

テーブル型タッチディスプレイを用いた協調作業を促進する 同時マルチタッチジェスチャ操作機構

藪井淳史[†] 中島誠[†]

概要: 近年、テーブル型ディスプレイのような大型のデバイスにもマルチタッチ機能を有するタッチパネルが使われるようになった。これまで、我々は複数ユーザがテーブル型タッチディスプレイを取り囲んで行う協調作業を支援するため、シングルユーザ向けアプリケーションを複数人で使えるようにするミドルウェア、コラボトレイを開発してきた。本論文では、複数人がマルチタッチジェスチャによる操作を同時に行えるようにする仕組みの実現方法を提案する。被験者実験を通して、協調作業におけるマルチタッチジェスチャ操作性の検証を行い、マルチタッチジェスチャ操作機構の有効性を確かめた。

キーワード: テーブル型タッチディスプレイ, 協調作業支援, 同時マルチタッチジェスチャ

Handling Simultaneous Multi-Touch Gestures on a Touchscreen Table to Facilitate Collaborative Work

ATSUSHI YABUI[†] MAKOTO NAKASHIMA[†]

Abstract: In recent years, a touch screen having a multi-touch function has been used for a large-sized device, such as a table type display. We have developed a middleware, called CollaboTray that can be loaded with any off-the-shelf single-user application while retaining its functionality and usability, for boosting daily collaborative work with a table type display. We here propose a mechanism to enhance the CollaboTray technology by enabling multiple users to operate multiple applications with simultaneous multi-touch gestures. The efficacy of the proposed mechanism for collaborative work was clarified in an experiment with 10 university students.

Keywords: Touch screen table, Computer Supported Collaborative work, Multi-touch gesture

1. はじめに

現在、スマートフォンやタブレットのようなマルチタッチ機能を搭載した様々なデバイスが広く普及している。その中で、テーブル型ディスプレイのような大型のタッチデバイスが増え、これを囲んでの協調作業をする機会が多くなってきた。これまで、このような協調作業を支援するため、日常的に利用するシングルユーザ用の Windows アプリケーションを改変することなくマルチユーザで共有できるようにするミドルウェア、コラボトレイを提案してきた[1]。

コラボトレイは、通常、OS とアプリケーション間で行う、アプリケーションのウィンドウイメージと、マウスやキーボードなどによるイベントの送信経路に介入する仕組みにより、本来のアプリケーションが行う事ができないアプリケーションウィンドウの複製や回転を実現し、協調作業を支援可能としている。また、シングルユーザ用のアプリケーションで、複数のユーザによる同時操作を可能にする、同時操作制御機構[2]や、通常の PC 上で複数のアプリケーションを同時操作可能にする、複数アプリケーション制御機構[3]が構築してきた。しかしながら、日常的に利用するアプリケーションの多くは、マルチタッチ機能に対応して

いないことから、スマートフォンなどでよく利用される、ピンチインやピンチアウトなどの、複数指でのマルチタッチジェスチャ操作には対応できていない。

本論文では、テーブル型ディスプレイでの円滑な協調作業環境の提供のため、タッチ機能に対応していないアプリケーションを、複数のユーザが効果的にマルチタッチジェスチャで利用できるようにする、同時マルチタッチジェスチャ操作機構を提案する。複数ユーザが同時に操作を行う状況では、それぞれの操作がどのような操作なのかを認識する必要がある。提案する仕組みでは、まず、マルチタッチジェスチャかどうかの認識を行い、それぞれのマルチタッチジェスチャに応じたイベントを発生させてアプリケーションに送信することで、複数人での同時マルチタッチジェスチャ操作を可能にする。

以下 2 章では、本研究の位置づけとして、マルチユーザでの共同作業に関する関連研究とその問題点、マルチタッチジェスチャの概要と認識について述べる。3 章では、複数アプリケーション上で同時マルチタッチとマルチタッチジェスチャを行うためのマルチタッチジェスチャ操作機構の仕組みについて述べる。4 章では、マルチタッチジェ

[†] 大分大学
Oita University

スチャ操作機構が組み込まれたコラボトレイを用いて行った、マルチユーザでの協調作業実験の結果と、考察を述べる。最後にまとめと今後の展望について述べる。

2. 本研究の位置づけ

関連研究として、マルチユーザがオンライン上で協調作業を行うことができる OneDrive[4], OfficeOnline[5], GoogleDrive[6]などがある。これらの共通点は、1つのファイルをオンライン上で複数人が同時に編集可能なアプリケーションを用意してあることである。これに対して、マルチユーザが対面で協調作業を行うことが出来る TeamPlayer3[7]や、コラボトレイの共通点は、既存のアプリケーションを改変することなく複数人での操作を支援している点にある。

