J-031

マルチモニタ環境のための ウィンドウへのカーソルジャンプによるポインティング法

CursorJump to Window : A Pointing Technique for Multiple Monitor Environments

小笠原 慧 † 小俣 昌樹‡ 今宮 淳美‡
Kei Ogasawara Masaki Omata Atsumi Imamiya

1．はじめに

近年，マルチモニタ環境はモニタの低価格化などの理由から，構築しやすく一般的となっている．マルチモニタを使用して情報表示領域を広げることで，ユーザはアプリケーションをモニタごとに振り分けて作業することができる．これにより，ユーザの作業効率の向上が見込まれる．しかし，Robertson らは，マルチモニタのように情報表示領域が広い環境では，以下のような問題が生じると指摘している[1]．

- ベゼル問題：モニタベゼルをまたがるカーソル移動をすると，ユーザが見るカーソルの軌跡が歪むため，ポインティング中に違和感を覚える．
- 末端情報へのアクセス問題：全体の画素数が増えたため，画面端までのポインティングに多くの時間が必要となる．
- カーソル消失問題：表示面積が増えたことで，カーソルを高速に移動する機会が多くなり，ユーザがカーソルを追跡できずに見失ってしまう．
- タスク管理問題：多数のウィンドウを開く機会が多くなるため，ユーザはそれぞれのウィンドウと表示されるモニタとの関係がわからなくなる．

これらの問題はユーザの負担を増やし，マルチモニタ環境によって向上した作業効率を下げる原因となる．そのため，マルチモニタ環境に適したポインティング方法が必要となる．

本研究では，これらの問題に対して，カーソルがデスクトップ上のどの位置にあっても，任意のウィンドウの中央へとカーソルを瞬間移動させる手法を提案する．これによって，マルチモニタ環境における無駄なカーソル移動を減らすことができるので，マルチモニタ環境を使用するユーザに効率的なポインティングを提供できると考える．本稿では，本提案手法を実装したポインティング支援システム「CursorJump to Window」の詳細と，マルチモニタ環境での有用性を評価するための実験と分析について述べる．

2．関連研究

本節では，マルチモニタや大画面のように情報表示領域が広い環境でのポインティングに関する研究を紹介する．

Robertson らは，前節のマルチモニタ環境のそれぞれの問題について Mouse Ether，Drag-and-pop，Auto-location Cursor および Group Bar を提案した[1]．Mouse Ether とは，モニタベゼルによる空間を補完することで，カーソルの軌跡が歪まないようにするシステムである．Drag-and-pop は，ユーザがアイコンをドラッグすると，アイコンに対応したアプリケーションがカーソルまで伸びるシステムである．これにより，ユーザは少量のカーソル移動量でアイコンのドラッグ＆ドロップを実行できる．Auto-location Cursor は，ユーザが Ctrl キーを押すと，カーソルを中心に波紋のようなアニメーションが現れるシステムである．Group Bar は，ユーザの好みにタスクのグループを作ることができるシステムである．これにより，グループごとの最小化や閉じるなどの操作ができる．

Necenta らは，ベゼル問題を解決するために，遠近法を用いたポインティング方法「Perspective Cursor」を提案した[2]．これは，ユーザの頭の位置から仮想的なレーザを伸ばし，レーザがモニタの画面と交差した位置へカーソルを表示するというシステムである．このとき，カーソルはユーザに対して一定のサイズと形を保って表示される．これにより，ユーザはベゼルによるカーソルの歪みやモニタごとの解像度の違いを意識せずにカーソルを操作できる．

Benko らは，要求したモニタへカーソルが瞬間移動する Pointer Warping を提案した[3]．このシステムでは，目的のモニタへ移動した後のカーソルの出現位置が 2 種類ある．ひとつは，各モニタでの最後のカーソル位置を記憶して，もう一度そのモニタに現われるとき，その位置へカーソルを表示する方法である（Frame-Memory 法）．もうひとつは，移動する前のモニタでのカーソル位置と同じ位置へカーソルを表示する方法である（Frame-Relative 法）．

