計測自動制御学会東北支部 第 295 回研究集会 (2015.6.26)
資料番号 295-2

OpenCL を用いた FPGA ベース画像処理プロセッサの設計

Design of an FPGA-Based Image Processor Using OpenCL

立見駿介, ○張山昌論, 伊藤康一, 青木孝文

Shunsuke TATSUMI, Masanori HARIYAMA, Koichi ITO, Takafumi AOKI

東北大学大学院情報科学研究科,

Graduate School of Information Sciences, Tohoku University,

キーワード： 画像処理 (image processing), 位相限定相関 (Phase-only correlation), 対応付け (correspondence matching), FPGA (FPGA)

連絡先： 〒 980-8579 仙台市青葉区荒巻字青葉 6-6-05 東北大学大学院情報科学研究科 張山昌論,
Tel.: (022)795-7153, Fax.: (022)263-9167, E-mail: hariyama@ecei.tohoku.ac.jp

1. はじめに

コンピュータビジョンにおいて，画像対応付けは重要な基礎技術の 1 つである [1]．特にステレオビジョンに基づく 3 次元計測において，ステレオ画像の対応付けは重要な役割を持つ．3 次元計測では，計測精度が対応付け精度に依存するため，高精度な対応付け手法が必要となる．また，対応付けた点数だけ計測物体の 3 次元点が得られるため，物体の詳細な構造を求めるためには密な対応付けが必要となる．さらに，計測にかかる処理時間の大部分は対応付け処理で占められるため，高速な対応付けが実際の応用で求められている．

本稿では，高精度かつ密な計測を行うために，Phase-Only Correlation (POC) [2] を用いた画像対応付け手法を扱う．POC は，画像の離散フーリエ変換から得られる位相成分を利用した対応付け手法である．しかし，POC は，離散フーリエ変換の計算コストが大きいため，密な計測を行う際に対応付けの計算コストが非常に大きくなることが問題であった．そのため，対応付け処理を CPU にマルチスレッドで実装した場合でも実時間処理が達成できなかった．さらに，CPU 実装では消費電力が大きいという問題もあった．この問題の解決策として，OpenCL というフレームワークを用いて，処理を GPU (Graphics Processing Unit) に実装する手法が提案されている [3]．GPU 実装により，CPU 実装と比べて最大で 5 倍程度の高速化が行え，実時間処理が可能であることが示されている．一方で消費電力が大きいという問題は依然として残されている．

本稿では，対応付けを高速かつ低消費電力で行うために，POC による画像対応付け処理の FPGA (Field Programmable Gate Array) 実装を提案するとともに，FPGA を用いてステレオビジョンに基づく 3 次元計測システムを試作する．FPGA アクセラレータを効率よく設計するために，OpenCL ベースの設計ツールを用いて GPU 向けのコードを再利用する．OpenCL とは，CPU と GPU のような，ヘテロジニアスな

演算器環境において並列プログラミングを行うためのフレームワークである [4]．これによりタスク並列とデータ並列に基づく効率の良い並列演算が実現できる．近年，Altera 社は FPGA 向けの OpenCL ベース設計ツールの提供を開始した [5]．しかし，FPGA 向け OpenCL では，GPU 向け OpenCL のコードを再利用することはできるが，GPU と FPGA のアーキテクチャは大きく異なるため，FPGA 向けにコードを最適化しなければならないことが問題となる．そこで本稿では，FPGA 向け OpenCL の設計手法について，データの再利用やパイプライン等の最適化について述べる．また，FPGA 実装によって，GPU と比べて同程度の処理速度かつ非常に低消費電力で POC による画像対応付けが行えることを示す．これにより，FPGA を用いることで高精度な対応付けが組込みシステムでも利用できるようになる．

2. 位相情報に基づく画像対応付け手法

Phase-Only Correlation (POC) 関数について簡単に説明する [6, 7]．$f(n)$ と $g(n)$ は画像の 1 次元信号であり，信号長は $N = 2M + 1$ $(-M \leqq n \leqq M)$ であるとする．このとき，正規化クロスパワースペクトル $R(k)$ は次のように定義される．

$$R(k) = \frac{F(k)\overline{G(k)}}{|F(k)\overline{G(k)}|} = e^{j(\theta_F(k) - \theta_G(k))} \qquad (1)$$

