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

山中 祥太　　栗原 一貴　　宮下 芳明*

概要.　近年視線計測技術の精度が向上しているものの，ユーザが操作したいと考え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ている．本稿では，注視していない領域においてカーソルの移動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とテキスト選択を想定したタスクを行い，操作時間とエラー率を比較した．実験で比較したのは，OS の標準速度及び最高速度，提案手法，提案手法の理想状態の 4 種類である．実験の結果，アイコンを模した矩形の選択タスクでは提案手法の理想状態において操作時間が標準速度より短縮され，注視外領域を利用する新たなインタラクションの可能性を示した．

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[1] 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．その研究手段としてソフトウェアによる操作支援が以前から多くなされてきたが（[2] 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研究されている（[3] 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多く開発されてきた．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，常に微細に動く現象）をしているため，たとえ視線計測機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 [3]．

そこで本稿では，図 1 のように注視点から離れた位置にあるカーソルほど高速に移動させる操作手法を提案する．注視していない領域の特定は，注視したいオブジェクトの特定よりも比較的容易であるため，この領域を適切に利用することで普段の PC 操作を妨げることなくポインティング速度を向上可能であると考える．

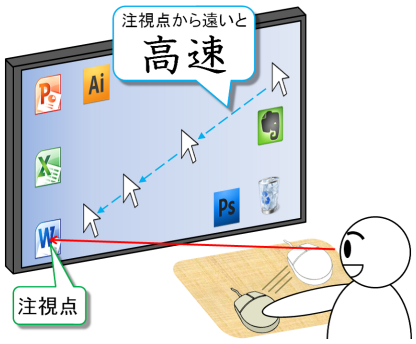

図 1. 注視点から離れた位置のカーソル移動高速化

2　提案手法

2.1　注視外領域の利用

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 1 章に述べた．これに対し，注視していない領域（以下，注視外領域と呼ぶ）に特定のオブジェクトが含まれるかは比較的容易に判別可能であるといえる．その理由を以下に述べる．

目から 50cm 離れたディスプレイ上の注視点を計測した場合，固視微動によって注視点は約 4mm 四方の範囲を動き，また計測誤差は最大約 17mm であることから，ユーザが画面内のオブジェクトに注目している際に注視点となっている可能性があるのは，計測された座標の半径約 21mm 以内の範囲であるといえる [3]．使用しているディスプレイのサイズを 27 インチ，解像度を 1920 × 1200 ピクセルとした場合，21mm は 70 ピクセル弱となる．よって，計測した注視点から 70 ピクセル以上離れた全ての領域は，（計測誤差や固視微動を含めたとしても）ユーザが注視しようとしている範囲の外であると判別できる．したがって，ユーザが注視したいと思ってい

* 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, Kazutaka Kurihara, 産業技術総合研究所, 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

るターゲットを特定するよりも，あるオブジェクトを注視していないことを認識する方が容易である．

2.2 注視外領域でのカーソル移動高速化

注視外領域において，カーソル移動を高速化させることでポインティング時間を短縮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，ターゲット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 [4]．これを活用し，ターゲットに向かう際に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 [3] 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．注視点付近にカーソルをジャンプさせるなど一部のポインティング高速化手法ではこういった状況に対応できないか，あるいは注視点付近へジャンプさせる機能の ON/OFF を何らかの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，注視外領域におけるカーソル移動に制約は設けられていない．注視外領域では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，「微細に操作したい位置から目を背けている」というのは，スクロールバーなど一部のオブジェクトを除けば極めて不自然な状況であるため，注視外領域でのカーソル高速化はユーザのカーソル制御精度に悪影響を及ぼさないと考えられる．もし微細に操作したいオブジェクトがあれば先に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．

