明治大学大学院理工学研究科

2015 年度

博士学位請求論文

グラフィカルユーザインタフェースでの位置決め操作が緩和された到達運動における支援と分析

Supporting Techniques and Analyses for Coarse Reaching Movements in Graphical User Interfaces

学位請求者　新領域創造専攻

山中 祥太

要旨

本論文では，ポインティングとステアリングの微細さに関する研究成果について述べる．PC 作業における Graphical user interface (GUI)操作の負担が年々高まっていることを背景に，従来から多くの研究者たちが操作を支援する方法を検討してきた. マウスカーソルの移動時間を短縮する手法や，小さなターゲットを選択しやすくする手法など，様々な観点でポインティング操作を便利にする手法が提案されている．また階層メニューを展開するような，一定の幅からはみ出さないように移動する操作（ステアリング操作）についても同様に，素早くゴールまで到達するための支援手法が多く提案されている．

こういった作業の効率を定量的に評価する方法も多く研究されており，GUI 研究で一般的に用いられるのがパフォーマンスモデルと呼ばれるものである. これはターゲットのサイズや配置などからタスクの難易度を算出し，操作時間との関係を表せるようにしたモデルである．タスクの難易度をより正確に表せるモデルを構築することで, 支援手法による操作の改善度合いなどを定量的に比較できるようになる．

ポインティングとステアリングはまとめて到達運動と呼ばれ，GUI 操作における基礎的な動作である．到達運動には，ウィンドウの細長い縁を選択するように微細な制御を求められる場面もあれば，画面端に置かれたターゲットを選択するような大雑把な（粗い）制御で済む場面も存在する．また，移動し始めたときは粗い制御でよいが，終盤では微細に動かす必要がある，というように制御の微細さが操作中に変化していくこともある．本論文の目的は，これらの操作におけるユーザの挙動を分析・モデル化し，また操作を改善する手法を開発することである．具体的には，ポインティングではターゲットのサイズが小さいほど，ステアリングでは経路の幅が狭いほど微細なカーソル制御を求められるため，ユーザの負担が重くなってしまう．これを粗い操作で済むように支援することが目的である. また粗い制御に関する基礎的な検討が不十分なタスクも存在するため，これらは定量的な評価と議論を可能にするために，挙動の分析とモデル化を行う

第 1 章では，近年の PC 利用の広まりと，それに伴う GUI 操作の負担について述べ，本論文が対象とする到達運動の背景を説明する．そのうえで，負担の重さを操作時間などに置き換えて定量的に議論するためのパフォーマンスモデルについて概説する．

第 2 章では，到達運動に関わる先行研究を紹介し，本論文の位置づけを述べる．従来から提案されているパフォーマンスモデルや操作支援手法を整理したうえで, 本論文で取り組む具体的な研究対象を述べる．

第 3 章では，ターゲットサイズが非常に広大なポインティングタスクの分析に焦点を当てる．大きなターゲットのポインティングタスクは GUI 研究においてこれまで十分に分析されてこなかった．しかしながら，実際の PC 操作において日常的に行われるタスクであり，またサイズ以外に影響しうるパラメータも存在するため，きわめて大きなサイズのターゲット

をポインティングする動作を分析することも基礎的な検討として重要であると考える．本研究で取り扱うのは，ターゲットの奥行きサイズ W，およびそれと垂直な方向のサイズ H がともに無限大のターゲットである．いずれか一方のサイズが無限大の場合のパフォーマンスモデルは先行研究で提案されているが，両者が無限大のときには従来の分析方法を適用することができない．そこで本研究で独自に実験を行い，操作時間やカーソルの軌跡を分析することで，無限大サイズのターゲットをポインティングするときの挙動を明らかにする．

第 4 章では，ステアリングの法則の修正モデルについて検討する．ステアリングの法則を初めて提案した文献では，一定幅の直線状経路，幅の狭まる直線状経路，幅の広がる螺旋状経路の操作時間がモデル化されている．しかし，ステアリングの法則によって算出される難易度がうまく適合しない例があることが指摘されている．たとえば経路内にカーブがあると，ユーザはなるべくコーナーの内側を通ろうとし，モデルで予測される操作時間よりも実測の操作時間が短くなることがわかっている．そこで適切な難易度を算出できるような修正モデルが提案されており，これによって特定のタスクの難易度や，あるいは新しい操作支援手法によって難易度がどの程度改善されるのか，といった検討項目に対して適切な定量評価ができるようになる．本研究では，幅が変化する直線状経路を通過する時間が，通過する方向（幅が狭まる方向，広がる方向）によって変化することを発見した．この原因を追求することで難易度の差を算出するモデルを構築し，実験によってモデルの妥当性を検証することを目指す．

第 5 章では，細長いターゲットをドラッグアンドドロップするタスクを支援する手法について検討する．小さなターゲットのポインティング支援手法はこれまでに多く提案されており，それぞれが評価実験で一定の有用性を示している．しかし，ターゲットの形状によって操作時間が変わることや，ターゲットの配置間隔によって既存手法の有用性が低減することを考慮すると，タスクの内容によっては既存手法では支援が不十分になる可能性がある．現代の PC 作業において，ウィンドウをリサイズするための外枠や，ウィンドウ内のレイアウトを変更するための境界線，表計算ソフトのセルをリサイズするための境界線など，細長いターゲットは多く存在する．従来はターゲットを拡大する手法やカーソル速度を低下させる手法が多く提案されていたが，本研究では線分を通過することでドラッグを開始する手法を提案する．これはターゲット上にカーソルを乗せるための微細な制御が必要なく，線分のターゲットにカーソルを「ぶつける」ようにするだけでドラッグを開始できる．またカーソル速度の低下も起こらないため，タスク全体の操作時間を短縮することが期待される．しかしターゲットが短かったり配置間隔が狭いことで効果が低減する懸念があるため，実験によって有用性と限界を評価し，提案手法の有効範囲を明らかにする．

第 6 章では，視線情報を用いて長距離のカーソル移動を支援する手法について検討する．近年では視線計測器の低価格化が進み，標準搭載する PC が販売されるまでに至っている．そこでポインティングを支援するインフラとして視線計測器を使用し，注視点とカーソルが離れているときにのみカーソルを高速化させる手法を開発する．この手法であれば，ユーザが大雑把な制御をしたいポインティング開始時点の時間を短縮でき，その後にカーソルが注視点付近にあるときは通常の速度に戻るため，時間短縮と操作精度を両立できると考える．先行研究でも視線情報を利用してカーソルをジャンプさせるなどの支援方法が提案

されているが，不必要なタイミングでジャンプが発生する設計になっている不備がある．本研究の提案手法はそういった問題を解消しつつ操作時間を短縮するものである．視線情報を利用するにあたって，計測誤差や眼球の固視微動など，考慮すべき事項について整理する必要がある．そのうえで支援手法を設計し，評価実験によって有用性を確認する．

第 7 章では，本論文の貢献をまとめ，また将来の展望について述べる．まず総合的な議論として，各章で述べたテーマの学術的貢献を説明する．具体的には，分析やモデル化が不十分であれば基礎的な分析を行い，強固なモデルがある場合には操作支援手法を開発・評価したことである．これらをまとめて，ユーザの挙動を理解し，操作の負担を軽減するための知見を提供したことを，本論文で行った「粗い」到達運動に関する研究全体の学術的貢献として結論する．

目次

第1章　序論

1.1　研究の背景

1.1.1 コンピュータ操作の負担と支援

総務省が実施した調査 [Soumu, 2015]によれば，日本におけるパーソナルコンピュータ（以降, PC）の保有率は減少傾向にあり, 2014 年時点では 78.0%だったと報告されている．一方でスマートフォン (64.2%)やタブレット端末 (26.3%)の保有率は増加し続けていることも報告されており，従来的な「PC の前に座ってマウスとキーボードを操作する」などとは異なるコンピュータの利用形態が広まっていることを示している．しかしながら，依然として PC の保有率は他の情報端末よりも高い割合である．PC や携帯機器の主な利用目的であるインターネット接続に関しても，使用率は自宅の PC が最も高く (53.5%)，スマートフォン (47.1%)やタブレット (14.8%)をしのぐ．したがって，現在もスマートフォンやタブレットは PC に取って代わるものではなく，従来型の利用形態が根強いことがうかがえる．

同一の資料によれば，平日のインターネットの利用時間は，2012 年には平均 71.6 分だったのに対し，2014 年には平均 83.6 分と，16.8%ほど増加している．PC や携帯機器の利用時間が増大したことで発生した問題として，PC ではマウス操作による腱鞘炎 [Fagarasanu, 2003]，スマートフォンやタブレットでは指を何度も往復させることによる腱鞘炎 [Vogel, 2012]の症例が報告されている．PC 利用に着目すると，近年では高解像度・大型のディスプレイが安価に入手でき，またディスプレイを複数枚並べて作業する環境（マルチディスプレイ）を利用するユーザもいることから，広範囲にわたってマウスカーソルを移動させる場面が多いといえる（図 1.1）．その結果，マウスなどのポインティングデバイスの操作量（操作時間や移動距離）は増大し，ユーザへの負担は重くなってきている．こういった長距離のポインティング操作の負担を軽減するために, 従来から Graphical user interface（以降, GUI）におけるポインティング支援手法が多く研究されているが，その必要性は今後ますます高まっていくと考えられる．

PC 操作におけるポインティング支援は，長距離の移動のみならず，ターゲット付近でカーソルを微細に制御しなければならない場面でも必要である．選択したいターゲットのサイズが小さいほどポインティング操作の難易度は上昇する（図 1.2）が，一般的な GUI 環境には小さいターゲットがいくつも配置されている．特に高齢者が小さなターゲットを選択するときには，若年者の 2 倍以上の時間を要してしまう [Worden, 1997]ことから，微細なポインティング操作支援は幅広い年代にとって必要であると考えられる．

長距離のポインティング支援と同様に，小さなターゲットの選択支援手法も従来から開発されており，実験によってその有用性が確認されている．しかし，こういったポインティング支援手法の多くは，ボタンやアイコンなどの矩形・円型ターゲットを対象としたものである．ポインティング支援手法の性能はターゲットの形状や配置密度などの要因で変化するため，タスクによっては有用性が低減するおそれがある．また，PC 操作ではポインティングだけでなく，ターゲットを別の場所へ移動するドラッグアンドドロップ (DnD)操作も多く行われる．DnD はポインティングよりも指を緊張させたままポインティングデバイスを移動させなければならず，操作ミスが増大することが知られている [MacKenzie, 1991]．したがって，既存のポインティング支援手法の有用性について，ターゲットの形状や配置密度が変化した場合，また DnD 操作に適用した場合について今一度検討しなければならないと考える．そのうえで，既存手法の有用性が低減する場合においても有効に機能する，新たな支援手法を構築するべきである．

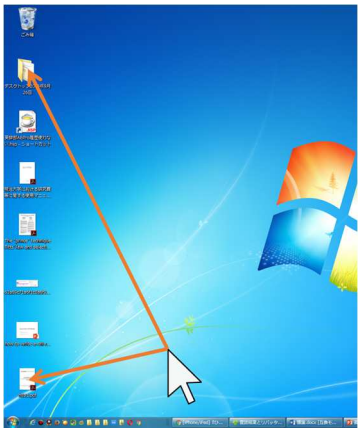


図 1.1　ターゲットまでの距離による選択時間の違い．ターゲットのサイズが同一であれば，カーソルからの距離が遠いほどポインティングに時間を要する．

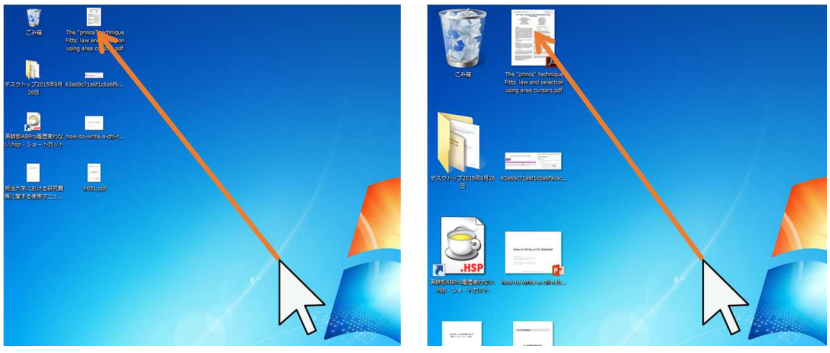


図 1.2　ターゲットのサイズによる選択しやすさの違い．同じ距離にあるアイコンでも，サイズを「小」（左）にすると，「大」（右）よりもカーソルの位置調整に時間を要する．

1.1.2 到達運動とパフォーマンスモデルの概要

1.1.2.1 フィッツの法則とステアリングの法則

第 1.1.1 章では PC 操作におけるポインティング動作について述べ，負担が大きな長距離移動，および小さいサイズのターゲット選択の問題について考えた．こういった問題を解消する方法を検討するにあたり，当然ながら定量的な議論は必須である．すなわち，ターゲットまでの距離やサイズによる難易度差や負担度といった項目を，実験を通して計測できるように数値化して評価・比較しなければならない．たとえば負担の軽減については，操作時間の短縮度合いで評価されるのが一般的である．

こういった定量評価のために有用なのが，タスクの難易度と所要時間の関係を数式化したパフォーマンスモデルである．広く知られたモデルの例として，ターゲットをポインティングする難易度と，選択するまでの時間の関係をモデル化したフィッツの法則 [Fitts, 1954]がある．これはターゲットのサイズが小さいほど，また距離が遠いほど難易度が高くなり，操作時間が増大することを表したモデルである．フィッツの法則を用いることで,「ターゲット A よりも B を選択する方が時間がかかる」などと分析したり，あるいは新しいポインティング支援手法に対して「この手法を用いると難易度がこれだけ低くなる」などと理論的な分析が可能になる．

フィッツの法則はターゲットを選択する動作に着目したモデルであるが，カーソル移動の連続的な動作を扱ったモデルも提案されている．これはステアリングの法則 [Accot, 1997]と呼ばれ，幅の決まった経路からカーソルがはみ出ないように移動するタスク（ステアリング操作と呼ばれる．図 1.3）をモデル化したものである．ステアリングの法則を用いると，経路の長さと幅から難易度を数値化できる．ポインティングが離散的な地点での操作精度を要するのに対して，ステアリングは連続的な地点での操作精度を要する点で異なる動作である．つまりポインティングがターゲット選択に至るまでの動作を考慮しないのに対して，ステアリングはスタートからゴールまで常に経路の幅から出ないように制御し続ける必要がある．PC 操作におけるステアリングタスクの例として，階層メニューの展開操作（図 1.4 左）や，ドローツールではみ出さないように色を塗る操作などがある．また，手本通りに文字を書く動作もステアリングタスクの一種であり，字の学習アプリ（図 1.4 右）の難易度もステアリングの法則で定量化できる．

ポインティングやステアリングなど，目標地点へ手を移動する動作は到達運動 (reaching movements)と呼ばれ，心理学分野で 19 世紀頃から盛んに研究されている（歴史的なレビューは文献 [Takashima, 2008]が詳しい）．GUI 操作における到達運動には DnD も含まれるが，これはポインティング操作の一種と捉えられる [MacKenzie, 1991]．また目標地点まで画面

を遷移させるスクローリング，目標物が適切な大きさで表示されるように拡大/縮小するズーミングなども到達運動に含まれるが，スクロールバー操作や拡大率変更はポインティング・ステアリングの組み合わせで実現される．したがって，本論文で到達運動と呼ぶときにはポインティングとステアリングの 2 種類を指すものとする．

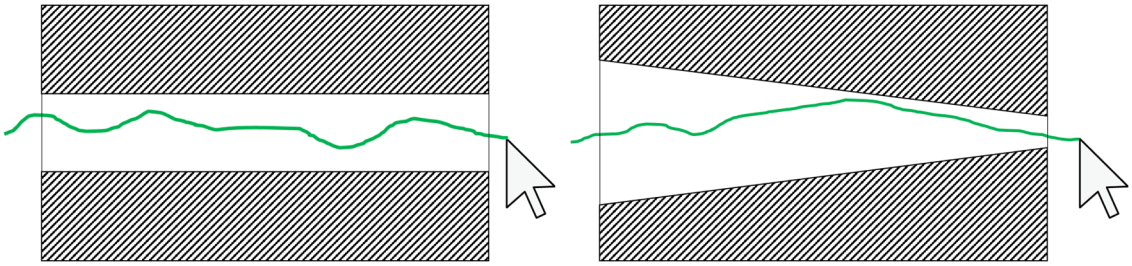


図 1.3　ステアリング法則でモデル化できるタスクの例．
（左）一定幅の直線状経路．（右）幅の狭まる直線状経路．

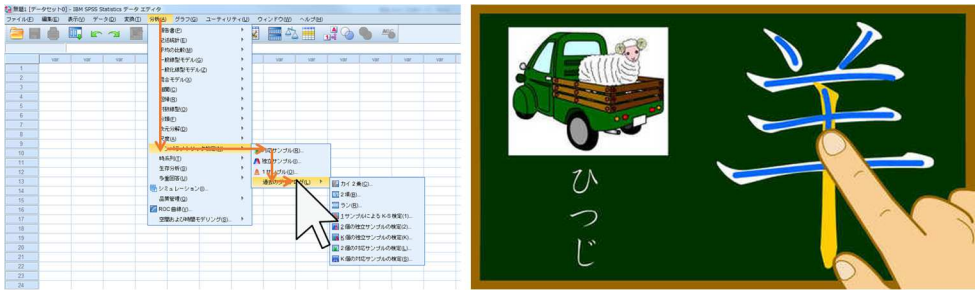


図 1.4　ステアリングタスクの例．（左）階層メニューを展開する操作．
（右）手本の字をなぞる操作（iPad 用アプリ「三年生の漢字」
http://damemoto.lolipop.jp/damemoto/damemoto.php より引用）．

1.1.2.2 モデルの適用可能性と修正

フィッツの法則やステアリングの法則は多くの追実験が行われ，ポインティングデバイスやディスプレイサイズを変更してもモデルがフィットする（数式が適合する）ことが知られている．しかし，モデルがうまくフィットしない条件も少なからず存在し，特にフィッツの法則はこれまでに何度も修正モデルが提案されている．例として，長方形のターゲットに対して斜め方向からカーソルが進入する場合のモデル [Accot, 2003]や，指でタッチスクリーンを直接操作する場合のモデル [Bi, 2013]などがある．こういった補正を加えることでタスクの難易度をより正確に定量化できるようになる．

ところが GUI 操作において，そもそもフィッツの法則を適用するのが不適切なけるポインティングタスクが存在する．フィッツの法則は，カーソルがターゲットをオーバーシュ

ート（行き過ぎること）してもミスになり，かつアンダーシュート（止まるのが手前すぎること）してもミスになる状況のパフォーマンスモデルである．したがって，ターゲットが画面端に設置されているような状況のポインティング操作にはそのまま適用できないのである．なぜなら，カーソルが画面端に到達すると，それ以上ポインティングデバイスを移動させてもカーソルは引っかかったままになるためである．つまり画面端にあるターゲットは奥行き方向に擬似的な無限大のサイズをもっていると見なせるため，オーバーシュートによるミスが定義されない．実際に，ターゲットが画面端にあるか否かで操作時間と難易度の関係が異なることが判明している [Appert, 2008].

文献 [Appert, 2008]の実験は画面端のアイコンやボタンなどを対象にしており，奥行き方向のオーバーシュートはしないものの，ある程度の位置調整が必要なタスクが想定されている．これに対して，ウィンドウ内からデスクトップにファイルを DnD する操作などを考えると，ウィンドウから出た後には微細な位置調整が必要ないことがわかる．つまりターゲットのサイズが二次元方向に無限大と見なすことができるが，こういった操作におけるパフォーマンスモデルは現在まで提案されておらず，タスクのパラメータと操作時間の間にどのような関係があるか不明である．操作自体はいたって容易であり，特別な支援を必要としないためか，他のポインティングタスクよりも注目されにくい条件であるといえる．しかしながら，大雑把な制御で済む操作も日常的に行われ，ユーザがどのような挙動を示すのかを調査するのは重要であると考える．

一方のステアリングの法則は比較的新しく提案されたこともあり，修正モデルは少ない．モデルの修正案を構築するには，まず既存のモデルがうまくフィットしない場面を発見し，その状況下でフィットしない理由を分析し，その条件を埋める・吸収するように補正できるモデルを構築しなければならない．一般的な GUI 環境ではステアリング操作を求められる場面がポインティングより少なく，フィットしない場面がこれまであまり発見されなかったのではないかと推測する．

実際にはステアリングの法則をそのまま適用すると不適切な難易度が算出される事例があり，難易度と操作時間の関係を正確に表すにはモデルの修正が必要である．これまでに発見された例として，曲がり角を含む経路を通過するタスクや，腕を大きく動かすほど大きな経路をスタイラス操作で通過するタスクなどがある．こういったタスクの難易度を正確に算出できるようにモデルを修正することで，GUI 操作における定量評価がより盤石なものとなる．

以上をまとめると，基礎的な分析すら十分に行われていないタスクについては，まずユーザがどのような操作をするのか観察することが必要である．また，パフォーマンスモデルが構築されていなかったり，既存のモデルでは正確に難易度を算出できないタスクがあればモデルを修正する必要がある．パフォーマンスモデルの正確さが十分に検証されているのであれば，あとはユーザが PC 作業時に不便を強いられている状況をピックアップし，その操作を支援する手法を提案すべきである．

1.1.2.3 粗い到達運動

ウィンドウの縁のように幅が 1〜10 pixels 程度しかないターゲットをポインティングするには，マウスカーソルの位置を慎重に決めてからボタンを押す必要がある．一方で，ウィンドウ内のファイルをデスクトップの広大なスペースに一旦移動しておく操作では，ドロップ位置を決めるためのカーソル制御は慎重にする必要がない．このように，ポインティング操作においてカーソルを細かく制御する必要があるのか，あるいは大雑把に動かすだけでよいのかという粒度に関して，本論文ではそれぞれ「微細な制御」,「粗い制御」と呼ぶことにする．

ポインティングは 1 回の操作に微細な制御と粗い制御が混在していることが知られている．ある程度距離が離れた位置にあるターゲットを選択するときに，まずはカーソルを勢いよく動かして距離を詰め，その後に時間をかけてターゲット上にカーソルを乗せる挙動をするのである．つまりポインティングでは初めに粗い制御，終盤にかけて微細な制御をする．しかしこれはデスクトップアイコンやボタンなどを操作する場合であり，ウィンドウの縁やデスクトップ領域などがターゲットであれば基本的な挙動や必要な操作支援方法が変わってくると考えられる．

ステアリングは移動中の制御の微細さが求められる操作であり，一定幅の経路（図 1.3 の左）であればスタートからゴールまで一定の微細さを保つ必要がある．幅が狭まる経路（図 1.3 の右）であれば，初めは粗い制御でよく，徐々に微細さを増していく必要がある．このように，本論文ではステアリングに関してもカーソル制御の細かさについて粗い/微細な，と表現することにする．

本論文では，第一にユーザの負担を軽減するために粗い制御で操作が済むような支援手法を開発することを目指す．これはフィッツの法則から考えて，ターゲットのサイズが小さい場合と，移動距離が長い場合の操作時間を短縮することで目的を達成する．つまり，サイズが小さくでも快適に選択できるようにし，また長距離の移動は高速化しつつも終盤の位置決めを阻害しないようなポインティング支援手法を検討する．第二に，粗い到達運動において挙動の分析やモデル化が不十分なものについては，定量的な評価を行うために基礎的な分析とパフォーマンスモデルの構築をする．具体的には，ターゲットサイズが無限大で微細な制御が求められないポインティングタスクの挙動を分析する．またステアリングに関して，幅が徐々に狭まる経路と，逆に広がっていく経路の難易度差をモデル化する．狭まる経路はポインティングのように粗い制御→微細な制御の順に操作粒度が変化していくが，広がる経路ではその逆である．従来のモデルではこのような幅の変化する経路の難易度差を正確に算出できず，定量的な評価が正確にできない問題があったため，これを解消することを目指す．

1.2　本論文の構成

本論文は全 7 章で構成される．第 2 章以降の構成を以下に示す．

第 2 章

ポインティングとステアリングに関わる先行研究を整理する．パフォーマンスモデルに関わる研究は，フィッツの法則とステアリングの法則の概要と，これらの修正モデルを紹介する．また操作支援手法に関わる研究について，これまでに提案されている手法をパフォーマンスモデルに基づいて分類して紹介する．そして先行研究で既に十分な分析・支援が達成されている部分と，未だに不十分だと思われる部分を整理し，本論文で取り組むべき具体的な研究テーマを設定する．

第 3 章

無限大のサイズをもつターゲットのポインティングタスクを分析した結果について述べる．ユーザの操作時の挙動を分析し，サイズの規定された一般的なターゲットポインティングと比較議論する．また，タスクのパラメータと操作時間の関係を表す予測式についても検討する．本章は関連発表番号[1]の文献を改訂したものである．

第 4 章

幅の変化する経路を逆向きに通過するタスクのモデルについて述べる．従来のステアリングの法則では正確に難易度を算出できなかった事例があることを指摘し，従来研究を補完するモデルを提案する．また実験によって提案モデルの妥当性を検証した結果についても報告する．本章は関連発表番号[2] [6]の文献を改訂したものである．

第 5 章

細長いターゲットの DnD を支援する手法について述べる．ウィンドウの縁などを通過する（クロッシングする）ことで選択する手法の有用性を検討し，さらに実験によって操作時間が短縮されるか検証した結果について報告する．本章は関連発表番号[3] [8] [9]の文献を改訂したものである．

第 6 章

視線情報を利用して長距離移動を支援する手法について述べる．ユーザがカーソルを注視していないときに高速化する手法の有用性を検討し，ポインティングタスクにおける操作時間短縮を検証した結果について報告する．本章は関連発表番号[5] [10]の文献を改訂したものである．

第 7 章

本論文の成果と貢献についてまとめ，さらに今後の展望を述べる．

第2章　ポインティングとステアリングに関する先行研究と本研究の位置づけ

本章ではポインティングとステアリングに関する先行研究を整理する．まずそれぞれの操作に関する基礎的な分析やパフォーマンスモデルについてまとめる．また，各操作を支援する手法を整理し，既に十分な支援がなされているタスクや，逆に支援が不十分だと思われるタスクをピックアップする．そのうえで，本研究で取り組む具体的な研究対象を述べる．

2.1　パフォーマンスモデル

2.1.1 ターゲット選択

2.1.1.1 ポインティング

GUI におけるポインティング操作はフィッツの法則 [Fitts, 1954]を用いて分析されるのが一般的である．フィッツの法則によれば，図 2.1 のように距離 A だけ離れた位置にある幅 W のターゲットをポインティングする時間は式(2.1)で表せる．

$$MT = a + b\log_2\left(\frac{A}{W} + 1\right) \tag{2.1}$$

a と b は実験条件によって決まる定数である. また対数項は難易度指標 ID (Index of difficulty) と呼ばれる．

$$ID = \log_2\left(\frac{A}{W} + 1\right) \tag{2.2}$$

フィッツの法則は，ターゲットが遠くにあり（A が大きく），またターゲットサイズが小さい（W が小さい）ほどポインティングが困難になることを示している．フィッツの法則は元々は 1 次元のポインティング動作をモデル化したものであり，カーソルの移動と垂直な方向のサイズは考慮しない．以下では文献 [Accot, 2003]にならって，カーソルの進行方向に沿ったサイズを W，垂直な方向のサイズを H とする（図 2.1）．

一般的な GUI におけるターゲット（ボタンやアイコンなど）を選択するときには，進行方向に対して奥行き方向のサイズ W のみを考慮するわけにはいかず，垂直な方向にも注意

を払ってカーソルを移動しなければならない．こういった現実的なタスクにフィッツの法則を適用するために，複数の修正モデルが提案されている．MacKenzie によれば，ターゲットへの進入角度が変化する場合に，W と H のうち小さい方をターゲットサイズに採用すると難易度が適切に表せると報告されている [MacKenzie, 1992]（図 2.2）．ここで「難易度が適切に表せる」とは，難易度の値と操作時間の関係が線形に近づき，モデルがフィットすることを指す．つまり，式(2.3)のように ID を算出するのがよいとされている．

$$ID = \log_2\left(\frac{A}{\min(W,H)} + 1\right) \tag{2.3}$$

これに対し，Accot らは W と H が近い値になったときには難易度が適切に表せなくなる問題を指摘し，実測の操作時間によりフィットする式(2.4)のモデルを提案した [Accot, 2003].

$$ID = \log_2\left(\sqrt{\left(\frac{A}{W}\right)^2 + \eta\left(\frac{A}{H}\right)^2} + 1\right) \tag{2.4}$$

η は自由な値をとる重み (Free weight)であり，実験では $\eta = 0.137$ と算出されていた．仮に $\eta = 0$ であれば式(2.4)は元のフィッツの法則と一致する．この式はつまり，ポインティングの難易度は，カーソルのメインの移動方向のサイズ W の影響を受けつつ，それと垂直な方向のサイズ H の影響をわずかに受ける，ということを示している．

これらの他にも, 3 次元空間中のポインティングに適用させたモデル [Grossman, 2004]や, 指でタッチスクリーンを操作するときのずれを補正するモデル [Bi, 2013]などがあり，特定の操作状況に合わせて修正されたモデルが提案されている．

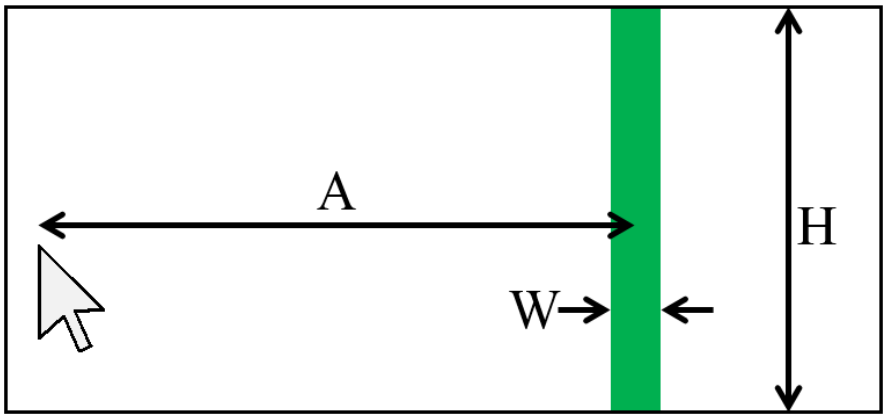


図 2.1　フィッツの法則で分析できるポインティングタスク．

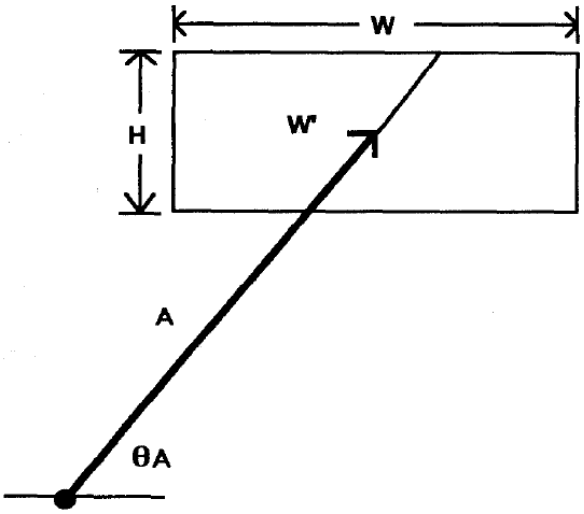


図 2.2　MacKenzie の実験のタスク [MacKenzie, 1992].
進入角度 θ_A は 0°，45°，90°の 3 種類が採用された.

2.1.1.2 クロッシング

図 2.3 のように，幅 W の線分ターゲットが距離 A だけ離れて 2 本置かれているとき，これらを通過する時間はフィッツの法則と同じ式(2.1)で表せることがわかっている [Accot, 1997] [Accot, 2002]. この実験は間接制御タイプのスタイラス操作で行われており，途中で一旦ペン先を浮かせる条件や，ターゲットを 90°だけ傾けた場合も同様に式(2.1)がフィットすることが確かめられている．また，W や A の値によってはポインティングよりも操作時間やエラー率が改善されることも知られている [Accot, 2002].

他にも，トラックボールを用いた実験 [Wobbrock, 2008-a]や，直接制御タイプのスタイラスでの実験 [Forlines, 2008]，指での直接タッチによる実験 [Luo, 2014]なども行われており，いずれもポインティングより高速あるいは低エラー率であると報告されている.

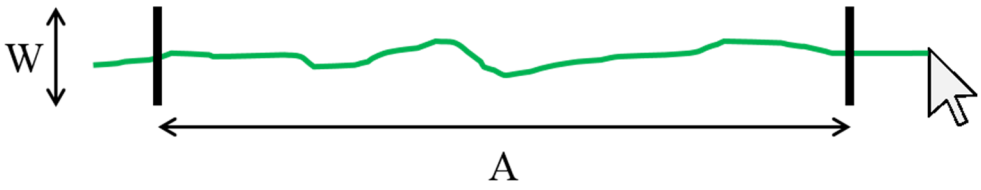


図 2.3　2 本の線分を通過するクロッシングタスク.

2.1.1.3 画面端のポインティング

画面端に置かれたターゲットをポインティングする場合，そのターゲットはサイズが無限大と見なせる．なぜなら，画面端ではカーソルが引っかかってそれ以上ポインティングデバイスを画面外側へ動かしてもカーソルが移動しないため，奥行き方向に無限大のサイズをもっているのと同等なためである．この事象については従来からウェブサイト [Tognazzini, 1999]や学術論文 [Appert, 2008] [Huot, 2011]，書籍 [Raskin, 2003]などで言及されている．無限大のサイズをもった GUI 部品の例として，Mac のメニューバーや Windows のタスクバーなどがある．こういったターゲットは図 2.4 のように H が有限，W が奥行き方向に無限大であると見なすことができ，本来はフィッツの法則の対象とされるタスクではないものの，フィッツの法則と同様の式(2.5)で難易度を表せることがわかっている [Accot, 2002].

$$ID = \log_2\left(\frac{A}{H} + 1\right) \tag{2.5}$$

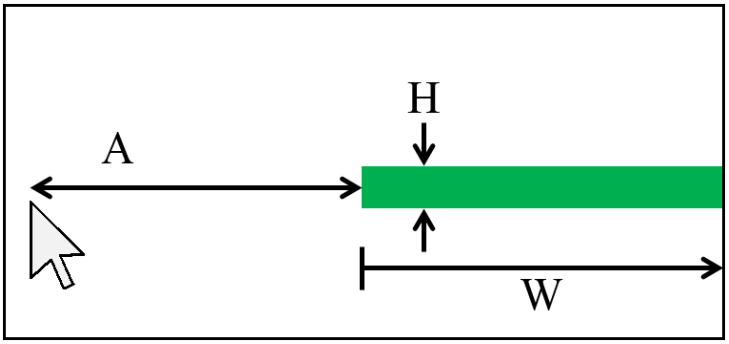


図 2.4　奥行きサイズ W が無限大，それと垂直なサイズ H が有限のターゲット．

Appert らはさらに複雑な条件下で実験しており，カーソルがターゲットに対して斜めに進入するときの角度や，カーソルの形状，ターゲット上にカーソルが乗ったときの視覚フィードバックなどの影響を詳細に分析している [Appert, 2008]．Appert は画面端のターゲットをポインティングするタスクをエッジポインティングと呼んでいる．この実験の中で興味深いのは，画面端のターゲットは奥行き方向のサイズが無限大と見なせるにも関わらず，ターゲットが表示されているサイズによって異なる操作時間になったことである（図 2.5）．また，難易度指標 ID は式(2.6)のように複雑な形式になることが示されている．エッジポインティングは，定数 a，b の値が通常のポインティングと大きく異なるため，ユーザの戦略が元々違うものであると考察されている．

$$ID = \log_2\left(\frac{A}{W} + \frac{A}{H} + 0.6 \times \sin(|Angle|) \times \frac{A}{\min(W,H)} + 1\right) \quad (2.6)$$

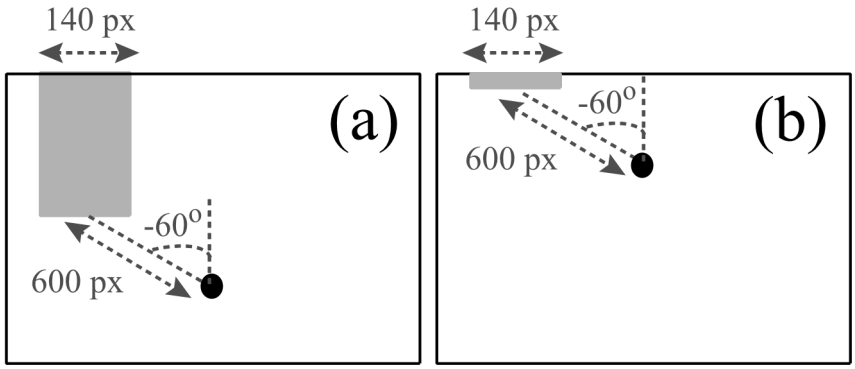


図 2.5 [Appert, 2008]より引用．距離や進入角度を変えて実験された結果，奥行き方向の描画サイズが 320 pixels (a)の方が 20 pixels (b)より操作時間が短いと報告されている．

Farris らも Appert らと同様にエッジポインティングの操作時間について分析しており，ターゲットへの進入角度は斜め直角の方が短時間であること [Farris, 2002-a]や，ターゲットまでの距離 A が短い方が短時間であること [Farris, 2002-b]，さらにターゲットが表示されている領域のサイズが大きいほど短時間であること [Farris, 2003]を確かめている．

2.1.2 ステアリングの法則

GUI 環境において軌道に沿う動きを求められることは多い．階層メニューから目的の項目を選んでいく操作などがこれに該当する．このような 2 次元方向に制約の課された動作をモデル化するために，Accot らによってステアリングの法則 [Accot, 1997]が提案された．これは図 2.6 のような幅の決まった経路を通過する時間が式(2.7)で表せるというものである．

$$MT = a + b \times ID, \quad ID = \frac{A}{W} \quad (2.7)$$

この実験は，ディスプレイ上のカーソルを机上のスタイラス操作で移動させる間接制御方式で行われている．また様々な環境下でこの法則が成立することが示されており，デバイスの変更 [Accot, 1999]，ポインティングデバイスの移動量に対するカーソル移動量（CD ゲイン）の変更 [Accot, 2001]，スタイラスを入力面から浮かせたまま動かすホバー操作 [Kattinakere, 2007]，カーソルサイズの変更 [Naito, 2004] [Naito, 2006]，スタイラス入力面の摩擦度合いの変更 [Sun, 2012]などの条件で実験が行われている．2 次元の GUI 環境以外では，3 次元入力 [Casiez, 2004] [Liu, 2011]や，カーソルを自動車に置き換えたドライブシミュレータ [Zhai, 2004]にもモデルを適用できることが示されている．またドライブシミュレー

タは図 2.7 のように視点によって操作時間に差が生じることがわかっている [Bateman, 2011].

