

## 距離画像生成処理におけるメディアプロセッサの評価

清水 雄歩<sup>†</sup> 津 邑 公 暁<sup>†</sup> 中 島 康 彦<sup>††</sup>  
五 島 正 裕<sup>†</sup> 森 眞 一 郎<sup>†</sup>  
北 村 俊 明<sup>†††</sup> 富 田 眞 治<sup>†</sup>

ステレオ距離画像生成処理に対し、VLIW 型メディアプロセッサ FR-V、およびメディア演算命令を搭載した汎用マイクロプロセッサを用い、処理速度および消費電力の観点から評価、比較を行った。いずれのプロセッサでもメディア演算命令を用いることで飛躍的に処理速度が向上したが、特に FR550 は処理速度、消費電力の両点において最も優れた結果が得られ、モバイル製品等への組込み用プロセッサとして適していることを示す。また、FR550 に絶対差の総和を求める拡張命令の存在を仮定すると、サイクル数を約 28%削減でき、さらに命令並列度の改善を施すことにより、これを約 36%まで引き上げることができることを示す。

### Evaluation of Media Processors with Stereo Depth Extraction

YUHO SHIMIZU,<sup>†</sup> TOMOAKI TSUMURA,<sup>†</sup> YASUHIKO NAKASHIMA,<sup>††</sup>  
MASAHIRO GOSHIMA,<sup>†</sup> SHIN-ICHIRO MORI,<sup>†</sup> TOSHIAKI KITAMURA<sup>†††</sup>  
and SHINJI TOMITA<sup>†</sup>

This paper describes an evaluation of media-enhanced superscalar processors and FR-V: a media-enhanced VLIW processor, using an application program for stereo depth extraction. We evaluate the delay and the power consumption of each processor, and the result shows that FR550 is most superior to the others in energy-delay product. Also it is shown that 28% of the execution cycles can be eliminated by the additional instruction which calculates the sum of absolute differences directly. Furthermore, increasing the available number of media-operations in a VLIW improves the performance by 36%.

#### 1. はじめに

近年、計算機の性能が急速に向上しているものの、膨大な計算量が必要な画像処理にはまだまだ十分とはいえない。たとえば、自動監視システムでは人の動きをリアルタイムに把握しなければならない。しかし、現在の汎用計算機を用いてこのようなシステムを実現することは困難であり、マルチプロセッサや専用ハードウェアを用いた専用並列処理システムが提案されている<sup>1)~5)</sup>。

一方、専用ハードウェアを用いたシステムの開発には多くの時間とコストを必要とするため、ソフトウェアによる処理が望まれている。また、携帯電話やデジ

タルカメラ等のモバイル製品を小型化するためには、電源装置の小型化やファンに頼らない自然空冷が不可欠であり、プロセッサの低消費電力化が重要な課題となっている。

そこで本稿では、画像処理を低消費電力かつ高速に行う汎用プロセッサが今後重要になると考え、富士通製 VLIW 型プロセッサ FR-V やメディア演算命令を搭載した汎用マイクロプロセッサの性能について定量的評価を行った。特に VLIW アーキテクチャは命令のスケジューリングをコンパイラが行うため、スケジューリングをハードウェアで行うスーパー scalar アーキテクチャより回路規模を小さくでき、消費電力も少ないことが予想されることから、モバイルシステムに適していると考えている。

評価に用いるプログラムには、ステレオ距離画像生成処理を選択した。これは、ステレオ画像から距離画像、すなわち物体までの距離に近い部分ほど白く、遠い部分ほど黒く表示した画像を生成するものである。単に距離を測るためだけであればレーザー等による能

<sup>†</sup> 京都大学

Kyoto University

<sup>††</sup> 京都大学/科学技術振興事業団さきかけ研究 21

Kyoto University/PRESTO, JST

<sup>†††</sup> 広島市立大学

Hiroshima City University

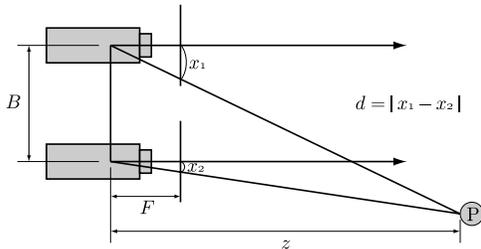


図 1 ステレオカメラにおける視差と距離の関係

Fig. 1 Relation between disparity and distance in a stereo camera.

動的手法により実現できるものの、多くの電力を必要とし、また、レーザーを受ける対象に影響を及ぼす可能性があることから、使用できる場合に限られる。一方、ステレオ画像を用いる手法では、このような問題が生じないことが利点である。

## 2. 距離画像生成アルゴリズム

### 2.1 視差と距離の関係

ステレオ画像において、物体の視差、すなわち左右の画像における対応点の位置の違い (disparity) を測ることによって、その物体までの距離を求めることができる。図 1 に、ステレオカメラにおける視差と距離の関係を示す。平行に置いた 2 台のカメラにより、ある物体 P を撮影したとき、左右の画像における P の  $x$  座標をそれぞれ  $x_1, x_2$  とすると、視差  $d$  は、

$$d = |x_1 - x_2| \quad (1)$$

である。また、 $B$  をベースライン距離 (カメラ間の距離)、 $F$  をカメラの焦点距離とすると、 $d$  とカメラから P までの距離  $z$  には次の関係が成り立つ<sup>6)</sup>。

$$d = BF \frac{1}{z} \quad (2)$$

すなわち、ステレオ画像における視差から、物体までの距離を求めることができる。

### 2.2 対応点探索アルゴリズム

視差を求めるためには、左右の画像から対応点を見つけなければならない。よく用いられるアルゴリズムに SSD (Sum of Squared Difference) がある。これは、対応点では、周囲の画素値の差が 0 に近いという性質を用いたアルゴリズムである。図 2 に、ステレオ画像と視差の関係を示す。

基準となる画像の点 A の周囲にスモールウィンドウを設け、他方のスモールウィンドウとの間の各画素値の差の自乗和を計算する。スモールウィンドウを動かしながらこの計算を行い、画素差の自乗和が最小となる点を探索する。対応点の  $x$  座標の差が視差とな

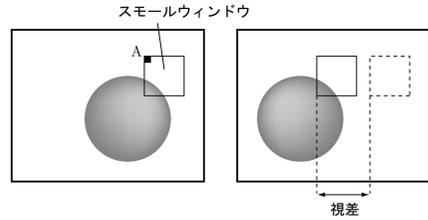


図 2 ステレオ画像と視差

Fig. 2 Stereo images and disparity.

