筑波大学大学院博士課程

システム情報工学研究科修士論文

手の動きを利用したメニュー操作による大画面インタラクション手法

中村　卓

(コンピュータサイエンス専攻)

指導教員　田中　二郎

2008年3月

概要

画面から離れて操作する手法の一つに手のジェスチャを利用する方法がある．手のジェスチャを利用することで，マウスなどの機器を利用せずに画面とのインタラクションを行うことができる．特に，近年広く普及しつつある大画面環境で操作を行う際に，大画面から離れた場所から操作することができるため，画面全体を把握しながらの操作が容易である．そのため，大画面環境では手のジェスチャを利用して操作を行うことは非常に有効な手段といえる．

しかし，手のジェスチャを利用したインタラクション手法では，操作に必要なジェスチャの数が多くなりやすく，また複雑なジェスチャを利用した場合には，設備やシステムが大掛かりになってしまうといった問題が生じる．そこで，我々は，手の動きをハンドジェスチャとみなし，その手の動きを利用してメニュー操作を行い，それによって大画面とのインタラクションを行う手法を提案する．本研究では，ペンベースのメニューインタフェースで利用されている操作手法を手の動きで操作を行いやすいように改良し，実際に 2 種類のインタラクション手法を設計・試作した．ひとつは，FlowMenu と呼ばれるメニューの操作時に描かれる軌跡を手の動きを利用して描くことでその軌跡に対応したメニュー操作を行い，それによって画面とのインタラクションを行う手法である．もう一つは，手の動きを利用してある特定のメニューアイテムを横切った際に，そのメニューを実行することで画面とのインタラクションを行う手法である．

本手法を利用することにより，大画面環境において手の動きのみで様々なインタラクションを行うことが可能で，一つ一つの操作は簡単であるためインタラクション手法としてはわかりやすい．また，ペンベースで利用されているメニュー操作手法を有効に利用することにより，流れるような連続操作を手の動きによって行うことができる．

目 次

図 目 次

第1章　大画面環境におけるインタラクション

1.1　既存のインタラクション手法

プラズマディスプレイやプロジェクタなどの発達・普及により，研究室をはじめ，駅などの公共の場など様々な場所で大画面が広く普及してきている．また，それに伴い，そのような大画面環境とのインタラクションを行う機会が増えてきている．

しかし，それらを操作するためのインタラクション手法はマウスやキーボード，タッチパネルなどを利用した従来のインタラクション手法をそのまま利用することが多い．しかし，それらの手法は，大画面とのインタラクションを行うには不向きであるといえる．

例えば，大画面を操作する際に，従来のマウスやキーボードを利用したインタラクションがそのまま利用されることが多い．しかし，マウスやキーボードはデスクトップ環境向けのインタラクション手法であるので，大画面環境で利用するには不向きなインタラクション手法であるといえる．例えば，駅などの公共の場に設置された大画面を操作しようとした場合，マウスを利用するための机などとの水平で広いスペースを確保することが難しい．また，それらを利用する場所によって使い勝手などが大きく変わってしまうため，大画面環境には不向きである．

マウスやキーボード以外のインタラクション手法の例として，タッチパネルの利用が挙げられる．プラズマディスプレイ程度のサイズの大画面環境では，タッチパネルは直観的で利用しやすいインタラクション手法であるといえる．しかし，プラズマディスプレイ以上のサイズ，例えば，壁一面がディスプレイになっている大画面のような場合には，手が届かない場所は直接操作することが困難である．このような問題を解決するために，手の届かない場所の操作を補助するためのインタフェースについても研究 [13] も行われているが，タッチパネルを利用した際の最大の問題点として操作する時には必ず画面に触れていなければならないため，画面と利用者までの距離が近く画面を広い範囲で把握することは困難であるということである．そのため，タッチパネルを利用したインタラクションも大画面向けとは言い難い．

1.2　大画面環境向けのインタラクション

大画面環境におけるインタラクション手法として望まれることとして，まず，図 1.1 のように画面の広い範囲を把握しながら操作できることである．また，机などといった特定のスペースを利用せずに操作を行うことができるということも重要である．そのため，近年，マウスやキーボード，タッチパネルなどを利用しないインタラクション手法の研究が盛んに行

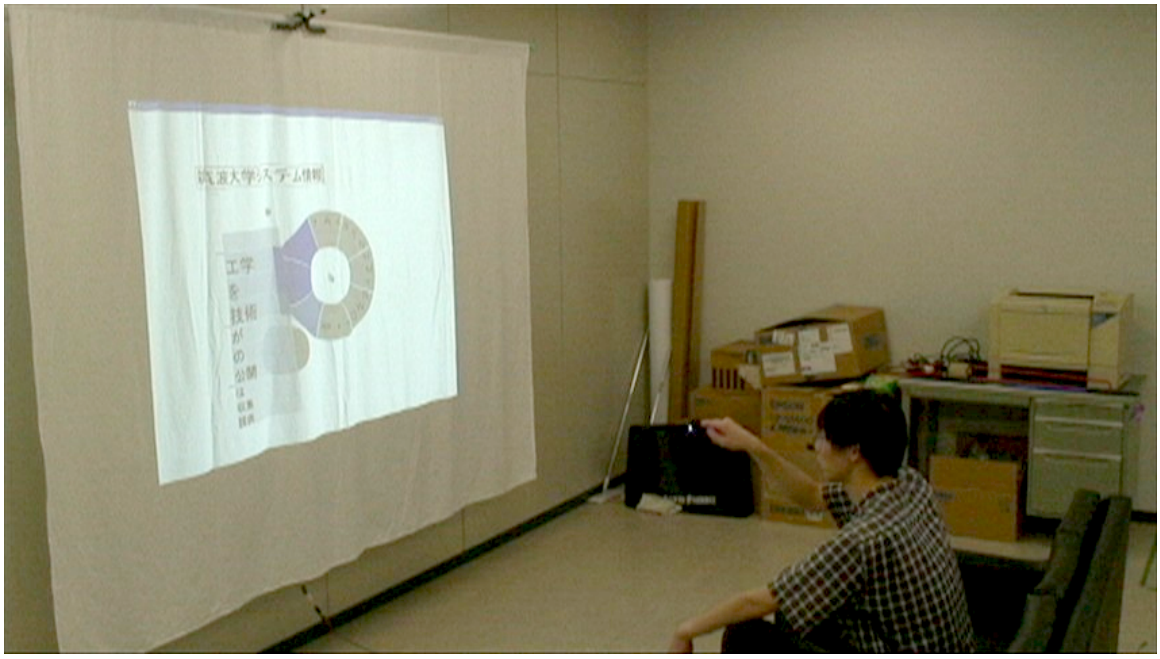

図 1.1: 大画面環境における操作例

われている．

その中の一つにレーザーポインタ [6, 12, 21] や携帯電話 [17] などを利用したジェスチャ操作によって画面とのインタラクションを行う手法が存在する．また加速度センサなどを利用してジェスチャを行い，それによってインタラクションを行う手法もある [3]．ジェスチャ操作の利点として，離れた場所からでも操作を行うことが可能であるため，図 1.1 のように画面を把握しながら操作を行うことが容易にできるため，大画面向けのインタラクション手法と言える．

ジェスチャを利用したインタラクション手法の一つに手のジェスチャ(ハンドジェスチャ) を利用するものがある [7, 16, 20, 28]．手は人間の身体の中で最も動かしやすい部位の一つであるため，非常に数多くのハンドジェスチャが存在し，それらを利用することで様々なインタラクションを行うことが可能である．本研究でもこのハンドジェスチャに着目し，大画面向けのインタラクションの設計を試みることにした．

しかし，ハンドジェスチャはユーザにとって手軽である反面，1) ジェスチャによるコマンドを増やすとジェスチャそのものが複雑になりユーザが覚え切れない，2) またコンピュータでの認識が難しくなりユーザとのインタラクションがスムーズにできない，3) ユーザの手の位置を正確に補足するためにセッティングが大掛かりになる，4) 手ぶれ等の原因によりポインティングを正確に行えない，などの問題があり，これらを考慮しなければならない．

1.3　本研究の目的

本研究では，大画面環境における操作手法として，ハンドジェスチャを用いたインタラクション手法の設計を行う．その際に，ハンドジェスチャを用いることによって生じる様々な問題を解消し，利用者への負担が少なく，また利用しやすいインタラクションの設計を目指す．さらに，一つ一つの操作がスムーズにかつ連続的に行えるような操作手法の設計・試作していく．そして，より使いやすいインタラクションを目指して，設計した手法について考察し，改良を行っていく．

本論文では，まず最初に，ハンドジェスチャを利用したインタラクション手法の現状とその問題点について述べる．その後，本研究で提案する手の動きを利用したインタラクション手法について説明し，2 種類のメニュー操作手法の設計・試作を行う．そして，それぞれの問題点などについて議論した後に結論を述べる．

第2章　ハンドジェスチャを利用したインタラクション

2.1　ハンドジェスチャ

身体を利用したジェスチャは，日常生活中の非言語的コミュニケーションの一つとしてよく利用されており，そのジェスチャを操作に利用することは自然なことと言える．特に，手は人間の身体のうち最も動かしやすい部位の一つであるため，非常に多くのジェスチャを行うことが可能である．例えば，手を振ったり・傾ける，手を動かす，手の形状 (グーやパー) など様々なジェスチャを行うことができる．また，それらを組み合わせることでより複雑なジェスチャを容易に行うことができる．例えば，図 2.1 にあるような手の形状に位置情報を組み合わせるだけで新しいジェスチャを容易に作成することができ，それだけで利用可能なジェスチャの数は何倍にも膨れ上がる．そのため，操作一つ一つにジェスチャを割り当てることは容易なことである．

これらのハンドジェスチャを組み合わせることで様々なインタラクション手法が設計可能になる．そのため，現在，様々なインタラクション手法の研究がなされている．

図 2.1: 手の形状の一例

2.2 大画面向けインタラクション手法

ハンドジェスチャを利用したインタラクションの試みとして，1980 年頃に Richard A. Bolt らの”put-that-there”[4] が登場した．その時は，本格的なジェスチャ操作には至らなかったが，近年では盛んに研究がおこなわれている．例えば，机上に投影されたコンピュータの画面を手や指を用いて机上で操作するインタフェースの研究がある [22, 30]．また，手をマウスとみなして実環境にある家電などの操作を試み [24] もされている．

その中でも特に盛んに研究されている内容として大画面とのインタラクションがある．先にも述べたとおり，ハンドジェスチャを用いることで離れた場所からでも操作を行うことができるため，大画面を操作するのに適切なインタフェースであるといえる．

Daniel Vogel らは指さしをした位置にポインタを移動させ，また指を曲げるなどのジェスチャをすることでクリックなどの操作を行うインタフェースを設計している [18]．また，手の形状に合わせてポインタの移動方法を変えたり，ポインタのキャリブレーションを行ったりすることが可能である．

また，木村らは広視野ディスプレイとハンドジェスチャ操作を組み合わせたインタフェースについて考察，試作している [29]．この研究では，大画面環境に適した作業について考察を行い，その作業を元に必要な作業を分類している．また，その分類に基づいてハンドジェスチャを割り当て，実際にビデオの編集を行っている．

このように，ハンドジェスチャを利用することで，様々な操作を行うことができ，大画面環境でのインタラクションの幅を容易に広げることが可能である．しかし，インタラクションの幅を容易に広げることができるために様々な問題が発生する．次節で，ハンドジェスチャを利用したインタラクション手法における問題点について議論する．

2.3 ハンドジェスチャによるインタラクションの問題

2.3.1 利用するハンドジェスチャ

ハンドジェスチャをインタラクション手法として利用する際には，どのジェスチャを利用するのが良いかということが重要な問題となる．手の形状一つとっても，図 2.1 のようにさまざまな種類が存在する．そのため，先に述べたとおり，操作一つ一つにジェスチャを割り当てることは容易である．しかし，割り当てるジェスチャの数が多くなるに従って，利用者が覚えなければならないジェスチャが増加してしい，それに伴って，利用者の作業効率が低下してしまう恐れがある．

また，利用するジェスチャについても簡単なジェスチャや行う操作と結びつきやすいジェスチャでなければ覚えるのが大変である．そのため，操作しやすいインタフェースを設計する上で利用するジェスチャは覚えやすいコマンドセットにする必要がある．特に普段のデスクトップ環境ではあまり行われない操作（例えばポインタのキャリブレーションなど）の場合には，その操作を覚えるだけで大変であるため，分かりやすいジェスチャ(もしくは操作と結びつきやすいジェスチャ) を利用しなければならない．

現在行われているハンドジェスチャの研究の多くは，ジェスチャが複雑であったり，操作と結びつきにくいものである．そのため，慣れるまでに相当の時間がかかってしまう恐れがある．また，操作に必要なジェスチャを思い出すのに時間がかかり，その度に操作を止めてしまうため，作業効率の面でも疑問が生じる．

そのため，Vogel や木村らが設計したインタフェースでは操作に必要なジェスチャの数が多かったり，特殊な操作があったりするため，必ずしも使いやすいインタフェースであるとは言い難い．

以前，我々も手を握るなどといったハンドジェスチャを利用したインタラクションを設計したことがあった [27]．しかし，すべてのジェスチャが操作と結びつきやすいものではなかったため，操作に慣れるまで時間がかかってしまった．また，両手を利用したため，片方の手で操作しているときにはもう片方の手まで注意が回らず，それがもとで誤動作するということが多々見られた．

