Vol.2012-DPS-150 No.32
Vol.2012-CSEC-56 No.32
2012/3/1

携帯端末を用いた実時間のプレビュー画像取得システム

高田　広平[†1]　　塚原　康仁[†1]
村瀬　結衣[†1]　　杉 浦　　一 徳[†1,†2]

本研究では Andorid 端末を用いたプレビュー画面の数秒前の画像取得が可能な Capture-A-Moment システムを開発し、実時間での画像取得を実現する。実地や遠隔地間における画像取得の場合、取得対象を発見して保存を開始するまでに時間差が生じるため、望む時間からの画像取得は難しい。そこで、プレビューを用いて対象を発見した瞬間からの画像の収集することができる環境を実現した。評価実験によって Capture-A-Moment システムは実時間での画像取得が可能であることを示した。

Capturing System by SmartPhone to Record Real-Time Scene

Kohei Takada,[†1] Yasuhito Tsukahara,[†1]
Yui Murase[†1] and Kazunori Sugiura [†1,†2]

We propose "Capture-A-Moment System" as a framework to record a moment referring to previewing image both in close and distant place. To take pictures of real-time scene, consideration of time passing is necessary just after finding preferable scenes, and then starting to record. Evaluation experiment shows the system enables us to capture a moment between close and remote location.

1. はじめに

本研究では、時間差における遅延の埋め合わせをした画像取得を実現する Capture-A-Moment システムを実装した。実時間とは画像取得したい対象を発見した瞬間と定義する。実地や遠隔地間における情報共有アプリケーションは数多く開発されてきたが、実時間での情報取得において課題がある。記録したい対象を発見した場合、ボタンを押した時点で時間差が生じるため実時間での画像収集は難しい。実時間で画像取得するためには時間差の影響を考慮する必要がある。

本研究では Capture-A-Moment システムを使用し、プレビュー画像を用いて録画ボタンを押す前の情報も含めた実地や遠隔地間での実時間の画像取得を実現する。数秒前の画像を取得することにより時間差を考慮しない実時間での画像取得が可能である。

2. 関連研究

実時間での画像を収集するツールは数多く存在する。動画の実時間転送を実現している Media-Rey は Android 端末を使用した映像伝送アプリケーションで、録画機能を実現している。画像をキャプチャする際はボタンによるスクリーンショットを採用しており、またマルチ画面にも対応している。実時間での情報の共有には適しているが、時間差による遅延を考慮しての実時間での情報の取得は難しい。[1] また、IP-webCam は Android 端末から PC 端末へ画像データを送ることで画像を共有し、web ブラウザや映像再生ソフトウェアを用いて画像を再生するアプリケーションである。アプリケーションを起動するとポート番号やホストネームに加え、デバッグ機能の設定等の画面が表示される。当アプリケーションは監視カメラとしての役割を担うが、必要な画像を取得する機能が備わっていない。[2]Sweet Home は Android 端末から取得した画像データや動画データを自動的に遠隔地へ転送する。画像をファイルとして残し、それを共有することは可能であるが、 実時間で遠隔地の画像を取得することはできない。[3]

瞬目の自動抽出の実験では、 カラー動画像から瞬目を自動検出する機能を実装している。色彩を分析して画像の取得を自動化している。[4] しかし、当システムではカメラまでの位置や背景の色彩について細かい指定が必要である。また、動画像からの顔画像抽出の研究では、大教室の講義の際にカメラから受講生の正面顔画像を取得する手法を紹介している。[5] 受講生が正面を向いた段階で画像を取得するが、受講生の身体の一部の誤検出にも言及されており、動画像の取得方法は適切とはいえない。遠隔ミーティングの記録、再生支援の研究では、ネットワークを利用した会議における膨大な情報の中からの画像の取得を実現するシステムを提案している。メモと音声を関連づけることによって画像の取得を実現する。[6] し