2.1 TeamPlayer3

TeamPlayer3[7]は、DicoLAB 社が提供する、同一の計算機上でマルチユーザでの操作を可能にするソフトウェアである。マウスやタブレットを複数接続することで、デスクトップ上に各ユーザを識別することが出来る独自のマウスカーソルを表示し、同時にマウスを操作する機能や、キーボードを複数接続することで、文字入力などをそれぞれの機器で行うことを可能とする機能を有している。特徴は、シングルユーザの作業環境を、マルチユーザでの協調作業が可能な環境に変え、アプリケーションやデバイスに依存しないことである。しかしながら、協調作業を行う場合、ユーザ数だけ接続機器が必要であること、1人のユーザがデスクトップ上で操作している間、その他のユーザはクリックや文字入力などの操作をすることが出来ない問題がある。

2.2 コラボトレイ

コラボトレイ[1]は、協調作業環境を構築するミドルウェアである。ユーザは、利用したい任意のアプリケーションをコラボトレイの上に載せる（ドラッグアンドドロップする）だけで、シングルユーザ環境下でマルチユーザの作業環境に移ることができる。コラボトレイでは、アプリケーションのウィンドウイメージを取得し、コラボトレイ上にウィンドウイメージのみを表示しており、その時に操作の邪魔をしないために、本物のアプリケーションは透明にして画面端に退避させている。そして、キーボードイベントや、マウスイベントなどのコラボトレイ上での操作イベントを取得し、アプリケーション本体に情報を送っている。また、協調作業を円滑にする機能として、共有しているアプリケーションウィンドウのクローン（操作可能）を作成する機能や、ユーザの向いている方向に、アプリケーションの向きを変更する回転機能、アプリケーションの表示を拡大・縮小する機能を有する。

図1に実際のコラボトレイを使用する様子を示す。円

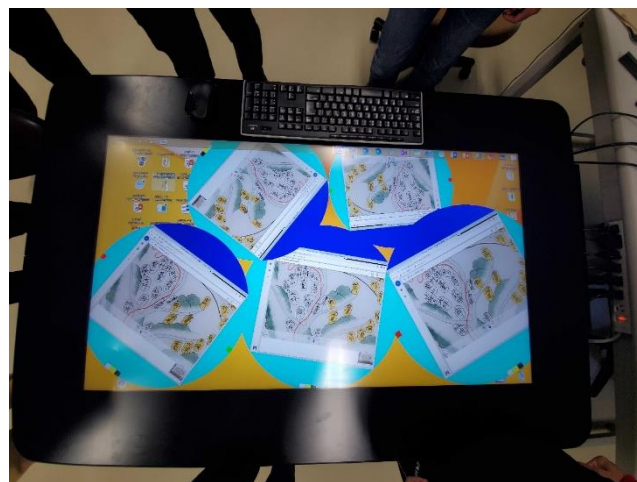


図1 コラボトレイの使用例

Figure 1 Example of CollaboTray.

形のコラボトレイのウィンドウ上に、画像閲覧アプリケーションのウィンドウが載っている。オリジナルのコラボトレイウィンドウの他、4つのクローンウィンドウが表示されている。コラボトレイウィンドウの水色の部分をドラッグすることで、コラボトレイを回転・拡大・縮小することが可能である。1つのアプリケーションウィンドウ上で、複数人が同時に操作可能な同時操作制御機構[3]や、複数のアプリケーションウィンドウ上で、それぞれ同時にシングルタッチ操作を可能にする複数アプリケーション制御機構[4]を備えている。しかしながら、複数のアプリケーション上で、複数人がマルチタッチ操作を可能にする仕様にはなっていない。

3. 複数アプリケーションでの同時マルチタッチジェスチャ操作機構

複数のアプリケーションを同時に操作できる「複数アプリケーション制御機構」およびシングルユーザ向けアプリケーションを複数人で同時に操作できる「同時操作制御機構」で制御することができるのは、シングルタッチ操作のみであって、マルチタッチ操作で構成されるマルチタッチジェスチャ操作には対応していない。本章では、前者の機構におけるイベント管理方法と後者のイベント収集方法を基にした、複数のアプリケーションで、同時にマルチタッチジェスチャ操作を行えるようにする仕組みを提案する。

3.1 マルチタッチジェスチャ操作の判別

テーブル型ディスプレイ、スマートフォン、タブレットなど、様々なマルチタッチデバイスで行うことのできるマルチタッチジェスチャは多数存在し、図2のようなものが挙げられる。これらのジェスチャは拡大・縮小・回転などの操作を直感的に行うことができ、マウスや複雑なタッチ操作のかわりとして行うことが出来る。



図2 マルチタッチジェスチャー一覧[8]

Figure 2 Multi-touch gestures.