2.3.2　手の認識の問題

ハンドジェスチャを行う上で，手の位置やジェスチャの認識を行うことは必要不可欠である．しかし，この認識についてもいくつか問題がある．

手を認識する方法としては様々な手法があるが，その認識手法は大まかに 2 種類に分けることが可能である．ひとつは，画像処理やパターン認識などを組み合わせることで手に何もデバイスをつけずに認識を行う方法である．もうひとつは，手にセンサなどをつけたデバイスを装着させ，そのデバイスから発せられる情報を利用して認識を行う方法がある．

デバイスを利用しない場合

手に何もデバイスを装着させずに認識を行う場合，デバイスを装着することへの抵抗感や圧迫感がないため，利用者にとっては利用しやすいインタフェースすることが可能である．

デバイスを利用しない場合，カメラなどの画像を利用して手の部分のみを抽出することで素手やジェスチャの認識 [23, 25, 31] を行っている研究が多い．しかし，素手の認識には様々な問題がある．例えば，肌色抽出を行う際に閾値などを利用して抽出を行うが，どんな人でも確実に抽出できるような閾値の設定は不可能であるため，利用するたびに調整が必要になる．また，光源や背景などの環境の変化が発生するとそれだけで正しく認識できなく恐れがある．さらに，3 次元座標や複雑なジェスチャを認識 [33] しようとした場合，設備が大規模になり設置に手間がかかる上に，キャリブレーションなどの調整が必要になる．そして，処理が複雑になるため，処理に時間がかかったりしてしまう．そのため，認識精度や処理時間などにどうしても不安が残ってしまう．

デバイスを利用する場合

デバイスを利用する場合，データグローブや磁気センサなどが付いた手袋を利用者に装着することになる．デバイスを利用によって，素手の時と比べて正確に認識することができ，また処理も早くすることが可能である．しかし，デバイスを装着するという行為そのものに問題が発生する恐れがある．手全体を覆うようなデバイスを装着する場合，利用者に拘束感をどうしても与えてしまう．また，デバイスを装着した状態で画面とのインタラクション以外の作業を行うのが難しいといった問題もある．さらに，認識精度や処理効率がいくら良いと言っても，複雑なジェスチャを利用しようとした場合，どうしても設備が大規模かつ複雑になるため，デバイスを装着させない場合と同様に設置やキャリブレーションなどの調整が必要になってしまう．また，設置コストなどもどうしても高くなってしまう．

2.3.3 連続操作

ハンドジェスチャを利用したインタラクションにおいてもうひとつ問題になることとして，連続操作が行いにくいといった問題がある．ハンドジェスチャを利用した場合，ある操作から次の操作に移るのに時間がかかってしまう．特に，手の形状を利用してハンドジェスチャを行う場合，手の形を変えるのにどうしても時間がかかってしまう．それによって，操作にかかる時間が長くなってしまう．

そこで，これらの問題点を解決する方法の一つとして，ハンドジェスチャのみで操作を行うのではなく，ハンドジェスチャを利用してメニュー操作を行い，そのメニュー操作を通じて大画面とのインタラクションを行うという方法が考えられる．

2.4 メニュー操作の利用

前節で上げた問題を解消する手段として，ハンドジェスチャの種類や数を少なくすることがあげられる．そこで，メニューを利用して操作を行うようにすることで，必要なジェスチャの数を減少させることが可能である．しかし，従来の Windows などの GUI 環境で利用されているメニューをハンドジェスチャで利用することは困難である．

2.4.1 従来の GUI 環境におけるメニューの利用

なぜなら，マウスの環境ではポインタを一定箇所に安定させることは容易であるので，狙った場所でクリックなどの操作を行うことができる．そのため，Windows などの GUI 環境におけるポップアップメニューなどの従来のメニューを選択することは容易なことである．しかし，マウス以外の環境ではポインタを一定箇所に安定させてとどめておくことは難しい．

特に，ハンドジェスチャを利用したインタラクションでは特に難しいといえる．手を空中でかざして操作する場合，かざしている手を固定することは困難であり，どうしても手ブレ

が発生してしまう．それらの要因は，ある程度であればプログラムで除去することは可能ではあるが，完全に取り除くのはほぼ不可能であるといえる．

さらに，ハンドジェスチャを利用したインタラクションの大半は，ポインタの移動を行う場合，手の動き，または指をさすなどハンドジェスチャを割り当てる．しかし，メニューの決定等の操作についてもハンドジェスチャで行われるため，誤認識などによって他のジェスチャを行った際にポインタが移動する恐れが高く，メニュー操作自体が困難である．

そのため，ハンドジェスチャを利用してメニュー操作を行うためには，ポインタを一定箇所にとどめておく必要のない操作手法を利用する，またはポインタを介さずに操作することができる手法が必要になる．

2.4.2 ペンベースのインタラクションにおけるメニュー操作手法の利用

このような解決策の一つとして，ペンベースのインタラクションで利用・研究されているメニュー操作手法 [5] を利用する方法がある．例えば，CrossY[2] では，ペンのストロークを利用してターゲット (メニュー) とクロスすることでクロスしたターゲットを選択することができ，また一筆書きの要領で複数のメニューを選択することが可能である．ペンベース向けのメニュー操作手法では，ペンのストロークを利用して操作を行うものが多いため，そのペンのストロークをハンドジェスチャを利用して模することで利用することが可能であり，またポインタを一定個所にとどめる必要もないため，従来の GUI 環境と比較しても操作が行いやすい．しかし，ペンのストロークを手で完全に模することは非常に難しいため，ハンドジェスチャ向けにメニューを改良するなどのといった工夫が必要になる．

ペンベースのメニューインタフェースをハンドジェスチャ向けに改良した例として，Sören らはハンドジェスチャと Marking Menu[14] などを利用して家庭内にある家電製品を操作するインタフェースがある [15]．この研究では，指さしした方向を利用して Marking Menu の操作を行っており，少ないジェスチャで多数のコマンドを実行することができる．しかし，この手法では，指さしの認識確認してから次の操作へと移るため，Marking Menu の連続的に操作をすることができるという利点が失われている．つまり，ハンドジェスチャを利用してペンベースのメニューを操作する際には，そのメニューの良さを保ちながらハンドジェスチャで利用しやすいように適応させる必要がある．

我々は，ハンドジェスチャとメニューの組み合わせ次第で，操作が分かりやすく，また様々な操作を流れるようにかつ連続的に行うことが可能なインタラクション手法が設計可能ではないかと考えた．そこで，我々は，2 種類のメニュー操作手法に注目した．ひとつは，ペンベースのメニューインタフェースである FlowMenu[9, 10, 11] の操作時に描かれる軌跡を利用した操作手法である．もう一つは，クロッシングと呼ばれる CrossY などのペンベースのインタフェースでよく用いられる操作手法である．本研究では，手の動きを利用することで，2 種類のメニュー操作手法の特徴を生かすことができると考え，それらを利用したインタラクション手法の設計を行うことを試みた．そして，それによって大画面環境でもスムーズなインタ

ラクションを行うことができるのではないかと考えた.

2.5 本研究の提案

本研究では，手の動きのみを利用してメニュー操作を行い，そのメニュー操作を通じて大画面の操作を行うインタラクション手法を提案する．メニューの操作手法として，ペンベースのメニューインタフェースである FlowMenu の操作時に描かれる軌跡を利用したメニュー操作手法と，クロッシングを利用したメニュー操作手法を利用し，実際にそれぞれの手法を利用した 2 種類のインタラクション手法を設計・試作した.

本手法を利用することで，大画面環境でも手の動きのみで様々なインタラクションを行うことができ，また流れるような連続操作も行うことが可能である．さらに，手の位置を高精度に認識するために，利用者にデバイスを装着させることにしたが，このデバイスについてもできる限りシンプルなデバイスにし，利用者への拘束感を軽減させた.

2.5.1 提案するメニュー操作手法の特徴

本研究では，手の動きによる軌跡を利用したメニュー操作とクロッシングを利用したメニュー操作の 2 種類を利用しているが，それぞれの特徴は以下のとおりである.

- **手の動きによる軌跡を利用したメニュー操作**
  FlowMenu の操作時に描かれる軌跡の特徴として，軌跡自体が短くまた操作の開始地点と終了地点がほぼ同地点である．そのため，一回当たりのメニュー操作は短時間で行うことができ，また次の操作へと移りやすいため，流れるような連続操作を行うことが可能である.

- **クロッシングを利用したメニュー操作**
  クロッシングを利用する場合，操作対象となるアイコンやメニューをクロスするだけで操作を行うことができるため，操作自体はわかりやすい．また，意図しない挙動による誤動作を防ぐための対処はクロッシングの回数を調整することで容易に行うことができる.

2.5.2 デバイスの利用について

手の動きを利用した 2 種類のメニュー操作手法について説明する前に，本研究で利用しているデバイスについて説明する.

本研究では，手の認識を高精度かつ高速におこなうためにデバイスを用いて認識を行っている．しかし，グローブなどの手全体を覆うようなデバイスでは利用者に拘束感を与えかねないため，図 2.2 のような指先の身に装着する LED デバイスを作成した．図 2.3 は実際に指に装着している様子である．このデバイスは市販の LED ライトを指先につけられるようにし

図 2.2: 作成した LED デバイス

図 2.3: 指先に装着した様子

たもので利き腕の人差し指に装着してもらう．本研究では，この指先につけた LED を利用して手の認識を行っている．また，図 2.3 のように指一本のみに LED をつけてもらう形で非常にシンプルであるので，デバイスを装着させることで利用者に与えてしまう拘束感などをかなり軽減できると思われる．

なお，本研究で作成したシステムでは，LED デバイスを装着しなければ操作ができないようなものではなく，携帯電話のライトなどといった強い光を放つものを利用しても十分に操作を行うことが可能になっているため，柔軟性が高いシステムとなっている．ただし，LED デバイスを利用して LED を指先に固定させることにより，より安定した認識を行うことができるようになっている．

次章より第 8 章にかけて 2 つの手法によるメニュー操作手法について説明する．まず，第 3 章と第 4 章で軌跡を利用したメニュー操作手法について説明する．次に，第 5 章と第 6 章でクロッシングを利用したメニュー操作手法について説明する．最後に第 7 章と第 8 章でそれぞれの手法に関する考察を述べる．

第3章　手の軌跡を利用したメニュー操作

第 2 章であげられた問題を解決する方法として，本研究では，手の動きによる軌跡を利用する方法とクロッシングを利用する方法の 2 種類の手法を提案した．本章では，手の動きによる軌跡を利用する方法について説明する．

本研究では，手の動きの積み重ねによってできる軌跡をメニュー操作に利用している．また，その際 FlowMenu の操作時に描かれる軌跡を手の動きで模することで一つ一つの操作をスムーズに行うことが可能になるる．

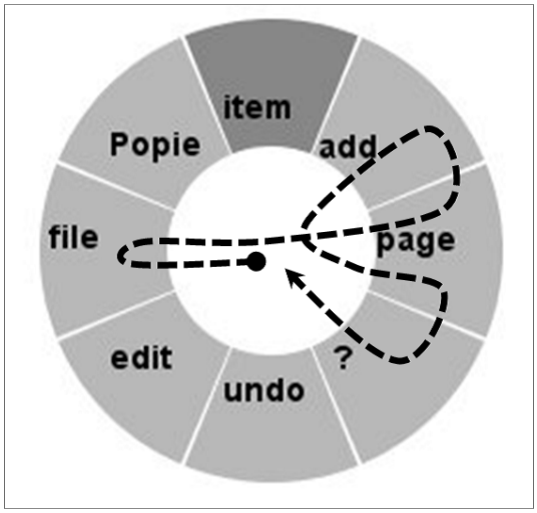

図 3.1: FlowMenu の外観と実際に描かれる軌跡の例

3.1　FlowMenu について

FlowMenu はもともとはペンベースのメニューインタフェースである．図 3.1 のような外観をもち，ペンが画面に触れた位置を中心にして展開される．そして，メニューが表示されているときにペンを動かすことで実際のメニュー操作が行われる．また，一回のペンストロークで複数のメニュー操作を行うことができるため，流れるような連続操作が可能である．

操作の全体の流れとしては，図 3.1 の矢印のような軌跡を描き，一つ一つの操作では，図 3.2 の (1) から (3) のように中心“ C ”から始まり，再び中心“ C ”に戻るという軌跡を描く．図 3.1 のような軌跡では，実際には図 3.2 の (1) から (3) の 3 つの軌跡を描いていることになる．こ

のように，一つの操作が終了した直後に次の操作に移ることができるため，流れるような操作が可能である．

FlowMenu は階層構造をとることが可能であり，理論的には無限階の改装を構成することが可能である．しかし，一般的には操作性の観点から 2 階層で利用されることが多い．例えば，図 3.2 の (1) の軌跡を描いた場合には，“ W ”のメニューが実行される．図 3.2 の (2)・(3) の場合には，メインメニュー“ E ”のサブメニュー“ NE ”((3) の場合は“ SE ”) が実行される．

