システムソフトウェアと
オペレーティング・システム 69−16
（１９９５．６．２３）

実時間処理のための並列画像クラスタリング

三野吉輝　若谷彰良　廉田浩

松下電器産業（株）　研究本部　中央研究所

本論文では、動画像を圧縮符号化する際に前処理として有望視されている画像クラスタリング処理を実時間で行なうための並列処理方式について述べる。並列クラスタリング処理では、画面内で複数の画素をサンプルし、それらの性質（色や輝度等）の近いものを集めて並列的にクラスタの生成とそのパラメータ更新を繰り返し、画面を最適に分割するクラスタの組み合わせを求める。更にこの処理アルゴリズムを専用の並列アーキテクチャ上に実装することで、処理速度が従来の数百倍となり、動画像の実時間クラスタリング処理も可能となる。

Parallel Clustering Method for Realtime Image Processing

Yoshiteru Mino　Akiyoshi Wakatani　Hiroshi Kadota

Central Research Laboratories

Corporate Research Division

Matsushita Electric Industrial Co.,Ltd.

Moriguchi-shi,570 Japan

This paper describes a high-speed pararell clustering method, which is suitable for a preprocessing of the motion picture compression and coding.In the parallel clustering process, prural pixels are sampled in one frame of image. They are gathered according to their characteristics; color, brightness etc.,to organaize clusters or to modify cluster-parameters, in pararell manner.This operation is repeated until the frame is optimally segmented by a set of clusters. Furthermore,by implementing this algorithm on a dedicated parallel architecture,the process speed increases by several hundred times,and a realtime clustering of motion picture will be available.

1 はじめに

画像圧縮において、フレーム画像内で性質の似通ったものを一つのレイヤとし、複数のレイヤから構成されたフレーム画像をレイヤ毎に異なる圧縮方式を適応選択することで圧縮率を向上させたり、必要領域の画質劣化を防止している。その際、フレーム画像を各レイヤに領域分割する方式の一つとしてクラスタリングがある [1][2]。自然画像内のクラスタは色、空間の広がり部分で動きベクトル等の性質が近い画素の小集合であって、レイヤは複数のクラスタから構成される。例えば、左から右に動いている車（レイヤ）は、座席に座っている人物、タイヤ、車体などのクラスタから構成されている。

領域抽出を行なう他方式としては緩和法によるラインプロセス [3] を用いたものがある。ラインプロセスは、全画素単位での動き推定を行なうため多くの反復演算を要するため処理時間が長く、画像依存の不連続判定を行なう必要がある。これに対して、クラスタリングは、ランダムに選択した全画素の数分の1程度を用いるため総演算量が少ないこと、複雑な輪郭への対応が可能であることが特徴である。尚、クラスタからレイヤを構成するには、別のアルゴリズムが必要である。

本論文では、従来の逐次クラスタリングのアルゴリズムの概要とその並列化手法及び効果について述べ、実時間処理の可能性について一般的な DSP 命令セットを用いて評価した結果を示す。

2 クラスタリングアルゴリズム

クラスタリングは大きく 2 つの処理から構成される。前半は、任意に選ばれた画素のパラメータからクラスタパラメータを求めクラスタ構成を決定する部分で、後半は、それぞれの画素が、決定されたクラスタのどれに属するかを決める部分である。

前半は、任意に選ばれた画素に対して、次の 3 ステップの処理を繰り返し、クラスタパラメータの更新率がある一定以下になるまで繰り返す。経験的には、全画素数の数分の1位のサンプリングで十分である。

1. 選ばれた画素と各クラスタとの尤度 (距離) 計算
2. 尤度が最も大きい (距離が最小) クラスタ選択
3. 選択されたクラスタのパラメータ更新

後半は、前半で決定されたクラスタに対して、各画素がどれに属するかを決定するもので、前半の第 1 ステップ、第 2 ステップとほぼ同じで、選ばれたクラスタがその画素の属すべきクラスタとなる。

