O-025

階層型フィルタリング手法を用いた監視カメラデモシステム開発とその評価

Implementation and evaluation of security-camera demonstration system for hierarchical masked image filtering method

柳原　裕樹† 本多　隼也† 熊木　武志‡ 藤野　毅‡
Yuuki YANAGIHARA Toshiya HONDA Takeshi KUMAKI Takeshi FUJINO

1．まえがき

近年，犯罪捜査における犯人特定や証拠画像等の重要な情報源として監視カメラの画像は有効な手がかりとなってきている．しかし監視カメラは，不特定多数の人物を許可なく撮影する点において肖像権といったプライバシー保護上の問題が生じる事にもなる．実際に「個人情報への配慮課題」[1]という点で肖像権の侵害に繋がると社会的にも問題視されてきている．そこで我々は人物の顔や個人情報に関する画像の特定の箇所に対してリアルタイムでマスクを施し，後から権限を持つ者だけが任意でマスクを解除する事が可能な階層型フィルタリング手法(Hierarchical Masked image Filtering method：HMF) [2]を開発している．本稿は，モバイル機器や監視カメラ等の低消費電力環境下で動作が求められる場合での効果を確認するため，超小型組込みボード BeagleBoard-xM [3]を用いて，HMF を用いたマスク処理を行うプログラムの実装を行った．また HMF が実際に導入できる環境として監視カメラをターゲットとして，マスク適用を行う監視カメラとマスク除去を行う管理者側を想定した監視カメラデモシステムを構築し，性能評価及び効果の検証を行った．

2．階層型フィルタリング手法(HMF)

HMF とは人物の顔やモニタ，窓といったプライバシー保護が必要な箇所の画素群を検出し，それらとは別に生成した乱数との排他的論理和演算を行うことでマスク処理を行う．また，マスクを除去する場合は同じ乱数で排他的論理和演算を行う．HMF は階層的にマスクを適用することができ，複数のマスクに対し，個別に指定して除去する事も可能である．擬似乱数生成アルゴリズムには安全性と高速処理のために CryptMT [4]を用いる．概略を図 1 に示す．

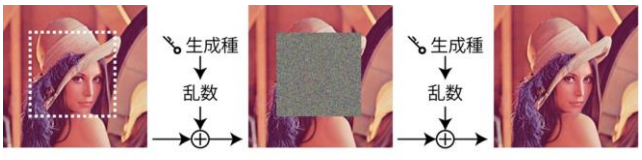

図 1　HMF による画像マスクの概略

3．実装環境，及び仕様

HMF の実装を行った環境である，超小型組込みボード BeagleBoard-xM について述べる． CPU は Texas instruments 社製の DM3730 (1GHz)を搭載し，プロセッサコアは ARM 社の開発した Cortex-A8 を採用している．

ARM コアを用いたプロセッサは低消費電力下で動作が可能であることからモバイル向けの端末に広く活用されている．本稿ではこの BeagleBoard-xM に WEB カメラを接続し，画像を取り込んだ後に人物の顔を検出し，動的にマスク処理を行なっていく．表 1 に BeagleBoard-xM の仕様を示す．図 2 に実装環境を示す．また，BeagleBoard-xM は監視カメラデモシステムにおける，監視カメラ本体の役割を担っており，OS には Ubutu11.10 をインストール，GUI は軽量且つ高速で組込み機器に適した LXDE [5]を採用した．また本稿では C 言語で実装と動作の確認を行った．

表 1　BeagleBoard-xM の仕様

OS	Ubuntu 11.10
CPU	DM3730(ARM Cortex-A8 1GHz)
メモリ	512MB LPDDR RAM
電源	DC5V 2.6A
コンパイラ	GCC4.6.1

図 2　BeagleBoard-xM の実装環境

4．実装結果

HMF が監視カメラにおいて有効であるかを検証するために，監視カメラを想定したデモシステムの構築を行った．図 3 に概略と図 4 に全景を示す．監視カメラデモシステムは監視カメラ側である BeagleBoard-xM と管理者側であるノート PC に区分される．

