モバイルカメラを用いたデバイス間アドホックアプリケーション共有

萩原 拓真[†1]　高嶋 和毅[†1]　モルテン フィールド[†2]　北村 喜文[†1]

概要：モバイルカメラを用いたデバイス間のアプリケーション共有を実現する手法（CamCutter）を提案する．ユーザは，ホストデバイスのディスプレイ上に表示されているアプリケーションウィンドウをモバイルデバイス上のカメラで撮影するだけで，そのアプリケーションをそのモバイル上で操作・実行することができる．CamCutter は，従来のデバイス間での情報共有の方法やツールと比べ，簡単で素早く利便性の高い情報共有を可能とするため，アドホックなミーティングや協調作業場面の利用に適している．本稿では，CamCutter プロトタイプの実装とその性能評価，およびユーザスタディにより示される有用性について報告する．

A Cross-Device Application Sharing Technique using Mobile Camera

TAKUMA HAGIWARA[†1]，　KAZUKI TAKASHIMA [†1]，　MORTEN FJELD[†2]，
and YOSHIFUMI KITAMURA[†1]

Abstract: We developed an ad-hoc cross-device interaction technique that allows the user to share an application window on a host computer by using the camera of a handheld mobile device. This technique uses our computer vision algorithm to identify the target application window from the screen capture of the host computer. The interaction flow is coherent where the user can continuously use the mobile's touchscreen to take a picture of a target window and interact with it. The interactions on the mobile device are synchronized to the original application on the host by using screen mirroring technique. We implemented a prototype and conducted a technical evaluation of its fundamental ability for desktop and meeting set-ups. A user study demonstrated that our proposal was more preferred compared with commercial screen sharing and file sharing tools.

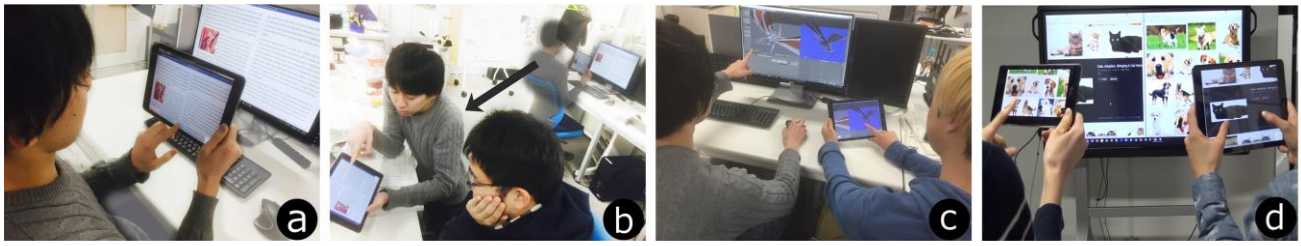


図 1 : CamCutter を用いたアドホックアプリケーション共有の例
（a）ディスプレイ上のアプリケーションをカメラで撮影（b）撮影したアプリケーションを近くの人と共有
（c）協調 3D モデリング（d）複数デバイスから大画面ディスプレイの操作
Figure 1 : Cross-device application sharing by using CamCutter

1. はじめに

電子メールを始め，リモートデスクトップやオンラインファイルホスティングサービスなど，複数のデバイスを跨いだデータやファイルの同期・共有が可能になってきた．これらの技術は主に遠隔地間を結ぶ協調作業に向けて開発されたものであり，通信やセキュリティの都合でユーザアカウント制御や共有 URL の送受信などの手順が必要である．一方で一人が複数台のデバイスを持つ現代においては，同じオフィス空間の中でも様々な形で身近なデバイス間の情報通信が必要となる場面も増えてきた．しかし，従来のファイル同期ツールなどを利用した場合は遠隔地と同じような手順が必要であり，アドホックな（即時的な）利用には向かない．

そのため，同一空間内における複数のデバイス間通信を円滑に実現するために様々な研究がなされてきた．例えば，ペン型のデバイスを用い，ID を利用して情報を複数のディスプレイに跨って管理する Pick-and-Drop[9]などがある．また，ユーザの位置情報やデバイス間の距離情報に基づいた情報共有の手法なども提案されている[4][8]．その中でも近年注目を集めているのが，デバイスに付属するのが当たり前のようになってきたカメラを利用して，ファイルを共有したり，ディスプレイ上のコンテンツを操作したりする方法である[5]．これは，カメラを利用することでデバイ

†1 東北大学 電気通信研究所
†2 Chalmers University of Technology, Dept. Applied IT.

スの特定や接続のトリガを簡単にしているだけではなく，カメラによる情報の記録という現実世界のメタファにも基づいているために有用性が非常に高いと考えられている．しかし，これまで提案されてきたカメラを用いるデバイス間の情報交換技術はファイル共有に基づくため[5][7]，デバイス毎に利用できるアプリケーションが異なる現状を考えると必ずしも最良の方法ではない．一方，リモートデスクトップによるスクリーン共有やミラーリングを利用することも考えられる．この方法ではデバイスにアプリケーションをインストールすることなく，他のデバイス上のコンテンツを閲覧・操作することが可能である．しかし，従来のリモートデスクトップをモバイルデバイスから利用する際には，デバイス間の解像度や操作方法の違いなどに課題がある他，アカウント制御による接続が必要で，アドホックな場面には向かない．

