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

物理コントローラとGUIの対応付けインタフェース

大江 龍人

修士（工学）

（コンピュータサイエンス専攻）

指導教員　田中 二郎

2013年3月

概要

マウスを用いたGUI操作は，複数要素の同時操作が不可能であり，また素早い操作には不向きである．そのために，いくつかのアプリケーションでは物理コントローラを用いた操作を提供している．物理コントローラを用いることにより，ユーザは素早く値を同時操作可能である．しかしながら，従来のインタフェースでは，任意のGUIに対して物理コントローラを使用することは出来なかった．本研究では，様々なGUIに対して物理コントローラを容易に対応付けることを目的とし，そのアプローチとして，物理コントローラとGUIの対応付けインタフェースを示す．本インタフェースを用いてユーザは，デスクトップ上に表示されるGUIをマウスを用いてなぞる様に描くことにより，GUIと物理コントローラとを対応付ける．対応付けされたGUIは以降物理コントローラを用いて操作可能になる．ユーザは対応付けるGUI要素に応じて，なぞる方法を変えることにより，様々なGUIに対して対応付けを行うことが可能である．本研究では物理コントローラとGUIの対応付けインタフェースを実装し，様々なアプリケーションに対して実際に本インタフェースを試用した．試用の結果，物理コントローラを用いて対応付けられたGUIをユーザは複数同時操作可能であることを確認した一方，様々なGUIに対して本インタフェースを適用するための課題が明らかになった．

目 次

第 1 章　序論 **1**
1.1　GUI を用いた操作 1
1.2　本研究の目的 3
1.3　本研究のアプローチ 3
1.4　本論文の構成 3

第 2 章　物理コントローラと GUI の対応付けインタフェース **4**
2.1　環境 4
2.2　対応付け手法 5
2.2.1　直線を用いた対応付け手法 5
2.2.2　自由曲線を用いた対応付け手法 5
2.2.3　矩形を用いた対応付け手法 7
2.2.4　円を用いた対応付け手法 8
2.2.5　複数 GUI 対応付け手法：スライダを囲む矩形を用いた対応付け手法 . 8
2.3　機能 11
2.3.1　対応付けの編集 11
2.3.2　対応付けの保存・復元 12
2.3.3　カーソル位置に応じた物理出力 13

第 3 章　実装 **14**
3.1　実装概要 14
3.2　スライダを囲む矩形を用いた対応付けの実装 16
3.2.1　スライダ群の認識 16
スライダの始点と終点の認識 16
類似スライダの認識 18

第 4 章　応用 **20**
4.1　画像編集アプリケーションのトーンカーブ調整 20
4.2　ドローイングアプリケーションの色調整 22
4.3　音楽アプリケーションのイコライザ調整 22

第 5 章　関連研究 **26**
5.1　GUI を用いた操作との対応付け 26

5.2 デスクトップ操作の拡張 27
5.3 デスクトップに対する画像処理 27
5.4 タンジブルユーザインタフェース 28

第 6 章 議論 **29**
6.1 対応付ける GUI に応じた物理コントローラの処理の変更 29
6.2 本研究にて使用される物理コントローラ 30

第 7 章 結論 **33**

謝辞 **34**

参考文献 **35**

図目次

1.1 現在の GUI を用いた操作の問題点．a) 複数同時操作の問題，b) ポイント&クリックの問題，c) ウィンドウ操作の問題． 1
1.2 物理コントローラ． 2

2.1 本インタフェースの利用環境．物理コントローラとしてスライダを用いる． . 4
2.2 物理コントローラと GUI の対応付けインタフェースを用いたインタラクションの流れ．a) 本インタフェースを立ち上げる，b) 対応付けする手法を選択する，c) CTRL キーを押しながらマウスを押す，d) マウスをドラッグする，e) 描いた箇所と物理コントローラが対応付けられる． 6
2.3 自由曲線を用いた対応付け手法．a) 対応付け手法の選択，b) CTRL キーを押しながら自由曲線の始点をプレスする，c) 自由曲線を描く． 7
2.4 矩形を用いた対応付け手法．a) 対応付け手法の選択，b) CTRL キーを押しながら矩形の始点をプレスする，c) 矩形を描く，d) 矩形の左上端が開始点として物理スライダの上端に割り当てられ右回りに動作する． 8
2.5 円を用いた対応付け手法．a) 対応付け手法の選択，b) CTRL キーを押しながら円の中心をプレスする，c) 円を描く，d) 円の最上点が開始点として物理スライダの上端に割り当てられ右回りに動作する． 9
2.6 スライダを囲む矩形を用いた対応付け手法．a) 対応付け手法の選択，b) CTRL キーを押しながらスライダを囲む矩形の左端をプレスする，c) マウスをドラッグさせスライダを囲む，d) 囲んだ矩形内のスライダに類似したスライダ群が認識される，e) 認識されたスライダ群が物理コントローラに対応付けられる． . 10
2.7 対応付けの編集機能．a) ボタンを押すと編集画面が立ち上がる，b) 編集画面には物理コントローラに対応付けられている GUI のサムネイルが表示される，c) 対応付けを解除するためにサムネイルを赤い領域外にドラッグ&ドロップする，d) 対応付けた物理コントローラの位置を変更するためにサムネイルの位置を変更する，e) 「CLOSE ボタン」を押すと編集結果が保存される． 11
2.8 対応付けの a) 保存と b) 復元．a-1) GUI と物理コントローラを対応付ける，a-2) 保存ファイル名を入力し “save” ボタンをクリックする，b-1) 復元ファイル名を入力する，b-2) “load” ボタンをクリックすると対応付けられたアプリケーションが復元される． 12
2.9 カーソル位置に応じた物理出力機能．a) GUI と物理コントローラを対応付ける，b) ユーザがカーソルを動かすと物理スライダがカーソル位置に応じて動く． 13

3.1 実装概要．a) 対応付け情報の保持，b) キーボードとマウスのフック，c) 物理コントローラの入力に応じたマウスイベントの発行，d) ウィンドウ情報の更新． 14
3.2 全体の処理の流れ．a) 矩形領域によって指定されたスライダ画像とアプリケーション画像を入力する，b) 入力されたスライダ画像と似ているスライダ群の線を認識する，c) 認識されたスライダの線群と物理スライダを対応付ける． . . 17
3.3 スライダの始点と終点の認識原理．画像処理によってスライダ両端の輝度値変化を取得し，スライダの始点と終点を認識する． 17
3.4 スライダの始点と終点の認識処理．a) 入力するスライダの元画像，b) グレースケール画像，c) 隣り合うピクセルの輝度値が変化する場合を 1 として 2 値化した画像，d) 認識結果の画像． 18
3.5 テンプレートマッチングを用いた類似スライダの認識結果． a) 入力画像，b) 入力画像に合致するスライダ群を認識する，c) b の出力にスライダの始点と終点を適用させる． 19

4.1 個々のスライダを用いてトーンカーブ調整の制御点を操作する例．a) 直線を用いた対応付けを複数回使用しトーンカーブの垂直線群とスライダ群とを対応付ける，b) スライダを動かすことによりトーンカーブの各制御点を動かすことが出来る． 21
4.2 個々のスライダをトーンカーブに割り当てた例． 21
4.3 ドローイングアプリケーションの色調整．a) 色相，彩度，明度，透明度等に物理コントローラを割り当てる．b) 手元のコントローラを用いて色を調整しドローイングを行う． 23
4.4 音楽アプリケーションのイコライザ調整．スライダを囲む矩形を用いた対応付け手法を用いて複数 GUI スライダと物理コントローラを対応付ける． 23
4.5 jackEQ のイコライザ調整では 1ch ごと縦方向にグルーピングされる． 24
4.6 jackEQ のスライダの挙動．クリックする毎に値が大まかに変化する． 24
4.7 本インタフェースを用いスライダの値を正確に制御するためには，a) つまみをプレスし，b) 入力された値までドラッグし，c) 入力終了後にリリースする必要がある． 25
4.8 スライダの値がそれぞれ異なる場合，スライダを囲む矩形を用いた対応付け手法を用いて同類スライダが認識されない． 25

6.1 GUI ノブの操作方法．ユーザがノブを操作する際ノブの円内をマウスプレスした後，円外の左又は右にドラッグし値を調整する． 29
6.2 対応付け時に GUI 要素を認識する．円を用いた対応付けの際はノブを認識し，直線を用いた対応付けの際はスライダを認識する． 29
6.3 対応付ける物理コントローラとしてタッチパネルを用いる例． 30
6.4 Villar らが示した VoodooIO（[VG07] より転載）． 30
6.5 Weiss らが示した Madgets（[WSJB10] より転載）． 31

6.6　対応付けの際にピクセルの輝度値変化を計算し，輝度値が変化した箇所を物理コントローラが触覚提示する．　. 31

第1章　序論

本章では，現在の GUI を用いた操作の問題点を述べ，それを受けた本研究の目的とアプローチを述べる．

1.1　GUI を用いた操作

現在のオフィスや家庭環境にて用いられる計算機のほとんどは Graphical User Interface（GUI）を採用している．GUI における操作では，ユーザはディスプレイ上に提示される視覚的な要素をポインタを用いて操作する．このポインタによる操作は，マウス，トラックパッド，及びトラックボール等を介して行われる．ユーザは提示される視覚要素をポインタを用いて選択することにより，容易に GUI を操作可能である．その一方，ユーザにとっては必然的に GUI を「選択し」操作する必要が生じる．

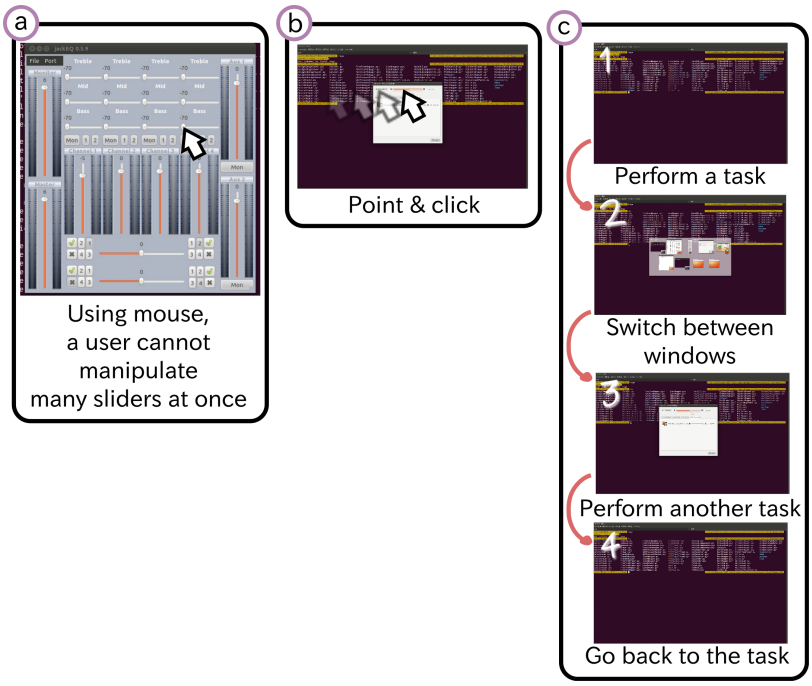

図 1.1: 現在の GUI を用いた操作の問題点．a) 複数同時操作の問題，b) ポイント&クリックの問題，c) ウィンドウ操作の問題．

ポインタを用いた GUI の選択には以下に挙げる問題点が存在する．

a. **複数同時操作が不可能である．** ポインタを用いて GUI をポインティングする場合 1 度にポインティング可能な GUI は 1 点のみであるため，複数同時操作が不可能である．例えば，図 1.1a の様に設定する値が複数項目に及ぶアプリケーションにおいて，ユーザは 1 度に 1 つずつポインタを用いて値を操作する必要があるため，複数の値を設定するためには項目分の時間を要する．

b. **ポイント&クリックに時間を要する．** ポインタを用いて GUI に働きかける場合，図 1.1b に示す様にユーザは GUI の位置までポインタを動かしてポイントし正確にクリックするための時間を要する．フィッツの法則 [Fit54] によると，このポインティングタスクの所要時間 T は式 1.1 として表され，対象までの距離が短く対象の幅が大きいほどポインティングの T は小さくなる．

$$T = a + b\log_2(1 + \frac{D}{W}) \tag{1.1}$$

ユーザは頻繁にポインティングのための時間を消費している．そのため，頻繁に行うポイント操作（例えば音量調節）において，ユーザはその回数分の時間を消費することになる．

c. **ウィンドウ操作に時間を要する．** 現在のデスクトップ環境はマルチウィンドウ環境であることが一般的であり，ユーザは複数ウィンドウを操作してタスクを行う．このウィンドウ操作は時間を要するものである．柴田 [柴田 09] によると，単一のディスプレイ環境下において，計算機を操作している時間のうち 8.5%をウィンドウの切替え，移動，サイズ変更のウィンドウ操作に費やしている．例えばユーザがあるタスクを実行中に（図 1.1c-1）別ウィンドウにある値を調節したくなった際，Alt+Tab を用いてウィンドウを選択し（図 1.1c-2），その後値を調節し（図 1.1c-3），以前のタスクに戻る（図 1.1c-4）．この様な作業は頻繁に起こりうる作業であり，ユーザはそのための時間を消費している．

幾つかのアプリケーションではポインタを用いた操作のみではなく，図 1.2 に示す様な物理コントローラを用いた操作を可能にしている．物理コントローラは，主に音楽アプリケー

図 1.2: 物理コントローラ.

