Vol.2013-HCI-152 No.3
2013/3/13

Draw-to-Map: 物理コントローラとGUIの対応付け手法

大江龍人[1,a)]　志築文太郎[2]　田中二郎[2]

概要：一部のアプリケーションでは，物理コントローラを用いることにより，複数の GUI 要素を同時かつ素早く操作可能である．しかしながら，物理コントローラの操作と対応付けられる GUI 操作は限られていた．本研究では，物理コントローラ操作と GUI 操作との対応付けを柔軟に定義することを可能にするインタフェースを示す．本インタフェースにおいて，ユーザはデスクトップ環境における GUI をポインタを用いてなぞる様に描くことにより，物理コントローラ操作と GUI 操作とを対応付けられる．ユーザは描く方法を変えることが可能であるため，様々な GUI 要素に対して対応付けを行うことが出来る．本稿では，本インタフェースのインタラクション手法，実装，及び適用例を示す．

キーワード：Draw-to-Map，物理コントローラ，GUI，対応付け，スライダ，デスクトップ環境

Draw-to-Map: Mapping Techniques between Physical Controller and GUI

Tatsuhito Oe[1,a)]　Buntarou Shizuki[2]　Jiro Tanaka[2]

Abstract: In some applications, a user can manipulate GUI elements simultaneously and rapidly using a physical controller. However, GUI manipulations which one can define mapping with the physical controller are limited. In our research, we introduce a technique where one can define the mapping flexibly between the physical controller and GUI manipulation. In the technique, one can define the mapping by drawing a stroke on a GUI element using a pointer. Because one can draw in various ways, one can perform mapping to various GUI elements in various ways. In this paper, we show our interaction techniques, implementation, and application examples.

Keywords: Draw-to-Map, Physical Controller, GUI, Mapping, Slider, Desktop Environment

1. 序論

一部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が可能である．例えば Ableton 社[*1]の Live や Propellerhead 社[*2]の Reason 等の音楽アプリケーションでは，物理コントローラは音量，音色，及びエフェクトのパラメータ調整等のために用いられる．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 GUI 要素を同時かつ素早く操作可能である． しかしながら，物理コントローラを使用出来るアプリケーションは一部に限られており，物理コントローラの操作と対応付けられる GUI 操作は限られていた .

本稿では，物理コントローラ操作と GUI 操作との対応付けを柔軟に定義することを可能とする Draw-to-Map を示す．Draw-to-Map では，ユーザはデスクトップ上の GUI をポインタを用いてなぞる様に描くことにより，対応付けを行う．ユーザは対応付け時に描く方法を変えることにより，様々な GUI 要素に対して様々な対応付けを行うことが可能となる．さらに，従来物理コントローラを用いて操作することが出来なかった GUI 操作も物理コントローラを用いて操作可能となる．

[1] 筑波大学大学院システム情報工学研究科コンピュータサイエンス専攻
Department of Computer Science, Graduate School of Systems and Information Engineering, University of Tsukuba
[2] 筑波大学システム情報系
Faculty of Engineering, Information and Systems, University of Tsukuba
[a)] tatsuhito@iplab.cs.tsukuba.ac.jp

[*1] https://www.ableton.com/
[*2] http://www.propellerheads.jp/

2. Draw-to-Map

Draw-to-Map 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，物理コントローラ操作と GUI 操作とを対応付ける手法である．Draw-to-Map を用いて，ユーザは図 1 に示す様に対応付ける GUI をなぞる．なお，今回，物理コントローラとして既存のアプリケーションにおいて用いられる物理スライダを用いることとした．

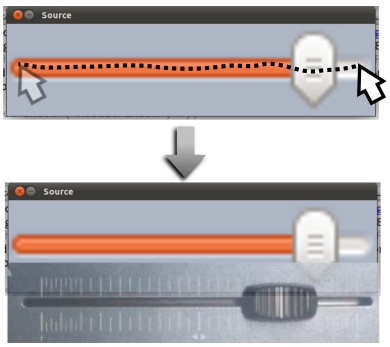

図 1 Draw-to-Map を用いて GUI と物理コントローラとを対応付ける．

Fig. 1 A user defines the mapping between the GUI and the physical controller using Draw-to-Map.