ステアリングの法則は，一定幅の直線状経路以外にも，一定幅の環状経路 [Accot, 1999] [Accot, 2001]に適用可能であることがわかっている．また幅が徐々に狭まっていく直線状の経路，幅が広がっていく螺旋状の経路の操作時間もモデル化されている [Accot, 1997]．ステアリングの法則の導出過程は第 4 章で詳細に述べる．

モデルの補正に関する議論もされている．経路の幅 *W* が広すぎると，実際に使われる幅（実効幅）の割合が低くなるため，これを考慮して *ID* を算出するモデルが提案されている [Kulikov, 2005]．この補正モデルでは，カーソルが経路から出てしまった後もタスクを継続して実効幅を算出すればモデルが適用できることが示されている．また，曲がり角を含む経路の通過時間を予測する補正方法も検証されており [Pastel, 2006]，より複雑なタスクを想定したモデルが構築されている．

ポインティングとステアリングが複合したタスクの操作時間を予測する方法も提案されている [Dennerlein , 2000] [Kulikov, 2006]．たとえば階層メニューでは，最後に 1 つの項目をポインティングする操作が加わり，こういった操作の時間を既存のモデルの組み合わせで予測できるか検証することは重要である．また操作時間に制約を設けた場合のエラー率を検証する実験も行われており [Zhou, 2009]，ステアリングタスクにおいて時間と正確さはトレードオフであることが示されている．

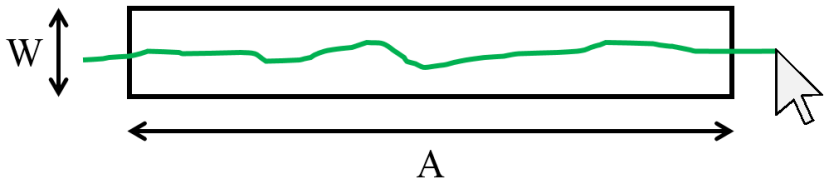


図 2.6　一定幅の直線状経路を通過するタスク．

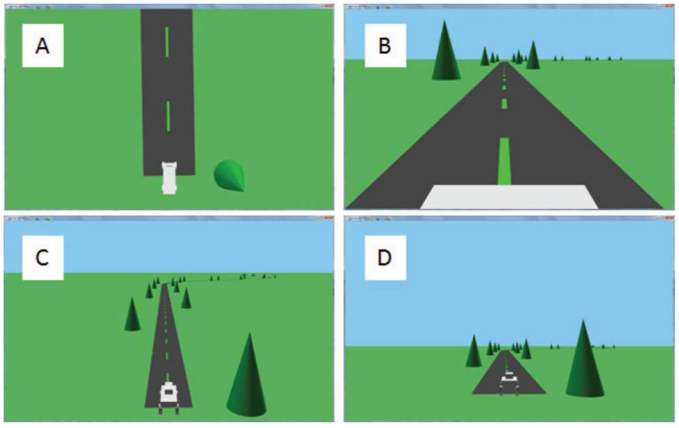


図 2.7　ドライブシミュレータにおける複数の視点．(A) Overhead，(B) Fitst-person，（C）Third-high，（D）Third-low．（A）が最も遅く，それ以外に有意差はない．

2.2 操作支援手法

2.2.1 ポインティング支援手法

フィッツの法則における距離 A を短縮できれば，タスクの難易度が低下し，操作時間を短縮できる．またサイズ W を大きくすれば同様に操作時間を短縮可能である．これらのどちらか一方，あるいは両方を実現することでポインティングを支援する手法が数多く提案されている．以下では，これら 3 種類に分類してポインティング支援手法を整理する．

2.2.1.1 ターゲット距離 A を短縮する手法

カーソルやターゲットの移動

カーソルをターゲット付近までジャンプさせることで移動距離を短縮する手法が多く提案されている．Object pointing [Guiard, 2004] は，ユーザが次に選択したいであろうターゲットを推測し，そこまでカーソルをジャンプさせる手法である．これはカーソルがターゲットから出て行く方向を基にして次のターゲットを決定している．Delphian desktop [Asano, 2005] は，カーソル移動のピーク速度によって目標地点を予測し，カーソルをジャンプさせる手法である．ピーク速度だけでなく，より長い時間の速度波形をテンプレートマッチングすることで終点を予測する手法も提案されている [Pasqual, 2014]．TorusDesktop [Huot, 2011] ではカーソルを画面端に押しつけ続けると反対端から出てくる手法の操作効率を調査している．

MAGIC [Zhai, 1999] は視線とマウスを切り替えてポインティングする手法である．マウスを動かした瞬間にカーソルを注視点付近までジャンプさせ，そこからターゲットまではマウス操作によって移動させる方法である. 大和らの提案するターゲット選択手法 [Yamato, 2001] も MAGIC と同様に，注視点にカーソルを移動させ，最後のターゲット選択はマウスで行うものである．これらの手法は，長距離移動は高速な視線で行い，微細な制御が求められるターゲット付近ではマウス操作でポインティングすることで，高速化と高精度化を両立している．

ターゲット側を移動する手法に Drag-and-Pop [Baudisch, 2003]や Drag-and-Guess [Nishida, 2007] がある．これらは DnD 操作において，ドラッグし始めたターゲットの属性などからドロップ先のターゲットを予測し，カーソル付近まで引き寄せる手法である．適用できる操作が DnD に限定されるものの，予測されたドロップ先が不適切であればそのまま本来のターゲットまで移動すればよいという利点をもつ.

カーソルをジャンプさせる手法の問題点として，視線を利用したジャンプ [Zhai, 1999]では視線計測器の計測誤差の問題がつきまとい，またカーソルの動きによる予測を利用したジャンプ [Guiard, 2004] [Asano, 2005]には予測誤差の影響が生じる．例としてピーク速度を利用したジャンプ [Asano, 2005]では，ユーザが弧を描くようにマウスを動かすため，遠いターゲットを狙うときほど角度の誤差が大きくなったと報告されている．この誤差を避けるために慎重に操作しようとすれば，今度は熟慮してからでないとマウスを動かせなくなってしまう問題が発生する [Kurihara, 2009]．これらのことを考慮すると，移動距離は確実に短縮できるものの，今まで存在しなかった新たな問題を生んでしまっているといえる．ゆえにカーソルをターゲット付近まで瞬時に移動させる手法はユーザにとってやや過剰な支援であると考える．

マルチカーソル

Ninja Cursors [Kobayashi, 2008] は，複数のカーソルを 1 つのマウスで同時に操作し，ターゲットから最も近いカーソルで選択する手法である．複数のターゲットを同時に選択してしまう問題については，ターゲットの領域に侵入可能なカーソルを 1 個に限定し，他のカーソルはターゲット手前で待機させることで対処している．この待機問題を解決するために, Ninja Cursors に視線計測器を導入する手法が提案されている [Blanch, 2009] [Räihä, 2009]. 注視点に最も近いカーソルをアクティブにすることで，カーソルが待機する必要がなくなり，より高速な選択が可能になる．さらに，1 つのターゲットにつき 1 つのカーソルを割り当て，常にカーソルの配置を最適化する Satellite cursor も提案されている [Yu, 2010].

マルチカーソルの課題として, 最適なカーソル数と, その配置の問題がある. Ninja Cursors ではカーソルが多すぎると待機が多く発生してしまい，操作時間がシングルカーソルよりも増大してしまうことが報告されている [Kobayashi, 2008]. また視線情報を導入した場合でも，カーソル数を増やしすぎると，結局は視線でポインティングするのと同等になってしまう．

以上はマルチカーソルの各手法の問題点を挙げたが，より根本的な問題として，総合的な作業時間が短縮されないことが指摘されている [Quinn, 2013]．これはポインティングの実験における「スタートボタンをクリックしてからターゲットをクリックするまでの時間」はマルチカーソルによって短縮されるものの，複数のカーソルの中からターゲットに最も近いものを吟味する時間で相殺されるという知見である．カーソルを画面の反対端へジャンプさせる TorusDesktop [Huot, 2011]についても同様であり，ターゲットを直接狙うか反対側から狙うかを考える時間によって，作業全体の時間は短くならないと報告されている．ユーザの運動量の軽減だけが目的であればマルチカーソルを採用するメリットはあるが，作業時間の短縮には結びつかない欠点を抱えている．

ウィンドウ操作に関わる移動距離の短縮

ウィンドウ操作の負担を軽減することでポインティングタスクを支援する研究も多い．ウィンドウ同士をドッキングして一括移動する手法 [Beaudouin-Lafon, 2001]や，オーバーラ

ップしたウィンドウをめくるようにして前後のウィンドウ間の DnD を実現する手法 [Dragicevic, 2004], カーソルをウィンドウの奥へ潜りこませる手法（関連発表番号[3] [6] [10] [11] [12]）などが提案されている．GUI 研究においてウィンドウ操作も重要なテーマであるが，本論文ではより基礎的なターゲットポインティングに焦点を当てることにする．

2.2.1.2 ターゲットサイズ *W* を拡大する手法

W を実際に拡大したり，あるいはそれと同等の効果を得るための手法も多い．カーソルが近づいたときにターゲットを拡大する手法は，ポインティング動作の終盤になってから拡大し始めても時間短縮の効果があるとされる [McGuffin, 2002]．ただし，これは Mac OS の Dock のように，ターゲットがどの程度拡大されるかをユーザが予め知っていなければならない [Zhai, 2003]．また，目的のターゲットが初期位置から移動してしまうことで運動計画が崩れたり，近辺のターゲットが視認できなくなる問題がある（図 2.8）．Sticky Icons [Worden, 1997]は，ターゲット上でカーソル速度を低下させることで擬似的に *W* を大きくする手法である．これはターゲットが多く配置されていると過度に速度減少が起こってしまい，操作時間が増大すると指摘されている [Tsukitani, 2011]．これに対して Birdlime Icon [Tsukitani, 2011]は，カーソルがターゲット上を通過するときのみターゲットを拡大する手法であり，ターゲットが多く配置されていても操作時間が増大しない．

カーソルの選択可能範囲を拡大することで操作時間を短縮する手法にエリアカーソル [Kabbash, 1995]がある．これはカーソルが 1 点を指すものではなく，ある程度の範囲（矩形）を指すものにする手法である．また，選択操作が困難な移動中にだけエリアを拡大する DynaSpot [Chapuis, 2009]も提案されている．

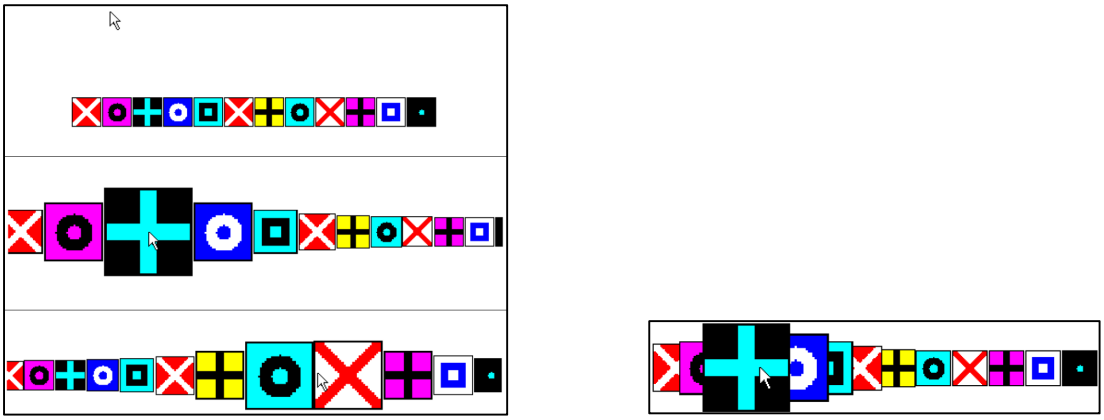


図 2.8　カーソルとの距離に応じてターゲットを拡大する手法．左のように，拡大されたターゲット同士が重ならないようにすると，ターゲットが初期位置から移動してしまって運動計画が崩れる．右のように重なることを許すと，ターゲットの視認性が低下する．

2.2.1.3 距離 *A* の短縮とサイズ *W* の拡大をする手法

ターゲットとの距離に応じてカーソル速度を変更する Semantic Pointing [Blanch, 2004]は，Sticky Icons のようにカーソルがターゲット上にあるときはカーソル速度を下げ，逆に何もないところでは高速化する手法である．これはつまり，マウスの移動量に対するカーソルの移動量（CD ゲイン）を動的に変更することで，長距離移動の時間を短縮することと，ターゲット付近での微細な制御を両立する手法である．ターゲットがない領域ではユーザが微細な制御をする必要がない，という現実的なニーズを考慮した手法であるといえる．これも Sticky Icons と同様に，ターゲットが密集している環境では常にカーソルが低速になってしまい，支援なしの場合よりも操作時間が増大するといわれている [Tsukitani, 2011].

Bubble Cursor [Grossman, 2005]（図 2.9）はエリアカーソルの発展版であり，カーソルが直近のターゲットを捕捉した状態を保ち，マウスクリックによって選択する手法である．ここで，「選択」とはターゲットを決定する操作を指し，「捕捉」とはエリアカーソルで選択可能状態にあることを指すものとする．Bubble Cursor を含めたエリアカーソルの問題の 1 つに，ターゲットを捕捉するための最短経路が認識しづらいことが挙げられる．たとえば Bubble Cursor を使用すると，どこまで近づけば目的のターゲットがエリア内に入るか分かりづらいため，遠くから捕捉できる機能を活かせないと指摘されている [Kuwabara, 2011].

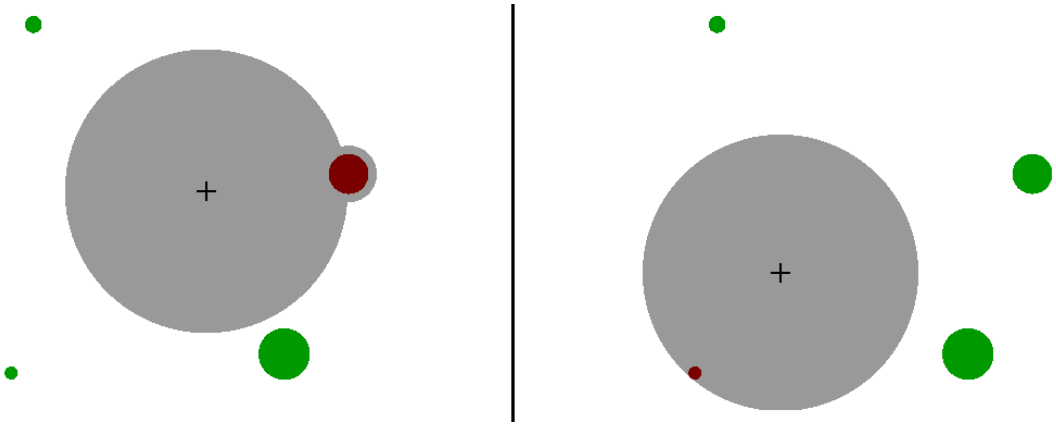


図 2.9　Bubble Cursor のデモシステム．カーソルの中心から最も近いターゲットを常に捕捉しており，マウスクリックで選択する．移動距離を短縮できるのに加えて，右のような小さなターゲットを選択するのが容易になる利点も持つ．ただし，カーソルをどこまで移動したときにこの小さなターゲットを捕捉するかわかりづらい問題がある．

2.2.2 ステアリング支援

ステアリングタスクにおいて，ユーザは経路からカーソルが出ないように注意し続ける必要があるが，力覚フィードバックを与えることでその負担を軽減する手法が提案されている．Dennerlein らはマウス自体を 2 次元方向に引っぱるデバイスを使用し，経路の中央に向かって 0.8 ニュートンの引力を発生させることで操作時間を 52%短縮させることに成功している [Dennerlein, 2000]．これと同様の効果をソフトウェアのみで実現したのが Force-fields である [Ahlström, 2005]．図 2.10 において，子メニューがない項目は x 軸方向中央に向かって力がかかるようにカーソル位置が修正され，その項目が選択しやすくなる．子メニューがある項目は，子メニューに向かって力がかかり，上下方向にずれてしまわないように支援される．

Sun らは複数のフィードバック方法の差を報告しており，カーソルがはみ出た部分だけ軌跡の色を変更する視覚フィードバック，通知音を鳴らす聴覚フィードバック，スタイラスに取り付けたバイブレータを振動させる触覚フィードバックを比較している [Sun, 2010]．触覚フィードバックを提示することで，経路からはみ出してしまった後に早期に経路内に戻れることが判明している．これは人間の触覚に対する反応時間が視聴覚より短いためであると考察されている．

魚眼レンズの視覚効果の加えることで操作時間が短縮されることが示されている [Gutwin, 2003]．CD ゲインを変化せずとも，カーソル付近の経路幅 W が広がったように見せるだけで通過が容易になることが確かめられている．

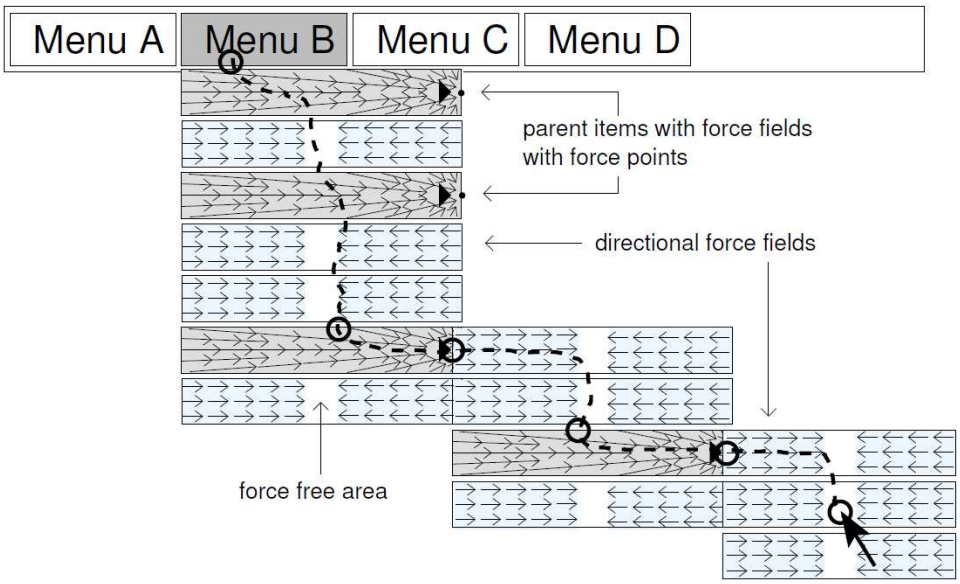


図 2.10　Force-field は，子メニューがあればそちらへ進みやすく，子メニューがなければその中央で止まりやすいようにカーソルに力をかける．

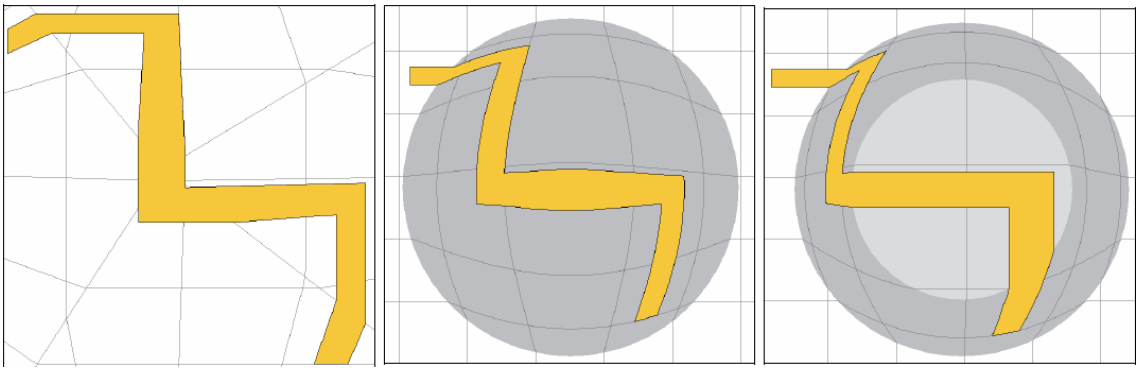


図 2.11　様々な魚眼レンズの視覚効果．カーソルが実際に通過しなければならない距離は短縮されないが，経路幅が広げて見せることで操作が支援される．

2.3　関連研究のまとめ

ポインティングとステアリングを支援する手法は，その手法がいかにして操作を改善するかをパフォーマンスモデルで説明するのが基本である．ポインティングであれば，「この手法はターゲットサイズ W を拡大することで難易度 ID を低減し，操作時間 MT を短縮する」といった具合である．したがって，定量評価の土台であるパフォーマンスモデルは，タスクの難易度と操作時間の関係をより正確に表せた方がよい．この観点から，GUI 操作に関する研究では，操作を支援する（例：操作時間を短縮する，操作ミスを軽減する）手法を開発することも重要だが，タスクの難易度や操作効率などを定量的に評価するためのモデル構築も重要である．

本章で挙げた関連研究を俯瞰すると，操作の改善手法が提案されている一方で，新しいパフォーマンスモデルを構築したり，既存のモデルを修正する研究がいくつも行われていることがわかる．またタスクのパラメータと操作時間の関係を分析することで，操作支援手法の構築に結びつけることもできる．たとえば Asano らは，ターゲット距離 A とカーソルのピーク速度の間に線形の関係があるという観察結果を使い，カーソルを目標地点までジャンプ手法させる手法を提案している [Asano, 2005] [Takashima, 2008]が，なぜターゲット距離とピーク速度が一次関数になるのかというモデル構築はしておらず，「ピーク速度以前の運動を特徴づける量はターゲット距離のみに依存し，サイズは影響しない」と述べるに留めている [Takashima, 2008]. これはタスクパラメータと操作時間の関係を分析するだけでも，操作支援手法の構築に役立てられる一例である．したがって，パフォーマンスモデルの構築にまで至らずとも，ユーザの操作を綿密に分析することで操作支援手法の開発に結び付くことが期待される．

本章で紹介したポインティングとステアリングの先行研究を整理すると，大きく「操作の分析とモデル化」と「操作支援手法の開発」に分類できる．「操作の分析とモデル化」は，

あるタスクにおいてユーザがどのような操作をするのか，タスクパラメータと結果（操作時間やエラー率）の間にどのような関係があるのかを分析し，パフォーマンスモデルを構築することである．Asano ら [Asano, 2005]のようにモデル化をせずに関係性を分析する場合もあれば，分析とモデル化を同時で行ったステアリングの法則の論文 [Accot, 1997]もあり，また既にモデル化されたものを修正する研究もある [Accot, 2003]．これに対し，「操作支援手法の開発」は，操作が困難な状況をピックアップし，それを解消するための方法を提供することである．これはどの程度操作が改善されるかをパフォーマンスモデルによって分析されることが多く，そのうえで実験によって改善度合いを評価するのが一般的である．

操作支援手法だけでは定量的な議論が成り立たず，また操作の分析やモデル構築だけではいつまでもユーザの不便は解消されないままになってしまう．したがって，「操作の分析とモデル化」および「操作支援手法の開発」両方の研究・開発を進めることが重要であると考える．本論文では，ポインティングおよびステアリングにおいてユーザの負担を軽減することを目指し，そのための基礎的な分析とモデル化，そして操作支援手法の開発を行う．

2.4　本論文の目的

本論文全体の目的は，粗い到達運動によってユーザの GUI 操作の負担を軽減することである．そのために，操作を支援する手法の開発と，負担の軽減度合いなどを定量的に評価するための分析・モデル化に取り組む．第 2.1 章，第 2.2 章で紹介した先行研究をふまえて，以下では本論文における具体的な分析，モデル化，支援手法開発の対象を示し，GUI 研究において期待される貢献をまとめる．

2.4.1 無限大サイズのターゲットをポインティングする動作の分析

サイズが小さいターゲットを選択するタスクについて，多くの研究者がポインティング動作の分析や操作支援手法の開発に取り組んでいる．それに対して大きなターゲットのポインティングタスクは GUI 研究において十分に分析されていないといえる．操作自体は容易なために，特段の支援が必要ないと見なされていることが一因だと思われる．しかしながら，実際の PC 操作において日常的に行われるタスクであり，またサイズ以外に影響しうるパラメータも存在するため，きわめて大きなサイズのターゲットをポインティングする

動作を分析することも基礎的な検討として重要であると考える．

本研究で取り扱うのは，ターゲットの奥行きサイズ W，およびそれと垂直な方向のサイズ H がともに無限大のターゲットである．いずれか一方のサイズが無限大の場合のパフォーマンスモデルは先行研究で提案されているが，両者が無限大のときには従来の分析方法を適用することができない．そこで本研究で独自に実験を行い，操作時間やカーソルの軌跡を分析することで，無限大サイズのターゲットをポインティングするときの挙動を明らかにする．これは GUI 研究のみならず，より広く HCI (Human-computer interaction)分野において重要なテーマである「人間の運動の観察，分析」の 1 つととらえることができる．本研究ではマウス操作で 2 次元平面上のターゲットをポインティングする動作を取り上げるが，従来から GUI 研究における知見が 3 次元空間中での運動に応用されることも多く（[Grossman, 2004]など），本研究は「荒っぽい」動作で済むタスクを分析する第 1 ステップとして重要なものであると考える．

本研究のような基礎的な分析が行われることで，操作支援手法に関する研究も発展していくことが期待される．たとえばカーソルのピーク速度からターゲットまでの距離を予測してジャンプさせる手法 [Asano, 2005] [Takashima, 2008]では，ターゲットサイズがピーク速度に影響しないことを検証している．それと同様に，実験結果（速度のピーク位置や操作時間など）に影響する/影響しないパラメータを特定したい，実験結果を予測する方法を検討しておくことで，将来的にユーザを支援する手法の開発に結びつくことを望む．

2.4.2 幅の変化する経路の通過方向による難易度差のモデル化

ステアリングの法則を初めて提案した文献 [Accot, 1997]では，一定幅の直線状経路，幅の狭まる直線状経路，幅の広がる螺旋状経路の操作時間がモデル化されている．このうち幅が変化する 2 種類の経路について，分析の対象となる具体的な操作例や，あるいは応用できるアプリケーションシナリオなどは示されていないものの，制御の細かさが連続的に変化するタスクの挙動分析としての価値は高い．すなわち，経路の形状（直線状，螺旋状，環状，S 字型など）や幅の変化（一定，狭まる，広がる）などによらずモデルがフィットすることで，幅の決まった経路を通過する操作がいずれもステアリングの法則でモデル化できることを強固に示したのである．

しかし，ステアリングの法則によって算出される難易度がうまく適合しない例があることが指摘されている．たとえば経路内にカーブがあると，ユーザはなるべくコーナーの内側を通るために，モデルで予測される操作時間よりも実測の操作時間が短くなることがわかっている．このようにステアリングの法則がうまく適用できない事例があるため，これ

までにも適切な難易度を算出できるようにモデルが修正されたことがある．それによって特定のタスクの難易度や，あるいは新しい操作支援手法によって難易度がどの程度改善されるのか，といった検討項目に対して適切な定量評価ができるようになる．

本研究では，幅が変化する直線状経路を通過する時間が，通過する方向（幅が狭まる方向，広がる方向）によって変化することを発見した．この原因を追求することで難易度の差を算出するモデルを構築し，実験によってモデルの妥当性を検証することを目指す．従来のステアリングの法則と同様に，本研究の貢献は人間の挙動に関する基礎的な検討にある．応用先となるアプリケーションシナリオに富んだものではないものの，「広がる方向へ通過した方が簡単である」ということを初めて指摘した研究であり，それと同時に「どの程度簡単なのか」を定量的に算出できるモデルを構築することが貢献である．

このモデルを使用すると，広がる方向への通過時間を複数の実験パラメータで測定すれば，狭まる方向への通過時間を少ないパラメータ数（最も容易な難易度）で実験するだけで高精度に予測できるようになる．つまり狭まる経路では，最も粗い制御で済むパラメータだけで実験すればタスクの難易度を測定でき，ユーザ（ここでは測定者・実験参加者）の負担を軽減できるのである．よって本研究は，制御の微細さが連続的に変化する経路の難易度をモデル化した貢献に加えて，粗いステアリング操作で済ませられることでユーザを支援する手法を開発したと解釈することもできる．

2.4.3 細長いターゲットのドラッグアンドドロップ支援

小さなターゲットのポインティング支援手法はこれまでに多く提案されており，それぞれが評価実験で一定の有用性を示している．しかし，ターゲットの形状によって操作時間が変わる [Accot, 2003]ことや，ターゲットの配置間隔によって既存手法の有用性が低減する [Shigemori, 2007] [Tsukitani, 2011]ことを考慮すると，タスクの内容によっては既存手法では支援が不十分になる可能性がある．

本研究で着目するのは，細長いターゲットのドラッグアンドドロップ (DnD)操作である．現代の PC 作業において，ウィンドウをリサイズするための外枠や，ウィンドウ内のレイアウトを変更するための境界線，表計算ソフトのセルをリサイズするための境界線など，細長いターゲットは多く存在する．従来はターゲットを拡大する手法やカーソル速度を低下させる手法が多く提案されていたが，本研究では線分を通過することでドラッグを開始する手法 Cross-drag を提案する．Cross-drag はターゲット上にカーソルを乗せるための微細な制御が必要なく，線分のターゲットにカーソルを「ぶつける」ようにするだけでドラッグを開始できる．またカーソル速度の低下も起こらないため，DnD タスク全体の操作時間を短縮することが期待される．しかしターゲットが短かったり配置間隔が狭いことで効果が低減する懸念があるため，実験によって有用性と限界を評価し，提案手法の有効範囲を明

らかにすることが必要である．本研究の貢献は，ユーザが普段の PC 作業で労力を割いている「きわめて微細な制御を要するポインティングタスク」を粗い制御で済むような操作手法を提案し，有効範囲を実験的に求めることにある．

2.4.4 長距離と短距離の移動が混在する条件下でのポインティングタスクの支援

長距離の移動を短時間で済ませられればユーザの負担を軽減できる．そしてそれだけを目指すのであれば，OS の設定でカーソル速度を上げれば解決できそうに思える．しかし，ポインティングは長距離の移動だけがあるのではなく，カーソルがターゲット付近に差し掛かったら速度を落とさなければならないし，最終的にはターゲット上でカーソルを停止させなければならない．したがって，カーソル速度を上昇させるだけでは問題の解決にはならず，長距離移動と短距離移動が混在していることを想定したポインティング支援方法を考えなければならない．

そこで本研究で提案するのが，視線情報を用いてカーソル速度を変更する手法である．近年では視線計測器の低価格化が進み，標準搭載する PC が販売されるまでに至っている．そこでポインティングを支援するインフラとして視線計測器を使用し，注視点とカーソルが離れているときにのみカーソルを高速化させる手法を開発する．この手法であれば，ユーザが粗い（大雑把な）制御をしたいポインティング開始時点の時間を短縮でき，その後にカーソルが注視点付近にあるときは通常の速度に戻るため，時間短縮と操作精度を両立できると考える．先行研究でも視線情報を利用してカーソルをジャンプさせるなどの支援方法が提案されているが，不必要なタイミングでジャンプが発生する設計になっている不備がある．本研究の提案手法はそういった問題を解消しつつ操作時間を短縮するものである．視線情報を利用するにあたって，計測誤差や眼球の固視微動など，考慮すべき事項について整理する必要がある．そのうえで支援手法を設計し，評価実験によって有用性を確認する．本研究の貢献は，既存の操作を阻害せずにポインティング効率を向上させる手法を提案し，実験でその有用性を示したことにある．

第3章　無限大のサイズをもつターゲットをポインティングする操作の分析

3.1　はじめに

本稿ではのように *W* と *H* がともに無限大とみなせる場合のポインティング動作について議論する．フィッツの法則でパフォーマンスをモデル化できるのは，図 3.1a あるい図 3.1b のようにサイズが規定されたターゲットをポインティングするタスクのみである．図 3.1c のように *W* と *H* が両方とも無限大のターゲットをポインティングするとき，カーソルをターゲットに乗せるための微調整は必要ない．つまり，求められる動作が「カーソルを距離 *A* 以上移動させてクリックする」のみになり，一般的な（ボタンやアイコンの）ポインティングタスクとは異なる動作になると考えられる．また，このようなタスクの操作時間 *MT* はフィッツの法則では予測できない．

図 3.1c のようなターゲットをポインティングする状況は GUI 環境においていくつも見られ，基本的な動作の一種といえるが，タスクのパラメータによって *MT* がどのように変化するのか，またタスクのパラメータから *MT* を予測可能かは不明である．図 3.1b のような状況では，ターゲットまでの距離 *A* や，画面内に表示されているターゲットのサイズによって操作時間 *MT* が有意に変化することがわかっているが，図 3.1c においても同様の結果が得られるかは未知である．そこで本章では，図 3.1c のようなタスクにおけるポインティング時の挙動を分析し，ターゲットサイズが規定された一般的なタスクとの差異について考察する．また，タスクのパラメータから *MT* を予測する方法を検討する．

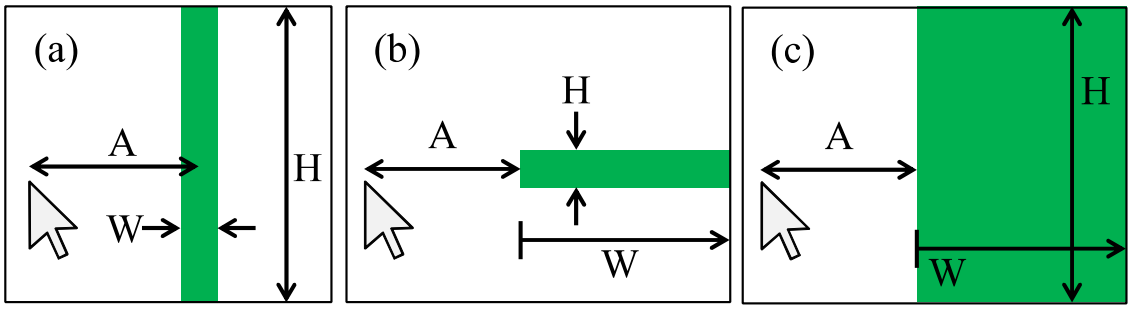


図 3.1　無限大のサイズをもつと見なせるターゲット．

3.2 関連研究

第 2.1.1 章において W だけが無限大の場合，および H だけが無限大のパフォーマンスモデルを紹介した．これに対し，実際の GUI 環境では図 3.2a-d のように W と H がともに無限大の場合があるが，このような状況下でのポインティング時間は十分に検討されていない．W と H がともに無限大になれば，x 軸および y 軸の両方向に対して微細な制御をする必要がなくなり，ポインティング時の戦略が変化すると想定される．具体的には，ユーザは一定距離 A 以上移動することだけに注意すればよいため，W あるいは H が規定されているときよりも「荒っぽい」動作で済むと考えられる．このような操作の違いが想定されるが，先行研究ではターゲットサイズ W，H の少なくとも一方が規定されることが多く，調査した限りでは両者を無限大に設定しているものは見あたらなかった．

では，W を奥行き方向に無限大，H を両方向に無限大に設定したときに，フィッツの法則の式 $MT = a + b\log_2(A/W+1)$ あるいは $MT = a + b\log_2(A/H+1)$ おけるターゲットサイズにどのような値を設定すればよいだろうか．たとえば図 3.2a のような状況でアイコンをデスクトップに移動するとき，W と H は無限大だが，$W = H = \infty$ を無限大にすると $MT = a$ と予測されてしまう．しかしこれでは距離 A に関わらず一定時間で選択できることになり，明らかに誤りである（実際に本稿の実験結果で否定される）．

実験データを分析してタスクの難易度やデバイスの性能を比較したいときには，クリックされた座標からターゲットの実効幅 W_e (Effective width [MacKenzie, 1992])を算出する分析方法が採用できる．これは左右方向に移動するポインティングタスクにおいて，クリックされた x 座標の標準偏差を σ とし，$W_e = 4.133\,\sigma$ をターゲットサイズに適用する方法である．しかしこの方法では，本稿で扱っているようなサイズが無限大の状況で新たなタスクを与えたときの操作時間を予測することはできない．なぜなら実効幅 W_e の値が A によって変化する可能性があるためである（これも本稿の実験結果で確認される）．また，実効幅 W_e を用いた分析手法はオーバーシュート（ターゲットを行き過ぎること）とアンダーシュート（ターゲットよりも手前で止まること）のエラーが発生するタスクが対象であり，本実験のタスクへの適用可能性は未知である．

以上をまとめると，既存のパフォーマンスモデルは次のような問題を抱えているといえる．

- フィッツの法則の式において，ターゲットサイズに∞を代入すると不適切な MT を算出してしまう．
- 実効幅 W_e が A に依存する可能性があるため，新たな実験パラメータにおける MT を予測することができない．

本稿ではターゲットサイズ W と H をともに無限大にしたポインティングタスクを行い，実効幅 W_e や操作時間 MT が実験パラメータとどのように関係しているかを分析する．たと

えば「実効幅 W_e は距離 A に依存することが判明し，フィッツの法則の式に代入する W の値を A から算出できる」などといった関係が見いだせれば，既知のターゲットパラメータから MT を予測できるようになる．また，W_e を用いた分析が有効なのか（フィッツの法則は成立するのか）を検証するなど，無限大サイズのターゲットポインティングタスクを複数の視点から分析する．

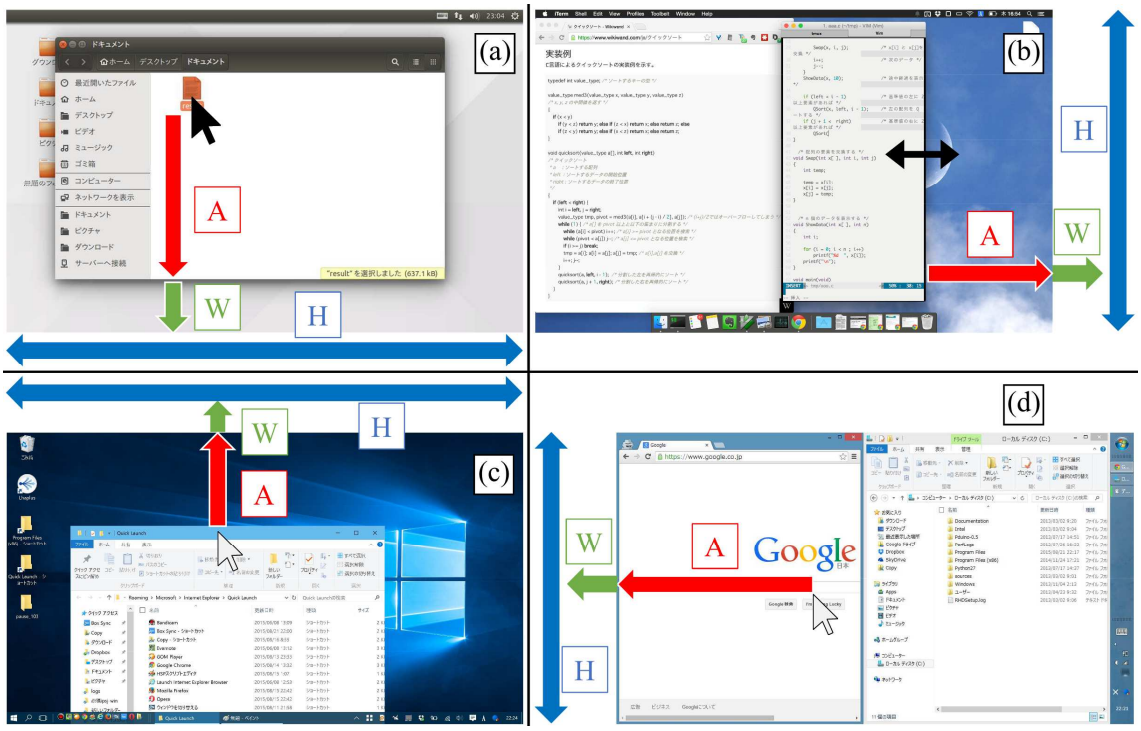


図 3.2　サイズ W と H を無限大とみなせる操作の例．距離 A 以上移動すれば W と H のサイズを考慮しなくてよい．なお，ドラッグアンドドロップ操作もフィッツの法則に従うことが知られており [MacKenzie, 1991]，ポインティングタスクの一例ととらえることができる．(a) ウィンドウ内のファイルをデスクトップに移動 (Ubuntu 15)，(b) ウィンドウの右縁をドラッグして画面右端まで拡大 (Mac OS X)，(c) ウィンドウを画面上端にドラッグアンドドロップして最大化 (Windows 10 の Aero Snap)，(d) ウィンドウをリサイズするために左縁を選択 (Windows 8 の Aero Snap でタイリングしたウィンドウに対して，ウィンドウの左縁をドラッグしてデスクトップの左端に置かれたショートカットを露出させる操作を想定).

