次世代カメラプラットフォームに向けたSIMDを利用した画像認識処理の最適化

Optimization Technology of the Image Recognition Processing by a SIMD for Image Recognition Platform Development of a Next-Generation Camera

篠原 利章 Toshiaki Shinohara
高瀬 行夫 Yukio Takase
斉藤 秀雄 Hideo Saitou
東沢 義人 Yoshito Touzawa
藤田 真継 Masatsugu Fujita
ラワンカル アビジート Abhijeet Ravankar

要 旨

スマートフォンのSoC（System-on-a-chip）の能力向上を背景に，監視カメラのSoCにも同様なARM構造（注1）をもつSoCが開発され，その能力も飛躍的に向上している．CPUが複数実装され，SIMD（Single Instruction Multiple Data）処理が可能なプロセッサを実装し，デスクトップPCに迫る能力をもつSoCもある．今回の報告では，監視機器においてハードウェアやDSP（Digital Signal Processor）により，実現可能になりつつある顔照合やAVMD（Advanced Video Motion Detector）のアルゴリズムについて，SIMD処理に最適化することで，約2.5～3倍の性能向上を確認し，ソフトウェア処理のみで実装可能であることを確認したので報告する．このソフトウェア最適化技術は，画像処理など連続的な処理を繰り返し行う際に有効である．

Abstract

In response to the improved capability of System-on-a-Chip (SoC) of a smartphone, the capability of SoCs in a surveillance camera, which has the same ARM architecture as that of smartphone, has been improving by leaps and bounds. Two or more CPUs are mounted in the SoC, and it allows the processor to carry out an Single Instruction Multiple Data (SIMD) process. Some such devices have a performance equivalent to that of a desktop PC. In recent years, realization of face matching or the algorithms of Advanced Video Motion Detector (AVMD) by hardware or Digital Signal Processor (DSP) has been attained gradually. We verified that performance had improved by 2.5 to 3 times by optimizing these algorithms for the SIMD process. In this report, however, we confirmed that such optimization is only possible by software processing. This technology to optimize software is effective when continuous processing like image processing is executed repeatedly, and expansion of its application range is also being considered.

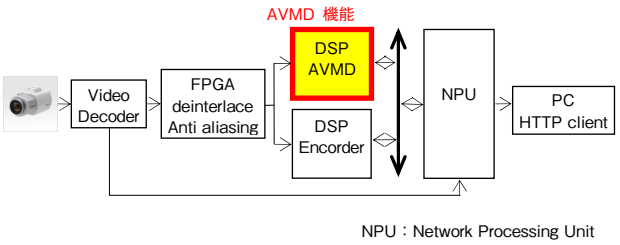
1. はじめに

近年，犯罪抑止へのニーズの高まりと，監視システムのIP（Internet Protocol）化，画像処理プロセッサの高性能化を背景に，顔照合機能などの画像認識技術が，監視用途で利用される機会が増えてきている．これまでの画像認識機能は，画像を画素単位に演算する処理があり，その演算量が大きいためFPGA（Field Programmable Gate Array）やLSI（Large Scale Integration）などのハードウェアや，DSP（Digital Signal Processor）により実現されてきた．筆者は，AVMD（Advanced Video Motion Detector）[1][2]を実装したネットワークインターフェースユニットの開発と商品化や，顔照合や年齢性別判定処理を行うネットワークディスクレコーダーの開発と商品化を行ってきた．AVMDは，DSPを使用し，顔照合や年齢性別判定処理は，CPUと連動するSIMD（Single Instruction Multiple Data）処理が可能なプロセッサを使用した．