†1 慶應義塾大学大学院メディアデザイン研究科
Keio University, Media Design
†2 准教授
Associate Professor

Vol.2012-DPS-150 No.32
Vol.2012-CSEC-56 No.32
2012/3/1

かし、映像とメモを関連づける手法は遠隔地間での時間差を考慮すると実時間での適切な画像取得は難しい。メモを写した画像の実時間での取得の観点が欠けている。

しきい値を用いた看板文字列取得の研究では文字列連結の条件の分析し、カメラから取得した画像を解析して文字列を抽出している。画像を分解し、候補を作成することで文字列抽出を実行する。[7] また、シーン内での文字列領域の研究では輝度や空間周波数の分析によって文字列の候補を選択する。色彩ではなく照明条件の指定をして画像抽出を行う。[8] これらの研究の情報取得では、文字列を含む情景画像の指定や色や明度といった撮影条件を細かく設定している。一回の撮影で画像取得することは難しく、実践には適さない。

映像データ収集方式の研究は、画像を映像の中のベンチマークとして使用することで効率的な映像検索を実現する。映像の中から内容を予測できる可能性のある場面の静止画像を読み込み、それを画像ファイルとして格納することでベンチマークを得る。[9] 当システムを遠隔地間で転送して使用する場合、画像の取得には遅延が影響してしまう。遅延により記録が遅れた画像はベンチマークとしての役割を果たせない。

既存のアプリケーションでは時間差を考慮せずに情報を記録する機能が不足している。ボタンを押した後の情報を記録する機能は備わっているが、時間差による遅延の影響を考慮すると目的の画像を取得することは難しい。そのため、ユーザーは保存したい場面に遭遇した時に実時間で画像を保存することができない。本研究では、保存したい対象を発見してからボタンを押し保存するまでの時間差も考慮した画像取得システムを実装することで実時間での画像取得を可能にする。

3. Captere-A-Moment システムの提案

実地や遠隔地での実時間で画像を取得するために、ボタンを押す数秒前の画像取得を可能にするシステムを提案する。取得対象を発見してからボタンを押すまでには時間が経過するため、時間差を考慮しない画像取得が必要である。Capture-A-Moment システムでは録画ボタンを押す数秒前にプレビューされている画像をファイルとして保存することによって時間差によるデータの埋め合わせをした画像取得を実現する。予め画像を保存するために、その手法について検討する。

1. 逐次的保存方式

画像を取得した順にデータを保存する方式である。保存された画像を順に取得することができるため、効率的な画像の抽出が可能である。しかし、複数回にわたって情報を取得した場合には前回に保存した部分と、これから保存する部分を分割する必要がある。実時間での画像取得は複数回行うことが一般的であるため、逐次的保存形式は実践的ではない。

2. スタック方式

画像データを次々に保存し、最後に保存したファイルから順に取り出す手法である。保存の仕組みは単純であり、データを高速に保存する点では優れている。しかし、保存したデータを取得する際に順番を指定することはできず、時間の流れに沿って画像を探索することはできない。実時間での画像取得では時間軸に沿った画像探索が求められるため、スタック方式は適していない。

3. 監視カメラ方式

情報を意図的に記録するのではなく、プレビュー画像を取得し続ける。持続的に録画を続けるため情報を取りこぼす可能性は低いが、データ領域を消費する。また、目的の画像を抽出するために膨大なデータを探索する必要がある。実時間における画像共有ではプレビュー画像の確認と保存を並行して行う必要があるため、データ探索に時間を割かれる監視カメラ方式は適さない。

4. リング方式

バッファを輪状にし、許容量を超えたデータを順に上書する形式である。データが順に並ぶため画像の抽出を時間軸に沿って行うことができる。データの上書きの観点から膨大なデータの保存には適さないが、一定量のデータを保持するにあたってメモリ等の節約が可能であり、効率の良い手法である。

5. ディレクトリ分割方式

過去に取得した画像を保存するディレクトリと、ボタンを押してから画像を取得するディレクトリを分割する方式である。ボタンを押した段階で保存先のディレクトリが代わるため、過去の画像と現在の画像の両方を取得することが可能である。それぞれのディレクトリでの保存形式を工夫することで実時間での画像取得に応用できる。

