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

山中 祥太　栗原 一貴　宮下 芳明

視線計測技術の精度は年々向上しているものの，ユーザが操作したいと考えて注視している画面内のオブジェクトを視線情報のみから特定することは困難であると言われている．一方で，注視していない範囲は容易に特定できるため，この領域を適切に利用することで新たなインタラクションが可能になると考えた．本論文では，注視していない領域においてカーソルの移動速度を上昇させ，ポインティングを高速化する手法を提案する．評価実験では，PC 操作において日常的に行われるアイコンクリックを想定したタスクを行い，選択時間とエラー率を計測した．実験に用いたカーソル速度は，OS の標準速度，OS の最高速度，提案手法の 3 種類である．実験の結果，提案手法は標準速度より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．

In a general GUI environment, it is difficult to detect an object that a user wants to look at. However, detecting objects within the non-gaze area is easy, and using this region enables us to perform a new interaction. In this paper, we propose a fast target-pointing technique using the non-gaze area. The cursor speeds up in the non-gaze area for fast pointing and slows down to normal speed near the gaze point for fine-grained pointing. In a square-clicking experiment, we compared the selection time and the error rate at three cursor speed settings: the default and the maximum speed settings of the OS, and the speed setting in our technique. Our technique reduces selection time while maintaining about the same error rate compared to default speed, thereby illustrating the effectiveness of our proposed method.

1　はじめに

マウスカーソルは現在 PC で広く利用されているポインティングインタフェースであり，Fitts' Law [5] に代表されるポインティング速度や精度に関する研究は今もなお盛んである．それらの研究のうち，ソフトウェアのみによる操作支援は以前から多くなされているが（[6] など），近年では視線計測技術を利用したポインティング支援手法も開発されている（[19] など）．

従来から肢体不自由者を対象として，ポインティング操作を視線のみで行うシステムが開発されてきた．その一方で，マウスなどのポインティングデバイスを利用可能なユーザを対象に，視線情報を利用することでポインティング時間の短縮を図る手法も提案されている．ただし，人間の目は固視微動（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）をしているため，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（選択したいアイコンやボタンなどのオブジェクト）を視線情報のみから特定するのは困難である [19].

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容易に判定できる．視線計測器の最

Interaction Technique Using Non-Gaze Area for Fast Target Pointing and Its Evaluation.

Shota Yamanaka, 明治大学大学院理工学研究科新領域創造専攻ディジタルコンテンツ系, Program in Digital Contents Studies, Programs in Frontier Science and Innovation, Graduate School of Science and Technology, Meiji University.

Kazutaka Kurihara, 独立行政法人産業技術総合研究所, National Institute of Advanced Industrial Science and Technology.

Homei Miyashita, 明治大学大学院理工学研究科新領域創造専攻ディジタルコンテンツ系，独立行政法人科学技術振興機構 CREST, Program in Digital Contents Studies, Programs in Frontier Science and Innovation, Graduate School of Science and Technology, Meiji University, JST, CREST.

コンピュータソフトウェア, Vol.29, No.1 (2012), pp.78–84.

[研究論文] 2011 年 12 月 15 日受付.

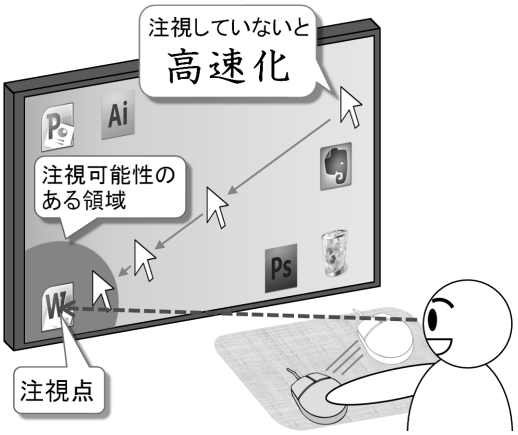

図 1 提案手法の概要．注視可能性のある領域の外側は，「確実に注視していない領域」であり，そこにカーソルが含まれていれば高速化する．

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．

そこで本論文では，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，図 1 のようにマウス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．これにより，カーソルを長距離移動させる際の操作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．

