NAIST-IS-DT9961029

博士論文

一般的なGUIに適した視線追加型インタフェース

大和 正武

2002年2月12日

奈良先端科学技術大学院大学
情報科学研究科 情報システム学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
博士 (工学) 授与の要件として提出した博士論文である。

大和 正武

審査委員： 松本 健一 教授
横矢 直和 教授
小笠原 司 教授
井上 克郎 教授

一般的なGUIに適した視線追加型インタフェース*

大和 正武

内容梗概

本論文では，MS-Windowsなどの一般的なGUI上でのアイコン選択やスクロールなどの操作の効率を上げることを目的として，従来から用いられているキーボードやマウスなどの入力デバイスに加え，計算機画面を見るユーザの視線の動きを用いた視線追加型インタフェースを提案する．視線の移動速度は極めて速いため，マウスと比較して高速にポインティング操作を行うことができるなど視線は計算機の入力手段として有望である．

本論文では(1)ウィンドウ，アイコン，メニューの選択(ターゲット選択)，(2)特定の場所へのターゲットの移動(ドラッグ&ドロップ)，(3)ウィンドウのスクロールの3種類の操作に対して，視線の動きを入力として追加する方式を提案する．これら3種類の操作に視線を追加することによって一般的なGUI上の操作全般の操作時間を短縮することができる．

ターゲット選択では，ポインティング操作に視線を，確定操作には従来通りマウスボタンを用いる．さらに一般的なGUIで用いられる1cm四方程度の細かなターゲットを選択できるよう，カーソル位置微調整方式としてAuto方式，Manual方式，SemiAuto方式の3つを提案する．一般的なGUIを想定した環境で評価実験を行った結果，SemiAuto方式を用いた場合，選択誤りを大幅に増やすことなく，操作時間を同程度かより短縮できることが分かった．特に，非連続操作（カーソルの初期位置が不定の場合の選択操作）においては，操作時間を約2/3に短縮できた．

ドラッグ&ドロップでは，操作をドラッグアイコンとドロップアイコンに対する2回の連続したターゲット選択であると見なし，ターゲット選択方式として最

*奈良先端科学技術大学院大学 情報科学研究科 情報システム学専攻 博士論文, NAIST-IS-DT9961029, 2002年2月12日.

も性能の優れていたSemiAuto方式を適用する．評価実験の結果，提案方式による操作は従来のマウスのみの操作に比べて，操作誤りを大幅に増やすことなく，1回の操作について平均で約17%(0.4秒)短く操作できた．また初心者を想定した実験では，操作誤りも減り，平均で約27%(0.8秒)短く操作できた．

スクロールでは，ウィンドウ中央からユーザの画面上の注目箇所(注視点)までの垂直方向の距離をスクロール速度に比例させる．ユーザはウィンドウ内で視線を移動させるだけで，表示したいテキスト部分をウィンドウ中央部に移動させ，表示させた状態でスクロールを停止させることができる．評価実験の結果から，テキストのスクロールにより文字列を検索するという作業(タスク)においては，視線による自動スクロールはキーボード操作によるスクロールと比較して同程度かそれ以上に有効であることが分った．

キーワード

ユーザインタフェース，視線追跡装置，入力デバイス，ターゲット選択，ドラッグ&ドロップ，スクロール

Gaze-Added Interface Suitable for General GUIs*

Masatake YAMATO

Abstract

The purpose of this thesis is to increase the efficiency of GUI operations under general GUIs such as MS-Windows. For this purpose, I propose employing user's eye-gaze on the computer screen as an additional input for general GUIs. The eye-gaze is used with conventional input devices such as a keyboard and a mouse. Using the eye-gaze as input for GUIs is promising because moving the eye-gaze is much faster than moving a mouse by hand.

In this thesis, I propose the 3 concrete methods to use the eye-gaze in the following operations: (1) selecting a window, menu and icon(target selection), (2) moving a target to a specified position(drag-and-drop) and (3) scrolling a window. With using these methods, users can operate GUIs faster than with using a mouse only.

In target selection, the eye-gaze is used for pointing operation; and the mouse button is used for click operation. However, it is difficult for the user to move the mouse cursor to the target on a general GUI on which small targets(about $1cm^2$) are put in narrow layout. I also propose three cursor adjustment methods (Automatic, Manual, and SemiAuto). I conducted an experiment to evaluate a proposed methods on a GUI modeled on a general GUI. The result shows SemiAuto method is 1.5 times faster than a mouse only operation in discontinuous

*Doctor's Thesis, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT9961029, February 12, 2002.

selections, and SemiAuto method is same or less errorful than a mouse only operation.

In drag-and-drop, I consider drag-and-drop as two continuous target selections; and SemiAuto method is used for target selection. The result of an experiment showed the proposed method is about 17%(0.4 sec.) faster than the mouse only operation per one drag and drop operation in average; and the error rate of the proposed method is almost the same as the mouse only operation. Furthermore, with using non-preferred hand, which is known to behave like a beginner's mouse operation, the error rate of the proposed method is smaller than the mouse only operation; and the proposed method is about 27 %(0.8 sec.) faster than the mouse only operation in average.

In scroll, a position of a eye-gaze within a text window determines how to scroll the window, what portion of the text to display, and where to stop. Speed of scrolling is proportioned to the distance of the gaze point from the center of the window. A user of the method can access a portion of the displayed document without using keyboard or mouse. the effectives of scrolling using this method is as good as one with conventional keyboard control as respects string search by scrolling.

Keywords:

user interface, eye tracking device, input device, target selection, drag-and-drop, scroll

研究業績

学術論文誌

1. 大和正武, 門田暁人, 高田義広, 松本健一, 鳥居宏次: 視線によるテキストウィンドウの自動スクロール, 情報処理学会論文誌, Vol.40, No.2, pp.613–622, 1999.2.

2. 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 に適した視線による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, 2001.6.

3. 大和正武, 神代知範, 門田暁人, 松本健一: 視線・マウス併用型インタフェースのドラッグ&ドロップ操作への適用, 情報処理学会論文誌 (条件付き採録).

国際会議

1. Masatake Yamato, Akito Monden, Ken-ichi Matsumoto, Katsuro Inoue and Koji Torii: Button selection for general GUIs using eye and hand together, *Proceedings of the 5th International Working Conference on Advanced Visual Interfaces* (AVI2000), pp.270–273, May, 2000.

2. Masatake Yamato, Akito Monden, Ken-ichi Matsumoto, Katsuro Inoue and Koji Torii: Quick button selection with eye gazing for general GUI environments, *Proceedings of Internatonal Conference on Software: Theory and Practice* (ICS2000), pp.712–719, August, 2000.

3. Masatake Yamato: Gaze added interface suitable for general GUIs, *Proceedings of 8th IFIP TC.13 Conference on Human-Computer Interaction* (INTERACT2001), pp.665–668, July, 2001.

研究会・シンポジウム

1. 大和正武, 高田義広, 鳥居宏次: 視線追跡によりプログラムエディタを自動スクロールする方式の比較, 電子情報通信学会技術研究報告, ソフトウェアサイエンス研究会, SS97–16, pp.1–8, 1997.7.

2. 大和正武, 神代知範, 門田暁人, 松本健一, 井上克郎: 視線によるマウスカーソルの自動移動, 情報処理学会ヒューマンインタフェース研究報告, 99–HI–84–13, pp.73–78, 1999.8.

3. 大和正武, 大野健彦: 視線を利用したアプリケーションの内部構造可視化インタフェース, 電子情報通信学会技術研究報告, ヒューマン情報処理研究会, HIP2000–12, pp.37–42, 2000.6.

全国大会

1. 大和正武, 宇和伸明, 金子寛彦, 金次保明: 奥行き運動刺激に対する重心動揺の時間的変化, 映像情報メディア学会冬季大会, pp.72, 1997.12.

目 次

図目次

表目次

第 1 章

はじめに

近年，人間 (ユーザ) の視線を計算機への入力に利用した「Gaze Interface (視線インタフェース)」の研究が盛んに行われている．視線インタフェースは，その目的によって大きく 2 つに分けられる (図 1.1参照)．視線インタフェースの研究目的の一つは，手を使えない状況でも文字入力やボタンの選択などの計算機の操作を行なう手段を提供することである．例えば，肢体不自由者のコミュニケーションを支援すること [1,3,33] やモバイル端末上で文字入力を行うこと [8] が挙げられる．このような目的を持つ視線インタフェースは特に Gaze-Centered Interface(視線中心型インタフェース) と呼ばれる [24]．視線中心型インタフェースはマウスやキーボード等を用いる従来のインタフェースと比べると必ずしも効率は良くないが，手が使えない状況では大変有用である．

視線インタフェースのもう一つの研究目的は，マウスやキーボード等の手で操作する従来の入力デバイスに加えて視線の動きをを入力として用いることで，より効率の良いインタフェースを実現することである．このような併用型のインタフェースは Gaze-Added Interface(視線追加型インタフェース) と呼ばれる [24,39]．視線の移動速度は極めて速く，眼球から 50cm 前方に位置する 21 インチディスプレイの対角上を端から端まで移動する場合でも 150msec 程度の時間しか要しない [20]．従って，例えばマウスカーソル (カーソル) を GUI 部品へ移動させる操作 (ポインティング操作) のために視線を入力として追加すれば，ボタンやアイコンなどのターゲット選択を効率良く行うことができる可能性がある [38]．

ただし従来より提案されている Gaze-Added Interface の多くは，視線で操作す

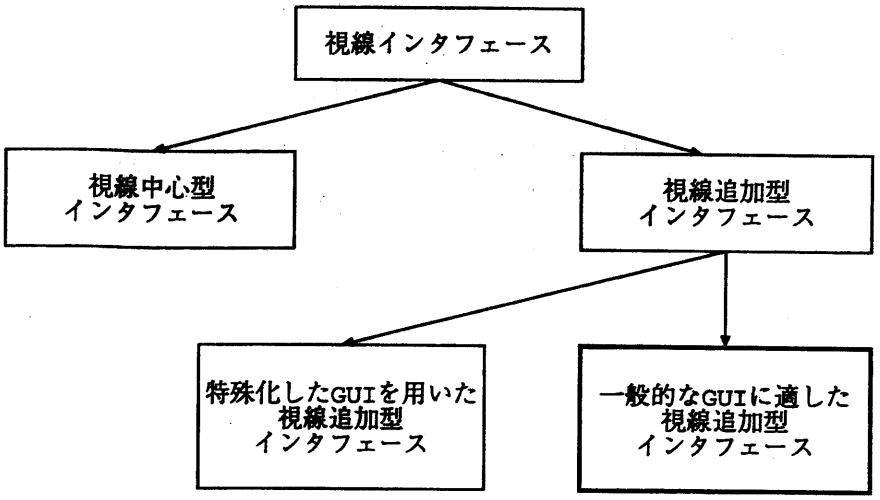

図 1.1 視線インタフェースの分類と本研究の位置付け

ることができるように特殊化されたGUI上で用いることを前提としていたり，あるいは特殊化されたGUI上で評価が行なわれているため一般的なGUI(Microsoft Windows，Mac OSなどが提供するGUI)にも適用可能であるか明らかでない．一般的なGUIに比べて，特殊化されたGUI上では巨大化されたGUI部品が広い間隔で配置されている．

本論文では，一般的なGUIに適したGaze-Added Interfaceの実現を目的とする．現在普及している一般的なGUI上でのGaze-Added Interfaceを実現できれば，既存のアプリケーションソフトウェアのGUIを新規に作り直す必要なく操作を効率化できるため，多くの計算機ユーザにとってより有用であると考えられる．

一般的なGUI上には様々な操作がある．視線の追加により操作が効率化できるか，効率化できるのであればどのように追加すれば効率化できるのか操作毎に検討する必要がある．一般的なGUIはWIMP(Window, Icons, Menu, Pointing device)から実現される[14]．ユーザはPointing device(ポインティングデバイス)を使ってIcons(アイコン)，Menu(メニュー)，Window(ウィンドウ)の3つのGUI部品(図1.2参照)を操作する．現状では，ポインティングデバイスとしてマウスが広く使われている．ユーザが行うことができる操作はGUI部品毎に異なる．次にGUI部品毎の操作を示す．

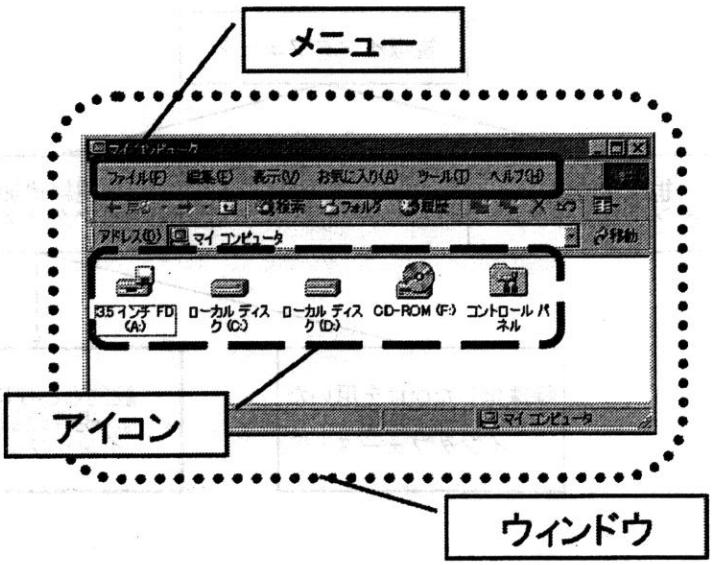

図 1.2 一般的な GUI を構成する GUI 部品

ウィンドウ 選択，移動，リサイズ，スクロール，ズーム，

アイコン 選択，移動，ドロップ

メニュー 選択

本論文では，一般的な GUI における (1) ウィンドウ，アイコン，メニューの選択 (ターゲット選択)，(2) アイコンの移動とドロップ，ウィンドウの移動とリサイズ (ドラッグ&ドロップ)，(3) ウィンドウのスクロールの 3 つの操作グループに対して，視線を入力として追加することで効率化する方法を述べる．視線を導入した 3 つの操作グループにより，一般的な GUI の大部分の操作をより効率良く行うことができる．

(1) ターゲット選択…本論文 2章では，視線とマウスを併用したターゲット選択方式を提案する．マウスのみを用いる場合，ウィンドウ，アイコン，メニューのいずれの GUI 部品に対しても，GUI 部品の領域へマウスカーソル (カーソル) を移動し指し示す操作 (ポインティング操作) とマウスボタンを押して選択を確定する操作 (確定操作) によって選択を行う．そこでウィンドウ，アイコン，メニュー

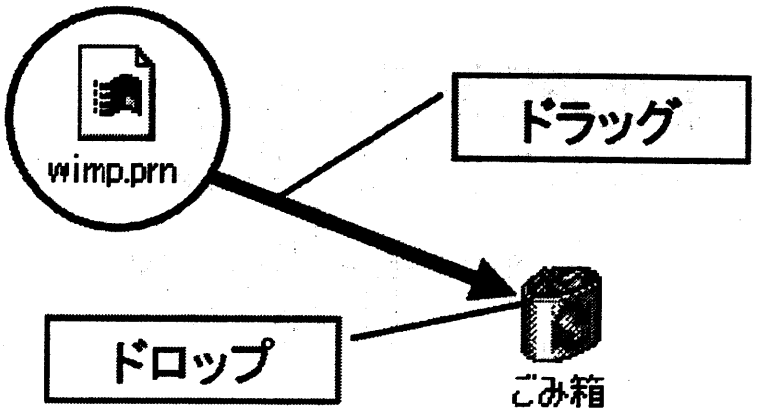

図 1.3 アイコンのドラッグ&ドロップ

の選択を総称してターゲット選択と呼ぶこととする．2章で提案する方式では，ポインティング操作に視線を用い，確定操作には従来通りマウスボタンを用いる．

(2) ドラッグ&ドロップ…本論文3章では，視線とマウスを併用したドラッグ&ドロップを提案する．図 1.3に示す通り，マウスのみを用いる場合，ドラッグ&ドロップは，あるアイコンの位置でマウスボタンを押した後，目的の位置までマウスボタンを離さずにドラッグ (移動) して，ドロップ先のアイコンの位置でマウスボタンをリリース (ドロップ) する操作である．(1) ターゲット選択と (2) のドラッグ&ドロップにより，視線を併用してアイコンに対する全ての操作を効率化できる．加えて，図 1.4に示す通りウィンドウの移動とは「ウィンドウのタイトルバーをアイコンのかわりドラッグして特定の場所へ移動する操作」であると見なせば，ウィンドウの移動についても視線を併用したドラッグ&ドロップ方式によって効率化できる可能性がある．同様にウィンドウのリサイズを「ウィンドウのリサイズエリアをアイコンのかわりドラッグして特定の場所へ移動する操作」であると見なせば (図 1.5参照)，ウィンドウのリサイズについても効率化できる可能性がある．

(3) スクロール…本論文4章では，視線による自動スクロール方式を提案する．自動スクロール方式では，ウィンドウ内での視線の移動方向によりスクロールの方向と速度を決定し，視線を移動させるだけでスクロールできる．視線によりスク

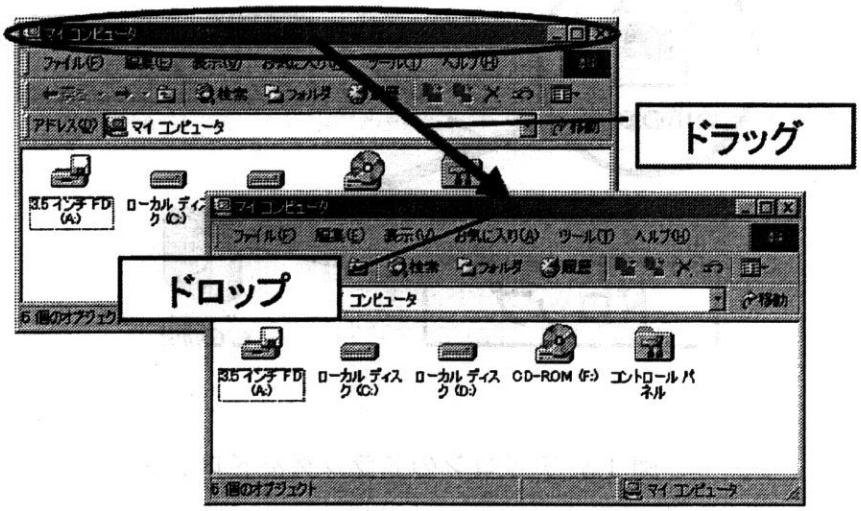

図 1.4 ウィンドウの移動

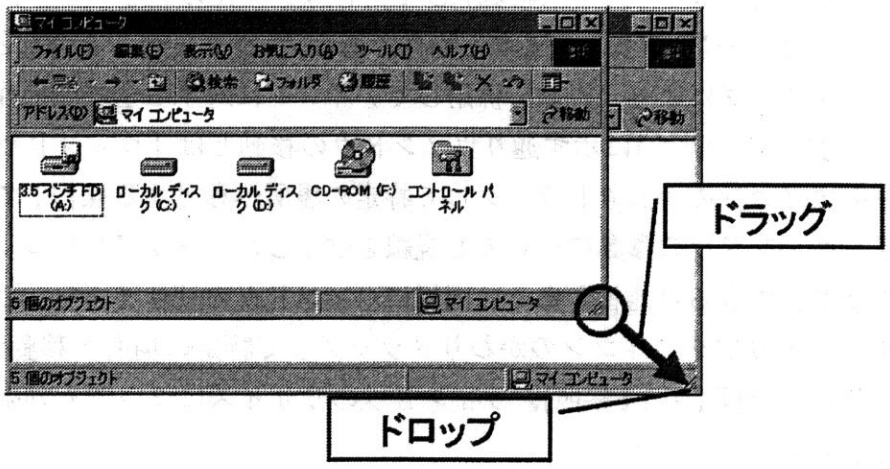

図 1.5 ウィンドウのリサイズ

ロール操作を行うことで，ウィンドウ上の長文テキストをスクロールによって参照しながら，それとは別のウィンドウ上でテキストを入力するといった作業において，テキストの入力を中断する必要なく，スクロールをすることができ利用者の負担が減ることが期待される．

最後に，5章において本論文の全体のまとめを述べる．

第 2 章

視線とマウスを併用したターゲット選択

2.1. あらまし

本章ではウィンドウ，アイコン，メニューなどの GUI 部品 (ターゲット) の選択を対象とする視線追加型インタフェースについて述べる [38].

