スイッチバックカーソル：重なりあったウィンドウ間を移動可能なマウスカーソル操作手法

Switchback Cursor: A Mouse Cursor Operation for Overlapped Windowing

山中 祥太　　宮下 芳明*

Summary.　複数のウィンドウを開きつつ作業をしている場合，手前のウィンドウの陰に隠れた部分は操作できないため，最前面のウィンドウを切り替えたり，一旦移動したりするといった行為が頻繁に要求される．そのようなウィンドウ操作が生じる原因は，ウィンドウが奥行きをもって 3 次元に配置されるのに対し，カーソルは 2 次元にしか移動できないことにあると我々は考えた．本稿では，マウスカーソルをウィンドウの奥に潜りこませるシステム“ スイッチバックカーソル ”を提案する．評価実験では，提案手法がマウス単独操作及びマウス・キーボード併用操作よりも 1%有意で速くタスクを完了させられることが明らかになった．実験後，被験者は普段のウィンドウ操作が煩わしく感じるようになった．

1　概要

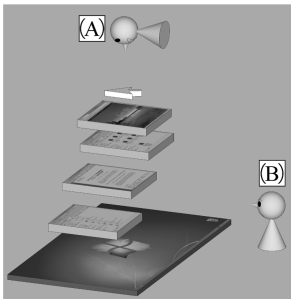


図 1. 重なりあったウィンドウに対する視点

図 2. カーソルが奥のレイヤに移動する

人は良くも悪くも，コンピュータのユーザインタフェースに段々と慣れていくものである．複雑な操作が求められる場面でも，特に疑問を持たずに使い続けてしまい，その不便さにも気付かない．例えば，非接触 IC カードが発明されるまで，目的地への切符を並んで現金で買う行為はそれほど不便でない当たり前のこととして認知されていた．ところが一度 IC カードに慣れてしまうと，現金での切符購入がとても煩わしく感じられるようになった．本稿で述べるウィンドウ操作の煩わしさも同様の問題であり，提案システムを使用した被験者は普段のウィンドウ操作を煩わしく思うようになっている．

PC で複数のウィンドウを立ち上げながら作業をしているとき，私たちは実に多くのウィンドウ操作を行っている．移動やサイズの調整はもちろんのこと，ウィンドウをクリックして最前面に出す操作も頻繁に行われる．これらはアプリケーション操作やファイル整理など，本来行いたいことに付随して必要になってしまうものであり，ときには煩雑なウィンドウ操作が要求される（具体例は 3 章で挙げる）．

こういった操作が必要になる原因は，ウィンドウとマウスカーソルの次元が異なることにあると筆者らは考えた．図 1 は複数のウィンドウが重なりあった状態の 3 次元イメージ，(A) は普段我々が見ている視点である．このようにウィンドウは奥行きを持って 3 次元に配置されるのに対し，カーソルは 2 次元にしか移動できないため，奥にあるウィンドウは最前面に出してから操作を行わなければならない．

我々が提案するのは，カーソルをウィンドウの奥へと潜りこませる操作手法である．マウスの左右ボタンを同時に押しながらカーソルを移動させ，ウィンドウの端から滑りこませることでウィンドウの奥にあるものを操作可能にする．カーソルが潜りこむ様子を図 1(B) 視点から見たイメージが図 2 である．

本稿では，まず従来のウィンドウシステムの問題点について述べ，それを解決するための提案手法を示した上で，具体的な利用例を挙げる．その後，システムの実装について説明し，評価実験と考察を行う．そして関連研究を紹介し，最後にまとめを述べる．

2　解決しようとする問題

本来行いたい操作をする際に，それに付随したウィンドウ操作が必要な場合がある．その原因は，ウィンドウとカーソルの次元が異なるからである．ウィンドウは手前から順に奥行きをもって配置されており，3 次元のレイヤ構造をなしている．一方，カーソルは 2 次元平面上でしか移動できないため，ウィンドウに隠れている奥の部分は操作できず，奥のウィ

* Shota Yamanaka, 明治大学大学院 理工学研究科 新領域創造専攻 ディジタルコンテンツ系, Homei Miyashita, 明治大学大学院 理工学研究科 新領域創造専攻 ディジタルコンテンツ系, 独立行政法人科学技術振興機構 CREST