一方，小俣らは，タブレットの向き情報を用いてモニタを選択するシステムを提案した[4]．これは，タブレットを回転してモニタを指定し，そのモニタの表示領域とタブレットのポインティング領域とを対応付けるシステムである．

Robertson らのシステムは，従来のシステムを拡張することで，ひとつひとつの問題を解決している．しかし，実際のポインティングタスクでは，それぞれの問題が組み合わさってユーザに負担をかける．そのため，複数の問題を同時に解決できるようなポインティング方法が必要であると考えられる．また，Necenta らのシステムは，ポインティングに遠近法を使用することで，常に一定の大きさと形のカーソルを操作できる．しかし，遠くにあるモニタには，モニタに対して非常に大きなカーソルが表示されるため，細部を操作するような作業には向かない．一方，Benko らや小俣らのシステムは，カーソルの瞬間移動やタブレットの向きとモニタとを連動させることで，マルチモニタ環境におけるモニタ切り替えにかかる

† 山梨大学大学院医学工学総合教育部修士課程 コンピュータ・メディア工学専攻，
University of Yamanashi

‡ 山梨大学大学院医学工学総合研究部，
University of Yamanashi

手間を減らしている．しかし，これらのシステムはモニタを切り替えるだけであり，その後のウィンドウ操作までは支援できていない．

3． CursorJump to Window

本研究では，マルチモニタ環境における問題点を解決するため，目標のウィンドウへ直接カーソルをジャンプさせる手法を提案する．以下では，本提案手法を実装したポインティング支援システム「CursorJump to Window（以下 CJtW と略記）」について述べる．

3.1. ウィンドウへのカーソルジャンプ

本研究では，マルチモニタ環境におけるモニタ間でのカーソル移動について，モニタの切り替え方法ではなく切り替えた後のカーソルの出現位置に着目する．これは，Benko らの研究では，マルチモニタ環境でのカーソルの瞬間移動の性能が，モニタの切り替え方法よりも切り替えた後のカーソルの出現位置に大きく依存するという結果に基づいている[3]．

カーソル出現位置を決定するために，ユーザがどのような場面でモニタを切り替えるかを考える．モニタを切り替えるひとつの例として，他のモニタ上にあるウィンドウの操作が挙げられる．本研究では図 1 のように，選択したウィンドウへカーソルが直接ジャンプする手法を提案する．これにより，ユーザは，モニタの切り替えと同時に，次の操作に取り掛かることができるので，ユーザの作業効率が向上すると考えられる．

3.2. 操作方法

ユーザがカーソルをジャンプさせるまでの一連の操作について説明する．なお，システムの操作にはサイドボタン付きのマウスを使用する．

はじめに，ユーザは，マウスのサイドボタンをクリックして，図 2 のようなアイコン選択ウィンドウをカーソルの近くに表示する．選択ウィンドウ上のアイコンは，各モニタに表示されているウィンドウのアイコンと対応している．カーソルをアイコンの上に移動するとウィンドウタイトルが表示される．次に，目標ウィンドウのアイコンを左クリックすると，カーソルは目標ウィンドウの中央にジャンプする．このとき，目標ウィンドウはアクティブ化されるので，ユーザは円滑にウィンドウ内でのタスクを実行できる．

本提案システムには，カーソルジャンプ以外にも次のような機能がある．アイコン選択ウィンドウ上のアイコンを右クリックすると，タスクバーでのウィンドウボタンの右クリックと同様に，最小化や最大化の操作ができる．また，アイコン選択ウィンドウ上のアイコンがない場所で右クリックすると，現在起動しているすべてのウィンドウに対して最小化や最大をしたり，システムを終了できる．アイコン選択ウィンドウを閉じるには，マウスのサイドボタンをもう一度クリックする．