効率的なデータ保存を実現するために、本研究では画像ファイルのリング形式とディレクトリ分割方式を組み合わせた保存を採用する。180 枚の画像を連続して取得し、181 枚目は 1 枚目の画像に上書きして保存される。リング形式の保存を過去のデータ用と現在のデータ用ディレクトリに分けて実行、画像取得対象を発見し、保存を開始するまでの時間差によって、保存を逃してしまう情報を含め、Chapture-A-Motion システムは実時間での画像取得を実現する。

また、遠隔地のユーザーに現場の情報を提供し、情報を共有するためにはプレビューを共有することが必要である。そのため、共有するためにプレビューの転送手法についても検討

Vol.2012-DPS-150 No.32
Vol.2012-CSEC-56 No.32
2012/3/1

する。

1. サーバーへのアップロード方式

送信側アプリケーションで取得したデータをサーバーへアップロードすることで共有する方式である。不特定多数の人々と画像共有する際に有効だが、データの共有のためにはサーバーへアクセスする必要があり、サーバーを介することで時間を消費する。

2. TCP 通信方式

TCP を用いて画像を送信する方式である。データが破損した場合には再送処理を行うため、少量のデータを確実に転送する際に活用する。本システムのようにプレビューなどの連続的データを扱う場合、輻輳等を考慮すると適切な手段ではない。

3. UDP 通信方式

UDP を用いて画像を転送する方式である。データの品質を保証せずに送信するためパケット喪失等の問題はあるが、画像の連続描画を行う際には適した転送方式である。本システムでは UDP 通信方式を採用する。動画ファイルの通信やサーバーへのアップロード方式では常に画像を保存するのは難しく、時間差を考慮しない画像取得は難しい。

Capture-A-Moment では、リング方式を用いて画像を実時間で保存することを可能にし、また実地だけではなく転送手法を用いることで遠隔地にいるユーザーもまた、プレビューを参照し、望む画像を実時間で取得することができるシステムを実装する。

4. 設　　計

保存したい対象を発見した瞬間からボタンを押して画像取得するまでの時間差を含む、実時間での携帯端末への保存方法の設計を示す。Android 端末の内蔵カメラから **onPreviewFrame** を繰り返し呼び出して画像を取得する。**OutputStream** を用いてデータを読み出して端末内にリング形式で保存する。ボタン操作でリング形式の保存を中断する処理を組み込むことにより、対象を発見してから保存するまでの時間差を考慮しない画像取得を実現する。

5. 実　　装

Capture-A-Moment システムは、送信側 Android アプリケーションと受信側アプリケーションの二つから構成されている。送信側は IP アドレス入力、ポート番号選択、送信ボタンのユーザーインターフェースを持ち、受信側はポート番号選択画面、録画ボタン、画像消去ボタン、画像再生ボタンを有している。開発環境は表 1 の通りである。

表 1　開発環境

IDE	Eclipse
言語	Java6
ライブラリ	Android SDK

5.1　システムの概要

図 1 にような処理体系で処理を実行する。送信側アプリケーションを起動すると IP アドレス入力とポート番号選択画面が表示される。入力後に送信ボタンを押すと Android 端末から取得した画像データを分割して UDP 通信によってデータを転送する。受信側ではアプリケーションを起動後にポート番号を選択して接続待ちに入る。ボタン操作によって、数秒前の画像も含めたプレビュー画像の録画、保存された画像の再生、画像の消去を実行する。また、ファイルのフォーマットは jpeg を用いる。

