計測自動制御学会東北支部 40 周年記念学術講演会
2004 年 12 月 22 日，23 日　日本大学工学部

講演番号：C05

オプティカルフローの実時間検出システム

○福士将　堀口進　（東北大学）

A Real-time Optical-flow Detection System

∗M. Fukushi, and S. Horiguchi (Tohoku University)

Abstract– This paper describes a processor to detect optical flow in real-time. Optical flow is an apparent velocity field in 2D image sequences and it can be used to obtain information about object motion and scene structure. We propose an efficient processor for well known spatial optimization method which combines the two valuable characteristics of accuracy and computational efficiency. Our processor can reduce the computation time by utilizing pre-computed data in adjacent local region and can change computational parallelism according to the size of local region. The prototype system is designed and implemented using an FPGA. The time estimation shows that the prototype system can process 700×700 image sequences at 30Hz frame rate.

Key Words: Optical-flow, Spatial local optimization method, Dedicated processor, FPGA

1　まえがき

動画像から運動物体の速度情報を計測する研究は，工学や医学をはじめ多くの分野で行われている．例えば，気流等の可視化された流れ場の解析や，臓器や血管中の血流の解析等がある．また，計測された速度情報を利用した，監視システムや物体追跡の研究が盛んに行われており，近年では，ロボットビジョンやオートナビゲーションにおいて，知能情報処理の研究も行われている．

一般に，観測者と物体の間の相対的な運動によって生じる画面上の見かけの速度ベクトル場は，オプティカルフローとして知られている．オプティカルフローの計算法はこれまで数多く提案されており，それらは，Barron ら [1] の分類によると，勾配法，ブロックマッチング法，エネルギーに基づく方法，位相に基づく方法の 4 種類に大別されている．いずれの手法においても，一般に，オプティカルフローの検出にはリアルタイム処理では許容されないほど時間がかかるため，リアルタイム性が要求されるアプリケーションでは，専用ハードウェアによる処理の高速化が不可欠となる．

Johannesson ら [2] は，128 個の PE（Processing Element）が集積された画像処理チップを 4 個用いて，空間的局所最適化法 [3] によりオプティカルフローを検出するハードウェアを開発した．小室ら [4] は，ロボットビジョン向けに，センサと PE を一体化した超並列ビジョンチップを開発し，探索領域を制限したブロックマッチング法により 1000Hz のフレームレートでオプティカルフローを計算可能なことを報告している．近年，Correia ら [5] は，MaxVideo2000 というパイプライン画像処理プロセッサを用い，専用命令により空間的局所最適化法の高速化を行っている．

本報告では，勾配法の一種として知られている空間的局所最適化法に注目し，並列パイプライン処理により高速計算が可能なプロセッサを提案する．また，FPGA を用いてハードウェアとして試作実装した結果を報告する．空間的局所最適化法では，局所領域内での重み付き局所和の計算に多くの時間がかかる．そこで，隣接する局所領域間で途中の計算結果を再利用することにより処理時間の短縮を図る．提案するプロセッサは，必要な局所領域のサイズに応じて計算並列度を変更可能であり，さらに多数の PE による大規模並列処理への拡張も容易であるという特徴を持つ．

以下，2 節で対象とする空間的局所最適化法を説明する．3 節で本報告で提案するプロセッサの構成を説明し，4 節では，プロセッサを FPGA 上に実装し，回路規模や処理時間を評価した結果を述べる．最後に 5 節で本報告をまとめる．

2　オプティカルフロー

2.1　勾配法

勾配法は空間輝度勾配と時間輝度勾配，オプティカルフローの三者間の関係を表す拘束式を用いて，オプティカルフローを解析的に求める手法である．いま，時刻 t においてフレーム上の点 (x, y) の輝度値を $I(x, y, t)$ とし，この点を含むある物体が，微小時間の間に移動したとする．移動の前後で物体上の点の輝度が不変であると仮定すると，次式が成立する．

$$I_x u + I_y v + I_t = 0 \tag{1}$$

ここで，u, v は点 (x, y) におけるフローベクトル $\mathbf{v}$ の x 成分と y 成分であり，I_x, I_y は x 方向, y 方向の空間輝度勾配，I_t は時間輝度勾配である．式 (1) はオプティカルフローの拘束式と呼ばれており，勾配法に属する手法で共通なものである．

