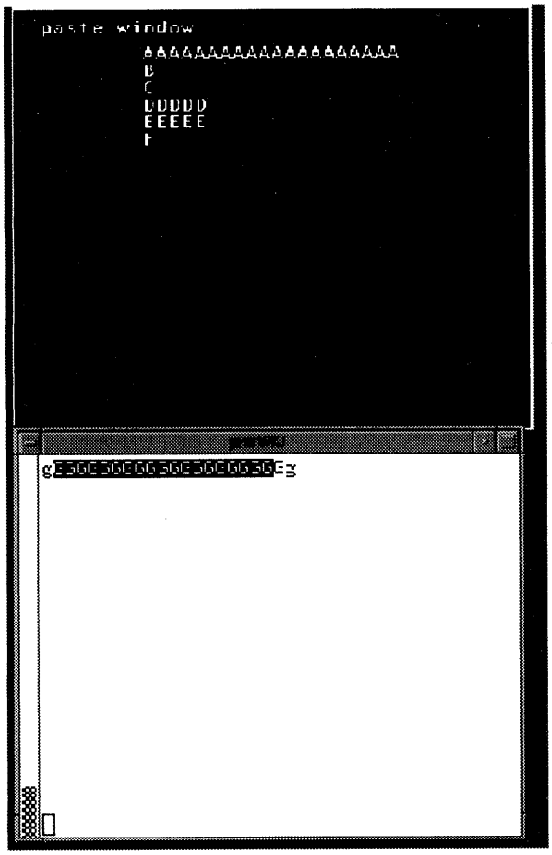



図4 実験中の画面
Fig. 4 Screen of evaluating system.

左右の4つの方向に配置した場合のそれぞれについて上記タスクを被験者が行う．課題文字列長は1，5，20文字の3種で，課題ウィンドウ中の4隅と中央のいずれかに表示される．表示位置と文字列長の出現順序はランダムである．

被験者数は11人で，1人あたりのタスク回数は300である．

この実験では，文字単位の選択を行うため，正しいドラッグ開始位置とドラッグ終了位置は点ではなく使用するフォントでの文字サイズ（7×14ピクセル）の大きさの矩形領域となる．ペースト操作は，現実の状況に合わせて目的ウィンドウ内であればどこで行ってもよいものとした．

実験は，タスクの成功失敗以外に利用者の操作記録を X のイベントレベルですべて記録した．画面に表示されるウィンドウはイベント記録機能を追加した kterm であり，copy&paste の操作および応答様式は実利用環境とまったく同じとなる．

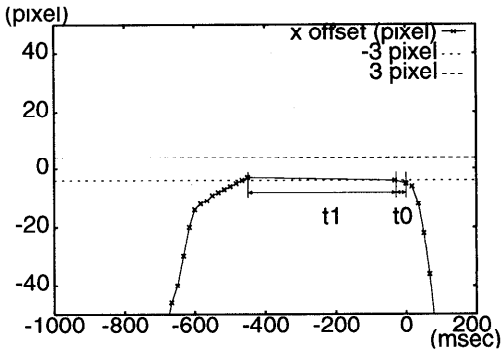



図5 エラーの状況例
Fig. 5 A situation of fallout error.

### 3.4 実験結果

選択操作の誤り率は，13.1%であった．これは，ドラッグ開始位置が文字列先頭から外れていたものも含んだ誤り率である．誤り事例のうちドラッグ終了位置のずれによるものは約半数（6.9%）であった．その多く（4.5%）は，いったん正しい位置でマウスカーソルが静止した後でマウスが動き始め，正しい目的領域を外れてから BR イベントが起きているものである．

仮説で考えた種類のポインティングエラーの発生率は4.5%であるが，その中には動き始めの MN と BR が同じタイムスタンプ値を持つものと，そうでないものがある．後者では静止後1〜3回程度の MN イベントが発生した後で BR が発生している．

前者は，2章で考察したように，ボタンを離す操作とその後のマウス移動がマウスの時間軸分解能以下の間隔で起こった可能性が疑われる．

**図5** は後者の状況を表した図である．横軸が時間，縦軸がその時点でのマウスの $x$ 座標として，各 MN，BR イベントの起こった時刻とそのときの $x$ 座標値を×でプロットしている．分かりやすさのため，横（時間）軸では BR が起こった時刻を0としている．縦軸は文字の中心を0としている．実験で使用したフォントの横幅は7ピクセルであるから，$x$ 軸の値が ±3 以下（点線の間）ならば文字列の最後の文字がポイントされていることになる．