る。ただし、スモールウィンドウごとに画素差の自乗和を計算するため、膨大な計算量が必要となる。

より計算量を減らすためのアルゴリズムとして、画素差の絶対値の総和を用いる SAD (Sum of Absolute Difference) がある。いずれのアルゴリズムにおいても、スモールウィンドウが大きいくほど正確な距離画像が得られるものの、計算量は二次関数的に増加する。

### 2.3 処理の流れ

本稿では、スモールウィンドウのサイズを  $9 \times 9$  画素とする SAD を用いた。また、各画素値は上位 24 bit を RGB 各 8 bit、下位 8 bit を 0 とする 32 bit とした。以下に、距離画像生成処理の手順を示す。

- (1) ぼかし ノイズがあると、次の輪郭検出処理においてノイズを輪郭と見なすことがあるため、2 台のカメラからのステレオ画像をぼかす。
- (2) 輪郭検出 SAD では、輪郭を含まない、同じような色からなるスモールウィンドウが連続する場合、対応点を検出することができない。これは、正しい対応点以外のスモールウィンドウに対しても SAD の値が最小となりうるためである。誤認識を防ぐために、まず、左右の画像において輪郭を検出し、輪郭部分における視差を求めることにした。
- (3) ノイズ除去 輪郭画像からノイズを除去する。
- (4) 距離計算 距離画像生成処理の大部分を占める。基準となる一方の画像において  $9 \times 9$  画素のスモールウィンドウを仮定し、他方の画像についても同じ座標にスモールウィンドウを仮定する。左画像のスモールウィンドウは右に、右画像のスモールウィンドウは左に動かしながら、スモールウィンドウ中の 81 画素について各画素差の絶対値の総和を求める。総和が最小、かつ、その点に対応するスモールウィンドウが輪郭を含む場合、対応点と見なし、視差 ( $x$  座標の差) を記録する。そうでなければ最近輪郭と見なした点から視差を補完する。これをすべての点について繰り返す。また、この操作を左右の画像に対して行い、2 枚の距離画像を生成する。
- (5) 距離画像の合成 2 枚の距離画像を比較し、各画

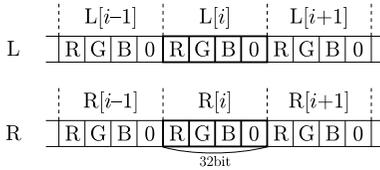


図 3 メモリ上の画像データ配列

Fig. 3 Image data arrangement on memory.

```

LOAD L[i]  Li
LOAD R[i]  Ri
DIFF Li, Ri  Di
ADD S + Di  S
    
```

図 4 一般的なカーネルコード

Fig. 4 General kernel code.

素ごとに距離の遠い方を最終的な距離として採用する。

- (6) ノイズ除去 距離画像からノイズを除去する。
- (7) 補完 輪郭が不明瞭なために生じる、くしの歯状の誤差を補正するために、上下の距離画像を基にして補完する。

距離画像生成処理では、約 97% をスモールウィンドウの比較処理が占める。

### 3. メディア演算命令を用いた高速化手法

#### 3.1 一般的な高速化手法

距離画像生成処理の大部分を占める比較処理をメディア演算命令を用いて高速化することにした。

メディア演算命令を用いた比較処理は、各プロセッサに搭載されている命令セットの種類によって異なるもの、おおむね以下のように実現することができる。

図 3 に示すように、画素データはメモリ上で配列として保持されている。図 4 に、一般的なカーネルコードを示す。左右のスモールウィンドウから 1 画素分のデータ  $L[i]$ ,  $R[i]$  を LOAD 命令でレジスタ  $L_i$ ,  $R_i$  にロードし、DIFF 命令で RGB 成分ごとに画素差の絶対値を計算し、その和  $D_i$  を求め、 $D_i$  を ADD 命令でレジスタ  $S$  に累積する。ここで、DIFF 命令は、左画像における画素の R 成分を  $L_R$  等と書くと、

$$D_i = |L_R - R_R| + |L_G - R_G| + |L_B - R_B| \quad (3)$$

を直接求める命令である。この命令が搭載されていないプロセッサの場合は、他の命令を使用して画素差の絶対値を求める必要がある。

#### 3.2 FR-V における高速化

FR-V には、整数演算命令セットとともに、画像処理や MPEG ビデオに適したメディア演算命令セット、オーディオ信号の圧縮・伸張や座標計算に適した浮動小数点演算命令セット、信号処理用の DSP 演算命令セット

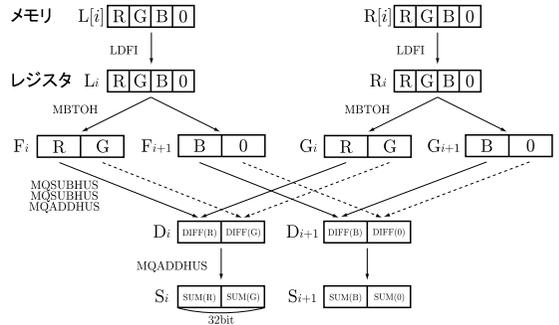


図 5 FR500 を用いた場合の比較処理の流れ

Fig. 5 Flow of comparison processing at the FR500.

ト、カスタマイズ可能なユーザ定義命令セット等が搭載されており、適切な命令セットを選択し、VLIW の枠組みを用いて並列実行させることができる。

評価に用いたのは、FR500 (4 並列実行型 VLIW) および FR550 (8 並列実行型 VLIW) である。

#### 3.2.1 FR500

FR500 は整数演算命令と、浮動小数点演算命令またはメディア命令をそれぞれ最大 2 個同時実行することができる<sup>7)</sup>。

FR-V のメディア演算命令では、画素値の各 8 bit の差の絶対値を直接求めることはできない。しかし、メディアクワッド 飽和付き加算/減算命令 MQADDHUS/MQSUBHUS を工夫して使うことにより、差の絶対値を求めることができる。たとえば、 $|a - b|$  を求めるとき、 $a - b$  と  $b - a$  の各飽和付き減算結果を求める。飽和減算では減算結果が負の数になると 0 に飽和するため、それらを加算すると差の絶対値  $|a - b|$  が求まる。MQSUBHUS は 1 命令で同時に 4 個の減算ができ、2 命令が並列実行できるので、1 ステップで 8 個の減算ができる。9 × 9 のスモールウィンドウの 1 行に対する比較処理を、メディア演算命令を使用してハンドコーディングした。

図 5 に、1 画素どうしの比較処理の流れを示す。左右のスモールウィンドウから 1 画素分の 32 bit データ  $L[i]$ ,  $R[i]$  を LDFI 命令によりレジスタ  $L_i$ ,  $R_i$  にロードし、MBTOH 命令により各 8 bit の RGB 値を 16 bit に拡張し、左右それぞれの画素データをレジスタ  $F_i$ ,  $F_{i+1}$  と  $G_i$ ,  $G_{i+1}$  に格納する。DIFF(R) は、R 成分について MQSUBHUS と MQADDHUS により飽和演算を行い、

$$\begin{aligned} \text{DIFF}(R) &= |F_{iR} - G_{iR}| \\ &= (F_{iR} - G_{iR}) + (G_{iR} - F_{iR}) \quad (4) \end{aligned}$$