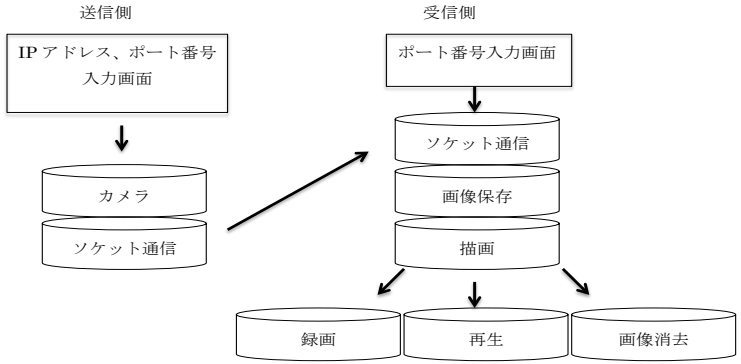

図 1　システム概要

5.2　送信側アプリケーションの転送処理

送信側アプリケーションでの画像転送は図 2 のようにして実現する。画面生成時にカメラを開き、その画像を画面に表示させる。**setPreviewCallback** で画像取得の周期を指定

Vol.2012-DPS-150 No.32
Vol.2012-CSEC-56 No.32
2012/3/1

することで、画面が表示されている間は **onPreviewFrame** が繰り返し呼ばれ、画像をキャプチャして引数にバイト配列型のデータを格納する。**YUVimage.compressToJpeg** によってキャプチャされたデータのフォーマットと画面サイズを指定し、**DatagramSocket** によって格納されたデータを読み出して UDP 通信を行う。一度に通信できるバイト配列の要素数に限りがあるため、分割してデータを送信する。以上の処理を繰り返し実行する。

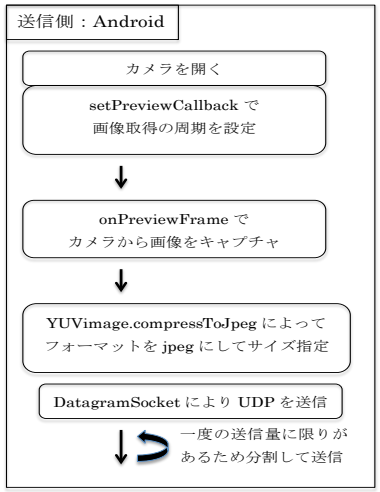

図 2 送信側アプリケーションの転送処理

5.3 受信アプリケーションの受信処理

受信側は図 3 のように UDP の接続が確立された後にパケットからデータを読み出し、数回に分けてデータをバッファする。パケットを jpeg ファイル 1 枚分までバッファした後に **OutputStream** によって jpeg ファイルをディレクトリに生成して保存する。保存されたファイルを **paint** に渡すことで画面描画を行う。以上の処理を送信側アプリケーションからパケットが送られなくなるまで続ける。

図 4 のように最初のデータは 0.jpeg、次のデータは 1.jpeg のようにファイル名の数字を 1 ずつ加算して保存する。151.jpeg を超えたデータは 0.jpeg から順に上書きされ、順にディレクトリからデータを読み込むことで画面に jpeg 画像を描画する。初期状態では jpeg ファイルの名称は-1.jpeg であり、通信が確認されるとファイル名の数字が加算されていく。

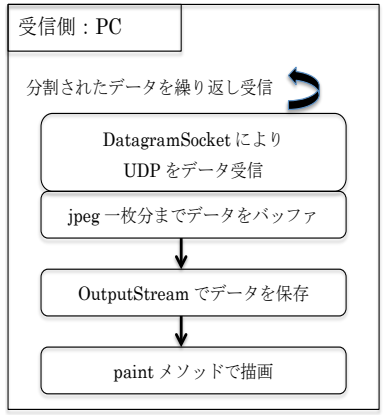

図 3 受信側アプリケーションの受信処理

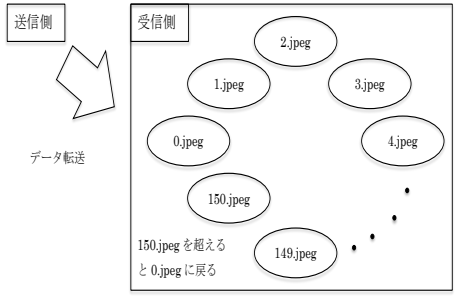

図 4 画像保存の仕組み

図 5 のように、受信側のソフトウェアは 3 つの機能を実装している。録画ボタンを押すと、プレビュー時の画像保存が実行されているディレクトリとは別のディレクトリに画像を連続して保存する。プレビュー時に保存されている画像は通常リング形式の保存によって上書きされるが、ディレクトリを代えることで上書きを防止する。その結果、数秒前にプレビューされていた画像はディレクトリに残るため、録画ボタンを押す前の情報を取得できる。