以下では，2 章で視線計測技術の概要と測定精度の限界について述べ，3 章で提案手法を説明する．次に 4 章で先行研究と提案手法の関係性について述べ，5 章で具体的なシステムの実装を説明する．その後に 6 章で評価実験について報告し，最後に 7 章で本研究のまとめと今後の課題を述べる．

2 視線計測技術の限界

本章では，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 環境においてはユーザが注視したいオブジェクトを特定することが困難である理由を説明する．

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ハードルが高いといえよう．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[12] が詳しい）．

本論文の評価実験では，視線計測器に Tobii Technology 社製 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 60Hz で動作し，計測精度（誤差の最大値）は視角 0.5 度である．次節ではこのデータを基にディスプレイ上での注視点の誤差を算出する．

2.2 視線計測器の計測誤差

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．これを本論文の評価実験環境 (目からディスプレイまでの距離 65cm，ディスプレイサイズ 27 インチ，ディスプレイ解像度 1920 × 1200 ピクセル) に当てはめると，計算誤差は約 18

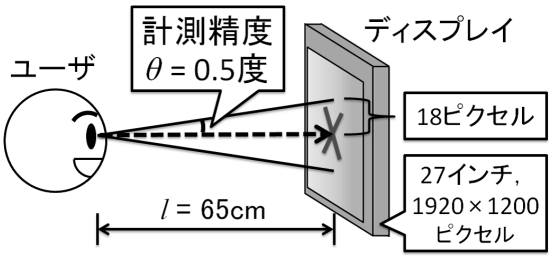

図 2 計測精度によるディスプレイ上の誤差範囲．×印の交点がユーザが見たい位置．

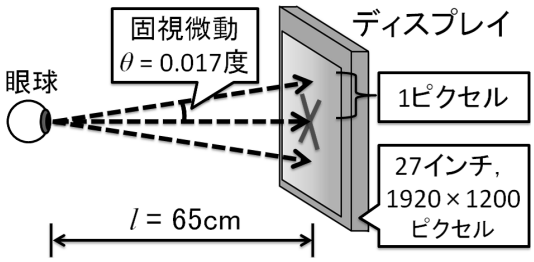

図 3　固視微動によるディスプレイ上の誤差範囲．×印の交点がユーザが見たい位置．

ピクセルになる（図 2）．また，固視微動は最大で瞬間角速度 1 度に達するランダムな運動であることがわかっている [21]．これは Tobii X60 で検出できる 1/60 秒あたりに換算すると最大約 0.017 度であり，上記の実験条件では約 1 ピクセルになる（図 3）．したがって，視線計測器によって得られる情報は，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．

2.3　注視したいオブジェクトを特定することの困難さ

前節の半径 19 ピクセルという値は 1/60 秒あたりの計測誤差であるが，ユーザが特定のオブジェクトを注視していることを判定するには，注視点が一定時間オブジェクト上に停留していることを基準にするため，合計の誤差はより大きくなる．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 ピクセルというのは十分な精度とはいえないと考えられる．図 4 は Windows7 の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の中心を注視しているつもりでも，システムからはその右

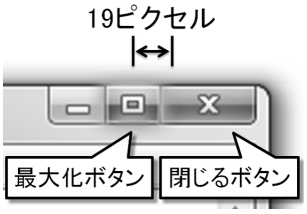

図 4　Windows7 のウィンドウの拡大図

側にある閉じるボタンを注視していると判定される危険がある．そのため，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」に類するシステムでは，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 [11] では横 100 × 縦 50 ピクセルのメニューが 40 ピクセル間隔で配置されており，計測誤差を吸収できるように設定されている（[11] の実験環境では 50 ピクセルが視角 1.4 度に相当する）．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オブジェク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オブジェクトを特定するのは困難である．

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第 2 章で述べた．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．第 2 章 2 節で求めた合計誤差の 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．

本論文では，この注視外領域においてカーソル移動を高速化させる手法を提案する．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ことが知られている [20]．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詳細なポインティングを可能にする．

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 [19] [20] も存在するが，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．たとえばカーソルが画面端にあるとき，それ以上外側に向かってマウスを動かしてもカーソルは移動しないため，画面端のオブジェクトは実質的に無限の大きさを持っている