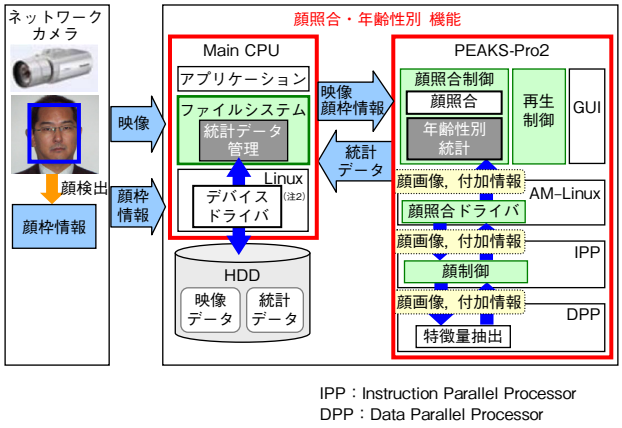
（注1）ARM は ARM Ltd.の登録商標または商標

第1図にAVMDを使った商品のブロック図を示す．この装置は，接続されたカメラの映像から，移動物体を検知し，移動物体を赤い枠で強調表示するとともに，その移動軌跡を表示し，どこの位置からカメラの画角内に侵入し，どこに向かっているかを示すことができる．処理は，FPGAとDSPで行うため，コスト低減が課題になる．

第2図にネットワークレコーダーの顔照合・年齢性別判定処理の流れを示す．顔照合は，廊下や店舗の出入り口などに設置された監視カメラで撮影した映像に通過す



第1図　ネットワークインターフェースユニットブロック図
Fig. 1　Block diagram of network interface unit



第2図　顔照合・年齢性別判定の処理構成
Fig. 2　Architecture of processing of face matching and age sex judging

る人物の顔と事前に登録された顔画像を照合して警報を出力する．年齢性別判定は，通過した人物の年齢と性別を推定し，集計結果を表示する．

この装置では，他の処理と併用するCPUとSIMD処理が行える別プロセッサが統合されており，比較的低コストで画像認識処理が実現できている．近年の監視カメラのSoCにおいても同様に，SIMD処理が可能なCPU（ARM社[3]のCortex-A9 (注3) など）が搭載されており，画像認識機能が低コストで実現できる可能性が高まっている．

これまでの画像認識を用いる監視システムの構成は，カメラで撮影した画像を圧縮伝送し，PCやレコーダーで処理するケースが多い．カメラのSoCの能力が飛躍的に向上している現状を考えると，今後，監視カメラ内部で画像認識処理できる可能性が高まってきている．カメラで処理したい理由はいくつかある．

(1)カメラのプロセッサの能力が高まることで，安価にシステム全体の処理能力を向上することができる．

(2)画像認識などの処理は，カメラの信号処理との連動で，今後性能が上がる余地がある．

(3)画像認識などの処理は，圧縮歪（ひず）みやカメラの信号処理の影響を受ける処理があるため，できるだけ非圧縮の画像で処理したい．

(1)について補足する．もし，カメラとレコーダーの台数比が1:1であれば，レコーダーの能力を上げることとカメラの能力を上げることは等価であり，システム全体の能力は変わらない．一方，現在利用されている多くの監視システムでは，1台のレコーダーに対するカメラの台数は8台以上が一般的である．例えば，この8台のカメラの能力を，安価に10倍に高めることが可能ならば，システム全体の能力を約80倍に引き上げることが可能となる．

今後は，カメラの高解像度化に伴い発生する多大なネットワークの更新コストをかける必要がないよう，記録や画像認識処理は分散されていくものと予想される．

本稿の2章では，画像認識処理をカメラのCPUのみで実現する可能性について述べる．3章では，NEON (注3) [3]を用いて画像認識処理をSIMD処理に最適化した例とその効果に加え，並列処理についての考察を示し，4章で今後の方向性や課題について示す．

2. 組込み用SoCの能力向上と画像処理のソフトウェア化

近年の監視カメラ用SoCの急激な能力向上について述べる．これまでの監視カメラのプラットフォーム（以下，PF）は，SoCのCPUの処理能力として500 DMips (注4) ~1000 Dmips（Dhrystone Million Per Second）程度の処理能力をもっていたが，今後の監視カメラのPFは，3000 DMips~7000 DMips程度の処理能力をもつ．例えば，ARM社は，汎用SIMDエンジンであるNEONや浮動小数点演算機能を実現するVFP（Vector Floating Point）を実装し処理能力を高めることができるSoCを提供している．一方，画像認識処理に必要な処理量は，画像サイズ，フレームレートにも依存するが，これまで開発してきた機器で実現してきた処理内容で，2000 DMips~4000 DMips程である．カメラの信号処理や記録の制御などで，500 DMips～1000 DMips程の処理も同一CPU内で動作させる必要があり，また，追加する画像処理の処理量を最大でも全体の50 %を超えないように実装したい．このままでも実装可能なSoCは存在するが，最新のカメラPFの5000 DMipsのSoCで画像処理を実現するためには，約2倍以上に高速化する必要があると考えている．