視線インタフェースの研究のなかでもターゲット選択を対象とする研究は特に盛んである [9,10,22]. 視線の移動速度は極めて速く，眼球から 50cm 前方に位置する 21 インチディスプレイの対角上を端から端まで L 移動する場合でも 150msec 程度の時間しか要しない [20]. また，Graphical User Interface(GUI) 上の操作では，ユーザはまず操作対象となるターゲットに視線を向ける場合が多いことも知られている [44]. ポインティング操作 (ターゲットへマウスカーソルを移動し指し示す操作) に限って言えば，マウスよりも視線の方が正確に効率よくしかも自然に行える可能性が高い. 従って，ターゲットの選択操作を「ポインティング操作」と「確定操作」に分け，前者を視線で，後者をマウスボタンのクリックで行えば，GUI 上のターゲット選択操作の正確さ，効率，自然さを向上できる可能性がある [37,38].

但し，一般的な GUI 上 (Microsoft Windows，Mac OS，X-Window などが提供する GUI 上) でポインティング操作を視線で行う場合，ユーザの目の固視微

動と視線計測装置 (アイカメラ) の計測誤差が問題となる．固視微動とは,「ユーザは一点を見つめているつもりでも実際の視線は絶えず動いている」という現象である [7,13]．眼球から 50cm 前方に位置する計算機ディスプレイ上では，固視微動によりユーザの注目する箇所 (注視点) がおよそ 0.4cm 四方の範囲で細かく振動し，それに計測誤差がおよそ 0.9〜1.7cm 加わる [20,31,35]．一般的な GUI 上では，ターゲットの幅 (サイズ) は小さいものでおよそ 1cm 四方となる．固視微動と計測誤差が発生する状況下で，ターゲットへマウスカーソルを正確に移動し，確定操作 (ターゲットを選択するためにマウスボタンをクリックする操作) の間ターゲットを指し示し続ける，という操作を視線で行うことは容易ではない．ターゲットが接近して配置されている場合には，誤選択となる可能性も高い．

ポインティング操作を視線で行う従来の研究では，固視微動や計測誤差がユーザ操作の妨げとならないよう，比較的大きな GUI ボタンを，間隔を広くとって配置した特別な GUI を用いる場合が多い [15,20]．また，マウスと視線で作業効率や誤り率を比較する実験も，そうした特別な GUI 上で行われる場合が多い [26,44]．一般的な GUI 上での視線の利用を考えるのであれば，従来の提案方式や評価実験の結果をそのまま利用することは適当とはいえない．

本章では，一般的な GUI 上でのポインティング操作に視線を利用する目的で，固視微動や計測誤差が発生してもターゲット選択を正確に効率よく視線で行うことの出来る，具体的な視線・マウス併用方式を検討し，実際の GUI 環境に近い状態で評価する．ここで，一般的な GUI とは，

(C1) ポインティング操作のターゲットの幅 (サイズ) が 1cm 程度．

(C2) 隣り合うターゲット間の距離が 0cm 以上 (ターゲットが隣接している場合も想定する)．

とする．なお，以降では便宜的に,「ポインティング操作」を視線のみで行ない，確定操作をマウスボタンのみで行う」といった，視線とマウスの比較的単純な組み合せでターゲットを選択する方式を「基本併用方式」と呼ぶ．一方，本章で比較検討する方式 (固視微動や計測誤差に対する対策を講じている方式) を「改良併用方式」と呼ぶ．改良併用方式は次の 3 つである．

- Auto 方式：各ターゲットの周囲に確定操作を行うための領域 (確定領域) を設けておき，この領域内で確定操作が行われた場合にカーソルをターゲッ

ト上へ自動的にジャンプさせる．ターゲットのサイズが仮想的に大きくなり，ユーザはターゲットを選択しやすくなる．

- Manual 方式：視線によってマウスカーソルをターゲットの近傍まで移動した (粗い位置決めを行った) 後，ポインティング操作用デバイスを視線からマウスに切り替える．視線によるマウスカーソルの高速移動という特性を生かしたまま，ターゲットへの正確なカーソル移動を容易にする．

- SemiAuto 方式：上記 2 つの方式を併用する．これにより，ターゲットの配置に応じたより柔軟なポインティングを可能にする．

なお，Auto 方式と SemiAuto 方式は本論文で新たに提案する方式である．一方，Manual 方式は，文献 [44] で提案されている方式とほぼ同じである．但し，文献 [44] では，(C1), (C2) を満す一般的な GUI 環境での評価は行われていない．

評価実験では，(C1) と (C2) を満たす実験用 GUI を準備し，これら 3 つの方式を従来方式 (マウスのみによるターゲット選択) と比較する．比較の観点は，ターゲット選択操作における操作時間と誤り率である．

以降 **2.2**では，方式の比較検討の準備として，Fitts の法則に基づいてターゲットの選択操作を基本動作に分解し，マウスのみによる操作，及び，基本併用方式による操作のそれぞれをモデル化する．**2.3**では，**2.2**での準備に基づいて，3 つの改良併用方式を比較検討し，それぞれの選択操作の流れを述べる．**2.4**では，5 名の被験者を対象に実施した評価実験の方法とその結果を示す．**2.5**で関連する研究を紹介し，**2.6**でまとめる．

2.2. ターゲット選択操作モデル

2.2.1 マウスによる選択操作

ターゲット選択操作を，ポインティング操作 A_M と確定操作 A_S に分ける．このうちポインティング操作 A_M は，Fitts の法則 [2,30] においては，知覚動作 A_p(ターゲットとカーソルの知覚)，認知動作 A_c(ターゲットとカーソルとの誤差の認知)，運動動作 A_m(カーソルのターゲットへの移動) の一連の動作を最小単位 (1 サイ

クル) とした繰り返しの操作であると捉えられる (図 2.1参照). ターゲットのサイズ (幅) を S, i サイクル目におけるカーソルとターゲット間の距離を X_i とおくと, 認知動作 A_c において $X_i < S/2$ ならばユーザは確定操作 A_S を行う.

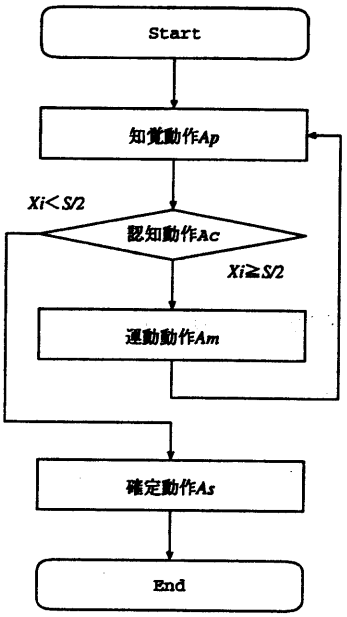

図 2.1 マウスによるターゲット選択の流れ

Fitts の法則では, 図 2.1の一連の操作に要する時間 T_{mouse} は次の式で与えられる.

$$T_{mouse} = I_{mouse} \cdot log_2 \frac{2D}{S} + T(A_S) \tag{2.1}$$

$$I_{mouse} = -\frac{\tau(A_p) + \tau(A_c) + \tau(A_m)}{log_2 \varepsilon}$$

ただし,

D : 初期状態におけるカーソルとターゲット間の距離

ε : 1 サイクルの動作 ($A_p \rightarrow A_c \rightarrow A_m$) によるターゲットの接近の精度.

τ : 1 サイクルにおける所要時間

T : 1 選択操作における所要時間

2.2.2 基本併用方式による選択操作

基本併用方式によるターゲットの選択操作は，主に次の3点においてマウスのみによる選択操作と異なる：

- GUI上のカーソル位置に関わらず，ターゲットに視線を移した時点で，ターゲット付近までカーソルを一瞬にして移動させることが可能である．
- 眼球には固視微動があるため，カーソルを一定箇所に停止させておくことが困難である [20,31]．
- 注視点の座標には計測誤差が含まれるため，意図した通りの箇所に正確にカーソルを移動させることが困難である [40,41]．

基本併用方式によるターゲット選択操作の流れは，図2.2のフローチャートで表される．マウスによる選択操作との違いは，「ターゲット初期知覚動作 (A_{pt})」が加わっている点である．基本併用方式による選択操作では，まず，ユーザが最初にターゲットに注目した (A_{pt}) 時点で，ターゲット付近 (注目箇所) へカーソルがジャンプする．次に，マウスのみによるターゲット選択と同様に，知覚動作 (A_p)，認知動作 (A_c)，運動動作 (A_m) が繰り返される．そして，カーソルがターゲットの領域に入ったとユーザが判断した場合に，マウスボタンをクリックすることで確定動作 (A_S) が行われる．

ここで，固視微動の幅の平均値を e_j，アイカメラの計測誤差の平均値を e_m とおくと，固視微動と計測誤差の合計値 ($e_j + e_m$) がターゲットのサイズより小さい場合 ($S > e_j + e_m$) には，動作 A_{pt} によりカーソルがターゲット上に来るため，知覚動作 (A_p)，認知動作 (A_c) の後，($A_m \rightarrow A_p \rightarrow A_c$ のループに入らずに) 確定動作 (A_S) が行われる．一方，$S \leq e_j + e_m$ となる場合にはループに入るが，カーソルをターゲットに正確に近付けていくことは困難であるため，ループを回る回数はマウスと比べて非常に大きくなる．

基本併用方式によるターゲット選択に要する時間 T_{eye} は，次の式で与えられる．

- $S > e_j + e_m$ の場合：

$$\begin{aligned} T_{eye} &= T(A_{pt}) + \tau(A_p) + \tau(A_c) + \\ &\quad + T(A_S) \end{aligned} \tag{2.2}$$

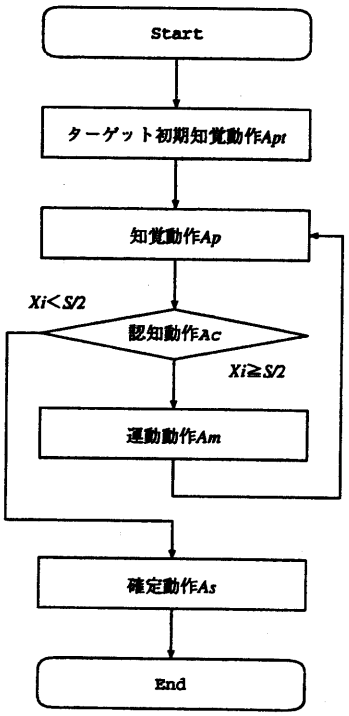

図 2.2 基本併用方式によるターゲット選択の流れ

- $S \leq e_j + e_m$の場合：

$$\begin{aligned} T_{eye} &= T(A_{pt}) + I_{eye} \cdot log_2 \frac{2D_{eye}}{S} \\ &\quad + T(A_S) \end{aligned} \tag{2.3}$$

$$I_{eye} = -\frac{\tau(A_p) + \tau(A_c) + \tau(A_m)}{log_2 \varepsilon}$$

ただし，

D_{eye}：ターゲット初期知覚動作後のカーソルとターゲットの距離
($D_{eye} \approx e_j + e_m$)

式(2.2), (2.3)において，$T(A_{pt})$の値は一般にごく小さい．眼球から50cm前方に位置する21インチディスプレイの対角上を端から端まで移動する場合でも150msec程度の時間しか要しない [20]．従って，$S > e_j + e_m$の場合には$T_{eye} < T_{mouse}$となる可能性が高い．一方，$S \leq e_j + e_m$の場合には$T_{eye} > T_{mouse}$となる可能性が高い．固視微動と計測誤差のためにターゲット接近の精度εの値が著しく小さな値になると予想されるためである．

一般的なGUIでは，式(2.2), (2.3)中の定数はおおよそ次の値をとる [20,31]．

- ターゲットのサイズ(幅)Sは1cm程度である．
- 眼球から50cm前方に位置するディスプレイ上では，
  - 固視微動の平均値e_jは0.4cm程度
  - 注視点の計測誤差e_mは0.9cm～1.7cm程度

  となる [20]．

従って，一般的なGUIにおいて$S \leq e_j + e_m$となるため$T_{eye} < T_{mouse}$となる可能性が高い．つまり，「ポインティング操作に視線を用い，確定操作にマウスボタンを用いる」という基本併用方式では，マウスのみを用いたターゲット選択よりも効率が悪くなる可能性が高い．実際に基本併用方式を用いて1cm四方のターゲットを選択する予備実験を行なったところ基本併用方式はマウスのみを用いた操作するよりも効率が悪いかった [35]．

2.3. 改良併用方式

2.3.1 Auto 方式

$S > e_j + e_m$とする一つの方法は，Sを仮想的に大きくすることである．Auto 方式では，各ターゲットの周囲に確定操作を行うための領域 (確定領域) を設けておき，この領域内で確定操作が行われた場合にカーソルをターゲット上へ自動的にジャンプさせる．任意のターゲット i から最も近傍のターゲットまでの距離を d_iとすると，ターゲット i の確定領域のサイズ (ターゲット i の仮想的なサイズ) は $S + d_i$となる (図 2.3).

Auto 方式におけるユーザの操作の流れを図 2.4に示す．ユーザは，認知動作 (A_c) においてカーソルが確定領域に入った ($X_i < (S + d_i)/2$ となった) と判断した時点で，確定動作に移ることができる．なお，$X_i < (S + d_i)/2$ となった時点で自動的にターゲットの表示を変化させることで，ユーザはカーソルが確定領域に入ったことを容易に知ることができる．なお，図中には明記していないが，例外処理として，カーソルがどの確定領域にも属さない状態で確定操作を行った場合には，カーソルに最も近いターゲットの選択が確定する．

Auto 方式を用いることで，ターゲットのサイズが事実上大きくなり ($S \rightarrow S + d_i$)，ユーザはマウスカーソルをターゲット上に移動しやすくなる．ただし，d_iが小さい場合には本方式の効果は小さい．

2.3.2 Manual 方式

$S > e_j + e_m$とするもう一つの方法は，$e_j + e_m$を実質的に無視できるほど小さくすることである．Manual 方式では，ユーザがマウスを動かした時点でポインティング操作用デバイスをアイカメラ (視線) からマウスに切り替わる．ポインティング操作用デバイスがマウスに切り替わった時点で実質的に $e_j + e_m = 0$ となり，ユーザは手でマウスを動かしてカーソルをターゲット上へ移動できる (図 2.5).