そこで本研究では，モバイルデバイスのカメラを用いてホストコンピュータ上に表示されているあらゆるアプリケーションウィンドウをアドホックにそのデバイス上に共有することができる手法 CamCutter を提案する．

なお，CamCutter はアドホックなデバイス間アプリケーション共有のためのツールであるが，本研究では，このアプリケーション共有を次のように定義する．

1. 特定のアプリケーションウィンドウのみをホストデバイスとクライアントデバイス間で共有する．スクリーン全体の共有に比べ画面が小さいモバイル上でもアプリケーションウィンドウに対して操作が行いやすく，また，無駄なネットワークトラフィックを抑えることができる．
2. アプリケーションウィンドウはリアルタイムにデバイス間で同期される．ファイルを共有する方法と比べ，操作を開始するまでのオーバーヘッドが短く，クライアントデバイス上で行った変更を再度ホストデバイスに送信する必要が無い．
3. アプリケーションに対する操作は，ホストデバイスとクライアントデバイスのいずれからも行うことが可能である．

もしカメラを使って，手軽にこれらの機能を持つアプリケーション共有を実現することができれば，アドホックな状況において円滑なデバイス間でのデータ共有を実現し，多様でダイナミックな作業をサポートできると期待できる．例えば，オフィスで行われるアドホックな情報の共有（図 1 a, b）や，デザイナとエンジニアが一つのアプリケーションを共有する協調作業（図 1 c），プロジェクタや大画面ディスプレイを用いたミーティング（図 1 d）などは，通常複数台のデバイスが存在し，相互の情報のやり取りには手間がかかる．本稿では，CamCutter の設計と実装およびユーザスタディについて報告し，カメラを用いたアドホックなデバイス間通信の可能性を検討する．

2. 関連研究

デバイス間のインタラクション（クロスデバイスインタラクション）に関する様々なシステムがこれまで提案されてきた．本章では，近年非常に盛んなモバイルデバイスのカメラを用いたデバイス間接続とインタラクションに関する研究を紹介する．このような技術の背景には SIFT や SURF などの画像処理特徴量技術の発展があり，我々も A-KAZE[1]を利用している．

2.1 リモートコントロール

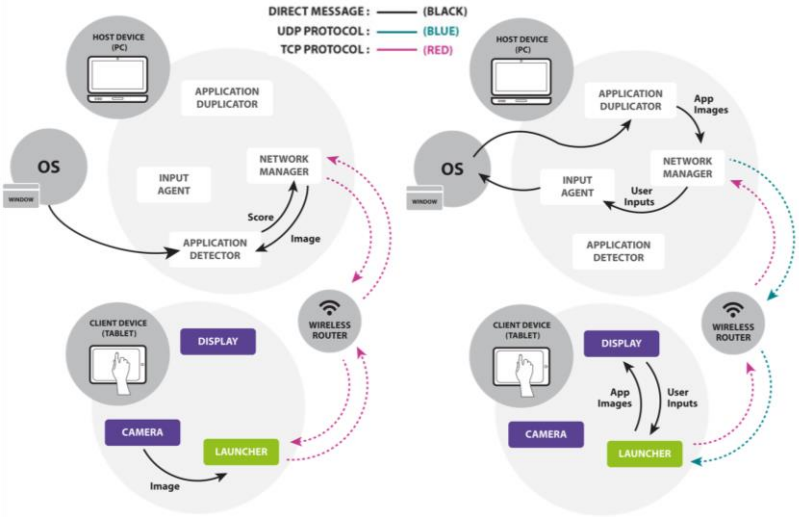
ネットワーク経由で複数のデバイスを接続し，データの送受信，同期，遠隔操作を行う研究は多くなされている．Boring らは離れた場所に置かれたディスプレイ上のコンテンツに対して，モバイルデバイスのカメラファインダーを通してリアルタイムに操作可能とする Touch projector を提案した[3]．また，Baur らはモバイルデバイスのカメラを用いてデスクトップ画面を認識し，モバイルデバイス上のデータをデスクトップ画面に対して投影し，モバイルデバイスの限られた画面を拡張する Virtual projection を提案している[2]．最近では，Freitas らによって，プリンターやプロジェクタ機器などをモバイルデバイスのカメラで撮影することで，それらの機器をモバイルデバイス上から操作可能とする Snap-to-it が提案されている[6]．このようにカメラと画像処理を組み合わせることで，簡単に異なる複数のデバイス間で情報をやり取りできるが，例えば，Snap-to-it では予め登録されたデバイスの画像に対してのみ動作をするため，動的な状況や変化するコンテンツには対応できない．

2.2 ファイル共有

複数のデバイス間でファイルを共有する際には USB メモリなどの外部記憶装置や近年ではオンラインファイルホスティングサービス（e.g., DropBox, Google Drive）が多く用いられている．Liu らはモバイルデバイスのカメラでデスクトップ上のファイルアイコンを撮影することで，そのファイルをモバイルデバイス上に転送する技術を提案した[7]．また，Chang らはデスクトップ上のアプリケーションウィンドウをカメラで撮影することで，そのアプリケーション情報を含め，データを転送する Deep Shot を提案した[5]．しかしファイルを共有する際には，それぞれのファイルに対応するアプリケーションがデバイス上にインストールされている必要があり，この方法では，我々が目指すデスクトップ上で動作するあらゆるアプリケーションに対応することは難しい．

