C-026

携帯端末とネットワーク上計算資源の協調による カメラセンサアプリ高速化の検討

A Study for Acceleration of Camera Sensor Application by Cooperation of Portable Devices and Servers

高橋 信宏† 寺西 良太† 清水 裕基† 吉永 努† 入江 英嗣†
Nobuhiro Takahashi Ryota Teranishi Hiroki Shimizu Tutomu Yosinaga Hidetugu Irie

スマートフォンの普及により情報を処理する環境が大きく変わってきている。特にスマートフォンでは、直感的に操作できる UI が多く登場している。中でも画像処理の需要は大きい。しかし携帯端末では資源の制限が厳しく、複雑な処理を行うのが難しい。

そこでネットワーク協調による携帯端末の計算力向上を提案する。本稿ではその前実験として、画像処理をネットワーク環境で行い、データサイズと処理時間の関連性などを調べた。結果、ネットワーク環境を通じた処理を行う場合、そのデータサイズによって処理時間が大幅に変わることがわかった。

1. はじめに

スマートフォンの爆発的な普及により、携帯端末におけるエンドユーザの情報処理環境は大きく変わってきている。従来に比べ高度なマルチメディア処理が手元で可能となり、またデスクトップ環境よりも直感的な UI が次々に登場している。携帯端末は、もはやモビリティのみを追求される非力なデバイスではなく、限られた電力の中でこれらの高度な機能を実現することを要求されている。

携帯端末において直感的な UI を実現するために需要の大きいものとして画像処理が挙げられる。例えばカメラセンサから得られた画像を元にした物体認識、ネットワークから得られた動画像の再生、さらに、AR のような画像認識と画像合成のリアルタイム処理などが挙げられる。これらが手元でスムーズに動作することは、携帯端末をより魅力的なプラットフォームとするために不可欠の処理である。また、AR のような処理は携帯端末上で動作することにより更なる応用が広がる。

しかし、このような処理は多量の資源を消費する。資源面などで様々な制限がある携帯端末において、高度な画像処理を行うことは非常に難しいといえる。

2. 携帯端末における画像処理

2.1 カメラセンサの使われ方

本研究では将来課題として携帯端末のカメラセンサ画像を入力とし、画像処理を行うことで直感的な UI 操作ができるアプリケーションを開発予定である。たとえばカメラセンサの画像を用いて携帯端末の位置や姿勢を推定したり、AR の利用したりといった例が挙げられる。

現在の携帯端末は、デスクトップマシンに比べて資源が少ない。そのため計算負荷の大きな処理を行うのは難しい。一般に画像処理は計算力を多く必要とするため、携帯端末上で複雑な処理をおこなうことは難しい。

† 電気通信大学大学院情報システム学研究科
The University of Electro-Communications' Graduate School of Information Systems

2.2 携帯端末の計算力を用いた画像認識

文献[1]では、携帯端末のカメラセンサ画像処理により、画像上のモーション検出を行っている。しかし、この端末で用いられているカメラセンサは現在流通している携帯端末よりも FPS が高い 70fps 程度で画像撮影を行っている。また演算処理を高速化するハードウェアを用いているため、現在の携帯端末上で同様のことを行うのは難しい。

2.3 精度の高い姿勢推定処理

文献[2]では 2 次元画像から撮影した物体の 3 次形状復元と撮影したカメラの 3 次元位置と撮影方向の復元を行う種法が示されている。この処理では推定精度を上げるため撮影するカメラセンサの他に 3 次元測量機材を用いている。また多くの特徴点抽出処理を行っているため計算力が必要となってくる。その現在の携帯端末上で同様の処理を行うことは難しい。

3.ネットワーク協調による携帯端末の計算力向上の提案

携帯端末の性能は日々向上しているが、デスクトップ環境には敵わない。また、携帯端末には駆動時間や通信速度といった部分にも制約が存在する。そこで本稿では端末に大きな負荷のかかる処理をネットワークを通して転送し、処理能力の高いデスクトップ環境で行い、結果を端末に送り返すという方法を想定する。今回はその予備実験として画像処理の計算時間や転送時間の計測実験を行った。

4.実験環境

4.1 実験概要

携帯端末のカメラセンサより得た画像から、端末自身の 3 次元位置と姿勢を検出するような処理を考える。位置の検出には、画像上の特徴点の抽出処理が必須である。そこで本稿では、カメラセンサから得た画像の特徴点を抽出し追跡する処理をネットワーク協調して行う手法を実装した。

複雑な画像処理を端末自身ではなく、ネットワーク上のデスクトップマシンで行うことで計算時間は速くなる。

しかし画像を転送する時間がかかってしまう。そこで画像のサイズを変えた時と、画像圧縮率を変えた時の各々の通信時間と計算処理時間を計測し変化を調べた。それによりどこがシステムのボトルネックになりうるかを検証した。

4.2 システム構成

システム構成表を表 1 に、システム構成図を図 1 に示す。今回の実験はサーバと USB カメラを用いた。また、Palantir[3]を利用することでカメラ画像をネットワーク配信できるようにした。ネットワーク環境は大学内の LAN を用いた。

