平成19年度

筑波大学第三学群情報学類

卒業研究論文

題目 円筒型マルチタッチインタフェース

主専攻　　情報科学主専攻

著者　　内藤 真樹

指導教員　田中 二郎　志築 文太郎　三末 和男　高橋 伸

要　旨

複数の指や手のひらを使って同時に触ることにより操作するマルチタッチインタフェースの研究・開発が進められている。その発展として本研究では、円筒側面を操作面に利用した円筒型マルチタッチインタフェースを提案する。このインタフェースでは操作空間が円柱座標系となるため、操作面は 2 次元ではあるが、その曲面の奥行きを使用することで 3 次元に対する操作が可能である。さらに、マルチタッチであることを利用し、両手を使ってオブジェクトの操作やカメラの操作を自然に行うことが出来る。実際に 3D オブジェクトの閲覧を行う 3D ビューアなどを試作し、円筒型マルチタッチインタフェースの利用例を述べる。

目 次

図 目 次

第1章　はじめに

マルチタッチインタフェースとは、従来のマウスやスタイラスによるシングルポインティングデバイスとは異なり、複数入力を利用し、操作を行うインタフェースである。特に指や、手のひらの接触点を利用することにより、シングルポインティングデバイスを介した操作に比べてより自然で直感的なジェスチャ操作などを可能とする。

例えば、従来のシングルポインティングデバイスを利用した画像編集の拡大や回転といった操作は、画像の端にあるそれらの操作に対応したアイコンなどをドラッグ & ドロップすることに対応付けられている。これに対し、マルチタッチインタフェースを利用した操作では画像の 2 点を押さえるように触り、“その間隔を広げたり狭くしたりする” や “1 点を中心にもう片方の指を回転させる” といったジェスチャに対応付けることが出来る。

近年では、さらに、複数人からの同時入力が可能なことからテーブルトップ型のマルチタッチインタフェースとして、会議や学習環境への応用も検討され、協調作業や、学習支援として研究されている [SRF+06][北原 06][藤村 06][WSR+06][石井 03]。また他にも、小型携帯デバイスへの応用も研究されており、これらはアップル社の iPod touch で実用化がなされている。

しかし、既存のマルチタッチインタフェースは操作面として平面を使用しており、その操作対象も平面であることがほとんどである。近年では、3 次元データを扱うアプリケーションも多く、このようなアプリケーションの従来のシングルポインティングデバイスを用いた操作ではユーザは 2 次元状の操作空間から 3 次元空間へのマッピングを行う必要がある。そこで、3 次元的な操作を可能とするマルチタッチインタフェースとして円筒曲面の操作面を備える円筒型マルチタッチインタフェースを提案する。

円筒型マルチタッチインタフェースでは、既存研究で挙げられているマルチタッチインタフェース特有の操作に加えて、インタフェース本体に対して “抱きつく” や両手で “はさむ” といった、従来の平面型のマルチタッチインタフェースとは異なる操作が可能になる。また、円筒の側面から奥部にかけての操作面の奥行きを利用することにより 3 次元物体の操作を行うことができる。本研究では実際に円筒形マルチタッチインタフェースを作成し、各種アプリケーションに対してどのような操作法が適用できるかを提案し検討した。

本論文の構成

本論文の構成は以下のとおりである。第 2 章では本研究で作成した円筒型マルチタッチインタフェースの概要を述べ、第 3,4 章では円筒型マルチタッチインタフェースの実装および、円筒型マルチタッチインタフェースを利用したアプリケーションについて述べる。第 5 章で

は円筒型マルチタッチインタフェースの考察を述べ、第 6 章で本研究と既存研究の関連を述べる。第 7 章で今後の展望を述べ、第 8 章で本論文をまとめる。

第2章　円筒型マルチタッチインタフェース

2.1　円筒型マルチタッチインタフェースの特徴

図 2.1 にこの円筒型マルチタッチインタフェースの概略図を示す。

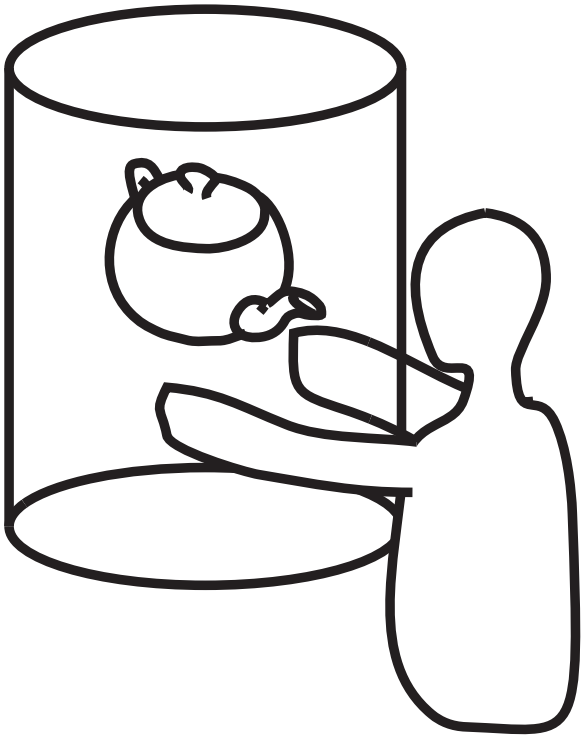


図 2.1: 円筒型マルチタッチインタフェースの概略

この円筒型マルチタッチインタフェースには、従来のシングルポインティングデバイスによる GUI や、平面型のマルチタッチインタフェースと異なり、以下の特徴がある。

- 作業領域の連続性

  円筒型であることから左右への作業領域に途切れがないことが挙げられる。例えばユーザがデバイスの周りを 360 度自由に動くことにより、没入方の VR 環境とは逆の、全方位から対象への操作が可能になる。

- 奥行きを利用した 3 次元的な操作性

  入力に使用する操作面は曲面という 2 次元であるが、操作面自体が曲がっているためユーザの位置に対して奥行きを利用する 3 次元的な操作感を与えることができる。また、このインタフェース自体が奥行きを持つことにより、従来の平面型マルチタッチインタフェースにおいては “触る” と表現される動作に加えて “抱きつく”、“はさむ” といった操作も可能になる。

2.2 円筒型マルチタッチインタフェースの利用例

円筒型マルチタッチインタフェースの利用例として以下のような使用を考えている。

- 公共施設での情報表示デバイス

  全方向からの操作が可能であることを利用し、また情報表示を的確に行えば、公共施設での地図案内システムや、実物大表示を行う仮想展示システムとして利用できる。

- 3 次元操作性を活かした 3D モデリングツール

  奥行き方向の操作を有効に活用することで 3 次元的な操作が可能になる。また、情報表示と操作を一体として行うことで従来の操作と表示が分離していた 3D 操作デバイスよりも直感的に操作を行うことができる。

第3章　円筒型マルチタッチインタフェースの実装

3.1　タッチパネルのデザイン

ユーザに自然な操作を提供するために以下の設計方針を定めた。

- ユーザが特殊な装置を身に着けずに操作が行えること。

  公共施設などの設置を考える場合には、操作に必要な装置をユーザが必ず持っているとは限らない。また、日常的に利用する場合においても操作を行うたびに操作装置を装着することはユーザへの負担となる。

- 従来のマウスによる操作を負担なく行えること。

  作業用にこのインタフェースを利用する場合には、このインタフェースのために作られた専用のアプリケーションのみが利用できるのではなく、従来のアプリケーション資産を使用できる必要がある。

- ユーザに操作の結果を示すための情報表示を行うこと。

  既存の 3D デバイスなどでは操作を行うデバイスとその作業を確認する表示ディスプレイが分離している。しかし、このようなデバイスではユーザは実際の操作量とコンピュータ上での操作量を対応付けて把握することを強制される。そのようなユーザへの負担は、操作部と情報表示とを同時に視界に入れることで軽減されると考えられる。

これらの方針を満たすために、我々は Han の方式 [Han05] に基づきマルチタッチインタフェースを作成した。

Han の方式ではタッチパネルの作成にアクリル素材および赤外線を用いる。アクリルに赤外線を照射することにより、アクリルの内部に入った赤外線がアクリルと空気の屈折率の関係によりアクリルの内部で全反射を起こす (図 3.1①)。しかし、そのアクリルの表面をユーザが触れると、接触点において全反射が止まり拡散反射が起こる (図 3.1②)。この拡散反射した赤外線を撮影することによって接触点を認識することができる。これによってユーザは何も特殊な装置を身に着けることなく素手による操作が可能となる。我々はこのアクリル素材としてアクリルパイプを用いた。

また、両手による操作として “抱きつく” ようにインタフェースを覆う操作を可能にするためにアクリルパイプは直径 600mm、高さ 1000mm の大きさとした。これは一般的な成人に対

して “抱きつく” ような両手を伸ばして行う操作の際に十分作業領域を確保することができる大きさである。

マウスの操作に関してはマウスエミュレータアプリケーションを作成し、マルチタッチインタフェースによってマウス操作を行えるようにした。詳細は 4.2 節に記す。

情報表示に関しては、マルチタッチインタフェースがアクリルであることを利用し、円筒の奥部へトレーシングペーパーを貼り、円筒後方に設置したプロジェクタから背面投影することにより、実現した。その投影領域は 30inch であり、一般的なコンピュータの表示サイズと比較し、同程度以上の情報表示を行うことが可能である。カメラによる赤外線の拡散反射の認識はトレーシングペーパー越しにも問題なく行うことができ、操作面との競合も問題はない。