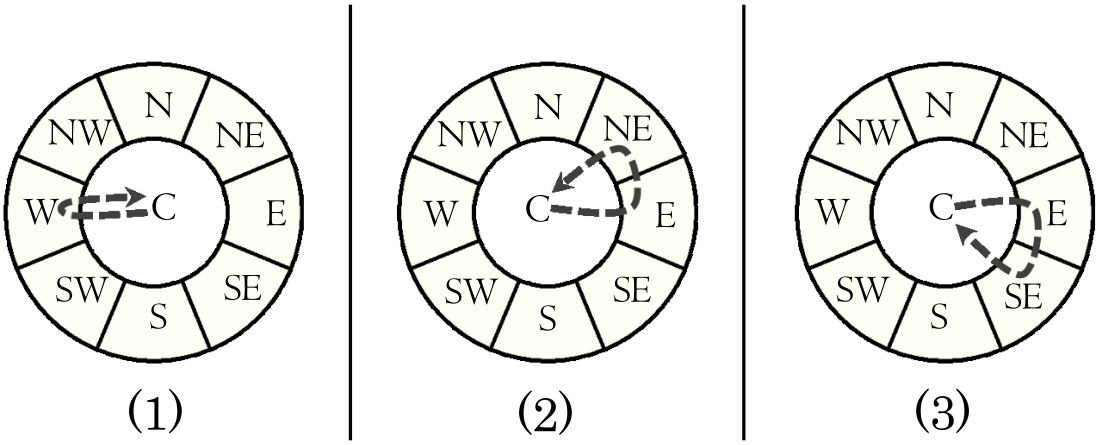

図 3.2: 一つの操作で描かれる軌跡の例

3.2 軌跡を利用したメニュー操作

本研究では，手の動きで FlowMenu の操作時に描かれる軌跡を描くことでメニュー操作が容易に行うことができると考え，手の動きを利用することで FlowMenu の操作を行うことにした．さらに，手の動きで FlowMenu の操作時に描かれる軌跡を模しているため，FlowMenu の特徴でもある流れるような連続操作も容易に行うことができる．

3.2.1 利用した軌跡

本手法では，FlowMenu の一つ一つのメニュー操作時に描かれる軌跡を利用するが，基本操作として次の 3 パターンの軌跡を利用することにした．

1. 図 3.2 の (1) のようにある方向に移動してそのまま元の位置に戻る
2. 図 3.2 の（2）のようにある方向に移動して隣接したメニューアイテムを通るようにして中心に戻る

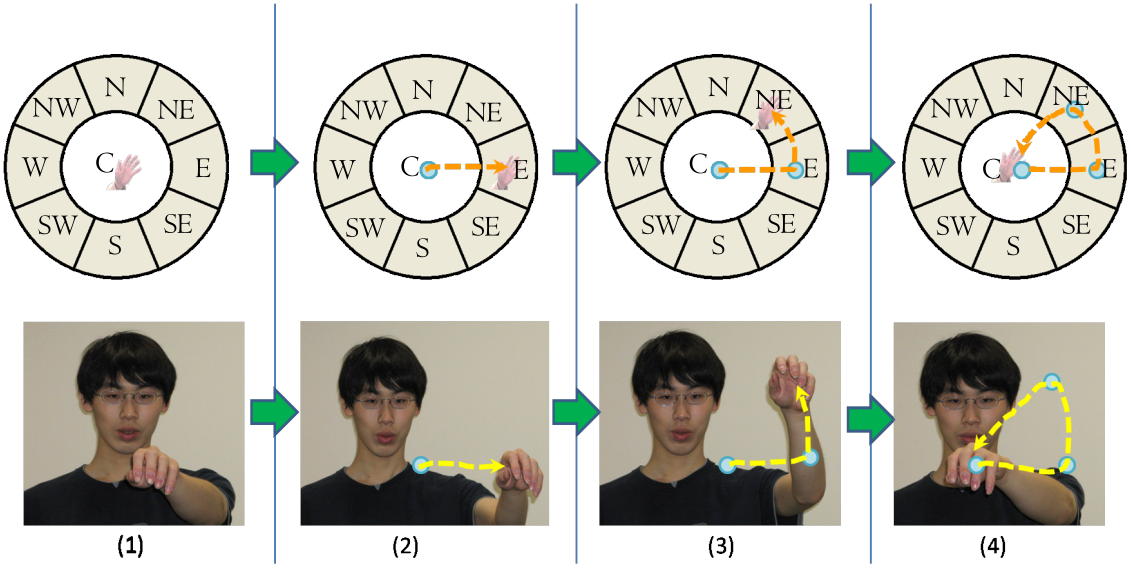

図 3.3: 手が描く軌跡の例

3. 図 3.2 の (3) のように，ある方向に移動したのち，図 3.2 の（2）とは別の隣接するメニューアイテムに移動して元の位置に戻る

この 1 から 3 までのパターンを各方向について行う．メニューの方向，つまりメニューアイテムの数は任意に設定することは可能であるが，利用する方向の数が少ない場合は，確実の操作が可能であるが利用できるメニュー操作の数が限られてしまう．反対に方向が多い場合は，利用できるメニュー操作の数は多くなるが誤認識による誤動作が起きやすくなってしまう．FlowMenu は，ペンで利用する場合には，8 から 12 が妥当であるといわれている．そのため，手を利用した場合についても 8 から 12 程度がよいのではと考え，本研究では，基本は 8 方向にした．

つまり，本手法において，メニュー操作の基本となる軌跡の全パターン数は，（8 方向）×（3 パターン）= 24 パターン となる．一見すると軌跡の数が多く感じられるかもしれないが，基本の動きは 3 種類しかなく，それが各方向にあるだけなのでその基本の動きである 3 つのパターンさへ覚えるだけで問題ない．また，図 3.1 のようなメニューを利用者側に表示するため，どの方向にどのメニューがあるかということを覚えていなくても問題ないようになっている．

3.2.2 メニュー操作の流れ

実際の操作は，図 3.3 の (1) のように手を構えた状態から (2) から (4) のように動かして FlowMenu で描かれる軌跡を模することで操作を行う．図 3.3 の例は図 3.2 の (2) の動きと同じであるため，メインメニュー“ E ”のサブメニュー“ NE ”が実行される．図 3.3 のようにメ

ニュー操作時の手の開始地点と終了地点はほぼ同地点であるため，図 3.3 の (4) の状態からすぐに次のメニュー操作を行うことが容易である．

図 3.3 では分かりやすいように大きく手を動かしているが，実際には手首の動きで描ける大きさ程度の軌跡でも十分である．

3.3 連続操作

一言で連続操作と言っても 2 種類の連続操作が考えられる．ひとつは，異なるメニューを短時間で複数選択することである．文字入力についても，一つ一つの文字をメニューと考えると，異なるメニューを短時間に連続操作していると捉えることが可能である．

もうひとつの連続操作としては，同じメニューを何度も選択・操作することである．例えば，画像の拡大・縮小の時の倍率の増減や回転角度の調整などといったパラメータ調節などがあげられる．このとき，メニューを選択して何度も同じメニュー (+のボタンなど) を選択して調節を行う必要がある．また，画面のスクロールや Web ブラウザにおける進む・戻るやタブの切り替えなども連続して行うことがある．この場合についても，何度も同じメニューが選択されることになる．

手の動きの軌跡を利用して操作する際には，異なるメニュー操作を行う場合，その各メニュー操作に必要な軌跡は異なることが多いため，連続操作を行う場合にはそれぞれの軌跡を手の動きで描けばよいので特に問題はない．

しかし，同じメニューを選択することによる連続操作の場合には問題が生じてしまう．なぜなら，同じ軌跡を何度も描き続ける場合，どうしても軌跡の形が少しずつずれ始めてしまう．そして，そのずれによって誤認識による誤操作が発生してしまう恐れが高い．そのため，連続操作の種類によって手法を変える必要がある．

そこで，本研究では，連続操作に次のようにして対応することにした．

- 異なるメニューを連続で選択するときは，それぞれのメニューに対応した軌跡を描くようにする．
- 同じメニューを連続で選択するときは，まず最初にメニューに対応した軌跡を描き，その後は手を回すように動かすことで何度も同じメニューを実行できるようにする．

手を回すような操作については，カメラに向かって手を回し，その手が一定角度以上回った場合に選択した操作を実行する．その際に，手を回す半径が大きければ，実行の頻度を高く，つまり少ない回転量で実行できるようにした．さらに，“ 増やす・減らす ”,“ 進む・戻る ”などといった反対の操作も同じ手を回す操作をしているときに行えるようにした．例えば，時計回りに回しているときには増やす (進む) 操作を行い，反時計回りに回しているときは減らす (戻る) の操作が実行されるといった感じである．これによって，パラメータの微調整などを容易に行えるようにした．

第4章　手の軌跡を利用したメニュー操作の実装

本章では，第 3 章で提案した手法をもとに作成したシステムの実装について説明する．

4.1　システム構成

システム全体の構成は 1 台のカメラと PC，ディスプレイ (大画面) から構成される．また，利用者は，第 2.5.2 節で説明したようなデバイスをに利き腕の人差し指に装着する．

図 4.1: 構成図

カメラや利用者などの位置関係は図 4.1 のようになっている．ユーザの正面に USB カメラを設置してユーザの動きを撮影し，その映像を処理することによって手の軌跡を得る．これらの処理はすべて 1 台の PC で行われる．

実装環境は，Visual Studio2005 で，VC++を用いて実装を行っている．また，カメラのキャプチャや画像処理には Intel 社の OpenCV(Intel Open Source Computer Vision Library) を利用した．

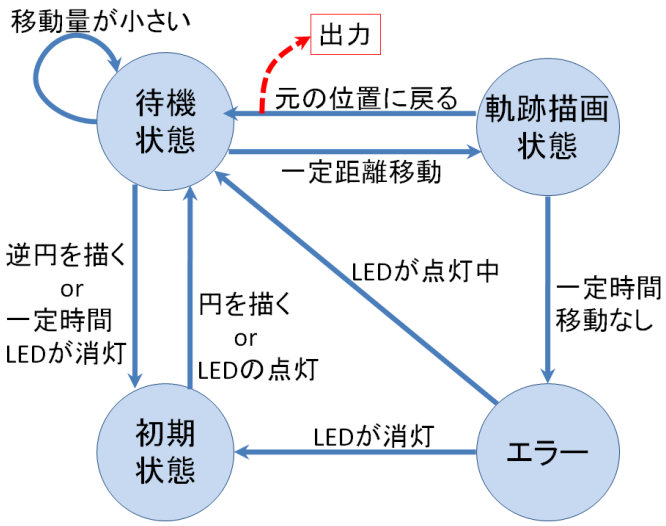

図 4.2: システムの状態遷移

4.2 全体の流れ

本システムにおいて，ジェスチャ操作の状態は図 4.2 のように遷移する．ジェスチャ操作を開始していないときは初期状態であり，その状態でジェスチャ操作を開始する動作を行うと待機状態へと遷移する．待機状態では，手が一定の距離を移動したと判断した場合，軌跡を描き始め，軌跡描画状態へと遷移する．そして，軌跡を描き始めた位置に手が戻ってきたと判断したら，待機状態に戻る．そして，そのときの結果によってメニューの操作が行われる．

パターンマッチングによる認識を行う場合には，軌跡描画状態から待機状態に遷移するときにそれまでに描いた軌跡をもとにパターンマッチングを行う．また，手の位置とメニューの位置を対応づける手法の場合は，軌跡描画状態のときに常に手の位置関係を調べている．

軌跡描画状態のときにある場所で一定時間動きがない場合は，エラー状態となり，そのとき描いていた軌跡はリセットされ，画面の操作などは行われない．また，手の位置が認識されている場合は待機状態へ，そうでなければ初期状態へと戻る．

メニューの中心の設定は，基本的には初期状態，もしくはエラーから待機状態へと遷移する時に行われる．また，待機状態中に移動量が小さいと判断した場合もメニューの中心の移動が行われる．連続的な操作が行われている際には，待機状態から軌跡描画状態に遷移する時に行われる．

4.3 手の位置の認識

手の位置の認識方法については，キャプチャしたカメラ画像から LED を検出することで行う．カメラ画像中の LED の検出方法については，次のとおりである．

1. 各画素について，あらかじめ撮影しておいた背景画像との差分をとる
2. 背景差分時にその画素の RGB 値および輝度を調べる
3. 各画素の背景差分値，RGB 値・輝度が閾値以上の画素を LED の光っている点とする．
4. 3. で該当したすべての座標の重心をとり，その重心を手が認識された点として扱う．

閾値については，差分の閾値については 30 前後で適当であると思われる．RGB 値および輝度の閾値については，利用する LED の色に合わせて適宜設定する．例えば，赤色の LED を利用した場合には，赤の閾値を高くして，青と緑の閾値は低く設定することで認識精度を容易に向上させることが可能である．また，シャッタースピードを速くしたり，カメラの明るさを調整することでも認識精度の向上を行うことができる．

4.4 メニューの中心領域について

本システムで描かれる軌跡は，開始地点と終了地点がほぼ同地点である．そのため，開始地点付近をメニューの中心領域とすると，軌跡の開始と終了はこのメニューの中心領域が基準になる．

しかし，空中に手をかざした状態で操作するため，手を動かすにつれて位置が少しずつずれ始めてしまう．そのため，開始時に設定された位置のまま固定した場合，手とメニューの中心との相対的な位置関係がずれ始めてしまう．つまり，手の動きに合わせてメニューの中心領域を移動させる必要がある．

そこで，待機状態であまり手が動いていないときにはメニューの中心をずらすようにした．具体的には，一定時間内の手の動いた量を調べ，その動いた量が一定値以下であった場合に現在の手の位置座標がメニューの中心領域の真ん中になるように位置をずらす．これによって，手ブレなどの影響によるメニューの中心領域のずれによって生じる誤操作を防ぐことができる．

4.5 軌跡の認識

本手法では，メニュー操作を行うに当たって利用者がどのような軌跡を描いたのかの判断を行うことが必要になってくる．軌跡の認識手法としていくつか考えられるが，本研究では，手が認識された位置からどの状態にあるかを推移してそれによって大まかな軌跡の形を調べる方法と，手が描いた軌跡全体で判断する 2 種類の手法を考えた．