2.2　空間的局所最適化法

Lucas と Kanade[3] によって提案された空間的局所最適化法は，精度が良く，かつ計算量も比較的少ない手法として知られている [1]．この手法では，式 (1) の拘束式を解くために，点 (x, y) を中心とするサイズ $r \times r$ の局所領域 $R_{(x,y,r)}$ 内ではフローベクトル $\mathbf{v}$ は一定であるという仮定をおく．この仮定の下で，次式で表される 2 乗誤差を最小にすることにより，$\mathbf{v}$ を決定する．

$$E = \sum_{x,y \in R} w(x, y)^2 (I_x u + I_y v + I_t)^2 \tag{2}$$

ここで，$w(x, y)$ は $R_{(x,y,r)}$ の中心により大きな重みをかけるような重み関数であり，通常，2 次元ガウス関数

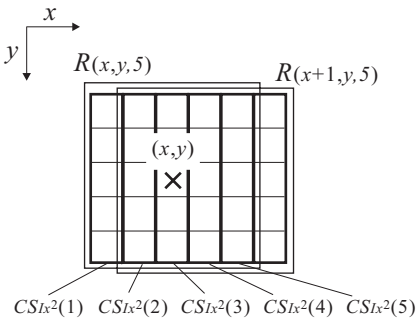


Fig. 1: Column-directional weighted sum in $R_{(x,y,5)}$ and $R_{(x+1,y,5)}$.

が用いられる．最小 2 乗法により，式 (2) を最小にする $\mathbf{v}$ は次式で与えられる [3]．

$$u = -\frac{\sum wI_y^2 \cdot \sum wI_xI_t - \sum wI_xI_y \cdot \sum wI_yI_t}{\sum wI_x^2 \cdot \sum wI_y^2 - (\sum wI_xI_y)^2} \quad (3)$$

$$v = -\frac{\sum wI_x^2 \cdot \sum wI_yI_t - \sum wI_xI_y \cdot \sum wI_xI_t}{\sum wI_x^2 \cdot \sum wI_y^2 - (\sum wI_xI_y)^2} \quad (4)$$

式 (3), (4) により点 (x, y) における $\mathbf{v}$ が計算され，この計算を 1 フレーム内の全画素に対して繰り返す．さらに，この一連の計算をフレームレート（フレーム/秒）で繰り返し行う必要があるため，他の手法よりも計算量が少ないとはいえ，ソフトウェアによるリアルタイム処理は困難である．

3 オプティカルフロー検出プロセッサ

3.1 計算の削減

重み付けの計算を行方向と列方向に分解可能なことに着目して，近隣の局所領域間で途中の計算結果の再利用を図る．$w(x, y)$ を 2 次元ガウス関数，$g(x)$ を 1 次元ガウス関数とすると，たとえば $R_{(x,y,r)}$ 内における $\sum wI_x^2$ の計算は次式のように変形できる．

$$\sum_{i,j\in R} w(i,j)I_x^2(i,j) = \sum_{i\in R} g(i) \sum_{j\in R} g(j)I_x^2(i,j) \quad (5)$$

ただし，$1 \le i, j \le r$ である．

ここで，式 (5) の右辺の $\sum_{j\in R} g(j)I_x^2$ を，i 列目の列方向重み付き局所和として $CS_{I_x^2}(i)$ で表す．Fig. 1 に示すように，$r = 5$ とした場合，$R_{(x,y,5)}$ で計算される $CS_{I_x^2}(2)$〜$CS_{I_x^2}(5)$ は，隣接する $R_{(x+1,y,5)}$ でも計算されることが分かる．すなわち，重み付けを行と列方向に分解する場合，計算点の x 座標のみを連続して変化させながら $\mathbf{v}$ を計算することにより，大部分の計算を削減することが可能である．これは I_x^2 だけでなく，他の I_y^2，I_xI_y，I_xI_t，I_yI_t も同様にあてはまる．以降の記述を簡略化するために，I_y^2，I_xI_y，I_xI_t，I_yI_t に対する i 列目の列方向重み付き局所和も同様に $CS_{I_y^2}(i)$, $CS_{I_xI_y}(i)$, $CS_{I_xI_t}(i)$, $CS_{I_yI_t}(i)$ で表す．