3.3　実験

本実験では，ターゲットサイズ W をカーソル移動の奥行き方向に無限大，H をそれと垂直な両方向に無限大に設定したポインティングタスクを行い，操作時間 MT が実験パラメータとどのように関係しているかを分析する．また，カーソル速度の時系列的な波形を分析することで，無限大サイズと規定サイズでポインティング時にどのような戦略の違いがあるかを分析する．

3.3.1 実験デザイン

3.3.1.1 タスク

タスクが開始されると，図 3.3 のように灰色背景の画面の一端に緑色のターゲットが表示される．そこから一定距離 A を挟んで赤色で直径 25 pixels のスタートボタンが y 軸中央に表示される．ターゲットは画面上下の端まで広がったサイズを持ち，ターゲットが描画されている奥行き方向のサイズは W_v（Visible width，可視領域のサイズと呼ぶ）である．

画面が表示されるとカーソルが自動的に画面中央に移動する．実験参加者はスタートボタンをマウスの左ボタンでクリックし，続いてターゲットの範囲内にカーソルを移動してクリックする．ターゲット内でクリックすればベル音が鳴って試行成功が通知され，ターゲットの範囲外であればビープ音が鳴ってエラーが通知される．試行の成否に関わらず，クリックした時点で画面とカーソルが 1 秒間消去され，そのあとに次のパラメータが選出されて画面に表示される．

計測するデータは，スタートボタンをクリックしてからターゲットをクリックするまでの時間，タイムスタンプ付きのカーソルの軌跡，エラーの回数である．このうちカーソルの軌跡は，カーソルが画面端に引っかかって以降にマウスを動かし続けた距離も記録される．すなわちカーソルは通常のデスクトップ環境と同様に画面端で引っかかった表示になるが，システム側ではカーソルが画面外へ移動していったとみなした場合の距離も合算して計測する．

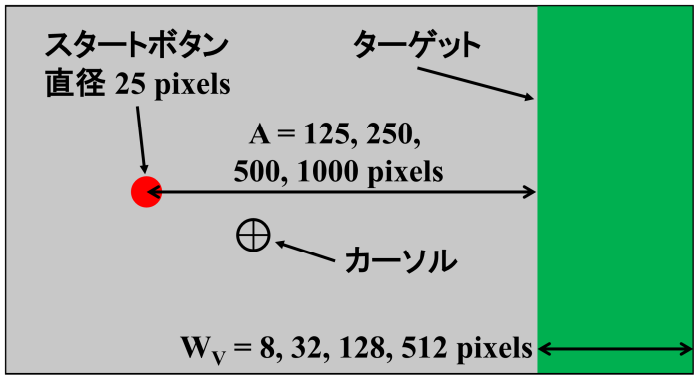


図 3.3　実験の画面構成の模式図（ターゲットが右方向にある場合）.

3.3.1.2 カーソルを試行ごとに画面中央へ戻す処置について

ユーザの次の行動が決まっていて，なるべくカーソルの行き過ぎを防ぎたい場合がある．たとえば 2 つのウィンドウ間を何度も往復してファイルを移動したい場合などである．こういったタスクでは，操作の合計時間を短縮するためになるべく往復移動の内側の狙うことが考えられる．あるいは「ファイルをドロップした後はウィンドウを閉じてよいので，次は右上の閉じるボタンに向けて移動する」，「次はスタートメニューを開きたいので左下に移動する」といった作業上のコンテクストによってユーザが操作を変える可能性がある．そのような次のタスクを予め考えた上で操作すると，メインの作業に何らかの影響が出ることが考えられる．

現実的な作業を考慮すれば操作の流れを設定することも重要だが，ここではそのベースとなるパフォーマンスに焦点を当てることにしたい．つまりサイズが無限大のときにどのような操作をするかを観察するのが主目的であるため，奥行きが無限大であることを活かさない戦略をとりうる上記のようなタスクは不適切である．したがって，1 回の試行ごとにカーソルの位置をリセットし，ターゲット内のどこをポインティングしても次の試行には影響しないようにした．

3.3.1.3 教示

可能な限り短時間で，かつエラーを起こさずにターゲットを選択するよう教示する．本実験では，ターゲットを正確に狙わずに，マウスを大きく動かした後にクリックすればエラーにならない．よって究極的にはスタート後に画面を見ずに試行を成功させることも可能である．しかしこれでは必要以上にカーソルを移動させてしまうことになり，その分だけ操作時間は増大してしまう．ゆえに選択ミスは避けられるものの，「可能な限り短時間で

行う」という教示を守れないことになる．ポインティングタスクを課した実験では，正確な操作と時間短縮の両方を目指すように教示されるのが一般的であり（たとえば[Accot, 2002]や[Appert, 2008]など），本実験でも一方を度外視するような操作は避けるべきだと考える．よって，意図的に長距離の移動をしてエラーを回避する行為，たとえば「必ずカーソルを画面端まで移動させてからクリックする」などという戦略は認めないことを事前に伝えた．エラー回避と短時間の両立を目指して，結果的に画面端までカーソルが移動することは問題ないと教示した．

3.3.1.4 視覚フィードバック

Appert らの報告 [Appert, 2008]によれば，画面の上下端に置かれたターゲットをポインティングするタスクにおいて，矢印型カーソルを使用すると環状カーソルより操作時間が有意に増大した．これは画面下端をポインティングするときにカーソルが視認できなくなってしまうことが一因であると考察されている．またカーソルが乗ったときにターゲットの色を変化させるフィードバックを付加すると，操作時間が有意に増大することも報告されている．事後アンケートでは，色が変化する機能によって操作が支援されていたと感じた実験参加者と，阻害されたと感じた参加者が半数ずつであった．

以上の知見から，本実験ではターゲットの色を変化させる視覚フィードバックは付加しない．またカーソルは環状を検討したが，事前実験においてホットスポット（実際にクリックされる座標）が正確に認識できない問題が観察された．そこで幅 1 pixel の線分を交差させた十字型カーソルの周りに，幅 2 pixels の線で円を描いた直径 25 pixels の複合型カーソルを用いることにした（図 3.3）．

3.3.1.5 使用機器等

PC は Sony VAIO Z SVZ1311AJ（2.1 GHz×4 コア，8 GB RAM, Windows 7）を使用した．ディスプレイは I-O Data LCD-TV241XWR（518.4×324.0 mm，1920×1200 pixels）であり，HSP 3.4 で実装した実験システムをフルスクリーンで表示する．システムは約 125 Hz で動作する．マウスは Logicool G300r（1000 DPI，レポートレート 1000 Hz，光学式，有線），マウスパッドは Perixx DX-1000XXL（900 mm×440 mm）を用いた．操作時間やカーソルの移動距離がマウスパッドの面積の影響を受けないように広大なものを設置し，またマウスのケーブルも 2.0 m あり，十分なゆとりをもって操作できる環境を設けた．カーソルの CD ゲインは OS のデフォルト（コントロールパネルで目盛り 11 段階の中央）に固定し，ソフトウェア加速をオフにした．

3.3.1.6 実験参加者

実験参加者は情報系の大学生及び大学院生の 10 名（女性 2 名，男性 8 名，平均 22.3 歳，標準偏差 2.69 歳）である．全員がマウス操作に習熟し，利き手の右手で操作した．

3.3.1.7 手順

距離 A は 125, 250, 500, 1000 pixels の 4 種類とした．これは近距離の移動でカーソルの速度があまり大きくならない距離から, フル HD ディスプレイの横半分以上を移動する中〜長距離程度をカバーする範囲である．またターゲットの可視領域のサイズ W_v は 8，32，128，512 pixels の 4 種類である．先行研究 [Appert, 2008]では画面端のターゲットを W_v = 20, 320 pixels にしていたが，実際の GUI 環境ではウィンドウの縁やデスクトップ領域など W_v の値がより広範囲であることを考慮して設定した．ターゲットの方向 Dir は左，右の 2 種類である．

各実験パラメータ (A, W_v, Dir)がランダムな順序で選出されるのを 1 ブロックとし，これを合計で 8 ブロック試行する．記録されるデータは実験参加者 1 名あたり 4 (A)×4(W_v)×2 (Dir)×8 (ブロック) = 256 回分である．本番の第 1 ブロックの前には練習を 1 ブロック分だけ試行する．ここで実験参加者に椅子の高さやディスプレイの角度などを調整させた．また各ブロックの間には小休止を設け，参加者がタスクに集中できるようにした．所要時間は，事前のインストラクションから全試行終了まで 15〜20 分程度である．

3.3.2 結果

実験全体で 2560 回の試行があり，このうち光学マウスに見られるカーソルの瞬間的なジャンプが 2 回発生した．具体的には，スタートボタンをクリックした時点でカーソルがターゲット内に移動した試行と，同様の原因で実験参加者がジャンプ後のカーソルを見失った試行であり，これらは分析から排除した．残りの 2558 試行に関して，ターゲットの領域外でクリックしたエラーが 45 回観察された．操作時間 MT の分析にはターゲットの実効幅 W_e を用いるため，エラーを起こした試行を排除していない．以降のデータ分析には実験参加者の対応ありの 3 元配置分散分析を用いる. また多重比較には Bonferroni の手法を用いる.

3.3.2.1 操作時間 *MT* の分析

実験パラメータごとの *MT* の値を図 3.4a-c に示す．主効果が認められたのは *A* ($F_{3,27}$ = 47.461, p < .001), *Dir* ($F_{1,9}$ = 7.351, p < .05)であった．W_v には主効果が認められなかった ($F_{3,27}$ = 2.227, p = .108)．多重比較では，*A* = 125 と 250 の間に有意差が認められず，また *A* = 250 の方が *A* = 500 より短時間であった (p < .01)．それ以外の *A* の間には全て p < .001 の有意差が認められ，距離 *A* が長いほど操作時間 *MT* が増大した．また *Dir* は右向きの方が短時間であった (p < .05)．いずれの実験パラメータ間にも交互作用は認められなかった．

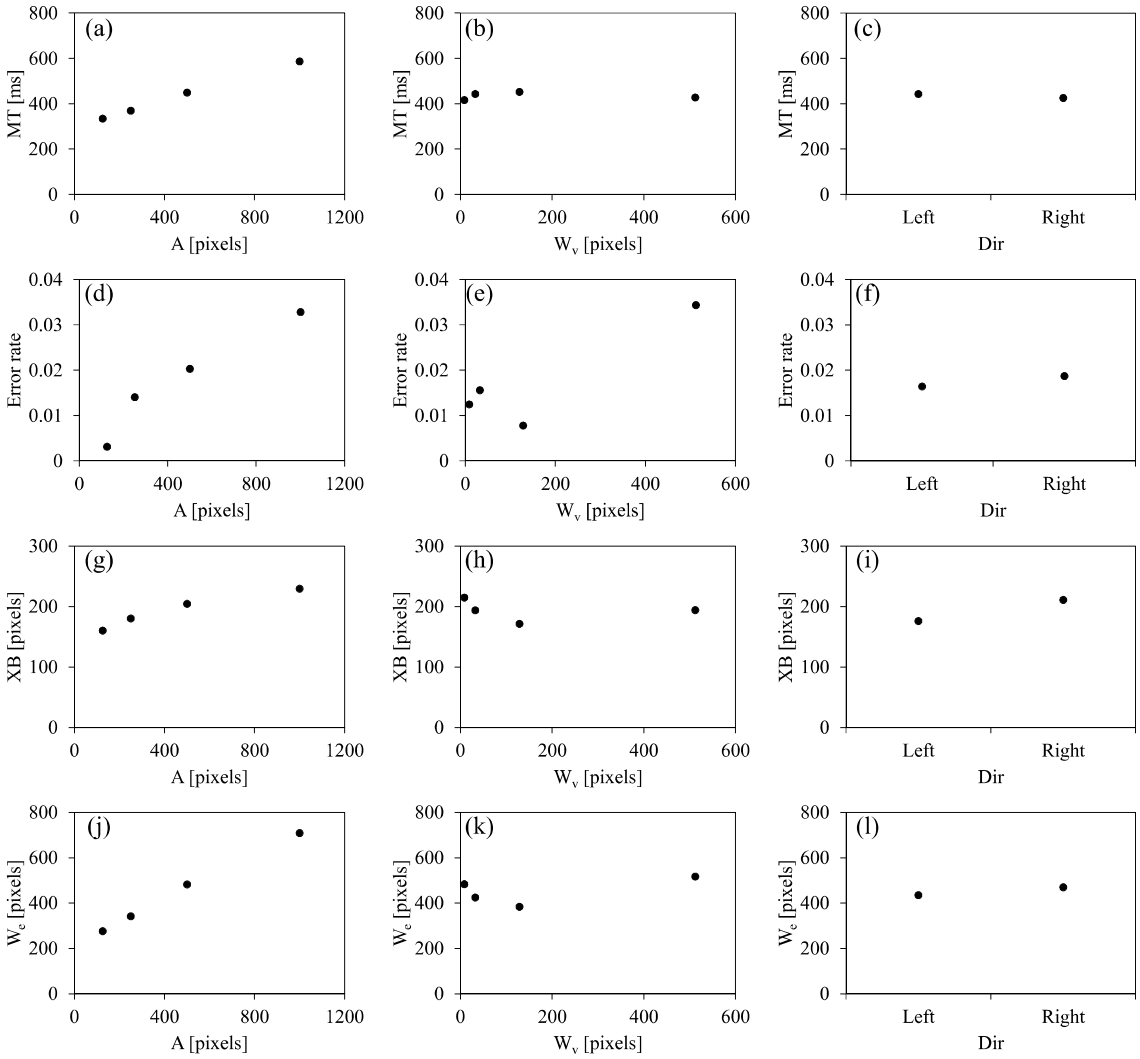


図 3.4　実験パラメータごとのデータ．

3.3.2.2 エラー率 *ER* の分析

実験パラメータごとのエラー率 *ER* (Error rate)の値を図 3.4d-f に示す．主効果が認められたのは *A* ($F_{3,27}$ = 3.183, p < .05), W_v ($F_{3,27}$ = 6.956, p < .01)であった．*Dir* には主効果が認められなかった ($F_{1,9}$ = .574, p = .468)．多重比較では，*A* = 125 とその他の *A* の間に全て p < .01，*A* = 250 と 500 および *A* = 250 と 1000 の間に p < .05，*A* = 500 と 1000 の間に p < .05 の有意差が認められた．距離 *A* が長いほどエラー率 *ER* が増大した．
また全ての W_v の間に p < .01～.05 の有意差が見られたが，最もエラー率 *ER* が低かったのは W_v = 128 のときであり，特定の傾向は観察できなかった．いずれの実験パラメータ間にも交互作用は認められなかった．

3.3.2.3 ターゲットの領域内に侵入してからクリックするまでの x 軸方向の移動距離 *XB* の分析

カーソルがターゲット内に入ってから可能な限り早くクリックした方が操作時間を短縮できるため，無駄な移動距離を 1 つの評価指標として分析する．ここではターゲット領域に入る境界線を越えてからの x 移動距離という意味で *XB* (X-position beyond the Boundary)と呼ぶことにする．ターゲットの領域よりも手前でクリックした場合，つまりエラーが発生した場合には負値が記録されている．

実験パラメータごとの *XB* の値を図 3.4g-i に示す. 主効果が認められたのは *A* ($F_{3,27}$ = 4.094, p < .05), *Dir* ($F_{1,9}$ = 9.671, p<.05)であった. W_v には主効果が認められなかった ($F_{3,27}$ = 2.353, p = .094)．多重比較では，*A* が大きくなるほど *XB* が大きくなる傾向が見られたが，全ての *A* の間に有意差が見られなかった．*Dir* は右向きの方が *XB* が大きくなった (p < .05．いずれの実験パラメータ間にも交互作用は認められなかった．

3.3.2.4 ターゲットの実効幅 W_e の分析

クリックされた x 座標の標準偏差を σ とし，W_e = 4.133 σ で実効幅を求める．実験パラメータごとの W_e の値を図 3.4 (j-l)に示す．主効果が認められたのは *A* ($F_{3,27}$ = 30.122, p < .001) のみであり，W_v ($F_{3,27}$ = 2.248, p = .106)と *Dir* ($F_{1,9}$ = 1.820, p = .210)には主効果が認められなかった．多重比較では，*A* = 125 と 250 の間以外の全ての *A* の間に少なくとも p < .05 の有意差

が見られ，A が大きくなるほど W_e が大きくなった．いずれの実験パラメータ間にも交互作用は認められなかった．

3.3.3 考察

3.3.3.1 エラー率 *ER* について

実験を通しての平均エラー率は 1.76%であり，一般的なポインティングタスクのエラー率である 4〜5% [Wobbrock, 2008-b]より低かった．本実験ではオーバーシュートが存在しないことが原因であると考えられる．距離 A が大きいほどエラー率が増大しているが（図 3.4d），これは A 自体が「クリックされたらエラーとみなす範囲」であるため妥当な結果であると考えられる．一方で W_v にはそのような役割がないためエラー率に影響しないと考えていたが，実際には図 3.4e のように主効果を与えていた．W_v が小さいとカーソルがターゲット内に入ったことを目視しづらいため，クリックするタイミングを慎重に判断したり，逆に W_v が大きすぎると「このくらいマウスを動かしたらターゲットに入っているだろう」などと考えて早めにクリックするといった影響が生じた可能性がある．エラー率が最も低かったのは W_v = 128 のときだが，操作時間 MT をそれほど増大させずに（図 3.4b）エラー率を低減できるため，本実験におけるターゲットの可視領域は 128 pixels 程度がよいといえる．

3.3.3.2 ターゲットの領域内に侵入してからクリックするまでの x 軸方向の移動距離 *XB* について

距離 A が大きいほど XB は増大することが確認された．遠くのターゲットを素早くクリックするためにカーソルを速く移動させ，そのためターゲットに進入したのを目視してからクリックするための移動距離が長くなったと考えられる．Dir が右の方が XB が大きかったが，これは右方向がマウスを動かしやすいためであると考えられ，操作時間 MT が短いという結果にも現れている（図 3.4c）．

3.3.3.3 実効幅 W_e を用いたフィッツの法則の適合度

MacKenzie [MacKenzie, 1992]によれば，実効幅 W_e をフィッツの法則の式に適用した難易

度指標 ID_e によって，モデルがよりよく適合するはずである．本実験データをこのモデルに適用すると，回帰分析の結果は図 3.5a のようになった．操作時間 *MT* に関して，可視領域のサイズ W_v の主効果は見られなかったため，ここでは距離 *A* と移動方向 *Dir* ごとに分類して示している．モデルの適合度を示す決定係数 R^2 は右方向の方が比較的高い値を得られた．また移動方向 *Dir* を統合して分析すると図 3.5b のようになった．

MacKenzie の分析手法はターゲットのオーバーシュートを考慮したモデルであるが，本実験のタスクに適用しても R^2 = .92 程度の適合度が得られることがわかった（図 3.5b）．しかし，一般的なサイズのポインティングにおいて「ターゲットを行き過ぎないようにカーソルを動かそう」と考えると，慎重に操作する影響で *MT* が増大するはずである．今回の実験ではオーバーシュートによるエラーは存在しないため，実験参加者はより高速にマウスを動かすことができ，W_e で算出される難易度より容易にクリアできたと考えられる．そこで，本実験の難易度と操作時間の関係をより適切に表すモデルについて 3.3.3.5 段で新たに検討することにする．

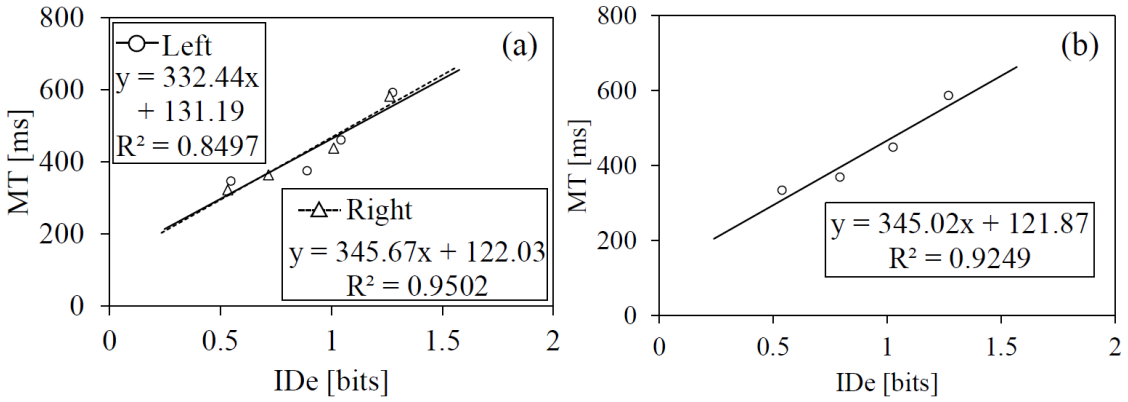


図 3.5　ID_e に対する *MT* の関係．(a) *A*，*Dir* ごとの関係，(b) *Dir* を統合した関係．

3.3.3.4 カーソル軌跡の分析

サイズの規定されたターゲットポインティングでは図 3.6a のように操作時間の前半に速度のピークがあり，ピーク以前の計画時間，ピーク以後の調整時間に分けられることが知られている [Asano, 2005] [Walker, 2003]．特にポインティング時間の多くを調整時間に割くことが指摘され，調整時間を短縮するための手法が提案されている [Asano, 2005]．実際に Ruiz らが行った左右方向へのポインティングタスク [Ruiz, 2003]では，図 3.6b のような時間－速度の波形が描かれ，文献 [Asano, 2005] [Walker, 2003]の結果を支持するものになっている．

Ruiz ら [Ruiz, 2003]と同様に，本実験の軌跡データを経過時間－速度の波形で表したものが図 3.7 である．A = 125, 250 では速度のピークが経過時間の前半に現れているとはいえず，先行研究[Asano, 2005] [Ruiz, 2003] [Walker, 2003]とは異なる結果が得られている．A = 500, 1000 では前半にピークが現れているものも見られるが，減速し始めるタイミングが後半のものや，終盤まで減速しない波形も見られる．これはカーソルを高速に動かしつつクリックしたことで生じたと考えられ，画面端のポインティング独特の特徴といえる．

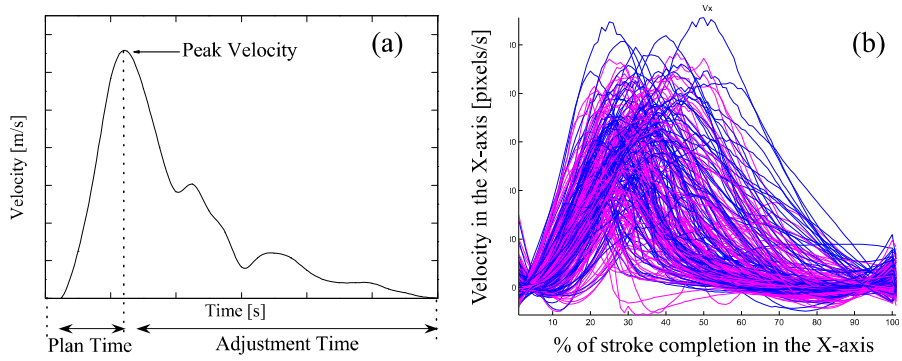


図 3.6　先行研究に示されている，経過時間に対するカーソル速度の関係．(a) は文献 [Asano, 2005]の図 1 より引用，(b) は文献 [Ruiz, 2003]の図 5 に著者が縦横軸のラベルを付加したもの．

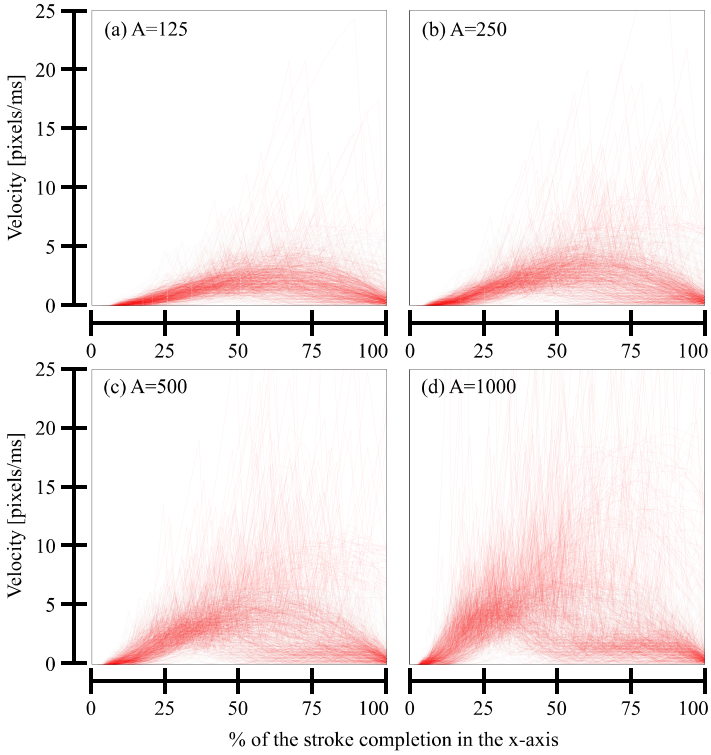


図 3.7　本実験の経過時間に対するカーソル速度の関係．

ポインティング動作終盤の速度をさらに詳細に観察するために，クリックした時刻から 8 ms ごと（システムの 1 サンプル間隔）に遡った平均カーソル速度を図 3.8 に示す．最短で選択した試行が 90 ms だったため，ここでは 88 ms (8 ms × 11 サンプル)まで遡ったデータを示している．一例として図 3.8 の右端の-8 ms のデータは，クリックした時刻を t = 0 として t = -8 から t = 0 に至るまでの移動距離を 8 で割った値を示している．いずれの *A* のグラフも，減速しつつもクリック時点まで継続してカーソルを移動させていることが読み取れる．

図 3.7 と図 3.8 が示すように，*W* と *H* が無限大のターゲットポインティングは，サイズが規定されている場合とは異なる挙動を示していることがわかる．このようなタスクの操作時間 *MT* を既存のモデルから予測することはできないため，実験パラメータと *MT* の関係を適切に表す関係式を新たに検討する必要がある．

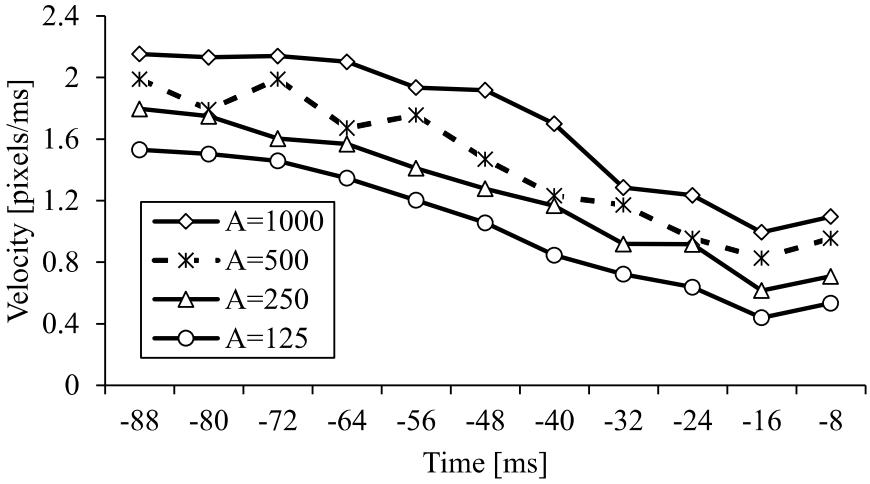


図 3.8　クリックした時刻から遡った時点でのカーソル速度.

3.3.3.5 距離 *A* から操作時間 *MT* を予測する方法の検討

ターゲットサイズの値が他の実験パラメータから求められない限り，新たなタスクを与えたときの操作時間 *MT* が予測できない．実験結果の分析より，W_v は *MT* に主効果を及ぼさないことがわかっており，また移動方向 *Dir* に関わらず *MT* を予測できるか検証したいため，ここでは *A* のみを実験パラメータとして考慮する．

考えうる複数の関係式を最小二乗法で求め，その結果を図 3.9 に示す．ここでは一般的な近似式として，MS Excel に搭載されているものから決定係数 R^2 が高かった式を記載している．3 種類の多項式近似によって $R^2 > .99$ の高い値が得られ，1 次式でも *A* と *MT* の関係を $R^2 = .999$ で表せることがわかった．図 3.9e の 2 次式では x^2 の係数が-0.00005 と非常に 0 に近い値である．同様に図 3.9f の 3 次式も，x^3 と x^2 の係数が極めて小さく，*A* と *MT* の関係はほぼ 1 次式で近似できると考えられる．

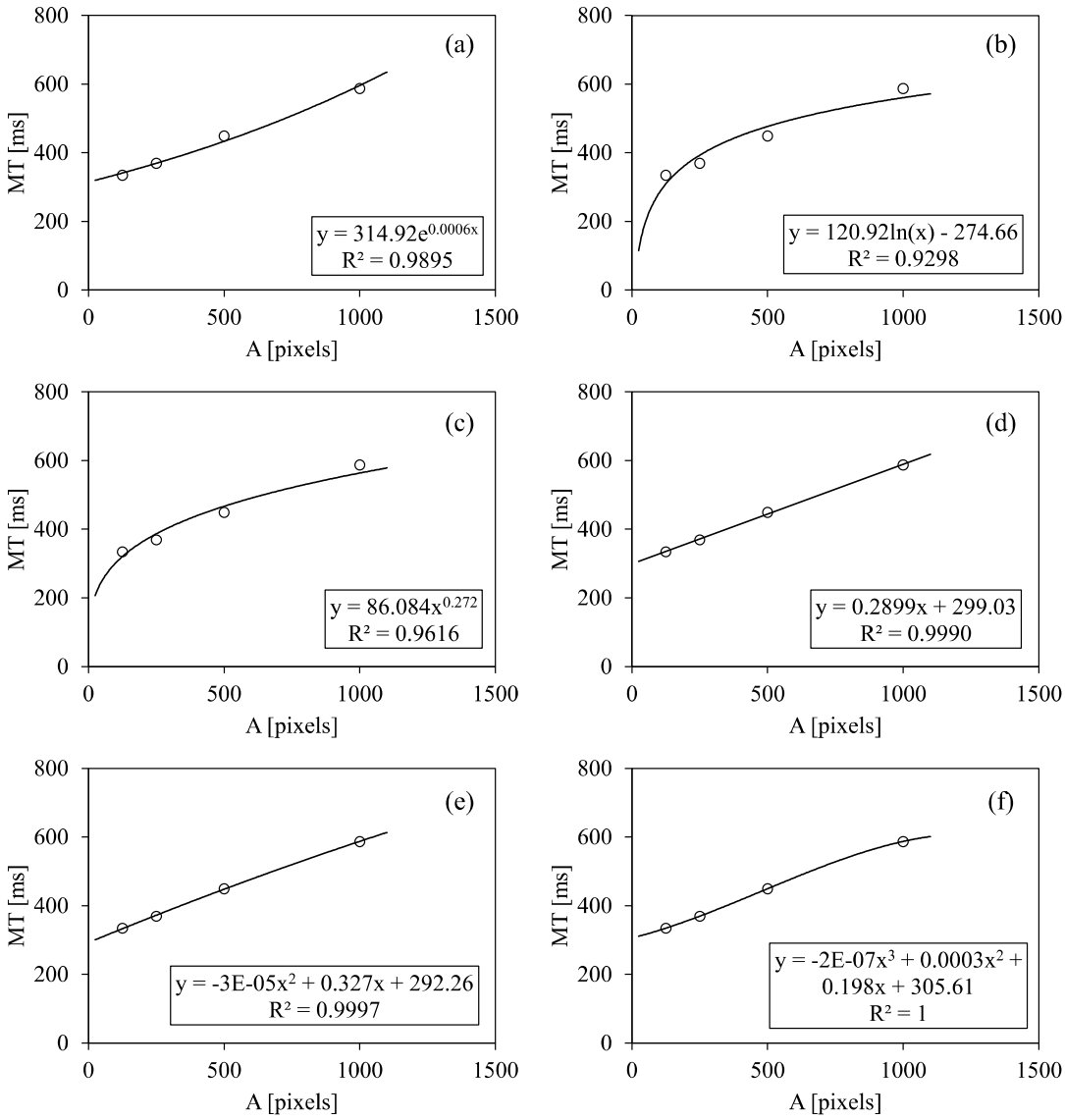


図 3.9　*A* に対する *MT* の関係．(a) 指数近似，(b) 対数近似，(c) 累乗近似，(d) 1 次式近似，(e) 2 次式近似，(f) 3 次式近似．

次に，より正確に近似できた多項式近似（1 次～3 次）に関して，未知の *MT* を *A* から予測しうるかという視点で分析する．つまり新たな距離 *A*（750 pixels など）で実験したときに，*MT* を高精度に予測できるかを検証する．検証方法は，今回実験した *A* のうち 1 種類に関して未計測であったと仮定し，その他 3 種類の実験データから *MT* を予測して，実際の *MT* の値と比較して精度を求める．

一例として，1 次近似で *A* = 125 が未計測である場合の予測精度を求める．他 3 種類の *A*, *MT* のデータを 1 次近似すると，$MT = 300.30 + 0.28826 \times A$ となる．これに *A* = 125 を代入すると，*MT* = 336.33 ms と予測される．実際には *MT* = 334.25 ms であるから，予測値と実測値の誤差は 336.33 – 334.25 = 2.08 ms であり，2.08/334.25 × 100 = 0.622%の誤差が生じる．

その他の A が未計測と仮定したときの予測精度も同様に求めると表 3.1 のようになった．平均誤差割合が最小なのは 1 次近似であり，高難易度 (A = 1000）の操作時間も誤差 2.66% 程度に抑えられた．よって無限大サイズのターゲットポインティングでは，操作時間 MT は距離 A を用いた 1 次式で高精度に予測できるといえる．

表 3.1　多項式近似における MT の予測精度．x^3〜x はその係数を示す．

次数	未知のA [pixels]	x^3	x^2	x	切片 [ms]	MT予測値 [ms]	MT実測値 [ms]	誤差値 [ms]	誤差 %	平均誤差%
1	125			0.288	300	336	334	2.08	0.622	1.45
	250			0.288	301	373	369	3.56	0.963	
	500			0.289	298	442	449	6.96	1.55	
	1000			0.308	294	603	587	15.6	2.66	
2	125		-5.89×10^{-5}	0.364	282	326	334	7.78	2.33	4.03
	250		-3.52×10^{-5}	0.328	294	374	369	4.45	1.20	
	500		1.22×10^{-5}	0.275	300	440	449	8.89	1.98	
	1000		1.07×10^{-4}	0.239	303	649	587	62.3	10.6	
3	125	-4.16×10^{-7}	0.000669	0	334	344	334	9.28	2.78	15.1
	250	-4.78×10^{-7}	0.000741	0	324	362	369	6.75	1.83	
	500	-6.77×10^{-6}	0.000943	0	321	472	449	22.8	5.09	
	1000	-1.09×10^{-6}	0.00107	0	320	290	587	297	50.6	

さらに，実験参加者ごとの操作時間をに示す．参加者によって長時間/短時間の差は見られるが，いずれの参加者も MT と A が 1 次関数になっており，決定係数 R^2 が最も低かった実験参加者 C でも R^2 = 0.982 であった．よってこれら 10 名分のデータを平均した結果が 1 次近似で表されることも妥当であると考えられる．

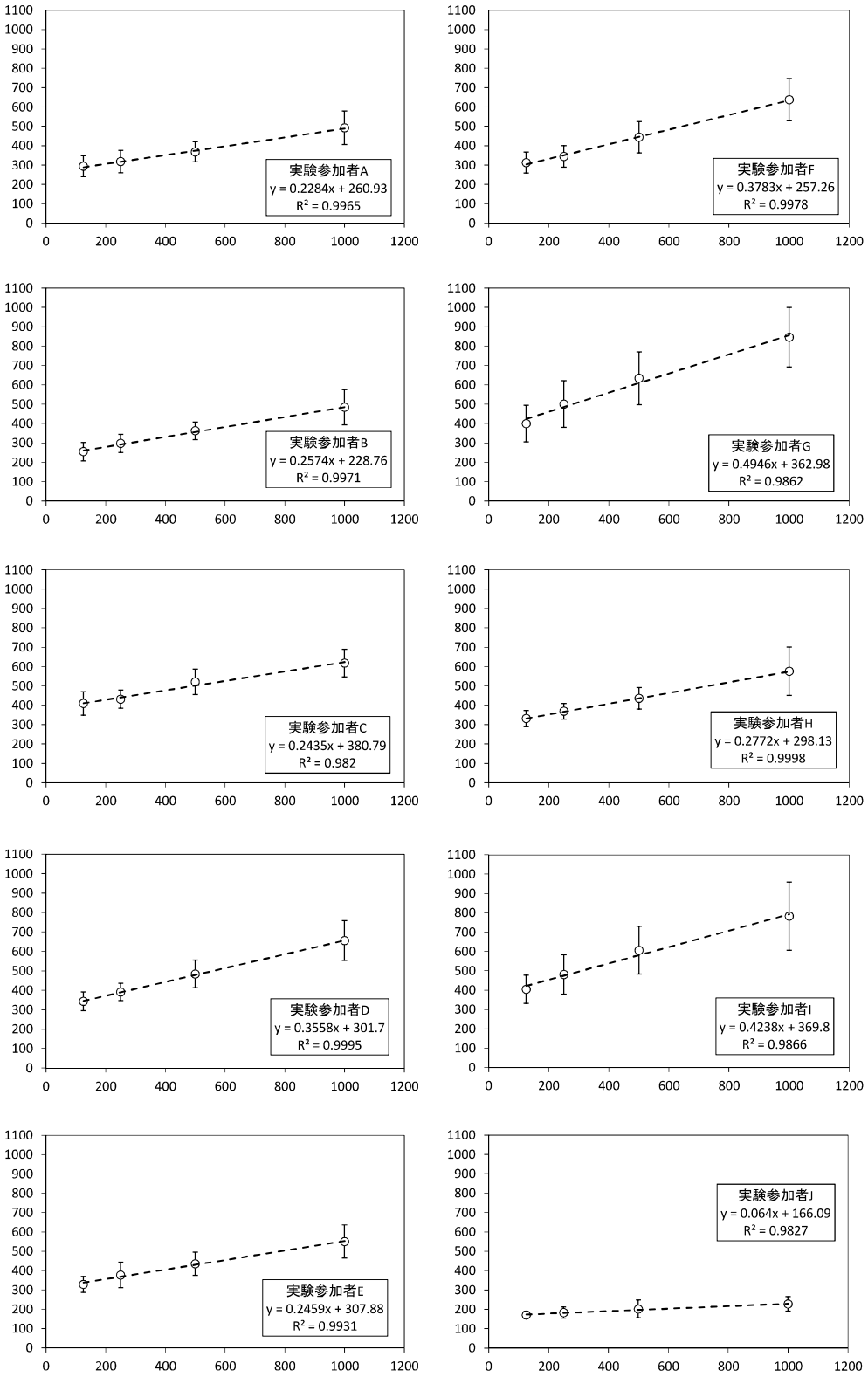


図 3.10　各実験参加者の *A* に対する *MT* の関係．グラフは全て横軸が *A*，縦軸が *MT*．エラーバーは標準偏差を示す．

3.4　議論と制約

本実験ではパラメータの組み合わせ数を抑えるために A を 4 種類としたが，近似式から MT を予測できると強固に裏付けるためには，より多くの A の値で実験する必要がある．また本実験では A が 125 から 1000 の範囲であったが，フル HD 以上の解像度をもつディスプレイも一般的に利用されるようになってきた現代では，さらに長距離のデータでも計測する必要があるだろう．より多種類・広範囲の A で実験したときには，本稿とは異なる結果と考察になる可能性もあるため，A = 125, 250, 500, 1000 pixels の 4 種類を採用したことが本実験の 1 つの制約といえる．

また本実験ではマウスの移動方向として左右の 2 種類を採用したが，マウスは移動方向によって操作時間 [Boritz, 1991] [Bützler, 2015]や操作精度 [Aceituno, 2013]が異なることが知られている．先行研究では移動方向を上下 [Appert, 2008]あるいは斜め方向 [Accot, 2003] [Asano, 2005]に設定した事例があるが，本実験では伝統的なフィッツの法則のタスクを模した左右方向のポインティングを設定した．無限大サイズのターゲットポインティングでは移動方向による腕の動かしやすさは強く影響すると考えられるため，今後の課題として上下や斜め方向の移動を含めた場合の検証が残されている．それと同時に，今回の知見はマウスを使用したときの結果を基にしている．たとえばタッチパッドではマウス操作よりもカーソルに勢いをつけられないし，またトラックボールはより勢いをつけられるため，本実験結果とは異なる知見が得られると考えられる．よって今回の考察は，実験に採用したパラメータに加えて，使用したデバイスに依存した結果を基にしているという制約がある．

本実験結果から得られた知見をまとめると以下のようになる．

(1) 可視領域のサイズ W_v は操作時間 MT に主効果を及ぼさない．これは W のみを無限大にした Appert らの実験 [Appert, 2008]や Farris らの実験 [Farris, 2003]とは異なる結果である．

(2) カーソル速度の変化を時系列的に観察したとき，サイズの規定されたターゲットをポインティングする場合とは異なる波形を示す．具体的には，一般的なポインティングでは操作時間の前半にピークをもつのに対し，無限大サイズのターゲットではピークが前半に現れない場合があった．

(3) MT は A と線形の関係がある (R^2 = .999).