といえる．ゆえに，最大化されたウィンドウの右上にある「閉じるボタン」や，WindowsOS の左下にある「スタートボタン」，画面右端のスクロールバーなどといったオブジェクトは容易にポインティング可能であり，特に注視せずとも操作が可能である．文献 [19] [20] の手法では，これらを注視外領域で操作したいときにも注視点付近にカーソルがジャンプして操作が妨げられたり，この問題を回避するために注視点付近へジャンプさせる機能の ON/OFF を必要なタイミングで明示的に切り替える必要がある．一方で本提案手法では，注視外領域でもカーソル移動は問題なく行える．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，画面端のオブジェクトが無限大の大きさを持っている性質上，微細な操作は必要とされないため大きな問題は生じないと考えられる．また，画面端以外の位置で微細に操作したいオブジェクトがあれば，自然と視線がそこへ移動するため，ターゲット付近に差し掛かったカーソルは自然と細かい制御が可能になる，という設計である．

4 関連研究

本論文では視線計測器を用いたカーソル移動の高速化を提案しているが，特殊な機器を用いずにポインティング速度を向上させる手法も多く研究されている．Kobayashi らの Ninja Cursors [9] は，複数のカーソルを表示して同時に移動させ，ターゲットに近いカーソルで選択操作をすることで移動距離を短縮させる手法である．アイコンの領域に侵入可能なカーソルを一度に 1 つに限定し，他のカーソルはアイコン付近で待機させることで，複数のターゲットを同時に選択してしまう問題に対処している．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複数提案されている [4] [13]．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．Asano らの Delphian Desktop [1] は，カーソル移動時のピーク速度からターゲットまでの距離を予測し，カーソルをジャンプさせる手法である．これとは逆に，カーソルの移動方向からターゲットを推定してカーソル付近まで引き寄せる Drag-and-Pop [2] がある．Delphian Desktop と Drag-and-Pop も視線計測技術と併用することでさらなる精度向上が期待できる．すなわち，ターゲットの推定や距離の予測をカーソルの動きから求めるだけでなく，ユーザが見ている位置まで考慮することでより正確な推定ができると考えられる．

視線のみでポインティングを行う手法も提案されている．Sibert らの実験では，従来の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の結果が出ている [14]．2.3 インチ間隔に並んだ直径 1.12 インチの円を選択するタスクにおいて，視線を用いて 0.15 秒間注視する選択手法の方がマウスクリックよりも高速であることが確認されている．しかし実際の GUI 環境ではターゲットが隣接するなど密に並んだ状況も多く，視線のみでの高速な選択は困難な状況も生じうると考えられる．また久野らは，肢体不自由者を対象として，ターゲットを一定時間注視し続けることでポインティングと確定操作を行う手法を提案している [7]．これは視線以外での操作を一切行わない条件下での操作手法であり，マウス操作との併用を考慮してはいない．また，視線のみでのポインティングは，意図的な視線移動とそれ以外との判別が困難である (Midas touch problem と呼ばれる [8]) 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．

視線とマウスを切り替えてポインティングを行う手法に，Zhai らの MAGIC [20] がある．マウスによってカーソルを移動させた時点で，視点から一定距離にカーソルをジャンプさせ，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 [19] 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，長距離の移動は高速な視線で行い，微細な移動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を行うことで，高速化と高精度化の両方を実現している．大和らの手法は，ターゲットの選択面積を拡大することでポインティング時間をさらに短縮している．ただし，たとえばスクロールバーを何度も操作しながらコンテンツをブラウジングする場面などでは，

マウスを動かす度にカーソルが注視点にジャンプしてしまい操作が破綻すると考えられる．これを回避するためには，視線情報を利用するか否かを明示的に切り替えることが必要になり，日常的な作業時の負担は増大してしまうと考えられる．これに対し本論文の提案手法では，注視点から離れた位置ではカーソルが OS の設定可能範囲内で高速化するのみであり，操作が破綻するには至らないと考えられる．

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，Grossman らの Bubble Cursor [6] がある．ポインティングを一点ではなく円形にすることで，カーソルの移動距離を短縮している．ただし，ターゲットが密に配置されると通常のカーソルよりも操作時間が増大してしまうことが判明している [10] ことから，実用上の問題は残されている．また，アイコンやボタンなどの特定の形状・サイズを持ったターゲットのポインティングを対象としており，長文から任意の文字列を選択するといった操作には適用が困難である．