3.2 プロセッサの構成

本報告では，入力画像は 256 階調のグレースケールとし，各輝度値を 0〜255 の値をとる 8 ビット固定小数点形式で表す．また，計算精度を保つために，必要に応じてビット数を拡張するものとする．

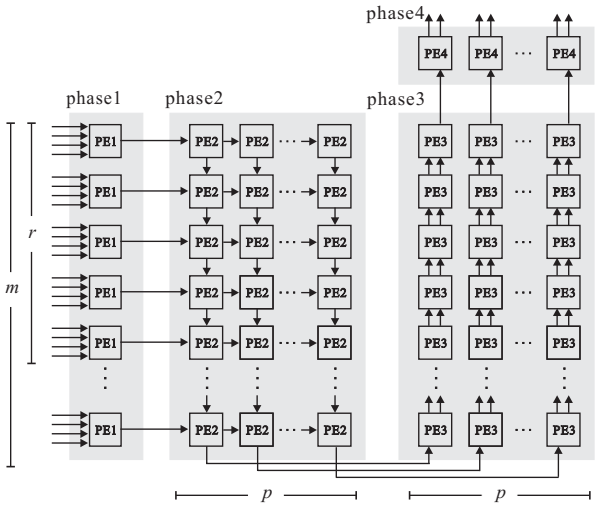
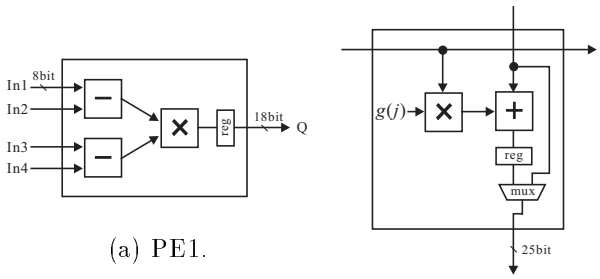
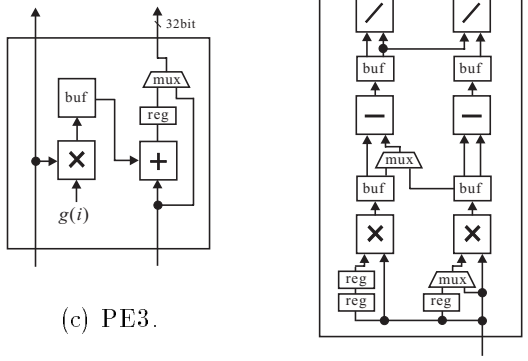


Fig. 2: Optical-flow detection processor.



(a) PE1.

(b) PE2.



(c) PE3.

(d) PE4.

Fig. 3: Structure of PE1, PE2, PE3, and PE4 (reg: register, mux: multiplexor, buf: buffer memory).

Fig. 2 に提案するプロセッサの構成を示す．計算処理のフェーズ毎にフェーズ 1〜4 の 4 つのブロックからなり，各ブロックは，Fig. 3(a)〜(d) に示されるように，それぞれ構成の異なる複数の PE からなる．各ブロックは直列に接続され，入力される輝度値のデータに対してパイプライン処理を行う．

このプロセッサは回路規模に関するパラメータとして m と p を持つ．m はフェーズ 1〜3 のブロックに存在する PE の行数であり，処理可能な局所領域の最大サイズを表す．p はフェーズ 2〜4 のブロックに存在する PE の列数であり，並列に処理可能な最大行数を表す．