Vol.2012-DPS-150 No.32
Vol.2012-CSEC-56 No.32
2012/3/1

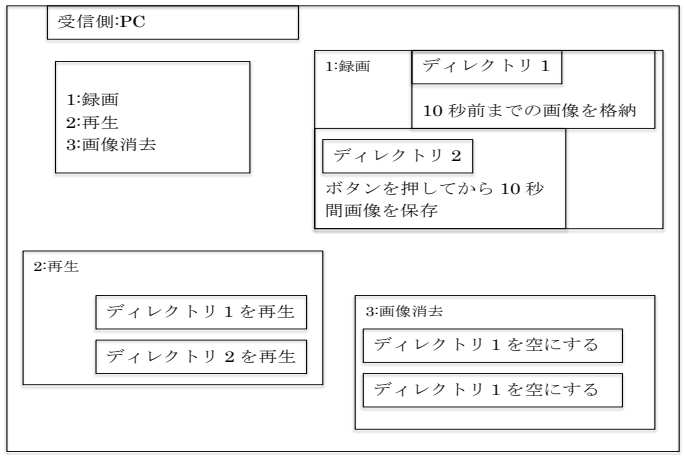

図 5 受信側アプリケーションの機能

もう一方のディレクトリには録画ボタンを押してからの画像が格納されており、これら二つのディレクトリを活用することにより時間差を考慮しない画像取得を実現する。再生機能はディレクトリに保存された画像をプレビューに適した速度で連続描画することで実現し、画像消去機能を活用することによって二つのディレクトリ内の全てのファイルを消去する。

5.4 アプリケーションの GUI の特徴

送信用アプリケーションと受信用のアプリケーションは簡易な GUI で操作を実現する。送信側アプリケーションでは図 6 のように、IP アドレスの入力画面と「○」「△」「□」の三つのボタンのみの画面構成が特徴である。この三つの図形のボタンはポート番号指定の役割を果たし、Android 端末と図 7 の受信側アプリケーションでこの記号を合わせることによって同じポート番号を共有できる仕様とした。ポート番号の入力の手間を省いた単純な操作で通信を実現し、アプリケーションの起動から画像の共有、保存までの時間を短縮する。