ションにおいて使用されており，音量の調整，音色の調整，エフェクトのパラメータ調整等に用いられる．物理コントローラを用いることにより，ユーザはアプリケーション内におい

て複数の値を同時に素早く操作可能になるため，複数同時操作の問題とポイント&クリックの問題が解決される．しかしながら従来の環境では，この様な物理コントローラを使用できるアプリケーションは一部に限られており，ユーザは任意の GUI に対して物理コントローラを使用することは出来なかった．

1.2 本研究の目的

本研究では，様々な GUI に対して物理コントローラを容易に対応付けることを目的とする．

GUI に物理コントローラを対応付けることにより，ユーザは物理コントローラを用いて対応付けた GUI を操作可能になる．対応付けた結果，物理コントローラを用いて複数の GUI を同時に操作可能になるため，複数同時操作の問題が解決される．さらに物理コントローラを手元に配置し，頻繁に使用する操作を物理コントローラに割り当てることにより，ポインタを用いて GUI に働きかける必要が無くなるためポイント&クリックの問題が解決される．加えて，ポインティングするためのウィンドウを出現させる必要が無くなるため，ウィンドウ操作の問題が解決される．

1.3 本研究のアプローチ

本研究の目的を実現するために，本研究では「物理コントローラと GUI の対応付けインタフェース」を示す．本インタフェースを用いてユーザは，デスクトップ上に表示される GUI をマウスを用いてなぞる様に描くことにより，GUI と物理コントローラとを対応付ける．対応付けされた GUI は以降物理コントローラを用いて操作可能になる．ユーザは対応付ける GUI 要素に応じて，なぞる方法を変えることにより，様々な GUI に対して対応付けを行うことが可能である．

1.4 本論文の構成

本論文の構成を以下に示す．まず第 2 章にて物理コントローラと GUI の対応付けインタフェースを用いたインタラクション手法を示す．次に第 3 章にて本システムの実装について示し，さらに第 4 章にて本システムの応用を示し，そして第 5 章にて関連研究を示する．加えて第 6 章にて本システムについて議論す．最後に第 7 章にて本論文の結論を示す．

第2章　物理コントローラとGUIの対応付けインタフェース

本章では「物理コントローラとGUIの対応付けインタフェース」を述べる．まず本インタフェースを使用する環境を説明し，次に本インタフェースを用いたインタラクションの流れを示す．そして直線，自由曲線，矩形，円，スライダを囲む矩形を用いた対応付け手法をそれぞれ述べる．さらに，本システムの機能である「対応付けの編集」機能,「対応付けの保存・復元」機能，及び「マウス位置に応じた物理出力」機能とをそれぞれ述べる．

2.1　環境

本インタフェースは図2.1に示す様に物理コントローラとしてMIDIスライダを用いることを想定する．ユーザはMIDIスライダを通常のデスクトップ環境に備わるマウスとキーボードに追加する．このMIDIスライダを用いて対応付けられたGUIに働きかけユーザは操作を行う．

図2.1: 本インタフェースの利用環境．物理コントローラとしてスライダを用いる．

2.2　対応付け手法

本インタフェースは，物理コントローラとそれに対応付けるデスクトップ上の GUI をマウスを用いてなぞることにより，物理コントローラと GUI とを対応付けるインタフェースである．本インタフェースを用いたインタラクションの流れを以下に示す.

a. 本インタフェースを立ち上げる．
b. 対応付けする手法を選択する．図 2.2b では，ユーザは第 2.2.1 節にて示す「直線を用いた対応付け手法」を選択している．
c. 対応付けされた際の始点を指定するために，CTRL キーを押しながらマウスをプレスする．図 2.2c では，ユーザは音量スライダと物理コントローラを対応付けるために，音量スライダの左端をプレスしている．
d. 対応付けされた際の終点を指定するために，対応付ける箇所までマウスをドラッグし，その後リリースする．図 2.2d では，ユーザは音量スライダの右端にてマウスをリリースしている．
e. マウスを用いて描かれた箇所と物理コントローラが対応付けられる．

以上の様に GUI と物理コントローラとを対応付けることにより，ユーザは以降物理コントローラを用いて GUI を操作することが可能となる．以降，本インタフェースの対応付け手法をそれぞれ述べる．

2.2.1　直線を用いた対応付け手法

直線を用いた対応付け手法は，マウスを用い，対応付け対象の GUI 上に直線を描くことにより，直線と物理コントローラの操作とを対応付ける手法である．この手法を用いたインタラクションの流れは，図 2.2b~ 図 2.2e に示される．対応付け後，直線の始点がスライダの上端，直線の終点がスライダの下端にそれぞれ割り当てられる．本手法は GUI 上に描かれた直線と物理スライダとを割り当てる単純な手法であり，GUI スライダの様な 1 次元の GUI と物理スライダを対応付ける際に有効である．

2.2.2　自由曲線を用いた対応付け手法

自由曲線を用いた対応付け手法は，マウスを用い，対応付け対象の GUI 上に自由曲線を描くことにより，曲線と物理コントローラとを対応付ける手法である．

図 2.3 を用いて，この手法のインタラクションの流れを以下に示す.

a. ユーザは対応付け手法として，自由曲線を用いた対応付けを選択する．
b. CTRL キーを押しながら自由曲線の始点をプレスする．図 2.3b では，ユーザは描画アプリケーションの彩度・明度の調整を物理コントローラに割り当てようとしている．
c. マウスをドラッグし自由曲線を描き，その後マウスをリリースする．描いた曲線の始点から終点までが物理コントローラに割り当てられる．

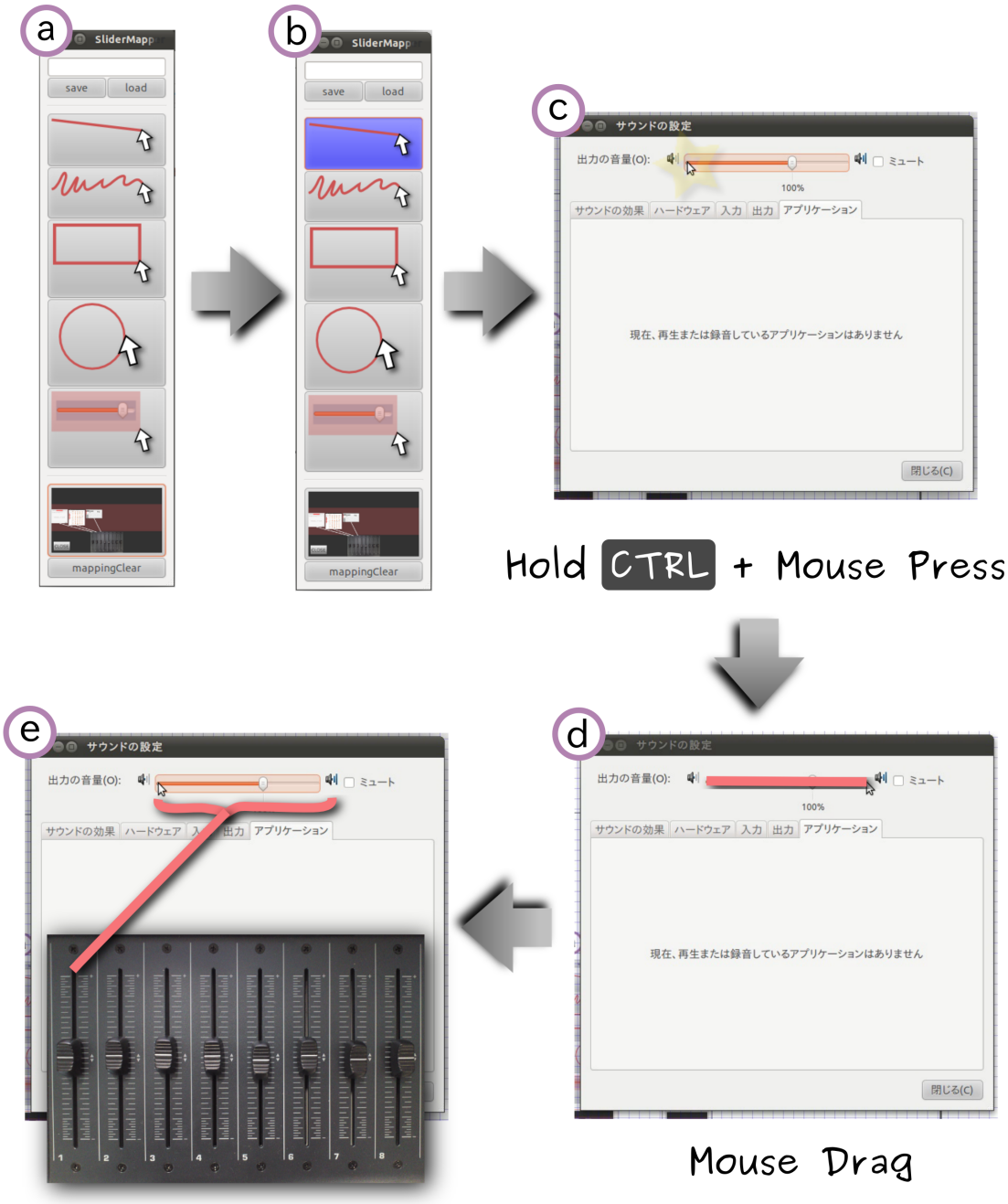

図 2.2: 物理コントローラと GUI の対応付けインタフェースを用いたインタラクションの流れ. a) 本インタフェースを立ち上げる，b) 対応付けする手法を選択する，c) CTRL キーを押しながらマウスを押す，d) マウスをドラッグする，e) 描いた箇所と物理コントローラが対応付けられる.

