Vol.2010-HCI-140 No.10
Vol.2010-UBI-28 No.10
2010/10/29

モバイル向け UI 操作省力化

川崎仁嗣[†]　菊地悠[†]　大久保信三[†]　稲村浩[†]

ユーザがモバイル機器において繰り返し行う操作を支援するための, UI 操作省力化手法を提案する．本手法は，ユーザの操作履歴を操作列に分割した上で，ユーザが次に行う可能性が高い操作列を推定，候補として提示し，候補選択が行われた場合に，ユーザに代わって操作列を自動実行する．また，ユーザの候補選択状況から，一度により多くの操作を実行できるよう操作列を連結する．本手法を用いたユーザ評価を実施し，繰り返し操作における操作時間を約 30～50%短縮できていることを確認した．

An UI Automation Framework for Mobile Devices

Satoshi Kawasaki[†], Haruka Kikuchi[†], Shinzo Ohkubo[†] and Hiroshi Inamura[†]

We propose an UI automation framework that reduces repetitive operations for mobile devices. This splits operation history to operations chunks and predicts next operation chunks, and displays predicted candidates, and automates operations if the candidates is selected. Then, the framework concatenates some chunks based on users' selection of candidates to automate more operations at a time. We performed a user evaluation, and obtained a result that this method reduces the operation time in about 30～50%.

1. はじめに

今日のモバイル機器，特に携帯電話機においては音声通話やウェブの閲覧，メールの送受信に限らず，電子決済機能や動画・音楽再生など機能面での成熟化が進んでいる．このような市場環境においては，電話やメールのような基本的な機能においてはどの端末を選んでもほとんど差がない状況となり，単なるコスト競争に陥りがちである．これを避けるため，キャリアや端末ベンダーは次々と新たな機能やアプリケーションを追加してきた．このような流れの中で，現在の携帯端末に搭載される機能やアプリケーションは非常に多岐にわたっており，一度も使わない機能があるというユーザも想定される．さらに，機能やアプリケーションの追加が進むことにより，ユーザが必要な機能を呼び出す操作手順などが以前に比べ煩雑になっている．例えば，サブメニューの項目が増えることや設定の階層が深くなることにより，目的の操作を達成するまでの操作回数が増加するなど，使い勝手の低下を招いてしまっている．本来はユーザに提供する付加価値を向上させるために機能やアプリケーションの追加が行われているにも関わらず，操作が煩雑化することで使い勝手が低下し，他社との差別化要因としての優位性も低下してしまっては機能追加の効果が薄くなってしまう. 特に，ユーザが頻繁に触れる機能やアプリケーションにおいて操作の手間が大きいことで，端末全体の評価が下がってしまう可能性は高い.

そこで我々は, ユーザが繰り返し行う操作を支援するための UI 操作省力手法を検討している．この操作支援により，ユーザはよく行うタスクを，より少ない操作手順や操作時間で完了できるようになる．タスクを達成するまでには，図 1 に示すように，目的のアプリケーションを起動する操作列と，アプリケーション内で目的とする機能を実行する操作列とに分かれるが，本研究ではどちらの操作についても省力化が行える方法を検討した．

本稿では，ユーザ操作の学習方法と学習結果を元に以降の操作を推定する方法について提案を行う．また，提案手法の有効性についてユーザ評価を実施したので，その結果についても述べる．

[†] 株式会社 NTT ドコモ　先進技術研究所
Research Labs, NTT DOCOMO,INC

Vol.2010-HCI-140 No.10
Vol.2010-UBI-28 No.10
2010/10/29

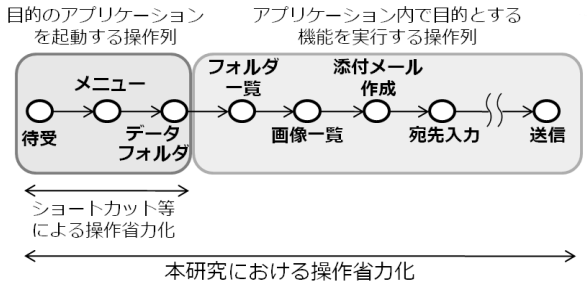

図 1. 目的達成までの操作
Fig 1. Operations for achieving a task

2. UI 操作省力化

我々が検討する UI 操作省力化では，ユーザの普段の操作履歴を学習しておき，今のユーザの操作から今後行うと予想される操作を提示して，ユーザがその候補を選択すると操作を自動的に行う方法を実現する．