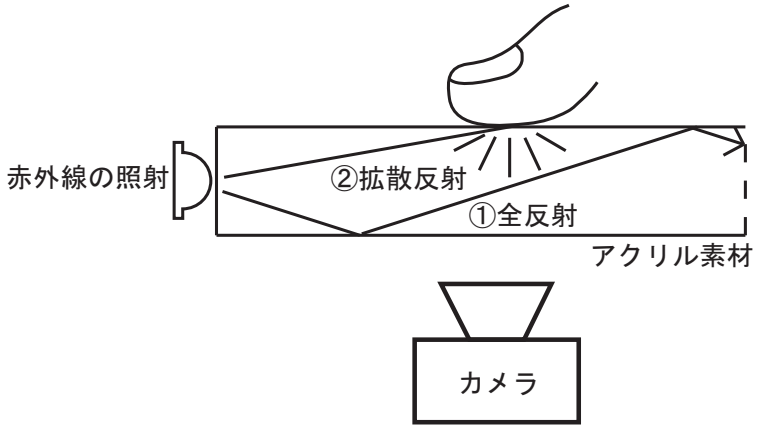


図 3.1: Han 方式のタッチインタフェースモデル

3.2 タッチパネルの実装

赤外線照射に関しては図 3.2 に示すように、アクリルパイプの上部切断面に赤外線 LED42 個を約 4cm 間隔で均等に配置した。また、その回路図を図 3.3 に示す。

今回、我々は赤外線 LED には赤外線のピーク発行波長が 870nm の東芝製 TLN231 を使用し、カメラには波長が 860nm 以下の可視光を除去する赤外線透過フィルタを装着した。これによって赤外線のみを撮影できる。赤外線のみを撮影することにより、投影画像の色空間を制限することなく色の変化に対して精度のよい接触点の識別をすることができる。

また、赤外線 LED からもれた赤外線やアクリルの断面に当たり反射した赤外線をカメラが取得してしまう。その映り込みを防ぐためにアクリルパイプの断面には黒色のビニールと断衝材を被せて遮光した。円筒面全体を撮影するためにカメラを複数個用いることとした。4 つのカメラを円筒上部に円の中心に向けて設置した場合、各カメラの画角が 45 度以上である必要がある。そこでカメラには、広角 71 度のレンズを備えた USB カメラの Microsoft ®LifeCam VX-6000 を使用した。

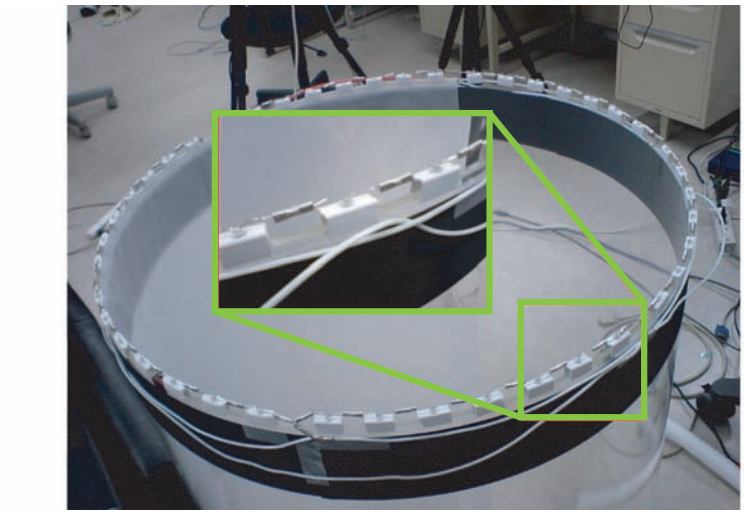


図 3.2: インタフェース上部に配置した赤外線 LED

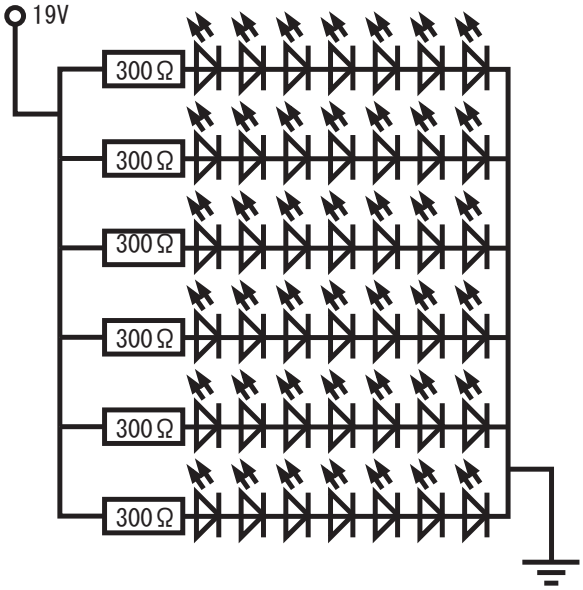


図 3.3: 赤外線 LED を発光させるための回路図

3.3　システム構成

図 3.4 に円筒型マルチタッチインタフェースのシステム構成を示す。このシステムは以下の 3 つの処理部からなる

- USB カメラからの画像を解析し、接触点の座標を得る画像解析部
- 解析したデータをマルチタッチインタフェースの操作を行うための情報に整理する座標解析部
- アプリケーションの情報表示用に投影画像を補正する投影変換部

画像解析部と座標解析部・投影変換部を分離し、ネットワーク上の別の計算機にそれぞれの処理を行わせている。これにより、高負荷のアプリケーションを実行する際にも対応できる。また、より大きな円筒型マルチタッチインタフェースを作成しカメラの台数が多くなる場合にも対応できる。

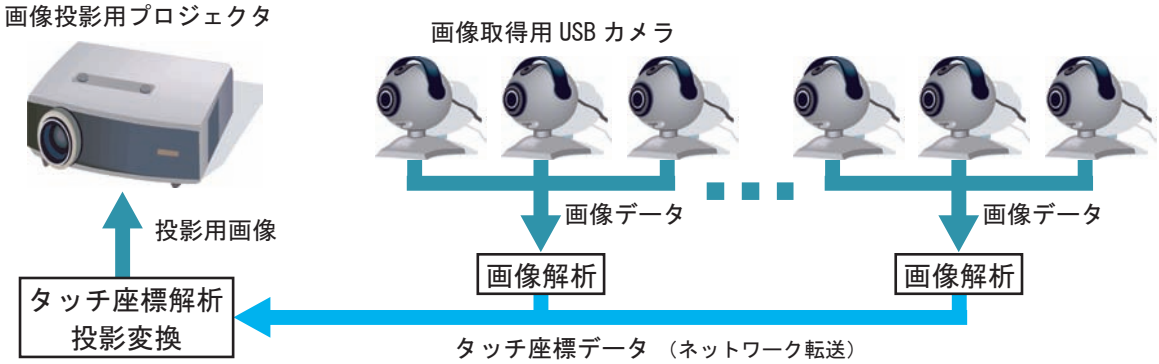


図 3.4: 円筒型マルチタッチインタフェースのシステム構成

3.3.1　画像解析

画像解析部では、まず USB カメラによって撮影した画像を全探索し接触領域を認識する。次に、そのそれぞれの接触領域の重心を得る。この重心を接触点とした。なお接触点の取得は、接触領域の大きさが縦横 3pixel 以下のものや明度が暗いものはノイズとみなして行わなかった。

また撮影対象が円筒曲面であることにより、撮影した画像上の座標から対応する円筒上の座標への変換も行う必要がある。この変換は円筒曲面をアクリルパイプに内接する多角柱に近似することで行っており、今回は図 3.5 に示すように八角柱にて近似を行っている。各カメラは八角柱の面をそれぞれ 2 面ずつ正面に捉えて撮影し、図 3.6 に示すような 2 つの四角形を撮影画像に収めることができる。この四角形の 4 つの頂点を利用して Homography 行列を作

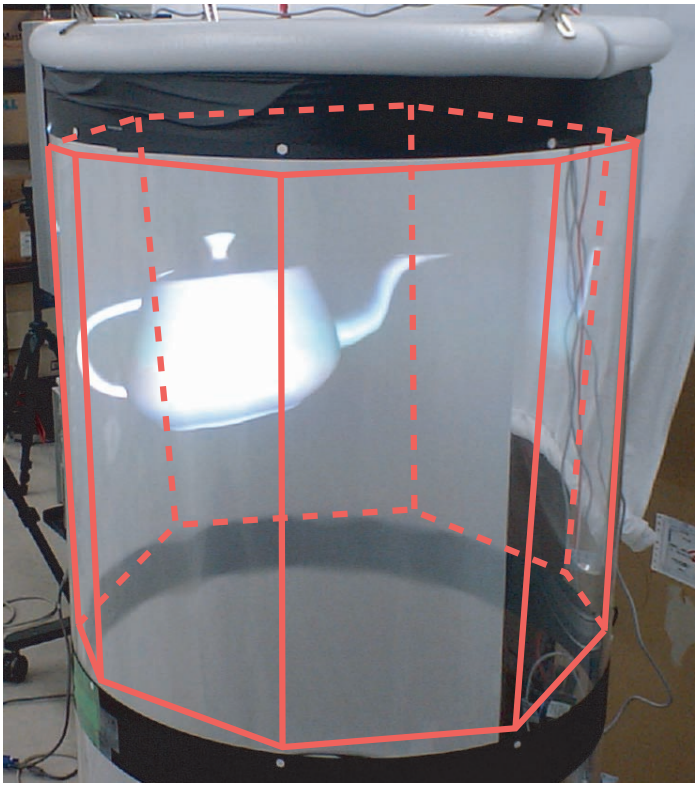


図 3.5: アクリルパイプの近似八角柱

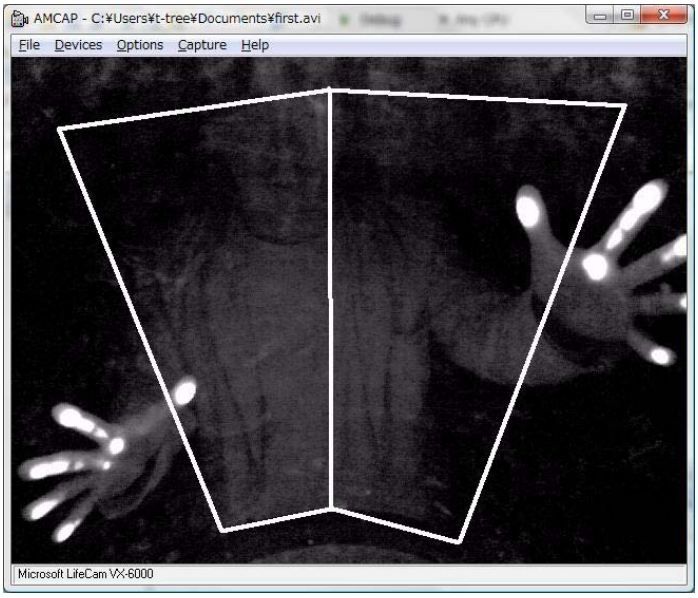


図 3.6: 接触点の撮影と近似した四角形の領域イメージ

成する。この Homography 行列を用いてカメラ画像上の点座標を、円柱座標系の点座標に変換した。

キャリブレーションの設定は八角柱の各頂点位置を順番にユーザに入力させることによって行う。これはインタフェースの設置時にのみ行えば良く、設置終了後は常に同じ設定を使うことができる。