対応付け操作には，描画ソフトウェアに備わる様な描画操作を用いることが可能である．この操作として今回，直線，自由曲線，矩形，円を用いた対応付けを実装した．また，GUI 要素を多く持つアプリケーションも存在するため，それらの GUI 要素をまとめて対応付けられるようにするための一括対応付けを実装した．

2.1 対応付け操作

Draw-to-Map を用いて対応付ける方法を以下に説明する．

(a) Draw-to-Map を行うための GUI を立ち上げる (図 2a).
(b) 対応付けする手法を選択する (図 2b).
(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 2c)．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 2d)．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 (図 2e).

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 Draw-to-Map の対応付け手法

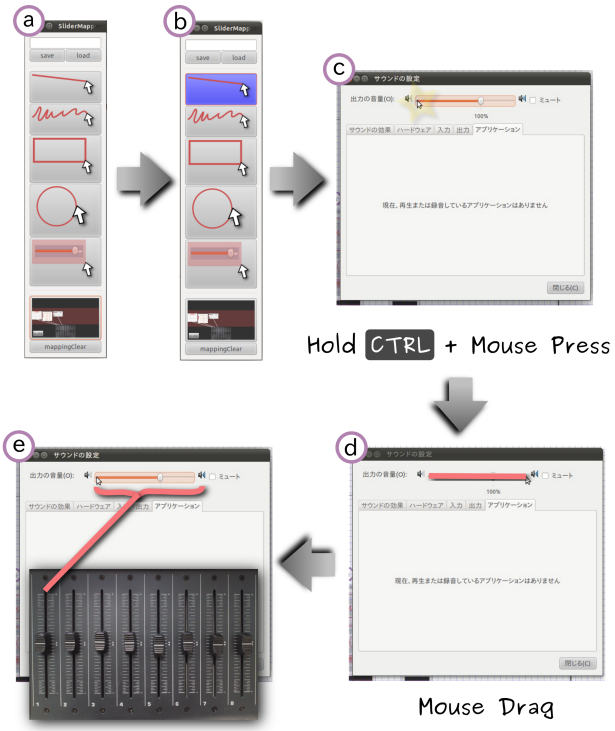

図 2 Draw-to-Map を用いたインタラクションの流れ．a) Draw-to-Map を行うための GUI を立ち上げる，b) 対応付けする手法を選択する，c) CTRL キーを押しながらマウスをプレスする，d) マウスをドラッグする，e) 描いた箇所と物理コントローラが対応付けられる．

Fig. 2 An interaction flow using Draw-to-Map.

をそれぞれ述べる．

2.2 対応付け手法

直線対応付け 直線対応付けは，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 2b~2e に示される．対応付け後，直線の始点が物理スライダの上端，直線の終点が物理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コントローラとを割り当てる手法であり，GUI スライダの様な 1 次元の GUI と物理コントローラを対応付ける際に有効である．

自由曲線対応付け 自由曲線対応付けは，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，自由曲線と物理コントローラとを対応付ける手法である．自由曲線対応付けのインタラクションの流れを図 3 に示す．

ユーザが自由曲線を描くと，自由曲線の始点が物理スライダの上端，自由曲線の終点が物理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コントローラとを

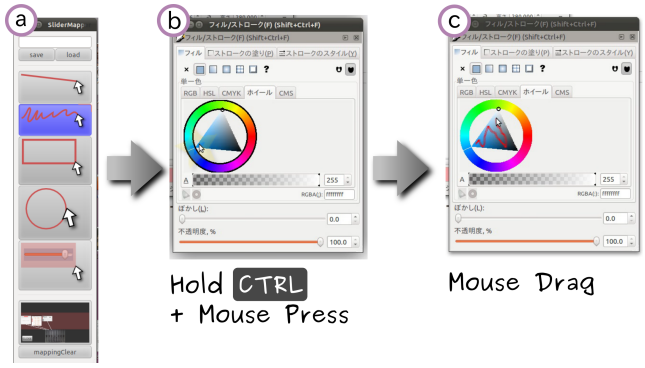

図 3 自由曲線対応付け．a) 対応付け手法の選択，b) CTRL キーを押しながら自由曲線の始点をプレスする，c) 自由曲線を描く．

Fig. 3 A mapping technique using free-form curve.