3.3. 実装方法

CJtW の実装には，Visual Studio 2005 の Visual C#言語を使用した．システムは，マウスのサイドボタンクリックのメッセージをフックして，アイコン選択ウィンドウをカーソルの近くに表示する．アイコン選択ウィンドウが呼び出されるたびに，起動中のウィンドウの情報（座標，サイズ，ウィンドウタイトル，ウィンドウハンドル，アイコン）を取得し，ウィンドウリストとアイコンリストを更新する．アイコンリストは，アイコン選択ウィンドウ上のアイコンの位置座標や，ウィンドウリストに対応するタグなどの情報を保持する．システムはこのアイコンリストの情報を使用して，アイコン選択ウィンドウにアイコンを配置する．

ユーザがアイコンを左クリックすると，システムはウィンドウリストからアイコンのタグと一致したウィンドウを探し，マウスカーソルの座標をウィンドウの中央へ変更する．同時に，選択されたウィンドウをデスクトップの最前面に表示してアクティブにする．

3.4. マルチモニタの問題との関係

第 1 節で述べたマルチモニタ環境の問題について，本提案システムは，カーソルジャンプによってカーソルが他のモニタへ移動するので，モニタベゼルによる影響を受けない．したがって，ベゼル問題を解決できる．また，カーソルのジャンプ先が目標としているウィンドウであるので，ユーザはカーソルの表示位置を予測できる．したがって，カーソル消失問題を解決できる．

また，本提案システムは，ウィンドウの選択にアイコンを使用する．このとき，アイコンはウィンドウが表示されているモニタごとに整理されているので，ユーザは多数のウィンドウを開いても簡単にウィンドウを管理できる．したがって，タスク管理問題を解決できる．また，目標ウィンドウまでの距離に関係なく，ユーザは一連の動作と一定のカーソル移動量でカーソルを移動できる．これにより，末端情報へのアクセス問題を解決できる．

4． 評価実験

マルチモニタ環境における本提案システムの有用性を検証するため，CursorJump to Window（本提案システム），Mouse（従来のマウス操作），Frame-Memory 法（先行研究）および Frame-Relative 法（先行研究）の 4 種類の手法の比較実験と分析を実施した．また，実験データからはわからない本提案システムの特徴や欠点を知るため，被験者に提案システムについてのインタビューを実施した．

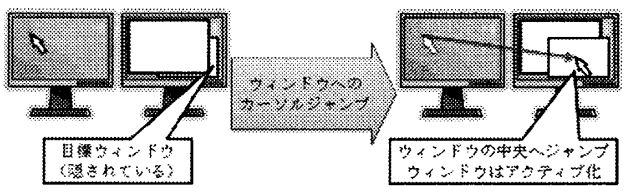

図 1：ウィンドウへのカーソルジャンプ

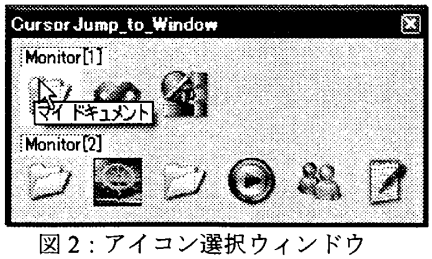

図 2：アイコン選択ウィンドウ

その結果，本提案システムは，目標のウィンドウが隠されている場合，他の手法よりも短いタスク遂行時間や少ないカーソル移動量でポインティングできることが示された．

4.1. 実験環境

本実験では，I-O DATA 社の液晶モニタ（LCD-A173）を 4 台配置し，図 3 に示すマルチモニタ環境を構築した．このとき，ユーザから見て左端のモニタをプライマリモニタに設定した．各モニタの解像度は 1280x1024 ドット，マウス感度は Microsoft 社 Windows2000 の初期設定のままである．各モニタには，それぞれボタンの付いている 2 枚のウィンドウ（図 3 参照）とモニタ全体を覆うための 1 枚のウィンドウが配置され，合計で 12 枚のウィンドウが配置された．サイドボタン付きのマウスには Timely 社の LS1600/Laser Mouse を使用した．