2.3 スクリーン・アプリケーション共有

スクリーン共有とは，あるデバイスのスクリーン全体をその他のデバイスから閲覧・操作を可能とする技術のことであり，VNC や RDP などのリモートデスクトップソフト



(a) アプリケーションウィンドウ認識モード
Application window detection mode

(b) インタラクションモード
Interaction mode

図 2 : 各コンポーネントとデータ遷移
Figure 2 : Components and data transition of CamCutter

ウェアで用いられてきた．ファイルを共有する場合とは異なり，リモート側には操作したいアプリケーションがインストールされていなくてもホスト側のアプリケーションを利用することができ，Skype（閲覧のみ）や TeamViewer [10] などのソフトウェアが一般にも広く用いられている．しかし，画面サイズの小さなモバイルデバイス上でデスクトップ画面全体を操作することは難しい．また，一般的にこれらのソフトウェアではデバイス間をアカウントによって接続制御しているが，これはアドホックな場面においては大きなオーバーヘッドになってしまう．なぜなら，アドホックな場面では，複数のユーザが自身のまたは他人のデバイスに簡単に素早くアクセスすることが望まれるからである．一方，アプリケーション共有は，１章で定義したように，デスクトップ画面全体ではなく，特定のアプリケーションのみを複数のデバイス間で共有する技術である．アプリケーションのみを共有するという点では，Google Docs/Sheet/Slides [11]などのウェブサービスも強く関連するが, これらは特定のアプリケーションに限定されており，我々の目指すあらゆるアプリケーションを共有するための仕組みはない．

以上，まとめると，我々の知る限りでは，アドホックな場面において起動中のアプリケーションを複数のデバイス間で簡単に素早く共有する手法はこれまで提案されていない．CamCutter はこのような場面でユーザがあらゆるアプリケーションを素早く共有でき，複数のデバイスを跨いだダイナミックな作業形態をサポートする．

3. システム設計と実装

CamCutter における基本的な共有の手順は次の通りである．まず，モバイルデバイスのカメラで異なるホスト上のアプリケーションウィンドウを撮影すると，それらモバイルとホストの接続が確立する．そして，撮影したホスト上のアプリケーションの画像情報が逐次的にモバイルデバイス上に表示されることになる．なお，その画像に対する操作も可能であり，その操作はホスト上のアプリケーションにも反映される．CamCutter では，タブレット上のタッチスクリーンを用いてアプリケーションをカメラで撮影し，そのままタッチスクリーン上でアプリケーションを利用できるので，ユーザは撮影からアプリケーションの操作に素早く移行することができる．以降では，カメラを使ったアプリケーションの特定とアプリケーション共有のための双方向通信について述べる．

3.1 基本構成

本システムはホストデバイスとクライアントデバイスで構成され，無線ネットワークを通じてデータをやり取りする. プロトタイプではホストアプリケーションを Widows Form アプリケーションで, クライアントアプリケーションを Unity で作成した．本システムはアプリケーションウィンドウ認識モード（図２ a）とインタラクションモード（図２ b）の２つのモードを持ち，複数のホストデバイス，クライアントデバイス間で動作するように設計する．

3.1.1 アプリケーションウィンドウ認識モード

図２ (a)，クライアントの CamCutter によってアプリケーションウィンドウの画像が撮影される．撮影された画像データはネットワークに接続されている全てのホストアプリケーションの Network Manager へと送信される（図３）．Network Manager は受信した画像を Application Detector へと渡す．Application Detector はホストデバイス上で展開されている全てのアプリケーションウィンドウ画像を取得し，画像処理技術 A-KAZE を用いて受信した画像データの特徴量を抽出して，それぞれのアプリケーションウィンドウ画像と比較する．その後，Application Detector は最も類似し

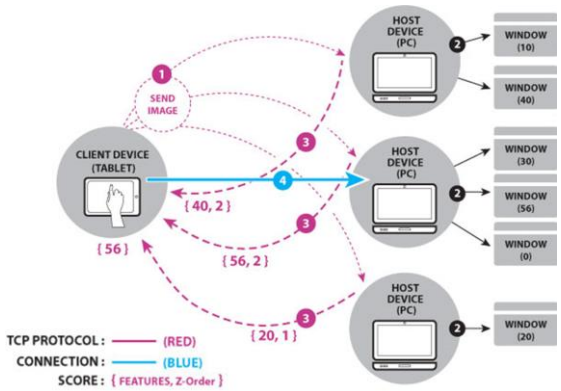


図 3 : 複数ホストデバイスとクライアントデバイスの通信
Figure 3 : Multi-host and client architecture

ているアプリケーションのウィンドウハンドルを保持し，Network Manager に対して合致した特徴量データを送信する．Network Manager はクライアント側に特徴量データとアプリケーションウィンドウの Z-オーダから計算されるスコアを渡す．クライアント側はすべてのホストデバイスからのスコアを受信した後，最大スコアを返したデバイスとの接続を確立し，インタラクションモードへと移行する．

同一またはよく似た外観のアプリケーションがデスクトップ上に展開されており，どちらかのウィンドウを撮影した場合，スコアに含まれる Z-オーダによってウィンドウは選択される．また，極めて類似するウィンドウが複数ある場合に正しいウィンドウが選択される確率は，A-KAZE の性能に依存している．