ンドウをクリックして最前面に出してから目的の操作を行わなければならない．この次元の不一致による問題は加藤ら [1] や大内ら [2] によって以前から指摘されている．この問題はウィンドウ同士が重なりあうことを認めるオーバーラップウィンドウ方式独特のものであり，ウィンドウを敷き詰めるタイルウィンドウ方式では問題ないが，こちらは個々のウィンドウサイズが小さくなることから視認性に問題が生じ，オーバーラップウィンドウ方式に比べるとあまり利用されていないのが現状である．

また，ウィンドウフォーカスの問題もある．ウィンドウフォーカスとは，キーボードやマウスからの入力を受け付けるウィンドウを示すものであり，フォーカスが当たっている状態を「ウィンドウがアクティブである」という．Windows7 では，クリックしたウィンドウにフォーカスが当たる方式（Focus Follows Click, FFC）が採用されているが，この方式ではアクティブになったウィンドウが最前面に出るため，手前にあったウィンドウが隠れてしまう．これは別の作業を始める場合には問題にならないが，奥のものに対して僅かな操作をする場合でも，その度にウィンドウの前後関係を変更しなければならない．カーソルの座標によってフォーカスが移る方式（Focus Follows Mouse, FFM）もあるが，マウスを不意に移動させると作業が妨げられるという問題があり，Windows OS や Mac OS では採用されていない．

このように，一般的なウィンドウ表示方法とマウスカーソル操作方法はいずれも問題点を抱えている．そもそも，奥にあるものに対して操作を行いたいときに本来必要のないウィンドウ操作が求められることは，オーバーラップウィンドウ方式の UI における重大な問題である．

3 提案手法

本稿で提案するのは，目的のウィンドウまでマウスカーソルを奥へと移動させて操作する手法である．カーソルがディスプレイ内を 3 次元移動し，マウスカーソル自身が奥のファイルやウィンドウに触れに行くような直感的操作を実現する．

筆者らの考えは，従来の方法でのウィンドウ切り替えをせずに提案手法を常に利用すべきだということではない．ある程度の時間他のアプリケーションだけを利用するのであればそのウィンドウを最前面に出した方が良いであろうし，キーボード操作がふさわしい場合にはそうすべきである．ただ，もしそういった操作が煩雑で，提案手法が有効に働きそうな場面ではこちらを利用すべきである，という主張である．そのため，提案手法は従来の操作方法と共存でき，場面に応じた柔軟な対応が可能である．現状のオーバーラップウィンドウ方式と親和性が高く，ユーザが慣れていた操作を一切阻害しないことも特長である．このポリシーに沿った実装にするため，潜りこむ機能を使うための専用デバイスを必要とせずに，一般的な左右ボタンを持ったマウスで操作可能にした．以下では，従来の GUI 操作において本稿で問題としている具体的な場面について例を挙げた上で，提案手法による解決方法を説明する．

3.1 奥のウィンドウを操作すると手前のウィンドウが隠れる場合

例えば図 3 のように，音楽プレイヤを起動しながらウェブブラウジングをしている状態で，ブラウザに隠れた位置にある音楽ファイルを再生する操作を考える．通常は，奥のフォルダをクリックして最前面に出し，目的のファイルをプレイヤにドラッグアンドドロップする方法が考えられる（プレイヤが関連付けられていれば，ファイルをダブルクリックでも可）．しかし，奥のフォルダをクリックした時点でブラウザの一部がフォルダに隠れてしまうため，操作が完了してからブラウザをクリックして再度最前面に出すか，あるいはウィンドウ同士が重ならないように位置を調節する必要があり，煩雑な操作が求められる．提案手法では，ブラウザの奥に潜って音楽ファイルをドラッグアンドドロップするだけでよく，奥のフォルダを一旦最前面に出す操作も，ブラウザを再度最前面に戻す操作も必要ない．

図 3. 音楽を再生しながらウェブブラウジングする様子

3.2 デスクトップのアイコンをダブルクリックする場合

図 4 左のように，ウィンドウが複数起動しているときにデスクトップのアイコンをダブルクリックしてアプリケーションを起動する場合を考える．通常は，起動中のウィンドウを全て最小化したり移動させたりした後に，デスクトップで目的のアイコンをダブルクリックし，さらに元の作業に復帰するため