マルチタッチ機能を搭載したテーブル型ディスプレイでは、マルチタッチを認識することは出来る。しかしながら、通常、タッチ操作、特にマルチタッチジェスチャーを想定していない、PC上で機能するアプリケーションを、マルチタッチジェスチャー操作に対応させるようにするには、複数のタッチ操作が、マルチタッチジェスチャー操作であるかどうかの判別が必要となる。その判別を行うための情報として以下の4つが挙げられる。

1. マルチタッチジェスチャーの時間
2. マルチタッチジェスチャー時の指の動作向き
3. マルチタッチジェスチャー時の指の間隔
4. マルチタッチジェスチャー時の回転角度

Windows OS 上でのタッチ操作の認識において、タッチパネル上に指が触れると、TOUCH_DOWN イベントがアプリケーションに送られる[9]。1については、そのTOUCH_DOWN イベントの時間差を求め判別を行う。2のマルチタッチジェスチャー時の指の動作は、つまんだり広げたりする動作で、それぞれの指に対して反対向きに動かしていることから、マルチタッチジェスチャー時の指の動作向きにより判別を行う。3の指の間隔は、例えば、ピンチ操作時には指をつまんだり広げたりすることによって、指の間隔が長くなったり短くなったりする。この間隔を計測することによって、判別を行う。4の回転角度については、回転動作時の指の動作は、それぞれの指に対して反対向きに動かしている。それぞれの指を動かしている角度を求め、適切な向きに回転動作の判別を行う。

マルチタッチジェスチャーの時間

マルチタッチジェスチャーを行っている時の、タッチしている指によって TOUCH_DOWN イベントが送られた時間が得られる。その時にタッチされている複数の指の TOUCH_DOWN の時間差を利用して判別を行う。アプリケ

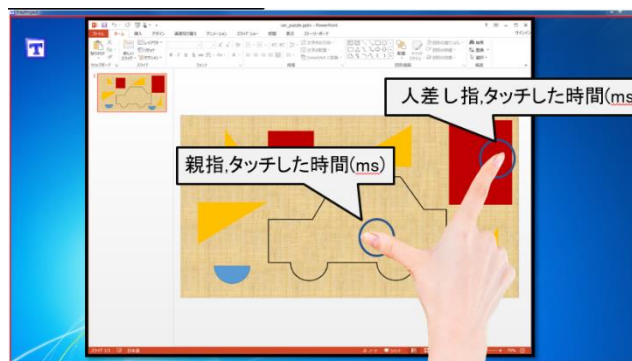


図3 TOUCH_DOWN の時間差の例

Figure 3 Example of a time difference between two TOUCH_DOWN events.

ーションに対して2本の指でタッチを行った際にそれぞれのタッチされた指で、タッチされた時間が出てくる様子を図3に示す。マルチタッチジェスチャーを行う際の、それぞれの指の TOUCH_DOWN の時間差をはかり、それをもとに判別を行う。先行研究[10]では、その差が50ms以内であれば、マルチタッチジェスチャーとして認識すると適切であると求められている。

マルチタッチジェスチャー時の指の動作向き

マルチタッチジェスチャー操作を行なっている時に、タッチしている指の動きについて、動かしている方向（角度）を8方向に分けて判別を行う。判別を行う方向を図4に示す。マルチタッチジェスチャー操作を行なっている時の指の動作方向は、摘んだり広げたり回転する動作から、それぞれ反対方向へ動かすことが基本となっている。

マルチタッチジェスチャー操作時の1つの指の方向に対して、別の指が反対方向の3つの方向に動かしていれば、マルチタッチジェスチャー操作と判別する。例えば、1本目の指が右方向（図4の2の方向）に動かした場合に、2本目の

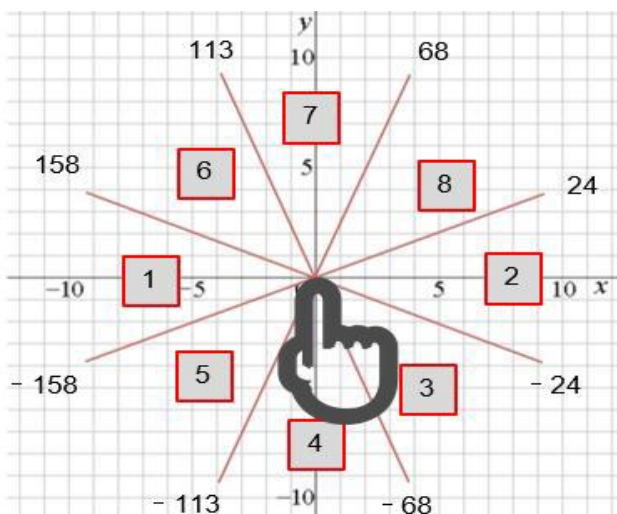


図4 指の動作向き

Figure 4 direction of movement of the fingers TOUCH_DOWN events.

指が左・左上・左下の3方向(図4の1,5,6の方向)に動かしているとマルチタッチジェスチャ操作と判別する. 上記の動作方向によりジェスチャの種類についても判別を行う.