対応付けることが可能となり，カーブやジグザグを描く様な GUI 操作と物理コントローラの操作を対応付けることが可能になる．例として図 3c に示す様に，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ることが出来る．

矩形対応付け 矩形対応付けは，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．矩形対応付けのインタラクションの流れを図 4 に示す．

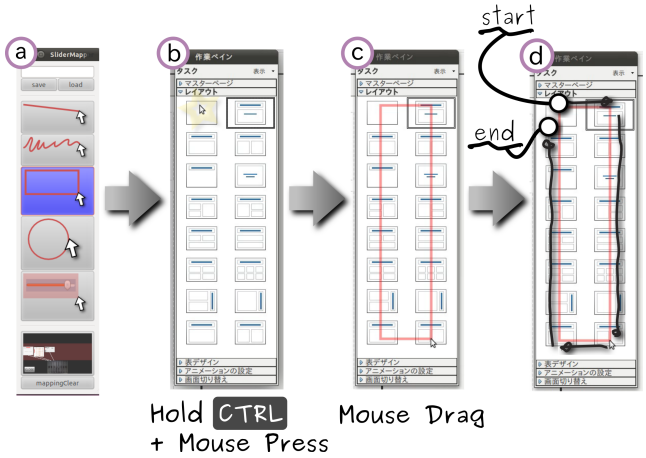

図 4 矩形対応付け．a) 対応付け手法の選択，b) CTRL キーを押しながら矩形の始点をプレスする，c) 矩形を描く，d) 矩形の左上端が開始点として物理スライダの上端に割り当てられ右回りに動作する．

Fig. 4 A mapping technique using rectangle.

ユーザが矩形を描くと，図 4d に示す様に矩形の左端が物理スライダの上端に対応付けられ，物理スライダを下に動かすと右回りに GUI が操作される．矩形対応付けは図 4b の様な GUI 要素群が 2 列に並んだメニューと物理コントローラとを対応付ける際に有効である．

円対応付け 円対応付けは，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．円対応付けのインタラクションの流れを図 5 に示す．

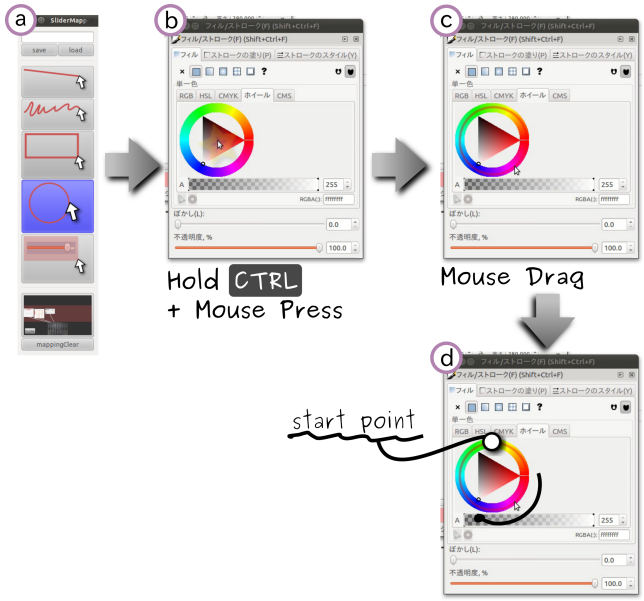

図 5 円対応付け．a) 対応付け手法の選択，b) CTRL キーを押しながら円の中心をプレスする，c) 円を描く，d) 円の最上点が開始点として物理スライダの上端に割り当てられ右回りに動作する．

Fig. 5 A mapping technique using circle.