3.1.2 インタラクションモード

図 2 (b), ホスト側の Application Duplicator はアプリケーションウィンドウ画像データ（スクリーンショット）を 30fps でクライアントデバイスへと送信する. 前後のフレーム間でアプリケーションウィンドウに変化がない場合はこのステップはスキップされる．ユーザがクライアントデバイス上のアプリケーションウィンドウ画像をタッチ操作すると，タッチ位置情報はネットワークを介してホスト側の Network Manager へと送られ，Input Agent がタッチ位置情報をホストディスプレイ上にマッピングし，OS へと通知される．Input Agent は排他制御されており，複数のクライアントからの入力に対応する．クライアントデバイス上で入力されたキー情報も同様に Input Agent を介して OS へ通知される．

4. CamCutter の利用場面

本章では CamCutter の実際の利用場面について例を示して説明する．

A さんは会社のオフィスへ着き，自分のデスクトップ PC で文章作成の仕事を始めた．その後，同僚の B さんと共同で進めている作業のミーティングを行うために資料を CamCutter でモバイルデバイスへと移行し（図 1 a），B さんのデスクへと向かう．彼のデスクに到着すると，手元のモバイルデバイスを見せながら B さんの意見を資料に反映させる（図 1 b）．この時点で A さんのデスクトップ PC 上の資料データも同時に変更される．ミーティングが終わると A さんは自分のデスクに戻る途中で C さんから製品デザインの確認を依頼され，その場で CamCutter を用いてアプリケーションウィンドウを撮影して自分のデスクへ持ち帰る．また，確認したデザインをエンジニアのデスクへ持って行き，話し合いをしながらデザインを変更する（図 1 c）．

上で述べた例のように，利用するデバイスや場所が動的に変化する場合にも即自的に適応できるデバイス間インタラクション手法は今まで提案されていない．なぜなら，従来の手法やサービスの多くはユーザアカウントによって制御されているために，接続に時間を要したり，利用できるアプリケーションが限定的だったりするからである. なお，実際には，TeamViewer など既存のスクリーン共有ソフトを利用したとしても，ユーザアカウントを保存するなどすれば，ログインに時間がかからないものもある．しかし，基本的なセッティングでは，スクリーン全体の共有であり，モバイルデバイスのために洗練されたものではない．

CamCutter が日常的に利用される場合，ユーザは常に正面からアプリケーションウィンドウを撮影できるとは限らない．他者のディスプレイを撮影する場合には，斜め横から撮影する場合も考えられる. そこで我々は，CamCutter の認識精度を次の章で検証し，アカウント制御を用いた従来手法と CamCutter を 6 章で比較する．

5. 性能評価

3 章で述べたプロトタイプについて，モバイルデバイスのカメラを用いたアプリケーションウィンドウ認識精度と反応時間を評価する．

5.1 実験機器と環境

4 章で述べた CamCutter が使用されることが想定されるデスクトップセットアップとミーティングルームセットアップの 2 つの環境について実験を行う．

5.1.1 デスクトップセットアップ

本実験では，クライアントデバイスとして iPad Air 2（カメラ解像度 1920×1200）を，ホストデバイスとして Windows 10 PC (Core i7 2600, 16GB RAM)と 27 インチディスプレイ（1920×1200）を使用する．実験者は手にクライアントデバイスを把持し，ディスプレイ上のアプリケーションウィンドウがカメラファインダーに映るようにして写真を撮影する．撮影された画像がホストデバイスに送信されると，アプリケーションウィンドウ認識が開始される．ここでは，日常的に利用されることを想定した際に発生する撮影角度によるアプリケーションウィンドウ画像の歪みに対するロ

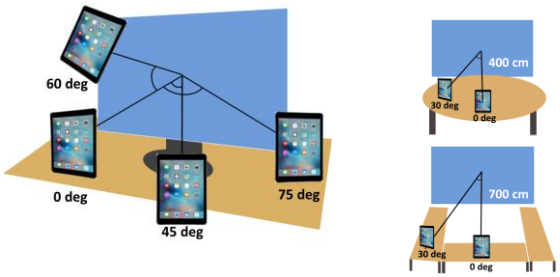


図 4: 実験条件（左：デスクトップセットアップ，
右：ミーティングルームセットアップ）
Figure 4 : Conditions in technical evaluation

バスト性と，ユーザが写真を撮影してからクライアントデバイス上にアプリケーションウィンドウが獲得されるまでの応答時間を検証する．

図 4 左に示すように，4 方向からアプリケーションウィンドウを撮影する（0 度，水平方向 45 度，水平方向 75 度，垂直方向 60 度）．水平方向は複数人で利用する際に横方向から撮影する場面を想定し，垂直方向はユーザが直立した状態のまま撮影することを想定している．全ての条件においてモバイルデバイスとスクリーンの距離は 65cm である．

5.1.2 ミーティングルームセットアップ

ミーティングルームセットアップでは，デスクトップセッアップで用いたコンピュータを使用し，ディスプレイの代わりに DLP プロジェクタ（PT-DW740, 7000 lm, WXGA）を用いて 130 インチの壁掛けスクリーンへデスクトップ画面を投影する．クライアントデバイスはデスクトップセットアップと同一のものを使用する．

図 4 右に示すように，角度は 0 度（正面）と水平方向 30 度, 距離はスクリーンから 400cm と 700cm の条件で撮影を行う．ミーティングルームセットアップでは垂直方向への変化条件は用いない．