ターゲットから離れているほどカーソルを高速化するシステムに Blanch らの Semantic Pointing [3] がある．また，カーソルがターゲットに重なっているときには低速化することで相対的にターゲットを大きくする Sticky Icons [18] がある．Sticky Icons は実際にカーソル速度を低減させる手法だが，築谷らの Birdlime Icon [17] はこれと同様の効果をアイコンの形状変化のみで実現している．本論文の提案手法は Semantic Pointing におけるターゲットの特定を視線計測によって行ったものといえる．ただし，これら 3 つの手法も Bubble Cursor と同様に適用可能なターゲットが限定的である．

ポインティング以外の GUI 操作に視線を活用する例として，高木はドローツール使用時に図形の配置を自動で揃えるシステムを開発している [15]．図形の揃えたい辺上のうち，目標図形に近い頂点付近を他の領域より有意に長時間注視する性質から，ユーザの操作意図を推定して自動化するものである．また同じく高木は，日本語対訳のついた英文を読む際の視線の動きから，ユーザが英文を滑らかに読んでいるのか，あるいは英単語を探索しているのかなどの意図を判定可能であるとしている [16]．さらに，ユーザの英語力まで推定可能であるとしており，語彙力や文章の読み込み状況に応じた翻訳支援が可能であるとしている．

5 実装

視線計測には Tobii Technology 社製 Tobii X60 Eye Tracker（計測精度 0.5 度）を使用し，60Hz で注視点を計測する．カーソルの速度は Hot Soup Processor 3.3 及び Win32API により，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」の設定値をマウス操作中に動的に変更することで実現する．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．

$$speed = l_{eye\_cursor}/(l_{max}/stages) + speed_{min} \quad (1)$$

ここで 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），$stages$ 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．本論文の評価実験では $speed_{min}$ 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 $stages$ は 11 となる．なお，ここでの距離の単位はピクセルで計算されるため，小数点以下の値は最終的に切り捨てられる．本論文の評価実験で用いた 1920 × 1200 ピクセルのディスプレイの

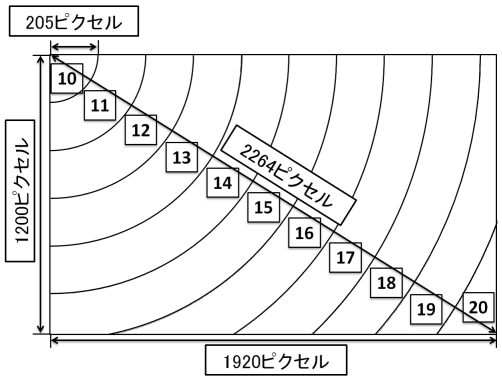

図 5 注視点が画面左上隅にある場合の，カーソル位置に応じた速度のマップ（10〜20 の範囲で変化）

場合，例として注視点が画面左上の座標 (0, 0) にあるときには，カーソルの位置によって図 5 のように速度が決定する．式 (1) における $(l_{max}/stages)$ の値は，図 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．

6 評価実験

本章では，タイル状に配置された矩形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ことから，既存研究で伝統的に行われてきたポインティングタスクである．以下に詳細を記す．

6.1 タスク

図 6 のように隣接した灰色の矩形の中から，1 つだけ赤色になったターゲットをクリックしていく．矩形の個数は，1920 × 1200 ピクセルのディスプレイを使用して 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．ただし，第 2 章 2 節に述べたように，画面端に置かれた矩形が無限の大きさを持ってしまうことを防ぐため，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 29mm）．

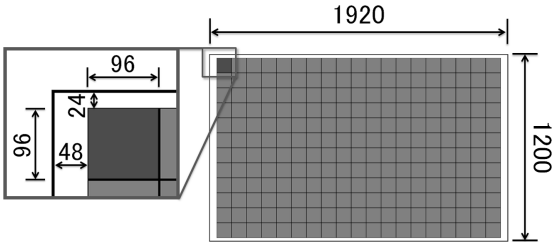

図 6 タスク開始時の画面と拡大図（数値の単位はピクセル）．ターゲットのみ赤色で他は灰色．

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置かれる．被験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 回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．

