N-036

プライバシー保護のための閲覧者に応じた画像フィルタリング手法の実装と評価
Implementation and evaluation of image filtering method with authorization for privacy protection

村上 佑馬‡ 本多 隼也† 熊木 武志† 藤野 毅†
Yuma MURAKAMI Toshiya HONDA Takeshi KUMAKI Takeshi FUJINO

1. はじめに

近年，監視カメラの普及によって，犯罪の防止に大きな効果が表れている．しかしながら，監視カメラによる不特定多数の人物撮影は，プライバシー保護の観点から望ましいものではなく，取得した画像をどのような立場の人間であっても，閲覧できる状態は問題が大きい．また，ユーザが簡単に撮影できる，カメラ付き携帯電話を用いた盗撮被害が後を絶たない背景からも，撮影画面や保存したデータに対して，プライバシー保護の観点から何らかの対策が必要である．以上の背景から，我々は撮影画像内の様々な場所に対して，柔軟，階層的，かつ高速にフィルタリング処理を施す，擬似乱数と排他的論理和を用いた復元可能なマスクをかける階層型マスクフィルタリング手法として HMF(Hierarichical Masked image Filtering)法を提案している．本稿では，モバイル機器に対する HMF 法の実装効果を実機にて検証するために，超小型組み込みボートである BeagleBoard[1]と USB カメラを用いた．BeagleBoard は ARM アーキテクチャの CPU を搭載しており，カメラ付き携帯電話等を想定した実験環境を構築することが可能である．これらを用いてカメラから保存したコンテンツに HMF 法による処理を施して，実装及び評価を行った．

2. 実験環境システム構成

2.1 概要

我々が提案する HMF 法は，プライバシー保護を施したい箇所の画素値と乱数を排他的論理和演算することでマスク処理を行う．実験に必要な構成を図 1 に示す．主な処理は 4 つであり，カメラから画像・動画を取り込む処理(Import images from camera)．取り込んだ画像，動画からマスク箇所を認識する処理(Recognition)．乱数を生成する処理(Random numbers generation)，そして認識した対象物の画素値と乱数の排他的論理和演算をする処理(EXOR)である．

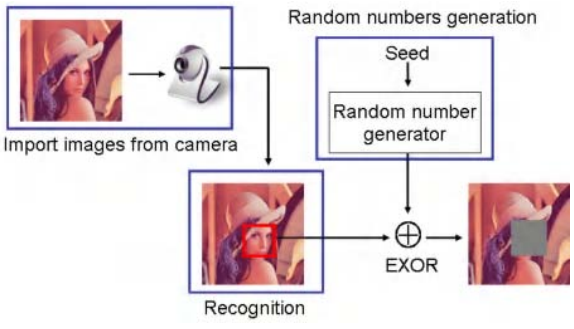

図1 HMF 法実験環境構成

本稿では，プライバシー保護の主対象として人物の顔とした．画像・動画の取り込み処理(Import images from camera)とマスク箇所の顔認識処理(Recognition)には OpenCV(Open Source Computer Vision Library)[2]を用いた．OpenCV とは，Intel 社によって開発され，Willow Garage 社によって開発が引き継がれている代表的な画像処理ライブラリである．マスク処理を実現する擬似乱数を生成する処理(Random numbers generation)には，Mersenne Twister MT19937 と呼ばれる擬似乱数アルゴリズムを用いた．なお，HMF 法に用いる擬似乱数アルゴリズムは Mersenne Twister MT19937 に限らず，柔軟に選択することが可能である．

2.2 超小型組み込みボート BeagleBoard

本節では，HMF 法の実験環境を構築するらめに用いた超小型組み込みボートである BeagleBoard について述べる．BeagleBoard とは BeagleBoard.org が開発，販売している小型のマザーボードである．搭載されている CPU は Texas instruments 社製の OMAP3530(720MHz)であり，ARM アーキテクチャを採用している．ARM アーキテクチャとは，ARM 社により開発されている CPU アーキテクチャのことであり，低消費電力を特徴とし，モバイル機器に多く用いられている．BeagleBoard Rev.C の仕様は図 2 の通りである．