5.1.3 表示コンテンツ

ホストデバイスのディスプレイ上に表示するコンテンツとして，文章のみ，写真のみ，２つの異なる文章，２つの異なる写真，文章と写真，の 5 つを採用する．ディスプレイ上に表示されるコンテンツが 1 つのみの場合はフルスクリーンモードで表示し，２つのコンテンツが表示されている場合, 画面を 2 分割するように表示した. 文章には Lorem ipsum を用い，フォントは Times New Roman 14 ポイント，白地に黒色の文字を利用した．写真はフルカラーの Lenna とグレースケールのクッキーの写真を利用する．

5.1.4 実験方法

実験者がディスプレイ上に表示されたアプリケーションウィンドウを各 5 回撮影し，正しいアプリケーションウィンドウが取得できたか，ウィンドウを取得するまでの時間を計測する．表示コンテンツが 2 つの場合は，モバイルデバイスに近いウィンドウ，遠いウィンドウのそれぞれに対して 3 回ずつ撮影を行う．実験者はアプリケーションウィンドウ撮影の際に，カメラビュー全体にアプリケーションウィンドウが収まるように光学ズーム機能を利用する．

5.2 結果と考察

5.2.1 デスクトップセットアップ

図 5 に，撮影された画像から抽出した特徴点とデスクトップ画面上のアプリケーションウィンドウの特徴点とのマッチング結果の例を示す．また，実験によって得られた認識精度の一覧を表 1 に示す．表より，0 度条件の場合にはすべてのコンテンツにおいて 100%の精度で正確にウィンドウを認識できた．水平方向 45 度条件の場合は 2 つの異なる文章は 50%の認識率であり，それ以外のコンテンツでは 100%に近い精度で認識が可能であった．垂直方向 60 度の条件では, 2 つの異なる文章の場合 50%の認識率であり，その他の条件では高い精度で認識することが可能であった（95.8%）. 水平方向 75 度の条件では，画像のほうが文章より良い認識率という結果となった（画像：83.3%, 文章：0%）. これらの結果は，本研究の目的である日常的な使用においては十分に満足できるものであると思われる．やや特殊な状況である水平方向 75 度の文章条件においては，さらに精度の高い画像処理手法を用いることで改善すると考えられる．

また，特徴点を抽出し，それぞれの画像を比較するまでにかかった時間は全ての条件で平均 302 ms（SD=94 ms）であり，画像をホストデバイスに送信するまでの時間は平均 84 ms（SD=28 ms）であったので，モバイルデバイス上にアプリケーションウィンドウが表示されるまでの時間はトータルで平均 386 ms であった．これは 1 秒以下でアプリケーションウィンドウを取得することが可能であることを示し，アドホックな場面に対して十分な速さであると考えられる．これらの結果はネットワーク環境にも依存するので，関連研究との単純な比較を行うことは不可能であるが，Deep Shot はデータを移行するのにかかる時間を 7.7 秒と報告したのに対して十分な速度であると考えられる．

5.2.2 ミーティングルームセットアップ

表 2 には，ミーティングセットアップにおけるウィンドウ認識精度を示す. 全ての条件の平均値は 99.2%となった. 最も認識が難しい 700cm かつ 30 度の条件のときのみ平均

表 1：デスクトップセットアップにおける認識精度
Table 1 : Application detection accuracy in desktop setup

コンテンツ/角度	0 度	45 度	75 度	60 度
文章	100%	100%	0%	100%
写真	100%	100%	100%	100%
2 つの異なる文章	100%	50%	0%	50%
2 つの異なる写真	100%	100%	50%	100%
文章と写真	100%	100%	50%	83%

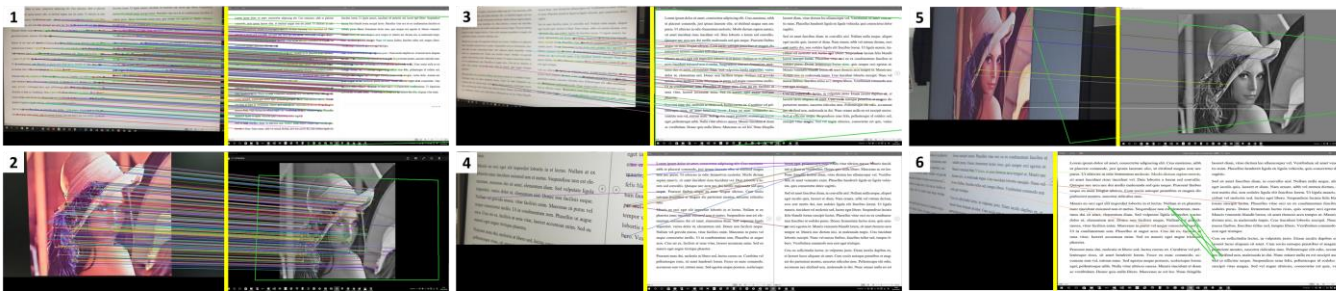


図 5: 特徴点抽出・画像マッチングの例．黄色い線の左側は撮影された画像，右側はアプリケーションのスクリーンショット．
1：文章のみ（0 度），2：写真のみ（0 度），3：文章のみ（45 度），
4：異なる 2 つの文章（45 度），5：写真のみ（75 度），6：文章のみ（75 度）
Figure 5 : Examples of feature extraction and image matching