なお，提案手法では注視点からの距離が長いほどカーソルが高速化されるため，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提案手法の有効性を検証するために，連続する 2 つのターゲットの中心距離が，ディスプレイの横幅を 100%としたとき 10%区切りの範囲内で各 3 回ずつ，1 試行で合計 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設定した．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．

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

ただし，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．

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

Control-Display 比は $10/speed$ で決定される．使用したマウスは BUFFALO 社製 BSMOU05M(1000dpi) であり，すなわち実距離でマウスを 1 インチ動かすと，カーソルは画面内で $1000 \times speed/10$ ピクセル移動する．また，「ポインターの精度を高める [†1]」のチェックを ON にしている．

†1 http://msdn.microsoft.com/ja-jp/library/windows/hardware/gg463319.aspx（2013 年 1 月 8 日閲覧）

6.3 実験方法

ディスプレイは DELL 社製 2707WFP（27 インチ, 1920 × 1200 ピクセル）を用いた．実験前に被験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 65cm，目と視線計測器の距離は約 45cm である．実験時の被験者と機器の配置を図 7 に示す.

被験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エラーをせずに操作するよう伝えた.

被験者には 1 種類のカーソル速度につき 5 回タスクを試行させた．休憩を試行ごとに 10 秒間，カーソル速度変更時に 2 分間設け，被験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．各カーソル速度での 5 回の試行の平均選択時間と平均エラー率を，その速度・ターゲット距離における被験者の記録とする．なお，被験者には各カーソル速度でタスクを試行する前に，カーソルの速度及び提案手法における速度変化のアルゴリズムを伝えた．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被験者によって変更している.

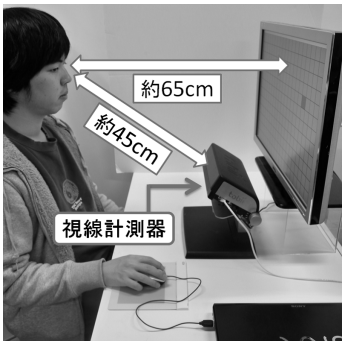

図 7　実験時の被験者と機器の配置

6.4 結果と考察

ターゲットを 1 つクリックするのに要した平均選択時間と平均エラー率を，ターゲットの出現距離ごとに図 8，9 に示す．∗ 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．また，選択時間とエラー率の標準偏差をそれぞれ表 1，2 に示し，さらに被験者ごとの対応ありで分散分析（反復測定）した結果の分散比（F 値）と有意確率を表 3 に示す．表 3 に網掛けを施しているのは，3 種類のカーソル速度間に有意差または有意傾向が見られた箇所である.

図 8 より，いずれのターゲット距離においても，提案手法は速度 10（標準）より早く目的の矩形を選択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選択時間を短縮できていることから，提案手法の有用性を示している．ディスプレイ上における 192 ピクセル（実測約 5.8cm）は，図 2 において視角 5.1 度に相当することから，提案手法は一般的に視角 5.1 度以上離れたターゲットをポインティングする際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が理由であると考えられる．また図 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速度 10（標準）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することが可能であると示された．一方で，速度 20（最速）はターゲット距離によってはエラー率が高くなる傾向が見られ，選択時間も速度 10(標準)より有意に増大することがあることがわかった.

カーソルの制御しやすさに関するアンケート結果を表 4 に示す．これによると，提案手法は速度 10（標準）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