3.3.2 座標解析

座標解析部においては、画像解析部にて取得した座標データをもとにアプリケーションが実際に使用できる形にデータを処理する。アプリケーションが複数の接触点を利用するには、前後フレーム間の時間的関係とそれぞれのフレームでの空間的関係を解析する必要がある。

接触点の追跡

複数点を取得しているので、各点を持続的にかつ一意に識別するために前フレームと現フレームを比較する。図 6 に接触点の変化の例を示す。接触点の変化は移動 (維持)、出現、消失の 3 種類からなる。取得した接触点 A が前フレームの接触領域 A' 内に存在するとき接触点 A' が接触点 A に連続して移動したものと判断した。また、前フレームに該当する領域がなかった接触点 B は新しく増えた接触点、現フレームに存在する接触点に重ならなかった接触領域 C' は消失した接触点として処理した。

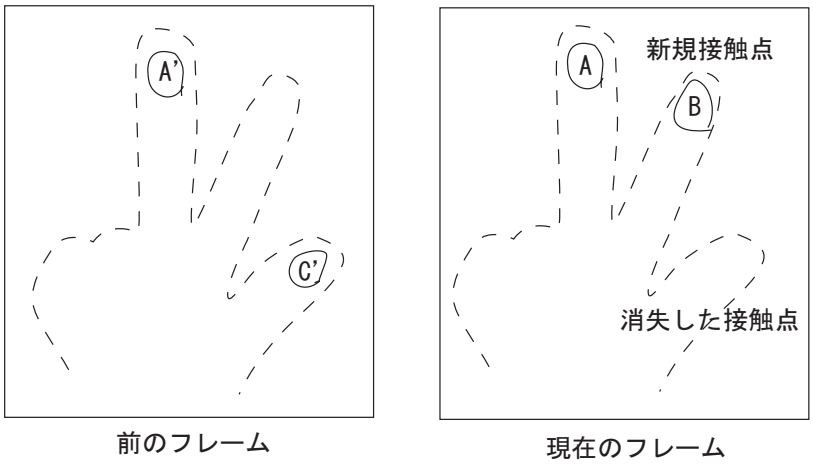


図 3.7: 接触点の移動、出現、消失の例

接触点のクラスタリング

手のひら全体を利用するようなジェスチャ操作においては、接触点をある手による集合だと判断する必要がある。そのため、接触点をクラスタリングした。

クラスタは近傍にある接触点をまとめたもので、クラスタリングでは、ある閾値以下の距離にある接触点同士を一つのクラスタとしてまとめた。また、クラスタの近傍に出現した接触点は随時そのクラスタに追加することにした。

クラスタ内の接触点が消失したり、クラスタ近傍に接触点が出現した場合にもクラスタは同一のものとして認識される。これにより例えば、人差し指で接触し、そのままその近傍に中指を接触し、人差し指を離した場合でも、中指は最初のクラスタとして認識される。これを利用すれば、右手で作成された接触点のクラスタを左手による接触点へと引き渡すことができ、作業領域を有効に活用することができる。なおクラスタの座標は、クラスタ内に存在する全ての接触点座標の重心位置として定義した。

3.3.3 アプリケーションへの通知

アプリケーションには図 3.8 に示すような形で接触点のデータが渡される。データ自体はカメラの撮影が行われる度にイベントとして発行され、その頻度は接触点の有無にはよらない。

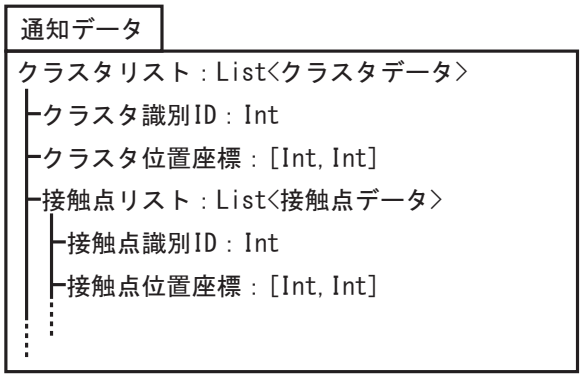


図 3.8: イベントとしてアプリケーションに渡されるデータ

実装は Windows 上で C # を用いて行った。プログラム自体は DLL として作成しているため、このシステムを利用するには作成した C # アプリケーションにインポートし組み込むことができる。そして、DLL 内にある管理オブジェクトを生成し、スレッドとして実行することでイベントの発行が始まる。もとのプログラム側ではこのイベントを受け取ることができる。

3.3.4 投影変換

投影変換部では、円筒側面に向けて投影を行う際に生じる、投影像の曲面に沿ってのゆがみを補正をする。本研究では図 3.9 のような方法でゆがみの補正を行った。

まず、元画像を仮想空間内での円筒曲面に平行投影する。その円筒を仮想カメラで撮影し、得た画像を実空間のプロジェクタから円筒型マルチタッチインタフェースに投影する。結果として、投影された画像はユーザにとってゆがみのないものに見える。

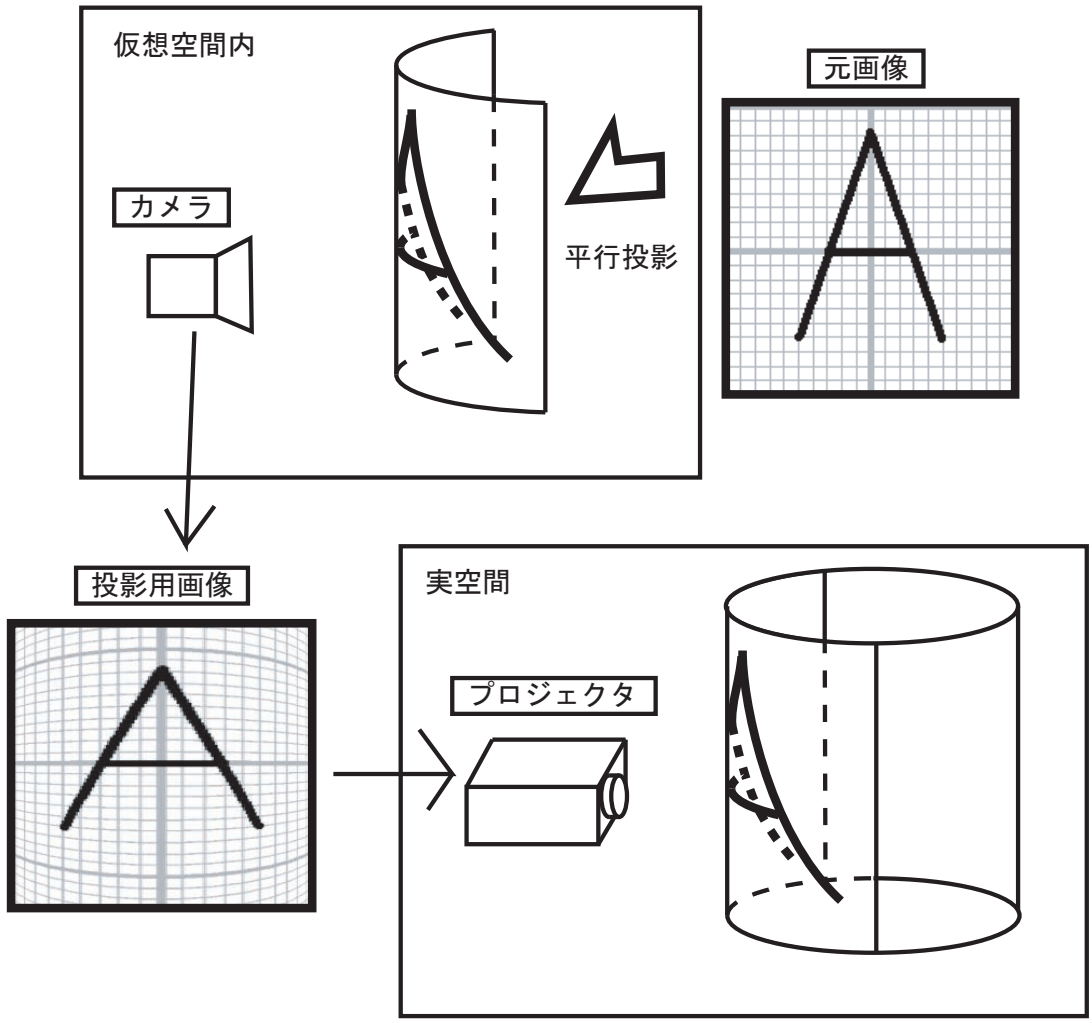


図 3.9: 投影画像のゆがみ補正

第4章　円筒型マルチタッチインタフェースの操作

円筒型マルチタッチインタフェースを利用したアプリケーションとして、マウスエミュレータと 3D オブジェクトを閲覧する 3D オブジェクトビューアの 2 つのアプリケーションを作成した。また、3D 仮想空間をウォークスルーするための操作法を作成した。実際に円筒型マルチタッチインタフェースで 3D オブジェクトビューアを操作している様子を図 4.1 に示す。