6. 評　　価

6.1 評 価 項 目

実時間の画像取得のために、取得対象を発見してから保存を開始するまでの時間差の影

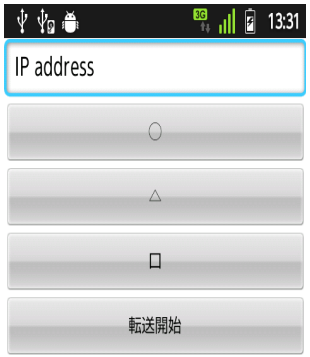

図 6 送信側アプリケーションの GUI

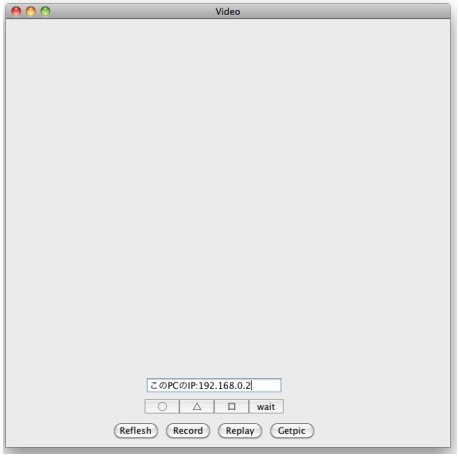

図 7 受信側アプリケーションの GUI

Vol.2012-DPS-150 No.32
Vol.2012-CSEC-56 No.32
2012/3/1

響を考慮する必要がある。本システムでは遠隔地間で実時間での画像取得を可能とする性能を有しているか評価実験を通して明らかにする。評価項目は以下である。

- 全体の処理時間
- 送信側アプリケーションの処理時間
- 受信側アプリケーションの処理時間
- 録画の処理時間

全体の処理時間では全ての種類の遅延を含めた時間を明らかにし、システム全体で消費する時間を示す。遠隔地へプレビューを転送し、必要な画像を取得する処理は端末への負荷が大きく、実時間での画像取得システムが適切に機能しない可能性が高い。したがって、遠隔地での性能評価を行うことによってシステムが実時間で画像取得が可能かどうか評価する。送信側アプリケーションではデータを分割して UDP 通信を行うまでに要する遅延を明らかにし、受信側アプリケーションではパケット受信から画像の保存、描画までの時間差を示す。これによって、システム性能が、ユーザが対象となる場面に遭遇してから録画を開始するまでの時間を含め、実時間での画像の保存が可能か示す。

6.2 評価方法

全体の処理時間は Android 端末で送信ボタンを押してから受信側で jpeg 画像が画面に描画されるまでと定義する。受信側で画像が描画された時間から送信ボタンを押した時間を減算する。Android 端末での処理時間と受信側アプリケーションでの処理時間によって生じる一連の時間差を明らかにする。

送信側アプリケーションの処理の時間は、Android 端末で送信ボタンを押してから jpeg データが送信されるまでと定義する。jpeg データが送信される時間から送信ボタンを押した時間を減算する。Android 端末でのパケット分割やソケット通信処理に要する時間を示す。

受信側アプリケーションの処理の時間は、UDP による通信を確認してから分割されたパケットを繰り返し受信し、jpeg 画像を描画するまでと定義する。jpeg 画像が描画された時間から UDP 接続を確認した時間を減算する。受信側アプリケーションでのパケット受信、バッファ、画像保存、画像描画時間を明確にする。

録画に要する時間は、受信側の端末で録画ボタンを押してから jpeg 画像がディレクトリに保存されるまでと定義する。

6.3 実験環境

実行環境は表 2 の通りである。当システムは送信側アプリケーション、受信側アプリケ表ーション共に java で実装している。画像データの転送や保存におけるデータを取得するため、測定したい場所にタイムスタンプを表示させるプログラムを挿入した。送信側アプリケーションと受信側アプリケーションを起動する。それぞれ通信環境は設定済みで、送信ボタンを押すと通信を開始する状態である。送信ボタンを押し、受信側で画像の描画が確認された後に録画ボタンを押す。その後、録画データを消去し双方のアプリケーションを停止させるまでを一回の測定とする。これにより互いの処理に干渉することなく、一貫したデータ測定を実現する。受信側アプリケーションで描画が全く見られない場合はその測定を中止し、アプリケーションを再起動させて実験を再開する。

表 2 実験環境

	送信側	受信側
OS	Android3.3	Mac OSX 10.6.6
メモリ	512MB	4GB
ネットワーク	21.19Mbps	

6.4 実験結果と分析

四つの項目をそれぞれグラフにまとめた。各グラフには x 軸に実験回数、y 軸に時間を示した。実験結果はプロットのグラフにした。また、平均値、最大値、最小値は直線で示す。全体の処理時間では図 8 のような結果が得られた。平均値は 1355ms で、最大値は 2386ms、最小値は 935ms である。平均値に近く安定した数値が多いが、値が平均値から離れる場合は受信側アプリケーションの処理時間に影響を受けている。

送信側アプリケーションは図 9 のように、平均値は 836ms で、最大値は 951ms、最小値は 771ms である。誤差は 400ms 以内に収まっており、安定してデータを送信している。送信側での処理が転送処理のみであるため、実験環境が処理に及ぼす影響は小さい。