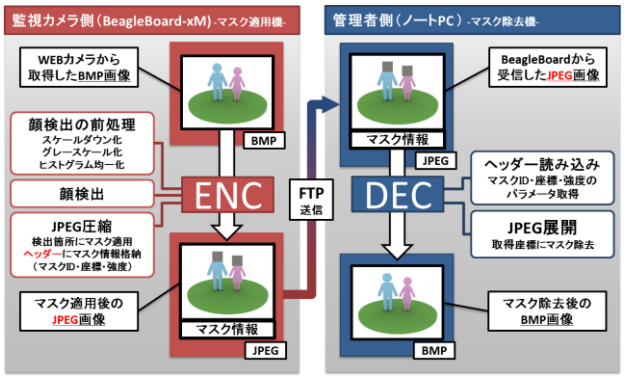

図 3　監視カメラデモシステム概略

†立命館大学理工学研究科
‡立命館大学理工学部

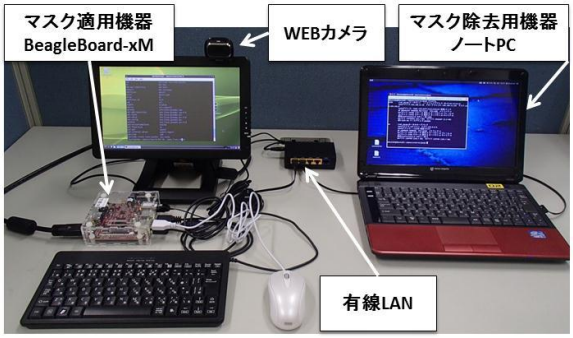

図 4　監視カメラデモシステムの全景

初めに監視カメラ側である BeagleBoard-xM について述べる．BeagleBoard-xM にはマスク適用プログラムとファイル転送スクリプトを実装した．図 3 に示す様に，マスク適用プログラムは WEB カメラから BMP 画像を取り込んだ後，顔検出の前処理としてグレースケール化，スケールダウン化，ヒストグラム均一化の処理を行い，顔検出と JPEG 圧縮を行う．JPEG 圧縮の際に検出された箇所の画素群に乱数との排他的論理和演算を行うことによりマスクを適用し，JPEG のヘッダーにマスク ID と位置座標，マスク強度のパラメータを格納する．このプログラムはフレーム毎にマスク適用された JPEG 画像を出力する．出力された画像はファイル転送スクリプトにより管理者側であるノート PC に FTP プロトコルで送信される．WEB カメラからの画像取り込みやマスク箇所の顔認識，グレースケール化には画像処理ライブラリである OpenCV 2.1 (Open Source Computer Vision Library) [6]を用いた．

次に管理者側であるノート PC について述べる．ここで用いるノート PC は市販されている通常のパソコンであり，Ubuntu12.04 がインストールされたものである．また BeagleBoard-xM から画像を受信させるために FTP サーバを導入した．これを用いてマスク除去プログラムを実装し，マスクが除去可能か検証を行う．図 3 に示す様に，マスク除去プログラムは受信した JPEG 画像のヘッダー部分からマスク情報を読み込み，取得したマスク位置座標からマスク位置を特定し，排他的論理和演算によりマスク除去を行う．

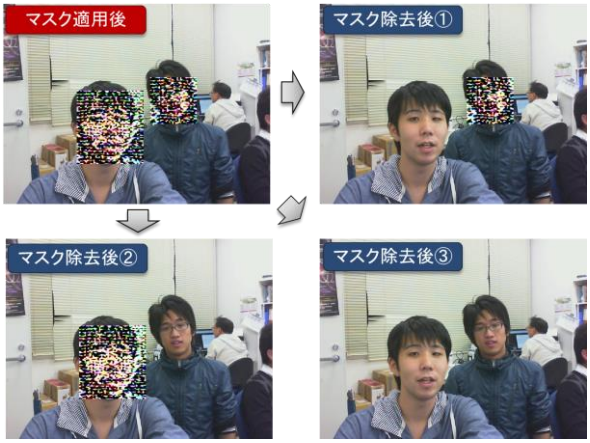

図 5　マスク除去の様子

実際のマスク適用と除去の様子を図 5 に示す．マスク適用後は取り込んだ画像中に写る人物二人の顔に対してマスクがかかっている事が確認できる．マスク除去後に関しては 3 種類画像があるが，それぞれ個別にマスクを指定して外しているものである．図中のマスク除去後①は左の人物の顔，マスク除去後②は右の人物の顔，マスク除去後③は両方の人物の顔写真に対してマスク除去を行なっており，階層的にマスクを適用できる事が確認できる．

5. 監視カメラデモシステムの評価

今回，BeagleBoard-xM に実装を行ったマスク適用プログラムは顔検出の前処理としてグレースケール化，スケールダウン化，ヒストグラム均一化を行い，その後に顔検出と JPEG 圧縮を実行する．本稿ではマスク処理の工程を大きく 5 つに分割し，400 フレーム処理させた後にそれぞれの工程 1 フレームあたりの平均処理時間を計測した．また画像内にマスク処理対象が存在する場合と無い場合での違いについて計測結果を比較した．図 6 に計測結果を示す．図 6 より 1 フレームあたりのマスク適用における処理時間は 0.45 秒以内であることから 2fps での実現が可能と言える．また，マスク対象が含まれない場合と比べて約 1.02 倍の処理時間でマスク処理が実現できる．

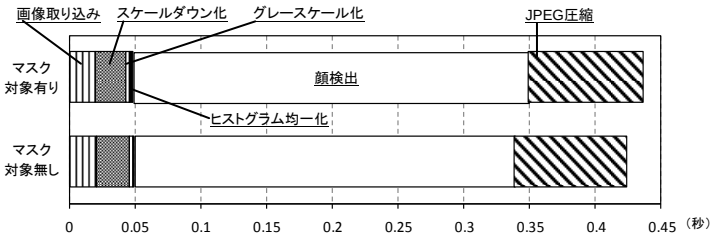

図 6　1 フレームあたりの処理時間

6. まとめ

本稿では超小型組み込みボード BeagleBoard-xM を用いて階層型フィルタリング手法を実装し，監視カメラを想定したデモシステムの構築を行った．監視カメラ側である BeagleBoard-xM は WEB カメラから取得した画像に対して，動的にマスク処理を行い，管理者側であるノート PC に対して画像を転送し，マスク除去できるよう実装を行った．マスク処理を行う工程を幾つかに分割して時間計測を行ったところ，1 フレームあたり 0.45 秒以内で実現できる事がわかり，顔検出処理に時間を要する事が確認できたため，今後はこの処理の高速化を目指す．

謝辞

本研究は，JST(科学技術振興機構)の研究成果最適展開支援プログラム(A-STEP)から助成を受けて行われた．

参考文献

[1] 日本経済新聞 2013 年 4 月 1 日
”個人情報への配慮課題,”
[2] 本多隼也他，”プライバシー保護のための閲覧者権限に応じた画像フィルタリング手法の提案,” FIT 2011, vol.4 pp.493-494，2011.
[3] http://beagleboard.org/
[4] http://www.ecrypt.eu.org/stream/cryptmtfubuki.html/
[5] http://lxde.org/
[6] http://opencv.jp/