- OMAP 3530(Cortex-A8 720MHz + C64x DSP + Graphics Accelerator)
- 2Gb MDDR SDRAM x32 (256MB @ 166MHz)
- 2Gb NAND x 16 (256MB)
- USB 2.0 OTG
- USB EHCI Host
- DVI-out x1
- SDスロット x1
- DC電源5V/1A or USBPower

図 2 BeagleBoard Rev.C の仕様

近年の組み込み向けプロセッサには，高速処理のニーズの高まりから SIMD(Single Instruction / Multiple Data)機構が搭載されていることが多い．本稿で用いた ARM Cortex-A8 にも，NEON テクノロジーと呼ばれる，マルチメディアアプリケーションを ARM プロセッサに効率的に導入する技術として開発された 64/128 ビットハイブリッド SIMD アーキテクチャが備えられている．

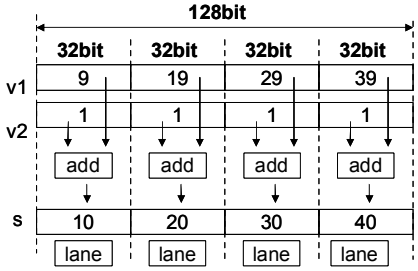

図 3 NEON テクノロジー

レジスタを同一のデータ型の複数要素として扱い，要素

† 立命館大学理工学部
‡ 立命館大学理工学研究科

を縦にレーンとして分割する．レジスタ v1，v2 にそれぞれ図 4 のように値が入れてある場合，例として加算の命令を与えると，1 つの命令で，加算を全てのレーンに対して実行する．算術演算の他に論理演算やシフト演算なども同様に行うことができ，このようにして並列処理を可能としている．

この NEON テクノロジーを用いるためには，コンパイルオプションで自動並列化を行うか，プログラム作成時に NEON 命令を直接記述する必要がある．

2.2.1 実験環境構築

HMF 法の実験環境を構築するために，図 4 に示す様に BeagleBoard にモニタ，マウス，キーボード，LAN，電源を繋ぎ，SD カードに OS として，Linux ubuntu9.04 をインストールすることで小型の LinuxPC を作成した．さらに USB カメラを接続して画像の取得を行えるようにした．

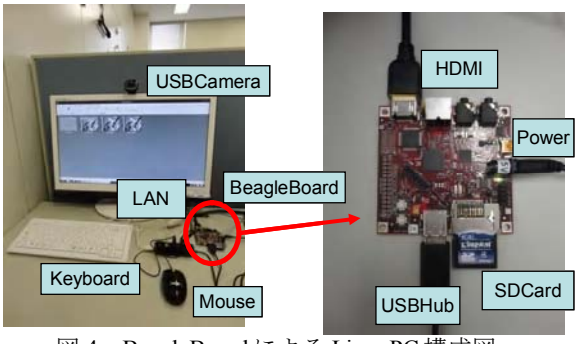

図 4　BeagleBoard による LinuxPC 構成図

BeagleBoard 上の OS を用いてプログラムのコンパイルを行うことは可能であるが，開発の効率を向上させるためデスクトップ PC を用いてクロスコンパイルを実行した．CodeSourcery G++ Lite 2011.03-41 を用いてコンパイルを行い，そこで作られた実行ファイルを BeagleBoard へ転送することでプログラムを実行した．(図 5)

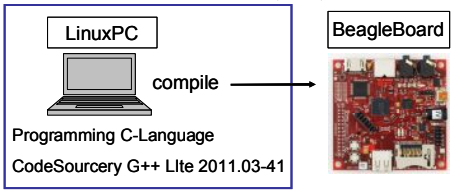

図 5　クロスコンパイル環境

クロスコンパイルを実行するに当たって，①ARM コアによる処理，②ARM コアコンパイラ最適化，③ARM コアコンパイラ最適化，かつ NEON の使用の 3 種類の実行ファイルを作成した．コマンドは以下の通り．なお，コンパイルオプションは，O3 による最適化，NEON は自動並列化とする．