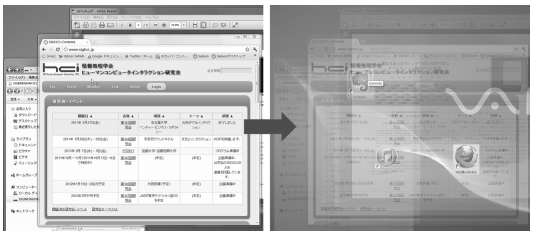


図 4. カーソルがデスクトップに到達する様子

にウィンドウの状態を戻さなければならず，操作量が多くなる．これに対し提案手法では，図 4 右のように，デスクトップへカーソルを潜りこませて目的のアイコンをダブルクリックするだけでよい．起動中のウィンドウに対する操作は一切必要なく，操作量が大幅に軽減される．

4 システム

本システムは Windows 上で動作し，各ウィンドウのハンドラや座標などを取得して，前面から順に配列に格納し，以後これらを監視し続ける．また，カーソル直下のウィンドウのインデックスを保持する．左右ボタンを同時押しをすると，カーソルは直下のウィンドウと同一レイヤに移動する．例えば図 2 の初期位置から，カーソルを右へ動かしてウィンドウ 1 上で同時押しすると，マウスカーソルはウィンドウ 1 のレイヤに移動する．同時押しをしたままマウスを左へと動かすと，カーソルはウインドウ 0 の裏側へと潜り込んだように表示される．ウィンドウがさらに奥にある場合には，同時押しをしたままウィンドウの裏側へマウスカーソルを移動させることで，順に深いレイヤへと移動していく．

同時押しした時点で，カーソルより手前にあるウィンドウは最前面化設定し（解除されるまで最前面にある状態にする），奥が見えるように半透過処理を行い，さらにマウスイベント透過設定をする．これにより，今後クリックやドラッグをしてもカーソルより手前のウィンドウには操作が行われず，かつカーソル直下のウィンドウにマウスイベントが起こってもウィンドウの前後関係は変更されない．カーソルより手前のウィンドウを完全透過にする方法も考えられるが，平岡の研究 [3] では，ウィンドウをタイトルバーのみの表示にすると見失うと指摘されており，概形を把握可能な半透過にとどめた．

非同時押し時には，カーソルは常に手前方向に移動しようとする．ただし，カーソルよりも手前にウィンドウがある状態で同時押しを解除しても，マウスカーソルは手前方向に移動しない．このときカーソルはそのウィンドウの 1 つ奥のレイヤに存在し続ける．つまり図 2 において，ウィンドウ 0 の奥で同時押しを解除すると，カーソルはウィンドウ 0 の裏側にとどまる．カーソルをウィンドウ 0 に当たらない位置まで動かせば最前面に戻ってくる．

同時押し中は，カーソルの形状を通常時よりも 15%程度縮小したものに変更し，ウィンドウにカーソルを押し付けているような表示にする．左クリック時も同様である．同時押し中にカーソルがウィンドウの周辺にあるときには，カーソルの向きを変更し，潜りこめる方向をユーザに示す．奥のレイヤに移動したタイミングで，カーソルを一瞬だけさらに縮小し，金属音を鳴らすことで，奥のウィンドウに衝突したことを表現する．

5 実験

提案手法の有効性を確かめるために，簡単なタスクを行う時間を計測し，他の操作方法と比較する実験を行った．実験で行ったのは，提案手法での操作，マウス単独操作，マウス・キーボード併用操作である．提案手法は元々マウス操作をサポートするために考案したシステムであり，マウス単独操作と比較実験を行った結果を過去に報告している [4] が，今回はキーボードによってマウス操作をサポートした場合とも比較を行った．

5.1 実験方法

異なるフォルダ間でのファイル移動を行うタスクを被験者に与え，操作を完了するまでの時間を計測した．被験者は大学生及び大学院生の合計 10 名である．操作の順序は被験者毎に変更し，各操作方法とも 5 回ずつタスクを行った．実験ではあらゆるマウス操作を認め，キーボードでも禁止した操作はない．ただし，マウス・キーボード併用操作では，マウス単独操作と実験内容を分けるために，必ず 1 回以上のキーボード操作を行うものとした．被験者には可能な限り速くタスクを行うように伝えた．実験開始時のフォルダの配置を図 5 に示す．また，移動前と移動後のファイル配置のイメージを図 6 に示す．ファイルを移動する順序は被験者の任意である．