ここで，$F(k)$ と $G(k)$ はそれぞれ $f(n)$ と $g(n)$ を離散フーリエ変換したものであり，$\overline{G(k)}$ は $G(k)$ の複素共役である．また $-M \leqq k \leqq M$ である．1 次元の POC 関数 $r(n)$ は，$R(k)$ を逆離散フーリエ変換することで得られる．2 枚の画像が似ている場合，POC 関数は極めて鋭いピークを示す．一方で 2 枚の画像が似ていない場合，ピークの高さは小さくなる．つまり，ピークの高さは画像対応付けにおいて類似度の尺度を表し，ピーク位置は画像間の平行移動量を表している．さらに，サブピクセルレベルの対応付け精度向上のために，以下の高精度化手法を用いる: (i) 関数フィッティング，(ii) 入力信号へ窓関数の適用，(iii) スペクトル重み付け，(iv) 1 次元信号の足し合わせ [2]．

ステレオ画像を平行化すると，基準点と対応点の座標の違いは水平方向のみに制限される [1]．そのため，1 次元 POC を用いて低計算コストで高精度な対応付けができる．ステレオ画像から正確な対応を得るため，サブピクセルレベルの対応付けを行う．さらに，ロバストな対応付けのために画像ピラミッドを用いて粗密探索を行う (図 1) [2]．$\boldsymbol{p}$ を画像 $I(n_1, n_2)$ における基準点の位置ベクトルとする．サブピクセルレベルの対応付け問題とは，画像 $I(n_1, n_2)$ 内の基準点 $\boldsymbol{p}$ に対応する，画像 $J(n_1, n_2)$ 内の対応点 $\boldsymbol{q}$ の実数の位置ベクトルを求めることである．以下に簡単な説明を示す．

Step 1: 各レイヤ $l = 1, 2, \cdots, l_{\max} - 1$ において，レイヤ l の画像である $I_l(n_1, n_2)$ と $J_l(n_1, n_2)$ を，以下に示すように再帰的に求める．

$$\begin{aligned} I_l(n_1, n_2) &= \frac{1}{4}\sum_{i_1=0}^{1}\sum_{i_2=0}^{1} I_{l-1}(2n_1 + i_1, 2n_2 + i_2) \\ J_l(n_1, n_2) &= \frac{1}{4}\sum_{i_1=0}^{1}\sum_{i_2=0}^{1} J_{l-1}(2n_1 + i_1, 2n_2 + i_2) \end{aligned}$$

Step 2: 各レイヤ $l = 1, 2, \cdots, l_{\max}$ において，原画像の基準点 $\boldsymbol{p}_0$ に対応する基準点 $\boldsymbol{p}_l = (p_{l1}, p_{l2})$ を，以下に示すように再帰的に求める．

$$\boldsymbol{p}_l = \lfloor \tfrac{1}{2}\boldsymbol{p}_{l-1} \rfloor = (\lfloor \tfrac{1}{2}p_{l-1\ 1} \rfloor, \lfloor \tfrac{1}{2}p_{l-1\ 2} \rfloor) \qquad (2)$$

ここで，$\lfloor z \rfloor$ は z を超えない最大の整数を意味する．

Step 3: 最上位レイヤにおいて $\boldsymbol{q}_{l_{\max}} = \boldsymbol{p}_{l_{\max}}$ であるとする．また $l = l_{\max} - 1$ とおく．

Step 4: レイヤ l の画像 $I_l(n_1, n_2)$ と $J_l(n_1, n_2)$ から，$\boldsymbol{p}_l$ と $2\boldsymbol{q}_{l+1}$ を中心とした小さな画像領域（探索ウィンドウ）$f_l(n_1, n_2)$ と $g_l(n_1, n_2)$ をそ