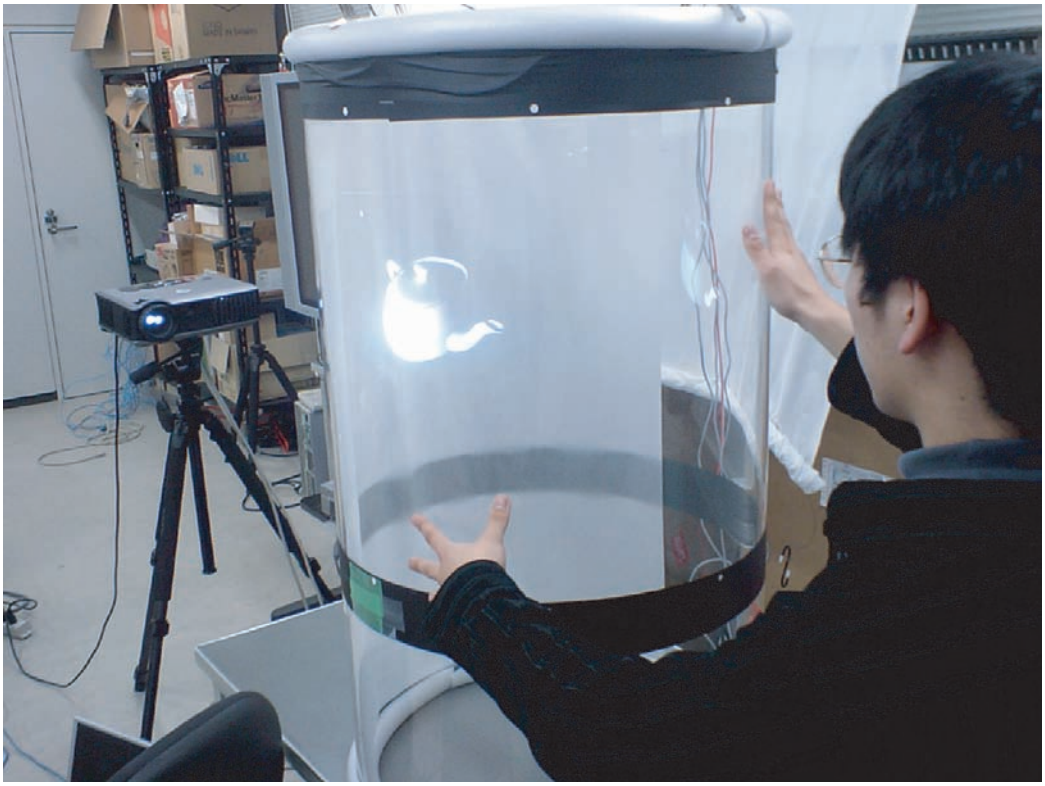


図 4.1: 円筒型マルチタッチインタフェースで 3D オブジェクトビューアを操作

4.1　円筒型マルチタッチインタフェースの操作法

円筒型マルチタッチインタフェースをアプリケーションに適応するための操作法を考察した。

マルチタッチインタフェースの動作によるジェスチャ

マルチタッチインタフェースの手を用いた操作の基本的なジェスチャとして次の 4 種類が挙げられる。

- 触る/離す

  操作面に対して手を接触させる、そして接触した手を操作面から離す操作。

- 移動

  接触点を操作面上で平行移動する操作。

- ひねる

  接触点をある点を中心に回転する操作 (図 4.2)。

- 集中/拡散

  接触点をある点を中心に近づける、遠ざける操作 (図 4.3)。

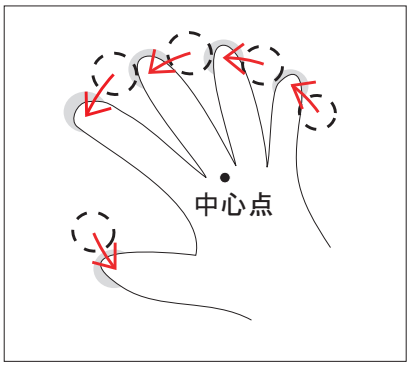


図 4.2: 接触点の回転

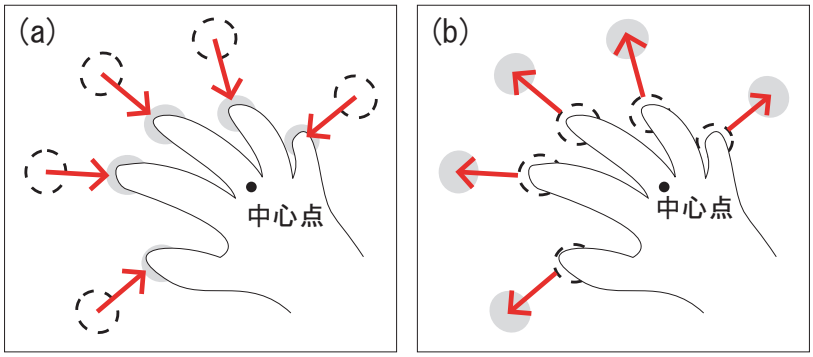


図 4.3: 接触点の (a) 集中/(b) 拡散

また、これらの動作も実際に接触している点の数などを調べることでより多くのジェスチャとして認識することができる。

このようなジェスチャを用いることで、”マウスでオブジェクト上でマウスボタンを押下し、目標地点でボタンを離す”という操作が”オブジェクトを中心に掴むように接触点を集中させ、そのまま目標地点まで移動し、目標地点を中心に拡散する”というように直感的に行うことができる。

円筒型マルチタッチインタフェース独自の操作

これらのジェスチャ自体は平面型のマルチタッチインタフェースにおいても実行可能である。しかし、両手を用いた操作において円筒型マルチタッチインタフェースでは点と点を結ぶような直線を考えるとき、表面を結ぶ線 (図 4.4a) とは別に円筒内部を通過する 3 次元の直線 (図 4.4b) を定義できる。例えば両手で接触することで直線を定義し回転ジェスチャを行えば、3 次元空間を指定した回転などを直感的に行うことができる。このように円筒型マルチタッチインタフェースを利用することで、3 次元の操作を直感的に行うことができる。

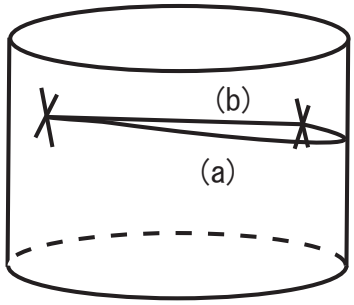


図 4.4: 2 点間の直線

4.2 マウスエミュレータ

円筒形マルチタッチインタフェースを用いて既存の GUI アプリケーションを操作するためのマウスエミュレータを作成した。

設計

円筒型マルチタッチインタフェースでは操作面全体を利き手だけで操作することが困難である。よって、マウスカーソルの操作は片手のみを利用し、どちらの手を使っても行えるように設計を行った。

ユーザはマウスカーソルの移動をどちらの手で行っても同じく 1 本の指を使うことで行う。マウスカーソルの移動は円筒上の接触点の位置とスクリーンの位置に 1 対 1 に対応している。

マウスボタンの押下、開放を行うには図 4.5 に示す操作を行う。ユーザは、まず 1 本目の指（図 4.5-①）と 2 本目の指（図 4.5-②）を順に接触させることで、図 4.5 の分割線に示すように領域を分ける。マウスボタンの押下は、分割線より①側に 3 本目の指を接触させる（図 4.5-③）ことによってマウスの左ボタンを、分割線より②側を 3 本目の指を接触させる（図 4.5-④）ことによってマウスの右ボタンを行うことができる。

ボタンの開放は③もしくは④を離すことで実行できる。ドラッグ & ドロップについてはクリック状態で 3 点を接触させたまま、4.5-①のみを動かせばドラッグでき、③もしくは④を離せばドロップできる。また、ダブルクリックは図 4.5-③をマウスでのダブルクリックと同じようにすばやく二度接触することで実行できる。

このように設計することによって右手、左手どちらを使っても同じ指を使ってそれぞれのクリックを行うことができる。また、2 つの点を先に接触させることによって 3 点目の入力を判断するため、手が横や下を向いた場合でも問題なく動作する。これは最初の接触点と 2 つ目の接触点の位置関係から 3 つ目の接触点の位置によって操作を選択しているため、指の制限はない（図 4.5-③’）。

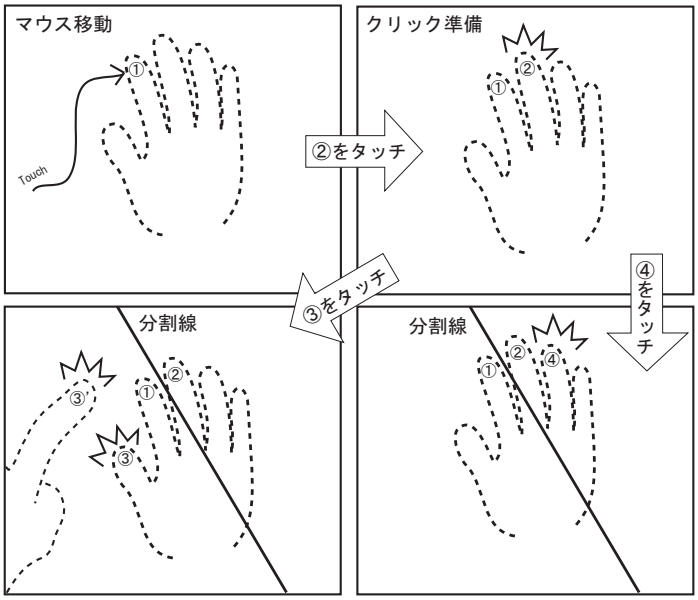


図 4.5: マウスエミュレータによるクリック操作方法

実装

マウスエミュレータは Windows 上で、SendInput API を利用して実装した。

マウスカーソルの移動位置はインタフェース表面にマッピングした座標系によって一意に決定している。

左右のボタン押下の判断は以下のように求めている。(それぞれのパラメータは図 4.6 を参照)