本論文では、前半に着目し、それに関するアルゴリズムの並列化を行なう。後半は、前半に準じ、並列化可能である。図 1にクラスタリングの流れを示す。

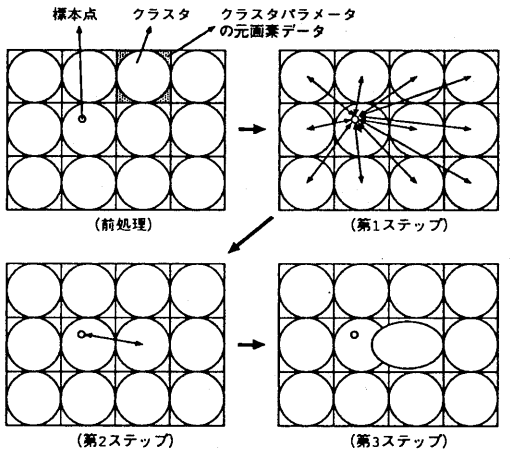


図 1: クラスタリングの流れ

3 アルゴリズムの並列化

クラスタリングを構成している 3 ステップにおいて、第 3 のステップは、第 2 の比較結果により、選択されたクラスタパラメータのみを更新するという逐次処理になっている。しかし、第 1 ステップ及び第 2 ステップを複数回行なった後に、クラスタパラメータの更新を複数回同時に行ない、第 3 ステップを複数のクラスタで同時に実行することにすれば、第 3 ステップは並列化できる。

しかし、クラスタリングの並列化によって、パラメータの更新が遅れ、そのため、逆にクラスタパラメータの収束までの標本数が増加すると推測される。この収束劣化を実験により確かめた。

図2は、第3ステップにおけるクラスタの同時実行度（並列度）における収束度を示したもので縦軸に孤立標本点の標本点に占める割合（%）を、横軸に標本点数をプロットしている。

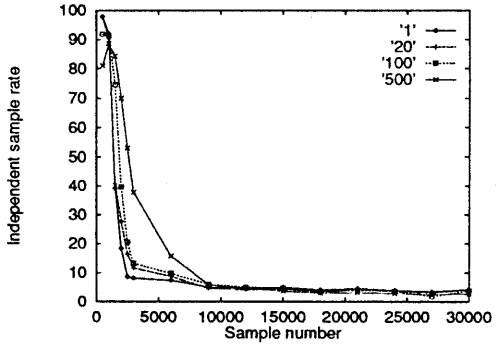


図2: 並列度1,20,100,500における孤立標本点数

標本点の距離がある一定以上である場合は、孤立標本点として扱い、孤立標本点の発生確率はクラスタの収束を表すと考える。孤立標本点数はクラスタの各パラメータの変化率と同様の傾向を示す。

このように、同時更新の数を増やすことによって収束が遅くなる傾向があるものの、それは、同時更新の数にそのまま比例するものではない。孤立標本点が4% 以下になった時点でクラスタリングが収束すると考えると、いずれの並列度でも約9000標本数が必要である。したがって、計算時間から見た場合、同時更新は、サンプリングの増加を招くが並列実行により処理全体では効果的であると考えられる。

3.1 並列化効果の評価

	距離計算	パラメータ更新
(0)	逐次	逐次
(1)	クラスタ並列	クラスタ並列
(2)	画素並列	クラスタ並列

上表に距離計算とクラスタパラメータ更新を処理方法により分類した。

(0) は、(尤度) 距離計算、クラスタパラメータ更新を逐次に行なうものである。(1),(2) は、並列クラスタリングを使ったアルゴリズムで、(1) は (少なくともクラスタ数よりは多い) 複数の距離計算の後に、まとめてパラメータ更新を行なう。(2) は (1) とは異なり、距離計算を画素毎に並列に行なう。

ここでは並列アルゴリズムの実行時間を評価する。ここで用いる記号を下表に示す。

Nc	クラスタ数	Cb	ブロードキャストコスト
Ni	画面分割数	Cm	最小値検出コスト
$\alpha=\frac{Ni}{Nc}$	状態数	Co	one-to-one通信コスト
Ed	距離計算時間	Eu	クラスタ更新時間

N_cは画面を構成するクラスタ数で本評価においては一定値とする。N_iは画面分割数で (2) の方式で用い、その上限は画素数である。C_bは特定のサイズのデータをブロードキャスト時間で、送り先のプロセッサ数に依存する。C_mは複数のプロセッサに格納されたデータの最小値検出時間で、送り元のプロセッサ数に依存する。C_oは任意のプロセッサ間の通信時間である。E_dはあるクラスタと画素との距離計算時間を示し、E_uはクラスタパラメータの更新時間を示す。