この例では，ドラッグは左から右方向に行われ，正しい位置にマウスが移動したあと，約 400 ms のあいだ（t1）マウスが静止している．そしてその後マウスは左に移動し始め急激に加速して左に動いている．BR は動き始めの MN の後 t0（約 50 ms）後に発生しており，このときマウスカーソルは目標とする文字の左隣の文字の上にある．

このことから，利用者がマウスボタンを離してから

マウスを他の位置に向けて動かし始めているつもりだが，実際には指の応答が遅れ，マウスボタンを放し終わらないうちに腕が動き始めていることが推測できる．またこれは，マウスボタンの遊びが0.5～1 mm程度であるのに対し，マウスの移動検出機構の遊びは数10分の1 mmであることにも影響を受けている可能性がある．ただ，マウスボタンの操作にクリック感を持たせることは人間工学的に必要と思われ，ボタンの遊びのないマウスで解決をはかるのは現実的ではない．

## 4. 補正手法

前章で示したエラーの補正のために必要なことは以下の3点である．

- ボタン操作イベントごとに，補正を行うべきかそうでないかを識別する手法を考える．
- 利用者が意図したポイント位置を推定する．
- ポイント点補正の実装手法を考案する．

**X**では，マウスを動かしながらボタン操作を行うことを要求するアプリケーションはほとんどない．よって，移動中のボタン操作は補正の候補である．ただし，ウィンドウのフォーカス操作やウィンドウ移動など許されるポイント点の範囲が大きなときにはマウスを動かしたままでボタン操作を行うことはあるので，すべてが補正の対象ではない．

前章で述べた調査・実験の結果より，停止していたマウスが動き始めた直後にあるいは動き始めと同時に行われたボタン操作は，利用者が静止中に行うことを意図したものだと考えられる．よって，静止していたマウスが動き始めたあとある一定限度時間（数10～300 ms程度）以内に起こったボタン操作イベントを補正の対象とすればよい．

**図6**は，オリジナルの**X**サーバでのマウスデータ処理の例である．左半分がマウスの生成したデータ，右半分がそれに対応して生成される**X**のイベントである．この例は，マウスをドラッグして座標(100,100)でボタンを離してから上方へマウスを動かそうとしたつもりが，ボタンを離す動作が腕に対して遅れてしまった場合である．マウスデータ3回分の合計である(1,−13)だけ意図する座標とBRイベントが持つ座標がずれている．

| time | data from mouse | | | generated events | | |
|---|---|---|---|---|---|---|
| (msec) | $\Delta x$ | $\Delta y$ | button | type | $x$ | $y$ |
| 0 | 10 | −5 | ON | MN | 100 | 100 |
| ↕150 ms の静止 | | | | | | |
| 150 | 1 | −1 | ON | MN | 101 | 99 |
| 180 | 0 | −2 | ON | MN | 101 | 97 |
| 210 | 0 | −10 | OFF | MN | 101 | 87 |
| 210 | | | | BR | 101 | 87 |

**図6** マウスからのデータの処理例
Fig. 6 Mouse data processing.

いったんマウスカーソルを静止させた後で動かし始め，動き始めた直後にマウスボタンを放す（あるいは押す）という操作は**X**では要求されない．人間が意図的に特定の点をポイントするためにそのような操作を行った場合には，マウスが移動し始めてからボタンが操作されるまでの時間は数100 ms程度のある程度長い時間になるはずである．そこで，利用者が意図したポイント位置は，ボタン操作に先立つ静止位置であると推測できる．

図6の例では，何らかの修正を加えてBRイベントが座標(100,100)を持って生成されるようにすればよい．

ポイント点の補正手法は，2種類を実装し，比較した．以降2つの手法で，マウスが静止したと判断する時間の長さの閾値を$t_{stop}$とする．また，「マウスを動かし始めた直後」と見なす時間幅を，$t_{start}$とする．

### 4.1 手法1

**図7**の中央が図6と同じ状況での手法1 ($t_{stop} = t_{start} = 100\,\mathrm{ms}$)による補正例である．**図8**は，補正対象外の入力（この例ではボタンを押したままでの移動）の処理例である．

手法1では，マウスが停止状態から動き始めたときにマウス移動データを蓄積して，**X**アプリケーションに渡さない（図7の*2～*3）．ある時間幅$t_{start}$以内にボタン操作が行われたら，蓄積した移動情報をボタン操作の次に出力する（図7の*4～*6）．$t_{start}$以内にボタン操作が行われなかった場合には，蓄積した情報を出力する（図8の*5～*7）．

