

平成17年度

筑波大学第三学群情報学類

卒業研究論文

題目

ハンドジェスチャを用いた大画面向けインタフェース

主専攻 情報科学主専攻

著者 中村 卓

指導教員 田中二郎, 高橋伸, 三末和男, 志築文太郎

要 旨

近年、プラズマディスプレイやプロジェクタなどを利用した大画面が広く普及してきた。しかし、大画面とインタラクションを行うためのインタフェースは、マウスなどといった従来のインタフェースをそのまま利用しているのが現状である。しかし、それらのインタフェースは大画面環境において操作しにくいなどといった問題点が多く存在する。そこで、大画面を操作する方法として、ユーザのハンドジェスチャによる大画面向けインタフェースを提案する。

本研究では、ハンドジェスチャを用いて大画面とのインタラクションを可能にする大画面向けインタフェースを試作した。これにより、様々な環境において、大画面とのインタラクションが容易になる。また、それに伴い、様々な場所で大画面をより広く利用することができるようになると考えられる。

目次

第1章 序論	1
1.1 本研究の目的	2
1.2 本稿の構成	2
第2章 背景と問題点	3
2.1 大画面の普及と現状	3
2.1.1 大画面とは	3
2.1.2 大画面の普及	4
2.1.3 大画面の利用状況	4
2.2 従来の操作インタフェース	5
2.2.1 マウスを用いたインタフェース	6
2.2.2 タッチパネルを用いたインタフェース	7
2.2.3 リモコン	8
2.3 新しいインタフェース	8
第3章 手を用いた大画面向けインタフェース	10
3.1 手を用いたインタフェース	10
3.2 大画面向けのインタフェース	10
3.3 ゴール	11
第4章 提案	12
4.1 デバイス	12
4.2 相対座標指定と絶対座標指定	13
4.3 ハンドジェスチャ	14
4.4 両手での動作	15
第5章 プロトタイプ的设计と試作	16
5.1 プロトタイプ的设计	16
5.1.1 システムの構成	16
5.2 プロトタイプの試作	17
5.2.1 開発環境	17
5.2.2 手の処理	18
5.2.3 両手の同時処理	21

5.2.4	動作	22
第 6 章	評価と今後の課題	27
6.1	試用による評価	27
6.1.1	移動	27
6.1.2	文字入力	27
6.1.3	動作の確認	28
6.2	今後の課題	28
6.2.1	フィードバック	28
6.2.2	メニュー	29
6.2.3	アプリケーションへの利用	30
第 7 章	関連研究	31
第 8 章	結論	32
	謝辞	33
	参考文献	34

図目次

1.1	駅の大画面	1
2.1	従来のディスプレイ	4
2.2	複数のプロジェクタを利用した大画面	5
2.3	タッチパネルを用いたインタフェース	6
2.4	マウス	6
2.5	リモコン	6
3.1	利用イメージ	11
4.1	様々な手の形状	14
5.1	システム図	18
5.2	背景	19
5.3	キャプチャ画像	19
5.4	背景差分を取った画像	19
5.5	肌色抽出の結果	19
5.6	腕まで抽出される例	20
5.7	結果	20
5.8	原画像	20
5.9	図 5.8 に対して収縮・膨張処理を施した画像	20
5.10	手を握っている画像	21
5.11	図 5.10 に対して収縮・膨張処理を施した画像	21
5.12	走査領域の分割	21
5.13	基準点と手の領域	21
5.14	移動前	22
5.15	右にカーソルを動かしている状態	23
5.16	右手を握った状態	24
5.17	決定（マウスの左クリック）後	24
5.18	スクロール前	25
5.19	スクロール中	26
5.20	Popie のインタフェース	26

6.1	様々なカーソル	29
6.2	メニューバー	29
6.3	ポップアップメニュー	29
6.4	階層が深くなったメニュー	29
6.5	浮かんでいるメニュー	30
6.6	戻るボタンに近づいた場合	30

表 目 次

4.1 座標指定方法の分類	13
5.1 動作	17

第1章 序論

近年、プロジェクタやプラズマディスプレイを利用した大画面がオフィスや研究室など様々な場所に普及してきている。また、それに伴って、コンピュータの利用シーンも広がりつつある。しかし、一方で大画面とインタラクションするためのインタフェースはマウスやペンなどの従来利用してきたデバイスをそのまま使用しているのが現状であり、大画面を操作する上で不便を感じる事が非常に多い。また、駅の構内(図 1.1)や街角に置かれている大画面は、設置場所の理由などから従来のデバイスを用いたインタフェースを利用しにくいことが多い。そのため、そのような場所では、情報が表示されているだけのものがほとんどである。情報を表示するという点だけで見れば十分に利用されているが、見る側からは見る以外のことができないものがほとんどであるため、その情報に興味のない人以外には利用されない。図 1.1 のようにその地域の情報が表示されているものもあるが、中にはただ意味のない画像を表示しているだけで、情報も何も表示していないものもある。つまり、設置はされているがあまり有効に利用されていないというのが現状である。



図 1.1: 駅の大画面

1.1 本研究の目的

本研究では、様々な場所で大画面を有効に利用できるようにするために、ハンドジェスチャで大画面とのインタラクションを行う大画面向けインタフェースを提案する。様々な場所で大画面を利用できるようにするために、極力、ユーザ側にかかる負担を少なくする。ここで言う負担とは、ユーザに操作する上で特殊な手袋などの装着といった準備や利用するために訓練が必要になる等といったことを指す。本研究では、そのような負担を減らすために、手には何も装着せず、また、簡単なハンドジェスチャで大画面とのインタラクションが可能なインタフェースを提案していく。また、同時に、それによって大画面でどのようなことができるかを考察する。

1.2 本稿の構成

本稿では、まず、大画面と大画面とのインタラクションに利用されているインタフェースの現状と問題点について述べる。次に、それらの問題点を踏まえ、ハンドジェスチャを用いた大画面とインタラクションを行うためのインタフェースを提案し、それを元に試作したシステムについて述べる。その後、それによりどのようなことができるかについて考察し、最後に結論を述べる。

第2章 背景と問題点

近年、大画面は急速に普及してきている。しかし、大画面とのインタラクションはあまりうまく行えていない。そこで、まずは大画面の現状と大画面とのインタラクションを行うために現在利用されているインタフェースの問題点について述べる。

2.1 大画面の普及と現状

近年、プロジェクタや複数のディスプレイを用いた大画面が、公共の場に広く普及してきている。それに伴い、大画面で作業を行う機会も多くなってきている。しかし、その一方で、それら进行操作するインタフェースは、従来のサイズのディスプレイ (図 2.1) で用いられているマウス等を利用したインタフェースをそのまま使用しているのが現状である。

2.1.1 大画面とは

まずはじめに、大画面とはどのようなものであるかについて説明する。ただ、大画面といった場合、画面のサイズなど物理的に大きいということとピクセル数的に大きいという2通りが考えられる。基本的には、物理的なサイズが大きいとそれに伴ってピクセル数的にも大きくなる。しかし、近年は画面の物理的なサイズが小さくてもピクセル的には大きいディスプレイも存在する。

しかし、画面が小さいけれどピクセル的に大きいということはここではさほど問題ではない。なぜなら、物理的なサイズが小さくてピクセル数が大きいディスプレイは表示されるものをよりきれいに見せることが目的であるのに対して、物理的なサイズが大きいディスプレイはより多くの情報をより大きく見せるのが目的である。また、物理的に小さいディスプレイがおかれている環境では、一度の操作で画面の端から別の端まで動かすためにマウスを移動させるスペースが画面のピクセル数に関わらず十分にとることが出来る。そのため、本研究の目的は、きれいな画像などを操作することではなく、多くの情報をどのように操作していくかにある。

よって、ここでは特に物理的なサイズが大きいディスプレイに焦点を当てていく。これ以降、本稿では大画面とは物理的なサイズが大きいディスプレイを指す。

2.1.2 大画面の普及

大画面は、現在、幅広い場所で様々な形で普及している。例えば、学校の講義室ではプロジェクタとスクリーン、研究室ではプラズマディスプレイ、駅などでは複数のディスプレイの組み合わせる(図 1.1) ことで大画面を構成している。また、複数のプロジェクタを組み合わせることにより、壁一面をひとつの大画面として構成しているものもある(図 2.2)。このように、様々な種類の大画面が様々な場所に設置されている。

大画面が普及してきた要因として、大型ディスプレイやプロジェクタの低価格化や解像度の向上などによるものが大きい。最近では、ひとつのオフィスや研究室でプラズマディスプレイやプロジェクタを複数台所持しているところも決して珍しくない。

また、大画面の普及に伴い、大画面を操作する機会が増え、また同時に、コンピュータを利用できる空間が広がりつつある。様々な場所で利用できる空間が広がっているという意味では、Mark Weiser が提唱した “Ubiquitous Computer”[1] に近づきつつある。また、大画面のみが見えるだけで中のコンピュータなどは目に見えるところから消えつつあるので、D.A.Norman による “Invisible Computer”[2] にも近づきつつあるといえるだろう。