を求めることを意味する。G, B 成分についても同様に求め、結果をアキュムレート用レジスタ  $S_i$ ,  $S_{i+1}$

LDLFI <sub>L</sub> ←L <sub>i</sub> LDFIR <sub>R</sub> ←R <sub>i</sub>	LDLFI <sub>L</sub> ←L <sub>i+1</sub> LDFIR <sub>R</sub> ←R <sub>i+1</sub>	MBTOH <sub>L</sub> ←F <sub>i</sub> F <sub>i+1</sub> MBTOH <sub>R</sub> ←G <sub>i</sub> G <sub>i+1</sub> MQSUBHUS <sub>F</sub> ←G <sub>i</sub> X MQSUBHUS <sub>G</sub> ←F <sub>i</sub> Y MQADDHUS <sub>X</sub> ←Y-D <sub>i</sub> MQADDHUS <sub>D</sub> ←S <sub>i</sub> -S <sub>i</sub>	MBTOH <sub>L</sub> ←F <sub>i+2</sub> F <sub>i+3</sub> MBTOH <sub>R</sub> ←G <sub>i+2</sub> G <sub>i+3</sub> MQSUBHUS <sub>F</sub> ←G <sub>i+2</sub> X MQSUBHUS <sub>G</sub> ←F <sub>i+2</sub> Y MQADDHUS <sub>X</sub> ←Y-D <sub>i+2</sub> MQADDHUS <sub>D</sub> ←S <sub>i+2</sub> -S <sub>i+2</sub>
LDLFI <sub>L</sub> ←L <sub>i+4</sub> LDFIR <sub>R</sub> ←R <sub>i+4</sub>	LDLFI <sub>L</sub> ←L <sub>i+5</sub> LDFIR <sub>R</sub> ←R <sub>i+5</sub>	MBTOH <sub>L</sub> ←F <sub>i+1</sub> F <sub>i+2</sub> MBTOH <sub>R</sub> ←G <sub>i+1</sub> G <sub>i+2</sub> MQSUBHUS <sub>F</sub> ←G <sub>i+1</sub> X MQSUBHUS <sub>G</sub> ←F <sub>i+1</sub> Y MQADDHUS <sub>X</sub> ←Y-D <sub>i+1</sub> MQADDHUS <sub>D</sub> ←S <sub>i+1</sub> -S <sub>i+1</sub>	MBTOH <sub>L</sub> ←F <sub>i+2</sub> F <sub>i+3</sub> MBTOH <sub>R</sub> ←G <sub>i+2</sub> G <sub>i+3</sub> MQSUBHUS <sub>F</sub> ←G <sub>i+2</sub> X MQSUBHUS <sub>G</sub> ←F <sub>i+2</sub> Y MQADDHUS <sub>X</sub> ←Y-D <sub>i+2</sub> MQADDHUS <sub>D</sub> ←S <sub>i+2</sub> -S <sub>i+2</sub>
LDLFI <sub>L</sub> ←L <sub>i+4</sub> LDFIR <sub>R</sub> ←R <sub>i+4</sub>	LDLFI <sub>L</sub> ←L <sub>i+5</sub> LDFIR <sub>R</sub> ←R <sub>i+5</sub>	MBTOH <sub>L</sub> ←F <sub>i+1</sub> F <sub>i+2</sub> MBTOH <sub>R</sub> ←G <sub>i+1</sub> G <sub>i+2</sub> MQSUBHUS <sub>F</sub> ←G <sub>i+1</sub> X MQSUBHUS <sub>G</sub> ←F <sub>i+1</sub> Y MQADDHUS <sub>X</sub> ←Y-D <sub>i+1</sub> MQADDHUS <sub>D</sub> ←S <sub>i+1</sub> -S <sub>i+1</sub>	MBTOH <sub>L</sub> ←F <sub>i+2</sub> F <sub>i+3</sub> MBTOH <sub>R</sub> ←G <sub>i+2</sub> G <sub>i+3</sub> MQSUBHUS <sub>F</sub> ←G <sub>i+2</sub> X MQSUBHUS <sub>G</sub> ←F <sub>i+2</sub> Y MQADDHUS <sub>X</sub> ←Y-D <sub>i+2</sub> MQADDHUS <sub>D</sub> ←S <sub>i+2</sub> -S <sub>i+2</sub>
LDLFI <sub>L</sub> ←L <sub>i+4</sub> LDFIR <sub>R</sub> ←R <sub>i+4</sub>	LDLFI <sub>L</sub> ←L <sub>i+5</sub> LDFIR <sub>R</sub> ←R <sub>i+5</sub>	MBTOH <sub>L</sub> ←F <sub>i+1</sub> F <sub>i+2</sub> MBTOH <sub>R</sub> ←G <sub>i+1</sub> G <sub>i+2</sub> MQSUBHUS <sub>F</sub> ←G <sub>i+1</sub> X MQSUBHUS <sub>G</sub> ←F <sub>i+1</sub> Y MQADDHUS <sub>X</sub> ←Y-D <sub>i+1</sub> MQADDHUS <sub>D</sub> ←S <sub>i+1</sub> -S <sub>i+1</sub>	MBTOH <sub>L</sub> ←F <sub>i+2</sub> F <sub>i+3</sub> MBTOH <sub>R</sub> ←G <sub>i+2</sub> G <sub>i+3</sub> MQSUBHUS <sub>F</sub> ←G <sub>i+2</sub> X MQSUBHUS <sub>G</sub> ←F <sub>i+2</sub> Y MQADDHUS <sub>X</sub> ←Y-D <sub>i+2</sub> MQADDHUS <sub>D</sub> ←S <sub>i+2</sub> -S <sub>i+2</sub>

図 6 FR500 でメディア演算命令を用いたカーネルコード

Fig. 6 Kernel code with media instructions at the FR500.

に累積する．各命令の動作は次のとおりである<sup>8)</sup>．

**LDLFI** Load FR register( Immediate)．メモリ  $L[i]$  から FR レジスタ  $L_i$  へ 4 バイトデータを転送する．FR は、浮動小数点演算やメディア演算で用いる 32 bit レジスタである．

**MBTOH** Byte To Halfword.  $FRL_i$  上の 32 bit 値を各 8 bit ずつ区切り、16 bit 値 4 つに変換し、2 個の FR  $F_i, F_{i+1}$  に格納する．

**MQSUBHUS** Quad Subtract Unsigned Halfword with Saturation. 2 個の FR を用い、 $F_i, F_{i+1}$  と  $G_i, G_{i+1}$  間において 4 つの 16 bit 整数の飽和付き SIMD 減算を行う．

**MQADDHUS** Quad Add Unsigned Halfword with Saturation. 2 個の FR を用い、 $F_i, F_{i+1}$  と  $G_i, G_{i+1}$  間において 4 つの 16 bit 整数の飽和付き SIMD 加算を行う．

図 6 に、FR500 におけるカーネルコードを示す．実線で囲った部分が、メディア演算命令を使用して差の絶対値を求める基本ブロックであり、1 行が 1 つの VLIW 命令に対応する．1 つの基本ブロックが 2 画素ずつを並列実行している．さらに、ソフトウェアパイプラインングによってこの基本ブロックを 2 つずつオーバーラップさせることにより、最大 4 命令(図 6 の網掛け部分)を同時実行するようにコーディングを行った．

3.2.2 FR550

FR550 は整数演算命令と、浮動小数点演算命令またはメディア命令をそれぞれ最大 4 個同時発行できる<sup>9)</sup>．

FR550 は、整数演算命令およびメディア演算命令をそれぞれ 4 命令並列実行できるものの、ロード命令等の一部の命令は 2 個までしか同時発行できない．そのため、FR500 での 4 並列プログラムをそのまま 8 並列に拡張することはできない．そこで、新たな工夫が必要となる．飽和加減算命令は 2 個までしか同時発行

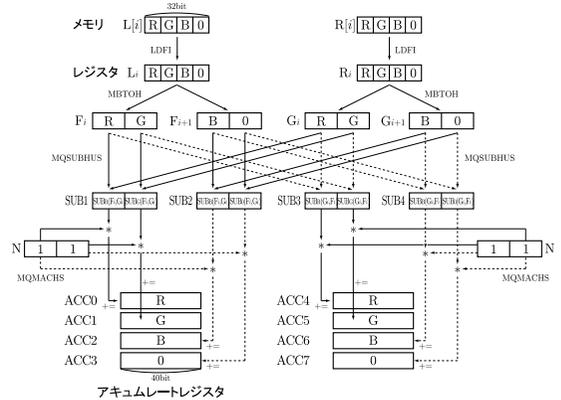


図 7 FR550 において積和演算命令を用いた場合の比較処理の流れ Fig. 7 Flow of comparison processing with MQMACHS at the FR550.