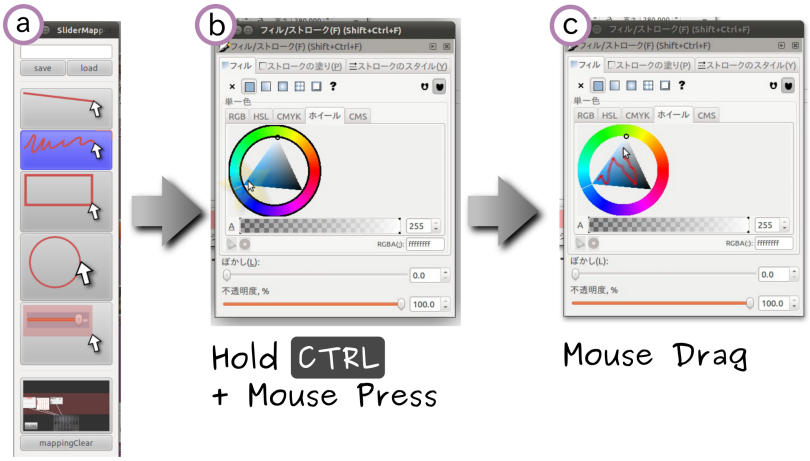

図 2.3: 自由曲線を用いた対応付け手法．a) 対応付け手法の選択，b) CTRL キーを押しながら自由曲線の始点をプレスする，c) 自由曲線を描く．

対応付け後，自由曲線の始点がスライダの上端，自由曲線の終点がスライダの下端にそれぞれ割り当てられる．本手法を用いることにより，ユーザは自由に描いた線と物理スライダとを対応付けることが可能となり，カーブやジグザグの様な GUI に対する操作と物理スライダの操作を対応付けることが可能になる．

2.2.3 矩形を用いた対応付け手法

矩形を用いた対応付け手法は，マウスを用い，対応付け対象の GUI 上に矩形を描くことにより，矩形と物理コントローラの操作とを対応付ける手法である．

図 2.4 を用いて，この手法のインタラクションの流れを以下に示す．

a. ユーザは対応付け手法として，矩形を用いた対応付けを選択する．
b. CTRL キーを押しながら矩形の始点をプレスする．図 2.4b では，ユーザはプレゼンテーションアプリケーションのスタイル変更を物理コントローラに割り当てようとしている．
c. マウスをドラッグし矩形を描き，その後マウスをリリースする．
d. 図 2.4d の「start」が物理スライダの上端，図 2.4d の「end」が物理スライダの下端に割り当てられる．ユーザが物理スライダを上から下に操作すると，矩形に沿って右回りに GUI が操作される．

本手法は GUI 上に描かれた矩形と物理スライダとを割り当てる手法であり，図 2.4b の様な GUI 要素群が 2 列に並んだメニューと物理スライダとを対応付ける際に有効である．

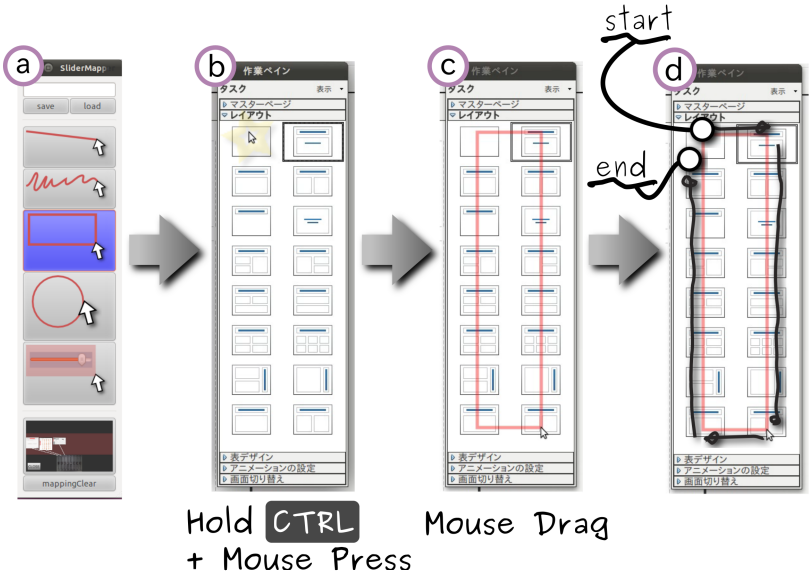

図 2.4: 矩形を用いた対応付け手法．a) 対応付け手法の選択，b) CTRL キーを押しながら矩形の始点をプレスする，c) 矩形を描く，d) 矩形の左上端が開始点として物理スライダの上端に割り当てられ右回りに動作する．

2.2.4　円を用いた対応付け手法

円を用いた対応付け手法は，マウスを用い，対応付け対象の GUI 上に円を描くことにより，円と物理コントローラの操作とを対応付ける手法である．

図 2.5 を用いて，この手法のインタラクションの流れを以下に示す．

a. ユーザは対応付け手法として，円を用いた対応付けを選択する．
b. CTRL キーを押しながら円の中心をプレスする．図 2.5b では，ユーザは描画アプリケーションの色相を物理コントローラに割り当てようとしている．
c. マウスをドラッグし円を描き，その後マウスをリリースする．
d. 円の最上点が開始点として物理スライダの上端に割り当てられる．ユーザが物理スライダを上から下に操作すると，円周に沿って右回りに GUI が操作される．物理スライダを下端まで操作すると円の最上点に戻る．

本手法は GUI 上に描かれた円と物理スライダとを割り当てる手法であり，図 2.4b の様な円形の GUI と物理スライダとを対応付ける際に有効である．

2.2.5　複数 GUI 対応付け手法：スライダを囲む矩形を用いた対応付け手法

スライダを囲む矩形を用いた対応付け手法は，複数の GUI スライダ群と手元の物理スライダ群とを一度に対応付ける手法である．ユーザは，対応付ける GUI 上のスライダをマウスを用いて囲むことにより，囲んだスライダと類似するスライダ群を対応付けることが出来る．対

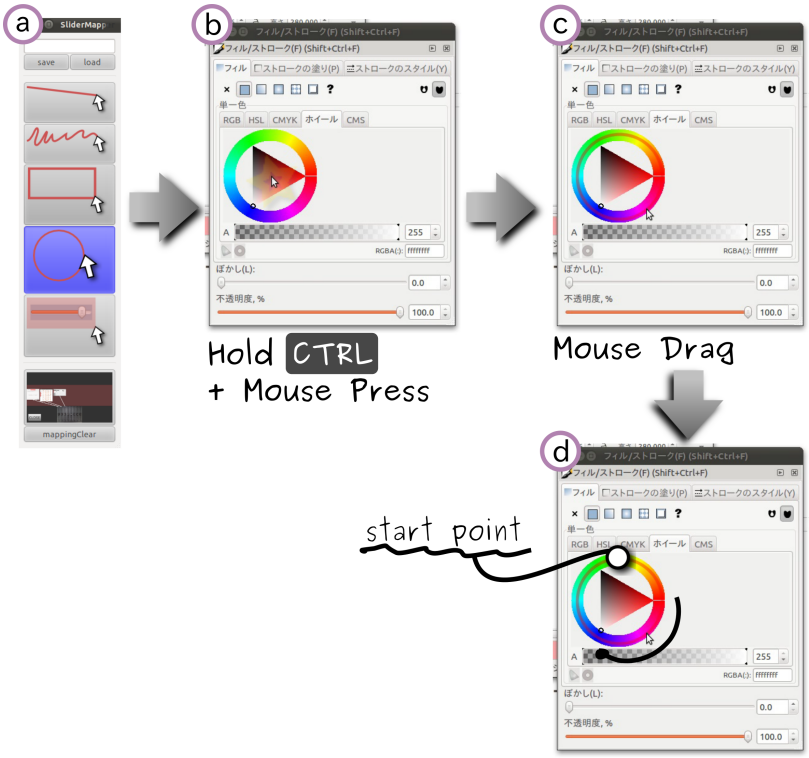

図 2.5: 円を用いた対応付け手法．a) 対応付け手法の選択，b) CTRL キーを押しながら円の中心をプレスする，c) 円を描く，d) 円の最上点が開始点として物理スライダの上端に割り当てられ右回りに動作する．

応付けられた個々の GUI には，直線を用いた対応付けがなされており，ユーザは物理スライダを用いて GUI のスライダを操作する．

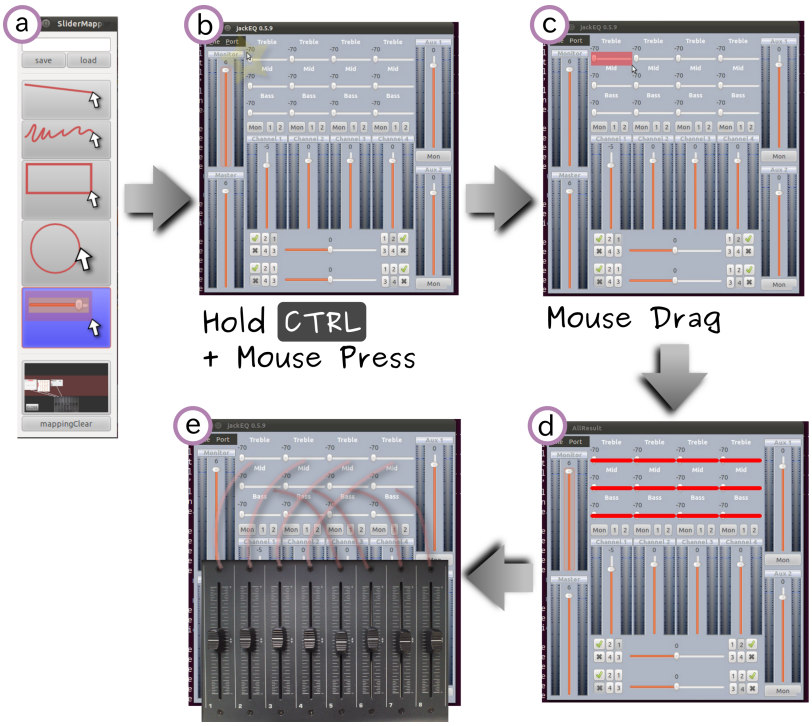

図 2.6: スライダを囲む矩形を用いた対応付け手法．a) 対応付け手法の選択，b) CTRL キーを押しながらスライダを囲む矩形の左端をプレスする，c) マウスをドラッグさせスライダを囲む，d) 囲んだ矩形内のスライダに類似したスライダ群が認識される，e) 認識されたスライダ群が物理コントローラに対応付けられる．

図 2.6 を用いて，この手法のインタラクションの流れを以下に示す．

a. ユーザは対応付け手法として，スライダを囲む矩形を用いた対応付けを選択する．
b. CTRL キーを押しながらスライダを囲む矩形の左端をプレスする．図 2.6b では，ユーザは音楽アプリケーションのスライダ群を物理コントローラに割り当てようとしている．
c. マウスをドラッグさせスライダを囲む矩形を描き，その後マウスをリリースする．
d. マウスをリリースした後，矩形によって囲まれたスライダとそれに類似するスライダ群が認識される．
e. 認識されたスライダ群と物理スライダ群が自動的に対応付けられる．この際，対応付けは認識されたスライダ群の左上から横方向に順になされる．

本対応付けを用いることにより，ユーザは GUI スライダが複数存在した場合に，それぞれの要素に対して直線を用い対応付ける必要が無くなる．そのため，要素が複数存在する際の対応付けに本手法は有効である．

2.3 機能

本節では，本インタフェースの機能である「対応付けの編集」機能,「対応付けの保存・復元」機能,「マウス位置に応じた物理出力」機能をそれぞれ述べる．

2.3.1 対応付けの編集

対応付けの編集機能は，GUI と物理コントローラとの対応付けを編集するための機能である．ユーザは本機能を用いることにより，対応付けの解除及び，対応付けの位置変更が可能である．