図 2.1: 従来のディスプレイ

2.1.3 大画面の利用状況

一方、大画面の利用状況については、大画面が置かれている場所によって大きく異なる。例えば、研究室やオフィスでは、大画面でプレゼンテーションを行ったり、デモンストレーション映像を流したりする。また、スマートボードやタッチパネルなどを利用して、会議などを行うなどといった使い道もある。このように、研究室やオフィスでは、大画面は様々なことに利用されていることがわかる。



図 2.2: 複数のプロジェクタを利用した大画面

しかし、駅の構内や街角においてある大画面について見た場合、大画面には広告やニュース等が流れてはいるが、それらは一方的に情報が流されているだけであり、その情報を見る側からは操作を行うことができないものがほとんどである。また、中にはただ映像を流しているだけのもの存在する。そのため、情報を表示する点においては利用されているものも多いが、実際はあまり有効に活用されていないのが現状である。タッチパネルが設置されている場所もあるが、タッチパネル自体は大画面に設置されているわけではなく、また大画面の隣に設置されている従来のサイズのディスプレイに設置されているため、大画面上で何か操作するというわけではない。

なぜ、駅の構内や街角に置かれている大画面はあまり利用されないのだろうかと考えたとき、その原因は、大画面を操作するためのインタフェースにあるのではないかと考えた。研究室やオフィスでは、マウス等を用いた従来のインタフェースを利用するための空間をとることは容易である。しかし、駅や街角では、人の往来などの問題からマウスを利用するための空間をとるのは容易ではない。そのため、研究室やオフィスなどで利用可能なインタフェースの多くが利用できない。

そこで、これらの問題点について、従来のインタフェースでは何がよくないのかを考える。

2.2 従来の操作インタフェース

ここでは、現在、大画面で利用されているインタフェースについて、いくつか見ていく。現在、利用されている大画面操作インタフェースは、マウス（図 2.4）やタッチパネル（図 2.3）



図 2.3: タッチパネルを用いたインタフェース



図 2.4: マウス



図 2.5: リモコン

を利用したものが多い。そこで、まずは、この2つのインタフェースについて見ていく。また、家庭にあるテレビやプロジェクタを操作するのによく利用されるリモコン（図 2.5）についても見ていく。

2.2.1 マウスを用いたインタフェース

マウスは、従来のサイズのディスプレイで非常によく利用されるインタフェースであり、また使い慣れているため、現在、大画面においてもよく利用されるインタフェースの一つである。マウスは、今まで利用されてきたインタフェースであるため、誰でも簡単に利用することができるという点においては非常に優れている。しかし、マウスは机の上で、かつ従来のサイズのディスプレイ上でのみで有用であり、大画面上で利用する場合、環境や動作が異なってくるためいろいろと不便を感じることもある。その理由は次のとおりである。

大画面は、机の上に置かれることはまずなく専用の台に置かれるか、もしくはプロジェクタによってスクリーンに映し出されることがほとんどである。また、画面が大きいため、全域を把握するためにはある程度離れて利用しなければならない。つまり、画面が置いてある台の上やスクリーン上でマウスを利用するのは困難であり、マウスを利用するためには離れた場所に別にマウスを使うためのスペースが必要になる。そのため、ワイヤレスマウスなどを利用する必要がある。しかし、ワイヤレスマウスを利用する場合でも、大画面から離れた場所に机のような水平である程度の広いスペースが必要になる。研究室やオフィスではそのようなスペースを用意することは容易である場合の方が多いが、人の往来などが激しい街頭ではそのようなスペースを用意することは困難な場合がほとんどである。

また、カーソルを移動させるためのマウスの運動量は画面のサイズが大きくなるにしたがって増加してしまう。例えば、図 2.2 のディスプレイ上で片方の端から反対側の端までカーソルを移動させるためには、マウスを 2・3 回往復させなければならない。そのため、カーソルの移動に時間がかかってきてしまう。プロパティなどでマウスの移動量を調整することは可能であるが、そこで移動量をあげた場合、正確さに欠けてしまう。しかも、1 回のマウスの移動でカーソルを端から端まで移動させることは不可能である。また、ディスプレイの近くにマウスを動かすための水平な台がない場合は、カーソルを動かすこと自体も困難になってくる。

光学式のマウスならば、人の足の上など水平な台がなくても操作は可能であるが、操作する場所が水平でないため操作しづらい。さらに、そのような場所は動かせる面積が狭いため、マウスを動かすための運動量が多くなってしまう。

つまり、マウスは従来のサイズのディスプレイ上では優れたインタフェースのひとつではあるが、大画面上で利用するには様々な問題が発生するため、使い勝手のよいインタフェースとは言い難い。

2.2.2 タッチパネルを用いたインタフェース

タッチパネルはプラズマディスプレイなどでよく利用されるインタフェースのひとつである。図 2.3 のように、画面に直接触れて操作を行う。そのため、非常に直感的で分かりやすい。また、指やペンを用いて操作するため、感覚的にフィードバックが得られる。さらに、ノートに書き込むような感覚で、大画面に直接書き込むことやカーソルを動かすことが可能である。

しかし、タッチパネルにもいくつか問題がある。タッチパネルはプラズマディスプレイなどのディスプレイでのみ利用可能であり、プロジェクタのスクリーンなどでは利用できないといったように、全ての大画面で利用できるものではない。また、画面が大きくなればなるほどタッチパネルの設置にコストがかかる。

さらに、タッチパネルで操作ができる範囲は、手が届く範囲のみであり、手が届かない範囲についてはタッチパネルのみでは操作が不可能である。つまり、壁サイズの大画面のような場合には、画面上部を操作することが非常に困難である。このように、タッチパネルを用いたインタフェースについても、画面のサイズによる制約が強いため、大画面向けのインタフェースであるとは言い難い。

2.2.3 リモコン

家庭のテレビやプロジェクタなどで最も多く利用されているのがリモコン（図 2.5）である。リモコンの最大の特徴は、離れた場所からでも操作できる点である。電波の届く範囲であれば、離れた位置からでも画面を操作することが可能である。リモコンの中には、ジョイスティックがついているものもあるため、画面内のカーソルを動かしたりすることなど簡単な操作を行うことも可能である。

しかし、リモコンの場合は離れた場所からでも操作できるということが設置場所によっては問題になってくる。例えば、オフィスや研究室・家庭などで遠くから操作できることは非常に便利であるが、駅や街角などの公共の場においては逆に問題が生じる。リモコンのような遠くから操作可能なデバイスは、画面の側面などからでも操作可能であるため、必ずしも画面の前にいる人が操作しているとは限らない。つまり、操作を行っているユーザが特定できないということである。そのため、あるユーザが画面を操作しようとしても、操作してもよいのかが判らないため、非常に利用しにくくなる。

さらに、駅や街角のディスプレイを操作するためにリモコンを設置した場合、壊されたり盗まれたりしてしまう可能性が非常に高い。だからといって、リモコンをワイヤーなどで外せないようにするのは、リモコンの特徴を生かされなくなるため本末転倒である。

2.3 新しいインタフェース

従来のデバイスを用いたインタフェースでは、大画面上での操作が難しいため、新しいインタフェースというもの盛んに研究されている。様々なインタフェースがあるが、従来のデバイスによるインタフェースの欠点を補うインタフェースと従来のインタフェースによらない全く新しいインタフェースを設計するという二種類に分けることが可能である。

従来のインタフェースの欠点を補うためのひとつの方法としては、手の届かないような領域を自分の作業しやすい場所に持ってくるという方法がある [5]。これらの特徴は、作業をしたい領域を作業しやすい場所に映し、そこで作業を行うと、そこでの作業が実際の場所に反映されることである。これによって、離れた場所でも容易に操作が可能になる。しかし、これらの問題点としては、遠くの場所を近くに表示するためにその領域を設定する必要があり、また、手元のスペースの一部を犠牲にしなければならないことである。

別の方法として新しいインタフェースを設計するといった研究も多くなされている。これは、従来のマウスやペンを用いるのではなく、レーザーポインタなど画面を指し示すために利用するもの自体をインタフェースにする方法がある。例えば、レーザーポインタを用いたインタフェース [11, 19] の場合、指し示した位置にカーソルを移動し、特定のジェスチャを行うとクリックなどの動作を行う。しかし、それらの問題点は、レーザポインタなどのデバイスを持っていなければ利用できないことである。そのため、利用可能な場所に制限をかけることになってしまう。

一方、デバイスを用いるのではなく、体の動きを利用して操作するインタフェースも存在する。例えば、視線を利用して操作を行うインタフェース [17] があり、これは視線の動きに

合わせてカーソルが移動し、注視したらクリックするといった感じである。目線を利用した場合、デバイスを用いる必要はないが、目の動きは非常に微妙で細かい動作が不可能でる。また、目は単純な動きしかできないため、動作の種類が少なすぎるといった問題がある。

そこで、大画面環境においてどのようなインタフェースが良いのだろうかと考えたとき、手によるジェスチャを用いればよいのではないかと考えた。手であれば、目線などとは違って様々な動作が可能であるため、より多くの機能をインタフェースにもたせることができる上に、デバイスを利用しないので場所の制限などを受けることがない。

そこで、本研究では、手を用いたインタフェースを設計する。次章では手を用いたインタフェースについて説明する。

第3章 手を用いた大画面向けインタフェース

第2章で挙げられた問題点を解決する手段の一つとして、大画面向けのインタフェースとして手を用いたインタフェースというものが考えられる。そこで、本章では手を用いたインタフェースについて説明していく。