できないため、メディア演算命令の残りの 2 スロットには、他のメディア演算命令と並列実行可能なメディアクワッド積和演算命令 MQMACHS を使用した．飽和減算により求めた値を本命令を用いて累積することによって、差の絶対値の総和を求めることができる．

図 7 に、メディアクワッド積和演算命令を用いた場合の比較処理の流れを示す．左右の小さいウィンドウから 1 画素分の 32 bit データ  $L[i], R[i]$  を LDFI 命令によりレジスタ  $L_i, R_i$  にロードし、MBTOH 命令により各 8 bit の RGB 値を 16 bit に拡張し、左右それぞれレジスタ  $F_i, F_{i+1}$  と  $G_i, G_{i+1}$  に格納する．次に MQSUBHUS 命令により、各 RGB 成分について飽和減算を行う．たとえば R 成分の場合、まず、MQSUBHUS 命令で左右の画素の R 成分の差

$$SUB_R(F_i, G_i) = F_i - G_i \tag{5}$$

$$SUB_R(G_i, F_i) = G_i - F_i \tag{6}$$

を求め、それぞれレジスタ SUB1 と SUB3 の上位 16 bit に格納する．さらにこれらの和を求めなければならないものの、飽和減算命令によってメディア演算命令スロットが埋まっているため、飽和加算命令 MQADDHUS は使えない．そこで、積和演算命令の代わりに、メディアクワッド積和演算命令を用いる．この命令は本来、行列の内積等を計算する命令であるものの、乗数を 1 にすることにより、加算命令として使うことができる．すなわち、上下 16 bit に 1 を格納したレジスタ N を用意し、SUB1 と N との積和計算により、SUB1 の上下 16 bit 値をそのままアキュムレートレジスタに累積できる．以上の方法により、R 成分の差を累積する．G, B 成分についても同様である．最後に各アキュムレータの値を足し合わせることで、小さいウィンドウ中の画素差の総和を求めるこ

ADD	ADDR				MOSUBHS8	MOSUBHS16	MQMACHS1	MQMACHS14
LDFU1	LDFR1				MQADHUSX	MQADHUS2	MQMACHS2	MQMACHS12
LDFU2	LDFR2				MQADHUSY	MQADHUS3	MQMACHS3	MQMACHS13
LDFU3	LDFR3						MQMACHS4	MQMACHS14
LDFU4	LDFR4	MBTOH1	MBTOHR1				MQMACHS5	MQMACHS15
LDFU5	LDFR5	MBTOH2	MBTOHR2		MOSUBHS1	MOSUBHS11	MQMACHS6	MQMACHS16
LDFU6	LDFR6	MBTOH3	MBTOHR3		MOSUBHS2	MOSUBHS12	MQMACHS7	MQMACHS17
LDFU7	LDFR7	MBTOH4	MBTOHR4		MOSUBHS3	MOSUBHS13	MQMACHS8	MQMACHS18
LDFU8	LDFR8	MBTOH5	MBTOHR5		MOSUBHS4	MOSUBHS14	MQMACHS9	MQMACHS19
		MBTOH6	MBTOHR6		MOSUBHS5	MOSUBHS15	MQMACHS10	MQMACHS20
		MBTOH7	MBTOHR7		MOSUBHS6	MOSUBHS16	MQMACHS11	MQMACHS21
		MBTOH8	MBTOHR8		MOSUBHS7	MOSUBHS17	MQMACHS12	MQMACHS22
					MOSUBHS8	MOSUBHS18	MQMACHS13	MQMACHS23
					MQADHUSX	MQADHUS2	MQMACHS14	MQMACHS24
					MQADHUSY	MQADHUS3	MQMACHS15	MQMACHS25
							MQMACHS16	MQMACHS26
							MQMACHS17	MQMACHS27
							MQMACHS18	MQMACHS28
							MQMACHS19	MQMACHS29
							MQMACHS20	MQMACHS30

図 8 FR550 でメディア演算命令を用いたカーネルコード  
Fig. 8 Kernel code with media instructions at the FR550.

とができる。各命令の動作は次のとおりである<sup>10)</sup>。  
**ADD** 32 bit 汎用レジスタ GR どちらの和を求める。ここでは、スモールウィンドウの、ある行の先頭を指すポインタを移動させるために用いる。  
**MQMACHS** Quad Multiply and Accumulate Signed Halfword. 2 個の FR を用いて、4 つの符号付き 16 bit 数のメディアクワッド積和演算を行い、結果を 40 bit のアキュムレートレジスタに足しこむ。レジスタ SUB1 と N 間の積和演算とは、SUB1 の上位 16 bit を SUB1<sub>u</sub>、下位 16 bit を SUB1<sub>l</sub> 等とすると、  

$$SUB1_u \cdot N_u + SUB1_l \cdot N_l \quad (7)$$
 を求める演算である。

図 8 に、FR550 におけるカーネルコードを示す。実線で囲った部分が、メディア演算命令を使用してスモールウィンドウの 1 行分の差の絶対値を求める基本ブロックであり、1 行が 1 つの VLIW 命令に対応する。図 8 において、ADD L は、メモリ配列 L[i] の先頭をスモールウィンドウのある行に移動することを意味する。LDFI L<sub>i</sub> は、メモリ L[i] からレジスタ L<sub>i</sub> へ 32 bit データをロードすることを意味する。MBTOH L<sub>i</sub> は、L<sub>i</sub> の各 8 bit の RGB 値を 16 bit に拡張し、レジスタ F<sub>i</sub>、F<sub>i+1</sub> に格納する。MQSUBHUS S<sub>i</sub> および MQSUBHUS T<sub>i</sub> は、それぞれ、

$$S_i = SUB(F_i, G_i) = F_i - G_i \quad (8)$$

$$T_i = SUB(G_i, F_i) = G_i - F_i \quad (9)$$

を計算する。MQADHUS X 等は、

$$X = S_{i+5} + S_{i+6}, Y = S_{i+7} + S_{i+8} \quad (10)$$

$$Z = T_{i+5} + T_{i+6}, U = T_{i+7} + T_{i+8} \quad (11)$$

を計算し、MQMACHS S<sub>i</sub> 等は、S<sub>i</sub> の値をアキュムレートレジスタへ累積することに対応する。さらに、ソフトウェアパイプラインングにより基本ブロックを 2 つずつオーバーラップさせ、最大 6 命令(図の網掛け部分)を同時実行するようにコーディングを行った。

### 3.3 Intel IA-32 における高速化

#### 3.3.1 MMX, SSE

MMX 命令は、MMX テクノロジー対応 Pentium プロセッサと Pentium II プロセッサから、IA-32 アーキテクチャに導入された SIMD 命令である<sup>11)</sup>。MMX は、MMX テクノロジーレジスタ(以下、MMX レジスタという)または汎用レジスタを用いてパックドバイト、パックドワード、パックドダブルワード、またはクワッドワード整数オペランドを操作することができる。

また、ストリーミング SIMD 拡張命令(SSE)は、Pentium III プロセッサファミリから IA-32 アーキテクチャに導入された命令である。これらの拡張命令は、高度な 2D および 3D グラフィックス、モーションビデオ、画像処理、音声認識、音声合成、テレフォニ、およびビデオ会議等のアプリケーション用に、IA-32 プロセッサのパフォーマンスを強化するために開発されたものである。SSE では、64 bit MMX レジスタ、64 bit パックド整数データ型、およびパックド整数に対して SIMD 演算を実行する命令が用意されている。また、SSE は MMX の SIMD 実行モデルを拡張したものであり、128 bit レジスタ内のパックドデータおよびスカラー単精度浮動小数点値を処理するための機能が追加されている。