ユーザが円を描くと，図 5d に示す様に，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．ユーザが物理コントローラ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．物理コントローラを下端まで操作すると円の最上点に戻る．円対応付けは図 5b の様な円形の GUI と物理コントローラとを対応付ける際に有効である．

2.3 一括対応付け

一括対応付けは，複数の GUI スライダ群と物理コントローラとを一度に対応付ける手法である．ユーザは，スライダの一つ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可能である．個々のスライダと物理コントローラとは直線対応付けにより対応付けられる．一括対応付け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，一括対応付けを選択する (図 6a).

(b) 一括に対応付けるスライダを選択するために，ユーザは 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 6b)．ユーザは音楽アプリケーションのスライダ群と物理コントローラを対応付けようとしている．

(c) マウスをドラッグし対応付けるスライダを囲んだのち，マウスをリリースする (図 6c).

(d) 矩形によって囲まれたスライダとそれに類似するス

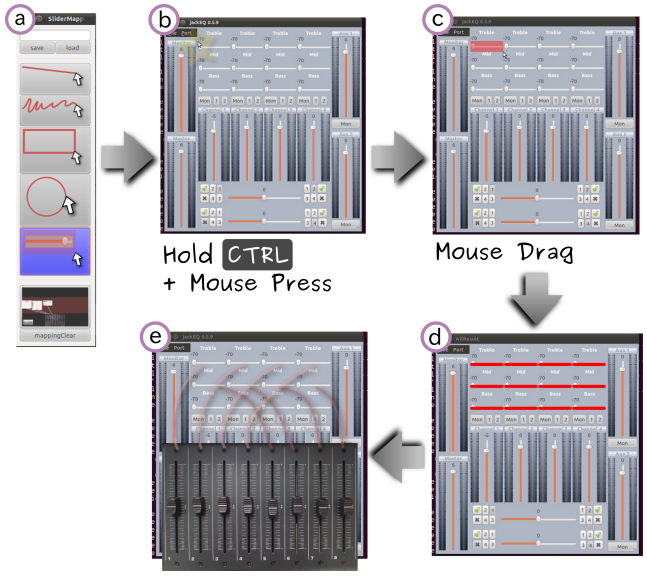

図 6 一括対応付け．a) 対応付け手法の選択，b) CTRL キーを押しながらスライダを囲む矩形の左端をプレスする，c) マウスをドラッグしスライダを囲む，d) 選択したスライダに類似するスライダ群が認識される，e) 認識されたスライダ群が物理コントローラに対応付けられる．

Fig. 6 A mapping technique in a lump.

ライダ群が認識される (図 6d).

(e) 認識されたスライダ群と物理コントローラが自動的に対応付けられる (図 6e)．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．

一括対応付けを用いることにより，ユーザは複数の GUI スライダ群に対して，まとめて直線対応付けを行うことが出来る．そのため，要素が複数存在する場合の対応付けに本手法は有効である．

3. 対応付けの編集

Draw-to-Map を用いて対応付けを行った結果，ユーザはその対応付けに失敗する場合がある．また，一括対応付け時に使用しない対応付けが存在する場合や，物理コントローラに対応付けられた順番を変更したい場合が存在する．そのため，我々は対応付けを編集する機能を実装した．ユーザは本機能を用いることにより，GUI と物理コントローラの対応付けの解除及び，対応付けの位置の変更が可能である．対応付けの編集機能の操作例を説明する．

(a) 対応付けの編集画面を立ち上げる (図 7a).

(b) 編集画面には対応付けられている GUI のサムネイルが表示される (図 7b)．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となる．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 (図 7c).

(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 (図 7d)．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 (図 7e).