83.3%となったが，その他の条件では 100%に近い精度でアプリケーションウィンドウを取得することができた．距離に関してはカメラの性能に大きく依存するため，今後モバイルデバイスのカメラの性能が向上するに従い，精度もより良くなると考えられる．

計算時間は，全条件平均 241 ms（SD=48 ms）という結果が得られた．また，写真の転送速度は 34 ms（SD=16ms）であった．これは，撮影の際に光学ズーム機能を用いて撮影を行ったことで画像サイズが小さくなった為と考えられる．

以上，2 つの代表的なセッティングにおける実験結果から，CamCutter を利用することで，正確に素早くウィンドウを特定して獲得できることが分かった．

6. ユーザスタディ

次に，実験参加者に CamCutter を利用してもらい，アンケートによる主観評価と，実験参加者の頭部に取り付けたカメラで撮影した操作の様子から，定量的にタスク完了時間を評価する．また，参加者に 2 つの既存のアプリケーション（TeamViewer, Google Apps）で同様のタスクを行ってもらい，CamCutter と比較する．今回，TeamViewer（スクリーンシェアリング）を比較対象として選んだ理由は，安定して動作するフリーソフトウェアであり，モバイルデバイス上でも利用が可能であることと，CamCutter が想定するアプリケーション共有が実現できるためである．同様に Google Apps も，特定の場面（文章作成やスライドの編集など）において CamCutter と同様に複数人で，そしてデスクトップ･モバイルを問わず使用できるため比較対象とした．参加者は情報科学分野の学生 12 人（男性 9 名，女性 3 名）で，7 名が日常的にタブレット型デバイスを利用しており，4 名が TeamViewer を，7 名が Google Apps を以前利用した経験があった．実験環境は性能評価で用いた構成と同じである．各手法間でカウンターバランスを取った．

表 2：ミーティングルームセットアップにおける認識精度
Table 2 : Application detection accuracy in meeting room setup

コンテンツ	距離（cm）	角度(度)	精度
文章	400	0	100%
		30	100%
	700	0	100%
		30	80%
写真	400	0	100%
		30	100%
	700	0	100%
		30	100%
2 つの異なる文章	400	0	100%
		30	100%
	700	0	100%
		30	83%
2 つの異なる写真	400	0	100%
		30	100%
	700	0	100%
		30	100%
文章と写真	400	0	100%
		30	100%
	700	0	100%
		30	50%

6.1 実験方法

実験で用いる手法は TeamViewer（スクリーン共有，マウスポインタ操作），Google Apps（アプリケーション共有，タッチ操作），CamCutter（アプリケーション共有，タッチ操作）の 3 手法である．実験は 2 つのタスク（リーディングタスク，コラボレーションタスク）から構成され，リーディングタスクは CamCutter の個人での利用を想定し，コラボレーションタスクは CamCutter を用いた複数人での協調作業を想定している．実験が開始される前に年齢，スクリーン共有の経験の有無などの一般的な項目に関するアンケート調査を行う．2 つのタスクは，共有フェーズ，作業フェーズ，事後アンケートの 3 段階に分かれている．次にそれぞれのタスクを詳細に説明する．

1. **共有フェーズ**

リーディングタスク: まず，デスクトップ画面上にはアプリケーションが何も表示されていない状態から実験は開始する. 参加者はキーボードの Win+D キーを押下することで，タスクアプリケーションウィンドウ

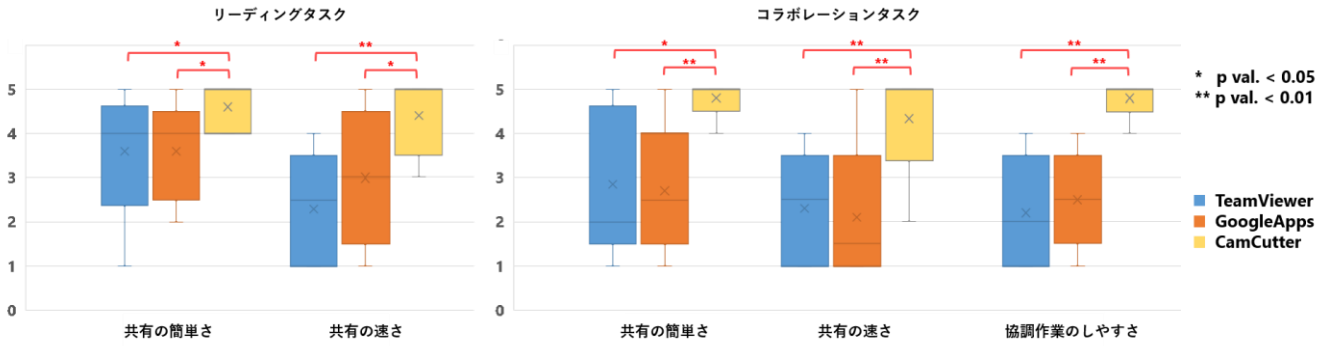


図 6 : 3 手法に対する 5 段階リッカート尺度を用いた主観評価
Figure 6 : Subjective evaluation of three tools by 5-point Likert scale

とその他の 2 つのアプリケーションウィンドウがディスプレイ上に表示される．参加者はタスクアプリケーションウィンドウ上に表示された短い文章（500 文字程度）の第一段落（250 文字程度）を読む．