さて、MMX には、差の絶対値を直接求める命令は搭載されていない。このため、MMX のみを用いた場合、FR-V と同様に飽和加減算命令を使用する必要がある。一方、SSE にはバイト整数の絶対差を求める命令 PSADBW が搭載されているため、IA-32 プロセッサの評価には SSE を用いることにした。ただし、SSE の PSADBW 命令は、128 bit の XMM レジスタではなく、64 bit の MMX レジスタを使用するため、ロード等には MMX 命令も用いる必要がある。1 つの PSADBW 命令により 8 つのバイト整数の絶対差を計算することができるので、2 画素分の比較計算を一度に行うことが可能である。

データの流れを図 9 に示す。左右のスモールウィンドウの 2 画素分のデータを、MOVQ 命令によりメモリ L[i]、L[i+1] から MMX レジスタ MM1、R[i]、R[i+1] から MM2 にそれぞれロードし、PSADBW 命令により 64 bit 中の各 8 bit ずつについて絶対差を求め、これらを加算した結果を MM2 に格納する。その後、PADDD 命令によりアキュムレート用レジスタ MM3 に累積する。各命令の動作は次のとおりである。  
**MOVQ** Move Quadword. 64 bit のパックドデータを、メモリから MMX レジスタ(またはその反対方向)に移動するか、MMX レジスタ間を移動させる。

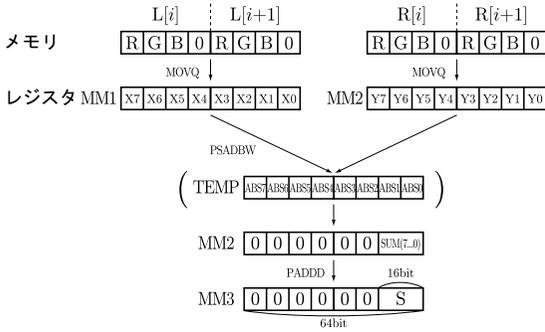


図 9 MMX, SSE を用いた場合のデータの流れ  
 Fig. 9 Data flow with MMX and SSE instructions.

```

movq  L[i], L[i+1]  mm1
movq  R[i], R[i+1]  mm2
psadbw mm1, mm2    mm2
paddq  mm2, mm3    mm3
  
```

図 10 MMX と SSE を用いたカーネルコード  
 Fig. 10 Kernel code with MMX and SSE instructions.

**PADDQ** Add Packed Integers. ラップアラウンドモードを使用して、ソースオペランドとデスティネーションオペランドの対応する符号付きまたは符号なしのデータを加算する。データ型は、パックドダブルワードである。

**PSADBW** Compute Sum of Absolute Differences. 2つのソースオペランドにおいて対応する符号なしバイト整数の絶対差を計算して、これらの差を加算し、得られた和をデスティネーションオペランドの最下位ワードに格納する。MMX レジスタを使用する。

MMX と SSE を用いたカーネルコードを図 10 に示す。本カーネルコードを、スモールウィンドウ 81 画素のうち 80 画素分 (40 回) を行い、残りの 1 画素は単独で計算を行った。なお、Intel IA-32 プロセッサはスーパースカラであり、VLIW のような命令スケジューリングは行っていない。

3.3.2 SSE2

ストリーミング SIMD 拡張命令 2 (SSE2) は、Pentium 4 プロセッサファミリから IA-32 アーキテクチャに導入された命令である。これらの拡張命令は、高度な 3D グラフィックス、ビデオデコーディング・エンコーディング、音声認識、電子商取引、インターネット、科学計算、および工学計算等のアプリケーション向けに、IA-32 プロセッサのパフォーマンスを強化するために開発されたものである。

SSE2 は、MMX や SSE と同様に、SIMD 実行モデルを使用する。SSE2 では、SSE の SIMD 実行モデルが拡張され、パックド 倍精度浮動小数点値と 128 bit

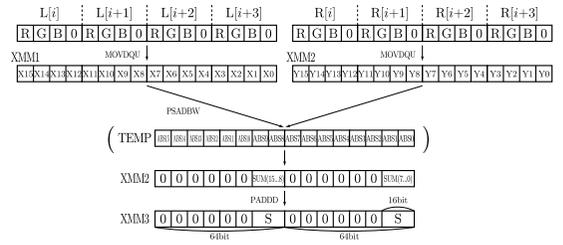


図 11 SSE2 を用いた場合のデータの流れ  
 Fig. 11 Data flow with SSE2 instructions.

```

movdqu L[i], L[i+1], L[i+2], L[i+3]  xmm1
movdqu R[i], R[i+1], R[i+2], R[i+3]  xmm2
psadbw xmm1, xmm2  xmm2
paddq  xmm2, xmm3  xmm3
  
```

図 12 SSE2 を用いたカーネルコード  
 Fig. 12 Kernel code with SSE2 instructions.

パックド 整数の処理がサポートされた。

SSE2 を用いた場合でも、考え方は SSE のときとまったく同じである。SSE2 の PSADBW 命令は SSE のものを 128 bit に拡張したものであるため、16 個のバイト整数の絶対差が計算でき、4 画素分の比較計算を行うことができる。

データの流れを図 11 に示す。左右のスモールウィンドウの 4 画素分のデータを、MOVQDU 命令によりメモリ L[i], L[i+1], L[i+2], L[i+3] から XMM レジスタ XMM1 に、また、R[i], R[i+1], R[i+2], R[i+3] から XMM2 にそれぞれロードし、PSADBW 命令で 128 bit データ中の各 8 bit ずつの絶対差を求め、これらを加算した結果を XMM2 に格納する。その後、PADDQ 命令によりアキュムレート用レジスタ XMM3 に累積する。各命令の動作は次のとおりである。

**MOVQDU** Move Unaligned Double Quadword. メモリから XMM レジスタ、XMM レジスタからメモリ、または XMM レジスタ間について、ダブルクワッドワードオペランドを転送する。

**PSADBW** SSE の PSADBW を 128 bit に拡張したものである。SSE レジスタを使用する。

**PADDQ** MMX の PADDQ を 128 bit に拡張したものである。SSE レジスタを使用する。

SSE2 を用いたカーネルコードを図 12 に示す。スモールウィンドウ 81 画素のうち 80 画素分 (20 回) を行い、残りの 1 画素は SSE の PSADBW 命令を用いて単独で SAD を行った。SSE と同様に、命令スケジューリングは行っていない。

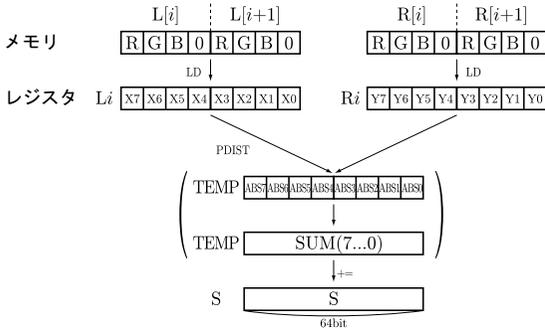


図 13 VIS を用いた場合のデータの流れ  
Fig. 13 Data flow with VIS instructions.