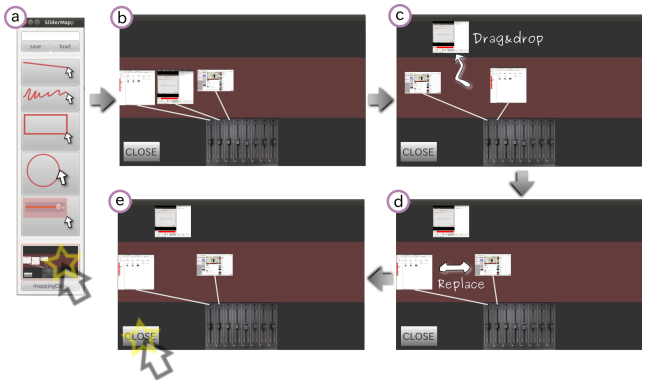

図 7 対応付けの編集機能．a) ボタンを押すと編集画面が立ち上がる，b) 編集画面には物理コントローラに対応付けられている GUI のサムネイルが表示される，c) 対応付けを解除するためにサムネイルを赤い領域外にドラッグ&ドロップする，d) 対応付けの位置を変更するためにサムネイルの位置を変更する，e) 「CLOSE ボタン」を押すと編集結果が保存される．

Fig. 7 A function to edit mapping.

4. 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，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となる．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を再利用することが可能となる．

5. 実装

本節では Draw-to-Map の実装について述べる．今回，実装を Ubuntu Linux[*3] 上にてプログラミング言語 Python[*4] を用いて行い，物理コントローラには BEHRINGER 社の BCF2000[*5] を用いた．

Draw-to-Map の実装の全体像は図 8 に示すものであり，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，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+マウス入力により対応付けが行われた際に，対応付け

[*3] http://www.ubuntu.com/
[*4] http://www.python.org/
[*5] http://www.behringer.com/EN/Products/BCF2000.aspx

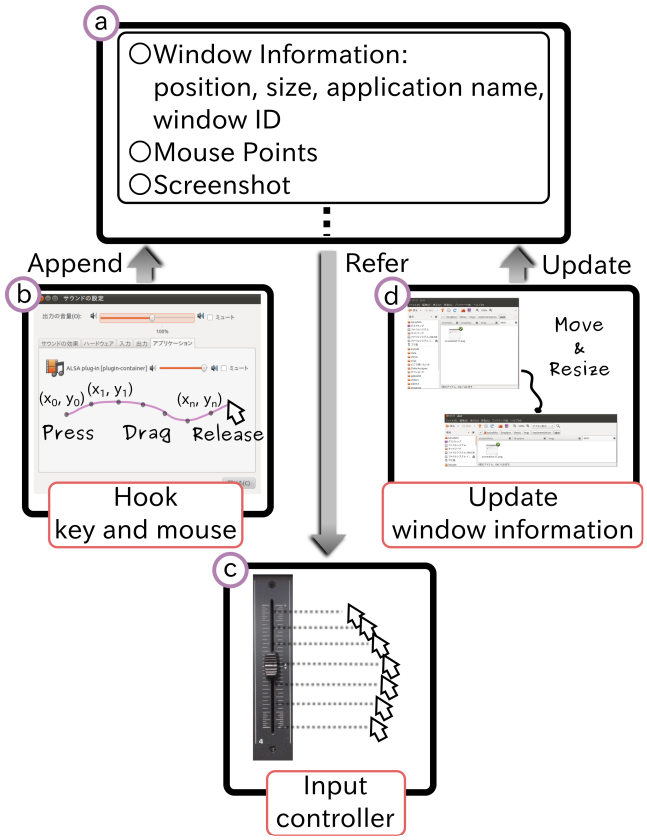

図 8 実装概要．a) 対応付け情報の保持，b) キーボードとマウスのフック，c) 物理コントローラの入力，d) ウィンドウ情報の更新．

Fig. 8 Implementaiton overview.