まず①、②それぞれの接触点座標を $\mathbf{P_1}$、$\mathbf{P_2}$ とし、$\mathbf{P_1}$ から $\mathbf{P_2}$ へのベクトル $\mathbf{d_{12}}$ を求める。ただし、$\mathbf{P}$ は 3 次のベクトルとし、z 軸座標は 0 とする。

$$\mathbf{d_{12}} = \mathit{normalize}(\mathbf{P_2} - \mathbf{P_1})$$

さらに、このベクトルに垂直なベクトル $\mathbf{d'_{12}}$ を求める。

$$\mathbf{d'_{12}} = \begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathbf{d_{12}}$$

この $\mathbf{d'_{12}}$ が図 4.5 の分割線となる。次に、3 点目の接触点座標を $\mathbf{P_a}$ としたときの $\mathbf{P_1}$ から $\mathbf{P_a}$ へのベクトル $\mathbf{d_{1a}}$ を求める。

$$\mathbf{d_{1a}} = \mathit{normalize}(\mathbf{P_a} - \mathbf{P_1})$$

この 2 つのベクトルの外積を求める。

$$\mathbf{a} = \mathbf{d'_{12}} \times \mathbf{d_{1a}}$$

そして、その $\mathbf{z}$ 軸方向の符号から 3 点目の接触点の位置を求めることができる。

$$\begin{cases} \mathbf{a_z} < 0 \rightarrow left \\ \mathbf{a_z} \geq 0 \rightarrow right \end{cases}$$

4.3　3D オブジェクトの操作

円筒型マルチタッチインタフェースの 3D 操作を検討するために 3D オブジェクトのビューアを作成した。

設計

3D オブジェクトビューアにおいてはオブジェクトの向きの調整とカメラの位置・姿勢の調整をそれぞれ “片手による操作” と “両手による操作” にて行う。また、オブジェクトは原点位置を中心に表示しており、カメラは常に原点を向いている。

図 4.7 に片手による操作の例を示す。インタフェースを片手で撫でることによって表示しているオブジェクトの回転を行う。図 4.8 に両手による操作の例を示す。両手の接触点の幅を調整することでカメラの前後移動 (図 4.8a) を、両手をそろえて同じ方向にスライドすることでカメラから原点までの距離は固定したまま、カメラの回転移動 (図 4.8b) を行う。また、両手をそれらの中間位置を中心に回転させることでカメラの傾きを変更する (図 4.8c)。なお、カメラの注視点を変更するような操作は実装しなかった。

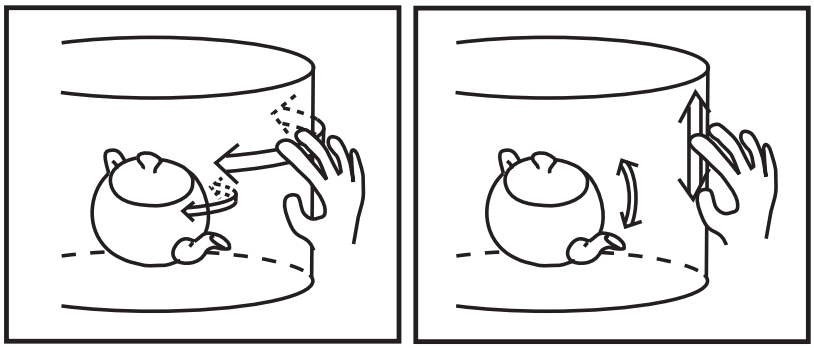


図 4.7: 3D ビューアの片手によるのモデルの操作

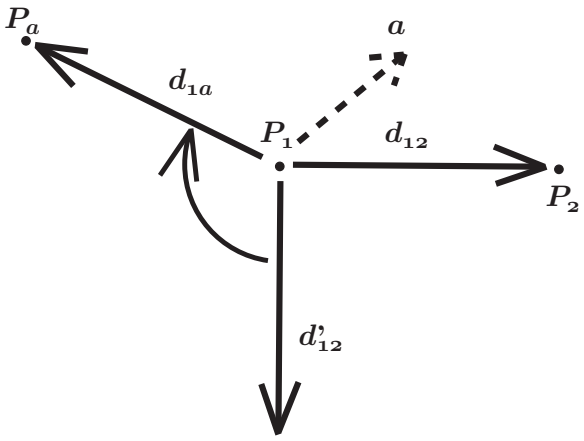
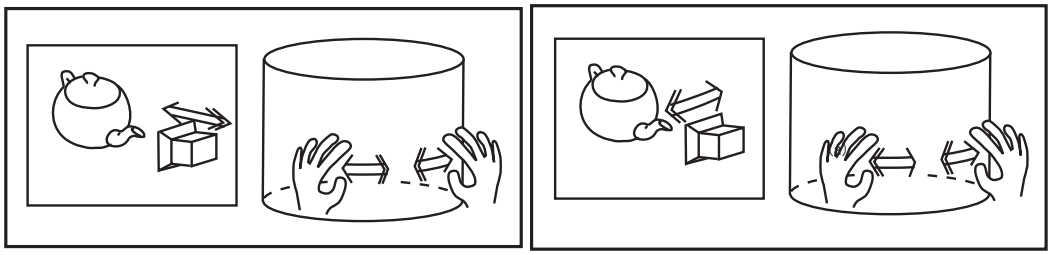
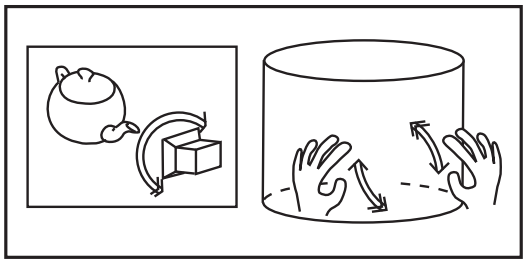


図 4.6: マウスエミュレータの算出パラメータ



(a) 両手の間隔によるズームの操作

(b) 空間の原点を中心にカメラの回転移動



(c) 注視点を固定したカメラの傾き操作

図 4.8: 3D ビューアの両手によるカメラの操作

実装

オブジェクトの回転は回転行列を随時求め、カメラの操作についてもカメラの位置ベクトルと、カメラの姿勢ベクトルを随時求めた。ここで使用する接触点は円周の角度と高さの 2 次元データとしてシステムから受け取っているが、3 次元座標に変換するために、円柱の半径を 1 とした円柱座標系上の座標 $\mathbf{P}$ に変換し、さらに原点を円柱の中心に持つ xyz 座標系上の座標 $\mathbf{V}$ へと変換した。

片手による操作

オブジェクトの回転行列は、前フレームの接触点 $\mathbf{V_0}$ と現フレームの接触点 $\mathbf{V_1}$ の位置を用いて算出する。(それぞれのパラメータは図 4.9 を参照)

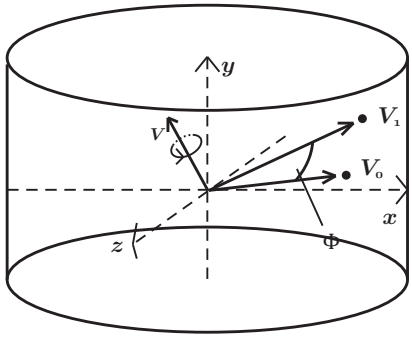


図 4.9: オブジェクトの回転行列算出のパラメータ (3D ビューアの片手操作)

まず、回転を行う軸となるベクトル $\mathbf{v}$ を求める。

$$\mathbf{v} = \mathit{normalize}(\mathbf{V_0} \times \mathbf{V_1})$$

次に、$\mathbf{v}$ を軸とした回転量 ϕ を求める。

$$\phi = \arccos(\mathit{normalize}(\mathbf{V_1}) \cdot \mathit{normalize}(\mathbf{V_0}))$$

オブジェクトの回転変化量を表す回転行列 $\mathbf{R}_\theta$ を $\mathbf{v}$、ϕ を使用して次のように定義する。

$$\begin{bmatrix} \mathbf{v_x}^2 + (1 - \mathbf{v_x}^2)\cos\phi & \mathbf{v_x}\mathbf{v_y}(1-\cos\phi) + \mathbf{v_z}\sin\phi & \mathbf{v_x}\mathbf{v_y}(1-\cos\phi) - \mathbf{v_y}\sin\phi \\ \mathbf{v_x}\mathbf{v_y}(1-\cos\phi) - \mathbf{v_z}\sin\phi & \mathbf{v_y}^2 + (1 - \mathbf{v_y}^2)\cos\phi & \mathbf{v_y}\mathbf{v_z}(1-\cos\phi) + \mathbf{v_x}\sin\phi \\ \mathbf{v_z}\mathbf{v_x}(1-\cos\phi) + \mathbf{v_y}\sin\phi & \mathbf{v_y}\mathbf{v_z}(1-\cos\phi) - \mathbf{v_x}\sin\phi & \mathbf{v_z}^2 + (1 - \mathbf{v_z}^2)\cos\phi \end{bmatrix}$$