これらのうち特に (3)について，現時点では適合度の高い近似方法を探索し，1 次関数が最もフィットすることを発見したにとどまる．これについて，人間の運動特性に基づいたモデルを構築することで数式の妥当性を示したり，あるいは線形の関係をもつことから逆算的に人間の運動特性を求めるのが今後の課題である．

先行研究と本研究の知見を基にすると，GUI のデザインや，作業時の戦略に関していくつかの指摘ができる．まず，H が規定されている場合には W_v が大きいほど操作時間が短縮

される [Appert, 2008] [Farris, 2003]ため，たとえばタスクバーであれば「厚み」のあるデザインにした方が操作時間が短くなる．しかし W_v を大きくするとそれだけ画面を占有するため，そのトレードオフを考慮して W_v のサイズを設計しなければならない．一方で H が無限大であれば W_v は操作時間に影響しないため，図 3.2d のように画面の左右にスナップしたウィンドウの画面端側の縁のサイズは本実験の結果から考えると 8 pixels 以下であっても問題ないと考えられる．このように，W と H の両方を無限大にしたポインティングタスクの実験は先行研究と異なる結果が得られ，またそれによって GUI の設計指針も変わる可能性があるといえる．

本研究では，図 3.2a-d のような操作例を抽象化したタスクで実験を行った．これによって上記(1)〜(3)の知見が得られたが，実際の PC 作業とは少なからず乖離があることは指摘しなければならない．たとえば図 3.2a のようにデスクトップにファイルを一旦移動するような操作で，本当に W_v が操作時間 MT に影響しないかというと，これは新たな実験条件による調査が必要であると考える．本実験では第 3.3.1.2 章「カーソルを試行ごとに画面中央へ戻す処置について」で述べたように，操作後のカーソル位置を考慮せずにマウスを動かせる条件を課した．これによって，カーソルを画面端まで移動させるような挙動も観察されたが，実際には「運動量が軽減された方がよいので画面端まで移動したくない」，「画面内を何度も往復するような作業全体の時間を短縮するなら別の戦略をとる」などという事情も存在するはずである．本研究では，無限大の W と H をもつターゲットポインティングの分析の第一段階として作業のコンテクストを排除した実験を行ったが，こういった「流れ」を考慮した場合の粗いポインティング操作の分析が今後の課題である．

3.5 おわりに

本稿では無限大のサイズ W と H をもつターゲットのポインティングに着目し，ターゲットまでの距離 A, 可視領域のサイズ W_v, 移動方向 Dir を変化させたタスクで実験を行った．カーソルの軌跡を分析すると，サイズの規定されたターゲットのポインティングでは速度のピークが操作時間の前半に出現するのに対し，無限大サイズのターゲットではそのような性質を持たないことがあった．速度のピークが操作時間の中腹にあったり，あるいは終盤まで加速し続ける挙動も観察された．さらに，操作時間 MT は距離 A と線形の関係があり，A から高精度に値を予測できることを示した．今後はより広い範囲・多くの種類の距離 A，ポインティングデバイスで実験するとともに，MT が A の 1 次式で表せることを説明可能なモデルを構築したいと考えている．

第4章　幅の変化する経路を逆向きに通過するステアリングタスクのモデル化

4.1　はじめに

ステアリングの法則を導出した Accot らの文献 [Accot, 1997]において，図 4.1a のように幅が狭まる経路（Narrowing Tunnel）の通過時間を予測するモデルが提案され，実験で精度の高さが示されている．著者はこのモデルを追検証する実験において，幅の狭まる経路を図 4.1b のように逆方向に通過すると操作時間が短縮されることを発見した．つまり，全く同じ形状の経路であっても幅の変化方向（狭まる方向，広がる方向）によって通過時間が変化するのである．

この通過時間の差について，幅の変化方向の間に何らかの関係があるのか，あるいは無関係なのかについて検証された文献は見当たらない．たとえば狭まる方向は広がる方向に比べて一定値だけ難易度 *ID* が大きい，あるいは一定割合だけ大きいといった関係が判明すれば，逆方向への通過時間を容易に予測できるようになる．そこで本章では，狭まる方向と広がる方向の通過時間の関係を理論的に求め，実験でその妥当性を検証する．

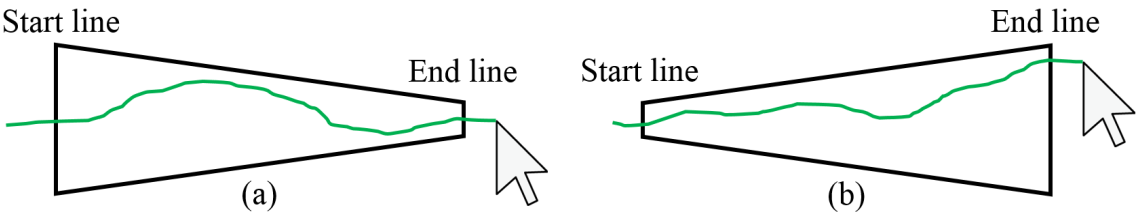


図 4.1　幅が狭まる経路(a)と広がる経路(b)の通過.

4.2　ステアリングの法則のモデル導出と通過方向による *ID* の差

ここでは Accot らが提案したステアリングの法則 [Accot, 1997]のうち, 一定幅の線形経路と幅が狭まる線形経路の導出方法を説明し，幅の変化方向によって通過時間に差が生じる理由を考察する．

4.2.1 ステアリングの法則のモデル導出方法

4.2.1.1 一定幅の直線状経路

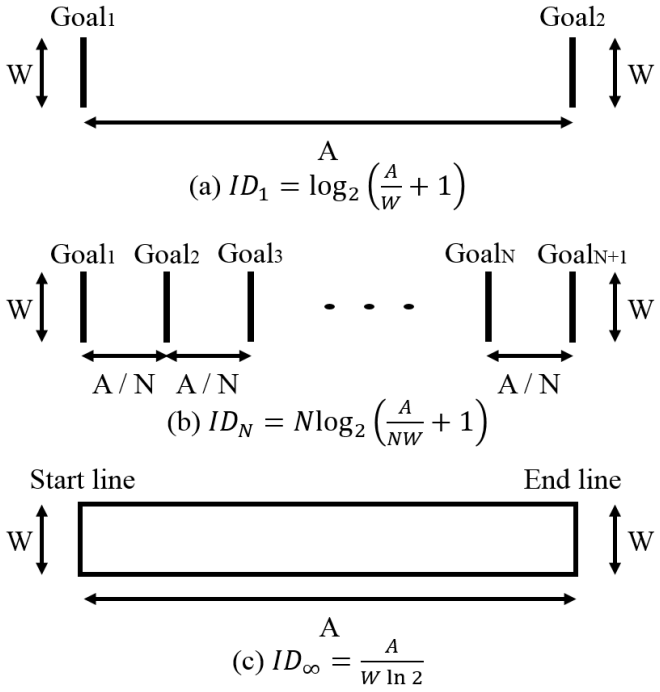


図 4.2　一定幅で直線形状の経路の難易度.

図 4.2 (a)のように，幅 W の線分が距離 A だけ離れて 2 本設置されているとき，これらを通過する時間 MT_1 は,

$$MT_1 = a + b \times ID_1, \qquad ID_1 = \log_2\left(\frac{A}{W}+1\right) \tag{4.1}$$

になることが実験で示されている．a と b は実験により決定される定数である．これを基にして，図 4.2 (b)のように距離 A を N 等分する位置に幅 W の線分を設置すると，その全てを通過するための難易度 ID_N は [Accot, 1997]によれば,

$$ID_N = N\log_2\left(\frac{A}{NW}+1\right) \tag{4.2}$$

となる．ここで図 4.2 (c)のように $N \to \infty$ とすると，$ID_\infty = A/(W \ln 2)$となり，ln2 を定数 b に吸収させれば,

$$MT_\infty = a + b \times ID_\infty, \qquad ID_\infty = \frac{A}{W} \tag{4.3}$$

となる．つまり無限個設置された線分を全て通過する時間 MT は，

$$MT = a + b \times \frac{A}{W} \tag{4.4}$$

で求められる．これは一定幅の経路を通過するタスクと同じ動作であるから，幅 W，距離 A の経路を通過するステアリングタスクの所要時間は式 (4.4)で求められる．

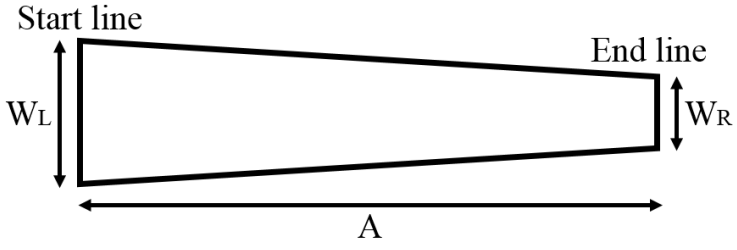
4.2.1.2 幅が狭まる直線状経路

図 4.3a のように，距離 A で幅が W_L から W_R に狭まる経路の通過時間 MT_{NT} (MT for Narrowing Tunnel)は式(4.4)を基に算出される．図 4.3b のように経路を微小区間 dx ごとに分割し，短距離のステアリングを連続して行う動作と見なせば，MT_{NT} は次のように表される．

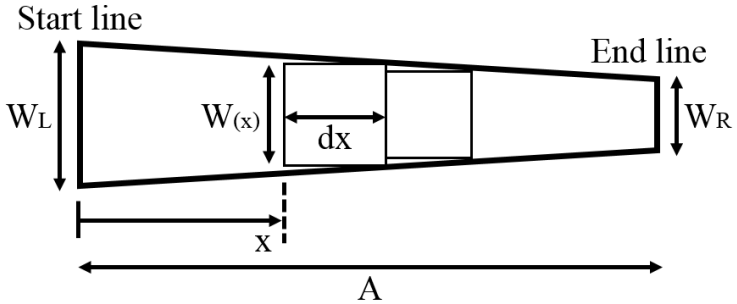
$$MT_{\mathrm{NT}} = a + b \times ID_{\mathrm{NT}} \tag{4.5}$$

$$ID_{\mathrm{NT}} = \frac{A}{W_R - W_L} \ln \frac{W_R}{W_L} \tag{4.6}$$

Accot らは MT_{NT} を幅が狭まる方向へ通過する時間であると定義し，実験でも狭まる方向へのみ移動するタスクでモデルの正確さを検証している．



(a) 狭まる経路



(b) $ID_{\mathrm{NT}} = ID_{\infty} = \int_0^A \frac{dx}{W(x)} = \int_0^A \frac{dx}{W_L + \frac{x}{A}(W_R - W_L)}$

$$= \frac{A}{W_R - W_L} \ln \frac{W_R}{W_L}$$

図 4.3　幅が狭まる直線形状の経路の難易度.

4.2.2 幅が広がる方向への移動

ここで著者が着目したのが，幅が広がる方向へ通過する難易度 ID_{WT} (*ID* for Widening Tunnel)も式(4.6)と同じ構造になる点である．図 4.3 (a)の経路を左向きに通過する難易度を ID_{WT} とすれば，

$$ID_{\mathrm{WT}} = \frac{A}{W_L - W_R}\ln\frac{W_L}{W_R} = \frac{A}{W_R - W_L}\ln\frac{W_R}{W_L} \tag{4.7}$$

となり，$ID_{\mathrm{WT}} = ID_{\mathrm{NT}}$ である．しかし実際には幅が広がる方向へ移動した方が短時間になるため，この導出方法では見落とされているポイントがあると考える．

著者が文献を調査した限りでは，幅が変化する経路の通過方向を切り替えた実験は見当たらず，操作時間に差が生じることがこれまで発見されなかったのだと推測する．Accot らのオリジナルの実験 [Accot, 1997]では幅が広がる螺旋経路で実験されているものの，これは幅が狭まる向きで実験されておらず，やはり同一経路で通過方向だけが異なるタスクは比較されていない．

4.3 逆方向への通過時間を予測する難易度差 ID_{Gap}

ポインティングタスクは加速と減速を幾度も繰り返しながら目標地点に到達することが知られている [Asano, 2005] [Pasqual, 2014]．著者は，狭まる経路のステアリングタスクも同様に, 少しずつ方向と速度を修正しながら図 4.4 のような移動を繰り返していくと考えた．一方で Accot らが提案したモデル [Accot, 1997]は，狭まる経路を N 分割してそれぞれの微小区間の合計 *ID* を求め, $N \to \infty$とすることで経路全体の難易度 ID_{NT} を決定していた. 図 4.4 のような移動方向の修正が，特定の距離を移動するたびに発生するのか，あるいは時間ベースで発生するのかは定かではない．しかし $N \to \infty$としたモデルは修正作業を無限回行うことになり，通過方向による難易度に差がないモデルが導出される．

1 つの微小区間でどの程度カーソル座標のずれ（図 4.4 であれば縦方向の揺れ動き）が生じてよいかを考えると，実際には Accot らが提案するような一定のものではなく，微小区間のエンドライン側の幅に依存する．なぜならカーソルは微小区間に進入した時点でエンドライン方向への速度をもっているため，進入直後にスタート側の経路幅いっぱいにまでずれてしまうとは考えにくいからからである．したがって，広がる経路では図 4.5a のように微小区間の広い方（エンドライン側）の幅いっぱいにまでカーソルがずれてよいのに対し，図 4.5a の狭まる経路では広い方（スタートライン側）の幅を有効に活用できず，通過する難易度が上昇する．当然ながら，図 4.5a では微小区間のスタートラインの幅が図 4.5b より狭いため，微小区間に進入するための時間は(b)より長いと考えられる．しかし経路全体で

考えると，広がる経路で最も通過が困難なのはスタートラインであり，そのための微調整に要する時間はステアリングタスクの所要時間には含まれない．よって全体の通過時間は広がる経路の方が短い($MT_{WT} < MT_{NT}$)というのが著者の仮説である．もちろん微小区間の距離 dx を限りなく小さくすれば Accot らのモデルと一致するが，移動する間の修正回数が有限だとすれば，通過方向によって難易度に差が生じることになる．

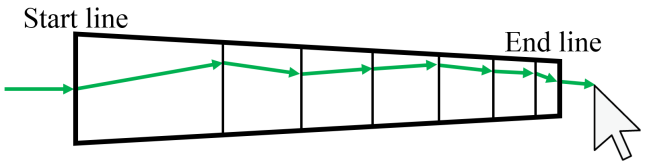
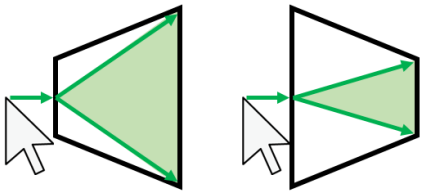


図 4.4　少しずつ方向を修正しながら狭まる経路を通過する運動.



(a) 広がる経路　　(b) 狭まる経路

図 4.5　微小区間で許される誤差範囲.

より具体的に，たとえば経路内で $N = 3$ 回の修正作業が等距離で起こると仮定すると，図 4.6 a のように幅が狭まる経路を右向きに通過する難易度 $ID_{NT(3)}$は，

$$\begin{aligned} ID_{\mathrm{NT}(3)} &= \frac{A/3}{(2W_L - W_R)/3} + \frac{A/3}{(W_L - 2W_R)/3} + \frac{A/3}{(3W_R)/3} \\ &= \frac{A}{2W_L + W_R} + \frac{A}{W_L + 2W_R} + \frac{A}{3W_R} \end{aligned} \tag{4.8}$$

である．これに対し，修正回数が $N = 3$ の場合の狭まる経路の通過(b)のように幅が広がる方向へ通過する難易度 $ID_{WT(3)}$は，

$$ID_{\mathrm{WT}(3)} = \frac{A}{3W_L} + \frac{A}{2W_L + W_R} + \frac{A}{W_L + 2W_R} \tag{4.9}$$

となる．この難易度の差を $ID_{Gap(3)}$とすると，

$$\begin{aligned} ID_{\mathrm{Gap}(3)} &= ID_{\mathrm{NT}(3)} - ID_{\mathrm{WT}(3)} \\ &= \frac{A}{3W_R} - \frac{A}{3W_L} = \frac{A(W_L - W_R)}{3W_L W_R} \end{aligned} \tag{4.10}$$

となる．分割数を一般的に N とした難易度差 $ID_{Gap(N)}$は次のようになる．

$$ID_{\mathrm{Gap}(N)} = \frac{A(W_L - W_R)}{NW_L W_R} \tag{4.11}$$

すなわち N は経路のパラメータをどの程度難易度差に反映させるか決める役割を果たす．

ここまでは距離 A を N 等分するという仮定で難易度差を算出したが，実際には距離や時間に関して等間隔に修正作業が発生するわけではなく，幅の変化度合いや距離 A，操作デバイスによって異なる値をとるはずである．したがって式(4.11)の等分割数 N を実験条件によって決定される重み (free weight [Accot, 2003]) k に置き換え，

$$ID_{\mathrm{Gap}(k)} = \frac{A(W_L - W_R)}{kW_L W_R} \tag{4.12}$$

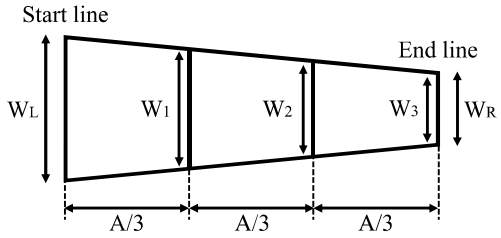
としたものが通過方向によるステアリングタスクの難易度差である．これは Accot らの提案するモデルの ID である式(4.6)とは異なる式の構造をもっているため，一定値または一定割合だけ差があるという第 4.1 章で挙げた例とは別の関係が示されている．

式(4.12)で $k \to \infty$とすれば $ID_{\mathrm{Gap}(\infty)} = 0$ となり，難易度は幅の変化方向に関係ないとする Accot らの導出方法と合致する．これは修正作業を無限回行うことになり，通過方向による難易度の差を適切に表せないと考える．逆に $k \to 0$ とすると $ID_{\mathrm{Gap}(0)} = \infty$となり，これも難易度差として不適切である．よって難易度差を最も適切に表す k の値 $(0 < k < \infty)$が存在すると考えられる．

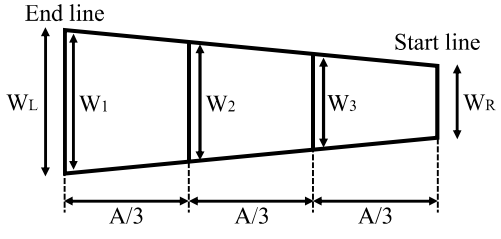
この難易度差 ID_{Gap} を利用する方法として，次の 2 つが考えられる．

- 狭まる経路の通過難易度 ID_{NT} を $ID_{\mathrm{NT}} + ID_{\mathrm{Gap}}$ に補正して，通過時間 MT_{NT} を予測する
- 広がる経路の通過難易度 ID_{WT} を $ID_{\mathrm{WT}} - ID_{\mathrm{Gap}}$ に補正して，通過時間 MT_{WT} を予測する

本章では，一方向に通過する時間から他方向の時間を予測するという利用例を想定しているため，操作時間が長い狭まる方向への通過時間 MT_{NT} を計算で求められた方が利得が大きいと考え，第 4.6 章の考察では前者を採用する．



(a) $ID_{\mathrm{NT}(3)} = \frac{A/3}{W_1} + \frac{A/3}{W_2} + \frac{A/3}{W_3} = \frac{A}{2W_L+W_R} + \frac{A}{W_L+2W_R} + \frac{A}{3W_R}$



(b) $ID_{\mathrm{WT}(3)} = \frac{A/3}{W_1} + \frac{A/3}{W_2} + \frac{A/3}{W_3} = \frac{A}{3W_L} + \frac{A}{2W_L+W_R} + \frac{A}{W_L+2W_R}$

図 4.6　修正回数が N=3 の場合の狭まる経路の通過．

4.4　実験

提案する難易度差の妥当性を検証するために，通過方向を切り替えて移動時間を計測する実験を行う．以下に詳細を述べる．

4.4.1 タスクと教示

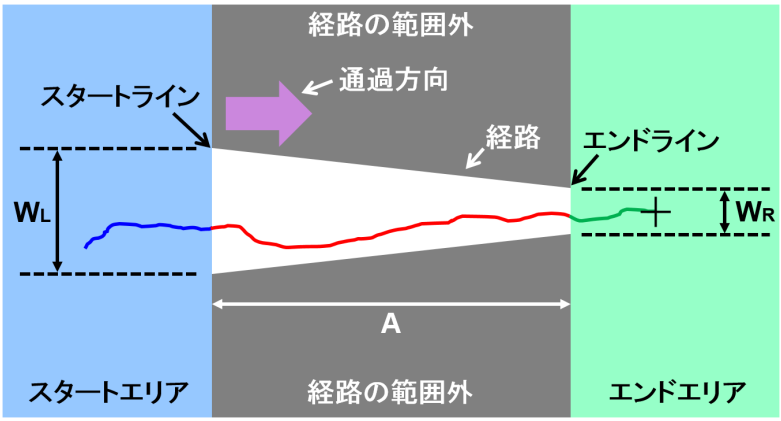


図 4.7　実験の画面構成．狭まる経路を右向きに移動する場合．

タスクが開始されると，液晶ペンタブレット上に図 4.7 のように色分けされた画面が表示される．実験参加者はスタートエリア内にスタイラスを接地させ，ピンク色の矢印で指定された方向へスタートラインを通過する．その後，スタイラス先を経路から出ないように移動させてエンドラインを通過する．エンドエリアからスタイラスを離すと次のパラメータが選出されて画面に表示される．

スタイラスをタブレット面に近接させると，黒色で長さ 25 ピクセルの十字型カーソルが表示される．スタートエリア内ではカーソル座標の軌跡が青色の線で描画される．スタートラインを通過した後は軌跡が赤色になり，データの計測開始がフィードバックされる．エンドラインを通過するとベル音が鳴って試行成功が通知され，エンドエリアでは軌跡が緑色になる．また経路外にはみ出た軌跡は明るいオレンジ色で描画される．

カーソルが経路からはみ出したときと，スタイラスをタブレット面から浮かせてしまった場合にはビープ音が鳴り，試行が中断される．実験参加者はスタイラスを離してから再びスタートエリア内に接地させ，タスクをやり直す．ただしエラーにカウントされるのは経路からはみ出したときだけである．スタイラスを離してしまうのは，本実験で観察したい幅変化の影響とは異なる性質のミスであると考え，エラーとしない．これは文献 [Accot, 2001]と同様の条件である．実験参加者には可能な限り速くかつミスせず操作するよう教示する．

4.4.2 使用機器等

PC は Sony VAIO Z SVZ1311AJ（2.1 GHz×4 コア，8 GB RAM）を使用した．ディスプレイおよび入力デバイスは液晶ペンタブレット Wacom Cintiq 12WX（IPS 液晶，261.12×163.2mm，1280×800 ピクセル）であり，HSP 3.32 で実装した実験システムをフルスクリーンで表示する．システムは約 125 Hz で動作する．

4.4.3 実験参加者

実験参加者は情報系の大学生及び大学院生の 11 名（男性 11 名，平均 21.9 歳，SD = 2.27 歳）である．全員が利き手の右手でスタイラスを操作した．スタイラス操作に習熟している参加者が 1 名おり，ペンタブレットを 5 年間，液晶ペンタブレットを 1 年間ほど継続利用していた．先行研究 [Accot, 2001] [Kulikov, 2005]ではスタイラス操作に習熟した参加者がいなかったが，ステアリングの法則は成立していたため，本実験でも全員がスタイラスを使用する．

4.4.4 手順

距離 A は 300, 600 ピクセルである. 経路の幅は W_L, W_R ともに 11, 31, 51 ピクセルであり，$W_L \neq W_R$ になる組み合わせだけ選出される．またスタートラインからエンドラインへ移動する左右方向 Movement direction (MD)は *right*，*left* である．それぞれのパラメータは文献 [Accot, 1997] [Accot, 1999] [Accot, 2001] [Dennerlein, 2000] [Kulikov, 2005]を参考に設定した．通過方向による幅の変化（狭まる方向，広がる方向）を Navigation direction (ND)と呼ぶことにすると，ND は W_R, W_L, MD の組み合わせによって決定される．

パラメータの組み合わせは合計で 2 (A) × {3(W_L) × 3(W_R) - 3(左右が同じ幅になる組み合わせを排除)} × 2 (MD) = 24 種類あり，ID_{Gap} による補正をする前の ID は（つまり Accot らの式(4.6)による難易度）は 7.47 から 31.1 までの範囲である．これら 24 パターンがランダムに選出させるのを 1 ブロックとし，実験参加者は合計 6 ブロック試行する．第 1 ブロックが練習，続く 5 ブロックが本番である．本番で記録されるデータは 24 パターン×5 ブロック×11 名 = 1320 回分である．練習時に椅子の高さやタブレットの角度などを調整させ，また各ブロックの間には 30 秒間の休憩を設けて，参加者がタスクに集中できるようにした．所要時間は，事前のインストラクションから全試行終了まで 20 分程度である．その後，実験の所感を尋ねるアンケート調査を行った．

4.4.5 計測するデータ

スタートラインを通過してからエンドラインを通過するまでの時間 *MT*，エラー率，タイムスタンプ付きのカーソル座標の軌跡を記録する．また軌跡のデータを基に，y 軸座標の標準偏差 SD_y を算出する．これは y 軸方向のスタイラスの揺らぎを観察することで，タスクがどの程度慎重な操作を要するのかを分析するために用いる．実効幅を利用した分析手法[Kulikov, 2005]は一定幅の経路に対して効果が議論されており，幅が変化する経路に対してそのまま適用可能かは不明であるため，ここでは採用しない．

4.4.6 結果

距離 *A* [Accot, 2001]，幅 *W* [Accot, 2001] [Kulikov, 2005]によって操作時間 *MT* に差が生じることが既知であるため，ここでは本研究で焦点を当てている幅の変化の影響について分析するべく独立変数は *A* および *W* を統合した *ID* を利用する．分析方法は，難易度 *ID*，左右の移動方向 *MD*，幅の変化方向 *ND* を要因とした参加者ごとの対応あり三元配置分散分析（反復測定）を用いる．多重比較には Bonferroni の手法を用いる．*MT* と SD_y の分析からはエラーを起こした試行を取り除いている．

4.4.6.1 操作時間 *MT*

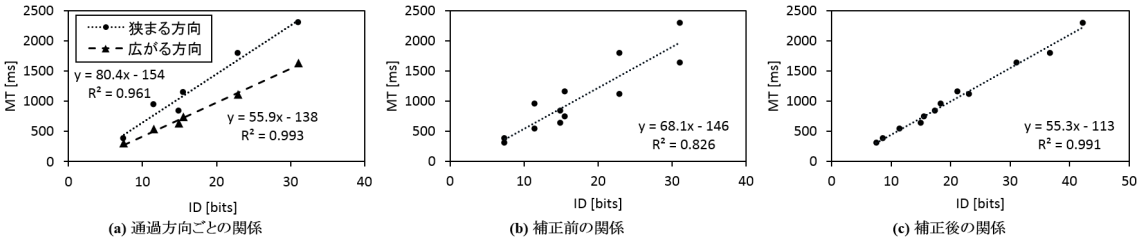


図 4.8　*ID* に対する *MT* の関係．

図 4.8a に幅の変化方向 *ND* ごとの *ID* と *MT* の関係を示す．プロットされた各点は，*ID* 以外の要因の 110 回の試行の平均値を示す（5 ブロック × 左右 × 11 名）．*ND* ごとに回帰分析すると，決定係数 R^2 がそれぞれ .96 を超えている．この決定係数 R^2 は，方程式の当てはまり具合(fitness)，つまり $MT = a + b \times ID$ で描かれる直線とプロットされた点とのずれの小ささを表す．ここでは狭まる方向/広がる方向ともに，ステアリングの法則が強く当てはまっているといえる．

主効果が見られたのは，難易度 *ID* ($F_{5, 50} = 29.449$, $p < .001$)，幅の変化方向 *ND* ($F_{1, 10} = 23.667$,

p < .01)であった．幅の変化方向 *ND* ごとの平均操作時間は，狭まる方向が 1233 ms，広がる方向が 826 ms であり，多重比較で有意差が確認された ($p < .01$)．左右方向 *MD* による主効果は確認されなかった ($F_{1, 10} = 0.083$, $p = .780$).

4.4.6.2 エラー率

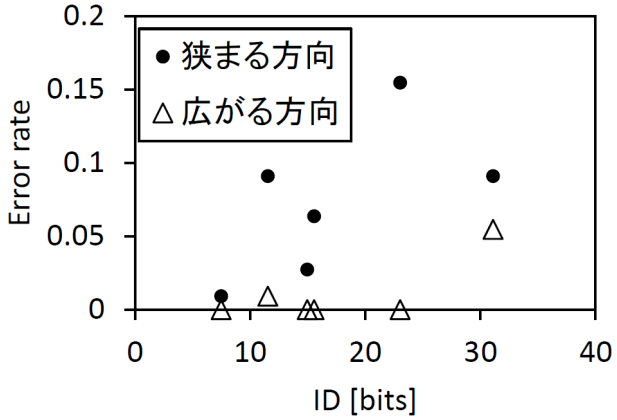


図 4.9　*ID* に対するエラー率.

図 4.9 に *ID* に対するエラー率を示す．主効果が見られたのは，難易度 *ID* ($F_{5, 50} = 4.204$, $p < .01$)，幅の変化方向 *ND* ($F_{1, 10} = 12.111$, $p < .01$)であった．幅の変化方向 *ND* ごとのエラー率は，狭まる経路が 6.78%（708 回中 48 回），広がる経路が 1.05%（667 回中 7 回）であり，多重比較で有意差が確認された ($p < .01$)．左右方向 *MD* による主効果は確認されなかった ($F_{1, 10} = 0.040$, $p = .846$)．同一の経路であっても，幅が狭まる方向への移動は操作ミスが増加することがわかった.

4.4.6.3 軌跡の y 軸座標の標準偏差 SD_y

カーソルの軌跡の y 軸座標の標準偏差 SD_y について, 主効果が見られたのは, 難易度 *ID* ($F_{5, 50} = 18.208$, $p < .001$), 左右方向 *MD* ($F_{1, 10} = 11.017$, $p < .01$), 幅の変化方向 *ND* ($F_{1, 10} = 11.176$, $p < .01$)であった．左右方向 *MD* に関する多重比較では，左向きは右向きよりも SD_y が大きかった ($p < .01$)．また幅の変化方向 *ND* ごとの SD_y は，狭まる方向が 2.46 pixels，広がる経路が 2.85 pixels であり，多重比較で有意差が認められた ($p < .01$)．したがって同一の経路であっても, 広がる方向への移動はカーソルの y 軸方向への揺れ動きが大きいことが示された.

4.5　考察

4.5.1 *MT*，エラー率，SD_y に関して

全く同一の経路であっても，狭まる方向へ通過した方が SD_y が小さかった．つまり y 軸方向の揺れ動きが小さく，広がる方向よりも慎重な操作を要しているといえる．これが狭まる方向の操作時間 *MT* が長かった要因の 1 つであると考えられる．事後アンケートでは，「出口側が広いと簡単だった」，「幅が太いと加速できた気がする」という旨の回答をした参加者が 11 名中 4 名おり，幅の変化方向 *ND* によって主観的にも難易度が異なっていると感じられていたことがわかる．

広がる方向への移動は，SD_y が有意に大きいにも関わらず，いずれの *ID* においてもエラー率は低減しており（図 4.9，平均値は 1/6 倍未満），揺れ動きが大きいからといってミスを誘発するほどではないことが確認された．以上をまとめると，幅が広がる方向へ通過する方が操作時間 *MT* が短く，SD_y が大きくなるものの，エラー率は低減することが確認された．

4.5.2 *ID* の補正

狭まる経路の難易度を $ID_{NT} + ID_{Gap}$ に補正し，通過方向に関わらず *MT* を *ID* の関係式で表せるか検証する．図 4.8a では幅の変化方向 *ND* ごとに *MT* の式を求めていたが，これを *ND* の区別なく回帰分析したものが図 4.8b である．$R^2 = .83$ 程度になり，モデルの適合度が低下していることが読み取れる．次に，狭まる方向の難易度 ID_{NT} を，$ID_{Gap(k)}$ を用いて $ID_{NT}+ID_{Gap(k)}$ に補正すると，*k* の値によって R^2 は図 4.10 のように変化し，$k = 3.14$ のときに R^2 が最大の .991 となった．よって幅の変化方向 *ND* による難易度差を最も適切に表す重みは $k = 3.14$ である．

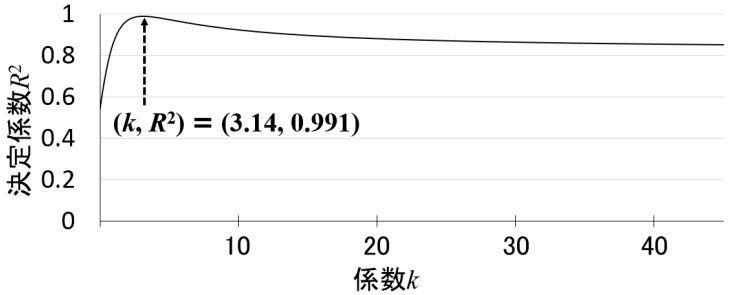


図 4.10　重み *k* によるモデルの一致度．

式(4.12)に k = 3.14 を代入して $ID_{\mathrm{Gap}(3.14)}$を求めると，各 *ID* は表 4.1 のように補正される．図 4.8c に補正後の *ID* と *MT* の関係を示す．前述の通り，R^2 がきわめて高い .991 となったことから，通過方向ごとの難易度 *ID* には式(4.12)の関係があるといえる．言い換えると，狭まる方向の *ID* を適切な k で補正すれば，広がる方向への移動時間のグラフとほぼ一致するということである．今回の実験結果であれば，図 4.8a の広がる方向の元の *ID* を補正した次式で k = 3.14 とすると，狭まる方向への移動時間と近い数値が得られる．

$$MT = -138 + 55.9 \times (ID + ID_{\mathrm{Gap}(k)}), \qquad ID_{\mathrm{Gap}(k)} = \frac{A(W_L - W_R)}{kW_L W_R} \tag{4.13}$$

ただし k は実験条件によって異なる値をとりうるため，ここで算出した k = 3.14 が常に最適な値であるとは限らない．k = 3.14 は実験参加者を含めた今回の実験条件における最適値である．

表 4.1　実験パラメータごとの補正前・補正後の *ID*.

A	一方の *W*	他方の *W*	補正前の *ID*	補正後の *ID*
300	31	51	7.47	8.68
300	11	51	11.5	18.3
600	31	51	14.9	17.4
300	11	31	15.5	21.1
600	11	51	23.0	36.6
600	11	31	31.1	42.3

4.5.3 高い *ID* における *MT* の推定

実験結果では k = 3.14 のときに最もモデルの適合度が高くなった．では次に，実験の工程を削減しても操作時間が予測できる利得について考察する．まず，より簡単な（短時間で計測が終わる）広がる方向への通過時間を全ての *ID* で測定し，図 4.8a のようにグラフを描く．これにより，広がる方向の通過時間には *MT* = -138 + 55.9 × *ID* の関係があることが導かれる．次に，狭まる方向は 1 種類の *ID*（最短で計測できる *ID* = 7.47）だけで測定する．そして *ID* = 7.47 における広がる/狭まる方向の操作時間 *MT* から，（最適であるか不明な）k の値を算出し，他の *ID* における狭まる方向への *MT* をその k の値を用いて予測する，という方法である．具体的な計算方法は次のようになる．

まず狭まる方向の操作時間 *MT* は, *ID* = 7.47 において 375 ms であった. 式(4.13)で *MT* = 375 となるための $ID_{\mathrm{Gap}(k)}$ は,

$$MT = -138 + 55.9 \times \left(7.47 + ID_{\mathrm{Gap}(k)}\right) \;\Leftrightarrow\; ID_{\mathrm{Gap}(k)} = 1.71 \tag{4.14}$$

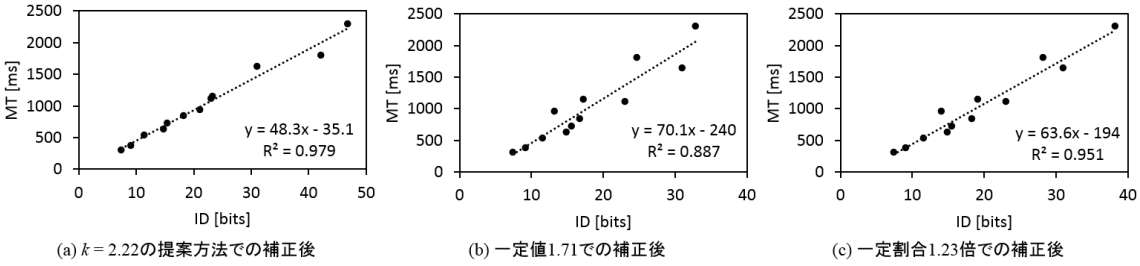
である．式(4.12)から k を求めると，

$$1.71 = \frac{300(51-31)}{k \cdot 51 \cdot 31} \quad \Leftrightarrow \quad k = 2.22 \tag{4.15}$$

となる．次に ID = 7.47 以外の 5 種類の ID について，式(4.13)に k = 2.22 を代入して ID を補正すると，表 4.2a のような値が得られる．この補正後の ID と MT との関係は図 4.11a のようになり，R^2 = .979 の高い決定係数が得られた．つまり，未実験のパラメータにおける難易度 ID を，実験済みのデータ（広がる方向の 6 種類の ID と，狭まる方向の 1 種類の ID における操作時間 MT）から推測した場合も，実測の操作時間 MT との関係は適切に表せるといえる．よって，最も容易な ID だけで実験を行っても，狭まる方向の未実験の ID における操作時間 MT を高精度に予測可能であると確認された．

表 4.2　各補正方法による ID．(a) k = 2.22 での提案方法の式による補正，(b) 一定値 1.71 の差による補正，(c) 一定割合 1.23 倍の差による補正．

補正前の ID	(a) 提案方法	(b) 一定値	(c) 一定割合
7.47	9.18	9.18	9.18
11.5	21.1	13.2	18.3
14.9	18.3	16.6	19.1
15.5	23.5	17.2	19.1
23.0	42.5	24.7	28.3
31.1	46.9	32.8	38.2



(a) k = 2.22の提案方法での補正後　(b) 一定値1.71での補正後　(c) 一定割合1.23倍での補正後

図 4.11　各方法で ID を補正した後の MT との関係．

先行研究 [Accot, 1997] [Accot, 2002]によれば，モデルの決定係数 R^2 が高いことを根拠に，「ステアリングタスクは $MT = a + b \times ID$ のモデルで説明できる」，「線分をクロスするタスクでは，MT と ID の間に強固な規則性がある」と主張されている．本実験結果を同様に考察すると，最適な k の値をとるときに R^2 = .991 の高い決定係数が得られ，また最低 ID から求めた k の値を用いたデータも R^2 = .979 で方程式がフィットしていたことから，幅の変化方向 ND による難易度差は式(4.12)で表せると考えられる．

4.5.4 一定値の差があると仮定した補正方法

提案する *ID* の補正方法を，その他に考えうる補正方法と比較して議論する．幅の変化方向 *ND* による難易度差を検証した研究はこれまでに存在しないため, ここでは最も簡易な補正方法として一定値の差，および一定割合の差があると仮定した補正方法を取り上げる．まず幅の変化方向 *ND* によって一定値だけ難易度の差 ID_{Const} があると仮定した場合を検討する．最低 *ID* = 7.47 において，狭まる方向の実測値は *MT* = 375 であるから，補正後の難易度を $ID_{\mathrm{Corrected}}$ とすると，

$$375 = -138 + 55.9 \times ID_{\mathrm{Corrected}} \quad \Leftrightarrow \quad ID_{\mathrm{Corrected}} = 9.18$$

となる．そして補正後の難易度は，（補正前の難易度 + 一定値の差）であるから，

$$9.18 = 7.47 + ID_{\mathrm{Const}} \quad \Leftrightarrow \quad ID_{\mathrm{Const}} = 1.71$$

となる. この難易度差 1.71 を他の *ID* にも加算すると表 4.2b のような補正後の *ID* が得られる．*MT* との関係は図 4.11b のようになった．これは *ID* が大きくなるほど *MT* の差が増大するという性質を反映できておらず, 高い決定係数が得られない (R^2 = .887)ことがわかった.