4.2. 実験タスクおよび実験手順

実験タスクは，2 つのウィンドウ間（開始ウィンドウと目標ウィンドウ）でのカーソル移動である．まず，スタートボタンの付いた開始ウィンドウが指定されるので，被験者はそのスタートボタンをクリックする．これによって，目標ウィンドウが指示されるので，目標ウィンドウのエンドボタンをクリックする．この一連の操作を 1 試行とする．

モニタ間の移動には CJtW，Mouse，および第 2 説で述べた Frame-Memory 法（以降 FM 法）と Frame-Relative 法（以降 FR 法）の 4 種類の移動方法を使用する．このとき，Mouse 手法では従来どおりモニタベゼルをまたがる移動であるが，それ以外の手法ではモニタベゼルをまたがない移動をするように，被験者に指示する．

開始ウィンドウと目標ウィンドウとの組み合わせは，それぞれのウィンドウのあるモニタ間の距離に基づいて 3 条件に分ける（Across-1，Across-2，Across-3）．この組み合わせは，モニタが 4 台あることから，全部で 12 通りとなる．たとえば，目標ウィンドウが開始ウィンドウから 3 台離れたモニタ上にある場合，タスク条件（距離）は Across-3 となる．更に，目標ウィンドウが他のウィンドウによって隠されているかどうかの状態の違いで 2 つの条件（Hide, non-Hide）に分ける．実験では，それぞれの距離と条件の組み合わせが無作為に選ばれ，各条件の組み合わせを手法ごとに 6 試行実施する．したがって，4 種類の手法，3 種類の距離の条件，2 種類の状態の条件，6 回の繰り返しから，ひとりの被験者につき 144 試行実施することになる．

実験は練習パートと本番パートに分かれる．練習パートでは，本番と同様のタスクをそれぞれの手法で 5 分間ずつ実施してもらう．5 分経過したら被験者には十分慣れたかどうかを尋ねる．被験者が十分に慣れたようであれば次の手法の練習に取り掛かる．練習する手法の順番は，被験者ごとに無作為で選ばれる．本番パートでは，練習パートと同じ手法の順番で実験タスクを実施する．各手法を開始する前に，手法の操作確認のための時間を 1 分間程度設ける．

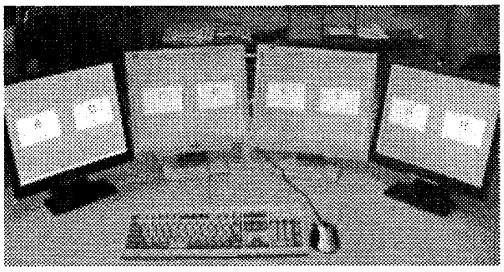

図 3：実験環境

計測データは，タスク遂行時間〔s〕，カーソル移動量〔pixel〕およびエラー回数である．タスク遂行時間とカーソル移動量は，スタートボタンのクリックからエンドボタンのクリックまでとする．エラー回数は，スタートボタンをクリックしてから目標ウィンドウ以外のボタンをクリックした回数である．

4. 3. 仮説

本実験では次のような仮説を立てる．

- 仮説 1 ：目標ウィンドウの状態が Hide の場合，距離に関係なく CJtW が最も早くタスクを遂行できる．
- 仮説 2 ：CJtW は他の手法に比べて少ないカーソル移動量でタスクを遂行できる．

4.4. 実験結果

被験者は，21 歳から 33 歳までの男性 14 名と女性 4 名である．このうち，普段からマルチモニタを使用している被験者は 2 名，普段からサイドボタン付きのマウスを使用している被験者は 2 名である．