2.1 ユースケース例

アプリケーション操作の省力化機能を，ケータイのカメラで写真を撮影し，ブログに投稿する場面を想定して説明する．

A さんは毎日，夕食をカメラで撮ってメールで送信することでモブログ投稿を行っている．A さんの普段の操作履歴を常に取得しており，カメラ機能での撮影モード切り替えやメール添付送信の操作が頻繁に行われていることも履歴として取得されている．

操作フロー

(1) 料理を撮影するため，カメラを起動させる．

(2) 料理モードへ切り替えようとサブメニューを開くと，サブメニューの上部に操作予測結果が 3 つ表示されており，その中に A さんがいつも行っている操作に対応して“シーン(料理)→撮影サイズ(WXGA)” が表示されていた．この項目を選択すると，自動的にサブメニューから“シーン”の設定画面へ移り，“料理”の項目が選択され，さらにサブメニューから“撮影サイズ”の設定画面へ移り，“WXGA”の項目が選択された．

(3) A さんは撮影ボタンを押し，写真を撮影／保存した．

(4) メール添付送信を行うためにサブメニューを開くと，サブメニューの上部に操作予測結果として“(写真一覧) メール添付”が表示されている．この項目を選択すると，自動的に“写真一覧”が選択され写真一覧に遷移し，最新の写真が選択され，“メール添付”項目を選択する操作まで自動的に実行され，メール作成画面に遷移した．

(5) A さんは本文などを入力して送信した．

なお，操作フロー（1）～（5）の一連の操作のように，ある最終目標を達成するまでの一連の順序つき操作列をタスクと呼び，”(1)カメラ起動”や”(2)シーン選択”といった操作のように，タスクを構成するより細かい単位の操作列のまとまりをサブタスクと呼ぶ．

2.2 要件

我々は UI 操作省力化を検討するうえで，以下の 3 つの要件を定めた．

ⅰ．複数のアプリケーションに対する操作を自動化する

ある特定のアプリケーション内の操作のみを自動化するのではなく，端末に搭載されている種々のアプリケーションの操作を自動化する．また，アプリケーション間をまたがる操作も自動化の対象とする．

ⅱ．意味的に関連のある複数の操作（サブタスク）を自動化する

自動化する操作は，“メニューを開く”という単一の操作ではなく，意味的に関連のある，“メニューを開いて設定画面から項目を選ぶ”までの操作列をサブタスクとして提示する．単なる任意長の操作列に比べ，サブタスクのように意味的にまとまりのある操作列を提示した方が，ユーザはその候補によりどのような処理が実行されるのかを把握しやすくなり，他の操作列と組み合わせる際の再利用性も高くなる．

ⅲ．いつも同じ組み合わせの操作列（サブタスク）は一度にまとめて自動化する

要件 ii により複数の操作からなるサブタスクが候補として提示されるが，より省力化を図るために，毎回同じ組み合わせで行われるサブタスク同士をまとめて一つの候補として提示する．

3. UI 操作省力化機能の設計

前章にて提案を行った UI 操作省力化機能の設計について述べる．

3.1 提供形態

要件 i に対応するため，UI 操作省力化機能を図 2 に示すように，各アプリケーションから分離されたフレームワーク構成として設計した．これにより，アプリケーション内に操作省力化機能を持たなくとも，端末内に本フレームワークを組み込むこと

Vol.2010-HCI-140 No.10
Vol.2010-UBI-28 No.10
2010/10/29

で，各アプリケーションの操作が省力化の対象となる．

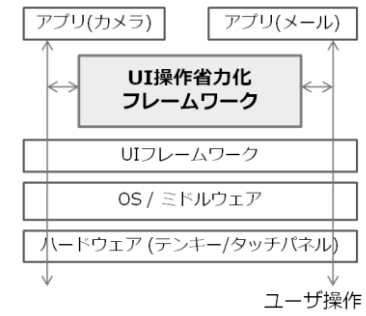

図 2.　UI 操作省力化フレームワークを搭載した
ソフトウェアスタック
Fig 2. Software stack with an UI automation framework

3.2 機能構成

UI 操作省力化フレームワークの機能構成を図 3 に示す．操作履歴を取得し，頻繁に行われる操作を学習する処理と，現在の操作に基づいて今後の操作を推定・実行する処理に分かれ，それぞれ以下に示す要素から成る．なお，学習・予測の処理はそれぞれ独立しており，学習処理は 1 タスク完了時点で実行されるが，予測処理は画面遷移の度に実行される．