マルチタッチジェスチャ時の指の間隔

図5に示すように, ピンチ操作の動作は, アプリケーションウィンドウ上のオブジェクトを摘んだり広げたりすることから, ピンチ操作をしている指の間隔は長くなったり短くなったりしており, それぞれピンチインとピンチアウトに対応している. TOUCH_DOWN イベントやタッチした指の移動情報をアプリケーションに伝える TOUCH_MOVE イベントなどのイベントは, タッチ操作をしている指ごとに同じ時間で送信されているので, 指の間隔を比較する時間は同じ時間となり, その時間内で2つの指の間隔が短くなればピンチイン, 長くなればピンチアウトと判別する.

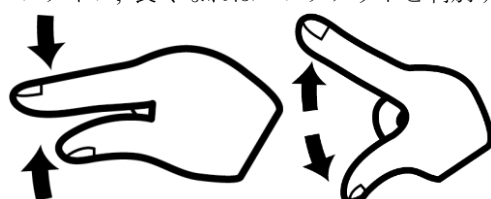


図5 ピンチアウトとピンチイン

Figure 5 Pinch out and pinch in.

マルチタッチジェスチャ時の回転角度

マルチタッチジェスチャ操作における回転操作について, 回転動作時の指はほとんど間隔を変えずに捻るような動きである. そのため指の間隔の変化は判別には有効でない. それ故, 指の動作角度を求め, 回転動作の判別を行う. 指の動かしている方向(角度)を利用して, 角度の差を取り, 回転動作を行っているのか判別する.

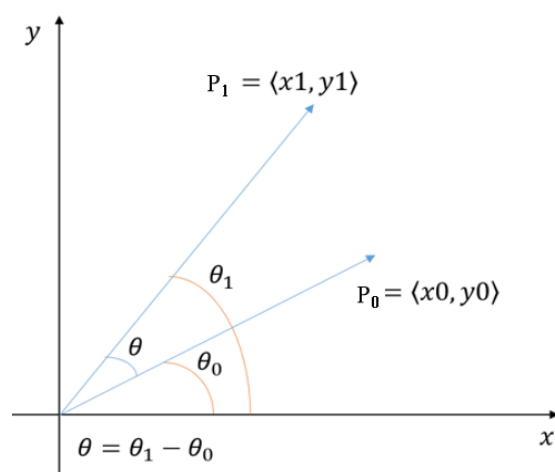


図6 回転角度

Figure 6 Rotation angle.

図6を例に説明する. 1つの指が P_0 から P_1 に移動したとする. その時に得られる角度は θ_0 と θ_1 である. この差を求めることで得られる角度は θ であり, この角度が正で

あれば反時計回り, 負であれば時計回りに回転動作を行っていることが求められる. このように回転角を求めることで指がどの向きに回転動作を行ったか求めることができる.

以上の4つの判別をもとに, マルチタッチジェスチャ操作を特定しアプリケーションへの反映を行う. タッチ操作を想定していないアプリケーションでは, OS からのタッチイベントを直接処理できない. そのため, 代替するショートカット用のキーイベントを送信することで, マルチタッチジェスチャ操作を実現する.

複数のアプリケーションを利用している場合では, マルチタッチジェスチャ操作を判別した上で, 適切なアプリケーションへとイベントを送信する必要がある. 以下では, シングルユーザ向けアプリケーションを複数人で同時に操作する「同時操作制御機構」と, 複数アプリケーションを同時操作できる「複数アプリケーション制御機構」の2つによるイベントの収集と管理の方法を説明する. 提案する, 同時マルチタッチジェスチャ機構は, 判別したマルチタッチジェスチャ操作に対応するキーイベントを, これら2つの機構を利用して, アプリケーションへ送信するものである.

3.2 同時操作制御機構

まず, シングルユーザ向けアプリケーション上で複数人が同時に操作できる同時操作制御機構のしくみについて述べる. 本来, 一人のユーザのみが操作可能なアプリケーションに対し, 複数ユーザで操作するためのタッチイベントの収集と管理, 反映の流れを述べる.

タッチイベントの収集と管理

複数のタッチイベントが同時に発生した場合には, 収集を行う際に, イベントの一連の操作を関連づける必要がある. 指が触る, 動かす, 離す, のタッチ操作に対して, TOUCH_DOWN, TOUCH_MOVE, TOUCH_UP までの一連のタッチイベントが生じ, これらに共通のタッチ ID が割り当てられる. このタッチ ID を用いて同時操作制御機構では, 同時に発生する複数のタッチイベントから一連のタッチ操作の識別を行っている. また, このタッチイベントには, タッチ ID の他にイベントの種類や, 発生した座標などの情報を含んでいるため, これらを利用して該当するアプリケーションへ送信する.