図 5. 実験開始時のウィンドウの配置

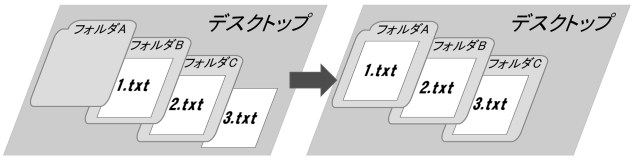


図 6. 移動前（左）と移動後（右）のファイルの配置

各操作方法での実験を行う前に，実験と同じファイル・ウィンドウの配置で 15 分間を上限に練習を行わせた．これは，実験を開始してからファイルを移動する順序やウィンドウを移動する位置などを試行錯誤しないようにし，操作をスムーズに行った場合の時間を比較するためである．さらに，実験時に誤った操作をした場合にも，同様の理由でタスクをやり直した．なお，提案手法による実験では，カーソルより手前のウィンドウの透過率を，筆者らが実際にタスクを行って操作しやすいと感じた 23%に固定した（実際にはユーザによって調整可能である）．

5.2 結果と考察

図 7 は，被験者毎の平均操作時間である．被験者 10 を除き，提案手法での操作時間が最も短い結果となった．平均操作時間を被験者毎の対応ありで分散分析（反復測定）した結果，提案手法での操作とマウス単独操作間で 1%水準，提案手法での操作とマウス・キーボード併用操作間で同じく 1%水準の有意差があり，提案手法の有効性が確認できた．

実験後に 5 段階評価のアンケートを実施し，本システムのユーザビリティについて尋ねた．質問項目と回答結果を表 1 に示す．この他に，タスクを最もこなしやすかった操作を選ぶ項目では，提案手法が 8 名，マウス単独及び併用操作が各 1 名であった．さらに，「今後，提案システムを使いたいと思うか」という質問に対して被験者全員が「そう思う」と回答したことから，提案手法が好意的に受け止められていると考えている．実験を通しての所感を自由記述で求めたところ，提案手法の便利さや音によるレイヤ移動の提示は概ね好評な一方で，潜っているレイヤを把握しづらい，手前のウィンドウの文字が重なって奥が視認しづらいといった指摘を受けた．

操作ミス回数は，被験者によってばらつきがあり，提案手法で 0〜9 回，マウス単独で 1〜5 回，マウス・キーボード併用で 0〜11 回見られた．ただし，被験者の誤操作（同時押しミスなど）か，システム側の問題（処理遅れなど）かを区別できない場合も全てカウントしている．

提案手法による実験を最初に行った被験者が，マウス単独操作の実験を行った際に「いつも行っている操作だが，いちいちウィンドウを操作するのがこんなに面倒だとは気付かなかった」という旨の感想を述べた．また，実験後に「一度潜りこむ機能を使ってからは，普段のウィンドウ操作が煩わしく感じるようになった」との発言があった．被験者が実験を通して，従来の GUI 操作の問題点を認識するとともに，提案手法の便利さを感じられたと考えている．

表 1．アンケート項目と回答の平均値

質問項目	回答
左右ボタン同時押しによって潜る，という操作のしやすさ	3.4
非同時押し時は手前に移動するシステムの操作しやすさ	3.5
潜った先での (ウィンドウの裏側での) マウスカーソル操作のしやすさ	3.7
目標のウィンドウへの辿り着きやすさ（意図したレイヤでカーソルを止められるか）	4.0
音によるレイヤ移動の分かりやすさ	4.4
マウスカーソルの拡大・縮小によるレイヤ移動の分かりやすさ	2.8
潜って操作するという機能の便利さをどのくらい感じたか	4.4
提案手法での，操作したいファイルへの辿り着きやすさ	4.3
マウス単独の場合の，操作したいファイルへの辿り着きやすさ	2.4
マウス・キーボード併用操作の場合の，操作したいファイルへの辿り着きやすさ	2.9

6 関連研究

ウィンドウが重なることで発生する問題については様々な解決方法が試みられている．ウィンドウ切り替えの煩雑さを解決するために，加藤らはスライダを用いてウィンドウを手前から順に非表示にするぱらぱらウィンドウを提案した [1]．Ishak らは，ウィンドウを内容に応じて透過処理し，同時に表示されるコンテンツを増やすことでウィンドウ操作回数を軽減するシステムを開発した [5]．Faure らは，マウスボタンを一定時間以上押してからドラッグすることで，任意のレイヤにあるウィンドウ（デスクトップ含む）を最前面に出すシステムを開発した [6]．小林らは，ドラッグ中にマウスホイールを回転させることで，カーソル直下のウィンドウを手前から順に非表示にする手法を提案している [7]．このシステムは，通常のスクロール操作との競合を避けるために，利用可能な場面をドラッグ操作中に限定している．神原らのちらりウィンドウは，ジョイスティックを操作することで視点を移動させ，前面のウィンドウに隠された部分も閲覧可能にするシステムである [8]．柴田らは，複数のウィンドウを一括することによってマルチタスクを支援する手法を提案しており，効率的なウィンドウ操作を実現している [9]．久納らは，カメラを用いて実世界での視点移動をディスプレイ