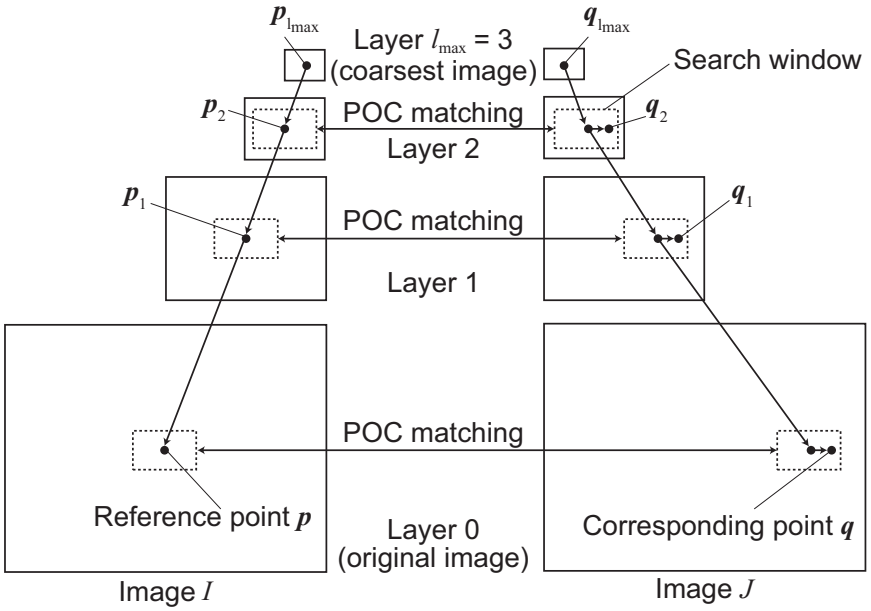

Fig. 1　POC による高精度な画像対応付け手法の概要

れぞれ抜き出す．探索ウィンドウは，信号長 N の 1 次元信号 L 行から構成される．

Step 5: ピクセルレベルの対応付けを用いて $f_l(n_1, n_2)$ と $g_l(n_1, n_2)$ 間の平行移動量を求める．求めた平行移動量を $\boldsymbol{\delta}_l$ とおくと，レイヤ l の対応点 $\boldsymbol{q}_l$ は以下のように定められる．

$$\boldsymbol{q}_l \quad = \quad 2\boldsymbol{q}_{l+1} + \boldsymbol{\delta}_l \tag{3}$$

Step 6: $l = l - 1$ とおく．$l \geq 0$ である間は Step 4 から Step 6 の処理を繰り返す．

Step 7: 原画像 $I_0(n_1, n_2)$ と $J_0(n_1, n_2)$ から，$\boldsymbol{p}_0$ と $\boldsymbol{q}_0$ を中心とした探索ウィンドウをそれぞれ抜き出す．サブピクセルレベルの対応付けを用いて 2 つのウィンドウ間の平行移動量を求める．求めた平行移動量を $\boldsymbol{\delta} = (\delta_1, \delta_2)$ とおくと，対応点は以下のように定められる．

$$\boldsymbol{q} \quad = \quad \boldsymbol{q}_0 + \boldsymbol{\delta} \tag{4}$$

3. FPGA 実装

3.1 FPGA 向け OpenCL の設計手法

OpenCL のプログラミングモデルについて簡単に説明する．図 2 に OpenCL のメモリモデルを示す．グローバルメモリとコンスタントメモリは，全ワークアイテムからアクセス可能なメモリである．2 つの違いは，グローバルメモリが読み書き可能であるのに対し，コンスタントメモリは読み込みしか出来ない点である．ローカルメモリはワークグループで共有されるメモリであり，プライベートメモリは各ワークアイテムで占有されるメモリである．図 3 に OpenCL のスレッド空間を示す．スレッド空間は階層構造を持ち，処理全体はワークグループによって構成され，ワークグループはワークアイテムによって構成されている．ここで，ワークアイテムはスレッドに対応する．OpenCL におけるカーネルの設計は，ワークアイテムが行う処理を記述することで行う．

FPGA と GPU のアーキテクチャの大きな違いとして，FPGA は大きなサイズのローカルメモリを持つことが挙げられる．例えば，近年のハイエンド FPGA である Stratix V は，50M bits という大きなメモリブロックを持つ．また，FPGA 向け OpenCL では，GPU 向け OpenCL と比べ，より柔軟にローカル，プライベートメモリを扱うことが出来る．これにより，一度グローバルメモリから読み込んだデータを出来る限り書き戻す事無く使うことで，グローバルメ