3 関連研究

本稿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[5] は，複数のカーソルを表示して同時に移動させることで移動距離を短縮させる手法である．複数のターゲットを同時に選択してしまう場合は，現在アイコンに乗っているカーソル以外を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複数提案されている [6][7]．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[8] 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[9] 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含め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選択が可能であるとの結果が出ている [10]．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，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 [11]．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) ことから，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[4] 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 [3] 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，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．これに対し本稿の提案手法では，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，ジャンプさせる手法ほどの問題を来さないと考えられる．

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[2] 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 [12] 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[13] 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[14] 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[15] 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．本稿の提案手法は 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．

4 システム

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 HSP3.3 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．

$$speed = dis/(length_{max}/stages) + speed_{min} \quad (1)$$

ここで dis は注視点とカーソルの距離，$length_{max}$ 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），$stages$ はカーソル速度を変更する段階数，$speed_{min}$ はカーソルの最低速度である．$stages$ はカーソルの最低速度と最高速度の差に 1 を加えた値であり，ここでは 10〜20 の 11 段階とした．なお，ここでの座標や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．本稿の評価実験で用いた 1920 × 1200 ピクセルのディスプレイの場合，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 2 のように速度が決定する．式 (1) における $length_{max}/stages$ の値は，図 2 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 70 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．

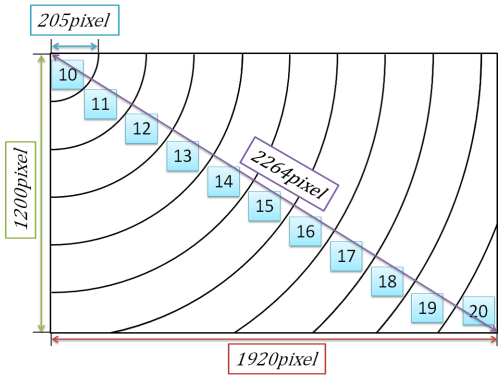

図 2．注視点が画面左上隅にある場合の，カーソル位置に応じた速度のマップ（10〜20 の範囲で変化）

5 評価実験

本章では提案手法の有効性を検証するため，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隣接した矩形選択タスクと，関連研究においてカーソル速度を変化させる手法の適用が困難なテキスト選択タスクを実施した．どちらも PC 操作において日常的に行う作業を想定したものであり，以下にそれぞれの詳細を記す．

5.1 タスク

矩形選択タスク

デスクトップに並んだアイコンをマウス操作によって選択することを想定し，図 3 のように隣接した灰色の矩形の中から，赤くハイライトされた矩形をクリックするタスクを設定した．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，矩形 1 個のサイズは外周の境界線を含めて縦 96 ×横 96 ピクセルとし，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．矩形のサイズはディスプレイ上で一辺 29mm である．

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ランダムに決定されてハイライトされ，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．ただし，視点とカーソルの距離が近いほど提案手法のカーソル速度変化は小さくなり，速度変化によるタスク達成時間への

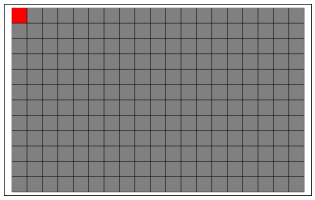

図 3. 矩形選択タスク開始時の画面

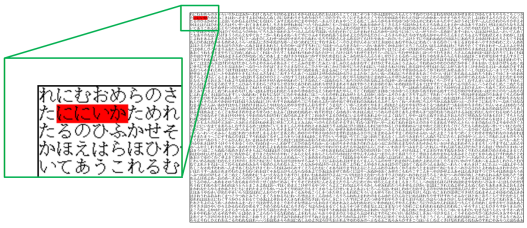

図 4. テキスト選択タスク開始時の画面と拡大図

影響が生じなくなることが想定される．そこで，注視点とカーソルの移動距離を大きくするために，n 個目と n+1 個目のターゲットの中心距離に制限を設けた．本タスクではその距離をディスプレイの横幅の 90%以上 100%以内とした．選択エラーは，ハイライトされた矩形以外の領域をクリックした場合にカウントされる．