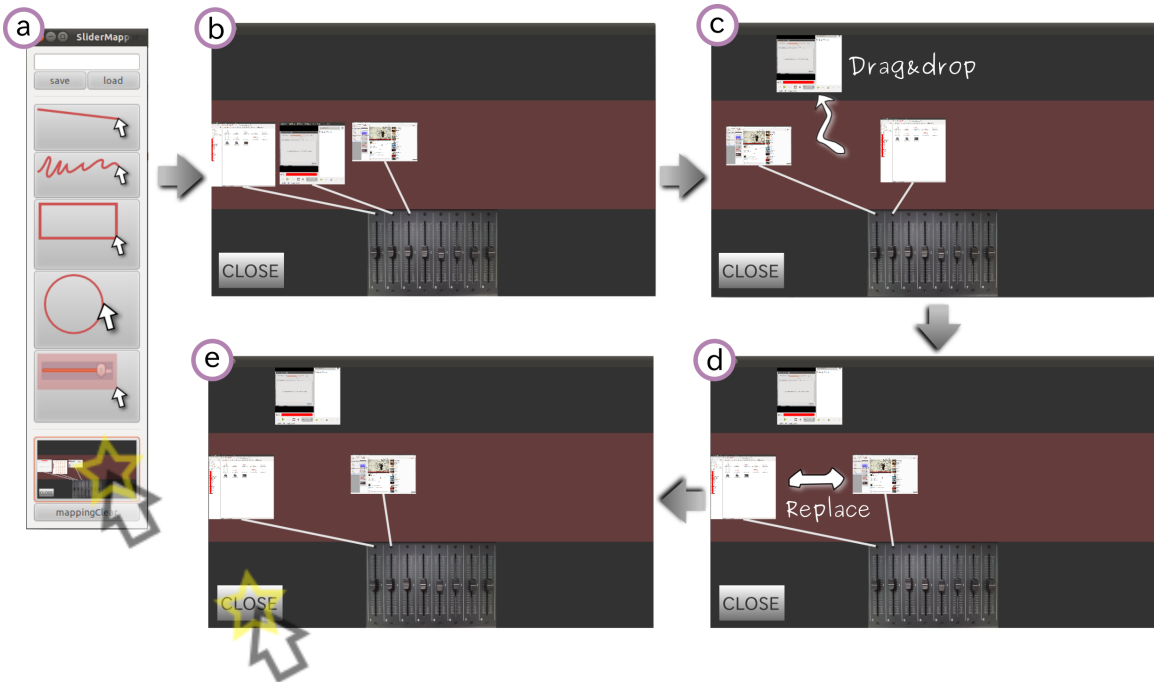

図 2.7: 対応付けの編集機能．a) ボタンを押すと編集画面が立ち上がる，b) 編集画面には物理コントローラに対応付けられている GUI のサムネイルが表示される，c) 対応付けを解除するためにサムネイルを赤い領域外にドラッグ&ドロップする，d) 対応付けた物理コントローラの位置を変更するためにサムネイルの位置を変更する，e)「CLOSE ボタン」を押すと編集結果が保存される．

図 2.7 を用いて，本機能の操作例を説明する．

a. 対応付けの編集機能を立ち上げるために，ユーザは図 2.7a のマウス位置に示されるボタンを押す．
b. 立ち上がった編集画面には物理コントローラに対応付けられている GUI のサムネイルが表示される．GUI のサムネイルとは，対応付けられた GUI のアプリケーションのス

クリーンショットに，対応付けの際に描かれた線が重畳表示されたものである．サムネイルと物理コントローラとの結線により，現在の対応付けの状態が表現される．

c. 対応付けを解除するためには，サムネイルを編集画面の赤い領域外にドラッグ&ドロップする．

d. 対応付けた物理コントローラの位置を変更するためには，マウスを用いてサムネイルの位置を変更する．サムネイルの位置を変更すると，編集画面の赤い領域内にあるサムネイルが左から順に物理コントローラに対応付けられる．

e. 「CLOSE ボタン」を押すと編集結果が保存される．

2.3.2 対応付けの保存・復元

対応付けの保存・復元機能は，GUI と物理コントローラとの対応付けを保存及び復元する機能である．ユーザは本機能を用いることにより，ユーザは頻繁に使用する対応付けをプリセットとして保存することが可能である．さらに，保存後は復元機能を用いることにより以前の対応付け操作を再利用することが可能となる．

図 2.8: 対応付けの a) 保存と b) 復元．a-1) GUI と物理コントローラを対応付ける，a-2) 保存ファイル名を入力し “save” ボタンをクリックする，b-1) 復元ファイル名を入力する，b-2) “load” ボタンをクリックすると対応付けられたアプリケーションが復元される．

対応付けの保存機能の使用例を以下に示す．

a. GUI と物理コントローラとを対応付ける．図 2.8a-1 では音量スライダとの対応付けを行っている．

b. 保存ファイル名を入力し “save” ボタンをクリックする．図 2.8a-2 では “volumeChange” とファイル名を名づけ，対応付けを保存している．

また，対応付けの復元機能の使用例は以下となる．

a. 保存したファイル名を入力する．図 2.8b-1 では，先に保存した “volumeChange” を入力している．

b. “load” ボタンをクリックすると，保存時に対応付けられたアプリケーションが保存した際と同じ位置に起動される．起動したアプリケーションは物理コントローラとの対応付けがなされており，ユーザは手元のコントローラを用いて GUI を操作することが出来る．

2.3.3 カーソル位置に応じた物理出力

カーソル位置に応じた物理出力機能は，対応付けられた GUI 上にカーソルをホバーさせた際に，そのカーソル位置に応じて物理コントローラが物理的に移動する機能である．本機能を用いることによりユーザは物理コントローラの動きを通して，対応付けの状態を知覚することが可能である．

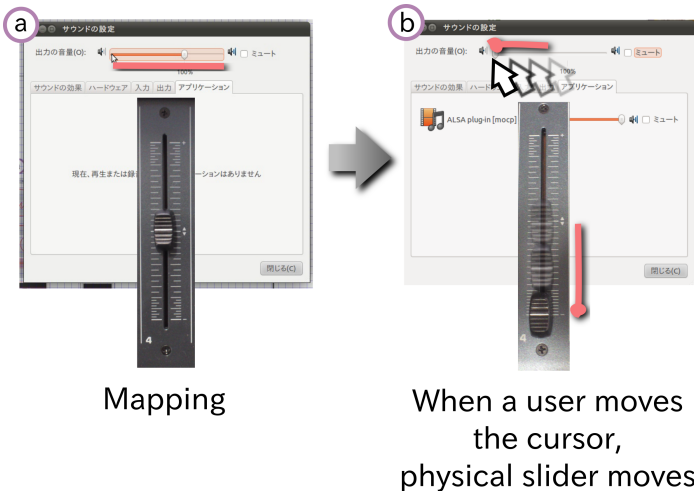

図 2.9: カーソル位置に応じた物理出力機能．a) GUI と物理コントローラを対応付ける，b) ユーザがカーソルを動かすと物理スライダがカーソル位置に応じて動く．

図 2.9 を用いて本機能を具体的に説明する．

a. ユーザが本インタフェースを用いて GUI と物理コントローラを対応付ける．

b. 対応付けられた GUI 上にユーザがカーソルをホバーすると，そのカーソル位置に応じて物理スライダが動く．

第3章　実装

本章では「物理コントローラと GUI の対応付けインタフェース」の実装を示す．最初に本システムの実装概要を示し，次に「スライダを囲む矩形を用いた対応付け」の実装を示す．

3.1　実装概要

本システムの実装の全体像は図 3.1 に示すものとなる．今回，実装を Ubuntu Linux[1] 上にてプ

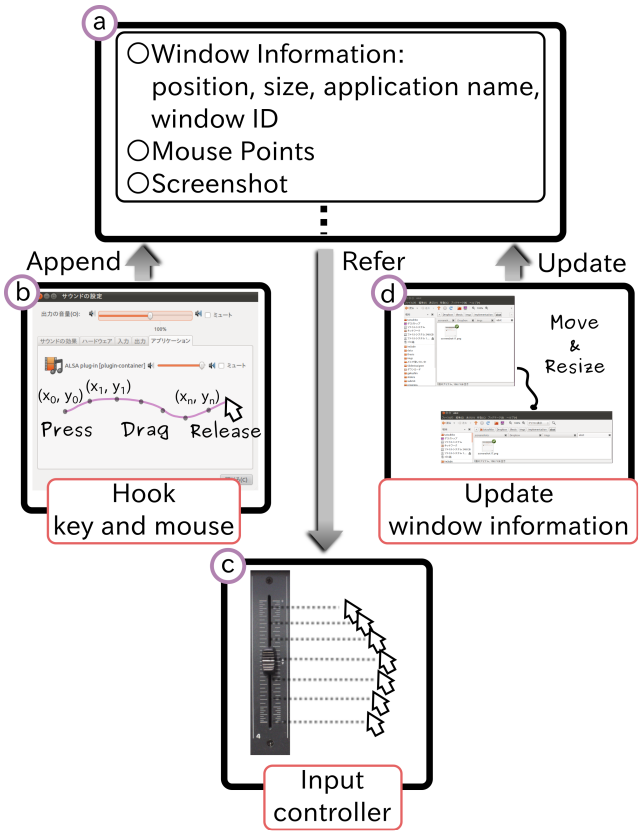

図 3.1: 実装概要．a) 対応付け情報の保持，b) キーボードとマウスのフック，c) 物理コントローラの入力に応じたマウスイベントの発行，d) ウィンドウ情報の更新．

ログラミング言語Python[2]を用いて行い，物理コントローラにはBEHRINGER社のBCF2000[3]を用いた．BCF2000を用いた理由は，8個のスライダがそれぞれ物理的に動くモータフェーダを搭載しており，第2章にて述べた「カーソル位置に応じた物理出力」を容易に実現可能であるためである．

本実装は，対応付け情報の保持，キーボードとマウスのフック，物理コントローラの入力に応じたマウスイベントの発行，ウィンドウ情報の更新から構成される．以下それぞれの処理内容を具体的に説明する．

対応付け情報の保持． 対応付け情報の保持では，CTRL+マウスイベントにより対応付けが行われた際に，対応付けられたGUIの情報をリスト形式にて保持する．対応付け情報は以下から構成される．

対応付けられたGUIアプリケーションのウィンドウ情報． ウィンドウ情報は，ウィンドウの位置，大きさ，名前，ウィンドウIDから構成される．ウィンドウIDとはX Window System上にて管理されるウィンドウの固有番号である．今回ウィンドウ情報の取得にはX Window Systemのコマンドである「xwininfo」と「xprop」を用いた．

ウィンドウ位置からの相対マウス座標群． マウス座標群は対応付けの際にユーザによって描かれた軌跡の座標群であり，ウィンドウ位置からの相対座標が保持される．

アプリケーションのスクリーンショット． アプリケーションのスクリーンショットは，対応付けされたGUIのアプリケーションのスクリーンショットである．このスクリーンショットは対応付け編集時に表示される画像として用いられ，また「スライダを囲む矩形を用いた対応付け」の入力画像としても用いられる．

なお，対応付け情報はJSON[4]形式にてシステム内に保持される．ユーザが「保存」や「復元」を行った際は，このJSONファイルが保存・復元されることにより，対応付け情報が保存・復元される．

キーボードとマウスのフック． キーボードとマウスのフックでは，デスクトップ環境に対するユーザのキーボード入力とマウス入力をグローバルに常に監視する．ユーザがCTRLキーを押しながらマウスイベントを発行させた際には，対応付け情報の末尾に値を追加する．

物理コントローラの入力に応じたマウスイベントの発行． 物理コントローラの入力に応じたマウスイベントの発行では，物理コントローラの入力を監視し，その入力に応じてマウスイベントを発行することによりGUIが操作される．発行されるマウスイベントは，デスクトップ上の指定された位置へのマウスクリックであり，その位置とは図3.1aにて保持されるウィンドウ位置とウィンドウ位置からの相対マウス座標とを足しあわせた位置となる．スライダが動かされた際にクリックイベントを連続して発行することによりGUIへの連続値を入力する． なお，複数のスライダを動かした際に，クリックイベン

[1] http://www.ubuntu.com/
[2] http://www.python.org/
[3] http://www.behringer.com/EN/Products/BCF2000.aspx
[4] http://www.json.org/

トは時分割により異なる位置に発行される．

ウィンドウ情報の更新． ウィンドウ情報の更新では，図 3.1a にて保持されるウィンドウ情報の位置と大きさを定期的に更新する．ウィンドウ情報の更新処理を行うことにより，対応付けられた GUI のウィンドウの位置が変更された場合においても，図 3.1c にて適切な GUI 位置へマウスイベントを発行することが可能となる．ウィンドウの大きさが変更された場合は，対応付けられた GUI が隠れて見えなくなってしまう場合がある．現在の実装ではこの場合に対応しておらず，隠れた GUI は物理コントローラを用いて操作することが出来ない．

3.2 スライダを囲む矩形を用いた対応付けの実装

スライダを囲む矩形を用いた対応付けは，複数の GUI スライダ群と手元の物理スライダ群とを一度に対応付ける手法である．この対応付け手法は，ユーザによって囲まれたスライダ画像とアプリケーション画像とを入力画像として，それらを用いた画像処理によって実現している．

図 3.2 を用いて処理の流れを説明する．

a. 矩形領域によって指定されたスライダ画像とそのアプリケーション画像を入力する．
b. 入力されたスライダ画像と類似するスライダ群のピクセル位置を認識する．
c. 認識されたスライダ群と物理スライダを対応付ける．物理スライダを動かした際に，対応付けられたスライダのピクセル位置をマウスクリックすることにより，スライダ操作が行われる．