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 のウィンドウ情報．ウィンドウ情報は，対応付けられた GUI のウィンドウの位置，大きさ，名前，及び ID から構成される．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．
- ウィンドウ位置からの相対マウス座標群．対応付け時にユーザによって描かれたマウス座標群が，ウィンドウ位置からの相対座標として保持される．
- 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，対応付けられた GUI のアプリケーションのスクリーンショットである．

対応付け情報は JSON[*6]形式にてシステム内に保持される．この JSON ファイルを保存・復元することにより，対応付けの状態が保存・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，対応付け情報を追加する．

[*6] http://www.json.org/

物理コントローラの入力． 物理コントローラの入力では，物理コントローラの入力に応じてシステムがマウスイベントを発行し GUI を操作する．発行されるマウスイベントは，デスクトップ上のある位置に対するクリックイベントであり，その位置とは対応付け情報として保持されるウィンドウ位置と相対マウス座標とを足しあわせた位置となる．

ウィンドウ情報の更新． ウィンドウ情報の更新では，対応付け情報内の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，適切な位置へマウスイベントが発行される．

5.1 一括対応付けの実装

一括対応付けは，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用いた画像処理によって実現している．

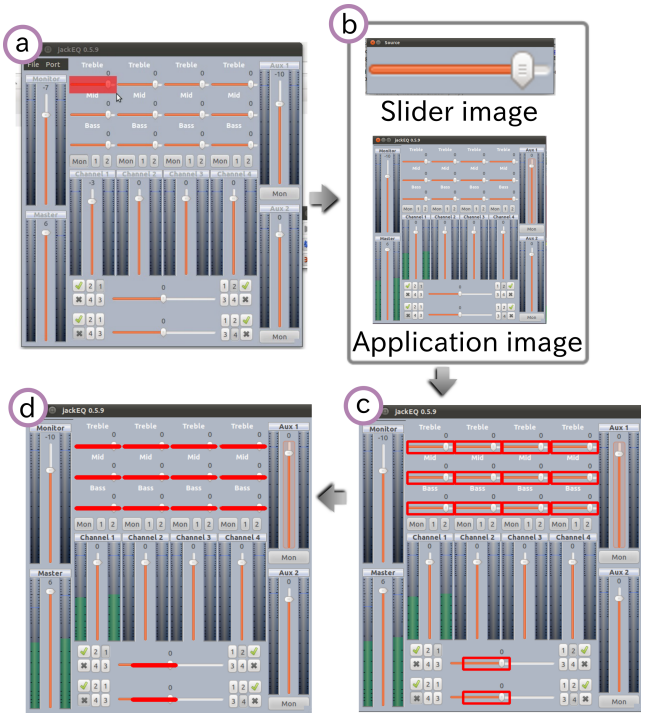

図 9 一括対応付けの処理の流れ．a) ユーザによりスライダが選択される，b) システムがスライダとアプリケーションの画像をキャプチャする，c) スライダ画像と類似したスライダ群が認識される，d) スライダ群と物理コントローラが直線対応付けにより対応付けられる．

Fig. 9 Processing of mapping in a lump.

一括対応付けの処理の流れを以下に示す．

(a) ユーザによりスライダが選択される (図 9a).

(b) システムが選択されたスライダとアプリケーションの画像をキャプチャする (図 9b).

(c) スライダ画像をテンプレート画像とし，テンプレートマッチングを用いてアプリケーション内の類似したスライダ群を認識する (図 9c).

(d) 類似したスライダ群に対して，スライダの始点と終点を認識する (図 9d)．それぞれのスライダが，直線対応付けにより物理コントローラと対応付けられる．

スライダの始点と終点は図 10 に示す様に，スライダ画像の輝度値変化を用いて認識される．具体的には，スライダ画像の短辺を 2 分割し，その直線上の輝度値変化を走査する．そして，輝度値が変化する左端と右端を，スライダの始点と終点として認識する．

図 10 スライダの始点と終点の認識原理．スライダ画像の輝度値変化を用いて始点と終点を認識する．

Fig. 10 A principle to recognize slider's start and end point.