Manual 方式におけるユーザの操作の流れを図 2.6に示す．フローチャートの前半では，ユーザは視線によるポインティング操作を行う．ユーザは，認知動作 A_c

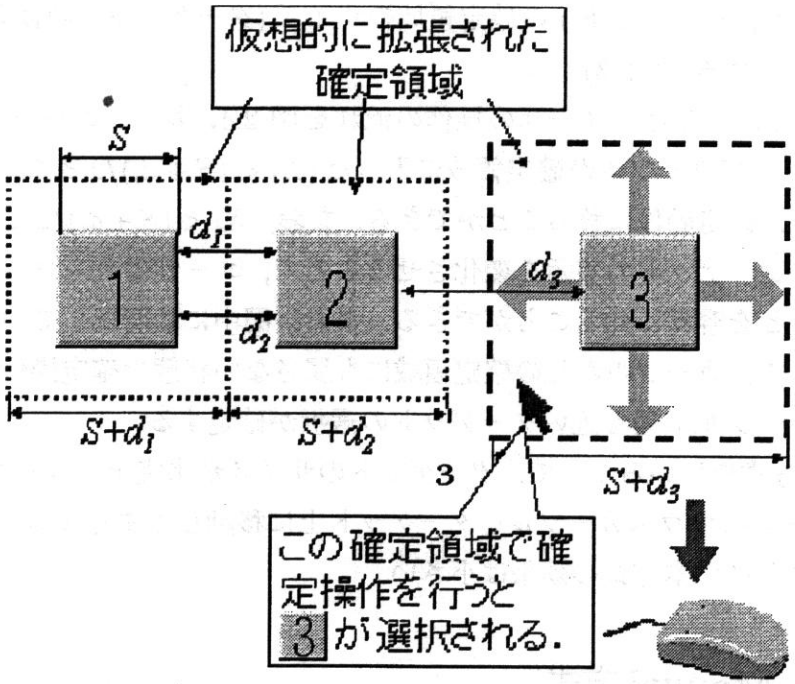
仮想的に拡張された
確定領域
S
d_1
d_2
d_3
1
2
3
$S+d_1$
$S+d_2$
$S+d_3$
3
この確定領域で確
定操作を行うと
3 が選択される.

図 2.3 Auto 方式

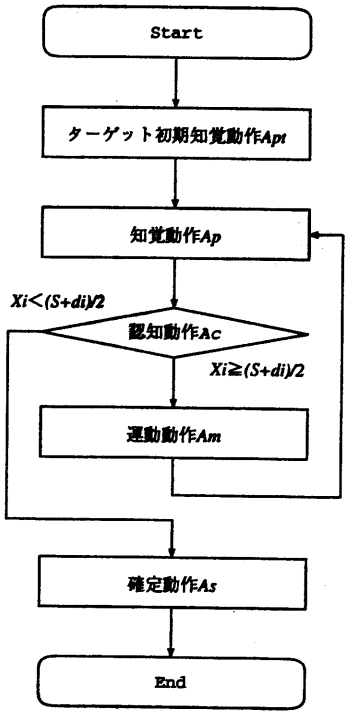

図 2.4 Auto 方式によるターゲット選択の流れ

において，カーソルがターゲットに接近した ($X_i < C$となった：Cは距離を表す定数) と判断した時点で，マウスを動してデバイスの切替えを行う (動作 A_{sw})．この時点から，ユーザの操作はフローチャートの後半に移り，マウスを用いたポインティング操作，および，確定操作を行う．なお，ユーザが (マウスをクリックして) 確定操作を行った時点で，ポインティング操作用デバイスは再びアイカメラに戻る．

Manual 方式では，視線によるマウスカーソルの高速移動という特性を生かしたまま，ターゲットへの正確なカーソル移動が容易になる．

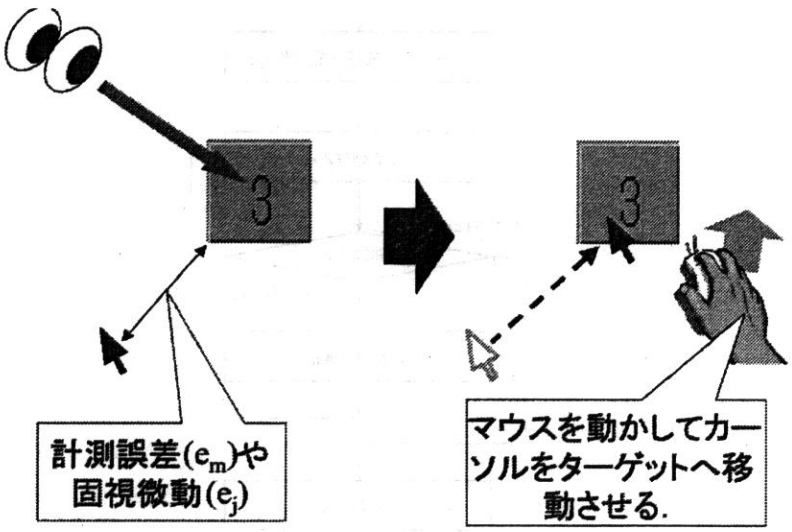

図 2.5 Manual 方式

2.3.3 SemiAuto 方式

SemiAuto 方式は Auto 方式と Manual 方式を組み合わせた方式である (図 2.7)．Auto 方式と同様に，各ターゲットの周囲に確定領域ができる．また，Manual 方式と同様に，ユーザがマウスを動かした時点でポインティング操作用デバイスがアイカメラからマウスに切り替わる．即ち，Sを仮想的に大きくすると同時に，$e_j + e_m$を実質的に無視できるほど小さくすることになる．

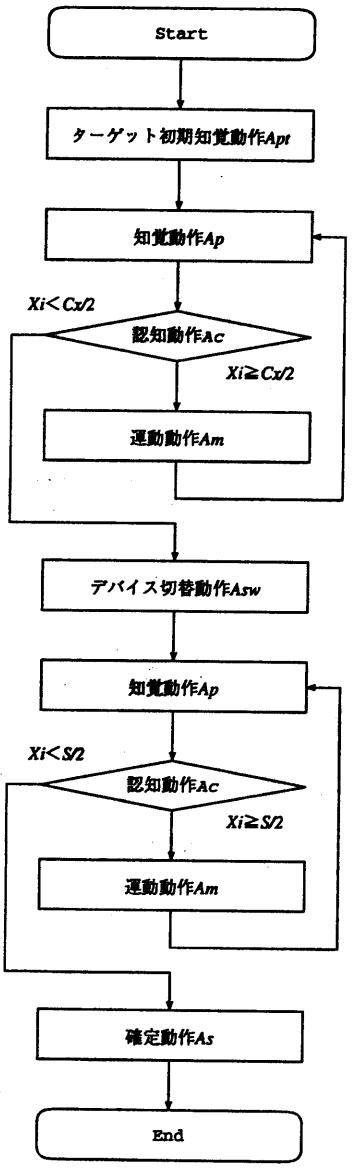

図 2.6 Manual 方式によるターゲット選択の流れ

SemiAuto 方式におけるユーザの操作の流れを図 2.8に示す．ユーザは，視線によるポインティング操作をまず試みる．カーソルが仮想的なターゲットの領域に入ったと判断した (認知動作 A_{c1}) ならば，選択動作に移ることができる．また，カーソルがターゲットに接近したと判断した (認知動作 A_{c2}) ならば，マウスを動してデバイスの切替え (A_{sw}) を行うこともできる．デバイスがマウスに切り替わった後も各ターゲットの確定領域は有効であり，ユーザはマウスを手で動かしてカーソルを確定領域へ移動させることで確定操作に移ることができる．

SemiAuto 方式は，ターゲットの仮想的なサイズを大きくすることでポインティングを容易にするという Auto 方式の利点と，マウスへのデバイス切替えを可能にすることで視線の計測誤差と固視微動を実質的にゼロにできるという Manual 方式の利点の，双方の利点を持つことになる．

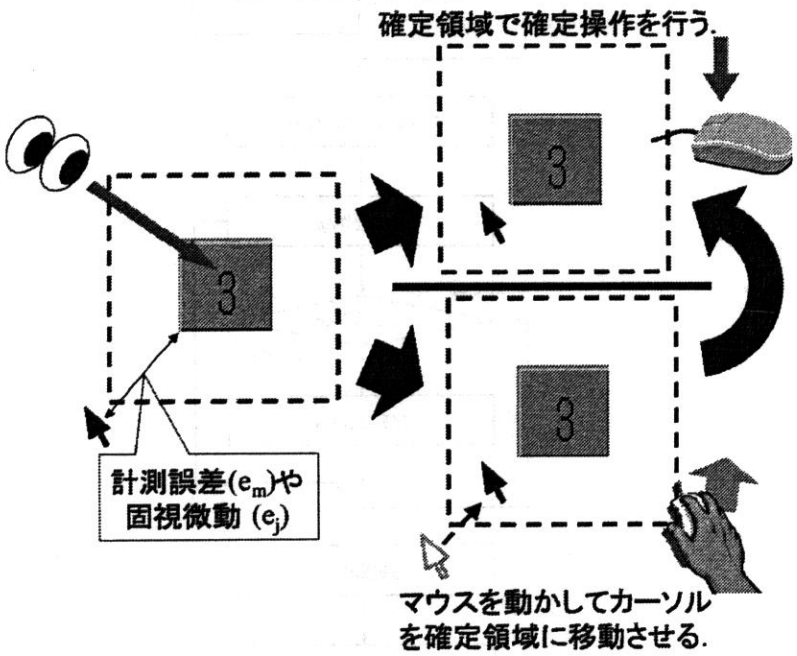

図 2.7 SemiAuto 方式

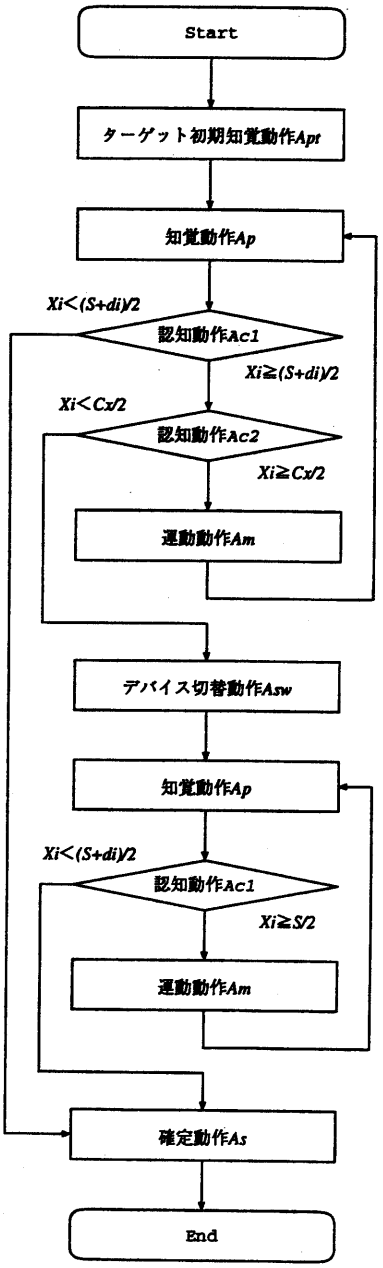

図 2.8 SemiAuto 方式によるターゲット選択の流れ

2.4. 評価実験

評価実験の目的は，3つの改良併用方式の有効性を一般的なGUI上で評価することである．

2.4.1 実験方法

操作方式

マウスのみを用いた操作方式(マウス方式)と，3つの改良併用方式(Auto方式，Manual方式，SemiAuto方式)を用いた．

タスク

Microsoft Windowsのデスクトップ上に開かれた1つのウィンドウ上に，一辺の長さが1cmの正方形のターゲットが9つ配置されている．この9つのターゲットの中から反転表示されているターゲットを選択する(図2.9)．反転表示されたターゲットは常に1つだけ存在し，ターゲットを1回選択するごとに反転するターゲットはランダムに変わる．このターゲット選択を50回行う．被験者に対しては,「選択はなるべく速くかつ正確に行うように.」との指示が実験前に与えられている．選択を誤った場合(表示が反転していないターゲットを選択した場合や，確定操作を行ってもどのターゲットも選択されない位置にカーソルがあるにもかかわらず確定操作を行った場合)には警告音を鳴らし，正しいターゲットが選べるまで選択操作を繰り返す．

なお，マウス方式に対しては，GUI上の複数のターゲットを連続的に選択する操作(連続操作)，及び，ターゲット選択以外の作業(キーボード入力など)を行った後に実施されるターゲット選択操作(非連続操作)の二通りの状況を想定したタスクを用意した．非連続操作を想定したタスクでは,「ユーザがターゲット選択に先だってカーソルをまず探す必要がある」という状況を想定し，一回のターゲットの選択ごとにウインドウ上のカーソル位置をランダムに再設定した．連続操作を想定したタスクでは，カーソル位置の再設定は行わなかった．なお，マウス方式以外の3方式(Auto方式，Manual方式，SemiAuto方式)では，カーソル位置の再設定は行っていない．これらの3方式では，カーソルは常に注視点の位置に設定され続けるため，カーソル位置の再設定を行うかどうかは実験結果に影響し

ないためである．

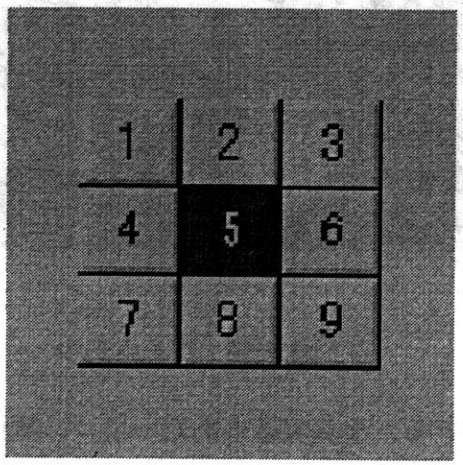

図 2.9 実験用ターゲット

ターゲットの大きさと配置

ターゲットの大きさは 1cm 四方である．9 つのターゲットを縦に 3 つ，横に 3 つづつ配置した．ターゲットの間の距離に応じて 4 つの実験用ウィンドウを用意した．ターゲット間の距離は，0cm(隣接)，1cm, 3cm, 5cm の 4 種類である．距離が 0cm になるようターゲットを配置したウィンドウを図 2.9に示す．

被験者

奈良先端科学技術大学院大学の教官と学生の計 5 名である．被験者はいずれも日常的に Microsoft Windows を使用しておりマウス操作には慣れている．

操作環境

実験には，21 インチディスプレイを用いる．ディスプレイの解像度は 1024 × 768 ピクセル，有効表示領域の大きさは縦 30cm 横 40cm である．被験者はディスプレイの前に座って操作を行うが，ディスプレイ画面から被験者の顔までの距離は約 50cm である．注視点の計測には，NAC 社製の非接触型アイカメラ EMR–NC [43] を用いた (図 2.10参照)．EMR–NC は被験者の視線だけでなく顔の位置も追跡する．顔がディスプレイ画面正面 50cm になるよう，被験者がディスプレイの前に座った場合，左右方向に 35cm，後方向に 50cm 被験者の顔が動いても注視点を

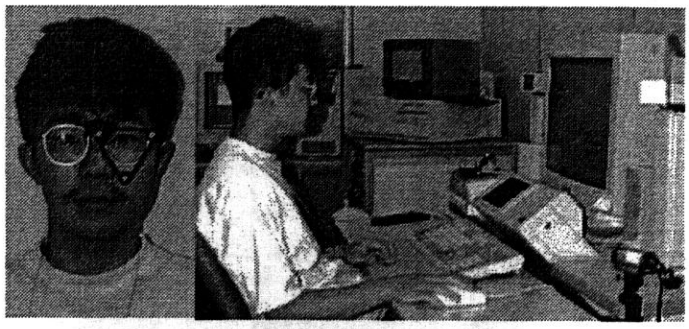

図 2.10 アイカメラ

計測をすることができる．なお実験にあたっては，画面上での計測誤差が 1cm 以内程度になるまで被験者毎にキャリブレーションを行なった．

被験者用計算機は Dell 社製 Dimension XPS R450 (CPU Pentium II 450MHz) である．OS は MS–Windows98 である．

計測データ

タスクが完了するまでの時間 (秒)，およびタスク実行中に発生した選択の誤り (エラー) の回数を収集した．

手順

タスクの実行に先だって，各操作方式ごとに 5 分間程度のタスクの練習を行った．

各被験者は，5 つの操作方式 (マウス (非連続選択)，マウス (連続選択)，Auto, Manual, SemiAuto) のそれぞれについて 4 つの実験用ウィンドウ (ターゲットの配置間隔はそれぞれ 0cm，1cm，3cm，5cm) を用いて，合計 20 回のタスクを実行した．タスクに対する被験者の慣れが実験結果に影響する恐れがあったので，次の 2 つの点に注意してタスクの実施順序を決めた．

- 実施する操作方式の順番を被験者毎に変える．
- 各操作方式に割り当てるターゲットの配置の順序を被験者毎に変える．

2.4.2 実験結果

改良併用方式間の比較

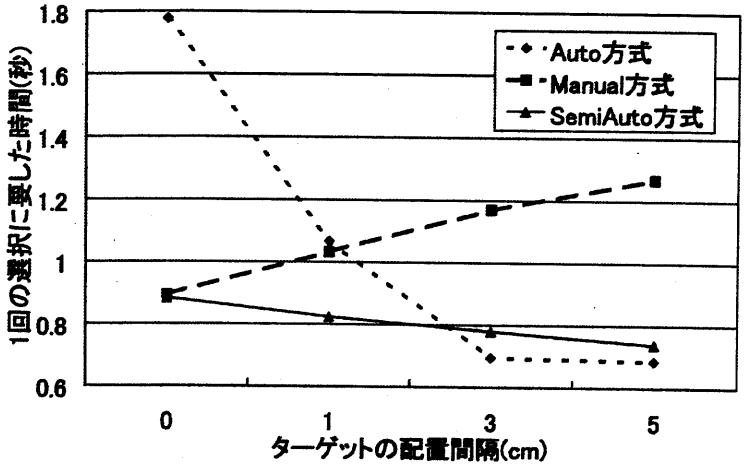

図 2.11 選択時間による改良併用方式間の比較

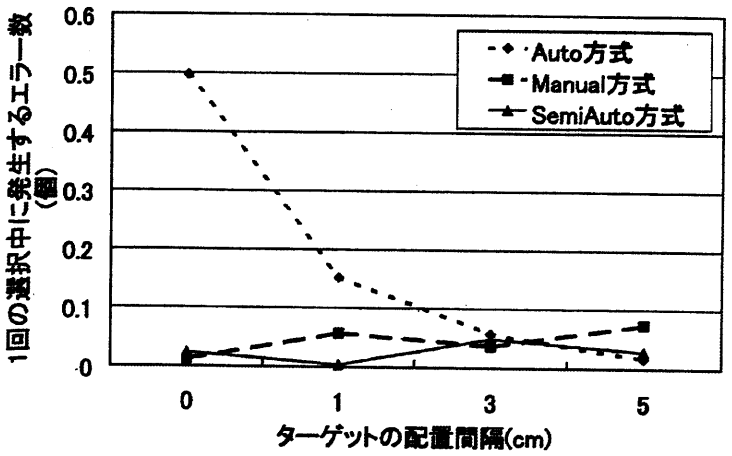

図 2.12 エラーの数による 改良併用方式間の比較

選択時間

ターゲットの平均選択時間とターゲット配置間隔の関係を図 2.11に示す.