4.5.5 一定割合の差があると仮定した補正方法

次に一定割合だけ差があると仮定する．つまり，（補正後の難易度÷補正前の難易度）が一定であるとして考える．最低 *ID* では倍率が 9.18 / 7.47 = 1.23 となる．元の *ID* を 1.23 倍した数値は表 4.2c のようになった．また *MT* との関係は図 4.11c のようになり，R^2 > .95 の高い決定係数が得られた.

一定割合の差があるとした補正方法でも高い適合度が得られたため，提案方法と予測精度に差があるか検証する．すなわち最低 *ID* から求めた *k* の値 2.22 を用いて算出した *ID*（表 4.2a）と，同じく最低 *ID* から求めた倍率 1.23 を用いて算出した *ID*（表 4.2c）から操作時間 *MT* を予測し，それが実際に測定された *MT* とどの程度合致しているかを検証する．

予測時間と実際の計測時間の差の絶対値を従属変数，補正方法（一定割合，提案方法）を独立変数としたウィルコクソンの符号順位検定の結果，*ID* = 7.47 以外の 5 点の *ID* の間には有意差が確認されなかった (p = .0679)．図 4.12 は予測時間を横軸，計測された時間を縦軸にプロットしたものである．R^2 が 1 に近いほど予測時間と計測時間が近く，高精度なモデルであることを意味する．提案方法の方がわずかに良好な決定係数を示しているものの，一定割合と提案方法とでは算出される予測時間に有意な差はなく，今回実験した *ID* の個数・範囲であれば，一定割合の差があるとした方法でも遜色ない予測精度が得られるといえる．より広範囲の *ID*，および多くのプロット点における予測精度の比較が今後の課題である.

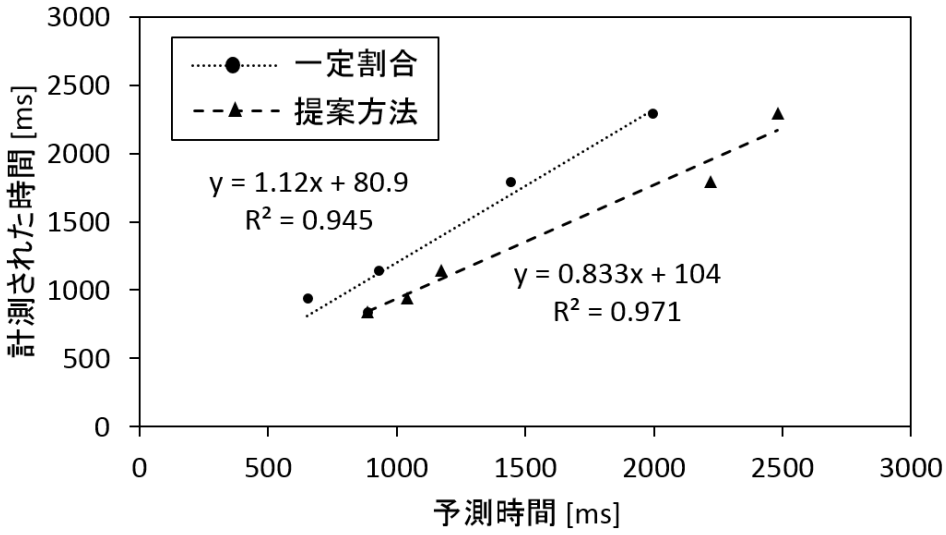


図 4.12　計測された時間と予測時間の関係.

4.5.6 速度波形の分析

図 4.13 に経路内の平均速度の波形を示す．元のステアリングの法則によれば，カーソルの速度は現在位置の経路幅に依存するはずである．つまり，速度が逆転するのは経路のちょうど半分に差しかかった地点のはずである．しかし，A = 300, 600 のいずれにおいても速度が逆転するのは経路の 25～30%付近であることが読み取れる．したがって，カーソル速度は現在位置の幅だけでなく，経路がその後に狭まるのか広がるのかにも依存することがわかる．これは図 4.5 に示した仮説の「広がる方向に通過する方が容易である」を裏付ける結果である．

その他にも興味深い事実が読み取れる．まず，狭まる経路ではスタートラインが最も通過するのが容易なため，経路内の最高速度を記録すると予想される．しかし実際にはスタート時点の速度は経路中で最低であった．次に，狭まる経路ではゴール直前でわずかに速度が上昇している点である．狭まる経路で最も通過が困難なのはゴールラインであり，そこへ向けて速度が上昇するのは一見不自然に思われる．しかしこれも図 4.5 の仮説が正しければ「ゴール直前の速度は，非常に広いゴールエリアに依存して高速になる」と説明できる．最後に，広がる経路では比較的安定して速度が上昇していくのに対し，狭まる経路では上昇→下降→上昇と大きく 3 段階に分かれていることが読み取れるため，幅の変化方向によって異なる戦略がとられていると考えられる．

図 4.14 は各方向の 1 試行ずつのデータを取り出したものである．動きの修正作業（速度の上昇/下降）は等距離で発生しているわけではないことがわかる．また狭まる方向の方が多く発生していることもわかる．さらに現在地点の幅が狭いほど修正回数が多いことも読

み取れる．したがって修正回数は実験パラメータ（幅の変化方向と現在地点の幅）に関係していると考えられるが，これら以外のパラメータや実験参加者ごとの戦略が影響するかについては今後の調査課題としたい．

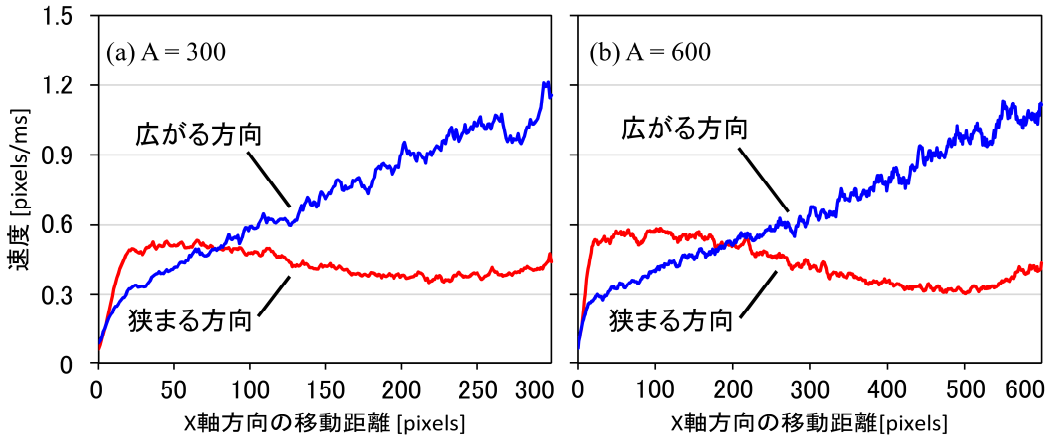


図 4.13　全試行の平均速度の波形．7 点の単純移動平均でフィルタリングしたもの．狭まる方向・広がる方向ともに，グラフの左端は経路の開始側を示す．

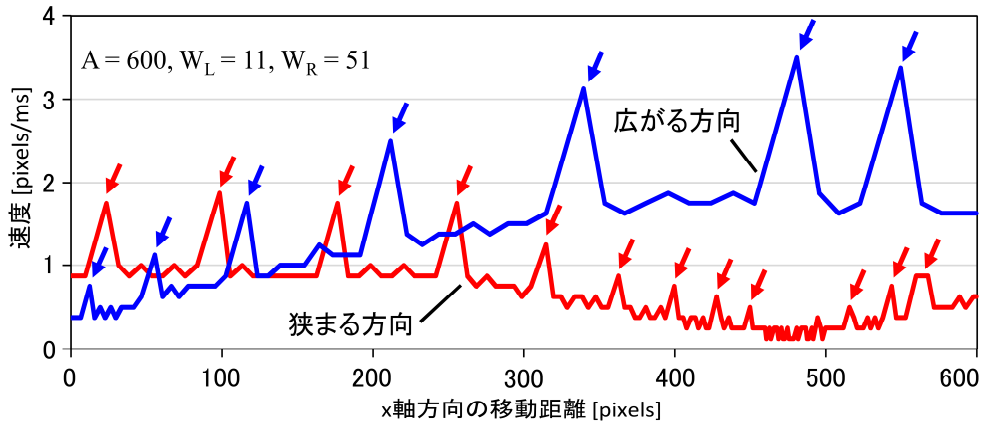


図 4.14　各方向につき 1 試行ぶんの速度波形．同一の実験参加者，同一ブロックにおけるもので，フィルタはかけていない．矢印は目立った速度ピークを示す．

4.6　限界と課題

同一の *ID* であっても，カーソルのサイズ [Naito, 2004] [Naito, 2006]や経路のスケール [Accot, 2001]，使用するデバイス [Accot, 1999]によって操作時間が異なることが知られている．つまり実験条件が異なるときには *ID* を統一的に扱えないことが複数の研究から判明している．本稿では完全に同一の経路・カーソルサイズ・デバイスであっても操作時間が変

化する新たな要素（幅の変化方向 *ND*）があることを発見した．そして差が発生する原因を理論的に求めてモデルを導出し，*ID* の関係が提案モデルに沿うことを実験で確かめた．さらに，一方の通過時間のデータを基にして，逆向きに通過する時間を少ない実験（最も容易・短時間で測定が終わる *ID*）の測定結果から高精度に予測できることを示した．もし逆方向の通過時間を十分に測定できるのであれば，幅の変化方向 *ND* ごとに *MT* と *ID* の関係式を求めればよいため, 本研究は Accot らが最初に提唱したモデル [Accot, 1997]の価値を低減するものではない．今後の課題として，$R^2 > .95$ を保つ最低限の *ID* の個数を求めたり，逆にそれ以上 *ID* の種類を増やしても精度を向上させられない上限数を求めることが挙げられる．また，広がる方向への通過は平均エラー率が有意に低減することから，測定時間には表出しない「測定実験全体の時間」も短縮することが期待でき，この観点での検証にも意義があると考える．

本研究の限界として，高精度に予測できるのはあくまで実験で採用した *ID* の範囲内であることが挙げられる．より高い *ID* では，低い *ID* の実験結果を基に予測すると精度が低下するおそれがある．また一定割合で補正した場合も $R^2 > .95$ と高精度であったが，こちらも同様に高い *ID* での予測精度は未知である．先行研究ではより高い *ID* で狭まる経路を通過する実験を行い，高い決定係数を得ている事例（[Accot, 1997]では *ID* が 10.9 から 76.4 の範囲で $R^2 = .978$）があるため，*ID* の範囲に関わらず予測できることを期待したいが，今回の結果から主張できるのは *ID* が 7.47 から 31.1 の範囲で $R^2 = .971$ の精度が得られたことである．より広範囲の *ID* での予測精度検証は今後の課題としたい.

今回の実験では重み *k* は 3.14 が最適値であったが, これは実験によって決定されるため, 実験環境やユーザによって値が変動しうる．同様に最低 *ID* における *k* の値 2.22 も実験環境によって変わるため，これを異なる実験条件/ユーザにそのまま適用しても操作時間は適切に予測できないと考えられる．本研究以外で重みを用いてモデルを構築した例では，Accot ら [Accot, 2003]が矩形のターゲットをポインティングするタスクにおいて，Accot らが独自に行った実験のデータでは重みを $\eta = 0.137$ とするのが最適だったのに対して，過去の文献のデータに適用した場合は $\eta = 0.32$ であると報告されている．よって，こういった重みはユーザや操作環境などによって値が変動しうるため，*MT* の予測に用いる *k* の値はそれぞれの実験条件下で測定することが必要である．

ステアリングの法則は元々GUI を対象に研究されていたが，バーチャルリアリティ空間において車が道路から逸脱しないように運転するタスク [Zhai, 2004]やレースゲーム [Bateman, 2011]にも適用できると報告されている．これらの文献では一定幅の経路のみが扱われていたが，道幅が変化する道路を採用することでより多彩なタスクを設計できるであろう．しかしその分だけパラメータは増加し，コースのレベルデザインは煩雑になる．もしドライビングタスクにも本稿の提案モデルが適用できれば，道路ごとの難易度設定が容易になると考えられる．たとえば文献 [Zhai, 2004]の実験では 1 回の試行に数十秒～百秒程度を要するタスクが設定されているが，提案モデルを用いて操作時間を予測すればタスク

の設計と検証に要する時間を大幅に削減できる．その他の利用例として，ビデオゲームにおいて操作キャラクタが落ちないように細長い道を歩かせたり，ダメージを受ける両脇の壁に接触しないように自機を操縦するシーンの設計支援が挙げられる．これらもドライビングタスクと同様に，道幅が変化するときの制限時間設定や，往路と復路の難易度をどの程度変えるかを設計する一助になると考えられる．

最後に，今回の実験は 2 次元の GUI 環境で，直接タッチ方式のスタイラスを用いた結果について議論した．カーソルを間接制御するスタイラス入力 [Accot, 1997]やその他のデバイス [Accot, 1999], 3 次元空間 [Casiez, 2004] [Liu, 2011]などで同様の予測方法が有効かは今後の検討課題である．また，環状および螺旋状の経路や，中央が膨らんだ/窪んだ形状をもつ複雑な経路，さらにドライブシミュレータの経路にも今回のような関係が存在するか検証し，提案方法の適用範囲を調査していきたい．

4.7 おわりに

本章では幅が線形に変化する直線状経路を対象に，通過するときの幅の変化方向 *ND*（狭まる方向，広がる方向）によって操作時間 *MT* が変化する原因を理論的に求め，*ND* 間の *ID* の差を導出した．画面を直接操作するタイプのスタイラスで実験した結果，補正 *ID* と *MT* の関係式が回帰分析で高い決定係数 (R^2 = .991)を得たことから，導出した *ID* 差が今回の実験環境において妥当であることを示した．また，広がる方向への通過時間を 6 種類の *ID* において測定し，また狭まる方向への通過時間を最小 *ID* において測定することで，残り 5 種類の *ID* における狭まる方向への通過時間を高精度 (R^2 = .971)に予測できることを示した．

この難易度差モデルを用いることで，従来は同一であると見なされていたステアリングタスクの難易度をより正確に算出できるようになった．また，幅の変化方向 *ND* の間でどの程度の難易度差を設けるかをコントロール可能になったことで，ドライブシミュレータやビデオゲームのレベルデザインが支援できると考えられる．今後は，環状経路や S 字型経路における提案モデルの適合度の検証や，他のポインティングデバイスを使用したときの適合度の検証をしていきたい．

第5章　細長いターゲットのドラッグアンドドロップ支援手法

5.1　はじめに

図 5.1 に示すような，ウィンドウの外枠，あるいはウィンドウ内のレイアウトを変更するための境界線など，細長いターゲットをドラッグアンドドロップ(DnD) する場面では正確な操作が要求される．こういったリサイズやレイアウト変更などの操作は，それ自体に集中して取り組みたいものではなく，メインの作業（ウェブブラウジングやプログラミングなど）に付随して必要になるものである．しかしながら，OS やソフトウェアによってはドラッグできるターゲットの幅が極めて細く設定されていることもあり，ユーザは慎重に操作しなければならない．こういった細長いターゲットを容易にドラッグできる手法を確立することで，WIMP 環境はさらに改善可能であると考えている．

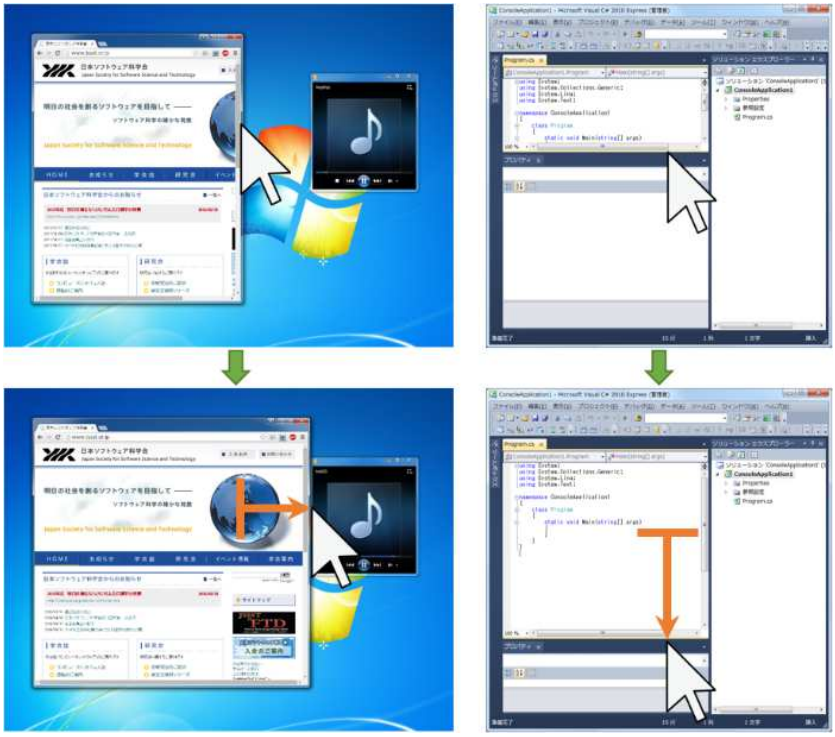


図 5.1　細長いターゲットを DnD する例．（左）ウィンドウのリサイズ，（右）ウィンドウ内のレイアウト変更．

本研究の目的は，図 5.1 の例のように線分と直交する一軸方向に動くターゲットのドラッグ開始操作の支援である．すなわち，図 5.2 のようにターゲットを DnD する方向のサイズ W が小さく，それと垂直なサイズ H が大きいターゲットのドラッグ開始操作を対象とする．

図 5.2　ターゲットの幅 W と長さ H の方向と DnD での移動方向．

従来のポインティングによるドラッグ開始操作を Point-drag と呼ぶことにする．Point-drag による DnD の手順を図 5.3（上）のように分解して考えたとき，ドラッグ開始に着目すると (Step 2)，ターゲットの幅 W が小さいほどカーソルを乗せる操作は困難になる．一方でドロップ位置を決定する操作 (Step 4)の難易度は W に依存せず，ユーザがそのときどきの要求に応じて調整しなければならない．

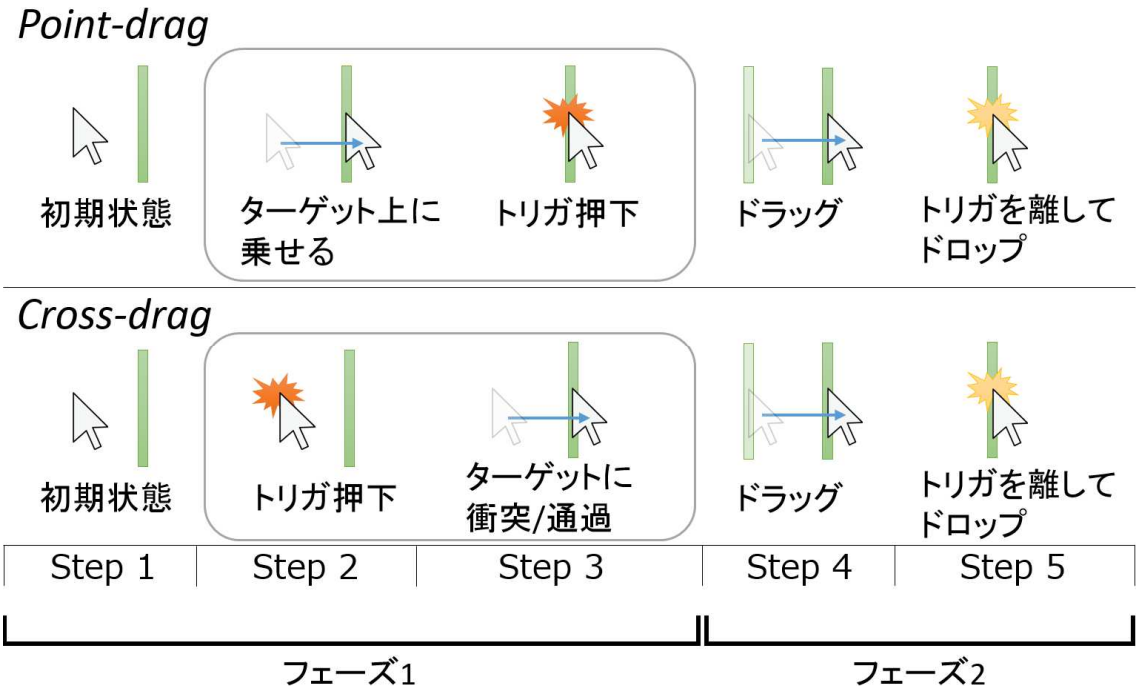


図 5.3　DnD 操作を手順ごとに分解したもの．

小さいターゲットをポインティングする時間を短縮するために様々な手法が提案されており，それらを DnD に適用することも可能である．例として，ターゲット上でカーソル速度を低下させる Sticky Icons [Worden, 1997]や，カーソルを広範囲にするエリアカーソル [Kabbash, 1995]などがあり，時間短縮やエラー率低減の効果を DnD でも発揮することが期待できる．こういったターゲット選択手法の 1 つであるクロッシング [Accot, 2002]は，線分の形状をしたターゲットをカーソルが通過することで選択を行う．ターゲットまでの距離や長さ H によっては，クロッシングはポインティングよりも高速であることが示されている [Accot, 2002]．このことから，クロッシングによって細長いターゲットを選択することで，カーソルをターゲットに乗せる微細な制御が必要なくなり，Point-drag よりもドラッグ開始が容易になると考えた．この操作手法を Cross-drag と呼ぶことにする．

Cross-drag では図 5.3（下）のようにターゲットに衝突することでドラッグを開始する．カーソルが移動速度を維持したままターゲットを選択できるため，操作開始からドラッグ開始まで（図 5.3 の Step 3 まで：フェーズ 1 と呼ぶ）の所要時間のみならず，その後のドロップ終了まで（同じくフェーズ 2 と呼ぶ）の時間を短縮することも可能だと考えた．

本章では Cross-drag の利点・制約を述べたうえで先行研究との比較議論を行い，実験によってタスクパフォーマンスを改善可能なターゲットの条件を求める．

5.2 Cross-drag 手法

5.2.1 操作方法

図 5.3（下）のように，クリック操作をするトリガ（マウスの左ボタンなど）を押している状態で，カーソルがターゲットの領域に侵入または通過した時点でドラッグ開始となり，トリガを離した時点でその場にターゲットを置く．カーソルがターゲットに乗っている状態でトリガを押下しても同様にドラッグ開始となる．従来の Point-drag との差異はトリガを押すタイミングであり，Cross-drag は先にトリガを押してからターゲットに近づくことができる．一方で Point-drag は，ターゲット上にカーソルを乗せてからトリガを押下するため，幅 W が細いと微細な制御が求められる．Cross-drag の利点は，トリガを押したままカーソルをターゲットに“当てる”ようにすればよく，ある程度の長さ H があれば微細な操作が必要ないことである．

ターゲットが一次元方向にのみ移動する DnD 操作では，カーソルがターゲット上から外れてしまってもドラッグを継続するように設計されることが多い．ウィンドウの枠やシークバーなどがよく知られた例である．Cross-drag でも同様に，一度ターゲットを捕捉するとトリガを離すまでドラッグが継続される．

5.2.2 複数のターゲットをドラッグしない設計指針

従来手法では，Ctrl キーや Cmd キーを押したまま何度もマウスボタンを押すことでドラッグするターゲットを追加できる．言い換えると，コマンド操作をしない限りは，最初に選択したターゲットのみをドラッグする設計になっている．Cross-drag でも同様に，単一のターゲットのみをドラッグすることとした．例として図 5.4 では，ファイルアイコンをドラッグしてウィンドウの外に出すときには，ウィンドウの縁を通過しても Cross-drag によるウィンドウリサイズは発生しない．

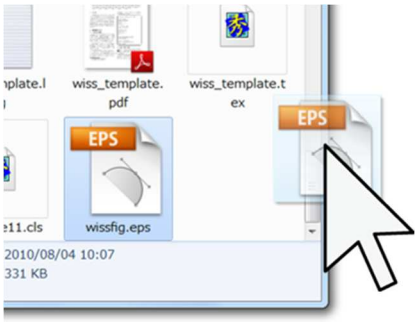


図 5.4　ファイルアイコンのドラッグ中には，ウィンドウの縁を Cross-drag しない．

これとは異なる設計として，衝突した複数個のターゲットを同時にドラッグする方針も考えられる．こうすれば図 5.5 のように，ウィンドウの左枠を広げている最中に上枠にも衝突し，左上方向に拡大してく操作が可能になる．しかし，左方向にだけ拡大したいときには上下の枠に衝突しないように移動させる必要があるため，ウィンドウの縦幅が小さいときにはステアリング操作が必要になり，従来は存在しなかった問題を生むことになってしまう．これを避けるためにも，既にターゲットをドラッグしている最中は Cross-drag が発生しない設計とした．以降，本章では別途のコマンドを併用しない操作について考え，Cross-drag 手法の基本的な性能に焦点を当てて議論する．

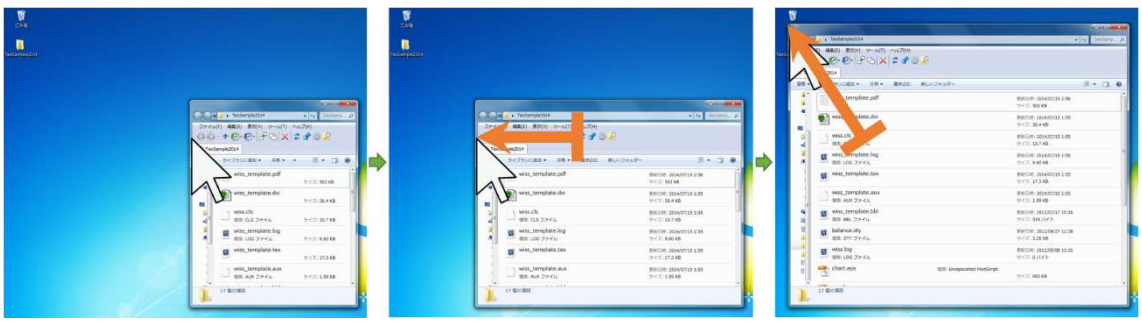


図 5.5　複数ターゲットへの Cross-drag を許す場合の操作例．ウィンドウを左方向へ拡大中に，上の縁にも衝突し，左上方向に拡大する．

5.2.3 システムの実装例

今回実装したウィンドウのリサイズシステム（図 5.6）は，Windows OS 上で起動するウィンドウに対して Cross-drag によるリサイズを可能にするものである．聴覚的・視覚的フィードバックのために，ターゲットへの衝突時に効果音を鳴らし，カーソルを赤色の拳形状に変化させて移動方向を向くようにする．またウィンドウ内のレイアウトを変更するシステム（図 5.7）も実装した．これはユーザがドラッグしたい境界線をシステムに登録することで Cross-drag を利用できる．これらのシステムは HSP3.32 および Win32API を用いて実装した．

図 5.6　ウィンドウのリサイズシステム．

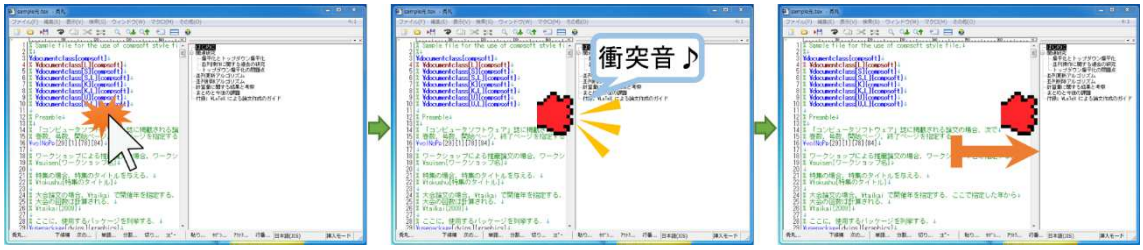


図 5.7　ウィンドウ内のレイアウト変更システム．

5.3　Cross-drag 手法の性能と使用方法に関する議論

5.3.1 ターゲットの配置間隔に関して

クロッシング操作の性質として，ターゲット同士が平行に並んでいるときには目的外のターゲットを避けなければならず，ポインティング操作に近くなるという制約がある．図 5.8 のように 3 枚のウィンドウがカスケード表示されているとき，中央のウィンドウの縁を

Cross-drag したければカーソル制御が非常に細かくなる．過去のクロッシングの実験では，ターゲット間の距離が 256 ピクセル [Accot, 2002] [Forlines, 2008]あるいは 128 ピクセル [Wobbrock, 2007]に設定されているが，実際の GUI 環境ではさらにターゲット同士の間隔が狭い場面がある．こういった状況下では，ターゲットから離れた位置でトリガを押してもよいという Cross-drag の利得が小さくなってしまう．

極端な場合には，3 つ以上のターゲットが完全に密着しているとき，間に挟まれたターゲットを Cross-drag する操作は完全に Point-drag と同じになる．原理的にはターゲット間に僅かでも隙間があれば Point-drag と同等以上の性能になるはずだが，実際の効果は評価実験で確かめる必要がある．そこで本章では，先行研究よりも狭いターゲット間隔を設定することで，実際の GUI 環境に近い条件下での性能を検証する．

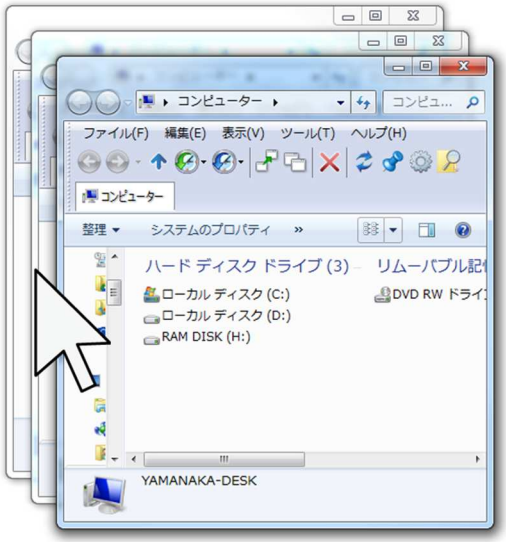


図 5.8　ターゲット同士の間隔が狭いと，カーソルの位置調整が難しくなり，ポインティングに近くなる．

また図 5.5 のウィンドウを拡大していく例において，ドラッグしたい左枠の長さ H が小さい場合には，ウィンドウの上下にある横枠線同士が近接してしまう．すると最初に目的外のターゲットに衝突してしまう危険が高まり，難度が上昇すると考えられる．こういった条件下での Cross-drag の有用性は未知であるため，本稿の実験で独自に検証する必要がある．さらに，ターゲットの幅 W が十分に大きいときには，従来の Point-drag でも問題なく操作できるはずである．すなわち，Cross-drag が有意義な貢献をする W の範囲があると考えられる．よって実験では，平行に並んだターゲット間のギャップ（G とする），長さ H，幅 W を変化させて有用性を検証する．

5.3.2 提案手法を使用するためのトリガ

Cross-drag を適用するシステムによっては操作しづらい状況が生じることがある．例として図 5.9 のように文字が敷き詰めているウィンドウの縁を Cross-drag する場合を考える．「文字列の選択はターゲットのドラッグである」と設定すれば，文字列の選択中に不用意にウィンドウの縁に衝突しても，既にターゲットをドラッグしている最中なので Cross-drag は起こらない．しかしウィンドウの縁を Cross-drag したいときには文字列を選択しないように避ける必要があり，微細な制御が求められる．逆に，「文字列の選択はターゲットのドラッグではない」と設定すると，ウィンドウ内の文字列を選択しているときに不用意に外枠を Cross-drag するミスが起こりうる．つまり図 5.9 の例では，どちらの設定を採用しても慎重な操作が求められるのである．これはトリガを押下できる範囲が狭まることと同じであり，Cross-drag の利得が小さくなる．

解決策の 1 つとして，Cross-drag の専用トリガの利用が考えられる．キーボードの特定のコマンド（Ctrl+Alt など）や多ボタンマウスなどを使用することで，既存の操作と競合せずに Cross-drag モードを発動できるようにする方法である．これは使用環境が限定されるデメリットがあるものの，普段から多ボタンマウスなどを使用しているユーザにとっては快適な操作方法となるはずである．本章の実験では，システムや操作環境を限定せず，ベースとなる手法の有用性検証を主眼とするため，Cross-drag 専用のトリガを設定しない条件で評価を行う．すなわち，クリック操作などと同一のボタンで Cross-drag を発動することにする．一連の作業中にトリガを使い分ける条件での有用性については今後の検証課題としたい．

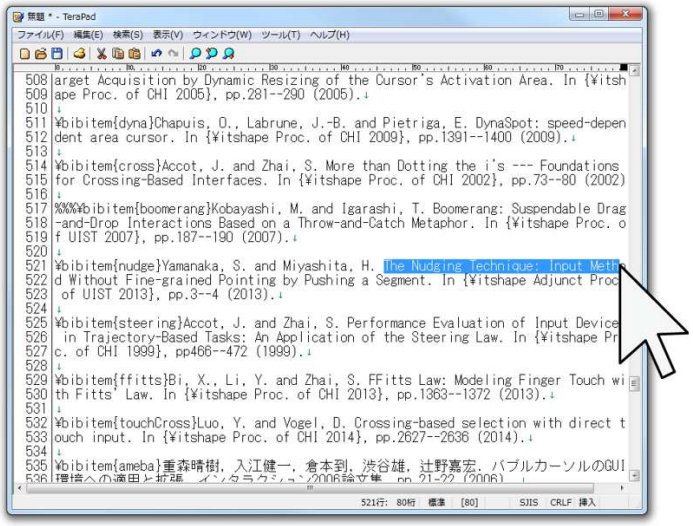


図 5.9　文字が敷き詰められたウィンドウを Cross-drag する操作．文字列を選択したいのか，ウィンドウを Cross-drag したいのかをシステム側から判別するのが難しい．

5.4　関連研究

ポインティング支援手法をドラッグ開始に利用することで DnD 時間を短縮することも可能であると考えられる．そこで，本研究の議論対象である細長いターゲットの DnD について，Cross-drag とその他のポインティング支援手法との比較議論を行う．なお，ターゲット選択を支援する手法については 2.2 節で既に紹介しているため，ここでは細長いターゲットの選択に特化した議論をする．

5.4.1　*W* を拡大する手法との比較

フェーズ 1 のドラッグ開始時において，エリアカーソルと Cross-drag の利得を比較する．ターゲットの拡大手法やスナッピングで得られる利得もカーソル拡大と同様であると解釈できるため，ここでは代表してカーソルから最も近いターゲットを捕捉する手法の Bubble Cursor [Grossman, 2005]と比較する．Bubble Cursor では，図 5.10 においてカーソルの中心が p_1〜p_4 の間にあるときにはターゲット B，p_4〜p_7 の間では C が捕捉される．一方 Cross-drag では，B をドラッグしたいときには p_0〜p_5 の範囲でトリガを押してからターゲットに衝突する．同様に C をドラッグしたければ p_3〜p_8 の間でトリガを押せばよい．Cross-drag ではターゲット B と C のトリガ押下可能範囲が p_3〜p_5 で共有されている．したがって，Cross-drag の方が微細な制御を要さず，この観点では利得が大きいといえる．

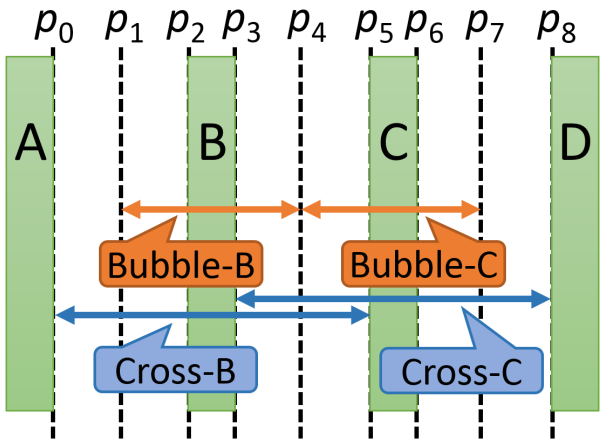


図 5.10　Bubble Cursor と Cross-drag で，ターゲット B と C をドラッグしたいときにトリガ押下が可能なカーソル範囲．「（手法）ー（捕捉されるターゲット）」の組み合わせで記載している．「Bubble-B」は，Bubble Cursor でターゲット B を選択できる範囲を示す．p_0〜p_8 は横軸方向の位置を示す．

5.4.2 移動方向を考慮したターゲット選択

移動方向を考慮するエリアカーソルに，Fan Cursor [Su, 2014]（図 5.11）やアメーバ状カーソル [Shigemori, 2006]がある．これらはカーソルの移動方向の先にあるターゲットを優先的に選択する機能をもち，遠くのターゲットをより素早く選択できる．Fan Cursor では，ターゲットをドロップしたい位置が図 5.12 のようにカーソルを引き返す方向にある場合に Cross-drag よりも移動距離を短縮できる．逆に，カーソル→ターゲット→ドロップ位置，という順の配置では，結局カーソルをドロップ位置まで移動させる必要があり，Cross-drag と移動距離は変わらない．このことから，フェーズ 1 におけるターゲットの選択手法として有用なものであっても，フェーズ 1〜2 を通した一連の DnD 操作に適用するとメリットが限定される状況があることがわかる．