### 3.4 SPARC における高速化

SPARC プロセッサには、VIS ( Visual Instruction Set ) が搭載されている<sup>12)</sup>。Intel の SSE 等と同様に、SIMD 型の並列演算、データのパック/アンパック、bit 操作等の命令がある。そこで、SPARC プロセッサでも同様のメディア演算命令を用いた距離画像生成プログラムを作成した。

SSE の PSADBW 命令と同様に、UltraSPARC の VIS にも PDIST 命令が用意されている。これは、64 bit レジスタ中の各 8 bit ずつ絶対差を計算し、総和を求める命令である。SSE の PSADBW 命令と異なるのは、計算結果を第 3 オペランドであるデスティネーションレジスタに足しこむ点である。SAD 計算の後に加算命令を必要としないため、全体のステップ数を減らせる利点がある。

図 13 に、VIS を用いた場合のデータの流れを示す。左右のスマールウィンドウの 2 画素分のデータを、LD 命令によりメモリ  $L[i]$ 、 $L[i+1]$  をからレジスタ  $L_i$  に、また、 $R[i]$ 、 $R[i+1]$  から  $R_i$  にそれぞれロードし、PDIST 命令により 64 ビット中の各 8 ビットずつ SIMD の絶対差を求め、これらを加算した結果をアキュムレート用レジスタ S に累積する。すなわち、PDIST 命令は、

$$S \leftarrow S + \text{SUM}(7 \dots 0) \quad (12)$$

を求める命令である。

図 14 に、VIS を用いたカーネルコードを示す。左右のスマールウィンドウの 2 画素分のデータを、LD 命令によりメモリ  $L[i]$ 、 $L[i+1]$  をからレジスタ  $L_i$  に、また、 $R[i]$ 、 $R[i+1]$  から  $R_i$  にそれぞれロードし、PDIST 命令により 64 bit 中の各 8 bit ずつ SIMD の絶対差を求め、これらを加算した結果をアキュムレート用レジスタ S に累積する。

ld	$L[i]$ 、 $L[i+1]$	$L_i$
ld	$R[i]$ 、 $R[i+1]$	$R_i$
pdist	$L_i$ 、 $R_i$ 、S	S

図 14 VIS を用いたカーネルコード  
Fig. 14 Kernel code with VIS instructions.

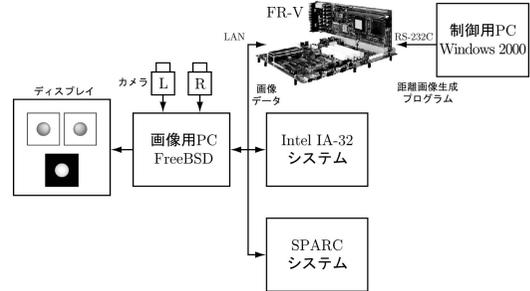


図 15 システム構成  
Fig. 15 System configuration.

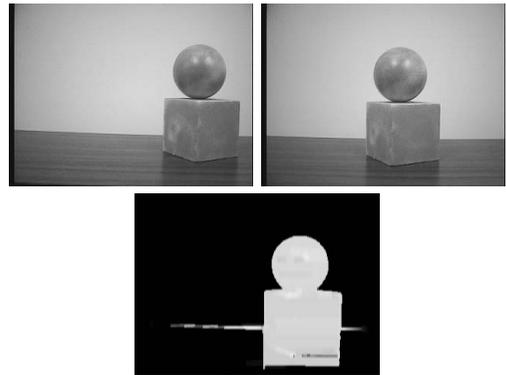


図 16 入力ステレオ画像と得られた距離画像  
Fig. 16 Input stereo images and output depth image.

## 4. 評価

### 4.1 評価システム

距離画像生成システムは、2 台のカメラで撮影したステレオ画像をいったん PC ( Pentium III 1 GHz , FreeBSD ) に取り込み、画像データを LAN 経由で各種プロセッサシステムへ転送し、各プロセッサ上で画像処理を行った後、処理結果を PC へ戻し、ディスプレイに表示する構成としている。FR-V 上で動作するプログラムは Windows 2000 上の富士通製統合開発環境 SOFTUNE により作成し、RS-232C 経由で FR-V へ転送する。システム構成を図 15 に示す。

図 16 に、測定に用いたステレオ画像および生成した距離画像を示す。以下では、カーネル部分であるスマールウィンドウの比較処理の実行時間を測定し、処理に必要な消費電力量を求めた。

表 1 に、測定環境を示す。表中の参考電力は、各ブ

表 1 測定環境<sup>13)~15)</sup>  
Table 1 Environment.

プロセッサ	[Hz]	参考電力 [W]	消費電力の測定方法	命令	最適化
FR500	240 M	1.8	CPU ボードへの +3.35 V 電源供給線	一般 Media	SOFTUNE (hand coding)
FR550	466 M	2.19	CPU ボードへの +5.10 V 電源供給線	一般 Media	SOFTUNE (hand coding)
Mobile Pentium III	700 M	15.0	マザーボードへの +16 V 電源供給線	一般 MMX, SSE	gcc -O2 (hand coding)
Xeon	2.8 G	74.0	CPU への +12 V 電源供給線	一般 SSE2	gcc -O2 (hand coding)
UltraSPARC IIe	502 M	13	マザーボードへの +5 V および +3.3 V 電源供給線	一般 VIS	gcc -O2 (hand coding)

表 2 測定結果  
Table 2 Results.

プロセッサ	一般命令			Media 命令		
	時間 [s]	消費電力 [W]	$Fps^2/Watt$	時間 [s]	消費電力 [W]	$Fps^2/Watt$
FR500	70.0	4.49	$4.55 \times 10^{-5}$	8.00	4.62	$3.38 \times 10^{-3}$
FR550	12.5	3.57	$1.79 \times 10^{-3}$	1.91	2.35	$1.17 \times 10^{-1}$
Mobile Pentium III	22.8	13.3	$1.45 \times 10^{-4}$	1.96	11.8	$2.21 \times 10^{-2}$
Xeon	6.05	72.0	$3.79 \times 10^{-4}$	0.43	64.8	$8.35 \times 10^{-2}$
UltraSPARC IIe	33.0	21.2	$4.33 \times 10^{-5}$	3.50	21.2	$3.85 \times 10^{-3}$

ロセッサの消費電力のカタログ値である<sup>13)~15)</sup>。実際に評価に用いた消費電力は、プログラム動作中に電流計により電流値を計測し、電圧を乗じることによって求めた。プロセッサ単体の消費電力を正確に測定することは困難であるため、プログラムが動作可能な最小限度の I/O 構成としたうえで、可能な限りプロセッサに近い位置での DC-DC コンバータの手前における電流測定値を用いて、消費電力を比較することにした。

具体的には、FR500 は、マザーボードから CPU ボード (CPU, 補助 LSI, メモリ) への +3.35 V 電源供給線の電流値を測定した。FR550 は、マザーボードから CPU ボード (CPU, 補助 LSI, メモリ) への +5.10 V 電源供給線の電流値を測定した。Mobile Pentium III は、富士通製ノート PC FMV6700MF9/X を用い、液晶バックライトを消し内蔵ハードディスクは残した状態で、AC アダプタからマザーボードへの +16 V 電源供給線の電流値を測定した。Xeon は、SUPER-MICRO 製マザーボード SUPER P4DPR-6GM+ を用い、CPU への +12 V 専用電源供給線 (ATX12V) の電流値を測定した。UltraSPARC IIe は、SUN 製 Blade100 を用い、PCI カードをすべて抜き内蔵ハードディスクは残した状態で、ATX 電源からマザーボードへの +5 V および +3.3 V 電源供給線の電流値をそれぞれ測定し電力を合計した。