3. HMF 法の実装，及び評価

HMF 法を実施するためには，まず乱数の生成が必要となる．Mersenne Twister MT19937 を BeagleBoard へ実装し，乱数を生成させた．

マスク処理を行うに当たっては，画像フォーマットのひとつである ppm(Portable Pixmap)を用いて評価を行った．ppm は画素ごとにそれぞれ 8bit の RGB カラー画像で表現が可能である．画像のサイズは，640 x 360 を用いた．ここでは図 6 の源画像(a)に対して，プライバシー保護を考慮して，顔と PC モニタにマスク処理を施した．(b)は顔のみにマスクを施し，(c)は PC モニタのみにマスクを施した．最後に(d)は両方にマスクを施した．(d)のように多重にマスクが施されている場合に，閲覧者の権限に応じて(a)，(b)，(c)のパターンでマスクの解除が可能である． このようにして階層的にマスクを施し，閲覧者に応じてマスクの解除を可能とする．

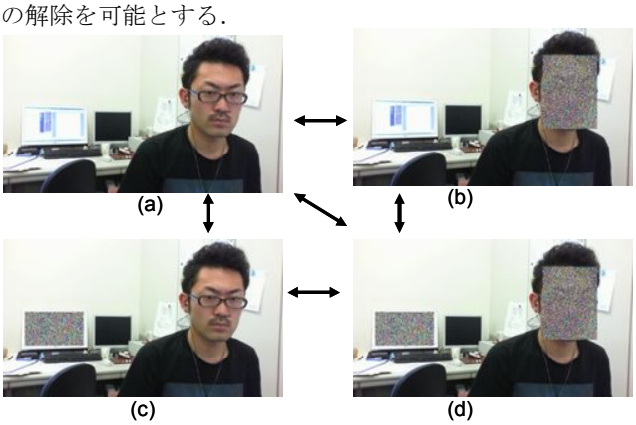

図 6　実装したマスク処理

マスク処理に必要な時間を画像読み込み時間，排他的論理和(EXOR)演算時間，画像書き込み時間の 3 つを図 6(b)の場合にて計測し，2.2.1 で述べたコンパイル別の 3 種類の実行ファイルで比較を行った．なお，比較にはそれぞれ 10 回計測した平均を用いた．

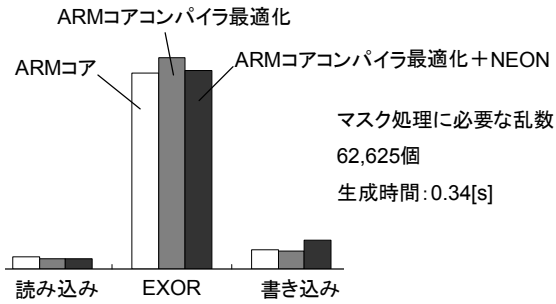

図 7　図 6(b)の画像に対する処理計測結果

図 7 より 3 種類の実行ファイルともに大差はなく，読み込み，書き込みに要する時間は約 0.30[s]であり，排他的論理和演算時間が最も遅い値で 6.06[s]という結果であった．またこの場合，マスク処理に必要な乱数は 62,625 個であり，乱数生成に必要な時間は 0.34[s]であった．結果，処理時間は排他的論理和演算に多く時間を要しており，それに対する乱数の生成時間は十分早いものであった．

4. まとめ

本稿では，モバイル機器に対する HMF 法の実装効果を検証した．排他的論理和演算に時間を要していることから，この部分の高速化が重要となる．また，静止画に対してマスク処理を行ったが，動画に対しても実装できるように，より高速に処理できるアルゴリズムを考えていく必要がある．

参考文献

[1]米田聡,”新「BeagleBoard」で最強 PC を作る” ,日経 Linux2009 年 7 月号.
[2]奈良先端科学技術大学院大学，OpenCV プログラミングブック制作チーム，”OpenCV プログラミングブック第 2 版”，2010.