コラボレーションタスク: 実験開始に伴い，参加者は実験者のいるデスクへ向かう．実験者は，参加者に対して共有するアプリケーションとその共有手法を指示する．

共通: 参加者は 3 手法の内の指定された手法で，手元のモバイルデバイス上にディスプレイ上のタスクアプリケーションウィンドウを共有する．

2. **作業フェーズ**

リーティングタスク: 参加者はアプリケーションウィンドウがモバイルデバイス上に共有できたことを確認したあと，同じ室内にいる実験者のもとへ向かう．実験者は，参加者が持つ短い文章に関する 5 つの質問を行う．この際，実験参加者は手元のデバイスで文章を確認しながら回答してよい．

コラボレーションタスク: 参加者はモバイルデバイスの画面上に表示された図形の色と位置をタッチ操作で変更し，実験者はそれらを組み立てて，家や車などの図形を作成する．

3. **事後アンケート**

共通:実験参加者は, 先ほど使用した手法について，共有の簡単さ，速さ，協調作業のしやすさ（コラボレーションタスクのみ）を 5 段階リッカート尺度で評価する．

4. **インタビュー**

全ての試行が終わった後，今回行った 2 つのシナリオを日常的に行うか，どの手法が最も使いやすかったかを質問する．

次にリーディングタスクにおけるそれぞれの手法の操作方法を説明する．リーディングタスクでは，個人使用の場面を想定しているため，アカウントのログインは自動で行われるものとした．

TeamViewer: モバイルデバイス上のクライアントアプリケーションを起動する．画面上に表示されたホストマシンを選択し，接続ボタンを押す．（3 ステップ）

GoogleApps: モバイルデバイスでアプリケーションを起動する．“実験”というフォルダを選択し，“タスク 1 ”というフォルダを選択する．文章ファイルを選択する．（4 ステップ）

CamCutter: モバイルデバイスでクライアントアプリケーションを起動する．デスクトップ画面上のアプリケーションウィンドウを撮影する．（2 ステップ）

次にコラボレーションタスクにおけるそれぞれの手法の操作方法を説明する．

TeamViewer: モバイルデバイスのクライアントアプリケーションを起動し，ソフトウェアキーボードを用いて実験者から説明された ID とパスワードを入力する．接続ボタンを押す．（3 ステップ）

GoogleApps: モバイルデバイスでブラウザアプリケーション（safari）を起動し，ソフトウェアキーボードを用いて共有 URL を打ち込む．画面上にファイルが表示された後，“アプリで開く”ボタンを押す．（3 ステップ）

CamCutter: モバイルデバイスでクライアントアプリケーションを起動する．デスクトップ画面上のアプリケーションウィンドウを撮影する．（2 ステップ）

6.2 結果と考察

本実験で得られたデータはコルモゴロフ–スミルノフ検定により正規分布に従わないと判定されたため，ウィルコクソンの符号順位検定を用いてそれぞれの手法間で一対比較を行った．図 6 のグラフにおいてリーディングタスクでは，共有の簡単さと共有の速さ，コラボレーションタスクでは，共有の簡単さ，共有の速さと協調作業のしやすさを 5 段階のリッカート尺度で実験参加者が評価した値を表した. グラフ間のバーは有意差が見られたことを示している．（* $p < 0.05$, ** $p < 0.01$）

リーディングタスクでは，図 6 左から分かるように，共有の簡単さと速さの両方で CamCutter とその他の 2 つの手

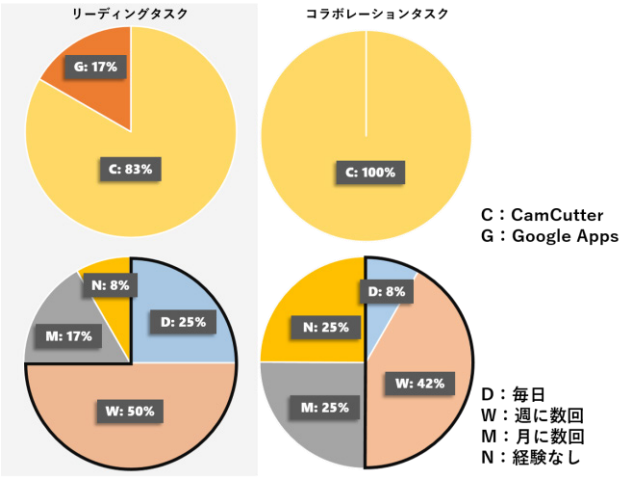


図 7：総合的に最も優れていた手法（上段），
実験シナリオと同じような状況をどれくらいの頻度で経験しているか（下段）
Figure 7 : “Preferred tool” (top). “Users’ answers to ‘how often do you meet this kind of situation’” (bottom)