4.5.1 手の現在の位置のみから判断する方法

手の現在の位置から描いた軌跡を判断する方法では，手が認識領域のどの位置で認識されたかということからどのように動いたかを判断し，それによってメニューの操作を行う．こ

の手法を用いる利点として考えられることとして，手の現在位置を利用して逐次現在の状態を調べるため，軌跡を描いている途中の状態を常に得ることができるため，途中経過を利用者側に与えることが容易であるため，現在の状態を確認しながら操作することが可能である．

領域の分割について

カメラの認識領域全体が図 4.3 の黒い矩形全体としたとき，その認識領域をメニューの中心部分と各方向のメニュー (本システムにおいては 8 方向) に分割する．分割した状態が図 4.3 である．そして，分割した各領域については，中心領域を“ C ”とし, “ C ”の周辺の領域を上から時計回りに“ N ”“ NE ”･･･“ NW ”とラベル付けする．

しかし，第 4.4 節で述べたようにメニューの中心領域は利用者の状態に合わせて変化する．そのため，これらの領域は常に固定ではなく，メニューの中心領域の変化に応じて領域の分割も変化する．

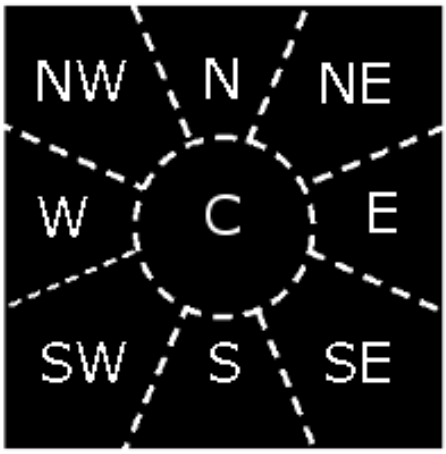

図 4.3: 領域の分割

判断の方法

この方法では，手の認識された位置が分割された領域のどの領域で認識されたかということを利用して手の大まかな動きを得ている．例えば，図 6 の“ C ”の領域内で手が認識された場合，現在の状態は“ C ”となる．この状態から手が“ E ”の領域へと移動した場合は，現在の状態が“ E ”となる．そして，このときの状態は“ C ”→“ E ”と遷移し，この遷移が実際の手の動きとして認識される．

本手法では，中心から動き始めて中心に戻るという軌跡で一つの操作となるため，最初は必ず“ C ”の領域から始まる．そのため，軌跡が区切れたか否かの判別については, “ C ”→ ･･･ →“ C ”という状態の遷移，即ち“ C ”の領域から出発して再び“ C ”の領域に戻った時に一つの軌跡が区切れたと判断する．

4.5.2　手が描いた軌跡全体で判断する手法

この手法では手を認識した位置を逐次蓄積しておき，その蓄積した位置情報から描いた軌跡を作成する．そしてその得られた軌跡とあらかじめ用意しておいたパターンとのマッチングを行うことによって，どの軌跡を描いたのかを判断する方法である．

手の位置の蓄積の開始・終了

描いた軌跡を把握するために特定の条件を満たしたら手の位置の蓄積を開始もしくは終了するようにする必要がある．

そこで，手の位置の蓄積の開始条件として，手の動き方から判断することにした．手の動きが一定時間に一定距離以上動いていた場合に，軌跡を描くためにい手を動かしていると判断して，手の位置の蓄積を開始する．また，手が蓄積を開始した地点の近くに戻ってきたと判断したら，蓄積を終了し，それまでに蓄積した点を利用して実際に利用者が描いた軌跡を取得する．そして，その軌跡と用意してあるパターンをを利用してパターンマッチングを行う．

軌跡の正規化

手の動きによって得られた軌跡とパターンテンプレートの比較を行うために，ある程度大きさを合わせる必要がある．得られた軌跡やパターンテンプレートのもとの軌跡の点列を P とし，各点を $(x_{p_i}, y_{p_i})(i = 0, 1, 2...)$ としたとき，本研究では，軌跡の正規化を以下のようにして行っている．

1. 得られた点列から n 点抽出し，点列 Q を作成する．また，Q の各点を $(x_{q_j}, y_{q_j})(0 \leq j \leq n-1)$ とする．
   (a) 点列 P の始点と終点をそれぞれ，(x_{q_0}, y_{q_0})，および $(x_{q_{n-1}}, y_{q_{n-1}})$ とする．
   (b) 点列 P 全体を一本の長い直線としてとらえ，その直線を $n-1$ 等分したものを考える．また，その軌跡全体の長さを l_p とする．
   (c) (x_{p_0}, y_{p_0}) から (x_{p_i}, y_{p_i}) までの総延長を l_i とする．
   (d) $1 \leq k \leq n-2$ について，初めて $l_i \geq l_p * k/(n-1)$ となる点を採用し，$(x_{q_k}, y_{q_k}) = (x_{p_i}, y_{p_i})$ とする．
2. 1. で得られた点列 Q の縦と横の長さを調べる (それぞれの長さを $height$，$width$ とする)．また，同時に Q によって得られる軌跡を囲むような矩形の左下に当たる座標を調べ，それぞれ bx，by とする．
3. 調べた $height$・$width$ のうち大きいほうの長さが 1 になるように倍率 $ratio$ を設定する
   - $height > width$ ならば，$ratio = heihgt$

- それ以外の場合は，$ratio = width$

4. Q の各点 (x_{q_j}, y_{q_j}) について，左下の座標を引いたのちに，$ratio$ で割る．そして，その点を (x_{t_j}, y_{t_j})　$(0 \leq j \leq n-1)$ とする．

   - $x_{t_j} = (x_{q_j} - bx)/ratio$
   - $y_{t_j} = (y_{q_j} - by)/ratio$

5. (x_{t_j}, y_{t_j}) によってできる点列を T とし，T の各点を結ぶことでできる軌跡を正規化された軌跡として利用する．

軌跡のパターンマッチング

パターンマッチングで軌跡を認識する方法では，手の現在の位置から判断するのではなく，認識された手の位置情報を蓄積しておき，その蓄積した位置情報から描いた軌跡を作成する．そして，その軌跡が用意しておいたパターンのうちどのパターンに近いのかを調べ，その結果からどのような軌跡を描いたのかを判断する．

パターンテンプレートについて

パターンマッチングを行うに当たって，マッチング用のテンプレートを用意する必要がある．本研究では，まず，パターンごとに軌跡の点列データを 10 個ずつあらかじめ用意しておく．次に，その 10 個のデータそれぞれに対して正規化を行う．そして，10 個のデータの各点の平均をとり，その結果をテンプレートして利用する．利用するデータについては，必ずしも 10 個である必要はなく，必要に応じて増減してもよい．

マッチング方法

本研究では，マッチングの手法として DTW(Dynamic Time Warping)[32] と呼ばれる手法を用いた．この手法は，動的計画法の一種であり，2 つのパターンの距離を比較する際，順番どおりに 2 つの点の間の距離を調べるのではなく，片方のパターンのある点に対して，もうひとつのパターンの点は，その点からもっとも近いと思われる点を利用して，その 2 点の距離を計算する．

DTW について簡単に説明する．まず，得られたパターン P と k 番目のパターンテンプレート Q が存在するとき，その各点を x_{p_i} および $x_{q_i}(1 \leq i \leq n)$ とする．ただし，2 つのパターンの点列の数はあらかじめ合わせておく．そして，k 番目のパターンテンプレートとの距離を $dist_k$ としたとき，$dist_k$ を次式のようにして距離を求める．

$$dist_k(p_i, q_j) \quad = \quad f_{p_i q_j} + min \begin{cases} dist_k(p_{i-1}, q_j) \\ dist_k(p_{i-1}, q_{j-1}) \\ dist_k(p_i, q_{j-1}) \end{cases} \tag{4.1}$$

$$
\begin{aligned}
f_{p_i q_j} &= \begin{cases} d_{p_i q_j} & (i = j) \\ d_{p_i q_j} * penalty * |j - i| & (i \neq j) \end{cases} \\
d_{p_i q_j} &= \sqrt{(x_{p_i} - x_{q_j})^2 + (y_{p_i} - y_{q_j})^2} \quad (1 \leq i\ j \leq n, 1 \leq k \leq p)
\end{aligned}
\tag{4.2}
$$

上式のように，P の各要素と Q の各要素を昇順にマッチングすることで距離を得ている．式中の $dist_k(i,j)$ は (p_i, q_j) までの P と Q の最短距離をあらわし，f_{ij} は p_i と qj 間のユークリッド距離である．ただし，$i \neq j$ の場合，つまり P のと Q の要素が同じ番号の要素ではないときには $penalty * |j - i|$ を掛けた分だけ距離が大きくなる ($penalty$ の値は任意の値)．このように P と Q の要素間の距離 $dist_k(p_i, q_j)$ を調べていき，最終的な結果は $dist_k = dist_k(p_n, q_n)$ となり，このときの $dist_k$ の値が P と Q の 2 つのパターン間の距離となる．

本システムでは，手を動かしたことで得られた軌跡に対して，正規化処理を施し，その正規化した軌跡と用意したパターンとの距離を DTW で計算する．そして，計算した結果のうち最も短い距離であったものを実際に描いたパターンとして認識を行い，それに対応した操作を実行する．

4.6 連続操作

同じメニュー操作を何度も連続で行う方法で最も単純な方法は同じ軌跡を繰り返し描くことであるが，同じ軌跡を描き続けることは誤操作のもとになりかねない．そこで，パラメータ調節や同じ操作を何度も繰り返す場合には，手を回すように手を動かすことで連続で操作できるようにした．

連続操作の流れは図 4.4 のようになっている．図 4.4 の r_0，r_1，r_2，θ_1，θ_2，t_1 はそれぞれ閾値で，それぞれ $r_0 < r_1 < r_2$，$\theta_1 > \theta_2$ という関係になっている．また，p_1，p_2，r_{p_1}，r_{p_2}，θ の関係については，図 4.5 のとおりである．p_1 は前回操作が行われた時の手の位置 (初めて $r_{p_2} > r_1$ となった場合にはその時の手の位置) で，p_2 は現在の手の位置を示している．なお，O はメニューの中心である．

連続操作時は，手の位置がメニューの中心から一定距離以上離れ，かつ一定角度以上動く度に操作が実行される．連続操作の頻度については，2 段階に設定し，手の位置がメニューの中心から離れているほうが連続操作の頻度が高くなるように設定した．

連続操作時は，手を回す方向によって操作が若干異なるようになっている．“ スクロール ” などといった行う操作自体は同じであるが，移動量や増減量などといったパラメータの値が回す方向で異なる．基本的には，時計回りに回すことでプラスの値を，反時計回りの場合にはマイナスの値を設定しており，その設定されたパラメータ量を利用して操作が行われる．そのため，一回の連続操作で上下のスクロールやパラメータの微調整などを行うことが可能である．

連続操作中は図 4.6 のように現在の状態がバーで表示され，このバーの長さと色から現在の状態を判断することができる．連続操作を行うことができない状態の場合 ($r_{p_2} < r_1$) はバーの色が青色（図 4.6 の (a)）で表示され，連続操作時にはバーの色が緑もしくは赤で表示され

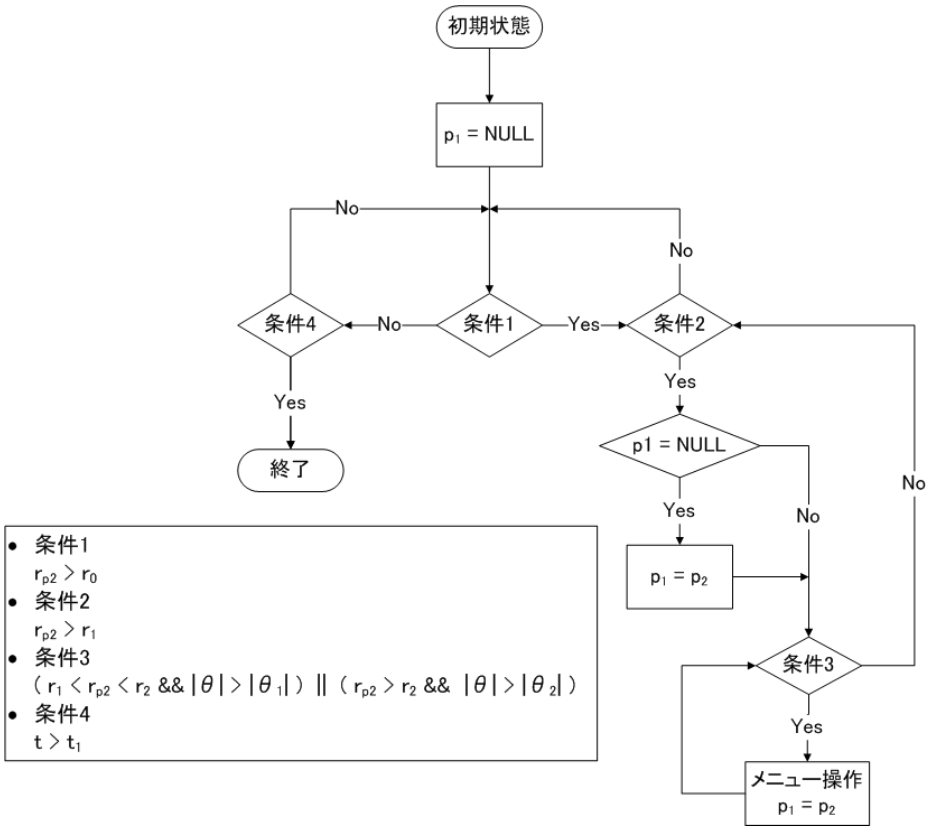

図 4.4: 連続操作の流れ

るようになっている (図 4.6 の (b) と (c)). なお，バーの色が赤の状態 ($r_{p_2} > r_2$) は緑の状態 ($r_{p_2} > r_1$) よりも連続操作の頻度が高い状態であることを示している.