3.1 手を用いたインタフェース

手を用いたインタフェースというものは、画面の大きさに関わらず様々な研究が行われてきている。手を用いたインタフェースに関する研究は大まかに2種類に分けられる。一つは、テーブルの上で手を動かすことにより画面を操作するインタフェース [15, 16] である。もうひとつは、画面に向かって手を動かすことによって操作するインタフェース [14] である。

テーブルの上で手を動かすことで操作するインタフェースはマウスに近い感覚で操作できるのが利点であるが、画面と手の動作を同時に確認しづらいため、画面と手元を交互に確認をしなければならないのが問題である。

一方、画面と向かい合って操作するインタフェースは、手の動作と画面を同時に確認することができるため、テーブルの上で手を動かす場合と比べて自分の現在の状況を把握しやすい。これは、画面のサイズが大きくなるほど状況把握は重要になってくるため、本研究では、画面と向かい合って手を動かすことで操作を行うようにした。

3.2 大画面向けのインタフェース

大画面向けのインタフェースとして、手を利用することは非常に有用である。なぜなら、手はユーザの体の中で最も動かしやすい箇所であるため、単純な動作から複雑な動作まで幅広い動きを行うことができる。そのため、目線を利用したものよりも、幅広いインタフェースが設計可能になり、大画面の利用シーンを広げることが可能になる。また、離れて操作させるようなインタフェースを設計することも容易なので、画面全体を把握しながら操作することができる。

大画面を手で操作しようという試みは、約四半世紀前に登場し [12]、現在でも盛んに行われている [3, 13]。しかし、それらの研究にはいくつか問題がある。ひとつは、それらの研究の多くはユーザの手に何かをデバイスを装着させる必要があるということである。ユーザに何かしらのデバイスを装着させることは、視覚的にコンピュータを使っているということを明示することになるため、D.A.Norman が提唱するような“ Invisible Computer ”とは程遠くなっ

てしまう。また、手に何も装着させず操作可能なインタフェース [4] も存在するが、それらの研究についても、テーブルの上で操作するようなものであったり、特殊なカメラを利用するものが多く、非常に技術的な面が前面に出てきてしまっている。

そこで、本研究では、手に何も装着させず、かつ、できる限りシステムも単純にした。本研究の簡単なイメージ図としては図 3.1 のとおりである。次章では、実際に設計したプロトタイプについて説明していく。

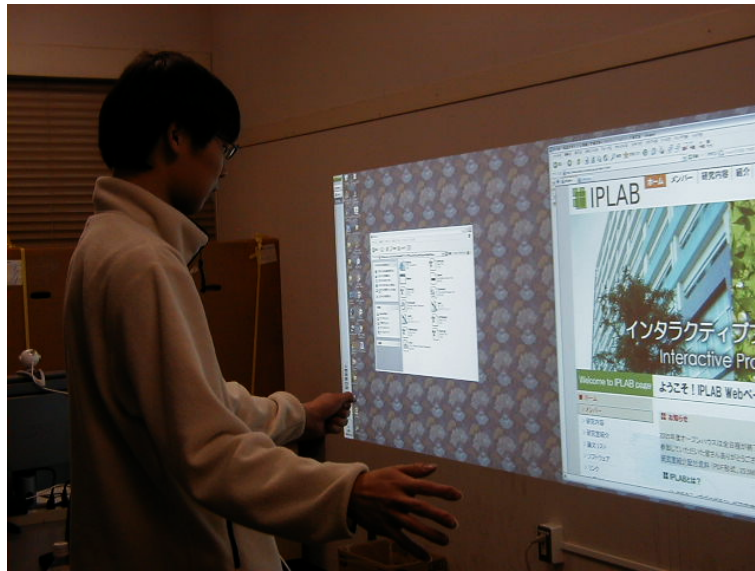


図 3.1: 利用イメージ

3.3 ゴール

手を用いた大画面向けインタフェースとしてのゴールは、両手によってポインタや文字入力などといった普段行っているような作業を全て行えるようにすることである。また、このとき手などになにもデバイスを装着せずに自然なジェスチャで操作が行えるようになることである。

しかし、ここでひとつ問題がある。それは、ユーザが連続して作業ができる時間にある。マウスを利用するような場合は、水平な台の上に手を置いて操作するため、長時間の作業が可能であるが、空中で手を動かす場合、台の上で作業するのと比べて疲れやすい。そのため、長時間の作業にはあまり向かない。

そこで、短時間で行う作業、例えば何かを調べるためにブラウジングが行いやすいようなインタフェースを設計していくが必要になる。

このようなゴールによって Mark Weiser や D.A.Norman が提唱したコンピュータ環境に近づけていくことができる。

第4章 提案

本章では、手を用いた大画面向けインタフェースの設計について、様々な観点からどのようなインタフェースが良いのかを提案する。また、どのようにしたら大画面環境において使いやすいインタフェースになるかということと同時に、Mark Weiser や D.A.Norman らが提唱しているコンピュータ環境に近づけていくためにどうしたらよいかということに重点を置いて提案していく。

4.1 デバイス

手を用いたインタフェースでは、手にセンサのついた手袋などのデバイスを装着することが多い。それには様々な理由があるが、デバイスを用いる最大の理由はより正確でより細かい動作を行いたいからである。現在の技術では人の手の細かい動きをデバイスなしで正確に把握することが困難である。また、指の位置などを3次元で把握することも難しい。なので、複雑な作業、例えば、細かい図形の描画などにはどうしてもデバイスの装着が必要になってしまう。

一方、デバイスを用いない場合、現在の技術では、デバイスを用いた場合と比べて、正確さに欠け、3次元の座標は調べにくい、細かい動作を把握しにくいといった問題点がある。しかし、大体の動きや動作を認識することならば可能である。そのため、簡単な作業、例えば、文章やwebの閲覧や大体の位置にオブジェクトを移動させることなどを行うには十分である。仮に、デバイスなしで3次元の位置座標や細かな動きを把握することが出来る技術があるのならば、デバイスなどが一切いらなくなる。そうなればユビキタスコンピュータの理想形に近づくことになる。

本来ならば、正確性などを上げるために手にデバイスなどを装着させるべきなのだろうが、本研究で手にデバイスを用いなかったのには理由がある。それは、本研究の目標の一つとして D.A.Norman の提唱した “Invisible Computer” にある。D.A.Norman の著書 [2] によると、“the technology of the computer fades away out of sight, and the new technology that replace it is as readily accepted and easy to use as a cassette tape recorder or CD player.” にある。つまり、コンピュータ技術は目に見えないところにあり、分かり易い技術に取って代わることが “Invisible Computer” の目的である。そのため、デバイスを用いてしまうとコンピュータ技術が前面に来てしまうことになる。そこで、本研究では、現在の技術で少しでも “Invisible Computer” に近づけるため、あえて手にデバイスを装着させなかった。

4.2 相対座標指定と絶対座標指定

インタフェースを設計する上で、カーソルを動かすことが必要になってくる。このカーソルの移動について、相対座標指定による移動と絶対座標指定による移動という2種類に分けることができる。相対座標指定による移動とは、マウスのように動かしたら、その動かした量にあわせてカーソルを移動させる。一方、絶対座標指定による移動とは、タッチパネルのように指した位置にカーソルを持っていくように、絶対的な座標を指定することによって、カーソルを動かす。

大抵のデバイスはこのどちらかでカーソルを動かしている。相対位置座標指定と絶対位置座標指定はどちらの方が優れているかは難しいところであるが、どちらの座標指定を利用するかについてはデバイスによって異なる。一般的には、画面を直接指し示す等してカーソル位置を指定するデバイスを利用する場合には絶対座標指定が利用され、そうでないデバイスの動かした大きさなどでカーソルの移動量を決める場合には相対位置座標指定が利用される。相対位置座標指定と絶対位置座標指定について分類したものが表 4.1 である。

表 4.1: 座標指定方法の分類

相対位置座標指定	絶対位置座標指定
マウス ジョイスティック タッチパッド	タッチパネル スタイラスペン

では、手を用いてカーソルを動かすとき、相対座標指定にするか、絶対座標指定にするかは議論の余地がある。ここで重要なのは、カーソルをどのようにして動かすかということである。例えば、直接画面を指し示すことでカーソルを動かす場合は絶対位置座標指定、手を動かした移動量に応じてカーソルを動かす場合は相対位置座標指定でカーソルを移動させる。また、どちらの位置座標指定を用いるかで注意しなければならない点が変わってくる。

絶対座標指定の場合、ユーザが指し示した位置にカーソルを持っていくために、いかに精度を良くするかが問題になる。この問題における最大の問題は計算をいくらソフトウェアの面を正確にしても、カメラなどのハードの面の精度を上げるのは非常に難しい。その上、指差してカーソルを持っていく場合、指し示した居場所を正確に指し示せるかと言った問題もある。そのため、意図した場所にカーソルをすぐに持っていくことは難しく、また細かい動きをさせることも難しくなってくる。そこで、本研究では、相対位置座標指定によってカーソルを動かすようにした。