収集したタッチイベントは, 同じ ID のものを同じイベントキューに格納する. 異なるタッチ ID を持つタッチイベントが発生した場合には, 新たなイベントキューを作成し格納する. これにより発生したタッチイベントを一連のタッチ操作に合わせて正常に分けることが可能となる. 図7に発生したタッチイベントをイベントキューへと振り分けるイメージを示す. これは, タッチ操作が二つ同時に発

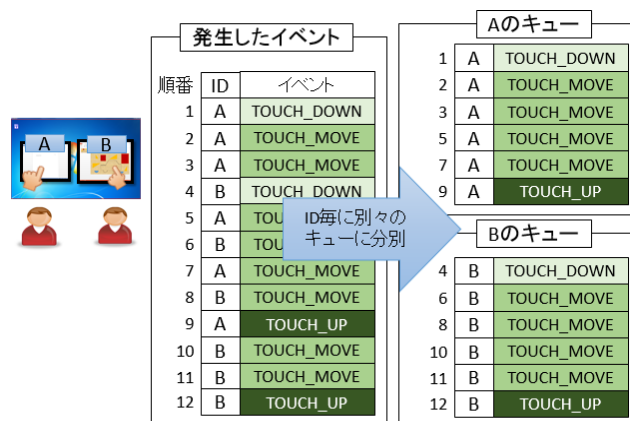


図7 イベントをキューに分けるイメージ

Figure 7 Examples of event queues.

生じたイメージで、中央のイベントのリストは、発生した順にタッチイベントを示している。右側は、それをタッチID毎のキューに振り分けている状態を示している。この時、タッチIDはAとBの二つ発生している。これらをタッチID毎にタッチイベントを振り分けてキューへと格納し、さらに、該当するアプリケーションへ送信するが、このままそれぞれのキュー内のイベントを発生順に送信してしまうと、他のユーザのタッチと混同してしまう可能性がある。

タッチイベントの送信

対処として、ユーザの一連のタッチ操作の開始と終了に対応する TOUCH_DOWN と TOUCH_UP を、実際にユーザが行った操作のイベントとは別に挿入して送信する。TOUCH_DOWN を挿入するときは、送信が途中で止まっているキュー内のイベントの送信を一時的に止める時に行う。そして、挿入するそれぞれのタッチイベントの座標は、正常にイベントを送信するためにキューを切り替える際の直前のイベント、直後のイベントの座標と同一とすることで、座標のズレが起こらないようにする。

図8に以上の処理の流れのイメージを示す。図8の左側のAのキューとBのキューは、タッチIDがAとBとしてタッチイベントを振り分けている状態となっており、この2つのキュー内のイベントは、上から発生した順に並んでいる。発生した順にAとBの切り替えを行いながら、該当するアプリケーションに送信する際に、例えば、Aのイベント TOUCH_MOVE の直後に同じ座標の TOUCH_UP を挿入し、Bのイベント TOUCH_MOVE の直前に同じ座標の TOUCH_DOWN を挿入して、AとBのキューそれぞれに該当するアプリケーションに送信する。この処理の後、同様の処理を行うことで、全ての一連のタッチ操作をそれぞれの該当するアプリケーションへ反映させる。

3.3 複数アプリケーション制御機構

複数のアプリケーションを同時に操作できる複数アプ

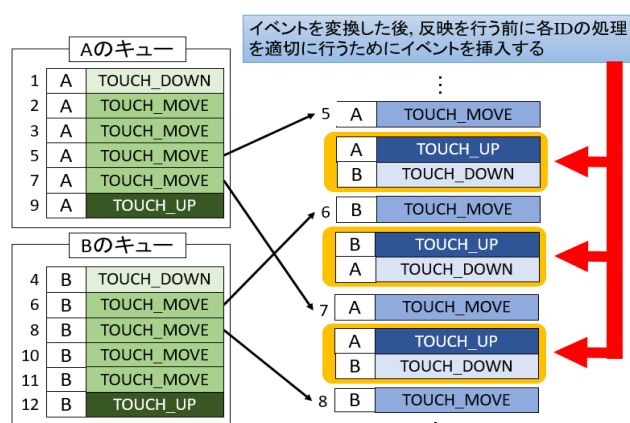


図8 二つのキューを切り替えるイメージ

Figure 8 Adjustment of events on two queues.

リケーション制御機構の仕組みについて述べる。Windows上では、複数アプリケーションを同時に操作することができないため、コラボトレイにアプリケーションを乗せることでタッチイベントの収集と管理、反映を行う。

タッチイベントの収集と管理

複数のアプリケーションを同時に操作した場合、同時操作制御機構と同様にタッチイベントの収集を行っていくが、発生したタッチイベントの情報だけでは、それぞれのアプリケーションへと正常にイベントを送信させることができない。なぜならば、同時操作制御機構は一つのコラボトレイ上で発生したタッチイベントの情報を収集、管理するため、複数のコラボトレイを用いた際には、タッチIDと座標だけでは正常な送信が行われなかったためである。それゆえ、この複数アプリケーション制御機構ではタッチイベントの情報に加え、それぞれのアプリケーションおよび、コラボトレイのハンドル情報を利用している。