図 4 に手法と距離についての平均タスク遂行時間のグラフを示す．また，図 5 に手法と状態についての平均タスク遂行時間のグラフを示す．平均タスク遂行時間について 3 要因の分散分析をおこなった結果，手法と距離，手法と状態で 1 次の交互作用が有意であることがわかった（$p < 0.01$）．各交互作用について手法の単純主効果の分析を行った結果（$p < 0.01$），手法と距離について，CJtW と

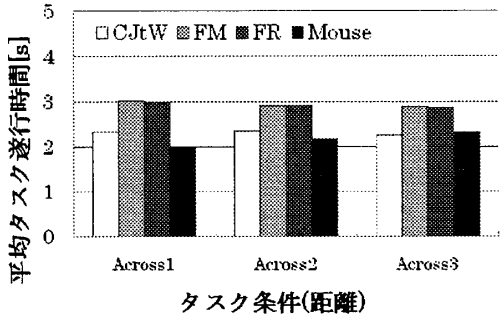

図 4：平均タスク遂行時間（手法と距離）

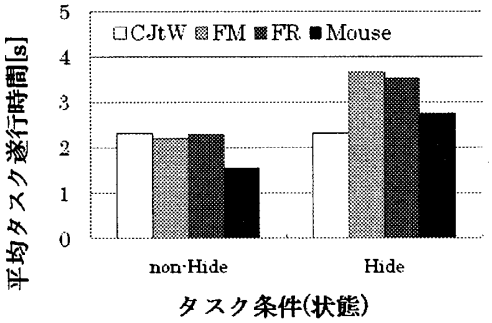

図 5：平均タスク遂行時間（手法と状態）

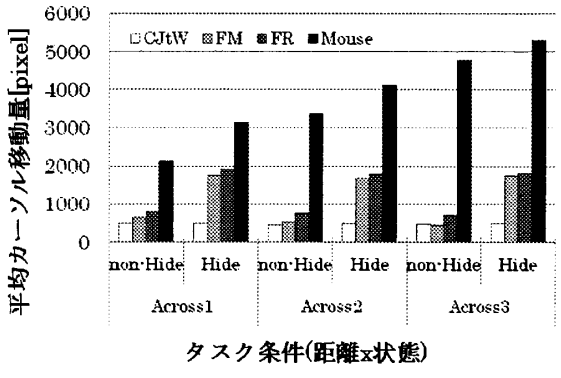

図６：タスク条件ごとの平均カーソル移動量

Mouse は FR 法と FM 法に対して，すべての距離で有意に早くタスクを遂行できることがわかった．また，Mouse は CJtW に対して，距離が Across-1 の場合，有意に早くタスクを遂行できることがわかった．しかし，距離が Across-2, Across-3 の場合，CJtW と Mouse の間に有意差は見られなかった．また，手法と状態については，状態が Hide の場合，CJtW は他の手法よりも有意に早くタスクを遂行できることがわかった．しかし，状態 non-Hide の場合，CJtW は FR 法と FM 法に対して有意差が見られなかった．これらの結果から，本提案システムはタスク遂行時間について，目標ウィンドウが隠れている場合に，他の手法よりも早くタスクを遂行できることがわかる．

図６にタスク条件ごとの平均カーソル移動量を示す．平均カーソル移動量について３要因の分散分析をおこなった結果，２次の交互作用が有意であった（$p < 0.01$）．下位検定をおこなった結果（$p < 0.01$），CJtW は FR 法と Mouse に対して，すべての距離と状態において有意に少ないカーソル移動量でタスクを遂行できることがわかった．また，CJtW は FM 法に対して，状態 Hide の場合，距離に関わらず，有意に少ないカーソル移動量でタスクを遂行できることがわかった．これらの結果から，本提案システムは目標ウィンドウの距離や状態に関係なく，Mouse と FR 法よりも少ないカーソル移動量でタスクを遂行できることがわかる．

また，平均タスクエラー率について３要因の分散分析をおこなった結果，主効果と交互作用ともに有意差は見られなかった（$p < 0.05$）．そのため，手法によるタスクエラー率の差は無いといえる．