この手法では時間幅$t_{start}$のタイムアウト処理のために，タイマを使用する必要がある．調査した限りではUNIX用の**X**サーバではsetitimerやualarmなどのタイマサービスは使っておらず，本手法は問題なく実装することができる．

### 4.2 手法2

図7の右側が手法2 ($t_{stop} = t_{start} = 100\,\mathrm{ms}$)による補正例である．

移動データはそのまま移動イベントを生成するが，同時にタイムスタンプとともに記録する．ボタン操作データをマウスから受信したら，推測した正しいポイント位置にマウスカーソルを動かす移動イベントを生成してから（*4）ボタンイベントを生成する（*5）．その後に，生成した移動をうち消す移動（*6）を生成する．*4の移動量は，*2と*3の移動量の和の符合を反転したものである．

| seq no. | time (msec) | data from mouse | | | generated Event by method 1 | | | generated Event by method 2 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $\Delta x$ | $\Delta y$ | button | type | $x$ | $y$ | type | $x$ | $y$ |
| *1 | 0 | 10 | −5 | ON | MN | 100 | 100 | MN | 100 | 100 |
| ↕ 150 ms の静止 | | | | | | | | | | |
| *2 | 150 | 1 | −1 | ON | | | | MN | 101 | 99 |
| *3 | 180 | 0 | −2 | ON | | | | MN | 101 | 97 |
| *4 | 210 | 0 | −10 | OFF | BR | 100 | 100 | MN | 100 | 100 |
| *5 | 210 | | | | MN | 101 | 99 | BR | 100 | 100 |
| *6 | 210 | | | | MN | 101 | 97 | MN | 101 | 97 |
| *7 | 210 | | | | MN | 101 | 87 | MN | 101 | 87 |

図7 手法1と2による補正例
Fig. 7 Mouse data processing by method 1 and 2.

| seq no. | time | data from mouse | | | generated Event by method 1 | | | generated Event by method 2 | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | $\Delta x$ | $\Delta y$ | button | type | $x$ | $y$ | type | $x$ | $y$ |
| *1 | 0 | 10 | −5 | ON | MN | 100 | 100 | MN | 100 | 100 |
| ↕ 150 ms の静止 | | | | | | | | | | |
| *2 | 150 | 1 | −1 | ON | | | | MN | 101 | 99 |
| *3 | 190 | 0 | −2 | ON | | | | MN | 101 | 97 |
| *4 | 230 | 0 | −10 | ON | | | | MN | 101 | 87 |
| *5 | 250 | | | | MN | 101 | 99 | | | |
| *6 | 250 | | | | MN | 101 | 97 | | | |
| *7 | 250 | | | | MN | 101 | 87 | | | |
| *8 | 260 | 0 | −20 | ON | MN | 101 | 67 | MN | 101 | 67 |

図8 補正対象外のデータの処理例
Fig. 8 Mouse data processing example, not the case to fix.

ただし，ボタン操作がマウスが動き出してから $t_{start}$ 以上後になって起こったときには，意図的にマウスを動かしながらボタンを操作したものと見なして補正は行わない．図8に見られるように，手法2では補正対象外の入力に対しては内容，時刻ともにまったく加工しない．

## 5. 実装と検証

上で提案した手法はどちらもC言語で100から200行程度の処理を **X** サーバに追加するだけでよい．たとえば，手法2を実装したXFree86のXサーバでは，修正したファイルは4つであった．

どちらの手法でも，アプリケーションが受け取るBRおよびBPイベントが持つ座標値は同じとなる．

### 5.1 検証実験1：文字列選択エラーへの効果

3.3節と同じ実験を手法1を組み込んだ **X** サーバのもとで行った．$t_{stop} = t_{start} = 100\,\text{ms}$ として実験した結果を**表2**に示す．エラー率は10.6%であり，ボタンを放す操作を原因とするエラー率は6.9%から3.9%へとほぼ半減した．