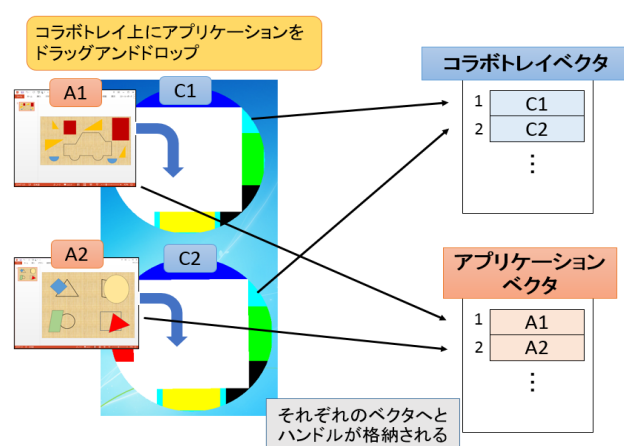


図9 ベクタへとハンドルを格納

Figure 9 A CollaboTray vector and an application vector.

起動したコラボトレイ上に既存のアプリケーションをドラッグアンドドロップした時、アプリケーションのハンドル（以下、アプリケーションハンドルと表記する）をア

アプリケーションのハンドルを格納するベクタ（以下、アプリケーションベクタと表記する）へ、コラボトレイのハンドル（以下、コラボトレイハンドルと表記する）をコラボトレイのハンドルを格納するベクタ（以下、コラボトレイベクタと表記する）へ、それぞれ格納する。こうすることで、それぞれのベクタに格納された順にアプリケーションとコラボトレイが対応する。これらの流れを図9に示す。

図9の中では簡易的に1つ目のアプリケーションハンドルをA1, 1つ目のコラボトレイハンドルをC1と表記し、2つ目以降のアプリケーションハンドルやコラボトレイハンドルをそれぞれA2, C2と表記してある。また、POINTER_INFO構造体を用いて、この構造体に何らかのイベントが発生した際に情報を取得することで、その情報を使用している。その情報の中にイベントが発生したウィンドウのハンドルの情報が含まれているため、上記のハンドル情報と同時に用いている。

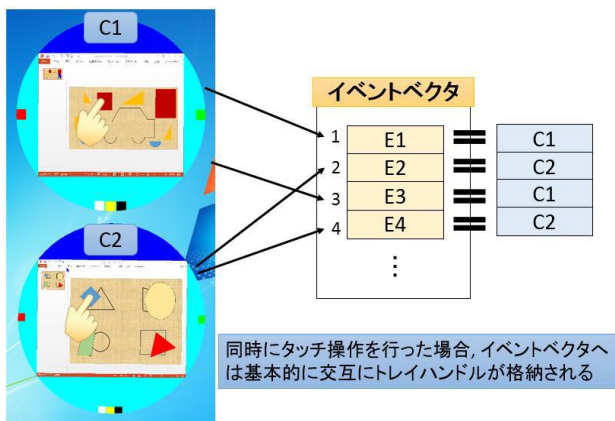


図10 タッチ時のイベントベクタへの格納
Figure 10 Storing touch events.

それぞれのコラボトレイ上でタッチ操作を行った際の流れのイメージを図10に示す。ここでは、2つのアプリケーションを使用した際の流れを表している。まず、同時操作制御機構と同様にタッチイベントの収集を行う際に、タッチイベントが発生したコラボトレイハンドルを取得し、イベントが発生したコラボトレイハンドルを格納するベクタ（以下、イベントベクタと表記する）に格納する。ここでは簡易的にイベントベクタに格納された順にE1, E2, ...と表記する。これらのハンドルはコラボトレイハンドルであるので、コラボトレイベクタと要素が同じとなる。この場合は、2つのコラボトレイ上で同時にタッチ操作を行っているため、イベントベクタに格納されるのはC1, C2, C1, C2, ...という順番になる。

タッチイベントの送信

収集したタッチイベントを正しいアプリケーションへ送信するための問題点として、実際にイベントをアプリケ

ーションへと反映させる際にコラボトレイハンドルだけでは正常な反映が行われないことが挙げられる。ユーザがテーブル型ディスプレイにタッチした際に、タッチイベントが発生するのはコラボトレイ上なので、取得されるウィンドウハンドルはコラボトレイハンドルとなるが、実際にイベントを反映させるのはアプリケーションなので、アプリケーションハンドルへと置換を行う必要がある。前述のアプリケーションベクタとコラボトレイベクタを用いる。これらのベクタに格納されたそれぞれのハンドルA1, ...とC1, ...は対応しているため、A1のアプリケーションはC1のコラボトレイ上にドラッグアンドドロップされているものとなる。よって、反映させる際に、まずイベントベクタに