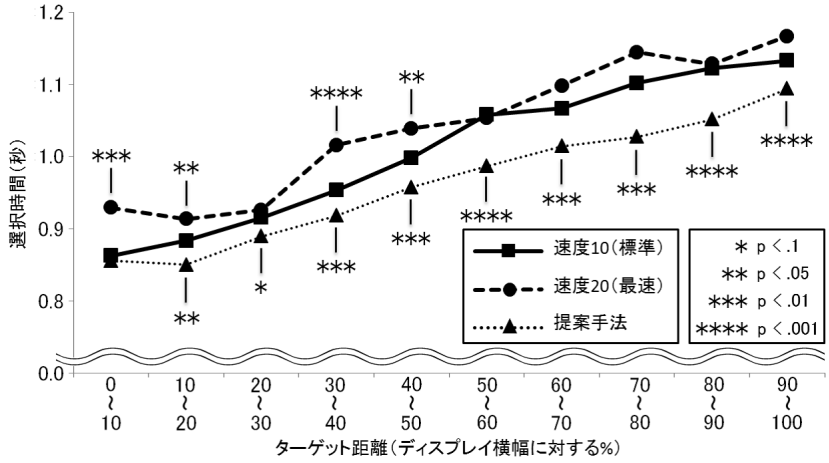

図 8　選択時間．速度 10（標準）と比較して，速度 20（最速）が有意に遅い箇所には ＊ マークをグラフの上側に付し，提案手法が有意に早い箇所には下側に付している．

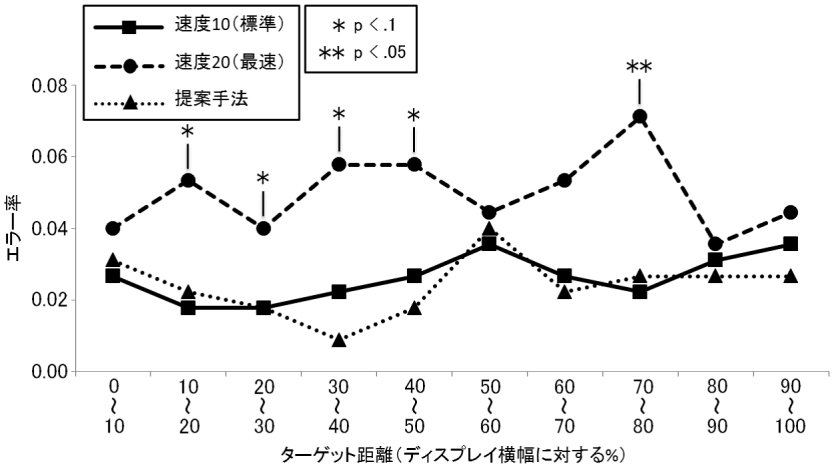

図 9　エラー率．速度 10（標準）と比べて速度 20（最速）が有意に高い箇所には ＊ マークを付している．

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．これらは提案手法の狙いである，特に意識せずとも速度を適切に制御する機能がうまくはたらいた結果であると考えられる．一方で，速度 20（最速）は速度 10（標準）と比べて制御しづらいと感じられていることが表 4 からわかる．図 9 において速度 20（最速）のエラー率が速度 10（標準）より高かったことが示されているが，これが操作しづらいという主観評価につながっていると考えられる．自由記述によるアンケートでも，速度 20（最速）は速すぎて操作しづらいと回答した被験者が 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被験者が 5 名いたことから，被験者が意識するほど操作性が悪化していることがわかる．

提案手法に関するアンケート結果を表 5 に示す．カーソル速度が動的に変化することを意識していた 2 名は，注視点付近でカーソルが標準速度に近づくこと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操作時間短縮にはたらいていると感じている旨を併せて述べている．また，

表 1　ターゲット距離ごとの選択時間の標準偏差（秒）

カーソル速度	0%|10%	10%|20%	20%|30%	30%|40%	40%|50%	50%|60%	60%|70%	70%|80%	80%|90%	90%|100%
速度 10	0.072	0.059	0.063	0.067	0.083	0.087	0.064	0.102	.078	0.159
速度 20	0.088	0.063	0.085	0.104	0.092	0.075	0.107	0.093	0.097	0.073
提案手法	0.069	0.062	0.049	0.060	0.059	0.084	0.074	0.061	0.076	0.074

表 2　ターゲット距離ごとのエラー率の標準偏差

カーソル速度	0%|10%	10%|20%	20%|30%	30%|40%	40%|50%	50%|60%	60%|70%	70%|80%	80%|90%	90%|100%
速度 10	.042	.031	.031	.033	.042	.042	.034	.033	.056	.066
速度 20	.070	.072	.042	.050	.061	.074	.062	.082	.043	.060
提案手法	.056	.033	.031	.023	.040	.070	.041	.055	.061	.042