実際のプロセッサ単体での消費電力はこれより少なくなると見込まれるが、一般的な比較としては、ほぼ

正しい傾向を導き出せる程度の精度は確保できていると考えている。

FR-V の評価には、C プログラムを SOFTUNE で最適化したものと、メディア演算命令によるものを用いた。また、汎用スーパースカラプロセッサの評価には、C プログラムを gcc -O2 で最適化したものと、メディア演算命令によるものを用いた。なお、OS は、Intel IA-32 は FreeBSD, SPARC は Solaris である。

#### 4.2 測定結果および考察

測定結果を表 2 に示す。実行時間を図 17、消費電力量を図 18 に示す。まず、図 17 から、どのプロセッサにおいても、メディア演算命令を用いることによって距離画像生成処理能力が飛躍的に向上していることが分かる。通常のプログラミング言語により記述されたプログラムに対し、メディア演算命令を用いた効率の良い命令列を生成することは、現状の FR-V コンパイラでは、まだ実現されていない。一方、C 言語により記述されたプログラムから SSE 等を用いた命令列を生成するコンパイラには Intel C++ Compiler<sup>16)</sup> があるものの、ハンドコーディングに勝る性能を出すことは現状では不可能である。ハンドコーディングには非常に手間がかかるため、自動的に効率の良い SIMD 命令列を生成するコンパイラが望まれる。

また、図 18 より、消費電力量は FR550 が最も小さいことが分かる。

さて、低消費電力の評価尺度には、命令実行速

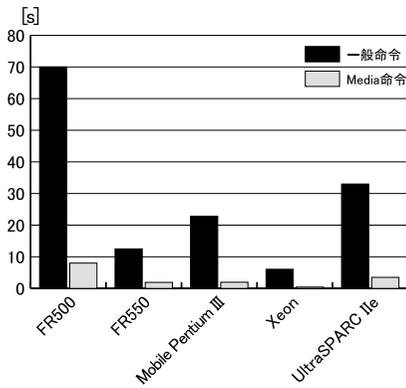


図 17 実行時間 [s]  
Fig. 17 Execution time.

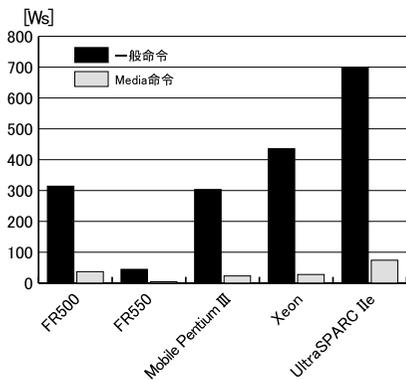


図 18 消費電力量 [Ws]  
Fig. 18 Energy [Ws].

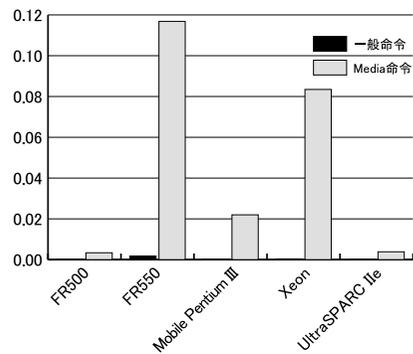


図 19  $Fps^2/Watt$   
Fig. 19  $Fps^2/Watt$ .

	LDRL#6	LDAR#6	MBTOHL#3	MBTOHR#3	POIST#1
	LDRL#7	LDAR#7	MBTOHL#4	MBTOHR#4	POIST#2
ADDL ADDR	LDRL#8	LDAR#8	MBTOHL#5	MBTOHR#5	POIST#3
	LDRL#1	LDAR#1	MBTOHL#6	MBTOHR#6	POIST#4
	LDRL#2	LDAR#2	MBTOHL#7	MBTOHR#7	POIST#5
	LDRL#3	LDAR#3	MBTOHL#8	MBTOHR#8	POIST#6
	LDRL#4	LDAR#4	MBTOHL#1	MBTOHR#1	POIST#7
	LDRL#5	LDAR#5	MBTOHL#2	MBTOHR#2	POIST#8
	LDRL#6	LDAR#6	MBTOHL#3	MBTOHR#3	POIST#1
	LDRL#7	LDAR#7	MBTOHL#4	MBTOHR#4	POIST#2
ADDL ADDR	LDRL#8	LDAR#8	MBTOHL#5	MBTOHR#5	POIST#3
	LDRL#1	LDAR#1	MBTOHL#6	MBTOHR#6	POIST#4
	LDRL#2	LDAR#2	MBTOHL#7	MBTOHR#7	POIST#5
	LDRL#3	LDAR#3	MBTOHL#8	MBTOHR#8	POIST#6
	LDRL#4	LDAR#4	MBTOHL#1	MBTOHR#1	POIST#7
	LDRL#5	LDAR#5	MBTOHL#2	MBTOHR#2	POIST#8
	LDRL#6	LDAR#6	MBTOHL#3	MBTOHR#3	POIST#1

図 20 FR550 に PDIST 相当の命令を追加した場合のカーネルコード

Fig. 20 Kernel code with PDIST at the FR550.

度をどの程度重要視するかによって、 $Mips/Watt$ 、 $Mips^2/Watt$ 、 $Mips^3/Watt$  が考えられる<sup>17)</sup>。本稿では、画像処理速度も考慮に入れるために、 $Mips^2/Watt$  を評価尺度として用いることにする。ただし、比較対象はアーキテクチャや命令数が互いに異なるため、Mips の代わりに、単位時間あたりに処理可能な画面数 (Frames per second) を用いることにする。

$Fps^2/Watt$  は、

$$\frac{(\text{総処理画面数}/\text{実行時間})^2}{\text{総電力量}/\text{実行時間}} \quad (13)$$

すなわち、

$$\frac{\text{総処理画面数}^2}{\text{総電力量} \times \text{実行時間}} \quad (14)$$

であり、各プログラムの総処理画面数を 1 とすると、 $1/(\text{総電力量} \times \text{実行時間})$ 。 (15)

つまり、Energy-Delay 積の逆数を比較しているといえる。

本評価尺度により作成したグラフを図 19 に示す。この結果から、性能と消費電力の両方を考慮した比較

においても FR550 が最も優れているといえる。

さて、FR550 において、スモールウィンドウ全体の 81 画素に対する差の絶対値の総和計算は、机上計算で 141 ステップである。この部分が 5 M 回実行されるので 705 M ステップを要する。466 MHz で 1.91 秒かかるため 890 M サイクル動いたことになる。よって CPI (Cycle per Instruction, ただし VLIW 命令を 1 命令と数えている) は 1.26 となった。

ところで、FR-V には VIS の PDIST 命令のような差の絶対値を直接求める命令がないため、その計算と結果の累積に 4 命令を要したものの、PDIST 命令に相当する専用命令がある場合には 1 命令により実行できる。図 20 に、FR550 に PDIST 命令に相当する命令を追加した場合のカーネルコードを示す。この命令のデータハザードによるペナルティを 3 サイクルと仮定すると、スモールウィンドウ 81 画素に対する差の絶対値の総和計算は、107 サイクルとなる。PDIST 相当の命令がない場合は机上計算で 148 サイクルであり、この命令を追加することにより、サイクル数を約 28% 減少させることができる。ペナルティサイクル数