各フェーズで行う処理を以下に示す．

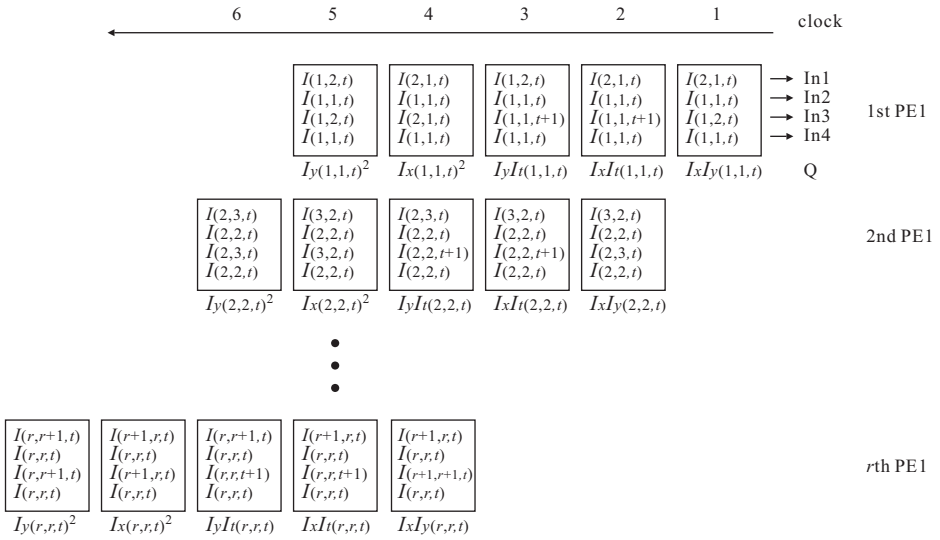


Fig. 4: Timing of data imput.

フェーズ 1: 空間輝度勾配 I_x，I_y，時間輝度勾配 I_t を計算し，それらの積 $I_x^2, I_y^2, I_xI_y, I_xI_t, I_yI_t$ を計算する．

計算に必要なデータは図 4 に示すタイミングで各 PE1 に入力され，PE1 では減算と乗算を含めて 1 クロックで計算を行い，結果がレジスタに格納される．この処理は各 PE1 で独立して行う．各 PE1 では 1 クロック毎に I_xI_y，I_xI_t，I_yI_t, I_x^2，I_y^2 の順で計算が終了し，それらは順次フェーズ 2 のブロックへと入力される．

フェーズ 2: 5 種類の列方向重み付き局所和 $CS_{I_xI_y}(i)$, $CS_{I_xI_t}(i), CS_{I_yI_t}(i)$ $CS_{I_x^2}(i), CS_{I_y^2}(i)$ を計算する．

r 行の PE2 により，局所領域 r 行分の重み乗算と累算をパイプライン式に計算する．r クロック後に $CS_{I_xI_y}(i)$ がフェーズ 3 のブロックへと出力され，以降，フェーズ 1 での出力順に，$CS_{I_xI_t}(i), CS_{I_yI_t}(i)$ $CS_{I_x^2}(i), CS_{I_y^2}(i)$ が 1 クロック毎に出力される．

重みの値は，局所領域のサイズが通常よく用いられる $r = 5$ の場合，$g(j) = [0.05, 0.25, 0.4, 0.25, 0.05]^T$ で近似できる [6]．これらの値に適当な係数をかけても式 (3), (4) の計算結果は変わらないため，$g(j) = [1, 5, 8, 5, 1]^T$ の整数値を用いる．

フェーズ 3: 5 種類の重み付き局所和 $\sum wI_xI_y$, $\sum wI_xI_t, \sum wI_yI_t, \sum wI_x^2, \sum wI_y^2$ を計算する．

フェーズ 2 で計算された列方向重み付き局所和の行方向重み付き局所和を計算する．この計算は，フェーズ 2 と同様に，r 行の PE3 により重み乗算と累算をパイプライン式に計算することにより行う．

フェーズ 2 のブロックから入力される列方向重み付き局所和は，近隣の局所領域の計算で再利用するために，各 PE3 のバッファメモリに格納される．局所領域のサイズを $r = 5$ とした場合の，m 行目から $m - 4$ 行目の PE3 内のバッファに格納されるデータを Fig. 5 に示す．

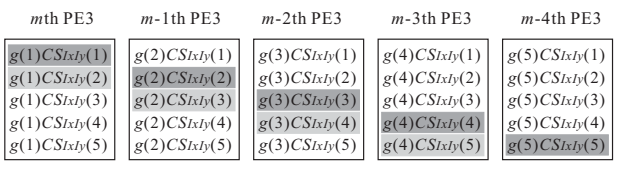


Fig. 5: Data in buffer memory.