1．ユーザの操作履歴を取得する機能
2．操作履歴をサブタスク単位で分割する機能　　学習処理（事前）
3．操作履歴より操作列同士の相関性を解析する機能
4．後続操作の候補を推定する機能
5．選択された候補に基づいて，UI 操作を自動実行（操作履歴再現）する機能　　予測処理
6．候補の選択状況に応じて，操作列同士を連結する機能　　学習処理（事後）

操作履歴を取得するにあたって，候補推定の精度向上のためにユーザ操作からより多くの情報を収集できる手法が有利である．そこで，単なるキー操作だけではなく，アプリケーションの種類や状態などの情報を収集するための仕組みとして我々は UI フレームワークと呼ばれる技術に着目した．本研究では操作履歴の取得において，UI フレームワークの機能を用いることとした．

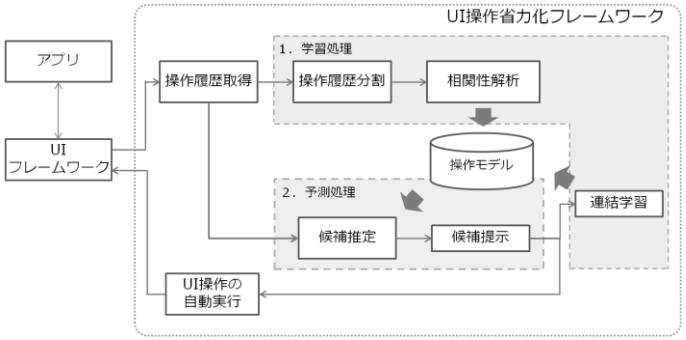

図 3．UI 操作省力化フレームワークの全体構成
Fig 3. Overview of an UI automation framework

3.3 操作履歴の取得

要件 ii に示した通り，候補の提示においてはサブタスク単位で行うこととした．サブタスク単位の操作列を得るためには，ユーザがどのような目的で操作を行ったのか判断するための情報が必要であり，単なるキーイベントだけでは情報として不足している．そこで我々は， UI フレームワークとアプリケーションとで受渡しされる操作イベントを記録することにより，“決定キー押下”という単なるキーイベントではなく，“項目 XX の選択“という抽象化された操作イベントとして履歴を取得する方法を採用した．

UI フレームワークとはアプリケーションの UI 開発において，画面上に表示される画像やボタン，テキストフィールドなどの画面構成要素（以降，UI 部品と呼ぶ）の表示処理など定型的な処理を提供するものであり，実際に開発者が記述しなければならないコードの量を減らし，開発を効率化するための技術である．モバイル端末で用いられる UI フレームワークとしては， GTK＋7)や Qt8)などが知られている．また，Android や iPhone にも独自の UI フレームワークが搭載されている．

UI フレームワークを利用したアプリケーションの動作概要を図 4 を用いて説明する．アプリケーションは画面上にボタンを表示したい場合，ボタン上の文言や表示位置を指定して UI フレームワークが持つボタン表示処理を呼び出す．また，表示したボタンをユーザが押下すると，UI フレームワークがボタン押下を検知してアプリケーションを呼び出し，ボタン操作に対応した処理が行われる．

Vol.2010-HCI-140 No.10
Vol.2010-UBI-28 No.10
2010/10/29

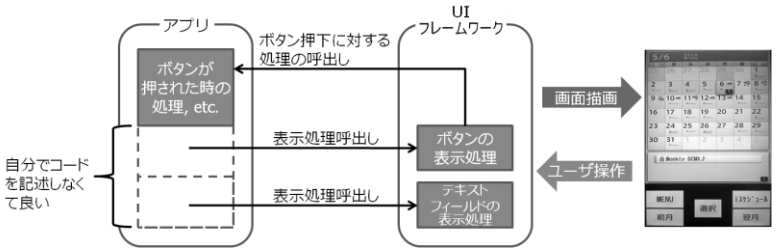

図 4. UI フレームワークの概要
Fig 4. Overview of a UI framework

キーイベントに対する画面状態の変化や UI フレームワークから取得した操作履歴の例を図 5 に示す.