2.1　近年のSoCの能力向上

AMD社のGeode (注5) は，x86互換のSoCであり，MPEGのデコード機能や汎用CPUに加えてさまざまなインターフェースを集積し，セットトップボックスなどの用途に特化した専用回路を混在させており，組込み機器の価格を大きく抑えることが可能なLSIである．近年はプロセスの微細化に伴い，スマートフォンなどの用途に向けて，

（注2）Linus Torvalds氏の日本およびその他の国における登録商標または商標
（注3）CortexおよびNEONはARM Ltd.の商標もしくは登録商標
（注4）Dhrystoneベンチマークプログラムを実行して算出した，コンピュータの性能指標の1つ．測定結果のDhrystone数を1757で割ることで得られる（1757は1 MIPSマシンと言われるVAX 11/780の1秒あたりのDhrystone数）．
（注5）Advanced Micro Devices, Inc.の商標または登録商標

さらに処理能力の高いSoCが商品化されている．Qualcomm社のSnapdragon（注6）S4は，Cortex-A9に独自のカスタマイズを加え，電力あたりのパフォーマンスを向上しており，存在感を増してきている．Samsung Electronics社のExynos（注7），Apple社のA6など，スマートフォンの性能競争の影響で，監視用途にも活用可能で高速な画像処理が可能な環境が整いつつある．近年のスマートフォンのSoCと，同様な構造をもつ監視カメラ用のSoCの例を**第1表**に記載する．

第1表　プロセッサ比較
Table 1　Comparison table of processors

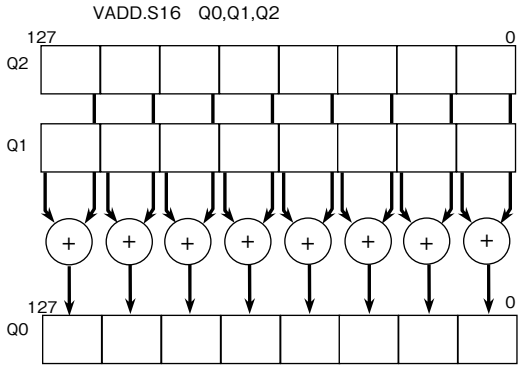
	Smartphone / Tablet PF			Security camera PF
	Apple社 A6X	Qualcomm社 Snapdragon S4	Nvidia社 Tegra4（注8）	X社
Typical MIPS@ [DMIPS/MHz]	3.5	3.3	3.5	2.5
Clock[GHz]	1.4	1.9	1.9	1.0
Core configration	2	4	4+1	2
NEON	2	4	4	2
DMIPS	9800	25 080	26 600	5000
FPU	VFPv4	VFPv4	VFPv4	VFPv3
CPU	独自Swift	独自Krait	Cortex-A15	Cortex-A9
GPU	PowerVR SGX 554 MP4	Adreno 320（注6）	ULP GrForce	-
L2 Cache[MB]	1	1	2	0.5

FPU：Floating Point Unit
GPU：Graphics Processing Unit

2.2　NEONアーキテクチャと開発環境

まず，NEONの構造とその開発環境について述べる．ARM社のNEONの構造は，大容量のレジスタ群を備えSIMDによるベクトル演算命令により，1命令で複数データを処理することができる．具体的には，128 bit長レジスタを16本(Q0～Q15)もち，1本のレジスタは8/ 16/ 32/ 64 bit データをパックして格納し，パックレーンごとの演算が可能である．また，これらレジスタは，64 bit長を32本（D0～D31）としても使用可能である．例えば**第3図**に示すベクトル加算命令は，符号付き16 bitデータ（S16）を8個パックしたレジスタQ2とQ1をレーンごとに加算しQ0レジスタに格納する．すなわち1命令で8個のデータ処理が可能である．