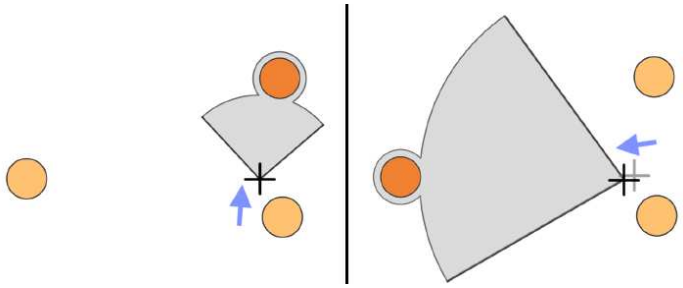


図 5.11　Fan Cursor の概要．カーソルの前方に捕捉エリアを広げることで，「距離は近いが選択したくないターゲット」を無視できる．

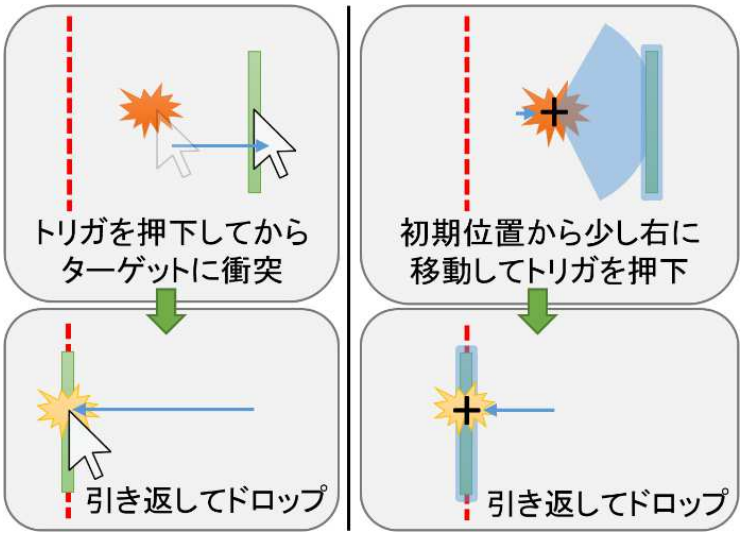


図 5.12　Cross-drag（左）と Fan Cursor（右）で，初期移動から引き返した方向にドロップする操作の比較．赤色の破線はドロップしたい目標位置．Fan Cursor の方が移動距離が短縮できることがわかる．

ここまでの議論は，ウィンドウの枠のように長さ H が十分に大きく，図 5.10 のようにターゲットが 1 次元方向に並んでいることを想定していた．一方で図 5.13 のように，上下のターゲットが近接している状況下では，移動方向を考慮したエリアカーソルの利得が小さくなる問題が生じる．たとえば Fan Cursor やアメーバ状カーソルでは，左右のターゲットにかなり接近しないと捕捉できず，それまでは上下のターゲットを捕捉しているためである．同様に Cross-drag も，上下のターゲットを避けるためにターゲット付近でトリガを押すようになり，利得が小さくなるおそれがある．Fan Cursor と Cross-drag のいずれも，2 次元方向に妨害刺激（選択したくないターゲット）が近接している条件では有用性が低減するといえる．

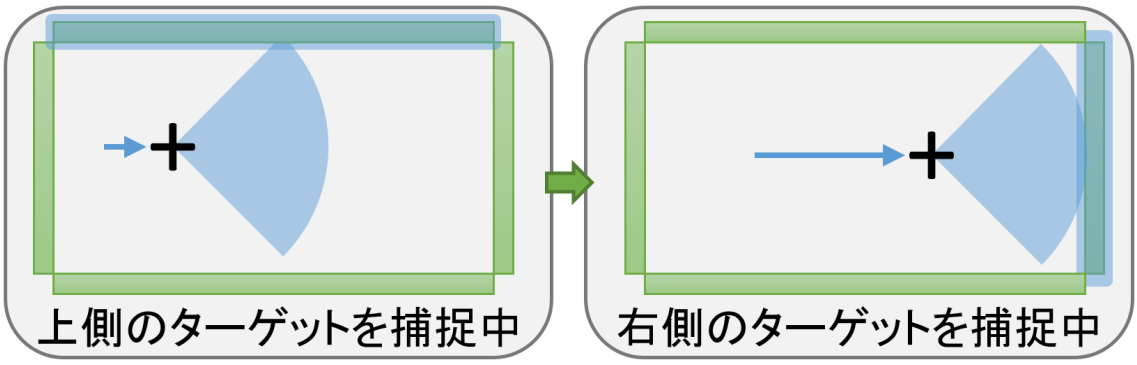


図 5.13　上下左右にターゲットが近接した状況．Fan Cursor は速度に応じて扇型を 90°～180°に広げるが，ここでは最も前方まで選択できる 90°の例を示す．

5.5　実験

Cross-drag が有効に機能する条件を求めるための実験を行う．細長いターゲットを DnD するタスクにおいて，Cross-drag と Point-drag の操作時間およびエラー率を比較する．以下に詳細を述べる．

5.5.1　タスク

図 5.14 のように，タスクが開始されると，灰色の画面内にスタート領域，ターゲット，障害物（ダミーターゲット），ドロップ領域が表示され，カーソルがスタート領域の中心に置かれる．実験参加者はスタート領域をクリックした後，ターゲットをドロップ領域に DnD する．スタートからゴールへの方向は上下左右の 4 パターン存在する．ターゲットはスタートとゴールを結ぶ 1 次元方向にのみ移動するが，カーソルは 2 次元方向に移動できる．図

5.14 において，ターゲットと障害物の間隔 *G* のみは（中心同士の距離ではなく）ターゲットとダミーに挟まれた空白領域のサイズである．操作時間およびエラーの計測は，スタート領域をクリックしてから，ターゲット全体がドロップ領域に含まれるように移動してトリガを離すまでとする．

ターゲットの前後にダミーを設置したのは，ターゲットの間隔が狭まるとポインティングに近くなるという議論に基づく．またターゲットの側面にダミーを設置したのは，妨害刺激が側面にあるとターゲットに近づいてからトリガを押すようになるという議論に基づく．タスクのパラメータで説明すると，移動方向の前後にあるダミーはギャップ *G*，側面のダミーはターゲットの長さ *H* に依存して Cross-drag の性能に影響を与える妨害刺激である．Cross-drag で目的のターゲットよりも先にダミーに接触した場合は，意図しないドラッグ開始であると見なしてエラーとする．すなわち，実験参加者は図 5.14 の緑色のダミーで囲まれた領域でトリガを押し，ダミーを避けつつ赤いターゲットに衝突する必要がある．

ターゲットを正しくドロップするとベル音が鳴って全画面が 1 秒間灰色で描画され，次のスタート領域・ターゲット・ダミー・ドロップ領域の配置（以降，画面配置と呼ぶ）が決定し，カーソルがスタート領域の中心に移動する．画面配置の選出順はランダムである．エラーが発生すると，その時点でビープ音を鳴らすとともに全画面が 1 秒間灰色で描画され，試行が中断される．その後に同一の画面配置でスタート領域のクリックからタスクをやり直す．

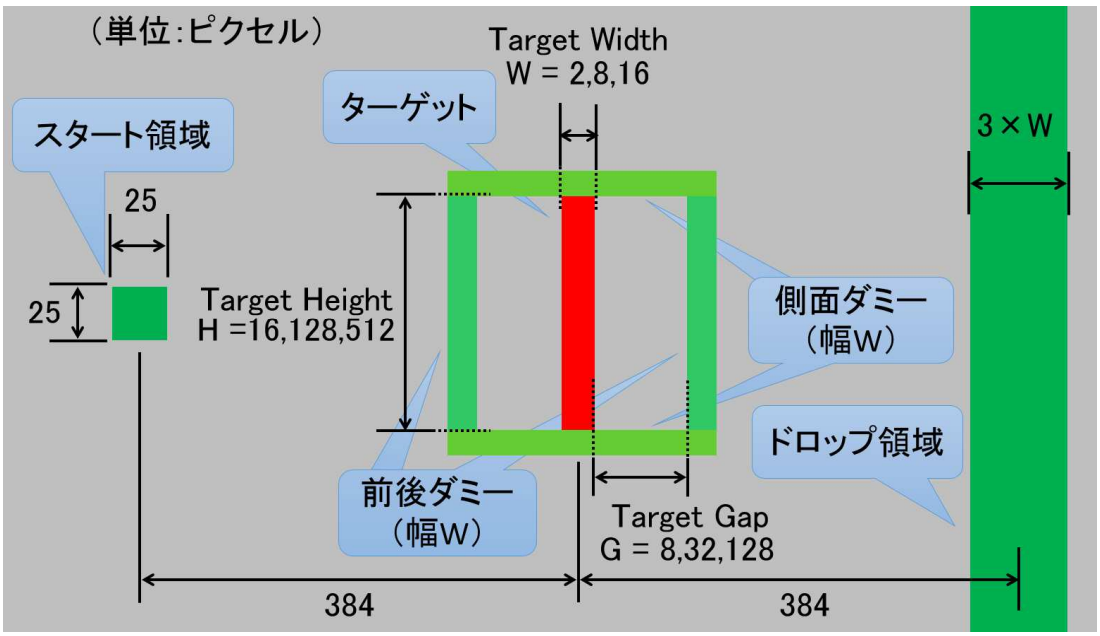


図 5.14　タスクの画面配置模式図．スタート位置が左の場合．

5.5.2 エラー

以下の操作をエラーに計上する.

- トリガを押した状態でダミーに衝突 (Cross-drag)
- ターゲット・ダミーのいずれにも衝突せずにトリガを離す (Cross-drag)
- ターゲットの領域外でトリガを押す (Point-drag)
- ターゲット全体がドロップ領域に入っていない状態でトリガを離す（両手法）

先行研究 [Wobbrock, 2007]におけるクロッシングのエラーは, トリガを押したまま線分の延長線上を通過した場合と定義されている（図 5.15）. しかし図 5.13 の議論にあるように, 本実験ではターゲット同士が縦横に近接した環境での有用性も検証したい. よって先行研究よりもエラーの解釈を拡大している.

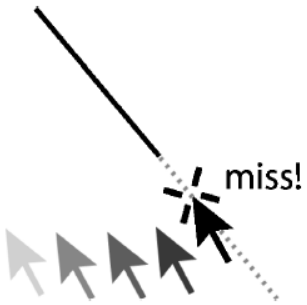


図 5.15　先行研究 [Wobbrock, 2007]におけるクロッシングのエラーの定義.

5.5.3 実験パラメータ

5.5.3.1 平行なターゲット同士のギャップ *G*

ギャップ *G* は 8, 32, 128 ピクセルである. これらの値は, Windows 8.1, Mac OS X Yosemite 10.10.2, Ubuntu14.04.1, Debian 7.8.0 においてカスケード表示されたウィンドウのギャップを参考に設定し, ターゲット同士が近接している状況から離れている状況までを含む範囲になっている.

5.5.3.2 ターゲットの幅 *W*

ターゲットの幅 *W* は 2, 8, 16 ピクセルとした. まず,「Point-drag にとってはドラッグ開

始が困難なほど細いターゲットであっても，Cross-drag であれば素早くできる」という仮説を検証するために，下限値を 2 ピクセルとした．また各種 OS のウィンドウ枠のドラッグ領域サイズを参考に 8, 16 ピクセルを採用した．

5.5.3.3 ターゲットの長さ *H*

ターゲットの長さ *H* は 16, 128, 512 ピクセルとした. まずターゲットの長さ *H* が小さい条件として 16，128 ピクセルを採用した．また，「ターゲットが十分に長ければ微細な制御を要さない」ことを検証するために，最大値を 512 ピクセルとした．これは，「フル HD の画面で，中程度のサイズのウィンドウ（960×540 ピクセル）をリサイズする」操作を想定して設定した．

5.5.4 実験機器等

マウスは Buffalo BSMBU05（光学式，1000 dpi，USB 有線接続）を用いる．ディスプレイは 21.5 インチ，1920×1080 ピクセルの LCD を用い，Windows 7 で実験システムを全画面表示する. カーソルは長さ 25 ピクセル, 幅 1 ピクセルの線分を交差させた黒色十字を用いる．カーソルの速度は OS の標準（コントロールパネルの中央）に設定し，ソフトウェア加速をオフにする．

5.5.5 実験参加者

実験参加者は情報系の大学生及び大学院生の 10 名（男性 9 名，女性 1 名，平均 22.9 歳）であり，全員がマウス操作に習熟している．9 名が右利き，1 名が左利きであった．普段の PC 作業時にマウスを利用しているのは 9 名であり，全参加者が利き手で操作した．左利きの参加者 1 名は普段からマウスの左ボタンでクリック操作をしていたため，使用するトリガは全員とも左ボタンである．

5.5.6 実験手順

Cross-drag の画面配置は合計で 3(*W*)×3(*H*)×3(*G*) ×4（スタート位置）=108 パターンである．Point-drag ではダミーの有無がタスクの難易度に影響しないが，視覚刺激量を同程度に

するために *G* を中距離の 32 ピクセルに固定して描画する．したがって Point-drag の画面配置は 3(*W*)×3(*H*)×1(*G*) ×4（スタート位置）=36 パターンとなる．各手法での本番前に 18 試行ずつの練習を設け，タスク内容の理解および使用機器への慣れを促した．記録されるデータは（Point-drag 36 パターン + Cross-drag 108 パターン）×10 名=1440 回分となる．

実験参加者にはタスク内容やエラーとなる操作を教示したうえで，可能な限り素早くかつエラーをせずに操作するよう伝えた．実験参加者の半数が Cross-drag を先に使用し，残りの半数が Point-drag を先に使用する．練習と本番の間，および Cross-drag 本番の 36 回ごとに 30 秒間の休憩を設ける．全試行の後，主観評価を行うためのアンケートを実施した．実験に要する時間は，事前のインストラクションから全試行終了まで 25 分程度である．練習時に椅子の高さやマウスパッドの位置，ディスプレイの位置・角度を調節させ，各実験参加者が実験に集中できるよう配慮した．

5.5.7 その他の実験条件の検討

タスクパフォーマンスに影響を与えるその他の要因として，スタート・ターゲット・ゴールの位置関係やサイズの比率，ポインティングデバイスの dpi，カーソル速度やソフトウェア加速の有無など様々なものが考えられる．これらの条件によって結果が変わる可能性は残されているが，本実験では特に重視したいターゲットの幅 *W*，長さ *H*，ギャップ *G* の 3 条件に絞って観察する．

また，本実験ではシステム実装例に含まれるような視覚的・聴覚的支援をするためのフィードバックを排除した実験デザインを採用する．すなわち，カーソルは拳型ではなく常に十字型であり，Cross-drag によってドラッグを開始したときの効果音は鳴らない．操作手法の設計にフィードバック方法を含める考え方もあるが，実装例に含まれるフィードバックが Cross-drag にとって望ましいか否かは新たな議論が必要になる．本実験ではシステム例のベースとなる操作方法の有用性を検証したいと考えているため，フィードバック方法の良し悪しと操作手法の性能を切り分けて検証するべく，フィードバックは行わないこととした．同様に，Point-drag で一般的に行われるフィードバック（ターゲット上の乗ったカーソルの形状変更など）についても搭載せず，Cross-drag と同一条件にする．

5.6 実験結果

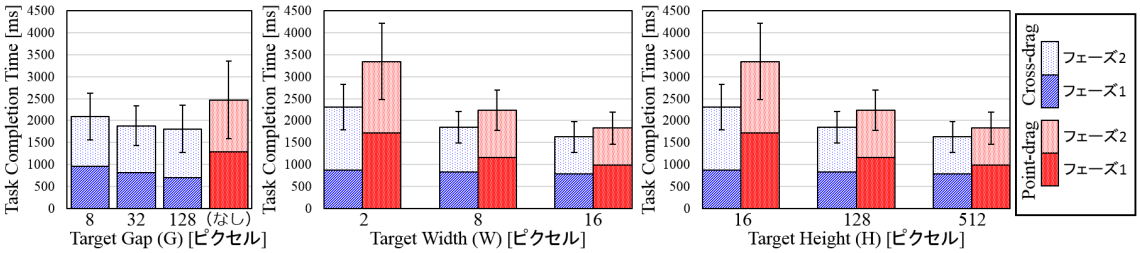
実験全体で 1527 回の試行があり，その内 87 回 (5.70%)がエラーであった．エラーの無かった試行 1440 回について，ターゲットの幅 *W*，長さ *H*，手法を要因とした対応あり三元配置分散分析（反復測定）を行った．Point-drag は *G* の要因を含まないため，手法の分析では

Cross-drag は G の値ごとに異なる手法として分析する. すなわち, Cross-drag 手法は Cross-8, Cross-32, Cross-128 の 3 種類として扱う（数値は G の値）. また, 以下では Point-drag は Point と表記する. 多重比較には Bonferroni の手法を用いた.

5.6.1 操作時間

図 5.16 にパラメータとフェーズごとの操作時間を示す. 以下では, タスク完了時間（フェーズ 1 と 2 の合計時間）, フェーズ 1 の所要時間, フェーズ 2 の所要時間に分けて分析する.

5.6.1.1 タスク完了時間



(a) ダミーターゲットのギャップG　(b) ターゲットの幅W　(c) ターゲットの長さH

図 5.16　実験パラメータごとの平均タスク完了時間.

分析の結果, W ($F_{2, 18} = 774.84$, $p < .001$), H ($F_{2, 18} = 56.36$, $p < .001$), 手法 ($F_{3, 27} = 69.04$, $p < .001$)による主効果が見られた. 多重比較では, 全ての W の間に $p < .001$, $H = 16$ と 128 の間に $p < .001$, $H = 16$ と 512 の間に $p < .001$ の有意差が見られ, それぞれ値が大きいほどタスク完了時間が短縮された. $H = 128$ と 512 の間には有意差が見られなかった. 手法に関して, Cross-32 と Cross-128 の間以外で有意差が見られた. Point と Cross-8 の間に $p < .01$ の有意差が見られた以外は, 全て $p < .001$ の有意差が見られた. 分析の結果, G の値によらず Cross-drag は Point-drag よりも高速であり, Cross-drag の中では G が大きいほど高速であることがわかった.

手法×H の間に交互作用が見られ($F_{6, 54} = 2.73$, $p < .05$), Cross-8 ではいずれの H の間にも有意差がなかった. Cross-32 では $H = 16$ と 128 の間($p < .001$), H=16 と 128 の間($p < .001$), $H = 16$ と 512 の間($p < .01$)に有意差が見られた. Cross-128 では, $H = 16$ と 128 の間および $H = 16$ と 512 の間に $p < .001$ の有意差が見られた. Point は, $H = 16$ と 128 の間および $H = 16$ と 512 の間に $p < .001$ の有意差が見られた. また, いずれの H においても Cross-drag の方が高速であった($p < .001$). 以上のいずれも G と H が大きいほど時間が短縮された.

また手法×W の間に交互作用が見られた($F_{6, 54} = 29.10$, $p < .001$). 手法間に有意差が見られ

なかったのは，*W* = 8，16 のときに Cross-8 を使用した場合である．つまり幅 *W* が大きくギャップ *G* が狭いときには，Cross-drag は操作時間を短縮しないことがわかった．

Cross-drag の中で比較すると，有意差が見られなかったは *W* = 128 のときの Cross-32 と Cross-128 の間だけであった．つまりギャップ *G* が大きいほど時間は短縮されるが，幅 W が十分に大きい時には *G* ≧32 であれば十分高速化できていることがわかった．これら以外の組み合わせでは少なくとも $p < .05$ の有意差が見られ，*G* と *W* が大きいほど時間が短縮されていた．

5.6.1.2 フェーズ 1 の所要時間

フェーズ 1 の所要時間に関して同様に分析したところ，*W* ($F_{2, 18} = 86.87$, $p < .001$)，*H* ($F_{2, 18} = 91.22$, $p < .001$)，手法 ($F_{3, 27} = 101.09$, $p < .001$)による主効果が見られた．多重比較では，全ての *W* の間に $p < .001$，*H* = 16 と 128 の間および *H* = 16 と 512 の間に $p < .001$ の有意差が見られ，それぞれ値が大きいほど短時間になった．*H* = 128 と 512 の間には有意差が見られなかった．また Point と Cross-8 の間，および Cross-8 と Cross-32 の間に $p < .01$，その他の全ての手法の間に $p < .01$ の有意差が見られた．*G* の値によらず Cross-drag は Point-drag よりも高速であり，Cross-drag は *G* が大きいほど高速であることがわかった．

手法×*H* の間に交互作用が見られた ($F_{6, 54} = 3.50$, $p < .01$)．*H* = 32 と *H* = 128 のときには，全ての手法の間に少なくとも $p < .01$ の有意差が見られ，*G* と *H* の値が大きいほど高速であった．*H* = 16 のときには，Point と Cross-8 の間に $p < .01$，Point と Cross-32，Cross-128 の間に $p < .001$ の有意差が見られた．また *H* = 16 のときの Cross-drag は，Cross-8 と Cross-128 の間に $p < .01$ の有意差が見られたが，Cross-32 と Cross-128 の間には有意差がなかった．すなわち，Cross-drag はいずれの *G* の値においても Point-drag より高速であり，*G* が大きくなるほど時間が短縮されることがわかった．

また手法×*W* の間に交互作用が見られた ($F_{6, 54} = 40.23$, $p < .001$). Cross-8 のときには，*W* = 2 と 8 の間 ($p < .01$)，*W* = 8 と 16 の間 ($p < .01$)，*W* = 2 と 16 の間 ($p < .001$)の有意差が見られた．また Cross-32 のときには，*W* = 2 と 16 の間 ($p < .05$)，*W* = 8 と 16 の間 ($p < .05$)に有意差が見られた．また Point では，全ての *W* の間に $p < .001$ の有意差が見られ，*W* が大きいほど時間が短縮された．Cross-drag の視点では，*W* = 16，Cross-8 のときに Point に対して有意差が見られなかったのを除き，全ての *W* と *G* の組み合わせにおいて少なくとも $p < .05$ で Point-drag よりも高速であった．

5.6.1.3 フェーズ 2 の所要時間

フェーズ 2 の所要時間に関して同様に分析したところ, *W* ($F_{2, 18}$ = 88.12, p < .001), 手法 ($F_{3, 27}$ = 5.35, p < .01)による主効果が見られた．*H* による主効果は見られなかった．多重比較では，全ての *W* の間に p < .001 の有意差が見られ，値が大きいほど時間が短縮された．手法については Cross-8 と Cross-32 の間に p < .05 の有意差が見られた．*G* の値に関わらず Cross-drag は Point-drag より高速であったが，有意差は見られなかった．

交互作用が見られたのは手法×*W* の間のみであった($F_{6, 54}$ = 3.16, p < .05)．*W* = 8 のとき，いずれの手法の間にも有意差は見られなかった．また *W* = 16 のときに有意差が見られたのは Cross-8 と Cross-32 の間 (p < .05)のみであった．したがって，*W* が十分大きければ Cross-drag による時間短縮は起こらないことが分かった．最後に *W* = 2 のときに有意差が見られたのは Point と Cross-32 の間 (p < .05)のみであった．よってフェーズ 2 では，Cross-drag が時間を短縮するのは限られた条件下であることがわかる．

5.6.2 エラー率

Cross-drag と Point-drag ではエラーを認める操作とフェーズが異なるため，各手法の合計のエラー率のみを比較する．Cross-drag の 5.26%（1140 回中の 60 回），Point-drag の 6.97%（387 回中の 27 回）のエラー率に関して，操作時間と同様の分散分析では *W* のみに主効果が認められた ($F_{2, 18}$ = 11.26, p < .01)．多重比較では *W* = 2 と 8 の間 (p < .05)，*W* = 2 と 16 の間(p < .05)の間に有意差が見られた．

手法×*H* のみ交互作用が見られ ($F_{6, 54}$ = 3.06, p < .05)，Cross-8 のときに *H* = 128 と 512 の間 (p < .05)に有意差が見られた．Point では *H* による有意差は見られなかった．また，いずれの *H* に関しても Cross-drag と Point-drag の間には有意差は見られなかった．

5.6.3 主観評価

事後アンケートでは，各手法でのターゲット捕捉の難易度について 1:容易～7:困難の 7 段階で尋ねた．その結果，Point-drag が平均 4.8 (*SD* = 0.74)，Cross-drag が平均 2.6 (SD = 0.80) となり，10 名全員が Cross-drag の方が容易であると回答した．

自由記述によって Cross-drag 操作の所感を尋ねたところ，「ギャップ *G* が十分に大きい場合は，多少の減速程度でスムーズにドラッグに移れた」，「*G* が狭い場合は Point-drag とほぼ同じ感覚で操作した．空白部分でボタンを押しても問題ない分，少しだけ楽に感じた」と

いった意見が得られ，Cross-drag の支援によって操作が容易になったと感じられたとの意見が見られた．一方で，「*G* が小さいとき，Point-drag のように[ターゲット上でトリガを押す]操作ができても，あえてギャップの部分でボタンを押す操作をしてしまった」，「ターゲットから離れた位置からボタンを押し始めるとミスを誘発するので, *G* によらずターゲットの近くからボタンを押し始めるよう心がけた」といった指摘があり，Cross-drag の利点を活かす使われ方をしていなかった事例も確認された．

5.7 考察

5.7.1 ターゲット間のギャップ *G*

Cross-drag のギャップ *G* に関して，タスク完了時間は Cross-8 と Cross-32 の間，および Cross-8 と Cross-128 の間では有意差があったが($p < .001$)，Cross-32 と Cross-128 の間には差がなく, *G* は 32 ピクセル以上あれば十分だと考えられる．また *G* の値に関わらず Cross-drag は Point-drag よりタスク完了時間が短く，ターゲット間に僅かでもギャップがあれば Cross-drag が有効であることが確認された．

フェーズ 1 では *G* が大きいほど所要時間が短縮され, Cross-drag のドラッグ開始操作は *G* が大きいほど容易に行えることが確認された．一方でフェーズ 2 では，Cross-8 と Cross-32 の間にのみ有意差が見られた．よって「高速なままドラッグを開始できるため，フェーズ 2 の所要時間は *G* が大きいほど短縮される」という性質は限定的なものであった．この理由として，高速なドラッグ開始で得られる短縮時間に対して，フェーズ 2 の終盤におけるドロップ操作に要する時間の割合が大きく，Cross-drag による利得の影響が弱まった可能性が考えられる．

5.7.2 ターゲットの幅 *W*

フェーズ 1 では全ての手法×*W* の組み合わせにおいて Cross-drag の方が高速であり, フェーズ 2 では Cross-8，$W = 16$ 以外の組み合わせにおいて Cross-drag の方が高速であった．また手法間でタスク完了時間に有意差が無かったのは Cross-8 で $W = 8$, および Cross-8 で $W = 16$ の場合であった．これは *W* に関して Point-drag が好条件，かつ *G* に関して Cross-drag が最も悪条件の場合であり，Cross-drag の方が短時間であったものの有意差は見られなかった．したがって，一般的な GUI における細長いターゲットの幅 *W*，ギャップ *G* においては，いずれの *W*，*G* の組み合わせにおいても Cross-drag は Point-drag と同等以上の性能を発揮でき

ることが確認された.

5.7.3 ターゲットの長さ *H*

ターゲットの長さ *H* に関して，両手法ともターゲットの長さ *H* が 128 ピクセルあれば，それ以上の場合と遜色なく操作できるといえる．特に Cross-drag は *H* が大きいほど操作が容易になるが, その場合は 128 ピクセルで十分であることが分かった. ターゲットが短い (*H* = 16)場合には慎重さを要するが，いずれの *H* に対しても Point-drag より有意に ($p < .001$)高速化されており，*H* によらずドラッグ開始操作が支援されていることが確認された．フェーズ 2 の所要時間は *H* による差が見られなかったが，これはドロップ操作が *H* に依存しないタスクであるため予想通りの結果であった．したがって Cross-drag のタスク完了時間が H によらず Point-drag より短縮されていたのは，フェーズ 1 のドラッグ開始操作が支援されていたことに起因するといえる．

5.7.4 ドロップ領域の幅

今回の実験ではドロップ領域の幅を 6，24，48 ピクセルに設定しており，フェーズ 2 の操作もある程度慎重に行う必要があったと考えられる．これに対して，ドロップ位置を細かく指定しないようなタスク，たとえば「コンテンツ全体が見られるようにウィンドウを拡大できればよい」などという要求に対しては，ターゲットを一定距離以上ドラッグした後はどこでドロップしてもよいことになる．つまりドロップ操作は，第 3 章で実験した「ターゲットサイズが無限大のポインティングタスク」になり，ドラッグ開始からドロップまで全く減速しないことも可能である．よってフェーズ 2 の所要時間が今回の実験よりも短縮されると考えられる．一方で Point-drag はドラッグ開始後から加速し始める必要があるため，本実験結果よりもフェーズ 2 の所要時間差が大きくなる可能性がある．

ドロップ位置が限定されないタスクでは，Cross-drag の性能を一般化して議論するのが困難であるため，本実験ではドロップ位置の調整にもある程度慎重な操作を要するタスクを設定した．実験目的はあくまでドラッグ開始時のパラメータによる影響の検証であるため，ドロップ領域の幅は「慎重さを要しつつも，ターゲット幅 *W* と同じにするほど微細な操作を求めなくてよい」ものにするため $3 \times W$ とした．ドロップ領域の幅によって，特にフェーズ 2 の所要時間が変わることが予想されるため，今後の検証課題としてターゲット以外のパラメータを変更しての有用性評価が残されている．

5.7.5 エラー率

エラー率は手法間に有意差が見られず，Cross-drag によってエラー率を改善することはないことが分かった．一般的なポインティングタスクではエラー率が 4%程度 [Wobbrock, 2008-b]とされているが，DnD タスクでは 2 回の選択操作があるのに加え，何もないところでトリガを離してしまうエラーも生じるため，マウス操作ではエラー率が 10%以上になる [MacKenzie, 1991]との報告がある. これに対し本実験の Point-drag のエラー率は低めの 6.97%であった．この原因として，文献 [MacKenzie, 1991]ではターゲットの幅 W が 8〜64 ピクセルであったのに対し, 本実験では W が 2〜16 ピクセルと全体的に小さい条件だったために，実験参加者が速度よりも精度を優先していた可能性が考えられる.

またマウスによるクロッシング操作は 3%程度のエラー率になる [Wobbrock, 2007]とされているが，本実験の Cross-drag のエラー率は 5.26%であった．Cross-drag はターゲットの捕捉がクロッシング，ターゲットのドロップ位置の決定はポインティングで行うため，文献 [Wobbrock, 2007]の 3%と文献 [Accot, 2003]の 4%を加算した 7%程度のエラー率だと予想されたが，こちらも Point-drag と同様の理由でやや低めになったと考えられる.

Cross-drag ではカーソルが高速なままターゲットに衝突することを許すため，実験参加者の戦略によってはダミーへの衝突が多く発生し，Point-drag よりもエラー率が高くなることも予想された．しかし G と H の値に関わらず Point-drag より悪化することはなかった．また Point-drag にとって最も好条件の $W = 16$ の場合においても Cross-drag のエラー率は悪化しないことが確認された.

5.7.6 Cross-drag が効果を発揮する条件

実験結果をまとめると，Cross-drag の性能はターゲットの条件ごとに次のようになる.

- G: ターゲット同士が密着している場合には Point-drag と同一の操作になるが，わずか（8 ピクセル）でもターゲット間にギャップがあれば Point-drag より操作時間が改善される．G が 32 ピクセルあれば，それ以上ギャップがあるときと操作時間に差はない.
- W: 幅に関わらず Point-drag と同等以上に高速である．W が大きく（8，16 ピクセル），かつギャップ G が小さい（8 ピクセル）場合には Point-drag と操作時間に差がない.
- H: 長さに関わらず Point-drag より操作時間が改善される．H が 16 ピクセルのときより 128 ピクセルの方が操作時間が短縮され，128 ピクセル以上の長さでは差はない.

以上の結果より，Cross-drag は Point-drag よりもエラー率を悪化させることなく，多くの条件下で操作時間のみを改善する手法であるといえる．先行研究で提案されている手法では，条件によって従来手法より性能が悪化する例もあり，間接的な比較ながらこの点にお

いても Cross-drag の有用性が認められると考える．たとえばターゲットが密集した環境で Bubble Cursor [Grossman, 2005]を使うと，一点を指すカーソルよりも操作時間および主観的な操作性が悪化することが分かっている [Shigemori, 2007]．またターゲットを拡大する手法 [McGuffin, 2002] [Zhai, 2003]は，ターゲット間の空白を利用した DnD タスクで操作時間が増大すると報告されている [Tsukitani, 2011]．既存の GUI に新たな操作手法を導入するとき，従来手法よりパフォーマンスを悪化させないのが望ましいと考える． Cross-drag はこの要件をクリアしつつ，Point-drag より操作時間を短縮することが多かったため，ドラッグ開始操作の支援に有用であると考える．

5.7.7 制約

今回はマウスを利用したが，たとえば指での直接タッチでは $G = 8$ だと Cross-drag するのに細すぎるため操作時間が改善されない可能性もある．逆に，他のデバイスでは Point-drag がさらに困難になり，Cross-drag を使うことで改善度合いが大きくなることも考えられる．また指やペンでの直接タッチでは専用トリガが必須であるため，キーボードなど他のデバイスを使用するか, 画面上のボタン操作などで明示的に Cross-drag モードに切り替える必要がある. こういった複数デバイスの連携やモード切り替えの手間を含めた Cross-drag の性能は今回の実験結果からは議論できず，将来の検討課題として残されている．

5.7.8 フィードバック方法

システムの実装例では，視覚的フィードバックとして拳型のカーソルを表示し，またドラッグ開始時に音を鳴らして聴覚的フィードバックを加えていた．本実験ではこれらのフィードバックをせずに性能を評価したが，今後は適切なフィードバック方法を検討したいと考えている．たとえば聴覚的フィードバックについて，当初はドラッグ中に効果音を継続して鳴らし，モードの違いを提示する実装案があった．しかし，これは煩わしく感じられるであろうと考え，衝突時に一瞬だけ鳴るように変更した経緯がある．効果音を加えるタイミングや音の種類によっても Cross-drag の快適さが変わってくると考えられるため, より良いデザインを検証していきたい.

視覚的フィードバックとして実装した拳型カーソルは，線分のターゲットを押し込んでいくイメージを伝えるために採用したデザインである．これに対して Adobe Reader などでは画面を掴んでスクロールするために握り拳型のカーソルが採用されており，こちらに慣れ親しんでいるユーザも多いと思われる．拳型以外にも，親指と人差し指で細いターゲットをつまんでいることを表現したり，ドラッグされているターゲット側に視覚効果を加え

ることも可能である．本実験では操作手法の性能評価に焦点を絞ったが，フィードバックを加えた場合の性能や，操作感がどのように変化するかを定性的に評価することで，適切なフィードバック方法を検討する必要があると考える．

5.8 おわりに

本章では，細長いターゲットのドラッグ開始操作を支援する手法 Cross-drag を提案し，実際の GUI 環境で Cross-drag を利用するためのシステムを複数実装した．Cross-drag を利用すると，ターゲットにカーソルを乗せるための微細な制御が必要なくなることから，DnD タスクの所要時間を短縮可能であるとの仮説を立てた．また，ターゲットの配置間隔やトリガの必要性の議論を通して，提案手法の利点と制約をまとめた．先行研究との比較議論では，ポインティング支援手法を DnD 操作に適用した場合の利点や，移動距離の利得が得られないターゲット条件などを考察した．評価実験では，ターゲット間の間隔 G，ターゲットの幅 W，ターゲットの長さ H を変更した DnD タスクを行い，Cross-drag が効果的に機能する条件をまとめた．

Cross-drag を利用すれば，ターゲットがある程度密集した条件下でも高速に DnD できることが示され，タスク全体の所要時間が短縮可能であるという仮説が支持された．一方で，ターゲットをドロップする時間を短縮する効果はきわめて限定的であったことから，Cross-drag は特にドラッグ開始操作を支援する手法であるといえる．今後の課題として，細長いターゲットをタッチ操作でドラッグする手法との比較実験と，適切なフィードバック方法の検討を考えている．

第6章　視線情報による長距離移動の支援

6.1　はじめに

視線のみで GUI を操作するシステムが従来から研究されており，画面上のターゲットを見続けることで選択したり [Biswas, 2011]，同様にしてテキストを入力するシステム [MacKenzie, 2011]などが提案され，入力速度や精度が検証されている．また，ポインティングデバイスと視線情報を併用することで，従来よりも操作効率を向上させるシステムも提案されている [Zhai, 1999] [Yamato, 2001]．視線は手や腕よりも高速に移動させられるため，GUI の操作効率向上のために活用される動きが高まっている．

視線情報のみでポインティングする手法の問題点として，固視微動がある．これは人間が 1 点に注目しているつもりでも，眼球が常に微細に動く現象である．たとえ視線計測器の計測精度が向上したとしても，一般的な GUI 上のターゲットを視線情報のみから特定するのは困難である [Yamato, 2001].