最後に、前フレームでのオブジェクトの回転行列 $\mathbf{R_0}$ と $\mathbf{R}_\theta$ を使用して現フレームでの回転行列 $\mathbf{R_1}$ を以下のように求める。

$$\mathbf{R_1} = \mathbf{R}_\theta \mathbf{R_0}$$

この回転行列 $\mathbf{R_1}$ をオブジェクトに適用し傾いたオブジェクトを作成する。

両手による操作

カメラの位置も同様に、接触しているそれぞれの手 a、b の前フレームの接触点 $\mathbf{V_{0a}}$、$\mathbf{V_{0b}}$ と手 a、b の現フレームの接触点 $\mathbf{V_{1a}}$、$\mathbf{V_{1b}}$ の座標を用いて算出する。(それぞれのパラメータは図 4.10 を参照)

まず、前フレームの 2 点間の中心座標 $\mathbf{V_{0c}}$ と現フレームの 2 点間の中心座標 $\mathbf{V_{1c}}$ を求める。

$$\mathbf{V_{0c}} = \frac{\mathbf{V_{0a}} + \mathbf{V_{0b}}}{2}$$

$$\mathbf{V_{1c}} = \frac{\mathbf{V_{1a}} + \mathbf{V_{1b}}}{2}$$

この 2 点間の差分 ΔV を求めることでカメラの円周方向の回転量 $\Delta\theta$ と縦方向の回転量 $\Delta\phi$ を求めている。

$$\mathbf{\Delta V} = \mathbf{V_{1c}} - \mathbf{V_{0c}}$$

$$\Delta\theta = \arctan(\frac{\mathbf{\Delta V_z}}{\mathbf{\Delta V_x}})$$

$$\Delta\phi = \arccos(\frac{\sqrt{\mathbf{\Delta V_x}^2 + \mathbf{\Delta V_z}^2}}{\sqrt{\mathbf{\Delta V_x}^2 + \mathbf{\Delta V_y}^2 + \mathbf{\Delta V_z}^2}})$$

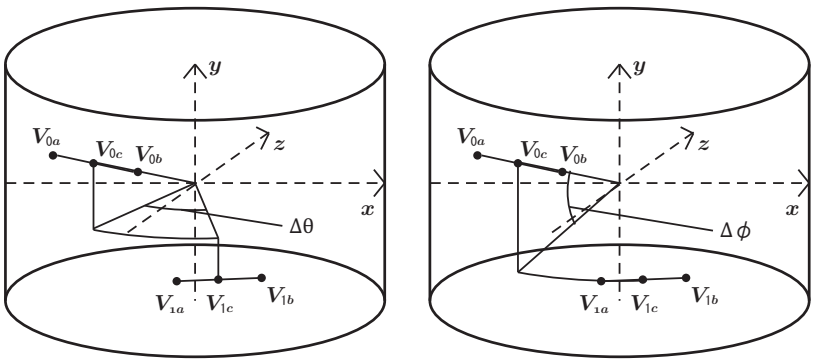


図 4.10: カメラの回転移動量算出のパラメータ (3D ビューアの両手操作)

次に、前フレームの 2 点間の距離 l_0 と現フレームの 2 点間の距離 l_1 を求め、カメラの原点までの距離の変更（カメラの前後移動）量 Δr を求める。(それぞれのパラメータは図 4.11 を参照)

$$l_0 = \|\mathbf{P_{0a}} - \mathbf{P_{0b}}\|$$
$$l_1 = \|\mathbf{P_{1a}} - \mathbf{P_{1b}}\|$$
$$\Delta r = l_1 - l_0$$

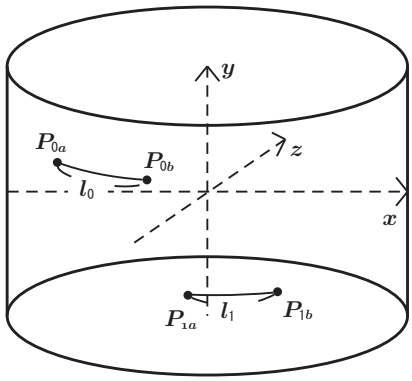


図 4.11: カメラの前後移動量算出のパラメータ (3D ビューアの両手操作)

最後にカメラの傾きを更新するための姿勢制御ベクトルを更新する。姿勢制御ベクトルを更新するためにはまず、前フレームでの接触点の 2 点間の向きベクトル $\mathbf{d_0}$ と、現フレームでの接触点の 2 点間の向きベクトル $\mathbf{d_1}$ とのなす角 θ を求める。

$$\mathbf{d_0} = \mathit{normalize}(\mathbf{P_{0b}} - \mathbf{P_{0a}})$$
$$\mathbf{d_1} = \mathit{normalize}(\mathbf{P_{1b}} - \mathbf{P_{1a}})$$
$$\theta = \arccos(\mathbf{d_0} \cdot \mathbf{d_1})$$

次に、回転方向を調べるために $\mathbf{d_0}$ と $\mathbf{d_1}$ の外積 $\mathbf{a}$ を計算し回転方向を求め、回転量 ϕ を再度求める。

$$\mathbf{a} = \mathbf{d_0} \times \mathbf{d_1}$$

$$\begin{cases} \mathbf{a_z} < 0 \rightarrow \phi = \theta \\ \mathbf{a_z} \geq 0 \rightarrow \phi = -\theta \end{cases}$$

カメラの現フレームの回転の変化量を表す行列 $\mathbf{T}_\phi$ を ϕ および現在のカメラの位置 $\mathbf{C}$ により以下のように定義する。

$$\begin{bmatrix} \mathbf{C_x}^2 + (1 - \mathbf{C_x}^2)\cos\phi & \mathbf{C_x}\mathbf{C_y}(1-\cos\phi) + \mathbf{C_z}\sin\phi & \mathbf{C_x}\mathbf{C_y}(1-\cos\phi) - \mathbf{C_y}\sin\phi \\ \mathbf{C_x}\mathbf{C_y}(1-\cos\phi) - \mathbf{C_z}\sin\phi & \mathbf{C_y}^2 + (1 - \mathbf{C_y}^2)\cos\phi & \mathbf{C_y}\mathbf{C_z}(1-\cos\phi) + \mathbf{C_x}\sin\phi \\ \mathbf{C_z}\mathbf{C_x}(1-\cos\phi) + \mathbf{C_y}\sin\phi & \mathbf{C_y}\mathbf{C_z}(1-\cos\phi) - \mathbf{C_x}\sin\phi & \mathbf{C_z}^2 + (1 - \mathbf{C_z}^2)\cos\phi \end{bmatrix}$$

$\mathbf{T}_\phi$ と前フレームの回転行列 $\mathbf{T_0}$ を用いて現フレームでのカメラの回転行列 $\mathbf{T_1}$ を求める。

$$\mathbf{T_1} = \mathbf{T}_\phi \mathbf{T_0}$$

現フレームのカメラの位置座標 $\mathbf{C}$ における基準となるカメラの上向きベクトル $\mathbf{v}$ を求める。

$$\mathbf{v} = \mathit{normalize}(\mathit{normalize}(\mathbf{C}) \times \mathit{normalize}(\mathit{normalize}(\mathbf{C}) \times e_y))$$

最後に、カメラの基準となる上向きベクトル $\mathbf{v}$ に回転行列 $\mathbf{T_1}$ をかけることによって正しい上向きベクトル $\mathbf{v_1}$ を求める。

$$\mathbf{v_1} = \mathbf{T_1 v}$$

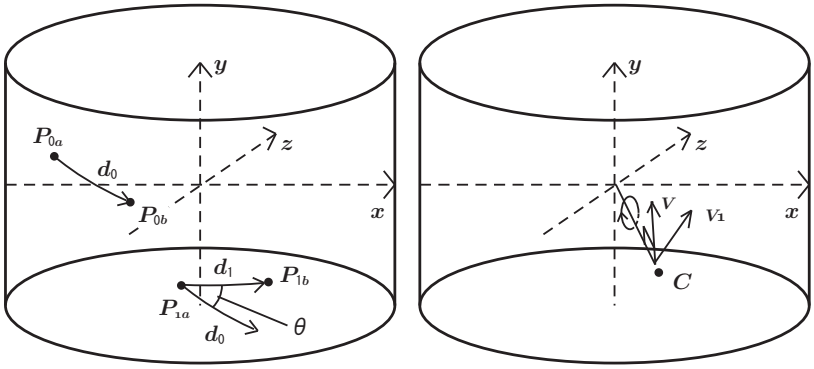


図 4.12: カメラの回転量算出のパラメータ (3D ビューアの両手操作)