連続操作の終了については，メニューの中心付近に一定時間滞留した場合 ($r_{p_2} < r_0$) に終了する.

4.7 フィードバック

利用者に，現在の状態などをフィードバックとして与えることは利用者自身が状態をしっかり把握する上で重要である．フィードバックとして与える必要があると思われる内容としては，手が現在認識領域のどの場所にあるかという情報や軌跡の描画開始・終了，描画途中のメニューの状態などが考えられる.

そこで，それらの情報をフィードバックとして与えるために，軌跡の描画開始・終了については，音を鳴らすことで利用者に提示することにした．この音については，開始と終了で異なる音を設定した.

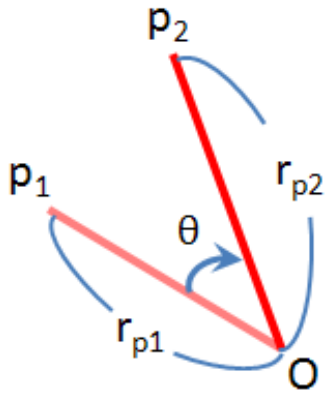

図 4.5: 連続操作時の回転角度の検出方法

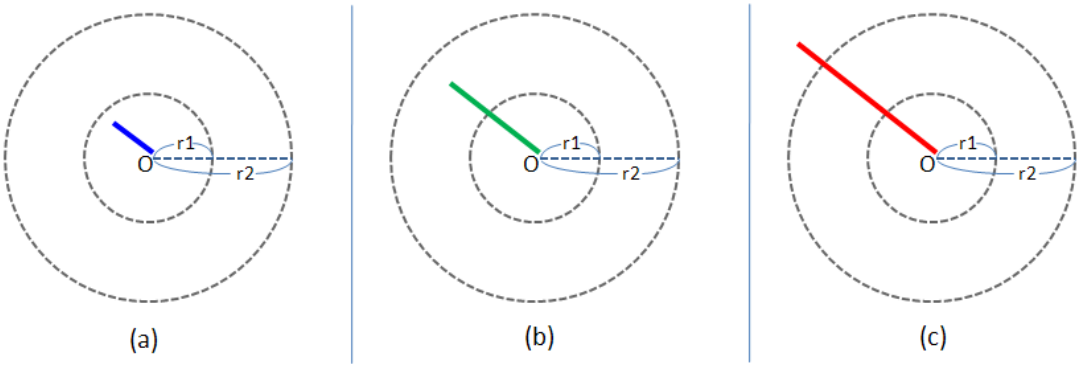

図 4.6: 連続操作を示すバーの状態

4.8 利用例

本研究で述べたシステムの利用例として，文字入力と Web ブラウジングをあげる．

4.8.1 文字入力

文字入力を行うことができるようになれば，検索などといったより様々なインタラクションを行うことができるようになる．また，ジェスチャで入力することができれば，キーボードなどの機器を利用する必要がないため，大画面上での文字入力もスムーズに行うことができる．

キーボードを利用せずに文字入力を行う方法のひとつとして，ソフトキーボードの利用が考えられる．しかし，ジェスチャを利用してソフトキーボードを操作する場合，文字の数 (メニューの数) が非常に多く，また一つ一つのメニューが小さいため非常に操作しづらい．

文字自体を空中で描くことで入力を行う方法 [19] もあるが，この方法では，一文字一文字ジェスチャで描かなければならないため，文字を入力するのに時間がかかる上に，ジェスチャ時の手の移動量が多くなってしまうため，疲れやすいといった問題がある．

そこで，本研究では，提案したシステムと佐藤らが開発した Popie[26] と呼ばれる文字入力手法を組み合わせることで文字入力を行うことにした．これによって，短時間で文字を入力することが可能になる．

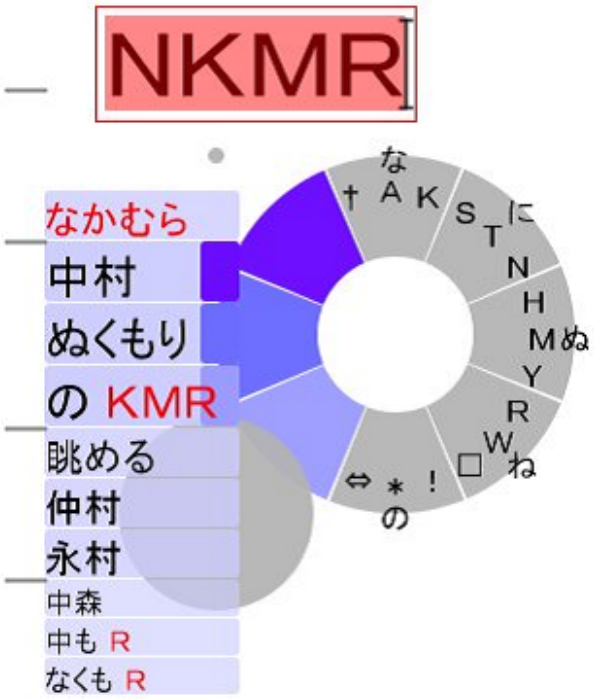

図 4.7: Popie の外観

Popie について

Popie は図 4.7 のような外観を持つインタフェースであり，元々はペンベースのインタフェースにおける日本語文字入力手法である．Popie の入力方法は，まず子音のみを入力する．すると，その入力した子音に対して予測候補を提示する．そして，その提示された予測候補から意図した文字列を選択することで文字入力が行われる．例えば“ 中村 ”と入力する場合には，子音“ NKMR ”を入力し提示された候補から意図した文字列を選択する．

Popie は本研究のジェスチャを用いた操作方法とも相性が良く，著者の行った簡単な実験では 25 文字／分程度の文字入力をすることが可能であった．

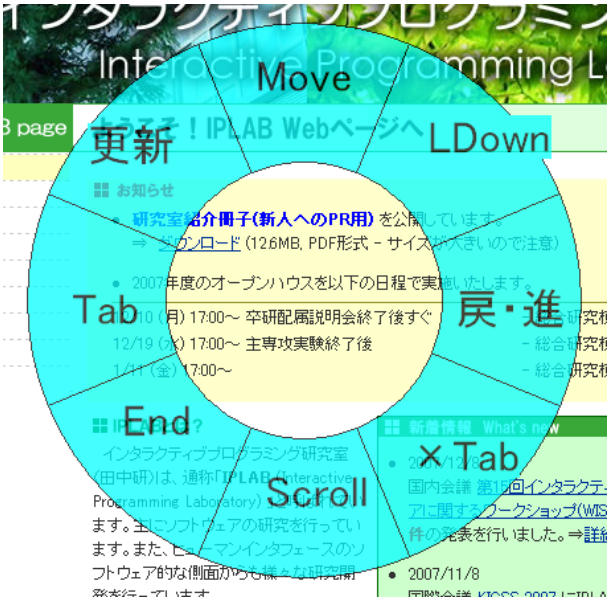

図 4.8: Web ブラウジング用のメニュー

操作方法

子音の入力方法については，従来の Popie の入力方法と同様に該当する軌跡を手で描くことで入力を行うようにした．しかし，変換候補のスクロールについては，該当するジェスチャを何度も描き続けると途中で誤認識による誤操作が起きやすくなってしまうため，第 4.6 節で説明したような連続操作手法を利用して変換候補のスクロールを行うことにした．

4.8.2 Web ブラウジング

Web のブラウジングについては，図 4.8 のようなメニューを用意した．このインタフェースでは，上方向から時計回りに“ ポインタの移動 ”,“ マウスの左クリック ”,“ ページの進む・戻る ”,“ タブを閉じる ”,“ 上下のスクロール ”,“ メニューを閉じる ”,“ タブの切り替え ”,“ ページの更新 ”といったメニュー操作を行うことが可能である．また,“ ページの進む・戻る ”,“ 上下のスクロール ”,“ タブの切り替え ”の 3 つのメニュー操作については，一回のみの操作と連続操作のどちらも行うことができるようになっている．

メニューの設定について

メニューの大まかな配置は図 4.8 の通りであり，基本的には，手を該当するメニュー方向に動かして，元の位置に戻るという軌跡を描けばそのメニューを選択・操作できるようになっている．しかし,“ ページの進む・戻る ”,“ 上下のスクロール ”,“ タブの切り替え ”の 3 つ

のメニューについては，2 通りの操作手法を行えるようにしてあるため，描いた軌跡によって一回のみの操作か連続操作かが決まるようになっている．手をその方向に動かして中心に戻るという軌跡を描いた場合は，連続操作が行われる．また，手を該当する方向に動かして隣接するメニューを通るように中心に戻る軌跡を描いた場合には，一回のみ操作が実行される．

連続操作時には，手を時計回りに回すことでそれぞれ“ 下方向へのスクロール ”,“ 進む ”,“ 次のタブ ”の操作が実行される．また，反時計回りに回した場合には,“ 上方向のスクロール ”,“ 戻る ”,“ 前のタブ ”の操作が実行される．

一回のみの操作の時は，連続操作時と同様に，時計回りの方向の軌跡を描いた場合には，それぞれ“ 下方向へのスクロール ”,“ 進む ”,“ 次のタブ ”の操作が実行される．反時計回りの場合には,“ 上方向のスクロール ”,“ 戻る ”,“ 前のタブ ”の操作が実行される．

第5章　手の動きとクロッシングを利用したメニュー操作

本章と次章では，本研究で提案するもう一つの手法でクロッシングを利用したメニュー操作手法，およびその試作システムについて説明していく．

この手法では，手の動きをジェスチャとしてとらえ，その動きに応じてポインタを動かすようにした．そして，そのポインタを利用してクロッシングを行うことで画面とのインタラクションを可能にした．また，専用のウィジェットを用意し，そのウィジェット上のアイコンをクロッシングすることで Windows などの従来の GUI 環境でも利用できるようにした．

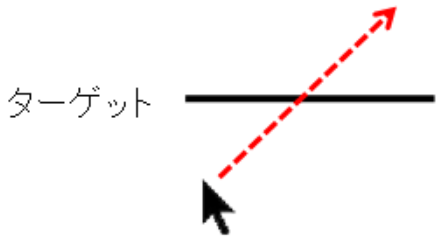

図 5.1: クロッシング

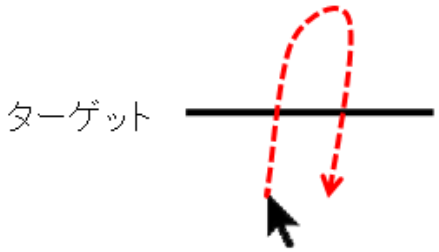

図 5.2: ダブルクロッシング

5.1　クロッシング

クロッシングにはいくつか種類があるが，本研究では，Accot らが提唱したゴール・クロッシング [1] を利用している．この手法では，図 5.1 のようにポインタなどでターゲット（ゴール）などを横切る（クロス）ことによって操作を実行する．クロッシング手法を利用する利点として，ポインタの動きを利用してクロッシングを行うことができ，操作手法として比較的わかりやすい．クロッシングのターゲットについてもボタンやウィンドウの縁など様々なものを設定することができるため，直観的な操作を行うことが可能である．

しかし，クロッシングの手法は元々はペンベースのインタフェースにおける操作手法として提案されたものであるため，そのままハンドジェスチャに利用しただけでは，誤動作などに十分に対応しきれない恐れがある．そこで，本研究ではハンドジェスチャを利用した際に操作が行いやすいように改良した手法を用いている．

5.1.1 ダブルクロッシング

ハンドジェスチャを利用した場合，ペンベースのインタフェースと比べると手ブレなどの影響を受けやすい．また，手を空中でかざして操作を行う場合には，先に述べたとおり手を一定箇所にとどめておくのは困難であるため，より顕著に手ブレの影響を受ける．そのため，利用者の意図しない挙動が起きやすいため，1 度のクロスだけではまだ誤操作が起こる恐れが高い．

そこで，誤操作をなるべく減少させるために，図 5.2 のように特定の操作については一定時間内に 2 度クロス（ダブルクロッシング）しなければ実行されないようにした．2 度クロスさせることで手ブレなどの影響による意図しない操作を軽減することが可能である．

5.2 ダブルクロッシングによるインタラクション例

専用のウィジェットを Windows 上にオーバーレイ表示し，そのウィジェット上にあるアイコンをクロッシングすることで，“ カット ”“ コピー ”などといった Windows のポップアップメニューにあるメニュー操作を割り当てることができる．また，一つ一つのメニューに文字を割り当てることで，キーボードを利用せずに文字入力を行うことも可能になる．しかし，文字入力を行う場合，キーボード中のキーをすべてメニューとして利用すると数が多くなりすぎるため，Popie で利用されているような子音入力などを利用して，入力の簡略化やメニューの数を減らす必要がある．

次章で，本章で説明したメニュー操作手法を利用したインタラクション例として，Web のブラウジングを行うためのインタフェース Hand-Bin について説明する．

第6章　試作システム:Hand-Bin

6.1　システム構成

基本的なシステム構成は軌跡を利用したメニュー操作のときと同様に 1 台の Web カメラと PC から構成される．利用者の指に LED を装着し，Web カメラからのキャプチャ画像からその LED を認識することで手の位置を調べている．Hand-Bin では，その手の位置を利用してポインタの移動を行い，また，そのポインタの移動を利用してクロッシングを行っている．