Auto 方式では，ターゲットの配置間隔が広い (3cm 以上) 場合には，選択時間は小さい．しかし，配置間隔が狭くなると選択時間は大きくなった．特に，配置間隔が 0cm の場合，Manual 方式及び SemiAuto 方式による選択時間の 2 倍以上大きい (危険率 5 %で有意差あり)．配置間隔 0cm では，確定領域がターゲットのサイズと一致するため，選択が著しく困難になったと考えられる．

Manual 方式では，配置間隔が狭い (0cm) 場合には，選択時間は小さい．しかし，配置間隔が広くなるに従い大きくなった．特に，配置間隔が 5cm のとき，Manual 方式による選択時間は Auto 方式及び SemiAuto 方式による選択時間の 1.5 倍以上大きい (危険率 5 %で有意差あり)．

SemiAuto 方式では，配置間隔が広くにあるにつれて選択時間が小さくなった．配置間隔がさら広くなればさらに選択時間が小さくなる可能性がある．各配置間隔における選択時間は，0cm では Manual 方式と同程度であり，1cm では Auto 方式及び Manual 方式よりも小さく，3cm 以上では Auto 方式よりわずかに大きかった．

一般的な GUI 上ではターゲットの配置間隔が一様でないことを考えると，3 つの改良併用方式のなかでは，配置間隔に関わらず安定して短い時間で選択できた SemiAuto 方式が有望であると言える．

エラー数

ターゲットの平均エラー数とターゲット配置間隔の関係を図 2.12に示す.

Manual 方式と SemiAuto 方式の間には，大きな差はなかった (危険率 5 %で有意差なし)．Auto 方式はターゲットの配置間隔が狭くなるにつれエラーの数が増加した．特に，ターゲットの配置間隔が 0cm の場合に，Auto 方式を用いると Manual 方式及び SemiAuto 方式よりも 10 倍以上多くエラーが多く発生した (危険率 5 %で有意差あり)．

一般的な GUI 上では，ターゲットが隣接して配置される場合があることを考えると，Auto 方式は現実的でないと言える．

選択時間とエラーの数の結果をまとめると，3 つの改良併用方式のなかでは SemiAuto 方式がターゲットの配置間隔に依らず効率良くターゲットを選択でき

る方式だと言える.

マウス方式との比較

3 つの改良併用方式の中で，配置間隔に依らず安定して効率良く選択できた SemiAuto 方式を，マウス方式と比較する.

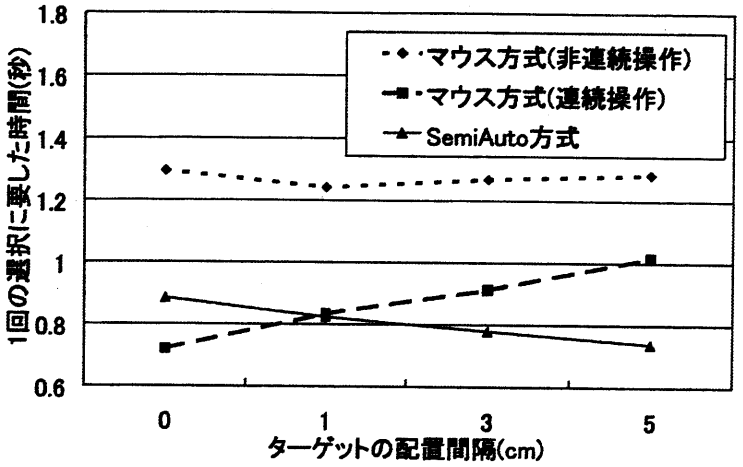

図 2.13 選択時間によるマウス方式と SemiAuto 方式の比較

選択時間

ターゲットの平均選択時間とターゲット配置間隔の関係を図 2.13に示す.

非連続操作を想定した場合，GUI ボタンの配置間隔に関わらず，SemiAuto 方式の選択時間はマウス方式の選択時間の約 3 分の 2 であった (危険率 5 %で有意差あり). 被験者の行動をビデオにより分析したところ，このような結果となった理由の一つが，マウス方式ではユーザがマウスカーソルを画面内から探し出す必要があるのに対して，SemiAuto 方式ではその必要がないためであることが分った.「マウスカーソルを画面内から探し出す」という動作は，ターゲット選択操作には本来不要なものであり，視線の利用がターゲット選択操作をより自然なものにした一例と考えることが出来る.

連続操作を想定した場合，配置間隔が 3cm 以上の場合には，SemiAuto 方式はマウス方式より短い時間で選択できた (危険率 5 %で有意差あり). 配置間隔が

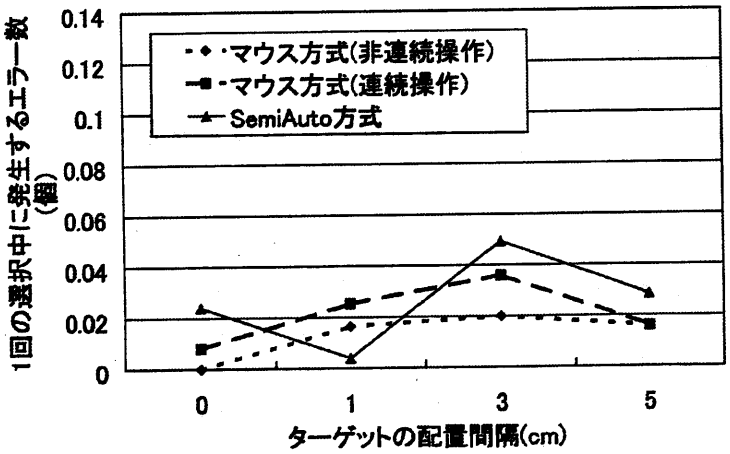

図 2.14 エラーの数によるマウス方式と SemiAuto 方式の比較

1cm 以上の場合には，SemiAuto 方式はマウス方式の選択時間は同程度であった (危険率 5 %で有意差なし)．配置間隔が 0cm の場合にのみ SemiAuto 方式による選択時間はマウス方式よる選択時間よりも約 0.2 秒大きかった．(危険率 5 %で有意差あり)．

配置間隔 0cm のターゲットを連続的に選択するという特殊な状況 (電卓ソフトウェアで計算を行う場合など) を除けば，SemiAuto 方式はマウス方式と同程度かより短い時間でターゲット選択を行うことができ，特に，非連続操作ではかなりの時間短縮が見込めると言える．

エラー数

ターゲットの平均エラー数とターゲット配置間隔の関係を図 2.14に示す．

マウス方式，SemiAuto 方式のどちらもエラーの数は少なく，一回の選択に発生するエラーの数は最大でも 0.05 回以下であった．また，ほとんどの場合において，マウス方式と SemiAuto 方式のエラー数に大きな差はなかった．連続操作を想定した場合には，全ての配置間隔において危険率 5 %で有意差はなかった．ただし，非連続操作を想定した場合には，配置間隔が 0cm 及び 3cm のときのみ，SemiAuto 方式がマウス方式よりもエラー数が多かった (危険率 5 %で有意差あり)．

SemiAuto 方式は，マウス方式と比較してエラーの数を大幅に増すことなくター

ゲット選択できると言える．

2.5. 従来研究との比較

従来よりポインティング操作を視線で行う方法が提案されてる．ただしそれらのインターフェースを一般的なGUIに適用するには問題がある．

1つ目の問題は，手でマウス等を操作して使う通常のインタフェースに比べ，非効率的なことである．久野らは，重度肢体不自由者のコミュニケーション支援装置のインターフェースとして視線を入力に用いることを提案している [15]．久野らの提案する方式では，ユーザは4秒間選択対象を注視することで，選択対象の選択を確定することができる．この方式は利用者が手を使うことができないという状況では，実用的である．しかしユーザが手を使うことができる場合には，選択に確定するために時間を要するので実用的ではない．今回私が比較検討した方式は，ユーザが手を使えることを想定して，マウスをクリックして確定操作を行うため，確定操作をより短時間で行うことができる．

2つ目の問題は，一般的なGUIを想定して評価を行っていないことである．Sibertらは，直径3cmの円形をしたターゲットを6cm間隔で配置したGUIを用いて，提案する視線インターフェースを用いると従来のマウスのみを用いるよりも高速にターゲットを選択できると述べている [26]．しかし，一般的なGUIでは，1cm四方程度の大きさのターゲットが隣接して配置されている場合が少なくない．一般的なGUI上でもSibertらの提案するインターフェースがマウスよりも高速であるかどうかは定かではない．ZhaiらのMAGIC [44] は私の比較検討したManual方式と同様に視線とマウスを併用してポインティング操作を行う方式である．しかしMAGICの評価にはターゲットを約6cm以上の幅をあけて配置したGUIを用いている．また選択中に発生するエラーの数については述べられていないため，一般的なGUIへの適用可能性が不明である．

2.6. まとめ

本章では，一般的なGUI上で効率良くターゲット選択操作を行う方式として，視線とマウスを併用する「Auto方式，Manual方式，SemiAuto方式」の3つの方式を比較検討した．

一般的なGUIを想定した環境で評価実験を行った結果，SemiAuto方式は，従来のマウスのみを用いた操作方式に比べてほぼ同程度かより効率が良いことがわかった．特に非連続操作においては，選択誤りを増すことなく，ターゲット選択時間を約3分の2に短縮できた．

従来研究においても，視線を利用したターゲット選択方式は数多く提案されてきたが，一般的なGUI環境での使用を想定した方式はほとんどなかった．これに対し，本章では，一般的なGUI上におけるターゲット選択操作モデルを定義した上で，妥当な性質を持った3つのターゲット選択方式(改良併用方式)を比較検討し，その有効性を実験により確認した．

今後の課題は，カーソルの表示方式の検討である．評価実験では，ユーザの視線上(画面上の注視点)にカーソルを常に表示した．しかし，通常のアプリケーションの利用において，カーソルを常に表示すると画面の可視性が低下する可能性がある．例えば，文献[44]では，ユーザが手でマウスを動かしたときに初めてカーソルを出現させることでこの問題を回避している．別の解決策としては，ユーザがターゲット近傍を見ている場合にのみカーソルを表示する方法が考えられる．

第3章

視線とマウスを併用したドラッグ&ドロップ操作

3.1. あらまし

本章では，MS–Windows や MacOS などの一般的な GUI 上で日常的に行われる操作のうち，ドラッグ&ドロップ操作を対象とした視線追加型インタフェースについて述べる [39]．従来は，視線を用いるために特殊化された GUI 上での視線追加型が主に研究されてきたが [10,24]，現在普及している一般的な GUI 上での視線追加型インタフェースが実現できれば，既存のアプリケーションソフトウェアの GUI を新規に作り直す必要なく操作を効率化できるため，多くの計算機ユーザにとってより有用であると考えられる．一般的な GUI 上の操作には，アイコンやメニューなどのターゲット選択，ドラッグ&ドロップ，ウィンドウのスクロール，ウィンドウの選択など多数あるが，従来研究では，主にターゲット選択操作を対象とした視線追加型インタフェースが提案されてきた [24,44]．一般的な GUI 上の操作をさらに効率化するためには，ターゲット選択操作以外の GUI 操作に対しても視線の適用範囲を広げていくことが重要である．ドラッグ&ドロップ操作は多くの計算機ユーザが日常的に行う GUI 操作の一つであり，かつ，従来研究されてきたターゲット選択操作との類似点が多いことから，本章での研究対象に選んだ．

視線用に特殊化された GUI と比較すると，一般的な GUI での Gaze-Added Interface の実現は容易でない．視線用に特殊化された GUI では大きな GUI 部品（アイコン，メニューアイテム等）が広い間隔で設置されているのに対し，一般的な GUI では，小さな GUI 部品が狭い間隔で配置されているため，次に示す視線追跡装置の計測誤差の問題，および，固視微動の問題を克服することが困難となる．

視線追跡装置の計測誤差 ユーザが画面上の見ている場所 (注視点) を正確に特定することは，現在の視線追跡装置では困難である．視線追跡装置の測定値とユーザが実際に見ている画面上の場所との間には 0.9cm～1.7cm 程度のずれがある (眼球から 50cm 前方に計算機ディスプレイが位置する場合) [20,31].

固視微動 眼は常に微動している．注視点は 0.4cm 四方の範囲で細かく振動している．視線で一点を指示し続けることはできない (眼球から 50cm 前方に計算機ディスプレイが位置する場合) [7].

本章では，計測誤差と固視微動の問題を克服するために，2 章で提案した 3 つのターゲット選択方式のうち最も効率の良かった SemiAuto 方式をドラッグ&ドロップ操作に応用する．SemiAuto 方式は，狭い間隔で配置された小さなターゲットを正確にかつ効率良くポインティングできるように，ターゲットの近傍でのカーソル位置を自動的にもしくは手動で微調整できる．本章で提案する方式は，ドラッグ&ドロップ操作を，ドラッグの対象となるアイコン（ドラッグアイコン）とドロップ先のアイコン（ドロップアイコン）の選択という連続する 2 回のターゲット選択操作であるとみなし，それぞれの選択操作に対して SemiAuto 方式を適用する．これにより，各選択操作を正確にかつ効率良く行うことが可能となる．

本章では，提案する方式を一般的な GUI 環境に似せた環境で行った評価実験の結果についても述べる．詳しくは **3.4**で述べるが，従来提案されている Gaze-Added Interface は，GUI 部品の配置間隔を広くとるなどした特殊化したインタフェース上でしか評価されておらず，一般的な GUI での有効性が明らかにされていない．本章では評価実験により，一般的な GUI での提案方式の効率と正確性を調べる．

以降，3.2で方式を提案する．3.3で提案方式の評価実験について述べる．3.4で関連する研究について述べ，3.5でまとめる．

3.2. 提案方式

提案方式について述べるにあたり，まず3.2.1において，従来のマウスのみによるドラッグ&ドロップ操作について整理する．3.2.2 では，ドラッグ&ドロップ操作に視線を割り当てた単純な方式（基本方式）について述べる．この基本方式は提案方式の基本となる考え方を示すためのものであり，視線追跡装置と固視微動の問題に対して特別な工夫を行っていない．次に3.2.3において，基本方式をベースにしてSemiAuto方式 [38] を応用した方式(拡張方式)について述べる．この拡張方式が本稿における提案方式である．

3.2.1 マウスのみを用いたドラッグ&ドロップ操作

ドラッグ&ドロップ操作は次に示す4つのより細かい操作の系列である [12]．

1. ポインティング操作1: ドラッグアイコンへカーソルを移動する操作(図3.1左上参照)
2. プレス操作: マウスボタンを押してドラッグアイコンの選択を確定する操作(図3.1左下参照)
3. ポインティング操作2: ドロップアイコンへカーソルを移動する操作(図3.1右上参照)
4. リリース操作: マウスボタンを離してドロップアイコンの選択を確定する操作(図3.1右下参照)

以降では，プレス操作とリリース操作をまとめて確定操作と呼ぶこととする．

3.2.2 基本方式

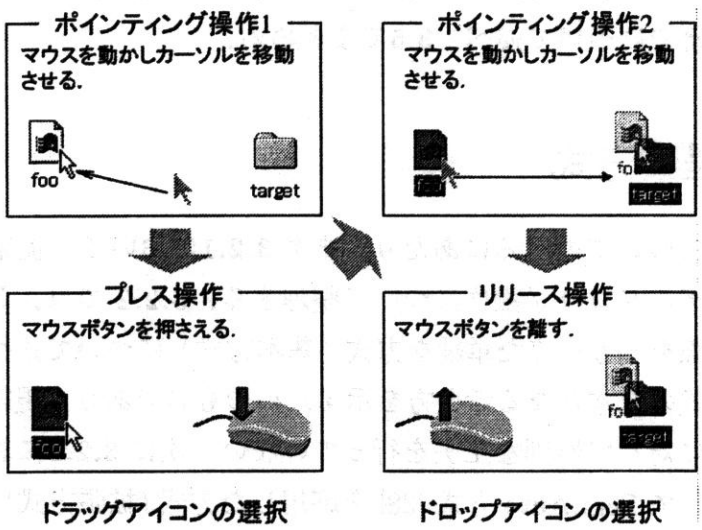

図 3.1 マウスのみを用いたドラッグ&ドロップ操作

基本方式では，ドラッグ&ドロップ操作を構成する基本操作のうち，ポインティング操作を視線で行う．カーソルはユーザの画面上の注目箇所 (注視点) に追従し移動させる．確定操作は従来通りマウスボタンで行う (図 3.2 参照)．従ってユーザは次に示す細かい操作の連続によってドラッグ&ドロップ操作を行うことができる．

1. ポインティング操作 1: 選択したいドラッグアイコンを見る (図 3.2左上参照)．
2. プレス操作: マウスボタンを押してドラッグアイコンの選択を確定する (図 3.2左下参照)．
3. ポインティング操作 2: 選択したいドロップアイコンを見る (図 3.2右上参照)．
4. リリース操作: マウスボタンを離してドロップアイコンの選択を確定する (図 3.2右下参照)．