ただし、相対位置座標による移動についても、さらに2つに分けることができる。ひとつは、マウスのように動かしている間だけ動くものである。もうひとつは、ジョイスティックのように傾けている間、その傾きにあわせて移動距離を変化させるものである。前者の方は直前の状態からどのぐらい動いたかを見るのに対し、後者は基準点からどのぐらい離れているかで動かし方を決める。

ここで重要なことは、目的の位置までカーソルを動かすのにどれだけデバイスを動かさな

ければいけないかということである。マウスのようなタイプの場合、先に述べたように、画面のサイズが大きくなるに従って、デバイスの移動量も増大してしまう。また、カーソルの移動量はプロパティなどで調整することはできるが、移動量を大きくすると、その分だけ動きが荒くなり、細かい動きがやりにくくなる。一方、ジョイスティックのようなタイプの場合、傾けている間だけ動き続けるため、デバイス自体の移動量はあまりない。また、傾き具合で移動量を調整することで小さな動きから大きな動きまで幅広く対応することができるため、細かな動きと荒い動きを混在させることができる。そこで、本研究では、カーソルの移動についてはジョイスティックのような動作をさせることにした。

4.3 ハンドジェスチャ

手を用いたインタフェースの最大の特徴として、手の形状(図 4.1)や手振りなど様々な動作を行うことができることである。さらに、手の形状と手振りを組み合わせることで様々なジェスチャが可能になる。それらジェスチャによって画面上でのカーソルの移動や決定などといった様々な動作をさせることができる。例えば、手を握ってそのまま話した場合はマウスのクリックの動作を、握った状態で動かした場合はドラッグの動作を行うようにすることができる。



図 4.1: 様々な手の形状

しかし、ここでひとつ問題がある。それは、ジェスチャの種類が非常に多いことである。手の形状一つとっても、図 4.1 のように単純な形状から複雑な形状まで様々な形状を取ることができる。さらに、手振りについては手の移動だけでなく手の傾き具合なども手振りとして考えられるため、それらを組み合わせるとジェスチャの数は膨大な量になる。そのため、動作を増やすためにジェスチャの数を増やすことは非常に簡単であるが、あまりに多いとユーザが覚えることが困難になるため使い勝手の悪いインタフェースになってしまうおそれがある。

そのため、ジェスチャを用いる場合には、ジェスチャの数を気にする必要がある。また、用いるジェスチャは単純なもので、かつ動作とジェスチャが結びつきやすい直感的なものにする

のが好ましい。

ここで、単純でかつ直感的なジェスチャとは何かについて考えてみる。まず、単純なジェスチャとは何かというと、手を動かさずに手の形状を変えるだけ、手を握った状態で動かすなどのように、簡単な形状+簡単な動きの組み合わせであると考えられる。また、直感的なジェスチャとは、ユーザの日常の行動と大画面上での動作が結びつきやすいということである。例えば、手を握った状態で移動させればマウスのドラッグと同じ動作を行うといった感じである。つまり、単純で直感的なジェスチャを設計するためには、簡単な手の形状と動きのジェスチャにユーザの行動と結びつきやすい動作を割り当てることである。

4.4 両手での動作

手を用いたインタフェースを設計する場合、両手を用いることでより様々な動作が可能になる。例えば、片手で行ってた動作を両手に割り振るだけでも片手のときと比べて倍の動作を割り当てることが可能である。

両手の動作の割り当て方については、様々な方法が考えられるが、左右の手が全く同じように動かせるわけではなく、また、同時に動かした場合、ユーザが混乱する可能性があるため、左右で全く同じ動作を行わせることは好ましくない。そのため、左右の手の動作の割り当てるとき、手の運動パターンについて知っておく必要があり、左右の手を動かす場合、次のような運動パターンがある [9]。

1. 左手は右手に対して、相対位置関係を指示する働きがある
2. 左手は大きな動きをするが、右手は細かい操作ができる
3. 左右の手を動かす順序は、左が先でその次が右である

ここに挙げた特徴については、右利きの人についていえることであるが、左利きの人であっても、左右が反対であるだけで基本的には全く同じである。左右の手の動作には、このような特徴があるため、左右の手で同じ動作をさせることは難しい。つまり、左右の手で2つのカーソルを同時に動かすという動作を行わせるのはあまりよくない。

先に挙げた特徴を元に、左右の手にそれぞれどのような動作を割り当てればよいかを考えた場合、いくつか割り当て方が考えられる。例えば、カーソルの移動や決定などよく行うような動作を右手でそのほかの補助的な動作を左手に割り当てる方法が考えられる。もしくは、右手はカーソルの移動のみ行い、そのほかの動作は全て左手で行うといった方法も考えられる。

このようなことを踏まえて、次章では、提案を元に設計したプロトタイプについて説明していく。

第5章 プロトタイプの設計と試作

第4章で提案された内容を受け、本章では手を用いたインタフェースのプロトタイプを設計および試作を行った。そのプロトタイプについて詳しく説明していく。

5.1 プロトタイプの設計

手を用いたインタフェースのプロトタイプとして、以下のような機能を持ったプロトタイプを設計した。

- ポインティングデバイス
- 文字入力

ポインティングデバイスとしての機能は主にマウスの動作を手当てはめることで実現する。文字入力については、キーボードをそのまま利用するわけにはいかないので工夫が必要になる。そこで、本プロトタイプでは、子音の入力により文字を入力する Popie[18] を利用した。このプロトタイプの特徴は以下のとおりである。

1. ユーザの手には何も装着させない
2. 両手による操作が可能

1 の特徴により、ユーザ側に制約をほとんどかけさせないことができる。また、2 の特徴を利用することによりユーザは幅広い操作を大画面上で行うことができる。

5.1.1 システムの構成

プロトタイプのシステム構成は主に入力部・処理部・出力部の3つからなる。それぞれの役割については以下のとおりである。

入力部

この部分では、手の形状や位置を入力として検出する。手の形状や位置を検出するために、計算機に接続されたカメラからキャプチャした画像を用いる。そのキャプチャした画像に処理を施して手のみを抽出し、そこから手の位置や形状を判別し、その情報を処理部へと受け渡す。

処理部

入力部から受けた情報をもとに、動作を決定する。プロトタイプでの動作について、手を握っているかどうか(手を握ったら grab で、手を開いた状態を open とする)と手の位置によって動作を決定する。手の位置については、右手の場合は基準点からどれぐらい動いているか、左手の場合はどの場所にあるか(上の方であれば upper、真ん中付近ならば center、下のほうであれば under)によって定める。また、手を握り続けている (grab の状態を維持している) 状態を hold とする。主な動作としては、カーソルの移動・決定、ドラッグ、Popie の起動である。動作については、表 5.1 のとおりである。

表 5.1: 動作

右手の形状	右手の移動	左手の形状	左手の位置	ドラッグ中	動作
open	○	-	-	-	カーソルの移動
open→grab	-	-	-	-	決定
-	○	open→grab	center	×	ドラッグの開始
-	○	hold	center	○	ドラッグ
-	-	hold→open	center	○	ドラッグの終了
-	-	open→grab	center 以外	×	スクロールの開始
-	-	hold	upper	×	スクロールアップ
-	-	hold	under	×	スクロールダウン
-	-	hold→open	-	×	スクロールの終了
-	×	open→hold	center	×	Popie を起動

なお、表 5.1 中の“-”はその状態によらないことを示し、○や×はその動作が行われた否かを示している。また、ドラッグ中とはドラッグの処理中であるか否かを示している。

出力部

処理部で決定した動作を出力として実行する。

5.2 プロトタイプの試作

5.2.1 開発環境

試作システムの実装には、VC++を利用し、カメラからの画像の取得や画像を処理するためのライブラリとしてインテル社が提供している OpenCV[21]を利用した。また、カメラはアートレイ社の USB カメラ (ARTCAM-130MI) を利用した

大まかなシステム図は図 5.1 のようになっている。図 5.1 の内、手の形状認識までが入力部、動作の決定が処理部、出力が出力部に当たる。

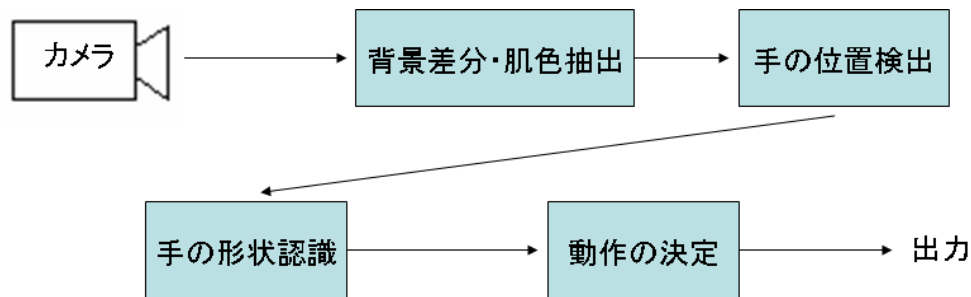


図 5.1: システム図

5.2.2 手の処理

システム全体の実装について説明する前に、まずは、最も重要である手の位置や形状の認識について説明する。

前処理

手の位置把握や手の形状認識を行う前に、手がどこにあるのかという候補を絞るためにカメラからキャプチャした画像を編集する必要がある。そこで、前処理として、背景差分と肌色抽出を行う。