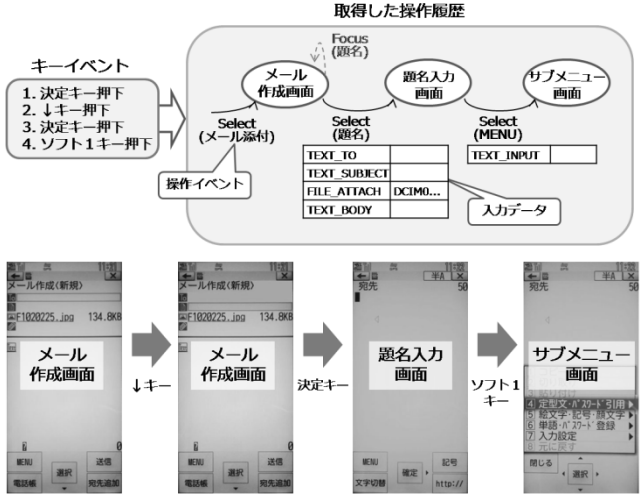

図 5. キーイベントに対する画面状態の変化と
UI フレームワークから取得した操作履歴
Fig 5. Comparison of screen transitions from key events,
operation histories taken from a UI framework

また，操作履歴から同じ挙動を再現するためには，チェックボックスやテキストフィールドに入力したデータも保持する必要がある．従って，①現在の画面状態，②画面遷移発生時の操作イベントと入力データ,③遷移先の画面状態 の３つを操作履歴として記録する．画面状態とは，画面上に表示されている UI 部品の一覧である．なお，これら３つの情報は，一般的な UI フレームワークの持つ機能で取得することが可能である．

3.4 操作履歴の分割

操作履歴の分割処理には，タスク単位への分割とサブタスク単位への分割の 2 種類がある．

タスク単位への分割

操作履歴分割処理は，ユーザが一つのタスクを完了した後に行う．タスクの分割における終端記号の考え方を示す．タスクとはユーザがある最終目的を達成するまでの一連の操作であり，ケータイの場合は待受画面から操作が始まり全ての作業が終わると終話キー等を押して再び待受画面に戻ってくることになるため，次に待受画面が現れるまでの操作履歴を蓄積し,待受画面に戻った時点で以降の学習処理を行う.また，タスク切り替え画面の起動はユーザが明示的にタスクの切り替えを行っている操作なので，以降の操作は別のタスクとして取り扱う．さらに，メール着信時等の通知画面はユーザがあるタスクを実行中に割り込みして表示され，その通知内容により以降のタスクが変化すると想定されるため，以降の操作は別タスクとして扱う．従って，(1) 待受（ホーム）画面，(2) タスク切り替え画面，(3) メール等の通知画面の 3 つの画面状態をタスクの終端記号とした．

サブタスク単位への分割

要件 ii で示すように予測候補をサブタスク単位とするため，さらにサブタスクに対応した操作列へと分割する必要がある．しかし，図 6 に示すように，一連の操作履歴列のうち，どこからどこまでがサブタスクに対応する操作列であるかを判断する方法が必要となる．本研究では以下の 2 種類の分割手法を適用した．

1）ハブとなる画面状態の検出
アプリが図 7 に示すように，起動直後の画面を起点としたハブ構造の画面遷移であることが多いことに着目し，ハブとなる画面状態を見つけ，そこから再びハブ状態に戻るまでの一連の操作を，サブタスク単位の操作列であるとみなす．つまり，操作列中に複数回出現する画面状態を区切りの識別子として利用する．

2）動的コンテンツを表示する UI 部品の検出
カメラアプリにおける撮影操作や動画再生アプリにおける再生・停止操作は，タイミングが毎回異なる操作であると考えられ，自動実行してはならない操作である場合が多い．一般的に，次の操作タイミングが毎回変化するような場面は，画面内のコンテンツが動的に変化する場合が想定される．従って，動的に変化する