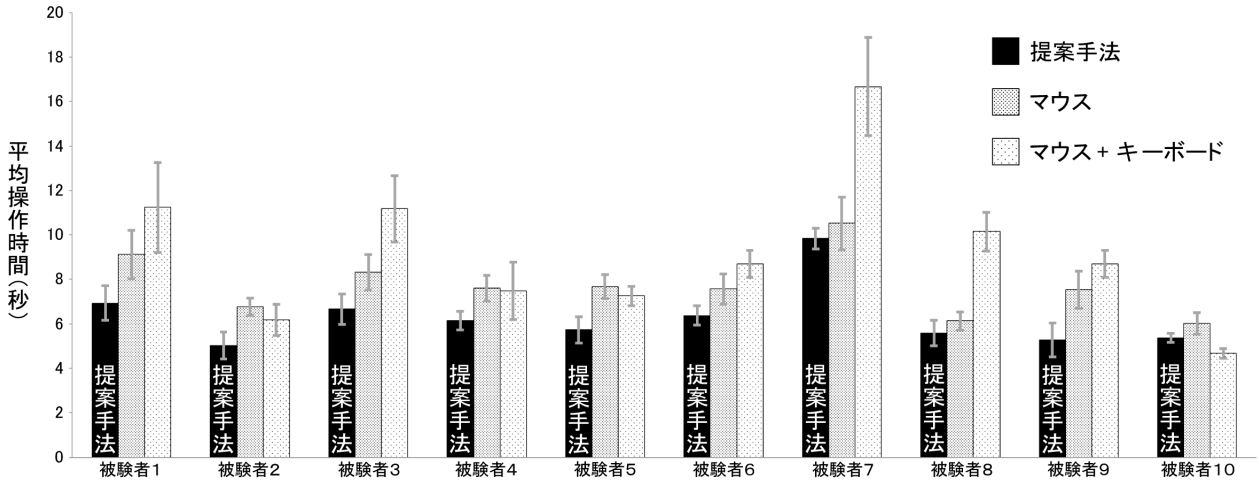


図 7．被験者毎の平均操作時間の比較（提案手法は他 2 つより 1%有意で速い）

内に反映させるシステムを提案した [10]．大野は，視線を利用してウィンドウの移動や最大化，最小化などを行う手法を提案した [11]．大和らは，視線とマウスを併用し，クリック箇所は視線で選択し，確定操作をマウスで行う手法を提案した [12]．Endert らの ChairMouse は，マルチディスプレイ環境でのカーソル移動距離を低減させるために，ユーザが座っている回転式椅子の動きをカーソル移動に反映させるシステムである [13]．これらの中にはマウス以外のデバイスを必要としているものが多いが，本研究と非常に近い問題設定をしている．

本提案手法と同様に，マウスカーソルに通常とは異なる挙動をさせ，それに伴ってカーソルの表示を変更するシステムも存在する．Grossman らの Bubble Cursor[14] は，カーソルを円形に表示し，ポインティングを一点ではなく広範囲の円にすることで，移動時間を短縮させるシステムである．重森らは Bubble Cursor を拡張し，一点と円形とを自動で切り替えるシステムや，目標物の位置とカーソルの移動方向を考慮して選択範囲が変化するシステムを開発した [15]．マウスの移動量を低減するシステムに，カーソルを目標物までワープさせる Delphian Desktop[16] や，目標物をカーソル付近まで引き寄せる Drag-and-Pop[17] がある．Kobayashi らは，複数のカーソルを表示することで移動距離を短縮させる Ninja Cursors[18] を提案している．中村らは 2 つのマウスを用いてウィンドウを操作する環境を提案している [19]．目標物に近い方のカーソルを動かすことでマウス移動量の軽減が可能である．