テキスト選択タスク

画面内のテキストから単語を選択する操作を想定し，図 4 のようにランダムに赤くハイライトされた 4 文字を 15 回選択するタスクを設定した．ただし，キャレットの周囲にある文字によっては（特に画数の多い漢字では），キャレットの閲覧性が著しく低下するため，本タスクで表示されるのはランダムに選出された平仮名のみとした．これも矩形選択タスクと同様に最初の 1 つを反転させてから時間計測を開始する．テキストのフォントは MS 明朝，サイズは 10 ポイントであり，137 文字× 60 行の横書きである．ハイライトされた 4 文字分の赤い矩形は縦 20 ピクセル (実測 6mm)，横 80 ピクセル (25mm) である．また，キャレットは縦 16 ピクセル（4.5mm），横 7 ピクセル（1.8mm）である．

矩形選択タスクと同様の理由で，画面の上下左右端の文字はハイライト文字列として選出されないようにした．また，ターゲットとなる n 個目と n+1 個目の文字列の中心距離は，ディスプレイの横幅の 90%以上 100%以内である．

5.2 カーソル速度

本実験で比較する 4 種類のカーソル速度について説明する．

常時標準速度 常に速度 10（OS の標準速度）

常時最高速度 常に速度 20（OS の最高速度）

視点中心変速 視点付近は速度 10 で，視点から離れるほど高速，最高速度は 20（提案手法）

目標中心変速 視点中心変速と同様の速度変化をするが，ターゲットが中心（提案手法の理想状態）

標準速度をベースラインとし，単純な高速化，提案手法，提案手法の理想状態を比較した．理想状態は，ユーザがターゲットの中心を注視しているとしたものである．次のターゲットがハイライトされた瞬間に注視点も移動するという実際には起こり得ない状況だが（人間の視線は，50cm 離れた 21 インチディスプレイの対角線を移動するのに 150msec かかる [16]），注視点の特定が理想的にうまくいったと仮定したときに提案手法がどう働くかを検証するため比較対象に含めた．使用したマウスは BUFFALO 社製 BSMOU05M(1000dpi) であり，実距離で 1 インチ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,「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．

5.3 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の下端まで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．

被験者は大学生及び大学院生の 10 名（男性 9 名，女性 1 名，平均 21.9 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．

被験者にはまず矩形選択タスクを各カーソル速度で 5 回ずつ試行させた．休憩を試行毎に 10 秒間，速度毎に 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．アンケートの回答を含めて 10 分間の休憩をとり，その後テキスト選択タスクも同様に行った．各速度での 5 回の試行の平均操作時間と平均エラー率を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速度変化のアルゴリズムを伝えたが，試行前や休憩中にカーソルを動かす行為は認めなかった．また，タスクを試行する際のカーソル速度は被験者によって順序を変更した．

[1] http://msdn.microsoft.com/ja-jp/library/windows/hardware/gg463319.aspx（2012 年 10 月 19 日閲覧）