まずは、予め撮影した背景画像(図 5.2)とカメラからキャプチャしてきた画像(図 5.3)との差分をとる。このとき、各画素について、一定の閾値以上の場合はキャプチャした画像の画素はそのままにし、閾値未満の場合には 0 とする。これにより、背景と人物とを分離させることが可能である(図 5.4)。さらに、その差分画像を YCC 表色系に変換する。YCC 表色系とは YCbCr と呼ばれ、Y は輝度、Cb は青の色差、Cr は赤の色差をあらわす。RGB から YCC 表色系への変換式は以下のとおりである。

$$Y = 0.29900 * R + 0.58700 * G + 0.11400 * B (0 \leq Y \leq 255)$$

$$Cb = -0.16874 * R - 0.33126 * G + 0.50000 * B + 128 (0 \leq Cb \leq 255)$$

$$Cr = 0.50000 * R - 0.41869 * G - 0.08131 * B + 128 (0 \leq Cr \leq 255)$$

YCC 表色系に変換する理由として、RGB の場合と比べて色差や輝度を比較しやすいため、より肌色を抽出しやすいためである [22]。肌色と言っても、白っぽい色から黒っぽい色まで様々であるため、RGB 値のみで肌色のフィルタを作成することは非常に困難である。

YCC 表色系に変換された画像を元に肌色抽出を行う。条件に当てはまる画素を肌色であると判断する。ここでは、以下の閾値に当てはまるものを肌色と判断する [23]。

$$Y < 200$$

$$Cr < 170$$

$$Cb + 0.6 * Cr < 200$$

各画素について上記の閾値に当てはまるものを肌色領域として二値化し、最後にノイズを取り除く。このような処理を全て施した最終的な結果は図 5.5 になる。



図 5.2: 背景



図 5.3: キャプチャ画像



図 5.4: 背景差分を取った画像

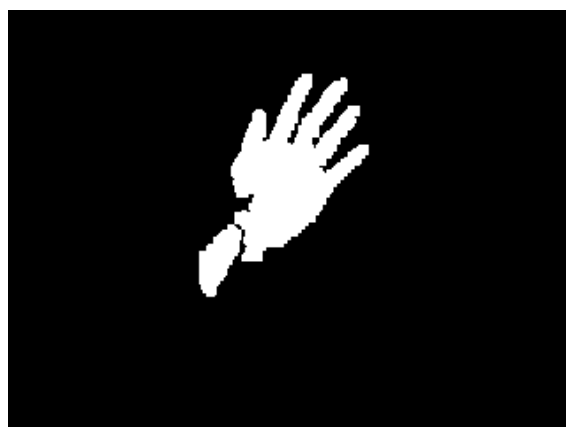


図 5.5: 肌色抽出の結果

手の位置把握

手の位置把握を行う場合、様々な方法が考えられるが、ここでは前処理で手と腕のみが抽出され顔などは映っていないと仮定する。その中で手がどの位置にあるかを探す場合、手のどの座標を検出するべきかが問題になる。手のみが完全に抽出されている場合は、どの座標を調べても問題はないが、図 5.6 のように肌色に似た服や半そでを着ていた場合は腕まで抽出される (図 5.7)。そのため、左下の座標で検出を行った場合は腕を検出してしまう。そこで、左下ではなく、右上の座標を検出することで、腕などの余計な箇所を検出せずに手だけを検出可能になる。



図 5.6: 腕まで抽出される例

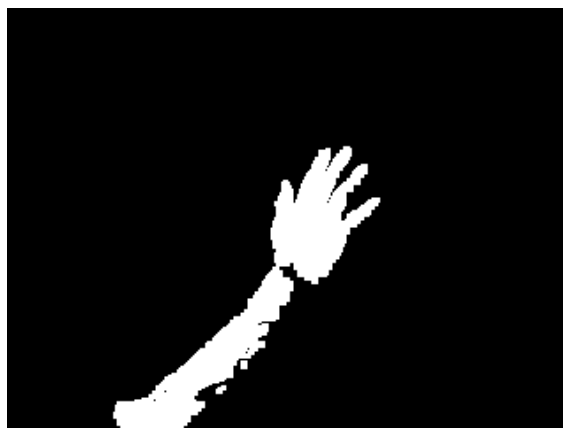


図 5.7: 結果

手の形状認識

手と認識した範囲の画像について、手を握っているか否かを調べる方法は色々あるが、本プロトタイプでは指の数から調べている [20]。指の数は本プロトタイプでは、画像の収縮と膨張処理を行うことで調べている。手順は以下のとおりである。

1. 手の画像を何回か収縮処理を行う
2. 1 の画像について、さらに膨張処理を何回か行う。

この方法は、指は手の平より細いことを利用している。手の画像 (図 5.8) に対して収縮処理を何回か行くと指の部分は消えるが、手のひらの部分は残る。その画像に対して膨張処理を施せば、手のひらの部分だけが復元される (図 5.9)。そして、二つの差分をとることで、指の面積を調べることができる。



図 5.8: 原画像



図 5.9: 図 5.8 に対して収縮・膨張処理を施した画像

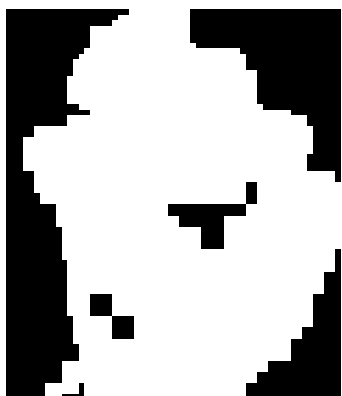


図 5.10: 手を握っている画像



図 5.11: 図 5.10 に対して収縮・膨張処理を施した画像

手が開いている状態では、図 5.8 と図 5.9 から分かるように二つの画像の違いがはっきりしているため、手を開いていると判断することができる。一方、図 5.10 と図 5.11 のように手を握っている状態で同様の処理を行った場合はほとんど違いが見られないため、握っていると判断することができる。

5.2.3 両手の同時処理

5.2.2 で説明した方法を本プロトタイプでは両手に利用している。ただし、キャプチャした画像の全域について操作を行うわけではない。本プロトタイプでは、図 5.12 のように右手領域 (図 5.12 中の赤の矩形領域内) と左手の領域 (図 5.12 中の白の矩形領域内) に分け、それぞれの領域について、手の位置や形状の認識を行う。ただし、左手の手の位置を検出するときは右上の座標を検出するのではなく、左上の座標を検出する必要がある。



図 5.12: 走査領域の分割

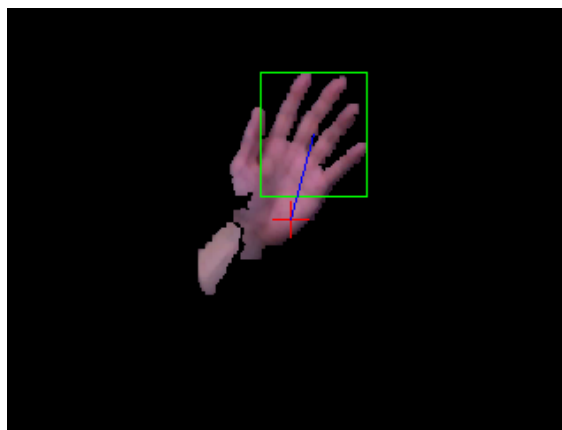


図 5.13: 基準点と手の領域

5.2.4 動作

ここでは、本プロトタイプで設計した動作の実装方法について、ひとつずつ説明していく。

カーソルの移動

カーソルの移動は、移動方向と移動量によって決める。移動方向と移動量の決定方法は手が基準点から見てどの位置にあり、どのぐらい離れているかによって決める。図 5.13 の赤い十字の中心が基準点であり、緑色の矩形で囲まれているのが手の領域である。この基準点から手の領域の中心までどの方向にどのぐらい離れているかを x 方向と y 方向それぞれについて調べ、そのときの位置関係から移動方向を、離れている距離に応じて移動量を決定する。移動量については、あまり基準点から離れていない場合には移動量を小さくし、離れるに従って移動量を多くしている。