次に開発環境について述べる．理想的には，C言語のソースプログラムからNEONベクトル命令を自動的に生成できれば良いのだが，gccコンパイラオプションで



第3図　NEONベクトル加算命令
Fig. 3　NEON vector add instruction

「-mfpu=neon」を指定してもコンパイラに#pragma指示などでプログラム構造情報を与える手段が少なく，現時点では意図したようなNEON命令が生成されることは少ない．そこで筆者らは，NEON命令に対応したコンパイラ組込み関数（Intrinsics関数）を使用してプログラミングした．この方法ではNEON命令を熟知しアセンブラレベルの思考でアルゴリズムを組み立てることに等しいが，レジスタレベルまでは意識する必要はなくコンパイラの最適化に任せることが可能である．例えば，先に示したベクトル加算命令は次のように記述でき，C言語レベルでNEON命令が使用できる．

```
int16x8_t vaddq_s16(int16x8_t a, int16x8_t b);
```

C言語でさらに高速化を行うためには，コンパイラが生成したアセンブラコードを分析してソースコードの記述順を組み替えることにより，コンパイルが効率の良いオブジェクトコードを生成できるようにしたり，インラインアセンブラを使用して直接アセンブラ命令を埋め込むなどが必要である．

このように，画像認識の処理量に見合うSoCの能力が備わり，かつ，それを実行するための開発環境も整いつつあり，画像認識処理をソフトウェアで実現する可能性が見えてきている．

3.　顔照合，AVMDの最適化結果

近年の，処理能力の高い監視カメラ用のSoCは，約5000 DMipsの処理能力があり，顔照合のアルゴリズムは4000 DMipsである．カメラのSoC内のCPUの処理能力としてカメラのソフトウェア全体で3000 DMips程度に抑えることを目標とし画像処理の処理に使うリソースをSoCの能力の約30 %，1500 DMipsに抑えることを目標とした．そのためには顔照合処理を2.5倍以上にアクセラレーショ

（注6）SnapdragonおよびAdrenoはQualcomm Inc.の米国およびその他の国における登録商標または商標
（注7）Samsung Electronics Co., Ltd.の登録商標または商標
（注8）TegraはNvidia Corp.の米国およびその他の国における登録商標または商標

ンする必要がある．

3.1 顔照合の例

まず顔照合であるが，**第2表**中、最適化前の全体の処理の65%を占める顔処理Fを用いてSIMD最適化の具体的な効果を説明する．処理Fは顔の特定の矩形（くけい）画素の成分に対して2種類の空間フィルタを複数段処理するものである．**第4図**に処理全体の流れを示す．

第2表 顔照合の最適化結果
Table 2 Face recognition optimization result
1．顔照合

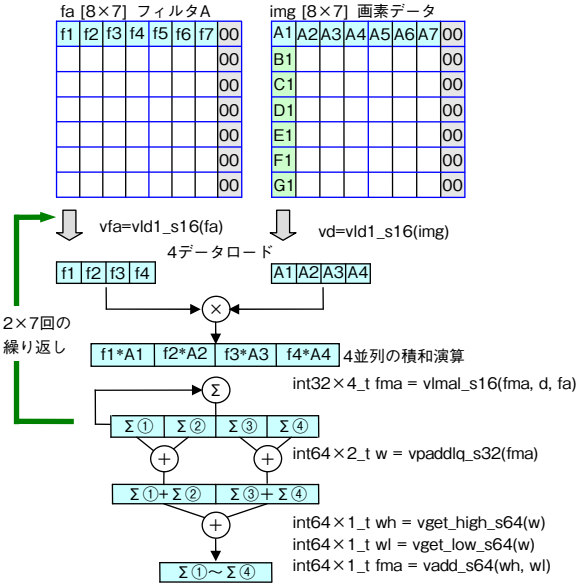
Function	NEON無	NEON有
顔処理A	3.75	3.75
顔処理B	2.50	2.50
顔処理C	7.61	3.61
顔処理D	0.83	0.75
顔処理E	12.0	6.00
顔処理F	65.0	19.0
顔処理G	4.00	0.14
顔処理H	4.31	4.31
Total	100	40.1
		2.50倍