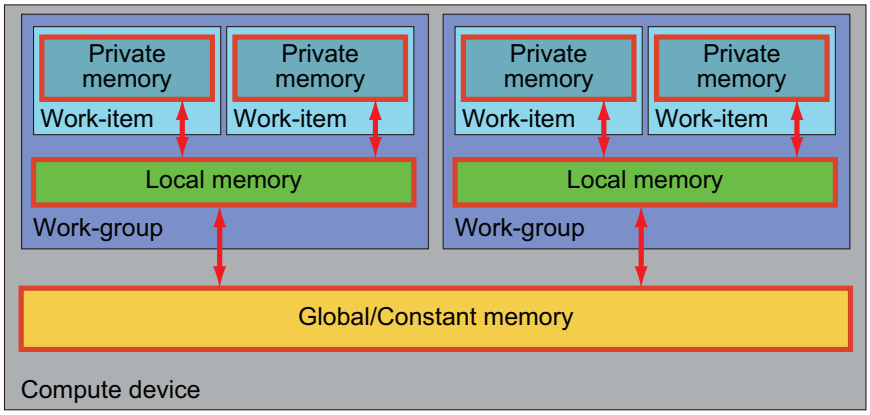

Fig. 2　OpenCL におけるメモリモデル

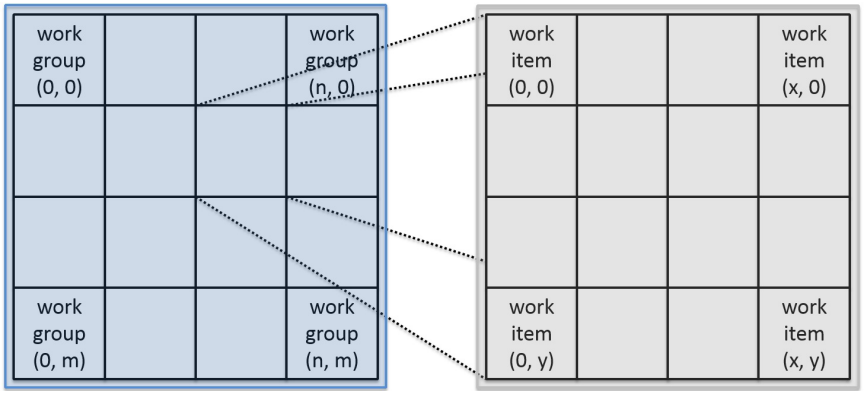

Fig. 3　OpenCL におけるスレッド空間

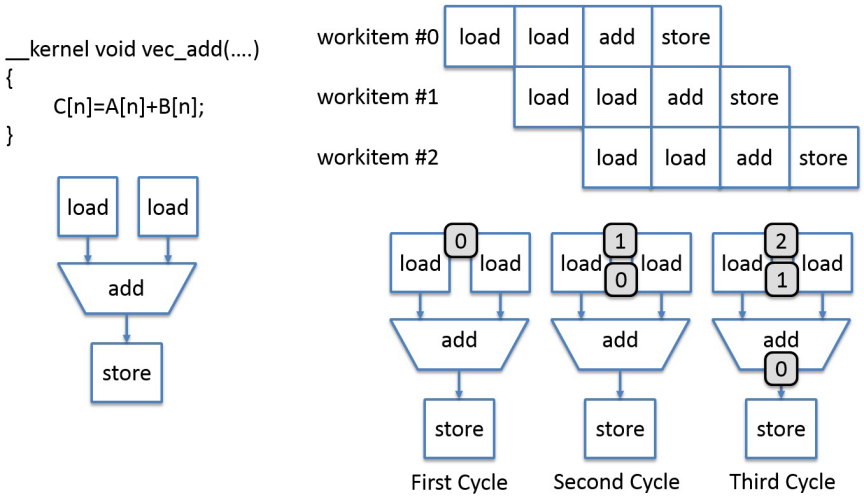

Fig. 4　FPGA 向け OpenCL で作成されるパイプライン

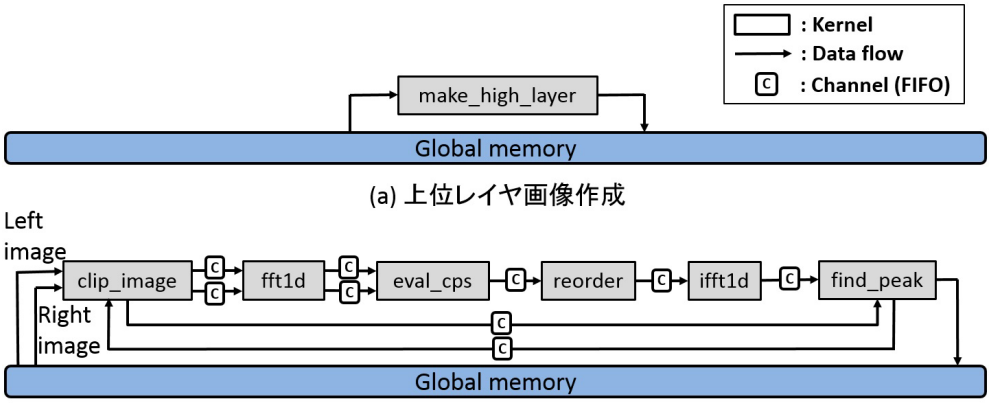

Fig. 5　実装したカーネルの構造

モリの帯域を効率良く使用することが出来る．メモリ構造の柔軟性に基づきデータを最大限に再利用するために，以下の工夫を行う:

- 窓関数，重み付け関数の係数と FFT で用いる回転因子は全てコンスタントメモリに持つ
- 除算のように計算コストが大きい演算は出来る限り定数の演算に変換し，係数はコンスタントメモリに持つ
- グローバルメモリからの読み込み処理では，アクセス毎にキャッシュが作られるため，キャッシュを考慮した設計を行う

また，FGPA 向け OpneCL と GPU 向け OpenCL の違いとして，FPGA ではパイプラインが作成されることが挙げられる．図 4 に示すように，カーネルの各演算は，各パイプラインステージで実行される．各スレッドはサイクル毎にパイプラインへ投入される．パイプラインで処理を行うことにより，性能（スループット）を保ちつつメモリ帯域を効率よく扱うことが出来る．これに対し，GPU 向け OpenCL では，ワークグループ内のスレッドをデータ並列で実行するため，大きなメモリ帯域が必要となる．また，パイプライン設計の利点を最大限に生かすため，FPGA 向け OpenCL では “Channel” という機能が提供されている．Channel を用いると異なるカーネル間において FIFO バッファを用いたデータのやり取りが実現できる．図 5 に POC を用いた画像対応付けを行う演算器の構成を示す．各カーネルの機能を以下に示す:

make high layer: 上位レイヤ画像の作成
clip image: 探索ウィンドウの抜き出し
fft1d: 1 次元フーリエ変換の計算