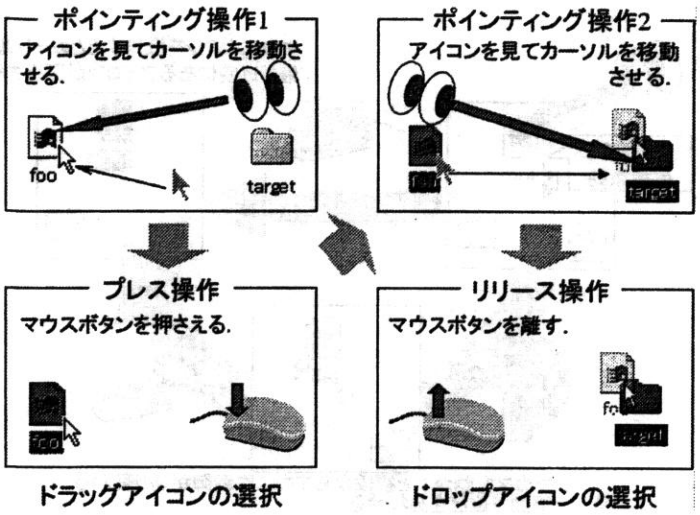

図 3.2 基本方式

3.2.3 SemiAuto 方式による拡張

拡張方式では，視線追跡装置の計測誤差や固視微動があってもカーソルを正確にアイコンへ移動させることができるように，カーソル位置の微調整を行う．

微調整は 1 回のドラッグ&ドロップ操作において 2 回行う．ドラッグアイコン選択操作時 (ポインティング操作 1 とプレス操作) とドロップアイコン選択操作時 (ポインティング操作 2 とリリース操作) の 2 回である．微調整には 2.3.3においてターゲット選択方式として提案した SemiAuto 方式を用いる．ドラッグアイコンの選択とドロップアイコンの選択を 2 回のターゲット選択操作であるとみなし，各選択において SemiAuto 方式によりカーソル位置の微調整を行う．

SemiAuto 方式による微調整を図 3.3に示す．計測誤差あるいは固視微動によりユーザが目的のアイコン上へカーソルを移動させることが困難であると感じた場合，確定操作を行うかあるいはマウスを動かすことで微調整を行うことができる．確定操作を行った場合，カーソルをカーソルから最も近傍にあるアイコン上へ移動させた後に確定する (図 3.3 矢印 1)．マウスを動かした場合は，カーソル位置の制御を視線からマウスに切り替える．ユーザは手でマウスを動かしてアイコン上にカーソルを移動させることができる (図 3.3矢印 2A)．カーソル位置の制御をマウスに切り替えた後に確定操作を行った場合にも，カーソルをカーソルから最

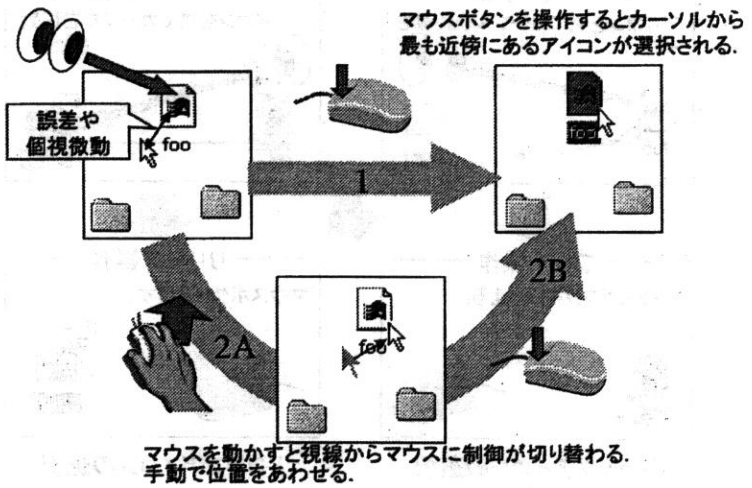

図 3.3 SemiAuto 方式 (ドラッグアイコンの選択の場合)

も近傍にあるアイコン上へ移動させた後に確定する (図 3.3 矢印 2B).

SemiAuto 方式のユーザは次に示す細かい操作の連続によって一般的な GUI 上でも効率良くドラッグ&ドロップ操作を行うことができる.

1. おおまかなポインティング操作1: 選択したいドラッグアイコンを見る. カーソルはドラッグアイコンに近づく.
2. 手動によるカーソル位置の微調整: 必要と感じた場合, マウスを手で動かして目的のドラッグアイコンへカーソルを合せる.
3. プレス操作: マウスボタンを押してドラッグアイコンの選択を確定する.
4. おおまかなポインティング操作2: 選択したいドロップアイコンを見る. カーソルはドロップアイコンに近づく.
5. 手動によるカーソル位置の微調整: 必要と感じた場合, マウスを手で動かして目的のドロップアイコンへカーソルを合せる.
6. リリース操作: マウスボタンを離してドロップアイコンの選択を確定する.

ここで「必要と感じた場合」とは，例えば視線により移動したカーソルの近傍に複数のアイコンがあり，選択したいアイコンがカーソルから最も近傍にあることが明かでない場合を指す．このような場合，確定操作によって複数あるアイコンのうちどのアイコンにカーソル位置が微調整されるかわかりにくいため，手でマウスを動かして目的のアイコンへカーソル位置を調整する必要がある．

提案方式はカーソルのおおまかな移動に視線を用いるため，マウス方式と比べてマウスの移動量が少なくて済む．従って，特にマウスを動かすことに慣れていない初心者に有効であると考えられる．

3.3. 実験

提案方式の有効性を確認するため，一般的なGUIに似せた環境で提案方式と従来のマウスのみによる操作(マウス方式)の効率を比較する実験を行った．また提案方式の利用者として，マウス方式の経験者だけでなく初心者も想定してその有効性を調べた．

3.3.1 実験方法

タスク: 実験用ウィンドウ(図3.4)に配置されたアイコンの中から，ドラッグ元グループ中の表示が反転しているアイコンをドラッグして，ドロップ先グループ中の表示が反転しているアイコンへドロップする．ドラッグ元グループ，ドロップ先グループにはそれぞれ9つのアイコンからなる．反転しているアイコンは各グループにつき1つだけである．

被験者は，10回のドラッグ&ドロップ操作を連続して行う．反転するアイコンは1回の操作が成功するたびにランダムに変わる．10回のドラッグ&ドロップ操作を成功させることを1セットとし，以降で述べる実験条件(2種類のアイコンの配置間隔，2つの操作方式，実験に用いる左右の手)毎に，5セットづつタスクを実行した．

なお試行の前には，操作方式に慣れてもらうために，操作方式毎に10分程度の練習を行った．

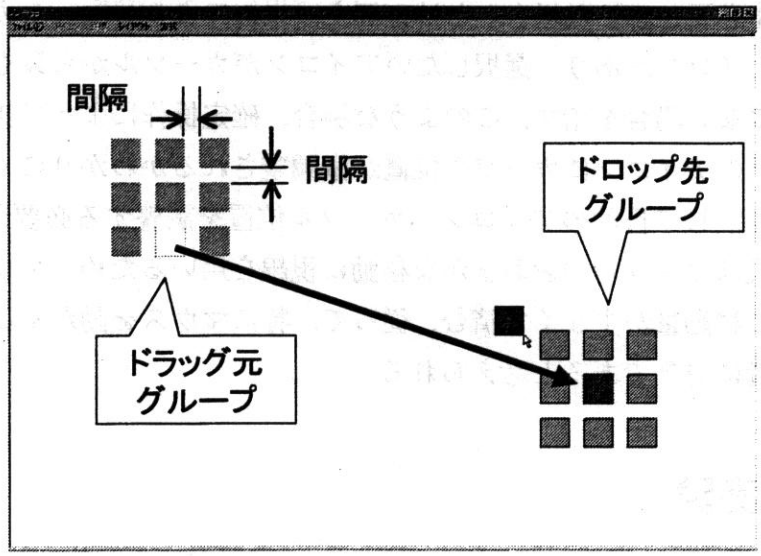

図 3.4 実験用ウィンドウ

被験者: 日常的にマウスを用いて MS-Windows を利用している大学院生 3 名を被験者とした．従って，それぞれの実験条件ごとにのべ 150 回のドラッグ&ドロップ操作が行われた．なお，被験者は 3 名とも右利きであった．

操作環境: 実験には，解像度を 1024 × 768 ピクセルに設定した 21 インチディスプレイを用いた．有効表示領域の大きさは縦 30cm 横 40cm である．顔とディスプレイとの距離が約 50cm となるように被験者はディスプレイの前に座った．

MS-Windows のデスクトップ上のアイコンとほぼ同じ大きさになるように，アイコンの大きさは，画面上で 1.4cm 四方の正方形とした．またアイコンの配置は，0.7cm 間隔 (狭い) と 1.4cm 間隔 (広い) の 2 種類を用意した．ドラッグ元グループの中心に配置されたアイコンとドロップ先グループの中心に配置されたアイコンの距離は画面上で 24.5cm である．

操作方式: マウスのみを使った操作方式（マウス方式）と提案方式の 2 種類を用いた．それぞれの操作方式に対して，利き手である右手と利き手でない左手を用いてタスクを実行してもらった．左手による操作は初心者による操作を想定したものである．本実験では，マウス方式に不慣れな初心者（マウスをほとんど使ったことのない人）を確保することができなかったため，次善の策として，経験者

の左手によるマウス方式の操作を初心者の操作の近似とした．以降で述べるとおり，経験者の左手を用いたとしても，初心者に対する提案方式の有効性を不当に高く見積もることにはならない．

Kabbash らの分析 [11] によると，アイコン近傍までカーソルを大まかに移動させる操作は，経験者の左手の方が初心者の利き手よりも効率がよい．一方，アイコン近傍からアイコンの真上へカーソルを動かす微調整の操作に限っていえば，経験者の左手と初心者の利き手はほぼ同程度の操作効率である．提案方式では，アイコン近傍までカーソルを大まかに移動させる操作には（手ではなく）視線を用いるため，経験者の左手を用いて実験したとしても，「提案方式を用いた初心者」を想定した操作時間を不当に短くすることは避けられる．

計測データ: 10 回のドラッグ&ドロップ操作を 1 セットとして，1 セットに要した操作時間（秒）および試行中に発生したアイコンの選択誤り (エラー) の回数を収集した．

3.3.2 実験結果

各操作方式ごとに右手および左手による 1 回の試行に要した時間とエラー数の平均を求めた．以降では，右手で操作を行った場合，および，左手で操作を行った場合のそれぞれについての実験結果を述べる．

右手 (経験者による操作)

右手を用いた場合の 1 回の操作時間を図 3.5に，1 回のエラー発生回数を図 3.6に示す．

アイコンの配置間隔によらずマウス方式よりも提案方式による操作時間は短かった．配置間隔が 0.7cm の場合は 17%(0.41 秒)，1.4cm の場合は 19%(0.45 秒)，提案方式の方がマウス方式よりも短かった．

エラーの発生回数については，アイコンの配置間隔によらず提案方式の方がマウス方式より多くエラーが発生した．ただし，その差は小さかった．配置間隔が 0.7cm の場合 0.0060 回，1.4cm の場合 0.014 回，提案方式の方がマウス方式よりも多かった．

提案方式とマウス方式の操作時間の母平均の差を統計的に検定した．その結果，「提案方式による操作時間がマウス方式による操作時間よりも短い」，と有意水準

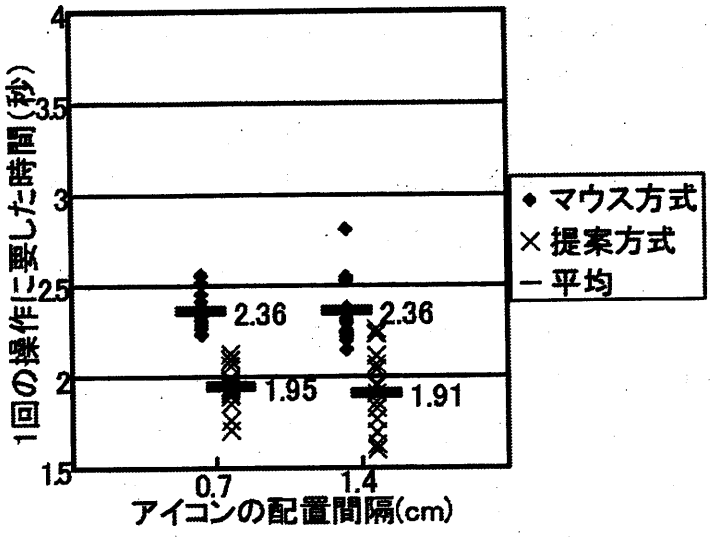

図 3.5 操作時間 (右手:経験者)

5%で言えることが分った．エラーの回数についても，操作時間と同様に検定を行った結果，提案方式によるエラーの回数とマウス方式によるエラーの回数には有意な差はないことがわかった．

左手 (初心者を想定した操作)

左手を用いた場合の 1 回の操作時間を図 3.7に，1 回のエラー発生回数を図 3.8に示す．

左手を用いた場合，アイコンの配置間隔に関係なく，マウス方式よりも提案方式の方が操作時間が短かった．配置間隔が 0.7cm の場合 27%(0.80 秒)，1.4cm の場合 35%(1.05 秒)，提案方式の方がマウス方式よりも短かった．提案方式は，右手による操作においてもマウス方式より効率が良かったが，左手による操作ではマウス方式との差がさらに顕著に表れた．実際の初心者は，熟練者の左手よりも大まかなカーソルの移動が遅いため [11]，マウス方式との差がさらに広がる可能性がある．

エラーの発生回数についても，アイコンの配置間隔によらず提案方式の方がマウス方式よりも少なかった．配置間隔が 0.7cm の場合 0.020 回，1.4cm の場合 0.074 回，提案方式の方がマウス方式よりも少なかった．

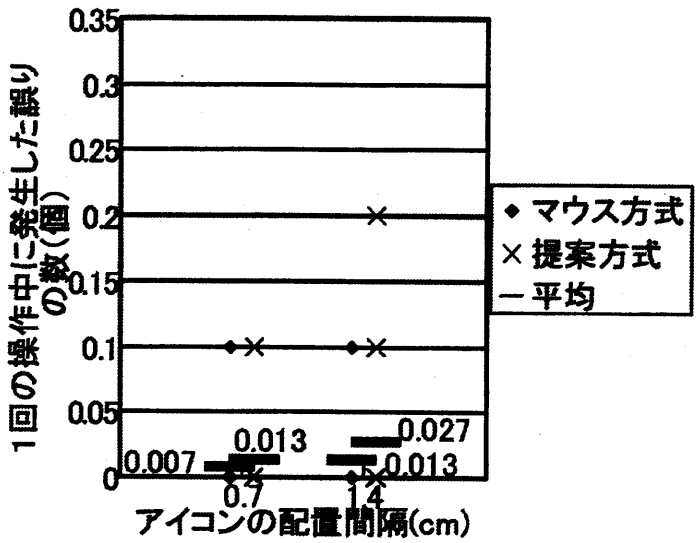

図 3.6 エラーの数 (右手:経験者)

左手による操作時間についても，右手と同様に検定を行った結果，提案方式による操作時間がマウス方式による操作時間よりも短い，と有意水準 5%で言えることが分った．エラーの回数については，アイコンの配置間隔が 1.4cm の場合には，提案方式によるエラーの回数は，マウス方式によるエラーの回数よりも少ないと有意水準 5%で言えることが分った．アイコンの配置間隔が 0.7cm 場合には，SemiAuto 方式とマウス方式には有意な差が無いことがわかった．

以上の結果より提案方式は従来のマウスのみによる操作に比べて，エラーの数を大幅に増やすことなく操作時間を短縮できることがわかった．特にマウスを動かすことに慣れていない初心者を想定した場合には，操作時間が平均で 27%短縮され，エラーの数も減る可能性があることが分かった．

3.4. 従来研究との比較

従来より，主にターゲット選択を対象とした Gaze-Added Interface の研究が行われて来た．Salvucci らはターゲット選択操作を視線とキーボードを併用して効率良く行う方式を提案している [24]．また Zhai らは視線とマウスを併用して

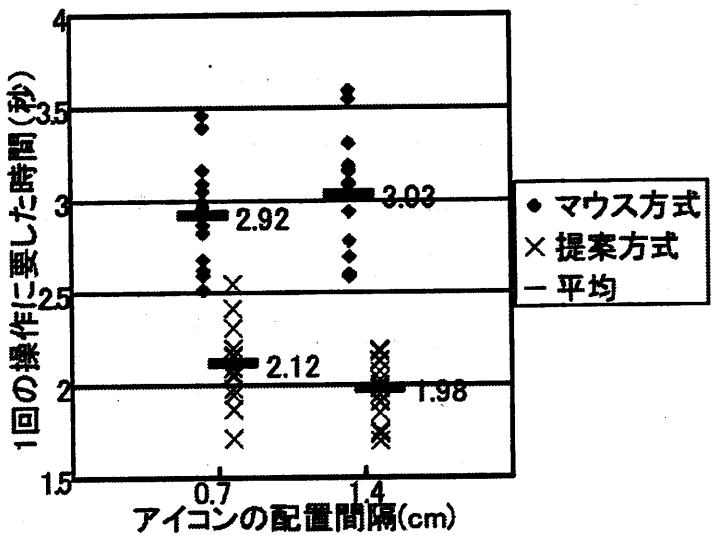

図 3.7 操作時間 (左手:初心者)

ターゲット選択操作を効率良く行う方式を提案している [44]. これらの操作方式は，一般的な GUI で用いられている GUI 部品よりもサイズの大きなものを用いる，あるいは，GUI 部品の配置間隔を広くとるなどした特殊化した GUI 上での操作効率の向上を目指している．また，そのような特殊化した GUI 上で方式の評価が行なわれている．一方，本章では，ターゲット選択ではなくドラッグ&ドロップ操作を対象としており，一般的な GUI を想定した環境において提案方式の評価を行った.