簡単な例で説明すると，7×7の矩形の各々の画素vdに対しフィルタAとフィルタBをそれぞれを積和演算してfmaとfmbを算出するものである．C言語で記述すると以下のようになる．このまま処理すると**第2表**のNEON無の状態で，全体の65%を占める重い処理となっている．

```
int32 fma = 0;
int32 fmb = 0;
for (i=0; i<7*7; i++) {
  int16 vd = img[i];
  fma += vd * fa[i];   // フィルタA (int16)
  fmb += vd * fb[i];   // フィルタB (int16)
}
```

NEONレジスタは1つあたり128 bit長であるため，32 bitの演算結果を効率的にNEON命令で高速化するには，4画素の並列演算が効率的である．そのため，右に1列値0の列を追加して7×7の矩形画素を8×7の矩形画素に変換する．ここではフィルタAの処理について説明するがフィルタBの処理も同様である．矩形画素とフィルタAから16 bitデータを4データずつ64 bitレジスタにロードし，16 bit×4個のデータを積和演算して32 bit×4個の128 bitの演算結果を得る．これを2×7回繰り返して全画素の積和結果を求める．最後に4つの積和結果を加算して1つの積和演算結果のfmaを求める．

このような，NEONの構造に合わせ並列化可能な処理にソフトウェアの構造変更を行うことで，効率的に処理が可能となっている．



第4図 NEON積和演算のフロー
Fig. 4 NEON flow of multiply and accumulation

計測の結果，顔処理Fは約3.4倍に高速化された．NEONの4並列演算を使用していることから，右端に1列を追加したオーバヘッドを考慮してこの性能向上は妥当であると考える．**第2表**に記載する顔照合に必要な全体の処理において，データの並列演算に着目してSIMD化することにより高速化できたが，一部の処理はデータの並列性が少なくSIMD処理最適化の効果が得られない処理もあり，全処理合計でおおむね2.5倍にアクセラレーションされた．通常ソフトウェアのみで構成した場合，カメラ内では実現できない処理量であったが，CPUリソース30 %程度で実時間処理が可能になった．**第2表**に顔照合に必要な処理のアクセラレーションの度合いを評価した結果を示す．

3.2 AVMDの例

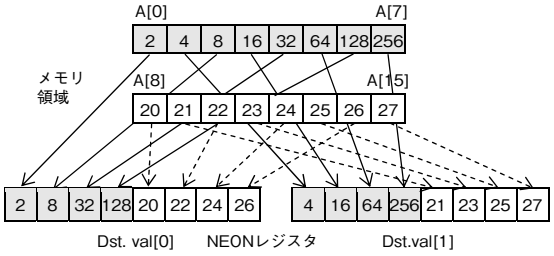
次に，AVMDであるが，**第3表**のAVMD処理の集計の結果でわかるように、全体の合計で約3.08倍に最適化され，監視カメラ内でソフト処理が可能と判断できる処理速度となった．これまで，FPGA，DSPとCPUで処理を分担していた処理が，CPUリソースの20 %程度で実現できるため，大幅なコストダウンが可能となった．

ここで，具体的な例として**第3表**のAVMDの処理Hついて説明する．処理Hは一般的なテンプレートマッチング処理である．テンプレート比較する処理画素の間隔が12画素ごと，探索ステップが2画素であるため，処理対象の画素が連続せずに飛び飛びとなる課題がある．もちろん1画素おきにマスクすることで処理は可能であるが，並列

度が半分に落ちてしまう．そこで，**第5図**に示すようにNEONのインターリーブを用いて，1画素おきの画素値を連続した配置に並べ替える配置転換を行うことで，無駄のない並列演算を実現した．