6. Draw-to-Map の適用例

本節では Draw-to-Map の適用例を示し，それぞれの例における利点及び問題点を述べる．

6.1 画像編集アプリケーションのトーンカーブ調整

画像編集アプリケーションとして GIMP[*7]を用い，Draw-to-Map をトーンカーブに適用させ，物理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．ユーザは図 11a に示す様に直線を用いた対応付けを複数回使用し対応付けを行い，図 11b に示す様に物理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．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 を見ずに画像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ながら編集対象の画像を閲覧する必要があるため，画像を注視した編集が困難である．それに対し物理コントローラを用いる場合，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，画像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．

[*7] http://www.gimp.org/

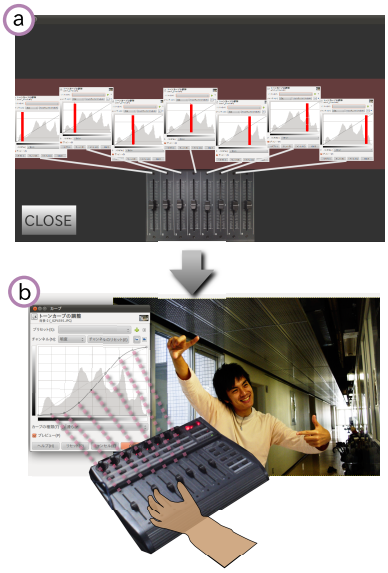

図 11 Draw-to-Map を用いてトーンカーブ調整の GUI に対応付ける例．a) 直線対応付けを複数回使用し対応付けを行う，b) 物理コントローラを動かすことによりトーンカーブの各制御点を動かすことが出来る．

Fig. 11 An example of mapping to tone curve GUIs using Draw-to-Map.

6.2 ドローイングアプリケーションの色調整

図 12a に示す様に，ドローイングアプリケーションの色相，彩度，明度，透明度等の色調整機能に物理コントローラを割り当て，図 12b に示す様に手元の物理コントローラを用いて色調整を行った．今回，ドローイングアプリケーションとしては Inkscape[*8]を用いた．色調整に適用させた際の利点として，色のパラメータの同時変更が可能であった．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．

6.3 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[*9]を用い，一括対応付け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た．

物理コントローラを用いて GUI スライダ群を操作可能となった一方，実際に使用し以下の問題が明らかになった．

物理コントローラを用いて正確な値の制御が不可能である． この原因は，jackEQ のスライダはスライダ上の位置をクリックする毎に，値が大まかに変化するためである．そのため，本インタフェースの絶対座標にクリックイベントを発行する実装では，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正確に GUI スライダを操作するためには，スライダのつまみをプレスし，入力された値までドラッグし，入力終了後に

[*8] http://inkscape.org/
[*9] http://djcj.org/jackeq/

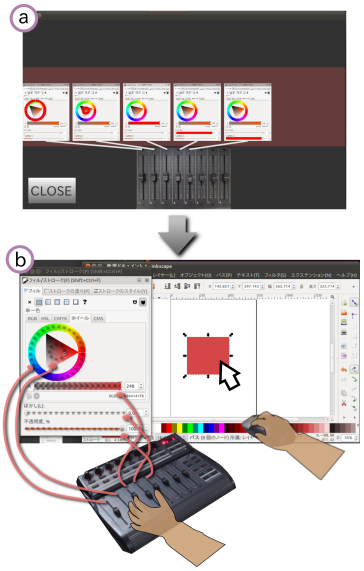

図 12 Draw-to-Map を用いて色調整の GUI に対応付ける例．a) 色相，彩度，明度，透明度等に物理コントローラを割り当てる．b) 物理コントローラを用いて色を調整する．

Fig. 12 An example of mapping to color selection GUIs using Draw-to-Map.