これとは逆に，画面上の特定のオブジェクトを注視していないことはより正確に判定できる．視線計測器の最大誤差は保証されており，さらに先行研究によって固視微動の上限値も判明しているため，検出された注視点の座標からこれらの誤差を合計した距離以上離れている位置は，確実に注視していないといえるからである．

そこで本章では，視線計測器を「確実に注視していない領域」の特定に用いて，カーソルを注視していない場合にのみ高速化させる手法を提案する（図 6.1）．これにより，カーソルを長距離移動させる時間を短縮するとともに，注視点付近では微細な操作を可能にし，高速化と操作精度の両立を図る．

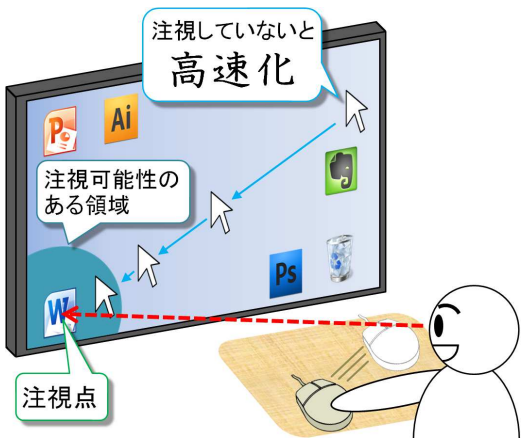


図 6.1　提案手法の概要．注視可能性のある領域の外側は，「確実に注視していない領域」であり，そこにカーソルが含まれていれば高速化する．

6.2　視線計測技術と計測機器の概説

ここでは視線計測技術について概説し，具体的な計測誤差について述べる．それを基に，一般的な GUI 環境においてユーザが注視したいオブジェクトを特定するのが困難である理由を説明する．また，近年の視線計測器の利用形態とその広がりについても述べる．

6.2.1 視線計測技術の分類

視線を計測する手法は，角膜反射法，強膜トラッカー法，EOG 法，サーチコイル法の 4 種類に大別される．それぞれに長所・短所があるが，強膜トラッカー法は頭部の固定，EOG 法は皮膚への電極貼り付け（位置に関するノウハウを要する），サーチコイル法は専用コンタクトレンズの装着が必要であり，日常的な PC 操作時に利用し続けることを考慮すると導入が比較的難しい．角膜反射法には頭部装着型と卓上型があり，後者は机上にアイトラッカーを設置するだけで頭部を固定する必要がないことから簡易に利用できる．そのうえ，性能が優れたものでは視角 0.5 度の高精度で視線を検出できる．以上の分類，性質については文献 [Ohno, 2002]が詳しい．

本章の実験では角膜反射法・卓上型の視線計測器を使用する．以降，本章で単に「視線計測器」と呼ぶ場合，このタイプを指すものとする．

6.2.2 計測誤差

本章の評価実験では，視線計測器に Tobii X60 を使用している．これは角膜反射法・卓上型であり，ユーザは使用前に 20 秒程度のキャリブレーションを行う．機器は 60 Hz で動作し，計測精度（誤差の最大値）は視角 0.5°である．

目からディスプレイまでの距離を l，視線計測器の計測精度を視角 θ とすると，ディスプレイ上での計測誤差は最大 $l \times \tan\theta$ となる．これを本章の評価実験環境（目からディスプレイまでの距離が約 65 cm，ディスプレイサイズが 27 インチ，ディスプレイ解像度が 1920×1200 ピクセル）に当てはめると，計算誤差は約 18 ピクセルになる（図 6.2）．また，固視微動は最大で瞬間角速度 1°に達するランダムな運動であることがわかっている [NTT, 2004]．これは Tobii X60 で検出できる 1/60 秒 (= 0.017 秒) あたりに換算すると最大約 0.017°であり，上記の実験条件下では約 1 ピクセルになる（図 6.3）．

したがって，今回使用する視線計測器から得られる情報は，検出された座標から半径約 19 ピクセル以内のどこかをユーザが注視している，ということのみである．

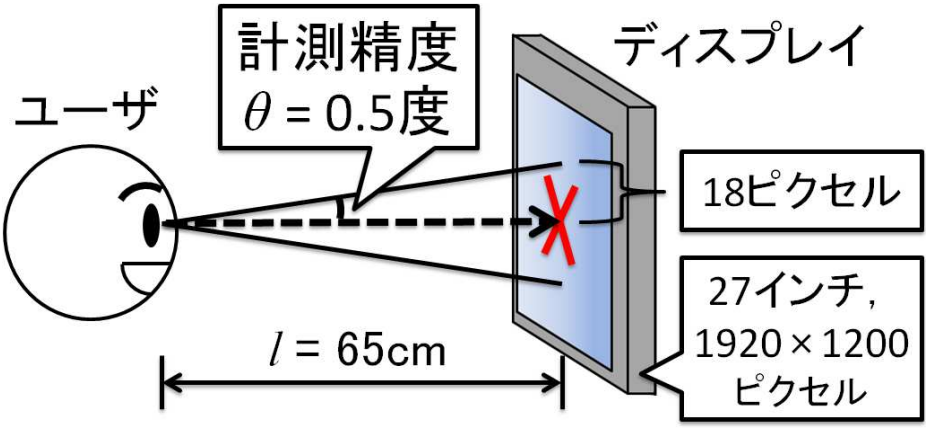


図 6.2　計測精度によるディスプレイ上の誤差範囲．×印の交点がユーザが見たい位置．

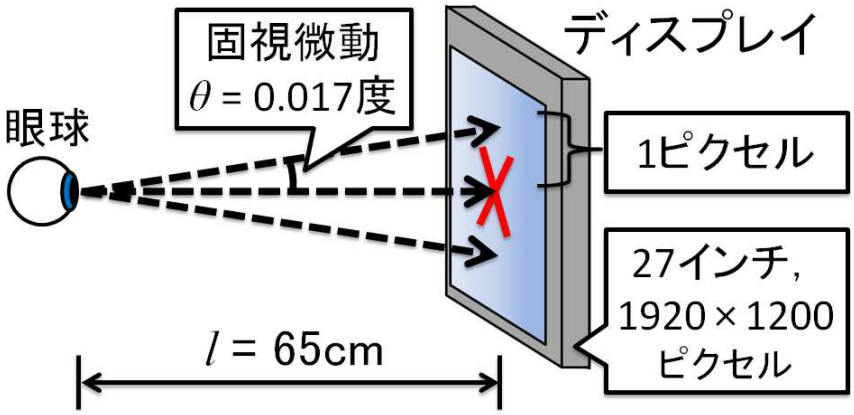


図 6.3　固視微動によるディスプレイ上の誤差範囲．×印の交点がユーザが見たい位置．

6.2.3 注視したいターゲットを特定することの難しさ

ユーザが特定のターゲットを注視していることを判定するには，注視点が一定時間だけオブジェクト上に停留する必要がある．この制約を考慮すると，一般的な GUI 環境において最大誤差 19 pixels/Hz というのは十分な精度とはいえない．図 6.4 は各種 OS のファイルエクスプローラウィンドウを拡大したものであるが，たとえばユーザが最大化ボタンや最小化ボタンの中心を注視しているつもりでも，システムからは閉じるボタンを注視していると判定される危険がある．

既存研究において「一定時間注視したボタンをクリックする」方式を採用したシステムでは，大きなボタンを配置することで選択ミスを防ぐよう試みられている．たとえば文献[Ohno, 1999]では横 100×縦 50 ピクセルのメニューが 40 ピクセルの間隔を空けて配置されており，計測誤差があっても問題なく選択できるように設定されている（[Ohno, 1999]の実験環境では 50 ピクセルが視角 1.4°に相当する）．このように，計測精度と固視微動の問題を

回避するためにはターゲットのサイズを大きくしなければならず，一般的な GUI 環境ではユーザが操作したいターゲットを特定するのは困難である．

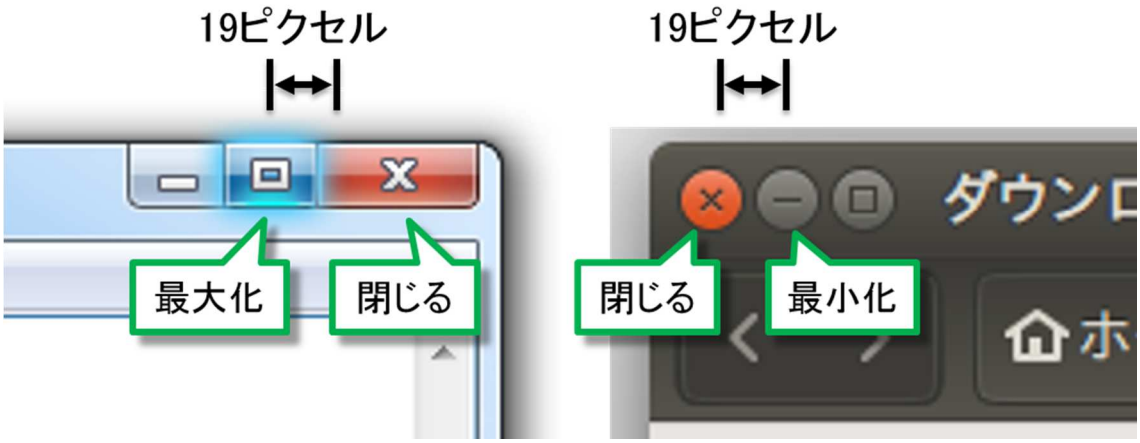


図 6.4　Windows7（左）と Ubuntu14.04（右）のウィンドウの拡大図．

6.3　提案手法

ユーザが注目している画面内の位置，すなわち注視点を計測することによるターゲットの特定が困難であることは既に述べた．これに対し，注視していない領域（以下，注視外領域と呼ぶ）は比較的容易に判別可能であるといえる．先に求めた合計誤差の 19 ピクセルは視線計測器の計測誤差や固視微動の最大値で計算したものであるため，検出された座標から 20 ピクセル以上離れていればユーザは確実に注視していないと判別できるからである．

本章では，この注視外領域においてカーソル速度を上昇させるポインティング支援手法を提案する．ユーザは画面内のターゲットをポインティングする際に，その位置を見てからカーソルを動かすことが多い [Zhai, 1999]．これを活用し，カーソルから離れた位置にあるターゲットを注視しているときに，そこへ向かうカーソルを高速化し，かつターゲット付近では通常時の速度に戻すことで微細な位置調整を可能にする．

カーソルを注視点付近へジャンプさせることで高速化する手法も多く提案されている [Drewes, 2009] [Fares, 2012] [Fares, 2013] [Yamato, 2001] [Zhai, 1999]が，現実的な作業においては必ずしもターゲットを注視しているとは限らないため，不都合が生じることがある．たとえば画面右端のスクロールバーを何度も上下に操作しながらウェブブラウジングする状況を想定すると，カーソルが注視点へ移動してしまうのは望ましくない．またこの問題を回避するためにジャンプ機能の ON/OFF を作業中に切り替えるのは煩雑である．これに対し提案手法では，注視外領域でもカーソルは問題なく移動できる．移動が高速化されているために微細な操作ができない懸念はあるものの，あくまで OS の設定可能な範囲での高

速化であり，大きな問題は生じないと考えられる．提案手法は，カーソルを注視点にジャンプさせるほどの時間短縮にはならないものの，現実的に起こりうる問題を回避しつつ着実に操作効率を改善することを狙ったものである．

6.4 関連研究

ここではポインティング時間を短縮する手法のうち，特に長距離のカーソル移動を支援するものについて述べる．

視線のみでポインティングすると，マウス操作よりも高速にターゲットを選択することが可能であるとされている [Sibert, 2000]．この実験では 5.8 cm 間隔で並んだ直径 2.8 cm の円を選択するタスクを設定し，150 ms 注視する手法が採用されていた．しかし実際の GUI 環境では，図 6.4 のようにターゲットが密に並ぶこともあり，視線のみでの高速な選択は困難であると考えられる．また肢体不自由者を対象とした視線のみでのポインティングの実験も行われている [Kuno, 1998]．視線のみでのポインティングは，意図的な視線移動や停留とそれ以外との判別が困難である（Midas touch problem と呼ばれる [Jacob, 1991]）ことから，一般的な GUI 環境ではかえって操作効率が悪くなる可能性がある．近年では，ウェブページのリンクごとに異なる色をつけ，それと同じ色の矩形を注視することで選択する手法 [Lutteroth, 2015]や，ターゲットごとに異なる動き方をさせて，目で追ったものを選択する手法 [Vidal, 2013]が提案されている．これらは専用のターゲットを設置する必要があったり，固定されたターゲットに適用できないなどの限界があるが，制約を課さないのであればマウスよりも高速に選択できる [Lutteroth, 2015].

テーブルトップインタフェースなどの大型ディスプレイを用いた操作では，腕を伸ばして遠くにあるターゲットをタッチするのは負担が大きいことから，視線情報を用いてタッチ座標にオフセットを付加する手法が提案されている [Voelker, 2015] [Pfeuffer, 2014] [Pfeuffer, 2015]．これらの手法では，手元でのタッチ操作をユーザの見ている場所で発生したイベントとして扱うことで，擬似的にマウスのような間接制御を実現している．

6.5 実装

視線計測には Tobii X60 Eye Tracker を使用し，60 Hz で注視点を計測する．カーソルの速度の変更システムの実装には HSP3.3 および Win32API を使用する．本システムは Windows7 のコントロールパネルにおける「ポインターの速度を選択する」の設定値を動的に変更する．Windows7 ではカーソルの速度が 1〜20 の整数値で設定され，デフォルトでは 10 である．本システムは，注視点とカーソルの座標に応じて速度 *speed* を次式で決定する．

$$speed = l_{eye\_cursor} / (l_{max} / steps) + speed_{min}$$

上式において l_{eye_cursor} は現在の注視点とカーソルの距離，l_{max} は注視点とカーソルの距離がとりうる最大値（つまりディスプレイの対角線の長さ），*steps* はカーソル速度が変化する段階数，$speed_{min}$ はカーソルの最低速度である．本章の評価実験では $speed_{min}$ を OS の標準速度である 10 に設定し，最大値は 20 に設定した．すなわち *steps* は 11 となる．実験に用いた 1920×1200 ピクセルのディスプレイの場合，例として注視点が画面左上の座標 (x, y) = (0, 0) にあるときには，カーソルの位置によって図 6.5 のように速度が決定する．上式における $(l_{max} / steps)$ の値は，図 6.5 においてカーソル速度が変わる距離の閾値である 205 となる．これは注視外領域を特定可能な 20 ピクセルを大幅に上回っていることから，カーソル移動を高速化するのは注視外領域であることが十分に保証される．提案手法と同様に，注視点から遠いほどカーソルを高速化する手法が近年提案されているが [Velloso, 2015]，Velloso らは周辺視野にあるカーソルが速度 20 になるよう設定しており，最大速度の領域を非常に広くとることで時間短縮を目指している．本研究ではユーザがカーソルの高速化に気づかない程度に滑らかな速度変更をする．この点については実験参加者に事後アンケートを行って調査する．

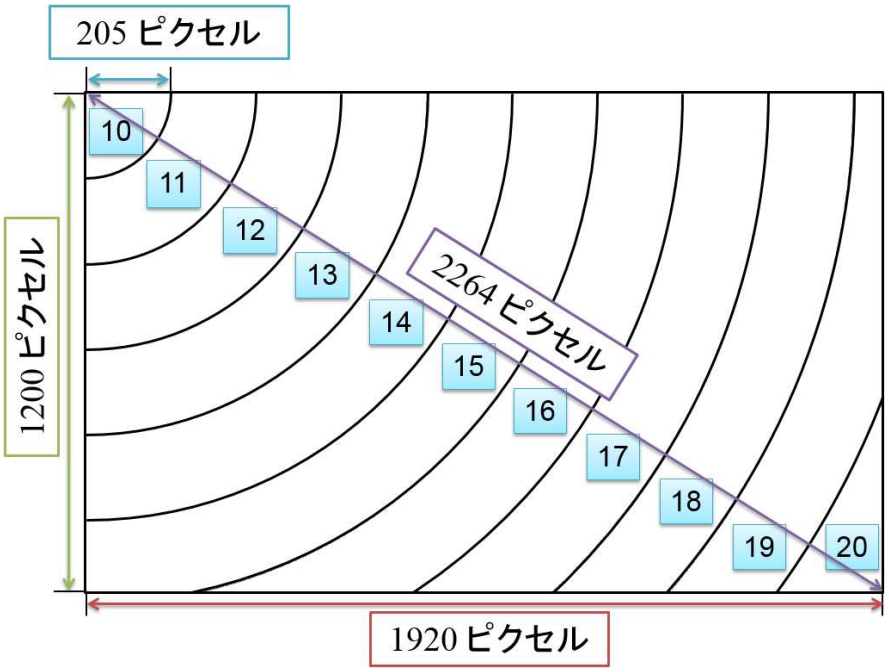


図 6.5　注視点が画面左上隅にある場合の，カーソル位置に応じた速度のマップ．Windows7 の設定値の 10（標準速度）から 20（最高速度）の範囲で変化．

6.6　評価実験

タイル状に配置された矩形ターゲットを順次クリックしていくタスクによって，提案手法の有効性を検証する．これはデスクトップアイコンをカーソルで選択する操作を模したものであり，PC 操作において日常的に行う作業である．以下に詳細を記す．

6.6.1 タスク

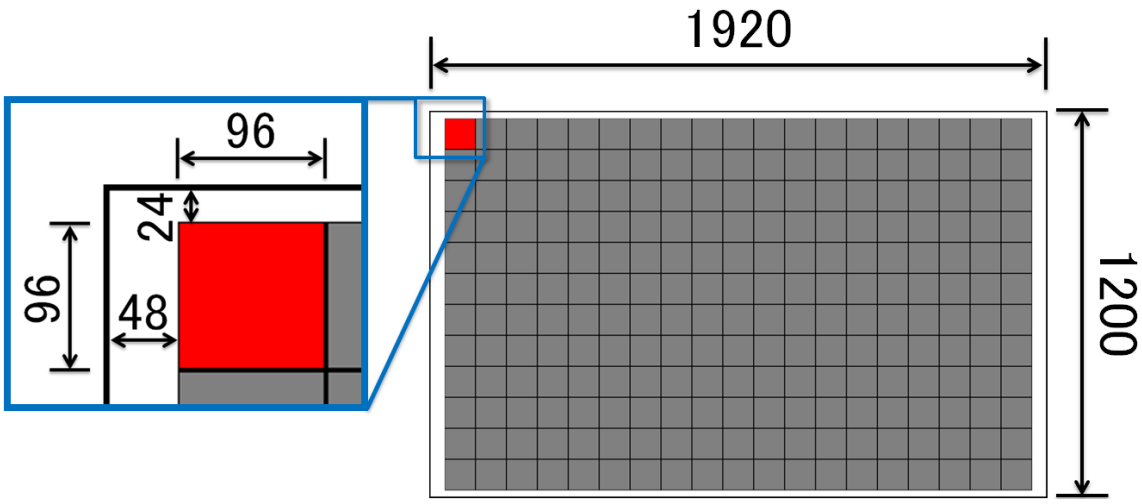


図 6.6　タスク開始時の画面と拡大図．数値の単位はピクセル．
ターゲットのみ赤色，他は灰色で描画される．

図 6.6 のように隣接した灰色の矩形の中から，1 つだけ赤色で描画されたターゲットをクリックしていく．矩形の個数は，Windows7 のデスクトップに中サイズのアイコンを敷き詰めた場合と同じ 12 行 19 列の 228 個である．画面端に置かれた矩形が無限大のサイズをもたないようにするため，画面の左右端に 48 ピクセルずつ，上下に 24 ピクセルずつの隙間を設けた．矩形 1 個のサイズは外周 1 ピクセルの境界線を含めて縦 96×横 96 ピクセルとした（実測で一辺約 29 mm）．

実験開始時には画面左上の矩形がターゲットとなり，カーソルはその中心に自動的に移動する．参加者がこの矩形をクリックした時点から操作時間の計測を開始する．以降，ターゲットをクリックする度に次のターゲットが決定されて赤色になり，最初の 1 個を除く 30 個のターゲットをクリックした時点で 1 ブロックの試行が完了となる．選択エラーは，ターゲット以外の領域をクリックした場合にカウントされる．

提案手法では注視点からターゲットまでの距離が長いほど高速化されるため，次のターゲットの出現位置によって有効性が変化する可能性がある．そこで，ターゲットまでの距

離による有効性の違いを検証するために，近距離から長距離までが均等な回数だけ出現するように設定する．連続する 2 回のターゲット位置が，ディスプレイの横幅を 100%としたとき 10%区切り（192 ピクセル）の範囲内で各 3 回ずつ，1 ブロックで合計 30 回出現するようにし，この制約内でランダムな位置になるようにした．10%区切りというのは，厳密にはディスプレイ横幅に対するターゲット距離の割合を l_{percent} として次の範囲である．

$$n \times 10 \leqq l_{\mathrm{percent}} < (n+1) \times 10 \quad (n = 0, 1, \cdots, 9)$$

また，同一の矩形が連続してターゲットに選出されることはないことを制約に加えた．

6.6.2 カーソル速度

本実験では以下のように，OS の標準速度をベースラインとし，単純に高速化した場合と提案手法の 3 種類でタスクを試行する．

- [速度 10]　常に OS の標準速度
- [速度 20]　常に OS の最高速度
- [提案手法]　速度 10〜20 の変速

使用したマウスは Buffalo BSMOU05M (1000 dpi)である．実距離でマウスを 1 インチ (2.54 cm)動かすと，カーソルは画面内で (1000 × *speed* / 10)ピクセル移動する．また，「マウスを速く動かしたときはカーソルが大きく，遅く動かしたときは小さく移動してほしい」という要求で一般的に利用されるのはソフトウェア加速であると考えられるため，実験ではコントロールパネルの「ポインターの精度を高める」のチェックを ON にしている．

6.6.3 手順

ディスプレイは Dell 2707WFP（27 インチ，1920×1200 ピクセル）を用いた．実験参加者にはマウスパッドの位置や椅子の高さを調整させ，姿勢を正した状態でディスプレイ全体が視認できることを確認した．顔とディスプレイの距離は約 65 cm，目と視線計測器の距離は約 45 cm である．実験時の参加者と機器の配置を図 6.7 に示す．

参加者は大学生及び大学院生の 15 名（男性 14 名，女性 1 名，平均 22.5 歳）であり，全員右利きで，マウス操作に習熟している．タスク内容やエラーとなる操作を教示したうえで，可能な限り速くかつ正確に操作するよう伝えた．

参加者には 1 種類のカーソル速度につき 30 回の選択を 5 ブロック試行させた．ブロックごとに休憩を 10 秒間，カーソル速度変更時に 2 分間設け，参加者が集中して実験に臨めるよう配慮した．全てのカーソル速度でタスクを試行した後，主観評価を行うためのアンケートを実施した．

各カーソル速度でタスクを試行する前に，カーソル速度および提案手法における速度変化のアルゴリズムを参加者に伝えた．ただし試行前や休憩中にマウスを動かして速度を確認する行為は認めなかった．また，タスクの習熟の影響を平滑化するため，試行するカーソル速度の順序は参加者 5 名ごとに変更している．

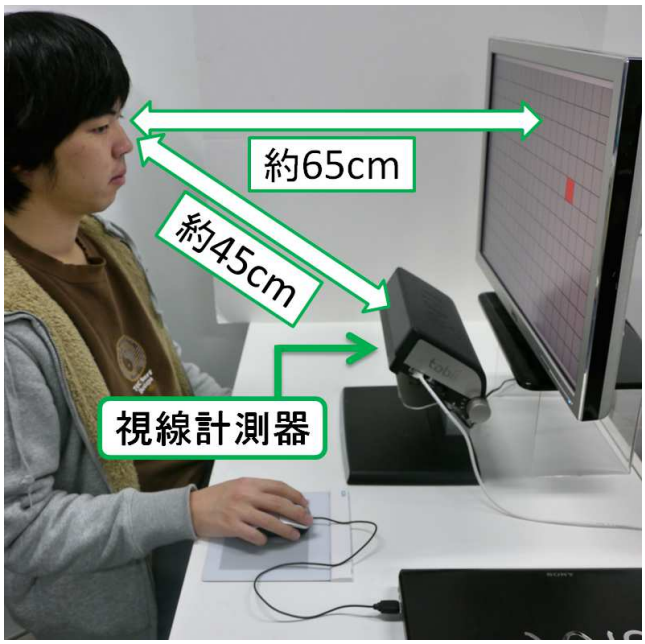


図 6.7　実験時の参加者と機器の配置．

6.6.4 結果

ターゲットを 1 つクリックするのに要した選択時間とエラー率を，ターゲットの出現距離ごとに反復測定（カーソル速度を要因とした一元配置分散分析）した結果を表 6.1 に示す．また，出現距離ごとの選択時間を図 6.8，エラー率を図 6.9 に示す．図 6.8 と図 6.9 の中でアスタリスク（＊マーク）を付しているのは，3 種類のカーソル速度をペアごとに多重比較し，ベースラインである速度 10（標準）との間に有意差または有意傾向が見られた箇所である．多重比較には Bonferroni の手法を用いた．

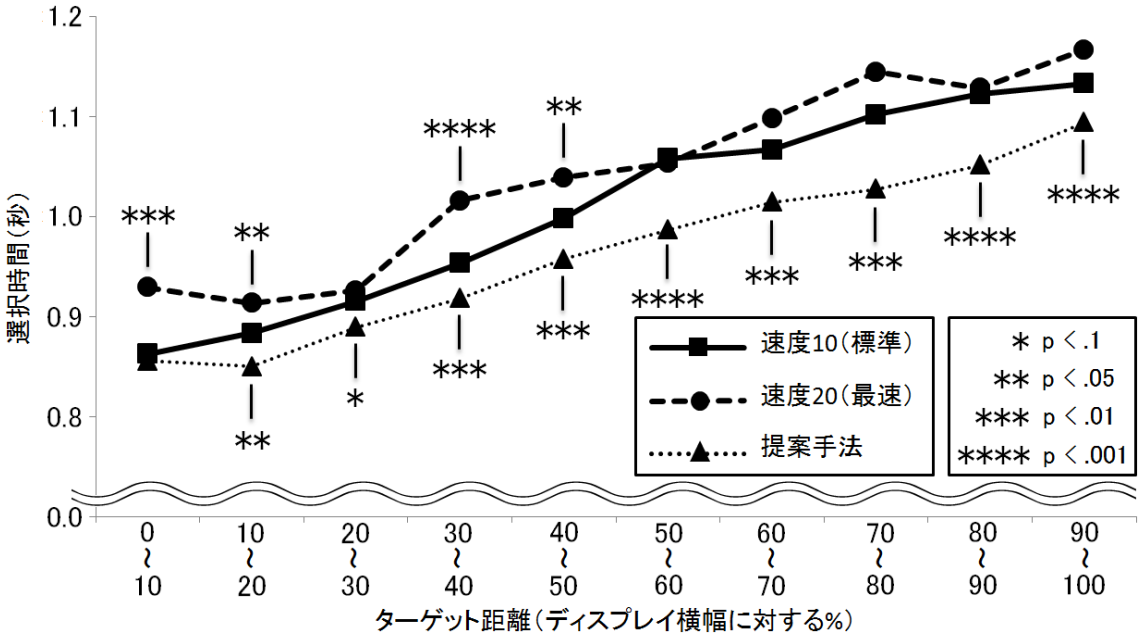


図 6.8　選択時間．速度 10（標準）と比較して，速度 20（最速）が有意に遅い箇所には＊マークをグラフの上側に付し，提案手法が有意に早い箇所には下側に付している．

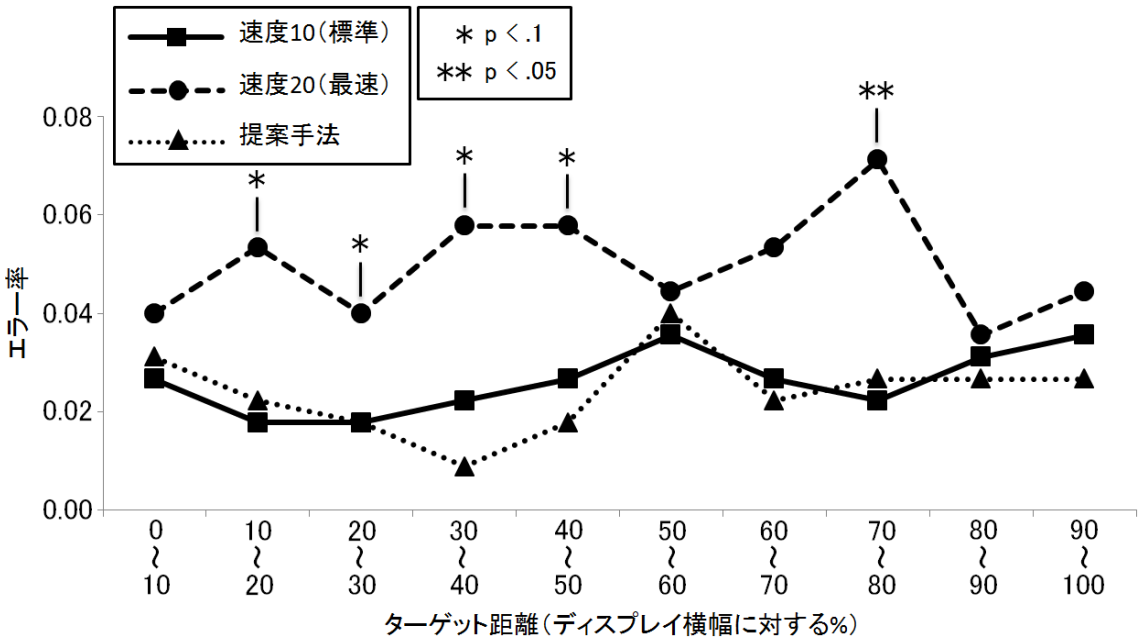


図 6.9　エラー率．速度 10（標準）と比べて速度 20（最速）が有意に高い箇所には＊マークを付している．速度 10（標準）と提案手法の間には有意差が見られなかった．

表 6.1　ターゲットの出現距離ごとの選択時間及びエラー率の分散比（F 値）と有意確率．分散比はいずれも $F_{2, 28}$ の値．
色が塗られているのは，3 種類のカーソル速度の間に有意差または有意傾向が見られた箇所．

	選択時間		エラー率	
距離	分散比	有意確率	分散比	有意確率
0－10%	12.484	p < .001	0.189	p < .001
10－20%	9.160	p < .01	2.771	p < .1
20－30%	3.068	p < .1	3.182	p < .1
30－40%	24.546	p < .001	7.850	p < .01
40－50%	14.376	p < .001	2.904	p < .1
50－60%	11.978	p < .001	0.135	p = .875
60－70%	7.945	p < .01	2.488	p = .101
70－80%	14.261	p < .001	3.895	p < .05
80－90%	8.6927	p < .01	0.118	p = .889
90－100%	11.955	p < .001	0.427	p = .656

6.7　考察

図 6.8 より，いずれのターゲット距離においても提案手法はベースラインより高速にポインティングできたことがわかる．さらに，ターゲット距離がディスプレイ横幅の 10%（192 ピクセル）以上であれば有意に短縮できていることから，提案手法の有効性が示された．ディスプレイ上における 192 ピクセル（実測約 5.8 cm）は視角 5.1°に相当することから，提案手法は一般的に視角 5.1°以上離れたターゲットをポインティングするときに有効にはたらくといえる．ターゲット距離が 0%～10%では有意な時間短縮が見られなかったが，これは提案手法においてカーソルが高速化される 205 ピクセルよりもターゲット距離が短いことから考えて妥当である．

図 6.9 のエラー率に関して，全てのターゲット距離において提案手法とベースラインの間に危険率 5%で差が見られなかった．したがって，提案手法は操作精度を落とさずにポインティング時間を短縮できると示された．一方で，速度 20（最速）はターゲット距離によってエラー率が高くなる傾向が見られた．選択時間もベースラインより有意に増大することがあり，単純に 2 倍のカーソル速度にするのは避けるべきであることが読み取れる．

カーソルの制御しやすさに関するアンケート結果を表 6.2 に示す．これによると，提案手法はベースラインよりもカーソルを制御しやすいと感じられていることがわかる．自由記述のアンケートでは，「注視点にカーソルが近づくと，わかるかわからないかという程度で速度が落ち，最後のマウス移動の調整がしやすかった」，「提案手法ではターゲット付近

でカーソルが減速するので，素早くマウスを動かしてもうまくクリックできた」といった意見を得られた．一方で，速度 20（最速）はベースラインと比べて制御しづらいと感じられていることが表 6.2 からわかる．図 6.9 において速度 20（最速）のエラー率がベースラインより高かったことが示されているが，これが操作しづらさの一因だと考えられる．自由記述でも，速度 20（最速）は速すぎて操作しづらいと回答した参加者が 15 名中 8 名，ターゲットをオーバーシュートすることが多かったと指摘した参加者が 5 名いたことから，参加者が意識できるほど操作性が悪化していることがわかる．

表 6.2　カーソルの制御しやすさのアンケート結果の平均値．
5 段階評価で高いほど制御しやすい．

速度 10（標準）	速度 20（最速）	提案手法
3.1	2.3	4.2

提案手法に関するアンケート結果を表 6.3　提案手法に関するアンケート項目と回答（人）に示す．速度が動的に変化することを意識していた 2 名は，特に注視点付近で減速するのを意識したと回答した．さらにその 2 名とも，この速度変化が操作のしやすさや時間短縮にはたらいていると述べている．また，タスクの試行中に視線を制御することを意識していた参加者は 7 名おり，このうち 3 名はカーソルを動かすより先にターゲットを見るように意図的にしていたと回答し，他 1 名は少々目が疲れたように思うと述べた．これは実験前に提案手法のアルゴリズムを伝えていたため，自身の目の動きを意識してしまったことが一因だと考えられる．普段の作業時には意識してターゲットを見つめようとしないため，提案手法によって自然な操作ではなくなっている点であるといえる．また，視線を制御することを意識しなかったと回答した 7 名のうち 6 名が「ターゲットを見てからそこへカーソルを移動させる操作は普段と同じであり，意識することはなかった」という旨の所感を述べた．これは提案手法の狙いである，自然な視線の動きを利用することで操作効率を向上するというシステム設計がうまく機能したものと考えている．

表 6.3　提案手法に関するアンケート項目と回答（人）

質問内容	はい	いいえ	どちらでもない
速度の動的な変化を意識したか	2	11	2
視線を制御することを意識したか	7	7	1

自由記述によるその他の回答では，「視点付近ではカーソルがもっと低速になるとよい」と述べた参加者がいた．本章の実験よりも微細な制御が求められる作業，たとえば長文から任意の文字列を選択する操作などにおいては，カーソル移動を低速化することで作業全

体の時間短縮を図れる可能性があると考えられる．今回はOSの標準速度及び最高速度との比較実験のみ実施したが，標準より低速にするのが有効な作業においては，注視点から離れるほど標準速度に近づき，注視点付近では低速にすることで操作性・操作時間が改善される余地があると考えられる．

6.8　まとめと今後の課題

視線計測器によって確実に注視していない範囲を特定し，その領域においてマウスカーソルを高速化させ，ポインティング時間を短縮する手法を提案した．評価実験により，カーソルからターゲットまでの距離がディスプレイ横幅の10%（視角5.1°）以上のとき選択時間が短縮され，かつエラー率は上昇しないことから有効性を示した．

本章では1回の選択が短時間で完了するタスクを設定したが，今後は提案手法を長期的に利用したときの影響についても調査したい．特に，タスク試行中に意識的に目を動かしたと回答した参加者がいたことから，長時間の作業ではこの負担が高くなることが懸念される．一方で，提案手法を長期間使用することで，次第に目の動きを意識しなくなっていくことも考えられる．すなわち，目の動きに自然と手が連携するような協調的な挙動になっていく可能性もあると著者は考えており，この点について調査したい．また，利用継続時間が増大するにつれて使用中のカーソル速度に慣れていくと述べた参加者がいるため，習熟度の向上による操作時間の変化を観察したい．

システムの改良案として，カーソルが注視点の方向へ移動するときのみ高速化するアイディアがある．こうすることで，ユーザにとって望ましくない加速を回避することが期待できる．現在のシステムは注視点からの距離によってカーソルを一律に高速化する設計であり，先に挙げたスクロールバー操作ではカーソルは注視点から離れているため高速に移動してしまう．こういった場面で，注視点方向への移動だけ高速化させることで，スクロールバーを上下に動かすときには意図しない高速化を防ぎ，操作性を向上させられると考えられる．さらに，本章の実験ではカーソルの最高・最低速度や，速度変化の閾値（*steps*の値)などを著者が設定したが，これをカスタマイズ可能にすることで個々のユーザに適したカーソル速度が得られる．これらの改良を行ったうえでの評価実験も実施したい．

第7章　結論

本論文では到達運動の分析と支援を目的とした研究の成果について述べた．具体的には，ポインティングにおける粗い制御の操作支援手法の開発と，ポインティングの基礎的な挙動分析およびステアリングのパフォーマンスモデルの修正を行った．第 1 章では近年のコンピュータ利用時の GUI 操作の負担について述べ，到達運動の支援の必要性を説明した．また操作支援手法を開発するにあたり，定量的な評価をするために挙動の分析と正確なパフォーマンスモデルの構築が重要であることを説明した．第 2 章では，パフォーマンスモデルと操作支援手法に関する先行研究を整理し，本研究で取り組む研究対象を述べた．第 3 章から第 6 章では，無限大サイズをもつターゲットポインティングの分析，幅の変化する経路を通過するパフォーマンスモデルの修正, 細長いターゲットを DnD する操作支援手法, 視線計測器を用いた長距離移動の支援手法を説明した．以下では各研究の貢献および本論文全体の貢献をあらためて整理し，今後の展望を述べる．

7.1　総合的な議論

7.1.1 各研究の学術的貢献

7.1.1.1 分析とモデル化

第 3 章では無限大のサイズをもつターゲットのポインティング操作を分析した．これはカーソル操作が極端に粗くても操作を完遂させられるタスクであり，これまで十分に分析されていなかった．実験におけるマウスカーソルの速度波形を分析すると，サイズの規定されたターゲットを選択する場合とは異なる特徴をもっていることがわかり，ユーザは通常のポインティング時とは異なる戦略でマウスを動かしていることを明らかにした．また操作時間 MT が，ターゲットまでの距離 A と線形の関係があることを示した．さらに，ターゲットの可視領域のサイズ W_v は操作時間 MT に影響しないことを確かめた．これはサイズ H が規定されたターゲットが画面端に設置されているときに，選択時間 MT が W_v の影響を受けるとした先行研究の結果 [Appert, 2008] [Farris, 2003]とは異なる知見である．つまり，H を無限大にするか否かが MT に影響することを初めて指摘したのが本研究である．

先行研究ではターゲットサイズ W と H のうち，一方か両方を規定したタスクでの分析が多く，両方を無限大にしたタスクは詳細に分析されていなかった．GUI 環境では両方が無

限大のターゲットをポインティングする事例があるため，操作時間やエラー率に対する実験パラメータの影響を分析するのは基礎的な研究として有意義である．本研究の貢献は，第一に W と H の両方を無限大にしたポインティングタスクを分析したことにあり，第二に実験結果から無限大サイズのターゲットポインティング特有の性質を明らかにしたことにある．今回の実験では単一デバイス（マウス）を使用した左右方向のポインティングタスクについて検証したが，本研究を足がかりとして今後も多くの条件下で実験が行われ，粗い制御で済む操作の分析が進むことを期待する．

第 4 章では，幅が変化する経路のステアリング時間が通過方向によって異なることを取り上げた．元々のステアリングの法則では幅の変化方に関わらず同一の難易度が算出されるが，実際には狭まる方向へ通過すると操作時間が増大するため，難易度を高く算出しなければ整合性がとれない問題があった．そこで本研究では幅の変化方向による難易度の差を算出するモデルを構築し，実験によって妥当性を確かめた．またこのモデルを使用することで，少数の実験データから狭まる方向の通過時間を高精度に予測できることを示した．

先行研究でも，同一の難易度が算出されるが通過時間の異なる事例は発見されている．当然ながらポインティングデバイスが異なれば通過時間も変わり [Accot, 1999]，またカーソルのサイズ [Naito, 2004] [Naito, 2006]や経路のスケール [Accot, 2001] [Pastel, 2006]によっても操作時間は変化する．逆に言えば，デバイス・カーソルサイズ・経路のスケールの 3 条件を揃えた場合，難易度 ID が同じであれば操作時間も同一になるというのが従来の知見であった．これに対して本研究では，上記の 3 条件を揃えても操作時間が変化する要素として，幅が変化する経路の通過方向があることを発見した．従来のモデルでは行きも帰りも同じ時間で通過すると計算されてしまう問題を，本研究において初めて指摘したのが第一の貢献である．さらに，通過方向による難易度の差 ID_{Gap} を理論的に導出し，実験データが高精度にフィットすることを確かめたのが第二の貢献である．この難易度差モデルを利用することで，広がる方向への通過時間を多くの実験パラメータで計測しておけば，狭まる方向への通過時間を少ない実験データ（最低 ID における操作時間）から高精度に予測できるようになる．よって本研究は，制御の粗さが連続的に変化するタスクのモデル化についての価値をもち，さらにユーザ（操作時間の測定者）がタスクを試行する負担を軽減できるという点で操作支援手法としての価値ももつといえる．

ステアリングの法則は Accot と Zhai によって 1997 年に提案され [Accot, 1997]，現在までにこの論文が引用された回数は 529 回にのぼる（Google Scholar Citations による． 2015 年 12 月 20 日参照）．ステアリングの法則は人間工学やバーチャルリアリティなどの分野からも参照され，GUI 研究以外にも強い影響をもつモデルであり，モデルの修正や適用範囲を検証する研究の意義は大きいといえる．今後，より適切なモデルが導出されたり，本研究の仮説と異なる戦略をとっているはずだと主張する研究が発表されるのであれば，本研究は後の議論の土台として大きな貢献を果たすはずである．

7.1.1.2 操作支援

第 5 章では細長いターゲットをクロッシングで DnD する手法 Cross-drag について述べた．Cross-drag はトリガを押したままターゲットに衝突することでドラッグを開始する手法であり，ターゲットにカーソルを乗せるための微細な制御が必要ない．ターゲット近辺での制御を支援する手法はこれまでに多く提案されているが，それらの先行研究と比較して Cross-drag の利得を考察し，Cross-drag が効果を発揮する条件や，有効に機能しない条件をまとめた．さらに評価実験によって DnD 作業全体の時間を短縮することを示し，従来手法よりも操作効率が改善されることを明らかにした．