eval cps: 正規化クロスパワースペクトルの計算
reorder: 逆フーリエ変換用のデータ並び替え
ifft1d: 1 次元逆フーリエ変換の計算
find peak: 平行移動量の推定

Channel を用いるとデータをグローバルメモリに書き戻す事無く別のカーネルへ送ることが出来るため，データの再利用が容易に行える．例えば，本実装において，図 5 (b) に示す POC による対応付け処理では，find peak カーネルにおいて基準点数分の対応付け結果が得られるが，結果をグローバルメモリに書き出すこと無く Channel を通じて clip image カーネルに渡している．これにより次レイヤにおいて，対応結果に基づいて探索ウィンドウの抜き出しができる．

3.2　評価

POC による画像対応付け処理を CPU，GPU，および FPGA へ実装した結果を表 1 に示す．対

Table 1　CPU，GPU，FPGA 各実装における処理時間と消費電力

	Corei7-3960X (1 thread)	Corei7-3960X (12 threads)	Geforce GTX 580	Geforce GTX 680	FPGA board (Nallatech PCIE-395 D8) *2
Processing time[ms]	394.42	62.92	15.63	14.43	23.11
Power consumption*1 [W]	61.40	162.50	123.90	141.20	16.60
Power-delay product[W×s]	24.22	10.22	1.94	2.04	0.38

*1 消費電力は処理実行時と何も処理を行わない時の差分を表す
*2 Nallatech PCIE-395 D8 (Altera 社 Stratix-V FPGA 搭載)

Table 2　FPGA 実装におけるリソース使用量

	Logic	Registers	RAM blocks	DSP blocks
Stratix V D8	262,400	1,049,600	2,567	1,963
Implementation	189,756(72%)	394,022(36%)	1,375(54%)	346(18%)