以降，図 3.2b の「スライダ群の認識」について具体的に述べる．

3.2.1 スライダ群の認識

スライダ群の認識は，スライダ画像内の輝度値変化を用いた「スライダの始点と終点の認識」とテンプレートマッチングを用いた「類似スライダの認識」から構成される．

スライダの始点と終点の認識

「スライダの始点と終点の認識」の原理を，図 3.3 を用いて以下に説明する．

a. ユーザによって矩形選択されたスライダ画像の大きさを調べる．スライダ画像の幅が高さよりも大きい場合は横スライダ，それ以外は縦スライダであると識別する．
b. 横スライダの場合，矩形領域の高さを半分として，その直線上のピクセルの輝度値変化を走査する．縦スライダの場合は，幅を半分として直線上の輝度値変化を走査する．なお，矩形領域を半分として走査する理由は，ユーザがスライダを囲む際は，スライダを中心として上下左右均等に囲むと推測したためである．

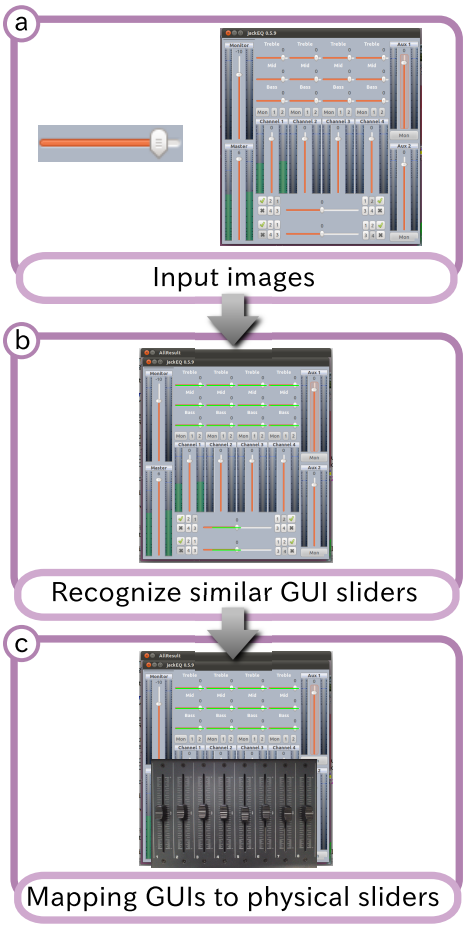

図 3.2: 全体の処理の流れ．a) 矩形領域によって指定されたスライダ画像とアプリケーション画像を入力する，b) 入力されたスライダ画像と似ているスライダ群の線を認識する，c) 認識されたスライダの線群と物理スライダを対応付ける．

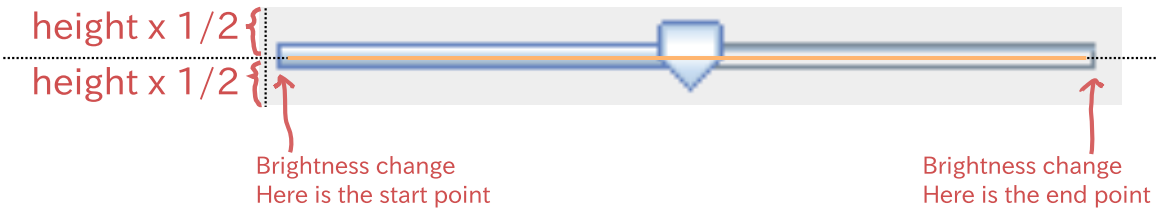

図 3.3: スライダの始点と終点の認識原理．画像処理によってスライダ両端の輝度値変化を取得し，スライダの始点と終点を認識する．

c. 直線上の輝度値変化を走査し，輝度値が変化する両端をスライダの始点と終点として認識する．

実際に実行させたスライダの始点と終点の認識処理を図 3.4 に示す．実際の処理は以下から構成される．

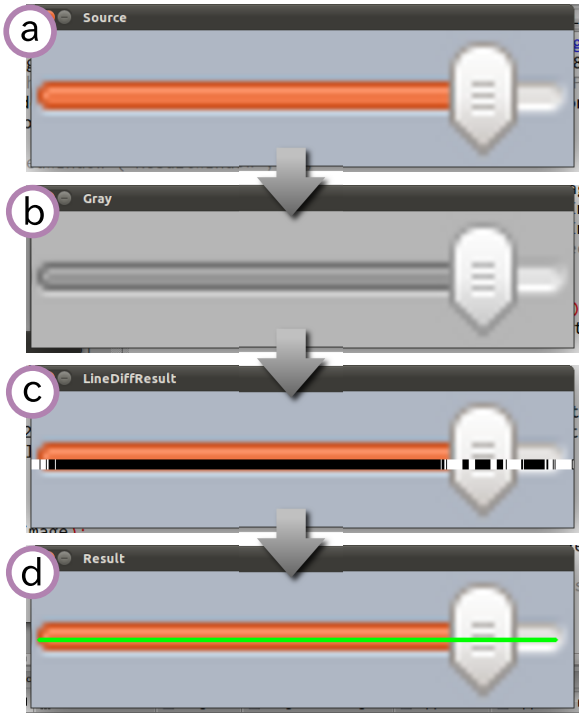

図 3.4: スライダの始点と終点の認識処理．a) 入力するスライダの元画像，b) グレースケール画像，c) 隣り合うピクセルの輝度値が変化する場合を 1 として 2 値化した画像，d) 認識結果の画像．

a. 認識させるスライダの元画像を入力する．
b. スライダの元画像をグレースケールに変換する．
c. 直線上の隣り合うピクセルをそれぞれ比較し，輝度値に変化が有る場合は 1，変化がない場合は 0 として 2 値化する．
d. 図 3.4c にて取得された 2 値化された直線において，1 が数ピクセル続いた場合の両端をスライダの始点と終点に設定する．

類似スライダの認識

類似スライダの認識は，スライダの画像をテンプレート画像として，アプリケーション画像に対するテンプレートマッチングを用いて行う．テンプレートマッチングを用いて類似スライダを認識させた結果を図 3.5 に示す．図 3.5b の赤い四角によって囲まれた領域は類似す

るスライダ群を表し，また，図 3.5c は図 3.5a の結果に「スライダの始点と終点の認識」が適用された結果を示す．

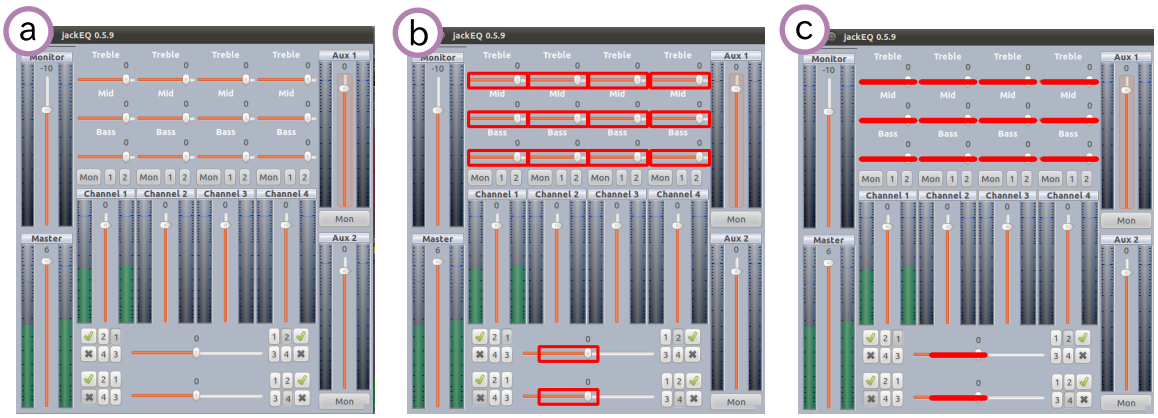

図 3.5: テンプレートマッチングを用いた類似スライダの認識結果． a) 入力画像，b) 入力画像に合致するスライダ群を認識する，c) b の出力にスライダの始点と終点を適用させる．

第4章　応用

本章では本インタフェースの応用を示し，それぞれの応用を試用した際の利点及び問題点を述べる．まず画像編集アプリケーションのトーンカーブ調整に適用させた事例を述べ，次にドローイングアプリケーションの色調整に適用させた事例，さらに音楽アプリケーションのイコライザ調整に適用させた事例を述べる．

4.1　画像編集アプリケーションのトーンカーブ調整

トーンカーブ調整は画像編集アプリケーションにおいて，画像の全体的な明るさやコントラストを調整する際に用いられる機能であり，ユーザは通常マウスを用いて複数点の値を入力してカーブを調整する．トーンカーブ調整機能に物理コントローラを対応付けることにより，手元のコントローラを用い複数の値を制御しながらカーブを調整することが可能になる．

今回，画像編集アプリケーションとして GIMP[1] を用い，本インタフェースをトーンカーブに適用させ，個々のスライダを用いてトーンカーブ調整の制御点を操作した．個々のスライダを用いてトーンカーブを調整する例を図 4.1 に示す．ユーザは図 4.1a に示す様に直線を用いた対応付けを複数回使用しトーンカーブの垂直線群とスライダ群とを対応付け，図 4.1b に示す様に各スライダを動かすことによりトーンカーブの各制御点を動かす．また，本インタフェースを用いて，個々のスライダを各カーブに割り当てた例を図 4.2 に示す．図 4.2 ではユーザは個々のスライダに明るめ，普通，暗め，ハイコントラストのカーブを対応付け，それぞれのスライダを動かすことによりトーンカーブを調整する．

トーンカーブ調整に適用させ実際に使用した際の利点を以下に示す．

トーンカーブの UI を見ずに対象を注視した編集が可能． マウスを用いてトーンカーブの UI を操作する際，ユーザは制御点をクリックし操作しながら編集対象の画像を閲覧する必要があるため，編集対象を注視した編集が困難である．それに対し個々のスライダを用いてトーンカーブを調整する場合，ユーザは手元の物理コントローラとトーンカーブ UI を見ずに操作可能であるため，編集対象を注視した編集が可能であった．

複数の制御点が操作可能． マウスを用いトーンカーブの UI を操作する際，ユーザは一つ一つの制御点を操作しカーブを調整する必要があり，複数制御点の同時操作が不可能である．本インタフェースを用いることにより，ユーザは複数の制御点を同時に操作することが可能になるため，例えば明るい部分を上げつつ暗い部分を下げる制御が可能であった．

[1] http://www.gimp.org/

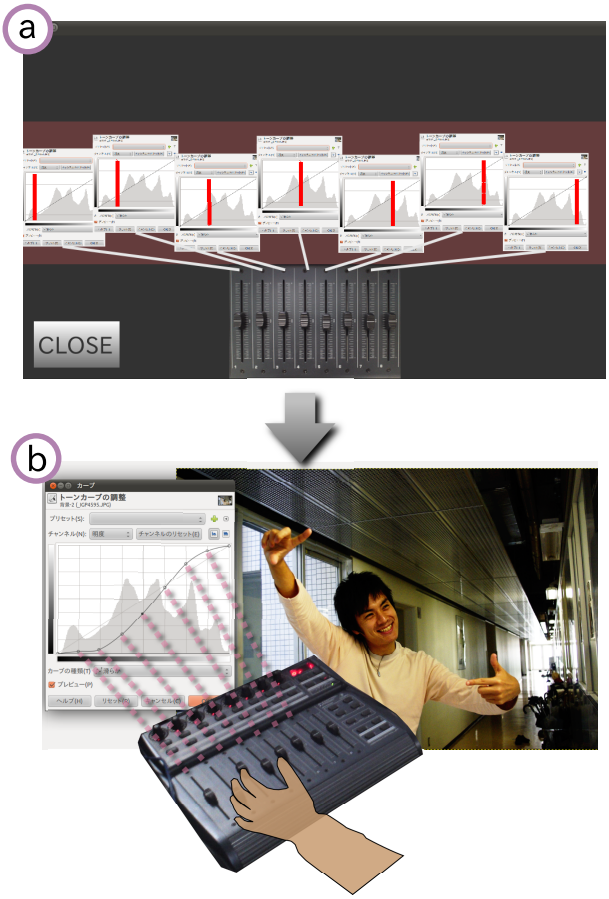

図 4.1: 個々のスライダを用いてトーンカーブ調整の制御点を操作する例．a) 直線を用いた対応付けを複数回使用しトーンカーブの垂直線群とスライダ群とを対応付ける，b) スライダを動かすことによりトーンカーブの各制御点を動かすことが出来る．