マウスをリリースする実装に変更する必要がある．

類似スライダが認識されない． 一括対応付けではテンプレートマッチングを用いて類似スライダを認識するため，ユーザが入力する画像によっては類似度が低く類似スライダが認識されない場合があった．

7. 関連研究

本研究に関連する研究には，GUI の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，及びデスクトップに対する画像処理を行う研究が挙げられる．本節ではこれら本研究に関連する研究をそれぞれ述べる．

7.1 GUI の操作との対応付け

実世界の操作とアイコン・キー押下等の 0 次元の GUI 操作とを対応付ける研究が示されている．Gohlke ら [1] 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 [2] は，Villar ら [3], [4] 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 [5] 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 [6] ら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．これらの研究に対して本手法では，マウスを用いて GUI をなぞる様に描くことにより，GUI の 2 次元操作と物理コントローラ操作とを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 [7] 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る．さらに，Greenberg ら [8] の Customizable Physical Interfaces は，Phidgets [9] 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，幅広い 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 GUI へのなぞりを用い対応付けを行うため，幅広い GUI に適用可能である利点を有する．

7.2 デスクトップ操作の拡張

本研究ではマウスによる操作を用い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．

マウスを用いた操作によりデスクトップ操作を拡張する研究が示されている．Appert ら [10] はマウスの逆操作により，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 [11] は，過去のマウスを用いた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 [12] 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，マウスの描く操作を用い対応付けを行うことにより，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 [13] が示した ScreenCrayons は任意のアプリケーションに対して，スクリーンショットを用いアノテーションを付けることが可能なシステムである．また，Nakamura ら [14] 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．

7.3 デスクトップに対する画像処理

本研究では一括対応付け時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 [15] 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 を示した．また，坂本ら [16] 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 [17] 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 [18] や Phosphor [19] を適用させるために，GUI 要素をピクセル解析する Prefab [20] を用いて GUI の認識を行った．

Yeh ら [15] や坂本ら [16] 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用いる．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，類似 GUI を認識している．

8. 結論と今後の課題

本稿では，物理コントローラ操作と GUI 操作との対応付けを柔軟に定義することを可能とする Draw-to-Map を示した．Draw-to-Map を様々なアプリケーションに実際に適用させたところ，従来物理コントローラを用いて操作することが出来なかった GUI 操作も物理コントローラを用いて操作可能であった．

今後は 6.3 節における問題を受け，対応付け時に GUI スライダか否かを認識し，発行するマウスイベントを変更させることにより，対応付け後に GUI スライダを正確に操作可能にしたい．また一括対応付け時に，テンプレートマッチングより類似スライダが認識されない場合があった．今後は，テンプレート画像としてスライダのつまみ画像を用いることにより，認識精度を向上させることを考えている．また，対応付けの編集インタフェースを強化することにより，1 つの物理コントローラを用いて複数 GUI を同時操作，あるいは連続操作出来るようにすることも考えている．

参考文献

[1] Gohlke, K., Hlatky, M. and Loviscach, J.: TapShot: Screenshot Snippets as GUI Shortcuts, in *SIGGRAPH Posters*, ACM (2010).

[2] Block, F., Villar, N. and Gellersen, H.: 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).

[3] Villar, N., Gilleade, K. M., Ramduny-Ellis, D. and Gellersen, H.: 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).

[4] Villar, N. and Gellersen, H.: 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).

[5] Hudson, S. E. and Mankoff, J.: 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).

[6] 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).

[7] Block, F., Gellersen, H. and Villar, N.: 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).

[8] Greenberg, S. and Boyle, M.: 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).

[9] Greenberg, S. and Fitchett, C.: 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).

[10] Appert, C., Chapuis, O. and Pietriga, E.: 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).

[11] 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).

[12] Baudisch, P.: 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).

[13] Olsen, D. R., Jr., Taufer, T. and Fails, J. A.: ScreenCrayons: Annotating Anything, in *Proceedings of the 17th annual ACM symposium on User interface software and technology*, UIST '04, pp. 165–174, ACM (2004).

[14] Nakamura, T. and Igarashi, T.: 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).

[15] Yeh, T., Chang, T.-H. and Miller, R. C.: 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).

[16] 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).

[17] Dixon, M., Leventhal, D. and Fogarty, J.: 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).

[18] Grossman, T. and Balakrishnan, R.: 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).

[19] Baudisch, P., Tan, D., Collomb, M., Robbins, D., Hinckley, K., Agrawala, M., Zhao, S. and Ramos, G.: 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).

[20] Dixon, M. and Fogarty, J.: 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).