Vol.2010-HCI-140 No.10
Vol.2010-UBI-28 No.10
2010/10/29

コンテンツを表示する UI 部品が含まれている画面状態を区切りの識別子として利用する．

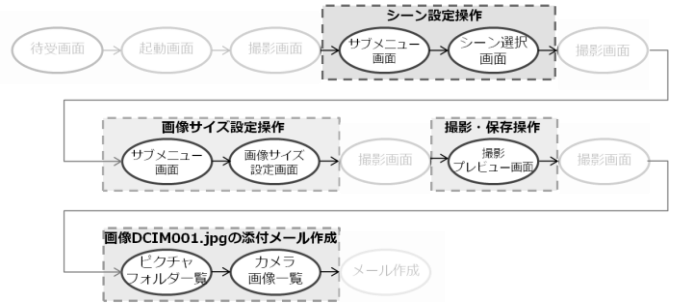

図 6．サブタスクに対応する操作列
Fig 6. Operations sequence of subtasks

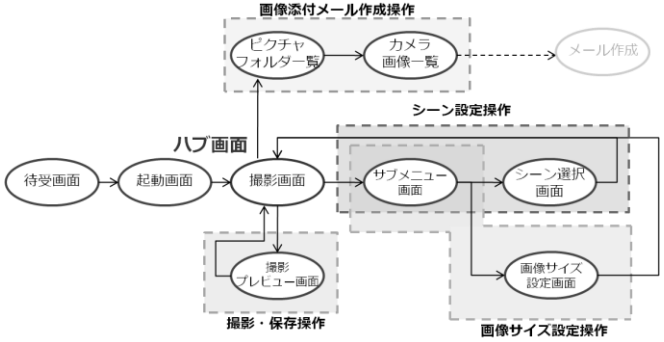

図 7．アプリの画面遷移構造
Fig 7. A constitution of screen transitions in an application

1 タスクごとになった操作履歴列をサブタスクに対応する操作列へと分割し，それぞれの操作列に一意な ID （例えば Tr1,Tr2,…）を付けて辞書形式として保存する．この際，全く同じ操作列であれば同じ ID を割り振る．

3.5 相関性解析

相関性解析では，1 タスクの操作履歴列に対して操作列の並びを解析し，予測処理において現在の画面状態から次に行われる操作列を推定するためのデータを生成する．操作履歴列での操作列の順序をツリー構造として，各辺に操作列が現れる確率を付すことでモデル化する．つまり，図 8 のように操作列 Tr1 の後に，Tr2 や Tr6 が現れる操作履歴列が過去にそれぞれ 1 回ずつ実行されている場合，Tr2 や Tr6 へ遷移する確率はそれぞれ 0.5 であり，P(Tr1|Tr2)=0.5，P(Tr1|Tr6)=0.5 となる．

学習処理が完了すると，モデルデータには 3.4 節で生成した辞書形式の操作列一覧と，相関性解析により生成された重み付きツリーとが含まれた状態になる．

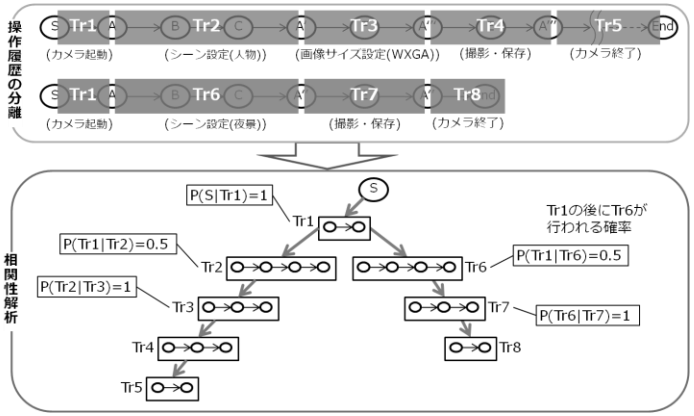

図 8．相関性解析処理の概要
Fig 8. Overview of a correlativity analysis

3.6 候補推定

候補推定処理は，ユーザの操作状況に応じてインクリメンタルに候補を提示する必要があるため，画面状態が遷移する度に行われる．候補推定処理の概要を図 9 に示す．まず，ユーザが現在までに行った操作履歴列を操作列へと分割するが，この際に，モデルデータに含まれる操作列一覧を参照して分割を行う．なお，モデルデータにない操作履歴列が現れた場合，候補推定に必要な過去の履歴がないので候補推定は行わない．

次に，操作列の並びをモデルデータ内のツリーと比較し，現在の画面状態に対応するツリー上のノードを特定する．このノードから辿ることができるノードに対応する操作列が，今後ユーザが行うと推定される操作であるため，これらのノードを候補として抽出する．抽出した候補それぞれに対して，現在の画面状態からの遷移確率を計算し，値の大きい上位候補が推定結果として提示される．