応付けパラメタは，基準点数を 10,000 点，レイヤ数を 4，探索ウィンドウサイズを 32 ピクセル×15 行とした．CPU は Intel 社の Corei7-3960X (3.3GHz−3.9GHz) を用いた．GPU は NVIDIA 社の Geforce GTX580 と Geforce GTX680 を用いた．FPGA ボードは Altera 社の Stratix V と 4 バンクの DDR3 メモリが搭載された，Nallatech 社の PCIE-395-D8 を用いた．本設計における FPGA の最大動作周波数は 167MHz である．表 2 に本設計のリソース使用量を示す．上段が FPGA (Stratix V D8) のリソース量，下段がリソース使用量を表す．

評価尺度として，処理時間，消費電力，電力遅延時間積を用いる．消費電力は，PC 全体の消費電力を測定器 (HIOKI AC/DC POWER HiTESTER 3334) を用いて計測した．表 1 の消費電力は，処理中の消費電力と何も処理を行わない時の消費電力の差を表している．電力遅延時間積は，処理時間と消費電力の積であり，処理にかかる消費電力量を表している．GPU 実装によって 1 スレッド CPU 実装の 25∼27 倍程度，12 スレッド CPU 実装の 4.0∼4.3 倍程度の高速化が実現できた．また，FPGA 実装によって 1 スレッド CPU 実装の 17 倍程度，12 スレッド CPU 実装の 2.7 倍程度の高速化が実現できた．以上により処理時間において，FPGA のメモリ帯域は GPU よりはるかに小さいにもかかわらず，FPGA 実装は GPU 実装と同程度の性能が出ていることが確認できた．電力遅延時間積において，GPU 実装によって消費電力を 1 スレッド CPU 実装の約 8%，12 スレッド CPU 実装の約 19%まで削減できた．また FPGA 実装によって消費電力を 1 スレッド CPU 実装の約 1.5%，12 スレッド CPU 実装の約 3.7%，GPU (GTX 580) 実装の約 20%，GPU (GTX 680) 実装の約 19%まで削減できた．以上の結果から，FPGA 実装は，CPU 実装と比べて非常に低消費電力であり，GPU 実装と比べて同程度の処理速度かつ非常に低消費電力であることが確認できた．

4. 結論

本稿では，OpenCL を用いた POC による画像対応付け処理の FPGA 実装を提案した．自由度の高いメモリ構造によるデータの再利用とパイプラインの活用により，FPGA による高速かつ低消費電力な画像対応付けを実現した．今後の課題として，GPU 向け OpenCL のコードを FPGA 向けに最適化する手法を設計論としてまとめたいと考えている．

参考文献

1) R. Szeliski, [*Computer Vision: Algorithms and Applications*], Springer-Verlag New York Inc., 2010

2) T. Shibahara, T. Aoki, H. Nakajima, and K. Kobayashi, "A sub-pixel stereo correspondence technique based on 1D phase-only correlation," *Proc. Int 'l Conf. Image Processing*, V–221–V–224, 2007

3) M. Miura, K. Fudano, K. Ito, T. Aoki, H. Takizawa, and H. Kobayashi, "Performance evaluation of phase-based correspondence matching on GPUs," *Proc. SPIE Vol.8856, Applications of Digital Image Processing XXXVI*, 885614, pp.1–9, 2013

4) "The OpenCL Specification, Version 1.2." `http://www.khronos.org/registry/cl/specs/opencl-1.2.pdf`

5) "Altera SDK for OpenCL." `https://www.altera.com/products/design-software/embedded-software-developers/opencl/overview.html`

6) C. D. Kuglin, and D. C. Hines, "The phase correlation image alignment method," *Proc. Int'l Conf. Cybernetics and Society*, 163–165, 1975

7) K. Takita, T. Aoki, Y. Sasaki, T. Higuchi, and K. Kobayashi, "High-accuracy subpixel image registration based on phase-only correlation," *IEICE Trans. Fundamentals* **E86-A**, 1925–1934, 2003

8) D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *Int'l J. Computer Vision* **60**(2), 91–110, 2004

9) M. Miura, S. Sakai, J. Ishii, K. Ito, and T. Aoki, "An easy-to-use and accurate 3D shape measurement system using two snapshots," *Proc. Int'l Workshop on Advanced Image Technology*, 1103–1106, 2013