Robertson らの Task Gallery[20] は，仮想的な 3 次元空間表示を利用することでウィンドウの重なりをなくし，さらに配置を記憶しやすくしたシステムである．大内らは，3 次元デスクトップ環境においてマウスが 2 次元でしか操作できないことを問題視し，3 次元入力が可能なデバイスを使用する手法を提案した [2]．Dragicevic は，紙のようにウィンドウをめくることで，奥にあるウィンドウへのドラッグアンドドロップを可能にするシステム fold-and-drop を提案した [21]．本提案手法が奥から手前へのドラッグアンドドロップに有効なのと対照的であるが，手前のウィンドウが移動元か移動先かによって有効な手法は異なると考えられる．また，fold-and-drop はめくる操作をドラッグ中に限定しているが，本提案手法は奥のウィンドウでフォルダを探索したり，デスクトップ上のショートカットを起動したりと，より多岐にわたる場面での利用が可能である．

7　まとめ

本稿では，オーバーラップウィンドウ方式においてカーソルをウィンドウの奥へと潜りこませる操作手法を提案し，システムの実装と評価を行った．提案手法は従来の操作よりタスクを 1%有意で速く完了させられることが分かり，有効性を示した．今後は，6 章に挙げた既存の手法や，ファイルの切り取り・貼り付け専用のボタンが搭載されたキーボードやマウスで操作を行った場合との比較実験を行う予定である．また，左右ボタンの同時押しや，ウィンドウの半透過処理の妥当性について検討していく．

実は不便なものであっても，慣れてしまうと不便さを感じなくなってしまうことは多い．それは冒頭の非接触 IC カードの例のみならず，マルチタッチによる拡大・縮小操作や予測変換機能についてもいえる．切符を購入する不便さ，シングルタッチで全ての操作を行う不便さ，文章を全て入力してから変換する不便さは，これらの発明によってむしろ実感されるものとなった．他にも，もしかしたら我々が不便だと気付いていないだけで，UI の世界にはまだ問題が山積みなのかもしれない．本稿で被験者が提案システム使用後に普段のウィンドウ操作を煩わしく感じたように，本当に不便なことは，便利なものに出会って初めて気付くのである．

参考文献

[1] 加藤 直樹, 小國 健. ぱらぱらウィンドウ：ウィンドウの切り替えを容易にするインタフェース. インタラクション 2003 論文集, pp. 123–130, 2003.

[2] 大内 勇佑, 西野 浩明, 宇津宮 孝一, 賀川 経夫. 触感提示機能を有する 3 次元デスクトップ環境の開発. 火の国情報シンポジウム 2011, 2011.

[3] 平岡 茂夫. ぼかしの効果を活用したインターフェースデザイン. 福岡工業大学情報科学研究所所報, Vol. 12, pp. 7–13, 2001.

[4] 山中 祥太, 宮下 芳明. 重なりあったウィンドウ間を移動可能なマウスカーソル操作手法の提案. 情報処理学会研究報告, ヒューマンコンピュータインタラクション研究会報告, Vol. 2011, No. 13, pp. 1-8, 2011.

[5] Edward W. Ishak and Steven K. Feiner. Interacting with Hidden Content Using Content-Aware Free-Space Transparency. In Proceedings of the 17th annual ACM Symposium on UIST '04, pp. 189–192, 2004.

[6] Guillaume Faure et al. Power tools for copying and moving: useful stuff for your desktop. In Proceedings of CHI '09, pp. 1675–1678, 2009.

[7] 小林 正朋, 五十嵐 健夫. 活用：マウスホイール. インタラクション 2005 論文集, pp. 175–176, 2005.

[8] 神原 啓介, 安村 通晃. ちらりウィンドウ：隠れたウィンドウを覗き見る. インタラクション 2004 論文集, pp. 47–48, 2004.

[9] 柴田 博仁, 大村 賢悟. ウィンドウをドッキングすることによるマルチタスキング支援. インタラクション 2011 論文集, pp. 391–394, 2011.

[10] 久納 章寛, 岡本 壮平, 武藤 直美, 中島 誠, 伊藤 哲郎. 層構造の作業環境におけるユーザ意図の把握. FIT2002 情報科学技術フォーラム 情報技術レターズ, pp. 199–200, 2002.

[11] 大野 健彦. 視線を利用したウインドウ操作環境. 電子情報通信学会技術研究報告, Vol. HIP99–29, pp. 17–24, 1999.

[12] 大和 正武, 門田 暁人, 松本 健一, 井上 克郎, 鳥居 宏次. 一般的な GUI に適した視線・マウス併用型ターゲット選択方式. 情報処理学会論文誌, Vol. 42, No. 6, pp. 1320–1329, 2001.