4.4 3D ウォークスルー用インタラクション手法

3D 仮想空間をウォークスルーするための操作インタラクションを設計し、実装した。

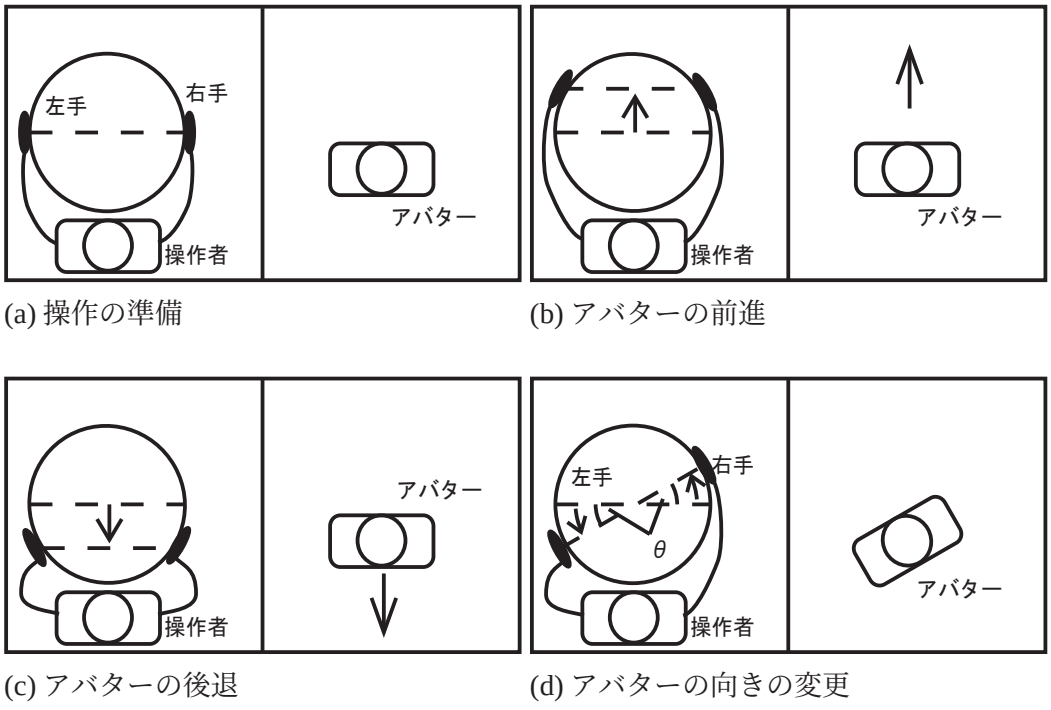
設計

仮想空間内の移動

図 4.13 に示すように、ユーザは両手のひらを使って円筒型マルチタッチインタフェースをはさむように操作する。まずユーザは両手を円筒側面に接触させる (図 4.13a)。次に、最初に接触させた位置から両手を前方にスライドさせることでアバターを前進させ (図 4.13b)、同様に両手を手前にスライドさせることでアバターを後退させることが出来る (図 4.13c)。また、ユーザが両手をスライドさせた状態で操作面を触り続けることで、アバターはその方向に移動し続ける。

アバターの移動方向を変更するには、ユーザは円筒の中心を軸として回転するように両手をスライドさせる (図 4.13d)。スライドした角度 (図 4.13d の θ) に応じてアバターの向く方向は変化する。

また、ユーザがアバターを前進させながら両手を横にスライドさせることでアバターは移動しながら方向転換を行い、弧を描くように移動する。これは人間が方向を変更する際にわざわざ立ち止まり旋回するのではなく、前進しながら方向を変更することと同じ動作である。



(a) 操作の準備　(b) アバターの前進

(c) アバターの後退　(d) アバターの向きの変更

図 4.13: ウォークスルー操作のインタラクション

仮想空間内の視線変更

仮想空間内においても現実世界同様にビルの屋上や壁などに情報が表示されることが想定される。こういった情報を閲覧するために、アバターの視線方向を上や下に変更する必要がある。これには図 4.14 に示すような、接触している手をひねる様に回す操作を対応づけた。移動操作中にそのまま手をひねるだけでスムーズに適応できるので、視点変更のために手を置き換えなくても良い。

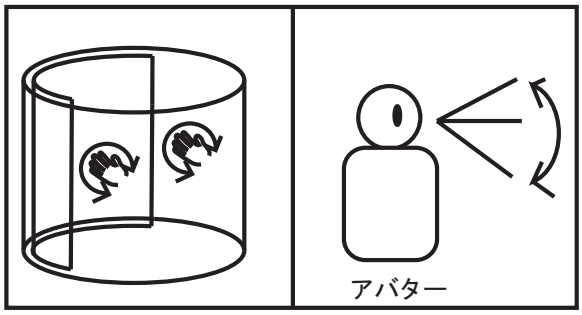


図 4.14: ウォークスルーの視線変更

実装

ウォークスルーアプリケーションとして SecondLife ®を用いた。円筒型マルチタッチインタフェースへの操作をキーボードやマウスイベントへ変換し、それを発行することで SecondLife® の操作を行った。実際に操作を行っている様子を図 4.15 に示す。

図 4.15: 円筒型マルチタッチインタフェースによるセカンドライフの操作

仮想空間内の移動

移動方向の決定は接触点座標の円筒周囲のラジアン角のみを用いて算出する。(それぞれのパラメータは図 4.16 を参照)

まず、接触点座標 $\mathbf{P_a}$ と $\mathbf{P_b}$ の各ラジアン角 θ_a と θ_b より円筒の半径を 1 と仮定し、2 点を通るの直線 l を作る。

$$l : y = \frac{\sin\theta_b - \sin\theta_a}{\cos\theta_b - \cos\theta_a}(x - \cos\theta_a) + \sin\theta_a$$

次に、原点から求めた直線 l までの距離 d を求める。これがアバターの歩行速度となる。

$$d = \frac{\left| -\cos\theta_a \dfrac{\sin\theta_b - \sin\theta_a}{\cos\theta_b - \cos\theta_a} + \sin\theta_a \right|}{\sqrt{\left(\dfrac{\sin\theta_b - \sin\theta_a}{\cos\theta_b - \cos\theta_a} \right)^2 + 1}}$$

さらに、進行方向は直線 l の垂線方向ベクトル $\mathbf{v}$ とした。

$$\mathbf{v} = \begin{bmatrix} \cos\theta_a - \cos\theta_b \\ \sin\theta_b - \sin\theta_a \end{bmatrix}$$

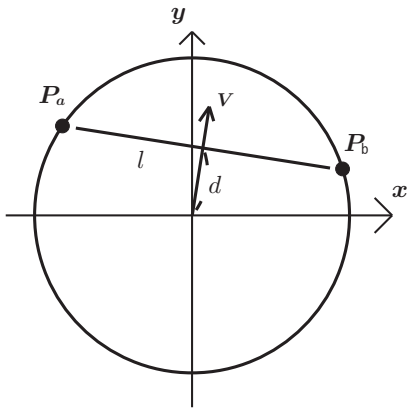


図 4.16: 移動方向算出のパラメータ (ウォークスルーの移動)

仮想空間内の視線変更

仮想空間内での視線変更は、手のひらを円筒型マルチタッチインタフェースに押し当ててひねることによって実行する。ひねり操作の回転量取得は 4.3 節での両手操作による傾きの検出法と同じ方法で行う。ただし、4.3 節では両手の位置関係を元に算出したが、今回は一つの

クラスタの中の識別 ID のもっとも小さい接触点 $\mathbf{P_a}$ とその次に小さい接触点 $\mathbf{P_a}$ を利用して求める。それぞれの接触点の前フレームにおける座標を $\mathbf{P_{0a}}$ と $\mathbf{P_{0b}}$、現フレームにおける座標を $\mathbf{P_{1a}}$ と $\mathbf{P_{1b}}$ として回転量 θ は以下のように求められる。

$$\mathbf{d_0} = \textit{normalize}(\mathbf{P_{0b}} - \mathbf{P_{0a}})$$

$$\mathbf{d_1} = \textit{normalize}(\mathbf{P_{1b}} - \mathbf{P_{1a}})$$

$$\theta = \arccos(\mathbf{d_0} \cdot \mathbf{d_1})$$

また、今回の場合は両手のひねり量の平均を現フレームでのひねり量として使用している。

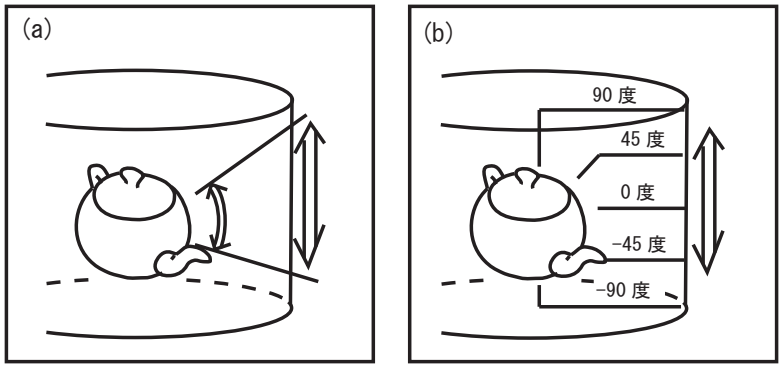
第5章　議論

円筒型マルチタッチインタフェースをサンプルアプリケーションにて実際に使用したところ、操作の認識精度は良好だった。しかし、アプリケーションを操作するうえで幾つか問題点が挙がった。

八角柱に近似した際の誤差に関しては、誤差はほとんどの場合において問題にはならなかった。しかし、マウスエミュレータを用いて描画ソフトなどの実際に移動した軌跡を表示するソフトを動かしてみる場合では誤差が目立った。縦方向に関しては誤差は感じられないが、横方向に関してはまっすぐ線を引くことができなかった。これに関してはキャリブレーションの手法を多角形近似ではなく、より正確な手法に改善することで解決できる。

また、マウスエミュレータに関しては操作面を全て使う必要はなく、操作面の前面のみを使うことで画面全体を操作することが出来た。操作面の側面にユーザの望むショートカットボタン等を実装することで、より操作が行いやすくなると考えられる。他にも、操作面をユーザの意図に合わせて拡大するような仕組みを用意すれば、ユーザが細かい作業を行いたい場合などで有効になる。これについては今後も検討を行う。

3D ビューアのオブジェクトの回転操作において、縦方向の回転量を当初図 5.1a のように実際の回転量にあわせて定義していたが、ユーザの意図したようには動かなかった。そこで、図 5.1b のように縦方向の操作量に角度をリニアに振り分けたところユーザは違和感なく操作を行うことができた。ただし、この対応付けはアプリケーションによって最適な設定が異なると考えられる。今後のアプリケーションの開発の際、各種の対応付けを検討してゆく予定である。



(a) 手の移動量から回転量を算出

(b) 円筒側面の高さを 90 度から-90 度まで均等に振り分ける

図 5.1: 3D ビューアの縦方向の回転

第6章　関連研究

6.1　非平面型マルチタッチインタフェース

平面型ではないマルチタッチインタフェースシステムとしてGrossmanらの半球状のデバイスがある[GWB04]。この研究では半球状のマルチタッチインタフェースの内部にボリュームディスプレイを設置し、内部に表示されている3Dオブジェクトの操作を行う。ただし、このシステムにおいては、ユーザの指の位置を認識するためにセンサーを取り付ける必要がある。情報表示に関しても、現時点においてはボリュームディスプレイの精度としての問題点があり、一般的な作業を行うための十分な情報提示を行えていない。

