6T-1

1次割当問題の近似計算に基づくブロックフロー計算ハードウェア

内苑孝俊 † 富岡洋一 † 北澤仁志 †
Takatoshi Uchizono Yoichi Tomioka Hitoshi Kitazawa

† 東京農工大学
Tokyo University of Agriculture and Technology

1 はじめに

動画像解析において有用である，移動物体のトラッキングのための最適画像間対応を求める手法の一つとして画像ブロック間の1対1対応を求める排他的ブロックマッチング [1] が提案されている．しかし，ブロック間対応を求めるためには多大な計算時間を要する．

本研究ではブロック間の対応を高速に計算することを目的として，排他的ブロックマッチングにおける処理の一部である1次割当問題の各種近似計算手法について調べ，絶対値を考慮した Saving Regret 近似が品質，ハードウェア量の点で適切であることを示した．また，FPGA の回路規模，処理速度，および高並列実行の可能性について検証を行った．

2 1次割当による最適ブロック間対応

動画像におけるブロック間対応を，排他的ブロックマッチングを用いて求める．これは，現フレームと過去フレームの各ブロック間類似度を N × N の行列で表し，1次割当問題として最適対応を求める手法である．さらに，移動物体の隠蔽と再出現に対応するため，複数の過去フレームを並べる．また，背景の再出現と移動物体の生成に対応するため，Bg と Create を追加する．その結果図1のコスト行列が得られ，これよりブロック間の対応 (ブロックフロー) が得られる．

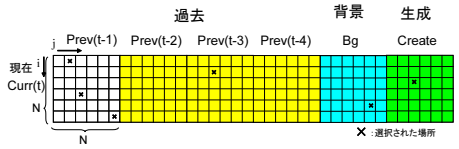

図1 最適ブロック間対応を求めるコスト行列

3 1次割当問題

1次割当問題の計算量は通常 $O(n^3)$ であり，最速 $O(n^2)$ で厳密解が得られることが報告されているが，並列化に適した手法ではない．ここでは並列化に適した1次割当問題の近似解法を検討する．

3.1 1次割当問題の近似解法

1次割当問題の近似解法として最小値の小さいものから順に選択する Greedy のほかに Saving Regret[2] という手法がある．Saving Regret ではコスト行列の行の最小値と二番目に小さい値との差 (Diff) が最大の行の最小値から順に選択することを繰り返すことで近似的に合計を最小化する手法である．最小値の計算は各行ごとに独立して行うことが可能であり，並列化に適した手法である．1次割当問題では各行，各列に定数を加算しても結果は影響を受けないため，照明変動の影響に頑健となる．Saving Regret では各行の最小値と次の値との差を比較しているため，列方向の頑健性は失われてしまうが，行方向の頑健性は保たれている．これに対し，Greedy では絶対値を用いて選択を行うため，行方向，列方向共に頑健性が失なわれてしまう．このことから，局所的な照明変動などに対し Saving Regret は Greedy より優れていると考えられる．

3.2 ソフトウェアシミュレーション

各近似手法の品質を調べるためソフトウェアによるシミュレーションを行った．評価基準であるブロックフローの精度の指標として，抽出された全フローにおける交差するフローの割合 (cross/cnt)，色類似度 (cost(color))，形状コスト (cost(shape)) の3つを用いた．cost(color)，cost(shape) はそれぞれ以下の式 (1) の第一項，第二項に対応する．

$$E = \sum_p c(p, f_p^k) + \lambda \sum_p \sum_{q \in N_p} d(p, q, f_p^k, f_q^k) \qquad (1)$$

p, q : 現在フレームのブロック

f_p^k, f_q^k : p, q に対応する過去フレームのブロック

c : ブロックの HSV 色類似度

N_p : p の8隣接近傍で，$Prev$ での相対位置が p と同じ $object$ に属する $pixel$

d : マンハッタン距離

シミュレーション結果を表 1 に示す．結果から cross/cnt，cost(color) については Saving Regret が優れ，cost(shape) に関しては Greedy が優れていることがわかる．また，実際のフローを確認すると Greedy の方が優れる場面も多く見られた．

表 1 1次割当問題の近似解法の比較

	cross/cnt	cost(color)	cost(shape)
厳密解	0.02102	50345.32	0.568915
Greedy	0.06921	52306.36	0.468524
Saving Regret	0.06009	51882.33	0.735880
Hybrid Saving Regret	0.05843	51827.64	0.455128
Saving Regret(分割)	0.07360	52718.72	0.728891

3.3 Hybrid Saving Regret

シミュレーション結果から最小値の絶対値も有効な情報となると考えられる．そこで，以下の式 (2) のように最小値の絶対値を用いて Saving Regret 手法での Diff の値を修正した．この手法を Hybrid Saving Regret とする．α は実験結果から 2 とし，β は Diff が 0 以下とならない値に設定した．表1のシミュレーション結果から Saving Regret において最小値の絶対値を考慮することで結果が改善されたことが確認できた．以上の結果から

Hybrid Saving Regret を FPGA に実装した．

$$Diff = (\text{二番目に小さい値}) - \alpha \times (\text{最小値}) + \beta \quad (2)$$

4 ハードウェアの構成

ハードウェアの構成を図 2 に示す．ハードウェアは大きくブロック類似度計算回路と 1 次割当計算回路に分けられる．ブロック類似度計算回路ではカメラから入力された画像をブロックごとに分けて特徴量を抽出し，現在フレームと過去フレームのブロック間類似度を求める．1 次割当計算回路では，得られたブロック間類似度から最適ブロック間対応を求める．現時点では 1 次割当計算回路 (図 2 の Assignment の部分) のみの作成となっているため，検証の際には類似度の計算をソフトウェアで行い，外部メモリ (SSRAM) 経由で 1 次割当計算回路へのデータ入力を行っている．