図 6.1: Hand-Bin の概観

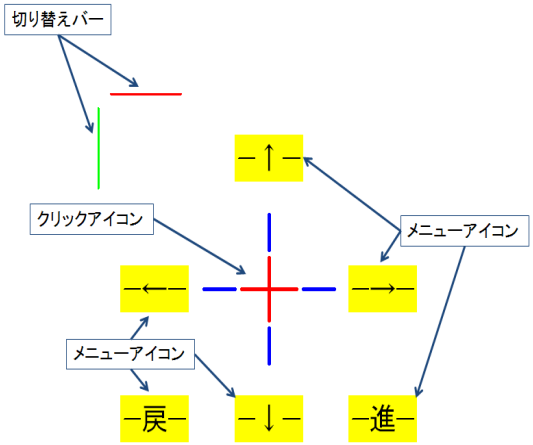

図 6.2: Hand-Bin の詳細

6.2　Hand-Bin の概観

Hand-Bin は図 6.2 の中心にある“ クリックアイコン ”(詳細は図 6.3) とその周囲にある 6 つの“ メニューアイコン ”(詳細は図 6.4)，およびそれらの表示状態を“ 切り替えバー ”(図 6.2 の左上の 2 本のバー) から構成される．Hand-Bin は図 6.1 のように通常の Windows の GUI の上にオーバーレイ表示して利用できる．そのため，従来の Windows などの GUI 環境でも容易に利用することが可能である．

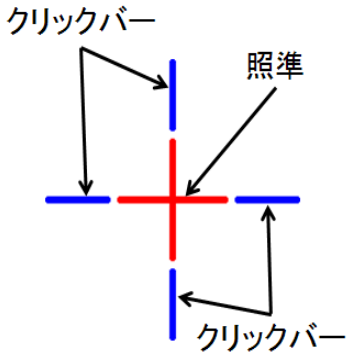

図 6.3: クリックアイコン

図 6.4: メニューアイコン

6.3 クリックアイコン

クリックアイコンには図 6.3 のようにクリックを行う場所を示す照準 (中心部の十字) とクリックの動作をおこなうためのクリックバー (照準の周囲にある 4 本のバー) からなっている．クリックの動作はクリックバーをクロッシングすることで照準の中心で行われる．ただし，誤動作を防止するために一定時間内に 2 度のクロッシング（ダブルクロッシング）を行うことによって動作を実行する．

6.4 メニューアイコン

ページの上下のスクロールやタブの切り替えなどはすべてこの図 6.4 のメニューアイコンを利用して行う．メニューアイコンごとに操作が定められている．Hand-Bin では，図 6.2 中の“ ↑ ”と“ ↓ ”のメニューアイコンで画面の上下のスクロールを行うことができる．また，“ ← ”と“ → ”のメニューアイコンでタブの切り替え，“ 戻 ”と“ 進 ”で Web ページの戻ると進むの操作を行うことができる．

メニューアイコンに割り当てられた操作を実行するために，メニューアイコンには縦に 2 等分するような形で実行線が設置されている（2 本に見えるが実際には 1 本の線である）．この実行線を一定時間内に 2 度クロスすることで操作が実行される．また，スクロールなどの操作はを連続して実行したい場合があるので，3 度目以降のクロスでは，クロスが起きるたびにその操作が実行される．なお，最後のクロスから一定時間経過した場合は，連続操作は終了する．

6.5 各種アイコンの切り替え

各アイコンの表示・非表示を切り替えは図 6.2 の左上にある 2 本の切り替えバーで行う．上方にあるバーでクリックアイコンを左方にあるバーでメニューアイコンの表示・非表示を切

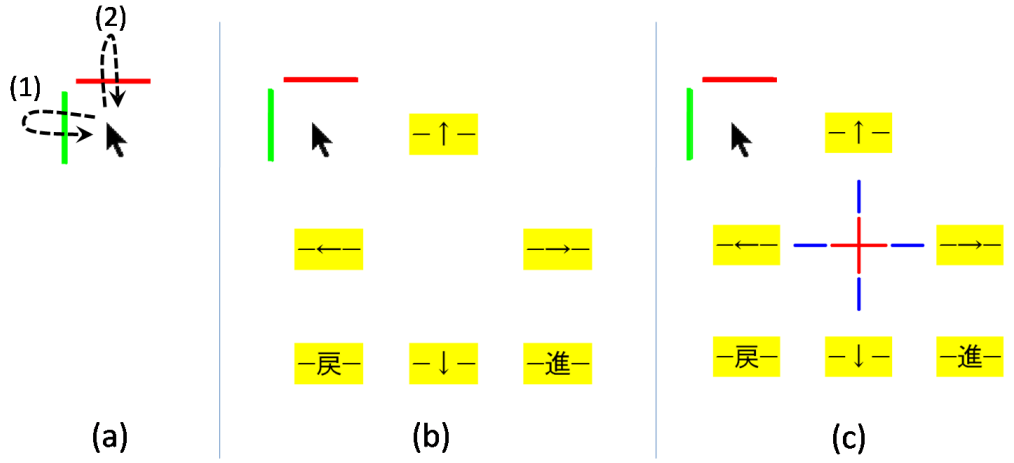

図 6.5: 各種アイコンの切り替え

り替えることができる．通常時は，図 6.5(a) のようにクリックアイコンやメニューアイコンは表示されておらず，左上の 2 本のバーのみが表示されている．その状態で図 6.5(a) の（1）のように左方にあるバーを一定時間内に 2 度クロスすると 6 つのメニューアイコンが表示される (図 6.5(b))．同様に，図 6.5(a) の（2）のように上方のバーを一定時間内に 2 度クロスするとクリックアイコンが表示される (図 6.5(c))．クリックアイコンやメニューアイコンが表示されている状態で，切り替えバーを 2 度クロスした場合は，クロスしたバーに対応したアイコンが非表示になる．例えば，図 6.5(c) の状態から図 6.5(a) の（2）の操作を行った場合，クリックアイコンが非表示状態にな（図 6.5(b) の状態に戻る）．

6.6　ポインタの移動方法

クロッシングの操作を行う上で，ポインタの移動が必要になる．そのポインタの移動方法として，手が認識された位置を利用してその位置に合わせてポインタの移動を行う．手が認識された点を (x, y) とした場合，ポインタを移動させる点 (px, py) は一般的に次式のようにあらわされる．

$$px = ScreenWidth * x/width \tag{6.1}$$

$$py = ScreenHeight * y/height \tag{6.2}$$

常識の $ScreenWidth$ と $ScreenHeight$ はそれぞれ画面の幅と高さ，$width$ と $height$ はカメラ画像の幅と高さである．ただし，実装環境が C++や C ＃などの場合は，画面の幅や高さに関係なく 1 画面あたり 0〜65535 の範囲であるため，横に並べたディスプレイの数を m，縦に並べたディスプレイの数を n とすると，次式で表現される．

$$px = m * 65535 * x/width \tag{6.3}$$

$$py = n * 65535 * y/height \tag{6.4}$$

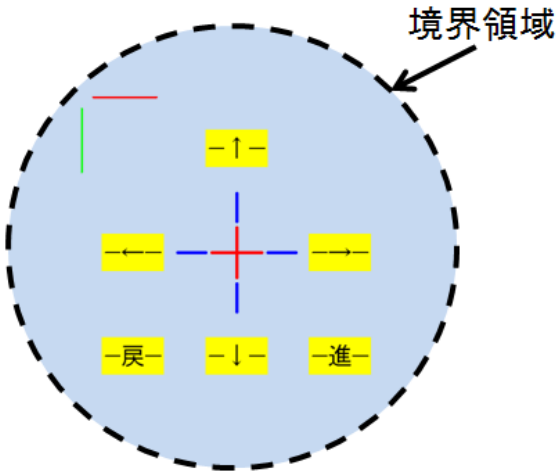

図 6.6: Hand-Bin の境界領域

6.7 Hand-Bin の移動方法

Hand-Bin のインタフェースは常にポインタの付近に表示されるようにした．また，Tracking-Menu[8] のようにポインタの動きに合わせて Hand-Bin のウィンドウも移動するようにした．Hand-Bin の移動方法については，境界領域を利用した方法とクリックアイコンの照準を利用した方法の 2 種類がある．

6.7.1 境界領域を利用した移動

Hand-Bin には図 6.6 に示した境界領域があらかじめ設定されている．ポインタは常にこの境界領域の内側に存在するようになっている．ポインタがこの境界領域を越えようとした場合，その境界領域を超えた方向と量に応じて，Hand-Bin のウィンドウ全体を移動させる．

6.7.2 クリックアイコンを利用した移動

クリックアイコンの照準でも Hand-Bin のウィンドウを移動させることができる．ポインタがクリックアイコンの照準部をクロスしようとした際，そのクロスした方向と量に合わせて照準部を移動させるように Hand-Bin ウィンドウ全体を移動させる．つまり，クリックアイコンがポインタに押されるようにして Hand ― Bin のウィンドウ全体が移動する．

6.8　フィードバック

Hand-Bin のフィードバックについては，軌跡を利用したメニュー操作手法と同様に音によるフィードバックを与えている．具体的には，何かしらのメニュー操作が行われる度に音が鳴るようにした．この音については，すべての操作で同じ音が鳴るようにしてあり，連続操作時にも操作が行われるたびに鳴るようになっている．

第7章　手の軌跡を利用したメニュー操作に関する実験

7.1　実験内容

被験者は著者の所属する研究室の男性 7 人と外部の男性 2 人の計 9 人である．いずれの被験者も 20 代の右利きで，本研究で提案している操作手法には慣れていない．

実験では，被験者に選択するメニューの順番を表示し，指定されたメニューの順番どおりに手を動かしてメニューの選択を行ってもらった．

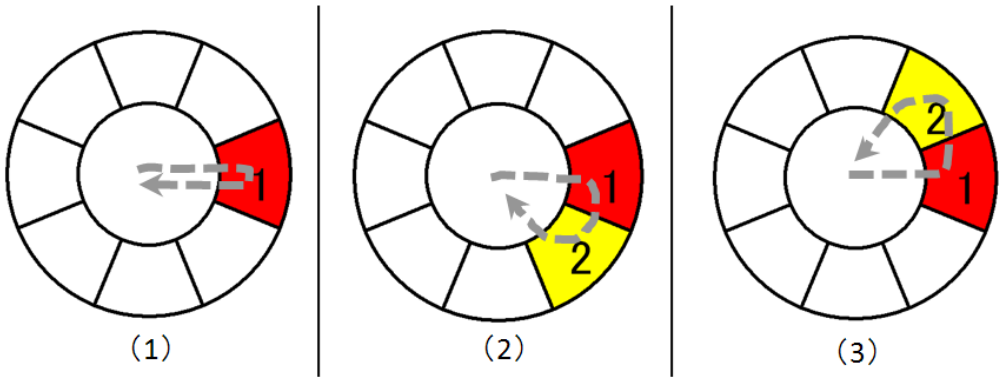

図 7.1: 実験で利用したメニューと描く軌跡の例

メニューの選択順番の提示方法は図 7.1 のように Flow-Menu 上に動かす順番を表示する．例えば，図 7.1 の（1）の場合は，右に行ったのち中心に戻る，図 7.1 の（2）の場合は．右に行って右下のメニューアイテムを通るようにして戻る，図 7.1 の（3）の場合には，右に行って右上のメニューアイテムを通るようにして戻るといった軌跡を描くことを意味する．実際に描かれる軌跡の例は図 7.1 の（1）から（3）の点線のとおりである．FlowMenu は 8 方向に分割されたものを利用し，それぞれの手法について,（各方向）×（3 パターン）×（2 回）×（2 セット）のメニュー選択を行ってもらった．セット間は被験者に自由に休憩をとってもらった．また，今回の実験では 2 つの認識手法で実験しているため全部で 4 セット実験を行ったため，一つのセットが終了したら，次のセットは別の手法で行うといったように，2 つの手法を交互に実施した．なお，実験に利用した PC の環境は，CPU が Pentium D 3.0GHz，Memory が 2GB である．

この実験では，メニューの選択にかかった時間とエラーの回数を計測した．メニューの選択時間は指定された軌跡が正しく認識されるのにかかった時間であり，誤認識などのエラーがあった場合には正しく入力できるまで繰り返し軌跡を描いてもらった．またエラー率は（誤認識があった回数）÷（全体の回数）である．

なお，被験者へのフィードバックは，手が現在どの位置にあるのかという認識結果とメニューの中心に当たる円の位置だけを表示した．描いてもらう軌跡の大きさについては，どちらの場合でも手首を動かすだけで描くことができる大きさである．また，パターン認識のためのパターンテンプレートはすべての被験者で同じテンプレートを使用した．また，議論の便宜上，パターンマッチングによる認識手法を利用した方を“ 手法１ ”手の位置とメニューの位置関係を利用して操作する手法については“ 手法２ ”と呼ぶことにする．

7.2 実験結果

図 7.2，図 7.4 はそれぞれ被験者ごとの入力時間およびエラー率を示している．各被験者の 2 つの手法について，8 方向それぞれの入力（3 方向 ×2 回 ×2 セット）について平均し，さらに平均した入力時間およびエラー率を示している．入力時間については，システム側の処理も含めた時間になっている．また，図 7.5 は，2 つの手法について方向別のエラー率の平均を集計したものである．なお，この方向は必要な軌跡を描画するときに最初に動かす方向のことである．