第3表 AVMDの最適化結果
Table3　AVMD optimization result

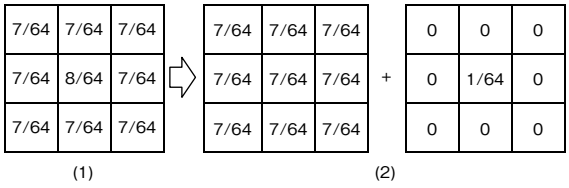
Function	NEON無	NEON有
AVMD処理A	4.00	1.40
AVMD処理B	1.50	1.00
AVMD処理C	50.4	14.1
AVMD処理D	4.40	1.00
AVMD処理E	8.10	8.10
AVMD処理F	3.20	0.50
AVMD処理G	7.10	1.20
AVMD処理H	16.10	4.00
AVMD処理I	5.20	1.20
Total	100	32.5
		3.08倍



第5図　インターリーブの活用
Fig. 5　Practical use of interleave

3.3　AVMDの例を用いたSIMD並列化に関する考察

AVMDの処理Aを用いてNEON並列化の具体的な効果を説明する．処理Aは3×3の一般的な平滑化フィルタであり，**第6図**（1）に示す係数を与えている．また，SIMD命令とその構造に合わせ，複数画素で，同一の積和演算に展開するために，本フィルタを**第6図**（2）のように展開した．



第6図　平滑化フィルタ
Fig. 6　Smoothing filter

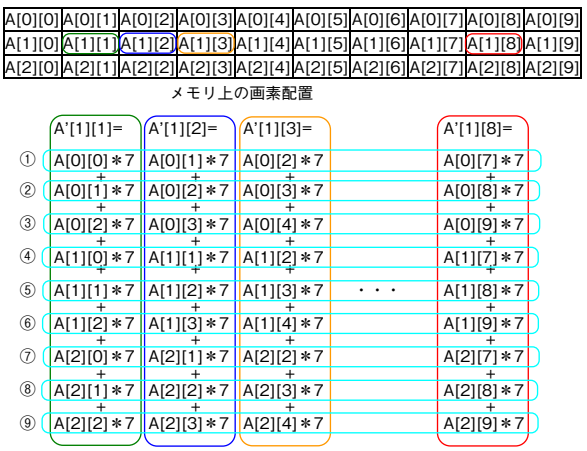
カメラ内での処理対象である画像データは通常1画素あたり8 bit長であり，そのデータに対して平滑化フィルタのような加減乗除を行うためには一時的に8 bitより長いデータ長のメモリを準備する必要がある．具体的には1画素あたり16 bit長のメモリとなる．そしてNEONレジスタは1つあたり128 bit長であるため，通常8並列での処理が可能となる．もちろん，入力データのビット長や処理内容によっては4並列や16並列での処理となる．例えば，2つの8 bit長の入力の差分絶対値を求める処理では，NEON命令にサチュレーションを起こさない差分絶対値演算が準備されているため，16並列が可能である．

第6図（2）の第一項の演算において，入力画素をA[][]，出力画素をA'[][]とした場合，A'[1][1]からA'[1][8]の各画素の計算式は，**第7図**に示す縦方向の積和演算となる．このとき，①から⑨の演算において，各入力画素が①のA[0][0]からA[0][7]のようにメモリに連続に配置されていることがわかる．そこで，A[0][0]からA[0][7]の8つの入力画素をNEONレジスタに配置し，A'[1][1]からA'[1][8]の処理を一括で行うことで8並列の実現が可能である．ここで，並列化と性能向上の関係について考察する．アムダールの法則によれば，性能向上率Sは以下の式で表される．