５．考察

仮説１について検証する．実験結果から，仮説どおり，CJtW は，目標ウィンドウが隠れている場合，他の手法よりも早くタスクを遂行できることがわかった．他の手法の場合，モニタ間を移動する操作の後に目標ウィンドウをアクティブ化する操作が必要である．一方，CJtW の場合，カーソルジャンプによってモニタを切り替えるのと同時に目標ウィンドウがアクティブ化するので，他の手法よりも早くタスクを遂行できたと考えられる．

仮説２について，実験結果から，CJtW は Mouse と FR 法に対して，少ないカーソル移動量でタスクを遂行できることがわかった．しかし，FM 法に対しては，状態 non-Hide の場合，カーソルの移動量に有意な差は見られなかった．そのため，仮説２のように，CJtW がすべての条件で最もカーソル移動量の少ない手法であるとは言えない．FM 法との間にカーソル移動量に差がなかった理由について考察する．FM 法は各モニタでの最後の位置にカーソルを表示する方法である．そのため，ジャンプ先のモニタの最後のカーソル位置が目標ウィンドウの上であった場合，FM 法は CJtW を使った場合と同じような位置へカーソルを表示する．これにより，状態が non-Hide だった場合には，同じくらいのカーソル移動量になったと考えられる．一方，CJtW を使用すると，選択したウィンドウはアクティブとなり，カーソルはウィンドウの中央へと表示される．これにより，ポインティング中の無駄なカーソル移動がなくなるので，Mouse と FR 法よりも少ないカーソル移動量でタスクを遂行できたと考えられる．

また，実験結果をみると，CJtW は他の手法に比べてデータの分散が少ない．このことから，CJtW はタスクの条件に関係なく，一定の時間とカーソル移動量で目標のタスクを遂行できることがわかる．特に，カーソル移動量は約 500pixel 前後と少ないカーソル移動量でタスクを遂行している．これは，モニタの中央からタスクバーまでの距離とほぼ等しい．

６．おわりに

本研究では，カーソルを目標ウィンドウへ直接移動させるポインティング法として「CursorJump to Window」を開発した．さらに，本システムのマルチモニタ環境における有用性を検証するために，従来手法との比較実験を実施した．その結果，本提案システムは，目標ウィンドウが隠されている場合やカーソルを移動する距離が長い場合，従来よりも短い時間と少ないカーソル移動量でタスクを遂行できることがわかった．このことから，本提案システムはマルチモニタ環境において有用であるといえる．

今後の課題として，ウィンドウの選択方法を改善する．現システムでは，モニタごとに整理されたアイコンをクリックしてウィンドウを選択する．しかし，この方法では，ウィンドウの数に比例してアイコンの数が増えてしまうため，ひとつのモニタに多数のウィンドウがある場合，アイコンの選択に時間がかかってしまう．したがって，ウィンドウの選択頻度に応じてアイコンの大きさを変える新たなウィンドウ選択方法を検討する．

参考文献

[1] Robertson, G., Czerwinski, M., Baudisch, P., Meyers, B., Robbins, D., Smith, G. and Tan, D.: “The Large-Display User Experience,” IEEE Computer Graphics and Applications, pp.2-9, 2005.

[2] Necenta, A, M., Sallam, S., Champoux, B., Subramanian, S. and Gutwin, C.: “Perspective Cursor : “Perspective-Based Interaction for Multi-Display Environments,” CHI 2006, pp.289-298, 2006.

[3] Benko, H., Feiner, S.: “Pointer Warping in Heterogeneous Multi-Monitor Environments,” Graphics Interface Conference 2007, pp.111-117, 2007.

[4] 小俣昌樹, 小坂真裕, 今宮淳美: “マルチディスプレイ環境のためのタブレットの向きに連動するポインティング法,” FIT2007, pp.299-302, 2007.