Vol.2010-HCI-140 No.10
Vol.2010-UBI-28 No.10
2010/10/29

候補の提示においては, 画面下部に候補表示領域を設けて提示する方法や, Windows におけるツールチップ表示の様に吹き出しとして表示させる方法などいくつかの提示方法が考えられる．しかし，携帯端末のように表示場所が限られている場合，候補表示のために場所を割くことは容易ではない．特に，既存のアプリケーションは画面表示領域のサイズが変わることを想定していない場合が多く，あまり現実的ではない．以上より，本稿での評価における実装では，既存のサブメニュー表示の項目として候補を提示する方法を用いた．サブメニュー表示とは，アプリケーションの実行中にソフトキーを押下することで，その時点で利用可能な機能を選択するためのポップアップウィンドウが表示され，そこから任意の機能を選択する表示方法である．この方式による候補提示のイメージを図 10 に示す.

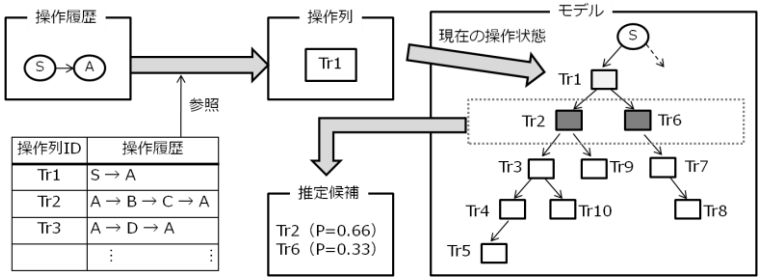

図 9. 候補推定処理の概要
Fig 9. Overview of candidates' prediction

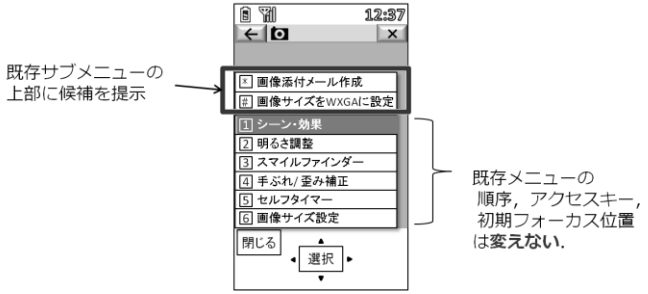

図 10. 候補提示方法の一例
Fig 10. An example of a method of displaying candidates'

3.7 UI 操作の自動実行

提示した候補をユーザが選択すると，対応する操作列をモデルデータから生成し，次の画面状態に遷移するための操作イベントをアプリケーションへ受け渡す. この際, 入力データも記録されていれば，対応する UI 部品のデータとして埋め込みを行う．例えば，テキストフィールドにユーザ ID を入力して，「ログイン」ボタンを押下するという操作履歴を再現する場合，現在の画面から対応するテキストフィールドを探し出して，入力データ用の領域にユーザ ID を書き込み，ログインボタン押下時に発生する操作イベントを擬似的に生成してアプリケーションへ通知を行う．これにより，アプリケーションはユーザが実際に操作した場合と同じように，操作イベントを UI フレームワークから受け取って処理を行う.

3.8 操作列の連結学習

要件 iii に示すように，繰り返し行う操作では複数のサブタスクをまとめて実行したい．例えば，カメラ機能の操作におけるモード設定と画像サイズ設定はまとめて一度に行えたほうが，より操作手順を減らすことができる．そこで, 図 11 に示すように, 提示した候補を連続してユーザが採用した場合，それらの候補をまとめて一度に実行しても良いと判断し，モデルデータ中の操作列同士を連結する処理を行う．もし，候補を連続では選択しなかった場合, つまり, 候補の選択後に一度ユーザが操作を行い, その後候補を選択した場合，それぞれの候補を一度に実行してもよいかどうかは判断できないため，操作列の連結は行わない.