図 4.2: 個々のスライダをトーンカーブに割り当てた例．

以上の利点がある一方，以下の問題点が明らかになった．

複数点の制御時に処理が遅くなる． 本インタフェースの実装上の問題により，複数点の制御時に処理が遅くなることがあった．本インタフェースではスライダを動かすたびにクリックイベントを発行する実装となっており，複数スライダを同時に動かした際に，トーンカーブ UI に多数のクリックイベントが発行される．多数のクリックイベントがトーンカーブ UI に発行される場合，制御点が都度操作され画像処理が反映・表示されるため処理が重くなった．

4.2 ドローイングアプリケーションの色調整

ドローイングアプリケーションの色調整機能に物理コントローラを割り当て，手元のコントローラを用い色相，彩度，明度，透明度等を操作し描画を行った．今回，ドローイングアプリケーションとして Inkscape[2]を用い，スライダ群を色調整機能に割り当てた．

色調整に適用させ実際に使用した際の利点を以下に示す．

色のパラメータの同時変更が可能． マウスを用いて色を変更する場合，ユーザはパラメータを一つずつ変更する．それに対しスライダ群を用いて色を変更する場合，複数のパラメータを同時に変更することが可能であった．例えば，色相を変更しながら透明度を変更することが可能であった．

以上の利点がある一方，以下の要求点が明らかになった．

色を変更しながらの描画操作． 物理コントローラを用いて色を変更する際に，色の変更と同時に描画操作を行いたいと思う場合があった．これは例えば，複数の描画オブジェクトが存在した場合に，ある描画オブジェクトの透明度を変更しながら，その描画オブジェクトを移動させたい場合である．現在の実装では，物理コントローラを操作した際，対応付けられた GUI にクリックイベントを発行させるために 1 つのマウスポインタを占有する．そのため本要求を満たすためには，中村ら [中村 00] が示した DoubleMouse の様に複数のマウスポインタを用意し，物理コントローラ操作のためのポインタとマウス操作のためのポインタをユーザに提供する必要がある．

4.3 音楽アプリケーションのイコライザ調整

音楽アプリケーションのイコライザ調整機能に物理コントローラを割り当て，手元のコントローラを用い音色を調整した．今回音楽アプリケーションとして jackEQ[3]を採用し，図 4.4 に示す様に，スライダを囲む矩形を用いた対応付け手法によりイコライザ調整スライダと物理コントローラを対応付ける．その後，物理コントローラを用いてイコライザの Treble，Middle，Bass を操作した．

実際に使用した所，以下の問題点が明らかになった．

[2] http://inkscape.org/
[3] http://djcj.org/jackeq/

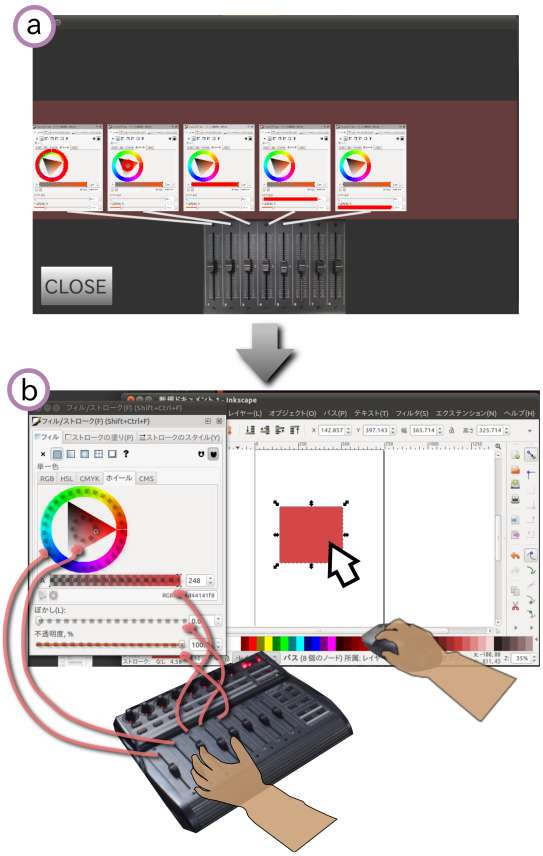

図 4.3: ドローイングアプリケーションの色調整．a) 色相，彩度，明度，透明度等に物理コントローラを割り当てる．b) 手元のコントローラを用いて色を調整しドローイングを行う．

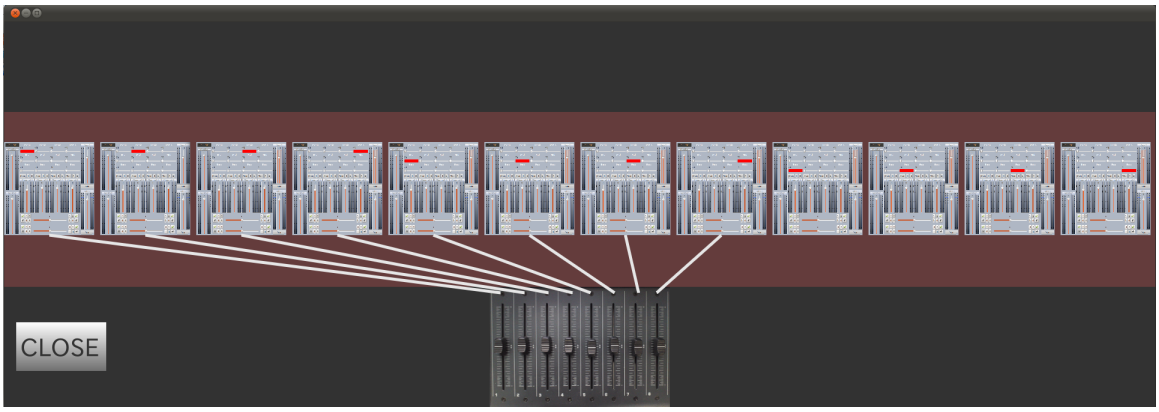

図 4.4: 音楽アプリケーションのイコライザ調整．スライダを囲む矩形を用いた対応付け手法を用いて複数 GUI スライダと物理コントローラを対応付ける．

対応付けの位置変更は手間がかかる． 今回の応用にて対応付けられた GUI は，図 4.4 に示す様にアプリケーションのスライダの上から順に横方向に対応付けられた．しかしながら，実際に jackEQ を使用する際は図 4.5 に示す様に 1ch ごと縦方向に使用するものであった．実際に使用する際は ch ごとに物理スライダをグルーピングして割り当てた方が使用しやすいため，対応付けの位置を編集する必要があった．この位置変更は手間がかかるものであった．この問題を回避するために，対応付けの際に，マウスの移動方向を用いて対応付けされる方向を指定可能にするインタラクションを考える．例えば，矩形を用いてスライダを選択した後，マウスが上下方向に移動した場合は縦方向に対応付けられ，マウスが左右方向に移動した場合は横方向に対応付けられる．この様なマウスの動作を入力として用いることにより，複数対応付けの際にそれぞれの対応付け方法をユーザがより詳細に記述出来るようにする．

正確な値の制御が不可能． jackEQ は GUI のフレームワークとして GTK[4]を採用している．GTK のスライダは図 4.6 に示す様に，スライダ上の位置をクリックする毎に，その位置に向かって値が大まかに変化する[5]．したがって，デスクトップ上の位置にクリックイベントを発行する本インタフェースの実装では，値が大まかに変化するのみであり正確な値の制御は出来なかった．本インタフェースを用いて，GUI スライダの値を正確に制御可能にするためには，物理コントローラを操作した際にスライダのつまみを操作する実装にする必要がある．それを実現するためには，まずシステムがスライダのつまみの位置を認識する．さらに物理コントローラが動かされた際には図 4.7 に示す様に，つまみをプレスし（図 4.7a），入力された値までドラッグし（図 4.7b），入力終了後にマウスをリリースする（図 4.7c）実装に変更する必要がある．

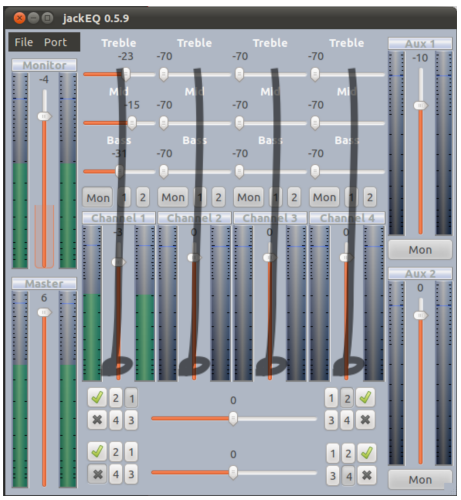

図 4.5: jackEQ のイコライザ調整では 1ch ごと縦方向にグルーピングされる．

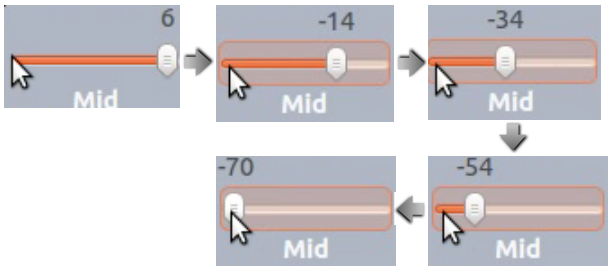

図 4.6: jackEQ のスライダの挙動．クリックする毎に値が大まかに変化する．

[4]http://www.gtk.org/
[5]jackEQ のスライダではクリックごとに-70~6 の間を 20 ずつ値が変化する．

スライダを囲む矩形を用いた対応付けにより同類スライダが認識されない． スライダを囲む矩形を用いた対応付けでは，矩形により範囲選択されたスライダをテンプレート画像としてテンプレートマッチングを用いて同類スライダを認識する．そのため，図 4.8 に示す様にスライダの値がそれぞれ異なる場合，入力画像との類似度が低くなるために同類スライダとして認識されない．この問題を解決するためには，類似スライダの認識を工夫する必要がある．例えば，同様のスライダを認識する場合はスライダのつまみの画像をテンプレート画像とし，アプリケーション上の同じつまみをテンプレートマッチングにより認識する．さらに，スライダの両端を認識することにより同類スライダが認識される．

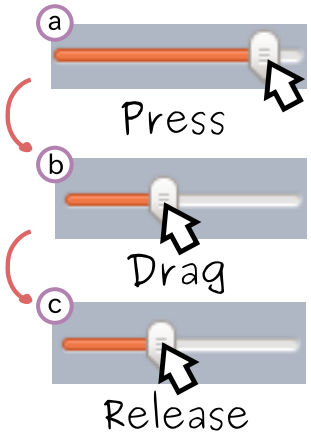

図 4.7: 本インタフェースを用いスライダの値を正確に制御するためには，a) つまみをプレスし，b) 入力された値までドラッグし，c) 入力終了後にリリースする必要がある．

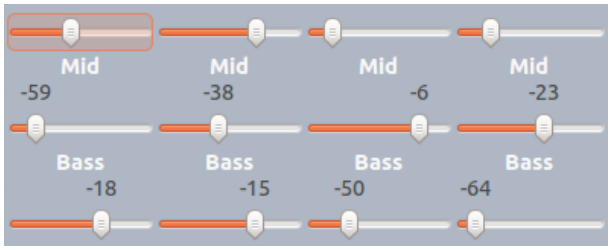

図 4.8: スライダの値がそれぞれ異なる場合，スライダを囲む矩形を用いた対応付け手法を用いて同類スライダが認識されない．

第5章　関連研究

本研究では，物理コントローラに対応付ける GUI をマウスを用いてなぞる様に描画することにより，物理コントローラと GUI とを対応付けるインタフェースを示す．本研究に関連する研究には，GUI を用いた操作との対応付けを行う研究，デスクトップ操作の拡張を行う研究，デスクトップに対する画像処理を行う研究，タンジブルユーザインタフェースに関する研究が挙げられる 本章ではこれら本研究に関連する研究をそれぞれ述べる．