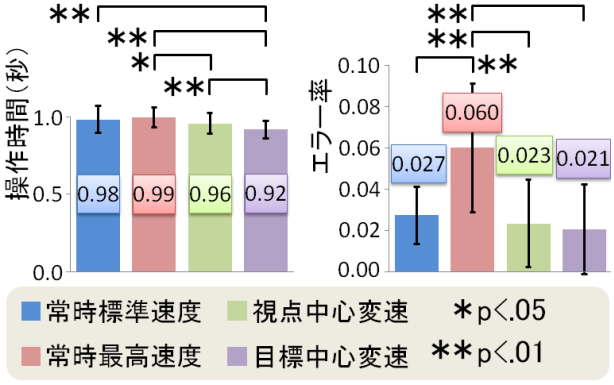

図 5. 矩形選択タスクの操作時間とエラー率

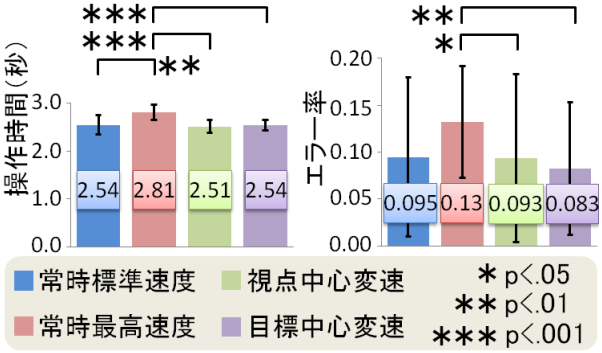

図 6. テキスト選択タスクの操作時間とエラー率

表 1. カーソルの制御しやすさ（5 段階評価の平均値）

項目	矩形選択	テキスト選択
常時標準速度のカーソルの制御しやすさ	3.7	3.3
常時最高速度のカーソルの制御しやすさ	2.3	1.9
視点中心変速のカーソルの制御しやすさ	4.3	4.1
目標中心変速のカーソルの制御しやすさ	4.1	3.9

表 2. 両タスクを通しての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
視点中心と目標中心で，速度のダイナミックな変化を意識したか	3	5	2
視点中心で，視線を制御することを意識したか	5	5	0
次のターゲット出現位置を予測できたか	7	3	0

5.4 結果と考察

各タスクにおいてターゲットを 1 つ選択するための平均操作時間及び平均エラー率を図 5，6 に示す．また，アンケート項目と結果を表 1，2 に示す．

矩形選択タスクにおいて操作時間を被験者毎の対応ありで分散分析（反復測定）をした結果，4 種類のカーソル速度間に有意差が見られた ($F(3,27)=8.416$, $p<.001$)．さらにカーソル速度をペア毎に多重比較したところ，目標中心変速と他 3 種類の速度との間で全て $p<.01$ の有意差が見られた．また，常時最高速度と視点中心変速との間で $p<.05$ の有意差が見られた．エラー率を同様に分散分析した結果，4 種類のカーソル速度間に有意差が見られ ($F(3,27)=9.670$, $p<.001$)，ペア毎の多重比較では常時最高速度と他 3 種類の間で全て $p<.01$ の有意差が見られた．

テキスト選択タスクにおいても同様に分析したところ，操作時間は 4 種類のカーソル速度間に有意差が見られ ($F(3,27)=14.989$, $p<.001$)，ペア毎の多重比較では常時最高速度と常時標準速度との間で $p<.01$，常時最高速度と視点中心変速・目標中心変速との間で $p<.001$ の有意差が見られた．エラー率は 4 種類のカーソル速度間に有意差が見られ ($F(3,27)=3.472$, $p<.05$)，ペア毎の多重比較では，常時最高速度と視点中心変速との間で $p<.05$，常時最高速度と目標中心変速との間で $p<.01$ の有意差が見られた．

矩形選択タスクでは，被験者全員とも目標中心変速が最短であり，理想的状況では注視外領域における長距離移動時にカーソルを高速化することの有用性が示せた．目標中心変速が視点中心変速よりも操作時間が有意に短かった理由として以下の 3 つが考えられる．第一に，視点中心変速では注視点の移動に時間を要することである．注視点がターゲット付近を注視するのが早いほどカーソルの高速化が早くなり，試行全体におけるカーソルの平均速度は高くなる．しかし，目の動く速さに限界があるため，この問題は避けられない．第二に，視点中心変速ではターゲット以外に注視点がおかれる時間があることである．被験者が移動中のカーソル位置を確認したり，固視微動によって注視点が移動することで，理想状態ほどのパフォーマンスが得られなかった可能性がある．これについてはカーソルと注視点の軌跡を分析し，視点中心変速では理想状態にどの程度近い性能を得られていたのかを検証する必要がある．第三に，視線計測器の計測精度の影響である．これについては，今後視線計測技術やハードウェアの性能が向上していくことで影響を小さくできると考えられる．