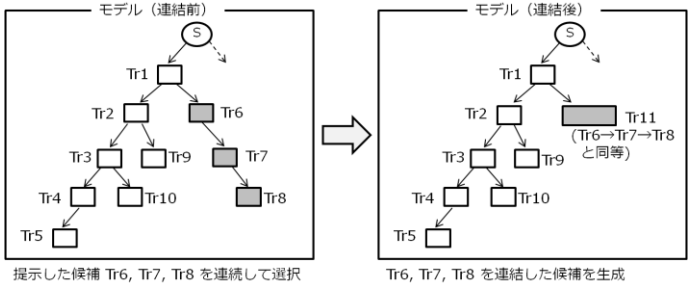

図 11. 操作列の連結学習処理の概要
Fig 11. Overview of a concatenation

Vol.2010-HCI-140 No.10
Vol.2010-UBI-28 No.10
2010/10/29

4. 関連研究・既存技術

ユーザの操作履歴を学習し,次回以降の操作を自動化する UI として例示インタフェース 1)が知られており，繰り返し行われる操作を自動化する場合に適するとされている 2)．例示により学習を行い，操作を省力化する技術としては以下のものが知られている． それぞれの手法ごとに本研究との差分を議論する．

アプリケーションの備えるマクロ機能

Microsoft Excel3)や Adobe 社の Photoshop4)が備えるマクロおよびアクション機能が有名である．これは各操作に対応するアプリ内の機能呼び出し履歴を記録する方法である.本研究では要件 i に示す通り,個々のアプリケーション内の操作列のみならず,複数のアプリケーションをまたがる操作も対象とした省力化が可能である．

キーボード/マウス操作記録型マクロ

増井らの Dynamic Macro5)や Darragh らの Reactive Keyboard システム 6)が知られている．これはキー操作を OS やミドルウェアなどから取得して記録する方法である．Dynamic Macro ではどのマクロを実行するかはユーザが指示するがマクロの始点・終点は自動的に認識する．Reactive Keyboard では入力文字列から，次の文字列を予測・提示する．本研究では，キー操作ではなく UI フレームワークから取得可能な操作イベントを操作履歴として記録する方法をとっており，設定変更によるメニュー項目の増減などが発生しても正しく動作する．

操作イベント記録型マクロ

Smart Bookmarks9)や Koala10)，DemoOffice11)，Dong らの研究 12)13)が知られている．Smart Bookmarks や Koala，Dong らの研究では，Web ページ上のボタンやテキスト入力欄に対する操作イベントを記録する．一方，DemoOffice はアプリケーションに対する操作イベントを記録する．この方法では単純にキーストロークを記録するのではなく，“決定”という名前のボタンを押す，というように抽象化した操作を記録するため，ボタンの位置が変わっても正しく動作する． Smart Bookmarks や Koala ではマクロの始点・終点をユーザが指示する．DemoOffice では，指定したデータに対する一連の操作イベントだけを自動的に抽出してマクロ化する．Dong らの研究では，推定される後続の操作全体を提示し，その中からユーザが到達したい任意の状態を選択すると，その状態までの操作を自動化する．本研究では，サブタスクに対応する操作列の区切りを自動的に認識し，ユーザの候補選択操作からサブタスクの操作列を連結する学習も行っており，ユーザは特別な操作なしにマクロの始点・終点が設定される．

5. 実装・評価

本稿にて提案を行った UI 操作省力化フレームワークについて，あるタスクを達成するまでの操作時間について評価を実施した．なお，評価においては Ubuntu Linux 10.04 の動作する PC 上において，ケータイの UI をエミュレートする評価用ソフトウェアを作成，利用して行った．

評価方法

予測なし，予測あり，予測＋連結学習ありの 3 パターンにおいて，下記の 2 タスクを実施．ケータイの操作に慣れている 10 人の被験者に対して操作時間を測定し，平均値を算出した．また，本研究で対象とするのは繰り返し操作の省略化であるため，ユーザが操作に慣れた場合での操作時間を比較する必要がある．そこで，5 回同じタスクを繰り返し，慣れに従って操作時間がどのように変化するか測定した．

タスク

1）カメラ機能を起動し，シーン(料理)，および画像サイズ(WXGA)の設定後に，撮影・保存を行い，カメラ機能を終了する
2）新規メール作成機能を起動し，宛先，題名の入力，添付ファイルの選択，本文入力を行い，メールを送信し，新規メール作成機能を終了する