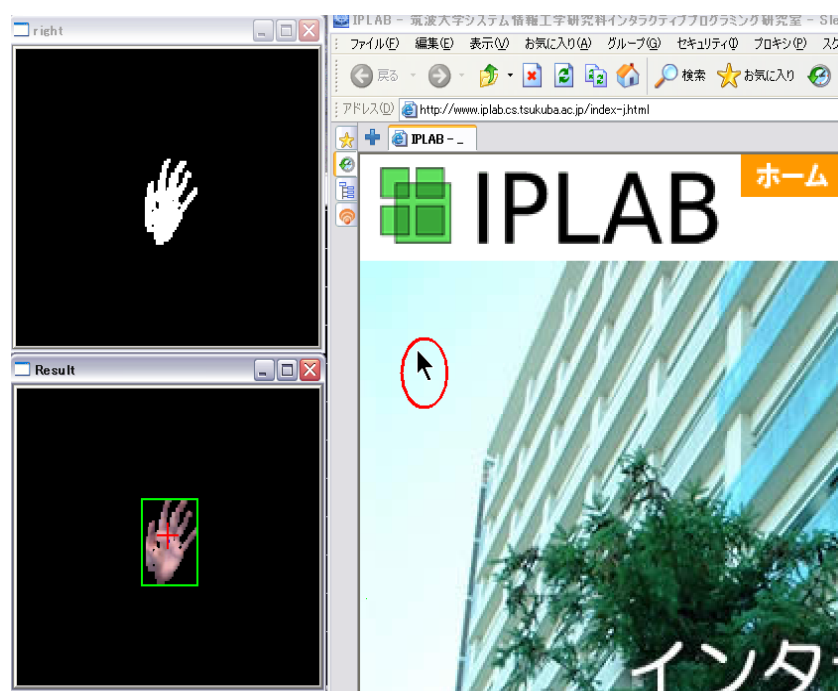


図 5.14: 移動前

図 5.14 と図 5.15 は実際の移動例である。図 5.14 のように、手の位置（図 5.14 の左側のウィンドウ中の緑の矩形領域内）が中心付近にある場合はカーソルは停止（図 5.14 の赤い円内）し、図 5.15 のように手の位置（図 5.15 の左側のウィンドウ中の緑の矩形領域内）が中心から離れている場合はその離れた距離に応じてカーソルが移動し続ける（図 5.15 中の赤い円内）。このとき、中心にある基準点（図 5.14, 5.15 中の赤い十字）から離れているほど移動量が大きくなる。

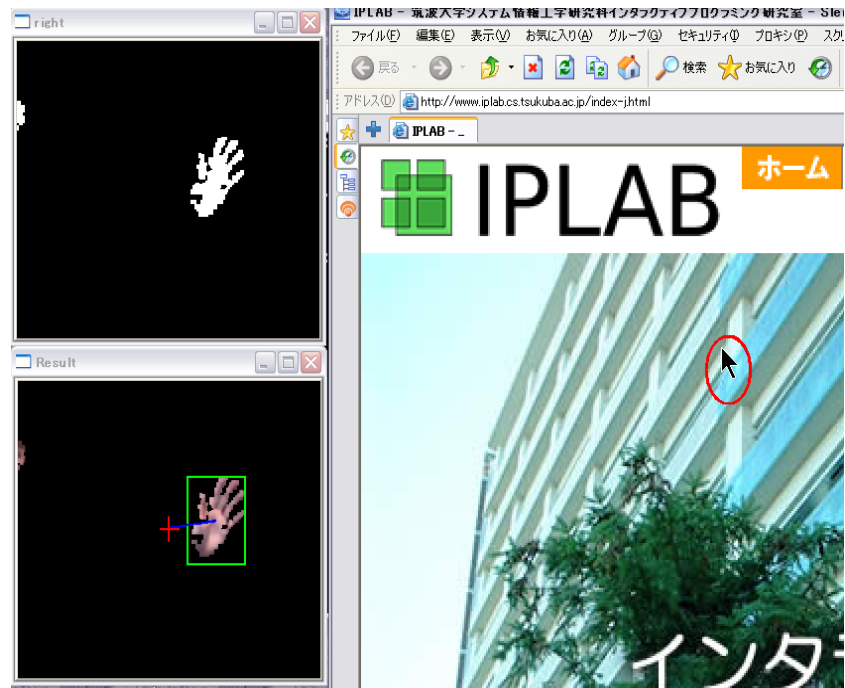


図 5.15: 右にカーソルを動かしている状態

決定

右手が握れられた場合、マウスの左クリックに当たる動作を行う。このときは、カーソルの移動が行われないようにする。例えば、のように手を握る（図 5.16 の緑の矩形領域内）とマウスの左クリックの動作が行われ、図 5.17 のようにクリックしたページが表示される。決定の動作が行われるタイミングについては、手を閉じた直後に起きようになっている。手を閉じた状態で動かした場合は、カーソルの移動が行われるのみである。

ドラッグ

左手が基準点の中央付近で握られて、かつその直後にカーソルの移動が行われる場合は、ドラッグの動作を行う。ドラッグは左手が握られている状態の間だけ行い、左手が開いた状態になったらドラッグを終了する。このとき、左手が握られている状態で、しばらくの間、右手の移動がない場合は文字入力を行う。ドラッグを終了するためには、左手を開いた状態にすればよい。

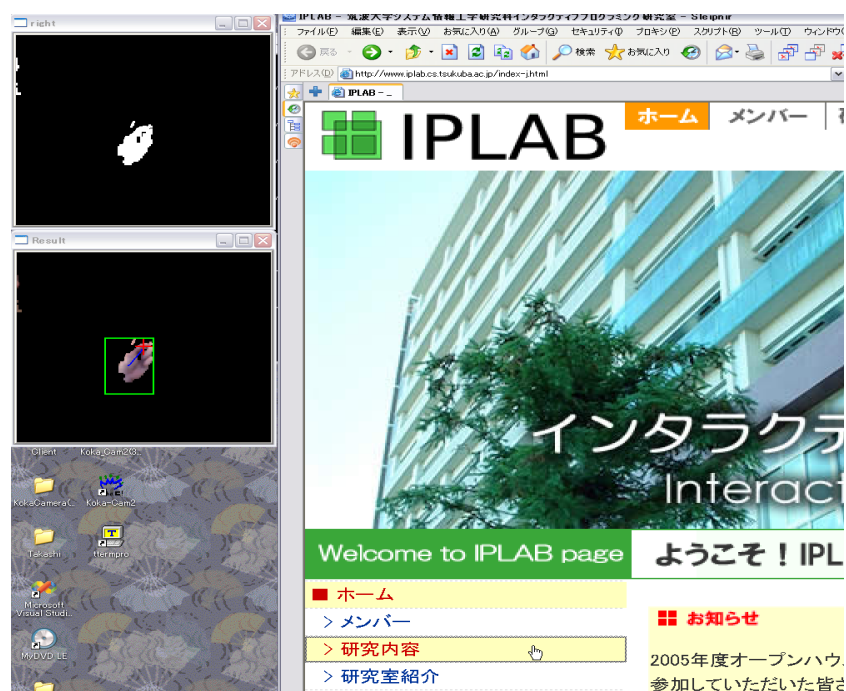


図 5.16: 右手を握った状態



図 5.17: 決定（マウスの左クリック）後

スクロール

左手が基準点の中央付近以外で握られた場合は、画面の上下のスクロールを行う。スクロールの方向や量はカーソルの移動時と同様に、基準点から見てどの位置でどのくらい離れているかより決定する。

例えば、手を開いてる状態（図 5.18 中の緑の矩形領域内）の場合は画面はスクロールしないが、手を基準点（図 5.19 中の赤の十字）より下のほうで握っている場合（図 5.19 中の緑の矩形領域内）、画面が下にスクロールしていく。また、このときもカーソルの移動のときと同様に基準点から離れるほどスクロール量は大きくなる。また、下にスクロールさせている途中で上にスクロールさせることも可能である。

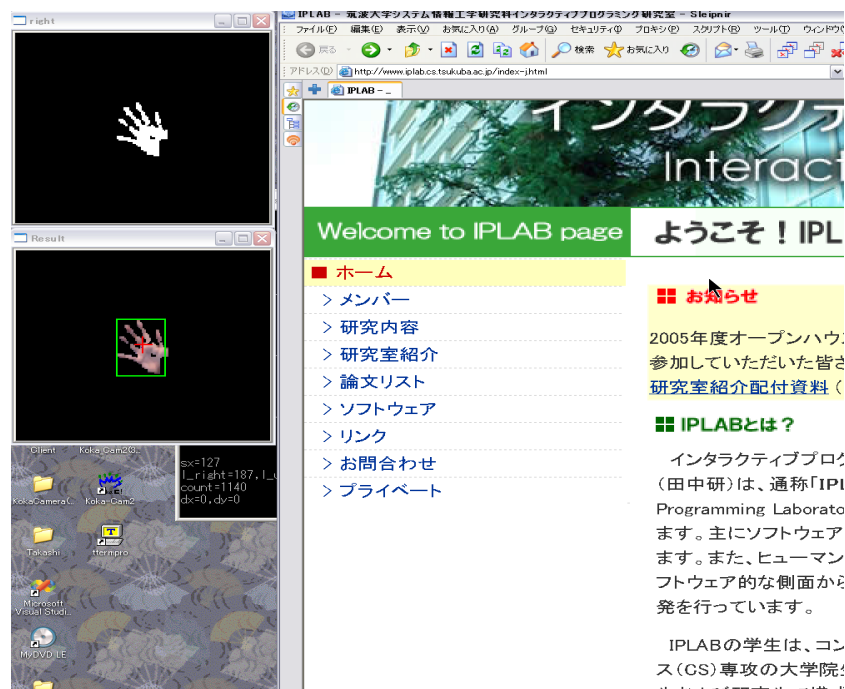


図 5.18: スクロール前

文字入力

左手が基準点の中央で一定時間握られていて、かつカーソルが動かない場合は、文字入力を行う。文字入力としては、Popie をそのまま利用する形であるが、このときのカーソルの移動に関しては、Popie にあわせて移動量などを調節される。