データ数が多いために Fig. 5 には $CS_{I_xI_t}(i)$ のみを示しているが，実際にはこの他に $CS_{I_xI_t}(i)$, $CS_{I_yI_t}(i)$, $CS_{I_x^2}(i), CS_{I_y^2}(i)$ も続いて格納されている．

$m - 4$ 行目の PE3 のバッファに $g(5)CS_{I_xI_y}(5)$ が格納された時点で，$\sum wI_xI_y$ の計算に必要なデータが全て揃う．バッファメモリから加算器に入力するデータを適切に選択し，加算のタイミングを調整することで，$g(5)CS_{I_xI_y}(5)$ が格納された次のクロックで $\sum wI_xI_y$ の累算を終了して，フェーズ 4 のブロックへと出力させることができる．以降，1 クロック毎に $\sum wI_xI_t, \sum wI_yI_t$, $\sum wI_x^2, \sum wI_y^2$ の順でフェーズ 4 に出力される．

ここで，隣接する点 $(x + 1, y)$ での計算に必要な $g(1)CS_{I_xI_y}(2)$ 等のデータはすでに各 PE3 のバッファに格納されているため，$m-4$ 行目の PE に $g(5)CS_{I_xI_y}(6)$ が格納された時点で，点 $(x + 1, y)$ での $\sum I_xI_y$ が計算可能である．すなわち，点 (x, y) での 5 種類の重み付き局所和の出力に引き続き，点 $(x+1, y)$ での出力が開始可能である．

フェーズ 4: (x, y) 上でのフローベクトル $\mathbf{v}$ を計算する．

計算に必要なデータが $\sum wI_xI_y, \sum wI_xI_t, \sum wI_yI_t$, $\sum wI_x^2, \sum wI_y^2$ の順で入力されるため，式 (3)，(4) に従ってフローベクトルを計算する．PE4 の 2 個の除算器は配列型除算器であり，データの入力から信号遅延分の時間で除算結果が求まる．

計算で必要とされる局所領域のサイズ r はアプリケーションにより異なる可能性があるが，本プロセッサでは最大 m まで対応可能である．局所領域の 1 列を単位としてデータが入力されるため，点 (x, y) での計算には，フェーズ 1 の r 行の PE，フェーズ 2 と 3 の r 行 1 列の PE，そしてフェーズ 4 の 1 列の PE が使用される．決められた r に対して，プロセッサ内の PE の行数が多い場合 $(m > r)$，フェーズ 1 の 1 行の PE と，フェーズ 2〜4 の 1 列の PE を利用して，隣接する点 $(x, y+1)$ での計算を行うことが可能である．すなわち，Fig. 2 に示すプロセッサでは，$\min(p, m - r + 1)$ 行でフローベクトルを並列に計算可能である．

Table 1: Hardware cost of each phase block.

	slice	utilization ratio (%)	multiplier (18×18)	frequency (MHz)
phase1	122	0.4	6	–
phase2	160	0.7	12	213.4
phase3	6802	28	20	124.2
phase4	9658	40	16	42.8
total	16412	71	54	42.8

4 FPGA を用いた試作システムと評価

4.1 実装環境

プロセッサのパラメータを $m = 6, p = 2, r = 5$ として，VHDL により Fig. 2 に示したプロセッサの回路設計を行い，FPGA 上に実装した．本報告での実装環境を以下に示す．

- FPGA ボード: BenONE (NALLATECH 社)
- FPGA チップ: XC2V4000 (Xilinx 社)
- 設計ツール: ISE 6.2 (Xilinx 社)
- 波形シミュレータ: ModelSim XE 5.6 (Model Technology 社)

PC と FPGA ボード間の通信機能は実装されていないため，プロセッサの動作確認は波形シミュレータから得られる出力波形により行った．

4.2 回路規模の評価

使用した回路設計ツールのレポートファイルにより，FPGA 内の使用リソース量を評価する．Table 1 は，FPGA 内のスライス数と全スライス数に対する割合，内蔵 18 ビット乗算器数，および，最大動作周波数を各フェーズ毎に示したものである．フェーズ 3 と 4 で多くのスライスを消費しているが，これは，それぞれバッファメモリと配列型除算器を持つことが原因である．プロセッサ全体の最大動作周波数は，約 42.8MHz であるという結果が得られた．