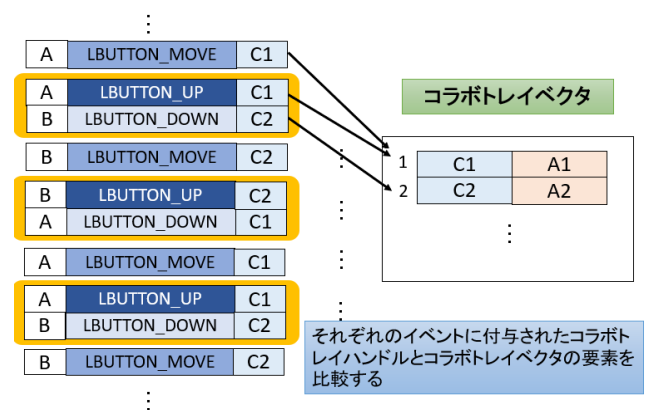


図11 ハンドルの比較
Figure 11 Handle comparison.

格納されたコラボトレイハンドルとコラボトレイベクタに格納されたコラボトレイハンドルを比べる。この流れを図11に示す。

一致した場合、対応したアプリケーションベクタから参照したアプリケーションハンドルを用いて、それぞれのアプリケーションへとイベントを送信する。この流れを図12に示す。送信の際に、3.1に述べたように、マルチタッチジェスチャ操作に対応した一連のイベントと判別されれば、代替するショートカット用のキーイベントを送信することで、マルチタッチジェスチャ操作を実現する。

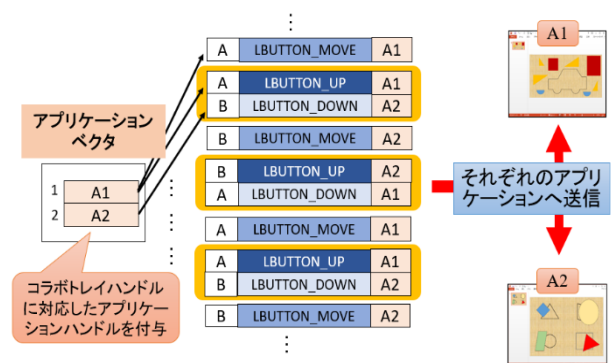


図12 正しいアプリケーションへと送信
Figure 12 Sending events to corresponding applications.



図 13 実験風景

Figure 13 An experimental collaborative work using two CollaboTrays.

4. 実験

同時マルチタッチジェスチャ操作機構が協調作業に有効かどうか考察を行うために、被験者実験を実施した。

4.1 実験概要

実験は、被験者が 2 人 1 組で行い、コラボトレイに 2 つのアプリケーションを載せ、特定のタスクをしてもらった。被験者は大分大学の学生 10 名で、特定のタスクとは、Google Chrome と MS PowerPoint を載せたそれぞれのコラボトレイ上で、二人でそれぞれ操作を行いながら、Google Chrome では特定のサイトへの遷移、PowerPoint ではパズルの完成を行なってもらおうというものである。パズルのピースは拡大・縮小・回転が必要であり、被験者は適宜マルチタッチジェスチャ操作を行う必要がある。実験の様子を図 13 に示す。被験者は、タスク終了後、表 1 に示す 6 つのアンケートに対して 7 段階のリッカート尺度 (7:非常に同意できる, 6:同意できる, 5:やや同意できる, 4:どちらでもない, 3:やや同意できない, 2:同意できない, 1:全く同意できない) で回答してもらった。

表 1 アンケート項目

Table 1 Questionnaire items

Q1. マルチタッチジェスチャ (ピンチ) は意図した通りに操作できたか
Q2. マルチタッチジェスチャ (回転) は意図した通りに操作できたか
Q3. マルチタッチジェスチャ (スクロール) は意図した通りに操作できたか
Q4. タッチイベントの反映は適切に行われたか
Q5. 協調作業はスムーズに行えたか
Q6. 総合的に評価して、本機構は協調作業に適しているか

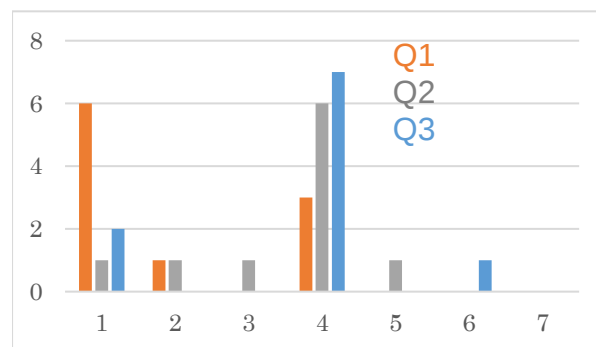


図 14 アンケート結果 Q1~Q3

Figure 14 Questionnaire results for Q1~Q3.