なお、距離比較は他の時間に比べて無視できるほど小さいと考える。しかし、並列アルゴリズムにおいては比較のための通信時間は考慮する。

逐次アルゴリズムは、クラスタ毎に距離計算をした後に、パラメータ更新を行ない、1 サンプリング当たりの実行時間は次のようになる。

$$(0) = N_c \times E_d + E_u$$

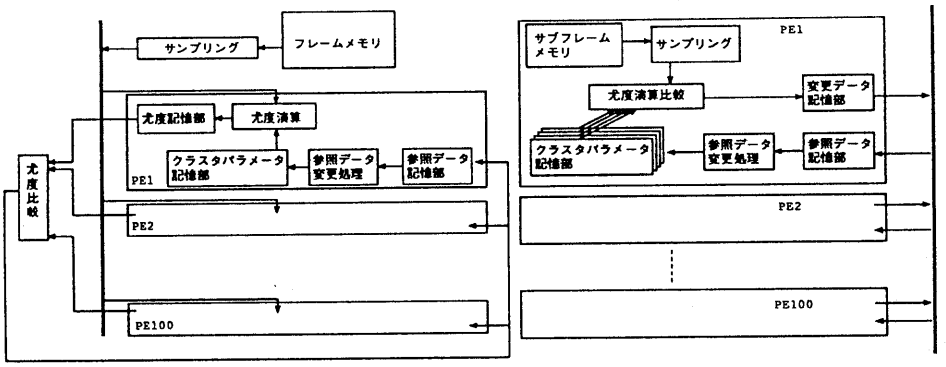
3.1.1 クラスタ並列

距離計算とパラメータ更新を並列化する場合は、尤度を計算し、その比較結果をパラメータ更新せずに蓄えておく必要がある。この実現例を図3-(a) に示す。

図において複数のサンプリングが終了するまで、比較結果は参照データ記憶部に格納されており、あとで同時に更新を行なう。この場合の実行時間を示す。

$$(1) = \frac{E_d \times N_c + E_u + C_m(N_c) \times N_c + C_b(N_c) \times N_c}{N_c}$$

(1) は最低でもクラスタ数 (N_c) 分のサンプリングをまとめて行なうので、この時間を求め、それをクラスタ数 (N_c) で割った時間を求める。画素情報を全プロ



(a) タイプ1　　(b) タイプ2

図3: 並列画像クラスタリング装置の構成

セッサにブロードキャストする時間、最小値検出及び距離計算は、(0) の N_c倍の時間が必要であるが、パラメータ更新は、各プロセッサで1回のみ行なうので、(0) と同じである。以上を足し合わせたものを$\frac{1}{N_c}$したものが1サンプリング当たりの実行時間になるが、この値は (0) よりも小さい。

3.1.2 画素並列

画素を並列の単位とする場合、それぞれのプロセッサは全クラスタパラメータを持ち、分割された画面に対応するフレームデータの内容を持っている。つまり、サンプリングはプロセッサ内で行なうので、画素情報をサンプリング毎にブロードキャストする必要はない。しかし、クラスタパラメータの更新は複雑になる。図3-(b) に画素並列の実現例を示す。

図において距離比較した結果はパラメータ更新するまで変更データ記憶部に格納され、パラメータ更新の際に、変更データを対応したプロセッサに送る。この実行時間を下記に示す。尚、全プロセッサで1サンプリングするとすると、全体ではN_iサンプリングすることになるので、個々に1サンプリングする場合の時間をN_iで割ることになる。

$$(2) \quad = \frac{E_d \times N_c + N_i \times C_o + E_u \times \frac{N_i}{N_c} + N_c \times C_b(N_i)}{N_i}$$

(2) の場合は、距離計算に関しては プロセッサ内では逐次になるので、E_dの N_c倍である。変更データは、特定のプロセッサに1対1通信される。また、パラメータ更新は分割画素数/クラスタ数回、それぞれのプロセッサで行なわれ、更新されたクラスタパラメータは全プロセッサにブロードキャストされる。