テキスト選択タスクでは提案手法の有効性は確認できなかった．これはカーソルの長距離移動がテキスト選択に要する時間より短く，高速化の影響が小さくなったためであると考えられる．「ターゲットや視点の付近ではもっと低速になるとよい」と述べた被験者がいたが，今回の文字選択タスクのような微細な操作が求められる作業においては，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．今回は標準速度及び最高速度との比較を行ったのみであるが，標準より低速にするのが有効なタスクにおいては，注視点から離れるほど標準速度に近づき，注視点に近いほど低速にすることで改善される余地があると思われる．

表 2 では，カーソル速度がダイナミックに変化することを意識していた被験者が 3 名おり，そのうち 1 名は意図的に速度変化を利用していたと述べた．意識的に目を動かすと長時間の作業では負荷が高くなる危険があるが，次第に意識しなくなっていく可能性もある．あるいは，本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ることも考えられるため，長期的な評価実験もしていきたい．また，意識していなかったと答えたうちの 3 名が「注視点付近では標準速度になるので気にならなかった」という旨を回答をしており，提案手法の狙いがうまく機能したものと考えている．視線を制御することを意識していた被験者 5 名は，全員ターゲットを注視するようにしていたと回答している．そのうち 1 名は「それによって疲労することはなかった」としているが，特に注意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．

6 まとめと今後の展望

注視点からの距離によっ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，カーソルの移動がディスプレイの横幅の 90%〜100%に及ぶ長距離の場合において，ターゲットを中心に速度を変化させる手法の有用性を確かめた．視点を中心にした場合も，有意差は確認できなかったものの操作時間やエラー率の改善が見られる被験者が両タスクにおいて過半数であった．また，カーソルの速度は利用時間が増大するにつれて段々と慣れていくと述べた被験者もいるため，提案手法を長期的に利用した場合の習熟度向上による操作時間の変化を観察したい．また，カーソルが注視点の方向へ移動するときのみ加速させたり，一定距離以上移動してから加速させるといった処理によって，意図しない高速化を低減し，操作性を向上させることを検討している．さらに，本稿の実験ではカーソルの速度や閾値などを筆者が設定したが，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ある．

参考文献

[1] Paul M. Fitts: 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).

[2] Tovi Grossman and Ravin Balakrishnan: The Bubble Cursor: Enhancing Target Acquisition by Dynamic Resizing of the Cursor's Activation Area, In Proc. of CHI '05, pp. 281–290(2005).

[3] 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).

[4] Shumin Zhai et al.: Manual And Gaze Input Cascaded (MAGIC) Pointing, In Proc. of CHI '99, pp 246–253(1999).

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

[6] Kari-Jouko Räihä and Oleg Špakov: Disambiguating Ninja Cursors with Eye Gaze, In Proc. of CHI '09, pp. 1411–1414(2009).

[7] Renaud Blanch and Michaël Ortega: Rake Cursor: Improving Pointing Performance with Concurrent Input Channels, In Proc. of CHI '09, pp. 1415–1418(2009).

[8] Takeshi Asano et al.: Predictive interaction using the Delphian Desktop, In Proc. of UIST '05, pp. 133–141(2005).

[9] Patrick Baudisch et al.: Drag-and-Pop and Drag-and-Pick: techniques for accessing remote screen content on touch- and pen-operated systems, In Proc. of Interact '03, pp. 57–64(2003).

[10] Linda E. Sibert and Robert J.K. Jacob: Evaluation of Eye Gaze Interaction, In Proc. of CHI '00, pp. 281–288(2000).

[11] 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).

[12] 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).

[13] Renaud Blanch et al.: Semantic pointing: improving target acquisition with control-display ratio adaptation, In Proc. of CHI '04, pp. 519–526(2004).

[14] Aileen Worden et al.: Making Computers Easier for Older Adults to Use: Area Cursors and Sticky Icons, In Proc. of CHI '97, pp. 266–271(1997).

[15] 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, 日本ソフトウェア科学会論文誌（コンピュータソフトウェア）, Vol. 28, No. 2, pp. 140–152(2011).

[16] 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).