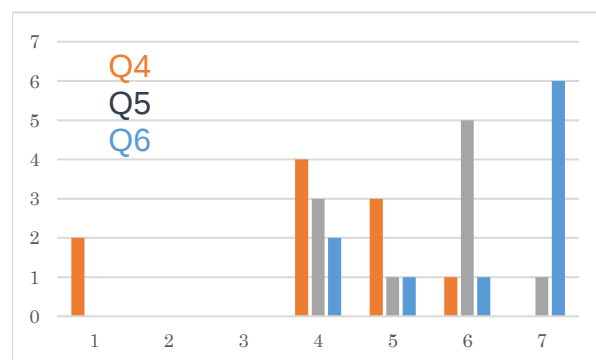


図 15 アンケート結果 Q4~Q6

Figure 15 Questionnaire results for Q4~Q6.

4.2 考察

図 14 に Q1~Q3 のマルチタッチジェスチャ操作に関する質問の結果、図 15 に Q4~Q6 の質問の結果を示す。はじめに、Q4 のタッチイベントの反映に関する質問について、どちらかといえば肯定意見 (評価 5,6,7) が多かったが、否定意見 (評価 1,2,3) も得られているため、タッチイベントの反映が適切に行われていない状況があり、マルチタッチジェスチャにも影響していると考えられる。Q1 から Q3 への回答を見ると、多くの意見として“どちらでもない”が得られている。この意見より、マルチタッチジェスチャ操作は出来ているが、タッチイベントの反映が適切に行われていないために、意図した通りに操作が出来たとは言えないという意見となっていると考えられる。

Q1 のピンチ操作に関する回答は、否定意見が多かった。ピンチ操作のイベントの反映が行われていない場面が多くみられたことが原因である。Q2 の回転操作と Q3 のスクロール操作については、いずれも有意差は見られなかった。各マルチタッチジェスチャ操作はある程度反映出来ていたことによるものだと考えられる。Q5 については、有意に肯定意見が多く、協調作業を行っているという観点でみるとスムーズに行えていると感じる被験者が多かった。しかしながら、動作がスムーズになればより良いという意見も得られたため、更なる改善が必要である。最後に Q6 につ

いては有意に肯定意見が多く、同時マルチタッチジェスチャ操作を行うことが出来る本機構は、マルチタッチジェスチャ操作が意図した通りに出来ていない場面があった場合でも、協調作業を支援し促進に繋げることが出来ると考えられる。それ故、イベントの反映を適切に行い、動作をスムーズにすることで更なる協調作業の促進に繋がれると考えられる。

5. おわりに

本研究で提案した同時機構は、シングルユーザ向けの任意のアプリケーションを使って、マルチタッチが可能なテーブル型ディスプレイでの協調作業を可能にものである。従来研究の協調作業支援環境であるコラボトレイにおける同時操作制御機構と複数アプリケーション制御機構を利用し、マルチタッチジェスチャ操作の判別の仕組みにより、複数ユーザが同時にマルチタッチジェスチャを行って、複数のアプリケーションを同時に操作することができる。被験者実験におけるアンケートから、同時マルチタッチジェスチャ操作機構は協調作業に適していることを確認し、これまでよりも円滑な協調作業環境が実現できた。

今後の課題としては、スムーズな操作の実現には、マルチタッチジェスチャ操作に対応するイベントのアプリケーションへの送信遅延を改善する必要がある。

謝辞

本研究の一部は、JSPS 科研費 17K00274 の助成を受けたものです。

参考文献

- [1] Y. Abe, K. Matsusako, K. Kirimura, M. Tamura, M. Nakashima, and T. Ito, "Tolerant sharing of a single-user application among multiple users in collaborative work," Companion Proc. CSCW2010, pp. 555-556, Savannah, GA, February 2010.
- [2] 武谷充謙, "マルチタッチテーブル上での複数ユーザによるアプリケーションの同時操作を円滑にする作業環境の構築", 大分大学工学研究科修士論文, 2016.
- [3] 松本京介, "マルチタッチテーブル上での協調作業を促進する複数アプリケーション制御機構", 大分大学工学研究科修士論文, 2019.
- [4] <https://onedrive.live.com/about/ja-jp/>
- [5] <https://products.office.com/ja-jp/office-online/documents-spreadsheets-presentations-office-online>
- [6] https://www.google.com/intl/ja_ALL/drive/47
- [7] TeamPlayer3, <https://www.dicolab.com/>
- [8] https://jp.freepik.com/free-vector/touch-gestures-icons_1537226.htm
- [9] <https://developer.mozilla.org/ja/docs/Web/API/TouchEvent>
- [10] 藪井淳史, "テーブル型ディスプレイを用いた協調作業を促進するシングルユーザ向けアプリケーションにおけるマルチタッチジェスチャ操作の実現", 大分大学工学部学士論文, 2018.
- [11] https://docs.microsoft.com/en-us/windows/desktop/api/winuser/ns-winuser-tagpointer_info