ここで、Popie について簡単に説明する。Popie はペン入力のための文字入力インタフェースである（図 5.20）。主な特徴としては、Popie は FlowMenu を利用したインタフェースであり、また、子音を入力することで、その入力に対して予測される候補を選択することで日本語入

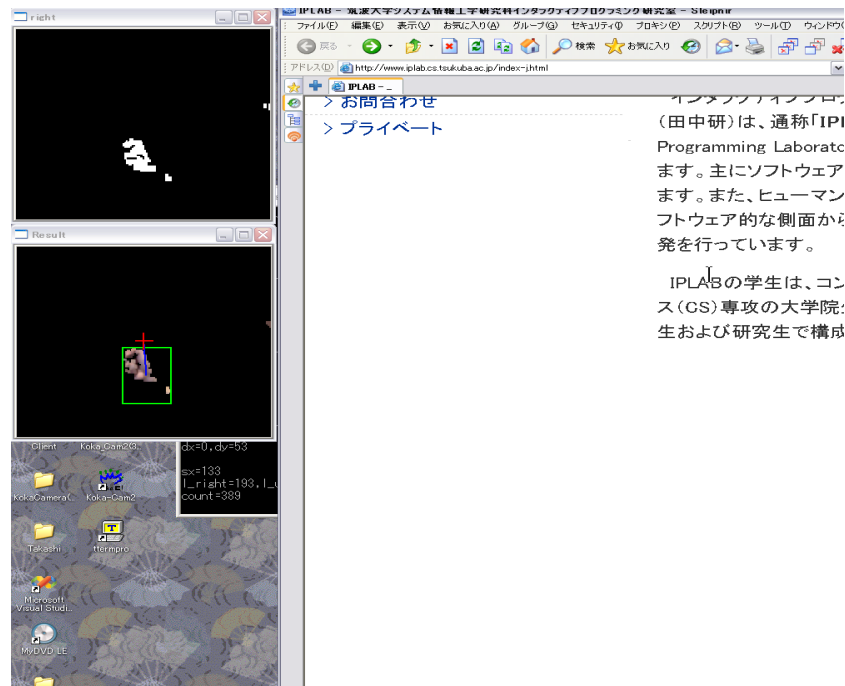


図 5.19: スクロール中

力を行うことが出来る。例えば、“田中”と入力したい場合、“TNK”と子音を入力して変換候補から“田中”を選択することで入力する。

Popie はももとはペンインタフェース向けの文字入力システムであるが、本プロトタイプでは、基本的にはそのまま利用する。ただし、そのときのカーソルの移動については Popie を利用しやすいように調整している。

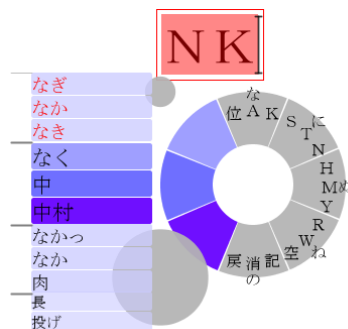


図 5.20: Popie のインタフェース

第6章 評価と今後の課題

第5章で設計したプロトタイプについて、本章では、本プロトタイプを試用し、評価した。また実装した機能のほかにどのような機能が必要になってくるかについて議論する。また、本プロトタイプを利用例についても考える。

6.1 試用による評価

実際に設計したプロトタイプを利用して、web ページの閲覧やペイント、文字入力などコンピュータ上でよく行う作業を試してみた。その結果判明した問題点などをいくつかまとめた。

6.1.1 移動

手を利用した場合、やはりマウスなどと比べて細かい動作は難しい。例えば、web ページのリンクをクリックするような場合、そのターゲットがある程度の大きさならば容易にカーソルを合わせてクリックすることができる。一方、ターゲットが小さい場合、カーソルを少しだけずらすということが困難であるため、カーソルを合わせるのが困難だった。

このような問題が生じた原因としては、移動量の設定にある。本プロトタイプでは移動量は基準点との距離に比例している。そのため、増加の割合は常に一定であるため移動量の微調整が非常に難しい。このような問題を解決するためには、手が基準点から近ければ移動量をより小さくし、離れるに従って移動量を指数関数的に増加させるなど、手が近くにある場合と離れている場合で増加の割合を変えてあげればよい。

しかし、離れた場所にカーソルを持っていくのは非常に楽であった。マウスとは異なり、基準点から離れている間だけ動き続けるため、一度手を動かせば十分なため、手の移動量は非常に少なく済んだ。

6.1.2 文字入力

文字入力については Popie をそのまま利用した形であるが、ひとつ問題がある。それは Popie はもともとペン入力のためのインタフェースであり、その Popie の動作にプロトタイプのカーソルの移動方法をあわせたことである。本来ならば、手の動きに合わせて Popie の動作を調整するのが好ましい。なぜなら、Popie の動作にあわせるので、Popie を利用していると

きだけ操作方法が変わってしまうためユーザの混乱を招きかねない。つまり、プロトタイプ
の動作にあわせて Popie の動作を調整する方が好ましい。例えば、右手で子音の入力を行い、
左手で変換候補の選択を行うように Popie を調整する。

6.1.3 動作の確認

設計したプロトタイプで最も問題なのは、ユーザが今どの動作を行っているかの確認をする
のが難しいことである。例えば、カーソルの移動についても移動方向や量を決めるための
基準点は図 5.12 のような認識結果の画面を見ながらではないと難しい。そのため、その画面
が他のウィンドウで隠れてしまう場合、意図した方向にカーソルを移動させること自体が大
変になってしまう。また、手を握ってるつもりでもコンピュータ側が認識していないという
ことが確認しづらいといった問題もある。

そのような問題点を解決する方法としては、認識結果の画面が常にユーザの目に見える位
置に表示されるように画面が自動的に移動し、またその画面内に現在どの動作だと認識して
いるかを表示させることが考えられる。しかし、このとき注意しなければならない点は、そ
の画面の表示位置をユーザの作業の邪魔にならない位置に表示しなければならない。

6.2 今後の課題

試用で得られた問題点以外に、手を用いた大画面向けインタフェースを設計するに当たっ
て必要なことについて本節では議論していく。

6.2.1 フィードバック

手を用いたインタフェースにおいて、フィードバックの与え方は議論の余地がある。手
を用いたインタフェースにおいてほしいフィードバックは現在カーソルがどこにあるかと手が
どの位置にあってどういう動作を行ったかをユーザ側に示すことである。

ある程度はなれた場所から手で操作する場合、レーザーポインタのように画面を直接指し
示すことができないため、フィードバックをしっかりと与えなければカーソルなどを見失い、操
作に支障をきたしかねない。そのような自体を回避するための方法はいくつか考えられる。最
も単純な方法としては、カーソルを目立たせることが重要になってくる。カーソルを目立たせ
る最も簡単な手段としてはカーソルを大きくすることである。しかし、これだけではカーソル
を見失ったときに見つけるのに時間がかかってしまう。この問題に対して、例えば、Windows
ではマウスの設定を変更することで Ctrl キーを押すことでカーソルの位置を表示させるこ
とができる。

また、ジェスチャなどの動作を行った場合についても、意図した動作がなされているかど
うかということをフィードバックとして与えることで、ユーザは意図した動作であるか否か
を確認することができる。多くの GUI では、動作によってカーソルの形が変化する (図 6.1)。

例えばマウスの場合、普段は矢印であるが、移動を行う場合には十字の矢印に変形したり、拡大縮小を行う際には斜めの矢印に変化する。このように、カーソルの変化はユーザにとっては非常に分かりやすいフィードバックのひとつである。

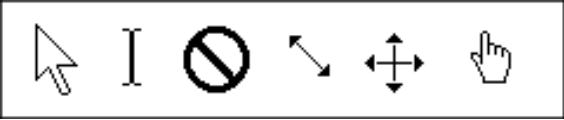


図 6.1: 様々なカーソル

6.2.2 メニュー

大画面上で操作する上で問題なのが、メニューにある。従来のメニューというと図 6.2 や図 6.3 のようなマウスの右クリックによるポップアップメニューやプログラムの上部にあるメニューバーがある。しかし、それらは大画面でそのまま利用するのは難しい。ポップアップメニューやメニューバーなどの問題点は、一つ一つのメニューのサイズが小さいため、離れた位置からでは非常に見づらく、また選択しづらい。特に、ポップアップメニューのようなメニューの場合、ひとつあたりの項目の領域が狭く、また隣合っているため、誤選択の可能性が高くなる。さらに、これらのメニューの最大の問題は、階層が深くなるにつれ広がってしまう (図 6.4) ため、選択に時間がかかる上に誤選択の可能性がさらに高くなってしまう。