### 5.2 検証実験2：補正のマウスの手ぶれへの効果

マウスの手ぶれに対する効果を調べるための実験を行った．**図9**が実験システムのスクリーンである．画面に表示される一方のターゲットをドラッグして他方に重ねる．ボタンプレス/リリースともに「1ドットもずれないように」と被験者には指示した．

**表2** 文字列選択のエラー率
Table 2 Error rate in string selection.

| | 補正なし | 補正あり |
|---|---|---|
| 全体 | 13.2% | 10.6% |
| ボタンを放す操作の誤り | 6.9% | 3.9% |
| それ以外 | 6.3% | 6.7% |

この実験ではドラッグ操作を1人あたり100回ずつ行う．その内訳は，手法2での補正を有効化したドラッグが50回，無効化したドラッグが50回である．有効/無効の切替えは被験者には知らせずにランダムに行う．この切替えは実験者にも分からない．

この実験で用いたのは200 count/インチのPS/2マウスであり，ディスプレイは1024×768ドットの17インチCRT，実験のためのウィンドウのサイズは640×480である．また実験での補正パラメータは $t_{stop} = 100\,\text{ms}$, $t_{start} = 130\,\text{ms}$ とした．

マウスボタンを押したときのカーソル位置と始点ターゲット，あるいは離した時点でのカーソル位置と終点ターゲットが1ドットでもずれていたものを，それぞれ独立した失敗としてカウントした．また，ターゲットから大きく（ターゲットサイズの2倍以上）離れた位置でのマウス操作は，人間の勘違い操作と見なして統計から除外した，

被験者は情報科学科の学生，大学院生，および教官

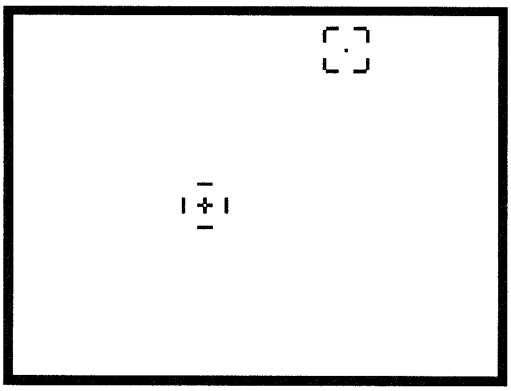


図9 精密ポインティング実験の画面
Fig. 9 Screen of precision pointing evaluator.

表3 精密ポインティングのエラー率
Table 3 Error rate in precision pointing.

Press 位置誤り率

| | 合計 | フォールアウト | その他 |
|---|---|---|---|
| 補正なし | 9.1% | 6.5% | 2.5% |
| 補正あり | 4.5% | 2.5% | 2.0% |

Release 位置誤り率

| | 合計 | フォールアウト | その他 |
|---|---|---|---|
| 補正なし | 15.8% | 15.4% | 0.4% |
| 補正あり | 0.4% | 0.2% | 0.2% |

合計10人である.

**表3**が補正の有無によるエラー率の比較である.エラーの内訳は,目標点にマウスカーソルがいったん到達しながらその後移動して誤った位置でボタン操作が行われたフォールアウトエラー(fallout error)と,それ以外である.

マウスボタンを押す位置の誤りは半減し,離す位置の誤りはほぼなくなっている(0.4%は10人×50回中2回だけのエラーを意味する).

### 5.3 その他の実験結果

手法1を組み込んだ**X**サーバにに関して,$t_{start}$を100 ms, 200 ms, 300 msと変化させて操作感について被験者の意見を求めたところ,200 msや300 msではマウスカーソルの追従性に違和感があり,特に300 msでは円滑な操作が行えないとの感想が得られた.100 msの場合は多くの被験者は無修正のものと同じ使用感であると答えたが,一部にマウスカーソルの応答性の低下に気づいた者もいた.

手法2に関しては,マウスカーソルの応答速度の問題はないが,いったん目標を外れたマウスカーソルがボタン操作時に戻るため,図3のように誤った領域を選択した状態が一瞬表示されてしまうことがある.

また,精密ポインティング実験でのフォールアウトエラーを解析したところ興味深いデータが得られた.目標を外れてからボタンが操作されるまでの時間間隔の分布は,ボタンを押す場合と離す場合でかなり異なる.ボタンを離す場合は,ほとんどが150 ms以下である.このため$t_{start}$ = 130 msとした補正でほとんどを救うことができたと考えられる.一方,ボタンを押す場合は分布がなだらかで0～250 ms程度まで広がっている.これは,マウスボタンのスイッチの遊び量が押す場合と離す場合で異なるか,ボタンに組み込まれたスプリングが指を離す動作を高速化しているからだと推測される.