PC	USBカメラ	ネットワーク回線
IntelCore2Duo2.66GHz,2.67GHz	最大200万画素	
Memory4GB	解像度最大1600×1200,	約10Mbps
OS:windowsVista	最大30fps(VGAモード)	

表 1　システム構成表

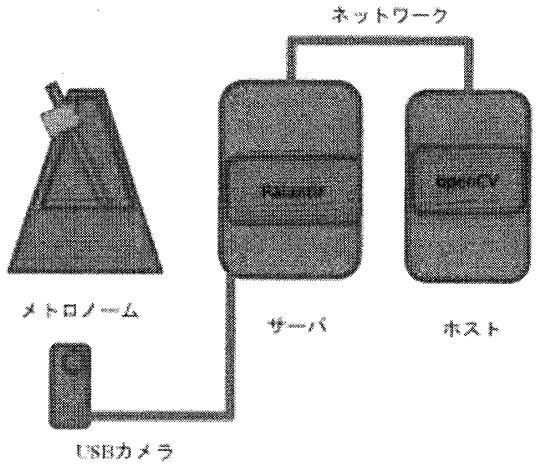

図 1　システム構成

4.3 実験内容とアルゴリズム

実験は次のような内容である。

1. メトロノームのおもり部分に色紙を貼りつけ振動させ、その様子をサーバに取り付けたカメラが撮影する。
2. ホストはサーバから画像を取得し、ホスト上の画像処理プログラムにより特徴点を検出することを行った。この時、特徴点としてメトロノームに貼りつけた色紙が検出されるようにする。

画像処理アルゴリズムとして色相による特徴点抽出を実装した。これは取得した画像に対し特徴点を色相のマッチング処理により抽出するものである。抽出方法は次のアルゴリズムで行う。

1. 画像に対して抽出したいものを範囲指定し、指定部分の色相平均を算出する。以下抽出したいものをマーカ、算出した色相平均を設定値と呼ぶ。
2. 次の画像を取得する。取得画像に対して探索窓の大きさを十分大きなものに設定する。
3. 探索窓を用いて窓内の色相平均を算出する。窓は画像の左上から右下まで滑らかに走査する。この時探索窓の内部にマーカーが多く含まれると設定値に近い値が出力される。
4. 窓内の色相平均が設定値にしきい値以上近ければ特徴点として抽出する。
5. 窓の大きさがしきい値よりも小さければ 2 へ、そうでない場合は窓のサイズを小さくして 3 を行う。

このアルゴリズムは、検索窓が縦方向と横方向になめらかに走査する。そのため、画像が大きくなると、計算量は 2 次関数的増加をすると予想される。

各フレームについて、画像取得処理と画像追跡処理それぞれにかかった時間を OpenCV の cvGetTickCount 関数により計測する。画像取得時間は Palantir にリクエストを投げ、画像がホストのメモリに載るまでの時間とする。処理時間は各々画像に対してアルゴリズムが終了するまでの時間とした。

5.結果

計測した処理時間を図 2 に示す。横軸にフレーム数、縦軸は処理時間とした。各々フレームに差はあるが、全体として一定時間で取得出来ていることがわかる。また、画像

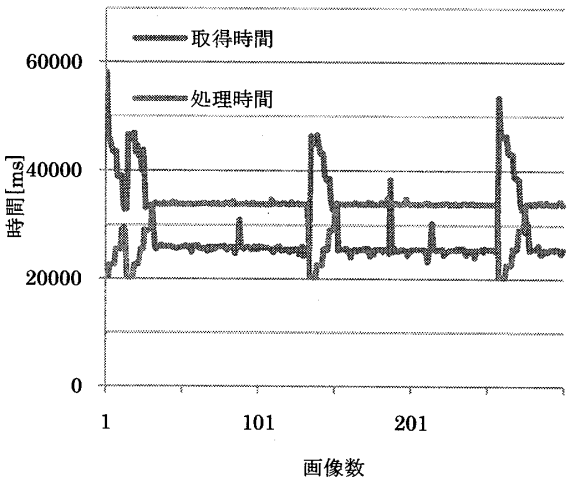

図 2　取得時間と処理時間の時間経過(640x480)