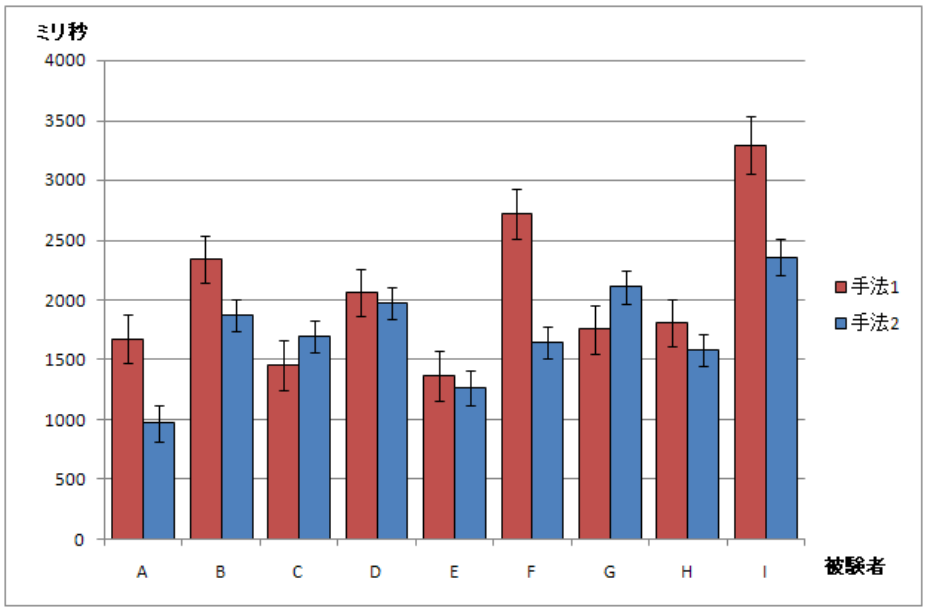

図 7.2: 被験者ごとのメニューの選択時間

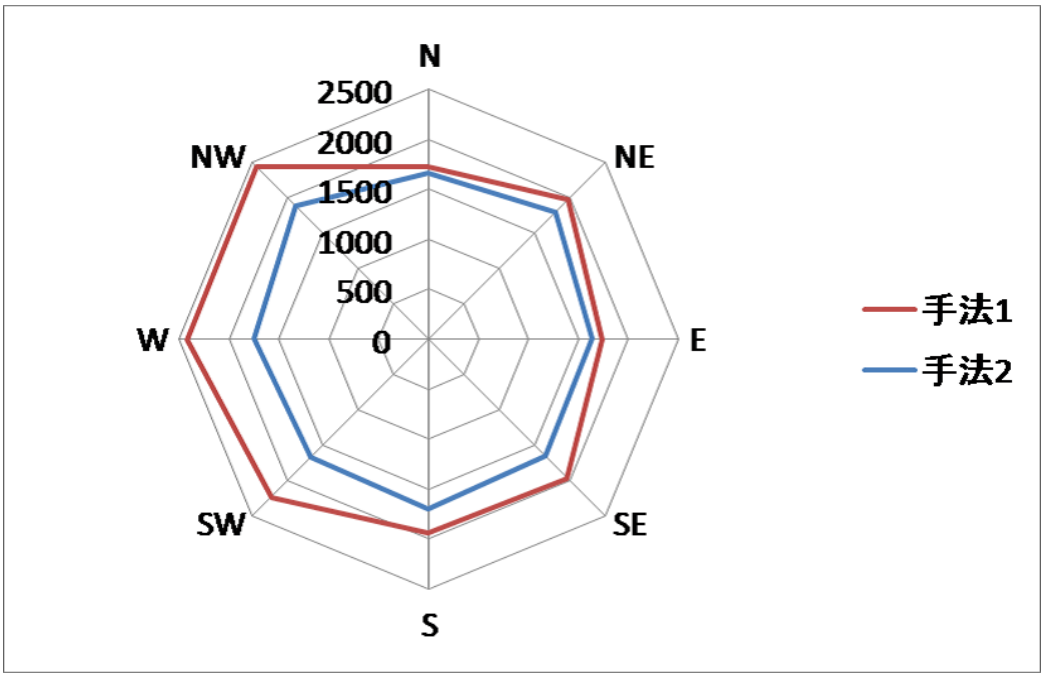

図 7.3: 方向別のメニューの選択時間

7.2.1 選択時間

選択時間については，大半の被験者が手法 2，つまり手の位置とメニューとの位置関係による選択手法の方が早い結果となった．被験者ごとでみると，2 つの手法であまり差が見られない被験者もあるが，半分近くの被験者で，500 ミリ秒近く差が見られた．

方向別で比較した場合，右方向 (“ NE ”“ E ”“ SE ”) ではほとんど差が見られなかったが，左方向 (“ SW ”“ W ”“ NW ”) では，手法 1 と手法 2 で 500 ミリ秒以上の差が見られた．

なお，全体の平均は手法 1 で 2052 ミリ秒，手法 2 で 1720 ミリ秒であり，2 つの手法の選択時間の差はおよそ 300 ミリ秒であった．

7.2.2 エラー率

エラー率については，被験者によって大きな差が見られた．およそ半分の被験者は手法 1，つまりパターンマッチング手法の方がエラー率が低かったが，被験者 F については，手法 1 の方がエラー率が高く，手法 2 と比較したときに倍以上の差が見られた．

方向別に見た場合，右方向 (“ NE ”“ E ”“ SE ”) と左方向 (“ SW ”“ W ”“ NW ”) を比較すると，利き手側とは反対側の方がエラー率が高いという結果となった．特に，手法 1 の方ではその傾向が顕著に見られた．また，上下左右 (“ N ”“ E ”“ S ”“ W ”) よりも斜め方向 (“ NE ”“ SE ”“ SW ”“ NW ”) の方がエラー率が高い傾向にあった．

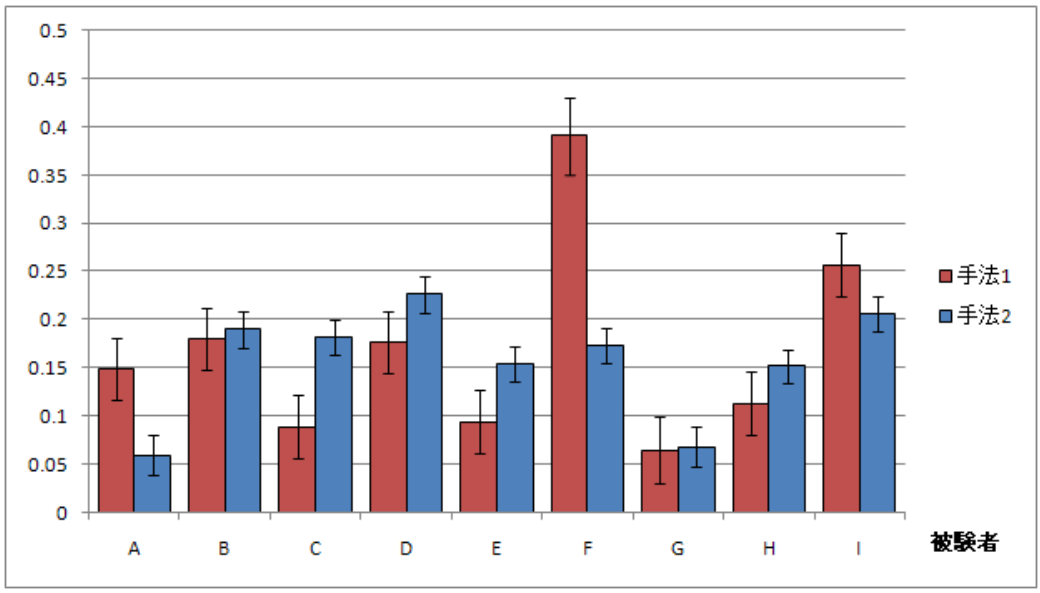

図 7.4: 被験者ごとのエラー率

なお，全体の平均については，手法 1 で約 0.168，手法 2 で 0.156 とあまり大きな差は見られなかった．

7.3 考察

メニューの選択時間を比較すると手法 2，つまり手の位置と領域の関係を利用したほうが早いように思われる．しかし，実験環境下で 2 つの手法の処理時間を調べて比較してみたところ，手法 2 はほぼリアルタイムで処理できているのに対し，手法 1 では軌跡の認識の部分で平均 150ms 処理に時間がかかっていた．実験で行った環境よりも遅い環境 (CPU:Celeron 2.53GHz, Memory:1GB) でも試したところ，軌跡の認識に約 250ms かかっていた．つまり，手法 1 は認識部分がボトルネックになっていたものと思われる．

150ms 処理に時間がかかってしまうと，30fps のウェブカメラを利用した場合には約 5 フレーム分は遅れる計算になる．たとえ認識部分をスレッド立てて別処理した場合，その 5 フレームの間に次の軌跡の描画に移ることは十分に可能である．しかし，その時に前に描いた軌跡が誤認識などのために意図しない操作を行い，描きなおす必要が出た場合，描画中の軌跡もキャンセルしなければならない恐れが出てしまう．今回の実験においてもこのような処理の遅れによって，意図しない操作が何度も起きてしまい，それがエラーを増長させる要因になっていた．

そのため，軌跡全体で判断する場合についても，認識の処理時間を短縮することができれば，操作時間のみではなくエラー率も大幅に減少できると思われる．図 7.6 にエラーした時も含めた 1 回あたりの選択時間である．図 7.2 と比較してみると，手法 1 における処理時間を

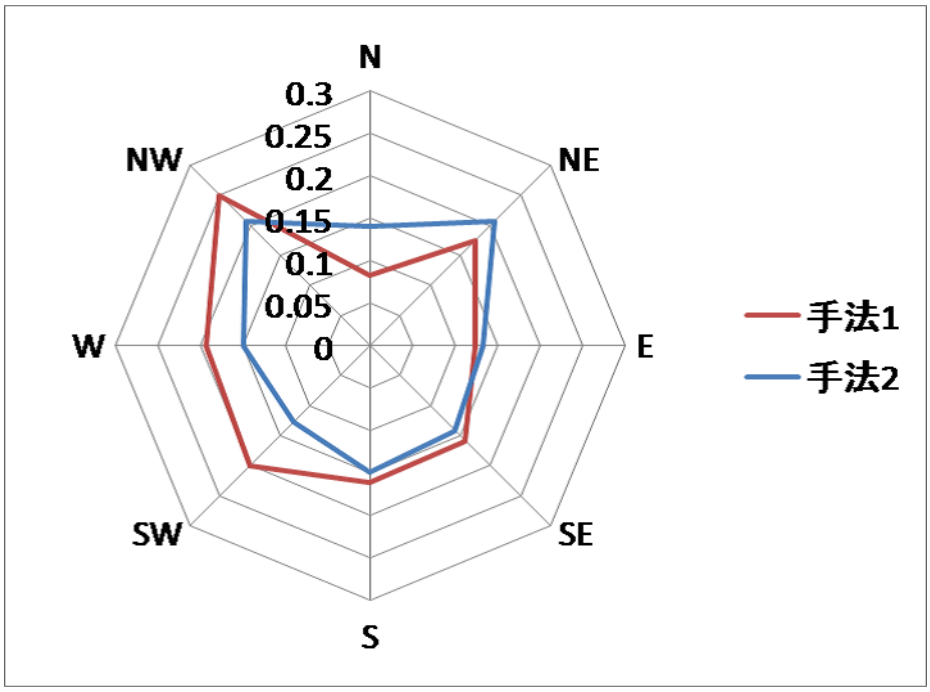

図 7.5: 方向別のエラー率

考慮すると多くの被験者は二つの手法で選択時間に大きな差は見られないことがわかる．

方向別に見た場合，手法 2 は全体的に同じ処理時間とエラー率であるが，手法 1 については利き手側（N・NE・E・SE・S）は手法 2 と同程度もしくは優れている結果になったが，利き手と反対側（SW・W・NW）は手法 2 よりも選択時間・エラー率ともに高い結果となっている．おそらく，聞き手の反対側の軌跡を描く場合，最初の方の軌跡の形がいびつになってしまい，うまく認識が行われなかったものと思われる．しかし，これらを考慮することができれば，もっとスムーズな操作を行うことができるのではないかと思われる．

7.4　システムの改良

本節では，第 7.3 節で議論された問題点を改善するためにシステムの改良方法について議論していく．

7.4.1　マッチング候補の絞り込み

DTW を利用した場合，処理時間は (マッチングするパターン数) と (マッチングに利用する点の数)^2 に比例するため，パターン数が多いとどうしても処理時間がかかってしまう．そのため，処理時間の短縮を行うためには最初に明らかに違う候補を除外した上で DTW を行うようにする．