Gaze-Centered Interface においても多くのターゲット選択操作方式が提案されているが，それらはドラッグ&ドロップ操作に応用するのには適していない. [5] ではターゲットを一定時間見続けることで選択できる方法が提案されている. [23] ではターゲットを見て眉間の筋肉を動かすことで選択できる方法が提案されている. [25] ではターゲットを見て瞬きすることで選択できる方法が提案されている. しかし，ドラッグ&ドロップを 2 回のターゲット選択であるとして，これらの手法を用いてドラッグ&ドロップを行なおうとすると，ドラッグ操作が困難になると考えられる. [5] を用いる場合，ドラッグアイコンを見続ける必要があるため，ドロップアイコンを見ることができない. [23] を用いる場合，ユーザはドラッグ

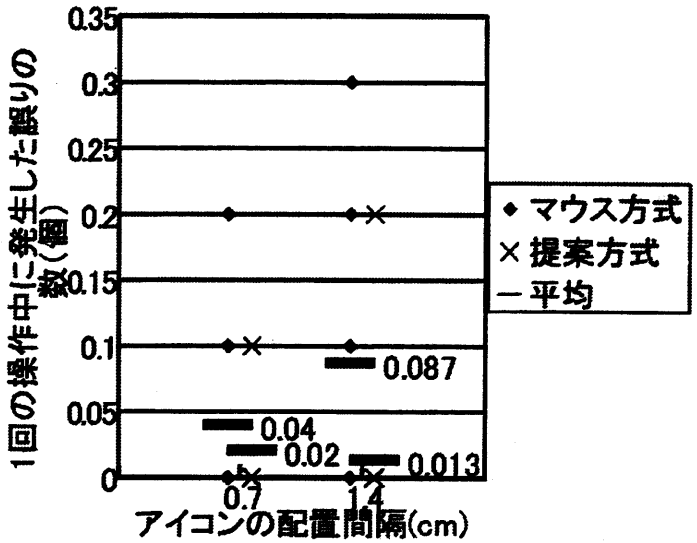

図 3.8 エラーの数 (左手:初心者)

操作を行なっている間，筋肉を緊張させ続ける必要があるためユーザに負担となる恐れがある．[25] を用いる場合には，ドラッグ操作を行なっている間，目を閉じておく必要があるため，選択したいドロップアイコンを見ることができない．私の提案方式は，マウスを併用する必要はあるが，Gaze-Centered Interface では困難なドラッグ&ドロップ操作を可能とした．

3.5. まとめ

一般的な GUI 上で頻繁に行われるドラッグ&ドロップ操作を視線とマウスを併用して効率良く行う方式を提案した．提案方式は，ドラッグ&ドロップ操作をドラッグアイコンの選択とドロップアイコンの選択の 2 回の連続したターゲット選択操作であるとみなし，すでにターゲット選択方式として提案されている SemiAuto 方式 [38] を適用した．SemiAuto 方式によりカーソル位置の微調整を行うことで，小さなアイコンが狭い間隔で配置されているような一般的な GUI 上においても，ドラッグ&ドロップ操作を正確にかつ効率良く行うことができる．

一般的な GUI を想定した環境で提案方式の評価実験を行った．その結果，提

案方式による操作は従来のマウスのみによる操作に比べて，エラーの数を大幅に増やすことなく，1回のドラッグ&ドロップ操作について平均で約17%(0.4秒)短縮できることがわかった．また，マウスの利用初心者を想定した実験では，操作誤りも減り，平均で約27%(0.8秒)短縮できることがわかった．

今後の主な課題は2つある．1つは，提案方式の評価をさらに進めることである．本章における評価実験では，ドラッグアイコン群をデスクトップ左上に設置し，ドロップアイコン群をそこから右下方向へ平均24.5cm離れた位置に設置して実験を行った．アイコン群の配置方向および配置距離が異なる場合における評価を行うことが今後の課題となる．

もう1つは，アイコンとは形状が異なるGUI部品を対象としてドラッグ&ドロップを操作を行なう場合における提案方式の改良と評価を行なうことである．例えばウィンドウのタイトルバーは垂直方向に比べて水平方向に長い形状を持つ．この形状の特徴を利用して，タイトルバーの移動に特化したカーソル位置の微調整方式について検討する．

第4章

視線によるテキストウィンドウの自動スクロール

4.1. あらまし

計算機画面や画面内のウィンドウに何らかの情報を表示するソフトウェア（テキストエディタ，ウェブブラウザ等）の多くは，ユーザからの入力に応じて表示内容をスクロールする機能を持つ．スクロール機能は，同時に表示可能な情報量を増やすことは出来ないが，大きさや解像度に制限のある計算機画面やウィンドウで，より大量の情報を扱うことを可能にする．例えば，テキストエディタが同時に表示可能な行数は，一般に，高々数十行であるが，数百行，数千行のプログラムファイルを編集することがスクロール機能により可能となっている．

表示内容をスクロールさせるには，通常，キーボードからスクロールのためのコマンドを入力したり，マウスを操作してスクロールバーのクリックやドラッグを行う必要がある．しかし，情報を読み書きしている最中にコマンド入力やマウス操作を頻繁に行うと，ユーザの思考が中断され，作業効率の低下やユーザへの高い負荷を招く恐れがある．ページ単位でのスクロールや指定した文字列の存在箇所までスクロールさせるコマンドによって，スクロール操作の回数や時間を短くすることは可能であるが，同時に，スクロール操作をより繁雑にする．

本章では，キーボードやマウスによる操作を必ずしも行わなくても，ユーザが読

み書きしたいと考える情報がウィンドウ内に表示されるインタフェースの実現を目指して，視線によるテキストウィンドウの自動スクロール方式を提案する [34]. 具体的には，視線追跡装置によって作業者の視線を計測し，視線とテキストウィンドウとの交点（注視点）の座標に基づいて縦スクロールの方向と速度を決定する．注視点がテキストウィンドウ上部にある場合には下方向へのスクロールを，下部にある場合には上方向へのスクロールを行う．また，スクロールの速度（あるいは加速度）は，ウインドウの垂直方向の中心と注視点との距離に比例させる．これにより，ウィンドウ内に読みたいと思うテキストが存在する場合はもちろんのこと，ウィンドウ外にテキストがある場合でも，ユーザはウィンドウ内で視線を移動させるだけで，テキストをウィンドウ中央部に表示させ（移動させ），スクロールを停止させることが出来る．

キーボードやマウスの代わりとして視線を入力とするインタフェース（視線インタフェース）の研究は盛んに行われている [3,10,19,27,28]．その多くは視線によるメニュー選択に関するものであるが，Jacob は，視線によってテキストのスクロールを制御する方式を提案している [10]．彼が提案する方式では，予め用意された特定のアイコンに視線が向けられるとスクロールが開始し，視線がテキストに戻るとスクロールは停止する．従って，ポインティングデバイスによる入力に代るものとして視線を利用していると言える．アイコンへの視線の移動は，表示されているテキストから一時的にではあるが視線を外すことを意味し，作業効率を低下させる可能性が大きい．これに対して，本章で提案する方式では，ウィンドウ内で視線を移動させるだけでスクロールを制御することができ，表示されているテキストから視線を外す必要はない．特に，テキストを先頭から順に読む作業においては,「読む」という動作に追従して自動的にテキストがスクロールすることになる．

本章の以降の構成は次の通りである．**4.2**では，視線による自動スクロールの概略を示し，必要な機能を列挙する．**4.3**では，ウィンドウ上の注視点座標をパラメータとして，スクロールの向きや速度を制御する具体的な方式を 4 つ提案する．**4.4**では，視線による自動スクロール機能を持つシステムとして試作したテキストブラウザについて述べる．**4.5**では，試作したテキストブラウザを用いて行った自動スクロール機能の評価実験について説明し，その結果を示す．**4.6**で

は，まとめと今後の課題を述べる．

4.2. 視線による自動スクロール

視線による自動スクロールとは,「コンピュータユーザが作業対象中の見たい，あるいは，読みたいと考える部分が表示用ソフトウェアの情報表示部（ウィンドウ）の中心に表示される様，ユーザの注視点の座標に基づいて，表示用ソフトウェアのスクロール制御を自動的に行う」ことである．ここで,「注視点」とはユーザの視線とウィンドウとの交点を意味する.「スクロール制御」とは，スクロールの方向，及び，速度を決定することである．また,「作業対象」としては，プログラムリスト，ワープロ文書，画像等を,「表示用ソフトウェア」としては，テキストエディタ，ウェブブラウザ等が考えられる．

本節では，簡単のため，作業対象としてプログラムリストを，表示用ソフトウェアとしてはテキストのみ表示可能なソフトウェア（テキストブラウザ）を，それぞれ考える．また，スクロールの方向は，縦方向のみを考える．従って，ウィンドウの中心とは，テキストブラウザの情報表示部（テキストウィンドウ）の垂直方向の中心線（以降では，単に「中心線」と呼ぶ）であり，この中心線と注視点の距離，即ち，注視点の垂直座標の値（変位）によってスクロールの方向と速度が決定されることになる．

視線を向けるという自然な動作によって，見たい，あるいは，読みたいと考えるテキスト（目標テキスト）をウィンドウの中心へと移動させるためには，注視点が中心線より上部にある場合には下方向に，下部にある場合には上方向に，それぞれスクロールさせれば良い（図 4.1参照）．目標テキストがウィンドウ内にある場合，目標テキストにユーザが視線を向けるとスクロールによって目標テキストは中心線に近づき，それをユーザが目で追うことにより注視点も中心線に近づく．比較的短い時間で，目標テキストと注視点が共に中心線上に位置し，スクロールは停止する．目標テキストがウィンドウ外にある場合,（スクロールが停止しないように）中心線より少し離れたウィンドウの上部，あるいは，下部にユーザが注視点を留めておけば，スクロールによって目標テキストがウィンドウ内に現れる．現れた目標テキストをユーザが見つけ，視線を向け続ければ，目標テキ

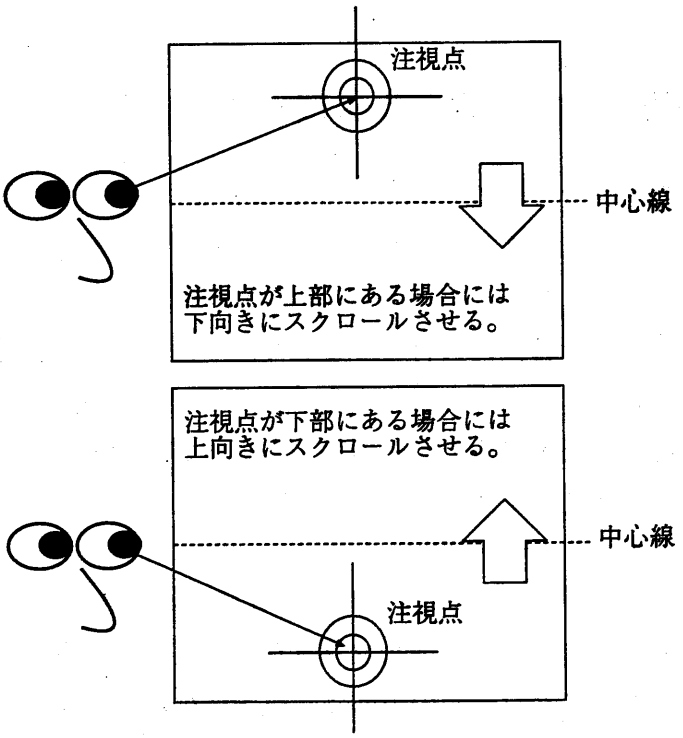

図 4.1 視線による自動スクロール

ストと注視点はやはり中心線上に移動し，スクロールは停止する．

「視線による自動スクロール」の利用者からすれば，次のように視線を移動するだけで，見たい，あるいは，読みたいと考えるテキストがウィンドウ中央に表示されることになる．

- 目標テキストをウィンドウ内に見つけた場合には，目標テキストに視線を向ける．
- 目標テキストがウィンドウ内に見つからない場合は，目標テキストがウィンドウ内に現れるまで，ウィンドウの上部，または，下部に視線をとどめる．

「視線による自動スクロール」を実現するためには，次の 4 つの機能が必要となる（図 4.2参照）．

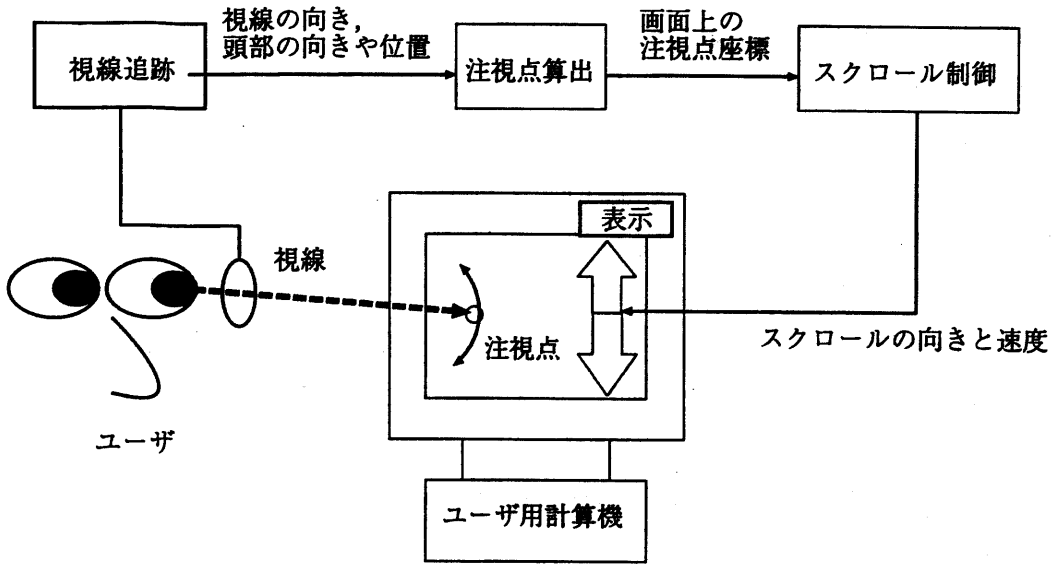

図 4.2 自動スクロールに必要な機能

(1) **視線追跡機能**…ユーザの視線の向きを精密に測定する．ユーザの頭部を固定しない場合には，頭部の向きや位置も測定する必要がある．スクロールを円滑に行い，停止させるためには，測定誤差が 1〜2 度以下で，1 秒あたり 10 回程度の測定性能は必要である．

(2) **注視点算出機能**…ウィンドウの位置，及び，「(1) 視線追跡機能」により得られた視線の向きとユーザの頭部の向きや位置から，注視点を算出する．

(3) **スクロール制御機能**…「(2) 注視点算出機能」により得られた注視点の座標に基づいて，スクロールの向きと速度を決定する．

(4) **表示機能**…作業対象を表示し，「(3) スクロール制御機能」で決定されたスクロールの向きと速度に基づいて実際にスクロールを行う．

4.3. スクロール制御方式

4.2で述べたように，視線を向ける目標テキストをウィンドウの中心へと移動させるためには，注視点が中心線より上部にある場合には下方向に，下部にある場

合には上方向に，それぞれスクロールさせれば良い．

スクロール速度の制御では，

(1) 目標テキストのウィンドウの中心への迅速な移動

(2) スクロールのスムーズな停止

の2点が重要である．本節では，(1) を実現する制御方式として「速度方式」と「加速度方式」の2方式を提案する．更に，(2) を実現するために「速度3分割方式」と「加速度3分割方式」の2方式を追加し，合計4つの制御方式を提案する（表4.1参照）．なお，名称に統一性を持たせるため，以降では「速度方式」を「速度2分割方式」と，「加速度方式」を「加速度2分割方式」と，それぞれ呼ぶ．

4.3.1 速度2分割方式と加速度2分割方式

目標テキストをウィンドウの中心へすばやく移動させるには，中心線と注視点の距離，即ち，注視点の垂直座標の値（変位）が大きくなるほど，スクロール速度を大きくすればよい（図4.3参照）．「速度2分割方式」は，スクロール速度を変位に比例させる制御方式である．スクロール速度の算出式は次のとおりである．

$$v(t) = -M_v e(t) \tag{4.1}$$

ここで，$e(t)$ は時刻 t における注視点の変位（単位はページ）を，$v(t)$ は時刻 t におけるスクロール速度（単位はページ/秒）を，それぞれ表す．ページとはテキストウィンドウの縦方向の長さである．また，M_v は最大速度を調整するための比例定数である．なお，中心線よりウィンドウ下方向の変位や速度は正の値で，上方向の変位や速度は負の値で，それぞれ表すものとする．

「速度2分割方式」では，注視点が大きく移動すると，スクロール速度もそれに比例して大きく変化し，ユーザの目がスクロールに追従できない可能性がある．変位をスクロールの加速度に比例させれば，スクロール速度の変化を緩やかにすることが出来る．但し，スクロールの加速度を単に変位の定数倍にすると，注視点が中心線を跨がず，変位が正，または，負の値のままでいる間は速度の絶対値は単調に増加し続け，やがて，ユーザの目がスクロールに追従できなくなる．「加速度2分割方式」では，スクロールの加速度を変位に比例させると共に，速度に比例した抵抗（摩擦）を表す項により，スクロール速度の絶対値が際限なく大き

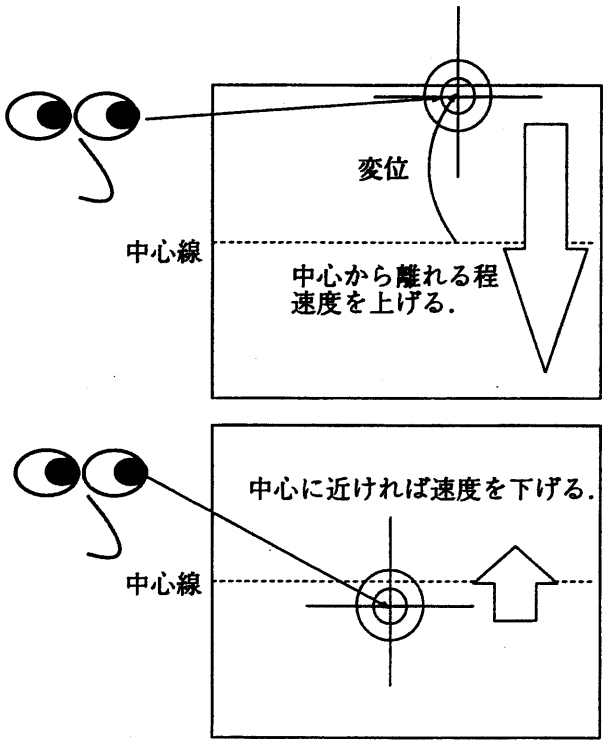

図 4.3 変位によるスクロール速度の制御

くなるのを防ぐ.「加速度2分割方式」におけるスクロール速度の算出式は次のとおりである.

$$v'(t) = -M_a e(t) - Rv(t) \tag{4.2}$$

ここで, $e(t)$, $v(t)$ は式 (4.1) と同じである. また, M_aは最大加速度を調整するための比例定数である. Rは抵抗 (摩擦) の大きさを表す定数である.