法の間に有意差が認められた．共有の簡単さの観点では，GoogleApps は文章を共有するまでに，複数のディレクトリ階層を探索し，文章ファイルを見つける必要があったため，その他の手法に比べて簡単だと被験者が感じなかったと考えられる．また，TeamViewer は GoogleApps に比べて簡単であったが，モバイルデバイス上で TeamViewer を起動した後にログインの時間（10 秒程度）がかかったため，共有の速さにおいて低いスコアとなった．ビデオコーディングによって，それぞれの手法が共有フェーズを開始してから，完了するまでに掛かった時間を測定したところ，GoogleApps（7.8 秒）と CamCutter（10.8 秒）の間に有意差が見られ，CamCutter と TeamViewer（21 秒）の間にも有意差が見られた．ビデオコーディングによる結果は，実験参加者の共有の速さにおける主観評価の結果と異なるものであった．これは，GoogleApps はアプリケーションを共有するまでにファイルの探索をするため，心理的にはそれほど素早く感じなかったものと考えられる．

コラボレーションタスクでは，CamCutter は TeamViewer と GoogleApps よりも簡単に速く共有できるという結果を得られた．これらの結果から協調作業の場ではカメラでアプリケーションウィンドウを撮影するという操作がより速く，適していると考えられる．ビデオコーディングの結果によっても CamCutter の速さは明らかである（TeamViewer: 40.6 秒，GoogleApps: 62.9 秒，CamCutter: 10.2 秒）．また，協調作業のしやすさにおいては，CamCutter は他の 2 つの手法に比べて有意に良いという結果が得られた．TeamViewer においては，インタビューで実験参加者はマウスカーソルをスワイプで動かす操作方法が難しかったと述べ，結果に影響したと考えられる．GoogleApps においては，タブレット上で操作を行ってから少しの遅延の後，変更がデスクトップ画面上で反映されたため，うまく協調作業が行えないと実験参加者が感じたと考えられる．ビデオコーディングの結果は，TeamViewer（98.4 秒），GoogleApps（69.8 秒），CamCutter（60.8 秒）であった．実験参加者の主観評価とビデオコーディングによる定量的な評価の両面から見て，CamCutter は有意に最も良い手法であった．

２つの実験を通して多くの実験参加者は CamCutter が一番使いやすいと感じた（図 7 上段）．また，75%の実験参加者がリーディングタスクで体験したような自分のコンピュータ上のアプリケーションやファイルを自分の席から離れて利用することが週に数回以上あり，50%の実験参加者がコラボレーションタスクで行ったような複数人での協調作業を週に数回以上行っていることが分かった（図 7 下段）．

7. まとめ

以上，モバイルデバイスのカメラを用いたアドホックアプリケーション共有手法を提案した．本研究で行った性能評価により，複数のデバイス間で CamCutter が素早く，正確にアプリケーションを共有できることを示した．また，ユーザスタディにより，我々が想定したアプリケーションシナリオにおいて，代表的な従来手法に比べ簡単に，素早くユーザが CamCutter を扱えることを明らかにした．CamCutter はアプリケーション共有とモバイルデバイスのカメラを組み合わせることで，図 1 に示したようなアドホックな場面において優れた効果を発揮することを確認した．

性能評価において，文章ウィンドウの認識精度が十分ではない結果が得られた．これは，我々がプロトタイプで用いた A-KAZE などの画像処理技術の発展に伴い，更なる精度で認識が可能になると考えられる．

プライバシやセキュリティに関する問題はカメラを利用する際の一般的な問題であるため，本研究ではこれについて特に言及をして来なかったが，あらかじめ認証されていない端末では利用できない，写真を撮影した際にポップアップ（アラート）を表示するなどの対策が考えられる．また，写真の撮影された角度と距離を推定することで機密性の高い情報を持つアプリケーションは有効範囲を狭め，プレゼンテーションなどのパブリックな環境では広い範囲からの撮影を許可する，などの運用ポリシーを設定することも考えられる．本稿では，複数ホストデバイスと複数クライアントデバイス間でのインタラクションを検討しなかったが，今回提案したシステムは，これらの可能性を考慮した上で設計されており，実効速度やトラフィックの調査などが今後の課題である．

参考文献

[1] Alcantarilla, P., Nuevo, J., and Bartoli, A. Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces. *Proc. BMVC'13*, 13.1–13.11.

[2] Baur, D., Boring, S., and Feiner, S. Virtual projection: exploring optical projection as a metaphor for multi-device interaction. *Proc. CHI '12*, 1693–1702.

[3] Boring, S., Baur, D., Butz, A., Gustafson, S., and Baudisch, P. Touch projector: mobile interaction through video. *Proc. CHI'10*, 2287–2296.

[4] Bragdon, A., DeLine, R., Hinckley, K., and Ringel, M. Code space: touch + air gesture hybrid interactions for supporting developer meetings. *Proc. ITS '11*, 1–10.

[5] Chang, T.-H. and Li, Y. Deep shot: a framework for migrating tasks across devices using mobile phone cameras. *Proc. CHI'11*, 2163–2172.

[6] Freitas, A.A. De, Nebeling, M., Chen, X.A., et al. Snap-To-It : A User-Inspired Platform for Opportunistic Device Interactions. *Proc. CHI '16*, 5909–5920.

[7] Liu, Q., McEvoy, P., and Lai, C.-J. Mobile camera supported document redirection. *Proc. MULTIMEDIA '06*, 791-792.

[8] Marquardt, N., Hinckley, K., and Greenberg, S. Cross-device interaction via micro-mobility and f-formations. *Proc. UIST '12*, 13-22.

[9] Rekimoto, J. Pick-and-drop. *Proc. UIST '97*, 31–39.

[10] TeamViewer. https://www.teamviewer.com/.

[11] Google Apps. https://apps.google.co.jp/.