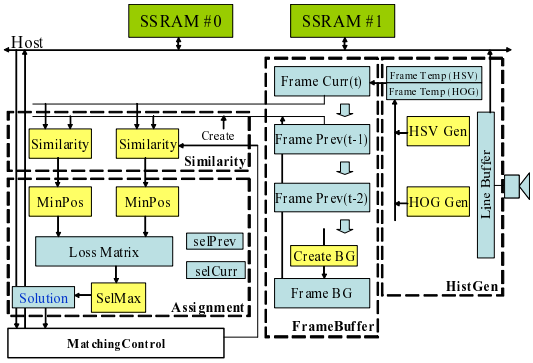

図 2　ブロックフロー抽出ハードウェアの構成

1 次割当問題計算回路の処理の流れとしては，MinPos(最小値位置検出回路) で各行の Diff と最小値の列番号を並列に求め，SelMax(最大値検出回路) で最大の Diff を選択し，Diff が属する行の最小値の列番号を Solution へ保存していく．

- MinPos(最小値位置検出回路)

16bit の cost 値と列番号 (col) を入力とし．各行の最小値 (min1)，二番目に小さい値 (min2)，三番目に小さい値 (min3) と比較し，min1，min2，min3 を順次更新していく．行内の値を全て比較し終えたら最終的な min1，min2，min3 から Diff1，Diff2，col（最小値の位置）を求め，出力する．回路規模を小さくするため比較器には Bit-Serial 演算器を用いている．

- SelMax(最大値検出回路)

Winner-Take-All 回路を用いて各行の Diff から最大となるものを選択．すでに選択済みの場合は MinPos に戻り，Diff を再計算する．

5 回路規模と実行速度

今回使用した FPGA は Altera 社の”Stratix3 EP3SL150F780C4”である．処理画像のサイズは QVGA(320 × 240) とし，ブロックサイズを 8 × 8，MinPos 回路の並列数は 8 とした．回路規模を表 2 に示す．外部メモリ経由のテストではクロック周波数 48 MHz で動作し，実行速度は Diff 計算の繰り返し回数を 30 とした場合 (実験により検証) およそ 2.05[fps] であった．尚，PC と FPGA ボード間のデータ転送時間は含まれていない．

表 2　回路規模

	ALUT 数	レジスタ	内部メモリ
実験回路 1(MinPos8 並列)	9,275	5,671	112,128
高並列実験回路 (MinPos150 並列)	32,942	26,966	2,581,888

5.1 高並列実験回路

外部メモリ経由でのデータ入力ではメモリネックにより並列度は大きくできない．そこで，並列アクセス可能な内部メモリを経由し，データを入力することで MinPos の並列数を増やした実験回路の実装を行った．処理画像のサイズは QVGA(320 × 240) とし，ブロックサイズを 16 × 16，MinPos 回路の並列数は 150 とした．回路規模を表 2 に示す．FPGA ボードの内部メモリの容量により実装可能なブロックサイズは 16 × 16 が限界であった．実験回路から得られた実行速度を基にブロックサイズが 8 × 8 で実装可能となった場合の実行速度は動作周波数 48[MHz] でおよそ 37.3[fps] であると予測された．

5.2 領域分割

外部メモリに比べ，内部メモリの容量は小さく，処理画像のサイズの拡大に伴い容量が足りなくなってしまう．そこで画像領域を分割し，各分割領域での割当結果を合わせることでメモリ消費，回路規模を抑えたまま大規模データ処理への拡張を考える．画像を 1/4 のサイズで分割し，境界での影響を軽減するため以下の図 3 右のような 9 個の領域の結果から解を求める．領域が重なる部分の解は単純に図 3 左のように予め各領域から解を採用する場所を指定して決定している．ソフトウェアでのシミュレーション結果を表 1 の Saving Regret(分割) に示す．シミュレーション結果から，オリジナルの Saving Regret に比べて各評価値は悪くなるが，極端に大きい差は見られない．このことから，重複した領域での解のとり方を工夫することでオリジナルの Saving Regret に近い解を得ることができると考えられる．

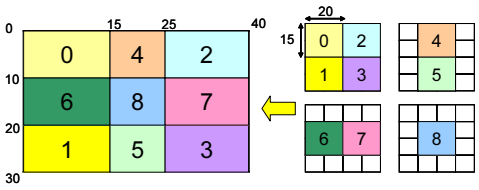

図 3　領域分割と解の採用範囲

6 まとめと今後の課題

1 次割当問題の近似解法の品質の評価を行い，絶対値を考慮した Saving Regret 近似解法をハードウェアに実装し，回路規模，実行速度を検証した．

今後の課題として，領域分割回路の品質向上を行い，大規模データへの拡張を進める．

本研究の一部は科研費基盤 (C)22500149 による．

参考文献

[1] Zhu Li, et al., ”Exclusive ...”, IEICE Trans.D, 2010.5.

[2] Michael A Trick. ”A LINEAR RELAXATION HEURISTIC...” April, 1989; Revised March, 1991.