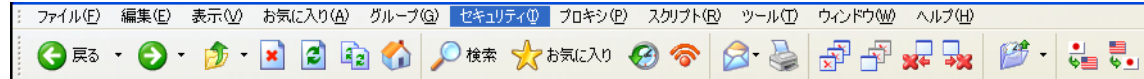


図 6.2: メニューバー

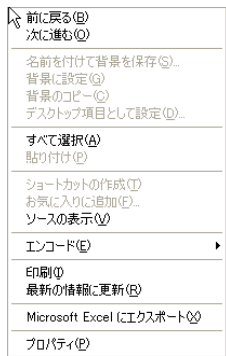


図 6.3: ポップアップメニュー

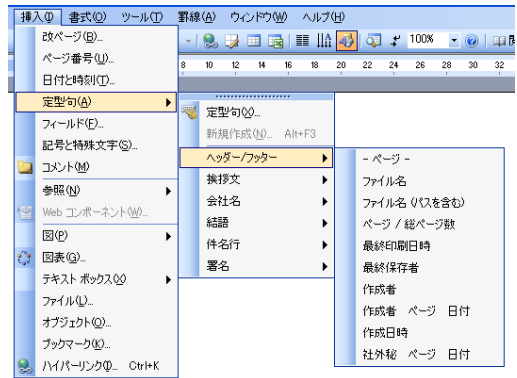


図 6.4: 階層が深くなったメニュー

そこで、従来のようなメニューを利用するのではなく、FlowMenu[8] 等といったメニューを利用したり、CrossY[7] などのように横切ることでメニューを選択する手法を利用したりする方が使い勝手がよくなる。例えば、本プロトタイプで文字入力として利用した Popie は FlowMenu を利用している。

また、既存の研究を利用するのではなく、大画面向けに使いやすいメニューを考えるのもひとつの方法である。例えば、カーソルの近くにいくつかメニューが浮かぶように配置されていて、ユーザがそのメニューにカーソルを近づけると、メニューの方から寄ってきて、ユーザはそれを選択するだけでよいといった方法がある。ここで重要なことは、ユーザから見やすく、かつ選択しやすいメニューを設計することである。例えば、ブラウザを開いている場合、一定時間カーソルをとどめておくと図 6.5 のようによく利用するメニューが出てくる。そこで、戻るのボタンに近づくとそのボタンがカーソルに吸い付くようにする (図 6.6)。



図 6.5: 浮かんでいるメニュー



図 6.6: 戻るボタンに近づいた場合

6.2.3 アプリケーションへの利用

本プロトタイプを利用したアプリケーション例について考えてみた。まず、このとき考えなければならないことは、手を用いたインタフェースは長時間の作業にはあまり向かず、またマウスなどと比べて精密な動きをすることが難しいということである。つまり、文章作成やペイント等といった作業には向かないと考えられる。

それでは、短時間で行う作業について考えてみると、プレゼンテーションや調べ物のために簡単なブラウジングを行う等ということがある。または、本プロトタイプの特徴をいかして何か大画面上で伝言を残すようなアプリケーションを設計するなど、様々な利用例があると考えられる。

第7章 関連研究

Daniel Vogel らは手にデバイスを装着することで、大画面を指差すことでカーソルの移動を行い、また、指を動かすことでクリックの動作を行うインタフェースを開発している [3]。Daniel Vogel らの研究の目的は、大画面を離れた場所から手を用いて直接操作し、また大画面を3次元空間上で走査することである。Daniel Vogel らの研究ではカーソルを動かし、クリックを行うだけだが、本研究は文字入力が可能などより幅広い動作が可能になっている。

また、木村らは映画『マイノリティ・リポート』に登場する広視野ディスプレイとジェスチャ操作の組み合わせによるインタフェース [13] を設計している。この研究では、ジェスチャによる広視野のワイド画面で映像再生・編集システムを試作している。しかし、システム自体も複雑であるため、利用可能な場所も制限されてしまう。また、ジェスチャの種類も多いため、ユーザが利用するにあたって混乱を招く可能性がある。

一方、Miyaoku らは、手ではなく携帯電話を用いて大画面を操作するインタフェース [6] を設計した。これは、携帯電話から色相差を利用した光信号を送ることで、カーソルを動かしたり画像を貼り付けたりすることができる。携帯電話というデバイスを利用しているためデータのやり取りが可能になってくる。しかし、本研究と違い、自分の体以外のものを利用するため、コンピュータを利用しているという意識が非常に強くなり、また、携帯電話が手元になければ大画面を操作をすることができない。さらに、Iwabuchi らの Natural Storage [10] の理論を利用することで、データを体に保存することができるため、携帯電話がなくても画像などのデータを大画面上に貼り付けることができるため、そのようなデバイスを持ちいらなくても設計することが可能である。

第8章 結論

本稿では、まず、両手を用いた大画面向け操作インタフェースを提案した。次に、その提案を元にカーソルの移動や文字入力を行うことが出来るプロトタイプの実装について述べた。最後に、設計したプロトタイプの問題点や手を用いたインタフェースを設計する際に重要になることについて議論し、プロトタイプの利用例についても述べた。

今後の展望としては、先に挙げたプロトタイプの問題点を解決していく。また、フィードバックを与えたりメニューを工夫したりすることで、ユーザにとって使いやすいインタフェースというものを追求し、ユーザが中心となるようなインタフェースへと改良していく。そして、Mark Weiser や D. A. Norman らが提唱したコンピューティング環境に近づけていく。

謝辞

本研究を進めるにあたって、指導教官である田中二郎教授、チームリーダーである高橋伸講師、ならびに三末和男助教授、志築文太郎講師から様々な助言、ご指導をいただきました。心より感謝いたします。

また、田中研究室の皆さんにも大変お世話になりました。とくに、同じユビキタス・大画面チームの皆さんには大変お世話になりました。皆さん、本当にありがとうございました。

参考文献

- [1] Mark Weiser. The computer for the twenty-first century. In *Scientific American*, pp. 94–104, September 1991.
- [2] Donald A. Norman. THE INVISIBLE COMPUTER. THE MIT PRESS. Cambridge, Massachusetts, London, England.
- [3] Daniel Vogel and Ravin Balakrishnan. Distant Freehand Pointing and Clicking on Very Large. In *UIST'05*, pp. 33–42, 2005.
- [4] Shahzad Malik, Abhishek Ranjan, and Ravin Balakrishnan. Interacting with Large Displays from a Distance with Vision-Tracked Multi-Finger Gestural Input. In *UIST'05*, pp. 43–52
- [5] Azam Khan, George Fitzmaurice, Don Almeida, Nicolas Burtnyk, and Gordon Kurtenbach. A Remote Control Interface for Large Displays. In *UIST'04*, pp. 127–136
- [6] Kento Miyaoku, Suguru Higashino, and Yoshinobu Tonomura. C-Blink: A Hue-Difference-Based Light Signal Marker for Large Screen Interaction via Any Mobile Terminal. In *UIST'04*, pp. 147–156.
- [7] Georg Apitz and Francois Guimbretiere. CrossY: A Crossing-Based Drawing Application. In *UIST'04*, pp. 3–12
- [8] Francois Guimbretiere and Terry Winograd. FlowMenu: Combining command, text, and data entry. In *UIST2000*, pp. 213–216.
- [9] Ken Hinckley, Randy Pausch, Dennis Proffitt, and Neal F. Kassell. Two-Handed Virtual Manipulation. In *ACM Transactions on Computer-Human Interaction*, Vol. 5, No. 3, September 1998, pp. 260–302.
- [10] S. Iwabuchi, B. Shizuki, K. Misue, and J. Tanaka. Natural Storage in Human Body. *KES2005*, Part IV, pp. 430–436.
- [11] Xing Chen and James Davis. LumiPoint: Multi-user laser-based interaction on large tiled displays. *Displays*, Vol. 23, pp. 205–211.
- [12] Richard A. Bolt. “Put-that-there”: Voice and gesture at the graphics interface. *SIGGRAPH '80*, 1980, pp. 262–270.

- [13] 木村 朝子, 鶴田 剛史, 酒井 理生, 鬼柳 牧子, 柴田 史久, 田村 秀行. 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・レポート型 I/F とその発展形. インタラクション 2005, pp.143–150.
- [14] 蔵田武志, 興梠正克, 加藤丈和, 大隈隆史, 坂上勝彦. 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.47–52.
- [15] 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.
- [16] 陳 欣蕾, 岡 兼司, 中西 泰人, 佐藤 洋一, 小池 英樹 机型インタフェースにおける両手直接操作による描画システム. WISS2001, pp.179–184.
- [17] 長澤 篤, 中西 泰人, 小池 英樹, 佐藤 洋一. EnhancedWall: 大型ディスプレイにおける顔追跡システムの利用とそのアプリケーション. WISS2002, pp.105–110.
- [18] 佐藤大介, 志築文太郎, 三浦元喜, 田中二郎. ペンによるメニュー選択に基づく子音を用いた日本語入力手法. インタラクション 2004 論文集, pp.205–206.
- [19] 小山慎哉, 小野寺光, 葛岡英明, 山崎敬一. 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.
- [20] 入江 耕太, 梅田 和昇, 若村 直弘. ジェスチャ認識を利用したマン・マシン・インタフェースの構築—インテリジェントルームへの適用—. 日本機械学会 [No.03-4] ロボティクス・メカトロニクス講演会 '03 講演論文集, 1A1-3F-D1(1)–1A1-3F-D1(2).
- [21] Intel. <http://www.intel.com/technology/computing/opencv/index.htm>
- [22] 武岡さおり, 尾崎正弘, 足達義則. 個人認証のための顔画像抽出と顔方向の自動認識. 名古屋女子大学紀要 50 (人・社), 2004, pp.145–151.
- [23] OpenCV 情報 (for Win). <http://ace.seman.cs.uec.ac.jp/takahashi/OpenCV.htm>.