なお，タスク 2 における文字入力は最短で操作が終わった場合を想定し，任意のアルファベット一文字を入力することとした．

各試行回数での操作時間を図 12 に示す．この結果は，3 回目以降の操作時間がほぼ変わらないことを示している．

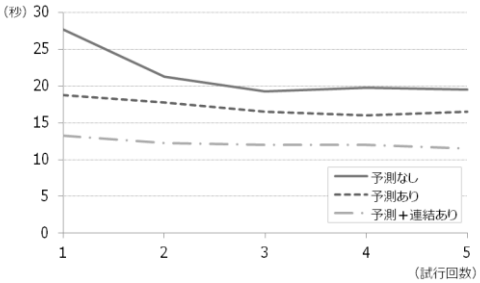

図 12. 試行回数と操作時間の関係
Fig 12. A mutuality of an operation time and number of trials

操作時間の変化が十分に少なくなった 5 回目の結果のみに注目した比較結果を図 13 に示す．いずれの場合においても，予測・連結学習ありの場合が最も操作時間が短くなっている．これは複数のサブタスクをまとめて実行していることで，ユーザが候補を選択する回数が減ったためである． タスク 1 の場合で，操作時間が 19 秒（予測

なし）から 10 秒（予測・連結あり）まで減少しており，48%ほどの操作時間削減効果があった．タスク 2 の場合で，操作時間が 24 秒（予測なし）から 14 秒（予測・連結あり）まで減少しており，40%ほどの操作時間削減効果があった．また，予測ありの評価結果は連結学習が全く行われていない状態に相当し，この場合でも 28%ほどの操作時間削減効果を確認できた．

予測ありの場合よりも予測・連結ありの方が 2 番目のサブタスク（シーン設定，宛先入力）の操作時間が長くなっているが，これは操作列の連結時に候補名も連結していることで候補名が長くなり，ユーザが候補の処理内容を認知する時間が長くなったためであると考えられる．

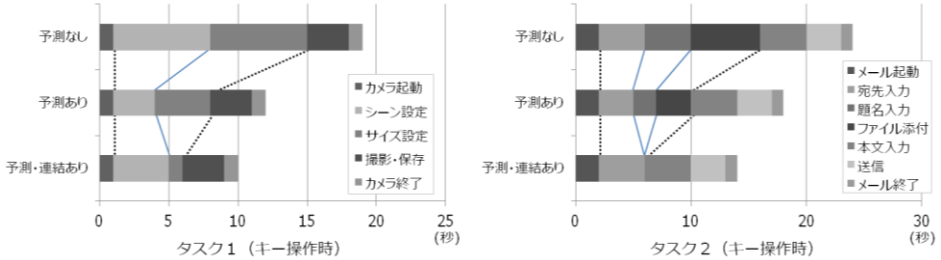

図 13. サブタスクごとの操作時間
Fig 13. An operation time per subtask

6. おわりに

本稿では UI 操作省力化フレームワークを用いたモバイル端末における操作の省力化について説明した．また，評価結果より，繰り返し行う操作については提案手法を利用することで約 30%～50%の操作時間低減が見込めることが分かった．

今後は，評価における対象ユーザや対象タスクの範囲を広げることで，タスクの種類に依存せずに操作数を削減できることを検証する予定である．

参考文献

1) Allen Cypher, Watch What I Do: Programming by Demonstration, 1993.
2) 増井俊之, “予測/例示インタフェースの研究動向”, コンピュータソフトウェア, Vol.14, No.3, pp.4−19, 1997.
3) Microsoft, Microsoft Excel, http://office.microsoft.com/ja-jp/excel/
4) Adobe, Photoshop, http://www.adobe.com/jp/products/photoshop/
5) 増井俊之，太和田誠．操作の繰返しによるマクロの自動生成．情報処理学会ヒューマンインタフェース研究会研究報告 93-HI-48, Vol.93, pp. 65-72, 1993.
6) John J, et al, The reactive keyboard: A predictive typing aid, IEEE Computer, Vol. 23, No. 11, pp. 41–49, 1990
7) GNOME Foundation, GTK+, http://www.gtk.org/
8) Nokia, Qt, http://qt.nokia.com/
9) Darris Hupp, et al, Smart Bookmarks: Automatic Retroactive Macro Recording on the Web, UIST07, 2007.
10) Greg Little, et al, Koala: Capture, Share, Automate, Personalize Business Processes on the Web, CHI, 2007.
11) Atsushi Sugiura, et al, Simplifying Macro Definition in Programming by Demonstration, UIST96, 1996.
12) Dong Zhou, et al, Optimizing User Interaction for Mobile Web Browsing, MobileHCI09, 2009.
13) Dong Zhou, et al, Optimizing User Interaction for Web-based Mobile Tasks, SAINT10, 2010.