受信側アプリケーションの結果は図 10 に示した通りである。平均値は 192ms で、最大値は 1084ms、最小値は 80ms である。最大値と最小値の間に乖離が見られるが、これはソケット通信によるデータ受信に加えて、画像の保存や描画処理の必要もあるため動作が安定しないことが原因である。

録画に要する時間では図 11 のような結果が得られた。平均値は 341ms で、最大値は 3580ms、最小値は 11ms である。ディレクトリ分割と画像保存のみの処理のため動作時間は短いが、受信側端末の環境によっては高い数値が検出された。

Vol.2012-DPS-150 No.32
Vol.2012-CSEC-56 No.32
2012/3/1

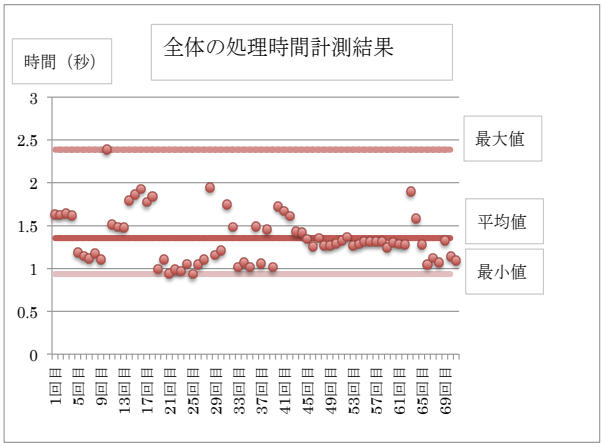

図 8 全体の処理時間計測結果

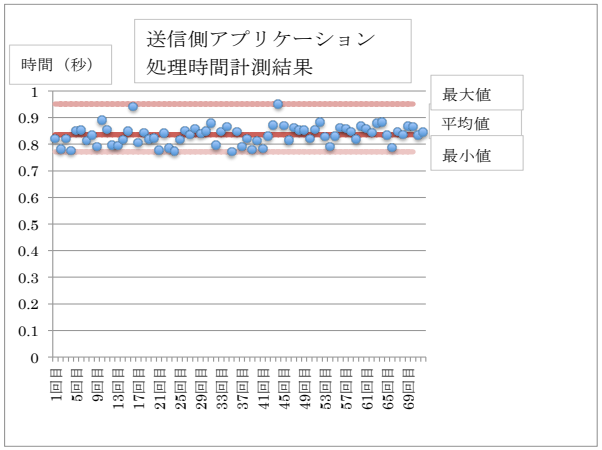

図 9 送信側アプリケーション処理時間計測結果

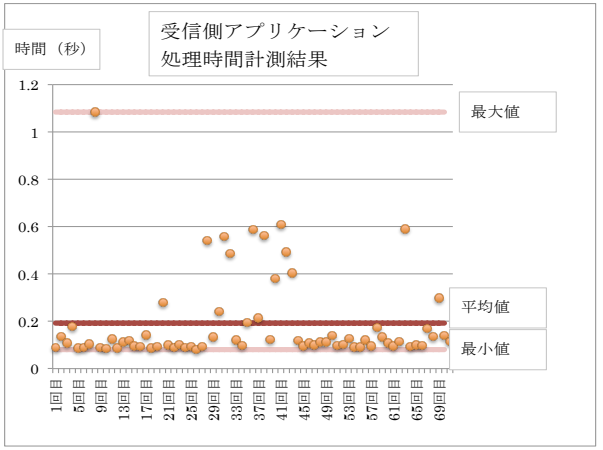

図 10 受信側アプリケーション処理時間計測結果

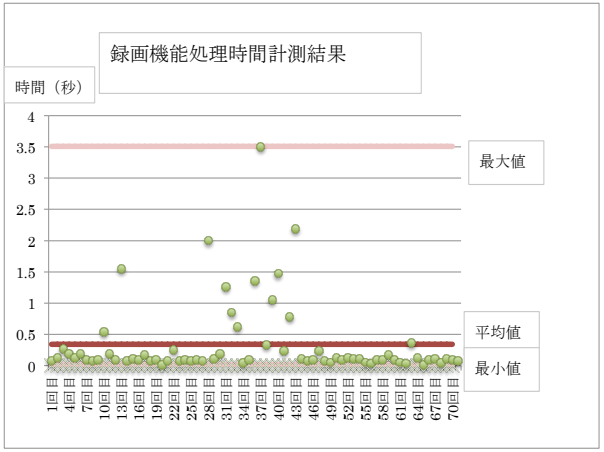

図 11 録画機能処理時間計測結果

Vol.2012-DPS-150 No.32
Vol.2012-CSEC-56 No.32
2012/3/1

6.5 評価の分析

操作による時間差や処理による遅延が影響してもシステムが正常に機能する必要がある。全体の処理時間による時間差は平均 1355ms であり、録画処理にに要する時間は平均 341ms である。Capture-A-Moment システムは最大 10000ms 前の時間までの画像保存が可能であるため、。システムの時間差の許容量を踏まえると実時間での画像取得を十分実現できる。また、送信側や受信側の処理は無視できるほど小さく、システムに与える影響は少ない。録画機能の処理も安定しており、評価結果から実時間の画像取得が可能であるといえる。

7. 今後の課題

AR や QR コードのシステムには画像抽出技術が応用されている。一つの端末内で処理を完結させる方式が一般的であるが、今後は遠隔地間で複数人が共有できる環境が求められる。その際に遠隔地間での遅延の影響を考慮する必要がある。Capture-A-Moment システムは遅延を考慮しないで遠隔地の画像取得が可能なため、画像抽出技術を応用したアプリケーションの発展の基盤に利用できる。