これに対して本研究ではユーザへの特殊な装置の装着は行うことなく操作ができることを可能にしている。また、情報の提示についてもプロジェクタからのリアプロジェクションを行うことにより、一般的な大画面投影と同じ程度の情報表示が行え、作業を行うのに十分な情報量を提供できているといえる。

またCassinelliらは伸縮可能なスクリーンをタッチインタフェースとして利用したKhronos projectorを作成している[CI05]。このKhronos projectorでは、ユーザからの操作が接触と非接触の2値的なものではなく、スクリーンを押した奥行き量によって操作を行う。実際の奥行き量は押されたスクリーンの伸びた部分を認識することで取得している。しかし、この方法では実際に押し込んだ部分の他にその周辺もスクリーンが伸び、操作位置の認識精度が不十分である。

6.2　円筒型インタフェース

円筒型のインタフェースとして川西らの円筒ディスプレイを用いた研究がある[川西 02]。この研究では、円筒型ディスプレイを全方位からのアクセス可能な情報提示インタフェースとして利用している。しかし、この研究では音声や、ユーザの位置情報のみを入力として利用しているため、アプリケーションの操作のような詳細な操作は行えない。本研究では、円筒型という全方位からのアクセスを可能にしつつ、タッチインタフェースとしてユーザからの細かい操作を可能にしている。

他にも、柴田らはBiblioRollと呼ばれる円筒型小型デバイスを作成している[柴田 06]。これは円筒内部に情報表示装置を組み込み、操作と情報表示を同時に行えるようにしている点で本研究と同じである。しかし、このデバイスは操作法が円筒本体の回転と本体に設置された2つのボタンのみである。そのため、操作の種類は本研究では操作をマルチタッチインタ

フェースによって行うため、より多彩な操作を行うことができる。

6.3　マルチタッチインタフェースの 3 次元操作への応用

Gingold らは 3D オブジェクトへのテクスチャのマッピングをマルチタッチインタフェースによってインタラクティブに行う手法を示している [GDHZ06]。また、Nakashima らや、Hancock らも平面型マルチタッチインタフェースを用いて 3D オブジェクトの操作手法を提案している [HCC07][NMKT05]。しかし、これらはいずれも 3D オブジェクトの位置の固定や姿勢制御などを行うために、実際の操作を目的としたジェスチャとは別に、それを補助するためのジェスチャなどが必要となる。

上記の方法に対して本研究では円筒型マルチタッチインタフェースにより全方向から操作を行うことが可能なので、姿勢制御などの補助的な操作を削減することができる。

第7章　今後の展望

本研究で作成した円筒型マルチタッチインタフェースは、情報表示をリアプロジェクションによって行っている。そのため、ユーザが裏側から操作を行うと、操作面に影を落としてしまうため、全方位から操作を行うことは困難となっている。円筒型マルチタッチインタフェースの表示手法の最終目標は円筒の内部にボリュームディスプレイを設置し、全方位から立体物を閲覧、操作できるようになることである。しかし、近年ボリュームディスプレイは表示サイズ、精度ともに向上が見られるが、実際の利用にはまだ不十分である。そこで、研究の次の段階として人間の目の付近に設置したカメラからの取得映像と、ヘッドマウントディスプレイを利用し円筒内部に仮想的に 3 次元オブジェクトをカメラからの映像と合成し、ボリュームディスプレイを使用しない全方向から閲覧可能な情報表示を実装する。そして、ボリュームディスプレイを利用し、全方位から操作が可能となった場合の情報の表示方法、およびその操作手法の検討を行う予定である。

本研究において円筒型マルチタッチインタフェースの操作は接触点の動きをによるジェスチャを用いて行った。しかしマルチタッチインタフェースで使用できるジェスチャには、動作を基準にしたジェスチャのほかに接触領域を利用したジェスチャもある。平面型マルチタッチインタフェース上で利用を検討した先行研究もなされている [WB03]。しかし、本研究ではこの様なジェスチャは検討しておらず、今後の課題として研究したいと考えている。

第8章　まとめ

本研究では従来の平面型マルチタッチインタフェースとは異なる円筒形状の操作面を持つ円筒型マルチタッチインタフェースを作成した。この円筒型マルチタッチインタフェースは、円筒型の特徴として 360 度連続した作業領域を持つ。

作成にあたっては、Han の提案した手法に基づいてアクリルパイプを用いてパネルを作成した。これにより、設計において定めた “素手による操作”、“操作部と一体化した情報表示” を備えた円筒型マルチタッチインタフェースを構築することができた。

さらに、円筒型マルチタッチインタフェースでのジェスチャ操作や奥行きを利用した操作手法を検討した。そして、円筒型マルチタッチインタフェースを利用したアプリケーションとして 3D オブジェクトビューアや、マウスエミュレータを作成し、さらにウォークスルーアプリケーションの操作法を作成した。これによって、“従来のアプリケーション資産を利用するためのマウスエミュレーション” を実現することができ、さらに操作面の奥行きを利用することで 3 次元的な操作が実現できることを確かめた。

謝辞

本論文の執筆にあたり、指導教員である田中二郎教授、志築文太郎講師をはじめ、三末和男准教授、高橋伸講師には多くのご助言やご指導をいただきました。心より感謝申し上げます。

また、IPLAB の皆様にも大変お世話になりました。特に WAVE チームの皆様にはチームゼミだけでなく日常的にご意見を頂きました。ここに深く感謝いたします。最後に日頃より私を支えてくれました家族や友人たちに心より感謝いたします。

参考文献

[CI05] Alvaro Cassinelli and Masatoshi Ishikawa. Khronos projector. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Emerging technologies*, New York, NY, USA, August 2005. ACM Press.

[GDHZ06] Yotam I. Gingold, Philip L. Davidson, Jefferson Y. Han, and Denis Zorin. A direct texture placement and editing interface. In *UIST '06: Proceedings of the 19th annual ACM symposium on User interface software and technology*, pp. 23–32, New York, NY, USA, October 2006. ACM.

[GWB04] Tovi Grossman, Daniel Wigdor, and Ravin Balakrishnan. Multi-finger gestural interaction with 3d volumetric displays. In *UIST '04: Proceedings of the 17th annual ACM symposium on User interface software and technology*, pp. 61–70, New York, NY, USA, October 2004. ACM.

[Han05] Jefferson Y. Han. Low-cost multi-touch sensing through frustrated total internal reflection. In *UIST '05: Proceedings of the 18th annual ACM symposium on User interface software and technology*, pp. 115–118, New York, NY, USA, October 2005. ACM Press.

[HCC07] Mark Hancock, Sheelagh Carpendale, and Andy Cockburn. Shallow-depth 3d interaction: design and evaluation of one-, two- and three-touch techniques. In *CHI '07: Proceedings of the SIGCHI conference on Human factors in computing systems*, pp. 1147–1156, New York, NY, USA, April 2007. ACM.

[NMKT05] Kousuke Nakashima, Takashi Machida, Kiyoshi Kiyokawa, and Haruo Takemura. A 2d-3d integrated environment for cooperative work. In *VRST '05: Proceedings of the ACM symposium on Virtual reality software and technology*, pp. 16–22, New York, NY, USA, November 2005. ACM.

[SRF+06] Chia Shen, Kathy Ryall, Clifton Forlines, Alan Esenther, Frederic D. Vernier, Katherine Everitt, Mike Wu, Daniel Wigdor, Meredith R. Morris, Mark Hancock, and Edward Tse. Informing the design of direct-touch tabletops. *IEEE Computer Graphics and Applications*, Vol. 26, No. 5, pp. 36–46, September 2006.

[WB03] Mike Wu and Ravin Balakrishnan. Multi-finger and whole hand gestural interaction techniques for multi-user tabletop displays. In *UIST '03: Proceedings of the 16th annual ACM symposium on User interface software and technology*, pp. 193–202, New York, NY, USA, November 2003. ACM Press.

[WSR+06] Mike Wu, Chia Shen, Kathy Ryall, Clifton Forlines, and Ravin Balakrishnan. Gesture registration, relaxation, and reuse for multi-point direct-touch surfaces. In *TABLETOP '06: Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*, pp. 185–192, Washington, DC, USA, January 2006. IEEE Computer Society.

[柴田 06] 柴田樹, 落合香, 洲巻和也, 奥出直人. Biblioroll: 読書活動支援のためのポータブルデバイス. インタラクション 2006 論文集, pp. 85–86. 情報処理学会, March 2006.

[石井 03] 石井陽子, 中西泰人, 小池英樹, 岡兼司, 佐藤洋一. Enhancedmovie : 机型インタフェースを用いた動画編集システム. 第 11 回 日本ソフトウェア科学会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 (WISS 2003) 予稿集, pp. 41–46, December 2003.

[川西 02] 川西隆仁, 土田勝, 村瀬洋, 高木茂. 擬人化エージェントのための小型円筒ディスプレイとその応用. 電子情報通信学会技術研究報告. MVE, マルチメディア・仮想環境基礎, Vol. 102, No. 219, pp. 45–50, July 2002.

[藤村 06] 藤村憲之, 藤吉賢, 石田啓介, 西村拓一. テーブルトップコミュニティ〜コミュニティ支援のための実世界指向インターフェース〜. 電子情報通信学会技術研究報告. パターン認識・メディア理解, Vol. 105, No. 534, PRMU2005–181, pp. 189–194, January 2006.

[北原 06] 北原圭吾, 井上智雄, 重野寛, 岡田謙一. 協調学習支援を目的としたテーブルトップインタフェース (学習支援). 情報処理学会論文誌, Vol. 47, No. 11, pp. 3054–3062, November 2006.