が増加しても、ソフトウェアパイプラインングによりその影響を抑えることができる。

さらに、ロード命令と MBTOH 命令が 4 命令、PDIST 命令に相当する命令が 2 命令並列実行できると仮定すると、スモールウィンドウ全体に対する差の絶対値の総和計算は 94 サイクルとなり、約 36% のサイクル数の減少が期待できる。この場合、演算器が増えることにより消費電力が増加するものの、仮に消費電力が 2 倍になると見積もっても、 $Fps^2/Watt$  の値は  $1.42 \times 10^{-1}$  となり、PDIST 相当の命令を追加することにより、さらに、 $Fps^2/Watt$  を向上することができる。

## 5. おわりに

本稿では、VLIW 型メディアプロセッサ FR-V や汎用スーパースカラプロセッサを用いた距離画像生成について、処理速度や消費電力の観点から比較を行った。

その結果、いずれのプロセッサにおいてもメディア演算命令を用いることによって処理速度が飛躍的に向上することを示した。また、性能および消費電力に関する評価尺度  $Fps^2/Watt$  についても FR550 が最も優れていることを示した。以上のことと、FR-V が冷却装置を必要としないことを考慮すると、低消費電力を保ちつつ高い処理能力が要求されるシステムへの組み込み用プロセッサには FR-V が適しているといえる。

さらに、FR550 については、VIS の PDIST 命令に相当する専用命令を追加することにより、サイクル数を約 28% 削減でき、また、ロード命令と MBTOH 命令が 4 命令、PDIST 命令に相当する命令を 2 命令並列実行することが可能であれば、サイクル数を約 36% 削減できることを示した。

本稿では、性能評価として  $Fps^2/Watt$  を用いたが、モバイルシステムにおいてはチップ面積コストも重要な問題であるため、チップ面積を考慮した  $Fps/cm^2$  も性能の評価尺度として考えられる。また、FR-V 以外の組み込み用プロセッサを用いた性能評価も必要である。これらの評価は今後の課題である。

謝辞 本研究でお世話になった富士通研究所システム LSI 開発研究所第二開発プロジェクト部長の高橋宏政氏に心から感謝いたします。

## 参 考 文 献

1) Kanade, T.: A Stereo Machine for Video-Rate Dense Depth Mapping and Its New Applications, *Proc. 15th Computer Vision and Pattern Recognition Conference (CVPR)* (1996).

2) SONY 木原研究所: Entertainment Vision Sensor, <http://www.sony.co.jp/SonyInfo/News/Press/200202/02-0207/> (2002).

3) CANESTA: <http://www.canesta.com/> (2002).

4) TYZX: Real-time Stereo Vision for Real-world Object Tracking (2002).

5) United States Patent No.US 6,215,898 B1 (2001).

6) Okutomi, M.: A Multiple-Baseline Stereo, *IEEE Transaction on Pattern Analysis and Machine Intelligence*, Vol.15, No.4 (1993).

7) 富士通: FR-V Family FR500 シリーズ MB 93501 LSI 仕様書 (2001).

8) 富士通: FR-V Family FR500 Series Instruction Set Manual (2001).

9) 富士通: FR-V Family FR550 シリーズ MB 93551 LSI 仕様書 (2002).

10) 富士通: FR-V Family FR550 Series Instruction Set Manual (2002).

11) Intel: IA-32 Intel Architecture Software Developer's Manual (2001).

12) Sun Microsystems: The VIS Instruction Set, ver.1.0 (Jun. 2002).

13) 富士通: FR-V Series, <http://www.fme.fujitsu.com/products/micro/fr/> (2002).

14) Intel: Intel Architecture Processors, <http://developer.intel.com/design/processor/> (2002).

15) Sun: UltraSPARC Processors, <http://www.sun.com/processors/> (2002).

16) Intel: Intel Software Development Products, <http://developer.intel.com/software/products/compilers/> (2002).

17) Brook, D.: Power-Aware Microarchitecture: Design and Modeling Challenges for Next-Generation Microprocessors, *IEEE MICRO*, Nov/Dec, pp.26-44 (2000).

(平成 15 年 2 月 3 日受付)

(平成 15 年 5 月 8 日採録)



清水 雄歩 (学生会員)

1979 年生。2003 年京都大学工学部情報学科卒業。現在、同大学大学院情報学研究科通信情報システム専攻修士課程在籍。マルチメディア、ネットワーク等に興味を持つ。



津邑 公暁(正会員)

1973年生。1996年京都大学工学部情報工学科卒業。1998年同大学大学院工学研究科情報工学専攻修士課程修了。2001年同大学大学院情報学研究科博士後期課程単位取得退学。同年より同大学経済学研究科助手、現在に至る。計算機アーキテクチャ、並列処理応用、脳型情報処理等に興味を持つ。人工知能学会、日本神経回路学会各会員。



中島 康彦(正会員)

1963年生。1986年京都大学工学部情報工学科卒業。1988年同大学院修士課程修了。同年富士通(株)入社。スーパーコンピュータVPPシリーズのVLIW型CPU、命令エミュレーション、高速CMOS回路設計等に関する研究開発に従事。工学博士。1999年京都大学総合情報メディアセンター助手。同年同大学院経済学研究科助教授、現在に至る。2002年より(兼)科学技術振興事業団さきがけ研究21(情報基盤と利用環境)。計算機アーキテクチャに興味を持つ。IEEECS, ACM各会員。



五島 正裕(正会員)

1968年生。1992年京都大学工学部情報工学科卒業。1994年同大学大学院工学研究科情報工学専攻修士課程修了。同年より日本学術振興会特別研究員。1996年京都大学大学院工学研究科情報工学専攻博士後期課程退学、同年より同大学工学部助手。1998年同大学大学院情報学研究科助手。高性能計算機システムの研究に従事。2001年情報処理学会山下記念研究賞、2002年同学会論文賞受賞。IEEE会員。



森 眞一郎(正会員)

1963年生。1987年熊本大学工学部電子工学科卒業。1989年九州大学大学院総合理工学研究科情報システム専攻修士課程修了。1992年同大学大学院総合理工学研究科情報システム専攻博士課程単位取得退学。同年京都大学工学部助手。1995年同助教授。1998年同大学大学院情報学研究科助教授。工学博士。並列/分散処理、可視化、計算機アーキテクチャの研究に従事。IEEE, ACM各会員。



北村 俊明(正会員)

1955年生。1978年京都大学工学部情報工学科卒業。1983年同大学大学院博士課程研究指導認定退学。同年富士通(株)入社。汎用コンピュータ、スーパーコンピュータVPPシリーズのVLIW型CPU、Mアーキテクチャ・命令エミュレーション、米国HAL社においてSPARCプロセッサ等の研究開発に従事。工学博士。2000年京都大学総合情報メディアセンター助教授。2002年広島市立大学情報科学部教授、現在に至る。計算機アーキテクチャに興味を持つ。電子情報通信学会、IEEE, ACM各会員。



富田 眞治(正会員)

1945年生。1968年京都大学工学部電子工学科卒業。1973年同大学大学院博士課程修了。工学博士。同年京都大学工学部情報工学教室助手。1978年同助教授。1986年九州大学大学院総合理工学研究科教授、1991年京都大学工学部教授、1998年同大学院情報学研究科教授、現在に至る。計算機アーキテクチャ、並列処理システム等に興味を持つ。著書「並列コンピュータ工学」(1996)、「コンピュータアーキテクチャ第2版」(2000)等。電子情報通信学会、IEEE, ACM各会員。平成7, 8年度, 10, 11年度本会理事。平成13, 14年度同関西支部長。