このことから,$t_{start}$をたとえば250 msにまで増やすか,あるいはボタンを押す場合と離す場合で異なる$t_{start}$を用いる(たとえば150 msと250 ms)ように手法2を改良すれば,押す場合の精密ポインティングのエラー率も離す場合と同等に改善すると期待される.ただし,手法1では$t_{start}$を増やすと応答性が悪化するため,このアイディアは適用できない.

手法1では時間$t_{start}$のタイマを使用する必要があるため手法2よりは負荷が重いと考えられる.しかし,手法1, 2とも,これによる**X**サーバの負荷の増大は検出限界以下であった.

## 6. 考　　察

### 6.1 補正アルゴリズムの改良

手法2を組み込んだXFree86を実際に日常的に使用したところ,時に目だった副作用は感じられなかった.ただし,注目するウインドウを隠している重なったウィンドウをどける操作など目標位置の精度を必要としない場合には,人間は時として移動先の位置を目で確認せずにマウスボタンから指を離すようである.$t_{start}$が長すぎた(たとえば500 ms)場合には補正が行われて,移動の途中にマウスを持ち替えた位置がBR位置とされてしまうことがある.そこで,$t_{start}$の実用的な最大値は100～200 msと考えられる.

今回は停止した場所を利用者が意図するポイント位置と推測したが,マウスを短時間静止させたつもりでも実際は数カウントの移動が発生し続けている場合がある.このような場合には今回の手法は適用できない.

単純にマウスの静止を検出するのではなく,静止とほぼ等しい低速度の移動や,マウスの移動方向の変化点あるいは,移動速度が極小となる時点を検出するなどの手法を追加すれば,さらに補正範囲を広げられる可能性がある.

**X**などに実装されている通常のマウスドライバでは,ポイントエラーを低減するためにはかなり慎重か

つ微妙な操作を必要とする．本論文での手法は，マウス操作が多少荒くてもよいので，本手法を適用したマウスに習熟すれば，ポインティング効率（所要時間）がある程度向上（短縮）すると考えられる．

### 6.2 他方式との比較

マウスなどのポインティング装置に関するソフトウェア手法は多く提案しされている．

- マウスの必要とする作業領域の縮小
- ポイント時間の短縮
- ポイントエラーの低減

といったマウスの操作性のうちどれが改善されるかは手法によって異なる．

ここで，マウスの移動量（カウント）とマウスカーソルの移動量（ドット）の比を，$R$ で表すとする．

マウスの動きがある限度以下なら $R = 1$，以上なら $R = R_A > 1$（たとえば $R_A = 2$）に切り替える非リニアマウスは，より狭い作業領域でも他の特性を悪化させることなく操作可能であることが知られている．また，**X** や Microsoft Windows にも組み込まれている．

精密なポインティング時に $R$ を 1 未満とするような手法を用いると，ドット単位の細かいポインティング操作に対しフォールアウトエラー率低減の効果があることが知られている（ノンリニアマウス[4]，ブレーキマウス[5]）．

用いたマウスの分解能が異なるため単純な比較はできないが，フォールアウトエラーの改善率を文献 5) と比べると，本論文で提案する手法が勝っている．

本論文で提案する手法の特徴は，ノンリニアマウスやブレーキマウスとの共存が可能なことである．先に延べた実装例では **X** のマウス加速と呼ばれるノンリニアマウス機能を取り除くことなく補正手法を追加している．

ただし，手法 2 はノンリニアマウスの処理の結果に対して適用する必要がある．手法 1 はそのような制約はなく，たとえばマウスに内蔵されるワンチップマイコンに組み込むことも可能である．

最近マウスの分解能は高度化の傾向があり，高精度のポインティングの効率は悪化している．たとえば，640×400 ドットのスクリーンと分解能 0.25 mm のマウスで実験を行った文献 5) ではマウスカーソルが目標をとらえてから，位置確定操作を行ってボタン操作を行うまでの時間は平均 1.5 秒程度となっているが，本研究で行った検証実験 2 では，マウスカーソルがターゲットの近傍（8 ドット以内）に到達してからボタン操作が行われるまでの時間は，平均約 2 秒であった．また，手法 2 を適用した場合としない場合では変わらないことも分かった．