4.3.2 速度3分割方式と加速度3分割方式

「速度2分割方式」では, 変位が0とならない限り, 即ち, ウィンドウの中心線に作業者が正確に視線を向けない限り, スクロールを停止させることは出来ない.「加速度2分割方式」では視線を更に微妙に移動させなければ, 速度, 加速度を共に0とすることが出来ず, スクロールは停止しない. このことは, スクロールにおける新たな負荷をユーザに科す可能性があるばかりでなく, 視線の測定精度が悪ければ, 視線による自動スクロールそのものが実現困難となる.

そこで, ウィンドウを3つの領域に分割し, 中心線付近の領域では, 変位による速度や加速度を0とする方式 (3分割方式) を提案する (図4.4参照). 具体的には, 閾値を設定し, 変位の絶対値が閾値以下であれば, 変位による速度や加速度を0とする. 変位を速度に比例させる「速度3分割方式」では, スクロール速度を次式により算出する.

$$v(t) = \begin{cases} -M_v(e(t) + \frac{1}{n}) & e(t) < -\frac{1}{n} \\ -M_v(e(t) - \frac{1}{n}) & e(t) > \frac{1}{n} \\ 0 & \text{その他} \end{cases} \tag{4.3}$$

ここで, $e(t)$, $v(t)$, M_vは式 (4.1) と同じである. また, nは中心線付近の領域の幅を表す定数である. ここでは, 中心線付近の領域の幅は中心線の上下で同じとしている.

同様に, 変位を加速度に比例させる「加速度3分割方式」では, スクロール速度を次式により算出する.

$$v'(t)$$

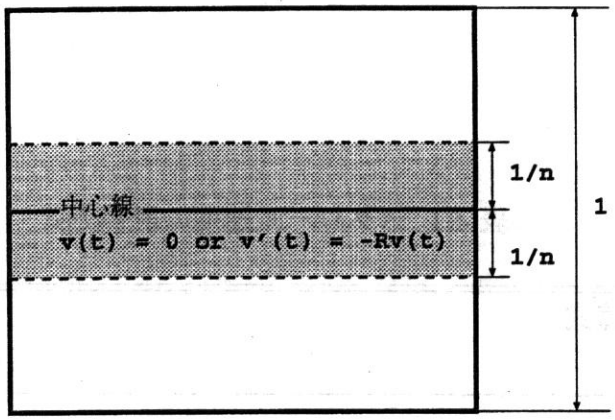

図 4.4 ウインドウ領域の 3 分割

$$= \begin{cases} -M_a(e(t) + \frac{1}{n}) - Rv(t) & e(t) < -\frac{1}{n} \\ -M_a(e(t) - \frac{1}{n}) - Rv(t) & e(t) > \frac{1}{n} \\ -Rv(t) & \text{その他} \end{cases} \tag{4.4}$$

ここで，$e(t)$，$v(t)$，M_a，Rは式（4.2）と同じである．また，n は式（4.3）と同じである．

4.4. 自動スクロール機能を持つテキストブラウザ

試作したテキストブラウザでは，4.2で示した,「視線による自動スクロール」に必要な 4 つの機能を次のように実現している [42].

（1） 視線追跡機能

ユーザの視線の向き，及び，頭部の向きや位置の測定には，米国 Applied Science Laboratories 社製の Eye Tracker 4100H と Magnetic Head Tracker を用いる．Eye Tracker 4100H は，ユーザの頭部に装着するヘルメット型の部分と据え置き型の部分とからなる（図 4.5参照）．測定誤差は 1〜2 度以下で，1 秒あたり 10 回程度の測定が可能である．

（2） 注視点算出機能

注視点の算出はソフトウェアで行う．ソフトウェアは GNU/Linux 上で C++

表 4.1 スクロール速度の計算方式

方式	計算式	試作したテキストブラウザでの値
速度 2 分割	$v(t) = -M_v e(t)$	$M_v = 3$
速度 3 分割	$v(t) = \begin{cases} -M_v(e(t) + \frac{1}{n}) & e(t) < -\frac{1}{n} \\ -M_v(e(t) - \frac{1}{n}) & e(t) > \frac{1}{n} \\ 0 & \text{その他} \end{cases}$	$M_v = 6,$ $n = 6$
加速度 2 分割	$v'(t) = -M_a e(t) - Rv(t)$	$M_a = 3,$ $R = 1$
加速度 3 分割	$v'(t) = \begin{cases} -M_a(e(t) + \frac{1}{n}) - Rv(t) & e(t) < -\frac{1}{n} \\ -M_a(e(t) - \frac{1}{n}) - Rv(t) & e(t) > \frac{1}{n} \\ -Rv(t) & \text{その他} \end{cases}$	$M_a = 6,$ $R = 1,$ $n = 6$

$e(t)$: 時刻 t における注視点の変位（単位はページ）.

$v(t)$: 時刻 t におけるスクロールの速度（単位はページ/秒）.

$v'(t)$: 時刻 t におけるスクロールの加速度（単位はページ/秒^2）.

M_v : 最大速度を調整するための比例定数.

M_a : 最大加速度を調整するための比例定数.

R : 抵抗（摩擦）の大きさを表す定数（単位は 1/秒）.

n : 中心線付近の領域の幅を表す定数（単位はページ）.

図 4.5 視線追跡装置

で開発した．Ethernet を介して実時間で送られてくる測定データを受け取り，注視点を算出し，算出結果を Ethernet を介して実時間で送信することが出来る．注視点の算出精度は，ユーザにも依存するが，平均誤差 10mm 前後である．なお，注視点の算出精度を高めるためのキャリブレーション用ソフトウェアも合わせて開発した．

(3) スクロール制御機能

スクロール制御はソフトウェアで行う．ソフトウェアは Linux 上で C++で開発した．4.3で述べた 4 つの方式での制御が可能である．スクロール速度や加速度の計算式中の定数値としては，表 4.1の最右列に示す値を用いている．特に,「3 分割方式」では，変位による速度や加速度を 0 とする領域の幅を $2n = 1/3$ (ページ)とし，また，ウィンドウの上端，あるいは，下端から中央に向かって 1/6 ページの位置に注視点があるときに，変位による速度が 1 ページ/秒に，加速度が 1 ページ/秒^{2}となる．

(4) 表示機能

図 4.6 試作したテキストブラウザの利用の様子

表示機能は，テキストブラウザとして X-Window を用いて実装した．テキストブラウザはテキストのみ表示可能で，表示部（テキストウィンドウ）の大きさは横 80 文字，縦 25 行である．スクロールは縦方向のみ可能であるが，スクロール制御ソフトウェアからの指示を受け取り，1ドット単位でテキストをスクロールさせることが出来る．なお，スクロール操作は，キーボードからも可能である．Eye Tracker 4100H を装着してテキストブラウザを利用している様子を図 4.6に示す．

4.5. 評価実験

4.5.1 概要

「視線による自動スクロール」の有効性を確かめるため，4.4で述べたテキストブラウザを用いて，スクロール方式の比較実験を行った．実験では，スクロールを頻繁に行う必要のある作業（タスク）を，テキストブラウザを用いて被験者に行ってもらい，被験者へのインタビューや実験データに基づいて，スクロール方式間での使用感や作業効率の違いを評価した．

表 4.2 タスクで用いたプログラム

プログラム	P1	P2	P3	P4	P5
行数（LOC）	236	252	255	218	271
printf 文の個数	2	3	2	3	3

被験者には次のような指示を与えた．

- 与えられたCプログラムをテキストブラウザ上で読み，プログラム中に含まれている全ての printf 文をできるだけ早く発見して下さい．
- printf 文を発見する毎に，その旨を声を出して実験実施者に報告して下さい．
- 全ての printf 文を発見したと思ったら，ユーザの終了を声に出して宣言して下さい．

なお，与えられたCプログラムに含まれている printf 文の個数は，タスク開始時に被験者に知らせておいた．また被験者が作業の終了を宣言した時点で，タスクは終了したとみなした．

タスクで用いるCプログラムとして，P1，P2，P3，P4，P5 の 5 つを用意した．各プログラムの行数と含まれる printf 文の個数を表 4.2に示す．表 4.2からも分るように，用意したプログラムの行数は，実験で用いたテキストブラウザが一度に表示できる行数（25 行）のおよそ 10 倍となっている．また，含まれる printf 文の個数が 2，3 個と少ないため，被験者はテキストブラウザのスクロール機能を使い，注意深くプログラムを見ていく必要がある．

被験者は 5 名（A，B，C，D，E）で，いずれも奈良先端科学技術大学院大学の教官または学生である．被験者は全て，C 言語によるプログラムの読み書きが出来，UNIX のテキストエディタ（mule や vi 等）を日常的に使用している．従って，テキストブラウザやそのスクロール機能，タスク，タスクで用いるCプログラム等については，簡単な説明と練習だけで，被験者はそれらを正しく理解することが出来た．

表 4.3 被験者への対象プログラムとスクロール方式の割り当て

スクロール制御方式	被験者				
	A	B	C	D	E
速度 2 分割	P5	P2	P4	P3	P1
速度 3 分割	P2	P3	P1	P5	P4
加速度 2 分割	P3	P4	P2	P1	P5
加速度 3 分割	P4	P1	P5	P2	P3
キーボード	P1	P5	P3	P4	P2

4.5.2 タスク実行

タスク実行に先立って，視線追跡装置のキャリブレーションと練習タスクを，被験者毎に行った．キャリブレーションは筆者らが開発したソフトウェアを用いて，ウィンドウ上での注視点の平均測定誤差が 20mm 未満になるまで行った．練習タスクは，テキストブラウザやスクロール方式に被験者が慣れ，被験者にタスクを正しく理解してもらうために行った．練習タスクで用いたプログラムの行数は 267 行，含まれる printf 文は 5 個であった．なお，必要に応じて練習タスクは繰り返し行い，練習タスク後に再度キャリブレーションを実施した．

キャリブレーションと練習タスクを終えた 5 名の被験者にはそれぞれ，P1〜P5 を対象として計 5 回のタスクを行ってもらった．但し，各タスクは異なるスクロール方式で行ってもらった．即ち,「速度 2 分割方式による自動スクロール」,「速度 3 分割による自動スクロール」,「加速度 2 分割方式による自動スクロール」,「加速度 3 分割による自動スクロール」，そして,「キーボード操作によるスクロール」の 5 方式でそれぞれ 1 回ずつタスクを行ってもらった．被験者がタスクで用いたスクロール方式とプログラムの対応関係を表 4.3に示す．表 4.3からも分るように，同一プログラムに同一スクロール方式が 2 回以上用いられない様に，また，同一プログラムに同一被験者が 2 回以上タスクを行わない様に割り当てた．

各タスクにおいては，タスク終了までに要した時間（タスク時間），及び，誤りの発生回数を計測した．ここで，タスクにおける誤りの発生とは，スクロールに

表 4.4 スクロールの計算方式の比較

評価項目	スクロール制御方式			
	速度 2分割	速度 3分割	加速度 2分割	加速度 3分割
(1) キーボード操作と比較した作業効率	○	○	○	○
(2) 反応の良さ	○	○	×	×
(3) 速度変化の自然さ	○	×	○	×

よってウィンドウ内に現れたprintf文に被験者が気づかず，視線が向けられないままウィンドウ外に移動した場合を指す．また，全てのタスク終了後，スクロール方式の違いによる使用感や作業効率の差，タスク中に発生した誤りの原因等について，被験者へのインタビューを行った．

4.5.3 実験結果

被験者へのインタビューの結果を表4.4にまとめる．表4.4の評価項目のうち(2),(3)は，インタビューの中で被験者が述べた感想において，スクロール方式によってその優劣がはっきりと分れた項目を列挙している．表中の○は比較的優れていることを，×は比較的劣っていることを，それぞれ表す．但し，図4.6に示したように，実験に用いた視線追跡装置はヘルメット型であり，自動スクロールのために日常的に使用するのは現実的ではない．インタビューにおいては,「視線追跡装置が小型軽量化され，装着感もほとんどなく，ユーザの動作も制限しない」という状況を仮定した上で回答してもらった．各タスクで計測したタスク時間と誤り発生回数を表4.5にまとめる．表4.5において，*はタスク中に誤りが発生したことを表す．いずれのタスクおいても高々1回の誤りしか発生しなかった．以上の実験データに基づき，以下では「視線による自動スクロール」の有効性を定性的，及び，定量的に評価する．

(1) 定性的評価

表4.4の評価項目 (1) の結果から分るように，スクロールの制御方式に関わら

表 4.5 実験データ

方式	タスク時間 (秒)						誤り
	A	B	C	D	E	平均	回数
速度2分割	27	38	28	45	45*	36.6	1
速度3分割	73*	61*	35	24*	36*	45.8	4
加速度2分割	32	37	75*	36	50	46.0	1
加速度3分割	30	87*	33	70*	36	51.2	2
キーボード	55	59	44	36	95	57.8	0

(「*」は，誤りが1回発生したことを示す.)

ず,「視線による自動スクロール」は,「キーボード操作によるスクロール」と同程度かそれ以上の作業効率が得られると感じる被験者がほとんどであった．特に，速度2分割方式が最も使いやすいという意見が多かった.

スクロールの制御方式による違いを見てみると，まず，評価項目（2）の結果から分るように，加速度方式は視線移動に対する反応が悪い，と感じる被験者が多かった．これは，加速度方式が，スクロール速度の変化を緩やかにする反面，スクロール開始時にはスクロール速度を徐々にしか大きくできないことに起因していると思われる.

評価項目（3）の結果からは，3分割方式においてスクロール速度の変化が不自然であると感じる被験者が多かったことが分る．これは，4.3.2で述べたように，スクロールの停止を容易にするために，変位による速度や加速度を0とする領域を中心線付近に設定したためと考えられる．即ち,「視線の移動」が必ずしも「スクロール速度の変化」とならないことが，被験者をかえって混乱させる結果になったと思われる.

（2） 定量的評価

表4.5のデータを用いて，スクロール制御方式間におけるタスク時間の母平均の差を統計的に検定した．その結果,「速度2分割法による自動スクロール」のタスク時間のみが「キーボード操作によるスクロール」のタスク時間より小さい，と有意水準5%で言えることが分った．また，その他の制御方式による自動スク

ロールのタスク時間については,「キーボード操作によるスクロール」のタスク時間と有意な差はないことが分った．これらの結果は，文字列検索における被験者の定性的評価と一致する．即ち前述の「スクロールの制御方式に関わらず，視線による自動スクロールは，キーボード操作によるスクロールと同程度かそれ以上の作業効率が得られた」，及び,「自動スクロールでは速度2分割法が最も使いやすかった」と一致する．

タスク中の誤り回数についても同様に検定した結果,「速度3分割法による自動スクロール」の誤り回数のみが「キーボード操作によるスクロール」の誤り回数より大きい，と有意水準5%で言えることが分った．また，その他の制御方式による自動スクロールの誤り回数については,「キーボード操作によるスクロール」の誤り回数と有意な差はないことが分った．

スクロールのスムーズな停止を考慮した速度3分割法では，タスク実行により多くの時間を要し誤りが多発した．その原因は，中心付近に停止の領域を設けた分，スクロール速度を加減するための領域が狭くなりスクロール制御が困難になった点にあると思われる．しかも本実験では文字列検索をタスクとしており，スクロールをスムーズに停止させる必要がほとんどなかった.「プログラム中の誤り(バグ)の内容を調べる」といった，目標テキストが必ずしも明確でなく，テキストを熟読する必要のあるタスクでは，速度3分割法の優位性が示される可能性がある．また3つの領域やその境界を提示することにより，速度3分割法におけるスクロール速度の変化の不自然さが解決されタスク時間や誤り回数が改善されるかもしれない．

以上の評価結果をまとめると，視線追跡装置を装着することそのものの負荷を無視し，文字列検索というタスクに限定すると，視線による自動スクロールは，キーボード操作によるスクロールと比較して同程度かそれ以上に有効であると言える．同様に，文字列検索というタスクに限定した場合，4つのスクロール制御方式の中では速度2分割方式が定性的にも定量的にも優れていると言える．

4.6. まとめ

本章では，キーボードやマウスによる操作を必ずしも行わなくても，コンピュータユーザが読み書きしたいと考える情報がウィンドウ内に表示されるインタフェースの実現を目指して，視線によるテキストウィンドウの自動スクロール方式を提案した．また，4つのスクロール制御方式を考案し，それらの制御方式による自動スクロールが可能な「テキストブラウザ」を試作した．更に，試作したテキストブラウザを用いた実験を行い，視線によるテキストウィンドウの自動スクロールの有効性を，主にキーボード操作によるスクロールとの比較により，定性的，及び，定量的に示した．特に，文字列検索というタスクに限定すると速度2分割方式は，統計的検定によっても，キーボード操作との有意な差が認められ，考案した4つのスクロール制御方式の中で最も優れていることが分った．