[13] Alex Endert et al. ChairMouse: leveraging natural chair rotation for cursor navigation on large, high-resolution displays. CHI Extended Abstracts ACM, pp. 571–580, 2011.

[14] Tovi Grossman and Ravin Balakrishnan. The Bubble Cursor: Enhancing target acquisition by dynamic resizing of the cursor 's activation area. In Proceedings of CHI '05, pp. 281–290, 2005.

[15] 重森 晴樹, 入江 健一, 倉本 到, 渋谷 雄, 辻野 嘉宏. バブルカーソルの GUI 環境への適用と拡張. インタラクション 2006 論文集, pp. 21–22, 2006.

[16] Asano Takeshi et al. Predictive interaction using the delphian desktop. In Proceedings of the 18th annual ACM Symposium on UIST '05, pp. 133–141, 2005.

[17] Patrick Baudisch et al. Drag-and-Pop and Drag-and-Pick: techniques for accessing remote screen content on touch- and pen-operated systems. In Proceedings of Interact 2003, pp. 57–64, 2003.

[18] Masatomo Kobayashi and Takeo Igarashi. Ninja Cursors: Using Multiple Cursors to Assist Target Acquisition on Large Screens. In Proceedings of CHI '08, pp. 949–958, 2008.

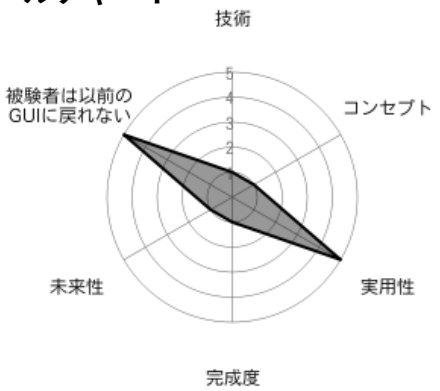
[19] 中村 聡史, 塚本 昌彦, 西尾 章治郎. 活用：2 つのマウスを用いたウィンドウ操作機構の設計と実装. 情報処理学会研究報告, Vol. 99, No. 35, pp. 1–6, 1999.

[20] George Robertson et al. The Task Gallery:A 3D window manager. In Proceedings of CHI '00, pp. 494–501, 2000.

[21] Pierre Dragicevic. Combining Crossing-Based and Paper-Based Interaction Paradigms for Dragging and Dropping Between Overlapping Windows. In Proceedings of the 17th annual ACM Symposium on UIST '04, pp. 193–196, 2004.

[22] 渡邊 恵太, 安村 通晃. RUI: Realizable User Interface カーソルを用いた情報リアライゼーション. 第 27 回ヒューマンインタフェース学会研究会「VR の心理と生理」, ヒューマンインタフェース学会研究報告集, pp. 35–38, 2004.

アピールチャート



未来ビジョン

シンプルで効果的な インタフェースデザイン

本稿で実現できているかはわからないが，筆者らが理想としている研究のあり方は，シンプルで効果的なインタフェースデザインである．例えば，マウス操作で触覚を提示する RUI[22] がある．通常ならマウスにバイブレータのような触覚提示機構を埋め込む方法が思い浮かぶが，この研究は視覚的・聴覚的情報によって擬似触覚 (VisualHaptics) の提示に成功している．筆者らも，研究のアイデアを探っているときに，どうしても「大掛かり」な問題解決を考えがちであるが，VisualHaptics のような解決法は，とてもシンプルかつ効果的で，しかもそれゆえに容易に普及させることができるメリットがある．（ウェブサイト上で体験を提供することができる長所もある）

単純かつ有効な考え方に出会ったとき，筆者らは「やられた」と感じることが多い．いつか，このようなアイデアを提案して人に「やられた」と思ってもらいたいと夢見ている．WISS の会議では，限られた時間ではあるが，ぜひ「シンプルで効果的なインタフェースデザイン」について，いくつかの具体例やそのエッセンスを取り出し，単純な解法があるならそれをいち早く考案するための議論を行いたい．「わざわざそんな大掛かりなことしなくても，○○しちゃえばそれでいいじゃん」ということに即座に気付けるようになれば，より多くのことが解決される未来社会が到来するはずである．本稿第 1 章では問題を発見することの難しさを述べたが，WISS ではそれと併せて解決法を発見する難しさについても議論したい．