細長いターゲットを選択しづらい原因は，フィッツの法則から考えるとターゲットサイズ W が小さいことに起因しているため，W を大きくして問題解消を狙うのがいたって真っ当である．したがって，直接的にターゲットを大きくする手法や，逆にカーソルサイズを拡大する手法，カーソルを低速化することで相対的に W を大きくするのがこれまでの一般的な手段であった．しかしこれらの手法はターゲット周辺の領域を侵略したり過度な減速が起こったりする不利益を生じさせる問題があった．W を大きくする手法は，いわばコンピュータが人間側に合わせて性能を劣化させることであり，望ましくないアプローチであると著者は考える．すなわち，入出力の粒度に関して，コンピュータ側はドットを非常に小さく描画できるのに対し，人間は手をそこまで微細に制御するのが困難なため，ターゲットサイズを大きく確保する方法が広く採用されているのである．それに対して本研究では，ターゲットの幅 W が小さいことはそのままに，ターゲットの長さ H が大きいことを利用する操作手法を提案した点で新規性がある．また，クロッシングを利用した操作手法では専用ウィジェットを実装することが多かったのに対し，本研究では既存の GUI オブジェクト自体は改変せずにクロッシングを活用できるように実装しており，これも従来研究には見られなかった点である．したがって，現状の PC 環境における GUI オブジェクトを改変せず，またユーザに新たな不利益をもたらさない制約を守りつつ，粗い制御で十分正確にかつ短時間で操作できる手法を構築したのが本研究の貢献である．また，細長いターゲットをクロッシングで選択する利得を既存手法と詳細に比較しており，この議論自体が新たな手法を考案するさいに活用できる知見になると考える．

第 6 章では，視線計測器を用いて長距離のカーソル移動を支援し，ポインティングを高速化する手法について述べた．ポインティングの目標地点では微細な操作をしたいのに対し，そこに至るまでの移動時には粗い制御だけでよいため，注視点から遠くにあるほどカーソルを高速化して時間短縮するのが狙いであった．評価実験ではターゲットを隙間なく隣接させたポインティングタスクを行い，エラー率を上昇させずに操作時間を短縮できることを確認した．長距離の移動は，まず大雑把な操作でターゲットまでの距離を縮め，その後に微細な制御で選択操作をしたいが，単純に高速化すると終盤の操作も粗くなってし

まい，ベースラインよりも操作時間が増大することも判明した．この結果から，操作の粒度を適切に粗く/細かくしなければ，かえってユーザの負担を重くしてしまうことが確認された．

視線情報を利用したポインティング支援手法は従来から提案されており，それらに共通するアイディアは「ユーザはポインティングしたいところにまず目を向ける」という性質を利用するものである．最も代表的な手法は MAGIC [Zhai, 1999]であり，これはカーソルの長距離移動を視線で行い，ターゲット付近での微細な位置調整をマウスで制御するものである．MAGIC をさらに改良した手法も多く提案されているが，基本的には注視点付近までカーソルをジャンプさせる方法を採用している．しかし，カーソル座標と注視したい位置が離れている場面（スクロールバー操作など）があるため，注視点付近にカーソルをジャンプさせると既存の GUI 操作が阻害される場面があることを指摘した．よって本研究ではカーソルをジャンプさせる手法は避けるべきだと主張し，既存の操作に悪影響を与えないことを守りつつ操作時間を短縮する手法を検討した．提案手法での操作時間短縮は最大 10%程度だったが，カーソル速度を上昇させているにも関わらずエラー率は悪化していないことから，注視していない領域で高速化する手法の有効性を示した．操作で本研究の学術的貢献は，ポインティング支援における視線情報の利用方法として求められる/避けるべき事項を明らかにしたことと，その指針に沿って設計した支援手法の有効性を確認したことである．

7.1.2 本論文全体の学術的貢献

本論文は粗い到達運動の分析と支援を目的としていた．そのために，定量的な評価や比較議論ができるように基礎分析やパフォーマンスモデルの構築が重要であることを説明した．そのうえで，パフォーマンスモデルによって有用性が裏付けられた操作支援手法を開発し，ユーザの利便性を向上させることが必要であると述べた．

本論文では，まず第 2 章で既存の GUI 研究を整理した．また現在までに検討が不十分だと思われるタスクをピックアップし，本論文で具体的に取り組むべき研究対象をまとめた．そして第 3 章から第 6 章において各研究に取り組み，全ての研究において実験的に有効性・妥当性を確認した．各章において取り組んだのは次のような研究である．

- 第 3 章，基礎的な分析：　　無限大のサイズをもつターゲットのポインティング
- 第 4 章，モデル化：　　幅の変化する経路を逆向きに通過する難易度
- 第 5 章，局所的な制御の支援：細長いターゲットのドラッグ開始操作
- 第 6 章，大局的な移動の支援：視線情報によるカーソル速度の動的な変更

第 5 章の Cross-drag では，ターゲットサイズ W が 2 pixels ときわめて微細な制御を要する実験を行い，従来のポインティングでは選択が非常に難しいことを確かめた．これとは対

照的に，第 3 章ではサイズ W が無限大と極端に粗い制御で済む実験を行い，実験結果に影響するパラメータや MT の予測式について検討した．これら 2 つの研究ではターゲットサイズが小さい/大きい場合の操作支援と基礎分析を行った．また第 6 章では粗い/細かい制御が 1 回の試行中に変化していくポインティング操作を支援した．これは最終的な位置決めで成功/失敗が決まるポインティング操作を支援したが，第 4 章では移動中の微細さも要求されるステアリング操作の微細さが連続的に変化するタスクをモデル化した．以上の全 4 種類の研究によって，本論文の目的であった「分析やモデル化が不十分であれば基礎的な分析を行い，モデルが強固であれば支援手法を開発する」が果たされたと考える．粗い到達運動におけるユーザの挙動を理解し，操作の負担を軽減するための知見を提供したことが本論文全体の貢献である．

7.2 展望

第 3 章から第 6 章で各研究における今後の課題を述べたが，ここではより総合的な展望と今後への期待を述べる．

7.2.1 人間工学分野，心理学分野との協調

フィッツの法則やステアリングの法則が到達運動における代表的なパフォーマンスモデルであることを述べたが，分析やモデル化に関するインクリメンタルな研究として重要なのが，従来とは異なる条件下でもモデルがフィットするか検証することである．フィッツの法則自体が元々は心理学分野のテーマを扱ったもので，実世界のスタイラス操作が対象であり，それがコンピュータ操作を対象とする GUI/HCI 研究に導入された経緯がある．つまりパフォーマンスモデルの研究は根本的に領域横断的なのである．今でこそ GUI 研究では 2 次元平面上でのタスクを扱うのが一般的だが，既存のモデルが多くの条件下で成立することを検証すれば，今度はより広く HCI 分野や人間工学分野に貢献できる．

人間工学分野への発展の一例が，3 次元的に制約のある経路を通過するステアリングタスクである [Liu, 2011]（図 7.1）．つまり，「PC の画面操作はこの数式でモデル化できる」という知見からスタートして，「人間の挙動はこういうものである」という普遍的な知見が得られる可能性を秘めているのである．そのためには，最終的にモーションキャプチャシステムなどの大掛かりなセットアップで検証する必要があるが，まずは 2 次元の GUI 環境で基礎的な調査をするのがよいと考える．実験環境の構築が容易であり，条件のコントロールに関する知見も集約されているため，研究基盤として GUI は非常に強力である．

本論文では扱わなかったが，動く物体をポインティングするのも到達運動として基本的

な動作の一種であり，これに関する人間工学分野の研究 [Hoffmann, 1991]や心理学分野の研究 [Tresilian, 2002]も存在する．たとえばスポーツではボールのキャッチや打ち返し，ガンシューティングゲームでは歩行中の敵キャラクタに照準を合わせて撃つ操作などがこれに該当する．これは GUI 研究でも調査が進んでいるテーマであり，2 次元平面上 [Brenner, 2007] [Hajri, 2011]や 3 次元空間中 [Ortega, 2013]を動く物体のポインティング動作に適用させるため，移動速度や角度などをフィッツの法則に導入したモデルが提案されている．また，本論文の第 5 章では細長いターゲットをポインティングではなくクロッシングで選択する支援手法について述べたが，移動中のターゲットについてもクロッシングで選択ミスを減らす手法が提案されている [You, 2014]．また残像のように尾を引かせることでターゲットサイズを拡大する手法 [Gunn, 2009] [Hasan, 2011]や，MAGIC を利用する手法 [Hild, 2014]なども提案されており，最近 5〜6 年ほどで盛んに扱われているトピックである．本論文では静止したターゲットや経路を対象にした到達運動を扱ったが，これらのように数年前から新しい研究テーマが流行したり，また 1997 年に全く新しいパフォーマンスモデル（ステアリングの法則）が発表されたりと，GUI における到達運動はまだ解決すべき課題を抱えているとともに，大きな発展の余地を残していると考える．

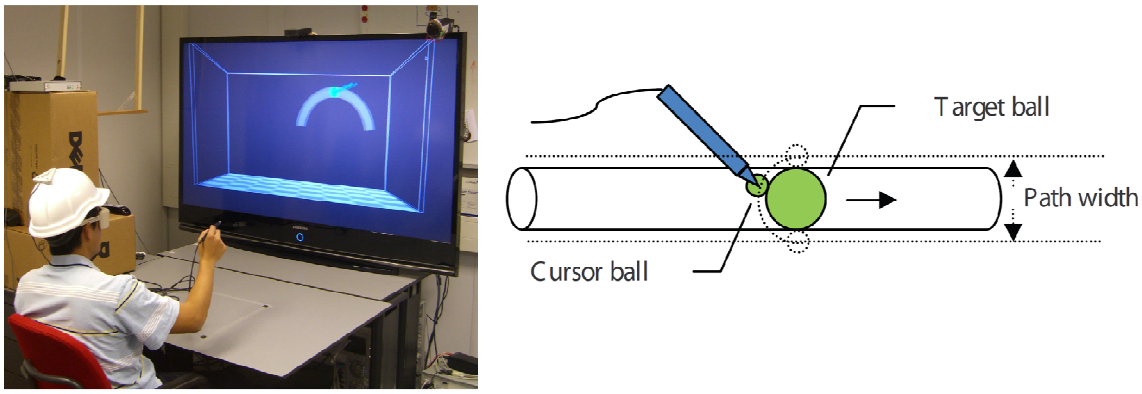


図 7.1　ボールを 3 次元空間内で制御してパイプを通過させるステアリング [Liu, 2011].

7.2.2 ハードウェアやソフトウェアとの協調的な発展

ハードウェアおよびソフトウェアの進歩はいちじるしく，それにともなって GUI 研究にも高度な技術が導入されるようになってきた．第 6 章の視線計測器を用いたポインティング支援手法がよい事例である．近年では MAGIC [Zhai, 1999]をメガネ型のインタフェースに搭載したり [Jalaliniya, 2015]，視線計測器をノート PC に標準搭載する [MSI, 2015]までに小型化・低コスト化が進んでいる．またモーションキャプチャを利用した GUI 研究も盛んに行われており，マウスにトラッキングマーカーを付けて入力の遅延を計測する実験が行われたり [Teather, 2009]，タブレット端末の外部まで指を移動させてスクロール操作を延長す

る手法 [Takashima, 2015]なども提案されている．またソフトウェア面では，近年の視線計測器の例を挙げると，ユーザの目の位置を 3 次元的にトラッキングする機能や，画面上における周辺視野の領域を自動的に求める機能が備わっている (Tobii EyeX Controller)．さらに一般的な USB カメラのみを用いて，視角 1.5°の精度で注視点を検出する画像処理アルゴリズムも提案されている[Ferhat, 2014]．また Asano らはカーソルのピーク速度から終点を予測する手法[Asano, 2005]を提案しているが，速度波形をテンプレートマッチングすることで高精度に終点を予測する手法 [Pasqual, 2014]が提案されるなど，コンピュータビジョンなどの他分野で発展した技術を導入する動きも活発である．こういった研究は，GUI や HCI 分野の発展のみでは実現できず，より広い研究領域の進歩に伴って実現されたものである．

これらの研究とは逆に，ハードウェアの発達によって到達運動の支援が「必要になってしまっている」という捉え方も可能である．第 1 章の冒頭で述べた「ディスプレイの高解像度化に伴ってマウス操作の負担が高まっている」というのが一例である．コンピュータからの映像出力が豊かになるぶんだけ，画面へ入力する操作の負担も重くなっており，何らかの支援がなければ腱鞘炎などの健康被害は拡大する一方であろう．また，スマートウォッチに代表される小型ディスプレイの製造が可能になったことで，ユーザがきわめて微細な制御をする必要も出てきた．最近ではスマートウォッチ操作を支援するための極小スタイラス [Xia, 2015]が提案されるなど，ディスプレイ側の進化に合わせたポインティングデバイスが開発されている．今後も GUI 環境はハードウェア・ソフトウェアの両面で進化し続けると思われるが，操作対象が小さくなれば微細な制御が必要になり，広大になれば移動距離が増大してしまう．人間の微細な制御や素早い運動には限界があるため，コンピュータ側で適切に支援することが必要である．そこで本論文では，粗い制御をするだけで到達運動ができるように支援し，ユーザの負担を軽減することを目指して研究を行った．まず操作の負担や改善度合いを定量的に評価するための基礎的な挙動分析とモデル化を行い，また操作支援手法の開発と評価を行った．今後もコンピュータ側の進化に後れをとらないよう，GUI 研究が絶えず発展し続けることを期待したい．

謝辞

本論文は著者が明治大学大学院 理工学研究科 新領域創造専攻 ディジタルコンテンツ系 博士後期課程在籍時に行った研究をまとめたものです．研究を遂行するにあたり，明治大学 総合数理学部 宮下芳明 教授には多大なご指導を賜りました．論文執筆のアドバイスや研究アイディアの議論などの直接的なご指導をいただいたのみならず，研究への向き合い方，研究コミュニティに貢献することの重要性など，多くのことを教えていただきました．心より感謝を申し上げます．

明治大学 総合数理学部 荒川薫 教授には，急なお願いだったにも関わらず本論文の審査を快く引き受けていただきました．明治大学 総合数理学部 福地健太郎 准教授には，博士前期課程の頃から何度も研究について議論していただき，さらに本論文の審査を引き受けていただきました．筑波大学 システム情報工学研究科 志築文太郎 准教授には，ゼミのミーティングに参加させていただいたり，学会で研究の相談にのっていただくなど，たびたびお世話になりました．深く感謝いたします．

宮下研究室の皆様には，評価実験のたびに快く参加していただきました．博士後期課程の後輩である加藤邦拓くん，高橋治輝くんとは，日頃の議論を通して沢山の着想をいただきました．博士後期課程の先輩である中村美惠子 博士（現 東京藝術大学）と中村裕美 博士（現 東京大学）からは，在学中・卒業後にも多くのアドバイスをいただきました．宮下研究室の卒業生の皆様にも大変お世話になりました．深く感謝いたします．

インターンでお世話になりました，後藤真孝 首席研究員をはじめとする産業技術総合研究所 情報技術研究部門 メディアインタラクション研究グループの皆様に感謝を申し上げます．インターンのホストをお願いした栗原一貴 主任研究員（現 津田塾大学）には毎日昼食後に議論していただき，日々の研究活動として話し合うことの大切さを教えていただきました．また発表番号 [5] [10]は栗原氏との共同研究の成果です．深く感謝いたします．

明治大学 総合数理学部 先端メディアサイエンス学会の諸先生方には，所属が異なるにも関わらず私の研究や進路をずっと気にかけていただきました．明治大学大学院 理工学研究科 新領域創造専攻の先生方には，大学院の授業や博士論文の審査で大変お世話になりました．明治大学 中野キャンパス事務室，生田キャンパス 理工学部事務室，駿河台キャンパス 大学院事務室の皆様には，学振の手続きや博士論文提出で大変お世話になりました．事務室の皆様のサポートがあったからこそ，日々の研究活動に邁進でき，博士論文の提出が叶いました．深く感謝いたします．

本研究の一部は JSPS 科研費 15J11634 および 23700155, JST CREST の支援を受けました．

最後に，大学に入る前から現在にいたるまでずっと応援し続けてくれた両親に，心より感謝を申し上げます．博士後期課程への進学を理解していただき，また健康や精神面など多くのところで私を支え続けていただきました．ありがとうございました．

参考文献

[Accot, 1997]
Accot, J. and Zhai, S. Beyond Fitts' law: models for trajectory-based HCI tasks. In *Proc. of CHI 1997*, pp.295–302.

[Accot, 1999]
Accot, J. and Zhai, S. Performance evaluation of input devices in trajectory-based tasks: an application of the steering law. In *Proc. of CHI 1999*, pp.466–472.

[Accot, 2001]
Accot, J. and Zhai, S. Scale effects in steering law tasks. In *Proc. of CHI 2001*, pp.1–8.

[Accot, 2002]
Accot, J. and Zhai, S. More than dotting the i's --- foundations for crossing-based interfaces. In *Proc. of CHI 2002*, pp.73–80.

[Accot, 2003]
Accot, J. and Zhai, S. Refining Fitts' law models for bivariate pointing. In *Proc. of CHI 2003*, pp.193–200.

[Aceituno, 2013]
Aceituno, J., Casiez, G., and Roussel, N. How low can you go? human limits in small unidirectional mouse movements. In *Proc. of CHI 2013*, pp.1383–1386.

[Ahlström, 2005]
Ahlström, D. Modeling and improving selection in cascading pull-down menus using Fitts' law, the steering law and force fields. In *Proc. of CHI 2005*, pp.61–70.

[Appert, 2008]
Appert, C., Chapuis, O., and Beaudouin-Lafon, M. Evaluation of Pointing Performance on Screen Edges. In *Proc. of AVI 2008*, pp.119–126.

[Asano, 2005]
Asano, T., Sharlin, E., Kitamura, Y., Takashima, K., and Kishino, F. Predictive interaction using the delphian desktop. In *Proc. of UIST 2005*, pp.133–141.

[Tognazzini, 1999]
Tognazzini, B. AskTog. : A quiz designed to give you Fitts.
http://www.asktog.com/columns/022DesignedToGiveFitts.html

[Bateman, 2011]
Bateman, S., Doucette, A., Xiao, R., Gutwin, C., Mandryk, R. L., and Cockburn, A. Effects of view, input device, and track width on video game driving. In *Proc. of GI 2011*, pp.207–214.

[Baudisch, 2003]
Baudisch, P., Cutrell, E., Robbins, D., Czerwinski, M., Tandler, P., Bederson, B. and Zierlinger, A. Drag-and-pop and drag-and-pick: techniques for accessing remote screen content on touch- and pen-operated systems. In *Proc. of INTERACT 2003*, pp.57–64.

[Beaudouin-Lafon, 2001]
Beaudouin-Lafon, M. Novel interaction techniques for overlapping windows. In *Proc. of UIST 2001*, pp.153–154.

[Boritz, 1991]
Boritz, J., Booth, K. S., and Cowan, W. B. Fitts' law studies of directional mouse movement. In *Proc. GI 1991*, pp.216–223.

[Bi, 2013]
Bi, X., Li, Y., and Zhai, S. FFitts law: modeling finger touch with Fitts' law. In *Proc. of CHI 2013*, pp.1363–1372.

[Blanch, 2004]
Blanch, R., Guiard, Y. and Beaudouin-Lafon, M. Semantic pointing: improving target acquisition with control-display ratio adaptation. In *Proc. of CHI 2004*, pp.519–526.

[Blanch, 2009]
Blanch, R. and Ortega, M. Rake cursor: improving pointing performance with concurrent input channels. In *Proc. of CHI 2009*, pp.1415–1418.

[Brenner, 2007]
Brenner, E. and Smeets J. B. J. Flexibility in intercepting moving objects. *Journal of Vision*, Vol.7, No.5, pp.1–17.

[Bützler, 2015]
Bützler, J. and Schlick, C. M. Investigation of the angle and age effect using Fitts’ Law for mouse

input. In *Proc. of IEA 2015*. （ページ番号不明）

[Casiez, 2004]
Casiez, G., Plénacoste, P., and Chaillou, C. Does DOF separation on elastic devices improve user 3D steering task performance? In *Proc. of APCHI 2004*, pp.70–80.

[Casiez, 2011]
Casiez, G. and Roussel, N. No more bricolage!: methods and tools to characterize, replicate and compare pointing transfer functions. In *Proc. of UIST 2011*, pp.603–614.

[Chapuis, 2009]
Chapuis, O., Labrune, J. -B. and Pietriga, E. Dynaspot: speed-dependent area cursor. In *Proc. of CHI 2009*, pp.1391–1400.

[Dennerlein, 2000]
Dennerlein, J. T., Martin, D. B., and Hasser, C. Force-feedback improves performance for steering and combined steering-targeting tasks. In *Proc. of CHI 2000*, pp.423–429.

[Dragicevic, 2004]
Dragicevic, P. Combining crossing-based and paper-based interaction paradigms for dragging and dropping between overlapping windows. In *Proc. of UIST 2004*, pp.193–196.

[Drewes, 2009]
Drewes, H. and Schmidt, A. The MAGIC touch: combining MAGIC-pointing with a touch-sensitive mouse. In *Proc. of INTERACT 2009*, pp.415–428.

[Fagarasanu, 2003]
Fagarasanu, M. and Kumar, S. Carpal tunnel syndrome due to keyboarding and mouse tasks: a review. *International Journal of Industrial Ergonomics*, Vol.31, No.2, pp.119–136.

[Fares, 2012]
Fares, R., Downing, D., and Komogortsev, O. Magic-sense: dynamic cursor sensitivity-based magic pointing. In *Ext. Abst. of CHI 2012*, pp.2489–2494.

[Farris, 2002-a]
Farris, J. S., Jones, K. S., and Anders, B. A. Using impenetrable borders in a graphical web browser: are all angles equal? In *Proc. of HFES 2002*, pp.1251–1255.

[Farris, 2002-b]
Farris, J. S., Jones, K. S., and Anders, B. A. Using impenetrable borders in a graphical web browser:

how does distance influence target selection speed? In *Proc. of HFES 2002*, pp.1300–1304.

[Farris, 2003]
Farris, J. S., Jones, K. S., and Anders, B. A. Selection of web browser controls with and without impenetrable borders: does width make a difference? In *Proc. of HFES 2003*, pp.1380–1384.

[Fares, 2013]
Fares, R., Fang, S., and Komogortsev, O. Can we beat the mouse with MAGIC? In *Proc. of CHI 2013*, pp.1387–1390.

[Ferhat, 2014]
Ferhat, O., Vilarino, F., and Sanchez, F. J. A cheap portable eye-tracker solution for common setups. *Journal of Eye Movement Research*, Vol.7, No.3, pp.2:1–2:10.

[Fitts, 1954]
Fitts, P. M. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, Vol.47, No.6, pp.381–391.

[Forlines, 2008]
Forlines, C. and Balakrishnan, R. Evaluating tactile feedback and direct vs. indirect stylus input in pointing and crossing selection tasks. In *Proc. of CHI 2008*, pp.1563–1572.

[Grossman, 2004]
Grossman, T. and Balakrishnan, R. Pointing at trivariate targets in 3D environments. In *Proc. of CHI 2004*, pp.447–454.

[Grossman, 2005]
Grossman, T. and Balakrishnan, R. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proc. of CHI 2005*, pp.281–290.

[Guiard, 2004]
Guiard, Y., Blanch, R., and Beaudouin-Lafon, M. Object pointing: a complement to bitmap pointing in GUIs. In *Proc. of GI 2004*, 9–16.

[Gunn, 2009]
Gunn, T. J., Irani, P., and Anderson, J. An evaluation of techniques for selecting moving targets. In *Ext. Abst. of CHI 2009*, pp.3329–3334.

[Gutwin, 2003]
Gutwin, C. and Skopik, A. Fisheyes are good for large steering tasks. In *Proc. of CHI 2003*,

pp.201–208.

[Hajri, 2011]
Hajri, A. A., Fels, S., Miller, G., and Ilich, M. Moving target selection in 2D graphical user interfaces. In *Proc. of INTERACT 2011*, pp.141–161.

[Hasan, 2011]
Hasan, K., Grossman, T., and Irani, P. Comet and target ghost: techniques for selecting moving targets. In *Proc. of CHI 2011*, pp.839–848.

[Hild, 2014]
Hild, H., Gill, D., and Beyerer, J. Comparing mouse and MAGIC pointing for moving target acquisition. In *Proc. of ETRA 2014*, pp.131–134.

[Hoffmann, 1991]
Hoffmann, E. R. Capture of moving targets: a modification of Fitts' law. *Ergonomics*, Vol.34, No.2, pp.211–220.

[Huot, 2011]
Huot, S., Chapuis, O., and Dragicevic, P. TorusDesktop: pointing via the backdoor is sometimes shorter. In *Proc. of CHI 2011*, 829–838.

[Jacob, 1991]
Jacob, R.J.K. The use of eye movements in human-computer interaction techniques: what you look at is what you get. *ACM Transactions on Information Systems*, Vol.9, No.3, pp.152–169.

[Jalaliniya, 2015]
Jalaliniya, S., Mardanbegi, D., and Pederson, T. MAGIC pointing for eyewear computers. In *Proc. of ISWC 2015*, pp.155–158.

[Kabbash, 1995]
Kabbash, P. and Buxton, W. A. S. The “prince” technique: Fitts' law and selection using area cursors. In *Proc. of CHI 1995*, pp.273–279.

[Kattinakere, 2007]
Kattinakere, R. S., Grossman, T., and Subramanian, S. Modeling steering within above-the-surface interaction layers. In *Proc. of CHI 2007*, pp.317–326.

[Kobayashi, 2003]
Kobayashi, M. and Igarashi, T. Considering the direction of cursor movement for efficient traversal

of cascading menus. In *Proc. of UIST 2003*, pp.91–94.

[Kobayashi, 2008]
Kobayashi, M. and Igarashi, T. Ninja cursors: using multiple cursors to assist target acquisition on large screens. In *Proc. of CHI 2008*, pp.949–958.

[Kulikov, 2005]
Kulikov, S., MacKenzie, I. S., and Stuerzlinger, W. Measuring the effective parameters of steering motions. In *Ext. Abst. of CHI 2005*, pp.1569–1572.

[Kulikov, 2006]
Kulikov, S. and Stuerzlinger, W. Targeted steering motions. In *Ext. Abst. of CHI 2006*, pp.983–988.

[Kuno, 1998]
久野悦章，八木透，藤井一幸，古賀一男，内川嘉樹．EOG を用いた視線入力インタフェースの開発．情報処理学会論文誌，Vol.39，No.5，pp.1455–1462.

[Kurihara, 2009]
栗原一貴. マイクロスリップの Human Computer Interaction 研究への応用. 生態心理学研究, Vol.4，No.1，pp.7–13.

[Kuwabara, 2011]
桑原智大，山本景子，倉本到，辻野嘉宏，水口充．ゴーストハンティング: 疑似オブジェクト提示によるオブジェクト選択最適化手法．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，Vol.2011，No.12，pp.1–8.

[Liu, 2011]
Liu, L., Martensb, J.-B., and Liere, R. v. Revisiting path steering for 3D manipulation tasks. *International Journal of Human-Computer Studies*, Vol.69, Issue 3, pp.170–181.

[Luo, 2014]
Luo, Y. and Vogel, D. Crossing-based selection with direct touch input. In *Proc. of CHI 2014*, pp.2627–2636.

[Lutteroth, 2015]
Lutteroth, C., Penkar, M., and Weber, G. Gaze vs. mouse: a fast and accurate gaze-only click alternative. In *Proc. of UIST 2015*, pp.385–394.

[MacKenzie, 1991]
MacKenzie, I. S., Sellen, A., and Buxton, W. A. S. A comparison of input devices in element

pointing and dragging tasks. In *Proc. of CHI 1991*, pp.161–166.

[MacKenzie, 1992-a]
MacKenzie, I. S. Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, Vol.7, No.1, pp.91–139.

[MacKenzie, 1992-b]
MacKenzie, I. S. and Buxton, W. Extending Fitts' law to two-dimensional tasks. In *Proc. of CHI 1992*, pp.219–226.

[MacKenzie, 2001]
MacKenzie, I. S., Kauppinen, T., and Silfverberg, M. Accuracy measures for evaluating computer pointing devices. In *Proc. of CHI 2001*, pp.9–16.

[MacKenzie, 2011]
Mackenzie, I. S. and Ashtiani, B. BlinkWrite: efficient text entry using eye blinks. *Universal Access in the Information Society*, Vol.10, 69–80.

[McGuffin, 2002]
McGuffin, M. and Balakrishnan, R. Acquisition of expanding targets. In *Proc. of CHI 2002*, pp.57–64.

[MSI, 2015]
MSI，世界初のアイトラッキングセンサー内蔵ゲーミングノート．（PC Watch サイト内）
http://pc.watch.impress.co.jp/docs/news/event/20150602_705078.html

[Naito, 2004]
内藤悟史，北村喜文，岸野文郎．2 次元直接指示環境におけるステアリングの法則に関する検討，電子情報通信学会論文誌，Vol.J87-D-II，No.9，pp.1834–1841.

[Naito, 2006]
内藤悟史，北村喜文，岸野文郎．間接指示環境のステアリングの法則におけるポインタの大きさと軌道幅に関する検討，ヒューマンインタフェース学会論文誌，Vol.8，pp.143–150.

[Nishida, 2007]
Nishida, T. and Igarashi, T. Drag-and-guess: drag-and-drop with prediction". In *Proc. of INTERACT 2007*, pp.461–474.

[NTT, 2004]

NTT 技術ジャーナル．固視微動とは何ですか？　Vol.16，No.10.

[Ohno, 1997]
大野健彦．視線を用いた高速なメニュー選択作業．情報処理学会論文誌，Vol.40，No.2，pp.602–612.

[Ohno, 2002]
大野健彦. 視線から何がわかるか －視線測定に基づく高次認知処理の解明. 認知科学, Vol.9, No.4，pp.565–579.

[Ortega, 2013]
Ortega, M. Hook: heuristics for selecting 3D moving objects in dense target environments. In *Proc. of 3DUI 2013*, pp.119–122.

[Pasqual, 2014]
Pasqual, P. T. and Wobbrock, J. O. Mouse pointing endpoint prediction using kinematic template matching. In *Proc. of CHI 2014*, pp.743–7526.

[Pastel, 2006]
Pastel, R. Measuring the difficulty of steering through corners. In *Proc. of CHI 2006*, pp.1087–1096.

[Pfeuffer, 2014]
Pfeuffer, K., Alexander, J., Chong, M. K., and Gellersen, H. Gaze-touch: combining gaze with multi-touch for interaction on the same surface. In *Proc. of UIST 2014*, pp.509–518.

[Pfeuffer, 2015]
Pfeuffer, K., Alexander, J., Chong, M. K., Zhang, Y., and Gellersen, H. Gaze-shifting: direct-indirect input with pen and touch modulated by gaze. In *Proc. of UIST 2015*, pp.373–383.

[Quinn, 2013]
Quinn, P., Cockburn, A., and Delamarche, J. Examining the costs of multiple trajectory pointing techniques. *International Journal of Human-Computer Studies*, Vol.71, No.4, pp.492–509.

[Ruiz, 2003]
Ruiz, J., Tausky, D., Bunt, A., Lank, E., and Mann, R. Analyzing the kinematics of bivariate pointing. In *Proc. of GI 2008*, pp.251–258.

[Raskin, 2003]
Raskin, J. *The Humane Interface: New Directions for Designing Interactive Systems*, Addison-Wesley Professional, 2000.

[Räihä, 2009]
Räihä, K.-J. and Špakov, O. Disambiguating ninja cursors with eye gaze. In *Proc. of CHI 2009*, pp.1411–1414.

[Sentry, 2015]
Sentry gaming eye tracker.
https://steelseries.com/gaming-controllers/sentry-gaming-eye-tracker

[Shigemori, 2006]
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.

[Shigemori, 2007]
重森晴樹，入江健一，倉本到，渋谷雄，辻野嘉宏．GUI 環境でのバブルカーソルの実用的評価．情報処理学会論文誌，Vol.48，No.12，pp.4076–4079.

[Sibert, 2000]
Sibert, L. E. and Jacob, R. J. K. Evaluation of eye gaze interaction. In *Proc. of CHI 2000*, pp.281–288.

[Soumu, 2015]
総務省．平成 26 年通信利用動向調査の結果．2015 年 7 月 27 日発表．
http://www.soumu.go.jp/johotsusintokei/statistics/data/150717_1.pdf

[Su, 2014]
Su, X., Au, O. K. -C., and Lau, R. W. H. The implicit fan cursor: a velocity dependent area cursor. In *Proc. of CHI 2014*, pp.753–762.

[Sun, 2010]
Sun, M., Ren, X., and Cao, X. Effects of multimodal error feedback on human performance in steering tasks. *Journal of Information Processing*, Vol.18, pp.2375–2383.

[Sun, 2012]
Sun, M., Ren, X., Zhai, S., and Mukai, T. An investigation of the relationship between texture and human performance in steering tasks. In *Proc. of APCHI 2012*, pp.1–6.

[Takashima, 2015]
Takashima, K., Shinshi, N., and Kitamura, Y. exploring boundless scroll by extending motor space. In *Proc. of MobileHCI 2015*, pp.557–566.

[Teather, 2009]
Teather, R. J., Pavlovych, A., Stuerzlinger, W., and MacKenzie, I. S. Effects of tracking technology, latency, and spatial jitter on object movement. In *Proc. of 3DUI 2009*, pp.43–50.

[Tresilian, 2002]
Tresilian, J. R. and Lonergan, A. Intercepting a moving target: effects of temporal precision constraints and movement amplitude. *Experimental Brain Research*, Vol.142, No.2, pp.193–207.

[Tsukitani, 2011]
築谷喬之，高嶋和毅，朝日元生，伊藤雄一，北村喜文，岸野文郎．Birdlime Icon: 動的にターゲットを変形するポインティング支援手法．コンピュータソフトウェア，Vol.28，No.2，pp.140–152.

[Velloso, 2015]
Velloso, E., Fleming, A., Alexander, J., and Gellersen, H. Gaze-supported gaming: MAGIC techniques for first person shooters. In *Proc. of CHI PLAY 2015*, pp.343–347.

[Vidal, 2013]
Vidal, M., Bulling, A., and Gellersen, H. Pursuits: spontaneous interaction with displays based on smooth pursuit eye movement and moving targets. In *Proc. of UbiComp 2013*, pp.439–448.

[Voelker, 2015]
Voelker, S., Matviienko, A., Schöning, J., and Borchers, J. Combining direct and indirect touch input for interactive workspaces using gaze input. In *Proc. of SUI 2015*, pp.79–88.

[Vogel, 2012]
Vogel, D. and Casiez, G. Hand occlusion on a multi-touch tabletop. In *Proc. of CHI 2012*, pp.2307–2316.

[Walker, 2003]
Walker, N., Meyer, D. E. and Smelcer, J. B. Spatial and temporal characteristics of rapid cursor-positioning movements with electromechanical mice in human-computer interaction. *Human Factors*, Vo.35, No.3, pp.431–458, 1993.

[Wobbrock, 2007]
Wobbrock, J. O. and Gajos, K. Z. A comparison of area pointing and goal crossing for people with and without motor impairments. In *Proc. of ASSETS 2007*, pp.3–10.

[Wobbrock, 2008-a]

Wobbrock, J. O. and Gajos, K. Z. Goal crossing with mice and trackballs for people with motor impairments: performance, submovements, and design directions. *ACM Transactions on Accessible Computing*, Vol.1, No.1, pp.4:1–4:37.

[Wobbrock, 2008-b]
Wobbrock, J. O., Cutrell, E., Harada, S., and MacKenzie, I. S. An error model for pointing based on Fitts' law. In *Proc. of CHI 2008*, pp.1613–1622.

[Worden, 1997]
Worden, A., Walker, N., Bharat, K. and Hudson, S. Making computers easier for older adults to use: area cursors and sticky icons. In *Proc. of CHI 1997*, pp.266–271.

[Xia, 2015]
Xia, H., Grossman, T., and Fitzmaurice, G. NanoStylus: enhancing input on ultra-small displays with a finger-mounted stylus. In *Proc. of UIST 2015*, pp.447–456.

[Yamato, 2001]
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.

[You, 2014]
You, C.-W., Hsieh, Y.-H., Cheng, W.-H., and Hsieh, Y.-H. AttachedShock: design of a crossing-based target selection technique on augmented reality devices and its implications. *International Journal of Human-Computer Studies*, Vol.72, No.7, pp.606–626.

[Yu, 2010]
Yu, C., Shi, Y., and Balakrishnan, R. The satellite cursor: achieving MAGIC pointing without gaze tracking using multiple cursors. In *Proc. of UIST 2010*, 163–172.

[Zhai, 1999]
Zhai, S., Morimoto, C., and Ihde, S. Manual and gaze input cascaded (MAGIC) pointing. In *Proc. of CHI 1999*, pp.246–253.

[Zhai, 2003]
Zhai, S., Conversy, S., Beaudouin-Lafon, M. and Guiard, Y. Human on-line response to target expansion. In *Proc. of CHI 2003*, pp.177–184.

[Zhai, 2004]
Zhai, S., Accot, J., and Woltjer. R. Human action laws in electronic virtual worlds --- an empirical

study of path steering performance in VR. *Presence*, pp.113–127.

[Zhou, 2009]
Zhou, X., Cao, X., and Ren, X. Speed-accuracy tradeoff in trajectory-based tasks with temporal constraint. In *Proc. of INTERACT 2009*, pp.906–919.

URL は全て 2015 年 12 月 20 日参照.

本研究に関する発表

学術雑誌論文（5 件）

[1] 山中祥太，宮下芳明．無限大のサイズをもつターゲットのポインティングに関する調査．情報処理学会論文誌．Vol.57，No.4，2016.（採録決定）

[2] 山中祥太，宮下芳明．幅の変化する経路を逆向きにステアリングする時間の予測方法の提案．情報処理学会論文誌，Vol.57，No.2，2016.（採録決定）

[3] 山中祥太，宮下芳明．細長いターゲットのドラッグ開始を支援する手法とその評価．コンピュータソフトウェア，Vol.33，No.1，2016.（採録決定）

[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とその評価．ヒューマンインタフェース学会論文誌, Vol.15, No.3, pp.313-326, 2013.（第 15 回 HI 学会論文賞）

[5] 山中祥太，栗原一貴，宮下芳明．注視していないことを利用したポインティング高速化手法とその評価．コンピュータソフトウェア，Vol.30，No.3，pp.53-63，2013.（WISS 2012 推薦論文）

査読付き国際会議（3 件）

口頭発表

[6] Shota Yamanaka and Homei Miyashita. Modeling the steering time difference between narrowing and widening tunnels. In *Proc. of CHI 2016*, 2016. (to appear)

[7] Shota Yamanaka and Homei Miyashita. Switchback cursor: mouse cursor operation for overlapped windowing. In *Proc. of INTERACT 2013*, pp.746-753, 2013.

デモ・ポスター発表

[8] Shota Yamanaka and Homei Miyashita. The nudging technique: input method without fine-grained pointing by pushing a segment. In *Adjunct Proc. of UIST 2013*, pp.3-4, 2013.

査読付き国内会議（4 件）

口頭発表

[9] 山中祥太, 宮下芳明. Cross-drag: 細長いターゲットのドラッグを容易にする操作手法. WISS2014 第 22 回インタラクティブシステムとソフトウェアに関するワークショップ論文集，pp.97-102, 2014.

[10] 山中祥太，栗原一貴，宮下芳明．注視していないことを利用したマウスカーソル高速化手法．WISS2012 第 20 回インタラクティブシステムとソフトウェアに関するワークショップ論文集，pp.127-132，2012.

[11] 山中祥太，宮下芳明．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．WISS2011 第 19 回インタラクティブシステムとソフトウェアに関するワークショップ論文集，pp.66-71，2011.

デモ・ポスター発表

[12] 山中祥太，宮下芳明．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．WISS2011 第 19 回インタラクティブシステムとソフトウェアに関するワークショップ論文集，pp.66-71，2011.（発表番号[11]に関連したデモ発表）

査読なし研究会

[13] 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，Vol.2011，No.13，pp.1-8, 2011.（HCI 研究会学生奨励賞）