3.2 アーキテクチャの並列化

ここまで提案した並列クラスタリングは、(2) 方式が最も並列度も高くできるので、実行時間も速いと予想されるが、通信が増えることによってオーバーヘッドが増加する。

前述の実行時間評価は、1) プロセッサ間の結合構成、及び 2) 通信能力と演算能力の比によって変わってくる。ここでは、システム構成として、共有バス結合する場合と多段結合網を用いる場合の2通りについて並列アルゴリズムの効果が通信能力によってどのように変化するかを見る。

3.2.1 共有バス結合

共有バス結合の場合、実用的に接続できるプロセッサの数は上限が存在する。通常は100プロセッサぐらいでバスの能力を飽和させると考えられるので、この節での考察も画面分割数 ($= N_i$) は100以下とする。また、プロセッサ間の通信は共有バスを使用するので、ブロードキャストのコストも1対1通信のコストも同じである。最小値探索のコストは、ブロード

キャストをプロセッサ数回繰り返すのに一致するとする。さらに、$E_d \simeq E_u$と考え、演算時間、通信コストを下記のように書き直す。

$$\begin{aligned} C_b(n) &= C_0 \\ C_m(n) &= n \times C_0 \\ C_o(n) &= C_0 \\ E_d &= E_u \\ &= CR \times C_0 \end{aligned}$$

ここで、C_0, CRはシステムに固有の定数であり、前者は通信コストで、後者は演算時間と通信コストの比を表す。CRは100 ~ 10000の値を持つと考え、(1),(2)の実行時間を書き直すと以下のようになる。

$$\begin{aligned} (1) &= C_0 \times (CR + \frac{CR}{N_c} + N_c + 1) \\ (2) &= C_0 \times (\frac{CR}{\alpha} + 1 + \frac{CR}{N_c} + \frac{1}{\alpha}) \end{aligned}$$

(1)は(2)よりも常に大きく、共有バス構成をとる場合(例えばプロセッサ数が100以下)は、並列アルゴリズム(2)をインプリメントすることが最も効率良い。

3.2.2 多段結合網

多段結合網の場合、接続できるプロセッサ数は限界がないと考えてよい。プロセッサ間の通信は結合網の形態に依存するが、プロセッサ数の対数に比例した値でよい。ブロードキャストのコストは、ソースから全プロセッサへ木構造にしたがって通信を伝搬させるので、プロセッサ数の対数回の通信が必要である。

1対1通信のコストも、最大プロセッサ数の対数回の通信が必要であるが、平均的には、その半分でよい。また、最小値探索のコストは、ブロードキャストと同じパスで最小値を探索し、その反対のパスで最小値を全体にブロードキャストすることになるので、通信時間はブロードキャストの2倍とする。さらに、前節と同様に、$E_d \simeq E_u$と考える。以上のことより、演算時間、通信コストは、以下の様になる。

$$\begin{aligned} C_b(n) &= C_1 \times log(n) \\ C_m(n) &= 2 \times C_1 \times log(n) \\ C_o(n) &= C_1 \times \frac{log(n)}{2} \\ E_d &= E_u \\ &= CR \times C_1 \end{aligned}$$

これを用いて、(1),(2)の実行時間を書き直し、その大小について以下の式を用いる。

$$\begin{aligned} (1) &= C_1 \times (CR + \frac{CR}{N_c} + 3 \times log(N_c)) \\ (2) &= C_1 \times (\frac{CR}{\alpha} + \frac{log(N_i)}{2} + \frac{CR}{N_c} + \frac{log(N_i)}{\alpha}) \\ \frac{(1)-(2)}{C_1} &= (1 - \frac{1}{\alpha})CR + (3log(N_c) - (\frac{1}{2} + \frac{1}{\alpha})log(N_i)) \end{aligned}$$

CR, N_iが正であるかぎり$\frac{1}{\alpha} < 1$であることに注意すると、クラスタ数の2乗以下の画面分割数の場合には、任意のCRに対して(2)は、(1)よりも効率的であり、それ以上の画面分割数でも、適当に大きなCRの場合は(2)が効率的になる。