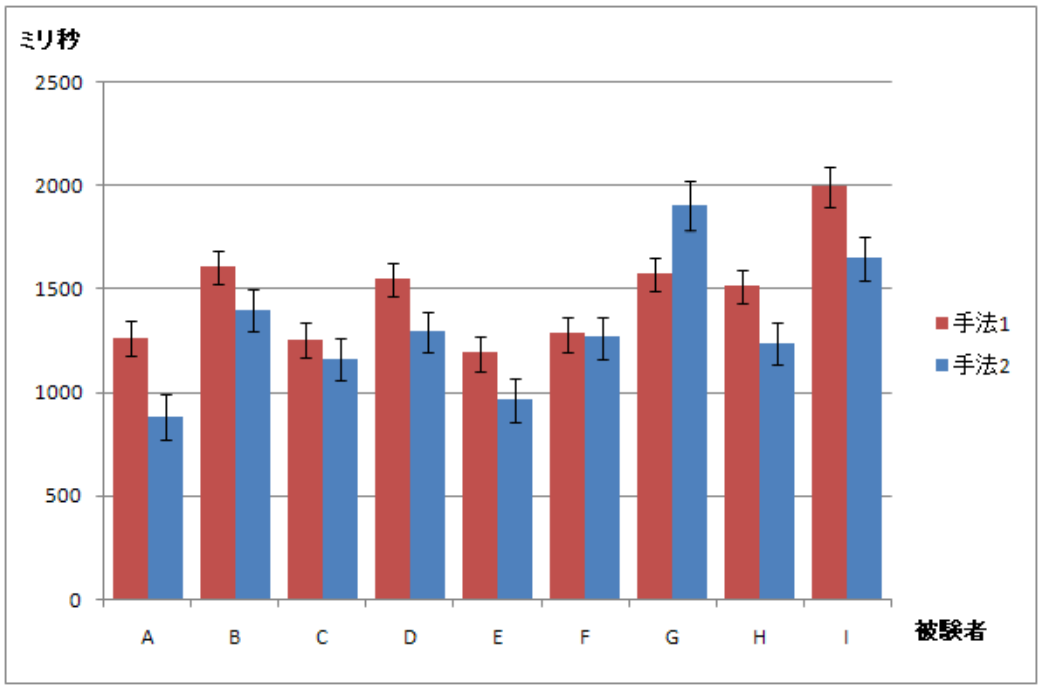

図 7.6: 被験者ごとの一回のメニューの選択時間

マッチングの候補の絞り込みの方法については，手の動きによって得られた軌跡を利用して行う．絞り込みの手順については，次のようにして行う．

1. 得られた軌跡を正規化したものの最初の数点から軌跡の最初の方の方向ベクトルを得る
2. 得られた方向ベクトルからその方向に該当するパターンを決定し，決定したパターンを利用してマッチングを行う．
    - その方向が明らかに特定のメニューの方向であると判断した場合には，その該当する方向の 3 パターンのみで DTW を行う．
    - どの方向かをはっきり判断できない場合 (例えば，方向ベクトルが“ N ”と“ NE ”の間を指している場合) には，その可能性のある方向のパターンすべてについて DTW を行う．

この方法によって，マッチングを行うパターンを最初に 3 から 6 つに絞り込むことが可能である．そのため，処理時間も従来の $\frac{1}{8}$ から $\frac{1}{4}$ に削減することができる．つまり，処理時間を 30ms 前後まで改善することができるため，あまり不快を感じることなく操作を行うことができる．

7.4.2　2 つの手法の組み合わせ

マッチングの処理時間を軽減させることで，軌跡を利用した手法のみでも十分な精度と選択時間を得ることはできると思われる．しかし，軌跡を利用した認識手法の最大の問題点として，軌跡の状態やメニューとの位置関係の途中経過を示すことが難しい点があげられる．よどみなくスムーズに手を動かして操作できているときには途中経過はあまり気にする必要はないと思われるが，途中で動きが止まった時などには現在の状態が表示されると利用者にとって使いやすいのではないかと思われる．

そこで，状況に合わせて 2 つの手法を切り替える方法を考えた．基本は手の動きの軌跡を利用した認識手法を利用して操作を行う．ただし，特定の条件を満たしたときは，それまでに得られた手の位置情報を利用して現在の状況を判断し，それ以降は一つの操作が終了するまでは手の位置と領域を利用した認識方法に切り替える．手法の切り替えを行うための条件はいくつか考えられるが，条件の一つとして，手の動きが途中で止まったかどうかで判断する方法が考えられる．

第8章　クロッシングを利用したメニュー操作に関する考察

8.1　照準を利用したクリック操作

本研究では，マウスのクリックに相当する操作を行うために，照準を利用してクリックを行う位置を合わせ，その照準がある場所でクリックの操作を行っている．ポインタを利用することによって生じてしまうブレなどを考慮するために照準を利用して位置を固定した後にクリックするようにしているが，ポインタ以外の場所でクリックが行うため，どうしても利用者に違和感を与えてしまう．特に，照準の位置の調節のためにポインタを利用して手動で調節しなければならないためであると思われる．

この違和感を解消するために，照準をある程度自動で位置調節できるようにするといった方法をとる必要があると思われる．例えば，Web ページの場合には，そのページを解析して現在のポインタの位置に近いリンクに自動的に照準を合わせるようにする．もしくは，ある程度ポインタと連動して照準を移動させるなどといった方法も考えられる．

8.2　メニューの設定・配置

現在の実装方法では，ウィンドウに表示させるメニューはあらかじめプログラム上で固定されたものであり，また配置についてもあらかじめ決められている．しかし，利用する人によって，使いたいメニューや場所が異なると思われる．そのため，利用者ごとにメニューの設定や配置を決められるようにするのが望ましいと思われる．

メニューの設定や配置は専用のアプリケーションを作成して，そのアプリケーション上でウィンドウを利用者が設計するようにすれば十分に対処できると思われる．しかし，そのようなアプリケーションを作成するのであれば，作成しやすいことも大切であるが，クロッシング専用のメニューウィンドウの作成ソフトにしてしまうのは非常に勿体ないと思われる．そのため，クロッシング意外に操作手法，例えば従来の GUI におけるクリックや軌跡を利用した手法など，にも適用できるようにすることが好ましいのではないかと思われる．

8.3　連続操作

連続操作についても一考の余地がある．現在の操作手法では同じ個所を何度もクロッシングすることで連続操作を行っている．この手法でも十分に連続操作には対応できると思われ

るが，操作としては直観的とは言い難い．

　連続操作を行うためのほかの手法としては，軌跡を利用したメニュー操作時に実装した連続操作手法，手を回すような動きで連続操作を行う手法も利用することが可能ではないかと思われる．しかし，一回の操作と連続操作を分けて実行できるようにした場合には，メニューの数が増えてしまい，メニューの配置を工夫したり，階層的にするなどの工夫が必要になってくると思われる．

第9章　まとめ

本研究では，手の動きとメニュー操作を組み合わせた大画面向けインタラクション手法の提案・試作を行った．また，手の動きを利用したメニュー操作手法として，手の動きによる軌跡を利用した操作手法とクロッシングを利用した操作手法の 2 種類の操作手法を提案した．これらの操作手法を利用することで手の動きのみで様々なインタラクションを行うことを可能にし，流れるような連続操作を行うことが可能である．

今後の課題としては，2 つの操作手法それぞれについて議論した内容を元にシステムの改善を行うことが挙げられる．また，2 つの操作手法についての比較も行う必要があると思われる．そして，2 つの操作手法を組み合わせることができるか検討し，新しい操作手法についても，設計・試作していくつもりである．

謝辞

本研究を進めるにあたって多くの方々から様々な方々から助言や指導をいただきました．特に，指導教員である田中二郎教授をはじめ，研究全体を通してご指導していただきました高橋伸講師，様々な助言をしてくださった三末和男准教授・志築文太郎講師には深くお礼を申し上げたいと思います．また，システムの実装にあたり，社会人ドクターである佐藤大介さんには多大なるご尽力を頂きました．心から感謝いたします．最後に，ユビキタスチームの皆様をはじめ，田中研究室の皆さん，そして私を支えてくださったすべての方々に感謝を申し上げます．

参考文献

[1] Accot, J. and Zhai, S.: More than dotting the i's - foundation for crossing-based interfaces, *CHI' 02*, pp. 73–80 (2002).

[2] Apitz, G. and Guimbretiere, F.: CrossY: A Crossing-Based Drawing Application, UIST '04, pp. 3–12 (2004).

[3] Atia, A., Takahashi, S. and Tanaka, J.: Coin Size Wireless Sensor Interface for Interaction with Remote Displays, *HCI International 2007*, pp. 733–742 (2007).

[4] Bolt, R. A.: " Put-that-there ": Voice and gesture at the graphics interface, SIGGRAPH '80, pp. 262–270 (1980).

[5] Callahan, J., Hopkins, D., Weiser, M. and Shneiderman, B.: An Empirical Comparsion of Pie vs. Linear Menus, *CHI'88*, pp. 95–100 (1988).

[6] Chen, X. and Davis, J.: Multi-user laser-based interaction on large tiled displays, *Displays*, pp. 205–211 (2002).

[7] Dhawale, P., Masoodian, M. and Rogers, B.: Bare-Hand 3D Gesture Input to Interactive Systems, *Chinz'06*, pp. 25–32 (2006).

[8] Fitzmaurice, G., Khan, A., Pieke, R., Buxton, B. and Kurtenbach, G.: Tracking Menus, *UIST'03*, pp. 71–79 (2003).

[9] Guimbretière, F., Martin, A. and Winograd, T.: Measuring FlowMenu Performance, Technical report, Stanford CS (2001). CS-TR-2001-02.

[10] Guimbretière, F., Martin, A. and Winograd, T.: Benefits of merging command selection and direct manipulation, *ACM Trans. Comput.-Hum. Interact.*, Vol. 12, No. 3, pp. 460–476 (2005).

[11] Guimbretiere, F. and Winograd, T.: FlowMenu: Combining command,text, and data entry, *UIST2000*, pp. 213–216 (2000).

[12] Hisamatsu, T., Shizuki, B., Takahashi, S. and Tanaka, J.: A novel click-free interaction technique for large-screen interfaces, *APCHI2006* (2006).

[13] Khan, A., Fitzmaurice, G., Almeida, D., Burtnyk, N. and Kurtenbach, G.: A remote control interface for large displays, *UIST'04*, pp. 127–136 (2004).

[14] Kurtenbach, G. and Buxton, W.: The limits of expert performance using hierarchic marking menus, *CHI'93*, pp. 482–487 (1993).

[15] Lenman, S., Bretzner, L. and Thuresson, B.: Using Marking Menus to Develop Command Sets for Computer Vision Based Hand Gesture Interfaces, *NordiCHI*, pp. 239–242 (2002).

[16] Malik, S., Ranjan, A. and Balakrishnan, R.: Interacting with Large Displays from a Distance with Vision-Tracked Multi-Finger Gestural Input, *UIST'05*, pp. 43–52 (2005).

[17] Miyaoku, K., Higashino, S. and Tonomura, Y.: C-Blink: A Hue-Difference-Based Light Signal Marker for Large Screen Interaction via Any Mobile Terminal, *UIST'04*, pp. 147–156 (2004).

[18] Vogel, D. and Balakrishnan, R.: Distant Freehand Pointing and Clicking on Very Large, High Resolution Displays, *UIST'05*, pp. 33–42 (2005).

[19] 保呂　毅，稲葉雅幸：複数カメラを用いた手書き文字認識システム，第 14 回インタラクティブシステムとソフトウェアに関するワークショップ，pp. 121–122 (2006).

[20] 塚田浩二，安村通晃：Ubi-Finger:モバイル指向ジェスチャー入力デバイスの研究，情報処理学会論文誌 Vol.43 No.12，pp. 3675–3684 (2002).

[21] 小山慎哉，小野寺光，葛岡英明，山崎敬一：ハンドマウスとその応用：色情報と輪郭情報に基づく手の検出と追跡，映情学技報, VIS2001-103,Vol.25,No.85，pp. 43–52 (2005).

[22] 小林貴訓，佐藤洋一，小池英樹：EnhancedDesk のための赤外線画像を用いた実時間指先認識インターフェース，*WISS'99*, pp. 49–54 (1999).

[23] 西村托一，向井理朗，　岡隆一：白黒動画像からの形状特徴を用いたジェスチャのスポッティング認識システム，電子情報通信学会論文誌 D-II Vol.J81-D-II No.8，pp. 1812–1821 (1998).

[24] 蔵田武志，興梠正克，加藤丈和，大隈隆史，坂上勝彦：レーザポインタによる遠隔作業指示支援システムの実用可能性，ヒューマンインタフェースシンポジウム 2000，pp. 435–438 (2000).

[25] 天田泰亨，鈴木基之，後藤英昭，牧野正三：動作位置の変化に頑健なジェスチャ認識，電子情報通信学会技術研究報告. PRMU, パターン認識・メディア理解 Vol.99 No.448，pp. 39–46 (1999).

[26] 佐藤大介，志築文太郎，三浦元喜，田中二郎：Popie：フローメニューに基づく日本語入力手法，情報処理学会論文誌, Vol.47, No.7，pp. 2305–2316 (2006).

[27] 中村　卓，高橋　伸，田中二郎：ハンドジェスチャを用いた公共大画面向けインタフェース，*DICOMO 2006*, pp. 833–836 (2006).

[28] 中村聡史，塚本昌彦，西尾章治朗：Protractor: 機能の可視化に注目した両手操作可能な図形描画システム，*WISS2000*, pp. 1–6 (2000).

[29] 木村朝子，鶴田 剛史他：広視野電子作業空間に関する考察とシステム試作〜マイノリティ・リポート型 I/F とその発展形，インタラクション 2005 論文集，pp. 143–150 (2005).

[30]　陳欣蕾，小池英樹，中西泰人，佐藤洋一，小林貴訓：机型実世界指向システムにおける両手による直接操作の評価，*WISS 200*, pp. 215–216 (2000).

[31]　畠直志，岩井儀雄，谷内田正彦：動き情報と情報圧縮を用いたロバストなジェスチャ認識手法，電子情報通信学会論文誌 D-II Vol.J81-D-II No.9，pp. 1983–1992 (1998).

[32] 櫻井保志，吉川正俊：ダイナミックワーピングのための類似検索手法，情報処理学会論文誌 Vol.45 No.SIG 4(TOD 21)，pp. 23–36 (2004).

[33] 林健太郎，久野義徳，白井良明：ユーザ位置の拘束のないジェスチャによるヒューマンインタフェース，情報処理学会論文誌 Vol.40 No.2，pp. 556–565 (1999).