水平方向のスクロールについて検討すること，文字列検索以外のタスクにおいて自動スクロールを評価することの2つが今後の課題である．

第5章

おわりに

本論文では，MS–Windows や MacOS などの一般的な GUI 上での操作の効率を上げることを目的として，マウスやキーボードなど従来から用いられている入力デバイスに加え，計算機画面を見るユーザの視線の動きを用いた視線追加型インタフェースについて述べた．本論文ではターゲット選択，ドラッグ&ドロップ，スクロールの 3 つの操作に対して，視線の動きを入力として追加する具体的な方式を提案した．提案した方式により，GUI 操作全般の操作時間を短縮することが可能となる．

2章では，ターゲット選択をポインティング操作と確定操作に分け，ポインティング操作に視線を，確定操作には従来通りマウスボタンを用いた．さらにカーソル位置微調整方式として Auto 方式，Manual 方式，SemiAuto 方式の 3 つを提案した．一般的な GUI を想定した環境で評価実験を行った結果，SemiAuto 方式は，選択誤りを大幅に増やすことなく，操作時間を同程度かより短くできることが分かった．特に，非連続操作においては，操作時間を約 2/3 に短縮できた．

3章では，ドラッグ&ドロップをドラッグアイコンとドロップアイコンに対する 2 回の連続したターゲット選択であると見なし，ターゲット選択方式として最も性能の優れていた SemiAuto 方式を適用した．評価実験の結果，提案方式による操作は従来のマウスのみの操作に比べて，操作誤りを大幅に増やすことなく，1 回の操作について平均で約 17%(0.4 秒) 短く操作できた．また初心者を想定した実験では，操作誤りも減り，平均で約 27%(0.8 秒) 短く操作できた．

4章では，ウィンドウ中央からユーザの画面上の注視点までの垂直方向の距離

をスクロール速度に比例させて自動的にスクロールを行う方式を提案した．ユーザはウィンドウ内で視線を移動させるだけで，表示したいテキスト部分をウィンドウ中央部に移動させ，表示させた状態でスクロールを停止させることが出来る．評価実験の結果から，テキストのスクロールにより文字列を検索するというタスクにおいては，視線による自動スクロールはキーボード操作によるスクロールと比較して同程度かそれ以上に有効であることが分った．

3つ提案方式は操作時間を短縮できるだけでなく，マウスやキーボードのみを用いた場合に比べてより自然に操作を行うことができる可能性がある．マウスのみを用いてターゲット選択やドラッグ&ドロップを行う場合，ユーザはマウスを動かすよりも先に，まず選択したいGUI部品やドラッグあるいはドロップしたいアイコンを見る場合が多い．視線を追加したターゲット選択及びドラッグ&ドロップでは,「GUI部品やアイコンをまず見る」というユーザの行動によってカーソル位置をおおまかに移動させた．またウィンドウ上のテキストを読む場合，テキストを読み進めるに従いユーザは，ウィンドウの下端方向へと視線を移動させる．逆に遡って読む場合には，ウィンドウの上端方向へと視線を移動させる．自動スクロール方式では，このウィンドウ内の視線の動きを用いてスクロールを制御した．より自然に操作を行うことができることは，特に計算機利用の初心者に有効であると考えられる．実際，視線を追加したドラッグ&ドロップ方式の評価結果は，初心者に有効であることを示した．

本論文の成果は，視線追跡装置の導入を促進するものである．従来の研究では視線追跡装置の導入とともにGUIを大幅に変更することを前提としていた．一方，本論文で提案した方式は，GUIの大幅な変更を必要としないため，導入に必要なコストが少ない．さらに近年，比較的安価に実現可能な視線追跡装置の研究開発がなされている [6,17,18,21] ため視線追跡装置は次世代の入力デバイスとして有望である．

今後は，提案方式の普及を目指し，操作方式をMS-WindowsやMacOSなどのデスクトップやウェブブラウザなどのアプリケーションに組込む．本論文において提案方式の評価のために作成したプログラムコードを再利用できる形にライブラリとしてまとめ，安価な視線追跡装置が入手可能になり次第すぐに提案方式を利用できるようにする．また本論文で対象とした3つのGUI操作ではカバーす

ることのできなったウィンドウのズームについても視線の動きを入力として追加して効率良く操作できる方法を検討する．具体的な検討内容は,「ユーザがディスプレイに顔を近づけたときに，ユーザの注視点にある対象を拡大して表示する．逆に遠ざけた場合には縮小して表示する」という方法である [36].

謝辞

本研究を進めるにあたり多くの方々に，御指導，御協力，ご支援頂きました．ここに謝意を表します．

奈良先端科学技術大学院大学 情報科学研究科 松本 健一教授には，本研究の全過程において多大な御指導頂きました．本研究を通して何を考えれば良いか，考えをどのようにまとめれば良いか，そしてそれを人に正確に伝えるにはどうすれば良いかということを丁寧に教示して頂きました．心より感謝致します．

大阪大学大学院 基礎工学研究科 井上 克郎教授には，博士後期課程 1，2 年次において建設的な御意見と励ましの言葉を頂きました．心より感謝致します．

奈良先端科学技術大学院大学 情報科学研究科 横矢 直和教授には，本論文を審査して頂きました．心より感謝致します．

奈良先端科学技術大学院大学 情報科学研究科 小笠原 司教授には，本論文を審査して頂きました．心より感謝致します．

奈良先端科学技術大学院大学 鳥居 宏次学長 には，博士前期課程において，多くの御指導，御助言，激励を頂きました．特に博士前期課程 1 年次に頂いた激励なしには博士前期課程を修了できることなく挫折したと思います．心より感謝致します．

奈良先端科学技術大学院大学 情報科学センター 小山 正樹 教授には，博士前期課程において研究中だった自動スクロール方式について多くの御指導，御助言頂きました．心より感謝致します．

奈良先端科学技術大学院大学 情報科学センター 飯田 元助教授には，本研究を進めるにあたって，特にポストの作成と発表の仕方について御指導，御助言頂きました．締切や発表直前で不安な状態にある私への御指導は常に建設的かつ具体的なものでありたいへん有益でした．心より感謝致します．

奈良先端科学技術大学院大学 情報科学研究科 中小路 久美代助教授には，本研究を進めるにあたって貴重な御助言を頂きました．また関連研究と国際会議を紹介して頂きました．特に博士後期課程 3 年次に紹介頂いた INTERACT へ参加したことは，本研究をまとめるにあたって極めて有益でした．英語表現においても御指導頂きました．心より感謝致します．

奈良先端科学技術大学院大学 情報科学研究科 島 和之助手には，本研究を進め

るにあたって多くの御助言頂きました．博士前期課程１年次には，私の抱いた研究生活に関する疑問に相談にのって頂きました．心より感謝致します．

奈良先端科学技術大学院大学 情報科学研究科 門田 暁人助手には，本研究を進めるにあたって多大な御指導，御助力頂きました．研究の進め方に思い悩む私にいつも懇切丁寧に指導して下さいました．評価実験の設計にも多大な御助力を頂きました．心より感謝致します．

奈良先端科学技術大学院大学 情報科学研究科 蔵川 圭助手には，本研究を進めるにあたって御助言及び関連研究の紹介をして頂きました．心より感謝致します．

奈良先端科学技術大学院大学 情報科学研究科 高田 義広 元助手 (現オムロン株式会社) には，本研究を進めるにあたって御指導頂きました．特に研究を進める上での心構えを御指導頂きました．心より感謝致します．

奈良先端科学技術大学院大学 情報科学研究科 高田 眞吾 元助手 (現 慶應義塾大学 理工学部 情報工学科 専任講師) には，本研究を進めるにあたって御助言，御相談頂きました．また私が初めて外部発表に行ったときにかけて頂いた御言葉は大変な励みになりました．心より感謝致します．

NTT コミュニケーション科学基礎研究所 大野 健彦研究員には，インターンとして同研究所で研究できる機会を設けて頂きました．さらに研究所において熱心に御指導，御議論頂きました．視線インタフェースに関する具体的な議論は大変刺激になりました．心より感謝致します．

奈良先端科学技術大学院大学 情報科学研究科 森崎 修司氏 (現 株式会社インターネットイニシアティブ) には，本研究を進めりにあたり，御議論，御助力頂きました．研究はもちろんのこと生活の面においても何度も助けて頂きました．特に博士後期課程１年次に腹痛をおこした折には，深夜にもかかわらず何度も病院まで運んで頂きました．もしあのとき病院に行くことができなければ，本研究を遂行することはおろか，今生きていることもできなかったかもしれません．心より感謝致します．

奈良先端科学技術大学院大学 情報科学研究科 武村 泰宏氏には，本論文の作成にあたり多大な御助力を頂きました．心より感謝致します．

奈良先端科学技術大学院大学 情報科学研究科 神代 知範氏 (現 TIS 株式会社) には，本研究を進めりにあたり，御議論，御助力頂きました．特に評価実験を進

めるにあたり多大な御助力を頂きました．心より感謝致します．

奈良先端科学技術大学院大学 情報科学研究科 山本 恭裕氏 (現 日本学術振興会) には，本研究を進めるにあたり，御議論，御助力頂きました．心より感謝致します．

奈良先端科学技術大学院大学 情報科学研究科 大平 雅雄氏には，本研究を進めるにあたり，御議論，御助力頂きました．心より感謝致します．

本研究を進めるにあたってに御助力，御協力頂いた松本研究室のみなさまに感謝致します．

最後に，これまで私を暖かく見守って下さった両親に感謝致します．

参考文献

[1] 坂尚幸, 菅又生磨, 板倉直明, 坂本和義, 北本拓: ガイド領域を用いた視線文字入力インタフェースの提案, 電子情報通信学会論文誌, Vol. J84–D–II, No. 5, pp. 799–804 (2001).

[2] Card, S., Moran, T. and Newell, A.: *The Psychology of Human-Compubter Interaction*, Lawrence Erlbaum Associates (1983).

[3] 江崎信也, 海老澤嘉伸, 杉岡明, 小西学: ビデオ式注視点検出によるコミュニケータによる瞬きを用いた高速メニュー選択法, 1997 年電子情報通信学会情報・システムソサイエティ大会講演論文集 (1997).

[4] Goldberg, J. H. and Schryver, J. C.: Eye-gaze determination of user intent at the computer interface, *Eye movement research: Mechanisms, processes and applications* (Findaly, J., Walker, R. and Kentridge, R.(eds.)), North-Holland, pp. 491–502 (1995).

[5] Hansen, J., Allan, W. and Peter, R.: Eye-gaze control of multimedia systems, *Symbosis of Human and Artifact* (Anzai, Y., Ogawa, K. and Mori, H.(eds.)), Vol. 20A, Elsevier Science, pp. 37–42 (1995).

[6] Hansen, J., Hansen, D. and Johnasen, A.: Bringing gaze-based interaction back to basics, *Proc. Universal access in CHI: Towards an information society for all* (Stephanidis, C.(ed.)), Lawrence Erlbaum Associates, pp. 325–328 (2001).

[7] 池田光男: 視覚の心理物理学, 森北出版 (1975).

[8] 入鹿山剛堂: ケータイ文字入力の現状と将来, 電子情報通信学会誌, Vol. 84, No. 11, pp. 819–827 (2001).

[9] 伊藤和幸, 数藤康雄: 任意文字連続注視時の視線移動の計測—視線入力式コミュニケーション機器開発への応用—, 情報処理学会研究報告, ヒューマンインターフェース 68–5, pp. 31–38 (1996).

[10] Jacob, R.: What you look at is what you get: Eye movement-based interaction techniques, *Proc. ACM CHI'90 Human Factors in Computing System Conference*, pp. 11–18 (1990).

[11] Kabbash, P., MacKenzie, I. S. and Buxton, W.: Human performance using computer input devices in the preferred and non-preferred hands, *Proc. INTERCHI'93*, ACM Press, pp. 474–481 (1993).

[12] 神代知範, 大和正武, 門田暁人, 松本健一, 井上克郎: 視線とマウスの併用によるドラッグ&ドロップ方式の実験的評価, 電子情報通信学会技術研究報告, Vol. HIP99–81, pp. 37–44 (2000).

[13] 小松崎篤, 篠田義一, 丸尾敏夫 (編): 眼球運動の神経学, 医学書院 (1985).

[14] 久野靖, 多木敦雄, 角田博保, 粕川正充: 「アイコン投げ」ユーザインタフェース, コンピュータソフトウェア, Vol. 13, No. 3, pp. 38–48 (1996).

[15] 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).

[16] Maglio, P. P., Matlock, T., Campbell, C. S., Zhai, S. and Smith, B. A.: Gaze and speech in attentive user interfaces, *Proc. Advaces in Multimodal Interfaces (ICMI'2000)*, pp. 1–7 (2000).

[17] Matsumoto, Y. and Zelinsky, A.: An Algorithm for Real-time Stereo Vision Implementation of Head Pose and Gaze Direction Measurement, *Proc. IEEE Fourth International Conference on Face and Gesture Recognition (FG'2000)*, pp. 499–505 (2000).

[18] 加藤真弓, 佐藤淳: 未校正カメラと未校正ディスプレイによりビジュアルインタフェース, 電子情報通信学会論文誌, Vol. J84–D–II, No. 5, pp. 769–777 (2001).

[19] 大野健彦: 階層メニュー選択における視線の利用, 情報処理学会研究報告, ヒューマンインターフェース 71–13, pp. 83–90 (1997).

[20] 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).

[21] 大野健彦, 武川直樹, 吉川厚: 眼球形状モデルに基づく視線測定システム—視線入力デバイスの実現に向けて—, 情報処理学会研究報告, ヒューマンインターフェース 93, pp. 47–54 (2001).

[22] 大野健彦: 視線インタフェースから視線コミュニケーションへ – 視線のある環境を目指して –, 情報処理学会研究報告, ヒューマンインターフェース 95, pp. 171–178 (2001).

[23] Partala, T., Aula, A. and Surakka, V.: Combined voluntary gaze direction and facial muscle acitivity as a new pointing technique, *Proc. IFIF TC.13 Conference on Human–Computer Interaction*, IOS Press, pp. 100–107 (2001).

[24] Salvucci, D. D. and Anderson, J.: Intelligent Gaze-added interface, *Proc. ACM CHI'2000 Human Factors in Computing System Conference*, ACM Press, pp. 273–280 (2000).

[25] Shaw, R., Crisman, E., Loomis, A. and Laszewski, Z.: Eye wink control interface: Using the computer to provide the severely disabled with increased flexibility and comfort, *Proc. 3rd Annual IEEE Symposium on Computer-Based Medical Systems*, pp. 105–111 (1990).

[26] Sibert, L. E. and Jacob, R.: Evaluation of Eye Gaze Interaction, *Proc. ACM CHI'2000 Human Factors in Computing System Conference*, pp. 281–288 (2000).

[27] 高木啓伸: 視線を用いたユーザインターフェース開発: EyeTk, 情報処理学会研究報告, ヒューマンインターフェース 65–15, pp. 81–86 (1996).

[28] 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～, 日本ソフトウェア科学会 WISS'97, インタラクティブシステムとソフトウェア V (尾内理紀夫 (編)) (1997).

[29] 高木啓伸: 視線の移動パターンに基づくユーザの迷いの検出—効果的な作業支援を目指して, 情報処理学会論文誌, Vol. 41, No. 5, pp. 1317–1327 (2000).

[30] 田村博 (編): ヒューマンインタフェース, オーム社 (1998).

[31] 田崎京二, 大山正, 樋渡涓二 (編): 視覚情報処理, 朝倉書店 (1979).

[32] Velichkovsky, B. and Hansen, J.: New technological windows into mind: there is more in eyes and brains for human–computer interaction, *Proc. ACM CHI'96 Human Factors in Computing System Conference*, pp. 465–503 (1996).

[33] 山田光穂, 福田忠彦: 眼球運動による文章作成・周辺機器制御装置, 電子情報通信学会論文誌, Vol. J69–D, No. 7, pp. 1103–1107 (1986).

[34] 大和正武, 門田暁人, 高田義広, 松本健一, 鳥居宏次: 視線によるテキストウィンドウの自動スクロール, 情報処理学会論文誌, Vol. 40, No. 2, pp. 613–622 (1999).

[35] 大和正武, 神代知範, 門田暁人, 松本健一, 井上克郎: 視線によるマウスカーソルの自動移動, 情報処理学会研究報告, ヒューマンインターフェース 84–13, pp. 73–78 (1999).

[36] 大和正武, 大野健彦: 視線を利用したアプリケーションの内部構造可視化インタフェース, 電子情報通信学会技術研究報告, Vol. HIP2000–12, pp. 37–42 (2000).

[37] Yamato, M.: Gaze added interface suitable for General GUIs, *Proc. IFIF TC.13 Conference on Human–Computer Interaction*, IOS Press, pp. 665–668 (2001).

[38] 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).

[39] 大和正武, 神代知範, 門田暁人, 松本健一: 視線・マウス併用型インタフェースのドラッグ&ドロップ操作への適用, 情報処理学会論文誌 (投稿中).

[40] Yamato, M., Monden, A., Matsumoto, K., Inoue, K. and Torii, K.: Button selection for general GUIs using eye and hand together, *Proc. AVI'2000 Advanced Visual Interface*, pp. 270–273 (2000).

[41] Yamato, M., Monden, A., Matsumoto, K., Inoue, K. and Torii, K.: Quick Button Selection with Eye Gazing for General GUI Environments, *Proc. ICS'2000 International Conference on Software: Theroy and Practive*, pp. 712–719 (2000).

[42] 大和正武, 高田義広, 鳥居宏次: 視線追跡によりプログラムエディタを自動スクロールさせる方式の比較, 電子情報通信学会技術研究報告, Vol. SS97–16, pp. 1–8 (1997).

[43] 吉川厚, 大野健彦: 視線を読む -ユーザにやさしい視線測定環境-, *NTT R & D*, Vol. 48, No. 4, pp. 399–408 (1999).

[44] Zhai, S., Morimoto, C. and Ihde, S.: Manual and Gaze Input Cascaded (MAGIC) Pointing, *Proc. ACM CHI'99 Human Factors in Computing System Conference*, ACM Press, pp. 246–253 (1999).