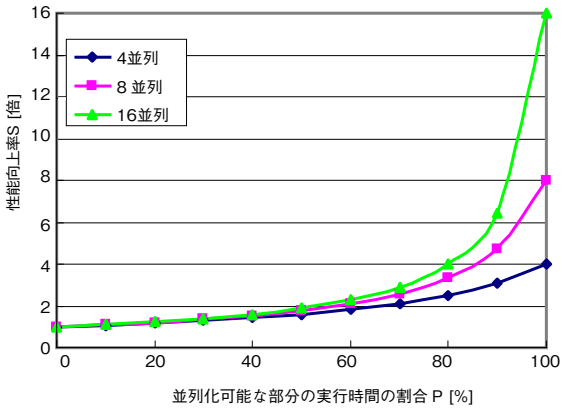
$$S=1 / ((1-P) + P/N)$$

ただし，Nは並列数，Pはプログラムのうち並列化可能な部分の実行時間の割合である．

本フィルタは，N=8，**第3表**のAVMD処理AよりS(N)=2.9であるため，P=0.74となる．つまりメモリアクセスを含めた本処理のうち74 %を並列化したこととなり，単純な画像フィルタにおけるNEONの有用性を確認することができた．言い換えると，アムダールの法則から導き出される**第8図**によれば，P=0.6では，4並列と16並列で効果は変わらないため，並列化可能な部分が少ない場合，並列度に対して高速化の効果が小さいことに注意が必要である．



第7図　積和演算
Fig. 7　MAC (Multiply and accumulation)



第8図　アムダールの法則 性能向上率
Fig. 8　Amdahl's law performance gain

以上，顔照合，AVMDともに，これまでのレコーダーや，専用のエンコーダー装置で実現してきた画像認識アルゴリズムをNEON実装することで，性能を落とすことなく，カメラ内でリアルタイムに処理しても，CPUリソース30 %以下で実現できることが実証できた．

4. まとめ

監視カメラに利用可能なARM構造のSoC内のCPUとNEONを活用し，2つの画像認識アルゴリズムが，監視カメラ内で実現可能な状況にあることが確認できた．今回の検討は，監視カメラに必要な画像認識以外の処理が決定している状態での評価ではない．今後，商品化に向けては，カメラの画像認識以外の処理を含めたキャッシュを含むメモリI/O（Input/Output）などのシステム性能について，評価が必要である．

また，今回，画像処理アルゴリズムのSIMD最適化を行うことで，分岐処理が少なく，積和演算に展開可能で，同一処理を繰り返すような処理や，同時に同一な処理を並行して加工する処理に効果が発揮されていることがわかった．このような最適化は音声処理やカメラの信号処理でも効果が期待できる．将来的には信号処理と認識処理をソフトウェアで統合することも視野に入れた開発を目指す．

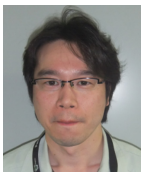
参考文献

[1] 佐藤正章　他, “侵入検知機能を搭載したネットワークインターフェースユニット”, 画像センシングシンポジウム講演論文集 14th, 画像の認識・理解シンポジウム（MIRU2008-01） 2008年7月
[2] 篠原利章　他, “AVMD（Advanced Video Motion Detector）画像認識システム，” パナソニック技報, vol. 54, no. 4, pp.8-12,
[3] ARM Ltd., “NEON,” http://www.arm.com/ja/products/processors/technologies/neon.php, 参照 Oct. 21, 2013

執筆者紹介

篠原　利章　Toshiaki Shinohara
パナソニック システムネットワークス（株）
セキュリティシステム事業部
Security Systems Business Div.,
Panasonic System Networks Co., Ltd.

高瀬　行夫　Yukio Takase
（株）パナソニック システムネットワークス
開発研究所
Panasonic System Networks R&D Lab. Co., Ltd.



斉藤　秀雄　Hideo Saitou
（株）パナソニック システムネットワークス
開発研究所
Panasonic System Networks R&D Lab. Co., Ltd.

東沢　義人　Yoshito Touzawa
（株）パナソニック システムネットワークス
開発研究所
Panasonic System Networks R&D Lab. Co., Ltd.

藤田　真継　Masatsugu Fujita
Ｒ＆Ｄ本部　新規事業開発センター
New Business Development Center, R&D Div.

ラワンカル アビジート　Abhijeet Ravankar
Ｒ＆Ｄ本部　新規事業開発センター
New Business Development Center, R&D Div.