従って、多段結合網構成をとる場合も、並列アルゴリズム(2)が実用上効率が良い。

3.2.3 性能例

それぞれの並列化手法によって、どれくらいの高速化が図られるかを予測する。測定結果では、$E_d \sim E_u = 3.0sec$である。また、通信すべきワード数や既存の通信ハードウエアの能力より$C_0 \sim C_1 = 0.002sec$あたりに通信パラメータをおくのは妥当であるので、$CR = 1500$とする。多段結合網に対する実行時間を図4に示す。

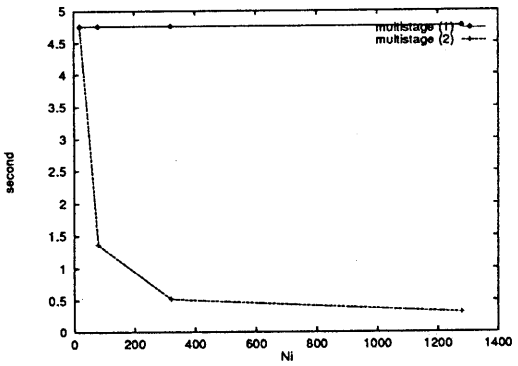


図4: $CR = 1500$の場合

図から明らかなように、パラメータ更新はクラスタ単位で並列化することの優位性が示せた。

4 実時間処理の可能性

アルゴリズムの並列化により、演算量の削減が可能となった。ここでは画素並列アーキテクチャ上で動作させることでクラスタリングの実時間処理が可能かどうかを考察する。

距離計算部分について一般的な DSP(40ns 秒で積和演算可能) でアセンブラコーディングし、予想性能評価を行なった。クラスタパラメータの更新部分に関しては、距離計算部分の結果と WS における実行評価から類推することにする。

1 ループの命令数、及びクラスタ数 40 で 25,000 画素からなる画像に対して距離計算を行なった場合、DSP において、命令数 92 を並行演算数 22 で処理し、1 フレームの処理時間 3.7sec($= 3.7\mu sec \times 40 \times 25000$) となり、一方、WS の測定推定においては約 50sec となった。

以上を用いてアルゴリズム全体の 1 フレームに要する処理時間を見積り、1.938sec という値を得た。よって逐次アルゴリズムの単一 DSP 上での処理では、毎秒 1 フレームのクラスタリングでも実時間では行なえない。そこで、将来的な半導体デバイスについて以下の仮定を行なう。

- マシンサイクルの高速化 (40nsec から 10nsec に)
- 1 チップ上に複数の DSP コアが実装可能
- 各 DSP コアには複数の演算器を実装

この仮定において、マシンサイクル 10nsec の場合に DSP コアの数を 1,4,8,16 まで変化させ、クラスタ数 40 で 25,000 画素からなる画像に対して距離計算を行なった場合、1 秒間に 30 フレーム (1 フレームの処理が 0.0606 秒以下) のクラスタリングを行なう最小の組合せは、「8 並列の 2 乗算器 2 加減算器」もしくは、「16 並列の 1 乗算器 1 加減算器」となった。

5 まとめ

本論文では、画像クラスタリングアルゴリズムの高速化のための並列化手法について述べた後、仮想的な DSP での予想評価から実時間処理の可能性を検討した。

クラスタリングは距離 (尤度) 計算とクラスタパラメータの更新の 2 つの部分からなるが、距離計算は画素 (分割画面) 単位で、パラメータ更新はクラスタ単位で並列化するのが最も効率の良いことが分かった。

また、現状の DSP 単体でのクラスタリングの実時間処理は困難であるが、複数 DSP を用いたシステムか、もしくはマシンサイクルが 4 倍の DSP コアを 16 並列以上実装できれば実時間処理の可能性が高いことが分かった。

謝辞本研究に日ごろから御助言頂いている松下電器 (株) 栄藤主任研究員に感謝します。

参考文献

[1] 栄藤稔, 白井良明,"色, 位置, 輝度こう配に基づく領域分割による 2 次元動き推定", 信学論 (D-II),J76-D-II,11,pp.2333-2340(1993-11).

[2] Hanson A. and Riseman E.,"Segmentation of Natural Scenes",in Computer Vision Systems,pp. 129-163,Academic Press(1978).

[3] Schunk B. ,"Image Flow Segmentation and Estimation by Constraint Line Clustering", IEEE Trans. Pattern Anal. & Mach. Intel.,11,10, pp.1010-1027(1989).