5.1　GUI を用いた操作との対応付け

実世界の操作とアイコン・キー押下等の 0 次元の GUI とを対応付ける研究が示されている．Gohlke ら [GHL10] が示した TapShot は，デスクトップ上のアイコンのクリックをマルチタッチデバイスのタップに対応付けるインタフェースである．また，Block ら [BVG08] は，Villar ら [VGREG06, VG07] の電子部品を自由に追加し操作可能にするサーフェスである VoodooIO プラットフォームを用いて，デスクトップのコピー&ペーストを物理ボタンに割り当てる手法を示した． さらに，Hudson ら [HM06] は，物理的な紙とボタンを用いたハードウェアのプロトタイピングを容易にするために，物理ボタンの押下と GUI へのクリック又はキーイベントの発行を対応付けるインタフェースを示した．金子 [金子 12] らが示したインタフェースも同様にプロトタイピングを容易にするために，紙への押下と GUI への操作とを対応付ける．これらの研究は GUI への 0 次元操作とデバイスの操作を対応付けている．それに対して本手法では，物理コントローラに対応付ける GUI をマウスを用いてなぞる様に描画することにより，GUI の 2 次元操作を物理コントローラに対応付けることが可能である．

アイコン・キー押下のみではなく，2 次元の GUI と実世界の操作とを対応付ける研究が示されている．Block ら [BGV10] が示した Touch-Display Keyboards ではデスクトップのアイコンやスライダを手元のキーボードに対応付けた．さらに，Greenberg ら [GB02] の Customizable Physical Interfaces は，Phidgets [GF01] に繋がれた物理デバイスと GUI のボタン，メニュー，スライダ等を対応付けるインタフェースである．これらの研究では Windows の API を用い GUI とデバイスとの対応付けを行っており，画面外に隠れた GUI に対しても物理デバイスを用いて操作可能である利点を有するが，独自 GUI に対応しておらず利用可能な環境が限られる欠点がある． それに対し，本手法ではマウスイベントを用い GUI とデバイスとの対応付けを行うため，マウスを用いて操作可能である GUI ならば幅広く適用可能である利点を有する．

5.2 デスクトップ操作の拡張

本研究ではユーザのマウスを用いた操作をキャプチャし，それを用いGUIと物理コントローラとの対応付けを行うことにより，ユーザのデスクトップ操作を拡張している．

ユーザのマウスを用いた操作を活用しデスクトップ操作を拡張する研究が示されている．Appertら[ACP12]はマウスの逆操作をすることにより，操作のUndoとキャンセルが可能であるDwell-and-Springを示した．また大江ら[大江12]は，キャプチャした過去のマウスによる操作をなぞることによりUndo/Redoを可能にするインタフェースを示した．さらに，Baudischら[Bau98]は，マウスドラッグを用いて複数のトグルスイッチを一度に操作する手法を示した．本研究では，マウスを用いた操作の軌跡を物理コントローラの操作に対応付けることによりデスクトップ操作を拡張する．

マウスを用いた操作をキャプチャしアプリケーションの画像に重畳させることにより，ユーザの操作を視覚化する研究が示されている．Olsenら[OTF04]が示したScreenCrayonsは，アプリケーションのスクリーンショットを用いることにより，任意のアプリケーションに対してアノテーションを付けることが可能なシステムを示した．また，Nakamuraら[NI08]は，アプリケーションのスクリーンショット上にGUIに対する操作を重畳表示させ視覚化することにより，履歴操作を拡張させた．本研究においても対応付け編集の際に，対応付けにより描かれた線をアプリケーションのスクリーンショット上に重畳表示させることにより，マウスを用いた操作を視覚化している．

5.3 デスクトップに対する画像処理

本研究では複数のGUI群を対応付ける際に，デスクトップに対する画像処理を行い類似GUIを検索する．

本研究と同様にデスクトップに対する画像処理を行い，ユーザの操作を支援する研究がこれまでに示されてきた．Yehら[YCM09]はGUI画像を用いたプログラミングが可能な環境であるSikuliを示し，さらにSikuliを用いた項目ヘルプの生成ツールを示した[YCX+11]．また，坂本ら[坂本12]は撮りためたデスクトップのスクリーンショットを用いて画面間の類似度を認識することにより，取り消し操作を可視化する手法を示した．加えて，Dixonら[DLF11]は任意のGUI要素に対して，先行研究であるバブルカーソル[GB05]やPhosphor[BTC+06]を適用させるために，GUI要素をピクセル解析するPrefab[DF10]を用いてGUIの識別を行った．

Yehら[YCM09]や坂本ら[坂本12]の研究では，類似する要素を認識する際にデスクトップ上の画像に対するテンプレートマッチングを行っている．本研究においてもデスクトップ画像に対するテンプレートマッチングを用いることにより，デスクトップ上の類似GUIを認識している．

5.4 タンジブルユーザインタフェース

MIT の Tangible Media Group はユーザが触って（タンジブル），直接操作可能なシステムとして，タンジブルユーザインタフェース（TUI）と呼ばれるシステムを提案してきた．本研究は物理コントローラを触って GUI を操作可能にする点において TUI と関連する．

TUI の研究として，Ullmer ら [UI97] の metaDESK が挙げられる．metaDESK では，GUI の window，icon，menu が TUI の lens，phicon，tray に置き換えられ，ユーザはテーブルトップ上のタンジブルオブジェクトを操作してデジタル情報に働きかける．また，Jorda ら [JGAK07] はタンジブルオブジェクトを用いたライブ音楽パフォーマンス用のテーブルトップシステムである reacTable を示した．さらに，Weiss ら [WWJ+09] はボタン，スライダ，ノブ等の透明な物理コントローラを用いてテーブルトップシステムの操作を拡張し，Jansen ら [JDF12] は手元のスレート端末に物理スライダを取り付け大画面を操作する手法を示した．

これらの研究はインタフェースに GUI を置き換えるものとして TUI を導入し，直接触って操作可能であるインタフェースを実現している．その一方，本研究では現在一般的に使用されている GUI と物理コントローラを対応付け，物理コントローラを触って GUI を操作可能にすることを目標としている．

第6章　議論

本章では本研究の議論について述べる．最初に「対応付ける GUI に応じた処理の変更」について考察する．さらに,「本研究にて使用される物理コントローラ」について議論する．

6.1　対応付ける GUI に応じた物理コントローラの処理の変更

本研究では，様々な GUI に対して物理コントローラを容易に対応付けることを目的とし，物理コントローラと GUI の対応付けインタフェースを示した．第 4 章にて示した様に本インタフェースを実際のアプリケーションにて試用した結果，物理コントローラを操作した際にクリック処理では値を正確に入力できない GUI が存在した．例えば GUI スライダではつまみの位置をマウスプレスし，そのつまみをドラッグ&ドロップして操作しなければ正確な値を入力することが出来ない．また，図 6.1 にて示す様に一部の GUI ノブでは，ノブの円内をマウスプレスした後，円外の左又は右にドラッグし値を調整する必要がある．

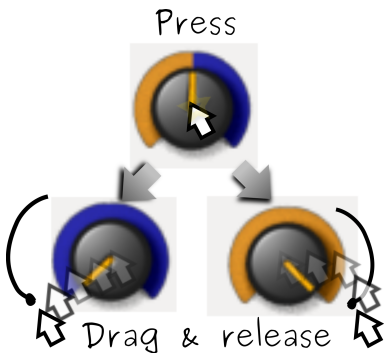

図 6.1: GUI ノブの操作方法．ユーザがノブを操作する際ノブの円内をマウスプレスした後，円外の左又は右にドラッグし値を調整する．

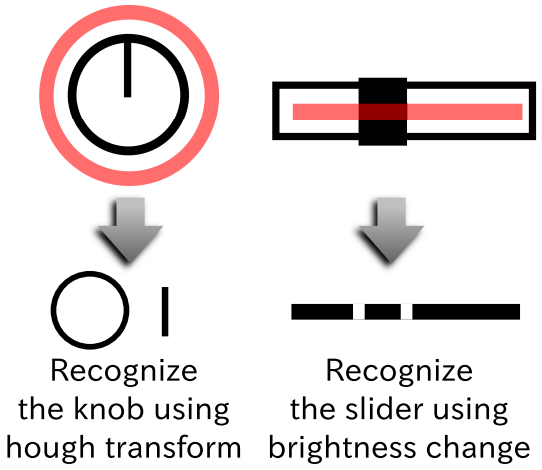

図 6.2: 対応付け時に GUI 要素を認識する．円を用いた対応付けの際はノブを認識し，直線を用いた対応付けの際はスライダを認識する．

以上を考慮すると，本インタフェースを用いて様々な GUI を操作可能にするためには，対応付ける GUI 要素を認識し，さらに認識結果に応じて物理コントローラを動かした際の処理を決定する必要がある．

対応付ける GUI 要素を認識するために，図 6.2 に示す手法を用いることを考える．例えば，円を用いた対応付けを行う際に円内の GUI 要素に対して画像処理を行う．具体的にはハフ変換を用いて円内の直線と丸を認識し，その認識結果を用いてノブを認識する．また，直線を用いた対応付けの際に直線上のピクセルの輝度値変化を算出し，その算出結果を用いてスライダを認識する．

以上の様に対応付ける GUI 要素を認識し，その認識結果に応じて物理コントローラを操作した際の処理を変更する．具体的には，ノブの要素に対して物理コントローラを操作した際は，図 6.1 に示す様にノブの中心をマウスプレスした後に，対応付けの円に沿ってマウスをドラッグさせる．また，スライダの要素に対して物理コントローラを操作した際は，スライダのつまみをクリックした後に対応付けの線に沿ってマウスをドラッグさせる．

これらの実装の様に対応付ける GUI を認識し，GUI に応じて処理を変更することによって，本インタフェースを様々な GUI に対応させることが可能になる．その一方，実際の GUI 要素は，その外見や操作時の挙動は様々である．そのため，様々な GUI に対応可能なシステムを構築するためには，さらなる工夫が必要である．

6.2 本研究にて使用される物理コントローラ

本研究では使用される物理コントローラとして物理スライダを用いた．本議論では使用するコントローラとして，図 6.3 に示す様に手元のタッチパネルを用いることを考える．

図 6.3: 対応付ける物理コントローラとしてタッチパネルを用いる例.

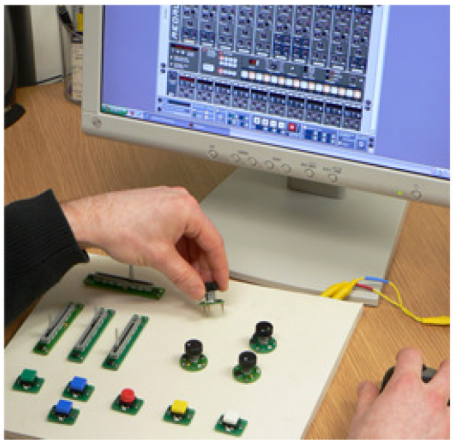

図 6.4: Villar らが示した VoodooIO（[VG07] より転載）.