表 3　ターゲット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．

	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	$p = .656$

表 4　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

タスクの試行中に視線を制御することを意識していた被験者は 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，できるだけカーソルを高速化しようと目の動きを意識した結果であると考えられる．特に意識してターゲットを見つめようとすることは普段の作業時にはないと思われ，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 5　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」と述べた被験者がいた．本章の実験よりも微細な操作が求められる作業，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

作業全体では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思われる．

7 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，ポインティング操作に要する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度）以上のとき，選択時間がOS標準速度の場合より有意に短縮され，かつエラー率は上昇しないことから有効性を示した．

本論文では比較的短時間で完了するタスクにおけるターゲットの選択時間とエラー率について評価したが，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被験者がいたが，これは長時間の作業において負荷が高くなる危険がある一方で，次第に意識しなくなっていくことも考えられる．すなわち，提案手法を利用し続けることで自然に手と目の動きを連携させられるようになり，意識せずともカーソルを高速化するような目の動きになっていく可能性もあると筆者らは考えており，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被験者がいるため，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ことで，ユーザにとって望ましくない加速を回避する機能が挙げられる．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るがゆえに，たとえばウェブブラウジングをするとき，画面の中央を見ながら画面端のスクロールバーを操作すると，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．また，こういったスクロールバー操作を含む一般のドラッグ操作中の速度変更方法には様々なものが考えられるが，これについては現在調査中である．さらに，本論文の実験では提案手法におけるカーソルの最高・最低速度や，図5における速度変化の閾値などを筆者らが設定したが，これをカスタマイズ可能にすることでユーザにとってさらに使いやすくなり，操作時間が短縮できる可能性がある．これらの改良を行ったうえでの評価実験も実施したい．

謝辞 本研究の一部は科研費 (23700155) の助成を受けたものです．

参 考 文 献

[1] Asano, T., Sharlin, E., Kitamura, Y., Takashima, K. and Kishino, F. : Predictive Interaction Using the Delphian Desktop, *Proc. of UIST '05*, pp. 133–141(2005).

[2] 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, *Proc. of Interact '03*, pp. 57–64(2003).

[3] Blanch, R., Guiard, Y. and Beaudouin-Lafon, M. : Semantic Pointing: Improving Target Acquisition with Control-Display Ratio Adaptation, *Proc. of CHI '04*, pp. 519–526(2004).

[4] Blanch, R. and Ortega, M. : Rake Cursor: Improving Pointing Performance with Concurrent Input Channels, *Proc. of CHI '09*, pp. 1415–1418(2009).

[5] 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(1954).

[6] Grossman, T. and Balakrishnan, R. : The Bubble Cursor: Enhancing Target Acquisition by Dynamic Resizing of the Cursor's Activation Area, *Proc. of CHI '05*, pp. 281–290(2005).

[7] 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(1998).

[8] 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(1991).

[9] Kobayashi, M. and Igarashi, T. : Ninja Cursors:

Using Multiple Cursors to Assist Target Acquisition on Large Screens, *Proc. of CHI '08*, pp. 949–958(2008).

[10] 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(2007).

[11] 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(1999).

[12] 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(2002).

[13] Räihä, K.-J. and Špakov, O.：Disambiguating Ninja Cursors with Eye Gaze, *Proc. of CHI '09*, pp. 1411–1414(2009).

[14] Sibert, L.E. and Jacob, R.J.K.：Evaluation of Eye Gaze Interaction, *Proc. of CHI '00*, pp. 281–288(2000).

[15] 高木啓伸：セレクションタスクにおける視線, インタラクティブシステムとソフトウェア IV, 近代科学社, pp. 131–140(1996).

[16] 高木啓伸：視線からのユーザ情報の解析〜翻訳支援システムを題材として〜, インタラクティブシステムとソフトウェア V, 近代科学社, pp. 55–64(1997).

[17] 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(2011).

[18] Worden, A., Walker, N., Bharat, K. and Hudson, S.：Making Computers Easier for Older Adults to Use: Area Cursors and Sticky Icons, *Proc. of CHI '97*, pp. 266–271(1997).

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

[20] Zhai, S., Morimoto, C. and Ihde, S.：Manual And Gaze Input Cascaded (MAGIC) Pointing, *Proc. of CHI '99*, pp. 246–253(1999).

[21] NTT 技術ジャーナル,「固視微動とは何ですか?」, Vol. 16, No. 10, (2004).