サイズは一定で変化しないため、処理時間に関してもほぼ一定の値が取得出来ていることがわかる。

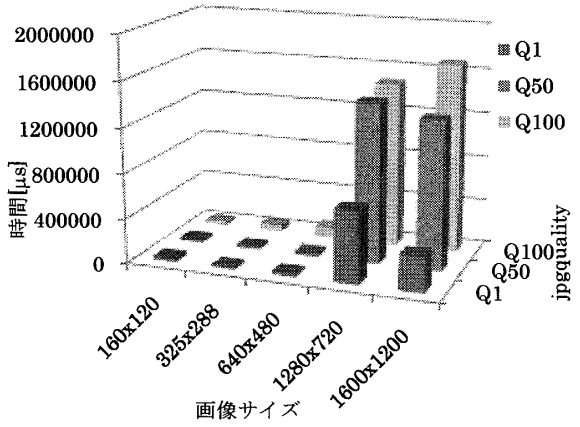

図 4　処理時間と画像サイズと圧縮率

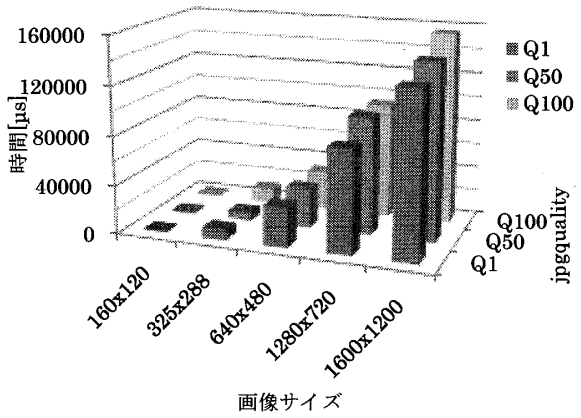

図 3　取得時間と画像サイズと圧縮率

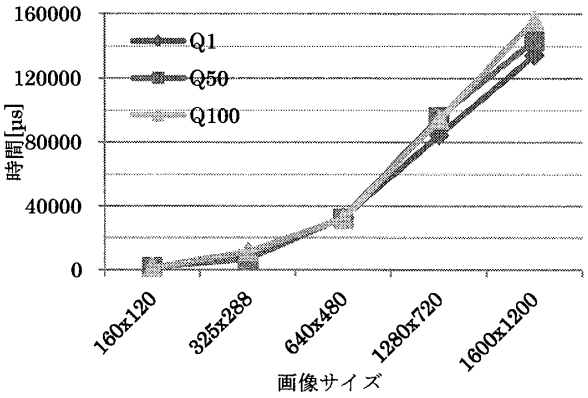

図 5　取得時間と画像サイズ

図 3、図 4 は横軸に画像サイズを、縦軸に画像圧縮率とし、高さに画像取得時間と画像処理時間を示したグラフである。

まず、画像サイズと取得時間の相関に着目すると、図 3 では画像サイズの増加と共に時間が増加していることが分かる。また、画像サイズが 1280×720 を超えると急激に増加している。また、画像圧縮率に着目すると圧縮率が高い方が、取得時間を削減していることが分かる。この傾向は、とくに大きい画像サイズで顕著となっている。

一方、画像処理時間に着目すると、図 4 では画像処理時間は画像サイズと共に増加しているが、圧縮率に関してはそれほど影響が見られない。ただし画像圧縮率を高めると、特徴点抽出のための初期色相の設定が難しくなった。

図 5 より画像サイズと計算時間の相関に着目すると画像処理時間は画像サイズに対して 2 次関数的に増加していることが分かる。これはアルゴリズムから予想される性質と一致している。

5 考察

実験より画像サイズ、画像圧縮の度合いにより特徴点抽出にどの程度の処理時間がかかるかわかった。一般に UI として利用する時、許容される遅延は 50[ms]程度と言われている。その点から考えると 640×480、圧縮率 50 までの範囲なら 50[ms]以内での処理が行えるため、本稿提案手法が有効であると言える。

本稿で用いた特徴点抽出のアルゴリズムは比較的単純なものであるが、画像処理時間は 2 次関数的に増加する傾向がある。今後より複雑なアルゴリズムを用いて処理を行うためには計算時間の短縮が課題となる。本稿手法のように、計算をネットワーク上のサーバで行えば、計算力の強化が行いやすく、有効であると考えられる。

UI としての反応速度を考えると、転送遅延は課題である。転送する画像サイズを小さくすること、圧縮率を上げることが、遅延の削減には有効である。ただし、転送する情報量の削減は、画像解析の精度とのトレードオフとなる。今回の実験では画像サイズを 160×120、高圧縮率時には特徴点抽出が困難となった。サイズと圧縮率の組み合わせは、アルゴリズムに適したものを選ぶ必要がある。

6 終わりに

今回の実験では今後の研究のための予備実験を行い、ボトルネックの判断をすることができた。問題点をこれからの研究に生かしたい。また携帯端末自身での処理速度などを測定し、実際にどの程度の処理からデスクトップマシンに行わせればよいかなどを研究していく。

参 考

[1] Yuki Hirobe, Takehiro Niikura,,Yoshihiro Watanabe, Takashi Komuro, Masatoshi Ishikawa," Vision-based Input Interface for Mobile Devices with High-speed Fingertip Tracking" 22nd ACM Symposium on User Interface Software and Technology (UIST2009) ,(2009)

[2] 佐藤智和, 神原誠之, 横矢直和, 竹村治雄, "マーカと自然特徴点の追跡による動画像からのカメラ移動パラメータの復元" 電子情報通信学会論文誌, Vol.j86-D-Ⅱ, No.10 (2003).

[3] palantir: http://www.fastpath.it