ポインティングの所要時間を短縮するには，ブレーキマウスなどの手法と本手法を組み合わせればよいのではないかと考えられる．

### 6.3 マウス以外の装置

今後の課題としては，マウス以外の装置での入力補正手法の研究がある．マウスと同様な特性を持つポインティングデバイスでは本手法はそのまま適用可能と考えられる．

マウスとは異なる特性を持つ他のポインティング装置の場合，たとえば，タッチパッドや小径のトラックボールなどでは，精密な位置調整のあと指をパッドやボールから離す際に，意図しない動き情報が入力されることがある．これには本論文で提案する手法はそのままは適用できない．この場合も，「装置からのデータの過去の履歴に基づいて移動情報の補正を行う」という本論文の手法の基本的枠組みを適用して補正することは可能であると考えられる．

## 7. むすび

マウスにおけるポインティング操作の際に起こるエラーを取り上げ，指と腕の動作の同期ずれによるエラーが起きていることを示した．また，ボタンにかける力がボールに伝わったために起こるぶれについても調べた．

さらに，これらによるエラーを打ち消す手法を考案し，**X** サーバに組み込むことでポインティング操作のエラー率が低下することを示した．本稿では特に **X** でのマウスを取り上げたが，マウスを用いる他のウィンドウシステムにも本手法は適用可能と考えられる．

**謝辞** 快く被験者を引き受けていただいた都倉研究室諸氏に感謝いたします．

## 参考文献

1) Card, S.K., English, W.K. and Burr, B.J.: Evaluation of mouse, rate-controlled isometric joystick, step keys and textkeys for text selection of a CRT, *Ergonomics*, Vol.21, No.8, pp.601–613 (1978).
2) Whitefield, D., Bell, R.G. and Bird, J.M.: Some comparisons of on-display and off-display touch input devices for interaction with computer generated displays, *Ergonomics*, Vol.26, No.11, pp.1033–1053 (1983).
3) ms – Sun mouse and STREAMS module, SUN OS 4.1 Devices And Network Interfaces manual (1989).

4) 西中ほか：ソフトウェア手法によるポイント効率改善について，電子情報通信学会論文誌（D），J70-D, 12, pp.2402–2409 (1987).
5) 西中ほか：座標入力のためのポイント手法について，電子情報通信学会論文誌（D），J71-D, 12, pp.2604–1612 (1988).
6) The XFree86 Project, Inc.: What is XFree86, http://www.xfree86.org/.
7) 西田ほか：GUIにおける実操作履歴の取得とその意図分析，情報処理学会ヒューマンインターフェース研究会報告，66-2, pp.7–14 (1996).
8) Card, S.K., Moran, T.P. and Newell, A.: *The Psychology of Human Computer Interaction*, Lawrence Erlbaum Associates (1983).

(平成10年6月3日受付)
(平成10年9月7日採録)

**齊藤 明紀**（正会員）

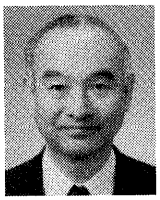
平成3年大阪大学大学院博士課程修了．同年同大学基礎工学部助手．現在，同大学情報処理教育センター講師．工学博士．分散システム運用技術，教育工学，ユーザインタフェース等の研究に従事．電子情報通信学会会員．

**西田 知博**（正会員）

昭和43年生．平成3年大阪大学基礎工学部情報工学科卒業．平成5年同大学大学院修士課程修了．平成8年同大学情報処理教育センター助手．主にHCIおよび教育工学に関する研究に従事．電子情報通信学会会員．

**辻野 嘉宏**（正会員）

昭和32年生．昭和54年大阪大学基礎工学部情報工学科卒業．昭和59年同大学大学院博士課程修了．同年同大学基礎工学部助手．現在，同大学大学院基礎工学研究科助教授（情報数理系）．工学博士．HCI，計算機言語，並行処理記述，ソフトウェア工学に関する研究に従事．IEEE, ACM，電子情報通信学会，日本ソフトウェア科学会各会員．



**都倉 信樹**（正会員）

昭和38年大阪大学工学部電子工学卒業．昭和43年同大学大学院博士課程修了．同年同大学基礎工学部講師．現在，同教授（情報数理系）情報教育，プログラムの技法と理論，計算機言語，アルゴリズム等の研究に従事．IEEE, ACM，日本ソフトウェア科学会各会員．