コントローラとして手元のタッチパネルを用いることにより，Gohlke ら [GHL10] が示した TapShot の様に，対応付けた GUI を手元のタッチパネルに視覚的に表示させることが可能と

なる．タッチパネル上に視覚要素を表示させる場合，今回用いた物理スライダと比較すると，対応付けた要素を自由に配置することが可能になる．そのため，複数 GUI を対応付けた際に対応付けの関係をタッチパネル上の位置に反映させることが出来る．したがって，第 4 章にて示した音楽アプリケーションのイコライザ調整の際に発生した，対応付けの位置変更の手間が解決される可能性がある．その一方タッチパネルを用いた場合，物理コントローラを用いた際の様に触覚フィードバックが無い．そのため，第 4 章にて示したトーンカーブ調整での利点の様に，編集対象を注視した操作が困難となる．

この問題を解決するために，使用する物理コントローラとして Villar ら [VGREG06, VG07] が示した VoodooIO を用いることを考える．VoodooIO は図 6.4 に示す様な，ユーザが自由に電子部品を配置することが可能なサーフェスである．この VoodooIO を物理コントローラとして用いることにより，触覚フィードバックがある物理コントローラの利点を活かすことが出来る．さらに，対応付けの際に電子部品の配置を変更することにより，対応付けの関係の把握が容易になる．その一方，対応付けの度にユーザが電子部品の配置を変更する必要があり，そのことはユーザにとって手間となる可能性がある．

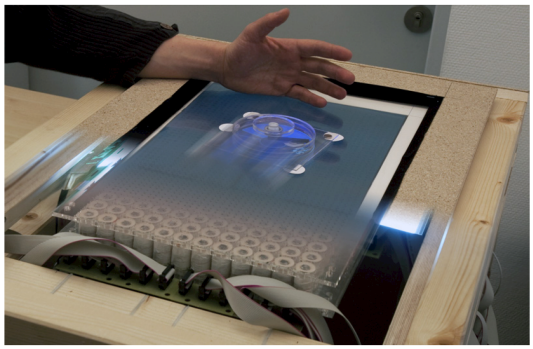

図 6.5: Weiss らが示した Madgets（[WSJB10] より転載）．

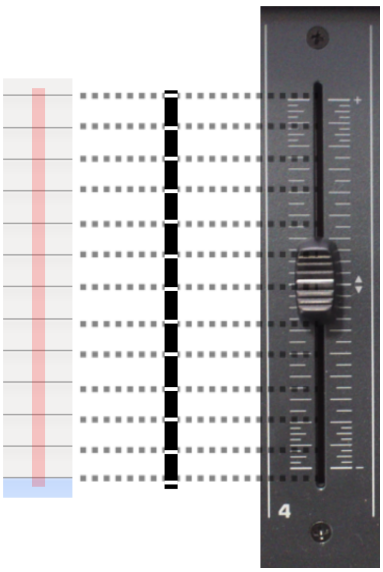

図 6.6: 対応付けの際にピクセルの輝度値変化を計算し，輝度値が変化した箇所を物理コントローラが触覚提示する．

使用する物理コントローラとして，Weiss ら [WSJB10] が示した Madgets を用いることを考える．Madgets は図 6.5 に示すテーブルトップであり，このテーブルトップ上に置かれた物理コントローラは電磁石によってその配置が自由に変更される．また，物理コントローラの状態（例えば，スイッチのオン/オフ）も電磁石によって制御される．物理コントローラとして Madgets を用い，対応付けた GUI の配置関係を物理コントローラに反映させる．そのことにより，ユーザにとって対応付けの配置が把握しやすくなると考える．

さらに使用する物理コントローラの触覚フィードバックを活用することを考える．例えば図 6.6 に示す様に，対応付けの際にピクセルの輝度値変化を計算し，輝度値が変化した箇所を物理コントローラが触覚提示する．この触覚提示により，物理コントローラを見ずに操作した際に，より正確に GUI を操作可能であると考える．この考えは触覚提示可能な物理コントローラに適用可能であり，今回の研究にて使用したモーターフェーダが搭載された BCF2000 や，Weiss ら [WSJB10] の Madgets にも適用可能である．物理コントローラの触覚出力を積極的に活用することにより，対応付けた GUI をマウスでは無く物理コントローラを用いて操作した際の，見ずに操作可能な利点をより活かせる．

第7章　結論

本論文では，様々なGUIに対して物理コントローラを容易に対応付けることを目的とし，そのアプローチとして，物理コントローラとGUIの対応付けインタフェースを示した．本インタフェースを用いてユーザは，デスクトップ上に表示されるGUIをマウスを用いてなぞる様に描くことにより，GUIと物理コントローラとを対応付ける．対応付けされたGUIは以降物理コントローラを用いて操作可能になる．本論文ではさらに，本インタフェースの実装を示し，様々なアプリケーションに対して実際に本インタフェースを試用した結果を示した．試用の結果，物理コントローラを用いて対応付けられたGUIをユーザは複数同時操作可能であることを確認した一方，様々なGUIに対して本インタフェースを適用するための課題が明らかになった．

謝辞

本論文の執筆に当たって，志築文太郎先生，三末和男先生，田中二郎先生，高橋伸先生には丁寧なご指導と貴重なご意見を頂きました．心より感謝致します．また，志築文太郎先生には日頃からきめ細かいご指導を頂き，大変お世話になりました．厚くお礼申し上げます．田中研究室のメンバーには研究生活を通じ多くのアドバイスを頂き，さらに，研究以外の部分でも大変お世話になりました．また，WAVE チームの皆様にはチームゼミで多くのご意見を頂きました．ここに深く感謝致します．

最後に，これまで自分を支えてくれた家族や友人に感謝致します．ありがとうございました．

参考文献

[ACP12] Caroline Appert, Olivier Chapuis, and Emmanuel Pietriga. Dwell-and-Spring: Undo for Direct Manipulation. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pp. 1957–1966. ACM, 2012.

[Bau98] Patrick Baudisch. Don't Click, Paint! Using Toggle Maps to Manipulate Sets of Toggle Switches. In *Proceedings of the 11th annual ACM symposium on User interface software and technology*, UIST '98, pp. 65–66. ACM, 1998.

[BGV10] Florian Block, Hans Gellersen, and Nicolas Villar. Touch-Display Keyboards: Transforming Keyboards into Interactive Surfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pp. 1145–1154. ACM, 2010.

[BTC+06] Patrick Baudisch, Desney Tan, Maxime Collomb, Dan Robbins, Ken Hinckley, Maneesh Agrawala, Shengdong Zhao, and Gonzalo Ramos. Phosphor: Explaining Transitions in the User Interface Using Afterglow Effects. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, UIST '06, pp. 169–178. ACM, 2006.

[BVG08] Florian Block, Nicolas Villar, and Hans Gellersen. A Malleable Physical Interface for Copying, Pasting, and Organizing Digital Clips. In *Proceedings of the 2nd international conference on Tangible and embedded interaction*, TEI '08, pp. 117–120. ACM, 2008.

[DF10] Morgan Dixon and James Fogarty. Prefab: Implementing Advanced Behaviors Using Pixel-Based Reverse Engineering of Interface Structure. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '10, pp. 1525–1534. ACM, 2010.

[DLF11] Morgan Dixon, Daniel Leventhal, and James Fogarty. Content and Hierarchy in Pixel-Based Methods for Reverse Engineering Interface Structure. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pp. 969–978. ACM, 2011.

[Fit54] Paul Morris Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of experimental psychology*, Vol. 47, No. 6, pp. 381–391, 1954.

[GB02] Saul Greenberg and Michael Boyle. Customizable Physical Interfaces for Interacting with Conventional Applications. In *Proceedings of the 15th annual ACM symposium on User interface software and technology*, UIST '02, pp. 31–40. ACM, 2002.

[GB05] Tovi Grossman and Ravin Balakrishnan. The Bubble Cursor: Enhancing Target Acquisition by Dynamic Resizing of the Cursor's Activation Area. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '05, pp. 281–290. ACM, 2005.

[GF01] Saul Greenberg and Chester Fitchett. Phidgets: Easy Development of Physical Interfaces through Physical Widgets. In *Proceedings of the 14th annual ACM symposium on User interface software and technology*, UIST '01, pp. 209–218. ACM, 2001.

[GHL10] Kristian Gohlke, Michael Hlatky, and Jorn Loviscach. TapShot: Screenshot Snippets as GUI Shortcuts. In *SIGGRAPH Posters*. ACM, 2010.

[HM06] Scott E. Hudson and Jennifer Mankoff. Rapid Construction of Functioning Physical Interfaces from Cardboard, Thumbtacks, Tin Foil and Masking Tape. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, UIST '06, pp. 289–298. ACM, 2006.

[JDF12] Yvonne Jansen, Pierre Dragicevic, and Jean-Daniel Fekete. Tangible Remote Controllers for Wall-Size Displays. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '12, pp. 2865–2874. ACM, 2012.

[JGAK07] Sergi Jordà, Günter Geiger, Marcos Alonso, and Martin Kaltenbrunner. The reacTable: Exploring the Synergy between Live Music Performance and Tabletop Tangible Interfaces. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, TEI '07, pp. 139–146. ACM, 2007.

[NI08] Toshio Nakamura and Takeo Igarashi. An Application-Independent System for Visualizing User Operation History. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*, UIST '08, pp. 23–32. ACM, 2008.

[OTF04] Dan R. Olsen, Jr., Trent Taufer, and Jerry Alan Fails. ScreenCrayons: Annotating Anything. In *Proceedings of the 17th annual ACM symposium on User interface software and technology*, UIST '04, pp. 165–174. ACM, 2004.

[UI97] Brygg Ullmer and Hiroshi Ishii. The metaDESK: Models and Prototypes for Tangible User Interfaces. In *Proceedings of the 10th annual ACM symposium on User interface software and technology*, UIST '97, pp. 223–232. ACM, 1997.

[VG07] Nicolas Villar and Hans Gellersen. A Malleable Control Structure for Softwired User Interfaces. In *Proceedings of the 1st international conference on Tangible and embedded interaction*, TEI '07, pp. 49–56. ACM, 2007.

[VGREG06] Nicolas Villar, Kiel Mark Gilleade, Devina Ramduny-Ellis, and Hans Gellersen. The VoodooIO Gaming Kit: A real-time adaptable gaming controller. In *Proceedings of the 2006 ACM SIGCHI international conference on Advances in computer entertainment technology*, ACE '06. ACM, 2006.

[WSJB10] Malte Weiss, Florian Schwarz, Simon Jakubowski, and Jan Borchers. Madgets: Actuating Widgets on Interactive Tabletops. In *Proceedings of the 23nd annual ACM symposium on User interface software and technology*, UIST '10, pp. 293–302. ACM, 2010.

[WWJ+09] Malte Weiss, Julie Wagner, Yvonne Jansen, Roger Jennings, Ramsin Khoshabeh, James D. Hollan, and Jan Borchers. SLAP Widgets: Bridging the Gap Between Virtual and Physical Controls on Tabletops. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '09, pp. 481–490. ACM, 2009.

[YCM09] Tom Yeh, Tsung-Hsiang Chang, and Robert C. Miller. Sikuli: Using GUI Screenshots for Search and Automation. In *Proceedings of the 22nd annual ACM symposium on User interface software and technology*, UIST '09, pp. 183–192. ACM, 2009.

[YCX+11] Tom Yeh, Tsung-Hsiang Chang, Bo Xie, Greg Walsh, Ivan Watkins, Krist Wongsuphasawat, Man Huang, Larry S. Davis, and Benjamin B. Bederson. Creating Contextual Help for GUIs Using Screenshots. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, UIST '11, pp. 145–154. ACM, 2011.

[金子 12] 金子将大, 田中二郎. 紙製タッチ入力インタフェース作成システム. 第 20 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 (WISS2012), pp. 189–190. 日本ソフトウェア科学会, 2012.

[坂本 12] 坂本有沙, 片山拓也, 寺田努, 塚本昌彦. デスクトップ上の画面変化に基づく取り消し操作の可視化機構の設計と実装. 情報処理学会研究報告 (149 回ヒューマンコンピュータインタラクション研究会), pp. 1–8. 情報処理学会, 2012.

[柴田 09] 柴田博仁. 大画面ディスプレイ・多画面ディスプレイの導入による業務効率化の測定. 情報処理学会論文誌, Vol. 50, No. 3, pp. 1204–1213, 2009.

[大江 12] 大江龍人, 志築文太郎, 田中二郎. 軌跡に基づいた undo/redo インタフェース. 情報処理学会研究報告 (149 回ヒューマンコンピュータインタラクション研究会), pp. 1–8. 情報処理学会, 2012.

[中村 00] 中村聡史, 塚本昌彦, 西尾章治郎. 2 つのマウスを用いたウィンドウ操作機構の設計と実装. ヒューマンインタフェース学会論文誌, Vol. 2, No. 4, pp. 35–47, 2000.