8. 結　　論

Capture-A-Moment システムによって実地や遠隔地で起こったその瞬間を記録することができる。既存のアプリケーションでは実地や遠隔地における遅延時間の影響により実時間で画像を取得することは難しい。そこで、本研究におけるシステムが採用しているリング型とディレクトリ分割型の保存によって数秒前の画像の取得を実現した。評価結果から当システムの画像保存方式は遅延の影響を受けないことがわかる。当システムを活用することにより遠隔地における実時間での画像取得を実現する。

参 考 文 献

1) Studio REMO.:Media-Rey, available from ⟨http://www.studioremo.jp/⟩ (accessed 2012-01-26).
2) Pas.: IP-webCam, available from ⟨https://market.android.com/developer?pub=Pas⟩ (accessed 2012-01-26).
3) SweeSoft.: Sweet Home, available from ⟨https://market.android.com/developer?pub=SweeSoft⟩ (accessed 2012-01-26).
4) 田辺　喜一,　杉山　誠：心理実験のための瞬目の自動抽出法, 電子情報通信学会論文誌, Vol.J76-D-2, No5, pp.959–966(1993).
5) 先山　卓郎,　亀田　能成, 美濃　導彦, 池田　克夫：テンプレートマッチングによる動画像からの受講者の顔画像抽出, 情報処理学会,, pp.341–342(1996).
6) 栗原　崇,　井上　直人：遠隔ミーティングの記録・再生を支援するシステムの研究, 情報処理学会研究報告, GN, 56, pp.31–36(2005).
7) 松尾　賢一, 上田　勝彦, 梅田　三千雄：適応しきい値を用いた情景画像からの看板文字列領域の抽出, 電子情報通信学会論文誌 Vol.J80-D-2, No.6, pp.1617–1626(1997).
8) 劉　詠梅, 山村　毅, 大西　昇, 杉江　昇：シーン内の文字列領域の抽出について, 電子情報通信学会論文誌 Vol.J81-D-2, No.4, pp.641–650(1998).
9) 星野　聰：映像データの収集方式の開発,NTT-Electronic Library Service, pp. 21–28(1993).