4.3 処理時間の評価

フレームサイズを $X \times Y$ とした場合の処理に必要なクロック数を導出し，プロセッサの処理時間を評価する．画像フレーム上の各行頭の点では，データの再利用ができないために，フローベクトルが出力されるまで $6r + 9$ クロックかかる．以降は 5 クロック毎に各点でのフローベクトルが出力される．これらの処理は $\min(m - r + 1, p)$ 行で並列に行われるため，計算に必要クロック数は次式で表すことができる．

$$T = \frac{(6r + 9 + 5X)Y}{min(m - r + 1, p)} \qquad (6)$$

クロック周波数を F とすると，1 フレームあたりの処理時間は T/F となる．今回設計した $m = 6, p = 2, r = 5$ のパラメータを用い，F=40MHz とした場合の処理時間を Fig. 6 に示す．NTSC のように 30Hz の処理を想定する場合，1 フレームあたりの処理時間が 33ms 以下であればよいため，実装したシステムではサイズ約 700×700 以下のフレームを処理可能である．

ここで，同じ空間的局所最適化法を採用している従来法 [2]，[5] とスループットを比較する．従来法とスループットを比較する．Table 2 から，提案するプロセッサは従来法よりも高いスループットを実現可能なことが分かる．

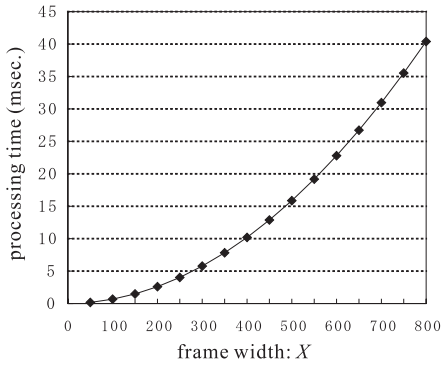


Fig. 6: Processing time as a function of frame width ($X = Y$).

Table 2: Comparison of throughput.

	throughput (M pixel/second)
Johannesson [2]	13.11
Correia [5]	1.67
proposed processor	15.81

5 まとめ

本報告では，オプティカルフローの高速検出プロセッサを実現することを目的に，空間的局所最適化法の高速並列パイプライン処理法を提案し，ハードウェアとして FPGA 上に実装した．このプロセッサは，重み付き局所和の繰り返し計算において，計算結果を再利用するアプローチにより高速並列パイプライン処理を実現する．局所領域のサイズの変更にも対応でき，さらに，計算並列度を簡単に上げることができるという特徴を持つ．処理時間の評価により，サイズ 700×700 のフレームを 30Hz で処理可能であり，さらに従来のプロセッサよりもスループットが高いことが示された．

今後の課題は，試作システムにおいて，PC と FPGA 間の通信プログラムを作成し，実画像データを処理させることである．

参考文献

[1] J. L. Barron, D. J. Fleet and S. S. Beauchemin, System and Experiment Performance of Optical Flow Techniques, Int'l Journal of Cmputer Vision, vol.12, issue1, pp. 43–77, 1994.

[2] M. Johannesson and M. Gokstorp, "Video-tate Pyramid Optical Flow Computaion on the Linear SIMD Array IVIP", Proc. of the Computer Architectures for Machine Perception pp.280–287, 1995

[3] B. Lucas and T. Kanade, "An Intensice Image Registration Technique with An Application to Stereo Vision", Proc. of DARPA Image Understanding Workshop, pp.121–130, 1981.

[4] 小室 孝, 鈴木 伸介, 石井 抱, 石川 正俊, "汎用プロセッシングエレメントを用いた超並列・超高速ビジョンチップの設計", 信学論 (D-I), vol.J81-D-I, no.2, pp.70–76, 1998.

[5] M. V. Correia and A. C. Campilho, "Real-Time Implementation of an Optical Flow Algorithm", Proc. of the 16th Int'l Conf. on Pattern Recognition, vol.4, pp. 40243–40246, 2002

[6] P. J. Burt. "Fast Filiter Transform for Image Processing". Computer Vision and Image Processing, vol.16, pp. 20–51, 1981.